Biblioteca de códigos

AWS Exemplos de código SDK

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

AWS Exemplos de código SDK: Biblioteca de códigos

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

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

..... xxxi
O que é a biblioteca de exemplos de código? 1
Trabalhando com AWS SDKs 1
Exemplos de código por serviço 3
ACM 8
Conceitos básicos 9
API Gateway 113
Conceitos básicos 114
Cenários 163
AWS contribuições da comunidade 189
A API de gerenciamento do API Gateway 191
Conceitos básicos 191
Application Auto Scaling 194
Conceitos básicos 195
Application Recovery Controller 232
Conceitos básicos 233
Athena 243
Cenários 244
Audit Manager 260
Cenários 260
Aurora 273
Conceitos básicos 275
Cenários 665
ajuste de escala automático 671
Conceitos básicos 673
Cenários 1014
AWS Batch 1182
Conceitos básicos 1183
Cenários 1249
Amazon Bedrock 1262
Conceitos básicos 1263
Cenários 1290
Amazon Bedrock Runtime 1291
Conceitos básicos 1294

Cenários ... 1302
Amazon Nova ... 1367
Amazon Nova Canvas ... 1441
Amazon Nova Reel ... 1452
Gerador de Imagens do Amazon Titan ... 1459
Amazon Titan Text ... 1466
Incorporações de texto Amazon Titan ... 1467
Claude da Anthropic ... 1472
Command da Cohere ... 1527
DeepSeek ... 1554
Llama da Meta ... 1556
Mistral AI ... 1576
Stable Image Core ... 1593
Amazon Bedrock Agents ... 1605
Conceitos básicos ... 1607
Cenários ... 1672
Amazon Bedrock Agents Runtime ... 1706
Conceitos básicos ... 1707
Cenários ... 1725
AWS Budgets ... 1743
Conceitos básicos ... 1744
AWS Cloud Map ... 1747
Cenários ... 1748
AWS Cloud9 ... 1790
Conceitos básicos ... 1791
CloudFormation ... 1812
Conceitos básicos ... 1814
Cenários ... 1850
CloudFront ... 1874
Conceitos básicos ... 1876
Cenários ... 1945
CloudFront Exemplos de funções ... 1985
CloudTrail ... 2002
Conceitos básicos ... 2003
CloudWatch ... 2026
Conceitos básicos ... 2028

Cenários ... 2293
CloudWatch Eventos ... 2340
Conceitos básicos ... 2341
CloudWatch Registros ... 2354
Conceitos básicos ... 2355
Cenários ... 2422
CodeBuild ... 2490
Conceitos básicos ... 2491
CodeCommit ... 2507
Conceitos básicos ... 2508
CodeDeploy ... 2526
Conceitos básicos ... 2528
CodePipeline ... 2590
Conceitos básicos ... 2591
Identidade do Amazon Cognito ... 2648
Conceitos básicos ... 2649
Cenários ... 2674
Provedor de identidades do Amazon Cognito ... 2676
Conceitos básicos ... 2678
Cenários ... 2839
Amazon Cognito Sync ... 3016
Conceitos básicos ... 3017
Amazon Comprehend ... 3019
Conceitos básicos ... 3020
Cenários ... 3104
AWS Config ... 3137
Conceitos básicos ... 3138
Cenários ... 3176
Connect Customer ... 3188
Cenários ... 3188
AWS Control Tower ... 3200
Conceitos básicos ... 3201
Firehose ... 3365
Conceitos básicos ... 3366
Cenários ... 3378
Device Farm ... 3391

Conceitos básicos 3392
Cenários 3395
Direct Connect 3406
Cenários 3406
Directory Service 3416
Conceitos básicos 3416
AWS DMS 3421
Conceitos básicos 3422
Amazon DocumentDB 3426
Cenários 3426
Exemplos sem servidor 3441
DynamoDB 3453
Conceitos básicos 3457
Cenários 4033
Exemplos sem servidor 4702
AWS contribuições da comunidade 4722
Amazon EBS 4724
Conceitos básicos 4725
Amazon EC2 4728
Conceitos básicos 4736
Cenários 5906
Amazon ECR 6356
Conceitos básicos 6357
Cenários 6475
Amazon ECS 6483
Conceitos básicos 6484
Cenários 6567
Amazon EFS 6625
Conceitos básicos 6625
Amazon EKS 6641
Conceitos básicos 6643
Cenários 6707
Elastic Beanstalk 6719
Conceitos básicos 6720
Elastic Load Balancing Versão 1 6730
Conceitos básicos 6731

Elastic Load Balancing versão 2 6797
Conceitos básicos 6799
Cenários 6941
ElastiCache 7118
Cenários 7118
MediaStore 7126
Conceitos básicos 7127
Amazon EMR 7148
Conceitos básicos 7149
Cenários 7173
AWS Entity Resolution 7187
Conceitos básicos 7188
OpenSearch Serviço 7269
Conceitos básicos 7270
Cenários 7306
EventBridge 7321
Conceitos básicos 7322
Cenários 7442
EventBridge Agendador 7465
Conceitos básicos 7466
Cenários 7494
AWS FIS 7544
Cenários 7545
Previsão 7560
Conceitos básicos 7561
Amazon Glacier 7576
Conceitos básicos 7577
Cenários 7644
AWS Glue 7655
Conceitos básicos 7657
Cenários 7960
AWS Health 7968
Conceitos básicos 7969
HealthImaging 7976
Conceitos básicos 7978
Cenários 8137

HealthLake 8212
Conceitos básicos 8213
IAM 8259
Conceitos básicos 8266
Cenários 9055
Amazon Inspector 9544
Conceitos básicos 9546
AWS IoT 9598
Conceitos básicos 9599
Cenários 9829
AWS IoT data 9860
Conceitos básicos 9860
AWS IoT SiteWise 9876
Conceitos básicos 9878
Amazon Keyspaces 10010
Conceitos básicos 10012
Kinesis 10131
Conceitos básicos 10132
Exemplos sem servidor 10194
AWS KMS 10218
Conceitos básicos 10219
Cenários 10468
Lambda 10470
Conceitos básicos 10473
Cenários 10755
Exemplos sem servidor 10902
AWS contribuições da comunidade 11027
Amazon Lex 11029
Cenários 11030
Lightsail 11031
Cenários 11032
Amazon Location 11040
Conceitos básicos 11041
Location Service Places 11154
Conceitos básicos 11155
Amazon Managed Grafana 11161

Cenários 11161
Managed Service for Apache Flink 11169
Conceitos básicos 11171
Gerador de dados 11186
AWS Marketplace API do catálogo 11200
Produtos de AMI 11204
Ofertas de parceiros de canal 11264
Produtos de contêiner 11304
Entidades 11318
Ofertas 11329
Produtos 11467
Autorização de revenda 11479
Produtos de SaaS do 11578
Serviços públicos 11635
AWS Marketplace API do contrato 11642
Contratos 11644
MediaConnect 11904
Cenários 11904
MediaConvert 11913
Conceitos básicos 11914
MediaLive 11962
Conceitos básicos 11962
MediaPackage 11964
Conceitos básicos 11964
Migration Hub 11970
Conceitos básicos 11971
Amazon MSK 11985
Cenários 11985
Exemplos sem servidor 12012
Neptune 12021
Conceitos básicos 12023
Cenários 12114
Organizações 12125
Conceitos básicos 12126
Cenários 12167
Central de Parceiros 12174

Conceitos básicos 12176
Cenários 12241
AWS Payment Cryptography 12247
Cenários 12247
Amazon Personalize 12252
Conceitos básicos 12254
Eventos do Amazon Personalize 12303
Conceitos básicos 12304
Runtime do Amazon Personalize 12311
Conceitos básicos 12312
Amazon Pinpoint 12321
Conceitos básicos 12322
Cenários 12413
API de SMS e voz do Amazon Pinpoint 12425
Conceitos básicos 12426
Amazon Polly 12445
Conceitos básicos 12446
Cenários 12497
AWS Price List 12512
Conceitos básicos 12513
Amazon RDS 12523
Conceitos básicos 12524
Cenários 12794
Exemplos sem servidor 12806
Serviços de dados do Amazon RDS 12825
Conceitos básicos 12826
Cenários 12829
banco de dados de origem 12833
Conceitos básicos 12835
Cenários 12965
Amazon Rekognition 12994
Conceitos básicos 12996
Cenários 13171
Resource Groups (Grupos de recursos) 13250
Conceitos básicos 13251
Resource Groups Tagging API 13278

Conceitos básicos 13278
Route 53 13292
Conceitos básicos 13292
Registro de domínios do Route 53 13320
Conceitos básicos 13322
Amazon S3 13403
Conceitos básicos 13408
Cenários 14099
Exemplos sem servidor 14551
Amazon S3 Control 14563
Conceitos básicos 14564
Buckets de diretório do S3 14644
Conceitos básicos 14645
Cenários 14768
SageMaker IA 14771
Conceitos básicos 14773
Cenários 14819
Secrets Manager 14935
Conceitos básicos 14936
Cenários 14954
Amazon SES 15003
Conceitos básicos 15005
Cenários 15136
API v2 do Amazon SES 15183
Conceitos básicos 15185
Cenários 15267
Amazon SNS 15324
Conceitos básicos 15326
Cenários 15521
Exemplos sem servidor 15743
Amazon SQS 15753
Conceitos básicos 15755
Cenários 15942
Exemplos sem servidor 16161
Step Functions 16180
Conceitos básicos 16181

Cenários 16293
AWS STS 16316
Conceitos básicos 16318
Cenários 16350
Suporte 16599
Conceitos básicos 16601
Cenários 16741
Systems Manager 16750
Conceitos básicos 16754
Cenários 17154
Amazon Textract 17197
Conceitos básicos 17198
Cenários 17234
Amazon Transcribe 17253
Conceitos básicos 17254
Cenários 17325
Streaming do Amazon Transcribe 17347
Conceitos básicos 17348
Cenários 17363
Amazon Translate 17383
Conceitos básicos 17384
Cenários 17402
Estrutura de PVC 17415
Cenários 17415
AWS WAFV2 17426
Conceitos básicos 17426
Cenários 17431
WorkSpaces 17443
Conceitos básicos 17445
Cenários 17478
X-Ray 17491
Conceitos básicos 17491
Exemplos de código por SDK 17519
SDK para .NET 17520
ACM 17522
API Gateway 17527

Aurora 17528
ajuste de escala automático 17530
Amazon Bedrock Runtime 17582
CloudWatch Registros 17623
Provedor de identidades do Amazon Cognito 17638
Amazon Comprehend 17662
Amazon DocumentDB 17674
DynamoDB 17678
Amazon EC2 17745
Amazon ECS 17869
Elastic Load Balancing versão 2 17880
EventBridge 17938
EventBridge Agendador 17978
Amazon Glacier 18008
AWS Glue 18018
IAM 18050
Amazon Keyspaces 18147
Kinesis 18175
AWS KMS 18193
Lambda 18205
MediaConvert 18256
Amazon MSK 18267
Organizações 18268
Central de Parceiros 18287
Amazon Pinpoint 18294
Amazon Polly 18300
Amazon RDS 18313
Serviços de dados do Amazon RDS 18352
Amazon Rekognition 18353
Registro de domínios do Route 53 18385
Amazon S3 18411
SageMaker IA 18554
Secrets Manager 18589
Amazon SES 18592
API v2 do Amazon SES 18606
Amazon SNS 18645

Amazon SQS 18691
Funções de etapa 18735
AWS STS 18762
Suporte 18765
Amazon Textract 18793
Amazon Transcribe 18794
Amazon Translate 18806
SDK para .NET (v4) 18819
Aurora 18820
ajuste de escala automático 18862
Amazon Bedrock 18898
Amazon Bedrock Runtime 18902
CloudFormation 18921
CloudWatch 18925
CloudWatch Registros 18983
Provedor de identidades do Amazon Cognito 19008
AWS Control Tower 19009
DynamoDB 19049
Amazon EC2 19090
Amazon ECS 19092
AWS IoT 19094
AWS IoT data 19153
banco de dados de origem 19156
Amazon S3 19187
AWS CLI com script Bash 19223
API Gateway 19225
Athena 19234
AWS Batch 19249
AWS Cloud Map 19262
CloudFormation 19304
CloudFront 19326
CloudWatch 19349
CloudWatch Registros 19384
Provedor de identidades do Amazon Cognito 19413
AWS Config 19423
Connect Customer 19435

Direct Connect 19446
Amazon DocumentDB 19456
DynamoDB 19470
Amazon EC2 19576
Amazon ECR 19960
Amazon ECS 19967
Amazon EKS 20016
Elastic Load Balancing versão 2 20029
ElastiCache 20038
Amazon EMR 20045
OpenSearch Serviço 20056
AWS FIS 20070
AWS Glue 20085
HealthImaging 20093
IAM 20102
AWS IoT 20429
AWS KMS 20460
Lambda 20462
Lightsail 20501
Amazon Managed Grafana 20509
MediaConnect 20517
Amazon MSK 20526
Neptune 20552
AWS Payment Cryptography 20562
Amazon Pinpoint 20567
Amazon Polly 20579
Amazon RDS 20586
banco de dados de origem 20594
Amazon S3 20621
SageMaker IA 20714
Secrets Manager 20728
Amazon SES 20775
Amazon SNS 20783
Funções de etapa 20821
AWS STS 20842
Suporte 21074

Systems Manager 21082
Amazon Textract 21124
Malha de PVC 21132
AWS WAFV2 21142
WorkSpaces 21155
SDK para C++ 21167
ACM 21169
API Gateway 21199
Aurora 21200
ajuste de escala automático 21245
CloudTrail 21276
CloudWatch 21281
CloudWatch Registros 21291
CodeBuild 21296
Provedor de identidades do Amazon Cognito 21301
DynamoDB 21324
Amazon EC2 21402
EventBridge 21438
AWS Glue 21442
HealthImaging 21480
IAM 21515
AWS IoT 21556
AWS IoT data 21599
Lambda 21602
MediaConvert 21627
Amazon RDS 21635
Serviços de dados do Amazon RDS 21673
Amazon Rekognition 21674
Amazon S3 21679
Secrets Manager 21762
Amazon SES 21764
Amazon SNS 21787
Amazon SQS 21829
AWS STS 21866
Streaming do Amazon Transcribe 21867
SDK para Go V2 21875

API Gateway 21876
Aurora 21877
Amazon Bedrock 21917
Amazon Bedrock Runtime 21921
CloudFormation 21929
CloudWatch Registros 21930
Provedor de identidades do Amazon Cognito 21933
Amazon DocumentDB 22015
DynamoDB 22016
IAM 22089
Kinesis 22151
Lambda 22154
Amazon MSK 22266
Central de parceiro 22267
Amazon RDS 22270
banco de dados de origem 22306
Amazon S3 22325
Amazon SNS 22415
Amazon SQS 22443
SDK para JavaScript (v2) 22475
CloudWatch 22476
CloudWatch Eventos 22485
CloudWatch Registros 22488
DynamoDB 22491
AWS Entity Resolution 22507
EventBridge 22525
Amazon Glacier 22528
IAM 22532
Lambda 22553
Amazon Pinpoint 22554
API de SMS e voz do Amazon Pinpoint 22559
Amazon SNS 22562
Amazon SQS 22563
AWS STS 22574
SDK para JavaScript (v3) 22575
API Gateway 22578

Aurora 22580
ajuste de escala automático 22581
Amazon Bedrock 22623
Amazon Bedrock Runtime 22628
Amazon Bedrock Agents 22671
Amazon Bedrock Agents Runtime 22685
CloudWatch 22690
CloudWatch Eventos 22699
CloudWatch Registros 22704
CodeBuild 22720
Identidade do Amazon Cognito 22723
Provedor de identidades do Amazon Cognito 22724
Amazon Comprehend 22765
Amazon DocumentDB 22771
DynamoDB 22772
Amazon EC2 22966
Elastic Load Balancing versão 2 23063
AWS Entity Resolution 23113
EventBridge 23126
Amazon Glacier 23132
AWS Glue 23134
HealthImaging 23161
IAM 23223
AWS IoT SiteWise 23317
Kinesis 23349
Lambda 23356
Amazon Lex 23412
Amazon Location 23413
AWS Marketplace API de contrato 23445
Amazon MSK 23489
Amazon Personalize 23491
Eventos do Amazon Personalize 23508
Runtime do Amazon Personalize 23512
Amazon Pinpoint 23516
Amazon Polly 23521
Amazon RDS 23526

Serviços de dados do Amazon RDS 23530
banco de dados de origem 23531
Amazon Rekognition 23537
Amazon S3 23539
SageMaker IA 23673
Secrets Manager 23711
Amazon SES 23713
Amazon SNS 23739
Amazon SQS 23780
Funções de etapa 23812
AWS STS 23813
Suporte 23815
Systems Manager 23833
Amazon Textract 23861
Amazon Transcribe 23866
Amazon Translate 23875
SDK para Java 2.x 23881
ACM 23884
API Gateway 23903
Application Auto Scaling 23909
Application Recovery Controller 23918
Aurora 23920
ajuste de escala automático 23956
AWS Batch 24020
Amazon Bedrock 24070
Amazon Bedrock Runtime 24075
CloudFront 24157
CloudWatch 24188
CloudWatch Eventos 24265
CloudWatch Registros 24271
Identidade do Amazon Cognito 24289
Provedor de identidades do Amazon Cognito 24297
Amazon Comprehend 24324
AWS Control Tower 24337
Firehose 24399
Amazon DocumentDB 24408

DynamoDB 24410
Amazon EC2 24672
Amazon ECR 24771
Amazon ECS 24810
Elastic Load Balancing versão 2 24825
MediaStore 24870
AWS Entity Resolution 24885
OpenSearch Serviço 24934
EventBridge 24964
EventBridge Agendador 24999
Previsão 25024
Amazon Glacier 25037
AWS Glue 25053
HealthImaging 25085
IAM 25116
Amazon Inspector 25201
AWS IoT 25249
AWS IoT data 25289
AWS IoT SiteWise 25292
Amazon Keyspaces 25331
Kinesis 25358
AWS KMS 25371
Lambda 25427
Amazon Lex 25464
Amazon Location 25465
Location Service Places 25513
AWS Marketplace API do catálogo 25518
AWS Marketplace API de contrato 25712
MediaConvert 25822
Migration Hub 25843
Amazon MSK 25856
Neptune 25858
Central de Parceiros 25901
Amazon Personalize 25936
Eventos do Amazon Personalize 25966
Runtime do Amazon Personalize 25969

Amazon Pinpoint 25974
API de SMS e voz do Amazon Pinpoint 26018
Amazon Polly 26022
Amazon RDS 26029
Serviços de dados do Amazon RDS 26072
banco de dados de origem 26073
Amazon Rekognition 26112
Registro de domínios do Route 53 26184
Amazon S3 26207
Amazon S3 Control 26398
Buckets de diretório do S3 26441
SageMaker IA 26524
Secrets Manager 26553
Amazon SES 26555
API v2 do Amazon SES 26571
Amazon SNS 26592
Amazon SQS 26661
Funções de etapa 26761
AWS STS 26785
Suporte 26789
Systems Manager 26812
Amazon Textract 26855
Amazon Transcribe 26867
Streaming do Amazon Transcribe 26878
Amazon Translate 26897
X-Ray 26899
SDK para Kotlin 26912
API Gateway 26914
Aurora 26915
ajuste de escala automático 26943
Amazon Bedrock 26961
Amazon Bedrock Runtime 26962
CloudWatch 26966
CloudWatch Registros 27007
Provedor de identidades do Amazon Cognito 27010
Amazon Comprehend 27026

DynamoDB 27027
Amazon EC2 27056
Amazon ECR 27086
OpenSearch Serviço 27116
EventBridge 27120
AWS Glue 27151
IAM 27162
AWS IoT 27182
AWS IoT data 27206
Amazon Keyspaces 27208
AWS KMS 27233
Lambda 27243
Amazon Location 27252
MediaConvert 27284
Amazon Pinpoint 27289
Amazon RDS 27299
Serviços de dados do Amazon RDS 27317
banco de dados de origem 27318
Amazon Rekognition 27323
Registro de domínios do Route 53 27341
Amazon S3 27361
SageMaker IA 27383
Secrets Manager 27408
Amazon SES 27409
Amazon SNS 27412
Amazon SQS 27438
Funções de etapa 27461
Suporte 27483
Amazon Translate 27501
X-Ray 27502
SDK para PHP 27508
API Gateway 27509
Aurora 27516
ajuste de escala automático 27517
Amazon Bedrock 27533
Amazon Bedrock Runtime 27534

Amazon DocumentDB 27538
DynamoDB 27539
Amazon EC2 27572
AWS Glue 27578
IAM 27598
Kinesis 27615
AWS KMS 27619
Lambda 27655
Amazon MSK 27685
Amazon RDS 27687
Serviços de dados do Amazon RDS 27695
Amazon Rekognition 27696
Amazon S3 27697
Buckets de diretório do S3 27720
Amazon SES 27737
Amazon SNS 27738
Amazon SQS 27758
Ferramentas para PowerShell V5 27762
ACM 27764
Application Auto Scaling 27769
WorkSpaces Aplicações 27776
Aurora 27803
ajuste de escala automático 27804
AWS Budgets 27840
AWS Cloud9 27842
CloudFormation 27848
CloudFront 27861
CloudTrail 27869
CloudWatch 27874
CodeCommit 27878
CodeDeploy 27884
CodePipeline 27903
Identidade do Amazon Cognito 27921
AWS Config 27925
Device Farm 27944
Directory Service 27945

AWS DMS 27971
DynamoDB 27972
Amazon EC2 27987
Amazon ECR 28120
Amazon ECS 28122
Amazon EFS 28128
Amazon EKS 28135
Elastic Load Balancing Versão 1 28148
Elastic Load Balancing versão 2 28168
Amazon FSx 28192
Amazon Glacier 28200
AWS Glue 28204
AWS Health 28206
IAM 28207
Kinesis 28282
Lambda 28286
Amazon ML 28299
Macie 28305
AWS Price List 28306
Resource Groups (Grupos de recursos) 28309
Resource Groups Tagging API 28317
Route 53 28322
Amazon S3 28337
CSPM do Security Hub 28377
Amazon SES 28379
API v2 do Amazon SES 28380
Amazon SNS 28381
Amazon SQS 28382
AWS STS 28395
Suporte 28399
Systems Manager 28405
Amazon Translate 28480
AWS WAFV2 28481
WorkSpaces 28482
SDK for Python 28497
DynamoDB 28498

SDK para Python (Boto3) 28499
ACM 28502
API Gateway 28525
Application Recovery Controller 28557
Audit Manager 28560
Aurora 28573
ajuste de escala automático 28619
Amazon Bedrock 28696
Amazon Bedrock Runtime 28702
Amazon Bedrock Agents 28771
Amazon Bedrock Agents Runtime 28848
CloudFormation 28873
CloudFront 28874
CloudWatch 28879
CloudWatch Registros 28899
Identidade do Amazon Cognito 28914
Provedor de identidades do Amazon Cognito 28915
Amazon Comprehend 28957
AWS Config 29000
AWS Control Tower 29004
Firehose 29044
Device Farm 29055
Amazon DocumentDB 29065
DynamoDB 29066
Amazon EC2 29268
Amazon ECR 29410
Elastic Load Balancing versão 2 29439
Amazon EMR 29496
EventBridge 29509
EventBridge Agendador 29510
Amazon Glacier 29529
AWS Glue 29555
HealthImaging 29592
HealthLake 29645
IAM 29664
AWS IoT 29790

AWS IoT data 29827
AWS IoT SiteWise 29830
Amazon Keyspaces 29871
Kinesis 29903
AWS KMS 29912
Lambda 29967
Serviço Gerenciado para Apache Flink 30005
AWS Marketplace API do catálogo 30032
AWS Marketplace API de contrato 30270
Amazon MSK 30372
Neptune 30373
Organizações 30418
Central de Parceiros 30424
Amazon Pinpoint 30450
API de SMS e voz do Amazon Pinpoint 30459
Amazon Polly 30462
Amazon RDS 30473
Serviços de dados do Amazon RDS 30513
banco de dados de origem 30514
Amazon Rekognition 30537
Amazon S3 30593
Amazon S3 Control 30684
Buckets de diretório do S3 30706
Secrets Manager 30728
Amazon SES 30732
API v2 do Amazon SES 30787
Amazon SNS 30828
Amazon SQS 30878
Funções de etapa 30933
AWS STS 30964
Suporte 30983
Systems Manager 31014
Amazon Textract 31049
Amazon Transcribe 31059
SDK para Ruby 31081
Aurora 31082

ajuste de escala automático 31083
CloudTrail 31085
CloudWatch 31089
Provedor de identidades do Amazon Cognito 31102
Amazon Comprehend 31104
Amazon DocumentDB 31105
DynamoDB 31106
Amazon EC2 31135
Pé de feijão elástico 31169
EventBridge 31175
AWS Glue 31193
IAM 31220
Kinesis 31278
AWS KMS 31281
Lambda 31285
Amazon MSK 31315
Amazon Polly 31316
Amazon RDS 31321
Amazon S3 31328
Amazon SES 31360
API v2 do Amazon SES 31366
Amazon SNS 31367
Amazon SQS 31377
AWS STS 31391
Amazon Textract 31392
Amazon Translate 31393
SDK para Rust 31395
API Gateway 31397
A API de gerenciamento do API Gateway 31399
Application Auto Scaling 31401
Aurora 31402
ajuste de escala automático 31550
Amazon Bedrock Runtime 31584
Amazon Bedrock Agents Runtime 31609
Provedor de identidades do Amazon Cognito 31614
Amazon Cognito Sync 31615

Firehose 31617
Amazon DocumentDB 31618
DynamoDB 31620
Amazon EBS 31638
Amazon EC2 31641
Amazon ECR 31703
Amazon ECS 31705
Amazon EKS 31708
AWS Glue 31710
IAM 31725
AWS IoT 31752
Kinesis 31754
AWS KMS 31762
Lambda 31771
MediaLive 31821
MediaPackage 31823
Amazon MSK 31825
Amazon Polly 31827
Amazon RDS 31832
Serviços de dados do Amazon RDS 31835
Amazon Rekognition 31837
Route 53 31839
Amazon S3 31840
SageMaker IA 31891
Secrets Manager 31893
API v2 do Amazon SES 31894
Amazon SNS 31911
Amazon SQS 31917
AWS STS 31922
Systems Manager 31923
Amazon Transcribe 31926
SDK para SAP ABAP 31927
ACM 31930
API Gateway 31938
Application Recovery Controller 31947
Aurora 31952

ajuste de escala automático 31958
Amazon Bedrock Runtime 31967
Amazon Bedrock Agents Runtime 31970
CloudFront 31972
CloudWatch 31974
CloudWatch Registros 31984
Provedor de identidades do Amazon Cognito 31987
Amazon Comprehend 31993
AWS Config 32006
AWS Control Tower 32009
Firehose 32020
DynamoDB 32022
Amazon EC2 32037
Amazon ECR 32060
Amazon EMR 32066
EventBridge Agendador 32072
AWS Glue 32077
HealthImaging 32091
HealthLake 32108
IAM 32122
AWS IoT 32148
AWS IoT data 32159
AWS IoT SiteWise 32161
Amazon Keyspaces 32170
Kinesis 32178
AWS KMS 32188
Lambda 32209
Organizações 32223
Amazon Pinpoint 32228
API de SMS e voz do Amazon Pinpoint 32233
Amazon Polly 32243
Amazon RDS 32251
banco de dados de origem 32257
Amazon Rekognition 32264
Amazon S3 32279
Amazon S3 Control 32305

SageMaker IA 32314
Secrets Manager 32332
Amazon SES 32334
API v2 do Amazon SES 32351
Amazon SNS 32362
Amazon SQS 32373
Funções de etapa 32383
Systems Manager 32391
Amazon Textract 32401
Amazon Transcribe 32412
Amazon Translate 32420
SDK para Swift 32428
Amazon Bedrock 32429
Amazon Bedrock Runtime 32435
Identidade do Amazon Cognito 32451
Provedor de identidades do Amazon Cognito 32455
DynamoDB 32484
Amazon EC2 32522
AWS Glue 32576
IAM 32617
Lambda 32639
Amazon RDS 32669
Amazon S3 32706
Amazon SNS 32731
Amazon SQS 32758
AWS STS 32786
Streaming do Amazon Transcribe 32787

Há mais exemplos de AWS SDK disponíveis no repositório AWS Doc SDK Examples GitHub .

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.

O que é a biblioteca de exemplos de código?

A biblioteca de exemplos de código é uma coleção de exemplos de código que mostram como usar kits de desenvolvimento de AWS software (SDKs) com AWS.

Os exemplos são organizados por AWS service (Serviço da AWS) e por AWS SDK. Os mesmos exemplos podem ser encontrados em cada seção.

- Exemplos de código por serviço— Uma lista Serviços da AWS que contém exemplos de como usar cada serviço com AWS SDKs. Use esta seção se souber qual serviço deseja usar.
- Exemplos de código por SDK— Uma lista AWS SDKs contém exemplos de como usar Serviços da AWS com cada SDK. Use esta seção se souber qual SDK deseja usar.

Em cada seção, os exemplos são divididos nas seguintes categorias:

- As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.
- Ações são trechos de código que mostram como chamar funções de serviço individuais.
- Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.
- Exemplos de categorias adicionais são definidos para alguns serviços. Esses exemplos de código mostram algo especializado sobre um serviço ou uma integração.

Todos os exemplos dessa biblioteca também podem ser encontrados no GitHub repositório de exemplos de AWS código. O GitHub repositório também contém instruções sobre como configurar, executar e testar os exemplos.

Usando Serviços da AWS com um AWS SDK

AWS kits de desenvolvimento de software (SDKs) estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ exemplos de código
AWS CLI	AWS CLI exemplos de código
AWS SDK para Go	AWS SDK para Go exemplos de código
AWS SDK para Java	AWS SDK para Java exemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript exemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin exemplos de código
AWS SDK para .NET	AWS SDK para .NET exemplos de código
AWS SDK para PHP	AWS SDK para PHP exemplos de código
Ferramentas da AWS para PowerShell	Ferramentas da AWS para PowerShell exemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) exemplos de código
AWS SDK para Ruby	AWS SDK para Ruby exemplos de código
AWS SDK para Rust	AWS SDK para Rust exemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP exemplos de código
AWS SDK para Swift	AWS SDK para Swift exemplos de código

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na parte inferior desta página.

Exemplos de código por serviço usando AWS SDKs

Os exemplos de código a seguir mostram como usar Serviços da AWS com um kit de desenvolvimento de AWS software (SDK). Esta seção é organizada por AWS service (Serviço da AWS).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos de código para ACM usando AWS SDKs
- Exemplos de código para o API Gateway usando AWS SDKs
- Exemplos de código para API Gateway Management API usando AWS SDKs
- Exemplos de código para Application Auto Scaling usando AWS SDKs
- Exemplos de código para o Application Recovery Controller usando AWS SDKs
- Exemplos de código para Athena usando AWS SDKs
- Exemplos de código para o Audit Manager usando AWS SDKs
- Exemplos de código para Aurora usando AWS SDKs

- Exemplos de código para Auto Scaling usando AWS SDKs
- Exemplos de código para AWS Batch usar AWS SDKs
- Exemplos de código para o Amazon Bedrock usando AWS SDKs
- Exemplos de código para o Amazon Bedrock Runtime usando AWS SDKs
- Exemplos de código para Amazon Bedrock Agents usando AWS SDKs
- Exemplos de código para Amazon Bedrock Agents Runtime usando AWS SDKs
- Exemplos de código para AWS Budgets usar AWS SDKs
- Exemplos de código para AWS Cloud Map usar AWS SDKs
- Exemplos de código para AWS Cloud9 usar AWS SDKs
- Exemplos de código para CloudFormation usar AWS SDKs
- Exemplos de código para CloudFront usar AWS SDKs
- Exemplos de código para CloudTrail usar AWS SDKs
- Exemplos de código para CloudWatch usar AWS SDKs
- Exemplos de código para CloudWatch eventos usando AWS SDKs
- Exemplos de código para CloudWatch registros usando AWS SDKs
- Exemplos de código para CodeBuild usar AWS SDKs
- Exemplos de código para CodeCommit usar AWS SDKs
- Exemplos de código para CodeDeploy usar AWS SDKs
- Exemplos de código para CodePipeline usar AWS SDKs
- Exemplos de código para o Amazon Cognito Identity usando AWS SDKs
- Exemplos de código para o Amazon Cognito Identity Provider usando AWS SDKs
- Exemplos de código para o Amazon Cognito Sync usando AWS SDKs
- Exemplos de código para o Amazon Comprehend usando AWS SDKs
- Exemplos de código para AWS Config usar AWS SDKs
- Exemplos de código para Connect Customer usando AWS SDKs
- Exemplos de código para AWS Control Tower usar AWS SDKs
- Exemplos de código para Firehose usando AWS SDKs
- Exemplos de código para Device Farm usando AWS SDKs
- Exemplos de código para Direct Connect usar AWS SDKs

- Exemplos de código para Directory Service usar AWS SDKs
- Exemplos de código para AWS DMS usar AWS SDKs
- Exemplos de código para o Amazon DocumentDB usando AWS SDKs
- Exemplos de código para o DynamoDB usando AWS SDKs
- Exemplos de código para o Amazon EBS usando AWS SDKs
- Exemplos de código para o Amazon EC2 usando AWS SDKs
- Exemplos de código para o Amazon ECR usando AWS SDKs
- Exemplos de código para o Amazon ECS usando AWS SDKs
- Exemplos de código para o Amazon EFS usando AWS SDKs
- Exemplos de código para o Amazon EKS usando AWS SDKs
- Exemplos de código para o Elastic Beanstalk usando AWS SDKs
- Exemplos de código para o Elastic Load Balancing - Versão 1 usando AWS SDKs
- Exemplos de código para o Elastic Load Balancing - Versão 2 usando AWS SDKs
- Exemplos de código para ElastiCache usar AWS SDKs
- Exemplos de código para MediaStore usar AWS SDKs
- Exemplos de código para o Amazon EMR usando AWS SDKs
- Exemplos de código para AWS Entity Resolution usar AWS SDKs
- Exemplos de código para o OpenSearch serviço usando AWS SDKs
- Exemplos de código para EventBridge usar AWS SDKs
- Exemplos de código para o EventBridge Scheduler usando AWS SDKs
- Exemplos de código para AWS FIS usar AWS SDKs
- Exemplos de código para Forecast usando AWS SDKs
- Exemplos de código para o Amazon Glacier usando AWS SDKs
- Exemplos de código para AWS Glue usar AWS SDKs
- Exemplos de código para AWS Health usar AWS SDKs
- Exemplos de código para HealthImaging usar AWS SDKs
- Exemplos de código para HealthLake usar AWS SDKs
- Exemplos de código para o IAM usando AWS SDKs
- Exemplos de código para o Amazon Inspector usando AWS SDKs
- Exemplos de código para AWS IoT usar AWS SDKs

- Exemplos de código para AWS IoT data usar AWS SDKs
- Exemplos de código para AWS IoT SiteWise usar AWS SDKs
- Exemplos de código para Amazon Keyspaces usando AWS SDKs
- Exemplos de código para Kinesis usando AWS SDKs
- Exemplos de código para AWS KMS usar AWS SDKs
- Exemplos de código para Lambda usando AWS SDKs
- Exemplos de código para o Amazon Lex usando AWS SDKs
- Exemplos de código para o Lightsail usando AWS SDKs
- Exemplos de código para Amazon Location usando AWS SDKs
- Exemplos de código para Location Service Places usando AWS SDKs
- Exemplos de código para Amazon Managed Grafana usando AWS SDKs
- Exemplos de código para Managed Service for Apache Flink usando AWS SDKs
- Exemplos de código para AWS Marketplace API de catálogo usando AWS SDKs
- Exemplos de código para AWS Marketplace API de contrato usando AWS SDKs
- Exemplos de código para MediaConnect usar AWS SDKs
- Exemplos de código para MediaConvert usar AWS SDKs
- Exemplos de código para MediaLive usar AWS SDKs
- Exemplos de código para MediaPackage usar AWS SDKs
- Exemplos de código para o Migration Hub usando AWS SDKs
- Exemplos de código para o Amazon MSK usando AWS SDKs
- Exemplos de código para Neptune usando AWS SDKs
- Exemplos de código para organizações que usam AWS SDKs
- Exemplos de código para o Partner Central usando AWS SDKs
- Exemplos de código para AWS Payment Cryptography usar AWS SDKs
- Exemplos de código para o Amazon Personalize usando AWS SDKs
- Exemplos de código para Amazon Personalize Events usando AWS SDKs
- Exemplos de código para o Amazon Personalize Runtime usando AWS SDKs
- Exemplos de código para o Amazon Pinpoint usando AWS SDKs
- Exemplos de código para Amazon Pinpoint SMS e Voice API usando AWS SDKs
- Exemplos de código para o Amazon Polly usando AWS SDKs

- Exemplos de código para AWS Price List usar AWS SDKs
- Exemplos de código para o Amazon RDS usando AWS SDKs
- Exemplos de código para o Amazon RDS Data Service usando AWS SDKs
- Exemplos de código para o Amazon Redshift usando AWS SDKs
- Exemplos de código para o Amazon Rekognition usando AWS SDKs
- Exemplos de código para Resource Groups usando AWS SDKs
- Exemplos de código para a API Resource Groups Tagging usando AWS SDKs
- Exemplos de código para o Route 53 usando AWS SDKs
- Exemplos de código para registro de domínio do Route 53 usando AWS SDKs
- Exemplos de código para o Amazon S3 usando AWS SDKs
- Exemplos de código para o Amazon S3 Control usando AWS SDKs
- Exemplos de código para buckets do S3 Directory usando AWS SDKs
- Exemplos de código para uso de SageMaker IA AWS SDKs
- Exemplos de código para o Secrets Manager usando AWS SDKs
- Exemplos de código para o Amazon SES usando AWS SDKs
- Exemplos de código para a API v2 do Amazon SES usando AWS SDKs
- Exemplos de código para o Amazon SNS usando AWS SDKs
- Exemplos de código para o Amazon SQS usando AWS SDKs
- Exemplos de código para Step Functions usando AWS SDKs
- Exemplos de código para AWS STS usar AWS SDKs
- Exemplos de código para Suporte usar AWS SDKs
- Exemplos de código para Systems Manager usando AWS SDKs
- Exemplos de código para o Amazon Textract usando AWS SDKs
- Exemplos de código para o Amazon Transcribe usando AWS SDKs
- Exemplos de código para Amazon Transcribe Streaming usando AWS SDKs
- Exemplos de código para o Amazon Translate usando AWS SDKs
- Exemplos de código para VPC Lattice usando AWS SDKs
- Exemplos de código para AWS WAFV2 usar AWS SDKs
- Exemplos de código para WorkSpaces usar AWS SDKs
- Exemplos de código para X-Ray usar AWS SDKs

Exemplos de código para ACM usando AWS SDKs

Os exemplos de código a seguir mostram como usar Gerenciador de certificados da AWS com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário do ACM: mais informações sobre o ACM.
- Referência da API ACM: detalhes sobre todas as ações do ACM disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do ACM AWS SDKs
  - Aprenda as noções básicas do ACM com um AWS SDK
  - Ações para uso do ACM AWS SDKs
    - Use AddTagsToCertificatecom um AWS SDK ou CLI
    - Use DeleteCertificatecom um AWS SDK ou CLI
    - Use DescribeCertificatecom um AWS SDK ou CLI
    - Use ExportCertificatecom um AWS SDK ou CLI
    - Use GetCertificatecom um AWS SDK ou CLI
    - Use ImportCertificatecom um AWS SDK ou CLI
    - Use ListCertificatescom um AWS SDK ou CLI
    - Use ListTagsForCertificatecom um AWS SDK ou CLI
    - Use RemoveTagsFromCertificatecom um AWS SDK ou CLI
    - Use RenewCertificatecom um AWS SDK

- Use RequestCertificatecom um AWS SDK ou CLI
- Use ResendValidationEmailcom um AWS SDK ou CLI
- Use UpdateCertificateOptionscom um AWS SDK ou CLI

Exemplos básicos para o uso do ACM AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do Gerenciador de certificados da AWS com AWS SDKs.

Exemplos

- Aprenda as noções básicas do ACM com um AWS SDK
- Ações para uso do ACM AWS SDKs
  - Use AddTagsToCertificatecom um AWS SDK ou CLI
  - Use DeleteCertificatecom um AWS SDK ou CLI
  - Use DescribeCertificatecom um AWS SDK ou CLI
  - Use ExportCertificatecom um AWS SDK ou CLI
  - Use GetCertificatecom um AWS SDK ou CLI
  - Use ImportCertificatecom um AWS SDK ou CLI
  - Use ListCertificatescom um AWS SDK ou CLI
  - Use ListTagsForCertificatecom um AWS SDK ou CLI
  - Use RemoveTagsFromCertificatecom um AWS SDK ou CLI
  - Use RenewCertificatecom um AWS SDK
  - Use RequestCertificatecom um AWS SDK ou CLI
  - Use ResendValidationEmailcom um AWS SDK ou CLI
  - Use UpdateCertificateOptionscom um AWS SDK ou CLI

Aprenda as noções básicas do ACM com um AWS SDK

O exemplo de código a seguir mostra como:

- Solicitar um certificado do ACM.
- Importar um certificado autoassinado.
- Listar e descrever certificados.

- Remover certificados.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva operações do ACM.

```
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


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

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


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

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

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

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

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

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


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

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

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


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

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

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


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

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


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

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

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


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

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


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

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

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


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

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

Use a classe wrapper para gerenciar certificados para sua conta.

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

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

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

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

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

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

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

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

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

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

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AddTagsToCertificate
  - DeleteCertificate
  - DescribeCertificate
  - GetCertificate
  - ImportCertificate
  - ListCertificates
  - ListTagsForCertificate
  - RemoveTagsFromCertificate
  - RequestCertificate
  - ResendValidationEmail

Ações para uso do ACM AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do ACM com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do Gerenciador de certificados da AWS.

Exemplos

- Use AddTagsToCertificatecom um AWS SDK ou CLI
- Use DeleteCertificatecom um AWS SDK ou CLI
- Use DescribeCertificatecom um AWS SDK ou CLI
- Use ExportCertificatecom um AWS SDK ou CLI
- Use GetCertificatecom um AWS SDK ou CLI
- Use ImportCertificatecom um AWS SDK ou CLI
- Use ListCertificatescom um AWS SDK ou CLI
- Use ListTagsForCertificatecom um AWS SDK ou CLI
- Use RemoveTagsFromCertificatecom um AWS SDK ou CLI
- Use RenewCertificatecom um AWS SDK
- Use RequestCertificatecom um AWS SDK ou CLI
- Use ResendValidationEmailcom um AWS SDK ou CLI
- Use UpdateCertificateOptionscom um AWS SDK ou CLI

Use **`AddTagsToCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AddTagsToCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AddTagsToCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como adicionar tags a um certificado do ACM existente

O comando `add-tags-to-certificate` a seguir adiciona duas tags ao certificado especificado. Use um espaço para separar várias tags:

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

- Para obter detalhes da API, consulte AddTagsToCertificatena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

            Usage:    <certArn>

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

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

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

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

- Para obter detalhes da API, consulte AddTagsToCertificatea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

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

- Para obter detalhes da API, consulte a AddTagsToCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a AddTagsToCertificatereferência da API AWS SDK for SAP ABAP.

Use **`DeleteCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir um certificado do ACM da sua conta

O seguinte comando `delete-certificate` exclui o certificado com o ARN especificado:

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

- Para obter detalhes da API, consulte DeleteCertificatena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteCert {

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

            Usage:    <certArn>

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

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

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

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

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

- Para obter detalhes da API, consulte DeleteCertificatea Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui o certificado identificado pelo ARN fornecido e pela chave privada associada. O cmdlet solicitará confirmação antes de prosseguir. Adicione a opção -Force para ignorar a confirmação.

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

- Para obter detalhes da API, consulte DeleteCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui o certificado identificado pelo ARN fornecido e pela chave privada associada. O cmdlet solicitará confirmação antes de prosseguir. Adicione a opção -Force para ignorar a confirmação.

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

- Para obter detalhes da API, consulte DeleteCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

- Para obter detalhes da API, consulte a DeleteCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->deletecertificate( iv_certificatearn = iv_certificate_arn ).
        MESSAGE 'Certificate deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acmresourceinuseex.
        MESSAGE 'Certificate is in use and cannot be deleted.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCertificatereferência da API AWS SDK for SAP ABAP.

Use **`DescribeCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

```
namespace DescribeCertificate
{
    class DescribeCertificate
    {
        // The following example retrieves and displays the metadata for a
        // certificate using the AWS Certificate Manager (ACM) service.

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

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

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

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

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

        /// <summary>
        /// Displays detailed metadata about a certificate retrieved
        /// using the ACM service.
        /// </summary>
        /// <param name="certificateDetail">The object that contains details
        /// returned from the call to DescribeCertificateAsync.</param>
        static void DisplayCertificateDetails(CertificateDetail
 certificateDetail)
        {
```

```
            Console.WriteLine("\nCertificate Details: ");
            Console.WriteLine($"Certificate Domain:
{certificateDetail.DomainName}");
            Console.WriteLine($"Certificate Arn:
{certificateDetail.CertificateArn}");
            Console.WriteLine($"Certificate Subject:
{certificateDetail.Subject}");
            Console.WriteLine($"Certificate Status: {certificateDetail.Status}");
            foreach (var san in certificateDetail.SubjectAlternativeNames)
            {
                Console.WriteLine($"Certificate SubjectAlternativeName: {san}");
            }
        }

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

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

            return response;
```

```
        }
    }

}
```

- Para obter detalhes da API, consulte DescribeCertificatea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

```
    else {
        Aws::ACM::Model::CertificateDetail certificate =
                outcome.GetResult().GetCertificate();

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

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

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

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

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

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

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

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

                switch (recordType) {
                    case Aws::ACM::Model::RecordType::CNAME:
                        type = "CNAME";
                        break;
                    case Aws::ACM::Model::RecordType::NOT_SET:
```

```
                        type = "Not set";
                        break;
                    default:
                        type = "Cannot determine.";
                        break;
                }

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

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

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

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

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

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

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

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

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

```
                    default:
                        method = "Cannot determine";
                }

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

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

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

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

            }
        }

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

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

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

```
                Aws::String name;

                switch (usageName) {
                    case Aws::ACM::Model::ExtendedKeyUsageName::ANY:
                        name = "Any";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::CODE_SIGNING:
                        name = "Code signing";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::CUSTOM:
                        name = "Custom";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::EMAIL_PROTECTION:
                        name = "Email protection";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_END_SYSTEM:
                        name = "IPSEC end system";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_TUNNEL:
                        name = "IPSEC tunnel";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_USER:
                        name = "IPSEC user";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::NONE:
                        name = "None";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::NOT_SET:
                        name = "Not set";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::OCSP_SIGNING:
                        name = "OCSP signing";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::TIME_STAMPING:
                        name = "Time stamping";
                        break;
                    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_CLIENT_AUTHENTICATION:
                        name = "TLS web client authentication";
                        break;
                    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_SERVER_AUTHENTICATION:
                        name = "TLS web server authentication";
                        break;
```

```
                    default:
                        name = "Cannot determine";
                }

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

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

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

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

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

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

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

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

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

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

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

            switch (revocationReason) {
                case Aws::ACM::Model::RevocationReason::AFFILIATION_CHANGED:
                    reason = "Affiliation changed";
                    break;
                case Aws::ACM::Model::RevocationReason::A_A_COMPROMISE:
                    reason = "AA compromise";
                    break;
```

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

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

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

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

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

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

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

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

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

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

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

```
                algorithm = "RSA 2048";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_4096:
                algorithm = "RSA 4096";
                break;
            default:
                algorithm = "Cannot determine";
        }

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

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

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

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

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

```
        }

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

        std::cout << "Serial:              " << certificate.GetSerial() <<
                  std::endl;
        std::cout << "Signature algorithm: "
                  << certificate.GetSignatureAlgorithm() << std::endl;
        std::cout << "Subject:             " << certificate.GetSubject() <<
                  std::endl;

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

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

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeCertificate a Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para recuperar os campos contidos em um certificado do ACM

O seguinte comando `describe-certificate` recupera todos os campos do certificado com o ARN especificado:

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

Uma saída semelhante à seguinte é exibida:

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

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

- Para obter detalhes da API, consulte DescribeCertificatena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

public class DescribeCert {

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

            Usage:    <certArn>

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

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

    /**
     * Describes the details of an SSL/TLS certificate.
```

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

        try {
            DescribeCertificateResponse response =
 acmClient.describeCertificate(req);

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

- Para obter detalhes da API, consulte DescribeCertificatea Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna detalhes do certificado especificado.

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

Saída:

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

- Para obter detalhes da API, consulte DescribeCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes do certificado especificado.

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

Saída:

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

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

- Para obter detalhes da API, consulte DescribeCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

        :param certificate_arn: The Amazon Resource Name (ARN) of the
 certificate.
        :return: Metadata about the certificate.
```

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

- Para obter detalhes da API, consulte a DescribeCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

```
        MESSAGE 'Certificate not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeCertificatereferência da API AWS SDK for SAP ABAP.

Use **ExportCertificate**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ExportCertificate`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

    Aws::ACM::Model::ExportCertificateOutcome outcome =
```

```
            acm_client.ExportCertificate(request);

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

        auto result = outcome.GetResult();

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

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ExportCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para exportar um certificado privado emitido por uma CA privada.

O comando `export-certificate` a seguir exporta um certificado privado, uma cadeia de certificados e uma chave privada para a sua exibição:

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

Para exportar o certificado, a cadeia e a chave privada para um arquivo local, use o seguinte comando:

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

- Para obter detalhes da API, consulte ExportCertificatena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

            Usage:    <certArn>

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

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

    /**
     * Exports an SSL/TLS certificate and its associated private key and
certificate chain from AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate that you
want to export.
     * @throws IOException If an I/O error occurs while reading the private key
passphrase file or exporting the certificate.
     */
    public static void exportCert(String certArn) throws IOException {
        AcmClient acmClient = AcmClient.create();

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

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

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

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

        // Create a request object.
        ExportCertificateRequest req = ExportCertificateRequest.builder()
```

```
            .certificateArn(certArn)
            .passphrase(SdkBytes.fromByteBuffer(bufPassphrase))
            .build();

        // Export the certificate.
        ExportCertificateResponse result = null;
        try {
            result = acmClient.exportCertificate(req);
        } catch (InvalidArnException | InvalidTagException |
 ResourceNotFoundException ex) {
            throw ex;
        }

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

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

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

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

- Para obter detalhes da API, consulte ExportCertificatea Referência AWS SDK for Java 2.x da API.

Use **`GetCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

        auto result = outcome.GetResult();

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

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para recuperar um certificado do ACM

O seguinte comando `get-certificate` recupera o certificado para o ARN especificado e a cadeia de certificados:

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

Uma saída semelhante à seguinte é exibida:

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

  "CertificateChain": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
```

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

```
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----"
}
```

- Para obter detalhes da API, consulte GetCertificatena Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra como retornar um certificado e sua cadeia usando o ARN do certificado.

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

- Para obter detalhes da API, consulte GetCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como retornar um certificado e sua cadeia usando o ARN do certificado.

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

- Para obter detalhes da API, consulte GetCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

- Para obter detalhes da API, consulte a GetCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        oo_result = lo_acm->getcertificate( iv_certificatearn =
 iv_certificate_arn ).
        MESSAGE 'Certificate body and chain retrieved.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acmrequestinprgssex.
        MESSAGE 'Certificate request is in progress.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetCertificatereferência da API AWS SDK for SAP ABAP.

Use **`ImportCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ImportCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Import an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateFile: Path to certificate to import.
  \param privateKeyFile: Path to file containing a private key.
  \param certificateChainFile: Path to file containing a PEM encoded certificate
 chain.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::importCertificate(const Aws::String &certificateFile,
                                    const Aws::String &privateKeyFile,
                                    const Aws::String &certificateChainFile,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    std::ifstream certificateInStream(certificateFile.c_str());
    if (!certificateInStream) {
        std::cerr << "Error: The certificate file '" << certificateFile <<
                  "' does not exist." << std::endl;

        return false;
    }

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

        return false;
    }

    std::ifstream certificateChainInStream(certificateChainFile.c_str());
    if (!certificateChainInStream) {
```

```
        std::cerr << "Error: The certificate chain file '"
                  << certificateChainFile << "' does not exist." << std::endl;

        return false;
    }

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

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

    Aws::String certificateChain;

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

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

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

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

certificateChain.c_str(),

certificateChain.size()));

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

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

        return false;
    }
```

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

        return true;
    }
}
```

- Para obter detalhes da API, consulte ImportCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como importar um certificado para o ACM.

O comando `import-certificate` a seguir importa um certificado para o ACM. Substitua os nomes dos arquivos pelos seus próprios:

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

- Para obter detalhes da API, consulte ImportCertificatena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
```

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

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

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

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

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

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

    /**
     * Imports an SSL/TLS certificate and private key from S3 into AWS
 Certificate Manager (ACM).
     *
     * @param bucketName     The name of the S3 bucket.
     * @param certificateKey The key for the SSL/TLS certificate file in S3.
     * @param privateKeyKey  The key for the private key file in S3.
     * @return The ARN of the imported certificate.
     */
```

```
   public static String importCertificate(String bucketName, String
certificateKey, String privateKeyKey) {
       AcmClient acmClient = AcmClient.create();
       S3Client s3Client = S3Client.create();

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

           ImportCertificateRequest request = ImportCertificateRequest.builder()

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

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

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

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

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

```
                .bucket(bucketName)
                .key(objectKey)
                .build();

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

- Para obter detalhes da API, consulte ImportCertificatea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


    def import_certificate(self, certificate_body, private_key):
```

```
        """
        Imports a self-signed certificate to ACM.

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

- Para obter detalhes da API, consulte a ImportCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Only pass certificate chain if it's provided (it's optional)
        IF iv_certificate_chain IS NOT INITIAL.
          DATA(lo_result) = lo_acm->importcertificate(
            iv_certificate = iv_certificate
```

```
            iv_privatekey = iv_private_key
            iv_certificatechain = iv_certificate_chain
          ).
        ELSE.
          lo_result = lo_acm->importcertificate(
            iv_certificate = iv_certificate
            iv_privatekey = iv_private_key
          ).
        ENDIF.
        ov_certificate_arn = lo_result->get_certificatearn( ).
        MESSAGE 'Certificate imported successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'I'.
      CATCH /aws1/cx_acmlimitexceededex.
        MESSAGE 'Certificate limit exceeded.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ImportCertificatereferência da API AWS SDK for SAP ABAP.

Use **`ListCertificates`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListCertificates`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

        private static readonly RegionEndpoint ACMRegion =
 RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

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

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

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

        /// <summary>
        /// Retrieves a list of the certificates defined in this Region.
        /// </summary>
        /// <param name="client">The ACM client object passed to the
        /// ListCertificateResAsync method call.</param>
        /// <param name="request"></param>
        /// <returns>The ListCertificatesResponse.</returns>
```

```
        static async Task<ListCertificatesResponse>
 ListCertificatesResponseAsync(
            AmazonCertificateManagerClient client)
        {
            var request = new ListCertificatesRequest();

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

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

```
            request.SetNextToken(nextToken);
        }

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

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

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

            const Aws::Vector<Aws::ACM::Model::CertificateSummary> &certificates
 =
                    result.GetCertificateSummaryList();
            allCertificates.insert(allCertificates.end(), certificates.begin(),
                                   certificates.end());

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

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

    return true;
}
```

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar os certificados do ACM para uma AWS conta

O seguinte comando `list-certificates` lista os ARNs dos certificados na conta:

```
aws acm list-certificates
```

O comando anterior produz uma saída semelhante à seguinte:

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

Você pode decidir quantos certificados deseja exibir sempre que chamar `list-certificates`. Por exemplo, se você tem quatro certificados e deseja exibir apenas dois por vez, defina o argumento `max-items` como 2, conforme o seguinte exemplo:

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

Dois ARNs de certificado e um valor `NextToken` serão exibidos:

```
"CertificateSummaryList": [
```

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

Para exibir os próximos dois certificados em sua conta, defina o valor `NextToken` na próxima chamada:

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

É possível filtrar a saída usando o argumento `certificate-statuses`. O seguinte comando exibe certificados com o status PENDING_VALIDATION:

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

Também é possível filtrar a saída usando o argumento `includes`. O comando a seguir exibe os certificados filtrados nas propriedades a seguir. Os certificados a serem exibidos:

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

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

- Para obter detalhes da API, consulte ListCertificatesna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCerts {
    public static void main(String[] args) {
        listCertificates();
    }

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

            // Print the certificate details using streams
            listResponse.certificateSummaryList().stream()
                .forEach(certificate -> {
```

```
                    System.out.println("Certificate ARN: " +
 certificate.certificateArn());
                    System.out.println("Certificate Domain Name: " +
 certificate.domainName());
                    System.out.println("Certificate Status: " +
 certificate.statusAsString());
                    System.out.println("---");
                });

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

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: recupera uma lista todos os ARNs de certificado e o nome de domínio de cada. O cmdlet fará a paginação automaticamente para recuperar todos os ARNs. Para controlar manualmente a paginação, use o MaxItem parâmetro - para controlar quantos ARNs de certificado são retornados para cada chamada de serviço e o NextToken parâmetro - para indicar o ponto de partida de cada chamada.

```
Get-ACMCertificateList
```

Saída:

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

Exemplo 2: recupera uma lista de todos os ARNs do certificado em que o status corresponde aos estados fornecidos.

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

Exemplo 3: esse exemplo retorna uma lista de todos os certificados na região us-east-1 que têm um tipo de chave RSA_2048 e um uso estendido da chave, ou uma finalidade, de CODE_SIGNING. Você pode encontrar os valores desses parâmetros de filtragem no tópico de referência da API de ListCertificates filtros: https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html.

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

Saída:

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

- Para obter detalhes da API, consulte ListCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: recupera uma lista todos os ARNs de certificado e o nome de domínio de cada. O cmdlet fará a paginação automaticamente para recuperar todos os ARNs. Para controlar manualmente a paginação, use o MaxItem parâmetro - para controlar quantos ARNs de certificado são retornados para cada chamada de serviço e o NextToken parâmetro - para indicar o ponto de partida de cada chamada.

```
Get-ACMCertificateList
```

Saída:

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

Exemplo 2: recupera uma lista de todos os ARNs do certificado em que o status corresponde aos estados fornecidos.

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

Exemplo 3: esse exemplo retorna uma lista de todos os certificados na região us-east-1 que têm um tipo de chave RSA_2048 e um uso estendido da chave, ou uma finalidade, de CODE_SIGNING. Você pode encontrar os valores desses parâmetros de filtragem no tópico de referência da API de ListCertificates filtros: https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html.

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

Saída:

```
CertificateArn
   DomainName
--------------
   ----------
```

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

- Para obter detalhes da API, consulte ListCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

```
        except ClientError:
            logger.exception("Couldn't get certificates.")
            raise
        else:
            return certificates
```

- Para obter detalhes da API, consulte a ListCertificatesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a ListCertificatesreferência da API AWS SDK for SAP ABAP.

Use **`ListTagsForCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTagsForCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

        return false;
    }
```

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

        auto result = outcome.GetResult();

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

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

        return true;
    }
}
```

- Para obter detalhes da API, consulte ListTagsForCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar as tags aplicadas a um certificado do ACM

O seguinte comando `list-tags-for-certificate` lista as tags aplicadas a um certificado na conta:

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

O comando anterior produz uma saída semelhante à seguinte:

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

- Para obter detalhes da API, consulte ListTagsForCertificatena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

    public static void main(String[] args) {

        final String usage = """
```

```
            Usage:    <certArn>

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

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

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

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

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

- Para obter detalhes da API, consulte ListTagsForCertificatea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

- Para obter detalhes da API, consulte a ListTagsForCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a ListTagsForCertificatereferência da API AWS SDK for SAP ABAP.

Use **`RemoveTagsFromCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RemoveTagsFromCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

    tags.push_back(tag);

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

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

    if (!outcome.IsSuccess()) {
```

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

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

        return true;
    }
}
```

- Para obter detalhes da API, consulte RemoveTagsFromCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como remover uma tag de um certificado do ACM

O comando `remove-tags-from-certificate` a seguir remove duas tags do certificado especificado. Use um espaço para separar várias tags:

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

- Para obter detalhes da API, consulte RemoveTagsFromCertificatena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

public class RemoveTagsFromCert {

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

            Usage:    <certArn>

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

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

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

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

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

- Para obter detalhes da API, consulte RemoveTagsFromCertificatea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

- Para obter detalhes da API, consulte a RemoveTagsFromCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a RemoveTagsFromCertificatereferência da API AWS SDK for SAP ABAP.

Use **`RenewCertificate`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `RenewCertificate`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

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

        return true;
    }
}
```

- Para obter detalhes da API, consulte RenewCertificatea Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

            Usage:    <certArn>

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

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

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

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

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

- Para obter detalhes da API, consulte RenewCertificatea Referência AWS SDK for Java 2.x da API.

Use **`RequestCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RequestCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

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

```
        return true;
    }
}
```

- Para obter detalhes da API, consulte RequestCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como solicitar um novo certificado do ACM

O seguinte comando `request-certificate` solicita um novo certificado para o domínio www.example.com usando a validação de DNS:

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

Você pode inserir um token de idempotência para distinguir entre chamadas para `request-certificate`:

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

Você pode inserir um ou mais nomes alternativos de assunto para solicitar um certificado que protegerá mais de um domínio do apex:

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

Você pode inserir um nome alternativo que também pode ser usado para acessar seu site:

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

Você pode usar um asterisco (*) como um caractere curinga para criar um certificado para vários subdomínios no mesmo domínio:

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

Você também pode inserir vários nomes alternativos:

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

Se você estiver usando e-mail para validação, poderá inserir as opções de validação de domínio para especificar o domínio para o qual o e-mail de validação será enviado:

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

O seguinte comando desativa o registro em log de transparência do certificado ao solicitar um novo certificado:

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

- Para obter detalhes da API, consulte RequestCertificatena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
```

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

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

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

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

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

- Para obter detalhes da API, consulte RequestCertificatea Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria um certificado. O serviço retorna o ARN do novo certificado.

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

Saída:

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

Exemplo 2: cria um certificado. O serviço retorna o ARN do novo certificado.

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

Saída:

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

- Para obter detalhes da API, consulte RequestCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria um certificado. O serviço retorna o ARN do novo certificado.

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

Saída:

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

Exemplo 2: cria um certificado. O serviço retorna o ARN do novo certificado.

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

Saída:

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

- Para obter detalhes da API, consulte RequestCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

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

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

- Para obter detalhes da API, consulte a RequestCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a RequestCertificatereferência da API AWS SDK for SAP ABAP.

Use **`ResendValidationEmail`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ResendValidationEmail`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

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

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

        return true;
    }
}
```

- Para obter detalhes da API, consulte ResendValidationEmailna Referência AWS SDK para C ++ da API.

CLI

AWS CLI

Como reenviar o e-mail de validação para a solicitação de certificado do ACM

O seguinte `resend-validation-email` comando instrui a autoridade de certificação da Amazon a enviar e-mails de validação para os endereços apropriados:

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

- Para obter detalhes da API, consulte ResendValidationEmailna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: solicita que o e-mail para validar a propriedade do domínio para “www.example.com” seja enviado. Se o $ do seu shell ConfirmPreference estiver definido como 'Médio' ou inferior, o cmdlet solicitará a confirmação antes de continuar. Adicione a opção -Force para ignorar as solicitações de confirmação.

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

- Para obter detalhes da API, consulte ResendValidationEmailem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: solicita que o e-mail para validar a propriedade do domínio para “www.example.com” seja enviado. Se o $ do seu shell ConfirmPreference estiver definido como 'Médio' ou inferior, o cmdlet solicitará a confirmação antes de continuar. Adicione a opção -Force para ignorar as solicitações de confirmação.

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

- Para obter detalhes da API, consulte ResendValidationEmailem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
```

```
    """
    Encapsulates ACM functions.
    """

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


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

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

- Para obter detalhes da API, consulte a ResendValidationEmailReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a ResendValidationEmailreferência da API AWS SDK for SAP ABAP.

Use **`UpdateCertificateOptions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateCertificateOptions`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

    if (loggingEnabled) {
        options.SetCertificateTransparencyLoggingPreference(

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

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

    request.SetOptions(options);

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

```
            acmClient.UpdateCertificateOptions(request);

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

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

        return true;
    }
}
```

- Para obter detalhes da API, consulte UpdateCertificateOptionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para atualizar as opções de certificado

O comando `update-certificate-options` a seguir desativa o registro em log de transparência do certificado:

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

- Para obter detalhes da API, consulte UpdateCertificateOptionsna Referência de AWS CLI Comandos.

Exemplos de código para o API Gateway usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon API Gateway com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS as contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Mais atributos

- Guia do desenvolvedor do API Gateway: mais informações sobre o API Gateway.
- Referência da API API Gateway: detalhes sobre todas as ações disponíveis do API Gateway.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de API Gateway usando AWS SDKs
  - Ações para o API Gateway usando AWS SDKs
    - Use CreateDeploymentcom um AWS SDK ou CLI
    - Use CreateResourcecom um AWS SDK ou CLI
    - Use CreateRestApicom um AWS SDK ou CLI
    - Use DeleteDeploymentcom um AWS SDK ou CLI
    - Use DeleteRestApicom um AWS SDK ou CLI
    - Use GetBasePathMappingcom um AWS SDK ou CLI
    - Use GetResourcescom um AWS SDK ou CLI
    - Use GetRestApiscom um AWS SDK ou CLI

    - Use ListBasePathMappingscom um AWS SDK
    - Use PutIntegrationcom um AWS SDK ou CLI
    - Use PutIntegrationResponsecom um AWS SDK ou CLI
    - Use PutMethodcom um AWS SDK ou CLI
    - Use PutMethodResponsecom um AWS SDK ou CLI
    - Use UpdateBasePathMappingcom um AWS SDK ou CLI
- Cenários para o API Gateway usando AWS SDKs
  - Crie uma API REST do API Gateway para rastrear COVID-19 dados
  - Criar uma API REST de biblioteca de empréstimos
  - Criar uma API REST com integração de proxy de função
  - Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
  - Criar uma aplicação de chat websocket com o API Gateway
  - Crie e implante uma API REST usando um AWS SDK
  - Usar o API Gateway para invocar uma função do Lambda
- AWS contribuições da comunidade para o API Gateway
  - Compilar e testar uma aplicação com tecnologia sem servidor

Exemplos básicos de API Gateway usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon API Gateway com AWS SDKs.

Exemplos

- Ações para o API Gateway usando AWS SDKs
  - Use CreateDeploymentcom um AWS SDK ou CLI
  - Use CreateResourcecom um AWS SDK ou CLI
  - Use CreateRestApicom um AWS SDK ou CLI
  - Use DeleteDeploymentcom um AWS SDK ou CLI
  - Use DeleteRestApicom um AWS SDK ou CLI
  - Use GetBasePathMappingcom um AWS SDK ou CLI
  - Use GetResourcescom um AWS SDK ou CLI

- Use GetRestApiscom um AWS SDK ou CLI
- Use ListBasePathMappingscom um AWS SDK
- Use PutIntegrationcom um AWS SDK ou CLI
- Use PutIntegrationResponsecom um AWS SDK ou CLI
- Use PutMethodcom um AWS SDK ou CLI
- Use PutMethodResponsecom um AWS SDK ou CLI
- Use UpdateBasePathMappingcom um AWS SDK ou CLI

Ações para o API Gateway usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do API Gateway com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do API Gateway e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o API Gateway usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa em Referência de API do Amazon API Gateway.

Exemplos

- Use CreateDeploymentcom um AWS SDK ou CLI
- Use CreateResourcecom um AWS SDK ou CLI
- Use CreateRestApicom um AWS SDK ou CLI
- Use DeleteDeploymentcom um AWS SDK ou CLI
- Use DeleteRestApicom um AWS SDK ou CLI
- Use GetBasePathMappingcom um AWS SDK ou CLI
- Use GetResourcescom um AWS SDK ou CLI
- Use GetRestApiscom um AWS SDK ou CLI
- Use ListBasePathMappingscom um AWS SDK
- Use PutIntegrationcom um AWS SDK ou CLI
- Use PutIntegrationResponsecom um AWS SDK ou CLI

- Use PutMethodcom um AWS SDK ou CLI
- Use PutMethodResponsecom um AWS SDK ou CLI
- Use UpdateBasePathMappingcom um AWS SDK ou CLI

Use **`CreateDeployment`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateDeployment`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma API REST com integração de proxy de função
- Criar e implantar uma API REST

CLI

AWS CLI

Como implantar os recursos configurados de uma API em um novo estágio

Comando:

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

Como implantar os recursos configurados de uma API em um estágio existente

Comando:

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

Como implantar os recursos configurados de uma API em um estágio existente com variáveis de estágio

aws apigateway create-deployment --rest-api-id 1234123412 --stage-name dev --description 'Terceira implantação no estágio de desenvolvimento' --variables key='value',otherKey='otherValue'

- Para obter detalhes da API, consulte CreateDeploymentna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

- Para obter detalhes da API, consulte CreateDeploymenta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

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



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

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

- Para obter detalhes da API, consulte a CreateDeploymentReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

- Para obter detalhes da API, consulte a CreateDeploymentreferência da API AWS SDK for SAP ABAP.

Use **`CreateResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateResource`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma API REST com integração de proxy de função
- Criar e implantar uma API REST

CLI

AWS CLI

Como criar um recurso em uma API

Comando:

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

- Para obter detalhes da API, consulte CreateResourcena Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

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

- Para obter detalhes da API, consulte a CreateResourceReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateResourcereferência da API AWS SDK for SAP ABAP.

Use **`CreateRestApi`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateRestApi`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma API REST com integração de proxy de função
- Criar e implantar uma API REST

CLI

AWS CLI

Como criar uma API

Comando:

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

Como criar uma API duplicada a partir de uma API existente

Comando:

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

- Para obter detalhes da API, consulte CreateRestApina Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

- Para obter detalhes da API, consulte CreateRestApia Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

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

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

        return self.api_id
```

- Para obter detalhes da API, consulte a CreateRestApiReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

- Para obter detalhes da API, consulte a CreateRestApireferência da API AWS SDK for SAP ABAP.

Use **`DeleteDeployment`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteDeployment`.

CLI

AWS CLI

Como excluir uma implantação em uma API

Comando:

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

- Para obter detalhes da API, consulte DeleteDeploymentna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

- Para obter detalhes da API, consulte DeleteDeploymenta Referência AWS SDK for Java 2.x da API.

Use **`DeleteRestApi`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteRestApi`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma API REST com integração de proxy de função
- Criar e implantar uma API REST

CLI

AWS CLI

Como excluir uma API

Comando:

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

- Para obter detalhes da API, consulte DeleteRestApina Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

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

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

- Para obter detalhes da API, consulte DeleteRestApia Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

- Para obter detalhes da API, consulte a DeleteRestApiReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

- Para obter detalhes da API, consulte a DeleteRestApireferência da API AWS SDK for SAP ABAP.

Use **`GetBasePathMapping`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBasePathMapping`.

CLI

AWS CLI

Como obter o mapeamento de caminho base para um nome de domínio personalizado

Comando:

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

Saída:

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

- Para obter detalhes da API, consulte GetBasePathMappingna Referência de AWS CLI Comandos.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
```

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

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

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

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

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- Para obter detalhes da API, consulte GetBasePathMappinga Referência AWS SDK para PHP da API.

Use **`GetResources`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetResources`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma API REST com integração de proxy de função
- Criar e implantar uma API REST

CLI

AWS CLI

Como obter uma lista de recursos para uma API REST

Comando:

```
aws apigateway get-resources --rest-api-id 1234123412
```

Saída:

```
{
    "items": [
        {
            "path": "/resource/subresource",
            "resourceMethods": {
                "POST": {}
            },
            "id": "024ace",
            "pathPart": "subresource",
            "parentId": "ai5b02"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetResourcesna Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


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

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

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

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

        return self.api_id
```

- Para obter detalhes da API, consulte a GetResourcesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_agw->getresources(
          iv_restapiid = iv_rest_api_id ).
        DATA(lt_resources) = oo_result->get_items( ).
        DATA(lv_count) = lines( lt_resources ).
        MESSAGE 'Found ' && lv_count && ' resources' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
```

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

- Para obter detalhes da API, consulte a GetResourcesreferência da API AWS SDK for SAP ABAP.

Use **`GetRestApis`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetRestApis`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma API REST com integração de proxy de função
- Criar e implantar uma API REST

CLI

AWS CLI

Como obter uma lista de APIs REST

Comando:

```
aws apigateway get-rest-apis
```

Saída:

```
{
    "items": [
        {
            "createdDate": 1438884790,
            "id": "12s44z21rb",
```

```
            "name": "My First API"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetRestApisna Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


    def get_rest_api_id(self, api_name):
        """
        Gets the ID of a REST API from its name by searching the list of REST
 APIs
        for the current account. Because names need not be unique, this returns
 only
        the first API with the specified name.
```

```
        :param api_name: The name of the API to look up.
        :return: The ID of the specified API.
        """
        try:
            rest_api = None
            paginator = self.apig_client.get_paginator("get_rest_apis")
            for page in paginator.paginate():
                rest_api = next(
                    (item for item in page["items"] if item["name"] == api_name),
None
                )
                if rest_api is not None:
                    break
            self.api_id = rest_api["id"]
            logger.info("Found ID %s for API %s.", rest_api["id"], api_name)
        except ClientError:
            logger.exception("Couldn't find ID for API %s.", api_name)
            raise
        else:
            return rest_api["id"]
```

- Para obter detalhes da API, consulte a GetRestApisReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exiba as APIs REST do Amazon API Gateway na região.

```
async fn show_apis(client: &Client) -> Result<(), Error> {
    let resp = client.get_rest_apis().send().await?;
```

```
    for api in resp.items() {
        println!("ID:          {}", api.id().unwrap_or_default());
        println!("Name:        {}", api.name().unwrap_or_default());
        println!("Description: {}", api.description().unwrap_or_default());
        println!("Version:     {}", api.version().unwrap_or_default());
        println!(
            "Created:     {}",
            api.created_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetRestApisreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_agw->getrestapis( ).
        DATA(lt_apis) = oo_result->get_items( ).
        DATA(lv_count) = lines( lt_apis ).
        MESSAGE 'Found ' && lv_count && ' REST APIs' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetRestApisreferência da API AWS SDK for SAP ABAP.

Use **`ListBasePathMappings`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListBasePathMappings`.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 * /////////////////////////////////////////////////////////////////////////// */

function listBasePathMappings($apiGatewayClient, $domainName)
{
    try {
```

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

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

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listTheBasePathMappings();
```

- Para obter detalhes da API, consulte ListBasePathMappingsa Referência AWS SDK para PHP da API.

Use **`PutIntegration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutIntegration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma API REST com integração de proxy de função
- Criar e implantar uma API REST

CLI

AWS CLI

Como criar uma solicitação de integração MOCK

Comando:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-id a1b2c3
 --http-method GET --type MOCK --request-templates '{ "application/json":
 "{\"statusCode\": 200}" }'
```

Como criar uma solicitação de integração HTTP

Comando:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-id a1b2c3
 --http-method GET --type HTTP --integration-http-method GET --uri 'https://
domain.tld/path'
```

Para criar uma solicitação de AWS integração com um endpoint da Função Lambda

Comando:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-
id a1b2c3 --http-method GET --type AWS --integration-http-method POST
 --uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:function_name/invocations'
```

- Para obter detalhes da API, consulte PutIntegrationna Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
 integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
 with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
 POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
```

```
                        Gateway permission to use the specified action with the
                        service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
```

```
                requestTemplates={"application/json":
 json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Para obter detalhes da API, consulte a PutIntegrationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        oo_result = lo_agw->putintegration(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_type = 'AWS_PROXY'
          iv_integrationhttpmethod = 'POST'
          iv_uri = iv_integration_uri ).
        MESSAGE 'Integration configured for method' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutIntegrationreferência da API AWS SDK for SAP ABAP.

Use **`PutIntegrationResponse`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutIntegrationResponse`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e implantar uma API REST

CLI

AWS CLI

Como criar uma resposta de integração como a resposta padrão com um modelo de mapeamento definido

Comando:

```
aws apigateway put-integration-response --rest-api-id 1234123412 --resource-
id a1b2c3 --http-method GET --status-code 200 --selection-pattern "" --response-
templates '{"application/json": "{\"json\": \"template\"}"}'
```

Como criar uma resposta de integração com um regex de 400 e um valor de cabeçalho definido de forma estática

Comando:

```
aws apigateway put-integration-response --rest-api-id 1234123412 --resource-
id a1b2c3 --http-method GET --status-code 400 --selection-pattern 400 --response-
parameters '{"method.response.header.custom-header": "'"'"'custom-value'"'"'"}'
```

- Para obter detalhes da API, consulte PutIntegrationResponsena Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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

```
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
```

```
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json":
json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
```

```
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Para obter detalhes da API, consulte a PutIntegrationResponseReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_agw->putintegrationresponse(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_statuscode = '200' ).
        MESSAGE 'Integration response configured for status 200' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutIntegrationResponsereferência da API AWS SDK for SAP ABAP.

Use **PutMethod**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutMethod`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma API REST com integração de proxy de função
- Criar e implantar uma API REST

CLI

AWS CLI

Como criar um método para um recurso em uma API sem autorização, sem chave de API e com um cabeçalho de solicitação de método personalizado

Comando:

```
aws apigateway put-method --rest-api-id 1234123412 --resource-id a1b2c3 --
http-method PUT --authorization-type "NONE" --no-api-key-required --request-
parameters "method.request.header.custom-header=false"
```

- Para obter detalhes da API, consulte PutMethodna Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
 integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
 with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
 POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
```

```
                        Gateway permission to use the specified action with the
                        service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
```

```
                requestTemplates={"application/json":
 json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Para obter detalhes da API, consulte a PutMethodReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        oo_result = lo_agw->putmethod(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_authorizationtype = 'NONE' ).
        MESSAGE 'Method ' && iv_http_method && ' added to resource' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutMethodreferência da API AWS SDK for SAP ABAP.

Use **`PutMethodResponse`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutMethodResponse`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e implantar uma API REST

CLI

AWS CLI

Como criar uma resposta de método de acordo com o código de status especificado com um cabeçalho de resposta de método personalizado

Comando:

```
aws apigateway put-method-response --rest-api-id 1234123412 --
resource-id a1b2c3 --http-method GET --status-code 400 --response-
parameters "method.response.header.custom-header=false"
```

- Para obter detalhes da API, consulte PutMethodResponsena Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
```

```
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
```

```
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json":
json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Para obter detalhes da API, consulte a PutMethodResponseReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_agw->putmethodresponse(
      iv_restapiid = iv_rest_api_id
      iv_resourceid = iv_resource_id
      iv_httpmethod = iv_http_method
      iv_statuscode = '200' ).
    MESSAGE 'Method response configured for status 200' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutMethodResponsereferência da API AWS SDK for SAP ABAP.

Use **`UpdateBasePathMapping`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateBasePathMapping`.

CLI

AWS CLI

Como alterar o caminho base de um nome de domínio personalizado

Comando:

```
aws apigateway update-base-path-mapping --domain-name api.domain.tld --base-path prod --patch-operations op='replace',path='/basePath',value='v1'
```

Saída:

```
{
    "basePath": "v1",
    "restApiId": "1234123412",
    "stage": "api"
}
```

- Para obter detalhes da API, consulte UpdateBasePathMappingna Referência de AWS CLI Comandos.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

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


/* ///////////////////////////////////////////////////////////////////////////
 *
 * Purpose: Updates the base path mapping for a custom domain name
```

```
 * in Amazon API Gateway.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 * - $patchOperations: The base path update operations to apply.
 *
 * Returns: Information about the updated base path mapping, if available;
 * otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////// */

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

function updateTheBasePathMapping()
{
    $patchOperations = array([
        'op' => 'replace',
        'path' => '/stage',
        'value' => 'stage2'
    ]);

    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
```

```
        'version' => '2015-07-09'
    ]);

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

// Uncomment the following line to run this code in an AWS account.
// updateTheBasePathMapping();
```

- Para obter detalhes da API, consulte UpdateBasePathMappinga Referência AWS SDK para PHP da API.

Cenários para o API Gateway usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no API Gateway com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no API Gateway ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie uma API REST do API Gateway para rastrear COVID-19 dados
- Criar uma API REST de biblioteca de empréstimos
- Criar uma API REST com integração de proxy de função
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Criar uma aplicação de chat websocket com o API Gateway
- Crie e implante uma API REST usando um AWS SDK
- Usar o API Gateway para invocar uma função do Lambda

Crie uma API REST do API Gateway para rastrear COVID-19 dados

O exemplo de código a seguir mostra como criar uma API REST que simula um sistema para rastrear casos diários nos Estados Unidos da América, usando dados fictícios. COVID-19

Python

SDK para Python (Boto3)

Mostra como usar o AWS Chalice com o AWS SDK para Python (Boto3) para criar uma API REST sem servidor que usa o Amazon API Gateway e o Amazon DynamoDB AWS Lambda. A API REST simula um sistema que rastreia casos diários nos Estados Unidos da América, usando dados fictícios. COVID-19 Aprenda como:

- Use o AWS Chalice para definir rotas nas funções do Lambda que são chamadas para lidar com solicitações REST que chegam por meio do API Gateway.
- Usar as funções do Lambda para recuperar e armazenar dados em uma tabela do DynamoDB para atender a solicitações REST.
- Defina a estrutura da tabela e os recursos da função de segurança em um AWS CloudFormation modelo.
- Use AWS Chalice e CloudFormation para empacotar e implantar todos os recursos necessários.
- Use CloudFormation para limpar todos os recursos criados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice para criar uma API REST apoiada por um banco de dados Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Use AWS Secrets Manager para gerenciar as credenciais do banco de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST sem servidor no Amazon API Gateway e. AWS Lambda
- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Criar uma API REST com integração de proxy de função

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para execução do Lambda.
- Criar e implantar uma função do Lambda.
- Criar uma API REST
- Configurar a integração de proxy do Lambda.
- Implantar e testar a API
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
```

```
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
```

```
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
```

```
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"
```

```
# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2
```

```
aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação de chat websocket com o API Gateway

O exemplo de código a seguir mostra como criar uma aplicação de chat que é atendido por uma API de Websocket criada no Amazon API Gateway.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon API Gateway V2 para criar uma API de websocket que se integre ao Amazon AWS Lambda DynamoDB.

- Crie uma API de Websocket atendida pelo API Gateway.
- Defina um manipulador do Lambda que armazena conexões no DynamoDB e publica mensagens para outros participantes do chat.
- Conecte-se à aplicação de chat websocket e envie mensagens com o pacote Websockets.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Crie e implante uma API REST usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar uma API REST atendida pelo API Gateway.
- Adicionar recursos à API REST para representar um perfil de usuário.
- Adicionar métodos de integração para que a API REST use uma tabela do DynamoDB para armazenar dados de perfil de usuário.

- Enviar solicitações HTTP para a API REST para adicionar e recuperar perfis de usuário.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva operações do API Gateway.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


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

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

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

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

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

        return self.api_id


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

        :param parent_id: The ID of the parent resource.
        :param resource_path: The path of the new resource, relative to the
parent.
        :return: The ID of the new resource.
        """
        try:
```

```
            result = self.apig_client.create_resource(
                restApiId=self.api_id, parentId=parent_id, pathPart=resource_path
            )
            resource_id = result["id"]
            logger.info("Created resource %s.", resource_path)
        except ClientError:
            logger.exception("Couldn't create resource %s.", resource_path)
            raise
        else:
            return resource_id


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
```

```
                                    body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json":
json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
```

```
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise


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

        :param stage_name: The stage of the API to deploy, such as 'test'.
        :return: The base URL of the deployed REST API.
        """
        try:
            self.apig_client.create_deployment(
                restApiId=self.api_id, stageName=stage_name
            )
            self.stage = stage_name
            logger.info("Deployed stage %s.", stage_name)
        except ClientError:
            logger.exception("Couldn't deploy stage %s.", stage_name)
            raise
        else:
            return self.api_url()



    def api_url(self, resource=None):
        """
```

```
        Builds the REST API URL from its parts.

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

Implantar uma API REST e chamá-la com o pacote Requests.

```
def usage_demo(table_name, role_name, rest_api_name):
    """
    Demonstrates how to used API Gateway to create and deploy a REST API, and how
    to use the Requests package to call it.

    :param table_name: The name of the demo DynamoDB table.
    :param role_name: The name of the demo role that grants API Gateway
 permission to
                        call DynamoDB.
    :param rest_api_name: The name of the demo REST API created by the demo.
    """
    gateway = ApiGatewayToService(boto3.client("apigateway"))
    role = boto3.resource("iam").Role(role_name)

    print("Creating REST API in API Gateway.")
    gateway.create_rest_api(rest_api_name)

    print("Adding resources to the REST API.")
    profiles_id = gateway.add_rest_resource(gateway.root_id, "profiles")
    username_id = gateway.add_rest_resource(profiles_id, "{username}")

    # The DynamoDB service requires that all integration requests use POST.
    print("Adding integration methods to read and write profiles in Amazon
 DynamoDB.")
    gateway.add_integration_method(
```

```
        profiles_id,
        "GET",
        "dynamodb",
        "Scan",
        "POST",
        role.arn,
        {"TableName": table_name},
    )
    gateway.add_integration_method(
        profiles_id,
        "POST",
        "dynamodb",
        "PutItem",
        "POST",
        role.arn,
        {
            "TableName": table_name,
            "Item": {
                "username": {"S": "$input.path('$.username')"},
                "name": {"S": "$input.path('$.name')"},
                "title": {"S": "$input.path('$.title')"},
            },
        },
    )
    gateway.add_integration_method(
        username_id,
        "GET",
        "dynamodb",
        "GetItem",
        "POST",
        role.arn,
        {
            "TableName": table_name,
            "Key": {"username": {"S": "$method.request.path.username"}},
        },
    )

    stage = "test"
    print(f"Deploying the {stage} stage.")
    gateway.deploy_api(stage)

    profiles_url = gateway.api_url("profiles")
    print(
```

```
        f"Using the Requests package to post some people to the profiles REST API
 at "
        f"{profiles_url}."
    )
    requests.post(
        profiles_url,
        json={"username": "will", "name": "William Shakespeare", "title":
 "playwright"},
    )
    requests.post(
        profiles_url,
        json={
            "username": "ludwig",
            "name": "Ludwig van Beethoven",
            "title": "composer",
        },
    )
    requests.post(
        profiles_url,
        json={"username": "jane", "name": "Jane Austen", "title": "author"},
    )
    print("Getting the list of profiles from the REST API.")
    profiles = requests.get(profiles_url).json()
    pprint(profiles)
    print(f"Getting just the profile for username 'jane' (URL: {profiles_url}/
jane).")
    jane = requests.get(f"{profiles_url}/jane").json()
    pprint(jane)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDeployment
  - CreateResource
  - CreateRestApi
  - DeleteRestApi
  - GetResources
  - GetRestApis

- PutIntegration
- PutIntegrationResponse
- PutMethod
- PutMethodResponse

Usar o API Gateway para invocar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Lambda Java. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon

API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3).

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.
- Use o pacote Requests para enviar solicitações à API REST.
- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon SNS

AWS contribuições da comunidade para o API Gateway

AWS as contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Exemplos

- Compilar e testar uma aplicação com tecnologia sem servidor

Compilar e testar uma aplicação com tecnologia sem servidor

Os exemplos de código a seguir mostram como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

.NET

SDK para .NET

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK .NET.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Go

SDK para Go V2

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Go.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Java

SDK para Java 2.x

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Java.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Rust

SDK para Rust

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Rust.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda

Exemplos de código para API Gateway Management API usando AWS SDKs

Os exemplos de código a seguir mostram como usar a API de gerenciamento do Amazon API Gateway com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor da API de gerenciamento do API Gateway: mais informações sobre a API de gerenciamento do API Gateway.
- Referência da API de gerenciamento do API Gateway: detalhes sobre todas as ações disponíveis da API de gerenciamento do API Gateway.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos da API Gateway Management API usando AWS SDKs
  - Ações para API Gateway Management API usando AWS SDKs
    - Use PostToConnectioncom um AWS SDK ou CLI

Exemplos básicos da API Gateway Management API usando AWS SDKs

Os exemplos de código a seguir mostram como usar os elementos básicos da API de gerenciamento do Amazon API Gateway com AWS SDKs.

Exemplos

- Ações para API Gateway Management API usando AWS SDKs

- Use PostToConnectioncom um AWS SDK ou CLI

Ações para API Gateway Management API usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais da API API API Gateway Management com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa em Referência da API de gerenciamento do Amazon API Gateway.

Exemplos

- Use PostToConnectioncom um AWS SDK ou CLI

Use **`PostToConnection`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PostToConnection`.

CLI

AWS CLI

Para enviar dados para uma WebSocket conexão

O `post-to-connection` exemplo a seguir envia uma mensagem para um cliente conectado à WebSocket API especificada.

```
aws apigatewaymanagementapi post-to-connection \
    --connection-id L0SM9cOFvHcCIhw= \
    --data "Hello from API Gateway!" \
    --endpoint-url https://aabbccddee.execute-api.us-west-2.amazonaws.com/prod
```

Este comando não produz saída.

Para obter mais informações, consulte Usar os comandos @connections em seu serviço de backend no Guia do desenvolvedor do Amazon API Gateway.

- Para obter detalhes da API, consulte PostToConnectionna Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn send_data(
    client: &aws_sdk_apigatewaymanagement::Client,
    con_id: &str,
    data: &str,
) -> Result<(), aws_sdk_apigatewaymanagement::Error> {
    client
        .post_to_connection()
        .connection_id(con_id)
        .data(Blob::new(data))
        .send()
        .await?;

    Ok(())
}

    let endpoint_url = format!(
        "https://{api_id}.execute-api.{region}.amazonaws.com/{stage}",
        api_id = api_id,
        region = region,
        stage = stage
    );

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let api_management_config = config::Builder::from(&shared_config)
        .endpoint_url(endpoint_url)
        .build();
    let client = Client::from_conf(api_management_config);
```

- Para obter detalhes da API, consulte a PostToConnectionreferência da API AWS SDK for Rust.

Exemplos de código para Application Auto Scaling usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Application Auto Scaling com um kit de desenvolvimento AWS de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário do aplicativo Auto Scaling: mais informações sobre o aplicativo Auto Scaling.
- Referência da API aplicativo Auto Scaling: detalhes sobre todas as ações disponíveis do aplicativo Auto Scaling.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Application Auto Scaling usando AWS SDKs
  - Ações para o Application Auto Scaling usando AWS SDKs
    - Use DeleteScalingPolicycom um AWS SDK ou CLI
    - Usar DeleteScheduledAction com uma CLI
    - Usar DeregisterScalableTarget com uma CLI
    - Usar DescribeScalableTargets com uma CLI
    - Usar DescribeScalingActivities com uma CLI
    - Use DescribeScalingPoliciescom um AWS SDK ou CLI
    - Usar DescribeScheduledActions com uma CLI
    - Usar PutScalingPolicy com uma CLI
    - Usar PutScheduledAction com uma CLI
    - Use RegisterScalableTargetcom um AWS SDK ou CLI

Exemplos básicos de Application Auto Scaling usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Application Auto Scaling com AWS SDKs.

Exemplos

- Ações para o Application Auto Scaling usando AWS SDKs
  - Use DeleteScalingPolicycom um AWS SDK ou CLI
  - Usar DeleteScheduledAction com uma CLI
  - Usar DeregisterScalableTarget com uma CLI
  - Usar DescribeScalableTargets com uma CLI
  - Usar DescribeScalingActivities com uma CLI
  - Use DescribeScalingPoliciescom um AWS SDK ou CLI
  - Usar DescribeScheduledActions com uma CLI
  - Usar PutScalingPolicy com uma CLI
  - Usar PutScheduledAction com uma CLI
  - Use RegisterScalableTargetcom um AWS SDK ou CLI

Ações para o Application Auto Scaling usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Application Auto Scaling com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para conferir uma lista completa, consulte a Referência de API do Application Auto Scaling.

Exemplos

- Use DeleteScalingPolicycom um AWS SDK ou CLI
- Usar DeleteScheduledAction com uma CLI
- Usar DeregisterScalableTarget com uma CLI
- Usar DescribeScalableTargets com uma CLI
- Usar DescribeScalingActivities com uma CLI
- Use DescribeScalingPoliciescom um AWS SDK ou CLI

- Usar DescribeScheduledActions com uma CLI
- Usar PutScalingPolicy com uma CLI
- Usar PutScheduledAction com uma CLI
- Use RegisterScalableTargetcom um AWS SDK ou CLI

Use **`DeleteScalingPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteScalingPolicy`.

CLI

AWS CLI

Para excluir uma política de escalabilidade

Este exemplo exclui uma política de escalabilidade para a aplicação web do serviço Amazon ECS em execução no cluster padrão.

Comando:

```
aws application-autoscaling delete-scaling-policy --policy-name web-app-cpu-lt-25
 --scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-
app --service-namespace ecs
```

- Para obter detalhes da API, consulte DeleteScalingPolicyna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExcep
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReq
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

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

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

            Usage:
               <tableId> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).\s
               policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }

    public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
        try {
            DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
                .policyName(policyName)
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deleteScalingPolicy(delSPRequest);
            System.out.println(policyName +" was deleted successfully.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the scaling policy was deleted
```

```
   public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
       DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
           .scalableDimension(tableWCUs)
           .serviceNamespace(ns)
           .resourceId(tableId)
           .build();

       DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
       System.out.println("DescribeScalableTargets result: ");
       System.out.println(response);
   }

   public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
       try {
           DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
               .scalableDimension(tableWCUs)
               .serviceNamespace(ns)
               .resourceId(tableId)
               .build();

           appAutoScalingClient.deregisterScalableTarget(targetRequest);
           System.out.println("The scalable target was deregistered.");

       } catch (ApplicationAutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
   }

   public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
       DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
           .scalableDimension(tableWCUs)
           .serviceNamespace(ns)
           .resourceIds(tableId)
           .build();
```

```
        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- Para obter detalhes da API, consulte DeleteScalingPolicya Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet exclui a política de escalabilidade especificada para um destino escalável do Application Auto Scaling.

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-
out" -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- Para obter detalhes da API, consulte DeleteScalingPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui a política de escalabilidade especificada para um destino escalável do Application Auto Scaling.

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-
out" -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- Para obter detalhes da API, consulte DeleteScalingPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteScheduledAction`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteScheduledAction`.

CLI

AWS CLI

Para excluir uma ação programada

O `delete-scheduled-action` exemplo a seguir exclui a ação programada especificada da frota Amazon AppStream 2.0 especificada:

```
aws application-autoscaling delete-scheduled-action \
    --service-namespace appstream \
    --scalable-dimension appstream:fleet:DesiredCapacity \
    --resource-id fleet/sample-fleet \
    --scheduled-action-name my-recurring-action
```

Este comando não produz saída.

Para obter mais informações, consulte Escalabilidade programada no Guia do usuário do Application Auto Scaling.

- Para obter detalhes da API, consulte DeleteScheduledActionna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet exclui a ação programada especificada para um destino escalável do Application Auto Scaling.

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteScheduledActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui a ação programada especificada para um destino escalável do Application Auto Scaling.

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteScheduledActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeregisterScalableTarget`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterScalableTarget`.

CLI

AWS CLI

Como cancelar o registro de um destino escalável

Este exemplo cancela o registro de um destino escalável para um serviço do Amazon ECS chamado web-app que está sendo executado no cluster padrão.

Comando:

```
aws application-autoscaling deregister-scalable-target --service-namespace ecs --
scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-app
```

Este exemplo cancela o registro de uma meta escalável para um recurso personalizado. O arquivo custom-resource-id.txt contém uma string que identifica o ID do recurso, que, para um recurso personalizado, é o caminho para o recurso personalizado por meio do endpoint do Amazon API Gateway.

Comando:

```
aws application-autoscaling deregister-scalable-target --
service-namespace custom-resource --scalable-dimension custom-
resource:ResourceType:Property --resource-id file://~/custom-resource-id.txt
```

Conteúdo do arquivo custom-resource-id.txt:

```
https://example.execute-api.us-west-2.amazonaws.com/prod/
scalableTargetDimensions/1-23456789
```

- Para obter detalhes da API, consulte DeregisterScalableTargetna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet cancela o registro de um destino escalável do Application Auto Scaling. O cancelamento do registro de um destino escalável exclui as políticas de escalabilidade associadas a ele.

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeregisterScalableTargetem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cancela o registro de um destino escalável do Application Auto Scaling. O cancelamento do registro de um destino escalável exclui as políticas de escalabilidade associadas a ele.

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeregisterScalableTargetem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeScalableTargets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeScalableTargets`.

CLI

AWS CLI

Para descrever destinos escaláveis

O exemplo `describe-scalable-targets` a seguir descreve os destinos escaláveis para o namespace do serviço `ecs`.

```
aws application-autoscaling describe-scalable-targets \
```

```
    --service-namespace ecs
```

Saída:

```
{
    "ScalableTargets": [
        {
            "ServiceNamespace": "ecs",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "MinCapacity": 1,
            "MaxCapacity": 10,
            "RoleARN": "arn:aws:iam::123456789012:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService",
            "CreationTime": 1462558906.199,
            "SuspendedState": {
                "DynamicScalingOutSuspended": false,
                "ScheduledScalingSuspended": false,
                "DynamicScalingInSuspended": false
            },
            "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
        }
    ]
}
```

Para obter mais informações, consulte Serviços da AWS que podem ser usados com o Application Auto Scaling no Guia do usuário do Application Auto Scaling.

- Para obter detalhes da API, consulte DescribeScalableTargetsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo fornecerá informações sobre os destinos escaláveis do Application Auto Scaling no namespace especificado.

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

Saída:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- Para obter detalhes da API, consulte DescribeScalableTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo fornecerá informações sobre os destinos escaláveis do Application Auto Scaling no namespace especificado.

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

Saída:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- Para obter detalhes da API, consulte DescribeScalableTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeScalingActivities`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeScalingActivities`.

CLI

AWS CLI

Exemplo 1: descrever as atividades de ajuste de escala do serviço do Amazon ECS especificado

O exemplo `describe-scaling-activities` a seguir descreve as atividades de escalabilidade de um serviço do Amazon ECS chamado `web-app` que está sendo executado no cluster `default`. A saída mostra uma atividade de ajuste de escala iniciada por uma política de ajuste de escala.

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace ecs \
    --resource-id service/default/web-app
```

Saída:

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "ecs:service:DesiredCount",
            "Description": "Setting desired count to 1.",
            "ResourceId": "service/default/web-app",
            "ActivityId": "e6c5f7d1-dbbb-4a3f-89b2-51f33e766399",
            "StartTime": 1462575838.171,
            "ServiceNamespace": "ecs",
            "EndTime": 1462575872.111,
            "Cause": "monitor alarm web-app-cpu-lt-25 in state ALARM triggered
 policy web-app-cpu-lt-25",
            "StatusMessage": "Successfully set desired count to 1. Change
 successfully fulfilled by ecs.",
            "StatusCode": "Successful"
        }
    ]
}
```

Para obter mais informações, consulte Atividades de escalabilidade para o Application Auto Scaling no Guia do usuário do Application Auto Scaling.

Exemplo 2: descrever as ações de ajuste de escala da tabela do DynamoDB especificada

O exemplo `describe-scaling-activities` a seguir descreve as atividades de ajuste de escala de uma tabela do DynamoDB chamada `TestTable`. O resultado mostra as atividades de ajuste de escala iniciadas por duas ações agendadas diferentes.

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace dynamodb \
    --resource-id table/TestTable
```

Saída:

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 10.",
            "ResourceId": "table/my-table",
            "ActivityId": "4d1308c0-bbcf-4514-a673-b0220ae38547",
            "StartTime": 1561574415.086,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574449.51,
            "Cause": "maximum capacity was set to 10",
            "StatusMessage": "Successfully set write capacity units to 10. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 5 and max capacity to 10",
            "ResourceId": "table/my-table",
            "ActivityId": "f2b7847b-721d-4e01-8ef0-0c8d3bacc1c7",
            "StartTime": 1561574414.644,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-second-scheduled-action was
 triggered",
            "StatusMessage": "Successfully set min capacity to 5 and max capacity
 to 10",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 15.",
            "ResourceId": "table/my-table",
            "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
```

```
            "StartTime": 1561574108.904,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574140.255,
            "Cause": "minimum capacity was set to 15",
            "StatusMessage": "Successfully set write capacity units to 15. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 15 and max capacity to 20",
            "ResourceId": "table/my-table",
            "ActivityId": "3250fd06-6940-4e8e-bb1f-d494db7554d2",
            "StartTime": 1561574108.512,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-first-scheduled-action was
 triggered",
            "StatusMessage": "Successfully set min capacity to 15 and max
 capacity to 20",
            "StatusCode": "Successful"
        }
    ]
}
```

Para obter mais informações, consulte Atividades de escalabilidade para o Application Auto Scaling no Guia do usuário do Application Auto Scaling.

- Para obter detalhes da API, consulte DescribeScalingActivitiesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: fornece informações descritivas sobre as atividades de escalabilidade no namespace especificado das últimas seis semanas.

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

Saída:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
```

```
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in
 state ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- Para obter detalhes da API, consulte DescribeScalingActivitiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: fornece informações descritivas sobre as atividades de escalabilidade no namespace especificado das últimas seis semanas.

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

Saída:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in
 state ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- Para obter detalhes da API, consulte DescribeScalingActivitiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeScalingPolicies`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeScalingPolicies`.

CLI

AWS CLI

Para descrever políticas de ajuste de escala

Este exemplo de comando descreve as políticas de ajuste de escala para o namespace do serviço ecs.

Comando:

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs
```

Saída:

```
{
    "ScalingPolicies": [
        {
            "PolicyName": "web-app-cpu-gt-75",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "CreationTime": 1462561899.23,
            "StepScalingPolicyConfiguration": {
                "Cooldown": 60,
                "StepAdjustments": [
                    {
                        "ScalingAdjustment": 200,
                        "MetricIntervalLowerBound": 0.0
                    }
                ],
                "AdjustmentType": "PercentChangeInCapacity"
            },
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/web-app-cpu-gt-75",
            "PolicyType": "StepScaling",
            "Alarms": [
                {
```

```
                    "AlarmName": "web-app-cpu-gt-75",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-gt-75"
                }
            ],
            "ServiceNamespace": "ecs"
        },
        {
            "PolicyName": "web-app-cpu-lt-25",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "CreationTime": 1462562575.099,
            "StepScalingPolicyConfiguration": {
                "Cooldown": 1,
                "StepAdjustments": [
                    {
                        "ScalingAdjustment": -50,
                        "MetricIntervalUpperBound": 0.0
                    }
                ],
                "AdjustmentType": "PercentChangeInCapacity"
            },
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/web-app-cpu-lt-25",
            "PolicyType": "StepScaling",
            "Alarms": [
                {
                    "AlarmName": "web-app-cpu-lt-25",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-lt-25"
                }
            ],
            "ServiceNamespace": "ecs"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeScalingPoliciesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet descreve as políticas de escalabilidade do Application Auto Scaling para o namespace de serviço especificado.

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

Saída:

```
Alarms                                  : {Appstream2-LabFleet-default-scale-
out-Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-out
PolicyName                              : default-scale-out
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                  : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-in
PolicyName                              : default-scale-in
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- Para obter detalhes da API, consulte DescribeScalingPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet descreve as políticas de escalabilidade do Application Auto Scaling para o namespace de serviço especificado.

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

Saída:

```
Alarms                                  : {Appstream2-LabFleet-default-scale-
out-Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-out
PolicyName                              : default-scale-out
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                  : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-in
PolicyName                              : default-scale-in
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- Para obter detalhes da API, consulte DescribeScalingPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_policies(client: &Client) -> Result<(), Error> {
    let response = client
        .describe_scaling_policies()
        .service_namespace(ServiceNamespace::Ec2)
        .send()
        .await?;
    println!("Auto Scaling Policies:");
    for policy in response.scaling_policies() {
        println!("{:?}\n", policy);
    }
    println!("Next token: {:?}", response.next_token());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeScalingPoliciesreferência da API AWS SDK for Rust.

Usar **`DescribeScheduledActions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeScheduledActions`.

CLI

AWS CLI

Para descrever ações programadas

O exemplo `describe-scheduled-actions` a seguir exibe detalhes das ações agendadas para o namespace de serviço especificado:

```
aws application-autoscaling describe-scheduled-actions \
    --service-namespace dynamodb
```

Saída:

```
{
    "ScheduledActions": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:35:00)",
            "ResourceId": "table/my-table",
            "CreationTime": 1561571888.361,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-
b290-81db519b227d:resource/dynamodb/table/my-table:scheduledActionName/my-first-
scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 15,
                "MaxCapacity": 20
            },
            "ScheduledActionName": "my-first-scheduled-action",
            "ServiceNamespace": "dynamodb"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:40:00)",
            "ResourceId": "table/my-table",
            "CreationTime": 1561571946.021,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-
b290-81db519b227d:resource/dynamodb/table/my-table:scheduledActionName/my-second-
scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 5,
```

```
                "MaxCapacity": 10
            },
            "ScheduledActionName": "my-second-scheduled-action",
            "ServiceNamespace": "dynamodb"
        }
    ]
}
```

Para obter mais informações, consulte Escalabilidade programada no Guia do usuário do Application Auto Scaling.

- Para obter detalhes da API, consulte DescribeScheduledActionsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet lista as ações programadas para o grupo do Auto Scaling que ainda não foram executadas ou que ainda não atingiram o horário de término.

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

Saída:

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- Para obter detalhes da API, consulte DescribeScheduledActionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet lista as ações programadas para o grupo do Auto Scaling que ainda não foram executadas ou que ainda não atingiram o horário de término.

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

Saída:

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- Para obter detalhes da API, consulte DescribeScheduledActionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutScalingPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutScalingPolicy`.

CLI

AWS CLI

Exemplo 1: como aplicar uma política de escalabilidade com monitoramento do objetivo com uma especificação de métrica predefinida

O exemplo `put-scaling-policy` a seguir aplica uma política de ajuste de escala de rastreamento de destino com uma especificação de métrica predefinida a um serviço do ECS chamado web-app no cluster padrão. A política mantém a utilização média da CPU do serviço em 75%, com períodos de espera de aumento e redução de escala horizontalmente de 60

segundos. A saída contém os ARNs e os nomes dos dois CloudWatch alarmes criados em seu nome.

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
--scalable-dimension ecs:service:DesiredCount \
--resource-id service/default/web-app \
--policy-name cpu75-target-tracking-scaling-policy --policy-
type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

Este exemplo pressupõe que você tenha um arquivo config.json no diretório atual com o seguinte conteúdo:

```
{
     "TargetValue": 75.0,
     "PredefinedMetricSpecification": {
         "PredefinedMetricType": "ECSServiceAverageCPUUtilization"
     },
     "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
}
```

Saída:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/cpu75-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca",
            "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d"
```

```
        }
    ]
}
```

Exemplo 2: como aplicar uma política de escalabilidade com monitoramento do objetivo com uma especificação de métrica personalizada

O exemplo `put-scaling-policy` a seguir aplica uma política de ajuste de escala de rastreamento de destino com uma especificação de métrica personalizada a um serviço do ECS chamado web-app no cluster padrão. A política mantém a utilização média do serviço em 75%, com períodos de espera de aumento e redução de escala horizontalmente de 60 segundos. A saída contém os ARNs e os nomes dos dois CloudWatch alarmes criados em seu nome.

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
--scalable-dimension ecs:service:DesiredCount \
--resource-id service/default/web-app \
--policy-name cms75-target-tracking-scaling-policy
--policy-type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

Este exemplo pressupõe que você tenha um arquivo config.json no diretório atual com o seguinte conteúdo:

```
{
    "TargetValue":75.0,
    "CustomizedMetricSpecification":{
        "MetricName":"MyUtilizationMetric",
        "Namespace":"MyNamespace",
        "Dimensions": [
            {
                "Name":"MyOptionalMetricDimensionName",
                "Value":"MyOptionalMetricDimensionValue"
            }
        ],
        "Statistic":"Average",
        "Unit":"Percent"
    },
    "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
}
```

Saída:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:
 8784a896-b2ba-47a1-b08c-27301cc499a1:resource/ecs/service/default/web-
app:policyName/cms75-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4"
        }
    ]
}
```

Exemplo 3: como aplicar uma política de escalabilidade com monitoramento do objetivo somente para expansão

O exemplo `put-scaling-policy` a seguir aplica uma política de ajuste de escala de rastreamento de destino a um serviço do ECS chamado `web-app` no cluster padrão. A política é usada para aumentar a escala horizontalmente o serviço ECS quando a métrica `RequestCountPerTarget` do Application Load Balancer excede o limite. A saída contém o ARN e o nome do CloudWatch alarme criado em seu nome.

```
aws application-autoscaling put-scaling-policy \
    --service-namespace ecs \
    --scalable-dimension ecs:service:DesiredCount \
    --resource-id service/default/web-app \
    --policy-name alb-scale-out-target-tracking-scaling-policy \
    --policy-type TargetTrackingScaling \
    --target-tracking-scaling-policy-configuration file://config.json
```

Conteúdo de `config.json`:

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
         "PredefinedMetricType": "ALBRequestCountPerTarget",
         "ResourceLabel": "app/EC2Co-EcsEl-1TKLTMITMM0EO/f37c06a68c1748aa/
targetgroup/EC2Co-Defau-LDNM7Q3ZH1ZN/6d4ea56ca2d6a18d"
     },
     "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60,
    "DisableScaleIn": true
}
```

Saída:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/alb-scale-out-target-tracking-scaling-
policy",
    "Alarms": [
        {
            "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca",
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca"
        }
    ]
}
```

Para obter mais informações, consulte Políticas de ajuste de escala de rastreamento de destino para o Application Auto Scaling no Guia do usuário do AWS Application Auto Scaling.

- Para obter detalhes da API, consulte PutScalingPolicyna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet cria ou atualiza uma política para um destino escalável do Application Auto Scaling. Cada destino escalável é identificado por um namespace de serviço, ID de recurso e dimensão escalável.

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

Saída:

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- Para obter detalhes da API, consulte PutScalingPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cria ou atualiza uma política para um destino escalável do Application Auto Scaling. Cada destino escalável é identificado por um namespace de serviço, ID de recurso e dimensão escalável.

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

Saída:

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- Para obter detalhes da API, consulte PutScalingPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutScheduledAction`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutScheduledAction`.

CLI

AWS CLI

Para adicionar uma ação programada a uma tabela do DynamoDB

Este exemplo adiciona uma ação programada a uma tabela do DynamoDB TestTable chamada para escalar de acordo com uma programação recorrente. Na programação especificada (todos os dias às 12h15 UTC), se a capacidade atual estiver abaixo do valor especificado para MinCapacity, o Application Auto Scaling se expande até o valor especificado por. MinCapacity

Comando:

```
aws application-autoscaling put-scheduled-action --service-
namespace dynamodb --scheduled-action-name my-recurring-action --
schedule "cron(15 12 * * ? *)" --resource-id table/TestTable --
scalable-dimension dynamodb:table:WriteCapacityUnits --scalable-target-
action MinCapacity=6
```

Para obter mais informações, consulte Escalabilidade programada no Guia do usuário do Application Auto Scaling.

- Para obter detalhes da API, consulte PutScheduledActionna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet cria ou atualiza uma ação programada para um destino escalável do Application Auto Scaling. Cada destino escalável é identificado por um namespace de serviço, ID de recurso e dimensão escalável.

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- Para obter detalhes da API, consulte PutScheduledActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cria ou atualiza uma ação programada para um destino escalável do Application Auto Scaling. Cada destino escalável é identificado por um namespace de serviço, ID de recurso e dimensão escalável.

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- Para obter detalhes da API, consulte PutScheduledActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`RegisterScalableTarget`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RegisterScalableTarget`.

CLI

AWS CLI

Exemplo 1: registrar um serviço ECS como um destino escalável

O exemplo `register-scalable-target` a seguir inscreve um serviço do Amazon ECS com o Application Auto Scaling. Ele também adiciona uma tag com o nome `environment` e o valor `production` da chave ao destino escalável.

```
aws application-autoscaling register-scalable-target \
    --service-namespace ecs \
    --scalable-dimension ecs:service:DesiredCount \
    --resource-id service/default/web-app \
    --min-capacity 1 --max-capacity 10 \
    --tags environment=production
```

Saída:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

Para obter exemplos de outros AWS serviços e recursos personalizados, consulte os tópicos em AWS serviços que você pode usar com o Application Auto Scaling no Guia do Usuário do Application Auto Scaling.

Exemplo 2: suspender as atividades de escalabilidade de um destino escalável

O exemplo `register-scalable-target` a seguir suspende as atividades de ajuste de escala de um destino escalável existente.

```
aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --suspended-
state DynamicScalingInSuspended=true,DynamicScalingOutSuspended=true,ScheduledScalingSusp
```

Saída:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

Para obter mais informações, consulte Como suspender e retomar o ajuste de escala do Application Auto Scaling no Guia do usuário do Application Auto Scaling.

Exemplo 3: retomar as atividades de escalabilidade para um destino escalável

O exemplo `register-scalable-target` a seguir retoma as atividades de ajuste de escala de um destino escalável existente.

```
aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --suspended-
state DynamicScalingInSuspended=false,DynamicScalingOutSuspended=false,ScheduledScalingSu
```

Saída:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

Para obter mais informações, consulte Como suspender e retomar o ajuste de escala do Application Auto Scaling no Guia do usuário do Application Auto Scaling.

- Para obter detalhes da API, consulte RegisterScalableTargetna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
```

```
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExcep
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResp
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecificati
import
 software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetReque
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicy
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <roleARN> <policyName>\s
```

```
            Where:
               tableId - The table Id value (for example, table/Music).
               roleARN - The ARN of the role that has ApplicationAutoScaling
permissions.
               policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table,
which is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }

    public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
        try {
            RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .roleARN(roleARN)
                .minCapacity(5)
```

```
                .maxCapacity(10)
                .build();

            appAutoScalingClient.registerScalableTarget(targetRequest);
            System.out.println("You have registered " + tableId);

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the target was created.
    public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    // Configure a scaling policy.
    public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
        // Check if the policy exists before creating a new one.
        DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
            .serviceNamespace(ns)
            .resourceId(tableId)
            .scalableDimension(tableWCUs)
            .build());

        if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
            // If policies exist, consider updating an existing policy instead of
creating a new one.
```

```
            System.out.println("Policy already exists. Consider updating it
 instead.");
            List<ScalingPolicy> polList =
 describeScalingPoliciesResponse.scalingPolicies();
            for (ScalingPolicy pol : polList) {
                System.out.println("Policy name:" +pol.policyName());
            }
        } else {
            // If no policies exist, proceed with creating a new policy.
            PredefinedMetricSpecification specification =
 PredefinedMetricSpecification.builder()

 .predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
                .build();

            TargetTrackingScalingPolicyConfiguration policyConfiguration =
 TargetTrackingScalingPolicyConfiguration.builder()
                .predefinedMetricSpecification(specification)
                .targetValue(50.0)
                .scaleInCooldown(60)
                .scaleOutCooldown(60)
                .build();

            PutScalingPolicyRequest putScalingPolicyRequest =
 PutScalingPolicyRequest.builder()
                .targetTrackingScalingPolicyConfiguration(policyConfiguration)
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .policyName(policyName)
                .policyType(PolicyType.TARGET_TRACKING_SCALING)
                .build();

            try {
                appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
                System.out.println("You have successfully created a scaling
 policy for an Application Auto Scaling scalable target");
            } catch (ApplicationAutoScalingException e) {
                System.err.println("Error: " +
 e.awsErrorDetails().errorMessage());
            }
        }
    }
}
```

- Para obter detalhes da API, consulte RegisterScalableTargeta Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet registra ou atualiza um destino escalável. Um destino escalável é um recurso cuja escala pode ser aumentada ou reduzida horizontalmente pelo Application Auto Scaling.

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- Para obter detalhes da API, consulte RegisterScalableTargetem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet registra ou atualiza um destino escalável. Um destino escalável é um recurso cuja escala pode ser aumentada ou reduzida horizontalmente pelo Application Auto Scaling.

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- Para obter detalhes da API, consulte RegisterScalableTargetem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para o Application Recovery Controller usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Route 53 Application Recovery Controller com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Application Recovery Controller: mais informações sobre o Application Recovery Controller.
- Referência da API Application Recovery Controller: detalhes sobre todas as ações disponíveis do Application Recovery Controller.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Application Recovery Controller usando AWS SDKs
  - Ações para o Application Recovery Controller usando AWS SDKs
    - Use GetRoutingControlStatecom um AWS SDK
    - Use UpdateRoutingControlStatecom um AWS SDK

Exemplos básicos do Application Recovery Controller usando AWS SDKs

Os exemplos de código a seguir mostram como usar os elementos básicos do Controlador de Recuperação de Aplicações do Amazon Route 53 com os AWS SDKs.

Exemplos

- Ações para o Application Recovery Controller usando AWS SDKs
  - Use GetRoutingControlStatecom um AWS SDK
  - Use UpdateRoutingControlStatecom um AWS SDK

Ações para o Application Recovery Controller usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Application Recovery Controller com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para uma lista completa, consulte a Referência de API do Controlador de recuperação de aplicações de Amazon Route 53.

Exemplos

- Use GetRoutingControlStatecom um AWS SDK
- Use UpdateRoutingControlStatecom um AWS SDK

Use **`GetRoutingControlState`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetRoutingControlState`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static GetRoutingControlStateResponse
 getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
```

```
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region())).build();
                return client.getRoutingControlState(
                        GetRoutingControlStateRequest.builder()
                                .routingControlArn(routingControlArn).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- Para obter detalhes da API, consulte GetRoutingControlStateа Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
```

```
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def get_routing_control_state(routing_control_arn, cluster_endpoints):
    """
    Gets the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to look up.
    :param cluster_endpoints: The list of cluster endpoints to query.
    :return: The routing control state response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get
 or set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.get_routing_control_state(
                RoutingControlArn=routing_control_arn
            )
            return response
        except Exception as error:
            print(error)
            raise error
```

- Para obter detalhes da API, consulte a GetRoutingControlStateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA lo_exception TYPE REF TO /aws1/cx_rt_generic.
    DATA lo_session TYPE REF TO /aws1/cl_rt_session_base.
    DATA lo_client TYPE REF TO /aws1/if_r5v.
    DATA lt_endpoints TYPE TABLE OF string.
    DATA lv_endpoint TYPE string.
    DATA lv_region TYPE /aws1/rt_region_id.

    " Parse the comma-separated cluster endpoints
    " Expected format: "https://endpoint1.com|us-west-2,https://endpoint2.com|us-
east-1"
    SPLIT iv_cluster_endpoints AT ',' INTO TABLE lt_endpoints.

    " As a best practice, shuffle cluster endpoints to distribute load
    " For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    " For simplicity, we'll try them in order (shuffling can be added if needed)

    " Try each endpoint in order
    LOOP AT lt_endpoints INTO lv_endpoint.
      TRY.
          " Parse endpoint and region from the format "url|region"
          DATA(lv_pos) = find( val = lv_endpoint sub = '|' ).
          IF lv_pos > 0.
            DATA(lv_url) = substring( val = lv_endpoint len = lv_pos ).
            lv_region = substring( val = lv_endpoint off = lv_pos + 1 ).
          ELSE.
            " If no region specified, use default
            lv_url = lv_endpoint.
            lv_region = 'us-east-1'.
          ENDIF.
```

```
        " Create session for this region
        lo_session = /aws1/cl_rt_session_aws=>create( cv_pfl ).

        " Create client with the specific endpoint
        lo_client = create_recovery_client(
          iv_endpoint = lv_url
          iv_region   = lv_region
          io_session  = lo_session ).

        " Try to get the routing control state
        oo_result = lo_client->getroutingcontrolstate(
          iv_routingcontrolarn = iv_routing_control_arn ).

        " If successful, return the result
        RETURN.

      CATCH /aws1/cx_r5vendpttmpyunavailex INTO DATA(lo_endpoint_ex).
        " This endpoint is temporarily unavailable, try the next one
        lo_exception = lo_endpoint_ex.
        CONTINUE.

      CATCH /aws1/cx_r5vaccessdeniedex
            /aws1/cx_r5vinternalserverex
            /aws1/cx_r5vresourcenotfoundex
            /aws1/cx_r5vthrottlingex
            /aws1/cx_r5vvalidationex
            /aws1/cx_rt_generic INTO lo_exception.
        " For other errors, re-raise immediately
        RAISE EXCEPTION lo_exception.
    ENDTRY.
  ENDLOOP.

  " If we get here, all endpoints failed - re-raise the last exception
  IF lo_exception IS BOUND.
    RAISE EXCEPTION lo_exception.
  ENDIF.
```

- Para obter detalhes da API, consulte a GetRoutingControlStatereferência da API AWS SDK for SAP ABAP.

Use **`UpdateRoutingControlState`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `UpdateRoutingControlState`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static UpdateRoutingControlStateResponse
 updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn,
            String routingControlState) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region()))
                        .build();
                return client.updateRoutingControlState(
                        UpdateRoutingControlStateRequest.builder()

 .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- Para obter detalhes da API, consulte UpdateRoutingControlStatea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def update_routing_control_state(
    routing_control_arn, cluster_endpoints, routing_control_state
):
    """
    Updates the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.
```

```
    :param routing_control_arn: The ARN of the routing control to update the
 state for.
    :param cluster_endpoints: The list of cluster endpoints to try.
    :param routing_control_state: The new routing control state.
    :return: The routing control update response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get
 or set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.update_routing_control_state(
                RoutingControlArn=routing_control_arn,
                RoutingControlState=routing_control_state,
            )
            return response
        except Exception as error:
            print(error)
```

- Para obter detalhes da API, consulte a UpdateRoutingControlStateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
```

```
    DATA lo_exception TYPE REF TO /aws1/cx_rt_generic.
    DATA lo_session TYPE REF TO /aws1/cl_rt_session_base.
    DATA lo_client TYPE REF TO /aws1/if_r5v.
    DATA lt_endpoints TYPE TABLE OF string.
    DATA lv_endpoint TYPE string.
    DATA lv_region TYPE /aws1/rt_region_id.

    " Parse the comma-separated cluster endpoints
    " Expected format: "https://endpoint1.com|us-west-2,https://endpoint2.com|us-
east-1"
    SPLIT iv_cluster_endpoints AT ',' INTO TABLE lt_endpoints.

    " As a best practice, shuffle cluster endpoints to distribute load
    " For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    " For simplicity, we'll try them in order (shuffling can be added if needed)

    " Try each endpoint in order
    LOOP AT lt_endpoints INTO lv_endpoint.
      TRY.
          " Parse endpoint and region from the format "url|region"
          DATA(lv_pos) = find( val = lv_endpoint sub = '|' ).
          IF lv_pos > 0.
            DATA(lv_url) = substring( val = lv_endpoint len = lv_pos ).
            lv_region = substring( val = lv_endpoint off = lv_pos + 1 ).
          ELSE.
            " If no region specified, use default
            lv_url = lv_endpoint.
            lv_region = 'us-east-1'.
          ENDIF.

          " Create session for this region
          lo_session = /aws1/cl_rt_session_aws=>create( cv_pfl ).

          " Create client with the specific endpoint
          lo_client = create_recovery_client(
            iv_endpoint = lv_url
            iv_region   = lv_region
            io_session  = lo_session ).

          " Try to update the routing control state
          oo_result = lo_client->updateroutingcontrolstate(
            iv_routingcontrolarn     = iv_routing_control_arn
            iv_routingcontrolstate   = iv_routing_control_state
```

```
            it_safetyrulestooverride = it_safety_rules_override ).

          " If successful, return the result
          RETURN.

        CATCH /aws1/cx_r5vendpttmpyunavailex INTO DATA(lo_endpoint_ex).
          " This endpoint is temporarily unavailable, try the next one
          lo_exception = lo_endpoint_ex.
          CONTINUE.

        CATCH /aws1/cx_r5vaccessdeniedex
              /aws1/cx_r5vconflictexception
              /aws1/cx_r5vinternalserverex
              /aws1/cx_r5vresourcenotfoundex
              /aws1/cx_r5vthrottlingex
              /aws1/cx_r5vvalidationex
              /aws1/cx_rt_generic INTO lo_exception.
          " For other errors, re-raise immediately
          RAISE EXCEPTION lo_exception.
      ENDTRY.
    ENDLOOP.

    " If we get here, all endpoints failed - re-raise the last exception
    IF lo_exception IS BOUND.
      RAISE EXCEPTION lo_exception.
    ENDIF.
```

- Para obter detalhes da API, consulte a UpdateRoutingControlStatereferência da API AWS SDK for SAP ABAP.

Exemplos de código para Athena usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Athena com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Athena — Mais informações sobre o Athena.

- Referência da API Athena — Detalhes sobre todas as ações disponíveis do Athena.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para Athena usando AWS SDKs
  - Introdução à análise de consulta

Cenários para Athena usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Athena com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Athena ou combinadas com outras. Serviços da AWS Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução à análise de consulta

Introdução à análise de consulta

O exemplo de código a seguir mostra como:

- Criar um bucket do S3 para resultados de consulta.
- Criar um banco de dados.
- Criar uma tabela.
- Executar uma consulta.
- Criar e usar consultas nomeadas.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
```

```
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
 [ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
```

```
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi
```

```
# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~
 ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
 2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'

    # Security: Enable S3 bucket encryption with KMS validation
    echo "Enabling default encryption on S3 bucket..."
    if ! aws s3api put-bucket-encryption \
        --bucket "$S3_BUCKET" \
        --server-side-encryption-configuration '{
            "Rules": [{
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }]
        }' 2>&1; then
        echo "Warning: Could not enable encryption on bucket"
    fi

    # Security: Block public access
    echo "Blocking public access to S3 bucket..."
    if ! aws s3api put-public-access-block \
        --bucket "$S3_BUCKET" \
        --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1; then
```

```
        echo "Warning: Could not block public access on bucket"
    fi

    # Security: Enable versioning for data protection
    echo "Enabling versioning on S3 bucket..."
    if ! aws s3api put-bucket-versioning \
        --bucket "$S3_BUCKET" \
        --versioning-configuration Status=Enabled 2>&1; then
        echo "Warning: Could not enable versioning on bucket"
    fi

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
```

```
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
 "$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.
$TABLE_NAME (
  \`Date\` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
```

```
  \"input.regex\" = \"^(?!#)([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s
+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+[^
\\\\(]+[\\\\(]([^\\\\;]+).*\\\\%20([^\\\\/]+)[\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
```

```
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
```

```
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
 "$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
```

```
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID"
 \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
```

```
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
```

```
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
 "$NAMED_QUERY_ID" \
        --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
```

```
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
 | cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Table deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Table deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \
        --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
```

```
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Empty and delete S3 bucket (only if not shared)
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi

        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
        if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
        else
            echo "S3 bucket deleted successfully."
```

```
        fi
    else
        echo "Skipping S3 bucket deletion (shared resource)"
    fi

    # Security: Remove downloaded query results
    if [ -f "./query-results.csv" ]; then
        if command -v shred &>/dev/null; then
            shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - Cp
  - CreateNamedQuery
  - DeleteNamedQuery
  - GetNamedQuery
  - GetQueryExecution
  - GetQueryResults
  - ListDatabases
  - ListNamedQueries
  - ListTableMetadata
  - Mb
  - Rb
  - Rm
  - StartQueryExecution

Exemplos de código para o Audit Manager usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Audit Manager com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Audit Manager: mais informações sobre o Audit Manager.
- Referência da API Audit Manager: detalhes sobre todas as ações disponíveis do Audit Manager.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para o Audit Manager usando AWS SDKs
  - Crie uma estrutura personalizada do Audit Manager a partir de um AWS Config pacote de conformidade usando um AWS SDK
  - Crie uma estrutura personalizada do Audit Manager que contenha controles CSPM do Security Hub usando um AWS SDK
  - Crie um relatório de avaliação do Audit Manager que contenha um dia de evidência usando um AWS SDK

Cenários para o Audit Manager usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Audit Manager com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Audit Manager ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie uma estrutura personalizada do Audit Manager a partir de um AWS Config pacote de conformidade usando um AWS SDK
- Crie uma estrutura personalizada do Audit Manager que contenha controles CSPM do Security Hub usando um AWS SDK
- Crie um relatório de avaliação do Audit Manager que contenha um dia de evidência usando um AWS SDK

Crie uma estrutura personalizada do Audit Manager a partir de um AWS Config pacote de conformidade usando um AWS SDK

O exemplo de código a seguir mostra como:

- Obtenha uma lista de pacotes de AWS Config conformidade.
- Criar um controle personalizado do Audit Manager para cada regra gerenciada em um pacote de conformidade.
- Criar uma estrutura personalizada do Audit Manager que contenha os controles.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class ConformancePack:
    def __init__(self, config_client, auditmanager_client):
```

```
        self.config_client = config_client
        self.auditmanager_client = auditmanager_client

    def get_conformance_pack(self):
        """
        Return a selected conformance pack from the list of conformance packs.

        :return: selected conformance pack
        """
        try:
            conformance_packs = self.config_client.describe_conformance_packs()
            print(
                "Number of conformance packs fetched: ",
                len(conformance_packs.get("ConformancePackDetails")),
            )
            print("Fetched the following conformance packs: ")
            all_cpack_names = {
                cp["ConformancePackName"]
                for cp in conformance_packs.get("ConformancePackDetails")
            }
            for pack in all_cpack_names:
                print(f"\t{pack}")
            cpack_name = input(
                "Provide ConformancePackName that you want to create a custom "
                "framework for: "
            )
            if cpack_name not in all_cpack_names:
                print(f"{cpack_name} is not in the list of conformance packs!")
                print(
                    "Provide a conformance pack name from the available list of "
                    "conformance packs."
                )
                raise Exception("Invalid conformance pack")
            print("-" * 88)
        except ClientError:
            logger.exception("Couldn't select conformance pack.")
            raise
        else:
            return cpack_name

    def create_custom_controls(self, cpack_name):
        """
        Create custom controls for all managed AWS Config rules in a conformance
pack.
```

```
        :param cpack_name: The name of the conformance pack to create controls
for.
        :return: The list of custom control IDs.
        """
        try:
            rules_in_pack =
self.config_client.describe_conformance_pack_compliance(
                ConformancePackName=cpack_name
            )
            print(
                "Number of rules in the conformance pack: ",
                len(rules_in_pack.get("ConformancePackRuleComplianceList")),
            )
            for rule in rules_in_pack.get("ConformancePackRuleComplianceList"):
                print(f"\t{rule.get('ConfigRuleName')}")
            print("-" * 88)
            print(
                "Creating a custom control for each rule and a custom framework "
                "consisting of these rules in Audit Manager."
            )
            am_controls = []
            for rule in rules_in_pack.get("ConformancePackRuleComplianceList"):
                config_rule = self.config_client.describe_config_rules(
                    ConfigRuleNames=[rule.get("ConfigRuleName")]
                )
                source_id = (
                    config_rule.get("ConfigRules")[0]
                    .get("Source", {})
                    .get("SourceIdentifier")
                )
                custom_control = self.auditmanager_client.create_control(
                    name="Config-" + rule.get("ConfigRuleName"),
                    controlMappingSources=[
                        {
                            "sourceName": "ConfigRule",
                            "sourceSetUpOption": "System_Controls_Mapping",
                            "sourceType": "AWS_Config",
                            "sourceKeyword": {
                                "keywordInputType": "SELECT_FROM_LIST",
                                "keywordValue": source_id,
                            },
                        }
                    ],
```

```
                ).get("control", {})
                am_controls.append({"id": custom_control.get("id")})
            print("Successfully created a control for each config rule.")
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom controls.")
            raise
        else:
            return am_controls

    def create_custom_framework(self, cpack_name, am_control_ids):
        """
        Create a custom Audit Manager framework from a selected AWS Config
 conformance
        pack.

        :param cpack_name: The name of the conformance pack to create a framework
 from.
        :param am_control_ids: The IDs of the custom controls created from the
                               conformance pack.
        """
        try:
            print("Creating custom framework...")
            custom_framework =
 self.auditmanager_client.create_assessment_framework(
                name="Config-Conformance-pack-" + cpack_name,
                controlSets=[{"name": cpack_name, "controls": am_control_ids}],
            )
            print(
                f"Successfully created the custom framework: ",
                f"{custom_framework.get('framework').get('name')}: ",
                f"{custom_framework.get('framework').get('id')}",
            )
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom framework.")
            raise


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager custom framework demo!")
    print("-" * 88)
    print(
```

```
        "You can use this sample to select a conformance pack from AWS Config and
 "
        "use AWS Audit Manager to create a custom control for all the managed "
        "rules under the conformance pack. A custom framework is also created "
        "with these controls."
    )
    print("-" * 88)
    conf_pack = ConformancePack(boto3.client("config"),
 boto3.client("auditmanager"))
    cpack_name = conf_pack.get_conformance_pack()
    am_controls = conf_pack.create_custom_controls(cpack_name)
    conf_pack.create_custom_framework(cpack_name, am_controls)


if __name__ == "__main__":
    run_demo()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAssessmentFramework
  - CreateControl

Crie uma estrutura personalizada do Audit Manager que contenha controles CSPM do Security Hub usando um AWS SDK

O exemplo de código a seguir mostra como:

- Obtenha uma lista de todos os controles padrão que têm o Security Hub CSPM como fonte de dados.
- Criar uma estrutura personalizada do Audit Manager que contenha os controles.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class SecurityHub:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_sechub_controls(self):
        """
        Gets the list of controls that use Security Hub as their data source.

        :return: The list of Security Hub controls.
        """
        print("-" * 88)
        next_token = None
        page = 1
        sechub_control_list = []
        while True:
            print("Page [" + str(page) + "]")
            if next_token is None:
                control_list = self.auditmanager_client.list_controls(
                    controlType="Standard", maxResults=100
                )
            else:
                control_list = self.auditmanager_client.list_controls(
                    controlType="Standard", nextToken=next_token, maxResults=100
                )
```

```
            print("Total controls found:",
 len(control_list.get("controlMetadataList")))
            for control in control_list.get("controlMetadataList"):
                control_details = self.auditmanager_client.get_control(
                    controlId=control.get("id")
                ).get("control", {})
                if "AWS Security Hub" in control_details.get("controlSources"):
                    sechub_control_list.append({"id": control_details.get("id")})
            next_token = control_list.get("nextToken")
            if not next_token:
                break
            page += 1
        print("Number of Security Hub controls found: ",
 len(sechub_control_list))
        return sechub_control_list

    def create_custom_framework(self, am_controls):
        """
        Create a custom framework with a list of controls.

        :param am_controls: The list of controls to include in the framework.
        """
        try:
            print("Creating custom framework...")
            custom_framework =
 self.auditmanager_client.create_assessment_framework(
                name="All Security Hub Controls Framework",
                controlSets=[{"name": "Security-Hub", "controls": am_controls}],
            )
            print(
                f"Successfully created the custom framework: "
                f"{custom_framework.get('framework').get('name')}: "
                f"{custom_framework.get('framework').get('id')}"
            )
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom framework.")
            raise


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager Security Hub demo!")
    print("-" * 88)
```

```
    print(" This script creates a custom framework with all Security Hub
 controls.")
    print("-" * 88)
    sechub = SecurityHub(boto3.client("auditmanager"))
    am_controls = sechub.get_sechub_controls()
    sechub.create_custom_framework(am_controls)


if __name__ == "__main__":
    run_demo()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAssessmentFramework
  - GetControl
  - ListControls

Crie um relatório de avaliação do Audit Manager que contenha um dia de evidência usando um AWS SDK

O exemplo de código a seguir mostra como criar um relatório de avaliação do Audit Manager que contenha um dia de evidência.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import dateutil.parser
import logging
import time
import urllib.request
import uuid
```

```
import boto3
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


class AuditReport:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_input(self):
        print("-" * 40)
        try:
            assessment_id = input("Provide assessment id [uuid]: ").lower()
            try:
                assessment_uuid = uuid.UUID(assessment_id)
            except ValueError:
                logger.error("Assessment Id is not a valid UUID: %s",
 assessment_id)
                raise
            evidence_folder = input("Provide evidence date [yyyy-mm-dd]: ")
            try:
                evidence_date = dateutil.parser.parse(evidence_folder).date()
            except ValueError:
                logger.error("Invalid date : %s", evidence_folder)
                raise
            try:
                self.auditmanager_client.get_assessment(
                    assessmentId=str(assessment_uuid)
                )
            except ClientError:
                logger.exception("Couldn't get assessment %s.", assessment_uuid)
                raise
        except (ValueError, ClientError):
            return None, None
        else:
            return assessment_uuid, evidence_date

    def clear_staging(self, assessment_uuid, evidence_date):
        """
        Find all the evidence in the report and clear it.
        """
        next_token = None
```

```
        page = 1
        interested_folder_id_list = []
        while True:
            print(f"Page [{page}]")
            if next_token is None:
                folder_list = (
                    self.auditmanager_client.get_evidence_folders_by_assessment(
                        assessmentId=str(assessment_uuid), maxResults=1000
                    )
                )
            else:
                folder_list = (
                    self.auditmanager_client.get_evidence_folders_by_assessment(
                        assessmentId=str(assessment_uuid),
                        nextToken=next_token,
                        maxResults=1000,
                    )
                )
            folders = folder_list.get("evidenceFolders")
            print(f"Got {len(folders)} folders.")
            for folder in folders:
                folder_id = folder.get("id")
                if folder.get("name") == str(evidence_date):
                    interested_folder_id_list.append(folder_id)
                if folder.get("assessmentReportSelectionCount") == folder.get(
                    "totalEvidence"
                ):
                    print(
                        f"Removing folder from report selection :
{folder.get('name')} "
                        f"{folder_id} {folder.get('controlId')}"
                    )

self.auditmanager_client.disassociate_assessment_report_evidence_folder(
                        assessmentId=str(assessment_uuid),
evidenceFolderId=folder_id
                    )
                elif folder.get("assessmentReportSelectionCount") > 0:
                    # Get all evidence in the folder and
                    # add selected evidence in the selected_evidence_list.
                    evidence_list = (
                        self.auditmanager_client.get_evidence_by_evidence_folder(
                            assessmentId=str(assessment_uuid),
                            controlSetId=folder_id,
```

```
                            evidenceFolderId=folder_id,
                            maxResults=1000,
                        )
                    )
                    selected_evidence_list = []
                    for evidence in evidence_list.get("evidence"):
                        if evidence.get("assessmentReportSelection") == "Yes":
                            selected_evidence_list.append(evidence.get("id"))
                    print(
                        f"Removing evidence report selection :
{folder.get('name')} "
                        f"{len(selected_evidence_list)}"
                    )

self.auditmanager_client.batch_disassociate_assessment_report_evidence(
                        assessmentId=str(assessment_uuid),
                        evidenceFolderId=folder_id,
                        evidenceIds=selected_evidence_list,
                    )
            next_token = folder_list.get("nextToken")
            if not next_token:
                break
            page += 1
        return interested_folder_id_list

    def add_folder_to_staging(self, assessment_uuid, folder_id_list):
        print(f"Adding folders to report : {folder_id_list}")
        for folder in folder_id_list:
            self.auditmanager_client.associate_assessment_report_evidence_folder(
                assessmentId=str(assessment_uuid), evidenceFolderId=folder
            )

    def get_report(self, assessment_uuid):
        report = self.auditmanager_client.create_assessment_report(
            name="ReportViaScript",
            description="testing",
            assessmentId=str(assessment_uuid),
        )
        if self._is_report_generated(report.get("assessmentReport").get("id")):
            report_url = self.auditmanager_client.get_assessment_report_url(
                assessmentReportId=report.get("assessmentReport").get("id"),
                assessmentId=str(assessment_uuid),
            )
            print(report_url.get("preSignedUrl"))
```

```
            urllib.request.urlretrieve(
                report_url.get("preSignedUrl").get("link"),
                report_url.get("preSignedUrl").get("hyperlinkName"),
            )
            print(
                f"Report saved as
 {report_url.get('preSignedUrl').get('hyperlinkName')}."
            )
        else:
            print("Report generation did not finish in 15 minutes.")
            print(
                "Failed to download report. Go to the console and manually
 download "
                "the report."
            )

    def _is_report_generated(self, assessment_report_id):
        max_wait_time = 0
        while max_wait_time < 900:
            print(f"Checking status of the report {assessment_report_id}")
            report_list =
 self.auditmanager_client.list_assessment_reports(maxResults=1)
            if (
                report_list.get("assessmentReports")[0].get("id")
                == assessment_report_id
                and report_list.get("assessmentReports")[0].get("status") ==
 "COMPLETE"
            ):
                return True
            print("Sleeping for 5 seconds...")
            time.sleep(5)
            max_wait_time += 5


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager samples demo!")
    print("-" * 88)
    print(
        "This script creates an assessment report for an assessment with all the
 "
        "evidence collected on the provided date."
    )
    print("-" * 88)
```

```
    report = AuditReport(boto3.client("auditmanager"))
    assessment_uuid, evidence_date = report.get_input()
    if assessment_uuid is not None and evidence_date is not None:
        folder_id_list = report.clear_staging(assessment_uuid, evidence_date)
        report.add_folder_to_staging(assessment_uuid, folder_id_list)
        report.get_report(assessment_uuid)


if __name__ == "__main__":
    run_demo()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AssociateAssessmentReportEvidenceFolder
  - BatchDisassociateAssessmentReportEvidence
  - CreateAssessmentReport
  - DisassociateAssessmentReportEvidenceFolder
  - GetAssessment
  - GetAssessmentReportUrl
  - GetEvidenceByEvidenceFolder
  - GetEvidenceFoldersByAssessment
  - ListAssessmentReports

Exemplos de código para Aurora usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Aurora com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Aurora: mais informações sobre o Aurora.
- Referência da API Aurora: detalhes sobre todas as ações disponíveis do Aurora.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Aurora usando AWS SDKs
  - Olá, Aurora
  - Aprenda os conceitos básicos do Aurora com um AWS SDK
  - Ações para Aurora usando AWS SDKs
    - Use createdBCluster com um AWS SDK
    - Use CreateDBClusterParameterGroupcom um AWS SDK
    - Use CreateDBClusterSnapshotcom um AWS SDK
    - Use createdBInstance com um AWS SDK
    - Use DeleteDBCluster com um AWS SDK
    - Use DeleteDBClusterParameterGroupcom um AWS SDK
    - Use DeletedBInstance com um AWS SDK
    - Use DescribeDBClusterParameterGroupscom um AWS SDK
    - Use DescribeDBClusterParameterscom um AWS SDK
    - Use DescribeDBClusterSnapshotscom um AWS SDK
    - Use DescribedBClusters com um AWS SDK
    - Use DescribeDBEngineVersionscom um AWS SDK
    - Use DescribedBInstances com um AWS SDK
    - Use DescribeOrderableDBInstanceOptionscom um AWS SDK ou CLI
    - Use ModifyDBClusterParameterGroupcom um AWS SDK

- Cenários para o Aurora usando AWS SDKs
  - Criar uma API REST de biblioteca de empréstimos
  - Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Exemplos básicos de Aurora usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Aurora com SDKs da AWS .

Exemplos

- Olá, Aurora
- Aprenda os conceitos básicos do Aurora com um AWS SDK
- Ações para Aurora usando AWS SDKs
  - Use createdBCluster com um AWS SDK
  - Use CreateDBClusterParameterGroupcom um AWS SDK
  - Use CreateDBClusterSnapshotcom um AWS SDK
  - Use createdBInstance com um AWS SDK
  - Use DeleteDBCluster com um AWS SDK
  - Use DeleteDBClusterParameterGroupcom um AWS SDK
  - Use DeletedBInstance com um AWS SDK
  - Use DescribeDBClusterParameterGroupscom um AWS SDK
  - Use DescribeDBClusterParameterscom um AWS SDK
  - Use DescribeDBClusterSnapshotscom um AWS SDK
  - Use DescribedBClusters com um AWS SDK
  - Use DescribeDBEngineVersionscom um AWS SDK
  - Use DescribedBInstances com um AWS SDK
  - Use DescribeOrderableDBInstanceOptionscom um AWS SDK ou CLI
  - Use ModifyDBClusterParameterGroupcom um AWS SDK

Olá, Aurora

Os exemplos de código a seguir mostram como começar a usar o Aurora.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
```

```
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
```

```
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                       # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());
```

```
        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para C++ .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
```

```
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
```

```
# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
```

```
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Rust.

Aprenda os conceitos básicos do Aurora com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um grupo de parâmetros de cluster do banco de dados do Aurora e defina os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Crie um snapshot do cluster do banco de dados e limpe os recursos.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
```

```
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using
 the DescribeDBEngineVersionsAsync method.
    2.  Select an engine family and create a custom DB cluster parameter group
 using the CreateDBClusterParameterGroupAsync method.
    3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
 method.
    4.  Get some parameters in the group using the
 DescribeDBClusterParametersAsync method.
    5.  Parse and display some parameters in the group.
    6.  Modify the auto_increment_offset and auto_increment_increment parameters
        using the ModifyDBClusterParameterGroupAsync method.
    7.  Get and display the updated parameters using the
 DescribeDBClusterParametersAsync method with a source of "user".
    8.  Get a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Create an Aurora DB cluster that contains a MySql database and uses the
 parameter group.
        using the CreateDBClusterAsync method.
    10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
 method.
    11. Display and select from a list of instance classes available for the
 selected engine and version
        using the paginated DescribeOrderableDBInstanceOptions method.
    12. Create a database instance in the cluster using the CreateDBInstanceAsync
 method.
    13. Wait for the DB instance to be ready using the DescribeDBInstances
 method.
```

```
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the
CreateDBClusterSnapshotAsync method.
   16. Wait for DB snapshot to be ready using the
DescribeDBClusterSnapshotsAsync method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync
methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonRDS>()
                   .AddTransient<AuroraWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger<AuroraScenario>();

       auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();

       Console.WriteLine(sepBar);
       Console.WriteLine(
```

```
            "Welcome to the Amazon Aurora: get started with DB clusters
example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await
ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName, parameters);

            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;
```

```
            newInstance = await CreateNewInstance(
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the Aurora DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamilyAsync()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

        Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

        var parameterGroupFamilies =
            engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
        for (var i = 1; i <= parameterGroupFamilies.Count; i++)
        {
            var parameterGroupFamily = parameterGroupFamilies[i - 1];
            // List the available parameter group families.
            Console.WriteLine(
```

```
                $"\t{i}. Family: {parameterGroupFamily.Key}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("2. Select an available DB parameter group family
 by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
 1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
 new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBClusterParameterGroup>
 CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
 {dbParameterGroupFamily}:");

        var parameterGroup = await
 auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
 auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParamet

        Console.WriteLine(
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
```

```
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <param name="parameterNames">Optional specific names of parameters to
 describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>>
 DescribeParametersInGroupAsync(string parameterGroupName, List<string>?
 parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
 parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
```

```
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
 List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await
 auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
 parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
 parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
 group.");

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName,
 "user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }

    /// <summary>
```

```
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
 choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
 dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
 dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
 family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
```

```
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
 cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
 {clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
 clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
 clusterIdentifier);
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
            );

            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
```

```
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
 auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }

        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDBInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await
 auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
```

```
        {
            Console.WriteLine("11. Select an available DB instance class by
 entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateNewInstance(
        string clusterIdentifier,
        string engineName,
        string engineVersion,
        string instanceClass,
        string instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"12. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
 "available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
```

```
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
 instanceIdentifier);
        }
        else
        {
            newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
                clusterIdentifier,
                instanceIdentifier,
                engineName,
                engineVersion,
                instanceClass
            );

            Console.WriteLine("13. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                instances = await
 auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
 "available";
                newInstance = instances.First();
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");
```

```
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
{cluster.DBClusterIdentifier}.");
        var snapshot = await
auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Clean up resources from the scenario.
    /// </summary>
```

```
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
{newCluster.DBClusterIdentifier}.");
            await
auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
newCluster.DBClusterIdentifier);
            }
```

```
            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
 group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
 {parameterGroup.DBClusterParameterGroupName}.");
            await
 auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGr
            Console.WriteLine("Parameter group deleted.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

Métodos de encapsulamento que são chamados pelo cenário para gerenciar as ações do Aurora.

```
using Amazon.RDS;
using Amazon.RDS.Model;

namespace AuroraActions;

/// <summary>
```

```
/// Wrapper for the Amazon Aurora cluster client operations.
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
```

```
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
_amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }

    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }

    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
```

```
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }


    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
```

```
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
```

```
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
```

```
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
```

```
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });
```

```
        return response.DBInstance;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon
 Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
```

```
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " <<
DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB cluster parameter group.
```

```
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter
group."
              << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME,
AUTO_INCREMENT_PREFIX,
                                NO_SOURCE,
                                autoIncrementParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
```

```
                    autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value between ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                                userParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
    else {
        std::cout << "Let's create a DB cluster." << std::endl;
```

```
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters):
");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of engine versions for the parameter group family.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                 client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group family are:"
                  << std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
            std::cout << "  " << index << ": " <<
engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                                1];

        engineName = engineVersion.GetEngine();
        engineVersionName = engineVersion.GetEngineVersion();
        std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB cluster is configured to use your custom cluster
parameter group '"
                  << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;
```

```
        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB cluster to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB cluster to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for cluster to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
```

```
                                DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                         client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
 "",
                                client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
```

```
                  << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }
```

```
        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbCluster);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
        Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
```

```
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for the snapshot to become available." <<
std::endl;

        Aws::RDS::Model::DBClusterSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
```

```
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB cluster, DB instance, and parameter
 group (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
```

```
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
```

```
                                                const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
```

```
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

```
//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
```

```
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);
```

```
            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." <<
std::endl;
            }

            if (clusterDeleting) {
```

```
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups

- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/google/uuid"
)
```

```
// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service
 (Amazon RDS)
// client and create wrappers for the actions used in the scenario.
func NewGetStartedClusters(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB cluster parameter group that is compatible
 with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
```

```
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
```

```
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName,
 updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source
 of 'user'.")
 userParameters, err := scenario.dbClusters.GetParameters(ctx,
 parameterGroupName, "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a
 database
// of a specified type. The database is also configured to use a custom DB
 cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

 log.Println("Checking for an existing DB cluster.")
```

```
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?
\n", engineChoices)
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName,
 dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
   log.Println("Cluster created and available.")
  }
 }
```

```
 log.Println("Cluster data:")
 log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
 log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
 log.Printf("\tStatus: %v\n", *cluster.Status)
 log.Printf("\tEngine: %v\n", *cluster.Engine)
 log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
 log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
 log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
 log.Println(strings.Repeat("-", 88))
 return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
 log.Println("Checking for an existing database instance.")
 dbInstance, err := scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
 if err != nil {
  panic(err)
 }
 if dbInstance == nil {
  log.Println("Let's create a database instance in your DB cluster.")
  log.Println("First, choose a DB instance type:")
  instOpts, err := scenario.dbClusters.GetOrderableInstances(
   ctx, *cluster.Engine, *cluster.EngineVersion)
  if err != nil {
   panic(err)
  }
  var instChoices []string
  for _, opt := range instOpts {
   instChoices = append(instChoices, *opt.DBInstanceClass)
  }
  slices.Sort(instChoices)
  instChoices = slices.Compact(instChoices)
  instIndex := scenario.questioner.AskChoice(
   "Which DB instance class do you want to use?\n", instChoices)
  log.Println("Creating a database instance. This typically takes several
 minutes.")
  dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
```

```
    ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier,
  *cluster.Engine,
    instChoices[instIndex])
   if err != nil {
    panic(err)
   }
   for *dbInstance.DBInstanceStatus != "available" {
    scenario.helper.Pause(30)
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
  *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Println("Instance data:")
  log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
  log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
  log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
  log.Printf("\tEngine: %v\n", *dbInstance.Engine)
  log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
  log.Println(strings.Repeat("-", 88))
  return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster
 and tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
 log.Println(
  "You can now connect to your database using your favorite MySql client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your database cluster. Pass the endpoint,
\n" +
   "port, and administrator user name to 'mysql' and enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
 log.Println("For more information, see the User Guide for Aurora:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
 log.Println(strings.Repeat("-", 88))
}
```

```
// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context,
 clusterName string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster
 parameter group.
// Before the DB cluster parameter group can be deleted, all associated DB
 instances and
// DB clusters must first be deleted.
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {
```

```
 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
  log.Printf("Deleting database instance %v.\n",
 *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}
```

```
// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Aurora.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}


// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
```

```
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
```

```
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

```
// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
```

```
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}
```

```
// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
```

```
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
```

```
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {
```

```
 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances

- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
```

```
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
                "    dbClusterGroupName - The name of the DB cluster parameter
 group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group
 family name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster
 identifier value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n"
 +
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials\"\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
 dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)
```

```
.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
               .build();
   }

   private static String getSecretValues(String secretName) {
       SecretsManagerClient secretClient = getSecretClient();
       GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
               .secretId(secretName)
               .build();

       GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
       return valueResponse.secretString();
   }

   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(clusterDBARN) == 0) {
                       System.out.println(clusterDBARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
```

```
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();

            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
```

```
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
            String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }
```

```
            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
```

```
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

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

```
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
```

```
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify the auto_increment_offset parameter.
    public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
```

```
            ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dClusterGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
            System.out.println(
                    "The parameter group " +
response.dbClusterParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
```

```
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
```

```
    }

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
```

```
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
```

```
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
 User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()
```

```
    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
 dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)

    println("10. Get a list of instance classes available for the selected
 engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
 dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)
```

```
    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
```

```
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
```

```
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
```

```
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
```

```
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
```

```
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
```

```
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.

- CreateDBCluster
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBInstance
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
```

```
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database
engine and
        create a DB cluster parameter group that is compatible with a selected
engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created
parameter group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
{parameter_group_name}."
        )
        parameter_group =
self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group
{parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
```

```
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster
parameter group.
```

```
        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the
DB cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
```

```
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new
DB cluster
        contains no DB instances, so you must add one. The first DB instance that
is added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"] + ", storage type: " +
opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
minutes."
            )
```

```
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on
 how to
        connect to it.

        :param cluster: The DB cluster to display.
        """
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
        )
        print("-" * 88)
```

```
    def create_snapshot(self, cluster_name):
        """
        Shows how to create a DB cluster snapshot and wait until it's available.

        :param cluster_name: The name of a DB cluster to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB cluster (y/n)? ",
q.is_yesno
        ):
            snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
few minutes."
            )
            snapshot = self.aurora_wrapper.create_cluster_snapshot(
                snapshot_id, cluster_name
            )
            while snapshot.get("Status") != "available":
                wait(30)
                snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)

    def cleanup(self, db_inst, cluster, parameter_group):
        """
        Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
        Before the DB cluster parameter group can be deleted, all associated DB
instances and
        DB clusters must first be deleted.

        :param db_inst: The DB instance to delete.
        :param cluster: The DB cluster to delete.
        :param parameter_group: The DB cluster parameter group to delete.
        """
        cluster_name = cluster["DBClusterIdentifier"]
        parameter_group_name = parameter_group["DBClusterParameterGroupName"]
        if q.ask(
            "\nDo you want to delete the database instance, DB cluster, and
parameter "
            "group (y/n)? ",
            q.is_yesno,
```

```
        ):
            print(f"Deleting database instance
{db_inst['DBInstanceIdentifier']}.")

self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, cluster_name,
db_name):
        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)
```

```
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Aurora.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.
```

```
        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
```

```
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
```

```
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
```

```
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

```
    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
```

```
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
```

```
                            This must be compatible with the configured parameter
group
                            of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
```

```
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

```
    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Uma biblioteca contendo as funções específicas para o cenário do Aurora.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str =
 "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
struct MetadataError {
    message: Option<String>,
```

```
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
```

```
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display
 them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
            f,
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
```

```
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
 ScenarioError> {
        let describe_db_engine_versions =
 self.rds.describe_db_engine_versions(DB_ENGINE).await;
        trace!(versions=?describe_db_engine_versions, "full list of versions");

        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
```

```
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
```

```
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }
```

```
        Ok(())
    }

    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
 Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
```

```
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
           .collect::<Vec<_>>();

       Ok(parameters)
   }

   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
```

```
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
    // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
    // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
    // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
    pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
        if self.password.is_none() {
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
```

```
            ));
        }
        let create_db_cluster = self
            .rds
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
```

```
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
```

```
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
```

```
   // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
Status == 'available'.
   pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
       let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
       let snapshot = self
           .rds
           .snapshot_cluster(id, format!("{id}_{name}").as_str())
           .await;
       match snapshot {
           Ok(output) => match output.db_cluster_snapshot {
               Some(snapshot) => Ok(snapshot),
               None => Err(ScenarioError::with("Missing Snapshot")),
           },
           Err(err) => Err(ScenarioError::new("Failed to create snapshot",
&err)),
       }
   }

   pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
       let mut clean_up_errors: Vec<ScenarioError> = vec![];

       // Delete the instance. rds.DeleteDbInstance.
       let delete_db_instance = self
           .rds
           .delete_db_instance(
               self.db_instance_identifier
                   .as_deref()
                   .expect("instance identifier"),
           )
           .await;
       if let Err(err) = delete_db_instance {
           let identifier = self
               .db_instance_identifier
               .as_deref()
               .unwrap_or("Missing Instance Identifier");
           let message = format!("failed to delete db instance {identifier}");
           clean_up_errors.push(ScenarioError::new(message, &err));
       } else {
           // Wait for the instance to delete
           let waiter = Waiter::default();
           while waiter.sleep().await.is_ok() {
               let describe_db_instances =
self.rds.describe_db_instances().await;
```

```
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;
```

```
        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
```

```
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
 rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}

#[cfg(test)]
pub mod tests;
```

Testes da biblioteca usando automocks em torno do wrapper do RDS Client.

```
use crate::rds::MockRdsImpl;
```

```
use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
```

```
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
```

```
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
```

```
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
```

```
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
```

```
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}
```

```
#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                   ErrorKind::Other,
                   "modify_db_cluster_parameter_group_error",
               ))),
               Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
```

```
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
```

```
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()
```

```
 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
```

```
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
```

```
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
```

```
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
```

```
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")

 .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}
```

```
#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _|
 Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Um binário para executar o cenário do início ao fim usando o inquiridor para que o usuário possa tomar algumas decisões.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }
```

```
    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to
 the Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario,
 anyhow::Error> {
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
```

```
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
 rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
    )?;

    let set_engine = scenario.set_engine(engine.as_str(),
 version.as_str()).await;
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for
 engine: {err}")),
    }

    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
```

```
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings)
 -> Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings,
 "auto_increment_increment", 3);

    // Modify both the auto_increment_offset and auto_increment_increment
 parameters in one call in the custom parameter group. Set their ParameterValue
 fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }

    // Get and display the updated parameters. Specify Source of 'user' to get
 just the modified parameters. rds.DescribeDbClusterParameters(Source='user')
    show_parameters(scenario, warnings).await;

    let username = inquire::Text::new("Username for the database (default
 'testuser')")
        .with_default("testuser")
        .with_initial_value("testuser")
        .prompt();

    if let Err(error) = username {
        warnings.push(
            "Failed to get username, using default",
            ScenarioError::with(format!("Error from inquirer: {error}")),
        );
        return Err(());
    }
    let username = username.unwrap();

    let password = inquire::Text::new("Password for the database (minimum 8
 characters)")
        .with_validator(|i: &str| {
            if i.len() >= 8 {
                Ok(inquire::validator::Validation::Valid)
            } else {
```

```
                Ok(inquire::validator::Validation::Invalid(
                    "Password must be at least 8 characters".into(),
                ))
            }
        })
        .prompt();

    let password: Option<SecretString> = match password {
        Ok(password) => Some(SecretString::from(password)),
        Err(error) => {
            warnings.push(
                "Failed to get password, using none (and not starting a DB)",
                ScenarioError::with(format!("Error from inquirer: {error}")),
            );
            return Err(());
        }
    };

    scenario.set_login(Some(username), password);

    Ok(())
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError>
 {
    // Create an Aurora DB cluster database cluster that contains a MySql
 database and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
 for DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
 Status == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
```

```
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;

    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();

    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the
 instance and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
```

```
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
    let parameters = scenario.cluster_parameters().await;

    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) ->
 u8 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
```

```
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Um wrapper do serviço Amazon RDS que permite automocking para testes.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError,
 DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
```

```
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;

#[cfg(test)]
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
```

```
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
```

```
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
```

```
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
        &self,
```

```
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
        &self,
```

```
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }
}
```

O Cargo.toml com dependências usadas nesse cenário.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-smithy-types = { version = "1.0.1" }
```

```
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Ações para Aurora usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Aurora com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Aurora, e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Aurora usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de API do Amazon Aurora.

Exemplos

- Use createdBCluster com um AWS SDK
- Use CreateDBClusterParameterGroupcom um AWS SDK
- Use CreateDBClusterSnapshotcom um AWS SDK
- Use createdBInstance com um AWS SDK
- Use DeleteDBCluster com um AWS SDK
- Use DeleteDBClusterParameterGroupcom um AWS SDK
- Use DeletedBInstance com um AWS SDK
- Use DescribeDBClusterParameterGroupscom um AWS SDK
- Use DescribeDBClusterParameterscom um AWS SDK
- Use DescribeDBClusterSnapshotscom um AWS SDK
- Use DescribedBClusters com um AWS SDK
- Use DescribeDBEngineVersionscom um AWS SDK
- Use DescribedBInstances com um AWS SDK
- Use DescribeOrderableDBInstanceOptionscom um AWS SDK ou CLI
- Use ModifyDBClusterParameterGroupcom um AWS SDK

Use **`createdBCluster`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDBCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create a new cluster and database.
/// </summary>
/// <param name="dbName">The name of the new database.</param>
/// <param name="clusterIdentifier">The identifier of the cluster.</param>
/// <param name="parameterGroupName">The name of the parameter group.</param>
/// <param name="dbEngine">The engine to use for the new cluster.</param>
/// <param name="dbEngineVersion">The version of the engine to use.</param>
/// <param name="adminName">The admin username.</param>
/// <param name="adminPassword">The primary admin password.</param>
/// <returns>The cluster object.</returns>
public async Task<DBCluster> CreateDBClusterWithAdminAsync(
    string dbName,
    string clusterIdentifier,
    string parameterGroupName,
    string dbEngine,
    string dbEngineVersion,
    string adminName,
    string adminPassword)
{
    var request = new CreateDBClusterRequest
    {
        DatabaseName = dbName,
        DBClusterIdentifier = clusterIdentifier,
        DBClusterParameterGroupName = parameterGroupName,
        Engine = dbEngine,
        EngineVersion = dbEngineVersion,
        MasterUsername = adminName,
        MasterUserPassword = adminPassword,
    };

    var response = await _amazonRDS.CreateDBClusterAsync(request);
```

```
        return response.DBCluster;
    }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
```

```
            return false;
        }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para C++ .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
```

```
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
           String dbClusterIdentifier, String userName, String password) {
       try {
           CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                   .databaseName(dbName)
                   .dbClusterIdentifier(dbClusterIdentifier)
```

```
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
 rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
```

```
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
```

```
        return cls(rds_client)


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
```

```
        if self.password.is_none() {
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
            ));
        }
        let create_db_cluster = self
            .rds
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
```

```
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
```

```
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
```

```
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });
```

```
    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
```

```
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
```

```
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
```

```
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Rust.

Use **`CreateDBClusterParameterGroup`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDBClusterParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
 _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- Para obter detalhes da API, consulte CreateDBClusterParameterGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte CreateDBClusterParameterGroupa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

- Para obter detalhes da API, consulte CreateDBClusterParameterGroupa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
           String dbParameterGroupFamily) {
       try {
           CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CreateDBClusterParameterGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Para obter detalhes da API, consulte a CreateDBClusterParameterGroupreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
 parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
 group.
        :param parameter_group_family: The family that is used as the basis of
 the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateDBClusterParameterGroupReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
       let create_db_cluster_parameter_group = self
           .rds
           .create_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
               engine,
           )
           .await;

       match create_db_cluster_parameter_group {
           Ok(CreateDbClusterParameterGroupOutput {
```

```
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
 do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
 SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
```

```
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- Para obter detalhes da API, consulte a CreateDBClusterParameterGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_output) = lo_rds->createdbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
      iv_dbparametergroupfamily = iv_param_group_family
      iv_description = iv_description
    ).
    oo_result = lo_output->get_dbclusterparametergroup( ).
  CATCH /aws1/cx_rdsdbparmgralrexfault.
    " Re-raise exception - parameter group already exists
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbparmgralrexfault.
  CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
    " Re-raise exception - quota exceeded
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrquotaexcd00.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDBClusterParameterGroupreferência da API AWS SDK for SAP ABAP.

Use **`CreateDBClusterSnapshot`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDBClusterSnapshot`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }
```

- Para obter detalhes da API, consulte CreateDBClusterSnapshota Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Para obter detalhes da API, consulte CreateDBClusterSnapshota Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
```

```
}
```

- Para obter detalhes da API, consulte CreateDBClusterSnapshota Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
           CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
           System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CreateDBClusterSnapshota Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Para obter detalhes da API, consulte a CreateDBClusterSnapshotreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
 DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Para obter detalhes da API, consulte a CreateDBClusterSnapshotReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
```

```
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
```

```
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }
```

```
            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
```

```
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
```

```
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
```

```
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
```

```
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
```

```
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
```

```
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte a CreateDBClusterSnapshotreferência da API AWS SDK for Rust.

Use **`createdBInstance`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDBInstance`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
```

```
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
```

```
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
 "",
                             client);
            return false;
        }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para C++ .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createDBInstanceCluster(RdsClient rdsClient,
           String dbInstanceIdentifier,
           String dbInstanceClusterIdentifier,
           String instanceClass) {
       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .engine("aurora-mysql")
                   .dbInstanceClass(instanceClass)
                   .build();

           CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
           System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
           return response.dbInstance().dbInstanceArn();

       } catch (RdsException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
       return "";
   }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
 that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
 created.
        :param db_engine: The database engine of a database to create in the DB
 instance.
```

```
                            This must be compatible with the configured parameter
group
                            of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
```

```
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });
```

```
    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
```

```
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
```

```
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
```

```
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
```

```
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Rust.

Use **`DeleteDBCluster`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteDBCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para C++ .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
```

```
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
```

```
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
```

```
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
```

```
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }
```

```
        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
```

```
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
```

```
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
```

```
                    "describe db instances error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Rust.

Use **`DeleteDBClusterParameterGroup`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteDBClusterParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteDBClusterParameterGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(parameterGroupName);

    Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
            client.DeleteDBClusterParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully deleted."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
```

- Para obter detalhes da API, consulte DeleteDBClusterParameterGroupa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obter detalhes da API, consulte DeleteDBClusterParameterGroupa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(clusterDBARN) == 0) {
                       System.out.println(clusterDBARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
```

```
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
 DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


 rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteDBClusterParameterGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
```

```
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteDBClusterParameterGroupreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteDBClusterParameterGroupReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
```

```
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
```

```
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
```

```
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
```

```
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
```

```
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Cluster
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
```

```
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte a DeleteDBClusterParameterGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_rds->deletedbclusterparamgroup(
```

```
          iv_dbclusterparamgroupname = iv_param_group_name
        ).
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        " Re-raise exception - parameter group not found
        RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        " Re-raise exception - invalid state
        RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDBClusterParameterGroupreferência da API AWS SDK for SAP ABAP.

Use **`DeletedBInstance com`** um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteDBInstance`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
```

```
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
```

```
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
 parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para C++ .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
```

```
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
```

```
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
```

```
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
```

```
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
```

```
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }
```

```
        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
```

```
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
```

```
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
```

```
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Rust.

Use **`DescribeDBClusterParameterGroups`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeDBClusterParameterGroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterParameterGroupsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte DescribeDBClusterParameterGroupsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
```

```
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- Para obter detalhes da API, consulte DescribeDBClusterParameterGroupsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterParameterGroupsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameterGroupsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameterGroupsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_output) = lo_rds->describedbclusterparamgroups(
          iv_dbclusterparamgroupname = iv_param_group_name
        ).
        DATA(lt_param_groups) = lo_output->get_dbclusterparametergroups( ).
        IF lines( lt_param_groups ) > 0.
          oo_result = lt_param_groups[ 1 ].
        ENDIF.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameterGroupsreferência da API AWS SDK for SAP ABAP.

Use **`DescribeDBClusterParameters`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeDBClusterParameters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
```

```
        do
        {
            response = await
 _amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterParametersa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
```

```
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String
 &parameterGroupName,
                                             const Aws::String &namePrefix,
                                             const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                             const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Para obter detalhes da API, consulte DescribeDBClusterParametersa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
```

```
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Para obter detalhes da API, consulte DescribeDBClusterParametersa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset
or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                   System.out.println("*** The parameter name is  " + paraName);
                   System.out.println("*** The parameter value is  " +
para.parameterValue());
                   System.out.println("*** The parameter data type is " +
para.dataType());
                   System.out.println("*** The parameter description is " +
para.description());
                   System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
               }
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterParametersa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
```

```
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeDBClusterParametersreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return parameters
```

- Para obter detalhes da API, consulte a DescribeDBClusterParametersReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }
```

```
        let parameters = parameters_output
            .unwrap()
            .into_iter()
            .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
            .filter(|p|
 FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
            .map(AuroraScenarioParameter::from)
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
 SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
```

```
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- Para obter detalhes da API, consulte a DescribeDBClusterParametersreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA lv_marker TYPE /aws1/rdsstring VALUE ''.
        DATA lt_all_parameters TYPE /aws1/cl_rdsparameter=>tt_parameterslist.

        DO.
          DATA(lo_output) = lo_rds->describedbclusterparameters(
            iv_dbclusterparamgroupname = iv_param_group_name
            iv_source = iv_source
            iv_marker = lv_marker
          ).

          LOOP AT lo_output->get_parameters( ) INTO DATA(lo_param).
            IF iv_name_prefix IS INITIAL OR
               lo_param->get_parametername( ) CP |{ iv_name_prefix }*|.
              APPEND lo_param TO lt_all_parameters.
            ENDIF.
          ENDLOOP.

          lv_marker = lo_output->get_marker( ).
          IF lv_marker IS INITIAL.
            EXIT.
          ENDIF.
        ENDDO.

        ot_parameters = lt_all_parameters.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
```

```
        " Re-raise exception - parameter group not found
        RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBClusterParametersreferência da API AWS SDK for SAP ABAP.

Use **`DescribeDBClusterSnapshots`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeDBClusterSnapshots`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
```

```
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterSnapshotsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
```

```
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Para obter detalhes da API, consulte DescribeDBClusterSnapshotsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- Para obter detalhes da API, consulte DescribeDBClusterSnapshotsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void waitForSnapshotReady(RdsClient rdsClient, String
 dbSnapshotIdentifier,
            String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
 DescribeDbClusterSnapshotsRequest.builder()
```

```
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterSnapshotsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Para obter detalhes da API, consulte a DescribeDBClusterSnapshotsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Para obter detalhes da API, consulte a DescribeDBClusterSnapshotsReferência da API AWS SDK for Python (Boto3).

Use **`DescribedBClusters`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeDBClusters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
```

```
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para C++ .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
```

```
    err = nil
   } else {
    log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
   }
   return nil, err
  } else {
   return &output.DBClusters[0], err
  }
 }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void describeDbClusterParameters(RdsClient rdsClient, String
 dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
 DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
 DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }
```

```
            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
```

```
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
```

```
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
```

```
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Rust.

Use **`DescribeDBEngineVersions`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeDBEngineVersions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersionsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
 list of
                                        engine versions to those that are
 compatible with
                                        this parameter group family.
        :return: The list of database engine versions.
        """
```

```
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersionsReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");
```

```
        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }
```

```
#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}
```

```
#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersionsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " iv_engine                 = 'mysql'
   " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter
group family)
   TRY.
       oo_result = lo_rds->describedbengineversions(
         iv_engine                 = iv_engine
         iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
       DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
       MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersionsreferência da API AWS SDK for SAP ABAP.

Use **`DescribedBInstances`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeDBInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
```

```
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
 dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
```

```
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para C++ .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
```

```
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Waits until the database instance is available.
   public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .build();

           while (!instanceReady) {
               DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
               List<DBCluster> clusterList = response.dbClusters();
               for (DBCluster cluster : clusterList) {
                   instanceReadyStr = cluster.status();
                   if (instanceReadyStr.contains("available")) {
```

```
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
```

```
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
```

```
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe db clusters error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Cluster
    tokio::time::resume();
```

```
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Rust.

Use **`DescribeOrderableDBInstanceOptions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeOrderableDBInstanceOptions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
```

```
        var paginateInstanceOptions =
 _amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
 paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
```

```
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
 AuroraClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo lista as versões do mecanismo de banco de dados compatíveis com uma classe de instância de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Exemplo 2: este exemplo lista as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo lista as versões do mecanismo de banco de dados compatíveis com uma classe de instância de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Exemplo 2: este exemplo lista as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptionsReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
       let describe_orderable_db_instance_options_items = self
           .rds
           .describe_orderable_db_instance_options(
               DB_ENGINE,
               self.engine_version
                   .as_ref()
                   .expect("engine version for db instance options")
                   .as_str(),
           )
           .await;

       describe_orderable_db_instance_options_items
           .map(|options| {
               options
                   .iter()
                   .filter(|o| o.storage_type() == Some("aurora"))
                   .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                   .collect::<Vec<String>>()
           })
           .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
   }

   pub async fn describe_orderable_db_instance_options(
       &self,
       engine: &str,
       engine_version: &str,
   ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
```

```
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
```

```
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;
```

```
    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptionsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_engine        = 'mysql'
    " iv_engineversion = '8.0.35'
    TRY.
        oo_result = lo_rds->descrorderabledbinstoptions(
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result-
>get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.|
 TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptionsreferência da API AWS SDK for SAP ABAP.

Use **`ModifyDBClusterParameterGroup`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ModifyDBClusterParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
```

```
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- Para obter detalhes da API, consulte ModifyDBClusterParameterGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
 modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte ModifyDBClusterParameterGroupa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obter detalhes da API, consulte ModifyDBClusterParameterGroupa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
 String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
 DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();
```

```
            List<DBClusterParameterGroup> groups =
 rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
 group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
 group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ModifyDBClusterParameterGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
```

```
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Para obter detalhes da API, consulte a ModifyDBClusterParameterGroupreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a ModifyDBClusterParameterGroupReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               vec![
                   Parameter::builder()
                       .parameter_name("auto_increment_offset")
                       .parameter_value(format!("{offset}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
                   Parameter::builder()
                       .parameter_name("auto_increment_increment")
                       .parameter_value(format!("{increment}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
               ],
           )
           .await;

       if let Err(error) = modify_db_cluster_parameter_group {
           return Err(ScenarioError::new(
               "Failed to modify cluster parameter group",
               &error,
           ));
```

```
        }

        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
 SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
```

```
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}
```

- Para obter detalhes da API, consulte a ModifyDBClusterParameterGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_rds->modifydbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
      it_parameters = it_update_parameters
    ).
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    " Re-raise exception - parameter group not found
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
  CATCH /aws1/cx_rdsinvdbprmgrstatef00.
    " Re-raise exception - invalid state
    RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
ENDTRY.
```

- Para obter detalhes da API, consulte a ModifyDBClusterParameterGroupreferência da API AWS SDK for SAP ABAP.

Cenários para o Aurora usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Aurora com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Aurora ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma API REST de biblioteca de empréstimos

- Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice para criar uma API REST apoiada por um banco de dados Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Use AWS Secrets Manager para gerenciar as credenciais do banco de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST sem servidor no Amazon API Gateway e. AWS Lambda
- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Os exemplos de código a seguir mostram como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

.NET

SDK para .NET

Mostra como usar o AWS SDK para .NET para criar um aplicativo web que rastreia itens de trabalho em um banco de dados Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful .NET.

- Integre um aplicativo web React com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

C++

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar uma API REST C++ que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Java

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar um aplicativo web que rastreia itens de trabalho em um banco de dados Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um Node.js back-end Express.

- Integre um aplicativo React.js web com Serviços da AWS o.
- Liste, adicione e atualize itens em uma tabela do Aurora.

- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

PHP

SDK para PHP

Mostra como usar o AWS SDK para PHP para criar uma aplicação web que rastreia itens de trabalho em um banco de dados do Amazon RDS e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end PHP RESTful.

- Integre um aplicativo React.js da web com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um AWS Secrets Manager segredo que contenha as credenciais do banco de dados e use-o para autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS

- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos de código para Auto Scaling usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon EC2 Auto Scaling AWS com um kit de desenvolvimento de software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Auto Scaling: mais informações sobre o Auto Scaling.
- Referência da API Auto Scaling: detalhes sobre todas as ações disponíveis do Auto Scaling.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Auto Scaling usando AWS SDKs
  - Olá, Auto Scaling
  - Aprenda as noções básicas do Auto Scaling com um AWS SDK
  - Ações para Auto Scaling usando AWS SDKs
    - Usar AttachInstances com uma CLI
    - Use AttachLoadBalancerTargetGroupscom um AWS SDK ou CLI
    - Usar AttachLoadBalancers com uma CLI
    - Usar CompleteLifecycleAction com uma CLI

- Use CreateAutoScalingGroupcom um AWS SDK ou CLI
- Usar CreateLaunchConfiguration com uma CLI
- Usar CreateOrUpdateTags com uma CLI
- Use DeleteAutoScalingGroupcom um AWS SDK ou CLI
- Usar DeleteLaunchConfiguration com uma CLI
- Usar DeleteLifecycleHook com uma CLI
- Usar DeleteNotificationConfiguration com uma CLI
- Usar DeletePolicy com uma CLI
- Usar DeleteScheduledAction com uma CLI
- Usar DeleteTags com uma CLI
- Usar DescribeAccountLimits com uma CLI
- Usar DescribeAdjustmentTypes com uma CLI
- Use DescribeAutoScalingGroupscom um AWS SDK ou CLI
- Use DescribeAutoScalingInstancescom um AWS SDK ou CLI
- Usar DescribeAutoScalingNotificationTypes com uma CLI
- Usar DescribeLaunchConfigurations com uma CLI
- Usar DescribeLifecycleHookTypes com uma CLI
- Usar DescribeLifecycleHooks com uma CLI
- Usar DescribeLoadBalancers com uma CLI
- Usar DescribeMetricCollectionTypes com uma CLI
- Usar DescribeNotificationConfigurations com uma CLI
- Usar DescribePolicies com uma CLI
- Use DescribeScalingActivitiescom um AWS SDK ou CLI
- Usar DescribeScalingProcessTypes com uma CLI
- Usar DescribeScheduledActions com uma CLI
- Usar DescribeTags com uma CLI
- Usar DescribeTerminationPolicyTypes com uma CLI
- Usar DetachInstances com uma CLI
- Usar DetachLoadBalancers com uma CLI

- Use DisableMetricsCollectioncom um AWS SDK ou CLI

    - Use EnableMetricsCollectioncom um AWS SDK ou CLI
    - Usar EnterStandby com uma CLI
    - Usar ExecutePolicy com uma CLI
    - Usar ExitStandby com uma CLI
    - Usar PutLifecycleHook com uma CLI
    - Usar PutNotificationConfiguration com uma CLI
    - Usar PutScalingPolicy com uma CLI
    - Usar PutScheduledUpdateGroupAction com uma CLI
    - Usar RecordLifecycleActionHeartbeat com uma CLI
    - Usar ResumeProcesses com uma CLI
    - Use SetDesiredCapacitycom um AWS SDK ou CLI
    - Usar SetInstanceHealth com uma CLI
    - Usar SetInstanceProtection com uma CLI
    - Usar SuspendProcesses com uma CLI
    - Use TerminateInstanceInAutoScalingGroupcom um AWS SDK ou CLI
    - Use UpdateAutoScalingGroupcom um AWS SDK ou CLI
- Cenários para o Auto Scaling usando AWS SDKs
  - Crie e gerencie um serviço resiliente usando um AWS SDK

Exemplos básicos de Auto Scaling usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon EC2 Auto Scaling com SDK AWS .

Exemplos

- Olá, Auto Scaling
- Aprenda as noções básicas do Auto Scaling com um AWS SDK
- Ações para Auto Scaling usando AWS SDKs
  - Usar AttachInstances com uma CLI
  - Use AttachLoadBalancerTargetGroupscom um AWS SDK ou CLI
  - Usar AttachLoadBalancers com uma CLI
  - Usar CompleteLifecycleAction com uma CLI

- Use CreateAutoScalingGroupcom um AWS SDK ou CLI
- Usar CreateLaunchConfiguration com uma CLI
- Usar CreateOrUpdateTags com uma CLI
- Use DeleteAutoScalingGroupcom um AWS SDK ou CLI
- Usar DeleteLaunchConfiguration com uma CLI
- Usar DeleteLifecycleHook com uma CLI
- Usar DeleteNotificationConfiguration com uma CLI
- Usar DeletePolicy com uma CLI
- Usar DeleteScheduledAction com uma CLI
- Usar DeleteTags com uma CLI
- Usar DescribeAccountLimits com uma CLI
- Usar DescribeAdjustmentTypes com uma CLI
- Use DescribeAutoScalingGroupscom um AWS SDK ou CLI
- Use DescribeAutoScalingInstancescom um AWS SDK ou CLI
- Usar DescribeAutoScalingNotificationTypes com uma CLI
- Usar DescribeLaunchConfigurations com uma CLI
- Usar DescribeLifecycleHookTypes com uma CLI
- Usar DescribeLifecycleHooks com uma CLI
- Usar DescribeLoadBalancers com uma CLI
- Usar DescribeMetricCollectionTypes com uma CLI
- Usar DescribeNotificationConfigurations com uma CLI
- Usar DescribePolicies com uma CLI
- Use DescribeScalingActivitiescom um AWS SDK ou CLI
- Usar DescribeScalingProcessTypes com uma CLI
- Usar DescribeScheduledActions com uma CLI
- Usar DescribeTags com uma CLI
- Usar DescribeTerminationPolicyTypes com uma CLI
- Usar DetachInstances com uma CLI
- Usar DetachLoadBalancers com uma CLI

- Use DisableMetricsCollectioncom um AWS SDK ou CLI

- Use EnableMetricsCollectioncom um AWS SDK ou CLI
- Usar EnterStandby com uma CLI
- Usar ExecutePolicy com uma CLI
- Usar ExitStandby com uma CLI
- Usar PutLifecycleHook com uma CLI
- Usar PutNotificationConfiguration com uma CLI
- Usar PutScalingPolicy com uma CLI
- Usar PutScheduledUpdateGroupAction com uma CLI
- Usar RecordLifecycleActionHeartbeat com uma CLI
- Usar ResumeProcesses com uma CLI
- Use SetDesiredCapacitycom um AWS SDK ou CLI
- Usar SetInstanceHealth com uma CLI
- Usar SetInstanceProtection com uma CLI
- Usar SuspendProcesses com uma CLI
- Use TerminateInstanceInAutoScalingGroupcom um AWS SDK ou CLI
- Use UpdateAutoScalingGroupcom um AWS SDK ou CLI

Olá, Auto Scaling

Os exemplos de código a seguir mostram como começar a usar o Auto Scaling.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
```

```
public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling
 groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        if (response.AutoScalingGroups == null ||
 response.AutoScalingGroups.Count == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
            return;
        }
        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availabil
        });

    }
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_autoscalling.cpp.

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2
 Auto Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
&autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
"
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }
```

```
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
```

```
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
        });
    }
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
 client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
```

```
    hello_autoscaling(boto3.client("autoscaling"))
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroupsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
 operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
```

```
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:       #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
```

```
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroupsreferência da API AWS SDK for Rust.

Aprenda as noções básicas do Auto Scaling com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um grupo do Amazon EC2 Auto Scaling com um modelo de inicialização e zonas de disponibilidade e obtenha informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas de CloudWatch métricas e, em seguida, limpe os recursos.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.EC2;
using Microsoft.Extensions.Configuration;
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonCloudWatch>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalingWrapper>()
                .AddTransient<CloudWatchWrapper>()
                .AddTransient<EC2Wrapper>()
                .AddTransient<UIWrapper>()
            )
            .Build();


        var autoScalingWrapper =
host.Services.GetRequiredService<AutoScalingWrapper>();
        var cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();
        var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var imageId = configuration["ImageId"];
        var instanceType = configuration["InstanceType"];
        var launchTemplateName = configuration["LaunchTemplateName"];

        launchTemplateName += Guid.NewGuid().ToString();

        // The name of the Auto Scaling group.
        var groupName = configuration["GroupName"];

        uiWrapper.DisplayTitle("Auto Scaling Basics");
        uiWrapper.DisplayAutoScalingBasicsDescription();

        // Create the launch template and save the template Id to use when
deleting the
        // launch template at the end of the application.
        var launchTemplateId = await
ec2Wrapper.CreateLaunchTemplateAsync(imageId!, instanceType!,
launchTemplateName);
```

```
        // Confirm that the template was created by asking for a description of
it.
        await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

        uiWrapper.PressEnter();

        var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

        Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
        await autoScalingWrapper.CreateAutoScalingGroupAsync(
            groupName!,
            launchTemplateName,
            availabilityZones[0].ZoneName);

        // Keep checking the details of the new group until its lifecycle state
        // is "InService".
        Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

        List<AutoScalingInstanceDetails> instanceDetails;

        do
        {
            instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
        }
        while (instanceDetails.Count <= 0);

        Console.WriteLine($"Auto scaling group {groupName} successfully
created.");
        Console.WriteLine($"{instanceDetails.Count} instances were created for
the group.");

        // Display the details of the Auto Scaling group.
        instanceDetails.ForEach(detail =>
        {
            Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Metrics collection");
        Console.WriteLine($"Enable metrics collection for {groupName}");
        await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);
```

```
        // Show the metrics that are collected for the group.

        // Update the maximum size of the group to three instances.
        Console.WriteLine("--- Update the Auto Scaling group to increase max size
to 3 ---");
        int maxSize = 3;
        await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

        Console.WriteLine("--- Describe all Auto Scaling groups to show the
current state of the group ---");
        var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

        uiWrapper.DisplayGroupDetails(groups!);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Describe account limits");
        await autoScalingWrapper.DescribeAccountLimitsAsync();

        uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");

        uiWrapper.DisplayTitle("Set desired capacity");
        int desiredCapacity = 2;
        await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

        Console.WriteLine("Get the two instance Id values");

        // Empty the group before getting the details again.
        groups.Clear();
        groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
        if (groups.Any())
        {
            foreach (AutoScalingGroup group in groups)
            {
                Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
                Console.WriteLine($"The group ARN is
{group.AutoScalingGroupARN}");
                var instances = group.Instances;
```

```
                foreach (Amazon.AutoScaling.Model.Instance instance in instances)
                {
                    Console.WriteLine($"The instance id is
{instance.InstanceId}");
                    Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
                }
            }
        }

        uiWrapper.DisplayTitle("Scaling Activities");
        Console.WriteLine("Let's list the scaling activities that have occurred
for the group.");
        var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
        if (activities.Any())
        {
            activities.ForEach(activity =>
            {
                Console.WriteLine($"The activity Id is {activity.ActivityId}");
                Console.WriteLine($"The activity details are
{activity.Details}");
            });
        }

        // Display the Amazon CloudWatch metrics that have been collected.
        var metrics = await
cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
        if (metrics.Any())
        {
            Console.WriteLine($"Metrics collected for {groupName}:");
            metrics.ForEach(metric =>
            {
                Console.Write($"Metric name: {metric.MetricName}\t");
                Console.WriteLine($"Namespace: {metric.Namespace}");
            });
        }

        var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
        if (dataPoints.Any())
        {
            Console.WriteLine("Details for the metrics collected:");
            dataPoints.ForEach(detail => { Console.WriteLine(detail); });
```

```
        }

        // Disable metrics collection.
        Console.WriteLine("Disabling the collection of metrics for
{groupName}.");
        var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

        if (success)
        {
            Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
        }

        // Terminate all instances in the group.
        uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
        Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

        if (groups is not null)
        {
            groups.ForEach(group =>
            {
                // Only delete instances in the AutoScaling group we created.
                if (group.AutoScalingGroupName == groupName)
                {
                    group.Instances.ForEach(async instance =>
                    {
                        await
autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                    });
                }
            });
        }

        // After all instances are terminated, delete the group.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the Auto Scaling group.");
        await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);
```

```
        // Delete the launch template.
        var deletedLaunchTemplateName = await
 ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }

        Console.WriteLine("The demo is now concluded.");
    }
}


namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
    {
        Console.WriteLine("This code example performs the following
 operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
        Console.WriteLine(" 7. Changes the desired capacity of the Auto
 Scaling");
        Console.WriteLine("    group to use an additional instance.");
```

```
        Console.WriteLine(" 8. Shows that there are now instances in the
 group.");
        Console.WriteLine(" 9. Lists the scaling activities that have occurred
 for the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that
 have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
 group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>
        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }
```

```
    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

Defina as funções que são chamadas pelo cenário para gerenciar modelos e métricas de lançamento. Essas funções incluem Auto Scaling, Amazon EC2 e ações. CloudWatch

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
 client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)
    {
        _amazonAutoScaling = amazonAutoScaling;
    }


    /// <summary>
    /// Create a new Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name to use for the new Auto Scaling
    /// group.</param>
    /// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
    /// launch template to use to create instances in the group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        string availabilityZone)
    {
```

```
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };
        try
        {
            var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
            Console.WriteLine($"{groupName} Auto Scaling Group created");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }



    /// <summary>
    /// Retrieve information about Amazon EC2 Auto Scaling quotas to the
    /// active AWS account.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DescribeAccountLimitsAsync()
    {
        var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
response.MaxNumberOfAutoScalingGroups);
```

```
        Console.WriteLine("The current number of Auto Scaling groups is " +
 response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }



    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
```

```
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
 null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }



    /// <summary>
    /// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
```

```
    public async Task<List<AutoScalingGroup>> DescribeAutoScalingGroupsAsync(
        string groupName)
    {
        var groups = new List<AutoScalingGroup>();
        var groupList = new List<string>
            {
                groupName,
            };

        var request = new DescribeAutoScalingGroupsRequest
        {
            AutoScalingGroupNames = groupList,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
        if (response.AutoScalingGroups != null)
        {
            groups = response.AutoScalingGroups;
        }

        return groups;
    }


    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
 _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
```

```
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }


    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
```

```
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
 _amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
 _amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
 {desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
```

```
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }


    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
```

```
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
 {groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }

}


namespace AutoScalingActions;

using Amazon.EC2;
using Amazon.EC2.Model;

public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }

    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
```

```
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Delete an Amazon EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
    {
        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
```

```
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string
 launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.Write($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
    }

    /// <summary>
    /// Retrieve the availability zones for the current region.
    /// </summary>
    /// <returns>A collection of availability zones.</returns>
    public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
    {
        var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());

        return response.AvailabilityZones;
    }
}


namespace AutoScalingActions;
```

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</
returns>
    public async Task<List<Metric>> GetCloudWatchMetricsAsync(string groupName)
    {
        var metrics = new List<Metric>();
        var filter = new DimensionFilter
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        };

        var request = new ListMetricsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = new List<DimensionFilter> { filter },
            Namespace = "AWS/AutoScaling",
        };

        var response = await _amazonCloudWatch.ListMetricsAsync(request);
        if (response.Metrics != null)
```

```
        {
            metrics = response.Metrics;
        }
        return metrics;
    }

    /// <summary>
    /// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of data points.</returns>
    public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
    {
        var dataPoints = new List<Datapoint>();
        var metricDimensions = new List<Dimension>
            {
                new Dimension
                {
                    Name = "AutoScalingGroupName",
                    Value = $"{groupName}",
                },
            };

        // The start time will be yesterday.
        var startTime = DateTime.UtcNow.AddDays(-1);

        var request = new GetMetricStatisticsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = metricDimensions,
            Namespace = "AWS/AutoScaling",
            Period = 60, // 60 seconds.
            Statistics = new List<string>() { "Minimum" },
            StartTimeUtc = startTime,
            EndTimeUtc = DateTime.UtcNow,
        };

        var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);
        if (response.Datapoints != null)
        {
            dataPoints = response.Datapoints;
        }
```

```
        return dataPoints;
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
  \sa groupsAndInstancesScenario()
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
 */
```

```
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout
            << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
 Scaling "
            << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
              << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
            "Would you like to use an existing EC2 launch template (y/n)?  ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
                "Enter the name of the existing EC2 launch template.  ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateNames(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
                ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                      << "' exists by calling DescribeLaunchTemplate." <<
 std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    else { // 2.  Or create a new EC2 launch template.
        templateName = askQuestion("Enter the name for a new EC2 launch template:
 ");

        Aws::EC2::Model::CreateLaunchTemplateRequest request;
```

```
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
        requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

        request.SetLaunchTemplateData(requestLaunchTemplateData);

        Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
                ec2Client.CreateLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << " was
created."
                      << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
                 "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already
exists"
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
            "Enter a name for the Auto Scaling group:  ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
                ec2Client.DescribeAvailabilityZones(request);

        if (outcome.IsSuccess()) {
            std::cout
```

```
                    << "EC2 instances can be created in the following
Availability Zones:"
                    << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "   " << i + 1 << ".  "
                          << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    int availabilityZoneChoice = askQuestionForIntRange(
            "Choose an Availability Zone:  ", 1,
            static_cast<int>(availabilityZones.size()));
    // 4. Create an Auto Scaling group with the specified Availability Zone.
    {
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
```

```
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        std::cout << "Here is the Auto Scaling group description." << std::endl;
        if (!autoScalingGroups.empty()) {
            logAutoScalingGroupInfo(autoScalingGroups);
        }
    }
    else {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    std::cout
            << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
            << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    bool enableMetrics = askYesNoQuestion(
            "Do you want to collect metrics about the A"
            "Auto Scaling group during this demo (y/n)?  ");
    // 7. Optionally enable metrics collection for the Auto Scaling group.
    if (enableMetrics) {
        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
```

```
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
              "' from 1 to 3." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 8. Update the Auto Scaling group, setting a new maximum size.
    {
        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
```

```
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            const auto &instances = autoScalingGroups[0].GetInstances();
            std::cout
                    << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                    << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;
    std::cout << "Let's update the desired capacity in '" << groupName <<
              "' from 1 to 2." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    //  9. Update the Auto Scaling group, setting a new desired capacity.
    {
        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

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

```
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            std::cout
                    << "Here is the current state of the group." << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;

    std::cout << "Let's terminate one of the EC2 instances in " << groupName <<
"."
              << std::endl;
    std::cout << "Because the desired capacity is 2, another EC2 instance will
start "
              << "to replace the terminated EC2 instance."
              << std::endl;
    std::cout << "The currently running EC2 instances are:" << std::endl;

    if (autoScalingGroups.empty()) {
        std::cerr << "Error describing groups. No groups returned." << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
```

```
    }

    int instanceNumber = 1;
    Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
            autoScalingGroups[0].GetInstances());
    for (const Aws::String &instanceID: instanceIDs) {
        std::cout << "   " << instanceNumber << ". " << instanceID << std::endl;
        ++instanceNumber;
    }

    instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to
stop? ",
                                            1,

static_cast<int>(instanceIDs.size()));

    // 10. Terminate an EC2 instance in the Auto Scaling group.
    {
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);
```

```
   std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
             << std::endl;
   std::cout << "Let's get a report of scaling activities for EC2 launch group
'"
             << groupName << "'."
             << std::endl;
   askQuestion("Press enter to continue:  ", alwaysTrueTest);
   // 11. Get a description of activities for the Auto Scaling group.
   {
       Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
       request.SetAutoScalingGroupName(groupName);

       Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
       Aws::String nextToken; // Used for pagination;
       do {
           if (!nextToken.empty()) {
               request.SetNextToken(nextToken);
           }
           Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                   autoScalingClient.DescribeScalingActivities(request);

           if (outcome.IsSuccess()) {
               const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                       outcome.GetResult().GetActivities();
               allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
               nextToken  = outcome.GetResult().GetNextToken();
           }
           else {
               std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
                         << outcome.GetError().GetMessage()
                         << std::endl;
               cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
               return false;
           }
       } while (!nextToken.empty());

       std::cout << "Found " << allActivities.size() << " activities."
                 << std::endl;
```

```
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
    }

    if (enableMetrics) {
        if (!logAutoScalingMetrics(groupName, clientConfig)) {
            cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
            return false;
        }
    }

    std::cout << "Let's  clean up." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);

    // 13. Disable metrics collection if enabled.
    if (enableMetrics) {
        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
            return false;
        }
    }

    return cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
}
```

```
//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                            const
 Aws::AutoScaling::AutoScalingClient &client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
 instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
                break;
            }
            else {
                if ((count % 5) == 0) {
```

```
                    std::cout << "No instance IDs returned for group." <<
std::endl;
                }

                continue;
            }
        }

        // 6.  Check lifecycle state of the instances using
DescribeAutoScalingInstances.
        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const
Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();
            ready = instancesDetails.size() >= desiredCapacity;
            for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails
&details: instancesDetails) {
                if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                    ready = false;
                    break;
                }
            }
            // Log the status while waiting.
            if (((count % 5) == 1) || ready) {
                logInstancesLifecycleState(instancesDetails);
            }
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
```

```
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
\return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const
 Aws::AutoScaling::AutoScalingClient &autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client)
 {
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
        (askYesNoQuestion(
                Aws::String("Delete the Auto Scaling group '") + groupName +
                "'  (y/n)?"))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                    autoScalingClient.UpdateAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout
                        << "The minimum size and desired capacity of the Auto
 Scaling group "
                        << "was set to zero before terminating the instances."
                        << std::endl;
            }
            else {
```

```
                std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                          << outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
        }

        Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
        if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                               autoScalingClient)) {
            if (!autoScalingGroups.empty()) {
                Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                        autoScalingGroups[0].GetInstances());
                for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                    request.SetInstanceId(instanceID);
                    request.SetShouldDecrementDesiredCapacity(true);


Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =

autoScalingClient.TerminateInstanceInAutoScalingGroup(
                                     request);

                    if (outcome.IsSuccess()) {
                        std::cout << "Initiating termination of EC2 instance '"
                                  << instanceID << "'." << std::endl;
                    }
                    else {
                        std::cerr
                                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                                << outcome.GetError().GetMessage() << std::endl;
                        result = false;
                    }
                }
            }

            std::cout
                    << "Waiting for the EC2 instances to terminate before
deleting the "
                    << "Auto Scaling group..." << std::endl;
            waitForInstances(groupName, autoScalingGroups, autoScalingClient);
        }
```

```
        {
            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    // 15. Delete the EC2 launch template.
    if (!templateName.empty() && (askYesNoQuestion(
            Aws::String("Delete the EC2 launch template '") + templateName +
            "' (y/n)?"))) {
        Aws::EC2::Model::DeleteLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
                ec2Client.DeleteLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "EC2 launch template '" << templateName << "' was
deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::DeleteLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }
```

```
    return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
                                        const Aws::AutoScaling::AutoScalingClient
 &client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
            client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateAutoScalingGroup

- DeleteAutoScalingGroup
- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
```

```
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final String ROLES_STACK = "MyCdkAutoScaleStack";
    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        String groupName = "MyAutoScalingGroup2";
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_WEST_2)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
 scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("First, we will create a launch template using a
 CloudFormation script");
```

```
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String launchTemplateName = stackOutputs.get("LaunchTemplateNameOutput");
        String vpcZoneId = getVPC(ec2);
        updateTemlate(ec2, launchTemplateName );
        System.out.println("The VPC zone id created by the CloudFormation stack
is"+vpcZoneId);

        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, ec2, groupName,
launchTemplateName, vpcZoneId);

        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred
for the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling
group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group and cloud formation
resources");
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static String getVPC(Ec2Client ec2) {
        try {
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .filters(f -> f.name("isDefault").values("true"))
                    .build();

            DescribeVpcsResponse response = ec2.describeVpcs(request);

            if (!response.vpcs().isEmpty()) {
                Vpc defaultVpc = response.vpcs().get(0);
                System.out.println("Default VPC ID: " + defaultVpc.vpcId());
                return defaultVpc.vpcId();
            } else {
                System.out.println("No default VPC found.");
                return null; // Return null if no default VPC is found
            }

        } catch (Ec2Exception e) {
            System.err.println("EC2 error: " +
e.awsErrorDetails().errorMessage());
            return null; // Return null in case of an error
        }
    }


    public static void updateTemlate(Ec2Client ec2, String launchTemplateName ) {
        // Step 1: Create new launch template version
        String newAmiId = "ami-0025f0db847eb6254";
```

```
        RequestLaunchTemplateData launchTemplateData =
RequestLaunchTemplateData.builder()
                .imageId(newAmiId)
                .build();

        CreateLaunchTemplateVersionRequest createVersionRequest =
CreateLaunchTemplateVersionRequest.builder()
                .launchTemplateName(launchTemplateName)
                .versionDescription("Updated with valid AMI")
                .sourceVersion("1")
                .launchTemplateData(launchTemplateData)
                .build();

        CreateLaunchTemplateVersionResponse createResponse =
ec2.createLaunchTemplateVersion(createVersionRequest);
        int newVersionNumber =
createResponse.launchTemplateVersion().versionNumber().intValue();

        // Step 2: Modify default version
        ModifyLaunchTemplateRequest modifyRequest =
ModifyLaunchTemplateRequest.builder()
                .launchTemplateName(launchTemplateName)
                .defaultVersion(String.valueOf(newVersionNumber))
                .build();

        ec2.modifyLaunchTemplate(modifyRequest);
        System.out.println("Updated launch template to version " +
newVersionNumber + " with AMI " + newAmiId);
    }


    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
```

```
                System.out.println("The activity Id is " +
 activity.activityId());
                System.out.println("The activity details are " +
 activity.details());
            }

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

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
 String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
 SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

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

    public static void createAutoScalingGroup(AutoScalingClient
 autoScalingClient,
                                              Ec2Client ec2Client,
                                              String groupName,
                                              String launchTemplateName,
                                              String vpcId) {
        try {
            // Step 1: Get one subnet ID in the given VPC
            DescribeSubnetsRequest subnetRequest =
 DescribeSubnetsRequest.builder()
                    .filters(Filter.builder().name("vpc-
id").values(vpcId).build())
                    .build();
```

```
            DescribeSubnetsResponse subnetResponse =
ec2Client.describeSubnets(subnetRequest);

            if (subnetResponse.subnets().isEmpty()) {
                throw new RuntimeException("No subnets found in VPC: " + vpcId);
            }

            String subnetId = subnetResponse.subnets().get(0).subnetId(); // Use
first subnet
            System.out.println("Using subnet: " + subnetId);

            // Step 2: Create launch template reference
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            // Step 3: Create Auto Scaling group
            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .minSize(1)
                    .maxSize(1)
                    .vpcZoneIdentifier(subnetId)  // Correct: subnet ID, not VPC
ID
                    .build();

            autoScalingClient.createAutoScalingGroup(request);

            // Step 4: Wait until group is created
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
                    waiter.waitUntilGroupExists(groupsRequest);

            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (Ec2Exception | AutoScalingException e) {
```

```
            System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

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

    public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .maxRecords(10)
                    .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
```

```
                System.out.println("*** The service to use for the health checks:
" + group.healthCheckType());
            }

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

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }

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

```
   public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .granularity("1Minute")
                   .build();

           autoScalingClient.enableMetricsCollection(collectionRequest);
           System.out.println("The enable metrics collection operation was
successful");

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

   public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");

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

   public static void describeAccountLimits(AutoScalingClient autoScalingClient)
{
       try {
```

```
            DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
            System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
            System.out.println("The current number of auto scaling groups is " +
response.numberOfAutoScalingGroups());

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

    public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
" + groupName);

        } catch (AutoScalingException e) {
```

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

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
 autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
 TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

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

    public static void deleteAutoScalingGroup(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


 autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-
linked role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances
 in the Auto Scaling group can be created.
    """
```

```
    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of
the groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
```

```
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }
```

```
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}

suspend fun createAutoScalingGroup(
```

```
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
```

```
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
```

```
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances

- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("--------------------------------------\n");
```

```
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an
 Auto Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you
 created in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];
```

```
        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']
['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
```

```
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the
 group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups:
 {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this-
>autoScalingGroupName, 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
```

```
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that
 have occurred for the group so far.
         */
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
```

```
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
{$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
$metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']}
recorded:\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
```

```
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
```

```
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) ->
 None:
    """
    Runs the scenario demonstrating the management of Auto Scaling groups and
 instances.

    :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
 Scaling groups.
    :param svc_helper: An instance of the ServiceHelper that interacts with AWS
 services.
    :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")

    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
 instances."
    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
 new one: "
    )
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following
 specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
```

```
    print("-" * 88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask(
        "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) +
1)
    )
    group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_autoscaling_group(group_name, group_zones, template_name,
1, 1)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Created Auto Scaling group %s.", group_name)
    print("Created group:")
    pp(group)
    print("Waiting for instance to start...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    use_metrics = q.ask(
        "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
        q.is_yesno,
    )
    if use_metrics:
        as_wrapper.enable_metrics(
            group_name,
            [
                "GroupMinSize",
                "GroupMaxSize",
                "GroupDesiredCapacity",
                "GroupInServiceInstances",
                "GroupTotalInstances",
            ],
        )
        logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
        print(f"Metrics enabled for {group_name}.")
```

```
    print("-" * 88)

    print(f"Let's update the maximum number of instances in {group_name} from 1
to 3.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.update_group(group_name, MaxSize=3)
    group = as_wrapper.describe_group(group_name)
    logger.info("Updated maximum size for group %s to 3.", group_name)
    print("The group still has one running instance, but can have up to three:")
    print_simplified_group(group)
    print("-" * 88)

    print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.set_desired_capacity(group_name, 2)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Set desired capacity for group %s to 2.", group_name)
    print("Here's the current state of the group:")
    print_simplified_group(group)
    print("-" * 88)
    print("Waiting for the new instance to start...")
    instance_ids = wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's terminate one of the instances in {group_name}.")
    print("Because the desired capacity is 2, another instance will start.")
    print("The currently running instances are:")
    for index, inst_id in enumerate(instance_ids):
        print(f"\t{index+1}. {inst_id}")
    inst_sel = q.ask(
        "Which instance do you want to stop? ",
        q.is_int,
        q.in_range(1, len(instance_ids) + 1),
    )
    print(f"Stopping {instance_ids[inst_sel-1]}...")
    as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info(
        "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
group_name
    )
    print(f"Here's the state of {group_name}:")
```

```
    print_simplified_group(group)
    print("Waiting for the scaling activities to complete...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's get a report of scaling activities for {group_name}.")
    q.ask("Press Enter when you're ready.")
    activities = as_wrapper.describe_scaling_activities(group_name)
    logger.info(
        "Retrieved %d scaling activities for group %s.", len(activities),
group_name
    )
    print(
        f"Found {len(activities)} activities.\n"
        f"Activities are ordered with the most recent one first:"
    )
    for act in activities:
        pp(act)
    print("-" * 88)

    if use_metrics:
        print("Let's look at CloudWatch metrics.")
        metric_namespace = "AWS/AutoScaling"
        metric_dimensions = [{"Name": "AutoScalingGroupName", "Value":
group_name}]
        print(f"The following metrics are enabled for {group_name}:")
        done = False
        while not done:
            metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
            for index, metric in enumerate(metrics):
                print(f"\t{index+1}. {metric.name}")
            print(f"\t{len(metrics)+1}. None")
            metric_sel = q.ask(
                "Which metric do you want to see? ",
                q.is_int,
                q.in_range(1, len(metrics) + 1),
            )
            if metric_sel < len(metrics) + 1:
                span = 5
                metric = metrics[metric_sel - 1]
                print(f"Over the last {span} minutes, {metric.name} recorded:")
                # CloudWatch metric times are in the UTC+0 time zone.
                now = datetime.now(timezone.utc)
                metric_data = svc_helper.get_metric_statistics(
```

```
                    metric_dimensions, metric, now - timedelta(minutes=span), now
                )
                pp(metric_data)
                if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
                    done = True
            else:
                done = True

    print(f"Let's clean up.")
    q.ask("Press Enter when you're ready.")
    if use_metrics:
        print(f"Stopping metrics collection for {group_name}.")
        as_wrapper.disable_metrics(group_name)
        logger.info("Disabled metrics collection for group %s.", group_name)

    print(
        "You must terminate all instances in the group before you can delete the
group."
    )
    print("Set minimum size to 0.")
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
        logger.info("Terminated instance %s in group %s.", inst_id, group_name)
    print("Waiting for instances to stop...")
    wait_for_instances(instance_ids, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_autoscaling_group(group_name)
    logger.info("Deleted Auto Scaling group %s.", group_name)
    print("-" * 88)

    if template is not None:
        if q.ask(
            f"Do you want to delete launch template {template_name} used in this
demo (y/n)? "
        ):
            svc_helper.delete_template(template_name)
            logger.info("Deleted launch template %s.", template_name)
            print("Template deleted.")
```

```
    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")
```

Defina as funções que são chamadas pelo cenário para gerenciar modelos e métricas de lançamento. Essas funções envolvem o Amazon EC2 e CloudWatch as ações.

```
class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""

    def __init__(self, ec2_client, cloudwatch_resource):
        """
        :param ec2_client: A Boto3 Amazon EC2 client.
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.ec2_client = ec2_client
        self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for
 instances
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateNames=[template_name]
            )
            template = response["LaunchTemplates"][0]
```

```
            logger.info("Launch template %s retrieved successfully.",
template_name)
            return template
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                logger.warning("Launch template %s does not exist.",
template_name)
            else:
                logger.error(
                    "Couldn't verify launch template %s. Error: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.

        :param template_name: The name to give to the template.
        :param inst_type: The type of the instance, such as t1.micro.
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when
creating
                       an instance.
        :return: Information about the newly created template.
        :raises ClientError: If there is an error creating the launch template.
        """
        try:
            response = self.ec2_client.create_launch_template(
                LaunchTemplateName=template_name,
                LaunchTemplateData={"InstanceType": inst_type, "ImageId":
ami_id},
            )
            template = response["LaunchTemplate"]
            logger.info(
                "Created launch template %s with instance type %s and AMI ID
%s.",
                template_name,
```

```
                inst_type,
                ami_id,
            )
            return template
        except ClientError as err:
            logger.error(
                "Couldn't create launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete_template(self, template_name: str) -> None:
        """
        Deletes a launch template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If there is an error deleting the launch template.
        """
        try:

self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
            logger.info("Deleted launch template %s.", template_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_availability_zones(self) -> list:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        :raises ClientError: If there is an error retrieving availability zones.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
```

```
            logger.info("Retrieved availability zones: %s.", ", ".join(zones))
            return zones
        except ClientError as err:
            logger.error(
                "Couldn't get availability zones. Error: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_metrics(self, namespace: str, dimensions: list) -> list:
        """
        Gets a list of CloudWatch metrics filtered by namespace and dimensions.

        :param namespace: The namespace of the metrics to look up.
        :param dimensions: The dimensions of the metrics to look up.
        :return: The list of metrics.
        :raises ClientError: If there is an error retrieving CloudWatch metrics.
        """
        try:
            metrics = list(
                self.cloudwatch_resource.metrics.filter(
                    Namespace=namespace, Dimensions=dimensions
                )
            )
            logger.info(
                "Retrieved metrics for namespace %s with dimensions %s.",
                namespace,
                dimensions,
            )
            return metrics
        except ClientError as err:
            logger.error(
                "Couldn't get metrics for %s, %s. Error: %s: %s",
                namespace,
                dimensions,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    @staticmethod
    def get_metric_statistics(
        dimensions: list, metric, start: datetime, end: datetime
```

```
    ) -> list:
        """
        Gets statistics for a CloudWatch metric within a specified time span.

        :param dimensions: The dimensions of the metric.
        :param metric: The metric to look up.
        :param start: The start of the time span for retrieved metrics.
        :param end: The end of the time span for retrieved metrics.
        :return: The list of data points found for the specified metric.
        :raises ClientError: If there is an error retrieving metric statistics.
        """
        try:
            response = metric.get_statistics(
                Dimensions=dimensions,
                StartTime=start,
                EndTime=end,
                Period=60,
                Statistics=["Sum"],
            )
            data = response["Datapoints"]
            logger.info("Retrieved statistics for metric %s.", metric.name)
            return data
        except ClientError as err:
            logger.error(
                "Couldn't get statistics for metric %s. Error: %s: %s",
                metric.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


def print_simplified_group(group: dict) -> None:
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
 {group['DesiredCapacity']}"
```

```
    )
    if group["Instances"]:
        print(f"\tInstances:")
        for inst in group["Instances"]:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")


def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)


def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) ->
 list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
    """
    ready = False
    instances = []
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids
 else []
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
 instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
```

```
            f"Here are the details of the instance{'s' if len(instances) > 1 else
 ''}:"
        )
        for instance in instances:
            pp(instance)
    return instance_ids
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
[package]
name = "autoscaling-code-examples"
version = "0.1.0"
```

```
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
 <dpsouth@amazon.com>"]
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
clap = { version = "4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"


use std::{collections::BTreeSet, fmt::Display};

use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
```

```
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();

    // 1. Create an EC2 launch template that you'll use to create an auto scaling
 group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
 template.
    // 2. CreateAutoScalingGroup: pass it the launch template you created in step
 0. Give it min/max of 1 instance.
    // 4. EnableMetricsCollection: enable all metrics or a subset.
    let scenario = match
 AutoScalingScenario::prepare_scenario(&shared_config).await {
        Ok(scenario) => scenario,
        Err(errs) => {
            let err_str = errs
                .into_iter()
                .map(|e| e.to_string())
                .collect::<Vec<String>>()
                .join(", ");
            return Err(anyhow!("Failed to initialize scenario: {err_str}"));
        }
    };

    info!("Prepared autoscaling scenario:\n{scenario}");

    let stable = scenario.wait_for_stable(1).await;
    if let Err(err) = stable {
        warnings.push(
```

```
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 3. DescribeAutoScalingInstances: show that one instance has launched.
    show_scenario_description(
        &scenario,
        "show that the group was created and one instance has launched",
    )
    .await;

    // 5. UpdateAutoScalingGroup: update max size to 3.
    let scale_max_size = scenario.scale_max_size(3).await;
    if let Err(err) = scale_max_size {
        warnings.push("There was a problem scaling max size", err);
    }

    // 6. DescribeAutoScalingGroups: the current state of the group
    show_scenario_description(
        &scenario,
        "show the current state of the group after setting max size",
    )
    .await;

    // 7. SetDesiredCapacity: set desired capacity to 2.
    let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
    if let Err(err) = scale_desired_capacity {
        warnings.push("There was a problem setting desired capacity", err);
    }

    //   Wait for a second instance to launch.
    let stable = scenario.wait_for_stable(2).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 8. DescribeAutoScalingInstances: show that two instances are launched.
    show_scenario_description(
        &scenario,
        "show that two instances are launched after setting desired capacity",
```

```
    )
    .await;

    let ids_before = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    // 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in
 the group.
    let terminate_some_instance = scenario.terminate_some_instance().await;
    if let Err(err) = terminate_some_instance {
        warnings.push("There was a problem replacing an instance", err);
    }

    let wait_after_terminate = scenario.wait_for_stable(1).await;
    if let Err(err) = wait_after_terminate {
        warnings.push(
            "There was a problem waiting after terminating an instance",
            err,
        );
    }

    let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
    if let Err(err) = wait_scale_up_after_terminate {
        warnings.push(
            "There was a problem waiting for scale up after terminating an
 instance",
            err,
        );
    }

    let ids_after = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    let difference = ids_after.intersection(&ids_before).count();
    if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
        warnings.push(
            "Before and after set not different",
            ScenarioError::with(format!("{difference}")),
```

```
        );
    }

    // 10. DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
    show_scenario_description(
        &scenario,
        "list the scaling activities that have occurred for the group so far",
    )
    .await;

    // 11. DisableMetricsCollection
    let scale_group = scenario.scale_group_to_zero().await;
    if let Err(err) = scale_group {
        warnings.push("There was a problem scaling the group to 0", err);
    }
    show_scenario_description(&scenario, "Scenario scaled to 0").await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all
 instances):
    // 13. Delete LaunchTemplate.
    let clean_scenario = scenario.clean_scenario().await;
    if let Err(errs) = clean_scenario {
        for err in errs {
            warnings.push("There was a problem cleaning the scenario", err);
        }
    } else {
        info!("The scenario has been cleaned up!");
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        Err(anyhow!(
            "There were warnings during scenario execution:\n{warnings}"
        ))
    }
}

pub mod scenario;


use std::{
    error::Error,
```

```
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
const AUTOSCALING_GROUP_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25
 seconds.
const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at
 a time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
            ))
```

```
        } else {
            tokio::time::sleep(WAIT_TIME).await;
            Ok(())
        }
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\n",
            self.launch_template_arn
        ))?;
        f.write_fmt(format_args!(
            "\tScaling Group Name: {}\n",
            self.auto_scaling_group_name
        ))?;

        Ok(())
    }
}

pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t      Group status:")?;
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t- {status}")?;
                }
            }
```

```
            Err(e) => writeln!(f, "\t\t! - {e}")?,
        }
        writeln!(f, "\t         Instances:")?;
        match &self.instances {
            Ok(instances) => {
                for instance in instances {
                    writeln!(f, "\t\t- {instance}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        writeln!(f, "\t        Activities:")?;
        match &self.activities {
            Ok(activities) => {
                for activity in activities {
                    writeln!(
                        f,
                        "\t\t- {} Progress: {}% Status: {:?} End: {:?}",
                        activity.cause().unwrap_or("Unknown"),
                        activity.progress.unwrap_or(-1),
                        activity.status_code(),
                        // activity.status_message().unwrap_or_default()
                        activity.end_time(),
                    )?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        Ok(())
    }
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(|s| s.to_string()),
```

```
            code: err.code().map(|s| s.to_string()),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
 for this example
```

```
    // the ScenarioError captures the underlying information in MetadataError and
 treats it as a
    // single Error from this Crate. In other contexts, it may be appropriate to
 model the error
    // as including the SdkError as its source.
}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
 Vec<ScenarioError>> {
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);

        let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

        // Before creating any resources, prepare the list of AZs
        let availablity_zones = ec2.describe_availability_zones().send().await;
        if let Err(err) = availablity_zones {
            return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
        }

        let availability_zones: Vec<String> = availablity_zones
            .unwrap()
            .availability_zones
            .unwrap_or_default()
            .iter()
            .take(3)
            .map(|z| z.zone_name.clone().unwrap())
            .collect();

        // 1. Create an EC2 launch template that you'll use to create an auto
 scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
 template.
        //   * Recommended: InstanceType='t1.micro',
 ImageId='ami-0ca285d4c2cda3300'
        let create_launch_template = ec2
```

```
            .create_launch_template()
            .launch_template_name(LAUNCH_TEMPLATE_NAME)
            .launch_template_data(
                RequestLaunchTemplateData::builder()
                    .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
                    .image_id("ami-0ca285d4c2cda3300")
                    .build(),
            )
            .send()
            .await
            .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)])?;

        let launch_template_arn = match create_launch_template.launch_template {
            Some(launch_template) =>
launch_template.launch_template_id.unwrap_or_default(),
            None => {
                // Try to delete the launch template
                let _ = ec2
                    .delete_launch_template()
                    .launch_template_name(LAUNCH_TEMPLATE_NAME)
                    .send()
                    .await;
                return Err(vec![ScenarioError::with("Failed to load launch
template")]);
            }
        };

        // 2. CreateAutoScalingGroup: pass it the launch template you created in
step 0. Give it min/max of 1 instance.
        //   You can use EC2.describe_availability_zones() to get a list of AZs
(you have to specify an AZ when you create the group).
        //   Wait for instance to launch. Use a waiter if you have one, otherwise
DescribeAutoScalingInstances until LifecycleState='InService'
        if let Err(err) = autoscaling
            .create_auto_scaling_group()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .launch_template(
                LaunchTemplateSpecification::builder()
                    .launch_template_id(launch_template_arn.clone())
                    .version("$Latest")
                    .build(),
            )
            .max_size(1)
```

```
            .min_size(1)
            .set_availability_zones(Some(availability_zones))
            .send()
            .await
        {
            let mut errs = vec![ScenarioError::new(
                "Failed to create autoscaling group",
                &err,
            )];

            if let Err(err) = autoscaling
                .delete_auto_scaling_group()
                .auto_scaling_group_name(auto_scaling_group_name.as_str())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up autoscaling group",
                    &err,
                ));
            }

            if let Err(err) = ec2
                .delete_launch_template()
                .launch_template_id(launch_template_arn.clone())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up launch template",
                    &err,
                ));
            }
            return Err(errs);
        }

        let scenario = AutoScalingScenario {
            ec2,
            autoscaling: autoscaling.clone(), // Clients are cheap so cloning
here to prevent a move is ok.
            auto_scaling_group_name: auto_scaling_group_name.clone(),
            launch_template_arn,
        };
```

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;

        match enable_metrics_collection {
            Ok(_) => Ok(scenario),
            Err(err) => {
                scenario.clean_scenario().await?;
                Err(vec![ScenarioError::new(
                    "Failed to enable metrics collections for group",
                    &err,
                )])
            }
        }
    }

    pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
        let _ = self.wait_for_no_scaling().await;
        let delete_group = self
            .autoscaling
            .delete_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 14. Delete LaunchTemplate.
        let delete_launch_template = self
            .ec2
            .delete_launch_template()
            .launch_template_id(self.launch_template_arn.clone())
            .send()
            .await;

        let early_exit = match (delete_group, delete_launch_template) {
```

```
            (Ok(_), Ok(_)) => Ok(()),
            (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the launch template",
                &e,
            )]),
            (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the scale group",
                &e,
            )]),
            (Err(e1), Err(e2)) => Err(vec![
                ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
                ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
            ]),
        };

        if early_exit.is_err() {
            early_exit
        } else {
            // Wait for delete_group to finish
            let waiter = Waiter::new();
            let mut errors = Vec::<ScenarioError>::new();
            while errors.len() < 3 {
                if let Err(e) = waiter.sleep().await {
                    errors.push(e);
                    continue;
                }
                let describe_group = self
                    .autoscaling
                    .describe_auto_scaling_groups()

.auto_scaling_group_names(self.auto_scaling_group_name.clone())
                    .send()
                    .await;
                match describe_group {
                    Ok(group) => match group.auto_scaling_groups().first() {
                        Some(group) => {
                            if group.status() != Some("Delete in progress") {
                                errors.push(ScenarioError::with(format!(
                                    "Group in an unknown state while deleting:
{}",
                                    group.status().unwrap_or("unknown error")
                                )));
```

```
                                return Err(errors);
                            }
                        }
                        None => return Ok(()),
                    },
                    Err(err) => {
                        errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
                    }
                }
                if errors.len() > 3 {
                    return Err(errors);
                }
            }
            Err(vec![ScenarioError::with(
                "Exited cleanup wait loop without retuning success or failing
after three rounds",
            )])
        }
    }

    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });
```

```
        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

```
        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }

    pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
        let waiter = Waiter::new();
        let mut scaling = true;
        while scaling {
            waiter.sleep().await?;
            let describe_activities = self
                .autoscaling
                .describe_scaling_activities()
                .auto_scaling_group_name(self.auto_scaling_group_name.clone())
                .send()
                .await
                .map_err(|e| {
                    ScenarioError::new("Failed to get autoscaling activities for
group", &e)
                })?;
            let activities = describe_activities.activities();
```

```
            trace!(
                "Waiting for no scaling found {} activities",
                activities.len()
            );
            scaling = activities.iter().any(|a| a.progress() < Some(100));
        }
        Ok(())
    }

    pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError>
 {
        self.wait_for_no_scaling().await?;

        let mut group = self.get_group().await?;
        let mut count = count_group_instances(&group);

        let waiter = Waiter::new();
        while count != size {
            trace!("Waiting for stable {size} (current: {count})");
            waiter.sleep().await?;
            group = self.get_group().await?;
            count = count_group_instances(&group);
        }

        Ok(())
    }

    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
```

```
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }

    pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failer to update group to min size ({size}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
        // 5. UpdateAutoScalingGroup: update max size to 3.
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .max_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to max size ({size})").as_str(),
```

```
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
        // 7. SetDesiredCapacity: set desired capacity to 2.
        //   Wait for a second instance to launch.
        let update_group = self
            .autoscaling
            .set_desired_capacity()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .desired_capacity(capacity)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to desired capacity
({capacity}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
        //   UpdateAutoScalingGroup with MinSize=0
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
```

```
            .min_size(0)
            .desired_capacity(0)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                "Failed to update group for scaling down&",
                &err,
            ));
        }

        let stable = self.wait_for_stable(0).await;
        if let Err(err) = stable {
            return Err(ScenarioError::with(format!(
                "Error while waiting for group to be stable on scale down: {err}"
            )));
        }

        Ok(())
    }

    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
```

```
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Ações para Auto Scaling usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais de Auto Scaling com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Auto Scaling e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Auto Scaling usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API Amazon EC2 Auto Scaling.

Exemplos

- Usar AttachInstances com uma CLI
- Use AttachLoadBalancerTargetGroupscom um AWS SDK ou CLI
- Usar AttachLoadBalancers com uma CLI
- Usar CompleteLifecycleAction com uma CLI
- Use CreateAutoScalingGroupcom um AWS SDK ou CLI
- Usar CreateLaunchConfiguration com uma CLI
- Usar CreateOrUpdateTags com uma CLI
- Use DeleteAutoScalingGroupcom um AWS SDK ou CLI
- Usar DeleteLaunchConfiguration com uma CLI
- Usar DeleteLifecycleHook com uma CLI
- Usar DeleteNotificationConfiguration com uma CLI
- Usar DeletePolicy com uma CLI
- Usar DeleteScheduledAction com uma CLI
- Usar DeleteTags com uma CLI
- Usar DescribeAccountLimits com uma CLI
- Usar DescribeAdjustmentTypes com uma CLI
- Use DescribeAutoScalingGroupscom um AWS SDK ou CLI
- Use DescribeAutoScalingInstancescom um AWS SDK ou CLI
- Usar DescribeAutoScalingNotificationTypes com uma CLI
- Usar DescribeLaunchConfigurations com uma CLI
- Usar DescribeLifecycleHookTypes com uma CLI
- Usar DescribeLifecycleHooks com uma CLI
- Usar DescribeLoadBalancers com uma CLI
- Usar DescribeMetricCollectionTypes com uma CLI
- Usar DescribeNotificationConfigurations com uma CLI
- Usar DescribePolicies com uma CLI
- Use DescribeScalingActivitiescom um AWS SDK ou CLI

- Usar DescribeScalingProcessTypes com uma CLI
- Usar DescribeScheduledActions com uma CLI
- Usar DescribeTags com uma CLI
- Usar DescribeTerminationPolicyTypes com uma CLI
- Usar DetachInstances com uma CLI
- Usar DetachLoadBalancers com uma CLI
- Use DisableMetricsCollectioncom um AWS SDK ou CLI
- Use EnableMetricsCollectioncom um AWS SDK ou CLI
- Usar EnterStandby com uma CLI
- Usar ExecutePolicy com uma CLI
- Usar ExitStandby com uma CLI
- Usar PutLifecycleHook com uma CLI
- Usar PutNotificationConfiguration com uma CLI
- Usar PutScalingPolicy com uma CLI
- Usar PutScheduledUpdateGroupAction com uma CLI
- Usar RecordLifecycleActionHeartbeat com uma CLI
- Usar ResumeProcesses com uma CLI
- Use SetDesiredCapacitycom um AWS SDK ou CLI
- Usar SetInstanceHealth com uma CLI
- Usar SetInstanceProtection com uma CLI
- Usar SuspendProcesses com uma CLI
- Use TerminateInstanceInAutoScalingGroupcom um AWS SDK ou CLI
- Use UpdateAutoScalingGroupcom um AWS SDK ou CLI

Usar **`AttachInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AttachInstances`.

CLI

AWS CLI

Como anexar uma instância a um grupo do Auto Scaling

Este exemplo anexa a instância especificada ao grupo do Auto Scaling especificado.

```
aws autoscaling attach-instances \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

Este comando não produz saída.

- Para obter detalhes da API, consulte AttachInstancesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo anexa o grupo de destino especificado ao grupo do Auto Scaling especificado. O Auto Scaling aumenta automaticamente a capacidade desejada do grupo do Auto Scaling.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte AttachInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo anexa o grupo de destino especificado ao grupo do Auto Scaling especificado. O Auto Scaling aumenta automaticamente a capacidade desejada do grupo do Auto Scaling.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte AttachInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`AttachLoadBalancerTargetGroups`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AttachLoadBalancerTargetGroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }
```

- Para obter detalhes da API, consulte AttachLoadBalancerTargetGroupsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como anexar um grupo de destino a um grupo do Auto Scaling

Este exemplo anexa o grupo de destino especificado ao grupo do Auto Scaling especificado.

```
aws autoscaling attach-load-balancer-target-groups \
    --auto-scaling-group-name my-asg \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Este comando não produz saída.

Para obter mais informações, consulte Elastic Load Balancing e Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte AttachLoadBalancerTargetGroupsna Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
```

- Para obter detalhes da API, consulte AttachLoadBalancerTargetGroupsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```

```
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a AttachLoadBalancerTargetGroupsReferência da API AWS SDK for Python (Boto3).

Usar **`AttachLoadBalancers`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AttachLoadBalancers`.

CLI

AWS CLI

Como anexar um Classic Load Balancer ao grupo do Auto Scaling

Este exemplo anexa o Classic Load Balancer especificado ao grupo do Auto Scaling especificado.

```
aws autoscaling attach-load-balancers \
    --load-balancer-names my-load-balancer \
    --auto-scaling-group-name my-asg
```

Este comando não produz saída.

Para obter mais informações, consulte Elastic Load Balancing e Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte AttachLoadBalancersna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo anexa o Balanceador de Carga especificado ao grupo do Auto Scaling especificado.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte AttachLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo anexa o Balanceador de Carga especificado ao grupo do Auto Scaling especificado.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte AttachLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CompleteLifecycleAction`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CompleteLifecycleAction`.

CLI

AWS CLI

Como concluir a ação do ciclo de vida

Este exemplo notifica o Amazon EC2 Auto Scaling de que a ação do ciclo de vida especificada foi concluída para que ele possa concluir a execução ou o encerramento da instância.

```
aws autoscaling complete-lifecycle-action \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-result CONTINUE \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Este comando não produz saída.

Para obter mais informações, consulte Ganchos do ciclo de vida do Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte CompleteLifecycleActionna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo conclui a ação de ciclo de vida especificada.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Para obter detalhes da API, consulte CompleteLifecycleActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo conclui a ação de ciclo de vida especificada.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Para obter detalhes da API, consulte CompleteLifecycleActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateAutoScalingGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateAutoScalingGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
        {
            availabilityZone,
        };

    var request = new CreateAutoScalingGroupRequest
```

```
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };
        try
        {
            var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
            Console.WriteLine($"{groupName} Auto Scaling Group created");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }
```

- Para obter detalhes da API, consulte CreateAutoScalingGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
```

```
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Para obter detalhes da API, consulte CreateAutoScalingGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como criar um grupo do Auto Scaling

O exemplo de `create-auto-scaling-group` a seguir cria um grupo do Auto Scaling em sub-redes de várias zonas de disponibilidade de uma região. As instâncias são executadas com a versão padrão do modelo de execução especificado. Observe que os padrões são usados na maioria das outras configurações, como nas políticas de encerramento e na configuração de verificação de integridade.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --min-size 1 \
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Este comando não produz saída.

Para obter mais informações, consulte Grupos do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: como anexar o Application Load Balancer, o Network Load Balancer ou o Gateway Load Balancer

Este exemplo especifica o ARN de um grupo de destino para um balanceador de carga compatível com o tráfego esperado. O tipo de verificação de integridade especifica o `ELB`. Desta forma, quando o Elastic Load Balancing reportar uma instância como não íntegra, o grupo do Auto Scaling a substitui. O comando também define um período de carência de `600` segundos para a verificação de integridade. O período de carência ajuda a evitar o encerramento prematuro de instâncias recém-iniciadas.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/943f017f100becff \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --min-size 1 \
```

```
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Este comando não produz saída.

Para obter mais informações, consulte Elastic Load Balancing e Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 3: como especificar um grupo de posicionamento e usar a versão mais recente do modelo de execução

Este exemplo executa instâncias em um grupo de posicionamento dentro de uma única zona de disponibilidade. Isso pode ser útil para grupos de baixa latência com workloads de HPC. Esse exemplo também especifica o tamanho mínimo e máximo e a capacidade desejada do grupo.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest' \
    --min-size 1 \
    --max-size 5 \
    --desired-capacity 3 \
    --placement-group my-placement-group \
    --vpc-zone-identifier "subnet-6194ea3b"
```

Este comando não produz saída.

Para obter mais informações, consulte Grupos de posicionamento no Guia do usuário para instâncias do Linux do Amazon EC2.

Exemplo 4: como especificar um grupo do Auto Scaling de instância única e usar uma versão específica para iniciar o modelo

Este exemplo cria um grupo do Auto Scaling com capacidade mínima e máxima definida como 1 para impor que uma apenas instância seja executada. O comando também especifica a v1 de um modelo de execução no qual o ID de um ENI existente é especificado. Ao usar um modelo de execução que especifica um ENI existente para eth0, é necessário especificar uma zona de disponibilidade para o grupo do Auto Scaling que corresponda à interface de rede, mas sem especificar um ID de sub-rede na solicitação.

```
aws autoscaling create-auto-scaling-group \
```

```
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
 \
    --min-size 1 \
    --max-size 1 \
    --availability-zones us-west-2a
```

Este comando não produz saída.

Para obter mais informações, consulte Grupos do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 5: como especificar uma política de encerramento diferente

Este exemplo cria um grupo do Auto Scaling usando uma configuração de execução e define a política de encerramento para encerrar as instâncias mais antigas primeiro. O comando também aplica uma tag ao grupo e suas instâncias, com uma chave `Role` e valor de `WebServer`.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-configuration-name my-lc \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "OldestInstance" \
    --tags "ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Este comando não produz saída.

Para obter mais informações, consulte Trabalhar com políticas de término do Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 6: como especificar um gancho do ciclo de vida de lançamento

Este exemplo a seguir cria um grupo do Auto Scaling com um gancho do ciclo de vida que oferece suporte a uma ação personalizada na inicialização da instância.

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

Conteúdo do arquivo `config.json`:

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
        "LifecycleHookName": "my-launch-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
        "NotificationTargetARN": "arn:aws:sqs:us-west-2:123456789012:my-sqs-
queue",
        "RoleARN": "arn:aws:iam::123456789012:role/my-notification-role",
        "NotificationMetadata": "SQS message metadata",
        "HeartbeatTimeout": 4800,
        "DefaultResult": "ABANDON"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": [{
        "ResourceType": "auto-scaling-group",
        "ResourceId": "my-asg",
        "PropagateAtLaunch": true,
        "Value": "test",
        "Key": "environment"
    }]
}
```

Este comando não produz saída.

Para obter mais informações, consulte Ganchos do ciclo de vida do Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 7: como especificar um gancho do ciclo de vida de encerramento

Este exemplo a seguir cria um grupo do Auto Scaling com um gancho do ciclo de vida que oferece suporte a uma ação personalizada no encerramento da instância.

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

Conteúdo de `config.json`:

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
        "LifecycleHookName": "my-termination-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
        "HeartbeatTimeout": 120,
        "DefaultResult": "CONTINUE"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "TargetGroupARNs": [
        "arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067"
    ],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

Este comando não produz saída.

Para obter mais informações, consulte Ganchos do ciclo de vida do Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 8: como especificar uma política de encerramento personalizada

Este exemplo cria um grupo do Auto Scaling que especifica uma política de encerramento da função do Lambda personalizada que diz ao Amazon EC2 Auto Scaling quais instâncias podem ser encerradas com segurança em escala.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "arn:aws:lambda:us-
west-2:123456789012:function:HelloFunction:prod" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Este comando não produz saída.

Para obter mais informações, consulte Criar uma política de término personalizada com o Lambda no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte CreateAutoScalingGroupna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
<vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC)
where instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {

        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
```

```
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

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

- Para obter detalhes da API, consulte CreateAutoScalingGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

- Para obter detalhes da API, consulte a CreateAutoScalingGroupreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function createAutoScalingGroup(
    $autoScalingGroupName,
    $availabilityZones,
    $minSize,
    $maxSize,
    $launchTemplateId
) {
    return $this->autoScalingClient->createAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'AvailabilityZones' => $availabilityZones,
        'MinSize' => $minSize,
        'MaxSize' => $maxSize,
        'LaunchTemplate' => [
            'LaunchTemplateId' => $launchTemplateId,
        ],
    ]);
}
```

- Para obter detalhes da API, consulte CreateAutoScalingGroupa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria um grupo do Auto Scaling com o nome e os atributos especificados. A capacidade desejada padrão é o tamanho mínimo. Portanto, esse grupo do Auto Scaling inicia duas instâncias, uma em cada uma das duas zonas de disponibilidade especificadas.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-
lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Para obter detalhes da API, consulte CreateAutoScalingGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um grupo do Auto Scaling com o nome e os atributos especificados. A capacidade desejada padrão é o tamanho mínimo. Portanto, esse grupo do Auto Scaling inicia duas instâncias, uma em cada uma das duas zonas de disponibilidade especificadas.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-
lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Para obter detalhes da API, consulte CreateAutoScalingGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def create_group(
```

```
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
    ) -> None:
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
        :param group_zones: The Availability Zones in which instances can be
created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
template.
                                     The launch template specifies the
configuration of
                                     instances that are created by auto scaling
activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling
group.
        """
        try:
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=group_name,
                AvailabilityZones=group_zones,
                LaunchTemplate={
                    "LaunchTemplateName": launch_template_name,
                    "Version": "$Default",
                },
                MinSize=min_size,
                MaxSize=max_size,
            )

            # Wait for the group to exist.
            waiter = self.autoscaling_client.get_waiter("group_exists")
            waiter.wait(AutoScalingGroupNames=[group_name])

            logger.info(f"Successfully created Auto Scaling group {group_name}.")

        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to create Auto Scaling group {group_name}.")
            if error_code == "AlreadyExistsFault":
                logger.error(
                    f"An Auto Scaling group with the name '{group_name}' already
 exists. "
                    "Please use a different name or update the existing group.",
                )
            elif error_code == "LimitExceededFault":
                logger.error(
                    "The request failed because you have reached the limit "
                    "on the number of Auto Scaling groups or launch
 configurations. "
                    "Consider deleting unused resources or request a limit
 increase. "
                    "\nSee Auto Scaling Service Quota documentation here:"
                    "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/
ec2-auto-scaling-quotas.html"
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Para obter detalhes da API, consulte a CreateAutoScalingGroupReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error>
 {
    client
        .create_auto_scaling_group()
```

```
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateAutoScalingGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lo_launch_template TYPE REF TO /aws1/cl_asclaunchtemplatespec.

    " Example: iv_group_name = 'my-auto-scaling-group'
    " Example: iv_launch_template_name = 'my-launch-template'
    " Example: iv_min_size = 1
    " Example: iv_max_size = 3
    " Example: iv_vpc_zone_identifier = 'subnet-12345,subnet-67890' (for VPC)

    TRY.
        " Create launch template specification
        lo_launch_template = NEW /aws1/cl_asclaunchtemplatespec(
          iv_launchtemplatename = iv_launch_template_name
          iv_version = '$Default' ).

        " Create the Auto Scaling group
```

```
        " Use VPCZoneIdentifier for VPC subnets, or AvailabilityZones for EC2-
Classic
        IF iv_vpc_zone_identifier IS NOT INITIAL.
          " VPC-based deployment - use subnet IDs
          ao_asc->createautoscalinggroup(
            iv_autoscalinggroupname = iv_group_name
            iv_vpczoneidentifier = iv_vpc_zone_identifier
            io_launchtemplate = lo_launch_template
            iv_minsize = iv_min_size
            iv_maxsize = iv_max_size ).
        ELSE.
          " EC2-Classic or default VPC - use availability zones
          ao_asc->createautoscalinggroup(
            iv_autoscalinggroupname = iv_group_name
            it_availabilityzones = it_group_zones
            io_launchtemplate = lo_launch_template
            iv_minsize = iv_min_size
            iv_maxsize = iv_max_size ).
        ENDIF.

        " Wait for the group to be created (simplified - in production use proper
 polling)
        WAIT UP TO 10 SECONDS.

        MESSAGE 'Auto Scaling group created successfully' TYPE 'I'.

      CATCH /aws1/cx_ascalreadyexistsfault INTO DATA(lo_already_exists).
        RAISE EXCEPTION lo_already_exists.
      CATCH /aws1/cx_asclimitexceededfault INTO DATA(lo_limit_exceeded).
        RAISE EXCEPTION lo_limit_exceeded.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAutoScalingGroupreferência da API AWS SDK for SAP ABAP.

Usar **`CreateLaunchConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateLaunchConfiguration`.

CLI

AWS CLI

Exemplo 1: criar uma configuração de execução

Este exemplo cria uma configuração de inicialização simples.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large
```

Este comando não produz saída.

Para obter mais informações, consulte Criar uma configuração de execução no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: criar uma configuração de inicialização com um grupo de segurança, um par de chaves e um script de bootstrapping

Este exemplo cria uma configuração de execução com um grupo de segurança, um par de chaves e um script de bootstrapping contido nos dados do usuário.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --security-groups sg-eb2af88example \
    --key-name my-key-pair \
    --user-data file://myuserdata.txt
```

Este comando não produz saída.

Para obter mais informações, consulte Criar uma configuração de execução no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 3: criar uma configuração de inicialização com um perfil do IAM

Este exemplo cria uma configuração de execução com o nome do perfil de instância de um perfil do IAM.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --iam-instance-profile my-autoscaling-role
```

Este comando não produz saída.

Para obter mais informações, consulte Função do IAM para aplicações que são executadas em instâncias do Amazon EC2 no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 4: habilitar o monitoramento detalhado em uma configuração de execução

Este exemplo cria uma configuração de lançamento com o monitoramento detalhado do EC2 ativado, que envia métricas do EC2 CloudWatch em períodos de 1 minuto.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --instance-monitoring Enabled=true
```

Este comando não produz saída.

Para obter mais informações, consulte Configurando o monitoramento de instâncias do Auto Scaling na Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 5: criar uma configuração de execução para instâncias spot

Este exemplo cria uma configuração de execução que usa instâncias spot como a única opção de compra.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --spot-price "0.50"
```

Este comando não produz saída.

Para obter mais informações, consulte Solicitar instâncias spot no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 6: criar uma configuração de execução usando uma instância do EC2

Este exemplo cria uma configuração de execução com base nos atributos de uma instância existente. Ele substitui a locação de localização e determina se um endereço IP público é definido, incluindo as opções `--placement-tenancy` e `--no-associate-public-ip-address`.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc-from-instance \
    --instance-id i-0123a456700123456 \
    --instance-type m5.large \
    --no-associate-public-ip-address \
    --placement-tenancy dedicated
```

Este comando não produz saída.

Para obter mais informações, consulte Criação de uma configuração de execução usando uma instância do EC2 no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 7: criar uma configuração de execução com um mapeamento de dispositivos de blocos para um volume do Amazon EBS

Este exemplo cria uma configuração de execução com um mapeamento de dispositivos de blocos para um volume `gp3` do Amazon EBS com o nome do dispositivo `/dev/sdh` e um tamanho de volume de 20.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdh","Ebs":
{"VolumeSize":20,"VolumeType":"gp3"}}]'
```

Este comando não produz saída.

Para obter mais informações, consulte EBS na Referência da API do Amazon EC2 Auto Scaling.

Para obter informações sobre a sintaxe para citar valores de JSON-formatted parâmetros, consulte Usando aspas com cadeias de caracteres na AWS CLI no Guia do usuário da interface de linha de comando.AWS

Exemplo 8: criar uma configuração de execução com um mapeamento de dispositivos de blocos para um volume de armazenamento de instância

Este exemplo cria uma configuração de execução com `ephemeral1` como volume de armazenamento de instância com o nome do dispositivo `/dev/sdc`.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/
sdc","VirtualName":"ephemeral1"}]'
```

Este comando não produz saída.

Para obter mais informações, consulte a BlockDeviceMappingReferência da API Amazon EC2 Auto Scaling.

Para obter informações sobre a sintaxe para citar valores de JSON-formatted parâmetros, consulte Usando aspas com cadeias de caracteres na AWS CLI no Guia do usuário da interface de linha de comando.AWS

Exemplo 9: criar uma configuração de execução e impedir que um dispositivo de blocos se conecte no momento da execução

Este exemplo cria uma configuração de execução que suprime um dispositivo de blocos especificado pelo mapeamento de dispositivos de blocos da AMI (por exemplo, `/dev/sdf`).

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdf","NoDevice":""}]'
```

Este comando não produz saída.

Para obter mais informações, consulte a BlockDeviceMappingReferência da API Amazon EC2 Auto Scaling.

Para obter informações sobre a sintaxe para citar valores de JSON-formatted parâmetros, consulte Usando aspas com cadeias de caracteres na AWS CLI no Guia do usuário da interface de linha de comando.AWS

- Para obter detalhes da API, consulte CreateLaunchConfigurationna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria uma configuração de inicialização chamada “my-lc”. As instâncias do EC2 iniciadas por grupos do Auto Scaling que usam essa configuração de inicialização usam o tipo de instância, a AMI, o grupo de segurança e o perfil do IAM especificados.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
 "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
IamInstanceProfile "myIamRole"
```

- Para obter detalhes da API, consulte CreateLaunchConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma configuração de inicialização chamada “my-lc”. As instâncias do EC2 iniciadas por grupos do Auto Scaling que usam essa configuração de inicialização usam o tipo de instância, a AMI, o grupo de segurança e o perfil do IAM especificados.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
 "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
IamInstanceProfile "myIamRole"
```

- Para obter detalhes da API, consulte CreateLaunchConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateOrUpdateTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateOrUpdateTags`.

CLI

AWS CLI

Para criar ou atualizar tags de um grupo do Auto Scaling

Este exemplo adiciona duas tags ao grupo do Auto Scaling especificado.

```
aws autoscaling create-or-update-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true ResourceId=my-
asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research,PropagateAtLaunch=true
```

Este comando não produz saída.

Para obter mais informações, consulte Marcar instâncias e grupos de Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte CreateOrUpdateTagsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo adiciona uma única tag ao grupo do Auto Scaling especificado. A chave da tag é 'myTag' e o valor da tag é TagValue 'meu'. O Auto Scaling propaga essa tag para as instâncias do EC2 subsequentes lançadas pelo grupo do Auto Scaling. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
```

```
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- Para obter detalhes da API, consulte CreateOrUpdateTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo adiciona uma única tag ao grupo do Auto Scaling especificado. A chave da tag é 'myTag' e o valor da tag é TagValue 'meu'. O Auto Scaling propaga essa tag para as instâncias do EC2 subsequentes lançadas pelo grupo do Auto Scaling. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- Para obter detalhes da API, consulte CreateOrUpdateTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteAutoScalingGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteAutoScalingGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize o tamanho mínimo de um grupo do Auto Scaling para zero, encerre todas as instâncias no grupo e exclua o grupo.

```
    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
```

```
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
```

```
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
```

```
    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };
```

```
        var response = await
_amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
```

```
                std::cout << "Auto Scaling group '" << groupName << "' was
 deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como excluir o grupo do Auto Scaling especificado

Este exemplo exclui o grupo do Auto Scaling especificado.

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir infraestrutura do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: como forçar a exclusão do grupo do Auto Scaling especificado

Use a opção `--force-delete` para excluir o grupo do Auto Scaling sem precisar esperar que as instâncias do grupo sejam encerradas.

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --force-delete
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir infraestrutura do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DeleteAutoScalingGroupna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


 autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

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

- Para obter detalhes da API, consulte DeleteAutoScalingGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAutoScalingGroupreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
```

```
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroupa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui o grupo do Auto Scaling especificado se ele não tiver instâncias em execução. A confirmação será solicitada antes que a operação continue.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
 Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Exemplo 3: Este exemplo exclui o grupo do Auto Scaling especificado e encerra todas as instâncias em execução que ele contém.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui o grupo do Auto Scaling especificado se ele não tiver instâncias em execução. A confirmação será solicitada antes que a operação continue.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
 Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Exemplo 3: Este exemplo exclui o grupo do Auto Scaling especificado e encerra todas as instâncias em execução que ele contém.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize o tamanho mínimo de um grupo do Auto Scaling para zero, encerre todas as instâncias no grupo e exclua o grupo.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"
```

```
        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
```

```
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DeleteAutoScalingGroupReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(),
 Error> {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
```

```
}
```

- Para obter detalhes da API, consulte a DeleteAutoScalingGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " Example: iv_group_name = 'my-auto-scaling-group'

   TRY.
       ao_asc->deleteautoscalinggroup(
         iv_autoscalinggroupname = iv_group_name ).

       " Wait for the group to be deleted (simplified - in production use proper
polling)
       WAIT UP TO 10 SECONDS.

       MESSAGE 'Auto Scaling group deleted successfully' TYPE 'I'.

     CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
       RAISE EXCEPTION lo_activity_in_progress.
     CATCH /aws1/cx_ascresourceinusefault INTO DATA(lo_resource_in_use).
       RAISE EXCEPTION lo_resource_in_use.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
       RAISE EXCEPTION lo_generic_exception.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAutoScalingGroupreferência da API AWS SDK for SAP ABAP.

Usar **`DeleteLaunchConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteLaunchConfiguration`.

CLI

AWS CLI

Para excluir sua configuração de execução

Este exemplo exclui a configuração de execução especificada.

```
aws autoscaling delete-launch-configuration \
    --launch-configuration-name my-launch-config
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir infraestrutura do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DeleteLaunchConfigurationna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui a configuração de inicialização especificada se ela não estiver anexada a um grupo do Auto Scaling. A confirmação será solicitada antes que a operação continue.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
 on Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Para obter detalhes da API, consulte DeleteLaunchConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a configuração de inicialização especificada se ela não estiver anexada a um grupo do Auto Scaling. A confirmação será solicitada antes que a operação continue.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
 on Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Para obter detalhes da API, consulte DeleteLaunchConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteLifecycleHook`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteLifecycleHook`.

CLI

AWS CLI

Para excluir um gancho do ciclo de vida

Este exemplo exclui o gancho do ciclo de vida especificado.

```
aws autoscaling delete-lifecycle-hook \
    --lifecycle-hook-name my-lifecycle-hook \
    --auto-scaling-group-name my-asg
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteLifecycleHookna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o gancho do ciclo de vida especificado para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- Para obter detalhes da API, consulte DeleteLifecycleHookem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o gancho do ciclo de vida especificado para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- Para obter detalhes da API, consulte DeleteLifecycleHookem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteNotificationConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteNotificationConfiguration`.

CLI

AWS CLI

Para excluir uma notificação de ajuste de escala automático

Este exemplo exclui a notificação especificada do grupo do Auto Scaling especificado.

```
aws autoscaling delete-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir a configuração de notificação no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DeleteNotificationConfigurationna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui a ação de notificação especificada. A confirmação será solicitada antes que a operação continue.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Para obter detalhes da API, consulte DeleteNotificationConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a ação de notificação especificada. A confirmação será solicitada antes que a operação continue.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Para obter detalhes da API, consulte DeleteNotificationConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeletePolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeletePolicy`.

CLI

AWS CLI

Para excluir uma política de escalabilidade

Este exemplo exclui a política de escalabilidade especificada.

```
aws autoscaling delete-policy \
```

```
    --auto-scaling-group-name my-asg \
    --policy-name alb1000-target-tracking-scaling-policy
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeletePolicyna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui a política especificada para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
 "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Para obter detalhes da API, consulte DeletePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a política especificada para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
 "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Para obter detalhes da API, consulte DeletePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteScheduledAction`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteScheduledAction`.

CLI

AWS CLI

Para excluir uma ação programada de um grupo do Auto Scaling

Este exemplo exclui a ação programada especificada do grupo do Auto Scaling especificado.

```
aws autoscaling delete-scheduled-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteScheduledActionna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui a política especificada para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- Para obter detalhes da API, consulte DeleteScheduledActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a política especificada para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- Para obter detalhes da API, consulte DeleteScheduledActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteTags`.

CLI

AWS CLI

Para excluir uma tag de um grupo do Auto Scaling

Este exemplo exclui a tag especificada do grupo do Auto Scaling especificado.

```
aws autoscaling delete-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research
```

Este comando não produz saída.

Para obter mais informações, consulte Marcar instâncias e grupos de Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DeleteTagsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a tag especificada do grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Exemplo 3: Com o Powershell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Para obter detalhes da API, consulte DeleteTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag especificada do grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Exemplo 3: Com o Powershell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Para obter detalhes da API, consulte DeleteTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAccountLimits`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAccountLimits`.

CLI

AWS CLI

Para descrever os limites de sua conta do Amazon EC2 Auto Scaling

Este exemplo descreve os limites AWS do Amazon EC2 Auto Scaling para sua conta.

```
aws autoscaling describe-account-limits
```

Saída:

```
{
    "NumberOfLaunchConfigurations": 5,
    "MaxNumberOfLaunchConfigurations": 100,
    "NumberOfAutoScalingGroups": 3,
    "MaxNumberOfAutoScalingGroups": 20
}
```

Para obter mais informações, consulte Service quotas do Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeAccountLimitsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve os limites de recursos do Auto Scaling para sua AWS conta.

```
Get-ASAccountLimit
```

Saída:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- Para obter detalhes da API, consulte DescribeAccountLimitsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve os limites de recursos do Auto Scaling para sua AWS conta.

```
Get-ASAccountLimit
```

Saída:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- Para obter detalhes da API, consulte DescribeAccountLimitsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAdjustmentTypes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAdjustmentTypes`.

CLI

AWS CLI

Para descrever os tipos de ajuste de escala disponíveis

Este exemplo descreve os tipos de ajuste disponíveis.

```
aws autoscaling describe-adjustment-types
```

Saída:

```
{
    "AdjustmentTypes": [
        {
            "AdjustmentType": "ChangeInCapacity"
        },
        {
            "AdjustmentType": "ExactCapacity"
        },
```

```
        {
            "AdjustmentType": "PercentChangeInCapacity"
        }
    ]
}
```

Para obter mais informações, consulte Tipos de ajuste da escalabilidade no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeAdjustmentTypesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve os tipos de ajuste que são compatíveis com o Auto Scaling.

```
Get-ASAdjustmentType
```

Saída:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Para obter detalhes da API, consulte DescribeAdjustmentTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve os tipos de ajuste que são compatíveis com o Auto Scaling.

```
Get-ASAdjustmentType
```

Saída:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Para obter detalhes da API, consulte DescribeAdjustmentTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeAutoScalingGroups`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAutoScalingGroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
```

```
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como descrever o grupo do Auto Scaling especificado

Este exemplo descreve o grupo do Auto Scaling especificado.

```
aws autoscaling describe-auto-scaling-groups \
    --auto-scaling-group-names my-asg
```

Saída:

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-1234567890abcde12"
            },
            "MinSize": 0,
            "MaxSize": 1,
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a",
                "us-west-2b",
                "us-west-2c"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-06905f55584de02da",
                    "InstanceType": "t2.micro",
                    "AvailabilityZone": "us-west-2a",
```

```
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "ProtectedFromScaleIn": false,
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-1234567890abcde12"
                    }
                }
            ],
            "CreatedTime": "2023-10-28T02:39:22.152Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-
c934b782",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN":"arn",
            "TrafficSources": []
        }
    ]
}
```

Exemplo 2: como descrever os primeiros 100 grupos do Auto Scaling especificados

Este exemplo descreve os grupos do Auto Scaling especificados. Ele permite especificar até cem nomes de grupos.

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 100 \
    --auto-scaling-group-names "group1" "group2" "group3" "group4"
```

Consulte uma amostra de saída no exemplo 1.

Exemplo 3: como descrever um grupo do Auto Scaling na região especificada

Este exemplo descreve até 75 grupos do Auto Scaling na região especificada.

```
aws autoscaling describe-auto-scaling-groups \
```

```
    --max-items 75 \
    --region us-east-1
```

Consulte um exemplo de saída no exemplo 1.

Exemplo 4: como descrever o número especificado do grupo do Auto Scaling

Use a opção `--max-items` para retornar um número específico de grupos do Auto Scaling.

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 1
```

Consulte um exemplo de saída no exemplo 1.

Se a saída incluir um campo `NextToken`, há mais grupos. Para obter os grupos adicionais, use o valor desse campo com a opção `--starting-token` em uma chamada subsequente da seguinte maneira.

```
aws autoscaling describe-auto-scaling-groups \
    --starting-token Z3M3LMPEXAMPLE
```

Consulte um exemplo de saída no exemplo 1.

Exemplo 5: descrever grupos do Auto Scaling que usam configurações de execução

Este exemplo usa a opção `--query` para descrever os grupos do Auto Scaling que usam configurações de execução.

```
aws autoscaling describe-auto-scaling-groups \
    --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

Saída:

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
```

```
        "LaunchConfigurationName": "my-lc",
        "MinSize": 0,
        "MaxSize": 1,
        "DesiredCapacity": 1,
        "DefaultCooldown": 300,
        "AvailabilityZones": [
            "us-west-2a",
            "us-west-2b",
            "us-west-2c"
        ],
        "LoadBalancerNames": [],
        "TargetGroupARNs": [],
        "HealthCheckType": "EC2",
        "HealthCheckGracePeriod": 0,
        "Instances": [
            {
                "InstanceId": "i-088c57934a6449037",
                "InstanceType": "t2.micro",
                "AvailabilityZone": "us-west-2c",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService",
                "LaunchConfigurationName": "my-lc",
                "ProtectedFromScaleIn": false
            }
        ],
        "CreatedTime": "2023-10-28T02:39:22.152Z",
        "SuspendedProcesses": [],
        "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
        "EnabledMetrics": [],
        "Tags": [],
        "TerminationPolicies": [
            "Default"
        ],
        "NewInstancesProtectedFromScaleIn": false,
        "ServiceLinkedRoleARN":"arn",
        "TrafficSources": []
    }
]
```

Para obter mais informações, consulte a saída da AWS CLI do filtro no Guia do usuário da interface de linha de AWS comando.

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());

                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroupsreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function describeAutoScalingGroups($autoScalingGroupNames)
    {
        return $this->autoScalingClient->describeAutoScalingGroups([
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista os nomes de seus grupos do Auto Scaling.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Saída:

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Exemplo 2: Este exemplo descreve o grupo do Auto Scaling especificado.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Saída:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

Exemplo 3: Este exemplo descreve os grupos do Auto Scaling especificados.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

Exemplo 4: Este exemplo descreve as instâncias do Auto Scaling do grupo do Auto Scaling especificado.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Exemplo 5: Este exemplo descreve todos os seus grupos do Auto Scaling.

```
Get-ASAutoScalingGroup
```

Exemplo 6: Este LaunchTemplate exemplo descreve o grupo de Auto Scaling especificado. Este exemplo pressupõe que as “Opções de compra de instância” estejam definidas como “Aderir ao modelo de lançamento”. Caso essa opção esteja definida como “Combinar opções de compra e tipos de instância”, LaunchTemplate pode ser acessada usando a propriedade MixedInstancesPolicy.LaunchTemplate "”.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Saída:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os nomes de seus grupos do Auto Scaling.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Saída:

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Exemplo 2: Este exemplo descreve o grupo do Auto Scaling especificado.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Saída:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

Exemplo 3: Este exemplo descreve os grupos do Auto Scaling especificados.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

Exemplo 4: Este exemplo descreve as instâncias do Auto Scaling do grupo do Auto Scaling especificado.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Exemplo 5: Este exemplo descreve todos os seus grupos do Auto Scaling.

```
Get-ASAutoScalingGroup
```

Exemplo 6: Este LaunchTemplate exemplo descreve o grupo de Auto Scaling especificado. Este exemplo pressupõe que as “Opções de compra de instância” estejam definidas como “Aderir ao modelo de lançamento”. Caso essa opção esteja definida como “Combinar opções de compra e tipos de instância”, LaunchTemplate pode ser acessada usando a propriedade MixedInstancesPolicy.LaunchTemplate "”.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Saída:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
        """
        Gets information about an Auto Scaling group.

        :param group_name: The name of the group to look up.
        :return: A dictionary with information about the group if found,
 otherwise None.
        :raises ClientError: If there is an error describing the Auto Scaling
 group.
```

```
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_groups"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupNames=[group_name])
            groups = []
            for response in response_iterator:
                groups.extend(response.get("AutoScalingGroups", []))

            logger.info(
                f"Successfully retrieved information for Auto Scaling group
{group_name}."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to describe Auto Scaling group {group_name}.")
            if error_code == "ResourceContentionFault":
                logger.error(
                    "There is a conflict with another operation that is modifying
the "
                    f"Auto Scaling group '{group_name}' Please try again later."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
        else:
            return groups[0] if len(groups) > 0 else None
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroupsReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroupsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_group_names TYPE /aws1/
cl_ascautoscgroupnames_w=>tt_autoscalinggroupnames.
    DATA lo_group_name TYPE REF TO /aws1/cl_ascautoscgroupnames_w.

    " Example: iv_group_name = 'my-auto-scaling-group'

    TRY.
        " Build group names parameter
        CREATE OBJECT lo_group_name
          EXPORTING
            iv_value = iv_group_name.
        APPEND lo_group_name TO lt_group_names.

        " Describe the Auto Scaling group
        DATA(lo_output) = ao_asc->describeautoscalinggroups(
          it_autoscalinggroupnames = lt_group_names ).

        " Return the first (and only) group in the result
        DATA(lt_groups) = lo_output->get_autoscalinggroups( ).
        IF lines( lt_groups ) > 0.
          READ TABLE lt_groups INDEX 1 INTO DATA(lo_group).
          oo_output = lo_group.
        ENDIF.

        MESSAGE 'Auto Scaling group information retrieved successfully' TYPE 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroupsreferência da API AWS SDK for SAP ABAP.

Use **`DescribeAutoScalingInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAutoScalingInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
```

```
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstancesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const
 Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
 &instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();

        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como descrever uma ou mais instâncias

Este exemplo descreve a instância especificada.

```
aws autoscaling describe-auto-scaling-instances \
    --instance-ids i-06905f55584de02da
```

Saída:

```
{
    "AutoScalingInstances": [
        {
            "InstanceId": "i-06905f55584de02da",
            "InstanceType": "t2.micro",
            "AutoScalingGroupName": "my-asg",
            "AvailabilityZone": "us-west-2b",
            "LifecycleState": "InService",
            "HealthStatus": "HEALTHY",
            "ProtectedFromScaleIn": false,
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-1234567890abcde12",
                "LaunchTemplateName": "my-launch-template",
                "Version": "1"
            }
        }
    ]
}
```

Exemplo 2: como descrever uma ou mais instâncias

Este exemplo usa a opção `--max-items` para especificar quantas instâncias devem ser retornadas pela chamada.

```
aws autoscaling describe-auto-scaling-instances \
    --max-items 1
```

Se a saída incluir um campo `NextToken`, há mais instâncias. Para obter as instâncias adicionais, use o valor desse campo com a opção `--starting-token` em uma chamada subsequente da seguinte maneira.

```
aws autoscaling describe-auto-scaling-instances \
    --starting-token Z3M3LMPEXAMPLE
```

Consulte uma amostra de saída no exemplo 1.

Exemplo 3: descrever instâncias que usam configurações de execução

Este exemplo usa a opção `--query` para descrever instâncias que usam configurações de execução.

```
aws autoscaling describe-auto-scaling-instances \
```

```
    --query 'AutoScalingInstances[?LaunchConfigurationName!=`null`]'
```

Saída:

```
[
    {
        "InstanceId": "i-088c57934a6449037",
        "InstanceType": "t2.micro",
        "AutoScalingGroupName": "my-asg",
        "AvailabilityZone": "us-west-2c",
        "LifecycleState": "InService",
        "HealthStatus": "HEALTHY",
        "LaunchConfigurationName": "my-lc",
        "ProtectedFromScaleIn": false
    }
]
```

Para obter mais informações, consulte a saída da AWS CLI do filtro no Guia do usuário da interface de linha de AWS comando.

- Para obter detalhes da API, consulte DescribeAutoScalingInstancesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
       try {
           DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                   .builder()
                   .instanceIds(id)
                   .build();
```

```
            DescribeAutoScalingInstancesResponse response = autoScalingClient

 .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
 response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
 instance.lifecycleState());
            }

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

- Para obter detalhes da API, consulte DescribeAutoScalingInstancesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
```

```
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAutoScalingInstancesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function describeAutoScalingInstances($instanceIds)
    {
        return $this->autoScalingClient->describeAutoScalingInstances([
            'InstanceIds' => $instanceIds
        ]);
    }
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstancesa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista os IDs de suas instâncias do Auto Scaling.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Saída:

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

Exemplo 2: Este exemplo descreve a instância do Auto Scaling especificada.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Saída:

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Exemplo 3: Este exemplo descreve duas instâncias do Auto Scaling especificadas.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

Exemplo 4: Este exemplo descreve as instâncias do Auto Scaling do grupo do Auto Scaling especificado.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

Exemplo 5: Este exemplo descreve todas as instâncias do Auto Scaling.

```
Get-ASAutoScalingInstance
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os IDs de suas instâncias do Auto Scaling.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Saída:

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

Exemplo 2: Este exemplo descreve a instância do Auto Scaling especificada.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Saída:

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Exemplo 3: Este exemplo descreve duas instâncias do Auto Scaling especificadas.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

Exemplo 4: Este exemplo descreve as instâncias do Auto Scaling do grupo do Auto Scaling especificado.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

Exemplo 5: Este exemplo descreve todas as instâncias do Auto Scaling.

```
Get-ASAutoScalingInstance
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_instances(self, instance_ids: List[str]) -> List[Dict[str,
 Any]]:
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: A list of dictionaries with information about each instance,
                 or an empty list if none are found.
        :raises ClientError: If there is an error describing the instances.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_instances"
            )
            response_iterator = paginator.paginate(InstanceIds=instance_ids)

            instances = []
            for response in response_iterator:
                instances.extend(response.get("AutoScalingInstances", []))

            logger.info(f"Successfully described instances: {instance_ids}")
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe instances {instance_ids}. Error code:
 {error_code}, Message: {err.response['Error']['Message']}"
            )
            raise
        else:
            return instances
```

- Para obter detalhes da API, consulte a DescribeAutoScalingInstancesReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
```

```
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }
```

- Para obter detalhes da API, consulte a DescribeAutoScalingInstancesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Example: it_instance_ids contains a list of instance IDs

    TRY.
        DATA(lo_output) = ao_asc->describeautoscalinginstances(
          it_instanceids = it_instance_ids ).

        ot_output = lo_output->get_autoscalinginstances( ).

        MESSAGE 'Auto Scaling instances information retrieved successfully' TYPE
'I'.
```

```
      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAutoScalingInstancesreferência da API AWS SDK for SAP ABAP.

Usar **`DescribeAutoScalingNotificationTypes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAutoScalingNotificationTypes`.

CLI

AWS CLI

Para descrever os tipos de notificação disponíveis

Este exemplo descreve os tipos de notificação disponíveis.

```
aws autoscaling describe-auto-scaling-notification-types
```

Saída:

```
{
    "AutoScalingNotificationTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCH",
        "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
        "autoscaling:EC2_INSTANCE_TERMINATE",
        "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
        "autoscaling:TEST_NOTIFICATION"
    ]
}
```

Para obter mais informações, consulte Obter notificações do Amazon SNS quando o grupo de Auto Scaling é dimensionado no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeAutoScalingNotificationTypesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista os tipos de notificação compatíveis com o Auto Scaling.

```
Get-ASAutoScalingNotificationType
```

Saída:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Para obter detalhes da API, consulte DescribeAutoScalingNotificationTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os tipos de notificação compatíveis com o Auto Scaling.

```
Get-ASAutoScalingNotificationType
```

Saída:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Para obter detalhes da API, consulte DescribeAutoScalingNotificationTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLaunchConfigurations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLaunchConfigurations`.

CLI

AWS CLI

Exemplo 1: descrever a configuração de execução especificada

Este exemplo descreve a configuração de execução especificada.

```
aws autoscaling describe-launch-configurations \
    --launch-configuration-names my-launch-config
```

Saída:

```
{
    "LaunchConfigurations": [
        {
            "LaunchConfigurationName": "my-launch-config",
            "LaunchConfigurationARN": "arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:98d3b196-4cf9-4e88-8ca1-8547c24ced8b:launchConfig
my-launch-config",
            "ImageId": "ami-0528a5175983e7f28",
            "KeyName": "my-key-pair-uswest2",
            "SecurityGroups": [
                "sg-05eaec502fcdadc2e"
            ],
            "ClassicLinkVPCSecurityGroups": [],
            "UserData": "",
            "InstanceType": "t2.micro",
            "KernelId": "",
            "RamdiskId": "",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "SnapshotId": "snap-06c1606ba5ca274b1",
                        "VolumeSize": 8,
                        "VolumeType": "gp2",
                        "DeleteOnTermination": true,
                        "Encrypted": false
                    }
                }
            ],
            "InstanceMonitoring": {
```

```
                "Enabled": true
            },
            "CreatedTime": "2020-10-28T02:39:22.321Z",
            "EbsOptimized": false,
            "AssociatePublicIpAddress": true,
            "MetadataOptions": {
                "HttpTokens": "required",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "disabled"
            }
        }
    ]
}
```

Exemplo 2: descrever um número especificado de configurações de execução

Para retornar um número específico de configurações de execução, use a opção `--max-items`.

```
aws autoscaling describe-launch-configurations \
    --max-items 1
```

Se a saída incluir um campo `NextToken`, há mais configurações de execução. Para obter as configurações de execução adicionais, use o valor desse campo com a opção `--starting-token` em uma chamada subsequente da seguinte maneira.

```
aws autoscaling describe-launch-configurations \
    --starting-token Z3M3LMPEXAMPLE
```

- Para obter detalhes da API, consulte DescribeLaunchConfigurationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista os nomes de suas configurações de execução.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Saída:

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Exemplo 2: Este exemplo descreve a configuração de execução especificada.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Saída:

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                 : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN       : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                               e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName      : my-lc-1
PlacementTenancy             :
RamdiskId                    :
SecurityGroups               : {sg-67ef0308}
SpotPrice                    :
UserData                     :
```

Exemplo 3: Este exemplo descreve as duas configurações de execução especificadas.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Exemplo 4: Este exemplo descreve todas as suas configurações de execução.

```
Get-ASLaunchConfiguration
```

- Para obter detalhes da API, consulte DescribeLaunchConfigurationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os nomes de suas configurações de execução.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Saída:

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Exemplo 2: Este exemplo descreve a configuração de execução especificada.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Saída:

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                 : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN       : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
```

```
                                e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName      : my-lc-1
PlacementTenancy             :
RamdiskId                    :
SecurityGroups               : {sg-67ef0308}
SpotPrice                    :
UserData                     :
```

Exemplo 3: Este exemplo descreve as duas configurações de execução especificadas.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Exemplo 4: Este exemplo descreve todas as suas configurações de execução.

```
Get-ASLaunchConfiguration
```

- Para obter detalhes da API, consulte DescribeLaunchConfigurationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLifecycleHookTypes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLifecycleHookTypes`.

CLI

AWS CLI

Para descrever os tipos de ganchos do ciclo de vida disponíveis

Este exemplo descreve os tipos de ganchos do ciclo de vida disponíveis.

```
aws autoscaling describe-lifecycle-hook-types
```

Saída:

```
{
    "LifecycleHookTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCHING",
        "autoscaling:EC2_INSTANCE_TERMINATING"
    ]
```

```
}
```

- Para obter detalhes da API, consulte DescribeLifecycleHookTypesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista os tipos de gancho do ciclo de vida compatíveis com o Auto Scaling.

```
Get-ASLifecycleHookType
```

Saída:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Para obter detalhes da API, consulte DescribeLifecycleHookTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os tipos de gancho do ciclo de vida compatíveis com o Auto Scaling.

```
Get-ASLifecycleHookType
```

Saída:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Para obter detalhes da API, consulte DescribeLifecycleHookTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLifecycleHooks`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLifecycleHooks`.

CLI

AWS CLI

Para descrever seus ganchos do ciclo de vida

Este exemplo descreve os ganchos do ciclo de vida do grupo do Auto Scaling especificado.

```
aws autoscaling describe-lifecycle-hooks \
    --auto-scaling-group-name my-asg
```

Saída:

```
{
    "LifecycleHooks": [
        {
            "GlobalTimeout": 3000,
            "HeartbeatTimeout": 30,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-launch-hook",
            "DefaultResult": "ABANDON",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING"
        },
        {
            "GlobalTimeout": 6000,
            "HeartbeatTimeout": 60,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-termination-hook",
            "DefaultResult": "CONTINUE",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeLifecycleHooksem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve o gancho do ciclo de vida especificado.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Saída:

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

Exemplo 2: Descreve os ganchos do ciclo de vida do grupo de Auto Scaling especificado.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Exemplo 3: Este exemplo descreve todos ganchos do ciclo de vida os todos seus grupos do Auto Scaling.

```
Get-ASLifecycleHook
```

- Para obter detalhes da API, consulte DescribeLifecycleHooksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o gancho do ciclo de vida especificado.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Saída:

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
```

```
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

Exemplo 2: Descreve os ganchos do ciclo de vida do grupo de Auto Scaling especificado.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Exemplo 3: Este exemplo descreve todos ganchos do ciclo de vida os todos seus grupos do Auto Scaling.

```
Get-ASLifecycleHook
```

- Para obter detalhes da API, consulte DescribeLifecycleHooksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLoadBalancers`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLoadBalancers`.

CLI

AWS CLI

Para descrever os Classic Load Balancers de um grupo do Auto Scaling

Esse exemplo descreve os load balancers clássicos do grupo de Auto Scaling especificado.

```
aws autoscaling describe-load-balancers \
    --auto-scaling-group-name my-asg
```

Saída:

```
{
    "LoadBalancers": [
        {
            "State": "Added",
            "LoadBalancerName": "my-load-balancer"
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Descreve os load balancers do grupo de Auto Scaling especificado.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Saída:

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- Para obter detalhes da API, consulte DescribeLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Descreve os load balancers do grupo de Auto Scaling especificado.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Saída:

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- Para obter detalhes da API, consulte DescribeLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeMetricCollectionTypes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeMetricCollectionTypes`.

CLI

AWS CLI

Como descrever os tipos de coleta de métricas disponíveis

Este exemplo descreve os tipos de coleta de métricas disponíveis.

```
aws autoscaling describe-metric-collection-types
```

Saída:

```
{
    "Metrics": [
        {
            "Metric": "GroupMinSize"
        },
        {
            "Metric": "GroupMaxSize"
        },
        {
            "Metric": "GroupDesiredCapacity"
        },
        {
            "Metric": "GroupInServiceInstances"
        },
        {
            "Metric": "GroupInServiceCapacity"
        },
        {
            "Metric": "GroupPendingInstances"
        },
        {
            "Metric": "GroupPendingCapacity"
        },
        {
            "Metric": "GroupTerminatingInstances"
        },
        {
            "Metric": "GroupTerminatingCapacity"
        },
        {
            "Metric": "GroupStandbyInstances"
```

```
        },
        {
            "Metric": "GroupStandbyCapacity"
        },
        {
            "Metric": "GroupTotalInstances"
        },
        {
            "Metric": "GroupTotalCapacity"
        }
    ],
    "Granularities": [
        {
            "Granularity": "1Minute"
        }
    ]
}
```

Para obter mais informações, consulte Métricas de grupo do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeMetricCollectionTypesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista os tipos de coleta de métricas compatíveis com o Auto Scaling.

```
(Get-ASMetricCollectionType).Metrics
```

Saída:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
```

```
GroupStandbyInstances
GroupTotalInstances
```

Exemplo 2: Este exemplo lista as granularidades correspondentes.

```
(Get-ASMetricCollectionType).Granularities
```

Saída:

```
Granularity
-----------
1Minute
```

- Para obter detalhes da API, consulte DescribeMetricCollectionTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os tipos de coleta de métricas compatíveis com o Auto Scaling.

```
(Get-ASMetricCollectionType).Metrics
```

Saída:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Exemplo 2: Este exemplo lista as granularidades correspondentes.

```
(Get-ASMetricCollectionType).Granularities
```

Saída:

```
Granularity
-----------
1Minute
```

- Para obter detalhes da API, consulte DescribeMetricCollectionTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeNotificationConfigurations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeNotificationConfigurations`.

CLI

AWS CLI

Exemplo 1: Como descrever as configurações de notificação de um grupo especificado.

Este exemplo descreve as configurações de notificação do grupo do Auto Scaling especificado.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg
```

Saída:

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

Para obter mais informações, consulte Obter notificações do Amazon SNS quando o grupo de Auto Scaling é dimensionado no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 1: Como descrever um número especificado de configurações de notificação.

Para retornar um número especificado de configurações de notificação, use o parâmetro `max-items`.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-auto-scaling-group \
    --max-items 1
```

Saída:

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

Se a saída incluir um campo `NextToken`, há mais configurações de notificação. Para obter as notificações adicionais, use o valor desse campo com o parâmetro `starting-token` em uma chamada subsequente da seguinte maneira.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

Para obter mais informações, consulte Obter notificações do Amazon SNS quando o grupo de Auto Scaling é dimensionado no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeNotificationConfigurationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo descreve as ações de notificação associadas ao grupo de Auto Scaling especificado.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Saída:

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

Exemplo 2: Este exemplo descreve as ações de notificação associadas a todos os seus grupos do Auto Scaling.

```
Get-ASNotificationConfiguration
```

- Para obter detalhes da API, consulte DescribeNotificationConfigurationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo descreve as ações de notificação associadas ao grupo de Auto Scaling especificado.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Saída:

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
```

```
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

Exemplo 2: Este exemplo descreve as ações de notificação associadas a todos os seus grupos do Auto Scaling.

```
Get-ASNotificationConfiguration
```

- Para obter detalhes da API, consulte DescribeNotificationConfigurationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribePolicies`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribePolicies`.

CLI

AWS CLI

Exemplo 1: como descrever as escalas do Auto Scaling especificado

Este exemplo descreve as politicas de escalabilidade do grupo do Auto Scaling especificado.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg
```

Saída:

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "alb1000-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:3065d9c8-9969-4bec-
bb6a-3fbe5550fde6:autoScalingGroupName/my-asg:policyName/alb1000-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196",
```

```
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-f96f899d-
b8e7-4d09-a010-c1aaa35da296",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-a010-
c1aaa35da296"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ALBRequestCountPerTarget",
                    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-
alb-target-group/943f017f100becff"
                },
                "TargetValue": 1000.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        },
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "cpu40-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:5fd26f71-39d4-4690-82a9-
b8515c45cdde:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-139f9789-37b9-42ad-bea5-b5b147d7f473",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-139f9789-37b9-42ad-
bea5-b5b147d7f473"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-bd681c67-
fc18-4c56-8468-fb8e413009c9",
```

```
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-bd681c67-fc18-4c56-8468-
fb8e413009c9"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 40.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        }
    ]
}
```

Para obter mais informações, consulte Escalabilidade Dinâmica no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: Como descrever as políticas de escalabilidade de um nome especificado.

Para retornar políticas de escalabilidade específicas, use a opção `--policy-names`.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --policy-names cpu40-target-tracking-scaling-policy
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Escalabilidade dinâmica no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 3: Como descrever várias políticas de escalabilidade.

Use a opção `--max-items` para retornar um número específico de politicas.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

Consulte uma amostra de saída no exemplo 1.

Se a saída incluir um campo `NextToken`, use o valor desse campo com a opção `--starting-token` em uma chamada subsequente para obter as políticas adicionais.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg --starting-
token Z3M3LMPEXAMPLE
```

Para obter mais informações, consulte Escalabilidade dinâmica no Manual do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribePoliciesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo descreve o grupo do Auto Scaling especificado.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Saída:

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

Exemplo 2: Este exemplo descreve duas políticas especificadas para o grupo do Auto Scaling especificado.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

Exemplo 3: Este exemplo descreve todas as políticas para todos os seus grupos do Auto Scaling.

```
Get-ASPolicy
```

- Para obter detalhes da API, consulte DescribePoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo descreve o grupo do Auto Scaling especificado.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Saída:

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

Exemplo 2: Este exemplo descreve duas políticas especificadas para o grupo do Auto Scaling especificado.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

Exemplo 3: Este exemplo descreve todas as políticas para todos os seus grupos do Auto Scaling.

```
Get-ASPolicy
```

- Para obter detalhes da API, consulte DescribePoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeScalingActivities`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeScalingActivities`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
```

```
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
_amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }
```

- Para obter detalhes da API, consulte DescribeScalingActivitiesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
```

```
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
                          << outcome.GetError().GetMessage()
                          << std::endl;

            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
```

- Para obter detalhes da API, consulte DescribeScalingActivitiesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como descrever as ações de escalabilidade do grupo especificado

Este exemplo descreve as ações de escalabilidade do grupo do Auto Scaling especificado.

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg
```

Saída:

```
{
    "Activities": [
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

Para mais informações, consulte Verificar uma ação de escalabilidade para um grupo do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: como descrever as ações de escalabilidade de um grupo excluído

Adicione a opção `--include-deleted-groups` para descrever as ações de escalabilidade depois que o grupo do Auto Scaling for excluído.

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg \
    --include-deleted-groups
```

Saída:

```
{
```

```
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "Description": "Launching a new EC2 instance.  Status Reason: Your
 Spot request price of 0.001 is lower than the minimum required Spot request
 fulfillment price of 0.0031. Launching EC2 instance failed.",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of
 AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired
 capacity from 0 to 3.  At 2021-01-13T20:47:27Z an instance was started in
 response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 3.",
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than
 the minimum required Spot request fulfillment price of 0.0031. Launching EC2
 instance failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        }
    ]
}
```

Para obter mais informações, consulte Solucionar problemas do Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 3: como descrever um número especificado de ações de escalabilidade

Use a opção `--max-items` para retornar um número específico de ações.

```
aws autoscaling describe-scaling-activities \
    --max-items 1
```

Saída:

```
{
    "Activities": [
```

```
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

Se a saída incluir um campo `NextToken`, há mais ações. Para obter as ações adicionais, use o valor desse campo com a opção `--starting-token` em uma chamada subsequente da seguinte maneira.

```
aws autoscaling describe-scaling-activities \
    --starting-token Z3M3LMPEXAMPLE
```

Para mais informações, consulte Verificar uma ação de escalabilidade para um grupo do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeScalingActivitiesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void describeScalingActivities(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
 DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " +
 activity.activityId());
                System.out.println("The activity details are " +
 activity.details());
            }

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

- Para obter detalhes da API, consulte DescribeScalingActivitiesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
```

```
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeScalingActivitiesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function describeScalingActivities($autoScalingGroupName)
    {
        return $this->autoScalingClient->describeScalingActivities([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- Para obter detalhes da API, consulte DescribeScalingActivitiesa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve as ações de escalabilidade das últimas seis semanas do grupo do Auto Scaling especificado.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Saída:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
 group desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the
 capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity
 from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
```

```
StatusMessage        :
```

Exemplo 2: Este exemplo descreve a atividade de escala especificada.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Exemplo 3: Este exemplo descreve as atividades de escala das últimas seis semanas para todos os seus grupos do Auto Scaling.

```
Get-ASScalingActivity
```

- Para obter detalhes da API, consulte DescribeScalingActivitiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve as ações de escalabilidade das últimas seis semanas do grupo do Auto Scaling especificado.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Saída:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
 group desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the
 capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
```

```
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity
 from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

Exemplo 2: Este exemplo descreve a atividade de escala especificada.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Exemplo 3: Este exemplo descreve as atividades de escala das últimas seis semanas para todos os seus grupos do Auto Scaling.

```
Get-ASScalingActivity
```

- Para obter detalhes da API, consulte DescribeScalingActivitiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
```

```
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_scaling_activities(self, group_name: str) -> List[Dict[str,
Any]]:
        """
        Gets information about scaling activities for the group. Scaling
activities
        are things like instances stopping or starting in response to user
requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: A list of dictionaries representing the scaling activities for
the
                 group, ordered with the most recent activity first.
        :raises ClientError: If there is an error describing the scaling
activities.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_scaling_activities"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupName=group_name)
            activities = []
            for response in response_iterator:
                activities.extend(response.get("Activities", []))

            logger.info(
                f"Successfully described scaling activities for group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe scaling activities for group '{group_name}'.
Error code: {error_code}, Message: {err.response['Error']['Message']}"
            )
```

```
            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
        else:
            return activities
```

- Para obter detalhes da API, consulte a DescribeScalingActivitiesReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
```

```
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }
```

- Para obter detalhes da API, consulte a DescribeScalingActivitiesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
    DATA(lo_output) = ao_asc->describescalingactivities(
      iv_autoscalinggroupname = iv_group_name ).

    ot_output = lo_output->get_activities( ).

    MESSAGE 'Scaling activities retrieved successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeScalingActivitiesreferência da API AWS SDK for SAP ABAP.

Usar **`DescribeScalingProcessTypes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeScalingProcessTypes`.

CLI

AWS CLI

Como descrever os tipos de processo disponíveis

Este exemplo descreve os tipos de processo disponíveis.

```
aws autoscaling describe-scaling-process-types
```

Saída:

```
{
    "Processes": [
        {
            "ProcessName": "AZRebalance"
        },
        {
            "ProcessName": "AddToLoadBalancer"
        },
        {
            "ProcessName": "AlarmNotification"
        },
        {
            "ProcessName": "HealthCheck"
        },
        {
            "ProcessName": "InstanceRefresh"
        },
        {
            "ProcessName": "Launch"
        },
        {
            "ProcessName": "ReplaceUnhealthy"
        },
        {
            "ProcessName": "ScheduledActions"
        },
        {
            "ProcessName": "Terminate"
        }
    ]
}
```

Para obter mais informações, consulte Suspensão e retomada dos processos de escalabilidade no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeScalingProcessTypesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista os tipos de processo compatíveis com o Auto Scaling.

```
Get-ASScalingProcessType
```

Saída:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Para obter detalhes da API, consulte DescribeScalingProcessTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os tipos de processo compatíveis com o Auto Scaling.

```
Get-ASScalingProcessType
```

Saída:

```
ProcessName
-----------
AZRebalance
```

```
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Para obter detalhes da API, consulte DescribeScalingProcessTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeScheduledActions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeScheduledActions`.

CLI

AWS CLI

Exemplo 1: Como descrever todas as ações programadas.

Este exemplo descreve todas as ações programadas.

```
aws autoscaling describe-scheduled-actions
```

Saída:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
```

```
            "TimeZone": "America/New_York"
        }
    ]
}
```

Para obter mais informações, consulte Escalabilidade programada no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: Como descrever as ações agendadas do grupo especificado

Para descrever as ações programadas para um grupo do Auto Scaling específico, use a opção `--auto-scaling-group-name`.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg
```

Saída:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Para obter mais informações, consulte Escalabilidade programada no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 3: Como descrever a ação programada especificada.

Para descrever uma ação programada específica, use a opção `--scheduled-action-names`.

```
aws autoscaling describe-scheduled-actions \
    --scheduled-action-names my-recurring-action
```

Saída:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Para obter mais informações, consulte Escalabilidade programada no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 4: Como descrever ações programadas com um horário de início especificado.

Para descrever as ações programadas que começam em um horário específico, use a opção `--start-time`.

```
aws autoscaling describe-scheduled-actions \
    --start-time "2023-12-01T04:00:00Z"
```

Saída:

```
{
```

```
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Para obter mais informações, consulte Escalabilidade programada no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 5: Como descrever ações programadas que terminam em um horário especificado.

Para descrever as ações agendadas que terminam em um horário específico, use a opção `--end-time` .

```
aws autoscaling describe-scheduled-actions \
    --end-time "2023-12-01T04:00:00Z"
```

Saída:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
```

```
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Para obter mais informações, consulte Escalabilidade programada no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 6: como descrever um número especificado de ações agendadas

Use a opção `--max-items` para retornar um número específico de ações agendadas.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

Saída:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Se a saída incluir um campo `NextToken`, há mais ações agendadas. Para obter as ações agendadas, use o valor desse campo com a opção `--starting-token` em uma chamada subsequente da seguinte maneira.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

Para obter mais informações, consulte Escalabilidade programada no Manual do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeScheduledActionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve as ações de escalabilidade agendadas do grupo do Auto Scaling especificado.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Saída:

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

Exemplo 2: Este exemplo descreve as ações de escala programada especificadas.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

Exemplo 3: Este exemplo descreve as ações de escala programadas que começam no horário especificado.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Exemplo 4: Este exemplo descreve as ações de escala programadas que terminam no horário especificado.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Exemplo 5: Este exemplo descreve as ações de escala programadas para todos os seus grupos do Auto Scaling.

```
Get-ASScheduledAction
```

- Para obter detalhes da API, consulte DescribeScheduledActionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve as ações de escalabilidade agendadas do grupo do Auto Scaling especificado.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Saída:

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
```

```
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

Exemplo 2: Este exemplo descreve as ações de escala programada especificadas.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

Exemplo 3: Este exemplo descreve as ações de escala programadas que começam no horário especificado.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Exemplo 4: Este exemplo descreve as ações de escala programadas que terminam no horário especificado.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Exemplo 5: Este exemplo descreve as ações de escala programadas para todos os seus grupos do Auto Scaling.

```
Get-ASScheduledAction
```

- Para obter detalhes da API, consulte DescribeScheduledActionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTags`.

CLI

AWS CLI

Como descrever todas as tags

Este exemplo descreve todas as tags.

```
aws autoscaling describe-tags
```

Saída:

```
{
    "Tags": [
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "Research",
            "Key": "Dept"
        },
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "WebServer",
            "Key": "Role"
        }
    ]
}
```

Para obter mais informações, consulte Marcar instâncias e grupos de Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: Como descrever tags para um grupo especificado.

Use a opção para descrever as tags de um grupo do Auto Scaling específico use a opção `--filters`.

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

Para obter mais informações, consulte Marcar instâncias e grupos de Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 3: Para descrever o número específico de tags

Use a opção `--max-items` para retornar um número específico de tags.

```
aws autoscaling describe-tags \
```

```
    --max-items 1
```

Se a saída incluir um campo `NextToken`, há mais tags. Para obter os tags adicionais, use o valor desse campo com a opção `--starting-token` em uma chamada subsequente da seguinte maneira.

```
aws autoscaling describe-tags \
    --filters Name=auto-scaling-group,Values=my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

Para obter mais informações, consulte Marcar instâncias e grupos de Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve as tags com um valor-chave de “myTag” ou “myTag2”. Os valores possíveis para o nome do filtro são “grupo do auto scaling”, “chave”, “valor” e “propagar na execução”. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Saída:

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro para o parâmetro Filter.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Exemplo 3: Este exemplo descreve todas as tags para tudos seus grupos do Auto Scaling.

```
Get-ASTag
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve as tags com um valor-chave de “myTag” ou “myTag2”. Os valores possíveis para o nome do filtro são “grupo do auto scaling”, “chave”, “valor” e “propagar na execução”. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Saída:

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro para o parâmetro Filter.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Exemplo 3: Este exemplo descreve todas as tags para tudos seus grupos do Auto Scaling.

```
Get-ASTag
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTerminationPolicyTypes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTerminationPolicyTypes`.

CLI

AWS CLI

Como descrever os tipos de políticas de encerramento disponíveis

Este exemplo descreve os tipos de políticas de encerramento disponíveis.

```
aws autoscaling describe-termination-policy-types
```

Saída:

```
{
    "TerminationPolicyTypes": [
        "AllocationStrategy",
        "ClosestToNextInstanceHour",
        "Default",
        "NewestInstance",
```

```
        "OldestInstance",
        "OldestLaunchConfiguration",
        "OldestLaunchTemplate"
    ]
}
```

Para obter mais informações, consulte Controle de quais instâncias do Auto Scaling são terminadas durante a redução da escala horizontalmente no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DescribeTerminationPolicyTypesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista as políticas de terminação que são compatíveis com o Auto Scaling.

```
Get-ASTerminationPolicyType
```

Saída:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Para obter detalhes da API, consulte DescribeTerminationPolicyTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as políticas de terminação que são compatíveis com o Auto Scaling.

```
Get-ASTerminationPolicyType
```

Saída:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Para obter detalhes da API, consulte DescribeTerminationPolicyTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **DetachInstances** com uma CLI

Os exemplos de código a seguir mostram como usar o `DetachInstances`.

CLI

AWS CLI

Para desvincular uma instância de um grupo do Auto Scaling existente

Este exemplo desvincula a instância especificada do grupo do Auto Scaling especificado.

```
aws autoscaling detach-instances \
    --instance-ids i-030017cfa84b20135 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

Saída:

```
{
    "Activities": [
        {
            "ActivityId": "5091cb52-547a-47ce-a236-c9ccbc2cb2c9",
            "AutoScalingGroupName": "my-asg",
            "Description": "Detaching EC2 instance: i-030017cfa84b20135",
            "Cause": "At 2020-10-31T17:35:04Z instance i-030017cfa84b20135 was
 detached in response to a user request, shrinking the capacity from 2 to 1.",
            "StartTime": "2020-04-12T15:02:16.179Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
```

```
        }
    ]
}
```

- Para obter detalhes da API, consulte DetachInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo desvincula a instância especificada do grupo do Auto Scaling especificado e diminui a capacidade desejada para que o Auto Scaling não inicie uma instância substituta.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
 in response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo desvincula a instância especificada do grupo de Auto Scaling especificado sem diminuir a capacidade desejada. O Auto Scaling inicia uma nova instância de substituição.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
 in response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes da API, consulte DetachInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo desvincula a instância especificada do grupo do Auto Scaling especificado e diminui a capacidade desejada para que o Auto Scaling não inicie uma instância substituta.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
 in response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo desvincula a instância especificada do grupo de Auto Scaling especificado sem diminuir a capacidade desejada. O Auto Scaling inicia uma nova instância de substituição.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
 in response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes da API, consulte DetachInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DetachLoadBalancers`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DetachLoadBalancers`.

CLI

AWS CLI

Como desvincula um Classic Load Balancer de um grupo do Auto Scaling

Este exemplo desvincula o Classic Load Balancer especificado do grupo do Auto Scaling especificado.

```
aws autoscaling detach-load-balancers \
    --load-balancer-names my-load-balancer \
```

```
    --auto-scaling-group-name my-asg
```

Este comando não produz saída.

Para obter mais informações, consulte Anexar um balanceador de carga ao seu grupo Auto Scaling no Guia do Usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DetachLoadBalancersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo desvincula o balanceador de carga especificado do grupo de Auto Scaling especificado.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte DetachLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo desvincula o balanceador de carga especificado do grupo de Auto Scaling especificado.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte DetachLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DisableMetricsCollection`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DisableMetricsCollection`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DisableMetricsCollectiona Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
            autoScalingClient.DisableMetricsCollection(request);

    if (outcome.IsSuccess()) {
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

    }
```

- Para obter detalhes da API, consulte DisableMetricsCollectiona Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como desabilitar a coleta de métricas de um grupo do Auto Scaling

Este exemplo desativa a coleta da métrica `GroupDesiredCapacity` do grupo do Auto Scaling especificado.

```
aws autoscaling disable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity
```

Este comando não produz saída.

Para obter mais informações, consulte CloudWatch Métricas de monitoramento para seus grupos e instâncias do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte DisableMetricsCollectionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DisableMetricsCollectiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- Para obter detalhes da API, consulte a DisableMetricsCollectionreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function disableMetricsCollection($autoScalingGroupName)
{
    return $this->autoScalingClient->disableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Para obter detalhes da API, consulte DisableMetricsCollectiona Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Example 1: Desativa o monitoramento de métricas especificadas para o grupo de Auto Scaling.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
 @("GroupMinSize", "GroupMaxSize")
```

Exemplo 2: Esse exemplo desativa o monitoramento de todas as métricas do grupo do Auto Scaling especificado.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte DisableMetricsCollectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Example 1: Desativa o monitoramento de métricas especificadas para o grupo de Auto Scaling.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
 @("GroupMinSize", "GroupMaxSize")
```

Exemplo 2: Esse exemplo desativa o monitoramento de todas as métricas do grupo do Auto Scaling especificado.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte DisableMetricsCollectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def disable_metrics(self, group_name: str) -> Dict[str, Any]:
        """
        Stops CloudWatch metric collection for the Auto Scaling group.
```

```
        :param group_name: The name of the group.
        :return: A dictionary with the response from disabling the metrics
collection.
        :raises ClientError: If there is an error disabling metrics collection.
        """
        try:
            response = self.autoscaling_client.disable_metrics_collection(
                AutoScalingGroupName=group_name
            )
            logger.info(
                f"Successfully disabled metrics collection for group
'{group_name}'."
            )
            return response
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
```

- Para obter detalhes da API, consulte a DisableMetricsCollectionReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

- Para obter detalhes da API, consulte a DisableMetricsCollectionreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Example: iv_group_name = 'my-auto-scaling-group'

    TRY.
        ao_asc->disablemetricscollection(
          iv_autoscalinggroupname = iv_group_name ).

        MESSAGE 'Metrics collection disabled successfully' TYPE 'I'.
```

```
    CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
      RAISE EXCEPTION lo_contention.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
      RAISE EXCEPTION lo_generic_exception.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DisableMetricsCollectionreferência da API AWS SDK for SAP ABAP.

Use **`EnableMetricsCollection`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `EnableMetricsCollection`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Enable the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> EnableMetricsCollectionAsync(string groupName)
{
    var listMetrics = new List<string>
        {
            "GroupMaxSize",
        };
```

```
        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte EnableMetricsCollectiona Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
```

```
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Para obter detalhes da API, consulte EnableMetricsCollectiona Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como habilitar a coleta de métricas do grupo do Auto Scaling

Este exemplo habilita a coleta de dados do grupo do Auto Scaling especificado.

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --granularity "1Minute"
```

Este comando não produz saída.

Para obter mais informações, consulte CloudWatch Métricas de monitoramento para seus grupos e instâncias do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: como coletar dados da métrica especificada de um grupo de Auto Scaling

Use a opção `--metrics` para coletar dados de uma métrica específica.

```
aws autoscaling enable-metrics-collection \
```

```
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity --granularity "1Minute"
```

Este comando não produz saída.

Para obter mais informações, consulte CloudWatch Métricas de monitoramento para seus grupos e instâncias do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte EnableMetricsCollectionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .granularity("1Minute")
                   .build();

           autoScalingClient.enableMetricsCollection(collectionRequest);
           System.out.println("The enable metrics collection operation was
successful");

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

- Para obter detalhes da API, consulte EnableMetricsCollectiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- Para obter detalhes da API, consulte a EnableMetricsCollectionreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
    {
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- Para obter detalhes da API, consulte EnableMetricsCollectiona Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Ativa o monitoramento de métricas especificadas para o grupo de Auto Scaling.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Exemplo 2: Este exemplo permite o monitoramento de todas as métricas do grupo do Auto Scaling especificado.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Para obter detalhes da API, consulte EnableMetricsCollectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Ativa o monitoramento de métricas especificadas para o grupo de Auto Scaling.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Exemplo 2: Este exemplo permite o monitoramento de todas as métricas do grupo do Auto Scaling especificado.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Para obter detalhes da API, consulte EnableMetricsCollectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str,
 Any]:
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling
 activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        :return: A dictionary with the response from enabling the metrics
 collection.
        :raises ClientError: If there is an error enabling metrics collection.
        """
        try:
            response = self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
 Granularity="1Minute"
            )
            logger.info(
```

```
                f"Successfully enabled metrics for Auto Scaling group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            elif error_code == "InvalidParameterCombination":
                logger.error(
                    f"The combination of parameters provided for enabling metrics
on '{group_name}' is not valid. "
                    "Please check the parameters and try again."
                )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a EnableMetricsCollectionReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;
```

- Para obter detalhes da API, consulte a EnableMetricsCollectionreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " Example: iv_group_name = 'my-auto-scaling-group'
   " Example: it_metrics contains list of metrics like 'GroupMinSize',
'GroupMaxSize', etc.

   TRY.
       ao_asc->enablemetricscollection(
         iv_autoscalinggroupname = iv_group_name
         it_metrics = it_metrics
         iv_granularity = '1Minute' ).

       MESSAGE 'Metrics collection enabled successfully' TYPE 'I'.

     CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
```

```
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Para obter detalhes da API, consulte a EnableMetricsCollectionreferência da API AWS SDK for SAP ABAP.

Usar **`EnterStandby`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EnterStandby`.

CLI

AWS CLI

Como colocar instâncias no modo de espera

Este exemplo coloca a instância especificada no modo de espera. Isso é útil para atualizar ou solucionar problemas de uma instância que está atualmente em serviço.

```
aws autoscaling enter-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

Saída:

```
{
    "Activities": [
        {
            "ActivityId": "ffa056b4-6ed3-41ba-ae7c-249dfae6eba1",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance to Standby: i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:31:00Z instance i-061c63c5eb45f0416 was
 moved to standby in response to a user request, shrinking the capacity from 1 to
 0.",
            "StartTime": "2020-10-31T20:31:00.949Z",
            "StatusCode": "InProgress",
            "Progress": 50,
```

```
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

Para obter mais informações, consulte Amazon EC2 Auto Scaling instância de ciclo de vida no Manual do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte EnterStandbyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo coloca a instância especificada em modo de espera e diminui a capacidade desejada para que o Auto Scaling não inicie uma instância substituta.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo coloca a instância especificada em modo de espera sem diminuir a capacidade desejada. O Auto Scaling inicia uma nova instância de substituição.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes da API, consulte EnterStandbyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo coloca a instância especificada em modo de espera e diminui a capacidade desejada para que o Auto Scaling não inicie uma instância substituta.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
```

```
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo coloca a instância especificada em modo de espera sem diminuir a capacidade desejada. O Auto Scaling inicia uma nova instância de substituição.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes da API, consulte EnterStandbyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ExecutePolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ExecutePolicy`.

CLI

AWS CLI

Como executar uma política de escalabilidade

Este exemplo executa a política de escalabilidade chamada `my-step-scale-out-policy` para o grupo do Auto Scaling especificado.

```
aws autoscaling execute-policy \
    --auto-scaling-group-name my-asg \
    --policy-name  my-step-scale-out-policy \
    --metric-value 95 \
    --breach-threshold 80
```

Este comando não produz saída.

Para mais informações, consulte as políticas de escalonamento por passos e escalonamento simples no Guia do Usuário de Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte ExecutePolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo executa a política especificada para o grupo do Auto Scaling especificado.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Exemplo 2: Este exemplo executa a política especificada para o grupo do Auto Scaling especificado, depois de aguardar a conclusão do período de espera.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Para obter detalhes da API, consulte ExecutePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo executa a política especificada para o grupo do Auto Scaling especificado.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Exemplo 2: Este exemplo executa a política especificada para o grupo do Auto Scaling especificado, depois de aguardar a conclusão do período de espera.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Para obter detalhes da API, consulte ExecutePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ExitStandby`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ExitStandby`.

CLI

AWS CLI

Como tirar instâncias do modo de espera

Esse exemplo move a instância especificada para fora do modo em espera.

```
aws autoscaling exit-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

Saída:

```
{
    "Activities": [
        {
            "ActivityId": "142928e1-a2dc-453a-9b24-b85ad6735928",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance out of Standby:
 i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:32:50Z instance i-061c63c5eb45f0416 was
 moved out of standby in response to a user request, increasing the capacity from
 0 to 1.",
            "StartTime": "2020-10-31T20:32:50.222Z",
            "StatusCode": "PreInService",
            "Progress": 30,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
```

```
}
```

Para obter mais informações, consulte Desvincular temporariamente instâncias do grupo do Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte ExitStandbyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo tira as instâncias especificadas do modo de espera.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Saída:

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
 of standby in response to a user
                       request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- Para obter detalhes da API, consulte ExitStandbyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo tira as instâncias especificadas do modo de espera.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Saída:

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
 of standby in response to a user
                       request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- Para obter detalhes da API, consulte ExitStandbyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutLifecycleHook`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutLifecycleHook`.

CLI

AWS CLI

Exemplo 1: Como criar um gancho do ciclo de vida

Este exemplo cria um gancho do ciclo de vida que será invocado em qualquer instância recém-iniciada, com um tempo limite de 4800 segundos. Isso é útil para manter as instâncias em um estado de espera até que os scripts de dados do usuário terminem ou para invocar uma função AWS Lambda usando. EventBridge

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --heartbeat-timeout 4800
```

Este comando não produz saída. Se já existir um gancho do ciclo de vida com o mesmo nome, ele será substituído pelo novo gancho do ciclo de vida.

Para obter mais informações, consulte Ganchos do ciclo de vida do Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: Como enviar uma mensagem de e-mail do Amazon SNS para notificar sobre as transições de estado da instância.

Este exemplo cria um gancho do ciclo de vida com o tópico do Amazon SNS e o perfil do IAM a serem usados para receber notificações na execução da instância.

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --role-arn arn:aws:iam::123456789012:role/my-auto-scaling-role
```

Este comando não produz saída.

Para obter mais informações, consulte Ganchos do ciclo de vida do Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 3: Como publicar uma mensagem em uma fila do Amazon SQS.

Este exemplo cria um gancho do ciclo de vida que publica uma mensagem com metadados na fila especificada do Amazon SQS.

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sqs:us-west-2:123456789012:my-sqs-queue \
    --role-arn arn:aws:iam::123456789012:role/my-notification-role \
    --notification-metadata "SQS message metadata"
```

Este comando não produz saída.

Para obter mais informações, consulte Ganchos do ciclo de vida do Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte PutLifecycleHookem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo adiciona o gancho do ciclo de vida especificado ao grupo do Auto Scaling especificado.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- Para obter detalhes da API, consulte PutLifecycleHookem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo adiciona o gancho do ciclo de vida especificado ao grupo do Auto Scaling especificado.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- Para obter detalhes da API, consulte PutLifecycleHookem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutNotificationConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutNotificationConfiguration`.

CLI

AWS CLI

Para adicionar uma notificação

Este exemplo adiciona o grupo de destino especificado ao grupo do Auto Scaling especificado.

```
aws autoscaling put-notification-configuration \
```

```
    --auto-scaling-group name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --notification-type autoscaling:TEST_NOTIFICATION
```

Este comando não produz saída.

Para obter mais informações, consulte Obter notificações do Amazon SNS quando o grupo de Auto Scaling é dimensionado no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte PutNotificationConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo configura o grupo do Auto Scaling especificado para enviar uma notificação ao tópico SNS especificado quando ele iniciar instâncias do EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

Exemplo 2: Este exemplo configura o grupo do Auto Scaling especificado para enviar uma notificação ao tópico SNS especificado quando ele iniciar ou encerrar instâncias do EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Para obter detalhes da API, consulte PutNotificationConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo configura o grupo do Auto Scaling especificado para enviar uma notificação ao tópico SNS especificado quando ele iniciar instâncias do EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

Exemplo 2: Este exemplo configura o grupo do Auto Scaling especificado para enviar uma notificação ao tópico SNS especificado quando ele iniciar ou encerrar instâncias do EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Para obter detalhes da API, consulte PutNotificationConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutScalingPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutScalingPolicy`.

CLI

AWS CLI

Para adicionar uma política de escalabilidade com monitoramento do objetivo a um grupo do Auto Scaling

O exemplo `put-scaling-policy` a seguir aplica uma política de escalabilidade de monitoramento de destino ao grupo do Auto Scaling especificado. A saída contém os ARNs e os nomes dos dois CloudWatch alarmes criados em seu nome. Se já existir uma política de escalonamento com o mesmo nome, ela será substituída pela nova política de escalonamento.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
  --policy-name alb1000-target-tracking-scaling-policy \
  --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

Conteúdo de `config.json`:

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
          "PredefinedMetricType": "ALBRequestCountPerTarget",
          "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
```

```
    }
}
```

Saída:

```
{
    "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-
c665-4bfd-aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/alb1000-
target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
        }
    ]
 }
```

Para obter mais exemplos, consulte Exemplos de políticas de escalabilidade para a interface de linha de AWS comando (AWS CLI) no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte PutScalingPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo adiciona a política especificada ao grupo do Auto Scaling especificado. O tipo de ajuste especificado determina como interpretar o ScalingAdjustment parâmetro. Com 'ChangeInCapacity', um valor positivo aumenta a capacidade pelo número especificado de instâncias e um valor negativo diminui a capacidade pelo número especificado de instâncias.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Saída:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Para obter detalhes da API, consulte PutScalingPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo adiciona a política especificada ao grupo do Auto Scaling especificado. O tipo de ajuste especificado determina como interpretar o ScalingAdjustment parâmetro. Com 'ChangeInCapacity', um valor positivo aumenta a capacidade pelo número especificado de instâncias e um valor negativo diminui a capacidade pelo número especificado de instâncias.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Saída:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Para obter detalhes da API, consulte PutScalingPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutScheduledUpdateGroupAction`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutScheduledUpdateGroupAction`.

CLI

AWS CLI

Exemplo 1: adicionar uma ação programada a um grupo de Auto Scaling

Este exemplo adiciona a ação agendada especificada ao grupo do Auto Scaling especificado.

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action \
    --start-time "2023-05-12T08:00:00Z" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

Este comando não produz saída. Se uma ação programada com o mesmo nome já existir, ela será substituída pela nova ação programada.

Para obter mais exemplos, consulte Escalabilidade programada no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: especificar uma programação recorrente

Este exemplo cria uma ação programada para escalar em uma programação recorrente que está programada para ser executada às 00h30 horas nos primeiros dias de janeiro, junho e dezembro de cada ano.

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-recurring-action \
    --recurrence "30 0 1 1,6,12 *" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

Este comando não produz saída. Se uma ação programada com o mesmo nome já existir, ela será substituída pela nova ação programada.

Para obter mais exemplos, consulte Escalabilidade programada no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte PutScheduledUpdateGroupActionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria ou atualiza uma ação programada única para alterar a capacidade desejada na hora de início especificada.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- Para obter detalhes da API, consulte PutScheduledUpdateGroupActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria ou atualiza uma ação programada única para alterar a capacidade desejada na hora de início especificada.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- Para obter detalhes da API, consulte PutScheduledUpdateGroupActionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RecordLifecycleActionHeartbeat`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RecordLifecycleActionHeartbeat`.

CLI

AWS CLI

Para registrar a pulsação de uma ação do ciclo de vida

Este exemplo registra uma pulsação da ação do ciclo de vida para manter a instância em um estado pendente.

```
aws autoscaling record-lifecycle-action-heartbeat \
```

```
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Este comando não produz saída.

Para obter mais informações, consulte Ganchos do ciclo de vida do Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte RecordLifecycleActionHeartbeatem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo registra uma pulsação para a ação de ciclo de vida especificada. Isso mantém a instância em um estado pendente até que você conclua a ação personalizada.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Para obter detalhes da API, consulte RecordLifecycleActionHeartbeatem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo registra uma pulsação para a ação de ciclo de vida especificada. Isso mantém a instância em um estado pendente até que você conclua a ação personalizada.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Para obter detalhes da API, consulte RecordLifecycleActionHeartbeatem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ResumeProcesses`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ResumeProcesses`.

CLI

AWS CLI

Como retomar processos suspensos

Este exemplo retoma o processo de escalabilidade suspenso especificado para o grupo do Auto Scaling especificado.

```
aws autoscaling resume-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

Este comando não produz saída.

Para obter mais informações, consulte Suspensão e retomada dos processos de escalabilidade no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte ResumeProcessesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo reinicia o processo do Auto Scaling especificado para o grupo de Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Exemplo 2: Este exemplo retoma todos os processos suspensos do Auto Scaling para o grupo do Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte ResumeProcessesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo reinicia o processo do Auto Scaling especificado para o grupo de Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Exemplo 2: Este exemplo retoma todos os processos suspensos do Auto Scaling para o grupo do Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte ResumeProcessesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`SetDesiredCapacity`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetDesiredCapacity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
```

```
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte SetDesiredCapacitya Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);
```

```
        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

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

        }
```

- Para obter detalhes da API, consulte SetDesiredCapacitya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como definir a capacidade desejada para um grupo do Auto Scaling

Este exemplo define a capacidade desejada para o grupo do Auto Scaling especificado.

```
aws autoscaling set-desired-capacity \
    --auto-scaling-group-name my-asg \
    --desired-capacity 2 \
    --honor-cooldown
```

Esse comando retornará ao prompt, se houver êxito.

- Para obter detalhes da API, consulte SetDesiredCapacityem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
 String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
 SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

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

- Para obter detalhes da API, consulte SetDesiredCapacitya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
```

```
        println("You set the DesiredCapacity to 2")
    }
}
```

- Para obter detalhes da API, consulte a SetDesiredCapacityreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- Para obter detalhes da API, consulte SetDesiredCapacitya Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Define o tamanho do grupo de Auto Scaling especificado.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Exemplo 2: Este exemplo define o tamanho do grupo do Auto Scaling especificado e aguarda a conclusão do período de espera antes de fazer o escalonamento para o novo tamanho.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -
HonorCooldown $true
```

- Para obter detalhes da API, consulte SetDesiredCapacityem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Define o tamanho do grupo de Auto Scaling especificado.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Exemplo 2: Este exemplo define o tamanho do grupo do Auto Scaling especificado e aguarda a conclusão do período de espera antes de fazer o escalonamento para o novo tamanho.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -
HonorCooldown $true
```

- Para obter detalhes da API, consulte SetDesiredCapacityem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client
```

```
    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
keep the
        number of running instances equal to the desired capacity.

        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
                f"Successfully set desired capacity of {capacity} for Auto
Scaling group '{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
            )
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'. "
                    "Please wait for the activity to complete before attempting
to set the desired capacity."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Para obter detalhes da API, consulte a SetDesiredCapacityReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
       // 7. SetDesiredCapacity: set desired capacity to 2.
       //   Wait for a second instance to launch.
       let update_group = self
           .autoscaling
           .set_desired_capacity()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .desired_capacity(capacity)
           .send()
           .await;
       if let Err(err) = update_group {
           return Err(ScenarioError::new(
               format!("Failed to update group to desired capacity
({capacity}))").as_str(),
               &err,
           ));
       }
       Ok(())
   }
```

- Para obter detalhes da API, consulte a SetDesiredCapacityreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_capacity = 2

TRY.
    ao_asc->setdesiredcapacity(
      iv_autoscalinggroupname = iv_group_name
      iv_desiredcapacity = iv_capacity
      iv_honorcooldown = abap_false ).

    MESSAGE 'Desired capacity set successfully' TYPE 'I'.

  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Para obter detalhes da API, consulte a SetDesiredCapacityreferência da API AWS SDK for SAP ABAP.

Usar **`SetInstanceHealth`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetInstanceHealth`.

CLI

AWS CLI

Como definir o status de integridade de uma instância

Define o status da integridade da instância especificada para `Unhealthy`.

```
aws autoscaling set-instance-health \
    --instance-id i-061c63c5eb45f0416 \
    --health-status Unhealthy
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SetInstanceHealthem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo define o status da instância especificada como “Não saudável”, tirando-a de serviço. O Auto Scaling termina e substitui a instância.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Exemplo 2: Este exemplo define o status da instância especificada como “Saudável”, mantendo-a em serviço. Definir um período de carência da verificação de integridade para um grupo do Auto Scaling

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- Para obter detalhes da API, consulte SetInstanceHealthem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo define o status da instância especificada como “Não saudável”, tirando-a de serviço. O Auto Scaling termina e substitui a instância.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Exemplo 2: Este exemplo define o status da instância especificada como “Saudável”, mantendo-a em serviço. Definir um período de carência da verificação de integridade para um grupo do Auto Scaling

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- Para obter detalhes da API, consulte SetInstanceHealthem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SetInstanceProtection`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetInstanceProtection`.

CLI

AWS CLI

Exemplo 1: habilitar a configuração de proteção de instância para uma instância

Este exemplo habilita a proteção de instância na a instância especificada.

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg --protected-from-scale-in
```

Este comando não produz saída.

Exemplo 2: desativar a configuração de proteção de instância para uma instância

Este exemplo desabilita a proteção de instância na instância especificada.

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --no-protected-from-scale-in
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SetInstanceProtectionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo habilita a proteção de instância na a instância especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Exemplo 2: Este exemplo desabilita a proteção de instância na instância especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- Para obter detalhes da API, consulte SetInstanceProtectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo habilita a proteção de instância na a instância especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Exemplo 2: Este exemplo desabilita a proteção de instância na instância especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- Para obter detalhes da API, consulte SetInstanceProtectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SuspendProcesses`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SuspendProcesses`.

CLI

AWS CLI

Como suspender os processos de ajuste de escala automático

Este exemplo suspende o processo de escalabilidade especificado para o grupo do Auto Scaling especificado.

```
aws autoscaling suspend-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

Este comando não produz saída.

Para obter mais informações, consulte Suspensão e retomada dos processos de escalabilidade no Guia do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte SuspendProcessesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo suspende o processo de escalabilidade especificado para o grupo do Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess
 "AlarmNotification"
```

Exemplo 2: Este exemplo suspende todos os processos do Auto Scaling para o grupo do Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte SuspendProcessesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo suspende o processo de escalabilidade especificado para o grupo do Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess
 "AlarmNotification"
```

Exemplo 2: Este exemplo suspende todos os processos do Auto Scaling para o grupo do Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Para obter detalhes da API, consulte SuspendProcessesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`TerminateInstanceInAutoScalingGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TerminateInstanceInAutoScalingGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
```

```
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
```

```
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como encerrar uma instância em um grupo do Auto Scaling

Este exemplo encerra a instância especificada do grupo do Auto Scaling especificado sem atualizar o tamanho do grupo. O Amazon EC2 Auto Scaling executa uma instância de substituição após o encerramento da instância especificada.

```
aws autoscaling terminate-instance-in-auto-scaling-group \
    --instance-id i-061c63c5eb45f0416 \
    --no-should-decrement-desired-capacity
```

Saída:

```
{
    "Activities": [
        {
            "ActivityId": "8c35d601-793c-400c-fcd0-f64a27530df7",
            "AutoScalingGroupName": "my-asg",
            "Description": "Terminating EC2 instance: i-061c63c5eb45f0416",
            "Cause": "",
            "StartTime": "2020-10-31T20:34:25.680Z",
            "StatusCode": "InProgress",
            "Progress": 0,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroupem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
 autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
 TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);
```

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

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- Para obter detalhes da API, consulte a TerminateInstanceInAutoScalingGroupreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function terminateInstanceInAutoScalingGroup(
        $instanceId,
        $shouldDecrementDesiredCapacity = true,
        $attempts = 0
    ) {
        try {
            return $this->autoScalingClient-
>terminateInstanceInAutoScalingGroup([
                'InstanceId' => $instanceId,
                'ShouldDecrementDesiredCapacity' =>
 $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
 $attempts < 5) {
                error_log("Cannot terminate an instance while it is still
 pending. Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroupa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo encerra a instância especificada e diminui a capacidade desejada de seu grupo do Auto Scaling para que o Auto Scaling não inicie uma instância substituta.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo encerra a instância especificada sem diminuir a capacidade desejada de seu grupo do Auto Scaling. Auto Scaling inicia uma nova instância de substituição.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
```

```
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo encerra a instância especificada e diminui a capacidade desejada de seu grupo do Auto Scaling para que o Auto Scaling não inicie uma instância substituta.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo encerra a instância especificada sem diminuir a capacidade desejada de seu grupo do Auto Scaling. Auto Scaling inicia uma nova instância de substituição.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
```

```
    ) -> Dict[str, Any]:
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired
capacity
                                  of the group. When passing True for this
parameter,
                                  you can stop an instance without having a
replacement
                                  instance start when the desired capacity
threshold is
                                  crossed.
        :return: A dictionary containing details of the scaling activity that
occurs
                 in response to this action.
        :raises ClientError: If there is an error terminating the instance.
        """
        try:
            response =
self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
ShouldDecrementDesiredCapacity=decrease_capacity
            )
            logger.info(f"Successfully terminated instance {instance_id}.")
            return response["Activity"]

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to terminate instance {instance_id}.")
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    "A scaling activity is currently in progress for the Auto
Scaling group "
                    f"associated with instance '{instance_id}'. "
                    "Please wait for the activity to complete before attempting
to terminate the instance."
                )
            elif error_code == "ResourceInUse":
                logger.error(
                    f"The instance '{instance_id}' or an associated resource is
currently in use "
                    "and cannot be terminated. "
```

```
                    "Ensure the instance is not involved in any ongoing processes
and try again."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Para obter detalhes da API, consulte a TerminateInstanceInAutoScalingGroupReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
```

```
                    "There was a problem terminating an instance",
                    &err,
                ))
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
```

```
    }
```

- Para obter detalhes da API, consulte a TerminateInstanceInAutoScalingGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Example: iv_instance_id = 'i-1234567890abcdef0'
" Example: iv_decrease_capacity = abap_true

TRY.
    DATA(lo_output) = ao_asc->terminateinstinautoscgroup(
      iv_instanceid = iv_instance_id
      iv_shoulddecrementdesiredcap = iv_decrease_capacity ).

    oo_output = lo_output->get_activity( ).

    MESSAGE 'Instance terminated successfully' TYPE 'I'.

  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Para obter detalhes da API, consulte a TerminateInstanceInAutoScalingGroupreferência da API AWS SDK for SAP ABAP.

Use **`UpdateAutoScalingGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateAutoScalingGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
```

```
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
_amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
```

```
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

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

        }
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como atualizar os limites de tamanho de um grupo do Auto Scaling

Este exemplo atualiza o grupo do Auto Scaling especificado com um tamanho mínimo de 2 e máximo de 10.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --min-size 2 \
    --max-size 10
```

Este comando não produz saída.

Para obter mais informações, consulte Definir limites de escalabilidade para seu grupo do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 2: como adicionar verificações de integridade do Elastic Load Balancing e especificar quais zonas de disponibilidade e sub-redes usar

Este exemplo atualiza o grupo do Auto Scaling especificado para adicionar verificações de integridade do Elastic Load Balancing. Esse comando também atualiza o valor de `--vpc-zone-identifier` com uma lista de IDs de sub-rede em várias zonas de disponibilidade.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Este comando não produz saída.

Para obter mais informações, consulte Elastic Load Balancing e Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 3: como atualizar o grupo de posicionamento e a política de encerramento

Este exemplo atualiza o grupo de posicionamento e a política de encerramento que devem ser usados.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --placement-group my-placement-group \
    --termination-policies "OldestInstance"
```

Este comando não produz saída.

Para obter mais informações, consulte Grupos do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

Exemplo 4: como usar a versão mais recente do modelo de execução

Este exemplo atualiza o grupo do Auto Scaling especificado para que use a versão mais recente do modelo de execução especificado.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest'
```

Este comando não produz saída.

Para obter mais informações, consulte Modelos de execução no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 5: como usar uma versão específica do modelo de execução

Este exemplo atualiza o grupo do Auto Scaling especificado para que use uma versão específica do modelo de execução em vez da versão mais recente ou padrão.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

Este comando não produz saída.

Para obter mais informações, consulte Modelos de execução no Manual do usuário do Amazon EC2 Auto Scaling.

Exemplo 6: como definir uma política de instâncias mistas e habilitar o rebalanceamento de capacidade

Este exemplo atualiza o grupo do Auto Scaling especificado para que use uma política de instâncias mistas e permita o rebalanceamento de capacidade. Essa estrutura permite especificar grupos com Spot e On-Demand capacidades e usar modelos de lançamento diferentes para arquiteturas diferentes.

```
aws autoscaling update-auto-scaling-group \
    --cli-input-json file://~/config.json
```

Conteúdo de `config.json`:

```
{
    "AutoScalingGroupName": "my-asg",
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template-for-x86",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c6g.large",
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateName": "my-launch-template-for-arm",
                        "Version": "$Latest"
                    }
```

```
                },
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    }
}
```

Este comando não produz saída.

Para obter mais informações, consulte Grupos de Auto Scaling com vários tipos de instância e opções de compra no Manual do usuário do Amazon EC2 Auto Scaling.

- Para obter detalhes da API, consulte UpdateAutoScalingGroupem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
```

```
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
" + groupName);

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

- Para obter detalhes da API, consulte UpdateAutoScalingGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}
```

- Para obter detalhes da API, consulte a UpdateAutoScalingGroupreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function updateAutoScalingGroup($autoScalingGroupName, $args)
{
    if (array_key_exists('MaxSize', $args)) {
        $maxSize = ['MaxSize' => $args['MaxSize']];
    } else {
        $maxSize = [];
    }
    if (array_key_exists('MinSize', $args)) {
        $minSize = ['MinSize' => $args['MinSize']];
    } else {
        $minSize = [];
    }
    $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
    $parameters = array_merge($parameters, $minSize, $maxSize);
    return $this->autoScalingClient->updateAutoScalingGroup($parameters);
}
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroupa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo atualiza o tamanho mínimo e máximo do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Exemplo 2: Este exemplo atualiza o período de espera padrão do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Exemplo 3: Este exemplo atualiza as zonas de disponibilidade do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Exemplo 4: Este exemplo atualiza o grupo do Auto Scaling especificado para usar verificações de integridade do Elastic Load Balancing.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo atualiza o tamanho mínimo e máximo do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Exemplo 2: Este exemplo atualiza o período de espera padrão do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Exemplo 3: Este exemplo atualiza as zonas de disponibilidade do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Exemplo 4: Este exemplo atualiza o grupo do Auto Scaling especificado para usar verificações de integridade do Elastic Load Balancing.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def update_group(self, group_name: str, **kwargs: Any) -> None:
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        :return: None
        :raises ClientError: If there is an error updating the Auto Scaling
 group.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
```

```
                AutoScalingGroupName=group_name, **kwargs
            )
            logger.info(f"Successfully updated Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to update Auto Scaling group {group_name}.")
            if error_code == "ResourceInUse":
                logger.error(
                    "The Auto Scaling group '%s' is currently in use and cannot
 be modified. Please try again later.",
                    group_name,
                )
            elif error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
 Scaling group '{group_name}'."
                    "Please wait for the activity to complete before attempting
 to update the group."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Para obter detalhes da API, consulte a UpdateAutoScalingGroupReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(),
 Error> {
    client
```

```
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
        .send()
        .await?;

    println!("Updated AutoScaling group");

    Ok(())
}
```

- Para obter detalhes da API, consulte a UpdateAutoScalingGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Example: iv_group_name = 'my-auto-scaling-group'
    " Example: iv_max_size = 5

    TRY.
        ao_asc->updateautoscalinggroup(
          iv_autoscalinggroupname = iv_group_name
          iv_maxsize = iv_max_size
          iv_minsize = iv_min_size ).

        MESSAGE 'Auto Scaling group updated successfully' TYPE 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
        RAISE EXCEPTION lo_activity_in_progress.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
```

```
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateAutoScalingGroupreferência da API AWS SDK for SAP ABAP.

Cenários para o Auto Scaling usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Auto Scaling com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Aurora ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie e gerencie um serviço resiliente usando um AWS SDK

Crie e gerencie um serviço resiliente usando um AWS SDK

Os exemplos de código a seguir mostram como criar um serviço da Web com carga balanceada que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de saúde atualizando AWS Systems Manager os parâmetros.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
```

```
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
```

```
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
```

```
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
```

```
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");
```

```
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
```

```
                            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
```

```
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");
```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
```

```
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
 a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
 granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
```

```
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
```

```
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
```

```
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
 List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
```

```
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
```

```
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
```

```
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
```

```
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
```

```
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
 calling computer.
```

```
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                     "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
```

```
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
```

```
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
```

```
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
```

```
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
```

```
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
```

```
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
```

```
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
```

```
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
```

```
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
```

```
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
```

```
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs

- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";
```

```
    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your
 account, we can clean up all AWS resources
                that were created for this demo.
```

```
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
```

```
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

       System.out.println("Press Enter when you're ready.");
       in.nextLine();
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Creating and populating a DynamoDB table named " +
tableName);
       Database database = new Database();
       database.createTable(tableName, fileName);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("""
               Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
               This script starts a Python web server defined in the `server.py`
script. The web server
               listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
               For demo purposes, this server is run as the root user. In
production, the best practice is to
               run a web server, such as Apache, with least-privileged
credentials.

               The template also defines an IAM policy that each instance uses
to assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
```

```
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
```

```
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
       autoScaler.terminateInstance(badInstanceId);

       System.out.println("""
               Even while the instance is terminating and the new instance is
starting, sending a GET
               request to the web service continues to get a successful
recommendation response because
               the load balancer routes requests to the healthy instances. After
the replacement instance
               starts and reports as healthy, it is included in the load
balancing rotation.
               Note that terminating and replacing an instance typically takes
several minutes, during which time you
               can see the changing health check status until the new instance
is running and healthy.
               """);

       demoChoices(loadBalancer);
       System.out.println(
               "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       demoChoices(loadBalancer);
       paramHelper.reset();
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
```

```
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());
```

```
                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
 balancer targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
```

```
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
```

```
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
```

```
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();
```

```
            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
```

```
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
```

```
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

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

```
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }
```

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

```
   // Verify this computer can successfully send a GET request to the load
balancer
   // endpoint.
   public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
       boolean success = false;
       int retries = 3;
       CloseableHttpClient httpClient = HttpClients.createDefault();

       // Create an HTTP GET request to the ELB.
       HttpGet httpGet = new HttpGet("http://" + elbDnsName);
       try {
           while ((!success) && (retries > 0)) {
               // Execute the request and get the response.
               HttpResponse response = httpClient.execute(httpGet);
               int statusCode = response.getStatusLine().getStatusCode();
               System.out.println("HTTP Status Code: " + statusCode);
               if (statusCode == 200) {
                   success = true;
               } else {
                   retries--;
                   System.out.println("Got connection error from load balancer
endpoint, retrying...");
                   TimeUnit.SECONDS.sleep(15);
               }
           }

       } catch (org.apache.http.conn.HttpHostConnectException e) {
           System.out.println(e.getMessage());
       }

       System.out.println("Status.." + success);
       return success;
   }

   /*
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
```

```
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
```

```
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
        return false;
```

```
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();
```

```
            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
```

```
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
```

```
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
```

```
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Criar etapas para implantar todos os recursos.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
```

```
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
```

```
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
```

```
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
```

```
    new ScenarioOutput(
      "createdKeyPair",
      MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
    ),
    new ScenarioOutput(
      "creatingInstancePolicy",
      MESSAGES.creatingInstancePolicy.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      ),
    ),
    new ScenarioAction("createInstancePolicy", async (state) => {
      const client = new IAMClient({});
      const {
        Policy: { Arn },
      } = await client.send(
        new CreatePolicyCommand({
          PolicyName: NAMES.instancePolicyName,
          PolicyDocument: readFileSync(
            join(RESOURCES_PATH, "instance_policy.json"),
          ),
        }),
      );
      state.instancePolicyArn = Arn;
    }),
    new ScenarioOutput("createdInstancePolicy", (state) =>
      MESSAGES.createdInstancePolicy
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
    ),
    new ScenarioOutput(
      "creatingInstanceRole",
      MESSAGES.creatingInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      ),
    ),
    new ScenarioAction("createInstanceRole", () => {
      const client = new IAMClient({});
      return client.send(
        new CreateRoleCommand({
          RoleName: NAMES.instanceRoleName,
          AssumeRolePolicyDocument: readFileSync(
            join(ROOT, "assume-role-policy.json"),
```

```
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
```

```
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
```

```
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
```

```
        LaunchTemplate: {
          LaunchTemplateName: NAMES.launchTemplateName,
          Version: "$Default",
        },
        MinSize: 3,
        MaxSize: 3,
      }),
    ),
  );
}),
new ScenarioOutput(
  "createdAutoScalingGroup",
  /**
   * @param {{ availabilityZoneNames: string[] }} state
   */
  (state) =>
    MESSAGES.createdAutoScalingGroup
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
      .replace(
        "${AVAILABILITY_ZONE_NAMES}",
        state.availabilityZoneNames.join(", "),
      ),
),
new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
  type: "confirm",
}),
new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
new ScenarioAction("getVpc", async (state) => {
  const client = new EC2Client({});
  const { Vpcs } = await client.send(
    new DescribeVpcsCommand({
      Filters: [{ Name: "is-default", Values: ["true"] }],
    }),
  );
  state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
  MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
  const client = new EC2Client({});
  const { Subnets } = await client.send(
```

```
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
```

```
    }),
    new ScenarioOutput(
      "createdLoadBalancerTargetGroup",
      MESSAGES.createdLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      ),
    ),
    new ScenarioOutput(
      "creatingLoadBalancer",
      MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
    ),
    new ScenarioAction("createLoadBalancer", async (state) => {
      const client = new ElasticLoadBalancingV2Client({});
      const { LoadBalancers } = await client.send(
        new CreateLoadBalancerCommand({
          Name: NAMES.loadBalancerName,
          Subnets: state.subnets,
        }),
      );
      state.loadBalancerDns = LoadBalancers[0].DNSName;
      state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
      await waitUntilLoadBalancerAvailable(
        { client },
        { Names: [NAMES.loadBalancerName] },
      );
    }),
    new ScenarioOutput("createdLoadBalancer", (state) =>
      MESSAGES.createdLoadBalancer
        .replace("${LB_NAME}", NAMES.loadBalancerName)
        .replace("${DNS_NAME}", state.loadBalancerDns),
    ),
    new ScenarioOutput(
      "creatingListener",
      MESSAGES.creatingLoadBalancerListener
        .replace("${LB_NAME}", NAMES.loadBalancerName)
        .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
    ),
    new ScenarioAction("createListener", async (state) => {
      const client = new ElasticLoadBalancingV2Client({});
      const { Listeners } = await client.send(
        new CreateListenerCommand({
          LoadBalancerArn: state.loadBalancerArn,
          Protocol: state.targetGroupProtocol,
```

```
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
```

```
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
```

```
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
```

```
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Criar etapas para executar a demonstração.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
```

```
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
```

```
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
```

```
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
```

```
    ...statusSteps,
    new ScenarioInput(
      "brokenDependencyConfirmation",
      MESSAGES.demoBrokenDependencyConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("brokenDependency", async (state) => {
      if (!state.brokenDependencyConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        state.badTableName = `fake-table-${Date.now()}`;
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmTableNameKey,
            Value: state.badTableName,
            Overwrite: true,
            Type: "String",
          }),
        );
      }
    }),
    new ScenarioOutput("testBrokenDependency", (state) =>
      MESSAGES.demoTestBrokenDependency.replace(
        "${TABLE_NAME}",
        state.badTableName,
      ),
    ),
    ...statusSteps,
    new ScenarioInput(
      "staticResponseConfirmation",
      MESSAGES.demoStaticResponseConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("staticResponse", async (state) => {
      if (!state.staticResponseConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmFailureResponseKey,
            Value: "static",
            Overwrite: true,
```

```
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
```

```
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
        InstanceIds: [state.targetInstance.InstanceId],
        DocumentName: "AWS-RunShellScript",
        Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
```

```
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
```

```
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
```

```
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
```

```
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
```

```
    }),
  );

  return InstanceProfile;
}
```

Criar etapas para destruir todos os recursos.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
```

```
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
```

```
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
```

```
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
```

```
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
```

```
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
```

```
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
```

```
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
```

```
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
```

```
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
```

```
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
```

```
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
 state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
```

```
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
```

```
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
 DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
```

```
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)
```

```
        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
 from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
```

```
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
```

```
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()
```

```
        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()
```

```
        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
```

```
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
```

```
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
```

```
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
 definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
 {policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
```

```
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
```

```
        :param aws_managed_policies: A tuple of AWS-managed policy names to
attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
{role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
```

```
                                        use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn
```

```
    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.
```

```
        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
```

```
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
```

```
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
```

```
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
```

```
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
```

```
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
```

```
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
```

```
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
```

```
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
```

```
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
```

```
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
 "
                    "Ensure that your AWS IAM user or role has the correct
 permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
 parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

```
    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
```

```
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
```

```
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
```

```
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
 {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
```

```
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
```

```
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
```

```
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.
```

```
        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer
```

```
    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
```

```
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
```

```
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
```

```
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
 parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
```

```
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
```

```
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Crie uma classe que envolva as ações do Systems Manager.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
```

```
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer

- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Exemplos de código para AWS Batch usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Batch com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS Batch Guia do usuário — Mais informações sobre AWS Batch.
- AWS Batch Referência da API — Detalhes sobre todas as AWS Batch ações disponíveis.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Batch usar AWS SDKs
  - Hello AWS Batch
  - Aprenda as noções básicas do AWS Batch com um AWS SDK
  - Ações para AWS Batch usar AWS SDKs
    - Use CreateComputeEnvironmentcom um AWS SDK ou CLI
    - Use CreateJobQueuecom um AWS SDK ou CLI
    - Use DeleteComputeEnvironmentcom um AWS SDK ou CLI
    - Use DeleteJobQueuecom um AWS SDK ou CLI
    - Use DeregisterJobDefinitioncom um AWS SDK ou CLI
    - Use DescribeComputeEnvironmentscom um AWS SDK ou CLI
    - Use DescribeJobQueuescom um AWS SDK ou CLI
    - Use DescribeJobscom um AWS SDK ou CLI
    - Use ListJobsPaginatorcom um AWS SDK
    - Use RegisterJobDefinitioncom um AWS SDK ou CLI
    - Use SubmitJobcom um AWS SDK ou CLI
    - Use UpdateComputeEnvironmentcom um AWS SDK ou CLI
    - Use UpdateJobQueuecom um AWS SDK ou CLI
- Cenários para AWS Batch usar AWS SDKs
  - Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

Exemplos básicos para AWS Batch usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Batch com AWS SDKs.

Exemplos

- Hello AWS Batch

- Aprenda as noções básicas do AWS Batch com um AWS SDK
- Ações para AWS Batch usar AWS SDKs
  - Use CreateComputeEnvironmentcom um AWS SDK ou CLI
  - Use CreateJobQueuecom um AWS SDK ou CLI
  - Use DeleteComputeEnvironmentcom um AWS SDK ou CLI
  - Use DeleteJobQueuecom um AWS SDK ou CLI
  - Use DeregisterJobDefinitioncom um AWS SDK ou CLI
  - Use DescribeComputeEnvironmentscom um AWS SDK ou CLI
  - Use DescribeJobQueuescom um AWS SDK ou CLI
  - Use DescribeJobscom um AWS SDK ou CLI
  - Use ListJobsPaginatorcom um AWS SDK
  - Use RegisterJobDefinitioncom um AWS SDK ou CLI
  - Use SubmitJobcom um AWS SDK ou CLI
  - Use UpdateComputeEnvironmentcom um AWS SDK ou CLI
  - Use UpdateJobQueuecom um AWS SDK ou CLI

Hello AWS Batch

O exemplo de código a seguir mostra como começar a usar o AWS Batch.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloBatch {
    private static BatchAsyncClient batchClient;

    public static void main(String[] args) {
        List<JobSummary> jobs = listJobs("my-job-queue");
        jobs.forEach(job ->
            System.out.printf("Job ID: %s, Job Name: %s, Job Status: %s%n",
                job.jobId(), job.jobName(), job.status())
        );
    }

    public static List<JobSummary> listJobs(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or
 empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
 getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response ->
 {
            jobSummaries.addAll(response.jobSummaryList());
        });

        future.join();
        return jobSummaries;
    }
```

```
    private static BatchAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)  // Increase max concurrency to handle more
 simultaneous connections.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
 call attempt timeout.
            .retryPolicy(RetryPolicy.builder()  // Add a retry policy to handle
 transient errors.
                .numRetries(3)  // Number of retry attempts.
                .build())
            .build();

        if (batchClient == null) {
            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }
}
```

- Para obter detalhes da API, consulte a lista JobsPaginator na Referência AWS SDK for Java 2.x da API.

Aprenda as noções básicas do AWS Batch com um AWS SDK

O exemplo de código a seguir mostra como:

- Crie um ambiente AWS Batch computacional.

- Verificar o status do ambiente de computação.
- Configure uma fila de AWS Batch trabalhos e uma definição de trabalhos.
- Registrar uma definição de tarefa.
- Envie um AWS Batch Job.
- Obter uma lista de trabalhos aplicáveis à fila de trabalhos.
- Conferir o status do trabalho.
- Exclua AWS Batch recursos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS Batch recursos.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.ClientException;
import
 software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Subnet;
import software.amazon.awssdk.services.ec2.model.Vpc;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * NOTE
 * This scenario submits a job that pulls a Docker image named echo-text from
 Amazon ECR to Amazon Fargate.
 *
 * To place this Docker image on Amazon ECR, run the following Basics scenario.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/
ecr
 *
 */
public class BatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    // Define two stacks used in this Basics Scenario.
    private static final String ROLES_STACK = "RolesStack";
    private static String defaultSubnet;
    private static String defaultSecurityGroup;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchScenario.class);

    public static void main(String[] args) throws InterruptedException {

        BatchActions batchActions = new BatchActions();
        Scanner scanner = new Scanner(System.in);
        String computeEnvironmentName = "my-compute-environment";
        String jobQueueName = "my-job-queue";
```

```
        String jobDefinitionName = "my-job-definition";


        // See the NOTE in this Java code example (at start).
        String dockerImage = "dkr.ecr.us-east-1.amazonaws.com/echo-text:echo-
text";

        logger.info("""
            AWS Batch is a fully managed batch processing service that
 dynamically provisions the required compute
            resources for batch computing workloads. The Java V2
 `BatchAsyncClient` allows
            developers to automate the submission, monitoring, and management of
 batch jobs.

            This scenario provides an example of setting up a compute
 environment, job queue and job definition,
            and then submitting a job.

            This scenario submits a job that pulls a Docker image named echo-text
 from Amazon ECR to Amazon Fargate.

            To place this Docker image on Amazon ECR, run the following Basics
 scenario.

            https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/
example_code/ecr

            Let's get started...

            You have two choices:

            1 - Run the entire program.
            2 - Delete an existing Compute Environment (created from a previous
 execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                logger.info("Continuing with the program...");
               // logger.info("");
                break;
```

```
            } else if (input.trim().equalsIgnoreCase("2")) {
                String jobQueueARN = String.valueOf(batchActions.
describeJobQueueAsync(computeEnvironmentName));
                if (!jobQueueARN.isEmpty()) {
                    batchActions.disableJobQueueAsync(jobQueueARN);
                    countdown(1);
                    batchActions.deleteJobQueueAsync(jobQueueARN);
                }

                try {

batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                        .exceptionally(ex -> {
                            logger.info("Disable compute environment failed: " +
ex.getMessage());
                            return null;
                        })
                        .join();
                } catch (CompletionException ex) {
                    logger.info("Failed to disable compute environment: " +
ex.getMessage());
                }
                countdown(2);

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                return;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
        System.out.println(DASHES);

        waitForInputToContinue(scanner);
        // Get an AWS Account id used to retrieve the docker image from Amazon
ECR.
        // Create a single-element array to store the `accountId` value.
        String[] accId = new String[1];
        CompletableFuture<String> accountIdFuture = batchActions.getAccountId();
        accountIdFuture.thenAccept(accountId -> {
            logger.info("Account ID: " + accountId);
            accId[0] = accountId;
        }).join();
```

```
        dockerImage = accId[0]+"."+dockerImage;

        // Get a default subnet and default security associated with the default
VPC.
        getSubnetSecurityGroup();

        logger.info("Use AWS CloudFormation to create two IAM roles that are
required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(ROLES_STACK);
        String batchIAMRole = stackOutputs.get("BatchRoleArn");
        String executionRoleARN = stackOutputs.get("EcsRoleArn");

        logger.info("The IAM role needed to interact with AWS Batch is
"+batchIAMRole);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("1. Create a Batch compute environment");
        logger.info("""
            A compute environment is a resource where you can run your batch
jobs.
            After creating a compute environment, you can define job queues and
job definitions to submit jobs for
            execution.

            The benefit of creating a compute environment is it allows you to
easily configure and manage the compute
            resources that will be used to run your Batch jobs. By separating the
compute environment from the job definitions,
            you can easily scale your compute resources up or down as needed,
without having to modify your job definitions.
            This makes it easier to manage your Batch workloads and ensures that
your jobs have the necessary
            compute resources to run efficiently.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateComputeEnvironmentResponse> future =
batchActions.createComputeEnvironmentAsync(computeEnvironmentName, batchIAMRole,
defaultSubnet, defaultSecurityGroup);
```

```
            CreateComputeEnvironmentResponse response = future.join();
            logger.info("Compute Environment ARN: " +
response.computeEnvironmentArn());
        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The compute environment '" +
computeEnvironmentName + "' already exists. Moving on...");
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                    logger.info("An unexpected error occurred: {}", (cause !=
null ? cause.getMessage() : rte.getMessage()));
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Check the status of the "+computeEnvironmentName +"
Compute Environment.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName);
            String status = future.join();
            logger.info("Compute Environment Status: " + status);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return;
```

```
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a job queue");
        logger.info("""
             A job queue is an essential component that helps manage the
execution of your batch jobs.
             It acts as a buffer, where jobs are placed and then scheduled for
execution based on their
             priority and the available resources in the compute environment.
             """);
        waitForInputToContinue(scanner);

        String jobQueueArn = null;
        try {
            CompletableFuture<String> jobQueueFuture =
batchActions.createJobQueueAsync(jobQueueName, computeEnvironmentName);
            jobQueueArn = jobQueueFuture.join();
            logger.info("Job Queue ARN: " + jobQueueArn);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof BatchException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The job queue '" + jobQueueName + "' already
exists. Moving on...");
                    // Retrieve the ARN of the job queue.
                    CompletableFuture<String> jobQueueArnFuture =
batchActions.getJobQueueARN(jobQueueName);
                    jobQueueArn = jobQueueArnFuture.join();
                    logger.info("Job Queue ARN: " + jobQueueArn);
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
```

```
                return; // End the execution
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Register a Job Definition.");
        logger.info("""
            Registering a job in AWS Batch using the Fargate launch type ensures
that all
            necessary parameters, such as the execution role, command to run, and
so on
            are specified and reused across multiple job submissions.

             The job definition pulls a Docker image from Amazon ECR and executes
the Docker image.
            """);

        waitForInputToContinue(scanner);
        String jobARN;
        try {
            String platform = "";
            while (true) {
                logger.info("""
                    On which platform/CPU architecture combination did you build
the Docker image?:
                    1. Windows       X86_64
                    2. Mac or Linux  ARM64
                    3. Mac or Linux  X86_64

                    Please select 1, 2, or 3.
                    """);
                String platAns = scanner.nextLine().trim();
                if (platAns.equals("1")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("2")) {
                    platform = "ARM64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("3")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else {
```

```
                    System.out.println("Invalid input. Please select either 1 or
2.");
                }
            }

            jobARN = batchActions.registerJobDefinitionAsync(jobDefinitionName,
executionRoleARN, dockerImage, platform)
                .exceptionally(ex -> {
                    System.err.println("Register job definition failed: " +
ex.getMessage());
                    return null;
                })
                .join();
            if (jobARN != null) {
                logger.info("Job ARN: " + jobARN);
            }
        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while registering the job:
{}", rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Submit an AWS Batch job from a job definition.");
        waitForInputToContinue(scanner);
        String jobId;
        try {
            jobId = batchActions.submitJobAsync(jobDefinitionName, jobQueueName,
jobARN)
                .exceptionally(ex -> {
                    System.err.println("Submit job failed: " + ex.getMessage());
                    return null;
                })
                .join();

            logger.info("The job id is "+jobId);
            logger.info("Let's wait 2 minutes for the job to complete");
            countdown(2);

        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while submitting the job:
{}", rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
```

```
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info(DASHES);
        logger.info("6. Get a list of jobs applicable to the job queue.");

        waitForInputToContinue(scanner);
        try {
            List<JobSummary> jobs = batchActions.listJobsAsync(jobQueueName);
            jobs.forEach(job ->
                logger.info("Job ID: {}, Job Name: {}, Job Status: {}",
job.jobId(), job.jobName(), job.status()));

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job:
{}", rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Check the status of job "+jobId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
batchActions.describeJobAsync(jobId);
            String jobStatus = future.join();
            logger.info("Job Status: " + jobStatus);

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job:
{}", rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info("8. Delete Batch resources");
        logger.info(
            """
```

```
            When deleting an AWS Batch compute environment, it does not happen
instantaneously.
            There is typically a delay, similar to some other AWS resources.
            AWS Batch starts the deletion process.
            """);
        logger.info("Would you like to delete the AWS Batch resources such as the
compute environment? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS ECR resources.");
            logger.info("First, we will deregister the Job Definition.");
            waitForInputToContinue(scanner);
            try {
                batchActions.deregisterJobDefinitionAsync(jobARN)
                    .exceptionally(ex -> {
                        logger.info("Deregister job definition failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobARN + " was deregistered");
            } catch (RuntimeException rte) {
                logger.error("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            logger.info("Second, we will disable and then delete the Job
Queue.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableJobQueueAsync(jobQueueArn)
                    .exceptionally(ex -> {
                        logger.info("Disable job queue failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobQueueArn + " was disabled");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
```

```
            batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
            try {
                CompletableFuture<Void> future =
batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
                future.join();
                logger.info("Job queue is now disabled.");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            try {
                batchActions.deleteJobQueueAsync(jobQueueArn);
                logger.info(jobQueueArn +" was deleted");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            logger.info("Let's wait 2 minutes for the job queue to be deleted");
            countdown(2);
            waitForInputToContinue(scanner);

            logger.info("Third, we will delete the Compute Environment.");
            waitForInputToContinue(scanner);
            try {

batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                    .exceptionally(ex -> {
                        System.err.println("Disable compute environment failed: "
+ ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info("Compute environment disabled") ;
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
```

```
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName).thenAccept(state
-> {
               logger.info("Current State: " + state);
           }).join();

           logger.info("Lets wait 1 min for the compute environment to be
deleted");
           countdown(1);

           try {

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
               logger.info(computeEnvironmentName +" was deleted.");

           } catch (RuntimeException rte) {
               logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
               return;
           }
           waitForInputToContinue(scanner);
           CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
       }

       logger.info(DASHES);
       logger.info("This concludes the AWS Batch SDK scenario");
       logger.info(DASHES);
   }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           logger.info("");
           logger.info("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               logger.info("Continuing with the program...");
               logger.info("");
               break;
           } else {
               // Handle invalid input.
               logger.info("Invalid input. Please try again.");
           }
       }
```

```
    }

    public static void countdown(int minutes) throws InterruptedException {
        int seconds = 0;
        for (int i = minutes * 60 + seconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.print(String.format("\r%02d:%02d", displayMinutes,
displaySeconds));
            Thread.sleep(1000); // Wait for 1 second
        }
        logger.info("Countdown complete!");
    }

    private static void getSubnetSecurityGroup() {
        try (Ec2AsyncClient ec2Client = Ec2AsyncClient.create()) {
            CompletableFuture<Vpc> defaultVpcFuture =
ec2Client.describeVpcs(DescribeVpcsRequest.builder()
                            .filters(Filter.builder()
                                    .name("is-default")
                                    .values("true")
                                    .build())
                            .build())
                    .thenApply(response -> response.vpcs().stream()
                            .findFirst()
                            .orElseThrow(() -> new RuntimeException("Default VPC
not found")));

            CompletableFuture<String> defaultSubnetFuture = defaultVpcFuture
                    .thenCompose(vpc ->
ec2Client.describeSubnets(DescribeSubnetsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build(),
                                            Filter.builder()
                                                    .name("default-for-az")
                                                    .values("true")
                                                    .build())
                                    .build())
                            .thenApply(DescribeSubnetsResponse::subnets)
                            .thenApply(subnets -> subnets.stream()
                                    .findFirst()
                                    .map(Subnet::subnetId)
```

```
                                    .orElseThrow(() -> new RuntimeException("No
 default subnet found"))));

            CompletableFuture<String> defaultSecurityGroupFuture =
 defaultVpcFuture
                    .thenCompose(vpc ->
 ec2Client.describeSecurityGroups(DescribeSecurityGroupsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("group-name")
                                                    .values("default")
                                                    .build(),
                                            Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build())
                                    .build())

 .thenApply(DescribeSecurityGroupsResponse::securityGroups)
                            .thenApply(securityGroups -> securityGroups.stream()
                                    .findFirst()
                                    .map(SecurityGroup::groupId)
                                    .orElseThrow(() -> new RuntimeException("No
 default security group found"))));

            defaultSubnet = defaultSubnetFuture.join();
            defaultSecurityGroup = defaultSecurityGroupFuture.join();
        }
    }
}
```

Uma classe de invólucro para métodos do AWS Batch SDK.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.BatchClient;
import software.amazon.awssdk.services.batch.model.AssignPublicIp;
```

```
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.CEState;
import software.amazon.awssdk.services.batch.model.CEType;
import software.amazon.awssdk.services.batch.model.CRType;
import software.amazon.awssdk.services.batch.model.ComputeEnvironmentOrder;
import software.amazon.awssdk.services.batch.model.ComputeResource;
import software.amazon.awssdk.services.batch.model.ContainerProperties;
import
 software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentRequest;
import
 software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.CreateJobQueueRequest;
import
 software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentRequest;
import
 software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueResponse;
import
 software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionRequest;
import
 software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionResponse;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsRequest;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobsRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobsResponse;
import software.amazon.awssdk.services.batch.model.JQState;
import software.amazon.awssdk.services.batch.model.JobDefinitionType;
import software.amazon.awssdk.services.batch.model.JobDetail;
import software.amazon.awssdk.services.batch.model.JobQueueDetail;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionResponse;
import software.amazon.awssdk.services.batch.model.NetworkConfiguration;
import software.amazon.awssdk.services.batch.model.PlatformCapability;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.ResourceRequirement;
import software.amazon.awssdk.services.batch.model.ResourceType;
import software.amazon.awssdk.services.batch.model.RuntimePlatform;
```

```
import software.amazon.awssdk.services.batch.model.SubmitJobRequest;
import software.amazon.awssdk.services.batch.model.CreateJobQueueResponse;
import java.time.Duration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicBoolean;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.services.batch.model.SubmitJobResponse;
import
 software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentRequest;
import
 software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueResponse;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import software.amazon.awssdk.services.sts.StsAsyncClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;

public class BatchActions {
    private static BatchAsyncClient batchClient;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchActions.class);

    private static BatchAsyncClient getAsyncClient() {
        if (batchClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
```

```
                .build();

            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }

    /**
     * Asynchronously creates a new compute environment in AWS Batch.
     *
     * @param computeEnvironmentName the name of the compute environment to
create
     * @param batchIAMRole the IAM role to be used by the compute environment
     * @param subnet the subnet ID to be used for the compute environment
     * @param secGroup the security group ID to be used for the compute
environment
     * @return a {@link CompletableFuture} representing the asynchronous
operation, which will complete with the
     *         {@link CreateComputeEnvironmentResponse} when the compute
environment has been created
     * @throws BatchException if there is an error creating the compute
environment
     * @throws RuntimeException if there is an unexpected error during the
operation
     */
    public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
        String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
        CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
            .computeEnvironmentName(computeEnvironmentName)
            .type(CEType.MANAGED)
            .state(CEState.ENABLED)
            .computeResources(ComputeResource.builder()
                .type(CRType.FARGATE)
                .maxvCpus(256)
                .subnets(Collections.singletonList(subnet))
                .securityGroupIds(Collections.singletonList(secGroup))
                .build())
```

```
            .serviceRole(batchIAMRole)
            .build();

        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
            }
        });

        return response;
    }

    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return
getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                }
            });
    }

    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute
environment, or "ERROR" if an exception occurs
```

```
    */
   public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot
be null or empty");
       }

       DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
           .computeEnvironments(computeEnvironmentName)
           .build();

       CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
           }
       });

       return response.thenApply(resp -> resp.computeEnvironments().stream()
           .map(env -> env.statusAsString())
           .findFirst()
           .orElse("UNKNOWN"));
   }

   /**
    * Creates a job queue asynchronously.
    *
    * @param jobQueueName the name of the job queue to create
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName,
String computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
```

```
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot
be null or empty");
        }

        CreateJobQueueRequest request = CreateJobQueueRequest.builder()
            .jobQueueName(jobQueueName)
            .priority(1)
            .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
                .computeEnvironment(computeEnvironmentName)
                .order(1)
                .build())
            .build();

        CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }

    /**
     * Asynchronously lists the jobs in the specified job queue with the given
job status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or
empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();
```

```
        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response ->
{
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
    }

    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
     * When using Fargate as the compute environment, it is crucial to set the
     * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
     * ensure proper networking configuration for the Fargate tasks. This
     * allows the tasks to communicate with external services, access the
     * internet, or communicate within a VPC.
     *
     * @param jobDefinitionName the name of the job definition to be registered
     * @param executionRoleARN the ARN (Amazon Resource Name) of the execution
role
     *                         that provides permissions for the containers in
the job
     * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
     * @return a CompletableFuture that completes with the ARN of the registered
     *         job definition upon successful execution, or completes
exceptionally with
     *         an error if the registration fails
     */
    public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
        NetworkConfiguration networkConfiguration =
NetworkConfiguration.builder()
                .assignPublicIp(AssignPublicIp.ENABLED)
                .build();

        ContainerProperties containerProperties = ContainerProperties.builder()
                .image(image)
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
```

```
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
 RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }

    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
```

```
    * @return a CompletableFuture that completes when the job definition has
been deregistered
    * or an exception has occurred
    */
   public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
       DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
           .jobDefinition(jobDefinition)
           .build();

       CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }

   /**
    * Disables the specified job queue asynchronously.
    *
    * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
    * @return a {@link CompletableFuture} that completes when the job queue
update operation is complete,
    *         or completes exceptionally if an error occurs during the operation
    */
   public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
       UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .state(JQState.DISABLED)
           .build();

       CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
       return responseFuture.whenComplete((updateResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
```

```
            }
        }).thenApply(updateResponse -> null);
    }

    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to
delete.
     * @return A CompletableFuture that represents the asynchronous deletion of
the job queue.
     *         The future completes when the job queue has been successfully
deleted or if an error occurs.
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally
with the thrown exception.
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .build();

        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }

    /**
     * Asynchronously describes the job queue associated with the specified
compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to find
the associated job queue for
     * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
     * @throws RuntimeException if the job queue description fails
     */
```

```
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
               String jobQueueARN;
               for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                   for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                       String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                       String name =
getComputeEnvironmentName(computeEnvironment);
                       if (name.equals(computeEnvironmentName)) {
                           jobQueueARN = jobQueueDetail.jobQueueArn();
                           logger.info("Job queue ARN associated with the
compute environment: " + jobQueueARN);
                       }
                   }
               }
           } else {
               throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(describeJobQueuesResponse -> {
           String jobQueueARN = "";
           for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
               for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                   String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                   String name = getComputeEnvironmentName(computeEnvironment);
                   if (name.equals(computeEnvironmentName)) {
                       jobQueueARN = jobQueueDetail.jobQueueArn();
                   }
               }
           }
```

```
            return jobQueueARN;
        });
    }

    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to
disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
    public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
        UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .state(CEState.DISABLED)
            .build();

        CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disable compute
environment: " + ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Submits a job asynchronously to the AWS Batch service.
     *
     * @param jobDefinitionName the name of the job definition to use
     * @param jobQueueName the name of the job queue to submit the job to
     * @param jobARN the Amazon Resource Name (ARN) of the job definition
     * @return a CompletableFuture that, when completed, contains the job ID of
the submitted job
     */
    public CompletableFuture<String> submitJobAsync(String jobDefinitionName,
String jobQueueName, String jobARN) {
        SubmitJobRequest jobRequest = SubmitJobRequest.builder()
```

```
            .jobDefinition(jobARN)
            .jobName(jobDefinitionName)
            .jobQueue(jobQueueName)
            .build();

        CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }

    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }

    /**
     * Disables the specific job queue using the asynchronous Java client.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to wait
for
```

```
    * @return a {@link CompletableFuture} that completes when the job queue is
disabled
    */
   public CompletableFuture<Void> waitForJobQueueToBeDisabledAsync(String
jobQueueArn) {
       AtomicBoolean isDisabled = new AtomicBoolean(false);
       return CompletableFuture.runAsync(() -> {
           while (!isDisabled.get()) {
               DescribeJobQueuesRequest describeRequest =
DescribeJobQueuesRequest.builder()
                   .jobQueues(jobQueueArn)
                   .build();

               CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeRequest);
               responseFuture.whenComplete((describeResponse, ex) -> {
                   if (describeResponse != null) {
                       for (JobQueueDetail jobQueue :
describeResponse.jobQueues()) {
                           if (jobQueue.jobQueueArn().equals(jobQueueArn) &&
jobQueue.state() == JQState.DISABLED) {
                               isDisabled.set(true);
                               break;
                           }
                       }
                   } else {
                       throw new RuntimeException("Error describing job queues",
ex);
                   }
               }).join();

               if (!isDisabled.get()) {
                   try {
                       logger.info("Waiting for job queue to be disabled...");
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       Thread.currentThread().interrupt();
                       throw new RuntimeException("Thread interrupted while
waiting for job queue to be disabled", e);
                   }
               }
           }
       }).whenComplete((result, throwable) -> {
           if (throwable != null) {
```

```
                throw new RuntimeException("Error while waiting for job queue to
be disabled", throwable);
            }
        });
    }

    public CompletableFuture<String> getJobQueueARN(String jobQueueName) {
        // Describe the job queue asynchronously
        CompletableFuture<DescribeJobQueuesResponse> describeJobQueuesFuture =
batchClient.describeJobQueues(
            DescribeJobQueuesRequest.builder()
                .jobQueues(jobQueueName)
                .build()
        );

        // Handle the asynchronous response and return the Job Queue ARN in the
CompletableFuture<String>
        CompletableFuture<String> jobQueueArnFuture = new CompletableFuture<>();
        describeJobQueuesFuture.whenComplete((response, error) -> {
            if (error != null) {
                if (error instanceof BatchException) {
                    logger.info("Batch error: " + ((BatchException)
error).awsErrorDetails().errorMessage());
                } else {
                    logger.info("Error describing job queue: " +
error.getMessage());
                }
                jobQueueArnFuture.completeExceptionally(new
RuntimeException("Failed to retrieve Job Queue ARN", error));
            } else {
                if (response.jobQueues().isEmpty()) {
                    jobQueueArnFuture.completeExceptionally(new
RuntimeException("Job queue not found: " + jobQueueName));
                } else {
                    // Assuming only one job queue is returned for the given name
                    String jobQueueArn =
response.jobQueues().get(0).jobQueueArn();
                    jobQueueArnFuture.complete(jobQueueArn);
                }
            }
        });

        return jobQueueArnFuture;
    }
```

```
    private static String getComputeEnvironmentName(String computeEnvironment) {
        String[] parts = computeEnvironment.split("/");
        if (parts.length == 2) {
            return parts[1];
        }
        return null;
    }

    public CompletableFuture<String> getAccountId() {
        StsAsyncClient stsAsyncClient = StsAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        return stsAsyncClient.getCallerIdentity()
            .thenApply(GetCallerIdentityResponse::account);
    }


}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DeregisterJobDefinition
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - ListJobsPaginator
  - RegisterJobDefinition
  - SubmitJob
  - UpdateComputeEnvironment
  - UpdateJobQueue

Ações para AWS Batch usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Batch ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a AWS Batch API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para AWS Batch usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Batch.

Exemplos

- Use CreateComputeEnvironmentcom um AWS SDK ou CLI
- Use CreateJobQueuecom um AWS SDK ou CLI
- Use DeleteComputeEnvironmentcom um AWS SDK ou CLI
- Use DeleteJobQueuecom um AWS SDK ou CLI
- Use DeregisterJobDefinitioncom um AWS SDK ou CLI
- Use DescribeComputeEnvironmentscom um AWS SDK ou CLI
- Use DescribeJobQueuescom um AWS SDK ou CLI
- Use DescribeJobscom um AWS SDK ou CLI
- Use ListJobsPaginatorcom um AWS SDK
- Use RegisterJobDefinitioncom um AWS SDK ou CLI
- Use SubmitJobcom um AWS SDK ou CLI
- Use UpdateComputeEnvironmentcom um AWS SDK ou CLI
- Use UpdateJobQueuecom um AWS SDK ou CLI

Use **`CreateComputeEnvironment`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateComputeEnvironment`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos

- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para criar um ambiente computacional gerenciado com instâncias On-Demand

Este exemplo cria um ambiente computacional gerenciado com tipos específicos de instância C4 que são executados sob demanda. O ambiente computacional é chamado C4OnDemand.

Comando:

```
aws batch create-compute-environment --cli-input-json file://<path_to_json_file>/
C4OnDemand.json
```

Formato de arquivo JSON:

```
{
  "computeEnvironmentName": "C4OnDemand",
  "type": "MANAGED",
  "state": "ENABLED",
  "computeResources": {
    "type": "EC2",
    "minvCpus": 0,
    "maxvCpus": 128,
    "desiredvCpus": 48,
    "instanceTypes": [
      "c4.large",
      "c4.xlarge",
      "c4.2xlarge",
      "c4.4xlarge",
      "c4.8xlarge"
    ],
    "subnets": [
      "subnet-220c0e0a",
      "subnet-1a95556d",
      "subnet-978f6dce"
    ],
    "securityGroupIds": [
      "sg-cf5093b2"
    ],
    "ec2KeyPair": "id_rsa",
```

```
    "instanceRole": "ecsInstanceRole",
    "tags": {
      "Name": "Batch Instance - C4OnDemand"
    }
  },
  "serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole"
}
```

Saída:

```
{
    "computeEnvironmentName": "C4OnDemand",
    "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/C4OnDemand"
}
```

Para criar um ambiente de computação gerenciado com instâncias spot

Este exemplo cria um ambiente de computação gerenciado com o tipo de instância M4 que é lançado quando o preço da oferta spot é igual ou inferior a 20% do On-Demand preço do tipo de instância. O ambiente computacional é chamado de M4Spot.

Comando:

```
aws batch create-compute-environment --cli-input-json file://<path_to_json_file>/
M4Spot.json
```

Formato de arquivo JSON:

```
{
  "computeEnvironmentName": "M4Spot",
  "type": "MANAGED",
  "state": "ENABLED",
  "computeResources": {
    "type": "SPOT",
    "spotIamFleetRole": "arn:aws:iam::012345678910:role/aws-ec2-spot-fleet-role",
    "minvCpus": 0,
    "maxvCpus": 128,
    "desiredvCpus": 4,
    "instanceTypes": [
      "m4"
    ],
```

```
    "bidPercentage": 20,
    "subnets": [
      "subnet-220c0e0a",
      "subnet-1a95556d",
      "subnet-978f6dce"
    ],
    "securityGroupIds": [
      "sg-cf5093b2"
    ],
    "ec2KeyPair": "id_rsa",
    "instanceRole": "ecsInstanceRole",
    "tags": {
      "Name": "Batch Instance - M4Spot"
    }
  },
  "serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole"
}
```

Saída:

```
{
    "computeEnvironmentName": "M4Spot",
    "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/M4Spot"
}
```

- Para obter detalhes da API, consulte CreateComputeEnvironmentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously creates a new compute environment in AWS Batch.
```

```
    *
    * @param computeEnvironmentName the name of the compute environment to
create
    * @param batchIAMRole the IAM role to be used by the compute environment
    * @param subnet the subnet ID to be used for the compute environment
    * @param secGroup the security group ID to be used for the compute
environment
    * @return a {@link CompletableFuture} representing the asynchronous
operation, which will complete with the
    *         {@link CreateComputeEnvironmentResponse} when the compute
environment has been created
    * @throws BatchException if there is an error creating the compute
environment
    * @throws RuntimeException if there is an unexpected error during the
operation
    */
   public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
       String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
       CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
           .computeEnvironmentName(computeEnvironmentName)
           .type(CEType.MANAGED)
           .state(CEState.ENABLED)
           .computeResources(ComputeResource.builder()
               .type(CRType.FARGATE)
               .maxvCpus(256)
               .subnets(Collections.singletonList(subnet))
               .securityGroupIds(Collections.singletonList(secGroup))
               .build())
           .serviceRole(batchIAMRole)
           .build();

       CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
              throw new RuntimeException(errorMessage, ex);
           }
       });
```

```
        return response;
    }
```

- Para obter detalhes da API, consulte CreateComputeEnvironmenta Referência AWS SDK for Java 2.x da API.

Use **`CreateJobQueue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateJobQueue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para criar uma fila de trabalhos de baixa prioridade com um único ambiente computacional

Este exemplo cria uma fila de trabalhos chamada LowPriority que usa o ambiente de computação M4Spot.

Comando:

```
aws batch create-job-queue --cli-input-json file://<path_to_json_file>/
LowPriority.json
```

Formato de arquivo JSON:

```
{
  "jobQueueName": "LowPriority",
  "state": "ENABLED",
  "priority": 10,
  "computeEnvironmentOrder": [
    {
      "order": 1,
      "computeEnvironment": "M4Spot"
```

```
    }
  ]
}
```

Saída:

```
{
    "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/LowPriority",
    "jobQueueName": "LowPriority"
}
```

Para criar uma fila de trabalhos de alta prioridade com dois ambientes computacionais

Este exemplo cria uma fila de trabalhos chamada HighPriority que usa o ambiente de C4OnDemand computação com uma ordem de 1 e o ambiente de computação M4Spot com uma ordem de 2. O programador tentará primeiro colocar trabalhos no ambiente C4OnDemand computacional.

Comando:

```
aws batch create-job-queue --cli-input-json file://<path_to_json_file>/
HighPriority.json
```

Formato de arquivo JSON:

```
{
  "jobQueueName": "HighPriority",
  "state": "ENABLED",
  "priority": 1,
  "computeEnvironmentOrder": [
    {
      "order": 1,
      "computeEnvironment": "C4OnDemand"
    },
    {
      "order": 2,
      "computeEnvironment": "M4Spot"
    }
  ]
}
```

Saída:

```
{
    "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/HighPriority",
    "jobQueueName": "HighPriority"
}
```

- Para obter detalhes da API, consulte CreateJobQueueem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a job queue asynchronously.
    *
    * @param jobQueueName the name of the job queue to create
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName,
String computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot
be null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
```

```
            .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
                .computeEnvironment(computeEnvironmentName)
                .order(1)
                .build())
            .build();

        CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }
```

- Para obter detalhes da API, consulte CreateJobQueuea Referência AWS SDK for Java 2.x da API.

Use **`DeleteComputeEnvironment`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteComputeEnvironment`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para excluir um ambiente de computação

Este exemplo exclui o ambiente P2OnDemand computacional.

Comando:

```
aws batch delete-compute-environment --compute-environment P2OnDemand
```

- Para obter detalhes da API, consulte DeleteComputeEnvironmentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
       DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .build();

       return
getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
           .whenComplete((response, ex) -> {
               if (ex != null) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof BatchException) {
                       throw new RuntimeException(cause);
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               }
           });
   }
```

- Para obter detalhes da API, consulte DeleteComputeEnvironmenta Referência AWS SDK for Java 2.x da API.

Use **`DeleteJobQueue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteJobQueue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para excluir uma fila de trabalhos

Este exemplo exclui a fila de trabalho GPGPU.

Comando:

```
aws batch delete-job-queue --job-queue GPGPU
```

- Para obter detalhes da API, consulte DeleteJobQueueem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to
delete.
```

```
    * @return A CompletableFuture that represents the asynchronous deletion of
the job queue.
    *         The future completes when the job queue has been successfully
deleted or if an error occurs.
    *         If successful, the future will be completed with a {@code Void}
value.
    *         If an error occurs, the future will be completed exceptionally
with the thrown exception.
    */
   public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
       DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .build();

       CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
       return responseFuture.whenComplete((deleteResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(deleteResponse -> null);
   }
```

- Para obter detalhes da API, consulte DeleteJobQueuea Referência AWS SDK for Java 2.x da API.

Use **`DeregisterJobDefinition`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeregisterJobDefinition`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como cancelar o registro de uma definição de trabalho

Este exemplo cancela o registro de uma definição de trabalho chamada sleep10.

Comando:

```
aws batch deregister-job-definition --job-definition sleep10
```

- Para obter detalhes da API, consulte DeregisterJobDefinitionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deregisters a job definition asynchronously.
    *
    * @param jobDefinition the name of the job definition to be deregistered
    * @return a CompletableFuture that completes when the job definition has
been deregistered
    * or an exception has occurred
    */
   public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
       DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
           .jobDefinition(jobDefinition)
           .build();

       CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
```

```
        });

        return responseFuture;
    }
```

- Para obter detalhes da API, consulte DeregisterJobDefinitiona Referência AWS SDK for Java 2.x da API.

Use **`DescribeComputeEnvironments`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeComputeEnvironments`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Como descrever um ambiente de computação

Este exemplo descreve o ambiente P2OnDemand computacional.

Comando:

```
aws batch describe-compute-environments --compute-environments P2OnDemand
```

Saída:

```
{
    "computeEnvironments": [
        {
            "status": "VALID",
            "serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole",
            "computeEnvironmentArn": "arn:aws:batch:us-
east-1:012345678910:compute-environment/P2OnDemand",
```

```
            "computeResources": {
                "subnets": [
                    "subnet-220c0e0a",
                    "subnet-1a95556d",
                    "subnet-978f6dce"
                ],
                "tags": {
                    "Name": "Batch Instance - P2OnDemand"
                },
                "desiredvCpus": 48,
                "minvCpus": 0,
                "instanceTypes": [
                    "p2"
                ],
                "securityGroupIds": [
                    "sg-cf5093b2"
                ],
                "instanceRole": "ecsInstanceRole",
                "maxvCpus": 128,
                "type": "EC2",
                "ec2KeyPair": "id_rsa"
            },
            "statusReason": "ComputeEnvironment Healthy",
            "ecsClusterArn": "arn:aws:ecs:us-east-1:012345678910:cluster/
P2OnDemand_Batch_2c06f29d-d1fe-3a49-879d-42394c86effc",
            "state": "ENABLED",
            "computeEnvironmentName": "P2OnDemand",
            "type": "MANAGED"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeComputeEnvironmentsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Checks the status of the specified compute environment.
    *
    * @param computeEnvironmentName the name of the compute environment to check
    * @return a CompletableFuture containing the status of the compute
environment, or "ERROR" if an exception occurs
    */
   public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot
be null or empty");
       }

       DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
           .computeEnvironments(computeEnvironmentName)
           .build();

       CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
           }
       });

       return response.thenApply(resp -> resp.computeEnvironments().stream()
           .map(env -> env.statusAsString())
           .findFirst()
```

```
            .orElse("UNKNOWN"));
    }
```

- Para obter detalhes da API, consulte DescribeComputeEnvironmentsa Referência AWS SDK for Java 2.x da API.

Use **`DescribeJobQueues`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeJobQueues`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para descrever uma fila de trabalhos

Este exemplo descreve a fila HighPriority de trabalhos.

Comando:

```
aws batch describe-job-queues --job-queues HighPriority
```

Saída:

```
{
    "jobQueues": [
        {
            "status": "VALID",
            "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/
HighPriority",
            "computeEnvironmentOrder": [
                {
```

```
                    "computeEnvironment": "arn:aws:batch:us-
east-1:012345678910:compute-environment/C4OnDemand",
                    "order": 1
                }
            ],
            "statusReason": "JobQueue Healthy",
            "priority": 1,
            "state": "ENABLED",
            "jobQueueName": "HighPriority"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeJobQueuesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the job queue associated with the specified
compute environment.
    *
    * @param computeEnvironmentName the name of the compute environment to find
the associated job queue for
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();
```

```
        CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
        return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
            if (describeJobQueuesResponse != null) {
                String jobQueueARN;
                for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                    for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                        String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                        String name =
getComputeEnvironmentName(computeEnvironment);
                        if (name.equals(computeEnvironmentName)) {
                            jobQueueARN = jobQueueDetail.jobQueueArn();
                            logger.info("Job queue ARN associated with the
compute environment: " + jobQueueARN);
                        }
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }
```

- Para obter detalhes da API, consulte DescribeJobQueuesa Referência AWS SDK for Java 2.x da API.

Use **`DescribeJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeJobs`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para descrever um trabalho

O exemplo `describe-jobs` a seguir descreve um trabalho com o ID de trabalho especificado.

```
aws batch describe-jobs \
    --jobs bcf0b186-a532-4122-842e-2ccab8d54efb
```

Saída:

```
{
    "jobs": [
        {
            "status": "SUBMITTED",
            "container": {
                "mountPoints": [],
                "image": "busybox",
                "environment": [],
                "vcpus": 1,
                "command": [
                    "sleep",
                    "60"
                ],
                "volumes": [],
```

```
                "memory": 128,
                "ulimits": []
            },
            "parameters": {},
            "jobDefinition": "arn:aws:batch:us-east-1:012345678910:job-
definition/sleep60:1",
            "jobQueue": "arn:aws:batch:us-east-1:012345678910:job-queue/
HighPriority",
            "jobId": "bcf0b186-a532-4122-842e-2ccab8d54efb",
            "dependsOn": [],
            "jobName": "example",
            "createdAt": 1480483387803
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeJobsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
 getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
```

```
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }
```

- Para obter detalhes da API, consulte DescribeJobsa Referência AWS SDK for Java 2.x da API.

Use **`ListJobsPaginator`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListJobsPaginator`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists the jobs in the specified job queue with the given
job status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or
empty");
```

```
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response ->
{
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
    }
```

- Para obter detalhes da API, consulte ListJobsPaginatora Referência AWS SDK for Java 2.x da API.

Use **`RegisterJobDefinition`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RegisterJobDefinition`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para registrar uma definição de trabalho

Este exemplo registra uma definição de trabalho para um trabalho de contêiner simples.

Comando:

```
aws batch register-job-definition --job-definition-name sleep30 --type container
 --container-properties '{ "image": "busybox", "vcpus": 1, "memory": 128,
 "command": [ "sleep", "30"]}'
```

Saída:

```
{
    "jobDefinitionArn": "arn:aws:batch:us-east-1:012345678910:job-definition/
sleep30:1",
    "jobDefinitionName": "sleep30",
    "revision": 1
}
```

- Para obter detalhes da API, consulte RegisterJobDefinitionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
     * When using Fargate as the compute environment, it is crucial to set the
     * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
     * ensure proper networking configuration for the Fargate tasks. This
     * allows the tasks to communicate with external services, access the
     * internet, or communicate within a VPC.
     *
     * @param jobDefinitionName the name of the job definition to be registered
     * @param executionRoleARN the ARN (Amazon Resource Name) of the execution
 role
     *                         that provides permissions for the containers in
 the job
```

```
    * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
    * @return a CompletableFuture that completes with the ARN of the registered
    *         job definition upon successful execution, or completes
exceptionally with
    *         an error if the registration fails
    */
   public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
       NetworkConfiguration networkConfiguration =
NetworkConfiguration.builder()
               .assignPublicIp(AssignPublicIp.ENABLED)
               .build();

       ContainerProperties containerProperties = ContainerProperties.builder()
               .image(image)
               .executionRoleArn(executionRoleARN)
               .resourceRequirements(
                       Arrays.asList(
                               ResourceRequirement.builder()
                                       .type(ResourceType.VCPU)
                                       .value("1")
                                       .build(),
                               ResourceRequirement.builder()
                                       .type(ResourceType.MEMORY)
                                       .value("2048")
                                       .build()
                       )
               )
               .networkConfiguration(networkConfiguration)
              .runtimePlatform(b -> b
                       .cpuArchitecture(cpuArch)
                       .operatingSystemFamily("LINUX"))
               .build();

       RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
               .jobDefinitionName(jobDefinitionName)
               .type(JobDefinitionType.CONTAINER)
               .containerProperties(containerProperties)
               .platformCapabilities(PlatformCapability.FARGATE)
               .build();

       CompletableFuture<String> future = new CompletableFuture<>();
```

```
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }
```

- Para obter detalhes da API, consulte RegisterJobDefinitiona Referência AWS SDK for Java 2.x da API.

Use **`SubmitJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SubmitJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Como enviar um trabalho

Este exemplo envia um trabalho de contêiner simples chamado example para a fila de HighPriority trabalhos.

Comando:

```
aws batch submit-job --job-name example --job-queue HighPriority  --job-
definition sleep60
```

Saída:

```
{
    "jobName": "example",
    "jobId": "876da822-4198-45f2-a252-6cea32512ea8"
}
```

- Para obter detalhes da API, consulte SubmitJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Submits a job asynchronously to the AWS Batch service.
    *
    * @param jobDefinitionName the name of the job definition to use
    * @param jobQueueName the name of the job queue to submit the job to
    * @param jobARN the Amazon Resource Name (ARN) of the job definition
    * @return a CompletableFuture that, when completed, contains the job ID of
the submitted job
    */
   public CompletableFuture<String> submitJobAsync(String jobDefinitionName,
String jobQueueName, String jobARN) {
       SubmitJobRequest jobRequest = SubmitJobRequest.builder()
           .jobDefinition(jobARN)
           .jobName(jobDefinitionName)
           .jobQueue(jobQueueName)
           .build();

       CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
```

```
                throw new RuntimeException("Unexpected error occurred: " +
 ex.getMessage(), ex);
            }
        });

        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }
```

- Para obter detalhes da API, consulte SubmitJoba Referência AWS SDK for Java 2.x da API.

Use **`UpdateComputeEnvironment`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateComputeEnvironment`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para atualizar um ambiente de computação

Este exemplo desativa o ambiente P2OnDemand computacional para que ele possa ser excluído.

Comando:

```
aws batch update-compute-environment --compute-environment P2OnDemand --
state DISABLED
```

Saída:

```
{
    "computeEnvironmentName": "P2OnDemand",
```

```
    "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/P2OnDemand"
}
```

- Para obter detalhes da API, consulte UpdateComputeEnvironmentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Disables the specified compute environment asynchronously.
    *
    * @param computeEnvironmentName the name of the compute environment to
disable
    * @return a CompletableFuture that completes when the compute environment is
disabled
    */
   public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
       UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .state(CEState.DISABLED)
           .build();

       CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to disable compute
environment: " + ex.getMessage(), ex);
           }
       });
```

```
        return responseFuture;
    }
```

- Para obter detalhes da API, consulte UpdateComputeEnvironmenta Referência AWS SDK for Java 2.x da API.

Use **`UpdateJobQueue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateJobQueue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

CLI

AWS CLI

Para atualizar uma fila de trabalhos

Este exemplo desabilita uma fila de trabalhos para que ela possa ser excluída.

Comando:

```
aws batch update-job-queue --job-queue GPGPU --state DISABLED
```

Saída:

```
{
    "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/GPGPU",
    "jobQueueName": "GPGPU"
}
```

- Para obter detalhes da API, consulte UpdateJobQueueem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Disables the specified job queue asynchronously.
    *
    * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
    * @return a {@link CompletableFuture} that completes when the job queue
update operation is complete,
    *         or completes exceptionally if an error occurs during the operation
    */
   public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
       UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .state(JQState.DISABLED)
           .build();

       CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
       return responseFuture.whenComplete((updateResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(updateResponse -> null);
   }
```

- Para obter detalhes da API, consulte UpdateJobQueuea Referência AWS SDK for Java 2.x da API.

Cenários para AWS Batch usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS Batch com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS Batch ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um perfil de execução do IAM.
- Criar um ambiente de computação.
- Criar uma fila de trabalhos.
- Criar uma definição de trabalho.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Batch Fargate Getting Started Script - Security Hardened Version
# This script demonstrates creating AWS Batch resources with Fargate
 orchestration
```

```
#

set -euo pipefail  # Exit on any error, undefined variables, and pipe failures

# Configuration
SCRIPT_NAME="batch-fargate-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
RANDOM_SUFFIX=$(openssl rand -hex 6)
COMPUTE_ENV_NAME="batch-fargate-compute-${RANDOM_SUFFIX}"
JOB_QUEUE_NAME="batch-fargate-queue-${RANDOM_SUFFIX}"
JOB_DEF_NAME="batch-fargate-jobdef-${RANDOM_SUFFIX}"
JOB_NAME="batch-hello-world-${RANDOM_SUFFIX}"
ROLE_NAME="BatchEcsTaskExecutionRole-${RANDOM_SUFFIX}"
TRUST_POLICY_FILE="batch-trust-policy-${RANDOM_SUFFIX}.json"

# Security: Set restrictive umask
umask 0077

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Logging function with sanitization
log() {
    local message="${1//[$'\t\r\n']/}"
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] ${message}" | tee -a "${LOG_FILE}"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources created so far..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error ${LINENO}' ERR

# Validate AWS credentials
validate_aws_credentials() {
    if ! aws sts get-caller-identity &>/dev/null; then
        log "ERROR: AWS credentials are not configured or invalid"
        exit 1
    fi
```

```
}

# Function to wait for resource to be ready
wait_for_compute_env() {
    local env_name="${1}"
    local max_attempts=60
    local attempt=0

    log "Waiting for compute environment ${env_name} to be VALID..."

    while [ ${attempt} -lt ${max_attempts} ]; do
        local status
        status=$(aws batch describe-compute-environments \
            --compute-environments "${env_name}" \
            --query 'computeEnvironments[0].status' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${status}" = "VALID" ]; then
            log "Compute environment ${env_name} is ready"
            return 0
        elif [ "${status}" = "INVALID" ] || [ "${status}" = "NOT_FOUND" ]; then
            log "ERROR: Compute environment ${env_name} failed to create
 properly"
            return 1
        fi

        log "Compute environment status: ${status}. Waiting 10 seconds..."
        sleep 10
        ((attempt++))
    done

    log "ERROR: Timeout waiting for compute environment ${env_name}"
    return 1
}

# Function to wait for job queue to be ready
wait_for_job_queue() {
    local queue_name="${1}"
    local max_attempts=60
    local attempt=0

    log "Waiting for job queue ${queue_name} to be VALID..."

    while [ ${attempt} -lt ${max_attempts} ]; do
```

```
        local state
        state=$(aws batch describe-job-queues \
            --job-queues "${queue_name}" \
            --query 'jobQueues[0].state' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${state}" = "ENABLED" ]; then
            log "Job queue ${queue_name} is ready"
            return 0
        elif [ "${state}" = "DISABLED" ] || [ "${state}" = "NOT_FOUND" ]; then
            log "ERROR: Job queue ${queue_name} failed to create properly"
            return 1
        fi

        log "Job queue state: ${state}. Waiting 10 seconds..."
        sleep 10
        ((attempt++))
    done

    log "ERROR: Timeout waiting for job queue ${queue_name}"
    return 1
}

# Function to wait for job completion
wait_for_job() {
    local job_id="${1}"
    local max_attempts=120
    local attempt=0

    log "Waiting for job ${job_id} to complete..."

    while [ ${attempt} -lt ${max_attempts} ]; do
        local status
        status=$(aws batch describe-jobs \
            --jobs "${job_id}" \
            --query 'jobs[0].status' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${status}" = "SUCCEEDED" ]; then
            log "Job ${job_id} completed successfully"
            return 0
        elif [ "${status}" = "FAILED" ]; then
            log "ERROR: Job ${job_id} failed"
            return 1
```

```
        fi

        log "Job status: ${status}. Waiting 30 seconds..."
        sleep 30
        ((attempt++))
    done

    log "ERROR: Timeout waiting for job ${job_id}"
    return 1
}

# Function to wait for resource state before deletion
wait_for_resource_state() {
    local resource_type="${1}"
    local resource_name="${2}"
    local expected_state="${3}"
    local max_attempts=30
    local attempt=0

    log "Waiting for ${resource_type} ${resource_name} to reach state:
 ${expected_state}"

    while [ ${attempt} -lt ${max_attempts} ]; do
        local current_state=""

        case "${resource_type}" in
            "JOB_QUEUE")
                current_state=$(aws batch describe-job-queues \
                    --job-queues "${resource_name}" \
                    --query 'jobQueues[0].state' \
                    --output text 2>/dev/null || echo "NOT_FOUND")
                ;;
            "COMPUTE_ENV")
                current_state=$(aws batch describe-compute-environments \
                    --compute-environments "${resource_name}" \
                    --query 'computeEnvironments[0].status' \
                    --output text 2>/dev/null || echo "NOT_FOUND")
                ;;
            *)
                log "WARNING: Unknown resource type: ${resource_type}"
                return 1
                ;;
        esac
```

```
        if [ "${current_state}" = "${expected_state}" ]; then
            log "${resource_type} ${resource_name} is now in state:
 ${expected_state}"
            return 0
        fi

        log "${resource_type} ${resource_name} state: ${current_state} (waiting
 for ${expected_state})"
        sleep 10
        ((attempt++))
    done

    log "WARNING: ${resource_type} ${resource_name} did not reach expected state
 after ${max_attempts} attempts"
    return 1
}

# Cleanup function
cleanup_resources() {
    log "Starting cleanup of created resources..."

    # Clean up in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local resource="${CREATED_RESOURCES[i]}"
        local resource_type
        local resource_name

        resource_type=$(echo "${resource}" | cut -d: -f1)
        resource_name=$(echo "${resource}" | cut -d: -f2-)

        log "Cleaning up ${resource_type}: ${resource_name}"

        case "${resource_type}" in
            "JOB_QUEUE")
                aws batch update-job-queue --job-queue "${resource_name}" --state
 DISABLED 2>/dev/null || true
                wait_for_resource_state "JOB_QUEUE" "${resource_name}" "DISABLED"
 || true
                aws batch delete-job-queue --job-queue "${resource_name}" 2>/dev/
null || true
                ;;
            "COMPUTE_ENV")
                aws batch update-compute-environment --compute-environment
 "${resource_name}" --state DISABLED 2>/dev/null || true
```

```
                wait_for_resource_state "COMPUTE_ENV" "${resource_name}"
 "DISABLED" || true
                aws batch delete-compute-environment --compute-environment
 "${resource_name}" 2>/dev/null || true
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name
 "${resource_name}" --policy-arn "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" 2>/dev/null || true
                aws iam delete-role --role-name "${resource_name}" 2>/dev/null ||
 true
                ;;
            "FILE")
                rm -f "${resource_name}" 2>/dev/null || true
                ;;
            *)
                log "WARNING: Unknown resource type for cleanup:
 ${resource_type}"
                ;;
        esac
    done

    log "Cleanup completed"
}

# Validate input parameters
validate_inputs() {
    if [ -z "${ACCOUNT_ID:-}" ]; then
        log "ERROR: ACCOUNT_ID is not set"
        return 1
    fi

    if [ -z "${DEFAULT_VPC:-}" ]; then
        log "ERROR: DEFAULT_VPC is not set"
        return 1
    fi

    if [ -z "${SUBNETS:-}" ]; then
        log "ERROR: SUBNETS is not set"
        return 1
    fi
}

# Validate container image format
```

```
validate_container_image() {
    local image="${1}"

    # Check if image contains any shell metacharacters that could be dangerous
    if [[ "${image}" =~ [';$`|&<>()[]{}\\'] ]]; then
        log "ERROR: Container image contains potentially dangerous characters:
 ${image}"
        return 1
    fi

    # Basic ECR/Docker image format validation
    if ! [[ "${image}" =~ ^[a-zA-Z0-9._/:-]+$ ]]; then
        log "ERROR: Container image format is invalid: ${image}"
        return 1
    fi

    return 0
}

# Main script execution
main() {
    log "Starting AWS Batch Fargate tutorial script - Security Hardened Version"
    log "Log file: ${LOG_FILE}"

    # Validate AWS credentials
    validate_aws_credentials

    # Get AWS account ID
    log "Getting AWS account ID..."
    local ACCOUNT_ID
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    if [ -z "${ACCOUNT_ID}" ] || [ "${ACCOUNT_ID}" = "None" ]; then
        log "ERROR: Could not retrieve AWS account ID"
        exit 1
    fi
    log "Account ID: ${ACCOUNT_ID}"

    # Get default VPC and subnets
    log "Getting default VPC and subnets..."
    local DEFAULT_VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs \
        --filters "Name=is-default,Values=true" \
        --query 'Vpcs[0].VpcId' \
        --output text)
```

```
    if [ "${DEFAULT_VPC}" = "None" ] || [ "${DEFAULT_VPC}" = "null" ] || [ -z
 "${DEFAULT_VPC}" ]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi

    log "Default VPC: ${DEFAULT_VPC}"

    # Get subnets in the default VPC
    local SUBNETS
    SUBNETS=$(aws ec2 describe-subnets \
        --filters "Name=vpc-id,Values=${DEFAULT_VPC}" \
        --query 'Subnets[*].SubnetId' \
        --output text)

    if [ -z "${SUBNETS}" ]; then
        log "ERROR: No subnets found in default VPC"
        exit 1
    fi

    # Convert tab/space-separated subnets to JSON array format
    local SUBNET_ARRAY
    SUBNET_ARRAY=$(echo "${SUBNETS}" | tr '\t ' '\n' | sed 's/^/"/;s/$/"/' |
 paste -sd ',' -)
    log "Subnets: ${SUBNETS}"
    log "Subnet array: [${SUBNET_ARRAY}]"

    # Get default security group for the VPC
    local DEFAULT_SG
    DEFAULT_SG=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=${DEFAULT_VPC}" "Name=group-
name,Values=default" \
        --query 'SecurityGroups[0].GroupId' \
        --output text)

    if [ "${DEFAULT_SG}" = "None" ] || [ "${DEFAULT_SG}" = "null" ] || [ -z
 "${DEFAULT_SG}" ]; then
        log "ERROR: No default security group found in VPC"
        exit 1
    fi

    log "Default security group: ${DEFAULT_SG}"
```

```
    # Step 1: Create IAM execution role
    log "Step 1: Creating IAM execution role..."

    # Create trust policy document with proper escaping
    cat > "${TRUST_POLICY_FILE}" << 'EOFPOLICY'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOFPOLICY
    CREATED_RESOURCES+=("FILE:${TRUST_POLICY_FILE}")

    # Validate trust policy file before using it
    if ! jq empty "${TRUST_POLICY_FILE}" 2>/dev/null; then
        log "ERROR: Trust policy file is not valid JSON"
        exit 1
    fi

    # Create the role
    aws iam create-role \
        --role-name "${ROLE_NAME}" \
        --assume-role-policy-document "file://${TRUST_POLICY_FILE}"
    aws iam tag-role --role-name "${ROLE_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=getting-started-batch-fargate
    CREATED_RESOURCES+=("IAM_ROLE:${ROLE_NAME}")

    # Attach policy
    aws iam attach-role-policy \
        --role-name "${ROLE_NAME}" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"

    log "IAM role created: ${ROLE_NAME}"

    # Wait for IAM role propagation
    log "Waiting for IAM role propagation (15 seconds)..."
```

```
    sleep 15

    # Step 2: Create compute environment
    log "Step 2: Creating Fargate compute environment..."

    aws batch create-compute-environment \
        --compute-environment-name "${COMPUTE_ENV_NAME}" \
        --type MANAGED \
        --state ENABLED \
        --tags project=doc-smith,tutorial=getting-started-batch-fargate \
        --compute-resources "{
            \"type\": \"FARGATE\",
            \"maxvCpus\": 256,
            \"subnets\": [${SUBNET_ARRAY}],
            \"securityGroupIds\": [\"${DEFAULT_SG}\"]
        }"
    CREATED_RESOURCES+=("COMPUTE_ENV:${COMPUTE_ENV_NAME}")

    # Wait for compute environment to be ready
    if ! wait_for_compute_env "${COMPUTE_ENV_NAME}"; then
        log "ERROR: Compute environment failed to reach VALID state"
        exit 1
    fi

    # Step 3: Create job queue
    log "Step 3: Creating job queue..."

    aws batch create-job-queue \
        --job-queue-name "${JOB_QUEUE_NAME}" \
        --state ENABLED \
        --priority 900 \
        --tags project=doc-smith,tutorial=getting-started-batch-fargate \
        --compute-environment-order "order=1,computeEnvironment=
${COMPUTE_ENV_NAME}"
    CREATED_RESOURCES+=("JOB_QUEUE:${JOB_QUEUE_NAME}")

    # Wait for job queue to be ready
    if ! wait_for_job_queue "${JOB_QUEUE_NAME}"; then
        log "ERROR: Job queue failed to reach ENABLED state"
        exit 1
    fi

    # Step 4: Create job definition
    log "Step 4: Creating job definition..."
```

```
    local CONTAINER_IMAGE="busybox:latest"
    validate_container_image "${CONTAINER_IMAGE}"

    aws batch register-job-definition \
        --job-definition-name "${JOB_DEF_NAME}" \
        --type container \
        --platform-capabilities FARGATE \
        --container-properties "{
            \"image\": \"${CONTAINER_IMAGE}\",
            \"resourceRequirements\": [
                {\"type\": \"VCPU\", \"value\": \"0.25\"},
                {\"type\": \"MEMORY\", \"value\": \"512\"}
            ],
            \"command\": [\"echo\", \"hello world\"],
            \"networkConfiguration\": {
                \"assignPublicIp\": \"DISABLED\"
            },
            \"executionRoleArn\": \"arn:aws:iam::${ACCOUNT_ID}:role/
${ROLE_NAME}\"
        }"

    log "Job definition created: ${JOB_DEF_NAME}"

    # Step 5: Submit job
    log "Step 5: Submitting job..."

    local JOB_ID
    JOB_ID=$(aws batch submit-job \
        --job-name "${JOB_NAME}" \
        --job-queue "${JOB_QUEUE_NAME}" \
        --job-definition "${JOB_DEF_NAME}" \
        --query 'jobId' \
        --output text)

    if [ -z "${JOB_ID}" ] || [ "${JOB_ID}" = "None" ]; then
        log "ERROR: Failed to submit job"
        exit 1
    fi

    log "Job submitted with ID: ${JOB_ID}"

    # Step 6: Wait for job completion and view output
    log "Step 6: Waiting for job completion..."
```

```
    if ! wait_for_job "${JOB_ID}"; then
        log "ERROR: Job failed or timed out"
        exit 1
    fi

    # Get log stream name
    log "Getting job logs..."
    local LOG_STREAM
    LOG_STREAM=$(aws batch describe-jobs \
        --jobs "${JOB_ID}" \
        --query 'jobs[0].attempts[0].taskProperties.containers[0].logStreamName'
\
        --output text)

    if [ "${LOG_STREAM}" != "None" ] && [ "${LOG_STREAM}" != "null" ] && [ -n
"${LOG_STREAM}" ]; then
        log "Log stream: ${LOG_STREAM}"
        log "Job output:"
        aws logs get-log-events \
            --log-group-name "/aws/batch/job" \
            --log-stream-name "${LOG_STREAM}" \
            --query 'events[*].message' \
            --output text 2>/dev/null | tee -a "${LOG_FILE}" || true
    else
        log "No log stream available for job"
    fi

    log "Tutorial completed successfully!"

    # Show created resources
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${resource}"
    done
    echo ""
    echo "============================================"
    echo "CLEANUP"
    echo "============================================"

    cleanup_resources
```

```
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - RegisterJobDefinition
  - SubmitJob
  - UpdateComputeEnvironment
  - UpdateJobQueue

Exemplos de código para o Amazon Bedrock usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Bedrock com um kit de desenvolvimento AWS de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon Bedrock: mais informações sobre o Amazon Bedrock.
- Referência da API Amazon Bedrock: detalhes sobre todas as ações disponíveis do Amazon Bedrock.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon Bedrock AWS SDKs
  - Olá, Amazon Bedrock
  - Ações para o Amazon Bedrock usando AWS SDKs
    - Use GetFoundationModelcom um AWS SDK
    - Use ListFoundationModelscom um AWS SDK
- Cenários para o uso do Amazon Bedrock AWS SDKs
  - Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

Exemplos básicos para o uso do Amazon Bedrock AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Bedrock com SDKs da AWS .

Exemplos

- Olá, Amazon Bedrock
- Ações para o Amazon Bedrock usando AWS SDKs
  - Use GetFoundationModelcom um AWS SDK
  - Use ListFoundationModelscom um AWS SDK

Olá, Amazon Bedrock

Os exemplos de código a seguir mostram como começar a usar o Amazon Bedrock.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon;
using Amazon.Bedrock;
using Amazon.Bedrock.Model;

namespace BedrockActions;

/// <summary>
/// This example shows how to list foundation models.
/// </summary>
internal class HelloBedrock
{
    /// <summary>
    /// Main method to call the ListFoundationModelsAsync method.
    /// </summary>
    /// <param name="args"> The command line arguments. </param>
    static async Task Main(string[] args)
    {
        // Specify a region endpoint where Amazon Bedrock is available. For
 a list of supported region see https://docs.aws.amazon.com/bedrock/latest/
userguide/what-is-bedrock.html#bedrock-regions
        AmazonBedrockClient bedrockClient = new(RegionEndpoint.USWest2);

        await ListFoundationModelsAsync(bedrockClient);

    }


    /// <summary>
    /// List foundation models.
    /// </summary>
    /// <param name="bedrockClient"> The Amazon Bedrock client. </param>
```

```
    private static async Task ListFoundationModelsAsync(AmazonBedrockClient
bedrockClient)
    {
        Console.WriteLine("List foundation models with no filter.");

        try
        {
            var response = await bedrockClient.ListFoundationModelsAsync(new
ListFoundationModelsRequest()
            {
            });

            if (response?.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                foreach (var fm in response.ModelSummaries)
                {
                    WriteToConsole(fm);
                }
            }
            else
            {
                Console.WriteLine("Something wrong happened");
            }
        }
        catch (AmazonBedrockException e)
        {
            Console.WriteLine(e.Message);
        }
    }


    /// <summary>
    /// Write the foundation model summary to console.
    /// </summary>
    /// <param name="foundationModel"> The foundation model summary to write to
console. </param>
    private static void WriteToConsole(FoundationModelSummary foundationModel)
    {
        Console.WriteLine($"{foundationModel.ModelId}, Customization:
{string.Join(", ", foundationModel.CustomizationsSupported)}, Stream:
{foundationModel.ResponseStreamingSupported}, Input: {string.Join(",
", foundationModel.InputModalities)}, Output: {string.Join(", ",
foundationModel.OutputModalities)}");
    }
```

```
}
```

- Para obter detalhes da API, consulte ListFoundationModelsa Referência AWS SDK para .NET da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/bedrock"
)

const region = "us-east-1"

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock client and
// list the available foundation models in your account and the chosen region.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(region))
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
```

```
 }
 bedrockClient := bedrock.NewFromConfig(sdkConfig)
 result, err := bedrockClient.ListFoundationModels(ctx,
 &bedrock.ListFoundationModelsInput{})
 if err != nil {
  fmt.Printf("Couldn't list foundation models. Here's why: %v\n", err)
  return
 }
 if len(result.ModelSummaries) == 0 {
  fmt.Println("There are no foundation models.")
 }
 for _, modelSummary := range result.ModelSummaries {
  fmt.Println(*modelSummary.ModelId)
 }
}
```

- Para obter detalhes da API, consulte ListFoundationModelsa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  ListFoundationModelsCommand,
} from "@aws-sdk/client-bedrock";

const REGION = "us-east-1";
const client = new BedrockClient({ region: REGION });
```

```
export const main = async () => {
  const command = new ListFoundationModelsCommand({});

  const response = await client.send(command);
  const models = response.modelSummaries;

  console.log("Listing the available Bedrock foundation models:");

  for (const model of models) {
    console.log("=".repeat(42));
    console.log(` Model: ${model.modelId}`);
    console.log("-".repeat(42));
    console.log(` Name: ${model.modelName}`);
    console.log(` Provider: ${model.providerName}`);
    console.log(` Model ARN: ${model.modelArn}`);
    console.log(` Input modalities: ${model.inputModalities}`);
    console.log(` Output modalities: ${model.outputModalities}`);
    console.log(` Supported customizations: ${model.customizationsSupported}`);
    console.log(` Supported inference types: ${model.inferenceTypesSupported}`);
    console.log(` Lifecycle status: ${model.modelLifecycle.status}`);
    console.log(`${"=".repeat(42)}\n`);
  }

  const active = models.filter(
    (m) => m.modelLifecycle.status === "ACTIVE",
  ).length;
  const legacy = models.filter(
    (m) => m.modelLifecycle.status === "LEGACY",
  ).length;

  console.log(
    `There are ${active} active and ${legacy} legacy foundation models in
 ${REGION}.`,
  );

  return response;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await main();
}
```

- Para obter detalhes da API, consulte ListFoundationModelsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Lists the available Amazon Bedrock models.
"""
import logging
import json
import boto3


from botocore.exceptions import ClientError


logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


def list_foundation_models(bedrock_client):
    """
    Gets a list of available Amazon Bedrock foundation models.

    :return: The list of available bedrock foundation models.
    """

    try:
        response = bedrock_client.list_foundation_models()
        models = response["modelSummaries"]
```

```
        logger.info("Got %s foundation models.", len(models))
        return models

    except ClientError:
        logger.error("Couldn't list foundation models.")
        raise


def main():
    """Entry point for the example. Uses the AWS SDK for Python (Boto3)
    to create an Amazon Bedrock client. Then lists the available Bedrock models
    in the region set in the callers profile and credentials.
    """

    bedrock_client = boto3.client(service_name="bedrock")

    fm_models = list_foundation_models(bedrock_client)
    for model in fm_models:
        print(f"Model: {model['modelName']}")
        print(json.dumps(model, indent=2))
        print("---------------------------\n")

    logger.info("Done.")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ListFoundationModelsReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import Foundation

import AWSBedrock

struct ExampleCommand: ParsableCommand {
    static var configuration = CommandConfiguration(
        commandName: "ListFoundationModels",
        abstract: """
        This example demonstrates how to retrieve a list of the available
        foundation models from Amazon Bedrock.
        """,
        discussion: """
        """
    )

    /// Construct a string listing the specified modalities.
    ///
    /// - Parameter modalities: An array of the modalities to list.
    ///
    /// - Returns: A string with a human-readable list of modalities.
    func buildModalityList(modalities: [BedrockClientTypes.ModelModality]?) ->
 String {
        var first = true
        var str = ""

        if modalities == nil {
            return "<none>"
        }

        for modality in modalities! {
            if !first {
                str += ", "
            }
            first = false
            str += modality.rawValue
        }

        return str
    }

    /// Construct a string listing the specified customizations.
```

```
    ///
    /// - Parameter customizations: An array of the customizations to list.
    ///
    /// - Returns: A string listing the customizations.
    func buildCustomizationList(customizations:
[BedrockClientTypes.ModelCustomization]?) -> String {
        var first = true
        var str = ""

        if customizations == nil {
            return "<none>"
        }

        for customization in customizations! {
            if !first {
                str += ", "
            }
            first = false
            str += customization.rawValue
        }

        return str
    }

    /// Construct a string listing the specified inferences.
    ///
    /// - Parameter inferences: An array of inferences to list.
    ///
    /// - Returns: A string listing the specified inferences.
    func buildInferenceList(inferences: [BedrockClientTypes.InferenceType]?) ->
String {
        var first = true
        var str = ""

        if inferences == nil {
            return "<none>"
        }

        for inference in inferences! {
            if !first {
                str += ", "
            }
            first = false
            str += inference.rawValue
```

```
        }

        return str
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // Always use the Region "us-east-1" to have access to the most models.
        let config = try await BedrockClient.BedrockClientConfiguration(region:
 "us-east-1")
        let bedrockClient = BedrockClient(config: config)

        let output = try await bedrockClient.listFoundationModels(
            input: ListFoundationModelsInput()
        )

        guard let summaries = output.modelSummaries else {
            print("No models returned.")
            return
        }

        // Output a list of the models with their details.
        for summary in summaries {
            print("==========================================")
            print(" Model ID: \(summary.modelId ?? "<unknown>")")
            print("------------------------------------------")
            print(" Name: \(summary.modelName ?? "<unknown>")")
            print(" Provider: \(summary.providerName ?? "<unknown>")")
            print(" Input modalities: \(buildModalityList(modalities:
 summary.inputModalities))")
            print(" Output modalities: \(buildModalityList(modalities:
 summary.outputModalities))")
            print(" Supported customizations:
 \(buildCustomizationList(customizations: summary.customizationsSupported ))")
            print(" Supported inference types: \(buildInferenceList(inferences:
 summary.inferenceTypesSupported))")
            print("------------------------------------------\n")
        }

        print("\(summaries.count) models available.")
    }
}

/// The program's asynchronous entry point.
```

```
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte ListFoundationModelsa referência da API AWS SDK for Swift.

Ações para o Amazon Bedrock usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Bedrock com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Bedrock e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para o uso do Amazon Bedrock AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon Bedrock API Reference.

Exemplos

- Use GetFoundationModelcom um AWS SDK
- Use ListFoundationModelscom um AWS SDK

Use **`GetFoundationModel`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetFoundationModel`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha detalhes de um modelo de base usando o cliente síncrono Amazon Bedrock.

```
   /**
    * Get details about an Amazon Bedrock foundation model.
    *
    * @param bedrockClient   The service client for accessing Amazon Bedrock.
    * @param modelIdentifier The model identifier.
    * @return An object containing the foundation model's details.
    */
   public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
       try {
           GetFoundationModelResponse response =
bedrockClient.getFoundationModel(
                   r -> r.modelIdentifier(modelIdentifier)
           );

           FoundationModelDetails model = response.modelDetails();

           System.out.println(" Model ID:                     " +
model.modelId());
           System.out.println(" Model ARN:                    " +
model.modelArn());
           System.out.println(" Model Name:                   " +
model.modelName());
           System.out.println(" Provider Name:                " +
model.providerName());
           System.out.println(" Lifecycle status:             " +
model.modelLifecycle().statusAsString());
           System.out.println(" Input modalities:             " +
model.inputModalities());
           System.out.println(" Output modalities:            " +
model.outputModalities());
```

```
            System.out.println(" Supported customizations:     " +
model.customizationsSupported());
            System.out.println(" Supported inference types:    " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ValidationException e) {
            throw new IllegalArgumentException(e.getMessage());
        } catch (SdkException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

Obtenha detalhes de um modelo de base usando o cliente assíncrono Amazon Bedrock.

```
    /**
     * Get details about an Amazon Bedrock foundation model.
     *
     * @param bedrockClient   The async service client for accessing Amazon
Bedrock.
     * @param modelIdentifier The model identifier.
     * @return An object containing the foundation model's details.
     */
    public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
        try {
            CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
                    r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = future.get().modelDetails();

            System.out.println(" Model ID:                      " +
model.modelId());
            System.out.println(" Model ARN:                     " +
model.modelArn());
```

```
            System.out.println(" Model Name:                    " +
model.modelName());
            System.out.println(" Provider Name:                 " +
model.providerName());
            System.out.println(" Lifecycle status:              " +
model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:              " +
model.inputModalities());
            System.out.println(" Output modalities:             " +
model.outputModalities());
            System.out.println(" Supported customizations:      " +
model.customizationsSupported());
            System.out.println(" Supported inference types:     " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ExecutionException e) {
            if (e.getMessage().contains("ValidationException")) {
                throw new IllegalArgumentException(e.getMessage());
            } else {
                System.err.println(e.getMessage());
                throw new RuntimeException(e);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte GetFoundationModela Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha detalhes de um modelo de base.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  GetFoundationModelCommand,
} from "@aws-sdk/client-bedrock";

/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @return {FoundationModelDetails} - The list of available bedrock foundation
 models.
 */
export const getFoundationModel = async () => {
  const client = new BedrockClient();

  const command = new GetFoundationModelCommand({
    modelIdentifier: "amazon.titan-embed-text-v1",
  });

  const response = await client.send(command);

  return response.modelDetails;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const model = await getFoundationModel();
  console.log(model);
}
```

- Para obter detalhes da API, consulte GetFoundationModela Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha detalhes de um modelo de base.

```
    def get_foundation_model(self, model_identifier):
        """
        Get details about an Amazon Bedrock foundation model.

        :return: The foundation model's details.
        """

        try:
            return self.bedrock_client.get_foundation_model(
                modelIdentifier=model_identifier
            )["modelDetails"]
        except ClientError:
            logger.error(
                f"Couldn't get foundation models details for {model_identifier}"
            )
            raise
```

- Para obter detalhes da API, consulte a GetFoundationModelReferência da API AWS SDK for Python (Boto3).

Use **`ListFoundationModels`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListFoundationModels`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Bedrock disponíveis.

```
    /// <summary>
    /// List foundation models.
    /// </summary>
    /// <param name="bedrockClient"> The Amazon Bedrock client. </param>
    private static async Task ListFoundationModelsAsync(AmazonBedrockClient
bedrockClient)
    {
        Console.WriteLine("List foundation models with no filter.");

        try
        {
            var response = await bedrockClient.ListFoundationModelsAsync(new
ListFoundationModelsRequest()
            {
            });

            if (response?.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                foreach (var fm in response.ModelSummaries)
                {
                    WriteToConsole(fm);
                }
            }
            else
            {
                Console.WriteLine("Something wrong happened");
```

```
            }
        }
        catch (AmazonBedrockException e)
        {
            Console.WriteLine(e.Message);
        }
    }
```

- Para obter detalhes da API, consulte ListFoundationModelsa Referência AWS SDK para .NET da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Bedrock disponíveis.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/bedrock"
 "github.com/aws/aws-sdk-go-v2/service/bedrock/types"
)

// FoundationModelWrapper encapsulates Amazon Bedrock actions used in the
 examples.
// It contains a Bedrock service client that is used to perform foundation model
 actions.
type FoundationModelWrapper struct {
 BedrockClient *bedrock.Client
}
```

```
// ListPolicies lists Bedrock foundation models that you can use.
func (wrapper FoundationModelWrapper) ListFoundationModels(ctx context.Context)
 ([]types.FoundationModelSummary, error) {

 var models []types.FoundationModelSummary

 result, err := wrapper.BedrockClient.ListFoundationModels(ctx,
 &bedrock.ListFoundationModelsInput{})

 if err != nil {
  log.Printf("Couldn't list foundation models. Here's why: %v\n", err)
 } else {
  models = result.ModelSummaries
 }
 return models, err
}
```

- Para obter detalhes da API, consulte ListFoundationModelsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os modelos de base do Amazon Bedrock disponíveis usando o cliente de base do Amazon Bedrock síncrono.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
```

```
     * @param bedrockClient The service client for accessing Amazon Bedrock.
     * @return A list of objects containing the foundation models' details
     */
    public static List<FoundationModelSummary> listFoundationModels(BedrockClient
 bedrockClient) {

        try {
            ListFoundationModelsResponse response =
 bedrockClient.listFoundationModels(r -> {});

            List<FoundationModelSummary> models = response.modelSummaries();

            if (models.isEmpty()) {
                System.out.println("No available foundation models in " +
 region.toString());
            } else {
                for (FoundationModelSummary model : models) {
                    System.out.println("Model ID: " + model.modelId());
                    System.out.println("Provider: " + model.providerName());
                    System.out.println("Name:     " + model.modelName());
                    System.out.println();
                }
            }

            return models;

        } catch (SdkClientException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

Liste os modelos de base do Amazon Bedrock disponíveis usando o cliente de base do Amazon Bedrock assíncrono.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
     * @param bedrockClient The async service client for accessing Amazon
 Bedrock.
     * @return A list of objects containing the foundation models' details
```

```
     */
    public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
        try {
            CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

            List<FoundationModelSummary> models = future.get().modelSummaries();

            if (models.isEmpty()) {
                System.out.println("No available foundation models in " +
region.toString());
            } else {
                for (FoundationModelSummary model : models) {
                    System.out.println("Model ID: " + model.modelId());
                    System.out.println("Provider: " + model.providerName());
                    System.out.println("Name:     " + model.modelName());
                    System.out.println();
                }
            }

            return models;

        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        } catch (ExecutionException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte ListFoundationModelsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os modelos de base disponíveis.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  ListFoundationModelsCommand,
} from "@aws-sdk/client-bedrock";

/**
 * List the available Amazon Bedrock foundation models.
 *
 * @return {FoundationModelSummary[]} - The list of available bedrock foundation
 models.
 */
export const listFoundationModels = async () => {
  const client = new BedrockClient();

  const input = {
    // byProvider: 'STRING_VALUE',
    // byCustomizationType: 'FINE_TUNING' || 'CONTINUED_PRE_TRAINING',
    // byOutputModality: 'TEXT' || 'IMAGE' || 'EMBEDDING',
    // byInferenceType: 'ON_DEMAND' || 'PROVISIONED',
  };

  const command = new ListFoundationModelsCommand(input);

  const response = await client.send(command);

  return response.modelSummaries;
};
```

```
// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const models = await listFoundationModels();
  console.log(models);
}
```

- Para obter detalhes da API, consulte ListFoundationModelsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Amazon Bedrock disponíveis.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient.fromEnvironment { region = "us-east-1" }.use { bedrockClient ->
        val response =
 bedrockClient.listFoundationModels(ListFoundationModelsRequest {})
        response.modelSummaries?.forEach { model ->
            println("==========================================")
            println(" Model ID: ${model.modelId}")
            println("------------------------------------------")
            println(" Name: ${model.modelName}")
            println(" Provider: ${model.providerName}")
            println(" Input modalities: ${model.inputModalities}")
            println(" Output modalities: ${model.outputModalities}")
            println(" Supported customizations:
 ${model.customizationsSupported}")
            println(" Supported inference types:
 ${model.inferenceTypesSupported}")
            println("------------------------------------------\n")
        }
        return response.modelSummaries
```

```
    }
}
```

- Para obter detalhes da API, consulte a ListFoundationModelsreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Amazon Bedrock disponíveis.

```
    public function listFoundationModels()
    {
        $bedrockClient = new BedrockClient([
            'region' => 'us-west-2',
            'profile' => 'default'
        ]);
        $response = $bedrockClient->listFoundationModels();
        return $response['modelSummaries'];
    }
```

- Para obter detalhes da API, consulte ListFoundationModelsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Amazon Bedrock disponíveis.

```
    def list_foundation_models(self):
        """
        List the available Amazon Bedrock foundation models.

        :return: The list of available bedrock foundation models.
        """

        try:
            response = self.bedrock_client.list_foundation_models()
            models = response["modelSummaries"]
            logger.info("Got %s foundation models.", len(models))
            return models

        except ClientError:
            logger.error("Couldn't list foundation models.")
            raise
```

- Para obter detalhes da API, consulte a ListFoundationModelsReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSBedrock

        // Always use the Region "us-east-1" to have access to the most models.
        let config = try await BedrockClient.BedrockClientConfiguration(region:
 "us-east-1")
        let bedrockClient = BedrockClient(config: config)

        let output = try await bedrockClient.listFoundationModels(
            input: ListFoundationModelsInput()
        )

        guard let summaries = output.modelSummaries else {
            print("No models returned.")
            return
        }

        // Output a list of the models with their details.
        for summary in summaries {
            print("==========================================")
            print(" Model ID: \(summary.modelId ?? "<unknown>")")
            print("------------------------------------------")
            print(" Name: \(summary.modelName ?? "<unknown>")")
            print(" Provider: \(summary.providerName ?? "<unknown>")")
            print(" Input modalities: \(buildModalityList(modalities:
 summary.inputModalities))")
            print(" Output modalities: \(buildModalityList(modalities:
 summary.outputModalities))")
            print(" Supported customizations:
 \(buildCustomizationList(customizations: summary.customizationsSupported ))")
            print(" Supported inference types: \(buildInferenceList(inferences:
 summary.inferenceTypesSupported))")
            print("------------------------------------------\n")
```

```
        }
```

- Para obter detalhes da API, consulte ListFoundationModelsa referência da API AWS SDK for Swift.

Cenários para o uso do Amazon Bedrock AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Bedrock com AWS SDKs. Esses casos mostram como executar tarefas específicas chamando várias funções no Amazon Bedrock ou em combinação com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

Python

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.

- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Step Functions

Exemplos de código para o Amazon Bedrock Runtime usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon Bedrock Runtime: mais informações sobre o Amazon Bedrock Runtime.
- Referência da API Amazon Bedrock Runtime: detalhes sobre todas as ações disponíveis do Amazon Bedrock Runtime.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon Bedrock Runtime usando AWS SDKs
  - Olá, Amazon Bedrock Runtime
- Cenários para o Amazon Bedrock Runtime usando AWS SDKs
  - Crie um aplicativo de amostra que ofereça playgrounds para interagir com os modelos da Amazon Bedrock Foundation usando um AWS SDK
  - Um exemplo completo que mostra como criar e invocar prompts gerenciados do Amazon Bedrock usando um AWS SDK
  - Gere vídeos a partir de solicitações de texto usando o Amazon Bedrock e Nova-Reel
  - Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions
  - Exemplo de uso da ferramenta ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada
- Amazon Nova para Amazon Bedrock Runtime
  - Invocar o Amazon Nova no Amazon Bedrock usando a API Converse do Bedrock
  - Invocar o Amazon Nova no Amazon Bedrock usando a API Converse do Bedrock
  - Enviar e processar um documento com o Amazon Nova no Amazon Bedrock
  - Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada
- Amazon Nova Canvas para Amazon Bedrock Runtime
  - Invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem
- Amazon Nova Reel para Amazon Bedrock Runtime

- Usar o Amazon Nova Reel para gerar um vídeo de um prompt de texto
- Gerador de Imagens do Amazon Titan para Amazon Bedrock Runtime
  - Invocar o Amazon Titan Image no Amazon Bedrock para gerar uma imagem
- Amazon Titan Text para Amazon Bedrock Runtime
  - Invocar modelos Amazon Titan Text no Amazon Bedrock usando a API InvokeModel
- Incorporador de Texto do Amazon Titan para Amazon Bedrock Runtime
  - Invocar o Incorporador de Texto do Amazon Titan no Amazon Bedrock
- Claude da Anthropic para Amazon Bedrock Runtime
  - Invocar o Claude da Anthropic no Amazon Bedrock usando a API Converse do Bedrock
  - Invocar o Claude da Anthropic no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta
  - Enviar e processar um documento com o Claude da Anthropic no Amazon Bedrock
  - Usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock
  - Usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock
  - Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada
- Command da Cohere para Amazon Bedrock Runtime
  - Invocar o Command da Cohere no Amazon Bedrock usando a API Converse do Bedrock
  - Invocar o Command da Cohere no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta
  - Enviar e processar um documento com modelos Cohere Command no Amazon Bedrock
  - Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada
- DeepSeek para Amazon Bedrock Runtime
  - Envie e processe um documento com o DeepSeek Amazon Bedrock
- Llama da Meta para Amazon Bedrock Runtime
  - Invocar o Llama da Meta no Amazon Bedrock usando a API Converse do Bedrock
  - Invocar o Llama da Meta no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta
  - Enviar e processar um documento com o Llama no Amazon Bedrock
- Mistral AI para Amazon Bedrock Runtime

- Invocar a Mistral no Amazon Bedrock usando a API Converse do Bedrock
- Invocar a Mistral no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta
- Enviar e processar um documento com modelos da Mistral no Amazon Bedrock
- Núcleo de imagem estável para Amazon Bedrock Runtime
  - Invoque o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem

Exemplos básicos para o Amazon Bedrock Runtime usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Bedrock Runtime com SDKs da AWS .

Exemplos

- Olá, Amazon Bedrock Runtime

Olá, Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como começar a usar o Amazon Bedrock Runtime.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

namespace BedrockRuntimeActions;

/// <summary>
/// This example shows how to use the Converse API to send a text message
```

```
/// to Amazon Bedrock.
/// </summary>
internal class HelloBedrockRuntime
{
    static async Task Main(string[] args)
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // You can change the region to match your setup.
        var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USWest2);

        // Set the model ID, e.g., Claude Haiku.
        // The "global." prefix enables cross-region inference, allowing the
 request
        // to be routed to the nearest available region for the specified model.
        var modelId = "global.anthropic.claude-haiku-4-5-20251001-v1:0";

        // Define the user message.
        var userMessage = "Hi. In a short paragraph, explain what you can do.";

        // Create a request with the model ID, the user message, and an inference
 configuration.
        var request = new ConverseRequest
        {
            ModelId = modelId,
            Messages = new List<Message>
            {
                new Message
                {
                    Role = ConversationRole.User,
                    Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
                }
            }
        };

        try
        {
            // Send the request to the Bedrock Runtime and wait for the response.
            var response = await client.ConverseAsync(request);

            // Extract and print the response text.
            string responseText = response?.Output?.Message?.Content?[0]?.Text ??
 "";
            Console.WriteLine(responseText);
```

```
        }
        catch (AmazonBedrockRuntimeException e)
        {
            Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason:
 {e.Message}");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

namespace BedrockRuntimeActions;

/// <summary>
/// This example shows how to use the Converse API to send a text message
/// to Amazon Bedrock.
/// </summary>
internal class HelloBedrockRuntime
{
    static async Task Main(string[] args)
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // You can change the region to match your setup.
        var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USWest2);

        // Set the model ID, e.g., Claude Haiku.
```

```
        // The "global." prefix enables cross-region inference, allowing the
 request
        // to be routed to the nearest available region for the specified model.
        var modelId = "global.anthropic.claude-haiku-4-5-20251001-v1:0";

        // Define the user message.
        var userMessage = "Hi. In a short paragraph, explain what you can do.";

        // Create a request with the model ID, the user message, and an inference
 configuration.
        var request = new ConverseRequest
        {
            ModelId = modelId,
            Messages = new List<Message>
            {
                new Message
                {
                    Role = ConversationRole.User,
                    Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
                }
            }
        };

        try
        {
            // Send the request to the Bedrock Runtime and wait for the response.
            var response = await client.ConverseAsync(request);

            // Extract and print the response text.
            string responseText = response?.Output?.Message?.Content?[0]?.Text ??
 "";
            Console.WriteLine(responseText);
        }
        catch (AmazonBedrockRuntimeException e)
        {
            Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason:
 {e.Message}");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "flag"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock Runtime client
// and invokes Anthropic Claude using the Converse API.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {

 region := flag.String("region", "us-east-1", "The AWS region")
 flag.Parse()

 fmt.Printf("Using AWS region: %s\n", *region)

 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(*region))
```

```
if err != nil {
 fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
 fmt.Println(err)
 return
}

client := bedrockruntime.NewFromConfig(sdkConfig)

// Set the model ID, e.g., Claude Haiku.
// The "global." prefix enables cross-region inference, allowing the request
// to be routed to the nearest available region for the specified model.
modelId := "global.anthropic.claude-haiku-4-5-20251001-v1:0"

// Start a conversation with the user message.
prompt := "Hello. In a short paragraph, explain what you can do."

message := types.Message{
 Content: []types.ContentBlock{
  &types.ContentBlockMemberText{Value: prompt},
 },
 Role: types.ConversationRoleUser,
}

fmt.Printf("Model: %s\n", modelId)
fmt.Printf("Prompt: %s\n\n", prompt)

result, err := client.Converse(ctx, &bedrockruntime.ConverseInput{
 ModelId:  aws.String(modelId),
 Messages: []types.Message{message},
})

if err != nil {
 log.Fatalf("ERROR: Can't invoke '%s'. Reason: %v\n", modelId, err)
}

// Extract and print the response text.
responseMessage, ok := result.Output.(*types.ConverseOutputMemberMessage)
if !ok {
 log.Fatal("ERROR: Unexpected output type from Converse API")
}
responseContentBlock := responseMessage.Value.Content[0]
text, ok := responseContentBlock.(*types.ContentBlockMemberText)
if !ok {
```

```
  log.Fatal("ERROR: Unexpected content block type from Converse API")
 }
 fmt.Printf("Response: %s\n", text.Value)
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Go .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Send a prompt to Amazon Bedrock using the Converse API.

const AWS_REGION = "us-east-1";

// Set the model ID, e.g., Claude Haiku.
// The "global." prefix enables cross-region inference, allowing the request
// to be routed to the nearest available region for the specified model.
const MODEL_ID = "global.anthropic.claude-haiku-4-5-20251001-v1:0";
const PROMPT = "Hi. In a short paragraph, explain what you can do.";

const hello = async () => {
  console.log("=".repeat(35));
  console.log("Welcome to the Amazon Bedrock demo!");
  console.log("=".repeat(35));

  console.log("Model: Anthropic Claude Haiku");
```

```
  console.log(`Prompt: ${PROMPT}\n`);
  console.log("Invoking model...\n");

  // Create a new Bedrock Runtime client instance.
  const client = new BedrockRuntimeClient({ region: AWS_REGION });

  // Create the command with the model ID, the user message, and a basic
 configuration.
  const command = new ConverseCommand({
    modelId: MODEL_ID,
    messages: [
      {
        role: "user",
        content: [{ text: PROMPT }],
      },
    ],
  });

  // Send the command to the model and wait for the response.
  try {
    const response = await client.send(command);

    // Extract and print the response text.
    const responseText = response.output.message.content[0].text;
    console.log(`Response: ${responseText}`);
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "BedrockRuntimeServiceException"
    ) {
      console.error(
        `ERROR: Can't invoke '${MODEL_ID}'. Reason: ${caught.message}`,
      );
      throw caught;
    }
    throw caught;
  }
};

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await hello();
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

Cenários para o Amazon Bedrock Runtime usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Bedrock Runtime com AWS SDKs. Esses cenários mostram como executar tarefas específicas chamando vários perfis no Amazon Bedrock Runtime ou em conjunto com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie um aplicativo de amostra que ofereça playgrounds para interagir com os modelos da Amazon Bedrock Foundation usando um AWS SDK
- Um exemplo completo que mostra como criar e invocar prompts gerenciados do Amazon Bedrock usando um AWS SDK
- Gere vídeos a partir de solicitações de texto usando o Amazon Bedrock e Nova-Reel
- Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions
- Exemplo de uso da ferramenta ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada

Crie um aplicativo de amostra que ofereça playgrounds para interagir com os modelos da Amazon Bedrock Foundation usando um AWS SDK

Os exemplos de código a seguir mostram como criar playgrounds para interagir com os modelos da base do Amazon Bedrock em diferentes modalidades.

.NET

SDK para .NET

O .NET Foundation Model (FM) Playground é um aplicativo de amostra do .NET MAUI Blazor que mostra como usar o Amazon Bedrock a partir do código C#. Este exemplo mostra

como os desenvolvedores.NET e C# podem usar o Amazon Bedrock para criar aplicativos AI-enabled generativos. É possível testar e interagir com os modelos de base do Amazon Bedrock usando os quatro playgrounds a seguir:

- Um playground de texto.
- Um playground de chat.
- Um playground de chat por voz.
- Um playground de imagens.

O exemplo também lista e exibe os modelos de base aos quais você tem acesso e respectivas características. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock Runtime

Java

SDK para Java 2.x

O Java Foundation Model (FM) Playground é um aplicativo de amostra da Spring Boot que mostra como usar o Amazon Bedrock com Java. Este exemplo mostra como os desenvolvedores Java podem usar o Amazon Bedrock para criar aplicativos generativos AI-enabled . É possível testar e interagir com os modelos de base do Amazon Bedrock usando os três playgrounds a seguir:

- Um playground de texto.
- Um playground de chat.
- Um playground de imagens.

O exemplo também lista e exibe os modelos de base aos quais você tem acesso e respectivas características. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock Runtime

Python

SDK para Python (Boto3)

O Python Foundation Model (FM) Playground é um aplicativo de Python/FastAPI amostra que mostra como usar o Amazon Bedrock com Python. Este exemplo mostra como os desenvolvedores de Python podem usar o Amazon Bedrock para criar aplicativos generativos. AI-enabled É possível testar e interagir com os modelos de base do Amazon Bedrock usando os três playgrounds a seguir:

- Um playground de texto.
- Um playground de chat.
- Um playground de imagens.

O exemplo também lista e exibe os modelos de base aos quais você tem acesso e respectivas características. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock Runtime

Um exemplo completo que mostra como criar e invocar prompts gerenciados do Amazon Bedrock usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um prompt gerenciado.
- Criar uma versão do prompt.
- Invocar o prompt usando a versão.
- Limpar os recursos (opcional).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar e invocar um prompt gerenciado.

```
import argparse
import boto3
import logging
import time

# Now import the modules
from prompt import create_prompt, create_prompt_version, delete_prompt
from run_prompt import invoke_prompt

logging.basicConfig(
    level=logging.INFO,
    format='%(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)



def run_scenario(bedrock_client, bedrock_runtime_client, model_id, cleanup=True):
    """
    Runs the Amazon Bedrock managed prompt scenario.

    Args:
        bedrock_client: The Amazon Bedrock Agent client.
        bedrock_runtime_client: The Amazon Bedrock Runtime client.
        model_id (str): The model ID to use for the prompt.
        cleanup (bool): Whether to clean up resources at the end of the scenario.

    Returns:
        dict: A dictionary containing the created resources.
    """
```

```
    prompt_id = None

    try:
        # Step 1: Create a prompt
        print("\n=== Step 1: Creating a prompt ===")
        prompt_name = f"PlaylistGenerator-{int(time.time())}"
        prompt_description = "Playlist generator"
        prompt_template = """
          Make me a {{genre}} playlist consisting of the following number of
songs: {{number}}."""

        create_response = create_prompt(
            bedrock_client,
            prompt_name,
            prompt_description,
            prompt_template,
            model_id
        )

        prompt_id = create_response['id']
        print(f"Created prompt: {prompt_name} with ID: {prompt_id}")

        # Create a version of the prompt
        print("\n=== Creating a version of the prompt ===")
        version_response = create_prompt_version(
            bedrock_client,
            prompt_id,
            description="Initial version of the product description generator"
        )

        prompt_version_arn = version_response['arn']
        prompt_version = version_response['version']

        print(f"Created prompt version: {prompt_version}")
        print(f"Prompt version ARN: {prompt_version_arn}")

        # Step 2: Invoke the prompt directly
        print("\n=== Step 2: Invoking the prompt ===")
        input_variables = {
            "genre": "pop",
            "number": "2",
           }

        # Use the ARN from the create_prompt_version response
```

```
        result = invoke_prompt(
            bedrock_runtime_client,
            prompt_version_arn,
            input_variables
        )
        # Display the playlist
        print(f"\n{result}")


        # Step 3: Clean up resources (optional)
        if cleanup:
            print("\n=== Step 3: Cleaning up resources ===")

            # Delete the prompt
            print(f"Deleting prompt {prompt_id}...")
            delete_prompt(bedrock_client, prompt_id)

            print("Cleanup complete")
        else:
            print("\n=== Resources were not cleaned up ===")
            print(f"Prompt ID: {prompt_id}")



    except Exception as e:
        logger.exception("Error in scenario: %s", str(e))

        # Attempt to clean up if an error occurred and cleanup was requested
        if cleanup and prompt_id:
            try:
                print("\nCleaning up resources after error...")

                # Delete the prompt
                try:
                    delete_prompt(bedrock_client, prompt_id)
                    print("Cleanup after error complete")
                except Exception as cleanup_error:
                    logger.error("Error during cleanup: %s", str(cleanup_error))
            except Exception as final_error:
                logger.error("Final error during cleanup: %s", str(final_error))

        # Re-raise the original exception
        raise
```

```
def main():
    """
    Entry point for the Amazon Bedrock managed prompt scenario.
    """
    parser = argparse.ArgumentParser(
        description="Run the Amazon Bedrock managed prompt scenario."
    )
    parser.add_argument(
        '--region',
        default='us-east-1',
        help="The AWS Region to use."
    )
    parser.add_argument(
        '--model-id',
        default='anthropic.claude-v2',
        help="The model ID to use for the prompt."
    )
    parser.add_argument(
        '--cleanup',
        action='store_true',
        default=True,
        help="Clean up resources at the end of the scenario."
    )
    parser.add_argument(
        '--no-cleanup',
        action='store_false',
        dest='cleanup',
        help="Don't clean up resources at the end of the scenario."
    )
    args = parser.parse_args()

    bedrock_client = boto3.client('bedrock-agent', region_name=args.region)
    bedrock_runtime_client = boto3.client('bedrock-runtime',
 region_name=args.region)

    print("=== Amazon Bedrock Managed Prompt Scenario ===")
    print(f"Region: {args.region}")
    print(f"Model ID: {args.model_id}")
    print(f"Cleanup resources: {args.cleanup}")

    try:
        run_scenario(
            bedrock_client,
            bedrock_runtime_client,
```

```
            args.model_id,
            args.cleanup
        )

    except Exception as e:
        logger.exception("Error running scenario: %s", str(e))

if __name__ == "__main__":
    main()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - Converse
  - CreatePrompt
  - CreatePromptVersion
  - DeletePrompt

Gere vídeos a partir de solicitações de texto usando o Amazon Bedrock e Nova-Reel

O exemplo de código a seguir mostra como usar um aplicativo Spring Boot que gera vídeos a partir de solicitações de texto usando o Amazon Bedrock e o Nova-Reel modelo.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere vídeos a partir de solicitações de texto usando o Amazon Bedrock e. Nova-Reel

```
import org.springframework.stereotype.Service;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

@Service
public class VideoGenerationService {

    public GenerateVideoResponse generateVideo(String prompt) {

        // add S3 bucket you want to store your generated videos
        String s3Bucket = "s3://mygeneratedvidoenovatest";


        //Create json request as an instance of Document class
        Document novaRequest = prepareDocument(prompt);

        // Create request
        StartAsyncInvokeRequest request = StartAsyncInvokeRequest.builder()
                .modelId("amazon.nova-reel-v1:0")
                .modelInput(novaRequest)
                .outputDataConfig(AsyncInvokeOutputDataConfig.builder()

 .s3OutputDataConfig(AsyncInvokeS3OutputDataConfig.builder().s3Uri(s3Bucket).build())
                        .build())
                .build();

        try (BedrockRuntimeAsyncClient bedrockClient =
 getBedrockRuntimeAsyncClient()) {
            CompletableFuture<StartAsyncInvokeResponse>
 startAsyncInvokeResponseCompletableFuture =
 bedrockClient.startAsyncInvoke(request);

            //blocking operation to wait for the AWS API response
            StartAsyncInvokeResponse startAsyncInvokeResponse =
 startAsyncInvokeResponseCompletableFuture.get();
            System.out.println("invocation ARN: " +
 startAsyncInvokeResponse.invocationArn());

            GenerateVideoResponse response = new GenerateVideoResponse();
            response.setStatus("inProgress");
            response.setExecutionArn(startAsyncInvokeResponse.invocationArn());

            return response;
```

```
        } catch (Exception e) {
            System.out.println(e);
            throw new RuntimeException(e);
        }

    }

    public GenerateVideoResponse checkGenerationStatus(String invocationArn) {
        GenerateVideoResponse response = new GenerateVideoResponse();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            //creating async request to fetch status by invocation Arn
            GetAsyncInvokeRequest asyncRequest =
GetAsyncInvokeRequest.builder().invocationArn(invocationArn).build();

            CompletableFuture<GetAsyncInvokeResponse> asyncInvoke =
bedrockClient.getAsyncInvoke(asyncRequest);

            //blocking operation to wait for the AWS API response
            GetAsyncInvokeResponse asyncInvokeResponse = asyncInvoke.get();
            System.out.println("Invocation status =" +
asyncInvokeResponse.statusAsString());

            response.setExecutionArn(invocationArn);
            response.setStatus(asyncInvokeResponse.statusAsString());
            return response;
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }

    }

    private static BedrockRuntimeAsyncClient getBedrockRuntimeAsyncClient() {
        BedrockRuntimeAsyncClient bedrockClient =
BedrockRuntimeAsyncClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
        return bedrockClient;
    }

    private static Document prepareDocument(String prompt) {
```

```
        Document textToVideoParams = Document.mapBuilder()
                .putString("text", prompt)
                .build();

        Document videoGenerationConfig = Document.mapBuilder()
                .putNumber("durationSeconds", 6)
                .putNumber("fps", 24)
                .putString("dimension", "1280x720")
                .build();

        Document novaRequest = Document.mapBuilder()
                .putString("taskType", "TEXT_VIDEO")
                .putDocument("textToVideoParams", textToVideoParams)
                .putDocument("videoGenerationConfig", videoGenerationConfig)
                .build();
        return novaRequest;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetAsyncInvoke
  - StartAsyncInvoke

Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

Python

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Step Functions

Exemplo de uso da ferramenta ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada

Os exemplos de código a seguir mostram como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;

namespace ConverseToolScenario;

public static class ConverseToolScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
    The script interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
    input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve
 current weather data for a given location.
   */

    public static BedrockActionsWrapper _bedrockActionsWrapper = null!;
```

```
    public static WeatherTool _weatherTool = null!;
    public static bool _interactive = true;

    // Change this string to use a different model with Converse API.
    private static string model_id = "amazon.nova-lite-v1:0";

    private static string system_prompt = @"
        You are a weather assistant that provides current weather data for user-
specified locations using only
        the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
        If the user specifies a state, country, or region, infer the locations of
 cities within that state.
        If the user provides coordinates, infer the approximate location and
 refer to it in your response.
        To use the tool, you strictly apply the provided tool specification.

        - Explain your step-by-step process, and give brief updates before each
 step.
        - Only use the Weather_Tool for data. Never guess or make up
 information.
        - Repeat the tool use for subsequent requests if necessary.
        - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
        - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
          emojis where appropriate.
        - Only respond to weather queries. Remind off-topic users of your
 purpose.
        - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
        - Complete the entire process until you have all required data before
 sending the complete response.
    "
    ;

    private static string default_prompt = "What is the weather like in
 Seattle?";

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int max_recursions = 5;

    public static async Task Main(string[] args)
```

```
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Error)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddHttpClient()
                    .AddSingleton<IAmazonBedrockRuntime>(_ => new
AmazonBedrockRuntimeClient(RegionEndpoint.USEast1)) // Specify a region that has
access to the chosen model.
                    .AddTransient<BedrockActionsWrapper>()
                    .AddTransient<WeatherTool>()
                    .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            await RunConversationAsync();

        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            Console.WriteLine(new string('-', 80));
        }
        finally
        {
            Console.WriteLine(
                "Amazon Bedrock Converse API with Tool Use Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
```

```
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _bedrockActionsWrapper =
host.Services.GetRequiredService<BedrockActionsWrapper>();
        _weatherTool = host.Services.GetRequiredService<WeatherTool>();
    }

    /// <summary>
    /// Starts the conversation with the user and handles the interaction with
Bedrock.
    /// </summary>
    /// <returns>The conversation array.</returns>
    public static async Task<List<Message>> RunConversationAsync()
    {
        // Print the greeting and a short user guide
        PrintHeader();

        // Start with an empty conversation
        var conversation = new List<Message>();

        // Get the first user input
        var userInput = await GetUserInputAsync();

        while (userInput != null)
        {
            // Create a new message with the user input and append it to the
conversation
            var message = new Message { Role = ConversationRole.User, Content =
new List<ContentBlock> { new ContentBlock { Text = userInput } } };
            conversation.Add(message);

            // Send the conversation to Amazon Bedrock
            var bedrockResponse = await SendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has
returned its final response or the recursion counter has reached 0
            await ProcessModelResponseAsync(bedrockResponse, conversation,
max_recursions);

            // Repeat the loop until the user decides to exit the application
            userInput = await GetUserInputAsync();
        }
```

```
        PrintFooter();
        return conversation;
    }

    /// <summary>
    /// Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.
    /// </summary>
    /// <param name="conversation">The conversation history including the next
message to send.</param>
    /// <returns>The response from Amazon Bedrock.</returns>
    private static async Task<ConverseResponse>
SendConversationToBedrock(List<Message> conversation)
    {
        Console.WriteLine("\tCalling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and
return the response
        return await _bedrockActionsWrapper.SendConverseRequestAsync(model_id,
system_prompt, conversation, _weatherTool.GetToolSpec());
    }

    /// <summary>
    /// Processes the response received via Amazon Bedrock and performs the
necessary actions based on the stop reason.
    /// </summary>
    /// <param name="modelResponse">The model's response returned via Amazon
Bedrock.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls
allowed.</param>
    private static async Task ProcessModelResponseAsync(ConverseResponse
modelResponse, List<Message> conversation, int maxRecursion)
    {
        if (maxRecursion <= 0)
        {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            Console.WriteLine("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.Add(modelResponse.Output.Message);
```

```
        if (modelResponse.StopReason == "tool_use")
        {
            // If the stop reason is "tool_use", forward everything to the tool
use handler
            await HandleToolUseAsync(modelResponse.Output, conversation,
maxRecursion - 1);
        }

        if (modelResponse.StopReason == "end_turn")
        {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process
            PrintModelResponse(modelResponse.Output.Message.Content[0].Text);
            if (!_interactive)
            {
                default_prompt = "x";
            }
        }
    }

    /// <summary>
    /// Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
    /// The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.
    /// </summary>
    /// <param name="modelResponse">The model's response containing the tool use
request.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls
allowed.</param>
    public static async Task HandleToolUseAsync(ConverseOutput modelResponse,
List<Message> conversation, int maxRecursion)
    {
        // Initialize an empty list of tool results
        var toolResults = new List<ContentBlock>();

        // The model's response can consist of multiple content blocks
        foreach (var contentBlock in modelResponse.Message.Content)
        {
            if (!String.IsNullOrEmpty(contentBlock.Text))
            {
                // If the content block contains text, print it to the console
```

```
                PrintModelResponse(contentBlock.Text);
            }

            if (contentBlock.ToolUse != null)
            {
                // If the content block is a tool use request, forward it to the
tool
                var toolResponse = await InvokeTool(contentBlock.ToolUse);

                // Add the tool use ID and the tool's response to the list of
results
                toolResults.Add(new ContentBlock
                {
                    ToolResult = new ToolResultBlock()
                    {
                        ToolUseId = toolResponse.ToolUseId,
                        Content = new List<ToolResultContentBlock>()
                            { new ToolResultContentBlock { Json =
toolResponse.Content } }
                    }
                });
            }
        }

        // Embed the tool results in a new user message
        var message = new Message() { Role = ConversationRole.User, Content =
toolResults };

        // Append the new message to the ongoing conversation
        conversation.Add(message);

        // Send the conversation to Amazon Bedrock
        var response = await SendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
        await ProcessModelResponseAsync(response, conversation, maxRecursion);
    }

    /// <summary>
    /// Invokes the specified tool with the given payload and returns the tool's
response.
    /// If the requested tool does not exist, an error message is returned.
    /// </summary>
```

```
    /// <param name="payload">The payload containing the tool name and input
data.</param>
    /// <returns>The tool's response or an error message.</returns>
    public static async Task<ToolResponse> InvokeTool(ToolUseBlock payload)
    {
        var toolName = payload.Name;

        if (toolName == "Weather_Tool")
        {
            var inputData = payload.Input.AsDictionary();
            PrintToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            var weatherResponse = await
_weatherTool.FetchWeatherDataAsync(inputData["latitude"].ToString(),
inputData["longitude"].ToString());
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
weatherResponse };
        }
        else
        {
            var errorMessage = $"\tThe requested tool with name '{toolName}' does
not exist.";
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
new { error = true, message = errorMessage } };
        }
    }


    /// <summary>
    /// Prompts the user for input and returns the user's response.
    /// Returns null if the user enters 'x' to exit.
    /// </summary>
    /// <param name="prompt">The prompt to display to the user.</param>
    /// <returns>The user's input or null if the user chooses to exit.</returns>
    private static async Task<string?> GetUserInputAsync(string prompt = "\tYour
weather info request:")
    {
        var userInput = default_prompt;
        if (_interactive)
        {
            Console.WriteLine(new string('*', 80));
            Console.WriteLine($"{prompt} (x to exit): \n\t");
            userInput = Console.ReadLine();
```

```
        }

        if (string.IsNullOrWhiteSpace(userInput))
        {
            prompt = "\tPlease enter your weather info request, e.g. the name of
a city";
            return await GetUserInputAsync(prompt);
        }

        if (userInput.ToLowerInvariant() == "x")
        {
            return null;
        }

        return userInput;
    }

    /// <summary>
    /// Logs the welcome message and usage guide for the tool use demo.
    /// </summary>
    public static void PrintHeader()
    {
        Console.WriteLine(@"
        ==================================================
        Welcome to the Amazon Bedrock Tool Use demo!
        ==================================================

        This assistant provides current weather information for user-specified
locations.
        You can ask for weather details by providing the location name or
coordinates. Weather information
        will be provided using a custom Tool and open-meteo API.

        Example queries:
        - What's the weather like in New York?
        - Current weather for latitude 40.70, longitude -74.01
        - Is it warmer in Rome or Barcelona today?

        To exit the program, simply type 'x' and press Enter.

        P.S.: You're not limited to single locations, or even to using English!
        Have fun and experiment with the app!
        ");
    }
```

```
    /// <summary>
    /// Logs the footer information for the tool use demo.
    /// </summary>
    public static void PrintFooter()
    {
        Console.WriteLine(@"
        ==================================================
        Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you
        learned something new, or got some inspiration for your own apps today!

        For more Bedrock examples in different programming languages, have a look
 at:
        https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
        ==================================================
        ");
    }

    /// <summary>
    /// Logs information about the tool use.
    /// </summary>
    /// <param name="toolName">The name of the tool being used.</param>
    /// <param name="inputData">The input data for the tool.</param>
    public static void PrintToolUse(string toolName, Dictionary<string, Document>
 inputData)
    {
        Console.WriteLine($"\n\tInvoking tool: {toolName} with input:
 {inputData["latitude"].ToString()}, {inputData["longitude"].ToString()}...\n");
    }

    /// <summary>
    /// Logs the model's response.
    /// </summary>
    /// <param name="message">The model's response message.</param>
    public static void PrintModelResponse(string message)
    {
        Console.WriteLine("\tThe model's response:\n");
        Console.WriteLine(message);
        Console.WriteLine();
    }
}
```

A ferramenta de meteorologia usada pela demonstração. Esse arquivo define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.Logging;

namespace ConverseToolScenario;

/// <summary>
/// Weather tool that will be invoked when requested by the Bedrock response.
/// </summary>
public class WeatherTool
{
    private readonly ILogger<WeatherTool> _logger;
    private readonly IHttpClientFactory _httpClientFactory;

    public WeatherTool(ILogger<WeatherTool> logger, IHttpClientFactory
 httpClientFactory)
    {
        _logger = logger;
        _httpClientFactory = httpClientFactory;
    }

    /// <summary>
    /// Returns the JSON Schema specification for the Weather tool. The tool
 specification
    /// defines the input schema and describes the tool's functionality.
    /// For more information, see https://json-schema.org/understanding-json-
schema/reference.
    /// </summary>
    /// <returns>The tool specification for the Weather tool.</returns>
    public ToolSpecification GetToolSpec()
    {
        ToolSpecification toolSpecification = new ToolSpecification();

        toolSpecification.Name = "Weather_Tool";
        toolSpecification.Description = "Get the current weather for a given
 location, based on its WGS84 coordinates.";

        Document toolSpecDocument = Document.FromObject(
```

```
            new
            {
                type = "object",
                properties = new
                {
                    latitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 latitude of the
 location."
                    },
                    longitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 longitude of the
 location."
                    }
                },
                required = new[] { "latitude", "longitude" }
            });

        toolSpecification.InputSchema = new ToolInputSchema() { Json =
 toolSpecDocument };
        return toolSpecification;
    }

    /// <summary>
    /// Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    /// Returns the weather data or an error message if the request fails.
    /// </summary>
    /// <param name="latitude">The latitude of the location.</param>
    /// <param name="longitude">The longitude of the location.</param>
    /// <returns>The weather data or an error message.</returns>
    public async Task<Document> FetchWeatherDataAsync(string latitude, string
 longitude)
    {
        string endpoint = "https://api.open-meteo.com/v1/forecast";

        try
        {
            var httpClient = _httpClientFactory.CreateClient();
            var response = await httpClient.GetAsync($"{endpoint}?
latitude={latitude}&longitude={longitude}&current_weather=True");
```

```
            response.EnsureSuccessStatusCode();
            var weatherData = await response.Content.ReadAsStringAsync();

            Document weatherDocument = Document.FromObject(
                new { weather_data = weatherData });

            return weatherDocument;
        }
        catch (HttpRequestException e)
        {
            _logger.LogError(e, "Error fetching weather data: {Message}",
 e.Message);
            throw;
        }
        catch (Exception e)
        {
            _logger.LogError(e, "Unexpected error fetching weather data:
 {Message}", e.Message);
            throw;
        }
    }
}
```

A ação da API Converse com uma configuração de ferramenta.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/>
 class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
```

```
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);
```

```
            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse
 request.");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");
   private static String modelId = "amazon.nova-lite-v1:0";
   private static String defaultPrompt = "What is the weather like in Seattle?";
   private static WeatherTool weatherTool = new WeatherTool();

   // The maximum number of recursive calls allowed in the tool use function.
   // This helps prevent infinite loops and potential performance issues.
   private static int maxRecursions = 5;
   static BedrockActions bedrockActions = new BedrockActions();
   public static boolean interactive = true;

   private static final String systemPrompt = """
           You are a weather assistant that provides current weather data for
user-specified locations using only
           the Weather_Tool, which expects latitude and longitude. Infer the
coordinates from the location yourself.
           If the user provides coordinates, infer the approximate location and
refer to it in your response.
           To use the tool, you strictly apply the provided tool specification.

           - Explain your step-by-step process, and give brief updates before
each step.
           - Only use the Weather_Tool for data. Never guess or make up
information.
           - Repeat the tool use for subsequent requests if necessary.
           - If the tool errors, apologize, explain weather is unavailable, and
suggest other options.
           - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
reports concise. Sparingly use
             emojis where appropriate.
           - Only respond to weather queries. Remind off-topic users of your
purpose.
           - Never claim to search online, access external data, or use tools
besides Weather_Tool.
           - Complete the entire process until you have all required data before
sending the complete response.
           """;

   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       System.out.println("""
               ==================================================
               Welcome to the Amazon Bedrock Tool Use demo!
```

```
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
 ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
 Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
 Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
```

```
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has
returned its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
```

```
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool
use handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion -
1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with
the tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to
handle the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
```

```
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new
ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);
```

```
        // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + "
with input: " + inputData.get("latitude").toString() + ", " +
inputData.get("longitude").toString() + "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }
```

```
   private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
       System.out.println("Calling Bedrock...");

       try {
           return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
       } catch (ModelNotReadyException ex) {
            System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
           throw ex;
       } catch (BedrockRuntimeException ex) {
           System.err.println("Bedrock service error: " + ex.getMessage());
           throw ex;
       } catch (RuntimeException ex) {
           System.err.println("Unexpected error occurred: " + ex.getMessage());
           throw ex;
       }
   }

   private static ConverseResponse
sendConversationToBedrockwithSpec(List<Message> conversation, ToolSpecification
toolSpec) {
       System.out.println("Calling Bedrock...");

       // Send the conversation, system prompt, and tool configuration, and
return the response
       return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
   }

   public static String getUserInput(String prompt, Scanner scanner) {
       String userInput = defaultPrompt;
       if (interactive) {
           System.out.println("*".repeat(80));
           System.out.println(prompt + " (x to exit): \n\t");
           userInput = scanner.nextLine();
       }

       if (userInput == null || userInput.trim().isEmpty()) {
           return getUserInput("\tPlease enter your weather info request, e.g.,
the name of a city", scanner);
       }
```

```
        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We
 hope you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages,
 have a look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

A ferramenta de meteorologia usada pela demonstração. Esse arquivo define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
public class WeatherTool {

    private static final Logger logger =
 LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-
schema/reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
 longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
```

```
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on
 its WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
 string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
```

```
            HttpResponse<String> response = httpClient.send(request,
HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson,
new TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
```

```
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

A ação da API Converse com uma configuração de ferramenta.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for
 the converse request
     * @param systemPrompt the system prompt to be included in the converse
 request
     * @param conversation a list of messages representing the conversation
 history
     * @param toolSpec     the specification of the tool to be used in the
 converse request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();

        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
```

```
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
 ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: "
 + ex.getMessage(), ex);
        }
    }
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
/* Before running this JavaScript code example, set up your development
 environment, including your credentials.
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
```

```
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";
import data from "./questions.json" with { type: "json" };
import toolConfig from "./tool_config.json" with { type: "json" };

const __filename = fileURLToPath(import.meta.url);

const systemPrompt = [
  {
    text:
      "You are a weather assistant that provides current weather data for user-
specified locations using only\n" +
      "the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.\n" +
      "If the user provides coordinates, infer the approximate location and refer
 to it in your response.\n" +
      "To use the tool, you strictly apply the provided tool specification.\n" +
      "If the user specifies a state, country, or region, infer the locations of
 cities within that state.\n" +
      "\n" +
      "- Explain your step-by-step process, and give brief updates before each
 step.\n" +
      "- Only use the Weather_Tool for data. Never guess or make up information.
 \n" +
      "- Repeat the tool use for subsequent requests if necessary.\n" +
      "- If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.\n" +
      "- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use\n" +
      "  emojis where appropriate.\n" +
```

```
      "- Only respond to weather queries. Remind off-topic users of your purpose.
 \n" +
      "- Never claim to search online, access external data, or use tools besides
 Weather_Tool.\n" +
      "- Complete the entire process until you have all required data before
 sending the complete response.",
  },
];
const tools_config = toolConfig;

/// Starts the conversation with the user and handles the interaction with
 Bedrock.
async function askQuestion(userMessage) {
  // The maximum number of recursive calls allowed in the tool use function.
  // This helps prevent infinite loops and potential performance issues.
  const max_recursions = 5;
  const messages = [
    {
      role: "user",
      content: [{ text: userMessage }],
    },
  ];
  try {
    const response = await SendConversationtoBedrock(messages);
    await ProcessModelResponseAsync(response, messages, max_recursions);
  } catch (error) {
    console.log("error ", error);
  }
}

// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
// param "messages" - The conversation history including the next message to
 send.
// return - The response from Amazon Bedrock.
async function SendConversationtoBedrock(messages) {
  const bedRockRuntimeClient = new BedrockRuntimeClient({
    region: "us-east-1",
  });
  try {
    const modelId = "amazon.nova-lite-v1:0";
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
```

```
        messages: messages,
        system: systemPrompt,
        toolConfig: tools_config,
      }),
    );
    return response;
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        "`${caught.name}` - Model not ready, please wait and try again.",
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        '`${caught.name}` - "Error occurred while sending Converse request.',
      );
      throw caught;
    }
  }
}

// Processes the response received via Amazon Bedrock and performs the necessary
 actions based on the stop reason.
// param "response" - The model's response returned via Amazon Bedrock.
// param "messages" - The conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function ProcessModelResponseAsync(response, messages, max_recursions) {
  if (max_recursions <= 0) {
    await HandleToolUseAsync(response, messages);
  }
  if (response.stopReason === "tool_use") {
    await HandleToolUseAsync(response, messages, max_recursions - 1);
  }
  if (response.stopReason === "end_turn") {
    const messageToPrint = response.output.message.content[0].text;
    console.log(messageToPrint.replace(/<[^>]+>/g, ""));
  }
}
// Handles the tool use case by invoking the specified tool and sending the
 tool's response back to Bedrock.
// The tool response is appended to the conversation, and the conversation is
 sent back to Amazon Bedrock for further processing.
// param "response" - the model's response containing the tool use request.
```

```
// param "messages" - the conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function HandleToolUseAsync(response, messages, max_recursions) {
  const toolResultFinal = [];
  try {
    const output_message = response.output.message;
    messages.push(output_message);
    const toolRequests = output_message.content;
    const toolMessage = toolRequests[0].text;
    console.log(toolMessage.replace(/<[^>]+>/g, ""));
    for (const toolRequest of toolRequests) {
      if (Object.hasOwn(toolRequest, "toolUse")) {
        const toolUse = toolRequest.toolUse;
        const latitude = toolUse.input.latitude;
        const longitude = toolUse.input.longitude;
        const toolUseID = toolUse.toolUseId;
        console.log(
          `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
        );
        if (toolUse.name === "Weather_Tool") {
          try {
            const current_weather = await callWeatherTool(
              longitude,
              latitude,
            ).then((current_weather) => current_weather);
            const currentWeather = current_weather;
            const toolResult = {
              toolResult: {
                toolUseId: toolUseID,
                content: [{ json: currentWeather }],
              },
            };
            toolResultFinal.push(toolResult);
          } catch (err) {
            console.log("An error occurred. ", err);
          }
        }
      }
    }

    const toolResultMessage = {
      role: "user",
      content: toolResultFinal,
    };
```

```
    messages.push(toolResultMessage);
    // Send the conversation to Amazon Bedrock
    await ProcessModelResponseAsync(
      await SendConversationtoBedrock(messages),
      messages,
    );
  } catch (error) {
    console.log("An error occurred. ", error);
  }
}
// Call the Weathertool.
// param = longitude of location
// param = latitude of location
async function callWeatherTool(longitude, latitude) {
  // Open-Meteo API endpoint
  const apiUrl = `https://api.open-meteo.com/v1/forecast?latitude=
${latitude}&longitude=${longitude}&current_weather=true`;

  // Fetch the weather data.
  return fetch(apiUrl)
    .then((response) => {
      return response.json().then((current_weather) => {
        return current_weather;
      });
    })
    .catch((error) => {
      console.error("Error fetching weather data:", error);
    });
}
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  default: "",
});

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Bedrock Tool Use demo! \n" +
    "This assistant provides current weather information for user-specified
 locations. " +
```

```
    "You can ask for weather details by providing the location name or
 coordinates." +
    "Weather information will be provided using a custom Tool and open-meteo
 API." +
    "For the purposes of this example, we'll use in order the questions in ./
questions.json :\n" +
    "What's the weather like in Seattle? " +
    "What's the best kind of cat? " +
    "Where is the warmest city in Washington State right now? " +
    "What's the warmest city in California right now?\n" +
    "To exit the program, simply type 'x' and press Enter.\n" +
    "Have fun and experiment with the app by editing the questions in ./
questions.json! " +
    "P.S.: You're not limited to single locations, or even to using English! ",

  { header: true },
);
const displayAskQuestion1 = new ScenarioOutput(
  "displayAskQuestion1",
  "Press enter to ask question number 1 (default is 'What's the weather like in
 Seattle?')",
);

const askQuestion1 = new ScenarioAction(
  "askQuestion1",
  async (/** @type {State} */ state) => {
    const userMessage1 = data.questions["question-1"];
    await askQuestion(userMessage1);
  },
);

const displayAskQuestion2 = new ScenarioOutput(
  "displayAskQuestion2",
  "Press enter to ask question number 2 (default is 'What's the best kind of
 cat?')",
);

const askQuestion2 = new ScenarioAction(
  "askQuestion2",
  async (/** @type {State} */ state) => {
    const userMessage2 = data.questions["question-2"];
    await askQuestion(userMessage2);
  },
);
```

```
const displayAskQuestion3 = new ScenarioOutput(
  "displayAskQuestion3",
  "Press enter to ask question number 3 (default is 'Where is the warmest city in
 Washington State right now?')",
);

const askQuestion3 = new ScenarioAction(
  "askQuestion3",
  async (/** @type {State} */ state) => {
    const userMessage3 = data.questions["question-3"];
    await askQuestion(userMessage3);
  },
);

const displayAskQuestion4 = new ScenarioOutput(
  "displayAskQuestion4",
  "Press enter to ask question number 4 (default is 'What's the warmest city in
 California right now?')",
);

const askQuestion4 = new ScenarioAction(
  "askQuestion4",
  async (/** @type {State} */ state) => {
    const userMessage4 = data.questions["question-4"];
    await askQuestion(userMessage4);
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you\n" +
    "learned something new, or got some inspiration for your own apps today!\n" +
    "For more Bedrock examples in different programming languages, have a look
 at:\n" +
    "https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html",
);

const myScenario = new Scenario("Converse Tool Scenario", [
  greet,
  pressEnter,
  displayAskQuestion1,
  askQuestion1,
  pressEnter,
```

```
  displayAskQuestion2,
  askQuestion2,
  pressEnter,
  displayAskQuestion3,
  askQuestion3,
  pressEnter,
  displayAskQuestion4,
  askQuestion4,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O script de execução principal da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

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

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-
inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
```

```
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
```

```
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has
returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()
```

```
    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message
to send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and
return the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try
again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
```

```
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool
use handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text,
and finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use
request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the
tool
                tool_response = self._invoke_tool(content_block["toolUse"])
```

```
                # Add the tool use ID and the tool's response to the list of
results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
```

```
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
import requests
from requests.exceptions import RequestException
```

```
def get_tool_spec():
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based
 on its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
```

```
    :return: The weather data or an error message.
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather":
 True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O cenário principal e a lógica da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
#[derive(Debug)]
#[allow(dead_code)]
struct InvokeToolResult(String, ToolResultBlock);
struct ToolUseScenario {
    client: Client,
    conversation: Vec<Message>,
```

```
    system_prompt: SystemContentBlock,
    tool_config: ToolConfiguration,
}

impl ToolUseScenario {
    fn new(client: Client) -> Self {
        let system_prompt = SystemContentBlock::Text(SYSTEM_PROMPT.into());
        let tool_config = ToolConfiguration::builder()
            .tools(Tool::ToolSpec(
                ToolSpecification::builder()
                    .name(TOOL_NAME)
                    .description(TOOL_DESCRIPTION)
                    .input_schema(ToolInputSchema::Json(make_tool_schema()))
                    .build()
                    .unwrap(),
            ))
            .build()
            .unwrap();

        ToolUseScenario {
            client,
            conversation: vec![],
            system_prompt,
            tool_config,
        }
    }

    async fn run(&mut self) -> Result<(), ToolUseScenarioError> {
        loop {
            let input = get_input().await?;
            if input.is_none() {
                break;
            }

            let message = Message::builder()
                .role(User)
                .content(ContentBlock::Text(input.unwrap()))
                .build()
                .map_err(ToolUseScenarioError::from)?;
            self.conversation.push(message);

            let response = self.send_to_bedrock().await?;

            self.process_model_response(response).await?;
```

```
        }

        Ok(())
    }

    async fn send_to_bedrock(&mut self) -> Result<ConverseOutput,
ToolUseScenarioError> {
        debug!("Sending conversation to bedrock");
        self.client
            .converse()
            .model_id(MODEL_ID)
            .set_messages(Some(self.conversation.clone()))
            .system(self.system_prompt.clone())
            .tool_config(self.tool_config.clone())
            .send()
            .await
            .map_err(ToolUseScenarioError::from)
    }

    async fn process_model_response(
        &mut self,
        mut response: ConverseOutput,
    ) -> Result<(), ToolUseScenarioError> {
        let mut iteration = 0;

        while iteration < MAX_RECURSIONS {
            iteration += 1;
            let message = if let Some(ref output) = response.output {
                if output.is_message() {
                    Ok(output.as_message().unwrap().clone())
                } else {
                    Err(ToolUseScenarioError(
                        "Converse Output is not a message".into(),
                    ))
                }
            } else {
                Err(ToolUseScenarioError("Missing Converse Output".into()))
            }?;

            self.conversation.push(message.clone());

            match response.stop_reason {
                StopReason::ToolUse => {
                    response = self.handle_tool_use(&message).await?;
```

```
                }
                StopReason::EndTurn => {
                    print_model_response(&message.content[0])?;
                    return Ok(());
                }
                _ => (),
            }
        }

        Err(ToolUseScenarioError(
            "Exceeded MAX_ITERATIONS when calling tools".into(),
        ))
    }

    async fn handle_tool_use(
        &mut self,
        message: &Message,
    ) -> Result<ConverseOutput, ToolUseScenarioError> {
        let mut tool_results: Vec<ContentBlock> = vec![];

        for block in &message.content {
            match block {
                ContentBlock::Text(_) => print_model_response(block)?,
                ContentBlock::ToolUse(tool) => {
                    let tool_response = self.invoke_tool(tool).await?;
                    tool_results.push(ContentBlock::ToolResult(tool_response.1));
                }
                _ => (),
            };
        }

        let message = Message::builder()
            .role(User)
            .set_content(Some(tool_results))
            .build()?;
        self.conversation.push(message);

        self.send_to_bedrock().await
    }

    async fn invoke_tool(
        &mut self,
        tool: &ToolUseBlock,
    ) -> Result<InvokeToolResult, ToolUseScenarioError> {
```

```
        match tool.name() {
            TOOL_NAME => {
                println!(
                    "\x1b[0;90mExecuting tool: {TOOL_NAME} with input: {:?}...
\x1b[0m",
                    tool.input()
                );
                let content = fetch_weather_data(tool).await?;
                println!(
                    "\x1b[0;90mTool responded with {:?}\x1b[0m",
                    content.content()
                );
                Ok(InvokeToolResult(tool.tool_use_id.clone(), content))
            }
            _ => Err(ToolUseScenarioError(format!(
                "The requested tool with name {} does not exist",
                tool.name()
            ))),
        }
    }
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let mut scenario = ToolUseScenario::new(client);

    header();
    if let Err(err) = scenario.run().await {
        println!("There was an error running the scenario! {}", err.0)
    }
    footer();
}
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
const ENDPOINT: &str = "https://api.open-meteo.com/v1/forecast";
async fn fetch_weather_data(
    tool_use: &ToolUseBlock,
) -> Result<ToolResultBlock, ToolUseScenarioError> {
    let input = tool_use.input();
    let latitude = input
        .as_object()
        .unwrap()
        .get("latitude")
        .unwrap()
        .as_string()
        .unwrap();
    let longitude = input
        .as_object()
        .unwrap()
        .get("longitude")
        .unwrap()
        .as_string()
        .unwrap();
    let params = [
        ("latitude", latitude),
        ("longitude", longitude),
        ("current_weather", "true"),
    ];

    debug!("Calling {ENDPOINT} with {params:?}");

    let response = reqwest::Client::new()
        .get(ENDPOINT)
        .query(&params)
        .send()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error requesting weather:
 {e:?}")))?
        .error_for_status()
        .map_err(|e| ToolUseScenarioError(format!("Failed to request weather:
 {e:?}")))?;

    debug!("Response: {response:?}");
```

```
    let bytes = response
        .bytes()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error reading response:
 {e:?}")))?;

    let result = String::from_utf8(bytes.to_vec())
        .map_err(|_| ToolUseScenarioError("Response was not utf8".into()))?;

    Ok(ToolResultBlock::builder()
        .tool_use_id(tool_use.tool_use_id())
        .content(ToolResultContentBlock::Text(result))
        .build()?)
}
```

Utilitários para imprimir os blocos de conteúdo da mensagem.

```
fn print_model_response(block: &ContentBlock) -> Result<(), ToolUseScenarioError>
 {
    if block.is_text() {
        let text = block.as_text().unwrap();
        println!("\x1b[0;90mThe model's response:\x1b[0m\n{text}");
        Ok(())
    } else {
        Err(ToolUseScenarioError(format!(
            "Content block is not text ({block:?})"
        )))
    }
}
```

Use instruções, utilitário de erro e constantes.

```
use std::{collections::HashMap, io::stdin};

use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::{BuildError, SdkError},
    operation::converse::{ConverseError, ConverseOutput},
    types::{
```

```
        ContentBlock, ConversationRole::User, Message, StopReason,
 SystemContentBlock, Tool,
        ToolConfiguration, ToolInputSchema, ToolResultBlock,
 ToolResultContentBlock,
        ToolSpecification, ToolUseBlock,
    },
    Client,
};
use aws_smithy_runtime_api::http::Response;
use aws_smithy_types::Document;
use tracing::debug;

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

const SYSTEM_PROMPT: &str = "You are a weather assistant that provides current
 weather data for user-specified locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
";

// The maximum number of recursive calls allowed in the tool_use_demo function.
// This helps prevent infinite loops and potential performance issues.
const MAX_RECURSIONS: i8 = 5;

const TOOL_NAME: &str = "Weather_Tool";
```

```
const TOOL_DESCRIPTION: &str =
    "Get the current weather for a given location, based on its WGS84
 coordinates.";
fn make_tool_schema() -> Document {
    Document::Object(HashMap::<String, Document>::from([
        ("type".into(), Document::String("object".into())),
        (
            "properties".into(),
            Document::Object(HashMap::from([
                (
                    "latitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String("Geographical WGS84 latitude of the
 location.".into()),
                        ),
                    ])),
                ),
                (
                    "longitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String(
                                "Geographical WGS84 longitude of the
 location.".into(),
                            ),
                        ),
                    ])),
                ),
            ])),
        ),
        (
            "required".into(),
            Document::Array(vec![
                Document::String("latitude".into()),
                Document::String("longitude".into()),
            ]),
        ),
    ]))
}
```

```
#[derive(Debug)]
struct ToolUseScenarioError(String);
impl std::fmt::Display for ToolUseScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Tool use error with '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl From<&str> for ToolUseScenarioError {
    fn from(value: &str) -> Self {
        ToolUseScenarioError(value.into())
    }
}
impl From<BuildError> for ToolUseScenarioError {
    fn from(value: BuildError) -> Self {
        ToolUseScenarioError(value.to_string().clone())
    }
}
impl From<SdkError<ConverseError, Response>> for ToolUseScenarioError {
    fn from(value: SdkError<ConverseError, Response>) -> Self {
        ToolUseScenarioError(match value.as_service_error() {
            Some(value) => value.meta().message().unwrap_or("Unknown").into(),
            None => "Unknown".into(),
        })
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Rust.

Amazon Nova para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar o Amazon Nova no Amazon Bedrock usando a API Converse do Bedrock
- Invocar o Amazon Nova no Amazon Bedrock usando a API Converse do Bedrock
- Enviar e processar um documento com o Amazon Nova no Amazon Bedrock
- Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada

Invocar o Amazon Nova no Amazon Bedrock usando a API Converse do Bedrock

Os exemplos de código a seguir mostram como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

.NET

SDK para .NET

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Amazon Nova.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Amazon Nova Lite.
var modelId = "amazon.nova-lite-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
```

```
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

Envie uma conversa de mensagens para o Amazon Nova usando a API Converse do Bedrock com uma configuração de ferramenta.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;
```

```
    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/>
 class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
```

```
                            ToolSpec = toolSpec
                        }
                    }
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse
 request.");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
func (wrapper ConverseWrapper) ConverseNova(ctx context.Context, prompt string)
 (string, error) {
```

```
 var content = types.ContentBlockMemberText{
  Value: prompt,
 }
 var message = types.Message{
  Content: []types.ContentBlock{&content},
  Role:    "user",
 }
 modelId := "amazon.nova-lite-v1:0"
 var converseInput = bedrockruntime.ConverseInput{
  ModelId:  aws.String(modelId),
  Messages: []types.Message{message},
 }
 response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
 if err != nil {
  ProcessError(err, modelId)
 }

 responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
 responseContentBlock := responseText.Value.Content[0]
 text, _ := responseContentBlock.(*types.ContentBlockMemberText)
 return text.Value, nil
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock com o cliente Java assíncrono.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with an asynchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class ConverseAsync {

    public static String converseAsync() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy,
 speed, and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";
```

```
        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from
the user
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response
length
                                .temperature(0.5F)  // Using temperature for
randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
temperature
                ).build();

        // Step 5: Send and process the request asynchronously
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            CompletableFuture<ConverseResponse> asyncResponse =
client.converse(request);
            return asyncResponse.thenApply(
                    response ->
response.output().message().content().get(0).text()
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
```

```
    }

    public static void main(String[] args) {
        String response = converseAsync();
        System.out.println(response);
    }
}
```

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with a synchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class Converse {

    public static String converse() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
```

```
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy,
 speed, and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from
 the user
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response
 length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request
        // - Send the request to the model
        // - Extract and return the generated text from the response
```

```
        try {
            ConverseResponse response = client.converse(request);
            return response.output().message().content().get(0).text();

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converse();
        System.out.println(response);
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
// This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Create a message
// - Configure and send a request
// - Process the response

import {
```

```
  BedrockRuntimeClient,
  ConversationRole,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client
// Credentials will be automatically loaded from the environment.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Step 2: Specify which model to use:
// Available Amazon Nova models and their characteristics:
// - Amazon Nova Micro: Text-only model optimized for lowest latency and cost
// - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video, and
 text
// - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed, and
 cost
//
// For the most current model IDs, see:
// https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
const modelId = "amazon.nova-lite-v1:0";

// Step 3: Create the message
// The message includes the text prompt and specifies that it comes from the user
const inputText =
  "Describe the purpose of a 'hello world' program in one line.";
const message = {
  content: [{ text: inputText }],
  role: ConversationRole.USER,
};

// Step 4: Configure the request
// Optional parameters to control the model's response:
// - maxTokens: maximum number of tokens to generate
// - temperature: randomness (max: 1.0, default: 0.7)
//   OR
// - topP: diversity of word choice (max: 1.0, default: 0.9)
// Note: Use either temperature OR topP, but not both
const request = {
  modelId,
  messages: [message],
  inferenceConfig: {
    maxTokens: 500, // The maximum response length
    temperature: 0.5, // Using temperature for randomness control
    //topP: 0.9,        // Alternative: use topP instead of temperature
```

```
  },
};

// Step 5: Send and process the request
// - Send the request to the model
// - Extract and return the generated text from the response
try {
  const response = await client.send(new ConverseCommand(request));
  console.log(response.output.message.content[0].text);
} catch (error) {
  console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
  throw error;
}
```

Envie uma conversa de mensagens para o Amazon Nova usando a API Converse do Bedrock com uma configuração de ferramenta.

```
// This example demonstrates how to send a conversation of messages to Amazon
 Nova using Bedrock's Converse API with a tool configuration.
// It shows how to:
// - 1. Set up the Amazon Bedrock runtime client
// - 2. Define the parameters required enable Amazon Bedrock to use a tool when
 formulating its response (model ID, user input, system prompt, and the tool
 spec)
// - 3. Send the request to Amazon Bedrock, and returns the response.
// - 4. Add the tool response to the conversation, and send it back to Amazon
 Bedrock.
// - 5. Publish the response.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client

// Credentials will be automatically loaded from the environment
const bedRockRuntimeClient = new BedrockRuntimeClient({
  region: "us-east-1",
});
```

```
// Step 2. Define the parameters required enable Amazon Bedrock to use a tool
 when formulating its response.

// The Bedrock Model ID.
const modelId = "amazon.nova-lite-v1:0";

// The system prompt to help Amazon Bedrock craft it's response.
const system_prompt = [
  {
    text:
      "You are a music expert that provides the most popular song played on a
 radio station, using only the\n" +
      "the top_song tool, which he call sign for the radio station for which you
 want the most popular song. " +
      "Example calls signs are WZPZ and WKRP. \n" +
      "- Only use the top_song tool. Never guess or make up information. \n" +
      "- If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.\n" +
      "- Only respond to queries about the most popular song played on a radio
 station\n" +
      "Remind off-topic users of your purpose. \n" +
      "- Never claim to search online, access external data, or use tools besides
 the top_song tool.\n",
  },
];
//  The user's question.
const message = [
  {
    role: "user",
    content: [{ text: "What is the most popular song on WZPZ?" }],
  },
];
// The tool specification. In this case, it uses an example schema for
// a tool that gets the most popular song played on a radio station.
const tool_config = {
  tools: [
    {
      toolSpec: {
        name: "top_song",
        description: "Get the most popular song played on a radio station.",
        inputSchema: {
          json: {
            type: "object",
            properties: {
```

```
              sign: {
                type: "string",
                description:
                  "The call sign for the radio station for which you want the
 most popular song. Example calls signs are WZPZ and WKRP.",
              },
            },
            required: ["sign"],
          },
        },
      },
    },
  ],
};

// Helper function to return the song and artist from top_song tool.
async function get_top_song(call_sign) {
  try {
    if (call_sign === "WZPZ") {
      const song = "Elemental Hotel";
      const artist = "8 Storey Hike";
      return { song, artist };
    }
  } catch (error) {
    console.log(`${error.message}`);
  }
}

// 3. Send the request to Amazon Bedrock, and returns the response.
export async function SendConversationtoBedrock(
  modelId,
  message,
  system_prompt,
  tool_config,
) {
  try {
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: message,
        system: system_prompt,
        toolConfig: tool_config,
      }),
    );
```

```
    if (response.stopReason === "tool_use") {
      const toolResultFinal = [];
      try {
        const output_message = response.output.message;
        message.push(output_message);
        const toolRequests = output_message.content;
        const toolMessage = toolRequests[0].text;
        console.log(toolMessage.replace(/<[^>]+>/g, ""));
        for (const toolRequest of toolRequests) {
          if (Object.hasOwn(toolRequest, "toolUse")) {
            const toolUse = toolRequest.toolUse;
            const sign = toolUse.input.sign;
            const toolUseID = toolUse.toolUseId;
            console.log(
              `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
            );
            if (toolUse.name === "top_song") {
              const toolResult = [];
              try {
                const top_song = await get_top_song(toolUse.input.sign).then(
                  (top_song) => top_song,
                );
                const toolResult = {
                  toolResult: {
                    toolUseId: toolUseID,
                    content: [
                      {
                        json: { song: top_song.song, artist: top_song.artist },
                      },
                    ],
                  },
                };
                toolResultFinal.push(toolResult);
              } catch (err) {
                const toolResult = {
                  toolUseId: toolUseID,
                  content: [{ json: { text: err.message } }],
                  status: "error",
                };
              }
            }
          }
        }
        const toolResultMessage = {
```

```
          role: "user",
          content: toolResultFinal,
        };
        // Step 4. Add the tool response to the conversation, and send it back to
 Amazon Bedrock.

        message.push(toolResultMessage);
        await SendConversationtoBedrock(
          modelId,
          message,
          system_prompt,
          tool_config,
        );
      } catch (caught) {
        console.error(`${caught.message}`);
        throw caught;
      }
    }

    // 4. Publish the response.
    if (response.stopReason === "end_turn") {
      const finalMessage = response.output.message.content[0].text;
      const messageToPrint = finalMessage.replace(/<[^>]+>/g);
      console.log(messageToPrint.replace(/<[^>]+>/g));
      return messageToPrint;
    }
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        `${caught.name} - Model not ready, please wait and try again.`,
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        `${caught.name} - Error occurred while sending Converse request`,
      );
      throw caught;
    }
  }
}
await SendConversationtoBedrock(modelId, message, system_prompt, tool_config);
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
suspend fun main() {
    converse().also { println(it) }
}

suspend fun converse(): String {
    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->
```

```
        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in one
 line."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Send the request and process the model's response
        runCatching {
            val response = client.converse(request)
            return response.output!!.asMessage().content.first().asText()
        }.getOrElse { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId",
 error)
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência da API do AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Amazon Nova.

use Aws\BedrockRuntime\BedrockRuntimeClient;
use Aws\Exception\AwsException;
use RuntimeException;

class Converse
{
    public function converse(): string
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        $client = new BedrockRuntimeClient([
            'region' => 'us-east-1',
            'profile' => 'default'
        ]);

        // Set the model ID, e.g., Amazon Nova Lite.
        $modelId = 'amazon.nova-lite-v1:0';

        // Start a conversation with the user message.
        $userMessage = "Describe the purpose of a 'hello world' program in one
 line.";
        $conversation = [
            [
                "role" => "user",
                "content" => [["text" => $userMessage]]
            ]
        ];

        try {
```

```
            // Send the message to the model, using a basic inference
 configuration.
            $response = $client->converse([
                'modelId' => $modelId,
                'messages' => $conversation,
                'inferenceConfig' => [
                    'maxTokens' => 512,
                    'temperature' => 0.5
                ]
            ]);

            // Extract and return the response text.
            $responseText = $response['output']['message']['content'][0]['text'];
            return $responseText;
        } catch (AwsException $e) {
            echo "ERROR: Can't invoke {$modelId}. Reason: {$e-
>getAwsErrorMessage()}";
            throw new RuntimeException("Failed to invoke model: " . $e-
>getAwsErrorMessage(), 0, $e);
        }
    }
}

$demo = new Converse();
echo $demo->converse();
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Amazon Nova.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Amazon Nova.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "amazon.nova-micro-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
```

```
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Swift.

Invocar o Amazon Nova no Amazon Bedrock usando a API Converse do Bedrock

Os exemplos de código a seguir mostram como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processar o fluxo de respostas em tempo real.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
// Use the Converse API to send a text message to Amazon Nova
```

```
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Amazon Nova Lite.
var modelId = "amazon.nova-lite-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);
```

```
    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;
```

```
/**
 * This example demonstrates how to use the Amazon Nova foundation models with an
 * asynchronous Amazon Bedrock runtime client to generate streaming text
 responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ConverseStream {

    public static void converseStream() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy,
 speed, and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from
 the user
        var inputText = "Describe the purpose of a 'hello world' program in one
 paragraph";
        var message = Message.builder()
```

```
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response
length
                                .temperature(0.5F)  // Using temperature for
randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
temperature
                ).build();

        // Step 5: Set up the stream handler
        // The stream handler processes chunks of the response as they arrive
        // - onContentBlockDelta: Processes each text chunk
        // - onError: Handles any errors during streaming
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            System.out.print(chunk.delta().text());
                            System.out.flush();  // Ensure immediate output of
each chunk
                        }).build())
                .onError(err -> System.err.printf("Can't invoke '%s': %s",
modelId, err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
```

```
        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }

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

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
// This example demonstrates how to use the Amazon Nova foundation models
// to generate streaming text responses.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Create a message
// - Configure a streaming request
// - Process the streaming response

import {
  BedrockRuntimeClient,
  ConversationRole,
```

```
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client
// Credentials will be automatically loaded from the environment
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Step 2: Specify which model to use
// Available Amazon Nova models and their characteristics:
// - Amazon Nova Micro: Text-only model optimized for lowest latency and cost
// - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video, and
 text
// - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed, and
 cost
//
// For the most current model IDs, see:
// https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
const modelId = "amazon.nova-lite-v1:0";

// Step 3: Create the message
// The message includes the text prompt and specifies that it comes from the user
const inputText =
  "Describe the purpose of a 'hello world' program in one paragraph";
const message = {
  content: [{ text: inputText }],
  role: ConversationRole.USER,
};

// Step 4: Configure the streaming request
// Optional parameters to control the model's response:
// - maxTokens: maximum number of tokens to generate
// - temperature: randomness (max: 1.0, default: 0.7)
//   OR
// - topP: diversity of word choice (max: 1.0, default: 0.9)
// Note: Use either temperature OR topP, but not both
const request = {
  modelId,
  messages: [message],
  inferenceConfig: {
    maxTokens: 500, // The maximum response length
    temperature: 0.5, // Using temperature for randomness control
    //topP: 0.9,        // Alternative: use topP instead of temperature
  },
};
```

```
// Step 5: Send and process the streaming request
// - Send the request to the model
// - Process each chunk of the streaming response
try {
  const response = await client.send(new ConverseStreamCommand(request));

  for await (const chunk of response.stream) {
    if (chunk.contentBlockDelta) {
      // Print each text chunk as it arrives
      process.stdout.write(chunk.contentBlockDelta.delta?.text || "");
    }
  }
} catch (error) {
  console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
  process.exitCode = 1;
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamOutput
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamRequest
```

```
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message with a prompt
 * - Configure a streaming request with parameters
 * - Process the response stream in real time
 */
suspend fun main() {
    converseStream()
}

suspend fun converseStream(): String {
    // A buffer to collect the complete response
    val completeResponseBuffer = StringBuilder()

    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in a
 paragraph."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseStreamRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
```

```
        }

        // Process the streaming response
        runCatching {
            client.converseStream(request) { response ->
                response.stream?.collect { chunk ->
                    when (chunk) {
                        is ConverseStreamOutput.ContentBlockDelta -> {
                            // Process each text chunk as it arrives
                            chunk.value.delta?.asText()?.let { text ->
                                print(text)
                                System.out.flush() // Ensure immediate output
                                completeResponseBuffer.append(text)
                            }
                        }
                        else -> {} // Other output block types can be handled as
 needed
                    }
                }
            }
        }.onFailure { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId:
 $error", error)
        }
    }

    return completeResponseBuffer.toString()
}
```

- Para obter detalhes da API, consulte a ConverseStreamreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
# Use the Conversation API to send a text message to Amazon Nova Text
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )
```

```
    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStreamReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
// An example demonstrating how to use the Conversation API to send a text
 message
// to Amazon Nova and print the response stream

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)
```

```
    // Set the model ID.
    let modelId = "amazon.nova-lite-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nNova Lite:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStreama referência da API AWS SDK for Swift.

Enviar e processar um documento com o Amazon Nova no Amazon Bedrock

O exemplo de código a seguir mostra como enviar e processar um documento com o Amazon Nova no Amazon Bedrock.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com o Amazon Nova no Amazon Bedrock.

```
# Send and process a document with Amazon Nova on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()
```

```
# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada

Os exemplos de código a seguir mostram como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;

namespace ConverseToolScenario;

public static class ConverseToolScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
```

```
    This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
    The script interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
    input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve
 current weather data for a given location.
   */

    public static BedrockActionsWrapper _bedrockActionsWrapper = null!;
    public static WeatherTool _weatherTool = null!;
    public static bool _interactive = true;

    // Change this string to use a different model with Converse API.
    private static string model_id = "amazon.nova-lite-v1:0";

    private static string system_prompt = @"
        You are a weather assistant that provides current weather data for user-
specified locations using only
        the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
        If the user specifies a state, country, or region, infer the locations of
 cities within that state.
        If the user provides coordinates, infer the approximate location and
 refer to it in your response.
        To use the tool, you strictly apply the provided tool specification.

        - Explain your step-by-step process, and give brief updates before each
 step.
        - Only use the Weather_Tool for data. Never guess or make up
 information.
        - Repeat the tool use for subsequent requests if necessary.
        - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
        - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
          emojis where appropriate.
        - Only respond to weather queries. Remind off-topic users of your
 purpose.
        - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
        - Complete the entire process until you have all required data before
 sending the complete response.
    "
```

```
    ;

    private static string default_prompt = "What is the weather like in
Seattle?";

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int max_recursions = 5;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Error)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddHttpClient()
                    .AddSingleton<IAmazonBedrockRuntime>(_ => new
AmazonBedrockRuntimeClient(RegionEndpoint.USEast1)) // Specify a region that has
access to the chosen model.
                    .AddTransient<BedrockActionsWrapper>()
                    .AddTransient<WeatherTool>()
                    .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            await RunConversationAsync();

        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            Console.WriteLine(new string('-', 80));
        }
        finally
        {
```

```
            Console.WriteLine(
                "Amazon Bedrock Converse API with Tool Use Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _bedrockActionsWrapper =
host.Services.GetRequiredService<BedrockActionsWrapper>();
        _weatherTool = host.Services.GetRequiredService<WeatherTool>();
    }

    /// <summary>
    /// Starts the conversation with the user and handles the interaction with
Bedrock.
    /// </summary>
    /// <returns>The conversation array.</returns>
    public static async Task<List<Message>> RunConversationAsync()
    {
        // Print the greeting and a short user guide
        PrintHeader();

        // Start with an empty conversation
        var conversation = new List<Message>();

        // Get the first user input
        var userInput = await GetUserInputAsync();

        while (userInput != null)
        {
            // Create a new message with the user input and append it to the
conversation
            var message = new Message { Role = ConversationRole.User, Content =
new List<ContentBlock> { new ContentBlock { Text = userInput } } };
            conversation.Add(message);

            // Send the conversation to Amazon Bedrock
            var bedrockResponse = await SendConversationToBedrock(conversation);
```

```
            // Recursively handle the model's response until the model has
returned its final response or the recursion counter has reached 0
            await ProcessModelResponseAsync(bedrockResponse, conversation,
max_recursions);

            // Repeat the loop until the user decides to exit the application
            userInput = await GetUserInputAsync();
        }

        PrintFooter();
        return conversation;
    }

    /// <summary>
    /// Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.
    /// </summary>
    /// <param name="conversation">The conversation history including the next
message to send.</param>
    /// <returns>The response from Amazon Bedrock.</returns>
    private static async Task<ConverseResponse>
SendConversationToBedrock(List<Message> conversation)
    {
        Console.WriteLine("\tCalling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and
return the response
        return await _bedrockActionsWrapper.SendConverseRequestAsync(model_id,
system_prompt, conversation, _weatherTool.GetToolSpec());
    }

    /// <summary>
    /// Processes the response received via Amazon Bedrock and performs the
necessary actions based on the stop reason.
    /// </summary>
    /// <param name="modelResponse">The model's response returned via Amazon
Bedrock.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls
allowed.</param>
    private static async Task ProcessModelResponseAsync(ConverseResponse
modelResponse, List<Message> conversation, int maxRecursion)
    {
```

```
        if (maxRecursion <= 0)
        {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            Console.WriteLine("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.Add(modelResponse.Output.Message);

        if (modelResponse.StopReason == "tool_use")
        {
            // If the stop reason is "tool_use", forward everything to the tool
use handler
            await HandleToolUseAsync(modelResponse.Output, conversation,
maxRecursion - 1);
        }

        if (modelResponse.StopReason == "end_turn")
        {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process
            PrintModelResponse(modelResponse.Output.Message.Content[0].Text);
            if (!_interactive)
            {
                default_prompt = "x";
            }
        }
    }

    /// <summary>
    /// Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
    /// The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.
    /// </summary>
    /// <param name="modelResponse">The model's response containing the tool use
request.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls
allowed.</param>
    public static async Task HandleToolUseAsync(ConverseOutput modelResponse,
List<Message> conversation, int maxRecursion)
```

```
    {
        // Initialize an empty list of tool results
        var toolResults = new List<ContentBlock>();

        // The model's response can consist of multiple content blocks
        foreach (var contentBlock in modelResponse.Message.Content)
        {
            if (!String.IsNullOrEmpty(contentBlock.Text))
            {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.Text);
            }

            if (contentBlock.ToolUse != null)
            {
                // If the content block is a tool use request, forward it to the
tool
                var toolResponse = await InvokeTool(contentBlock.ToolUse);

                // Add the tool use ID and the tool's response to the list of
results
                toolResults.Add(new ContentBlock
                {
                    ToolResult = new ToolResultBlock()
                    {
                        ToolUseId = toolResponse.ToolUseId,
                        Content = new List<ToolResultContentBlock>()
                            { new ToolResultContentBlock { Json =
toolResponse.Content } }
                    }
                });
            }
        }

        // Embed the tool results in a new user message
        var message = new Message() { Role = ConversationRole.User, Content =
toolResults };

        // Append the new message to the ongoing conversation
        conversation.Add(message);

        // Send the conversation to Amazon Bedrock
        var response = await SendConversationToBedrock(conversation);
```

```
        // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
        await ProcessModelResponseAsync(response, conversation, maxRecursion);
    }

    /// <summary>
    /// Invokes the specified tool with the given payload and returns the tool's
response.
    /// If the requested tool does not exist, an error message is returned.
    /// </summary>
    /// <param name="payload">The payload containing the tool name and input
data.</param>
    /// <returns>The tool's response or an error message.</returns>
    public static async Task<ToolResponse> InvokeTool(ToolUseBlock payload)
    {
        var toolName = payload.Name;

        if (toolName == "Weather_Tool")
        {
            var inputData = payload.Input.AsDictionary();
            PrintToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            var weatherResponse = await
_weatherTool.FetchWeatherDataAsync(inputData["latitude"].ToString(),
inputData["longitude"].ToString());
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
weatherResponse };
        }
        else
        {
            var errorMessage = $"\tThe requested tool with name '{toolName}' does
not exist.";
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
new { error = true, message = errorMessage } };
        }
    }


    /// <summary>
    /// Prompts the user for input and returns the user's response.
    /// Returns null if the user enters 'x' to exit.
    /// </summary>
    /// <param name="prompt">The prompt to display to the user.</param>
```

```
    /// <returns>The user's input or null if the user chooses to exit.</returns>
    private static async Task<string?> GetUserInputAsync(string prompt = "\tYour
weather info request:")
    {
        var userInput = default_prompt;
        if (_interactive)
        {
            Console.WriteLine(new string('*', 80));
            Console.WriteLine($"{prompt} (x to exit): \n\t");
            userInput = Console.ReadLine();
        }

        if (string.IsNullOrWhiteSpace(userInput))
        {
            prompt = "\tPlease enter your weather info request, e.g. the name of
a city";
            return await GetUserInputAsync(prompt);
        }

        if (userInput.ToLowerInvariant() == "x")
        {
            return null;
        }

        return userInput;
    }

    /// <summary>
    /// Logs the welcome message and usage guide for the tool use demo.
    /// </summary>
    public static void PrintHeader()
    {
        Console.WriteLine(@"
        ==================================================
        Welcome to the Amazon Bedrock Tool Use demo!
        ==================================================

        This assistant provides current weather information for user-specified
locations.
        You can ask for weather details by providing the location name or
coordinates. Weather information
        will be provided using a custom Tool and open-meteo API.

        Example queries:
```

```
        - What's the weather like in New York?
        - Current weather for latitude 40.70, longitude -74.01
        - Is it warmer in Rome or Barcelona today?

        To exit the program, simply type 'x' and press Enter.

        P.S.: You're not limited to single locations, or even to using English!
        Have fun and experiment with the app!
        ");
    }

    /// <summary>
    /// Logs the footer information for the tool use demo.
    /// </summary>
    public static void PrintFooter()
    {
        Console.WriteLine(@"
        ==================================================
        Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you
        learned something new, or got some inspiration for your own apps today!

        For more Bedrock examples in different programming languages, have a look
 at:
        https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
        ==================================================
        ");
    }

    /// <summary>
    /// Logs information about the tool use.
    /// </summary>
    /// <param name="toolName">The name of the tool being used.</param>
    /// <param name="inputData">The input data for the tool.</param>
    public static void PrintToolUse(string toolName, Dictionary<string, Document>
 inputData)
    {
        Console.WriteLine($"\n\tInvoking tool: {toolName} with input:
 {inputData["latitude"].ToString()}, {inputData["longitude"].ToString()}...\n");
    }

    /// <summary>
    /// Logs the model's response.
    /// </summary>
```

```
    /// <param name="message">The model's response message.</param>
    public static void PrintModelResponse(string message)
    {
        Console.WriteLine("\tThe model's response:\n");
        Console.WriteLine(message);
        Console.WriteLine();
    }
}
```

A ferramenta de meteorologia usada pela demonstração. Esse arquivo define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.Logging;

namespace ConverseToolScenario;

/// <summary>
/// Weather tool that will be invoked when requested by the Bedrock response.
/// </summary>
public class WeatherTool
{
    private readonly ILogger<WeatherTool> _logger;
    private readonly IHttpClientFactory _httpClientFactory;

    public WeatherTool(ILogger<WeatherTool> logger, IHttpClientFactory
 httpClientFactory)
    {
        _logger = logger;
        _httpClientFactory = httpClientFactory;
    }

    /// <summary>
    /// Returns the JSON Schema specification for the Weather tool. The tool
 specification
    /// defines the input schema and describes the tool's functionality.
    /// For more information, see https://json-schema.org/understanding-json-
schema/reference.
    /// </summary>
```

```
    /// <returns>The tool specification for the Weather tool.</returns>
    public ToolSpecification GetToolSpec()
    {
        ToolSpecification toolSpecification = new ToolSpecification();

        toolSpecification.Name = "Weather_Tool";
        toolSpecification.Description = "Get the current weather for a given
 location, based on its WGS84 coordinates.";

        Document toolSpecDocument = Document.FromObject(
            new
            {
                type = "object",
                properties = new
                {
                    latitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 latitude of the
 location."
                    },
                    longitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 longitude of the
 location."
                    }
                },
                required = new[] { "latitude", "longitude" }
            });

        toolSpecification.InputSchema = new ToolInputSchema() { Json =
 toolSpecDocument };
        return toolSpecification;
    }

    /// <summary>
    /// Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    /// Returns the weather data or an error message if the request fails.
    /// </summary>
    /// <param name="latitude">The latitude of the location.</param>
    /// <param name="longitude">The longitude of the location.</param>
    /// <returns>The weather data or an error message.</returns>
```

```
    public async Task<Document> FetchWeatherDataAsync(string latitude, string
 longitude)
    {
        string endpoint = "https://api.open-meteo.com/v1/forecast";

        try
        {
            var httpClient = _httpClientFactory.CreateClient();
            var response = await httpClient.GetAsync($"{endpoint}?
latitude={latitude}&longitude={longitude}&current_weather=True");
            response.EnsureSuccessStatusCode();
            var weatherData = await response.Content.ReadAsStringAsync();

            Document weatherDocument = Document.FromObject(
                new { weather_data = weatherData });

            return weatherDocument;
        }
        catch (HttpRequestException e)
        {
            _logger.LogError(e, "Error fetching weather data: {Message}",
 e.Message);
            throw;
        }
        catch (Exception e)
        {
            _logger.LogError(e, "Unexpected error fetching weather data:
 {Message}", e.Message);
            throw;
        }
    }
}
```

A ação da API Converse com uma configuração de ferramenta.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
```

```
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/>
 class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
```

```
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse
 request.");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for
 user-specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before
 each step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
```

```
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
 ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
 Scenario is complete.");
```

```
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has
returned its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }
```

```
    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool
use handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion -
1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
```

```
    * @param conversation  the current conversation, which will be updated with
the tool use results
    * @param maxRecursion  the maximum number of recursive calls allowed to
handle the model's response
    */
   private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
       List<ContentBlock> toolResults = new ArrayList<>();

       // The model's response can consist of multiple content blocks
       for (ContentBlock contentBlock : modelResponse.message().content()) {
           if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
               // If the content block contains text, print it to the console
               PrintModelResponse(contentBlock.text());
           }

           if (contentBlock.toolUse() != null) {
               ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

               // Add the tool use ID and the tool's response to the list of
results
               List<ToolResultContentBlock> contentBlockList = new
ArrayList<>();
               ToolResultContentBlock block = ToolResultContentBlock.builder()
                       .json(toolResponse.getContent())
                       .build();
               contentBlockList.add(block);

               ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                       .toolUseId(toolResponse.getToolUseId())
                       .content(contentBlockList)
                       .build();

               ContentBlock contentBlock1 = ContentBlock.builder()
                       .toolResult(toolResultBlock)
                       .build();

               toolResults.add(contentBlock1);
           }
       }

       // Embed the tool results in a new user message
       Message message = Message.builder()
               .role(ConversationRole.USER)
```

```
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
```

```
        System.out.println("Invoking tool: " + toolName + "
with input: " + inputData.get("latitude").toString() + ", " +
inputData.get("longitude").toString() + "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse
sendConversationToBedrockwithSpec(List<Message> conversation, ToolSpecification
toolSpec) {
        System.out.println("Calling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and
return the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
```

```
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g.,
the name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We
hope you
                learned something new, or got some inspiration for your own apps
today!
```

```
                For more Bedrock examples in different programming languages,
 have a look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

A ferramenta de meteorologia usada pela demonstração. Esse arquivo define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
public class WeatherTool {

    private static final Logger logger =
 LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-
schema/reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
 longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
```

```
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on
its WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");
```

```
        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson,
 new TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
```

```
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

A ação da API Converse com uma configuração de ferramenta.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for
 the converse request
     * @param systemPrompt the system prompt to be included in the converse
 request
     * @param conversation a list of messages representing the conversation
 history
     * @param toolSpec     the specification of the tool to be used in the
 converse request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();

        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
```

```
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: "
+ ex.getMessage(), ex);
        }
    }
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
/* Before running this JavaScript code example, set up your development
 environment, including your credentials.
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";
import data from "./questions.json" with { type: "json" };
import toolConfig from "./tool_config.json" with { type: "json" };

const __filename = fileURLToPath(import.meta.url);

const systemPrompt = [
  {
    text:
      "You are a weather assistant that provides current weather data for user-
specified locations using only\n" +
      "the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.\n" +
      "If the user provides coordinates, infer the approximate location and refer
 to it in your response.\n" +
      "To use the tool, you strictly apply the provided tool specification.\n" +
      "If the user specifies a state, country, or region, infer the locations of
 cities within that state.\n" +
      "\n" +
      "- Explain your step-by-step process, and give brief updates before each
 step.\n" +
```

```
      "- Only use the Weather_Tool for data. Never guess or make up information.
 \n" +
      "- Repeat the tool use for subsequent requests if necessary.\n" +
      "- If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.\n" +
      "- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use\n" +
      "  emojis where appropriate.\n" +
      "- Only respond to weather queries. Remind off-topic users of your purpose.
 \n" +
      "- Never claim to search online, access external data, or use tools besides
 Weather_Tool.\n" +
      "- Complete the entire process until you have all required data before
 sending the complete response.",
  },
];
const tools_config = toolConfig;

/// Starts the conversation with the user and handles the interaction with
 Bedrock.
async function askQuestion(userMessage) {
  // The maximum number of recursive calls allowed in the tool use function.
  // This helps prevent infinite loops and potential performance issues.
  const max_recursions = 5;
  const messages = [
    {
      role: "user",
      content: [{ text: userMessage }],
    },
  ];
  try {
    const response = await SendConversationtoBedrock(messages);
    await ProcessModelResponseAsync(response, messages, max_recursions);
  } catch (error) {
    console.log("error ", error);
  }
}

// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
// param "messages" - The conversation history including the next message to
 send.
// return - The response from Amazon Bedrock.
async function SendConversationtoBedrock(messages) {
```

```
  const bedRockRuntimeClient = new BedrockRuntimeClient({
    region: "us-east-1",
  });
  try {
    const modelId = "amazon.nova-lite-v1:0";
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: messages,
        system: systemPrompt,
        toolConfig: tools_config,
      }),
    );
    return response;
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        "`${caught.name}` - Model not ready, please wait and try again.",
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        '`${caught.name}` - "Error occurred while sending Converse request.',
      );
      throw caught;
    }
  }
}

// Processes the response received via Amazon Bedrock and performs the necessary
 actions based on the stop reason.
// param "response" - The model's response returned via Amazon Bedrock.
// param "messages" - The conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function ProcessModelResponseAsync(response, messages, max_recursions) {
  if (max_recursions <= 0) {
    await HandleToolUseAsync(response, messages);
  }
  if (response.stopReason === "tool_use") {
    await HandleToolUseAsync(response, messages, max_recursions - 1);
  }
  if (response.stopReason === "end_turn") {
    const messageToPrint = response.output.message.content[0].text;
```

```
    console.log(messageToPrint.replace(/<[^>]+>/g, ""));
  }
}
// Handles the tool use case by invoking the specified tool and sending the
 tool's response back to Bedrock.
// The tool response is appended to the conversation, and the conversation is
 sent back to Amazon Bedrock for further processing.
// param "response" - the model's response containing the tool use request.
// param "messages" - the conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function HandleToolUseAsync(response, messages, max_recursions) {
  const toolResultFinal = [];
  try {
    const output_message = response.output.message;
    messages.push(output_message);
    const toolRequests = output_message.content;
    const toolMessage = toolRequests[0].text;
    console.log(toolMessage.replace(/<[^>]+>/g, ""));
    for (const toolRequest of toolRequests) {
      if (Object.hasOwn(toolRequest, "toolUse")) {
        const toolUse = toolRequest.toolUse;
        const latitude = toolUse.input.latitude;
        const longitude = toolUse.input.longitude;
        const toolUseID = toolUse.toolUseId;
        console.log(
          `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
        );
        if (toolUse.name === "Weather_Tool") {
          try {
            const current_weather = await callWeatherTool(
              longitude,
              latitude,
            ).then((current_weather) => current_weather);
            const currentWeather = current_weather;
            const toolResult = {
              toolResult: {
                toolUseId: toolUseID,
                content: [{ json: currentWeather }],
              },
            };
            toolResultFinal.push(toolResult);
          } catch (err) {
            console.log("An error occurred. ", err);
          }
```

```
        }
      }
    }

    const toolResultMessage = {
      role: "user",
      content: toolResultFinal,
    };
    messages.push(toolResultMessage);
    // Send the conversation to Amazon Bedrock
    await ProcessModelResponseAsync(
      await SendConversationtoBedrock(messages),
      messages,
    );
  } catch (error) {
    console.log("An error occurred. ", error);
  }
}
// Call the Weathertool.
// param = longitude of location
// param = latitude of location
async function callWeatherTool(longitude, latitude) {
  // Open-Meteo API endpoint
  const apiUrl = `https://api.open-meteo.com/v1/forecast?latitude=
${latitude}&longitude=${longitude}&current_weather=true`;

  // Fetch the weather data.
  return fetch(apiUrl)
    .then((response) => {
      return response.json().then((current_weather) => {
        return current_weather;
      });
    })
    .catch((error) => {
      console.error("Error fetching weather data:", error);
    });
}
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  default: "",
```

```
});

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Bedrock Tool Use demo! \n" +
    "This assistant provides current weather information for user-specified
 locations. " +
    "You can ask for weather details by providing the location name or
 coordinates." +
    "Weather information will be provided using a custom Tool and open-meteo
 API." +
    "For the purposes of this example, we'll use in order the questions in ./
questions.json :\n" +
    "What's the weather like in Seattle? " +
    "What's the best kind of cat? " +
    "Where is the warmest city in Washington State right now? " +
    "What's the warmest city in California right now?\n" +
    "To exit the program, simply type 'x' and press Enter.\n" +
    "Have fun and experiment with the app by editing the questions in ./
questions.json! " +
    "P.S.: You're not limited to single locations, or even to using English! ",

  { header: true },
);
const displayAskQuestion1 = new ScenarioOutput(
  "displayAskQuestion1",
  "Press enter to ask question number 1 (default is 'What's the weather like in
 Seattle?')",
);

const askQuestion1 = new ScenarioAction(
  "askQuestion1",
  async (/** @type {State} */ state) => {
    const userMessage1 = data.questions["question-1"];
    await askQuestion(userMessage1);
  },
);

const displayAskQuestion2 = new ScenarioOutput(
  "displayAskQuestion2",
  "Press enter to ask question number 2 (default is 'What's the best kind of
 cat?')",
);
```

```
const askQuestion2 = new ScenarioAction(
  "askQuestion2",
  async (/** @type {State} */ state) => {
    const userMessage2 = data.questions["question-2"];
    await askQuestion(userMessage2);
  },
);
const displayAskQuestion3 = new ScenarioOutput(
  "displayAskQuestion3",
  "Press enter to ask question number 3 (default is 'Where is the warmest city in
 Washington State right now?')",
);

const askQuestion3 = new ScenarioAction(
  "askQuestion3",
  async (/** @type {State} */ state) => {
    const userMessage3 = data.questions["question-3"];
    await askQuestion(userMessage3);
  },
);

const displayAskQuestion4 = new ScenarioOutput(
  "displayAskQuestion4",
  "Press enter to ask question number 4 (default is 'What's the warmest city in
 California right now?')",
);

const askQuestion4 = new ScenarioAction(
  "askQuestion4",
  async (/** @type {State} */ state) => {
    const userMessage4 = data.questions["question-4"];
    await askQuestion(userMessage4);
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you\n" +
    "learned something new, or got some inspiration for your own apps today!\n" +
    "For more Bedrock examples in different programming languages, have a look
 at:\n" +
    "https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html",
);
```

```
const myScenario = new Scenario("Converse Tool Scenario", [
  greet,
  pressEnter,
  displayAskQuestion1,
  askQuestion1,
  pressEnter,
  displayAskQuestion2,
  askQuestion2,
  pressEnter,
  displayAskQuestion3,
  askQuestion3,
  pressEnter,
  displayAskQuestion4,
  askQuestion4,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

Amazon Nova Canvas para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem

Invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem

Os exemplos de código a seguir mostram como invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
// Use the native inference API to create an image with Amazon Nova Canvas.

using System;
using System.IO;
using System.Text.Json;
using System.Text.Json.Nodes;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID.
var modelId = "amazon.nova-canvas-v1:0";

// Define the image generation prompt for the model.
```

```
var prompt = "A stylized picture of a cute old steampunk robot.";

// Create a random seed between 0 and 858,993,459
int seed = new Random().Next(0, 858993460);

//Format the request payload using the model's native structure.
var nativeRequest = JsonSerializer.Serialize(new
{
    taskType = "TEXT_IMAGE",
    textToImageParams = new
    {
        text = prompt
    },
    imageGenerationConfig = new
    {
        seed,
        quality = "standard",
        width = 512,
        height = 512,
        numberOfImages = 1
    }
});

// Create a request with the model ID and the model's native request payload.
var request = new InvokeModelRequest()
{
    ModelId = modelId,
    Body = new MemoryStream(System.Text.Encoding.UTF8.GetBytes(nativeRequest)),
    ContentType = "application/json"
};

try
{
    // Send the request to the Bedrock Runtime and wait for the response.
    var response = await client.InvokeModelAsync(request);

    // Decode the response body.
    var modelResponse = await JsonNode.ParseAsync(response.Body);

    // Extract the image data.
    var base64Image = modelResponse["images"]?[0].ToString() ?? "";

    // Save the image in a local folder
    string savedPath = AmazonNovaCanvas.InvokeModel.SaveBase64Image(base64Image);
```

```
    Console.WriteLine($"Image saved to: {savedPath}");
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.InvokeModelResponse;

import java.security.SecureRandom;
import java.util.Base64;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
```

```
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 */
public class InvokeModel {

    public static byte[] invokeModel() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-canvas-v1:0";

        // Step 3: Configure the generation parameters and create the request
        // First, set the main parameters:
        // - prompt: Text description of the image to generate
        // - seed: Random number for reproducible generation (0 to 858,993,459)
        String prompt = "A stylized picture of a cute old steampunk robot";
        int seed = new SecureRandom().nextInt(858_993_460);

        // Then, create the request using a template with the following
 structure:
        // - taskType: TEXT_IMAGE (specifies text-to-image generation)
        // - textToImageParams: Contains the text prompt
        // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
        // For a list of available request parameters, see:
        // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
        String request = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": {
```

```
                        "text": "{{prompt}}"
                    },
                    "imageGenerationConfig": {
                        "seed": {{seed}},
                        "quality": "standard"
                    }
                }"""
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", String.valueOf(seed));

        // Step 4: Send and process the request
        // - Send the request to the model using InvokeModelResponse
        // - Extract the Base64-encoded image from the JSON response
        // - Convert the encoded image to a byte array and return it
        try {
            InvokeModelResponse response = client.invokeModel(builder -> builder
                    .modelId(modelId)
                    .body(SdkBytes.fromUtf8String(request))
            );

            JSONObject responseBody = new
 JSONObject(response.body().asUtf8String());
            // Convert the Base64 string to byte array for better handling
            return Base64.getDecoder().decode(
                    new JSONPointer("/
images/0").queryFrom(responseBody).toString()
            );

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s%n", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");
        byte[] imageData = invokeModel();
        displayImage(imageData);
    }
}
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
import {
  BedrockRuntimeClient,
  InvokeModelCommand,
} from "@aws-sdk/client-bedrock-runtime";
import { saveImage } from "../../utils/image-creation.js";
import { fileURLToPath } from "node:url";

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 *
 * @returns {Promise<string>} Base64-encoded image data
 */
export const invokeModel = async () => {
  // Step 1: Create the Amazon Bedrock runtime client
  // Credentials will be automatically loaded from the environment
  const client = new BedrockRuntimeClient({ region: "us-east-1" });

  // Step 2: Specify which model to use
  // For the latest available models, see:
  // https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
  const modelId = "amazon.nova-canvas-v1:0";
```

```
  // Step 3: Configure the request payload
  // First, set the main parameters:
  // - prompt: Text description of the image to generate
  // - seed: Random number for reproducible generation (0 to 858,993,459)
  const prompt = "A stylized picture of a cute old steampunk robot";
  const seed = Math.floor(Math.random() * 858993460);

  // Then, create the payload using the following structure:
  // - taskType: TEXT_IMAGE (specifies text-to-image generation)
  // - textToImageParams: Contains the text prompt
  // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
  // For a list of available request parameters, see:
  // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
  const payload = {
    taskType: "TEXT_IMAGE",
    textToImageParams: {
      text: prompt,
    },
    imageGenerationConfig: {
      seed,
      quality: "standard",
    },
  };

  // Step 4: Send and process the request
  // - Embed the payload in a request object
  // - Send the request to the model
  // - Extract and return the generated image data from the response
  try {
    const request = {
      modelId,
      body: JSON.stringify(payload),
    };
    const response = await client.send(new InvokeModelCommand(request));

    const decodedResponseBody = new TextDecoder().decode(response.body);
    // The response includes an array of base64-encoded PNG images
    /** @type {{images: string[]}} */
    const responseBody = JSON.parse(decodedResponseBody);
    return responseBody.images[0]; // Base64-encoded image data
  } catch (error) {
```

```
    console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
    throw error;
  }
};

// If run directly, execute the example and save the generated image
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  console.log("Generating image. This may take a few seconds...");
  invokeModel()
    .then(async (imageData) => {
      const imagePath = await saveImage(imageData, "nova-canvas");
      // Example path: javascriptv3/example_code/bedrock-runtime/output/nova-
canvas/image-01.png
      console.log(`Image saved to: ${imagePath}`);
    })
    .catch((error) => {
      console.error("Execution failed:", error);
      process.exitCode = 1;
    });
}
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
# Use the native inference API to create an image with Amazon Nova Canvas

import base64
import json
import os
```

```
import random

import boto3

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID.
model_id = "amazon.nova-canvas-v1:0"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed between 0 and 858,993,459
seed = random.randint(0, 858993460)

# Format the request payload using the model's native structure.
native_request = {
    "taskType": "TEXT_IMAGE",
    "textToImageParams": {"text": prompt},
    "imageGenerationConfig": {
        "seed": seed,
        "quality": "standard",
        "height": 512,
        "width": 512,
        "numberOfImages": 1,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
```

```
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"nova_canvas_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"nova_canvas_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Para obter detalhes da API, consulte a InvokeModelReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
// Use the native inference API to create an image with Amazon Nova Canvas

import AWSBedrockRuntime
import AWSSDKIdentity
import Foundation

struct NovaImageOutput: Decodable {
    let images: [Data]
}

func generateImage(_ textPrompt: String) async throws {
    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
```

```
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    config.awsCredentialIdentityResolver = try SSOAWSCredentialIdentityResolver()

    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "amazon.nova-canvas-v1:0"

    //  Format the request payload using the model's native structure.
    let input = InvokeModelInput(
        accept: "application/json",
        body: """
            {
                "textToImageParams": {
                    "text": "\(textPrompt)"
                },
                "taskType": "TEXT_IMAGE",
                "imageGenerationConfig": {
                    "seed": 42,
                    "quality": "standard",
                    "width": 512,
                    "height": 512,
                    "numberOfImages": 1
                }
            }
            """.data(using: .utf8),
        modelId: modelId
    )

    // Invoke the model with the request.
    let response = try await client.invokeModel(input: input)

    // Decode the response body.
    let output = try JSONDecoder().decode(NovaImageOutput.self, from:
response.body!)

    // Extract the image data.
    guard let data = output.images.first else {
        print("No image data found")
        return
    }

    // Save the generated image to a local folder.
    let fileURL = URL.documentsDirectory.appending(path: "nova_canvas.png")
```

```
    print(fileURL)
    try data.write(to: fileURL)
    print("Image is saved at \(fileURL)")
}
```

- Para obter detalhes da API, consulte InvokeModela referência da API AWS SDK for Swift.

Amazon Nova Reel para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Usar o Amazon Nova Reel para gerar um vídeo de um prompt de texto

Usar o Amazon Nova Reel para gerar um vídeo de um prompt de texto

Os exemplos de código a seguir mostram como usar o Amazon Nova Reel para gerar um vídeo com base em um prompt de texto.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o Amazon Nova Reel para gerar um vídeo de um prompt de texto.

```
"""
This example demonstrates how to use Amazon Nova Reel to generate a video from a
 text prompt.

It shows how to:
- Set up the Amazon Bedrock runtime client
- Configure a text-to-video request
```

```
- Submit an asynchronous job for video generation
- Poll for job completion status
- Access the generated video from S3
"""

import random
import time

import boto3

# Replace with your own S3 bucket to store the generated video
# Format: s3://your-bucket-name
OUTPUT_S3_URI = "s3://REPLACE-WITH-YOUR-S3-BUCKET-NAME"


def start_text_to_video_generation_job(bedrock_runtime, prompt, output_s3_uri):
    """
    Starts an asynchronous text-to-video generation job using Amazon Nova Reel.

    :param bedrock_runtime: The Bedrock runtime client
    :param prompt: The text description of the video to generate
    :param output_s3_uri: S3 URI where the generated video will be stored

    :return: The invocation ARN of the async job
    """
    # Specify the model ID for text-to-video generation
    model_id = "amazon.nova-reel-v1:0"

    # Generate a random seed between 0 and 2,147,483,646
    # This helps ensure unique video generation results
    seed = random.randint(0, 2147483646)

    # Configure the video generation request with additional parameters
    model_input = {
        "taskType": "TEXT_VIDEO",
        "textToVideoParams": {"text": prompt},
        "videoGenerationConfig": {
            "fps": 24,
            "durationSeconds": 6,
            "dimension": "1280x720",
            "seed": seed,
        },
    }
```

```
    # Specify the S3 location for the output video
    output_config = {"s3OutputDataConfig": {"s3Uri": output_s3_uri}}

    # Invoke the model asynchronously
    response = bedrock_runtime.start_async_invoke(
        modelId=model_id, modelInput=model_input, outputDataConfig=output_config
    )

    invocation_arn = response["invocationArn"]

    return invocation_arn


def query_job_status(bedrock_runtime, invocation_arn):
    """
    Queries the status of an asynchronous video generation job.

    :param bedrock_runtime: The Bedrock runtime client
    :param invocation_arn: The ARN of the async invocation to check

    :return: The runtime response containing the job status and details
    """
    return bedrock_runtime.get_async_invoke(invocationArn=invocation_arn)


def main():
    """
    Main function that demonstrates the complete workflow for generating
    a video from a text prompt using Amazon Nova Reel.
    """
    # Create a Bedrock Runtime client
    # Note: Credentials will be loaded from the environment or AWS CLI config
    bedrock_runtime = boto3.client("bedrock-runtime", region_name="us-east-1")

    # Configure the text prompt and output location
    prompt = "Closeup of a cute old steampunk robot. Camera zoom in."

    # Verify the S3 URI has been set to a valid bucket
    if "REPLACE-WITH-YOUR-S3-BUCKET-NAME" in OUTPUT_S3_URI:
        print("ERROR: You must replace the OUTPUT_S3_URI with your own S3 bucket
 URI")
        return

    print("Submitting video generation job...")
```

```
    invocation_arn = start_text_to_video_generation_job(
        bedrock_runtime, prompt, OUTPUT_S3_URI
    )
    print(f"Job started with invocation ARN: {invocation_arn}")

    # Poll for job completion
    while True:
        print("\nPolling job status...")
        job = query_job_status(bedrock_runtime, invocation_arn)
        status = job["status"]

        if status == "Completed":
            bucket_uri = job["outputDataConfig"]["s3OutputDataConfig"]["s3Uri"]
            print(f"\nSuccess! The video is available at: {bucket_uri}/
output.mp4")
            break
        elif status == "Failed":
            print(
                f"\nVideo generation failed: {job.get('failureMessage', 'Unknown
 error')}"
            )
            break
        else:
            print("In progress. Waiting 15 seconds...")
            time.sleep(15)


if __name__ == "__main__":
    main()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - GetAsyncInvoke
  - StartAsyncInvoke

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o Amazon Nova Reel para gerar um vídeo de um prompt de texto.

```
// This example demonstrates how to use Amazon Nova Reel to generate a video from
 a text prompt.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Configure a text-to-video request
// - Submit an asynchronous job for video generation
// - Poll for job completion status
// - Access the generated video from S3

import AWSBedrockRuntime
import Foundation
import Smithy

func startTextToVideoGenerationJob(
    bedrockRuntimeClient: BedrockRuntimeClient, prompt: String, outputS3Uri:
 String
) async throws -> String? {
    // Specify the model ID for text-to-video generation
    let modelId = "amazon.nova-reel-v1:0"

    // Configure the video generation request with additional parameters
    let modelInputSource: [String: Any] = [
        "taskType": "TEXT_VIDEO",
        "textToVideoParams": [
            "text": "\(prompt)"
        ],
        "videoGenerationConfig": [
            "durationSeconds": 6,
            "fps": 24,
            "dimension": "1280x720",
        ],
```

```
    ]

    let modelInput = try Document.make(from: modelInputSource)

    let input = StartAsyncInvokeInput(
        modelId: modelId,
        modelInput: modelInput,
        outputDataConfig: .s3outputdataconfig(
            BedrockRuntimeClientTypes.AsyncInvokeS3OutputDataConfig(
                s3Uri: outputS3Uri
            )
        )
    )

    // Invoke the model asynchronously
    let output = try await bedrockRuntimeClient.startAsyncInvoke(input: input)
    return output.invocationArn
}

func queryJobStatus(
    bedrockRuntimeClient: BedrockRuntimeClient,
    invocationArn: String?
) async throws -> GetAsyncInvokeOutput {
    try await bedrockRuntimeClient.getAsyncInvoke(
        input: GetAsyncInvokeInput(invocationArn: invocationArn))
}

func main() async throws {
    // Create a Bedrock Runtime client
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Specify the S3 location for the output video
    let bucket = "s3://REPLACE-WITH-YOUR-S3-BUCKET-NAM"

    print("Submitting video generation job...")
    let invocationArn = try await startTextToVideoGenerationJob(
        bedrockRuntimeClient: client,
        prompt: "A pomegranate juice in a railway station",
        outputS3Uri: bucket
    )
    print("Job started with invocation ARN: \(String(describing:invocationArn))")
```

```
    // Poll for job completion
    var status: BedrockRuntimeClientTypes.AsyncInvokeStatus?
    var isReady = false
    var hasFailed = false

    while !isReady && !hasFailed {
        print("\nPolling job status...")
        status = try await queryJobStatus(
            bedrockRuntimeClient: client, invocationArn: invocationArn
        ).status
        switch status {
        case .completed:
            isReady = true
            print("Video is ready\nCheck S3 bucket: \(bucket)")
        case .failed:
            hasFailed = true
            print("Something went wrong")
        case .inProgress:
            print("Job is in progress...")
            try await Task.sleep(nanoseconds: 15 * 1_000_000_000)  // 15 seconds
        default:
            isReady = true
        }
    }
}

do {
    try await main()
} catch {
    print("An error occurred: \(error)")
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - GetAsyncInvoke
  - StartAsyncInvoke

Gerador de Imagens do Amazon Titan para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar o Amazon Titan Image no Amazon Bedrock para gerar uma imagem

Invocar o Amazon Titan Image no Amazon Bedrock para gerar uma imagem

Os exemplos de código a seguir mostram como invocar o Amazon Titan Image no Amazon Bedrock para gerar uma imagem.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Gerador de Imagens do Amazon Titan.

```
import (
 "context"
 "encoding/json"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
)

// InvokeModelWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke foundation models.
type InvokeModelWrapper struct {
 BedrockRuntimeClient *bedrockruntime.Client
}
```

```
type TitanImageRequest struct {
 TaskType              string                `json:"taskType"`
 TextToImageParams     TextToImageParams     `json:"textToImageParams"`
 ImageGenerationConfig ImageGenerationConfig `json:"imageGenerationConfig"`
}
type TextToImageParams struct {
 Text string `json:"text"`
}
type ImageGenerationConfig struct {
 NumberOfImages int     `json:"numberOfImages"`
 Quality        string  `json:"quality"`
 CfgScale       float64 `json:"cfgScale"`
 Height         int     `json:"height"`
 Width          int     `json:"width"`
 Seed           int64   `json:"seed"`
}

type TitanImageResponse struct {
 Images []string `json:"images"`
}

// Invokes the Titan Image model to create an image using the input provided
// in the request body.
func (wrapper InvokeModelWrapper) InvokeTitanImage(ctx context.Context, prompt
 string, seed int64) (string, error) {
 modelId := "amazon.titan-image-generator-v2:0"

 body, err := json.Marshal(TitanImageRequest{
  TaskType: "TEXT_IMAGE",
  TextToImageParams: TextToImageParams{
   Text: prompt,
  },
  ImageGenerationConfig: ImageGenerationConfig{
   NumberOfImages: 1,
   Quality:        "standard",
   CfgScale:       8.0,
   Height:         512,
   Width:          512,
   Seed:           seed,
  },
 })
```

```
 if err != nil {
  log.Fatal("failed to marshal", err)
 }

 output, err := wrapper.BedrockRuntimeClient.InvokeModel(ctx,
 &bedrockruntime.InvokeModelInput{
  ModelId:     aws.String(modelId),
  ContentType: aws.String("application/json"),
  Body:        body,
 })

 if err != nil {
  ProcessError(err, modelId)
 }

 var response TitanImageResponse
 if err := json.Unmarshal(output.Body, &response); err != nil {
  log.Fatal("failed to unmarshal", err)
 }

 base64ImageData := response.Images[0]

 return base64ImageData, nil

}
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Gerador de Imagens do Amazon Titan.

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Image G2.
        var modelId = "amazon.titan-image-generator-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response
 fields at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
        var nativeRequestTemplate = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": { "text": "{{prompt}}" },
                    "imageGenerationConfig": { "seed": {{seed}} }
                }""";

        // Define the prompt for the image generation.
```

```
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 31-bit seed for the image generation (max.
 2,147,483,647).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        var nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }
}
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Gerador de Imagens do Amazon Titan.

```
# Use the native inference API to create an image with Amazon Titan Image
 Generator

import base64
import boto3
import json
import os
import random

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Image Generator G1.
model_id = "amazon.titan-image-generator-v2:0"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed.
seed = random.randint(0, 2147483647)

# Format the request payload using the model's native structure.
native_request = {
    "taskType": "TEXT_IMAGE",
    "textToImageParams": {"text": prompt},
    "imageGenerationConfig": {
        "numberOfImages": 1,
```

```
        "quality": "standard",
        "cfgScale": 8.0,
        "height": 512,
        "width": 512,
        "seed": seed,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"titan_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"titan_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Para obter detalhes da API, consulte a InvokeModelReferência da API AWS SDK for Python (Boto3).

Amazon Titan Text para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar modelos Amazon Titan Text no Amazon Bedrock usando a API InvokeModel

Invocar modelos Amazon Titan Text no Amazon Bedrock usando a API InvokeModel

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Amazon Titan Text usando a API InvokeModel.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a API InvokeModel para enviar uma mensagem de texto.

```
# Use the native inference API to send a text message to Amazon Titan Text.

import boto3
import json

from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Text Premier.
model_id = "amazon.titan-text-premier-v1:0"

# Define the prompt for the model.
prompt = "Describe the purpose of a 'hello world' program in one line."

# Format the request payload using the model's native structure.
```

```
native_request = {
    "inputText": prompt,
    "textGenerationConfig": {
        "maxTokenCount": 512,
        "temperature": 0.5,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

try:
    # Invoke the model with the request.
    response = client.invoke_model(modelId=model_id, body=request)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract and print the response text.
response_text = model_response["results"][0]["outputText"]
print(response_text)
```

- Para obter detalhes da API, consulte a InvokeModelReferência da API AWS SDK for Python (Boto3).

Incorporador de Texto do Amazon Titan para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar o Incorporador de Texto do Amazon Titan no Amazon Bedrock

Invocar o Incorporador de Texto do Amazon Titan no Amazon Bedrock

Os exemplos de código a seguir mostram como:

- Começar a criar sua primeira incorporação.
- Criar incorporações configurando o número de dimensões e a normalização (somente v2).

Java

SDK para Java 2.x

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a primeira incorporação com o Incorporador de Texto do Titan v2.

```
// Generate and print an embedding with Amazon Titan Text Embeddings.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Embeddings V2.
        var modelId = "amazon.titan-embed-text-v2:0";

        // The InvokeModel API uses the model's native payload.
```

```
        // Learn more about the available inference parameters and response
 fields at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-embed-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{inputText}}\" }";

        // The text to convert into an embedding.
        var inputText = "Please recommend books with a theme similar to the movie
 'Inception'.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{inputText}}",
 inputText);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/
embedding").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

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

Invoque o Incorporador de Texto do Titan v2 configurando o número de dimensões e a normalização.

```
   /**
    * Invoke Amazon Titan Text Embeddings V2 with additional inference
parameters.
    *
    * @param inputText  - The text to convert to an embedding.
    * @param dimensions - The number of dimensions the output embeddings should
have.
    *                   Values accepted by the model: 256, 512, 1024.
    * @param normalize  - A flag indicating whether or not to normalize the
output embeddings.
    * @return The {@link JSONObject} representing the model's response.
    */
   public static JSONObject invokeModel(String inputText, int dimensions,
boolean normalize) {

       // Create a Bedrock Runtime client in the AWS Region of your choice.
       var client = BedrockRuntimeClient.builder()
               .region(Region.US_WEST_2)
               .build();

       // Set the model ID, e.g., Titan Embed Text v2.0.
       var modelId = "amazon.titan-embed-text-v2:0";

       // Create the request for the model.
       var nativeRequest = """
               {
                   "inputText": "%s",
                   "dimensions": %d,
                   "normalize": %b
               }
               """.formatted(inputText, dimensions, normalize);

       // Encode and send the request.
       var response = client.invokeModel(request -> {
           request.body(SdkBytes.fromUtf8String(nativeRequest));
           request.modelId(modelId);
       });

       // Decode the model's response.
```

```
        var modelResponse = new JSONObject(response.body().asUtf8String());

        // Extract and print the generated embedding and the input text token
 count.
        var embedding = modelResponse.getJSONArray("embedding");
        var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
        System.out.println("Embedding: " + embedding);
        System.out.println("\nInput token count: " + inputTokenCount);

        // Return the model's native response.
        return modelResponse;
    }
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a primeira incorporação com o Incorporador de Texto do Amazon Titan.

```
# Generate and print an embedding with Amazon Titan Text Embeddings V2.

import boto3
import json

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Text Embeddings V2.
model_id = "amazon.titan-embed-text-v2:0"

# The text to convert to an embedding.
```

```
input_text = "Please recommend books with a theme similar to the movie
 'Inception'."

# Create the request for the model.
native_request = {"inputText": input_text}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the model's native response body.
model_response = json.loads(response["body"].read())

# Extract and print the generated embedding and the input text token count.
embedding = model_response["embedding"]
input_token_count = model_response["inputTextTokenCount"]

print("\nYour input:")
print(input_text)
print(f"Number of input tokens: {input_token_count}")
print(f"Size of the generated embedding: {len(embedding)}")
print("Embedding:")
print(embedding)
```

- Para obter detalhes da API, consulte a InvokeModelReferência da API AWS SDK for Python (Boto3).

Claude da Anthropic para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar o Claude da Anthropic no Amazon Bedrock usando a API Converse do Bedrock
- Invocar o Claude da Anthropic no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta
- Enviar e processar um documento com o Claude da Anthropic no Amazon Bedrock

- Usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock
- Usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock
- Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada

Invocar o Claude da Anthropic no Amazon Bedrock usando a API Converse do Bedrock

Os exemplos de código a seguir mostram como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";
```

```
// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
import (
 "context"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// ConverseWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke Bedrock.
type ConverseWrapper struct {
 BedrockRuntimeClient *bedrockruntime.Client
}



func (wrapper ConverseWrapper) ConverseClaude(ctx context.Context, prompt string)
 (string, error) {
 var content = types.ContentBlockMemberText{
  Value: prompt,
 }
 var message = types.Message{
  Content: []types.ContentBlock{&content},
  Role:    "user",
 }
 modelId := "anthropic.claude-3-haiku-20240307-v1:0"
 var converseInput = bedrockruntime.ConverseInput{
  ModelId:  aws.String(modelId),
  Messages: []types.Message{message},
 }
 response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
```

```
 if err != nil {
  ProcessError(err, modelId)
 }

 responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
 responseContentBlock := responseText.Value.Content[0]
 text, _ := responseContentBlock.(*types.ContentBlockMemberText)
 return text.Value, nil

}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
response.output().message().content().getFirst().text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
```

```
    }

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

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock com o cliente Java assíncrono.

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
```

```
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().getFirst().text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

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

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Anthropic Claude.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Claude 3 Haiku.
const modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];
```

```
// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Anthropic Claude.

import boto3
from botocore.exceptions import ClientError
```

```
# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
#[tokio::main]
async fn main() -> Result<(), BedrockConverseError> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let response = client
        .converse()
        .model_id(MODEL_ID)
        .messages(
            Message::builder()
                .role(ConversationRole::User)
                .content(ContentBlock::Text(USER_MESSAGE.to_string()))
                .build()
                .map_err(|_| "failed to build message")?,
        )
        .send()
        .await;

    match response {
        Ok(output) => {
            let text = get_converse_output_text(output)?;
            println!("{}", text);
            Ok(())
        }
        Err(e) => Err(e
            .as_service_error()
            .map(BedrockConverseError::from)
```

```
            .unwrap_or_else(|| BedrockConverseError("Unknown service
 error".into()))),
    }
}

fn get_converse_output_text(output: ConverseOutput) -> Result<String,
 BedrockConverseError> {
    let text = output
        .output()
        .ok_or("no output")?
        .as_message()
        .map_err(|_| "output not a message")?
        .content()
        .first()
        .ok_or("no content in message")?
        .as_text()
        .map_err(|_| "content is not text")?
        .to_string();
    Ok(text)
}
```

Use instruções, utilitário de erro e constantes.

```
use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    operation::converse::{ConverseError, ConverseOutput},
    types::{ContentBlock, ConversationRole, Message},
    Client,
};

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

// Start a conversation with the user message.
const USER_MESSAGE: &str = "Describe the purpose of a 'hello world' program in
 one line.";

#[derive(Debug)]
struct BedrockConverseError(String);
impl std::fmt::Display for BedrockConverseError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
```

```
        write!(f, "Can't invoke '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl std::error::Error for BedrockConverseError {}
impl From<&str> for BedrockConverseError {
    fn from(value: &str) -> Self {
        BedrockConverseError(value.to_string())
    }
}
impl From<&ConverseError> for BedrockConverseError {
    fn from(value: &ConverseError) -> Self {
        BedrockConverseError::from(match value {
            ConverseError::ModelTimeoutException(_) => "Model took too long",
            ConverseError::ModelNotReadyException(_) => "Model is not ready",
            _ => "Unknown",
        })
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Rust.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Anthropic Claude.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "anthropic.claude-3-haiku-20240307-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Swift.

Invocar o Claude da Anthropic no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta

Os exemplos de código a seguir mostram como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
```

```
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";
```

```
        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach
 the handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Conversation API to send a text message to Anthropic Claude.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Claude 3 Haiku.
const modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});
```

```
try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
# Use the Conversation API to send a text message to Anthropic Claude
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStreamReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para Anthropic Claude e transmita tokens de resposta usando a API do Bedrock. ConverseStream

```
#[tokio::main]
async fn main() -> Result<(), BedrockConverseStreamError> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let response = client
        .converse_stream()
        .model_id(MODEL_ID)
        .messages(
            Message::builder()
                .role(ConversationRole::User)
                .content(ContentBlock::Text(USER_MESSAGE.to_string()))
                .build()
                .map_err(|_| "failed to build message")?,
        )
        .send()
        .await;

    let mut stream = match response {
        Ok(output) => Ok(output.stream),
        Err(e) => Err(BedrockConverseStreamError::from(
            e.as_service_error().unwrap(),
        )),
    }?;
```

```
    loop {
        let token = stream.recv().await;
        match token {
            Ok(Some(text)) => {
                let next = get_converse_output_text(text)?;
                print!("{}", next);
                Ok(())
            }
            Ok(None) => break,
            Err(e) => Err(e
                .as_service_error()
                .map(BedrockConverseStreamError::from)
                .unwrap_or(BedrockConverseStreamError(
                    "Unknown error receiving stream".into(),
                ))),
        }?
    }

    println!();

    Ok(())
}

fn get_converse_output_text(
    output: ConverseStreamOutputType,
) -> Result<String, BedrockConverseStreamError> {
    Ok(match output {
        ConverseStreamOutputType::ContentBlockDelta(event) => match event.delta()
 {
            Some(delta) => delta.as_text().cloned().unwrap_or_else(|_|
 "".into()),
            None => "".into(),
        },
        _ => "".into(),
    })
}
```

Use instruções, utilitário de erro e constantes.

```
use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
```

```
    error::ProvideErrorMetadata,
    operation::converse_stream::ConverseStreamError,
    types::{
        error::ConverseStreamOutputError, ContentBlock, ConversationRole,
        ConverseStreamOutput as ConverseStreamOutputType, Message,
    },
    Client,
};

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

// Start a conversation with the user message.
const USER_MESSAGE: &str = "Describe the purpose of a 'hello world' program in
 one line.";

#[derive(Debug)]
struct BedrockConverseStreamError(String);
impl std::fmt::Display for BedrockConverseStreamError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Can't invoke '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl std::error::Error for BedrockConverseStreamError {}
impl From<&str> for BedrockConverseStreamError {
    fn from(value: &str) -> Self {
        BedrockConverseStreamError(value.into())
    }
}

impl From<&ConverseStreamError> for BedrockConverseStreamError {
    fn from(value: &ConverseStreamError) -> Self {
        BedrockConverseStreamError(
            match value {
                ConverseStreamError::ModelTimeoutException(_) => "Model took too
 long",
                ConverseStreamError::ModelNotReadyException(_) => "Model is not
 ready",
                _ => "Unknown",
            }
            .into(),
        )
    }
```

```
}

impl From<&ConverseStreamOutputError> for BedrockConverseStreamError {
    fn from(value: &ConverseStreamOutputError) -> Self {
        match value {
            ConverseStreamOutputError::ValidationException(ve) =>
 BedrockConverseStreamError(
                ve.message().unwrap_or("Unknown ValidationException").into(),
            ),
            ConverseStreamOutputError::ThrottlingException(te) =>
 BedrockConverseStreamError(
                te.message().unwrap_or("Unknown ThrottlingException").into(),
            ),
            value => BedrockConverseStreamError(
                value
                    .message()
                    .unwrap_or("Unknown StreamOutput exception")
                    .into(),
            ),
        }
    }
}
```

- Para obter detalhes da API, consulte a ConverseStreamreferência da API AWS SDK for Rust.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// An example demonstrating how to use the Conversation API to send a text
 message
// to Anthropic Claude and print the response stream

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "anthropic.claude-3-haiku-20240307-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }
```

```
    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nAnthropic Claude:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStreama referência da API AWS SDK for Swift.

Enviar e processar um documento com o Claude da Anthropic no Amazon Bedrock

O exemplo de código a seguir mostra como enviar e processar um documento com o Anthropic Claude no Amazon Bedrock.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Enviar e processar um documento com o Claude da Anthropic no Amazon Bedrock.

```
# Send and process a document with Anthropic Claude on Amazon Bedrock.
```

```
import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)
```

```
except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock

O exemplo de código a seguir mostra como usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a capacidade de raciocínio do Anthropic Claude 3.7 Sonnet com o cliente de runtime assíncrono do Bedrock.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
```

```
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with an asynchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock async runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send an asynchronous request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class ReasoningAsync {

    public static ReasoningResponse reasoningAsync() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
```

```
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );

            // Process the response asynchronously
            return asyncResponse.thenApply(response -> {

                        var content = response.output().message().content();
                        ReasoningContentBlock reasoning = null;
                        String text = null;

                        // Process each content block to find reasoning and
 response text
                        for (ContentBlock block : content) {
                            if (block.reasoningContent() != null) {
                                reasoning = block.reasoningContent();
                            } else if (block.text() != null) {
                                text = block.text();
                            }
                        }

                        return new ReasoningResponse(reasoning, text);
                    }
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoningAsync();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

Use a capacidade de raciocínio do Anthropic Claude 3.7 Sonnet com o cliente de runtime síncrono do Bedrock.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with the synchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send a request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class Reasoning {

    public static ReasoningResponse reasoning() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one
 line.";
```

```
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            ConverseResponse bedrockResponse = client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );


            // Extract both reasoning and final response
            var content = bedrockResponse.output().message().content();
            ReasoningContentBlock reasoning = null;
            String text = null;

            // Process each content block to find reasoning and response text
            for (ContentBlock block : content) {
                if (block.reasoningContent() != null) {
                    reasoning = block.reasoningContent();
                } else if (block.text() != null) {
                    text = block.text();
                }
            }

            return new ReasoningResponse(reasoning, text);

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoning();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

Usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock

O exemplo de código a seguir mostra como usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a capacidade de raciocínio do Anthropic Claude 3.7 Sonnet para gerar respostas de streaming de texto.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;
```

```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.atomic.AtomicReference;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 * capability to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ReasoningStream {

    public static ReasoningResponse reasoningStream() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();
```

```
        // Configure the request with the message, model ID, and reasoning config
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .additionalModelRequestFields(reasoningConfig)
                .messages(message)
                .modelId(modelId)
                .build();

        StringBuilder reasoning = new StringBuilder();
        StringBuilder text = new StringBuilder();
        AtomicReference<ReasoningResponse> finalresponse = new
AtomicReference<>();

        // Set up the stream handler to processes chunks of the response as they
arrive
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            ContentBlockDelta delta = chunk.delta();
                            if (delta.reasoningContent() != null) {
                                if (reasoning.isEmpty()) {
                                    System.out.println("\n<thinking>");
                                }
                                if (delta.reasoningContent().text() != null) {

System.out.print(delta.reasoningContent().text());

reasoning.append(delta.reasoningContent().text());
                                }
                            } else if (delta.text() != null) {
                                if (text.isEmpty()) {
                                    System.out.println("\n</thinking>\n");
                                }
                                System.out.print(delta.text());
                                text.append(delta.text());
                            }
                            System.out.flush();  // Ensure immediate output of
each chunk
                        }).build())
                .onComplete(() -> finalresponse.set(new ReasoningResponse(
                        ReasoningContentBlock.fromReasoningText(t ->
t.text(reasoning.toString())),
                        text.toString()
                )))
```

```
                .onError(err -> System.err.printf("Can't invoke '%s': %s",
 modelId, err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
            return finalresponse.get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

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

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada

Os exemplos de código a seguir mostram como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O script de execução principal da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

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

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-
inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
```

```
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
```

```
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has
returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()
```

```
    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message
to send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and
return the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try
again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
```

```
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool
use handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text,
and finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use
request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the
tool
                tool_response = self._invoke_tool(content_block["toolUse"])
```

```
                # Add the tool use ID and the tool's response to the list of
results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
```

```
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
import requests
from requests.exceptions import RequestException
```

```
def get_tool_spec():
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based
 on its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
```

```
    :return: The weather data or an error message.
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather":
 True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O cenário principal e a lógica da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
#[derive(Debug)]
#[allow(dead_code)]
struct InvokeToolResult(String, ToolResultBlock);
struct ToolUseScenario {
    client: Client,
    conversation: Vec<Message>,
```

```
    system_prompt: SystemContentBlock,
    tool_config: ToolConfiguration,
}

impl ToolUseScenario {
    fn new(client: Client) -> Self {
        let system_prompt = SystemContentBlock::Text(SYSTEM_PROMPT.into());
        let tool_config = ToolConfiguration::builder()
            .tools(Tool::ToolSpec(
                ToolSpecification::builder()
                    .name(TOOL_NAME)
                    .description(TOOL_DESCRIPTION)
                    .input_schema(ToolInputSchema::Json(make_tool_schema()))
                    .build()
                    .unwrap(),
            ))
            .build()
            .unwrap();

        ToolUseScenario {
            client,
            conversation: vec![],
            system_prompt,
            tool_config,
        }
    }

    async fn run(&mut self) -> Result<(), ToolUseScenarioError> {
        loop {
            let input = get_input().await?;
            if input.is_none() {
                break;
            }

            let message = Message::builder()
                .role(User)
                .content(ContentBlock::Text(input.unwrap()))
                .build()
                .map_err(ToolUseScenarioError::from)?;
            self.conversation.push(message);

            let response = self.send_to_bedrock().await?;

            self.process_model_response(response).await?;
```

```
        }

        Ok(())
    }

    async fn send_to_bedrock(&mut self) -> Result<ConverseOutput,
ToolUseScenarioError> {
        debug!("Sending conversation to bedrock");
        self.client
            .converse()
            .model_id(MODEL_ID)
            .set_messages(Some(self.conversation.clone()))
            .system(self.system_prompt.clone())
            .tool_config(self.tool_config.clone())
            .send()
            .await
            .map_err(ToolUseScenarioError::from)
    }

    async fn process_model_response(
        &mut self,
        mut response: ConverseOutput,
    ) -> Result<(), ToolUseScenarioError> {
        let mut iteration = 0;

        while iteration < MAX_RECURSIONS {
            iteration += 1;
            let message = if let Some(ref output) = response.output {
                if output.is_message() {
                    Ok(output.as_message().unwrap().clone())
                } else {
                    Err(ToolUseScenarioError(
                        "Converse Output is not a message".into(),
                    ))
                }
            } else {
                Err(ToolUseScenarioError("Missing Converse Output".into()))
            }?;

            self.conversation.push(message.clone());

            match response.stop_reason {
                StopReason::ToolUse => {
                    response = self.handle_tool_use(&message).await?;
```

```
                }
                StopReason::EndTurn => {
                    print_model_response(&message.content[0])?;
                    return Ok(());
                }
                _ => (),
            }
        }

        Err(ToolUseScenarioError(
            "Exceeded MAX_ITERATIONS when calling tools".into(),
        ))
    }

    async fn handle_tool_use(
        &mut self,
        message: &Message,
    ) -> Result<ConverseOutput, ToolUseScenarioError> {
        let mut tool_results: Vec<ContentBlock> = vec![];

        for block in &message.content {
            match block {
                ContentBlock::Text(_) => print_model_response(block)?,
                ContentBlock::ToolUse(tool) => {
                    let tool_response = self.invoke_tool(tool).await?;
                    tool_results.push(ContentBlock::ToolResult(tool_response.1));
                }
                _ => (),
            };
        }

        let message = Message::builder()
            .role(User)
            .set_content(Some(tool_results))
            .build()?;
        self.conversation.push(message);

        self.send_to_bedrock().await
    }

    async fn invoke_tool(
        &mut self,
        tool: &ToolUseBlock,
    ) -> Result<InvokeToolResult, ToolUseScenarioError> {
```

```
        match tool.name() {
            TOOL_NAME => {
                println!(
                    "\x1b[0;90mExecuting tool: {TOOL_NAME} with input: {:?}...
\x1b[0m",
                    tool.input()
                );
                let content = fetch_weather_data(tool).await?;
                println!(
                    "\x1b[0;90mTool responded with {:?}\x1b[0m",
                    content.content()
                );
                Ok(InvokeToolResult(tool.tool_use_id.clone(), content))
            }
            _ => Err(ToolUseScenarioError(format!(
                "The requested tool with name {} does not exist",
                tool.name()
            ))),
        }
    }
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let mut scenario = ToolUseScenario::new(client);

    header();
    if let Err(err) = scenario.run().await {
        println!("There was an error running the scenario! {}", err.0)
    }
    footer();
}
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
const ENDPOINT: &str = "https://api.open-meteo.com/v1/forecast";
async fn fetch_weather_data(
    tool_use: &ToolUseBlock,
) -> Result<ToolResultBlock, ToolUseScenarioError> {
    let input = tool_use.input();
    let latitude = input
        .as_object()
        .unwrap()
        .get("latitude")
        .unwrap()
        .as_string()
        .unwrap();
    let longitude = input
        .as_object()
        .unwrap()
        .get("longitude")
        .unwrap()
        .as_string()
        .unwrap();
    let params = [
        ("latitude", latitude),
        ("longitude", longitude),
        ("current_weather", "true"),
    ];

    debug!("Calling {ENDPOINT} with {params:?}");

    let response = reqwest::Client::new()
        .get(ENDPOINT)
        .query(&params)
        .send()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error requesting weather:
 {e:?}")))?
        .error_for_status()
        .map_err(|e| ToolUseScenarioError(format!("Failed to request weather:
 {e:?}")))?;

    debug!("Response: {response:?}");
```

```
    let bytes = response
        .bytes()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error reading response:
 {e:?}")))?;

    let result = String::from_utf8(bytes.to_vec())
        .map_err(|_| ToolUseScenarioError("Response was not utf8".into()))?;

    Ok(ToolResultBlock::builder()
        .tool_use_id(tool_use.tool_use_id())
        .content(ToolResultContentBlock::Text(result))
        .build()?)
}
```

Utilitários para imprimir os blocos de conteúdo da mensagem.

```
fn print_model_response(block: &ContentBlock) -> Result<(), ToolUseScenarioError>
 {
    if block.is_text() {
        let text = block.as_text().unwrap();
        println!("\x1b[0;90mThe model's response:\x1b[0m\n{text}");
        Ok(())
    } else {
        Err(ToolUseScenarioError(format!(
            "Content block is not text ({block:?})"
        )))
    }
}
```

Use instruções, utilitário de erro e constantes.

```
use std::{collections::HashMap, io::stdin};

use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::{BuildError, SdkError},
    operation::converse::{ConverseError, ConverseOutput},
    types::{
```

```
        ContentBlock, ConversationRole::User, Message, StopReason,
 SystemContentBlock, Tool,
        ToolConfiguration, ToolInputSchema, ToolResultBlock,
 ToolResultContentBlock,
        ToolSpecification, ToolUseBlock,
    },
    Client,
};
use aws_smithy_runtime_api::http::Response;
use aws_smithy_types::Document;
use tracing::debug;

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

const SYSTEM_PROMPT: &str = "You are a weather assistant that provides current
 weather data for user-specified locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
";

// The maximum number of recursive calls allowed in the tool_use_demo function.
// This helps prevent infinite loops and potential performance issues.
const MAX_RECURSIONS: i8 = 5;

const TOOL_NAME: &str = "Weather_Tool";
```

```
const TOOL_DESCRIPTION: &str =
    "Get the current weather for a given location, based on its WGS84
 coordinates.";
fn make_tool_schema() -> Document {
    Document::Object(HashMap::<String, Document>::from([
        ("type".into(), Document::String("object".into())),
        (
            "properties".into(),
            Document::Object(HashMap::from([
                (
                    "latitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String("Geographical WGS84 latitude of the
 location.".into()),
                        ),
                    ])),
                ),
                (
                    "longitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String(
                                "Geographical WGS84 longitude of the
 location.".into(),
                            ),
                        ),
                    ])),
                ),
            ])),
        ),
        (
            "required".into(),
            Document::Array(vec![
                Document::String("latitude".into()),
                Document::String("longitude".into()),
            ]),
        ),
    ]))
}
```

```
#[derive(Debug)]
struct ToolUseScenarioError(String);
impl std::fmt::Display for ToolUseScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Tool use error with '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl From<&str> for ToolUseScenarioError {
    fn from(value: &str) -> Self {
        ToolUseScenarioError(value.into())
    }
}
impl From<BuildError> for ToolUseScenarioError {
    fn from(value: BuildError) -> Self {
        ToolUseScenarioError(value.to_string().clone())
    }
}
impl From<SdkError<ConverseError, Response>> for ToolUseScenarioError {
    fn from(value: SdkError<ConverseError, Response>) -> Self {
        ToolUseScenarioError(match value.as_service_error() {
            Some(value) => value.meta().message().unwrap_or("Unknown").into(),
            None => "Unknown".into(),
        })
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Rust.

Command da Cohere para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar o Command da Cohere no Amazon Bedrock usando a API Converse do Bedrock
- Invocar o Command da Cohere no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta
- Enviar e processar um documento com modelos Cohere Command no Amazon Bedrock

- Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada

Invocar o Command da Cohere no Amazon Bedrock usando a API Converse do Bedrock

Os exemplos de código a seguir mostram como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Cohere Command usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Cohere Command.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
```

```
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para .NET .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Cohere Command usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Cohere Command.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
```

```
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

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

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock com o cliente Java assíncrono.

```
// Use the Converse API to send a text message to Cohere Command
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();
```

```
        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

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

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Cohere Command usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Cohere Command.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Command R.
const modelId = "cohere.command-r-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
```

```
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Cohere Command usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Cohere Command.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Command R.
model_id = "cohere.command-r-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
```

```
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Invocar o Command da Cohere no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta

Os exemplos de código a seguir mostram como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};
```

```
try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
```

```
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach
 the handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Conversation API to send a text message to Cohere Command.
```

```
import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Command R.
const modelId = "cohere.command-r-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
# Use the Conversation API to send a text message to Cohere Command
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Command R.
model_id = "cohere.command-r-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
```

```
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStreamReferência da API AWS SDK for Python (Boto3).

Enviar e processar um documento com modelos Cohere Command no Amazon Bedrock

O exemplo de código a seguir mostra como enviar e processar um documento com modelos Cohere Command no Amazon Bedrock.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com modelos Cohere Command no Amazon Bedrock.

```
# Send and process a document with Cohere Command models on Amazon Bedrock.
```

```
import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Command R+.
model_id = "cohere.command-r-plus-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)
```

```
except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Demonstração do uso de ferramentas ilustrando como conectar modelos de IA no Amazon Bedrock a uma ferramenta ou API personalizada

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O script de execução principal da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""
```

```
import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

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

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-
inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
```

```
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
 Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()
```

```
        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has
returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()

    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message
to send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and
return the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
```

```
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try
again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool
use handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text,
and finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.
```

```
        :param model_response: The model's response containing the tool use
request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the
tool
                tool_response = self._invoke_tool(content_block["toolUse"])

                # Add the tool use ID and the tool's response to the list of
results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)
```

```
    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None
```

```
        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based
 on its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
```

```
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather":
 True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

DeepSeek para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Envie e processe um documento com o DeepSeek Amazon Bedrock

Envie e processe um documento com o DeepSeek Amazon Bedrock

O exemplo de código a seguir mostra como enviar e processar um documento DeepSeek no Amazon Bedrock.

Python

SDK para Python (Boto3)

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento DeepSeek no Amazon Bedrock.

```
# Send and process a document with DeepSeek on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. DeepSeek-R1
model_id = "us.deepseek.r1-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
```

```
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 2000, "temperature": 0.3},
    )

    # Extract and print the reasoning and response text.
    reasoning, response_text = "", ""
    for item in response["output"]["message"]["content"]:
        for key, value in item.items():
            if key == "reasoningContent":
                reasoning = value["reasoningText"]["text"]
            elif key == "text":
                response_text = value

    print(f"\nReasoning:\n{reasoning}")
    print(f"\nResponse:\n{response_text}")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Llama da Meta para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar o Llama da Meta no Amazon Bedrock usando a API Converse do Bedrock
- Invocar o Llama da Meta no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta
- Enviar e processar um documento com o Llama no Amazon Bedrock

Invocar o Llama da Meta no Amazon Bedrock usando a API Converse do Bedrock

Os exemplos de código a seguir mostram como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Meta Llama.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para .NET .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Meta Llama.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
```

```
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

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

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock com o cliente Java assíncrono.

```
// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
```

```
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

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

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Meta Llama.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Llama 3 8b Instruct.
const modelId = "meta.llama3-8b-instruct-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
```

```
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Meta Llama.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Llama 3 8b Instruct.
model_id = "meta.llama3-8b-instruct-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
```

```
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Meta Llama.
```

```
import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "meta.llama3-8b-instruct-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
```

```
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Swift.

Invocar o Llama da Meta no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta

Os exemplos de código a seguir mostram como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";
```

```
// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
```

```
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach
the handler.
            client.converseStream(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Conversation API to send a text message to Meta Llama.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Llama 3 8b Instruct.
const modelId = "meta.llama3-8b-instruct-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
```

```
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
# Use the Conversation API to send a text message to Meta Llama
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Llama 3 8b Instruct.
model_id = "meta.llama3-8b-instruct-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStreamReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// An example demonstrating how to use the Conversation API to send a text
 message
// to Meta Llama and print the response stream.

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "meta.llama3-8b-instruct-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
```

```
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nMeta Llama:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStreama referência da API AWS SDK for Swift.

Enviar e processar um documento com o Llama no Amazon Bedrock

O exemplo de código a seguir mostra como enviar e processar um documento com o Llama no Amazon Bedrock.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com o Llama no Amazon Bedrock.

```
# Send and process a document with Llama on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Llama 3.1 8B Instruct.
model_id = "us.meta.llama3-1-8b-instruct-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
```

```
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Mistral AI para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invocar a Mistral no Amazon Bedrock usando a API Converse do Bedrock
- Invocar a Mistral no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta
- Enviar e processar um documento com modelos da Mistral no Amazon Bedrock

Invocar a Mistral no Amazon Bedrock usando a API Converse do Bedrock

Os exemplos de código a seguir mostram como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Mistral.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
```

```
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para .NET .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Mistral.
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));
```

```
            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }

    }

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

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock com o cliente Java assíncrono.

```
// Use the Converse API to send a text message to Mistral
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });
```

```
        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

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

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Mistral.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";
```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Mistral Large.
const modelId = "mistral.mistral-large-2402-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Mistral.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)
```

```
except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Invocar a Mistral no Amazon Bedrock usando a API Converse do Bedrock com um fluxo de resposta

Os exemplos de código a seguir mostram como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para a Mistral usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);
```

```
// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
```

```
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para a Mistral usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach
the handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para a Mistral usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Conversation API to send a text message to Mistral.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Mistral Large.
const modelId = "mistral.mistral-large-2402-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
```

```
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte ConverseStreama Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para a Mistral usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
# Use the Conversation API to send a text message to Mistral
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStreamReferência da API AWS SDK for Python (Boto3).

Enviar e processar um documento com modelos da Mistral no Amazon Bedrock

O exemplo de código a seguir mostra como enviar e processar um documento com modelos Mistral no Amazon Bedrock.

Python

SDK para Python (Boto3)

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com modelos Mistral no Amazon Bedrock.

```
# Send and process a document with Mistral models on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
```

```
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Núcleo de imagem estável para Amazon Bedrock Runtime

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Runtime com AWS SDKs.

Exemplos

- Invoque o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem

Invoque o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem

Os exemplos de código a seguir mostram como invocar o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Image Core.

```
        /// <summary>
        /// Asynchronously invokes the Stability.ai Stable Image Core model to
 run an inference based on the provided input.
        /// </summary>
        /// <param name="prompt">The prompt that describes the image Stability.ai
 Stable Image Core has to generate.</param>
        /// <returns>A base-64 encoded image generated by model</returns>
        /// <remarks>
        /// The different model providers have individual request and response
 formats.
        /// For the format, ranges, and default values for Stability.ai Stable
 Image Core, refer to:
        ///     https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-diffusion-stable-image-core-text-image-request-response.html
        /// </remarks>
        public static async Task<string?> InvokeStableImageCoreAsync(string
 prompt, int seed)
        {
            string stableImageCoreModelId = "stability.stable-image-core-v1:1";

            AmazonBedrockRuntimeClient client = new(RegionEndpoint.USWest2);

            string payload = new JsonObject()
            {
```

```
                { "prompt", prompt },
                { "aspect_ratio", "1:1" },
                { "seed", seed },
                { "output_format", "png" }
            }.ToJsonString();

            try
            {
                InvokeModelResponse response = await client.InvokeModelAsync(new
 InvokeModelRequest()
                {
                    ModelId = stableImageCoreModelId,
                    Body = AWSSDKUtils.GenerateMemoryStreamFromString(payload),
                    ContentType = "application/json",
                    Accept = "application/json"
                });

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    var results = JsonNode.ParseAsync(response.Body).Result?
["images"]?.AsArray();

                    return results?[0]?.GetValue<string>();
                }
                else
                {
                    Console.WriteLine("InvokeModelAsync failed with status code "
 + response.HttpStatusCode);
                }
            }
            catch (AmazonBedrockRuntimeException e)
            {
                Console.WriteLine(e.Message);
            }
            return null;
        }
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK para .NET da API.

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Image Core.

```
        /// <summary>
        /// Asynchronously invokes the Stability.ai Stable Image Core model to
 run an inference based on the provided input.
        /// </summary>
        /// <param name="prompt">The prompt that describes the image Stability.ai
 Stable Image Core has to generate.</param>
        /// <returns>A base-64 encoded image generated by model</returns>
        /// <remarks>
        /// The different model providers have individual request and response
 formats.
        /// For the format, ranges, and default values for Stability.ai Stable
 Image Core, refer to:
        ///     https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-diffusion-stable-image-core-text-image-request-response.html
        /// </remarks>
        public static async Task<string?> InvokeStableImageCoreAsync(string
 prompt, int seed)
        {
            string stableImageCoreModelId = "stability.stable-image-core-v1:1";

            AmazonBedrockRuntimeClient client = new(RegionEndpoint.USWest2);

            string payload = new JsonObject()
            {
                { "prompt", prompt },
                { "aspect_ratio", "1:1" },
                { "seed", seed },
                { "output_format", "png" }
            }.ToJsonString();

            try
            {
```

```
                InvokeModelResponse response = await client.InvokeModelAsync(new
 InvokeModelRequest()
                {
                    ModelId = stableImageCoreModelId,
                    Body = AWSSDKUtils.GenerateMemoryStreamFromString(payload),
                    ContentType = "application/json",
                    Accept = "application/json"
                });

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    var results = JsonNode.ParseAsync(response.Body).Result?
["images"]?.AsArray();

                    return results?[0]?.GetValue<string>();
                }
                else
                {
                    Console.WriteLine("InvokeModelAsync failed with status code "
 + response.HttpStatusCode);
                }
            }
            catch (AmazonBedrockRuntimeException e)
            {
                Console.WriteLine(e.Message);
            }
            return null;
        }
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Diffusion.

```
// Create an image with Stability AI Stable Image Core.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_WEST_2)
                .build();

        // Set the model ID, e.g., Stable Image Core.
        var modelId = "stability.stable-image-core-v1:1";

        // The InvokeModel API uses the model's native payload.
```

```
        // Learn more about the available inference parameters and response
 fields at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
diffusion-stable-image-core-text-image-request-response.html
        var nativeRequestTemplate = """
                {
                    "prompt": "{{prompt}}",
                    "aspect_ratio": "1:1",
                    "seed": {{seed}},
                    "output_format": "png"
                }""";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random seed for the image generation (max. 4,294,967,294).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        String nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/images/0")
                    .queryFrom(responseBody)
                    .toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
```

```
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }

}
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Diffusion.

```
    public function invokeStableDiffusion(string $prompt, int $seed = 0, string
 $aspect_ratio = '1:1')
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and available parameters of Stable Diffusion
 models refer to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
stability-diffusion.html

        $base64_image_data = "";
        try {
            $modelId = 'stability.stable-image-core-v1:1';
```

```
            $body = [
                'prompt' => $prompt,
                'aspect_ratio' => $aspect_ratio,
                'seed' => $seed,
                'output_format' => 'png',
            ];

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->images[0];
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- Para obter detalhes da API, consulte InvokeModela Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Diffusion.

```
# Use the native inference API to create an image with Stability AI Stable Image
 Core

import base64
import boto3
import json
```

```
import os
import random

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-west-2")

# Set the model ID, e.g., Stable Image Core.
model_id = "stability.stable-image-core-v1:1"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed.
seed = random.randint(0, 4294967295)

# Format the request payload using the model's native structure.
native_request = {
    "prompt": prompt,
    "aspect_ratio": "1:1",
    "seed": seed,
    "output_format": "png",
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"stability_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)
```

```
image_path = os.path.join(output_dir, f"stability_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Para obter detalhes da API, consulte a InvokeModelReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Diffusion.

```
    "Stable Image Core Input Parameters should be in a format like this:
*   {
*     "prompt": "Draw a dolphin with a mustache, photorealistic",
*     "aspect_ratio": "1:1",
*     "seed": 0,
*     "output_format": "png"
*   }
    DATA: BEGIN OF ls_input,
            prompt        TYPE /aws1/rt_shape_string,
            aspect_ratio  TYPE /aws1/rt_shape_string,
            seed          TYPE /aws1/rt_shape_integer,
            output_format TYPE /aws1/rt_shape_string,
          END OF ls_input.

    ls_input-prompt = iv_prompt.
    ls_input-aspect_ratio = '1:1'.
    ls_input-seed = 0. "or better, choose a random integer.
    ls_input-output_format = 'png'.
```

```
    DATA(lv_json) = /ui2/cl_json=>serialize(
      data = ls_input
                pretty_name   = /ui2/cl_json=>pretty_mode-low_case ).

    TRY.
        DATA(lo_response) = lo_bdr->invokemodel(
          iv_body = /aws1/cl_rt_util=>string_to_xstring( lv_json )
          iv_modelid = 'stability.stable-image-core-v1:1'
          iv_accept = 'application/json'
          iv_contenttype = 'application/json' ).

        "Stable Image Core Result Format:
*       {
*         "seeds": ["0"],
*         "finish_reasons": [null],
*         "images": ["iVBORw0KGgoAAAANSUhEUgAAAgAAA...."]
*       }
        DATA: BEGIN OF ls_response,
                images TYPE STANDARD TABLE OF /aws1/rt_shape_string,
              END OF ls_response.

        /ui2/cl_json=>deserialize(
          EXPORTING jsonx = lo_response->get_body( )
                    pretty_name = /ui2/cl_json=>pretty_mode-camel_case
          CHANGING  data  = ls_response ).
        IF ls_response-images IS NOT INITIAL.
          DATA(lv_image) =
 cl_http_utility=>if_http_utility~decode_x_base64( ls_response-images[ 1 ] ).
        ENDIF.
      CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

Invoque o modelo básico do Stability.ai Stable Image Core para gerar imagens usando o cliente de alto nível L2.

```
    TRY.
```

```
        DATA(lo_bdr_l2_sd) = /aws1/
cl_bdr_l2_factory=>create_stable_diffusion_xl_1( lo_bdr ).
        " iv_prompt contains a prompt like 'Show me a picture of a unicorn
 reading an enterprise financial report'.
        DATA(lv_image) = lo_bdr_l2_sd->text_to_image( iv_prompt ).
      CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

- Para obter detalhes da API, consulte a InvokeModelreferência da API AWS SDK for SAP ABAP.

Exemplos de código para Amazon Bedrock Agents usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Agents com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon Bedrock Agents: mais informações sobre o Amazon Bedrock Agents.
- Referência de API do Amazon Bedrock Agents: detalhes sobre todas as ações disponíveis do Amazon Bedrock Agents.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para Amazon Bedrock Agents usando AWS SDKs
  - Olá, Amazon Bedrock Agents
  - Ações para Amazon Bedrock Agents usando AWS SDKs
    - Use CreateAgentcom um AWS SDK
    - Use CreateAgentActionGroupcom um AWS SDK
    - Use CreateAgentAliascom um AWS SDK
    - Use CreateFlowcom um AWS SDK
    - Use CreateFlowAliascom um AWS SDK
    - Use CreateFlowVersioncom um AWS SDK
    - Use CreateKnowledgeBasecom um AWS SDK
    - Use CreatePromptcom um AWS SDK
    - Use CreatePromptVersioncom um AWS SDK
    - Use DeleteAgentcom um AWS SDK
    - Use DeleteAgentAliascom um AWS SDK
    - Use DeleteFlowcom um AWS SDK
    - Use DeleteFlowAliascom um AWS SDK
    - Use DeleteFlowVersioncom um AWS SDK
    - Use DeleteKnowledgeBasecom um AWS SDK
    - Use DeletePromptcom um AWS SDK
    - Use GetAgentcom um AWS SDK
    - Use GetFlowcom um AWS SDK
    - Use GetFlowVersioncom um AWS SDK
    - Use GetKnowledgeBasecom um AWS SDK
    - Use GetPromptcom um AWS SDK
    - Use ListAgentActionGroupscom um AWS SDK
    - Use ListAgentKnowledgeBasescom um AWS SDK
    - Use ListAgentscom um AWS SDK
    - Use ListFlowAliasescom um AWS SDK
    - Use ListFlowVersionscom um AWS SDK

- Use ListFlowscom um AWS SDK
- Use ListKnowledgeBasescom um AWS SDK
- Use ListPromptscom um AWS SDK
- Use PrepareAgentcom um AWS SDK
- Use PrepareFlowcom um AWS SDK
- Use UpdateFlowcom um AWS SDK
- Use UpdateFlowAliascom um AWS SDK
- Use UpdateKnowledgeBasecom um AWS SDK

- Cenários para Amazon Bedrock Agents usando AWS SDKs
  - Um exemplo completo que mostra como criar e invocar um fluxo do Amazon Bedrock usando um AWS SDK
  - Um exemplo completo que mostra como criar e invocar prompts gerenciados do Amazon Bedrock usando um AWS SDK
  - Um exemplo completo que mostra como criar e invocar Amazon Bedrock Agents usando um AWS SDK
  - Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

Exemplos básicos para Amazon Bedrock Agents usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Agents com SDKs da AWS .

Exemplos

- Olá, Amazon Bedrock Agents
- Ações para Amazon Bedrock Agents usando AWS SDKs
  - Use CreateAgentcom um AWS SDK
  - Use CreateAgentActionGroupcom um AWS SDK
  - Use CreateAgentAliascom um AWS SDK
  - Use CreateFlowcom um AWS SDK
  - Use CreateFlowAliascom um AWS SDK
  - Use CreateFlowVersioncom um AWS SDK
  - Use CreateKnowledgeBasecom um AWS SDK
  - Use CreatePromptcom um AWS SDK

- Use CreatePromptVersioncom um AWS SDK
- Use DeleteAgentcom um AWS SDK
- Use DeleteAgentAliascom um AWS SDK
- Use DeleteFlowcom um AWS SDK
- Use DeleteFlowAliascom um AWS SDK
- Use DeleteFlowVersioncom um AWS SDK
- Use DeleteKnowledgeBasecom um AWS SDK
- Use DeletePromptcom um AWS SDK
- Use GetAgentcom um AWS SDK
- Use GetFlowcom um AWS SDK
- Use GetFlowVersioncom um AWS SDK
- Use GetKnowledgeBasecom um AWS SDK
- Use GetPromptcom um AWS SDK
- Use ListAgentActionGroupscom um AWS SDK
- Use ListAgentKnowledgeBasescom um AWS SDK
- Use ListAgentscom um AWS SDK
- Use ListFlowAliasescom um AWS SDK
- Use ListFlowVersionscom um AWS SDK
- Use ListFlowscom um AWS SDK
- Use ListKnowledgeBasescom um AWS SDK
- Use ListPromptscom um AWS SDK
- Use PrepareAgentcom um AWS SDK
- Use PrepareFlowcom um AWS SDK
- Use UpdateFlowcom um AWS SDK
- Use UpdateFlowAliascom um AWS SDK
- Use UpdateKnowledgeBasecom um AWS SDK

Olá, Amazon Bedrock Agents

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock Agents.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentClient,
  GetAgentCommand,
  paginateListAgents,
} from "@aws-sdk/client-bedrock-agent";

/**
 * @typedef {Object} AgentSummary
 */

/**
 * A simple scenario to demonstrate basic setup and interaction with the Bedrock
 Agents Client.
 *
 * This function first initializes the Amazon Bedrock Agents client for a
 specific region.
 * It then retrieves a list of existing agents using the streamlined paginator
 approach.
 * For each agent found, it retrieves detailed information using a command
 object.
 *
 * Demonstrates:
 * - Use of the Bedrock Agents client to initialize and communicate with the AWS
 service.
 * - Listing resources in a paginated response pattern.
 * - Accessing an individual resource using a command object.
 *
 * @returns {Promise<void>} A promise that resolves when the function has
 completed execution.
```

```
 */
export const main = async () => {
  const region = "us-east-1";

  console.log("=".repeat(68));

  console.log(`Initializing Amazon Bedrock Agents client for ${region}...`);
  const client = new BedrockAgentClient({ region });

  console.log("Retrieving the list of existing agents...");
  const paginatorConfig = { client };
  const pages = paginateListAgents(paginatorConfig, {});

  /** @type {AgentSummary[]} */
  const agentSummaries = [];
  for await (const page of pages) {
    agentSummaries.push(...page.agentSummaries);
  }

  console.log(`Found ${agentSummaries.length} agents in ${region}.`);

  if (agentSummaries.length > 0) {
    for (const agentSummary of agentSummaries) {
      const agentId = agentSummary.agentId;
      console.log("=".repeat(68));
      console.log(`Retrieving agent with ID: ${agentId}:`);
      console.log("-".repeat(68));

      const command = new GetAgentCommand({ agentId });
      const response = await client.send(command);
      const agent = response.agent;

      console.log(` Name: ${agent.agentName}`);
      console.log(` Status: ${agent.agentStatus}`);
      console.log(` ARN: ${agent.agentArn}`);
      console.log(` Foundation model: ${agent.foundationModel}`);
    }
  }
  console.log("=".repeat(68));
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await main();
```

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - GetAgent
  - ListAgents

Ações para Amazon Bedrock Agents usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Bedrock Agents com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Bedrock Agents e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para Amazon Bedrock Agents usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon Bedrock Agents API Reference.

Exemplos

- Use CreateAgentcom um AWS SDK
- Use CreateAgentActionGroupcom um AWS SDK
- Use CreateAgentAliascom um AWS SDK
- Use CreateFlowcom um AWS SDK
- Use CreateFlowAliascom um AWS SDK
- Use CreateFlowVersioncom um AWS SDK
- Use CreateKnowledgeBasecom um AWS SDK
- Use CreatePromptcom um AWS SDK
- Use CreatePromptVersioncom um AWS SDK
- Use DeleteAgentcom um AWS SDK
- Use DeleteAgentAliascom um AWS SDK
- Use DeleteFlowcom um AWS SDK

- Use DeleteFlowAliascom um AWS SDK
- Use DeleteFlowVersioncom um AWS SDK
- Use DeleteKnowledgeBasecom um AWS SDK
- Use DeletePromptcom um AWS SDK
- Use GetAgentcom um AWS SDK
- Use GetFlowcom um AWS SDK
- Use GetFlowVersioncom um AWS SDK
- Use GetKnowledgeBasecom um AWS SDK
- Use GetPromptcom um AWS SDK
- Use ListAgentActionGroupscom um AWS SDK
- Use ListAgentKnowledgeBasescom um AWS SDK
- Use ListAgentscom um AWS SDK
- Use ListFlowAliasescom um AWS SDK
- Use ListFlowVersionscom um AWS SDK
- Use ListFlowscom um AWS SDK
- Use ListKnowledgeBasescom um AWS SDK
- Use ListPromptscom um AWS SDK
- Use PrepareAgentcom um AWS SDK
- Use PrepareFlowcom um AWS SDK
- Use UpdateFlowcom um AWS SDK
- Use UpdateFlowAliascom um AWS SDK
- Use UpdateKnowledgeBasecom um AWS SDK

Use **`CreateAgent`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateAgent`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um agente do .

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  CreateAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Creates an Amazon Bedrock Agent.
 *
 * @param {string} agentName - A name for the agent that you create.
 * @param {string} foundationModel - The foundation model to be used by the agent
 you create.
 * @param {string} agentResourceRoleArn - The ARN of the IAM role with
 permissions required by the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").Agent>} An object
 containing details of the created agent.
 */
export const createAgent = async (
  agentName,
  foundationModel,
  agentResourceRoleArn,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  const command = new CreateAgentCommand({
    agentName,
```

```
    foundationModel,
    agentResourceRoleArn,
  });
  const response = await client.send(command);

  return response.agent;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentName and accountId, and roleName with a
 unique name for the new agent,
  // the id of your AWS account, and the name of an existing execution role that
 the agent can use inside your account.
  // For foundationModel, specify the desired model. Ensure to remove the
 brackets '[]' before adding your data.

  // A string (max 100 chars) that can include letters, numbers, dashes '-', and
 underscores '_'.
  const agentName = "[your-bedrock-agent-name]";

  // Your AWS account id.
  const accountId = "[123456789012]";

  // The name of the agent's execution role. It must be prefixed by
 `AmazonBedrockExecutionRoleForAgents_`.
  const roleName = "[AmazonBedrockExecutionRoleForAgents_your-role-name]";

  // The ARN for the agent's execution role.
  // Follow the ARN format: 'arn:aws:iam::account-id:role/role-name'
  const roleArn = `arn:aws:iam::${accountId}:role/${roleName}`;

  // Specify the model for the agent. Change if a different model is preferred.
  const foundationModel = "anthropic.claude-v2";

  // Check for unresolved placeholders in agentName and roleArn.
  checkForPlaceholders([agentName, roleArn]);

  console.log("Creating a new agent...");

  const agent = await createAgent(agentName, foundationModel, roleArn);
  console.log(agent);
}
```

- Para obter detalhes da API, consulte CreateAgenta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um agente do .

```
    def create_agent(self, agent_name, foundation_model, role_arn, instruction):
        """
        Creates an agent that orchestrates interactions between foundation
models,
        data sources, software applications, user conversations, and APIs to
carry
        out tasks to help customers.

        :param agent_name: A name for the agent.
        :param foundation_model: The foundation model to be used for
orchestration by the agent.
        :param role_arn: The ARN of the IAM role with permissions needed by the
agent.
        :param instruction: Instructions that tell the agent what it should do
and how it should
                            interact with users.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """
        try:
            response = self.client.create_agent(
                agentName=agent_name,
                foundationModel=foundation_model,
                agentResourceRoleArn=role_arn,
                instruction=instruction,
```

```
            )
        except ClientError as e:
            logger.error(f"Error: Couldn't create agent. Here's why: {e}")
            raise
        else:
            return response["agent"]
```

- Para obter detalhes da API, consulte a CreateAgentReferência da API AWS SDK for Python (Boto3).

Use **`CreateAgentActionGroup`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateAgentActionGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um grupo de ação do agente.

```
    def create_agent_action_group(
            self, name, description, agent_id, agent_version, function_arn,
api_schema
    ):
        """
        Creates an action group for an agent. An action group defines a set of
actions that an
        agent should carry out for the customer.
```

```
        :param name: The name to give the action group.
        :param description: The description of the action group.
        :param agent_id: The unique identifier of the agent for which to create
the action group.
        :param agent_version: The version of the agent for which to create the
action group.
        :param function_arn: The ARN of the Lambda function containing the
business logic that is
                                carried out upon invoking the action.
        :param api_schema: Contains the OpenAPI schema for the action group.
        :return: Details about the action group that was created.
        """
        try:
            response = self.client.create_agent_action_group(
                actionGroupName=name,
                description=description,
                agentId=agent_id,
                agentVersion=agent_version,
                actionGroupExecutor={"lambda": function_arn},
                apiSchema={"payload": api_schema},
            )
            agent_action_group = response["agentActionGroup"]
        except ClientError as e:
            logger.error(f"Error: Couldn't create agent action group. Here's why:
{e}")
            raise
        else:
            return agent_action_group
```

- Para obter detalhes da API, consulte a CreateAgentActionGroupReferência da API AWS SDK for Python (Boto3).

Use **`CreateAgentAlias`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateAgentAlias`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um alias de agente.

```
    def create_agent_alias(self, name, agent_id):
        """
        Creates an alias of an agent that can be used to deploy the agent.

        :param name: The name of the alias.
        :param agent_id: The unique identifier of the agent.
        :return: Details about the alias that was created.
        """
        try:
            response = self.client.create_agent_alias(
                agentAliasName=name, agentId=agent_id
            )
            agent_alias = response["agentAlias"]
        except ClientError as e:
            logger.error(f"Couldn't create agent alias. {e}")
            raise
        else:
            return agent_alias
```

- Para obter detalhes da API, consulte a CreateAgentAliasReferência da API AWS SDK for Python (Boto3).

Use **`CreateFlow`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateFlow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um fluxo do Amazon Bedrock.

```
def create_flow(client, flow_name, flow_description, role_arn, flow_def):
    """
    Creates an Amazon Bedrock flow.

    Args:
    client: Amazon Bedrock agent boto3 client.
    flow_name (str): The name for the new flow.
    role_arn (str):  The ARN for the IAM role that use flow uses.
    flow_def (json): The JSON definition of the flow that you want to create.

    Returns:
        dict: The response from CreateFlow.
    """
    try:

        logger.info("Creating flow: %s.", flow_name)

        response = client.create_flow(
            name=flow_name,
            description=flow_description,
            executionRoleArn=role_arn,
            definition=flow_def
        )

        logger.info("Successfully created flow: %s. ID: %s",
                    flow_name,
                    {response['id']})
```

```
        return response

    except ClientError as e:
        logger.exception("Client error creating flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error creating flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a CreateFlowReferência da API AWS SDK for Python (Boto3).

Use **`CreateFlowAlias`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateFlowAlias`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um alias para um fluxo do Amazon Bedrock.

```
def create_flow_alias(client, flow_id, flow_version, name, description):
    """
    Creates an alias for an Amazon Bedrock flow.

    Args:
```

```
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        str: The ID for the flow alias.
    """

    try:
        logger.info("Creating flow alias for flow: %s.", flow_id)

        response = client.create_flow_alias(
            flowIdentifier=flow_id,
            name=name,
            description=description,
            routingConfiguration=[
                {
                    "flowVersion": flow_version
                }
            ]
        )
        logger.info("Successfully created flow alias for %s.", flow_id)

        return response['id']

    except ClientError as e:
        logging.exception("Client error creating alias for flow: %s - %s",
                flow_id, str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error creating alias for flow : %s - %s",
                flow_id, str(e))
        raise
```

- Para obter detalhes da API, consulte a CreateFlowAliasReferência da API AWS SDK for Python (Boto3).

Use **`CreateFlowVersion`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateFlowVersion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma versão de um fluxo do Amazon Bedrock.

```
def create_flow_version(client, flow_id, description):
    """
    Creates a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
        description (str) : A description for the flow.

    Returns:
        str: The version for the flow.
    """
    try:

        logger.info("Creating flow version for flow: %s.", flow_id)

        # Call CreateFlowVersion operation
        response = client.create_flow_version(
            flowIdentifier=flow_id,
            description=description
        )

        logging.info("Successfully created flow version %s for flow %s.",
            response['version'], flow_id)

        return response['version']

    except ClientError as e:
```

```
        logging.exception("Client error creating flow: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error creating flow : %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a CreateFlowVersionReferência da API AWS SDK for Python (Boto3).

Use **`CreateKnowledgeBase`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateKnowledgeBase`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma base de conhecimento do Amazon Bedrock.

```
def create_knowledge_base(bedrock_agent_client, name, role_arn,
 description=None):
    """
    Creates a new knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        name (str): The name of the knowledge base.
        role_arn (str): The ARN of the IAM role that the knowledge base assumes
 to access resources.
        description (str, optional): A description of the knowledge base.

    Returns:
        dict: The details of the created knowledge base.
    """
    try:
```

```
        kwargs = {
            "name": name,
            "roleArn": role_arn,
            "knowledgeBaseConfiguration": {
                "type": "VECTOR",
                "vectorKnowledgeBaseConfiguration": {
                    "embeddingModelArn": "arn:aws:bedrock:us-east-1::foundation-
model/amazon.titan-embed-text-v1"
                }
            },
            "storageConfiguration": {
                "type": "OPENSEARCH_SERVERLESS",
                # Note: You will need to create an OpenSearch Serverless
 collection first and replace this ARN
                # with your actual collection ARN from the OpenSearch console. If
 you use the console instead,
                # you can use the quick-create flow to have Knowledge Bases
 create the collection for you.
                "opensearchServerlessConfiguration": {
                    "collectionArn": "arn:aws:aoss:us-
east-1::123456789012:collection/abcdefgh12345678defgh",
                        "fieldMapping": {
                        "metadataField": "metadata",
                        "textField": "text",
                        "vectorField": "vector"
                        },
                    "vectorIndexName": "test-uuid"
                    },
                },
            "clientToken": "test-client-token-" + str(uuid.uuid4())
        }

        if description:
            kwargs["description"] = description

        response = bedrock_agent_client.create_knowledge_base(**kwargs)

        logger.info("Created knowledge base with ID: %s",
 response["knowledgeBase"]["knowledgeBaseId"])
        return response["knowledgeBase"]

    except ClientError as err:
        logger.error(
            "Couldn't create knowledge base. Here's why: %s: %s",
```

```
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a CreateKnowledgeBaseReferência da API AWS SDK for Python (Boto3).

Use **`CreatePrompt`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreatePrompt`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um prompt gerenciado

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um prompt gerenciado do Amazon Bedrock.

```
def create_prompt(client, prompt_name, prompt_description, prompt_template,
 model_id=None):
    """
    Creates an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_name (str): The name for the new prompt.
    prompt_description (str): The description for the new prompt.
    prompt_template (str): The template for the prompt.
    model_id (str, optional): The model ID to associate with the prompt.
```

```
    Returns:
        dict: The response from CreatePrompt.
    """
    try:
        logger.info("Creating prompt: %s.", prompt_name)

        # Create a variant with the template
        variant = {
            "name": "default",
            "templateType": "TEXT",
            "templateConfiguration": {
                "text": {
                    "text": prompt_template,
                    "inputVariables": []
                }
            }
        }

        # Extract input variables from the template
        # Look for patterns like {{variable_name}}

        variables = re.findall(r'{{(.*?)}}', prompt_template)
        for var in variables:
            variant["templateConfiguration"]["text"]
["inputVariables"].append({"name": var.strip()})

        # Add model ID if provided
        if model_id:
            variant["modelId"] = model_id

        # Create the prompt with the variant
        create_params = {
            'name': prompt_name,
            'description': prompt_description,
            'variants': [variant]
        }

        response = client.create_prompt(**create_params)

        logger.info("Successfully created prompt: %s. ID: %s",
                    prompt_name,
                    response['id'])
```

```
        return response

    except ClientError as e:
        logger.exception("Client error creating prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error creating prompt: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a CreatePromptReferência da API AWS SDK for Python (Boto3).

Use **`CreatePromptVersion`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreatePromptVersion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um prompt gerenciado

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma versão de um prompt gerenciado do Amazon Bedrock.

```
def create_prompt_version(client, prompt_id, description=None):
    """
    Creates a version of an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
```

```
    prompt_id (str): The identifier of the prompt to create a version for.
    description (str, optional): A description for the version.

    Returns:
        dict: The response from CreatePromptVersion.
    """
    try:
        logger.info("Creating version for prompt ID: %s.", prompt_id)

        create_params = {
            'promptIdentifier': prompt_id
        }

        if description:
            create_params['description'] = description

        response = client.create_prompt_version(**create_params)

        logger.info("Successfully created prompt version: %s",
response['version'])
        logger.info("Prompt version ARN: %s", response['arn'])

        return response


    except ClientError as e:
        logger.exception("Client error creating prompt version: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error creating prompt version: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a CreatePromptVersionReferência da API AWS SDK for Python (Boto3).

Use **`DeleteAgent`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteAgent`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um agente.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  DeleteAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Deletes an Amazon Bedrock Agent.
 *
 * @param {string} agentId - The unique identifier of the agent to delete.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-
agent").DeleteAgentCommandOutput>} An object containing the agent id, the status,
 and some additional metadata.
 */
export const deleteAgent = (agentId, region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });
  const command = new DeleteAgentCommand({ agentId });
  return client.send(command);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId with an existing agent's id.
  // Ensure to remove the brackets (`[]`) before adding your data.
```

```
  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // Check for unresolved placeholders in agentId.
  checkForPlaceholders([agentId]);

  console.log(`Deleting agent with ID ${agentId}...`);

  const response = await deleteAgent(agentId);
  console.log(response);
}
```

- Para obter detalhes da API, consulte DeleteAgenta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um agente.

```
    def delete_agent(self, agent_id):
        """
        Deletes an Amazon Bedrock agent.

        :param agent_id: The unique identifier of the agent to delete.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
 raises an exception.
        """

        try:
            response = self.client.delete_agent(
                agentId=agent_id, skipResourceInUseCheck=False
            )
```

```
        except ClientError as e:
            logger.error(f"Couldn't delete agent. {e}")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteAgentReferência da API AWS SDK for Python (Boto3).

Use **`DeleteAgentAlias`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteAgentAlias`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um alias de agente.

```
    def delete_agent_alias(self, agent_id, agent_alias_id):
        """
        Deletes an alias of an Amazon Bedrock agent.

        :param agent_id: The unique identifier of the agent that the alias
belongs to.
        :param agent_alias_id: The unique identifier of the alias to delete.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
```

```
        """

        try:
            response = self.client.delete_agent_alias(
                agentId=agent_id, agentAliasId=agent_alias_id
            )
        except ClientError as e:
            logger.error(f"Couldn't delete agent alias. {e}")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteAgentAliasReferência da API AWS SDK for Python (Boto3).

Use **`DeleteFlow`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteFlow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um fluxo do Amazon Bedrock.

```
def delete_flow(client, flow_id):
    """
```

```
    Deletes an Amazon Bedrock flow.

    Args:
    client: Amazon Bedrock agent boto3 client.
    flow_id (str): The identifier of the flow that you want to delete.

    Returns:
        dict: The response from the DeleteFLow operation.
    """
    try:

        logger.info("Deleting flow ID: %s.",
                    flow_id)

        # Call DeleteFlow operation
        response = client.delete_flow(
            flowIdentifier=flow_id,
            skipResourceInUseCheck=True
        )

        logger.info("Finished deleting flow ID: %s", flow_id)

        return response

    except ClientError as e:
        logger.exception("Client error deleting flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error deleting flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a DeleteFlowReferência da API AWS SDK for Python (Boto3).

Use **`DeleteFlowAlias`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteFlowAlias`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um alias de um fluxo do Amazon Bedrock.

```
def delete_flow_alias(client, flow_id, flow_alias_id):
    """
    Deletes an Amazon Bedrock flow alias.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from the call to DetectFLowAlias
    """
    try:

        logger.info("Deleting flow alias %s for flow: %s.", flow_alias_id,
 flow_id)

        # Delete the flow alias.
        response = client.delete_flow_alias(
            aliasIdentifier=flow_alias_id,
            flowIdentifier=flow_id
        )

        logging.info("Successfully deleted flow version for %s.", flow_id)
        return response

    except ClientError as e:
        logging.exception("Client error deleting flow version: %s", str(e))
        raise
```

```
    except Exception as e:
        logging.exception("Unexpected deleting flow version: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a DeleteFlowAliasReferência da API AWS SDK for Python (Boto3).

Use **`DeleteFlowVersion`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteFlowVersion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua uma versão de um fluxo do Amazon Bedrock.

```
def delete_flow_version(client, flow_id, flow_version):
    """
    Deletes a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from DeleteFlowVersion.
    """
    try:
```

```
        logger.info("Deleting flow version %s for flow: %s.",flow_version,
flow_id)

        # Call DeleteFlowVersion operation
        response = client.delete_flow_version(
            flowIdentifier=flow_id,
            flowVersion=flow_version
        )

        logging.info("Successfully deleted flow version %s for %s.",
                flow_version,
                flow_id)
        return response

    except ClientError as e:
        logging.exception("Client error deleting flow version: %s ", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected deleting flow version: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a DeleteFlowVersionReferência da API AWS SDK for Python (Boto3).

Use **`DeleteKnowledgeBase`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteKnowledgeBase`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma base de conhecimento do Amazon Bedrock.

```
def delete_knowledge_base(bedrock_agent_client, knowledge_base_id):
    """
    Deletes a knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base to delete.

    Returns:
        bool: True if the deletion was successful.
    """
    try:
        bedrock_agent_client.delete_knowledge_base(
            knowledgeBaseId=knowledge_base_id
        )

        logger.info("Deleted knowledge base: %s", knowledge_base_id)
        return True
    except ClientError as err:
        logger.error(
            "Couldn't delete knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a DeleteKnowledgeBaseReferência da API AWS SDK for Python (Boto3).

Use **`DeletePrompt`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeletePrompt`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um prompt gerenciado

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um prompt gerenciado do Amazon Bedrock.

```
def delete_prompt(client, prompt_id):
    """
    Deletes an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt that you want to delete.

    Returns:
        dict: The response from the DeletePrompt operation.
    """
    try:
        logger.info("Deleting prompt ID: %s.", prompt_id)

        response = client.delete_prompt(
            promptIdentifier=prompt_id
        )

        logger.info("Finished deleting prompt ID: %s", prompt_id)

        return response

    except ClientError as e:
        logger.exception("Client error deleting prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error deleting prompt: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a DeletePromptReferência da API AWS SDK for Python (Boto3).

Use **`GetAgent`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetAgent`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um agente.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  GetAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves the details of an Amazon Bedrock Agent.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").Agent>} An object
 containing the agent details.
 */
```

```
export const getAgent = async (agentId, region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  const command = new GetAgentCommand({ agentId });
  const response = await client.send(command);
  return response.agent;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId with an existing agent's id.
  // Ensure to remove the brackets '[]' before adding your data.

  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // Check for unresolved placeholders in agentId.
  checkForPlaceholders([agentId]);

  console.log(`Retrieving agent with ID ${agentId}...`);

  const agent = await getAgent(agentId);
  console.log(agent);
}
```

- Para obter detalhes da API, consulte GetAgenta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um agente.

```
    def get_agent(self, agent_id, log_error=True):
```

```
        """
        Gets information about an agent.

        :param agent_id: The unique identifier of the agent.
        :param log_error: Whether to log any errors that occur when getting the
agent.
                          If True, errors will be logged to the logger. If False,
errors
                          will still be raised, but not logged.
        :return: The information about the requested agent.
        """

        try:
            response = self.client.get_agent(agentId=agent_id)
            agent = response["agent"]
        except ClientError as e:
            if log_error:
                logger.error(f"Couldn't get agent {agent_id}. {e}")
            raise
        else:
            return agent
```

- Para obter detalhes da API, consulte a GetAgentReferência da API AWS SDK for Python (Boto3).

Use **`GetFlow`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetFlow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um fluxo do Amazon Bedrock.

```
def get_flow(client, flow_id):
    """
    Gets an Amazon Bedrock flow.

    Args:
    client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow that you want to get.

    Returns:
        dict: The response from the GetFlow operation.
    """
    try:

        logger.info("Getting flow ID: %s.",
                    flow_id)

        # Call GetFlow operation.
        response = client.get_flow(
            flowIdentifier=flow_id
        )

        logger.info("Retrieved flow ID: %s. Name: %s", flow_id,
                    response['name'])

        return response

    except ClientError as e:
        logger.exception("Client error getting flow: %s", {str(e)})
        raise

    except Exception as e:
```

```
        logger.exception("Unexepcted error getting flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a GetFlowReferência da API AWS SDK for Python (Boto3).

Use **`GetFlowVersion`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetFlowVersion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha uma versão de um fluxo do Amazon Bedrock.

```
def get_flow_version(client, flow_id, flow_version):
    """
    Gets information about a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
        flow_version (str): The flow version of the flow.

    Returns:
        dict: The response from the call to GetFlowVersion.
    """
```

```
    try:

        logger.info("Deleting flow version for flow: %s.", flow_id)

        # Call GetFlowVersion operation
        response = client.get_flow_version(
            flowIdentifier=flow_id,
            flowVersion=flow_version
        )

        logging.info("Successfully got flow version %s information for flow %s.",
                    flow_version,
                    flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error getting flow version: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error getting flow version: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a GetFlowVersionReferência da API AWS SDK for Python (Boto3).

Use **`GetKnowledgeBase`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetKnowledgeBase`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha uma base de conhecimento do Amazon Bedrock.

```
def get_knowledge_base(bedrock_agent_client, knowledge_base_id):
    """
    Gets details about a specific knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base.

    Returns:
        dict: The details of the knowledge base.
    """
    try:
        response = bedrock_agent_client.get_knowledge_base(
            knowledgeBaseId=knowledge_base_id
        )

        logger.info("Retrieved knowledge base: %s", knowledge_base_id)
        return response["knowledgeBase"]
    except ClientError as err:
        logger.error(
            "Couldn't get knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a GetKnowledgeBaseReferência da API AWS SDK for Python (Boto3).

Use **`GetPrompt`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetPrompt`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um prompt gerenciado do Amazon Bedrock.

```
def get_prompt(client, prompt_id):
    """
    Gets an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt that you want to get.

    Returns:
        dict: The response from the GetPrompt operation.
    """
    try:
        logger.info("Getting prompt ID: %s.", prompt_id)

        response = client.get_prompt(
            promptIdentifier=prompt_id
        )

        logger.info("Retrieved prompt ID: %s. Name: %s",
                    prompt_id,
                    response['name'])

        return response

    except ClientError as e:
        logger.exception("Client error getting prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error getting prompt: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a GetPromptReferência da API AWS SDK for Python (Boto3).

Use **`ListAgentActionGroups`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListAgentActionGroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os grupos de ação de um agente.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  ListAgentActionGroupsCommand,
  paginateListAgentActionGroups,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves a list of Action Groups of an agent utilizing the paginator
 function.
 *
```

```
 * This function leverages a paginator, which abstracts the complexity of
 pagination, providing
 * a straightforward way to handle paginated results inside a `for await...of`
 loop.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} agentVersion - The version of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<ActionGroupSummary[]>} An array of action group summaries.
 */
export const listAgentActionGroupsWithPaginator = async (
  agentId,
  agentVersion,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  // Create a paginator configuration
  const paginatorConfig = {
    client,
    pageSize: 10, // optional, added for demonstration purposes
  };

  const params = { agentId, agentVersion };

  const pages = paginateListAgentActionGroups(paginatorConfig, params);

  // Paginate until there are no more results
  const actionGroupSummaries = [];
  for await (const page of pages) {
    actionGroupSummaries.push(...page.actionGroupSummaries);
  }

  return actionGroupSummaries;
};

/**
 * Retrieves a list of Action Groups of an agent utilizing the
 ListAgentActionGroupsCommand.
 *
 * This function demonstrates the manual approach, sending a command to the
 client and processing the response.
 * Pagination must manually be managed. For a simplified approach that abstracts
 away pagination logic, see
```

```
 * the `listAgentActionGroupsWithPaginator()` example below.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} agentVersion - The version of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<ActionGroupSummary[]>} An array of action group summaries.
 */
export const listAgentActionGroupsWithCommandObject = async (
  agentId,
  agentVersion,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  let nextToken;
  const actionGroupSummaries = [];
  do {
    const command = new ListAgentActionGroupsCommand({
      agentId,
      agentVersion,
      nextToken,
      maxResults: 10, // optional, added for demonstration purposes
    });

    /** @type {{actionGroupSummaries: ActionGroupSummary[], nextToken?: string}}
 */
    const response = await client.send(command);

    for (const actionGroup of response.actionGroupSummaries || []) {
      actionGroupSummaries.push(actionGroup);
    }

    nextToken = response.nextToken;
  } while (nextToken);

  return actionGroupSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId and agentVersion with an existing
 agent's id and version.
  // Ensure to remove the brackets '[]' before adding your data.
```

```
  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // A string either containing `DRAFT` or a number with 1-5 digits (e.g., '123'
 or 'DRAFT').
  const agentVersion = "[DRAFT]";

  // Check for unresolved placeholders in agentId and agentVersion.
  checkForPlaceholders([agentId, agentVersion]);

  console.log("=".repeat(68));
  console.log(
    "Listing agent action groups using ListAgentActionGroupsCommand:",
  );

  for (const actionGroup of await listAgentActionGroupsWithCommandObject(
    agentId,
    agentVersion,
  )) {
    console.log(actionGroup);
  }

  console.log("=".repeat(68));
  console.log(
    "Listing agent action groups using the paginateListAgents function:",
  );
  for (const actionGroup of await listAgentActionGroupsWithPaginator(
    agentId,
    agentVersion,
  )) {
    console.log(actionGroup);
  }
}
```

- Para obter detalhes da API, consulte ListAgentActionGroupsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os grupos de ação de um agente.

```
    def list_agent_action_groups(self, agent_id, agent_version):
        """
        List the action groups for a version of an Amazon Bedrock Agent.

        :param agent_id: The unique identifier of the agent.
        :param agent_version: The version of the agent.
        :return: The list of action group summaries for the version of the agent.
        """

        try:
            action_groups = []

            paginator = self.client.get_paginator("list_agent_action_groups")
            for page in paginator.paginate(
                    agentId=agent_id,
                    agentVersion=agent_version,
                    PaginationConfig={"PageSize": 10},
            ):
                action_groups.extend(page["actionGroupSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list action groups. {e}")
            raise
        else:
            return action_groups
```

- Para obter detalhes da API, consulte a ListAgentActionGroupsReferência da API AWS SDK for Python (Boto3).

Use **`ListAgentKnowledgeBases`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListAgentKnowledgeBases`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as bases de conhecimento associadas a um agente.

```
    def list_agent_knowledge_bases(self, agent_id, agent_version):
        """
        List the knowledge bases associated with a version of an Amazon Bedrock
Agent.

        :param agent_id: The unique identifier of the agent.
        :param agent_version: The version of the agent.
        :return: The list of knowledge base summaries for the version of the
agent.
        """

        try:
            knowledge_bases = []

            paginator = self.client.get_paginator("list_agent_knowledge_bases")
            for page in paginator.paginate(
                    agentId=agent_id,
                    agentVersion=agent_version,
                    PaginationConfig={"PageSize": 10},
            ):
                knowledge_bases.extend(page["agentKnowledgeBaseSummaries"])
```

```
        except ClientError as e:
            logger.error(f"Couldn't list knowledge bases. {e}")
            raise
        else:
            return knowledge_bases
```

- Para obter detalhes da API, consulte a ListAgentKnowledgeBasesReferência da API AWS SDK for Python (Boto3).

Use **`ListAgents`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListAgents`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os agentes que pertencem a uma conta.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentClient,
  ListAgentsCommand,
  paginateListAgents,
} from "@aws-sdk/client-bedrock-agent";
```

```
/**
 * Retrieves a list of available Amazon Bedrock agents utilizing the paginator
 function.
 *
 * This function leverages a paginator, which abstracts the complexity of
 pagination, providing
 * a straightforward way to handle paginated results inside a `for await...of`
 loop.
 *
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<AgentSummary[]>} An array of agent summaries.
 */
export const listAgentsWithPaginator = async (region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  const paginatorConfig = {
    client,
    pageSize: 10, // optional, added for demonstration purposes
  };

  const pages = paginateListAgents(paginatorConfig, {});

  // Paginate until there are no more results
  const agentSummaries = [];
  for await (const page of pages) {
    agentSummaries.push(...page.agentSummaries);
  }

  return agentSummaries;
};

/**
 * Retrieves a list of available Amazon Bedrock agents utilizing the
 ListAgentsCommand.
 *
 * This function demonstrates the manual approach, sending a command to the
 client and processing the response.
 * Pagination must manually be managed. For a simplified approach that abstracts
 away pagination logic, see
 * the `listAgentsWithPaginator()` example below.
 *
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<AgentSummary[]>} An array of agent summaries.
```

```
 */
export const listAgentsWithCommandObject = async (region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  let nextToken;
  const agentSummaries = [];
  do {
    const command = new ListAgentsCommand({
      nextToken,
      maxResults: 10, // optional, added for demonstration purposes
    });

    /** @type {{agentSummaries: AgentSummary[], nextToken?: string}} */
    const paginatedResponse = await client.send(command);

    agentSummaries.push(...(paginatedResponse.agentSummaries || []));

    nextToken = paginatedResponse.nextToken;
  } while (nextToken);

  return agentSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  console.log("=".repeat(68));
  console.log("Listing agents using ListAgentsCommand:");
  for (const agent of await listAgentsWithCommandObject()) {
    console.log(agent);
  }

  console.log("=".repeat(68));
  console.log("Listing agents using the paginateListAgents function:");
  for (const agent of await listAgentsWithPaginator()) {
    console.log(agent);
  }
}
```

- Para obter detalhes da API, consulte ListAgentsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os agentes que pertencem a uma conta.

```
    def list_agents(self):
        """
        List the available Amazon Bedrock Agents.

        :return: The list of available bedrock agents.
        """

        try:
            all_agents = []

            paginator = self.client.get_paginator("list_agents")
            for page in paginator.paginate(PaginationConfig={"PageSize": 10}):
                all_agents.extend(page["agentSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list agents. {e}")
            raise
        else:
            return all_agents
```

- Para obter detalhes da API, consulte a ListAgentsReferência da API AWS SDK for Python (Boto3).

Use **`ListFlowAliases`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListFlowAliases`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os aliases de um fluxo do Amazon Bedrock.

```
def list_flow_aliases(client, flow_id):
    """
    Lists the aliases of an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from ListFlowAliases.
    """
    try:

        finished = False

        logger.info("Listing flow aliases for flow: %s.", flow_id)

        print(f"Aliases for flow: {flow_id}")

        response = client.list_flow_aliases(
            flowIdentifier=flow_id,
            maxResults=10)

        while finished is False:

            for alias in response['flowAliasSummaries']:
                print(f"Alias Name: {alias['name']}")
                print(f"ID: {alias['id']}")
                print(f"Description: {alias.get('description', 'No
 description')}\n")
```

```
                if 'nextToken' in response:
                    next_token = response['nextToken']
                    response = client.list_flow_aliases(maxResults=10,
                                                        nextToken=next_token)
                else:
                    finished = True

        logging.info("Successfully listed flow aliases for flow %s.",
                flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error listing flow aliases: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow aliases: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a ListFlowAliasesReferência da API AWS SDK for Python (Boto3).

Use **`ListFlowVersions`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListFlowVersions`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as versões de um fluxo do Amazon Bedrock.

```
def list_flow_versions(client, flow_id):
    """
    Lists the versions of an Amazon Bedrock flow.
```

```
    Args:
        client: Amazon bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from ListFlowVersions.
    """
    try:

        finished = False

        logger.info("Listing flow versions for flow: %s.", flow_id)

        response = client.list_flow_versions(
            flowIdentifier=flow_id,
            maxResults=10)

        while finished is False:

            print(f"Versions for flow:{flow_id}")
            for version in response['flowVersionSummaries']:
                print(f"Version: {version['version']}")
                print(f"Status: {version['status']}\n")

                if 'nextToken' in response:
                    next_token = response['nextToken']
                    response = client.list_flow_versions(maxResults=10,
                                                         nextToken=next_token)
                else:
                    finished = True


        logging.info("Successfully listed flow versions for flow %s.",
                flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error listing flow versions: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow versions: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a ListFlowVersionsReferência da API AWS SDK for Python (Boto3).

Use **`ListFlows`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListFlows`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os fluxos do Amazon Bedrock.

```
def list_flows(client):
    """
    Lists versions of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        Nothing.
    """
    try:
        finished = False

        logger.info("Listing flows:")

        response = client.list_flows(maxResults=10)

        while finished is False:
```

```
            for flow in response['flowSummaries']:
                print(f"ID: {flow['id']}")
                print(f"Name: {flow['name']}")
                print(
                    f"Description: {flow.get('description', 'No description')}")
                print(f"Latest version: {flow['version']}")
                print(f"Status: {flow['status']}\n")

            if 'nextToken' in response:
                next_token = response['nextToken']
                response = client.list_flows(maxResults=10,
                                             nextToken=next_token)
            else:
                finished = True

        logging.info("Successfully listed flows.")


    except ClientError as e:
        logging.exception("Client error listing flow versions: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow versions: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a ListFlowsReferência da API AWS SDK for Python (Boto3).

Use **`ListKnowledgeBases`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListKnowledgeBases`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as bases de conhecimento do Amazon Bedrock.

```
def list_knowledge_bases(bedrock_agent_client, max_results=None):
    """
    Lists the knowledge bases in your AWS account.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        max_results (int, optional): The maximum number of knowledge bases to
 return.

    Returns:
        list: A list of knowledge base details.
    """
    try:
        kwargs = {}
        if max_results is not None:
            kwargs["maxResults"] = max_results

        # Initialize an empty list to store all knowledge bases
        all_knowledge_bases = []

        # Use paginator to handle pagination automatically
        paginator = bedrock_agent_client.get_paginator('list_knowledge_bases')
        page_iterator = paginator.paginate(**kwargs)

        # Iterate through each page of results
        for page in page_iterator:
            all_knowledge_bases.extend(page.get('knowledgeBaseSummaries', []))

        logger.info("Found %s knowledge bases.", len(all_knowledge_bases))
        return all_knowledge_bases
    except ClientError as err:
        logger.error(
            "Couldn't list knowledge bases. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a ListKnowledgeBasesReferência da API AWS SDK for Python (Boto3).

Use **`ListPrompts`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListPrompts`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os prompts gerenciados do Amazon Bedrock.

```
def list_prompts(client, max_results=10):
    """
    Lists Amazon Bedrock managed prompts.

    Args:
        client: Amazon Bedrock Agent boto3 client.
        max_results (int): Maximum number of results to return per page.

    Returns:
        list: A list of prompt summaries.
    """
    try:
        logger.info("Listing prompts:")

        # Create a paginator for the list_prompts operation
        paginator = client.get_paginator('list_prompts')

        # Create the pagination parameters
        pagination_config = {
            'maxResults': max_results
        }

        # Initialize an empty list to store all prompts
        all_prompts = []

        # Iterate through all pages
```

```
        for page in paginator.paginate(**pagination_config):
            all_prompts.extend(page.get('promptSummaries', []))

        logger.info("Successfully listed %s prompts.", len(all_prompts))
        return all_prompts

    except ClientError as e:
        logger.exception("Client error listing prompts: %s", str(e))
        raise
    except Exception as e:
        logger.exception("Unexpected error listing prompts: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a ListPromptsReferência da API AWS SDK for Python (Boto3).

Use **`PrepareAgent`**com um AWS SDK

O código de exemplo a seguir mostra como usar `PrepareAgent`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um agente

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Prepare um agente para testes internos.

```
    def prepare_agent(self, agent_id):
```

```
        """
        Creates a DRAFT version of the agent that can be used for internal
testing.

        :param agent_id: The unique identifier of the agent to prepare.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """
        try:
            prepared_agent_details = self.client.prepare_agent(agentId=agent_id)
        except ClientError as e:
            logger.error(f"Couldn't prepare agent. {e}")
            raise
        else:
            return prepared_agent_details
```

- Para obter detalhes da API, consulte a PrepareAgentReferência da API AWS SDK for Python (Boto3).

Use **`PrepareFlow`**com um AWS SDK

O código de exemplo a seguir mostra como usar `PrepareFlow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e invocar um fluxo

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Prepare um fluxo do Amazon Bedrock.

```
def prepare_flow(client, flow_id):
    """
    Prepares an Amazon Bedrock Flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow that you want to prepare.

    Returns:
        str: The status of the flow preparation
    """
    try:

        # Prepare the flow.
        logger.info("Preparing flow ID: %s",
                    flow_id)

        response = client.prepare_flow(
            flowIdentifier=flow_id
        )

        status = response.get('status')

        while status == "Preparing":
            logger.info("Preparing flow ID: %s. Status %s",
                        flow_id, status)

            sleep(5)
            response = client.get_flow(
                flowIdentifier=flow_id
            )
            status = response.get('status')
            print(f"Flow Status: {status}")

        if status == "Prepared":
            logger.info("Finished preparing flow ID: %s. Status %s",
                        flow_id, status)
        else:
            logger.warning("flow ID: %s not prepared. Status %s",
                           flow_id, status)

        return status
```

```
    except ClientError as e:
        logger.exception("Client error preparing flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error preparing flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a PrepareFlowReferência da API AWS SDK for Python (Boto3).

Use **`UpdateFlow`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UpdateFlow`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize um fluxo do Amazon Bedrock.

```
def update_flow(client, flow_id, flow_name, flow_description, role_arn,
 flow_def):
    """
    Updates an Amazon Bedrock flow.

    Args:
    client: bedrock agent boto3 client.
    flow_id (str): The ID for the flow that you want to update.
    flow_name (str): The name for the flow.
    role_arn (str):  The ARN for the IAM role that use flow uses.
    flow_def (json): The JSON definition of the flow that you want to create.

    Returns:
```

```
        dict: Flow information if successful.
    """
    try:

        logger.info("Updating flow: %s.", flow_id)

        response = client.update_flow(
            flowIdentifier=flow_id,
            name=flow_name,
            description=flow_description,
            executionRoleArn=role_arn,
            definition=flow_def
        )

        logger.info("Successfully updated flow: %s. ID: %s",
                    flow_name,
                    {response['id']})

        return response

    except ClientError as e:
        logger.exception("Client error updating flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error updating flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a UpdateFlowReferência da API AWS SDK for Python (Boto3).

Use **`UpdateFlowAlias`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UpdateFlowAlias`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize um alias para um fluxo do Amazon Bedrock.

```
def update_flow_alias(client, flow_id, alias_id, flow_version, name,
 description):
    """
    Updates an alias for an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        str: The response from UpdateFlowAlias.
    """

    try:
        logger.info("Updating flow alias %s for flow: %s.", alias_id, flow_id)

        response = client.update_flow_alias(
            aliasIdentifier=alias_id,
            flowIdentifier=flow_id,
            name=name,
            description=description,
            routingConfiguration=[
                {
                    "flowVersion": flow_version
                }
            ]
        )
        logger.info("Successfully updated flow alias %s for %s.", alias_id,
 flow_id)

        return response
```

```
    except ClientError as e:
        logging.exception("Client error updating alias %s for flow: %s - %s",
                alias_id, flow_id, str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error updating alias %s for flow : %s -
 %s",
                alias_id, flow_id, str(e))
        raise
```

- Para obter detalhes da API, consulte a UpdateFlowAliasReferência da API AWS SDK for Python (Boto3).

Use **`UpdateKnowledgeBase`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UpdateKnowledgeBase`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize uma base de conhecimento do Amazon Bedrock.

```
def update_knowledge_base(bedrock_agent_client, knowledge_base_id, name=None,
 description=None, role_arn=None):
    """
    Updates an existing knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base to update.
        name (str, optional): The new name for the knowledge base.
        description (str, optional): The new description for the knowledge base.
```

```
        role_arn (str, optional): The new IAM role ARN for the knowledge base.

    Returns:
        dict: The details of the updated knowledge base.
    """
    try:
        kwargs = {
            "knowledgeBaseId": knowledge_base_id,
            "knowledgeBaseConfiguration": {
                "type": "VECTOR",
                "vectorKnowledgeBaseConfiguration": {
                    "embeddingModelArn": "arn:aws:bedrock:us-east-1::foundation-
model/amazon.titan-embed-text-v1"
                }
            }
        }

        if name:
            kwargs["name"] = name
        if description:
            kwargs["description"] = description
        if role_arn:
            kwargs["roleArn"] = role_arn

        response = bedrock_agent_client.update_knowledge_base(**kwargs)

        logger.info("Updated knowledge base: %s", knowledge_base_id)
        return response["knowledgeBase"]

    except ClientError as err:
        logger.error(
            "Couldn't update knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a UpdateKnowledgeBaseReferência da API AWS SDK for Python (Boto3).

Cenários para Amazon Bedrock Agents usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Bedrock Agents com AWS SDKs. Esses cenários mostram como executar tarefas específicas chamando vários perfis no Amazon Bedrock Agents ou outros Serviços da AWS em conjunto. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Um exemplo completo que mostra como criar e invocar um fluxo do Amazon Bedrock usando um AWS SDK
- Um exemplo completo que mostra como criar e invocar prompts gerenciados do Amazon Bedrock usando um AWS SDK
- Um exemplo completo que mostra como criar e invocar Amazon Bedrock Agents usando um AWS SDK
- Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

Um exemplo completo que mostra como criar e invocar um fluxo do Amazon Bedrock usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um perfil de execução para o fluxo.
- Criar o fluxo.
- Implantar o fluxo totalmente configurado.
- Invocar o fluxo com prompts fornecidos pelo usuário.
- Exclua todos os recursos criados.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gera uma playlist de música com base no gênero e no número de músicas especificados pelo usuário.

```
from datetime import datetime
import logging
import boto3

from botocore.exceptions import ClientError

from roles import create_flow_role, delete_flow_role, update_role_policy
from flow import create_flow, prepare_flow, delete_flow
from run_flow import run_playlist_flow
from flow_version import create_flow_version, delete_flow_version
from flow_alias import create_flow_alias, delete_flow_alias

logging.basicConfig(
    level=logging.INFO
)
logger = logging.getLogger(__name__)

def create_input_node(name):
    """
    Creates an input node configuration for an Amazon Bedrock flow.

    The input node serves as the entry point for the flow and defines
    the initial document structure that will be passed to subsequent nodes.

    Args:
        name (str): The name of the input node.

    Returns:
        dict: The input node configuration.
```

```
    """
    return {
        "type": "Input",
        "name": name,
        "outputs": [
            {
                "name": "document",
                "type": "Object"
            }
        ]
    }


def create_prompt_node(name, model_id):
    """
    Creates a prompt node configuration for a Bedrock flow that generates music
 playlists.

    The prompt node defines an inline prompt template that creates a music
 playlist based on
    a specified genre and number of songs. The prompt uses two variables that are
 mapped from
    the input JSON object:
    - {{genre}}: The genre of music to create a playlist for
    - {{number}}: The number of songs to include in the playlist

    Args:
        name (str): The name of the prompt node.
        model_id (str): The identifier of the foundation model to use for the
 prompt.

    Returns:
        dict: The prompt node.

    """

    return {
        "type": "Prompt",
        "name": name,
        "configuration": {
            "prompt": {
                "sourceConfiguration": {
                    "inline": {
```

```
                        "modelId": model_id,
                        "templateType": "TEXT",
                        "inferenceConfiguration": {
                            "text": {
                                "temperature": 0.8
                            }
                        },
                        "templateConfiguration": {
                            "text": {
                                "text": "Make me a {{genre}} playlist consisting
 of the following number of songs: {{number}}."
                            }
                        }
                    }
                }
            }
        },
        "inputs": [
            {
                "name": "genre",
                "type": "String",
                "expression": "$.data.genre"
            },
            {
                "name": "number",
                "type": "Number",
                "expression": "$.data.number"
            }
        ],
        "outputs": [
            {
                "name": "modelCompletion",
                "type": "String"
            }
        ]
    }


def create_output_node(name):
    """
    Creates an output node configuration for a Bedrock flow.

    The output node validates that the output from the last node is a string
    and returns it unmodified. The input name must be "document".
```

```
    Args:
        name (str): The name of the output node.

    Returns:
        dict: The output node configuration containing the output node:

    """

    return {
        "type": "Output",
        "name": name,
        "inputs": [
            {
                "name": "document",
                "type": "String",
                "expression": "$.data"
            }
        ]
    }




def create_playlist_flow(client, flow_name, flow_description, role_arn,
 prompt_model_id):
    """
    Creates the playlist generator flow.
    Args:
        client: bedrock agent boto3 client.
        role_arn (str): Name for the new IAM role.
        prompt_model_id (str): The id of the model to use in the prompt node.
    Returns:
        dict: The response from the create_flow operation.
    """

    input_node = create_input_node("FlowInput")
    prompt_node = create_prompt_node("MakePlaylist", prompt_model_id)
    output_node = create_output_node("FlowOutput")

    # Create connections between the nodes
    connections = []

    #  First, create connections between the output of the flow
```

```
    # input node and each input of the prompt node.
    for prompt_node_input in prompt_node["inputs"]:
        connections.append(
            {
                "name": "_".join([input_node["name"], prompt_node["name"],
                                  prompt_node_input["name"]]),
                "source": input_node["name"],
                "target": prompt_node["name"],
                "type": "Data",
                "configuration": {
                    "data": {
                        "sourceOutput": input_node["outputs"][0]["name"],
                        "targetInput": prompt_node_input["name"]
                    }
                }
            }
        )

    # Then, create a connection between the output of the prompt node and the
input of the flow output node
    connections.append(
        {
            "name": "_".join([prompt_node["name"], output_node["name"]]),
            "source": prompt_node["name"],
            "target": output_node["name"],
            "type": "Data",
            "configuration": {
                "data": {
                    "sourceOutput": prompt_node["outputs"][0]["name"],
                    "targetInput": output_node["inputs"][0]["name"]
                }
            }
        }
    )

    flow_def = {
        "nodes": [input_node, prompt_node, output_node],
        "connections": connections
    }

    # Create the flow.

    response = create_flow(
        client, flow_name, flow_description, role_arn, flow_def)
```

```
    return response



def get_model_arn(client, model_id):
    """
    Gets the Amazon Resource Name (ARN) for a model.
    Args:
        client (str): Amazon Bedrock boto3 client.
        model_id (str): The id of the model.
    Returns:
        str: The ARN of the model.
    """

    try:
        # Call GetFoundationModelDetails operation
        response = client.get_foundation_model(modelIdentifier=model_id)

        # Extract model ARN from the response
        model_arn = response['modelDetails']['modelArn']

        return model_arn

    except ClientError as e:
        logger.exception("Client error getting model ARN: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexpected error getting model ARN: %s", {str(e)})
        raise


def prepare_flow_version_and_alias(bedrock_agent_client,
                                   flow_id):
    """
    Prepares the flow and then creates a flow version and flow alias.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        flowd_id (str): The ID of the flow that you want to prepare.
    Returns: The flow_version and flow_alias.

    """
```

```
    status = prepare_flow(bedrock_agent_client, flow_id)

    flow_version = None
    flow_alias = None

    if status == 'Prepared':

        # Create the flow version and alias.
        flow_version = create_flow_version(bedrock_agent_client,
                                           flow_id,
                                           f"flow version for flow {flow_id}.")

        flow_alias = create_flow_alias(bedrock_agent_client,
                                       flow_id,
                                       flow_version,
                                       "latest",
                                       f"Alias for flow {flow_id}, version
 {flow_version}")

    return flow_version, flow_alias



def delete_role_resources(bedrock_agent_client,
                          iam_client,
                          role_name,
                          flow_id,
                          flow_version,
                          flow_alias):
    """
    Deletes the flow, flow alias, flow version, and IAM roles.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        iam_client: Amazon IAM boto3 client.
        role_name (str): The name of the IAM role.
        flow_id (str): The id of the flow.
        flow_version (str): The version of the flow.
        flow_alias (str): The alias of the flow.
    """

    if flow_id is not None:
        if flow_alias is not None:
            delete_flow_alias(bedrock_agent_client, flow_id, flow_alias)
        if flow_version is not None:
```

```
            delete_flow_version(bedrock_agent_client,
                        flow_id, flow_version)
        delete_flow(bedrock_agent_client, flow_id)

    if role_name is not None:
        delete_flow_role(iam_client, role_name)



def main():
    """
    Creates, runs, and optionally deletes a Bedrock flow for generating music
 playlists.

    Note:
        Requires valid AWS credentials in the default profile
    """

    delete_choice = "y"
    try:

        # Get various boto3 clients.
        session = boto3.Session(profile_name='default')
        bedrock_agent_runtime_client = session.client('bedrock-agent-runtime')
        bedrock_agent_client = session.client('bedrock-agent')
        bedrock_client = session.client('bedrock')
        iam_client = session.client('iam')

        role_name = None
        flow_id = None
        flow_version = None
        flow_alias = None

        #Change the model as needed.
        prompt_model_id = "amazon.nova-pro-v1:0"

        # Base the flow name on the current date and time
        current_time = datetime.now()
        timestamp = current_time.strftime("%Y-%m-%d-%H-%M-%S")
        flow_name = f"FlowPlayList_{timestamp}"
        flow_description = "A flow to generate a music playlist."

        # Create a role for the flow.
        role_name = f"BedrockFlowRole-{flow_name}"
```

```
        role = create_flow_role(iam_client, role_name)
        role_arn = role['Arn']

        # Create the flow.
        response = create_playlist_flow(
            bedrock_agent_client, flow_name, flow_description, role_arn,
prompt_model_id)
        flow_id = response.get('id')

        if flow_id:
            # Update accessible resources in the role.
            model_arn = get_model_arn(bedrock_client, prompt_model_id)
            update_role_policy(iam_client, role_name, [
                               response.get('arn'), model_arn])

            # Prepare the flow and flow version.
            flow_version, flow_alias = prepare_flow_version_and_alias(
                bedrock_agent_client, flow_id)

            # Run the flow.
            if flow_version and flow_alias:
                run_playlist_flow(bedrock_agent_runtime_client,
                                  flow_id, flow_alias)

                delete_choice = input("Delete flow? y or n : ").lower()


            else:
                print("Couldn't run. Deleting flow and role.")
                delete_flow(bedrock_agent_client, flow_id)
                delete_flow_role(iam_client, role_name)
        else:
            print("Couldn't create flow.")


    except Exception as e:
        print(f"Fatal error: {str(e)}")

    finally:
        if delete_choice == 'y':
                delete_role_resources(bedrock_agent_client,
                                      iam_client,
                                      role_name,
                                      flow_id,
```

```
                                        flow_version,
                                        flow_alias)
        else:
            print("Flow not deleted. ")
            print(f"\tFlow ID: {flow_id}")
            print(f"\tFlow version: {flow_version}")
            print(f"\tFlow alias: {flow_alias}")
            print(f"\tRole ARN: {role_arn}")

        print("Done!")

if __name__ == "__main__":
    main()


def invoke_flow(client, flow_id, flow_alias_id, input_data):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.

    Returns:
        Dict containing flow status and flow output.
    """

    response = None
    request_params = None

    request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }


    response = client.invoke_flow(**request_params)

    flow_status = ""
    output= ""
```

```
    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Save the model output.
        elif 'flowOutputEvent' in event:
            output = event['flowOutputEvent']['content']['document']
            logger.info("Output : %s", output)

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "output": output

    }




def run_playlist_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Runs the playlist generator flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """



    print ("Welcome to the playlist generator flow.")
    # Get the initial prompt from the user.
    genre = input("Enter genre: ")
    number_of_songs = int(input("Enter number of songs: "))
```

```
    # Use prompt to create input data for the input node.
    flow_input_data = {
        "content": {
            "document": {
                "genre" : genre,
                "number" : number_of_songs
            }
        },
        "nodeName": "FlowInput",
        "nodeOutputName": "document"
    }

    try:

        result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data)

        status = result['flow_status']

        if status == "SUCCESS":
                # The flow completed successfully.
                logger.info("The flow %s successfully completed.", flow_id)
                print(result['output'])
        else:
            logger.warning("Flow status: %s",status)

    except ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})
        raise

    except Exception as e:
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})
        raise



def create_flow_role(client, role_name):
    """
    Creates an IAM role for Amazon Bedrock with permissions to run a flow.

    Args:
        role_name (str): Name for the new IAM role.
```

```
    Returns:
        str: The role Amazon Resource Name.
    """



    # Trust relationship policy - allows Amazon Bedrock service to assume this
role.
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "bedrock.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }]
    }

    # Basic inline policy for for running a flow.

    resources = "*"

    bedrock_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                # Using * as placeholder - Later you update with specific ARNs.
                "Resource": resources
            }
        ]
    }



    try:
        # Create the IAM role with trust policy
        logging.info("Creating role: %s",role_name)
        role = client.create_role(
```

```
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy),
            Description="Role for Amazon Bedrock operations"
        )

        # Attach inline policy to the role
        print("Attaching inline policy")
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(bedrock_policy)
        )

        logging.info("Create Role ARN: %s", role['Role']['Arn'])
        return role['Role']

    except ClientError as e:
        logging.warning("Error creating role: %s", str(e))
        raise
    except Exception as e:
        logging.warning("Unexpected error: %s", str(e))
        raise


def update_role_policy(client, role_name, resource_arns):
    """
    Updates an IAM role's inline policy with specific resource ARNs.

    Args:
        role_name (str): Name of the existing role.
        resource_arns (list): List of resource ARNs to allow access to.
    """



    updated_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:GetFlow",
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
```

```
                ],
                "Resource": resource_arns
            }
        ]
    }

    try:
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(updated_policy)
        )
        logging.info("Updated policy for role: %s",role_name)

    except ClientError as e:
        logging.warning("Error updating role policy: %s", str(e))
        raise


def delete_flow_role(client, role_name):
    """
    Deletes an IAM role.

    Args:
        role_name (str): Name of the role to delete.
    """




    try:
        # Detach and delete inline policies
        policies = client.list_role_policies(RoleName=role_name)['PolicyNames']
        for policy_name in policies:
            client.delete_role_policy(RoleName=role_name, PolicyName=policy_name)

        # Delete the role
        client.delete_role(RoleName=role_name)
        logging.info("Deleted role: %s", role_name)


    except ClientError as e:
        logging.info("Error Deleting role: %s", str(e))
        raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateFlow
  - CreateFlowAlias
  - CreateFlowVersion
  - DeleteFlow
  - DeleteFlowAlias
  - DeleteFlowVersion
  - GetFlow
  - GetFlowAlias
  - GetFlowVersion
  - InvokeFlow
  - PrepareFlow

Um exemplo completo que mostra como criar e invocar prompts gerenciados do Amazon Bedrock usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um prompt gerenciado.
- Criar uma versão do prompt.
- Invocar o prompt usando a versão.
- Limpar os recursos (opcional).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar e invocar um prompt gerenciado.

```
import argparse
import boto3
import logging
import time

# Now import the modules
from prompt import create_prompt, create_prompt_version, delete_prompt
from run_prompt import invoke_prompt

logging.basicConfig(
    level=logging.INFO,
    format='%(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)



def run_scenario(bedrock_client, bedrock_runtime_client, model_id, cleanup=True):
    """
    Runs the Amazon Bedrock managed prompt scenario.

    Args:
        bedrock_client: The Amazon Bedrock Agent client.
        bedrock_runtime_client: The Amazon Bedrock Runtime client.
        model_id (str): The model ID to use for the prompt.
        cleanup (bool): Whether to clean up resources at the end of the scenario.

    Returns:
        dict: A dictionary containing the created resources.
    """
    prompt_id = None

    try:
        # Step 1: Create a prompt
        print("\n=== Step 1: Creating a prompt ===")
        prompt_name = f"PlaylistGenerator-{int(time.time())}"
        prompt_description = "Playlist generator"
        prompt_template = """
          Make me a {{genre}} playlist consisting of the following number of
 songs: {{number}}."""
```

```
        create_response = create_prompt(
            bedrock_client,
            prompt_name,
            prompt_description,
            prompt_template,
            model_id
        )

        prompt_id = create_response['id']
        print(f"Created prompt: {prompt_name} with ID: {prompt_id}")

        # Create a version of the prompt
        print("\n=== Creating a version of the prompt ===")
        version_response = create_prompt_version(
            bedrock_client,
            prompt_id,
            description="Initial version of the product description generator"
        )

        prompt_version_arn = version_response['arn']
        prompt_version = version_response['version']

        print(f"Created prompt version: {prompt_version}")
        print(f"Prompt version ARN: {prompt_version_arn}")

        # Step 2: Invoke the prompt directly
        print("\n=== Step 2: Invoking the prompt ===")
        input_variables = {
            "genre": "pop",
            "number": "2",
           }

        # Use the ARN from the create_prompt_version response
        result = invoke_prompt(
            bedrock_runtime_client,
            prompt_version_arn,
            input_variables
        )
        # Display the playlist
        print(f"\n{result}")


        # Step 3: Clean up resources (optional)
        if cleanup:
```

```
            print("\n=== Step 3: Cleaning up resources ===")

            # Delete the prompt
            print(f"Deleting prompt {prompt_id}...")
            delete_prompt(bedrock_client, prompt_id)

            print("Cleanup complete")
        else:
            print("\n=== Resources were not cleaned up ===")
            print(f"Prompt ID: {prompt_id}")



    except Exception as e:
        logger.exception("Error in scenario: %s", str(e))

        # Attempt to clean up if an error occurred and cleanup was requested
        if cleanup and prompt_id:
            try:
                print("\nCleaning up resources after error...")

                # Delete the prompt
                try:
                    delete_prompt(bedrock_client, prompt_id)
                    print("Cleanup after error complete")
                except Exception as cleanup_error:
                    logger.error("Error during cleanup: %s", str(cleanup_error))
            except Exception as final_error:
                logger.error("Final error during cleanup: %s", str(final_error))

        # Re-raise the original exception
        raise

def main():
    """
    Entry point for the Amazon Bedrock managed prompt scenario.
    """
    parser = argparse.ArgumentParser(
        description="Run the Amazon Bedrock managed prompt scenario."
    )
    parser.add_argument(
        '--region',
        default='us-east-1',
        help="The AWS Region to use."
```

```
    )
    parser.add_argument(
        '--model-id',
        default='anthropic.claude-v2',
        help="The model ID to use for the prompt."
    )
    parser.add_argument(
        '--cleanup',
        action='store_true',
        default=True,
        help="Clean up resources at the end of the scenario."
    )
    parser.add_argument(
        '--no-cleanup',
        action='store_false',
        dest='cleanup',
        help="Don't clean up resources at the end of the scenario."
    )
    args = parser.parse_args()

    bedrock_client = boto3.client('bedrock-agent', region_name=args.region)
    bedrock_runtime_client = boto3.client('bedrock-runtime',
 region_name=args.region)

    print("=== Amazon Bedrock Managed Prompt Scenario ===")
    print(f"Region: {args.region}")
    print(f"Model ID: {args.model_id}")
    print(f"Cleanup resources: {args.cleanup}")

    try:
        run_scenario(
            bedrock_client,
            bedrock_runtime_client,
            args.model_id,
            args.cleanup
        )

    except Exception as e:
        logger.exception("Error running scenario: %s", str(e))

if __name__ == "__main__":
    main()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - Converse
  - CreatePrompt
  - CreatePromptVersion
  - DeletePrompt

Um exemplo completo que mostra como criar e invocar Amazon Bedrock Agents usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um perfil de execução para o agente.
- Criar o agente e implantar uma versão de RASCUNHO.
- Criar uma função do Lambda que implante os recursos do agente.
- Criar um grupo de ação que conecte o agente à função do Lambda.
- Implantar o agente totalmente configurado.
- Invocar o agente com prompts fornecidos pelo usuário.
- Exclua todos os recursos criados.

Python

SDK para Python (Boto3)

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie e invoque um agente.

```
REGION = "us-east-1"
ROLE_POLICY_NAME = "agent_permissions"
```

```
class BedrockAgentScenarioWrapper:
    """Runs a scenario that shows how to get started using Amazon Bedrock
 Agents."""

    def __init__(
            self, bedrock_agent_client, runtime_client, lambda_client,
 iam_resource, postfix
    ):
        self.iam_resource = iam_resource
        self.lambda_client = lambda_client
        self.bedrock_agent_runtime_client = runtime_client
        self.postfix = postfix

        self.bedrock_wrapper = BedrockAgentWrapper(bedrock_agent_client)

        self.agent = None
        self.agent_alias = None
        self.agent_role = None
        self.prepared_agent_details = None
        self.lambda_role = None
        self.lambda_function = None

    def run_scenario(self):
        print("=" * 88)
        print("Welcome to the Amazon Bedrock Agents demo.")
        print("=" * 88)

        # Query input from user
        print("Let's start with creating an agent:")
        print("-" * 40)
        name, foundation_model = self._request_name_and_model_from_user()
        print("-" * 40)

        # Create an execution role for the agent
        self.agent_role = self._create_agent_role(foundation_model)

        # Create the agent
        self.agent = self._create_agent(name, foundation_model)

        # Prepare a DRAFT version of the agent
        self.prepared_agent_details = self._prepare_agent()
```

```
        # Create the agent's Lambda function
        self.lambda_function = self._create_lambda_function()

        # Configure permissions for the agent to invoke the Lambda function
        self._allow_agent_to_invoke_function()
        self._let_function_accept_invocations_from_agent()

        # Create an action group to connect the agent with the Lambda function
        self._create_agent_action_group()

        # If the agent has been modified or any components have been added,
 prepare the agent again
        components = [self._get_agent()]
        components += self._get_agent_action_groups()
        components += self._get_agent_knowledge_bases()

        latest_update = max(component["updatedAt"] for component in components)
        if latest_update > self.prepared_agent_details["preparedAt"]:
            self.prepared_agent_details = self._prepare_agent()

        # Create an agent alias
        self.agent_alias = self._create_agent_alias()

        # Test the agent
        self._chat_with_agent(self.agent_alias)

        print("=" * 88)
        print("Thanks for running the demo!\n")

        if q.ask("Do you want to delete the created resources? [y/N] ",
 q.is_yesno):
            self._delete_resources()
            print("=" * 88)
            print(
                "All demo resources have been deleted. Thanks again for running
 the demo!"
            )
        else:
            self._list_resources()
            print("=" * 88)
            print("Thanks again for running the demo!")

    def _request_name_and_model_from_user(self):
        existing_agent_names = [
```

```
            agent["agentName"] for agent in self.bedrock_wrapper.list_agents()
        ]

        while True:
            name = q.ask("Enter an agent name: ", self.is_valid_agent_name)
            if name.lower() not in [n.lower() for n in existing_agent_names]:
                break
            print(
                f"Agent {name} conflicts with an existing agent. Please use a
different name."
            )

        models = ["anthropic.claude-instant-v1", "anthropic.claude-v2"]
        model_id = models[
            q.choose("Which foundation model would you like to use? ", models)
        ]

        return name, model_id

    def _create_agent_role(self, model_id):
        role_name = f"AmazonBedrockExecutionRoleForAgents_{self.postfix}"
        model_arn = f"arn:aws:bedrock:{REGION}::foundation-model/{model_id}*"

        print("Creating an an execution role for the agent...")

        try:
            role = self.iam_resource.create_role(
                RoleName=role_name,
                AssumeRolePolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {"Service":
"bedrock.amazonaws.com"},
                                "Action": "sts:AssumeRole",
                            }
                        ],
                    }
                ),
            )

            role.Policy(ROLE_POLICY_NAME).put(
```

```
                PolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "bedrock:InvokeModel",
                                "Resource": model_arn,
                            }
                        ],
                    }
                )
            )
        except ClientError as e:
            logger.error(f"Couldn't create role {role_name}. Here's why: {e}")
            raise

        return role

    def _create_agent(self, name, model_id):
        print("Creating the agent...")

        instruction = """
            You are a friendly chat bot. You have access to a function called
that returns
            information about the current date and time. When responding with
date or time,
            please make sure to add the timezone UTC.
            """
        agent = self.bedrock_wrapper.create_agent(
            agent_name=name,
            foundation_model=model_id,
            instruction=instruction,
            role_arn=self.agent_role.arn,
        )
        self._wait_for_agent_status(agent["agentId"], "NOT_PREPARED")

        return agent

    def _prepare_agent(self):
        print("Preparing the agent...")

        agent_id = self.agent["agentId"]
        prepared_agent_details = self.bedrock_wrapper.prepare_agent(agent_id)
```

```
        self._wait_for_agent_status(agent_id, "PREPARED")

        return prepared_agent_details

    def _create_lambda_function(self):
        print("Creating the Lambda function...")

        function_name = f"AmazonBedrockExampleFunction_{self.postfix}"

        self.lambda_role = self._create_lambda_role()

        try:
            deployment_package = self._create_deployment_package(function_name)

            lambda_function = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="Lambda function for Amazon Bedrock example",
                Runtime="python3.11",
                Role=self.lambda_role.arn,
                Handler=f"{function_name}.lambda_handler",
                Code={"ZipFile": deployment_package},
                Publish=True,
            )

            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)

        except ClientError as e:
            logger.error(
                f"Couldn't create Lambda function {function_name}. Here's why:
{e}"
            )
            raise

        return lambda_function

    def _create_lambda_role(self):
        print("Creating an execution role for the Lambda function...")

        role_name = f"AmazonBedrockExecutionRoleForLambda_{self.postfix}"

        try:
            role = self.iam_resource.create_role(
                RoleName=role_name,
```

```
                AssumeRolePolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {"Service": "lambda.amazonaws.com"},
                                "Action": "sts:AssumeRole",
                            }
                        ],
                    }
                ),
            )
            role.attach_policy(
                PolicyArn="arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"
            )
            print(f"Created role {role_name}")
        except ClientError as e:
            logger.error(f"Couldn't create role {role_name}. Here's why: {e}")
            raise

        print("Waiting for the execution role to be fully propagated...")
        wait(10)

        return role

    def _allow_agent_to_invoke_function(self):
        policy = self.iam_resource.RolePolicy(
            self.agent_role.role_name, ROLE_POLICY_NAME
        )
        doc = policy.policy_document
        doc["Statement"].append(
            {
                "Effect": "Allow",
                "Action": "lambda:InvokeFunction",
                "Resource": self.lambda_function["FunctionArn"],
            }
        )

 self.agent_role.Policy(ROLE_POLICY_NAME).put(PolicyDocument=json.dumps(doc))

    def _let_function_accept_invocations_from_agent(self):
        try:
```

```
            self.lambda_client.add_permission(
                FunctionName=self.lambda_function["FunctionName"],
                SourceArn=self.agent["agentArn"],
                StatementId="BedrockAccess",
                Action="lambda:InvokeFunction",
                Principal="bedrock.amazonaws.com",
            )
        except ClientError as e:
            logger.error(
                f"Couldn't grant Bedrock permission to invoke the Lambda
function. Here's why: {e}"
            )
            raise

    def _create_agent_action_group(self):
        print("Creating an action group for the agent...")

        try:
            with open("./scenario_resources/api_schema.yaml") as file:
                self.bedrock_wrapper.create_agent_action_group(
                    name="current_date_and_time",
                    description="Gets the current date and time.",
                    agent_id=self.agent["agentId"],
                    agent_version=self.prepared_agent_details["agentVersion"],
                    function_arn=self.lambda_function["FunctionArn"],
                    api_schema=json.dumps(yaml.safe_load(file)),
                )
        except ClientError as e:
            logger.error(f"Couldn't create agent action group. Here's why: {e}")
            raise

    def _get_agent(self):
        return self.bedrock_wrapper.get_agent(self.agent["agentId"])

    def _get_agent_action_groups(self):
        return self.bedrock_wrapper.list_agent_action_groups(
            self.agent["agentId"], self.prepared_agent_details["agentVersion"]
        )

    def _get_agent_knowledge_bases(self):
        return self.bedrock_wrapper.list_agent_knowledge_bases(
            self.agent["agentId"], self.prepared_agent_details["agentVersion"]
        )
```

```
    def _create_agent_alias(self):
        print("Creating an agent alias...")

        agent_alias_name = "test_agent_alias"
        agent_alias = self.bedrock_wrapper.create_agent_alias(
            agent_alias_name, self.agent["agentId"]
        )

        self._wait_for_agent_status(self.agent["agentId"], "PREPARED")

        return agent_alias

    def _wait_for_agent_status(self, agent_id, status):
        while self.bedrock_wrapper.get_agent(agent_id)["agentStatus"] != status:
            wait(2)

    def _chat_with_agent(self, agent_alias):
        print("-" * 88)
        print("The agent is ready to chat.")
        print("Try asking for the date or time. Type 'exit' to quit.")

        # Create a unique session ID for the conversation
        session_id = uuid.uuid4().hex

        while True:
            prompt = q.ask("Prompt: ", q.non_empty)

            if prompt == "exit":
                break

            response = asyncio.run(self._invoke_agent(agent_alias, prompt,
 session_id))

            print(f"Agent: {response}")

    async def _invoke_agent(self, agent_alias, prompt, session_id):
        response = self.bedrock_agent_runtime_client.invoke_agent(
            agentId=self.agent["agentId"],
            agentAliasId=agent_alias["agentAliasId"],
            sessionId=session_id,
            inputText=prompt,
        )

        completion = ""
```

```
        for event in response.get("completion"):
            chunk = event["chunk"]
            completion += chunk["bytes"].decode()

        return completion

    def _delete_resources(self):
        if self.agent:
            agent_id = self.agent["agentId"]

            if self.agent_alias:
                agent_alias_id = self.agent_alias["agentAliasId"]
                print("Deleting agent alias...")
                self.bedrock_wrapper.delete_agent_alias(agent_id, agent_alias_id)

            print("Deleting agent...")
            agent_status = self.bedrock_wrapper.delete_agent(agent_id)
["agentStatus"]
            while agent_status == "DELETING":
                wait(5)
                try:
                    agent_status = self.bedrock_wrapper.get_agent(
                        agent_id, log_error=False
                    )["agentStatus"]
                except ClientError as err:
                    if err.response["Error"]["Code"] ==
 "ResourceNotFoundException":
                        agent_status = "DELETED"

        if self.lambda_function:
            name = self.lambda_function["FunctionName"]
            print(f"Deleting function '{name}'...")
            self.lambda_client.delete_function(FunctionName=name)

        if self.agent_role:
            print(f"Deleting role '{self.agent_role.role_name}'...")
            self.agent_role.Policy(ROLE_POLICY_NAME).delete()
            self.agent_role.delete()

        if self.lambda_role:
            print(f"Deleting role '{self.lambda_role.role_name}'...")
            for policy in self.lambda_role.attached_policies.all():
                policy.detach_role(RoleName=self.lambda_role.role_name)
```

```
            self.lambda_role.delete()

    def _list_resources(self):
        print("-" * 40)
        print(f"Here is the list of created resources in '{REGION}'.")
        print("Make sure you delete them once you're done to avoid unnecessary
 costs.")
        if self.agent:
            print(f"Bedrock Agent:   {self.agent['agentName']}")
        if self.lambda_function:
            print(f"Lambda function: {self.lambda_function['FunctionName']}")
        if self.agent_role:
            print(f"IAM role:        {self.agent_role.role_name}")
        if self.lambda_role:
            print(f"IAM role:        {self.lambda_role.role_name}")

    @staticmethod
    def is_valid_agent_name(answer):
        valid_regex = r"^[a-zA-Z0-9_-]{1,100}$"
        return (
            answer
            if answer and len(answer) <= 100 and re.match(valid_regex, answer)
            else None,
            "I need a name for the agent, please. Valid characters are a-z, A-Z,
 0-9, _ (underscore) and - (hyphen).",
        )

    @staticmethod
    def _create_deployment_package(function_name):
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, "w") as zipped:
            zipped.write(
                "./scenario_resources/lambda_function.py", f"{function_name}.py"
            )
        buffer.seek(0)
        return buffer.read()


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

    postfix = "".join(
        random.choice(string.ascii_lowercase + "0123456789") for _ in range(8)
    )
```

```
    scenario = BedrockAgentScenarioWrapper(
        bedrock_agent_client=boto3.client(
            service_name="bedrock-agent", region_name=REGION
        ),
        runtime_client=boto3.client(
            service_name="bedrock-agent-runtime", region_name=REGION
        ),
        lambda_client=boto3.client(service_name="lambda", region_name=REGION),
        iam_resource=boto3.resource("iam"),
        postfix=postfix,
    )
    try:
        scenario.run_scenario()
    except Exception as e:
        logging.exception(f"Something went wrong with the demo. Here's what:
{e}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK para Python (Boto3).
  - CreateAgent
  - CreateAgentActionGroup
  - CreateAgentAlias
  - DeleteAgent
  - DeleteAgentAlias
  - GetAgent
  - ListAgentActionGroups
  - ListAgentKnowledgeBases
  - ListAgents
  - PrepareAgent

Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

Python

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Step Functions

Exemplos de código para Amazon Bedrock Agents Runtime usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Agents Runtime com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon Bedrock Agents Runtime: mais informações sobre o Amazon Bedrock Agents Runtime.
- Referência de API do Amazon Bedrock Agents Runtime: detalhes sobre todas as ações disponíveis do Amazon Bedrock Agents Runtime.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Amazon Bedrock Agents Runtime usando AWS SDKs
  - Converse com um fluxo do Amazon Bedrock
  - Ações para o Amazon Bedrock Agents Runtime usando AWS SDKs
    - Use InvokeAgentcom um AWS SDK
    - Use InvokeFlowcom um AWS SDK
- Cenários para o Amazon Bedrock Agents Runtime usando AWS SDKs
  - Um exemplo completo que mostra como criar e invocar um fluxo do Amazon Bedrock usando um AWS SDK

- Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

Exemplos básicos do Amazon Bedrock Agents Runtime usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Bedrock Agents Runtime com SDKs da AWS .

Exemplos

- Converse com um fluxo do Amazon Bedrock
- Ações para o Amazon Bedrock Agents Runtime usando AWS SDKs
  - Use InvokeAgentcom um AWS SDK
  - Use InvokeFlowcom um AWS SDK

Converse com um fluxo do Amazon Bedrock

O exemplo de código a seguir mostra como usar InvokeFlow para conversar com um fluxo do Amazon Bedrock que inclui um nó de agente.

Consulte mais informações em Converse with an Amazon Bedrock flow.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Shows how to run an Amazon Bedrock flow with InvokeFlow and handle muli-turn
 interaction
for a single conversation.
For more information, see https://docs.aws.amazon.com/bedrock/latest/userguide/
flows-multi-turn-invocation.html.
```

```
"""
import logging
import boto3
import botocore

import botocore.exceptions

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


def invoke_flow(client, flow_id, flow_alias_id, input_data, execution_id):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.
        execution_id: Execution ID for continuing a flow. Use the value None on
 first run.

    Returns:
        Dict containing flow_complete status, input_required info, and
 execution_id
    """

    response = None
    request_params = None

    if execution_id is None:
        # Don't pass execution ID for first run.
        request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }
    else:
        request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
```

```
            "executionId": execution_id,
            "inputs": [input_data],
            "enableTrace": True
        }

    response = client.invoke_flow(**request_params)

    if "executionId" not in request_params:
        execution_id = response['executionId']

    input_required = None
    flow_status = ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Check if more input us needed from user.
        elif 'flowMultiTurnInputRequestEvent' in event:
            input_required = event

        # Print the model output.
        elif 'flowOutputEvent' in event:
            print(event['flowOutputEvent']['content']['document'])

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "input_required": input_required,
        "execution_id": execution_id
    }


def converse_with_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Run a conversation with the supplied flow.

    Args:
```

```
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """

    flow_execution_id = None
    finished = False

    # Get the intial prompt from the user.
    user_input = input("Enter input: ")

    # Use prompt to create input data.
    flow_input_data = {
        "content": {
            "document": user_input
        },
        "nodeName": "FlowInputNode",
        "nodeOutputName": "document"
    }

    try:
        while not finished:
            # Invoke the flow until successfully finished.

            result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data,
 flow_execution_id)

            status = result['flow_status']
            flow_execution_id = result['execution_id']
            more_input = result['input_required']
            if status == "INPUT_REQUIRED":
                # The flow needs more information from the user.
                logger.info("The flow %s requires more input", flow_id)
                user_input = input(
                    more_input['flowMultiTurnInputRequestEvent']['content']
['document'] + ": ")
                flow_input_data = {
                    "content": {
                        "document": user_input
                    },
                    "nodeName": more_input['flowMultiTurnInputRequestEvent']
['nodeName'],
```

```
                    "nodeInputName": "agentInputText"

                }
            elif status == "SUCCESS":
                # The flow completed successfully.
                finished = True
                logger.info("The flow %s successfully completed.", flow_id)

    except botocore.exceptions.ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})

    except Exception as e:
        print(f"An error occurred: {str(e)}")
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})


def main():
    """
    Main entry point for the script.
    """

    # Replace these with your actual flow ID and flow alias ID.
    FLOW_ID = 'YOUR_FLOW_ID'
    FLOW_ALIAS_ID = 'YOUR_FLOW_ALIAS_ID'

    logger.info("Starting conversation with FLOW: %s ID: %s",
                FLOW_ID, FLOW_ALIAS_ID)

    # Get the Bedrock agent runtime client.
    session = boto3.Session(profile_name='default')
    bedrock_agent_client = session.client('bedrock-agent-runtime')

    # Start the conversation.
    converse_with_flow(bedrock_agent_client, FLOW_ID, FLOW_ALIAS_ID)

    logger.info("Conversation with FLOW: %s ID: %s finished",
                FLOW_ID, FLOW_ALIAS_ID)


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a InvokeFlowReferência da API AWS SDK for Python (Boto3).

Ações para o Amazon Bedrock Agents Runtime usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Bedrock Agents Runtime com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Bedrock Agents Runtime e são trechos de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para o Amazon Bedrock Agents Runtime usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon Bedrock Agents Runtime API Reference.

Exemplos

- Use InvokeAgentcom um AWS SDK
- Use InvokeFlowcom um AWS SDK

Use **`InvokeAgent`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `InvokeAgent`.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  BedrockAgentRuntimeClient,
```

```
  InvokeAgentCommand,
} from "@aws-sdk/client-bedrock-agent-runtime";

/**
 * @typedef {Object} ResponseBody
 * @property {string} completion
 */

/**
 * Invokes a Bedrock agent to run an inference using the input
 * provided in the request body.
 *
 * @param {string} prompt - The prompt that you want the Agent to complete.
 * @param {string} sessionId - An arbitrary identifier for the session.
 */
export const invokeBedrockAgent = async (prompt, sessionId) => {
  const client = new BedrockAgentRuntimeClient({ region: "us-east-1" });
  // const client = new BedrockAgentRuntimeClient({
  //   region: "us-east-1",
  //   credentials: {
  //     accessKeyId: "accessKeyId", // permission to invoke agent
  //     secretAccessKey: "accessKeySecret",
  //   },
  // });

  const agentId = "AJBHXXILZN";
  const agentAliasId = "AVKP1ITZAA";

  const command = new InvokeAgentCommand({
    agentId,
    agentAliasId,
    sessionId,
    inputText: prompt,
  });

  try {
    let completion = "";
    const response = await client.send(command);

    if (response.completion === undefined) {
      throw new Error("Completion is undefined");
    }

    for await (const chunkEvent of response.completion) {
```

```
      const chunk = chunkEvent.chunk;
      console.log(chunk);
      const decodedResponse = new TextDecoder("utf-8").decode(chunk.bytes);
      completion += decodedResponse;
    }

    return { sessionId: sessionId, completion };
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const result = await invokeBedrockAgent("I need help.", "123");
  console.log(result);
}
```

- Para obter detalhes da API, consulte InvokeAgenta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Invoque um agente.

```
    def invoke_agent(self, agent_id, agent_alias_id, session_id, prompt):
        """
        Sends a prompt for the agent to process and respond to.

        :param agent_id: The unique identifier of the agent to use.
        :param agent_alias_id: The alias of the agent to use.
```

```
        :param session_id: The unique identifier of the session. Use the same
value across requests
                           to continue the same conversation.
        :param prompt: The prompt that you want Claude to complete.
        :return: Inference response from the model.
        """

        try:
            # Note: The execution time depends on the foundation model,
complexity of the agent,
            # and the length of the prompt. In some cases, it can take up to a
minute or more to
            # generate a response.
            response = self.agents_runtime_client.invoke_agent(
                agentId=agent_id,
                agentAliasId=agent_alias_id,
                sessionId=session_id,
                inputText=prompt,
            )

            completion = ""

            for event in response.get("completion"):
                chunk = event["chunk"]
                completion = completion + chunk["bytes"].decode()

        except ClientError as e:
            logger.error(f"Couldn't invoke agent. {e}")
            raise

        return completion
```

- Para obter detalhes da API, consulte a InvokeAgentReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use aws_config::{BehaviorVersion, SdkConfig};
use aws_sdk_bedrockagentruntime::{
    self as bedrockagentruntime,
    types::{error::ResponseStreamError, ResponseStream},
};
#[allow(unused_imports)]
use mockall::automock;

const BEDROCK_AGENT_ID: &str = "AJBHXXILZN";
const BEDROCK_AGENT_ALIAS_ID: &str = "AVKP1ITZAA";
const BEDROCK_AGENT_REGION: &str = "us-east-1";

#[cfg(not(test))]
pub use EventReceiverImpl as EventReceiver;
#[cfg(test)]
pub use MockEventReceiverImpl as EventReceiver;

pub struct EventReceiverImpl {
    inner: aws_sdk_bedrockagentruntime::primitives::event_stream::EventReceiver<
        ResponseStream,
        ResponseStreamError,
    >,
}

#[cfg_attr(test, automock)]
impl EventReceiverImpl {
    #[allow(dead_code)]
    pub fn new(
        inner:
 aws_sdk_bedrockagentruntime::primitives::event_stream::EventReceiver<
            ResponseStream,
```

```
            ResponseStreamError,
        >,
    ) -> Self {
        Self { inner }
    }

    pub async fn recv(
        &mut self,
    ) -> Result<
        Option<ResponseStream>,
        aws_sdk_bedrockagentruntime::error::SdkError<
            ResponseStreamError,
            aws_smithy_types::event_stream::RawMessage,
        >,
    > {
        self.inner.recv().await
    }
}

#[tokio::main]
async fn main() -> Result<(), Box<bedrockagentruntime::Error>> {
    let result = invoke_bedrock_agent("I need help.".to_string(),
 "123".to_string()).await?;
    println!("{}", result);
    Ok(())
}

async fn invoke_bedrock_agent(
    prompt: String,
    session_id: String,
) -> Result<String, bedrockagentruntime::Error> {
    let sdk_config: SdkConfig = aws_config::defaults(BehaviorVersion::latest())
        .region(BEDROCK_AGENT_REGION)
        .load()
        .await;
    let bedrock_client = bedrockagentruntime::Client::new(&sdk_config);

    let command_builder = bedrock_client
        .invoke_agent()
        .agent_id(BEDROCK_AGENT_ID)
        .agent_alias_id(BEDROCK_AGENT_ALIAS_ID)
        .session_id(session_id)
        .input_text(prompt);
```

```
    let response = command_builder.send().await?;

    let response_stream = response.completion;

    let event_receiver = EventReceiver::new(response_stream);

    process_agent_response_stream(event_receiver).await
}

async fn process_agent_response_stream(
    mut event_receiver: EventReceiver,
) -> Result<String, bedrockagentruntime::Error> {
    let mut full_agent_text_response = String::new();

    while let Some(event_result) = event_receiver.recv().await? {
        match event_result {
            ResponseStream::Chunk(chunk) => {
                if let Some(bytes) = chunk.bytes {
                    match String::from_utf8(bytes.into_inner()) {
                        Ok(text_chunk) => {
                            full_agent_text_response.push_str(&text_chunk);
                        }
                        Err(e) => {
                            eprintln!("UTF-8 decoding error for chunk: {}", e);
                        }
                    }
                }
            }
            _ => {
                panic!("received an unhandled event type from Bedrock stream",);
            }
        }
    }
    Ok(full_agent_text_response)
}

#[cfg(test)]
mod test {

    use super::*;

    #[tokio::test]
    async fn test_process_agent_response_stream() {
        let mut mock = MockEventReceiverImpl::default();
```

```
        mock.expect_recv().times(1).returning(|| {
            Ok(Some(
                aws_sdk_bedrockagentruntime::types::ResponseStream::Chunk(
                    aws_sdk_bedrockagentruntime::types::PayloadPart::builder()
                        .set_bytes(Some(aws_smithy_types::Blob::new(vec![
                            116, 101, 115, 116, 32, 99, 111, 109, 112, 108, 101,
 116, 105, 111, 110,
                        ])))
                        .build(),
                ),
            ))
        });

        // end the stream
        mock.expect_recv().times(1).returning(|| Ok(None));

        let response = process_agent_response_stream(mock).await.unwrap();

        assert_eq!("test completion", response);
    }

    #[tokio::test]
    #[should_panic(expected = "received an unhandled event type from Bedrock
 stream")]
    async fn test_process_agent_response_stream_error() {
        let mut mock = MockEventReceiverImpl::default();
        mock.expect_recv().times(1).returning(|| {
            Ok(Some(
                aws_sdk_bedrockagentruntime::types::ResponseStream::Trace(

 aws_sdk_bedrockagentruntime::types::TracePart::builder().build(),
                ),
            ))
        });

        let _ = process_agent_response_stream(mock).await.unwrap();
    }
}
```

- Para obter detalhes da API, consulte a InvokeAgentreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   DATA(lo_result) = lo_bdz->invokeagent(
     iv_agentid      = iv_agentid
       iv_agentaliasid = iv_agentaliasid
       iv_enabletrace  = abap_true
       iv_sessionid    = CONV #( cl_system_uuid=>create_uuid_c26_static( ) )
       iv_inputtext    = |Let's play "rock, paper, scissors".  I choose
rock.| ).
   DATA(lo_stream) = lo_result->get_completion( ).
   TRY.
       " loop while there are still events in the stream
       WHILE lo_stream->/aws1/if_rt_stream_reader~data_available( ) = abap_true.
         DATA(lo_evt) = lo_stream->read( ).
         " each /AWS1/CL_BDZRESPONSESTREAM_EV event contains exactly one member
         " all others are INITIAL.  For each event, process the non-initial
         " member if desired
         IF lo_evt->get_chunk( ) IS NOT INITIAL.
           " Process a Chunk event
           DATA(lv_xstr) = lo_evt->get_chunk( )->get_bytes( ).
           DATA(lv_answer) = /aws1/cl_rt_util=>xstring_to_string( lv_xstr ).
           " the answer says something like "I chose paper, so you lost"
         ELSEIF lo_evt->get_files( ) IS NOT INITIAL.
           " process a Files event if desired
         ELSEIF lo_evt->get_returncontrol( ) IS NOT INITIAL.
           " process a ReturnControl event if desired
         ELSEIF lo_evt->get_trace( ) IS NOT INITIAL.
           " process a Trace event if desired
         ENDIF.
       ENDWHILE.
       " the stream of events can possibly contain an exception
       " which will be raised to break the loop
       " catch /AWS1/CX_BDZACCESSDENIEDEX.
```

```
        " catch /AWS1/CX_BDZINTERNALSERVEREX.
        " catch /AWS1/CX_BDZMODELNOTREADYEX.
        " catch /AWS1/CX_BDZVALIDATIONEX.
        " catch /AWS1/CX_BDZTHROTTLINGEX.
        " catch /AWS1/CX_BDZDEPENDENCYFAILEDEX.
        " catch /AWS1/CX_BDZBADGATEWAYEX.
        " catch /AWS1/CX_BDZRESOURCENOTFOUNDEX.
        " catch /AWS1/CX_BDZSERVICEQUOTAEXCDEX.
        " catch /AWS1/CX_BDZCONFLICTEXCEPTION.
    ENDTRY.
```

- Para obter detalhes da API, consulte a InvokeAgentreferência da API AWS SDK for SAP ABAP.

Use **`InvokeFlow`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `InvokeFlow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar e invocar um fluxo

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentRuntimeClient,
```

```
  InvokeFlowCommand,
} from "@aws-sdk/client-bedrock-agent-runtime";

/**
 * Invokes an alias of a flow to run the inputs that you specify and return
 * the output of each node as a stream.
 *
 * @param {{
 *  flowIdentifier: string,
 *  flowAliasIdentifier: string,
 *  prompt?: string,
 *  region?: string
 * }} options
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").FlowNodeOutput>} An
 object containing information about the output from flow invocation.
 */
export const invokeBedrockFlow = async ({
  flowIdentifier,
  flowAliasIdentifier,
  prompt = "Hi, how are you?",
  region = "us-east-1",
}) => {
  const client = new BedrockAgentRuntimeClient({ region });

  const command = new InvokeFlowCommand({
    flowIdentifier,
    flowAliasIdentifier,
    inputs: [
      {
        content: {
          document: prompt,
        },
        nodeName: "FlowInputNode",
        nodeOutputName: "document",
      },
    ],
  });

  let flowResponse = {};
  const response = await client.send(command);

  for await (const chunkEvent of response.responseStream) {
    const { flowOutputEvent, flowCompletionEvent } = chunkEvent;
```

```
    if (flowOutputEvent) {
      flowResponse = { ...flowResponse, ...flowOutputEvent };
      console.log("Flow output event:", flowOutputEvent);
    } else if (flowCompletionEvent) {
      flowResponse = { ...flowResponse, ...flowCompletionEvent };
      console.log("Flow completion event:", flowCompletionEvent);
    }
  }

  return flowResponse;
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    flowIdentifier: {
      type: "string",
      required: true,
    },
    flowAliasIdentifier: {
      type: "string",
      required: true,
    },
    prompt: {
      type: "string",
    },
    region: {
      type: "string",
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
```

```
    invokeBedrockFlow(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte InvokeFlowa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Invoque um fluxo.

```
    def invoke_flow(self, flow_id, flow_alias_id, input_data, execution_id):
        """
        Invoke an Amazon Bedrock flow and handle the response stream.

        Args:
            param flow_id: The ID of the flow to invoke.
            param flow_alias_id: The alias ID of the flow.
            param input_data: Input data for the flow.
            param execution_id: Execution ID for continuing a flow. Use the value
 None on first run.

        Return: Response from the flow.
        """
        try:

            request_params = None

            if execution_id is None:
                # Don't pass execution ID for first run.
                request_params = {
```

```
                "flowIdentifier": flow_id,
                "flowAliasIdentifier": flow_alias_id,
                "inputs": input_data,
                "enableTrace": True
            }
        else:
            request_params = {
                "flowIdentifier": flow_id,
                "flowAliasIdentifier": flow_alias_id,
                "executionId": execution_id,
                "inputs": input_data,
                "enableTrace": True
            }

        response = self.agents_runtime_client.invoke_flow(**request_params)

        if "executionId" not in request_params:
            execution_id = response['executionId']

        result = ""

        # Get the streaming response
        for event in response['responseStream']:
            result = result + str(event) + '\n'
        print(result)

    except ClientError as e:
        logger.error("Couldn't invoke flow %s.", {e})
        raise

    return result
```

- Para obter detalhes da API, consulte a InvokeFlowReferência da API AWS SDK for Python (Boto3).

Cenários para o Amazon Bedrock Agents Runtime usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Bedrock Agents Runtime com AWS SDKs. Esses cenários mostram como executar tarefas específicas chamando vários perfis no Amazon Bedrock Agents Runtime ou em conjunto com outros Serviços

da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Um exemplo completo que mostra como criar e invocar um fluxo do Amazon Bedrock usando um AWS SDK
- Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

Um exemplo completo que mostra como criar e invocar um fluxo do Amazon Bedrock usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um perfil de execução para o fluxo.
- Criar o fluxo.
- Implantar o fluxo totalmente configurado.
- Invocar o fluxo com prompts fornecidos pelo usuário.
- Exclua todos os recursos criados.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gera uma playlist de música com base no gênero e no número de músicas especificados pelo usuário.

```
from datetime import datetime
import logging
```

```
import boto3

from botocore.exceptions import ClientError

from roles import create_flow_role, delete_flow_role, update_role_policy
from flow import create_flow, prepare_flow, delete_flow
from run_flow import run_playlist_flow
from flow_version import create_flow_version, delete_flow_version
from flow_alias import create_flow_alias, delete_flow_alias

logging.basicConfig(
    level=logging.INFO
)
logger = logging.getLogger(__name__)

def create_input_node(name):
    """
    Creates an input node configuration for an Amazon Bedrock flow.

    The input node serves as the entry point for the flow and defines
    the initial document structure that will be passed to subsequent nodes.

    Args:
        name (str): The name of the input node.

    Returns:
        dict: The input node configuration.

    """
    return {
        "type": "Input",
        "name": name,
        "outputs": [
            {
                "name": "document",
                "type": "Object"
            }
        ]
    }


def create_prompt_node(name, model_id):
    """
```

```
    Creates a prompt node configuration for a Bedrock flow that generates music
playlists.

    The prompt node defines an inline prompt template that creates a music
playlist based on
    a specified genre and number of songs. The prompt uses two variables that are
mapped from
    the input JSON object:
    - {{genre}}: The genre of music to create a playlist for
    - {{number}}: The number of songs to include in the playlist

    Args:
        name (str): The name of the prompt node.
        model_id (str): The identifier of the foundation model to use for the
prompt.

    Returns:
        dict: The prompt node.

    """

    return {
        "type": "Prompt",
        "name": name,
        "configuration": {
            "prompt": {
                "sourceConfiguration": {
                    "inline": {
                        "modelId": model_id,
                        "templateType": "TEXT",
                        "inferenceConfiguration": {
                            "text": {
                                "temperature": 0.8
                            }
                        },
                        "templateConfiguration": {
                            "text": {
                                "text": "Make me a {{genre}} playlist consisting
of the following number of songs: {{number}}."
                            }
                        }
                    }
                }
            }
```

```
        },
        "inputs": [
            {
                "name": "genre",
                "type": "String",
                "expression": "$.data.genre"
            },
            {
                "name": "number",
                "type": "Number",
                "expression": "$.data.number"
            }
        ],
        "outputs": [
            {
                "name": "modelCompletion",
                "type": "String"
            }
        ]
    }


def create_output_node(name):
    """
    Creates an output node configuration for a Bedrock flow.

    The output node validates that the output from the last node is a string
    and returns it unmodified. The input name must be "document".

    Args:
        name (str): The name of the output node.

    Returns:
        dict: The output node configuration containing the output node:

    """

    return {
        "type": "Output",
        "name": name,
        "inputs": [
            {
                "name": "document",
                "type": "String",
```

```
                "expression": "$.data"
            }
        ]
    }




def create_playlist_flow(client, flow_name, flow_description, role_arn,
 prompt_model_id):
    """
    Creates the playlist generator flow.
    Args:
        client: bedrock agent boto3 client.
        role_arn (str): Name for the new IAM role.
        prompt_model_id (str): The id of the model to use in the prompt node.
    Returns:
        dict: The response from the create_flow operation.
    """

    input_node = create_input_node("FlowInput")
    prompt_node = create_prompt_node("MakePlaylist", prompt_model_id)
    output_node = create_output_node("FlowOutput")

    # Create connections between the nodes
    connections = []

    #  First, create connections between the output of the flow
    # input node and each input of the prompt node.
    for prompt_node_input in prompt_node["inputs"]:
        connections.append(
            {
                "name": "_".join([input_node["name"], prompt_node["name"],
                                  prompt_node_input["name"]]),
                "source": input_node["name"],
                "target": prompt_node["name"],
                "type": "Data",
                "configuration": {
                    "data": {
                        "sourceOutput": input_node["outputs"][0]["name"],
                        "targetInput": prompt_node_input["name"]
                    }
                }
            }
```

```
        )

    # Then, create a connection between the output of the prompt node and the
 input of the flow output node
    connections.append(
        {
            "name": "_".join([prompt_node["name"], output_node["name"]]),
            "source": prompt_node["name"],
            "target": output_node["name"],
            "type": "Data",
            "configuration": {
                "data": {
                    "sourceOutput": prompt_node["outputs"][0]["name"],
                    "targetInput": output_node["inputs"][0]["name"]
                }
            }
        }
    )

    flow_def = {
        "nodes": [input_node, prompt_node, output_node],
        "connections": connections
    }

    # Create the flow.

    response = create_flow(
        client, flow_name, flow_description, role_arn, flow_def)

    return response



def get_model_arn(client, model_id):
    """
    Gets the Amazon Resource Name (ARN) for a model.
    Args:
        client (str): Amazon Bedrock boto3 client.
        model_id (str): The id of the model.
    Returns:
        str: The ARN of the model.
    """

    try:
```

```
        # Call GetFoundationModelDetails operation
        response = client.get_foundation_model(modelIdentifier=model_id)

        # Extract model ARN from the response
        model_arn = response['modelDetails']['modelArn']

        return model_arn

    except ClientError as e:
        logger.exception("Client error getting model ARN: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexpected error getting model ARN: %s", {str(e)})
        raise


def prepare_flow_version_and_alias(bedrock_agent_client,
                                   flow_id):
    """
    Prepares the flow and then creates a flow version and flow alias.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        flowd_id (str): The ID of the flow that you want to prepare.
    Returns: The flow_version and flow_alias.

    """

    status = prepare_flow(bedrock_agent_client, flow_id)

    flow_version = None
    flow_alias = None

    if status == 'Prepared':

        # Create the flow version and alias.
        flow_version = create_flow_version(bedrock_agent_client,
                                           flow_id,
                                           f"flow version for flow {flow_id}.")

        flow_alias = create_flow_alias(bedrock_agent_client,
                                       flow_id,
                                       flow_version,
                                       "latest",
```

```
                                        f"Alias for flow {flow_id}, version
 {flow_version}")

    return flow_version, flow_alias



def delete_role_resources(bedrock_agent_client,
                          iam_client,
                          role_name,
                          flow_id,
                          flow_version,
                          flow_alias):
    """
    Deletes the flow, flow alias, flow version, and IAM roles.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        iam_client: Amazon IAM boto3 client.
        role_name (str): The name of the IAM role.
        flow_id (str): The id of the flow.
        flow_version (str): The version of the flow.
        flow_alias (str): The alias of the flow.
    """

    if flow_id is not None:
        if flow_alias is not None:
            delete_flow_alias(bedrock_agent_client, flow_id, flow_alias)
        if flow_version is not None:
            delete_flow_version(bedrock_agent_client,
                        flow_id, flow_version)
        delete_flow(bedrock_agent_client, flow_id)

    if role_name is not None:
        delete_flow_role(iam_client, role_name)



def main():
    """
    Creates, runs, and optionally deletes a Bedrock flow for generating music
 playlists.

    Note:
        Requires valid AWS credentials in the default profile
```

```
    """

    delete_choice = "y"
    try:

        # Get various boto3 clients.
        session = boto3.Session(profile_name='default')
        bedrock_agent_runtime_client = session.client('bedrock-agent-runtime')
        bedrock_agent_client = session.client('bedrock-agent')
        bedrock_client = session.client('bedrock')
        iam_client = session.client('iam')

        role_name = None
        flow_id = None
        flow_version = None
        flow_alias = None

        #Change the model as needed.
        prompt_model_id = "amazon.nova-pro-v1:0"

        # Base the flow name on the current date and time
        current_time = datetime.now()
        timestamp = current_time.strftime("%Y-%m-%d-%H-%M-%S")
        flow_name = f"FlowPlayList_{timestamp}"
        flow_description = "A flow to generate a music playlist."

        # Create a role for the flow.
        role_name = f"BedrockFlowRole-{flow_name}"
        role = create_flow_role(iam_client, role_name)
        role_arn = role['Arn']

        # Create the flow.
        response = create_playlist_flow(
            bedrock_agent_client, flow_name, flow_description, role_arn,
prompt_model_id)
        flow_id = response.get('id')

        if flow_id:
            # Update accessible resources in the role.
            model_arn = get_model_arn(bedrock_client, prompt_model_id)
            update_role_policy(iam_client, role_name, [
                               response.get('arn'), model_arn])

            # Prepare the flow and flow version.
```

```
            flow_version, flow_alias = prepare_flow_version_and_alias(
                bedrock_agent_client, flow_id)

            # Run the flow.
            if flow_version and flow_alias:
                run_playlist_flow(bedrock_agent_runtime_client,
                                  flow_id, flow_alias)

                delete_choice = input("Delete flow? y or n : ").lower()


            else:
                print("Couldn't run. Deleting flow and role.")
                delete_flow(bedrock_agent_client, flow_id)
                delete_flow_role(iam_client, role_name)
        else:
            print("Couldn't create flow.")


    except Exception as e:
        print(f"Fatal error: {str(e)}")

    finally:
        if delete_choice == 'y':
                delete_role_resources(bedrock_agent_client,
                                      iam_client,
                                      role_name,
                                      flow_id,
                                      flow_version,
                                      flow_alias)
        else:
            print("Flow not deleted. ")
            print(f"\tFlow ID: {flow_id}")
            print(f"\tFlow version: {flow_version}")
            print(f"\tFlow alias: {flow_alias}")
            print(f"\tRole ARN: {role_arn}")

        print("Done!")

if __name__ == "__main__":
    main()


def invoke_flow(client, flow_id, flow_alias_id, input_data):
```

```
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.

    Returns:
        Dict containing flow status and flow output.
    """

    response = None
    request_params = None

    request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }


    response = client.invoke_flow(**request_params)

    flow_status = ""
    output= ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Save the model output.
        elif 'flowOutputEvent' in event:
            output = event['flowOutputEvent']['content']['document']
            logger.info("Output : %s", output)

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])
```

```
    return {
        "flow_status": flow_status,
        "output": output

    }




def run_playlist_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Runs the playlist generator flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """


    print ("Welcome to the playlist generator flow.")
    # Get the initial prompt from the user.
    genre = input("Enter genre: ")
    number_of_songs = int(input("Enter number of songs: "))


    # Use prompt to create input data for the input node.
    flow_input_data = {
        "content": {
            "document": {
                "genre" : genre,
                "number" : number_of_songs
            }
        },
        "nodeName": "FlowInput",
        "nodeOutputName": "document"
    }

    try:

        result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data)
```

```
        status = result['flow_status']

        if status == "SUCCESS":
                # The flow completed successfully.
                logger.info("The flow %s successfully completed.", flow_id)
                print(result['output'])
        else:
            logger.warning("Flow status: %s",status)

    except ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})
        raise

    except Exception as e:
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})
        raise



def create_flow_role(client, role_name):
    """
    Creates an IAM role for Amazon Bedrock with permissions to run a flow.

    Args:
        role_name (str): Name for the new IAM role.
    Returns:
        str: The role Amazon Resource Name.
    """



    # Trust relationship policy - allows Amazon Bedrock service to assume this
 role.
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "bedrock.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }]
```

```
    }

    # Basic inline policy for for running a flow.

    resources = "*"

    bedrock_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                # Using * as placeholder - Later you update with specific ARNs.
                "Resource": resources
            }
        ]
    }



    try:
        # Create the IAM role with trust policy
        logging.info("Creating role: %s",role_name)
        role = client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy),
            Description="Role for Amazon Bedrock operations"
        )

        # Attach inline policy to the role
        print("Attaching inline policy")
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(bedrock_policy)
        )

        logging.info("Create Role ARN: %s", role['Role']['Arn'])
        return role['Role']
```

```
    except ClientError as e:
        logging.warning("Error creating role: %s", str(e))
        raise
    except Exception as e:
        logging.warning("Unexpected error: %s", str(e))
        raise


def update_role_policy(client, role_name, resource_arns):
    """
    Updates an IAM role's inline policy with specific resource ARNs.

    Args:
        role_name (str): Name of the existing role.
        resource_arns (list): List of resource ARNs to allow access to.
    """



    updated_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:GetFlow",
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                "Resource": resource_arns
            }
        ]
    }

    try:
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(updated_policy)
        )
        logging.info("Updated policy for role: %s",role_name)

    except ClientError as e:
        logging.warning("Error updating role policy: %s", str(e))
```

```
        raise


def delete_flow_role(client, role_name):
    """
    Deletes an IAM role.

    Args:
        role_name (str): Name of the role to delete.
    """




    try:
        # Detach and delete inline policies
        policies = client.list_role_policies(RoleName=role_name)['PolicyNames']
        for policy_name in policies:
            client.delete_role_policy(RoleName=role_name, PolicyName=policy_name)

        # Delete the role
        client.delete_role(RoleName=role_name)
        logging.info("Deleted role: %s", role_name)


    except ClientError as e:
        logging.info("Error Deleting role: %s", str(e))
        raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateFlow
  - CreateFlowAlias
  - CreateFlowVersion
  - DeleteFlow
  - DeleteFlowAlias
  - DeleteFlowVersion
  - GetFlow
  - GetFlowAlias

- GetFlowVersion
- InvokeFlow
- PrepareFlow

Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

Python

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Step Functions

Exemplos de código para AWS Budgets usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Budgets com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS Budgets Guia do desenvolvedor — Mais informações sobre AWS Budgets.
- AWS Budgets Referência da API — Detalhes sobre todas as AWS Budgets ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Budgets usar AWS SDKs
  - Ações para AWS Budgets usar AWS SDKs
    - Usar CreateBudget com uma CLI

Exemplos básicos para AWS Budgets usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Budgets com AWS SDKs.

Exemplos

- Ações para AWS Budgets usar AWS SDKs
  - Usar CreateBudget com uma CLI

Ações para AWS Budgets usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Budgets ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Budgets.

Exemplos

- Usar CreateBudget com uma CLI

Usar **`CreateBudget`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateBudget`.

CLI

AWS CLI

Para criar um orçamento de custo e uso

O comando `create-budget` a seguir cria um orçamento de custo e uso.

```
aws budgets create-budget \
    --account-id 111122223333 \
    --budget file://budget.json \
    --notifications-with-subscribers file://notifications-with-subscribers.json
```

Conteúdo de `budget.json`:

```
{
    "BudgetLimit": {
        "Amount": "100",
        "Unit": "USD"
    },
    "BudgetName": "Example Tag Budget",
    "BudgetType": "COST",
    "CostFilters": {
        "TagKeyValue": [
            "user:Key$value1",
            "user:Key$value2"
        ]
    },
    "CostTypes": {
        "IncludeCredit": true,
        "IncludeDiscount": true,
        "IncludeOtherSubscription": true,
        "IncludeRecurring": true,
        "IncludeRefund": true,
        "IncludeSubscription": true,
        "IncludeSupport": true,
        "IncludeTax": true,
        "IncludeUpfront": true,
        "UseBlended": false
    },
    "TimePeriod": {
        "Start": 1477958399,
        "End": 3706473600
    },
    "TimeUnit": "MONTHLY"
}
```

Conteúdo de `notifications-with-subscribers.json`:

```
[
    {
        "Notification": {
            "ComparisonOperator": "GREATER_THAN",
            "NotificationType": "ACTUAL",
            "Threshold": 80,
            "ThresholdType": "PERCENTAGE"
        },
        "Subscribers": [
```

```
            {
                "Address": "example@example.com",
                "SubscriptionType": "EMAIL"
            }
        ]
    }
]
```

- Para obter detalhes da API, consulte CreateBudgetem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria um orçamento com as restrições orçamentárias e de tempo especificadas com notificações por e-mail.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @("user@domain.com")
    SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- Para obter detalhes da API, consulte CreateBudgetem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria um orçamento com as restrições orçamentárias e de tempo especificadas com notificações por e-mail.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @("user@domain.com")
    SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- Para obter detalhes da API, consulte CreateBudgetem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para AWS Cloud Map usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Cloud Map com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS Cloud Map Guia do desenvolvedor — Mais informações sobre AWS Cloud Map.

- AWS Cloud Map Referência da API — Detalhes sobre todas as AWS Cloud Map ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para AWS Cloud Map usar AWS SDKs
  - Configurar a conectividade do serviço de contêiner
  - Comece com a descoberta de serviços
  - Usando a descoberta de serviços com atributos personalizados

Cenários para AWS Cloud Map usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS Cloud Map com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS Cloud Map ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Configurar a conectividade do serviço de contêiner
- Comece com a descoberta de serviços
- Usando a descoberta de serviços com atributos personalizados

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log

- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
```

```
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
```

```
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."
```

```
    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
```

```
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."
```

```
    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
```

```
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"
```

```
        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
```

```
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
```

```
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
```

```
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
```

```
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
```

```
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"
```

```
    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
```

```
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity

- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Comece com a descoberta de serviços

O exemplo de código a seguir mostra como:

- Criar um namespace de DNS público com a integração de zona hospedada do Route 53.
- Criar serviços detectáveis por meio de consultas ao DNS e chamadas de API.
- Registrar instâncias de serviço com diferentes configurações de descoberta.
- Descobrir serviços usando consultas ao DNS e a API do Cloud Map.
- Verificar a descoberta de serviços usando o comando dig e chamadas de API.
- Limpar os recursos na ordem correta (instâncias, serviços e namespace).

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Cloud Map Private Namespace Tutorial Script
# This script demonstrates how to use AWS Cloud Map for service discovery
# with DNS queries and API calls
```

```
# Exit on error
set -e

# Configuration
REGION="us-east-2"
NAMESPACE_NAME="cloudmap-tutorial.com"
LOG_FILE="cloudmap-tutorial.log"
CREATOR_REQUEST_ID=$(date +%s)

# Function to log messages
log() {
    local message="$1"
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $message" | tee -a "$LOG_FILE"
}

# Function to check operation status
check_operation() {
    local operation_id="$1"
    local status=""

    log "Checking operation status for $operation_id..."

    while [[ "$status" != "SUCCESS" ]]; do
        sleep 5
        status=$(aws servicediscovery get-operation \
            --operation-id "$operation_id" \
            --region "$REGION" \
            --query "Operation.Status" \
            --output text)

        log "Operation status: $status"

        if [[ "$status" == "FAIL" ]]; then
            log "Operation failed. Exiting."
            exit 1
        fi
    done

    log "Operation completed successfully."
}

# Function to clean up resources
cleanup() {
```

```
    log "Starting cleanup process..."

    if [[ -n "$FIRST_INSTANCE_ID" ]]; then
        log "Deregistering first service instance..."
        aws servicediscovery deregister-instance \
            --service-id "$PUBLIC_SERVICE_ID" \
            --instance-id "$FIRST_INSTANCE_ID" \
            --region "$REGION" || log "Failed to deregister first instance"
    fi

    if [[ -n "$SECOND_INSTANCE_ID" ]]; then
        log "Deregistering second service instance..."
        aws servicediscovery deregister-instance \
            --service-id "$BACKEND_SERVICE_ID" \
            --instance-id "$SECOND_INSTANCE_ID" \
            --region "$REGION" || log "Failed to deregister second instance"
    fi

    if [[ -n "$PUBLIC_SERVICE_ID" ]]; then
        log "Deleting public service..."
        aws servicediscovery delete-service \
            --id "$PUBLIC_SERVICE_ID" \
            --region "$REGION" || log "Failed to delete public service"
    fi

    if [[ -n "$BACKEND_SERVICE_ID" ]]; then
        log "Deleting backend service..."
        aws servicediscovery delete-service \
            --id "$BACKEND_SERVICE_ID" \
            --region "$REGION" || log "Failed to delete backend service"
    fi

    if [[ -n "$NAMESPACE_ID" ]]; then
        log "Deleting namespace..."
        aws servicediscovery delete-namespace \
            --id "$NAMESPACE_ID" \
            --region "$REGION" || log "Failed to delete namespace"
    fi

    log "Cleanup completed."
}

# Set up trap for cleanup on script exit
trap cleanup EXIT INT TERM
```

```
# Initialize log file
> "$LOG_FILE"
log "Starting AWS Cloud Map tutorial script"

# Step 1: Create an AWS Cloud Map namespace
log "Creating AWS Cloud Map namespace: $NAMESPACE_NAME"
OPERATION_RESULT=$(aws servicediscovery create-public-dns-namespace \
    --name "$NAMESPACE_NAME" \
    --creator-request-id "cloudmap-tutorial-$CREATOR_REQUEST_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-
discovery \
    --region "$REGION")

OPERATION_ID=$(echo "$OPERATION_RESULT" | jq -r '.OperationId')
log "Namespace creation initiated. Operation ID: $OPERATION_ID"

# Check operation status
check_operation "$OPERATION_ID"

# Get the namespace ID
log "Getting namespace ID..."
NAMESPACE_ID=$(aws servicediscovery list-namespaces \
    --region "$REGION" \
    --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" \
    --output text)

log "Namespace ID: $NAMESPACE_ID"

# Get the hosted zone ID
log "Getting Route 53 hosted zone ID..."
HOSTED_ZONE_ID=$(aws route53 list-hosted-zones-by-name \
    --dns-name "$NAMESPACE_NAME" \
    --query "HostedZones[0].Id" \
    --output text | sed 's|/hostedzone/||')

log "Hosted Zone ID: $HOSTED_ZONE_ID"

# Step 2: Create the AWS Cloud Map services
log "Creating public service..."
PUBLIC_SERVICE_RESULT=$(aws servicediscovery create-service \
    --name "public-service" \
    --namespace-id "$NAMESPACE_ID" \
    --dns-config "RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=A,TTL=300}]" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-
discovery \
    --region "$REGION")

PUBLIC_SERVICE_ID=$(echo "$PUBLIC_SERVICE_RESULT" | jq -r '.Service.Id')
log "Public service created. Service ID: $PUBLIC_SERVICE_ID"

log "Creating backend service..."
BACKEND_SERVICE_RESULT=$(aws servicediscovery create-service \
    --name "backend-service" \
    --namespace-id "$NAMESPACE_ID" \
    --type "HTTP" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-
discovery \
    --region "$REGION")

BACKEND_SERVICE_ID=$(echo "$BACKEND_SERVICE_RESULT" | jq -r '.Service.Id')
log "Backend service created. Service ID: $BACKEND_SERVICE_ID"

# Step 3: Register the AWS Cloud Map service instances
log "Registering first service instance..."
FIRST_INSTANCE_RESULT=$(aws servicediscovery register-instance \
    --service-id "$PUBLIC_SERVICE_ID" \
    --instance-id "first" \
    --attributes "AWS_INSTANCE_IPV4=192.168.2.1" \
    --region "$REGION")

FIRST_INSTANCE_ID="first"
FIRST_OPERATION_ID=$(echo "$FIRST_INSTANCE_RESULT" | jq -r '.OperationId')
log "First instance registration initiated. Operation ID: $FIRST_OPERATION_ID"

# Check operation status
check_operation "$FIRST_OPERATION_ID"

log "Registering second service instance..."
SECOND_INSTANCE_RESULT=$(aws servicediscovery register-instance \
    --service-id "$BACKEND_SERVICE_ID" \
    --instance-id "second" \
    --attributes "service-name=backend" \
    --region "$REGION")

SECOND_INSTANCE_ID="second"
SECOND_OPERATION_ID=$(echo "$SECOND_INSTANCE_RESULT" | jq -r '.OperationId')
log "Second instance registration initiated. Operation ID: $SECOND_OPERATION_ID"
```

```
# Check operation status
check_operation "$SECOND_OPERATION_ID"

# Step 4: Discover the AWS Cloud Map service instances
log "Getting Route 53 name servers..."
NAME_SERVERS=$(aws route53 get-hosted-zone \
    --id "$HOSTED_ZONE_ID" \
    --query "DelegationSet.NameServers[0]" \
    --output text)

log "Name server: $NAME_SERVERS"

log "Using dig to query DNS records (this will be simulated)..."
log "Command: dig @$NAME_SERVERS public-service.$NAMESPACE_NAME"
log "Expected output would show: public-service.$NAMESPACE_NAME. 300 IN A
 192.168.2.1"

log "Using AWS CLI to discover backend service instances..."
DISCOVER_RESULT=$(aws servicediscovery discover-instances \
    --namespace-name "$NAMESPACE_NAME" \
    --service-name "backend-service" \
    --region "$REGION")

log "Discovery result: $(echo "$DISCOVER_RESULT" | jq -c '.')"

# Display created resources
log "Resources created:"
log "- Namespace: $NAMESPACE_NAME (ID: $NAMESPACE_ID)"
log "- Public Service: public-service (ID: $PUBLIC_SERVICE_ID)"
log "- Backend Service: backend-service (ID: $BACKEND_SERVICE_ID)"
log "- Service Instance: first (Service: public-service)"
log "- Service Instance: second (Service: backend-service)"

# Ask user if they want to clean up resources
read -p "Do you want to clean up all created resources? (y/n): " CLEANUP_RESPONSE

if [[ "$CLEANUP_RESPONSE" == "y" || "$CLEANUP_RESPONSE" == "Y" ]]; then
    log "User confirmed cleanup. Proceeding with resource deletion."
    # Cleanup function will be called automatically on exit
else
    log "User chose not to clean up resources. Exiting without cleanup."
    trap - EXIT
    exit 0
```

```
fi
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreatePublicDnsNamespace
  - CreateService
  - DeleteNamespace
  - DeleteService
  - DeregisterInstance
  - DiscoverInstances
  - GetOperation
  - ListNamespaces
  - RegisterInstance

Usando a descoberta de serviços com atributos personalizados

O exemplo de código a seguir mostra como:

- Criar um namespace do Cloud Map.
- Criar uma tabela do DynamoDB.
- Criar um serviço de dados do Cloud Map e registrar a tabela do DynamoDB.
- Criar um perfil do IAM para funções do Lambda.
- Criar a função do Lambda para gravar dados.
- Criar um serviço de aplicação do Cloud Map e registrar a função de gravação do Lambda.
- Criar a função do Lambda para ler dados.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Cloud Map Tutorial Script
# This script demonstrates how to use AWS Cloud Map for service discovery with
 custom attributes

set -euo pipefail

# Set up logging
LOG_FILE="cloudmap-tutorial.log"
echo "AWS Cloud Map Tutorial Script" > "$LOG_FILE"
echo "Started at $(date)" >> "$LOG_FILE"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log commands and their output
log_cmd() {
  echo "$ $1" | tee -a "$LOG_FILE"
  eval "$1" | tee -a "$LOG_FILE"
}

# Function to handle errors
handle_error() {
  local LINE=$1
  echo "An error occurred at line $LINE" | tee -a "$LOG_FILE"
  echo "Resources created so far:" | tee -a "$LOG_FILE"
  for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
  done
  echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"
  cleanup
  exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Helper function to wait for Cloud Map operations to complete
wait_for_operation() {
  local OPERATION_ID=$1
  local TIMEOUT=300  # 5 minutes timeout
  local START_TIME
```

```
  START_TIME=$(date +%s)

  while true; do
    local STATUS
    STATUS=$(aws servicediscovery get-operation --operation-id "$OPERATION_ID" --
query 'Operation.Status' --output text 2>/dev/null || echo "UNKNOWN")

    if [ "$STATUS" == "SUCCESS" ]; then
      echo "Operation completed successfully" | tee -a "$LOG_FILE"
      break
    elif [ "$STATUS" == "FAIL" ]; then
      echo "Operation failed" | tee -a "$LOG_FILE"
      return 1
    fi

    local CURRENT_TIME
    CURRENT_TIME=$(date +%s)
    if [ $((CURRENT_TIME - START_TIME)) -gt $TIMEOUT ]; then
      echo "Operation timed out" | tee -a "$LOG_FILE"
      return 1
    fi

    sleep 5
  done

  return 0
}

# Function to clean up resources
cleanup() {
  echo "Cleaning up resources..." | tee -a "$LOG_FILE"

  # Reverse the order of created resources for proper deletion
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    echo "Deleting $resource..." | tee -a "$LOG_FILE"

    if [[ $resource == "instance:"* ]]; then
      # Extract service ID and instance ID
      SERVICE_ID=$(echo "$resource" | cut -d':' -f2)
      INSTANCE_ID=$(echo "$resource" | cut -d':' -f3)

      # Check if instance exists before trying to deregister
```

```
      INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$SERVICE_ID" --query "Instances[?Id=='$INSTANCE_ID'].Id" --output text 2>/dev/
null || echo "")
      if [[ -n "$INSTANCE_EXISTS" ]]; then
        OPERATION_ID=$(aws servicediscovery deregister-instance --service-id
 "$SERVICE_ID" --instance-id "$INSTANCE_ID" --query 'OperationId' --output text)

        # Wait for deregistration to complete
        echo "Waiting for instance deregistration to complete..." | tee -a
 "$LOG_FILE"
        wait_for_operation "$OPERATION_ID" || true
      else
        echo "Instance $INSTANCE_ID already deregistered" | tee -a "$LOG_FILE"
      fi
    elif [[ $resource == "lambda:"* ]]; then
      # Extract function name
      FUNCTION_NAME=$(echo "$resource" | cut -d':' -f2)
      aws lambda delete-function --function-name "$FUNCTION_NAME" 2>/dev/null ||
 echo "Function already deleted" | tee -a "$LOG_FILE"
    elif [[ $resource == "role:"* ]]; then
      # Extract role name
      ROLE_NAME=$(echo "$resource" | cut -d':' -f2)

      # Detach all policies first
      for POLICY_ARN in $(aws iam list-attached-role-policies --role-name
 "$ROLE_NAME" --query 'AttachedPolicies[*].PolicyArn' --output text); do
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
      done

      # Delete the role
      aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Role
 already deleted" | tee -a "$LOG_FILE"
    elif [[ $resource == "dynamodb:"* ]]; then
      # Extract table name
      TABLE_NAME=$(echo "$resource" | cut -d':' -f2)
      aws dynamodb delete-table --table-name "$TABLE_NAME" 2>/dev/null || echo
 "Table already deleted" | tee -a "$LOG_FILE"

      # Wait for table deletion to complete
      echo "Waiting for DynamoDB table deletion to complete..." | tee -a
 "$LOG_FILE"
      aws dynamodb wait table-not-exists --table-name "$TABLE_NAME" 2>/dev/null
 || true
```

```
    fi
  done

  # Handle services separately to ensure all instances are deregistered first
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    if [[ $resource == "service:"* ]]; then
      # Extract service ID
      SERVICE_ID=$(echo "$resource" | cut -d':' -f2)
      echo "Deleting service $SERVICE_ID..." | tee -a "$LOG_FILE"

      # Make sure all instances are deregistered
      INSTANCES=$(aws servicediscovery list-instances --service-id "$SERVICE_ID"
--query 'Instances[*].Id' --output text 2>/dev/null || echo "")
      if [[ -n "$INSTANCES" ]]; then
        echo "Service still has instances. Waiting before deletion..." | tee -a
"$LOG_FILE"
        sleep 10
      fi

      # Try to delete the service
      aws servicediscovery delete-service --id "$SERVICE_ID" 2>/dev/null || echo
"Service already deleted" | tee -a "$LOG_FILE"
      sleep 5
    fi
  done

  # Handle namespaces last to ensure all services are deleted first
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    if [[ $resource == "namespace:"* ]]; then
      # Extract namespace ID
      NAMESPACE_ID=$(echo "$resource" | cut -d':' -f2)
      echo "Deleting namespace $NAMESPACE_ID..." | tee -a "$LOG_FILE"

      # Check if namespace still has services
      SERVICES=$(aws servicediscovery list-services --filters
"Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query 'Services[*].Id'
--output text 2>/dev/null || echo "")
      if [[ -n "$SERVICES" ]]; then
        echo "Namespace still has services. Deleting them first..." | tee -a
"$LOG_FILE"
        for SERVICE_ID in $SERVICES; do
          echo "Deleting service $SERVICE_ID..." | tee -a "$LOG_FILE"
```

```
          aws servicediscovery delete-service --id "$SERVICE_ID" 2>/dev/null ||
 true
        done
        sleep 5
      fi

      # Try to delete the namespace
      OPERATION_ID=$(aws servicediscovery delete-namespace --id "$NAMESPACE_ID"
 --query 'OperationId' --output text 2>/dev/null || echo "")
      if [[ -n "$OPERATION_ID" ]]; then
        echo "Waiting for namespace deletion to complete..." | tee -a "$LOG_FILE"
        wait_for_operation "$OPERATION_ID" || true
      else
        echo "Failed to delete namespace or namespace already deleted" | tee -a
 "$LOG_FILE"
      fi
    fi
  done

  echo "Cleanup complete" | tee -a "$LOG_FILE"
}

# Step 1: Create an AWS Cloud Map namespace
echo "Step 1: Creating AWS Cloud Map namespace..." | tee -a "$LOG_FILE"

# Check if namespace already exists
NAMESPACE_ID=$(aws servicediscovery list-namespaces --query "Namespaces[?
Name=='cloudmap-tutorial'].Id" --output text 2>/dev/null || echo "")

if [[ -z "$NAMESPACE_ID" || "$NAMESPACE_ID" == "None" ]]; then
  log_cmd "aws servicediscovery create-http-namespace --name cloudmap-tutorial --
creator-request-id namespace-request-\$(date +%s)"
  OPERATION_ID=$(aws servicediscovery create-http-namespace --name cloudmap-
tutorial --creator-request-id "namespace-request-$(date +%s)" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-custom-attributes --
query 'OperationId' --output text)

  # Wait for namespace creation to complete
  echo "Waiting for namespace creation to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"

  # Get the namespace ID
  NAMESPACE_ID=$(aws servicediscovery list-namespaces --query "Namespaces[?
Name=='cloudmap-tutorial'].Id" --output text)
```

```
  echo "Namespace created with ID: $NAMESPACE_ID" | tee -a "$LOG_FILE"
else
  echo "Namespace cloudmap-tutorial already exists with ID: $NAMESPACE_ID" | tee
 -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("namespace:$NAMESPACE_ID")

# Step 2: Create a DynamoDB table
echo "Step 2: Creating DynamoDB table..." | tee -a "$LOG_FILE"

# Check if table already exists
TABLE_EXISTS=$(aws dynamodb describe-table --table-name cloudmap 2>&1 || echo
 "NOT_EXISTS")

if [[ $TABLE_EXISTS == *"ResourceNotFoundException"* || $TABLE_EXISTS ==
 "NOT_EXISTS" ]]; then
  log_cmd "aws dynamodb create-table --table-name cloudmap --
attribute-definitions AttributeName=id,AttributeType=S --key-schema
 AttributeName=id,KeyType=HASH --billing-mode PAY_PER_REQUEST --tags
 Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-custom-attributes"

  # Wait for DynamoDB table to become active
  echo "Waiting for DynamoDB table to become active..." | tee -a "$LOG_FILE"
  aws dynamodb wait table-exists --table-name cloudmap
else
  echo "DynamoDB table cloudmap already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("dynamodb:cloudmap")

# Step 3: Create an AWS Cloud Map data service
echo "Step 3: Creating AWS Cloud Map data service..." | tee -a "$LOG_FILE"

# Get all services in the namespace
echo "Listing all services in namespace $NAMESPACE_ID..." | tee -a "$LOG_FILE"
SERVICES=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query 'Services[*].
[Id,Name]' --output text 2>/dev/null || echo "")
echo "Services found: $SERVICES" | tee -a "$LOG_FILE"

# Check if data service already exists
DATA_SERVICE_ID=""
while read -r id name || [[ -n "$id" ]]; do
```

```
  echo "Checking service: ID=$id, Name=$name" | tee -a "$LOG_FILE"
  if [[ "$name" == "data-service" ]]; then
    DATA_SERVICE_ID="$id"
    break
  fi
done <<< "$SERVICES"

if [[ -z "$DATA_SERVICE_ID" ]]; then
  echo "Data service does not exist, creating it..." | tee -a "$LOG_FILE"
  # Create the service and capture the ID directly
  echo "$ aws servicediscovery create-service --name data-service --namespace-id
 $NAMESPACE_ID --creator-request-id data-service-request-\$(date +%s)" | tee -a
 "$LOG_FILE"
  CREATE_OUTPUT=$(aws servicediscovery create-service --name data-service
 --namespace-id "$NAMESPACE_ID" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes --creator-request-id "data-
service-request-$(date +%s)")
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"

  # Extract the service ID using AWS CLI query
  DATA_SERVICE_ID=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query "Services[?
Name=='data-service'].Id" --output text)
  echo "Data service created with ID: $DATA_SERVICE_ID" | tee -a "$LOG_FILE"
else
  echo "Data service already exists with ID: $DATA_SERVICE_ID" | tee -a
 "$LOG_FILE"
fi

CREATED_RESOURCES+=("service:$DATA_SERVICE_ID")

# Register DynamoDB table as a service instance
echo "Registering DynamoDB table as a service instance..." | tee -a "$LOG_FILE"

# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$DATA_SERVICE_ID" --query "Instances[?Id=='data-instance'].Id" --output text
 2>/dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  AWS_REGION=$(aws configure get region || echo "us-east-1")
  log_cmd "aws servicediscovery register-instance --service-id $DATA_SERVICE_ID
 --instance-id data-instance --attributes tablename=cloudmap,region=$AWS_REGION"
```

```
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$DATA_SERVICE_ID" --instance-id data-instance --attributes
 "tablename=cloudmap,region=$AWS_REGION" --query 'OperationId' --output text)

  # Wait for instance registration to complete
  echo "Waiting for instance registration to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance data-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$DATA_SERVICE_ID:data-instance")

# Step 4: Create an IAM role for Lambda
echo "Step 4: Creating IAM role for Lambda..." | tee -a "$LOG_FILE"

# Create a trust policy for Lambda
cat > lambda-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Check if role already exists
echo "Checking if IAM role already exists..." | tee -a "$LOG_FILE"
ROLE_EXISTS=$(aws iam get-role --role-name cloudmap-tutorial-role 2>&1 || echo
 "NOT_EXISTS")

if [[ $ROLE_EXISTS == *"NoSuchEntity"* || $ROLE_EXISTS == "NOT_EXISTS" ]]; then
    log_cmd "aws iam create-role --role-name cloudmap-tutorial-role --assume-
role-policy-document file://lambda-trust-policy.json"
    aws iam tag-role --role-name cloudmap-tutorial-role --tags
 Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-custom-attributes
else
```

```
    echo "Role cloudmap-tutorial-role already exists, using existing role" | tee
 -a "$LOG_FILE"
fi

# Create a custom policy with least privilege
cat > cloudmap-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:GetItem",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/cloudmap"
    },
    {
      "Effect": "Allow",
      "Action": [
        "servicediscovery:DiscoverInstances"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Check if policy already exists
POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='CloudMapTutorialPolicy'].Arn" --output text 2>/dev/null || echo "")

if [[ -z "$POLICY_ARN" ]]; then
  echo "Creating CloudMapTutorialPolicy..." | tee -a "$LOG_FILE"
```

```
  echo "$ aws iam create-policy --policy-name CloudMapTutorialPolicy --policy-
document file://cloudmap-policy.json" | tee -a "$LOG_FILE"
  CREATE_OUTPUT=$(aws iam create-policy --policy-name CloudMapTutorialPolicy --
policy-document file://cloudmap-policy.json)
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"
  POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='CloudMapTutorialPolicy'].Arn" --output text)
else
  echo "Policy CloudMapTutorialPolicy already exists with ARN: $POLICY_ARN" | tee
 -a "$LOG_FILE"
fi

echo "$ aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-
arn $POLICY_ARN" | tee -a "$LOG_FILE"
aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 "$POLICY_ARN" | tee -a "$LOG_FILE"

echo "$ aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-
arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole" | tee -a
 "$LOG_FILE"
aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole | tee -a
 "$LOG_FILE"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

ROLE_ARN=$(aws iam get-role --role-name cloudmap-tutorial-role --query 'Role.Arn'
 --output text)
CREATED_RESOURCES+=("role:cloudmap-tutorial-role")

# Step 5: Create an AWS Cloud Map app service
echo "Step 5: Creating AWS Cloud Map app service..." | tee -a "$LOG_FILE"

# Get all services in the namespace
SERVICES=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query 'Services[*].
[Id,Name]' --output text 2>/dev/null || echo "")

# Check if app service already exists
APP_SERVICE_ID=""
while read -r id name || [[ -n "$id" ]]; do
  if [[ "$name" == "app-service" ]]; then
```

```
    APP_SERVICE_ID="$id"
    break
  fi
done <<< "$SERVICES"

if [[ -z "$APP_SERVICE_ID" ]]; then
  echo "App service does not exist, creating it..." | tee -a "$LOG_FILE"
  # Create the service and capture the ID directly
  echo "$ aws servicediscovery create-service --name app-service --namespace-id
 $NAMESPACE_ID --creator-request-id app-service-request-\$(date +%s)" | tee -a
 "$LOG_FILE"
  CREATE_OUTPUT=$(aws servicediscovery create-service --name app-service
 --namespace-id "$NAMESPACE_ID" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes --creator-request-id "app-service-
request-$(date +%s)")
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"

  # Extract the service ID using AWS CLI query
  APP_SERVICE_ID=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query "Services[?
Name=='app-service'].Id" --output text)
  echo "App service created with ID: $APP_SERVICE_ID" | tee -a "$LOG_FILE"
else
  echo "App service already exists with ID: $APP_SERVICE_ID" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("service:$APP_SERVICE_ID")

# Step 6: Create a Lambda function to write data
echo "Step 6: Creating Lambda function to write data..." | tee -a "$LOG_FILE"

# Create Lambda function code
cat > writefunction.py << 'EOF'
import boto3
import json
import random
import os

def lambda_handler(event, context):
    # Use AWS Cloud Map to discover the DynamoDB table
    serviceclient = boto3.client('servicediscovery')

    try:
        # Discover the data service instance
```

```
        response = serviceclient.discover_instances(
            NamespaceName='cloudmap-tutorial',
            ServiceName='data-service'
        )

        if not response.get('Instances'):
            return {
                'statusCode': 404,
                'body': json.dumps('No service instances found')
            }

        # Extract table name and region from the instance attributes
        attributes = response['Instances'][0].get('Attributes', {})
        tablename = attributes.get('tablename', 'cloudmap')
        region = attributes.get('region', os.environ.get('AWS_REGION', 'us-
east-1'))

        # Create DynamoDB client in the specified region
        dynamodb = boto3.resource('dynamodb', region_name=region)
        table = dynamodb.Table(tablename)

        # Write data to the table
        table.put_item(
            Item={
                'id': str(random.randint(1, 100)),
                'todo': str(event)
            }
        )

        return {
            'statusCode': 200,
            'body': json.dumps('Data written successfully!')
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps(f'Error: {str(e)}')
        }
EOF

# Zip the function code
log_cmd "zip writefunction.zip writefunction.py"

# Create the Lambda function
```

```
FUNCTION_EXISTS=$(aws lambda list-functions --query "Functions[?
FunctionName=='writefunction'].FunctionName" --output text 2>/dev/null || echo
 "")
if [[ -z "$FUNCTION_EXISTS" ]]; then
  log_cmd "aws lambda create-function --function-name writefunction --runtime
 python3.12 --role $ROLE_ARN --handler writefunction.lambda_handler --zip-
file fileb://writefunction.zip --architectures x86_64 --timeout 10 --tags
 project=doc-smith,tutorial=cloudmap-custom-attributes"

  # Wait for the Lambda function to be active before updating
  echo "Waiting for Lambda function to become active..." | tee -a "$LOG_FILE"
  function_state="Pending"
  while [ "$function_state" == "Pending" ]; do
      sleep 5
      function_state=$(aws lambda get-function --function-name writefunction --
query 'Configuration.State' --output text)
      echo "Current function state: $function_state" | tee -a "$LOG_FILE"
  done
else
  echo "Lambda function writefunction already exists" | tee -a "$LOG_FILE"
fi
CREATED_RESOURCES+=("lambda:writefunction")

# Step 7: Register the Lambda write function as an AWS Cloud Map service instance
echo "Step 7: Registering Lambda write function as a service instance..." | tee -
a "$LOG_FILE"

# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$APP_SERVICE_ID" --query "Instances[?Id=='write-instance'].Id" --output text
 2>/dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  log_cmd "aws servicediscovery register-instance --service-id $APP_SERVICE_ID --
instance-id write-instance --attributes action=write,functionname=writefunction"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$APP_SERVICE_ID" --instance-id write-instance --attributes
 action=write,functionname=writefunction --query 'OperationId' --output text)

  # Wait for instance registration to complete
  echo "Waiting for write instance registration to complete..." | tee -a
 "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
```

```
  echo "Instance write-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$APP_SERVICE_ID:write-instance")

# Step 8: Create a Lambda function to read data
echo "Step 8: Creating Lambda function to read data..." | tee -a "$LOG_FILE"

# Create Lambda function code
cat > readfunction.py << 'EOF'
import boto3
import json
import os

def lambda_handler(event, context):
    # Use AWS Cloud Map to discover the DynamoDB table
    serviceclient = boto3.client('servicediscovery')

    try:
        # Discover the data service instance
        response = serviceclient.discover_instances(
            NamespaceName='cloudmap-tutorial',
            ServiceName='data-service'
        )

        if not response.get('Instances'):
            return {
                'statusCode': 404,
                'body': json.dumps('No service instances found')
            }

        # Extract table name and region from the instance attributes
        attributes = response['Instances'][0].get('Attributes', {})
        tablename = attributes.get('tablename', 'cloudmap')
        region = attributes.get('region', os.environ.get('AWS_REGION', 'us-
east-1'))

        # Create DynamoDB client in the specified region
        dynamodb = boto3.resource('dynamodb', region_name=region)
        table = dynamodb.Table(tablename)

        # Read data from the table
        response = table.scan()
```

```
        return {
            'statusCode': 200,
            'body': json.dumps(response.get('Items', []))
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps(f'Error: {str(e)}')
        }
EOF

# Zip the function code
log_cmd "zip readfunction.zip readfunction.py"

# Create the Lambda function
FUNCTION_EXISTS=$(aws lambda list-functions --query "Functions[?
FunctionName=='readfunction'].FunctionName" --output text 2>/dev/null || echo "")
if [[ -z "$FUNCTION_EXISTS" ]]; then
  log_cmd "aws lambda create-function --function-name readfunction --runtime
 python3.12 --role $ROLE_ARN --handler readfunction.lambda_handler --zip-file
 fileb://readfunction.zip --architectures x86_64 --timeout 10 --tags project=doc-
smith,tutorial=cloudmap-custom-attributes"

  # Wait for the Lambda function to be active before updating
  echo "Waiting for Lambda function to become active..." | tee -a "$LOG_FILE"
  function_state="Pending"
  while [ "$function_state" == "Pending" ]; do
      sleep 5
      function_state=$(aws lambda get-function --function-name readfunction --
query 'Configuration.State' --output text)
      echo "Current function state: $function_state" | tee -a "$LOG_FILE"
  done
else
  echo "Lambda function readfunction already exists" | tee -a "$LOG_FILE"
fi
CREATED_RESOURCES+=("lambda:readfunction")

# Step 9: Register the Lambda read function as an AWS Cloud Map service instance
echo "Step 9: Registering Lambda read function as a service instance..." | tee -a
 "$LOG_FILE"

# Check if instance already exists
```

```
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$APP_SERVICE_ID" --query "Instances[?Id=='read-instance'].Id" --output text 2>/
dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  log_cmd "aws servicediscovery register-instance --service-id $APP_SERVICE_ID --
instance-id read-instance --attributes action=read,functionname=readfunction"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$APP_SERVICE_ID" --instance-id read-instance --attributes
 action=read,functionname=readfunction --query 'OperationId' --output text)

  # Wait for read instance registration to complete
  echo "Waiting for read instance registration to complete..." | tee -a
 "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance read-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$APP_SERVICE_ID:read-instance")

# Step 10: Create Python clients to interact with the services
echo "Step 10: Creating Python clients..." | tee -a "$LOG_FILE"

cat > writeclient.py << 'EOF'
#!/usr/bin/env python3
import boto3
import json
import sys

try:
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='app-service',
        QueryParameters={'action': 'write'}
    )

    if not response.get('Instances'):
        print("No write service instances found", file=sys.stderr)
        sys.exit(1)

    functionname = response["Instances"][0]["Attributes"]["functionname"]
```

```
    lambdaclient = boto3.client('lambda')

    resp = lambdaclient.invoke(
        FunctionName=functionname,
        Payload=json.dumps('This is a test data'),
        InvocationType='RequestResponse'
    )

    print(resp["Payload"].read().decode())
except Exception as e:
    print(f"Error: {str(e)}", file=sys.stderr)
    sys.exit(1)
EOF

cat > readclient.py << 'EOF'
#!/usr/bin/env python3
import boto3
import json
import sys

try:
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='app-service',
        QueryParameters={'action': 'read'}
    )

    if not response.get('Instances'):
        print("No read service instances found", file=sys.stderr)
        sys.exit(1)

    functionname = response["Instances"][0]["Attributes"]["functionname"]

    lambdaclient = boto3.client('lambda')

    resp = lambdaclient.invoke(
        FunctionName=functionname,
        InvocationType='RequestResponse'
    )

    print(resp["Payload"].read().decode())
```

```
except Exception as e:
    print(f"Error: {str(e)}", file=sys.stderr)
    sys.exit(1)
EOF

chmod +x writeclient.py readclient.py

echo "Running write client..." | tee -a "$LOG_FILE"
log_cmd "python3 writeclient.py" || echo "Write client execution completed with
 warnings" | tee -a "$LOG_FILE"

echo "Running read client..." | tee -a "$LOG_FILE"
log_cmd "python3 readclient.py" || echo "Read client execution completed with
 warnings" | tee -a "$LOG_FILE"

# Step 11: Clean up resources
echo "Resources created:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]}"; do
  echo "- $resource" | tee -a "$LOG_FILE"
done

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"
cleanup

echo "Script completed at $(date)" | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateHttpNamespace
  - CreateService
  - DeleteNamespace
  - DeleteService
  - DeregisterInstance
  - GetOperation
  - ListNamespaces
  - ListServices
  - RegisterInstance

Exemplos de código para AWS Cloud9 usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Cloud9 com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS Cloud9 Guia do usuário — Mais informações sobre AWS Cloud9.
- AWS Cloud9 Referência da API — Detalhes sobre todas as AWS Cloud9 ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Cloud9 usar AWS SDKs
  - Ações para AWS Cloud9 usar AWS SDKs
    - Usar CreateEnvironmentEc2 com uma CLI
    - Usar CreateEnvironmentMembership com uma CLI
    - Usar DeleteEnvironment com uma CLI
    - Usar DeleteEnvironmentMembership com uma CLI
    - Usar DescribeEnvironmentMemberships com uma CLI
    - Usar DescribeEnvironmentStatus com uma CLI
    - Usar DescribeEnvironments com uma CLI
    - Usar ListEnvironments com uma CLI
    - Usar UpdateEnvironment com uma CLI
    - Usar UpdateEnvironmentMembership com uma CLI

Exemplos básicos para AWS Cloud9 usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Cloud9 com AWS SDKs.

Exemplos

- Ações para AWS Cloud9 usar AWS SDKs
  - Usar CreateEnvironmentEc2 com uma CLI
  - Usar CreateEnvironmentMembership com uma CLI
  - Usar DeleteEnvironment com uma CLI
  - Usar DeleteEnvironmentMembership com uma CLI
  - Usar DescribeEnvironmentMemberships com uma CLI
  - Usar DescribeEnvironmentStatus com uma CLI
  - Usar DescribeEnvironments com uma CLI
  - Usar ListEnvironments com uma CLI
  - Usar UpdateEnvironment com uma CLI
  - Usar UpdateEnvironmentMembership com uma CLI

Ações para AWS Cloud9 usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Cloud9 ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Cloud9.

Exemplos

- Usar CreateEnvironmentEc2 com uma CLI
- Usar CreateEnvironmentMembership com uma CLI
- Usar DeleteEnvironment com uma CLI
- Usar DeleteEnvironmentMembership com uma CLI
- Usar DescribeEnvironmentMemberships com uma CLI
- Usar DescribeEnvironmentStatus com uma CLI
- Usar DescribeEnvironments com uma CLI

- Usar ListEnvironments com uma CLI
- Usar UpdateEnvironment com uma CLI
- Usar UpdateEnvironmentMembership com uma CLI

Usar **`CreateEnvironmentEc2`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateEnvironmentEc2`.

CLI

AWS CLI

Para criar um ambiente de AWS desenvolvimento Cloud9 EC2

O `create-environment-ec2` exemplo a seguir cria um AWS ambiente de desenvolvimento Cloud9 com as configurações especificadas, inicia uma instância do Amazon Elastic Compute Cloud (Amazon EC2) e, em seguida, se conecta da instância ao ambiente.

```
aws cloud9 create-environment-ec2 \
    --name my-demo-env \
    --description "My demonstration development environment." \
    --instance-type t2.micro --image-id amazonlinux-2023-x86_64 \
    --subnet-id subnet-1fab8aEX \
    --automatic-stop-time-minutes 60 \
    --owner-arn arn:aws:iam::123456789012:user/MyDemoUser
```

Saída:

```
{
    "environmentId": "8a34f51ce1e04a08882f1e811bd706EX"
}
```

Para obter mais informações, consulte Criação de um ambiente do EC2 no Guia do usuário do AWS Cloud9.

- Para obter detalhes da API, consulte CreateEnvironmentEc2em Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria um ambiente de desenvolvimento AWS Cloud9 com as configurações especificadas, inicia uma instância do Amazon Elastic Compute Cloud (Amazon EC2) e, em seguida, se conecta da instância ao ambiente.

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
 "My demonstration development environment." -InstanceType t2.micro -OwnerArn
 arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

Saída:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes da API, consulte CreateEnvironmentEc2em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um ambiente de desenvolvimento AWS Cloud9 com as configurações especificadas, inicia uma instância do Amazon Elastic Compute Cloud (Amazon EC2) e, em seguida, se conecta da instância ao ambiente.

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
 "My demonstration development environment." -InstanceType t2.micro -OwnerArn
 arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

Saída:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes da API, consulte CreateEnvironmentEc2em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateEnvironmentMembership`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateEnvironmentMembership`.

CLI

AWS CLI

Para adicionar um membro do ambiente a um ambiente de desenvolvimento do AWS Cloud9

Este exemplo adiciona o membro do ambiente especificado ao ambiente de desenvolvimento do AWS Cloud9 especificado.

Comando:

```
aws cloud9 create-environment-membership --environment-
id 8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/
AnotherDemoUser --permissions read-write
```

Saída:

```
{
  "membership": {
    "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
    "userId": "AIDAJ3LOROMOUXTBSU6EX",
    "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",
    "permissions": "read-write"
  }
}
```

- Para obter detalhes da API, consulte CreateEnvironmentMembershipem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo adiciona o membro do ambiente especificado ao ambiente de desenvolvimento do AWS Cloud9 especificado.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
write
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Para obter detalhes da API, consulte CreateEnvironmentMembershipem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo adiciona o membro do ambiente especificado ao ambiente de desenvolvimento do AWS Cloud9 especificado.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
write
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Para obter detalhes da API, consulte CreateEnvironmentMembershipem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteEnvironment`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteEnvironment`.

CLI

AWS CLI

Para excluir um ambiente de desenvolvimento do AWS Cloud9

Este exemplo exclui o ambiente de desenvolvimento do AWS Cloud9 especificado. Se uma instância do Amazon EC2 estiver conectada ao ambiente, também encerrará a instância.

Comando:

```
aws cloud9 delete-environment --environment-id 8a34f51ce1e04a08882f1e811bd706EX
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte DeleteEnvironmentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui o ambiente de desenvolvimento AWS Cloud9 especificado. Se uma instância do Amazon EC2 estiver conectada ao ambiente, também encerrará a instância.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes da API, consulte DeleteEnvironmentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui o ambiente de desenvolvimento AWS Cloud9 especificado. Se uma instância do Amazon EC2 estiver conectada ao ambiente, também encerrará a instância.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes da API, consulte DeleteEnvironmentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteEnvironmentMembership`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteEnvironmentMembership`.

CLI

AWS CLI

Para excluir um membro do ambiente de um ambiente de desenvolvimento do AWS Cloud9

Este exemplo exclui o membro do ambiente especificado do ambiente de desenvolvimento do AWS Cloud9 especificado.

Comando:

```
aws cloud9 delete-environment-membership --environment-
id 8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/
AnotherDemoUser
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte DeleteEnvironmentMembershipem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui o membro do ambiente especificado do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes da API, consulte DeleteEnvironmentMembershipem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui o membro do ambiente especificado do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes da API, consulte DeleteEnvironmentMembershipem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeEnvironmentMemberships`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeEnvironmentMemberships`.

CLI

AWS CLI

Para obter informações sobre os membros do ambiente de um ambiente de desenvolvimento AWS Cloud9

Este exemplo obtém informações sobre os membros do ambiente do ambiente de desenvolvimento AWS Cloud9 especificado.

Comando:

```
aws cloud9 describe-environment-memberships --environment-
id 8a34f51ce1e04a08882f1e811bd706EX
```

Saída:

```
{
  "memberships": [
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJ3LOROMOUXTBSU6EX",
      "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",
      "permissions": "read-write"
    },
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
```

```
  ]
}
```

Para obter informações sobre o proprietário de um ambiente de desenvolvimento AWS Cloud9

Este exemplo obtém informações sobre o proprietário do ambiente de desenvolvimento AWS Cloud9 especificado.

Comando:

```
aws cloud9 describe-environment-memberships --environment-
id 8a34f51ce1e04a08882f1e811bd706EX --permissions owner
```

Saída:

```
{
  "memberships": [
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
  ]
}
```

Para obter informações sobre um membro do ambiente para vários ambientes de desenvolvimento do AWS Cloud9

Este exemplo obtém informações sobre o membro do ambiente especificado para vários ambientes de desenvolvimento do AWS Cloud9.

Comando:

```
aws cloud9 describe-environment-memberships --user-
arn arn:aws:iam::123456789012:user/MyDemoUser
```

Saída:

```
{
  "memberships": [
    {
```

```
      "environmentId": "10a75714bd494714929e7f5ec4125aEX",
      "lastAccess": 1516213427.0,
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    },
    {
      "environmentId": "1980b80e5f584920801c09086667f0EX",
      "lastAccess": 1516144884.0,
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
  ]
}
```

- Para obter detalhes da API, consulte DescribeEnvironmentMembershipsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo obtém informações sobre os membros do ambiente do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Exemplo 2: Este exemplo obtém informações sobre o proprietário do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Exemplo 3: Este exemplo obtém informações sobre o membro do ambiente especificado para vários ambientes de desenvolvimento do AWS Cloud9.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/
MyDemoUser
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

- Para obter detalhes da API, consulte DescribeEnvironmentMemberships em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém informações sobre os membros do ambiente do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Exemplo 2: Este exemplo obtém informações sobre o proprietário do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Exemplo 3: Este exemplo obtém informações sobre o membro do ambiente especificado para vários ambientes de desenvolvimento do AWS Cloud9.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/
MyDemoUser
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
```

```
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

- Para obter detalhes da API, consulte DescribeEnvironmentMembershipsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeEnvironmentStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeEnvironmentStatus`.

CLI

AWS CLI

Para obter informações de status de um ambiente de desenvolvimento AWS Cloud9

Este exemplo obtém informações de status para o ambiente de desenvolvimento AWS Cloud9 especificado.

Comando:

```
aws cloud9 describe-environment-status --environment-
id 685f892f431b45c2b28cb69eadcdb0EX
```

Saída:

```
{
  "status": "ready",
  "message": "Environment is ready to use"
}
```

- Para obter detalhes da API, consulte DescribeEnvironmentStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo obtém informações de status para o ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Saída:

```
Message                     Status
-------                     ------
Environment is ready to use ready
```

- Para obter detalhes da API, consulte DescribeEnvironmentStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém informações de status para o ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Saída:

```
Message                     Status
-------                     ------
Environment is ready to use ready
```

- Para obter detalhes da API, consulte DescribeEnvironmentStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeEnvironments`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeEnvironments`.

CLI

AWS CLI

Para obter informações sobre os ambientes de desenvolvimento do AWS Cloud9

Este exemplo obtém informações sobre os ambientes de desenvolvimento AWS Cloud9 especificados.

Comando:

```
aws cloud9 describe-environments --environment-
ids 685f892f431b45c2b28cb69eadcdb0EX 349c86d4579e4e7298d500ff57a6b2EX
```

Saída:

```
{
  "environments": [
    {
      "id": "685f892f431b45c2b28cb69eadcdb0EX",
      "name": "my-demo-ec2-env",
      "description": "Created from CodeStar.",
      "type": "ec2",
      "arn": "arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX",
      "ownerArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "lifecycle": {
        "status": "CREATED"
      }
    },
    {
      "id": "349c86d4579e4e7298d500ff57a6b2EX",
      "name": my-demo-ssh-env",
      "description": "",
      "type": "ssh",
      "arn": "arn:aws:cloud9:us-
east-1:123456789012:environment:349c86d4579e4e7298d500ff57a6b2EX",
      "ownerArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "lifecycle": {
        "status": "CREATED"
      }
    }
  ]
```

```
}
```

- Para obter detalhes da API, consulte DescribeEnvironmentsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo obtém informações sobre os ambientes de desenvolvimento do AWS Cloud9 especificados.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Saída:

```
Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
Description : Created from CodeStar.
Id          : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ec2-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ec2

Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description :
Id          : 1980b80e5f584920801c09086667f0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ssh-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ssh
```

Exemplo 2: Este exemplo obtém informações sobre o status do ciclo de vida do ambiente de desenvolvimento Cloud9 especificado AWS .

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Saída:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- Para obter detalhes da API, consulte DescribeEnvironmentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém informações sobre os ambientes de desenvolvimento do AWS Cloud9 especificados.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Saída:

```
Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
Description : Created from CodeStar.
Id          : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ec2-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ec2

Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description :
Id          : 1980b80e5f584920801c09086667f0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ssh-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ssh
```

Exemplo 2: Este exemplo obtém informações sobre o status do ciclo de vida do ambiente de desenvolvimento Cloud9 especificado AWS .

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Saída:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- Para obter detalhes da API, consulte DescribeEnvironmentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListEnvironments`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListEnvironments`.

CLI

AWS CLI

Para obter uma lista dos identificadores de ambiente de AWS desenvolvimento Cloud9 disponíveis

Este exemplo obtém uma lista dos identificadores de ambiente de AWS desenvolvimento do Cloud9 disponíveis.

Comando:

```
aws cloud9 list-environments
```

Saída:

```
{
  "environmentIds": [
    "685f892f431b45c2b28cb69eadcdb0EX",
    "1980b80e5f584920801c09086667f0EX"
  ]
}
```

- Para obter detalhes da API, consulte ListEnvironmentsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo obtém uma lista de identificadores de ambiente de desenvolvimento do AWS Cloud9 disponíveis.

```
Get-C9EnvironmentList
```

Saída:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- Para obter detalhes da API, consulte ListEnvironmentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém uma lista de identificadores de ambiente de desenvolvimento do AWS Cloud9 disponíveis.

```
Get-C9EnvironmentList
```

Saída:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- Para obter detalhes da API, consulte ListEnvironmentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateEnvironment`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateEnvironment`.

CLI

AWS CLI

Para alterar as configurações de um ambiente de desenvolvimento AWS Cloud9 existente

Este exemplo altera as configurações especificadas do ambiente de desenvolvimento AWS Cloud9 existente especificado.

Comando:

```
aws cloud9 update-environment --environment-id 8a34f51ce1e04a08882f1e811bd706EX
 --name my-changed-demo-env --description "My changed demonstration development
 environment."
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte UpdateEnvironmentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo altera as configurações especificadas do ambiente de desenvolvimento AWS Cloud9 existente especificado.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- Para obter detalhes da API, consulte UpdateEnvironmentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo altera as configurações especificadas do ambiente de desenvolvimento AWS Cloud9 existente especificado.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- Para obter detalhes da API, consulte UpdateEnvironmentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateEnvironmentMembership`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateEnvironmentMembership`.

CLI

AWS CLI

Para alterar as configurações de um membro de ambiente existente para um ambiente de desenvolvimento AWS Cloud9

Este exemplo altera as configurações do membro do ambiente existente especificado para o ambiente de desenvolvimento AWS Cloud9 especificado.

Comando:

```
aws cloud9 update-environment-membership --environment-
id 8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/
AnotherDemoUser --permissions read-only
```

Saída:

```
{
  "membership": {
    "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
    "userId": "AIDAJ3LOROMOUXTBSU6EX",
    "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",
    "permissions": "read-only"
  }
}
```

- Para obter detalhes da API, consulte UpdateEnvironmentMembershipem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo altera as configurações do membro do ambiente existente especificado para o ambiente de desenvolvimento AWS Cloud9 especificado.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Para obter detalhes da API, consulte UpdateEnvironmentMembershipem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo altera as configurações do membro do ambiente existente especificado para o ambiente de desenvolvimento AWS Cloud9 especificado.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Para obter detalhes da API, consulte UpdateEnvironmentMembershipem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para CloudFormation usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS CloudFormation com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- CloudFormation Guia do usuário — Mais informações sobre CloudFormation.
- CloudFormation Referência da API — Detalhes sobre todas as CloudFormation ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para CloudFormation usar AWS SDKs
  - Hello CloudFormation
  - Ações para CloudFormation usar AWS SDKs
    - Usar CancelUpdateStack com uma CLI
    - Usar ContinueUpdateRollback com uma CLI
    - Usar CreateStack com uma CLI
    - Usar DeleteStack com uma CLI
    - Usar DescribeStackEvents com uma CLI
    - Usar DescribeStackResource com uma CLI
    - Usar DescribeStackResources com uma CLI
    - Use DescribeStackscom um AWS SDK ou CLI
    - Usar EstimateTemplateCost com uma CLI
    - Usar GetTemplate com uma CLI
    - Usar ListStackResources com uma CLI
    - Usar ListStacks com uma CLI
    - Usar UpdateStack com uma CLI

    - Usar ValidateTemplate com uma CLI
- Cenários para CloudFormation usar AWS SDKs
  - Crie uma API REST do API Gateway para rastrear COVID-19 dados
  - Criação da sua primeira pilha de infraestrutura
  - Conceitos básicos de clusters gerenciados de kubernetes

Exemplos básicos para CloudFormation usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS CloudFormation com AWS SDKs.

Exemplos

- Hello CloudFormation
- Ações para CloudFormation usar AWS SDKs
  - Usar CancelUpdateStack com uma CLI
  - Usar ContinueUpdateRollback com uma CLI
  - Usar CreateStack com uma CLI
  - Usar DeleteStack com uma CLI
  - Usar DescribeStackEvents com uma CLI
  - Usar DescribeStackResource com uma CLI
  - Usar DescribeStackResources com uma CLI
  - Use DescribeStackscom um AWS SDK ou CLI
  - Usar EstimateTemplateCost com uma CLI
  - Usar GetTemplate com uma CLI
  - Usar ListStackResources com uma CLI
  - Usar ListStacks com uma CLI
  - Usar UpdateStack com uma CLI
  - Usar ValidateTemplate com uma CLI

Hello CloudFormation

O exemplo de código a seguir mostra como começar a usar o CloudFormation.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.Runtime;

namespace CloudFormationActions;

public static class HelloCloudFormation
{
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    static async Task Main(string[] args)
    {
        // Create the CloudFormation client
        _amazonCloudFormation = new AmazonCloudFormationClient();
        Console.WriteLine($"\nIn Region:
 {_amazonCloudFormation.Config.RegionEndpoint}");

        // List the resources for each stack
        await ListResources();
    }

    /// <summary>
    /// Method to list stack resources and other information.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> ListResources()
    {
        try
        {
            Console.WriteLine("Getting CloudFormation stack information...");

            // Get all stacks using the stack paginator.
```

```
            var paginatorForDescribeStacks =
                _amazonCloudFormation.Paginators.DescribeStacks(
                    new DescribeStacksRequest());
            if (paginatorForDescribeStacks.Stacks != null)
            {
                await foreach (Stack stack in paginatorForDescribeStacks.Stacks)
                {
                    // Basic information for each stack
                    Console.WriteLine(
                        "\n--------------------------------------------------");
                    Console.WriteLine($"\nStack: {stack.StackName}");
                    Console.WriteLine($"  Status: {stack.StackStatus.Value}");
                    Console.WriteLine($"  Created: {stack.CreationTime}");

                    // The tags of each stack (etc.)
                    if (stack.Tags != null && stack.Tags.Count > 0)
                    {
                        Console.WriteLine("  Tags:");
                        foreach (Tag tag in stack.Tags)
                            Console.WriteLine($"    {tag.Key}, {tag.Value}");
                    }

                    // The resources of each stack
                    DescribeStackResourcesResponse responseDescribeResources =
                        await _amazonCloudFormation.DescribeStackResourcesAsync(
                            new DescribeStackResourcesRequest
                            {
                                StackName = stack.StackName
                            });
                    if (responseDescribeResources.StackResources != null &&
responseDescribeResources.StackResources.Count > 0)
                    {
                        Console.WriteLine("  Resources:");
                        foreach (StackResource resource in
responseDescribeResources
                                     .StackResources)
                            Console.WriteLine(
                                $"    {resource.LogicalResourceId}:
{resource.ResourceStatus}");
                    }
                }
            }
```

```
Console.WriteLine("\n-----------------------------------------------");
            return true;
        }
        catch (AmazonCloudFormationException ex)
        {
            Console.WriteLine("Unable to get stack information:\n" + ex.Message);
            return false;
        }
        catch (AmazonServiceException ex)
        {
            if (ex.Message.Contains("Unable to get IAM security credentials"))
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine("If you are usnig SSO, be sure to install" +
                                  " the AWSSDK.SSO and AWSSDK.SSOOIDC
packages.");
            }
            else
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }

            return false;
        }
        catch (ArgumentNullException ex)
        {
            if (ex.Message.Contains("Options property cannot be empty:
ClientName"))
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine("If you are using SSO, have you logged in?");
            }
            else
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }

            return false;
        }
    }
```

- Para obter detalhes da API, consulte DescribeStackResourcesa Referência AWS SDK para .NET da API.

Ações para CloudFormation usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar CloudFormation ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a CloudFormation API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para CloudFormation usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS CloudFormation.

Exemplos

- Usar CancelUpdateStack com uma CLI
- Usar ContinueUpdateRollback com uma CLI
- Usar CreateStack com uma CLI
- Usar DeleteStack com uma CLI
- Usar DescribeStackEvents com uma CLI
- Usar DescribeStackResource com uma CLI
- Usar DescribeStackResources com uma CLI
- Use DescribeStackscom um AWS SDK ou CLI
- Usar EstimateTemplateCost com uma CLI
- Usar GetTemplate com uma CLI
- Usar ListStackResources com uma CLI
- Usar ListStacks com uma CLI
- Usar UpdateStack com uma CLI
- Usar ValidateTemplate com uma CLI

Usar **`CancelUpdateStack`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CancelUpdateStack`.

CLI

AWS CLI

Para cancelar uma atualização de pilha que está em andamento

O comando `cancel-update-stack` a seguir cancela uma atualização de pilha na pilha `myteststack`:

```
aws cloudformation cancel-update-stack --stack-name mytestsack
```

- Para obter detalhes da API, consulte CancelUpdateStackem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cancela uma atualização na pilha especificada.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Para obter detalhes da API, consulte CancelUpdateStackem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cancela uma atualização na pilha especificada.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Para obter detalhes da API, consulte CancelUpdateStackem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ContinueUpdateRollback`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ContinueUpdateRollback`.

CLI

AWS CLI

Para fazer nova tentativa de reversão de atualização

O exemplo `continue-update-rollback` a seguir retoma uma operação de reversão de uma atualização de pilha anterior em que houve falha.

```
aws cloudformation continue-update-rollback \
    --stack-name my-stack
```

Este comando não produz saída.

- Para obter detalhes da API, consulte ContinueUpdateRollbackem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: continua a reversão da pilha nomeada, que deve estar no estado "UPDATE_ROLLBACK_FAILED". Se a reversão contínua for bem-sucedida, a pilha entrará no estado "UPDATE_ROLLBACK_COMPLETE".

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Para obter detalhes da API, consulte ContinueUpdateRollbackem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: continua a reversão da pilha nomeada, que deve estar no estado "UPDATE_ROLLBACK_FAILED". Se a reversão contínua for bem-sucedida, a pilha entrará no estado "UPDATE_ROLLBACK_COMPLETE".

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Para obter detalhes da API, consulte ContinueUpdateRollbackem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateStack`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateStack`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criação da sua primeira pilha de infraestrutura
- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Para criar uma AWS CloudFormation pilha

O exemplo de `create-stacks` a seguir cria uma pilha com o nome de `myteststack` usando o arquivo de modelo `sampletemplate.json`:

```
aws cloudformation create-stack --stack-name myteststack --template-body file://
sampletemplate.json --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
 ParameterKey=SubnetIDs,ParameterValue=SubnetID1\\,SubnetID2
```

Saída:

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

Para obter mais informações, consulte Pilhas no Guia do AWS CloudFormation usuário.

- Para obter detalhes da API, consulte CreateStackem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria uma nova pilha com o nome especificado. O modelo é analisado a partir do conteúdo fornecido com parâmetros de personalização ("PK1" e "PK2" representam os nomes dos parâmetros declarados no conteúdo do modelo, "PV1" e "PV2" representam os valores

desses parâmetros. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se houver falha na criação da pilha, ela não será revertida.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Exemplo 2: cria uma nova pilha com o nome especificado. O modelo é analisado a partir do conteúdo fornecido com parâmetros de personalização ("PK1" e "PK2" representam os nomes dos parâmetros declarados no conteúdo do modelo, "PV1" e "PV2" representam os valores desses parâmetros. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se houver falha na criação da pilha, ela será revertida.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

Exemplo 3: cria uma nova pilha com o nome especificado. O modelo é obtido da URL do Amazon S3 com parâmetros de personalização ('PK1' representa o nome de um parâmetro declarado no conteúdo do modelo, 'PV1' representa o valor do parâmetro. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se a criação da pilha falhar, ela será revertida (o mesmo que especificar - DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
```

```
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 4: cria uma nova pilha com o nome especificado. O modelo é obtido da URL do Amazon S3 com parâmetros de personalização ('PK1' representa o nome de um parâmetro declarado no conteúdo do modelo, 'PV1' representa o valor do parâmetro. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se a criação da pilha falhar, ela será revertida (o mesmo que especificar - DisableRollback $false). Os AENs de notificação especificados receberão os eventos relacionados à pilha publicados.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- Para obter detalhes da API, consulte CreateStackem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria uma nova pilha com o nome especificado. O modelo é analisado a partir do conteúdo fornecido com parâmetros de personalização ("PK1" e "PK2" representam os nomes dos parâmetros declarados no conteúdo do modelo, "PV1" e "PV2" representam os valores desses parâmetros. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se houver falha na criação da pilha, ela não será revertida.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Exemplo 2: cria uma nova pilha com o nome especificado. O modelo é analisado a partir do conteúdo fornecido com parâmetros de personalização ("PK1" e "PK2" representam os nomes dos parâmetros declarados no conteúdo do modelo, "PV1" e "PV2" representam os valores desses parâmetros. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se houver falha na criação da pilha, ela será revertida.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

Exemplo 3: cria uma nova pilha com o nome especificado. O modelo é obtido da URL do Amazon S3 com parâmetros de personalização ('PK1' representa o nome de um parâmetro declarado no conteúdo do modelo, 'PV1' representa o valor do parâmetro. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se a criação da pilha falhar, ela será revertida (o mesmo que especificar - DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 4: cria uma nova pilha com o nome especificado. O modelo é obtido da URL do Amazon S3 com parâmetros de personalização ('PK1' representa o nome de um parâmetro declarado no conteúdo do modelo, 'PV1' representa o valor do parâmetro. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se a criação da pilha falhar, ela será revertida (o mesmo que especificar - DisableRollback $false). Os AENs de notificação especificados receberão os eventos relacionados à pilha publicados.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- Para obter detalhes da API, consulte CreateStackem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteStack`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteStack`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criação da sua primeira pilha de infraestrutura
- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Para excluir uma pilha

O exemplo de `delete-stack` a seguir exclui a pilha especificada.

```
aws cloudformation delete-stack \
    --stack-name my-stack
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteStackem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui a pilha especificada.

```
Remove-CFNStack -StackName "myStack"
```

- Para obter detalhes da API, consulte DeleteStackem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui a pilha especificada.

```
Remove-CFNStack -StackName "myStack"
```

- Para obter detalhes da API, consulte DeleteStackem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeStackEvents`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeStackEvents`.

CLI

AWS CLI

Para descrever os eventos da pilha

O exemplo `describe-stack-events` a seguir exibe os dois eventos mais recentes da pilha especificada.

```
aws cloudformation describe-stack-events \
    --stack-name my-stack \
    --max-items 2

{
    "StackEvents": [
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "EventId": "4e1516d0-e4d6-xmpl-b94f-0a51958a168c",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2019-10-02T05:34:29.556Z",
            "ResourceStatus": "UPDATE_COMPLETE"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
```

```
            "EventId": "4dd3c810-e4d6-xmpl-bade-0aaf8b31ab7a",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2019-10-02T05:34:29.127Z",
            "ResourceStatus": "UPDATE_COMPLETE_CLEANUP_IN_PROGRESS"
        }
    ],

 "NextToken": "eyJOZXh0VG9XMPLiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ==="
}
```

- Para obter detalhes da API, consulte DescribeStackEventsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna todos os eventos relacionados à pilha especificada.

```
Get-CFNStackEvent -StackName "myStack"
```

- Para obter detalhes da API, consulte DescribeStackEventsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna todos os eventos relacionados à pilha especificada.

```
Get-CFNStackEvent -StackName "myStack"
```

- Para obter detalhes da API, consulte DescribeStackEventsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeStackResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeStackResource`.

CLI

AWS CLI

Para obter informações detalhadas sobre o recurso de uma pilha

O exemplo de `describe-stack-resource` a seguir exibe os detalhes do recurso denominado `MyFunction` da pilha especificada.

```
aws cloudformation describe-stack-resource \
    --stack-name MyStack \
    --logical-resource-id MyFunction
```

Saída:

```
{
    "StackResourceDetail": {
        "StackName": "MyStack",
        "StackId": "arn:aws:cloudformation:us-east-2:123456789012:stack/MyStack/
d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
        "LogicalResourceId": "MyFunction",
        "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
        "ResourceType": "AWS::Lambda::Function",
        "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
        "ResourceStatus": "UPDATE_COMPLETE",
        "Metadata": "{}",
        "DriftInformation": {
            "StackResourceDriftStatus": "IN_SYNC"
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeStackResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna a descrição de um recurso identificado no modelo associado à pilha especificada pelo ID lógico "MyDBInstance".

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Para obter detalhes da API, consulte DescribeStackResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna a descrição de um recurso identificado no modelo associado à pilha especificada pelo ID lógico "MyDBInstance".

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Para obter detalhes da API, consulte DescribeStackResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeStackResources`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeStackResources`.

CLI

AWS CLI

Para obter informações detalhadas sobre o recurso de uma pilha

O exemplo de `describe-stack-resources` a seguir exibe os detalhes dos recursos na pilha especificada.

```
aws cloudformation describe-stack-resources \
    --stack-name my-stack
```

Saída:

```
{
    "StackResources": [
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "bucket",
```

```
            "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
            "ResourceType": "AWS::S3::Bucket",
            "Timestamp": "2019-10-02T04:34:11.345Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "function",
            "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
            "ResourceType": "AWS::Lambda::Function",
            "Timestamp": "2019-10-02T05:34:27.989Z",
            "ResourceStatus": "UPDATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "functionRole",
            "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
            "ResourceType": "AWS::IAM::Role",
            "Timestamp": "2019-10-02T04:34:06.350Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeStackResourcesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna as descrições dos AWS recursos de até 100 recursos associados à pilha especificada. Para obter detalhes de todos os recursos associados a uma pilha, use o Get-CFNStackResourceSummary, que também oferece suporte à paginação manual dos resultados.

```
Get-CFNStackResourceList -StackName "myStack"
```

Exemplo 2: retorna a descrição da instância do Amazon EC2 identificada no modelo associado à pilha especificada pelo ID lógico "Ec2Instance".

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Exemplo 3: retorna a descrição de até 100 recursos associados à pilha contendo uma instância do Amazon EC2 identificada pelo ID de instância "i-123456". Para obter detalhes de todos os recursos associados a uma pilha, use o Get-CFNStackResourceSummary, que também oferece suporte à paginação manual dos resultados.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Exemplo 4: retorna a descrição da instância do Amazon EC2 identificada pelo ID lógico "Ec2Instance" no modelo de uma pilha. A pilha é identificada usando o ID de recurso físico de um recurso que ela contém; nesse caso, também uma instância do Amazon EC2 com o ID de instância "i-123456". Um outro recurso físico também poderia ser usado para identificar a pilha, dependendo do conteúdo do modelo, por exemplo, um bucket do Amazon S3.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- Para obter detalhes da API, consulte DescribeStackResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna as descrições dos AWS recursos de até 100 recursos associados à pilha especificada. Para obter detalhes de todos os recursos associados a uma pilha, use

o Get-CFNStackResourceSummary, que também oferece suporte à paginação manual dos resultados.

```
Get-CFNStackResourceList -StackName "myStack"
```

Exemplo 2: retorna a descrição da instância do Amazon EC2 identificada no modelo associado à pilha especificada pelo ID lógico "Ec2Instance".

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Exemplo 3: retorna a descrição de até 100 recursos associados à pilha contendo uma instância do Amazon EC2 identificada pelo ID de instância "i-123456". Para obter detalhes de todos os recursos associados a uma pilha, use o Get-CFNStackResourceSummary, que também oferece suporte à paginação manual dos resultados.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Exemplo 4: retorna a descrição da instância do Amazon EC2 identificada pelo ID lógico "Ec2Instance" no modelo de uma pilha. A pilha é identificada usando o ID de recurso físico de um recurso que ela contém; nesse caso, também uma instância do Amazon EC2 com o ID de instância "i-123456". Um outro recurso físico também poderia ser usado para identificar a pilha, dependendo do conteúdo do modelo, por exemplo, um bucket do Amazon S3.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- Para obter detalhes da API, consulte DescribeStackResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeStacks`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeStacks`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criação da sua primeira pilha de infraestrutura

- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Para descrever AWS CloudFormation pilhas

O comando `describe-stacks` a seguir mostra informações resumidas da pilha `myteststack`:

```
aws cloudformation describe-stacks --stack-name myteststack
```

Saída:

```
{
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
mytestsstack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
 template showing how to create a publicly accessible S3 bucket. **WARNING** This
 template creates an S3 bucket. You will be billed for the AWS resources used if
 you create a stack from this template.",
            "Tags": [],
            "Outputs": [
                {
                    "Description": "Name of S3 bucket to hold website content",
                    "OutputKey": "BucketName",
                    "OutputValue": "myteststack-s3bucket-jssofi1zie2w"
                }
            ],
            "StackStatusReason": null,
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "CREATE_COMPLETE",
            "DisableRollback": false
        }
    ]
}
```

Para obter mais informações, consulte Pilhas no Guia do AWS CloudFormation usuário.

- Para obter detalhes da API, consulte DescribeStacksem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
```

```
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

- Para obter detalhes da API, consulte DescribeStacksa Referência AWS SDK para Go da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna um conjunto de instâncias de pilha descrevendo todas as pilhas do usuário.

```
Get-CFNStack
```

Exemplo 2: retorna uma instância de pilha descrevendo a pilha especificada

```
Get-CFNStack -StackName "myStack"
```

- Para obter detalhes da API, consulte DescribeStacksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna um conjunto de instâncias de pilha descrevendo todas as pilhas do usuário.

```
Get-CFNStack
```

Exemplo 2: retorna uma instância de pilha descrevendo a pilha especificada

```
Get-CFNStack -StackName "myStack"
```

- Para obter detalhes da API, consulte DescribeStacksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`EstimateTemplateCost`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EstimateTemplateCost`.

CLI

AWS CLI

Para estimar o custo do modelo

O exemplo de `estimate-template-cost` a seguir gera uma estimativa de custo para um modelo denominado `template.yaml` na pasta atual.

```
aws cloudformation estimate-template-cost \
    --template-body file://template.yaml
```

Saída:

```
{
    "Url": "http://calculator.s3.amazonaws.com/calc5.html?
key=cloudformation/7870825a-xmpl-4def-92e7-c4f8dd360cca"
}
```

- Para obter detalhes da API, consulte EstimateTemplateCostem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna uma URL AWS simples da calculadora mensal com uma sequência de caracteres de consulta que descreve os recursos necessários para executar o modelo. O modelo é obtido da URL do Amazon S3 especificada e do único parâmetro de personalização aplicado. O parâmetro também pode ser especificado usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-
bucket/templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Exemplo 2: retorna uma URL AWS simples da calculadora mensal com uma sequência de caracteres de consulta que descreve os recursos necessários para executar o modelo. O modelo é analisado a partir do conteúdo fornecido e os parâmetros de personalização aplicados (este exemplo pressupõe que o conteúdo do modelo teria declarado dois parâmetros, '' e 'KeyName')InstanceType. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                      @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Exemplo 3: usa New-Object para criar o conjunto de parâmetros do modelo e retorna um URL AWS simples da calculadora mensal com uma sequência de caracteres de consulta que descreve os recursos necessários para executar o modelo. O modelo é analisado a partir do conteúdo fornecido, com parâmetros de personalização (este exemplo pressupõe que o conteúdo do modelo teria declarado dois parâmetros, '' e KeyName ')InstanceType.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter
 @( $p1, $p2 )
```

- Para obter detalhes da API, consulte EstimateTemplateCostem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna uma URL AWS simples da calculadora mensal com uma sequência de caracteres de consulta que descreve os recursos necessários para executar o modelo. O modelo é obtido da URL do Amazon S3 especificada e do único parâmetro de personalização aplicado. O parâmetro também pode ser especificado usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-
bucket/templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Exemplo 2: retorna uma URL AWS simples da calculadora mensal com uma sequência de caracteres de consulta que descreve os recursos necessários para executar o modelo. O modelo é analisado a partir do conteúdo fornecido e os parâmetros de personalização aplicados (este exemplo pressupõe que o conteúdo do modelo teria declarado dois parâmetros, '' e 'KeyName')InstanceType. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                      @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Exemplo 3: usa New-Object para criar o conjunto de parâmetros do modelo e retorna um URL AWS simples da calculadora mensal com uma sequência de caracteres de consulta que descreve os recursos necessários para executar o modelo. O modelo é analisado a partir do conteúdo fornecido, com parâmetros de personalização (este exemplo pressupõe que o conteúdo do modelo teria declarado dois parâmetros, '' e KeyName ')InstanceType.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
```

```
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter
 @( $p1, $p2 )
```

- Para obter detalhes da API, consulte EstimateTemplateCostem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetTemplate`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetTemplate`.

CLI

AWS CLI

Para visualizar o corpo do modelo de uma AWS CloudFormation pilha

O comando `get-template` a seguir mostra o modelo da pilha `myteststack`:

```
aws cloudformation get-template --stack-name myteststack
```

Saída:

```
{
    "TemplateBody": {
        "AWSTemplateFormatVersion": "2010-09-09",
        "Outputs": {
            "BucketName": {
                "Description": "Name of S3 bucket to hold website content",
                "Value": {
                    "Ref": "S3Bucket"
                }
            }
        },
        "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
 template showing how to create a publicly accessible S3 bucket. **WARNING** This
 template creates an S3 bucket. You will be billed for the AWS resources used if
 you create a stack from this template.",
        "Resources": {
            "S3Bucket": {
```

```
                "Type": "AWS::S3::Bucket",
                "Properties": {
                    "AccessControl": "PublicRead"
                }
            }
        }
    }
}
```

- Para obter detalhes da API, consulte GetTemplateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna o modelo associado à pilha especificada.

```
Get-CFNTemplate -StackName "myStack"
```

- Para obter detalhes da API, consulte GetTemplateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna o modelo associado à pilha especificada.

```
Get-CFNTemplate -StackName "myStack"
```

- Para obter detalhes da API, consulte GetTemplateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListStackResources`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListStackResources`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação da sua primeira pilha de infraestrutura

CLI

AWS CLI

Para listar os recursos de uma pilha

O comando a seguir exibe a lista dos recursos da pilha especificada.

```
aws cloudformation list-stack-resources \
    --stack-name my-stack
```

Saída:

```
{
    "StackResourceSummaries": [
        {
            "LogicalResourceId": "bucket",
            "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
            "ResourceType": "AWS::S3::Bucket",
            "LastUpdatedTimestamp": "2019-10-02T04:34:11.345Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "LogicalResourceId": "function",
            "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
            "ResourceType": "AWS::Lambda::Function",
            "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
            "ResourceStatus": "UPDATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "LogicalResourceId": "functionRole",
            "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
            "ResourceType": "AWS::IAM::Role",
            "LastUpdatedTimestamp": "2019-10-02T04:34:06.350Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
```

```
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte ListStackResourcesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna as descrições de todos os recursos associados à pilha especificada.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- Para obter detalhes da API, consulte ListStackResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna as descrições de todos os recursos associados à pilha especificada.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- Para obter detalhes da API, consulte ListStackResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListStacks`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListStacks`.

CLI

AWS CLI

Para listar AWS CloudFormation pilhas

O comando `list-stacks` a seguir mostra um resumo de todas as pilhas que têm um status de `CREATE_COMPLETE`:

```
aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE
```

Saída:

```
[
    {
        "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
mytestestack/466df9e0-0dff-08e3-8e2f-5088487c4896",
        "TemplateDescription": "AWS CloudFormation Sample Template S3_Bucket:
 Sample template showing how to create a publicly accessible S3 bucket.
 **WARNING** This template creates an S3 bucket. You will be billed for the AWS
 resources used if you create a stack from this template.",
        "StackStatusReason": null,
        "CreationTime": "2013-08-26T03:27:10.190Z",
        "StackName": "mytestestack",
        "StackStatus": "CREATE_COMPLETE"
    }
]
```

- Para obter detalhes da API, consulte ListStacksem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna informações resumidas de todas as pilhas.

```
Get-CFNStackSummary
```

Exemplo 2: retorna informações resumidas de todas as pilhas que estão sendo criadas no momento.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Exemplo 3: retorna informações resumidas de todas as pilhas que estão sendo criadas ou atualizadas no momento.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS")
```

- Para obter detalhes da API, consulte ListStacksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna informações resumidas de todas as pilhas.

```
Get-CFNStackSummary
```

Exemplo 2: retorna informações resumidas de todas as pilhas que estão sendo criadas no momento.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Exemplo 3: retorna informações resumidas de todas as pilhas que estão sendo criadas ou atualizadas no momento.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS")
```

- Para obter detalhes da API, consulte ListStacksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateStack`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateStack`.

CLI

AWS CLI

Para atualizar as AWS CloudFormation pilhas

O comando `update-stack` a seguir atualiza o modelo e os parâmetros de entrada para a pilha `mystack`:

```
aws cloudformation update-stack --stack-name mystack --
template-url https://s3.amazonaws.com/sample/updated.template --
parameters ParameterKey=KeyPairName,ParameterValue=SampleKeyPair
 ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,SampleSubnetID2
```

O comando `update-stack` a seguir atualiza apenas o valor do parâmetro `SubnetIDs` da pilha `mystack`. Se você não especificar um valor para o parâmetro, o valor padrão especificado no modelo será usado:

```
aws cloudformation update-stack --stack-name mystack --
template-url https://s3.amazonaws.com/sample/updated.template
 --parameters ParameterKey=KeyPairName,UsePreviousValue=true
 ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,UpdatedSampleSubnetID2
```

O comando `update-stack` a seguir adiciona à pilha `mystack` dois tópicos de notificação de pilha:

```
aws cloudformation update-stack --stack-name mystack --use-previous-template --
notification-arns "arn:aws:sns:use-east-1:123456789012:mytopic1" "arn:aws:sns:us-
east-1:123456789012:mytopic2"
```

Para obter mais informações, consulte atualizações de AWS CloudFormation pilha no Guia do AWS CloudFormation usuário.

- Para obter detalhes da API, consulte UpdateStackem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "PK1" representa o nome de um parâmetro declarado no modelo e "PV1" representa seu valor. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 2: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemplo 3: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "PK1" representa o nome de um parâmetro declarado no modelo e "PV2" representa seu valor. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 4: atualiza a pilha "myStack" com o modelo obtido no Amazon S3 e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemplo 5: atualiza a pilha "myStack" que, este exemplo pressupõe conter recursos do IAM, com o modelo obtido do Amazon S3 e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. As pilhas contendo recursos do IAM exigem que você especifique o parâmetro -Capabilities “CAPABILITY_IAM”, caso contrário, a atualização falhará com um erro ". InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
```

```
                -Capabilities "CAPABILITY_IAM"
```

- Para obter detalhes da API, consulte UpdateStackem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "PK1" representa o nome de um parâmetro declarado no modelo e "PV1" representa seu valor. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 2: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemplo 3: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "PK1" representa o nome de um parâmetro declarado no modelo e "PV2" representa seu valor. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 4: atualiza a pilha "myStack" com o modelo obtido no Amazon S3 e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemplo 5: atualiza a pilha "myStack" que, este exemplo pressupõe conter recursos do IAM, com o modelo obtido do Amazon S3 e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. As pilhas contendo recursos do IAM exigem que você especifique o parâmetro -Capabilities “CAPABILITY_IAM”, caso contrário, a atualização falhará com um erro ". InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- Para obter detalhes da API, consulte UpdateStackem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ValidateTemplate`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ValidateTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação da sua primeira pilha de infraestrutura

CLI

AWS CLI

Para validar um modelo AWS CloudFormation

O comando `validate-template` a seguir valida o modelo `sampletemplate.json`:

```
aws cloudformation validate-template --template-body file://sampletemplate.json
```

Saída:

```
{
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template
 showing how to create a publicly accessible S3 bucket. **WARNING** This template
 creates an S3 bucket. You will be billed for the AWS resources used if you
 create a stack from this template.",
    "Parameters": [],
    "Capabilities": []
}
```

Para obter mais informações, consulte Trabalhando com AWS CloudFormation modelos no Guia AWS CloudFormation do usuário.

- Para obter detalhes da API, consulte ValidateTemplateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: valida o conteúdo do modelo especificado. A saída detalha os recursos, a descrição e os parâmetros do modelo.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Exemplo 2: valida o modelo especificado acessado por meio de uma URL do Amazon S3. A saída detalha os recursos, a descrição e os parâmetros do modelo.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- Para obter detalhes da API, consulte ValidateTemplateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: valida o conteúdo do modelo especificado. A saída detalha os recursos, a descrição e os parâmetros do modelo.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Exemplo 2: valida o modelo especificado acessado por meio de uma URL do Amazon S3. A saída detalha os recursos, a descrição e os parâmetros do modelo.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- Para obter detalhes da API, consulte ValidateTemplateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para CloudFormation usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns CloudFormation com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas CloudFormation ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie uma API REST do API Gateway para rastrear COVID-19 dados
- Criação da sua primeira pilha de infraestrutura
- Conceitos básicos de clusters gerenciados de kubernetes

Crie uma API REST do API Gateway para rastrear COVID-19 dados

O exemplo de código a seguir mostra como criar uma API REST que simula um sistema para rastrear casos diários nos Estados Unidos da América, usando dados fictícios. COVID-19

Python

SDK para Python (Boto3)

Mostra como usar o AWS Chalice com o AWS SDK para Python (Boto3) para criar uma API REST sem servidor que usa o Amazon API Gateway e o Amazon DynamoDB AWS Lambda. A API REST simula um sistema que rastreia casos diários nos Estados Unidos da América, usando dados fictícios. COVID-19 Aprenda como:

- Use o AWS Chalice para definir rotas nas funções do Lambda que são chamadas para lidar com solicitações REST que chegam por meio do API Gateway.
- Usar as funções do Lambda para recuperar e armazenar dados em uma tabela do DynamoDB para atender a solicitações REST.
- Defina a estrutura da tabela e os recursos da função de segurança em um AWS CloudFormation modelo.
- Use AWS Chalice e CloudFormation para empacotar e implantar todos os recursos necessários.
- Use CloudFormation para limpar todos os recursos criados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Criação da sua primeira pilha de infraestrutura

O exemplo de código a seguir mostra como:

- Crie um CloudFormation modelo
- Testar o servidor web.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudFormation Getting Started Script
# This script creates a CloudFormation stack with a web server and security
 group,
# monitors the stack creation, and provides cleanup options.

set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="cloudformation-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS CloudFormation Getting Started Tutorial"
echo "====================================================="
echo "This script will create a CloudFormation stack with:"
echo "- An EC2 instance running a simple web server"
echo "- A security group allowing HTTP access from your IP"
echo ""
echo "Starting at: $(date)"
echo ""

# Function to validate IP address format
validate_ip() {
    local ip=$1
    if ! [[ "$ip" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        return 1
    fi
```

```
    local IFS=.
    local -a octets=($ip)
    for octet in "${octets[@]}"; do
        if ((octet > 255)); then
            return 1
        fi
    done
    return 0
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="

    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME" --region
 "${AWS_REGION:-us-east-1}" || true

        echo "Waiting for stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
 --region "${AWS_REGION:-us-east-1}" 2>/dev/null || true

        echo "Stack deletion complete."
    fi

    if [ -f "${TEMPLATE_FILE:-}" ]; then
        echo "Removing local template file: $TEMPLATE_FILE"
        shred -vfz -n 3 "$TEMPLATE_FILE" 2>/dev/null || rm -f "$TEMPLATE_FILE"
    fi

    echo "Cleanup completed at: $(date)"
}

# Function to handle errors
handle_error() {
    echo ""
    echo "====================================================="
    echo "ERROR: $1"
    echo "====================================================="
    echo "Resources created before error:"
```

```
    if [ -n "${STACK_NAME:-}" ]; then
        echo "- CloudFormation stack: $STACK_NAME"
    fi
    echo ""

    echo "Cleaning up resources automatically..."
    cleanup

    exit 1
}

# Set up trap for script interruption
trap 'handle_error "Script interrupted"' INT TERM
trap 'cleanup' EXIT

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS_REGION format: $AWS_REGION"
fi

# Generate a unique stack name with timestamp
TIMESTAMP=$(date +%s)
STACK_NAME="MyTestStack-${TIMESTAMP}"
TEMPLATE_FILE="webserver-template-${TIMESTAMP}.yaml"

# Verify AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Verify curl is installed
if ! command -v curl &> /dev/null; then
    handle_error "curl is not installed or not in PATH"
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Step 1: Create the CloudFormation template file
echo "Creating CloudFormation template file: $TEMPLATE_FILE"
cat > "$TEMPLATE_FILE" << 'EOF'
```

```
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation Template for WebServer with Security Group and EC2
 Instance

Metadata:
  AWS::CloudFormation::Init:
    config:
      packages:
        yum:
          httpd: []
      services:
        sysvinit:
          httpd:
            enabled: true
            ensureRunning: true

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.

  MyIP:
    Description: Your IP address in CIDR format (e.g 203.0.113.1/32).
    Type: String
    MinLength: '9'
    MaxLength: '18'
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
```

```
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ec2.amazonaws.com
          Action: 'sts:AssumeRole'
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'

  WebServerInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
        - !Ref WebServerRole

  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP access via specified IP address
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
          Description: Allow HTTP outbound
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
          Description: Allow HTTPS outbound for package updates
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref MyIP
          Description: HTTP access from specified IP

  WebServer:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      IamInstanceProfile: !Ref WebServerInstanceProfile
      SecurityGroupIds:
```

```
        - !Ref WebServerSecurityGroup
      Monitoring: true
      MetadataOptions:
        HttpEndpoint: enabled
        HttpTokens: required
        HttpPutResponseHopLimit: 1
      UserData: !Base64 |
        #!/bin/bash
        set -euo pipefail
        exec > >(tee /var/log/user-data.log)
        exec 2>&1
        yum update -y
        yum install -y httpd
        systemctl start httpd
        systemctl enable httpd
        echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html
        chmod 644 /var/www/html/index.html

Outputs:
  WebsiteURL:
    Value: !Join
      - ''
      - - http://
        - !GetAtt WebServer.PublicDnsName
    Description: Website URL
  InstanceId:
    Value: !Ref WebServer
    Description: EC2 Instance ID
  SecurityGroupId:
    Value: !Ref WebServerSecurityGroup
    Description: Security Group ID
EOF

chmod 600 "$TEMPLATE_FILE"

if [ ! -f "$TEMPLATE_FILE" ]; then
    handle_error "Failed to create template file"
fi

# Step 2: Validate the template
echo ""
echo "Validating CloudFormation template..."
if ! VALIDATION_RESULT=$(aws cloudformation validate-template \
```

```
    --template-body "file://$TEMPLATE_FILE" \
    --region "$AWS_REGION" 2>&1); then
    handle_error "Template validation failed: $VALIDATION_RESULT"
fi
echo "Template validation successful."

# Step 3: Get the user's public IP address
echo ""
echo "Retrieving your public IP address..."

MY_IP=""
for endpoint in "https://checkip.amazonaws.com" "https://api.ipify.org" "https://
icanhazip.com"; do
    if MY_IP=$(curl -s --max-time 5 "$endpoint" 2>/dev/null); then
        MY_IP="${MY_IP//[[:space:]]/}"
        if validate_ip "$MY_IP"; then
            break
        fi
        MY_IP=""
    fi
done

if [ -z "$MY_IP" ]; then
    handle_error "Failed to retrieve public IP address from multiple sources"
fi

MY_IP_CIDR="${MY_IP}/32"
echo "Your public IP address: $MY_IP_CIDR"

# Step 4: Create the CloudFormation stack
echo ""
echo "Creating CloudFormation stack: $STACK_NAME"
echo "This will create an EC2 instance and security group."
if ! CREATE_RESULT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-body "file://$TEMPLATE_FILE" \
  --parameters \
    ParameterKey=InstanceType,ParameterValue=t2.micro \
    ParameterKey=MyIP,ParameterValue="$MY_IP_CIDR" \
  --capabilities CAPABILITY_IAM \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudformation-gs \
  --region "$AWS_REGION" \
  --output text 2>&1); then
    handle_error "Stack creation failed: $CREATE_RESULT"
```

```
fi

STACK_ID="${CREATE_RESULT//[[:space:]]/}"
echo "Stack creation initiated. Stack ID: $STACK_ID"

# Step 5: Monitor stack creation
echo ""
echo "Monitoring stack creation..."
echo "This may take a few minutes."

if ! aws cloudformation wait stack-create-complete \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" 2>/dev/null; then
    STACK_STATUS=$(aws cloudformation describe-stacks \
        --stack-name "$STACK_NAME" \
        --query "Stacks[0].StackStatus" \
        --region "$AWS_REGION" \
        --output text 2>/dev/null || echo "UNKNOWN")
    if [[ "$STACK_STATUS" =~ ROLLBACK|FAILED ]]; then
        handle_error "Stack creation failed. Status: $STACK_STATUS"
    fi
fi

echo "Stack creation completed successfully."

# Step 6: List stack resources
echo ""
echo "Resources created by the stack:"
aws cloudformation list-stack-resources \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "StackResourceSummaries[*].{LogicalID:LogicalResourceId,
 Type:ResourceType, Status:ResourceStatus}" \
    --output table

# Step 7: Get stack outputs
echo ""
echo "Stack outputs:"
if ! OUTPUTS=$(aws cloudformation describe-stacks \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "Stacks[0].Outputs" \
    --output json 2>&1); then
    handle_error "Failed to retrieve stack outputs"
```

```
fi

echo "$OUTPUTS"

# Extract the WebsiteURL
WEBSITE_URL=$(aws cloudformation describe-stacks \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "Stacks[0].Outputs[?OutputKey=='WebsiteURL'].OutputValue" \
    --output text 2>/dev/null || echo "")

if [ -z "$WEBSITE_URL" ]; then
    handle_error "Failed to extract WebsiteURL from stack outputs"
fi

echo ""
echo "WebsiteURL: $WEBSITE_URL"
echo ""
echo "You can access the web server by opening the above URL in your browser."
echo "You should see a simple 'Hello World!' message."

# Step 8: Test the connection via CLI with timeout
echo ""
echo "Testing connection to the web server..."
MAX_RETRIES=5
RETRY_COUNT=0

while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
    if HTTP_RESPONSE=$(curl -s -o /dev/null -w "%{http_code}" --max-time 10
 "$WEBSITE_URL" 2>/dev/null); then
        if [ "$HTTP_RESPONSE" == "200" ]; then
            echo "Connection successful! HTTP status code: $HTTP_RESPONSE"
            break
        elif [ "$HTTP_RESPONSE" == "000" ]; then
            RETRY_COUNT=$((RETRY_COUNT + 1))
            if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
                echo "Connection test attempt $RETRY_COUNT/$MAX_RETRIES -
 retrying in 10 seconds..."
                sleep 10
            else
                echo "Warning: Connection test failed after $MAX_RETRIES
 attempts"
            fi
        else
```

```
            echo "Warning: Connection test returned HTTP status code:
 $HTTP_RESPONSE"
            break
        fi
    else
        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Connection test attempt $RETRY_COUNT/$MAX_RETRIES - retrying in
 10 seconds..."
            sleep 10
        else
            echo "Warning: Connection test failed after $MAX_RETRIES attempts"
        fi
    fi
done

# Step 9: Auto-confirm cleanup
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Resources created:"
echo "- CloudFormation stack: $STACK_NAME"
echo "  - EC2 instance"
echo "  - Security group"
echo "  - IAM role and instance profile"
echo ""
echo "Proceeding with cleanup of all created resources..."

cleanup

echo ""
echo "==================================================="
echo "Tutorial completed at: $(date)"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateStack
  - DeleteStack
  - DescribeStacks
  - ListStackResources

- ValidateTemplate
- Wait

Conceitos básicos de clusters gerenciados de kubernetes

O exemplo de código a seguir mostra como:

- Criar uma VPC para um cluster do EKS.
- Criar perfis do IAM para um cluster do EKS.
- Criação do seu cluster de EKS
- Configurar o kubectl para se comunicar com o cluster.
- Crie um grupo de nós gerenciados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
```

```
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
```

```
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
```

```
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
```

```
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
```

```
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"
```

```
SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."
```

```
# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
```

```
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"
```

```
echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNodegroups
  - UpdateKubeconfig

- Wait

Exemplos de código para CloudFront usar AWS SDKs

Os exemplos de código a seguir mostram como usar a Amazon CloudFront com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- CloudFront Guia do desenvolvedor — Mais informações sobre CloudFront.
- CloudFront Referência da API — Detalhes sobre todas as CloudFront ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CloudFront uso AWS SDKs
  - Ações para CloudFront usar AWS SDKs
    - Use CreateDistributioncom um AWS SDK ou CLI
    - Use CreateFunctioncom um AWS SDK
    - Usar CreateInvalidation com uma CLI
    - Use CreateKeyGroupcom um AWS SDK
    - Use CreatePublicKeycom um AWS SDK ou CLI
    - Use DeleteDistributioncom um AWS SDK ou CLI
    - Usar GetCloudFrontOriginAccessIdentity com uma CLI
    - Usar GetCloudFrontOriginAccessIdentityConfig com uma CLI

    - Usar GetDistribution com uma CLI
    - Use GetDistributionConfigcom um AWS SDK ou CLI
    - Usar ListCloudFrontOriginAccessIdentities com uma CLI
    - Use ListDistributionscom um AWS SDK ou CLI
    - Use UpdateDistributioncom um AWS SDK ou CLI
- Cenários para CloudFront usar AWS SDKs
  - Crie recursos de gerenciamento de SaaS AWS SDK
  - Excluir recursos de CloudFront assinatura usando AWS SDK
  - Conceitos básicos de uma rede de distribuição básica de conteúdo
  - Conceitos básicos de firewalls do aplicativo Web
  - Crie URLs e cookies assinados usando um AWS SDK
- CloudFront Exemplos de funções para CloudFront
  - Adicionar cabeçalhos de segurança HTTP a um evento de resposta do visualizador do CloudFront Functions
  - Adicionar um cabeçalho CORS a um evento de resposta do visualizador de CloudFront funções
  - Adicionar um cabeçalho de controle de cache a um evento de resposta do visualizador do CloudFront Functions
  - Adicionar um cabeçalho IP de cliente verdadeiro a um evento de solicitação do visualizador do CloudFront Functions
  - Adicionar um cabeçalho de origem a um evento de solicitação do Visualizador de CloudFront Funções
  - Adicione index.html para solicitar URLs sem um nome de arquivo em um evento de solicitação do Visualizador de CloudFront Funções
  - Normalizar os parâmetros da sequência de caracteres de consulta em uma solicitação do visualizador CloudFront do Functions
  - Redirecionar para um novo URL em um evento de solicitação do Visualizador do CloudFront Functions
  - Reescrever um URI de solicitação com base na KeyValueStore configuração de um evento de solicitação do visualizador de CloudFront funções
  - Encaminhar solicitações para uma origem mais próxima do visualizador em um evento de solicitação do visualizador do CloudFront Functions

  - Use pares de valores-chave em uma solicitação do Visualizador de CloudFront Funções

- Validar um token simples em uma solicitação do visualizador CloudFront do Functions

Exemplos básicos de CloudFront uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico da Amazon CloudFront com AWS SDKs.

Exemplos

- Ações para CloudFront usar AWS SDKs
  - Use CreateDistributioncom um AWS SDK ou CLI
  - Use CreateFunctioncom um AWS SDK
  - Usar CreateInvalidation com uma CLI
  - Use CreateKeyGroupcom um AWS SDK
  - Use CreatePublicKeycom um AWS SDK ou CLI
  - Use DeleteDistributioncom um AWS SDK ou CLI
  - Usar GetCloudFrontOriginAccessIdentity com uma CLI
  - Usar GetCloudFrontOriginAccessIdentityConfig com uma CLI
  - Usar GetDistribution com uma CLI
  - Use GetDistributionConfigcom um AWS SDK ou CLI
  - Usar ListCloudFrontOriginAccessIdentities com uma CLI
  - Use ListDistributionscom um AWS SDK ou CLI
  - Use UpdateDistributioncom um AWS SDK ou CLI

Ações para CloudFront usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar CloudFront ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a CloudFront API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para CloudFront usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon CloudFront API Reference.

Exemplos

- Use CreateDistributioncom um AWS SDK ou CLI
- Use CreateFunctioncom um AWS SDK
- Usar CreateInvalidation com uma CLI
- Use CreateKeyGroupcom um AWS SDK
- Use CreatePublicKeycom um AWS SDK ou CLI
- Use DeleteDistributioncom um AWS SDK ou CLI
- Usar GetCloudFrontOriginAccessIdentity com uma CLI
- Usar GetCloudFrontOriginAccessIdentityConfig com uma CLI
- Usar GetDistribution com uma CLI
- Use GetDistributionConfigcom um AWS SDK ou CLI
- Usar ListCloudFrontOriginAccessIdentities com uma CLI
- Use ListDistributionscom um AWS SDK ou CLI
- Use UpdateDistributioncom um AWS SDK ou CLI

Use **`CreateDistribution`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateDistribution`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma distribuição multilocatária e um locatário de distribuição
- Conceitos básicos de uma rede de distribuição básica de conteúdo

CLI

AWS CLI

Exemplo 1: Para criar uma CloudFront distribuição

O exemplo a seguir `create-distribution` cria uma distribuição para um bucket do S3 chamado `amzn-s3-demo-bucket` e também especifica `index.html` como o objeto raiz padrão, usando argumentos de linha de comandos.

```
aws cloudfront create-distribution \
    --origin-domain-name amzn-s3-demo-bucket.s3.amazonaws.com \
    --default-root-object index.html
```

Saída:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EMLARXS9EXAMPLE",
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-11-22T00:55:15.705Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
                        "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
```

```
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
```

```
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

Exemplo 2: Para criar uma CloudFront distribuição usando um arquivo JSON

O exemplo a seguir `create-distribution` cria uma distribuição para um bucket do S3 chamado `amzn-s3-demo-bucket` e também especifica `index.html` como o objeto raiz padrão, usando um arquivo JSON.

```
aws cloudfront create-distribution \
```

```
    --distribution-config file://dist-config.json
```

Conteúdo de `dist-config.json`:

```
{
    "CallerReference": "cli-example",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
                "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
```

```
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
```

```
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

Consulte a amostra de saída no exemplo 1.

Exemplo 3: Para criar uma distribuição CloudFront multilocatária com um certificado

O `create-distribution` exemplo a seguir cria uma CloudFront distribuição com suporte multilocatário e especifica um certificado TLS.

```
aws cloudfront create-distribution \
    --distribution-config file://dist-config.json
```

Conteúdo de `dist-config.json`:

```
{
    "CallerReference": "cli-example-with-cert",
    "Comment": "CLI example distribution",
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                "DomainName": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                "OriginPath": "/{{tenantName}}",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
```

```
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
        "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e5ABC",
        "ViewerProtocolPolicy": "allow-all",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": ["HEAD", "GET"],
            "CachedMethods": {
                "Quantity": 2,
                "Items": ["HEAD", "GET"]
            }
        }
    },
    "Enabled": true,
    "ViewerCertificate": {
        "ACMCertificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/191306a1-db01-49ca-90ef-fc414ee5dabc",
        "SSLSupportMethod": "sni-only"
    },
    "HttpVersion": "http2",
    "ConnectionMode": "tenant-only",
    "TenantConfig": {
        "ParameterDefinitions": [
            {
                "Name": "tenantName",
                "Definition": {
                    "StringSchema": {
                        "Comment": "tenantName parameter",
                        "DefaultValue": "root",
                        "Required": false
                    }
                }
            }
        ]
    }
}
```

Saída:

```
{
    "Location": "https://cloudfront.amazonaws.com/2020-05-31/distribution/
E1HVIAU7UABC",
    "ETag": "E20LT7R1BABC",
    "Distribution": {
        "Id": "E1HVIAU7U12ABC",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/E1HVIAU7U12ABC",
        "Status": "InProgress",
        "LastModifiedTime": "2025-07-10T20:33:31.117000+00:00",
        "InProgressInvalidationBatches": 0,
        "DomainName": "example.com",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ActiveTrustedKeyGroups": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example-with-cert",
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                        "DomainName": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                        "OriginPath": "/{{tenantName}}",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        },
                        "ConnectionAttempts": 3,
                        "ConnectionTimeout": 10,
                        "OriginShield": {
                            "Enabled": false
                        },
                        "OriginAccessControlId": ""
                    }
```

```
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                "TrustedKeyGroups": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": ["HEAD", "GET"],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": ["HEAD", "GET"]
                    }
                },
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": "",
                "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e5ABC",
                "GrpcConfig": {
                    "Enabled": false
                }
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "CLI example distribution",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
```

```
                "Bucket": "",
                "Prefix": ""
            },
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": false,
                "ACMCertificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/1954f095-11b6-4daf-9952-0c308a00abc",
                "SSLSupportMethod": "sni-only",
                "MinimumProtocolVersion": "TLSv1.2_2021",
                "Certificate": "arn:aws:acm:us-
east-1:123456789012:certificate/1954f095-11b6-4daf-9952-0c308a00abc",
                "CertificateSource": "acm"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "TenantConfig": {
                "ParameterDefinitions": [
                    {
                        "Name": "tenantName",
                        "Definition": {
                            "StringSchema": {
                                "Comment": "tenantName parameter",
                                "DefaultValue": "root",
                                "Required": false
                            }
                        }
                    }
                ]
            },
            "ConnectionMode": "tenant-only"
        }
    }
}
```

Para obter mais informações, consulte Como trabalhar com distribuições no Amazon CloudFront Developer Guide.

Exemplo 4: Para criar uma distribuição CloudFront multilocatária sem um certificado

O `create-distribution` exemplo a seguir cria uma CloudFront distribuição com suporte multilocatário, mas sem um certificado TLS.

```
aws cloudfront create-distribution \
    --distribution-config file://dist-config.json
```

Conteúdo de `dist-config.json`:

```
{
    "CallerReference": "cli-example",
    "Comment": "CLI example distribution",
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                "DomainName": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                "OriginPath": "/{{tenantName}}",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
        "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e5ABC",
        "ViewerProtocolPolicy": "allow-all",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
```

```
                    "HEAD",
                    "GET"
                ]
            }
        }
    },
    "Enabled": true,
    "HttpVersion": "http2",
    "ConnectionMode": "tenant-only",
    "TenantConfig": {
        "ParameterDefinitions": [
            {
                "Name": "tenantName",
                "Definition": {
                    "StringSchema": {
                        "Comment": "tenantName parameter",
                        "DefaultValue": "root",
                        "Required": false
                    }
                }
            }
        ]
    }
}
```

Saída:

```
{
    "Location": "https://cloudfront.amazonaws.com/2020-05-31/distribution/
E2GJ5J9QN12ABC",
    "ETag": "E37YLVVQIABC",
    "Distribution": {
        "Id": "E2GJ5J9QNABC",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/E2GJ5J9QN12ABC",
        "Status": "InProgress",
        "LastModifiedTime": "2025-07-10T20:35:20.565000+00:00",
        "InProgressInvalidationBatches": 0,
        "DomainName": "example.com",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ActiveTrustedKeyGroups": {
```

```
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example-no-cert",
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                        "DomainName": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                        "OriginPath": "/{{tenantName}}",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        },
                        "ConnectionAttempts": 3,
                        "ConnectionTimeout": 10,
                        "OriginShield": {
                            "Enabled": false
                        },
                        "OriginAccessControlId": ""
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                "TrustedKeyGroups": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
```

```
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": "",
                "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e5ABC",
                "GrpcConfig": {
                    "Enabled": false
                }
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "CLI example distribution",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "SSLSupportMethod": "sni-only",
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
```

```
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "TenantConfig": {
                "ParameterDefinitions": [
                    {
                        "Name": "tenantName",
                        "Definition": {
                            "StringSchema": {
                                "Comment": "tenantName parameter",
                                "DefaultValue": "root",
                                "Required": false
                            }
                        }
                    }
                ]
            },
            "ConnectionMode": "tenant-only"
        }
    }
}
```

Para obter mais informações, consulte Configurar distribuições no Amazon CloudFront Developer Guide.

- Para obter detalhes da API, consulte CreateDistributionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo a seguir usa um bucket do Amazon Simple Storage Service (Amazon S3) como origem de conteúdo.

Depois de criar a distribuição, o código cria um CloudFrontWaiterpara esperar até que a distribuição seja implantada antes de retornar a distribuição.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

        private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

        public static Distribution createDistribution(CloudFrontClient
 cloudFrontClient, S3Client s3Client,
                        final String bucketName, final String keyGroupId, final
 String originAccessControlId) {

                final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                                .get("x-amz-bucket-region").get(0);
                final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
                String originId = originDomain; // Use the originDomain value for
 the originId.

                // The service API requires some deprecated methods, such as
                // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
```

```
                CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
                                .distributionConfig(b1 -> b1
                                                .origins(b2 -> b2
                                                                .quantity(1)
                                                                .items(b3 -> b3

.domainName(originDomain)

.id(originId)

.s3OriginConfig(builder4 -> builder4

              .originAccessIdentity(

                              ""))

.originAccessControlId(

              originAccessControlId)))
                                                .defaultCacheBehavior(b2 -> b2

.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)
                                                                .minTTL(200L)

.forwardedValues(b5 -> b5

.cookies(cp -> cp

              .forward(ItemSelection.NONE))

.queryString(true))

.trustedKeyGroups(b3 -> b3

.quantity(1)

.items(keyGroupId)

.enabled(true))

.allowedMethods(b4 -> b4
```

```
.quantity(2)

.items(Method.HEAD, Method.GET)

.cachedMethods(b5 -> b5

              .quantity(2)

              .items(Method.HEAD,

                             Method.GET))))
                                                .cacheBehaviors(b -> b
                                                                .quantity(1)
                                                                .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(

              ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

              .quantity(1)

              .items(keyGroupId)

              .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

              .cookies(cp -> cp

                             .forward(ItemSelection.NONE))

              .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)
```

```
                .items(Method.HEAD,

                               Method.GET)

                .cachedMethods(b6 -> b6

                               .quantity(2)

                               .items(Method.HEAD,

                                              Method.GET)))))
                                               .enabled(true)
                                               .comment("Distribution built with
 java")

 .callerReference(Instant.now().toString())));

                final Distribution distribution =
 createDistResponse.distribution();
                logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
 distribution.domainName(),
                               distribution.id());
                logger.info("Waiting for distribution to be deployed ...");
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                                        .matched();
                        responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                        distribution.id());
                }
                return distribution;
        }
}
```

- Para obter detalhes da API, consulte CreateDistributiona Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Cria uma CloudFront distribuição básica, configurada com registro e armazenamento em cache.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
      -Logging_Prefix "help/" `
      -DistributionConfig_CallerReference Client1 `
      -DistributionConfig_DefaultRootObject index.html `
      -DefaultCacheBehavior_TargetOriginId $origin.Id `
      -ForwardedValues_QueryString $true `
      -Cookies_Forward all `
      -WhitelistedNames_Quantity 0 `
      -TrustedSigners_Enabled $false `
      -TrustedSigners_Quantity 0 `
      -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
      -DefaultCacheBehavior_MinTTL 1000 `
      -DistributionConfig_PriceClass "PriceClass_All" `
      -CacheBehaviors_Quantity 0 `
      -Aliases_Quantity 0
```

- Para obter detalhes da API, consulte CreateDistributionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Cria uma CloudFront distribuição básica, configurada com registro e armazenamento em cache.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
      -Logging_Prefix "help/" `
      -DistributionConfig_CallerReference Client1 `
      -DistributionConfig_DefaultRootObject index.html `
      -DefaultCacheBehavior_TargetOriginId $origin.Id `
      -ForwardedValues_QueryString $true `
      -Cookies_Forward all `
      -WhitelistedNames_Quantity 0 `
      -TrustedSigners_Enabled $false `
      -TrustedSigners_Quantity 0 `
      -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
      -DefaultCacheBehavior_MinTTL 1000 `
      -DistributionConfig_PriceClass "PriceClass_All" `
      -CacheBehaviors_Quantity 0 `
      -Aliases_Quantity 0
```

- Para obter detalhes da API, consulte CreateDistributionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateFunction`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateFunction`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

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

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

                Usage:
                    <functionName> <filePath>

                Where:
                    functionName - The name of the function to create.\s
                    filePath - The path to a file that contains the application
 logic for the function.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        String funArn = createNewFunction(cloudFrontClient, functionName,
filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient,
String functionName, String filePath) {
        try {
            InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                    .comment("Created by using the CloudFront Java API")
                    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                    .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                    .name(functionName)
                    .functionCode(functionCode)
                    .functionConfig(config)
                    .build();

            CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateFunctiona Referência AWS SDK for Java 2.x da API.

Usar **`CreateInvalidation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateInvalidation`.

CLI

AWS CLI

Para criar uma invalidação para uma distribuição CloudFront

O `create-invalidation` exemplo a seguir cria uma invalidação para os arquivos especificados na CloudFront distribuição especificada:

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --paths "/example-path/example-file.jpg" "/example-path/example-file2.png"
```

Saída:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I1JLWSDAP8FU89",
    "Invalidation": {
        "Id": "I1JLWSDAP8FU89",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:24:51.407Z",
        "InvalidationBatch": {
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file2.png",
                    "/example-path/example-file.jpg"
                ]
```

```
            },
            "CallerReference": "cli-1575570291-670203"
        }
    }
}
```

No exemplo anterior, a AWS CLI gerou automaticamente um aleatório. `CallerReference` Para especificar seu próprio `CallerReference` ou evitar passar os parâmetros de invalidação como argumentos da linha de comando, é possível usar um arquivo JSON. O exemplo a seguir cria uma invalidação para dois arquivos fornecendo os parâmetros de invalidação em um arquivo JSON chamado `inv-batch.json`:

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --invalidation-batch file://inv-batch.json
```

Conteúdo de `inv-batch.json`:

```
{
    "Paths": {
        "Quantity": 2,
        "Items": [
            "/example-path/example-file.jpg",
            "/example-path/example-file2.png"
        ]
    },
    "CallerReference": "cli-example"
}
```

Saída:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I2J0I21PCUYOIK",
    "Invalidation": {
        "Id": "I2J0I21PCUYOIK",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:40:49.413Z",
        "InvalidationBatch": {
            "Paths": {
                "Quantity": 2,
```

```
                "Items": [
                    "/example-path/example-file.jpg",
                    "/example-path/example-file2.png"
                ]
            },
            "CallerReference": "cli-example"
        }
    }
}
```

- Para obter detalhes da API, consulte CreateInvalidationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma nova invalidação em uma distribuição com um ID de EXAMPLENSTXAXE. CallerReference É um ID exclusivo escolhido pelo usuário; nesse caso, é usado um carimbo de data/hora representando 15 de maio de 2019 às 9h. A variável $Paths armazena três caminhos para arquivos de imagem e mídia que o usuário não deseja como parte do cache distribuído. O valor do Paths_Quantity parâmetro - é o número total de caminhos especificados no Paths_Item parâmetro -.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -
Paths_Quantity 3
```

Saída:

```
Invalidation                        Location

------------                        --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- Para obter detalhes da API, consulte CreateInvalidationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma nova invalidação em uma distribuição com um ID de EXAMPLENSTXAXE. CallerReference É um ID exclusivo escolhido pelo usuário; nesse caso, é usado um carimbo de data/hora representando 15 de maio de 2019 às 9h. A variável $Paths armazena três caminhos para arquivos de imagem e mídia que o usuário não deseja como parte do cache distribuído. O valor do Paths_Quantity parâmetro - é o número total de caminhos especificados no Paths_Item parâmetro -.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -
Paths_Quantity 3
```

Saída:

```
Invalidation                        Location

------------                        --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- Para obter detalhes da API, consulte CreateInvalidationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateKeyGroup`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateKeyGroup`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Um grupo de chaves exige pelo menos uma chave pública usada para verificar URLs ou cookies assinados.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
 LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b ->
 b.keyGroupConfig(c -> c
                .items(publicKeyId)
                .name("JavaKeyGroup" + UUID.randomUUID())))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- Para obter detalhes da API, consulte CreateKeyGroupa Referência AWS SDK for Java 2.x da API.

Use **`CreatePublicKey`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreatePublicKey`.

CLI

AWS CLI

Para criar uma chave CloudFront pública

O exemplo a seguir cria uma chave CloudFront pública fornecendo os parâmetros em um arquivo JSON chamado`pub-key-config.json`. Antes de usar esse comando, você deve

ter uma chave PEM-encoded pública. Para obter mais informações, consulte Criar um par de chaves RSA no Amazon CloudFront Developer Guide.

```
aws cloudfront create-public-key \
    --public-key-config file://pub-key-config.json
```

O arquivo `pub-key-config.json` é um documento JSON na pasta atual que contém o seguinte. Observe que a chave pública é codificada no formato PEM.

```
{
    "CallerReference": "cli-example",
    "Name": "ExampleKey",
    "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
    "Comment": "example public key"
}
```

Saída:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/public-key/
KDFB19YGCR002",
    "ETag": "E2QWRUHEXAMPLE",
    "PublicKey": {
        "Id": "KDFB19YGCR002",
        "CreatedTime": "2019-12-05T18:51:43.781Z",
        "PublicKeyConfig": {
            "CallerReference": "cli-example",
            "Name": "ExampleKey",
            "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
```

```
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
            "Comment": "example public key"
        }
    }
}
```

- Para obter detalhes da API, consulte CreatePublicKeyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo de código a seguir lê uma chave pública e a carrega na Amazon CloudFront.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
 LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient,
 String publicKeyFileName) {
        try (InputStream is =
 CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
```

```
                    .createPublicKey(b -> b.publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte CreatePublicKeya Referência AWS SDK for Java 2.x da API.

Use **`DeleteDistribution`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteDistribution`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de uma rede de distribuição básica de conteúdo

CLI

AWS CLI

Para excluir uma CloudFront distribuição

O exemplo a seguir exclui a CloudFront distribuição com o `IDEDFDVBD6EXAMPLE`. Você deve desabilitar uma distribuição antes de exclui-la. Use o comando update-distribution para desabilitar uma distribuição. Para obter mais informações, consulte exemplos do comando update-distribution.

Uma distribuição pode ser excluída depois de ser desabilitada. Para excluir uma distribuição, use a opção `--if-match` para fornecer o `ETag` da distribuição. Use o comando get-distribution ou get-distribution-config para obter o `ETag`.

```
aws cloudfront delete-distribution \
    --id EDFDVBD6EXAMPLE \
    --if-match E2QWRUHEXAMPLE
```

Quando bem-sucedido, este comando não tem saída.

- Para obter detalhes da API, consulte DeleteDistributionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo de código a seguir atualiza uma distribuição para desativada, usa um waiter que aguarda a implantação da alteração e, em seguida, exclui a distribuição.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
        private static final Logger logger =
 LoggerFactory.getLogger(DeleteDistribution.class);

        public static void deleteDistribution(final CloudFrontClient
 cloudFrontClient, final String distributionId) {
                // First, disable the distribution by updating it.
                GetDistributionResponse response =
 cloudFrontClient.getDistribution(b -> b
```

```
                                .id(distributionId));
                String etag = response.eTag();
                DistributionConfig distConfig =
response.distribution().distributionConfig();

                cloudFrontClient.updateDistribution(builder -> builder
                                .id(distributionId)
                                .distributionConfig(builder1 -> builder1

.cacheBehaviors(distConfig.cacheBehaviors())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .enabled(false)
                                                .origins(distConfig.origins())
                                                .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())

.priceClass(distConfig.priceClass())
                                                .aliases(distConfig.aliases())
                                                .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                                                .webACLId(distConfig.webACLId())

.originGroups(distConfig.originGroups()))
                                .ifMatch(etag));

                logger.info("Distribution [{}] is DISABLED, waiting for
deployment before deleting ...",
                                distributionId);
                GetDistributionResponse distributionResponse;
```

```
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distributionId)).matched();
                        distributionResponse = responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Could not disable distribution"));
                }

                DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient
                                .deleteDistribution(builder -> builder
                                                .id(distributionId)

 .ifMatch(distributionResponse.eTag()));
                if (deleteDistributionResponse.sdkHttpResponse().isSuccessful())
 {
                        logger.info("Distribution [{}] DELETED", distributionId);
                }
        }
}
```

- Para obter detalhes da API, consulte DeleteDistributiona Referência AWS SDK for Java 2.x da API.

Usar **`GetCloudFrontOriginAccessIdentity`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetCloudFrontOriginAccessIdentity`.

CLI

AWS CLI

Para obter uma identidade de acesso de CloudFront origem

O exemplo a seguir obtém a identidade de acesso de CloudFront origem (OAI) com a `IDE74FTE3AEXAMPLE`, incluindo sua `ETag` e a ID canônica associada do S3. O ID da OAI é retornado na saída dos comandos create-cloud-front-origin-access-identity e list-cloud-front-origin-access-identities.

```
aws cloudfront get-cloud-front-origin-access-identity --id E74FTE3AEXAMPLE
```

Saída:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentity": {
        "Id": "E74FTE3AEXAMPLE",
        "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
        "CloudFrontOriginAccessIdentityConfig": {
            "CallerReference": "cli-example",
            "Comment": "Example OAI"
        }
    }
}
```

- Para obter detalhes da API, consulte GetCloudFrontOriginAccessIdentityem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo retorna uma identidade de acesso de CloudFront origem específica da Amazon, especificada pelo parâmetro -Id. Embora o parâmetro -Id não seja obrigatório, se você não o especificar, nenhum resultado será retornado.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Saída:

```
      CloudFrontOriginAccessIdentityConfig    Id
     S3CanonicalUserId
      -----------------------------------    --
     -----------------
      Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
     4b6e...
```

- Para obter detalhes da API, consulte GetCloudFrontOriginAccessIdentityem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo retorna uma identidade de acesso de CloudFront origem específica da Amazon, especificada pelo parâmetro -Id. Embora o parâmetro -Id não seja obrigatório, se você não o especificar, nenhum resultado será retornado.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Saída:

```
     CloudFrontOriginAccessIdentityConfig    Id
    S3CanonicalUserId
     ------------------------------------    --
    -----------------
     Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
    4b6e...
```

- Para obter detalhes da API, consulte GetCloudFrontOriginAccessIdentityem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetCloudFrontOriginAccessIdentityConfig`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetCloudFrontOriginAccessIdentityConfig`.

CLI

AWS CLI

Para obter uma configuração de identidade de acesso de CloudFront origem

O exemplo a seguir obtém metadados sobre a identidade de acesso de CloudFront origem (OAI) com o `IDE74FTE3AEXAMPLE`, incluindo seu. `ETag` O ID da OAI é retornado na saída dos comandos create-cloud-front-origin-access-identity e list-cloud-front-origin-access-identities.

```
aws cloudfront get-cloud-front-origin-access-identity-config --id E74FTE3AEXAMPLE
```

Saída:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentityConfig": {
        "CallerReference": "cli-example",
        "Comment": "Example OAI"
    }
}
```

- Para obter detalhes da API, consulte GetCloudFrontOriginAccessIdentityConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo retorna informações de configuração sobre uma única identidade de acesso de CloudFront origem da Amazon, especificada pelo parâmetro -Id. Ocorrem erros se nenhum parâmetro -Id for especificado.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Saída:

```
      CallerReference                                           Comment
      ---------------                                           -------
      mycallerreference: 2/1/2011 1:16:32 PM                    Caller
 reference: 2/1/2011 1:16:32 PM
```

- Para obter detalhes da API, consulte GetCloudFrontOriginAccessIdentityConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo retorna informações de configuração sobre uma única identidade de acesso de CloudFront origem da Amazon, especificada pelo parâmetro -Id. Ocorrem erros se nenhum parâmetro -Id for especificado.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Saída:

```
     CallerReference                                          Comment
     ---------------                                          -------
     mycallerreference: 2/1/2011 1:16:32 PM                   Caller
 reference: 2/1/2011 1:16:32 PM
```

- Para obter detalhes da API, consulte GetCloudFrontOriginAccessIdentityConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetDistribution`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetDistribution`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de uma rede de distribuição básica de conteúdo

CLI

AWS CLI

Para obter uma CloudFront distribuição

O `get-distribution` exemplo a seguir obtém a CloudFront distribuição com o IDEDFDVBD6EXAMPLE, incluindo seu`ETag`. O ID de distribuição é retornado nos comandos create-distribution e list-distributions.

```
aws cloudfront get-distribution \
    --id EDFDVBD6EXAMPLE
```

Saída:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "Deployed",
        "LastModifiedTime": "2019-12-04T23:35:41.433Z",
```

```
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
                        "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
```

```
                }
            },
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 0,
            "AllowedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
            "DefaultTTL": 86400,
            "MaxTTL": 31536000,
            "Compress": false,
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "FieldLevelEncryptionId": ""
        },
        "CacheBehaviors": {
            "Quantity": 0
        },
        "CustomErrorResponses": {
            "Quantity": 0
        },
        "Comment": "",
        "Logging": {
            "Enabled": false,
            "IncludeCookies": false,
            "Bucket": "",
            "Prefix": ""
        },
```

```
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- Para obter detalhes da API, consulte GetDistributionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: recupera as informações de uma distribuição específica.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Para obter detalhes da API, consulte GetDistributionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: recupera as informações de uma distribuição específica.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Para obter detalhes da API, consulte GetDistributionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetDistributionConfig`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetDistributionConfig`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de uma rede de distribuição básica de conteúdo

CLI

AWS CLI

Para obter uma configuração CloudFront de distribuição

O exemplo a seguir obtém metadados sobre a CloudFront distribuição com o `IDEDFDVBD6EXAMPLE`, incluindo seu`ETag`. O ID de distribuição é retornado nos comandos create-distribution e list-distributions.

```
aws cloudfront get-distribution-config \
    --id EDFDVBD6EXAMPLE
```

Saída:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "DistributionConfig": {
        "CallerReference": "cli-example",
        "Aliases": {
            "Quantity": 0
        },
        "DefaultRootObject": "index.html",
        "Origins": {
            "Quantity": 1,
            "Items": [
                {
                    "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
                    "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                    "OriginPath": "",
                    "CustomHeaders": {
                        "Quantity": 0
                    },
```

```
                    "S3OriginConfig": {
                        "OriginAccessIdentity": ""
                    }
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
        },
        "DefaultCacheBehavior": {
            "TargetOriginId": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
            "ForwardedValues": {
                "QueryString": false,
                "Cookies": {
                    "Forward": "none"
                },
                "Headers": {
                    "Quantity": 0
                },
                "QueryStringCacheKeys": {
                    "Quantity": 0
                }
            },
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 0,
            "AllowedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
```

```
            "DefaultTTL": 86400,
            "MaxTTL": 31536000,
            "Compress": false,
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "FieldLevelEncryptionId": ""
        },
        "CacheBehaviors": {
            "Quantity": 0
        },
        "CustomErrorResponses": {
            "Quantity": 0
        },
        "Comment": "",
        "Logging": {
            "Enabled": false,
            "IncludeCookies": false,
            "Bucket": "",
            "Prefix": ""
        },
        "PriceClass": "PriceClass_All",
        "Enabled": true,
        "ViewerCertificate": {
            "CloudFrontDefaultCertificate": true,
            "MinimumProtocolVersion": "TLSv1",
            "CertificateSource": "cloudfront"
        },
        "Restrictions": {
            "GeoRestriction": {
                "RestrictionType": "none",
                "Quantity": 0
            }
        },
        "WebACLId": "",
        "HttpVersion": "http2",
        "IsIPV6Enabled": true
    }
}
```

- Para obter detalhes da API, consulte GetDistributionConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: recupera a configuração de uma distribuição específica.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Para obter detalhes da API, consulte GetDistributionConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: recupera a configuração de uma distribuição específica.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Para obter detalhes da API, consulte GetDistributionConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
```

```
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- Para obter detalhes da API, consulte a GetDistributionConfigReferência da API AWS SDK for Python (Boto3).

Usar **`ListCloudFrontOriginAccessIdentities`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListCloudFrontOriginAccessIdentities`.

CLI

AWS CLI

Para listar identidades de acesso de CloudFront origem

O exemplo a seguir obtém uma lista das identidades de acesso de CloudFront origem (OAIs) em sua AWS conta:

```
aws cloudfront list-cloud-front-origin-access-identities
```

Saída:

```
{
    "CloudFrontOriginAccessIdentityList": {
        "Items": [
            {
                "Id": "E74FTE3AEXAMPLE",
                "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
                "Comment": "Example OAI"
            },
            {
                "Id": "EH1HDMBEXAMPLE",
                "S3CanonicalUserId":
 "1489f6f2e6faacaae7ff64c4c3e6956c24f78788abfc1718c3527c263bf7a17EXAMPLE",
                "Comment": "Test OAI"
            },
            {
                "Id": "E2X2C9TEXAMPLE",
                "S3CanonicalUserId":
 "cbfeebb915a64749f9be546a45b3fcfd3a31c779673c13c4dd460911ae402c2EXAMPLE",
                "Comment": "Example OAI #2"
            }
        ]
    }
}
```

- Para obter detalhes da API, consulte ListCloudFrontOriginAccessIdentitiesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo retorna uma lista de identidades de acesso de CloudFront origem da Amazon. Como o MaxItem parâmetro - especifica um valor de 2, os resultados incluem duas identidades.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Saída:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- Para obter detalhes da API, consulte ListCloudFrontOriginAccessIdentitiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo retorna uma lista de identidades de acesso de CloudFront origem da Amazon. Como o MaxItem parâmetro - especifica um valor de 2, os resultados incluem duas identidades.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Saída:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- Para obter detalhes da API, consulte ListCloudFrontOriginAccessIdentitiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListDistributions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListDistributions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de firewalls do aplicativo Web

CLI

AWS CLI

Para listar CloudFront distribuições

O exemplo a seguir obtém uma lista das CloudFront distribuições em sua AWS conta.

```
aws cloudfront list-distributions
```

Saída:

```
{
    "DistributionList": {
        "Items": [
            {
                "Id": "E23YS8OEXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
E23YS8OEXAMPLE",
                "Status": "Deployed",
                "LastModifiedTime": "2024-08-05T18:23:40.375000+00:00",
                "DomainName": "abcdefgh12ijk.cloudfront.net",
                "Aliases": {
                    "Quantity": 0
                },
                "Origins": {
                    "Quantity": 1,
                    "Items": [
                        {
                            "Id": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                            "DomainName": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                            "OriginPath": "",
                            "CustomHeaders": {
                                "Quantity": 0
                            },
                            "S3OriginConfig": {
                                "OriginAccessIdentity": ""
                            },
                            "ConnectionAttempts": 3,
                            "ConnectionTimeout": 10,
                            "OriginShield": {
```

```
                                "Enabled": false
                            },
                            "OriginAccessControlId": "EIAP8PEXAMPLE"
                        }
                    ]
                },
                "OriginGroups": {
                    "Quantity": 0
                },
                "DefaultCacheBehavior": {
                    "TargetOriginId": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                    "TrustedSigners": {
                        "Enabled": false,
                        "Quantity": 0
                    },
                    "TrustedKeyGroups": {
                        "Enabled": false,
                        "Quantity": 0
                    },
                    "ViewerProtocolPolicy": "allow-all",
                    "AllowedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ],
                        "CachedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ]
                        }
                    },
                    "SmoothStreaming": false,
                    "Compress": true,
                    "LambdaFunctionAssociations": {
                        "Quantity": 0
                    },
                    "FunctionAssociations": {
                        "Quantity": 0
                    },
                    "FieldLevelEncryptionId": "",
```

```
                    "CachePolicyId": "658327ea-f89d-4fab-a63d-7e886EXAMPLE"
                },
                "CacheBehaviors": {
                    "Quantity": 0
                },
                "CustomErrorResponses": {
                    "Quantity": 0
                },
                "Comment": "",
                "PriceClass": "PriceClass_All",
                "Enabled": true,
                "ViewerCertificate": {
                    "CloudFrontDefaultCertificate": true,
                    "SSLSupportMethod": "vip",
                    "MinimumProtocolVersion": "TLSv1",
                    "CertificateSource": "cloudfront"
                },
                "Restrictions": {
                    "GeoRestriction": {
                        "RestrictionType": "none",
                        "Quantity": 0
                    }
                },
                "WebACLId": "",
                "HttpVersion": "HTTP2",
                "IsIPV6Enabled": true,
                "Staging": false
            }
        ]
    }
}
```

- Para obter detalhes da API, consulte ListDistributionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorno de distribuições.

```
Get-CFDistributionList
```

- Para obter detalhes da API, consulte ListDistributionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorno de distribuições.

```
Get-CFDistributionList
```

- Para obter detalhes da API, consulte ListDistributionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions["DistributionList"]["Quantity"] > 0:
            for distribution in distributions["DistributionList"]["Items"]:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(
                    f"Certificate Source: "
```

```
                    f"{distribution['ViewerCertificate']['CertificateSource']}"
                )
                if distribution["ViewerCertificate"]["CertificateSource"] ==
 "acm":
                    print(
                        f"Certificate: {distribution['ViewerCertificate']
['Certificate']}"
                    )
                print("")
        else:
            print("No CloudFront distributions detected.")
```

- Para obter detalhes da API, consulte a ListDistributionsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_fnt->listdistributions( ). " oo_result is returned for
testing purposes. "
        MESSAGE 'Retrieved list of CloudFront distributions.' TYPE 'I'.
      CATCH /aws1/cx_fntinvalidargument.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListDistributionsreferência da API AWS SDK for SAP ABAP.

Use **`UpdateDistribution`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateDistribution`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de uma rede de distribuição básica de conteúdo

CLI

AWS CLI

Exemplo 1: Para atualizar o objeto raiz padrão de uma CloudFront distribuição

O exemplo a seguir atualiza o objeto raiz padrão `index.html` para a CloudFront distribuição com o IDEDFDVBD6EXAMPLE.

```
aws cloudfront update-distribution \
    --id EDFDVBD6EXAMPLE \
    --default-root-object index.html
```

Saída:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:55:39.870Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "6b10378d-49be-4c4b-a642-419ccaf8f3b5",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
```

```
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "example-website",
                        "DomainName": "www.example.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "CustomOriginConfig": {
                            "HTTPPort": 80,
                            "HTTPSPort": 443,
                            "OriginProtocolPolicy": "match-viewer",
                            "OriginSslProtocols": {
                                "Quantity": 2,
                                "Items": [
                                    "SSLv3",
                                    "TLSv1"
                                ]
                            },
                            "OriginReadTimeout": 30,
                            "OriginKeepaliveTimeout": 5
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "example-website",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 1,
                        "Items": [
                            "*"
                        ]
                    },
                    "QueryStringCacheKeys": {
```

```
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
```

```
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http1.1",
            "IsIPV6Enabled": true
        }
    }
}
```

Exemplo 2: Para atualizar uma CloudFront distribuição

O exemplo a seguir desativa a CloudFront distribuição com o ID `EMLARXS9EXAMPLE` fornecendo a configuração de distribuição em um arquivo JSON chamado. `dist-config-disable.json` Para atualizar uma distribuição, use a opção `--if-match` para fornecer o `ETag` da distribuição. Use o comando get-distribution ou get-distribution-config para obter o `ETag`. Observe que o campo `Enabled` está definido como `false` no arquivo JSON.

Depois de usar o exemplo a seguir para desabilitar uma distribuição, use o comando delete-distribution para exclui-la.

```
aws cloudfront update-distribution \
    --id EMLARXS9EXAMPLE \
    --if-match E2QWRUHEXAMPLE \
    --distribution-config file://dist-config-disable.json
```

Conteúdo de `dist-config-disable.json`:

```
{
    "CallerReference": "cli-1574382155-496510",
    "Aliases": {
```

```
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-1574382155-273939",
                "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "amzn-s3-demo-
bucket.s3.amazonaws.com-1574382155-273939",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
```

```
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": false,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
```

```
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

Saída:

```
{
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:32:35.553Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-1574382155-496510",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-
bucket.s3.amazonaws.com-1574382155-273939",
                        "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
```

```
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-
bucket.s3.amazonaws.com-1574382155-273939",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
```

```
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": false,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- Para obter detalhes da API, consulte UpdateDistributionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import
 software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <id>\s

                Where:
                    id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
 idVal) {
        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                    .id(idVal)
                    .build();

            GetDistributionResponse response =
 cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();

            // Create a new DistributionConfig object and add new values to
 comment and
            // aliases
            DistributionConfig config1 = DistributionConfig.builder()
                    .aliases(config.aliases()) // You can pass in new values here
                    .comment("New Comment")
                    .cacheBehaviors(config.cacheBehaviors())
                    .priceClass(config.priceClass())
                    .defaultCacheBehavior(config.defaultCacheBehavior())
                    .enabled(config.enabled())
                    .callerReference(config.callerReference())
                    .logging(config.logging())
                    .originGroups(config.originGroups())
                    .origins(config.origins())
                    .restrictions(config.restrictions())
                    .defaultRootObject(config.defaultRootObject())
                    .webACLId(config.webACLId())
                    .httpVersion(config.httpVersion())
                    .viewerCertificate(config.viewerCertificate())
```

```
                    .customErrorResponses(config.customErrorResponses())
                    .build();

            UpdateDistributionRequest updateDistributionRequest =
 UpdateDistributionRequest.builder()
                    .distributionConfig(config1)
                    .id(disObject.id())
                    .ifMatch(response.eTag())
                    .build();

            cloudFrontClient.updateDistribution(updateDistributionRequest);

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

- Para obter detalhes da API, consulte UpdateDistributiona Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client
```

```
    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- Para obter detalhes da API, consulte a UpdateDistributionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Get the current distribution configuration and ETag "
        DATA(lo_distribution_config_result) = lo_fnt-
>getdistributionconfig( iv_id = iv_distribution_id ).
        DATA(lo_old_config) = lo_distribution_config_result-
>get_distributionconfig( ).
        DATA(lv_etag) = lo_distribution_config_result->get_etag( ).

        " Create a new distribution config with the updated comment "
        " Since the config object is immutable, we need to create a new one with
 all existing values "
        DATA(lo_new_config) = NEW /aws1/cl_fntdistributionconfig(
          iv_callerreference = lo_old_config->get_callerreference( )
          io_aliases = lo_old_config->get_aliases( )
          iv_defaultrootobject = lo_old_config->get_defaultrootobject( )
          io_origins = lo_old_config->get_origins( )
          io_origingroups = lo_old_config->get_origingroups( )
          io_defaultcachebehavior = lo_old_config->get_defaultcachebehavior( )
          io_cachebehaviors = lo_old_config->get_cachebehaviors( )
          io_customerrorresponses = lo_old_config->get_customerrorresponses( )
          iv_comment = iv_comment
          io_logging = lo_old_config->get_logging( )
          iv_priceclass = lo_old_config->get_priceclass( )
          iv_enabled = lo_old_config->get_enabled( )
          io_viewercertificate = lo_old_config->get_viewercertificate( )
          io_restrictions = lo_old_config->get_restrictions( )
          iv_webaclid = lo_old_config->get_webaclid( )
          iv_httpversion = lo_old_config->get_httpversion( )
          iv_isipv6enabled = lo_old_config->get_isipv6enabled( ) ).

        " Update the distribution with the modified configuration "
        lo_fnt->updatedistribution(
          io_distributionconfig = lo_new_config
          iv_id = iv_distribution_id
          iv_ifmatch = lv_etag ).
        MESSAGE 'CloudFront distribution updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_fntnosuchdistribution.
        MESSAGE 'Distribution does not exist.' TYPE 'E'.
      CATCH /aws1/cx_fntpreconditionfailed.
        MESSAGE 'Precondition failed - ETag mismatch.' TYPE 'E'.
      CATCH /aws1/cx_fntinvalidifmatchvrs.
        MESSAGE 'Invalid If-Match version.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateDistributionreferência da API AWS SDK for SAP ABAP.

Cenários para CloudFront usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns CloudFront com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas CloudFront ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie recursos de gerenciamento de SaaS AWS SDK
- Excluir recursos de CloudFront assinatura usando AWS SDK
- Conceitos básicos de uma rede de distribuição básica de conteúdo
- Conceitos básicos de firewalls do aplicativo Web
- Crie URLs e cookies assinados usando um AWS SDK

Crie recursos de gerenciamento de SaaS AWS SDK

O exemplo a seguir mostra como criar uma distribuição multilocatária e um locatário de distribuição com várias configurações.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo a seguir demonstra como criar uma distribuição multilocatária com parâmetros e certificado curinga.

```
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.ConnectionMode;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.HttpVersion;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.SSLSupportMethod;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateMultiTenantDistribution {
    public static Distribution
 CreateMultiTenantDistributionWithCert(CloudFrontClient cloudFrontClient,
                                                                    S3Client
 s3Client,
                                                                    final String
 bucketName,
                                                                    final String
 certificateArn) {
        // fetch the origin info if necessary
        final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the
 originId.

        CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
```

```
                        .comment("Template Distribution with cert built with
 java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .viewerCertificate(certBuilder -> certBuilder
                                .acmCertificateArn(certificateArn)
                                .sslSupportMethod(SSLSupportMethod.SNI_ONLY))
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        "")))))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName
 value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
 cfWaiter
                    .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                    .matched();
```

```
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
created"));
        }
        return distribution;
    }

    public static Distribution
CreateMultiTenantDistributionNoCert(CloudFrontClient cloudFrontClient,
                                                                S3Client s3Client,
                                                                final String
bucketName) {
        // fetch the origin info if necessary
        final String region = s3Client.headBucket(b ->
b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the
originId.

        CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with
java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        "")))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
```

```
                                                                .comment("tenantName
 value")
                                                                .defaultValue("root")
                                                                .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
 cfWaiter
                    .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
 created"));
        }
        return distribution;
    }
}
```

O exemplo a seguir demonstra como criar um locatário de distribuição associado a esse modelo, incluindo a utilização do parâmetro que declaramos acima. Observe que não precisamos adicionar informações do certificado aqui porque nosso domínio já está coberto pelo modelo principal.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
```

```
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import
 software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantNoCert(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String
 domain,
                                                                    String
 hostedZoneId) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("no-cert-tenant")
        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection
 group that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));
```

```
        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

Se o certificado do visualizador fosse omitido do modelo principal, você precisaria adicionar informações do certificado sobre os inquilinos associados a ele. O exemplo a seguir demonstra como fazer isso por meio de um certificado ACM arn que cobre o domínio necessário para o inquilino.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import
 software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {
```

```
   public static DistributionTenant
createDistributionTenantWithCert(CloudFrontClient cloudFrontClient,

Route53Client route53Client,
                                                                        String
distributionId,
                                                                        String
domain,
                                                                        String
hostedZoneId,
                                                                        String
certificateArn) {
       CreateDistributionTenantResponse createResponse =
cloudFrontClient.createDistributionTenant(builder -> builder
               .distributionId(distributionId)
               .domains(b1 -> b1
                       .domain(domain))
               .enabled(false)
               .name("tenant-with-cert")
               .parameters(b2 -> b2
                       .name("tenantName")
                       .value("myTenant"))
               .customizations(b3 -> b3
                       .certificate(b4 -> b4
                               .arn(certificateArn))) // NOTE: Cert must be in
Us-East-1 and cover the domain provided in this request

       );

       final DistributionTenant distributionTenant =
createResponse.distributionTenant();

       // Then update the Route53 hosted zone to point your domain at the
distribution tenant
       // We fetch the RoutingEndpoint to point to via the default connection
group that was created for your tenant
       final GetConnectionGroupResponse fetchedConnectionGroup =
cloudFrontClient.getConnectionGroup(builder -> builder
               .identifier(distributionTenant.connectionGroupId()));

       route53Client.changeResourceRecordSets(builder -> builder
               .hostedZoneId(hostedZoneId)
               .changeBatch(b1 -> b1
                       .comment("ChangeBatch comment")
```

```
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

O exemplo a seguir demonstra como fazer isso com uma solicitação de certificado CloudFront-hosted gerenciado. Isso é ideal se você ainda não tiver tráfego para o seu domínio. Nesse caso, criamos um ConnectionGroup para gerar um RoutingEndpoint. Em seguida, usamos isso RoutingEndpoint para criar registros DNS que verificam a propriedade do domínio e apontam para CloudFront. CloudFront em seguida, fornecerá automaticamente um token para validar a propriedade do domínio e criar um certificado gerenciado.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import
 software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantCfHosted(CloudFrontClient cloudFrontClient,
```

```
Route53Client route53Client,
                                                                        String
distributionId,
                                                                        String
domain,
                                                                        String
hostedZoneId) throws InterruptedException {
       CreateConnectionGroupResponse createConnectionGroupResponse =
cloudFrontClient.createConnectionGroup(builder -> builder
               .ipv6Enabled(true)
               .name("cf-hosted-connection-group")
               .enabled(true));

       route53Client.changeResourceRecordSets(builder -> builder
               .hostedZoneId(hostedZoneId)
               .changeBatch(b1 -> b1
                       .comment("cf-hosted domain validation record")
                       .changes(b2 -> b2
                               .resourceRecordSet(b3 -> b3
                                       .name(domain)
                                       .type(RRType.CNAME)
                                       .ttl(300L)
                                       .resourceRecords(b4 -> b4

.value(createConnectionGroupResponse.connectionGroup().routingEndpoint())))
                               .action("CREATE"))
               ));

       // Give the R53 record time to propagate, if it isn't being returned by
servers yet, the following call will fail
       Thread.sleep(60000);

       CreateDistributionTenantResponse createResponse =
cloudFrontClient.createDistributionTenant(builder -> builder
               .distributionId(distributionId)
               .domains(b1 -> b1
                       .domain(domain))

.connectionGroupId(createConnectionGroupResponse.connectionGroup().id())
               .enabled(false)
               .name("cf-hosted-tenant")
               .parameters(b2 -> b2
                       .name("tenantName")
```

```
                        .value("myTenant"))
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.CLOUDFRONT)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

O exemplo a seguir demonstra como fazer isso com uma solicitação de certificado gerenciado auto-hospedado. Isso é ideal se você tiver tráfego para o seu domínio e não puder tolerar o tempo de inatividade durante uma migração. No final deste exemplo, o inquilino será criado em um estado aguardando a validação do domínio e a configuração do DNS. Siga as etapas [aqui] (https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/managed-cloudfront-certificates.html#complete -domain-ownership) para concluir a configuração quando estiver pronto para migrar o tráfego.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import
 software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantSelfHosted(CloudFrontClient cloudFrontClient,
                                                                        String
 distributionId,
```

```
                                                                                String
 domain) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("self-hosted-tenant")
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.SELF_HOSTED)
                        .primaryDomainName(domain)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDistribution
  - CreateDistributionTenant

Excluir recursos de CloudFront assinatura usando AWS SDK

O exemplo de código a seguir mostra como excluir recursos que são usados para obter acesso a conteúdo restrito em um bucket do Amazon Simple Storage Service (Amazon S3).

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient,
            final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
                .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }
```

```
    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient,
 final String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
 cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
 final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteKeyGroup
  - DeleteOriginAccessControl
  - DeletePublicKey

Conceitos básicos de uma rede de distribuição básica de conteúdo

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Fazer upload do conteúdo no bucket.
- Crie uma CloudFront distribuição com o OAC
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudFront Getting Started Tutorial Script
# This script creates an S3 bucket, uploads sample content, creates a CloudFront
 distribution with OAC,
# and demonstrates how to access content through CloudFront.

set -euo pipefail

# Set up logging
LOG_FILE="cloudfront-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting CloudFront Getting Started Tutorial at $(date)"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created before error:"
    if [ -n "${BUCKET_NAME:-}" ]; then
        echo "- S3 Bucket: $BUCKET_NAME"
    fi
    if [ -n "${OAC_ID:-}" ]; then
        echo "- CloudFront Origin Access Control: $OAC_ID"
    fi
```

```
    if [ -n "${DISTRIBUTION_ID:-}" ]; then
        echo "- CloudFront Distribution: $DISTRIBUTION_ID"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "Cleaning up resources..."

    if [ -n "${DISTRIBUTION_ID:-}" ]; then
        echo "Disabling CloudFront distribution $DISTRIBUTION_ID..."

        # Get the current configuration and ETag in one call
        DIST_CONFIG=$(aws cloudfront get-distribution-config --id
 "$DISTRIBUTION_ID" 2>/dev/null) || {
            echo "Failed to get distribution config. Continuing with cleanup..."
            DIST_CONFIG=""
        }

        if [ -n "$DIST_CONFIG" ]; then
            ETAG=$(echo "$DIST_CONFIG" | jq -r '.ETag')

            # Modify and update distribution in one pipeline
            if echo "$DIST_CONFIG" | jq '.DistributionConfig.Enabled = false' | \
                aws cloudfront update-distribution \
                    --id "$DISTRIBUTION_ID" \
                    --distribution-config "$(cat)" \
                    --if-match "$ETAG" 2>/dev/null; then

                echo "Waiting for distribution to be disabled (this may take
 several minutes)..."
                aws cloudfront wait distribution-deployed --id "$DISTRIBUTION_ID"
 2>/dev/null || {
                    echo "Distribution deployment wait timed out. Proceeding with
 deletion..."
                }

                # Get fresh ETag for deletion
                DIST_CONFIG=$(aws cloudfront get-distribution-config --id
 "$DISTRIBUTION_ID" 2>/dev/null) || {
```

```
                    echo "Failed to get updated config. Skipping distribution
 deletion..."
                    DIST_CONFIG=""
                }

                if [ -n "$DIST_CONFIG" ]; then
                    ETAG=$(echo "$DIST_CONFIG" | jq -r '.ETag')
                    aws cloudfront delete-distribution --id "$DISTRIBUTION_ID" --
if-match "$ETAG" 2>/dev/null && \
                        echo "CloudFront distribution deleted." || \
                        echo "Failed to delete distribution. You may need to
 delete it manually."
                fi
            else
                echo "Failed to disable distribution. Continuing with cleanup..."
            fi
        fi
    fi

    if [ -n "${OAC_ID:-}" ]; then
        echo "Deleting Origin Access Control $OAC_ID..."
        OAC_DATA=$(aws cloudfront get-origin-access-control --id "$OAC_ID" 2>/
dev/null) || {
            echo "Failed to get Origin Access Control. You may need to delete it
 manually."
            OAC_DATA=""
        }

        if [ -n "$OAC_DATA" ]; then
            OAC_ETAG=$(echo "$OAC_DATA" | jq -r '.ETag')
            aws cloudfront delete-origin-access-control --id "$OAC_ID" --if-match
 "$OAC_ETAG" 2>/dev/null && \
                echo "Origin Access Control deleted." || \
                echo "Failed to delete Origin Access Control. You may need to
 delete it manually."
        fi
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "$BUCKET_IS_SHARED" != "true" ]; then
        echo "Deleting S3 bucket $BUCKET_NAME and its contents..."
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || {
            echo "Failed to remove bucket contents. Continuing with bucket
 deletion..."
        }
```

```
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null && \
            echo "S3 bucket deleted." || \
            echo "Failed to delete bucket. You may need to delete it manually."
    fi

    # Clean up temporary files
    rm -f temp_disabled_config.json distribution-config.json bucket-policy.json
 2>/dev/null || true
    rm -rf temp_content 2>/dev/null || true
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Validate AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Validate jq is available
if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi

# Validate AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Initialize variables
BUCKET_NAME=""
OAC_ID=""
DISTRIBUTION_ID=""
BUCKET_IS_SHARED=false

# Generate a random identifier for the bucket name using secure random
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
    handle_error "Failed to generate random identifier"
fi

# Check for shared prereq bucket and get account ID in parallel calls
```

```
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null) || PREREQ_BUCKET=""

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="cloudfront-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Get AWS account ID early
ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
if [ $? -ne 0 ]; then
    handle_error "Failed to get AWS account ID"
fi

# Validate account ID format
if ! [[ "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    handle_error "Invalid AWS account ID format: $ACCOUNT_ID"
fi

# Create a temporary directory for content with restrictive permissions
TEMP_DIR="temp_content"
mkdir -p "$TEMP_DIR/css"
chmod 700 "$TEMP_DIR"
if [ $? -ne 0 ]; then
    handle_error "Failed to create temporary directory"
fi

# Step 1: Create an S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" != "true" ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    aws s3 mb "s3://$BUCKET_NAME" --region us-east-1
    if [ $? -ne 0 ]; then
        handle_error "Failed to create S3 bucket"
    fi
```

```
    aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=cloudfront-
gettingstarted}]'

    # Batch bucket configuration calls for efficiency
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled &
    aws s3api put-public-access-block \
        --bucket "$BUCKET_NAME" \
        --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 &
    aws s3api put-bucket-encryption \
        --bucket "$BUCKET_NAME" \
        --server-side-encryption-configuration '{
            "Rules": [
                {
                    "ApplyServerSideEncryptionByDefault": {
                        "SSEAlgorithm": "AES256"
                    }
                }
            ]
        }' &
    wait
fi

# Step 2: Create sample content
echo "Creating sample content..."
cat > "$TEMP_DIR/index.html" << 'EOF'
<!DOCTYPE html>
<html>
<head>
    <title>Hello World</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>
EOF

cat > "$TEMP_DIR/css/styles.css" << 'EOF'
body {
    font-family: Arial, sans-serif;
```

```
    margin: 40px;
    background-color: #f5f5f5;
}
h1 {
    color: #333;
    text-align: center;
}
EOF

# Set restrictive permissions on content files
chmod 600 "$TEMP_DIR/index.html" "$TEMP_DIR/css/styles.css"

# Step 3: Upload content to the S3 bucket with encryption and metadata
echo "Uploading content to S3 bucket..."
aws s3 cp "$TEMP_DIR/" "s3://$BUCKET_NAME/" --recursive \
    --sse AES256 \
    --metadata "Source=CloudFrontTutorial"
if [ $? -ne 0 ]; then
    handle_error "Failed to upload content to S3 bucket"
fi

# Step 4: Create Origin Access Control
echo "Creating Origin Access Control..."
OAC_RESPONSE=$(aws cloudfront create-origin-access-control \
    --origin-access-control-config Name="oac-for-
$BUCKET_NAME",SigningProtocol=sigv4,SigningBehavior=always,OriginAccessControlOriginType=

if [ $? -ne 0 ]; then
    handle_error "Failed to create Origin Access Control"
fi

OAC_ID=$(echo "$OAC_RESPONSE" | jq -r '.OriginAccessControl.Id')
if [ -z "$OAC_ID" ] || [ "$OAC_ID" = "null" ]; then
    handle_error "Failed to extract OAC ID from response"
fi

# Validate OAC ID format (alphanumeric and hyphens)
if ! [[ "$OAC_ID" =~ ^[A-Z0-9]+$ ]]; then
    handle_error "Invalid OAC ID format: $OAC_ID"
fi

echo "Created Origin Access Control with ID: $OAC_ID"

# Step 5: Create CloudFront distribution
```

```
echo "Creating CloudFront distribution..."

# Validate bucket name format
if ! [[ "$BUCKET_NAME" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]]; then
    handle_error "Invalid S3 bucket name format: $BUCKET_NAME"
fi

# Create distribution configuration with improved security settings
cat > distribution-config.json << EOF
{
    "CallerReference": "cli-tutorial-$(date +%s)",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "S3-$BUCKET_NAME",
                "DomainName": "$BUCKET_NAME.s3.amazonaws.com",
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                },
                "OriginAccessControlId": "$OAC_ID"
            }
        ]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "S3-$BUCKET_NAME",
        "ViewerProtocolPolicy": "redirect-to-https",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": ["GET", "HEAD"],
            "CachedMethods": {
                "Quantity": 2,
                "Items": ["GET", "HEAD"]
            }
        },
        "DefaultTTL": 86400,
        "MinTTL": 0,
        "MaxTTL": 31536000,
        "Compress": true,
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            }
```

```
        }
    },
    "Comment": "CloudFront distribution for tutorial",
    "Enabled": true,
    "WebACLId": "",
    "HttpVersion": "http2and3"
}
EOF

# Set restrictive permissions on config file before passing credentials
chmod 600 distribution-config.json

DIST_RESPONSE=$(aws cloudfront create-distribution --distribution-config file:// distribution-config.json)
if [ $? -ne 0 ]; then
    handle_error "Failed to create CloudFront distribution"
fi

DISTRIBUTION_ID=$(echo "$DIST_RESPONSE" | jq -r '.Distribution.Id')
DOMAIN_NAME=$(echo "$DIST_RESPONSE" | jq -r '.Distribution.DomainName')

if [ -z "$DISTRIBUTION_ID" ] || [ "$DISTRIBUTION_ID" = "null" ] || [ -z "$DOMAIN_NAME" ] || [ "$DOMAIN_NAME" = "null" ]; then
    handle_error "Failed to extract distribution ID or domain name from response"
fi

# Validate distribution ID format
if ! [[ "$DISTRIBUTION_ID" =~ ^[A-Z0-9]+$ ]]; then
    handle_error "Invalid distribution ID format: $DISTRIBUTION_ID"
fi

echo "Created CloudFront distribution with ID: $DISTRIBUTION_ID"
echo "CloudFront domain name: $DOMAIN_NAME"

# Tag the CloudFront distribution
aws cloudfront tag-resource --resource "arn:aws:cloudfront::$ACCOUNT_ID:distribution/$DISTRIBUTION_ID" --tags 'Items=[{Key=project,Value=doc-smith},{Key=tutorial,Value=cloudfront-gettingstarted}]'

# Step 6: Update S3 bucket policy
echo "Updating S3 bucket policy..."

cat > bucket-policy.json << EOF
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::$BUCKET_NAME/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn": "arn:aws:cloudfront::
$ACCOUNT_ID:distribution/$DISTRIBUTION_ID"
                }
            }
        }
    ]
}
EOF

# Set restrictive permissions on policy file
chmod 600 bucket-policy.json

aws s3api put-bucket-policy --bucket "$BUCKET_NAME" --policy file://bucket-
policy.json
if [ $? -ne 0 ]; then
    handle_error "Failed to update S3 bucket policy"
fi

# Step 7: Wait for distribution to deploy
echo "Waiting for CloudFront distribution to deploy (this may take 5-10
 minutes)..."
aws cloudfront wait distribution-deployed --id "$DISTRIBUTION_ID" 2>/dev/null ||
 {
    echo "Warning: Distribution deployment wait timed out. The distribution may
 still be deploying."
}

echo "CloudFront distribution is now deployed."

# Step 8: Display access information
echo ""
echo "===== CloudFront Distribution Setup Complete ======"
```

```
echo "You can access your content at: https://$DOMAIN_NAME/index.html"
echo ""
echo "Resources created:"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- CloudFront Origin Access Control: $OAC_ID"
echo "- CloudFront Distribution: $DISTRIBUTION_ID"
echo ""
echo "To clean up resources, run: cleanup"
echo ""

echo "Tutorial completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateDistribution
  - CreateOriginAccessControl
  - DeleteDistribution
  - DeleteOriginAccessControl
  - GetDistribution
  - GetDistributionConfig
  - GetOriginAccessControl
  - UpdateDistribution
  - WaitDistributionDeployed

Conceitos básicos de firewalls do aplicativo Web

O exemplo de código a seguir mostra como:

- Criar uma ACL da web.
- Adicionar uma regra de correspondência de string.
- Adicionar regras gerenciadas.
- Configurar o registro em log da
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
```

```
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
```

```
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken'
 2>/dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the
 AWS Console."
```

```
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi
```

```
# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "====================================================="
echo "STEP 2: Adding String Match Rule"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
```

```
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
```

```
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
```

```
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
```

```
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
```

```
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "==================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "==================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "===================================================="
```

```
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "====================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo
 "$first_dist" | grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution
 successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
```

```
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

Crie URLs e cookies assinados usando um AWS SDK

O exemplo de código a seguir mostra como criar URLs e cookies assinados que permitem acesso a recursos restritos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a CannedSignerRequestclasse para assinar URLs ou cookies com uma política predefinida.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
    }
```

```
}
```

Use a CustomSignerRequestclasse para assinar URLs ou cookies com uma política personalizada. O `activeDate` e `ipRange` são métodos opcionais.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                // .resourceUrlPattern("https://*.example.com/*")  // Optional.
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

O exemplo a seguir demonstra o uso da CloudFrontUtilitiesclasse para produzir cookies e URLs assinados. Veja este exemplo de código em GitHub.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
 getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
```

```
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
 getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- Para obter detalhes da API, consulte CloudFrontUtilitiesa Referência AWS SDK for Java 2.x da API.

CloudFront Exemplos de funções para CloudFront

Os exemplos de código a seguir mostram como usar CloudFront com AWS SDKs.

Exemplos

- Adicionar cabeçalhos de segurança HTTP a um evento de resposta do visualizador do CloudFront Functions
- Adicionar um cabeçalho CORS a um evento de resposta do visualizador de CloudFront funções
- Adicionar um cabeçalho de controle de cache a um evento de resposta do visualizador do CloudFront Functions
- Adicionar um cabeçalho IP de cliente verdadeiro a um evento de solicitação do visualizador do CloudFront Functions
- Adicionar um cabeçalho de origem a um evento de solicitação do Visualizador de CloudFront Funções
- Adicione index.html para solicitar URLs sem um nome de arquivo em um evento de solicitação do Visualizador de CloudFront Funções

- Normalizar os parâmetros da sequência de caracteres de consulta em uma solicitação do visualizador CloudFront do Functions
- Redirecionar para um novo URL em um evento de solicitação do Visualizador do CloudFront Functions
- Reescrever um URI de solicitação com base na KeyValueStore configuração de um evento de solicitação do visualizador de CloudFront funções
- Encaminhar solicitações para uma origem mais próxima do visualizador em um evento de solicitação do visualizador do CloudFront Functions
- Use pares de valores-chave em uma solicitação do Visualizador de CloudFront Funções
- Validar um token simples em uma solicitação do visualizador CloudFront do Functions

Adicionar cabeçalhos de segurança HTTP a um evento de resposta do visualizador do CloudFront Functions

O exemplo de código a seguir mostra como adicionar cabeçalhos de segurança HTTP a um evento de resposta do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
async function handler(event) {
    var response = event.response;
    var headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation
    headers['strict-transport-security'] = { value: 'max-age=63072000;
 includeSubdomains; preload'};
```

```
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
 'self'; script-src 'self'; style-src 'self'; object-src 'none'; frame-ancestors
 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};
    headers['referrer-policy'] = {value: 'same-origin'};

    // Return the response to viewers
    return response;
}
```

Adicionar um cabeçalho CORS a um evento de resposta do visualizador de CloudFront funções

O exemplo de código a seguir mostra como adicionar um cabeçalho CORS a um evento de resposta do visualizador de CloudFront funções.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
async function handler(event)  {
    var request = event.request;
    var response  = event.response;

    // If Access-Control-Allow-Origin CORS header is missing, add it.
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation.
    if (!response.headers['access-control-allow-origin'] &&
 request.headers['origin']) {
        response.headers['access-control-allow-origin'] = {value:
 request.headers['origin'].value};
        console.log("Access-Control-Allow-Origin was missing, adding it now.");
```

```
    }

    return response;
}
```

Adicionar um cabeçalho de controle de cache a um evento de resposta do visualizador do CloudFront Functions

O exemplo de código a seguir mostra como adicionar um cabeçalho de controle de cache a um evento de resposta do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
async function handler(event) {
    var response = event.response;
    var headers = response.headers;

    if (response.statusCode >= 200 && response.statusCode < 400) {
        // Set the cache-control header
        headers['cache-control'] = {value: 'public, max-age=63072000'};
    }

    // Return response to viewers
    return response;
}
```

Adicionar um cabeçalho IP de cliente verdadeiro a um evento de solicitação do visualizador do CloudFront Functions

O exemplo de código a seguir mostra como adicionar um cabeçalho IP de cliente verdadeiro a um evento de solicitação do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
async function handler(event) {
    var request = event.request;
    var clientIP = event.viewer.ip;

    //Add the true-client-ip header to the incoming request
    request.headers['true-client-ip'] = {value: clientIP};

    return request;
}
```

Adicionar um cabeçalho de origem a um evento de solicitação do Visualizador de CloudFront Funções

O exemplo de código a seguir mostra como adicionar um cabeçalho de origem a um evento de solicitação do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
async function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;

   // If origin header is missing, set it equal to the host header.
   if (!headers.origin)
       headers.origin = {value:`https://${host}`};

   return request;
}
```

Adicione index.html para solicitar URLs sem um nome de arquivo em um evento de solicitação do Visualizador de CloudFront Funções

O exemplo de código a seguir mostra como adicionar index.html para solicitar URLs sem um nome de arquivo em um evento de solicitação do Visualizador de CloudFront Funções.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
async function handler(event) {
    var request = event.request;
    var uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
}
```

Normalizar os parâmetros da sequência de caracteres de consulta em uma solicitação do visualizador CloudFront do Functions

O exemplo de código a seguir mostra como normalizar os parâmetros da sequência de caracteres de consulta em uma solicitação do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
function handler(event) {
     var qs=[];
     for (var key in event.request.querystring) {
         if (event.request.querystring[key].multiValue) {
             event.request.querystring[key].multiValue.forEach((mv) =>
 {qs.push(key + "=" + mv.value)});
         } else {
```

```
            qs.push(key + "=" + event.request.querystring[key].value);
        }
    };

    event.request.querystring = qs.sort().join('&');


    return event.request;
}
```

Redirecionar para um novo URL em um evento de solicitação do Visualizador do CloudFront Functions

O exemplo de código a seguir mostra como redirecionar para um novo URL em um evento de solicitação do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
async function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;
    var country = 'DE' // Choose a country code
    var newurl = `https://${host}/de/index.html`; // Change the redirect URL to
 your choice

    if (headers['cloudfront-viewer-country']) {
        var countryCode = headers['cloudfront-viewer-country'].value;
        if (countryCode === country) {
            var response = {
                statusCode: 302,
                statusDescription: 'Found',
```

```
                headers:
                    { "location": { "value": newurl } }
                }

            return response;
        }
    }
    return request;
}
```

Reescrever um URI de solicitação com base na KeyValueStore configuração de um evento de solicitação do visualizador de CloudFront funções

O exemplo de código a seguir mostra como reescrever um URI de solicitação com base na KeyValueStore configuração de um evento de solicitação do visualizador de CloudFront funções.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
import cf from 'cloudfront';

// (Optional) Replace KVS_ID with actual KVS ID
const kvsId = "KVS_ID";
// enable stickiness by setting a cookie from origin or using another edge
 function
const stickinessCookieName = "appversion";
// set to true to enable console logging
const loggingEnabled = false;

// function rewrites the request uri based on configuration in KVS
// example config in KVS in key:value format
// "latest": {"a_weightage": .8, "a_url": "v1", "b_url": "v2"}
// given above key and value in KVS the request uri will be rewritten
```

```
// for example http(s)://domain/latest/something/else will be rewritten as
 http(s)://domain/v1/something/else or http(s)://domain/v2/something/else
 depending on weightage
// if no configuration is found, then the request is returned as is
async function handler(event) {
    // NOTE: This example function is for a viewer request event trigger.
    // Choose viewer request for event trigger when you associate this function
 with a distribution.
    const request = event.request;
    const pathSegments = request.uri.split('/');
    const key = pathSegments[1];

    // if empty path segment or if there is valid stickiness cookie
    // then skip call to KVS and let the request continue.
    if (!key || hasValidSticknessCookie(request.cookies[stickinessCookieName],
 key)) {
        return event.request;
    }

    try {
        // get the prefix replacement from KVS
        const replacement = await getPathPrefixByWeightage(key);
        if (!replacement) {
            return event.request;
        }
        //Replace the first path with the replacement
        pathSegments[1] = replacement;
        log(`using prefix ${pathSegments[1]}`)
        const newUri = pathSegments.join('/');
        log(`${request.uri} -> ${newUri}`);
        request.uri = newUri;

        return request;
    } catch (err) {
        // No change to the path if the key is not found or any other error
        log(`request uri: ${request.uri}, error: ${err}`);
    }
    // no change to path - return request
    return event.request;
}

// function to get the prefix from KVS
async function getPathPrefixByWeightage(key) {
    const kvsHandle = cf.kvs(kvsId);
```

```
    // get the weightage config from KVS
    const kvsResponse = await kvsHandle.get(key);
    const weightageConfig = JSON.parse(kvsResponse);
    // no configuration - return null
    if (!weightageConfig || !isFinite(weightageConfig.a_weightage)) {
        return null;
    }
    // return the url based on weightage
    // return null if no url is configured
    if (Math.random() <= weightageConfig.a_weightage) {
        return weightageConfig.a_url ? weightageConfig.a_url: null;
    } else {
        return weightageConfig.b_url ? weightageConfig.b_url : null;
    }
}

// function to check if the stickiness cookie is valid
function hasValidStickinessCookie(stickinessCookie, pathSegment) {
    // if the value exists and it matches pathSegment
    return (stickinessCookie && stickinessCookie.value === pathSegment)
}

function log(message) {
    if (loggingEnabled) {
        console.log(message);
    }
}
```

Encaminhar solicitações para uma origem mais próxima do visualizador em um evento de solicitação do visualizador do CloudFront Functions

O exemplo de código a seguir mostra como rotear solicitações para uma origem mais próxima do visualizador em um evento de solicitação do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
import cf from 'cloudfront';

function handler(event) {
    const request = event.request;
    const headers = request.headers;
    const country = headers['cloudfront-viewer-country'] &&
        headers['cloudfront-viewer-country'].value;

    //List of Regions with S3 buckets containing content
    const countryToRegion = {
        'DE': 'eu-central-1',
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    };

    const DEFAULT_REGION = 'us-east-1';

    const selectedRegion = (country && countryToRegion[country]) ||
 DEFAULT_REGION;

    const domainName =
        `cloudfront-functions-demo-bucket-in-${selectedRegion}.s3.
${selectedRegion}.amazonaws.com`;

    cf.updateRequestOrigin({
        "domainName": domainName,
        "originAccessControlConfig": {
            "enabled": true,
            "region": selectedRegion,
```

```
            "signingBehavior": "always",
            "signingProtocol": "sigv4",
            "originType": "s3"
        },
    });

    return request;
}
```

Use pares de valores-chave em uma solicitação do Visualizador de CloudFront Funções

O exemplo de código a seguir mostra como usar pares de valores-chave em uma solicitação do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
import cf from 'cloudfront';

// This fails if there is no key value store associated with the function
const kvsHandle = cf.kvs();

// Remember to associate the KVS with your function before referencing KVS in
 your code.
// https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/kvs-with-
functions-associate.html
async function handler(event) {
    const request = event.request;
    // Use the first segment of the pathname as key
    // For example http(s)://domain/<key>/something/else
    const pathSegments = request.uri.split('/')
    const key = pathSegments[1]
    try {
```

```
        // Replace the first path of the pathname with the value of the key
        // For example http(s)://domain/<value>/something/else
        pathSegments[1] = await kvsHandle.get(key);
        const newUri = pathSegments.join('/');
        console.log(`${request.uri} -> ${newUri}`)
        request.uri = newUri;
    } catch (err) {
        // No change to the pathname if the key is not found
        console.log(`${request.uri} | ${err}`);
    }
    return request;
}
```

Validar um token simples em uma solicitação do visualizador CloudFront do Functions

O exemplo de código a seguir mostra como validar um token simples em uma solicitação do visualizador do CloudFront Functions.

JavaScript

JavaScript runtime 2.0 para CloudFront Funções

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório de exemplos do CloudFront Functions.

```
import crypto from 'crypto';
import cf from 'cloudfront';

//Response when JWT is not valid.
const response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

// Remember to associate the KVS with your function before calling the const
 kvsKey = 'jwt.secret'.
```

```
// https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/kvs-with-
functions-associate.html
const kvsKey = 'jwt.secret';
// set to true to enable console logging
const loggingEnabled = false;


function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
    }
    // check segments
    const segments = token.split('.');
    if (segments.length !== 3) {
        throw new Error('Not enough or too many segments');
    }

    // All segment should be base64
    const headerSeg = segments[0];
    const payloadSeg = segments[1];
    const signatureSeg = segments[2];

    // base64 decode and parse JSON
    const payload = JSON.parse(_base64urlDecode(payloadSeg));

    if (!noVerify) {
        const signingMethod = 'sha256';
        const signingType = 'hmac';

        // Verify signature. `sign` will return base64 string.
        const signingInput = [headerSeg, payloadSeg].join('.');

        if (!_verify(signingInput, key, signingMethod, signingType,
 signatureSeg)) {
            throw new Error('Signature verification failed');
        }

        // Support for nbf and exp claims.
        // According to the RFC, they should be in seconds.
        if (payload.nbf && Date.now() < payload.nbf*1000) {
            throw new Error('Token not yet active');
        }
```

```
        if (payload.exp && Date.now() > payload.exp*1000) {
            throw new Error('Token expired');
        }
    }

    return payload;
}

//Function to ensure a constant time comparison to prevent
//timing side channels.
function _constantTimeEquals(a, b) {
    if (a.length != b.length) {
        return false;
    }

    let xor = 0;
    for (let i = 0; i < a.length; i++) {
    xor |= (a.charCodeAt(i) ^ b.charCodeAt(i));
    }

    return 0 === xor;
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return _constantTimeEquals(signature, _sign(input, key, method));
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return Buffer.from(str, 'base64url')
}

async function handler(event) {
    let request = event.request;

    //Secret key used to verify JWT token.
```

```
    //Update with your own key.
    const secret_key = await getSecret()

    if(!secret_key) {
        return response401;
    }

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        log("Error: No JWT in the querystring");
        return response401;
    }

    const jwtToken = request.querystring.jwt.value;

    try{
        jwt_decode(jwtToken, secret_key);
    }
    catch(e) {
        log(e);
        return response401;
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    log("Valid JWT token");
    return request;
}

// get secret from key value store
async function getSecret() {
    // initialize cloudfront kv store and get the key value
    try {
        const kvsHandle = cf.kvs();
        return await kvsHandle.get(kvsKey);
    } catch (err) {
        log(`Error reading value for key: ${kvsKey}, error: ${err}`);
        return null;
    }

}

function log(message) {
    if (loggingEnabled) {
```

```
        console.log(message);
    }
}
```

Exemplos de código para CloudTrail usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS CloudTrail com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- CloudTrail Guia do desenvolvedor — Mais informações sobre CloudTrail.
- CloudTrail Referência da API — Detalhes sobre todas as CloudTrail ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CloudTrail uso AWS SDKs
  - Ações para CloudTrail usar AWS SDKs
    - Use CreateTrailcom um AWS SDK ou CLI
    - Use DeleteTrailcom um AWS SDK ou CLI
    - Use DescribeTrailcom um AWS SDK
    - Usar DescribeTrails com uma CLI
    - Usar GetTrailStatus com uma CLI
    - Use ListTrailscom um AWS SDK
    - Use LookupEventscom um AWS SDK ou CLI
    - Usar StartLogging com uma CLI

- Usar StopLogging com uma CLI
- Usar UpdateTrail com uma CLI

Exemplos básicos de CloudTrail uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS CloudTrail com AWS SDKs.

Exemplos

- Ações para CloudTrail usar AWS SDKs
  - Use CreateTrailcom um AWS SDK ou CLI
  - Use DeleteTrailcom um AWS SDK ou CLI
  - Use DescribeTrailcom um AWS SDK
  - Usar DescribeTrails com uma CLI
  - Usar GetTrailStatus com uma CLI
  - Use ListTrailscom um AWS SDK
  - Use LookupEventscom um AWS SDK ou CLI
  - Usar StartLogging com uma CLI
  - Usar StopLogging com uma CLI
  - Usar UpdateTrail com uma CLI

Ações para CloudTrail usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar CloudTrail ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS CloudTrail.

Exemplos

- Use CreateTrailcom um AWS SDK ou CLI
- Use DeleteTrailcom um AWS SDK ou CLI
- Use DescribeTrailcom um AWS SDK
- Usar DescribeTrails com uma CLI

- Usar GetTrailStatus com uma CLI
- Use ListTrailscom um AWS SDK
- Use LookupEventscom um AWS SDK ou CLI
- Usar StartLogging com uma CLI
- Usar StopLogging com uma CLI
- Usar UpdateTrail com uma CLI

Use **`CreateTrail`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateTrail`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Routine which creates an AWS CloudTrail trail.
/*!
  \param trailName: The name of the CloudTrail trail.
  \param bucketName: The Amazon S3 bucket designate for publishing logs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::createTrail(const Aws::String trailName,
                                     const Aws::String bucketName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);
    Aws::CloudTrail::Model::CreateTrailRequest request;
    request.SetName(trailName);
    request.SetS3BucketName(bucketName);

    Aws::CloudTrail::Model::CreateTrailOutcome outcome = trailClient.CreateTrail(
            request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Failed to create trail " << trailName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTraila Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar uma trilha

O comando `create-trail` a seguir cria uma trilha multirregional chamada `Trail1` e especifica um bucket do S3.

```
aws cloudtrail create-trail \
    --name Trail1 \
    --s3-bucket-name amzn-s3-demo-bucket \
    --is-multi-region-trail
```

Saída:

```
{
    "IncludeGlobalServiceEvents": true,
    "Name": "Trail1",
    "TrailARN": "arn:aws:cloudtrail:us-west-2:123456789012:trail/Trail1",
    "LogFileValidationEnabled": false,
    "IsMultiRegionTrail": true,
    "S3BucketName": "amzn-s3-demo-bucket"
}
```

- Para obter detalhes da API, consulte CreateTrailem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria uma trilha que usará o bucket “amzn-s3-demo-bucket” para armazenamento de arquivos de log.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

Exemplo 2: cria uma trilha que usará o bucket “amzn-s3-demo-bucket” para armazenamento de arquivos de log. Os objetos do S3 que representam os logs terão um prefixo de chave comum de “mylogs”. Quando novos registros forem entregues ao bucket, uma notificação será enviada para o tópico do SNS “mlog-deliverytopic”. Este exemplo usa o nivelamento para fornecer os valores dos parâmetros ao cmdlet.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- Para obter detalhes da API, consulte CreateTrailem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria uma trilha que usará o bucket “amzn-s3-demo-bucket” para armazenamento de arquivos de log.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

Exemplo 2: cria uma trilha que usará o bucket “amzn-s3-demo-bucket” para armazenamento de arquivos de log. Os objetos do S3 que representam os logs terão um prefixo de chave comum de “mylogs”. Quando novos registros forem entregues ao bucket, uma notificação será enviada para o tópico do SNS “mlog-deliverytopic”. Este exemplo usa o nivelamento para fornecer os valores dos parâmetros ao cmdlet.

```
$params = @{
    Name="awscloudtrail-example"
```

```
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- Para obter detalhes da API, consulte CreateTrailem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'
require 'aws-sdk-s3'
require 'aws-sdk-sts'

def create_trail_example(s3_client, sts_client, cloudtrail_client, trail_name,
 bucket_name)
  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to an Amazon Simple Storage Service (S3) bucket.
  s3_client.create_bucket(bucket: bucket_name)
  begin
    policy = {
      'Version' => '2012-10-17',
      'Statement' => [
        {
          'Sid' => 'AWSCloudTrailAclCheck20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:GetBucketAcl',
```

```
          'Resource' => "arn:aws:s3:::#{bucket_name}"
        },
        {
          'Sid' => 'AWSCloudTrailWrite20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:PutObject',
          'Resource' => "arn:aws:s3:::#{bucket_name}/AWSLogs/#{account_id}/*",
          'Condition' => {
            'StringEquals' => {
              's3:x-amz-acl' => 'bucket-owner-full-control'
            }
          }
        }
      ]
    }.to_json

    s3_client.put_bucket_policy(
      bucket: bucket_name,
      policy: policy
    )
    puts "Successfully added policy to bucket #{bucket_name}"
  end

  begin
    cloudtrail_client.create_trail({
                                     name: trail_name, # required
                                     s3_bucket_name: bucket_name # required
                                   })

    puts "Successfully created trail: #{trail_name}."
  rescue StandardError => e
    puts "Got error trying to create trail #{trail_name}:\n #{e}"
    puts e
    exit 1
  end
```

- Para obter detalhes da API, consulte CreateTrail a Referência AWS SDK para Ruby da API.

Use **`DeleteTrail`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteTrail`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Routine which deletes an AWS CloudTrail trail.
/*!
  \param trailName: The name of the CloudTrail trail.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::deleteTrail(const Aws::String trailName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);

    Aws::CloudTrail::Model::DeleteTrailRequest request;
    request.SetName(trailName);

    auto outcome = trailClient.DeleteTrail(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Error deleting trail " << trailName << " " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTraila Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como para excluir uma trilha

O comando `delete-trail` a seguir cria uma trilha chamada `Trail1`:

```
aws cloudtrail delete-trail --name Trail1
```

- Para obter detalhes da API, consulte DeleteTrailem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui a trilha especificada. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a confirmação, adicione o parâmetro -Force.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Para obter detalhes da API, consulte DeleteTrailem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui a trilha especificada. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a confirmação, adicione o parâmetro -Force.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Para obter detalhes da API, consulte DeleteTrailem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  client.delete_trail({
                        name: trail_name # required
                      })
  puts "Successfully deleted trail: #{trail_name}"
rescue StandardError => e
  puts "Got error trying to delete trail: #{trail_name}:"
  puts e
  exit 1
end
```

- Para obter detalhes da API, consulte DeleteTraila Referência AWS SDK para Ruby da API.

Use **`DescribeTrail`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeTrail`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Routine which describes the AWS CloudTrail trails in an account.
/*!
  \param clientConfig: Aws client configuration.
```

```
  \return bool: Function succeeded.
*/

bool AwsDoc::CloudTrail::describeTrails(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudTrailClient(clientConfig);
    Aws::CloudTrail::Model::DescribeTrailsRequest request;

    auto outcome = cloudTrailClient.DescribeTrails(request);
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::CloudTrail::Model::Trail> &trails =
 outcome.GetResult().GetTrailList();
        std::cout << trails.size() << " trail(s) found." << std::endl;
        for (const Aws::CloudTrail::Model::Trail &trail: trails) {
            std::cout << trail.GetName() << std::endl;
        }
    }
    else {
        std::cerr << "Failed to describe trails." <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeTraila Referência AWS SDK para C++ da API.

Usar **`DescribeTrails`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTrails`.

CLI

AWS CLI

Para descrever uma trilha

O exemplo de `describe-trails` a seguir retorna as configurações de `Trail1` e `Trail2`.

```
aws cloudtrail describe-trails \
    --trail-name-list Trail1 Trail2
```

Saída:

```
{
    "trailList": [
        {
            "IncludeGlobalServiceEvents": true,
            "Name": "Trail1",
            "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1",
            "LogFileValidationEnabled": false,
            "IsMultiRegionTrail": false,
            "S3BucketName": "amzn-s3-demo-bucket",
            "CloudWatchLogsRoleArn": "arn:aws:iam::123456789012:role/
CloudTrail_CloudWatchLogs_Role",
            "CloudWatchLogsLogGroupArn": "arn:aws:logs:us-
east-1:123456789012:log-group:CloudTrail:*",
            "SnsTopicName": "my-topic",
            "HomeRegion": "us-east-1"
        },
        {
            "IncludeGlobalServiceEvents": true,
            "Name": "Trail2",
            "S3KeyPrefix": "my-prefix",
            "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail2",
            "LogFileValidationEnabled": false,
            "IsMultiRegionTrail": false,
            "S3BucketName": "amzn-s3-demo-bucket2",
            "KmsKeyId": "arn:aws:kms:us-
east-1:123456789012:key/4c5ae5ac-3c13-421e-8335-c7868ef6a769",
            "HomeRegion": "us-east-1"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeTrailsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: recupera as configurações de todas as trilhas associadas à região atual de sua conta.

```
Get-CTTrail
```

Exemplo 2: retorna as configurações das trilhas especificadas.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Exemplo 3: retorna as configurações das trilhas especificadas que foram criadas em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt, eu-central-1).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- Para obter detalhes da API, consulte DescribeTrailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: recupera as configurações de todas as trilhas associadas à região atual de sua conta.

```
Get-CTTrail
```

Exemplo 2: retorna as configurações das trilhas especificadas.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Exemplo 3: retorna as configurações das trilhas especificadas que foram criadas em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt, eu-central-1).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- Para obter detalhes da API, consulte DescribeTrailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetTrailStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetTrailStatus`.

CLI

AWS CLI

Para obter o status de uma trilha

O comando `get-trail-status` a seguir retorna os detalhes de entrega e registro em log da `Trail1`:

```
aws cloudtrail get-trail-status --name Trail1
```

Saída:

```
{
  "LatestNotificationTime": 1454022144.869,
  "LatestNotificationAttemptSucceeded": "2016-01-28T23:02:24Z",
  "LatestDeliveryAttemptTime": "2016-01-28T23:02:24Z",
  "LatestDeliveryTime": 1454022144.869,
  "TimeLoggingStarted": "2015-11-06T18:36:38Z",
  "LatestDeliveryAttemptSucceeded": "2016-01-28T23:02:24Z",
  "IsLogging": true,
  "LatestCloudWatchLogsDeliveryTime": 1454022144.918,
  "StartLoggingTime": 1446834998.695,
  "StopLoggingTime": 1446834996.933,
  "LatestNotificationAttemptTime": "2016-01-28T23:02:24Z",
  "TimeLoggingStopped": "2015-11-06T18:36:36Z"
}
```

- Para obter detalhes da API, consulte GetTrailStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Retorna informações de status da trilha com o nome ExampleTrail 'meu'. Os dados retornados incluem informações sobre erros de envio, erros do Amazon SNS e do Amazon S3, além dos horários de início e término do registro em log da trilha. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Get-CTTrailStatus -Name myExampleTrail
```

Exemplo 2: retorna as informações de status de uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt, eu-central-1).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte GetTrailStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Retorna informações de status da trilha com o nome ExampleTrail 'meu'. Os dados retornados incluem informações sobre erros de envio, erros do Amazon SNS e do Amazon S3, além dos horários de início e término do registro em log da trilha. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Get-CTTrailStatus -Name myExampleTrail
```

Exemplo 2: retorna as informações de status de uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt, eu-central-1).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte GetTrailStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListTrails`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListTrails`.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'
```

```
def describe_trails_example(client)
  resp = client.describe_trails({})
  puts "Found #{resp.trail_list.count} trail(s)."

  resp.trail_list.each do |trail|
    puts "Name:           #{trail.name}"
    puts "S3 bucket name: #{trail.s3_bucket_name}"
    puts
  end
```

- Para obter detalhes da API, consulte ListTrailsa Referência AWS SDK para Ruby da API.

Use **`LookupEvents`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `LookupEvents`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Routine which looks up events captured by AWS CloudTrail.
/*!
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::lookupEvents(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudtrail(clientConfig);

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::CloudTrail::Model::Event> allEvents;

    Aws::CloudTrail::Model::LookupEventsRequest request;
```

```
    size_t count = 0;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::CloudTrail::Model::LookupEventsOutcome outcome =
 cloudtrail.LookupEvents(
                request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::CloudTrail::Model::Event> &events =
 outcome.GetResult().GetEvents();
            count += events.size();
            allEvents.insert(allEvents.end(), events.begin(), events.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!nextToken.empty() && count <= 50); // Limit to 50 events.

    std::cout << "Found " << allEvents.size() << " event(s)." << std::endl;

    for (auto &event: allEvents) {
        std::cout << "Event name: " << event.GetEventName() << std::endl;
        std::cout << "Event source: " << event.GetEventSource() << std::endl;
        std::cout << "Event id: " << event.GetEventId() << std::endl;
        std::cout << "Resources: " << std::endl;
        for (auto &resource: event.GetResources()) {
            std::cout << "  " << resource.GetResourceName() << std::endl;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte LookupEventsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como pesquisar eventos em uma trilha

O comando `lookup-events` a seguir pesquisa eventos de atividade da API pelo atributo `EventName`:

```
aws cloudtrail lookup-events --lookup-
attributes AttributeKey=EventName,AttributeValue=ConsoleLogin
```

Saída:

```
{
  "Events": [
     {
         "EventId": "654ccbc0-ba0d-486a-9076-dbf7274677a7",
         "Username": "my-session-name",
         "EventTime": "2021-11-18T09:41:02-08:00",
         "CloudTrailEvent": "{\"eventVersion\":\"1.02\",\"userIdentity
\":{\"type\":\"AssumedRole\",\"principalId\":\"AROAJIKPFTA72SWU4L7T4:my-
session-name\",\"arn\":\"arn:aws:sts::123456789012:assumed-role/my-
role/my-session-name\",\"accountId\":\"123456789012\",\"sessionContext
\":{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":
\"2016-01-26T21:42:12Z\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId
\":\"AROAJIKPFTA72SWU4L7T4\",\"arn\":\"arn:aws:iam::123456789012:role/my-
role\",\"accountId\":\"123456789012\",\"userName\":\"my-role\"}}},\"eventTime
\":\"2016-01-26T21:42:12Z\",\"eventSource\":\"signin.amazonaws.com\",
\"eventName\":\"ConsoleLogin\",\"awsRegion\":\"us-east-1\",\"sourceIPAddress\":
\"72.21.198.70\",\"userAgent\":\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36\",
\"requestParameters\":null,\"responseElements\":{\"ConsoleLogin\":\"Success\"},
\"additionalEventData\":{\"MobileVersion\":\"No\",\"MFAUsed\":\"No\"},\"eventID
\":\"654ccbc0-ba0d-486a-9076-dbf7274677a7\",\"eventType\":\"AwsConsoleSignIn\",
\"recipientAccountId\":\"123456789012\"}",
         "EventName": "ConsoleLogin",
         "Resources": []
     }
  ]
}
```

- Para obter detalhes da API, consulte LookupEventsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna todos os eventos que ocorreram nos últimos sete dias. Por padrão, o cmdlet faz automaticamente várias chamadas para entregar todos os eventos, saindo quando o serviço indica que não há mais dados disponíveis.

```
Find-CTEvent
```

Exemplo 2: retorna todos os eventos que ocorreram nos últimos sete dias, especificando uma região que não é o padrão atual do shell.

```
Find-CTEvent -Region eu-central-1
```

Exemplo 3: retorna todos os eventos associados à chamada da RunInstances API.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Exemplo 4: retorna os primeiros cinco eventos disponíveis.

```
Find-CTEvent -MaxResult 5
```

- Para obter detalhes da API, consulte LookupEventsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna todos os eventos que ocorreram nos últimos sete dias. Por padrão, o cmdlet faz automaticamente várias chamadas para entregar todos os eventos, saindo quando o serviço indica que não há mais dados disponíveis.

```
Find-CTEvent
```

Exemplo 2: retorna todos os eventos que ocorreram nos últimos sete dias, especificando uma região que não é o padrão atual do shell.

```
Find-CTEvent -Region eu-central-1
```

Exemplo 3: retorna todos os eventos associados à chamada da RunInstances API.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Exemplo 4: retorna os primeiros cinco eventos disponíveis.

```
Find-CTEvent -MaxResult 5
```

- Para obter detalhes da API, consulte LookupEventsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

# @param [Object] client
def lookup_events_example(client)
  resp = client.lookup_events
  puts "Found #{resp.events.count} events:"
  resp.events.each do |e|
    puts "Event name:   #{e.event_name}"
    puts "Event ID:     #{e.event_id}"
    puts "Event time:   #{e.event_time}"
    puts 'Resources:'

    e.resources.each do |r|
      puts "  Name:       #{r.resource_name}"
      puts "  Type:       #{r.resource_type}"
      puts ''
    end
  end
```

```
end
```

- Para obter detalhes da API, consulte LookupEventsa Referência AWS SDK para Ruby da API.

Usar **`StartLogging`** com uma CLI

Os exemplos de código a seguir mostram como usar o `StartLogging`.

CLI

AWS CLI

Para iniciar o registro de uma trilha

O comando `start-logging` a seguir ativa o registro em log da `Trail1`:

```
aws cloudtrail start-logging --name Trail1
```

- Para obter detalhes da API, consulte StartLoggingem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: inicia a gravação das chamadas de AWS API e a entrega do arquivo de log para a trilha chamada ExampleTrail 'my'. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Start-CTLogging -Name myExampleTrail
```

Exemplo 2: inicia a gravação das chamadas de AWS API e a entrega do arquivo de log para uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte StartLoggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: inicia a gravação das chamadas de AWS API e a entrega do arquivo de log para a trilha chamada ExampleTrail 'my'. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Start-CTLogging -Name myExampleTrail
```

Exemplo 2: inicia a gravação das chamadas de AWS API e a entrega do arquivo de log para uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte StartLoggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`StopLogging`** com uma CLI

Os exemplos de código a seguir mostram como usar o `StopLogging`.

CLI

AWS CLI

Como parar de registrar uma trilha

O comando `stop-logging` a seguir desativa o registro em log da `Trail1`:

```
aws cloudtrail stop-logging --name Trail1
```

- Para obter detalhes da API, consulte StopLoggingem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: suspende a gravação de chamadas de AWS API e a entrega do arquivo de log para a trilha chamada 'my'ExampleTrail. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Stop-CTLogging -Name myExampleTrail
```

Exemplo 2: suspende a gravação de chamadas de AWS API e a entrega de arquivos de log para uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte StopLoggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: suspende a gravação de chamadas de AWS API e a entrega do arquivo de log para a trilha chamada 'my'ExampleTrail. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Stop-CTLogging -Name myExampleTrail
```

Exemplo 2: suspende a gravação de chamadas de AWS API e a entrega de arquivos de log para uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes da API, consulte StopLoggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateTrail`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateTrail`.

CLI

AWS CLI

Para atualizar uma trilha

O exemplo de `update-trail` a seguir atualiza uma trilha para usar um bucket existente para entrega de logs.

```
aws cloudtrail update-trail \
    --name Trail1 \
    --s3-bucket-name amzn-s3-demo-bucket
```

Saída:

```
{
    "IncludeGlobalServiceEvents": true,
    "Name": "Trail1",
    "TrailARN": "arn:aws:cloudtrail:us-west-2:123456789012:trail/Trail1",
    "LogFileValidationEnabled": false,
    "IsMultiRegionTrail": true,
    "S3BucketName": "amzn-s3-demo-bucket"
}
```

- Para obter detalhes da API, consulte UpdateTrailem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: atualiza a trilha especificada para que os eventos globais do serviço (como os do IAM) sejam registrados e altera o prefixo de chave comum dos arquivos de log futuros para “globallogs”.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Exemplo 2: atualiza a trilha especificada para que as notificações sobre novas entregas de logs sejam enviadas ao tópico do SNS especificado.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Exemplo 3: atualiza a trilha especificada para que os logs sejam entregues em um bucket diferente.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Para obter detalhes da API, consulte UpdateTrailem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: atualiza a trilha especificada para que os eventos globais do serviço (como os do IAM) sejam registrados e altera o prefixo de chave comum dos arquivos de log futuros para “globallogs”.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Exemplo 2: atualiza a trilha especificada para que as notificações sobre novas entregas de logs sejam enviadas ao tópico do SNS especificado.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Exemplo 3: atualiza a trilha especificada para que os logs sejam entregues em um bucket diferente.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Para obter detalhes da API, consulte UpdateTrailem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para CloudWatch usar AWS SDKs

Os exemplos de código a seguir mostram como usar a Amazon CloudWatch com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- CloudWatch Guia do usuário — Mais informações sobre CloudWatch.

- CloudWatch Referência da API — Detalhes sobre todas as CloudWatch ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CloudWatch uso AWS SDKs
  - Olá CloudWatch
  - Aprenda as principais operações para CloudWatch usar um AWS SDK
  - Ações para CloudWatch usar AWS SDKs
    - Use DeleteAlarmscom um AWS SDK ou CLI
    - Use DeleteAnomalyDetectorcom um AWS SDK ou CLI
    - Use DeleteDashboardscom um AWS SDK ou CLI
    - Use DescribeAlarmHistorycom um AWS SDK ou CLI
    - Use DescribeAlarmscom um AWS SDK ou CLI
    - Use DescribeAlarmsForMetriccom um AWS SDK ou CLI
    - Use DescribeAnomalyDetectorscom um AWS SDK ou CLI
    - Use DisableAlarmActionscom um AWS SDK ou CLI
    - Use EnableAlarmActionscom um AWS SDK ou CLI
    - Use GetDashboardcom um AWS SDK ou CLI
    - Use GetMetricDatacom um AWS SDK ou CLI
    - Use GetMetricStatisticscom um AWS SDK ou CLI
    - Use GetMetricWidgetImagecom um AWS SDK ou CLI
    - Use ListDashboardscom um AWS SDK ou CLI
    - Use ListMetricscom um AWS SDK ou CLI
    - Use PutAnomalyDetectorcom um AWS SDK ou CLI
    - Use PutDashboardcom um AWS SDK ou CLI
    - Use PutMetricAlarmcom um AWS SDK ou CLI
    - Use PutMetricDatacom um AWS SDK ou CLI

- Cenários para CloudWatch usar AWS SDKs

- Como criar um painel de monitoramento com o nome da função como uma variável
- Comece a usar CloudWatch alarmes usando um AWS SDK
- Gerencie CloudWatch métricas e alarmes usando um AWS SDK
- Monitore o desempenho do Amazon DynamoDB usando um AWS SDK
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

Exemplos básicos de CloudWatch uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico da Amazon CloudWatch com AWS SDKs.

Exemplos

- Olá CloudWatch
- Aprenda as principais operações para CloudWatch usar um AWS SDK
- Ações para CloudWatch usar AWS SDKs
  - Use DeleteAlarmscom um AWS SDK ou CLI
  - Use DeleteAnomalyDetectorcom um AWS SDK ou CLI
  - Use DeleteDashboardscom um AWS SDK ou CLI
  - Use DescribeAlarmHistorycom um AWS SDK ou CLI
  - Use DescribeAlarmscom um AWS SDK ou CLI
  - Use DescribeAlarmsForMetriccom um AWS SDK ou CLI
  - Use DescribeAnomalyDetectorscom um AWS SDK ou CLI
  - Use DisableAlarmActionscom um AWS SDK ou CLI
  - Use EnableAlarmActionscom um AWS SDK ou CLI
  - Use GetDashboardcom um AWS SDK ou CLI
  - Use GetMetricDatacom um AWS SDK ou CLI
  - Use GetMetricStatisticscom um AWS SDK ou CLI
  - Use GetMetricWidgetImagecom um AWS SDK ou CLI
  - Use ListDashboardscom um AWS SDK ou CLI

- Use ListMetricscom um AWS SDK ou CLI
- Use PutAnomalyDetectorcom um AWS SDK ou CLI
- Use PutDashboardcom um AWS SDK ou CLI
- Use PutMetricAlarmcom um AWS SDK ou CLI
- Use PutMetricDatacom um AWS SDK ou CLI

Olá CloudWatch

O exemplo de código a seguir mostra como começar a usar o CloudWatch.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace CloudWatchActions;

public static class HelloCloudWatch
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon CloudWatch service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatch>()
            ).Build();
```

```
        // Now the client is available for injection.
        var cloudWatchClient =
host.Services.GetRequiredService<IAmazonCloudWatch>();

        // You can use await and any of the async methods to get a response.
        var metricNamespace = "AWS/Billing";
        var response = await cloudWatchClient.ListMetricsAsync(new
ListMetricsRequest
        {
            Namespace = metricNamespace
        });
        Console.WriteLine($"Hello Amazon CloudWatch! Following are some metrics
available in the {metricNamespace} namespace:");
        Console.WriteLine();
        if (response.Metrics != null)
        {
            foreach (var metric in response.Metrics.Take(5))
            {
                Console.WriteLine($"\tMetric: {metric.MetricName}");
                Console.WriteLine($"\tNamespace: {metric.Namespace}");
                Console.WriteLine(
                    $"\tDimensions: {string.Join(", ", metric.Dimensions.Select(m
=> $"{m.Name}:{m.Value}"))}");
                Console.WriteLine();
            }
        }
    }
}
```

- Para obter detalhes da API, consulte ListMetricsa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
```

```
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> System.out.println(" Retrieved metric is:
 " + metrics.metricName()));

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

- Para obter detalhes da API, consulte ListMetricsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
```

```
            <namespace>
        Where:
            namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListMetricsreferência da API AWS SDK for Kotlin.

Aprenda as principais operações para CloudWatch usar um AWS SDK

Os exemplos de código a seguir mostram como:

- Listar CloudWatch namespaces e métricas.
- Obter estatísticas para uma métrica e para faturamento estimado.
- Criar e atualizar um painel.

- Criar e adicionar dados a uma métrica.
- Criar e acionar um alarme e, em seguida, visualizar o histórico de alarmes.
- Criar um detector de anomalias.
- Obter uma imagem de métrica e, em seguida, limpar os recursos.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
public class CloudWatchScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
    https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

    This .NET example performs the following tasks:
        1. List and select a CloudWatch namespace.
        2. List and select a CloudWatch metric.
        3. Get statistics for a CloudWatch metric.
        4. Get estimated billing statistics for the last week.
        5. Create a new CloudWatch dashboard with two metrics.
        6. List current CloudWatch dashboards.
        7. Create a CloudWatch custom metric and add metric data.
        8. Add the custom metric to the dashboard.
        9. Create a CloudWatch alarm for the custom metric.
       10. Describe current CloudWatch alarms.
       11. Get recent data for the custom metric.
       12. Add data to the custom metric to trigger the alarm.
```

```
      13. Wait for an alarm state.
      14. Get history for the CloudWatch alarm.
      15. Add an anomaly detector.
      16. Describe current anomaly detectors.
      17. Get and display a metric image.
      18. Clean up resources.
   */

   private static ILogger logger = null!;
   private static CloudWatchWrapper _cloudWatchWrapper = null!;
   private static IConfiguration _configuration = null!;
   private static readonly List<string> _statTypes = new List<string>
{ "SampleCount", "Average", "Sum", "Minimum", "Maximum" };
   private static SingleMetricAnomalyDetector? anomalyDetector = null!;

   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
           services.AddAWSService<IAmazonCloudWatch>()
           .AddTransient<CloudWatchWrapper>()
       )
       .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();

       logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
           .CreateLogger<CloudWatchScenario>();

       _cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon CloudWatch example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            var selectedNamespace = await SelectNamespace();
            var selectedMetric = await SelectMetric(selectedNamespace);
            await GetAndDisplayMetricStatistics(selectedNamespace,
selectedMetric);
            await GetAndDisplayEstimatedBilling();
            await CreateDashboardWithMetrics();
            await ListDashboards();
            await CreateNewCustomMetric();
            await AddMetricToDashboard();
            await CreateMetricAlarm();
            await DescribeAlarms();
            await GetCustomMetricData();
            await AddMetricDataForAlarm();
            await CheckForMetricAlarm();
            await GetAlarmHistory();
            anomalyDetector = await AddAnomalyDetector();
            await DescribeAnomalyDetectors();
            await GetAndOpenMetricImage();
            await CleanupResources();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources();
        }

    }

    /// <summary>
    /// Select a namespace.
    /// </summary>
    /// <returns>The selected namespace.</returns>
    private static async Task<string> SelectNamespace()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. Select a CloudWatch Namespace from a list of
Namespaces.");
        var metrics = await _cloudWatchWrapper.ListMetrics();
```

```
        // Get a distinct list of namespaces.
        var namespaces = metrics.Select(m => m.Namespace).Distinct().ToList();
        for (int i = 0; i < namespaces.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {namespaces[i]}");
        }

        var namespaceChoiceNumber = 0;
        while (namespaceChoiceNumber < 1 || namespaceChoiceNumber >
namespaces.Count)
        {
            Console.WriteLine(
                "Select a namespace by entering a number from the preceding
list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out namespaceChoiceNumber);
        }

        var selectedNamespace = namespaces[namespaceChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedNamespace;
    }

    /// <summary>
    /// Select a metric from a namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <returns>The metric name.</returns>
    private static async Task<Metric> SelectMetric(string metricNamespace)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. Select a CloudWatch metric from a namespace.");

        var namespaceMetrics = await
_cloudWatchWrapper.ListMetrics(metricNamespace);

        for (int i = 0; i < namespaceMetrics.Count && i < 15; i++)
        {
            var dimensionsWithValues = namespaceMetrics[i].Dimensions
                .Where(d => !string.Equals("None", d.Value));
            Console.WriteLine($"\t{i + 1}. {namespaceMetrics[i].MetricName} " +
```

```
                                $"{string.Join(", :", dimensionsWithValues.Select(d
=> d.Value))}");
        }

        var metricChoiceNumber = 0;
        while (metricChoiceNumber < 1 || metricChoiceNumber >
namespaceMetrics.Count)
        {
            Console.WriteLine(
                "Select a metric by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out metricChoiceNumber);
        }

        var selectedMetric = namespaceMetrics[metricChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedMetric;
    }

    /// <summary>
    /// Get and display metric statistics for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayMetricStatistics(string
metricNamespace, Metric metric)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. Get CloudWatch metric statistics for the last
day.");

        for (int i = 0; i < _statTypes.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {_statTypes[i]}");
        }

        var statisticChoiceNumber = 0;
        while (statisticChoiceNumber < 1 || statisticChoiceNumber >
_statTypes.Count)
        {
            Console.WriteLine(
```

```
                "Select a metric statistic by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out statisticChoiceNumber);
        }

        var selectedStatistic = _statTypes[statisticChoiceNumber - 1];
        var statisticsList = new List<string> { selectedStatistic };

        var metricStatistics = await
_cloudWatchWrapper.GetMetricStatistics(metricNamespace, metric.MetricName,
statisticsList, metric.Dimensions, 1, 60);

        if (!metricStatistics.Any())
        {
            Console.WriteLine($"No {selectedStatistic} statistics found for
{metric} in namespace {metricNamespace}.");
        }

        metricStatistics = metricStatistics.OrderBy(s => s.Timestamp).ToList();
        for (int i = 0; i < metricStatistics.Count && i < 10; i++)
        {
            var metricStat = metricStatistics[i];
            var statValue =
metricStat.GetType().GetProperty(selectedStatistic)!.GetValue(metricStat, null);
            Console.WriteLine($"\t{i + 1}. Timestamp
{metricStatistics[i].Timestamp:G} {selectedStatistic}: {statValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get and display estimated billing statistics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayEstimatedBilling()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Get CloudWatch estimated billing for the last
week.");
```

```
        var billingStatistics = await SetupBillingStatistics();

        for (int i = 0; i < billingStatistics.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. Timestamp
{billingStatistics[i].Timestamp:G} : {billingStatistics[i].Maximum}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Create a dashboard with metrics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CreateDashboardWithMetrics()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Create a new CloudWatch dashboard with metrics.");
        var dashboardName = _configuration["dashboardName"];
```

```
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);
        var newDashboardString = JsonSerializer.Serialize(
            newDashboard,
            new JsonSerializerOptions
            {
                DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull
            });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} was created.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List dashboards.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDashboards()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. List the CloudWatch dashboards in the current
account.");

        var dashboards = await _cloudWatchWrapper.ListDashboards();

        for (int i = 0; i < dashboards.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {dashboards[i].DashboardName}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and add data for a new custom metric.
```

```
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateNewCustomMetric()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Create and add data for a new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var customData = await PutRandomMetricData(customMetricName,
customMetricNamespace);

        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
```

```
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
        return customData;
    }

    /// <summary>
    /// Add the custom metric to the dashboard.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricToDashboard()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Add the new custom metric to the dashboard.");

        var dashboardName = _configuration["dashboardName"];

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var validationMessages = await SetupDashboard(customMetricNamespace,
customMetricName, dashboardName);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} updated with metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
```

```
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);
```

```
        return validationMessages;
    }

    /// <summary>
    /// Create a CloudWatch alarm for the new metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Create a CloudWatch alarm for the new metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var alarmName = _configuration["exampleAlarmName"];
        var accountId = _configuration["accountId"];
        var region = _configuration["region"];
        var emailTopic = _configuration["emailTopic"];
        var alarmActions = new List<string>();

        if (GetYesNoResponse(
                $"\tAdd an email action for topic {emailTopic} to alarm
{alarmName}? (y/n)"))
        {
            _cloudWatchWrapper.AddEmailAlarmAction(accountId, region, emailTopic,
alarmActions);
        }

        await _cloudWatchWrapper.PutMetricEmailAlarm(
            "Example metric alarm",
            alarmName,
            ComparisonOperator.GreaterThanOrEqualToThreshold,
            customMetricName,
            customMetricNamespace,
            100,
            alarmActions);

        Console.WriteLine($"\tAlarm {alarmName} added for metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
```

```
    /// Describe Alarms.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAlarms()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Describe CloudWatch alarms in the current
account.");

        var alarms = await _cloudWatchWrapper.DescribeAlarms();
        alarms = alarms.OrderByDescending(a => a.StateUpdatedTimestamp).ToList();

        for (int i = 0; i < alarms.Count && i < 10; i++)
        {
            var alarm = alarms[i];
            Console.WriteLine($"\t{i + 1}. {alarm.AlarmName}");
            Console.WriteLine($"\tState: {alarm.StateValue} for
{alarm.MetricName} {alarm.ComparisonOperator} {alarm.Threshold}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the recent data for the metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetCustomMetricData()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. Get current data for new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var accountId = _configuration["accountId"];

        var query = new List<MetricDataQuery>
        {
            new MetricDataQuery
            {
                AccountId = accountId,
                Id = "m1",
                Label = "Custom Metric Data",
                MetricStat = new MetricStat
```

```
                {
                    Metric = new Metric
                    {
                        MetricName = customMetricName,
                        Namespace = customMetricNamespace,
                    },
                    Period = 1,
                    Stat = "Maximum"
                }
            }
        };

        var metricData = await _cloudWatchWrapper.GetMetricData(
            20,
            true,
            DateTime.UtcNow.AddMinutes(1),
            20,
            query);

        for (int i = 0; i < metricData.Count; i++)
        {
            if (metricData[i].Values != null)
            {
                for (int j = 0; j < metricData[i].Values.Count; j++)
                {
                    Console.WriteLine(
                        $"\tTimestamp {metricData[i].Timestamps[j]:G} Value:
{metricData[i].Values[j]}");
                }
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add metric data to trigger an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricDataForAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"12. Add metric data to the custom metric to trigger
an alarm.");
```

```
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var nowUtc = DateTime.UtcNow;
        List<MetricDatum> customData = new List<MetricDatum>
        {
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-2)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-1)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc
            }
        };
        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");
        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check for a metric alarm using the DescribeAlarmsForMetric action.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckForMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"13. Checking for an alarm state.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
```

```
        var customMetricName = _configuration["customMetricName"];
        var hasAlarm = false;
        var retries = 10;
        while (!hasAlarm && retries > 0)
        {
            var alarms = await
_cloudWatchWrapper.DescribeAlarmsForMetric(customMetricNamespace,
customMetricName);
            hasAlarm = alarms.Any(a => a.StateValue == StateValue.ALARM);
            retries--;
            Thread.Sleep(20000);
        }

        Console.WriteLine(hasAlarm
            ? $"\tAlarm state found for {customMetricName}."
            : $"\tNo Alarm state found for {customMetricName} after 10
retries.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get history for an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAlarmHistory()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"14. Get alarm history.");

        var exampleAlarmName = _configuration["exampleAlarmName"];

        var alarmHistory = await
_cloudWatchWrapper.DescribeAlarmHistory(exampleAlarmName, 2);

        for (int i = 0; i < alarmHistory.Count; i++)
        {
            var history = alarmHistory[i];
            Console.WriteLine($"\t{i + 1}. {history.HistorySummary}, time
{history.Timestamp:g}");
        }
        if (!alarmHistory.Any())
        {
```

```
            Console.WriteLine($"\tNo alarm history data found for
{exampleAlarmName}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an anomaly detector.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<SingleMetricAnomalyDetector> AddAnomalyDetector()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"15. Add an anomaly detector.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detector = new SingleMetricAnomalyDetector
        {
            MetricName = customMetricName,
            Namespace = customMetricNamespace,
            Stat = "Maximum"
        };
        await _cloudWatchWrapper.PutAnomalyDetector(detector);
        Console.WriteLine($"\tAdded anomaly detector for metric
{customMetricName}.");

        Console.WriteLine(new string('-', 80));
        return detector;
    }

    /// <summary>
    /// Describe anomaly detectors.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAnomalyDetectors()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"16. Describe anomaly detectors in the current
account.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
```

```
        var customMetricName = _configuration["customMetricName"];

        var detectors = await
_cloudWatchWrapper.DescribeAnomalyDetectors(customMetricNamespace,
customMetricName);

        for (int i = 0; i < detectors.Count; i++)
        {
            var detector = detectors[i];
            Console.WriteLine($"\t{i + 1}.
{detector.SingleMetricAnomalyDetector.MetricName}, state
{detector.StateValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Fetch and open a metrics image for a CloudWatch metric and namespace.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAndOpenMetricImage()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("17. Get a metric image from CloudWatch.");

        Console.WriteLine($"\tGetting Image data for custom metric.");
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var memoryStream = await
_cloudWatchWrapper.GetTimeSeriesMetricImage(customMetricNamespace,
customMetricName, "Maximum", 10);
        var file = _cloudWatchWrapper.SaveMetricImage(memoryStream,
"MetricImages");

        ProcessStartInfo info = new ProcessStartInfo();

        Console.WriteLine($"\tFile saved as {Path.GetFileName(file)}.");
        Console.WriteLine($"\tPress enter to open the image.");
        Console.ReadLine();
        info.FileName = Path.Combine("ms-photos://", file);
        info.UseShellExecute = true;
        info.CreateNoWindow = true;
```

```
        info.Verb = string.Empty;

        Process.Start(info);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up created resources.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"18. Clean up resources.");

        var dashboardName = _configuration["dashboardName"];
        if (GetYesNoResponse($"\tDelete dashboard {dashboardName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting dashboard.");
            var dashboardList = new List<string> { dashboardName };
            await _cloudWatchWrapper.DeleteDashboards(dashboardList);
        }

        var alarmName = _configuration["exampleAlarmName"];
        if (GetYesNoResponse($"\tDelete alarm {alarmName}? (y/n)"))
        {
            Console.WriteLine($"\tCleaning up alarms.");
            var alarms = new List<string> { alarmName };
            await _cloudWatchWrapper.DeleteAlarms(alarms);
        }

        if (GetYesNoResponse($"\tDelete anomaly detector? (y/n)") &&
anomalyDetector != null)
        {
            Console.WriteLine($"\tCleaning up anomaly detector.");

            await _cloudWatchWrapper.DeleteAnomalyDetector(
                anomalyDetector);
        }

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Métodos de embalagem usados pelo cenário para CloudWatch ações.

```
/// <summary>
/// Wrapper class for Amazon CloudWatch methods.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;
    private readonly ILogger<CloudWatchWrapper> _logger;

    /// <summary>
    /// Constructor for the CloudWatch wrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch,
 ILogger<CloudWatchWrapper> logger)

    {
        _logger = logger;
        _amazonCloudWatch = amazonCloudWatch;
    }
```

```
    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
int days, int period)
    {
```

```
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints ?? new List<Datapoint>();
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages ?? new
 List<DashboardValidationMessage>();
    }


    /// <summary>
    /// Get information on a dashboard.
    /// </summary>
```

```
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A JSON object with dashboard information.</returns>
    public async Task<string> GetDashboard(string dashboardName)
    {
        var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
            new GetDashboardRequest()
            {
                DashboardName = dashboardName
            });

        return dashboardResponse.DashboardBody;
    }


    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
```

```
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
```

```
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }

    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        // Add the plus sign for positive offsets.
        timeZoneString = timeZoneString.StartsWith('-') ? timeZoneString : "+" +
 timeZoneString;
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
```

```
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        if (paginatedMetricData.MetricDataResults != null)
        {
            await foreach (var data in paginatedMetricData.MetricDataResults)
            {
                metricData.Add(data);
            }
        }

        return metricData;
    }

    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
    {
        try
        {
            var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
                new PutMetricAlarmRequest()
                {
```

```
                    AlarmActions = alarmActions,
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
                    ComparisonOperator = comparison,
                    Threshold = threshold,
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
 has already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
 alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }
```

```
    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }

    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms ?? new List<MetricAlarm>();
    }

    /// <summary>
```

```
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }

    /// <summary>
    /// Delete a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAlarms(List<string> alarmNames)
    {
        var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
            new DeleteAlarmsRequest()
            {
                AlarmNames = alarmNames
            });

        return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
```

```
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
```

```
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }

    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
```

```
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
 _amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Exemplos de valores de settings.json para o cenário.

```
{
  "dashboardName": "example-new-dashboard",
  "exampleAlarmName": "example-metric-alarm",
  "accountId": "1234567890",
  "region": "us-east-1",
  "emailTopic": "Default_CloudWatch_Alarms_Topic",
  "customMetricNamespace": "example-namespace",
  "customMetricName": "example-custom-metric",
  "dashboardExampleBody": {
    "widgets": [
      {
        "height": 6,
        "width": 6,
        "y": 0,
        "x": 0,
```

```
        "type": "text",
        "properties": {
          "markdown": "# Code Example Dashboard \nThis dashboard was created by
example code.\n"
        }
      },
      {
        "height": 8,
        "width": 8,
        "y": 0,
        "x": 6,
        "type": "metric",
        "properties": {
          "metrics": [
            [
              "AWS/Billing",
              "EstimatedCharges",
              "Currency",
              "USD",
              { "region": "us-east-1" }
            ]
          ],
          "view": "timeSeries",
          "region": "us-east-1",
          "stat": "Maximum",
          "period": 86400,
          "yAxis": {
            "left": {
              "min": 0,
              "max": 100
            }
          },
          "stacked": false,
          "title": "Estimated Billing",
          "setPeriodToTimeRange": false,
          "liveData": true,
          "sparkline": true,
          "trend": true
        }
      },
      {
        "height": 8,
        "width": 8,
        "y": 0,
```

```
        "x": 14,
        "type": "metric",
        "properties": {
          "metrics": [
            [ "AWS/Usage", "CallCount", "Type", "API", "Resource", "ListMetrics",
 "Service", "CloudWatch", "Class", "None" ],
            [ "...", "GetMetricStatistics", ".", ".", ".", "." ],
            [ "...", "GetMetricData", ".", ".", ".", "." ],
            [ "...", "PutDashboard", ".", ".", ".", "." ],
            [ "...", "PutMetricData", ".", ".", ".", "." ]
          ],
          "view": "timeSeries",
          "yAxis": {
            "left": {
              "min": 0,
              "max": 200
            }
          },
          "stacked": false,
          "region": "us-east-1",
          "stat": "Sum",
          "period": 300,
          "title": "CloudWatch Usage",
          "setPeriodToTimeRange": false,
          "liveData": true,
          "sparkline": true,
          "trend": true
        }
      }
    ]
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory

- DescribeAlarms
- DescribeAlarmsForMetric
- DescribeAnomalyDetectors
- GetMetricData
- GetMetricStatistics
- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando CloudWatch recursos.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardInvalidInputErrorException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorResponse;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.LimitExceededException;
```

```
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");

   static CloudWatchActions cwActions = new CloudWatchActions();

   private static final Logger logger =
LoggerFactory.getLogger(CloudWatchScenario.class);
   static Scanner scanner = new Scanner(System.in);
   public static void main(String[] args) throws Throwable {

       final String usage = """

           Usage:
             <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

           Where:
             myDate - The start date to use to get metric statistics. (For
example, 2023-01-11T18:35:24.00Z.)\s
             costDateWeek - The start date to use to get AWS/Billing statistics.
(For example, 2023-01-11T18:35:24.00Z.)\s
             dashboardName - The name of the dashboard to create.\s
             dashboardJson - The location of a JSON file to use to create a
dashboard. (See jsonWidgets.json in javav2/example_code/cloudwatch.)\s
             dashboardAdd - The location of a JSON file to use to update a
dashboard. (See CloudDashboard.json in javav2/example_code/cloudwatch.)\s
             settings - The location of a JSON file from which various values
are read. (See settings.json in javav2/example_code/cloudwatch.)\s
             metricImage - The location of a BMP file that is used to create a
graph.\s
           """;

       if (args.length != 7) {
           logger.info(usage);
           return;
       }
       String myDate = args[0];
       String costDateWeek = args[1];
       String dashboardName = args[2];
       String dashboardJson = args[3];
       String dashboardAdd = args[4];
       String settings = args[5];
       String metricImage = args[6];
```

```
        logger.info(DASHES);
        logger.info("Welcome to the Amazon CloudWatch Basics scenario.");
        logger.info("""
            Amazon CloudWatch is a comprehensive monitoring and observability
service
            provided by Amazon Web Services (AWS). It is designed to help you
monitor your
            AWS resources, applications, and services, as well as on-premises
resources,
            in real-time.

            CloudWatch collects and tracks various types of data, including
metrics,
            logs, and events, from your AWS and on-premises resources. It allows
you to set
            alarms and automatically respond to changes in your environment,
            enabling you to quickly identify and address issues before they
impact your
            applications or services.

            With CloudWatch, you can gain visibility into your entire
infrastructure, from the cloud
            to the edge, and use this information to make informed decisions and
optimize your
            resource utilization.

            This scenario guides you through how to perform Amazon CloudWatch
tasks by using the
            AWS SDK for Java v2. Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
            runScenario(myDate, costDateWeek, dashboardName, dashboardJson,
dashboardAdd, settings, metricImage);
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
        logger.info(DASHES);
    }

    private static void runScenario(String myDate, String costDateWeek, String
dashboardName, String dashboardJson, String dashboardAdd, String settings,
String metricImage ) throws Throwable {
```

```
        Double dataPoint = Double.parseDouble("10.0");
        logger.info(DASHES);
        logger.info("""
        1. List at least five available unique namespaces from Amazon
CloudWatch.
        Select one from the list.
        """);
        String selectedNamespace;
        String selectedMetrics;
        int num;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listNameSpacesAsync();
            ArrayList<String> list = future.join();
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". {}", list.get(z));
            }

            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedNamespace = list.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedNamespace);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List available metrics within the selected namespace.");
        logger.info("""
```

```
            A metric is a measure of the performance or health of your AWS
resources,
            applications, or custom resources. Metrics are the basic building
blocks of CloudWatch
            and provide data points that represent a specific aspect of your
system or application over time.

            Select a metric from the list.
            """);

        Dimension myDimension = null;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listMetsAsync(selectedNamespace);
            ArrayList<String> metList = future.join();
            logger.info("Metrics successfully retrieved. Total metrics: {}",
metList.size());
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". " + metList.get(z));
            }
            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedMetrics = metList.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedMetrics);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            myDimension =
cwActions.getSpecificMetAsync(selectedNamespace).join();
```

```
            logger.info("Metric statistics successfully retrieved and
displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Get statistics for the selected metric over the last
day.");
        logger.info("""
            Statistics refer to the various mathematical calculations that can be
performed on the
            collected metrics to derive meaningful insights. Statistics provide a
way to summarize and
            analyze the data collected for a specific metric over a specified
time period.
            """);
        waitForInputToContinue(scanner);
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            logger.info("    " + (t + 1) + ". {}", statTypes.get(t));
        }
        logger.info("Select a metric statistic by entering a number from the
preceding list:");
        num = Integer.parseInt(scanner.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
```

```
        } else {
            logger.info("You did not select a valid option.");
            return;
        }
        logger.info("You selected " + metricOption);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getAndDisplayMetricStatisticsAsync(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
            future.join();
            logger.info("Metric statistics retrieved successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Get CloudWatch estimated billing for the last week.");
        waitForInputToContinue(scanner);
         try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getMetricStatisticsAsync(costDateWeek);
            future.join();

            logger.info("Metric statistics successfully retrieved and
displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
```

```
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Create a new CloudWatch dashboard with metrics.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.createDashboardWithMetricsAsync(dashboardName, dashboardJson);
            future.join();

        } catch (RuntimeException | IOException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List dashboards using a paginator.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.listDashboardsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create a new custom metric by adding data to it.");
        logger.info("""
            The primary benefit of using a custom metric in Amazon CloudWatch is
the ability to
            monitor and collect data that is specific to your application or
infrastructure.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.createNewCustomMetricAsync(dataPoint);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Add an additional metric to the dashboard.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.addMetricToDashboardAsync(dashboardAdd, dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
```

```
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Create an alarm for the custom metric.");
        waitForInputToContinue(scanner);
        String alarmName = "" ;
        try {
            CompletableFuture<String> future =
cwActions.createAlarmAsync(settings);
            alarmName = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof LimitExceededException cwEx) {
                logger.info("The quota for alarms has been
reached: Error message: {}, Error code {}", cwEx.getMessage(),
cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Describe ten current alarms.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.describeAlarmsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Get current data for new custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.getCustomMetricDataAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Push data into the custom metric to trigger the
alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.addMetricDataForAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.checkForMetricAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Get alarm history for the new alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.getAlarmHistoryAsync(settings, myDate);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("15. Add an anomaly detector for the custom metric.");
        logger.info("""
            An anomaly detector is a feature that automatically detects unusual
patterns or deviations in your
            monitored metrics. It uses machine learning algorithms to analyze the
historical behavior
            of your metrics and establish a baseline.

            The anomaly detector then compares the current metric values against
this baseline and
            identifies any anomalies or outliers that may indicate potential
issues or unexpected changes
            in your system's performance or behavior.

            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.addAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Describe current anomaly detectors.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.describeAnomalyDetectorsAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Get a metric image for the custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.downloadAndSaveMetricImageAsync(metricImage);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("18. Clean up the Amazon CloudWatch resources.");

        try {
            logger.info(". Delete the Dashboard.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteDashboardsResponse> future =
cwActions.deleteDashboardAsync(dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
```

```
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the alarm.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAlarmsResponse> future =
cwActions.deleteCWAlarmAsync(alarmName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the anomaly detector.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAnomalyDetectorResponse> future =
cwActions.deleteAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("The Amazon CloudWatch example scenario is complete.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de invólucro para métodos do CloudWatch SDK.

```
public class CloudWatchActions {

    private static CloudWatchAsyncClient cloudWatchAsyncClient;

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchActions.class);

    /**
     * Retrieves an asynchronous CloudWatch client instance.
     *
     * <p>
     * This method ensures that the CloudWatch client is initialized with the
 following configurations:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
```

```
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
     * </p>
     *
     * @return the asynchronous CloudWatch client instance
     */
    private static CloudWatchAsyncClient getAsyncClient() {
        if (cloudWatchAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            cloudWatchAsyncClient = CloudWatchAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return cloudWatchAsyncClient;
    }

    /**
     * Deletes an Anomaly Detector.
     *
     * @param fileName the name of the file containing the Anomaly Detector
configuration
     * @return a CompletableFuture that represents the asynchronous deletion of
the Anomaly Detector
     */
    public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
```

```
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root
node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly
Detector", exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }

    /**
     * Deletes a CloudWatch alarm.
```

```
    *
    * @param alarmName the name of the alarm to be deleted
    * @return a {@link CompletableFuture} representing the asynchronous
operation to delete the alarm
    * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
    * or a {@link RuntimeException} is thrown if the operation fails
    */
   public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
       DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
           .alarmNames(alarmName)
           .build();

       return getAsyncClient().deleteAlarms(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
               } else {
                   logger.info("Successfully deleted alarm {} ", alarmName);
               }
           });
   }

   /**
    * Deletes the specified dashboard.
    *
    * @param dashboardName the name of the dashboard to be deleted
    * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the dashboard
    * @throws RuntimeException if the dashboard deletion fails
    */
   public CompletableFuture<DeleteDashboardsResponse>
deleteDashboardAsync(String dashboardName) {
       DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
           .dashboardNames(dashboardName)
           .build();

       return getAsyncClient().deleteDashboards(dashboardsRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
```

```
                    throw new RuntimeException("Failed to delete the dashboard: "
+ dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }


    /**
     * Retrieves and saves a custom metric image to a file.
     *
     * @param fileName the name of the file to save the metric image to
     * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
     */
    public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String
fileName) {
        logger.info("Getting Image data for custom metric.");
        String myJSON = """
              {
                  "title": "Example Metric Graph",
                  "view": "timeSeries",
                  "stacked ": false,
                  "period": 10,
                  "width": 1400,
                  "height": 600,
                  "metrics": [
                      [
                      "AWS/Billing",
                      "EstimatedCharges",
                      "Currency",
                      "USD"
                     ]
                  ]
              }
            """;

        GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

        return getAsyncClient().getMetricWidgetImage(imageRequest)
```

```
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to
file", e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }


    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
     * @return a {@link CompletableFuture} that completes when the anomaly
detectors have been described
     * @throws RuntimeException if there is a failure during the operation, such
as when reading or parsing the JSON file,
     *                          or when describing the anomaly detectors
     */
    public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName)
{
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
```

```
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser);
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response
-> {
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly
detectors", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
```

```
    }


    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly
detector has been added
     */
    public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root
node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

                PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();
```

```
                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response
-> {
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }


    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });
```

```
        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response ->
response.alarmHistoryItems().forEach(item -> {
                        logger.info("History summary: {}",
item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
```

```
        });
    }



    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for
further processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);
```

```
            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified
metric.
     *
     * @param metricRequest    the request to describe the alarms for the
specified metric
     * @param customMetricName the name of the custom metric to check for an
alarm
     * @param retries          the number of retries to perform if no alarm is
found
     * @return a {@link CompletableFuture} that completes when an alarm is found
or the maximum number of retries has been reached
     */
    private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
        if (retries == 0) {
            return CompletableFuture.completedFuture(null).thenRun(() ->
                logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
            );
        }

        return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response ->
{
            if (response.hasMetricAlarms()) {
                logger.info("Alarm state found for {}", customMetricName);
                return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
```

```
            } else {
                return CompletableFuture.runAsync(() -> {
                    try {
                        Thread.sleep(20000);
                        logger.info(".");
                    } catch (InterruptedException e) {
                        throw new RuntimeException("Interrupted while waiting to
retry", e);
                    }
                }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
            }
        }));
    }


    /**
     * Adds metric data for an alarm asynchronously.
     *
     * @param fileName the name of the JSON file containing the metric data
     * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
     */
    public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for
further processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
```

```
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

                MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed
to parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
```

```
                logger.error("Failed to put metric data: " +
exception.getMessage(), exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }


    /**
     * Retrieves custom metric data from the AWS CloudWatch service.
     *
     * @param fileName the name of the file containing the custom metric
information
     * @return a {@link CompletableFuture} that completes when the metric data
has been retrieved
     */
    public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
        CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                // Read values from the JSON file.
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for
further processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
```

```
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
```

```
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}",
item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }


    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ",
alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}",
ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }
```

```
   /**
    * Creates an alarm based on the configuration provided in a JSON file.
    *
    * @param fileName the name of the JSON file containing the alarm
configuration
    * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
    * @throws RuntimeException if an exception occurs while reading the JSON
file or creating the alarm
    */
   public CompletableFuture<String> createAlarmAsync(String fileName) {
       com.fasterxml.jackson.databind.JsonNode rootNode;
       try {
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           rootNode = new ObjectMapper().readTree(parser);
       } catch (IOException e) {
           throw new RuntimeException("Failed to read the alarm configuration
file", e);
       }

       // Extract values from the JSON node.
       String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
       String customMetricName =
rootNode.findValue("customMetricName").asText();
       String alarmName = rootNode.findValue("exampleAlarmName").asText();
       String emailTopic = rootNode.findValue("emailTopic").asText();
       String accountId = rootNode.findValue("accountId").asText();
       String region = rootNode.findValue("region").asText();

       // Create a List for alarm actions.
       List<String> alarmActions = new ArrayList<>();
       alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

       PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
           .alarmActions(alarmActions)
           .alarmDescription("Example metric alarm")
           .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
           .threshold(100.00)
```

```
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }

    /**
     * Adds a metric to a dashboard asynchronously.
     *
     * @param fileName      the name of the file containing the dashboard content
     * @param dashboardName the name of the dashboard to be updated
     * @return a {@link CompletableFuture} representing the asynchronous
operation, which will complete with a
     * {@link PutDashboardResponse} when the dashboard is successfully updated
     */
    public CompletableFuture<PutDashboardResponse>
addMetricToDashboardAsync(String fileName, String dashboardName) {
        String dashboardBody;
        try {
            dashboardBody = readFileAsString(fileName);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the dashboard file", e);
        }

        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(dashboardBody)
            .build();
```

```
        return getAsyncClient().putDashboard(dashboardRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to update dashboard: {}",
ex.getMessage());
                    throw new RuntimeException("Error updating dashboard", ex);
                } else {
                    logger.info("{} was successfully updated.", dashboardName);
                    return response;
                }
            });
    }

    /**
     * Creates a new custom metric.
     *
     * @param dataPoint the data point to be added to the custom metric
     * @return a {@link CompletableFuture} representing the asynchronous
operation of adding the custom metric
     */
    public CompletableFuture<PutMetricDataResponse>
createNewCustomMetricAsync(Double dataPoint) {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object for the current time in UTC.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        // Create the MetricDatum.
        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
```

```
            .metricData(datum)
            .build();

        return getAsyncClient().putMetricData(request)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    throw new RuntimeException("Error adding custom metric", ex);
                } else {
                    logger.info("Successfully added metric values for
PAGES_VISITED.");
                }
            });
    }

    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing
the dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
    }


    /**
     * Creates a new dashboard with the specified name and metrics from the given
file.
     *
```

```
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous
operation of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from
the file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}",
ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                   if (messages.isEmpty()) {
                       logger.info("There are no messages in the new
Dashboard.");
                   } else {
                       for (DashboardValidationMessage message : messages) {
                           logger.info("Message: {}", message.message());
                       }
                   }
                   return response; // Return the response for further use
               }
           });
   }


   /**
    * Retrieves the metric statistics for the "EstimatedCharges" metric in the
"AWS/Billing" namespace.
```

```
    *
    * @param costDateWeek the start date for the metric statistics, in the
format of an ISO-8601 date string (e.g., "2023-04-05")
    * @return a {@link CompletableFuture} that, when completed, contains the
{@link GetMetricStatisticsResponse} with the retrieved metric statistics
    * @throws RuntimeException if the metric statistics cannot be retrieved
successfully
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getMetricStatisticsAsync(String costDateWeek) {
       Instant start = Instant.parse(costDateWeek);
       Instant endDate = Instant.now();

       // Define dimension
       Dimension dimension = Dimension.builder()
           .name("Currency")
           .value("USD")
           .build();

       List<Dimension> dimensionList = new ArrayList<>();
       dimensionList.add(dimension);

       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .metricName("EstimatedCharges")
           .namespace("AWS/Billing")
           .dimensions(dimensionList)
           .statistics(Statistic.MAXIMUM)
           .startTime(start)
           .endTime(endDate)
           .period(86400) // One day period
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {})",
datapoint.timestamp(), datapoint.maximum());
                       }
                   } else {
                       logger.info("The returned data list is empty");
```

```
                    }
                } else {
                    throw new RuntimeException("Failed to get metric statistics:
" + exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
     * @param metVal       the name of the metric
     * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
     * @param date         the date for which to retrieve the metric statistics,
in the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
     * @param myDimension  the dimension(s) to filter the metric statistics by
     * @return a {@link CompletableFuture} that completes when the metric
statistics have been retrieved and displayed
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

            String metricOption, String date, Dimension myDimension) {

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        // Building the request for metric statistics.
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400) // 1 day period
            .statistics(Statistic.fromValue(metricOption))
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
```

```
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric
statistics: " + exception.getMessage(), exception);
            });
    }


    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an
{@link ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
```

```
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists the available namespaces for the current AWS account.
     *
     * @return a {@link CompletableFuture} that, when completed, contains an
{@link ArrayList} of the available namespace names.
     * @throws RuntimeException if an error occurs while listing the namespaces.
     */
    public CompletableFuture<ArrayList<String>> listNameSpacesAsync() {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder().build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String namespace = metric.namespace();
                if (!nameSpaceList.contains(namespace)) {
                    nameSpaceList.add(namespace);
                }
            });
        });

        return future
            .thenApply(ignored -> nameSpaceList)
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list namespaces: " +
exception.getMessage(), exception);
            });
    }
    /**
     * Retrieves the specific metric asynchronously.
     *
     * @param namespace the namespace of the metric to retrieve
```

```
     * @return a CompletableFuture that completes with the first dimension of the
 first metric found in the specified namespace,
     * or throws a RuntimeException if an error occurs or no metrics or
 dimensions are found
     */
    public CompletableFuture<Dimension> getSpecificMetAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        return getAsyncClient().listMetrics(request).handle((response, exception)
 -> {
            if (exception != null) {
                logger.info("Error occurred while listing metrics: {} ",
 exception.getMessage());
                throw new RuntimeException("Failed to retrieve specific metric
 dimension", exception);
            } else {
                List<Metric> myList = response.metrics();
                if (!myList.isEmpty()) {
                    Metric metric = myList.get(0);
                    if (!metric.dimensions().isEmpty()) {
                        return metric.dimensions().get(0); // Return the first
 dimension
                    }
                }
                throw new RuntimeException("No metrics or dimensions found");
            }
        });
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards

- DescribeAlarmHistory
- DescribeAlarms
- DescribeAlarmsForMetric
- DescribeAnomalyDetectors
- GetMetricData
- GetMetricStatistics
- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando CloudWatch recursos.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
```

```
 This Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 2. List available metrics within the selected namespace.
 3. Get statistics for the selected metric over the last day.
 4. Get CloudWatch estimated billing for the last week.
 5. Create a new CloudWatch dashboard with metrics.
 6. List dashboards using a paginator.
 7. Create a new custom metric by adding data for it.
 8. Add the custom metric to the dashboard.
 9. Create an alarm for the custom metric.
 10. Describe current alarms.
 11. Get current data for the new custom metric.
 12. Push data into the custom metric to trigger the alarm.
 13. Check the alarm state using the action DescribeAlarmsForMetric.
 14. Get alarm history for the new alarm.
 15. Add an anomaly detector for the custom metric.
 16. Describe current anomaly detectors.
 17. Get a metric image for the custom metric.
 18. Clean up the Amazon CloudWatch resources.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a
 dashboard. (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
```

```
            metricImage - The location of a BMP file that is used to create a
graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)
```

```
    println(DASHES)
    println("2. List available metrics within the selected namespace and select
one from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList!![num - 1]
    } else {
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
```

```
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics,
 metricOption, myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)

    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
```

```
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action
 DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}
```

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
```

```
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
```

```
}

suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
```

```
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10
 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
```

```
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
```

```
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}

suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest =
        PutMetricAlarmRequest {
```

```
            alarmActions = alarmActionObs
            alarmDescription = "Example metric alarm"
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
            threshold = 100.00
            metricName = customMetricName
            namespace = customMetricNamespace
            evaluationPeriods = 1
            period = 10
            statistic = Statistic.Maximum
            datapointsToAlarm = 1
            treatMissingData = "ignore"
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}

suspend fun addMetricToDashboard(
    fileNameVal: String,
    dashboardNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension =
        Dimension {
            name = "UNIQUE_PAGES"
            value = "URLS"
        }
```

```
    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = "PAGES_VISITED"
            unit = StandardUnit.None
            value = dataPoint
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
            dimensions = listOf(dimension)
        }

    val request =
        PutMetricDataRequest {
            namespace = "SITE/TRAFFIC"
            metricData = listOf(datum)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
```

```
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}

fun readFileAsString(file: String): String =
 String(Files.readAllBytes(Paths.get(file)))

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension =
        Dimension {
            name = "Currency"
            value = "USD"
        }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest =
        GetMetricStatisticsRequest {
            metricName = "EstimatedCharges"
            namespace = "AWS/Billing"
            dimensions = dimensionList
            statistics = listOf(Statistic.Maximum)
            startTime =
                aws.smithy.kotlin.runtime.time
```

```
                    .Instant(start)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            period = 86400
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
```

```
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
```

```
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard

- PutMetricAlarm
- PutMetricData

Ações para CloudWatch usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar CloudWatch ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a CloudWatch API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para CloudWatch usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon CloudWatch API Reference.

Exemplos

- Use DeleteAlarmscom um AWS SDK ou CLI
- Use DeleteAnomalyDetectorcom um AWS SDK ou CLI
- Use DeleteDashboardscom um AWS SDK ou CLI
- Use DescribeAlarmHistorycom um AWS SDK ou CLI
- Use DescribeAlarmscom um AWS SDK ou CLI
- Use DescribeAlarmsForMetriccom um AWS SDK ou CLI
- Use DescribeAnomalyDetectorscom um AWS SDK ou CLI
- Use DisableAlarmActionscom um AWS SDK ou CLI
- Use EnableAlarmActionscom um AWS SDK ou CLI
- Use GetDashboardcom um AWS SDK ou CLI
- Use GetMetricDatacom um AWS SDK ou CLI
- Use GetMetricStatisticscom um AWS SDK ou CLI
- Use GetMetricWidgetImagecom um AWS SDK ou CLI
- Use ListDashboardscom um AWS SDK ou CLI
- Use ListMetricscom um AWS SDK ou CLI
- Use PutAnomalyDetectorcom um AWS SDK ou CLI
- Use PutDashboardcom um AWS SDK ou CLI

- Use PutMetricAlarmcom um AWS SDK ou CLI
- Use PutMetricDatacom um AWS SDK ou CLI

Use **`DeleteAlarms`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteAlarms`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Começar a usar alarmes
- Gerencie métricas e alarmes
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete a list of alarms from CloudWatch.
/// </summary>
/// <param name="alarmNames">A list of names of alarms to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteAlarms(List<string> alarmNames)
{
    var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
        new DeleteAlarmsRequest()
        {
            AlarmNames = alarmNames
        });

    return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- Para obter detalhes da API, consulte DeleteAlarmsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

Excluir o alarme.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DeleteAlarmsRequest request;
        request.AddAlarmNames(alarm_name);

        auto outcome = cw.DeleteAlarms(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- Para obter detalhes da API, consulte DeleteAlarmsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir um alarme

O exemplo a seguir usa o `delete-alarms` comando para excluir o CloudWatch alarme da Amazon chamado “myalarm”:

```
aws cloudwatch delete-alarms --alarm-names myalarm
```

Saída:

```
This command returns to the prompt if successful.
```

- Para obter detalhes da API, consulte DeleteAlarmsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
```

```
     * or a {@link RuntimeException} is thrown if the operation fails
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
 alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
 alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteAlarmsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
```

```
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAlarmsa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });
```

```
var params = {
  AlarmNames: ["Web_Server_CPU_Utilization"],
};

cw.deleteAlarms(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAlarmsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAlarmsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
 metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
 metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
```

```
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteAlarmsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_cwt->deletealarms(
      it_alarmnames = it_alarm_names ).
    MESSAGE 'Alarms deleted.' TYPE 'I'.
  CATCH /aws1/cx_cwtresourcenotfound.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAlarmsreferência da API AWS SDK for SAP ABAP.

Use **`DeleteAnomalyDetector`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteAnomalyDetector`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteAnomalyDetectora Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para excluir um modelo específico de detecção de anomalias

O exemplo `delete-anomaly-detector` a seguir exclui um modelo de detector de anomalias na conta específica.

```
aws cloudwatch delete-anomaly-detector \
    --namespace AWS/Logs \
    --metric-name IncomingBytes \
    --stat SampleCount
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir um modelo de detecção de anomalias no Guia CloudWatch do usuário da Amazon.

- Para obter detalhes da API, consulte DeleteAnomalyDetectorem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes an Anomaly Detector.
    *
    * @param fileName the name of the file containing the Anomaly Detector
configuration
    * @return a CompletableFuture that represents the asynchronous deletion of
the Anomaly Detector
    */
   public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root
node
           } catch (IOException e) {
```

```
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly
Detector", exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }
```

- Para obter detalhes da API, consulte DeleteAnomalyDetectora Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAnomalyDetectorreferência da API AWS SDK for Kotlin.

Use **`DeleteDashboards`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteDashboards`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
_amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteDashboardsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para excluir painéis específicos

O exemplo `delete-dashboards` a seguir exclui dois painéis denominados `Dashboard-A` e `Dashboard-B` na conta específica.

```
aws cloudwatch delete-dashboards \
    --dashboard-names Dashboard-A Dashboard-B
```

Este comando não produz saída.

Para obter mais informações, consulte os CloudWatch painéis da Amazon no Guia do CloudWatch usuário da Amazon.

- Para obter detalhes da API, consulte DeleteDashboardsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
```

```
    public CompletableFuture<DeleteDashboardsResponse>
 deleteDashboardAsync(String dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
 DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: "
 + dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteDashboardsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
```

```
}
```

- Para obter detalhes da API, consulte a DeleteDashboardsreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui o painel especificado, solicitando uma confirmação antes de continuar. Para ignorar a confirmação, adicione a opção -Force para o comando.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Para obter detalhes da API, consulte DeleteDashboardsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui o painel especificado, solicitando uma confirmação antes de continuar. Para ignorar a confirmação, adicione a opção -Force para o comando.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Para obter detalhes da API, consulte DeleteDashboardsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeAlarmHistory`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAlarmHistory`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }
```

- Para obter detalhes da API, consulte DescribeAlarmHistorya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar o histórico de um alarme

O exemplo a seguir usa o `describe-alarm-history` comando para recuperar o histórico do CloudWatch alarme da Amazon chamado “myalarm”:

```
aws cloudwatch describe-alarm-history --alarm-name "myalarm" --history-item-
type StateUpdate
```

Saída:

```
{
    "AlarmHistoryItems": [
        {
            "Timestamp": "2014-04-09T18:59:06.442Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"ALARM\",\"stateReason\":\"testing purposes\"},\"newState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were not
 greater than the threshold (70.0). The most recent datapoints: [38.958,
 40.292].\",\"stateReasonData\":{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}}}",
            "HistorySummary": "Alarm updated from ALARM to OK"
        },
        {
            "Timestamp": "2014-04-09T18:59:05.805Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were
 not greater than the threshold (70.0). The most recent datapoints:
 [38.839999999999996, 39.714].\",\"stateReasonData\":{\"version\":
\"1.0\",\"queryDate\":\"2014-03-11T22:45:41.569+0000\",\"startDate\":
\"2014-03-11T22:30:00.000+0000\",\"statistic\":\"Average\",\"period\":300,
\"recentDatapoints\":[38.839999999999996,39.714],\"threshold\":70.0}},\"newState
\":{\"stateValue\":\"ALARM\",\"stateReason\":\"testing purposes\"}}",
            "HistorySummary": "Alarm updated from OK to ALARM"
```

```
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeAlarmHistoryem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the alarm history for a given alarm name and date range.
    *
    * @param fileName the path to the JSON file containing the alarm name
    * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
    * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
    */
   public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file",
e);
```

```
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response ->
response.alarmHistoryItems().forEach(item -> {
                        logger.info("History summary: {}",
item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
```

```
                throw new RuntimeException("Error completing alarm history
 processing", exception);
            }
        });
    }
```

- Para obter detalhes da API, consulte DescribeAlarmHistorya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
```

```
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAlarmHistoryreferência da API AWS SDK for Kotlin.

Use **`DescribeAlarms`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAlarms`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Começar a usar alarmes
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }
```

- Para obter detalhes da API, consulte DescribeAlarmsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar informações sobre um alarme

O seguinte exemplo usa o comando `describe-alarms` para fornecer informações sobre o alarme chamado “myalarm”:

```
aws cloudwatch describe-alarms --alarm-names "myalarm"
```

Saída:

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:myalarm",
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2012-12-27T00:49:54.032Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:123456789012:myHighCpuAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myalarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
```

```
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeAlarmsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
 operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
```

```
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ",
 alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}",
 ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }
```

- Para obter detalhes da API, consulte DescribeAlarmsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAlarmsreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Retorna todos os alarmes, incluindo alarmes compostos e métricos de. CloudWatch

```
Get-CWAlarm -MaxRecords 1
```

Saída:

```
CompositeAlarms MetricAlarms         NextToken
--------------- ------------         ---------
                {MetricAlarms-01}    NextToken-01
                {MetricAlarms-02}    NextToken-02
                {MetricAlarms-03}    NextToken-03
```

Exemplo 2: Retorna somente os dados de alarmes compostos CloudWatch após a configuração - AlarmType parâmetro para. CompositeAlarms

```
Get-CWAlarm -AlarmType 'CompositeAlarms'
```

Saída:

```
CompositeAlarms        MetricAlarms NextToken
---------------        ------------ ---------
{CompositeAlarms-01}
{CompositeAlarms-02}
```

```
{CompositeAlarms-03}
```

- Para obter detalhes da API, consulte DescribeAlarmsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudwatch'

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- Para obter detalhes da API, consulte DescribeAlarmsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cwt->describealarms(                " oo_result is
 returned for testing purposes. "
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarms retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAlarmsreferência da API AWS SDK for SAP ABAP.

Use **`DescribeAlarmsForMetric`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAlarmsForMetric`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Gerencie métricas e alarmes

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms ?? new List<MetricAlarm>();
    }
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetrica Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

Descreva os alarmes.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DescribeAlarmsRequest request;
        request.SetMaxRecords(1);

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.DescribeAlarms(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to describe CloudWatch alarms:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left <<
                    std::setw(32) << "Name" <<
```

```
                    std::setw(64) << "Arn" <<
                    std::setw(64) << "Description" <<
                    std::setw(20) << "LastUpdated" <<
                    std::endl;
                header = true;
            }

            const auto &alarms = outcome.GetResult().GetMetricAlarms();
            for (const auto &alarm : alarms)
            {
                std::cout << std::left <<
                    std::setw(32) << alarm.GetAlarmName() <<
                    std::setw(64) << alarm.GetAlarmArn() <<
                    std::setw(64) << alarm.GetAlarmDescription() <<
                    std::setw(20) <<
                    alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                        SIMPLE_DATE_FORMAT_STR) <<
                    std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetrica Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como exibir informações sobre os alarmes associados a uma métrica

O seguinte exemplo usa o comando `describe-alarms-for-metric` para exibir informações sobre qualquer alarme associado à métrica CPUUtilization do Amazon EC2 e à instância com o ID i-0c986c72.:

```
aws cloudwatch describe-alarms-for-metric --metric-name CPUUtilization --
namespace AWS/EC2 --dimensions Name=InstanceId,Value=i-0c986c72
```

Saída:

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 10,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm2",
            "StateUpdatedTimestamp": "2013-10-30T03:03:51.479Z",
            "AlarmConfigurationUpdatedTimestamp": "2013-10-30T03:03:50.865Z",
            "ComparisonOperator": "GreaterThanOrEqualToThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:NotifyMe"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2013-10-30T03:03:51.479+0000\",\"startDate\":\"2013-10-30T02:08:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":
[40.698,39.612,42.432,39.796,38.816,42.28,42.854,40.088,40.760000000000005,41.316],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm2",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 10 datapoints were not
 greater than or equal to the threshold (70.0). The most recent datapoints:
 [40.760000000000005, 41.316].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        },
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm",
```

```
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2014-04-09T22:26:05.958Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:HighCPUAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": false,
            "MetricName": "CPUUtilization"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetricem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Checks for a metric alarm in AWS CloudWatch.
    *
    * @param fileName the name of the file containing the JSON configuration for
the custom metric
    * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
    */
   public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for
further processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified
metric.
     *
     * @param metricRequest    the request to describe the alarms for the
specified metric
     * @param customMetricName the name of the custom metric to check for an
alarm
     * @param retries          the number of retries to perform if no alarm is
found
     * @return a {@link CompletableFuture} that completes when an alarm is found
or the maximum number of retries has been reached
     */
    private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
        if (retries == 0) {
            return CompletableFuture.completedFuture(null).thenRun(() ->
                logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
            );
        }
```

```
        return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response ->
{
            if (response.hasMetricAlarms()) {
                logger.info("Alarm state found for {}", customMetricName);
                return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
            } else {
                return CompletableFuture.runAsync(() -> {
                    try {
                        Thread.sleep(20000);
                        logger.info(".");
                    } catch (InterruptedException e) {
                        throw new RuntimeException("Interrupted while waiting to
retry", e);
                    }
                }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
            }
        }));
    }
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetrica Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";
```

```
const run = async () => {
  const command = new DescribeAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeAlarmsForMetrica Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
```

```
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.describeAlarms({ StateValue: "INSUFFICIENT_DATA" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeAlarmsForMetrica Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
```

```
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10
 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAlarmsForMetricreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
```

```
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
metric_name)
        return alarm_iter
```

- Para obter detalhes da API, consulte a DescribeAlarmsForMetricReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
```

```
response = cloudwatch_client.describe_alarms

if response.metric_alarms.count.positive?
  response.metric_alarms.each do |alarm|
    puts '-' * 16
    puts "Name:           #{alarm.alarm_name}"
    puts "State value:    #{alarm.state_value}"
    puts "State reason:   #{alarm.state_reason}"
    puts "Metric:         #{alarm.metric_name}"
    puts "Namespace:      #{alarm.namespace}"
    puts "Statistic:      #{alarm.statistic}"
    puts "Period:         #{alarm.period}"
    puts "Unit:           #{alarm.unit}"
    puts "Eval. periods:  #{alarm.evaluation_periods}"
    puts "Threshold:      #{alarm.threshold}"
    puts "Comp. operator: #{alarm.comparison_operator}"

    if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
      puts 'OK actions:'
      alarm.ok_actions.each do |a|
        puts "  #{a}"
      end
    end

    if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
      puts 'Alarm actions:'
      alarm.alarm_actions.each do |a|
        puts "  #{a}"
      end
    end

    if alarm.key?(:insufficient_data_actions) &&
       alarm.insufficient_data_actions.count.positive?
      puts 'Insufficient data actions:'
      alarm.insufficient_data_actions.each do |a|
        puts "  #{a}"
      end
    end

    puts 'Dimensions:'
    if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
      alarm.dimensions.each do |d|
        puts "  Name: #{d.name}, Value: #{d.value}"
      end
```

```
      else
        puts '  None for this alarm.'
      end
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ''

  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby cw-ruby-example-show-alarms.rb REGION'
    puts 'Example: ruby cw-ruby-example-show-alarms.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts 'Available alarms:'
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetrica Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cwt->describealarmsformetric(          " oo_result is
 returned for testing purposes. "
          iv_namespace   = iv_namespace
          iv_metricname  = iv_metric_name
          it_dimensions  = it_dimensions
          iv_statistic   = iv_statistic
          iv_period      = iv_period
          iv_unit        = iv_unit ).
        MESSAGE 'Alarms for metric retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAlarmsForMetricreferência da API AWS SDK for SAP ABAP.

Use **`DescribeAnomalyDetectors`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAnomalyDetectors`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }
```

- Para obter detalhes da API, consulte DescribeAnomalyDetectorsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar uma lista de modelos de detecção de anomalias

O exemplo `describe-anomaly-detectors` a seguir exibe informações sobre modelos de detectores de anomalias associados ao namespace `AWS/Logs` na conta específica.

```
aws cloudwatch describe-anomaly-detectors \
    --namespace AWS/Logs
```

Saída:

```
{
    "AnomalyDetectors": [
        {
            "Namespace": "AWS/Logs",
            "MetricName": "IncomingBytes",
            "Dimensions": [],
            "Stat": "SampleCount",
            "Configuration": {
                "ExcludedTimeRanges": []
            },
            "StateValue": "TRAINED",
            "SingleMetricAnomalyDetector": {
                "AccountId": "123456789012",
                "Namespace": "AWS/Logs",
                "MetricName": "IncomingBytes",
                "Dimensions": [],
                "Stat": "SampleCount"
            }
        },
        {
            "Namespace": "AWS/Logs",
            "MetricName": "IncomingBytes",
            "Dimensions": [
                {
                    "Name": "LogGroupName",
                    "Value": "demo"
                }
            ],
            "Stat": "Average",
```

```
            "Configuration": {
                "ExcludedTimeRanges": []
            },
            "StateValue": "PENDING_TRAINING",
            "SingleMetricAnomalyDetector": {
                "AccountId": "123456789012",
                "Namespace": "AWS/Logs",
                "MetricName": "IncomingBytes",
                "Dimensions": [
                    {
                        "Name": "LogGroupName",
                        "Value": "demo"
                    }
                ],
                "Stat": "Average"
            }
        }
    ]
}
```

Para obter mais informações, consulte Usando a detecção de CloudWatch anomalias no Guia do CloudWatch usuário da Amazon.

- Para obter detalhes da API, consulte DescribeAnomalyDetectorsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
```

```
    * @return a {@link CompletableFuture} that completes when the anomaly
detectors have been described
    * @throws RuntimeException if there is a failure during the operation, such
as when reading or parsing the JSON file,
    *                          or when describing the anomaly detectors
    */
   public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName)
{
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser);
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file",
e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                   .maxResults(10)
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

               return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response
-> {
                   List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                   for (AnomalyDetector detector : anomalyDetectorList) {
                       logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                       logger.info("State: {} ", detector.stateValue());
                   }
```

```
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly
 detectors", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
 exception);
            }
        });
    }
```

- Para obter detalhes da API, consulte DescribeAnomalyDetectorsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAnomalyDetectorsreferência da API AWS SDK for Kotlin.

Use **`DisableAlarmActions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DisableAlarmActions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Começar a usar alarmes
- Gerencie métricas e alarmes

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
```

```
    {
        var disableAlarmActionsResult = await
 _amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DisableAlarmActionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Desabilite as ações de alarme.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::DisableAlarmActionsRequest
 disableAlarmActionsRequest;
        disableAlarmActionsRequest.AddAlarmNames(alarm_name);
```

```
        auto disableAlarmActionsOutcome =
cw.DisableAlarmActions(disableAlarmActionsRequest);
        if (!disableAlarmActionsOutcome.IsSuccess())
        {
            std::cout << "Failed to disable actions for alarm " << alarm_name <<
                ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully disabled actions for alarm " <<
                alarm_name << std::endl;
        }
```

- Para obter detalhes da API, consulte DisableAlarmActionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como desativar ações de um alarme

O seguinte exemplo usa o comando `disable-alarm-actions` para desabilitar todas as ações do alarme “myalarm”:

```
aws cloudwatch disable-alarm-actions --alarm-names myalarm
```

Esse comando retornará ao prompt, se houver êxito.

- Para obter detalhes da API, consulte DisableAlarmActionsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
```

```
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);

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

- Para obter detalhes da API, consulte DisableAlarmActionsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DisableAlarmActionsCommand({
    AlarmNames: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DisableAlarmActionsa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.disableAlarmActions(
  { AlarmNames: ["Web_Server_CPU_Utilization"] },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DisableAlarmActionsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- Para obter detalhes da API, consulte a DisableAlarmActionsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
 they
                       disabled.
        """
```

```
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- Para obter detalhes da API, consulte a DisableAlarmActionsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
```

```
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  false
end

# Example usage:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: "BucketName",
      value: "amzn-s3-demo-bucket"
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'
```

```
  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

  if alarm_actions_disabled?(cloudwatch_client, alarm_name)
    puts "Alarm '#{alarm_name}' disabled."
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte DisableAlarmActionsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Disables actions on the specified alarm. "
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarm actions disabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DisableAlarmActionsreferência da API AWS SDK for SAP ABAP.

Use **`EnableAlarmActions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `EnableAlarmActions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerencie métricas e alarmes

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte EnableAlarmActionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Habilite as ações de alarme.

```
    Aws::CloudWatch::CloudWatchClient cw;
    Aws::CloudWatch::Model::PutMetricAlarmRequest request;
    request.SetAlarmName(alarm_name);
    request.SetComparisonOperator(
        Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
    request.SetEvaluationPeriods(1);
    request.SetMetricName("CPUUtilization");
    request.SetNamespace("AWS/EC2");
    request.SetPeriod(60);
    request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
    request.SetThreshold(70.0);
    request.SetActionsEnabled(false);
    request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
    request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
    request.AddAlarmActions(actionArn);

    Aws::CloudWatch::Model::Dimension dimension;
    dimension.SetName("InstanceId");
    dimension.SetValue(instanceId);
    request.AddDimensions(dimension);

    auto outcome = cw.PutMetricAlarm(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create CloudWatch alarm:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
    enable_request.AddAlarmNames(alarm_name);
```

```
    auto enable_outcome = cw.EnableAlarmActions(enable_request);
    if (!enable_outcome.IsSuccess())
    {
        std::cout << "Failed to enable alarm actions:" <<
            enable_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    std::cout << "Successfully created alarm " << alarm_name <<
        " and enabled actions on it." << std::endl;
```

- Para obter detalhes da API, consulte EnableAlarmActionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para habilitar todas as ações de um alarme

O seguinte exemplo usa o comando `enable-alarm-actions` para habilitar todas as ações para o alarme “myalarm”:

```
aws cloudwatch enable-alarm-actions --alarm-names myalarm
```

Esse comando retornará ao prompt, se tiver êxito.

- Para obter detalhes da API, consulte EnableAlarmActionsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to enable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
```

```
            EnableAlarmActionsRequest request =
 EnableAlarmActionsRequest.builder()
                    .alarmNames(alarm)
                    .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

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

- Para obter detalhes da API, consulte EnableAlarmActionsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { EnableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new EnableAlarmActionsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
```

```
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte EnableAlarmActionsa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
```

```
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"],
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Alarm action added", data);
    var paramsEnableAlarmAction = {
      AlarmNames: [params.AlarmName],
    };
    cw.enableAlarmActions(paramsEnableAlarmAction, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Alarm action enabled", data);
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte EnableAlarmActionsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- Para obter detalhes da API, consulte a EnableAlarmActionsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
```

```
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- Para obter detalhes da API, consulte a EnableAlarmActionsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Enable actions on the specified alarm."
    TRY.
        lo_cwt->enablealarmactions(
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarm actions enabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a EnableAlarmActionsreferência da API AWS SDK for SAP ABAP.

Use **`GetDashboard`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetDashboard`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Como criar um painel de monitoramento com o nome da função como uma variável
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get information on a dashboard.
/// </summary>
/// <param name="dashboardName">The name of the dashboard.</param>
/// <returns>A JSON object with dashboard information.</returns>
public async Task<string> GetDashboard(string dashboardName)
{
    var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
        new GetDashboardRequest()
        {
            DashboardName = dashboardName
        });

    return dashboardResponse.DashboardBody;
}
```

- Para obter detalhes da API, consulte GetDashboarda Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar informações sobre um painel

O exemplo `get-dashboard` a seguir exibe as informações sobre o painel denominado `Dashboard-A` na conta específica.

```
aws cloudwatch get-dashboard \
```

```
    --dashboard-name Dashboard-A
```

Saída:

```
{
    "DashboardArn": "arn:aws:cloudwatch::123456789012:dashboard/Dashboard-A",
    "DashboardBody": "{\"widgets\":[{\"type\":\"metric\",\"x\":0,\"y\":0,\"width
\":6,\"height\":6,\"properties\":{\"view\":\"timeSeries\",\"stacked\":false,
\"metrics\":[[\"AWS/EC2\",\"NetworkIn\",\"InstanceId\",\"i-0131f062232ade043\"],
[\".\",\"NetworkOut\",\".\",\".\"]],\"region\":\"us-east-1\"}}]}",
    "DashboardName": "Dashboard-A"
}
```

Para obter mais informações, consulte os CloudWatch painéis da Amazon no Guia do CloudWatch usuário da Amazon.

- Para obter detalhes da API, consulte GetDashboardem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna o ARN do corpo do painel especificado.

```
Get-CWDashboard -DashboardName Dashboard1
```

Saída:

```
DashboardArn                                          DashboardBody
------------                                          -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Para obter detalhes da API, consulte GetDashboardem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna o ARN do corpo do painel especificado.

```
Get-CWDashboard -DashboardName Dashboard1
```

Saída:

```
DashboardArn                                          DashboardBody
------------                                          -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Para obter detalhes da API, consulte GetDashboardem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetMetricData`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetMetricData`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
```

```
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        // Add the plus sign for positive offsets.
        timeZoneString = timeZoneString.StartsWith('-') ? timeZoneString : "+" +
 timeZoneString;
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
 ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        if (paginatedMetricData.MetricDataResults != null)
        {
            await foreach (var data in paginatedMetricData.MetricDataResults)
            {
                metricData.Add(data);
            }
        }

        return metricData;
    }
```

- Para obter detalhes da API, consulte GetMetricDataa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: para obter a média total de IOPS para o EC2 especificado usando a expressão matemática

O `get-metric-data` exemplo a seguir recupera valores CloudWatch métricos para a instância do EC2 com instanceID `i-abcdef` usando uma expressão matemática métrica que combina métricas e métricas. `EBSReadOps` `EBSWriteOps`

```
aws cloudwatch get-metric-data \
    --metric-data-queries file://file.json \
    --start-time 2024-09-29T22:10:00Z \
    --end-time 2024-09-29T22:15:00Z
```

Conteúdo de `file.json`:

```
[
    {
        "Id": "m3",
        "Expression": "(m1+m2)/300",
        "Label": "Avg Total IOPS"
    },
    {
        "Id": "m1",
        "MetricStat": {
            "Metric": {
                "Namespace": "AWS/EC2",
                "MetricName": "EBSReadOps",
                "Dimensions": [
                    {
                        "Name": "InstanceId",
                        "Value": "i-abcdef"
                    }
                ]
            },
            "Period": 300,
            "Stat": "Sum",
            "Unit": "Count"
        },
        "ReturnData": false
    },
```

```
    {
        "Id": "m2",
        "MetricStat": {
            "Metric": {
                "Namespace": "AWS/EC2",
                "MetricName": "EBSWriteOps",
                "Dimensions": [
                    {
                        "Name": "InstanceId",
                        "Value": "i-abcdef"
                    }
                ]
            },
            "Period": 300,
            "Stat": "Sum",
            "Unit": "Count"
        },
        "ReturnData": false
    }
]
```

Saída:

```
{
    "MetricDataResults": [
        {
            "Id": "m3",
            "Label": "Avg Total IOPS",
            "Timestamps": [
                "2024-09-29T22:10:00+00:00"
            ],
            "Values": [
                96.85
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Exemplo 2: Para monitorar as AWS cobranças estimadas usando métricas de CloudWatch faturamento

O `get-metric-data` exemplo a seguir recupera a `EstimatedCharges` CloudWatch métrica do namespace AWS/Billing.

```
aws cloudwatch get-metric-data \
    --metric-data-queries '[{"Id":"m1","MetricStat":{"Metric":
{"Namespace":"AWS/Billing","MetricName":"EstimatedCharges","Dimensions":
[{"Name":"Currency","Value":"USD"}]},"Period":21600,"Stat":"Maximum"}}]' \
    --start-time 2024-09-26T12:00:00Z \
    --end-time 2024-09-26T18:00:00Z \
    --region us-east-1
```

Saída:

```
{
    "MetricDataResults": [
        {
            "Id": "m1",
            "Label": "EstimatedCharges",
            "Timestamps": [
                "2024-09-26T12:00:00+00:00"
            ],
            "Values": [
                542.38
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Para obter mais informações, consulte Uso de expressões matemáticas com CloudWatch métricas no Guia CloudWatch do usuário da Amazon.

- Para obter detalhes da API, consulte GetMetricDataem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves custom metric data from the AWS CloudWatch service.
    *
    * @param fileName the name of the file containing the custom metric
information
    * @return a {@link CompletableFuture} that completes when the metric data
has been retrieved
    */
   public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               // Read values from the JSON file.
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for
further processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               // Parse the JSON string to extract relevant values.
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
```

```
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}",
 item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }
```

- Para obter detalhes da API, consulte GetMetricDataa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
```

```
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte a GetMetricDatareferência da API AWS SDK for Kotlin.

Use **`GetMetricStatistics`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetMetricStatistics`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Gerencie métricas e alarmes

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
```

```
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints ?? new List<Datapoint>();
    }
```

- Para obter detalhes da API, consulte GetMetricStatisticsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como obter a utilização da CPU por instância do EC2

O exemplo a seguir usa o comando `get-metric-statistics` para obter a utilização da CPU de uma instância do EC2 com o ID i-abcdef.

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-
time 2014-04-08T23:18:00Z --end-time 2014-04-09T23:18:00Z --period 3600 --
namespace AWS/EC2 --statistics Maximum --dimensions Name=InstanceId,Value=i-
abcdef
```

Saída:

```
{
    "Datapoints": [
        {
            "Timestamp": "2014-04-09T11:18:00Z",
            "Maximum": 44.79,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T20:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T19:18:00Z",
            "Maximum": 50.85,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T09:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T03:18:00Z",
            "Maximum": 76.84,
            "Unit": "Percent"
        },
```

```
        {
            "Timestamp": "2014-04-09T21:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T14:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T08:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T16:18:00Z",
            "Maximum": 45.55,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T06:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T13:18:00Z",
            "Maximum": 45.08,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T05:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T18:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T17:18:00Z",
            "Maximum": 52.08,
            "Unit": "Percent"
```

```
        },
        {
            "Timestamp": "2014-04-09T07:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T02:18:00Z",
            "Maximum": 51.23,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T12:18:00Z",
            "Maximum": 47.67,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-08T23:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T10:18:00Z",
            "Maximum": 51.91,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T04:18:00Z",
            "Maximum": 47.13,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T15:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T00:18:00Z",
            "Maximum": 48.16,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T01:18:00Z",
            "Maximum": 49.18,
```

```
            "Unit": "Percent"
        }
    ],
    "Label": "CPUUtilization"
}
```

Especificar várias dimensões

O exemplo a seguir ilustra como especificar diversas dimensões. Cada dimensão é especificada como um Name/Value par, com uma vírgula entre o nome e o valor. As diversas dimensões são separadas por um espaço. Se uma única métrica incluir diversas dimensões, você deverá especificar um valor para cada dimensão definida.

Para obter mais exemplos de uso do `get-metric-statistics` comando, consulte Obter estatísticas para uma métrica no Amazon CloudWatch Developer Guide.

```
aws cloudwatch get-metric-statistics --metric-name Buffers --
namespace MyNameSpace --dimensions Name=InstanceID,Value=i-
abcdef Name=InstanceType,Value=m1.small --start-time 2016-10-15T04:00:00Z --end-
time 2016-10-19T07:00:00Z --statistics Average --period 60
```

- Para obter detalhes da API, consulte GetMetricStatisticsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
     * @param metVal       the name of the metric
```

```
    * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
    * @param date         the date for which to retrieve the metric statistics,
in the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
    * @param myDimension  the dimension(s) to filter the metric statistics by
    * @return a {@link CompletableFuture} that completes when the metric
statistics have been retrieved and displayed
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

           String metricOption, String date, Dimension myDimension) {

       Instant start = Instant.parse(date);
       Instant endDate = Instant.now();

       // Building the request for metric statistics.
       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .endTime(endDate)
           .startTime(start)
           .dimensions(myDimension)
           .metricName(metVal)
           .namespace(nameSpace)
           .period(86400) // 1 day period
           .statistics(Statistic.fromValue(metricOption))
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                       }
                   } else {
                       logger.info("The returned data list is empty");
                   }
               } else {
                   logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
               }
```

```
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric
statistics: " + exception.getMessage(), exception);
            });
    }
```

- Para obter detalhes da API, consulte GetMetricStatisticsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
```

```
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a GetMetricStatisticsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
```

```
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period
must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
value
                           or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
            raise
        else:
            return stats
```

- Para obter detalhes da API, consulte a GetMetricStatisticsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_cwt->getmetricstatistics(              " oo_result is
returned for testing purposes. "
         iv_namespace   = iv_namespace
         iv_metricname  = iv_metric_name
         iv_starttime   = iv_start_time
         iv_endtime     = iv_end_time
         iv_period      = iv_period
         it_statistics  = it_statistics ).
       MESSAGE 'Metric statistics retrieved.' TYPE 'I'.
     CATCH /aws1/cx_cwtinvparamvalueex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetMetricStatisticsreferência da API AWS SDK for SAP ABAP.

Use **`GetMetricWidgetImage`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetMetricWidgetImage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });
```

```
        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }
```

- Para obter detalhes da API, consulte GetMetricWidgetImagea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar um grafo de snapshot de CPUUtilization

O exemplo `get-metric-widget-image` a seguir recupera o grafo de snapshot da métrica `CPUUtilization` da instância do EC2 com o ID `i-abcde` e salva a imagem recuperada como um arquivo denominado "image.png" na máquina local.

```
aws cloudwatch get-metric-widget-image \
    --metric-widget '{"metrics":[["AWS/EC2","CPUUtilization","InstanceId","i-abcde"]]}' \
    --output-format png \
    --output text | base64 --decode > image.png
```

Este comando não produz saída.

- Para obter detalhes da API, consulte GetMetricWidgetImageem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves and saves a custom metric image to a file.
    *
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String
fileName) {
       logger.info("Getting Image data for custom metric.");
       String myJSON = """
             {
                 "title": "Example Metric Graph",
                 "view": "timeSeries",
                 "stacked ": false,
                 "period": 10,
                 "width": 1400,
                 "height": 600,
                 "metrics": [
                     [
                     "AWS/Billing",
                     "EstimatedCharges",
                     "Currency",
                     "USD"
                    ]
                 ]
             }
```

```
            """;

        GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

        return getAsyncClient().getMetricWidgetImage(imageRequest)
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to
file", e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }
```

- Para obter detalhes da API, consulte GetMetricWidgetImagea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
```

```
}
```

- Para obter detalhes da API, consulte a GetMetricWidgetImagereferência da API AWS SDK for Kotlin.

Use **`ListDashboards`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListDashboards`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Como criar um painel de monitoramento com o nome da função como uma variável

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get a list of dashboards.
/// </summary>
/// <returns>A list of DashboardEntry objects.</returns>
public async Task<List<DashboardEntry>> ListDashboards()
{
    var results = new List<DashboardEntry>();
    var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
        new ListDashboardsRequest());
    // Get the entire list using the paginator.
    await foreach (var data in paginateDashboards.DashboardEntries)
    {
        results.Add(data);
    }

    return results;
```

```
    }
```

- Para obter detalhes da API, consulte ListDashboardsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar uma lista de painéis

O exemplo `list-dashboards` a seguir lista todos os painéis na conta especificada.

```
aws cloudwatch list-dashboards
```

Saída:

```
{
    "DashboardEntries": [
        {
            "DashboardName": "Dashboard-A",
            "DashboardArn": "arn:aws:cloudwatch::123456789012:dashboard/
Dashboard-A",
            "LastModified": "2024-10-11T18:40:11+00:00",
            "Size": 271
        },
        {
            "DashboardName": "Dashboard-B",
            "DashboardArn": "arn:aws:cloudwatch::123456789012:dashboard/
Dashboard-B",
            "LastModified": "2024-10-11T18:44:41+00:00",
            "Size": 522
        }
    ]
}
```

Para obter mais informações, consulte os CloudWatch painéis da Amazon no Guia do CloudWatch usuário da Amazon.

- Para obter detalhes da API, consulte ListDashboardsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Lists the available dashboards.
    *
    * @return a {@link CompletableFuture} that completes when the operation is
finished.
    * The future will complete exceptionally if an error occurs while listing
the dashboards.
    */
   public CompletableFuture<Void> listDashboardsAsync() {
       ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
       ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
       return paginator.subscribe(response -> {
           response.dashboardEntries().forEach(entry -> {
               logger.info("Dashboard name is: {} ", entry.dashboardName());
               logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
           });
       }).exceptionally(ex -> {
           logger.info("Failed to list dashboards: {} ", ex.getMessage());
           throw new RuntimeException("Error occurred while listing dashboards",
ex);
       });
   }
```

- Para obter detalhes da API, consulte ListDashboardsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListDashboardsreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna a coleção de painéis para sua conta.

```
Get-CWDashboardList
```

Saída:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Exemplo 2: retorna a coleção de painéis para sua conta cujos nomes começam com o prefixo “dev”.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Para obter detalhes da API, consulte ListDashboardsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de painéis para sua conta.

```
Get-CWDashboardList
```

Saída:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Exemplo 2: retorna a coleção de painéis para sua conta cujos nomes começam com o prefixo “dev”.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Para obter detalhes da API, consulte ListDashboardsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListMetrics`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListMetrics`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Gerencie métricas e alarmes

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }
```

- Para obter detalhes da API, consulte ListMetricsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

Liste as métricas.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::ListMetricsRequest request;

        if (argc > 1)
        {
            request.SetMetricName(argv[1]);
        }

        if (argc > 2)
        {
            request.SetNamespace(argv[2]);
        }

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.ListMetrics(request);
            if (!outcome.IsSuccess())
            {
```

```
                std::cout << "Failed to list CloudWatch metrics:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left << std::setw(48) << "MetricName" <<
                    std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
                    std::endl;
                header = true;
            }

            const auto &metrics = outcome.GetResult().GetMetrics();
            for (const auto &metric : metrics)
            {
                std::cout << std::left << std::setw(48) <<
                    metric.GetMetricName() << std::setw(32) <<
                    metric.GetNamespace();
                const auto &dimensions = metric.GetDimensions();
                for (auto iter = dimensions.cbegin();
                    iter != dimensions.cend(); ++iter)
                {
                    const auto &dimkv = *iter;
                    std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
                    if (iter + 1 != dimensions.cend())
                    {
                        std::cout << ", ";
                    }
                }
                std::cout << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Para obter detalhes da API, consulte ListMetricsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar as métricas do Amazon SNS

O exemplo apresentado a seguir para `list-metrics` exibe as métricas do Amazon SNS.

```
aws cloudwatch list-metrics \
    --namespace "AWS/SNS"
```

Saída:

```
{
    "Metrics": [
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
```

```
            "MetricName": "NumberOfNotificationsFailed"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
```

```
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfNotificationsFailed"
        }
    ]
}
```

- Para obter detalhes da API, consulte ListMetricsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an
{@link ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
```

```
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
 exception.getMessage(), exception);
            });
    }
```

- Para obter detalhes da API, consulte ListMetricsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import {
  CloudWatchServiceException,
  ListMetricsCommand,
} from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

export const main = async () => {
  // Use the AWS console to see available namespaces and metric names. Custom
 metrics can also be created.
  // https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
viewing_metrics_with_cloudwatch.html
  const command = new ListMetricsCommand({
    Dimensions: [
      {
```

```
        Name: "LogGroupName",
      },
    ],
    MetricName: "IncomingLogEvents",
    Namespace: "AWS/Logs",
  });

  try {
    const response = await client.send(command);
    console.log(`Metrics count: ${response.Metrics?.length}`);
    return response;
  } catch (caught) {
    if (caught instanceof CloudWatchServiceException) {
      console.error(`Error from CloudWatch. ${caught.name}: ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListMetricsa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  Dimensions: [
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};

cw.listMetrics(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListMetricsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
```

- Para obter detalhes da API, consulte a ListMetricsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource
```

```
    def list_metrics(self, namespace, name, recent=False):
        """
        Gets the metrics within a namespace that have the specified name.
        If the metric has no dimensions, a single metric is returned.
        Otherwise, metrics for all dimensions are returned.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param recent: When True, only metrics that have been active in the last
                       three hours are returned.
        :return: An iterator that yields the retrieved metrics.
        """
        try:
            kwargs = {"Namespace": namespace, "MetricName": name}
            if recent:
                kwargs["RecentlyActive"] = "PT3H"  # List past 3 hours only
            metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
            logger.info("Got metrics for %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
            raise
        else:
            return metric_iter
```

- Para obter detalhes da API, consulte a ListMetricsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
```

```
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts '    Dimensions:'
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts 'No dimensions found.'
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      'Note that it could take up to 15 minutes for recently-added metrics ' \
      'to become available.'
  end
end

# Example usage:
def run_me
  metric_namespace = 'SITE/TRAFFIC'
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisitors',
    'SiteName',
```

```
    'example.com',
    5_885.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisits',
    'SiteName',
    'example.com',
    8_628.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'PageViews',
    'PageURL',
    'example.html',
    18_057.0,
    'Count'
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte ListMetricsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "The following list-metrics example displays the metrics for Amazon
CloudWatch."
   TRY.
       oo_result = lo_cwt->listmetrics(            " oo_result is returned for
testing purposes. "
         iv_namespace = iv_namespace ).
       DATA(lt_metrics) = oo_result->get_metrics( ).
       MESSAGE 'Metrics retrieved.' TYPE 'I'.
     CATCH /aws1/cx_cwtinvparamvalueex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListMetricsreferência da API AWS SDK for SAP ABAP.

Use **`PutAnomalyDetector`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutAnomalyDetector`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
 anomalyDetector)
    {
        var putAlarmDetectorResult = await
 _amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte PutAnomalyDetectora Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para criar um modelo de detecção de anomalias

O `put-anomaly-detector` exemplo a seguir cria um modelo de detecção de anomalias para uma CloudWatch métrica.

```
aws cloudwatch put-anomaly-detector \
    --namespace AWS/Logs \
    --metric-name IncomingBytes \
    --stat SampleCount
```

Este comando não produz saída.

Para obter mais informações, consulte Usando a detecção de CloudWatch anomalias no Guia do CloudWatch usuário da Amazon.

- Para obter detalhes da API, consulte PutAnomalyDetectorem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Adds an anomaly detector for the given file.
    *
    * @param fileName the name of the file containing the anomaly detector
configuration
    * @return a {@link CompletableFuture} that completes when the anomaly
detector has been added
    */
   public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root
node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file",
e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
```

```
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

                PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response
-> {
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }
```

- Para obter detalhes da API, consulte PutAnomalyDetectora Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- Para obter detalhes da API, consulte a PutAnomalyDetectorreferência da API AWS SDK for Kotlin.

Use **PutDashboard**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutDashboard`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
```

```
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
 newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages ?? new
 List<DashboardValidationMessage>();
    }
```

- Para obter detalhes da API, consulte PutDashboarda Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para criar um painel

O exemplo `put-dashboard` a seguir cria um painel denominado `Dashboard-A` na conta especificada.

```
aws cloudwatch put-dashboard \
    --dashboard-name Dashboard-A \
    --dashboard-body '{"widgets":
[{"height":6,"width":6,"y":0,"x":0,"type":"metric","properties":
{"view":"timeSeries","stacked":false,"metrics":
[["Namespace","CPUUtilization","Environment","Prod","Type","App"]],"region":"us-
east-1"}}]}'
```

Saída:

```
{
    "DashboardValidationMessages": []
}
```

Para obter mais informações, consulte Criação de um CloudWatch painel no Guia CloudWatch do usuário da Amazon.

- Para obter detalhes da API, consulte PutDashboardem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new dashboard with the specified name and metrics from the given
file.
    *
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous
operation of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from
the file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}",
ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
```

```
                    if (messages.isEmpty()) {
                        logger.info("There are no messages in the new
 Dashboard.");
                    } else {
                        for (DashboardValidationMessage message : messages) {
                            logger.info("Message: {}", message.message());
                        }
                    }
                    return response; // Return the response for further use
                }
            });
    }
```

- Para obter detalhes da API, consulte PutDashboarda Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
```

```
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a PutDashboardreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria ou atualiza o painel denominado “Dashboard1” para incluir dois widgets de métricas lado a lado.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
                      "CPUUtilization",
                      "InstanceId",
                      "i-012345"
                   ]
                ],
                "period":300,
                "stat":"Average",
                "region":"us-east-1",
```

```
                "title":"EC2 Instance CPU"
            }
        },
        {
            "type":"metric",
            "x":12,
            "y":0,
            "width":12,
            "height":6,
            "properties":{
                "metrics":[
                    [
                        "AWS/S3",
                        "BucketSizeBytes",
                        "BucketName",
                        "amzn-s3-demo-bucket"
                    ]
                ],
                "period":86400,
                "stat":"Maximum",
                "region":"us-east-1",
                "title":"amzn-s3-demo-bucket bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Exemplo 2: cria ou atualiza o painel, redirecionando o conteúdo que descreve o painel para o cmdlet.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Para obter detalhes da API, consulte PutDashboardem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria ou atualiza o painel denominado “Dashboard1” para incluir dois widgets de métricas lado a lado.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
                      "CPUUtilization",
                      "InstanceId",
                      "i-012345"
                   ]
                ],
                "period":300,
                "stat":"Average",
                "region":"us-east-1",
                "title":"EC2 Instance CPU"
             }
        },
        {
             "type":"metric",
             "x":12,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/S3",
                      "BucketSizeBytes",
                      "BucketName",
                      "amzn-s3-demo-bucket"
                   ]
                ],
```

```
                "period":86400,
                "stat":"Maximum",
                "region":"us-east-1",
                "title":"amzn-s3-demo-bucket bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Exemplo 2: cria ou atualiza o painel, redirecionando o conteúdo que descreve o painel para o cmdlet.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Para obter detalhes da API, consulte PutDashboardem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutMetricAlarm`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutMetricAlarm`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Começar a usar alarmes
- Gerencie métricas e alarmes
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Add a metric alarm to send an email when the metric passes a threshold.
   /// </summary>
   /// <param name="alarmDescription">A description of the alarm.</param>
   /// <param name="alarmName">The name for the alarm.</param>
   /// <param name="comparison">The type of comparison to use.</param>
   /// <param name="metricName">The name of the metric for the alarm.</param>
   /// <param name="metricNamespace">The namespace of the metric.</param>
   /// <param name="threshold">The threshold value for the alarm.</param>
   /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
       string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
   {
       try
       {
           var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
               new PutMetricAlarmRequest()
               {
                   AlarmActions = alarmActions,
                   AlarmDescription = alarmDescription,
                   AlarmName = alarmName,
                   ComparisonOperator = comparison,
                   Threshold = threshold,
                   Namespace = metricNamespace,
                   MetricName = metricName,
                   EvaluationPeriods = 1,
                   Period = 10,
```

```
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
 has already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
 alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }
```

- Para obter detalhes da API, consulte PutMetricAlarma Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Crie o alarme para vigiar a métrica.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::PutMetricAlarmRequest request;
        request.SetAlarmName(alarm_name);
        request.SetComparisonOperator(
            Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
        request.SetEvaluationPeriods(1);
        request.SetMetricName("CPUUtilization");
        request.SetNamespace("AWS/EC2");
        request.SetPeriod(60);
        request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
        request.SetThreshold(70.0);
        request.SetActionsEnabled(false);
        request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
        request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("InstanceId");
        dimension.SetValue(instanceId);

        request.AddDimensions(dimension);

        auto outcome = cw.PutMetricAlarm(request);
```

```
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- Para obter detalhes da API, consulte PutMetricAlarma Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como enviar uma mensagem de e-mail do Amazon Simple Notification Service quando a utilização da CPU exceder 70%

O seguinte exemplo usa o comando `put-metric-alarm` para enviar uma mensagem de e-mail do Amazon Simple Notification Service quando a utilização da CPU excede 70%:

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70 percent" --metric-name CPUUtilization --namespace AWS/
EC2 --statistic Average --period 300 --threshold 70 --comparison-
operator GreaterThanThreshold  --dimensions "Name=InstanceId,Value=i-12345678" --
evaluation-periods 2 --alarm-actions arn:aws:sns:us-east-1:111122223333:MyTopic
 --unit Percent
```

Esse comando retornará ao prompt, se houver êxito. Se já existir um alarme com o mesmo nome, ele será substituído pelo novo alarme.

Como especificar diversas dimensões

O exemplo a seguir ilustra como especificar diversas dimensões. Cada dimensão é especificada como um Name/Value par, com uma vírgula entre o nome e o valor. As diversas dimensões são separadas por um espaço:

```
aws cloudwatch put-metric-alarm --alarm-name "Default_Test_Alarm3" --alarm-
description "The default example alarm" --namespace "CW EXAMPLE METRICS"
 --metric-name Default_Test --statistic Average --period 60 --evaluation-
periods 3 --threshold 50 --comparison-operator GreaterThanOrEqualToThreshold --
dimensions Name=key1,Value=value1 Name=key2,Value=value2
```

- Para obter detalhes da API, consulte PutMetricAlarmem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm
 configuration
     * @return a CompletableFuture that represents the asynchronous operation of
 creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON
 file or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new
 File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
 file", e);
        }

        // Extract values from the JSON node.
```

```
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }
```

- Para obter detalhes da API, consulte PutMetricAlarma Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // This alarm triggers when CPUUtilization exceeds 70% for one minute.
  const command = new PutMetricAlarmCommand({
    AlarmName: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
    ComparisonOperator: "GreaterThanThreshold",
    EvaluationPeriods: 1,
    MetricName: "CPUUtilization",
    Namespace: "AWS/EC2",
    Period: 60,
    Statistic: "Average",
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: "Alarm when server CPU exceeds 70%",
    Dimensions: [
      {
        Name: "InstanceId",
        Value: process.env.EC2_INSTANCE_ID, // Set the value of EC_INSTANCE_ID to
 the Id of an existing Amazon EC2 instance.
      },
    ],
    Unit: "Percent",
  });
```

```
  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutMetricAlarma Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
```

```
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutMetricAlarma Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun putMetricAlarm(
```

```
    alarmNameVal: String,
    instanceIdVal: String,
) {
    val dimensionOb =
        Dimension {
            name = "InstanceId"
            value = instanceIdVal
        }

    val request =
        PutMetricAlarmRequest {
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanThreshold
            evaluationPeriods = 1
            metricName = "CPUUtilization"
            namespace = "AWS/EC2"
            period = 60
            statistic = Statistic.fromValue("Average")
            threshold = 70.0
            actionsEnabled = false
            alarmDescription = "An Alarm created by the Kotlin SDK when server
 CPU utilization exceeds 70%"
            unit = StandardUnit.fromValue("Seconds")
            dimensions = listOf(dimensionOb)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- Para obter detalhes da API, consulte a PutMetricAlarmreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over
 the
```

```
                              alarm threshold before the alarm is set into an
alarmed
                              state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the
threshold
                               against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm
```

- Para obter detalhes da API, consulte a PutMetricAlarmReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
```

```
#     [
#       {
#         name: 'BucketName',
#         value: 'amzn-s3-demo-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
```

```
  )
  true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  false
end
```

- Para obter detalhes da API, consulte PutMetricAlarma Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_cwt->putmetricalarm(
          iv_alarmname                = iv_alarm_name
          iv_comparisonoperator       = iv_comparison_operator
          iv_evaluationperiods        = iv_evaluation_periods
          iv_metricname               = iv_metric_name
          iv_namespace                = iv_namespace
          iv_statistic                = iv_statistic
          iv_threshold                = iv_threshold
          iv_actionsenabled           = iv_actions_enabled
          iv_alarmdescription         = iv_alarm_description
          iv_unit                     = iv_unit
          iv_period                   = iv_period
          it_dimensions               = it_dimensions ).
        MESSAGE 'Alarm created.' TYPE 'I'.
      CATCH /aws1/cx_cwtlimitexceededfault.
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutMetricAlarmreferência da API AWS SDK for SAP ABAP.

Use **PutMetricData**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutMetricData`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Gerencie métricas e alarmes

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
```

```
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
        return customData;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte PutMetricDataa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

Insira dados na métrica.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("UNIQUE_PAGES");
        dimension.SetValue("URLS");

        Aws::CloudWatch::Model::MetricDatum datum;
        datum.SetMetricName("PAGES_VISITED");
        datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
        datum.SetValue(data_point);
        datum.AddDimensions(dimension);

        Aws::CloudWatch::Model::PutMetricDataRequest request;
        request.SetNamespace("SITE/TRAFFIC");
        request.AddMetricData(datum);

        auto outcome = cw.PutMetricData(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to put sample metric data:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
```

```
        else
        {
            std::cout << "Successfully put sample metric data" << std::endl;
        }
```

- Para obter detalhes da API, consulte PutMetricDataa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para publicar uma métrica personalizada na Amazon CloudWatch

O exemplo a seguir usa o `put-metric-data` comando para publicar uma métrica personalizada na Amazon CloudWatch:

```
aws cloudwatch put-metric-data --namespace "Usage Metrics" --metric-data file:// metric.json
```

Os valores da própria métrica estão armazenados no arquivo em JSON, `metric.json`.

Veja o conteúdo desse arquivo:

```
[
  {
    "MetricName": "New Posts",
    "Timestamp": "Wednesday, June 12, 2013 8:28:20 PM",
    "Value": 0.50,
    "Unit": "Count"
  }
]
```

Para obter mais informações, consulte Publicação de métricas personalizadas no Amazon CloudWatch Developer Guide.

Como especificar diversas dimensões

O exemplo a seguir ilustra como especificar diversas dimensões. Cada dimensão é especificada como um par de nome/valor. As diversas dimensões são separadas por uma vírgula.

```
aws cloudwatch put-metric-data --metric-name Buffers --
namespace MyNameSpace --unit Bytes --value 231434333 --
dimensions InstanceID=1-23456789,InstanceType=m1.small
```

- Para obter detalhes da API, consulte PutMetricDataem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for
further processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
```

```
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

                MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed
to parse JSON content", e));
                return failedFuture;
            }
```

```
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " +
 exception.getMessage(), exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }
```

- Para obter detalhes da API, consulte PutMetricDataa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // See https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/
API_PutMetricData.html#API_PutMetricData_RequestParameters
  // and https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
publishingMetrics.html
  // for more information about the parameters in this command.
  const command = new PutMetricDataCommand({
    MetricData: [
      {
        MetricName: "PAGES_VISITED",
        Dimensions: [
          {
            Name: "UNIQUE_PAGES",
```

```
            Value: "URLS",
          },
        ],
        Unit: "None",
        Value: 1.0,
      },
    ],
    Namespace: "SITE/TRAFFIC",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutMetricDataa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
    {
      MetricName: "PAGES_VISITED",
      Dimensions: [
        {
          Name: "UNIQUE_PAGES",
          Value: "URLS",
        },
      ],
      Unit: "None",
      Value: 1.0,
    },
  ],
  Namespace: "SITE/TRAFFIC",
};

cw.putMetricData(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", JSON.stringify(data));
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutMetricDataa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }
```

```
    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- Para obter detalhes da API, consulte a PutMetricDatareferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria um novo MetricDatum objeto e o grava no Amazon Web Services CloudWatch Metrics.

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- Para obter detalhes da API, consulte PutMetricDataem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria um novo MetricDatum objeto e o grava no Amazon Web Services CloudWatch Metrics.

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- Para obter detalhes da API, consulte PutMetricDataem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is
 given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
```

```
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
 name)
            raise
```

Coloque um conjunto de dados em uma CloudWatch métrica.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
 set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
```

```
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise
```

- Para obter detalhes da API, consulte a PutMetricDataReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudwatch'

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
```

```
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
```

```
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  false
end
```

- Para obter detalhes da API, consulte PutMetricDataa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_metricdata TYPE /aws1/cl_cwtmetricdatum=>tt_metricdata.

    "Create metric data object.
    DATA(lo_metricdatum) = NEW /aws1/cl_cwtmetricdatum(
      iv_metricname = iv_metric_name
      iv_value      = iv_value
      iv_unit       = iv_unit ).

    INSERT lo_metricdatum INTO TABLE lt_metricdata.

    TRY.
        lo_cwt->putmetricdata(
          iv_namespace   = iv_namespace
          it_metricdata  = lt_metricdata ).
        MESSAGE 'Metric data added.' TYPE 'I'.
```

```
    CATCH /aws1/cx_cwtinvparamvalueex.
      MESSAGE 'The specified argument was not valid.' TYPE 'E'.
  ENDTRY.
```

Coloque um conjunto de dados em uma CloudWatch métrica.

```
  DATA lt_metricdata TYPE /aws1/cl_cwtmetricdatum=>tt_metricdata.

  "Create metric data object with values and counts.
  DATA(lo_metricdatum) = NEW /aws1/cl_cwtmetricdatum(
    iv_metricname = iv_metric_name
    iv_timestamp  = iv_timestamp
    iv_unit       = iv_unit
    it_values     = it_values
    it_counts     = it_counts ).

  INSERT lo_metricdatum INTO TABLE lt_metricdata.

  TRY.
      lo_cwt->putmetricdata(
        iv_namespace   = iv_namespace
        it_metricdata  = lt_metricdata ).
      MESSAGE 'Metric data set added.' TYPE 'I'.
    CATCH /aws1/cx_cwtinvparamvalueex.
      MESSAGE 'The specified argument was not valid.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a PutMetricDatareferência da API AWS SDK for SAP ABAP.

Cenários para CloudWatch usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns CloudWatch com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas CloudWatch ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Como criar um painel de monitoramento com o nome da função como uma variável
- Comece a usar CloudWatch alarmes usando um AWS SDK
- Gerencie CloudWatch métricas e alarmes usando um AWS SDK
- Monitore o desempenho do Amazon DynamoDB usando um AWS SDK
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

Como criar um painel de monitoramento com o nome da função como uma variável

O exemplo de código a seguir mostra como:

- Crie um CloudWatch painel
- Adicionar widgets de métricas do Lambda com uma variável de nome de função.
- Verificar o painel.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions
```

```
# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
```

```
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF
```

```
    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
```

```
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')
```

```
cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
```

```
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
```

```
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"
```

```
# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
```

```
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
```

```
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Comece a usar CloudWatch alarmes usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um alarme.
- Desabilitar ações de alarme.
- Descrever um alarme.

- Excluir um alarme.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.

"Create an alarm"
TRY.
    lo_cwt->putmetricalarm(
      iv_alarmname                 = iv_alarm_name
      iv_comparisonoperator        = iv_comparison_operator
      iv_evaluationperiods         = iv_evaluation_periods
      iv_metricname                = iv_metric_name
      iv_namespace                 = iv_namespace
      iv_statistic                 = iv_statistic
      iv_threshold                 = iv_threshold
      iv_actionsenabled            = iv_actions_enabled
      iv_alarmdescription          = iv_alarm_description
      iv_unit                      = iv_unit
      iv_period                    = iv_period
      it_dimensions                = it_dimensions ).
    MESSAGE 'Alarm created' TYPE 'I'.
  CATCH /aws1/cx_cwtlimitexceededfault.
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.

"Create an ABAP internal table for the created alarm."
lo_alarmname = NEW #( iv_value = iv_alarm_name ).
INSERT lo_alarmname INTO TABLE lt_alarmnames.

"Disable alarm actions."
TRY.
```

```
        lo_cwt->disablealarmactions(
          it_alarmnames                = lt_alarmnames ).
        MESSAGE 'Alarm actions disabled' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
        DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception-
>av_err_code }" - { lo_disablealarm_exception->av_err_msg }|.
        MESSAGE lv_disablealarm_error TYPE 'E'.
    ENDTRY.

    "Describe alarm using the same ABAP internal table."
    TRY.
        oo_result = lo_cwt->describealarms(                       " oo_result is
 returned for testing purpose "
          it_alarmnames                = lt_alarmnames ).
        MESSAGE 'Alarms retrieved' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
        DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
        MESSAGE lv_describealarms_error TYPE 'E'.
    ENDTRY.

    "Delete alarm."
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = lt_alarmnames ).
        MESSAGE 'Alarms deleted' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound.
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - DeleteAlarms
  - DescribeAlarms
  - DisableAlarmActions
  - PutMetricAlarm

Gerencie CloudWatch métricas e alarmes usando um AWS SDK

O exemplo de código a seguir mostra como:

- Crie um alarme para observar uma CloudWatch métrica.
- Inserir dados em uma métrica e acionar o alarme.
- Obter dados do alarme.
- Excluir o alarme.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as CloudWatch operações.

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
```

```
        Sends a set of data to CloudWatch for a metric. All of the data in the
set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
```

```
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over
the
                             alarm threshold before the alarm is set into an
alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the
threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
```

```
                metric_name,
            )
            raise
        else:
            return alarm


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is
given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
name)
            raise


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
period
```

```
                        must match the granularity of the metric, which depends on
                        the metric's age. For example, metrics that are older than
                        three hours have a one-minute granularity, so the period
must
                        be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
value
                            or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
            raise
        else:
            return stats


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
metric_name)
        return alarm_iter


    def delete_metric_alarms(self, metric_namespace, metric_name):
```

```
        """
        Deletes all of the alarms that are currently watching the specified
 metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
 metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

Use a classe criada para colocar dados em uma métrica, acionar um alarme que observa a métrica e obter dados do alarme.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
    print("-" * 88)

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

    cw_wrapper = CloudWatchWrapper(boto3.resource("cloudwatch"))

    minutes = 20
    metric_namespace = "doc-example-metric"
    metric_name = "page_views"
    start = datetime.utcnow() - timedelta(minutes=minutes)
    print(
```

```
        f"Putting data into metric {metric_namespace}.{metric_name} spanning the
"
        f"last {minutes} minutes."
    )
    for offset in range(0, minutes):
        stamp = start + timedelta(minutes=offset)
        cw_wrapper.put_metric_data_set(
            metric_namespace,
            metric_name,
            stamp,
            "Count",
            {
                "values": [
                    random.randint(bound, bound * 2)
                    for bound in range(offset + 1, offset + 11)
                ],
                "counts": [random.randint(1, offset + 1) for _ in range(10)],
            },
        )

    alarm_name = "high_page_views"
    period = 60
    eval_periods = 2
    print(f"Creating alarm {alarm_name} for metric {metric_name}.")
    alarm = cw_wrapper.create_metric_alarm(
        metric_namespace,
        metric_name,
        alarm_name,
        "Maximum",
        period,
        eval_periods,
        100,
        "GreaterThanThreshold",
    )
    print(f"Alarm ARN is {alarm.alarm_arn}.")
    print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Sending data to trigger the alarm. This requires data over the
threshold "
        f"for {eval_periods} periods of {period} seconds each."
    )
    while alarm.state_value == "INSUFFICIENT_DATA":
        print("Sending data for the metric.")
```

```
        cw_wrapper.put_metric_data(
            metric_namespace, metric_name, random.randint(100, 200), "Count"
        )
        alarm.load()
        print(f"Current alarm state is: {alarm.state_value}.")
        if alarm.state_value == "INSUFFICIENT_DATA":
            print(f"Waiting for {period} seconds...")
            time.sleep(period)
        else:
            print("Wait for a minute for eventual consistency of metric data.")
            time.sleep(period)
            if alarm.state_value == "OK":
                alarm.load()
                print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Getting data for metric {metric_namespace}.{metric_name} during
timespan "
        f"of {start} to {datetime.utcnow()} (times are UTC)."
    )
    stats = cw_wrapper.get_metric_statistics(
        metric_namespace,
        metric_name,
        start,
        datetime.utcnow(),
        60,
        ["Average", "Minimum", "Maximum"],
    )
    print(
        f"Got {len(stats['Datapoints'])} data points for metric "
        f"{metric_namespace}.{metric_name}."
    )
    pprint(sorted(stats["Datapoints"], key=lambda x: x["Timestamp"]))

    print(f"Getting alarms for metric {metric_name}.")
    alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
    for alarm in alarms:
        print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

    print(f"Deleting alarms for metric {metric_name}.")
    cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - DeleteAlarms
  - DescribeAlarmsForMetric
  - DisableAlarmActions
  - EnableAlarmActions
  - GetMetricStatistics
  - ListMetrics
  - PutMetricAlarm
  - PutMetricData

Monitore o desempenho do Amazon DynamoDB usando um AWS SDK

O exemplo de código, apresentado a seguir, mostra como configurar o uso do DynamoDB por uma aplicação para monitorar o desempenho.

Java

SDK para Java 2.x

Este exemplo mostra como configurar uma aplicação em Java para monitorar o desempenho do DynamoDB. O aplicativo envia dados métricos para CloudWatch onde você pode monitorar o desempenho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch
- DynamoDB

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
```

```
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi
```

```
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
```

```
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
```

```
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")
```

```
# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
```

```
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
```

```
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
```

```
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
```

```
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")
```

```
echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"
```

```
# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"
```

```
# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
```

```
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
experiment-template.json metric-query.json
```

```
    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile

- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar funções do Lambda para monitoramento.
- Crie um CloudWatch painel
- Adicionar uma variável de propriedade ao painel.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077
```

```
# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
```

```
    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1
 >> "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/
dev/null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
```

```
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
```

```
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
```

```
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /
dev/null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /
dev/null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a
 "$LOG_FILE"

# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
```

```
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name
 "$DASHBOARD_NAME" --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch
 dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/" | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee
 -a "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
```

```
# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME"
 --output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards

- DeleteFunction
- DeleteRole
- DetachRolePolicy
- GetDashboard
- Invoke
- PutDashboard

Exemplos de código para CloudWatch eventos usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon CloudWatch Events com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- CloudWatch Guia do usuário de eventos — Mais informações sobre CloudWatch eventos.
- CloudWatch Referência da API de eventos — Detalhes sobre todas as ações de CloudWatch eventos disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CloudWatch eventos usando AWS SDKs
  - Ações para CloudWatch eventos usando AWS SDKs
    - Use PutEventscom um AWS SDK
    - Use PutRulecom um AWS SDK
    - Use PutTargetscom um AWS SDK

Exemplos básicos de CloudWatch eventos usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon CloudWatch Events com AWS SDKs.

Exemplos

- Ações para CloudWatch eventos usando AWS SDKs
  - Use PutEventscom um AWS SDK
  - Use PutRulecom um AWS SDK
  - Use PutTargetscom um AWS SDK

Ações para CloudWatch eventos usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais de CloudWatch eventos com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API Amazon CloudWatch Events.

Exemplos

- Use PutEventscom um AWS SDK
- Use PutRulecom um AWS SDK
- Use PutTargetscom um AWS SDK

Use **`PutEvents`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PutEvents`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import
 software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <resourceArn>

                Where:
                   resourceArn - An Amazon Resource Name (ARN) related to the
 events.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String
 resourceArn) {
        try {
```

```
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\":
 \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                    .detail(EVENT_DETAILS)
                    .detailType("sampleSubmitted")
                    .resources(resourceArn)
                    .source("aws-sdk-java-cloudwatch-example")
                    .build();

            PutEventsRequest request = PutEventsRequest.builder()
                    .entries(requestEntry)
                    .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

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

- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutEventsCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";
```

```
const run = async () => {
  const command = new PutEventsCommand({
    // The list of events to send to Amazon CloudWatch Events.
    Entries: [
      {
        // The name of the application or service that is sending the event.
        Source: "my.app",

        // The name of the event that is being sent.
        DetailType: "My Custom Event",

        // The data that is sent with the event.
        Detail: JSON.stringify({ timeOfEvent: new Date().toISOString() }),
      },
    ],
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};

cwevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK para JavaScript da API.

Use **PutRule**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PutRule`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> roleArn>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  roleArn - A role ARN value (for example,
 arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWRule(cwe, ruleName, roleArn);
        cwe.close();
    }

    public static void putCWRule(CloudWatchEventsClient cwe, String ruleName,
 String roleArn) {
        try {
            PutRuleRequest request = PutRuleRequest.builder()
                    .name(ruleName)
                    .roleArn(roleArn)
                    .scheduleExpression("rate(5 minutes)")
                    .state(RuleState.ENABLED)
                    .build();

            PutRuleResponse response = cwe.putRule(request);
            System.out.printf(
                    "Successfully created CloudWatch events rule %s with arn %s",
                    roleArn, response.ruleArn());

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

- Para obter detalhes da API, consulte PutRulea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutRuleCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  // Request parameters for PutRule.
  // https://docs.aws.amazon.com/eventbridge/latest/APIReference/
API_PutRule.html#API_PutRule_RequestParameters
  const command = new PutRuleCommand({
    Name: process.env.CLOUDWATCH_EVENTS_RULE,

    // The event pattern for the rule.
    //  Example: {"source": ["my.app"]}
    EventPattern: process.env.CLOUDWATCH_EVENTS_RULE_PATTERN,

    // The state of the rule. Valid values: ENABLED, DISABLED
    State: "ENABLED",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutRulea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

cwevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutRulea Referência AWS SDK para JavaScript da API.

Use **`PutTargets`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PutTargets`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> <functionArn> <targetId>\s

                Where:
```

```
                  ruleName - A rule name (for example, myrule).
                  functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                  targetId - A target id value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
        String targetId = args[2];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWTargets(cwe, ruleName, functionArn, targetId);
        cwe.close();
    }

    public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
String functionArn, String targetId) {
        try {
            Target target = Target.builder()
                    .arn(functionArn)
                    .id(targetId)
                    .build();

            PutTargetsRequest request = PutTargetsRequest.builder()
                    .targets(target)
                    .rule(ruleName)
                    .build();

            cwe.putTargets(request);
            System.out.printf(
                    "Successfully created CloudWatch events target for rule %s",
                    ruleName);

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

```
}
```

- Para obter detalhes da API, consulte PutTargetsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutTargetsCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutTargetsCommand({
    // The name of the Amazon CloudWatch Events rule.
    Rule: process.env.CLOUDWATCH_EVENTS_RULE,

    // The targets to add to the rule.
    Targets: [
      {
        Arn: process.env.CLOUDWATCH_EVENTS_TARGET_ARN,
        // The ID of the target. Choose a unique ID for each target.
        Id: process.env.CLOUDWATCH_EVENTS_TARGET_ID,
      },
    ],
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};
```

```
export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutTargetsa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myCloudWatchEventsTarget",
    },
  ],
};
```

```
cwevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutTargetsa Referência AWS SDK para JavaScript da API.

Exemplos de código para CloudWatch registros usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon CloudWatch Logs com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- CloudWatch Guia do usuário de registros — Mais informações sobre CloudWatch registros.
- CloudWatch Referência da API de registros — detalhes sobre todas as ações de CloudWatch registros disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CloudWatch registros usando AWS SDKs
  - Ações para CloudWatch registros usando AWS SDKs
    - Use AssociateKmsKeycom um AWS SDK
    - Use CancelExportTaskcom um AWS SDK
    - Use CreateExportTaskcom um AWS SDK
    - Use CreateLogGroupcom um AWS SDK ou CLI
    - Use CreateLogStreamcom um AWS SDK ou CLI
    - Use DeleteLogGroupcom um AWS SDK ou CLI
    - Use DeleteSubscriptionFiltercom um AWS SDK
    - Use DescribeExportTaskscom um AWS SDK
    - Use DescribeLogGroupscom um AWS SDK ou CLI
    - Use DescribeLogStreamscom um AWS SDK ou CLI
    - Use DescribeSubscriptionFilterscom um AWS SDK
    - Use GetLogEventscom um AWS SDK ou CLI
    - Use GetQueryResultscom um AWS SDK
    - Use PutSubscriptionFiltercom um AWS SDK
    - Use StartLiveTailcom um AWS SDK
    - Use StartQuerycom um AWS SDK
- Cenários para CloudWatch registros usando AWS SDKs
  - Configurar a conectividade do serviço de contêiner
  - Como criar sua primeira função com tecnologia sem servidor
  - Use CloudWatch registros para executar uma consulta grande
  - Usar eventos programados para chamar uma função do Lambda

Exemplos básicos de CloudWatch registros usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon CloudWatch Logs com AWS SDKs.

Exemplos

- Ações para CloudWatch registros usando AWS SDKs
  - Use AssociateKmsKeycom um AWS SDK

- Use CancelExportTaskcom um AWS SDK
- Use CreateExportTaskcom um AWS SDK
- Use CreateLogGroupcom um AWS SDK ou CLI
- Use CreateLogStreamcom um AWS SDK ou CLI
- Use DeleteLogGroupcom um AWS SDK ou CLI
- Use DeleteSubscriptionFiltercom um AWS SDK
- Use DescribeExportTaskscom um AWS SDK
- Use DescribeLogGroupscom um AWS SDK ou CLI
- Use DescribeLogStreamscom um AWS SDK ou CLI
- Use DescribeSubscriptionFilterscom um AWS SDK
- Use GetLogEventscom um AWS SDK ou CLI
- Use GetQueryResultscom um AWS SDK
- Use PutSubscriptionFiltercom um AWS SDK
- Use StartLiveTailcom um AWS SDK
- Use StartQuerycom um AWS SDK

Ações para CloudWatch registros usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do CloudWatch Logs com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API CloudWatch Logs e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para CloudWatch registros usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API Amazon CloudWatch Logs.

Exemplos

- Use AssociateKmsKeycom um AWS SDK
- Use CancelExportTaskcom um AWS SDK
- Use CreateExportTaskcom um AWS SDK
- Use CreateLogGroupcom um AWS SDK ou CLI

- Use CreateLogStreamcom um AWS SDK ou CLI
- Use DeleteLogGroupcom um AWS SDK ou CLI
- Use DeleteSubscriptionFiltercom um AWS SDK
- Use DescribeExportTaskscom um AWS SDK
- Use DescribeLogGroupscom um AWS SDK ou CLI
- Use DescribeLogStreamscom um AWS SDK ou CLI
- Use DescribeSubscriptionFilterscom um AWS SDK
- Use GetLogEventscom um AWS SDK ou CLI
- Use GetQueryResultscom um AWS SDK
- Use PutSubscriptionFiltercom um AWS SDK
- Use StartLiveTailcom um AWS SDK
- Use StartQuerycom um AWS SDK

Use **`AssociateKmsKey`**com um AWS SDK

O código de exemplo a seguir mostra como usar `AssociateKmsKey`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to associate an AWS Key Management Service (AWS KMS) key with
/// an Amazon CloudWatch Logs log group.
/// </summary>
```

```
    public class AssociateKmsKey
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string kmsKeyId = "arn:aws:kms:us-west-2:<account-
number>:key/7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
            string groupName = "cloudwatchlogs-example-loggroup";

            var request = new AssociateKmsKeyRequest
            {
                KmsKeyId = kmsKeyId,
                LogGroupName = groupName,
            };

            var response = await client.AssociateKmsKeyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully associated KMS key ID:
 {kmsKeyId} with log group: {groupName}.");
            }
            else
            {
                Console.WriteLine("Could not make the association between:
 {kmsKeyId} and {groupName}.");
            }
        }
    }
```

- Para obter detalhes da API, consulte AssociateKmsKeya Referência AWS SDK para .NET da API.

Use **`CancelExportTask`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CancelExportTask`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to cancel an Amazon CloudWatch Logs export task.
/// </summary>
public class CancelExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskId = "exampleTaskId";

        var request = new CancelExportTaskRequest
        {
            TaskId = taskId,
        };

        var response = await client.CancelExportTaskAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"{taskId} successfully canceled.");
        }
        else
        {
```

```
                Console.WriteLine($"{taskId} could not be canceled.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CancelExportTaska Referência AWS SDK para .NET da API.

Use **`CreateExportTask`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateExportTask`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

    /// <summary>
    /// Shows how to create an Export Task to export the contents of the Amazon
    /// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon
S3)
    /// bucket.
    /// </summary>
    public class CreateExportTask
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
```

```
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskName = "export-task-example";
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string destination = "amzn-s3-demo-bucket";
            var fromTime = 1437584472382;
            var toTime = 1437584472833;

            var request = new CreateExportTaskRequest
            {
                From = fromTime,
                To = toTime,
                TaskName = taskName,
                LogGroupName = logGroupName,
                Destination = destination,
            };

            var response = await client.CreateExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The task, {taskName} with ID: " +
                                  $"{response.TaskId} has been created
successfully.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateExportTaska Referência AWS SDK para .NET da API.

Use **`CreateLogGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateLogGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Configurar a conectividade do serviço de contêiner

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs log group.
/// </summary>
public class CreateLogGroup
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new CreateLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.CreateLogGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully create log group with ID:
{logGroupName}.");
        }
```

```
            else
            {
                Console.WriteLine("Could not create log group.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateLogGroupa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando cria um grupo de logs chamado `my-logs`:

```
aws logs create-log-group --log-group-name my-logs
```

- Para obter detalhes da API, consulte CreateLogGroupem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new CreateLogGroupCommand({
    // The name of the log group.
```

```
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte CreateLogGroupa Referência AWS SDK para JavaScript da API.

Use **`CreateLogStream`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateLogStream`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

    /// <summary>
    /// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
    /// log group.
    /// </summary>
    public class CreateLogStream
    {
```

```
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string logStreamName = "cloudwatchlogs-example-logstream";

            var request = new CreateLogStreamRequest
            {
                LogGroupName = logGroupName,
                LogStreamName = logStreamName,
            };

            var response = await client.CreateLogStreamAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{logStreamName} successfully created for
{logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create stream.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateLogStreama Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando cria um fluxo de logs `20150601` no grupo de logs `my-logs`:

```
aws logs create-log-stream --log-group-name my-logs --log-stream-name 20150601
```

- Para obter detalhes da API, consulte CreateLogStreamem Referência de AWS CLI Comandos.

Use **`DeleteLogGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteLogGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Como criar sua primeira função com tecnologia sem servidor

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Uses the Amazon CloudWatch Logs Service to delete an existing
/// CloudWatch Logs log group.
/// </summary>
public class DeleteLogGroup
{
    public static async Task Main()
    {
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new DeleteLogGroupRequest
```

```
            {
                LogGroupName = logGroupName,
            };

            var response = await client.DeleteLogGroupAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted CloudWatch log group,
{logGroupName}.");
            }
        }
    }
```

- Para obter detalhes da API, consulte DeleteLogGroupa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando exclui um grupo de logs chamado `my-logs`:

```
aws logs delete-log-group --log-group-name my-logs
```

- Para obter detalhes da API, consulte DeleteLogGroupem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte DeleteLogGroupa Referência AWS SDK para JavaScript da API.

Use **`DeleteSubscriptionFilter`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteSubscriptionFilter`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
```

```
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Excluir o filtro de assinatura.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetLogGroupName(log_group);

        auto outcome = cwl.DeleteSubscriptionFilter(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Failed to delete CloudWatch log subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        } else {
            std::cout << "Successfully deleted CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- Para obter detalhes da API, consulte DeleteSubscriptionFiltera Referência AWS SDK para C ++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <logGroup>

                Where:
                  filter - The name of the subscription filter (for example,
 MyFilter).
                  logGroup - The name of the log group. (for example, testgroup).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
 String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
 DeleteSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .logGroupName(logGroup)
```

```
                    .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
 filter %s", filter);

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

- Para obter detalhes da API, consulte DeleteSubscriptionFiltera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteSubscriptionFilterCommand({
    // The name of the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
```

```
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte DeleteSubscriptionFiltera Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  filterName: "FILTER",
  logGroupName: "LOG_GROUP",
};

cwl.deleteSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteSubscriptionFiltera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- Para obter detalhes da API, consulte a DeleteSubscriptionFilterreferência da API AWS SDK for Kotlin.

Use **`DescribeExportTasks`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeExportTasks`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks.
/// </summary>
public class DescribeExportTasks
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeExportTasksRequest
        {
            Limit = 5,
        };

        var response = new DescribeExportTasksResponse();

        do
        {
            response = await client.DescribeExportTasksAsync(request);
            response.ExportTasks.ForEach(t =>
            {
```

```
                    Console.WriteLine($"{t.TaskName} with ID: {t.TaskId} has
status: {t.Status}");
                });
            }
            while (response.NextToken is not null);
        }
    }
```

- Para obter detalhes da API, consulte DescribeExportTasksa Referência AWS SDK para .NET da API.

Use **`DescribeLogGroups`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeLogGroups`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Retrieves information about existing Amazon CloudWatch Logs log groups
/// and displays the information on the console.
/// </summary>
public class DescribeLogGroups
{
    public static async Task Main()
    {
        // Creates a CloudWatch Logs client using the default
```

```
            // user. If you need to work with resources in another
            // AWS Region than the one defined for the default user,
            // pass the AWS Region as a parameter to the client constructor.
            var client = new AmazonCloudWatchLogsClient();

            bool done = false;
            string newToken = null;

            var request = new DescribeLogGroupsRequest
            {
                Limit = 5,
            };

            DescribeLogGroupsResponse response;

            do
            {
                if (newToken is not null)
                {
                    request.NextToken = newToken;
                }

                response = await client.DescribeLogGroupsAsync(request);

                response.LogGroups.ForEach(lg =>
                {
                    Console.WriteLine($"{lg.LogGroupName} is associated with the
key: {lg.KmsKeyId}.");
                    Console.WriteLine($"Created on:
{lg.CreationTime.Date.Date}");
                    Console.WriteLine($"Date for this group will be stored for:
{lg.RetentionInDays} days.\n");
                });

                if (response.NextToken is null)
                {
                    done = true;
                }
                else
                {
                    newToken = response.NextToken;
                }
            }
            while (!done);
```

```
        }
    }
```

- Para obter detalhes da API, consulte DescribeLogGroupsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando descreve um grupo de logs chamado `my-logs`:

```
aws logs describe-log-groups --log-group-name-prefix my-logs
```

Saída:

```
{
    "logGroups": [
        {
            "storedBytes": 0,
            "metricFilterCount": 0,
            "creationTime": 1433189500783,
            "logGroupName": "my-logs",
            "retentionInDays": 5,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:*"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeLogGroupsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  paginateDescribeLogGroups,
  CloudWatchLogsClient,
} from "@aws-sdk/client-cloudwatch-logs";

const client = new CloudWatchLogsClient({});

export const main = async () => {
  const paginatedLogGroups = paginateDescribeLogGroups({ client }, {});
  const logGroups = [];

  for await (const page of paginatedLogGroups) {
    if (page.logGroups?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
};
```

- Para obter detalhes da API, consulte DescribeLogGroupsa Referência AWS SDK para JavaScript da API.

Use **`DescribeLogStreams`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeLogStreams`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Como criar sua primeira função com tecnologia sem servidor

CLI

AWS CLI

O comando a seguir mostra todos os fluxos de log começando com o prefixo `2015` no grupo de logs `my-logs`:

```
aws logs describe-log-streams --log-group-name my-logs --log-stream-name-
prefix 2015
```

Saída:

```
{
    "logStreams": [
        {
            "creationTime": 1433189871774,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150531",
            "logStreamName": "20150531",
            "storedBytes": 0
        },
        {
            "creationTime": 1433189873898,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150601",
            "logStreamName": "20150601",
            "storedBytes": 0
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeLogStreamsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Pesquisa fluxos de log em um grupo de logs especificado que correspondam a um determinado prefixo.

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

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

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 WeathertopJavaContainerLogs).
                  logStreamName - The name of the log stream (for example,
 weathertop-java-stream).
                  pattern - the pattern to use (for example, INFO)

                """;

        if (args.length != 3) {
            System.out.print(usage);
```

```
            System.exit(1);
        }

        String logGroupName = args[0] ;
        String logStreamName = args[1] ;
        String pattern = args[2] ;

        CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
pattern);
    }

    /**
     * Searches for log streams with a specific prefix within a log group and
filters log events based on a specified pattern.
     *
     * @param cwlClient       the CloudWatchLogsClient used to interact with AWS
CloudWatch Logs
     * @param logGroupName    the name of the log group to search within
     * @param logStreamPrefix the prefix of the log streams to search for
     * @param pattern         the pattern to filter log events by
     */
    public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
        DescribeLogStreamsRequest describeLogStreamsRequest =
DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .logStreamNamePrefix(logStreamPrefix)
                .build();

        DescribeLogStreamsResponse describeLogStreamsResponse =
cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
FilterLogEventsRequest.builder()
                    .logGroupName(logGroupName)
```

```
                    .logStreamNames(logStreamName)
                    .filterPattern(pattern)
                    .build();

            FilterLogEventsResponse filterLogEventsResponse =
 cwlClient.filterLogEvents(filterLogEventsRequest);

            for (FilteredLogEvent event : filterLogEventsResponse.events()) {
                System.out.println(event.message());
            }


 System.out.println("--------------------------------------------------"); //
 Separator for better readability
        }
    }
}
```

Imprime metadados sobre o fluxo de logs mais recente em um grupo de logs especificado.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = """
                Usage:
                  <logGroupName>

                Where:
                  logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
                """;

        if (args.length != 1) {
```

```
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
        CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeMostRecentLogStream(logsClient, logGroupName);
    }

    /**
     * Describes and prints metadata about the most recent log stream in the
specified log group.
     *
     * @param logsClient   the CloudWatchLogsClient used to interact with AWS
CloudWatch Logs
     * @param logGroupName the name of the log group
     */
    public static void describeMostRecentLogStream(CloudWatchLogsClient
logsClient, String logGroupName) {
        DescribeLogStreamsRequest streamsRequest =
DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .orderBy(OrderBy.LAST_EVENT_TIME)
                .descending(true)
                .limit(1)
                .build();

        try {
            DescribeLogStreamsResponse streamsResponse =
logsClient.describeLogStreams(streamsRequest);
            List<LogStream> logStreams = streamsResponse.logStreams();

            if (logStreams.isEmpty()) {
                System.out.println("No log streams found for log group: " +
logGroupName);
                return;
            }

            LogStream stream = logStreams.get(0);
            System.out.println("Most Recent Log Stream:");
            System.out.println("  Name: " + stream.logStreamName());
```

```
            System.out.println("  ARN: " + stream.arn());
            System.out.println("  Creation Time: " + stream.creationTime());
            System.out.println("  First Event Time: " +
 stream.firstEventTimestamp());
            System.out.println("  Last Event Time: " +
 stream.lastEventTimestamp());
            System.out.println("  Stored Bytes: " + stream.storedBytes());
            System.out.println("  Upload Sequence Token: " +
 stream.uploadSequenceToken());

        } catch (CloudWatchLogsException e) {
            System.err.println("Failed to describe log stream: " +
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeLogStreamsa Referência AWS SDK for Java 2.x da API.

Use **`DescribeSubscriptionFilters`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeSubscriptionFilters`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
```

```
#include <iostream>
#include <iomanip>
```

Liste os filtros de assinatura.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
        request.SetLogGroupName(log_group);
        request.SetLimit(1);

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = cwl.DescribeSubscriptionFilters(
                    request);
            if (!outcome.IsSuccess()) {
                std::cout << "Failed to describe CloudWatch subscription filters
 "
                    << "for log group " << log_group << ": " <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(32) << "Name" <<
                    std::setw(64) << "FilterPattern" << std::setw(64) <<
                    "DestinationArn" << std::endl;
                header = true;
            }

            const auto &filters = outcome.GetResult().GetSubscriptionFilters();
            for (const auto &filter : filters) {
                std::cout << std::left << std::setw(32) <<
                    filter.GetFilterName() << std::setw(64) <<
                    filter.GetFilterPattern() << std::setw(64) <<
                    filter.GetDestinationArn() << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Para obter detalhes da API, consulte DescribeSubscriptionFiltersa Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroup>
```

```
                Where:
                  logGroup - A log group name (for example, myloggroup).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        describeFilters(logs, logGroup);
        logs.close();
    }

    public static void describeFilters(CloudWatchLogsClient logs, String
logGroup) {
        try {
            boolean done = false;
            String newToken = null;

            while (!done) {
                DescribeSubscriptionFiltersResponse response;
                if (newToken == null) {
                    DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                            .logGroupName(logGroup)
                            .limit(1).build();

                    response = logs.describeSubscriptionFilters(request);
                } else {
                    DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                            .nextToken(newToken)
                            .logGroupName(logGroup)
                            .limit(1).build();
                    response = logs.describeSubscriptionFilters(request);
                }

                for (SubscriptionFilter filter : response.subscriptionFilters())
{
```

```
                    System.out.printf("Retrieved filter with name %s, " +
 "pattern %s " + "and destination arn %s",
                            filter.filterName(),
                            filter.filterPattern(),
                            filter.destinationArn());
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Done");
    }
}
```

- Para obter detalhes da API, consulte DescribeSubscriptionFiltersa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
```

```
  // This will return a list of all subscription filters in your account
  // matching the log group name.
  const command = new DescribeSubscriptionFiltersCommand({
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
    limit: 1,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte DescribeSubscriptionFiltersa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  logGroupName: "GROUP_NAME",
  limit: 5,
};

cwl.describeSubscriptionFilters(params, function (err, data) {
  if (err) {
```

```
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeSubscriptionFiltersa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { cwlClient
 ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSubscriptionFiltersreferência da API AWS SDK for Kotlin.

Use **`GetLogEvents`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetLogEvents`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Como criar sua primeira função com tecnologia sem servidor

CLI

AWS CLI

O seguinte comando recupera eventos de logs de um fluxo de logs chamado `20150601` no grupo de logs `my-logs`:

```
aws logs get-log-events --log-group-name my-logs --log-stream-name 20150601
```

Saída:

```
{
    "nextForwardToken":
 "f/31961209122447488583055879464742346735121166569214640130",
    "events": [
        {
            "ingestionTime": 1433190494190,
            "timestamp": 1433190184356,
            "message": "Example Event 1"
        },
        {
            "ingestionTime": 1433190516679,
            "timestamp": 1433190184356,
            "message": "Example Event 1"
        },
        {
            "ingestionTime": 1433190494190,
            "timestamp": 1433190184358,
            "message": "Example Event 2"
```

```
        }
    ],
    "nextBackwardToken":
 "b/31961209122358285602261756944988674324553373268216709120"
}
```

- Para obter detalhes da API, consulte GetLogEventsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

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

```
    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 myloggroup).
                  logStreamName - The name of the log stream (for example,
 mystream).

                """;

       // if (args.length != 2) {
       //     System.out.print(usage);
       //     System.exit(1);
//        }

        String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
        String logStreamName = "weathertop-java-stream" ; //args[1];

        Region region = Region.US_EAST_1 ;
        CloudWatchLogsClient cloudWatchLogsClient =
 CloudWatchLogsClient.builder()
                .region(region)
                .build();

        getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
        cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                      String logGroupName,
                                      String logStreamPrefix) {
        try {
            // First, find the exact log stream name
            DescribeLogStreamsRequest describeRequest =
 DescribeLogStreamsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNamePrefix(logStreamPrefix)
                    .limit(1) // get the first matching stream
                    .build();
```

```
            DescribeLogStreamsResponse describeResponse =
 cloudWatchLogsClient.describeLogStreams(describeRequest);

            if (describeResponse.logStreams().isEmpty()) {
                System.out.println("No matching log streams found for prefix: " +
 logStreamPrefix);
                return;
            }

            String exactLogStreamName =
 describeResponse.logStreams().get(0).logStreamName();
            System.out.println("Using exact log stream: " + exactLogStreamName);

            long startTime = Instant.now().minus(7,
 ChronoUnit.DAYS).toEpochMilli();
            long endTime = Instant.now().toEpochMilli();

            GetLogEventsRequest getLogEventsRequest =
 GetLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamName(exactLogStreamName) // <-- exact name, not
 prefix
                    .startTime(startTime)
                    .endTime(endTime)
                    .startFromHead(true)
                    .build();

            GetLogEventsResponse response =
 cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

            if (response.events().isEmpty()) {
                System.out.println("No log events found in the past 7 days.");
            } else {
                response.events().forEach(e -> System.out.println(e.message()));
            }

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

- Para obter detalhes da API, consulte GetLogEventsa Referência AWS SDK for Java 2.x da API.

Use **`GetQueryResults`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetQueryResults`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Executar uma consulta grande

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets the results of a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="queryId">The ID of the query.</param>
    /// <returns>The query results response.</returns>
    public async Task<GetQueryResultsResponse?> GetQueryResultsAsync(string
queryId)
    {
        try
        {
            var request = new GetQueryResultsRequest
            {
                QueryId = queryId
            };

            var response = await
_amazonCloudWatchLogs.GetQueryResultsAsync(request);
```

```
            return response;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Query not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while getting query results:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte GetQueryResultsa Referência AWS SDK para .NET da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
  return this.client.send(new GetQueryResultsCommand({ queryId }));
}
```

- Para obter detalhes da API, consulte GetQueryResultsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- Para obter detalhes da API, consulte a GetQueryResultsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_cwl->getqueryresults(
         iv_queryid = iv_query_id ).

       " Display query status and result count
       DATA(lv_status) = oo_result->get_status( ).
       DATA(lt_results) = oo_result->get_results( ).
       DATA(lv_result_count) = lines( lt_results ).

       MESSAGE |Query status: { lv_status }. Retrieved { lv_result_count } log
event(s).| TYPE 'I'.
     CATCH /aws1/cx_cwlinvalidparameterex.
       MESSAGE 'Invalid parameter.' TYPE 'E'.
     CATCH /aws1/cx_cwlresourcenotfoundex.
       MESSAGE 'Resource not found.' TYPE 'E'.
     CATCH /aws1/cx_cwlserviceunavailex.
       MESSAGE 'Service unavailable.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetQueryResultsreferência da API AWS SDK for SAP ABAP.

Use **`PutSubscriptionFilter`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PutSubscriptionFilter`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Crie o filtro de assinatura.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetFilterPattern(filter_pattern);
        request.SetLogGroupName(log_group);
        request.SetDestinationArn(dest_arn);
        auto outcome = cwl.PutSubscriptionFilter(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch logs subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- Para obter detalhes da API, consulte PutSubscriptionFiltera Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 *
 * Also, before running this Java V2 code example,set up your development
 * environment,including your credentials.
 *
 * For more information,see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */

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

                Usage:
                  <filter> <pattern> <logGroup> <functionArn>\s

                Where:
                  filter - A filter name (for example, myfilter).
                  pattern - A filter pattern (for example, ERROR).
                  logGroup - A log group name (testgroup).
                  functionArn - An AWS Lambda function ARN (for example,
 arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
            String filter,
            String pattern,
            String logGroup,
            String functionArn) {
```

```
        try {
            PutSubscriptionFilterRequest request =
 PutSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .filterPattern(pattern)
                    .logGroupName(logGroup)
                    .destinationArn(functionArn)
                    .build();

            cwl.putSubscriptionFilter(request);
            System.out.printf(
                    "Successfully created CloudWatch logs subscription filter
 %s",
                    filter);

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

- Para obter detalhes da API, consulte PutSubscriptionFiltera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { PutSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutSubscriptionFilterCommand({
```

```
    // An ARN of a same-account Kinesis stream, Kinesis Firehose
    // delivery stream, or Lambda function.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
SubscriptionFilters.html
    destinationArn: process.env.CLOUDWATCH_LOGS_DESTINATION_ARN,

    // A name for the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,

    // A filter pattern for subscribing to a filtered stream of log events.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
FilterAndPatternSyntax.html
    filterPattern: process.env.CLOUDWATCH_LOGS_FILTER_PATTERN,

    // The name of the log group. Messages in this group matching the filter
 pattern
    // will be sent to the destination ARN.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte PutSubscriptionFiltera Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
```

```
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  destinationArn: "LAMBDA_FUNCTION_ARN",
  filterName: "FILTER_NAME",
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP",
};

cwl.putSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutSubscriptionFiltera Referência AWS SDK para JavaScript da API.

Use **`StartLiveTail`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartLiveTail`.

.NET

SDK para .NET

Inclua os arquivos necessários.

```
using Amazon;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

Inicie a sessão do Live Tail.

```
            var client = new AmazonCloudWatchLogsClient();
            var request = new StartLiveTailRequest
            {
                LogGroupIdentifiers = logGroupIdentifiers,
                LogStreamNames = logStreamNames,
                LogEventFilterPattern = filterPattern,
            };

            var response = await client.StartLiveTailAsync(request);

            // Catch if request fails
            if (response.HttpStatusCode != System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Failed to start live tail session");
                return;
            }
```

Você pode lidar com os eventos da sessão do Live Tail de duas maneiras:

```
            /* Method 1
            * 1). Asynchronously loop through the event stream
            * 2). Set a timer to dispose the stream and stop the Live Tail
  session at the end.
            */
            var eventStream = response.ResponseStream;
            var task = Task.Run(() =>
            {
                foreach (var item in eventStream)
                {
                    if (item is LiveTailSessionUpdate liveTailSessionUpdate)
                    {
                        foreach (var sessionResult in
  liveTailSessionUpdate.SessionResults)
                        {
                            Console.WriteLine("Message : {0}",
  sessionResult.Message);
                        }
                    }
                    if (item is LiveTailSessionStart)
                    {
```

```
                        Console.WriteLine("Live Tail session started");
                    }
                    // On-stream exceptions are processed here
                    if (item is CloudWatchLogsEventStreamException)
                    {
                        Console.WriteLine($"ERROR: {item}");
                    }
                }
            });
            // Close the stream to stop the session after a timeout
            if (!task.Wait(TimeSpan.FromSeconds(10))){
                eventStream.Dispose();
                Console.WriteLine("End of line");
            }
```

```
            /* Method 2
            * 1). Add event handlers to each event variable
            * 2). Start processing the stream and wait for a timeout using
AutoResetEvent
            */
            AutoResetEvent endEvent = new AutoResetEvent(false);
            var eventStream = response.ResponseStream;
            using (eventStream) // automatically disposes the stream to stop the
session after execution finishes
            {
                eventStream.SessionStartReceived += (sender, e) =>
                {
                    Console.WriteLine("LiveTail session started");
                };
                eventStream.SessionUpdateReceived += (sender, e) =>
                {
                    foreach (LiveTailSessionLogEvent logEvent in
e.EventStreamEvent.SessionResults){
                        Console.WriteLine("Message: {0}", logEvent.Message);
                    }
                };
                // On-stream exceptions are captured here
                eventStream.ExceptionReceived += (sender, e) =>
                {
                    Console.WriteLine($"ERROR:
{e.EventStreamException.Message}");
                };
```

```
                eventStream.StartProcessing();
                // Stream events for this amount of time.
                endEvent.WaitOne(TimeSpan.FromSeconds(10));
                Console.WriteLine("End of line");
            }
```

- Para obter detalhes da API, consulte StartLiveTaila Referência AWS SDK para .NET da API.

Go

SDK para Go V2

Inclua os arquivos necessários.

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

Gerencie os eventos da sessão do Live Tail.

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
 eventsChan := stream.Events()
 for {
  event := <-eventsChan
  switch e := event.(type) {
  case *types.StartLiveTailResponseStreamMemberSessionStart:
   log.Println("Received SessionStart event")
  case *types.StartLiveTailResponseStreamMemberSessionUpdate:
   for _, logEvent := range e.Value.SessionResults {
    log.Println(*logEvent.Message)
   }
  default:
```

```
    // Handle on-stream exceptions
    if err := stream.Err(); err != nil {
     log.Fatalf("Error occured during streaming: %v", err)
    } else if event == nil {
     log.Println("Stream is Closed")
     return
    } else {
     log.Fatalf("Unknown event type: %T", e)
    }
   }
  }
 }
```

Inicie a sessão do Live Tail.

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
 panic("configuration error, " + err.Error())
}
client := cloudwatchlogs.NewFromConfig(cfg)

request := &cloudwatchlogs.StartLiveTailInput{
 LogGroupIdentifiers:   logGroupIdentifiers,
 LogStreamNames:        logStreamNames,
 LogEventFilterPattern: logEventFilterPattern,
}

response, err := client.StartLiveTail(context.TODO(), request)
// Handle pre-stream Exceptions
if err != nil {
 log.Fatalf("Failed to start streaming: %v", err)
}

// Start a Goroutine to handle events over stream
stream := response.GetStream()
go handleEventStreamAsync(stream)
```

Interrompa a sessão do Live Tail após um período decorrido.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
```

```
 stream.Close()
 log.Println("Event stream closed")
```

- Para obter detalhes da API, consulte StartLiveTaila Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Inclua os arquivos necessários.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Gerencie os eventos da sessão do Live Tail.

```
    private static StartLiveTailResponseHandler
 getStartLiveTailResponseStreamHandler(
            AtomicReference<Subscription> subscriptionAtomicReference) {
        return StartLiveTailResponseHandler.builder()
            .onResponse(r -> System.out.println("Received initial response"))
```

```
            .onError(throwable -> {
                CloudWatchLogsException e = (CloudWatchLogsException)
throwable.getCause();
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            })
            .subscriber(() -> new FlowableSubscriber<>() {
                @Override
                public void onSubscribe(@NonNull Subscription s) {
                    subscriptionAtomicReference.set(s);
                    s.request(Long.MAX_VALUE);
                }

                @Override
                public void onNext(StartLiveTailResponseStream event) {
                    if (event instanceof LiveTailSessionStart) {
                        LiveTailSessionStart sessionStart =
(LiveTailSessionStart) event;
                        System.out.println(sessionStart);
                    } else if (event instanceof LiveTailSessionUpdate) {
                        LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                        List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                        logEvents.forEach(e -> {
                            long timestamp = e.timestamp();
                            Date date = new Date(timestamp);
                            System.out.println("[" + date + "] " + e.message());
                        });
                    } else {
                        throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                    }
                }

                @Override
                public void onError(Throwable throwable) {
                    System.out.println(throwable.getMessage());
                    System.exit(1);
                }

                @Override
                public void onComplete() {
                    System.out.println("Completed Streaming Session");
```

```
                }
            })
            .build();
    }
```

Inicie a sessão do Live Tail.

```
        CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
                CloudWatchLogsAsyncClient.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();

        StartLiveTailRequest request =
                StartLiveTailRequest.builder()
                    .logGroupIdentifiers(logGroupIdentifiers)
                    .logStreamNames(logStreamNames)
                    .logEventFilterPattern(logEventFilterPattern)
                    .build();

        /* Create a reference to store the subscription */
        final AtomicReference<Subscription> subscriptionAtomicReference = new
 AtomicReference<>(null);

        cloudWatchLogsAsyncClient.startLiveTail(request,
 getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

Interrompa a sessão do Live Tail após um período decorrido.

```
        /* Set a timeout for the session and cancel the subscription. This will:
         * 1). Close the stream
         * 2). Stop the Live Tail session
         */
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        if (subscriptionAtomicReference.get() != null) {
            subscriptionAtomicReference.get().cancel();
            System.out.println("Subscription to stream closed");
        }
```

- Para obter detalhes da API, consulte StartLiveTaila Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Inclua os arquivos necessários.

```
import { CloudWatchLogsClient, StartLiveTailCommand } from "@aws-sdk/client-
cloudwatch-logs";
```

Gerencie os eventos da sessão do Live Tail.

```
async function handleResponseAsync(response) {
    try {
      for await (const event of response.responseStream) {
        if (event.sessionStart !== undefined) {
          console.log(event.sessionStart);
        } else if (event.sessionUpdate !== undefined) {
          for (const logEvent of event.sessionUpdate.sessionResults) {
            const timestamp = logEvent.timestamp;
            const date = new Date(timestamp);
            console.log("[" + date + "] " + logEvent.message);
          }
        } else {
            console.error("Unknown event type");
        }
      }
    } catch (err) {
        // On-stream exceptions are captured here
        console.error(err)
    }
}
```

Inicie a sessão do Live Tail.

```
    const client = new CloudWatchLogsClient();
```

```
    const command = new StartLiveTailCommand({
        logGroupIdentifiers: logGroupIdentifiers,
        logStreamNames: logStreamNames,
        logEventFilterPattern: filterPattern
    });
    try{
        const response = await client.send(command);
        handleResponseAsync(response);
    } catch (err){
        // Pre-stream exceptions are captured here
        console.log(err);
    }
```

Interrompa a sessão do Live Tail após um período decorrido.

```
    /* Set a timeout to close the client. This will stop the Live Tail session.
 */
    setTimeout(function() {
        console.log("Client timeout");
        client.destroy();
      }, 10000);
```

- Para obter detalhes da API, consulte StartLiveTaila Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

Inclua os arquivos necessários.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Inicie a sessão do Live Tail.

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
        logEventFilterPattern = logEventFilterPatternVal
    }

    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is
StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- Para obter detalhes da API, consulte a StartLiveTailreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

Inclua os arquivos necessários.

```
import boto3
import time
from datetime import datetime
```

Inicie a sessão do Live Tail.

```
    # Initialize the client
    client = boto3.client('logs')

    start_time = time.time()

    try:
        response = client.start_live_tail(
            logGroupIdentifiers=log_group_identifiers,
            logStreamNames=log_streams,
            logEventFilterPattern=filter_pattern
        )
        event_stream = response['responseStream']
        # Handle the events streamed back in the response
        for event in event_stream:
            # Set a timeout to close the stream.
            # This will end the Live Tail session.
            if (time.time() - start_time >= 10):
                event_stream.close()
                break
            # Handle when session is started
            if 'sessionStart' in event:
                session_start_event = event['sessionStart']
                print(session_start_event)
            # Handle when log event is given in a session update
            elif 'sessionUpdate' in event:
                log_events = event['sessionUpdate']['sessionResults']
                for log_event in log_events:
                    print('[{date}]
 {log}'.format(date=datetime.fromtimestamp(log_event['timestamp']/1000),log=log_event['me
            else:
```

```
                # On-stream exceptions are captured here
                raise RuntimeError(str(event))
    except Exception as e:
        print(e)
```

- Para obter detalhes da API, consulte a StartLiveTailReferência da API AWS SDK for Python (Boto3).

Use **`StartQuery`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartQuery`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Executar uma consulta grande

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Starts a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="logGroupName">The name of the log group to query.</param>
    /// <param name="queryString">The CloudWatch Logs Insights query string.</
param>
    /// <param name="startTime">The start time for the query (seconds since
 epoch).</param>
    /// <param name="endTime">The end time for the query (seconds since epoch).</
param>
    /// <param name="limit">The maximum number of results to return.</param>
    /// <returns>The query ID if successful, null otherwise.</returns>
```

```
    public async Task<string?> StartQueryAsync(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit = 10000)
    {
        try
        {
            var request = new StartQueryRequest
            {
                LogGroupName = logGroupName,
                QueryString = queryString,
                StartTime = startTime,
                EndTime = endTime,
                Limit = limit
            };

            var response = await _amazonCloudWatchLogs.StartQueryAsync(request);
            return response.QueryId;
        }
        catch (InvalidParameterException ex)
        {
            _logger.LogError($"Invalid parameter for query: {ex.Message}");
            return null;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Log group not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while starting query:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte StartQuerya Referência AWS SDK para .NET da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
      new StartQueryCommand({
        logGroupNames: this.logGroupNames,
        queryString: "fields @timestamp, @message | sort @timestamp asc",
        startTime: startDate.valueOf(),
        endTime: endDate.valueOf(),
        limit: maxLogs,
      }),
    );
  } catch (err) {
    /** @type {string} */
    const message = err.message;
    if (message.startsWith("Query's end date and time")) {
      // This error indicates that the query's start or end date occur
      // before the log group was created.
      throw new DateOutOfBoundsError(message);
    }

    throw err;
  }
}
```

- Para obter detalhes da API, consulte StartQuerya Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
```

```
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
```

```
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- Para obter detalhes da API, consulte a StartQueryReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       " iv_log_group_name = '/aws/lambda/my-function'
       " iv_query_string = 'fields @timestamp, @message | sort @timestamp desc |
limit 20'
       " iv_start_time and iv_end_time must be in Unix epoch milliseconds (ms
since Jan 1, 1970 00:00:00 UTC)
       oo_result = lo_cwl->startquery(
         iv_loggroupname = iv_log_group_name
         iv_starttime    = iv_start_time
         iv_endtime      = iv_end_time
         iv_querystring  = iv_query_string
         iv_limit        = iv_limit ).

       " Display the query ID for tracking
       DATA(lv_query_id) = oo_result->get_queryid( ).
       MESSAGE |Query started successfully with ID: { lv_query_id }| TYPE 'I'.
     CATCH /aws1/cx_cwlinvalidparameterex.
       MESSAGE 'Invalid parameter.' TYPE 'E'.
     CATCH /aws1/cx_cwllimitexceededex.
       MESSAGE 'Limit exceeded.' TYPE 'E'.
     CATCH /aws1/cx_cwlmalformedqueryex.
```

```
        MESSAGE 'Malformed query.' TYPE 'E'.
      CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cwlserviceunavailex.
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartQueryreferência da API AWS SDK for SAP ABAP.

Cenários para CloudWatch registros usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns em CloudWatch Logs com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no CloudWatch Logs ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Configurar a conectividade do serviço de contêiner
- Como criar sua primeira função com tecnologia sem servidor
- Use CloudWatch registros para executar uma consulta grande
- Usar eventos programados para chamar uma função do Lambda

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.

- Verificar a implantação
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()
```

```
# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}
```

```
# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
```

```
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
```

```
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
```

```
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
```

```
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
```

```
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"
```

```
    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
```

```
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
```

```
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
```

```
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
```

```
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
```

```
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id
```

```
    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
the AWS Management Console."
```

```
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole

- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Como criar sua primeira função com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Lambda.
- Criar código da função.
- Criar uma função do Lambda
- Testar uma função do Lambda.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
```

```
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

###############################################################################
# Setup
###############################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

###############################################################################
# Helper functions
###############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
```

```
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function
deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
not detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
```

```
    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "==========================================="
    echo "ERROR: Script failed at $1"
    echo "==========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}
```

```
validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

##############################################################################
# Region pre-check
##############################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

##############################################################################
# Runtime selection
##############################################################################

echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
```

```
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
```

```
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

################################################################################
# Step 1: Create IAM execution role
################################################################################

echo ""
echo "==========================================="
echo "Step 1: Create IAM execution role"
echo "==========================================="
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi
```

```
echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
```

```
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "==========================================="
echo "Step 3: Invoke the function"
echo "==========================================="
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
```

```
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

##############################################################################
# Step 4: View CloudWatch logs
##############################################################################

echo ""
echo "==========================================="
echo "Step 4: View CloudWatch Logs"
echo "==========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."
```

```
LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################

echo ""
```

```
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction
  - GetFunctionConfiguration
  - GetLogEvents
  - Invoke

Use CloudWatch registros para executar uma consulta grande

Os exemplos de código a seguir mostram como usar o CloudWatch Logs para consultar mais de 10.000 registros.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o fluxo de trabalho principal que demonstra o grande cenário de consultas.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
using CloudWatchLogsActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace CloudWatchLogsScenario;

public class LargeQueryWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the CloudWatch Logs
 Large Query workflow:

    1. Prepare the Application:
       - Prompt the user to deploy CloudFormation stack and generate sample logs.
       - Deploy the CloudFormation template for resource creation.
       - Generate 50,000 sample log entries using CloudWatch Logs API.
       - Wait 5 minutes for logs to be fully ingested.

    2. Execute Large Query:
       - Perform recursive queries to retrieve all logs using binary search.
       - Display progress for each query executed.
       - Show total execution time and logs found.
```

```
    3. Clean up:
       - Prompt the user to delete the CloudFormation stack and all resources.
       - Destroy the CloudFormation stack and wait until removed.
    */

    public static ILogger<LargeQueryWorkflow> _logger = null!;
    public static CloudWatchLogsWrapper _wrapper = null!;
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _logGroupName = "/workflows/cloudwatch-logs/large-
query";
    private static string _logStreamName = "stream1";
    private static long _queryStartDate;
    private static long _queryEndDate;

    public static bool _interactive = true;
    public static string _stackName = "CloudWatchLargeQueryStack";
    private static string _stackResourcePath = "../../../../../../../scenarios/
features/cloudwatch_logs_large_query/resources/stack.yaml";

    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter("Microsoft", LogLevel.Information))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatchLogs>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<CloudWatchLogsWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<LargeQueryWorkflow>();

            _wrapper = host.Services.GetRequiredService<CloudWatchLogsWrapper>();
            _amazonCloudFormation =
 host.Services.GetRequiredService<IAmazonCloudFormation>();
        }
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Runs the scenario workflow. Used for testing.
    /// </summary>
    public static async Task RunScenario()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
```

```
    {
        Console.WriteLine("Preparing the application...");
        Console.WriteLine();

        try
        {
            var deployStack = !_interactive || GetYesNoResponse(
                "Would you like to deploy the CloudFormation stack and generate
sample logs? (y/n) ");

            if (deployStack)
            {
                if (_interactive)
                {
                    Console.Write(
                        $"Enter a path for the CloudFormation stack
resource .yaml file (or press Enter for default '{_stackResourcePath}'): ");
                    string? inputPath = Console.ReadLine();
                    if (!string.IsNullOrWhiteSpace(inputPath))
                    {
                        _stackResourcePath = inputPath;
                    }
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName);

                if (deploySuccess)
                {
                    Console.WriteLine();
                    Console.WriteLine("Generating 50,000 sample log entries...");
                    var generateSuccess = await GenerateSampleLogs();

                    if (generateSuccess)
                    {
                        Console.WriteLine();
                        Console.WriteLine("Sample logs created. Waiting 5 minutes
for logs to be fully ingested...");
                        await WaitWithCountdown(300);

                        Console.WriteLine("Application preparation complete.");
                        return true;
                    }
```

```
                }
            }
            else
            {
                _logGroupName = PromptUserForInput("Enter the log group name ",
_logGroupName);
                _logStreamName = PromptUserForInput("Enter the log stream name ",
_logStreamName);

                var startDateMs = PromptUserForLong("Enter the query start date
(milliseconds since epoch): ");
                var endDateMs = PromptUserForLong("Enter the query end date
(milliseconds since epoch): ");

                _queryStartDate = startDateMs / 1000;
                _queryEndDate = endDateMs / 1000;

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }

        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
```

```
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath)
            };

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await
WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created
successfully.");
                    return true;
                }
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName);
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }
```

```
            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Generates sample logs directly using CloudWatch Logs API.
    /// Creates 50,000 log entries spanning 5 minutes.
    /// </summary>
    /// <returns>True if logs were generated successfully.</returns>
    private static async Task<bool> GenerateSampleLogs()
    {
        const int totalEntries = 50000;
        const int entriesPerBatch = 10000;
        const int fiveMinutesMs = 5 * 60 * 1000;

        try
        {
            // Calculate timestamps
            var startTimeMs = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();
            var timestampIncrement = fiveMinutesMs / totalEntries;

            Console.WriteLine($"Generating {totalEntries} log entries...");

            var entryCount = 0;
            var currentTimestamp = startTimeMs;
            var numBatches = totalEntries / entriesPerBatch;

            // Generate and upload logs in batches
            for (int batchNum = 0; batchNum < numBatches; batchNum++)
            {
                var logEvents = new List<InputLogEvent>();

                for (int i = 0; i < entriesPerBatch; i++)
                {
                    logEvents.Add(new InputLogEvent
                    {
```

```
                        Timestamp =
 DateTimeOffset.FromUnixTimeMilliseconds(currentTimestamp).UtcDateTime,
                        Message = $"Entry {entryCount}"
                    });

                    entryCount++;
                    currentTimestamp += timestampIncrement;
                }

                // Upload batch
                var success = await _wrapper.PutLogEventsAsync(_logGroupName,
 _logStreamName, logEvents);
                if (!success)
                {
                    _logger.LogError($"Failed to upload batch {batchNum + 1}/
{numBatches}");
                    return false;
                }

                Console.WriteLine($"Uploaded batch {batchNum + 1}/{numBatches}");
            }

            // Set query date range (convert milliseconds to seconds for query
 API)
            _queryStartDate = startTimeMs / 1000;
            _queryEndDate = (currentTimestamp - timestampIncrement) / 1000;

            Console.WriteLine($"Query start date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryStartDate):yyyy-MM-
ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Query end date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryEndDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Successfully uploaded {totalEntries} log
 entries");

            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while generating sample
 logs.");
            return false;
        }
    }
```

```
    /// <summary>
    /// Executes the large query workflow.
    /// </summary>
    public static async Task ExecuteLargeQuery()
    {
        Console.WriteLine("Starting recursive query to retrieve all logs...");
        Console.WriteLine();

        var queryLimit = PromptUserForInteger("Enter the query limit (max 10000)
", 10000);
        if (queryLimit > 10000) queryLimit = 10000;

        var queryString = "fields @timestamp, @message | sort @timestamp asc";

        var stopwatch = Stopwatch.StartNew();
        var allResults = await PerformLargeQuery(_logGroupName, queryString,
_queryStartDate, _queryEndDate, queryLimit);
        stopwatch.Stop();

        Console.WriteLine();
        Console.WriteLine($"Queries finished in
{stopwatch.Elapsed.TotalSeconds:F3} seconds.");
        Console.WriteLine($"Total logs found: {allResults.Count}");

        // Check for duplicates
        Console.WriteLine();
        Console.WriteLine("Checking for duplicate logs...");
        var duplicates = FindDuplicateLogs(allResults);
        if (duplicates.Count > 0)
        {
            Console.WriteLine($"WARNING: Found {duplicates.Count} duplicate log
entries!");
            Console.WriteLine("Duplicate entries (showing first 10):");
            foreach (var dup in duplicates.Take(10))
            {
                Console.WriteLine($"  [{dup.Timestamp}] {dup.Message} (appears
{dup.Count} times)");
            }

            var uniqueCount = allResults.Count - duplicates.Sum(d => d.Count -
1);
            Console.WriteLine($"Unique logs: {uniqueCount}");
        }
```

```
        else
        {
            Console.WriteLine("No duplicates found. All logs are unique.");
        }
        Console.WriteLine();

        var viewSample = !_interactive || GetYesNoResponse("Would you like to see
a sample of the logs? (y/n) ");
        if (viewSample)
        {
            Console.WriteLine();
            Console.WriteLine($"Sample logs (first 10 of {allResults.Count}):");
            for (int i = 0; i < Math.Min(10, allResults.Count); i++)
            {
                var timestamp = allResults[i].Find(f => f.Field ==
"@timestamp")?.Value ?? "N/A";
                var message = allResults[i].Find(f => f.Field ==
"@message")?.Value ?? "N/A";
                Console.WriteLine($"[{timestamp}] {message}");
            }
        }
    }

    /// <summary>
    /// Performs a large query using recursive binary search.
    /// </summary>
    private static async Task<List<List<ResultField>>> PerformLargeQuery(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit)
    {
        var queryId = await _wrapper.StartQueryAsync(logGroupName, queryString,
startTime, endTime, limit);
        if (queryId == null)
        {
            return new List<List<ResultField>>();
        }

        var results = await PollQueryResults(queryId);
        if (results == null || results.Count == 0)
        {
            return new List<List<ResultField>>();
```

```
        }

        var startDate =
 DateTimeOffset.FromUnixTimeSeconds(startTime).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var endDate = DateTimeOffset.FromUnixTimeSeconds(endTime).ToString("yyyy-
MM-ddTHH:mm:ss.fffZ");
        Console.WriteLine($"Query date range: {startDate} ({startTime}s) to
 {endDate} ({endTime}s). Found {results.Count} logs.");

        if (results.Count < limit)
        {
            Console.WriteLine($"  -> Returning {results.Count} logs (less than
 limit of {limit})");
            return results;
        }

        Console.WriteLine($"  -> Hit limit of {limit}. Need to split and
 recurse.");

        // Get the timestamp of the last log (sorted to find the actual last one)
        var lastLogTimestamp = GetLastLogTimestamp(results);
        if (lastLogTimestamp == null)
        {
            Console.WriteLine($"  -> No timestamp found in results. Returning
 {results.Count} logs.");
            return results;
        }

        Console.WriteLine($"  -> Last log timestamp: {lastLogTimestamp}");

        // Parse the timestamp and add 1 millisecond to avoid querying the same
 log again
        var lastLogDate = DateTimeOffset.Parse(lastLogTimestamp + " +0000");
        Console.WriteLine($"  -> Last log as DateTimeOffset: {lastLogDate:yyyy-
MM-ddTHH:mm:ss.fffZ} ({lastLogDate.ToUnixTimeSeconds()}s)");

        var offsetLastLogDate = lastLogDate.AddMilliseconds(1);
        Console.WriteLine($"  -> Offset timestamp (last
 + 1ms): {offsetLastLogDate:yyyy-MM-ddTHH:mm:ss.fffZ}
 ({offsetLastLogDate.ToUnixTimeSeconds()}s)");

        // Convert to seconds, but round UP to the next second to avoid
 overlapping with logs in the same second
```

```
        // This ensures we don't re-query logs that share the same second as the
 last log
        var offsetLastLogTime = offsetLastLogDate.ToUnixTimeSeconds();
        if (offsetLastLogDate.Millisecond > 0)
        {
            offsetLastLogTime++; // Move to the next full second
            Console.WriteLine($"  -> Adjusted to next full second:
 {offsetLastLogTime}s
 ({DateTimeOffset.FromUnixTimeSeconds(offsetLastLogTime):yyyy-MM-
ddTHH:mm:ss.fffZ})");
        }

        Console.WriteLine($"  -> Comparing:
 offsetLastLogTime={offsetLastLogTime}s vs endTime={endTime}s");
        Console.WriteLine($"  -> End time as date:
 {DateTimeOffset.FromUnixTimeSeconds(endTime):yyyy-MM-ddTHH:mm:ss.fffZ}");

        // Check if there's any time range left to query
        if (offsetLastLogTime >= endTime)
        {
            Console.WriteLine($"  -> No time range left to query. Offset time
 ({offsetLastLogTime}s) >= end time ({endTime}s)");
            return results;
        }

        // Split the remaining date range in half
        var (range1Start, range1End, range2Start, range2End) =
 SplitDateRange(offsetLastLogTime, endTime);

        var range1StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range1Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range1EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range1End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range2Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range2End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");

        Console.WriteLine($"  -> Splitting remaining range:");
```

```
        Console.WriteLine($"     Range 1: {range1StartDate} ({range1Start}s) to
{range1EndDate} ({range1End}s)");
        Console.WriteLine($"     Range 2: {range2StartDate} ({range2Start}s) to
{range2EndDate} ({range2End}s)");

        // Query both halves recursively
        Console.WriteLine($"  -> Querying range 1...");
        var results1 = await PerformLargeQuery(logGroupName, queryString,
range1Start, range1End, limit);
        Console.WriteLine($"  -> Range 1 returned {results1.Count} logs");

        Console.WriteLine($"  -> Querying range 2...");
        var results2 = await PerformLargeQuery(logGroupName, queryString,
range2Start, range2End, limit);
        Console.WriteLine($"  -> Range 2 returned {results2.Count} logs");

        // Combine all results
        var allResults = new List<List<ResultField>>(results);
        allResults.AddRange(results1);
        allResults.AddRange(results2);

        Console.WriteLine($"  -> Combined total: {allResults.Count} logs
({results.Count} + {results1.Count} + {results2.Count})");

        return allResults;
    }

    /// <summary>
    /// Gets the timestamp string of the most recent log from a list of logs.
    /// Sorts timestamps to find the actual last one.
    /// </summary>
    private static string? GetLastLogTimestamp(List<List<ResultField>> logs)
    {
        var timestamps = logs
            .Select(log => log.Find(f => f.Field == "@timestamp")?.Value)
            .Where(t => !string.IsNullOrEmpty(t))
            .OrderBy(t => t)
            .ToList();

        if (timestamps.Count == 0)
        {
            return null;
        }
```

```
        return timestamps[timestamps.Count - 1];
    }

    /// <summary>
    /// Splits a date range in half.
    /// Range 2 starts at midpoint + 1 second to avoid overlap.
    /// </summary>
    private static (long range1Start, long range1End, long range2Start, long
range2End) SplitDateRange(long startTime, long endTime)
    {
        var midpoint = startTime + (endTime - startTime) / 2;
        // Range 2 starts at midpoint + 1 to avoid querying the same second twice
        return (startTime, midpoint, midpoint + 1, endTime);
    }

    /// <summary>
    /// Polls for query results until complete.
    /// </summary>
    private static async Task<List<List<ResultField>>?> PollQueryResults(string
queryId)
    {
        int retryCount = 0;
        const int maxRetries = 60;
        const int retryDelay = 1000;

        while (retryCount < maxRetries)
        {
            var response = await _wrapper.GetQueryResultsAsync(queryId);
            if (response == null)
            {
                return null;
            }

            if (response.Status == QueryStatus.Complete)
            {
                return response.Results;
            }

            if (response.Status == QueryStatus.Failed ||
                response.Status == QueryStatus.Cancelled ||
                response.Status == QueryStatus.Timeout ||
                response.Status == QueryStatus.Unknown)
            {
                _logger.LogError($"Query failed with status: {response.Status}");
```

```
                return null;
            }

            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for query results.");
        return null;
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    public static async Task<bool> Cleanup()
    {
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete the CloudFormation stack and all resources?
(y/n) ");

        if (cleanup)
        {
            try
            {
                var stackDeleteSuccess = await
DeleteCloudFormationStack(_stackName, false);
                return stackDeleteSuccess;
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "An error occurred while cleaning up the
resources.");
                return false;
            }
        }

        Console.WriteLine($"Resources will remain. Stack name: {_stackName}, Log
group: {_logGroupName}");
        _logger.LogInformation("CloudWatch Logs Large Query scenario is
complete.");
        return true;
    }

    /// <summary>
```

```
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
bool forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(stackName, forceDelete);

        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
            return true;
        }
        else
        {
            _logger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;
```

```
        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }

                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus
== StackStatus.DELETE_FAILED)
                {
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
```

```
    /// Waits with a countdown display.
    /// </summary>
    private static async Task WaitWithCountdown(int seconds)
    {
        for (int i = seconds; i > 0; i--)
        {
            Console.Write($"\rWaiting: {i} seconds remaining...  ");
            await Task.Delay(1000);
        }
        Console.WriteLine("\rWait complete.                     ");
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (_interactive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = "[a-zA-Z][-a-zA-Z0-9]*";
                if (!Regex.IsMatch(input, regex))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
{_stackName}");
                    return _stackName;
                }
                return input;
```

```
            }
        }
        return _stackName;
    }

    /// <summary>
    /// Prompts the user for input with a default value.
    /// </summary>
    private static string PromptUserForInput(string prompt, string defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'):
");
            string? input = Console.ReadLine();
            return string.IsNullOrWhiteSpace(input) ? defaultValue : input;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for an integer value.
    /// </summary>
    private static int PromptUserForInteger(string prompt, int defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'):
");
            string? input = Console.ReadLine();
            if (string.IsNullOrWhiteSpace(input) || !int.TryParse(input, out var
result))
            {
                return defaultValue;
            }
            return result;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for a long value.
    /// </summary>
    private static long PromptUserForLong(string prompt)
```

```
    {
        if (_interactive)
        {
            Console.Write(prompt);
            string? input = Console.ReadLine();
            if (long.TryParse(input, out var result))
            {
                return result;
            }
        }
        return 0;
    }

    /// <summary>
    /// Finds duplicate log entries based on timestamp and message.
    /// </summary>
    private static List<(string Timestamp, string Message, int Count)>
FindDuplicateLogs(List<List<ResultField>> logs)
    {
        var logSignatures = new Dictionary<string, int>();

        foreach (var log in logs)
        {
            var timestamp = log.Find(f => f.Field == "@timestamp")?.Value ?? "";
            var message = log.Find(f => f.Field == "@message")?.Value ?? "";
            var signature = $"{timestamp}|{message}";

            if (logSignatures.ContainsKey(signature))
            {
                logSignatures[signature]++;
            }
            else
            {
                logSignatures[signature] = 1;
            }
        }

        return logSignatures
            .Where(kvp => kvp.Value > 1)
            .Select(kvp =>
            {
                var parts = kvp.Key.Split('|');
                return (Timestamp: parts[0], Message: parts[1], Count:
kvp.Value);
```

```
            })
            .OrderByDescending(x => x.Count)
            .ToList();
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - GetQueryResults
  - StartQuery

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o ponto de entrada.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
import { CloudWatchQuery } from "./cloud-watch-query.js";

console.log("Starting a recursive query...");

if (!process.env.QUERY_START_DATE || !process.env.QUERY_END_DATE) {
  throw new Error(
    "QUERY_START_DATE and QUERY_END_DATE environment variables are required.",
  );
}

const cloudWatchQuery = new CloudWatchQuery(new CloudWatchLogsClient({}), {
  logGroupNames: ["/workflows/cloudwatch-logs/large-query"],
  dateRange: [
    new Date(Number.parseInt(process.env.QUERY_START_DATE)),
```

```
    new Date(Number.parseInt(process.env.QUERY_END_DATE)),
  ],
});

await cloudWatchQuery.run();

console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs
 found: ${cloudWatchQuery.results.length}`,
);
```

Essa é uma classe que divide as consultas em várias etapas, se necessário.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  StartQueryCommand,
  GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}

export class CloudWatchQuery {
  /**
   * Run a query for all CloudWatch Logs within a certain date range.
   * CloudWatch logs return a max of 10,000 results. This class
   * performs a binary search across all of the logs in the provided
   * date range if a query returns the maximum number of results.
   *
   * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient}
 client
   * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
 { limit: number } }} config
   */
  constructor(client, { logGroupNames, dateRange, queryConfig }) {
    this.client = client;
    /**
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;
```

```
    /**
     * The inclusive date range that is queried.
     */
    this.dateRange = dateRange;

    /**
     * CloudWatch Logs never returns more than 10,000 logs.
     */
    this.limit = queryConfig?.limit ?? 10000;

    /**
     * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
     */
    this.results = [];
  }

  /**
   * Run the query.
   */
  async run() {
    this.secondsElapsed = 0;
    const start = new Date();
    this.results = await this._largeQuery(this.dateRange);
    const end = new Date();
    this.secondsElapsed = (end - start) / 1000;
    return this.results;
  }

  /**
   * Recursively query for logs.
   * @param {[Date, Date]} dateRange
   * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[]
[]>}
   */
  async _largeQuery(dateRange) {
    const logs = await this._query(dateRange, this.limit);

    console.log(
      `Query date range: ${dateRange
        .map((d) => d.toISOString())
        .join(" to ")}. Found ${logs.length} logs.`,
    );
```

```
    if (logs.length < this.limit) {
      return logs;
    }

    const lastLogDate = this._getLastLogDate(logs);
    const offsetLastLogDate = new Date(lastLogDate);
    offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
    const subDateRange = [offsetLastLogDate, dateRange[1]];
    const [r1, r2] = splitDateRange(subDateRange);
    const results = await Promise.all([
      this._largeQuery(r1),
      this._largeQuery(r2),
    ]);
    return [logs, ...results].flat();
  }

  /**
   * Find the most recent log in a list of logs.
   * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
   */
  _getLastLogDate(logs) {
    const timestamps = logs
      .map(
        (log) =>
          log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,
      )
      .filter((t) => !!t)
      .map((t) => `${t}Z`)
      .sort();

    if (!timestamps.length) {
      throw new Error("No timestamp found in logs.");
    }

    return new Date(timestamps[timestamps.length - 1]);
  }

  /**
   * Simple wrapper for the GetQueryResultsCommand.
   * @param {string} queryId
   */
  _getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
  }
```

```
/**
 * Starts a query and waits for it to complete.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 */
async _query(dateRange, maxLogs) {
  try {
    const { queryId } = await this._startQuery(dateRange, maxLogs);
    const { results } = await this._waitUntilQueryDone(queryId);
    return results ?? [];
  } catch (err) {
    /**
     * This error is thrown when StartQuery returns an error indicating
     * that the query's start or end date occur before the log group was
     * created.
     */
    if (err instanceof DateOutOfBoundsError) {
      return [];
    }
    throw err;
  }
}

/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
      new StartQueryCommand({
        logGroupNames: this.logGroupNames,
        queryString: "fields @timestamp, @message | sort @timestamp asc",
        startTime: startDate.valueOf(),
        endTime: endDate.valueOf(),
        limit: maxLogs,
      }),
    );
  } catch (err) {
    /** @type {string} */
```

```
      const message = err.message;
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }

  /**
   * Call GetQueryResultsCommand until the query is done.
   * @param {string} queryId
   */
  _waitUntilQueryDone(queryId) {
    const getResults = async () => {
      const results = await this._getQueryResults(queryId);
      const queryDone = [
        "Complete",
        "Failed",
        "Cancelled",
        "Timeout",
        "Unknown",
      ].includes(results.status);

      return { queryDone, results };
    };

    return retry(
      { intervalInMs: 1000, maxRetries: 60, quiet: true },
      async () => {
        const { queryDone, results } = await getResults();
        if (!queryDone) {
          throw new Error("Query not done.");
        }

        return results;
      },
    );
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - GetQueryResults
  - StartQuery

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse arquivo invoca um módulo de exemplo para gerenciar CloudWatch consultas com mais de 10.000 resultados.

```
import logging
import os
import sys

import boto3
from botocore.config import Config

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"


class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
```

```
        Initializes the CloudWatchLogsQueryRunner class by setting up date
utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
        Creates and returns a CloudWatch Logs client with a specified retry
configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts":
10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and
end dates.
        Fetches the environment variable for log group, returning the default
value if it
        does not exist.

        :return: Tuple of query start date and end date as integers and the log
group.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables
are required."
            )
```

```
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)

        try:
            log_group = os.environ["QUERY_LOG_GROUP"]
        except KeyError:
            logging.warning("No QUERY_LOG_GROUP environment variable, using
default value")
            log_group = DEFAULT_QUERY_LOG_GROUP

        return query_start_date, query_end_date, log_group

    def convert_dates_to_iso8601(self, start_date, end_date):
        """
        Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

        :param start_date: The start date in UNIX timestamp.
        :type start_date: int
        :param end_date: The end date in UNIX timestamp.
        :type end_date: int
        :return: Start and end dates in ISO 8601 format.
        :rtype: tuple
        """
        start_date_iso8601 =
self.date_utilities.convert_unix_timestamp_to_iso8601(
            start_date
        )
        end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            end_date
        )
        return start_date_iso8601, end_date_iso8601

    def execute_query(
        self,
        start_date_iso8601,
        end_date_iso8601,
        log_group="/workflows/cloudwatch-logs/large-query",
        query="fields @timestamp, @message | sort @timestamp asc"
    ):
        """
        Creates a CloudWatchQuery instance and executes the query with provided
date range.
```

```
        :param start_date_iso8601: The start date in ISO 8601 format.
        :type start_date_iso8601: str
        :param end_date_iso8601: The end date in ISO 8601 format.
        :type end_date_iso8601: str
        :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
        :type log_group: str
        :param query: Query string to pass to the CloudWatchQuery instance
        :type query: str
        """
        cloudwatch_query = CloudWatchQuery(
            log_group=log_group,
            query_string=query
        )
        cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
        logging.info("Query executed successfully.")
        logging.info(
            f"Queries completed in {cloudwatch_query.query_duration} seconds.
 Total logs found: {len(cloudwatch_query.query_results)}"
        )


def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the
 query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
    runner = CloudWatchLogsQueryRunner()
    query_start_date, query_end_date, log_group =
 runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
 DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601,
 log_group=log_group)


if __name__ == "__main__":
    main()
```

Este módulo processa CloudWatch consultas com mais de 10.000 resultados.

```
import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass


class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    :log_group str: Name of the log group to query
    :query_string str: query
    """

    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
 str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000
```

```
    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and
calculates the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
        """
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n       START:     {start_date}"
            f"\n       END:       {end_date}"
            f"\n       LOG GROUP: {self.log_group}"
        )
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
recursively if necessary.

        :param date_range: The date range to fetch logs for, specified as a tuple
(start_timestamp, end_timestamp).
        :type date_range: tuple
        :return: None if the recursive fetching is continued or stops when the
final batch of logs is processed.
                 Although it doesn't explicitly return the query results, this
method accumulates all fetched logs
                 in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
```

```
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
            new_range = (most_recent_log_timestamp, date_range[1])
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()

            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.
        :return: log
        :type :return List containing log item details
        """
        most_recent_log = None
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
```

```
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
```

```
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")


    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.
```

```
        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - GetQueryResults
  - StartQuery

Usar eventos programados para chamar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

Java

SDK para Java 2.x

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma

mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função Lambda usando a API de tempo de execução do JavaScript Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento programado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função Lambda o alvo.
- Concede permissão para permitir a EventBridge invocação da função Lambda.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações programadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos de código para CodeBuild usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS CodeBuild com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- CodeBuild Guia do usuário — Mais informações sobre CodeBuild.
- CodeBuild Referência da API — Detalhes sobre todas as CodeBuild ações disponíveis.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CodeBuild uso AWS SDKs
  - Ações para CodeBuild usar AWS SDKs
    - Use CreateProjectcom um AWS SDK ou CLI
    - Use ListBuildscom um AWS SDK ou CLI
    - Use ListProjectscom um AWS SDK ou CLI
    - Use StartBuildcom um AWS SDK ou CLI

Exemplos básicos de CodeBuild uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS CodeBuild com AWS SDKs.

Exemplos

- Ações para CodeBuild usar AWS SDKs
  - Use CreateProjectcom um AWS SDK ou CLI
  - Use ListBuildscom um AWS SDK ou CLI
  - Use ListProjectscom um AWS SDK ou CLI
  - Use StartBuildcom um AWS SDK ou CLI

Ações para CodeBuild usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar CodeBuild ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS CodeBuild.

Exemplos

- Use CreateProjectcom um AWS SDK ou CLI
- Use ListBuildscom um AWS SDK ou CLI
- Use ListProjectscom um AWS SDK ou CLI
- Use StartBuildcom um AWS SDK ou CLI

Use **`CreateProject`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateProject`.

CLI

AWS CLI

Exemplo 1: Para criar um projeto de AWS CodeBuild construção

O `create-project` exemplo a seguir cria um projeto de CodeBuild compilação usando arquivos de origem de um bucket do S3

```
aws codebuild create-project \
    --name "my-demo-project" \
    --source "{\"type\": \"S3\",\"location\": \"codebuild-us-west-2-123456789012-input-bucket/my-source.zip\"}" \
    --artifacts {"\"type\": \"S3\",\"location\": \"codebuild-us-west-2-123456789012-output-bucket\""} \
    --environment "{\"type\": \"LINUX_CONTAINER\",\"image\": \"aws/codebuild/standard:1.0\",\"computeType\": \"BUILD_GENERAL1_SMALL\"}" \
    --service-role "arn:aws:iam::123456789012:role/service-role/my-codebuild-service-role"
```

Saída:

```
{
    "project": {
        "arn": "arn:aws:codebuild:us-west-2:123456789012:project/my-demo-project",
        "name": "my-cli-demo-project",
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-service-role",
        "lastModified": 1556839783.274,
        "badge": {
```

```
            "badgeEnabled": false
        },
        "queuedTimeoutInMinutes": 480,
        "environment": {
            "image": "aws/codebuild/standard:1.0",
            "computeType": "BUILD_GENERAL1_SMALL",
            "type": "LINUX_CONTAINER",
            "imagePullCredentialsType": "CODEBUILD",
            "privilegedMode": false,
            "environmentVariables": []
        },
        "artifacts": {
            "location": "codebuild-us-west-2-123456789012-output-bucket",
            "name": "my-cli-demo-project",
            "namespaceType": "NONE",
            "type": "S3",
            "packaging": "NONE",
            "encryptionDisabled": false
        },
        "source": {
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip",
            "insecureSsl": false
        },
        "timeoutInMinutes": 60,
        "cache": {
            "type": "NO_CACHE"
        },
        "created": 1556839783.274
    }
}
```

Exemplo 2: Para criar um projeto de AWS CodeBuild construção usando um arquivo de entrada JSON para os parâmetros

O `create-project` exemplo a seguir cria um projeto de CodeBuild construção passando todos os parâmetros necessários em um arquivo de entrada JSON. Crie o modelo do arquivo de entrada executando o comando somente com o `--generate-cli-skeleton parameter`.

```
aws codebuild create-project --cli-input-json file://create-project.json
```

O arquivo de entrada JSON `create-project.json` contém o seguinte conteúdo:

```
{
    "name": "codebuild-demo-project",
    "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "artifacts": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-output-bucket"
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:1.0",
        "computeType": "BUILD_GENERAL1_SMALL"
    },
    "serviceRole": "serviceIAMRole"
}
```

Saída:

```
{
    "project": {
        "name": "codebuild-demo-project",
        "serviceRole": "serviceIAMRole",
        "tags": [],
        "artifacts": {
            "packaging": "NONE",
            "type": "S3",
            "location": "codebuild-region-ID-account-ID-output-bucket",
            "name": "message-util.zip"
        },
        "lastModified": 1472661575.244,
        "timeoutInMinutes": 60,
        "created": 1472661575.244,
        "environment": {
            "computeType": "BUILD_GENERAL1_SMALL",
            "image": "aws/codebuild/standard:1.0",
            "type": "LINUX_CONTAINER",
            "environmentVariables": []
        },
        "source": {
```

```
            "type": "S3",
            "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
        },
        "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
        "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project"
    }
}
```

Para obter mais informações, consulte Criar um projeto de compilação (AWS CLI) no Guia do AWS CodeBuild usuário.

- Para obter detalhes da API, consulte CreateProjectem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um projeto.

```
import {
  ArtifactsType,
  CodeBuildClient,
  ComputeType,
  CreateProjectCommand,
  EnvironmentType,
  SourceType,
} from "@aws-sdk/client-codebuild";

// Create the AWS CodeBuild project.
export const createProject = async (
  projectName = "MyCodeBuilder",
  roleArn = "arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin",
  buildOutputBucket = "xxxx",
  githubUrl = "https://...",
```

```
) => {
  const codeBuildClient = new CodeBuildClient({});

  const response = await codeBuildClient.send(
    new CreateProjectCommand({
      artifacts: {
        // The destination of the build artifacts.
        type: ArtifactsType.S3,
        location: buildOutputBucket,
      },
      // Information about the build environment. The combination of
 "computeType" and "type" determines the
      // requirements for the environment such as CPU, memory, and disk space.
      environment: {
        // Build environment compute types.
        // https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
compute-types.html
        computeType: ComputeType.BUILD_GENERAL1_SMALL,
        // Docker image identifier.
        // See https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-
ref-available.html
        image: "aws/codebuild/standard:7.0",
        // Build environment type.
        type: EnvironmentType.LINUX_CONTAINER,
      },
      name: projectName,
      // A role ARN with permission to create a CodeBuild project, write to the
 artifact location, and write CloudWatch logs.
      serviceRole: roleArn,
      source: {
        // The type of repository that contains the source code to be built.
        type: SourceType.GITHUB,
        // The location of the repository that contains the source code to be
 built.
        location: githubUrl,
      },
    }),
  );
  console.log(response);
  //   {
  //     '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: 'b428b244-777b-49a6-a48d-5dffedced8e7',
  //       extendedRequestId: undefined,
```

```
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //     },
  //     project: {
  //       arn: 'arn:aws:codebuild:us-east-1:xxxxxxxxxxxx:project/MyCodeBuilder',
  //       artifacts: {
  //         encryptionDisabled: false,
  //         location: 'xxxxxx-xxxxxxx-xxxxxx',
  //         name: 'MyCodeBuilder',
  //         namespaceType: 'NONE',
  //         packaging: 'NONE',
  //         type: 'S3'
  //       },
  //       badge: { badgeEnabled: false },
  //       cache: { type: 'NO_CACHE' },
  //       created: 2023-08-18T14:46:48.979Z,
  //       encryptionKey: 'arn:aws:kms:us-east-1:xxxxxxxxxxxx:alias/aws/s3',
  //       environment: {
  //         computeType: 'BUILD_GENERAL1_SMALL',
  //         environmentVariables: [],
  //         image: 'aws/codebuild/standard:7.0',
  //         imagePullCredentialsType: 'CODEBUILD',
  //         privilegedMode: false,
  //         type: 'LINUX_CONTAINER'
  //       },
  //       lastModified: 2023-08-18T14:46:48.979Z,
  //       name: 'MyCodeBuilder',
  //       projectVisibility: 'PRIVATE',
  //       queuedTimeoutInMinutes: 480,
  //       serviceRole: 'arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin',
  //       source: {
  //         insecureSsl: false,
  //         location: 'https://...',
  //         reportBuildStatus: false,
  //         type: 'GITHUB'
  //       },
  //       timeoutInMinutes: 60
  //     }
  //   }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateProjecta Referência AWS SDK para JavaScript da API.

Use **`ListBuilds`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListBuilds`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the CodeBuild builds.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
    listBuildsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Used for pagination.

    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
        }
```

```
        Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
codeBuildClient.ListBuilds(
                listBuildsRequest);

        if (listBuildsOutcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &ids =
listBuildsOutcome.GetResult().GetIds();
            if (!ids.empty()) {

                std::cout << "Information about each build:" << std::endl;
                Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
                getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
                Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
codeBuildClient.BatchGetBuilds(
                        getBuildsRequest);

                if (getBuildsOutcome.IsSuccess()) {
                    const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
getBuildsOutcome.GetResult().GetBuilds();
                    std::cout << builds.size() << " build(s) found." <<
std::endl;
                    for (auto val: builds) {
                        std::cout << val.GetId() << std::endl;
                    }
                } else {
                    std::cerr << "Error getting builds"
                              << getBuildsOutcome.GetError().GetMessage() <<
std::endl;
                    return false;
                }
            } else {
                std::cout << "No builds found." << std::endl;
            }

            // Get the next token for pagination.

            nextToken = listBuildsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error listing builds"
                      << listBuildsOutcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    } while (!nextToken.

            empty()

            );

    return true;
}
```

- Para obter detalhes da API, consulte ListBuildsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para obter uma lista de AWS CodeBuild IDs de compilações.

O `list-builds` exemplo a seguir obtém uma lista de CodeBuild IDs classificadas em ordem crescente.

```
aws codebuild list-builds --sort-order ASCENDING
```

A saída inclui um valor `nextToken` que indica que há mais saída disponível.

```
{
    "nextToken": "4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==",
    "ids": [
        "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
        "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
            ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
    ]
}
```

Execute esse comando novamente e forneça o valor `nextToken` na resposta anterior como parâmetro para obter a próxima parte da saída. Repita até que você não receba um valor `nextToken` na resposta.

```
aws codebuild list-builds --sort-order ASCENDING --next-
token 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```

Próxima parte da saída:

```
{
    "ids": [
        "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
        "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
            ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
    ]
}
```

Para obter mais informações, consulte Exibir uma lista de IDs de compilação (AWS CLI) no Guia do usuário AWS CodeBuild

- Para obter detalhes da API, consulte ListBuildsem Referência de AWS CLI Comandos.

Use **`ListProjects`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListProjects`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the CodeBuild projects.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType
 sortType,
```

```
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Next token for pagination.
    Aws::Vector<Aws::String> allProjects;

    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
 codeBuildClient.ListProjects(
                listProjectsRequest);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
 outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(),
 projects.end());
            nextToken = outcome.GetResult().GetNextToken();
        }

        else {
            std::cerr << "Error listing projects" <<
 outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (!nextToken.empty());

    std::cout << allProjects.size() << " project(s) found." << std::endl;
    for (auto project: allProjects) {
        std::cout << project << std::endl;
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListProjectsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para obter uma lista dos nomes dos projetos de AWS CodeBuild construção.

O `list-projects` exemplo a seguir obtém uma lista de projetos de CodeBuild construção classificados por nome em ordem crescente.

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

A saída inclui um valor `nextToken` que indica que há mais saída disponível.

```
{
    "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U
+AkMx8=",
    "projects": [
        "codebuild-demo-project",
        "codebuild-demo-project2",
            ... The full list of build project names has been omitted for
 brevity ...
        "codebuild-demo-project99"
    ]
}
```

Execute esse comando novamente e forneça o valor `nextToken` da resposta anterior como parâmetro para obter a próxima parte da saída. Repita até que você não receba um valor `nextToken` na resposta.

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-
token Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=

{
    "projects": [
        "codebuild-demo-project100",
        "codebuild-demo-project101",

  ... The full list of build project names has been omitted for brevity ...
        "codebuild-demo-project122"
```

```
  ]
}
```

Para obter mais informações, consulte Exibir uma lista de nomes de projetos de construção (AWS CLI) no Guia do AWS CodeBuild usuário.

- Para obter detalhes da API, consulte ListProjectsem Referência de AWS CLI Comandos.

Use **StartBuild**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartBuild`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Start an AWS CodeBuild project build.
/*!
  \param projectName: A CodeBuild project name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome =
 codeBuildClient.StartBuild(
            startBuildRequest);

    if (outcome.IsSuccess()) {
```

```
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
                  << std::endl;
    }

    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte StartBuilda Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para começar a executar uma compilação de um projeto de AWS CodeBuild compilação.

O `start-build` exemplo a seguir inicia uma compilação para o CodeBuild projeto especificado. A compilação substitui a configuração do projeto pelo número de minutos que a compilação pode ficar na fila antes de atingir o tempo limite e as configurações de artefato do projeto.

```
aws codebuild start-build \
    --project-name "my-demo-project" \
    --queued-timeout-in-minutes-override 5 \
    --artifacts-override {"\"type\": \"S3\",\"location\":
 \"arn:aws:s3:::artifacts-override\",\"overrideArtifactName\":true"}
```

Saída:

```
{
    "build": {
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
        "buildStatus": "IN_PROGRESS",
        "buildComplete": false,
        "projectName": "my-demo-project",
```

```
        "timeoutInMinutes": 60,
        "source": {
            "insecureSsl": false,
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip"
        },
        "queuedTimeoutInMinutes": 5,
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "currentPhase": "QUEUED",
        "startTime": 1556905683.568,
        "environment": {
            "computeType": "BUILD_GENERAL1_MEDIUM",
            "environmentVariables": [],
            "type": "LINUX_CONTAINER",
            "privilegedMode": false,
            "image": "aws/codebuild/standard:1.0",
            "imagePullCredentialsType": "CODEBUILD"
        },
        "phases": [
            {
                "phaseStatus": "SUCCEEDED",
                "startTime": 1556905683.568,
                "phaseType": "SUBMITTED",
                "durationInSeconds": 0,
                "endTime": 1556905684.524
            },
            {
                "startTime": 1556905684.524,
                "phaseType": "QUEUED"
            }
        ],
        "logs": {
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logEvent:group=null;stream=null"
        },
        "artifacts": {
            "encryptionDisabled": false,
            "location": "arn:aws:s3:::artifacts-override/my-demo-project",
            "overrideArtifactName": true
        },
        "cache": {
            "type": "NO_CACHE"
        },
```

```
        "id": "my-demo-project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE",
        "initiator": "my-aws-account-name",
        "arn": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-
project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE"
    }
}
```

Para obter mais informações, consulte Executar uma compilação (AWS CLI) no Guia do AWS CodeBuild usuário.

- Para obter detalhes da API, consulte StartBuildem Referência de AWS CLI Comandos.

Exemplos de código para CodeCommit usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS CodeCommit com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- CodeCommit Guia do usuário — Mais informações sobre CodeCommit.
- CodeCommit Referência da API — Detalhes sobre todas as CodeCommit ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CodeCommit uso AWS SDKs
  - Ações para CodeCommit usar AWS SDKs
    - Usar BatchGetRepositories com uma CLI
    - Usar CreateBranch com uma CLI
    - Usar CreateRepository com uma CLI
    - Usar DeleteRepository com uma CLI

- Usar GetBranch com uma CLI
- Usar GetRepository com uma CLI
- Usar ListBranches com uma CLI
- Usar ListRepositories com uma CLI
- Usar UpdateDefaultBranch com uma CLI
- Usar UpdateRepositoryDescription com uma CLI
- Usar UpdateRepositoryName com uma CLI

Exemplos básicos de CodeCommit uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS CodeCommit com AWS SDKs.

Exemplos

- Ações para CodeCommit usar AWS SDKs
  - Usar BatchGetRepositories com uma CLI
  - Usar CreateBranch com uma CLI
  - Usar CreateRepository com uma CLI
  - Usar DeleteRepository com uma CLI
  - Usar GetBranch com uma CLI
  - Usar GetRepository com uma CLI
  - Usar ListBranches com uma CLI
  - Usar ListRepositories com uma CLI
  - Usar UpdateDefaultBranch com uma CLI
  - Usar UpdateRepositoryDescription com uma CLI
  - Usar UpdateRepositoryName com uma CLI

Ações para CodeCommit usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar CodeCommit ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS CodeCommit.

Exemplos

- Usar BatchGetRepositories com uma CLI
- Usar CreateBranch com uma CLI
- Usar CreateRepository com uma CLI
- Usar DeleteRepository com uma CLI
- Usar GetBranch com uma CLI
- Usar GetRepository com uma CLI
- Usar ListBranches com uma CLI
- Usar ListRepositories com uma CLI
- Usar UpdateDefaultBranch com uma CLI
- Usar UpdateRepositoryDescription com uma CLI
- Usar UpdateRepositoryName com uma CLI

Usar **`BatchGetRepositories`** com uma CLI

Os exemplos de código a seguir mostram como usar o `BatchGetRepositories`.

CLI

AWS CLI

Como visualizar detalhes sobre vários repositórios

Este exemplo mostra detalhes sobre vários AWS CodeCommit repositórios.

```
aws codecommit batch-get-repositories \
    --repository-names MyDemoRepo MyOtherDemoRepo
```

Saída:

```
{
    "repositoriesNotFound": [],
    "repositories": [
        {
```

```
            "creationDate": 1429203623.625,
            "defaultBranch": "main",
            "repositoryName": "MyDemoRepo",
            "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
            "lastModifiedDate": 1430783812.0869999,
            "repositoryDescription": "My demonstration repository",
            "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
            "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
            "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
            "accountId": "111111111111"
        },
        {
            "creationDate": 1429203623.627,
            "defaultBranch": "main",
            "repositoryName": "MyOtherDemoRepo",
            "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyOtherDemoRepo",
            "lastModifiedDate": 1430783812.0889999,
            "repositoryDescription": "My other demonstration repository",
            "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyOtherDemoRepo",
            "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE",
            "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
            "accountId": "111111111111"
        }
    ],
    "repositoriesNotFound": []
}
```

- Para obter detalhes da API, consulte BatchGetRepositoriesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo confirma quais dos repositórios especificados foram e não foram encontrados.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Saída:

```
Repositories                            RepositoriesNotFound
------------                            --------------------
{MyDemoRepo, MyNewRepo}                {AMissingRepo}
```

- Para obter detalhes da API, consulte BatchGetRepositoriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo confirma quais dos repositórios especificados foram e não foram encontrados.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Saída:

```
Repositories                            RepositoriesNotFound
------------                            --------------------
{MyDemoRepo, MyNewRepo}                {AMissingRepo}
```

- Para obter detalhes da API, consulte BatchGetRepositoriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateBranch`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateBranch`.

CLI

AWS CLI

Para criar uma ramificação

Este exemplo cria uma ramificação em um AWS CodeCommit repositório. Esse comando gerará saídas somente se houver erros.

Comando:

```
aws codecommit create-branch --repository-name MyDemoRepo --branch-
name MyNewBranch --commit-id 317f8570EXAMPLE
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte CreateBranchem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma ramificação com o nome especificado para o repositório especificado e o ID do commit especificado.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- Para obter detalhes da API, consulte CreateBranchem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma ramificação com o nome especificado para o repositório especificado e o ID do commit especificado.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- Para obter detalhes da API, consulte CreateBranchem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateRepository`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateRepository`.

CLI

AWS CLI

Para criar um repositório

Este exemplo cria um repositório e o associa à conta do AWS usuário.

Comando:

```
aws codecommit create-repository --repository-name MyDemoRepo --repository-
description "My demonstration repository"
```

Saída:

```
{
    "repositoryMetadata": {
        "repositoryName": "MyDemoRepo",
                "cloneUrlSsh": "ssh://git-codecommit.us-east-1.amazonaws.com/v1/
repos/MyDemoRepo",
                "lastModifiedDate": 1444766838.027,
        "repositoryDescription": "My demonstration repository",
                "cloneUrlHttp": "https://git-codecommit.us-east-1.amazonaws.com/
v1/repos/MyDemoRepo",
        "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
                "Arn": "arn:aws:codecommit:us-
east-1:111111111111EXAMPLE:MyDemoRepo",
        "accountId": "111111111111"
    }
}
```

- Para obter detalhes da API, consulte CreateRepositoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um repositório com o nome especificado e a descrição especificada.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Saída:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
```

```
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/18/2015 4:13:25 PM
DefaultBranch         :
LastModifiedDate      : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName        : MyDemoRepo
```

- Para obter detalhes da API, consulte CreateRepositoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um repositório com o nome especificado e a descrição especificada.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Saída:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/18/2015 4:13:25 PM
DefaultBranch         :
LastModifiedDate      : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName        : MyDemoRepo
```

- Para obter detalhes da API, consulte CreateRepositoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteRepository`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteRepository`.

CLI

AWS CLI

Para excluir um repositório

Este exemplo mostra como excluir um AWS CodeCommit repositório.

Comando:

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

Saída:

```
{
  "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
}
```

- Para obter detalhes da API, consulte DeleteRepositoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui forçosamente o repositório especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o repositório sem um aviso.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Saída:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Para obter detalhes da API, consulte DeleteRepositoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui forçosamente o repositório especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o repositório sem um aviso.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Saída:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Para obter detalhes da API, consulte DeleteRepositoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetBranch`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBranch`.

CLI

AWS CLI

Para obter informações sobre uma ramificação

Este exemplo obtém informações sobre uma ramificação em um AWS CodeCommit repositório.

Comando:

```
aws codecommit get-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

Saída:

```
{
  "BranchInfo": {
        "commitID": "317f8570EXAMPLE",
                "branchName": "MyNewBranch"
  }
}
```

- Para obter detalhes da API, consulte GetBranchem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre a ramificação especificada para o repositório especificado.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Saída:

```
BranchName                                CommitId
----------                                --------
MyNewBranch                               7763222d...561fc9c9
```

- Para obter detalhes da API, consulte GetBranchem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a ramificação especificada para o repositório especificado.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Saída:

```
BranchName                                CommitId
----------                                --------
MyNewBranch                               7763222d...561fc9c9
```

- Para obter detalhes da API, consulte GetBranchem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetRepository`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetRepository`.

CLI

AWS CLI

Para obter informações sobre um repositório

Este exemplo mostra detalhes sobre um AWS CodeCommit repositório.

```
aws codecommit get-repository \
    --repository-name MyDemoRepo
```

Saída:

```
{
    "repositoryMetadata": {
        "creationDate": 1429203623.625,
        "defaultBranch": "main",
        "repositoryName": "MyDemoRepo",
        "cloneUrlSsh": "ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/v1/
repos/MyDemoRepo",
        "lastModifiedDate": 1430783812.0869999,
        "repositoryDescription": "My demonstration repository",
        "cloneUrlHttp": "https://codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo",
        "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
        "Arn": "arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
        "accountId": "111111111111"
    }
}
```

- Para obter detalhes da API, consulte GetRepositoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações para o repositório especificado.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Saída:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/8/2015 3:21:33 PM
DefaultBranch         :
LastModifiedDate      : 9/8/2015 3:21:33 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName        : MyDemoRepo
```

- Para obter detalhes da API, consulte GetRepositoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações para o repositório especificado.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Saída:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/8/2015 3:21:33 PM
DefaultBranch         :
LastModifiedDate      : 9/8/2015 3:21:33 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName        : MyDemoRepo
```

- Para obter detalhes da API, consulte GetRepositoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListBranches`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListBranches`.

CLI

AWS CLI

Como visualizar uma lista de nomes de ramificação

Este exemplo lista todos os nomes de ramificações em um AWS CodeCommit repositório.

```
aws codecommit list-branches \
    --repository-name MyDemoRepo
```

Saída:

```
{
    "branches": [
        "MyNewBranch",
        "main"
    ]
}
```

- Para obter detalhes da API, consulte ListBranchesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma lista de nomes de ramificações para o repositório especificado.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Saída:

```
master
MyNewBranch
```

- Para obter detalhes da API, consulte ListBranchesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de nomes de ramificações para o repositório especificado.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Saída:

```
master
MyNewBranch
```

- Para obter detalhes da API, consulte ListBranchesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListRepositories`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListRepositories`.

CLI

AWS CLI

Como visualizar uma lista de repositórios

Este exemplo lista todos os AWS CodeCommit repositórios associados à AWS conta do usuário.

Comando:

```
aws codecommit list-repositories
```

Saída:

```
{
  "repositories": [
      {
```

```
            "repositoryName": "MyDemoRepo"
            "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
        },
        {
            "repositoryName": "MyOtherDemoRepo"
            "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE"
        }
    ]
}
```

- Para obter detalhes da API, consulte ListRepositoriesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todos os repositórios em ordem crescente por nome do repositório.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Saída:

```
RepositoryId                            RepositoryName
------------                            --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- Para obter detalhes da API, consulte ListRepositoriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os repositórios em ordem crescente por nome do repositório.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Saída:

```
RepositoryId                            RepositoryName
------------                            --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- Para obter detalhes da API, consulte ListRepositoriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateDefaultBranch`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateDefaultBranch`.

CLI

AWS CLI

Para alterar a ramificação padrão de um repositório

Este exemplo altera a ramificação padrão de um AWS CodeCommit repositório. Esse comando gerará saídas somente se houver erros.

Comando:

```
aws codecommit update-default-branch --repository-name MyDemoRepo --default-
branch-name MyNewBranch
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte UpdateDefaultBranchem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera a ramificação padrão do repositório especificado para a ramificação especificada.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Para obter detalhes da API, consulte UpdateDefaultBranchem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera a ramificação padrão do repositório especificado para a ramificação especificada.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Para obter detalhes da API, consulte UpdateDefaultBranchem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateRepositoryDescription`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateRepositoryDescription`.

CLI

AWS CLI

Para alterar a descrição de um repositório

Este exemplo altera a descrição de um AWS CodeCommit repositório. Esse comando gerará saídas somente se houver erros.

Comando:

```
aws codecommit update-repository-description --repository-name MyDemoRepo --
repository-description "This description was changed"
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte UpdateRepositoryDescriptionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera a descrição do repositório especificado.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- Para obter detalhes da API, consulte UpdateRepositoryDescriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera a descrição do repositório especificado.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- Para obter detalhes da API, consulte UpdateRepositoryDescriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateRepositoryName`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateRepositoryName`.

CLI

AWS CLI

Para alterar o nome de um repositório

Este exemplo altera o nome de um AWS CodeCommit repositório. Esse comando gerará saídas somente se houver erros. A alteração do nome do AWS CodeCommit repositório alterará os URLs SSH e HTTPS que os usuários precisam para se conectar ao repositório. Os usuários não poderão se conectar ao repositório até que atualizem suas configurações de conexão. Além disso, devido à alteração no ARN do repositório, alterar o nome do repositório invalidará todas as políticas de usuário do IAM que dependem do ARN desse repositório.

Comando:

```
aws codecommit update-repository-name --old-name MyDemoRepo --new-
name MyRenamedDemoRepo
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte UpdateRepositoryNameem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera o nome do repositório especificado.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Para obter detalhes da API, consulte UpdateRepositoryNameem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o nome do repositório especificado.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Para obter detalhes da API, consulte UpdateRepositoryNameem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para CodeDeploy usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS CodeDeploy com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- CodeDeploy Guia do desenvolvedor — Mais informações sobre CodeDeploy.
- CodeDeploy Referência da API — Detalhes sobre todas as CodeDeploy ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CodeDeploy uso AWS SDKs
  - Ações para CodeDeploy usar AWS SDKs
    - Usar AddTagsToOnPremisesInstances com uma CLI
    - Usar BatchGetApplications com uma CLI
    - Usar BatchGetDeployments com uma CLI
    - Usar BatchGetOnPremisesInstances com uma CLI
    - Usar CreateApplication com uma CLI
    - Usar CreateDeployment com uma CLI
    - Usar CreateDeploymentConfig com uma CLI
    - Usar CreateDeploymentGroup com uma CLI
    - Usar DeleteApplication com uma CLI
    - Usar DeleteDeploymentConfig com uma CLI
    - Usar DeleteDeploymentGroup com uma CLI
    - Usar DeregisterOnPremisesInstance com uma CLI
    - Usar GetApplication com uma CLI
    - Usar GetApplicationRevision com uma CLI
    - Usar GetDeployment com uma CLI
    - Usar GetDeploymentConfig com uma CLI
    - Usar GetDeploymentGroup com uma CLI
    - Usar GetDeploymentInstance com uma CLI
    - Usar GetOnPremisesInstance com uma CLI

- Usar ListApplicationRevisions com uma CLI
- Usar ListApplications com uma CLI
- Usar ListDeploymentConfigs com uma CLI
- Usar ListDeploymentGroups com uma CLI
- Usar ListDeploymentInstances com uma CLI
- Usar ListDeployments com uma CLI
- Usar ListOnPremisesInstances com uma CLI
- Usar RegisterApplicationRevision com uma CLI
- Usar RegisterOnPremisesInstance com uma CLI
- Usar RemoveTagsFromOnPremisesInstances com uma CLI
- Usar StopDeployment com uma CLI
- Usar UpdateApplication com uma CLI
- Usar UpdateDeploymentGroup com uma CLI

Exemplos básicos de CodeDeploy uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS CodeDeploy com AWS SDKs.

Exemplos

- Ações para CodeDeploy usar AWS SDKs
  - Usar AddTagsToOnPremisesInstances com uma CLI
  - Usar BatchGetApplications com uma CLI
  - Usar BatchGetDeployments com uma CLI
  - Usar BatchGetOnPremisesInstances com uma CLI
  - Usar CreateApplication com uma CLI
  - Usar CreateDeployment com uma CLI
  - Usar CreateDeploymentConfig com uma CLI
  - Usar CreateDeploymentGroup com uma CLI
  - Usar DeleteApplication com uma CLI
  - Usar DeleteDeploymentConfig com uma CLI
  - Usar DeleteDeploymentGroup com uma CLI

- Usar DeregisterOnPremisesInstance com uma CLI
- Usar GetApplication com uma CLI
- Usar GetApplicationRevision com uma CLI
- Usar GetDeployment com uma CLI
- Usar GetDeploymentConfig com uma CLI
- Usar GetDeploymentGroup com uma CLI
- Usar GetDeploymentInstance com uma CLI
- Usar GetOnPremisesInstance com uma CLI
- Usar ListApplicationRevisions com uma CLI
- Usar ListApplications com uma CLI
- Usar ListDeploymentConfigs com uma CLI
- Usar ListDeploymentGroups com uma CLI
- Usar ListDeploymentInstances com uma CLI
- Usar ListDeployments com uma CLI
- Usar ListOnPremisesInstances com uma CLI
- Usar RegisterApplicationRevision com uma CLI
- Usar RegisterOnPremisesInstance com uma CLI
- Usar RemoveTagsFromOnPremisesInstances com uma CLI
- Usar StopDeployment com uma CLI
- Usar UpdateApplication com uma CLI
- Usar UpdateDeploymentGroup com uma CLI

Ações para CodeDeploy usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar CodeDeploy ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS CodeDeploy.

Exemplos

- Usar AddTagsToOnPremisesInstances com uma CLI

- Usar BatchGetApplications com uma CLI
- Usar BatchGetDeployments com uma CLI
- Usar BatchGetOnPremisesInstances com uma CLI
- Usar CreateApplication com uma CLI
- Usar CreateDeployment com uma CLI
- Usar CreateDeploymentConfig com uma CLI
- Usar CreateDeploymentGroup com uma CLI
- Usar DeleteApplication com uma CLI
- Usar DeleteDeploymentConfig com uma CLI
- Usar DeleteDeploymentGroup com uma CLI
- Usar DeregisterOnPremisesInstance com uma CLI
- Usar GetApplication com uma CLI
- Usar GetApplicationRevision com uma CLI
- Usar GetDeployment com uma CLI
- Usar GetDeploymentConfig com uma CLI
- Usar GetDeploymentGroup com uma CLI
- Usar GetDeploymentInstance com uma CLI
- Usar GetOnPremisesInstance com uma CLI
- Usar ListApplicationRevisions com uma CLI
- Usar ListApplications com uma CLI
- Usar ListDeploymentConfigs com uma CLI
- Usar ListDeploymentGroups com uma CLI
- Usar ListDeploymentInstances com uma CLI
- Usar ListDeployments com uma CLI
- Usar ListOnPremisesInstances com uma CLI
- Usar RegisterApplicationRevision com uma CLI
- Usar RegisterOnPremisesInstance com uma CLI
- Usar RemoveTagsFromOnPremisesInstances com uma CLI

- Usar StopDeployment com uma CLI
- Usar UpdateApplication com uma CLI
- Usar UpdateDeploymentGroup com uma CLI

Usar **`AddTagsToOnPremisesInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddTagsToOnPremisesInstances`.

CLI

AWS CLI

Para adicionar tags a uma ou mais instâncias on-premises

O `add-tags-to-on-premises-instances` exemplo a seguir associa AWS CodeDeploy a mesma tag de instância local a duas instâncias locais. Ele não registra as instâncias locais com AWS CodeDeploy.

```
aws deploy add-tags-to-on-premises-instances \
    --instance-names AssetTag12010298EX AssetTag23121309EX \
    --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

Este comando não produz saída.

- Para obter detalhes da API, consulte AddTagsToOnPremisesInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona uma tag de instância on-premises com a chave e o valor especificados para a instância local especificada.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Para obter detalhes da API, consulte AddTagsToOnPremisesInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona uma tag de instância on-premises com a chave e o valor especificados para a instância local especificada.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Para obter detalhes da API, consulte AddTagsToOnPremisesInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`BatchGetApplications`** com uma CLI

Os exemplos de código a seguir mostram como usar o `BatchGetApplications`.

CLI

AWS CLI

Para obter informações sobre várias aplicações

O `batch-get-applications` exemplo a seguir exibe informações sobre vários aplicativos associados à AWS conta do usuário.

```
aws deploy batch-get-applications --application-names WordPress_App MyOther_App
```

Saída:

```
{
    "applicationsInfo": [
        {
            "applicationName": "WordPress_App",
            "applicationId": "d9dd6993-f171-44fa-a811-211e4EXAMPLE",
            "createTime": 1407878168.078,
            "linkedToGitHub": false
        },
        {
            "applicationName": "MyOther_App",
            "applicationId": "8ca57519-31da-42b2-9194-8bb16EXAMPLE",
            "createTime": 1407453571.63,
            "linkedToGitHub": false
        }
```

```
    ]
}
```

- Para obter detalhes da API, consulte BatchGetApplicationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre as aplicações especificadas.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Saída:

```
ApplicationId                           ApplicationName              CreateTime
            LinkedToGitHub
-------------                           ---------------              ----------
            --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication  8/13/2015
 5:53:26 PM    False
```

- Para obter detalhes da API, consulte BatchGetApplicationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as aplicações especificadas.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Saída:

```
ApplicationId                           ApplicationName              CreateTime
            LinkedToGitHub
-------------                           ---------------              ----------
            --------------
```

```
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication  8/13/2015
 5:53:26 PM    False
```

- Para obter detalhes da API, consulte BatchGetApplicationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`BatchGetDeployments`** com uma CLI

Os exemplos de código a seguir mostram como usar o `BatchGetDeployments`.

CLI

AWS CLI

Para obter informações sobre várias implantações

O `batch-get-deployments` exemplo a seguir exibe informações sobre várias implantações associadas à AWS conta do usuário.

```
aws deploy batch-get-deployments --deployment-ids d-A1B2C3111 d-A1B2C3222
```

Saída:

```
{
    "deploymentsInfo": [
        {
            "applicationName": "WordPress_App",
            "status": "Failed",
            "deploymentOverview": {
                "Failed": 0,
                "InProgress": 0,
                "Skipped": 0,
                "Succeeded": 1,
                "Pending": 0
            },
            "deploymentConfigName": "CodeDeployDefault.OneAtATime",
            "creator": "user",
            "deploymentGroupName": "WordPress_DG",
            "revision": {
                "revisionType": "S3",
```

```
                "s3Location": {
                "bundleType": "zip",
                "version": "uTecLusEXAMPLEFXtfUcyfV8bEXAMPLE",
                "bucket": "amzn-s3-demo-bucket",
                "key": "WordPressApp.zip"
                }
            },
            "deploymentId": "d-A1B2C3111",
            "createTime": 1408480721.9,
            "completeTime": 1408480741.822
        },
        {
            "applicationName": "MyOther_App",
            "status": "Failed",
            "deploymentOverview": {
                "Failed": 1,
                "InProgress": 0,
                "Skipped": 0,
                "Succeeded": 0,
                "Pending": 0
            },
            "deploymentConfigName": "CodeDeployDefault.OneAtATime",
            "creator": "user",
            "errorInformation": {
                "message": "Deployment failed: Constraint default violated: No
 hosts succeeded.",
                "code": "HEALTH_CONSTRAINTS"
            },
            "deploymentGroupName": "MyOther_DG",
            "revision": {
                "revisionType": "S3",
                "s3Location": {
                "bundleType": "zip",
                "eTag": "\"dd56cfdEXAMPLE8e768f9d77fEXAMPLE\"",
                "bucket": "amzn-s3-demo-bucket",
                "key": "MyOtherApp.zip"
                }
            },
            "deploymentId": "d-A1B2C3222",
            "createTime": 1409764576.589,
            "completeTime": 1409764596.101
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte BatchGetDeploymentsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre as implantações especificadas.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Saída:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded

ApplicationName               : CodePipelineDemoApplication
CompleteTime                  : 7/23/2015 6:07:30 PM
CreateTime                    : 7/23/2015 6:06:29 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodePipelineDemoFleet
DeploymentId                  : d-RR0T5KTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
```

```
StartTime                      : 1/1/0001 12:00:00 AM
Status                         : Succeeded
```

- Para obter detalhes da API, consulte BatchGetDeploymentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as implantações especificadas.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Saída:

```
ApplicationName                : CodeDeployDemoApplication
CompleteTime                   : 7/23/2015 11:26:04 PM
CreateTime                     : 7/23/2015 11:24:43 PM
Creator                        : user
DeploymentConfigName           : CodeDeployDefault.OneAtATime
DeploymentGroupName            : CodeDeployDemoFleet
DeploymentId                   : d-QZMRGSTEX
DeploymentOverview             : Amazon.CodeDeploy.Model.DeploymentOverview
Description                    :
ErrorInformation               :
IgnoreApplicationStopFailures  : False
Revision                       : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                      : 1/1/0001 12:00:00 AM
Status                         : Succeeded

ApplicationName                : CodePipelineDemoApplication
CompleteTime                   : 7/23/2015 6:07:30 PM
CreateTime                     : 7/23/2015 6:06:29 PM
Creator                        : user
DeploymentConfigName           : CodeDeployDefault.OneAtATime
DeploymentGroupName            : CodePipelineDemoFleet
DeploymentId                   : d-RR0T5KTEX
DeploymentOverview             : Amazon.CodeDeploy.Model.DeploymentOverview
Description                    :
ErrorInformation               :
IgnoreApplicationStopFailures  : False
Revision                       : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                      : 1/1/0001 12:00:00 AM
Status                         : Succeeded
```

- Para obter detalhes da API, consulte BatchGetDeploymentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`BatchGetOnPremisesInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `BatchGetOnPremisesInstances`.

CLI

AWS CLI

Para obter informações sobre uma ou mais instâncias on-premises

O exemplo `batch-get-on-premises-instances` a seguir obtém informações sobre duas instâncias on-premises.

```
aws deploy batch-get-on-premises-instances --instance-
names AssetTag12010298EX AssetTag23121309EX
```

Saída:

```
{
    "instanceInfos": [
        {
            "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/
AssetTag12010298EX",
            "tags": [
                {
                    "Value": "CodeDeployDemo-OnPrem",
                    "Key": "Name"
                }
            ],
            "instanceName": "AssetTag12010298EX",
            "registerTime": 1425579465.228,
            "instanceArn": "arn:aws:codedeploy:us-west-2:123456789012:instance/
AssetTag12010298EX_4IwLNI2Alh"
        },
        {
            "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/
AssetTag23121309EX",
            "tags": [
                {
```

```
                    "Value": "CodeDeployDemo-OnPrem",
                    "Key": "Name"
                }
            ],
            "instanceName": "AssetTag23121309EX",
            "registerTime": 1425595585.988,
            "instanceArn": "arn:aws:codedeploy:us-west-2:80398EXAMPLE:instance/
AssetTag23121309EX_PomUy64Was"
        }
    ]
}
```

- Para obter detalhes da API, consulte BatchGetOnPremisesInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre as instâncias on-premises especificadas.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX,
 AssetTag12010298EX-2
```

Saída:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName   : AssetTag12010298EX-2
RegisterTime   : 4/3/2015 6:38:52 PM
Tags           : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Para obter detalhes da API, consulte BatchGetOnPremisesInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as instâncias on-premises especificadas.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX,
 AssetTag12010298EX-2
```

Saída:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName   : AssetTag12010298EX-2
RegisterTime   : 4/3/2015 6:38:52 PM
Tags           : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Para obter detalhes da API, consulte BatchGetOnPremisesInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateApplication`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateApplication`.

CLI

AWS CLI

Para criar uma aplicação

O `create-application` exemplo a seguir cria um aplicativo e o associa à AWS conta do usuário.

```
aws deploy create-application --application-name MyOther_App
```

Saída:

```
{
    "applicationId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
}
```

- Para obter detalhes da API, consulte CreateApplicationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma aplicação com o nome especificado.

```
New-CDApplication -ApplicationName MyNewApplication
```

Saída:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Para obter detalhes da API, consulte CreateApplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma aplicação com o nome especificado.

```
New-CDApplication -ApplicationName MyNewApplication
```

Saída:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Para obter detalhes da API, consulte CreateApplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateDeployment`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateDeployment`.

CLI

AWS CLI

Exemplo 1: Para criar uma CodeDeploy implantação usando a plataforma de EC2/On-premises computação

O `create-deployment` exemplo a seguir cria uma implantação e a associa à AWS conta do usuário.

```
aws deploy create-deployment \
    --application-name WordPress_App \
    --deployment-config-name CodeDeployDefault.OneAtATime \
    --deployment-group-name WordPress_DG \
    --description "My demo deployment" \
    --s3-location bucket=amzn-s3-demo-
bucket,bundleType=zip,eTag=dd56cfdEXAMPLE8e768f9d77fEXAMPLE,key=WordPressApp.zip
```

Saída:

```
{
    "deploymentId": "d-A1B2C3111"
}
```

Exemplo 2: Para criar uma CodeDeploy implantação usando a plataforma computacional Amazon ECS

O exemplo `create-deployment` a seguir usa os dois arquivos a seguir para implantar um serviço do Amazon ECS.

Conteúdo do arquivo `create-deployment.json`:

```
{
    "applicationName": "ecs-deployment",
    "deploymentGroupName": "ecs-deployment-dg",
    "revision": {
        "revisionType": "S3",
```

```
        "s3Location": {
            "bucket": "ecs-deployment-bucket",
            "key": "appspec.yaml",
            "bundleType": "YAML"
        }
    }
}
```

Esse arquivo, por sua vez, recupera o seguinte arquivo `appspec.yaml` de um bucket do S3 chamado `ecs-deployment-bucket`.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:region:123456789012:task-definition/ecs-
task-def:2"
        LoadBalancerInfo:
          ContainerName: "sample-app"
          ContainerPort: 80
        PlatformVersion: "LATEST"
```

Comando:

```
aws deploy create-deployment \
    --cli-input-json file://create-deployment.json \
    --region us-east-1
```

Saída:

```
{
    "deploymentId": "d-1234ABCDE"
}
```

Para obter mais informações, consulte CreateDeployment na Referência de APIs do AWS CodeDeploy .

- Para obter detalhes da API, consulte CreateDeploymentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma implantação para a aplicação e o grupo de implantação especificados com a configuração de implantação e a revisão da aplicação especificadas.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -DeploymentConfigName
 CodeDeployDefault.OneAtATime -DeploymentGroupName MyNewDeploymentGroup -
IgnoreApplicationStopFailures $True -S3Location_Key aws-codedeploy_linux-
master.zip -RevisionType S3
```

Saída:

```
d-ZHROG7UEX
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -DeploymentConfigName
 CodeDeployDefault.OneAtATime -DeploymentGroupName MyNewDeploymentGroup
 -IgnoreApplicationStopFailures $True -S3Location_Key aws-
codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

Saída:

```
d-ZHROG7UEX
```

- Para obter detalhes da API, consulte CreateDeploymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma implantação para a aplicação e o grupo de implantação especificados com a configuração de implantação e a revisão da aplicação especificadas.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -DeploymentConfigName
```

```
 CodeDeployDefault.OneAtATime -DeploymentGroupName MyNewDeploymentGroup -
IgnoreApplicationStopFailures $True -S3Location_Key aws-codedeploy_linux-
master.zip -RevisionType S3
```

Saída:

```
d-ZHROG7UEX
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -DeploymentConfigName
 CodeDeployDefault.OneAtATime -DeploymentGroupName MyNewDeploymentGroup
 -IgnoreApplicationStopFailures $True -S3Location_Key aws-
codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

Saída:

```
d-ZHROG7UEX
```

- Para obter detalhes da API, consulte CreateDeploymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateDeploymentConfig`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateDeploymentConfig`.

CLI

AWS CLI

Para criar uma configuração de implantação personalizada

O `create-deployment-config` exemplo a seguir cria uma configuração de implantação personalizada e a associa à AWS conta do usuário.

```
aws deploy create-deployment-config \
```

```
    --deployment-config-name ThreeQuartersHealthy \
    --minimum-healthy-hosts type=FLEET_PERCENT,value=75
```

Saída:

```
{
    "deploymentConfigId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
}
```

- Para obter detalhes da API, consulte CreateDeploymentConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma configuração de implantação com o nome e o comportamento especificados.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Saída:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Para obter detalhes da API, consulte CreateDeploymentConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma configuração de implantação com o nome e o comportamento especificados.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Saída:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Para obter detalhes da API, consulte CreateDeploymentConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateDeploymentGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateDeploymentGroup`.

CLI

AWS CLI

Para criar um grupo de implantação

O `create-deployment-group` exemplo a seguir cria um grupo de implantação e o associa ao aplicativo especificado e à AWS conta do usuário.

```
aws deploy create-deployment-group \
    --application-name WordPress_App \
    --auto-scaling-groups CodeDeployDemo-ASG \
    --deployment-config-name CodeDeployDefault.OneAtATime \
    --deployment-group-name WordPress_DG \
    --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE \
    --service-role-arn arn:aws:iam::123456789012:role/CodeDeployDemoRole
```

Saída:

```
{
    "deploymentGroupId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
}
```

- Para obter detalhes da API, consulte CreateDeploymentGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um grupo de implantação com o nome, o grupo do Auto Scaling, a configuração de implantação, a tag e o perfil de serviço especificados para a aplicação especificada.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

Saída:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

Saída:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Para obter detalhes da API, consulte CreateDeploymentGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um grupo de implantação com o nome, o grupo do Auto Scaling, a configuração de implantação, a tag e o perfil de serviço especificados para a aplicação especificada.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

Saída:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

Saída:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Para obter detalhes da API, consulte CreateDeploymentGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteApplication`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteApplication`.

CLI

AWS CLI

Como excluir uma aplicação

O `delete-application` exemplo a seguir exclui o aplicativo especificado que está associado à AWS conta do usuário.

```
aws deploy delete-application --application-name WordPress_App
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteApplicationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a aplicação com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a aplicação sem um aviso.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Para obter detalhes da API, consulte DeleteApplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a aplicação com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a aplicação sem um aviso.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Para obter detalhes da API, consulte DeleteApplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteDeploymentConfig`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteDeploymentConfig`.

CLI

AWS CLI

Para excluir uma configuração de implantação

O `delete-deployment-config` exemplo a seguir exclui uma configuração de implantação personalizada associada à AWS conta do usuário.

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteDeploymentConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a configuração de implantação com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a configuração de implantação sem um aviso.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Para obter detalhes da API, consulte DeleteDeploymentConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a configuração de implantação com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a configuração de implantação sem um aviso.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Para obter detalhes da API, consulte DeleteDeploymentConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteDeploymentGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteDeploymentGroup`.

CLI

AWS CLI

Para excluir um grupo de implantação

O exemplo `delete-deployment-group` a seguir exclui um grupo de implantação associado à aplicação especificada.

```
aws deploy delete-deployment-group \
    --application-name WordPress_App \
    --deployment-group-name WordPress_DG
```

Saída:

```
{
    "hooksNotCleanedUp": []
}
```

- Para obter detalhes da API, consulte DeleteDeploymentGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o grupo de implantação com o nome especificado para a aplicação especificada. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o grupo de implantação sem um aviso.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- Para obter detalhes da API, consulte DeleteDeploymentGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o grupo de implantação com o nome especificado para a aplicação especificada. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o grupo de implantação sem um aviso.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- Para obter detalhes da API, consulte DeleteDeploymentGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeregisterOnPremisesInstance`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterOnPremisesInstance`.

CLI

AWS CLI

Como cancelar o registro de uma instância on-premises

O `deregister-on-premises-instance` exemplo a seguir cancela o registro de uma instância local com AWS CodeDeploy, mas não exclui o usuário do IAM associado à instância nem se desassocia da instância nas tags AWS CodeDeploy da instância local. Ele também não desinstala o AWS CodeDeploy Agente da instância nem remove o arquivo de configuração local da instância.

```
aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeregisterOnPremisesInstanceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela o registro de uma instância on-premises com o nome especificado.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Para obter detalhes da API, consulte DeregisterOnPremisesInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro de uma instância on-premises com o nome especificado.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Para obter detalhes da API, consulte DeregisterOnPremisesInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetApplication`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetApplication`.

CLI

AWS CLI

Para obter informações sobre uma aplicação

O `get-application` exemplo a seguir exibe informações sobre um aplicativo associado à AWS conta do usuário.

```
aws deploy get-application --application-name WordPress_App
```

Saída:

```
{
    "application": {
        "applicationName": "WordPress_App",
        "applicationId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
        "createTime": 1407878168.078,
        "linkedToGitHub": false
    }
}
```

- Para obter detalhes da API, consulte GetApplicationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre a aplicação especificada.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Saída:

```
ApplicationId                           ApplicationName             CreateTime
            LinkedToGitHub
-------------                           ---------------             ----------
            --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
```

- Para obter detalhes da API, consulte GetApplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a aplicação especificada.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Saída:

```
ApplicationId                           ApplicationName             CreateTime
            LinkedToGitHub
-------------                           ---------------             ----------
            --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
```

- Para obter detalhes da API, consulte GetApplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetApplicationRevision`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetApplicationRevision`.

CLI

AWS CLI

Para obter informações sobre uma revisão da aplicação

O exemplo `get-application-revision` a seguir exibe informações sobre uma revisão da aplicação associada à aplicação especificada.

```
aws deploy get-application-revision \
```

```
    --application-name WordPress_App \
    --s3-location bucket=amzn-s3-demo-
bucket,bundleType=zip,eTag=dd56cfdEXAMPLE8e768f9d77fEXAMPLE,key=WordPressApp.zip
```

Saída:

```
{
    "applicationName": "WordPress_App",
    "revisionInfo": {
        "description": "Application revision registered by Deployment ID: d-
A1B2C3111",
        "registerTime": 1411076520.009,
        "deploymentGroups": "WordPress_DG",
        "lastUsedTime": 1411076520.009,
        "firstUsedTime": 1411076520.009
    },
    "revision": {
        "revisionType": "S3",
        "s3Location": {
            "bundleType": "zip",
            "eTag": "dd56cfdEXAMPLE8e768f9d77fEXAMPLE",
            "bucket": "amzn-s3-demo-bucket",
            "key": "WordPressApp.zip"
        }
    }
}
```

- Para obter detalhes da API, consulte GetApplicationRevisionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre a revisão da aplicação especificada.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
 -S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Saída:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- Para obter detalhes da API, consulte GetApplicationRevisionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a revisão da aplicação especificada.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
 -S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Saída:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- Para obter detalhes da API, consulte GetApplicationRevisionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetDeployment`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetDeployment`.

CLI

AWS CLI

Para obter informações sobre uma implantação

O `get-deployment` exemplo a seguir exibe informações sobre uma implantação associada à AWS conta do usuário.

```
aws deploy get-deployment --deployment-id d-A1B2C3123
```

Saída:

```
{
    "deploymentInfo": {
        "applicationName": "WordPress_App",
        "status": "Succeeded",
        "deploymentOverview": {
            "Failed": 0,
            "InProgress": 0,
            "Skipped": 0,
            "Succeeded": 1,
            "Pending": 0
        },
        "deploymentConfigName": "CodeDeployDefault.OneAtATime",
        "creator": "user",
        "description": "My WordPress app deployment",
        "revision": {
            "revisionType": "S3",
            "s3Location":  {
            "bundleType": "zip",
            "eTag": "\"dd56cfdEXAMPLE8e768f9d77fEXAMPLE\"",
            "bucket": "amzn-s3-demo-bucket",
            "key": "WordPressApp.zip"
            }
        },
        "deploymentId": "d-A1B2C3123",
        "deploymentGroupName": "WordPress_DG",
        "createTime": 1409764576.589,
        "completeTime": 1409764596.101,
        "ignoreApplicationStopFailures": false
    }
}
```

- Para obter detalhes da API, consulte GetDeploymentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações de resumo sobre a implantação especificada.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Saída:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

Exemplo 2: esse exemplo obtém informações sobre o status das instâncias que estão participando da implantação especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Saída:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Exemplo 3: esse exemplo obtém informações sobre a revisão da aplicação para a implantação especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Saída:

```
Bucket     : amzn-s3-demo-bucket
BundleType : zip
```

```
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- Para obter detalhes da API, consulte GetDeploymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações de resumo sobre a implantação especificada.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Saída:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

Exemplo 2: esse exemplo obtém informações sobre o status das instâncias que estão participando da implantação especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Saída:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Exemplo 3: esse exemplo obtém informações sobre a revisão da aplicação para a implantação especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Saída:

```
Bucket     : amzn-s3-demo-bucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- Para obter detalhes da API, consulte GetDeploymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetDeploymentConfig`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetDeploymentConfig`.

CLI

AWS CLI

Para obter informações sobre uma configuração de implantação

O `get-deployment-config` exemplo a seguir exibe informações sobre uma configuração de implantação associada à AWS conta do usuário.

```
aws deploy get-deployment-config --deployment-config-name ThreeQuartersHealthy
```

Saída:

```
{
    "deploymentConfigInfo": {
        "deploymentConfigId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
        "minimumHealthyHosts": {
            "type": "FLEET_PERCENT",
            "value": 75
        },
```

```
        "createTime": 1411081164.379,
        "deploymentConfigName": "ThreeQuartersHealthy"
    }
}
```

- Para obter detalhes da API, consulte GetDeploymentConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações de resumo sobre a configuração da implantação especificada.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Saída:

```
CreateTime              DeploymentConfigId
 DeploymentConfigName    MinimumHealthyHosts
----------              ------------------
 --------------------    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE
 ThreeQuartersHealthy    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Exemplo 2: esse exemplo obtém informações sobre a configuração da implantação especificada.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Saída:

```
Type             Value
----             -----
FLEET_PERCENT    75
```

- Para obter detalhes da API, consulte GetDeploymentConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações de resumo sobre a configuração da implantação especificada.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Saída:

```
CreateTime              DeploymentConfigId
 DeploymentConfigName    MinimumHealthyHosts
----------              ------------------
 --------------------    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE
 ThreeQuartersHealthy    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Exemplo 2: esse exemplo obtém informações sobre a configuração da implantação especificada.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Saída:

```
Type             Value
----             -----
FLEET_PERCENT    75
```

- Para obter detalhes da API, consulte GetDeploymentConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetDeploymentGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetDeploymentGroup`.

CLI

AWS CLI

Como visualizar informações sobre um grupo de implantação

O exemplo `get-deployment-group` a seguir exibe informações sobre um grupo de implantação associado à aplicação especificada.

```
aws deploy get-deployment-group \
    --application-name WordPress_App \
    --deployment-group-name WordPress_DG
```

Saída:

```
{
    "deploymentGroupInfo": {
        "applicationName": "WordPress_App",
        "autoScalingGroups": [
            "CodeDeployDemo-ASG"
        ],
        "deploymentConfigName": "CodeDeployDefault.OneAtATime",
        "ec2TagFilters": [
            {
                "Type": "KEY_AND_VALUE",
                "Value": "CodeDeployDemo",
                "Key": "Name"
            }
        ],
        "deploymentGroupId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
        "serviceRoleArn": "arn:aws:iam::123456789012:role/CodeDeployDemoRole",
        "deploymentGroupName": "WordPress_DG"
    }
}
```

- Para obter detalhes da API, consulte GetDeploymentGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre o grupo de implantação especificado.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Saída:

```
ApplicationName              : CodeDeployDemoApplication
AutoScalingGroups            : {}
DeploymentConfigName         : CodeDeployDefault.OneAtATime
DeploymentGroupId            : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName          : CodeDeployDemoFleet
Ec2TagFilters                : {Name}
OnPremisesInstanceTagFilters : {}
ServiceRoleArn               : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- Para obter detalhes da API, consulte GetDeploymentGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre o grupo de implantação especificado.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Saída:

```
ApplicationName              : CodeDeployDemoApplication
AutoScalingGroups            : {}
DeploymentConfigName         : CodeDeployDefault.OneAtATime
DeploymentGroupId            : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName          : CodeDeployDemoFleet
Ec2TagFilters                : {Name}
OnPremisesInstanceTagFilters : {}
ServiceRoleArn               : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- Para obter detalhes da API, consulte GetDeploymentGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetDeploymentInstance`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetDeploymentInstance`.

CLI

AWS CLI

Para obter informações sobre uma instância de implantação

O exemplo `get-deployment-instance` a seguir exibe informações sobre uma instância de implantação associada à implantação especificada.

```
aws deploy get-deployment-instance --deployment-id d-QA4G4F9EX --instance-
id i-902e9fEX
```

Saída:

```
{
    "instanceSummary": {
        "instanceId": "arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-902e9fEX",
        "lifecycleEvents": [
            {
                "status": "Succeeded",
                "endTime": 1408480726.569,
                "startTime": 1408480726.437,
                "lifecycleEventName": "ApplicationStop"
            },
            {
                "status": "Succeeded",
                "endTime": 1408480728.016,
                "startTime": 1408480727.665,
                "lifecycleEventName": "DownloadBundle"
            },
            {
                "status": "Succeeded",
                "endTime": 1408480729.744,
                "startTime": 1408480729.125,
                "lifecycleEventName": "BeforeInstall"
            },
            {
                "status": "Succeeded",
                "endTime": 1408480730.979,
                "startTime": 1408480730.844,
                "lifecycleEventName": "Install"
            },
            {
```

```
                "status": "Failed",
                "endTime": 1408480732.603,
                "startTime": 1408480732.1,
                "lifecycleEventName": "AfterInstall"
            },
            {
                "status": "Skipped",
                "endTime": 1408480732.606,
                "lifecycleEventName": "ApplicationStart"
            },
            {
                "status": "Skipped",
                "endTime": 1408480732.606,
                "lifecycleEventName": "ValidateService"
            }
        ],
        "deploymentId": "d-QA4G4F9EX",
        "lastUpdatedAt": 1408480733.152,
        "status": "Failed"
    }
}
```

- Para obter detalhes da API, consulte GetDeploymentInstanceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre a instância especificada para a implantação especificada.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Saída:

```
DeploymentId    : d-QZMRGSTEX
InstanceId      : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt   : 7/23/2015 11:25:24 PM
LifecycleEvents : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status          : Succeeded
```

- Para obter detalhes da API, consulte GetDeploymentInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a instância especificada para a implantação especificada.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Saída:

```
DeploymentId    : d-QZMRGSTEX
InstanceId      : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt   : 7/23/2015 11:25:24 PM
LifecycleEvents : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status          : Succeeded
```

- Para obter detalhes da API, consulte GetDeploymentInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetOnPremisesInstance`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetOnPremisesInstance`.

CLI

AWS CLI

Para obter informações sobre uma única instância on-premises

O exemplo `get-on-premises-instance` a seguir retorna as informações sobre uma instância on-premises especificada.

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

Saída:

```
{
    "instanceInfo": {
    "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/
AssetTag12010298EX",
```

```
        "tags": [
        {
            "Value": "CodeDeployDemo-OnPrem",
            "Key": "Name"
        }
        ],
        "instanceName": "AssetTag12010298EX",
        "registerTime": 1425579465.228,
        "instanceArn": "arn:aws:codedeploy:us-east-1:123456789012:instance/
AssetTag12010298EX_4IwLNI2Alh"
    }
}
```

- Para obter detalhes da API, consulte GetOnPremisesInstanceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre a instância on-premises especificada.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Saída:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Para obter detalhes da API, consulte GetOnPremisesInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a instância on-premises especificada.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Saída:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Para obter detalhes da API, consulte GetOnPremisesInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListApplicationRevisions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListApplicationRevisions`.

CLI

AWS CLI

Para obter informações sobre revisões da aplicação

O exemplo `list-application-revisions` a seguir exibe informações sobre todas as revisões da aplicação associadas à aplicação especificada.

```
aws deploy list-application-revisions \
    --application-name WordPress_App \
    --s-3-bucket amzn-s3-demo-bucket \
    --deployed exclude \
    --s-3-key-prefix WordPress_ \
    --sort-by lastUsedTime \
    --sort-order descending
```

Saída:

```
{
    "revisions": [
        {
            "revisionType": "S3",
            "s3Location": {
                "version": "uTecLusvCB_JqHFXtfUcyfV8bEXAMPLE",
```

```
                "bucket": "amzn-s3-demo-bucket",
                "key": "WordPress_App.zip",
                "bundleType": "zip"
            }
        },
        {
            "revisionType": "S3",
            "s3Location": {
                "version": "tMk.UxgDpMEVb7V187ZM6wVAWEXAMPLE",
                "bucket": "amzn-s3-demo-bucket",
                "key": "WordPress_App_2-0.zip",
                "bundleType": "zip"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte ListApplicationRevisionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre as revisões disponíveis para a aplicação especificada.

```
ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
 CodeDeployDemoApplication -Deployed Ignore)) {
>>   If ($revision.RevisionType -Eq "S3") {
>>     Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
 + ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
 $revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>>   }
>>   If ($revision.RevisionType -Eq "GitHub") {
>>     Write-Output ("Type = GitHub, CommitId = " +
 $revision.GitHubLocation.CommitId + ", Repository = " +
 $revision.GitHubLocation.Repository)
>>   }
>> }
>>
```

Saída:

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Para obter detalhes da API, consulte ListApplicationRevisionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as revisões disponíveis para a aplicação especificada.

```
ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
 CodeDeployDemoApplication -Deployed Ignore)) {
>>   If ($revision.RevisionType -Eq "S3") {
>>     Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
 + ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
 $revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>>   }
>>   If ($revision.RevisionType -Eq "GitHub") {
>>     Write-Output ("Type = GitHub, CommitId = " +
 $revision.GitHubLocation.CommitId + ", Repository = " +
 $revision.GitHubLocation.Repository)
>>   }
>> }
>>
```

Saída:

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Para obter detalhes da API, consulte ListApplicationRevisionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListApplications`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListApplications`.

CLI

AWS CLI

Para obter informações sobre aplicações

O `list-applications` exemplo a seguir exibe informações sobre todos os aplicativos associados à AWS conta do usuário.

```
aws deploy list-applications
```

Saída:

```
{
    "applications": [
        "WordPress_App",
        "MyOther_App"
    ]
}
```

- Para obter detalhes da API, consulte ListApplicationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma lista das aplicações disponíveis.

```
Get-CDApplicationList
```

Saída:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Para obter detalhes da API, consulte ListApplicationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista das aplicações disponíveis.

```
Get-CDApplicationList
```

Saída:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Para obter detalhes da API, consulte ListApplicationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListDeploymentConfigs`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListDeploymentConfigs`.

CLI

AWS CLI

Para obter informações sobre configurações de implantação

O `list-deployment-configs` exemplo a seguir exibe informações sobre todas as configurações de implantação associadas à AWS conta do usuário.

```
aws deploy list-deployment-configs
```

Saída:

```
{
    "deploymentConfigsList": [
        "ThreeQuartersHealthy",
        "CodeDeployDefault.AllAtOnce",
        "CodeDeployDefault.HalfAtATime",
        "CodeDeployDefault.OneAtATime"
```

```
    ]
}
```

- Para obter detalhes da API, consulte ListDeploymentConfigsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma lista das configurações de implantação disponíveis.

```
Get-CDDeploymentConfigList
```

Saída:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Para obter detalhes da API, consulte ListDeploymentConfigsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista das configurações de implantação disponíveis.

```
Get-CDDeploymentConfigList
```

Saída:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Para obter detalhes da API, consulte ListDeploymentConfigsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListDeploymentGroups`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListDeploymentGroups`.

CLI

AWS CLI

Para obter informações sobre grupos de implantação

O exemplo `list-deployment-groups` a seguir exibe informações sobre todos os grupos de implantação associados à aplicação especificada.

```
aws deploy list-deployment-groups --application-name WordPress_App
```

Saída:

```
{
    "applicationName": "WordPress_App",
    "deploymentGroups": [
        "WordPress_DG",
        "WordPress_Beta_DG"
    ]
}
```

- Para obter detalhes da API, consulte ListDeploymentGroupsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma lista de grupos de implantação para a aplicação especificada.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Saída:

```
ApplicationName              DeploymentGroups
 NextToken
```

```
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Para obter detalhes da API, consulte ListDeploymentGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de grupos de implantação para a aplicação especificada.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Saída:

```
ApplicationName              DeploymentGroups
 NextToken
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Para obter detalhes da API, consulte ListDeploymentGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListDeploymentInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListDeploymentInstances`.

CLI

AWS CLI

Para obter informações sobre instâncias de implantação

O exemplo `list-deployment-instances` a seguir exibe informações sobre todas as instâncias de implantação associadas à implantação especificada.

```
aws deploy list-deployment-instances \
    --deployment-id d-A1B2C3111 \
```

```
    --instance-status-filter Succeeded
```

Saída:

```
{
    "instancesList": [
        "i-EXAMPLE11",
        "i-EXAMPLE22"
    ]
}
```

- Para obter detalhes da API, consulte ListDeploymentInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma lista de IDs de instância para a implantação especificada.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Saída:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Para obter detalhes da API, consulte ListDeploymentInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de IDs de instância para a implantação especificada.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Saída:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Para obter detalhes da API, consulte ListDeploymentInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListDeployments`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListDeployments`.

CLI

AWS CLI

Para obter informações sobre implantações

O exemplo `list-deployments` a seguir exibe informações sobre todas as implantações associadas à aplicação e ao grupo de implantação especificados.

```
aws deploy list-deployments \
    --application-name WordPress_App \
    --create-time-range start=2014-08-19T00:00:00,end=2014-08-20T00:00:00 \
    --deployment-group-name WordPress_DG \
    --include-only-statuses Failed
```

Saída:

```
{
    "deployments": [
        "d-EXAMPLE11",
        "d-EXAMPLE22",
        "d-EXAMPLE33"
    ]
}
```

- Para obter detalhes da API, consulte ListDeploymentsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma lista de IDs de implantação para a aplicação e o grupo de implantação especificados.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Saída:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- Para obter detalhes da API, consulte ListDeploymentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de IDs de implantação para a aplicação e o grupo de implantação especificados.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Saída:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- Para obter detalhes da API, consulte ListDeploymentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListOnPremisesInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListOnPremisesInstances`.

CLI

AWS CLI

Para obter informações sobre uma ou mais instâncias on-premises

O `list-on-premises-instances` exemplo a seguir recupera uma lista de nomes de instâncias locais disponíveis para instâncias registradas AWS CodeDeploy e que também têm a tag de instância local especificada associada à instância. AWS CodeDeploy

```
aws deploy list-on-premises-instances \
    --registration-status Registered \
    --tag-filters Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE
```

Saída:

```
{
    "instanceNames": [
        "AssetTag12010298EX"
    ]
}
```

- Para obter detalhes da API, consulte ListOnPremisesInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma lista de nomes de instâncias on-premises disponíveis.

```
Get-CDOnPremiseInstanceList
```

Saída:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- Para obter detalhes da API, consulte ListOnPremisesInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de nomes de instâncias on-premises disponíveis.

```
Get-CDOnPremiseInstanceList
```

Saída:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- Para obter detalhes da API, consulte ListOnPremisesInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RegisterApplicationRevision`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterApplicationRevision`.

CLI

AWS CLI

Para registrar informações sobre uma revisão de aplicação já carregada

O `register-application-revision` exemplo a seguir registra informações sobre uma revisão de aplicativo já carregada armazenada no Amazon S3 com. AWS CodeDeploy

```
aws deploy register-application-revision \
    --application-name WordPress_App \
    --description "Revised WordPress application" \
    --s3-location bucket=amzn-s3-demo-
bucket,key=RevisedWordPressApp.zip,bundleType=zip,eTag=cecc9b8a08eac650a6e71fdb88EXAMPLE
```

Este comando não produz saída.

- Para obter detalhes da API, consulte RegisterApplicationRevisionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra uma revisão da aplicação com o local do Amazon S3 especificado para a aplicação especificada.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -
S3Location_Bucket amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key
 aws-codedeploy_linux-master.zip -Revision_RevisionType S3
```

- Para obter detalhes da API, consulte RegisterApplicationRevisionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma revisão da aplicação com o local do Amazon S3 especificado para a aplicação especificada.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -
S3Location_Bucket amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key
 aws-codedeploy_linux-master.zip -Revision_RevisionType S3
```

- Para obter detalhes da API, consulte RegisterApplicationRevisionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RegisterOnPremisesInstance`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterOnPremisesInstance`.

CLI

AWS CLI

Para registrar uma instância on-premises

O `register-on-premises-instance` exemplo a seguir registra uma instância local com. AWS CodeDeploy Ele não cria o usuário do IAM especificado nem associa tags à instância registrada em AWS CodeDeploy nenhuma instância local.

```
aws deploy register-on-premises-instance \
```

```
    --instance-name AssetTag12010298EX \
    --iam-user-arn arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser-OnPrem
```

Este comando não produz saída.

- Para obter detalhes da API, consulte RegisterOnPremisesInstanceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra uma instância on-premises com o nome e o usuário do IAM especificados.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Para obter detalhes da API, consulte RegisterOnPremisesInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma instância on-premises com o nome e o usuário do IAM especificados.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Para obter detalhes da API, consulte RegisterOnPremisesInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RemoveTagsFromOnPremisesInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemoveTagsFromOnPremisesInstances`.

CLI

AWS CLI

Para remover tags de uma ou mais instâncias on-premises

O `remove-tags-from-on-premises-instances` exemplo a seguir dissocia as tags locais especificadas das instâncias locais. AWS CodeDeploy Ele não cancela o registro das instâncias locais AWS CodeDeploy, nem desinstala o AWS CodeDeploy Agente da instância, nem remove o arquivo de configuração local das instâncias, nem exclui os usuários do IAM associados às instâncias.

```
aws deploy remove-tags-from-on-premises-instances \
    --instance-names AssetTag12010298EX AssetTag23121309EX \
    --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

Este comando não produz saída.

- Para obter detalhes da API, consulte RemoveTagsFromOnPremisesInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a tag especificada para a instância on-premises com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a tag sem um aviso.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Para obter detalhes da API, consulte RemoveTagsFromOnPremisesInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a tag especificada para a instância on-premises com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a tag sem um aviso.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Para obter detalhes da API, consulte RemoveTagsFromOnPremisesInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`StopDeployment`** com uma CLI

Os exemplos de código a seguir mostram como usar o `StopDeployment`.

CLI

AWS CLI

Como tentar interromper uma implantação

O `stop-deployment` exemplo a seguir tenta interromper uma implantação em andamento associada à AWS conta do usuário.

aws deploy stop-deployment --deployment-id d-A1B2C3111

Saída:

```
{
    "status": "Succeeded",
    "statusMessage": "No more commands will be scheduled for execution in the
 deployment instances"
}
```

- Para obter detalhes da API, consulte StopDeploymentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo tenta interromper a implantação com o ID de implantação especificado.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Saída:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- Para obter detalhes da API, consulte StopDeploymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo tenta interromper a implantação com o ID de implantação especificado.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Saída:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- Para obter detalhes da API, consulte StopDeploymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateApplication`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateApplication`.

CLI

AWS CLI

Para alterar os detalhes de uma aplicação

O `update-application` exemplo a seguir altera o nome de um aplicativo associado à AWS conta do usuário.

```
aws deploy update-application \
    --application-name WordPress_App \
    --new-application-name My_WordPress_App
```

Este comando não produz saída.

- Para obter detalhes da API, consulte UpdateApplicationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera o nome da aplicação especificada.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- Para obter detalhes da API, consulte UpdateApplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o nome da aplicação especificada.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- Para obter detalhes da API, consulte UpdateApplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateDeploymentGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateDeploymentGroup`.

CLI

AWS CLI

Para alterar as informações sobre um grupo de implantação

O exemplo `update-deployment-group` a seguir altera as configurações de um grupo de implantação associado à aplicação especificada.

```
aws deploy update-deployment-group \
    --application-name WordPress_App \
    --auto-scaling-groups My_CodeDeployDemo_ASG \
    --current-deployment-group-name WordPress_DG \
    --deployment-config-name CodeDeployDefault.AllAtOnce \
    --ec2-tag-filters Key=Name,Type=KEY_AND_VALUE,Value=My_CodeDeployDemo \
```

```
    --new-deployment-group-name My_WordPress_DepGroup \
    --service-role-arn arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo-2
```

Este comando não produz saída.

- Para obter detalhes da API, consulte UpdateDeploymentGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera o nome do grupo de implantação especificado para a aplicação especificada.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication
 -CurrentDeploymentGroupName MyNewDeploymentGroup -
NewDeploymentGroupName MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

- Para obter detalhes da API, consulte UpdateDeploymentGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o nome do grupo de implantação especificado para a aplicação especificada.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication
 -CurrentDeploymentGroupName MyNewDeploymentGroup -
NewDeploymentGroupName MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

- Para obter detalhes da API, consulte UpdateDeploymentGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para CodePipeline usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS CodePipeline com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- CodePipeline Guia do usuário — Mais informações sobre CodePipeline.
- CodePipeline Referência da API — Detalhes sobre todas as CodePipeline ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de CodePipeline uso AWS SDKs
  - Ações para CodePipeline usar AWS SDKs
    - Usar AcknowledgeJob com uma CLI
    - Usar CreateCustomActionType com uma CLI
    - Usar CreatePipeline com uma CLI

- Usar DeleteCustomActionType com uma CLI
- Usar DeletePipeline com uma CLI
- Usar DisableStageTransition com uma CLI
- Usar EnableStageTransition com uma CLI
- Usar GetJobDetails com uma CLI
- Usar GetPipeline com uma CLI
- Usar GetPipelineState com uma CLI
- Usar ListActionTypes com uma CLI
- Usar ListPipelines com uma CLI
- Usar PollForJobs com uma CLI
- Usar StartPipelineExecution com uma CLI
- Usar UpdatePipeline com uma CLI

Exemplos básicos de CodePipeline uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS CodePipeline com AWS SDKs.

Exemplos

- Ações para CodePipeline usar AWS SDKs
  - Usar AcknowledgeJob com uma CLI
  - Usar CreateCustomActionType com uma CLI
  - Usar CreatePipeline com uma CLI
  - Usar DeleteCustomActionType com uma CLI
  - Usar DeletePipeline com uma CLI
  - Usar DisableStageTransition com uma CLI
  - Usar EnableStageTransition com uma CLI
  - Usar GetJobDetails com uma CLI
  - Usar GetPipeline com uma CLI
  - Usar GetPipelineState com uma CLI
  - Usar ListActionTypes com uma CLI
  - Usar ListPipelines com uma CLI

- Usar PollForJobs com uma CLI
- Usar StartPipelineExecution com uma CLI
- Usar UpdatePipeline com uma CLI

Ações para CodePipeline usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar CodePipeline ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS CodePipeline.

Exemplos

- Usar AcknowledgeJob com uma CLI
- Usar CreateCustomActionType com uma CLI
- Usar CreatePipeline com uma CLI
- Usar DeleteCustomActionType com uma CLI
- Usar DeletePipeline com uma CLI
- Usar DisableStageTransition com uma CLI
- Usar EnableStageTransition com uma CLI
- Usar GetJobDetails com uma CLI
- Usar GetPipeline com uma CLI
- Usar GetPipelineState com uma CLI
- Usar ListActionTypes com uma CLI
- Usar ListPipelines com uma CLI
- Usar PollForJobs com uma CLI
- Usar StartPipelineExecution com uma CLI
- Usar UpdatePipeline com uma CLI

Usar **`AcknowledgeJob`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AcknowledgeJob`.

CLI

AWS CLI

Para recuperar informações sobre um trabalho

Este exemplo retorna informações sobre um trabalho especificado, incluindo o status desse trabalho, se ele existir. Isso é usado somente para trabalhadores e ações personalizadas. Para determinar o valor de nonce e o ID do trabalho, use o poll-for-jobs do aws codepipeline.

Comando:

```
aws codepipeline acknowledge-job --job-id f4f4ff82-2d11-EXAMPLE --nonce 3
```

Saída:

```
{
  "status": "InProgress"
}
```

- Para obter detalhes da API, consulte AcknowledgeJobem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém o status do trabalho especificado.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Saída:

```
Value
-----
InProgress
```

- Para obter detalhes da API, consulte AcknowledgeJobem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém o status do trabalho especificado.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Saída:

```
Value
-----
InProgress
```

- Para obter detalhes da API, consulte AcknowledgeJobem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateCustomActionType`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateCustomActionType`.

CLI

AWS CLI

Para criar uma ação personalizada

Este exemplo cria uma ação personalizada para AWS CodePipeline usar um arquivo JSON já criado (aqui denominado MyCustomAction.json) que contém a estrutura da ação personalizada. Para obter mais informações sobre os requisitos para criar uma ação personalizada, incluindo a estrutura do arquivo, consulte o Guia AWS CodePipeline do usuário.

```
aws codepipeline create-custom-action-type --cli-input-json file:// MyCustomAction.json
```

Conteúdo do arquivo JSON `MyCustomAction.json`:

```
{
    "category": "Build",
    "provider": "MyJenkinsProviderName",
```

```
    "version": "1",
    "settings": {
        "entityUrlTemplate": "https://192.0.2.4/job/{Config:ProjectName}/",
        "executionUrlTemplate": "https://192.0.2.4/job/{Config:ProjectName}/
lastSuccessfulBuild/{ExternalExecutionId}/"
    },
    "configurationProperties": [
        {
            "name": "MyJenkinsExampleBuildProject",
            "required": true,
            "key": true,
            "secret": false,
            "queryable": false,
            "description": "The name of the build project must be provided when
 this action is added to the pipeline.",
            "type": "String"
        }
    ],
    "inputArtifactDetails": {
        "maximumCount": 1,
        "minimumCount": 0
    },
    "outputArtifactDetails": {
        "maximumCount": 1,
        "minimumCount": 0
    }
}
```

Este comando retorna a estrutura da ação personalizada.

- Para obter detalhes da API, consulte CreateCustomActionTypeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma ação personalizada com as propriedades especificadas.

```
New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
 = "The name of the build project must be provided when this action is
 added to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
 = $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
```

```
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
 -Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
 1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1
```

Saída:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Para obter detalhes da API, consulte CreateCustomActionTypeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma ação personalizada com as propriedades especificadas.

```
New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
 = "The name of the build project must be provided when this action is
 added to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
 = $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
 -Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
 1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1
```

Saída:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Para obter detalhes da API, consulte CreateCustomActionTypeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreatePipeline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreatePipeline`.

CLI

AWS CLI

Para criar um pipeline

Este exemplo cria um pipeline AWS CodePipeline usando um arquivo JSON já criado (aqui denominado MySecondPipeline.json) que contém a estrutura do pipeline. Para obter mais informações sobre os requisitos para criar um pipeline, incluindo a estrutura do arquivo, consulte o Guia AWS CodePipeline do usuário.

Comando:

```
aws codepipeline create-pipeline --cli-input-json file://MySecondPipeline.json
```

Conteúdo de amostra do arquivo JSON:

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
  "stages": [
    {
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "configuration": {
```

```
            "S3Bucket": "awscodepipeline-demo-bucket",
            "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Beta",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "name": "CodePipelineDemoFleet",
          "actionTypeId": {
            "category": "Deploy",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeDeploy"
          },
          "outputArtifacts": [],
          "configuration": {
            "ApplicationName": "CodePipelineDemoApplication",
            "DeploymentGroupName": "CodePipelineDemoFleet"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "codepipeline-us-east-1-11EXAMPLE11"
  },
  "name": "MySecondPipeline",
  "version": 1
 }
}
```

Saída:

```
This command returns the structure of the pipeline.
```

- Para obter detalhes da API, consulte CreatePipelineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um pipeline com as configurações especificadas.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object
 Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName",
 "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
```

```
$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Para obter detalhes da API, consulte CreatePipelineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um pipeline com as configurações especificadas.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
```

```
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object
 Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName",
 "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Para obter detalhes da API, consulte CreatePipelineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteCustomActionType`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteCustomActionType`.

CLI

AWS CLI

Para excluir uma ação personalizada

Este exemplo exclui uma ação personalizada AWS CodePipeline usando um arquivo JSON já criado (aqui denominado DeleteMyCustomAction.json) que contém o tipo de ação, o nome do provedor e o número da versão da ação a ser excluída. Use o comando list-action-types para visualizar os valores corretos para categoria, versão e provedor.

Comando:

```
aws codepipeline delete-custom-action-type --cli-input-json file://
DeleteMyCustomAction.json
```

Conteúdo de amostra do arquivo JSON:

```
{
  "category": "Build",
  "version": "1",
  "provider": "MyJenkinsProviderName"
}
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte DeleteCustomActionTypeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a ação personalizada especificada. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a ação personalizada sem um aviso.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version
 1
```

- Para obter detalhes da API, consulte DeleteCustomActionTypeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a ação personalizada especificada. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a ação personalizada sem um aviso.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version
 1
```

- Para obter detalhes da API, consulte DeleteCustomActionTypeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeletePipeline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeletePipeline`.

CLI

AWS CLI

Para excluir um pipeline

Este exemplo exclui um pipeline chamado MySecondPipeline de AWS CodePipeline. Use o comando list-pipelines para ver uma lista de pipelines associados à sua conta. AWS

Comando:

```
aws codepipeline delete-pipeline --name MySecondPipeline
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte DeletePipelineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o pipeline especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o pipeline sem um aviso.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Para obter detalhes da API, consulte DeletePipelineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o pipeline especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o pipeline sem um aviso.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Para obter detalhes da API, consulte DeletePipelineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DisableStageTransition`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DisableStageTransition`.

CLI

AWS CLI

Como desativar uma transição para um estágio em um pipeline

Este exemplo desativa as transições para o estágio Beta do MyFirstPipeline pipeline em. AWS CodePipeline

Comando:

```
aws codepipeline disable-stage-transition \
--pipeline-name MyFirstPipeline \
--stage-name Beta  \
--transition-type Inbound \
--reason "An example reason"
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte DisableStageTransitionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo desativa a transição de entrada para o estágio especificado no pipeline especificado.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- Para obter detalhes da API, consulte DisableStageTransitionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desativa a transição de entrada para o estágio especificado no pipeline especificado.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- Para obter detalhes da API, consulte DisableStageTransitionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`EnableStageTransition`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EnableStageTransition`.

CLI

AWS CLI

Como permitir a transição para um estágio em um pipeline

Este exemplo permite transições para o estágio Beta do MyFirstPipeline pipeline em AWS CodePipeline.

Comando:

```
aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-
name Beta  --transition-type Inbound
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte EnableStageTransitionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo ativa a transição de entrada para o estágio especificado no pipeline especificado.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- Para obter detalhes da API, consulte EnableStageTransitionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo ativa a transição de entrada para o estágio especificado no pipeline especificado.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- Para obter detalhes da API, consulte EnableStageTransitionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetJobDetails`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetJobDetails`.

CLI

AWS CLI

Para obter detalhes de um trabalho

Este exemplo retorna detalhes sobre um trabalho cujo ID é representado por f4f4ff82-2d11-EXAMPLE. Esse comando é usado somente para ações personalizadas. Quando esse comando é chamado, AWS CodePipeline retorna credenciais temporárias para o bucket Amazon S3 usado para armazenar artefatos para o pipeline, se necessário para a ação personalizada. Esse comando também retornará quaisquer valores secretos definidos para a ação, se houver algum definido.

Comando:

```
aws codepipeline get-job-details --job-id f4f4ff82-2d11-EXAMPLE
```

Saída:

```
{
 "jobDetails": {
  "accountId": "111111111111",
  "data": {
    "actionConfiguration": {
      "__type": "ActionConfiguration",
      "configuration": {
        "ProjectName": "MyJenkinsExampleTestProject"
      }
    },
    "actionTypeId": {
      "__type": "ActionTypeId",
```

```
      "category": "Test",
      "owner": "Custom",
      "provider": "MyJenkinsProviderName",
      "version": "1"
    },
    "artifactCredentials": {
      "__type": "AWSSessionCredentials",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
      "sessionToken":
 "fICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQH
+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQE
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8
    },
    "inputArtifacts": [
      {
        "__type": "Artifact",
        "location": {
          "s3Location": {
            "bucketName": "codepipeline-us-east-1-11EXAMPLE11",
            "objectKey": "MySecondPipeline/MyAppBuild/EXAMPLE"
          },
          "type": "S3"
        },
        "name": "MyAppBuild"
      }
    ],
    "outputArtifacts": [],
    "pipelineContext": {
      "__type": "PipelineContext",
      "action": {
        "name": "MyJenkinsTest-Action"
      },
      "pipelineName": "MySecondPipeline",
      "stage": {
        "name": "Testing"
      }
    }
  },
  "id": "f4f4ff82-2d11-EXAMPLE"
 }
```

```
}
```

- Para obter detalhes da API, consulte GetJobDetailsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações gerais sobre o trabalho especificado.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Saída:

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o trabalho especificado.

```
$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output ("  AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output ("  Configuration:")
ForEach ($key in $jobData.ActionConfiguration.Keys) {
  $value = $jobData.ActionConfiguration.$key
  Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
 $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
```

```
  Write-Output ("    " + $ia.Name)
}
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
  Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)
```

Saída:

```
For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
    MyApp
  OutputArtifacts:
    MyAppBuild
  PipelineContext:
    Name = Build
    PipelineName = CodePipelineDemo
    Stage = Build
```

- Para obter detalhes da API, consulte GetJobDetailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações gerais sobre o trabalho especificado.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Saída:

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o trabalho especificado.

```
$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output ("  AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output ("  Configuration:")
ForEach ($key in $jobData.ActionConfiguration.Keys) {
  $value = $jobData.ActionConfiguration.$key
  Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
 $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
  Write-Output ("    " + $ia.Name)
}
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
  Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)
```

Saída:

```
For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
    MyApp
  OutputArtifacts:
    MyAppBuild
  PipelineContext:
    Name = Build
    PipelineName = CodePipelineDemo
    Stage = Build
```

- Para obter detalhes da API, consulte GetJobDetailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetPipeline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetPipeline`.

CLI

AWS CLI

Como visualizar a estrutura de um pipeline

Este exemplo retorna a estrutura de um pipeline chamado MyFirstPipeline.

Comando:

```
aws codepipeline get-pipeline --name MyFirstPipeline
```

Saída:

```
{
  "pipeline": {
      "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
      "stages": [
          {
              "name": "Source",
              "actions": [
                  {
                      "inputArtifacts": [],
                      "name": "Source",
                      "actionTypeId": {
                          "category": "Source",
                          "owner": "AWS",
                          "version": "1",
                          "provider": "S3"
                      },
                      "outputArtifacts": [
                          {
                              "name": "MyApp"
                          }
                      ],
                      "configuration": {
                          "S3Bucket": "awscodepipeline-demo-bucket",
                          "S3ObjectKey": "aws-codepipeline-s3-aws-
codedeploy_linux.zip"
                      },
                      "runOrder": 1
                  }
              ]
          },
          {
              "name": "Beta",
              "actions": [
                  {
                      "inputArtifacts": [
                          {
                              "name": "MyApp"
                          }
                      ],
                      "name": "CodePipelineDemoFleet",
                      "actionTypeId": {
                          "category": "Deploy",
                          "owner": "AWS",
```

```
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "CodePipelineDemoApplication",
                            "DeploymentGroupName": "CodePipelineDemoFleet"
                        },
                        "runOrder": 1
                    }
                ]
            }
        ],
        "artifactStore": {
            "type": "S3",
            "location": "codepipeline-us-east-1-11EXAMPLE11"
        },
        "name": "MyFirstPipeline",
        "version": 1
    }
}
```

- Para obter detalhes da API, consulte GetPipelineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações gerais sobre o pipeline especificado.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o pipeline especificado.

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
Write-Output ("  Location = " + $pipeline.ArtifactStore.Location)
Write-Output ("  Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
  Write-Output ("  Name = " + $stage.Name)
  Write-Output ("    Actions:")
  ForEach ($action in $stage.Actions) {
    Write-Output ("      Name = " + $action.Name)
 Write-Output ("        Category = " + $action.ActionTypeId.Category)
 Write-Output ("        Owner = " + $action.ActionTypeId.Owner)
 Write-Output ("        Provider = " + $action.ActionTypeId.Provider)
 Write-Output ("        Version = " + $action.ActionTypeId.Version)
 Write-Output ("        Configuration:")
 ForEach ($key in $action.Configuration.Keys) {
   $value = $action.Configuration.$key
   Write-Output ("          " + $key + " = " + $value)
 }
 Write-Output ("        InputArtifacts:")
 ForEach ($ia in $action.InputArtifacts) {
   Write-Output ("          " + $ia.Name)
 }
 ForEach ($oa in $action.OutputArtifacts) {
   Write-Output ("          " + $oa.Name)
 }
 Write-Output ("        RunOrder = " + $action.RunOrder)
  }
}
```

Saída:

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = amzn-s3-demo-bucket
  Type = S3
Stages:
  Name = Source
```

```
    Actions:
      Name = Source
        Category = Source
        Owner = ThirdParty
        Provider = GitHub
        Version = 1
        Configuration:
          Branch = master
          OAuthToken = ****
          Owner = my-user-name
          Repo = MyRepoName
        InputArtifacts:
          MyApp
        RunOrder = 1
  Name = Build
    Actions:
      Name = Build
        Category = Build
        Owner = Custom
        Provider = MyCustomProviderName
        Version = 1
        Configuration:
          ProjectName = MyProjectName
        InputArtifacts:
          MyApp
          MyAppBuild
        RunOrder = 1
  Name = Beta
    Actions:
      Name = CodePipelineDemoFleet
        Category = Deploy
        Owner = AWS
        Provider = CodeDeploy
        Version = 1
        Configuration:
          ApplicationName = CodePipelineDemoApplication
          DeploymentGroupName = CodePipelineDemoFleet
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
  Name = TestStage
    Actions:
      Name = MyJenkinsTestAction
        Category = Test
```

```
        Owner = Custom
        Provider = MyCustomTestProvider
        Version = 1
        Configuration:
          ProjectName = MyJenkinsProjectName
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
```

- Para obter detalhes da API, consulte GetPipelineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações gerais sobre o pipeline especificado.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o pipeline especificado.

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
Write-Output ("  Location = " + $pipeline.ArtifactStore.Location)
Write-Output ("  Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
  Write-Output ("  Name = " + $stage.Name)
  Write-Output ("    Actions:")
  ForEach ($action in $stage.Actions) {
    Write-Output ("      Name = " + $action.Name)
 Write-Output ("        Category = " + $action.ActionTypeId.Category)
 Write-Output ("        Owner = " + $action.ActionTypeId.Owner)
```

```
 Write-Output ("        Provider = " + $action.ActionTypeId.Provider)
 Write-Output ("        Version = " + $action.ActionTypeId.Version)
 Write-Output ("        Configuration:")
 ForEach ($key in $action.Configuration.Keys) {
   $value = $action.Configuration.$key
   Write-Output ("          " + $key + " = " + $value)
 }
 Write-Output ("        InputArtifacts:")
 ForEach ($ia in $action.InputArtifacts) {
   Write-Output ("          " + $ia.Name)
 }
 ForEach ($oa in $action.OutputArtifacts) {
   Write-Output ("          " + $oa.Name)
 }
 Write-Output ("        RunOrder = " + $action.RunOrder)
  }
}
```

Saída:

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = amzn-s3-demo-bucket
  Type = S3
Stages:
  Name = Source
    Actions:
      Name = Source
        Category = Source
        Owner = ThirdParty
        Provider = GitHub
        Version = 1
        Configuration:
          Branch = master
          OAuthToken = ****
          Owner = my-user-name
          Repo = MyRepoName
        InputArtifacts:
          MyApp
        RunOrder = 1
  Name = Build
```

```
    Actions:
      Name = Build
        Category = Build
        Owner = Custom
        Provider = MyCustomProviderName
        Version = 1
        Configuration:
          ProjectName = MyProjectName
        InputArtifacts:
          MyApp
          MyAppBuild
        RunOrder = 1
  Name = Beta
    Actions:
      Name = CodePipelineDemoFleet
        Category = Deploy
        Owner = AWS
        Provider = CodeDeploy
        Version = 1
        Configuration:
          ApplicationName = CodePipelineDemoApplication
          DeploymentGroupName = CodePipelineDemoFleet
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
  Name = TestStage
    Actions:
      Name = MyJenkinsTestAction
        Category = Test
        Owner = Custom
        Provider = MyCustomTestProvider
        Version = 1
        Configuration:
          ProjectName = MyJenkinsProjectName
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
```

- Para obter detalhes da API, consulte GetPipelineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetPipelineState`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetPipelineState`.

CLI

AWS CLI

Para obter informações sobre o estado de um pipeline

Este exemplo retorna o estado mais recente de um pipeline chamado MyFirstPipeline.

Comando:

```
aws codepipeline get-pipeline-state --name MyFirstPipeline
```

Saída:

```
{
 "created": 1446137312.204,
 "pipelineName": "MyFirstPipeline",
 "pipelineVersion": 1,
 "stageStates": [
  {
    "actionStates": [
      {
        "actionName": "Source",
        "entityUrl": "https://console.aws.amazon.com/s3/home?#",
        "latestExecution": {
          "lastStatusChange": 1446137358.328,
          "status": "Succeeded"
        }
      }
    ],
    "stageName": "Source"
  },
  {
    "actionStates": [
      {
        "actionName": "CodePipelineDemoFleet",
        "entityUrl": "https://console.aws.amazon.com/codedeploy/
home?#/applications/CodePipelineDemoApplication/deployment-groups/
CodePipelineDemoFleet",
```

```
        "latestExecution": {
          "externalExecutionId": "d-EXAMPLE",
          "externalExecutionUrl": "https://console.aws.amazon.com/codedeploy/
home?#/deployments/d-EXAMPLE",
          "lastStatusChange": 1446137493.131,
          "status": "Succeeded",
          "summary": "Deployment Succeeded"
        }
      }
    ],
    "inboundTransitionState": {
      "enabled": true
    },
    "stageName": "Beta"
  }
 ],
 "updated": 1446137312.204
}
```

- Para obter detalhes da API, consulte GetPipelineStateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações gerais sobre os estágios do pipeline especificado.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Saída:

```
Created         : 8/13/2015 10:17:54 PM
PipelineName    : CodePipelineDemo
PipelineVersion : 1
StageStates     : {Source, Build, Beta, TestStage}
Updated         : 8/13/2015 10:17:54 PM
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o estado do pipeline especificado.

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
  Write-Output ("For " + $stageState.StageName + ":")
  Write-Output ("  InboundTransitionState:")
  Write-Output ("    DisabledReason = " +
 $stageState.InboundTransitionState.DisabledReason)
  Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
  Write-Output ("    LastChangedAt = " +
 $stageState.InboundTransitionState.LastChangedAt)
  Write-Output ("    LastChangedBy = " +
 $stageState.InboundTransitionState.LastChangedBy)
  Write-Output ("  ActionStates:")
  ForEach ($actionState in $stageState.ActionStates) {
    Write-Output ("    For " + $actionState.ActionName + ":")
 Write-Output ("      CurrentRevision:")
    Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
 Write-Output ("        RevisionChangeId = " +
 $actionState.CurrentRevision.RevisionChangeId)
 Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
 Write-Output ("      EntityUrl = " + $actionState.EntityUrl)
 Write-Output ("      LatestExecution:")
    Write-Output ("        ErrorDetails:")
    Write-Output ("          Code = " +
 $actionState.LatestExecution.ErrorDetails.Code)
 Write-Output ("          Message = " +
 $actionState.LatestExecution.ErrorDetails.Message)
 Write-Output ("        ExternalExecutionId = " +
 $actionState.LatestExecution.ExternalExecutionId)
 Write-Output ("        ExternalExecutionUrl = " +
 $actionState.LatestExecution.ExternalExecutionUrl)
 Write-Output ("        LastStatusChange = " +
 $actionState.LatestExecution.LastStatusChange)
 Write-Output ("        PercentComplete = " +
 $actionState.LatestExecution.PercentComplete)
 Write-Output ("        Status = " + $actionState.LatestExecution.Status)
 Write-Output ("        Summary = " + $actionState.LatestExecution.Summary)
 Write-Output ("      RevisionUrl = " + $actionState.RevisionUrl)
  }
}
```

Saída:

```
For Source:
  InboundTransitionState:
```

```
    DisabledReason =
    Enabled =
    LastChangedAt =
    LastChangedBy =
  ActionStates:
    For Source:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/20/2015 23:28:45
        PercentComplete = 0
        Status = Succeeded
        Summary =
      RevisionUrl =
For Build:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For Build:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code = TimeoutError
          Message = The action failed because a job worker exceeded its time
 limit. If this is a custom action, make sure that the job worker is configured
 correctly.
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/21/2015 00:29:29
```

```
        PercentComplete = 0
        Status = Failed
        Summary =
      RevisionUrl =
For Beta:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For CodePipelineDemoFleet:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = d-D5LTCZXEX
        ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
        LastStatusChange = 07/08/2015 22:07:42
        PercentComplete = 0
        Status = Succeeded
        Summary = Deployment Succeeded
      RevisionUrl =
For TestStage:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For MyJenkinsTestAction25:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
```

```
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = 5
        ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
        LastStatusChange = 07/08/2015 22:09:03
        PercentComplete = 0
        Status = Succeeded
        Summary = Finished
      RevisionUrl =
```

- Para obter detalhes da API, consulte GetPipelineStateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações gerais sobre os estágios do pipeline especificado.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Saída:

```
Created         : 8/13/2015 10:17:54 PM
PipelineName    : CodePipelineDemo
PipelineVersion : 1
StageStates     : {Source, Build, Beta, TestStage}
Updated         : 8/13/2015 10:17:54 PM
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o estado do pipeline especificado.

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
  Write-Output ("For " + $stageState.StageName + ":")
  Write-Output ("  InboundTransitionState:")
  Write-Output ("    DisabledReason = " +
 $stageState.InboundTransitionState.DisabledReason)
  Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
  Write-Output ("    LastChangedAt = " +
 $stageState.InboundTransitionState.LastChangedAt)
  Write-Output ("    LastChangedBy = " +
 $stageState.InboundTransitionState.LastChangedBy)
```

```
  Write-Output ("  ActionStates:")
  ForEach ($actionState in $stageState.ActionStates) {
    Write-Output ("    For " + $actionState.ActionName + ":")
 Write-Output ("      CurrentRevision:")
    Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
 Write-Output ("        RevisionChangeId = " +
 $actionState.CurrentRevision.RevisionChangeId)
 Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
 Write-Output ("      EntityUrl = " + $actionState.EntityUrl)
 Write-Output ("      LatestExecution:")
    Write-Output ("        ErrorDetails:")
    Write-Output ("          Code = " +
 $actionState.LatestExecution.ErrorDetails.Code)
 Write-Output ("          Message = " +
 $actionState.LatestExecution.ErrorDetails.Message)
 Write-Output ("        ExternalExecutionId = " +
 $actionState.LatestExecution.ExternalExecutionId)
 Write-Output ("        ExternalExecutionUrl = " +
 $actionState.LatestExecution.ExternalExecutionUrl)
 Write-Output ("        LastStatusChange = " +
 $actionState.LatestExecution.LastStatusChange)
 Write-Output ("        PercentComplete = " +
 $actionState.LatestExecution.PercentComplete)
 Write-Output ("        Status = " + $actionState.LatestExecution.Status)
 Write-Output ("        Summary = " + $actionState.LatestExecution.Summary)
 Write-Output ("      RevisionUrl = " + $actionState.RevisionUrl)
  }
}
```

Saída:

```
For Source:
  InboundTransitionState:
    DisabledReason =
    Enabled =
    LastChangedAt =
    LastChangedBy =
  ActionStates:
    For Source:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
```

```
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/20/2015 23:28:45
        PercentComplete = 0
        Status = Succeeded
        Summary =
      RevisionUrl =
For Build:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For Build:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code = TimeoutError
          Message = The action failed because a job worker exceeded its time
 limit. If this is a custom action, make sure that the job worker is configured
 correctly.
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/21/2015 00:29:29
        PercentComplete = 0
        Status = Failed
        Summary =
      RevisionUrl =
For Beta:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
```

```
  ActionStates:
    For CodePipelineDemoFleet:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = d-D5LTCZXEX
        ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
        LastStatusChange = 07/08/2015 22:07:42
        PercentComplete = 0
        Status = Succeeded
        Summary = Deployment Succeeded
      RevisionUrl =
For TestStage:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For MyJenkinsTestAction25:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = 5
        ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
        LastStatusChange = 07/08/2015 22:09:03
        PercentComplete = 0
        Status = Succeeded
        Summary = Finished
```

```
        RevisionUrl =
```

- Para obter detalhes da API, consulte GetPipelineStateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListActionTypes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListActionTypes`.

CLI

AWS CLI

Como ver os tipos de ação disponíveis

Usado sozinho, o comando list-action-types retorna a estrutura de todas as ações disponíveis para sua conta. AWS Este exemplo usa a opção --action-owner-filter para retornar somente ações personalizadas.

Comando:

```
aws codepipeline list-action-types --action-owner-filter Custom
```

Saída:

```
{
  "actionTypes": [
      {
          "inputArtifactDetails": {
              "maximumCount": 5,
              "minimumCount": 0
          },
          "actionConfigurationProperties": [
              {
                  "secret": false,
                  "required": true,
                  "name": "MyJenkinsExampleBuildProject",
                  "key": true,
                  "queryable": true
              }
          ],
          "outputArtifactDetails": {
```

```
                "maximumCount": 5,
                "minimumCount": 0
            },
            "id": {
                "category": "Build",
                "owner": "Custom",
                "version": "1",
                "provider": "MyJenkinsProviderName"
            },
            "settings": {
                "entityUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}",
                "executionUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}/
{ExternalExecutionId}"
            }
        },
        {
            "inputArtifactDetails": {
                "maximumCount": 5,
                "minimumCount": 0
            },
            "actionConfigurationProperties": [
                {
                    "secret": false,
                    "required": true,
                    "name": "MyJenkinsExampleTestProject",
                    "key": true,
                    "queryable": true
                }
            ],
            "outputArtifactDetails": {
                "maximumCount": 5,
                "minimumCount": 0
            },
            "id": {
                "category": "Test",
                "owner": "Custom",
                "version": "1",
                "provider": "MyJenkinsProviderName"
            },
            "settings": {
                "entityUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}",
                "executionUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}/
{ExternalExecutionId}"
            }
```

```
      }
  ]
}
```

- Para obter detalhes da API, consulte ListActionTypesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre as ações disponíveis para o proprietário especificado.

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {
  Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +
 $actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version =
 " + $actionType.Id.Version + ":")
  Write-Output ("  ActionConfigurationProperties:")
  ForEach ($acp in $actionType.ActionConfigurationProperties) {
    Write-Output ("    For " + $acp.Name + ":")
    Write-Output ("      Description = " + $acp.Description)
    Write-Output ("      Key = " + $acp.Key)
    Write-Output ("      Queryable = " + $acp.Queryable)
    Write-Output ("      Required = " + $acp.Required)
    Write-Output ("      Secret = " + $acp.Secret)
  }
  Write-Output ("  InputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.InputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.InputArtifactDetails.MinimumCount)
  Write-Output ("  OutputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.OutputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.OutputArtifactDetails.MinimumCount)
  Write-Output ("  Settings:")
  Write-Output ("    EntityUrlTemplate = " +
 $actionType.Settings.EntityUrlTemplate)
  Write-Output ("    ExecutionUrlTemplate = " +
 $actionType.Settings.ExecutionUrlTemplate)
```

```
}
```

Saída:

```
For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS Elastic Beanstalk Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For EnvironmentName:
      Description = The AWS Elastic Beanstalk Environment name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS CodeDeploy Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For DeploymentGroupName:
      Description = The AWS CodeDeploy Deployment Group name
      Key = True
      Queryable = False
      Required = True
      Secret = False
```

```
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/
home?#/applications/{Config:ApplicationName}/deployment-groups/
{Config:DeploymentGroupName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}
```

- Para obter detalhes da API, consulte ListActionTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as ações disponíveis para o proprietário especificado.

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {
  Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +
 $actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version =
 " + $actionType.Id.Version + ":")
  Write-Output ("  ActionConfigurationProperties:")
  ForEach ($acp in $actionType.ActionConfigurationProperties) {
    Write-Output ("    For " + $acp.Name + ":")
    Write-Output ("      Description = " + $acp.Description)
    Write-Output ("      Key = " + $acp.Key)
    Write-Output ("      Queryable = " + $acp.Queryable)
    Write-Output ("      Required = " + $acp.Required)
    Write-Output ("      Secret = " + $acp.Secret)
  }
  Write-Output ("  InputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.InputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.InputArtifactDetails.MinimumCount)
  Write-Output ("  OutputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.OutputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.OutputArtifactDetails.MinimumCount)
```

```
  Write-Output ("  Settings:")
  Write-Output ("    EntityUrlTemplate = " +
 $actionType.Settings.EntityUrlTemplate)
  Write-Output ("    ExecutionUrlTemplate = " +
 $actionType.Settings.ExecutionUrlTemplate)
}
```

Saída:

```
For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS Elastic Beanstalk Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For EnvironmentName:
      Description = The AWS Elastic Beanstalk Environment name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS CodeDeploy Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For DeploymentGroupName:
```

```
      Description = The AWS CodeDeploy Deployment Group name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/
home?#/applications/{Config:ApplicationName}/deployment-groups/
{Config:DeploymentGroupName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}
```

- Para obter detalhes da API, consulte ListActionTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListPipelines`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListPipelines`.

CLI

AWS CLI

Como visualizar uma lista de pipelines

Este exemplo lista todos os AWS CodePipeline pipelines associados à AWS conta do usuário.

Comando:

```
aws codepipeline list-pipelines
```

Saída:

```
{
  "pipelines": [
      {
```

```
            "updated": 1439504274.641,
            "version": 1,
            "name": "MyFirstPipeline",
            "created": 1439504274.641
        },
        {
            "updated": 1436461837.992,
            "version": 2,
            "name": "MySecondPipeline",
            "created": 1436460801.381
        }
        ]
}
```

- Para obter detalhes da API, consulte ListPipelinesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma lista dos pipelines disponíveis.

```
Get-CPPipelineList
```

Saída:

```
Created                  Name                Updated                  Version
-------                  ----                -------                  -------
8/13/2015 10:17:54 PM    CodePipelineDemo    8/13/2015 10:17:54 PM    3
7/8/2015 2:41:53 AM      MyFirstPipeline     7/22/2015 9:06:37 PM     7
```

- Para obter detalhes da API, consulte ListPipelinesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista dos pipelines disponíveis.

```
Get-CPPipelineList
```

Saída:

```
Created                  Name                Updated                  Version
-------                  ----                -------                  -------
8/13/2015 10:17:54 PM    CodePipelineDemo    8/13/2015 10:17:54 PM    3
7/8/2015 2:41:53 AM      MyFirstPipeline     7/22/2015 9:06:37 PM     7
```

- Para obter detalhes da API, consulte ListPipelinesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PollForJobs`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PollForJobs`.

CLI

AWS CLI

Como visualizar todos os trabalhos disponíveis

Este exemplo retorna informações sobre quaisquer trabalhos nos quais um funcionário possa atuar. Este exemplo usa um arquivo JSON predefinido (MyActionTypeInfo.json) para fornecer informações sobre o tipo de ação para o qual o funcionário processa trabalhos. Esse comando é usado somente para ações personalizadas. Quando esse comando é chamado, AWS CodePipeline retorna credenciais temporárias para o bucket Amazon S3 usado para armazenar artefatos para o pipeline. Esse comando também retornará quaisquer valores secretos definidos para a ação, se houver algum definido.

Comando:

```
aws codepipeline poll-for-jobs --cli-input-json file://MyActionTypeInfo.json
```

Conteúdo de amostra do arquivo JSON:

```
{
  "actionTypeId": {
    "category": "Test",
    "owner": "Custom",
    "provider": "MyJenkinsProviderName",
    "version": "1"
  },
  "maxBatchSize": 5,
```

```
  "queryParam": {
      "ProjectName": "MyJenkinsTestProject"
  }
}
```

Saída:

```
{
 "jobs": [
  {
    "accountId": "111111111111",
    "data": {
      "actionConfiguration": {
        "__type": "ActionConfiguration",
        "configuration": {
          "ProjectName": "MyJenkinsExampleTestProject"
        }
      },
      "actionTypeId": {
        "__type": "ActionTypeId",
        "category": "Test",
        "owner": "Custom",
        "provider": "MyJenkinsProviderName",
        "version": "1"
      },
      "artifactCredentials": {
        "__type": "AWSSessionCredentials",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "sessionToken":
 "fICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQH
+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQE
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8
      },
      "inputArtifacts": [
        {
          "__type": "Artifact",
          "location": {
            "s3Location": {
              "bucketName": "codepipeline-us-east-1-11EXAMPLE11",
              "objectKey": "MySecondPipeline/MyAppBuild/EXAMPLE"
```

```
            },
            "type": "S3"
          },
          "name": "MyAppBuild"
        }
      ],
      "outputArtifacts": [],
      "pipelineContext": {
        "__type": "PipelineContext",
        "action": {
          "name": "MyJenkinsTest-Action"
        },
        "pipelineName": "MySecondPipeline",
        "stage": {
          "name": "Testing"
        }
      }
    },
    "id": "ef66c259-64f9-EXAMPLE",
    "nonce": "3"
  }
 ]
}
```

- Para obter detalhes da API, consulte PollForJobsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém informações sobre todos os trabalhos acionáveis para a categoria de ação, o proprietário, o provedor, a versão, os parâmetros de consulta especificados.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Saída:

```
AccountId       Data                                    Id
            Nonce
```

```
---------       ----                                --
            -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData    0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- Para obter detalhes da API, consulte PollForJobsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre todos os trabalhos acionáveis para a categoria de ação, o proprietário, o provedor, a versão, os parâmetros de consulta especificados.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Saída:

```
AccountId       Data                                Id
            Nonce
---------       ----                                --
            -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData    0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- Para obter detalhes da API, consulte PollForJobsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`StartPipelineExecution`** com uma CLI

Os exemplos de código a seguir mostram como usar o `StartPipelineExecution`.

CLI

AWS CLI

Para executar a revisão mais recente por meio de um pipeline

Este exemplo executa a revisão mais recente presente no estágio de origem de um pipeline por meio do pipeline chamado "MyFirstPipeline”.

Comando:

```
aws codepipeline start-pipeline-execution --name MyFirstPipeline
```

Saída:

```
{
  "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE"
}
```

- Para obter detalhes da API, consulte StartPipelineExecutionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo começa a executar o pipeline especificado.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Para obter detalhes da API, consulte StartPipelineExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo começa a executar o pipeline especificado.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Para obter detalhes da API, consulte StartPipelineExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdatePipeline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdatePipeline`.

CLI

AWS CLI

Para atualizar a estrutura de um pipeline

Este exemplo usa o comando update-pipeline com o argumento --cli-input-json. Este exemplo usa um arquivo JSON predefinido (MyFirstPipeline.json) para atualizar a estrutura de um pipeline. AWS CodePipeline reconhece o nome do pipeline contido no arquivo JSON e, em seguida, aplica todas as alterações dos campos modificados na estrutura do pipeline para atualizar o pipeline.

Use as diretrizes a seguir ao criar o arquivo JSON predefinido:

Se você estiver trabalhando com uma estrutura de pipeline recuperada usando o comando get-pipeline, deverá remover a seção de metadados da estrutura do pipeline no arquivo JSON (as linhas “metadata”: { } e os campos “created”, “pipelineARN” e “updated”). O nome do pipeline não pode ser alterado.

Comando:

```
aws codepipeline update-pipeline --cli-input-json file://MyFirstPipeline.json
```

Amostra de conteúdo do arquivo JSON:

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
  "stages": [
    {
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
```

```
              "name": "MyApp"
            }
          ],
          "configuration": {
            "S3Bucket": "awscodepipeline-demo-bucket2",
            "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Beta",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "name": "CodePipelineDemoFleet",
          "actionTypeId": {
            "category": "Deploy",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeDeploy"
          },
          "outputArtifacts": [],
          "configuration": {
            "ApplicationName": "CodePipelineDemoApplication",
            "DeploymentGroupName": "CodePipelineDemoFleet"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "codepipeline-us-east-1-11EXAMPLE11"
  },
  "name": "MyFirstPipeline",
  "version": 1
}
```

```
}
```

Saída:

```
{
 "pipeline": {
  "artifactStore": {
    "location": "codepipeline-us-east-1-11EXAMPLE11",
    "type": "S3"
  },
  "name": "MyFirstPipeline",
  "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
  "stages": [
    {
      "actions": [
        {
          "actionTypeId": {
            "__type": "ActionTypeId",
            "category": "Source",
            "owner": "AWS",
            "provider": "S3",
            "version": "1"
          },
          "configuration": {
            "S3Bucket": "awscodepipeline-demo-bucket2",
            "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
          },
          "inputArtifacts": [],
          "name": "Source",
          "outputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "runOrder": 1
        }
      ],
      "name": "Source"
    },
    {
      "actions": [
        {
          "actionTypeId": {
```

```
            "__type": "ActionTypeId",
            "category": "Deploy",
            "owner": "AWS",
            "provider": "CodeDeploy",
            "version": "1"
          },
          "configuration": {
            "ApplicationName": "CodePipelineDemoApplication",
            "DeploymentGroupName": "CodePipelineDemoFleet"
          },
          "inputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "name": "CodePipelineDemoFleet",
          "outputArtifacts": [],
          "runOrder": 1
        }
      ],
      "name": "Beta"
    }
  ],
  "version": 3
 }
}
```

- Para obter detalhes da API, consulte UpdatePipelineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza o pipeline existente especificado com as configurações especificadas.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
```

```
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object
 Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName",
 "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
```

```
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Para obter detalhes da API, consulte UpdatePipelineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza o pipeline existente especificado com as configurações especificadas.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object
 Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName",
 "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
```

```
$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Para obter detalhes da API, consulte UpdatePipelineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para o Amazon Cognito Identity usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Cognito Identity com um kit de desenvolvimento AWS de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Cognito Identity: mais informações sobre o Amazon Cognito Identity.
- Referência da API Amazon Cognito Identity: detalhes sobre todas as ações disponíveis do Amazon Cognito Identity.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Amazon Cognito Identity usando AWS SDKs
  - Ações para o Amazon Cognito Identity usando AWS SDKs
    - Use CreateIdentityPoolcom um AWS SDK ou CLI
    - Use DeleteIdentityPoolcom um AWS SDK ou CLI
    - Usar DescribeIdentityPool com uma CLI
    - Use GetCredentialsForIdentitycom um AWS SDK
    - Usar GetIdentityPoolRoles com uma CLI
    - Use ListIdentityPoolscom um AWS SDK ou CLI
    - Usar SetIdentityPoolRoles com uma CLI
    - Usar UpdateIdentityPool com uma CLI
- Cenários para o Amazon Cognito Identity usando AWS SDKs
  - Criar uma aplicação de exploração do Amazon Textract

Exemplos básicos do Amazon Cognito Identity usando AWS SDKs

Os exemplos de código a seguir mostram como usar os fundamentos do Amazon Cognito Identity com SDKs da AWS .

Exemplos

- Ações para o Amazon Cognito Identity usando AWS SDKs
  - Use CreateIdentityPoolcom um AWS SDK ou CLI

- Use DeleteIdentityPoolcom um AWS SDK ou CLI
- Usar DescribeIdentityPool com uma CLI
- Use GetCredentialsForIdentitycom um AWS SDK
- Usar GetIdentityPoolRoles com uma CLI
- Use ListIdentityPoolscom um AWS SDK ou CLI
- Usar SetIdentityPoolRoles com uma CLI
- Usar UpdateIdentityPool com uma CLI

Ações para o Amazon Cognito Identity usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Cognito Identity com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API de identidade do Amazon Cognito e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Cognito Identity usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para ter uma lista completa, consulte a Referência de API do Amazon Cognito Identity.

Exemplos

- Use CreateIdentityPoolcom um AWS SDK ou CLI
- Use DeleteIdentityPoolcom um AWS SDK ou CLI
- Usar DescribeIdentityPool com uma CLI
- Use GetCredentialsForIdentitycom um AWS SDK
- Usar GetIdentityPoolRoles com uma CLI
- Use ListIdentityPoolscom um AWS SDK ou CLI
- Usar SetIdentityPoolRoles com uma CLI
- Usar UpdateIdentityPool com uma CLI

Use **`CreateIdentityPool`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateIdentityPool`.

CLI

AWS CLI

Como criar um banco de identidades com o provedor de banco de identidades Cognito

Este exemplo cria um grupo de identidades chamado MyIdentityPool. Ele tem um provedor de banco de identidades Cognito. Identidades não autenticadas não são permitidas.

Comando:

```
aws cognito-identity create-identity-pool --identity-pool-
name MyIdentityPool --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_aaaaaaaaa",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

Saída:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- Para obter detalhes da API, consulte CreateIdentityPoolem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

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

- Para obter detalhes da API, consulte CreateIdentityPoola Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Cria um novo banco de identidades que permite identidades não autenticadas.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Saída:

```
LoggedAt                       : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName               : CommonTests13
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 136
HttpStatusCode                 : OK
```

- Para obter detalhes da API, consulte CreateIdentityPoolem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Cria um novo banco de identidades que permite identidades não autenticadas.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Saída:

```
LoggedAt                       : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName               : CommonTests13
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 136
HttpStatusCode                 : OK
```

- Para obter detalhes da API, consulte CreateIdentityPoolem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSCognitoIdentity


    /// Create a new identity pool and return its ID.
    ///
    /// - Parameters:
    ///     - name: The name to give the new identity pool.
    ///
    /// - Returns: A string containing the newly created pool's ID, or `nil`
    ///   if an error occurred.
    ///
    func createIdentityPool(name: String) async throws -> String? {
        do {
            let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
 "com.exampleco.CognitoIdentityDemo",
                                                           identityPoolName:
 name)

            let result = try await
 cognitoIdentityClient.createIdentityPool(input: cognitoInputCall)
            guard let poolId = result.identityPoolId else {
                return nil
            }

            return poolId
        } catch {
            print("ERROR: createIdentityPool:", dump(error))
            throw error
        }
    }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Swift.
- Para obter detalhes da API, consulte CreateIdentityPoola referência da API AWS SDK for Swift.

Use **`DeleteIdentityPool`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteIdentityPool`.

CLI

AWS CLI

Como excluir um banco de identidades

O exemplo `delete-identity-pool` a seguir exclui o banco de identidades especificado.

Comando:

```
aws cognito-identity delete-identity-pool \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteIdentityPoolem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
```

```
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

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

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

                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {
```

```
            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

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

- Para obter detalhes da API, consulte DeleteIdentityPoola Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Exclui um banco de identidades específico.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Para obter detalhes da API, consulte DeleteIdentityPoolem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Exclui um banco de identidades específico.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Para obter detalhes da API, consulte DeleteIdentityPoolem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSCognitoIdentity


    /// Delete the specified identity pool.
    ///
    /// - Parameters:
    ///   - id: The ID of the identity pool to delete.
    ///
    func deleteIdentityPool(id: String) async throws {
        do {
            let input = DeleteIdentityPoolInput(
                identityPoolId: id
            )

            _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
        } catch {
            print("ERROR: deleteIdentityPool:", dump(error))
            throw error
        }
    }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Swift.
- Para obter detalhes da API, consulte DeleteIdentityPoola referência da API AWS SDK for Swift.

Usar **`DescribeIdentityPool`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeIdentityPool`.

CLI

AWS CLI

Para descrever um banco de identidades

Este exemplo descreve um banco de identidades.

Comando:

```
aws cognito-identity describe-identity-pool --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

Saída:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- Para obter detalhes da API, consulte DescribeIdentityPoolem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Recupera informações sobre um banco de identidades específico por meio de seu ID.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Saída:

```
LoggedAt                        : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : CommonTests1
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 142
HttpStatusCode                  : OK
```

- Para obter detalhes da API, consulte DescribeIdentityPoolem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Recupera informações sobre um banco de identidades específico por meio de seu ID.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Saída:

```
LoggedAt                        : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : CommonTests1
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 142
HttpStatusCode                  : OK
```

- Para obter detalhes da API, consulte DescribeIdentityPoolem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetCredentialsForIdentity`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetCredentialsForIdentity`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
```

```
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

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

- Para obter detalhes da API, consulte GetCredentialsForIdentitya Referência AWS SDK for Java 2.x da API.

Usar **`GetIdentityPoolRoles`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetIdentityPoolRoles`.

CLI

AWS CLI

Para obter funções no banco de identidades

Este exemplo lista bancos de identidades.

Comando:

```
aws cognito-identity get-identity-pool-roles --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

Saída:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "Roles": {
      "authenticated": "arn:aws:iam::111111111111:role/Cognito_MyIdentityPoolAuth_Role",
      "unauthenticated": "arn:aws:iam::111111111111:role/Cognito_MyIdentityPoolUnauth_Role"
  }
}
```

- Para obter detalhes da API, consulte GetIdentityPoolRolesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Obtém as informações sobre as funções de um banco de identidades específico.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Saída:

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- Para obter detalhes da API, consulte GetIdentityPoolRolesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Obtém as informações sobre as funções de um banco de identidades específico.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Saída:

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- Para obter detalhes da API, consulte GetIdentityPoolRolesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListIdentityPools`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListIdentityPools`.

CLI

AWS CLI

Para listar bancos de identidades

Este exemplo lista bancos de identidades. Há no máximo vinte identidades listadas.

Comando:

```
aws cognito-identity list-identity-pools --max-results 20
```

Saída:

```
{
  "IdentityPools": [
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "MyIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "AnotherIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "IdentityPoolRegionA"
      }
  ]
}
```

- Para obter detalhes da API, consulte ListIdentityPoolsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
```

```
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

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

- Para obter detalhes da API, consulte ListIdentityPoolsa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Recupera uma lista de bancos de identidades existentes.

```
Get-CGIIdentityPoolList
```

Saída:

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- Para obter detalhes da API, consulte ListIdentityPoolsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Recupera uma lista de bancos de identidades existentes.

```
Get-CGIIdentityPoolList
```

Saída:

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
```

```
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- Para obter detalhes da API, consulte ListIdentityPoolsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        let listPoolsInput = ListIdentityPoolsInput(maxResults: 25)
        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'nextToken' field in
 "ListIdentityPoolsOutput".
        let pages = cognitoIdentityClient.listIdentityPoolsPaginated(input:
 listPoolsInput)

        do {
            for try await page in pages {
                guard let identityPools = page.identityPools else {
                    print("ERROR: listIdentityPoolsPaginated returned nil
 contents.")
                    continue
                }
```

```
                /// Read pages of identity pools from Cognito until one is found
                /// whose name matches the one specified in the `name` parameter.
                /// Return the matching pool's ID.

                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }
        } catch {
            print("ERROR: getIdentityPoolID:", dump(error))
            throw error
        }

        return nil
    }
```

Obtenha o ID de um banco de identidades existente ou crie-o se ainda não existir.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned
    ///
    /// - Returns: A string containing the ID of the specified identity pool.
    ///   Returns `nil` if there's an error or if the pool isn't found.
    ///
    public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
        // See if the pool already exists. If it doesn't, create it.

        do {
            guard let poolId = try await getIdentityPoolID(name: name) else {
                return try await createIdentityPool(name: name)
            }

            return poolId
```

```
        } catch {
            print("ERROR: getOrCreateIdentityPoolID:", dump(error))
            throw error
        }
    }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Swift.
- Para obter detalhes da API, consulte ListIdentityPoolsa referência da API AWS SDK for Swift.

Usar **`SetIdentityPoolRoles`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetIdentityPoolRoles`.

CLI

AWS CLI

Como definir funções do banco de identidades

O exemplo `set-identity-pool-roles` a seguir define funções para um banco de identidades.

```
aws cognito-identity set-identity-pool-roles \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111" \
    --roles authenticated="arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role"
```

- Para obter detalhes da API, consulte SetIdentityPoolRolesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Configura o banco de identidades específico para ter um perfil do IAM não autenticado.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Para obter detalhes da API, consulte SetIdentityPoolRolesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Configura o banco de identidades específico para ter um perfil do IAM não autenticado.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Para obter detalhes da API, consulte SetIdentityPoolRolesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateIdentityPool`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateIdentityPool`.

CLI

AWS CLI

Para atualizar um banco de identidades

Este exemplo atualiza um banco de identidades. Ele define o nome como MyIdentityPool. Ele tem um provedor de banco de identidades Cognito. Ele não permite identidades não autenticadas.

Comando:

```
aws cognito-identity update-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111" --identity-pool-
name "MyIdentityPool" --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

Saída:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- Para obter detalhes da API, consulte UpdateIdentityPoolem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Atualiza algumas das propriedades do banco de identidades, neste caso, o nome do banco de identidades.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Saída:

```
LoggedAt                       : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : NewPoolName
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 135
HttpStatusCode                 : OK
```

- Para obter detalhes da API, consulte UpdateIdentityPoolem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Atualiza algumas das propriedades do banco de identidades, neste caso, o nome do banco de identidades.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Saída:

```
LoggedAt                       : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : NewPoolName
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 135
HttpStatusCode                 : OK
```

- Para obter detalhes da API, consulte UpdateIdentityPoolem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para o Amazon Cognito Identity usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Cognito Identity com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon Cognito Identity ou em combinação com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação de exploração do Amazon Textract

Criar uma aplicação de exploração do Amazon Textract

Os exemplos de código a seguir mostram como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.

- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Exemplos de código para o Amazon Cognito Identity Provider usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Cognito Identity Provider com um kit de desenvolvimento AWS de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Cognito Identity Provider: mais informações sobre o Amazon Cognito Identity Provider.
- Referência da API Amazon Cognito Identity Provider: detalhes sobre todas as ações disponíveis do Amazon Cognito Identity Provider.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon Cognito Identity Provider usando AWS SDKs
  - Olá, Amazon Cognito
  - Ações para o Amazon Cognito Identity Provider usando AWS SDKs
    - Use AdminCreateUsercom um AWS SDK ou CLI
    - Use AdminGetUsercom um AWS SDK ou CLI
    - Use AdminInitiateAuthcom um AWS SDK ou CLI
    - Use AdminRespondToAuthChallengecom um AWS SDK ou CLI
    - Use AdminSetUserPasswordcom um AWS SDK ou CLI
    - Use AssociateSoftwareTokencom um AWS SDK ou CLI
    - Use ConfirmDevicecom um AWS SDK ou CLI
    - Use ConfirmForgotPasswordcom um AWS SDK ou CLI
    - Use ConfirmSignUpcom um AWS SDK ou CLI
    - Use CreateUserPoolcom um AWS SDK ou CLI
    - Use CreateUserPoolClientcom um AWS SDK ou CLI
    - Use DeleteUsercom um AWS SDK ou CLI
    - Use ForgotPasswordcom um AWS SDK ou CLI
    - Use InitiateAuthcom um AWS SDK ou CLI
    - Use ListUserPoolscom um AWS SDK ou CLI
    - Use ListUserscom um AWS SDK ou CLI
    - Use ResendConfirmationCodecom um AWS SDK ou CLI
    - Use RespondToAuthChallengecom um AWS SDK ou CLI
    - Use SignUpcom um AWS SDK ou CLI
    - Use UpdateUserPoolcom um AWS SDK ou CLI
    - Use VerifySoftwareTokencom um AWS SDK ou CLI
  - Cenários para o Amazon Cognito Identity Provider usando AWS SDKs

- Confirme automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK
- Migre automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK
- Conceitos básicos dos grupos de usuários
- Inscrever um usuário com um grupo de usuários do Amazon Cognito que exige MFA usando um AWS SDK
- Usar bancos de identidades e fluxos de identidades do Amazon Cognito
- Grave dados de atividades personalizados com uma função Lambda após a autenticação do usuário do Amazon Cognito usando um AWS SDK

Exemplos básicos para o Amazon Cognito Identity Provider usando AWS SDKs

Os exemplos de código a seguir mostram como usar os fundamentos do Amazon Cognito Identity Provider com AWS SDKs.

Exemplos

- Olá, Amazon Cognito
- Ações para o Amazon Cognito Identity Provider usando AWS SDKs
  - Use AdminCreateUsercom um AWS SDK ou CLI
  - Use AdminGetUsercom um AWS SDK ou CLI
  - Use AdminInitiateAuthcom um AWS SDK ou CLI
  - Use AdminRespondToAuthChallengecom um AWS SDK ou CLI
  - Use AdminSetUserPasswordcom um AWS SDK ou CLI
  - Use AssociateSoftwareTokencom um AWS SDK ou CLI
  - Use ConfirmDevicecom um AWS SDK ou CLI
  - Use ConfirmForgotPasswordcom um AWS SDK ou CLI
  - Use ConfirmSignUpcom um AWS SDK ou CLI
  - Use CreateUserPoolcom um AWS SDK ou CLI
  - Use CreateUserPoolClientcom um AWS SDK ou CLI
  - Use DeleteUsercom um AWS SDK ou CLI

- Use ForgotPasswordcom um AWS SDK ou CLI
- Use InitiateAuthcom um AWS SDK ou CLI
- Use ListUserPoolscom um AWS SDK ou CLI
- Use ListUserscom um AWS SDK ou CLI
- Use ResendConfirmationCodecom um AWS SDK ou CLI
- Use RespondToAuthChallengecom um AWS SDK ou CLI
- Use SignUpcom um AWS SDK ou CLI
- Use UpdateUserPoolcom um AWS SDK ou CLI
- Use VerifySoftwareTokencom um AWS SDK ou CLI

Olá, Amazon Cognito

Os exemplos de código a seguir mostram como começar a usar o Amazon Cognito.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

# Set this project's name.
project("hello_cognito")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)
```

```
# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_cognito.cpp.

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito
 client and lists the Amazon Cognito
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
 listUserPoolsRequest;
            if (!nextToken.empty()) {
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
 listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
 listUserPoolsOutcome.GetResult().GetUserPools()) {

                    userPools.push_back(userPool.GetName());
                }
```

```
                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "ListUserPools error: " <<
 listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListUserPoolsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- Para obter detalhes da API, consulte ListUserPoolsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
```

```
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

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

- Para obter detalhes da API, consulte ListUserPoolsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
```

```
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});

  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- Para obter detalhes da API, consulte ListUserPoolsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")
```

```
# Create a PageIterator from the paginator
page_iterator = paginator.paginate(MaxResults=10)

# Initialize variables for pagination
user_pools = []

# Handle pagination
for page in page_iterator:
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- Para obter detalhes da API, consulte a ListUserPoolsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
```

```
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

- Para obter detalhes da API, consulte ListUserPoolsa Referência AWS SDK para Ruby da API.

Ações para o Amazon Cognito Identity Provider usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Cognito Identity Provider com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Provedor de identidade do Amazon Cognito e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Cognito Identity Provider usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para ter uma lista completa, consulte a Referência de API do Amazon Cognito Identity Provider.

Exemplos

- Use AdminCreateUsercom um AWS SDK ou CLI
- Use AdminGetUsercom um AWS SDK ou CLI
- Use AdminInitiateAuthcom um AWS SDK ou CLI
- Use AdminRespondToAuthChallengecom um AWS SDK ou CLI
- Use AdminSetUserPasswordcom um AWS SDK ou CLI
- Use AssociateSoftwareTokencom um AWS SDK ou CLI
- Use ConfirmDevicecom um AWS SDK ou CLI
- Use ConfirmForgotPasswordcom um AWS SDK ou CLI
- Use ConfirmSignUpcom um AWS SDK ou CLI
- Use CreateUserPoolcom um AWS SDK ou CLI
- Use CreateUserPoolClientcom um AWS SDK ou CLI
- Use DeleteUsercom um AWS SDK ou CLI
- Use ForgotPasswordcom um AWS SDK ou CLI
- Use InitiateAuthcom um AWS SDK ou CLI
- Use ListUserPoolscom um AWS SDK ou CLI
- Use ListUserscom um AWS SDK ou CLI
- Use ResendConfirmationCodecom um AWS SDK ou CLI
- Use RespondToAuthChallengecom um AWS SDK ou CLI
- Use SignUpcom um AWS SDK ou CLI
- Use UpdateUserPoolcom um AWS SDK ou CLI
- Use VerifySoftwareTokencom um AWS SDK ou CLI

Use **`AdminCreateUser`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AdminCreateUser`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos dos grupos de usuários
- Grave dados de atividades personalizados com uma função do Lambda após a autenticação do usuário do Amazon Cognito

CLI

AWS CLI

Para criar um usuário

O `admin-create-user` exemplo a seguir cria um usuário com as configurações especificadas de endereço de e-mail e número de telefone.

```
aws cognito-idp admin-create-user \
    --user-pool-id us-west-2_aaaaaaaaa \
    --username diego \
    --user-attributes Name=email,Value=diego@example.com
 Name=phone_number,Value="+15555551212" \
    --message-action SUPPRESS
```

Saída:

```
{
    "User": {
        "Username": "diego",
        "Attributes": [
            {
                "Name": "sub",
                "Value": "7325c1de-b05b-4f84-b321-9adc6e61f4a2"
            },
            {
                "Name": "phone_number",
                "Value": "+15555551212"
            },
            {
                "Name": "email",
                "Value": "diego@example.com"
            }
```

```
        ],
        "UserCreateDate": 1548099495.428,
        "UserLastModifiedDate": 1548099495.428,
        "Enabled": true,
        "UserStatus": "FORCE_CHANGE_PASSWORD"
    }
}
```

- Para obter detalhes da API, consulte AdminCreateUserem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
```

```
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- Para obter detalhes da API, consulte AdminCreateUsera Referência AWS SDK para Go da API.

Use **`AdminGetUser`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AdminGetUser`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }
```

- Para obter detalhes da API, consulte AdminGetUsera Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                        outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Para obter detalhes da API, consulte AdminGetUsera Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como obter um usuário

Este exemplo obtém informações sobre o nome de usuário jane@example.com.

Comando:

```
aws cognito-idp admin-get-user --user-pool-id us-west-2_aaaaaaaaa --
username jane@example.com
```

Saída:

```
{
  "Username": "4320de44-2322-4620-999b-5e2e1c8df013",
  "Enabled": true,
  "UserStatus": "FORCE_CHANGE_PASSWORD",
  "UserCreateDate": 1548108509.537,
  "UserAttributes": [
      {
          "Name": "sub",
          "Value": "4320de44-2322-4620-999b-5e2e1c8df013"
      },
      {
          "Name": "email_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number",
          "Value": "+01115551212"
      },
      {
          "Name": "email",
          "Value": "jane@example.com"
      }
  ],
  "UserLastModifiedDate": 1548108509.537
```

```
}
```

- Para obter detalhes da API, consulte AdminGetUserem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
           String poolId) {
       try {
           AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                   .username(userName)
                   .userPoolId(poolId)
                   .build();

           AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
           System.out.println("User status " + response.userStatusAsString());

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

- Para obter detalhes da API, consulte AdminGetUsera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const adminGetUser = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte AdminGetUsera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
```

```
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- Para obter detalhes da API, consulte a AdminGetUserreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
```

```
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
```

```
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- Para obter detalhes da API, consulte a AdminGetUserReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
```

```
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }
```

- Para obter detalhes da API, consulte AdminGetUsera referência da API AWS SDK for Swift.

Use **`AdminInitiateAuth`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AdminInitiateAuth`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Initiate an admin auth request.
```

```
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
 userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }
```

- Para obter detalhes da API, consulte AdminInitiateAutha Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

   Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
   request.SetClientId(clientID);
   request.SetUserPoolId(userPoolID);
   request.AddAuthParameters("USERNAME", userName);
   request.AddAuthParameters("PASSWORD", password);
   request.SetAuthFlow(

Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


   Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
           client.AdminInitiateAuth(request);

   if (outcome.IsSuccess()) {
       std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
       sessionResult = outcome.GetResult().GetSession();
   }
   else {
       std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                 << outcome.GetError().GetMessage()
                 << std::endl;
   }
```

- Para obter detalhes da API, consulte AdminInitiateAuth a Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como fazer login como usuário administrador

O exemplo de `admin-initiate-auth` a seguir faz login com o usuário diego@example.com. Esse exemplo também inclui metadados para proteção contra ameaças e ClientMetadata para acionadores Lambda. O usuário está configurado para MFA TOTP e

recebe um desafio para fornecer um código da aplicação autenticadora antes de concluir a autenticação.

```
aws cognito-idp admin-initiate-auth \
    --user-pool-id us-west-2_EXAMPLE \
    --client-id 1example23456789 \
    --auth-flow ADMIN_USER_PASSWORD_AUTH \
    --auth-parameters USERNAME=diego@example.com,PASSWORD="My@Example
$Password3!",SECRET_HASH=ExampleEncodedClientIdSecretAndUsername= \
    --context-data="{\"EncodedData\":\"abc123example\",\"HttpHeaders\":
[{\"headerName\":\"UserAgent\",\"headerValue\":\"Mozilla/5.0 (Windows NT
 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0\"}],\"IpAddress\":
\"192.0.2.1\",\"ServerName\":\"example.com\",\"ServerPath\":\"/login\"}" \
    --client-metadata="{\"MyExampleKey\": \"MyExampleValue\"}"
```

Saída:

```
{
    "ChallengeName": "SOFTWARE_TOKEN_MFA",
    "Session": "AYABeExample...",
    "ChallengeParameters": {
        "FRIENDLY_DEVICE_NAME": "MyAuthenticatorApp",
        "USER_ID_FOR_SRP": "diego@example.com"
    }
}
```

Consulte mais informações em Fluxo de autenticação de administração no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte AdminInitiateAuthem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId)
{
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);

           AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                   .clientId(clientId)
                   .userPoolId(userPoolId)
                   .authParameters(authParameters)
                   .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                   .build();

           AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
           System.out.println("Result Challenge is : " +
response.challengeName());
           return response;

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

       return null;
   }
```

- Para obter detalhes da API, consulte AdminInitiateAuth a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte AdminInitiateAutha Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkAuthMethod(
    clientIdVal: String,
```

```
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- Para obter detalhes da API, consulte a AdminInitiateAuthreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
```

```
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
```

```
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Para obter detalhes da API, consulte a AdminInitiateAuthReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Set up authentication parameters
        DATA(lt_auth_params) = VALUE /aws1/
cl_cgpauthparamstype_w=>tt_authparameterstype(
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'PASSWORD'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_password ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpauthparamstype_w( iv_secret_hash )
          ) INTO TABLE lt_auth_params.
        ENDIF.

        oo_result = lo_cgp->admininitiateauth(
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_authflow = 'ADMIN_USER_PASSWORD_AUTH'
          it_authparameters = lt_auth_params
        ).

        DATA(lv_challenge) = oo_result->get_challengename( ).

        IF lv_challenge IS INITIAL.
          MESSAGE 'User successfully signed in.' TYPE 'I'.
        ELSE.
```

```
          MESSAGE |Authentication challenge required: { lv_challenge }.| TYPE
 'I'.
        ENDIF.

      CATCH /aws1/cx_cgpusernotfoundex INTO DATA(lo_user_ex).
        MESSAGE |User { iv_user_name } not found.| TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized. Check credentials.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AdminInitiateAuthreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Begin an authentication session.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
```

```
                        userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte AdminInitiateAuth a referência da API AWS SDK for Swift.

Use **AdminRespondToAuthChallenge**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AdminRespondToAuthChallenge`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Respond to an admin authentication challenge.
/// </summary>
/// <param name="userName">The name of the user.</param>
/// <param name="clientId">The client ID.</param>
/// <param name="mfaCode">The multi-factor authentication code.</param>
/// <param name="session">The current application session.</param>
/// <param name="clientId">The user pool ID.</param>
/// <returns>The result of the authentication response.</returns>
public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
    string userName,
    string clientId,
    string mfaCode,
    string session,
    string userPoolId)
{
    Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

    var challengeResponses = new Dictionary<string, string>();
    challengeResponses.Add("USERNAME", userName);
    challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallengea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
```

```
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
        request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallengea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como responder a um desafio de autenticação

Há muitas maneiras de responder a diferentes desafios de autenticação, dependendo do fluxo de autenticação, da configuração do grupo de usuários e das configurações do usuário.

O exemplo de `admin-respond-to-auth-challenge` a seguir fornece um código de MFA TOTP para diego@example.com e conclui o login. Esse grupo de usuários tem a memorização de dispositivos ativada, então o resultado da autenticação também retorna uma nova chave de dispositivo.

```
aws cognito-idp admin-respond-to-auth-challenge \
    --user-pool-id us-west-2_EXAMPLE \
    --client-id 1example23456789 \
    --challenge-name SOFTWARE_TOKEN_MFA \
    --challenge-
responses USERNAME=diego@example.com,SOFTWARE_TOKEN_MFA_CODE=000000 \
    --session AYABeExample...
```

Saída:

```
{
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-ExAmPlE1"
        }
    }
}
```

Consulte mais informações em Fluxo de autenticação de administração no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte AdminRespondToAuthChallengeem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Respond to an authentication challenge.
   public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
           String userName, String clientId, String mfaCode, String session) {
       System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
       Map<String, String> challengeResponses = new HashMap<>();

       challengeResponses.put("USERNAME", userName);
       challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

       AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
               .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
               .clientId(clientId)
               .challengeResponses(challengeResponses)
               .session(session)
               .build();

       AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
               .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
               + respondToAuthChallengeResult.authenticationResult());
   }
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallengea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const adminRespondToAuthChallenge = ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallengea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- Para obter detalhes da API, consulte a AdminRespondToAuthChallengereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Responda a um desafio de MFA fornecendo um código gerado por uma aplicação de MFA associada.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
 of
        a two-factor sign-in. When sign-in is successful, it returns an access
 token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
```

```
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

- Para obter detalhes da API, consulte a AdminRespondToAuthChallengeReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Build challenge responses
        DATA(lt_challenge_responses) = VALUE /aws1/
cl_cgpchallengerspstyp00=>tt_challengeresponsestype(
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpchallengerspstyp00( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'SOFTWARE_TOKEN_MFA_CODE'
              value = NEW /aws1/cl_cgpchallengerspstyp00( iv_mfa_code ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpchallengerspstyp00( iv_secret_hash )
          ) INTO TABLE lt_challenge_responses.
        ENDIF.

        DATA(lo_result) = lo_cgp->adminrespondtoauthchallenge(
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_challengename = 'SOFTWARE_TOKEN_MFA'
          it_challengeresponses = lt_challenge_responses
```

```
          iv_session = iv_session
        ).

        oo_auth_result = lo_result->get_authenticationresult( ).

        IF oo_auth_result IS BOUND.
          MESSAGE 'MFA challenge completed successfully.' TYPE 'I'.
        ELSE.
          " Another challenge might be required
          DATA(lv_next_challenge) = lo_result->get_challengename( ).
          MESSAGE |Additional challenge required: { lv_next_challenge }.| TYPE
 'I'.
        ENDIF.

      CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
        MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

      CATCH /aws1/cx_cgpexpiredcodeex INTO DATA(lo_expired_ex).
        MESSAGE 'MFA code has expired.' TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized. Check MFA configuration.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AdminRespondToAuthChallengereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider
```

```
    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                     clientId: String, userPoolId: String,
mfaCode: String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
```

```
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallengea referência da API AWS SDK for Swift.

Use **`AdminSetUserPassword`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AdminSetUserPassword`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Grave dados de atividades personalizados com uma função do Lambda após a autenticação do usuário do Amazon Cognito

CLI

AWS CLI

Como definir uma senha de usuário como administrador

O exemplo de `admin-set-user-password` a seguir define permanentemente a senha para diego@example.com.

```
aws cognito-idp admin-set-user-password \
    --user-pool-id us-west-2_EXAMPLE \
    --username diego@example.com \
    --password MyExamplePassword1! \
    --permanent
```

Este comando não produz saída.

Consulte mais informações em Passwords, password recovery, and password policies no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte AdminSetUserPasswordem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

- Para obter detalhes da API, consulte AdminSetUserPasswordda Referência AWS SDK para Go da API.

Use **`AssociateSoftwareToken`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AssociateSoftwareToken`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }
```

- Para obter detalhes da API, consulte AssociateSoftwareTokena Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
```

```
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte AssociateSoftwareTokena Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como gerar uma chave secreta para uma aplicação autenticadora de MFA

O exemplo de `associate-software-token` a seguir gera uma chave privada TOTP para um usuário que fez login e recebeu um token de acesso. A chave privada resultante pode ser inserida manualmente em uma aplicação autenticadora ou as aplicações podem renderizá-la como um código QR que o usuário pode escanear.

```
aws cognito-idp associate-software-token \
    --access-token eyJra456defEXAMPLE
```

Saída:

```
{
    "SecretCode": "QWERTYUIOP123456EXAMPLE"
}
```

Consulte mais informações em MFA de token de software TOTP no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte AssociateSoftwareTokenem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
       AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
               .session(session)
               .build();

       AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
               .associateSoftwareToken(softwareTokenRequest);
       String secretCode = tokenResponse.secretCode();
       System.out.println("Enter this token into Google Authenticator");
       System.out.println(secretCode);
       return tokenResponse.session();
   }
```

- Para obter detalhes da API, consulte AssociateSoftwareTokena Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const associateSoftwareToken = (session) => {
```

```
  const client = new CognitoIdentityProviderClient({});
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte AssociateSoftwareTokena Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}
```

- Para obter detalhes da API, consulte a AssociateSoftwareTokenreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
 user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
```

```
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Para obter detalhes da API, consulte a AssociateSoftwareTokenReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_cgp->associatesoftwaretoken(
          iv_session = iv_session
        ).

        ov_secret_code = lo_result->get_secretcode( ).

        MESSAGE 'MFA secret code generated successfully.' TYPE 'I'.

      CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
        MESSAGE 'Session not found or expired.' TYPE 'E'.
```

```
      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized to associate software token.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AssociateSoftwareTokenreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
 authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )
```

```
            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
app's MFA.")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte AssociateSoftwareTokena referência da API AWS SDK for Swift.

Use **`ConfirmDevice`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ConfirmDevice`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }
```

- Para obter detalhes da API, consulte ConfirmDevicea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como confirmar um dispositivo de usuário

O exemplo de `confirm-device` a seguir adiciona um novo dispositivo memorizado para o usuário atual.

```
aws cognito-idp confirm-device \
   --access-token eyJra456defEXAMPLE \
   --device-key us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
   --device-secret-verifier-
config PasswordVerifier=TXlWZXJpZmllclN0cmluZw,Salt=TXlTUlBTYWx0
```

Saída:

```
{
     "UserConfirmationNecessary": false
}
```

Consulte mais informações em Trabalhar com dispositivos de usuários no grupo de usuários no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte ConfirmDeviceem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const confirmDevice = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt,
    },
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ConfirmDevicea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
 new
```

```
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
```

```
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- Para obter detalhes da API, consulte a ConfirmDeviceReferência da API AWS SDK for Python (Boto3).

Use **`ConfirmForgotPassword`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ConfirmForgotPassword`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Migre automaticamente usuários conhecidos com uma função do Lambda

CLI

AWS CLI

Para confirmar uma senha esquecida

Este exemplo confirma a inscrição para o nome de usuário diego@example.com.

Comando:

```
aws cognito-idp confirm-forgot-password --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --password PASSWORD --confirmation-code CONF_CODE
```

- Para obter detalhes da API, consulte ConfirmForgotPasswordem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
```

```
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

- Para obter detalhes da API, consulte ConfirmForgotPasswordda Referência AWS SDK para Go da API.

Use **`ConfirmSignUp`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ConfirmSignUp`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
 string userName)
    {
```

```
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- Para obter detalhes da API, consulte ConfirmSignUpa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
```

```
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte ConfirmSignUpa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como confirmar a inscrição

Este exemplo confirma a inscrição para o nome de usuário diego@example.com.

Comando:

```
aws cognito-idp confirm-sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --confirmation-code CONF_CODE
```

- Para obter detalhes da API, consulte ConfirmSignUpem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
           String userName) {
       try {
           ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                   .clientId(clientId)
                   .confirmationCode(code)
                   .username(userName)
                   .build();

           identityProviderClient.confirmSignUp(signUpRequest);
           System.out.println(userName + " was confirmed");

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

- Para obter detalhes da API, consulte ConfirmSignUpa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ConfirmSignUpa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
```

```
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- Para obter detalhes da API, consulte a ConfirmSignUpreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
```

```
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- Para obter detalhes da API, consulte a ConfirmSignUpReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise
 `false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
 String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
```

```
        }
    }
```

- Para obter detalhes da API, consulte ConfirmSignUpa referência da API AWS SDK for Swift.

Use **`CreateUserPool`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateUserPool`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos dos grupos de usuários

CLI

AWS CLI

Como criar um grupo de usuários minimamente configurado

Este exemplo cria um grupo de usuários chamado MyUserPool usando valores padrão. Não há atributos nem clientes da aplicação obrigatórios. A MFA e a segurança avançada estão desabilitadas.

Comando:

```
aws cognito-idp create-user-pool --pool-name MyUserPool
```

Saída:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
```

```
                "AttributeDataType": "String",
                "Mutable": false
            },
            {
                "Name": "name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "given_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "family_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "middle_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
```

```
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "nickname",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "preferred_username",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "profile",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "picture",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
```

```
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "website",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "email",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
```

```
            "Name": "birthdate",
            "StringAttributeConstraints": {
                "MinLength": "10",
                "MaxLength": "10"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "zoneinfo",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "locale",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "phone_number",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
```

```
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "phone_number_verified",
            "Mutable": true
        },
        {
            "Name": "address",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
            "NumberAttributeConstraints": {
                "MinValue": "0"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "Number",
            "Mutable": true
        }
    ],
    "MfaConfiguration": "OFF",
    "Name": "MyUserPool",
    "LastModifiedDate": 1547833345.777,
    "AdminCreateUserConfig": {
        "UnusedAccountValidityDays": 7,
        "AllowAdminCreateUserOnly": false
    },
    "EmailConfiguration": {},
    "Policies": {
        "PasswordPolicy": {
            "RequireLowercase": true,
            "RequireSymbols": true,
            "RequireNumbers": true,
            "MinimumLength": 8,
            "RequireUppercase": true
        }
```

```
        },
        "CreationDate": 1547833345.777,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
        "LambdaConfig": {}
    }
}
```

Como criar um grupo de usuários com dois atributos obrigatórios

Este exemplo cria um grupo de usuários MyUserPool. O grupo é configurado para aceitar o e-mail como o atributo de nome de usuário. Ele também define o endereço de origem do e-mail como um endereço validado usando o Amazon Simple Email Service.

Comando:

```
aws cognito-idp create-user-pool --pool-name MyUserPool --username-
attributes "email" --email-configuration=SourceArn="arn:aws:ses:us-
east-1:111111111111:identity/
jane@example.com",ReplyToEmailAddress="jane@example.com"
```

Saída:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
```

```
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "given_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "family_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "middle_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "nickname",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
```

```
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "preferred_username",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "profile",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "picture",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "website",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
```

```
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "email",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "email_verified",
            "Mutable": true
        },
        {
            "Name": "gender",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "birthdate",
            "StringAttributeConstraints": {
                "MinLength": "10",
                "MaxLength": "10"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
```

```
            },
            {
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "phone_number",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "phone_number_verified",
                "Mutable": true
            },
            {
                "Name": "address",
                "StringAttributeConstraints": {
```

```
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "updated_at",
                "NumberAttributeConstraints": {
                    "MinValue": "0"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "Number",
                "Mutable": true
            }
        ],
        "MfaConfiguration": "OFF",
        "Name": "MyUserPool",
        "LastModifiedDate": 1547837788.189,
        "AdminCreateUserConfig": {
            "UnusedAccountValidityDays": 7,
            "AllowAdminCreateUserOnly": false
        },
        "EmailConfiguration": {
            "ReplyToEmailAddress": "jane@example.com",
            "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/
jane@example.com"
        },
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
        "UsernameAttributes": [
            "email"
        ],
        "CreationDate": 1547837788.189,
```

```
      "EstimatedNumberOfUsers": 0,
      "Id": "us-west-2_aaaaaaaaa",
      "LambdaConfig": {}
  }
}
```

- Para obter detalhes da API, consulte CreateUserPoolem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
 created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
 String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response =
 cognitoClient.createUserPool(request);
            return response.userPool().id();

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

- Para obter detalhes da API, consulte CreateUserPoola Referência AWS SDK for Java 2.x da API.

Use **`CreateUserPoolClient`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateUserPoolClient`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos dos grupos de usuários

CLI

AWS CLI

Para criar um cliente de grupo de usuários

O `create-user-pool-client` exemplo a seguir cria um novo cliente de grupo de usuários com um segredo de cliente, atributos explícitos de leitura e gravação, login com fluxos de nome de usuário, senha e SRP, login com três, acesso a um subconjunto de escopos do OAuth IdPs, análises e uma validade estendida da sessão de autenticação. PinPoint

```
aws cognito-idp create-user-pool-client \
    --user-pool-id us-west-2_EXAMPLE \
    --client-name MyTestClient \
    --generate-secret \
    --refresh-token-validity 10 \
    --access-token-validity 60 \
    --id-token-validity 60 \
    --token-validity-units AccessToken=minutes,IdToken=minutes,RefreshToken=days
 \
    --read-attributes email phone_number email_verified phone_number_verified \
    --write-attributes email phone_number \
    --explicit-auth-
flows ALLOW_USER_PASSWORD_AUTH ALLOW_USER_SRP_AUTH ALLOW_REFRESH_TOKEN_AUTH \
    --supported-identity-providers Google Facebook MyOIDC \
    --callback-urls https://www.amazon.com https://example.com http://
localhost:8001 myapp://example \
```

```
    --allowed-o-auth-flows code implicit \
    --allowed-o-auth-scopes openid profile aws.cognito.signin.user.admin solar-
system-data/asteroids.add \
    --allowed-o-auth-flows-user-pool-client \
    --analytics-configuration ApplicationArn=arn:aws:mobiletargeting:us-
west-2:767671399759:apps/thisisanexamplepinpointapplicationid,UserDataShared=TRUE
 \
    --prevent-user-existence-errors ENABLED \
    --enable-token-revocation \
    --enable-propagate-additional-user-context-data \
    --auth-session-validity 4
```

Saída:

```
{
    "UserPoolClient": {
        "UserPoolId": "us-west-2_EXAMPLE",
        "ClientName": "MyTestClient",
        "ClientId": "123abc456defEXAMPLE",
        "ClientSecret": "this1234is5678my91011example1213client1415secret",
        "LastModifiedDate": 1726788459.464,
        "CreationDate": 1726788459.464,
        "RefreshTokenValidity": 10,
        "AccessTokenValidity": 60,
        "IdTokenValidity": 60,
        "TokenValidityUnits": {
            "AccessToken": "minutes",
            "IdToken": "minutes",
            "RefreshToken": "days"
        },
        "ReadAttributes": [
            "email_verified",
            "phone_number_verified",
            "phone_number",
            "email"
        ],
        "WriteAttributes": [
            "phone_number",
            "email"
        ],
        "ExplicitAuthFlows": [
            "ALLOW_USER_PASSWORD_AUTH",
            "ALLOW_USER_SRP_AUTH",
```

```
            "ALLOW_REFRESH_TOKEN_AUTH"
        ],
        "SupportedIdentityProviders": [
            "Google",
            "MyOIDC",
            "Facebook"
        ],
        "CallbackURLs": [
            "https://example.com",
            "https://www.amazon.com",
            "myapp://example",
            "http://localhost:8001"
        ],
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "AllowedOAuthScopes": [
            "aws.cognito.signin.user.admin",
            "openid",
            "profile",
            "solar-system-data/asteroids.add"
        ],
        "AllowedOAuthFlowsUserPoolClient": true,
        "AnalyticsConfiguration": {
            "ApplicationArn": "arn:aws:mobiletargeting:us-
west-2:123456789012:apps/thisisanexamplepinpointapplicationid",
            "RoleArn": "arn:aws:iam::123456789012:role/aws-service-role/cognito-
idp.amazonaws.com/AWSServiceRoleForAmazonCognitoIdp",
            "UserDataShared": true
        },
        "PreventUserExistenceErrors": "ENABLED",
        "EnableTokenRevocation": true,
        "EnablePropagateAdditionalUserContextData": true,
        "AuthSessionValidity": 4
    }
}
```

Para obter mais informações, consulte Application-specific as configurações com clientes de aplicativos no Guia do Desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte CreateUserPoolClientem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientReques
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRespon

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s
```

```
                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient
 cognitoClient, String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " +
 response.userPoolClient().clientName() + " created. ID: "
                    + response.userPoolClient().clientId());

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

- Para obter detalhes da API, consulte CreateUserPoolClienta Referência AWS SDK for Java 2.x da API.

Use **`DeleteUser`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteUser`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Confirme automaticamente usuários conhecidos com uma função do Lambda
- Migre automaticamente usuários conhecidos com uma função do Lambda
- Grave dados de atividades personalizados com uma função do Lambda após a autenticação do usuário do Amazon Cognito

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir um usuário

Este exemplo exclui um usuário.

Comando:

```
aws cognito-idp delete-user --access-token ACCESS_TOKEN
```

- Para obter detalhes da API, consulte DeleteUserem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Delete the signed-in user. Useful for allowing a user to delete their
```

```
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para JavaScript da API.

Use **`ForgotPassword`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ForgotPassword`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Migre automaticamente usuários conhecidos com uma função do Lambda

CLI

AWS CLI

Para forçar uma alteração de senha

O `forgot-password` exemplo a seguir envia uma mensagem para jane@example.com para alterar a senha.

```
aws cognito-idp forgot-password --client-id 38fjsnc484p94kpqsnet7mpld0 --
username jane@example.com
```

Saída:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

- Para obter detalhes da API, consulte ForgotPasswordem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

- Para obter detalhes da API, consulte ForgotPasswordda Referência AWS SDK para Go da API.

Use **`InitiateAuth`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `InitiateAuth`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Confirme automaticamente usuários conhecidos com uma função do Lambda
- Migre automaticamente usuários conhecidos com uma função do Lambda
- Conceitos básicos dos grupos de usuários
- Inscrever um usuário em um grupo de usuários que exija MFA
- Grave dados de atividades personalizados com uma função do Lambda após a autenticação do usuário do Amazon Cognito

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- Para obter detalhes da API, consulte InitiateAuth a Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como conectar um usuário

O exemplo de `initiate-auth` a seguir conecta um usuário com o fluxo básico de nome de usuário e senha e sem desafios adicionais.

```
aws cognito-idp initiate-auth \
    --auth-flow USER_PASSWORD_AUTH \
    --client-id 1example23456789 \
    --analytics-metadata AnalyticsEndpointId=d70b2ba36a8c4dc5a04a0451aEXAMPLE \
    --auth-parameters USERNAME=testuser,PASSWORD=[Password] --user-context-
data EncodedData=mycontextdata --client-metadata MyTestKey=MyTestValue
```

Saída:

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-v7w9UcY6"
        }
    }
}
```

Consulte mais informações em Authentication no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte InitiateAuthem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
```

```
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

- Para obter detalhes da API, consulte InitiateAuth a Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const initiateAuth = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte InitiateAuth a Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo mostra como iniciar a autenticação com um dispositivo rastreado. Para concluir o login, o cliente deve responder corretamente aos desafios de Secure Remote Password (SRP).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
```

```
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
```

```
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- Para obter detalhes da API, consulte a InitiateAuthReferência da API AWS SDK for Python (Boto3).

Use **`ListUserPools`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListUserPools`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- Para obter detalhes da API, consulte ListUserPoolsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar grupos de usuários

O `list-user-pools` exemplo a seguir lista três dos grupos de usuários disponíveis na AWS conta das credenciais atuais da CLI.

```
aws cognito-idp list-user-pools \
    --max-results 3
```

Saída:

```
{
    "NextToken": "[Pagination token]",
    "UserPools": [
        {
            "CreationDate": 1681502497.741,
            "Id": "us-west-2_EXAMPLE1",
            "LambdaConfig": {
                "CustomMessage": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreSignUp": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreTokenGeneration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreTokenGenerationConfig": {
                    "LambdaArn": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                    "LambdaVersion": "V1_0"
                }
            },
            "LastModifiedDate": 1681502497.741,
            "Name": "user pool 1"
        },
        {
            "CreationDate": 1686064178.717,
            "Id": "us-west-2_EXAMPLE2",
```

```
            "LambdaConfig": {
            },
            "LastModifiedDate": 1686064178.873,
            "Name": "user pool 2"
        },
        {
            "CreationDate": 1627681712.237,
            "Id": "us-west-2_EXAMPLE3",
            "LambdaConfig": {
                "UserMigration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction"
            },
            "LastModifiedDate": 1678486942.479,
            "Name": "user pool 3"
        }
    ]
}
```

Para obter mais informações, consulte Grupos de usuários do Amazon Cognito no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte ListUserPoolsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- Para obter detalhes da API, consulte ListUserPoolsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }
```

```
    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

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

- Para obter detalhes da API, consulte ListUserPoolsa Referência AWS SDK for Java 2.x da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
```

```
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListUserPoolsreferência da API AWS SDK for Rust.

Use **`ListUsers`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListUsers`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos dos grupos de usuários
- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: como listar usuários com um filtro do lado do servidor

O exemplo de `list-users` a seguir lista três usuários no grupo de usuários solicitado cujos endereços de e-mail começam com `testuser`.

```
aws cognito-idp list-users \
    --user-pool-id us-west-2_EXAMPLE \
    --filter email^=\"testuser\" \
    --max-items 3
```

Saída:

```
{
    "PaginationToken": "efgh5678EXAMPLE",
    "Users": [
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "eaad0219-2117-439f-8d46-4db20e59268f"
                },
                {
                    "Name": "email",
                    "Value": "testuser@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1682955829.578,
            "UserLastModifiedDate": 1689030181.63,
            "UserStatus": "CONFIRMED",
            "Username": "testuser"
        },
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "3b994cfd-0b07-4581-be46-3c82f9a70c90"
                },
                {
                    "Name": "email",
                    "Value": "testuser2@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1684427979.201,
            "UserLastModifiedDate": 1684427979.201,
            "UserStatus": "UNCONFIRMED",
            "Username": "testuser2"
        },
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "5929e0d1-4c34-42d1-9b79-a5ecacfe66f7"
                },
```

```
                {
                    "Name": "email",
                    "Value": "testuser3@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1684427823.641,
            "UserLastModifiedDate": 1684427823.641,
            "UserStatus": "UNCONFIRMED",
            "Username": "testuser3@example.com"
        }
    ]
}
```

Consulte mais informações em Managing and searching for users no Guia do desenvolvedor do Amazon Cognito.

Exemplo 2: como listar usuários com um filtro do lado do cliente

O exemplo de `list-users` a seguir lista os atributos de três usuários que têm um atributo, nesse caso, o endereço de e-mail, que contém o domínio de e-mail “@example.com”. Se outros atributos contivessem essa string, eles também seriam exibidos. O segundo usuário não tem atributos que correspondam à consulta e é excluído da saída exibida, mas não da resposta do servidor.

```
aws cognito-idp list-users \
    --user-pool-id us-west-2_EXAMPLE \
    --max-items 3
    --query Users\[\*\].Attributes\[\?Value\.contains\(\@\,\'@example.com\'\)\]
```

Saída:

```
[
    [
        {
            "Name": "email",
            "Value": "admin@example.com"
        }
    ],
    [],
    [
        {
```

```
            "Name": "email",
            "Value": "operator@example.com"
        }
    ]
]
```

Consulte mais informações em Managing and searching for users no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte ListUsersem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUsers {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's
created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                        + user.userCreateDate());
            });

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

```
        }
    }

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient
 cognitoClient, String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username()
 + " Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

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

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const listUsers = ({ userPoolId }) => {
```

```
  const client = new CognitoIdentityProviderClient({});

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListUsersreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
 self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
```

```
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

- Para obter detalhes da API, consulte a ListUsersReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_cgp->listusers(
      iv_userpoolid = iv_user_pool_id
    ).

    ot_users = lo_result->get_users( ).

    MESSAGE |Found { lines( ot_users ) } users in the pool.| TYPE 'I'.

  CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE |User pool { iv_user_pool_id } not found.| TYPE 'E'.

  CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
    MESSAGE 'Not authorized to list users.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListUsersreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        do {
            let output = try await cognitoClient.listUsers(
                input: ListUsersInput(
                    userPoolId: poolId
                )
            )

            guard let users = output.users else {
                print("No users found.")
                return
            }

            print("\(users.count) user(s) found.")
            for user in users {
                print("  \(user.username ?? "<unknown>")")
            }
        } catch _ as NotAuthorizedException {
            print("*** Please authenticate with AWS before using this command.")
            return
        } catch _ as ResourceNotFoundException {
            print("*** The specified User Pool was not found.")
            return
        } catch {
            print("*** An unexpected type of error occurred.")
            return
        }
```

- Para obter detalhes da API, consulte ListUsersa referência da API AWS SDK for Swift.

Use **`ResendConfirmationCode`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ResendConfirmationCode`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);
```

```
        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- Para obter detalhes da API, consulte ResendConfirmationCodea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
```

```
                    << std::endl;
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte ResendConfirmationCodea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como reenviar um código de confirmação

O exemplo `resend-confirmation-code` a seguir envia um código de confirmação ao usuário `jane`.

```
aws cognito-idp resend-confirmation-code \
    --client-id 12a3b456c7de890f11g123hijk \
    --username jane
```

Saída:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

Para obter mais informações, consulte Como cadastrar e confirmar contas de usuários no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte ResendConfirmationCodeem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
           String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                   .clientId(clientId)
                   .username(userName)
                   .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

- Para obter detalhes da API, consulte ResendConfirmationCodea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ResendConfirmationCodea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
```

```
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- Para obter detalhes da API, consulte a ResendConfirmationCodereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
```

```
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- Para obter detalhes da API, consulte a ResendConfirmationCodeReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
 clientId: String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
 else {
                print("*** Unable to get the delivery method for the resent
 code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
```

```
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user
 \(userName).")
            return false
        }
    }
```

- Para obter detalhes da API, consulte ResendConfirmationCodea referência da API AWS SDK for Swift.

Use **`RespondToAuthChallenge`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RespondToAuthChallenge`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

CLI

AWS CLI

Exemplo 1: como responder ao desafio NEW_PASSWORD_REQUIRED

O exemplo de `respond-to-auth-challenge` a seguir responde a um desafio NEW_PASSWORD_REQUIRED que initiate-auth retornou. Ele define uma senha para o usuário `jane@example.com`.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789 \
    --challenge-name NEW_PASSWORD_REQUIRED \
    --challenge-responses USERNAME=jane@example.com,NEW_PASSWORD=[Password] \
    --session AYABeEv5HklEXAMPLE
```

Saída:

```
{
    "ChallengeParameters": {},
```

```
    "AuthenticationResult": {
        "AccessToken": "ACCESS_TOKEN",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "REFRESH_TOKEN",
        "IdToken": "ID_TOKEN",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-wt2ha1Zd"
        }
    }
}
```

Consulte mais informações em Authentication no Guia do desenvolvedor do Amazon Cognito.

Exemplo 2: como responder a um desafio SELECT_MFA_TYPE

O exemplo de `respond-to-auth-challenge` a seguir escolhe a MFA TOTP como a opção de MFA para o usuário atual. O usuário foi solicitado a selecionar um tipo de MFA e, depois, será solicitado a inserir o código da MFA.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789
    --session AYABeEv5HklEXAMPLE
    --challenge-name SELECT_MFA_TYPE
    --challenge-responses USERNAME=testuser,ANSWER=SOFTWARE_TOKEN_MFA
```

Saída:

```
{
    "ChallengeName": "SOFTWARE_TOKEN_MFA",
    "Session": "AYABeEv5HklEXAMPLE",
    "ChallengeParameters": {
        "FRIENDLY_DEVICE_NAME": "transparent"
    }
}
```

Consulte mais informações em Adding MFA no Guia do desenvolvedor do Amazon Cognito.

Exemplo 3: como responder a um desafio SOFTWARE_TOKEN_MFA

O exemplo de `respond-to-auth-challenge` a seguir fornece um código de MFA TOTP e conclui o login.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789 \
    --session AYABeEv5HklEXAMPLE \
    --challenge-name SOFTWARE_TOKEN_MFA \
    --challenge-responses USERNAME=testuser,SOFTWARE_TOKEN_MFA_CODE=123456
```

Saída:

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-v7w9UcY6"
        }
    }
}
```

Consulte mais informações em Adding MFA no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte RespondToAuthChallengeem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
```

```
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte RespondToAuthChallengea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça login com um dispositivo rastreado. Para concluir o login, o cliente deve responder corretamente aos desafios de Secure Remote Password (SRP).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
```

```
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
```

```
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
```

```
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- Para obter detalhes da API, consulte a RespondToAuthChallengeReferência da API AWS SDK for Python (Boto3).

Use **`SignUp`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SignUp`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Confirme automaticamente usuários conhecidos com uma função do Lambda
- Migre automaticamente usuários conhecidos com uma função do Lambda
- Conceitos básicos dos grupos de usuários
- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- Para obter detalhes da API, consulte SignUpa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was
successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
```

```
                    << "The username already exists. Please enter a different
 username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte SignUpa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como inscrever um usuário

Este exemplo inscreve jane@example.com.

Comando:

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
 Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

Saída:

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- Para obter detalhes da API, consulte SignUpem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
```

```
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

- Para obter detalhes da API, consulte SignUpa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void signUp(CognitoIdentityProviderClient
 identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();
```

```
            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

- Para obter detalhes da API, consulte SignUpa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte SignUpa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Para obter detalhes da API, consulte a SignUpreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
 Cognito
        to send an email to the specified email address. The email contains a
 code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
 whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
```

```
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- Para obter detalhes da API, consulte a SignUpReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
 userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )
```

```
            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }
```

- Para obter detalhes da API, consulte SignUpa referência da API AWS SDK for Swift.

Use **`UpdateUserPool`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateUserPool`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Confirme automaticamente usuários conhecidos com uma função do Lambda
- Migre automaticamente usuários conhecidos com uma função do Lambda
- Grave dados de atividades personalizados com uma função do Lambda após a autenticação do usuário do Amazon Cognito

CLI

AWS CLI

Para atualizar um grupo de usuários

O `update-user-pool` exemplo a seguir modifica um grupo de usuários com um exemplo de sintaxe para cada uma das opções de configuração disponíveis. Para atualizar um grupo de usuários, você deve especificar todas as opções configuradas anteriormente ou elas serão redefinidas para um valor padrão.

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_EXAMPLE \
    --policies PasswordPolicy=
\{MinimumLength=6,RequireUppercase=true,RequireLowercase=true,RequireNumbers=true,Require
 \
    --deletion-protection ACTIVE \
    --lambda-config PreSignUp="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-presignup-
function",PreTokenGeneration="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-pretoken-function" \
    --auto-verified-attributes "phone_number" "email" \
    --verification-message-template \{\"SmsMessage\":\""Your code is
 {####}"\",\"EmailMessage\":\""Your code is {####}"\",\"EmailSubject\":\""Your
 verification code"\",\"EmailMessageByLink\":\""Click {##here##} to verify
 your email address."\",\"EmailSubjectByLink\":\""Your verification link"\",
\"DefaultEmailOption\":\"CONFIRM_WITH_LINK\"\} \
    --sms-authentication-message "Your code is {####}" \
    --user-attribute-update-settings
 AttributesRequireVerificationBeforeUpdate="email","phone_number" \
    --mfa-configuration "OPTIONAL" \
    --device-
configuration ChallengeRequiredOnNewDevice=true,DeviceOnlyRememberedOnUserPrompt=true
 \
    --email-configuration SourceArn="arn:aws:ses:us-
west-2:123456789012:identity/admin@example.com",ReplyToEmailAddress="amdin
+noreply@example.com",EmailSendingAccount=DEVELOPER,From="admin@amazon.com",Configuration
configuration-set" \
    --sms-configuration SnsCallerArn="arn:aws:iam::123456789012:role/service-
role/SNS-SMS-Role",ExternalId="12345",SnsRegion="us-west-2" \
    --admin-create-user-config
 AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailMessage=\""Welcome
```

```
 {username}. Your confirmation code is {####}"\",EmailSubject=\""Welcome to
 MyMobileGame"\"\} \
    --user-pool-tags "Function"="MyMobileGame","Developers"="Berlin" \
    --admin-create-user-config
 AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailSubject=\""Welcome to
 MyMobileGame"\"\} \
    --user-pool-add-ons AdvancedSecurityMode="AUDIT" \
    --account-recovery-setting RecoveryMechanisms=
\[\{Priority=1,Name="verified_email"\},
\{Priority=2,Name="verified_phone_number"\}\]
```

Este comando não produz saída.

Para obter mais informações, consulte Configurar um cliente de aplicação de grupo de usuários no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte UpdateUserPoolem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)
```

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
```

```
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte UpdateUserPoola Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
```

```
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Para obter detalhes da API, consulte UpdateUserPoola Referência AWS SDK para JavaScript da API.

Use **`VerifySoftwareToken`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `VerifySoftwareToken`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inscrever um usuário em um grupo de usuários que exija MFA

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- Para obter detalhes da API, consulte VerifySoftwareTokena Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte VerifySoftwareTokena Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como confirmar o registro de um autenticador TOTP

O exemplo de `verify-software-token` a seguir completa o registro TOTP para o usuário atual.

```
aws cognito-idp verify-software-token \
    --access-token eyJra456defEXAMPLE \
    --user-code 123456
```

Saída:

```
{
```

```
    "Status": "SUCCESS"
}
```

Para obter mais informações, consulte Adicionar MFA a um grupo de usuários no Guia do desenvolvedor do Amazon Cognito.

- Para obter detalhes da API, consulte VerifySoftwareTokenem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

- Para obter detalhes da API, consulte VerifySoftwareTokena Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte VerifySoftwareTokena Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- Para obter detalhes da API, consulte a VerifySoftwareTokenreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
```

```
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Para obter detalhes da API, consulte a VerifySoftwareTokenReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_cgp->verifysoftwaretoken(
          iv_session = iv_session
          iv_usercode = iv_user_code
        ).

        ov_status = lo_result->get_status( ).

        IF ov_status = 'SUCCESS'.
          MESSAGE 'MFA token verified successfully.' TYPE 'I'.
        ELSE.
          MESSAGE |MFA verification status: { ov_status }.| TYPE 'I'.
        ENDIF.
```

```
      CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
        MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

      CATCH /aws1/cx_cgpenbsoftwaretokmf00 INTO DATA(lo_enabled_ex).
        MESSAGE 'Software token MFA is already enabled.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a VerifySoftwareTokenreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
 mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
```

```
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }
```

- Para obter detalhes da API, consulte VerifySoftwareTokena referência da API AWS SDK for Swift.

Cenários para o Amazon Cognito Identity Provider usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Cognito Identity Provider com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon Cognito Identity Provider ou em combinação com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Confirme automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK
- Migre automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK
- Conceitos básicos dos grupos de usuários
- Inscrever um usuário com um grupo de usuários do Amazon Cognito que exige MFA usando um AWS SDK
- Usar bancos de identidades e fluxos de identidades do Amazon Cognito
- Grave dados de atividades personalizados com uma função Lambda após a autenticação do usuário do Amazon Cognito usando um AWS SDK

Confirme automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK

Os exemplos de código a seguir mostram como confirmar automaticamente usuários conhecidos do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PreSignUp`.
- Inscreva-se para ser um usuário no Amazon Cognito.
- A função do Lambda verifica uma tabela do DynamoDB e confirma automaticamente os usuários conhecidos.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
```

```
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
```

```
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string,
 userName string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")
```

```
 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `PreSignUp` com uma função do Lambda.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
```

```
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
```

```
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
```

```
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
```

```
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
```

```
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
```

```
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
```

```
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
```

```
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
```

```
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
```

```
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
```

```
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
```

```
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}
```

```
// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}
```

```
// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}
```

```
func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
```

```
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure uma execução interativa de “Cenário”. Os exemplos JavaScript (v3) compartilham um executor de cenários para simplificar exemplos complexos. O código-fonte completo está ativado GitHub.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
```

```
      UserEmail: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

Esse cenário demonstra a confirmação automática de um usuário conhecido. Ele orquestra as etapas do exemplo.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
```

```
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */
```

```
const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);
```

```
const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
```

```
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
```

```
    const signUp = (password) =>
      signUpUser({
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
```

```
    if (logStreamErr) {
      state.errors.push(logStreamErr);
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }
```

```
    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with:
 ${state.token.slice(0, 11)}`,
  { skipWhen: skipWhenErrors },
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
```

```
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

Essas são etapas compartilhadas com outros cenários.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);
```

```
export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
  { skipWhen: skipWhenErrors },
);
```

Um manipulador do gatilho `PreSignUp` com uma função do Lambda.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user
 '${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
```

```
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
 '${storedUserInfo.UserName}' does not match supplied UserName
 '${event.userName}'. Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

Módulo de ações de CloudWatch registros.

```
import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
```

```
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[]
 | null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Módulo de ações do Amazon Cognito.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
```

```
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
```

```
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password:
 string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
```

```
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

Módulo de ações do DynamoDB.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string,
 unknown>[] }} config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteUser
  - InitiateAuth

- SignUp
- UpdateUserPool

Migre automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK

O exemplo de código a seguir mostra como migrar automaticamente usuários conhecidas do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `MigrateUser`.
- Faça login no Amazon Cognito com um nome de usuário e e-mail que não estejam no grupo de usuários.
- A função do Lambda verifica uma tabela do DynamoDB e migra automaticamente os usuários conhecidos para o grupo de usuários.
- Realize um fluxo de senha esquecida para redefinir a senha para o usuário migrado.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
```

```
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
```

```
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
```

```
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string,
 user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
  log.Println("To complete this example and successfully migrate a user to
 Cognito, you must enter an email\n" +
   "you own that can receive a confirmation code.")
  return
 }
 codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
 user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("\nA confirmation code has been sent to %v.",
 *codeDelivery.Destination)
 code := runner.questioner.Ask("Check your email and enter it here:")

 confirmed := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
```

```
  "(the password will not display as you type):", 8)
 for !confirmed {
  log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
 user.UserName, password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   confirmed = true
  }
 }
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
 password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))
```

```
 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `MigrateUser` com uma função do Lambda.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)
```

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n",
 user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
 if err != nil {
```

```
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
 map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
```

```
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
```

```
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
```

```
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
```

```
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
```

```
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
```

```
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
```

```
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
```

```
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
```

```
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
```

```
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}
```

```
// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}
```

```
// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}
```

```
func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
```

```
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Conceitos básicos dos grupos de usuários

O exemplo de código a seguir mostra como:

- Criar um grupo de usuários.
- Criar um cliente de aplicação.
- Configurar um domínio para o grupo de usuários.
- Criar um usuário como administrador.
- Habilitar o autorregistro.
- Listar usuários no grupo de usuários.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Cognito User Pools Getting Started Script
# This script creates and configures an Amazon Cognito user pool with an app
 client

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cognito-user-pool-setup.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Cognito User Pool setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -qi "error\|failed"; then
    echo "ERROR: Command failed: $cmd" >&2
    echo "Output: $output" >&2
    cleanup_on_error
    exit 1
  fi
}
```

```
# Function to check AWS CLI return code
check_aws_error() {
  local exit_code=$?
  local cmd=$1

  if [ $exit_code -ne 0 ]; then
    echo "ERROR: AWS CLI command failed with exit code $exit_code: $cmd" >&2
    cleanup_on_error
    exit "$exit_code"
  fi
}

# Function to clean up resources on error
cleanup_on_error() {
  echo "Error encountered. Attempting to clean up resources..." >&2

  if [ -n "${DOMAIN_NAME:-}" ] && [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool domain: $DOMAIN_NAME" >&2
    aws cognito-idp delete-user-pool-domain \
      --user-pool-id "$USER_POOL_ID" \
      --domain "$DOMAIN_NAME" 2>/dev/null || true
  fi

  if [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool: $USER_POOL_ID" >&2
    aws cognito-idp delete-user-pool \
      --user-pool-id "$USER_POOL_ID" 2>/dev/null || true
  fi
}

# Set trap for cleanup on exit
trap cleanup_on_error EXIT ERR

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
  echo "ERROR: AWS CLI is not installed" >&2
  exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
  echo "ERROR: jq is not installed" >&2
  exit 1
```

```
fi

# Validate openssl is installed
if ! command -v openssl &> /dev/null; then
  echo "ERROR: openssl is not installed" >&2
  exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
  echo "ERROR: AWS CLI is not configured or credentials are invalid" >&2
  exit 1
fi

# Get the current AWS region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
  AWS_REGION="us-east-1" # Default region if not configured
fi
echo "Using AWS Region: $AWS_REGION"

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
  echo "ERROR: Invalid AWS region format: $AWS_REGION" >&2
  exit 1
fi

# Generate random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
  echo "ERROR: Failed to generate random identifier" >&2
  exit 1
fi

USER_POOL_NAME="MyUserPool-${RANDOM_ID}"
APP_CLIENT_NAME="MyAppClient-${RANDOM_ID}"
DOMAIN_NAME="my-auth-domain-${RANDOM_ID}"

# Validate resource names don't exceed limits
if [ ${#USER_POOL_NAME} -gt 128 ]; then
  echo "ERROR: User pool name exceeds maximum length of 128 characters" >&2
  exit 1
fi

if [ ${#APP_CLIENT_NAME} -gt 128 ]; then
```

```
  echo "ERROR: App client name exceeds maximum length of 128 characters" >&2
  exit 1
fi

echo "Using random identifier: $RANDOM_ID"
echo "User pool name: $USER_POOL_NAME"
echo "App client name: $APP_CLIENT_NAME"
echo "Domain name: $DOMAIN_NAME"

# Step 1: Create a User Pool
echo "Creating user pool..."
USER_POOL_OUTPUT=$(aws cognito-idp create-user-pool \
  --pool-name "$USER_POOL_NAME" \
  --auto-verified-attributes email \
  --username-attributes email \
  --policies '{"PasswordPolicy":
{"MinimumLength":12,"RequireUppercase":true,"RequireLowercase":true,"RequireNumbers":true
 \
  --schema '[{"Name":"email","Required":true,"Mutable":true}]' \
  --mfa-configuration OFF \
  --user-attribute-update-settings '{"AttributesRequireVerificationBeforeUpdate":
["email"]}' \
  --account-recovery-setting
 'RecoveryMechanisms=[{Name=verified_email,Priority=1}]' \
  --deletion-protection INACTIVE \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool"

# Extract the User Pool ID using jq for safety
USER_POOL_ID=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Id // empty')
if [ -z "$USER_POOL_ID" ]; then
  echo "ERROR: Failed to extract User Pool ID" >&2
  exit 1
fi

# Validate User Pool ID format
if ! [[ "$USER_POOL_ID" =~ ^[a-z]{2}-[a-z]+-[0-9]+_[a-zA-Z0-9]+$ ]]; then
  echo "ERROR: Invalid User Pool ID format: $USER_POOL_ID" >&2
  exit 1
fi

echo "User Pool created with ID: $USER_POOL_ID"
```

```
USER_POOL_ARN=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Arn // empty')
aws cognito-idp tag-resource \
  --resource-arn "$USER_POOL_ARN" \
  --tags project=doc-smith,tutorial=amazon-cognito-gs

# Wait for user pool to be ready
echo "Waiting for user pool to be ready..."
sleep 5

# Step 2: Create an App Client with enhanced security
echo "Creating app client..."
APP_CLIENT_OUTPUT=$(aws cognito-idp create-user-pool-client \
  --user-pool-id "$USER_POOL_ID" \
  --client-name "$APP_CLIENT_NAME" \
  --no-generate-secret \
  --explicit-auth-flows ALLOW_REFRESH_TOKEN_AUTH ALLOW_USER_PASSWORD_AUTH \
  --callback-urls '["https://localhost:3000/callback"]' \
  --allowed-o-auth-flows 'code' \
  --allowed-o-auth-scopes 'openid' 'email' 'profile' \
  --allowed-o-auth-flows-user-pool-client \
  --prevent-user-existence-errors ENABLED \
  --enable-token-revocation \
  --access-token-validity 1 \
  --id-token-validity 1 \
  --refresh-token-validity 30 \
  --token-validity-units 'AccessToken=hours,IdToken=hours,RefreshToken=days' \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-client"

# Extract the Client ID using jq for safety
CLIENT_ID=$(echo "$APP_CLIENT_OUTPUT" | jq -r '.UserPoolClient.ClientId //
 empty')
if [ -z "$CLIENT_ID" ]; then
  echo "ERROR: Failed to extract Client ID" >&2
  cleanup_on_error
  exit 1
fi

# Validate Client ID format
if ! [[ "$CLIENT_ID" =~ ^[a-z0-9]{26}$ ]]; then
  echo "ERROR: Invalid Client ID format: $CLIENT_ID" >&2
  cleanup_on_error
  exit 1
```

```
fi

echo "App Client created with ID: $CLIENT_ID"

# Step 3: Set Up a Domain for Your User Pool
echo "Setting up user pool domain..."
DOMAIN_OUTPUT=$(aws cognito-idp create-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-domain"
echo "Domain created: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"

# Step 4: View User Pool Details
echo "Retrieving user pool details..."
USER_POOL_DETAILS=$(aws cognito-idp describe-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool"
echo "User Pool details retrieved successfully"

# Step 5: View App Client Details
echo "Retrieving app client details..."
APP_CLIENT_DETAILS=$(aws cognito-idp describe-user-pool-client \
  --user-pool-id "$USER_POOL_ID" \
  --client-id "$CLIENT_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool-client"
echo "App Client details retrieved successfully"

# Step 6: Create a User (Admin)
echo "Creating admin user..."
ADMIN_USER_EMAIL="admin@example.com"
TEMP_PASSWORD="$(openssl rand -base64 12 | tr -d '\n')!@#"
if [ -z "$TEMP_PASSWORD" ]; then
  echo "ERROR: Failed to generate temporary password" >&2
  cleanup_on_error
  exit 1
fi

ADMIN_USER_OUTPUT=$(aws cognito-idp admin-create-user \
```

```
  --user-pool-id "$USER_POOL_ID" \
  --username "$ADMIN_USER_EMAIL" \
  --user-attributes Name=email,Value="$ADMIN_USER_EMAIL"
 Name=email_verified,Value=true \
  --temporary-password "$TEMP_PASSWORD" \
  --message-action SUPPRESS \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-create-user"
echo "Admin user created: $ADMIN_USER_EMAIL"

# Securely clear temporary password from memory
unset TEMP_PASSWORD

# Step 7: Self-Registration
echo "Demonstrating self-registration..."
USER_EMAIL="user@example.com"
USER_PASSWORD="SecurePassword123!"
SIGNUP_OUTPUT=$(aws cognito-idp sign-up \
  --client-id "$CLIENT_ID" \
  --username "$USER_EMAIL" \
  --password "$USER_PASSWORD" \
  --user-attributes Name=email,Value="$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "sign-up"
echo "User signed up: $USER_EMAIL"
echo "A confirmation code would be sent to the user's email in a real scenario"

# Securely clear password from memory
unset USER_PASSWORD

echo ""
echo "====================================================="
echo "IMPORTANT: In a real scenario, the user would receive"
echo "a confirmation code via email. For this demo, we'll"
echo "use admin-confirm-sign-up instead."
echo "====================================================="
echo ""

# Step 8: Confirm User Registration (using admin privileges for demo)
echo "Confirming user registration (admin method)..."
CONFIRM_OUTPUT=$(aws cognito-idp admin-confirm-sign-up \
  --user-pool-id "$USER_POOL_ID" \
```

```
  --username "$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-confirm-sign-up"
echo "User confirmed: $USER_EMAIL"

# Step 9: Set permanent password for user
echo "Setting permanent password for user..."
SET_PASSWORD="SecureUserPassword123!"
SET_PASS_OUTPUT=$(aws cognito-idp admin-set-user-password \
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --password "$SET_PASSWORD" \
  --permanent \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-set-user-password"
echo "Permanent password set for user"

unset SET_PASSWORD

# Step 10: Authenticate a User
echo "Authenticating user..."
AUTH_PASSWORD="SecureUserPassword123!"
AUTH_OUTPUT=$(aws cognito-idp initiate-auth \
  --client-id "$CLIENT_ID" \
  --auth-flow USER_PASSWORD_AUTH \
  --auth-parameters "USERNAME=$USER_EMAIL,PASSWORD=$AUTH_PASSWORD" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "initiate-auth"
echo "User authenticated successfully"

unset AUTH_PASSWORD

# Extract auth tokens securely
ID_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.IdToken // empty')
ACCESS_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.AccessToken //
 empty')

# Validate tokens exist
if [ -z "$ID_TOKEN" ] || [ -z "$ACCESS_TOKEN" ]; then
  echo "WARNING: Failed to extract authentication tokens" >&2
else
```

```
  echo "Authentication tokens obtained successfully"
fi

# Securely clear tokens from memory
unset ID_TOKEN
unset ACCESS_TOKEN

# Step 11: List Users in the User Pool
echo "Listing users in the user pool..."
USERS_OUTPUT=$(aws cognito-idp list-users \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "list-users"
echo "Users listed successfully"

# Display summary of created resources
echo ""
echo "====================================================="
echo "RESOURCE SUMMARY"
echo "====================================================="
echo "User Pool ID: $USER_POOL_ID"
echo "User Pool Name: $USER_POOL_NAME"
echo "App Client ID: $CLIENT_ID"
echo "App Client Name: $APP_CLIENT_NAME"
echo "Domain: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"
echo "Admin User: $ADMIN_USER_EMAIL"
echo "Regular User: $USER_EMAIL"
echo "====================================================="
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting cleanup process..."

# Step 12: Clean Up Resources
echo "Deleting user pool domain..."
DELETE_DOMAIN_OUTPUT=$(aws cognito-idp delete-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
```

```
  2>&1)
check_aws_error "delete-user-pool-domain"
echo "Domain deleted successfully"

# Wait for domain deletion to complete
echo "Waiting for domain deletion to complete..."
sleep 5

echo "Deleting user pool (this will also delete the app client)..."
DELETE_POOL_OUTPUT=$(aws cognito-idp delete-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "delete-user-pool"
echo "User pool deleted successfully"

echo "All resources have been cleaned up"

echo "Script completed at $(date)"

# Remove trap to prevent cleanup on successful exit
trap - EXIT ERR
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AdminConfirmSignUp
  - AdminCreateUser
  - CreateUserPool
  - CreateUserPoolClient
  - CreateUserPoolDomain
  - DeleteUserPool
  - DeleteUserPoolDomain
  - DescribeUserPool
  - DescribeUserPoolClient
  - InitiateAuth
  - ListUsers
  - SignUp

Inscrever um usuário com um grupo de usuários do Amazon Cognito que exige MFA usando um AWS SDK

Os exemplos de código a seguir mostram como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
            )
            .Build();
```

```
        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK
script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
```

```
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address:
{email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();
```

```
        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user
 pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        Console.WriteLine($"Setting up authenticator for {userName} in the user
 pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google
 Authenticator: ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}


using System.Net;
```

```
namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
```

```
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
```

```
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
```

```
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AassociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }


    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }

    /// <summary>
```

```
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
 string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
```

```
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }


    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
```

```
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
_cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }


    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
```

```
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice

- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
            << "Welcome to the Amazon Cognito example scenario."
            << std::endl;
    printAsterisksLine();
```

```
   std::cout
           << "This scenario will add a user to an Amazon Cognito user pool."
           << std::endl;
   const Aws::String userName = askQuestion("Enter a new username: ");
   const Aws::String password = askQuestion("Enter a new password: ");
   const Aws::String email = askQuestion("Enter a valid email for the user: ");

   std::cout << "Signing up " << userName << std::endl;

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
   bool userExists = false;
   do {
       // 1. Add a user with a username, password, and email address.
       Aws::CognitoIdentityProvider::Model::SignUpRequest request;
       request.AddUserAttributes(
               Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                       "email").WithValue(email));
       request.SetUsername(userName);
       request.SetPassword(password);
       request.SetClientId(clientID);
       Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
               client.SignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "The signup request for " << userName << " was
successful."
                     << std::endl;
       }
       else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
           std::cout
                   << "The username already exists. Please enter a different
username."
                   << std::endl;
           userExists = true;
       }
       else {
           std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
```

```
        }
    } while (userExists);

    printAsterisksLine();
    std::cout << "Retrieving status of " << userName << " in the user pool."
              << std::endl;
    // 2. Confirm that the user was added to the user pool.
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    std::cout << "A confirmation code was sent to " << email << "." << std::endl;

    bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
    if (resend) {
        // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();

    {
```

```
        // 4. Send the confirmation code that's received in the email.
(ConfirmSignUp)
        const Aws::String confirmationCode = askQuestion(
                "Enter the confirmation code that was emailed: ");
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
              << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout << "Initiating authorization using the username and password."
              << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
session, client)) {
        return false;
    }

    printAsterisksLine();
```

```
    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-
factor authentication (MFA)."
            << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
    askQuestion("Type enter to continue...", alwaysTrueTest);
```

```
    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");

        //  7. Send the MFA code copied from an authenticator app.
(VerifySoftwareToken)
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
        Aws::String mfaCode = askQuestion(
```

```
                "Re-enter the 6 digit code displayed in the authenticator app:
");

       // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
       request.AddChallengeResponses("USERNAME", userName);
       request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
       request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
       request.SetClientId(clientID);
       request.SetUserPoolId(userPoolID);
       request.SetSession(session);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
               client.AdminRespondToAuthChallenge(request);

       if (outcome.IsSuccess()) {
           std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                     << std::endl;

           accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
       }
       else {
           std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }

       std::cout << "You have successfully added a user to Amazon Cognito."
                 << std::endl;
   }

   if (askYesNoQuestion("Would you like to delete the user that you just added?
(y/n) ")) {
       // 10. Delete the user that you just added. (DeleteUser)
```

```
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
```

```
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
 std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

 Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
```

```
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRespons
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
```

```
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

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

```
                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the
AWS CDK script.
                    poolId - The pool Id that you can get from the AWS CDK
script.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);
```

```
        System.out
                .println("*** Conformation code sent to " + userName + ". Would
you like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient,
mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
```

```
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId)
{
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
```

```
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " +
response.challengeName());
            return response;

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

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
```

```
            System.out.println(userName + " was confirmed");

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

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
```

```
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp

- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para obter a melhor experiência, clone o GitHub repositório e execute este exemplo. O código a seguir representa uma amostra da aplicação de exemplo completa.

```
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-
up' command.`,
```

```
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Signing up.");
    await signUp({ clientId, username, password, email });
    logger.log(`Signed up. A confirmation email has been sent to: ${email}.`);
    logger.log(
      `Run 'confirm-sign-up ${username} <code>' to confirm your account.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
```

```
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Confirming user.");
    await confirmSignUp({ clientId, username, code });
    logger.log(
```

```
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { confirmSignUpHandler };

const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-
software-token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
```

```
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-
initiate-auth' command.`,
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    logger.log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
```

```
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      logger.log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      logger.log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    logger.error(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
```

```
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    logger.log("Successfully authenticated.");
  } catch (err) {
    logger.error(err);
  }
};
```

```
export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    logger.log("Verifying TOTP.");
```

```
    await verifySoftwareToken(totp);
    logger.log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    logger.error(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers

- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 MFA.
```

```
 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
 script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a
 new code? (Yes/No)")
    val ans = inOb.nextLine()
```

```
    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }
```

```
    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
```

```
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
```

```
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
```

```
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp

- VerifySoftwareToken

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que englobe as funções do Amazon Cognito usadas no cenário.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.

        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
```

```
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
```

```
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
```

```
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return users


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
```

```
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

```
    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
of
        a two-factor sign-in. When sign-in is successful, it returns an access
token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
```

```
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
```

```
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
```

```
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
```

```
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
```

```
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
 user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
 self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

Crie uma classe que execute o cenário. Este exemplo também registra um dispositivo de MFA a ser rastreado pelo Amazon Cognito e mostra como fazer login usando uma senha e informações do dispositivo rastreado. Isso evita a necessidade de inserir um novo código de MFA.

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)
```

```
    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )

    user_name = q.ask("Let's sign up a new user. Enter a user name: ",
q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ",
q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print("-" * 88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
```

```
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response["ChallengeName"]
        elif response["ChallengeName"] == "MFA_SETUP":
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA
"
                "application, such as Google Authenticator."
            )
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to
appear in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                    user_name, response["Session"], mfa_code
                )
            print(f"You're signed in as {user_name}.")
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
            print("And your device information:")
            pp(auth_tokens["NewDeviceMetadata"])
```

```
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge
 {response['ChallengeName']}")
    print("-" * 88)

    device_group_key = auth_tokens["NewDeviceMetadata"]["DeviceGroupKey"]
    device_key = auth_tokens["NewDeviceMetadata"]["DeviceKey"]
    device_password = base64.standard_b64encode(os.urandom(40)).decode("utf-8")

    print("Let's confirm your MFA device so you don't have re-enter MFA tokens
 for it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name,
        device_key,
        device_group_key,
        device_password,
        auth_tokens["AccessToken"],
        aws_srp,
    )
    print(f"Your device {device_key} is confirmed.")
    print("-" * 88)

    print(
        f"Now let's sign in as {user_name} from your confirmed device
 {device_key}.\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code."
    )
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password,
 aws_srp
    )
    print("You're signed in. Your access token is:")
    pp(auth_tokens["AccessToken"])
    print("-" * 88)

    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
    print("-" * 88)
```

```
def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the
 example."
    )
    args = parser.parse_args()
    try:
        run_scenario(boto3.client("cognito-idp"), args.user_pool_id,
 args.client_id)
    except Exception:
        logging.exception("Something went wrong with the demo.")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp

- VerifySoftwareToken

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "cognito-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
```

```
        // from dependencies.
        .executableTarget(
            name: "cognito-scenario",
            dependencies: [
                .product(name: "AWSCognitoIdentityProvider", package: "aws-sdk-
swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo de código do Swift.

```
// An example demonstrating various features of Amazon Cognito. Before running
// this Swift code example, set up your development environment, including
// your credentials.
//
// For more information, see the following documentation:
// https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
//
// TIP: To set up the required user pool, run the AWS Cloud Development Kit
// (AWS CDK) script provided in this GitHub repo at
// resources/cdk/cognito_scenario_user_pool_with_mfa.
//
// This example performs the following functions:
//
// 1. Invokes the signUp method to sign up a user.
// 2. Invokes the adminGetUser method to get the user's confirmation status.
// 3. Invokes the ResendConfirmationCode method if the user requested another
//    code.
// 4. Invokes the confirmSignUp method.
// 5. Invokes the initiateAuth to sign in. This results in being prompted to
//    set up TOTP (time-based one-time password). (The response is
//    “ChallengeName”: “MFA_SETUP”).
// 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
//    key. This can be used with Google Authenticator.
// 7. Invokes the VerifySoftwareToken method to verify the TOTP and register
//    for MFA.
// 8. Invokes the AdminInitiateAuth to sign in again. This results in being
```

```
//    prompted to submit a TOTP (Response: “ChallengeName”:
//    “SOFTWARE_TOKEN_MFA”).
// 9. Invokes the AdminRespondToAuthChallenge to get back a token.

import ArgumentParser
import Foundation

import AWSClientRuntime
import AWSCognitoIdentityProvider

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The application clientId.")
    var clientId: String
    @Argument(help: "The user pool ID to use.")
    var poolId: String
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "cognito-scenario",
        abstract: """
        Demonstrates various features of Amazon Cognito.
        """,
        discussion: """
        """
    )

    /// Prompt for an input string of at least a minimum length.
    ///
    /// - Parameters:
    ///   - prompt: The prompt string to display.
    ///   - minLength: The minimum number of characters to allow in the
    ///     response. Default value is 0.
    ///
    /// - Returns: The entered string.
    func stringRequest(_ prompt: String, minLength: Int = 1) -> String {
        while true {
            print(prompt, terminator: "")
            let str = readLine()

            guard let str else {
                continue
            }
            if str.count >= minLength {
```

```
                return str
            } else {
                print("*** Response must be at least \(minLength) character(s)
long.")
            }
        }
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(_ prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
```

```
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )

            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
```

```
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
clientId: String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
else {
                print("*** Unable to get the delivery method for the resent
code.")
                return false
            }
```

```
            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user
\(userName).")
            return false
        }
    }

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise
`false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }

    /// Begin an authentication session.
    ///
```

```
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
```

```
            return nil
        }
    }

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
app's MFA.")
            return nil
        }
    }

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
```

```
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
 mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
```

```
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                     clientId: String, userPoolId: String,
mfaCode: String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
```

```
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await
CognitoIdentityProviderClient.CognitoIdentityProviderClientConfiguration(region:
region)
        let cipClient = CognitoIdentityProviderClient(config: config)

        print("""
              This example collects information about a user, then creates that
user in the
              specified user pool. Then, it enables Multi-Factor Authentication
(MFA) for that
              user by associating an authenticator application (such as Google
Authenticator
              or a password manager that supports TOTP). Then, the user uses a
code from their
              authenticator application to sign in.

              """)

        let userName = stringRequest("Please enter a new username: ")
        let password = stringRequest("Enter a password: ")
        let email = stringRequest("Enter your email address: ", minLength: 5)

        // Submit the sign-up request to AWS.

        print("==> Signing up user \(userName)...")
        if await signUp(cipClient: cipClient, clientId: clientId,
                        userName: userName, password: password,
                        email: email) == false {
```

```
            return
        }

        // Check the user's status. This time, it should come back "unconfirmed".

        print("==> Getting the status of user \(userName) from the user pool
(should be 'unconfirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
            return
        }

        // Ask the user if they want a replacement code sent, such as if the
        // code hasn't arrived yet. If the user responds with a "yes," send a
        // new code.

        if yesNoRequest("==> A confirmation code was sent to \(userName). Would
you like to send a new code (Y/N)? ") {
            print("==> Sending a new confirmation code...")
            if await resendConfirmationCode(cipClient: cipClient, clientId:
clientId, userName: userName) == false {
                return
            }
        }

        // Ask the user to enter the confirmation code, then send it to Amazon
        // Cognito to verify it.

        let code = stringRequest("==> Enter the confirmation code sent to
\(userName): ")
        if await confirmSignUp(cipClient: cipClient, clientId: clientId,
userName: userName, code: code) == false {
            // The code didn't match. Your application may wish to offer to
            // re-send the confirmation code here and try again.
            return
        }

        // Check the user's status again. This time it should come back
        // "confirmed".

        print("==> Rechecking status of user \(userName) in the user pool (should
be 'confirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
```

```
            return
        }
        // Check the challenge mode. Here, it should be "mfaSetup", indicating
        // that the user needs to add MFA before using it. This returns a
        // session that can be used to register MFA, or nil if an error occurs.

        let authSession = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                             userName: userName, password:
password,
                                             userPoolId: poolId)
        if authSession == nil {
            return
        }

        // Ask Cognito for an MFA secret token that the user should enter into
        // their authenticator software (such as Google Authenticator) or
        // password manager to configure it for this user account. This
        // returns a new session that should be used for the new stage of the
        // authentication process.

        let newSession = await getSecretForAppMFA(cipClient: cipClient,
authSession: authSession)
        if newSession == nil {
            return
        }

        // Ask the user to enter the current 6-digit code displayed by their
        // authenticator. Then verify that it matches the value expected for
        // the session.

        let mfaCode1 = stringRequest("==> Enter the 6-digit code displayed in
your authenticator: ",
                                     minLength: 6)
        await verifyTOTP(cipClient: cipClient, session: newSession, mfaCode:
mfaCode1)

        // Ask the user to authenticate now that the authenticator has been
        // configured. This creates a new session using the user's username
        // and password as already entered.

        print("\nNow starting the sign-in process for user \(userName)...\n")
```

```
        let session2 = await initiateAuth(cipClient: cipClient, clientId:
 clientId,
                                    userName: userName, password: password,
 userPoolId: poolId)
        guard let session2 else {
            return
        }

        // Now that we have a new auth session, `session2`, ask the user for a
        // new 6-digit code from their authenticator, and send it to the auth
        // session.

        let mfaCode2 = stringRequest("==> Wait for your authenticator to show a
 new 6-digit code, then enter it: ",
                                    minLength: 6)
        await adminRespondToAuthChallenge(cipClient: cipClient, userName:
 userName,
                                          clientId: clientId, userPoolId: poolId,
                                          mfaCode: mfaCode2, session: session2)
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - AdminGetUser
  - AdminInitiateAuth

- AdminRespondToAuthChallenge
- AssociateSoftwareToken
- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Usar bancos de identidades e fluxos de identidades do Amazon Cognito

O exemplo de código a seguir mostra como criar uma aplicação de demonstração baseada na web que demonstra fluxos de identidades de bancos de identidades.

Python

SDK para Python (Boto3)

Mostra uma aplicação de demonstração baseada na web que demonstra os fluxos de autenticação dos bancos de identidades do Amazon Cognito, permitindo que os usuários explorem interativamente os fluxos de autenticação aprimorada e básica com vários provedores de identidade.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Provedor de identidades do Amazon Cognito

Grave dados de atividades personalizados com uma função Lambda após a autenticação do usuário do Amazon Cognito usando um AWS SDK

O exemplo de código a seguir mostra como gravar dados de atividade personalizados com uma função do Lambda depois da autenticação do usuário do Amazon Cognito.

- Use as funções de administrador para adicionar um usuário a um grupo de usuários.
- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PostAuthentication`.
- Faça login do novo usuário no Amazon Cognito.
- A função Lambda grava informações personalizadas em CloudWatch Logs e em uma tabela do DynamoDB.
- Obtenha e veja dados personalizados da tabela do DynamoDB e, em seguida, limpe os recursos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
```

```
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
```

```
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
     password = runner.questioner.AskPassword("\nEnter another password:", 8)
    } else {
     panic(err)
    }
   } else {
    pwSet = true
   }
  }

  log.Println(strings.Repeat("-", 88))

  return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string,
 userName string, password string) {
```

```
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()
```

```
 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `PostAuthentication` com uma função do Lambda.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
```

```
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
```

```
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
```

```
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}
```

```
// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
```

```
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
```

```
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
```

```
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

```
// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
```

```
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
```

```
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}
```

```
// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
```

```
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
```

```
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}
```

```
// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}
```

```
func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
```

```
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Exemplos de código para o Amazon Cognito Sync usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Cognito Sync com um kit de desenvolvimento AWS de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Amazon Cognito Sync: mais informações sobre o Amazon Cognito Sync.
- Referência da API Amazon Cognito Sync: detalhes sobre todas as ações disponíveis do Amazon Cognito Sync.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Amazon Cognito Sync usando AWS SDKs
  - Ações para o Amazon Cognito Sync usando AWS SDKs
    - Use ListIdentityPoolUsagecom um AWS SDK

Exemplos básicos do Amazon Cognito Sync usando AWS SDKs

Os exemplos de código a seguir mostram como usar fundamentos do Amazon Cognito Sync com SDKs do AWS .

Exemplos

- Ações para o Amazon Cognito Sync usando AWS SDKs
  - Use ListIdentityPoolUsagecom um AWS SDK

Ações para o Amazon Cognito Sync usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Cognito Sync com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para ter uma lista completa, consulte a Referência de API do Amazon Cognito Sync.

Exemplos

- Use ListIdentityPoolUsagecom um AWS SDK

Use **`ListIdentityPoolUsage`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListIdentityPoolUsage`.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());
```

```
    Ok(())
}
```

- Para obter detalhes da API, consulte a ListIdentityPoolUsagereferência da API AWS SDK for Rust.

Exemplos de código para o Amazon Comprehend usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Comprehend com AWS um kit de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Comprehend: mais informações sobre o Amazon Comprehend.
- Referência da API Amazon Comprehend: detalhes sobre todas as ações disponíveis do Amazon Comprehend.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon Comprehend usando AWS SDKs
  - Ações para o Amazon Comprehend usando AWS SDKs
    - Use CreateDocumentClassifiercom um AWS SDK ou CLI
    - Use DeleteDocumentClassifiercom um AWS SDK ou CLI
    - Use DescribeDocumentClassificationJobcom um AWS SDK ou CLI

- Use DescribeDocumentClassifiercom um AWS SDK ou CLI
- Use DescribeTopicsDetectionJobcom um AWS SDK ou CLI
- Use DetectDominantLanguagecom um AWS SDK ou CLI
- Use DetectEntitiescom um AWS SDK ou CLI
- Use DetectKeyPhrasescom um AWS SDK ou CLI
- Use DetectPiiEntitiescom um AWS SDK ou CLI
- Use DetectSentimentcom um AWS SDK ou CLI
- Use DetectSyntaxcom um AWS SDK ou CLI
- Use ListDocumentClassificationJobscom um AWS SDK ou CLI
- Use ListDocumentClassifierscom um AWS SDK ou CLI
- Use ListTopicsDetectionJobscom um AWS SDK ou CLI
- Use StartDocumentClassificationJobcom um AWS SDK ou CLI
- Use StartTopicsDetectionJobcom um AWS SDK ou CLI

- Cenários para o Amazon Comprehend usando AWS SDKs
  - Criar uma aplicação de transmissão do Amazon Transcribe
  - Crie um chatbot do Amazon Lex para engajar os visitantes do site
  - Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS
  - Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
  - Detecte elementos do documento com o Amazon Comprehend e um AWS SDK
  - Detecte entidades no texto extraído de uma imagem usando um AWS SDK
  - Execute um trabalho de modelagem de tópicos do Amazon Comprehend em dados de amostra usando um AWS SDK
  - Treine um classificador Amazon Comprehend personalizado e classifique documentos usando um AWS SDK

Exemplos básicos para o Amazon Comprehend usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Comprehend com AWS SDKs.

Exemplos

- Ações para o Amazon Comprehend usando AWS SDKs

- Use CreateDocumentClassifiercom um AWS SDK ou CLI
- Use DeleteDocumentClassifiercom um AWS SDK ou CLI
- Use DescribeDocumentClassificationJobcom um AWS SDK ou CLI
- Use DescribeDocumentClassifiercom um AWS SDK ou CLI
- Use DescribeTopicsDetectionJobcom um AWS SDK ou CLI
- Use DetectDominantLanguagecom um AWS SDK ou CLI
- Use DetectEntitiescom um AWS SDK ou CLI
- Use DetectKeyPhrasescom um AWS SDK ou CLI
- Use DetectPiiEntitiescom um AWS SDK ou CLI
- Use DetectSentimentcom um AWS SDK ou CLI
- Use DetectSyntaxcom um AWS SDK ou CLI
- Use ListDocumentClassificationJobscom um AWS SDK ou CLI
- Use ListDocumentClassifierscom um AWS SDK ou CLI
- Use ListTopicsDetectionJobscom um AWS SDK ou CLI
- Use StartDocumentClassificationJobcom um AWS SDK ou CLI
- Use StartTopicsDetectionJobcom um AWS SDK ou CLI

Ações para o Amazon Comprehend usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Comprehend com SDKs. AWS Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Comprehend e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Comprehend usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de API do Amazon Comprehend.

Exemplos

- Use CreateDocumentClassifiercom um AWS SDK ou CLI
- Use DeleteDocumentClassifiercom um AWS SDK ou CLI
- Use DescribeDocumentClassificationJobcom um AWS SDK ou CLI

- Use DescribeDocumentClassifiercom um AWS SDK ou CLI
- Use DescribeTopicsDetectionJobcom um AWS SDK ou CLI
- Use DetectDominantLanguagecom um AWS SDK ou CLI
- Use DetectEntitiescom um AWS SDK ou CLI
- Use DetectKeyPhrasescom um AWS SDK ou CLI
- Use DetectPiiEntitiescom um AWS SDK ou CLI
- Use DetectSentimentcom um AWS SDK ou CLI
- Use DetectSyntaxcom um AWS SDK ou CLI
- Use ListDocumentClassificationJobscom um AWS SDK ou CLI
- Use ListDocumentClassifierscom um AWS SDK ou CLI
- Use ListTopicsDetectionJobscom um AWS SDK ou CLI
- Use StartDocumentClassificationJobcom um AWS SDK ou CLI
- Use StartTopicsDetectionJobcom um AWS SDK ou CLI

Use **`CreateDocumentClassifier`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateDocumentClassifier`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Treinar um classificador personalizado e classificar documentos

CLI

AWS CLI

Para criar um classificador de documentos para categorizar documentos

O exemplo de `create-document-classifier` a seguir inicia o processo de treinamento para um modelo de classificador de documentos. O arquivo de dados de treinamento, `training.csv`, está localizado na tag `--input-data-config`. `training.csv` é um documento de duas colunas em que os rótulos ou classificações são fornecidos na primeira coluna e os documentos são fornecidos na segunda coluna.

```
aws comprehend create-document-classifier \
```

```
    --document-classifier-name example-classifier \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket/" \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket/" \
    --language-code en
```

Saída:

```
{
    "DocumentClassifierArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/example-classifier"
}
```

Para obter mais informações, consulte Classificação personalizada no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte CreateDocumentClassifierem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
 software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
```

```
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-
using-amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
                  dataAccessRoleArn - The ARN value of the role used for this
 operation.
                  s3Uri - The Amazon S3 bucket that contains the CSV file.
                  documentClassifierName - The name of the document classifier.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dataAccessRoleArn = args[0];
        String s3Uri = args[1];
        String documentClassifierName = args[2];

        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
 documentClassifierName);
        comClient.close();
    }
```

```
    public static void createDocumentClassifier(ComprehendClient comClient,
 String dataAccessRoleArn, String s3Uri,
            String documentClassifierName) {
        try {
            DocumentClassifierInputDataConfig config =
 DocumentClassifierInputDataConfig.builder()
                    .s3Uri(s3Uri)
                    .build();

            CreateDocumentClassifierRequest createDocumentClassifierRequest =
 CreateDocumentClassifierRequest.builder()
                    .documentClassifierName(documentClassifierName)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .languageCode("en")
                    .inputDataConfig(config)
                    .build();

            CreateDocumentClassifierResponse createDocumentClassifierResult =
 comClient
                    .createDocumentClassifier(createDocumentClassifierRequest);
            String documentClassifierArn =
 createDocumentClassifierResult.documentClassifierArn();
            System.out.println("Document Classifier ARN: " +
 documentClassifierArn);

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

- Para obter detalhes da API, consulte CreateDocumentClassifiera Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it
 immediately
        starts training on the data found in the specified Amazon S3 bucket.
 Training
        can take 30 minutes or longer. The `describe_document_classifier`
 function
        can be used to get training status and returns a status of TRAINED when
 the
        classifier is ready to use.

        :param name: The name of the classifier.
```

```
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
 data.
        :param training_key: The prefix used to find training data in the
 training
                             bucket. If multiple objects have the same prefix,
 all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
 that
                                     grants Comprehend permission to read from
 the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/
{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
 self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn
```

- Para obter detalhes da API, consulte a CreateDocumentClassifierReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->createdocumentclassifier(
      iv_documentclassifiername = iv_classifier_name
      iv_languagecode = iv_language_code
      io_inputdataconfig = NEW /aws1/cl_cpddocclifierinpdat00(
        iv_s3uri = iv_training_s3_uri
      )
      iv_dataaccessrolearn = iv_data_access_role_arn
      iv_mode = iv_mode
    ).
    MESSAGE 'Document classifier creation started.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdresrclimitexcdex.
    MESSAGE 'Resource limit exceeded.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanytagsex.
    MESSAGE 'Too many tags.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDocumentClassifierreferência da API AWS SDK for SAP ABAP.

Use **`DeleteDocumentClassifier`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteDocumentClassifier`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Treinar um classificador personalizado e classificar documentos

CLI

AWS CLI

Para excluir um classificador de documentos personalizado

O exemplo de `delete-document-classifier` a seguir exclui um modelo de classificador de documentos personalizado.

```
aws comprehend delete-document-classifier \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/example-classifier-1
```

Este comando não produz saída.

Para obter mais informações, consulte Gerenciar endpoints do Amazon Comprehend no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DeleteDocumentClassifierem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
```

```
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.",
self.classifier_arn)
            raise
```

- Para obter detalhes da API, consulte a DeleteDocumentClassifierReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->deletedocumentclassifier(
          iv_documentclassifierarn = iv_classifier_arn
        ).
        MESSAGE 'Document classifier deleted.' TYPE 'I'.
```

```
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourceinuseex.
        MESSAGE 'Resource in use.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDocumentClassifierreferência da API AWS SDK for SAP ABAP.

Use **`DescribeDocumentClassificationJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeDocumentClassificationJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Treinar um classificador personalizado e classificar documentos

CLI

AWS CLI

Para descrever um trabalho de classificação de documentos

O exemplo de `describe-document-classification-job` a seguir obtém as propriedades de um trabalho assíncrono de classificação de documentos.

```
aws comprehend describe-document-classification-job \
    --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

Saída:

```
{
    "DocumentClassificationJobProperties": {
```

```
        "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
        "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
        "JobName": "exampleclassificationjob",
        "JobStatus": "COMPLETED",
        "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
        "EndTime": "2023-06-14T17:15:58.582000+00:00",
        "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/mymodel/version/1",
        "InputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "OutputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-destination-bucket/
testfolder/111122223333-CLN-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-servicerole"
    }
}
```

Para obter mais informações, consulte Classificação personalizada no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DescribeDocumentClassificationJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response =
self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job
```

- Para obter detalhes da API, consulte a DescribeDocumentClassificationJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->describedocclassificationjob(
          iv_jobid = iv_job_id
        ).
        MESSAGE 'Document classification job described.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdjobnotfoundex.
        MESSAGE 'Job not found.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDocumentClassificationJobreferência da API AWS SDK for SAP ABAP.

Use **`DescribeDocumentClassifier`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeDocumentClassifier`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Treinar um classificador personalizado e classificar documentos

CLI

AWS CLI

Para descrever um classificador de documentos

O exemplo de `describe-document-classifier` a seguir obtém as propriedades de um modelo de classificador de documentos personalizado.

```
aws comprehend describe-document-classifier \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/example-classifier-1
```

Saída:

```
{
    "DocumentClassifierProperties": {
        "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/example-classifier-1",
        "LanguageCode": "en",
        "Status": "TRAINED",
        "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
        "EndTime": "2023-06-13T19:42:31.752000+00:00",
        "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
        "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
        "InputDataConfig": {
            "DataFormat": "COMPREHEND_CSV",
            "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
        },
        "OutputDataConfig": {},
        "ClassifierMetadata": {
            "NumberOfLabels": 3,
            "NumberOfTrainedDocuments": 5016,
            "NumberOfTestDocuments": 557,
            "EvaluationMetrics": {
                "Accuracy": 0.9856,
                "Precision": 0.9919,
                "Recall": 0.9459,
                "F1Score": 0.9673,
                "MicroPrecision": 0.9856,
                "MicroRecall": 0.9856,
                "MicroF1Score": 0.9856,
                "HammingLoss": 0.0144
            }
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role",
        "Mode": "MULTI_CLASS"
    }
}
```

Para obter mais informações, consulte Criação e gerenciamento de modelos personalizados no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DescribeDocumentClassifierem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier
```

- Para obter detalhes da API, consulte a DescribeDocumentClassifierReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->describedocumentclassifier(
      iv_documentclassifierarn = iv_classifier_arn
    ).
    MESSAGE 'Document classifier described.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdresourcenotfoundex.
    MESSAGE 'Resource not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDocumentClassifierreferência da API AWS SDK for SAP ABAP.

Use **`DescribeTopicsDetectionJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeTopicsDetectionJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Execute uma tarefa de modelagem de tópicos em dados de amostra

CLI

AWS CLI

Para descrever um trabalho de detecção de tópicos

O exemplo de `describe-topics-detection-job` a seguir obtém as propriedades de um trabalho assíncrono de detecção de tópicos.

```
aws comprehend describe-topics-detection-job \
    --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

Saída:

```
{
    "TopicsDetectionJobProperties": {
        "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
        "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
        "JobName": "example_topics_detection",
        "JobStatus": "IN_PROGRESS",
        "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
        "InputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-bucket",
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "OutputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-destination-bucket/
testfolder/111122223333-TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
        },
        "NumberOfTopics": 10,
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-examplerole"
    }
}
```

Para obter mais informações, consulte Análise assíncrona para insights do Amazon Comprehend no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DescribeTopicsDetectionJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job
```

- Para obter detalhes da API, consulte a DescribeTopicsDetectionJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->describetopicsdetectionjob(
      iv_jobid = iv_job_id
    ).
    MESSAGE 'Topics detection job described.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdjobnotfoundex.
    MESSAGE 'Job not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeTopicsDetectionJobreferência da API AWS SDK for SAP ABAP.

Use **`DetectDominantLanguage`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectDominantLanguage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Detectar elementos do documento

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// This example calls the Amazon Comprehend service to determine the
/// dominant language.
/// </summary>
public static class DetectDominantLanguage
{
    /// <summary>
    /// Calls Amazon Comprehend to determine the dominant language used in
    /// the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle.";

        var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

        Console.WriteLine("Calling DetectDominantLanguage\n");
        var detectDominantLanguageRequest = new
DetectDominantLanguageRequest()
        {
```

```
                Text = text,
            };

            var detectDominantLanguageResponse = await
 comprehendClient.DetectDominantLanguageAsync(detectDominantLanguageRequest);
            foreach (var dl in detectDominantLanguageResponse.Languages)
            {
                Console.WriteLine($"Language Code: {dl.LanguageCode}, Score:
 {dl.Score}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectDominantLanguagea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para detectar o idioma dominante do texto de entrada

O `detect-dominant-language` a seguir analisa o texto de entrada e identifica o idioma dominante. A pontuação de confiança do modelo pré-treinado também é gerada.

```
aws comprehend detect-dominant-language \
    --text "It is a beautiful day in Seattle."
```

Saída:

```
{
    "Languages": [
        {
            "LanguageCode": "en",
            "Score": 0.9877256155014038
        }
    ]
```

```
}
```

Para obter mais informações, consulte Idioma dominante no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DetectDominantLanguageem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
```

```
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient,
 String text) {
        try {
            DetectDominantLanguageRequest request =
 DetectDominantLanguageRequest.builder()
                    .text(text)
                    .build();

            DetectDominantLanguageResponse resp =
 comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

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

- Para obter detalhes da API, consulte DetectDominantLanguagea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages
```

- Para obter detalhes da API, consulte a DetectDominantLanguageReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->detectdominantlanguage( iv_text = iv_text ).
    MESSAGE 'Languages detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectDominantLanguagereferência da API AWS SDK for SAP ABAP.

Use **`DetectEntities`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectEntities`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Detectar elementos do documento

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// This example shows how to use the AmazonComprehend service detect any
/// entities in submitted text.
/// </summary>
public static class DetectEntities
{
    /// <summary>
    /// The main method calls the DetectEntitiesAsync method to find any
    /// entities in the sample code.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new AmazonComprehendClient();

        Console.WriteLine("Calling DetectEntities\n");
        var detectEntitiesRequest = new DetectEntitiesRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        var detectEntitiesResponse = await
comprehendClient.DetectEntitiesAsync(detectEntitiesRequest);

        foreach (var e in detectEntitiesResponse.Entities)
        {
```

```
                Console.WriteLine($"Text: {e.Text}, Type: {e.Type}, Score:
 {e.Score}, BeginOffset: {e.BeginOffset}, EndOffset: {e.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectEntiesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para detectar entidades nomeadas no texto de entrada

O exemplo de `detect-entities` a seguir analisa o texto de entrada e retorna as entidades nomeadas. A pontuação de confiança do modelo pré-treinado também é gerada para cada previsão.

```
aws comprehend detect-entities \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
    account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by
 July 31st. Based on your autopay settings, \
    we will withdraw your payment on the due date from your bank account number
 XXXXXX1111 with the routing number XXXXX0000. \
    Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to
 Alice at AnySpa@example.com."
```

Saída:

```
{
    "Entities": [
        {
            "Score": 0.9994556307792664,
            "Type": "PERSON",
            "Text": "Zhang Wei",
```

```
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9981022477149963,
            "Type": "PERSON",
            "Text": "John",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.9986887574195862,
            "Type": "ORGANIZATION",
            "Text": "AnyCompany Financial Services, LLC",
            "BeginOffset": 33,
            "EndOffset": 67
        },
        {
            "Score": 0.9959119558334351,
            "Type": "OTHER",
            "Text": "1111-XXXX-1111-XXXX",
            "BeginOffset": 88,
            "EndOffset": 107
        },
        {
            "Score": 0.9708039164543152,
            "Type": "QUANTITY",
            "Text": ".53",
            "BeginOffset": 133,
            "EndOffset": 136
        },
        {
            "Score": 0.9987268447875977,
            "Type": "DATE",
            "Text": "July 31st",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9858865737915039,
            "Type": "OTHER",
            "Text": "XXXXXX1111",
            "BeginOffset": 271,
            "EndOffset": 281
```

```
        },
        {
            "Score": 0.9700471758842468,
            "Type": "OTHER",
            "Text": "XXXXX0000",
            "BeginOffset": 306,
            "EndOffset": 315
        },
        {
            "Score": 0.9591118693351746,
            "Type": "ORGANIZATION",
            "Text": "Sunshine Spa",
            "BeginOffset": 340,
            "EndOffset": 352
        },
        {
            "Score": 0.9797496795654297,
            "Type": "LOCATION",
            "Text": "123 Main St",
            "BeginOffset": 354,
            "EndOffset": 365
        },
        {
            "Score": 0.994929313659668,
            "Type": "PERSON",
            "Text": "Alice",
            "BeginOffset": 394,
            "EndOffset": 399
        },
        {
            "Score": 0.9949769377708435,
            "Type": "OTHER",
            "Text": "AnySpa@example.com",
            "BeginOffset": 403,
            "EndOffset": 418
        }
    ]
}
```

Para obter mais informações, consulte Entidades no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DetectEntiesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }
```

```
    public static void detectAllEntities(ComprehendClient comClient, String text)
 {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
 DetectEntitiesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectEntitiesResponse detectEntitiesResult =
 comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }

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

- Para obter detalhes da API, consulte DetectEntiticsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities
```

- Para obter detalhes da API, consulte a DetectEntitiesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->detectentities(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'Entities detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectEntiesreferência da API AWS SDK for SAP ABAP.

Use **`DetectKeyPhrases`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectKeyPhrases`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Detectar elementos do documento

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// This example shows how to use the Amazon Comprehend service to
/// search text for key phrases.
/// </summary>
public static class DetectKeyPhrase
{
    /// <summary>
    /// This method calls the Amazon Comprehend method DetectKeyPhrasesAsync
    /// to detect any key phrases in the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

        // Call DetectKeyPhrases API
        Console.WriteLine("Calling DetectKeyPhrases");
        var detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        var detectKeyPhrasesResponse = await
comprehendClient.DetectKeyPhrasesAsync(detectKeyPhrasesRequest);
        foreach (var kp in detectKeyPhrasesResponse.KeyPhrases)
```

```
            {
                Console.WriteLine($"Text: {kp.Text}, Score: {kp.Score},
 BeginOffset: {kp.BeginOffset}, EndOffset: {kp.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectKeyPhrasesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para detectar frases-chave no texto de entrada

O exemplo de `detect-key-phrases` a seguir analisa o texto de entrada e identifica as principais frases nominais. A pontuação de confiança do modelo pré-treinado também é gerada para cada previsão.

```
aws comprehend detect-key-phrases \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
        account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due
 by July 31st. Based on your autopay settings, \
        we will withdraw your payment on the due date from your bank account
 number XXXXXX1111 with the routing number XXXXX0000. \
        Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments
 to Alice at AnySpa@example.com."
```

Saída:

```
{
    "KeyPhrases": [
        {
            "Score": 0.8996376395225525,
            "Text": "Zhang Wei",
```

```
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9992469549179077,
            "Text": "John",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.988385021686554,
            "Text": "Your AnyCompany Financial Services",
            "BeginOffset": 28,
            "EndOffset": 62
        },
        {
            "Score": 0.8740853071212769,
            "Text": "LLC credit card account 1111-XXXX-1111-XXXX",
            "BeginOffset": 64,
            "EndOffset": 107
        },
        {
            "Score": 0.9999437928199768,
            "Text": "a minimum payment",
            "BeginOffset": 112,
            "EndOffset": 129
        },
        {
            "Score": 0.9998900890350342,
            "Text": ".53",
            "BeginOffset": 133,
            "EndOffset": 136
        },
        {
            "Score": 0.9979453086853027,
            "Text": "July 31st",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9983011484146118,
            "Text": "your autopay settings",
            "BeginOffset": 172,
            "EndOffset": 193
```

```
        },
        {
            "Score": 0.9996572136878967,
            "Text": "your payment",
            "BeginOffset": 211,
            "EndOffset": 223
        },
        {
            "Score": 0.9995037317276001,
            "Text": "the due date",
            "BeginOffset": 227,
            "EndOffset": 239
        },
        {
            "Score": 0.9702621698379517,
            "Text": "your bank account number XXXXXX1111",
            "BeginOffset": 245,
            "EndOffset": 280
        },
        {
            "Score": 0.9179925918579102,
            "Text": "the routing number XXXXX0000.Customer feedback",
            "BeginOffset": 286,
            "EndOffset": 332
        },
        {
            "Score": 0.9978160858154297,
            "Text": "Sunshine Spa",
            "BeginOffset": 337,
            "EndOffset": 349
        },
        {
            "Score": 0.9706913232803345,
            "Text": "123 Main St",
            "BeginOffset": 351,
            "EndOffset": 362
        },
        {
            "Score": 0.9941995143890381,
            "Text": "comments",
            "BeginOffset": 379,
            "EndOffset": 387
        },
        {
```

```
            "Score": 0.9759287238121033,
            "Text": "Alice",
            "BeginOffset": 391,
            "EndOffset": 396
        },
        {
            "Score": 0.8376792669296265,
            "Text": "AnySpa@example.com",
            "BeginOffset": 400,
            "EndOffset": 415
        }
    ]
}
```

Para obter mais informações, consulte Frases-chave no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DetectKeyPhrasesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String
 text) {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
 DetectKeyPhrasesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
 comClient.detectKeyPhrases(detectKeyPhrasesRequest);
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }

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

- Para obter detalhes da API, consulte DetectKeyPhrasesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and
 its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
```

```
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases
```

- Para obter detalhes da API, consulte a DetectKeyPhrasesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->detectkeyphrases(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'Key phrases detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectKeyPhrasesreferência da API AWS SDK for SAP ABAP.

Use **`DetectPiiEntities`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectPiiEntities`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Detectar elementos do documento

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// This example shows how to use the Amazon Comprehend service to find
/// personally identifiable information (PII) within text submitted to the
/// DetectPiiEntitiesAsync method.
/// </summary>
public class DetectingPII
{
    /// <summary>
    /// This method calls the DetectPiiEntitiesAsync method to locate any
    /// personally dientifiable information within the supplied text.
    /// </summary>
    public static async Task Main()
    {
        var comprehendClient = new AmazonComprehendClient();
        var text = @"Hello Paul Santos. The latest statement for your
                    credit card account 1111-0000-1111-0000 was
                    mailed to 123 Any Street, Seattle, WA 98109.";
```

```
            var request = new DetectPiiEntitiesRequest
            {
                Text = text,
                LanguageCode = "EN",
            };

            var response = await
 comprehendClient.DetectPiiEntitiesAsync(request);

            if (response.Entities.Count > 0)
            {
                foreach (var entity in response.Entities)
                {
                    var entityValue = text.Substring(entity.BeginOffset,
 entity.EndOffset - entity.BeginOffset);
                    Console.WriteLine($"{entity.Type}: {entityValue}");
                }
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectPiiEntitiesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para detectar entidades PII no texto de entrada

O exemplo de `detect-pii-entities` a seguir analisa o texto de entrada e identifica entidades que contêm informações de identificação pessoal (PII). A pontuação de confiança do modelo pré-treinado também é gerada para cada previsão.

```
aws comprehend detect-pii-entities \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
        account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due
 by July 31st. Based on your autopay settings, \
```

```
        we will withdraw your payment on the due date from your bank account
 number XXXXXX1111 with the routing number XXXXX0000. \
        Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments
 to Alice at AnySpa@example.com."
```

Saída:

```
{
    "Entities": [
        {
            "Score": 0.9998322129249573,
            "Type": "NAME",
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9998878240585327,
            "Type": "NAME",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.9994089603424072,
            "Type": "CREDIT_DEBIT_NUMBER",
            "BeginOffset": 88,
            "EndOffset": 107
        },
        {
            "Score": 0.9999760985374451,
            "Type": "DATE_TIME",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9999449253082275,
            "Type": "BANK_ACCOUNT_NUMBER",
            "BeginOffset": 271,
            "EndOffset": 281
        },
        {
            "Score": 0.9999847412109375,
            "Type": "BANK_ROUTING",
            "BeginOffset": 306,
```

```
            "EndOffset": 315
        },
        {
            "Score": 0.999925434589386,
            "Type": "ADDRESS",
            "BeginOffset": 354,
            "EndOffset": 365
        },
        {
            "Score": 0.9989161491394043,
            "Type": "NAME",
            "BeginOffset": 394,
            "EndOffset": 399
        },
        {
            "Score": 0.9994171857833862,
            "Type": "EMAIL",
            "BeginOffset": 403,
            "EndOffset": 418
        }
    ]
}
```

Para obter mais informações, consulte Informações de identificação pessoal (PII) no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DetectPiiEntiesem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can
be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities
```

- Para obter detalhes da API, consulte a DetectPiiEntitiesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->detectpiientities(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'PII entities detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectPiiEntitiesreferência da API AWS SDK for SAP ABAP.

Use **`DetectSentiment`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectSentiment`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Detectar elementos do documento

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// This example shows how to detect the overall sentiment of the supplied
/// text using the Amazon Comprehend service.
/// </summary>
public static class DetectSentiment
{
    /// <summary>
    /// This method calls the DetetectSentimentAsync method to analyze the
    /// supplied text and determine the overal sentiment.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

        // Call DetectKeyPhrases API
        Console.WriteLine("Calling DetectSentiment");
        var detectSentimentRequest = new DetectSentimentRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        var detectSentimentResponse = await
comprehendClient.DetectSentimentAsync(detectSentimentRequest);
        Console.WriteLine($"Sentiment: {detectSentimentResponse.Sentiment}");
```

```
            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectSentimenta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para detectar o sentimento de um texto de entrada

O exemplo de `detect-sentiment` a seguir analisa o texto de entrada e retorna uma inferência do sentimento predominante (`POSITIVE`, `NEUTRAL`, `MIXED` ou `NEGATIVE`).

```
aws comprehend detect-sentiment \
    --language-code en \
    --text "It is a beautiful day in Seattle"
```

Saída:

```
{
    "Sentiment": "POSITIVE",
    "SentimentScore": {
        "Positive": 0.9976957440376282,
        "Negative": 9.653854067437351e-05,
        "Neutral": 0.002169104292988777,
        "Mixed": 3.857641786453314e-05
    }
}
```

Para obter mais informações, consulte Sentimento no Guia do desenvolvedor do Amazon Comprehend

- Para obter detalhes da API, consulte DetectSentimentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }
```

```
    public static void detectSentiments(ComprehendClient comClient, String text)
 {
        try {
            DetectSentimentRequest detectSentimentRequest =
 DetectSentimentRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSentimentResponse detectSentimentResult =
 comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
 detectSentimentResult.sentimentScore().neutral());

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

- Para obter detalhes da API, consulte DetectSentimenta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
```

```
        """
        self.comprehend_client = comprehend_client


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DetectSentimentReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->detectsentiment(
```

```
          iv_text = iv_text
          iv_languagecode = iv_language_code
        ).
        MESSAGE 'Sentiment detected.' TYPE 'I'.
      CATCH /aws1/cx_cpdtextsizelmtexcdex.
        MESSAGE 'Text size exceeds limit.' TYPE 'E'.
      CATCH /aws1/cx_cpdunsuppedlanguageex.
        MESSAGE 'Unsupported language.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetectSentimentreferência da API AWS SDK for SAP ABAP.

Use **`DetectSyntax`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectSyntax`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Detectar elementos do documento

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
```

```
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use Amazon Comprehend to detect syntax
    /// elements by calling the DetectSyntaxAsync method.
    /// </summary>
    public class DetectingSyntax
    {
        /// <summary>
        /// This method calls DetectSynaxAsync to identify the syntax elements
        /// in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new AmazonComprehendClient();

            // Call DetectSyntax API
            Console.WriteLine("Calling DetectSyntaxAsync\n");
            var detectSyntaxRequest = new DetectSyntaxRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            DetectSyntaxResponse detectSyntaxResponse = await
comprehendClient.DetectSyntaxAsync(detectSyntaxRequest);
            foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
            {
                Console.WriteLine($"Text: {s.Text}, PartOfSpeech:
{s.PartOfSpeech.Tag}, BeginOffset: {s.BeginOffset}, EndOffset: {s.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectSyntaxa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para detectar as partes da fala em um texto de entrada

O exemplo de `detect-syntax` a seguir analisa a sintaxe do texto de entrada e retorna as diferentes partes da fala. A pontuação de confiança do modelo pré-treinado também é gerada para cada previsão.

```
aws comprehend detect-syntax \
    --language-code en \
    --text "It is a beautiful day in Seattle."
```

Saída:

```
{
    "SyntaxTokens": [
        {
            "TokenId": 1,
            "Text": "It",
            "BeginOffset": 0,
            "EndOffset": 2,
            "PartOfSpeech": {
                "Tag": "PRON",
                "Score": 0.9999740719795227
            }
        },
        {
            "TokenId": 2,
            "Text": "is",
            "BeginOffset": 3,
            "EndOffset": 5,
            "PartOfSpeech": {
                "Tag": "VERB",
                "Score": 0.999901294708252
            }
        },
        {
            "TokenId": 3,
            "Text": "a",
            "BeginOffset": 6,
            "EndOffset": 7,
```

```
            "PartOfSpeech": {
                "Tag": "DET",
                "Score": 0.9999938607215881
            }
        },
        {
            "TokenId": 4,
            "Text": "beautiful",
            "BeginOffset": 8,
            "EndOffset": 17,
            "PartOfSpeech": {
                "Tag": "ADJ",
                "Score": 0.9987351894378662
            }
        },
        {
            "TokenId": 5,
            "Text": "day",
            "BeginOffset": 18,
            "EndOffset": 21,
            "PartOfSpeech": {
                "Tag": "NOUN",
                "Score": 0.9999796748161316
            }
        },
        {
            "TokenId": 6,
            "Text": "in",
            "BeginOffset": 22,
            "EndOffset": 24,
            "PartOfSpeech": {
                "Tag": "ADP",
                "Score": 0.9998047947883606
            }
        },
        {
            "TokenId": 7,
            "Text": "Seattle",
            "BeginOffset": 25,
            "EndOffset": 32,
            "PartOfSpeech": {
                "Tag": "PROPN",
                "Score": 0.9940530061721802
            }
```

```
        }
    ]
}
```

Para obter mais informações, consulte Análise de sintaxe no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte DetectSyntaxem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
```

```
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest =
 DetectSyntaxRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSyntaxResponse detectSyntaxResult =
 comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
                System.out.println("Language is " + token.text());
                System.out.println("Part of speech is " +
 token.partOfSpeech().tagAsString());
            }

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

- Para obter detalhes da API, consulte DetectSyntaxa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

- Para obter detalhes da API, consulte a DetectSyntaxReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->detectsyntax(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'Syntax tokens detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectSyntaxreferência da API AWS SDK for SAP ABAP.

Use **`ListDocumentClassificationJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListDocumentClassificationJobs`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Treinar um classificador personalizado e classificar documentos

CLI

AWS CLI

Para listar trabalhos de classificação de documento

O exemplo de `list-document-classification-jobs` a seguir lista todos os trabalhos de classificação de documentos.

```
aws comprehend list-document-classification-jobs
```

Saída:

```
{
    "DocumentClassificationJobPropertiesList": [
        {
            "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
            "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
            "JobName": "exampleclassificationjob",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
            "EndTime": "2023-06-14T17:15:58.582000+00:00",
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/1234567890101-CLN-e758dd56b824aa717ceab551f11749fb/output/
output.tar.gz"
            },
            "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
```

```
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
            "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a1EXAMPLE2",
            "JobName": "exampleclassificationjob2",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-14T17:22:39.829000+00:00",
            "EndTime": "2023-06-14T17:28:46.107000+00:00",
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/1234567890101-CLN-123456abcdeb0e11022f22a1EXAMPLE2/output/
output.tar.gz"
            },
            "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
        }
    ]
}
```

Para obter mais informações, consulte Classificação personalizada no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte ListDocumentClassificationJobsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
```

```
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

- Para obter detalhes da API, consulte a ListDocumentClassificationJobsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->listdocclassificationjobs( ).
    MESSAGE 'Document classification jobs listed.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidfilterex.
    MESSAGE 'Invalid filter.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListDocumentClassificationJobsreferência da API AWS SDK for SAP ABAP.

Use **`ListDocumentClassifiers`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListDocumentClassifiers`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Treinar um classificador personalizado e classificar documentos

CLI

AWS CLI

Para listar todos os classificadores de documentos

O exemplo de `list-document-classifiers` a seguir lista todos os modelos de classificadores de documentos treinados e em treinamento.

```
aws comprehend list-document-classifiers
```

Saída:

```
{
    "DocumentClassifierPropertiesList": [
        {
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier1",
            "LanguageCode": "en",
            "Status": "TRAINED",
            "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
            "EndTime": "2023-06-13T19:42:31.752000+00:00",
            "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
            "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
            "InputDataConfig": {
                "DataFormat": "COMPREHEND_CSV",
                "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
            },
            "OutputDataConfig": {},
            "ClassifierMetadata": {
                "NumberOfLabels": 3,
                "NumberOfTrainedDocuments": 5016,
                "NumberOfTestDocuments": 557,
                "EvaluationMetrics": {
                    "Accuracy": 0.9856,
                    "Precision": 0.9919,
                    "Recall": 0.9459,
                    "F1Score": 0.9673,
                    "MicroPrecision": 0.9856,
                    "MicroRecall": 0.9856,
                    "MicroF1Score": 0.9856,
                    "HammingLoss": 0.0144
                }
            },
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-testorle",
            "Mode": "MULTI_CLASS"
        },
        {
```

```
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier2",
            "LanguageCode": "en",
            "Status": "TRAINING",
            "SubmitTime": "2023-06-13T21:20:28.690000+00:00",
            "InputDataConfig": {
                "DataFormat": "COMPREHEND_CSV",
                "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
            },
            "OutputDataConfig": {},
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-testorle",
            "Mode": "MULTI_CLASS"
        }
    ]
}
```

Para obter mais informações, consulte Criação e gerenciamento de modelos personalizados no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte ListDocumentClassifiersem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None
```

```
    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers
```

- Para obter detalhes da API, consulte a ListDocumentClassifiersReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->listdocumentclassifiers( ).
        MESSAGE 'Document classifiers listed.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
```

```
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidfilterex.
        MESSAGE 'Invalid filter.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListDocumentClassifiersreferência da API AWS SDK for SAP ABAP.

Use **`ListTopicsDetectionJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTopicsDetectionJobs`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Execute uma tarefa de modelagem de tópicos em dados de amostra

CLI

AWS CLI

Para listar todos os trabalhos de detecção de tópicos

O exemplo de `list-topics-detection-jobs` a seguir lista todos os trabalhos de detecção de tópicos assíncronos em andamento e concluídos.

```
aws comprehend list-topics-detection-jobs
```

Saída:

```
{
    "TopicsDetectionJobPropertiesList": [
        {
            "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a11EXAMPLE",
            "JobName" "topic-analysis-1"
            "JobStatus": "IN_PROGRESS",
```

```
            "SubmitTime": "2023-06-09T18:40:35.384000+00:00",
            "EndTime": "2023-06-09T18:46:41.936000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a1EXAMPLE2",
            "JobName": "topic-analysis-2",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
            "EndTime": "2023-06-09T18:50:50.872000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE2/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE3",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a1EXAMPLE3",
            "JobName": "topic-analysis-2",
            "JobStatus": "IN_PROGRESS",
            "SubmitTime": "2023-06-09T18:50:56.737000+00:00",
            "InputDataConfig": {
```

```
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE3/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        }
    ]
}
```

Para obter mais informações, consulte Análise assíncrona para insights do Amazon Comprehend no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte ListTopicsDetectionJobsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
```

```
    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

- Para obter detalhes da API, consulte a ListTopicsDetectionJobsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->listtopicsdetectionjobs( ).
        MESSAGE 'Topics detection jobs listed.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidfilterex.
        MESSAGE 'Invalid filter.' TYPE 'E'.
```

```
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTopicsDetectionJobsreferência da API AWS SDK for SAP ABAP.

Use **`StartDocumentClassificationJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartDocumentClassificationJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Treinar um classificador personalizado e classificar documentos

CLI

AWS CLI

Para iniciar um trabalho de classificação de documento

O exemplo de `start-document-classification-job` a seguir inicia um trabalho de classificação de documentos com um modelo personalizado em todos os arquivos no endereço especificado pela tag `--input-data-config`. Neste exemplo, o bucket do S3 de entrada contém `SampleSMStext1.txt`, `SampleSMStext2.txt` e `SampleSMStext3.txt`. O modelo foi previamente treinado na classificação de documentos de mensagens de spam e não spam, ou “ham”, e mensagens SMS. Quando o trabalho é concluído, `output.tar.gz` é colocado no local especificado pela tag `--output-data-config`. `output.tar.gz` contém `predictions.jsonl`, que lista a classificação de cada documento. A saída Json é impressa em uma linha por arquivo, mas foi formatada aqui para facilitar a leitura.

```
aws comprehend start-document-classification-job \
    --job-name exampleclassificationjob \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket-INPUT/jobdata/" \
    --output-data-config "S3Uri=s3://amzn-s3-demo-destination-bucket/testfolder/"
 \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
```

```
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/mymodel/version/12
```

Conteúdo de `SampleSMStext1.txt`:

```
"CONGRATULATIONS! TXT 2155550100 to win $5000"
```

Conteúdo de `SampleSMStext2.txt`:

```
"Hi, when do you want me to pick you up from practice?"
```

Conteúdo de `SampleSMStext3.txt`:

```
"Plz send bank account # to 2155550100 to claim prize!!"
```

Saída:

```
{
    "JobId": "e758dd56b824aa717ceab551fEXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-classification-job/e758dd56b824aa717ceab551fEXAMPLE",
    "JobStatus": "SUBMITTED"
}
```

Conteúdo de `predictions.jsonl`:

```
{"File": "SampleSMSText1.txt", "Line": "0", "Classes": [{"Name": "spam", "Score": 0.9999}, {"Name": "ham", "Score": 0.0001}]}
{"File": "SampleSMStext2.txt", "Line": "0", "Classes": [{"Name": "ham", "Score": 0.9994}, {"Name": "spam", "Score": 0.0006}]}
{"File": "SampleSMSText3.txt", "Line": "0", "Classes": [{"Name": "spam", "Score": 0.9999}, {"Name": "ham", "Score": 0.0001}]}
```

Para obter mais informações, consulte Classificação personalizada no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte StartDocumentClassificationJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you
 can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
```

```
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a StartDocumentClassificationJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->startdocclassificationjob(
      iv_jobname = iv_job_name
      iv_documentclassifierarn = iv_classifier_arn
      io_inputdataconfig = NEW /aws1/cl_cpdinputdataconfig(
        iv_s3uri = iv_input_s3_uri
        iv_inputformat = iv_input_format
      )
      io_outputdataconfig = NEW /aws1/cl_cpdoutputdataconfig(
        iv_s3uri = iv_output_s3_uri
      )
      iv_dataaccessrolearn = iv_data_access_role_arn
    ).
    MESSAGE 'Document classification job started.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdresourcenotfoundex.
    MESSAGE 'Resource not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdresourceunavailex.
    MESSAGE 'Resource unavailable.' TYPE 'E'.
  CATCH /aws1/cx_cpdkmskeyvalidationex.
    MESSAGE 'KMS key validation error.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanytagsex.
    MESSAGE 'Too many tags.' TYPE 'E'.
  CATCH /aws1/cx_cpdresrclimitexcdex.
    MESSAGE 'Resource limit exceeded.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a StartDocumentClassificationJobreferência da API AWS SDK for SAP ABAP.

Use **`StartTopicsDetectionJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartTopicsDetectionJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Execute uma tarefa de modelagem de tópicos em dados de amostra

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// This example scans the documents in an Amazon Simple Storage Service
/// (Amazon S3) bucket and analyzes it for topics. The results are stored
/// in another bucket and then the resulting job properties are displayed
/// on the screen. This example was created using the AWS SDK for .NEt
/// version 3.7 and .NET Core version 5.0.
/// </summary>
public static class TopicModeling
{
    /// <summary>
    /// This methos calls a topic detection job by calling the Amazon
    /// Comprehend StartTopicsDetectionJobRequest.
    /// </summary>
    public static async Task Main()
```

```
        {
            var comprehendClient = new AmazonComprehendClient();

            string inputS3Uri = "s3://input bucket/input path";
            InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
            string outputS3Uri = "s3://output bucket/output path";
            string dataAccessRoleArn = "arn:aws:iam::account ID:role/data access
role";
            int numberOfTopics = 10;

            var startTopicsDetectionJobRequest = new
StartTopicsDetectionJobRequest()
            {
                InputDataConfig = new InputDataConfig()
                {
                    S3Uri = inputS3Uri,
                    InputFormat = inputDocFormat,
                },
                OutputDataConfig = new OutputDataConfig()
                {
                    S3Uri = outputS3Uri,
                },
                DataAccessRoleArn = dataAccessRoleArn,
                NumberOfTopics = numberOfTopics,
            };

            var startTopicsDetectionJobResponse = await
comprehendClient.StartTopicsDetectionJobAsync(startTopicsDetectionJobRequest);

            var jobId = startTopicsDetectionJobResponse.JobId;
            Console.WriteLine("JobId: " + jobId);

            var describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
            {
                JobId = jobId,
            };

            var describeTopicsDetectionJobResponse = await
comprehendClient.DescribeTopicsDetectionJobAsync(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);
```

```
            var listTopicsDetectionJobsResponse = await
comprehendClient.ListTopicsDetectionJobsAsync(new
ListTopicsDetectionJobsRequest());
            foreach (var props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
            {
                PrintJobProperties(props);
            }
        }

        /// <summary>
        /// This method is a helper method that displays the job properties
        /// from the call to StartTopicsDetectionJobRequest.
        /// </summary>
        /// <param name="props">A list of properties from the call to
        /// StartTopicsDetectionJobRequest.</param>
        private static void PrintJobProperties(TopicsDetectionJobProperties
props)
        {
            Console.WriteLine($"JobId: {props.JobId}, JobName: {props.JobName},
JobStatus: {props.JobStatus}");
            Console.WriteLine($"NumberOfTopics:
{props.NumberOfTopics}\nInputS3Uri: {props.InputDataConfig.S3Uri}");
            Console.WriteLine($"InputFormat: {props.InputDataConfig.InputFormat},
OutputS3Uri: {props.OutputDataConfig.S3Uri}");
        }
    }
```

- Para obter detalhes da API, consulte StartTopicsDetectionJoba Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para iniciar um trabalho de análise de detecção de tópicos

O exemplo de `start-topics-detection-job` a seguir inicia um trabalho assíncrono de detecção de tópicos para todos os arquivos localizados no endereço especificado pela tag `--input-data-config`. Quando o trabalho é concluído, a pasta, `output`, é colocada no

local especificado pela tag `--ouput-data-config`. `output` contém topic-terms.csv e doc-topics.csv. O primeiro arquivo de saída, topic-terms.csv, é uma lista de tópicos na coleção. Para cada tópico, a lista inclui por padrão os principais termos por tópico de acordo com seu peso. O segundo arquivo, `doc-topics.csv`, lista os documentos associados a um tópico e a proporção do documento relacionada ao tópico.

```
aws comprehend start-topics-detection-job \
    --job-name example_topics_detection_job \
    --language-code en \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket/" \
    --output-data-config "S3Uri=s3://amzn-s3-demo-destination-bucket/testfolder/"
 \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
    --language-code en
```

Saída:

```
{
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
    "JobStatus": "SUBMITTED"
}
```

Para obter mais informações, consulte Modelagem de tópicos no Guia do desenvolvedor do Amazon Comprehend.

- Para obter detalhes da API, consulte StartTopicsDetectionJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
 stored
        in a tar archive compressed in gzip format. The job runs asynchronously,
 so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                              bucket. If multiple objects have the same prefix,
 all
                              of them are used.
        :param input_format: The format of the input data, either one document
 per
                              file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
 that
```

```
                                    grants Comprehend permission to read from
the
                                    input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a StartTopicsDetectionJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->starttopicsdetectionjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_cpdinputdataconfig(
```

```
            iv_s3uri = iv_input_s3_uri
            iv_inputformat = iv_input_format
          )
          io_outputdataconfig = NEW /aws1/cl_cpdoutputdataconfig(
            iv_s3uri = iv_output_s3_uri
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
        ).
        MESSAGE 'Topics detection job started.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdkmskeyvalidationex.
        MESSAGE 'KMS key validation error.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanytagsex.
        MESSAGE 'Too many tags.' TYPE 'E'.
      CATCH /aws1/cx_cpdresrclimitexcdex.
        MESSAGE 'Resource limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartTopicsDetectionJobreferência da API AWS SDK for SAP ABAP.

Cenários para o Amazon Comprehend usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon AWS Comprehend com SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon Comprehend ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação de transmissão do Amazon Transcribe
- Crie um chatbot do Amazon Lex para engajar os visitantes do site

- Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS
- Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
- Detecte elementos do documento com o Amazon Comprehend e um AWS SDK
- Detecte entidades no texto extraído de uma imagem usando um AWS SDK
- Execute um trabalho de modelagem de tópicos do Amazon Comprehend em dados de amostra usando um AWS SDK
- Treine um classificador Amazon Comprehend personalizado e classifique documentos usando um AWS SDK

Criar uma aplicação de transmissão do Amazon Transcribe

O exemplo de código a seguir mostra como construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Transcribe para construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Crie um chatbot do Amazon Lex para engajar os visitantes do site

Os exemplos de código a seguir mostram como criar um chatbot para engajar os visitantes do seu site.

Java

SDK para Java 2.x

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o exemplo completo Criando um chatbot Amazon Lex no guia do AWS SDK para JavaScript desenvolvedor.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS

Os exemplos de código a seguir mostram como criar uma aplicação de mensagens usando o Amazon SQS.

Java

SDK para Java 2.x

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Kotlin

SDK para Kotlin

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio

Os exemplos de código a seguir mostram como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

.NET

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O

hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado nas funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
```

```
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
```

```
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);
```

```
  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Detecte elementos do documento com o Amazon Comprehend e um AWS SDK

O exemplo de código a seguir mostra como:

- Detecte idiomas, entidades e frases-chave em um documento.
- Detecte informações de identificação pessoal (PII) em um documento.
- Detecte o sentimento de um documento.
- Detectar elementos sintáticos em um documento.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as ações do Amazon Comprehend.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
```

```
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and
its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
```

```
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can
be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
```

```
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

Chame funções na classe wrapper para detectar entidades, frases e muito mais em um documento.

```
def usage_demo():
    print("-" * 88)
```

```
    print("Welcome to the Amazon Comprehend detection demo!")
    print("-" * 88)

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

    comp_detect = ComprehendDetect(boto3.client("comprehend"))
    with open("detect_sample.txt") as sample_file:
        sample_text = sample_file.read()

    demo_size = 3

    print("Sample text used for this demo:")
    print("-" * 88)
    print(sample_text)
    print("-" * 88)

    print("Detecting languages.")
    languages = comp_detect.detect_languages(sample_text)
    pprint(languages)
    lang_code = languages[0]["LanguageCode"]

    print("Detecting entities.")
    entities = comp_detect.detect_entities(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(entities[:demo_size])

    print("Detecting key phrases.")
    phrases = comp_detect.detect_key_phrases(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(phrases[:demo_size])

    print("Detecting personally identifiable information (PII).")
    pii_entities = comp_detect.detect_pii(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(pii_entities[:demo_size])

    print("Detecting sentiment.")
    sentiment = comp_detect.detect_sentiment(sample_text, lang_code)
    print(f"Sentiment: {sentiment['Sentiment']}")
    print("SentimentScore:")
    pprint(sentiment["SentimentScore"])

    print("Detecting syntax elements.")
    syntax_tokens = comp_detect.detect_syntax(sample_text, lang_code)
```

```
    print(f"The first {demo_size} are:")
    pprint(syntax_tokens[:demo_size])

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - DetectDominantLanguage
  - DetectEntities
  - DetectKeyPhrases
  - DetectPiiEntities
  - DetectSentiment
  - DetectSyntax

Detecte entidades no texto extraído de uma imagem usando um AWS SDK

O exemplo de código a seguir mostra como usar o Amazon Comprehend para detectar entidades em texto extraído pelo Amazon Textract de uma imagem armazenada no Amazon S3.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) em um notebook Jupyter para detectar entidades no texto extraído de uma imagem. Este exemplo usa o Amazon Textract para extrair texto de uma imagem armazenada no Amazon Simple Storage Service (Amazon S3) e no Amazon Comprehend para detectar entidades no texto extraído.

Este exemplo é um caderno Jupyter e deve ser executado em um ambiente que possa hospedar blocos de anotações. Para obter instruções sobre como executar o exemplo usando o Amazon SageMaker AI, consulte as instruções em TextractAndComprehendNotebook.ipynb.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Execute um trabalho de modelagem de tópicos do Amazon Comprehend em dados de amostra usando um AWS SDK

O exemplo de código a seguir mostra como:

- Execute uma tarefa de modelagem de tópicos do Amazon Comprehend em dados de amostra.
- Informações sobre a tarefa.
- Extrair dados de saída do trabalho no Amazon S3.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe wrapper para chamar as ações de modelagem de tópicos do Amazon Comprehend.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
```

```
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
stored
        in a tar archive compressed in gzip format. The job runs asynchronously,
so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                            bucket. If multiple objects have the same prefix,
all
                            of them are used.
        :param input_format: The format of the input data, either one document
per
                            file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                    grants Comprehend permission to read from
the
                                    input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
```

```
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
```

```
            return jobs
```

Use a classe wrapper para executar uma tarefa de modelagem de tópicos e obter dados dela.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend topic modeling demo!")
    print("-" * 88)

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

    input_prefix = "input/"
    output_prefix = "output/"
    demo_resources = ComprehendDemoResources(
        boto3.resource("s3"), boto3.resource("iam")
    )
    topic_modeler = ComprehendTopicModeler(boto3.client("comprehend"))

    print("Setting up storage and security resources needed for the demo.")
    demo_resources.setup("comprehend-topic-modeler-demo")
    print("Copying sample data from public bucket into input bucket.")
    demo_resources.bucket.copy(
        {"Bucket": "public-sample-us-west-2", "Key": "TopicModeling/Sample.txt"},
        f"{input_prefix}sample.txt",
    )

    print("Starting topic modeling job on sample data.")
    job_info = topic_modeler.start_job(
        "demo-topic-modeling-job",
        demo_resources.bucket.name,
        input_prefix,
        JobInputFormat.per_line,
        demo_resources.bucket.name,
        output_prefix,
        demo_resources.data_access_role.arn,
    )

    print(
        f"Waiting for job {job_info['JobId']} to complete. This typically takes "
        f"20 - 30 minutes."
```

```
    )
    job_waiter = JobCompleteWaiter(topic_modeler.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = topic_modeler.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from the output Amazon S3 bucket: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = demo_resources.extract_job_output(job)
    lines = 10
    print(f"First {lines} lines of document topics output:")
    pprint(job_output["doc-topics.csv"]["data"][:lines])
    print(f"First {lines} lines of terms output:")
    pprint(job_output["topic-terms.csv"]["data"][:lines])

    print("Cleaning up resources created for the demo.")
    demo_resources.cleanup()

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - DescribeTopicsDetectionJob
  - ListTopicsDetectionJobs
  - StartTopicsDetectionJob

Treine um classificador Amazon Comprehend personalizado e classifique documentos usando um AWS SDK

O exemplo de código a seguir mostra como:

- Crie um classificador de vários rótulos do Amazon Comprehend.

- Treine o classificador em dados de amostra.
- Execute uma tarefa de classificação em um segundo conjunto de dados.
- Extrair dados de saída da tarefa do Amazon S3.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe wrapper para chamar as ações do classificador de documentos Amazon Comprehend.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it
 immediately
```

```
        starts training on the data found in the specified Amazon S3 bucket.
 Training
        can take 30 minutes or longer. The `describe_document_classifier`
 function
        can be used to get training status and returns a status of TRAINED when
 the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
 data.
        :param training_key: The prefix used to find training data in the
 training
                             bucket. If multiple objects have the same prefix,
 all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
 that
                                     grants Comprehend permission to read from
 the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/
{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
 self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn


    def describe(self, classifier_arn=None):
```

```
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers


    def delete(self):
        """
        Deletes the classifier.
        """
```

```
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.",
self.classifier_arn)
            raise


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you
can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
```

```
                                    grants Comprehend permission to read from
the
                                    input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response =
self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
```

```
        else:
            return job


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

Crie uma classe para ajudar na execução do cenário.

```
class ClassifierDemo:
    """
    Encapsulates functions used to run the demonstration.
    """

    def __init__(self, demo_resources):
        """
        :param demo_resources: A ComprehendDemoResources class that manages
 resources
                               for the demonstration.
        """
        self.demo_resources = demo_resources
        self.training_prefix = "training/"
        self.input_prefix = "input/"
        self.input_format = JobInputFormat.per_line
        self.output_prefix = "output/"
```

```
    def setup(self):
        """Creates AWS resources used by the demo."""
        self.demo_resources.setup("comprehend-classifier-demo")

    def cleanup(self):
        """Deletes AWS resources used by the demo."""
        self.demo_resources.cleanup()

    @staticmethod
    def _sanitize_text(text):
        """Removes characters that cause errors for the document parser."""
        return text.replace("\r", " ").replace("\n", " ").replace(",", ";")

    @staticmethod
    def _get_issues(query, issue_count):
        """
        Gets issues from GitHub using the specified query parameters.

        :param query: The query string used to request issues from the GitHub
 API.
        :param issue_count: The number of issues to retrieve.
        :return: The list of issues retrieved from GitHub.
        """
        issues = []
        logger.info("Requesting issues from %s?%s.", GITHUB_SEARCH_URL, query)
        response = requests.get(f"{GITHUB_SEARCH_URL}?
{query}&per_page={issue_count}")
        if response.status_code == 200:
            issue_page = response.json()["items"]
            logger.info("Got %s issues.", len(issue_page))
            issues = [
                {
                    "title": ClassifierDemo._sanitize_text(issue["title"]),
                    "body": ClassifierDemo._sanitize_text(issue["body"]),
                    "labels": {label["name"] for label in issue["labels"]},
                }
                for issue in issue_page
            ]
        else:
            logger.error(
                "GitHub returned error code %s with message %s.",
                response.status_code,
                response.json(),
```

```
            )
        logger.info("Found %s issues.", len(issues))
        return issues

    def get_training_issues(self, training_labels):
        """
        Gets issues used for training the custom classifier. Training issues are
        closed issues from the Boto3 repo that have known labels. Comprehend
        requires a minimum of ten training issues per label.

        :param training_labels: The issue labels to use for training.
        :return: The set of issues used for training.
        """
        issues = []
        per_label_count = 15
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:closed+label:{label}",
                per_label_count,
            )
            for issue in issues:
                issue["labels"] = issue["labels"].intersection(training_labels)
        return issues

    def get_input_issues(self, training_labels):
        """
        Gets input issues from GitHub. For demonstration purposes, input issues
        are open issues from the Boto3 repo with known labels, though in practice
        any issue could be submitted to the classifier for labeling.

        :param training_labels: The set of labels to query for.
        :return: The set of issues used for input.
        """
        issues = []
        per_label_count = 5
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:open+label:{label}",
                per_label_count,
            )
        return issues

    def upload_issue_data(self, issues, training=False):
        """
```

```
        Uploads issue data to an Amazon S3 bucket, either for training or for
input.
        The data is first put into the format expected by Comprehend. For
training,
        the set of pipe-delimited labels is prepended to each document. For
        input, labels are not sent.

        :param issues: The set of issues to upload to Amazon S3.
        :param training: Indicates whether the issue data is used for training or
                         input.
        """
        try:
            obj_key = (
                self.training_prefix if training else self.input_prefix
            ) + "issues.txt"
            if training:
                issue_strings = [
                    f"{'|'.join(issue['labels'])},{issue['title']}
{issue['body']}"
                    for issue in issues
                ]
            else:
                issue_strings = [
                    f"{issue['title']} {issue['body']}" for issue in issues
                ]
            issue_bytes = BytesIO("\n".join(issue_strings).encode("utf-8"))
            self.demo_resources.bucket.upload_fileobj(issue_bytes, obj_key)
            logger.info(
                "Uploaded data as %s to bucket %s.",
                obj_key,
                self.demo_resources.bucket.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload data to bucket %s.",
self.demo_resources.bucket.name
            )
            raise

    def extract_job_output(self, job):
        """Extracts job output from Amazon S3."""
        return self.demo_resources.extract_job_output(job)

    @staticmethod
```

```
    def reconcile_job_output(input_issues, output_dict):
        """
        Reconciles job output with the list of input issues. Because the input
 issues
        have known labels, these can be compared with the labels added by the
        classifier to judge the accuracy of the output.

        :param input_issues: The list of issues used as input.
        :param output_dict: The dictionary of data that is output by the
 classifier.
        :return: The list of reconciled input and output data.
        """
        reconciled = []
        for archive in output_dict.values():
            for line in archive["data"]:
                in_line = int(line["Line"])
                in_labels = input_issues[in_line]["labels"]
                out_labels = {
                    label["Name"]
                    for label in line["Labels"]
                    if float(label["Score"]) > 0.3
                }
                reconciled.append(
                    f"{line['File']}, line {in_line} has labels {in_labels}.\n"
                    f"\tClassifier assigned {out_labels}."
                )
        logger.info("Reconciled input and output labels.")
        return reconciled
```

Treine um classificador em um conjunto de GitHub problemas com rótulos conhecidos e, em seguida, envie um segundo conjunto de GitHub problemas ao classificador para que eles possam ser rotulados.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend custom document classifier demo!")
    print("-" * 88)

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

```
    comp_demo = ClassifierDemo(
        ComprehendDemoResources(boto3.resource("s3"), boto3.resource("iam"))
    )
    comp_classifier = ComprehendClassifier(boto3.client("comprehend"))
    classifier_trained_waiter = ClassifierTrainedWaiter(
        comp_classifier.comprehend_client
    )
    training_labels = {"bug", "feature-request", "dynamodb", "s3"}

    print("Setting up storage and security resources needed for the demo.")
    comp_demo.setup()

    print("Getting training data from GitHub and uploading it to Amazon S3.")
    training_issues = comp_demo.get_training_issues(training_labels)
    comp_demo.upload_issue_data(training_issues, True)

    classifier_name = "doc-example-classifier"
    print(f"Creating document classifier {classifier_name}.")
    comp_classifier.create(
        classifier_name,
        "en",
        comp_demo.demo_resources.bucket.name,
        comp_demo.training_prefix,
        comp_demo.demo_resources.data_access_role.arn,
        ClassifierMode.multi_label,
    )
    print(
        f"Waiting until {classifier_name} is trained. This typically takes "
        f"30-40 minutes."
    )
    classifier_trained_waiter.wait(comp_classifier.classifier_arn)

    print(f"Classifier {classifier_name} is trained:")
    pprint(comp_classifier.describe())

    print("Getting input data from GitHub and uploading it to Amazon S3.")
    input_issues = comp_demo.get_input_issues(training_labels)
    comp_demo.upload_issue_data(input_issues)

    print("Starting classification job on input data.")
    job_info = comp_classifier.start_job(
        "issue_classification_job",
        comp_demo.demo_resources.bucket.name,
        comp_demo.input_prefix,
```

```
        comp_demo.input_format,
        comp_demo.demo_resources.bucket.name,
        comp_demo.output_prefix,
        comp_demo.demo_resources.data_access_role.arn,
    )
    print(f"Waiting for job {job_info['JobId']} to complete.")
    job_waiter = JobCompleteWaiter(comp_classifier.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = comp_classifier.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from Amazon S3: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = comp_demo.extract_job_output(job)
    print("Job output:")
    pprint(job_output)

    print("Reconciling job output with labels from GitHub:")
    reconciled_output = comp_demo.reconcile_job_output(input_issues, job_output)
    print(*reconciled_output, sep="\n")

    answer = input(f"Do you want to delete the classifier {classifier_name} (y/
n)? ")
    if answer.lower() == "y":
        print(f"Deleting {classifier_name}.")
        comp_classifier.delete()

    print("Cleaning up resources created for the demo.")
    comp_demo.cleanup()

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

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateDocumentClassifier

- DeleteDocumentClassifier
- DescribeDocumentClassificationJob
- DescribeDocumentClassifier
- ListDocumentClassificationJobs
- ListDocumentClassifiers
- StartDocumentClassificationJob

Exemplos de código para AWS Config usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Config com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS Config Guia do desenvolvedor — Mais informações sobre AWS Config.
- AWS Config Referência da API — Detalhes sobre todas as AWS Config ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Config usar AWS SDKs
  - Ações para AWS Config usar AWS SDKs
    - Use DeleteConfigRulecom um AWS SDK ou CLI
    - Usar DescribeComplianceByConfigRule com uma CLI
    - Usar DescribeComplianceByResource com uma CLI

- Usar DescribeConfigRuleEvaluationStatus com uma CLI
- Use DescribeConfigRulescom um AWS SDK ou CLI
- Usar DescribeConfigurationRecorderStatus com uma CLI
- Usar DescribeConfigurationRecorders com uma CLI
- Usar DescribeDeliveryChannels com uma CLI
- Usar GetComplianceDetailsByConfigRule com uma CLI
- Usar GetComplianceDetailsByResource com uma CLI
- Usar GetComplianceSummaryByConfigRule com uma CLI
- Usar GetComplianceSummaryByResourceType com uma CLI
- Use PutConfigRulecom um AWS SDK ou CLI
- Usar PutDeliveryChannel com uma CLI

- Cenários para AWS Config usar AWS SDKs
  - Conceitos básicos do gerenciamento de configurações

Exemplos básicos para AWS Config usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Config com AWS SDKs.

Exemplos

- Ações para AWS Config usar AWS SDKs
  - Use DeleteConfigRulecom um AWS SDK ou CLI
  - Usar DescribeComplianceByConfigRule com uma CLI
  - Usar DescribeComplianceByResource com uma CLI
  - Usar DescribeConfigRuleEvaluationStatus com uma CLI
  - Use DescribeConfigRulescom um AWS SDK ou CLI
  - Usar DescribeConfigurationRecorderStatus com uma CLI
  - Usar DescribeConfigurationRecorders com uma CLI
  - Usar DescribeDeliveryChannels com uma CLI
  - Usar GetComplianceDetailsByConfigRule com uma CLI
  - Usar GetComplianceDetailsByResource com uma CLI
  - Usar GetComplianceSummaryByConfigRule com uma CLI

- Usar GetComplianceSummaryByResourceType com uma CLI
- Use PutConfigRulecom um AWS SDK ou CLI
- Usar PutDeliveryChannel com uma CLI

Ações para AWS Config usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Config ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a AWS Config API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para AWS Config usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Config.

Exemplos

- Use DeleteConfigRulecom um AWS SDK ou CLI
- Usar DescribeComplianceByConfigRule com uma CLI
- Usar DescribeComplianceByResource com uma CLI
- Usar DescribeConfigRuleEvaluationStatus com uma CLI
- Use DescribeConfigRulescom um AWS SDK ou CLI
- Usar DescribeConfigurationRecorderStatus com uma CLI
- Usar DescribeConfigurationRecorders com uma CLI
- Usar DescribeDeliveryChannels com uma CLI
- Usar GetComplianceDetailsByConfigRule com uma CLI
- Usar GetComplianceDetailsByResource com uma CLI
- Usar GetComplianceSummaryByConfigRule com uma CLI
- Usar GetComplianceSummaryByResourceType com uma CLI
- Use PutConfigRulecom um AWS SDK ou CLI
- Usar PutDeliveryChannel com uma CLI

Use **`DeleteConfigRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteConfigRule`.

CLI

AWS CLI

Para excluir uma regra de AWS Config

O comando a seguir exclui uma regra de AWS Config chamada: `MyConfigRule`

```
aws configservice delete-config-rule --config-rule-name MyConfigRule
```

- Para obter detalhes da API, consulte DeleteConfigRuleem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def delete_config_rule(self, rule_name):
```

```
        """
        Delete the specified rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.config_client.delete_config_rule(ConfigRuleName=rule_name)
            logger.info("Deleted rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't delete rule %s.", rule_name)
            raise
```

- Para obter detalhes da API, consulte a DeleteConfigRuleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
lo_cfs->deleteconfigrule( iv_rule_name ).
MESSAGE 'Deleted AWS Config rule.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a DeleteConfigRulereferência da API AWS SDK for SAP ABAP.

Usar **`DescribeComplianceByConfigRule`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeComplianceByConfigRule`.

CLI

AWS CLI

Para obter informações de conformidade para suas regras de AWS Config

O comando a seguir retorna informações de conformidade para cada regra AWS do Config violada por um ou mais recursos: AWS

```
aws configservice describe-compliance-by-config-rule --compliance-
types NON_COMPLIANT
```

Na saída, o valor de cada atributo `CappedCount` indica quantos recursos não estão em conformidade com a regra relacionada. Por exemplo, a saída a seguir indica que três recursos não estão em conformidade com a regra chamada `InstanceTypesAreT2micro`.

Saída:

```
{
    "ComplianceByConfigRules": [
        {
            "Compliance": {
                "ComplianceContributorCount": {
                    "CappedCount": 3,
                    "CapExceeded": false
                },
                "ComplianceType": "NON_COMPLIANT"
            },
            "ConfigRuleName": "InstanceTypesAreT2micro"
        },
        {
            "Compliance": {
                "ComplianceContributorCount": {
                    "CappedCount": 10,
                    "CapExceeded": false
                },
                "ComplianceType": "NON_COMPLIANT"
            },
            "ConfigRuleName": "RequiredTagsForVolumes"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeComplianceByConfigRuleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera detalhes de conformidade da regra ebs-optimized-instance, para a qual não há resultados de avaliação atuais para a regra e, portanto, retorna INSUFFICIENT_DATA

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Saída:

```
ComplianceContributorCount ComplianceType
-------------------------- --------------
                           INSUFFICIENT_DATA
```

Exemplo 2: este exemplo retorna o número de recursos que não estão em conformidade com a regra ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName
 ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -ComplianceType
 NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Saída:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- Para obter detalhes da API, consulte DescribeComplianceByConfigRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera detalhes de conformidade da regra ebs-optimized-instance, para a qual não há resultados de avaliação atuais para a regra e, portanto, retorna INSUFFICIENT_DATA

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Saída:

```
ComplianceContributorCount ComplianceType
-------------------------- --------------
                           INSUFFICIENT_DATA
```

Exemplo 2: este exemplo retorna o número de recursos que não estão em conformidade com a regra ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName
 ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -ComplianceType
 NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Saída:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- Para obter detalhes da API, consulte DescribeComplianceByConfigRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeComplianceByResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeComplianceByResource`.

CLI

AWS CLI

Para obter informações de conformidade para seus AWS recursos

O comando a seguir retorna informações de conformidade para cada instância do EC2 que é registrada pelo AWS Config e que viola uma ou mais regras:

```
aws configservice describe-compliance-by-resource --resource-
type AWS::EC2::Instance --compliance-types NON_COMPLIANT
```

Na saída, o valor de cada atributo `CappedCount` indica quantas regras o recurso viola. Por exemplo, a saída a seguir indica que a instância `i-1a2b3c4d` viola duas regras.

Saída:

```
{
    "ComplianceByResources": [
        {
            "ResourceType": "AWS::EC2::Instance",
            "ResourceId": "i-1a2b3c4d",
            "Compliance": {
                "ComplianceContributorCount": {
                    "CappedCount": 2,
                    "CapExceeded": false
                },
                "ComplianceType": "NON_COMPLIANT"
            }
        },
        {
            "ResourceType": "AWS::EC2::Instance",
            "ResourceId": "i-2a2b3c4d ",
            "Compliance": {
                "ComplianceContributorCount": {
                    "CappedCount": 3,
                    "CapExceeded": false
                },
                "ComplianceType": "NON_COMPLIANT"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeComplianceByResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo verifica o tipo de recurso **`AWS::SSM::ManagedInstanceInventory`** quanto ao tipo de conformidade “COMPLIANT”.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Saída:

```
Compliance                            ResourceId          ResourceType
----------                            ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- Para obter detalhes da API, consulte DescribeComplianceByResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo verifica o tipo de recurso **`AWS::SSM::ManagedInstanceInventory`** quanto ao tipo de conformidade “COMPLIANT”.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Saída:

```
Compliance                            ResourceId          ResourceType
----------                            ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- Para obter detalhes da API, consulte DescribeComplianceByResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeConfigRuleEvaluationStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeConfigRuleEvaluationStatus`.

CLI

AWS CLI

Para obter informações de status de uma regra AWS Config

O comando a seguir retorna as informações de status de uma regra AWS Config chamada: `MyConfigRule`

```
aws configservice describe-config-rule-evaluation-status --config-rule-
names MyConfigRule
```

Saída:

```
{
    "ConfigRulesEvaluationStatus": [
        {
            "ConfigRuleArn": "arn:aws:config:us-east-1:123456789012:config-rule/
config-rule-abcdef",
            "FirstActivatedTime": 1450311703.844,
            "ConfigRuleId": "config-rule-abcdef",
            "LastSuccessfulInvocationTime": 1450314643.156,
            "ConfigRuleName": "MyConfigRule"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeConfigRuleEvaluationStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna as informações de status das regras de configuração fornecidas.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Saída:

```
ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk
ConfigRuleId                 : config-rule-kvq1wk
ConfigRuleName               : root-account-mfa-enabled
FirstActivatedTime           : 8/27/2019 8:05:17 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime : 12/13/2019 8:12:03 AM

ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
ConfigRuleId                 : config-rule-z1s23b
ConfigRuleName               : vpc-flow-logs-enabled
FirstActivatedTime           : 8/14/2019 6:23:44 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime : 12/13/2019 7:12:01 AM
```

- Para obter detalhes da API, consulte DescribeConfigRuleEvaluationStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna as informações de status das regras de configuração fornecidas.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Saída:

```
ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk
ConfigRuleId                 : config-rule-kvq1wk
ConfigRuleName               : root-account-mfa-enabled
```

```
FirstActivatedTime           : 8/27/2019 8:05:17 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime : 12/13/2019 8:12:03 AM

ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
ConfigRuleId                 : config-rule-z1s23b
ConfigRuleName               : vpc-flow-logs-enabled
FirstActivatedTime           : 8/14/2019 6:23:44 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime : 12/13/2019 7:12:01 AM
```

- Para obter detalhes da API, consulte DescribeConfigRuleEvaluationStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeConfigRules`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeConfigRules`.

CLI

AWS CLI

Para obter detalhes de uma regra de AWS Config

O comando a seguir retorna detalhes de uma regra de AWS Config chamada: `InstanceTypesAreT2micro`

```
aws configservice describe-config-rules --config-rule-
names InstanceTypesAreT2micro
```

Saída:

```
{
    "ConfigRules": [
        {
            "ConfigRuleState": "ACTIVE",
            "Description": "Evaluates whether EC2 instances are the t2.micro
 type.",
            "ConfigRuleName": "InstanceTypesAreT2micro",
            "ConfigRuleArn": "arn:aws:config:us-east-1:123456789012:config-rule/
config-rule-abcdef",
            "Source": {
                "Owner": "CUSTOM_LAMBDA",
                "SourceIdentifier": "arn:aws:lambda:us-
east-1:123456789012:function:InstanceTypeCheck",
                "SourceDetails": [
                    {
                        "EventSource": "aws.config",
                        "MessageType": "ConfigurationItemChangeNotification"
                    }
                ]
            },
            "InputParameters": "{\"desiredInstanceType\":\"t2.micro\"}",
            "Scope": {
                "ComplianceResourceTypes": [
                    "AWS::EC2::Instance"
                ]
            },
            "ConfigRuleId": "config-rule-abcdef"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeConfigRulesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo lista as regras de configuração da conta, com propriedades selecionadas.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Saída:

```
ConfigRuleName                                    ConfigRuleId
 ConfigRuleArn
 ConfigRuleState
--------------                                    ------------
 -------------
 ---------------
ALB_REDIRECTION_CHECK                             config-rule-12iyn3
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-12iyn3
 ACTIVE
access-keys-rotated                               config-rule-aospfr
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-aospfr
 ACTIVE
autoscaling-group-elb-healthcheck-required        config-rule-cn1f2x
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x
 ACTIVE
```

- Para obter detalhes da API, consulte DescribeConfigRulesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo lista as regras de configuração da conta, com propriedades selecionadas.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Saída:

```
ConfigRuleName                                    ConfigRuleId
 ConfigRuleArn
 ConfigRuleState
--------------                                    ------------
 -------------
 ---------------
```

```
ALB_REDIRECTION_CHECK                              config-rule-12iyn3
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-12iyn3
 ACTIVE
access-keys-rotated                                config-rule-aospfr
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-aospfr
 ACTIVE
autoscaling-group-elb-healthcheck-required         config-rule-cn1f2x
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x
 ACTIVE
```

- Para obter detalhes da API, consulte DescribeConfigRulesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def describe_config_rule(self, rule_name):
        """
        Gets data for the specified rule.

        :param rule_name: The name of the rule to retrieve.
        :return: The rule data.
```

```
        """
        try:
            response = self.config_client.describe_config_rules(
                ConfigRuleNames=[rule_name]
            )
            rule = response["ConfigRules"]
            logger.info("Got data for rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't get data for rule %s.", rule_name)
            raise
        else:
            return rule
```

- Para obter detalhes da API, consulte a DescribeConfigRulesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA(lo_result) = lo_cfs->describeconfigrules(
      it_configrulenames = VALUE /aws1/
cl_cfsconfigrulenames_w=>tt_configrulenames(
        ( NEW /aws1/cl_cfsconfigrulenames_w( iv_rule_name ) )
      )
    ).
    ot_cfg_rules = lo_result->get_configrules( ).
    MESSAGE 'Retrieved AWS Config rule data.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a DescribeConfigRulesreferência da API AWS SDK for SAP ABAP.

Usar **`DescribeConfigurationRecorderStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeConfigurationRecorderStatus`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos do gerenciamento de configurações

CLI

AWS CLI

Para obter informações de status do gravador de configuração

O seguinte comando retorna o status do gravador de configuração padrão:

```
aws configservice describe-configuration-recorder-status
```

Saída:

```
{
    "ConfigurationRecordersStatus": [
        {
            "name": "default",
            "lastStatus": "SUCCESS",
            "recording": true,
            "lastStatusChangeTime": 1452193834.344,
            "lastStartTime": 1441039997.819,
            "lastStopTime": 1441039992.835
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeConfigurationRecorderStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna o status dos gravadores de configuração.

```
Get-CFGConfigurationRecorderStatus
```

Saída:

```
LastErrorCode        :
LastErrorMessage     :
LastStartTime        : 10/11/2019 10:13:51 AM
LastStatus           : Success
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime         : 10/11/2019 10:13:46 AM
Name                 : default
Recording            : True
```

- Para obter detalhes da API, consulte DescribeConfigurationRecorderStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o status dos gravadores de configuração.

```
Get-CFGConfigurationRecorderStatus
```

Saída:

```
LastErrorCode        :
LastErrorMessage     :
LastStartTime        : 10/11/2019 10:13:51 AM
LastStatus           : Success
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime         : 10/11/2019 10:13:46 AM
Name                 : default
Recording            : True
```

- Para obter detalhes da API, consulte DescribeConfigurationRecorderStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeConfigurationRecorders`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeConfigurationRecorders`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos do gerenciamento de configurações

CLI

AWS CLI

Para obter detalhes sobre o gravador de configuração

O seguinte comando retorna detalhes sobre o gravador de configuração padrão:

```
aws configservice describe-configuration-recorders
```

Saída:

```
{
    "ConfigurationRecorders": [
        {
            "recordingGroup": {
                "allSupported": true,
                "resourceTypes": [],
                "includeGlobalResourceTypes": true
            },
            "roleARN": "arn:aws:iam::123456789012:role/config-ConfigRole-
A1B2C3D4E5F6",
            "name": "default"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeConfigurationRecordersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna os detalhes dos gravadores de configuração.

```
Get-CFGConfigurationRecorder | Format-List
```

Saída:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Para obter detalhes da API, consulte DescribeConfigurationRecordersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna os detalhes dos gravadores de configuração.

```
Get-CFGConfigurationRecorder | Format-List
```

Saída:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Para obter detalhes da API, consulte DescribeConfigurationRecordersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeDeliveryChannels`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeDeliveryChannels`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos do gerenciamento de configurações

CLI

AWS CLI

Para obter detalhes sobre o canal de entrega

O seguinte comando retorna detalhes sobre o canal de entrega:

```
aws configservice describe-delivery-channels
```

Saída:

```
{
    "DeliveryChannels": [
        {
            "snsTopicARN": "arn:aws:sns:us-east-1:123456789012:config-topic",
            "name": "default",
            "s3BucketName": "config-bucket-123456789012"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeDeliveryChannelsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera o canal de entrega da região e exibe detalhes.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Saída:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- Para obter detalhes da API, consulte DescribeDeliveryChannelsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera o canal de entrega da região e exibe detalhes.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Saída:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- Para obter detalhes da API, consulte DescribeDeliveryChannelsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetComplianceDetailsByConfigRule`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetComplianceDetailsByConfigRule`.

CLI

AWS CLI

Para obter os resultados da avaliação de uma regra AWS Config

O comando a seguir retorna os resultados da avaliação de todos os recursos que não estão em conformidade com uma regra de AWS Config chamada: `InstanceTypesAreT2micro`

```
aws configservice get-compliance-details-by-config-rule --config-rule-
name InstanceTypesAreT2micro --compliance-types NON_COMPLIANT
```

Saída:

```
{
    "EvaluationResults": [
        {
```

```
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-1a2b3c4d",
                    "ConfigRuleName": "InstanceTypesAreT2micro"
                }
            },
            "ResultRecordedTime": 1450314645.261,
            "ConfigRuleInvokedTime": 1450314642.948,
            "ComplianceType": "NON_COMPLIANT"
        },
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-2a2b3c4d",
                    "ConfigRuleName": "InstanceTypesAreT2micro"
                }
            },
            "ResultRecordedTime": 1450314645.18,
            "ConfigRuleInvokedTime": 1450314642.902,
            "ComplianceType": "NON_COMPLIANT"
        },
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-3a2b3c4d",
                    "ConfigRuleName": "InstanceTypesAreT2micro"
                }
            },
            "ResultRecordedTime": 1450314643.346,
            "ConfigRuleInvokedTime": 1450314643.124,
            "ComplianceType": "NON_COMPLIANT"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetComplianceDetailsByConfigRuleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo obtém os resultados da avaliação para a regra access-keys-rotated e retorna a saída agrupada por compliance-type

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Saída:

```
Count Name                      Group
----- ----                      -----
    2 COMPLIANT                 {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult}
    5 NON_COMPLIANT             {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationRes...
```

Exemplo 2: este exemplo consulta os detalhes de conformidade da regra access-keys-rotated para recursos COMPLIANT.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Saída:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Para obter detalhes da API, consulte GetComplianceDetailsByConfigRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo obtém os resultados da avaliação para a regra access-keys-rotated e retorna a saída agrupada por compliance-type

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Saída:

```
Count Name                      Group
----- ----                      -----
    2 COMPLIANT                 {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult}
    5 NON_COMPLIANT             {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationRes...
```

Exemplo 2: este exemplo consulta os detalhes de conformidade da regra access-keys-rotated para recursos COMPLIANT.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Saída:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Para obter detalhes da API, consulte GetComplianceDetailsByConfigRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetComplianceDetailsByResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetComplianceDetailsByResource`.

CLI

AWS CLI

Para obter os resultados da avaliação de um AWS recurso

O comando a seguir retorna os resultados da avaliação de cada regra com a qual a instância do EC2 `i-1a2b3c4d` não está em conformidade:

```
aws configservice get-compliance-details-by-resource --resource-
type AWS::EC2::Instance --resource-id i-1a2b3c4d --compliance-types NON_COMPLIANT
```

Saída:

```
{
    "EvaluationResults": [
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-1a2b3c4d",
                    "ConfigRuleName": "InstanceTypesAreT2micro"
                }
            },
            "ResultRecordedTime": 1450314643.288,
            "ConfigRuleInvokedTime": 1450314643.034,
            "ComplianceType": "NON_COMPLIANT"
        },
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-1a2b3c4d",
                    "ConfigRuleName": "RequiredTagForEC2Instances"
                }
            },
            "ResultRecordedTime": 1450314645.261,
            "ConfigRuleInvokedTime": 1450314642.948,
            "ComplianceType": "NON_COMPLIANT"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetComplianceDetailsByResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo avalia resultados para um determinado recurso.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -
ResourceType 'AWS::IAM::User'
```

Saída:

```
Annotation                 :
ComplianceType             : COMPLIANT
ConfigRuleInvokedTime      : 8/25/2019 11:34:56 PM
EvaluationResultIdentifier :
 Amazon.ConfigService.Model.EvaluationResultIdentifier
ResultRecordedTime         : 8/25/2019 11:34:56 PM
ResultToken                :
```

- Para obter detalhes da API, consulte GetComplianceDetailsByResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo avalia resultados para um determinado recurso.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -
ResourceType 'AWS::IAM::User'
```

Saída:

```
Annotation                 :
ComplianceType             : COMPLIANT
ConfigRuleInvokedTime      : 8/25/2019 11:34:56 PM
EvaluationResultIdentifier :
 Amazon.ConfigService.Model.EvaluationResultIdentifier
ResultRecordedTime         : 8/25/2019 11:34:56 PM
ResultToken                :
```

- Para obter detalhes da API, consulte GetComplianceDetailsByResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetComplianceSummaryByConfigRule`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetComplianceSummaryByConfigRule`.

CLI

AWS CLI

Para obter o resumo de conformidade de suas regras de AWS Config

O seguinte comando retorna o número de regras que estão em conformidade e o número das que não estão:

```
aws configservice get-compliance-summary-by-config-rule
```

Na saída, o valor de cada atributo `CappedCount` indica quantas regras estão em conformidade ou não.

Saída:

```
{
    "ComplianceSummary": {
        "NonCompliantResourceCount": {
            "CappedCount": 3,
            "CapExceeded": false
        },
        "ComplianceSummaryTimestamp": 1452204131.493,
        "CompliantResourceCount": {
            "CappedCount": 2,
            "CapExceeded": false
        }
    }
}
```

- Para obter detalhes da API, consulte GetComplianceSummaryByConfigRuleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna o número de regras do Config que não estão em conformidade.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Saída:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- Para obter detalhes da API, consulte GetComplianceSummaryByConfigRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o número de regras do Config que não estão em conformidade.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Saída:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- Para obter detalhes da API, consulte GetComplianceSummaryByConfigRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetComplianceSummaryByResourceType`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetComplianceSummaryByResourceType`.

CLI

AWS CLI

Para obter o resumo de conformidade para todos os tipos de recurso

O comando a seguir retorna o número de AWS recursos que não estão em conformidade e o número que estão em conformidade:

```
aws configservice get-compliance-summary-by-resource-type
```

Na saída, o valor de cada atributo `CappedCount` indica quantos recursos estão em conformidade ou não.

Saída:

```
{
    "ComplianceSummariesByResourceType": [
        {
            "ComplianceSummary": {
                "NonCompliantResourceCount": {
                    "CappedCount": 16,
                    "CapExceeded": false
                },
                "ComplianceSummaryTimestamp": 1453237464.543,
                "CompliantResourceCount": {
                    "CappedCount": 10,
                    "CapExceeded": false
                }
            }
        }
    ]
}
```

Para obter o resumo de conformidade para um tipo específico de recurso

O seguinte comando retorna o número de instâncias do EC2 que não estão em conformidade e o número das que estão:

```
aws configservice get-compliance-summary-by-resource-type --resource-
types AWS::EC2::Instance
```

Na saída, o valor de cada atributo `CappedCount` indica quantos recursos estão em conformidade ou não.

Saída:

```
{
    "ComplianceSummariesByResourceType": [
        {
            "ResourceType": "AWS::EC2::Instance",
            "ComplianceSummary": {
                "NonCompliantResourceCount": {
                    "CappedCount": 3,
                    "CapExceeded": false
                },
                "ComplianceSummaryTimestamp": 1452204923.518,
                "CompliantResourceCount": {
                    "CappedCount": 7,
                    "CapExceeded": false
                }
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte GetComplianceSummaryByResourceTypeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna o número de recursos que estão em conformidade ou não e converte a saída em json.

```
Get-CFGComplianceSummaryByResourceType -Select
 ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
```

```
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- Para obter detalhes da API, consulte GetComplianceSummaryByResourceTypeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o número de recursos que estão em conformidade ou não e converte a saída em json.

```
Get-CFGComplianceSummaryByResourceType -Select
 ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- Para obter detalhes da API, consulte GetComplianceSummaryByResourceTypeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutConfigRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutConfigRule`.

CLI

AWS CLI

Para adicionar uma regra de Config AWS gerenciada

O comando a seguir fornece código JSON para adicionar uma regra Config AWS gerenciada:

```
aws configservice put-config-rule --config-rule file://
RequiredTagsForEC2Instances.json
```

`RequiredTagsForEC2Instances.json` é um arquivo JSON que contém a configuração da regra:

```
{
  "ConfigRuleName": "RequiredTagsForEC2Instances",
  "Description": "Checks whether the CostCenter and Owner tags are applied to EC2
 instances.",
  "Scope": {
    "ComplianceResourceTypes": [
      "AWS::EC2::Instance"
    ]
  },
  "Source": {
    "Owner": "AWS",
    "SourceIdentifier": "REQUIRED_TAGS"
  },
  "InputParameters": "{\"tag1Key\":\"CostCenter\",\"tag2Key\":\"Owner\"}"
}
```

Para o `ComplianceResourceTypes` atributo, esse código JSON limita o escopo a recursos do `AWS::EC2::Instance` tipo, então o AWS Config avaliará somente instâncias do EC2 em relação à regra. Como a regra é uma regra gerenciada, o atributo `Owner` é definido como `AWS` e o atributo `SourceIdentifier` é definido como o identificador da regra, `REQUIRED_TAGS`. Para o atributo `InputParameters`, as chaves de tag exigidas pela regra, `CostCenter` e `Owner` são especificados.

Se o comando for bem-sucedido, o AWS Config não retornará nenhuma saída. Para verificar a configuração da regra, execute o comando describe-config-rules e especifique o nome da regra.

Como adicionar uma regra do Config gerenciada pelo cliente

O seguinte comando fornece o código JSON para adicionar uma regra do Config gerenciada pelo cliente:

```
aws configservice put-config-rule --config-rule file://
InstanceTypesAreT2micro.json
```

`InstanceTypesAreT2micro.json` é um arquivo JSON que contém a configuração da regra:

```
{
  "ConfigRuleName": "InstanceTypesAreT2micro",
  "Description": "Evaluates whether EC2 instances are the t2.micro type.",
  "Scope": {
    "ComplianceResourceTypes": [
      "AWS::EC2::Instance"
    ]
  },
  "Source": {
    "Owner": "CUSTOM_LAMBDA",
    "SourceIdentifier": "arn:aws:lambda:us-
east-1:123456789012:function:InstanceTypeCheck",
    "SourceDetails": [
      {
        "EventSource": "aws.config",
        "MessageType": "ConfigurationItemChangeNotification"
      }
    ]
  },
  "InputParameters": "{\"desiredInstanceType\":\"t2.micro\"}"
}
```

Para o `ComplianceResourceTypes` atributo, esse código JSON limita o escopo a recursos do `AWS::EC2::Instance` tipo, então o AWS Config avaliará somente instâncias do EC2 em relação à regra. Como essa regra é uma regra gerenciada pelo cliente, o `Owner` atributo é definido como`CUSTOM_LAMBDA`, e o `SourceIdentifier` atributo é definido como o ARN da função Lambda AWS . O `SourceDetails` objeto é obrigatório. Os parâmetros especificados para o `InputParameters` atributo são passados para a função AWS Lambda quando o AWS Config a invoca para avaliar os recursos em relação à regra.

Se o comando for bem-sucedido, o AWS Config não retornará nenhuma saída. Para verificar a configuração da regra, execute o comando describe-config-rules e especifique o nome da regra.

- Para obter detalhes da API, consulte PutConfigRuleem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def put_config_rule(self, rule_name):
        """
        Sets a configuration rule that prohibits making Amazon S3 buckets
 publicly
        readable.

        :param rule_name: The name to give the rule.
        """
        try:
            self.config_client.put_config_rule(
                ConfigRule={
                    "ConfigRuleName": rule_name,
                    "Description": "S3 Public Read Prohibited Bucket Rule",
                    "Scope": {
                        "ComplianceResourceTypes": [
                            "AWS::S3::Bucket",
                        ],
                    },
                    "Source": {
                        "Owner": "AWS",
```

```
                        "SourceIdentifier": "S3_BUCKET_PUBLIC_READ_PROHIBITED",
                    },
                    "InputParameters": "{}",
                    "ConfigRuleState": "ACTIVE",
                }
            )
            logger.info("Created configuration rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't create configuration rule %s.", rule_name)
            raise
```

- Para obter detalhes da API, consulte a PutConfigRuleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create a config rule for S3 bucket public read prohibition
    lo_cfs->putconfigrule(
      io_configrule = NEW /aws1/cl_cfsconfigrule(
        iv_configrulename = iv_rule_name
        iv_description = |S3 Public Read Prohibited Bucket Rule|
        io_scope = NEW /aws1/cl_cfsscope(
          it_complianceresourcetypes = VALUE /aws1/
cl_cfscplncresrctypes_w=>tt_complianceresourcetypes(
            ( NEW /aws1/cl_cfscplncresrctypes_w( |AWS::S3::Bucket| ) )
          )
        )
        io_source = NEW /aws1/cl_cfssource(
          iv_owner = |AWS|
          iv_sourceidentifier = |S3_BUCKET_PUBLIC_READ_PROHIBITED|
        )
```

```
        iv_inputparameters = '{}'
        iv_configrulestate = |ACTIVE|
      )
    ).
    MESSAGE 'Created AWS Config rule.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a PutConfigRulereferência da API AWS SDK for SAP ABAP.

Usar **`PutDeliveryChannel`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutDeliveryChannel`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos do gerenciamento de configurações

CLI

AWS CLI

Para criar um canal de entrega

O seguinte comando fornece as configurações do canal de entrega como código JSON:

```
aws configservice put-delivery-channel --delivery-channel file://
deliveryChannel.json
```

O arquivo `deliveryChannel.json` especifica os atributos do canal de entrega:

```
{
    "name": "default",
    "s3BucketName": "config-bucket-123456789012",
    "snsTopicARN": "arn:aws:sns:us-east-1:123456789012:config-topic",
    "configSnapshotDeliveryProperties": {
        "deliveryFrequency": "Twelve_Hours"
    }
}
```

Este exemplo define os seguintes atributos:

`name`: o nome do canal de entrega. Por padrão, o AWS Config atribui o nome `default` a um novo canal de entrega. Você não pode atualizar o nome do canal de entrega com o comando. `put-delivery-channel` Para ver as etapas para alterar o nome, consulte Como renomear o canal de entrega.`s3BucketName` - O nome do bucket do Amazon S3 para o qual o AWS Config entrega snapshots de configuração e arquivos de histórico de configuração. Se você especificar um bucket que pertença a outra AWS conta, esse bucket deve ter políticas que concedam permissões de acesso ao Config. AWS Para obter mais informações, consulte Permissões para o bucket do Amazon S3.

`snsTopicARN`- O Amazon Resource Name (ARN) do tópico do Amazon SNS para o qual o AWS Config envia notificações sobre alterações na configuração. Se você escolher um tópico de outra conta, o tópico deve ter políticas que concedam permissões de acesso ao Config. AWS Para obter mais informações, consulte o tópico Permissões para o Amazon SNS.

`configSnapshotDeliveryProperties`- Contém o `deliveryFrequency` atributo, que define com que frequência o AWS Config fornece instantâneos de configuração e com que frequência ele invoca avaliações para regras periódicas do Config.

Se o comando for bem-sucedido, o AWS Config não retornará nenhuma saída. Para verificar as configurações do seu canal de entrega, execute o comando describe-delivery-channels.

- Para obter detalhes da API, consulte PutDeliveryChannelem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo altera a propriedade deliveryFrequency de um canal de entrega existente.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName
 amzn-s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- Para obter detalhes da API, consulte PutDeliveryChannelem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo altera a propriedade deliveryFrequency de um canal de entrega existente.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName
 amzn-s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- Para obter detalhes da API, consulte PutDeliveryChannelem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para AWS Config usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS Config com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS Config ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos do gerenciamento de configurações

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
```

```
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
```

```
    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
```

```
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
```

```
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
```

```
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
```

```
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Exemplos de código para Connect Customer usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Connect Customer com um kit AWS de desenvolvimento de software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do administrador do Connect Customer — Mais informações sobre o Connect Customer.
- Referência da API Connect Customer — Detalhes sobre todas as ações disponíveis do Connect Customer.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para o Connect Customer usando AWS SDKs
  - Criação de uma instância de central de atendimento

Cenários para o Connect Customer usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Connect Customer with AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Connect Customer ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criação de uma instância de central de atendimento

Criação de uma instância de central de atendimento

O exemplo de código a seguir mostra como:

- Criar uma instância do Amazon Connect.
- Configurar um usuário administrador.
- Configurar opções de telefonia.
- Configurar um número de telefone.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create an Amazon Connect instance using AWS CLI
# This script follows the steps in the Amazon Connect instance creation tutorial

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="connect-instance-creation.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting Amazon Connect instance creation at $(date)" > "$LOG_FILE"

# Set default region
AWS_REGION="${AWS_REGION:-us-west-2}"
echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"

# Validate AWS CLI is installed and credentials are available
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
```

```
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid" | tee -a
 "$LOG_FILE"
    exit 1
fi

# Function to log commands and their output
log_cmd() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        return 1
    fi
    return 0
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 "$LOG_FILE"

    if [[ -n "${INSTANCE_ID:-}" ]]; then
        echo "Deleting Amazon Connect instance: $INSTANCE_ID" | tee -a
 "$LOG_FILE"
        log_cmd "aws connect delete-instance --instance-id '$INSTANCE_ID' --
region '$AWS_REGION'" || true
    fi

    echo "Cleanup completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
}
```

```
# Set trap to clean up on error
trap cleanup_on_error ERR EXIT

# Function to wait for instance to be fully active
wait_for_instance() {
    local instance_id="$1"
    local max_attempts=30
    local attempt=1

    echo "Waiting for instance $instance_id to become fully active..." | tee -a
 "$LOG_FILE"

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt of $max_attempts: Checking instance status..." |
 tee -a "$LOG_FILE"

        # Try to describe the instance
        local result
        result=$(log_cmd "aws connect describe-instance --instance-id
 '$instance_id' --region '$AWS_REGION' --output json" 2>&1) || true

        # Check if the command was successful and instance status is ACTIVE
        if [[ $? -eq 0 && "$result" =~ "ACTIVE" ]]; then
            echo "Instance is now fully active and ready to use." | tee -a
 "$LOG_FILE"
            return 0
        fi

        echo "Instance not fully active yet. Waiting 30 seconds before next
 check..." | tee -a "$LOG_FILE"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for instance to become fully active." | tee -a
 "$LOG_FILE"
    return 1
}

# Function to check and handle existing instances
check_existing_instances() {
    echo "Checking for existing Amazon Connect instances..." | tee -a "$LOG_FILE"
```

```
    local instances
    instances=$(log_cmd "aws connect list-instances --region '$AWS_REGION' --
output json" 2>&1) || true
    if [[ $? -ne 0 ]]; then
        echo "ERROR: Failed to list existing instances" | tee -a "$LOG_FILE"
        return 1
    fi

    # Check if there are any instances
    local instance_count
    instance_count=$(echo "$instances" | jq '.InstanceSummaryList | length' 2>/
dev/null || echo 0)

    if [[ $instance_count -gt 0 ]]; then
        echo "Found $instance_count existing Amazon Connect instance(s)" | tee -a
 "$LOG_FILE"
        echo "$instances" | jq '.InstanceSummaryList[] | {Id, Alias}' 2>/dev/null
 | tee -a "$LOG_FILE" || true

        echo ""
        echo "==========================================="
        echo "EXISTING INSTANCES FOUND"
        echo "==========================================="
        echo "Found $instance_count existing Amazon Connect instance(s)."
        echo "Auto-deleting existing instances to free up quota..." | tee -a
 "$LOG_FILE"

        echo "Deleting existing instances..." | tee -a "$LOG_FILE"

        # Extract instance IDs and delete each one
        local instance_ids
        instance_ids=$(echo "$instances" | jq -r '.InstanceSummaryList[].Id' 2>/
dev/null || echo "")

        while IFS= read -r id; do
            if [[ -n "$id" ]]; then
                echo "Deleting instance: $id" | tee -a "$LOG_FILE"
                log_cmd "aws connect delete-instance --instance-id '$id' --region
 '$AWS_REGION'" || {
                    echo "WARNING: Failed to delete instance $id" | tee -a
 "$LOG_FILE"
                }

                # Wait a bit between deletions
```

```
                sleep 5
            fi
        done <<< "$instance_ids"

        echo "Waiting for deletions to complete..." | tee -a "$LOG_FILE"
        sleep 30
    else
        echo "No existing Amazon Connect instances found" | tee -a "$LOG_FILE"
    fi

    return 0
}

# Check for existing instances before proceeding
check_existing_instances

# Generate a random instance alias to avoid naming conflicts
INSTANCE_ALIAS="connect-instance-$(openssl rand -hex 6 2>/dev/null || date +%s
%N)"
echo "Using instance alias: $INSTANCE_ALIAS" | tee -a "$LOG_FILE"

# Step 1: Create Amazon Connect instance
echo "Step 1: Creating Amazon Connect instance..." | tee -a "$LOG_FILE"
INSTANCE_RESULT=$(log_cmd "aws connect create-instance --identity-management-type
 CONNECT_MANAGED --instance-alias '$INSTANCE_ALIAS' --inbound-calls-enabled --
outbound-calls-enabled --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$INSTANCE_RESULT" $? "Failed to create Amazon Connect
 instance"; then
    # Check if the error is due to quota limit
    if [[ "$INSTANCE_RESULT" =~ "ServiceQuotaExceededException" ||
 "$INSTANCE_RESULT" =~ "Quota limit reached" ]]; then
        echo "Quota limit reached for Amazon Connect instances. Please delete
 existing instances or request a quota increase." | tee -a "$LOG_FILE"
    fi
    exit 1
fi

# Extract instance ID from the result using jq
INSTANCE_ID=$(echo "$INSTANCE_RESULT" | jq -r '.InstanceId' 2>/dev/null || echo
 "")
INSTANCE_ARN=$(echo "$INSTANCE_RESULT" | jq -r '.InstanceArn' 2>/dev/null || echo
 "")
```

```
if [[ -z "$INSTANCE_ID" ]]; then
    echo "ERROR: Failed to extract instance ID from the result" | tee -a
 "$LOG_FILE"
    exit 1
fi

echo "Successfully created Amazon Connect instance with ID: $INSTANCE_ID" | tee -
a "$LOG_FILE"
echo "Instance ARN: $INSTANCE_ARN" | tee -a "$LOG_FILE"

# Tag the instance
aws connect tag-resource --resource-arn "$INSTANCE_ARN" \
    --tags project=doc-smith,tutorial=connect-gs \
    --region "$AWS_REGION"

# Wait for the instance to be fully created and active
if ! wait_for_instance "$INSTANCE_ID"; then
    echo "ERROR: Instance did not become fully active within the timeout period"
 | tee -a "$LOG_FILE"
    exit 1
fi

# Step 2: Get security profiles to find the Admin profile ID
echo "Step 2: Getting security profiles..." | tee -a "$LOG_FILE"
SECURITY_PROFILES=$(log_cmd "aws connect list-security-profiles --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$SECURITY_PROFILES" $? "Failed to list security profiles"; then
    exit 1
fi

# Extract Admin security profile ID using jq
ADMIN_PROFILE_ID=$(echo "$SECURITY_PROFILES" | jq -r
 '.SecurityProfileSummaryList[] | select(.Name=="Admin") | .Id' 2>/dev/null |
 head -1 || echo "")

if [[ -z "$ADMIN_PROFILE_ID" ]]; then
    echo "ERROR: Failed to find Admin security profile ID" | tee -a "$LOG_FILE"
    echo "Available security profiles:" | tee -a "$LOG_FILE"
    echo "$SECURITY_PROFILES" | jq '.SecurityProfileSummaryList[] | {Id,
 Name}' 2>/dev/null | tee -a "$LOG_FILE" || echo "$SECURITY_PROFILES" | tee -a
 "$LOG_FILE"
    exit 1
fi
```

```
echo "Found Admin security profile ID: $ADMIN_PROFILE_ID" | tee -a "$LOG_FILE"

# Step 3: Get routing profiles to find a default routing profile ID
echo "Step 3: Getting routing profiles..." | tee -a "$LOG_FILE"
ROUTING_PROFILES=$(log_cmd "aws connect list-routing-profiles --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$ROUTING_PROFILES" $? "Failed to list routing profiles"; then
    exit 1
fi

# Extract the first routing profile ID using jq
ROUTING_PROFILE_ID=$(echo "$ROUTING_PROFILES" | jq -r
 '.RoutingProfileSummaryList[0].Id' 2>/dev/null || echo "")

if [[ -z "$ROUTING_PROFILE_ID" ]]; then
    echo "ERROR: Failed to find a routing profile ID" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Found routing profile ID: $ROUTING_PROFILE_ID" | tee -a "$LOG_FILE"

# Step 4: Create an admin user
echo "Step 4: Creating admin user..." | tee -a "$LOG_FILE"

# Generate a secure password
ADMIN_PASSWORD="Connect$(openssl rand -base64 12 2>/dev/null || head -c 12 /dev/
urandom | base64)"

USER_RESULT=$(log_cmd "aws connect create-user --instance-id '$INSTANCE_ID'
 --username admin --password '$ADMIN_PASSWORD' --identity-info
 FirstName=Admin,LastName=User,Email=admin@example.com --phone-config
 PhoneType=DESK_PHONE,AutoAccept=true,AfterContactWorkTimeLimit=30,DeskPhoneNumber=
+12065550100 --security-profile-ids '$ADMIN_PROFILE_ID' --routing-profile-id
 '$ROUTING_PROFILE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$USER_RESULT" $? "Failed to create admin user"; then
    exit 1
fi

# Extract user ID using jq
USER_ID=$(echo "$USER_RESULT" | jq -r '.UserId' 2>/dev/null || echo "")
```

```
if [[ -z "$USER_ID" ]]; then
    echo "ERROR: Failed to extract user ID from the result" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Successfully created admin user with ID: $USER_ID" | tee -a "$LOG_FILE"
echo "Admin password: $ADMIN_PASSWORD" | tee -a "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Step 5: Configure telephony options
echo "Step 5: Configuring telephony options..." | tee -a "$LOG_FILE"

# Enable early media
EARLY_MEDIA_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-id
 '$INSTANCE_ID' --attribute-type EARLY_MEDIA --value true --region '$AWS_REGION'"
 2>&1) || true

if ! check_error "$EARLY_MEDIA_RESULT" $? "Failed to enable early media"; then
    exit 1
fi

# Enable multi-party calls and enhanced monitoring for voice
MULTI_PARTY_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-
id '$INSTANCE_ID' --attribute-type MULTI_PARTY_CONFERENCE --value true --region
 '$AWS_REGION'" 2>&1) || true

if ! check_error "$MULTI_PARTY_RESULT" $? "Failed to enable multi-party calls";
 then
    exit 1
fi

# Enable multi-party chats and enhanced monitoring for chat
MULTI_PARTY_CHAT_RESULT=$(log_cmd "aws connect update-instance-attribute --
instance-id '$INSTANCE_ID' --attribute-type MULTI_PARTY_CHAT_CONFERENCE --value
 true --region '$AWS_REGION'" 2>&1) || true

if ! check_error "$MULTI_PARTY_CHAT_RESULT" $? "Failed to enable multi-party
 chats"; then
    exit 1
fi

echo "Successfully configured telephony options" | tee -a "$LOG_FILE"

# Step 6: View storage configurations
```

```
echo "Step 6: Viewing storage configurations..." | tee -a "$LOG_FILE"

# List storage configurations for chat transcripts
STORAGE_CONFIGS=$(log_cmd "aws connect list-instance-storage-configs --instance-
id '$INSTANCE_ID' --resource-type CHAT_TRANSCRIPTS --region '$AWS_REGION' --
output json" 2>&1) || true

if ! check_error "$STORAGE_CONFIGS" $? "Failed to list storage configurations";
 then
    exit 1
fi

echo "Successfully retrieved storage configurations" | tee -a "$LOG_FILE"

# Step 7: Verify instance details
echo "Step 7: Verifying instance details..." | tee -a "$LOG_FILE"
INSTANCE_DETAILS=$(log_cmd "aws connect describe-instance --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$INSTANCE_DETAILS" $? "Failed to describe instance"; then
    exit 1
fi

echo "Successfully verified instance details" | tee -a "$LOG_FILE"

# Step 8: Search for available phone numbers (optional)
echo "Step 8: Searching for available phone numbers..." | tee -a "$LOG_FILE"
PHONE_NUMBERS=$(log_cmd "aws connect search-available-phone-numbers --target-arn
 '$INSTANCE_ARN' --phone-number-type TOLL_FREE --phone-number-country-code US --
max-results 5 --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$PHONE_NUMBERS" $? "Failed to search for available phone
 numbers"; then
    exit 1
fi

# Extract the first phone number if available using jq
PHONE_NUMBER=$(echo "$PHONE_NUMBERS" | jq -r
 '.AvailableNumbersList[0].PhoneNumber' 2>/dev/null || echo "")

if [[ -n "$PHONE_NUMBER" ]]; then
    echo "Found available phone number: $PHONE_NUMBER" | tee -a "$LOG_FILE"

    echo ""
```

```
    echo "============================================"
    echo "CLAIM PHONE NUMBER"
    echo "============================================"
    echo "Auto-claiming available phone number $PHONE_NUMBER..." | tee -a
 "$LOG_FILE"

    CLAIM_RESULT=$(log_cmd "aws connect claim-phone-number --target-arn
 '$INSTANCE_ARN' --phone-number '$PHONE_NUMBER' --region '$AWS_REGION' --output
 json" 2>&1) || true

    if ! check_error "$CLAIM_RESULT" $? "Failed to claim phone number"; then
        echo "WARNING: Failed to claim phone number, but continuing with script"
 | tee -a "$LOG_FILE"
    else
        echo "Successfully claimed phone number" | tee -a "$LOG_FILE"
        # Extract the phone number ID from the claim result using jq
        PHONE_NUMBER_ID=$(echo "$CLAIM_RESULT" | jq -r '.PhoneNumberId' 2>/dev/
null || echo "")
    fi
else
    echo "No available phone numbers found" | tee -a "$LOG_FILE"
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "Amazon Connect Instance ID: $INSTANCE_ID"
echo "Amazon Connect Instance ARN: $INSTANCE_ARN"
echo "Admin User ID: $USER_ID"
echo "Admin Username: admin"
echo "Admin Password: $ADMIN_PASSWORD"
if [[ -n "${PHONE_NUMBER:-}" ]]; then
    echo "Claimed Phone Number: $PHONE_NUMBER"
    if [[ -n "${PHONE_NUMBER_ID:-}" ]]; then
        echo "Claimed Phone Number ID: $PHONE_NUMBER_ID"
    fi
fi
echo "============================================"
echo ""

# Auto-confirm cleanup and clean up resources
echo ""
```

```
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Auto-starting cleanup..." | tee -a "$LOG_FILE"

# Release claimed phone number if applicable
if [[ -n "${PHONE_NUMBER_ID:-}" ]]; then
    echo "Releasing phone number: $PHONE_NUMBER_ID" | tee -a "$LOG_FILE"
    RELEASE_RESULT=$(log_cmd "aws connect release-phone-number --phone-number-id
 '$PHONE_NUMBER_ID' --region '$AWS_REGION'" 2>&1) || true

    if ! check_error "$RELEASE_RESULT" $? "Failed to release phone number"; then
        echo "WARNING: Failed to release phone number" | tee -a "$LOG_FILE"
    else
        echo "Successfully released phone number" | tee -a "$LOG_FILE"
    fi

    echo "Waiting for phone number release to complete..." | tee -a "$LOG_FILE"
    sleep 10
fi

# Delete the Amazon Connect instance (this will also delete all associated
 resources)
echo "Deleting Amazon Connect instance: $INSTANCE_ID" | tee -a "$LOG_FILE"
DELETE_RESULT=$(log_cmd "aws connect delete-instance --instance-id '$INSTANCE_ID'
 --region '$AWS_REGION'" 2>&1) || true

if ! check_error "$DELETE_RESULT" $? "Failed to delete instance"; then
    echo "WARNING: Failed to delete instance" | tee -a "$LOG_FILE"
else
    echo "Successfully deleted instance" | tee -a "$LOG_FILE"
fi

echo "Cleanup completed. All resources have been deleted." | tee -a "$LOG_FILE"

echo "Script completed successfully. See $LOG_FILE for details." | tee -a
 "$LOG_FILE"

trap - ERR EXIT
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - ClaimPhoneNumber
  - CreateInstance

- CreateUser
- DeleteInstance
- DescribeInstance
- ListInstanceStorageConfigs
- ListInstances
- ListRoutingProfiles
- ListSecurityProfiles
- ReleasePhoneNumber
- SearchAvailablePhoneNumbers
- UpdateInstanceAttribute

Exemplos de código para AWS Control Tower usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Control Tower com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS Control Tower Guia do usuário — Mais informações sobre AWS Control Tower.
- AWS Control Tower Referência da API — Detalhes sobre todas as AWS Control Tower ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Control Tower usar AWS SDKs
  - Hello AWS Control Tower
  - Aprenda as noções básicas do AWS Control Tower com um AWS SDK
  - Ações para AWS Control Tower usar AWS SDKs
    - Use DisableBaselinecom um AWS SDK ou CLI
    - Use DisableControlcom um AWS SDK ou CLI
    - Use EnableBaselinecom um AWS SDK ou CLI
    - Use EnableControlcom um AWS SDK ou CLI
    - Use GetBaselineOperationcom um AWS SDK ou CLI
    - Use GetControlOperationcom um AWS SDK ou CLI
    - Use ListBaselinescom um AWS SDK ou CLI
    - Use ListEnabledBaselinescom um AWS SDK ou CLI
    - Use ListEnabledControlscom um AWS SDK ou CLI
    - Use ListLandingZonescom um AWS SDK ou CLI
    - Use ResetEnabledBaselinecom um AWS SDK ou CLI

Exemplos básicos para AWS Control Tower usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Control Tower com AWS SDKs.

Exemplos

- Hello AWS Control Tower
- Aprenda as noções básicas do AWS Control Tower com um AWS SDK
- Ações para AWS Control Tower usar AWS SDKs
  - Use DisableBaselinecom um AWS SDK ou CLI
  - Use DisableControlcom um AWS SDK ou CLI
  - Use EnableBaselinecom um AWS SDK ou CLI
  - Use EnableControlcom um AWS SDK ou CLI
  - Use GetBaselineOperationcom um AWS SDK ou CLI
  - Use GetControlOperationcom um AWS SDK ou CLI
  - Use ListBaselinescom um AWS SDK ou CLI

- Use ListEnabledBaselinescom um AWS SDK ou CLI
- Use ListEnabledControlscom um AWS SDK ou CLI
- Use ListLandingZonescom um AWS SDK ou CLI
- Use ResetEnabledBaselinecom um AWS SDK ou CLI

Hello AWS Control Tower

O exemplo de código a seguir mostra como começar a usar o AWS Control Tower.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
using LogLevel = Microsoft.Extensions.Logging.LogLevel;

namespace ControlTowerActions;

/// <summary>
/// A class that introduces the AWS Control Tower by listing the
/// available baselines for the account.
/// </summary>
public class HelloControlTower
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
```

```
    {
        // Set up dependency injection for AWS Control Tower.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonControlTower>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloControlTower>();

        var amazonClient =
host.Services.GetRequiredService<IAmazonControlTower>();

        Console.Clear();
        Console.WriteLine("Hello, AWS Control Tower! Let's list available
baselines:");
        Console.WriteLine();

        var baselines = new List<BaselineSummary>();

        try
        {
            var baselinesPaginator = amazonClient.Paginators.ListBaselines(new
ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            Console.WriteLine($"{baselines.Count} baseline(s) retrieved.");
            foreach (var baseline in baselines)
            {
                Console.WriteLine($"\t{baseline.Name}");
            }
        }
        catch (Amazon.ControlTower.Model.AccessDeniedException)
```

```
        {
            Console.WriteLine("Access denied. Please ensure you have the
 necessary permissions.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListBaselinesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class HelloControlTower {

    public static void main(String[] args) {
        try {
            ControlTowerClient controlTowerClient = ControlTowerClient.builder()
                .build() ;
            helloControlTower(controlTowerClient);
        } catch (ControlTowerException e) {
            System.out.println("Control Tower error occurred: " +
 e.awsErrorDetails().errorMessage());
        }
    }

    /**
     * Use the AWS SDK for Java (v2) to create an AWS Control Tower client
```

```
     * and list all available baselines.
     * This example uses the default settings specified in your shared
 credentials
     * and config files.
     *
     * @param controlTowerClient A ControlTowerClient object. This object wraps
     *                           the low-level AWS Control Tower service API.
     */
    public static void helloControlTower(ControlTowerClient controlTowerClient) {
        System.out.println("Hello, AWS Control Tower! Let's list available
 baselines:\n");

        ListBaselinesIterable paginator =
 controlTowerClient.listBaselinesPaginator(
                ListBaselinesRequest.builder().build());
        List<String> baselineNames = new ArrayList<>();

        try {
            paginator.stream()
                .flatMap(response -> response.baselines().stream())
                    .forEach(baseline -> baselineNames.add(baseline.name()));

            System.out.println(baselineNames.size() + " baseline(s) retrieved.");
            for (String baselineName : baselineNames) {
                System.out.println("\t" + baselineName);
            }

        } catch (ControlTowerException e) {
            if ("AccessDeniedException".equals(e.awsErrorDetails().errorCode()))
 {
                System.out.println("Access denied. Please ensure you have the
 necessary permissions.");
            } else {
                System.out.println("An error occurred: " + e.getMessage());
            }
        }
    }
}
```

- Para obter detalhes da API, consulte ListBaselinesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
from typing import Any, List


def hello_controltower(controltower_client: Any) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Control Tower client
    and list all available baselines.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param controltower_client: A Boto3 AWS Control Tower Client object. This
 object wraps
                                the low-level AWS Control Tower service API.
    """
    print("Hello, AWS Control Tower! Let's list available baselines:\n")
    paginator = controltower_client.get_paginator("list_baselines")
    page_iterator = paginator.paginate()

    baseline_names: List[str] = []
    try:
        for page in page_iterator:
            for baseline in page["baselines"]:
                baseline_names.append(baseline["name"])

        print(f"{len(baseline_names)} baseline(s) retrieved.")
        for baseline_name in baseline_names:
            print(f"\t{baseline_name}")

    except controltower_client.exceptions.AccessDeniedException:
        print("Access denied. Please ensure you have the necessary permissions.")
    except Exception as e:
```

```
        print(f"An error occurred: {str(e)}")


if __name__ == "__main__":
    hello_controltower(boto3.client("controltower"))
```

- Para obter detalhes da API, consulte a ListBaselinesReferência da API AWS SDK for Python (Boto3).

Aprenda as noções básicas do AWS Control Tower com um AWS SDK

Os exemplos de código a seguir mostram como:

- Liste as zonas de pouso.
- Liste, ative, obtenha, redefina e desative as linhas de base.
- Liste, ative, obtenha e desative controles.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS Control Tower recursos.

```
using Amazon.ControlCatalog;
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using Amazon.Organizations;
using Amazon.Organizations.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using ControlTowerActions;
using Microsoft.Extensions.DependencyInjection;
```

```
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace ControlTowerBasics;

/// <summary>
/// Scenario class for AWS Control Tower basics.
/// </summary>
public class ControlTowerBasics
{
    public static bool isInteractive = true;
    public static ILogger logger = null!;
    public static IAmazonOrganizations? orgClient = null;
    public static IAmazonSecurityTokenService? stsClient = null;
    public static ControlTowerWrapper? wrapper = null;
    private static string? ouArn;
    private static bool useLandingZone = false;

    /// <summary>
    /// Main entry point for the AWS Control Tower basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonControlTower>()
                .AddAWSService<IAmazonControlCatalog>()
                .AddAWSService<IAmazonOrganizations>()
                .AddAWSService<IAmazonSecurityTokenService>()
                .AddTransient<ControlTowerWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<ControlTowerBasics>();

        wrapper = host.Services.GetRequiredService<ControlTowerWrapper>();
        orgClient = host.Services.GetRequiredService<IAmazonOrganizations>();
        stsClient =
 host.Services.GetRequiredService<IAmazonSecurityTokenService>();

        await RunScenario();
    }
```

```
    /// <summary>
    /// Runs the example scenario.
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("\tWelcome to the AWS Control Tower with ControlCatalog
 example scenario.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("This demo will walk you through working with AWS
 Control Tower for landing zones,");
        Console.WriteLine("managing baselines, and working with controls.");

        try
        {
            var accountId = (await stsClient!.GetCallerIdentityAsync(new
 GetCallerIdentityRequest())).Account;
            Console.WriteLine($"\nAccount ID: {accountId}");

            Console.WriteLine("\nSome demo operations require the use of a
 landing zone.");
            Console.WriteLine("You can use an existing landing zone or opt out of
 these operations in the demo.");
            Console.WriteLine("For instructions on how to set up a landing
 zone,");
            Console.WriteLine("see https://docs.aws.amazon.com/controltower/
latest/userguide/getting-started-from-console.html");

            // List available landing zones
            var landingZones = await wrapper!.ListLandingZonesAsync();
            if (landingZones.Count > 0)
            {
                Console.WriteLine("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.Count; i++)
                {
                    Console.WriteLine($"{i + 1}. {landingZones[i].Arn}");
                }

                Console.Write($"\nDo you want to use the first landing zone in
 the list ({landingZones[0].Arn})? (y/n): ");
                if (GetUserConfirmation())
                {
                    useLandingZone = true;
```

```
                    Console.WriteLine($"Using landing zone:
 {landingZones[0].Arn}");
                    ouArn = await SetupOrganizationAsync();
                }
            }

            // Managing Baselines
            Console.WriteLine("\nManaging Baselines:");
            var baselines = await wrapper.ListBaselinesAsync();
            Console.WriteLine("\nListing available Baselines:");
            BaselineSummary? controlTowerBaseline = null;
            foreach (var baseline in baselines)
            {
                if (baseline.Name == "AWSControlTowerBaseline")
                    controlTowerBaseline = baseline;
                Console.WriteLine($"  - {baseline.Name}");
            }

            EnabledBaselineSummary? identityCenterBaseline = null;
            string? baselineArn = null;

            if (useLandingZone && ouArn != null)
            {
                Console.WriteLine("\nListing enabled baselines:");
                var enabledBaselines = await wrapper.ListEnabledBaselinesAsync();
                foreach (var baseline in enabledBaselines)
                {
                    if (baseline.BaselineIdentifier.Contains("baseline/
LN25R72TTG6IGPTQ"))
                        identityCenterBaseline = baseline;
                    Console.WriteLine($"  - {baseline.BaselineIdentifier}");
                }

                if (controlTowerBaseline != null)
                {
                    Console.Write("\nDo you want to enable the Control Tower
 Baseline? (y/n): ");
                    if (GetUserConfirmation())
                    {
                        Console.WriteLine("\nEnabling Control Tower Baseline.");
                        var icBaselineArn = identityCenterBaseline?.Arn;
                        baselineArn = await wrapper.EnableBaselineAsync(ouArn,
                            controlTowerBaseline.Arn, "5.0", icBaselineArn ??
 "");
```

```
                        var alreadyEnabled = false;
                        if (baselineArn != null)
                        {
                            Console.WriteLine($"Enabled baseline ARN:
{baselineArn}");
                        }
                        else
                        {
                            // Find the enabled baseline
                            foreach (var enabled in enabledBaselines)
                            {
                                if (enabled.BaselineIdentifier ==
controlTowerBaseline.Arn)
                                {
                                    baselineArn = enabled.Arn;
                                    alreadyEnabled = true;
                                    Console.WriteLine("No change, the selected
baseline was already enabled.");
                                    break;
                                }
                            }
                        }

                        if (baselineArn != null)
                        {
                            Console.Write("\nDo you want to reset the Control
Tower Baseline? (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine($"\nResetting Control Tower
Baseline: {baselineArn}");
                                var operationId = await
wrapper.ResetEnabledBaselineAsync(baselineArn);
                                Console.WriteLine($"Reset baseline operation id:
{operationId}");
                            }

                            Console.Write("\nDo you want to disable the Control
Tower Baseline? (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine($"Disabling baseline ARN:
{baselineArn}");
```

```
                                var operationId = await
wrapper.DisableBaselineAsync(baselineArn);
                                Console.WriteLine($"Disabled baseline operation
id: {operationId}");
                                if (alreadyEnabled)
                                {
                                    Console.WriteLine($"\nRe-enabling Control
Tower Baseline: {baselineArn}");
                                    // Re-enable the Control Tower baseline if it
was originally enabled.
                                    await wrapper.EnableBaselineAsync(ouArn,
                                        controlTowerBaseline.Arn, "5.0",
icBaselineArn ?? "");
                                }
                            }
                        }
                    }
                }
            }

            // Managing Controls
            Console.WriteLine("\nManaging Controls:");
            var controls = await wrapper.ListControlsAsync();
            Console.WriteLine("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.Min(5, controls.Count); i++)
            {
                Console.WriteLine($"{i + 1}. {controls[i].Name} -
{controls[i].Arn}");
            }

            if (useLandingZone && ouArn != null)
            {
                var enabledControls = await
wrapper.ListEnabledControlsAsync(ouArn);
                Console.WriteLine("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.Count; i++)
                {
                    Console.WriteLine($"{i + 1}.
{enabledControls[i].ControlIdentifier}");
                }

                // Find first non-enabled control
                var enabledControlArns = enabledControls.Select(c =>
c.Arn).ToHashSet();
```

```
                var controlArn = controls.FirstOrDefault(c => !
enabledControlArns.Contains(c.Arn))?.Arn;

                if (controlArn != null)
                {
                    Console.Write($"\nDo you want to enable the control
 {controlArn}? (y/n): ");
                    if (GetUserConfirmation())
                    {
                        Console.WriteLine($"\nEnabling control: {controlArn}");
                        var operationId = await
 wrapper.EnableControlAsync(controlArn, ouArn);
                        if (operationId != null)
                        {
                            Console.WriteLine($"Enabled control with operation
 id: {operationId}");

                            Console.Write("\nDo you want to disable the control?
 (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine("\nDisabling the control...");
                                var disableOpId = await
 wrapper.DisableControlAsync(controlArn, ouArn);
                                Console.WriteLine($"Disable operation ID:
 {disableOpId}");
                            }
                        }
                    }
                }
            }

            Console.WriteLine("\nThis concludes the example scenario.");
            Console.WriteLine("Thanks for watching!");
            Console.WriteLine(new string('-', 88));
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "An error occurred during the Control Tower
 scenario.");
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
```

```
    /// <summary>
    /// Sets up AWS Organizations and creates or finds a Sandbox OU.
    /// </summary>
    /// <returns>The ARN of the Sandbox organizational unit.</returns>
    private static async Task<string> SetupOrganizationAsync()
    {
        Console.WriteLine("\nChecking organization status...");

        try
        {
            var orgResponse = await orgClient!.DescribeOrganizationAsync(new
DescribeOrganizationRequest());
            var orgId = orgResponse.Organization.Id;
            Console.WriteLine($"Account is part of organization: {orgId}");
        }
        catch (AWSOrganizationsNotInUseException)
        {
            Console.WriteLine("No organization found. Creating a new
organization...");
            var createResponse = await orgClient!.CreateOrganizationAsync(new
CreateOrganizationRequest { FeatureSet = OrganizationFeatureSet.ALL });
            var orgId = createResponse.Organization.Id;
            Console.WriteLine($"Created new organization: {orgId}");
        }

        // Look for Sandbox OU
        var roots = await orgClient.ListRootsAsync(new ListRootsRequest());
        var rootId = roots.Roots[0].Id;

        Console.WriteLine("Checking for Sandbox OU...");
        var ous = await orgClient.ListOrganizationalUnitsForParentAsync(new
ListOrganizationalUnitsForParentRequest { ParentId = rootId });
        var sandboxOu = ous.OrganizationalUnits.FirstOrDefault(ou => ou.Name ==
"Sandbox");

        if (sandboxOu == null)
        {
            Console.WriteLine("Creating Sandbox OU...");
            var createOuResponse = await
orgClient.CreateOrganizationalUnitAsync(new CreateOrganizationalUnitRequest
{ ParentId = rootId, Name = "Sandbox" });
            sandboxOu = createOuResponse.OrganizationalUnit;
            Console.WriteLine($"Created new Sandbox OU: {sandboxOu.Id}");
        }
```

```
        else
        {
            Console.WriteLine($"Found existing Sandbox OU: {sandboxOu.Id}");
        }

        return sandboxOu.Arn;
    }

    /// <summary>
    /// Gets user confirmation by waiting for input or returning true if not
 interactive.
    /// </summary>
    /// <returns>True if user enters 'y' or if isInteractive is false, otherwise
 false.</returns>
    private static bool GetUserConfirmation()
    {
        return Console.ReadLine()?.ToLower() == "y" || !isInteractive;
    }
}
```

Métodos de encapsulamento que são chamados pelo cenário para gerenciar as ações do Aurora.

```
using Amazon.ControlCatalog;
using Amazon.ControlCatalog.Model;
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using ValidationException = Amazon.ControlTower.Model.ValidationException;

namespace ControlTowerActions;

/// <summary>
/// Methods to perform AWS Control Tower actions.
/// </summary>
public class ControlTowerWrapper
{
    private readonly IAmazonControlTower _controlTowerService;
    private readonly IAmazonControlCatalog _controlCatalogService;

    /// <summary>
```

```
    /// Constructor for the wrapper class containing AWS Control Tower actions.
    /// </summary>
    /// <param name="controlTowerService">The AWS Control Tower client object.</
param>
    /// <param name="controlCatalogService">The AWS Control Catalog client
 object.</param>
    public ControlTowerWrapper(IAmazonControlTower controlTowerService,
 IAmazonControlCatalog controlCatalogService)
    {
        _controlTowerService = controlTowerService;
        _controlCatalogService = controlCatalogService;
    }

    /// <summary>
    /// List the AWS Control Tower landing zones for an account.
    /// </summary>
    /// <returns>A list of LandingZoneSummary objects.</returns>
    public async Task<List<LandingZoneSummary>> ListLandingZonesAsync()
    {
        try
        {
            var landingZones = new List<LandingZoneSummary>();

            var landingZonesPaginator =
 _controlTowerService.Paginators.ListLandingZones(new ListLandingZonesRequest());

            await foreach (var response in landingZonesPaginator.Responses)
            {
                landingZones.AddRange(response.LandingZones);
            }

            return landingZones;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list landing zones. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all baselines.
```

```
    /// </summary>
    /// <returns>A list of baseline summaries.</returns>
    public async Task<List<BaselineSummary>> ListBaselinesAsync()
    {
        try
        {
            var baselines = new List<BaselineSummary>();

            var baselinesPaginator =
_controlTowerService.Paginators.ListBaselines(new ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            return baselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list baselines. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all enabled baselines.
    /// </summary>
    /// <returns>A list of enabled baseline summaries.</returns>
    public async Task<List<EnabledBaselineSummary>> ListEnabledBaselinesAsync()
    {
        try
        {
            var enabledBaselines = new List<EnabledBaselineSummary>();

            var enabledBaselinesPaginator =
_controlTowerService.Paginators.ListEnabledBaselines(new
ListEnabledBaselinesRequest());

            await foreach (var response in enabledBaselinesPaginator.Responses)
            {
                enabledBaselines.AddRange(response.EnabledBaselines);
```

```
            }

            return enabledBaselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled baselines. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Enable a baseline for the specified target.
    /// </summary>
    /// <param name="targetIdentifier">The ARN of the target.</param>
    /// <param name="baselineIdentifier">The identifier of baseline to enable.</
param>
    /// <param name="baselineVersion">The version of baseline to enable.</param>
    /// <param name="identityCenterBaseline">The identifier of identity center
 baseline if it is enabled.</param>
    /// <returns>The enabled baseline ARN or null.</returns>
    public async Task<string?> EnableBaselineAsync(string targetIdentifier,
 string baselineIdentifier, string baselineVersion, string
 identityCenterBaseline)
    {
        try
        {
            var parameters = new List<EnabledBaselineParameter>();
            if (!string.IsNullOrEmpty(identityCenterBaseline))
            {
                parameters.Add(
                    new EnabledBaselineParameter
                    {
                        Key = "IdentityCenterEnabledBaselineArn",
                        Value = identityCenterBaseline
                    });
            }
            var request = new EnableBaselineRequest
            {
                BaselineIdentifier = baselineIdentifier,
                BaselineVersion = baselineVersion,
                TargetIdentifier = targetIdentifier,
```

```
                Parameters = parameters
            };

            var response = await
_controlTowerService.EnableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return response.Arn;
        }
        catch (ValidationException ex)
        {
            if (ex.Message.Contains("already enabled"))
                Console.WriteLine("Baseline is already enabled for this target");
            else { Console.WriteLine(ex.Message); }
            // Write the message and return null if baseline cannot be enabled.
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Disable a baseline for a specific target and wait for the operation to
complete.
    /// </summary>
```

```
    /// <param name="enabledBaselineIdentifier">The identifier of the baseline to
disable.</param>
    /// <returns>The operation ID or null if there was a conflict.</returns>
    public async Task<string?> DisableBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new DisableBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.DisableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (ConflictException ex)
        {
            Console.WriteLine($"Conflict disabling baseline: {ex.Message}.
Skipping disable step.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
```

```
    }


    /// <summary>
    /// Reset an enabled baseline for a specific target.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the enabled
baseline to reset.</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> ResetEnabledBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new ResetEnabledBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.ResetEnabledBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Target not found, unable to reset enabled
baseline.");
            throw;
        }
```

```
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't reset enabled baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Get the status of a baseline operation.
    /// </summary>
    /// <param name="operationId">The ID of the baseline operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<BaselineOperationStatus> GetBaselineOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetBaselineOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetBaselineOperationAsync(request);
            return response.BaselineOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get baseline operation status. Here's
why: {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List enabled controls for a target organizational unit.
```

```
    /// </summary>
    /// <param name="targetIdentifier">The target organizational unit
identifier.</param>
    /// <returns>A list of enabled control summaries.</returns>
    public async Task<List<EnabledControlSummary>>
ListEnabledControlsAsync(string targetIdentifier)
    {
        try
        {
            var request = new ListEnabledControlsRequest
            {
                TargetIdentifier = targetIdentifier
            };

            var enabledControls = new List<EnabledControlSummary>();

            var enabledControlsPaginator =
_controlTowerService.Paginators.ListEnabledControls(request);

            await foreach (var response in enabledControlsPaginator.Responses)
            {
                enabledControls.AddRange(response.EnabledControls);
            }

            return enabledControls;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return new List<EnabledControlSummary>();
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled controls. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Enable a control for a specified target.
```

```
    /// </summary>
    /// <param name="controlArn">The ARN of the control to enable.</param>
    /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
    /// <returns>The operation ID or null if already enabled.</returns>
    public async Task<string?> EnableControlAsync(string controlArn, string
targetIdentifier)
    {
        try
        {
            Console.WriteLine(controlArn);
            Console.WriteLine(targetIdentifier);

            var request = new EnableControlRequest
            {
                ControlIdentifier = controlArn,
                TargetIdentifier = targetIdentifier
            };

            var response = await
_controlTowerService.EnableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ValidationException ex) when
(ex.Message.Contains("already enabled"))
        {
            Console.WriteLine("Control is already enabled for this target");
            return null;
        }
```

```
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Disable a control for a specified target.
    /// </summary>
    /// <param name="controlArn">The ARN of the control to disable.</param>
    /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> DisableControlAsync(string controlArn, string
targetIdentifier)
    {
        try
        {
            var request = new DisableControlRequest
            {
                ControlIdentifier = controlArn,
                TargetIdentifier = targetIdentifier
            };

            var response = await
_controlTowerService.DisableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
```

```
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Control not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Get the status of a control operation.
    /// </summary>
    /// <param name="operationId">The ID of the control operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<ControlOperationStatus> GetControlOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetControlOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetControlOperationAsync(request);
            return response.ControlOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
```

```
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get control operation status. Here's
 why: {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all controls in the Control Tower control catalog.
    /// </summary>
    /// <returns>A list of control summaries.</returns>
    public async Task<List<ControlSummary>> ListControlsAsync()
    {
        try
        {
            var controls = new List<ControlSummary>();

            var controlsPaginator =
 _controlCatalogService.Paginators.ListControls(new
 Amazon.ControlCatalog.Model.ListControlsRequest());

            await foreach (var response in controlsPaginator.Responses)
            {
                controls.AddRange(response.Controls);
            }

            return controls;
        }
        catch (AmazonControlCatalogException ex)
        {
            Console.WriteLine($"Couldn't list controls. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS Control Tower recursos.

```
public class ControlTowerScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Scanner scanner = new Scanner(in);

    private static OrganizationsClient orgClient;
```

```
    private static ControlCatalogClient catClient;

    private static String ouId = null;
    private static String ouArn = null;
    private static String landingZoneArn = null;
    private static boolean useLandingZone = false;

    private String stack = null;
    private String accountId = null;

    public static void main(String[] args) {

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Control Tower basics scenario!");
        System.out.println(DASHES);

        try {
            runScenarioAsync();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // ----------------------------
    // Utilities
    // ----------------------------
    private static boolean askYesNo(String msg) {
        System.out.println(msg);
        return scanner.nextLine().trim().toLowerCase().startsWith("y");
    }

    private static void runScenarioAsync() {
        try {
            ControlTowerActions actions = new ControlTowerActions();

            // ----------------------------
            // Step 1: Landing Zones
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("""
        Some demo operations require the use of a landing zone.
        You can use an existing landing zone or opt out of these operations in
the demo.
        For instructions on how to set up a landing zone,
```

```
        see https://docs.aws.amazon.com/controltower/latest/userguide/getting-
started-from-console.html
        """);

            System.out.println("Step 1: Listing landing zones...");
            waitForInputToContinue(scanner);

            List<LandingZoneSummary> landingZones =
                    actions.listLandingZonesAsync().join();

            if (landingZones.isEmpty()) {
                System.out.println("No landing zones found. Landing-zone-
dependent steps will be skipped.");
                useLandingZone = false;
                waitForInputToContinue(scanner);
            } else {
                System.out.println("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.size(); i++) {
                    System.out.printf("%d) %s%n", i + 1,
 landingZones.get(i).arn());
                }

                if (askYesNo("Do you want to use the first landing zone in the
 list (" +
                        landingZones.get(0).arn() + ")? (y/n): ")) {
                    useLandingZone = true;
                    landingZoneArn = landingZones.get(0).arn();
                } else if (askYesNo("Do you want to use a different existing
 Landing Zone for this demo? (y/n): ")) {
                    useLandingZone = true;
                    System.out.println("Enter landing zone ARN: ");
                    landingZoneArn = scanner.nextLine().trim();
                } else {
                    System.out.println("Proceeding without a landing zone.");
                    useLandingZone = false;
                    waitForInputToContinue(scanner);
                }
            }

            // ------------------------------
            // Setup Organization + Sandbox OU
            // ------------------------------
            if (useLandingZone) {
                System.out.println("Using landing zone ARN: " + landingZoneArn);
```

```
                ControlTowerActions.OrgSetupResult result =
                        actions.setupOrganizationAsync().join();

                ouArn = result.sandboxOuArn();
                ouId  = result.sandboxOuArn();

                System.out.println("Organization ID: " + result.orgId());
                System.out.println("Using Sandbox OU ARN: " + ouArn);
            }

            // -----------------------------
            // Step 2: Baselines
            // -----------------------------
            System.out.println(DASHES);
            System.out.println("Step 2: Listing available baselines...");
            System.out.println("""
In this step, the program lists available AWS Control Tower baselines and may
 perform
baseline-related operations (enable, disable, reset) if requested.

NOTE:
AWS Control Tower enforces governance through baselines and mandatory controls
(guardrails). Mandatory controls are required for landing zone governance and may
restrict certain operations depending on the account, region, or organizational
 policy.

For more information, see:
- Types of baselines in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/userguide/types-of-
baselines.html
- Mandatory controls (guardrails) in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/controlreference/mandatory-
controls.html
- Baseline API examples:
  https://docs.aws.amazon.com/controltower/latest/userguide/baseline-api-
examples.html
""");



            waitForInputToContinue(scanner);
            List<BaselineSummary> baselines =
                    actions.listBaselinesAsync().join();
```

```
            BaselineSummary controlTowerBaseline = null;
            for (BaselineSummary b : baselines) {
                System.out.println("Baseline: " + b.name());
                System.out.println("  ARN: " + b.arn());
                if ("AWSControlTowerBaseline".equals(b.name())) {
                    controlTowerBaseline = b;
                }
            }

            waitForInputToContinue(scanner);

            if (useLandingZone && controlTowerBaseline != null) {

                System.out.println("\nListing enabled baselines:");
                List<EnabledBaselineSummary> enabledBaselines =
                        actions.listEnabledBaselinesAsync().join();

                String enabledBaselineArn = null;
                for (EnabledBaselineSummary eb : enabledBaselines) {
                    System.out.println("Checking enabled baseline ARN: " +
 eb.arn());
                    if
 (eb.baselineIdentifier().equals(controlTowerBaseline.arn())) {
                        enabledBaselineArn = eb.arn(); // correct enabled ARN for
 this baseline
                        break; // stop after finding the matching one
                    }
                }

                if (enabledBaselineArn == null) {
                    System.out.println("No enabled baseline found for " +
 controlTowerBaseline.arn());
                } else {
                    System.out.println("Selected enabled baseline ARN for reset/
disable: " + enabledBaselineArn);
                }


                // Enable the Baseline
                if (askYesNo("Do you want to enable the Control Tower Baseline?
 (y/n): ")) {
                    System.out.println("\nEnabling Control Tower Baseline...");
```

```
                    String baselineId = controlTowerBaseline.arn();
                    String enabledBaselineId =
                            actions.enableBaselineAsync(
                                    ouArn,                  // targetIdentifier #
the OU or account ARN
                                    baselineId,             // baselineIdentifier
# the Control Tower baseline ARN
                                    "5.0"                   // baselineVersion #
version string
                            ).join();


                    System.out.println("Enabled baseline operation ID: " +
enabledBaselineId);
                    if (enabledBaselineId == null) {
                       enabledBaselineId = enabledBaselineArn;
                    }

                    // Reset the Baseline
                    if (askYesNo("Do you want to reset the Control Tower
Baseline? (y/n): ")) {
                        String operationId =

actions.resetEnabledBaselineAsync(enabledBaselineId).join();
                        System.out.println("Reset baseline operation ID: " +
operationId);
                    }

                    if (askYesNo("Do you want to disable the Control Tower
Baseline? (y/n): ")) {
                        String operationId =

actions.disableBaselineAsync(enabledBaselineId).join();
                        System.out.println("Disabled baseline operation ID: " +
operationId);

                        System.out.println("Now we will re-enable the baseline
and wait 1 minute before making the call...");
                        try {
                            Thread.sleep(Duration.ofMinutes(1).toMillis());
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                            System.out.println("Wait interrupted");
                        }
```

```
                        String reEnabledBaselineId = actions.enableBaselineAsync(
                                ouArn,
                                baselineId,  // reuse baseline definition ARN
                                "5.0"
                        ).join();

                        System.out.println("Re-enabled baseline operation ID: " +
reEnabledBaselineId);
                    }
                }
            }

            // ----------------------------
            // Step 3: Controls
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("Step 3: Managing Controls:");
            waitForInputToContinue(scanner);

            List<ControlSummary> controls =
                    actions.listControlsAsync().join();

            System.out.println("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.min(5, controls.size()); i++) {
                ControlSummary c = controls.get(i);
                System.out.println("%d. %s - %s".formatted(i + 1, c.name(),
c.arn()));
            }

            if (useLandingZone) {
                waitForInputToContinue(scanner);

                List<EnabledControlSummary> enabledControls =
                        actions.listEnabledControlsAsync(ouArn).join();

                System.out.println("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.size(); i++) {
                    System.out.println("%d. %s".formatted(i + 1,
enabledControls.get(i).controlIdentifier()));
                }

                String controlArnToEnable = null;
                for (ControlSummary control : controls) {
                    boolean enabled = enabledControls.stream()
```

```
                            .anyMatch(ec ->
ec.controlIdentifier().equals(control.arn()));
                    if (!enabled) {
                        controlArnToEnable = control.arn();
                        break;
                    }
                }

                waitForInputToContinue(scanner);
                if (controlArnToEnable != null &&
                        askYesNo("Do you want to enable the control " +
controlArnToEnable + "? (y/n): ")) {

                    String operationId =
                            actions.enableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Enabled control with operation ID: " +
operationId);
                }

                waitForInputToContinue(scanner);

                if (controlArnToEnable != null &&
                        askYesNo("Do you want to disable the control? (y/n): ")
{

                    String operationId =
                            actions.disableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Disable operation ID: " + operationId);
                }
            }
            System.out.println("\nThis concludes the example scenario.");
            System.out.println("Thanks for watching!");
            System.out.println(DASHES);

        } catch (CompletionException e) {
            Throwable cause = e.getCause() != null ? e.getCause() : e;
            System.out.println("Scenario failed: " + cause.getMessage());
            throw e; // bubble up for tests / callers
        } catch (Exception e) {
```

```
            System.out.println("Unexpected error running scenario: " +
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    private static void waitForInputToContinue(Scanner sc) {
        System.out.println("\nEnter 'c' then <ENTER> to continue:");
        while (true) {
            String input = sc.nextLine();
            if ("c".equalsIgnoreCase(input.trim())) {
                System.out.println("Continuing...");
                break;
            }
        }
    }
}

public class ControlTowerActions {
    private static ControlCatalogAsyncClient controlCatalogAsyncClient;
    private static ControlTowerAsyncClient controlTowerAsyncClient;
    private static OrganizationsAsyncClient orgAsyncClient;

    private static OrganizationsAsyncClient getAsyncOrgClient() {
        if (orgAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(50)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .build();

            orgAsyncClient = OrganizationsAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return orgAsyncClient;
```

```
    }

    private static ControlCatalogAsyncClient getAsyncCatClient() {
        if (controlCatalogAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            controlCatalogAsyncClient = ControlCatalogAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return controlCatalogAsyncClient;
    }

    private static ControlTowerAsyncClient getAsyncClient() {
        if (controlTowerAsyncClient == null) {

            SdkAsyncHttpClient httpClient =
                    AwsCrtAsyncHttpClient.builder()
                            .maxConcurrency(100)
                            .connectionTimeout(Duration.ofSeconds(60))
                            .build();

            ClientOverrideConfiguration overrideConfig =
                    ClientOverrideConfiguration.builder()
                            .apiCallTimeout(Duration.ofMinutes(2))
                            .apiCallAttemptTimeout(Duration.ofSeconds(90))
                            .retryStrategy(RetryMode.STANDARD)
                            .build();

            controlTowerAsyncClient =
                    ControlTowerAsyncClient.builder()
```

```
                            .httpClient(httpClient)
                            .overrideConfiguration(overrideConfig)
                            .build();
        }

        return controlTowerAsyncClient;
    }

    public record OrgSetupResult(String orgId, String sandboxOuArn) {
    }

    public CompletableFuture<OrgSetupResult> setupOrganizationAsync() {
        System.out.println("Starting organization setup…");

        OrganizationsAsyncClient client = getAsyncOrgClient();

        // Step 1: Describe or create organization
        CompletableFuture<Organization> orgFuture = client.describeOrganization()
                .thenApply(desc -> {
                    System.out.println("Organization exists: " +
desc.organization().id());
                    return desc.organization();
                })
                .exceptionallyCompose(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof AwsServiceException awsEx &&

"AWSOrganizationsNotInUseException".equals(awsEx.awsErrorDetails().errorCode()))
{
                        System.out.println("No organization found. Creating
one…");
                        return
client.createOrganization(CreateOrganizationRequest.builder()
                                        .featureSet(OrganizationFeatureSet.ALL)
                                        .build())
                                .thenApply(createResp -> {
                                    System.out.println("Created organization: {}"
+ createResp.organization().id());
                                    return createResp.organization();
                                });
                    }
                    return CompletableFuture.failedFuture(
                            new CompletionException("Failed to describe or create
organization", cause)
```

```
                    );
                });

        // Step 2: Locate Sandbox OU
        return orgFuture.thenCompose(org -> {
            String orgId = org.id();
            System.out.println("Organization ID: {}" + orgId);

            return client.listRoots()
                    .thenCompose(rootsResp -> {
                        if (rootsResp.roots().isEmpty()) {
                            return CompletableFuture.failedFuture(
                                    new RuntimeException("No root found in
organization")
                            );
                        }
                        String rootId = rootsResp.roots().get(0).id();

                        ListOrganizationalUnitsForParentRequest ouRequest =
                                ListOrganizationalUnitsForParentRequest.builder()
                                        .parentId(rootId)
                                        .build();

                        ListOrganizationalUnitsForParentPublisher paginator =

client.listOrganizationalUnitsForParentPaginator(ouRequest);

                        AtomicReference<String> sandboxOuArnRef = new
AtomicReference<>();
                        return paginator.subscribe(page -> {
                                    for (OrganizationalUnit ou :
page.organizationalUnits()) {
                                        if ("Sandbox".equals(ou.name())) {
                                            sandboxOuArnRef.set(ou.arn());
                                            System.out.println("Found Sandbox OU:
" + ou.id());
                                            break;
                                        }
                                    }
                                })
                                .thenApply(v -> {
                                    String sandboxArn = sandboxOuArnRef.get();
                                    if (sandboxArn == null) {
```

```
                                        System.out.println("Sandbox OU not
 found.");
                                    }
                                    return new OrgSetupResult(orgId, sandboxArn);
                                });
                    });
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
            System.out.println("Failed to setup organization: {}" +
 cause.getMessage());
            throw new CompletionException(cause);
        });
    }

    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request =
 ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
```

```
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing
zones: " + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
cause);
                });
    }


    /**
     * Lists all available baselines using pagination to retrieve complete
results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
```

```
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total:
 " + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
 %s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
 %s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
 %s".formatted(cause.getMessage()),
                                cause
                        );
                    }
```

```
                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }

    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
        System.out.println("Starting list enabled baselines paginator…");

        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();

        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {

                        response.enabledBaselines().forEach(baseline -> {
                            enabledBaselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no enabled
baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println(
                                "Successfully listed enabled baselines. Total: "
+
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
```

```
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled
baselines: %s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }


    /**
     * Asynchronously enables a baseline for the specified target if not already
enabled.
     *
     * @param targetIdentifier       The ARN of the target (OU or account).
     * @param baselineIdentifier     The baseline definition ARN to enable.
     * @param baselineVersion        The baseline version to enable.
     * @return A CompletableFuture containing the enabled baseline ARN, or null
if already enabled.
     */
```

```
    public CompletableFuture<String> enableBaselineAsync(
            String targetIdentifier,
            String baselineIdentifier,
            String baselineVersion
    ) {
        EnableBaselineRequest request = EnableBaselineRequest.builder()
                .baselineIdentifier(baselineIdentifier)
                .baselineVersion(baselineVersion)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableBaseline(request)
                .handle((resp, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ControlTowerException e) {
                            String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                            String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled
for this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN
synchronously
                            }

                            throw new RuntimeException("Error enabling baseline:
" + code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
```

```
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op =
opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED)
{
                                    return enabledBaselineArn;
                                } else if (status ==
BaselineOperationStatus.FAILED) {
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline
operation failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {

Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
```

```
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target
asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines())
{
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                &&
targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
    }


    /**
     * Disables a baseline for a specified target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {
```

```
        System.out.println("Starting disable of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        DisableBaselineRequest request = DisableBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().disableBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable baseline operation ID: " +
operationId);

                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status)
|| "FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
```

```
                                        System.out.println("Error checking
baseline operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }



    /**
     * Gets the status of a baseline operation.
     *
```

```
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
            String operationIdentifier) {

        GetBaselineOperationRequest request =
GetBaselineOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getBaselineOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null
                                ? exception.getCause()
                                : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Baseline operation not found: %s"
                                                .formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
```

```
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }


    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
 listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target "
 + targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
```

```
                    } else {
                        System.out.println("Page contained no enabled
controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +"
enabled controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to
work with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
%s".formatted(e.getMessage()), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
```

```
                    }

                    throw new CompletionException("Failed to list enabled
controls", cause);
                });
    }


    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started.
Operation ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);
```

```
                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ?
e.getMessage() : "";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for
this target");
                            return null;
                        }

                        if ("ValidationException".equals(errorCode)
                                && message.contains("enforced by default")) {
                            System.out.println("Control is enforced by default
and cannot be enabled manually.");
                            return null;
```

```
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS
Control Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }


    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
```

```
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started.
 Operation ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
 CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
 status: " + status);

                                        if (status ==
 ControlOperationStatus.SUCCEEDED
                                                || status ==
 ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
 TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // start polling immediately
```

```
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling:
" + e.getMessage());
                            return null;
                        }

                        throw new CompletionException(
                                "Error disabling control: " + e.getMessage(), e);
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error disabling control: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }



    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {
```

```
        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }
```

```
    /**
     * Lists all controls in the Control Tower control catalog.
     *
     * @return a list of controls
     * @throws SdkException if a service-specific error occurs
     */
    public CompletableFuture<List<ControlSummary>> listControlsAsync() {
        System.out.println("Starting list controls paginator…");

        ListControlsRequest request = ListControlsRequest.builder().build();
        ListControlsPublisher paginator =
 getAsyncCatClient().listControlsPaginator(request);
        List<ControlSummary> controls = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.controls() != null && !
response.controls().isEmpty()) {
                        response.controls().forEach(control -> {
                            controls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no controls.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved " +
 controls.size() +" controls."))
                .thenApply(v -> controls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof SdkException sdkEx) {
                        if (sdkEx.getMessage() != null &&
 sdkEx.getMessage().contains("AccessDeniedException")) {
                            throw new CompletionException(
                                    "Access denied when listing controls. Please
 ensure you have the necessary permissions.",
                                    sdkEx
                            );
                        } else {
                            throw new CompletionException(
                                    "SDK error listing controls:
 %s".formatted(sdkEx.getMessage()),
                                    sdkEx
```

```
                            );
                        }
                    }

                    throw new CompletionException("Failed to list controls",
cause);
                });
    }

    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting reset of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        ResetEnabledBaselineRequest request =
ResetEnabledBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().resetEnabledBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Reset enabled baseline operation ID: " +
operationId);

                    // Polling loop
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
```

```
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status)
|| "FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Baseline
operation finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking
baseline operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code="
+ errorCode + ", Message=" + errorMessage);
                        return null;
```

```
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
 sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline:
 " + cause.getMessage());
                    return null;
                });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS Control Tower recursos.

```
class ControlTowerScenario:
    IDENTITY_CENTER_BASELINE = "baseline/LN25R72TTG6IGPTQ"
    stack_name = ""

    def __init__(
        self, controltower_wrapper: ControlTowerWrapper, org_client: boto3.client
    ):
        """
        :param controltower_wrapper: An instance of the ControlTowerWrapper
 class.
        :param org_client: A Boto3 Organization client.
        """
        self.controltower_wrapper = controltower_wrapper
        self.org_client = org_client
        self.stack = None
        self.ou_id = None
        self.ou_arn = None
        self.account_id = None
        self.landing_zone_id = None
        self.use_landing_zone = False

    def run_scenario(self) -> None:
        print("-" * 88)
        print(
            "\tWelcome to the AWS Control Tower with ControlCatalog example
 scenario."
        )
        print("-" * 88)

        print(
```

```
            "This demo will walk you through working with AWS Control Tower for
 landing zones,\n"
            "managing baselines, and working with controls."
        )

        self.account_id = boto3.client("sts").get_caller_identity()["Account"]

        print(
            "Some demo operations require the use of a landing zone. "
            "\nYou can use an existing landing zone or opt out of these
 operations in the demo."
            "\nFor instructions on how to set up a landing zone, "
            "\nsee https://docs.aws.amazon.com/controltower/latest/userguide/
getting-started-from-console.html"
        )
        # List available landing zones
        landing_zones = self.controltower_wrapper.list_landing_zones()
        if landing_zones:
            print("\nAvailable Landing Zones:")
            for i, lz in enumerate(landing_zones, 1):
                print(f"{i} {lz['arn']}")

            # Ask if user wants to use the first landing zone in the list
            if q.ask(
                f"Do you want to use the first landing zone in the list
 ({landing_zones[0]['arn']})? (y/n) ",
                q.is_yesno,
            ):
                self.use_landing_zone = True
                self.landing_zone_id = landing_zones[0]["arn"]
                print(f"Using landing zone ID: {self.landing_zone_id})")
                # Set up organization and get Sandbox OU ID.
                sandbox_ou_id = self.setup_organization()
                # Store the OU ID for use in the CloudFormation template.
                self.ou_id = sandbox_ou_id
            elif q.ask(
                f"Do you want to use a different existing Landing Zone for this
 demo? (y/n) ",
                q.is_yesno,
            ):
                self.use_landing_zone = True
                self.landing_zone_id = q.ask("Enter landing zone id: ",
 q.non_empty)
                # Set up organization and get Sandbox OU ID.
```

```
                sandbox_ou_id = self.setup_organization()
                # Store the OU ID for use in the CloudFormation template.
                self.ou_id = sandbox_ou_id

        # List and Enable Baseline.
        print("\nManaging Baselines:")
        control_tower_baseline = None
        identity_center_baseline = None
        baselines = self.controltower_wrapper.list_baselines()
        print("\nListing available Baselines:")
        for baseline in baselines:
            if baseline["name"] == "AWSControlTowerBaseline":
                control_tower_baseline = baseline
            print(f"{baseline['name']}")

        if self.use_landing_zone:
            print("\nListing enabled baselines:")
            enabled_baselines =
self.controltower_wrapper.list_enabled_baselines()
            for baseline in enabled_baselines:
                # If the Identity Center baseline is enabled, the identifier must
be used for other baselines.
                if self.IDENTITY_CENTER_BASELINE in
baseline["baselineIdentifier"]:
                    identity_center_baseline = baseline
                print(f"{baseline['baselineIdentifier']}")

            if q.ask(
                f"Do you want to enable the Control Tower Baseline? (y/n) ",
                q.is_yesno,
            ):
                print("\nEnabling Control Tower Baseline.")
                ic_baseline_arn = (
                    identity_center_baseline["arn"]
                    if identity_center_baseline
                    else None
                )
                baseline_arn = self.controltower_wrapper.enable_baseline(
                    self.ou_arn, ic_baseline_arn, control_tower_baseline["arn"],
"5.0"
                )
                if baseline_arn:
                    print(f"Enabled baseline ARN: {baseline_arn}")
                else:
```

```
                    # Find the enabled baseline so we can reset it.
                    for enabled_baseline in enabled_baselines:
                        if (
                            enabled_baseline["baselineIdentifier"]
                            == control_tower_baseline["arn"]
                        ):
                            baseline_arn = enabled_baseline["arn"]
                    if baseline_arn:
                        print("No change, the selected baseline was already
enabled.")

                if baseline_arn and q.ask(
                    f"Do you want to reset the Control Tower Baseline? (y/n) ",
                    q.is_yesno,
                ):
                    print(f"\nResetting Control Tower Baseline. {baseline_arn}")
                    operation_id =
self.controltower_wrapper.reset_enabled_baseline(
                        baseline_arn
                    )
                    print(f"\nReset baseline operation id {operation_id}.")

                if baseline_arn and q.ask(
                    f"Do you want to disable the Control Tower Baseline? (y/n) ",
                    q.is_yesno,
                ):
                    print(f"Disabling baseline ARN: {baseline_arn}")
                    operation_id = self.controltower_wrapper.disable_baseline(
                        baseline_arn
                    )
                    print(f"\nDisabled baseline operation id {operation_id}.")

                    # Re-enable the baseline for the next step.
                    print("\nEnabling Control Tower Baseline.")
                    self.controltower_wrapper.enable_baseline(
                        self.ou_arn,
                        ic_baseline_arn,
                        control_tower_baseline["arn"],
                        "5.0",
                    )

        # List and Enable Controls.
        print("\nManaging Controls:")
        controls = self.controltower_wrapper.list_controls()
```

```
        print("\nListing first 5 available Controls:")
        for i, control in enumerate(controls[:5], 1):
            print(f"{i}. {control['Name']} - {control['Arn']}")

        if self.use_landing_zone:
            target_ou = self.ou_arn
            enabled_controls = self.controltower_wrapper.list_enabled_controls(
                target_ou
            )
            print("\nListing enabled controls:")
            for i, control in enumerate(enabled_controls, 1):
                print(f"{i}. {control['controlIdentifier']}")

            # Enable first non-enabled control as an example.
            enabled_control_arns = [control["arn"] for control in
enabled_controls]
            control_arn = next(
                control["Arn"]
                for control in controls
                if control["Arn"] not in enabled_control_arns
            )

            if control_arn and q.ask(
                f"Do you want to enable the control {control_arn}? (y/n) ",
                q.is_yesno,
            ):
                print(f"\nEnabling control: {control_arn}")
                operation_id = self.controltower_wrapper.enable_control(
                    control_arn, target_ou
                )

                if operation_id:
                    print(f"Enabled control with operation id {operation_id}")

            if control_arn and q.ask(
                f"Do you want to disable the control? (y/n) ",
                q.is_yesno,
            ):
                print("\nDisabling the control...")
                operation_id = self.controltower_wrapper.disable_control(
                    control_arn, target_ou
                )
                print(f"Disable operation ID: {operation_id}")
```

```
        print("\nThis concludes the example scenario.")

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

    def setup_organization(self):
        """
        Checks if the current account is part of an organization and creates one
if needed.
        Also ensures a Sandbox OU exists and returns its ID.

        :return: The ID of the Sandbox OU
        """
        print("\nChecking organization status...")

        try:
            # Check if account is part of an organization
            org_response = self.org_client.describe_organization()
            org_id = org_response["Organization"]["Id"]
            print(f"Account is part of organization: {org_id}")

        except ClientError as error:
            if error.response["Error"]["Code"] ==
"AWSOrganizationsNotInUseException":
                print("No organization found. Creating a new organization...")
                try:
                    create_response = self.org_client.create_organization(
                        FeatureSet="ALL"
                    )
                    org_id = create_response["Organization"]["Id"]
                    print(f"Created new organization: {org_id}")

                    # Wait for organization to be available.
                    waiter = self.org_client.get_waiter("organization_active")
                    waiter.wait(
                        Organization=org_id,
                        WaiterConfig={"Delay": 5, "MaxAttempts": 12},
                    )

                except ClientError as create_error:
                    logger.error(
                        "Couldn't create organization. Here's why: %s: %s",
                        create_error.response["Error"]["Code"],
                        create_error.response["Error"]["Message"],
```

```
                )
                raise
        else:
            logger.error(
                "Couldn't describe organization. Here's why: %s: %s",
                error.response["Error"]["Code"],
                error.response["Error"]["Message"],
            )
            raise

    # Look for Sandbox OU.
    sandbox_ou_id = None
    paginator = self.org_client.get_paginator(
        "list_organizational_units_for_parent"
    )

    try:
        # Get root ID first.
        roots = self.org_client.list_roots()["Roots"]
        if not roots:
            raise ValueError("No root found in organization")
        root_id = roots[0]["Id"]

        # Search for existing Sandbox OU.
        print("Checking for Sandbox OU...")
        for page in paginator.paginate(ParentId=root_id):
            for ou in page["OrganizationalUnits"]:
                if ou["Name"] == "Sandbox":
                    sandbox_ou_id = ou["Id"]
                    self.ou_arn = ou["Arn"]
                    print(f"Found existing Sandbox OU: {sandbox_ou_id}")
                    break
            if sandbox_ou_id:
                break

        # Create Sandbox OU if it doesn't exist.
        if not sandbox_ou_id:
            print("Creating Sandbox OU...")
            create_ou_response = self.org_client.create_organizational_unit(
                ParentId=root_id, Name="Sandbox"
            )
            sandbox_ou_id = create_ou_response["OrganizationalUnit"]["Id"]
            print(f"Created new Sandbox OU: {sandbox_ou_id}")
```

```
                # Wait for OU to be available.
                waiter = self.org_client.get_waiter("organizational_unit_active")
                waiter.wait(
                    OrganizationalUnitId=sandbox_ou_id,
                    WaiterConfig={"Delay": 5, "MaxAttempts": 12},
                )

        except ClientError as error:
            logger.error(
                "Couldn't set up Sandbox OU. Here's why: %s: %s",
                error.response["Error"]["Code"],
                error.response["Error"]["Message"],
            )
            raise

        return sandbox_ou_id


if __name__ == "__main__":
    try:
        org = boto3.client("organizations")
        control_tower_wrapper = ControlTowerWrapper.from_client()

        scenario = ControlTowerScenario(control_tower_wrapper, org)
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the scenario.")



class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client
```

```
    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_baselines(self):
        """
        Lists all baselines.

        :return: List of baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_baselines")
            baselines = []
            for page in paginator.paginate():
                baselines.extend(page["baselines"])
            return baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def enable_baseline(
        self,
        target_identifier: str,
        identity_center_baseline: str,
        baseline_identifier: str,
        baseline_version: str,
    ):
        """
```

```
        Enables a baseline for the specified target if it's not already enabled.

        :param target_identifier: The ARN of the target.
        :param baseline_identifier: The identifier of baseline to enable.
        :param identity_center_baseline: The identifier of identity center
baseline if it is enabled.
        :param baseline_version: The version of baseline to enable.
        :return: The enabled baseline ARN or None if already enabled.
        :raises ClientError: If enabling the baseline fails for reasons other
than it being already enabled.
        """
        try:
            # Only include parameters if identity_center_baseline is not empty
            parameters = []
            if identity_center_baseline:
                parameters = [
                    {
                        "key": "IdentityCenterEnabledBaselineArn",
                        "value": identity_center_baseline,
                    }
                ]

            response = self.controltower_client.enable_baseline(
                baselineIdentifier=baseline_identifier,
                baselineVersion=baseline_version,
                targetIdentifier=target_identifier,
                parameters=parameters,
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["arn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                if "already enabled" in err.response["Error"]["Message"]:
                    print("Baseline is already enabled for this target")
                else:
                    print(
```

```
                        "Unable to enable baseline due to validation exception:
%s: %s",
                        err.response["Error"]["Code"],
                        err.response["Error"]["Message"],
                    )
            logger.error(
                "Couldn't enable baseline. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            return None


    def list_controls(self):
        """
        Lists all controls in the Control Tower control catalog.

        :return: List of controls.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controlcatalog_client.get_paginator("list_controls")
            controls = []
            for page in paginator.paginate():
                controls.extend(page["Controls"])
            return controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def enable_control(self, control_arn: str, target_identifier: str):
        """
```

```
        Enables a control for a specified target.

        :param control_arn: The ARN of the control to enable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If enabling the control fails.
        """
        try:
            print(control_arn)
            print(target_identifier)
            response = self.controltower_client.enable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id

        except ClientError as err:
            if (
                err.response["Error"]["Code"] == "ValidationException"
                and "already enabled" in err.response["Error"]["Message"]
            ):
                logger.info("Control is already enabled for this target")
                return None
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with
controls.")
                return None
            logger.error(
                "Couldn't enable control. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise


    def get_control_operation(self, operation_id: str):
        """
        Gets the status of a control operation.

        :param operation_id: The ID of the control operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_control_operation(
                operationIdentifier=operation_id
            )
            return response["controlOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get control operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def get_baseline_operation(self, operation_id: str):
        """
        Gets the status of a baseline operation.

        :param operation_id: The ID of the baseline operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_baseline_operation(
                operationIdentifier=operation_id
            )
            return response["baselineOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
```

```
            else:
                logger.error(
                    "Couldn't get baseline operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def disable_control(self, control_arn: str, target_identifier: str):
        """
        Disables a control for a specified target.

        :param control_arn: The ARN of the control to disable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If disabling the control fails.
        """
        try:
            response = self.controltower_client.disable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Control not found.")
            else:
                logger.error(
                    "Couldn't disable control. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
    def list_landing_zones(self):
        """
        Lists all landing zones.

        :return: List of landing zones.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
self.controltower_client.get_paginator("list_landing_zones")
            landing_zones = []
            for page in paginator.paginate():
                landing_zones.extend(page["landingZones"])
            return landing_zones

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list landing zones. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def list_enabled_baselines(self):
        """
        Lists all enabled baselines.

        :return: List of enabled baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_baselines")
            enabled_baselines = []
            for page in paginator.paginate():
                enabled_baselines.extend(page["enabledBaselines"])
            return enabled_baselines
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't list enabled baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def reset_enabled_baseline(self, enabled_baseline_identifier: str):
        """
        Resets an enabled baseline for a specific target.

        :param enabled_baseline_identifier: The identifier of the enabled
baseline to reset.
        :return: The operation ID.
        :raises ClientError: If resetting the baseline fails.
        """
        try:
            response = self.controltower_client.reset_enabled_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )
            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't reset enabled baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
    def disable_baseline(self, enabled_baseline_identifier: str):
        """
        Disables a baseline for a specific target and waits for the operation to
 complete.

        :param enabled_baseline_identifier: The identifier of the baseline to
 disable.
        :return: The operation ID.
        :raises ClientError: If disabling the baseline fails.
        """
        try:
            response = self.controltower_client.disable_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["operationIdentifier"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                print(
                    f"Conflict disabling baseline: {err.response['Error']
['Message']}. Skipping disable step."
                )
                return None
            else:
                logger.error(
                    "Couldn't disable baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise


    def list_enabled_controls(self, target_identifier: str):
        """
```

```
        Lists all enabled controls for a specific target.

        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: List of enabled controls.
        :raises ClientError: If the listing operation fails.
        """
        enabled_controls = []
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_controls")

            for page in paginator.paginate(targetIdentifier=target_identifier):
                enabled_controls.extend(page["enabledControls"])
            return enabled_controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
                return enabled_controls
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with
controls.")
                return enabled_controls
            else:
                logger.error(
                    "Couldn't list enabled controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Ações para AWS Control Tower usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Control Tower ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Control Tower.

Exemplos

- Use DisableBaselinecom um AWS SDK ou CLI
- Use DisableControlcom um AWS SDK ou CLI
- Use EnableBaselinecom um AWS SDK ou CLI
- Use EnableControlcom um AWS SDK ou CLI
- Use GetBaselineOperationcom um AWS SDK ou CLI
- Use GetControlOperationcom um AWS SDK ou CLI
- Use ListBaselinescom um AWS SDK ou CLI

- Use ListEnabledBaselinescom um AWS SDK ou CLI
- Use ListEnabledControlscom um AWS SDK ou CLI
- Use ListLandingZonescom um AWS SDK ou CLI
- Use ResetEnabledBaselinecom um AWS SDK ou CLI

Use **`DisableBaseline`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DisableBaseline`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Disable a baseline for a specific target and wait for the operation to
complete.
   /// </summary>
   /// <param name="enabledBaselineIdentifier">The identifier of the baseline to
disable.</param>
   /// <returns>The operation ID or null if there was a conflict.</returns>
   public async Task<string?> DisableBaselineAsync(string
enabledBaselineIdentifier)
   {
       try
       {
           var request = new DisableBaselineRequest
           {
               EnabledBaselineIdentifier = enabledBaselineIdentifier
```

```
            };

            var response = await
_controlTowerService.DisableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (ConflictException ex)
        {
            Console.WriteLine($"Conflict disabling baseline: {ex.Message}.
Skipping disable step.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DisableBaselinea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Desativar uma linha de base do Control Tower

O `disable-baseline` exemplo a seguir desativa uma linha de base do AWS Control Tower.

```
aws controltower disable-baseline \
    --enabled-baseline-identifier arn:aws:controltower:us-
east-1:123456789012:enabledbaseline/XOM12BEL4YD578CQ2
```

Saída:

```
{
    "operationIdentifier": "b33486d7-5396-4ad0-9eae-3a57969fe8cd"
}
```

Para obter mais informações, consulte Tipos de linhas de base no Guia do usuário do AWS Control Tower.

- Para obter detalhes da API, consulte DisableBaselineem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Disables a baseline for a specified target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
 disable
     * @return the operation identifier
```

```
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting disable of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        DisableBaselineRequest request = DisableBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().disableBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable baseline operation ID: " +
operationId);

                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status)
|| "FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
```

```
                                                CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                        .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking
baseline operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }
```

- Para obter detalhes da API, consulte DisableBaselinea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def disable_baseline(self, enabled_baseline_identifier: str):
        """
```

```
        Disables a baseline for a specific target and waits for the operation to
 complete.

        :param enabled_baseline_identifier: The identifier of the baseline to
 disable.
        :return: The operation ID.
        :raises ClientError: If disabling the baseline fails.
        """
        try:
            response = self.controltower_client.disable_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["operationIdentifier"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                print(
                    f"Conflict disabling baseline: {err.response['Error']
['Message']}. Skipping disable step."
                )
                return None
            else:
                logger.error(
                    "Couldn't disable baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a DisableBaselineReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Disable the baseline
        DATA(lo_output) = io_ctt->disablebaseline(
          iv_enabledbaselineidentifier = iv_enabled_baseline_identifier
        ).

        DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

        " Wait for operation to complete
        DATA lv_status TYPE /aws1/cttbaselineopstatus.
        DO 100 TIMES.
          lv_status = get_baseline_operation(
            io_ctt = io_ctt
            iv_operation_id = lv_operation_id
          ).

          DATA(lv_msg) = |Baseline operation status: { lv_status }|.
          MESSAGE lv_msg TYPE 'I'.

          IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
            EXIT.
          ENDIF.

          " Wait 30 seconds
          WAIT UP TO 30 SECONDS.
        ENDDO.

        ov_operation_id = lv_operation_id.
        MESSAGE 'Baseline disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_cttconflictexception INTO DATA(lo_conflict).
        " Log conflict but don't fail - return empty operation ID
```

```
        DATA(lv_msg2) = |Conflict disabling baseline: { lo_conflict-
>get_text( ) }. Skipping disable step.|.
        MESSAGE lv_msg2 TYPE 'I'.
        CLEAR ov_operation_id.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DisableBaselinereferência da API AWS SDK for SAP ABAP.

Use **`DisableControl`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DisableControl`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Disable a control for a specified target.
   /// </summary>
   /// <param name="controlArn">The ARN of the control to disable.</param>
   /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
   /// <returns>The operation ID.</returns>
   public async Task<string> DisableControlAsync(string controlArn, string
targetIdentifier)
   {
       try
```

```
        {
            var request = new DisableControlRequest
            {
                ControlIdentifier = controlArn,
                TargetIdentifier = targetIdentifier
            };

            var response = await
_controlTowerService.DisableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Control not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DisableControla Referência AWS SDK para .NET da API.

CLI

AWS CLI

Desativar um controle do Control Tower

O `disable-control` exemplo a seguir desativa um AWS controle ativado pela Control Tower.

```
aws controltower disable-control \
    --control-
identifier arn:aws:controlcatalog:::control/497wrm2xnk1wxlf4obrxxxxxx \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryxxxxx/ou-
oqxx-i5wnxxxx
```

Saída:

```
{
    "operationIdentifier": "b8f0dxxx-08xx-43xx-a2xx-568e9922xxxx"
}
```

Para obter mais informações, consulte Sobre os controles na AWS Control Tower no Guia do usuário da AWS Control Tower.

- Para obter detalhes da API, consulte DisableControlem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
```

```
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started.
Operation ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
```

```
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                            }
                        };

                        // start polling immediately
                        poller.run();

                        return resultFuture;
                    })
                    .exceptionally(ex -> {
                        Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                // SPEC: notify user and continue
                                System.out.println("Control not found for disabling:
" + e.getMessage());
                                return null;
                            }

                            throw new CompletionException(
                                    "Error disabling control: " + e.getMessage(), e);
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error disabling control: " +
cause.getMessage(), cause);
                        }

                        throw new CompletionException(
                                "Failed to disable control", cause);
                    });
    }
```

- Para obter detalhes da API, consulte DisableControla Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def disable_control(self, control_arn: str, target_identifier: str):
        """
        Disables a control for a specified target.

        :param control_arn: The ARN of the control to disable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If disabling the control fails.
        """
```

```
        try:
            response = self.controltower_client.disable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Control not found.")
            else:
                logger.error(
                    "Couldn't disable control. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DisableControlReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Disable the control
```

```
    DATA(lo_output) = io_ctt->disablecontrol(
      iv_controlidentifier = iv_control_arn
      iv_targetidentifier  = iv_target_identifier
    ).

    DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttcontrolopstatus.
    DO 100 TIMES.
      lv_status = get_control_operation(
        io_ctt = io_ctt
        iv_operation_id = lv_operation_id
      ).

      DATA(lv_msg) = |Control operation status: { lv_status }|.
      MESSAGE lv_msg TYPE 'I'.

      IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
        EXIT.
      ENDIF.

      " Wait 30 seconds
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Control disabled successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a DisableControlreferência da API AWS SDK for SAP ABAP.

Use **`EnableBaseline`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `EnableBaseline`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Enable a baseline for the specified target.
    /// </summary>
    /// <param name="targetIdentifier">The ARN of the target.</param>
    /// <param name="baselineIdentifier">The identifier of baseline to enable.</
param>
    /// <param name="baselineVersion">The version of baseline to enable.</param>
    /// <param name="identityCenterBaseline">The identifier of identity center
 baseline if it is enabled.</param>
    /// <returns>The enabled baseline ARN or null.</returns>
    public async Task<string?> EnableBaselineAsync(string targetIdentifier,
 string baselineIdentifier, string baselineVersion, string
 identityCenterBaseline)
    {
        try
        {
            var parameters = new List<EnabledBaselineParameter>();
            if (!string.IsNullOrEmpty(identityCenterBaseline))
            {
                parameters.Add(
                    new EnabledBaselineParameter
                    {
                        Key = "IdentityCenterEnabledBaselineArn",
                        Value = identityCenterBaseline
                    });
            }
            var request = new EnableBaselineRequest
            {
                BaselineIdentifier = baselineIdentifier,
                BaselineVersion = baselineVersion,
                TargetIdentifier = targetIdentifier,
                Parameters = parameters
```

```
            };

            var response = await
_controlTowerService.EnableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return response.Arn;
        }
        catch (ValidationException ex)
        {
            if (ex.Message.Contains("already enabled"))
                Console.WriteLine("Baseline is already enabled for this target");
            else { Console.WriteLine(ex.Message); }
            // Write the message and return null if baseline cannot be enabled.
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte EnableBaselinea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: Habilitar uma linha de base do Control Tower que está desativada

O `enable-baseline` exemplo a seguir habilita uma linha de base do AWS Control Tower se a linha de base estiver `IdentityCenterBaseline` desativada.

```
aws controltower enable-baseline \
    --baseline-identifier arn:aws:controltower:us-
east-1::baseline/17BSJV3IGJ2QSGA2 \
    --baseline-version 4.0 \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryixxxx/ou-
oq9f-i5wnxxxx
```

Saída:

```
{
    "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XOM12BEL4YD578CQ2",
    "operationIdentifier": "51e190ac-8a37-4f6d-b63c-fb5104b5db38"
}
```

Para obter mais informações, consulte Tipos de linhas de base no Guia do usuário do AWS Control Tower.

Exemplo 2: Habilitar uma linha de base do Control Tower que está ativada

O `enable-baseline` exemplo a seguir habilita uma linha de base do AWS Control Tower se a linha de base estiver `IdentityCenterBaseline` ativada.

```
aws controltower enable-baseline \
    --baseline-identifier arn:aws:controltower:us-
east-1::baseline/17BSJV3IGJ2QSGA2 \
    --baseline-version 4.0 \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryixxxx/ou-
oqxx-i5wnxxxx \
    --parameters
 '[{"key":"IdentityCenterEnabledBaselineArn","value":"arn:aws:controltower:us-
east-1:123456789012:enabledbaseline/XAJNZNCBC1I386C7B"}]'
```

Saída:

```
{
    "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XOM12BEL4YD578CQ2",
    "operationIdentifier": "51e190ac-8a37-4f6d-b63c-fb5104b5db38"
}
```

Para obter mais informações, consulte Tipos de linhas de base no Guia do usuário do AWS Control Tower.

- Para obter detalhes da API, consulte EnableBaselineem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously enables a baseline for the specified target if not already
enabled.
    *
    * @param targetIdentifier       The ARN of the target (OU or account).
    * @param baselineIdentifier     The baseline definition ARN to enable.
    * @param baselineVersion        The baseline version to enable.
    * @return A CompletableFuture containing the enabled baseline ARN, or null
if already enabled.
    */
   public CompletableFuture<String> enableBaselineAsync(
           String targetIdentifier,
           String baselineIdentifier,
           String baselineVersion
   ) {
       EnableBaselineRequest request = EnableBaselineRequest.builder()
```

```
                .baselineIdentifier(baselineIdentifier)
                .baselineVersion(baselineVersion)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableBaseline(request)
                .handle((resp, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ControlTowerException e) {
                            String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                            String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled
for this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN
synchronously
                            }

                            throw new RuntimeException("Error enabling baseline:
" + code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);
```

```
                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op =
opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED)
{
                                    return enabledBaselineArn;
                                } else if (status ==
BaselineOperationStatus.FAILED) {
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline
operation failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {

Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
```

```
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target
asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines())
{
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                &&
targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
    }
```

- Para obter detalhes da API, consulte EnableBaselinea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def enable_baseline(
        self,
        target_identifier: str,
        identity_center_baseline: str,
        baseline_identifier: str,
        baseline_version: str,
    ):
        """
        Enables a baseline for the specified target if it's not already enabled.

        :param target_identifier: The ARN of the target.
        :param baseline_identifier: The identifier of baseline to enable.
        :param identity_center_baseline: The identifier of identity center
 baseline if it is enabled.
        :param baseline_version: The version of baseline to enable.
        :return: The enabled baseline ARN or None if already enabled.
        :raises ClientError: If enabling the baseline fails for reasons other
 than it being already enabled.
        """
        try:
            # Only include parameters if identity_center_baseline is not empty
            parameters = []
            if identity_center_baseline:
```

```
                parameters = [
                    {
                        "key": "IdentityCenterEnabledBaselineArn",
                        "value": identity_center_baseline,
                    }
                ]

            response = self.controltower_client.enable_baseline(
                baselineIdentifier=baseline_identifier,
                baselineVersion=baseline_version,
                targetIdentifier=target_identifier,
                parameters=parameters,
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["arn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                if "already enabled" in err.response["Error"]["Message"]:
                    print("Baseline is already enabled for this target")
                else:
                    print(
                        "Unable to enable baseline due to validation exception:
%s: %s",
                        err.response["Error"]["Code"],
                        err.response["Error"]["Message"],
                    )
            logger.error(
                "Couldn't enable baseline. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            return None
```

- Para obter detalhes da API, consulte a EnableBaselineReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Prepare parameters for enabling baseline
    DATA lt_parameters TYPE /aws1/
cl_cttenbdbaselineparam=>tt_enabledbaselineparameters.

    " Add Identity Center baseline parameter if provided
    IF iv_identity_center_baseline IS NOT INITIAL.
      " Create a JSON document with the baseline ARN value
      DATA(lv_json) = |\{ "IdentityCenterEnabledBaselineArn":
 "{ iv_identity_center_baseline }" \}|.
      DATA(lo_param) = NEW /aws1/cl_cttenbdbaselineparam(
        iv_key = 'IdentityCenterEnabledBaselineArn'
        io_value = /aws1/cl_rt_document=>from_json_str( lv_json )
      ).
      APPEND lo_param TO lt_parameters.
    ENDIF.

    " Enable the baseline
    DATA(lo_output) = io_ctt->enablebaseline(
      iv_baselineidentifier = iv_baseline_identifier
      iv_baselineversion    = iv_baseline_version
      iv_targetidentifier   = iv_target_identifier
      it_parameters         = lt_parameters
    ).

    DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttbaselineopstatus.
```

```
    DO 100 TIMES.
      lv_status = get_baseline_operation(
        io_ctt = io_ctt
        iv_operation_id = lv_operation_id
      ).

      DATA(lv_msg) = |Baseline operation status: { lv_status }|.
      MESSAGE lv_msg TYPE 'I'.

      IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
        EXIT.
      ENDIF.

      " Wait 30 seconds
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_enabled_baseline_arn = lo_output->get_arn( ).
    MESSAGE 'Baseline enabled successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a EnableBaselinereferência da API AWS SDK for SAP ABAP.

Use **`EnableControl`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `EnableControl`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Enable a control for a specified target.
    /// </summary>
    /// <param name="controlArn">The ARN of the control to enable.</param>
    /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
    /// <returns>The operation ID or null if already enabled.</returns>
    public async Task<string?> EnableControlAsync(string controlArn, string
targetIdentifier)
    {
        try
        {
            Console.WriteLine(controlArn);
            Console.WriteLine(targetIdentifier);

            var request = new EnableControlRequest
            {
                ControlIdentifier = controlArn,
                TargetIdentifier = targetIdentifier
            };

            var response = await
_controlTowerService.EnableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ValidationException ex) when
(ex.Message.Contains("already enabled"))
        {
```

```
            Console.WriteLine("Control is already enabled for this target");
            return null;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
 (ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
 enabling controls.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable control. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte EnableControla Referência AWS SDK para .NET da API.

CLI

AWS CLI

Ativar um controle do Control Tower

O `enable-control` exemplo a seguir ativa um AWS controle Control Tower.

```
aws controltower enable-control \
    --control-
identifier arn:aws:controlcatalog:::control/497wrm2xnk1wxlf4obrxxxxxx \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryxxxxx/ou-
oqxx-i5wnxxxx
```

Saída:

```
{
    "arn": "arn:aws:controltower:us-
east-1:123456789012:enabledcontrol/18J5KBJ3W3VTIRLV",
```

```
    "operationIdentifier": "7691fc5a-de87-4540-8c95-b0aabd56382c"
}
```

Para obter mais informações, consulte Sobre os controles na AWS Control Tower no Guia do usuário da AWS Control Tower.

- Para obter detalhes da API, consulte EnableControlem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started.
Operation ID: " + operationId);
```

```
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ?
e.getMessage() : "";
```

```
                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for
this target");
                            return null;
                        }

                        if ("ValidationException".equals(errorCode)
                                && message.contains("enforced by default")) {
                            System.out.println("Control is enforced by default
and cannot be enabled manually.");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS
Control Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }
```

- Para obter detalhes da API, consulte EnableControla Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def enable_control(self, control_arn: str, target_identifier: str):
        """
        Enables a control for a specified target.
```

```
        :param control_arn: The ARN of the control to enable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If enabling the control fails.
        """
        try:
            print(control_arn)
            print(target_identifier)
            response = self.controltower_client.enable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id

        except ClientError as err:
            if (
                err.response["Error"]["Code"] == "ValidationException"
                and "already enabled" in err.response["Error"]["Message"]
            ):
                logger.info("Control is already enabled for this target")
                return None
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with
controls.")
                return None
            logger.error(
                "Couldn't enable control. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a EnableControlReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Enable the control
DATA(lo_output) = io_ctt->enablecontrol(
  iv_controlidentifier = iv_control_arn
  iv_targetidentifier  = iv_target_identifier
).

DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

" Wait for operation to complete
DATA lv_status TYPE /aws1/cttcontrolopstatus.
DO 100 TIMES.
  lv_status = get_control_operation(
    io_ctt = io_ctt
    iv_operation_id = lv_operation_id
  ).

  DATA(lv_msg) = |Control operation status: { lv_status }|.
  MESSAGE lv_msg TYPE 'I'.

  IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
    EXIT.
  ENDIF.

  " Wait 30 seconds
  WAIT UP TO 30 SECONDS.
```

```
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Control enabled successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a EnableControlreferência da API AWS SDK for SAP ABAP.

Use **`GetBaselineOperation`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBaselineOperation`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the status of a baseline operation.
    /// </summary>
    /// <param name="operationId">The ID of the baseline operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<BaselineOperationStatus> GetBaselineOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetBaselineOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetBaselineOperationAsync(request);
            return response.BaselineOperation.Status;
```

```
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get baseline operation status. Here's
 why: {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte GetBaselineOperationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Obter uma operação de linha de base do Control Tower

O `get-baseline-operation` exemplo a seguir obtém detalhes de uma operação de linha de base da AWS Control Tower.

```
aws controltower get-baseline-operation \
    --operation-identifier "51e190ac-8a37-4f6d-b63c-fb5104b5db38"
```

Saída:

```
{
    "baselineOperation": {
        "endTime": "2025-04-17T23:48:46+00:00",
        "operationIdentifier": "51e190ac-8a37-4f6d-b63c-fb5104b5db38",
        "operationType": "ENABLE_BASELINE",
        "startTime": "2025-04-17T23:46:37+00:00",
        "status": "SUCCEEDED",
        "statusMessage": "AWS Control Tower completed the baseline operation
 successfully."
```

```
    }
}
```

Para obter mais informações, consulte Tipos de linhas de base no Guia do usuário do AWS Control Tower.

- Para obter detalhes da API, consulte GetBaselineOperationem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Gets the status of a baseline operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
            String operationIdentifier) {

        GetBaselineOperationRequest request =
GetBaselineOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getBaselineOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null
                                ? exception.getCause()
                                : exception;
```

```
                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Baseline operation not found: %s"
                                                .formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }
```

- Para obter detalhes da API, consulte GetBaselineOperationa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def get_baseline_operation(self, operation_id: str):
        """
        Gets the status of a baseline operation.

        :param operation_id: The ID of the baseline operation.
```

```
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_baseline_operation(
                operationIdentifier=operation_id
            )
            return response["baselineOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get baseline operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a GetBaselineOperationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA(lo_output) = io_ctt->getbaselineoperation(
  iv_operationidentifier = iv_operation_id
).

ov_status = lo_output->get_baselineoperation( )->get_status( ).
```

- Para obter detalhes da API, consulte a GetBaselineOperationreferência da API AWS SDK for SAP ABAP.

Use **`GetControlOperation`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetControlOperation`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the status of a control operation.
    /// </summary>
    /// <param name="operationId">The ID of the control operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<ControlOperationStatus> GetControlOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetControlOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetControlOperationAsync(request);
            return response.ControlOperation.Status;
```

```
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get control operation status. Here's
 why: {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte GetControlOperationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Obter operações de controle do Control Tower

O `get-control-operation` exemplo a seguir obtém detalhes de uma operação de AWS controle da Control Tower.

```
aws controltower get-control-operation \
    --operation-identifier "7691fc5a-de87-4540-8c95-b0aabd56382c"
```

Saída:

```
{
    "controlOperation": {
        "controlIdentifier":
 "arn:aws:controlcatalog:::control/497wrm2xnk1wxlf4obrdo7mej",
        "enabledControlIdentifier": "arn:aws:controltower:us-
east-1:123456789012:enabledcontrol/18J5KBJ3W3VTIRLV",
        "endTime": "2025-04-17T03:08:55+00:00",
        "operationIdentifier": "7691fc5a-de87-4540-8c95-b0aabd56382c",
```

```
        "operationType": "ENABLE_CONTROL",
        "startTime": "2025-04-17T03:07:52+00:00",
        "status": "SUCCEEDED",
        "statusMessage": "Operation was successful.",
        "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-s64ryixxxx/
ou-oqxx-i5wnxxxx"
    }
}
```

Para obter mais informações, consulte Sobre os controles na AWS Control Tower no Guia do usuário da AWS Control Tower.

- Para obter detalhes da API, consulte GetControlOperationem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();
```

```
        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }
```

- Para obter detalhes da API, consulte GetControlOperationa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def get_control_operation(self, operation_id: str):
        """
        Gets the status of a control operation.

        :param operation_id: The ID of the control operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
```

```
            response = self.controltower_client.get_control_operation(
                operationIdentifier=operation_id
            )
            return response["controlOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get control operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a GetControlOperationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA(lo_output) = io_ctt->getcontroloperation(
  iv_operationidentifier = iv_operation_id
).

ov_status = lo_output->get_controloperation( )->get_status( ).
```

- Para obter detalhes da API, consulte a GetControlOperationreferência da API AWS SDK for SAP ABAP.

Use **`ListBaselines`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListBaselines`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List all baselines.
    /// </summary>
    /// <returns>A list of baseline summaries.</returns>
    public async Task<List<BaselineSummary>> ListBaselinesAsync()
    {
        try
        {
            var baselines = new List<BaselineSummary>();

            var baselinesPaginator =
_controlTowerService.Paginators.ListBaselines(new ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            return baselines;
        }
        catch (AmazonControlTowerException ex)
        {
```

```
            Console.WriteLine($"Couldn't list baselines. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListBaselinesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Listar as linhas de base do Control Tower

O `list-baselines` exemplo a seguir lista todas as linhas de base disponíveis da AWS Control Tower.

```
aws controltower list-baselines
```

Saída:

```
{
    "baselines": [
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/4T4HA1KMO10S6311",
            "description": "Sets up resources to monitor security and compliance
 of accounts in your organization.",
            "name": "AuditBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/J8HX46AHS5MIKQPD",
            "description": "Sets up a central repository for logs of API
 activities and resource configurations from accounts in your organization.",
            "name": "LogArchiveBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/LN25R72TTG6IGPTQ",
```

```
            "description": "Sets up shared resources for AWS Identity Center,
 which prepares the AWSControlTowerBaseline to set up Identity Center access for
 accounts.",
            "name": "IdentityCenterBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/17BSJV3IGJ2QSGA2",
            "description": "Sets up resources and mandatory controls for member
 accounts within the target OU, required for AWS Control Tower governance.",
            "name": "AWSControlTowerBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/3WPD0NA6TJ9AOMU2",
            "description": "Sets up a central AWS Backup vault in your
 organization.",
            "name": "BackupCentralVaultBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/H6C5JFCJJ3CPU3J5",
            "description": "Sets up AWS Backup Audit Manager.",
            "name": "BackupAdminBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/APO9ATVPBKFRRGLK",
            "description": "Sets up a local AWS Backup vault and attaches
 multiple AWS Backup plans.",
            "name": "BackupBaseline"
        }
    ]
}
```

Para obter mais informações, consulte Tipos de linhas de base no Guia do usuário do AWS Control Tower.

- Para obter detalhes da API, consulte ListBaselinesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all available baselines using pagination to retrieve complete
 results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total:
 " + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
```

```
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
%s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }
```

- Para obter detalhes da API, consulte ListBaselinesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_baselines(self):
        """
        Lists all baselines.

        :return: List of baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_baselines")
```

```
            baselines = []
            for page in paginator.paginate():
                baselines.extend(page["baselines"])
            return baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListBaselinesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_baselines TYPE /aws1/cl_cttbaselinesummary=>tt_baselines.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all baselines using pagination
    DO.
      DATA(lo_output) = io_ctt->listbaselines(
        iv_nexttoken = lv_nexttoken
```

```
    ).

    APPEND LINES OF lo_output->get_baselines( ) TO lt_baselines.

    lv_nexttoken = lo_output->get_nexttoken( ).
    IF lv_nexttoken IS INITIAL.
      EXIT.
    ENDIF.
  ENDDO.

  ot_baselines = lt_baselines.
  MESSAGE 'Listed baselines successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ListBaselinesreferência da API AWS SDK for SAP ABAP.

Use **`ListEnabledBaselines`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListEnabledBaselines`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// List all enabled baselines.
/// </summary>
/// <returns>A list of enabled baseline summaries.</returns>
public async Task<List<EnabledBaselineSummary>> ListEnabledBaselinesAsync()
```

```
    {
        try
        {
            var enabledBaselines = new List<EnabledBaselineSummary>();

            var enabledBaselinesPaginator =
_controlTowerService.Paginators.ListEnabledBaselines(new
ListEnabledBaselinesRequest());

            await foreach (var response in enabledBaselinesPaginator.Responses)
            {
                enabledBaselines.AddRange(response.EnabledBaselines);
            }

            return enabledBaselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled baselines. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListEnabledBaselinesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Listar as linhas de base habilitadas no Control Tower

O `list-enabled-baselines` exemplo a seguir lista todas as linhas de base ativadas AWS do Control Tower.

```
aws controltower list-enabled-baselines
```

Saída:

```
{
    "enabledBaselines": [
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XAJNZNCBC1I386C7B",
            "baselineIdentifier": "arn:aws:controltower:us-east-1::baseline/
LN25R72TTG6IGPTQ",
            "statusSummary": {
                "status": "SUCCEEDED"
            },
            "targetIdentifier": "arn:aws:organizations::123456789012:account/
o-3onqfuxxxx/123456789012"
        },
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XAH3ZJL9DWA386CA5",
            "baselineIdentifier": "arn:aws:controltower:us-
east-1::baseline/4T4HA1KMO10S6311",
            "statusSummary": {
                "status": "SUCCEEDED"
            },
            "targetIdentifier": "arn:aws:organizations::123456789012:account/
o-3onqfuxxxx/012345098765"
        },
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XALFJ9548TL386CBT",
            "baselineIdentifier": "arn:aws:controltower:us-east-1::baseline/
J8HX46AHS5MIKQPD",
            "statusSummary": {
                "status": "SUCCEEDED"
            },
            "targetIdentifier": "arn:aws:organizations::123456789012:account/
o-3onqfuxxxx/098765432109"
        }
    ]
}
```

Para obter mais informações, consulte Tipos de linhas de base no Guia do usuário do AWS Control Tower.

- Para obter detalhes da API, consulte ListEnabledBaselinesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Lists all enabled baselines using pagination to retrieve complete results.
    *
    * @return a list of all enabled baselines
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
       System.out.println("Starting list enabled baselines paginator…");

       ListEnabledBaselinesRequest request =
               ListEnabledBaselinesRequest.builder().build();

       ListEnabledBaselinesPublisher paginator =
               getAsyncClient().listEnabledBaselinesPaginator(request);

       List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
       return paginator.subscribe(response -> {
                   if (response.enabledBaselines() != null
                           && !response.enabledBaselines().isEmpty()) {

                       response.enabledBaselines().forEach(baseline -> {
                           enabledBaselines.add(baseline);
                       });
                   } else {
                       System.out.println("Page contained no enabled
baselines.");
                   }
               })
               .thenRun(() ->
                       System.out.println(
```

```
                                "Successfully listed enabled baselines. Total: "
+
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled
baselines: %s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }
```

- Para obter detalhes da API, consulte ListEnabledBaselinesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_enabled_baselines(self):
        """
        Lists all enabled baselines.

        :return: List of enabled baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
```

```
            paginator =
 self.controltower_client.get_paginator("list_enabled_baselines")
            enabled_baselines = []
            for page in paginator.paginate():
                enabled_baselines.extend(page["enabledBaselines"])
            return enabled_baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't list enabled baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListEnabledBaselinesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_enabled_baselines TYPE /aws1/
cl_cttenbdbaselinesumm=>tt_enabledbaselines.
    DATA lv_nexttoken TYPE /aws1/cttlstenbdbaselinesnex00.

    " List all enabled baselines using pagination
    DO.
      DATA(lo_output) = io_ctt->listenabledbaselines(
        iv_nexttoken = lv_nexttoken
```

```
      ).

      APPEND LINES OF lo_output->get_enabledbaselines( ) TO lt_enabled_baselines.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_enabled_baselines = lt_enabled_baselines.
    MESSAGE 'Listed enabled baselines successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ListEnabledBaselinesreferência da API AWS SDK for SAP ABAP.

Use **`ListEnabledControls`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListEnabledControls`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List enabled controls for a target organizational unit.
    /// </summary>
    /// <param name="targetIdentifier">The target organizational unit
identifier.</param>
```

```
    /// <returns>A list of enabled control summaries.</returns>
    public async Task<List<EnabledControlSummary>>
ListEnabledControlsAsync(string targetIdentifier)
    {
        try
        {
            var request = new ListEnabledControlsRequest
            {
                TargetIdentifier = targetIdentifier
            };

            var enabledControls = new List<EnabledControlSummary>();

            var enabledControlsPaginator =
_controlTowerService.Paginators.ListEnabledControls(request);

            await foreach (var response in enabledControlsPaginator.Responses)
            {
                enabledControls.AddRange(response.EnabledControls);
            }

            return enabledControls;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return new List<EnabledControlSummary>();
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled controls. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListEnabledControlsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Listar os controles habilitados no Control Tower

O `list-enabled-controls` exemplo a seguir obtém detalhes dos controles ativados pela AWS Control Tower.

```
aws controltower list-enabled-controls \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryxxxxx/ou-oqxx-i5wnxxxx
```

Saída:

```
{
    "enabledControls": [
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledcontrol/26RGJRSLXCP1KW8D",
            "controlIdentifier": "arn:aws:controltower:us-east-1::control/AWS-GR_CLOUDTRAIL_CHANGE_PROHIBITED",
            "driftStatusSummary": {
                "driftStatus": "NOT_CHECKING"
            },
            "statusSummary": {
                "status": "SUCCEEDED"
            },
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-s64ryxxxxx/ou-oqxx-i5wnxxxx"
        },
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledcontrol/18AY24CWKM6IVSLU",
            "controlIdentifier": "arn:aws:controltower:us-east-1::control/AWS-GR_CLOUDTRAIL_CLOUDWATCH_LOGS_ENABLED",
            "driftStatusSummary": {
                "driftStatus": "NOT_CHECKING"
            },
            "statusSummary": {
                "status": "SUCCEEDED"
            },
```

```
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-
s64ryxxxxx/ou-oqxx-i5wnxxxx"
        }
    ]
}
```

Para obter mais informações, consulte Sobre os controles na AWS Control Tower no Guia do usuário da AWS Control Tower.

- Para obter detalhes da API, consulte ListEnabledControlsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Lists all enabled controls for a specific target using pagination.
    *
    * @param targetIdentifier the identifier of the target (e.g., OU ARN)
    * @return a list of enabled controls
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<List<EnabledControlSummary>>
listEnabledControlsAsync(String targetIdentifier) {
       System.out.println("Starting list enabled controls paginator for target "
+ targetIdentifier);
       ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
               .targetIdentifier(targetIdentifier)
               .build();

       ListEnabledControlsPublisher paginator =
getAsyncClient().listEnabledControlsPaginator(request);
       List<EnabledControlSummary> enabledControls = new ArrayList<>();
```

```
        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled
 controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +"
 enabled controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
 e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to
 work with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
```

```
                                        "Error listing enabled controls:
 %s".formatted(e.getMessage()), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
 %s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled
 controls", cause);
                });
    }
```

- Para obter detalhes da API, consulte ListEnabledControlsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
```

```
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_enabled_controls(self, target_identifier: str):
        """
        Lists all enabled controls for a specific target.

        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: List of enabled controls.
        :raises ClientError: If the listing operation fails.
        """
        enabled_controls = []
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_controls")

            for page in paginator.paginate(targetIdentifier=target_identifier):
                enabled_controls.extend(page["enabledControls"])
            return enabled_controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
                return enabled_controls
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with
controls.")
                return enabled_controls
```

```
            else:
                logger.error(
                    "Couldn't list enabled controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListEnabledControlsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_enabled_controls TYPE /aws1/
cl_cttenabledcontrolsumm=>tt_enabledcontrols.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all enabled controls using pagination
    DO.
      DATA(lo_output) = io_ctt->listenabledcontrols(
        iv_targetidentifier = iv_target_identifier
        iv_nexttoken        = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_enabledcontrols( ) TO lt_enabled_controls.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.
```

```
    ot_enabled_controls = lt_enabled_controls.
    MESSAGE 'Listed enabled controls successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ListEnabledControlsreferência da API AWS SDK for SAP ABAP.

Use **`ListLandingZones`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListLandingZones`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the AWS Control Tower landing zones for an account.
    /// </summary>
    /// <returns>A list of LandingZoneSummary objects.</returns>
    public async Task<List<LandingZoneSummary>> ListLandingZonesAsync()
    {
        try
        {
            var landingZones = new List<LandingZoneSummary>();

            var landingZonesPaginator =
_controlTowerService.Paginators.ListLandingZones(new ListLandingZonesRequest());
```

```
            await foreach (var response in landingZonesPaginator.Responses)
            {
                landingZones.AddRange(response.LandingZones);
            }

            return landingZones;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list landing zones. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListLandingZonesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Listar um identificador da zona de pouso do Control Tower

O exemplo de `list-landing-zones` a seguir lista uma zona de pouso do `identifier` Control Tower.

```
aws controltower list-landing-zones
```

Saída:

```
{
    "landingZones": [
        {
            "arn": "arn:aws:controltower:us-
east-1:123456789012:landingzone/13CJG46WZKXXX4X5"
        }
    ]
}
```

Para obter mais informações, consulte Introdução à AWS Control Tower no Guia do usuário da AWS Control Tower.

- Para obter detalhes da API, consulte ListLandingZonesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request =
 ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
```

```
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing
zones: " + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
cause);
                });
    }
```

- Para obter detalhes da API, consulte ListLandingZonesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_landing_zones(self):
        """
        Lists all landing zones.

        :return: List of landing zones.
        :raises ClientError: If the listing operation fails.
        """
        try:
```

```
            paginator =
 self.controltower_client.get_paginator("list_landing_zones")
            landing_zones = []
            for page in paginator.paginate():
                landing_zones.extend(page["landingZones"])
            return landing_zones

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
 permissions."
                )
            else:
                logger.error(
                    "Couldn't list landing zones. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListLandingZonesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_landing_zones TYPE /aws1/
cl_cttlandingzonesummary=>tt_landingzonesummaries.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all landing zones using pagination
```

```
    DO.
      DATA(lo_output) = io_ctt->listlandingzones(
        iv_nexttoken = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_landingzones( ) TO lt_landing_zones.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_landing_zones = lt_landing_zones.
    MESSAGE 'Listed landing zones successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ListLandingZonesreferência da API AWS SDK for SAP ABAP.

Use **`ResetEnabledBaseline`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ResetEnabledBaseline`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Reset an enabled baseline for a specific target.
```

```
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the enabled
baseline to reset.</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> ResetEnabledBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new ResetEnabledBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.ResetEnabledBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Target not found, unable to reset enabled
baseline.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't reset enabled baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
```

```
        }
    }
```

- Para obter detalhes da API, consulte ResetEnabledBaselinea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Redefinir uma linha de base ativada no Control Tower

O `reset-enabled-baseline` exemplo a seguir redefine uma linha de base ativada pelo AWS Control Tower.

```
aws controltower reset-enabled-baseline \
    --enabled-baseline-identifier arn:aws:controltower:us-
east-1:123456789012:enabledbaseline/XOM12BEL4YD578CQ2
```

Saída:

```
{
    "operationIdentifier": "214cde95-5c39-46b9-b429-4fad550a7096"
}
```

Para obter mais informações, consulte Tipos de linhas de base no Guia do usuário do AWS Control Tower.

- Para obter detalhes da API, consulte ResetEnabledBaselineem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Resets an enabled baseline for a specific target.
    *
    * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
    * @return the operation identifier
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting reset of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       ResetEnabledBaselineRequest request =
ResetEnabledBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().resetEnabledBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Reset enabled baseline operation ID: " +
operationId);

                   // Polling loop
                   CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                   Runnable poller = new Runnable() {
```

```
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status)
|| "FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Baseline
operation finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking
baseline operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();
```

```
                        System.out.println("ControlTowerException caught: Code="
 + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
 sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline:
 " + cause.getMessage());
                    return null;
                });
    }
```

- Para obter detalhes da API, consulte ResetEnabledBaselinea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
```

```
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def reset_enabled_baseline(self, enabled_baseline_identifier: str):
        """
        Resets an enabled baseline for a specific target.

        :param enabled_baseline_identifier: The identifier of the enabled
baseline to reset.
        :return: The operation ID.
        :raises ClientError: If resetting the baseline fails.
        """
        try:
            response = self.controltower_client.reset_enabled_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )
            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't reset enabled baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ResetEnabledBaselineReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Reset the enabled baseline
DATA(lo_output) = io_ctt->resetenabledbaseline(
  iv_enabledbaselineidentifier = iv_enabled_baseline_identifier
).

DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

" Wait for operation to complete
DATA lv_status TYPE /aws1/cttbaselineopstatus.
DO 100 TIMES.
  lv_status = get_baseline_operation(
    io_ctt = io_ctt
    iv_operation_id = lv_operation_id
  ).

  DATA(lv_msg) = |Baseline operation status: { lv_status }|.
  MESSAGE lv_msg TYPE 'I'.

  IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
    EXIT.
  ENDIF.

  " Wait 30 seconds
  WAIT UP TO 30 SECONDS.
ENDDO.
```

```
    ov_operation_id = lv_operation_id.
    MESSAGE 'Baseline reset successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ResetEnabledBaselinereferência da API AWS SDK for SAP ABAP.

Exemplos de código para Firehose usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Data Firehose com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Firehose: mais informações sobre o Firehose.
- Referência de API do Firehose: detalhes sobre todas as ações disponíveis do Firehose.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de uso do Firehose AWS SDKs
  - Ações para Firehose usando AWS SDKs
    - Use PutRecordcom um AWS SDK ou CLI
    - Use PutRecordBatchcom um AWS SDK ou CLI
- Cenários para o uso do Firehose AWS SDKs
  - Uso do Amazon Data Firehose para processar registros individuais e em lote

Exemplos básicos de uso do Firehose AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Data Firehose os AWS SDKs.

Exemplos

- Ações para Firehose usando AWS SDKs
  - Use PutRecordcom um AWS SDK ou CLI
  - Use PutRecordBatchcom um AWS SDK ou CLI

Ações para Firehose usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Firehose com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Firehose e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Firehose AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de API do Amazon Data Firehose.

Exemplos

- Use PutRecordcom um AWS SDK ou CLI
- Use PutRecordBatchcom um AWS SDK ou CLI

Use **`PutRecord`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutRecord`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inserção de registros no Firehose

CLI

AWS CLI

Para gravar um registro em um fluxo

O exemplo `put-record` a seguir grava dados em um fluxo. Os dados são codificados no formato Base64.

```
aws firehose put-record \
    --delivery-stream-name my-stream \
    --record '{"Data":"SGVsbG8gd29ybGQ="}'
```

Saída:

```
{
    "RecordId": "RjB5K/nnoGFHqwTsZlNd/
TTqvjE8V5dsyXZTQn2JXrdpMTOwssyEb6nfC8fwf1whhwnItt4mvrn+gsqeK5jB7QjuLg283+Ps4Sz/
j1Xujv31iDhnPdaLw4BOyM9Amv7PcCuB2079RuM0NhoakbyUymlwY8yt20G8X2420wu1jlFafhci4erAt7QhDEvpw
    "Encrypted": false
}
```

Para obter mais informações, consulte Sending Data to an Amazon Kinesis Data Firehose Delivery Stream no Guia do desenvolvedor do Amazon Kinesis Data Firehose.

- Para obter detalhes da API, consulte PutRecordem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery
 stream.
```

```
    *
    * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
               .build();

           PutRecordRequest putRecordRequest = PutRecordRequest.builder()
               .deliveryStreamName(deliveryStreamName)
               .record(firehoseRecord)
               .build();

           getFirehoseClient().putRecord(putRecordRequest);
           System.out.println("Record sent: " + jsonRecord);
       } catch (Exception e) {
           throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
       }
   }
```

- Para obter detalhes da API, consulte PutRecorda Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
```

```
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise
```

- Para obter detalhes da API, consulte a PutRecordReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_record) = NEW /aws1/cl_frhrecord( iv_data = iv_data ).

        DATA(lo_result) = lo_frh->putrecord(
          iv_deliverystreamname = iv_deliv_stream_name
          io_record             = lo_record ).
```

```
        MESSAGE 'Record sent to Firehose delivery stream.' TYPE 'I'.
      CATCH /aws1/cx_frhresourcenotfoundex.
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
      CATCH /aws1/cx_frhinvalidargumentex.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
      CATCH /aws1/cx_frhserviceunavailex.
        MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutRecordreferência da API AWS SDK for SAP ABAP.

Use **`PutRecordBatch`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutRecordBatch`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Inserção de registros no Firehose

CLI

AWS CLI

Como gravar vários registros em um fluxo

O exemplo de `put-record-batch` a seguir grava três registro em um fluxo. Os dados são codificados no formato Base64.

```
aws firehose put-record-batch \
    --delivery-stream-name my-stream \
    --records file://records.json
```

Conteúdo de `myfile.json`:

```
[
    {"Data": "Rmlyc3QgdGhpbmc="},
```

```
    {"Data": "U2Vjb25kIHRoaW5n"},
    {"Data": "VGhpcmQgdGhpbmc="}
]
```

Saída:

```
{
    "FailedPutCount": 0,
    "Encrypted": false,
    "RequestResponses": [
        {
            "RecordId": "9D2OJ6t2EqCTZTXwGzeSv/EVHxRoRCw89xd+o3+sXg8DhYOaWKPSmZy/
CGlRVEys1u1xbeKh6VofEYKkoeiDrcjrxhQp9iF7sUW7pujiMEQ5LzlrzCkGosxQn
+3boDnURDEaD42V7GiixpOyLJkYZcae1i7HzlCEoy9LJhMr8EjDSi4Om/9Vc2uhwwuAtGE0XKpxJ2WD7ZRWtAnYlK
        },
        {
            "RecordId": "jFirejqxCLlK5xjH/UNmlMVcjktEN76I7916X9PaZ
+PVaOSXDfU1WGOqEZhxq2js7xcZ552eoeDxsuTU1MSq9nZTbVfb6cQTIXnm/
GsuF37Uhg67GKmR5z90l6XKJ+/
+pDloFv7Hh9a3oUS6wYm3DcNRLTHHAimANp1PhkQvWpvLRfzbuCUkBphR2QVzhP9OiHLbzGwy8/
DfH8sqWEUYASNJKS8GXP5s"
        },
        {
            "RecordId":
 "oy0amQ40o5Y2YV4vxzufdcMOOw6n3EPr3tpPJGoYVNKH4APPVqNcbUgefo1stEFRg4hTLrf2k6eliHu/9+YJ5R3
DTBt3qBlmTj7Xq8SKVbO1S7YvMTpWkMKA86f8JfmT8BMKoMb4XZS/sOkQLe+qh0sYKXWl"
        }
    ]
}
```

Para obter mais informações, consulte Sending Data to an Amazon Kinesis Data Firehose Delivery Stream no Guia do desenvolvedor do Amazon Kinesis Data Firehose.

- Para obter detalhes da API, consulte PutRecordBatchem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param records           a list of maps representing the records to be
sent
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
```

```
                        String jsonRecord =
objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                            .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose
record", e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }
```

- Para obter detalhes da API, consulte PutRecordBatcha Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
```

```
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")
```

- Para obter detalhes da API, consulte a PutRecordBatchReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn put_record_batch(
```

```
    client: &Client,
    stream: &str,
    data: Vec<Record>,
) -> Result<PutRecordBatchOutput, SdkError<PutRecordBatchError>> {
    client
        .put_record_batch()
        .delivery_stream_name(stream)
        .set_records(Some(data))
        .send()
        .await
}
```

- Para obter detalhes da API, consulte a PutRecordBatchreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_frh->putrecordbatch(
          iv_deliverystreamname = iv_deliv_stream_name
          it_records            = it_records ).

        DATA(lv_failed_count) = lo_result->get_failedputcount( ).

        IF lv_failed_count > 0.
          MESSAGE |{ lv_failed_count } records failed to send.| TYPE 'I'.
        ELSE.
          MESSAGE 'All records sent successfully to Firehose delivery stream.'
 TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_frhresourcenotfoundex.
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
```

```
    CATCH /aws1/cx_frhinvalidargumentex.
      MESSAGE 'Invalid argument provided.' TYPE 'E'.
    CATCH /aws1/cx_frhserviceunavailex.
      MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a PutRecordBatchreferência da API AWS SDK for SAP ABAP.

Cenários para o uso do Firehose AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Firehose com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções dentro do Firehose ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Uso do Amazon Data Firehose para processar registros individuais e em lote

Uso do Amazon Data Firehose para processar registros individuais e em lote

Os exemplos de código a seguir mostram como usar o Firehose para processar registros individuais e em lote.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo insere registros individuais e em lote no Firehose.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
 objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
 e.getMessage());
                }
```

```
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
```

```
    * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
               .build();

           PutRecordRequest putRecordRequest = PutRecordRequest.builder()
               .deliveryStreamName(deliveryStreamName)
               .record(firehoseRecord)
               .build();

           getFirehoseClient().putRecord(putRecordRequest);
           System.out.println("Record sent: " + jsonRecord);
       } catch (Exception e) {
           throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
       }
   }


   /**
    * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param records           a list of maps representing the records to be
sent
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
```

```
    * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord =
objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose
record", e);
                   }
               }).collect(Collectors.toList());

               PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                   .deliveryStreamName(deliveryStreamName)
                   .records(batchRecords)
                   .build();

               PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

               if (response.failedPutCount() > 0) {
                   response.requestResponses().stream()
                       .filter(r -> r.errorCode() != null)
```

```
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime,
Instant endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)

.dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).bui
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
```

```
            System.err.println("Failed to monitor metric " + metricName + ": " +
 e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
 FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8))
 {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - PutRecord
  - PutRecordBatch

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este script insere registros individuais e em lote no Firehose.

```
import json
import logging
import random
from datetime import datetime, timedelta

import backoff
import boto3

from config import get_config


def load_sample_data(path: str) -> dict:
    """
    Load sample data from a JSON file.

    Args:
        path (str): The file path to the JSON file containing sample data.

    Returns:
        dict: The loaded sample data as a dictionary.
    """
    with open(path, "r") as f:
        return json.load(f)


# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """
```

```
    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
```

```
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")


    def get_metric_statistics(
        self,
        metric_name: str,
        start_time: datetime,
        end_time: datetime,
        period: int,
        statistics: list = ["Sum"],
    ) -> list:
        """
        Retrieve metric statistics from CloudWatch.

        Args:
            metric_name (str): The name of the metric.
            start_time (datetime): The start time for the metric statistics.
            end_time (datetime): The end time for the metric statistics.
            period (int): The granularity, in seconds, of the returned data
points.
```

```
            statistics (list): A list of statistics to retrieve. Default is
['Sum'].

        Returns:
            list: List of datapoints containing the metric statistics.
        """
        response = self.cloudwatch.get_metric_statistics(
            Namespace="AWS/Firehose",
            MetricName=metric_name,
            Dimensions=[
                {"Name": "DeliveryStreamName", "Value":
self.delivery_stream_name},
            ],
            StartTime=start_time,
            EndTime=end_time,
            Period=period,
            Statistics=statistics,
        )
        return response["Datapoints"]

    def monitor_metrics(self):
        """
        Monitor Firehose metrics for the last 5 minutes.

        This method retrieves and logs the 'IncomingBytes', 'IncomingRecords',
and 'FailedPutCount' metrics
        from CloudWatch for the last 5 minutes.
        """
        end_time = datetime.utcnow()
        start_time = end_time - timedelta(minutes=10)
        period = int((end_time - start_time).total_seconds())

        metrics = {
            "IncomingBytes": self.get_metric_statistics(
                "IncomingBytes", start_time, end_time, period
            ),
            "IncomingRecords": self.get_metric_statistics(
                "IncomingRecords", start_time, end_time, period
            ),
            "FailedPutCount": self.get_metric_statistics(
                "FailedPutCount", start_time, end_time, period
            ),
        }
```

```
        for metric, datapoints in metrics.items():
            if datapoints:
                total_sum = sum(datapoint["Sum"] for datapoint in datapoints)
                if metric == "IncomingBytes":
                    logger.info(
                        f"{metric}: {round(total_sum)} ({total_sum / (1024 *
1024):.2f} MB)"
                    )
                else:
                    logger.info(f"{metric}: {round(total_sum)}")
            else:
                logger.info(f"No data found for {metric} over the last 5
minutes")


    def _create_record_entry(self, record: dict) -> dict:
        """
        Create a record entry for Firehose.

        Args:
            record (dict): The data record to be sent.

        Returns:
            dict: The record entry formatted for Firehose.

        Raises:
            Exception: If a simulated network error occurs.
        """
        if random.random() < 0.2:
            raise Exception("Simulated network error")
        elif random.random() < 0.1:
            return {"Data": '{"malformed": "data"'}
        else:
            return {"Data": json.dumps(record)}

    def _log_response(self, response: dict, entry: dict):
        """
        Log the response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record API call.
            entry (dict): The record entry that was sent.
        """
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
```

```
            logger.info(f"Sent record: {entry}")
        else:
            logger.info(f"Fail record: {entry}")

    def _log_batch_response(self, response: dict, batch_size: int):
        """
        Log the batch response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record_batch API
 call.
            batch_size (int): The number of records in the batch.
        """
        if response.get("FailedPutCount", 0) > 0:
            logger.info(
                f'Failed to send {response["FailedPutCount"]} records in batch of
 {batch_size}'
            )
        else:
            logger.info(f"Successfully sent batch of {batch_size} records")


if __name__ == "__main__":
    config = get_config()
    data = load_sample_data(config.sample_data_file)
    client = FirehoseClient(config)

    # Process the first 100 sample network records
    for record in data[:100]:
        try:
            client.put_record(record)
        except Exception as e:
            logger.info(f"Put record failed after retries and backoff: {e}")
    client.monitor_metrics()

    # Process remaining records using the batch method
    try:
        client.put_record_batch(data[100:])
    except Exception as e:
        logger.info(f"Put record batch failed after retries and backoff: {e}")
    client.monitor_metrics()
```

Este arquivo contém a configuração do script acima.

```
class Config:
    def __init__(self):
        self.delivery_stream_name = "ENTER YOUR DELIVERY STREAM NAME HERE"
        self.region = "us-east-1"
        self.sample_data_file = (
            "../../../../../scenarios/features/firehose/resources/
sample_records.json"
        )


def get_config():
    return Config()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - PutRecord
  - PutRecordBatch

Exemplos de código para Device Farm usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Device Farm com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Device Farm: mais informações sobre o Device Farm.
- Referência da API Device Farm: detalhes sobre todas as ações disponíveis do Device Farm.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.

- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Device Farm AWS SDKs
  - Ações para Device Farm usando AWS SDKs
    - Usar CreateUpload com uma CLI
- Cenários para o uso do Device Farm AWS SDKs
  - Execute testes de navegador com o Device Farm e faça capturas de tela usando um AWS SDK
  - Faça upload e teste pacotes de dispositivos móveis com o Device Farm usando um AWS SDK

Exemplos básicos para o uso do Device Farm AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Device Farm com AWS SDKs.

Exemplos

- Ações para Device Farm usando AWS SDKs
  - Usar CreateUpload com uma CLI

Ações para Device Farm usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Device Farm com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Device Farm e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Device Farm AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Device Farm.

Exemplos

- Usar CreateUpload com uma CLI

Usar **`CreateUpload`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateUpload`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Fazer upload e testar pacotes de dispositivos

CLI

AWS CLI

Para criar um upload

O comando a seguir cria um upload para uma aplicação do Android:

```
aws devicefarm create-upload --project-arn "arn:aws:devicefarm:us-
west-2:123456789012:project:070fc3ca-7ec1-4741-9c1f-d3e044efc506" --name app.apk
 --type ANDROID_APP
```

É possível obter o ARN do projeto a partir da saída de create-project ou list-projects.

Saída:

```
{
    "upload": {
        "status": "INITIALIZED",
        "name": "app.apk",
        "created": 1503614408.769,
        "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
arn%3Aaws%3Adevicefarm%3Aus-west-2%3A123456789012%3Aproject%3A070fc3ca-
c7e1-4471-91cf-d3e4efc50604/uploads/arn%3Aaws%3Adevicefarm%3Aus-
west-2%3A123456789012%3Aupload%3A070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514/app.apk?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20170824T224008Z&X-Amz-SignedHeaders=host&X-Amz-Expires=86400&X-Amz-
Credential=AKIAEXAMPLEPBUMBC3GA%2F20170824%2Fus-west-2%2Fs3%2Faws4_request&X-Amz-
Signature=05050370c38894ef5bd09f5d009f36fc8f96fa4bb04e1bba9aca71b8dbe49a0f",
        "type": "ANDROID_APP",
        "arn": "arn:aws:devicefarm:us-
west-2:123456789012:upload:070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514"
```

```
    }
}
```

Use o URL assinado na saída para carregar um arquivo no Device Farm:

```
curl -T app.apk "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
arn%3Aaws%3Adevicefarm%3Aus-west-2%3A123456789012%3Aproject%3A070fc3ca-
c7e1-4471-91cf-d3e4efc50604/uploads/arn%3Aaws%3Adevicefarm%3Aus-
west-2%3A123456789012%3Aupload%3A070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514/app.apk?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20170824T224008Z&X-Amz-SignedHeaders=host&X-Amz-Expires=86400&X-Amz-
Credential=AKIAEXAMPLEPBUMBC3GA%2F20170824%2Fus-west-2%2Fs3%2Faws4_request&X-Amz-
Signature=05050370c38894ef5bd09f5d009f36fc8f96fa4bb04e1bba9aca71b8dbe49a0f"
```

- Para obter detalhes da API, consulte CreateUploadem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria um upload AWS do Device Farm para um aplicativo Android. Você pode obter o ARN do projeto a partir da saída de New-DFProject ou. Get-DFProjectList Use o URL assinado na New-DFUpload saída para carregar um arquivo no Device Farm.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn
 "arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- Para obter detalhes da API, consulte CreateUploadem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um upload AWS do Device Farm para um aplicativo Android. Você pode obter o ARN do projeto a partir da saída de New-DFProject ou. Get-DFProjectList Use o URL assinado na New-DFUpload saída para carregar um arquivo no Device Farm.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn
 "arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- Para obter detalhes da API, consulte CreateUploadem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para o uso do Device Farm AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Device Farm com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Device Farm ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Execute testes de navegador com o Device Farm e faça capturas de tela usando um AWS SDK
- Faça upload e teste pacotes de dispositivos móveis com o Device Farm usando um AWS SDK

Execute testes de navegador com o Device Farm e faça capturas de tela usando um AWS SDK

O exemplo de código a seguir mostra como executar testes de navegador com o Device Farm e fazer capturas de tela.

Python

SDK para Python (Boto3)

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o PyTest Selenium para navegar até sites específicos, fazer capturas de tela e comparar o conteúdo real do site com o conteúdo esperado.

```
import datetime
```

```
import os
import subprocess
import boto3
import pytest
from selenium import webdriver
from selenium.webdriver import DesiredCapabilities
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait


def get_git_hash():
    """
    Get the short Git hash of the current commit of the repository
    """
    try:
        return (
            subprocess.check_output(["git", "rev-parse", "--short", "HEAD"])
            .decode("utf-8")
            .strip()
        )
    except:
        return "norepo"


class TestHelloSuite:
    """
    Our test suite.

    This style of test suite allows us to use setup_method and teardown_method.

    """

    def save_screenshot(self, name):
        self.driver.save_screenshot(os.path.join(self.screenshot_path, name))

    def setup_method(self, method):
        """
        Set up a test.

        This makes sure that the session for an individual test is ready.
```

```
        The AWS credentials are read from the default ~/.aws/credentials or from
the
        command line by setting the AWS_ACCESS_KEY_ID and AWS_SECRET_KEY
environment
        variables.

        The project Amazon Resource Name (ARN) is determined by the PROJECT_ARN
        environment variable.
        """
        devicefarm_client = boto3.client("devicefarm")
        project_arn = os.environ.get("PROJECT_ARN", None)
        if project_arn is None:
            raise ValueError("Must set PROJECT_ARN")
        # Request a driver hub URL for the Selenium client
        testgrid_url_response = devicefarm_client.create_test_grid_url(
            projectArn=project_arn, expiresInSeconds=300
        )

        # We want a directory to save our files into. We're going to make a
directory
        # in the current directory that holds our results.
        self.screenshot_path = os.path.join(
            ".", "results", get_git_hash() + "-" +
(datetime.date.today().isoformat())
        )
        if not os.path.exists(self.screenshot_path):
            os.makedirs(self.screenshot_path, exist_ok=True)

        # We want a Firefox instance on Windows
        desired_cap = DesiredCapabilities.FIREFOX
        desired_cap["platform"] = "windows"
        desired_cap["BrowserVersion"] = "latest"

        # Configure the webdriver with the appropriate remote endpoint.
        self.driver = webdriver.Remote(testgrid_url_response["url"], desired_cap)

        #
        # Auto-Tagging
        #

        # In order to get the Session ARN, we need to look up the session by the
        # Project ARN and session ID (from the driver).
        testgrid_session_arn_response = devicefarm_client.get_test_grid_session(
            projectArn=project_arn, sessionId=self.driver.session_id
```

```
        )

        # Save the session's ARN so we can tag the session.
        self.session_arn = testgrid_session_arn_response["testGridSession"]
["arn"]

        # In order to tag it, we're going to use the resourcegroupstaggingapi
 client to
        # add a tag to the session ARN that we just got.
        tag_client = boto3.client("resourcegroupstaggingapi")
        tag_client.tag_resources(
            ResourceARNList=[self.session_arn],
            Tags={"TestSuite": f"testsuite {method.__name__}", "GitId":
 get_git_hash()},
        )

    def teardown_method(self, method):
        """
        Clean up resources used by each method.
        """
        # End the Selenium session so we're off the clock.
        self.driver.quit()

    @pytest.mark.parametrize(
        "query,leading",
        [
            pytest.param(
                "Seattle",
                "Seattle (/si#ætəl/ (listen) see-AT-əl) is a seaport city on the
 West Coast of the United States.",
            ),
            pytest.param(
                "Selenium",
                "Selenium is a chemical element with the symbol Se and atomic
 number 34.",
            ),
            pytest.param(
                "Amazon Locker",
                "Amazon Locker is a self-service package delivery service offered
 by online retailer Amazon.",
            ),
            pytest.param(
                "Kootenai Falls",
```

```
                "Kootenai Falls is a waterfall on the Kootenay River located in
Lincoln County, Montana, just off U.S. Route 2.",
            ),
            pytest.param(
                "Dorayaki",
                "Dorayaki (####, ####, ####, ####) is a type of Japanese
confection.",
            ),
            pytest.param("Robot Face", "<|°_°|> (also known as Robot Face or
Robot)"),
        ],
    )
    def test_first_paragraph_text(self, query, leading):
        """
        This test looks at the first paragraph of a page on Wikipedia, comparing
it to
        a known leading sentence.

        If the leading sentence matches, the test passes. A screenshot is taken
before
        the final assertion is made, letting us debug if something isn't right.
        """
        # Open the main page of Wikipedia
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
        # Find the search box, enter a query, and press enter
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # Wait for the search box to go stale -- This means we've navigated
fully.
        WebDriverWait(self.driver, 5).until(
            expected_conditions.staleness_of(search_input)
        )
        # Get the leading paragraph of the article.
        lead = leading.lower()
        # Find the element...
        lead_para = self.driver.find_element(
            By.XPATH, "//div[@class='mw-parser-output']//p[not(@class)]"
        )
        # ... and copy out its text.
        our_text = lead_para.text.lower()
        our_text = our_text[: len(lead)]
        # Take a screenshot and compare the strings.
```

```
        self.save_screenshot(f"leadingpara_{query}.png")
        assert our_text.startswith(lead)

    @pytest.mark.parametrize(
        "query,expected",
        [
            pytest.param("Automation Testing", "Test Automation"),
            pytest.param("DevOps", "DevOps"),
            pytest.param("Jackdaws Love My Big Sphinx Of Quartz", "Pangram"),
            pytest.param("EarthBound", "EarthBound"),
            pytest.param("Covered Bridges Today", "Covered Bridges Today"),
            pytest.param("Kurt Godel", "Kurt Gödel"),
            pytest.param("N//ng language", "N#ng language"),
            pytest.param(
                "Who the Frick Is Jackson Pollock?", "Who the $&% Is Jackson
Pollock?"
            ),
        ],
    )
    def test_redirect_titles(self, query, expected):
        """
        A test comparing pages we expect to (or not to) redirect on Wikipedia.

        This test checks to see that the page ("query") redirects (or doesn't) to
the
        "expected" page title. Several of these are common synonyms
("Jackdaws...")
        while others are because of characters untypable by most keyboards ("N#ng
language")

        A screenshot is taken just before the final assertion is made to aid in
        debugging and verification.
        """
        # Open the main page of Wikipedia
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
        # Find the search box, enter some text into it, and send an enter key.
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # wait until the page has rolled over -- once the search input handle is
stale,
        # the browser has navigated.
        WebDriverWait(self.driver, 5).until(
```

```
            expected_conditions.staleness_of(search_input)
        )
        # Get the first heading & take a screenshot
        our_text = self.driver.find_element(By.ID, "firstHeading").text.lower()
        self.save_screenshot(f"redirect_{query}.png")
        # did it match?
        assert our_text == expected.lower()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateTestGridUrl
  - GetTestGridSession

Faça upload e teste pacotes de dispositivos móveis com o Device Farm usando um AWS SDK

O exemplo de código a seguir mostra como fazer upload e testar pacotes de dispositivos móveis com o Device Farm.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de aplicativos Android compilados e pacotes de teste para o Device Farm, inicie um teste, aguarde a conclusão do teste e relate os resultados.

```
import boto3
import os
import requests
import string
import random
```

```
import datetime
import time

# Update this dict with your own values before you run the example:
config = {
    # This is our app under test.
    "appFilePath": "app-debug.apk",
    "projectArn": "arn:aws:devicefarm:us-west-2:111222333444:project:581f5703-
e040-4ac9-b7ae-0ba007bfb8e6",
    # Since we care about the most popular devices, we'll use a curated pool.
    "testSpecArn": "arn:aws:devicefarm:us-west-2::upload:20fcf771-eae3-4137-
aa76-92e17fb3131b",
    "poolArn": "arn:aws:devicefarm:us-
west-2::devicepool:4a869d91-6f17-491f-9a95-0a601aee2406",
    "namePrefix": "MyAppTest",
    # This is our test package. This tutorial won't go into how to make these.
    "testPackage": "tests.zip",
}

client = boto3.client("devicefarm")

unique = (
    config["namePrefix"]
    + "-"
    + (datetime.date.today().isoformat())
    + ("".join(random.sample(string.ascii_letters, 8)))
)

print(
    f"The unique identifier for this run is '{unique}'. All uploads will be
 prefixed "
    f"with this."
)


def upload_df_file(filename, type_, mime="application/octet-stream"):
    upload_response = client.create_upload(
        projectArn=config["projectArn"],
        name=unique + "_" + os.path.basename(filename),
        type=type_,
        contentType=mime,
    )
    upload_arn = upload_response["upload"]["arn"]
    # Extract the URL of the upload and use Requests to upload it.
```

```
    upload_url = upload_response["upload"]["url"]
    with open(filename, "rb") as file_stream:
        print(
            f"Uploading {filename} to Device Farm as "
            f"{upload_response['upload']['name']}... ",
            end="",
        )
        put_req = requests.put(
            upload_url, data=file_stream, headers={"content-type": mime}
        )
        print(" done")
        if not put_req.ok:
            raise Exception(f"Couldn't upload. Requests says: {put_req.reason}")
    started = datetime.datetime.now()
    while True:
        print(
            f"Upload of {filename} in state {upload_response['upload']['status']}
 "
            f"after " + str(datetime.datetime.now() - started)
        )
        if upload_response["upload"]["status"] == "FAILED":
            raise Exception(
                f"The upload failed processing. Device Farm says the reason is:
 \n"
                f"{+upload_response['upload']['message']}"
            )
        if upload_response["upload"]["status"] == "SUCCEEDED":
            break
        time.sleep(5)
        upload_response = client.get_upload(arn=upload_arn)
    print("")
    return upload_arn


our_upload_arn = upload_df_file(config["appFilePath"], "ANDROID_APP")
our_test_package_arn = upload_df_file(
    config["testPackage"], "APPIUM_PYTHON_TEST_PACKAGE"
)
print(our_upload_arn, our_test_package_arn)

response = client.schedule_run(
    projectArn=config["projectArn"],
    appArn=our_upload_arn,
    devicePoolArn=config["poolArn"],
```

```
    name=unique,
    test={
        "type": "APPIUM_PYTHON",
        "testSpecArn": config["testSpecArn"],
        "testPackageArn": our_test_package_arn,
    },
)
run_arn = response["run"]["arn"]
start_time = datetime.datetime.now()
print(f"Run {unique} is scheduled as arn {run_arn} ")

state = "UNKNOWN"
try:
    while True:
        response = client.get_run(arn=run_arn)
        state = response["run"]["status"]
        if state == "COMPLETED" or state == "ERRORED":
            break
        else:
            print(
                f" Run {unique} in state {state}, total "
                f"time {datetime.datetime.now() - start_time}"
            )
            time.sleep(10)
except:
    client.stop_run(arn=run_arn)
    exit(1)

print(f"Tests finished in state {state} after {datetime.datetime.now() -
 start_time}")
# Pull all the logs.
jobs_response = client.list_jobs(arn=run_arn)
# Save the output somewhere, using the unique value.
save_path = os.path.join(os.getcwd(), "results", unique)
os.mkdir(save_path)
# Save the last run information.
for job in jobs_response["jobs"]:
    job_name = job["name"]
    os.makedirs(os.path.join(save_path, job_name), exist_ok=True)
    # Get each suite within the job.
    suites = client.list_suites(arn=job["arn"])["suites"]
    for suite in suites:
        for test in client.list_tests(arn=suite["arn"])["tests"]:
            # Get the artifacts.
```

```
            for artifact_type in ["FILE", "SCREENSHOT", "LOG"]:
                artifacts = client.list_artifacts(type=artifact_type,
 arn=test["arn"])[
                    "artifacts"
                ]
                for artifact in artifacts:
                    # Replace `:` because it has a special meaning in Windows &
 macOS.
                    path_to = os.path.join(
                        save_path,
                        job_name,
                        suite["name"],
                        test["name"].replace(":", "_"),
                    )
                    os.makedirs(path_to, exist_ok=True)
                    filename = (
                        artifact["type"]
                        + "_"
                        + artifact["name"]
                        + "."
                        + artifact["extension"]
                    )
                    artifact_save_path = os.path.join(path_to, filename)
                    print(f"Downloading {artifact_save_path}")
                    with open(artifact_save_path, "wb") as fn:
                        with requests.get(
                            artifact["url"], allow_redirects=True
                        ) as request:
                            fn.write(request.content)
print("Finished")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateUpload
  - GetRun
  - GetUpload
  - ListArtifacts
  - ListJobs
  - ListSuites

- ListTests
- ScheduleRun
- StopRun

Exemplos de código para Direct Connect usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Direct Connect com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Direct Connect Guia do usuário — Mais informações sobre Direct Connect.
- Direct Connect Referência da API — Detalhes sobre todas as Direct Connect ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para Direct Connect usar AWS SDKs
  - Comece com conexões de rede dedicadas usando a CLI

Cenários para Direct Connect usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns Direct Connect com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas Direct Connect ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Comece com conexões de rede dedicadas usando a CLI

Comece com conexões de rede dedicadas usando a CLI

O exemplo de código a seguir mostra como:

- Use ec2 CreateVpnGateway
- Use ec2 DeleteVpnGateway
- Use ec2 DescribeVpnGateways
- Use a conexão direta CreateConnection
- Use a conexão direta CreatePrivateVirtualInterface
- Use a conexão direta DeleteConnection

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Direct Connect Connection Management Script - Version 6
# This script demonstrates how to create and manage AWS Direct Connect
 connections using the AWS CLI
# This version includes fixes for user input handling and better error reporting

# Set up logging
LOG_FILE="directconnect-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting AWS Direct Connect script v6"

# Function to check for errors in command output
```

```
check_error() {
    local output=$1
    local command=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $command"
        echo "Output: $output"
        cleanup_resources
        exit 1
    fi
}

# Function to wait for VGW to be available
wait_for_vgw() {
    local vgw_id=$1
    local max_attempts=30
    local attempt=1

    echo "Waiting for virtual private gateway $vgw_id to become available..."

    while [ $attempt -le $max_attempts ]; do
        VGW_STATE=$(aws ec2 describe-vpn-gateways --vpn-gateway-ids "$vgw_id" --
query 'VpnGateways[0].State' --output text)

        if [ "$VGW_STATE" == "available" ]; then
            echo "Virtual private gateway is now available"
            return 0
        elif [ "$VGW_STATE" == "failed" ]; then
            echo "Virtual private gateway failed to become available"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: VGW state is $VGW_STATE, waiting 10
 seconds..."
        sleep 10
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for VGW to become available"
    return 1
}

# Function to wait for connection to be available
wait_for_connection() {
```

```
    local connection_id=$1
    local max_attempts=60
    local attempt=1

    echo "Waiting for connection $connection_id to become available..."
    echo "Note: This can take 30+ minutes in production as AWS provisions the
 physical connection"

    while [ $attempt -le $max_attempts ]; do
        CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$connection_id" --query 'connections[0].connectionState' --output text)

        if [ "$CONNECTION_STATE" == "available" ]; then
            echo "Connection is now available"
            return 0
        elif [ "$CONNECTION_STATE" == "rejected" ] || [ "$CONNECTION_STATE" ==
 "deleted" ]; then
            echo "Connection failed with state: $CONNECTION_STATE"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: Connection state is
 $CONNECTION_STATE, waiting 30 seconds..."
        sleep 30
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for connection to become available"
    return 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete virtual interfaces if they exist
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "Deleting private virtual interface: $PRIVATE_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PRIVATE_VIF_ID"
    fi

    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "Deleting public virtual interface: $PUBLIC_VIF_ID"
```

```
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PUBLIC_VIF_ID"
    fi

    # Delete connection if it exists
    if [ -n "$CONNECTION_ID" ]; then
        echo "Deleting connection: $CONNECTION_ID"
        aws directconnect delete-connection --connection-id "$CONNECTION_ID"
    fi

    # Delete VGW if it exists
    if [ -n "$VGW_ID" ]; then
        echo "Deleting virtual private gateway: $VGW_ID"
        aws ec2 delete-vpn-gateway --vpn-gateway-id "$VGW_ID"
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 6)
CONNECTION_NAME="DxConn-${RANDOM_ID}"

# Step 1: List available Direct Connect locations
echo "Listing available Direct Connect locations..."
LOCATIONS_OUTPUT=$(aws directconnect describe-locations)
check_error "$LOCATIONS_OUTPUT" "describe-locations"
echo "$LOCATIONS_OUTPUT"

# Extract the first location code for demonstration purposes
LOCATION_CODE=$(aws directconnect describe-locations --query
 'locations[0].locationCode' --output text)
if [ -z "$LOCATION_CODE" ] || [ "$LOCATION_CODE" == "None" ]; then
    echo "Error: Could not extract location code from the output."
    exit 1
fi
echo "Using location: $LOCATION_CODE"

# Step 2: Create a dedicated connection
echo "Creating a dedicated connection at location $LOCATION_CODE with bandwidth
 1Gbps..."
CONNECTION_OUTPUT=$(aws directconnect create-connection \
    --location "$LOCATION_CODE" \
    --bandwidth "1Gbps" \
    --connection-name "$CONNECTION_NAME" \
    --tags key=project,value=doc-smith key=tutorial,value=aws-direct-connect-gs)
```

```
check_error "$CONNECTION_OUTPUT" "create-connection"
echo "$CONNECTION_OUTPUT"

# Extract connection ID directly from the output
CONNECTION_ID=$(echo "$CONNECTION_OUTPUT" | grep -o '"connectionId": "[^"]*' |
 cut -d'"' -f4)
if [ -z "$CONNECTION_ID" ]; then
    echo "Error: Could not extract connection ID from the output."
    exit 1
fi
echo "Connection created with ID: $CONNECTION_ID"

# Step 3: Describe the connection
echo "Retrieving connection details..."
DESCRIBE_OUTPUT=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID")
check_error "$DESCRIBE_OUTPUT" "describe-connections"
echo "$DESCRIBE_OUTPUT"

# Step 4: Update the connection name
NEW_CONNECTION_NAME="${CONNECTION_NAME}-updated"
echo "Updating connection name to $NEW_CONNECTION_NAME..."
UPDATE_OUTPUT=$(aws directconnect update-connection \
    --connection-id "$CONNECTION_ID" \
    --connection-name "$NEW_CONNECTION_NAME")
check_error "$UPDATE_OUTPUT" "update-connection"
echo "$UPDATE_OUTPUT"

# Step 5: Check if we can download the LOA-CFA
# Note: In a real scenario, the LOA-CFA might not be immediately available
echo "Attempting to download the LOA-CFA (this may not be available yet)..."
LOA_OUTPUT=$(aws directconnect describe-loa --connection-id "$CONNECTION_ID"
 2>&1)
if echo "$LOA_OUTPUT" | grep -i "error" > /dev/null; then
    echo "LOA-CFA not available yet. This is expected for newly created
 connections."
    echo "The LOA-CFA will be available once AWS begins provisioning your
 connection."
else
    LOA_CONTENT=$(echo "$LOA_OUTPUT" | grep -o '"loaContent": "[^"]*' | cut -d'"'
 -f4)
    echo "$LOA_CONTENT" | base64 --decode > "loa-cfa-${CONNECTION_ID}.pdf"
    echo "LOA-CFA downloaded to loa-cfa-${CONNECTION_ID}.pdf"
fi
```

```
# Step 6: Create a virtual private gateway (required for private virtual
 interface)
echo "Creating a virtual private gateway..."
VGW_OUTPUT=$(aws ec2 create-vpn-gateway --type ipsec.1 \
    --tag-specifications 'ResourceType=vpn-gateway,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=aws-direct-connect-gs}]')
check_error "$VGW_OUTPUT" "create-vpn-gateway"
echo "$VGW_OUTPUT"

# Extract VGW ID directly from the output
VGW_ID=$(echo "$VGW_OUTPUT" | grep -o '"VpnGatewayId": "[^"]*' | cut -d'"' -f4)
if [ -z "$VGW_ID" ]; then
    echo "Error: Could not extract VPN gateway ID from the output."
    exit 1
fi
echo "Virtual private gateway created with ID: $VGW_ID"

# Wait for VGW to become available
if ! wait_for_vgw "$VGW_ID"; then
    echo "Failed to wait for VGW to become available. Skipping virtual interface
 creation."
    VIF_CREATION_SKIPPED=true
else
    VIF_CREATION_SKIPPED=false
fi

# Step 7: Create a private virtual interface (only if VGW is available)
if [ "$VIF_CREATION_SKIPPED" = false ]; then
    echo "Creating a private virtual interface..."
    PRIVATE_VIF_OUTPUT=$(aws directconnect create-private-virtual-interface \
        --connection-id "$CONNECTION_ID" \
        --new-private-virtual-interface '{
            "virtualInterfaceName": "PrivateVIF-'"$RANDOM_ID"'",
            "vlan": 100,
            "asn": 65000,
            "authKey": "'"$RANDOM_ID"'key",
            "amazonAddress": "192.168.1.1/30",
            "customerAddress": "192.168.1.2/30",
            "addressFamily": "ipv4",
            "virtualGatewayId": "'"$VGW_ID"'"
        }' 2>&1)

    if echo "$PRIVATE_VIF_OUTPUT" | grep -i "error" > /dev/null; then
```

```
        echo "Could not create private virtual interface. This is expected if the
 connection is not yet available."
        echo "Error: $PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=""
    else
        echo "$PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=$(echo "$PRIVATE_VIF_OUTPUT" | grep -o
 '"virtualInterfaceId": "[^"]*' | cut -d'"' -f4)
        echo "Private virtual interface created with ID: $PRIVATE_VIF_ID"
    fi
else
    echo "Skipping private virtual interface creation due to VGW not being
 available"
    PRIVATE_VIF_ID=""
fi

# Step 8: Check connection state and provide guidance for public virtual
 interface
CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID" --query 'connections[0].connectionState' --output text)
echo "Current connection state: $CONNECTION_STATE"

if [ "$CONNECTION_STATE" != "available" ]; then
    echo ""
    echo "============================================"
    echo "CONNECTION NOT YET AVAILABLE"
    echo "============================================"
    echo "The connection is in '$CONNECTION_STATE' state."
    echo "In production, you would:"
    echo "1. Wait for AWS to provision the connection (can take 30+ minutes)"
    echo "2. Download the LOA-CFA when available"
    echo "3. Provide the LOA-CFA to your network provider for cross-connect"
    echo "4. Create virtual interfaces once connection is 'available'"
    echo ""

    # Ask if user wants to wait for connection to become available
    echo ""
    echo "============================================"
    echo "CONNECTION WAIT CONFIRMATION"
    echo "============================================"
    echo -n "Do you want to wait for the connection to become available? (y/n): "
    read -r WAIT_CHOICE

    if [[ "$WAIT_CHOICE" =~ ^[Yy]$ ]]; then
```

```
        if wait_for_connection "$CONNECTION_ID"; then
            echo "Connection is now available! You could now create virtual
 interfaces."
        else
            echo "Connection did not become available within the timeout period."
        fi
    else
        echo "Skipping wait for connection availability."
    fi
else
    echo "Connection is available! Virtual interfaces can be created."
fi

# Step 9: List all virtual interfaces
echo "Listing all virtual interfaces..."
VIF_LIST_OUTPUT=$(aws directconnect describe-virtual-interfaces)
check_error "$VIF_LIST_OUTPUT" "describe-virtual-interfaces"
echo "$VIF_LIST_OUTPUT"

# Step 10: Display important information about production usage
echo ""
echo "============================================"
echo "IMPORTANT PRODUCTION NOTES"
echo "============================================"
echo "1. Direct Connect connections take time to be provisioned by AWS"
echo "2. You cannot create virtual interfaces until the connection is
 'available'"
echo "3. For public virtual interfaces, you must own the public IP addresses"
echo "4. LOA-CFA (Letter of Authorization) is needed for cross-connect at the
 facility"
echo "5. This demo creates resources that incur costs (~\$300/month for 1Gbps)"
echo "6. Always test connectivity before putting into production"
echo ""

# Step 11: Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo -n "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
```

```
    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually delete them later."
    echo ""
    echo "Created resources:"
    echo "- Connection ID: $CONNECTION_ID"
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "- Private Virtual Interface ID: $PRIVATE_VIF_ID"
    fi
    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "- Public Virtual Interface ID: $PUBLIC_VIF_ID"
    fi
    echo "- Virtual Private Gateway ID: $VGW_ID"
    echo ""
    echo "Manual cleanup commands:"
    echo "aws directconnect delete-connection --connection-id $CONNECTION_ID"
    echo "aws ec2 delete-vpn-gateway --vpn-gateway-id $VGW_ID"
    echo ""
    echo "Remember: Direct Connect resources incur ongoing costs!"
fi

echo "$(date): Script completed"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateConnection
  - CreatePrivateVirtualInterface
  - CreateVpnGateway
  - DeleteConnection
  - DeleteVirtualInterface
  - DeleteVpnGateway
  - DescribeConnections
  - DescribeLoa
  - DescribeLocations
  - DescribeVirtualInterfaces
  - DescribeVpnGateways
  - UpdateConnection

Exemplos de código para Directory Service usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Directory Service com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Directory Service Guia de administração — Mais informações sobre Directory Service.
- Directory Service Referência da API — Detalhes sobre todas as Directory Service ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para Directory Service usar AWS SDKs
  - Ações para Directory Service usar AWS SDKs
    - Usar DescribeDirectories com uma CLI
    - Usar DescribeTrusts com uma CLI

Exemplos básicos para Directory Service usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Directory Service com AWS SDKs.

Exemplos

- Ações para Directory Service usar AWS SDKs
  - Usar DescribeDirectories com uma CLI
  - Usar DescribeTrusts com uma CLI

Ações para Directory Service usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar Directory Service ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Directory Service.

Exemplos

- Usar DescribeDirectories com uma CLI
- Usar DescribeTrusts com uma CLI

Usar **`DescribeDirectories`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeDirectories`.

CLI

AWS CLI

Para obter detalhes sobre todos os seus diretório

O exemplo `describe-directories` a seguir exibe os detalhes do diretório especificado.

```
aws ds describe-directories \
   --directory-id d-a1b2c3d4e5
```

Saída:

```
{
    "DirectoryDescriptions": [
        {
            "DirectoryId": "d-a1b2c3d4e5",
            "Name": "mydirectory.example.com",
            "ShortName": "mydirectory",
            "Size": "Small",
            "Edition": "Standard",
            "Alias": "d-a1b2c3d4e5",
            "AccessUrl": "d-a1b2c3d4e5.awsapps.com",
            "Stage": "Active",
```

```
            "ShareStatus": "Shared",
            "ShareMethod": "HANDSHAKE",
            "ShareNotes": "These are my share notes",
            "LaunchTime": "2019-07-08T15:33:46.327000-07:00",
            "StageLastUpdatedDateTime": "2019-07-08T15:59:12.307000-07:00",
            "Type": "SharedMicrosoftAD",
            "SsoEnabled": false,
            "DesiredNumberOfDomainControllers": 0,
            "OwnerDirectoryDescription": {
                "DirectoryId": "d-b2c3d4e5f6",
                "AccountId": "123456789111",
                "DnsIpAddrs": [
                    "203.113.0.248",
                    "203.113.0.253"
                ],
                "VpcSettings": {
                    "VpcId": "vpc-a1b2c3d4",
                    "SubnetIds": [
                        "subnet-a1b2c3d4",
                        "subnet-d4c3b2a1"
                    ],
                    "AvailabilityZones": [
                        "us-west-2a",
                        "us-west-2c"
                    ]
                }
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeDirectoriesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse comando obtém informações sobre os diretórios que pertencem a essa conta.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Saída:

```
DirectoryId  Name           DnsIpAddrs                     Type
-----------  ----           ----------                     ----
d-123456abcd abcd.example.com {172.31.74.189, 172.31.13.145} SimpleAD
d-123456efgh wifi.example.com {172.31.16.108, 172.31.10.56}  ADConnector
d-123456ijkl lan2.example.com {172.31.10.56, 172.31.16.108}  MicrosoftAD
```

- Para obter detalhes da API, consulte DescribeDirectoriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém informações sobre os diretórios que pertencem a essa conta.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Saída:

```
DirectoryId  Name           DnsIpAddrs                     Type
-----------  ----           ----------                     ----
d-123456abcd abcd.example.com {172.31.74.189, 172.31.13.145} SimpleAD
d-123456efgh wifi.example.com {172.31.16.108, 172.31.10.56}  ADConnector
d-123456ijkl lan2.example.com {172.31.10.56, 172.31.16.108}  MicrosoftAD
```

- Para obter detalhes da API, consulte DescribeDirectoriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTrusts`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTrusts`.

CLI

AWS CLI

Para obter detalhes sobre suas relações de confiança

O exemplo `describe-trusts` a seguir exibe detalhes sobre as relações de confiança do diretório especificado.

```
aws ds describe-trusts \
   --directory-id d-a1b2c3d4e5
```

Saída:

```
{
    "Trusts": [
        {
            "DirectoryId": "d-a1b2c3d4e5",
            "TrustId": "t-9a8b7c6d5e",
            "RemoteDomainName": "other.example.com",
            "TrustType": "Forest",
            "TrustDirection": "Two-Way",
            "TrustState": "Verified",
            "CreatedDateTime": "2017-06-20T18:08:45.614000-07:00",
            "LastUpdatedDateTime": "2019-06-04T10:52:12.410000-07:00",
            "StateLastUpdatedDateTime": "2019-06-04T10:52:12.410000-07:00",
            "SelectiveAuth": "Disabled"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeTrustsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse comando obtém as informações das relações de confiança criadas para o ID de diretório especificado.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Saída:

```
CreatedDateTime          : 7/5/2019 4:55:42 AM
DirectoryId              : d-123456abcd
LastUpdatedDateTime      : 7/5/2019 4:56:04 AM
RemoteDomainName         : contoso.com
SelectiveAuth            : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
```

```
TrustDirection           : One-Way: Incoming
TrustId                  : t-9067157123
TrustState               : Created
TrustStateReason         :
TrustType                : Forest
```

- Para obter detalhes da API, consulte DescribeTrustsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém as informações das relações de confiança criadas para o ID de diretório especificado.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Saída:

```
CreatedDateTime          : 7/5/2019 4:55:42 AM
DirectoryId              : d-123456abcd
LastUpdatedDateTime      : 7/5/2019 4:56:04 AM
RemoteDomainName         : contoso.com
SelectiveAuth            : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
TrustDirection           : One-Way: Incoming
TrustId                  : t-9067157123
TrustState               : Created
TrustStateReason         :
TrustType                : Forest
```

- Para obter detalhes da API, consulte DescribeTrustsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para AWS DMS usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Database Migration Service com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS DMS Guia do desenvolvedor — Mais informações sobre AWS DMS.
- AWS DMS Referência da API — Detalhes sobre todas as AWS DMS ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS DMS usar AWS SDKs
  - Ações para AWS DMS usar AWS SDKs
    - Usar CreateReplicationTask com uma CLI

Exemplos básicos para AWS DMS usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Database Migration Service com AWS SDKs.

Exemplos

- Ações para AWS DMS usar AWS SDKs
  - Usar CreateReplicationTask com uma CLI

Ações para AWS DMS usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS DMS ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Database Migration Service.

Exemplos

- Usar CreateReplicationTask com uma CLI

Usar **`CreateReplicationTask`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateReplicationTask`.

CLI

AWS CLI

Para criar uma tarefa de replicação

O exemplo `create-replication-task` a seguir cria uma tarefa de replicação.

```
aws dms create-replication-task \
    --replication-task-identifier movedata \
    --source-endpoint-arn arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA \
    --target-endpoint-arn arn:aws:dms:us-
east-1:123456789012:endpoint:EOM4SFKCZEYHZBFGAGZT3QEC5U \
    --replication-instance-arn $RI_ARN \
    --migration-type full-load \
    --table-mappings file://table-mappings.json
```

Conteúdo de `table-mappings.json`:

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "prodrep",
                "table-name": "%"
            },
            "rule-action": "include",
            "filters": []
        }
    ]
}
```

Saída:

```
{
```

```
    "ReplicationTask": {
        "ReplicationTaskIdentifier": "moveit2",
        "SourceEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",
        "TargetEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:EOM4SFKCZEYHZBFGAGZT3QEC5U",
        "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T3OM7OUB5NM2LCVZF7JPGJRNUE",
        "MigrationType": "full-load",
        "TableMappings": ...output omitted... ,
        "ReplicationTaskSettings": ...output omitted... ,
        "Status": "creating",
        "ReplicationTaskCreationDate": 1590524772.505,
        "ReplicationTaskArn": "arn:aws:dms:us-
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"
    }
}
```

Para obter mais informações, consulte Trabalhando com tarefas do AWS DMS no Guia do Usuário do AWS Database Migration Service.

- Para obter detalhes da API, consulte CreateReplicationTaskem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria uma nova tarefa de replicação do AWS Database Migration Service que usa CdcStartTime em vez de CdcStartPosition. O MigrationType está definido como “full-load-and-cdc”, o que significa que a tabela de destino deve estar vazia. A nova tarefa é marcada com uma tag que tem uma chave de Estágio e um valor de chave de Teste. Para obter mais informações sobre os valores usados por esse cmdlet, consulte Creating a Task (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) no AWS Database Migration Service User Guide.

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-
east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE"`
  -CdcStartTime "2019-08-08T12:12:12"`
  -CdcStopPosition "server_time:2019-08-09T12:12:12"`
  -MigrationType "full-load-and-cdc"`
  -ReplicationTaskIdentifier "task1"`
```

```
  -ReplicationTaskSetting ""`
  -SourceEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE"`
  -TableMapping "file:////home/testuser/table-mappings.json"`
  -Tag @{"Key"="Stage";"Value"="Test"}`
  -TargetEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- Para obter detalhes da API, consulte CreateReplicationTaskem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma nova tarefa de replicação do AWS Database Migration Service que usa CdcStartTime em vez de CdcStartPosition. O MigrationType está definido como “full-load-and-cdc”, o que significa que a tabela de destino deve estar vazia. A nova tarefa é marcada com uma tag que tem uma chave de Estágio e um valor de chave de Teste. Para obter mais informações sobre os valores usados por esse cmdlet, consulte Creating a Task (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) no AWS Database Migration Service User Guide.

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-
east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE"`
  -CdcStartTime "2019-08-08T12:12:12"`
  -CdcStopPosition "server_time:2019-08-09T12:12:12"`
  -MigrationType "full-load-and-cdc"`
  -ReplicationTaskIdentifier "task1"`
  -ReplicationTaskSetting ""`
  -SourceEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE"`
  -TableMapping "file:////home/testuser/table-mappings.json"`
  -Tag @{"Key"="Stage";"Value"="Test"}`
  -TargetEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- Para obter detalhes da API, consulte CreateReplicationTaskem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para o Amazon DocumentDB usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon DocumentDB (com compatibilidade com o MongoDB) com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon DocumentDB: mais informações sobre o Amazon DocumentDB.
- Referência de API do Amazon DocumentDB: detalhes sobre todas as ações disponíveis do Amazon DocumentDB.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para o Amazon DocumentDB usando AWS SDKs
  - Introdução aos bancos de dados de documentos
- Exemplos de tecnologia sem servidor para o Amazon DocumentDB
  - Invocar uma função do Lambda de um acionador do Amazon DocumentDB

Cenários para o Amazon DocumentDB usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon DocumentDB com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon DocumentDB ou combinadas com outras. Serviços da AWS Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução aos bancos de dados de documentos

Introdução aos bancos de dados de documentos

O exemplo de código a seguir mostra como:

- Criar um grupo de sub-redes de banco de dados
- Criar um cluster do DocumentDB.
- Criar uma instância do DocumentDB.
- Configurar a segurança e a conectividade.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

################################################################################
# Configuration
################################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
```

```
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

###############################################################################
# Logging
###############################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""

###############################################################################
# Region pre-check
###############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

###############################################################################
# Error handler
###############################################################################
handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
```

```
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

################################################################################
# Wait function
################################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"
```

```
        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound
\|not found"; then
            echo "  $resource_type '$resource_id' deleted."
```

```
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

################################################################################
# Cleanup
################################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG
 ingress rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed
 to delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi
```

```
    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete
 cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}

##############################################################################
# Step 1: Generate password and store in Secrets Manager
##############################################################################
echo "==========================================="
echo "Step 1: Create master password in Secrets Manager"
echo "==========================================="
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
```

```
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "==========================================="
echo "Step 2: Find default VPC and subnets"
echo "==========================================="
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
```

```
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""
```

```
##############################################################################
# Step 3: Create subnet group
##############################################################################
echo "============================================="
echo "Step 3: Create DocumentDB subnet group"
echo "============================================="
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

##############################################################################
# Step 4: Create DocumentDB cluster
##############################################################################
echo "============================================="
echo "Step 4: Create DocumentDB cluster"
echo "============================================="
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
```

```
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "==========================================="
echo "Step 5: Create DocumentDB instance"
echo "==========================================="
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""
```

```
wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

##############################################################################
# Step 6: Get cluster endpoint and security group
##############################################################################
echo "============================================"
echo "Step 6: Get cluster endpoint and security group"
echo "============================================"
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
##############################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null
 || true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
```

```
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "==========================================="
echo "Step 8: Download Amazon DocumentDB CA certificate"
```

```
echo "==========================================="
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

##############################################################################
# Step 9: Display connection information
##############################################################################
echo "==========================================="
echo "CONNECTION INFORMATION"
echo "==========================================="
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
```

```
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH}
 \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString
 --output text)"
echo ""

################################################################################
# Step 10: Cleanup
################################################################################
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance

- DeleteDbSubnetGroup
- DeleteSecret
- DescribeDbClusters
- DescribeDbInstances
- DescribeSubnets
- DescribeVpcs
- GetSecretValue
- RevokeSecurityGroupIngress
- Wait

Exemplos de tecnologia sem servidor para o Amazon DocumentDB

Os exemplos de código a seguir mostram como usar o Amazon DocumentDB com AWS SDKs.

Exemplos

- Invocar uma função do Lambda de um acionador do Amazon DocumentDB

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Processando um evento do Amazon DocumentDB com o Lambda ao usar .NET.

```
using Amazon.Lambda.Core;
```

```
using System.Text.Json;
using System;
using System.Collections.Generic;
using System.Text.Json.Serialization;
//Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace LambdaDocDb;

public class Function
{

     /// <summary>
    /// Lambda function entry point to process Amazon DocumentDB events.
    /// </summary>
    /// <param name="event">The Amazon DocumentDB event.</param>
    /// <param name="context">The Lambda context object.</param>
    /// <returns>A string to indicate successful processing.</returns>
    public string FunctionHandler(Event evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Events)
        {
            ProcessDocumentDBEvent(record, context);
        }

        return "OK";
    }

     private void ProcessDocumentDBEvent(DocumentDBEventRecord record,
 ILambdaContext context)
    {

        var eventData = record.Event;
        var operationType = eventData.OperationType;
        var databaseName = eventData.Ns.Db;
        var collectionName = eventData.Ns.Coll;
        var fullDocument = JsonSerializer.Serialize(eventData.FullDocument, new
 JsonSerializerOptions { WriteIndented = true });

        context.Logger.LogLine($"Operation type: {operationType}");
        context.Logger.LogLine($"Database: {databaseName}");
```

```
        context.Logger.LogLine($"Collection: {collectionName}");
        context.Logger.LogLine($"Full document:\n{fullDocument}");
    }



    public class Event
    {
        [JsonPropertyName("eventSourceArn")]
        public string EventSourceArn { get; set; }

        [JsonPropertyName("events")]
        public List<DocumentDBEventRecord> Events { get; set; }

        [JsonPropertyName("eventSource")]
        public string EventSource { get; set; }
    }

    public class DocumentDBEventRecord
    {
        [JsonPropertyName("event")]
        public EventData Event { get; set; }
    }

    public class EventData
    {
        [JsonPropertyName("_id")]
        public IdData Id { get; set; }

        [JsonPropertyName("clusterTime")]
        public ClusterTime ClusterTime { get; set; }

        [JsonPropertyName("documentKey")]
        public DocumentKey DocumentKey { get; set; }

        [JsonPropertyName("fullDocument")]
        public Dictionary<string, object> FullDocument { get; set; }

        [JsonPropertyName("ns")]
        public Namespace Ns { get; set; }

        [JsonPropertyName("operationType")]
        public string OperationType { get; set; }
    }
```

```
    public class IdData
    {
        [JsonPropertyName("_data")]
        public string Data { get; set; }
    }

    public class ClusterTime
    {
        [JsonPropertyName("$timestamp")]
        public Timestamp Timestamp { get; set; }
    }

    public class Timestamp
    {
        [JsonPropertyName("t")]
        public long T { get; set; }

        [JsonPropertyName("i")]
        public int I { get; set; }
    }

    public class DocumentKey
    {
        [JsonPropertyName("_id")]
        public Id Id { get; set; }
    }

    public class Id
    {
        [JsonPropertyName("$oid")]
        public string Oid { get; set; }
    }

    public class Namespace
    {
        [JsonPropertyName("db")]
        public string Db { get; set; }

        [JsonPropertyName("coll")]
        public string Coll { get; set; }
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Go.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"

 "github.com/aws/aws-lambda-go/lambda"
)

type Event struct {
 Events []Record `json:"events"`
}

type Record struct {
 Event struct {
  OperationType string `json:"operationType"`
  NS            struct {
   DB   string `json:"db"`
   Coll string `json:"coll"`
  } `json:"ns"`
  FullDocument interface{} `json:"fullDocument"`
 } `json:"event"`
}

func main() {
 lambda.Start(handler)
```

```
}

func handler(ctx context.Context, event Event) (string, error) {
 fmt.Println("Loading function")
 for _, record := range event.Events {
  logDocumentDBEvent(record)
 }

 return "OK", nil
}

func logDocumentDBEvent(record Record) {
 fmt.Printf("Operation type: %s\n", record.Event.OperationType)
 fmt.Printf("db: %s\n", record.Event.NS.DB)
 fmt.Printf("collection: %s\n", record.Event.NS.Coll)
 docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")
 fmt.Printf("Full document: %s\n", string(docBytes))
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
```

```
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do Amazon DocumentDB com o uso do Lambda. JavaScript

```
console.log('Loading function');
exports.handler = async (event, context) => {
    event.events.forEach(record => {
        logDocumentDBEvent(record);
    });
    return 'OK';
};

const logDocumentDBEvent = (record) => {
    console.log('Operation type: ' + record.event.operationType);
    console.log('db: ' + record.event.ns.db);
    console.log('collection: ' + record.event.ns.coll);
    console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

Consumindo um evento do Amazon DocumentDB com o Lambda usando TypeScript

```
import { DocumentDBEventRecord, DocumentDBEventSubscriptionContext } from 'aws-
lambda';

console.log('Loading function');

export const handler = async (
  event: DocumentDBEventSubscriptionContext,
  context: any
): Promise<string> => {
  event.events.forEach((record: DocumentDBEventRecord) => {
    logDocumentDBEvent(record);
  });
  return 'OK';
};

const logDocumentDBEvent = (record: DocumentDBEventRecord): void => {
  console.log('Operation type: ' + record.event.operationType);
  console.log('db: ' + record.event.ns.db);
  console.log('collection: ' + record.event.ns.coll);
  console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
```

```
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando PHP.

```
<?php

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

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];
```

```
        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Python.

```
import json

def lambda_handler(event, context):
    for record in event.get('events', []):
        log_document_db_event(record)
    return 'OK'

def log_document_db_event(record):
    event_data = record.get('event', {})
    operation_type = event_data.get('operationType', 'Unknown')
    db = event_data.get('ns', {}).get('db', 'Unknown')
    collection = event_data.get('ns', {}).get('coll', 'Unknown')
    full_document = event_data.get('fullDocument', {})

    print(f"Operation type: {operation_type}")
    print(f"db: {db}")
    print(f"collection: {collection}")
```

```
    print("Full document:", json.dumps(full_document, indent=2))
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::documentdb::{DocumentDbEvent, DocumentDbInnerEvent},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<DocumentDbEvent>) ->Result<(),
 Error> {

    tracing::info!("Event Source ARN: {:?}", event.payload.event_source_arn);
    tracing::info!("Event Source: {:?}", event.payload.event_source);

    let records = &event.payload.events;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_document_db_event(record);
```

```
    }

    tracing::info!("Document db records processed");

    // Prepare the response
    Ok(())

}

fn log_document_db_event(record: &DocumentDbInnerEvent)-> Result<(), Error>{
    tracing::info!("Change Event: {:?}", record.event);

    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Exemplos de código para o DynamoDB usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon DynamoDB com AWS um kit de desenvolvimento de software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS as contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes em todo o mundo AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Mais atributos

- Guia do desenvolvedor do DynamoDB: mais informações sobre o DynamoDB.
- Referência da API DynamoDB: detalhes sobre todas as ações disponíveis do DynamoDB.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de uso do DynamoDB AWS SDKs
  - Olá, DynamoDB
  - Aprenda as noções básicas do DynamoDB com um AWS SDK
  - Ações para o DynamoDB usando AWS SDKs
    - Use BatchExecuteStatementcom um AWS SDK
    - Use BatchGetItemcom um AWS SDK ou CLI
    - Use BatchWriteItemcom um AWS SDK ou CLI
    - Use CreateTablecom um AWS SDK ou CLI
    - Use DeleteItemcom um AWS SDK ou CLI
    - Use DeleteTablecom um AWS SDK ou CLI
    - Use DescribeTablecom um AWS SDK ou CLI
    - Use DescribeTimeToLivecom um AWS SDK ou CLI
    - Use ExecuteStatementcom um AWS SDK
    - Use GetItemcom um AWS SDK ou CLI
    - Use ListTablescom um AWS SDK ou CLI
    - Use PutItemcom um AWS SDK ou CLI

    - Use o Query com um AWS SDK ou CLI
    - Use o Scan com um AWS SDK ou CLI
    - Use UpdateItemcom um AWS SDK ou CLI
    - Use UpdateTablecom um AWS SDK ou CLI
    - Use UpdateTimeToLivecom um AWS SDK ou CLI
- Cenários para o uso do DynamoDB AWS SDKs
  - Acelere as leituras do DynamoDB com o DAX usando um AWS SDK
  - Trabalhe com cenários avançados do Índice Secundário Global do DynamoDB usando AWS Command Line Interface v2 da
  - Criar uma aplicação para enviar dados para uma tabela do DynamoDB
  - Compare vários valores com um único atributo no DynamoDB com um AWS SDK
  - Atualize condicionalmente um item do DynamoDB com um TTL usando um AWS SDK
  - Conecte-se a uma instância local do DynamoDB usando um AWS SDK
  - Conte operadores de expressão no DynamoDB com um AWS SDK
  - Crie uma API REST do API Gateway para rastrear COVID-19 dados
  - Criar uma aplicação de mensageiro com o Step Functions
  - Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
  - Crie uma tabela do DynamoDB com um índice secundário global usando o AWS SDK
  - Crie uma tabela do DynamoDB com configuração de taxa de transferência quente usando um AWS SDK
  - Criar uma aplicação Web para monitorar dados do DynamoDB
  - Criar uma aplicação de chat websocket com o API Gateway
  - Crie um item do DynamoDB com um TTL usando um AWS SDK
  - Crie e gerencie tabelas globais do DynamoDB Multi-Region com forte consistência usando um AWS SDK
  - Crie e gerencie tabelas globais do DynamoDB demonstrando o MREC usando um AWS SDK
  - Exclua dados do DynamoDB usando instruções DELETE do PartiQL com um AWS SDK
  - Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK
  - Conceitos básicos de bancos de dados NoSQL

  - Insira dados do DynamoDB usando instruções INSERT do PartiQL com um AWS SDK

- Invocar uma função do Lambda em um navegador
- Gerencie os índices secundários globais do DynamoDB usando AWS Command Line Interface v2 da
- Gerencie políticas baseadas em recursos do DynamoDB usando AWS Command Line Interface v2 da
- Monitore o desempenho do Amazon DynamoDB usando um AWS SDK
- Execute operações avançadas de consulta do DynamoDB usando um AWS SDK
- Execute operações de lista no DynamoDB com um AWS SDK
- Execute operações de mapa no DynamoDB com um AWS SDK
- Execute operações de conjunto no DynamoDB com um AWS SDK
- Consulte uma tabela do DynamoDB usando lotes de instruções partiQL e um AWS SDK
- Consulte uma tabela do DynamoDB usando partiQL e um AWS SDK
- Consulte uma tabela do DynamoDB usando um índice secundário global com um AWS SDK
- Consulte uma tabela do DynamoDB usando uma condição begins_with com um AWS SDK
- Consulte uma tabela do DynamoDB usando um intervalo de datas na chave de classificação com um AWS SDK
- Consulte uma tabela do DynamoDB com uma expressão de filtro complexa com um AWS SDK
- Consulte uma tabela do DynamoDB com uma expressão de filtro dinâmico com um AWS SDK
- Consulte uma tabela do DynamoDB com uma expressão de filtro e limite com uma AWS SDK
- Consulte uma tabela do DynamoDB com atributos aninhados usando um AWS SDK
- Consulte uma tabela do DynamoDB com paginação usando um AWS SDK
- Consulte uma tabela do DynamoDB com leituras altamente consistentes usando um AWS SDK
- Consulte dados do DynamoDB usando instruções PARTIQL SELECT com um AWS SDK
- Consulte uma tabela do DynamoDB para itens TTL usando um AWS SDK
- Consulte tabelas do DynamoDB usando padrões de data e hora com um AWS SDK
- Salve EXIF e outras informações da imagem usando um AWS SDK
- Configure o controle de Attribute-Based acesso para o DynamoDB usando AWS Command Line Interface v2 da
- Entenda a ordem da expressão de atualização no DynamoDB com um AWS SDK
- Atualize uma configuração de tabela do DynamoDB com taxa de transferência quente usando um AWS SDK

- Atualizar um item do DynamoDB com um TTL usando um AWS SDK
- Atualize os dados do DynamoDB usando instruções PARTIQL UPDATE com um AWS SDK
- Usar o API Gateway para invocar uma função do Lambda
- Usar Step Functions para invocar funções do Lambda
- Use um modelo de documento para o DynamoDB usando um AWS SDK
- Use um modelo de persistência de objetos de alto nível para o DynamoDB usando um AWS SDK
- Use operações de contador atômico no DynamoDB com um AWS SDK
- Use operações condicionais no DynamoDB com um AWS SDK
- Use nomes de atributos de expressão no DynamoDB com um AWS SDK
- Usar eventos programados para chamar uma função do Lambda
- Trabalhe com índices secundários locais do DynamoDB usando AWS Command Line Interface v2 da
- Trabalhe com o DynamoDB Streams e use Time-to-Live AWS Command Line Interface v2 da
- Trabalhe com tabelas globais do DynamoDB e replicação multirregional com consistência eventual (MREC) usando AWS Command Line Interface v2 da
- Trabalhe com a marcação de recursos do DynamoDB usando AWS Command Line Interface v2 da
- Trabalhe com a criptografia de tabelas do DynamoDB usando AWS Command Line Interface v2 da

- Exemplos de tecnologia sem servidor para o DynamoDB
  - Invocar uma função do Lambda em um gatilho do DynamoDB
  - Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB
- AWS contribuições da comunidade para o DynamoDB
  - Compilar e testar uma aplicação com tecnologia sem servidor

Exemplos básicos de uso do DynamoDB AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon DynamoDB com SDKs da AWS .

Exemplos

- Olá, DynamoDB
- Aprenda as noções básicas do DynamoDB com um AWS SDK
- Ações para o DynamoDB usando AWS SDKs
  - Use BatchExecuteStatementcom um AWS SDK
  - Use BatchGetItemcom um AWS SDK ou CLI
  - Use BatchWriteItemcom um AWS SDK ou CLI
  - Use CreateTablecom um AWS SDK ou CLI
  - Use DeleteItemcom um AWS SDK ou CLI
  - Use DeleteTablecom um AWS SDK ou CLI
  - Use DescribeTablecom um AWS SDK ou CLI
  - Use DescribeTimeToLivecom um AWS SDK ou CLI
  - Use ExecuteStatementcom um AWS SDK
  - Use GetItemcom um AWS SDK ou CLI
  - Use ListTablescom um AWS SDK ou CLI
  - Use PutItemcom um AWS SDK ou CLI
  - Use o Query com um AWS SDK ou CLI
  - Use o Scan com um AWS SDK ou CLI
  - Use UpdateItemcom um AWS SDK ou CLI
  - Use UpdateTablecom um AWS SDK ou CLI
  - Use UpdateTimeToLivecom um AWS SDK ou CLI

Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
using Microsoft.Extensions.DependencyInjection;

namespace DynamoDBActions;

/// <summary>
/// A simple example that demonstrates basic DynamoDB operations.
/// </summary>
public class HelloDynamoDB
{
    /// <summary>
    /// HelloDynamoDB lists the existing DynamoDB tables for the default user.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon DynamoDB.
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonDynamoDB>()
            )
            .Build();

        // Now the client is available for injection.
        var dynamoDbClient = host.Services.GetRequiredService<IAmazonDynamoDB>();

        try
        {
            var request = new ListTablesRequest();
            var tableNames = new List<string>();

            var paginatorForTables =
 dynamoDbClient.Paginators.ListTables(request);

            await foreach (var tableName in paginatorForTables.TableNames)
            {
                tableNames.Add(tableName);
            }
```

```
            Console.WriteLine("Welcome to the DynamoDB Hello Service example. " +
                              "\nLet's list your DynamoDB tables:");
            tableNames.ForEach(table =>
            {
                Console.WriteLine($"Table: {table}");
            });
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB service error occurred while
 listing tables. {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing tables.
 {ex.Message}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS dynamodb)
```

```
# Set this project's name.
project("hello_dynamodb")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_dynamodb.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_dynamodb.cpp.

```
#include <aws/core/Aws.h>
```

```
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ListTablesRequest.h>
#include <iostream>

/*
 *  A "Hello DynamoDB" starter application which initializes an Amazon DynamoDB
 (DynamoDB) client and lists the
 *  DynamoDB tables.
 *
 *  main function
 *
 *  Usage: 'hello_dynamodb'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.

    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::DynamoDB::DynamoDBClient dynamodbClient(clientConfig);
        Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
        listTablesRequest.SetLimit(50);
        do {
            const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
 dynamodbClient.ListTables(
                    listTablesRequest);
            if (!outcome.IsSuccess()) {
                std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
                result = 1;
                break;
            }

            for (const auto &tableName: outcome.GetResult().GetTableNames()) {
                std::cout << tableName << std::endl;
            }
```

```
            listTablesRequest.SetExclusiveStartTableName(
                    outcome.GetResult().GetLastEvaluatedTableName());

        } while (!listTablesRequest.GetExclusiveStartTableName().empty());
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

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

```
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request =
 ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

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

```
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
}
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para obter mais detalhes sobre como trabalhar com o DynamoDB AWS SDK para JavaScript em, consulte Programando o DynamoDB com. JavaScript

```
import { ListTablesCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new ListTablesCommand({});

  const response = await client.send(command);
  console.log(response.TableNames.join("\n"));
  return response;
};
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para JavaScript da API.

Python

SDK for Python

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
This is a test output. Examples for AWS SDK for Python
for AWS DynamoDB will be available here soon.
```

- Para obter detalhes da API, consulte a ListTablesReferência da API AWS SDK for Python.

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3

# Create a DynamoDB client using the default credentials and region
dynamodb = boto3.client("dynamodb")

# Initialize a paginator for the list_tables operation
paginator = dynamodb.get_paginator("list_tables")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(Limit=10)

# List the tables in the current AWS account
print("Here are the DynamoDB tables in your account:")

# Use pagination to list all tables
```

```
table_names = []

for page in page_iterator:
    for table_name in page.get("TableNames", []):
        print(f"- {table_name}")
        table_names.append(table_name)

if not table_names:
    print("You don't have any DynamoDB tables in your account.")
else:
    print(f"\nFound {len(table_names)} tables.")
```

- Para obter detalhes da API, consulte a ListTablesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-dynamodb'
require 'logger'

# DynamoDBManager is a class responsible for managing DynamoDB operations
# such as listing all tables in the current AWS account.
class DynamoDBManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all DynamoDB tables in the current AWS account.
  def list_tables
```

```
    @logger.info('Here are the DynamoDB tables in your account:')

    paginator = @client.list_tables(limit: 10)
    table_names = []

    paginator.each_page do |page|
      page.table_names.each do |table_name|
        @logger.info("- #{table_name}")
        table_names << table_name
      end
    end

    if table_names.empty?
      @logger.info("You don't have any DynamoDB tables in your account.")
    else
      @logger.info("\nFound #{table_names.length} tables.")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  dynamodb_client = Aws::DynamoDB::Client.new
  manager = DynamoDBManager.new(dynamodb_client)
  manager.list_tables
end
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para Ruby da API.

Aprenda as noções básicas do DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// This example application performs the following basic Amazon DynamoDB
/// functions:
///     CreateTableAsync
///     PutItemAsync
///     UpdateItemAsync
///     BatchWriteItemAsync
///     GetItemAsync
///     DeleteItemAsync
///     Query
///     Scan
///     DeleteItemAsync.
/// </summary>
public class DynamoDbBasics
{
    public static bool IsInteractive = true;

    // Separator for the console display.
    private static readonly string SepBar = new string('-', 80);

    /// <summary>
    /// The main entry point for the DynamoDB Basics example application.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon DynamoDB.
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
```

```
                services.AddAWSService<IAmazonDynamoDB>()
                    .AddTransient<DynamoDbWrapper>())
            .Build();

        // Now the wrapper is available for injection.
        var dynamoDbWrapper =
host.Services.GetRequiredService<DynamoDbWrapper>();

        var tableName = "movie_table";

        var movieFileName = @"movies.json";

        DisplayInstructions();

        // Create a new table and wait for it to be active.
        Console.WriteLine($"Creating the new table: {tableName}");

        var success = await dynamoDbWrapper.CreateMovieTableAsync(tableName);

        Console.WriteLine(success
            ? $"\nTable: {tableName} successfully created."
            : $"\nCould not create {tableName}.");

        WaitForEnter();

        // Add a single new movie to the table.
        var newMovie = new Movie
        {
            Year = 2021,
            Title = "Spider-Man: No Way Home",
        };

        success = await dynamoDbWrapper.PutItemAsync(newMovie, tableName);
        if (success)
        {
            Console.WriteLine($"Added {newMovie.Title} to the table.");
        }
        else
        {
            Console.WriteLine("Could not add movie to table.");
        }

        WaitForEnter();
```

```
        // Update the new movie by adding a plot and rank.
        var newInfo = new MovieInfo
        {
            Plot = "With Spider-Man's identity now revealed, Peter asks" +
                   "Doctor Strange for help. When a spell goes wrong, dangerous"
+
                   "foes from other worlds start to appear, forcing Peter to" +
                   "discover what it truly means to be Spider-Man.",
            Rank = 9,
        };

        success = await dynamoDbWrapper.UpdateItemAsync(newMovie, newInfo,
tableName);
        if (success)
        {
            Console.WriteLine($"Successfully updated the movie:
{newMovie.Title}");
        }
        else
        {
            Console.WriteLine("Could not update the movie.");
        }

        WaitForEnter();

        // Add a batch of movies to the DynamoDB table from a list of
        // movies in a JSON file.
        var itemCount = await dynamoDbWrapper.BatchWriteItemsAsync(movieFileName,
tableName);
        Console.WriteLine($"Added {itemCount} movies to the table.");

        WaitForEnter();

        // Get a movie by key. (partition + sort)
        var lookupMovie = new Movie
        {
            Title = "Jurassic Park",
            Year = 1993,
        };

        Console.WriteLine("Looking for the movie \"Jurassic Park\".");
        var item = await dynamoDbWrapper.GetItemAsync(lookupMovie, tableName);
        if (item?.Count > 0)
        {
```

```
            dynamoDbWrapper.DisplayItem(item);
        }
        else
        {
            Console.WriteLine($"Couldn't find {lookupMovie.Title}");
        }

        WaitForEnter();

        // Delete a movie.
        var movieToDelete = new Movie
        {
            Title = "The Town",
            Year = 2010,
        };

        success = await dynamoDbWrapper.DeleteItemAsync(tableName,
 movieToDelete);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {movieToDelete.Title}.");
        }
        else
        {
            Console.WriteLine($"Could not delete {movieToDelete.Title}.");
        }

        WaitForEnter();

        // Use Query to find all the movies released in 2010.
        int findYear = 2010;
        Console.WriteLine($"Movies released in {findYear}");
        var queryCount = await dynamoDbWrapper.QueryMoviesAsync(tableName,
 findYear);
        Console.WriteLine($"Found {queryCount} movies released in {findYear}");

        WaitForEnter();

        // Use Scan to get a list of movies from 2001 to 2011.
        int startYear = 2001;
        int endYear = 2011;
        var scanCount = await dynamoDbWrapper.ScanTableAsync(tableName,
 startYear, endYear);
```

```
        Console.WriteLine($"Found {scanCount} movies released between {startYear}
and {endYear}");

        WaitForEnter();

        // Delete the table.
        success = await dynamoDbWrapper.DeleteTableAsync(tableName);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {tableName}");
        }
        else
        {
            Console.WriteLine($"Could not delete {tableName}");
        }

        Console.WriteLine("The DynamoDB Basics example application is
complete.");

        WaitForEnter();
    }

    /// <summary>
    /// Displays the description of the application on the console.
    /// </summary>
    private static void DisplayInstructions()
    {
        if (!IsInteractive)
        {
            return;
        }

        Console.Clear();
        Console.WriteLine();
        Console.Write(new string(' ', 28));
        Console.WriteLine("DynamoDB Basics Example");
        Console.WriteLine(SepBar);
        Console.WriteLine("This demo application shows the basics of using
DynamoDB with the AWS SDK.");
        Console.WriteLine(SepBar);
        Console.WriteLine("The application does the following:");
        Console.WriteLine("\t1. Creates a table with partition: year and
sort:title.");
```

```
        Console.WriteLine("\t2. Adds a single movie to the table.");
        Console.WriteLine("\t3. Adds movies to the table from moviedata.json.");
        Console.WriteLine("\t4. Updates the rating and plot of the movie that was
 just added.");
        Console.WriteLine("\t5. Gets a movie using its key (partition + sort).");
        Console.WriteLine("\t6. Deletes a movie.");
        Console.WriteLine("\t7. Uses QueryAsync to return all movies released in
 a given year.");
        Console.WriteLine("\t8. Uses ScanAsync to return all movies released
 within a range of years.");
        Console.WriteLine("\t9. Finally, it deletes the table that was just
 created.");
        WaitForEnter();
    }

    /// <summary>
    /// Simple method to wait for the Enter key to be pressed.
    /// </summary>
    private static void WaitForEnter()
    {
        if (IsInteractive)
        {
            Console.WriteLine("\nPress <Enter> to continue.");
            Console.WriteLine(SepBar);
            _ = Console.ReadLine();
        }
    }
}
```

Usar o cliente injetado para operações de tabela.

```
using System.Text.Json;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DataModel;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.DynamoDBv2.Model;

namespace DynamoDBActions;

/// <summary>
/// Methods of this class perform Amazon DynamoDB operations.
```

```
/// </summary>
public class DynamoDbWrapper
{
    private readonly IAmazonDynamoDB _amazonDynamoDB;

    /// <summary>
    /// Constructor for the DynamoDbWrapper class.
    /// </summary>
    /// <param name="amazonDynamoDB">The injected DynamoDB client.</param>
    public DynamoDbWrapper(IAmazonDynamoDB amazonDynamoDB)
    {
        _amazonDynamoDB = amazonDynamoDB;
    }
```

Cria uma tabela para conter dados de filmes.

```
    /// <summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    /// </summary>
    /// <param name="tableName">The name of the table to create.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> CreateMovieTableAsync(string tableName)
    {
        try
        {
            var response = await _amazonDynamoDB.CreateTableAsync(new
 CreateTableRequest
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition
                    {
                        AttributeName = "title",
                        AttributeType = ScalarAttributeType.S,
                    },
                    new AttributeDefinition
                    {
                        AttributeName = "year",
```

```
                        AttributeType = ScalarAttributeType.N,
                    },
                },
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement
                    {
                        AttributeName = "year",
                        KeyType = KeyType.HASH,
                    },
                    new KeySchemaElement
                    {
                        AttributeName = "title",
                        KeyType = KeyType.RANGE,
                    },
                },
                BillingMode = BillingMode.PAY_PER_REQUEST,
            });

            // Wait until the table is ACTIVE and then report success.
            Console.Write("Waiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = response.TableDescription.TableName,
            };

            TableStatus status;

            int sleepDuration = 2000;

            do
            {
                Thread.Sleep(sleepDuration);

                var describeTableResponse = await
 _amazonDynamoDB.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
```

```
        }
        catch (ResourceInUseException ex)
        {
            Console.WriteLine($"Table {tableName} already exists. {ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while creating
 table {tableName}. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating table
 {tableName}. {ex.Message}");
            throw;
        }
    }
```

Adiciona um único filme à tabela.

```
    /// <summary>
    /// Adds a new item to the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing informtation for
    /// the movie to add to the table.</param>
    /// <param name="tableName">The name of the table where the item will be
 added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
returns>
    public async Task<bool> PutItemAsync(Movie newMovie, string tableName)
    {
        try
        {
            var item = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
```

```
            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = item,
            };

            await _amazonDynamoDB.PutItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while putting
 item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while putting item.
 {ex.Message}");
            throw;
        }
    }
```

Atualiza um item único em uma tabela.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
    /// <param name="tableName">The name of the table that contains the movie.</
 param>
```

```
    /// <returns>A Boolean value that indicates the success of the operation.</
returns>
    public async Task<bool> UpdateItemAsync(
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
            var updates = new Dictionary<string, AttributeValueUpdate>
            {
                ["info.plot"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { S = newInfo.Plot },
                },

                ["info.rating"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { N = newInfo.Rank.ToString() },
                },
            };

            var request = new UpdateItemRequest
            {
                AttributeUpdates = updates,
                Key = key,
                TableName = tableName,
            };

            await _amazonDynamoDB.UpdateItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} or item was not found.
 {ex.Message}");
            return false;
```

```
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while updating
 item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while updating item.
 {ex.Message}");
            throw;
        }
    }
```

Recupera um único item de uma tabela de filmes.

```
    /// <summary>
    /// Gets information about an existing movie from the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information about
    /// the movie to retrieve.</param>
    /// <param name="tableName">The name of the table containing the movie.</
param>
    /// <returns>A Dictionary object containing information about the item
    /// retrieved.</returns>
    public async Task<Dictionary<string, AttributeValue>> GetItemAsync(Movie
 newMovie, string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new GetItemRequest
            {
                Key = key,
```

```
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.GetItemAsync(request);
            return response.Item;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return new Dictionary<string, AttributeValue>();
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while getting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while getting item.
{ex.Message}");
            throw;
        }
    }
```

Grava um lote de itens na tabela de filmes.

```
    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the DynamoDB table.
    /// </summary>
    /// <param name="movieFileName">The name of the JSON file.</param>
    /// <returns>A generic list of movie objects.</returns>
    public List<Movie> ImportMovies(string movieFileName)
    {
        var moviesList = new List<Movie>();
        if (!File.Exists(movieFileName))
        {
            return moviesList;
        }
```

```
        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();
        var allMovies = JsonSerializer.Deserialize<List<Movie>>(
            json,
            new JsonSerializerOptions
            {
                PropertyNameCaseInsensitive = true
            });

        // Now return the first 250 entries.
        if (allMovies != null && allMovies.Any())
        {
            moviesList = allMovies.GetRange(0, 250);
        }
        return moviesList;
    }

    /// <summary>
    /// Writes 250 items to the movie table.
    /// </summary>
    /// <param name="movieFileName">A string containing the full path to
    /// the JSON file containing movie data.</param>
    /// <param name="tableName">The name of the table to write items to.</param>
    /// <returns>A long integer value representing the number of movies
    /// imported from the JSON file.</returns>
    public async Task<long> BatchWriteItemsAsync(
        string movieFileName, string tableName)
    {
        try
        {
            var movies = ImportMovies(movieFileName);
            if (!movies.Any())
            {
                Console.WriteLine("Couldn't find the JSON file with movie
data.");
                return 0;
            }

            var context = new DynamoDBContextBuilder()
                // Optional call to provide a specific instance of
IAmazonDynamoDB
                .WithDynamoDBClient(() => _amazonDynamoDB)
                .Build();
```

```
            var movieBatch = context.CreateBatchWrite<Movie>(
                new BatchWriteConfig()
                {
                    OverrideTableName = tableName
                });
            movieBatch.AddPutItems(movies);

            Console.WriteLine("Adding imported movies to the table.");
            await movieBatch.ExecuteAsync();

            return movies.Count;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table was not found during batch write operation.
{ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred during batch
write operation. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during batch write operation.
{ex.Message}");
            throw;
        }
    }
```

Exclui um único item da tabela.

```
    /// <summary>
    /// Deletes a single item from a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table from which the item
    /// will be deleted.</param>
```

```
    /// <param name="movieToDelete">A movie object containing the title and
    /// year of the movie to delete.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// delete operation.</returns>
    public async Task<bool> DeleteItemAsync(
        string tableName,
        Movie movieToDelete)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = movieToDelete.Title },
                ["year"] = new AttributeValue { N =
movieToDelete.Year.ToString() },
            };

            var request = new DeleteItemRequest { TableName = tableName, Key =
key, };

            await _amazonDynamoDB.DeleteItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting item.
{ex.Message}");
            throw;
        }
    }
```

Consulta a tabela de filmes lançados em determinado ano.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public async Task<int> QueryMoviesAsync(string tableName, int year)
{
    try
    {
        var movieTable = new TableBuilder(_amazonDynamoDB, tableName)
            .AddHashKey("year", DynamoDBEntryType.Numeric)
            .AddRangeKey("title", DynamoDBEntryType.String)
            .Build();

        var filter = new QueryFilter("year", QueryOperator.Equal, year);

        Console.WriteLine("\nFind movies released in: {year}:");

        var config = new QueryOperationConfig()
        {
            Limit = 10, // 10 items per page.
            Select = SelectValues.SpecificAttributes,
            AttributesToGet = new List<string>
            {
                "title",
                "year",
            },
            ConsistentRead = true,
            Filter = filter,
        };

        // Value used to track how many movies match the
        // supplied criteria.
        var moviesFound = 0;

        var search = movieTable.Query(config);
        do
        {
```

```
                var movieList = await search.GetNextSetAsync();
                moviesFound += movieList.Count;

                foreach (var movie in movieList)
                {
                    DisplayDocument(movie);
                }
            }
            while (!search.IsDone);

            return moviesFound;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while querying
 movies. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while querying movies.
 {ex.Message}");
            throw;
        }
    }
```

Busca na tabela os filmes lançados em um intervalo de anos.

```
    /// <summary>
    /// Scans the table for movies released between the specified years.
    /// </summary>
    /// <param name="tableName">The name of the table to scan.</param>
    /// <param name="startYear">The starting year for the range.</param>
    /// <param name="endYear">The ending year for the range.</param>
    /// <returns>The number of movies found in the specified year range.</
returns>
```

```
    public async Task<int> ScanTableAsync(
        string tableName,
        int startYear,
        int endYear)
    {
        try
        {
            var request = new ScanRequest
            {
                TableName = tableName,
                ExpressionAttributeNames = new Dictionary<string, string>
                {
                    { "#yr", "year" },
                },
                ExpressionAttributeValues = new Dictionary<string,
AttributeValue>
                {
                    { ":y_a", new AttributeValue { N = startYear.ToString() } },
                    { ":y_z", new AttributeValue { N = endYear.ToString() } },
                },
                FilterExpression = "#yr between :y_a and :y_z",
                ProjectionExpression = "#yr, title, info.actors[0],
info.directors, info.running_time_secs",
                Limit = 10 // Set a limit to demonstrate using the
LastEvaluatedKey.
            };

            // Keep track of how many movies were found.
            int foundCount = 0;

            var response = new ScanResponse();
            do
            {
                response = await _amazonDynamoDB.ScanAsync(request);
                foundCount += response.Items.Count;
                response.Items.ForEach(i => DisplayItem(i));
                request.ExclusiveStartKey = response.LastEvaluatedKey;
            }
            while (response?.LastEvaluatedKey?.Count > 0);
            return foundCount;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
```

```
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while scanning
 table. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while scanning table.
 {ex.Message}");
            throw;
        }
    }
```

Exclui a tabela de filmes.

```
    /// <summary>
    /// Deletes a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> DeleteTableAsync(string tableName)
    {
        try
        {
            var request = new DeleteTableRequest
            {
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.DeleteTableAsync(request);

            Console.WriteLine($"Table {response.TableDescription.TableName}
 successfully deleted.");
            return true;

        }
        catch (ResourceNotFoundException ex)
```

```
        {
            Console.WriteLine($"Table {tableName} was not found and cannot be
deleted. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
table {tableName}. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting table
{tableName}. {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O cenário de conceitos básicos do DynamoDB.

```
###############################################################################
# function dynamodb_getting_started_movies
#
# Scenario to create an Amazon DynamoDB table and perform a series of operations
 on the table.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function dynamodb_getting_started_movies() {

  source ./dynamodb_operations.sh

  key_schema_json_file="dynamodb_key_schema.json"
  attribute_definitions_json_file="dynamodb_attr_def.json"
  item_json_file="movie_item.json"
  key_json_file="movie_key.json"
  batch_json_file="batch.json"
  attribute_names_json_file="attribute_names.json"
  attributes_values_json_file="attribute_values.json"

  echo_repeat "*" 88
  echo
  echo "Welcome to the Amazon DynamoDB getting started demo."
  echo
  echo_repeat "*" 88
  echo

  local table_name
  echo -n "Enter a name for a new DynamoDB table: "
```

```
  get_input
  table_name=$get_input_result

  echo '[
  {"AttributeName": "year", "KeyType": "HASH"},
   {"AttributeName": "title", "KeyType": "RANGE"}
  ]' >"$key_schema_json_file"

  echo '[
  {"AttributeName": "year", "AttributeType": "N"},
   {"AttributeName": "title", "AttributeType": "S"}
  ]' >"$attribute_definitions_json_file"

  if dynamodb_create_table -n "$table_name" -a "$attribute_definitions_json_file"
 \
    -k "$key_schema_json_file" 1>/dev/null; then
    echo "Created a DynamoDB table named $table_name"
  else
    errecho "The table failed to create. This demo will exit."
    clean_up
    return 1
  fi

  echo "Waiting for the table to become active...."

  if dynamodb_wait_table_active -n "$table_name"; then
    echo "The table is now active."
  else
    errecho "The table failed to become active. This demo will exit."
    cleanup "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo -n "Enter the title of a movie you want to add to the table: "
  get_input
  local added_title
  added_title=$get_input_result

  local added_year
  get_int_input "What year was it released? "
```

```
  added_year=$get_input_result

  local rating
  get_float_input "On a scale of 1 - 10, how do you rate it? " "1" "10"
  rating=$get_input_result

  local plot
  echo -n "Summarize the plot for me: "
  get_input
  plot=$get_input_result

  echo '{
    "year": {"N" :"'"$added_year"'"},
    "title": {"S" :  "'"$added_title"'"},
    "info": {"M" : {"plot": {"S" : "'"$plot"'"}, "rating":
{"N" :"'"$rating"'"} } }
  }' >"$item_json_file"

  if dynamodb_put_item -n "$table_name" -i "$item_json_file"; then
    echo "The movie '$added_title' was successfully added to the table
'$table_name'."
  else
    errecho "Put item failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo "Let's update your movie '$added_title'."
  get_float_input "You rated it $rating, what new rating would you give it? " "1"
"10"
  rating=$get_input_result

  echo -n "You summarized the plot as '$plot'."
  echo "What would you say now? "
  get_input
  plot=$get_input_result

  echo '{
    "year": {"N" :"'"$added_year"'"},
    "title": {"S" :  "'"$added_title"'"}
```

```
    }' >"$key_json_file"

  echo '{
    ":r": {"N" :"'"$rating"'"},
    ":p": {"S" : "'"$plot"'"}
   }' >"$item_json_file"

  local update_expression="SET info.rating = :r, info.plot = :p"

  if dynamodb_update_item -n "$table_name" -k "$key_json_file" -e
"$update_expression" -v "$item_json_file"; then
    echo "Updated '$added_title' with new attributes."
  else
    errecho "Update item failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo "We will now use batch write to upload 150 movie entries into the table."

  local batch_json
  for batch_json in movie_files/movies_*.json; do
    echo "{ \"$table_name\" : $(<"$batch_json") }" >"$batch_json_file"
    if dynamodb_batch_write_item -i "$batch_json_file" 1>/dev/null; then
      echo "Entries in $batch_json added to table."
    else
      errecho "Batch write failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi
  done

  local title="The Lord of the Rings: The Fellowship of the Ring"
  local year="2001"

  if get_yes_no_input "Let's move on...do you want to get info about '$title'?
(y/n) "; then
    echo '{
  "year": {"N" :"'"$year"'"},
  "title": {"S" :  "'"$title"'"}
```

```
  }' >"$key_json_file"
    local info
    info=$(dynamodb_get_item -n "$table_name" -k "$key_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
      errecho "Get item failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi

    echo "Here is what I found:"
    echo "$info"
  fi

  local ask_for_year=true
  while [[ "$ask_for_year" == true ]]; do
    echo "Let's get a list of movies released in a given year."
    get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
    year=$get_input_result
    echo '{
    "#n": "year"
    }' >"$attribute_names_json_file"

    echo '{
    ":v": {"N" :"'"$year"'"}
    }' >"$attributes_values_json_file"

    response=$(dynamodb_query -n "$table_name" -k "#n=:v" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
      errecho "Query table failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi

    echo "Here is what I found:"
    echo "$response"

    if ! get_yes_no_input "Try another year? (y/n) "; then
      ask_for_year=false
    fi
```

```
  done

  echo "Now let's scan for movies released in a range of years. Enter a year: "
  get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
  local start=$get_input_result

  get_int_input "Enter another year: " "1972" "2018"
  local end=$get_input_result

  echo '{
    "#n": "year"
    }' >"$attribute_names_json_file"

  echo '{
    ":v1": {"N" : "'"$start"'"},
    ":v2": {"N" : "'"$end"'"}
    }' >"$attributes_values_json_file"

  response=$(dynamodb_scan -n "$table_name" -f "#n BETWEEN :v1 AND :v2" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "Scan table failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo "Here is what I found:"
  echo "$response"

  echo
  echo_repeat "*" 88
  echo

  echo "Let's remove your movie '$added_title' from the table."

  if get_yes_no_input "Do you want to remove '$added_title'? (y/n) "; then
    echo '{
  "year": {"N" :"'"$added_year"'"},
  "title": {"S" :  "'"$added_title"'"}
  }' >"$key_json_file"

    if ! dynamodb_delete_item -n "$table_name" -k "$key_json_file"; then
```

```
      errecho "Delete item failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi
  fi

  if get_yes_no_input "Do you want to delete the table '$table_name'? (y/n) ";
 then
    if ! clean_up "$table_name"; then
      return 1
    fi
  else
    if ! clean_up; then
      return 1
    fi
  fi

  return 0
}
```

As funções do DynamoDB usadas nesse cenário.

```
###############################################################################
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to create.
#       -a attribute_definitions -- JSON file path of a list of attributes and
 their types.
#       -k key_schema -- JSON file path of a list of attributes and their key
 types.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_create_table() {
  local table_name attribute_definitions key_schema response
  local option OPTARG # Required to use getopts command in a function.
```

```
 ######################################
 # Function usage explanation
 ######################################
 function usage() {
   echo "function dynamodb_create_table"
   echo "Creates an Amazon DynamoDB table with on-demand billing."
   echo " -n table_name  -- The name of the table to create."
   echo " -a attribute_definitions -- JSON file path of a list of attributes and
their types."
   echo " -k key_schema -- JSON file path of a list of attributes and their key
types."
   echo ""
 }

 # Retrieve the calling parameters.
 while getopts "n:a:k:h" option; do
   case "${option}" in
     n) table_name="${OPTARG}" ;;
     a) attribute_definitions="${OPTARG}" ;;
     k) key_schema="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$table_name" ]]; then
   errecho "ERROR: You must provide a table name with the -n parameter."
   usage
   return 1
 fi

 if [[ -z "$attribute_definitions" ]]; then
   errecho "ERROR: You must provide an attribute definitions json file path the
-a parameter."
   usage
   return 1
```

```
  fi

  if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k
 parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    attribute_definitions:   $attribute_definitions"
  iecho "    key_schema:   $key_schema"
  iecho ""

  response=$(aws dynamodb create-table \
    --table-name "$table_name" \
    --attribute-definitions file://"$attribute_definitions" \
    --billing-mode PAY_PER_REQUEST \
    --key-schema file://"$key_schema" )

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
  fi

  return 0
}

###############################################################################
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#
#  Response:
#       - TableStatus:
#     And:
#       0 - Table is active.
```

```
#       1 - If it fails.
##############################################################################
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_describe_table"
    echo "Describe the status of a DynamoDB table."
    echo "  -n table_name  -- The name of the table."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  local table_status
    table_status=$(
      aws dynamodb describe-table \
        --table-name "$table_name" \
        --output text \
```

```
        --query 'Table.TableStatus'
    )

   local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log "$error_code"
      errecho "ERROR: AWS reports describe-table operation failed.$table_status"
      return 1
    fi

  echo "$table_status"

  return 0
}

###############################################################################
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -i item  -- Path to json file containing the item values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_put_item() {
  local table_name item response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_put_item"
    echo "Put an item into a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -i item  -- Path to json file containing the item values."
    echo ""
  }
```

```
  while getopts "n:i:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""
  iecho ""

  response=$(aws dynamodb put-item \
    --table-name "$table_name" \
    --item file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports put-item operation failed.$response"
```

```
    return 1
  fi

  return 0

}

##############################################################################
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to update.
#       -e update expression  -- An expression that defines one or more
 attributes to be updated.
#       -v values  -- Path to json file containing the update values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_update_item() {
  local table_name keys update_expression values response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_update_item"
    echo "Update an item in a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to update."
    echo " -e update expression  -- An expression that defines one or more
 attributes to be updated."
    echo " -v values  -- Path to json file containing the update values."
    echo ""
  }
```

```
  while getopts "n:k:e:v:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      e) update_expression="${OPTARG}" ;;
      v) values="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi
  if [[ -z "$update_expression" ]]; then
    errecho "ERROR: You must provide an update expression with the -e parameter."
    usage
    return 1
  fi

  if [[ -z "$values" ]]; then
    errecho "ERROR: You must provide a values json file path the -v parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
```

```
  iecho "    keys:   $keys"
  iecho "    update_expression:   $update_expression"
  iecho "    values:   $values"

  response=$(aws dynamodb update-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --update-expression "$update_expression" \
    --expression-attribute-values file://"$values")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_batch_write_item
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the items to write.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_batch_write_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_batch_write_item"
    echo "Write a batch of items into a DynamoDB table."
```

```
    echo " -i item  -- Path to json file containing the items to write."
    echo ""
  }
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""

  response=$(aws dynamodb batch-write-item \
    --request-items file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  return 0
}
```

```
###########################################################################
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to get.
#       [-q query]  -- Optional JMESPath query expression.
#
#  Returns:
#       The item as text output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_get_item() {
  local table_name keys query response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_get_item"
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
  }
  query=""
  while getopts "n:k:q:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi

if [[ -n "$query" ]]; then
  response=$(aws dynamodb get-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --output text \
    --query "$query")
else
  response=$(
    aws dynamodb get-item \
      --table-name "$table_name" \
      --key file://"$keys" \
      --output text
  )
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports get-item operation failed.$response"
  return 1
fi

if [[ -n "$query" ]]; then
```

```
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
 query inserts on some strings.
  else
    echo "$response"
  fi

  return 0
}

##############################################################################
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k key_condition_expression -- The key condition expression.
#       -a attribute_names -- Path to JSON file containing the attribute names.
#       -v attribute_values -- Path to JSON file containing the attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_query() {
  local table_name key_condition_expression attribute_names attribute_values
 projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_query"
    echo "Query a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k key_condition_expression -- The key condition expression."
    echo " -a attribute_names -- Path to JSON file containing the attribute
 names."
    echo " -v attribute_values -- Path to JSON file containing the attribute
 values."
```

```
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) key_condition_expression="${OPTARG}" ;;
      a) attribute_names="${OPTARG}" ;;
      v) attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
 parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_values" ]]; then
```

```
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values")
  else
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -f filter_expression  -- The filter expression.
#       -a expression_attribute_names -- Path to JSON file containing the
 expression attribute names.
```

```
#       -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_scan() {
  local table_name filter_expression expression_attribute_names
 expression_attribute_values projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_scan"
    echo "Scan a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -f filter_expression  -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
 expression attribute names."
    echo " -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      f) filter_expression="${OPTARG}" ;;
      a) expression_attribute_names="${OPTARG}" ;;
      v) expression_attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_names" ]]; then
    errecho "ERROR: You must provide expression attribute names with the -a
parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values")
  else
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
```

```
      --expression-attribute-values file://"$expression_attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###########################################################################
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_delete_item() {
  local table_name keys response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_delete_item"
    echo "Delete an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to delete."
```

```
  echo ""
}
while getopts "n:k:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    k) keys="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    keys:   $keys"
iecho ""

response=$(aws dynamodb delete-item \
  --table-name "$table_name" \
  --key file://"$keys")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
  fi

  return 0

}

##############################################################################
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_delete_table() {
  local table_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function dynamodb_delete_table"
    echo "Deletes an Amazon DynamoDB table."
    echo " -n table_name  -- The name of the table to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
```

```
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho ""

  response=$(aws dynamodb delete-table \
    --table-name "$table_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-table operation failed.$response"
    return 1
  fi

  return 0
}
```

As funções utilitárias usadas nesse cenário.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}
```

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    {
        Aws::Client::ClientConfiguration clientConfig;
        //  1. Create a table with partition: year (N) and sort: title (S).
(CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::dynamodbGettingStartedScenario(clientConfig);

            // 9. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }
```

```
    }

//! Scenario to modify and query a DynamoDB table.
/*!
  \sa dynamodbGettingStartedScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout << "Welcome to the Amazon DynamoDB getting started demo." <<
 std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie.
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate
 it? ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::PutItemRequest putItemRequest;
        putItemRequest.SetTableName(MOVIE_TABLE_NAME);

        putItemRequest.AddItem(YEAR_KEY,

 Aws::DynamoDB::Model::AttributeValue().SetN(year));
        putItemRequest.AddItem(TITLE_KEY,

 Aws::DynamoDB::Model::AttributeValue().SetS(title));

        // Create attribute for the info map.
```

```
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);

        putItemRequest.AddItem(INFO_KEY, infoMapAttribute);

        Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
                putItemRequest);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add an item: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
              << std::endl;

    // 3. Update the rating and plot of the movie by using an update expression.
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);
        plot = askQuestion(Aws::String("You summarized the plot as '") + plot +
                           "'.\nWhat would you say now? ");

        Aws::DynamoDB::Model::UpdateItemRequest request;
        request.SetTableName(MOVIE_TABLE_NAME);
        request.AddKey(TITLE_KEY,
Aws::DynamoDB::Model::AttributeValue().SetS(title));
        request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(year));
```

```
        std::stringstream expressionStream;
        expressionStream << "set " << INFO_KEY << "." << RATING_KEY << " =:r, "
                         << INFO_KEY << "." << PLOT_KEY << " =:p";
        request.SetUpdateExpression(expressionStream.str());
        request.SetExpressionAttributeValues({
                                                    {":r",
Aws::DynamoDB::Model::AttributeValue().SetN(
                                                            rating)},
                                                    {":p",
Aws::DynamoDB::Model::AttributeValue().SetS(
                                                            plot)}
                                                });

        request.SetReturnValues(Aws::DynamoDB::Model::ReturnValue::UPDATED_NEW);

        const Aws::DynamoDB::Model::UpdateItemOutcome &result =
dynamoClient.UpdateItem(
                request);
        if (!result.IsSuccess()) {
            std::cerr << "Error updating movie " + result.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

    // 4. Put 250 movies in the table from moviedata.json.
    {
        std::cout << "Adding movies from a json file to the database." <<
std::endl;
        const size_t MAX_SIZE_FOR_BATCH_WRITE = 25;
        const size_t MOVIES_TO_WRITE = 10 * MAX_SIZE_FOR_BATCH_WRITE;
        Aws::String jsonString = getMovieJSON();
        if (!jsonString.empty()) {
            Aws::Utils::Json::JsonValue json(jsonString);
            Aws::Utils::Array<Aws::Utils::Json::JsonView> movieJsons =
json.View().AsArray();
            Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;

            // To add movies with a cross-section of years, use an appropriate
increment
            // value for iterating through the database.
            size_t increment = movieJsons.GetLength() / MOVIES_TO_WRITE;
```

```
            for (size_t i = 0; i < movieJsons.GetLength(); i += increment) {
                writeRequests.push_back(Aws::DynamoDB::Model::WriteRequest());
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
putItems = movieJsonViewToAttributeMap(
                        movieJsons[i]);
                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(putItems);
                writeRequests.back().SetPutRequest(putRequest);
                if (writeRequests.size() == MAX_SIZE_FOR_BATCH_WRITE) {
                    Aws::DynamoDB::Model::BatchWriteItemRequest request;
                    request.AddRequestItems(MOVIE_TABLE_NAME, writeRequests);
                    const Aws::DynamoDB::Model::BatchWriteItemOutcome &outcome =
dynamoClient.BatchWriteItem(
                            request);
                    if (!outcome.IsSuccess()) {
                        std::cerr << "Unable to batch write movie data: "
                                  << outcome.GetError().GetMessage()
                                  << std::endl;
                        writeRequests.clear();
                        break;
                    }
                    else {
                        std::cout << "Added batch of " << writeRequests.size()
                                  << " movies to the database."
                                  << std::endl;
                    }
                    writeRequests.clear();
                }
            }
        }
    }

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;

    // 5. Get a movie by Key (partition + sort).
    {
        Aws::String titleToGet("King Kong");
        Aws::String answer = askQuestion(Aws::String(
                "Let's move on...Would you like to get info about '" + titleToGet
+
                "'? (y/n) "));
        if (answer == "y") {
            Aws::DynamoDB::Model::GetItemRequest request;
```

```
            request.SetTableName(MOVIE_TABLE_NAME);
            request.AddKey(TITLE_KEY,

Aws::DynamoDB::Model::AttributeValue().SetS(titleToGet));
            request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(1933));

            const Aws::DynamoDB::Model::GetItemOutcome &result =
dynamoClient.GetItem(
                    request);
            if (!result.IsSuccess()) {
                std::cerr << "Error " << result.GetError().GetMessage();
            }
            else {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = result.GetResult().GetItem();
                if (!item.empty()) {
                    std::cout << "\nHere's what I found:" << std::endl;
                    printMovieInfo(item);
                }
                else {
                    std::cout << "\nThe movie was not found in the database."
                              << std::endl;
                }
            }
        }
    }

    // 6. Use Query with a key condition expression to return all movies
    //    released in a given year.
    Aws::String doAgain = "n";
    do {
        Aws::DynamoDB::Model::QueryRequest req;

        req.SetTableName(MOVIE_TABLE_NAME);

        // "year" is a DynamoDB reserved keyword and must be replaced with an
        // expression attribute name.
        req.SetKeyConditionExpression("#dynobase_year = :valueToMatch");
        req.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

        int yearToMatch = askQuestionForIntRange(
                "\nLet's get a list of movies released in"
                " a given year. Enter a year between 1972 and 2018 ",
```

```
                1972, 2018);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
        attributeValues.emplace(":valueToMatch",
                                Aws::DynamoDB::Model::AttributeValue().SetN(
                                        yearToMatch));
        req.SetExpressionAttributeValues(attributeValues);

        const Aws::DynamoDB::Model::QueryOutcome &result =
dynamoClient.Query(req);
        if (result.IsSuccess()) {
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
            if (!items.empty()) {
                std::cout << "\nThere were " << items.size()
                          << " movies in the database from "
                          << yearToMatch << "." << std::endl;
                for (const auto &item: items) {
                    printMovieInfo(item);
                }
                doAgain = "n";
            }
            else {
                std::cout << "\nNo movies from " << yearToMatch
                          << " were found in the database"
                          << std::endl;
                doAgain = askQuestion(Aws::String("Try another year? (y/n) "));
            }
        }
        else {
            std::cerr << "Failed to Query items: " <<
result.GetError().GetMessage()
                      << std::endl;
        }

    } while (doAgain == "y");

    //  7. Use Scan to return movies released within a range of years.
    //     Show how to paginate data using ExclusiveStartKey. (Scan +
FilterExpression)
    {
        int startYear = askQuestionForIntRange("\nNow let's scan a range of years
"
```

```
                                                "for movies in the database. Enter
a start year: ",
                                                1972, 2018);
       int endYear = askQuestionForIntRange("\nEnter an end year: ",
                                            startYear, 2018);
       Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
exclusiveStartKey;
       do {
           Aws::DynamoDB::Model::ScanRequest scanRequest;
           scanRequest.SetTableName(MOVIE_TABLE_NAME);
           scanRequest.SetFilterExpression(
                   "#dynobase_year >= :startYear AND #dynobase_year
<= :endYear");
           scanRequest.SetExpressionAttributeNames({{"#dynobase_year",
YEAR_KEY}});

           Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
           attributeValues.emplace(":startYear",
                                   Aws::DynamoDB::Model::AttributeValue().SetN(
                                           startYear));
           attributeValues.emplace(":endYear",
                                   Aws::DynamoDB::Model::AttributeValue().SetN(
                                           endYear));
           scanRequest.SetExpressionAttributeValues(attributeValues);

           if (!exclusiveStartKey.empty()) {
               scanRequest.SetExclusiveStartKey(exclusiveStartKey);
           }

           const Aws::DynamoDB::Model::ScanOutcome &result = dynamoClient.Scan(
                   scanRequest);
           if (result.IsSuccess()) {
               const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
               if (!items.empty()) {
                   std::stringstream stringStream;
                   stringStream << "\nFound " << items.size() << " movies in one
scan."
                                << " How many would you like to see? ";
                   size_t count = askQuestionForInt(stringStream.str());
                   for (size_t i = 0; i < count && i < items.size(); ++i) {
                       printMovieInfo(items[i]);
                   }
```

```
                }
                else {
                    std::cout << "\nNo movies in the database between " <<
startYear <<
                              " and " << endYear << "." << std::endl;
                }

                exclusiveStartKey = result.GetResult().GetLastEvaluatedKey();
                if (!exclusiveStartKey.empty()) {
                    std::cout << "Not all movies were retrieved. Scanning for
more."
                              << std::endl;
                }
                else {
                    std::cout << "All movies were retrieved with this scan."
                              << std::endl;
                }
            }
            else {
                std::cerr << "Failed to Scan movies: "
                          << result.GetError().GetMessage() << std::endl;
            }
        } while (!exclusiveStartKey.empty());
    }

    // 8. Delete a movie. (DeleteItem)
    {
        std::stringstream stringStream;
        stringStream << "\nWould you like to delete the movie " << title
                     << " from the database? (y/n) ";
        Aws::String answer = askQuestion(stringStream.str());
        if (answer == "y") {
            Aws::DynamoDB::Model::DeleteItemRequest request;
            request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(year));
            request.AddKey(TITLE_KEY,
                           Aws::DynamoDB::Model::AttributeValue().SetS(title));
            request.SetTableName(MOVIE_TABLE_NAME);

            const Aws::DynamoDB::Model::DeleteItemOutcome &result =
dynamoClient.DeleteItem(
                    request);
            if (result.IsSuccess()) {
                std::cout << "\nRemoved \"" << title << "\" from the database."
```

```
                          << std::endl;
            }
            else {
                std::cerr << "Failed to delete the movie: "
                          << result.GetError().GetMessage()
                          << std::endl;
            }
        }
    }

    return true;
}

//! Routine to convert a JsonView object to an attribute map.
/*!
  \sa movieJsonViewToAttributeMap()
  \param jsonView: Json view object.
  \return map: Map that can be used in a DynamoDB request.
 */
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
AwsDoc::DynamoDB::movieJsonViewToAttributeMap(
        const Aws::Utils::Json::JsonView &jsonView) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> result;

    if (jsonView.KeyExists(YEAR_KEY)) {
        result[YEAR_KEY].SetN(jsonView.GetInteger(YEAR_KEY));
    }
    if (jsonView.KeyExists(TITLE_KEY)) {
        result[TITLE_KEY].SetS(jsonView.GetString(TITLE_KEY));
    }
    if (jsonView.KeyExists(INFO_KEY)) {
        Aws::Map<Aws::String, const
 std::shared_ptr<Aws::DynamoDB::Model::AttributeValue>> infoMap;
        Aws::Utils::Json::JsonView infoView = jsonView.GetObject(INFO_KEY);
        if (infoView.KeyExists(RATING_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue
 = std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetN(infoView.GetDouble(RATING_KEY));
            infoMap.emplace(std::make_pair(RATING_KEY, attributeValue));
        }
        if (infoView.KeyExists(PLOT_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue
 = std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetS(infoView.GetString(PLOT_KEY));
```

```
            infoMap.emplace(std::make_pair(PLOT_KEY, attributeValue));
        }

        result[INFO_KEY].SetM(infoMap);
    }

    return result;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
```

```
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." <<
 std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
 dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
```

```
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);
```

```
    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo para criar a tabela e executar ações nela.

```
import (
 "context"
 "fmt"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunMovieScenario is an interactive example that shows you how to use the AWS
 SDK for Go
// to create and use an Amazon DynamoDB table that stores data about movies.
//
//  1. Create a table that can hold movie data.
//  2. Put, get, and update a single movie in the table.
//  3. Write movie data to the table from a sample JSON file.
//  4. Query for movies that were released in a given year.
//  5. Scan for movies that were released in a range of years.
//  6. Delete a movie from the table.
//  7. Delete the table.
//
// This example creates a DynamoDB service client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
```

```
// This package can be found in the ..\..\demotools folder of this repo.
//
// The specified movie sampler is used to get sample data from a URL that is
 loaded
// into the named table.
func RunMovieScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner,
 tableName string,
 movieSampler actions.IMovieSampler) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB getting started demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{TableName: tableName,
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig)}

 exists, err := tableBasics.TableExists(ctx)
 if err != nil {
  panic(err)
 }
 if !exists {
  log.Printf("Creating table %v...\n", tableName)
  _, err = tableBasics.CreateMovieTable(ctx)
  if err != nil {
   panic(err)
  } else {
   log.Printf("Created table %v.\n", tableName)
  }
 } else {
  log.Printf("Table %v already exists.\n", tableName)
 }

 var customMovie actions.Movie
 customMovie.Title = questioner.Ask("Enter a movie title to add to the table:",
  demotools.NotEmpty{})
 customMovie.Year = questioner.AskInt("What year was it released?",
  demotools.NotEmpty{}, demotools.InIntRange{Lower: 1900, Upper: 2030})
 customMovie.Info = map[string]interface{}{}
```

```
customMovie.Info["rating"] = questioner.AskFloat64(
 "Enter a rating between 1 and 10:",
 demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
customMovie.Info["plot"] = questioner.Ask("What's the plot? ",
 demotools.NotEmpty{})
err = tableBasics.AddMovie(ctx, customMovie)
if err == nil {
 log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's update your movie. You previously rated it %v.\n",
customMovie.Info["rating"])
customMovie.Info["rating"] = questioner.AskFloat64(
 "What new rating would you give it?",
 demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
log.Printf("You summarized the plot as '%v'.\n", customMovie.Info["plot"])
customMovie.Info["plot"] = questioner.Ask("What would you say now?",
 demotools.NotEmpty{})
attributes, err := tableBasics.UpdateMovie(ctx, customMovie)
if err == nil {
 log.Printf("Updated %v with new values.\n", customMovie.Title)
 for _, attVal := range attributes {
  for valKey, val := range attVal {
   log.Printf("\t%v: %v\n", valKey, val)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting movie data from %v and adding 250 movies to the table...\n",
 movieSampler.GetURL())
movies := movieSampler.GetSampleMovies()
written, err := tableBasics.AddMovieBatch(ctx, movies, 250)
if err != nil {
 panic(err)
} else {
 log.Printf("Added %v movies to the table.\n", written)
}

show := 10
if show > written {
 show = written
}
```

```
log.Printf("The first %v movies in the table are:", show)
for index, movie := range movies[:show] {
 log.Printf("\t%v. %v\n", index+1, movie.Title)
}
movieIndex := questioner.AskInt(
 "Enter the number of a movie to get info about it: ",
 demotools.InIntRange{Lower: 1, Upper: show},
)
movie, err := tableBasics.GetMovie(ctx, movies[movieIndex-1].Title,
movies[movieIndex-1].Year)
if err == nil {
 log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Println("Let's get a list of movies released in a given year.")
releaseYear := questioner.AskInt("Enter a year between 1972 and 2018: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018},
)
releases, err := tableBasics.Query(ctx, releaseYear)
if err == nil {
 if len(releases) == 0 {
  log.Printf("I couldn't find any movies released in %v!\n", releaseYear)
 } else {
  for _, movie = range releases {
   log.Println(movie)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Println("Now let's scan for movies released in a range of years.")
startYear := questioner.AskInt("Enter a year: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018})
endYear := questioner.AskInt("Enter another year: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018})
releases, err = tableBasics.Scan(ctx, startYear, endYear)
if err == nil {
 if len(releases) == 0 {
  log.Printf("I couldn't find any movies released between %v and %v!\n",
startYear, endYear)
 } else {
  log.Printf("Found %v movies. In this list, the plot is <nil> because "+
   "we used a projection expression when scanning for items to return only "+
```

```
    "the title, year, and rating.\n", len(releases))
   for _, movie = range releases {
    log.Println(movie)
   }
  }
 }
 log.Println(strings.Repeat("-", 88))

 var tables []string
 if questioner.AskBool("Do you want to list all of your tables? (y/n) ", "y") {
  tables, err = tableBasics.ListTables(ctx)
  if err == nil {
   log.Printf("Found %v tables:", len(tables))
   for _, table := range tables {
    log.Printf("\t%v", table)
   }
  }
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Let's remove your movie '%v'.\n", customMovie.Title)
 if questioner.AskBool("Do you want to delete it from the table? (y/n) ", "y") {
  err = tableBasics.DeleteMovie(ctx, customMovie)
 }
 if err == nil {
  log.Printf("Deleted %v.\n", customMovie.Title)
 }

 if questioner.AskBool("Delete the table, too? (y/n)", "y") {
  err = tableBasics.DeleteTable(ctx)
 } else {
  log.Println("Don't forget to delete the table when you're done or you might " +
   "incur charges on your account.")
 }
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}
```

```
// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Crie uma estrutura e métodos que chamem ações do DynamoDB.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
 exists := true
 _, err := basics.DynamoDbClient.DescribeTable(
  ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
 )
 if err != nil {
```

```
  var notFoundEx *types.ResourceNotFoundException
  if errors.As(err, &notFoundEx) {
   log.Printf("Table %v does not exist.\n", basics.TableName)
   err = nil
  } else {
   log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
 basics.TableName, err)
  }
  exists = false
 }
 return exists, err
}



// CreateMovieTable creates a DynamoDB table with a composite primary key defined
 as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
 (*types.TableDescription, error) {
 var tableDesc *types.TableDescription
 table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
  AttributeDefinitions: []types.AttributeDefinition{{
   AttributeName: aws.String("year"),
   AttributeType: types.ScalarAttributeTypeN,
  }, {
   AttributeName: aws.String("title"),
   AttributeType: types.ScalarAttributeTypeS,
  }},
  KeySchema: []types.KeySchemaElement{{
   AttributeName: aws.String("year"),
   KeyType:       types.KeyTypeHash,
  }, {
   AttributeName: aws.String("title"),
   KeyType:       types.KeyTypeRange,
  }},
  TableName:   aws.String(basics.TableName),
  BillingMode: types.BillingModePayPerRequest,
 })
 if err != nil {
  log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
 } else {
```

```
  waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
  err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
   TableName: aws.String(basics.TableName)}, 5*time.Minute)
  if err != nil {
   log.Printf("Wait for table exists failed. Here's why: %v\n", err)
  }
  tableDesc = table.TableDescription
  log.Printf("Ccreating table test")
 }
 return tableDesc, err
}



// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
 var tableNames []string
 var output *dynamodb.ListTablesOutput
 var err error
 tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
 &dynamodb.ListTablesInput{})
 for tablePaginator.HasMorePages() {
  output, err = tablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't list tables. Here's why: %v\n", err)
   break
  } else {
   tableNames = append(tableNames, output.TableNames...)
  }
 }
 return tableNames, err
}



// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
 item, err := attributevalue.MarshalMap(movie)
 if err != nil {
  panic(err)
 }
 _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
  TableName: aws.String(basics.TableName), Item: item,
 })
```

```
 if err != nil {
  log.Printf("Couldn't add item to table. Here's why: %v\n", err)
 }
 return err
}



// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the
 update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
 (map[string]map[string]interface{}, error) {
 var err error
 var response *dynamodb.UpdateItemOutput
 var attributeMap map[string]map[string]interface{}
 update := expression.Set(expression.Name("info.rating"),
 expression.Value(movie.Info["rating"]))
 update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
 expr, err := expression.NewBuilder().WithUpdate(update).Build()
 if err != nil {
  log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
 } else {
  response, err = basics.DynamoDbClient.UpdateItem(ctx,
 &dynamodb.UpdateItemInput{
   TableName:                aws.String(basics.TableName),
   Key:                      movie.GetKey(),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   UpdateExpression:         expr.Update(),
   ReturnValues:             types.ReturnValueUpdatedNew,
  })
  if err != nil {
   log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
  } else {
   err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
   if err != nil {
    log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
   }
  }
 }
 return attributeMap, err
}
```

```
// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
 maxMovies int) (int, error) {
 var err error
 var item map[string]types.AttributeValue
 written := 0
 batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
 start := 0
 end := start + batchSize
 for start < maxMovies && start < len(movies) {
  var writeReqs []types.WriteRequest
  if end > len(movies) {
   end = len(movies)
  }
  for _, movie := range movies[start:end] {
   item, err = attributevalue.MarshalMap(movie)
   if err != nil {
    log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
 movie.Title, err)
   } else {
    writeReqs = append(
     writeReqs,
     types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
   }
  }
  _, err = basics.DynamoDbClient.BatchWriteItem(ctx,
 &dynamodb.BatchWriteItemInput{
   RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
  if err != nil {
   log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
 basics.TableName, err)
  } else {
   written += len(writeReqs)
  }
  start = end
  end += batchSize
 }
```

```
 return written, err
}



// GetMovie gets movie data from the DynamoDB table by using the primary
 composite key
// made of title and year.
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 movie := Movie{Title: title, Year: year}
 response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key: movie.GetKey(), TableName: aws.String(basics.TableName),
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Item, &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}



// Query gets all movies in the DynamoDB table that were released in the
 specified year.
// The function uses the `expression` package to build the key condition
 expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
 error) {
 var err error
 var response *dynamodb.QueryOutput
 var movies []Movie
 keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
 expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
 if err != nil {
  log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
 } else {
  queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
 &dynamodb.QueryInput{
```

```
    TableName:                aws.String(basics.TableName),
    ExpressionAttributeNames:  expr.Names(),
    ExpressionAttributeValues: expr.Values(),
    KeyConditionExpression:    expr.KeyCondition(),
   })
   for queryPaginator.HasMorePages() {
    response, err = queryPaginator.NextPage(ctx)
    if err != nil {
     log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
  releaseYear, err)
     break
    } else {
     var moviePage []Movie
     err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
     if err != nil {
      log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
      break
     } else {
      movies = append(movies, moviePage...)
     }
    }
   }
  }
  return movies, err
}



// Scan gets all movies in the DynamoDB table that were released in a range of
 years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
 ([]Movie, error) {
 var movies []Movie
 var err error
 var response *dynamodb.ScanOutput
 filtEx := expression.Name("year").Between(expression.Value(startYear),
 expression.Value(endYear))
 projEx := expression.NamesList(
  expression.Name("year"), expression.Name("title"),
 expression.Name("info.rating"))
```

```
 expr, err :=
 expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
 if err != nil {
  log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
 } else {
  scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
 &dynamodb.ScanInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   FilterExpression:          expr.Filter(),
   ProjectionExpression:      expr.Projection(),
  })
  for scanPaginator.HasMorePages() {
   response, err = scanPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't scan for movies released between %v and %v. Here's why:
 %v\n",
     startYear, endYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}



// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
 _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
  TableName: aws.String(basics.TableName), Key: movie.GetKey(),
 })
 if err != nil {
```

```
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}



// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
 _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
  TableName: aws.String(basics.TableName)})
 if err != nil {
  log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma tabela do DynamoDB.

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE)
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);
```

```
        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
```

```
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }
```

Obtenha um item de uma tabela.

```
    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
```

```
            Map<String, AttributeValue> returnedItem =
 ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Exemplo completo.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the
 * Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
```

```
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */

public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        String tableName = "Movies";
        String fileName = "../../../resources/sample_files/movies.json";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "1. Creating an Amazon DynamoDB table named Movies with a key named
 year and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB
 table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();
```

```
        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Query the table.
    public static void queryTable(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();
```

```
            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            QueryConditional queryConditional = QueryConditional
                .keyEqualTo(Key.builder()
                    .partitionValue(2013)
                    .build());

            // Get items in the table and write out the ID value.
            Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
            String result = "";

            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The title of the movie is " +
rec.getTitle());
                System.out.println("The movie information  is " + rec.getInfo());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Scan the table.
    public static void scanMovies(DynamoDbClient ddb, String tableName) {
        System.out.println("******* Scanning all movies.\n");
        try {
            DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            Iterator<Movies> results = custTable.scan().items().iterator();
            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The movie title is " + rec.getTitle());
                System.out.println("The movie year is " + rec.getYear());
            }

        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
 fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
 TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName) {
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
```

```
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s("{\"directors\":[\"Merian C. Cooper
\",\"Ernest B. Schoedsack\"]")
                .build())
            .action(AttributeAction.PUT)
            .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(itemKey)
            .attributeUpdates(updatedValues)
            .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Item was updated!");
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
 {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
```

```
    }

    public static void putRecord(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

            // Populate the Table.
            Movies record = new Movies();
            record.setYear(2020);
            record.setTitle("My Movie2");
            record.setInfo("no info");
            table.putItem(record);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Added a new movie to the table.");
    }

    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
```

```
            Map<String, AttributeValue> returnedItem =
 ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { readFileSync } from "node:fs";
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";

/**
 * This module is a convenience library. It abstracts Amazon DynamoDB's data type
 * descriptors (such as S, N, B, and BOOL) by marshalling JavaScript objects into
 * AttributeValue shapes.
 */
import {
  BatchWriteCommand,
  DeleteCommand,
  DynamoDBDocumentClient,
  GetCommand,
  PutCommand,
  UpdateCommand,
  paginateQuery,
  paginateScan,
} from "@aws-sdk/lib-dynamodb";

// These modules are local to our GitHub repository. We recommend cloning
// the project from GitHub if you want to run this example.
// For more information, see https://github.com/awsdocs/aws-doc-sdk-examples.
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { chunkArray } from "@aws-doc-sdk-examples/lib/utils/util-array.js";
```

```
const dirname = dirnameFromMetaUrl(import.meta.url);
const tableName = getUniqueName("Movies");
const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);

export const main = async () => {
  /**
   * Create a table.
   */

  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "year",
        // 'N' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "N",
      },
      { AttributeName: "title", AttributeType: "S" },
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [
      // The way your data is accessed determines how you structure your keys.
      // The movies table will be queried for movies by year. It makes sense
      // to make year our partition (HASH) key.
      { AttributeName: "year", KeyType: "HASH" },
      { AttributeName: "title", KeyType: "RANGE" },
    ],
  });
```

```
  log("Creating a table.");
  const createTableResponse = await client.send(createTableCommand);
  log(`Table created: ${JSON.stringify(createTableResponse.TableDescription)}`);

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Add a movie to the table.
   */

  log("Adding a single movie to the table.");
  // PutCommand is the first example usage of 'lib-dynamodb'.
  const putCommand = new PutCommand({
    TableName: tableName,
    Item: {
      // In 'client-dynamodb', the AttributeValue would be required (`year: { N:
1981 }`)
      // 'lib-dynamodb' simplifies the usage ( `year: 1981` )
      year: 1981,
      // The preceding KeySchema defines 'title' as our sort (RANGE) key, so
'title'
      // is required.
      title: "The Evil Dead",
      // Every other attribute is optional.
      info: {
        genres: ["Horror"],
      },
    },
  });
  await docClient.send(putCommand);
  log("The movie was added.");

  /**
   * Get a movie from the table.
   */

  log("Getting a single movie from the table.");
  const getCommand = new GetCommand({
    TableName: tableName,
    // Requires the complete primary key. For the movies table, the primary key
```

```
    // is only the id (partition key).
    Key: {
      year: 1981,
      title: "The Evil Dead",
    },
    // Set this to make sure that recent writes are reflected.
    // For more information, see https://docs.aws.amazon.com/amazondynamodb/
latest/developerguide/HowItWorks.ReadConsistency.html.
    ConsistentRead: true,
  });
  const getResponse = await docClient.send(getCommand);
  log(`Got the movie: ${JSON.stringify(getResponse.Item)}`);

  /**
   * Update a movie in the table.
   */

  log("Updating a single movie in the table.");
  const updateCommand = new UpdateCommand({
    TableName: tableName,
    Key: { year: 1981, title: "The Evil Dead" },
    // This update expression appends "Comedy" to the list of genres.
    // For more information on update expressions, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.UpdateExpressions.html
    UpdateExpression: "set #i.#g = list_append(#i.#g, :vals)",
    ExpressionAttributeNames: { "#i": "info", "#g": "genres" },
    ExpressionAttributeValues: {
      ":vals": ["Comedy"],
    },
    ReturnValues: "ALL_NEW",
  });
  const updateResponse = await docClient.send(updateCommand);
  log(`Movie updated: ${JSON.stringify(updateResponse.Attributes)}`);

  /**
   * Delete a movie from the table.
   */

  log("Deleting a single movie from the table.");
  const deleteCommand = new DeleteCommand({
    TableName: tableName,
    Key: { year: 1981, title: "The Evil Dead" },
  });
```

```
  await docClient.send(deleteCommand);
  log("Movie deleted.");

  /**
   * Upload a batch of movies.
   */

  log("Adding movies from local JSON file.");
  const file = readFileSync(
    `${dirname}../../../../resources/sample_files/movies.json`,
  );
  const movies = JSON.parse(file.toString());
  // chunkArray is a local convenience function. It takes an array and returns
  // a generator function. The generator function yields every N items.
  const movieChunks = chunkArray(movies, 25);
  // For every chunk of 25 movies, make one BatchWrite request.
  for (const chunk of movieChunks) {
    const putRequests = chunk.map((movie) => ({
      PutRequest: {
        Item: movie,
      },
    }));

    const command = new BatchWriteCommand({
      RequestItems: {
        [tableName]: putRequests,
      },
    });

    await docClient.send(command);
  }
  log("Movies added.");

  /**
   * Query for movies by year.
   */

  log("Querying for all movies from 1981.");
  const paginatedQuery = paginateQuery(
    { client: docClient },
    {
      TableName: tableName,
      //For more information about query expressions, see
```

```
      // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Query.html#Query.KeyConditionExpressions
      KeyConditionExpression: "#y = :y",
      // 'year' is a reserved word in DynamoDB. Indicate that it's an attribute
      // name by using an expression attribute name.
      ExpressionAttributeNames: { "#y": "year" },
      ExpressionAttributeValues: { ":y": 1981 },
      ConsistentRead: true,
    },
  );
  /**
   * @type { Record<string, any>[] };
   */
  const movies1981 = [];
  for await (const page of paginatedQuery) {
    movies1981.push(...page.Items);
  }
  log(`Movies: ${movies1981.map((m) => m.title).join(", ")}`);

  /**
   * Scan the table for movies between 1980 and 1990.
   */

  log("Scan for movies released between 1980 and 1990");
  // A 'Scan' operation always reads every item in the table. If your design
 requires
  // the use of 'Scan', consider indexing your table or changing your design.
  // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-query-
scan.html
  const paginatedScan = paginateScan(
    { client: docClient },
    {
      TableName: tableName,
      // Scan uses a filter expression instead of a key condition expression.
 Scan will
      // read the entire table and then apply the filter.
      FilterExpression: "#y between :y1 and :y2",
      ExpressionAttributeNames: { "#y": "year" },
      ExpressionAttributeValues: { ":y1": 1980, ":y2": 1990 },
      ConsistentRead: true,
    },
  );
  /**
   * @type { Record<string, any>[] };
```

```
   */
  const movies1980to1990 = [];
  for await (const page of paginatedScan) {
    movies1980to1990.push(...page.Items);
  }
  log(
    `Movies: ${movies1980to1990
      .map((m) => `${m.title} (${m.year})`)
      .join(", ")}`,
  );

  /**
   * Delete the table.
   */

  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  log(`Deleting table ${tableName}.`);
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma tabela do DynamoDB.

```
suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
```

```
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}
```

```
suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

Obtenha um item de uma tabela.

```
suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }
```

```
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

Exemplo completo.

```
suspend fun main() {
    val tableName = "Movies"
    val fileName = "../../../resources/sample_files/movies.json"
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id
 and a sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deletIssuesTable(tableName)
}

suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
```

```
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
```

```
    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
```

```
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")
```

```
    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem

- PutItem
- Query
- Scan
- UpdateItem

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Como esse exemplo usa arquivos de suporte, não deixe de ler as orientações no README.md arquivo de exemplos do PHP.

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!
\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDBService();
```

```
        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
$tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
|| $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
```

```
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];
        $attributes = ["rating" =>
            [
                'AttributeName' => 'rating',
                'AttributeType' => 'N',
                'Value' => $rating,
            ],
            'plot' => [
                'AttributeName' => 'plot',
                'AttributeType' => 'S',
                'Value' => $plot,
            ]
        ];
        $service->updateItemAttributesByKey($tableName, $key, $attributes);
        echo "Movie added and updated.";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);


        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
 {$movie['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Item']['title']['S']}?
\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
```

```
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

        $movie = $service->getItemByKey($tableName, $key);
        echo "Ok, you have rated {$movie['Item']['title']['S']} as a
{$movie['Item']['rating']['N']}\n";

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something
lighter, in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were
born:\n";
        $oops = "Oops! There were no movies released in that year (that we know
of).\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
```

```
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que encapsule uma tabela do DynamoDB.

```
from decimal import Decimal
from io import BytesIO
import json
import logging
import os
from pprint import pprint
import requests
from zipfile import ZipFile
import boto3
from boto3.dynamodb.conditions import Key
from botocore.exceptions import ClientError
from question import Question

logger = logging.getLogger(__name__)

class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
```

```
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            table.load()
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                exists = False
            else:
                logger.error(
                    "Couldn't check for existence of %s. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            self.table = table
        return exists
```

```
    def create_table(self, table_name):
        """
        Creates an Amazon DynamoDB table that can be used to store movie data.
        The table uses the release year of the movie as the partition key and the
        title as the sort key.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
        try:
            self.table = self.dyn_resource.create_table(
                TableName=table_name,
                KeySchema=[
                    {"AttributeName": "year", "KeyType": "HASH"},  # Partition
key
                    {"AttributeName": "title", "KeyType": "RANGE"},  # Sort key
                ],
                AttributeDefinitions=[
                    {"AttributeName": "year", "AttributeType": "N"},
                    {"AttributeName": "title", "AttributeType": "S"},
                ],
                BillingMode='PAY_PER_REQUEST',
            )
            self.table.wait_until_exists()
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.table


    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
```

```
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tables


    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts
sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at
least
                       the keys required by the schema that was specified when
the
                       table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    "year": year,
                    "title": title,
                    "info": {"plot": plot, "rating": Decimal(str(rating))},
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title,
                self.table.name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Item"]


    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to the give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={":r": Decimal(str(rating)), ":p":
plot},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]


    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
```

```
        :return: The list of movies that were released in the specified year.
        """
        try:
            response =
self.table.query(KeyConditionExpression=Key("year").eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
                year,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Items"]


    def scan_movies(self, year_range):
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            "FilterExpression": Key("year").between(
                year_range["first"], year_range["second"]
            ),
            "ProjectionExpression": "#yr, title, info.rating",
            "ExpressionAttributeNames": {"#yr": "year"},
        }
        try:
            done = False
            start_key = None
            while not done:
                if start_key:
                    scan_kwargs["ExclusiveStartKey"] = start_key
                response = self.table.scan(**scan_kwargs)
                movies.extend(response.get("Items", []))
                start_key = response.get("LastEvaluatedKey", None)
                done = start_key is None
        except ClientError as err:
```

```
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

        return movies


    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
def get_sample_movie_data(movie_file_name):
    """
    Gets sample movie data, either from a local file or by first downloading it
 from
    the Amazon DynamoDB developer guide.

    :param movie_file_name: The local file name where the movie data is stored in
 JSON format.
    :return: The movie data as a dict.
    """
    if not os.path.isfile(movie_file_name):
        print(f"Downloading {movie_file_name}...")
        movie_content = requests.get(
            "https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
samples/moviedata.zip"
        )
        movie_zip = ZipFile(BytesIO(movie_content.content))
        movie_zip.extractall()

    try:
        with open(movie_file_name) as movie_file:
            movie_data = json.load(movie_file, parse_float=Decimal)
    except FileNotFoundError:
        print(
            f"File {movie_file_name} not found. You must first download the file
 to "
            "run this demo. See the README for instructions."
        )
        raise
    else:
        # The sample file lists over 4000 movies, return only the first 250.
        return movie_data[:250]
```

Execute um cenário interativo para criar a tabela e executar ações nela.

```
def run_scenario(table_name, movie_file_name, dyn_resource):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB getting started demo.")
    print("-" * 88)

    movies = Movies(dyn_resource)
    movies_exists = movies.exists(table_name)
    if not movies_exists:
        print(f"\nCreating table {table_name}...")
        movies.create_table(table_name)
        print(f"\nCreated table {movies.table.name}.")

    my_movie = Question.ask_questions(
        [
            Question(
                "title", "Enter the title of a movie you want to add to the
 table: "
            ),
            Question("year", "What year was it released? ", Question.is_int),
            Question(
                "rating",
                "On a scale of 1 - 10, how do you rate it? ",
                Question.is_float,
                Question.in_range(1, 10),
            ),
            Question("plot", "Summarize the plot for me: "),
        ]
    )
    movies.add_movie(**my_movie)
    print(f"\nAdded '{my_movie['title']}' to '{movies.table.name}'.")
    print("-" * 88)

    movie_update = Question.ask_questions(
        [
            Question(
                "rating",
                f"\nLet's update your movie.\nYou rated it {my_movie['rating']},
 what new "
                f"rating would you give it? ",
                Question.is_float,
                Question.in_range(1, 10),
```

```
            ),
            Question(
                "plot",
                f"You summarized the plot as '{my_movie['plot']}'.\nWhat would
you say now? ",
            ),
        ]
    )
    my_movie.update(movie_update)
    updated = movies.update_movie(**my_movie)
    print(f"\nUpdated '{my_movie['title']}' with new attributes:")
    pprint(updated)
    print("-" * 88)

    if not movies_exists:
        movie_data = get_sample_movie_data(movie_file_name)
        print(f"\nReading data from '{movie_file_name}' into your table.")
        movies.write_batch(movie_data)
        print(f"\nWrote {len(movie_data)} movies into {movies.table.name}.")
    print("-" * 88)

    title = "The Lord of the Rings: The Fellowship of the Ring"
    if Question.ask_question(
        f"Let's move on...do you want to get info about '{title}'? (y/n) ",
        Question.is_yesno,
    ):
        movie = movies.get_movie(title, 2001)
        print("\nHere's what I found:")
        pprint(movie)
    print("-" * 88)

    ask_for_year = True
    while ask_for_year:
        release_year = Question.ask_question(
            f"\nLet's get a list of movies released in a given year. Enter a year
between "
            f"1972 and 2018: ",
            Question.is_int,
            Question.in_range(1972, 2018),
        )
        releases = movies.query_movies(release_year)
        if releases:
            print(f"There were {len(releases)} movies released in
{release_year}:")
```

```
            for release in releases:
                print(f"\t{release['title']}")
            ask_for_year = False
        else:
            print(f"I don't know about any movies released in {release_year}!")
            ask_for_year = Question.ask_question(
                "Try another year? (y/n) ", Question.is_yesno
            )
    print("-" * 88)

    years = Question.ask_questions(
        [
            Question(
                "first",
                f"\nNow let's scan for movies released in a range of years. Enter
a year: ",
                Question.is_int,
                Question.in_range(1972, 2018),
            ),
            Question(
                "second",
                "Now enter another year: ",
                Question.is_int,
                Question.in_range(1972, 2018),
            ),
        ]
    )
    releases = movies.scan_movies(years)
    if releases:
        count = Question.ask_question(
            f"\nFound {len(releases)} movies. How many do you want to see? ",
            Question.is_int,
            Question.in_range(1, len(releases)),
        )
        print(f"\nHere are your {count} movies:\n")
        pprint(releases[:count])
    else:
        print(
            f"I don't know about any movies released between {years['first']} "
            f"and {years['second']}."
        )
    print("-" * 88)

    if Question.ask_question(
```

```
        f"\nLet's remove your movie from the table. Do you want to remove "
        f"'{my_movie['title']}'? (y/n)",
        Question.is_yesno,
    ):
        movies.delete_movie(my_movie["title"], my_movie["year"])
        print(f"\nRemoved '{my_movie['title']}' from the table.")
    print("-" * 88)

    if Question.ask_question(f"\nDelete the table? (y/n) ", Question.is_yesno):
        movies.delete_table()
        print(f"Deleted {table_name}.")
    else:
        print(
            "Don't forget to delete the table when you're done or you might incur
 "
            "charges on your account."
        )

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        run_scenario(
            "doc-example-table-movies", "moviedata.json",
 boto3.resource("dynamodb")
        )
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

Esse cenário usa a classe auxiliar a seguir para fazer perguntas em um prompt de comando.

```
class Question:
    """
    A helper class to ask questions at a command prompt and validate and convert
    the answers.
    """

    def __init__(self, key, question, *validators):
        """
        :param key: The key that is used for storing the answer in a dict, when
```

```
                    multiple questions are asked in a set.
        :param question: The question to ask.
        :param validators: The answer is passed through the list of validators
until
                           one fails or they all pass. Validators may also
convert the
                           answer to another form, such as from a str to an int.
        """
        self.key = key
        self.question = question
        self.validators = Question.non_empty, *validators

    @staticmethod
    def ask_questions(questions):
        """
        Asks a set of questions and stores the answers in a dict.

        :param questions: The list of questions to ask.
        :return: A dict of answers.
        """
        answers = {}
        for question in questions:
            answers[question.key] = Question.ask_question(
                question.question, *question.validators
            )
        return answers

    @staticmethod
    def ask_question(question, *validators):
        """
        Asks a single question and validates it against a list of validators.
        When an answer fails validation, the complaint is printed and the
question
        is asked again.

        :param question: The question to ask.
        :param validators: The list of validators that the answer must pass.
        :return: The answer, converted to its final form by the validators.
        """
        answer = None
        while answer is None:
            answer = input(question)
            for validator in validators:
                answer, complaint = validator(answer)
```

```
                if answer is None:
                    print(complaint)
                    break
        return answer

    @staticmethod
    def non_empty(answer):
        """
        Validates that the answer is not empty.
        :return: The non-empty answer, or None.
        """
        return answer if answer != "" else None, "I need an answer. Please?"

    @staticmethod
    def is_yesno(answer):
        """
        Validates a yes/no answer.
        :return: True when the answer is 'y'; otherwise, False.
        """
        return answer.lower() == "y", ""

    @staticmethod
    def is_int(answer):
        """
        Validates that the answer can be converted to an int.
        :return: The int answer; otherwise, None.
        """
        try:
            int_answer = int(answer)
        except ValueError:
            int_answer = None
        return int_answer, f"{answer} must be a valid integer."

    @staticmethod
    def is_letter(answer):
        """
        Validates that the answer is a letter.
        :return The letter answer, converted to uppercase; otherwise, None.
        """
        return (
            answer.upper() if answer.isalpha() else None,
            f"{answer} must be a single letter.",
        )
```

```
    @staticmethod
    def is_float(answer):
        """
        Validate that the answer can be converted to a float.
        :return The float answer; otherwise, None.
        """
        try:
            float_answer = float(answer)
        except ValueError:
            float_answer = None
        return float_answer, f"{answer} must be a valid float."

    @staticmethod
    def in_range(lower, upper):
        """
        Validate that the answer is within a range. The answer must be of a type
that can
        be compared to the lower and upper bounds.
        :return: The answer, if it is within the range; otherwise, None.
        """

        def _validate(answer):
            return (
                answer if lower <= answer <= upper else None,
                f"{answer} must be between {lower} and {upper}.",
            )

        return _validate
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem

- PutItem
- Query
- Scan
- UpdateItem

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que encapsule uma tabela do DynamoDB.

```
# Creates an Amazon DynamoDB table that can be used to store movie data.
# The table uses the release year of the movie as the partition key and the
# title as the sort key.
#
# @param table_name [String] The name of the table to create.
# @return [Aws::DynamoDB::Table] The newly created table.
def create_table(table_name)
  @table = @dynamo_resource.create_table(
    table_name: table_name,
    key_schema: [
      { attribute_name: 'year', key_type: 'HASH' }, # Partition key
      { attribute_name: 'title', key_type: 'RANGE' } # Sort key
    ],
    attribute_definitions: [
      { attribute_name: 'year', attribute_type: 'N' },
      { attribute_name: 'title', attribute_type: 'S' }
    ],
    billing_mode: 'PAY_PER_REQUEST'
  )
  @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
  @table
rescue Aws::DynamoDB::Errors::ServiceError => e
  @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
```

```
    raise
  end
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
  # Gets sample movie data, either from a local file or by first downloading it
 from
  # the Amazon DynamoDB Developer Guide.
  #
  # @param movie_file_name [String] The local file name where the movie data is
 stored in JSON format.
  # @return [Hash] The movie data as a Hash.
  def fetch_movie_data(movie_file_name)
    if !File.file?(movie_file_name)
      @logger.debug("Downloading #{movie_file_name}...")
      movie_content = URI.open(
        'https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
samples/moviedata.zip'
      )
      movie_json = ''
      Zip::File.open_buffer(movie_content) do |zip|
        zip.each do |entry|
          movie_json = entry.get_input_stream.read
        end
      end
    else
      movie_json = File.read(movie_file_name)
    end
    movie_data = JSON.parse(movie_json)
    # The sample file lists over 4000 movies. This returns only the first 250.
    movie_data.slice(0, 250)
  rescue StandardError => e
    puts("Failure downloading movie data:\n#{e}")
    raise
  end
```

Execute um cenário interativo para criar a tabela e executar ações nela.

```
  table_name = "doc-example-table-movies-#{rand(10**4)}"
  scaffold = Scaffold.new(table_name)
  dynamodb_wrapper = DynamoDBBasics.new(table_name)
```

```
 new_step(1, 'Create a new DynamoDB table if none already exists.')
 unless scaffold.exists?(table_name)
   puts("\nNo such table: #{table_name}. Creating it...")
   scaffold.create_table(table_name)
   print "Done!\n".green
 end

 new_step(2, 'Add a new record to the DynamoDB table.')
 my_movie = {}
 my_movie[:title] = CLI::UI::Prompt.ask('Enter the title of a movie to add to
the table. E.g. The Matrix')
 my_movie[:year] = CLI::UI::Prompt.ask('What year was it released? E.g.
1989').to_i
 my_movie[:rating] = CLI::UI::Prompt.ask('On a scale of 1 - 10, how do you rate
it? E.g. 7').to_i
 my_movie[:plot] = CLI::UI::Prompt.ask('Enter a brief summary of the plot. E.g.
A man awakens to a new reality.')
 dynamodb_wrapper.add_item(my_movie)
 puts("\nNew record added:")
 puts JSON.pretty_generate(my_movie).green
 print "Done!\n".green

 new_step(3, 'Update a record in the DynamoDB table.')
 my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with
a new rating, e.g. 3:").to_i
 response = dynamodb_wrapper.update_item(my_movie)
 puts("Updated '#{my_movie[:title]}' with new attributes:")
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(4, 'Get a record from the DynamoDB table.')
 puts("Searching for #{my_movie[:title]} (#{my_movie[:year]})...")
 response = dynamodb_wrapper.get_item(my_movie[:title], my_movie[:year])
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(5, 'Write a batch of items into the DynamoDB table.')
 download_file = 'moviedata.json'
 puts("Downloading movie database to #{download_file}...")
 movie_data = scaffold.fetch_movie_data(download_file)
 puts("Writing movie data from #{download_file} into your table...")
 scaffold.write_batch(movie_data)
 puts("Records added: #{movie_data.length}.")
```

```
 print "Done!\n".green

 new_step(5, 'Query for a batch of items by key.')
 loop do
   release_year = CLI::UI::Prompt.ask('Enter a year between 1972 and 2018, e.g.
1999:').to_i
   results = dynamodb_wrapper.query_items(release_year)
   if results.any?
     puts("There were #{results.length} movies released in #{release_year}:")
     results.each do |movie|
       print "\t #{movie['title']}".green
     end
     break
   else
     continue = CLI::UI::Prompt.ask("Found no movies released in
#{release_year}! Try another year? (y/n)")
     break unless continue.eql?('y')
   end
 end
 print "\nDone!\n".green

 new_step(6, 'Scan for a batch of items using a filter expression.')
 years = {}
 years[:start] = CLI::UI::Prompt.ask('Enter a starting year between 1972 and
2018:')
 years[:end] = CLI::UI::Prompt.ask('Enter an ending year between 1972 and
2018:')
 releases = dynamodb_wrapper.scan_items(years)
 if !releases.empty?
   puts("Found #{releases.length} movies.")
   count = Question.ask(
     'How many do you want to see? ', method(:is_int), in_range(1,
releases.length)
   )
   puts("Here are your #{count} movies:")
   releases.take(count).each do |release|
     puts("\t#{release['title']}")
   end
 else
   puts("I don't know about any movies released between #{years[:start]} "\
        "and #{years[:end]}.")
 end
 print "\nDone!\n".green
```

```
  new_step(7, 'Delete an item from the DynamoDB table.')
  answer = CLI::UI::Prompt.ask("Do you want to remove '#{my_movie[:title]}'? (y/
n) ")
  if answer.eql?('y')
    dynamodb_wrapper.delete_item(my_movie[:title], my_movie[:year])
    puts("Removed '#{my_movie[:title]}' from the table.")
    print "\nDone!\n".green
  end

  new_step(8, 'Delete the DynamoDB table.')
  answer = CLI::UI::Prompt.ask('Delete the table? (y/n)')
  if answer.eql?('y')
    scaffold.delete_table
    puts("Deleted #{table_name}.")
  else
    puts("Don't forget to delete the table when you're done!")
  end
  print "\nThanks for watching!\n".green
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo.')
rescue Errno::ENOENT
  true
end
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create an Amazon Dynamo DB table.

    TRY.
        DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
        DATA(lo_dyn) = /aws1/cl_dyn_factory=>create( lo_session ).
        DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                              iv_keytype = 'HASH' ) )
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                              iv_keytype = 'RANGE' ) ) ).
        DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                           iv_attributetype = 'N' ) )
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                           iv_attributetype = 'S' ) ) ).

        " Adjust read/write capacities as desired.
        DATA(lo_dynprovthroughput)  = NEW /aws1/cl_dynprovthroughput(
          iv_readcapacityunits = 5
          iv_writecapacityunits = 5 ).
        DATA(oo_result) = lo_dyn->createtable(
          it_keyschema = lt_keyschema
          iv_tablename = iv_table_name
          it_attributedefinitions = lt_attributedefinitions
          io_provisionedthroughput = lo_dynprovthroughput ).
        " Table creation can take some time. Wait till table exists before
 returning.
        lo_dyn->get_waiter( )->tableexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
```

```
        " It throws exception if the table already exists.
      CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
 { lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

    " Describe table
    TRY.
        DATA(lo_table) = lo_dyn->describetable( iv_tablename = iv_table_name ).
        DATA(lv_tablename) = lo_table->get_table( )->ask_tablename( ).
        MESSAGE 'The table name is ' && lv_tablename TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table does not exist' TYPE 'E'.
    ENDTRY.

    " Put items into the table.
    TRY.
        DATA(lo_resp_putitem) = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Jaws' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1975' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.5' }| ) ) )
          ) ).
        lo_resp_putitem = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s = 'Star
 Wars' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1978' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
```

```
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '8.1' }| ) ) )
          ) ).
        lo_resp_putitem = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Speed' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1994' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.9' }| ) ) )
          ) ).
        " TYPE REF TO /AWSEX/CL_AWS1_dyn_PUT_ITEM_OUTPUT
        MESSAGE '3 rows inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Get item from table.
    TRY.
        DATA(lo_resp_getitem) = lo_dyn->getitem(
          iv_tablename                = iv_table_name
          it_key                      = VALUE /aws1/cl_dynattributevalue=>tt_key(
           ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
             key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Jaws' ) ) )
           ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
             key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n =
 '1975' ) ) )
          ) ).
        DATA(lt_attr) = lo_resp_getitem->get_item( ).
        DATA(lo_title) = lt_attr[ key = 'title' ]-value.
        DATA(lo_year) = lt_attr[ key = 'year' ]-value.
        DATA(lo_rating) = lt_attr[ key = 'year' ]-value.
```

```
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Query item from table.
    TRY.
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
              ( NEW /aws1/cl_dynattributevalue( iv_n = '1975' ) ) ).
        DATA(lt_keyconditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
          ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
          key = 'year'
          value = NEW /aws1/cl_dyncondition(
            it_attributevaluelist = lt_attributelist
            iv_comparisonoperator = |EQ|
          ) ) ) ).
        DATA(lo_query_result) = lo_dyn->query(
          iv_tablename = iv_table_name
          it_keyconditions = lt_keyconditions ).
        DATA(lt_items) = lo_query_result->get_items( ).
        READ TABLE lo_query_result->get_items( ) INTO DATA(lt_item) INDEX 1.
        lo_title = lt_item[ key = 'title' ]-value.
        lo_year = lt_item[ key = 'year' ]-value.
        lo_rating = lt_item[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Scan items from table.
    TRY.
        DATA(lo_scan_result) = lo_dyn->scan( iv_tablename = iv_table_name ).
        lt_items = lo_scan_result->get_items( ).
        " Read the first item and display the attributes.
        READ TABLE lo_query_result->get_items( ) INTO lt_item INDEX 1.
        lo_title = lt_item[ key = 'title' ]-value.
        lo_year = lt_item[ key = 'year' ]-value.
        lo_rating = lt_item[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
```

```
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Update items from table.
    TRY.
        DATA(lt_attributeupdates) = VALUE /aws1/
cl_dynattrvalueupdate=>tt_attributeupdates(
          ( VALUE /aws1/cl_dynattrvalueupdate=>ts_attributeupdates_maprow(
          key = 'rating' value = NEW /aws1/cl_dynattrvalueupdate(
            io_value  = NEW /aws1/cl_dynattributevalue( iv_n = '7.6' )
            iv_action = |PUT| ) ) ) ).
        DATA(lt_key) = VALUE /aws1/cl_dynattributevalue=>tt_key(
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n =
 '1975' ) ) )
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 '1980' ) ) ) ).
        DATA(lo_resp) = lo_dyn->updateitem(
          iv_tablename        = iv_table_name
          it_key              = lt_key
          it_attributeupdates = lt_attributeupdates ).
        MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Delete table.
    TRY.
        lo_dyn->deletetable( iv_tablename = iv_table_name ).
        lo_dyn->get_waiter( )->tablenotexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table deleted.' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
```

```
      CATCH /aws1/cx_dynresourceinuseex.
        MESSAGE 'The table cannot be deleted as it is in use' TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Uma classe Swift que gerencia chamadas do DynamoDB para o SDK para Swift.

```
import AWSDynamoDB
import Foundation

/// An enumeration of error codes representing issues that can arise when using
/// the `MovieTable` class.
enum MoviesError: Error {
    /// The specified table wasn't found or couldn't be created.
```

```
    case TableNotFound
    /// The specified item wasn't found or couldn't be created.
    case ItemNotFound
    /// The Amazon DynamoDB client is not properly initialized.
    case UninitializedClient
    /// The table status reported by Amazon DynamoDB is not recognized.
    case StatusUnknown
    /// One or more specified attribute values are invalid or missing.
    case InvalidAttributes
}

/// A class representing an Amazon DynamoDB table containing movie
/// information.
public class MovieTable {
    var ddbClient: DynamoDBClient?
    let tableName: String

    /// Create an object representing a movie table in an Amazon DynamoDB
    /// database.
    ///
    /// - Parameters:
    ///   - region: The optional Amazon Region to create the database in.
    ///   - tableName: The name to assign to the table. If not specified, a
    ///     random table name is generated automatically.
    ///
    /// > Note: The table is not necessarily available when this function
    /// returns. Use `tableExists()` to check for its availability, or
    /// `awaitTableActive()` to wait until the table's status is reported as
    /// ready to use by Amazon DynamoDB.
    ///
    init(region: String? = nil, tableName: String) async throws {
        do {
            let config = try await DynamoDBClient.DynamoDBClientConfiguration()
            if let region = region {
                config.region = region
            }

            self.ddbClient = DynamoDBClient(config: config)
            self.tableName = tableName

            try await self.createTable()
        } catch {
            print("ERROR: ", dump(error, name: "Initializing Amazon
 DynamoDBClient client"))
```

```
            throw error
        }
    }

    ///
    /// Create a movie table in the Amazon DynamoDB data store.
    ///
    private func createTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = CreateTableInput(
                attributeDefinitions: [
                    DynamoDBClientTypes.AttributeDefinition(attributeName:
"year", attributeType: .n),
                    DynamoDBClientTypes.AttributeDefinition(attributeName:
"title", attributeType: .s)
                ],
                billingMode: DynamoDBClientTypes.BillingMode.payPerRequest,
                keySchema: [
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "year",
keyType: .hash),
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "title",
keyType: .range)
                ],
                tableName: self.tableName
            )
            let output = try await client.createTable(input: input)
            if output.tableDescription == nil {
                throw MoviesError.TableNotFound
            }
        } catch {
            print("ERROR: createTable:", dump(error))
            throw error
        }
    }


    /// Check to see if the table exists online yet.
    ///
    /// - Returns: `true` if the table exists, or `false` if not.
    ///
```

```
    func tableExists() async throws -> Bool {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DescribeTableInput(
                tableName: tableName
            )
            let output = try await client.describeTable(input: input)
            guard let description = output.table else {
                throw MoviesError.TableNotFound
            }

            return description.tableName == self.tableName
        } catch {
            print("ERROR: tableExists:", dump(error))
            throw error
        }
    }


    ///
    /// Waits for the table to exist and for its status to be active.
    ///
    func awaitTableActive() async throws {
        while try (await self.tableExists() == false) {
            do {
                let duration = UInt64(0.25 * 1_000_000_000) // Convert .25
seconds to nanoseconds.
                try await Task.sleep(nanoseconds: duration)
            } catch {
                print("Sleep error:", dump(error))
            }
        }

        while try (await self.getTableStatus() != .active) {
            do {
                let duration = UInt64(0.25 * 1_000_000_000) // Convert .25
seconds to nanoseconds.
                try await Task.sleep(nanoseconds: duration)
            } catch {
                print("Sleep error:", dump(error))
            }
```

```
        }
    }


    ///
    /// Deletes the table from Amazon DynamoDB.
    ///
    func deleteTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteTableInput(
                tableName: self.tableName
            )
            _ = try await client.deleteTable(input: input)
        } catch {
            print("ERROR: deleteTable:", dump(error))
            throw error
        }
    }


    /// Get the table's status.
    ///
    /// - Returns: The table status, as defined by the
    ///   `DynamoDBClientTypes.TableStatus` enum.
    ///
    func getTableStatus() async throws -> DynamoDBClientTypes.TableStatus {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DescribeTableInput(
                tableName: self.tableName
            )
            let output = try await client.describeTable(input: input)
            guard let description = output.table else {
                throw MoviesError.TableNotFound
            }
            guard let status = description.tableStatus else {
                throw MoviesError.StatusUnknown
```

```
            }
            return status
        } catch {
            print("ERROR: getTableStatus:", dump(error))
            throw error
        }
    }


    /// Populate the movie database from the specified JSON file.
    ///
    /// - Parameter jsonPath: Path to a JSON file containing movie data.
    ///
    func populate(jsonPath: String) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Create a Swift `URL` and use it to load the file into a `Data`
            // object. Then decode the JSON into an array of `Movie` objects.

            let fileUrl = URL(fileURLWithPath: jsonPath)
            let jsonData = try Data(contentsOf: fileUrl)

            var movieList = try JSONDecoder().decode([Movie].self, from:
jsonData)

            // Truncate the list to the first 200 entries or so for this example.

            if movieList.count > 200 {
                movieList = Array(movieList[...199])
            }

            // Before sending records to the database, break the movie list into
            // 25-entry chunks, which is the maximum size of a batch item
request.

            let count = movieList.count
            let chunks = stride(from: 0, to: count, by: 25).map {
                Array(movieList[$0 ..< Swift.min($0 + 25, count)])
            }
```

```
            // For each chunk, create a list of write request records and
populate
            // them with `PutRequest` requests, each specifying one movie from
the
            // chunk. Once the chunk's items are all in the `PutRequest` list,
            // send them to Amazon DynamoDB using the
            // `DynamoDBClient.batchWriteItem()` function.

            for chunk in chunks {
                var requestList: [DynamoDBClientTypes.WriteRequest] = []

                for movie in chunk {
                    let item = try await movie.getAsItem()
                    let request = DynamoDBClientTypes.WriteRequest(
                        putRequest: .init(
                            item: item
                        )
                    )
                    requestList.append(request)
                }

                let input = BatchWriteItemInput(requestItems: [tableName:
requestList])
                _ = try await client.batchWriteItem(input: input)
            }
        } catch {
            print("ERROR: populate:", dump(error))
            throw error
        }
    }


    /// Add a movie specified as a `Movie` structure to the Amazon DynamoDB
    /// table.
    ///
    /// - Parameter movie: The `Movie` to add to the table.
    ///
    func add(movie: Movie) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Get a DynamoDB item containing the movie data.
```

```
            let item = try await movie.getAsItem()

            // Send the `PutItem` request to Amazon DynamoDB.

            let input = PutItemInput(
                item: item,
                tableName: self.tableName
            )
            _ = try await client.putItem(input: input)
        } catch {
            print("ERROR: add movie:", dump(error))
            throw error
        }
    }


    /// Given a movie's details, add a movie to the Amazon DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title as a `String`.
    ///   - year: The release year of the movie (`Int`).
    ///   - rating: The movie's rating if available (`Double`; default is
    ///     `nil`).
    ///   - plot: A summary of the movie's plot (`String`; default is `nil`,
    ///     indicating no plot summary is available).
    ///
    func add(title: String, year: Int, rating: Double? = nil,
             plot: String? = nil) async throws
    {
        do {
            let movie = Movie(title: title, year: year, rating: rating, plot:
plot)
            try await self.add(movie: movie)
        } catch {
            print("ERROR: add with fields:", dump(error))
            throw error
        }
    }


    /// Return a `Movie` record describing the specified movie from the Amazon
    /// DynamoDB table.
    ///
    /// - Parameters:
```

```
    ///   - title: The movie's title (`String`).
    ///   - year: The movie's release year (`Int`).
    ///
    /// - Throws: `MoviesError.ItemNotFound` if the movie isn't in the table.
    ///
    /// - Returns: A `Movie` record with the movie's details.
    func get(title: String, year: Int) async throws -> Movie {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = GetItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            let output = try await client.getItem(input: input)
            guard let item = output.item else {
                throw MoviesError.ItemNotFound
            }

            let movie = try Movie(withItem: item)
            return movie
        } catch {
            print("ERROR: get:", dump(error))
            throw error
        }
    }


    /// Get all the movies released in the specified year.
    ///
    /// - Parameter year: The release year of the movies to return.
    ///
    /// - Returns: An array of `Movie` objects describing each matching movie.
    ///
    func getMovies(fromYear year: Int) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }
```

```
            let input = QueryInput(
                expressionAttributeNames: [
                    "#y": "year"
                ],
                expressionAttributeValues: [
                    ":y": .n(String(year))
                ],
                keyConditionExpression: "#y = :y",
                tableName: self.tableName
            )
            // Use "Paginated" to get all the movies.
            // This lets the SDK handle the 'lastEvaluatedKey' property in
"QueryOutput".

            let pages = client.queryPaginated(input: input)

            var movieList: [Movie] = []
            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Convert the found movies into `Movie` objects and return an
array
                // of them.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList
        } catch {
            print("ERROR: getMovies:", dump(error))
            throw error
        }
    }


    /// Return an array of `Movie` objects released in the specified range of
    /// years.
    ///
```

```
    /// - Parameters:
    ///   - firstYear: The first year of movies to return.
    ///   - lastYear: The last year of movies to return.
    ///   - startKey: A starting point to resume processing; always use `nil`.
    ///
    /// - Returns: An array of `Movie` objects describing the matching movies.
    ///
    /// > Note: The `startKey` parameter is used by this function when
    ///   recursively calling itself, and should always be `nil` when calling
    ///   directly.
    ///
    func getMovies(firstYear: Int, lastYear: Int,
                   startKey: [Swift.String: DynamoDBClientTypes.AttributeValue]?
 = nil)
        async throws -> [Movie]
    {
        do {
            var movieList: [Movie] = []

            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = ScanInput(
                consistentRead: true,
                exclusiveStartKey: startKey,
                expressionAttributeNames: [
                    "#y": "year" // `year` is a reserved word, so use `#y`
 instead.
                ],
                expressionAttributeValues: [
                    ":y1": .n(String(firstYear)),
                    ":y2": .n(String(lastYear))
                ],
                filterExpression: "#y BETWEEN :y1 AND :y2",
                tableName: self.tableName
            )

            let pages = client.scanPaginated(input: input)

            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
```

```
                }

                // Build an array of `Movie` objects for the returned items.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList

        } catch {
            print("ERROR: getMovies with scan:", dump(error))
            throw error
        }
    }


    /// Update the specified movie with new `rating` and `plot` information.
    ///
    /// - Parameters:
    ///   - title: The title of the movie to update.
    ///   - year: The release year of the movie to update.
    ///   - rating: The new rating for the movie.
    ///   - plot: The new plot summary string for the movie.
    ///
    /// - Returns: An array of mappings of attribute names to their new
    ///   listing each item actually changed. Items that didn't need to change
    ///   aren't included in this list. `nil` if no changes were made.
    ///
    func update(title: String, year: Int, rating: Double? = nil, plot: String? =
 nil) async throws
        -> [Swift.String: DynamoDBClientTypes.AttributeValue]?
    {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Build the update expression and the list of expression attribute
            // values. Include only the information that's changed.

            var expressionParts: [String] = []
```

```
            var attrValues: [Swift.String: DynamoDBClientTypes.AttributeValue] =
[:]

            if rating != nil {
                expressionParts.append("info.rating=:r")
                attrValues[":r"] = .n(String(rating!))
            }
            if plot != nil {
                expressionParts.append("info.plot=:p")
                attrValues[":p"] = .s(plot!)
            }
            let expression = "set \(expressionParts.joined(separator: ", "))"

            let input = UpdateItemInput(
                // Create substitution tokens for the attribute values, to ensure
                // no conflicts in expression syntax.
                expressionAttributeValues: attrValues,
                // The key identifying the movie to update consists of the
release
                // year and title.
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                returnValues: .updatedNew,
                tableName: self.tableName,
                updateExpression: expression
            )
            let output = try await client.updateItem(input: input)

            guard let attributes: [Swift.String:
DynamoDBClientTypes.AttributeValue] = output.attributes else {
                throw MoviesError.InvalidAttributes
            }
            return attributes
        } catch {
            print("ERROR: update:", dump(error))
            throw error
        }
    }


    /// Delete a movie, given its title and release year.
    ///
```

```
    /// - Parameters:
    ///   - title: The movie's title.
    ///   - year: The movie's release year.
    ///
    func delete(title: String, year: Int) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            _ = try await client.deleteItem(input: input)
        } catch {
            print("ERROR: delete:", dump(error))
            throw error
        }
    }
}
```

As estruturas usadas pela MovieTable classe para representar filmes.

```
import Foundation
import AWSDynamoDB

/// The optional details about a movie.
public struct Details: Codable {
    /// The movie's rating, if available.
    var rating: Double?
    /// The movie's plot, if available.
    var plot: String?
}

/// A structure describing a movie. The `year` and `title` properties are
/// required and are used as the key for Amazon DynamoDB operations. The
/// `info` sub-structure's two properties, `rating` and `plot`, are optional.
```

```
public struct Movie: Codable {
    /// The year in which the movie was released.
    var year: Int
    /// The movie's title.
    var title: String
    /// A `Details` object providing the optional movie rating and plot
    /// information.
    var info: Details

    /// Create a `Movie` object representing a movie, given the movie's
    /// details.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The year in which the movie was released (`Int`).
    ///   - rating: The movie's rating (optional `Double`).
    ///   - plot: The movie's plot (optional `String`)
    init(title: String, year: Int, rating: Double? = nil, plot: String? = nil) {
        self.title = title
        self.year = year

        self.info = Details(rating: rating, plot: plot)
    }

    /// Create a `Movie` object representing a movie, given the movie's
    /// details.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The year in which the movie was released (`Int`).
    ///   - info: The optional rating and plot information for the movie in a
    ///     `Details` object.
    init(title: String, year: Int, info: Details?){
        self.title = title
        self.year = year

        if info != nil {
            self.info = info!
        } else {
            self.info = Details(rating: nil, plot: nil)
        }
    }

    ///
```

```
    /// Return a new `MovieTable` object, given an array mapping string to Amazon
    /// DynamoDB attribute values.
    ///
    /// - Parameter item: The item information provided to the form used by
    ///   DynamoDB. This is an array of strings mapped to
    ///   `DynamoDBClientTypes.AttributeValue` values.
    init(withItem item: [Swift.String:DynamoDBClientTypes.AttributeValue]) throws
  {
        // Read the attributes.

        guard let titleAttr = item["title"],
              let yearAttr = item["year"] else {
            throw MoviesError.ItemNotFound
        }
        let infoAttr = item["info"] ?? nil

        // Extract the values of the title and year attributes.

        if case .s(let titleVal) = titleAttr {
            self.title = titleVal
        } else {
            throw MoviesError.InvalidAttributes
        }

        if case .n(let yearVal) = yearAttr {
            self.year = Int(yearVal)!
        } else {
            throw MoviesError.InvalidAttributes
        }

        // Extract the rating and/or plot from the `info` attribute, if
        // they're present.

        var rating: Double? = nil
        var plot: String? = nil

        if infoAttr != nil, case .m(let infoVal) = infoAttr {
            let ratingAttr = infoVal["rating"] ?? nil
            let plotAttr = infoVal["plot"] ?? nil

            if ratingAttr != nil, case .n(let ratingVal) = ratingAttr {
                rating = Double(ratingVal) ?? nil
            }
            if plotAttr != nil, case .s(let plotVal) = plotAttr {
```

```
                plot = plotVal
            }
        }

        self.info = Details(rating: rating, plot: plot)
    }

    ///
    /// Return an array mapping attribute names to Amazon DynamoDB attribute
    /// values, representing the contents of the `Movie` record as a DynamoDB
    /// item.
    ///
    /// - Returns: The movie item as an array of type
    ///   `[Swift.String:DynamoDBClientTypes.AttributeValue]`.
    ///
    func getAsItem() async throws ->
[Swift.String:DynamoDBClientTypes.AttributeValue]  {
        // Build the item record, starting with the year and title, which are
        // always present.

        var item: [Swift.String:DynamoDBClientTypes.AttributeValue] = [
            "year": .n(String(self.year)),
            "title": .s(self.title)
        ]

        // Add the `info` field with the rating and/or plot if they're
        // available.

        var details: [Swift.String:DynamoDBClientTypes.AttributeValue] = [:]
        if (self.info.rating != nil || self.info.plot != nil) {
            if self.info.rating != nil {
                details["rating"] = .n(String(self.info.rating!))
            }
            if self.info.plot != nil {
                details["plot"] = .s(self.info.plot!)
            }
        }
        item["info"] = .m(details)

        return item
    }
}
```

Um programa que usa a MovieTable classe para acessar um banco de dados do DynamoDB.

```
import ArgumentParser
import ClientRuntime
import Foundation

import AWSDynamoDB


@testable import MovieList

extension String {
    // Get the directory if the string is a file path.
    func directory() -> String {
        guard let lastIndex = lastIndex(of: "/") else {
            print("Error: String directory separator not found.")
            return ""
        }
        return String(self[...lastIndex])
    }
}

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The path of the sample movie data JSON file.")
    var jsonPath: String = #file.directory() + "../../../../../resources/
sample_files/movies.json"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion: String?

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"
```

```
    /// Configuration details for the command.
    static var configuration = CommandConfiguration(
        commandName: "basics",
        abstract: "A basic scenario demonstrating the usage of Amazon DynamoDB.",
        discussion: """
        An example showing how to use Amazon DynamoDB to perform a series of
        common database activities on a simple movie database.
        """
    )

    /// Called by ``main()`` to asynchronously run the AWS example.
    func runAsync() async throws {
        print("Welcome to the AWS SDK for Swift basic scenario for Amazon
DynamoDB!")

        //=====================================================================
        // 1. Create the table. The Amazon DynamoDB table is represented by
        //    the `MovieTable` class.
        //=====================================================================

        let tableName = "ddb-movies-sample-\(Int.random(in: 1 ... Int.max))"

        print("Creating table \"\(tableName)\"...")

        let movieDatabase = try await MovieTable(region: awsRegion,
                                                 tableName: tableName)

        print("\nWaiting for table to be ready to use...")
        try await movieDatabase.awaitTableActive()

        //=====================================================================
        // 2. Add a movie to the table.
        //=====================================================================

        print("\nAdding a movie...")
        try await movieDatabase.add(title: "Avatar: The Way of Water", year:
2022)
        try await movieDatabase.add(title: "Not a Real Movie", year: 2023)

        //=====================================================================
        // 3. Update the plot and rating of the movie using an update
        //    expression.
        //=====================================================================
```

```
        print("\nAdding details to the added movie...")
        _ = try await movieDatabase.update(title: "Avatar: The Way of Water",
year: 2022,
                                           rating: 9.2, plot: "It's a sequel.")

        //=====================================================================
        // 4. Populate the table from the JSON file.
        //=====================================================================

        print("\nPopulating the movie database from JSON...")
        try await movieDatabase.populate(jsonPath: jsonPath)

        //=====================================================================
        // 5. Get a specific movie by key. In this example, the key is a
        //    combination of `title` and `year`.
        //=====================================================================

        print("\nLooking for a movie in the table...")
        let gotMovie = try await movieDatabase.get(title: "This Is the End",
year: 2013)

        print("Found the movie \"\(gotMovie.title)\", released in
\(gotMovie.year).")
        print("Rating: \(gotMovie.info.rating ?? 0.0).")
        print("Plot summary: \(gotMovie.info.plot ?? "None.")")

        //=====================================================================
        // 6. Delete a movie.
        //=====================================================================

        print("\nDeleting the added movie...")
        try await movieDatabase.delete(title: "Avatar: The Way of Water", year:
2022)

        //=====================================================================
        // 7. Use a query with a key condition expression to return all movies
        //    released in a given year.
        //=====================================================================

        print("\nGetting movies released in 1994...")
        let movieList = try await movieDatabase.getMovies(fromYear: 1994)
        for movie in movieList {
            print("    \(movie.title)")
        }
```

```
        //=====================================================================
        // 8. Use `scan()` to return movies released in a range of years.
        //=====================================================================

        print("\nGetting movies released between 1993 and 1997...")
        let scannedMovies = try await movieDatabase.getMovies(firstYear: 1993,
 lastYear: 1997)
        for movie in scannedMovies {
            print("    \(movie.title) (\(movie.year))")
        }

        //=====================================================================
        // 9. Delete the table.
        //=====================================================================

        print("\nDeleting the table...")
        try await movieDatabase.deleteTable()
    }
}

@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - BatchWriteItem
  - CreateTable
  - DeleteItem

- DeleteTable
- DescribeTable
- GetItem
- PutItem
- Query
- Scan
- UpdateItem

Ações para o DynamoDB usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do DynamoDB com SDKs. AWS Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do DynamoDB e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do DynamoDB AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de API do Amazon DynamoDB.

Exemplos

- Use BatchExecuteStatementcom um AWS SDK
- Use BatchGetItemcom um AWS SDK ou CLI
- Use BatchWriteItemcom um AWS SDK ou CLI
- Use CreateTablecom um AWS SDK ou CLI
- Use DeleteItemcom um AWS SDK ou CLI
- Use DeleteTablecom um AWS SDK ou CLI
- Use DescribeTablecom um AWS SDK ou CLI
- Use DescribeTimeToLivecom um AWS SDK ou CLI
- Use ExecuteStatementcom um AWS SDK
- Use GetItemcom um AWS SDK ou CLI
- Use ListTablescom um AWS SDK ou CLI
- Use PutItemcom um AWS SDK ou CLI

- Use o Query com um AWS SDK ou CLI
- Use o Scan com um AWS SDK ou CLI
- Use UpdateItemcom um AWS SDK ou CLI
- Use UpdateTablecom um AWS SDK ou CLI
- Use UpdateTimeToLivecom um AWS SDK ou CLI

Use **`BatchExecuteStatement`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `BatchExecuteStatement`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Excluir dados usando DELETE do PartiQL
- Inserir dados usando INSERT do PartiQL
- Consultar uma tabela usando lotes de instruções PartiQL
- Consultar dados usando SELECT do PartiQL
- Atualizar dados usando UPDATE do PartiQL

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use lotes de instruções INSERT para adicionar itens.

```
        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table into which the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
```

```
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE
{{'title': ?, 'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset +=
25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await
Client.BatchExecuteStatementAsync(new BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });
```

```
                        // Wait between batches for movies to be successfully
added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
```

```
            {
                return null!;
            }
        }
```

Use lotes de instruções SELECT para obter itens.

```
        /// <summary>
        /// Gets movies from the movie table by
        /// using an Amazon DynamoDB PartiQL SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year1">The year of the first movie.</param>
        /// <param name="year2">The year of the second movie.</param>
        /// <returns>True if successful.</returns>
        public static async Task<bool> GetBatch(
            string tableName,
            string title1,
            string title2,
            int year1,
            int year2)
        {
            var getBatch = $"SELECT * FROM {tableName} WHERE title = ? AND year
 = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
```

```
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            if (response.Responses.Count > 0)
            {
                response.Responses.ForEach(r =>
                {
                    if (r.Item.Any())
                    {

Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
                    }
                });
                return true;
            }
            else
            {
                Console.WriteLine($"Couldn't find either {title1} or {title2}.");
                return false;
            }

        }
```

Use lotes de instruções UPDATE para atualizar itens.

```
        /// <summary>
        /// Updates information for multiple movies.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// movies to be updated.</param>
        /// <param name="producer1">The producer name for the first movie
        /// to update.</param>
```

```
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year that the first movie was released.</
param>
        /// <param name="producer2">The producer name for the second
        /// movie to update.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year that the second movie was released.</
param>
        /// <returns>A Boolean value that indicates the success of the update.</
returns>
        public static async Task<bool> UpdateBatch(
            string tableName,
            string producer1,
            string title1,
            int year1,
            string producer2,
            string title2,
            int year2)
        {

            string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title
 = ? AND year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer1 },
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer2 },
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
```

```
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
 BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Use lotes de instruções DELETE para excluir itens.

```
        /// <summary>
        /// Deletes multiple movies using a PartiQL BatchExecuteAsync
        /// statement.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// moves that will be deleted.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year the first movie was released.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year the second movie was released.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeleteBatch(
            string tableName,
            string title1,
            int year1,
            string title2,
            int year2)
        {

            string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND
 year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
```

```
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use lotes de instruções INSERT para adicionar itens.

```
    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
                "On a scale of 1 - 10, how do you rate it? ",
                1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/
n) "));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
              << (titles.size() == 1 ? " movie " : " movies ")
              << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
```

```
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

            // Create attribute for the info map.
            Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute
= Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            ratingAttribute->SetN(ratings[i]);
            infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            plotAttribute->SetS(plots[i]);
            infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
            attributes.push_back(infoMapAttribute);
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

Use lotes de instruções SELECT para obter itens.

```
   // 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
   {
       Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
               titles.size());
       std::stringstream sqlStream;
       sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                 << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

       std::string sql(sqlStream.str());

       for (size_t i = 0; i < statements.size(); ++i) {
           statements[i].SetStatement(sql);
           Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
           attributes.push_back(
                   Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
           statements[i].SetParameters(attributes);
       }

       Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

       request.SetStatements(statements);

       Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
               request);
       if (outcome.IsSuccess()) {
           const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

           const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

           for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
               const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

               printMovieInfo(item);
           }
       }
```

```
        else {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

Use lotes de instruções UPDATE para atualizar itens.

```
    // 4. Update the data for multiple movies using "Update" statements.
 (BatchExecuteStatement)

    for (size_t i = 0; i < titles.size(); ++i) {
        ratings[i] = askQuestionForFloatRange(
                Aws::String("\nLet's update your the movie, \"") + titles[i] +
                ".\nYou rated it  " + std::to_string(ratings[i])
                + ", what new rating would you give it? ", 1, 10);
    }

    std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
 std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";


        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
```

```
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

Use lotes de instruções DELETE para excluir itens.

```
    // 6. Delete multiple movies using "Delete" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
```

```
        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
 dynamoClient.BatchExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina uma estrutura de receptor de função para o exemplo.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)
```

```
// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

Use lotes de instruções INSERT para adicionar itens.

```
// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies
 to the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year, movie.Info})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(fmt.Sprintf(
    "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n",
 err)
 }
 return err
```

```
}
```

Use lotes de instruções SELECT para obter itens.

```
// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
 from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
 ([]Movie, error) {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 var outMovies []Movie
 if err != nil {
  log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
 } else {
  for _, response := range output.Responses {
   var movie Movie
   err = attributevalue.UnmarshalMap(response.Item, &movie)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    outMovies = append(outMovies, movie)
   }
```

```
  }
 }
 return outMovies, err
}
```

Use lotes de instruções UPDATE para atualizar itens.

```
// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the
 rating of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
 ratings []float64) error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{ratings[index],
 movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

Use lotes de instruções DELETE para excluir itens.

```
// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
 movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
```

```
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um lote de itens usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const breakfastFoods = ["Eggs", "Bacon", "Sausage"];
  const command = new BatchExecuteStatementCommand({
    Statements: breakfastFoods.map((food) => ({
      Statement: `INSERT INTO BreakfastFoods value {'Name':?}`,
      Parameters: [food],
    })),
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Obtenha um lote de itens usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
```

```
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchExecuteStatementCommand({
    Statements: [
      {
        Statement: "SELECT * FROM PepperMeasurements WHERE Unit=?",
        Parameters: ["Teaspoons"],
        ConsistentRead: true,
      },
      {
        Statement: "SELECT * FROM PepperMeasurements WHERE Unit=?",
        Parameters: ["Grams"],
        ConsistentRead: true,
      },
    ],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Atualize um lote de itens usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const eggUpdates = [
```

```
    ["duck", "fried"],
    ["chicken", "omelette"],
  ];
  const command = new BatchExecuteStatementCommand({
    Statements: eggUpdates.map((change) => ({
      Statement: "UPDATE Eggs SET Style=? where Variety=?",
      Parameters: [change[1], change[0]],
    })),
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Exclua um lote de itens usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchExecuteStatementCommand({
    Statements: [
      {
        Statement: "DELETE FROM Flavors where Name=?",
        Parameters: ["Grape"],
      },
      {
        Statement: "DELETE FROM Flavors where Name=?",
        Parameters: ["Strawberry"],
      },
    ],
  });

  const response = await docClient.send(command);
```

```
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function insertItemByPartiQLBatch(string $statement, array
 $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
```

```
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because
 the
        resource transforms input and output from plain old Python objects
 (POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated
 with
                           each statement. This list must be in the same order as
 the
                           statements.
        :return: The responses returned from running the statements, if any.
        """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
```

```
                Statements=[
                    {"Statement": statement, "Parameters": params}
                    for statement, params in zip(statements, param_list)
                ]
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the
 table "
                    "does not exist."
                )
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why:
 %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

- Para obter detalhes da API, consulte a BatchExecuteStatementReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Leia um lote de itens usando o PartiQL.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table
```

```
  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Selects a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_select(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "SELECT * FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

Exclua um lote de itens usando o PartiQL.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_write(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
```

```
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para Ruby da API.

Use **`BatchGetItem`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `BatchGetItem`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;

namespace LowLevelBatchGet
{
    public class LowLevelBatchGet
    {
        private static readonly string _table1Name = "Forum";
        private static readonly string _table2Name = "Thread";

        public static async void
 RetrieveMultipleItemsBatchGet(AmazonDynamoDBClient client)
        {
            var request = new BatchGetItemRequest
            {
                RequestItems = new Dictionary<string, KeysAndAttributes>()
            {
```

```
                    { _table1Name,
                      new KeysAndAttributes
                      {
                          Keys = new List<Dictionary<string, AttributeValue> >()
                          {
                              new Dictionary<string, AttributeValue>()
                              {
                                  { "Name", new AttributeValue {
                                    S = "Amazon DynamoDB"
                                  } }
                              },
                              new Dictionary<string, AttributeValue>()
                              {
                                  { "Name", new AttributeValue {
                                    S = "Amazon S3"
                                  } }
                              }
                          }
                      }},
                    {
                        _table2Name,
                        new KeysAndAttributes
                        {
                            Keys = new List<Dictionary<string, AttributeValue> >()
                            {
                                new Dictionary<string, AttributeValue>()
                                {
                                    { "ForumName", new AttributeValue {
                                          S = "Amazon DynamoDB"
                                      } },
                                    { "Subject", new AttributeValue {
                                          S = "DynamoDB Thread 1"
                                      } }
                                },
                                new Dictionary<string, AttributeValue>()
                                {
                                    { "ForumName", new AttributeValue {
                                          S = "Amazon DynamoDB"
                                      } },
                                    { "Subject", new AttributeValue {
                                          S = "DynamoDB Thread 2"
                                      } }
                                },
                                new Dictionary<string, AttributeValue>()
```

```
                            {
                                { "ForumName", new AttributeValue {
                                      S = "Amazon S3"
                                  } },
                                { "Subject", new AttributeValue {
                                      S = "S3 Thread 1"
                                  } }
                            }
                        }
                    }
                }
            }
            };

            BatchGetItemResponse response;
            do
            {
                Console.WriteLine("Making request");
                response = await client.BatchGetItemAsync(request);

                // Check the response.
                var responses = response.Responses; // Attribute list in the
response.

                foreach (var tableResponse in responses)
                {
                    var tableResults = tableResponse.Value;
                    Console.WriteLine("Items retrieved from table {0}",
tableResponse.Key);
                    foreach (var item1 in tableResults)
                    {
                        PrintItem(item1);
                    }
                }

                // Any unprocessed keys? could happen if you exceed
ProvisionedThroughput or some other error.
                Dictionary<string, KeysAndAttributes> unprocessedKeys =
response.UnprocessedKeys;
                foreach (var unprocessedTableKeys in unprocessedKeys)
                {
                    // Print table name.
                    Console.WriteLine(unprocessedTableKeys.Key);
                    // Print unprocessed primary keys.
```

```
                    foreach (var key in unprocessedTableKeys.Value.Keys)
                    {
                        PrintItem(key);
                    }
                }

                request.RequestItems = unprocessedKeys;
            } while (response.UnprocessedKeys.Count > 0);
        }

        private static void PrintItem(Dictionary<string, AttributeValue>
 attributeList)
        {
            foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)
            {
                string attributeName = kvp.Key;
                AttributeValue value = kvp.Value;

                Console.WriteLine(
                    attributeName + " " +
                    (value.S == null ? "" : "S=[" + value.S + "]") +
                    (value.N == null ? "" : "N=[" + value.N + "]") +
                    (value.SS == null ? "" : "SS=[" + string.Join(",",
 value.SS.ToArray()) + "]") +
                    (value.NS == null ? "" : "NS=[" + string.Join(",",
 value.NS.ToArray()) + "]")
                    );
            }

 Console.WriteLine("************************************************");
        }

        static void Main()
        {
            var client = new AmazonDynamoDBClient();

            RetrieveMultipleItemsBatchGet(client);
        }
    }
}
```

- Para obter detalhes da API, consulte BatchGetItema Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_batch_get_item
#
# This function gets a batch of items from a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the keys of the items to get.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function dynamodb_batch_get_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_batch_get_item"
    echo "Get a batch of items from a DynamoDB table."
    echo " -i item  -- Path to json file containing the keys of the items to
 get."
    echo ""
  }
```

```
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  response=$(aws dynamodb batch-get-item \
    --request-items file://"$item")
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-get-item operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
```

```
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte BatchGetItemem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Batch get items from different Amazon DynamoDB tables.
/*!
  \sa batchGetItem()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::batchGetItem(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchGetItemRequest request;

    // Table1: Forum.
    Aws::String table1Name = "Forum";
    Aws::DynamoDB::Model::KeysAndAttributes table1KeysAndAttributes;

    // Table1: Projection expression.
    table1KeysAndAttributes.SetProjectionExpression("#n, Category, Messages,
 #v");

    // Table1: Expression attribute names.
    Aws::Http::HeaderValueCollection headerValueCollection;
    headerValueCollection.emplace("#n", "Name");
    headerValueCollection.emplace("#v", "Views");
    table1KeysAndAttributes.SetExpressionAttributeNames(headerValueCollection);

    // Table1: Set key name, type, and value to search.
    std::vector<Aws::String> nameValues = {"Amazon DynamoDB", "Amazon S3"};
```

```
    for (const Aws::String &name: nameValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetS(name);
        keys.emplace("Name", key);
        table1KeysAndAttributes.AddKeys(keys);
    }

    Aws::Map<Aws::String, Aws::DynamoDB::Model::KeysAndAttributes> requestItems;
    requestItems.emplace(table1Name, table1KeysAndAttributes);

    // Table2: ProductCatalog.
    Aws::String table2Name = "ProductCatalog";
    Aws::DynamoDB::Model::KeysAndAttributes table2KeysAndAttributes;
    table2KeysAndAttributes.SetProjectionExpression("Title, Price, Color");

    // Table2: Set key name, type, and value to search.
    std::vector<Aws::String> idValues = {"102", "103", "201"};
    for (const Aws::String &id: idValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetN(id);
        keys.emplace("Id", key);
        table2KeysAndAttributes.AddKeys(keys);
    }

    requestItems.emplace(table2Name, table2KeysAndAttributes);

    bool result = true;
    do {  // Use a do loop to handle pagination.
        request.SetRequestItems(requestItems);
        const Aws::DynamoDB::Model::BatchGetItemOutcome &outcome =
dynamoClient.BatchGetItem(
                request);

        if (outcome.IsSuccess()) {
            for (const auto &responsesMapEntry:
outcome.GetResult().GetResponses()) {
                Aws::String tableName = responsesMapEntry.first;
                const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &tableResults = responsesMapEntry.second;
                std::cout << "Retrieved " << tableResults.size()
                          << " responses for table '" << tableName << "'.\n"
                          << std::endl;
```

```
                if (tableName == "Forum") {

                    std::cout << "Name | Category | Message | Views" <<
std::endl;
                    for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Name").GetS() << " | ";
                        std::cout << item.at("Category").GetS() << " | ";
                        std::cout << (item.count("Message") == 0 ? "" : item.at(
                                "Messages").GetN()) << " | ";
                        std::cout << (item.count("Views") == 0 ? "" : item.at(
                                "Views").GetN()) << std::endl;
                    }
                }
                else {
                    std::cout << "Title | Price | Color" << std::endl;
                    for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Title").GetS() << " | ";
                        std::cout << (item.count("Price") == 0 ? "" : item.at(
                                "Price").GetN());
                        if (item.count("Color")) {
                            std::cout << " | ";
                            for (const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> &listItem: item.at(
                                    "Color").GetL())
                                std::cout << listItem->GetS() << " ";
                        }
                        std::cout << std::endl;
                    }
                }
                std::cout << std::endl;
            }

            // If necessary, repeat request for remaining items.
            requestItems = outcome.GetResult().GetUnprocessedKeys();
        }
        else {
            std::cerr << "Batch get item failed: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
            break;
        }
```

```
    } while (!requestItems.empty());

    return result;
}
```

- Para obter detalhes da API, consulte BatchGetItema Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para recuperar vários itens de uma tabela

O exemplo de `batch-get-items` a seguir lê vários itens da tabela `MusicCollection` usando um lote de três solicitações `GetItem` e solicita o número de unidades de capacidade de leitura consumidas pela operação. O comando retorna somente o atributo `AlbumTitle`.

```
aws dynamodb batch-get-item \
    --request-items file://request-items.json \
    --return-consumed-capacity TOTAL
```

Conteúdo de `request-items.json`:

```
{
    "MusicCollection": {
        "Keys": [
            {
                "Artist": {"S": "No One You Know"},
                "SongTitle": {"S": "Call Me Today"}
            },
            {
                "Artist": {"S": "Acme Band"},
                "SongTitle": {"S": "Happy Day"}
            },
            {
                "Artist": {"S": "No One You Know"},
                "SongTitle": {"S": "Scared of My Shadow"}
            }
        ],
```

```
        "ProjectionExpression":"AlbumTitle"
    }
}
```

Saída:

```
{
    "Responses": {
        "MusicCollection": [
            {
                "AlbumTitle": {
                    "S": "Somewhat Famous"
                }
            },
            {
                "AlbumTitle": {
                    "S": "Blue Sky Blues"
                }
            },
            {
                "AlbumTitle": {
                    "S": "Louder Than Ever"
                }
            }
        ]
    },
    "UnprocessedKeys": {},
    "ConsumedCapacity": [
        {
            "TableName": "MusicCollection",
            "CapacityUnits": 1.5
        }
    ]
}
```

Para obter mais informações, consulte Operações em lote no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte BatchGetItemem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Mostra como receber itens em lote usando o cliente de serviço.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class BatchReadItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
```

```
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItems(dynamoDbClient, tableName);
    }

    public static void getBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Make the batchGetItem request.
        BatchGetItemResponse batchGetItemResponse =
 dynamoDbClient.batchGetItem(batchGetItemRequest);

        // Extract and print the retrieved items.
        Map<String, List<Map<String, AttributeValue>>> responses =
 batchGetItemResponse.responses();
        if (responses.containsKey(tableName)) {
            List<Map<String, AttributeValue>> musicItems =
 responses.get(tableName);
            for (Map<String, AttributeValue> item : musicItems) {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            }
        } else {
            System.out.println("No items retrieved.");
```

```
        }
    }
}
```

Mostra como receber itens em lote usando o cliente de serviço e um paginador.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

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

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient,
 String tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());
```

```
        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Use batchGetItemPaginator for paginated requests.
        dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
            .flatMap(response -> response.responses().getOrDefault(tableName,
 Collections.emptyList()).stream())
            .forEach(item -> {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            });
    }
}
```

- Para obter detalhes da API, consulte BatchGetItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte BatchGet.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { BatchGetCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchGetCommand({
    // Each key in this object is the name of a table. This example refers
    // to a Books table.
    RequestItems: {
      Books: {
        // Each entry in Keys is an object that specifies a primary key.
        Keys: [
          {
            Title: "How to AWS",
          },
          {
            Title: "DynamoDB for DBAs",
          },
        ],
        // Only return the "Title" and "PageCount" attributes.
        ProjectionExpression: "Title, PageCount",
      },
    },
  });

  const response = await docClient.send(command);
  console.log(response.Responses.Books);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte BatchGetItema Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  RequestItems: {
    TABLE_NAME: {
      Keys: [
        { KEY_NAME: { N: "KEY_VALUE_1" } },
        { KEY_NAME: { N: "KEY_VALUE_2" } },
        { KEY_NAME: { N: "KEY_VALUE_3" } },
      ],
      ProjectionExpression: "KEY_NAME, ATTRIBUTE",
    },
  },
};

ddb.batchGetItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    data.Responses.TABLE_NAME.forEach(function (element, index, array) {
      console.log(element);
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

- Para obter detalhes da API, consulte BatchGetItema Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: obtém o item com o SongTitle “Somewhere Down The Road” das tabelas 'Music' e 'Songs' do DynamoDB.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
 'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
 Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-
DDBItem
```

Saída:

```
Name                           Value
----                           -----
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
CriticRating                   10
Genre                          Country
Price                          1.94
Artist                         No One You Know
```

```
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
CriticRating                    10
Genre                           Country
Price                           1.94
```

- Para obter detalhes da API, consulte BatchGetItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: obtém o item com o SongTitle “Somewhere Down The Road” das tabelas 'Music' e 'Songs' do DynamoDB.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
 'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
 Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-
DDBItem
```

Saída:

```
Name                            Value
----                            -----
Artist                          No One You Know
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
CriticRating                    10
```

```
Genre                          Country
Price                          1.94
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
CriticRating                   10
Genre                          Country
Price                          1.94
```

- Para obter detalhes da API, consulte BatchGetItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import decimal
import json
import logging
import os
import pprint
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
dynamodb = boto3.resource("dynamodb")

MAX_GET_SIZE = 100  # Amazon DynamoDB rejects a get batch larger than 100 items.


def do_batch_get(batch_keys):
    """
    Gets a batch of items from Amazon DynamoDB. Batches can contain keys from
    more than one table.
```

```
    When Amazon DynamoDB cannot process all items in a batch, a set of
unprocessed
    keys is returned. This function uses an exponential backoff algorithm to
retry
    getting the unprocessed keys until all are retrieved or the specified
    number of tries is reached.

    :param batch_keys: The set of keys to retrieve. A batch can contain at most
100
                       keys. Otherwise, Amazon DynamoDB returns an error.
    :return: The dictionary of retrieved items grouped under their respective
             table names.
    """
    tries = 0
    max_tries = 5
    sleepy_time = 1  # Start with 1 second of sleep, then exponentially increase.
    retrieved = {key: [] for key in batch_keys}
    while tries < max_tries:
        response = dynamodb.batch_get_item(RequestItems=batch_keys)
        # Collect any retrieved items and retry unprocessed keys.
        for key in response.get("Responses", []):
            retrieved[key] += response["Responses"][key]
        unprocessed = response["UnprocessedKeys"]
        if len(unprocessed) > 0:
            batch_keys = unprocessed
            unprocessed_count = sum(
                [len(batch_key["Keys"]) for batch_key in batch_keys.values()]
            )
            logger.info(
                "%s unprocessed keys returned. Sleep, then retry.",
unprocessed_count
            )
            tries += 1
            if tries < max_tries:
                logger.info("Sleeping for %s seconds.", sleepy_time)
                time.sleep(sleepy_time)
                sleepy_time = min(sleepy_time * 2, 32)
        else:
            break

    return retrieved
```

- Para obter detalhes da API, consulte a BatchGetItemReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Gets an array of `Movie` objects describing all the movies in the
    /// specified list. Any movies that aren't found in the list have no
    /// corresponding entry in the resulting array.
    ///
    /// - Parameters
    ///     - keys: An array of tuples, each of which specifies the title and
    ///       release year of a movie to fetch from the table.
    ///
    /// - Returns:
    ///     - An array of `Movie` objects describing each match found in the
    ///     table.
    ///
    /// - Throws:
    ///     - `MovieError.ClientUninitialized` if the DynamoDB client has not
    ///     been initialized.
    ///     - DynamoDB errors are thrown without change.
    func batchGet(keys: [(title: String, year: Int)]) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MovieError.ClientUninitialized
            }

            var movieList: [Movie] = []
```

```
            var keyItems: [[Swift.String: DynamoDBClientTypes.AttributeValue]] =
[]

            // Convert the list of keys into the form used by DynamoDB.

            for key in keys {
                let item: [Swift.String: DynamoDBClientTypes.AttributeValue] = [
                    "title": .s(key.title),
                    "year": .n(String(key.year))
                ]
                keyItems.append(item)
            }

            // Create the input record for `batchGetItem()`. The list of
requested
            // items is in the `requestItems` property. This array contains one
            // entry for each table from which items are to be fetched. In this
            // example, there's only one table containing the movie data.
            //
            // If we wanted this program to also support searching for matches
            // in a table of book data, we could add a second `requestItem`
            // mapping the name of the book table to the list of items we want to
            // find in it.
            let input = BatchGetItemInput(
                requestItems: [
                    self.tableName: .init(
                        consistentRead: true,
                        keys: keyItems
                    )
                ]
            )

            // Fetch the matching movies from the table.

            let output = try await client.batchGetItem(input: input)

            // Get the set of responses. If there aren't any, return the empty
            // movie list.

            guard let responses = output.responses else {
                return movieList
            }

            // Get the list of matching items for the table with the name
```

```
            // `tableName`.

            guard let responseList = responses[self.tableName] else {
                return movieList
            }

            // Create `Movie` items for each of the matching movies in the table
            // and add them to the `MovieList` array.

            for response in responseList {
                try movieList.append(Movie(withItem: response))
            }

            return movieList
        } catch {
            print("ERROR: batchGet", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte BatchGetItema referência da API AWS SDK for Swift.

Use **`BatchWriteItem`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `BatchWriteItem`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Grava um lote de itens na tabela de filmes.

```
/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The name of the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public List<Movie> ImportMovies(string movieFileName)
{
    var moviesList = new List<Movie>();
    if (!File.Exists(movieFileName))
    {
        return moviesList;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonSerializer.Deserialize<List<Movie>>(
        json,
        new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        });

    // Now return the first 250 entries.
    if (allMovies != null && allMovies.Any())
    {
        moviesList = allMovies.GetRange(0, 250);
    }
    return moviesList;
}

/// <summary>
/// Writes 250 items to the movie table.
/// </summary>
/// <param name="movieFileName">A string containing the full path to
/// the JSON file containing movie data.</param>
/// <param name="tableName">The name of the table to write items to.</param>
/// <returns>A long integer value representing the number of movies
/// imported from the JSON file.</returns>
public async Task<long> BatchWriteItemsAsync(
    string movieFileName, string tableName)
```

```
    {
        try
        {
            var movies = ImportMovies(movieFileName);
            if (!movies.Any())
            {
                Console.WriteLine("Couldn't find the JSON file with movie
data.");
                return 0;
            }

            var context = new DynamoDBContextBuilder()
                // Optional call to provide a specific instance of
IAmazonDynamoDB
                .WithDynamoDBClient(() => _amazonDynamoDB)
                .Build();

            var movieBatch = context.CreateBatchWrite<Movie>(
                new BatchWriteConfig()
                {
                    OverrideTableName = tableName
                });
            movieBatch.AddPutItems(movies);

            Console.WriteLine("Adding imported movies to the table.");
            await movieBatch.ExecuteAsync();

            return movies.Count;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table was not found during batch write operation.
{ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred during batch
write operation. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine($"An error occurred during batch write operation.
 {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte BatchWriteItema Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_batch_write_item
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the items to write.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_batch_write_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
```

```
  echo "function dynamodb_batch_write_item"
  echo "Write a batch of items into a DynamoDB table."
  echo " -i item  -- Path to json file containing the items to write."
  echo ""
}
while getopts "i:h" option; do
  case "${option}" in
    i) item="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$item" ]]; then
  errecho "ERROR: You must provide an item with the -i parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    item:   $item"
iecho ""

response=$(aws dynamodb batch-write-item \
  --request-items file://"$item")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports batch-write-item operation failed.$response"
  return 1
fi

return 0
```

```
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
#############################################################################
function aws_cli_error_log() {
  local err_code=$1
```

```
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte BatchWriteItemem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Batch write items from a JSON file.
/*!
  \sa batchWriteItem()
  \param jsonFilePath: JSON file path.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
/*
 * The input for this routine is a JSON file that you can download from the
 following URL:
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
SampleData.html.
 *
 * The JSON data uses the BatchWriteItem API request syntax. The JSON strings are
 * converted to AttributeValue objects. These AttributeValue objects will then
 generate
 * JSON strings when constructing the BatchWriteItem request, essentially
 outputting
 * their input.
 *
 * This is perhaps an artificial example, but it demonstrates the APIs.
 */

bool AwsDoc::DynamoDB::batchWriteItem(const Aws::String &jsonFilePath,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    std::ifstream fileStream(jsonFilePath);

    if (!fileStream) {
        std::cerr << "Error: could not open file '" << jsonFilePath << "'."
                  << std::endl;
    }

    std::stringstream stringStream;
    stringStream << fileStream.rdbuf();
    Aws::Utils::Json::JsonValue jsonValue(stringStream);

    Aws::DynamoDB::Model::BatchWriteItemRequest batchWriteItemRequest;
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> level1Map =
 jsonValue.View().GetAllObjects();
    for (const auto &level1Entry: level1Map) {
        const Aws::Utils::Json::JsonView &entriesView = level1Entry.second;
        const Aws::String &tableName = level1Entry.first;
        // The JSON entries at this level are as follows:
        //  key - table name
        //  value - list of request objects
        if (!entriesView.IsListType()) {
            std::cerr << "Error: JSON file entry '"
                      << tableName << "' is not a list." << std::endl;
            continue;
```

```
        }

        Aws::Utils::Array<Aws::Utils::Json::JsonView> entries =
 entriesView.AsArray();

        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;
        if (AwsDoc::DynamoDB::addWriteRequests(tableName, entries,
                                               writeRequests)) {
            batchWriteItemRequest.AddRequestItems(tableName, writeRequests);
        }
    }

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchWriteItemOutcome outcome =
 dynamoClient.BatchWriteItem(
            batchWriteItemRequest);

    if (outcome.IsSuccess()) {
        std::cout << "DynamoDB::BatchWriteItem was successful." << std::endl;
    }
    else {
        std::cerr << "Error with DynamoDB::BatchWriteItem. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}

//! Convert requests in JSON format to a vector of WriteRequest objects.
/*!
  \sa addWriteRequests()
  \param tableName: Name of the table for the write operations.
  \param requestsJson: Request data in JSON format.
  \param writeRequests: Vector to receive the WriteRequest objects.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::addWriteRequests(const Aws::String &tableName,
                                        const
 Aws::Utils::Array<Aws::Utils::Json::JsonView> &requestsJson,

 Aws::Vector<Aws::DynamoDB::Model::WriteRequest> &writeRequests) {
```

```
    for (size_t i = 0; i < requestsJson.GetLength(); ++i) {
        const Aws::Utils::Json::JsonView &requestsEntry = requestsJson[i];
        if (!requestsEntry.IsObject()) {
            std::cerr << "Error: incorrect requestsEntry type "
                      << requestsEntry.WriteReadable() << std::endl;
            return false;
        }

        Aws::Map<Aws::String, Aws::Utils::Json::JsonView> requestsMap =
requestsEntry.GetAllObjects();

        for (const auto &request: requestsMap) {
            const Aws::String &requestType = request.first;
            const Aws::Utils::Json::JsonView &requestJsonView = request.second;

            if (requestType == "PutRequest") {
                if (!requestJsonView.ValueExists("Item")) {
                    std::cerr << "Error: item key missing for requests "
                              << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributes;
                if (!getAttributeObjectsMap(requestJsonView.GetObject("Item"),
                                            attributes)) {
                    std::cerr << "Error getting attributes "
                              << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }

                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(attributes);
                writeRequests.push_back(
                        Aws::DynamoDB::Model::WriteRequest().WithPutRequest(
                                putRequest));
            }
            else {
                std::cerr << "Error: unimplemented request type '" << requestType
                          << "'." << std::endl;
            }
        }
    }

    return true;
```

```
}

//! Generate a map of AttributeValue objects from JSON records.
/*!
  \sa getAttributeObjectsMap()
  \param jsonView: JSONView of attribute records.
  \param writeRequests: Map to receive the AttributeValue objects.
  \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::getAttributeObjectsMap(const Aws::Utils::Json::JsonView
 &jsonView,
                                          Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue> &attributes) {
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> objectsMap =
 jsonView.GetAllObjects();
    for (const auto &entry: objectsMap) {
        const Aws::String &attributeKey = entry.first;
        const Aws::Utils::Json::JsonView &attributeJsonView = entry.second;

        if (!attributeJsonView.IsObject()) {
            std::cerr << "Error: attribute not an object "
                      << attributeJsonView.WriteReadable() << std::endl;
            return false;
        }

        attributes.emplace(attributeKey,

 Aws::DynamoDB::Model::AttributeValue(attributeJsonView));
    }

    return true;
}
```

- Para obter detalhes da API, consulte BatchWriteItema Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como adicionar vários itens a uma tabela

O exemplo de `batch-write-item` a seguir adiciona três novos itens à tabela `MusicCollection` usando um lote de três solicitações `PutItem`. Ele também solicita informações sobre o número de unidades de capacidade de gravação consumidas pela operação e quaisquer coleções de itens modificadas pela operação.

```
aws dynamodb batch-write-item \
    --request-items file://request-items.json \
    --return-consumed-capacity INDEXES \
    --return-item-collection-metrics SIZE
```

Conteúdo de `request-items.json`:

```
{
    "MusicCollection": [
        {
            "PutRequest": {
                "Item": {
                    "Artist": {"S": "No One You Know"},
                    "SongTitle": {"S": "Call Me Today"},
                    "AlbumTitle": {"S": "Somewhat Famous"}
                }
            }
        },
        {
            "PutRequest": {
                "Item": {
                    "Artist": {"S": "Acme Band"},
                    "SongTitle": {"S": "Happy Day"},
                    "AlbumTitle": {"S": "Songs About Life"}
                }
            }
        },
        {
            "PutRequest": {
                "Item": {
                    "Artist": {"S": "No One You Know"},
                    "SongTitle": {"S": "Scared of My Shadow"},
                    "AlbumTitle": {"S": "Blue Sky Blues"}
                }
            }
        }
    ]
```

```
}
```

Saída:

```
{
    "UnprocessedItems": {},
    "ItemCollectionMetrics": {
        "MusicCollection": [
            {
                "ItemCollectionKey": {
                    "Artist": {
                        "S": "No One You Know"
                    }
                },
                "SizeEstimateRangeGB": [
                    0.0,
                    1.0
                ]
            },
            {
                "ItemCollectionKey": {
                    "Artist": {
                        "S": "Acme Band"
                    }
                },
                "SizeEstimateRangeGB": [
                    0.0,
                    1.0
                ]
            }
        ]
    },
    "ConsumedCapacity": [
        {
            "TableName": "MusicCollection",
            "CapacityUnits": 6.0,
            "Table": {
                "CapacityUnits": 3.0
            },
            "LocalSecondaryIndexes": {
                "AlbumTitleIndex": {
                    "CapacityUnits": 3.0
                }
```

```
            }
        }
    ]
}
```

Para obter mais informações, consulte Operações em lote no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte BatchWriteItemem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
```

```
}



// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
 maxMovies int) (int, error) {
 var err error
 var item map[string]types.AttributeValue
 written := 0
 batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
 start := 0
 end := start + batchSize
 for start < maxMovies && start < len(movies) {
  var writeReqs []types.WriteRequest
  if end > len(movies) {
   end = len(movies)
  }
  for _, movie := range movies[start:end] {
   item, err = attributevalue.MarshalMap(movie)
   if err != nil {
    log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
 movie.Title, err)
   } else {
    writeReqs = append(
     writeReqs,
     types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
   }
  }
  _, err = basics.DynamoDbClient.BatchWriteItem(ctx,
 &dynamodb.BatchWriteItemInput{
   RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
  if err != nil {
   log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
 basics.TableName, err)
  } else {
   written += len(writeReqs)
  }
  start = end
  end += batchSize
 }
```

```
 return written, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
```

```
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte BatchWriteItema Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Insere vários itens em uma tabela usando o cliente de serviço.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
```

```
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        addBatchItems(dynamoDbClient, tableName);
    }

    public static void addBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Specify the updates you want to perform.
        List<WriteRequest> writeRequests = new ArrayList<>();

        // Set item 1.
        Map<String, AttributeValue> item1Attributes = new HashMap<>();
        item1Attributes.put("Artist",
 AttributeValue.builder().s("Artist1").build());
        item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
        item1Attributes.put("Comments", AttributeValue.builder().s("Great
 song!").build());
        item1Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle1").build());

 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attri
```

```
        // Set item 2.
        Map<String, AttributeValue> item2Attributes = new HashMap<>();
        item2Attributes.put("Artist",
 AttributeValue.builder().s("Artist2").build());
        item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
        item2Attributes.put("Comments", AttributeValue.builder().s("Nice
 melody.").build());
        item2Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle2").build());

 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attri

        try {
            // Create the BatchWriteItemRequest.
            BatchWriteItemRequest batchWriteItemRequest =
 BatchWriteItemRequest.builder()
                .requestItems(Map.of(tableName, writeRequests))
                .build();

            // Execute the BatchWriteItem operation.
            BatchWriteItemResponse batchWriteItemResponse =
 dynamoDbClient.batchWriteItem(batchWriteItemRequest);

            // Process the response.
            System.out.println("Batch write successful: " +
 batchWriteItemResponse);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Insira vários itens em uma tabela usando o cliente aprimorado.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
```

```
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import
 software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named
 Customer with these columns:
 *   - id - the id of the record that is the key
 *   - custName - the customer name
 *   - email - the email value
 *   - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnhancedBatchWriteItems {
        public static void main(String[] args) {
                Region region = Region.US_EAST_1;
                DynamoDbClient ddb = DynamoDbClient.builder()
                                .region(region)
                                .build();
                DynamoDbEnhancedClient enhancedClient =
 DynamoDbEnhancedClient.builder()
                                .dynamoDbClient(ddb)
                                .build();
                putBatchRecords(enhancedClient);
                ddb.close();
        }

        public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient)
 {
```

```
                try {
                        DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                                        TableSchema.fromBean(Customer.class));
                        DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                                        TableSchema.fromBean(Music.class));
                        LocalDate localDate = LocalDate.parse("2020-04-07");
                        LocalDateTime localDateTime = localDate.atStartOfDay();
                        Instant instant =
localDateTime.toInstant(ZoneOffset.UTC);

                        Customer record2 = new Customer();
                        record2.setCustName("Fred Pink");
                        record2.setId("id110");
                        record2.setEmail("fredp@noserver.com");
                        record2.setRegistrationDate(instant);

                        Customer record3 = new Customer();
                        record3.setCustName("Susan Pink");
                        record3.setId("id120");
                        record3.setEmail("spink@noserver.com");
                        record3.setRegistrationDate(instant);

                        Customer record4 = new Customer();
                        record4.setCustName("Jerry orange");
                        record4.setId("id101");
                        record4.setEmail("jorange@noserver.com");
                        record4.setRegistrationDate(instant);

                        BatchWriteItemEnhancedRequest
batchWriteItemEnhancedRequest = BatchWriteItemEnhancedRequest
                                        .builder()
                                        .writeBatches(

WriteBatch.builder(Customer.class) // add items to the Customer

         // table

.mappedTableResource(customerMappedTable)

.addPutItem(builder -> builder.item(record2))

.addPutItem(builder -> builder.item(record3))
```

```
 .addPutItem(builder -> builder.item(record4))
                                                                        .build(),

 WriteBatch.builder(Music.class) // delete an item from the Music

       // table

 .mappedTableResource(musicMappedTable)

 .addDeleteItem(builder -> builder.key(

       Key.builder().partitionValue(

                        "Famous Band")

                        .build()))
                                                                        .build())
                                        .build();

                        // Add three items to the Customer table and delete one
 item from the Music
                        // table.

 enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
                        System.out.println("done");

                } catch (DynamoDbException e) {
                        System.err.println(e.getMessage());
                        System.exit(1);
                }
        }
}
```

- Para obter detalhes da API, consulte BatchWriteItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte BatchWrite.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";
import { readFileSync } from "node:fs";

// These modules are local to our GitHub repository. We recommend cloning
// the project from GitHub if you want to run this example.
// For more information, see https://github.com/awsdocs/aws-doc-sdk-examples.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { chunkArray } from "@aws-doc-sdk-examples/lib/utils/util-array.js";

const dirname = dirnameFromMetaUrl(import.meta.url);

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const file = readFileSync(
    `${dirname}../../../../../resources/sample_files/movies.json`,
  );

  const movies = JSON.parse(file.toString());

  // chunkArray is a local convenience function. It takes an array and returns
  // a generator function. The generator function yields every N items.
  const movieChunks = chunkArray(movies, 25);
```

```
  // For every chunk of 25 movies, make one BatchWrite request.
  for (const chunk of movieChunks) {
    const putRequests = chunk.map((movie) => ({
      PutRequest: {
        Item: movie,
      },
    }));

    const command = new BatchWriteCommand({
      RequestItems: {
        // An existing table is required. A composite key of 'title' and 'year'
 is recommended
        // to account for duplicate titles.
        BatchWriteMoviesTable: putRequests,
      },
    });

    await docClient.send(command);
  }
};
```

- Para obter detalhes da API, consulte BatchWriteItema Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  RequestItems: {
```

```
    TABLE_NAME: [
      {
        PutRequest: {
          Item: {
            KEY: { N: "KEY_VALUE" },
            ATTRIBUTE_1: { S: "ATTRIBUTE_1_VALUE" },
            ATTRIBUTE_2: { N: "ATTRIBUTE_2_VALUE" },
          },
        },
      },
      {
        PutRequest: {
          Item: {
            KEY: { N: "KEY_VALUE" },
            ATTRIBUTE_1: { S: "ATTRIBUTE_1_VALUE" },
            ATTRIBUTE_2: { N: "ATTRIBUTE_2_VALUE" },
          },
        },
      },
    ],
  },
};

ddb.batchWriteItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte BatchWriteItema Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function writeBatch(string $TableName, array $Batch, int $depth = 2)
    {
        if (--$depth <= 0) {
            throw new Exception("Max depth exceeded. Please try with fewer batch
 items or increase depth.");
        }

        $marshal = new Marshaler();
        $total = 0;
        foreach (array_chunk($Batch, 25) as $Items) {
            foreach ($Items as $Item) {
                $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' =>
 ['Item' => $marshal->marshalItem($Item)]];
            }
            try {
                echo "Batching another " . count($Items) . " for a total of " .
 ($total += count($Items)) . " items!\n";
                $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
                $BatchWrite = [];
            } catch (Exception $e) {
                echo "uh oh...";
                echo $e->getMessage();
                die();
            }
            if ($total >= 250) {
                echo "250 movies is probably enough. Right? We can stop there.
\n";
                break;
            }
        }
    }
```

- Para obter detalhes da API, consulte BatchWriteItema Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria um item ou substitui um item por um novo item nas tabelas Music e Songs do DynamoDB.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
    Genre = 'Country'
    CriticRating = 10.0
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Para obter detalhes da API, consulte BatchWriteItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria um item ou substitui um item por um novo item nas tabelas Music e Songs do DynamoDB.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
```

```
    Genre = 'Country'
    CriticRating = 10.0
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Para obter detalhes da API, consulte BatchWriteItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
```

```
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts
sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at
least
                       the keys required by the schema that was specified when
the
                       table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a BatchWriteItemReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Fills an Amazon DynamoDB table with the specified data. Items are sent in
  # batches of 25 until all items are written.
  #
  # @param movies [Enumerable] The data to put in the table. Each item must
 contain at least
  #                            the keys required by the schema that was specified
 when the
  #                            table was created.
  def write_batch(movies)
    index = 0
    slice_size = 25
    while index < movies.length
      movie_items = []
```

```
      movies[index, slice_size].each do |movie|
        movie_items.append({ put_request: { item: movie } })
      end
      @dynamo_resource.client.batch_write_item({ request_items: { @table.name =>
 movie_items } })
      index += slice_size
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts(
      "Couldn't load data into table #{@table.name}. Here's why:"
    )
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte BatchWriteItema Referência AWS SDK para Ruby da API.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Populate the movie database from the specified JSON file.
    ///
    /// - Parameter jsonPath: Path to a JSON file containing movie data.
    ///
    func populate(jsonPath: String) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }
```

```
            // Create a Swift `URL` and use it to load the file into a `Data`
            // object. Then decode the JSON into an array of `Movie` objects.

            let fileUrl = URL(fileURLWithPath: jsonPath)
            let jsonData = try Data(contentsOf: fileUrl)

            var movieList = try JSONDecoder().decode([Movie].self, from:
jsonData)

            // Truncate the list to the first 200 entries or so for this example.

            if movieList.count > 200 {
                movieList = Array(movieList[...199])
            }

            // Before sending records to the database, break the movie list into
            // 25-entry chunks, which is the maximum size of a batch item
request.

            let count = movieList.count
            let chunks = stride(from: 0, to: count, by: 25).map {
                Array(movieList[$0 ..< Swift.min($0 + 25, count)])
            }

            // For each chunk, create a list of write request records and
populate
            // them with `PutRequest` requests, each specifying one movie from
the
            // chunk. Once the chunk's items are all in the `PutRequest` list,
            // send them to Amazon DynamoDB using the
            // `DynamoDBClient.batchWriteItem()` function.

            for chunk in chunks {
                var requestList: [DynamoDBClientTypes.WriteRequest] = []

                for movie in chunk {
                    let item = try await movie.getAsItem()
                    let request = DynamoDBClientTypes.WriteRequest(
                        putRequest: .init(
                            item: item
                        )
                    )
                    requestList.append(request)
```

```
                }

                let input = BatchWriteItemInput(requestItems: [tableName:
requestList])
                _ = try await client.batchWriteItem(input: input)
            }
        } catch {
            print("ERROR: populate:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte BatchWriteItema referência da API AWS SDK for Swift.

Use **`CreateTable`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Acelerar leituras com o DAX
- Cenários avançados do índice secundário global
- Criar uma tabela com um índice secundário global
- Criar uma tabela com o throughput a quente habilitado
- Crie e gerencie tabelas globais demonstrando o MREC
- Criar e gerenciar tabelas globais com MRSC
- Conceitos básicos de bancos de dados NoSQL
- Gerenciar índices secundários globais
- Gerenciar políticas baseadas em recursos
- Configurar o controle de Attribute-Based acesso
- Trabalhar com tabelas globais e a consistência final de replicação mutirregional (MREC)
- Trabalhar com índices secundários locais
- Trabalhar com a marcação de recursos

- Trabalhe com Streams e Time-to-Live
- Trabalhar com a criptografia de tabelas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    /// </summary>
    /// <param name="tableName">The name of the table to create.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> CreateMovieTableAsync(string tableName)
    {
        try
        {
            var response = await _amazonDynamoDB.CreateTableAsync(new
 CreateTableRequest
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition
                    {
                        AttributeName = "title",
                        AttributeType = ScalarAttributeType.S,
                    },
                    new AttributeDefinition
                    {
                        AttributeName = "year",
                        AttributeType = ScalarAttributeType.N,
                    },
```

```
                },
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement
                    {
                        AttributeName = "year",
                        KeyType = KeyType.HASH,
                    },
                    new KeySchemaElement
                    {
                        AttributeName = "title",
                        KeyType = KeyType.RANGE,
                    },
                },
                BillingMode = BillingMode.PAY_PER_REQUEST,
            });

            // Wait until the table is ACTIVE and then report success.
            Console.Write("Waiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = response.TableDescription.TableName,
            };

            TableStatus status;

            int sleepDuration = 2000;

            do
            {
                Thread.Sleep(sleepDuration);

                var describeTableResponse = await
_amazonDynamoDB.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException ex)
```

```
        {
            Console.WriteLine($"Table {tableName} already exists. {ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while creating
 table {tableName}. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating table
 {tableName}. {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to create.
#       -a attribute_definitions -- JSON file path of a list of attributes and
 their types.
```

```
#       -k key_schema -- JSON file path of a list of attributes and their key
 types.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_create_table() {
  local table_name attribute_definitions key_schema response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_create_table"
    echo "Creates an Amazon DynamoDB table with on-demand billing."
    echo " -n table_name  -- The name of the table to create."
    echo " -a attribute_definitions -- JSON file path of a list of attributes and
 their types."
    echo " -k key_schema -- JSON file path of a list of attributes and their key
 types."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:a:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      a) attribute_definitions="${OPTARG}" ;;
      k) key_schema="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_definitions" ]]; then
    errecho "ERROR: You must provide an attribute definitions json file path the
 -a parameter."
    usage
    return 1
  fi

  if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k
 parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    attribute_definitions:   $attribute_definitions"
  iecho "    key_schema:   $key_schema"
  iecho ""

  response=$(aws dynamodb create-table \
    --table-name "$table_name" \
    --attribute-definitions file://"$attribute_definitions" \
    --billing-mode PAY_PER_REQUEST \
    --key-schema file://"$key_schema" )

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
#############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
```

```
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte CreateTableem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon DynamoDB table.
/*!
  \sa createTable()
  \param tableName: Name for the DynamoDB table.
  \param primaryKey: Primary key for the DynamoDB table.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createTable(const Aws::String &tableName,
                                   const Aws::String &primaryKey,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
```

```
    std::cout << "Creating table " << tableName <<
              " with a simple primary key: \"" << primaryKey << "\"." <<
 std::endl;

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey;
    hashKey.SetAttributeName(primaryKey);
    hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey);

    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement;
    keySchemaElement.WithAttributeName(primaryKey).WithKeyType(
            Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(keySchemaElement);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::CreateTableOutcome &outcome =
 dynamoClient.CreateTable(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Table \""
                  << outcome.GetResult().GetTableDescription().GetTableName() <<
                  " created!" << std::endl;
    }
    else {
        std::cerr << "Failed to create table: " <<
 outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como criar uma tabela com tags

O exemplo de `create-table` a seguir usa os atributos especificados e o esquema de chaves para criar uma tabela chamada `MusicCollection`. Essa tabela usa taxa de transferência provisionada e é criptografada em repouso usando a CMK de propriedade padrão AWS . O comando também aplica uma tag à tabela, com uma chave `Owner` e valor de `blueTeam`.

```
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-
definitions AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S
 \
    --key-
schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
    --tags Key=Owner,Value=blueTeam
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 5,
```

```
            "ReadCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "TableName": "MusicCollection",
        "TableStatus": "CREATING",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "Artist"
            },
            {
                "KeyType": "RANGE",
                "AttributeName": "SongTitle"
            }
        ],
        "ItemCount": 0,
        "CreationDateTime": "2020-05-26T16:04:41.627000-07:00",
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    }
}
```

Para obter mais informações, consulte Operações básicas nas tabelas no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 2: Para criar uma tabela no On-Demand Modo

O exemplo a seguir cria uma tabela chamada `MusicCollection` usando o modo sob demanda, em vez do modo de throughput provisionado. Esse método é útil para tabelas com workloads imprevisíveis.

```
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-
definitions AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S
 \
    --key-
schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-27T11:44:10.807000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 0,
            "WriteCapacityUnits": 0
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "BillingModeSummary": {
            "BillingMode": "PAY_PER_REQUEST"
        }
    }
}
```

Para obter mais informações, consulte Operações básicas nas tabelas no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 3: como criar uma tabela e criptografá-la com uma CMK gerenciada pelo cliente

O exemplo a seguir cria uma tabela chamada `MusicCollection` e a criptografa usando uma CMK gerenciada pelo cliente.

```
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-
definitions AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S
 \
    --key-
schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=abcd1234-
abcd-1234-a123-ab1234a1b234
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-27T11:12:16.431000-07:00",
```

```
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "SSEDescription": {
            "Status": "ENABLED",
            "SSEType": "KMS",
            "KMSMasterKeyArn": "arn:aws:kms:us-west-2:123456789012:key/abcd1234-
abcd-1234-a123-ab1234a1b234"
        }
    }
}
```

Para obter mais informações, consulte Operações básicas nas tabelas no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 4: como criar uma tabela com um índice secundário local

O exemplo a seguir usa os atributos especificados e o esquema de chaves para criar uma tabela chamada `MusicCollection` com um índice secundário local chamado `AlbumTitleIndex`.

```
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-
definitions AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S
 \
    --key-
schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --local-secondary-indexes \
        "[
            {
                \"IndexName\": \"AlbumTitleIndex\",
                \"KeySchema\": [
                    {\"AttributeName\": \"Artist\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\": \"AlbumTitle\",\"KeyType\":\"RANGE\"}
```

```
                ],
                \"Projection\": {
                    \"ProjectionType\": \"INCLUDE\",
                    \"NonKeyAttributes\": [\"Genre\", \"Year\"]
                }
            }
        ]"
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
```

```
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "LocalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitleIndex",
                "KeySchema": [
                    {
                        "AttributeName": "Artist",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "INCLUDE",
                    "NonKeyAttributes": [
                        "Genre",
                        "Year"
                    ]
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
            }
        ]
    }
}
```

Para obter mais informações, consulte Operações básicas nas tabelas no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 5: como criar uma tabela com um índice secundário global

O exemplo a seguir cria uma tabela chamada `GameScores` com um índice secundário global chamado `GameTitleIndex`. A tabela-base tem uma chave de partição `UserId` e uma chave de classificação `GameTitle`, permitindo que você encontre a melhor pontuação de

um usuário individual para um jogo específico de forma eficiente, enquanto o GSI tem uma chave de partição `GameTitle` e uma chave de classificação `TopScore`, permitindo que você encontre rapidamente a pontuação mais alta geral para um jogo específico.

```
aws dynamodb create-table \
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-schema AttributeName=UserId,KeyType=HASH \
                 AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"GameTitleIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"GameTitle\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"TopScore\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {
                    \"ProjectionType\":\"INCLUDE\",
                    \"NonKeyAttributes\":[\"UserId\"]
                },
                \"ProvisionedThroughput\": {
                    \"ReadCapacityUnits\": 10,
                    \"WriteCapacityUnits\": 5
                }
            }
        ]"
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "TopScore",
                "AttributeType": "N"
```

```
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-26T17:28:15.602000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "GameTitleIndex",
                "KeySchema": [
                    {
                        "AttributeName": "GameTitle",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "TopScore",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "INCLUDE",
                    "NonKeyAttributes": [
```

```
                        "UserId"
                    ]
                },
                "IndexStatus": "CREATING",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 5
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/index/GameTitleIndex"
            }
        ]
    }
}
```

Para obter mais informações, consulte Operações básicas nas tabelas no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 6: como criar uma tabela com vários índices secundários globais ao mesmo tempo

O exemplo a seguir cria uma tabela chamada `GameScores` com dos índices secundários globais. Os esquemas do GSI são passados por meio de um arquivo, e não pela linha de comando.

```
aws dynamodb create-table \
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --global-secondary-indexes file://gsi.json
```

Conteúdo de `gsi.json`:

```
[
    {
        "IndexName": "GameTitleIndex",
```

```
        "KeySchema": [
            {
                "AttributeName": "GameTitle",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "TopScore",
                "KeyType": "RANGE"
            }
        ],
        "Projection": {
            "ProjectionType": "ALL"
        },
        "ProvisionedThroughput": {
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        }
    },
    {
        "IndexName": "GameDateIndex",
        "KeySchema": [
            {
                "AttributeName": "GameTitle",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "Date",
                "KeyType": "RANGE"
            }
        ],
        "Projection": {
            "ProjectionType": "ALL"
        },
        "ProvisionedThroughput": {
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        }
    }
]
```

Saída:

```
{
```

```
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Date",
                "AttributeType": "S"
            },
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "TopScore",
                "AttributeType": "N"
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-08-04T16:40:55.524000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "GameTitleIndex",
```

```
                "KeySchema": [
                    {
                        "AttributeName": "GameTitle",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "TopScore",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "CREATING",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 5
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/index/GameTitleIndex"
            },
            {
                "IndexName": "GameDateIndex",
                "KeySchema": [
                    {
                        "AttributeName": "GameTitle",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "Date",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "CREATING",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 5
```

```
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/index/GameDateIndex"
            }
        ]
    }
}
```

Para obter mais informações, consulte Operações básicas nas tabelas no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 7: como criar uma tabela com o Streams habilitado

O exemplo a seguir cria uma tabela chamada `GameScores` com o DynamoDB Streams habilitado. Imagens novas e antigas de cada item serão gravadas no fluxo.

```
aws dynamodb create-table \
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --stream-specification StreamEnabled=TRUE,StreamViewType=NEW_AND_OLD_IMAGES
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
```

```
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-27T10:49:34.056000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "StreamSpecification": {
            "StreamEnabled": true,
            "StreamViewType": "NEW_AND_OLD_IMAGES"
        },
        "LatestStreamLabel": "2020-05-27T17:49:34.056",
        "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/stream/2020-05-27T17:49:34.056"
    }
}
```

Para obter mais informações, consulte Operações básicas nas tabelas no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 8: Para criar uma tabela com o Keys-Only Stream ativado

O exemplo a seguir cria uma tabela chamada `GameScores` com o DynamoDB Streams habilitado. Somente os atributos-chave dos itens modificados são gravados no fluxo.

```
aws dynamodb create-table \
    --table-name GameScores \
```

```
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --stream-specification StreamEnabled=TRUE,StreamViewType=KEYS_ONLY
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2023-05-25T18:45:34.140000+00:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
```

```
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "StreamSpecification": {
            "StreamEnabled": true,
            "StreamViewType": "KEYS_ONLY"
        },
        "LatestStreamLabel": "2023-05-25T18:45:34.140",
        "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/stream/2023-05-25T18:45:34.140",
        "DeletionProtectionEnabled": false
    }
}
```

Para obter mais informações, consulte Captura de dados de alterações com o Amazon DynamoDB no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 9: como criar uma tabela com a classe Standard-Infrequent Access

O exemplo a seguir cria uma tabela chamada `GameScores` e atribui a classe de tabela Standard-Infrequent Access (Standard-IADynamoDB). Essa classe de tabela é otimizada para que o armazenamento seja o custo dominante.

```
aws dynamodb create-table \
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --table-class STANDARD_INFREQUENT_ACCESS
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "UserId",
```

```
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2023-05-25T18:33:07.581000+00:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "TableClassSummary": {
            "TableClass": "STANDARD_INFREQUENT_ACCESS"
        },
        "DeletionProtectionEnabled": false
    }
}
```

Para obter mais informações, consulte Classes de tabela no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 10: como criar uma tabela com a proteção contra exclusão habilitada

O exemplo a seguir cria uma tabela chamada `GameScores` e habilita a proteção contra exclusão.

```
aws dynamodb create-table \
    --table-name GameScores \
```

```
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --deletion-protection-enabled
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2023-05-25T23:02:17.093000+00:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
```

```
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "DeletionProtectionEnabled": true
    }
}
```

Para obter mais informações, consulte Usar a proteção contra exclusão no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte CreateTableem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

```
// CreateMovieTable creates a DynamoDB table with a composite primary key defined
 as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
 (*types.TableDescription, error) {
 var tableDesc *types.TableDescription
 table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
  AttributeDefinitions: []types.AttributeDefinition{{
   AttributeName: aws.String("year"),
   AttributeType: types.ScalarAttributeTypeN,
  }, {
   AttributeName: aws.String("title"),
   AttributeType: types.ScalarAttributeTypeS,
  }},
  KeySchema: []types.KeySchemaElement{{
   AttributeName: aws.String("year"),
   KeyType:       types.KeyTypeHash,
  }, {
   AttributeName: aws.String("title"),
   KeyType:       types.KeyTypeRange,
  }},
  TableName:   aws.String(basics.TableName),
  BillingMode: types.BillingModePayPerRequest,
 })
 if err != nil {
  log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
 } else {
  waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
  err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
   TableName: aws.String(basics.TableName)}, 5*time.Minute)
  if err != nil {
   log.Printf("Wait for table exists failed. Here's why: %v\n", err)
  }
  tableDesc = table.TableDescription
  log.Printf("Ccreating table test")
 }
 return tableDesc, err
}
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.BillingMode;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.OnDemandThroughput;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

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

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

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
Music3).
                key - The key for the Amazon DynamoDB table (for example,
Artist).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + "
with a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
            .attributeDefinitions(AttributeDefinition.builder()
                .attributeName(key)
                .attributeType(ScalarAttributeType.S)
                .build())
            .keySchema(KeySchemaElement.builder()
                .attributeName(key)
                .keyType(KeyType.HASH)
                .build())
```

```
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
 scales based on traffic.
            .tableName(tableName)
            .build();

        String newTable;
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            newTable = response.tableDescription().tableName();
            return newTable;

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import { CreateTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new CreateTableCommand({
    TableName: "EspressoDrinks",
    // For more information about data types,
    // see https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
HowItWorks.NamingRulesDataTypes.html#HowItWorks.DataTypes and
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
    AttributeDefinitions: [
      {
        AttributeName: "DrinkName",
        AttributeType: "S",
      },
    ],
    KeySchema: [
      {
        AttributeName: "DrinkName",
        KeyType: "HASH",
      },
    ],
    BillingMode: "PAY_PER_REQUEST",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  AttributeDefinitions: [
    {
      AttributeName: "CUSTOMER_ID",
      AttributeType: "N",
    },
    {
      AttributeName: "CUSTOMER_NAME",
      AttributeType: "S",
    },
  ],
  KeySchema: [
    {
      AttributeName: "CUSTOMER_ID",
      KeyType: "HASH",
    },
    {
      AttributeName: "CUSTOMER_NAME",
      KeyType: "RANGE",
    },
  ],
  ProvisionedThroughput: {
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 1,
  },
  TableName: "CUSTOMER_LIST",
```

```
  StreamSpecification: {
    StreamEnabled: false,
  },
};

// Call DynamoDB to create the table
ddb.createTable(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Table Created", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewTable(
    tableNameVal: String,
    key: String,
): String? {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
```

```
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

- Para obter detalhes da API, consulte a CreateTablereferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Crie uma tabela.

```
        $tableName = "ddb_demo_table_$uuid";
```

```
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

    public function createTable(string $tableName, array $attributes)
    {
        $keySchema = [];
        $attributeDefinitions = [];
        foreach ($attributes as $attribute) {
            if (is_a($attribute, DynamoDBAttribute::class)) {
                $keySchema[] = ['AttributeName' => $attribute->AttributeName,
'KeyType' => $attribute->KeyType];
                $attributeDefinitions[] =
                    ['AttributeName' => $attribute->AttributeName,
'AttributeType' => $attribute->AttributeType];
            }
        }

        $this->dynamoDbClient->createTable([
            'TableName' => $tableName,
            'KeySchema' => $keySchema,
            'AttributeDefinitions' => $attributeDefinitions,
            'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,
'WriteCapacityUnits' => 10],
        ]);
    }
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave). O esquema usado para construir a tabela pode ser canalizado para cada cmdlet conforme mostrado ou especificado usando o parâmetro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {}
```

Exemplo 2: Este exemplo cria uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave). Um índice secundário local também é definido. A chave do índice secundário local será definida automaticamente a partir da chave de hash primária na tabela (ForumName). O esquema usado para construir a tabela pode ser canalizado para cada cmdlet conforme mostrado ou especificado usando o parâmetro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
```

```
LocalSecondaryIndexes  : {LastPostIndex}
```

Exemplo 3: Este exemplo mostra como usar um único pipeline para criar uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave) e um índice secundário local. O Add-DDBKeySchema e Add-DDBIndexSchema cria um novo TableSchema objeto para você se um não for fornecido pelo pipeline ou pelo parâmetro -Schema.

```
New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
                     -RangeKeyName "LastPostDateTime" `
                     -RangeKeyDataType "S" `
                     -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

- Para obter detalhes da API, consulte CreateTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave). O esquema usado para construir a tabela pode ser canalizado para cada cmdlet conforme mostrado ou especificado usando o parâmetro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
```

```
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {}
```

Exemplo 2: Este exemplo cria uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave). Um índice secundário local também é definido. A chave do índice secundário local será definida automaticamente a partir da chave de hash primária na tabela (ForumName). O esquema usado para construir a tabela pode ser canalizado para cada cmdlet conforme mostrado ou especificado usando o parâmetro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

Exemplo 3: Este exemplo mostra como usar um único pipeline para criar uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Subject' (intervalo de tipos de chave) e um índice secundário local. O Add-DDBKeySchema e Add-DDBIndexSchema cria um novo TableSchema objeto para você se um não for fornecido pelo pipeline ou pelo parâmetro -Schema.

```
New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
                     -RangeKeyName "LastPostDateTime" `
                     -RangeKeyDataType "S" `
                     -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

- Para obter detalhes da API, consulte CreateTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Crie uma tabela para armazenar dados de filmes.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def create_table(self, table_name):
        """
        Creates an Amazon DynamoDB table that can be used to store movie data.
        The table uses the release year of the movie as the partition key and the
        title as the sort key.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
```

```
        try:
            self.table = self.dyn_resource.create_table(
                TableName=table_name,
                KeySchema=[
                    {"AttributeName": "year", "KeyType": "HASH"},  # Partition
key
                    {"AttributeName": "title", "KeyType": "RANGE"},  # Sort key
                ],
                AttributeDefinitions=[
                    {"AttributeName": "year", "AttributeType": "N"},
                    {"AttributeName": "title", "AttributeType": "S"},
                ],
                BillingMode='PAY_PER_REQUEST',
            )
            self.table.wait_until_exists()
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.table
```

- Para obter detalhes da API, consulte a CreateTableReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        { attribute_name: 'year', key_type: 'HASH' }, # Partition key
        { attribute_name: 'title', key_type: 'RANGE' } # Sort key
      ],
      attribute_definitions: [
        { attribute_name: 'year', attribute_type: 'N' },
        { attribute_name: 'title', attribute_type: 'S' }
      ],
      billing_mode: 'PAY_PER_REQUEST'
    )
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_table(
    client: &Client,
    table: &str,
    key: &str,
) -> Result<CreateTableOutput, Error> {
    let a_name: String = key.into();
    let table_name: String = table.into();

    let ad = AttributeDefinition::builder()
        .attribute_name(&a_name)
        .attribute_type(ScalarAttributeType::S)
        .build()
        .map_err(Error::BuildError)?;

    let ks = KeySchemaElement::builder()
        .attribute_name(&a_name)
        .key_type(KeyType::Hash)
        .build()
        .map_err(Error::BuildError)?;

    let create_table_response = client
        .create_table()
        .table_name(table_name)
        .key_schema(ks)
        .attribute_definitions(ad)
        .billing_mode(BillingMode::PayPerRequest)
        .send()
        .await;

    match create_table_response {
        Ok(out) => {
            println!("Added table {} with key {}", table, key);
```

```
            Ok(out)
        }
        Err(e) => {
            eprintln!("Got an error creating table:");
            eprintln!("{}", e);
            Err(Error::unhandled(e))
        }
    }
}
```

- Para obter detalhes da API, consulte a CreateTablereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                              iv_keytype = 'HASH' ) )
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                              iv_keytype = 'RANGE' ) ) ).
        DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                           iv_attributetype = 'N' ) )
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                           iv_attributetype = 'S' ) ) ).

        " Adjust read/write capacities as desired.
        DATA(lo_dynprovthroughput)  = NEW /aws1/cl_dynprovthroughput(
          iv_readcapacityunits = 5
          iv_writecapacityunits = 5 ).
        oo_result = lo_dyn->createtable(
          it_keyschema = lt_keyschema
```

```
          iv_tablename = iv_table_name
          it_attributedefinitions = lt_attributedefinitions
          io_provisionedthroughput = lo_dynprovthroughput ).
        " Table creation can take some time. Wait till table exists before
returning.
        lo_dyn->get_waiter( )->tableexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
        " This exception can happen if the table already exists.
      CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
{ lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTablereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    ///
    /// Create a movie table in the Amazon DynamoDB data store.
    ///
    private func createTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }
```

```
            let input = CreateTableInput(
                attributeDefinitions: [
                    DynamoDBClientTypes.AttributeDefinition(attributeName:
"year", attributeType: .n),
                    DynamoDBClientTypes.AttributeDefinition(attributeName:
"title", attributeType: .s)
                ],
                billingMode: DynamoDBClientTypes.BillingMode.payPerRequest,
                keySchema: [
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "year",
keyType: .hash),
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "title",
keyType: .range)
                ],
                tableName: self.tableName
            )
            let output = try await client.createTable(input: input)
            if output.tableDescription == nil {
                throw MoviesError.TableNotFound
            }
        } catch {
            print("ERROR: createTable:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateTablea referência da API AWS SDK for Swift.

Use **`DeleteItem`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteItem`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Usar operações condicionais

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes a single item from a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table from which the item
    /// will be deleted.</param>
    /// <param name="movieToDelete">A movie object containing the title and
    /// year of the movie to delete.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// delete operation.</returns>
    public async Task<bool> DeleteItemAsync(
        string tableName,
        Movie movieToDelete)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = movieToDelete.Title },
                ["year"] = new AttributeValue { N =
movieToDelete.Year.ToString() },
            };

            var request = new DeleteItemRequest { TableName = tableName, Key =
key, };

            await _amazonDynamoDB.DeleteItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
```

```
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting item.
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DeleteItema Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to delete.
#
#  Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function dynamodb_delete_item() {
  local table_name keys response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_delete_item"
    echo "Delete an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to delete."
    echo ""
  }
  while getopts "n:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
```

```
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho ""

  response=$(aws dynamodb delete-item \
    --table-name "$table_name" \
    --key file://"$keys")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
  fi

  return 0

}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
```

```
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteItemem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an item from an Amazon DynamoDB table.
/*!
  \sa deleteItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::deleteItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteItemRequest request;

    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteItemOutcome &outcome =
 dynamoClient.DeleteItem(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Item \"" << partitionValue << "\" deleted!" << std::endl;
    }
    else {
        std::cerr << "Failed to delete item: " << outcome.GetError().GetMessage()
                  << std::endl;
```

```
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
```

```
                        << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte DeleteItema Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como excluir um item

O exemplo de `delete-item` a seguir exclui um item da tabela `MusicCollection` e solicita detalhes sobre o item excluído e a capacidade usada pela solicitação.

```
aws dynamodb delete-item \
    --table-name MusicCollection \
    --key file://key.json \
    --return-values ALL_OLD \
    --return-consumed-capacity TOTAL \
    --return-item-collection-metrics SIZE
```

Conteúdo de `key.json`:

```
{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Scared of My Shadow"}
}
```

Saída:

```
{
    "Attributes": {
        "AlbumTitle": {
            "S": "Blue Sky Blues"
        },
        "Artist": {
```

```
            "S": "No One You Know"
        },
        "SongTitle": {
            "S": "Scared of My Shadow"
        }
    },
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 2.0
    },
    "ItemCollectionMetrics": {
        "ItemCollectionKey": {
            "Artist": {
                "S": "No One You Know"
            }
        },
        "SizeEstimateRangeGB": [
            0.0,
            1.0
        ]
    }
}
```

Para obter mais informações, consulte Gravar um item no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 2: como excluir um item de forma condicional

O exemplo a seguir exclui um item da tabela `ProductCatalog` somente se a `ProductCategory` for `Sporting Goods` ou `Gardening Supplies` e o preço estiver entre 500 e 600. Ele retorna detalhes sobre o item que foi excluído.

```
aws dynamodb delete-item \
    --table-name ProductCatalog \
    --key '{"Id":{"N":"456"}}' \
    --condition-expression "(ProductCategory IN (:cat1, :cat2)) and (#P
 between :lo and :hi)" \
    --expression-attribute-names file://names.json \
    --expression-attribute-values file://values.json \
    --return-values ALL_OLD
```

Conteúdo de `names.json`:

```
{
    "#P": "Price"
}
```

Conteúdo de `values.json`:

```
{
    ":cat1": {"S": "Sporting Goods"},
    ":cat2": {"S": "Gardening Supplies"},
    ":lo": {"N": "500"},
    ":hi": {"N": "600"}
}
```

Saída:

```
{
    "Attributes": {
        "Id": {
            "N": "456"
        },
        "Price": {
            "N": "550"
        },
        "ProductCategory": {
            "S": "Sporting Goods"
        }
    }
}
```

Para obter mais informações, consulte Gravar um item no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte DeleteItemem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
 _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
  TableName: aws.String(basics.TableName), Key: movie.GetKey(),
 })
 if err != nil {
```

```
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
```

```
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte DeleteItema Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

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

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

                Usage:
                    <tableName> <key> <keyval>

                Where:
                    tableName - The Amazon DynamoDB table to delete the item from
 (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
                    keyval - The key value that represents the item to delete
 (for example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
 String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
                .tableName(tableName)
```

```
                .key(keyToGet)
                .build();

        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte DeleteCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, DeleteCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new DeleteCommand({
    TableName: "Sodas",
    Key: {
      Flavor: "Cola",
    },
  });
```

```
  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteItema Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Exclua um item de uma tabela.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Key: {
    KEY_NAME: { N: "VALUE" },
  },
};

// Call DynamoDB to delete the item from the table
ddb.deleteItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
```

```
  }
});
```

Exclua um item de uma tabela usando o cliente de documento do DynamoDB.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  Key: {
    HASH_KEY: VALUE,
  },
  TableName: "TABLE",
};

docClient.delete(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteItema Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDynamoDBItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        DeleteItemRequest {
            tableName = tableNameVal
            key = keyToGet
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- Para obter detalhes da API, consulte a DeleteItemreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

    public function deleteItemByKey(string $tableName, array $key)
    {
        $this->dynamoDbClient->deleteItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteItema Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: remove o item do DynamoDB que corresponde à chave fornecida.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Para obter detalhes da API, consulte DeleteItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: remove o item do DynamoDB que corresponde à chave fornecida.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Para obter detalhes da API, consulte DeleteItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
```

```
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

É possível especificar uma condição para que um item seja excluído somente quando ele atender a determinados critérios.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def delete_underrated_movie(self, title, year, rating):
        """
        Deletes a movie only if it is rated below a specified value. By using a
        condition expression in a delete operation, you can specify that an item
 is
        deleted only when it meets certain criteria.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        :param rating: The rating threshold to check before deleting the movie.
        """
        try:
            self.table.delete_item(
                Key={"year": year, "title": title},
                ConditionExpression="info.rating <= :val",
                ExpressionAttributeValues={":val": Decimal(str(rating))},
            )
        except ClientError as err:
            if err.response["Error"]["Code"] ==
 "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't delete %s because its rating is greater than %s.",
                    title,
                    rating,
                )
            else:
                logger.error(
                    "Couldn't delete movie %s. Here's why: %s: %s",
                    title,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteItemReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
  # @param year [Integer] The release year of the movie to delete.
  def delete_item(title, year)
    @table.delete_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteItema Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_item(
    client: &Client,
    table: &str,
    key: &str,
    value: &str,
) -> Result<DeleteItemOutput, Error> {
    match client
        .delete_item()
        .table_name(table)
        .key(key, AttributeValue::S(value.into()))
        .send()
        .await
    {
        Ok(out) => {
            println!("Deleted item from table");
            Ok(out)
        }
        Err(e) => Err(Error::unhandled(e)),
    }
}
```

- Para obter detalhes da API, consulte a DeleteItemreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_resp) = lo_dyn->deleteitem(
          iv_tablename                = iv_table_name
          it_key                      = it_key_input ).
        MESSAGE 'Deleted one item.' TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteItemreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB
```

```
    /// Delete a movie, given its title and release year.
    ///
    /// - Parameters:
    ///   - title: The movie's title.
    ///   - year: The movie's release year.
    ///
    func delete(title: String, year: Int) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            _ = try await client.deleteItem(input: input)
        } catch {
            print("ERROR: delete:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteItema referência da API AWS SDK for Swift.

Use **`DeleteTable`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Acelerar leituras com o DAX
- Criar e gerenciar tabelas globais com MRSC
- Conceitos básicos de bancos de dados NoSQL

- Gerenciar índices secundários globais
- Trabalhar com tabelas globais e a consistência final de replicação mutirregional (MREC)

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> DeleteTableAsync(string tableName)
    {
        try
        {
            var request = new DeleteTableRequest
            {
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.DeleteTableAsync(request);

            Console.WriteLine($"Table {response.TableDescription.TableName}
 successfully deleted.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found and cannot be
 deleted. {ex.Message}");
            return false;
        }
```

```
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
 table {tableName}. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting table
 {tableName}. {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_delete_table() {
  local table_name response
```

```
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function dynamodb_delete_table"
    echo "Deletes an Amazon DynamoDB table."
    echo " -n table_name  -- The name of the table to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho ""

  response=$(aws dynamodb delete-table \
    --table-name "$table_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports delete-table operation failed.$response"
    return 1
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
```

```
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteTableem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon DynamoDB table.
/*!
  \sa deleteTable()
  \param tableName: The DynamoDB table name.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteTable(const Aws::String &tableName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir uma tabela

O exemplo de `delete-table` a seguir exclui a tabela `MusicCollection`.

```
aws dynamodb delete-table \
    --table-name MusicCollection
```

Saída:

```
{
    "TableDescription": {
        "TableStatus": "DELETING",
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableName": "MusicCollection",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 5,
            "ReadCapacityUnits": 5
        }
    }
}
```

Para obter mais informações, consulte Excluir uma tabela no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte DeleteTableem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
```

```
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
 _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
  TableName: aws.String(basics.TableName)})
 if err != nil {
  log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
```

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

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

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to delete (for example,
 Music3).

                **Warning** This program will delete the table that you specify!
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Deleting the Amazon DynamoDB table %s...\n",
 tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
 {
        DeleteTableRequest request = DeleteTableRequest.builder()
```

```
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
}
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new DeleteTableCommand({
    TableName: "DecafCoffees",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: process.argv[2],
};

// Call DynamoDB to delete the specified table
ddb.deleteTable(params, function (err, data) {
  if (err && err.code === "ResourceNotFoundException") {
    console.log("Error: Table not found");
  } else if (err && err.code === "ResourceInUseException") {
    console.log("Error: Table in use");
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTablereferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function deleteTable(string $TableName)
    {
        $this->customWaiter(function () use ($TableName) {
            return $this->dynamoDbClient->deleteTable([
                'TableName' => $TableName,
```

```
            ]);
        });
    }
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui a tabela especificada. A confirmação será solicitada antes que a operação continue.

```
Remove-DDBTable -TableName "myTable"
```

Exemplo 2: exclui a tabela especificada. A confirmação não será solicitada antes que a operação continue.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Para obter detalhes da API, consulte DeleteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui a tabela especificada. A confirmação será solicitada antes que a operação continue.

```
Remove-DDBTable -TableName "myTable"
```

Exemplo 2: exclui a tabela especificada. A confirmação não será solicitada antes que a operação continue.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Para obter detalhes da API, consulte DeleteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTableReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
```

```
    @logger.level = Logger::DEBUG
  end

  # Deletes the table.
  def delete_table
    @table.delete
    @table = nil
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete table. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_table(client: &Client, table: &str) ->
 Result<DeleteTableOutput, Error> {
    let resp = client.delete_table().table_name(table).send().await;

    match resp {
        Ok(out) => {
            println!("Deleted table");
            Ok(out)
        }
        Err(e) => Err(Error::Unhandled(e.into())),
    }
}
```

- Para obter detalhes da API, consulte a DeleteTablereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_dyn->deletetable( iv_tablename = iv_table_name ).
        " Wait till the table is actually deleted.
        lo_dyn->get_waiter( )->tablenotexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'Table ' && iv_table_name && ' deleted.' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table ' && iv_table_name && ' does not exist' TYPE 'E'.
      CATCH /aws1/cx_dynresourceinuseex.
        MESSAGE 'The table cannot be deleted since it is in use' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTablereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB
```

```
    ///
    /// Deletes the table from Amazon DynamoDB.
    ///
    func deleteTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteTableInput(
                tableName: self.tableName
            )
            _ = try await client.deleteTable(input: input)
        } catch {
            print("ERROR: deleteTable:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteTablea referência da API AWS SDK for Swift.

Use **`DescribeTable`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Crie e gerencie tabelas globais demonstrando o MREC
- Criar e gerenciar tabelas globais com MRSC
- Conceitos básicos de bancos de dados NoSQL
- Trabalhar com tabelas globais e a consistência final de replicação mutirregional (MREC)
- Trabalhe com Streams e Time-to-Live
- Trabalhar com a criptografia de tabelas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
private static async Task GetTableInformation()
{
    Console.WriteLine("\n*** Retrieving table information ***");

    var response = await Client.DescribeTableAsync(new DescribeTableRequest
    {
        TableName = ExampleTableName
    });

    var table = response.Table;
    Console.WriteLine($"Name: {table.TableName}");
    Console.WriteLine($"# of items: {table.ItemCount}");

}
```

- Para obter detalhes da API, consulte DescribeTablea Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
```

```
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#
#  Response:
#       - TableStatus:
#     And:
#       0 - Table is active.
#       1 - If it fails.
##############################################################################
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_describe_table"
    echo "Describe the status of a DynamoDB table."
    echo "  -n table_name  -- The name of the table."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  local table_status
    table_status=$(
      aws dynamodb describe-table \
        --table-name "$table_name" \
        --output text \
        --query 'Table.TableStatus'
    )

   local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log "$error_code"
      errecho "ERROR: AWS reports describe-table operation failed.$table_status"
      return 1
    fi

  echo "$table_status"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
```

```
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeTableem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe an Amazon DynamoDB table.
/*!
  \sa describeTable()
  \param tableName: The DynamoDB table name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::describeTable(const Aws::String &tableName,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DescribeTableOutcome &outcome =
 dynamoClient.DescribeTable(
            request);

    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::TableDescription &td =
 outcome.GetResult().GetTable();
        std::cout << "Table name  : " << td.GetTableName() << std::endl;
        std::cout << "Table ARN   : " << td.GetTableArn() << std::endl;
        std::cout << "Status      : "
                  <<
 Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(
                          td.GetTableStatus()) << std::endl;
        std::cout << "Item count  : " << td.GetItemCount() << std::endl;
        std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;
```

```
        const Aws::DynamoDB::Model::ProvisionedThroughputDescription &ptd =
 td.GetProvisionedThroughput();
        std::cout << "Throughput" << std::endl;
        std::cout << "  Read Capacity : " << ptd.GetReadCapacityUnits() <<
 std::endl;
        std::cout << "  Write Capacity: " << ptd.GetWriteCapacityUnits() <<
 std::endl;

        const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition> &ad =
 td.GetAttributeDefinitions();
        std::cout << "Attributes" << std::endl;
        for (const auto &a: ad)
            std::cout << "  " << a.GetAttributeName() << " (" <<

 Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(
                              a.GetAttributeType()) <<
                      ")" << std::endl;
    }
    else {
        std::cerr << "Failed to describe table: " <<
 outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeTablea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como descrever uma tabela

O exemplo a seguir de `describe-table` descreve a tabela `MusicCollection`.

```
aws dynamodb describe-table \
    --table-name MusicCollection
```

Saída:

```
{
    "Table": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 5,
            "ReadCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "TableName": "MusicCollection",
        "TableStatus": "ACTIVE",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "Artist"
            },
            {
                "KeyType": "RANGE",
                "AttributeName": "SongTitle"
            }
        ],
        "ItemCount": 0,
        "CreationDateTime": 1421866952.062
    }
}
```

Para obter mais informações, consulte Descrever uma tabela no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte DescribeTableem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
 exists := true
 _, err := basics.DynamoDbClient.DescribeTable(
  ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
 )
 if err != nil {
```

```
  var notFoundEx *types.ResourceNotFoundException
  if errors.As(err, &notFoundEx) {
   log.Printf("Table %v does not exist.\n", basics.TableName)
   err = nil
  } else {
   log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
 basics.TableName, err)
  }
  exists = false
 }
 return exists, err
}
```

- Para obter detalhes da API, consulte DescribeTablea Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
 software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information
 about (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Getting description for %s\n\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        describeDymamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDymamoDBTable(DynamoDbClient ddb, String
 tableName) {
        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
                System.out.format("Table name  : %s\n", tableInfo.tableName());
                System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
                System.out.format("Status      : %s\n", tableInfo.tableStatus());
```

```
                System.out.format("Item count  : %d\n", tableInfo.itemCount());
                System.out.format("Size (bytes): %d\n",
 tableInfo.tableSizeBytes());

                ProvisionedThroughputDescription throughputInfo =
 tableInfo.provisionedThroughput();
                System.out.println("Throughput");
                System.out.format("  Read Capacity : %d\n",
 throughputInfo.readCapacityUnits());
                System.out.format("  Write Capacity: %d\n",
 throughputInfo.writeCapacityUnits());

                List<AttributeDefinition> attributes =
 tableInfo.attributeDefinitions();
                System.out.println("Attributes");
                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n", a.attributeName(),
 a.attributeType());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
}
```

- Para obter detalhes da API, consulte DescribeTablea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import { DescribeTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new DescribeTableCommand({
    TableName: "Pastries",
  });

  const response = await client.send(command);
  console.log(`TABLE NAME: ${response.Table.TableName}`);
  console.log(`TABLE ITEM COUNT: ${response.Table.ItemCount}`);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeTablea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: process.argv[2],
};
```

```
// Call DynamoDB to retrieve the selected table descriptions
ddb.describeTable(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Table.KeySchema);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeTablea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exibe detalhes da tabela especificada.

```
Get-DDBTable -TableName "myTable"
```

- Para obter detalhes da API, consulte DescribeTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exibe detalhes da tabela especificada.

```
Get-DDBTable -TableName "myTable"
```

- Para obter detalhes da API, consulte DescribeTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            table.load()
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                exists = False
            else:
                logger.error(
                    "Couldn't check for existence of %s. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            self.table = table
        return exists
```

- Para obter detalhes da API, consulte a DescribeTableReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DescribeTablea Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_dyn->describetable( iv_tablename = iv_table_name ).
    DATA(lv_tablename) = oo_result->get_table( )->ask_tablename( ).
    DATA(lv_tablearn) = oo_result->get_table( )->ask_tablearn( ).
    DATA(lv_tablestatus) = oo_result->get_table( )->ask_tablestatus( ).
    DATA(lv_itemcount) = oo_result->get_table( )->ask_itemcount( ).
    MESSAGE 'The table name is ' && lv_tablename
        && '. The table ARN is ' && lv_tablearn
        && '. The tablestatus is ' && lv_tablestatus
        && '. Item count is ' && lv_itemcount TYPE 'I'.
  CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table ' && lv_tablename && ' does not exist' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeTablereferência da API AWS SDK for SAP ABAP.

Use **`DescribeTimeToLive`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeTimeToLive`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Trabalhe com Streams e Time-to-Live

CLI

AWS CLI

Como ver as configurações de vida útil de uma tabela

O exemplo `describe-time-to-live` a seguir exibe as configurações de vida útil da tabela `MusicCollection`.

```
aws dynamodb describe-time-to-live \
    --table-name MusicCollection
```

Saída:

```
{
    "TimeToLiveDescription": {
        "TimeToLiveStatus": "ENABLED",
        "AttributeName": "ttl"
    }
}
```

Para obter mais informações, consulte Vida útil no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte DescribeTimeToLiveem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

Descreva a configuração de TTL em uma tabela existente do DynamoDB usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;

    public DescribeTimeToLiveResponse describeTTL(final String tableName, final
 Region region) {
        final DescribeTimeToLiveRequest request =
            DescribeTimeToLiveRequest.builder().tableName(tableName).build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.describeTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DescribeTimeToLivea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Descreva a configuração de TTL em uma tabela existente do DynamoDB usando o AWS SDK para JavaScript.

```
import { DynamoDBClient, DescribeTimeToLiveCommand } from "@aws-sdk/client-
dynamodb";

export const describeTTL = async (tableName, region) => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    try {
```

```
        const ttlDescription = await client.send(new
 DescribeTimeToLiveCommand({ TableName: tableName }));

        if (ttlDescription.TimeToLiveDescription.TimeToLiveStatus === 'ENABLED')
 {
            console.log("TTL is enabled for table %s.", tableName);
        } else {
            console.log("TTL is not enabled for table %s.", tableName);
        }

        return ttlDescription;
    } catch (e) {
        console.error(`Error describing table: ${e}`);
        throw e;
    }
}

// Example usage (commented out for testing)
// describeTTL('your-table-name', 'us-east-1');
```

- Para obter detalhes da API, consulte DescribeTimeToLivea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Descreva a configuração de TTL em uma tabela existente do DynamoDB usando o AWS SDK para Python (Boto3).

```
import boto3


def describe_ttl(table_name, region):
    """
    Describes TTL on an existing table, as well as a region.

    :param table_name: String representing the name of the table
    :param region: AWS Region of the table - example `us-east-1`
    :return: Time to live description.
    """
    try:
```

```
        dynamodb = boto3.resource("dynamodb", region_name=region)
        ttl_description = dynamodb.describe_time_to_live(TableName=table_name)
        print(
            f"TimeToLive for table {table_name} is status
 {ttl_description['TimeToLiveDescription']['TimeToLiveStatus']}"
        )

        return ttl_description
    except Exception as e:
        print(f"Error describing table: {e}")
        raise


# Enter your own table name and AWS region
describe_ttl("your-table-name", "us-east-1")
```

- Para obter detalhes da API, consulte a DescribeTimeToLiveReferência da API AWS SDK for Python (Boto3).

Use **`ExecuteStatement`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ExecuteStatement`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Excluir dados usando DELETE do PartiQL
- Inserir dados usando INSERT do PartiQL
- Consultar uma tabela usando o PartiQL
- Consultar dados usando SELECT do PartiQL
- Atualizar dados usando UPDATE do PartiQL

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use uma instrução INSERT para adicionar um item.

```
        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Use uma instrução SELECT para obter um item.

```
        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }
```

Use uma instrução SELECT para obter uma lista de itens.

```
        /// <summary>
        /// Retrieve multiple movies by year using a SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="year">The year the movies were released.</param>
        /// <returns></returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
```

```
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { N = year.ToString() },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }
```

Use uma instrução UPDATE para atualizar um item.

```
        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
```

```
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Use uma instrução DELETE para excluir um único filme.

```
        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use uma instrução INSERT para adicionar um item.

```
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate
it? ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
```

```
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

Use uma instrução SELECT para obter um item.

```
    //  3. Get the data for the movie using a "Select" statement.
(ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());
```

```
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            // Print the retrieved movie information.
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were
retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }
```

Use uma instrução UPDATE para atualizar um item.

```
    //  4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);
```

```
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update a movie: "
                      << outcome.GetError().GetMessage();
            return false;
        }
    }
```

Use uma instrução DELETE para excluir um item.

```
    // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
```

```
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
 dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina uma estrutura de receptor de função para o exemplo.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
```

```
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

Use uma instrução INSERT para adicionar um item.

```
// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
 }
 return err
}
```

Use uma instrução SELECT para obter um item.

```
// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB
 table by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
```

```
 var movie Movie
 params, err := attributevalue.MarshalList([]interface{}{title, year})
 if err != nil {
  panic(err)
 }
 response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Items[0], &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}
```

Use uma instrução SELECT para obter uma lista de itens e projetar os resultados.

```
// GetAllMovies runs a PartiQL SELECT statement to get all movies from the
 DynamoDB table.
// pageSize is not typically required and is used to show how to paginate the
 results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
 ([]map[string]interface{}, error) {
 var output []map[string]interface{}
 var response *dynamodb.ExecuteStatementOutput
 var err error
 var nextToken *string
 for moreData := true; moreData; {
  response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
   Statement: aws.String(
```

```
    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
   Limit:     aws.Int32(pageSize),
   NextToken: nextToken,
  })
  if err != nil {
   log.Printf("Couldn't get movies. Here's why: %v\n", err)
   moreData = false
  } else {
   var pageOutput []map[string]interface{}
   err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    log.Printf("Got a page of length %v.\n", len(response.Items))
    output = append(output, pageOutput...)
   }
   nextToken = response.NextToken
   moreData = nextToken != nil
  }
 }
 return output, err
}
```

Use uma instrução UPDATE para atualizar um item.

```
// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie
 that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
 float64) error {
 params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
    runner.TableName)),
```

```
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
 }
 return err
}
```

Use uma instrução DELETE para excluir um item.

```
// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the
 DynamoDB table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
```

```
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um item usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: `INSERT INTO Flowers value {'Name':?}`,
    Parameters: ["Rose"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Obtenha um item usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
```

```
import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "SELECT * FROM CloudTypes WHERE IsStorm=?",
    Parameters: [false],
    ConsistentRead: true,
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Atualize um item usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "UPDATE EyeColors SET IsRecessive=? where Color=?",
    Parameters: [true, "blue"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Exclua um item usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "DELETE FROM PaintColors where Name=?",
    Parameters: ["Purple"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function insertItemByPartiQL(string $statement, array $parameters)
```

```
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because
 the
        resource transforms input and output from plain old Python objects
 (POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                       statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params
            )
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not
 exist.",
                    statement,
                )
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

- Para obter detalhes da API, consulte a ExecuteStatementReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Selecione um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
```

```
  end

  # Gets a single record from a table using PartiQL.
  # Note: To perform more fine-grained selects,
  # use the Client.query instance method instead.
  #
  # @param title [String] The title of the movie to search.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def select_item_by_title(title)
    request = {
      statement: "SELECT * FROM \"#{@table.name}\" WHERE title=?",
      parameters: [title]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Atualize um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Updates a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def update_rating_by_title(title, year, rating)
    request = {
      statement: "UPDATE \"#{@table.name}\" SET info.rating=? WHERE title=? and
 year=?",
      parameters: [{ "N": rating }, title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Adicione um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Adds a single record to a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param plot [String] The plot of the movie.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def insert_item(title, year, plot, rating)
    request = {
      statement: "INSERT INTO \"#{@table.name}\" VALUE {'title': ?, 'year': ?,
 'info': ?}",
      parameters: [title, year, { 'plot': plot, 'rating': rating }]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Exclua um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
```

```
# @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
def delete_item_by_title(title, year)
  request = {
    statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
    parameters: [title, year]
  }
  @dynamodb.client.execute_statement(request)
end
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para Ruby da API.

Use **`GetItem`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetItem`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Acelerar leituras com o DAX
- Criar e gerenciar tabelas globais com MRSC
- Conceitos básicos de bancos de dados NoSQL
- Trabalhar com tabelas globais e a consistência final de replicação mutirregional (MREC)

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Gets information about an existing movie from the table.
```

```
    /// </summary>
    /// <param name="newMovie">A Movie object containing information about
    /// the movie to retrieve.</param>
    /// <param name="tableName">The name of the table containing the movie.</
param>
    /// <returns>A Dictionary object containing information about the item
    /// retrieved.</returns>
    public async Task<Dictionary<string, AttributeValue>> GetItemAsync(Movie
 newMovie, string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new GetItemRequest
            {
                Key = key,
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.GetItemAsync(request);
            return response.Item;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return new Dictionary<string, AttributeValue>();
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while getting
 item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while getting item.
 {ex.Message}");
            throw;
        }
```

```
    }
```

- Para obter detalhes da API, consulte GetItema Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to get.
#       [-q query]  -- Optional JMESPath query expression.
#
#  Returns:
#       The item as text output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_get_item() {
  local table_name keys query response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_get_item"
```

```
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
item to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
  }
  query=""
  while getopts "n:k:q:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi

  if [[ -n "$query" ]]; then
    response=$(aws dynamodb get-item \
      --table-name "$table_name" \
      --key file://"$keys" \
      --output text \
      --query "$query")
```

```
  else
    response=$(
      aws dynamodb get-item \
        --table-name "$table_name" \
        --key file://"$keys" \
        --output text
    )
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports get-item operation failed.$response"
    return 1
  fi

  if [[ -n "$query" ]]; then
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
 query inserts on some strings.
  else
    echo "$response"
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
```

```
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte GetItemem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get an item from an Amazon DynamoDB table.
/*!
  \sa getItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::getItem(const Aws::String &tableName,
                               const Aws::String &partitionKey,
                               const Aws::String &partitionValue,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::GetItemRequest request;

    // Set up the request.
    request.SetTableName(tableName);
    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));

    // Retrieve the item's fields and values.
    const Aws::DynamoDB::Model::GetItemOutcome &outcome =
 dynamoClient.GetItem(request);
    if (outcome.IsSuccess()) {
        // Reference the retrieved fields/values.
        const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item =
 outcome.GetResult().GetItem();
        if (!item.empty()) {
            // Output each retrieved field and its value.
```

```
            for (const auto &i: item)
                std::cout << "Values: " << i.first << ": " << i.second.GetS()
                          << std::endl;
        }
        else {
            std::cout << "No item found with the key " << partitionKey <<
 std::endl;
        }
    }
    else {
        std::cerr << "Failed to get item: " << outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetItema Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como ler um item em uma tabela

O exemplo de `get-item` a seguir recupera um item da tabela `MusicCollection`. A tabela tem uma chave primária de hash e intervalo (`Artist` e `SongTitle`), portanto, você deve especificar esses dois atributos. O comando também solicita informações sobre a capacidade de leitura consumida pela operação.

```
aws dynamodb get-item \
    --table-name MusicCollection \
    --key file://key.json \
    --return-consumed-capacity TOTAL
```

Conteúdo de `key.json`:

```
{
    "Artist": {"S": "Acme Band"},
    "SongTitle": {"S": "Happy Day"}
}
```

Saída:

```
{
    "Item": {
        "AlbumTitle": {
            "S": "Songs About Life"
        },
        "SongTitle": {
            "S": "Happy Day"
        },
        "Artist": {
            "S": "Acme Band"
        }
    },
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 0.5
    }
}
```

Para obter mais informações, consulte Ler um item no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 2: como ler um item usando uma leitura consistente

O exemplo a seguir recupera um item da tabela `MusicCollection` usando leituras altamente consistentes.

```
aws dynamodb get-item \
    --table-name MusicCollection \
    --key file://key.json \
    --consistent-read \
    --return-consumed-capacity TOTAL
```

Conteúdo de `key.json`:

```
{
    "Artist": {"S": "Acme Band"},
    "SongTitle": {"S": "Happy Day"}
}
```

Saída:

```
{
    "Item": {
        "AlbumTitle": {
            "S": "Songs About Life"
        },
        "SongTitle": {
            "S": "Happy Day"
        },
        "Artist": {
            "S": "Acme Band"
        }
    },
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 1.0
    }
}
```

Para obter mais informações, consulte Ler um item no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 3: como recuperar atributos específicos de um item

O exemplo a seguir usa uma expressão de projeção para recuperar apenas três atributos do item desejado.

```
aws dynamodb get-item \
    --table-name ProductCatalog \
    --key '{"Id": {"N": "102"}}' \
    --projection-expression "#T, #C, #P" \
    --expression-attribute-names file://names.json
```

Conteúdo de `names.json`:

```
{
    "#T": "Title",
    "#C": "ProductCategory",
    "#P": "Price"
}
```

Saída:

```
{
    "Item": {
        "Price": {
            "N": "20"
        },
        "Title": {
            "S": "Book 102 Title"
        },
        "ProductCategory": {
            "S": "Book"
        }
    }
}
```

Para obter mais informações, consulte Ler um item no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte GetItemem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
```

```
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// GetMovie gets movie data from the DynamoDB table by using the primary
 composite key
// made of title and year.
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 movie := Movie{Title: title, Year: year}
 response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key: movie.GetKey(), TableName: aws.String(basics.TableName),
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Item, &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
```

```
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte GetItema Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtém um item de uma tabela usando DynamoDbClient o.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal>

                Where:
```

```
                    tableName - The Amazon DynamoDB table from which an item is
retrieved (for example, Music3).\s
                    key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
                    keyval - The key value that represents the item to get (for
example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        getDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void getDynamoDBItem(DynamoDbClient ddb, String tableName,
String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName(tableName)
                .build();

        try {
            // If there is no matching item, GetItem does not return any data.
            Map<String, AttributeValue> returnedItem =
ddb.getItem(request).item();
            if (returnedItem.isEmpty())
```

```
                System.out.format("No item found with the key %s!\n", key);
            else {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");
                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte GetCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, GetCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new GetCommand({
```

```
    TableName: "AngryAnimals",
    Key: {
      CommonName: "Shoebill",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte GetItema Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Obtenha um item de uma tabela.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Key: {
    KEY_NAME: { N: "001" },
  },
  ProjectionExpression: "ATTRIBUTE_NAME",
};

// Call DynamoDB to read the item from the table
ddb.getItem(params, function (err, data) {
```

```
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

Obtenha um item de uma tabela usando o cliente de documento do DynamoDB.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  TableName: "EPISODES_TABLE",
  Key: { KEY_NAME: VALUE },
};

docClient.get(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetItema Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSpecificItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- Para obter detalhes da API, consulte a GetItemreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

    public function getItemByKey(string $tableName, array $key)
    {
        return $this->dynamoDbClient->getItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- Para obter detalhes da API, consulte GetItema Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna o item do DynamoDB com a chave de partição e a SongTitle chave de classificação Artist.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Saída:

```
Name                            Value
----                            -----
Genre                           Country
SongTitle                       Somewhere Down The Road
Price                           1.94
Artist                          No One You Know
CriticRating                    9
AlbumTitle                      Somewhat Famous
```

- Para obter detalhes da API, consulte GetItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna o item do DynamoDB com a chave de partição e a SongTitle chave de classificação Artist.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Saída:

```
Name                            Value
----                            -----
Genre                           Country
SongTitle                       Somewhere Down The Road
Price                           1.94
Artist                          No One You Know
CriticRating                    9
AlbumTitle                      Somewhat Famous
```

- Para obter detalhes da API, consulte GetItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Item"]
```

- Para obter detalhes da API, consulte a GetItemReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table
```

```
  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Gets movie data from the table for a specific movie.
  #
  # @param title [String] The title of the movie.
  # @param year [Integer] The release year of the movie.
  # @return [Hash] The data about the requested movie.
  def get_item(title, year)
    @table.get_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't get movie #{title} (#{year}) from table #{@table.name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte GetItema Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_item = lo_dyn->getitem(
          iv_tablename                = iv_table_name
          it_key                      = it_key ).
        DATA(lt_attr) = oo_item->get_item( ).
        DATA(lo_title) = lt_attr[ key = 'title' ]-value.
        DATA(lo_year) = lt_attr[ key = 'year' ]-value.
        DATA(lo_rating) = lt_attr[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( )
          && 'Movie year is: ' && lo_year->get_n( )
```

```
          && 'Moving rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetItemreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Return a `Movie` record describing the specified movie from the Amazon
    /// DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The movie's release year (`Int`).
    ///
    /// - Throws: `MoviesError.ItemNotFound` if the movie isn't in the table.
    ///
    /// - Returns: A `Movie` record with the movie's details.
    func get(title: String, year: Int) async throws -> Movie {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = GetItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
```

```
                tableName: self.tableName
            )
            let output = try await client.getItem(input: input)
            guard let item = output.item else {
                throw MoviesError.ItemNotFound
            }

            let movie = try Movie(withItem: item)
            return movie
        } catch {
            print("ERROR: get:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte GetItema referência da API AWS SDK for Swift.

Use **`ListTables`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTables`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Configurar o controle de Attribute-Based acesso

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    private static async Task ListMyTables()
    {
        Console.WriteLine("\n*** Listing tables ***");
```

```
        string lastTableNameEvaluated = null;
        do
        {
            var response = await Client.ListTablesAsync(new ListTablesRequest
            {
                Limit = 2,
                ExclusiveStartTableName = lastTableNameEvaluated
            });

            foreach (var name in response.TableNames)
            {
                Console.WriteLine(name);
            }

            lastTableNameEvaluated = response.LastEvaluatedTableName;
        } while (lastTableNameEvaluated != null);
    }
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_list_tables
#
# This function lists all the tables in a DynamoDB.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function dynamodb_list_tables() {
```

```
  response=$(aws dynamodb list-tables \
    --output text \
    --query "TableNames")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  echo "$response" | tr -s "[:space:]" "\n"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
```

```
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte ListTablesem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the Amazon DynamoDB tables for the current AWS account.
/*!
  \sa listTables()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::DynamoDB::listTables(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
    listTablesRequest.SetLimit(50);
    do {
        const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
 dynamoClient.ListTables(
                listTablesRequest);
        if (!outcome.IsSuccess()) {
            std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }

        for (const auto &tableName: outcome.GetResult().GetTableNames())
            std::cout << tableName << std::endl;
        listTablesRequest.SetExclusiveStartTableName(
                outcome.GetResult().GetLastEvaluatedTableName());

    } while (!listTablesRequest.GetExclusiveStartTableName().empty());

    return true;
}
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: listar tabelas

O `list-tables` exemplo a seguir lista todas as tabelas associadas à AWS conta atual e à região.

```
aws dynamodb list-tables
```

Saída:

```
{
    "TableNames": [
        "Forum",
        "ProductCatalog",
        "Reply",
        "Thread"
    ]
}
```

Para obter mais informações, consulte Listar nomes de tabela no Guia do desenvolvedor Amazon DynamoDB.

Exemplo 2: como limitar o tamanho da página

O exemplo a seguir retorna uma lista de todas as tabelas existentes, mas recupera apenas um item em cada chamada. Pode ser necessário realizar várias chamadas para obter a lista completa. Limitar o tamanho da página é útil ao executar os comandos da lista em um grande número de recursos, o que pode resultar em um erro de “tempo limite” ao usar o tamanho de página padrão de 1.000.

```
aws dynamodb list-tables \
    --page-size 1
```

Saída:

```
{
    "TableNames": [
        "Forum",
        "ProductCatalog",
        "Reply",
        "Thread"
    ]
}
```

Para obter mais informações, consulte Listar nomes de tabela no Guia do desenvolvedor Amazon DynamoDB.

Exemplo 3: como limitar o número de itens retornados

O exemplo a seguir limita o número de itens retornados para dois. A resposta inclui um valor `NextToken` a ser usado para recuperar a próxima página de resultados.

```
aws dynamodb list-tables \
    --max-items 2
```

Saída:

```
{
    "TableNames": [
        "Forum",
        "ProductCatalog"
    ],
    "NextToken":
 "abCDeFGhiJKlmnOPqrSTuvwxYZ1aBCdEFghijK7LM51nOpqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9"
}
```

Para obter mais informações, consulte Listar nomes de tabela no Guia do desenvolvedor Amazon DynamoDB.

Exemplo 4: como recuperar a próxima página de resultados

O comando a seguir usa o valor `NextToken` de uma chamada anterior feita ao comando `list-tables` para recuperar outra página de resultados. Como a resposta nesse caso não inclui um valor para `NextToken`, sabemos que chegamos ao final dos resultados.

```
aws dynamodb list-tables \
    --starting-
token abCDeFGhiJKlmnOPqrSTuvwxYZ1aBCdEFghijK7LM51nOpqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9
```

Saída:

```
{
    "TableNames": [
        "Reply",
        "Thread"
    ]
}
```

Para obter mais informações, consulte Listar nomes de tabela no Guia do desenvolvedor Amazon DynamoDB.

- Para obter detalhes da API, consulte ListTablesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
 var tableNames []string
 var output *dynamodb.ListTablesOutput
 var err error
 tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
 &dynamodb.ListTablesInput{})
```

```
 for tablePaginator.HasMorePages() {
  output, err = tablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't list tables. Here's why: %v\n", err)
   break
  } else {
   tableNames = append(tableNames, output.TableNames...)
  }
 }
 return tableNames, err
}
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

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

```
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request =
 ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
```

```
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
}
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import { ListTablesCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new ListTablesCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

// Call DynamoDB to retrieve the list of tables
ddb.listTables({ Limit: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err.code);
  } else {
    console.log("Table names are ", data.TableNames);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllTables() {
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListTablesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function listTables($exclusiveStartTableName = "", $limit = 100)
    {
        $this->dynamoDbClient->listTables([
            'ExclusiveStartTableName' => $exclusiveStartTableName,
            'Limit' => $limit,
        ]);
    }
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exibe detalhes de todas as tabelas, iterando automaticamente até que o serviço indique que não existem mais tabelas.

```
Get-DDBTableList
```

- Para obter detalhes da API, consulte ListTablesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exibe detalhes de todas as tabelas, iterando automaticamente até que o serviço indique que não existem mais tabelas.

```
Get-DDBTableList
```

- Para obter detalhes da API, consulte ListTablesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
```

```
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tables
```

- Para obter detalhes da API, consulte a ListTablesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Determine se uma tabela existe.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_tables(client: &Client) -> Result<Vec<String>, Error> {
    let paginator = client.list_tables().into_paginator().items().send();
    let table_names = paginator.collect::<Result<Vec<_>, _>>().await?;

    println!("Tables:");

    for name in &table_names {
        println!("  {}", name);
    }

    println!("Found {} tables", table_names.len());
    Ok(table_names)
}
```

Determine se a tabela existe.

```
pub async fn table_exists(client: &Client, table: &str) -> Result<bool, Error> {
    debug!("Checking for table: {table}");
    let table_list = client.list_tables().send().await;

    match table_list {
        Ok(list) => Ok(list.table_names().contains(&table.into())),
        Err(e) => Err(e.into()),
    }
}
```

- Para obter detalhes da API, consulte a ListTablesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_dyn->listtables( ).
        " You can loop over the oo_result to get table properties like this.
        LOOP AT oo_result->get_tablenames( ) INTO DATA(lo_table_name).
          DATA(lv_tablename) = lo_table_name->get_value( ).
        ENDLOOP.
        DATA(lv_tablecount) = lines( oo_result->get_tablenames( ) ).
        MESSAGE 'Found ' && lv_tablecount && ' tables' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTablesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB
```

```
/// Get a list of the DynamoDB tables available in the specified Region.
///
/// - Returns: An array of strings listing all of the tables available
///   in the Region specified when the session was created.
public func getTableList() async throws -> [String] {
    let input = ListTablesInput(
    )
    return try await session.listTables(input: input)
}
```

- Para obter detalhes da API, consulte ListTablesa referência da API AWS SDK for Swift.

Use **`PutItem`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutItem`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Acelerar leituras com o DAX
- Cenários avançados do índice secundário global
- Criar um item com TTL
- Crie e gerencie tabelas globais demonstrando o MREC
- Criar e gerenciar tabelas globais com MRSC
- Conceitos básicos de bancos de dados NoSQL
- Usar operações condicionais
- Trabalhar com tabelas globais e a consistência final de replicação mutirregional (MREC)
- Trabalhe com Streams e Time-to-Live

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Adds a new item to the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing informtation for
    /// the movie to add to the table.</param>
    /// <param name="tableName">The name of the table where the item will be
 added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
returns>
    public async Task<bool> PutItemAsync(Movie newMovie, string tableName)
    {
        try
        {
            var item = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = item,
            };

            await _amazonDynamoDB.PutItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
```

```
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while putting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while putting item.
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte PutItema Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -i item  -- Path to json file containing the item values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
##############################################################################
function dynamodb_put_item() {
  local table_name item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_put_item"
    echo "Put an item into a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -i item  -- Path to json file containing the item values."
    echo ""
  }

  while getopts "n:i:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi
```

```
  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""
  iecho ""

  response=$(aws dynamodb put-item \
    --table-name "$table_name" \
    --item file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports put-item operation failed.$response"
    return 1
  fi

  return 0

}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
```

```
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte PutItemem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Put an item in an Amazon DynamoDB table.
/*!
  \sa putItem()
  \param tableName: The table name.
  \param artistKey: The artist key. This is the partition key for the table.
  \param artistValue: The artist value.
  \param albumTitleKey: The album title key.
  \param albumTitleValue: The album title value.
  \param awardsKey: The awards key.
  \param awardsValue: The awards value.
  \param songTitleKey: The song title key.
  \param songTitleValue: The song title value.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::putItem(const Aws::String &tableName,
                               const Aws::String &artistKey,
                               const Aws::String &artistValue,
                               const Aws::String &albumTitleKey,
                               const Aws::String &albumTitleValue,
                               const Aws::String &awardsKey,
                               const Aws::String &awardsValue,
                               const Aws::String &songTitleKey,
                               const Aws::String &songTitleValue,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::PutItemRequest putItemRequest;
    putItemRequest.SetTableName(tableName);
```

```
    putItemRequest.AddItem(artistKey,
 Aws::DynamoDB::Model::AttributeValue().SetS(
            artistValue)); // This is the hash key.
    putItemRequest.AddItem(albumTitleKey,
 Aws::DynamoDB::Model::AttributeValue().SetS(
            albumTitleValue));
    putItemRequest.AddItem(awardsKey,

 Aws::DynamoDB::Model::AttributeValue().SetS(awardsValue));
    putItemRequest.AddItem(songTitleKey,

 Aws::DynamoDB::Model::AttributeValue().SetS(songTitleValue));

    const Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
            putItemRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully added Item!" << std::endl;
    }
    else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
```

```
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte PutItema Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como adicionar um item a uma tabela

O `put-item` exemplo a seguir adiciona um novo item à MusicCollectiontabela.

```
aws dynamodb put-item \
    --table-name MusicCollection \
    --item file://item.json \
```

```
    --return-consumed-capacity TOTAL \
    --return-item-collection-metrics SIZE
```

Conteúdo de `item.json`:

```
{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"},
    "AlbumTitle": {"S": "Greatest Hits"}
}
```

Saída:

```
{
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 1.0
    },
    "ItemCollectionMetrics": {
        "ItemCollectionKey": {
            "Artist": {
                "S": "No One You Know"
            }
        },
        "SizeEstimateRangeGB": [
            0.0,
            1.0
        ]
    }
}
```

Para obter mais informações, consulte Gravar um item no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 2: como substituir condicionalmente um item em uma tabela

O exemplo de `put-item` a seguir substitui um item existente na tabela `MusicCollection` somente se o item existente tiver um atributo `AlbumTitle` com o valor `Greatest Hits`. O comando retorna o valor anterior do item.

```
aws dynamodb put-item \
```

```
    --table-name MusicCollection \
    --item file://item.json \
    --condition-expression "#A = :A" \
    --expression-attribute-names file://names.json \
    --expression-attribute-values file://values.json \
    --return-values ALL_OLD
```

Conteúdo de `item.json`:

```
{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"},
    "AlbumTitle": {"S": "Somewhat Famous"}
}
```

Conteúdo de `names.json`:

```
{
    "#A": "AlbumTitle"
}
```

Conteúdo de `values.json`:

```
{
    ":A": {"S": "Greatest Hits"}
}
```

Saída:

```
{
    "Attributes": {
        "AlbumTitle": {
            "S": "Greatest Hits"
        },
        "Artist": {
            "S": "No One You Know"
        },
        "SongTitle": {
            "S": "Call Me Today"
        }
    }
```

```
}
```

Se a chave já existir, você verá a seguinte saída:

```
A client error (ConditionalCheckFailedException) occurred when calling the
 PutItem operation: The conditional request failed.
```

Para obter mais informações, consulte Gravar um item no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte PutItemem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
```

```
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
 item, err := attributevalue.MarshalMap(movie)
 if err != nil {
  panic(err)
 }
 _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
  TableName: aws.String(basics.TableName), Item: item,
 })
 if err != nil {
  log.Printf("Couldn't add item to table. Here's why: %v\n", err)
 }
 return err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
```

```
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte PutItema Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloca um item em uma tabela usando DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedPutItem example.
 */
public class PutItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <albumtitle> <albumtitleval>
 <awards> <awardsval> <Songtitle> <songtitleval>

                Where:
                    tableName - The Amazon DynamoDB table in which an item is
 placed (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).
                    keyval - The key value that represents the item to get (for
 example, Famous Band).
                    albumTitle - The Album title (for example, AlbumTitle).
                    AlbumTitleValue - The name of the album (for example, Songs
 About Life ).
                    Awards - The awards column (for example, Awards).
                    AwardVal - The value of the awards (for example, 10).
```

```
                    SongTitle - The song title (for example, SongTitle).
                    SongTitleVal - The value of the song title (for example,
Happy Day).
               **Warning** This program will  place an item that you specify
into a table!
               """;

       if (args.length != 9) {
           System.out.println(usage);
           System.exit(1);
       }

       String tableName = args[0];
       String key = args[1];
       String keyVal = args[2];
       String albumTitle = args[3];
       String albumTitleValue = args[4];
       String awards = args[5];
       String awardVal = args[6];
       String songTitle = args[7];
       String songTitleVal = args[8];

       Region region = Region.US_EAST_1;
       DynamoDbClient ddb = DynamoDbClient.builder()
               .region(region)
               .build();

       putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
               songTitleVal);
       System.out.println("Done!");
       ddb.close();
   }

   public static void putItemInTable(DynamoDbClient ddb,
           String tableName,
           String key,
           String keyVal,
           String albumTitle,
           String albumTitleValue,
           String awards,
           String awardVal,
           String songTitle,
           String songTitleVal) {
```

```
        HashMap<String, AttributeValue> itemValues = new HashMap<>();
        itemValues.put(key, AttributeValue.builder().s(keyVal).build());
        itemValues.put(songTitle,
 AttributeValue.builder().s(songTitleVal).build());
        itemValues.put(albumTitle,
 AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item(itemValues)
                .build();

        try {
            PutItemResponse response = ddb.putItem(request);
            System.out.println(tableName + " was successfully updated. The
 request id is "
                    + response.responseMetadata().requestId());

        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.err.println("Be sure that it exists and that you've typed its
 name correctly!");
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte PutCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { PutCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new PutCommand({
    TableName: "HappyAnimals",
    Item: {
      CommonName: "Shiba Inu",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte PutItema Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloque um item em uma tabela.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "CUSTOMER_LIST",
  Item: {
    CUSTOMER_ID: { N: "001" },
    CUSTOMER_NAME: { S: "Richard Roe" },
  },
};

// Call DynamoDB to add the item to the table
ddb.putItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Coloque um item em uma tabela usando o cliente de documento do DynamoDB.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Item: {
    HASHKEY: VALUE,
    ATTRIBUTE_1: "STRING_VALUE",
    ATTRIBUTE_2: VALUE_2,
  },
};

docClient.put(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutItema Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
```

```
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
    songTitle: String,
    songTitleVal: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- Para obter detalhes da API, consulte a PutItemreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
```

```
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

    public function putItem(array $array)
    {
        $this->dynamoDbClient->putItem($array);
    }
```

- Para obter detalhes da API, consulte PutItema Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria um item ou substitui um item por um novo item.

```
$item = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
        AlbumTitle = 'Somewhat Famous'
        Price = 1.94
        Genre = 'Country'
        CriticRating = 9.0
} | ConvertTo-DDBItem
```

```
Set-DDBItem -TableName 'Music' -Item $item
```

- Para obter detalhes da API, consulte PutItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria um item ou substitui um item por um novo item.

```
$item = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
        AlbumTitle = 'Somewhat Famous'
        Price = 1.94
        Genre = 'Country'
        CriticRating = 9.0
} | ConvertTo-DDBItem
Set-DDBItem -TableName 'Music' -Item $item
```

- Para obter detalhes da API, consulte PutItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
```

```
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    "year": year,
                    "title": title,
                    "info": {"plot": plot, "rating": Decimal(str(rating))},
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title,
```

```
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a PutItemReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Adds a movie to the table.
  #
  # @param movie [Hash] The title, year, plot, and rating of the movie.
  def add_item(movie)
    @table.put_item(
      item: {
        'year' => movie[:year],
        'title' => movie[:title],
        'info' => { 'plot' => movie[:plot], 'rating' => movie[:rating] }
      }
    )
```

```
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte PutItema Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn add_item(client: &Client, item: Item, table: &String) ->
 Result<ItemOut, Error> {
    let user_av = AttributeValue::S(item.username);
    let type_av = AttributeValue::S(item.p_type);
    let age_av = AttributeValue::S(item.age);
    let first_av = AttributeValue::S(item.first);
    let last_av = AttributeValue::S(item.last);

    let request = client
        .put_item()
        .table_name(table)
        .item("username", user_av)
        .item("account_type", type_av)
        .item("age", age_av)
        .item("first_name", first_av)
        .item("last_name", last_av);

    println!("Executing request [{request:?}] to add item...");

    let resp = request.send().await?;

    let attributes = resp.attributes().unwrap();
```

```
    let username = attributes.get("username").cloned();
    let first_name = attributes.get("first_name").cloned();
    let last_name = attributes.get("last_name").cloned();
    let age = attributes.get("age").cloned();
    let p_type = attributes.get("p_type").cloned();

    println!(
        "Added user {:?}, {:?} {:?}, age {:?} as {:?} user",
        username, first_name, last_name, age, p_type
    );

    Ok(ItemOut {
        p_type,
        age,
        username,
        first_name,
        last_name,
    })
}
```

- Para obter detalhes da API, consulte a PutItemreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_resp) = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = it_item ).
        MESSAGE '1 row inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
```

```
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutItemreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Add a movie specified as a `Movie` structure to the Amazon DynamoDB
    /// table.
    ///
    /// - Parameter movie: The `Movie` to add to the table.
    ///
    func add(movie: Movie) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Get a DynamoDB item containing the movie data.
            let item = try await movie.getAsItem()

            // Send the `PutItem` request to Amazon DynamoDB.

            let input = PutItemInput(
                item: item,
                tableName: self.tableName
            )
            _ = try await client.putItem(input: input)
```

```
        } catch {
            print("ERROR: add movie:", dump(error))
            throw error
        }
    }


    ///
    /// Return an array mapping attribute names to Amazon DynamoDB attribute
    /// values, representing the contents of the `Movie` record as a DynamoDB
    /// item.
    ///
    /// - Returns: The movie item as an array of type
    ///   `[Swift.String:DynamoDBClientTypes.AttributeValue]`.
    ///
    func getAsItem() async throws ->
[Swift.String:DynamoDBClientTypes.AttributeValue]  {
        // Build the item record, starting with the year and title, which are
        // always present.

        var item: [Swift.String:DynamoDBClientTypes.AttributeValue] = [
            "year": .n(String(self.year)),
            "title": .s(self.title)
        ]

        // Add the `info` field with the rating and/or plot if they're
        // available.

        var details: [Swift.String:DynamoDBClientTypes.AttributeValue] = [:]
        if (self.info.rating != nil || self.info.plot != nil) {
            if self.info.rating != nil {
                details["rating"] = .n(String(self.info.rating!))
            }
            if self.info.plot != nil {
                details["plot"] = .s(self.info.plot!)
            }
        }
        item["info"] = .m(details)

        return item
    }
```

- Para obter detalhes da API, consulte PutItem a referência da API AWS SDK for Swift.

Use o **`Query`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Query`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Acelerar leituras com o DAX
- Cenários avançados do índice secundário global
- Comparar vários valores com um único atributo
- Conceitos básicos de bancos de dados NoSQL
- Gerenciar índices secundários globais
- Executar operações de consulta avançada
- Consultar uma tabela usando uma condição begins_with
- Consulte uma tabela usando um intervalo de datas
- Consultar uma tabela usando um índice secundário global
- Consultar uma tabela com uma expressão de filtro complexa
- Consultar uma tabela com uma expressão de filtro dinâmica
- Consultar uma tabela com uma expressão de filtro e limite
- Consultar uma tabela com atributos aninhados
- Consultar uma tabela com paginação
- Consultar uma tabela com leituras altamente consistentes
- Consultar itens com TTL
- Consultar tabelas usando padrões de data e hora
- Usar nomes de atributo de expressão
- Trabalhar com índices secundários locais

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public async Task<int> QueryMoviesAsync(string tableName, int year)
{
    try
    {
        var movieTable = new TableBuilder(_amazonDynamoDB, tableName)
            .AddHashKey("year", DynamoDBEntryType.Numeric)
            .AddRangeKey("title", DynamoDBEntryType.String)
            .Build();

        var filter = new QueryFilter("year", QueryOperator.Equal, year);

        Console.WriteLine("\nFind movies released in: {year}:");

        var config = new QueryOperationConfig()
        {
            Limit = 10, // 10 items per page.
            Select = SelectValues.SpecificAttributes,
            AttributesToGet = new List<string>
            {
                "title",
                "year",
            },
            ConsistentRead = true,
```

```
                Filter = filter,
            };

            // Value used to track how many movies match the
            // supplied criteria.
            var moviesFound = 0;

            var search = movieTable.Query(config);
            do
            {
                var movieList = await search.GetNextSetAsync();
                moviesFound += movieList.Count;

                foreach (var movie in movieList)
                {
                    DisplayDocument(movie);
                }
            }
            while (!search.IsDone);

            return moviesFound;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while querying
movies. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while querying movies.
{ex.Message}");
            throw;
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para .NET .

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k key_condition_expression -- The key condition expression.
#       -a attribute_names -- Path to JSON file containing the attribute names.
#       -v attribute_values -- Path to JSON file containing the attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function dynamodb_query() {
  local table_name key_condition_expression attribute_names attribute_values
 projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_query"
    echo "Query a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k key_condition_expression -- The key condition expression."
```

```
    echo " -a attribute_names -- Path to JSON file containing the attribute
 names."
    echo " -v attribute_values -- Path to JSON file containing the attribute
 values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) key_condition_expression="${OPTARG}" ;;
      a) attribute_names="${OPTARG}" ;;
      v) attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
 parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
```

```
    return 1
  fi

  if [[ -z "$attribute_values" ]]; then
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values")
  else
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
```

```
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Consulte detalhes da API em Query na Referência de comandos da AWS CLI .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Perform a query on an Amazon DynamoDB Table and retrieve items.
/*!
  \sa queryItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param projectionExpression: The projections expression, which is ignored if
 empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */

/*
 * The partition key attribute is searched with the specified value. By default,
 all fields and values
 * contained in the item are returned. If an optional projection expression is
 * specified on the command line, only the specified fields and values are
 * returned.
 */

bool AwsDoc::DynamoDB::queryItems(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::String &projectionExpression,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::QueryRequest request;

    request.SetTableName(tableName);
```

```
   if (!projectionExpression.empty()) {
       request.SetProjectionExpression(projectionExpression);
   }

   // Set query key condition expression.
   request.SetKeyConditionExpression(partitionKey + "= :valueToMatch");

   // Set Expression AttributeValues.
   Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
   attributeValues.emplace(":valueToMatch", partitionValue);

   request.SetExpressionAttributeValues(attributeValues);

   bool result = true;

   // "exclusiveStartKey" is used for pagination.
   Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
exclusiveStartKey;
   do {
       if (!exclusiveStartKey.empty()) {
           request.SetExclusiveStartKey(exclusiveStartKey);
           exclusiveStartKey.clear();
       }
       // Perform Query operation.
       const Aws::DynamoDB::Model::QueryOutcome &outcome =
dynamoClient.Query(request);
       if (outcome.IsSuccess()) {
           // Reference the retrieved items.
           const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
           if (!items.empty()) {
               std::cout << "Number of items retrieved from Query: " <<
items.size()
                         << std::endl;
               // Iterate each item and print.
               for (const auto &item: items) {
                   std::cout
                           <<
"*******************************************************"
                           << std::endl;
                   // Output each retrieved field and its value.
                   for (const auto &i: item)
```

```
                        std::cout << i.first << ": " << i.second.GetS() <<
 std::endl;
                }
            }
            else {
                std::cout << "No item found in table: " << tableName <<
 std::endl;
            }

            exclusiveStartKey = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Query items: " <<
 outcome.GetError().GetMessage();
            result = false;
            break;
        }
    } while (!exclusiveStartKey.empty());

    return result;
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Exemplo 1: como consultar uma tabela

O exemplo da `query` a seguir consulta itens da tabela `MusicCollection`. A tabela tem uma chave primária de hash e intervalo (`Artist` e `SongTitle`), mas essa consulta especifica apenas o valor da chave de hash. Ela retorna nomes de músicas do artista “No One You Know”.

```
aws dynamodb query \
    --table-name MusicCollection \
    --projection-expression "SongTitle" \
    --key-condition-expression "Artist = :v1" \
    --expression-attribute-values file://expression-attributes.json \
    --return-consumed-capacity TOTAL
```

Conteúdo de `expression-attributes.json`:

```
{
    ":v1": {"S": "No One You Know"}
}
```

Saída:

```
{
    "Items": [
        {
            "SongTitle": {
                "S": "Call Me Today"
            },
            "SongTitle": {
                "S": "Scared of My Shadow"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 0.5
    }
}
```

Para obter mais informações, consulte Operações de consulta no DynamoDB no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 2: como consultar uma tabela usando leituras altamente consistentes e percorrer o índice em ordem decrescente

O exemplo a seguir executa a mesma consulta do primeiro exemplo, mas retorna os resultados na ordem inversa e usa leituras altamente consistentes.

```
aws dynamodb query \
    --table-name MusicCollection \
    --projection-expression "SongTitle" \
    --key-condition-expression "Artist = :v1" \
    --expression-attribute-values file://expression-attributes.json \
    --consistent-read \
```

```
    --no-scan-index-forward \
    --return-consumed-capacity TOTAL
```

Conteúdo de `expression-attributes.json`:

```
{
    ":v1": {"S": "No One You Know"}
}
```

Saída:

```
{
    "Items": [
        {
            "SongTitle": {
                "S": "Scared of My Shadow"
            }
        },
        {
            "SongTitle": {
                "S": "Call Me Today"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 1.0
    }
}
```

Para obter mais informações, consulte Operações de consulta no DynamoDB no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 3: como filtrar resultados específicos

O exemplo a seguir consulta o `MusicCollection`, mas exclui os resultados com valores específicos no atributo `AlbumTitle`. Observe que isso não afeta `ScannedCount` ou `ConsumedCapacity` já que o filtro é aplicado após a leitura dos itens.

```
aws dynamodb query \
```

```
    --table-name MusicCollection \
    --key-condition-expression "#n1 = :v1" \
    --filter-expression "NOT (#n2 IN (:v2, :v3))" \
    --expression-attribute-names file://names.json \
    --expression-attribute-values file://values.json \
    --return-consumed-capacity TOTAL
```

Conteúdo de `values.json`:

```
{
    ":v1": {"S": "No One You Know"},
    ":v2": {"S": "Blue Sky Blues"},
    ":v3": {"S": "Greatest Hits"}
}
```

Conteúdo de `names.json`:

```
{
    "#n1": "Artist",
    "#n2": "AlbumTitle"
}
```

Saída:

```
{
    "Items": [
        {
            "AlbumTitle": {
                "S": "Somewhat Famous"
            },
            "Artist": {
                "S": "No One You Know"
            },
            "SongTitle": {
                "S": "Call Me Today"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 2,
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
```

```
        "CapacityUnits": 0.5
    }
}
```

Para obter mais informações, consulte Operações de consulta no DynamoDB no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 4: como recuperar somente uma contagem de itens

O exemplo a seguir recupera uma contagem de itens que correspondem à consulta, mas não recupera os itens em si.

```
aws dynamodb query \
    --table-name MusicCollection \
    --select COUNT \
    --key-condition-expression "Artist = :v1" \
    --expression-attribute-values file://expression-attributes.json
```

Conteúdo de `expression-attributes.json`:

```
{
    ":v1": {"S": "No One You Know"}
}
```

Saída:

```
{
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": null
}
```

Para obter mais informações, consulte Operações de consulta no DynamoDB no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 5: como consultar um índice

O exemplo a seguir consulta o índice secundário local `AlbumTitleIndex`. A consulta retorna todos os atributos da tabela base projetados no índice secundário local. Ao consultar um índice secundário local ou global, você deve fornecer o nome da tabela base usando o parâmetro `table-name`.

```
aws dynamodb query \
    --table-name MusicCollection \
    --index-name AlbumTitleIndex \
    --key-condition-expression "Artist = :v1" \
    --expression-attribute-values file://expression-attributes.json \
    --select ALL_PROJECTED_ATTRIBUTES \
    --return-consumed-capacity INDEXES
```

Conteúdo de `expression-attributes.json`:

```
{
    ":v1": {"S": "No One You Know"}
}
```

Saída:

```
{
    "Items": [
        {
            "AlbumTitle": {
                "S": "Blue Sky Blues"
            },
            "Artist": {
                "S": "No One You Know"
            },
            "SongTitle": {
                "S": "Scared of My Shadow"
            }
        },
        {
            "AlbumTitle": {
                "S": "Somewhat Famous"
            },
            "Artist": {
                "S": "No One You Know"
            },
            "SongTitle": {
                "S": "Call Me Today"
            }
        }
    ],
    "Count": 2,
```

```
    "ScannedCount": 2,
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 0.5,
        "Table": {
            "CapacityUnits": 0.0
        },
        "LocalSecondaryIndexes": {
            "AlbumTitleIndex": {
                "CapacityUnits": 0.5
            }
        }
    }
}
```

Para obter mais informações, consulte Operações de consulta no DynamoDB no Guia do desenvolvedor do Amazon DynamoDB.

- Consulte detalhes da API em Query na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
```

```
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// Query gets all movies in the DynamoDB table that were released in the
 specified year.
// The function uses the `expression` package to build the key condition
 expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
 error) {
 var err error
 var response *dynamodb.QueryOutput
 var movies []Movie
 keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
 expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
 if err != nil {
  log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
 } else {
  queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
 &dynamodb.QueryInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   KeyConditionExpression:    expr.KeyCondition(),
  })
  for queryPaginator.HasMorePages() {
   response, err = queryPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
 releaseYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
```

```
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
```

```
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Consulte detalhes da API em Query na Referência da API do AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Consulta uma tabela usando DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedQueryRecords example.
 */
public class Query {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <partitionKeyName> <partitionKeyVal>

                Where:
                    tableName - The Amazon DynamoDB table to put the item in (for
 example, Music3).
                    partitionKeyName - The partition key name of the Amazon
 DynamoDB table (for example, Artist).
                    partitionKeyVal - The value of the partition key that should
 match (for example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String partitionKeyName = args[1];
        String partitionKeyVal = args[2];

        // For more information about an alias, see:
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
        String partitionAlias = "#a";
```

```
        System.out.format("Querying %s", tableName);
        System.out.println("");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
partitionAlias);
        System.out.println("There were " + count + "  record(s) returned");
        ddb.close();
    }

    public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal,
            String partitionAlias) {
        // Set up an alias for the partition key name in case it's a reserved
word.
        HashMap<String, String> attrNameAlias = new HashMap<String, String>();
        attrNameAlias.put(partitionAlias, partitionKeyName);

        // Set up mapping of the partition name with the value.
        HashMap<String, AttributeValue> attrValues = new HashMap<>();
        attrValues.put(":" + partitionKeyName, AttributeValue.builder()
                .s(partitionKeyVal)
                .build());

        QueryRequest queryReq = QueryRequest.builder()
                .tableName(tableName)
                .keyConditionExpression(partitionAlias + " = :" +
partitionKeyName)
                .expressionAttributeNames(attrNameAlias)
                .expressionAttributeValues(attrValues)
                .build();

        try {
            QueryResponse response = ddb.query(queryReq);
            return response.count();

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return -1;
```

```
    }
}
```

Consulta uma tabela usando o `DynamoDbClient` e um índice secundário.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
 * **year-index**. For more information, see:
 *
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
 */
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        queryIndex(ddb, tableName);
        ddb.close();
    }
```

```
    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
            expressionAttributeValues.put(":yearValue",
 AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                    .tableName(tableName)
                    .indexName("year-index")
                    .keyConditionExpression("#year = :yearValue")
                    .expressionAttributeNames(expressionAttributesNames)
                    .expressionAttributeValues(expressionAttributeValues)
                    .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                    .forEach(movie ->
 System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte QueryCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { QueryCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new QueryCommand({
    TableName: "CoffeeCrop",
    KeyConditionExpression:
      "OriginCountry = :originCountry AND RoastDate > :roastDate",
    ExpressionAttributeValues: {
      ":originCountry": "Ethiopia",
      ":roastDate": "2023-05-01",
    },
    ConsistentRead: true,
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
```

```
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  ExpressionAttributeValues: {
    ":s": 2,
    ":e": 9,
    ":topic": "PHRASE",
  },
  KeyConditionExpression: "Season = :s and Episode > :e",
  FilterExpression: "contains (Subtitle, :topic)",
  TableName: "EPISODES_TABLE",
};

docClient.query(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Items);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun queryDynTable(
    tableNameVal: String,
```

```
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
```

```
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);

    public function query(string $tableName, $key)
    {
        $expressionAttributeValues = [];
        $expressionAttributeNames = [];
        $keyConditionExpression = "";
        $index = 1;
        foreach ($key as $name => $value) {
            $keyConditionExpression .= "#" . array_key_first($value) . " = :v
$index,";
            $expressionAttributeNames["#" . array_key_first($value)] =
 array_key_first($value);
            $hold = array_pop($value);
            $expressionAttributeValues[":v$index"] = [
                array_key_first($hold) => array_pop($hold),
            ];
        }
        $keyConditionExpression = substr($keyConditionExpression, 0, -1);
        $query = [
            'ExpressionAttributeValues' => $expressionAttributeValues,
            'ExpressionAttributeNames' => $expressionAttributeNames,
            'KeyConditionExpression' => $keyConditionExpression,
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->query($query);
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para PHP .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: invoca uma consulta que retorna itens do DynamoDB com o especificado e o artista. SongTitle

```
$invokeDDBQuery = @{
    TableName = 'Music'
```

```
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Para ver detalhes da API, consulte Query na Ferramentas da AWS para PowerShell Cmdlet Reference (V4).

Ferramentas para PowerShell V5

Exemplo 1: invoca uma consulta que retorna itens do DynamoDB com o especificado e o artista. SongTitle

```
$invokeDDBQuery = @{
    TableName = 'Music'
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Country
```

```
Artist                          No One You Know
Price                           1.94
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
```

- Para ver detalhes da API, consulte Query na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Consulte itens usando uma expressão de condição de chave.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
```

```
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
        :return: The list of movies that were released in the specified year.
        """
        try:
            response =
 self.table.query(KeyConditionExpression=Key("year").eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
                year,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Items"]
```

Consulte itens e projete-os para retornar um subconjunto de dados.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def query_and_project_movies(self, year, title_bounds):
```

```
        """
        Query for movies that were released in a specified year and that have
titles
        that start within a range of letters. A projection expression is used
        to return a subset of data for each movie.

        :param year: The release year to query.
        :param title_bounds: The range of starting letters to query.
        :return: The list of movies.
        """
        try:
            response = self.table.query(
                ProjectionExpression="#yr, title, info.genres, info.actors[0]",
                ExpressionAttributeNames={"#yr": "year"},
                KeyConditionExpression=(
                    Key("year").eq(year)
                    & Key("title").between(
                        title_bounds["first"], title_bounds["second"]
                    )
                ),
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                logger.warning(
                    "There's a validation error. Here's the message: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't query for movies. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return response["Items"]
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.
  # @return [Array] The list of movies that were released in the specified year.
  def query_items(year)
    response = @table.query(
      key_condition_expression: '#yr = :year',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: { ':year' => year }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't query for movies released in #{year}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.items
  end
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Encontre os filmes feitos no ano especificado.

```
pub async fn movies_in_year(
    client: &Client,
    table_name: &str,
    year: u16,
) -> Result<Vec<Movie>, MovieError> {
    let results = client
        .query()
        .table_name(table_name)
        .key_condition_expression("#yr = :yyyy")
        .expression_attribute_names("#yr", "year")
        .expression_attribute_values(":yyyy",
 AttributeValue::N(year.to_string()))
        .send()
        .await?;

    if let Some(items) = results.items {
        let movies = items.iter().map(|v| v.into()).collect();
        Ok(movies)
    } else {
        Ok(vec![])
    }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Query movies for a given year .
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
            ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_year }| ) ) ).
        DATA(lt_key_conditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
          ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
          key = 'year'
          value = NEW /aws1/cl_dyncondition(
          it_attributevaluelist = lt_attributelist
          iv_comparisonoperator = |EQ|
          ) ) ) ).
        oo_result = lo_dyn->query(
          iv_tablename = iv_table_name
          it_keyconditions = lt_key_conditions ).
        DATA(lt_items) = oo_result->get_items( ).
        "You can loop over the results to get item attributes.
        LOOP AT lt_items INTO DATA(lt_item).
          DATA(lo_title) = lt_item[ key = 'title' ]-value.
          DATA(lo_year) = lt_item[ key = 'year' ]-value.
        ENDLOOP.
        DATA(lv_count) = oo_result->get_count( ).
        MESSAGE 'Item count is: ' && lv_count TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Query na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Get all the movies released in the specified year.
    ///
    /// - Parameter year: The release year of the movies to return.
    ///
    /// - Returns: An array of `Movie` objects describing each matching movie.
    ///
    func getMovies(fromYear year: Int) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = QueryInput(
                expressionAttributeNames: [
                    "#y": "year"
                ],
                expressionAttributeValues: [
                    ":y": .n(String(year))
                ],
                keyConditionExpression: "#y = :y",
                tableName: self.tableName
            )
            // Use "Paginated" to get all the movies.
            // This lets the SDK handle the 'lastEvaluatedKey' property in
 "QueryOutput".

            let pages = client.queryPaginated(input: input)

            var movieList: [Movie] = []
```

```
            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Convert the found movies into `Movie` objects and return an
array
                // of them.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList
        } catch {
            print("ERROR: getMovies:", dump(error))
            throw error
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Swift.

Use o **`Scan`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Scan`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Acelerar leituras com o DAX
- Comparar vários valores com um único atributo

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Scans the table for movies released between the specified years.
    /// </summary>
    /// <param name="tableName">The name of the table to scan.</param>
    /// <param name="startYear">The starting year for the range.</param>
    /// <param name="endYear">The ending year for the range.</param>
    /// <returns>The number of movies found in the specified year range.</
returns>
    public async Task<int> ScanTableAsync(
        string tableName,
        int startYear,
        int endYear)
    {
        try
        {
            var request = new ScanRequest
            {
                TableName = tableName,
                ExpressionAttributeNames = new Dictionary<string, string>
                {
                    { "#yr", "year" },
                },
                ExpressionAttributeValues = new Dictionary<string,
 AttributeValue>
                {
                    { ":y_a", new AttributeValue { N = startYear.ToString() } },
                    { ":y_z", new AttributeValue { N = endYear.ToString() } },
                },
                FilterExpression = "#yr between :y_a and :y_z",
                ProjectionExpression = "#yr, title, info.actors[0],
 info.directors, info.running_time_secs",
```

```
                Limit = 10 // Set a limit to demonstrate using the
LastEvaluatedKey.
            };

            // Keep track of how many movies were found.
            int foundCount = 0;

            var response = new ScanResponse();
            do
            {
                response = await _amazonDynamoDB.ScanAsync(request);
                foundCount += response.Items.Count;
                response.Items.ForEach(i => DisplayItem(i));
                request.ExclusiveStartKey = response.LastEvaluatedKey;
            }
            while (response?.LastEvaluatedKey?.Count > 0);
            return foundCount;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while scanning
table. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while scanning table.
{ex.Message}");
            throw;
        }
    }
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para .NET .

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -f filter_expression  -- The filter expression.
#       -a expression_attribute_names -- Path to JSON file containing the
 expression attribute names.
#       -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_scan() {
  local table_name filter_expression expression_attribute_names
 expression_attribute_values projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_scan"
    echo "Scan a DynamoDB table."
    echo " -n table_name  -- The name of the table."
```

```
    echo " -f filter_expression  -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
expression attribute names."
    echo " -v expression_attribute_values -- Path to JSON file containing the
expression attribute values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      f) filter_expression="${OPTARG}" ;;
      a) expression_attribute_names="${OPTARG}" ;;
      v) expression_attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_names" ]]; then
    errecho "ERROR: You must provide expression attribute names with the -a
parameter."
```

```
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
 parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values")
  else
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
```

```
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Consulte detalhes da API em Scan na Referência de comandos da AWS CLI .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Scan an Amazon DynamoDB table.
/*!
  \sa scanTable()
  \param tableName: Name for the DynamoDB table.
  \param projectionExpression: An optional projection expression, ignored if
 empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::scanTable(const Aws::String &tableName,
                                 const Aws::String &projectionExpression,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::ScanRequest request;
    request.SetTableName(tableName);

    if (!projectionExpression.empty())
        request.SetProjectionExpression(projectionExpression);

    Aws::Vector<Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>>
 all_items;
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 last_evaluated_key; // Used for pagination;
    do {
        if (!last_evaluated_key.empty()) {
            request.SetExclusiveStartKey(last_evaluated_key);
```

```
        }
        const Aws::DynamoDB::Model::ScanOutcome &outcome =
 dynamoClient.Scan(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            all_items.insert(all_items.end(), items.begin(), items.end());

            last_evaluated_key = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Scan items: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!last_evaluated_key.empty());

    if (!all_items.empty()) {
        std::cout << "Number of items retrieved from scan: " << all_items.size()
                  << std::endl;
        // Iterate each item and print.
        for (const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 &itemMap: all_items) {
            std::cout << "*****************************************************"
                      << std::endl;
            // Output each retrieved field and its value.
            for (const auto &itemEntry: itemMap)
                std::cout << itemEntry.first << ": " << itemEntry.second.GetS()
                          << std::endl;
        }
    }

    else {
        std::cout << "No items found in table: " << tableName << std::endl;
    }

    return true;
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Como verificar uma tabela

O exemplo de `scan` faz uma varredura da tabela `MusicCollection` e restringe os resultados a músicas do artista “No One You Know”. Em cada item, somente o nome do álbum e da música são retornados.

```
aws dynamodb scan \
    --table-name MusicCollection \
    --filter-expression "Artist = :a" \
    --projection-expression "#ST, #AT" \
    --expression-attribute-names file://expression-attribute-names.json \
    --expression-attribute-values file://expression-attribute-values.json
```

Conteúdo de `expression-attribute-names.json`:

```
{
    "#ST": "SongTitle",
    "#AT":"AlbumTitle"
}
```

Conteúdo de `expression-attribute-values.json`:

```
{
    ":a": {"S": "No One You Know"}
}
```

Saída:

```
{
    "Count": 2,
    "Items": [
        {
            "SongTitle": {
                "S": "Call Me Today"
            },
```

```
            "AlbumTitle": {
                "S": "Somewhat Famous"
            }
        },
        {
            "SongTitle": {
                "S": "Scared of My Shadow"
            },
            "AlbumTitle": {
                "S": "Blue Sky Blues"
            }
        }
    ],
    "ScannedCount": 3,
    "ConsumedCapacity": null
}
```

Para obter mais informações, consulte Trabalhar com verificações no DynamoDB no Guia do desenvolvedor do Amazon DynamoDB.

- Consulte detalhes da API em Scan na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
```

```
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// Scan gets all movies in the DynamoDB table that were released in a range of
 years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
 ([]Movie, error) {
 var movies []Movie
 var err error
 var response *dynamodb.ScanOutput
 filtEx := expression.Name("year").Between(expression.Value(startYear),
 expression.Value(endYear))
 projEx := expression.NamesList(
  expression.Name("year"), expression.Name("title"),
 expression.Name("info.rating"))
 expr, err :=
 expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
 if err != nil {
  log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
 } else {
  scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
 &dynamodb.ScanInput{
   TableName:                aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   FilterExpression:         expr.Filter(),
   ProjectionExpression:     expr.Projection(),
  })
  for scanPaginator.HasMorePages() {
```

```
    response, err = scanPaginator.NextPage(ctx)
    if err != nil {
     log.Printf("Couldn't scan for movies released between %v and %v. Here's why:
 %v\n",
      startYear, endYear, err)
     break
    } else {
     var moviePage []Movie
     err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
     if err != nil {
      log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
      break
     } else {
      movies = append(movies, moviePage...)
     }
    }
   }
  }
  return movies, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
```

```
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Consulte detalhes da API em Scan na Referência da API do AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Escaneia uma tabela do Amazon DynamoDB DynamoDbClientusando.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */

public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information from
 (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
```

```
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        scanItems(ddb, tableName);
        ddb.close();
    }

    public static void scanItems(DynamoDbClient ddb, String tableName) {
        try {
            ScanRequest scanRequest = ScanRequest.builder()
                    .tableName(tableName)
                    .build();

            ScanResponse response = ddb.scan(scanRequest);
            for (Map<String, AttributeValue> item : response.items()) {
                Set<String> keys = item.keySet();
                for (String key : keys) {
                    System.out.println("The key name is " + key + "\n");
                    System.out.println("The value is " + item.get(key).s());
                }
            }

        } catch (DynamoDbException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte ScanCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, ScanCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ScanCommand({
    ProjectionExpression: "#Name, Color, AvgLifeSpan",
    ExpressionAttributeNames: { "#Name": "Name" },
    TableName: "Birds",
  });

  const response = await docClient.send(command);
  for (const bird of response.Items) {
    console.log(`${bird.Name} - (${bird.Color}, ${bird.AvgLifeSpan})`);
  }
  return response;
};
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para JavaScript .

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js.
var AWS = require("aws-sdk");
// Set the AWS Region.
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object.
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });
```

```
const params = {
  // Specify which items in the results are returned.
  FilterExpression: "Subtitle = :topic AND Season = :s AND Episode = :e",
  // Define the expression attribute value, which are substitutes for the values
 you want to compare.
  ExpressionAttributeValues: {
    ":topic": { S: "SubTitle2" },
    ":s": { N: 1 },
    ":e": { N: 2 },
  },
  // Set the projection expression, which are the attributes that you want.
  ProjectionExpression: "Season, Episode, Title, Subtitle",
  TableName: "EPISODES_TABLE",
};

ddb.scan(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
    data.Items.forEach(function (element, index, array) {
      console.log(
        "printing",
        element.Title.S + " (" + element.Subtitle.S + ")"
      );
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Scan na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

    public function scan(string $tableName, array $key, string $filters)
    {
        $query = [
            'ExpressionAttributeNames' => ['#year' => 'year'],
            'ExpressionAttributeValues' => [
                ":min" => ['N' => '1990'],
                ":max" => ['N' => '1999'],
            ],
            'FilterExpression' => "#year between :min and :max",
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->scan($query);
    }
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para PHP .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exibe todos os itens da tabela Music.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Saída:

```
Name                            Value
----                            -----
Genre                           Country
Artist                          No One You Know
Price                           1.94
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
Genre                           Country
Artist                          No One You Know
Price                           1.98
CriticRating                    8.4
SongTitle                       My Dog Spot
AlbumTitle                      Hey Now
```

Exemplo 2: Retorna itens na tabela Música com um valor CriticRating maior ou igual a nove.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-
DDBItem
```

Saída:

```
Name                            Value
----                            -----
Genre                           Country
Artist                          No One You Know
Price                           1.94
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
```

- Para ver detalhes da API, consulte Scan na Ferramentas da AWS para PowerShell Cmdlet Reference (V4).

Ferramentas para PowerShell V5

Exemplo 1: exibe todos os itens da tabela Music.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
Genre                          Country
Artist                         No One You Know
Price                          1.98
CriticRating                   8.4
SongTitle                      My Dog Spot
AlbumTitle                     Hey Now
```

Exemplo 2: Retorna itens na tabela Música com um valor CriticRating maior ou igual a nove.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-
DDBItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
```

```
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
```

- Para ver detalhes da API, consulte Scan na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
```

```
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def scan_movies(self, year_range):
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            "FilterExpression": Key("year").between(
                year_range["first"], year_range["second"]
            ),
            "ProjectionExpression": "#yr, title, info.rating",
            "ExpressionAttributeNames": {"#yr": "year"},
        }
        try:
            done = False
            start_key = None
            while not done:
                if start_key:
                    scan_kwargs["ExclusiveStartKey"] = start_key
                response = self.table.scan(**scan_kwargs)
                movies.extend(response.get("Items", []))
                start_key = response.get("LastEvaluatedKey", None)
                done = start_key is None
        except ClientError as err:
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

        return movies
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
    movies = []
    scan_hash = {
      filter_expression: '#yr between :start_yr and :end_yr',
      projection_expression: '#yr, title, info.rating',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: {
        ':start_yr' => year_range[:start], ':end_yr' => year_range[:end]
      }
    }
    done = false
    start_key = nil
    until done
      scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
```

```
      response = @table.scan(scan_hash)
      movies.concat(response.items) unless response.items.empty?
      start_key = response.last_evaluated_key
      done = start_key.nil?
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't scan for movies. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    movies
  end
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_items(client: &Client, table: &str, page_size: Option<i32>) ->
 Result<(), Error> {
    let page_size = page_size.unwrap_or(10);
    let items: Result<Vec<_>, _> = client
        .scan()
        .table_name(table)
        .limit(page_size)
        .into_paginator()
        .items()
        .send()
        .collect()
        .await;

    println!("Items in table (up to {page_size}):");
    for item in items? {
        println!("   {:?}", item);
```

```
    }

    Ok(())
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Scan movies for rating greater than or equal to the rating specified
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
            ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_rating }| ) ) ).
        DATA(lt_filter_conditions) = VALUE /aws1/
cl_dyncondition=>tt_filterconditionmap(
          ( VALUE /aws1/cl_dyncondition=>ts_filterconditionmap_maprow(
          key = 'rating'
          value = NEW /aws1/cl_dyncondition(
          it_attributevaluelist = lt_attributelist
          iv_comparisonoperator = |GE|
          ) ) ) ).
        oo_scan_result = lo_dyn->scan( iv_tablename = iv_table_name
          it_scanfilter = lt_filter_conditions ).
        DATA(lt_items) = oo_scan_result->get_items( ).
        LOOP AT lt_items INTO DATA(lo_item).
          " You can loop over to get individual attributes.
          DATA(lo_title) = lo_item[ key = 'title' ]-value.
          DATA(lo_year) = lo_item[ key = 'year' ]-value.
        ENDLOOP.
        DATA(lv_count) = oo_scan_result->get_count( ).
        MESSAGE 'Found ' && lv_count && ' items' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
```

```
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Scan na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Return an array of `Movie` objects released in the specified range of
    /// years.
    ///
    /// - Parameters:
    ///   - firstYear: The first year of movies to return.
    ///   - lastYear: The last year of movies to return.
    ///   - startKey: A starting point to resume processing; always use `nil`.
    ///
    /// - Returns: An array of `Movie` objects describing the matching movies.
    ///
    /// > Note: The `startKey` parameter is used by this function when
    ///   recursively calling itself, and should always be `nil` when calling
    ///   directly.
    ///
    func getMovies(firstYear: Int, lastYear: Int,
                   startKey: [Swift.String: DynamoDBClientTypes.AttributeValue]?
 = nil)
        async throws -> [Movie]
    {
        do {
            var movieList: [Movie] = []
```

```
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = ScanInput(
                consistentRead: true,
                exclusiveStartKey: startKey,
                expressionAttributeNames: [
                    "#y": "year" // `year` is a reserved word, so use `#y`
instead.
                ],
                expressionAttributeValues: [
                    ":y1": .n(String(firstYear)),
                    ":y2": .n(String(lastYear))
                ],
                filterExpression: "#y BETWEEN :y1 AND :y2",
                tableName: self.tableName
            )

            let pages = client.scanPaginated(input: input)

            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Build an array of `Movie` objects for the returned items.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList

        } catch {
            print("ERROR: getMovies with scan:", dump(error))
            throw error
        }
    }
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Swift.

Use **`UpdateItem`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateItem`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Atualizar condicionalmente a TTL de um item
- Contar operadores de expressão
- Criar e gerenciar tabelas globais com MRSC
- Conceitos básicos de bancos de dados NoSQL
- Executar operações de lista
- Executar operações de mapa
- Executar operações de conjunto
- Como funciona a ordem da expressão de atualização
- Atualiza a TTL de um item
- Usar operações de contador atômico
- Usar operações condicionais
- Usar nomes de atributo de expressão

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
    /// <param name="tableName">The name of the table that contains the movie.</
param>
    /// <returns>A Boolean value that indicates the success of the operation.</
returns>
    public async Task<bool> UpdateItemAsync(
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
            var updates = new Dictionary<string, AttributeValueUpdate>
            {
                ["info.plot"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { S = newInfo.Plot },
                },

                ["info.rating"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { N = newInfo.Rank.ToString() },
                },
            };

            var request = new UpdateItemRequest
            {
                AttributeUpdates = updates,
                Key = key,
                TableName = tableName,
            };
```

```
            await _amazonDynamoDB.UpdateItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} or item was not found.
{ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while updating
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while updating item.
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
```

```
#
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to update.
#       -e update expression  -- An expression that defines one or more
 attributes to be updated.
#       -v values  -- Path to json file containing the update values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_update_item() {
  local table_name keys update_expression values response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_update_item"
    echo "Update an item in a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to update."
    echo " -e update expression  -- An expression that defines one or more
 attributes to be updated."
    echo " -v values  -- Path to json file containing the update values."
    echo ""
  }

  while getopts "n:k:e:v:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      e) update_expression="${OPTARG}" ;;
      v) values="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi
if [[ -z "$update_expression" ]]; then
  errecho "ERROR: You must provide an update expression with the -e parameter."
  usage
  return 1
fi

if [[ -z "$values" ]]; then
  errecho "ERROR: You must provide a values json file path the -v parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    keys:   $keys"
iecho "    update_expression:   $update_expression"
iecho "    values:   $values"

response=$(aws dynamodb update-item \
  --table-name "$table_name" \
  --key file://"$keys" \
  --update-expression "$update_expression" \
  --expression-attribute-values file://"$values")

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
  fi

  return 0

}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
```

```
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte UpdateItemem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update an Amazon DynamoDB table item.
/*!
  \sa updateItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param attributeKey: The key for the attribute to be updated.
  \param attributeValue: The value for the attribute to be updated.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */

/*
 *  The example code only sets/updates an attribute value. It processes
 *  the attribute value as a string, even if the value could be interpreted
 *  as a number. Also, the example code does not remove an existing attribute
 *  from the key value.
 */

bool AwsDoc::DynamoDB::updateItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::String &attributeKey,
                                  const Aws::String &attributeValue,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // *** Define UpdateItem request arguments.
    // Define TableName argument.
    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(tableName);

    // Define KeyName argument.
    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(partitionValue);
    request.AddKey(partitionKey, attribValue);

    // Construct the SET update expression argument.
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument.
```

```
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = attributeKey;
    request.SetExpressionAttributeNames(expressionAttributeNames);

    // Construct attribute value argument.
    Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
    attributeUpdatedValue.SetS(attributeValue);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 expressionAttributeValues;
    expressionAttributeValues[":valueA"] = attributeUpdatedValue;
    request.SetExpressionAttributeValues(expressionAttributeValues);

    // Update the item.
    const Aws::DynamoDB::Model::UpdateItemOutcome &outcome =
 dynamoClient.UpdateItem(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Item was updated" << std::endl;
    } else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
```

```
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como atualizar um item em uma tabela

O exemplo da `update-item` a seguir atualiza um item da tabela `MusicCollection`. Ele adiciona um novo atributo (`Year`) e modifica o atributo `AlbumTitle`. Todos os atributos no item, conforme aparecem após a atualização, são retornados na resposta.

```
aws dynamodb update-item \
    --table-name MusicCollection \
```

```
    --key file://key.json \
    --update-expression "SET #Y = :y, #AT = :t" \
    --expression-attribute-names file://expression-attribute-names.json \
    --expression-attribute-values file://expression-attribute-values.json  \
    --return-values ALL_NEW \
    --return-consumed-capacity TOTAL \
    --return-item-collection-metrics SIZE
```

Conteúdo de `key.json`:

```
{
    "Artist": {"S": "Acme Band"},
    "SongTitle": {"S": "Happy Day"}
}
```

Conteúdo de `expression-attribute-names.json`:

```
{
    "#Y":"Year", "#AT":"AlbumTitle"
}
```

Conteúdo de `expression-attribute-values.json`:

```
{
    ":y":{"N": "2015"},
    ":t":{"S": "Louder Than Ever"}
}
```

Saída:

```
{
    "Attributes": {
        "AlbumTitle": {
            "S": "Louder Than Ever"
        },
        "Awards": {
            "N": "10"
        },
        "Artist": {
            "S": "Acme Band"
        },
        "Year": {
```

```
            "N": "2015"
        },
        "SongTitle": {
            "S": "Happy Day"
        }
    },
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 3.0
    },
    "ItemCollectionMetrics": {
        "ItemCollectionKey": {
            "Artist": {
                "S": "Acme Band"
            }
        },
        "SizeEstimateRangeGB": [
            0.0,
            1.0
        ]
    }
}
```

Para obter mais informações, consulte Gravar um item no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 2: como atualizar um item de forma condicional

O exemplo a seguir atualiza um item na tabela `MusicCollection`, mas somente se o item existente ainda não tiver um atributo `Year`.

```
aws dynamodb update-item \
    --table-name MusicCollection \
    --key file://key.json \
    --update-expression "SET #Y = :y, #AT = :t" \
    --expression-attribute-names file://expression-attribute-names.json \
    --expression-attribute-values file://expression-attribute-values.json  \
    --condition-expression "attribute_not_exists(#Y)"
```

Conteúdo de `key.json`:

```
{
    "Artist": {"S": "Acme Band"},
```

```
    "SongTitle": {"S": "Happy Day"}
}
```

Conteúdo de `expression-attribute-names.json`:

```
{
    "#Y":"Year",
    "#AT":"AlbumTitle"
}
```

Conteúdo de `expression-attribute-values.json`:

```
{
    ":y":{"N": "2015"},
    ":t":{"S": "Louder Than Ever"}
}
```

Se o item já tiver um atributo `Year`, o DynamoDB retornará saída a seguir.

```
An error occurred (ConditionalCheckFailedException) when calling the UpdateItem
 operation: The conditional request failed
```

Para obter mais informações, consulte Gravar um item no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte UpdateItemem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
```

```
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the
 update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
 (map[string]map[string]interface{}, error) {
 var err error
 var response *dynamodb.UpdateItemOutput
 var attributeMap map[string]map[string]interface{}
 update := expression.Set(expression.Name("info.rating"),
 expression.Value(movie.Info["rating"]))
 update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
 expr, err := expression.NewBuilder().WithUpdate(update).Build()
 if err != nil {
  log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
 } else {
  response, err = basics.DynamoDbClient.UpdateItem(ctx,
 &dynamodb.UpdateItemInput{
   TableName:                aws.String(basics.TableName),
   Key:                      movie.GetKey(),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
```

```
    UpdateExpression:          expr.Update(),
    ReturnValues:              types.ReturnValueUpdatedNew,
   })
   if err != nil {
    log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
   } else {
    err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
    if err != nil {
     log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
    }
   }
  }
  return attributeMap, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}
```

```
// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualiza um item em uma tabela usando DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
 * practice to use the
 * Enhanced Client, See the EnhancedModifyItem example.
 */
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <name> <updateVal>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music3).
                    key - The name of the key in the table (for example, Artist).
                    keyVal - The value of the key (for example, Famous Band).
                    name - The name of the column where the value is updated (for
 example, Awards).
                    updateVal - The value used to update an item (for example,
 14).
                 Example:
                    UpdateItem Music3 Artist Famous Band Awards 14
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
```

```
        String keyVal = args[2];
        String name = args[3];
        String updateVal = args[4];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
        ddb.close();
    }

    public static void updateTableItem(DynamoDbClient ddb,
            String tableName,
            String key,
            String keyVal,
            String name,
            String updateVal) {

        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put(name, AttributeValueUpdate.builder()
                .value(AttributeValue.builder().s(updateVal).build())
                .action(AttributeAction.PUT)
                .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(itemKey)
                .attributeUpdates(updatedValues)
                .build();

        try {
            ddb.updateItem(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("The Amazon DynamoDB table was updated!");
    }
```

```
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte UpdateCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, UpdateCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new UpdateCommand({
    TableName: "Dogs",
    Key: {
      Breed: "Labrador",
    },
    UpdateExpression: "set Color = :color",
    ExpressionAttributeValues: {
      ":color": "black",
    },
    ReturnValues: "ALL_NEW",
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
```

```
};
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
    name: String,
    updateVal: String,
) {
    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] =
        AttributeValueUpdate {
            value = AttributeValue.S(updateVal)
            action = AttributeAction.Put
        }

    val request =
        UpdateItemRequest {
            tableName = tableNameVal
            key = itemKey
            attributeUpdates = updatedValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
```

```
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- Para obter detalhes da API, consulte a UpdateItemreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "What rating would you like to give {$movie['Item']['title']['S']}?
\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
 $rating);

    public function updateItemAttributeByKey(
        string $tableName,
        array $key,
        string $attributeName,
        string $attributeType,
        string $newValue
    ) {
        $this->dynamoDbClient->updateItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
            'UpdateExpression' => "set #NV=:NV",
            'ExpressionAttributeNames' => [
                '#NV' => $attributeName,
            ],
```

```
            'ExpressionAttributeValues' => [
                ':NV' => [
                    $attributeType => $newValue
                ]
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: define o atributo de gênero como 'Rap' no item do DynamoDB com a chave de partição e a SongTitle chave de classificação Artist.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$updateDdbItem = @{
    TableName = 'Music'
    Key = $key
    UpdateExpression = 'set Genre = :val1'
    ExpressionAttributeValue = (@{
        ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
    })
}
Update-DDBItem @updateDdbItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Rap
```

- Para obter detalhes da API, consulte UpdateItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: define o atributo de gênero como 'Rap' no item do DynamoDB com a chave de partição e a SongTitle chave de classificação Artist.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$updateDdbItem = @{
    TableName = 'Music'
    Key = $key
    UpdateExpression = 'set Genre = :val1'
    ExpressionAttributeValue = (@{
        ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
    })
}
Update-DDBItem @updateDdbItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Rap
```

- Para obter detalhes da API, consulte UpdateItemem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize um item usando uma expressão de atualização.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to the give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
```

```
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={":r": Decimal(str(rating)), ":p":
 plot},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]
```

Atualize um item usando uma expressão de atualização que inclui uma operação aritmética.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def update_rating(self, title, year, rating_change):
        """
        Updates the quality rating of a movie in the table by using an arithmetic
        operation in the update expression. By specifying an arithmetic
 operation,
        you can adjust a value in a single request, rather than first getting its
        value and then setting its new value.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating_change: The amount to add to the current rating for the
 movie.
        :return: The updated rating.
        """
```

```
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating = info.rating + :val",
                ExpressionAttributeValues={":val": Decimal(str(rating_change))},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]
```

Atualize um item somente quando ele atender a determinadas condições.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def remove_actors(self, title, year, actor_threshold):
        """
        Removes an actor from a movie, but only when the number of actors is
 greater
        than a specified threshold. If the movie does not list more than the
 threshold,
        no actors are removed.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param actor_threshold: The threshold of actors to check.
        :return: The movie data after the update.
        """
        try:
            response = self.table.update_item(
```

```
                Key={"year": year, "title": title},
                UpdateExpression="remove info.actors[0]",
                ConditionExpression="size(info.actors) > :num",
                ExpressionAttributeValues={":num": actor_threshold},
                ReturnValues="ALL_NEW",
            )
        except ClientError as err:
            if err.response["Error"]["Code"] ==
 "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't update %s because it has fewer than %s actors.",
                    title,
                    actor_threshold + 1,
                )
            else:
                logger.error(
                    "Couldn't update movie %s. Here's why: %s: %s",
                    title,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["Attributes"]
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
```

```
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Updates rating and plot data for a movie in the table.
  #
  # @param movie [Hash] The title, year, plot, rating of the movie.
  def update_item(movie)
    response = @table.update_item(
      key: { 'year' => movie[:year], 'title' => movie[:title] },
      update_expression: 'set info.rating=:r',
      expression_attribute_values: { ':r' => movie[:rating] },
      return_values: 'UPDATED_NEW'
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't update movie #{movie[:title]} (#{movie[:year]}) in table
#{@table.name}\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.attributes
  end
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_output = lo_dyn->updateitem(
```

```
          iv_tablename        = iv_table_name
          it_key              = it_item_key
          it_attributeupdates = it_attribute_updates ).
        MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateItemreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Update the specified movie with new `rating` and `plot` information.
    ///
    /// - Parameters:
    ///   - title: The title of the movie to update.
    ///   - year: The release year of the movie to update.
    ///   - rating: The new rating for the movie.
    ///   - plot: The new plot summary string for the movie.
    ///
    /// - Returns: An array of mappings of attribute names to their new
    ///   listing each item actually changed. Items that didn't need to change
    ///   aren't included in this list. `nil` if no changes were made.
```

```
    ///
    func update(title: String, year: Int, rating: Double? = nil, plot: String? =
nil) async throws
        -> [Swift.String: DynamoDBClientTypes.AttributeValue]?
    {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Build the update expression and the list of expression attribute
            // values. Include only the information that's changed.

            var expressionParts: [String] = []
            var attrValues: [Swift.String: DynamoDBClientTypes.AttributeValue] =
[:]

            if rating != nil {
                expressionParts.append("info.rating=:r")
                attrValues[":r"] = .n(String(rating!))
            }
            if plot != nil {
                expressionParts.append("info.plot=:p")
                attrValues[":p"] = .s(plot!)
            }
            let expression = "set \(expressionParts.joined(separator: ", "))"

            let input = UpdateItemInput(
                // Create substitution tokens for the attribute values, to ensure
                // no conflicts in expression syntax.
                expressionAttributeValues: attrValues,
                // The key identifying the movie to update consists of the
release
                // year and title.
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                returnValues: .updatedNew,
                tableName: self.tableName,
                updateExpression: expression
            )
            let output = try await client.updateItem(input: input)
```

```
            guard let attributes: [Swift.String:
DynamoDBClientTypes.AttributeValue] = output.attributes else {
                throw MoviesError.InvalidAttributes
            }
            return attributes
        } catch {
            print("ERROR: update:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema referência da API AWS SDK for Swift.

Use **`UpdateTable`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Crie e gerencie tabelas globais demonstrando o MREC
- Criar e gerenciar tabelas globais com MRSC
- Gerenciar índices secundários globais
- Atualizar a configuração de throughput a quente de uma tabela
- Trabalhar com tabelas globais e a consistência final de replicação mutirregional (MREC)
- Trabalhar com a criptografia de tabelas

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update a DynamoDB table.
/*!
  \sa updateTable()
  \param tableName: Name for the DynamoDB table.
  \param readCapacity: Provisioned read capacity.
  \param writeCapacity: Provisioned write capacity.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::updateTable(const Aws::String &tableName,
                                   long long readCapacity, long long
 writeCapacity,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Updating " << tableName << " with new provisioned throughput
 values"
              << std::endl;
    std::cout << "Read capacity : " << readCapacity << std::endl;
    std::cout << "Write capacity: " << writeCapacity << std::endl;

    Aws::DynamoDB::Model::UpdateTableRequest request;
    Aws::DynamoDB::Model::ProvisionedThroughput provisionedThroughput;

 provisionedThroughput.WithReadCapacityUnits(readCapacity).WithWriteCapacityUnits(
            writeCapacity);

 request.WithProvisionedThroughput(provisionedThroughput).WithTableName(tableName);

    const Aws::DynamoDB::Model::UpdateTableOutcome &outcome =
 dynamoClient.UpdateTable(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated the table." << std::endl;
    } else {
        const Aws::DynamoDB::DynamoDBError &error = outcome.GetError();
        if (error.GetErrorType() == Aws::DynamoDB::DynamoDBErrors::VALIDATION &&
            error.GetMessage().find("The provisioned throughput for the table
 will not change") != std::string::npos) {
            std::cout << "The provisioned throughput for the table will not
 change." << std::endl;
        } else {
```

```
            std::cerr << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
```

```
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte UpdateTablea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como modificar o modo de faturamento de uma tabela

O exemplo `update-table` a seguir aumenta a capacidade de leitura e gravação provisionada na tabela `MusicCollection`.

```
aws dynamodb update-table \
    --table-name MusicCollection \
    --billing-mode PROVISIONED \
    --provisioned-throughput ReadCapacityUnits=15,WriteCapacityUnits=10
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
```

```
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "UPDATING",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": "2020-07-28T13:18:18.921000-07:00",
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 15,
            "WriteCapacityUnits": 10
        },
        "TableSizeBytes": 182,
        "ItemCount": 2,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
        "BillingModeSummary": {
            "BillingMode": "PROVISIONED",
            "LastUpdateToPayPerRequestDateTime":
 "2020-07-28T13:14:48.366000-07:00"
        }
    }
}
```

Para ter mais informações, consulte Updating a Table no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 2: como criar um índice secundário global

O exemplo a seguir adiciona um índice secundário global à tabela `MusicCollection`.

```
aws dynamodb update-table \
```

```
    --table-name MusicCollection \
    --attribute-definitions AttributeName=AlbumTitle,AttributeType=S \
    --global-secondary-index-updates file://gsi-updates.json
```

Conteúdo de `gsi-updates.json`:

```
[
    {
        "Create": {
            "IndexName": "AlbumTitle-index",
            "KeySchema": [
                {
                    "AttributeName": "AlbumTitle",
                    "KeyType": "HASH"
                }
            ],
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 10,
                "WriteCapacityUnits": 10
            },
            "Projection": {
                "ProjectionType": "ALL"
            }
        }
    }
]
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
```

```
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "UPDATING",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 15,
            "WriteCapacityUnits": 10
        },
        "TableSizeBytes": 182,
        "ItemCount": 2,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
        "BillingModeSummary": {
            "BillingMode": "PROVISIONED",
            "LastUpdateToPayPerRequestDateTime":
 "2020-07-28T13:14:48.366000-07:00"
        },
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitle-index",
                "KeySchema": [
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "HASH"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "CREATING",
```

```
                "Backfilling": false,
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 10
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
            }
        ]
    }
}
```

Para ter mais informações, consulte Updating a Table no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 3: como habilitar o DynamoDB Streams em uma tabela

O comando a seguir habilita o DynamoDB Streams na tabela `MusicCollection`.

```
aws dynamodb update-table \
    --table-name MusicCollection \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_IMAGE
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
```

```
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "UPDATING",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 15,
            "WriteCapacityUnits": 10
        },
        "TableSizeBytes": 182,
        "ItemCount": 2,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
        "BillingModeSummary": {
            "BillingMode": "PROVISIONED",
            "LastUpdateToPayPerRequestDateTime":
 "2020-07-28T13:14:48.366000-07:00"
        },
        "LocalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitleIndex",
                "KeySchema": [
                    {
                        "AttributeName": "Artist",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
```

```
                    "ProjectionType": "INCLUDE",
                    "NonKeyAttributes": [
                        "Year",
                        "Genre"
                    ]
                },
                "IndexSizeBytes": 139,
                "ItemCount": 2,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
            }
        ],
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitle-index",
                "KeySchema": [
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "HASH"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "ACTIVE",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 10
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
            }
        ],
        "StreamSpecification": {
            "StreamEnabled": true,
            "StreamViewType": "NEW_IMAGE"
        },
        "LatestStreamLabel": "2020-07-28T21:53:39.112",
        "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/stream/2020-07-28T21:53:39.112"
    }
```

```
}
```

Para ter mais informações, consulte Updating a Table no Guia do desenvolvedor do Amazon DynamoDB.

Exemplo 4: como habilitar a criptografia do lado do servidor

O exemplo a seguir habilita a criptografia do lado do servidor na tabela `MusicCollection`.

```
aws dynamodb update-table \
    --table-name MusicCollection \
    --sse-specification Enabled=true,SSEType=KMS
```

Saída:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "ACTIVE",
```

```
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 15,
            "WriteCapacityUnits": 10
        },
        "TableSizeBytes": 182,
        "ItemCount": 2,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
        "BillingModeSummary": {
            "BillingMode": "PROVISIONED",
            "LastUpdateToPayPerRequestDateTime":
 "2020-07-28T13:14:48.366000-07:00"
        },
        "LocalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitleIndex",
                "KeySchema": [
                    {
                        "AttributeName": "Artist",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "INCLUDE",
                    "NonKeyAttributes": [
                        "Year",
                        "Genre"
                    ]
                },
                "IndexSizeBytes": 139,
                "ItemCount": 2,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
            }
        ],
        "GlobalSecondaryIndexes": [
```

```
            {
                "IndexName": "AlbumTitle-index",
                "KeySchema": [
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "HASH"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "ACTIVE",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 10
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
            }
        ],
        "StreamSpecification": {
            "StreamEnabled": true,
            "StreamViewType": "NEW_IMAGE"
        },
        "LatestStreamLabel": "2020-07-28T21:53:39.112",
        "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/stream/2020-07-28T21:53:39.112",
        "SSEDescription": {
            "Status": "UPDATING"
        }
    }
}
```

Para ter mais informações, consulte Updating a Table no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte UpdateTablena Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: atualiza os valores de throughput provisionado da tabela especificada.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Para obter detalhes da API, consulte UpdateTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: atualiza os valores de throughput provisionado da tabela especificada.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Para obter detalhes da API, consulte UpdateTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`UpdateTimeToLive`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateTimeToLive`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Trabalhe com Streams e Time-to-Live

CLI

AWS CLI

Como atualizar as configurações de vida útil de uma tabela

O exemplo `update-time-to-live` a seguir habilita a vida útil na tabela especificada.

```
aws dynamodb update-time-to-live \
    --table-name MusicCollection \
    --time-to-live-specification Enabled=true,AttributeName=ttl
```

Saída:

```
{
    "TimeToLiveSpecification": {
        "Enabled": true,
        "AttributeName": "ttl"
    }
}
```

Para obter mais informações, consulte Vida útil no Guia do desenvolvedor do Amazon DynamoDB.

- Para obter detalhes da API, consulte UpdateTimeToLivena Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

Habilite o TTL em uma tabela existente do DynamoDB usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse enableTTL(final String tableName, final
 String attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(true)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();
```

```
        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

Desabilite o TTL em uma tabela existente do DynamoDB. usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse disableTTL(
        final String tableName, final String attributeName, final Region region)
 {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(false)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
```

```
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte UpdateTimeToLivea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Habilite a TTL em uma tabela existente do DynamoDB.

```
import { DynamoDBClient, UpdateTimeToLiveCommand } from "@aws-sdk/client-
dynamodb";

export const enableTTL = async (tableName, ttlAttribute, region = 'us-east-1') =>
 {

    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const params = {
        TableName: tableName,
        TimeToLiveSpecification: {
            Enabled: true,
            AttributeName: ttlAttribute
        }
    };

    try {
        const response = await client.send(new UpdateTimeToLiveCommand(params));
        if (response.$metadata.httpStatusCode === 200) {
```

```
            console.log(`TTL enabled successfully for table ${tableName}, using
 attribute name ${ttlAttribute}.`);
        } else {
            console.log(`Failed to enable TTL for table ${tableName}, response
 object: ${response}`);
        }
        return response;
    } catch (e) {
        console.error(`Error enabling TTL: ${e}`);
        throw e;
    }
};

// Example usage (commented out for testing)
// enableTTL('ExampleTable', 'exampleTtlAttribute');
```

Desabilite a TTL em uma tabela existente do DynamoDB.

```
import { DynamoDBClient, UpdateTimeToLiveCommand } from "@aws-sdk/client-
dynamodb";

export const disableTTL = async (tableName, ttlAttribute, region = 'us-east-1')
 => {

    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const params = {
        TableName: tableName,
        TimeToLiveSpecification: {
            Enabled: false,
            AttributeName: ttlAttribute
        }
    };

    try {
        const response = await client.send(new UpdateTimeToLiveCommand(params));
        if (response.$metadata.httpStatusCode === 200) {
            console.log(`TTL disabled successfully for table ${tableName}, using
 attribute name ${ttlAttribute}.`);
        } else {
```

```
            console.log(`Failed to disable TTL for table ${tableName}, response
 object: ${response}`);
        }
        return response;
    } catch (e) {
        console.error(`Error disabling TTL: ${e}`);
        throw e;
    }
};

// Example usage (commented out for testing)
// disableTTL('ExampleTable', 'exampleTtlAttribute');
```

- Para obter detalhes da API, consulte UpdateTimeToLivea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Habilite a TTL em uma tabela existente do DynamoDB.

```
import boto3


def enable_ttl(table_name, ttl_attribute_name):
    """
    Enables TTL on DynamoDB table for a given attribute name
        on success, returns a status code of 200
        on error, throws an exception

    :param table_name: Name of the DynamoDB table
    :param ttl_attribute_name: The name of the TTL attribute being provided to
 the table.
    """
    try:
        dynamodb = boto3.client("dynamodb")

        # Enable TTL on an existing DynamoDB table
        response = dynamodb.update_time_to_live(
            TableName=table_name,
            TimeToLiveSpecification={"Enabled": True, "AttributeName":
 ttl_attribute_name},
```

```
        )

        # In the returned response, check for a successful status code.
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            print("TTL has been enabled successfully.")
        else:
            print(
                f"Failed to enable TTL, status code {response['ResponseMetadata']
['HTTPStatusCode']}"
            )
        return response
    except Exception as ex:
        print("Couldn't enable TTL in table %s. Here's why: %s" % (table_name,
 ex))
        raise


# your values
enable_ttl("your-table-name", "expireAt")
```

Desabilite a TTL em uma tabela existente do DynamoDB.

```
import boto3


def disable_ttl(table_name, ttl_attribute_name):
    """
    Disables TTL on DynamoDB table for a given attribute name
        on success, returns a status code of 200
        on error, throws an exception

    :param table_name: Name of the DynamoDB table being modified
    :param ttl_attribute_name: The name of the TTL attribute being provided to
 the table.
    """
    try:
        dynamodb = boto3.client("dynamodb")

        # Enable TTL on an existing DynamoDB table
        response = dynamodb.update_time_to_live(
            TableName=table_name,
```

```
            TimeToLiveSpecification={"Enabled": False, "AttributeName":
 ttl_attribute_name},
        )

        # In the returned response, check for a successful status code.
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            print("TTL has been disabled successfully.")
        else:
            print(
                f"Failed to disable TTL, status code
 {response['ResponseMetadata']['HTTPStatusCode']}"
            )
    except Exception as ex:
        print("Couldn't disable TTL in table %s. Here's why: %s" % (table_name,
 ex))
        raise


# your values
disable_ttl("your-table-name", "expireAt")
```

- Para obter detalhes da API, consulte a UpdateTimeToLiveReferência da API AWS SDK for Python (Boto3).

Cenários para o uso do DynamoDB AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no DynamoDB com SDKs AWS . Estes cenários mostram como realizar tarefas específicas chamando várias funções dentro do DynamoDB ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Acelere as leituras do DynamoDB com o DAX usando um AWS SDK
- Trabalhe com cenários avançados do Índice Secundário Global do DynamoDB usando AWS Command Line Interface v2 da

- Criar uma aplicação para enviar dados para uma tabela do DynamoDB
- Compare vários valores com um único atributo no DynamoDB com um AWS SDK
- Atualize condicionalmente um item do DynamoDB com um TTL usando um AWS SDK
- Conecte-se a uma instância local do DynamoDB usando um AWS SDK
- Conte operadores de expressão no DynamoDB com um AWS SDK
- Crie uma API REST do API Gateway para rastrear COVID-19 dados
- Criar uma aplicação de mensageiro com o Step Functions
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Crie uma tabela do DynamoDB com um índice secundário global usando o AWS SDK
- Crie uma tabela do DynamoDB com configuração de taxa de transferência quente usando um AWS SDK
- Criar uma aplicação Web para monitorar dados do DynamoDB
- Criar uma aplicação de chat websocket com o API Gateway
- Crie um item do DynamoDB com um TTL usando um AWS SDK
- Crie e gerencie tabelas globais do DynamoDB Multi-Region com forte consistência usando um AWS SDK
- Crie e gerencie tabelas globais do DynamoDB demonstrando o MREC usando um AWS SDK
- Exclua dados do DynamoDB usando instruções DELETE do PartiQL com um AWS SDK
- Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK
- Conceitos básicos de bancos de dados NoSQL
- Insira dados do DynamoDB usando instruções INSERT do PartiQL com um AWS SDK
- Invocar uma função do Lambda em um navegador
- Gerencie os índices secundários globais do DynamoDB usando AWS Command Line Interface v2 da
- Gerencie políticas baseadas em recursos do DynamoDB usando AWS Command Line Interface v2 da
- Monitore o desempenho do Amazon DynamoDB usando um AWS SDK
- Execute operações avançadas de consulta do DynamoDB usando um AWS SDK
- Execute operações de lista no DynamoDB com um AWS SDK
- Execute operações de mapa no DynamoDB com um AWS SDK

- Execute operações de conjunto no DynamoDB com um AWS SDK
- Consulte uma tabela do DynamoDB usando lotes de instruções partiQL e um AWS SDK
- Consulte uma tabela do DynamoDB usando partiQL e um AWS SDK
- Consulte uma tabela do DynamoDB usando um índice secundário global com um AWS SDK
- Consulte uma tabela do DynamoDB usando uma condição begins_with com um AWS SDK
- Consulte uma tabela do DynamoDB usando um intervalo de datas na chave de classificação com um AWS SDK
- Consulte uma tabela do DynamoDB com uma expressão de filtro complexa com um AWS SDK
- Consulte uma tabela do DynamoDB com uma expressão de filtro dinâmico com um AWS SDK
- Consulte uma tabela do DynamoDB com uma expressão de filtro e limite com uma AWS SDK
- Consulte uma tabela do DynamoDB com atributos aninhados usando um AWS SDK
- Consulte uma tabela do DynamoDB com paginação usando um AWS SDK
- Consulte uma tabela do DynamoDB com leituras altamente consistentes usando um AWS SDK
- Consulte dados do DynamoDB usando instruções PARTIQL SELECT com um AWS SDK
- Consulte uma tabela do DynamoDB para itens TTL usando um AWS SDK
- Consulte tabelas do DynamoDB usando padrões de data e hora com um AWS SDK
- Salve EXIF e outras informações da imagem usando um AWS SDK
- Configure o controle de Attribute-Based acesso para o DynamoDB usando AWS Command Line Interface v2 da
- Entenda a ordem da expressão de atualização no DynamoDB com um AWS SDK
- Atualize uma configuração de tabela do DynamoDB com taxa de transferência quente usando um AWS SDK
- Atualizar um item do DynamoDB com um TTL usando um AWS SDK
- Atualize os dados do DynamoDB usando instruções PARTIQL UPDATE com um AWS SDK
- Usar o API Gateway para invocar uma função do Lambda
- Usar Step Functions para invocar funções do Lambda
- Use um modelo de documento para o DynamoDB usando um AWS SDK
- Use um modelo de persistência de objetos de alto nível para o DynamoDB usando um AWS SDK
- Use operações de contador atômico no DynamoDB com um AWS SDK
- Use operações condicionais no DynamoDB com um AWS SDK
- Use nomes de atributos de expressão no DynamoDB com um AWS SDK

- Usar eventos programados para chamar uma função do Lambda
- Trabalhe com índices secundários locais do DynamoDB usando AWS Command Line Interface v2 da
- Trabalhe com o DynamoDB Streams e use Time-to-Live AWS Command Line Interface v2 da
- Trabalhe com tabelas globais do DynamoDB e replicação multirregional com consistência eventual (MREC) usando AWS Command Line Interface v2 da
- Trabalhe com a marcação de recursos do DynamoDB usando AWS Command Line Interface v2 da
- Trabalhe com a criptografia de tabelas do DynamoDB usando AWS Command Line Interface v2 da

Acelere as leituras do DynamoDB com o DAX usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar e gravar dados em uma tabela com os clientes DAX e SDK.
- Obter, consultar e verificar a tabela com ambos os clientes e comparar a respectiva performance.

Para obter mais informações, consulte Desenvolver com o cliente do DynamoDB Accelerator.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma tabela com o cliente do DAX ou Boto3.

```
import boto3


def create_dax_table(dyn_resource=None):
    """
    Creates a DynamoDB table.

    :param dyn_resource: Either a Boto3 or DAX resource.
```

```
    :return: The newly created table.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table_name = "TryDaxTable"
    params = {
        "TableName": table_name,
        "KeySchema": [
            {"AttributeName": "partition_key", "KeyType": "HASH"},
            {"AttributeName": "sort_key", "KeyType": "RANGE"},
        ],
        "AttributeDefinitions": [
            {"AttributeName": "partition_key", "AttributeType": "N"},
            {"AttributeName": "sort_key", "AttributeType": "N"},
        ],
        "BillingMode": "PAY_PER_REQUEST",
    }
    table = dyn_resource.create_table(**params)
    print(f"Creating {table_name}...")
    table.wait_until_exists()
    return table


if __name__ == "__main__":
    dax_table = create_dax_table()
    print(f"Created table.")
```

Grave dados de teste na tabela.

```
import boto3


def write_data_to_dax_table(key_count, item_size, dyn_resource=None):
    """
    Writes test data to the demonstration table.

    :param key_count: The number of partition and sort keys to use to populate
 the
                      table. The total number of items is key_count * key_count.
    :param item_size: The size of non-key data for each test item.
    :param dyn_resource: Either a Boto3 or DAX resource.
```

```
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    some_data = "X" * item_size

    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.put_item(
                Item={
                    "partition_key": partition_key,
                    "sort_key": sort_key,
                    "some_data": some_data,
                }
            )
            print(f"Put item ({partition_key}, {sort_key}) succeeded.")


if __name__ == "__main__":
    write_key_count = 10
    write_item_size = 1000
    print(
        f"Writing {write_key_count*write_key_count} items to the table. "
        f"Each item is {write_item_size} characters."
    )
    write_data_to_dax_table(write_key_count, write_item_size)
```

Obtenha itens para várias iterações associadas ao cliente do DAX e ao cliente do Boto3 e relate o tempo gasto para cada um.

```
import argparse
import sys
import time
import amazondax
import boto3


def get_item_test(key_count, iterations, dyn_resource=None):
    """
    Gets items from the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
```

```
    and reported.

    :param key_count: The number of items to get from the table in each
 iteration.
    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    start = time.perf_counter()
    for _ in range(iterations):
        for partition_key in range(1, key_count + 1):
            for sort_key in range(1, key_count + 1):
                table.get_item(
                    Key={"partition_key": partition_key, "sort_key": sort_key}
                )
                print(".", end="")
                sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_key_count = 10
    test_iterations = 50
    if args.endpoint_url:
        print(
            f"Getting each item from the table {test_iterations} times, "
            f"using the DAX client."
        )
```

```
        # Use a with statement so the DAX client closes the cluster after
completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url)
as dax:
            test_start, test_end = get_item_test(
                test_key_count, test_iterations, dyn_resource=dax
            )
    else:
        print(
            f"Getting each item from the table {test_iterations} times, "
            f"using the Boto3 client."
        )
        test_start, test_end = get_item_test(test_key_count, test_iterations)
    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/ test_iterations}."
    )
```

Consulte a tabela para várias iterações associadas ao cliente do DAX e ao cliente do Boto3 e relate o tempo gasto para cada um.

```
import argparse
import time
import sys
import amazondax
import boto3
from boto3.dynamodb.conditions import Key


def query_test(partition_key, sort_keys, iterations, dyn_resource=None):
    """
    Queries the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param partition_key: The partition key value to use in the query. The query
                          returns items that have partition keys equal to this
value.
    :param sort_keys: The range of sort key values for the query. The query
returns
                      items that have sort key values between these two values.
    :param iterations: The number of iterations to run.
```

```
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    key_condition_expression = Key("partition_key").eq(partition_key) & Key(
        "sort_key"
    ).between(*sort_keys)

    start = time.perf_counter()
    for _ in range(iterations):
        table.query(KeyConditionExpression=key_condition_expression)
        print(".", end="")
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_partition_key = 5
    test_sort_keys = (2, 9)
    test_iterations = 100
    if args.endpoint_url:
        print(f"Querying the table {test_iterations} times, using the DAX
 client.")
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url)
 as dax:
            test_start, test_end = query_test(
```

```
                test_partition_key, test_sort_keys, test_iterations,
 dyn_resource=dax
            )
    else:
        print(f"Querying the table {test_iterations} times, using the Boto3
 client.")
        test_start, test_end = query_test(
            test_partition_key, test_sort_keys, test_iterations
        )

    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/test_iterations}."
    )
```

Verifique a tabela para várias iterações associadas ao cliente do DAX e ao cliente do Boto3 e relate o tempo gasto para cada um.

```
import argparse
import time
import sys
import amazondax
import boto3


def scan_test(iterations, dyn_resource=None):
    """
    Scans the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    start = time.perf_counter()
    for _ in range(iterations):
        table.scan()
```

```
        print(".", end="")
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_iterations = 100
    if args.endpoint_url:
        print(f"Scanning the table {test_iterations} times, using the DAX
 client.")
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url)
 as dax:
            test_start, test_end = scan_test(test_iterations, dyn_resource=dax)
    else:
        print(f"Scanning the table {test_iterations} times, using the Boto3
 client.")
        test_start, test_end = scan_test(test_iterations)
    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/test_iterations}."
    )
```

Exclua a tabela.

```
import boto3


def delete_dax_table(dyn_resource=None):
```

```
    """
    Deletes the demonstration table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    table.delete()

    print(f"Deleting {table.name}...")
    table.wait_until_not_exists()


if __name__ == "__main__":
    delete_dax_table()
    print("Table deleted!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTable
  - DeleteTable
  - GetItem
  - PutItem
  - Query
  - Scan

Trabalhe com cenários avançados do Índice Secundário Global do DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como trabalhar com configurações de índice secundário global (GSI).

- Crie uma tabela com vários GSIs.
- Crie uma tabela com capacidade e GSI sob demanda.
- Insira itens em uma tabela com vários GSIs.

- Consulte vários GSIs com condições diferentes.

Bash

AWS CLI com script Bash

Crie uma tabela com vários GSIs.

```
# Create a table with multiple GSIs
aws dynamodb create-table \
    --table-name MusicLibrary \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=AlbumTitle,AttributeType=S \
        AttributeName=Genre,AttributeType=S \
        AttributeName=Year,AttributeType=N \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"AlbumIndex\",
                \"KeySchema\": [{\"AttributeName\":\"AlbumTitle\",\"KeyType\":
\"HASH\"}],
                \"Projection\": {\"ProjectionType\":\"ALL\"}
            },
            {
                \"IndexName\": \"GenreYearIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"Year\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"INCLUDE\",
\"NonKeyAttributes\":[\"Artist\",\"SongTitle\"]}
            }
        ]"
```

Crie uma tabela com capacidade e GSI sob demanda.

```
# Create a table with on-demand capacity and GSI
aws dynamodb create-table \
    --table-name MusicOnDemand \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=Genre,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"GenreIndex\",
                \"KeySchema\": [{\"AttributeName\":\"Genre\",\"KeyType\":\"HASH
\"}],
                \"Projection\": {\"ProjectionType\":\"ALL\"}
            }
        ]"
```

Insira itens em uma tabela com vários GSIs.

```
# Add items to MusicLibrary table
aws dynamodb put-item \
    --table-name MusicLibrary \
    --item '{
        "Artist": {"S": "The Beatles"},
        "SongTitle": {"S": "Hey Jude"},
        "AlbumTitle": {"S": "Past Masters"},
        "Genre": {"S": "Rock"},
        "Year": {"N": "1968"}
    }'

aws dynamodb put-item \
    --table-name MusicLibrary \
    --item '{
        "Artist": {"S": "Miles Davis"},
        "SongTitle": {"S": "So What"},
        "AlbumTitle": {"S": "Kind of Blue"},
        "Genre": {"S": "Jazz"},
        "Year": {"N": "1959"}
```

```
    }'
```

Consulte os itens de uma tabela com vários GSIs.

```
# Query the AlbumIndex GSI
echo "Querying AlbumIndex GSI:"
aws dynamodb query \
    --table-name MusicLibrary \
    --index-name AlbumIndex \
    --key-condition-expression "AlbumTitle = :album" \
    --expression-attribute-values '{":album":{"S":"Kind of Blue"}}'

# Query the GenreYearIndex GSI with a range condition
echo "Querying GenreYearIndex GSI with range condition:"
aws dynamodb query \
    --table-name MusicLibrary \
    --index-name GenreYearIndex \
    --key-condition-expression "Genre = :genre AND #yr > :year" \
    --expression-attribute-names '{"#yr": "Year"}' \
    --expression-attribute-values '{":genre":{"S":"Rock"},":year":{"N":"1965"}}'
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - PutItem
  - Consulta

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

Os exemplos de código a seguir mostram como criar uma aplicação que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

Java

SDK para Java 2.x

Mostra como criar uma aplicação Web dinâmica que envia dados usando a API Java do Amazon DynamoDB e envia uma mensagem de texto usando a API Java do Amazon Simple Notification Service.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Este exemplo mostra como criar uma aplicação que permite que os usuários enviem dados para uma tabela do Amazon DynamoDB e enviem uma mensagem de texto ao administrador usando o Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Compare vários valores com um único atributo no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como comparar vários valores com um único atributo no DynamoDB.

- Use o operador IN para comparar vários valores com um único atributo.
- Compare o operador IN com várias condições OR.
- Saiba quais são os benefícios de desempenho e complexidade de expressão oferecidos pelo IN.

Java

SDK para Java 2.x

Compare vários valores com um único atributo no DynamoDB usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

    /**
     * Queries a table using the IN operator to compare multiple values with a
 single attribute.
     *
     * <p>This method demonstrates how to use the IN operator in a filter
 expression
     * to match an attribute against multiple values.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareMultipleValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {
```

```
        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Queries a table using multiple OR conditions to compare multiple values
with a single attribute.
     *
     * <p>This method demonstrates the alternative approach to using the IN
operator,
     * by using multiple OR conditions.
```

```
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareWithOrConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for OR conditions
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the OR conditions
        StringBuilder orConditions = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                orConditions.append(" OR ");
            }
            orConditions.append("#attrName = :val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
```

```
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression(orConditions.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Compares the performance of using the IN operator versus multiple OR
conditions.
     *
     * <p>This method demonstrates the performance difference between using the
IN operator
     * and using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return Map containing the performance comparison results
     */
    public static Map<String, Object> comparePerformance(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Measure performance of IN operator
            long inStartTime = System.nanoTime();
            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long inEndTime = System.nanoTime();
            long inDuration = inEndTime - inStartTime;
```

```
            // Measure performance of OR conditions
            long orStartTime = System.nanoTime();
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long orEndTime = System.nanoTime();
            long orDuration = orEndTime - orStartTime;

            // Record results
            results.put("inOperatorDuration", inDuration);
            results.put("orConditionsDuration", orDuration);
            results.put("inOperatorItems", inResponse.count());
            results.put("orConditionsItems", orResponse.count());
            results.put("inOperatorExpression", "IN operator with " +
valuesList.size() + " values");
            results.put("orConditionsExpression", valuesList.size() + " OR
conditions");
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Scans a table using the IN operator with a large number of values.
     *
     * <p>This method demonstrates how to use the IN operator with a large number
of values,
     * which can help stay within the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The scan response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithLargeInClause(
```

```
        DynamoDbClient dynamoDbClient, String tableName, String attributeName,
List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }

    /**
     * Generates a list of sample values for testing.
     *
     * <p>Helper method to generate a list of sample values for testing.
     *
     * @param valueType The type of values to generate (string, number, or
boolean)
     * @param count The number of values to generate
```

```
     * @return List of generated attribute values
     */
    public static List<AttributeValue> generateSampleValues(String valueType, int
count) {
        List<AttributeValue> values = new ArrayList<>();

        for (int i = 0; i < count; i++) {
            AttributeValue value;

            switch (valueType.toLowerCase(Locale.ROOT)) {
                case "string":
                    value = AttributeValue.builder().s("Value" + i).build();
                    break;
                case "number":
                    value =
AttributeValue.builder().n(String.valueOf(i)).build();
                    break;
                case "boolean":
                    value = AttributeValue.builder().bool(i % 2 == 0).build();
                    break;
                default:
                    throw new IllegalArgumentException("Unsupported value type: "
+ valueType);
            }

            values.add(value);
        }

        return values;
    }
```

Exemplo de uso da comparação de vários valores com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
        System.out.println("Demonstrating how to compare multiple values with a
single attribute in DynamoDB");

        try {
            // Example 1: Using the IN operator
            System.out.println("\nExample 1: Using the IN operator");
            List<AttributeValue> categories = List.of(
```

```
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Computers").build(),
                AttributeValue.builder().s("Accessories").build());

            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + inResponse.count() + " items using IN
operator");
            System.out.println("Items: " + inResponse.items());

            // Example 2: Using multiple OR conditions
            System.out.println("\nExample 2: Using multiple OR conditions");
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + orResponse.count() + " items using OR
conditions");
            System.out.println("Items: " + orResponse.items());

            // Example 3: Performance comparison
            System.out.println("\nExample 3: Performance comparison");
            Map<String, Object> perfComparison = comparePerformance(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            if ((boolean) perfComparison.get("success")) {
                System.out.println("IN operator duration: " +
perfComparison.get("inOperatorDuration") + " ns");
```

```
                System.out.println("OR conditions duration: " +
perfComparison.get("orConditionsDuration") + " ns");
                System.out.println("IN operator found " +
perfComparison.get("inOperatorItems") + " items");
                System.out.println("OR conditions found " +
perfComparison.get("orConditionsItems") + " items");
                System.out.println("Expression complexity comparison:");
                System.out.println("  IN operator: " +
perfComparison.get("inOperatorExpression"));
                System.out.println("  OR conditions: " +
perfComparison.get("orConditionsExpression"));
            } else {
                System.out.println("Performance comparison failed: " +
perfComparison.get("error"));
            }

            // Example 4: Using IN with a large number of values
            System.out.println("\nExample 4: Using IN with a large number of
values");
            List<AttributeValue> productIds = generateSampleValues("string", 20);

            ScanResponse largeInResponse = scanWithLargeInClause(dynamoDbClient,
tableName, "ProductId", productIds);

            System.out.println(
                "Found " + largeInResponse.count() + " items using IN with " +
productIds.size() + " values");

            // Explain the benefits of using IN
            System.out.println("\nKey points about using the IN operator in
DynamoDB:");
            System.out.println("1. The IN operator allows comparing a single
attribute against multiple values");
            System.out.println("2. IN is more concise than using multiple OR
conditions");
            System.out.println("3. IN counts as only 1 operator regardless of the
number of values");
            System.out.println("4. Multiple OR conditions count as 1 operator per
condition plus 1 per OR");
            System.out.println("5. Using IN helps stay within the 300 operator
limit for complex expressions");
            System.out.println("6. IN can be used in filter expressions and
condition expressions");
```

```
            System.out.println("7. The IN operator supports up to 100 comparison
 values");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - Query
  - Scan

JavaScript

SDK para JavaScript (v3)

Compare vários valores com um único atributo usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  ScanCommand,
  QueryCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Query or scan a DynamoDB table to find items where an attribute matches any
 value from a list.
 *
 * This function demonstrates the use of the IN operator to compare a single
 attribute
 * against multiple possible values, which is more efficient than using multiple
 OR conditions.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} attributeName - The name of the attribute to compare against
 the values list
 * @param {Array} valuesList - List of values to compare the attribute against
```

```
 * @param {string} [partitionKeyName] - Optional name of the partition key
 attribute for query operations
 * @param {string} [partitionKeyValue] - Optional value of the partition key to
 query
 * @returns {Promise<Object>} - The response from DynamoDB containing the
 matching items
 */
async function compareMultipleValues(
  config,
  tableName,
  attributeName,
  valuesList,
  partitionKeyName,
  partitionKeyValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the filter expression using the IN operator
  const filterExpression = `${attributeName} IN (${valuesList.map((_, index) =>
 `:val${index}`).join(', ')})`;

  // Create expression attribute values for the values list
  const expressionAttributeValues = valuesList.reduce((acc, val, index) => {
    acc[`:val${index}`] = val;
    return acc;
  }, {});

  // If partition key is provided, perform a query operation
  if (partitionKeyName && partitionKeyValue) {
    const keyCondition = `${partitionKeyName} = :partitionKey`;
    expressionAttributeValues[':partitionKey'] = partitionKeyValue;

    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        KeyConditionExpression: keyCondition,
        FilterExpression: filterExpression,
```

```
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous
query
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new QueryCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  } else {
    // Otherwise, perform a scan operation
    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous scan
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }
```

```
      const response = await docClient.send(new ScanCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  }
}

/**
 * Alternative implementation using multiple OR conditions instead of the IN
 operator.
 *
 * This function is provided for comparison to show why using the IN operator is
 preferable.
 * With many values, this approach becomes verbose and less efficient.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} attributeName - The name of the attribute to compare against
 the values list
 * @param {Array} valuesList - List of values to compare the attribute against
 * @param {string} [partitionKeyName] - Optional name of the partition key
 attribute for query operations
 * @param {string} [partitionKeyValue] - Optional value of the partition key to
 query
 * @returns {Promise<Object>} - The response from DynamoDB containing the
 matching items
 */
async function compareWithOrConditions(
  config,
  tableName,
  attributeName,
  valuesList,
```

```
  partitionKeyName,
  partitionKeyValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // If no values provided, return empty result
  if (!valuesList || valuesList.length === 0) {
    return {
      Items: [],
      Count: 0
    };
  }

  // Create the filter expression using multiple OR conditions
  const filterConditions = valuesList.map((_, index) => `${attributeName} = :val
${index}`);
  const filterExpression = filterConditions.join(' OR ');

  // Create expression attribute values for the values list
  const expressionAttributeValues = valuesList.reduce((acc, val, index) => {
    acc[`:val${index}`] = val;
    return acc;
  }, {});

  // If partition key is provided, perform a query operation
  if (partitionKeyName && partitionKeyValue) {
    const keyCondition = `${partitionKeyName} = :partitionKey`;
    expressionAttributeValues[':partitionKey'] = partitionKeyValue;

    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        KeyConditionExpression: keyCondition,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };
```

```
      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous
query
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new QueryCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  } else {
    // Otherwise, perform a scan operation
    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous scan
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new ScanCommand(params));

      // Add the items from this page to our collection
```

```
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  }
}
```

Exemplo de uso da comparação de vários valores com AWS SDK para JavaScript.

```
/**
 * Example of how to use the compareMultipleValues function.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const attributeName = "Category";
  const valuesList = ["Electronics", "Computers", "Accessories"];

  console.log(`Searching for products in any of these categories:
 ${valuesList.join(', ')}`);

  try {
    // Using the IN operator (recommended approach)
    console.log("\nApproach 1: Using the IN operator");
    const response = await compareMultipleValues(
      config,
      tableName,
      attributeName,
      valuesList
    );

    console.log(`Found ${response.Count} products in the specified categories`);
```

```
    // Using multiple OR conditions (alternative approach)
    console.log("\nApproach 2: Using multiple OR conditions");
    const response2 = await compareWithOrConditions(
      config,
      tableName,
      attributeName,
      valuesList
    );

    console.log(`Found ${response2.Count} products in the specified categories`);

    // Example with a query operation
    console.log("\nQuerying a specific manufacturer's products in multiple
 categories");
    const partitionKeyName = "Manufacturer";
    const partitionKeyValue = "Acme";

    const response3 = await compareMultipleValues(
      config,
      tableName,
      attributeName,
      valuesList,
      partitionKeyName,
      partitionKeyValue
    );

    console.log(`Found ${response3.Count} Acme products in the specified
 categories`);

    // Explain the benefits of using the IN operator
    console.log("\nBenefits of using the IN operator:");
    console.log("1. More concise expression compared to multiple OR conditions");
    console.log("2. Better readability and maintainability");
    console.log("3. Potentially better performance with large value lists");
    console.log("4. Simpler code that's less prone to errors");
    console.log("5. Easier to modify when adding or removing values");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - Query
  - Scan

Python

SDK para Python (Boto3)

Compare vários valores com um único atributo usando AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key
from typing import Any, Dict, List, Optional


def compare_multiple_values(
    table_name: str,
    attribute_name: str,
    values_list: List[Any],
    partition_key_name: Optional[str] = None,
    partition_key_value: Optional[str] = None,
) -> Dict[str, Any]:
    """
    Query or scan a DynamoDB table to find items where an attribute matches any
 value from a list.

    This function demonstrates the use of the IN operator to compare a single
 attribute
    against multiple possible values, which is more efficient than using multiple
 OR conditions.

    Args:
        table_name (str): The name of the DynamoDB table.
        attribute_name (str): The name of the attribute to compare against the
 values list.
        values_list (List[Any]): List of values to compare the attribute against.
        partition_key_name (Optional[str]): The name of the partition key
 attribute for query operations.
        partition_key_value (Optional[str]): The value of the partition key to
 query.
```

```
    Returns:
        Dict[str, Any]: The response from DynamoDB containing the matching items.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create the filter expression using the is_in method
    filter_expression = Attr(attribute_name).is_in(values_list)

    # If partition key is provided, perform a query operation
    if partition_key_name and partition_key_value:
        key_condition = Key(partition_key_name).eq(partition_key_value)
        response = table.query(
            KeyConditionExpression=key_condition,
 FilterExpression=filter_expression
        )
    else:
        # Otherwise, perform a scan operation
        response = table.scan(FilterExpression=filter_expression)

    # Handle pagination if there are more results
    items = response.get("Items", [])
    while "LastEvaluatedKey" in response:
        if partition_key_name and partition_key_value:
            response = table.query(
                KeyConditionExpression=key_condition,
                FilterExpression=filter_expression,
                ExclusiveStartKey=response["LastEvaluatedKey"],
            )
        else:
            response = table.scan(
                FilterExpression=filter_expression,
 ExclusiveStartKey=response["LastEvaluatedKey"]
            )
        items.extend(response.get("Items", []))

    # Return the complete result
    return {"Items": items, "Count": len(items)}


def compare_with_or_conditions(
    table_name: str,
    attribute_name: str,
```

```
    values_list: List[Any],
    partition_key_name: Optional[str] = None,
    partition_key_value: Optional[str] = None,
) -> Dict[str, Any]:
    """
    Alternative implementation using multiple OR conditions instead of the IN
 operator.

    This function is provided for comparison to show why using the IN operator is
 preferable.
    With many values, this approach becomes verbose and less efficient.

    Args:
        table_name (str): The name of the DynamoDB table.
        attribute_name (str): The name of the attribute to compare against the
 values list.
        values_list (List[Any]): List of values to compare the attribute against.
        partition_key_name (Optional[str]): The name of the partition key
 attribute for query operations.
        partition_key_value (Optional[str]): The value of the partition key to
 query.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the matching items.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create a filter expression with multiple OR conditions
    filter_expression = None
    for value in values_list:
        condition = Attr(attribute_name).eq(value)
        if filter_expression is None:
            filter_expression = condition
        else:
            filter_expression = filter_expression | condition

    # If partition key is provided, perform a query operation
    if partition_key_name and partition_key_value and filter_expression:
        key_condition = Key(partition_key_name).eq(partition_key_value)
        response = table.query(
            KeyConditionExpression=key_condition,
 FilterExpression=filter_expression
```

```
        )
    elif filter_expression:
        # Otherwise, perform a scan operation
        response = table.scan(FilterExpression=filter_expression)
    else:
        # Return empty response if no values provided
        return {"Items": [], "Count": 0}

    # Handle pagination if there are more results
    items = response.get("Items", [])
    while "LastEvaluatedKey" in response:
        if partition_key_name and partition_key_value:
            response = table.query(
                KeyConditionExpression=key_condition,
                FilterExpression=filter_expression,
                ExclusiveStartKey=response["LastEvaluatedKey"],
            )
        else:
            response = table.scan(
                FilterExpression=filter_expression,
 ExclusiveStartKey=response["LastEvaluatedKey"]
            )
        items.extend(response.get("Items", []))

    # Return the complete result
    return {"Items": items, "Count": len(items)}
```

Exemplo de uso da comparação de vários valores com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the compare_multiple_values function."""
    # Example parameters
    table_name = "Products"
    attribute_name = "Category"
    values_list = ["Electronics", "Computers", "Accessories"]

    print(f"Searching for products in any of these categories: {values_list}")

    # Using the IN operator (recommended approach)
    print("\nApproach 1: Using the IN operator")
```

```
    response = compare_multiple_values(
        table_name=table_name, attribute_name=attribute_name,
values_list=values_list
    )

    print(f"Found {response['Count']} products in the specified categories")

    # Using multiple OR conditions (alternative approach)
    print("\nApproach 2: Using multiple OR conditions")
    response2 = compare_with_or_conditions(
        table_name=table_name, attribute_name=attribute_name,
values_list=values_list
    )

    print(f"Found {response2['Count']} products in the specified categories")

    # Example with a query operation
    print("\nQuerying a specific manufacturer's products in multiple categories")
    partition_key_name = "Manufacturer"
    partition_key_value = "Acme"

    response3 = compare_multiple_values(
        table_name=table_name,
        attribute_name=attribute_name,
        values_list=values_list,
        partition_key_name=partition_key_name,
        partition_key_value=partition_key_value,
    )

    print(f"Found {response3['Count']} Acme products in the specified
categories")

    # Explain the benefits of using the IN operator
    print("\nBenefits of using the IN operator:")
    print("1. More concise expression compared to multiple OR conditions")
    print("2. Better readability and maintainability")
    print("3. Potentially better performance with large value lists")
    print("4. Simpler code that's less prone to errors")
    print("5. Easier to modify when adding or removing values")
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - Query
  - Scan

Atualize condicionalmente um item do DynamoDB com um TTL usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar condicionalmente a TTL de um item.

Java

SDK para Java 2.x

Atualize a TTL em um item do DynamoDB existente em uma tabela, com uma condição.

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Map;
import java.util.Optional;

/**
 * Updates an item in a DynamoDB table with TTL attributes using a conditional
 expression.
 * This class demonstrates how to conditionally update TTL expiration timestamps.
 */
public class UpdateTTLConditional {

    private static final String USAGE =
        """
            Usage:
```

```
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
                sortKey - The name of the sort key. Also known as the range
attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table
is located in. (Default: us-east-1)
            """;
   private static final int DAYS_TO_EXPIRE = 90;
   private static final int SECONDS_PER_DAY = 24 * 60 * 60;
   private static final String PRIMARY_KEY_ATTR = "primaryKey";
   private static final String SORT_KEY_ATTR = "sortKey";
   private static final String UPDATED_AT_ATTR = "updatedAt";
   private static final String EXPIRE_AT_ATTR = "expireAt";
   private static final String UPDATE_EXPRESSION = "SET " + UPDATED_AT_ATTR +
"=:c, " + EXPIRE_AT_ATTR + "=:e";
   private static final String CONDITION_EXPRESSION = EXPIRE_AT_ATTR + " > :c";
   private static final String SUCCESS_MESSAGE = "%s UpdateItem operation with
TTL successful.";
   private static final String CONDITION_FAILED_MESSAGE = "Condition check
failed. The item may have already expired.";
   private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon
DynamoDB table \"%s\" can't be found.";

   private final DynamoDbClient dynamoDbClient;

   /**
    * Constructs an UpdateTTLConditional with a default DynamoDB client.
    */
   public UpdateTTLConditional() {
       this.dynamoDbClient = null;
   }

   /**
    * Constructs an UpdateTTLConditional with the specified DynamoDB client.
    *
    * @param dynamoDbClient The DynamoDB client to use
    */
   public UpdateTTLConditional(final DynamoDbClient dynamoDbClient) {
       this.dynamoDbClient = dynamoDbClient;
   }
```

```
    /**
     * Main method to demonstrate conditionally updating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new UpdateTTLConditional().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and conditionally update an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this
code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] :
null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
```

```
        final Map<String, AttributeValue> keyMap = Map.of(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKey).build(),
            SORT_KEY_ATTR, AttributeValue.builder().s(sortKey).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = Map.of(
            ":c",
 AttributeValue.builder().n(String.valueOf(currentTime)).build(),
            ":e",
 AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .conditionExpression(CONDITION_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return 0;
        } catch (ConditionalCheckFailedException e) {
            System.err.println(CONDITION_FAILED_MESSAGE);
            throw e;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Atualize a TTL em um item do DynamoDB existente em uma tabela, com uma condição.

```
import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const updateItemConditional = async (tableName, partitionKey, sortKey,
 region = 'us-east-1', newAttribute = 'default-value') => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);

    const params = {
        TableName: tableName,
        Key: marshall({
            artist: partitionKey,
            album: sortKey
        }),
        UpdateExpression: "SET newAttribute = :newAttribute",
        ConditionExpression: "expireAt > :expiration",
        ExpressionAttributeValues: marshall({
            ':newAttribute': newAttribute,
            ':expiration': currentTime
        }),
        ReturnValues: "ALL_NEW"
    };

    try {
        const response = await client.send(new UpdateItemCommand(params));
        const responseData = unmarshall(response.Attributes);
        console.log("Item updated successfully: ", responseData);
        return responseData;
    } catch (error) {
        if (error.name === "ConditionalCheckFailedException") {
            console.log("Condition check failed: Item's 'expireAt' is expired.");
        } else {
            console.error("Error updating item: ", error);
        }
```

```
        throw error;
    }
};

// Example usage (commented out for testing)
// updateItemConditional('your-table-name', 'your-partition-key-value', 'your-
sort-key-value');
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Atualize a TTL em um item do DynamoDB existente em uma tabela, com uma condição.

```
from datetime import datetime, timedelta

import boto3
from botocore.exceptions import ClientError


def update_dynamodb_item_ttl(table_name, region, primary_key, sort_key,
 ttl_attribute):
    """
    Updates an existing record in a DynamoDB table with a new or updated TTL
 attribute.

    :param table_name: Name of the DynamoDB table
    :param region: AWS Region of the table - example `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :param ttl_attribute: name of the TTL attribute in the target DynamoDB table
    :return:
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Generate updated TTL in epoch second format
```

```
        updated_expiration_time = int((datetime.now() +
 timedelta(days=90)).timestamp())

        # Define the update expression for adding/updating a new attribute
        update_expression = "SET newAttribute = :val1"

        # Define the condition expression for checking if 'expireAt' is not
 expired
        condition_expression = "expireAt > :val2"

        # Define the expression attribute values
        expression_attribute_values = {":val1": ttl_attribute, ":val2":
 updated_expiration_time}

        response = table.update_item(
            Key={"primaryKey": primary_key, "sortKey": sort_key},
            UpdateExpression=update_expression,
            ConditionExpression=condition_expression,
            ExpressionAttributeValues=expression_attribute_values,
        )

        print("Item updated successfully.")
        return response["ResponseMetadata"]["HTTPStatusCode"]  # Ideally a 200 OK
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            print("Condition check failed: Item's 'expireAt' is expired.")
        else:
            print(f"Error updating item: {e}")
    except Exception as e:
        print(f"Error updating item: {e}")


# replace with your values
update_dynamodb_item_ttl(
    "your-table-name",
    "us-east-1",
    "your-partition-key-value",
    "your-sort-key-value",
    "your-ttl-attribute-value",
)
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Conecte-se a uma instância local do DynamoDB usando um AWS SDK

O exemplo de código a seguir mostra como substituir uma URL de endpoint para se conectar a uma implantação de desenvolvimento local do DynamoDB e de um SDK. AWS

Para obter mais informações, consulte DynamoDB Local.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// Lists your tables from a local DynamoDB instance by setting the SDK Config's
/// endpoint_url and test_credentials.
#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();

    let config = aws_config::defaults(aws_config::BehaviorVersion::latest())
        .test_credentials()
        // DynamoDB run locally uses port 8000 by default.
        .endpoint_url("http://localhost:8000")
        .load()
        .await;
    let dynamodb_local_config =
 aws_sdk_dynamodb::config::Builder::from(&config).build();

    let client = aws_sdk_dynamodb::Client::from_conf(dynamodb_local_config);

    let list_resp = client.list_tables().send().await;
    match list_resp {
        Ok(resp) => {
```

```
            println!("Found {} tables", resp.table_names().len());
            for name in resp.table_names() {
                println!("  {}", name);
            }
        }
        Err(err) => eprintln!("Failed to list local dynamodb tables: {err:?}"),
    }
}
```

Conte operadores de expressão no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como contar operadores de expressão no DynamoDB.

- Entenda o limite de 300 operadores do DynamoDB.
- Conte operadores em expressões complexas.
- Otimize as expressões para permanecer dentro dos limites.

Java

SDK para Java 2.x

Demonstre a contagem de operadores de expressão usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

    /**
     * Creates a complex filter expression with a specified number of conditions.
     *
     * <p>This method demonstrates how to generate a complex expression with
     * a specific number of operators to test the 300 operator limit.
     *
```

```
    * @param conditionsCount Number of conditions to include
    * @param useAnd Whether to use AND (true) or OR (false) between conditions
    * @return Map containing the filter expression, attribute values, and
operator count
    */
   public static Map<String, Object> createComplexFilterExpression(int
conditionsCount, boolean useAnd) {
       // Initialize the expression parts and attribute values
       StringBuilder filterExpression = new StringBuilder();
       Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

       // Generate the specified number of conditions
       for (int i = 0; i < conditionsCount; i++) {
           // Add the operator between conditions (except for the first one)
           if (i > 0) {
               filterExpression.append(useAnd ? " AND " : " OR ");
           }

           // Alternate between different comparison operators for variety
           String valueKey = ":val" + i;

           switch (i % 5) {
               case 0:
                   filterExpression.append("attribute").append(i).append(" =
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey, AttributeValue.builder().s("value" +
i).build());
                   break;
               case 1:
                   filterExpression.append("attribute").append(i).append(" >
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                   break;
               case 2:
                   filterExpression.append("attribute").append(i).append(" <
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey,
                       AttributeValue.builder().n(String.valueOf(i *
10)).build());
                   break;
```

```
                case 3:
                    filterExpression
                        .append("contains(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("substring" +
i).build());
                    break;
                case 4:
                    filterExpression
                        .append("attribute_exists(attribute")
                        .append(i)
                        .append(")");
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each condition has 1 operator (=, >, <, contains, attribute_exists)
        // Each AND or OR between conditions is 1 operator
        int operatorCount = conditionsCount + (conditionsCount > 0 ?
conditionsCount - 1 : 0);

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("filterExpression", filterExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
     * Creates a complex update expression with a specified number of operations.
     *
     * <p>This method demonstrates how to generate a complex update expression
with
```

```
     * a specific number of operators to test the 300 operator limit.
     *
     * @param operationsCount Number of operations to include
     * @return Map containing the update expression, attribute values, and
operator count
     */
    public static Map<String, Object> createComplexUpdateExpression(int
operationsCount) {
        // Initialize the expression parts and attribute values
        StringBuilder updateExpression = new StringBuilder("SET ");
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Generate the specified number of SET operations
        for (int i = 0; i < operationsCount; i++) {
            // Add comma between operations (except for the first one)
            if (i > 0) {
                updateExpression.append(", ");
            }

            // Alternate between different types of SET operations
            String valueKey = ":val" + i;

            switch (i % 3) {
                case 0:
                    // Simple assignment (1 operator: =)
                    updateExpression.append("attribute").append(i).append(" =
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" +
i).build());
                    break;
                case 1:
                    // Addition (2 operators: = and +)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = attribute")
                        .append(i)
                        .append(" + ")
                        .append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
```

```
                case 2:
                    // Conditional assignment with if_not_exists (2 operators: =
and if_not_exists)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = if_not_exists(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i *
10)).build());
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each operation has 1-2 operators as noted above
        int operatorCount = 0;
        for (int i = 0; i < operationsCount; i++) {
            operatorCount += (i % 3 == 0) ? 1 : 2;
        }

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("updateExpression", updateExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
     * Test the operator limit by attempting an operation with a complex
expression.
     *
     * <p>This method demonstrates what happens when an expression approaches or
```

```
     * exceeds the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param operatorCount Target number of operators to include
     * @return Map containing the result of the operation attempt
     */
    public static Map<String, Object> testOperatorLimit(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, int operatorCount) {

        // Create a complex update expression with the specified operator count
        Map<String, Object> expressionData =
            createComplexUpdateExpression((int) Math.ceil(operatorCount /
1.5)); // Adjust to get close to target count

        String updateExpression = (String)
expressionData.get("updateExpression");
        @SuppressWarnings("unchecked")
        Map<String, AttributeValue> expressionAttributeValues =
            (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
        int actualCount = (int) expressionData.get("operatorCount");

        System.out.println("Generated update expression with approximately " +
actualCount + " operators");

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression(updateExpression)
            .expressionAttributeValues(expressionAttributeValues)
            .returnValues("UPDATED_NEW")
            .build();

        try {
            // Attempt the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            Map<String, Object> result = new HashMap<>();
            result.put("success", true);
```

```
            result.put("message", "Operation succeeded with " + actualCount + "
operators");
            result.put("data", response);
            return result;

        } catch (DynamoDbException e) {
            // Check if the error is due to exceeding the operator limit
            if (e.getMessage().contains("too many operators")) {
                Map<String, Object> result = new HashMap<>();
                result.put("success", false);
                result.put("message", "Operation failed: " + e.getMessage());
                result.put("operatorCount", actualCount);
                return result;
            }

            // Return other errors
            Map<String, Object> result = new HashMap<>();
            result.put("success", false);
            result.put("message", "Operation failed: " + e.getMessage());
            result.put("error", e);
            return result;
        }
    }

    /**
     * Break down a complex expression into multiple simpler operations.
     *
     * <p>This method demonstrates how to handle expressions that would exceed
     * the 300 operator limit by breaking them into multiple operations.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param totalOperations Total number of operations to perform
     * @return Map containing the results of the operations
     */
    public static Map<String, Object> breakDownComplexExpression(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, int totalOperations) {

        // Calculate how many operations we can safely include in each batch
        // Using 150 as a conservative limit (well below 300)
        final int operationsPerBatch = 100;
```

```
        final int batchCount = (int) Math.ceil((double) totalOperations /
operationsPerBatch);

        System.out.println("Breaking down " + totalOperations + " operations into
" + batchCount + " batches");

        Map<String, Object> results = new HashMap<>();
        results.put("totalBatches", batchCount);

        Map<Integer, Map<String, Object>> batchResults = new HashMap<>();

        // Process each batch
        for (int batch = 0; batch < batchCount; batch++) {
            // Calculate the operations for this batch
            int batchStart = batch * operationsPerBatch;
            int batchEnd = Math.min(batchStart + operationsPerBatch,
totalOperations);
            int batchSize = batchEnd - batchStart;

            System.out.println(
                "Processing batch " + (batch + 1) + "/" + batchCount + " with " +
batchSize + " operations");

            // Create an update expression for this batch
            Map<String, Object> expressionData =
createComplexUpdateExpression(batchSize);

            String updateExpression = (String)
expressionData.get("updateExpression");
            @SuppressWarnings("unchecked")
            Map<String, AttributeValue> expressionAttributeValues =
                (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
            int operatorCount = (int) expressionData.get("operatorCount");

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression(updateExpression)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues("UPDATED_NEW")
                .build();
```

```
            try {
                // Perform the update operation for this batch
                UpdateItemResponse response = dynamoDbClient.updateItem(request);

                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", true);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("attributes", response.attributes());

                batchResults.put(batch, batchResult);

            } catch (DynamoDbException e) {
                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", false);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("error", e.getMessage());

                batchResults.put(batch, batchResult);

                // Continue with next batch instead of breaking
                continue;
            }
        }

        results.put("results", batchResults);
        return results;
    }

    /**
     * Count operators in a DynamoDB expression based on the rules in the
documentation.
     *
     * <p>This method demonstrates how operators are counted according to the
     * DynamoDB documentation.
     *
     * @param expression The DynamoDB expression to analyze
     * @return Map containing the breakdown of operator counts
     */
    public static Map<String, Integer> countOperatorsInExpression(String
expression) {
        // Initialize counters for different operator types
        Map<String, Integer> counts = new HashMap<>();
```

```
        counts.put("comparisonOperators", 0);
        counts.put("logicalOperators", 0);
        counts.put("functions", 0);
        counts.put("arithmeticOperators", 0);
        counts.put("specialOperators", 0);
        counts.put("total", 0);

        // Count comparison operators (=, <>, <, <=, >, >=)
        // This is a simplified approach and may not catch all cases
        int comparisonCount = 0;
        Pattern comparisonPattern = Pattern.compile("(=|<>|<=|>=|<|>)");
        Matcher comparisonMatcher = comparisonPattern.matcher(expression);
        while (comparisonMatcher.find()) {
            comparisonCount++;
        }
        counts.put("comparisonOperators", comparisonCount);

        // Count logical operators (AND, OR, NOT)
        int andCount = countOccurrences(expression, "\\bAND\\b");
        int orCount = countOccurrences(expression, "\\bOR\\b");
        int notCount = countOccurrences(expression, "\\bNOT\\b");
        counts.put("logicalOperators", andCount + orCount + notCount);

        // Count functions (attribute_exists, attribute_not_exists,
 attribute_type, begins_with, contains, size)
        int functionCount = countOccurrences(
            expression,
            "\\b(attribute_exists|attribute_not_exists|attribute_type|
begins_with|contains|size|if_not_exists)\\(");
        counts.put("functions", functionCount);

        // Count arithmetic operators (+ and -)
        // This is a simplified approach and may not catch all cases
        int arithmeticCount = 0;
        Pattern arithmeticPattern = Pattern.compile("[a-zA-Z0-9_)\\]]\\s*[\\+\
\-]\\s*[a-zA-Z0-9_:(]");
        Matcher arithmeticMatcher = arithmeticPattern.matcher(expression);
        while (arithmeticMatcher.find()) {
            arithmeticCount++;
        }
        counts.put("arithmeticOperators", arithmeticCount);

        // Count special operators (BETWEEN, IN)
        int betweenCount = countOccurrences(expression, "\\bBETWEEN\\b");
```

```
        int inCount = countOccurrences(expression, "\\bIN\\b");
        counts.put("specialOperators", betweenCount + inCount);

        // Add extra operators for BETWEEN (each BETWEEN includes an AND)
        int currentLogicalOps = counts.getOrDefault("logicalOperators", 0);
        counts.put("logicalOperators", currentLogicalOps + betweenCount);

        // Calculate total
        int total = counts.getOrDefault("comparisonOperators", 0)
            + counts.getOrDefault("logicalOperators", 0)
            + counts.getOrDefault("functions", 0)
            + counts.getOrDefault("arithmeticOperators", 0)
            + counts.getOrDefault("specialOperators", 0);
        counts.put("total", total);

        return counts;
    }

    /**
     * Helper method to count occurrences of a pattern in a string.
     *
     * @param text The text to search in
     * @param regex The regular expression pattern to search for
     * @return The number of occurrences
     */
    private static int countOccurrences(String text, String regex) {
        final Pattern pattern = Pattern.compile(regex);
        final Matcher matcher = pattern.matcher(text);
        int count = 0;
        while (matcher.find()) {
            count++;
        }
        return count;
    }
```

Exemplo de uso do operador de expressão contando com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());
```

```
        System.out.println("Demonstrating DynamoDB expression operator counting
and the 300 operator limit");

        try {
            // Example 1: Analyze a simple expression
            System.out.println("\nExample 1: Analyzing a simple expression");
            String simpleExpression = "Price = :price AND Rating > :rating AND
Category IN (:cat1, :cat2, :cat3)";
            Map<String, Integer> simpleCount =
countOperatorsInExpression(simpleExpression);

            System.out.println("Expression: " + simpleExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
simpleCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
simpleCount.get("logicalOperators"));
            System.out.println("- Functions: " + simpleCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
simpleCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
simpleCount.get("specialOperators"));
            System.out.println("- Total operators: " + simpleCount.get("total"));

            // Example 2: Analyze a complex expression
            System.out.println("\nExample 2: Analyzing a complex expression");
            String complexExpression = "(attribute_exists(Category) AND Size
BETWEEN :min AND :max) OR "
                + "(Price > :price AND contains(Description, :keyword) AND "
                + "(Rating >= :minRating OR Reviews > :minReviews))";
            Map<String, Integer> complexCount =
countOperatorsInExpression(complexExpression);

            System.out.println("Expression: " + complexExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
complexCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
complexCount.get("logicalOperators"));
            System.out.println("- Functions: " + complexCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
complexCount.get("arithmeticOperators"));
```

```
            System.out.println("- Special operators: " +
complexCount.get("specialOperators"));
            System.out.println("- Total operators: " +
complexCount.get("total"));

            // Example 3: Test approaching the operator limit
            System.out.println("\nExample 3: Testing an expression approaching
the operator limit");
            Map<String, Object> approachingLimit =
testOperatorLimit(dynamoDbClient, tableName, key, 290);
            System.out.println(approachingLimit.get("message"));

            // Example 4: Test exceeding the operator limit
            System.out.println("\nExample 4: Testing an expression exceeding the
operator limit");
            Map<String, Object> exceedingLimit =
testOperatorLimit(dynamoDbClient, tableName, key, 310);
            System.out.println(exceedingLimit.get("message"));

            // Example 5: Breaking down a complex expression
            System.out.println("\nExample 5: Breaking down a complex expression
into multiple operations");
            Map<String, Object> breakdownResult =
breakDownComplexExpression(dynamoDbClient, tableName, key, 500);
            @SuppressWarnings("unchecked")
            Map<Integer, Map<String, Object>> results =
                (Map<Integer, Map<String, Object>>)
breakdownResult.get("results");
            System.out.println(
                "Processed " + results.size() + " of " +
breakdownResult.get("totalBatches") + " batches");

            // Explain the operator counting rules
            System.out.println("\nKey points about DynamoDB expression operator
counting:");
            System.out.println("1. The maximum number of operators in any
expression is 300");
            System.out.println("2. Each comparison operator (=, <>, <, <=, >, >=)
counts as 1 operator");
            System.out.println("3. Each logical operator (AND, OR, NOT) counts as
1 operator");
            System.out.println("4. Each function call (attribute_exists,
contains, etc.) counts as 1 operator");
```

```
            System.out.println("5. Each arithmetic operator (+ or -) counts as 1
 operator");
            System.out.println("6. BETWEEN counts as 2 operators (BETWEEN itself
 and the AND within it)");
            System.out.println("7. IN counts as 1 operator regardless of the
 number of values");
            System.out.println("8. Parentheses for grouping and attribute paths
 don't count as operators");
            System.out.println("9. When you exceed the limit, the error always
 reports '301 operators'");
            System.out.println("10. For complex operations, break them into
 multiple smaller operations");

        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Demonstre a contagem de operadores de expressão usando o AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  QueryCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Create a complex filter expression with a specified number of conditions.
 *
 * This function demonstrates how to generate a complex expression with
 * a specific number of operators to test the 300 operator limit.
 *
 * @param {number} conditionsCount - Number of conditions to include
```

```
 * @param {boolean} useAnd - Whether to use AND (true) or OR (false) between
 conditions
 * @returns {Object} - Object containing the filter expression and attribute
 values
 */
function createComplexFilterExpression(conditionsCount, useAnd = true) {
  // Initialize the expression parts and attribute values
  const conditions = [];
  const expressionAttributeValues = {};

  // Generate the specified number of conditions
  for (let i = 0; i < conditionsCount; i++) {
    // Alternate between different comparison operators for variety
    let condition;
    const valueKey = `:val${i}`;

    switch (i % 5) {
      case 0:
        condition = `attribute${i} = ${valueKey}`;
        expressionAttributeValues[valueKey] = `value${i}`;
        break;
      case 1:
        condition = `attribute${i} > ${valueKey}`;
        expressionAttributeValues[valueKey] = i;
        break;
      case 2:
        condition = `attribute${i} < ${valueKey}`;
        expressionAttributeValues[valueKey] = i * 10;
        break;
      case 3:
        condition = `contains(attribute${i}, ${valueKey})`;
        expressionAttributeValues[valueKey] = `substring${i}`;
        break;
      case 4:
        condition = `attribute_exists(attribute${i})`;
        break;
    }

    conditions.push(condition);
  }

  // Join the conditions with AND or OR
  const operator = useAnd ? " AND " : " OR ";
  const filterExpression = conditions.join(operator);
```

```
  // Calculate the operator count
  // Each condition has 1 operator (=, >, <, contains, attribute_exists)
  // Each AND or OR between conditions is 1 operator
  const operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount
 - 1 : 0);

  return {
    filterExpression,
    expressionAttributeValues,
    operatorCount
  };
}

/**
 * Create a complex update expression with a specified number of operations.
 *
 * This function demonstrates how to generate a complex update expression with
 * a specific number of operators to test the 300 operator limit.
 *
 * @param {number} operationsCount - Number of operations to include
 * @returns {Object} - Object containing the update expression and attribute
 values
 */
function createComplexUpdateExpression(operationsCount) {
  // Initialize the expression parts and attribute values
  const setOperations = [];
  const expressionAttributeValues = {};

  // Generate the specified number of SET operations
  for (let i = 0; i < operationsCount; i++) {
    // Alternate between different types of SET operations
    let operation;
    const valueKey = `:val${i}`;

    switch (i % 3) {
      case 0:
        // Simple assignment (1 operator: =)
        operation = `attribute${i} = ${valueKey}`;
        expressionAttributeValues[valueKey] = `value${i}`;
        break;
      case 1:
        // Addition (2 operators: = and +)
        operation = `attribute${i} = attribute${i} + ${valueKey}`;
```

```
        expressionAttributeValues[valueKey] = i;
        break;
      case 2:
        // Conditional assignment with if_not_exists (2 operators: = and
 if_not_exists)
        operation = `attribute${i} = if_not_exists(attribute${i}, ${valueKey})`;
        expressionAttributeValues[valueKey] = i * 10;
        break;
    }

    setOperations.push(operation);
  }

  // Create the update expression
  const updateExpression = `SET ${setOperations.join(", ")}`;

  // Calculate the operator count
  // Each operation has 1-2 operators as noted above
  let operatorCount = 0;
  for (let i = 0; i < operationsCount; i++) {
    operatorCount += (i % 3 === 0) ? 1 : 2;
  }

  return {
    updateExpression,
    expressionAttributeValues,
    operatorCount
  };
}

/**
 * Test the operator limit by attempting an operation with a complex expression.
 *
 * This function demonstrates what happens when an expression approaches or
 * exceeds the 300 operator limit.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {number} operatorCount - Target number of operators to include
 * @returns {Promise<Object>} - Result of the operation attempt
 */
async function testOperatorLimit(
  config,
```

```
  tableName,
  key,
  operatorCount
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create a complex update expression with the specified operator count
  const { updateExpression, expressionAttributeValues, operatorCount:
 actualCount } =
    createComplexUpdateExpression(Math.ceil(operatorCount / 1.5)); // Adjust to
 get close to target count

  console.log(`Generated update expression with approximately ${actualCount}
 operators`);

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Attempt the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return {
      success: true,
      message: `Operation succeeded with ${actualCount} operators`,
      data: response
    };
  } catch (error) {
    // Check if the error is due to exceeding the operator limit
    if (error.name === "ValidationException" &&
        error.message.includes("too many operators")) {
      return {
        success: false,
        message: `Operation failed: ${error.message}`,
        operatorCount: actualCount
      };
    }
```

```
    // Return other errors
    return {
      success: false,
      message: `Operation failed: ${error.message}`,
      error
    };
  }
}

/**
 * Break down a complex expression into multiple simpler operations.
 *
 * This function demonstrates how to handle expressions that would exceed
 * the 300 operator limit by breaking them into multiple operations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {number} totalOperations - Total number of operations to perform
 * @returns {Promise<Object>} - Result of the operations
 */
async function breakDownComplexExpression(
  config,
  tableName,
  key,
  totalOperations
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Calculate how many operations we can safely include in each batch
  // Using 150 as a conservative limit (well below 300)
  const operationsPerBatch = 100;
  const batchCount = Math.ceil(totalOperations / operationsPerBatch);

  console.log(`Breaking down ${totalOperations} operations into ${batchCount}
 batches`);

  const results = [];

  // Process each batch
  for (let batch = 0; batch < batchCount; batch++) {
```

```
    // Calculate the operations for this batch
    const batchStart = batch * operationsPerBatch;
    const batchEnd = Math.min(batchStart + operationsPerBatch, totalOperations);
    const batchSize = batchEnd - batchStart;

    console.log(`Processing batch ${batch + 1}/${batchCount} with ${batchSize}
operations`);

    // Create an update expression for this batch
    const { updateExpression, expressionAttributeValues, operatorCount } =
      createComplexUpdateExpression(batchSize);

    // Define the update parameters
    const params = {
      TableName: tableName,
      Key: key,
      UpdateExpression: updateExpression,
      ExpressionAttributeValues: expressionAttributeValues,
      ReturnValues: "UPDATED_NEW"
    };

    try {
      // Perform the update operation for this batch
      const response = await docClient.send(new UpdateCommand(params));

      results.push({
        batch: batch + 1,
        success: true,
        operatorCount,
        attributes: response.Attributes
      });
    } catch (error) {
      results.push({
        batch: batch + 1,
        success: false,
        operatorCount,
        error: error.message
      });

      // Stop processing if an error occurs
      break;
    }
  }
```

```
  return {
    totalBatches: batchCount,
    results
  };
}

/**
 * Count operators in a DynamoDB expression based on the rules in the
 documentation.
 *
 * This function demonstrates how operators are counted according to the
 * DynamoDB documentation.
 *
 * @param {string} expression - The DynamoDB expression to analyze
 * @returns {Object} - Breakdown of operator counts
 */
function countOperatorsInExpression(expression) {
  // Initialize counters for different operator types
  const counts = {
    comparisonOperators: 0,
    logicalOperators: 0,
    functions: 0,
    arithmeticOperators: 0,
    specialOperators: 0,
    total: 0
  };

  // Count comparison operators (=, <>, <, <=, >, >=)
  const comparisonRegex = /[^<>]=[^=]|<>|<=|>=|[^<]>[^=]|[^>]<[^=]/g;
  const comparisonMatches = expression.match(comparisonRegex) || [];
  counts.comparisonOperators = comparisonMatches.length;

  // Count logical operators (AND, OR, NOT)
  const andMatches = expression.match(/\bAND\b/g) || [];
  const orMatches = expression.match(/\bOR\b/g) || [];
  const notMatches = expression.match(/\bNOT\b/g) || [];
  counts.logicalOperators = andMatches.length + orMatches.length +
 notMatches.length;

  // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
  const functionRegex = /\b(attribute_exists|attribute_not_exists|attribute_type|
begins_with|contains|size|if_not_exists)\(/g;
  const functionMatches = expression.match(functionRegex) || [];
```

```
  counts.functions = functionMatches.length;

  // Count arithmetic operators (+ and -)
  const arithmeticMatches = expression.match(/[a-zA-Z0-9_)\]]\s*[\+\-]\s*[a-zA-
Z0-9_(:]/g) || [];
  counts.arithmeticOperators = arithmeticMatches.length;

  // Count special operators (BETWEEN, IN)
  const betweenMatches = expression.match(/\bBETWEEN\b/g) || [];
  const inMatches = expression.match(/\bIN\b/g) || [];
  counts.specialOperators = betweenMatches.length + inMatches.length;

  // Add extra operators for BETWEEN (each BETWEEN includes an AND)
  counts.logicalOperators += betweenMatches.length;

  // Calculate total
  counts.total = counts.comparisonOperators +
                 counts.logicalOperators +
                 counts.functions +
                 counts.arithmeticOperators +
                 counts.specialOperators;

  return counts;
}
```

Exemplo de uso do operador de expressão contando com AWS SDK para JavaScript.

```
/**
 * Example of how to work with expression operator counting.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating DynamoDB expression operator counting and the 300
 operator limit");

  try {
    // Example 1: Analyze a simple expression
    console.log("\nExample 1: Analyzing a simple expression");
```

```
   const simpleExpression = "Price = :price AND Rating > :rating AND Category IN
(:cat1, :cat2, :cat3)";
   const simpleCount = countOperatorsInExpression(simpleExpression);

   console.log(`Expression: ${simpleExpression}`);
   console.log("Operator count breakdown:");
   console.log(`- Comparison operators: ${simpleCount.comparisonOperators}`);
   console.log(`- Logical operators: ${simpleCount.logicalOperators}`);
   console.log(`- Functions: ${simpleCount.functions}`);
   console.log(`- Arithmetic operators: ${simpleCount.arithmeticOperators}`);
   console.log(`- Special operators: ${simpleCount.specialOperators}`);
   console.log(`- Total operators: ${simpleCount.total}`);

   // Example 2: Analyze a complex expression
   console.log("\nExample 2: Analyzing a complex expression");
   const complexExpression =
     "(attribute_exists(Category) AND Size BETWEEN :min AND :max) OR " +
     "(Price > :price AND contains(Description, :keyword) AND " +
     "(Rating >= :minRating OR Reviews > :minReviews))";
   const complexCount = countOperatorsInExpression(complexExpression);

   console.log(`Expression: ${complexExpression}`);
   console.log("Operator count breakdown:");
   console.log(`- Comparison operators: ${complexCount.comparisonOperators}`);
   console.log(`- Logical operators: ${complexCount.logicalOperators}`);
   console.log(`- Functions: ${complexCount.functions}`);
   console.log(`- Arithmetic operators: ${complexCount.arithmeticOperators}`);
   console.log(`- Special operators: ${complexCount.specialOperators}`);
   console.log(`- Total operators: ${complexCount.total}`);

   // Example 3: Test approaching the operator limit
   console.log("\nExample 3: Testing an expression approaching the operator
limit");
   const approachingLimit = await testOperatorLimit(config, tableName, key,
290);
   console.log(approachingLimit.message);

   // Example 4: Test exceeding the operator limit
   console.log("\nExample 4: Testing an expression exceeding the operator
limit");
   const exceedingLimit = await testOperatorLimit(config, tableName, key, 310);
   console.log(exceedingLimit.message);

   // Example 5: Breaking down a complex expression
```

```
    console.log("\nExample 5: Breaking down a complex expression into multiple
 operations");
    const breakdownResult = await breakDownComplexExpression(config, tableName,
 key, 500);
    console.log(`Processed ${breakdownResult.results.length} of
 ${breakdownResult.totalBatches} batches`);

    // Explain the operator counting rules
    console.log("\nKey points about DynamoDB expression operator counting:");
    console.log("1. The maximum number of operators in any expression is 300");
    console.log("2. Each comparison operator (=, <>, <, <=, >, >=) counts as 1
 operator");
    console.log("3. Each logical operator (AND, OR, NOT) counts as 1 operator");
    console.log("4. Each function call (attribute_exists, contains, etc.) counts
 as 1 operator");
    console.log("5. Each arithmetic operator (+ or -) counts as 1 operator");
    console.log("6. BETWEEN counts as 2 operators (BETWEEN itself and the AND
 within it)");
    console.log("7. IN counts as 1 operator regardless of the number of values");
    console.log("8. Parentheses for grouping and attribute paths don't count as
 operators");
    console.log("9. When you exceed the limit, the error always reports '301
 operators'");
    console.log("10. For complex operations, break them into multiple smaller
 operations");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Demonstre a contagem de operadores de expressão usando o AWS SDK para Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, List, Optional, Tuple



def create_complex_filter_expression(
    attribute_name: str, values: List[Any], use_or: bool = True
) -> Tuple[str, Dict[str, Any], Dict[str, str], int]:
    """
    Create a complex filter expression with multiple conditions.

    This function demonstrates how to build a complex filter expression
    and count the number of operators used.

    Args:
        attribute_name (str): The name of the attribute to filter on.
        values (List[Any]): List of values to compare against.
        use_or (bool, optional): Whether to use OR between conditions. Defaults
 to True.

    Returns:
        Tuple[str, Dict[str, Any], Dict[str, str], int]: A tuple containing:
            - The filter expression string
            - Expression attribute values
            - Expression attribute names
            - The number of operators used
    """
    if not values:
        return "", {}, {}, 0

    # Initialize expression components
    filter_expression = ""
    expression_attribute_values = {}
    expression_attribute_names = {"#attr": attribute_name}
    operator_count = 0

    # Build the filter expression
    for i, value in enumerate(values):
        value_placeholder = f":val{i}"
        expression_attribute_values[value_placeholder] = value

        if i > 0:
```

```
            # Add OR or AND operator between conditions
            filter_expression += " OR " if use_or else " AND "
            operator_count += 1  # Count the OR/AND operator

        # Add the condition
        filter_expression += f"#attr = {value_placeholder}"
        operator_count += 1  # Count the = operator

    return (
        filter_expression,
        expression_attribute_values,
        expression_attribute_names,
        operator_count,
    )


def create_nested_filter_expression(
    depth: int, conditions_per_level: int
) -> Tuple[str, Dict[str, Any], Dict[str, str], int]:
    """
    Create a deeply nested filter expression with multiple conditions.

    This function demonstrates how to build a complex nested filter expression
    and count the number of operators used.

    Args:
        depth (int): The depth of nesting.
        conditions_per_level (int): Number of conditions at each level.

    Returns:
        Tuple[str, Dict[str, Any], Dict[str, str], int]: A tuple containing:
            - The filter expression string
            - Expression attribute values
            - Expression attribute names
            - The number of operators used
    """
    if depth <= 0 or conditions_per_level <= 0:
        return "", {}, {}, 0

    # Initialize expression components
    expression_attribute_values = {}
    expression_attribute_names = {}
    operator_count = 0
```

```
    def build_nested_expression(current_depth: int, prefix: str) -> str:
        nonlocal operator_count

        if current_depth <= 0:
            return ""

        # Build conditions at this level
        conditions = []
        for i in range(conditions_per_level):
            attr_name = f"attr{prefix}_{i}"
            attr_placeholder = f"#attr{prefix}_{i}"
            val_placeholder = f":val{prefix}_{i}"

            expression_attribute_names[attr_placeholder] = attr_name
            expression_attribute_values[val_placeholder] = i

            conditions.append(f"{attr_placeholder} = {val_placeholder}")
            operator_count += 1  # Count the = operator

        # Join conditions with AND
        level_expression = " AND ".join(conditions)
        operator_count += max(0, len(conditions) - 1)  # Count the AND operators

        # If not at the deepest level, add nested expressions
        if current_depth > 1:
            nested_expr = build_nested_expression(current_depth - 1,
f"{prefix}_{current_depth}")
            if nested_expr:
                level_expression = f"({level_expression}) OR ({nested_expr})"
                operator_count += 1  # Count the OR operator

        return level_expression

    # Build the expression starting from the top level
    filter_expression = build_nested_expression(depth, "1")

    return (
        filter_expression,
        expression_attribute_values,
        expression_attribute_names,
        operator_count,
    )
```

```
def count_operators_in_update_expression(update_expression: str) -> int:
    """
    Count the number of operators in an update expression.

    This function demonstrates how to count operators in an update expression
    based on DynamoDB's rules.

    Args:
        update_expression (str): The update expression to analyze.

    Returns:
        int: The number of operators in the expression.
    """
    operator_count = 0

    # Count SET operations
    if "SET" in update_expression:
        set_section = (
            update_expression.split("SET")[1].split("REMOVE")[0].split("ADD")
[0].split("DELETE")[0]
        )

        # Count assignment operators (=)
        operator_count += set_section.count("=")

        # Count arithmetic operators (+, -)
        operator_count += set_section.count("+")
        operator_count += set_section.count("-")

        # Count list_append function calls (each counts as 1 operator)
        operator_count += set_section.lower().count("list_append")

        # Count if_not_exists function calls (each counts as 1 operator)
        operator_count += set_section.lower().count("if_not_exists")

    # Count REMOVE operations (no additional operators)

    # Count ADD operations (each ADD counts as 1 operator)
    if "ADD" in update_expression:
        add_section = (
            update_expression.split("ADD")[1].split("DELETE")[0].split("SET")
[0].split("REMOVE")[0]
        )
        operator_count += add_section.count(",") + 1
```

```
    # Count DELETE operations (each DELETE counts as 1 operator)
    if "DELETE" in update_expression:
        delete_section = (
            update_expression.split("DELETE")[1].split("SET")[0].split("ADD")
[0].split("REMOVE")[0]
        )
        operator_count += delete_section.count(",") + 1

    return operator_count


def count_operators_in_condition_expression(condition_expression: str) -> int:
    """
    Count the number of operators in a condition expression.

    This function demonstrates how to count operators in a condition expression
    based on DynamoDB's rules.

    Args:
        condition_expression (str): The condition expression to analyze.

    Returns:
        int: The number of operators in the expression.
    """
    operator_count = 0

    # Count comparison operators
    comparison_operators = ["=", "<>", "<", "<=", ">", ">="]
    for op in comparison_operators:
        operator_count += condition_expression.count(op)

    # Count logical operators
    operator_count += condition_expression.upper().count(" AND ")
    operator_count += condition_expression.upper().count(" OR ")
    operator_count += condition_expression.upper().count("NOT ")

    # Count BETWEEN operator (counts as 2: BETWEEN + AND)
    between_count = condition_expression.upper().count(" BETWEEN ")
    operator_count += between_count * 2

    # Count IN operator (counts as 1 regardless of number of values)
    operator_count += condition_expression.upper().count(" IN ")
```

```
    # Count functions (each counts as 1 operator)
    functions = [
        "attribute_exists",
        "attribute_not_exists",
        "attribute_type",
        "begins_with",
        "contains",
        "size",
    ]
    for func in functions:
        operator_count += condition_expression.lower().count(func)

    return operator_count


# Note: This function is for demonstration purposes only and should be called
 from example_usage()
# It's not meant to be used directly as a test function
def _test_expression_limit(
    table_name: str, key: Dict[str, Any], operator_count: int, attribute_name:
 str = "TestAttribute"
) -> Tuple[bool, Optional[str]]:
    """
    Test if an expression with a specific number of operators exceeds the limit.

    This function demonstrates how to test the 300 operator limit by creating
    an expression with a specified number of operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        operator_count (int): The number of operators to include in the
 expression.
        attribute_name (str, optional): The name of the attribute to update.
 Defaults to "TestAttribute".

    Returns:
        Tuple[bool, Optional[str]]: A tuple containing:
            - A boolean indicating if the operation succeeded
            - The error message if it failed, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Create an update expression with the specified number of operators
    update_expression = f"SET #{attribute_name} = :val0"
    expression_attribute_names = {f"#{attribute_name}": attribute_name}
    expression_attribute_values = {":val0": 0}

    # Add additional SET operations to reach the desired operator count
    # Each assignment adds 1 operator
    for i in range(1, operator_count):
        attr_name = f"{attribute_name}{i}"
        attr_placeholder = f"#attr{i}"
        val_placeholder = f":val{i}"

        update_expression += f", {attr_placeholder} = {val_placeholder}"
        expression_attribute_names[attr_placeholder] = attr_name
        expression_attribute_values[val_placeholder] = i

    try:
        # Attempt the update operation
        table.update_item(
            Key=key,
            UpdateExpression=update_expression,
            ExpressionAttributeNames=expression_attribute_names,
            ExpressionAttributeValues=expression_attribute_values,
        )
        return True, None
    except ClientError as e:
        error_message = e.response["Error"]["Message"]

        if "expression contains too many operators" in error_message.lower():
            return False, error_message
        else:
            # Other error occurred
            raise
```

Exemplo de uso do operador de expressão contando com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the expression operator counting functions."""
```

```
    print("Example 1: Creating a complex filter expression with multiple
conditions")
    attribute_name = "Status"
    values = ["Active", "Pending", "Processing", "Shipped", "Delivered"]

    filter_expr, expr_attr_vals, expr_attr_names, op_count =
create_complex_filter_expression(
        attribute_name=attribute_name, values=values, use_or=True
    )

    print(f"Filter Expression: {filter_expr}")
    print(f"Expression Attribute Values: {expr_attr_vals}")
    print(f"Expression Attribute Names: {expr_attr_names}")
    print(f"Operator Count: {op_count}")

    print("\nExample 2: Creating a nested filter expression")
    nested_expr, nested_vals, nested_names, nested_count =
create_nested_filter_expression(
        depth=3, conditions_per_level=2
    )

    print(f"Nested Filter Expression: {nested_expr}")
    print(f"Operator Count: {nested_count}")

    print("\nExample 3: Counting operators in an update expression")
    update_expression = "SET #name = :name, #age = :age + :increment,
#address.#city = :city, #status = if_not_exists(#status, :default_status) REMOVE
#old_field ADD #counter :value DELETE #set_attr :set_val"
    update_op_count = count_operators_in_update_expression(update_expression)

    print(f"Update Expression: {update_expression}")
    print(f"Operator Count: {update_op_count}")

    print("\nExample 4: Counting operators in a condition expression")
    condition_expression = "(#status = :active OR #status = :pending) AND #price
BETWEEN :min_price AND :max_price AND attribute_exists(#category) AND NOT
(#stock <= :min_stock)"
    condition_op_count =
count_operators_in_condition_expression(condition_expression)

    print(f"Condition Expression: {condition_expression}")
    print(f"Operator Count: {condition_op_count}")

    print("\nExample 5: Testing the 300 operator limit")
```

```
    # This is just for demonstration - in a real application, you would use your
actual table
    # Note: This function is renamed to _test_expression_limit to avoid pytest
trying to run it
    print("In a real application, you would test with _test_expression_limit
function")
    print("Expression with 250 operators would be under the limit")
    print("Expression with 350 operators would exceed the 300 operator limit")

    print("\nOperator Counting Rules in DynamoDB:")
    print("1. Comparison Operators (=, <>, <, <=, >, >=): 1 operator each")
    print("2. Logical Operators (AND, OR, NOT): 1 operator each")
    print("3. BETWEEN: 2 operators (BETWEEN + AND)")
    print("4. IN: 1 operator (regardless of number of values)")
    print("5. Functions (attribute_exists, begins_with, etc.): 1 operator each")
    print("6. Arithmetic Operators (+, -): 1 operator each")
    print("7. SET assignments (=): 1 operator each")
    print("8. ADD and DELETE operations: 1 operator each")

    print("\nStrategies for Working Within the 300 Operator Limit:")
    print("1. Break operations into multiple requests")
    print("2. Use DynamoDB Transactions for complex operations")
    print("3. Optimize data model to reduce query complexity")
    print("4. Use application-side filtering for less critical filters")
    print("5. Consider using IN operator instead of multiple OR conditions")
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Crie uma API REST do API Gateway para rastrear COVID-19 dados

O exemplo de código a seguir mostra como criar uma API REST que simula um sistema para rastrear casos diários nos Estados Unidos da América, usando dados fictícios. COVID-19

Python

SDK para Python (Boto3)

Mostra como usar o AWS Chalice com o AWS SDK para Python (Boto3) para criar uma API REST sem servidor que usa o Amazon API Gateway e o Amazon DynamoDB AWS Lambda. A API REST simula um sistema que rastreia casos diários nos Estados Unidos da América, usando dados fictícios. COVID-19 Aprenda como:

- Use o AWS Chalice para definir rotas nas funções do Lambda que são chamadas para lidar com solicitações REST que chegam por meio do API Gateway.
- Usar as funções do Lambda para recuperar e armazenar dados em uma tabela do DynamoDB para atender a solicitações REST.
- Defina a estrutura da tabela e os recursos da função de segurança em um AWS CloudFormation modelo.
- Use AWS Chalice e CloudFormation para empacotar e implantar todos os recursos necessários.
- Use CloudFormation para limpar todos os recursos criados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Criar uma aplicação de mensageiro com o Step Functions

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.
- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Crie uma tabela do DynamoDB com um índice secundário global usando o AWS SDK

O código de exemplo a seguir mostra como criar uma tabela com um índice secundário global.

Java

SDK para Java 2.x

Crie uma tabela do DynamoDB com o Índice Secundário Global usando o. AWS SDK for Java 2.x

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProjectionType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

import java.util.ArrayList;
import java.util.HashMap;
```

```
import java.util.List;
import java.util.Map;

    public void createTable() {
        try {
            // Attribute definitions
            final List<AttributeDefinition> attributeDefinitions = new
 ArrayList<>();
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(ISSUE_ID_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(TITLE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(CREATE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(DUE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());

            // Key schema for table
            final List<KeySchemaElement> tableKeySchema = new ArrayList<>();
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.HASH)
                .build()); // Partition key
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.RANGE)
                .build()); // Sort key

            // Initial provisioned throughput settings for the indexes
            final ProvisionedThroughput ptIndex = ProvisionedThroughput.builder()
                .readCapacityUnits(1L)
                .writeCapacityUnits(1L)
                .build();

            // CreateDateIndex
            final List<KeySchemaElement> createDateKeySchema = new ArrayList<>();
```

```
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(CREATE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection createDateProjection = Projection.builder()
                .projectionType(ProjectionType.INCLUDE)
                .nonKeyAttributes(DESCRIPTION_ATTR, STATUS_ATTR)
                .build();

            final GlobalSecondaryIndex createDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(CREATE_DATE_INDEX)
                .keySchema(createDateKeySchema)
                .projection(createDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // TitleIndex
            final List<KeySchemaElement> titleKeySchema = new ArrayList<>();
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection titleProjection =

Projection.builder().projectionType(ProjectionType.KEYS_ONLY).build();

            final GlobalSecondaryIndex titleIndex =
GlobalSecondaryIndex.builder()
                .indexName(TITLE_INDEX)
                .keySchema(titleKeySchema)
                .projection(titleProjection)
                .provisionedThroughput(ptIndex)
                .build();
```

```
            // DueDateIndex
            final List<KeySchemaElement> dueDateKeySchema = new ArrayList<>();
            dueDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(DUE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());

            final Projection dueDateProjection =
                Projection.builder().projectionType(ProjectionType.ALL).build();

            final GlobalSecondaryIndex dueDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(DUE_DATE_INDEX)
                .keySchema(dueDateKeySchema)
                .projection(dueDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            final CreateTableRequest createTableRequest =
CreateTableRequest.builder()
                .tableName(TABLE_NAME)
                .keySchema(tableKeySchema)
                .attributeDefinitions(attributeDefinitions)
                .globalSecondaryIndexes(createDateIndex, titleIndex,
dueDateIndex)
                .provisionedThroughput(ProvisionedThroughput.builder()
                    .readCapacityUnits(1L)
                    .writeCapacityUnits(1L)
                    .build())
                .build();

            System.out.println("Creating table " + TABLE_NAME + "...");
            dynamoDbClient.createTable(createTableRequest);

            // Wait for table to become active
            System.out.println("Waiting for " + TABLE_NAME + " to become
ACTIVE...");
            final DynamoDbWaiter waiter = dynamoDbClient.waiter();
            final DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder().tableName(TABLE_NAME).build();

            final WaiterResponse<DescribeTableResponse> waiterResponse =
                waiter.waitUntilTableExists(describeTableRequest);
```

```
            waiterResponse.matched().response().ifPresent(response ->
 System.out.println("Table is now ready for use"));

        } catch (DynamoDbException e) {
            System.err.println("Error creating table: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK for Java 2.x da API.

Crie uma tabela do DynamoDB com configuração de taxa de transferência quente usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma tabela com o throughput a quente habilitado.

Java

SDK para Java 2.x

Crie uma tabela do DynamoDB com uma configuração de throughput a quente usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long
 readUnitsPerSecond, final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
```

```
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Builds a ProvisionedThroughput object with the specified read and write
capacity units.
     *
     * @param readCapacityUnits The read capacity units
     * @param writeCapacityUnits The write capacity units
     * @return A configured ProvisionedThroughput object
     */
    public static ProvisionedThroughput buildProvisionedThroughput(
        final Long readCapacityUnits, final Long writeCapacityUnits) {
        return ProvisionedThroughput.builder()
            .readCapacityUnits(readCapacityUnits)
            .writeCapacityUnits(writeCapacityUnits)
            .build();
    }

    /**
     * Builds an AttributeDefinition with the specified name and type.
     *
     * @param attributeName The attribute name
     * @param scalarAttributeType The attribute type
     * @return A configured AttributeDefinition
     */
    private static AttributeDefinition buildAttributeDefinition(
        final String attributeName, final ScalarAttributeType
scalarAttributeType) {
        return AttributeDefinition.builder()
            .attributeName(attributeName)
            .attributeType(scalarAttributeType)
            .build();
    }

    /**
     * Builds a KeySchemaElement with the specified name and key type.
     *
     * @param attributeName The attribute name
     * @param keyType The key type (HASH or RANGE)
     * @return A configured KeySchemaElement
     */
```

```
   private static KeySchemaElement buildKeySchemaElement(final String
attributeName, final KeyType keyType) {
       return KeySchemaElement.builder()
           .attributeName(attributeName)
           .keyType(keyType)
           .build();
   }

   /**
    * Creates a DynamoDB table with the specified configuration including warm
throughput settings.
    *
    * @param ddb The DynamoDB client
    * @param tableName The name of the table to create
    * @param partitionKey The partition key attribute name
    * @param sortKey The sort key attribute name
    * @param miscellaneousKeyAttribute Additional key attribute name for GSI
    * @param nonKeyAttribute Non-key attribute to include in GSI projection
    * @param tableReadCapacityUnits Read capacity units for the table
    * @param tableWriteCapacityUnits Write capacity units for the table
    * @param tableWarmReadUnitsPerSecond Warm read units per second for the
table
    * @param tableWarmWriteUnitsPerSecond Warm write units per second for the
table
    * @param globalSecondaryIndexName The name of the GSI to create
    * @param globalSecondaryIndexReadCapacityUnits Read capacity units for the
GSI
    * @param globalSecondaryIndexWriteCapacityUnits Write capacity units for the
GSI
    * @param globalSecondaryIndexWarmReadUnitsPerSecond Warm read units per
second for the GSI
    * @param globalSecondaryIndexWarmWriteUnitsPerSecond Warm write units per
second for the GSI
    */
   public static void createDynamoDBTable(
       final DynamoDbClient ddb,
       final String tableName,
       final String partitionKey,
       final String sortKey,
       final String miscellaneousKeyAttribute,
       final String nonKeyAttribute,
       final Long tableReadCapacityUnits,
       final Long tableWriteCapacityUnits,
       final Long tableWarmReadUnitsPerSecond,
```

```
        final Long tableWarmWriteUnitsPerSecond,
        final String globalSecondaryIndexName,
        final Long globalSecondaryIndexReadCapacityUnits,
        final Long globalSecondaryIndexWriteCapacityUnits,
        final Long globalSecondaryIndexWarmReadUnitsPerSecond,
        final Long globalSecondaryIndexWarmWriteUnitsPerSecond) {

        // Define the table attributes
        final AttributeDefinition partitionKeyAttribute =
buildAttributeDefinition(partitionKey, ScalarAttributeType.S);
        final AttributeDefinition sortKeyAttribute =
buildAttributeDefinition(sortKey, ScalarAttributeType.S);
        final AttributeDefinition miscellaneousKeyAttributeDefinition =
            buildAttributeDefinition(miscellaneousKeyAttribute,
ScalarAttributeType.N);
        final AttributeDefinition[] attributeDefinitions = {
            partitionKeyAttribute, sortKeyAttribute,
miscellaneousKeyAttributeDefinition
        };

        // Define the table key schema
        final KeySchemaElement partitionKeyElement =
buildKeySchemaElement(partitionKey, KeyType.HASH);
        final KeySchemaElement sortKeyElement = buildKeySchemaElement(sortKey,
KeyType.RANGE);
        final KeySchemaElement[] keySchema = {partitionKeyElement,
sortKeyElement};

        // Define the provisioned throughput for the table
        final ProvisionedThroughput provisionedThroughput =
            buildProvisionedThroughput(tableReadCapacityUnits,
tableWriteCapacityUnits);

        // Define the Global Secondary Index (GSI)
        final KeySchemaElement globalSecondaryIndexPartitionKeyElement =
buildKeySchemaElement(sortKey, KeyType.HASH);
        final KeySchemaElement globalSecondaryIndexSortKeyElement =
            buildKeySchemaElement(miscellaneousKeyAttribute, KeyType.RANGE);
        final KeySchemaElement[] gsiKeySchema = {
            globalSecondaryIndexPartitionKeyElement,
globalSecondaryIndexSortKeyElement
        };

        final Projection gsiProjection = Projection.builder()
```

```
            .projectionType(PROJECTION_TYPE_INCLUDE)
            .nonKeyAttributes(nonKeyAttribute)
            .build();

        final ProvisionedThroughput gsiProvisionedThroughput =
            buildProvisionedThroughput(globalSecondaryIndexReadCapacityUnits,
globalSecondaryIndexWriteCapacityUnits);

        // Define the warm throughput for the Global Secondary Index (GSI)
        final WarmThroughput gsiWarmThroughput = buildWarmThroughput(
            globalSecondaryIndexWarmReadUnitsPerSecond,
globalSecondaryIndexWarmWriteUnitsPerSecond);

        final GlobalSecondaryIndex globalSecondaryIndex =
GlobalSecondaryIndex.builder()
            .indexName(globalSecondaryIndexName)
            .keySchema(gsiKeySchema)
            .projection(gsiProjection)
            .provisionedThroughput(gsiProvisionedThroughput)
            .warmThroughput(gsiWarmThroughput)
            .build();

        // Define the warm throughput for the table
        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableWarmReadUnitsPerSecond,
tableWarmWriteUnitsPerSecond);

        final CreateTableRequest request = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(attributeDefinitions)
            .keySchema(keySchema)
            .provisionedThroughput(provisionedThroughput)
            .globalSecondaryIndexes(globalSecondaryIndex)
            .warmThroughput(tableWarmThroughput)
            .build();

        final CreateTableResponse response = ddb.createTable(request);
        System.out.println(response);
    }
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Crie uma tabela do DynamoDB com uma configuração de throughput a quente usando o AWS SDK para JavaScript.

```
import { DynamoDBClient, CreateTableCommand } from "@aws-sdk/client-dynamodb";

export async function createDynamoDBTableWithWarmThroughput(
  tableName,
  partitionKey,
  sortKey,
  miscKeyAttr,
  nonKeyAttr,
  tableProvisionedReadUnits,
  tableProvisionedWriteUnits,
  tableWarmReads,
  tableWarmWrites,
  indexName,
  indexProvisionedReadUnits,
  indexProvisionedWriteUnits,
  indexWarmReads,
  indexWarmWrites,
  region = "us-east-1"
) {
  try {
    const ddbClient = new DynamoDBClient({ region: region });
    const command = new CreateTableCommand({
      TableName: tableName,
      AttributeDefinitions: [
          { AttributeName: partitionKey, AttributeType: "S" },
          { AttributeName: sortKey, AttributeType: "S" },
          { AttributeName: miscKeyAttr, AttributeType: "N" },
      ],
      KeySchema: [
          { AttributeName: partitionKey, KeyType: "HASH" },
          { AttributeName: sortKey, KeyType: "RANGE" },
      ],
      ProvisionedThroughput: {
          ReadCapacityUnits: tableProvisionedReadUnits,
          WriteCapacityUnits: tableProvisionedWriteUnits,
      },
      WarmThroughput: {
```

```
            ReadUnitsPerSecond: tableWarmReads,
            WriteUnitsPerSecond: tableWarmWrites,
        },
        GlobalSecondaryIndexes: [
            {
              IndexName: indexName,
              KeySchema: [
                  { AttributeName: sortKey, KeyType: "HASH" },
                  { AttributeName: miscKeyAttr, KeyType: "RANGE" },
              ],
              Projection: {
                  ProjectionType: "INCLUDE",
                  NonKeyAttributes: [nonKeyAttr],
              },
              ProvisionedThroughput: {
                  ReadCapacityUnits: indexProvisionedReadUnits,
                  WriteCapacityUnits: indexProvisionedWriteUnits,
              },
              WarmThroughput: {
                  ReadUnitsPerSecond: indexWarmReads,
                  WriteUnitsPerSecond: indexWarmWrites,
              },
            },
        ],
      });
      const response = await ddbClient.send(command);
      console.log(response);
      return response;
    } catch (error) {
      console.error(`Error creating table: ${error}`);
      throw error;
    }
  }

  // Example usage (commented out for testing)
  /*
  createDynamoDBTableWithWarmThroughput(
    'example-table',
    'pk',
    'sk',
    'gsiKey',
    'data',
    10, 10, 5, 5,
    'example-index',
```

```
  5, 5, 2, 2
);
*/
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Crie uma tabela do DynamoDB com uma configuração de throughput a quente usando o AWS SDK para Python (Boto3).

```
from boto3 import client
from botocore.exceptions import ClientError


def create_dynamodb_table_warm_throughput(
    table_name,
    partition_key,
    sort_key,
    misc_key_attr,
    non_key_attr,
    table_provisioned_read_units,
    table_provisioned_write_units,
    table_warm_reads,
    table_warm_writes,
    gsi_name,
    gsi_provisioned_read_units,
    gsi_provisioned_write_units,
    gsi_warm_reads,
    gsi_warm_writes,
    region_name="us-east-1",
):
    """
    Creates a DynamoDB table with a warm throughput setting configured.

    :param table_name: The name of the table to be created.
    :param partition_key: The partition key for the table being created.
```

```
    :param sort_key: The sort key for the table being created.
    :param misc_key_attr: A miscellaneous key attribute for the table being
created.
    :param non_key_attr: A non-key attribute for the table being created.
    :param table_provisioned_read_units: The newly created table's provisioned
read capacity units.
    :param table_provisioned_write_units: The newly created table's provisioned
write capacity units.
    :param table_warm_reads: The read units per second setting for the table's
warm throughput.
    :param table_warm_writes: The write units per second setting for the table's
warm throughput.
    :param gsi_name: The name of the Global Secondary Index (GSI) to be created
on the table.
    :param gsi_provisioned_read_units: The configured Global Secondary Index
(GSI) provisioned read capacity units.
    :param gsi_provisioned_write_units: The configured Global Secondary Index
(GSI) provisioned write capacity units.
    :param gsi_warm_reads: The read units per second setting for the Global
Secondary Index (GSI)'s warm throughput.
    :param gsi_warm_writes: The write units per second setting for the Global
Secondary Index (GSI)'s warm throughput.
    :param region_name: The AWS Region name to target. defaults to us-east-1
    """
    try:
        ddb = client("dynamodb", region_name=region_name)

        # Define the table attributes
        attribute_definitions = [
            {"AttributeName": partition_key, "AttributeType": "S"},
            {"AttributeName": sort_key, "AttributeType": "S"},
            {"AttributeName": misc_key_attr, "AttributeType": "N"},
        ]

        # Define the table key schema
        key_schema = [
            {"AttributeName": partition_key, "KeyType": "HASH"},
            {"AttributeName": sort_key, "KeyType": "RANGE"},
        ]

        # Define the provisioned throughput for the table
        provisioned_throughput = {
            "ReadCapacityUnits": table_provisioned_read_units,
            "WriteCapacityUnits": table_provisioned_write_units,
```

```
        }

        # Define the global secondary index
        gsi_key_schema = [
            {"AttributeName": sort_key, "KeyType": "HASH"},
            {"AttributeName": misc_key_attr, "KeyType": "RANGE"},
        ]
        gsi_projection = {"ProjectionType": "INCLUDE", "NonKeyAttributes":
 [non_key_attr]}
        gsi_provisioned_throughput = {
            "ReadCapacityUnits": gsi_provisioned_read_units,
            "WriteCapacityUnits": gsi_provisioned_write_units,
        }
        gsi_warm_throughput = {
            "ReadUnitsPerSecond": gsi_warm_reads,
            "WriteUnitsPerSecond": gsi_warm_writes,
        }
        global_secondary_indexes = [
            {
                "IndexName": gsi_name,
                "KeySchema": gsi_key_schema,
                "Projection": gsi_projection,
                "ProvisionedThroughput": gsi_provisioned_throughput,
                "WarmThroughput": gsi_warm_throughput,
            }
        ]

        # Define the warm throughput for the table
        warm_throughput = {
            "ReadUnitsPerSecond": table_warm_reads,
            "WriteUnitsPerSecond": table_warm_writes,
        }

        # Create the DynamoDB client and create the table
        response = ddb.create_table(
            TableName=table_name,
            AttributeDefinitions=attribute_definitions,
            KeySchema=key_schema,
            ProvisionedThroughput=provisioned_throughput,
            GlobalSecondaryIndexes=global_secondary_indexes,
            WarmThroughput=warm_throughput,
        )

        print(response)
```

```
        return response
    except ClientError as e:
        print(f"Error creating table: {e}")
        raise e
```

- Para obter detalhes da API, consulte a CreateTableReferência da API AWS SDK for Python (Boto3).

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Java

SDK para Java 2.x

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho no Amazon DynamoDB e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em uma tabela do DynamoDB.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo no Repositório de exemplos de AWS código em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Criar uma aplicação de chat websocket com o API Gateway

O exemplo de código a seguir mostra como criar uma aplicação de chat que é atendido por uma API de Websocket criada no Amazon API Gateway.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon API Gateway V2 para criar uma API de websocket que se integre ao Amazon AWS Lambda DynamoDB.

- Crie uma API de Websocket atendida pelo API Gateway.
- Defina um manipulador do Lambda que armazena conexões no DynamoDB e publica mensagens para outros participantes do chat.
- Conecte-se à aplicação de chat websocket e envie mensagens com o pacote Websockets.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Crie um item do DynamoDB com um TTL usando um AWS SDK

Os exemplos de código a seguir mostram como criar um item com TTL.

Java

SDK para Java 2.x

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

/**
 * Creates an item in a DynamoDB table with TTL attributes.
 * This class demonstrates how to add TTL expiration timestamps to DynamoDB
 items.
 */
public class CreateTTL {

    private static final String USAGE =
        """
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
 or partition key.
                sortKey - The name of the sort key. Also known as the range
 attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table
 is located in. (Default: us-east-1)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String CREATION_DATE_ATTR = "creationDate";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String SUCCESS_MESSAGE = "%s PutItem operation with TTL
 successful.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon
 DynamoDB table \"%s\" can't be found.";

    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs a CreateTTL instance with the specified DynamoDB client.
     *
```

```
     * @param dynamoDbClient The DynamoDB client to use
     */
    public CreateTTL(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Constructs a CreateTTL with a default DynamoDB client.
     */
    public CreateTTL() {
        this.dynamoDbClient = null;
    }

    /**
     * Main method to demonstrate creating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new CreateTTL().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and create an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this
code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
```

```
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] :
null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final CreateTTL createTTL = new CreateTTL(ddb);
            createTTL.createItemWithTTL(tableName, primaryKey, sortKey);
            return 0;
        } catch (Exception e) {
            throw e;
        }
    }

    /**
     * Creates an item in the specified table with TTL attributes.
     *
     * @param tableName The name of the table
     * @param primaryKeyValue The value for the primary key
     * @param sortKeyValue The value for the sort key
     * @return The response from the PutItem operation
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     */
    public PutItemResponse createItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
sortKeyValue) {
        // Get current time in epoch second format
        final long createDate = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = createDate + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        final Map<String, AttributeValue> itemMap = new HashMap<>();
        itemMap.put(
            PRIMARY_KEY_ATTR,
AttributeValue.builder().s(primaryKeyValue).build());
        itemMap.put(SORT_KEY_ATTR,
AttributeValue.builder().s(sortKeyValue).build());
        itemMap.put(
            CREATION_DATE_ATTR,
```

```
            AttributeValue.builder().n(String.valueOf(createDate)).build());
        itemMap.put(
            EXPIRE_AT_ATTR,
            AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final PutItemRequest request =
            PutItemRequest.builder().tableName(tableName).item(itemMap).build();

        try {
            final PutItemResponse response = dynamoDbClient.putItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- Para obter detalhes da API, consulte PutItem a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

```
import { DynamoDBClient, PutItemCommand } from "@aws-sdk/client-dynamodb";

export function createDynamoDBItem(table_name, region, partition_key, sort_key) {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    // Get the current time in epoch second format
    const current_time = Math.floor(new Date().getTime() / 1000);

    // Calculate the expireAt time (90 days from now) in epoch second format
    const expire_at = Math.floor((new Date().getTime() + 90 * 24 * 60 * 60 *
 1000) / 1000);
```

```
    // Create DynamoDB item
    const item = {
        'partitionKey': {'S': partition_key},
        'sortKey': {'S': sort_key},
        'createdAt': {'N': current_time.toString()},
        'expireAt': {'N': expire_at.toString()}
    };

    const putItemCommand = new PutItemCommand({
        TableName: table_name,
        Item: item,
        ProvisionedThroughput: {
            ReadCapacityUnits: 1,
            WriteCapacityUnits: 1,
        },
    });

    client.send(putItemCommand, function(err, data) {
        if (err) {
            console.log("Exception encountered when creating item %s, here's what
 happened: ", data, err);
            throw err;
        } else {
            console.log("Item created successfully: %s.", data);
            return data;
        }
    });
}

// Example usage (commented out for testing)
// createDynamoDBItem('your-table-name', 'us-east-1', 'your-partition-key-value',
 'your-sort-key-value');
```

- Para obter detalhes da API, consulte PutItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

```
from datetime import datetime, timedelta
```

```
import boto3


def create_dynamodb_item(table_name, region, primary_key, sort_key):
    """
    Creates a DynamoDB item with an attached expiry attribute.

    :param table_name: Table name for the boto3 resource to target when creating
 an item
    :param region: string representing the AWS region. Example: `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :return: Void (nothing)
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Get the current time in epoch second format
        current_time = int(datetime.now().timestamp())

        # Calculate the expiration time (90 days from now) in epoch second format
        expiration_time = int((datetime.now() + timedelta(days=90)).timestamp())

        item = {
            "primaryKey": primary_key,
            "sortKey": sort_key,
            "creationDate": current_time,
            "expireAt": expiration_time,
        }
        response = table.put_item(Item=item)

        print("Item created successfully.")
        return response
    except Exception as e:
        print(f"Error creating item: {e}")
        raise e


# Use your own values
create_dynamodb_item(
    "your-table-name", "us-west-2", "your-partition-key-value", "your-sort-key-
value"
```

```
)
```

- Para obter detalhes da API, consulte a PutItemReferência da API AWS SDK for Python (Boto3).

Crie e gerencie tabelas globais do DynamoDB Multi-Region com forte consistência usando um AWS SDK

Os exemplos de código a seguir mostram como criar e gerenciar tabelas globais do DynamoDB Multi-Region com Strong Consistency (MRSC).

- Crie uma tabela com consistência Multi-Region forte.
- Verifique a configuração de MRSC e o status da réplica.
- Teste a consistência forte entre regiões com leituras imediatas.
- Faça gravações condicionais com garantia de MRSC.
- Limpe os recursos da tabela global com MRSC.

Bash

AWS CLI com script Bash

Crie uma tabela com consistência Multi-Region forte.

```
# Step 1: Create a new table in us-east-2 (primary region for MRSC)
# Note: Table must be empty when enabling MRSC
aws dynamodb create-table \
    --table-name MusicTable \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --region us-east-2

# Wait for table to become active
aws dynamodb wait table-exists --table-name MusicTable --region us-east-2
```

```
# Step 2: Add replica and witness with Multi-Region Strong Consistency
# MRSC requires exactly three replicas in supported regions
aws dynamodb update-table \
    --table-name MusicTable \
    --replica-updates '[{"Create": {"RegionName": "us-east-1"}}]' \
    --global-table-witness-updates '[{"Create": {"RegionName": "us-west-2"}}]' \
    --multi-region-consistency STRONG \
    --region us-east-2
```

Verifique a configuração de MRSC e o status da réplica.

```
# Verify the global table configuration and MRSC setting
aws dynamodb describe-table \
    --table-name MusicTable \
    --region us-east-2 \
    --query 'Table.
{TableName:TableName,TableStatus:TableStatus,MultiRegionConsistency:MultiRegionConsistenc
{Region:RegionName,Status:ReplicaStatus}}'
```

Teste a consistência forte com leituras imediatas entre regiões.

```
# Write an item to the primary region
aws dynamodb put-item \
    --table-name MusicTable \
    --item '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"},"Album":
 {"S":"The Beatles 1967-1970"},"Year": {"N":"1968"}}' \
    --region us-east-2

# Read the item from replica region to verify strong consistency (cannot read or
 write to witness)
# No wait time needed - MRSC provides immediate consistency
echo "Reading from us-east-1 (immediate consistency):"
aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"}}' \
    --consistent-read \
    --region us-east-1
```

Faça gravações condicionais com garantia de MRSC.

```
# Perform a conditional update from a different region
# This demonstrates that conditions work consistently across all regions
aws dynamodb update-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"}}' \
    --update-expression "SET #rating = :rating" \
    --condition-expression "attribute_exists(Artist)" \
    --expression-attribute-names '{"#rating": "Rating"}' \
    --expression-attribute-values '{":rating": {"N":"5"}}' \
    --region us-east-1
```

Limpe os recursos da tabela global com MRSC.

```
# Remove replica tables (must be done before deleting the primary table)
aws dynamodb update-table \
    --table-name MusicTable \
    --replica-updates '[{"Delete": {"RegionName": "us-east-1"}}]' \
    --global-table-witness-updates '[{"Delete": {"RegionName": "us-west-2"}}]' \
    --region us-east-2

# Wait for replicas to be deleted
echo "Waiting for replicas to be deleted..."
sleep 30

# Delete the primary table
aws dynamodb delete-table \
    --table-name MusicTable \
    --region us-east-2
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

Java

SDK para Java 2.x

Crie uma tabela regional pronta para conversão MRSC usando AWS SDK for Java 2.x.

```
   public static CreateTableResponse createRegionalTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }

       try {
           LOGGER.info("Creating regional table: " + tableName + " (must be
empty for MRSC)");

           CreateTableRequest createTableRequest = CreateTableRequest.builder()
               .tableName(tableName)
               .attributeDefinitions(
                   AttributeDefinition.builder()
                       .attributeName("Artist")
                       .attributeType(ScalarAttributeType.S)
                       .build(),
                   AttributeDefinition.builder()
                       .attributeName("SongTitle")
                       .attributeType(ScalarAttributeType.S)
                       .build())
               .keySchema(
                   KeySchemaElement.builder()
                       .attributeName("Artist")
                       .keyType(KeyType.HASH)
                       .build(),
                   KeySchemaElement.builder()
                       .attributeName("SongTitle")
                       .keyType(KeyType.RANGE)
                       .build())
               .billingMode(BillingMode.PAY_PER_REQUEST)
               .build();
```

```
            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Regional table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create regional table: " + tableName + " - "
+ e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to create regional table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Converta uma tabela regional em MRSC com réplicas e uso de testemunhas. AWS SDK for Java 2.x

```
    public static UpdateTableResponse convertToMRSCWithWitness(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
```

```
            LOGGER.info("Converting table to MRSC with replica in " +
replicaRegion.id() + " and witness in "
                + witnessRegion.id());

            // Create replica update using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate =
ReplicationGroupUpdate.builder()
                .create(CreateReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Create witness update
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .create(CreateGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .multiRegionConsistency(MultiRegionConsistency.STRONG)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC conversion initiated. Status: "
                + response.tableDescription().tableStatus());
            LOGGER.info("UpdateTableResponse full object: " + response);
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to convert table to MRSC: " + tableName + " - "
+ e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to convert table to MRSC: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Descreva uma configuração de tabela global do MRSC usando AWS SDK for Java 2.x.

```
   public static DescribeTableResponse describeMRSCTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }

       try {
           LOGGER.info("Describing MRSC global table: " + tableName);

           DescribeTableRequest request =
               DescribeTableRequest.builder().tableName(tableName).build();

           DescribeTableResponse response =
dynamoDbClient.describeTable(request);

           LOGGER.info("Table status: " + response.table().tableStatus());
           LOGGER.info("Multi-region consistency: " +
response.table().multiRegionConsistency());

           if (response.table().replicas() != null
               && !response.table().replicas().isEmpty()) {
               LOGGER.info("Number of replicas: " +
response.table().replicas().size());
               response.table()
                   .replicas()
                   .forEach(replica -> LOGGER.info(
                       "Replica region: " + replica.regionName() + ", Status: "
+ replica.replicaStatus()));
           }

           if (response.table().globalTableWitnesses() != null
               && !response.table().globalTableWitnesses().isEmpty()) {
               LOGGER.info("Number of witnesses: "
                   + response.table().globalTableWitnesses().size());
```

```
                response.table()
                    .globalTableWitnesses()
                    .forEach(witness -> LOGGER.info(
                        "Witness region: " + witness.regionName() + ", Status: "
+ witness.witnessStatus()));
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Table not found: " + tableName)
                .cause(e)
                .build();
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to describe table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Adicione itens de teste para verificar a forte consistência do MRSC usando AWS SDK for Java 2.x.

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String album,
        final String year) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
```

```
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to MRSC global table: " + tableName);

            Map<String, AttributeValue> item = new HashMap<>();
            item.put("Artist", AttributeValue.builder().s(artist).build());
            item.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            if (album != null && !album.trim().isEmpty()) {
                item.put("Album", AttributeValue.builder().s(album).build());
            }
            if (year != null && !year.trim().isEmpty()) {
                item.put("Year", AttributeValue.builder().n(year).build());
            }

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully with strong consistency");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " -
" + e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to add test item to table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Leia itens com leituras consistentes de réplicas do MRSC usando. AWS SDK for Java 2.x

```
   public static GetItemResponse getItemWithConsistentRead(
       final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }
       if (artist == null || artist.trim().isEmpty()) {
           throw new IllegalArgumentException("Artist cannot be null or empty");
       }
       if (songTitle == null || songTitle.trim().isEmpty()) {
           throw new IllegalArgumentException("Song title cannot be null or
empty");
       }

       try {
           LOGGER.info("Reading item from MRSC global table with consistent
read: " + tableName);

           Map<String, AttributeValue> key = new HashMap<>();
           key.put("Artist", AttributeValue.builder().s(artist).build());
           key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

           GetItemRequest getItemRequest = GetItemRequest.builder()
               .tableName(tableName)
               .key(key)
               .consistentRead(true)
               .build();

           GetItemResponse response = dynamoDbClient.getItem(getItemRequest);

           if (response.hasItem()) {
               LOGGER.info("Item found with strong consistency - no wait time
needed");
           } else {
               LOGGER.info("Item not found");
           }
```

```
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to read item from table: " + tableName + " - "
+ e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to read item from table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Execute atualizações condicionais com garantias do MRSC usando. AWS SDK for Java 2.x

```
    public static UpdateItemResponse performConditionalUpdate(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String rating) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }
        if (rating == null || rating.trim().isEmpty()) {
            throw new IllegalArgumentException("Rating cannot be null or empty");
        }

        try {
            LOGGER.info("Performing conditional update on MRSC global table: " +
tableName);
```

```
            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            Map<String, String> expressionAttributeNames = new HashMap<>();
            expressionAttributeNames.put("#rating", "Rating");

            Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
            expressionAttributeValues.put(
                ":rating", AttributeValue.builder().n(rating).build());

            UpdateItemRequest updateItemRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #rating = :rating")
                .conditionExpression("attribute_exists(Artist)")
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            UpdateItemResponse response =
dynamoDbClient.updateItem(updateItemRequest);
            LOGGER.info("Conditional update successful - demonstrates strong
consistency");

            return response;

        } catch (ConditionalCheckFailedException e) {
            LOGGER.warning("Conditional check failed: " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to perform conditional update: " + tableName +
" - " + e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to perform conditional update: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Aguarde até que as réplicas e testemunhas do MRSC se tornem ativas usando. AWS SDK for Java 2.x

```
   public static void waitForMRSCReplicasActive(
       final DynamoDbClient dynamoDbClient, final String tableName, final int
maxWaitTimeSeconds)
       throws InterruptedException {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }
       if (maxWaitTimeSeconds <= 0) {
           throw new IllegalArgumentException("Max wait time must be positive");
       }

       try {
           LOGGER.info("Waiting for MRSC replicas and witnesses to become
active: " + tableName);

           final long startTime = System.currentTimeMillis();
           final long maxWaitTimeMillis = maxWaitTimeSeconds * 1000L;
           int backoffSeconds = 5; // Start with 5 second intervals
           final int maxBackoffSeconds = 30; // Cap at 30 seconds

           while (System.currentTimeMillis() - startTime < maxWaitTimeMillis) {
               DescribeTableResponse response =
describeMRSCTable(dynamoDbClient, tableName);

               boolean allActive = true;
               StringBuilder statusReport = new StringBuilder();

               if (response.table().multiRegionConsistency() == null
                   || !MultiRegionConsistency.STRONG
                       .toString()

.equals(response.table().multiRegionConsistency().toString())) {
                   allActive = false;
                   statusReport
                       .append("MultiRegionConsistency: ")
```

```
                        .append(response.table().multiRegionConsistency())
                        .append(" ");
                }
                if (response.table().replicas() == null
                    || response.table().replicas().isEmpty()) {
                    allActive = false;
                    statusReport.append("No replicas found. ");
                }
                if (response.table().globalTableWitnesses() == null
                    || response.table().globalTableWitnesses().isEmpty()) {
                    allActive = false;
                    statusReport.append("No witnesses found. ");
                }

                // Check table status
                if (!"ACTIVE".equals(response.table().tableStatus().toString()))
{
                    allActive = false;
                    statusReport
                        .append("Table: ")
                        .append(response.table().tableStatus())
                        .append(" ");
                }

                // Check replica status
                if (response.table().replicas() != null) {
                    for (var replica : response.table().replicas()) {
                        if (!"ACTIVE".equals(replica.replicaStatus().toString()))
{
                            allActive = false;
                            statusReport
                                .append("Replica(")
                                .append(replica.regionName())
                                .append("): ")
                                .append(replica.replicaStatus())
                                .append(" ");
                        }
                    }
                }

                // Check witness status
                if (response.table().globalTableWitnesses() != null) {
                    for (var witness : response.table().globalTableWitnesses()) {
```

```
                        if (!"ACTIVE".equals(witness.witnessStatus().toString()))
{
                            allActive = false;
                            statusReport
                                .append("Witness(")
                                .append(witness.regionName())
                                .append("): ")
                                .append(witness.witnessStatus())
                                .append(" ");
                        }
                    }
                }

                if (allActive) {
                    LOGGER.info("All MRSC replicas and witnesses are now active:
" + tableName);
                    return;
                }

                LOGGER.info("Waiting for MRSC components to become active.
Status: " + statusReport.toString());
                LOGGER.info("Next check in " + backoffSeconds + " seconds...");

                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, maxBackoffSeconds);
            }

            throw DynamoDbException.builder()
                .message("Timeout waiting for MRSC replicas to become active
after " + maxWaitTimeSeconds + " seconds")
                .build();

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("Failed to wait for MRSC replicas to become active: " +
tableName + " - " + e.getMessage());
            throw e;
        }
    }
```

Limpe as réplicas e testemunhas do MRSC usando. AWS SDK for Java 2.x

```
    public static UpdateTableResponse cleanupMRSCReplicas(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Cleaning up MRSC replicas and witnesses for table: " +
tableName);

            // Remove replica using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate =
ReplicationGroupUpdate.builder()
                .delete(DeleteReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Remove witness
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .delete(DeleteGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
```

```
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC cleanup initiated - removing replica and witness.
Response: " + response);

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to cleanup MRSC replicas: " + tableName + " - "
+ e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to cleanup MRSC replicas: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Demonstração completa do fluxo de trabalho do MRSC usando AWS SDK for Java 2.x.

```
    public static void demonstrateCompleteMRSCWorkflow(
        final DynamoDbClient primaryClient,
        final DynamoDbClient replicaClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion)
        throws InterruptedException {

        if (primaryClient == null) {
            throw new IllegalArgumentException("Primary DynamoDB client cannot be
null");
        }
        if (replicaClient == null) {
            throw new IllegalArgumentException("Replica DynamoDB client cannot be
null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
```

```
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("=== Starting Complete MRSC Workflow Demonstration ===");

            // Step 1: Create an empty single-Region table
            LOGGER.info("Step 1: Creating empty single-Region table");
            createRegionalTable(primaryClient, tableName);

            // Use the existing GlobalTableOperations method for basic table
waiting
            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "]
to become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 2: Convert to MRSC with replica and witness
            LOGGER.info("Step 2: Converting to MRSC with replica and witness");
            convertToMRSCWithWitness(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for MRSC conversion to complete using MRSC-specific waiter
            LOGGER.info("Waiting for MRSC conversion to complete...");
            waitForMRSCReplicasActive(primaryClient, tableName);

            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "]
to become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 3: Verify MRSC configuration
            LOGGER.info("Step 3: Verifying MRSC configuration");
            describeMRSCTable(primaryClient, tableName);

            // Step 4: Test strong consistency with data operations
            LOGGER.info("Step 4: Testing strong consistency with data
operations");

            // Add test item to primary region
            putTestItem(primaryClient, tableName, "The Beatles", "Hey Jude", "The
Beatles 1967-1970", "1968");
```

```
            // Immediately read from replica region (no wait needed with MRSC)
            LOGGER.info("Reading from replica region immediately (strong
consistency):");
            GetItemResponse getResponse =
                getItemWithConsistentRead(replicaClient, tableName, "The
Beatles", "Hey Jude");

            if (getResponse.hasItem()) {
                LOGGER.info("# Strong consistency verified - item immediately
available in replica region");
            } else {
                LOGGER.warning("# Item not found in replica region");
            }

            // Test conditional update from replica region
            LOGGER.info("Testing conditional update from replica region:");
            performConditionalUpdate(replicaClient, tableName, "The Beatles",
"Hey Jude", "5");
            LOGGER.info("# Conditional update successful - demonstrates strong
consistency");

            // Step 5: Cleanup
            LOGGER.info("Step 5: Cleaning up resources");
            cleanupMRSCReplicas(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for cleanup to complete using basic table waiter
            LOGGER.info("Waiting for replica cleanup to complete...");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // "Halt" until replica/witness cleanup is complete
            DescribeTableResponse cleanupVerification =
describeMRSCTable(primaryClient, tableName);
            int backoffSeconds = 5; // Start with 5 second intervals
            while (cleanupVerification.table().multiRegionConsistency() != null)
{
                LOGGER.info("Waiting additional time (" + backoffSeconds + "
seconds) for MRSC cleanup to complete...");
                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, 30);
```

```
                cleanupVerification = describeMRSCTable(primaryClient,
tableName);
            }

            // Delete the primary table
            deleteTable(primaryClient, tableName);

            LOGGER.info("=== MRSC Workflow Demonstration Complete ===");
            LOGGER.info("");
            LOGGER.info("Key benefits of Multi-Region Strong Consistency
(MRSC):");
            LOGGER.info("- Immediate consistency across all regions (no eventual
consistency delays)");
            LOGGER.info("- Simplified application logic (no need to handle
eventual consistency)");
            LOGGER.info("- Support for conditional writes and transactions across
regions");
            LOGGER.info("- Consistent read operations from any region without
waiting");

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("MRSC workflow failed: " + e.getMessage());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

Crie e gerencie tabelas globais do DynamoDB demonstrando o MREC usando um AWS SDK

O exemplo de código a seguir mostra como criar e gerenciar tabelas globais do DynamoDB com réplicas entre várias regiões.

- Crie uma tabela com um índice secundário global e DynamoDB Streams.
- Adicione réplicas em diferentes regiões para criar uma tabela global.
- Remova réplicas de uma tabela global.
- Adicione itens de teste para verificar a replicação entre regiões.
- Descreva a configuração da tabela global e o status da réplica.

Java

SDK para Java 2.x

Crie uma tabela com o Global Secondary Index e o DynamoDB Streams usando o. AWS SDK for Java 2.x

```
   public static CreateTableResponse createTableWithGSI(
       final DynamoDbClient dynamoDbClient, final String tableName, final String
indexName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }
       if (indexName == null || indexName.trim().isEmpty()) {
           throw new IllegalArgumentException("Index name cannot be null or
empty");
       }

       try {
           LOGGER.info("Creating table: " + tableName + " with GSI: " +
indexName);

           CreateTableRequest createTableRequest = CreateTableRequest.builder()
```

```
                .tableName(tableName)
                .attributeDefinitions(
                    AttributeDefinition.builder()
                        .attributeName("Artist")
                        .attributeType(ScalarAttributeType.S)
                        .build(),
                    AttributeDefinition.builder()
                        .attributeName("SongTitle")
                        .attributeType(ScalarAttributeType.S)
                        .build())
                .keySchema(
                    KeySchemaElement.builder()
                        .attributeName("Artist")
                        .keyType(KeyType.HASH)
                        .build(),
                    KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .globalSecondaryIndexes(GlobalSecondaryIndex.builder()
                    .indexName(indexName)
                    .keySchema(KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.HASH)
                        .build())
                    .projection(

Projection.builder().projectionType(ProjectionType.ALL).build())
                    .build())
                .streamSpecification(StreamSpecification.builder()
                    .streamEnabled(true)
                    .streamViewType(StreamViewType.NEW_AND_OLD_IMAGES)
                    .build())
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
```

```
            LOGGER.severe("Failed to create table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Aguarde até que uma tabela se torne ativa usando AWS SDK for Java 2.x.

```
    public static void waitForTableActive(final DynamoDbClient dynamoDbClient,
final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Waiting for table to become active: " + tableName);

            try (DynamoDbWaiter waiter =
                DynamoDbWaiter.builder().client(dynamoDbClient).build()) {
                DescribeTableRequest request =
                    DescribeTableRequest.builder().tableName(tableName).build();

                waiter.waitUntilTableExists(request);
                LOGGER.info("Table is now active: " + tableName);
            }

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to wait for table to become active: " +
tableName + " - " + e.getMessage());
            throw e;
        }
    }
```

Adicione uma réplica para criar ou estender uma tabela global usando AWS SDK for Java 2.x.

```
    public static UpdateTableResponse addReplica(
```

```
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final String indexName,
        final Long readCapacity) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (indexName == null || indexName.trim().isEmpty()) {
            throw new IllegalArgumentException("Index name cannot be null or
empty");
        }
        if (readCapacity == null || readCapacity <= 0) {
            throw new IllegalArgumentException("Read capacity must be a positive
number");
        }

        try {
            LOGGER.info("Adding replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for adding a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .create(builder -> builder.regionName(replicaRegion.id())
                    .globalSecondaryIndexes(ReplicaGlobalSecondaryIndex.builder()
                        .indexName(indexName)

.provisionedThroughputOverride(ProvisionedThroughputOverride.builder()
                            .readCapacityUnits(readCapacity)
                            .build())
                        .build())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
```

```
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica addition initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add replica in region: " +
replicaRegion.id() + " - " + e.getMessage());
            throw e;
        }
    }
```

Remova uma réplica de uma tabela global usando o. AWS SDK for Java 2.x

```
    public static UpdateTableResponse removeReplica(
        final DynamoDbClient dynamoDbClient, final String tableName, final Region
replicaRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }

        try {
            LOGGER.info("Removing replica in region: " + replicaRegion.id() + "
for table: " + tableName);

            // Create a ReplicationGroupUpdate for removing a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
```

```
                .delete(builder ->
builder.regionName(replicaRegion.id()).build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica removal initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to remove replica in region: " +
replicaRegion.id() + " - " + e.getMessage());
            throw e;
        }
    }
```

Adicione itens de teste para verificar a replicação usando AWS SDK for Java 2.x.

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
```

```
        }

        try {
            LOGGER.info("Adding test item to table: " + tableName);

            Map<String,
software.amazon.awssdk.services.dynamodb.model.AttributeValue> item = new
HashMap<>();
            item.put(
                "Artist",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(artist)
                    .build());
            item.put(
                "SongTitle",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(songTitle)
                    .build());

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " -
" + e.getMessage());
            throw e;
        }
    }
```

Descreva a configuração e as réplicas da tabela global usando AWS SDK for Java 2.x.

```
    public static DescribeTableResponse describeTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
```

```
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Describing table: " + tableName);

            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();

            DescribeTableResponse response =
dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
                        "Replica region: " + replica.regionName() + ", Status: "
+ replica.replicaStatus()));
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " +
e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Exemplo completo de operações de tabela global usando AWS SDK for Java 2.x o.

```
   public static void exampleUsage(final Region sourceRegion, final Region
replicaRegion) {

       String tableName = "Music";
       String indexName = "SongTitleIndex";
       Long readCapacity = 15L;

       // Create DynamoDB client for the source region
       try (DynamoDbClient dynamoDbClient =
           DynamoDbClient.builder().region(sourceRegion).build()) {

           try {
               // Step 1: Create the initial table with GSI and streams
               LOGGER.info("Step 1: Creating table in source region: " +
sourceRegion.id());
               createTableWithGSI(dynamoDbClient, tableName, indexName);

               // Step 2: Wait for table to become active
               LOGGER.info("Step 2: Waiting for table to become active");
               waitForTableActive(dynamoDbClient, tableName);

               // Step 3: Add replica in destination region
               LOGGER.info("Step 3: Adding replica in region: " +
replicaRegion.id());
               addReplica(dynamoDbClient, tableName, replicaRegion, indexName,
readCapacity);

               // Step 4: Wait a moment for replica creation to start
               Thread.sleep(5000);

               // Step 5: Describe table to view replica information
               LOGGER.info("Step 5: Describing table to view replicas");
               describeTable(dynamoDbClient, tableName);

               // Step 6: Add a test item to verify replication
               LOGGER.info("Step 6: Adding test item to verify replication");
               putTestItem(dynamoDbClient, tableName, "TestArtist", "TestSong");

               LOGGER.info("Global table setup completed successfully!");
               LOGGER.info("You can verify replication by checking the item in
region: " + replicaRegion.id());

               // Step 7: Remove replica and clean up table
```

```
                LOGGER.info("Step 7: Removing replica from region: " +
replicaRegion.id());
                removeReplica(dynamoDbClient, tableName, replicaRegion);
                DeleteTableResponse deleteTableResponse =
dynamoDbClient.deleteTable(
                    DeleteTableRequest.builder().tableName(tableName).build());
                LOGGER.info("MREC global table demonstration completed
successfully!");

            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException("Thread was interrupted", e);
            } catch (DynamoDbException e) {
                LOGGER.severe("DynamoDB operation failed: " + e.getMessage());
                throw e;
            }
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTable
  - DescribeTable
  - PutItem
  - UpdateTable

Exclua dados do DynamoDB usando instruções DELETE do PartiQL com um AWS SDK

O exemplo de código a seguir mostra como excluir dados usando declarações DELETE do PartiQL.

JavaScript

SDK para JavaScript (v3)

Exclua itens de uma tabela do DynamoDB usando instruções PartiQL DELETE com. AWS SDK para JavaScript

```
/**
 * This example demonstrates how to delete items from a DynamoDB table using
 PartiQL.
 * It shows different ways to delete documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Delete a single item by its partition key using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemByPartitionKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ?`,
    Parameters: [partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item:", err);
    throw err;
  }
};

/**
```

```
 * Delete an item by its composite key (partition key + sort key) using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemByCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${sortKeyName} = ?`,
    Parameters: [partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item:", err);
    throw err;
  }
};

/**
 * Delete an item with a condition to ensure the delete only happens if a
 condition is met.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param conditionAttribute - The attribute to check in the condition
 * @param conditionValue - The value to compare against in the condition
```

```
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemWithCondition = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  conditionAttribute: string,
  conditionValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${conditionAttribute} = ?`,
    Parameters: [partitionKeyValue, conditionValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted with condition successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item with condition:", err);
    throw err;
  }
};

/**
 * Batch delete multiple items using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param keys - Array of objects containing key information
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchDeleteItems = async (
  tableName: string,
  keys: Array<{
    partitionKeyName: string;
    partitionKeyValue: string | number;
    sortKeyName?: string;
    sortKeyValue?: string | number;
  }>
) => {
```

```
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each delete
  const statements = keys.map((key) => {
    if (key.sortKeyName && key.sortKeyValue !== undefined) {
      return {
        Statement: `DELETE FROM "${tableName}" WHERE ${key.partitionKeyName} = ?
 AND ${key.sortKeyName} = ?`,
        Parameters: [key.partitionKeyValue, key.sortKeyValue],
      };
    } else {
      return {
        Statement: `DELETE FROM "${tableName}" WHERE ${key.partitionKeyName} = ?
`,
        Parameters: [key.partitionKeyValue],
      };
    }
  });

  const params = {
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items batch deleted successfully");
    return data;
  } catch (err) {
    console.error("Error batch deleting items:", err);
    throw err;
  }
};

/**
 * Delete multiple items that match a filter condition.
 * Note: This performs a scan operation which can be expensive on large tables.
 *
 * @param tableName - The name of the DynamoDB table
 * @param filterAttribute - The attribute to filter on
 * @param filterValue - The value to filter by
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemsByFilter = async (
```

```
  tableName: string,
  filterAttribute: string,
  filterValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${filterAttribute} = ?`,
    Parameters: [filterValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items deleted by filter successfully");
    return data;
  } catch (err) {
    console.error("Error deleting items by filter:", err);
    throw err;
  }
};

/**
 * Example usage showing how to delete items with different index types
 */
export const deleteExamples = async () => {
  // Delete an item by partition key (simple primary key)
  await deleteItemByPartitionKey("UsersTable", "userId", "user123");

  // Delete an item by composite key (partition key + sort key)
  await deleteItemByCompositeKey(
    "OrdersTable",
    "orderId",
    "order456",
    "productId",
    "prod789"
  );

  // Delete with a condition
  await deleteItemWithCondition(
    "UsersTable",
    "userId",
    "user789",
    "userStatus",
```

```
    "inactive"
  );

  // Batch delete multiple items
  await batchDeleteItems("UsersTable", [
    { partitionKeyName: "userId", partitionKeyValue: "user234" },
    { partitionKeyName: "userId", partitionKeyValue: "user345" },
  ]);

  // Batch delete items with composite keys
  await batchDeleteItems("OrdersTable", [
    {
      partitionKeyName: "orderId",
      partitionKeyValue: "order567",
      sortKeyName: "productId",
      sortKeyValue: "prod123",
    },
    {
      partitionKeyName: "orderId",
      partitionKeyValue: "order678",
      sortKeyName: "productId",
      sortKeyValue: "prod456",
    },
  ]);

  // Delete items by filter (use with caution)
  await deleteItemsByFilter("UsersTable", "userStatus", "deleted");
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchExecuteStatement
  - ExecuteStatement

Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Conceitos básicos de bancos de dados NoSQL

O exemplo de código a seguir mostra como:

- Criar uma tabela no DynamoDB.
- Atualizar os dados em uma tabela do DynamoDB.
- Consultar os dados em uma tabela do DynamoDB.
- Excluir uma tabela do DynamoDB.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# DynamoDB Getting Started Tutorial Script
```

```
# This script demonstrates basic operations with Amazon DynamoDB:
# - Creating a table
# - Writing data to the table
# - Reading data from the table
# - Updating data in the table
# - Querying data in the table
# - Deleting the table (cleanup)

set -uo pipefail

# Set up logging with secure permissions
LOG_DIR="${XDG_STATE_HOME:-.}/dynamodb-tutorial-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/dynamodb-tutorial-$(date +%Y%m%d-%H%M%S).log"
chmod 700 "$LOG_DIR"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting DynamoDB Getting Started Tutorial at $(date)"
echo "Logging to $LOG_FILE"

# Validate AWS CLI is configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed or not in PATH"
    exit 1
fi

# Check AWS credentials are available
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd_name=$2

    if echo "$output" | grep -qi "error\|failed"; then
        echo "ERROR detected in $cmd_name command:" >&2
        echo "$output" >&2
        return 1
    fi
```

```
    return 0
}

# Function to wait for table to be in ACTIVE state
wait_for_table_active() {
    local table_name=$1
    local max_attempts=60
    local attempt=0
    local status=""

    echo "Waiting for table $table_name to become ACTIVE..."

    while [[ "$status" != "ACTIVE" && $attempt -lt $max_attempts ]]; do
        sleep 5
        status=$(aws dynamodb describe-table --table-name "$table_name" --query
 "Table.TableStatus" --output text 2>/dev/null || echo "UNKNOWN")
        echo "Current status: $status"
        ((attempt++))
    done

    if [[ "$status" != "ACTIVE" ]]; then
        echo "ERROR: Table $table_name did not become ACTIVE within timeout
 period" >&2
        return 1
    fi

    echo "Table $table_name is now ACTIVE"
    return 0
}

# Track created resources for cleanup
declare -a RESOURCES=()

# Cleanup function
cleanup() {
    local exit_code=$?

    if [[ $exit_code -ne 0 ]]; then
        echo "Script encountered an error (exit code: $exit_code)" >&2
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP"
```

```
    echo "==========================================="
    echo "Resources to clean up:"
    for resource in "${RESOURCES[@]+"${RESOURCES[@]}"}"; do
        echo "- $resource"
    done
    echo ""

    if [[ ${#RESOURCES[@]} -gt 0 ]]; then
        echo "Proceeding with cleanup of all created resources..."

        for resource in "${RESOURCES[@]+"${RESOURCES[@]}"}"; do
            if [[ "$resource" == Table:* ]]; then
                local table_name="${resource#Table:}"
                echo "Deleting table: $table_name"
                if aws dynamodb delete-table --table-name "$table_name" 2>/dev/
null; then
                    echo "Waiting for table deletion to complete..."
                    aws dynamodb wait table-not-exists --table-name "$table_name"
 2>/dev/null || true
                else
                    echo "Warning: Failed to delete table $table_name" >&2
                fi
            fi
        done

        echo "Cleanup completed."
    fi

    return $exit_code
}

trap cleanup EXIT

# Validate table name
validate_table_name() {
    local name=$1
    if [[ ! $name =~ ^[a-zA-Z0-9._-]+$ ]] || [[ ${#name} -gt 255 ]]; then
        echo "ERROR: Invalid table name: $name" >&2
        return 1
    fi
    return 0
}

# Step 1: Create a table in DynamoDB
```

```
echo "Step 1: Creating Music table in DynamoDB..."

TABLE_NAME="Music"
validate_table_name "$TABLE_NAME"

CREATE_TABLE_OUTPUT=$(aws dynamodb create-table \
    --table-name "$TABLE_NAME" \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema AttributeName=Artist,KeyType=HASH
 AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --table-class STANDARD \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-dynamodb-gs
 2>&1) || {
    echo "ERROR: Failed to create table" >&2
    exit 1
}

check_error "$CREATE_TABLE_OUTPUT" "create-table"
echo "$CREATE_TABLE_OUTPUT"

# Add table to resources list
RESOURCES+=("Table:$TABLE_NAME")

# Wait for table to be active
wait_for_table_active "$TABLE_NAME"

# Enable point-in-time recovery (best practice)
echo "Enabling point-in-time recovery for the $TABLE_NAME table..."

PITR_OUTPUT=$(aws dynamodb update-continuous-backups \
    --table-name "$TABLE_NAME" \
    --point-in-time-recovery-specification PointInTimeRecoveryEnabled=true 2>&1)
 || {
    echo "ERROR: Failed to enable PITR" >&2
    exit 1
}

check_error "$PITR_OUTPUT" "update-continuous-backups"
echo "$PITR_OUTPUT"

# Step 2: Write data to the DynamoDB table
```

```
echo "Step 2: Writing data to the $TABLE_NAME table..."

# Use a temporary file for item data
ITEMS_TEMP=$(mktemp)
trap "rm -f '$ITEMS_TEMP'" EXIT

cat > "$ITEMS_TEMP" << 'EOF'
{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Call Me Today"},
 "AlbumTitle": {"S": "Somewhat Famous"}, "Awards": {"N": "1"}}
{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Howdy"}, "AlbumTitle":
 {"S": "Somewhat Famous"}, "Awards": {"N": "2"}}
{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}, "AlbumTitle":
 {"S": "Songs About Life"}, "Awards": {"N": "10"}}
{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "PartiQL Rocks"}, "AlbumTitle":
 {"S": "Another Album Title"}, "Awards": {"N": "8"}}
EOF

declare -i item_num=0
while IFS= read -r item_data; do
    ((item_num++))
    ITEM_OUTPUT=$(aws dynamodb put-item \
        --table-name "$TABLE_NAME" \
        --item "$item_data" 2>&1) || {
        echo "ERROR: Failed to put item $item_num" >&2
        exit 1
    }
    check_error "$ITEM_OUTPUT" "put-item (item $item_num)"
    echo "Item $item_num added successfully"
done < "$ITEMS_TEMP"

# Step 3: Read data from the DynamoDB table
echo "Step 3: Reading data from the $TABLE_NAME table..."

# Get a specific item
GET_ITEM_OUTPUT=$(aws dynamodb get-item --consistent-read \
    --table-name "$TABLE_NAME" \
    --key '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}}' 2>&1)
 || {
    echo "ERROR: Failed to get item" >&2
    exit 1
}

check_error "$GET_ITEM_OUTPUT" "get-item"
echo "Retrieved item:"
```

```
echo "$GET_ITEM_OUTPUT"

# Step 4: Update data in the DynamoDB table
echo "Step 4: Updating data in the $TABLE_NAME table..."

# Update an item
UPDATE_ITEM_OUTPUT=$(aws dynamodb update-item \
    --table-name "$TABLE_NAME" \
    --key '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}}' \
    --update-expression "SET AlbumTitle = :newval" \
    --expression-attribute-values '{":newval": {"S": "Updated Album Title"}}' \
    --return-values ALL_NEW 2>&1) || {
    echo "ERROR: Failed to update item" >&2
    exit 1
}

check_error "$UPDATE_ITEM_OUTPUT" "update-item"
echo "Updated item:"
echo "$UPDATE_ITEM_OUTPUT"

# Step 5: Query data in the DynamoDB table
echo "Step 5: Querying data in the $TABLE_NAME table..."

# Query items by Artist
QUERY_OUTPUT=$(aws dynamodb query \
    --table-name "$TABLE_NAME" \
    --key-condition-expression "Artist = :name" \
    --expression-attribute-values '{":name": {"S": "Acme Band"}}' 2>&1) || {
    echo "ERROR: Failed to query table" >&2
    exit 1
}

check_error "$QUERY_OUTPUT" "query"
echo "Query results:"
echo "$QUERY_OUTPUT"

echo "DynamoDB Getting Started Tutorial completed successfully at $(date)"
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteTable

- DescribeTable
- GetItem
- PutItem
- Consulta
- UpdateContinuousBackups
- UpdateItem
- Wait

Insira dados do DynamoDB usando instruções INSERT do PartiQL com um AWS SDK

O exemplo de código a seguir mostra como inserir dados usando declarações INSERT do PartiQL.

JavaScript

SDK para JavaScript (v3)

Insira itens em uma tabela do DynamoDB usando instruções PARTIQL INSERT com. AWS SDK para JavaScript

```
/**
 * This example demonstrates how to insert items into a DynamoDB table using
 PartiQL.
 * It shows different ways to insert documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Insert a single item into a DynamoDB table using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param item - The item to insert
 * @returns The response from the ExecuteStatementCommand
 */
```

```
export const insertItem = async (tableName: string, item: Record<string, any>) =>
 {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Convert the item to a string representation for PartiQL
  const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');

  const params = {
    Statement: `INSERT INTO "${tableName}" VALUE ${itemString}`,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item inserted successfully");
    return data;
  } catch (err) {
    console.error("Error inserting item:", err);
    throw err;
  }
};

/**
 * Insert multiple items into a DynamoDB table using PartiQL batch operation.
 * This is more efficient than inserting items one by one.
 *
 * @param tableName - The name of the DynamoDB table
 * @param items - Array of items to insert
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchInsertItems = async (tableName: string, items: Record<string,
 any>[]) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each item
  const statements = items.map((item) => {
    const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');
    return {
      Statement: `INSERT INTO "${tableName}" VALUE ${itemString}`,
    };
  });

  const params = {
```

```
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items inserted successfully");
    return data;
  } catch (err) {
    console.error("Error batch inserting items:", err);
    throw err;
  }
};

/**
 * Insert an item with a condition to prevent overwriting existing items.
 * This is useful for ensuring you don't accidentally overwrite data.
 *
 * @param tableName - The name of the DynamoDB table
 * @param item - The item to insert
 * @param partitionKeyName - The name of the partition key attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const insertItemWithCondition = async (
  tableName: string,
  item: Record<string, any>,
  partitionKeyName: string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');
  const partitionKeyValue = JSON.stringify(item[partitionKeyName]);

  const params = {
    Statement: `INSERT INTO "${tableName}" VALUE ${itemString} WHERE
 attribute_not_exists(${partitionKeyName})`,
    Parameters: [{ S: partitionKeyValue }],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item inserted with condition successfully");
    return data;
  } catch (err) {
```

```
    console.error("Error inserting item with condition:", err);
    throw err;
  }
};

/**
 * Example usage showing how to insert items with different index types
 */
export const insertExamples = async () => {
  // Example table with a simple primary key (just partition key)
  const simpleKeyItem = {
    userId: "user123",
    name: "John Doe",
    email: "john@example.com",
  };
  await insertItem("UsersTable", simpleKeyItem);

  // Example table with composite key (partition key + sort key)
  const compositeKeyItem = {
    orderId: "order456",
    productId: "prod789",
    quantity: 2,
    price: 29.99,
  };
  await insertItem("OrdersTable", compositeKeyItem);

  // Example with Global Secondary Index (GSI)
  // The GSI might be on the email attribute
  const gsiItem = {
    userId: "user789",
    email: "jane@example.com",
    name: "Jane Smith",
    userType: "premium",  // This could be part of a GSI
  };
  await insertItem("UsersTable", gsiItem);

  // Example with Local Secondary Index (LSI)
  // LSI uses the same partition key but different sort key
  const lsiItem = {
    orderId: "order567",  // Partition key
    productId: "prod123", // Sort key for the table
    orderDate: "2023-11-15", // Potential sort key for an LSI
    quantity: 1,
    price: 19.99,
```

```
  };
  await insertItem("OrdersTable", lsiItem);

  // Batch insert example with multiple items
  const batchItems = [
    {
      userId: "user234",
      name: "Alice Johnson",
      email: "alice@example.com",
    },
    {
      userId: "user345",
      name: "Bob Williams",
      email: "bob@example.com",
    },
  ];
  await batchInsertItems("UsersTable", batchItems);
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchExecuteStatement
  - ExecuteStatement

Invocar uma função do Lambda em um navegador

O exemplo de código a seguir mostra como invocar uma AWS Lambda função em um navegador.

JavaScript

SDK para JavaScript (v2)

Você pode criar um aplicativo baseado em navegador que usa uma AWS Lambda função para atualizar uma tabela do Amazon DynamoDB com as seleções do usuário.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB

- Lambda

SDK para JavaScript (v3)

Você pode criar um aplicativo baseado em navegador que usa uma AWS Lambda função para atualizar uma tabela do Amazon DynamoDB com as seleções do usuário. Este aplicativo usa AWS SDK para JavaScript v3.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda

Gerencie os índices secundários globais do DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como gerenciar o ciclo de vida completo dos índices secundários globais.

- Crie uma tabela com um índice secundário global.
- Adicione um novo GSI a uma tabela existente.
- Atualize (aumente) o throughput a quente do GSI.
- Consulte dados usando GSIs.
- Exclua um GSI.

Bash

AWS CLI com script Bash

Crie uma tabela com um índice secundário global.

```
# Create a table with a GSI
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
```

```
        AttributeName=AlbumTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "IndexName=AlbumIndex,\
        KeySchema=[{AttributeName=AlbumTitle,KeyType=HASH}],\
        Projection={ProjectionType=ALL}"
```

Adicione um novo GSI (sob demanda) a uma tabela existente.

```
# Add a new GSI to an existing table
aws dynamodb update-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Genre,AttributeType=S \
    --global-secondary-index-updates \
        "[{\"Create\":{\"IndexName\":\"GenreIndex\",\
        \"KeySchema\":[{\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"}],\
        \"Projection\":{\"ProjectionType\":\"ALL\"}}}]"
```

Atualize (aumente) o throughput a quente do GSI.

```
# Increase the warm throughput of a GSI (default values are 12k reads, 4k writes)
aws dynamodb update-table \
    --table-name MusicCollection \
    --global-secondary-index-updates \
        "[{\"Update\":{\"IndexName\":\"AlbumIndex\",\
        \"WarmThroughput\":{\"ReadUnitsPerSecond\":15000,\"WriteUnitsPerSecond
\":6000}}}]"
```

Consulte dados usando GSIs.

```
# Query the AlbumIndex GSI
aws dynamodb query \
    --table-name MusicCollection \
    --index-name AlbumIndex \
    --key-condition-expression "AlbumTitle = :album" \
```

```
    --expression-attribute-values '{":album":{"S":"Let It Be"}}'

# Query the GenreIndex GSI
aws dynamodb query \
    --table-name MusicCollection \
    --index-name GenreIndex \
    --key-condition-expression "Genre = :genre" \
    --expression-attribute-values '{":genre":{"S":"Jazz"}}'
```

Exclua um GSI.

```
# Delete a GSI from a table
aws dynamodb update-table \
    --table-name MusicCollection \
    --global-secondary-index-updates \
        "[{\"Delete\":{\"IndexName\":\"GenreIndex\"}}]"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteTable
  - Consulta
  - UpdateTable

Gerencie políticas baseadas em recursos do DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como gerenciar o ciclo de vida completo das políticas baseadas em recursos para tabelas do DynamoDB.

- Crie uma tabela com uma política de recursos.
- Obtenha uma política de recursos.
- Atualize uma política de recursos.
- Exclua uma política de recursos.

Bash

AWS CLI com script Bash

Crie uma tabela com uma política de recursos.

```
# Step 1: Create a DynamoDB table
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST

# Step 2: Create a resource-based policy document
cat > policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/DynamoDBReadOnly"
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection"
    }
  ]
}
EOF

# Step 3: Attach the resource-based policy to the table
aws dynamodb put-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
 \
    --policy file://policy.json
```

Obtenha uma política de recursos.

```
# Get the resource-based policy attached to a table
aws dynamodb get-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

Atualize uma política de recursos.

```
# Step 1: Create an updated policy document
cat > updated-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:role/DynamoDBReadOnly",
          "arn:aws:iam::123456789012:role/DynamoDBAnalytics"
        ]
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection"
    }
  ]
}
EOF

# Step 2: Update the resource-based policy on the table
aws dynamodb put-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
 \
    --policy file://updated-policy.json
```

Exclua uma política de recursos.

```
# Delete the resource-based policy from a table
aws dynamodb delete-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteResourcePolicy
  - GetResourcePolicy
  - PutResourcePolicy

Monitore o desempenho do Amazon DynamoDB usando um AWS SDK

O exemplo de código, apresentado a seguir, mostra como configurar o uso do DynamoDB por uma aplicação para monitorar o desempenho.

Java

SDK para Java 2.x

Este exemplo mostra como configurar uma aplicação em Java para monitorar o desempenho do DynamoDB. O aplicativo envia dados métricos para CloudWatch onde você pode monitorar o desempenho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch
- DynamoDB

Execute operações avançadas de consulta do DynamoDB usando um AWS SDK

Os exemplos de código a seguir mostram como realizar operações de consulta avançada no DynamoDB.

- Consulte tabelas usando várias técnicas de filtragem e condição.
- Implemente a paginação para grandes conjuntos de resultados.
- Use índices secundários globais para padrões de acesso alternativos.
- Aplique controles de consistência com base nos requisitos da aplicação.

Java

SDK para Java 2.x

Consulte com leituras fortemente consistentes usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
```

```
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
 response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

Consulte usando um índice secundário global com AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
```

```
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
```

```
            throw new DynamoDbQueryException("Table or index not found: " +
 tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI",
 e);
        }
    }
```

Consulta com paginação usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
```

```
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;
```

```
            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total
 of " + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " +
 e.getMessage());
            throw e;
        }
    }
```

Consulte com filtros complexos usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
```

```
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
 priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
 categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
 Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
 Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
 statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
 priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
 categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
```

```
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

Consulte com uma expressão de filtro construída dinamicamente usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
```

```
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName,
partitionKeyName, partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " +
e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
```

```
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " +
filterAttrName + " = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
```

```
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items
 with multiple filters:");
            multiFilterResponse.items().forEach(item ->
 System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

Consulta com uma expressão de filtro e limite de uso AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
```

```
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
```

```
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
 {0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
 filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
 e.getMessage());
            throw e;
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte com leituras fortemente consistentes usando AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with configurable read consistency
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {boolean} useConsistentRead - Whether to use strongly consistent reads
 * @returns {Promise<Object>} - The query response
 */
async function queryWithConsistentRead(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
```

```
  useConsistentRead = false
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue }
      },
      ConsistentRead: useConsistentRead
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with consistent read: ${error}`);
    throw error;
  }
}
```

Consulte usando um índice secundário global com AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table using the primary key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} userId - The user ID to query by (partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryTable(
  config,
```

```
  tableName,
  userId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the base table
    const input = {
      TableName: tableName,
      KeyConditionExpression: "user_id = :userId",
      ExpressionAttributeValues: {
        ":userId": { S: userId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying table: ${error}`);
    throw error;
  }
}

/**
 * Queries a DynamoDB Global Secondary Index (GSI)
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} indexName - The name of the GSI to query
 * @param {string} gameId - The game ID to query by (GSI partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryGSI(
  config,
  tableName,
  indexName,
  gameId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);
```

```
    // Construct the query input for the GSI
    const input = {
      TableName: tableName,
      IndexName: indexName,
      KeyConditionExpression: "game_id = :gameId",
      ExpressionAttributeValues: {
        ":gameId": { S: gameId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying GSI: ${error}`);
    throw error;
  }
}
```

Consulta com paginação usando. AWS SDK para JavaScript

```
/**
 * Example demonstrating how to handle large query result sets in DynamoDB using
 pagination
 *
 * This example shows:
 * - How to use pagination to handle large result sets
 * - How to use LastEvaluatedKey to retrieve the next page of results
 * - How to construct subsequent query requests using ExclusiveStartKey
 */
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with pagination to handle large result sets
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number} pageSize - Number of items per page
 * @returns {Promise<Array>} - All items from the query
```

```
 */
async function queryWithPagination(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  pageSize = 25
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize variables for pagination
    let lastEvaluatedKey = undefined;
    const allItems = [];
    let pageCount = 0;

    // Loop until all pages are retrieved
    do {
      // Construct the query input
      const input = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pkValue",
        Limit: pageSize,
        ExpressionAttributeNames: {
          "#pk": partitionKeyName
        },
        ExpressionAttributeValues: {
          ":pkValue": { S: partitionKeyValue }
        }
      };

      // Add ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
      if (lastEvaluatedKey) {
        input.ExclusiveStartKey = lastEvaluatedKey;
      }

      // Execute the query
      const command = new QueryCommand(input);
      const response = await client.send(command);

      // Process the current page of results
      pageCount++;
```

```
      console.log(`Processing page ${pageCount} with ${response.Items.length}
 items`);

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems.push(...response.Items);
      }

      // Get the LastEvaluatedKey for the next page
      lastEvaluatedKey = response.LastEvaluatedKey;

    } while (lastEvaluatedKey); // Continue until there are no more pages

    console.log(`Query complete. Retrieved ${allItems.length} items in
 ${pageCount} pages.`);
    return allItems;
  } catch (error) {
    console.error(`Error querying with pagination: ${error}`);
    throw error;
  }
}

/**
 * Example usage:
 *
 * // Query all items in the "AWS DynamoDB" forum with pagination
 * const allItems = await queryWithPagination(
 *   { region: "us-west-2" },
 *   "ForumThreads",
 *   "ForumName",
 *   "AWS DynamoDB",
 *   25 // 25 items per page
 * );
 *
 * console.log(`Total items retrieved: ${allItems.length}`);
 *
 * // Notes on pagination:
 * // - LastEvaluatedKey contains the primary key of the last evaluated item
 * // - When LastEvaluatedKey is undefined/null, there are no more items to
 retrieve
 * // - ExclusiveStartKey tells DynamoDB where to start the next page
 * // - Pagination helps manage memory usage for large result sets
 * // - Each page requires a separate network request to DynamoDB
 */
```

```
module.exports = { queryWithPagination };
```

Consulte com filtros complexos usando AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with a complex filter expression
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number|string} minViews - Minimum number of views for filtering
 * @param {number|string} minReplies - Minimum number of replies for filtering
 * @param {string} requiredTag - Tag that must be present in the item's tags set
 * @returns {Promise<Object>} - The query response
 */
async function queryWithComplexFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  minViews,
  minReplies,
  requiredTag
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      FilterExpression: "views >= :minViews AND replies >= :minReplies AND
 contains(tags, :tag)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
```

```
        ":pkValue": { S: partitionKeyValue },
        ":minViews": { N: minViews.toString() },
        ":minReplies": { N: minReplies.toString() },
        ":tag": { S: requiredTag }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with complex filter: ${error}`);
    throw error;
  }
}
```

Consulte com uma expressão de filtro construída dinamicamente usando AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

async function queryWithDynamicFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  sortKeyValue,
  filterParams = {}
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize filter expression components
    let filterExpressions = [];
    const expressionAttributeValues = {
      ":pkValue": { S: partitionKeyValue },
      ":skValue": { S: sortKeyValue }
    };
    const expressionAttributeNames = {
      "#pk": partitionKeyName,
```

```
      "#sk": sortKeyName
    };

    // Add status filter if provided
    if (filterParams.status) {
      filterExpressions.push("status = :status");
      expressionAttributeValues[":status"] = { S: filterParams.status };
    }

    // Add minimum views filter if provided
    if (filterParams.minViews !== undefined) {
      filterExpressions.push("views >= :minViews");
      expressionAttributeValues[":minViews"] = { N:
filterParams.minViews.toString() };
    }

    // Add author filter if provided
    if (filterParams.author) {
      filterExpressions.push("author = :author");
      expressionAttributeValues[":author"] = { S: filterParams.author };
    }

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND #sk = :skValue"
    };

    // Add filter expression if any filters were provided
    if (filterExpressions.length > 0) {
      input.FilterExpression = filterExpressions.join(" AND ");
    }

    // Add expression attribute names and values
    input.ExpressionAttributeNames = expressionAttributeNames;
    input.ExpressionAttributeValues = expressionAttributeValues;

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with dynamic filter: ${error}`);
    throw error;
  }
```

```
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte com leituras fortemente consistentes usando AWS SDK para Python (Boto3).

```
import time

import boto3
from boto3.dynamodb.conditions import Key


def query_with_consistent_read(
    table_name,
    partition_key_name,
    partition_key_value,
    sort_key_name=None,
    sort_key_value=None,
    consistent_read=True,
):
    """
    Query a DynamoDB table with the option for strongly consistent reads.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str, optional): The name of the sort key attribute.
        sort_key_value (str, optional): The value of the sort key to query.
        consistent_read (bool, optional): Whether to use strongly consistent
 reads. Defaults to True.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    if sort_key_name and sort_key_value:
        key_condition = key_condition & Key(sort_key_name).eq(sort_key_value)

    # Perform the query with the consistent read option
    response = table.query(KeyConditionExpression=key_condition,
 ConsistentRead=consistent_read)

    return response
```

Consulte usando um índice secundário global com AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Key


def query_table(table_name, partition_key_name, partition_key_value):
    """
    Query a DynamoDB table using its primary key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the table's primary key
    response =
 table.query(KeyConditionExpression=Key(partition_key_name).eq(partition_key_value))

    return response
```

```
def query_gsi(table_name, index_name, partition_key_name, partition_key_value):
    """
    Query a Global Secondary Index (GSI) on a DynamoDB table.

    Args:
        table_name (str): The name of the DynamoDB table.
        index_name (str): The name of the Global Secondary Index.
        partition_key_name (str): The name of the GSI's partition key attribute.
        partition_key_value (str): The value of the GSI's partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the GSI
    response = table.query(
        IndexName=index_name,
 KeyConditionExpression=Key(partition_key_name).eq(partition_key_value)
    )

    return response
```

Consulta com paginação usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_pagination(
    table_name, partition_key_name, partition_key_value, page_size=25,
 max_pages=None
):
    """
    Query a DynamoDB table with pagination to handle large result sets.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page.
Defaults to 25.
        max_pages (int, optional): The maximum number of pages to retrieve. If
None, retrieves all pages.

    Returns:
        list: All items retrieved from the query across all pages.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_count = 0
    all_items = []

    # Paginate through the results
    while True:
        # Check if we've reached the maximum number of pages
        if max_pages is not None and page_count >= max_pages:
            break

        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Process the current page of results
        items = response.get("Items", [])
```

```
        all_items.extend(items)

        # Update pagination tracking
        page_count += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if not last_evaluated_key:
            break

    return all_items


def query_with_pagination_generator(
    table_name, partition_key_name, partition_key_value, page_size=25
):
    """
    Query a DynamoDB table with pagination using a generator to handle large
 result sets.
    This approach is memory-efficient as it yields one page at a time.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page.
 Defaults to 25.

    Yields:
        tuple: A tuple containing (items, page_number, last_page) where:
            - items is a list of items for the current page
            - page_number is the current page number (starting from 1)
            - last_page is a boolean indicating if this is the last page
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_number = 0
```

```
    # Paginate through the results
    while True:
        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Get the current page of results
        items = response.get("Items", [])
        page_number += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # Determine if this is the last page
        is_last_page = last_evaluated_key is None

        # Yield the current page of results
        yield (items, page_number, is_last_page)

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if is_last_page:
            break
```

Consulte com filtros complexos usando AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key
```

```
def query_with_complex_filter(
    table_name,
    partition_key_name,
    partition_key_value,
    min_rating=None,
    status_list=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        min_rating (float, optional): Minimum rating value for filtering.
        status_list (list, optional): List of status values to include.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize the filter expression and expression attribute values
    filter_expression = None
    expression_attribute_values = {}

    # Build the filter expression based on provided parameters
    if min_rating is not None:
        filter_expression = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if status_list and len(status_list) > 0:
        status_condition = None
        for i, status in enumerate(status_list):
            status_value_name = f":status{i}"
            expression_attribute_values[status_value_name] = status
```

```
            if status_condition is None:
                status_condition = Attr("status").eq(status)
            else:
                status_condition = status_condition | Attr("status").eq(status)

        if filter_expression is None:
            filter_expression = status_condition
        else:
            filter_expression = filter_expression & status_condition

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if filter_expression is None:
            filter_expression = price_condition
        else:
            filter_expression = filter_expression & price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] =
 expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response


def query_with_complex_filter_and_or(
    table_name,
    partition_key_name,
    partition_key_value,
    category=None,
    min_rating=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression using AND and OR
 operators.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        category (str, optional): Category value for filtering.
        min_rating (float, optional): Minimum rating value for filtering.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build a complex filter expression with AND and OR operators
    filter_expression = None
    expression_attribute_values = {}

    # Build the category condition
    if category:
        filter_expression = Attr("category").eq(category)
        expression_attribute_values[":category"] = category

    # Build the rating and price condition (rating >= min_rating OR price <=
max_price)
    rating_price_condition = None

    if min_rating is not None:
        rating_price_condition = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if rating_price_condition is None:
            rating_price_condition = price_condition
        else:
            rating_price_condition = rating_price_condition | price_condition
```

```
    # Combine the conditions
    if rating_price_condition:
        if filter_expression is None:
            filter_expression = rating_price_condition
        else:
            filter_expression = filter_expression & rating_price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] =
 expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response
```

Consulte com uma expressão de filtro construída dinamicamente usando AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_dynamic_filter(
    table_name, partition_key_name, partition_key_value, filter_conditions=None
):
    """
    Query a DynamoDB table with a dynamically constructed filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_conditions (dict, optional): A dictionary of filter conditions
 where
```

```
            keys are attribute names and values are dictionaries with 'operator'
and 'value'.
            Example: {'rating': {'operator': '>=', 'value': 4}, 'status':
{'operator': '=', 'value': 'active'}}

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize variables for the filter expression and attribute values
    filter_expression = None
    expression_attribute_values = {":pk_val": partition_key_value}

    # Dynamically build the filter expression if filter conditions are provided
    if filter_conditions:
        for attr_name, condition in filter_conditions.items():
            operator = condition.get("operator")
            value = condition.get("value")
            attr_value_name = f":{attr_name}"
            expression_attribute_values[attr_value_name] = value

            # Create the appropriate filter expression based on the operator
            current_condition = None
            if operator == "=":
                current_condition = Attr(attr_name).eq(value)
            elif operator == "!=":
                current_condition = Attr(attr_name).ne(value)
            elif operator == ">":
                current_condition = Attr(attr_name).gt(value)
            elif operator == ">=":
                current_condition = Attr(attr_name).gte(value)
            elif operator == "<":
                current_condition = Attr(attr_name).lt(value)
            elif operator == "<=":
                current_condition = Attr(attr_name).lte(value)
            elif operator == "contains":
                current_condition = Attr(attr_name).contains(value)
            elif operator == "begins_with":
```

```
                current_condition = Attr(attr_name).begins_with(value)

            # Combine with existing filter expression using AND
            if current_condition:
                if filter_expression is None:
                    filter_expression = current_condition
                else:
                    filter_expression = filter_expression & current_condition

    # Perform the query with the dynamically built filter expression
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression

    response = table.query(**query_params)
    return response
```

Consulta com uma expressão de filtro e limite de uso AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_filter_and_limit(
    table_name,
    partition_key_name,
    partition_key_value,
    filter_attribute=None,
    filter_value=None,
    limit=10,
):
    """
    Query a DynamoDB table with a filter expression and limit the number of
 results.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
```

```
        filter_attribute (str, optional): The attribute name to filter on.
        filter_value (any, optional): The value to compare against in the filter.
        limit (int, optional): The maximum number of items to evaluate. Defaults
to 10.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition, "Limit": limit}

    # Add the filter expression if filter attributes are provided
    if filter_attribute and filter_value is not None:
        query_params["FilterExpression"] =
Attr(filter_attribute).gt(filter_value)
        query_params["ExpressionAttributeValues"] = {":filter_value":
filter_value}

    # Execute the query
    response = table.query(**query_params)
    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Execute operações de lista no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como realizar operações de lista no DynamoDB.

- Adicione elementos a um atributo de lista.
- Remova elementos de um atributo de lista.
- Atualize elementos específicos em uma lista por índice.

- Use as funções list.append e list.index.

Java

SDK para Java 2.x

Demonstre as operações de lista usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Appends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the end of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToAppend The items to append to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse appendToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToAppend) {

        // Create a list value from the items to append
```

```
        AttributeValue listValue =
AttributeValue.builder().l(itemsToAppend).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName =
list_append(if_not_exists(#attrName, :emptyList), :newItems)")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Prepends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the beginning of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToPrepend The items to prepend to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse prependToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToPrepend) {
```

```
        // Create a list value from the items to prepend
        AttributeValue listValue =
AttributeValue.builder().l(itemsToPrepend).build();

        // Define the update parameters
        // Note: To prepend, we put the new items first in the list_append
function
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName = list_append(:newItems,
if_not_exists(#attrName, :emptyList))")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates a specific element in a list attribute.
     *
     * <p>This method demonstrates how to update a specific element in a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to update
     * @param newValue The new value for the element
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
```

```
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index,
        AttributeValue newValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName[" + index + "] = :newValue")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(":newValue", newValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes a specific element from a list attribute.
     *
     * <p>This method demonstrates how to remove a specific element from a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
```

```
            .updateExpression("REMOVE #attrName[" + index + "]")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a list attribute.
     *
     * <p>Helper method to retrieve the current value of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param listAttributeName The name of the list attribute
     * @return The list attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static List<AttributeValue> getListAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String listAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(listAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the list attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(listAttributeName)) {
                return response.item().get(listAttributeName).l();
            }

            return null;
        } catch (DynamoDbException e) {
```

```
            throw DynamoDbException.builder()
                .message("Failed to get list attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Exemplo de uso de operações de lista com AWS SDK for Java 2.x.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating list operations in DynamoDB");

       try {
           // Example 1: Append items to a list
           System.out.println("\nExample 1: Appending items to a list");
           List<AttributeValue> tagsToAppend = List.of(
               AttributeValue.builder().s("Electronics").build(),
               AttributeValue.builder().s("Gadget").build());

           UpdateItemResponse appendResponse = appendToList(dynamoDbClient,
tableName, key, "Tags", tagsToAppend);

           System.out.println("Updated list attribute: " +
appendResponse.attributes());

           // Example 2: Prepend items to a list
           System.out.println("\nExample 2: Prepending items to a list");
           List<AttributeValue> tagsToPrepend = List.of(
               AttributeValue.builder().s("Featured").build(),
               AttributeValue.builder().s("New").build());

           UpdateItemResponse prependResponse = prependToList(dynamoDbClient,
tableName, key, "Tags", tagsToPrepend);

           System.out.println("Updated list attribute: " +
prependResponse.attributes());
```

```
            // Example 3: Update a specific element in a list
            System.out.println("\nExample 3: Updating a specific element in a
list");
            UpdateItemResponse updateResponse = updateListElement(
                dynamoDbClient,
                tableName,
                key,
                "Tags",
                0,
                AttributeValue.builder().s("BestSeller").build());

            System.out.println("Updated list attribute: " +
updateResponse.attributes());

            // Example 4: Remove a specific element from a list
            System.out.println("\nExample 4: Removing a specific element from a
list");
            UpdateItemResponse removeResponse = removeListElement(dynamoDbClient,
tableName, key, "Tags", 1);

            System.out.println("Updated list attribute: " +
removeResponse.attributes());

            // Example 5: Get the current value of a list attribute
            System.out.println("\nExample 5: Getting the current value of a list
attribute");
            List<AttributeValue> currentList = getListAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentList != null) {
                System.out.println("Current list attribute:");
                for (int i = 0; i < currentList.size(); i++) {
                    System.out.println("  [" + i + "]: " +
currentList.get(i).s());
                }
            } else {
                System.out.println("List attribute not found");
            }

            // Explain list operations
            System.out.println("\nKey points about DynamoDB list operations:");
            System.out.println("1. Lists are ordered collections of attributes");
            System.out.println("2. Use list_append to add items to a list");
```

```
            System.out.println("3. To append items, use list_append(existingList,
 newItems)");
            System.out.println("4. To prepend items, use list_append(newItems,
 existingList)");
            System.out.println("5. Use index notation (list[0]) to access or
 update specific elements");
            System.out.println("6. Use REMOVE to delete elements from a list");
            System.out.println("7. List indices are zero-based");
            System.out.println("8. Use if_not_exists to handle the case where the
 list doesn't exist yet");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Demonstre as operações de lista usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Append elements to a list attribute.
 *
 * This function demonstrates how to use the list_append function to add elements
 * to the end of a list.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} values - The values to append to the list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function appendToList(
  config,
  tableName,
  key,
  listName,
  values
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} =
 list_append(if_not_exists(${listName}, :empty_list), :values)`,
    ExpressionAttributeValues: {
      ":empty_list": [],
      ":values": values
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Prepend elements to a list attribute.
 *
 * This function demonstrates how to use the list_append function to add elements
 * to the beginning of a list.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
```

```
 * @param {string} listName - The name of the list attribute
 * @param {Array} values - The values to prepend to the list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function prependToList(
  config,
  tableName,
  key,
  listName,
  values
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  // Note: To prepend, we put the new values first in the list_append function
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} = list_append(:values,
 if_not_exists(${listName}, :empty_list))`,
    ExpressionAttributeValues: {
      ":empty_list": [],
      ":values": values
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update a specific element in a list by index.
 *
 * This function demonstrates how to update a specific element in a list
 * using the index notation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
```

```
 * @param {string} listName - The name of the list attribute
 * @param {number} index - The index of the element to update
 * @param {any} value - The new value for the element
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateListElement(
  config,
  tableName,
  key,
  listName,
  index,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using index notation
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName}[${index}] = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Remove an element from a list by index.
 *
 * This function demonstrates how to remove a specific element from a list
 * using the REMOVE action with index notation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
```

```
 * @param {number} index - The index of the element to remove
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function removeListElement(
  config,
  tableName,
  key,
  listName,
  index
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using REMOVE with index notation
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `REMOVE ${listName}[${index}]`,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Concatenate two lists.
 *
 * This function demonstrates how to concatenate two lists using the list_append
 function.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName1 - The name of the first list attribute
 * @param {string} listName2 - The name of the second list attribute
 * @param {string} resultListName - The name of the attribute to store the
 concatenated list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function concatenateLists(
```

```
  config,
  tableName,
  key,
  listName1,
  listName2,
  resultListName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${resultListName} =
 list_append(if_not_exists(${listName1}, :empty_list),
 if_not_exists(${listName2}, :empty_list))`,
    ExpressionAttributeValues: {
      ":empty_list": []
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Create a nested list structure.
 *
 * This function demonstrates how to create and work with nested lists.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} nestedLists - An array of arrays to create a nested list
 structure
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function createNestedList(
```

```
  config,
  tableName,
  key,
  listName,
  nestedLists
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters to create a nested list
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} = :nested_lists`,
    ExpressionAttributeValues: {
      ":nested_lists": nestedLists
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update an element in a nested list.
 *
 * This function demonstrates how to update an element in a nested list
 * using multiple index notations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} outerIndex - The index in the outer list
 * @param {number} innerIndex - The index in the inner list
 * @param {any} value - The new value for the element
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateNestedListElement(
  config,
```

```
  tableName,
  key,
  listName,
  outerIndex,
  innerIndex,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using multiple index notations
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName}[${outerIndex}][${innerIndex}] = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
```

```
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Exemplo de uso de operações de lista com AWS SDK para JavaScript.

```
/**
 * Example of how to work with lists in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "UserProfiles";
  const key = { UserId: "U12345" };

  console.log("Demonstrating list operations in DynamoDB");

  try {
    // Example 1: Append elements to a list
    console.log("\nExample 1: Appending elements to a list");
    const response1 = await appendToList(
      config,
      tableName,
      key,
      "RecentSearches",
      ["laptop", "headphones", "monitor"]
    );

    console.log("Appended to list:", response1.Attributes);
```

```
    // Example 2: Prepend elements to a list
    console.log("\nExample 2: Prepending elements to a list");
    const response2 = await prependToList(
      config,
      tableName,
      key,
      "RecentSearches",
      ["keyboard", "mouse"]
    );

    console.log("Prepended to list:", response2.Attributes);

    // Get the current state of the item
    let currentItem = await getItem(config, tableName, key);
    console.log("\nCurrent state of RecentSearches:",
currentItem?.RecentSearches);

    // Example 3: Update a specific element in a list
    console.log("\nExample 3: Updating a specific element in a list");
    const response3 = await updateListElement(
      config,
      tableName,
      key,
      "RecentSearches",
      0, // Update the first element
      "mechanical keyboard" // New value
    );

    console.log("Updated list element:", response3.Attributes);

    // Example 4: Remove an element from a list
    console.log("\nExample 4: Removing an element from a list");
    const response4 = await removeListElement(
      config,
      tableName,
      key,
      "RecentSearches",
      2 // Remove the third element
    );

    console.log("List after removing element:", response4.Attributes);

    // Example 5: Create and concatenate lists
    console.log("\nExample 5: Creating and concatenating lists");
```

```
    // First, create two separate lists
    await updateWithMultipleActions(
      config,
      tableName,
      key,
      "SET WishList = :wishlist, SavedItems = :saveditems",
      null,
      {
        ":wishlist": ["gaming laptop", "wireless earbuds"],
        ":saveditems": ["smartphone", "tablet"]
      }
    );

    // Then, concatenate them
    const response5 = await concatenateLists(
      config,
      tableName,
      key,
      "WishList",
      "SavedItems",
      "AllItems"
    );

    console.log("Concatenated lists:", response5.Attributes);

    // Example 6: Create a nested list structure
    console.log("\nExample 6: Creating a nested list structure");
    const response6 = await createNestedList(
      config,
      tableName,
      key,
      "Categories",
      [
        ["Electronics", "Computers", "Accessories"],
        ["Books", "Magazines", "E-books"],
        ["Clothing", "Shoes", "Watches"]
      ]
    );

    console.log("Created nested list:", response6.Attributes);

    // Example 7: Update an element in a nested list
    console.log("\nExample 7: Updating an element in a nested list");
```

```
    const response7 = await updateNestedListElement(
      config,
      tableName,
      key,
      "Categories",
      0, // First inner list
      1, // Second element in that list
      "Laptops" // New value
    );

    console.log("Updated nested list element:", response7.Attributes);

    // Get the final state of the item
    currentItem = await getItem(config, tableName, key);
    console.log("\nFinal state of the item:", JSON.stringify(currentItem, null,
 2));

    // Explain list operations
    console.log("\nKey points about list operations in DynamoDB:");
    console.log("1. Use list_append to add elements to a list");
    console.log("2. To append elements, use list_append(existingList,
 newElements)");
    console.log("3. To prepend elements, use list_append(newElements,
 existingList)");
    console.log("4. Use if_not_exists to handle cases where the list might not
 exist yet");
    console.log("5. Use index notation (list[0]) to access or update specific
 elements");
    console.log("6. Use REMOVE with index notation to remove elements from a
 list");
    console.log("7. Lists can contain elements of different types");
    console.log("8. Lists can be nested (lists of lists)");
    console.log("9. Use multiple index notations (list[0][1]) to access nested
 list elements");

  } catch (error) {
    console.error("Error:", error);
  }
}

/**
 * Helper function for the examples.
 *
 * @param {Object} config - AWS configuration object
```

```
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} updateExpression - The update expression
 * @param {Object} expressionAttributeNames - Expression attribute name
 placeholders
 * @param {Object} expressionAttributeValues - Expression attribute value
 placeholders
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleActions(
  config,
  tableName,
  key,
  updateExpression,
  expressionAttributeNames,
  expressionAttributeValues
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Prepare the update parameters
  const updateParams = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ReturnValues: "UPDATED_NEW"
  };

  // Add expression attribute names if provided
  if (expressionAttributeNames) {
    updateParams.ExpressionAttributeNames = expressionAttributeNames;
  }

  // Add expression attribute values if provided
  if (expressionAttributeValues) {
    updateParams.ExpressionAttributeValues = expressionAttributeValues;
  }

  // Execute the update
  const response = await docClient.send(new UpdateCommand(updateParams));

  return response;
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Demonstre as operações de lista usando AWS SDK para Python (Boto3).

```
import boto3
import json
from typing import Any, Dict, List, Optional, Union


def create_list_attribute(
    table_name: str, key: Dict[str, Any], list_name: str, list_values: List[Any]
) -> Dict[str, Any]:
    """
    Create a new list attribute or replace an existing one.

    This function demonstrates how to create a new list attribute or replace
    an existing list with new values.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        list_values (List[Any]): The values to set in the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the SET operation to create or replace the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = :list_values",
```

```
        ExpressionAttributeValues={":list_values": list_values},
        ReturnValues="UPDATED_NEW",
    )

    return response


def append_to_list(
    table_name: str, key: Dict[str, Any], list_name: str, values_to_append:
 List[Any]
) -> Dict[str, Any]:
    """
    Append values to the end of a list attribute.

    This function demonstrates how to use the list_append function to add
 elements
    to the end of a list attribute.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_append (List[Any]): The values to append to the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use list_append to add values to the end of the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = list_append({list_name}, :values)",
        ExpressionAttributeValues={":values": values_to_append},
        ReturnValues="UPDATED_NEW",
    )

    return response


def prepend_to_list(
```

```
    table_name: str, key: Dict[str, Any], list_name: str, values_to_prepend:
 List[Any]
) -> Dict[str, Any]:
    """
    Prepend values to the beginning of a list attribute.

    This function demonstrates how to use the list_append function to add
 elements
    to the beginning of a list attribute.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_prepend (List[Any]): The values to prepend to the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use list_append with reversed order to add values to the beginning of the
 list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = list_append(:values, {list_name})",
        ExpressionAttributeValues={":values": values_to_prepend},
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_list_element(
    table_name: str, key: Dict[str, Any], list_name: str, index: int, new_value:
 Any
) -> Dict[str, Any]:
    """
    Update a specific element in a list attribute.
```

```
    This function demonstrates how to update a specific element in a list
 attribute
    using the index notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        index (int): The zero-based index of the element to update.
        new_value (Any): The new value for the element.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the index notation to update a specific element
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name}[{index}] = :value",
        ExpressionAttributeValues={":value": new_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def remove_list_element(
    table_name: str, key: Dict[str, Any], list_name: str, index: int
) -> Dict[str, Any]:
    """
    Remove a specific element from a list attribute.

    This function demonstrates how to remove a specific element from a list
 attribute
    using the REMOVE action with index notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
```

```
        index (int): The zero-based index of the element to remove.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the REMOVE action with index notation to remove a specific element
    response = table.update_item(
        Key=key, UpdateExpression=f"REMOVE {list_name}[{index}]",
 ReturnValues="UPDATED_NEW"
    )

    return response


def update_nested_list_element(
    table_name: str, key: Dict[str, Any], path: str, new_value: Any
) -> Dict[str, Any]:
    """
    Update an element in a nested list structure.

    This function demonstrates how to update an element in a nested list
 structure
    using expression attribute names for the path components.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path (str): The path to the nested element (e.g., "parent[0].child[1]").
        new_value (Any): The new value for the element.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Define a type for path parts
    path_part = Dict[str, Union[str, int]]
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Parse the path to extract attribute names and indices
    path_parts: List[path_part] = []
    current_part = ""
    in_bracket = False

    for char in path:
        if char == "[":
            if current_part:
                path_parts.append({"type": "attribute", "value": current_part})
                current_part = ""
            in_bracket = True
        elif char == "]":
            if current_part:
                # Fix for mypy: Use a properly typed dictionary with Union type
                path_parts.append({"type": "index", "value": int(current_part)})
                current_part = ""
            in_bracket = False
        elif char == "." and not in_bracket:
            if current_part:
                path_parts.append({"type": "attribute", "value": current_part})
                current_part = ""
        else:
            current_part += char

    if current_part:
        path_parts.append({"type": "attribute", "value": current_part})

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        if part["type"] == "attribute":
            name_placeholder = f"#attr{i}"
            expression_attribute_names[name_placeholder] = part["value"]

            if path_expression:
                path_expression += "."
            path_expression += name_placeholder
        elif part["type"] == "index":
            path_expression += f"[{part['value']}]"
```

```
    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": new_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def create_list_if_not_exists(
    table_name: str, key: Dict[str, Any], list_name: str, default_values:
 List[Any]
) -> Dict[str, Any]:
    """
    Create a list attribute if it doesn't exist.

    This function demonstrates how to use if_not_exists to create a list
 attribute
    with default values if it doesn't already exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        default_values (List[Any]): The default values for the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use if_not_exists to create the list if it doesn't exist
    response = table.update_item(
        Key=key,
```

```
        UpdateExpression=f"SET {list_name} =
 if_not_exists({list_name}, :default)",
        ExpressionAttributeValues={":default": default_values},
        ReturnValues="UPDATED_NEW",
    )

    return response


def append_to_list_safely(
    table_name: str,
    key: Dict[str, Any],
    list_name: str,
    values_to_append: List[Any],
    default_values: Optional[List[Any]] = None,
) -> Dict[str, Any]:
    """
    Append values to a list, creating it if it doesn't exist.

    This function demonstrates how to safely append values to a list attribute,
    creating the list with default values if it doesn't exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_append (List[Any]): The values to append to the list.
        default_values (Optional[List[Any]]): The default values if the list
 doesn't exist.
            If not provided, values_to_append will be used as the default.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # If default_values is not provided, use values_to_append
    if default_values is None:
        default_values = values_to_append

    # Use if_not_exists with list_append to safely append to the list
```

```
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} =
 list_append(if_not_exists({list_name}, :default), :values)",
        ExpressionAttributeValues={
            ":default": default_values if default_values else [],
            ":values": values_to_append,
        },
        ReturnValues="UPDATED_NEW",
    )

    return response
```

Exemplo de uso de operações de lista com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use list operations in DynamoDB."""
    # Example parameters
    table_name = "UserData"
    key = {"UserId": "user123"}

    print("Example 1: Creating a list attribute")
    try:
        response = create_list_attribute(
            table_name=table_name,
            key=key,
            list_name="Interests",
            list_values=["Reading", "Hiking", "Photography"],
        )
        print(
            f"List attribute created successfully:
 {json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error creating list attribute: {e}")

    print("\nExample 2: Appending values to a list")
    try:
        response = append_to_list(
            table_name=table_name,
```

```
            key=key,
            list_name="Interests",
            values_to_append=["Cooking", "Gardening"],
        )
        print(
            f"Values appended to list successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error appending to list: {e}")

    print("\nExample 3: Prepending values to a list")
    try:
        response = prepend_to_list(
            table_name=table_name,
            key=key,
            list_name="Interests",
            values_to_prepend=["Travel", "Music"],
        )
        print(
            f"Values prepended to list successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error prepending to list: {e}")

    print("\nExample 4: Updating a specific list element")
    try:
        response = update_list_element(
            table_name=table_name,
            key=key,
            list_name="Interests",
            index=2,
            new_value="Mountain Hiking",
        )
        print(
            f"List element updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error updating list element: {e}")

    print("\nExample 5: Removing a list element")
    try:
```

```
        response = remove_list_element(
            table_name=table_name, key=key, list_name="Interests", index=0
        )
        print(
            f"List element removed successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error removing list element: {e}")

    print("\nExample 6: Working with nested lists")
    try:
        # First, create an item with a nested structure
        dynamodb = boto3.resource("dynamodb")
        table = dynamodb.Table(table_name)

        table.update_item(
            Key={"UserId": "user456"},
            UpdateExpression="SET #skills = :skills",
            ExpressionAttributeNames={"#skills": "Skills"},
            ExpressionAttributeValues={
                ":skills": [
                    {"Category": "Programming", "Languages": ["Python", "Java",
"JavaScript"]},
                    {"Category": "Database", "Systems": ["DynamoDB", "MongoDB",
"PostgreSQL"]},
                ]
            },
        )

        # Now update a nested element
        response = update_nested_list_element(
            table_name=table_name,
            key={"UserId": "user456"},
            path="Skills[0].Languages[1]",
            new_value="TypeScript",
        )
        print(
            f"Nested list element updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error working with nested lists: {e}")
```

```
    print("\nExample 7: Creating a list if it doesn't exist")
    try:
        response = create_list_if_not_exists(
            table_name=table_name,
            key={"UserId": "user789"},
            list_name="Preferences",
            default_values=["Default1", "Default2", "Default3"],
        )
        print(
            f"List created with default values:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error creating list with default values: {e}")

    print("\nExample 8: Safely appending to a list")
    try:
        response = append_to_list_safely(
            table_name=table_name,
            key={"UserId": "user789"},
            list_name="Notifications",
            values_to_append=["New message received"],
            default_values=[],
        )
        print(f"Safely appended to list: {json.dumps(response.get('Attributes',
{}), default=str)}")
    except Exception as e:
        print(f"Error safely appending to list: {e}")

    print("\nKey Points About Working with Lists in DynamoDB:")
    print("1. Lists are ordered collections of elements that can be of different
types")
    print("2. Use the SET operation with direct assignment to create or replace a
list")
    print("3. Use list_append() to add elements to a list without replacing the
entire list")
    print("4. To append to the end: list_append(list_name, :values)")
    print("5. To prepend to the beginning: list_append(:values, list_name)")
    print("6. Use index notation list_name[index] to access or update specific
elements")
    print("7. Use the REMOVE action with index notation to remove specific
elements")
    print("8. Lists can contain nested structures like maps and other lists")
```

```
    print("9. Use if_not_exists() to create a list with default values if it
 doesn't exist")
    print("10. List indices are zero-based (the first element is at index 0)")
    print("11. Attempting to access an index beyond the list bounds will result
 in an error")
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Execute operações de mapa no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como realizar operações de mapa no DynamoDB.

- Adicione e atualize atributos aninhados em estruturas de mapa.
- Remova campos específicos dos mapas.
- Trabalhe com atributos de mapa profundamente aninhados.

Java

SDK para Java 2.x

Demonstre as operações do mapa usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates a map attribute that may not exist.
     *
```

```
     * <p>This method demonstrates how to safely update a map attribute
     * by using if_not_exists to handle the case where the map doesn't exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateMapAttributeSafe(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey,
        AttributeValue value) {

        // A single UpdateExpression can't reference both a map and a path inside
        // that same map (for example "#mapName" and "#mapName.#mapKey"):
DynamoDB
        // rejects overlapping document paths with a ValidationException.
Instead,
        // perform the update in two steps.

        // Step 1: create the map with an empty value if it doesn't already
exist.
        dynamoDbClient.updateItem(UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName =
if_not_exists(#mapName, :emptyMap)")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(
                ":emptyMap", AttributeValue.builder().m(new
HashMap<>()).build()))
            .build());

        // Step 2: set the key within the now-guaranteed-to-exist map.
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
```

```
            .updateExpression("SET #mapName.#mapKey = :value")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds an attribute to a nested map.
     *
     * <p>This method demonstrates how to update a nested attribute without
     * overwriting the entire map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param path The path to the nested attribute as a list
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNestedMap(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> path,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < path.size(); i++) {
            expressionAttributeNames.put("#attr" + i, path.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < path.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
```

```
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes an attribute from a map.
     *
     * <p>This method demonstrates how to remove a specific attribute from a map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeMapAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #mapName.#mapKey")
```

```
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Creates a map with multiple attributes in a single operation.
     *
     * <p>This method demonstrates how to create a map with multiple attributes
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param attributes The attributes to set in the map
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createMapWithAttributes(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        Map<String, AttributeValue> attributes) {

        // Create a map value from the attributes
        AttributeValue mapValue = AttributeValue.builder().m(attributes).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = :mapValue")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(":mapValue", mapValue))
            .returnValues("UPDATED_NEW")
            .build();
```

```
        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a map attribute.
     *
     * <p>Helper method to retrieve the current value of a map attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param mapName The name of the map attribute
     * @return The map attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getMapAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String mapName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(mapName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the map attribute if it exists, otherwise null
            if (response.item() != null && response.item().containsKey(mapName))
{
                return response.item().get(mapName).m();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get map attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
```

```
    }
```

Exemplo de uso de operações de mapa com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating map operations in DynamoDB");

        try {
            // Example 1: Create a map with multiple attributes
            System.out.println("\nExample 1: Creating a map with multiple
attributes");
            Map<String, AttributeValue> productDetails = new HashMap<>();
            productDetails.put("Color",
AttributeValue.builder().s("Red").build());
            productDetails.put("Weight",
AttributeValue.builder().n("2.5").build());
            productDetails.put(
                "Dimensions", AttributeValue.builder().s("10x20x5").build());

            UpdateItemResponse createResponse =
                createMapWithAttributes(dynamoDbClient, tableName, key,
"Details", productDetails);

            System.out.println("Created map attribute: " +
createResponse.attributes());

            // Example 2: Update a specific attribute in a map
            System.out.println("\nExample 2: Updating a specific attribute in a
map");
            UpdateItemResponse updateResponse = updateMapAttributeSafe(
                dynamoDbClient,
                tableName,
                key,
                "Details",
                "Color",
                AttributeValue.builder().s("Blue").build());
```

```
            System.out.println("Updated map attribute: " +
updateResponse.attributes());

            // Example 3: Add an attribute to a nested map
            System.out.println("\nExample 3: Adding an attribute to a nested
map");
            UpdateItemResponse nestedResponse = addToNestedMap(
                dynamoDbClient,
                tableName,
                key,
                List.of("Specifications", "Technical", "Resolution"),
                AttributeValue.builder().s("1920x1080").build());

            System.out.println("Added to nested map: " +
nestedResponse.attributes());

            // Example 4: Remove an attribute from a map
            System.out.println("\nExample 4: Removing an attribute from a map");
            UpdateItemResponse removeResponse =
                removeMapAttribute(dynamoDbClient, tableName, key, "Details",
"Dimensions");

            System.out.println("Updated map after removal: " +
removeResponse.attributes());

            // Example 5: Get the current value of a map attribute
            System.out.println("\nExample 5: Getting the current value of a map
attribute");
            Map<String, AttributeValue> currentMap =
getMapAttribute(dynamoDbClient, tableName, key, "Details");

            if (currentMap != null) {
                System.out.println("Current map attribute:");
                for (Map.Entry<String, AttributeValue> entry :
currentMap.entrySet()) {
                    System.out.println("  " + entry.getKey() + ": " +
entry.getValue());
                }
            } else {
                System.out.println("Map attribute not found");
            }

            // Explain map operations
            System.out.println("\nKey points about DynamoDB map operations:");
```

```
            System.out.println("1. Maps are unordered collections of name-value
 pairs");
            System.out.println("2. Use dot notation (map.key) to access or update
 specific attributes");
            System.out.println("3. You can update individual attributes without
 overwriting the entire map");
            System.out.println("4. Maps can be nested to create complex data
 structures");
            System.out.println("5. Use REMOVE to delete attributes from a map");
            System.out.println("6. You can create a map with multiple attributes
 in a single operation");
            System.out.println("7. Map keys are case-sensitive");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Demonstre as operações do mapa usando AWS SDK para JavaScript.

```
/**
 * Example of updating map attributes in DynamoDB.
 *
 * This module demonstrates how to update map attributes that may not exist,
 * how to update nested attributes, and how to handle various map update
 scenarios.
 */

const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
```

```
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update a map attribute safely, handling the case where the map might not
 exist.
 *
 * This function demonstrates using the if_not_exists function to safely update
 * a map attribute that might not exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {string} mapKey - The key within the map to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMapAttributeSafe(
  config,
  tableName,
  key,
  mapName,
  mapKey,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with if_not_exists
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName}.${mapKey} = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return response;
```

```
  } catch (error) {
    // If the error is because the map doesn't exist, create it
    if (error.name === "ValidationException" &&
        error.message.includes("The document path provided in the update
 expression is invalid")) {

      // Create the map with the specified key-value pair
      const createParams = {
        TableName: tableName,
        Key: key,
        UpdateExpression: `SET ${mapName} = :map`,
        ExpressionAttributeValues: {
          ":map": { [mapKey]: value }
        },
        ReturnValues: "UPDATED_NEW"
      };

      return await docClient.send(new UpdateCommand(createParams));
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Update a map attribute using the if_not_exists function.
 *
 * This function demonstrates a more elegant approach using if_not_exists
 * to handle the case where the map doesn't exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {string} mapKey - The key within the map to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMapAttributeWithIfNotExists(
  config,
  tableName,
  key,
  mapName,
```

```
  mapKey,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // A single UpdateExpression can't reference both a map and a path inside that
  // same map (for example `${mapName}` and `${mapName}.${mapKey}`): DynamoDB
  // rejects overlapping document paths with a ValidationException. Instead,
  // perform the update in two steps.

  // Step 1: create the map with an empty value if it doesn't already exist.
  await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName} = if_not_exists(${mapName}, :emptyMap)`,
    ExpressionAttributeValues: {
      ":emptyMap": {}
    }
  }));

  // Step 2: set the key within the now-guaranteed-to-exist map.
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName}.${mapKey} = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Add a value to a deeply nested map, creating parent maps if they don't exist.
 *
 * This function demonstrates how to update a deeply nested attribute,
 * creating any parent maps that don't exist along the way.
```

```
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string[]} path - The path to the nested attribute as an array of keys
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function addToNestedMap(
  config,
  tableName,
  key,
  path,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = "SET";
  const expressionAttributeValues = {};

  // For each level in the path, create a map if it doesn't exist
  for (let i = 0; i < path.length; i++) {
    const currentPath = path.slice(0, i + 1).join(".");
    const parentPath = i > 0 ? path.slice(0, i).join(".") : null;

    if (parentPath) {
      updateExpression += ` ${parentPath} =
 if_not_exists(${parentPath}, :emptyMap${i}),`;
      expressionAttributeValues[`:emptyMap${i}`] = {};
    }
  }

  // Set the final value
  const fullPath = path.join(".");
  updateExpression += ` ${fullPath} = :value`;
  expressionAttributeValues[":value"] = value;

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
```

```
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update multiple fields in a map attribute in a single operation.
 *
 * This function demonstrates how to update multiple fields in a map
 * in a single DynamoDB operation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {Object} updates - Object containing key-value pairs to update
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMultipleMapFields(
  config,
  tableName,
  key,
  mapName,
  updates
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = `SET ${mapName} = if_not_exists(${mapName}, :emptyMap)`;
  const expressionAttributeValues = {
    ":emptyMap": {}
  };

  // Add each update to the expression
  Object.entries(updates).forEach(([field, value], index) => {
    updateExpression += `, ${mapName}.${field} = :val${index}`;
```

```
    expressionAttributeValues[`:val${index}`] = value;
  });

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
```

```
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}

/**
 * Example of how to use the map attribute update functions.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Users";
  const key = { UserId: "U12345" };

  console.log("Demonstrating different approaches to update map attributes in
 DynamoDB");

  try {
    // Example 1: Update a map attribute that might not exist (two-step approach)
    console.log("\nExample 1: Updating a map attribute that might not exist (two-
step approach)");
    const response1 = await updateMapAttributeSafe(
      config,
      tableName,
      key,
      "Preferences",
      "Theme",
      "Dark"
    );

    console.log("Updated preferences:", response1.Attributes);

    // Example 2: Update a map attribute using if_not_exists (elegant approach)
    console.log("\nExample 2: Updating a map attribute using if_not_exists
 (elegant approach)");
    const response2 = await updateMapAttributeWithIfNotExists(
      config,
      tableName,
      key,
      "Settings",
      "NotificationsEnabled",
      true
    );
```

```
    console.log("Updated settings:", response2.Attributes);

    // Example 3: Update a deeply nested attribute
    console.log("\nExample 3: Updating a deeply nested attribute");
    const response3 = await addToNestedMap(
      config,
      tableName,
      key,
      ["Profile", "Address", "City"],
      "Seattle"
    );

    console.log("Updated nested attribute:", response3.Attributes);

    // Example 4: Update multiple fields in a map
    console.log("\nExample 4: Updating multiple fields in a map");
    const response4 = await updateMultipleMapFields(
      config,
      tableName,
      key,
      "ContactInfo",
      {
        Email: "user@example.com",
        Phone: "555-123-4567",
        PreferredContact: "Email"
      }
    );

    console.log("Updated multiple fields:", response4.Attributes);

    // Get the final state of the item
    console.log("\nFinal state of the item:");
    const item = await getItem(config, tableName, key);
    console.log(JSON.stringify(item, null, 2));

    // Explain the benefits of different approaches
    console.log("\nKey points about updating map attributes:");
    console.log("1. Use if_not_exists to handle maps that might not exist");
    console.log("2. Multiple updates can be combined in a single operation");
    console.log("3. Deeply nested attributes require creating parent maps");
    console.log("4. DynamoDB expressions are atomic - the entire update succeeds
or fails");
```

```
    console.log("5. Using a single operation is more efficient than multiple
 separate updates");

  } catch (error) {
    console.error("Error:", error);
  }
}

// Export the functions
module.exports = {
  updateMapAttributeSafe,
  updateMapAttributeWithIfNotExists,
  addToNestedMap,
  updateMultipleMapFields,
  getItem,
  exampleUsage
};

// Run the example if this file is executed directly
if (require.main === module) {
  exampleUsage();
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Demonstre as operações do mapa usando AWS SDK para Python (Boto3).

```
"""
Example of updating map attributes in DynamoDB.

This module demonstrates how to update map attributes in DynamoDB, including
handling cases where the map attribute might not exist yet.
"""


import boto3
```

```
from typing import Any, Dict, Optional



def update_map_attribute_safe(
    table_name: str, key: Dict[str, Any], map_name: str, map_key: str, value: Any
) -> Dict[str, Any]:
    """
    Update a specific key in a map attribute, creating the map if it doesn't
 exist.

    This function demonstrates how to safely update a key within a map attribute,
    even if the map doesn't exist yet in the item.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        map_key (str): The key within the map to update.
        value (Any): The value to set for the map key.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use SET with attribute_not_exists to safely update the map
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #map.#key = :value",
        ExpressionAttributeNames={"#map": map_name, "#key": map_key},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def add_to_nested_map(
    table_name: str, key: Dict[str, Any], path: str, value: Any
) -> Dict[str, Any]:
```

```
    """
    Add or update a value in a deeply nested map structure.

    This function demonstrates how to update a value at a specific path in a
    nested map structure, creating any intermediate maps as needed.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path (str): The path to the nested attribute (e.g.,
"user.preferences.theme").
        value (Any): The value to set at the specified path.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Split the path into components
    path_parts = path.split(".")

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        name_placeholder = f"#attr{i}"
        expression_attribute_names[name_placeholder] = part

        if i == 0:
            path_expression = name_placeholder
        else:
            path_expression += f".{name_placeholder}"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
```

```
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_map_with_if_not_exists(
    table_name: str,
    key: Dict[str, Any],
    map_name: str,
    map_key: str,
    value: Any,
    default_map: Optional[Dict[str, Any]] = None,
) -> Dict[str, Any]:
    """
    Update a key in a map, creating the map with default values if it doesn't
 exist.

    This function demonstrates how to use if_not_exists to initialize a map with
    default values if it doesn't exist yet, and then update a specific key.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        map_key (str): The key within the map to update.
        value (Any): The value to set for the map key.
        default_map (Optional[Dict[str, Any]]): Default map values if the map
 doesn't exist.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Set default map if not provided
    if default_map is None:
```

```
        default_map = {}

    # Create a map with the new key-value pair
    updated_map = default_map.copy()
    updated_map[map_key] = value

    # Use if_not_exists to initialize the map if it doesn't exist
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #map = if_not_exists(#map, :default_map)",
        ExpressionAttributeNames={"#map": map_name},
        ExpressionAttributeValues={":default_map": updated_map},
        ReturnValues="UPDATED_NEW",
    )

    return response


def merge_into_map(
    table_name: str, key: Dict[str, Any], map_name: str, values_to_merge:
 Dict[str, Any]
) -> Dict[str, Any]:
    """
    Merge multiple key-value pairs into a map attribute.

    This function demonstrates how to update multiple keys in a map attribute
    in a single operation, without overwriting the entire map.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        values_to_merge (Dict[str, Any]): Key-value pairs to merge into the map.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the update expression for each key-value pair
    update_expression = "SET "
```

```
    expression_attribute_names = {"#map": map_name}
    expression_attribute_values = {}

    # Add each key-value pair to the update expression
    for i, (k, v) in enumerate(values_to_merge.items()):
        key_placeholder = f"#key{i}"
        value_placeholder = f":value{i}"

        expression_attribute_names[key_placeholder] = k
        expression_attribute_values[value_placeholder] = v

        if i > 0:
            update_expression += ", "
        update_expression += f"#map.{key_placeholder} = {value_placeholder}"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues="UPDATED_NEW",
    )

    return response




def example_usage():
    """Example of how to use the map attribute update functions."""
    # Example parameters
    table_name = "UserProfiles"
    key = {"UserId": "user123"}

    print("Example 1: Updating a specific key in a map attribute")
    try:
        response = update_map_attribute_safe(
            table_name=table_name, key=key, map_name="Preferences",
 map_key="Theme", value="Dark"
        )
        print(f"Map attribute updated successfully: {response.get('Attributes',
 {})}")
    except Exception as e:
```

```
        print(f"Error updating map attribute: {e}")

    print("\nExample 2: Adding a value to a deeply nested map")
    try:
        response = add_to_nested_map(
            table_name=table_name, key=key, path="Settings.Notifications.Email",
value=True
        )
        print(f"Nested map updated successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error updating nested map: {e}")

    print("\nExample 3: Initializing a map with default values if it doesn't
exist")
    try:
        default_map = {"Language": "English", "Currency": "USD", "TimeZone":
"UTC"}

        response = update_map_with_if_not_exists(
            table_name=table_name,
            key={"UserId": "newuser456"},
            map_name="Preferences",
            map_key="Theme",
            value="Light",
            default_map=default_map,
        )
        print(f"Map initialized with defaults: {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error initializing map: {e}")

    print("\nExample 4: Merging multiple values into a map")
    try:
        values_to_merge = {
            "NotificationsEnabled": True,
            "EmailFrequency": "Daily",
            "PushNotifications": False,
        }

        response = merge_into_map(
            table_name=table_name,
            key=key,
            map_name="NotificationSettings",
            values_to_merge=values_to_merge,
```

```
        )
        print(f"Multiple values merged into map: {response.get('Attributes',
 {})}")
    except Exception as e:
        print(f"Error merging values into map: {e}")

    print("\nBest practices for working with map attributes in DynamoDB:")
    print("1. Use dot notation to access and update nested attributes")
    print("2. Use ExpressionAttributeNames to handle reserved words and special
 characters")
    print("3. Use if_not_exists() to handle cases where attributes might not
 exist")
    print("4. Update specific map keys rather than overwriting the entire map")
    print("5. Use a single update operation to modify multiple map keys for
 better performance")
    print("6. Consider your data model carefully to minimize the need for deeply
 nested attributes")



if __name__ == "__main__":
    example_usage()
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Execute operações de conjunto no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como realizar operações de conjunto no DynamoDB.

- Adicione elementos a um atributo de conjunto.
- Remova elementos de um atributo de conjunto.
- Use as operações ADD e DELETE com conjuntos.

Java

SDK para Java 2.x

Demonstre as operações definidas usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

    /**
     * Adds values to a string set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a string set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToStringSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<String> valuesToAdd) {

        // Create a string set value from the values to add
        AttributeValue setValue =
 AttributeValue.builder().ss(valuesToAdd).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
```

```
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds values to a number set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a number set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNumberSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<Number> valuesToAdd) {

        // Convert numbers to strings for DynamoDB
        Set<String> stringValues = new HashSet<>();
        for (Number value : valuesToAdd) {
            stringValues.add(value.toString());
        }

        // Create a number set value from the values to add
        AttributeValue setValue =
AttributeValue.builder().ns(stringValues).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
```

```
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes values from a set attribute.
     *
     * <p>This method demonstrates how to use the DELETE operation to remove
values
     * from a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToRemove The values to remove from the set
     * @param isNumberSet Whether the set is a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeFromSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> valuesToRemove,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
```

```
                } else {
                    throw new IllegalArgumentException("Values must be numbers
for a number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("DELETE #setAttr :valuesToRemove")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToRemove", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Checks if a value exists in a set attribute.
     *
     * <p>This method demonstrates how to use the contains function to check
     * if a value exists in a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to check
     * @param setAttributeName The name of the set attribute
     * @param valueToCheck The value to check for
     * @return Map containing the result of the check
     * @throws DynamoDbException if an error occurs during the operation
```

```
     */
    public static Map<String, Object> checkIfValueInSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        String valueToCheck) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters with a condition expression
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #tempAttr = :tempVal")
                .conditionExpression("contains(#setAttr, :valueToCheck)")
                .expressionAttributeNames(Map.of("#setAttr", setAttributeName,
"#tempAttr", "TempAttribute"))
                .expressionAttributeValues(Map.of(
                    ":valueToCheck",
AttributeValue.builder().s(valueToCheck).build(),
                    ":tempVal", AttributeValue.builder().s("TempValue").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Attempt the update operation
            dynamoDbClient.updateItem(request);

            // If we get here, the condition was met
            result.put("exists", true);
            result.put("message", "Value '" + valueToCheck + "' exists in the
set");

            // Clean up the temporary attribute
            UpdateItemRequest cleanupRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("REMOVE #tempAttr")
                .expressionAttributeNames(Map.of("#tempAttr", "TempAttribute"))
                .build();

            dynamoDbClient.updateItem(cleanupRequest);
```

```
        } catch (DynamoDbException e) {
            if (e.getMessage().contains("ConditionalCheckFailed")) {
                // The condition was not met
                result.put("exists", false);
                result.put("message", "Value '" + valueToCheck + "' does not
exist in the set");
            } else {
                // Some other error occurred
                result.put("exists", false);
                result.put("message", "Error checking set: " + e.getMessage());
                result.put("error", e.getClass().getSimpleName());
            }
        }

        return result;
    }

    /**
     * Creates a set with multiple values in a single operation.
     *
     * <p>This method demonstrates how to create a set with multiple values
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param setValues The values to include in the set
     * @param isNumberSet Whether to create a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createSetWithValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> setValues,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
```

```
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers
for a number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a set attribute.
     *
     * <p>Helper method to retrieve the current value of a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
```

```
     * @param setAttributeName The name of the set attribute
     * @return The set attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static AttributeValue getSetAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String setAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(setAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the set attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(setAttributeName)) {
                return response.item().get(setAttributeName);
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get set attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Exemplo de uso de operações definidas com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());
```

```
        System.out.println("Demonstrating set operations in DynamoDB");

        try {
            // Example 1: Create a string set with multiple values
            System.out.println("\nExample 1: Creating a string set with multiple
values");
            Set<String> tags = new HashSet<>();
            tags.add("Electronics");
            tags.add("Gadget");
            tags.add("Smartphone");

            UpdateItemResponse createResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Tags", tags, false // Not a
number set
                );

            System.out.println("Created set attribute: " +
createResponse.attributes());

            // Example 2: Add values to a string set
            System.out.println("\nExample 2: Adding values to a string set");
            Set<String> additionalTags = new HashSet<>();
            additionalTags.add("Mobile");
            additionalTags.add("Wireless");

            UpdateItemResponse addResponse = addToStringSet(dynamoDbClient,
tableName, key, "Tags", additionalTags);

            System.out.println("Updated set attribute: " +
addResponse.attributes());

            // Example 3: Create a number set with multiple values
            System.out.println("\nExample 3: Creating a number set with multiple
values");
            Set<Number> ratings = new HashSet<>();
            ratings.add(4);
            ratings.add(5);
            ratings.add(4.5);

            UpdateItemResponse createNumberSetResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Ratings", ratings, true // Is a
number set
                );
```

```
            System.out.println("Created number set attribute: " +
createNumberSetResponse.attributes());

            // Example 4: Add values to a number set
            System.out.println("\nExample 4: Adding values to a number set");
            Set<Number> additionalRatings = new HashSet<>();
            additionalRatings.add(3.5);
            additionalRatings.add(4.2);

            UpdateItemResponse addNumberResponse =
                addToNumberSet(dynamoDbClient, tableName, key, "Ratings",
additionalRatings);

            System.out.println("Updated number set attribute: " +
addNumberResponse.attributes());

            // Example 5: Remove values from a set
            System.out.println("\nExample 5: Removing values from a set");
            Set<String> tagsToRemove = new HashSet<>();
            tagsToRemove.add("Gadget");

            UpdateItemResponse removeResponse = removeFromSet(
                dynamoDbClient, tableName, key, "Tags", tagsToRemove, false //
Not a number set
                );

            System.out.println("Updated set after removal: " +
removeResponse.attributes());

            // Example 6: Check if a value exists in a set
            System.out.println("\nExample 6: Checking if a value exists in a
set");
            Map<String, Object> checkResult = checkIfValueInSet(dynamoDbClient,
tableName, key, "Tags", "Electronics");

            System.out.println("Check result: " + checkResult.get("message"));

            // Example 7: Get the current value of a set attribute
            System.out.println("\nExample 7: Getting the current value of a set
attribute");
            AttributeValue currentStringSet = getSetAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentStringSet != null && currentStringSet.ss() != null) {
```

```
                System.out.println("Current string set values: " +
currentStringSet.ss());
            } else {
                System.out.println("String set attribute not found");
            }

            AttributeValue currentNumberSet = getSetAttribute(dynamoDbClient,
tableName, key, "Ratings");

            if (currentNumberSet != null && currentNumberSet.ns() != null) {
                System.out.println("Current number set values: " +
currentNumberSet.ns());
            } else {
                System.out.println("Number set attribute not found");
            }

            // Explain set operations
            System.out.println("\nKey points about DynamoDB set operations:");
            System.out.println(
                "1. DynamoDB supports three set types: string sets (SS), number
sets (NS), and binary sets (BS)");
            System.out.println("2. Sets can only contain elements of the same
type");
            System.out.println("3. Use ADD to add elements to a set");
            System.out.println("4. Use DELETE to remove elements from a set");
            System.out.println("5. Sets automatically remove duplicate values");
            System.out.println("6. Sets are unordered collections");
            System.out.println("7. Use the contains function to check if a value
exists in a set");
            System.out.println("8. You can create a set with multiple values in a
single operation");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Demonstre as operações definidas usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Add elements to a set attribute.
 *
 * This function demonstrates using the ADD operation to add elements to a set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The values to add to the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function addToSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
```

```
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using ADD
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `ADD ${setName} :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Remove elements from a set attribute.
 *
 * This function demonstrates using the DELETE operation to remove elements from
 a set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The values to remove from the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function removeFromSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
```

```
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using DELETE
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `DELETE ${setName} :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Create a new set attribute with initial values.
 *
 * This function demonstrates using the SET operation to create a new set
 attribute.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
```

```
 * @param {Array} values - The initial values for the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function createSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using SET
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${setName} = :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}
```

```
/**
 * Replace an entire set attribute with a new set of values.
 *
 * This function demonstrates using the SET operation to replace an entire set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The new values for the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function replaceSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // This is the same as createSet, but included for clarity of intent
  return await createSet(config, tableName, key, setName, values, setType);
}

/**
 * Remove the last element from a set and handle the empty set case.
 *
 * This function demonstrates what happens when you delete the last element of a
 set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @returns {Promise<Object>} - The result of the operation
 */
async function removeLastElementFromSet(
  config,
  tableName,
  key,
  setName
) {
```

```
// Initialize the DynamoDB client
const client = new DynamoDBClient(config);
const docClient = DynamoDBDocumentClient.from(client);

// First, get the current item to check the set
const currentItem = await getItem(config, tableName, key);

// Check if the set exists and has elements
if (!currentItem || !currentItem[setName] || currentItem[setName].size === 0) {
  return {
    success: false,
    message: "Set doesn't exist or is already empty",
    item: currentItem
  };
}

// Get the set values
const setValues = Array.from(currentItem[setName]);

// If there's only one element left, remove the attribute entirely
if (setValues.length === 1) {
  // Define the update parameters to remove the attribute
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `REMOVE ${setName}`,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  await docClient.send(new UpdateCommand(params));

  return {
    success: true,
    message: "Last element removed, attribute has been deleted",
    removedValue: setValues[0]
  };
} else {
  // Otherwise, remove just the last element
  // Create a set with just the last element
  const lastElement = setValues[setValues.length - 1];
  const setType = typeof lastElement === 'number' ? 'number' : 'string';

  // Remove the last element
```

```
    const response = await removeFromSet(
      config,
      tableName,
      key,
      setName,
      [lastElement],
      setType
    );

    return {
      success: true,
      message: "Last element removed, set still contains elements",
      removedValue: lastElement,
      remainingSet: response.Attributes[setName]
    };
  }
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
```

```
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Exemplo de uso de operações definidas com AWS SDK para JavaScript.

```
/**
 * Example of how to work with sets in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Users";
  const key = { UserId: "U12345" };

  console.log("Demonstrating set operations in DynamoDB");

  try {
    // Example 1: Create a string set
    console.log("\nExample 1: Creating a string set");
    const response1 = await createSet(
      config,
      tableName,
      key,
      "Interests",
      ["Reading", "Hiking", "Cooking"],
      "string"
    );

    console.log("Created set:", response1.Attributes);

    // Example 2: Add elements to a set
    console.log("\nExample 2: Adding elements to a set");
    const response2 = await addToSet(
      config,
      tableName,
      key,
      "Interests",
      ["Photography", "Travel"],
      "string"
```

```
    );

    console.log("Updated set after adding elements:", response2.Attributes);

    // Example 3: Remove elements from a set
    console.log("\nExample 3: Removing elements from a set");
    const response3 = await removeFromSet(
      config,
      tableName,
      key,
      "Interests",
      ["Cooking"],
      "string"
    );

    console.log("Updated set after removing elements:", response3.Attributes);

    // Example 4: Create a number set
    console.log("\nExample 4: Creating a number set");
    const response4 = await createSet(
      config,
      tableName,
      key,
      "FavoriteNumbers",
      [7, 42, 99],
      "number"
    );

    console.log("Created number set:", response4.Attributes);

    // Example 5: Replace an entire set
    console.log("\nExample 5: Replacing an entire set");
    const response5 = await replaceSet(
      config,
      tableName,
      key,
      "Interests",
      ["Gaming", "Movies", "Music"],
      "string"
    );

    console.log("Replaced set:", response5.Attributes);

    // Example 6: Remove the last element from a set
```

```
    console.log("\nExample 6: Removing the last element from a set");

    // First, create a set with just one element
    await createSet(
      config,
      tableName,
      { UserId: "U67890" },
      "Tags",
      ["LastTag"],
      "string"
    );

    // Then, remove the last element
    const response6 = await removeLastElementFromSet(
      config,
      tableName,
      { UserId: "U67890" },
      "Tags"
    );

    console.log(response6.message);
    console.log("Removed value:", response6.removedValue);

    // Get the final state of the items
    console.log("\nFinal state of the items:");
    const item1 = await getItem(config, tableName, key);
    console.log("User U12345:", JSON.stringify(item1, null, 2));

    const item2 = await getItem(config, tableName, { UserId: "U67890" });
    console.log("User U67890:", JSON.stringify(item2, null, 2));

    // Explain set operations
    console.log("\nKey points about set operations in DynamoDB:");
    console.log("1. Use ADD to add elements to a set (duplicates are
automatically removed)");
    console.log("2. Use DELETE to remove elements from a set");
    console.log("3. Use SET to create a new set or replace an existing one");
    console.log("4. DynamoDB supports three types of sets: string sets, number
sets, and binary sets");
    console.log("5. When you delete the last element from a set, the attribute
remains as an empty set");
    console.log("6. To remove an empty set, use the REMOVE operation");
    console.log("7. Sets automatically maintain unique values (no duplicates)");
    console.log("8. You cannot mix data types within a set");
```

```
  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Demonstre as operações definidas usando AWS SDK para Python (Boto3).

```
import boto3
from typing import Any, Dict, List


def create_set_attribute(
    table_name: str,
    key: Dict[str, Any],
    set_name: str,
    set_values: List[Any],
    set_type: str = "string",
) -> Dict[str, Any]:
    """
    Create a new set attribute or add elements to an existing set.

    This function demonstrates how to use the ADD operation to create a new set
    or add elements to an existing set.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        set_values (List[Any]): The values to add to the set.
        set_type (str, optional): The type of set to create: "string", "number",
 or "binary".
            Defaults to "string".

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a DynamoDB set based on the specified type
    if set_type == "string":
        dynamo_set = set(str(value) for value in set_values)
    elif set_type == "number":
        # We need to use actual float values for the DynamoDB API
        # but mypy expects strings in sets, so we need to use type: ignore
        dynamo_set = set(float(value) for value in set_values)  # type: ignore
    else:  # binary set is not directly supported in high-level API, handled
 differently
        raise ValueError("Binary sets are not supported in this example")

    # Use the ADD operation to create or update the set
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #set_attr :set_values",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":set_values": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def add_to_set(
    table_name: str, key: Dict[str, Any], set_name: str, values_to_add: List[Any]
) -> Dict[str, Any]:
    """
    Add elements to an existing set attribute.

    This function demonstrates how to use the ADD operation to add elements to an
 existing set.
    If the set doesn't exist, it will be created.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
```

```
        values_to_add (List[Any]): The values to add to the set.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a set (assuming string set for simplicity)
    dynamo_set = set(str(value) for value in values_to_add)

    # Use the ADD operation to add values to the set
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #set_attr :values_to_add",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":values_to_add": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def remove_from_set(
    table_name: str, key: Dict[str, Any], set_name: str, values_to_remove:
 List[Any]
) -> Dict[str, Any]:
    """
    Remove elements from a set attribute.

    This function demonstrates how to use the DELETE operation to remove elements
 from a set.
    If the last element is removed, the attribute will be deleted entirely.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        values_to_remove (List[Any]): The values to remove from the set.

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a set (assuming string set for simplicity)
    dynamo_set = set(str(value) for value in values_to_remove)

    # Use the DELETE operation to remove values from the set
    response = table.update_item(
        Key=key,
        UpdateExpression="DELETE #set_attr :values_to_remove",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":values_to_remove": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def check_if_set_exists(table_name: str, key: Dict[str, Any], set_name: str) ->
 bool:
    """
    Check if a set attribute exists in an item.

    This function demonstrates how to check if a set attribute exists after
    potentially removing all elements from it.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to check.
        set_name (str): The name of the set attribute.

    Returns:
        bool: True if the set attribute exists, False otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Get the item
    response = table.get_item(
```

```
        Key=key, ProjectionExpression="#set_attr",
 ExpressionAttributeNames={"#set_attr": set_name}
    )

    # Check if the item exists and has the set attribute
    return "Item" in response and set_name in response["Item"]


def demonstrate_last_element_removal(
    table_name: str, key: Dict[str, Any], set_name: str
) -> Dict[str, Any]:
    """
    Demonstrate what happens when you remove the last element from a set.

    This function creates a set with a single element, then removes that element,
    showing that the attribute is completely removed when the last element is
 deleted.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Step 1: Create a set with a single element
    create_response = create_set_attribute(
        table_name=table_name,
        key=key,
        set_name=set_name,
        set_values=["last_element"],
        set_type="string",
    )

    # Step 2: Check that the set exists
    exists_before = check_if_set_exists(table_name, key, set_name)

    # Step 3: Remove the last element
    delete_response = remove_from_set(
        table_name=table_name, key=key, set_name=set_name,
 values_to_remove=["last_element"]
    )
```

```
    # Step 4: Check if the set still exists
    exists_after = check_if_set_exists(table_name, key, set_name)

    # Return the results
    return {
        "create_response": create_response,
        "exists_before": exists_before,
        "delete_response": delete_response,
        "exists_after": exists_after,
    }


def work_with_number_set(
    table_name: str,
    key: Dict[str, Any],
    set_name: str,
    initial_values: List[float],
    values_to_add: List[float],
    values_to_remove: List[float],
) -> Dict[str, Any]:
    """
    Demonstrate working with a number set in DynamoDB.

    This function shows how to create and manipulate a set of numbers.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        initial_values (List[float]): The initial values for the set.
        values_to_add (List[float]): Values to add to the set.
        values_to_remove (List[float]): Values to remove from the set.

    Returns:
        Dict[str, Any]: A dictionary containing the responses from each
 operation.
    """
    # Step 1: Create the number set
    create_response = create_set_attribute(
        table_name=table_name,
        key=key,
        set_name=set_name,
        set_values=initial_values,
        set_type="number",
```

```
    )

    # Step 2: Add more numbers to the set
    add_response = add_to_set(
        table_name=table_name, key=key, set_name=set_name,
values_to_add=values_to_add
    )

    # Step 3: Remove some numbers from the set
    remove_response = remove_from_set(
        table_name=table_name, key=key, set_name=set_name,
values_to_remove=values_to_remove
    )

    # Step 4: Get the final state
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    get_response = table.get_item(
        Key=key,
        ProjectionExpression=f"#{set_name}",
        ExpressionAttributeNames={f"#{set_name}": set_name},
    )

    # Return all responses
    return {
        "create_response": create_response,
        "add_response": add_response,
        "remove_response": remove_response,
        "final_state": get_response.get("Item", {}),
    }
```

Exemplo de uso de operações definidas com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the set operations functions."""
    # Example parameters
    table_name = "UserPreferences"
    key = {"UserId": "user123"}
```

```
    print("Example 1: Creating a string set attribute")
    try:
        response = create_set_attribute(
            table_name=table_name,
            key=key,
            set_name="FavoriteTags",
            set_values=["AWS", "DynamoDB", "NoSQL"],
            set_type="string",
        )
        print(f"Set attribute created successfully: {response.get('Attributes',
 {})}")
    except Exception as e:
        print(f"Error creating set attribute: {e}")

    print("\nExample 2: Adding elements to an existing set")
    try:
        response = add_to_set(
            table_name=table_name,
            key=key,
            set_name="FavoriteTags",
            values_to_add=["Database", "Serverless"],
        )
        print(f"Elements added to set successfully: {response.get('Attributes',
 {})}")
    except Exception as e:
        print(f"Error adding to set: {e}")

    print("\nExample 3: Removing elements from a set")
    try:
        response = remove_from_set(
            table_name=table_name, key=key, set_name="FavoriteTags",
 values_to_remove=["NoSQL"]
        )
        print(f"Elements removed from set successfully:
 {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error removing from set: {e}")

    print("\nExample 4: Demonstrating what happens when you remove the last
 element from a set")
    try:
        results = demonstrate_last_element_removal(
            table_name=table_name, key={"UserId": "tempUser"},
 set_name="SingleElementSet"
```

```
        )

        print(f"Set exists before removal: {results['exists_before']}")
        print(f"Set exists after removal: {results['exists_after']}")

        if not results["exists_after"]:
            print("The set attribute was completely removed when the last element
was deleted.")
        else:
            print("The set attribute still exists after removing the last
element.")
    except Exception as e:
        print(f"Error in last element removal demonstration: {e}")

    print("\nExample 5: Working with a number set")
    try:
        results = work_with_number_set(
            table_name=table_name,
            key={"UserId": "user456"},
            set_name="LuckyNumbers",
            initial_values=[7, 13, 42],
            values_to_add=[99, 100],
            values_to_remove=[13],
        )

        print(f"Initial number set: {results['create_response'].get('Attributes',
{})}")
        print(f"After adding numbers: {results['add_response'].get('Attributes',
{})}")
        print(f"After removing numbers:
{results['remove_response'].get('Attributes', {})}")
        print(f"Final state: {results['final_state']}")
    except Exception as e:
        print(f"Error working with number set: {e}")

    print("\nKey Points About DynamoDB Sets:")
    print("1. Sets can only contain elements of the same type (string, number, or
binary)")
    print("2. Sets automatically eliminate duplicate values")
    print("3. The ADD operation creates a set if it doesn't exist")
    print("4. The DELETE operation removes specified elements from a set")
    print("5. When the last element is removed from a set, the entire attribute
is deleted")
    print("6. Empty sets are not allowed in DynamoDB")
```

```
    print("7. Sets are unordered collections")
    print("8. The ADD operation is atomic for sets")
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Consulte uma tabela do DynamoDB usando lotes de instruções partiQL e um AWS SDK

Os exemplos de código a seguir mostram como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Before you run this example, download 'movies.json' from
// https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
GettingStarted.Js.02.html,
// and put it in the same folder as the example.

// Separator for the console display.
var SepBar = new string('-', 80);
const string tableName = "movie_table";
```

```
const string movieFileName = @"..\..\..\..\..\..\..\..\resources\sample_files
\movies.json";

DisplayInstructions();

// Create the table and wait for it to be active.
Console.WriteLine($"Creating the movie table: {tableName}");

var success = await DynamoDBMethods.CreateMovieTableAsync(tableName);
if (success)
{
    Console.WriteLine($"Successfully created table: {tableName}.");
}

WaitForEnter();

// Add movie information to the table from moviedata.json. See the
// instructions at the top of this file to download the JSON file.
Console.WriteLine($"Inserting movies into the new table. Please wait...");
success = await PartiQLBatchMethods.InsertMovies(tableName, movieFileName);
if (success)
{
    Console.WriteLine("Movies successfully added to the table.");
}
else
{
    Console.WriteLine("Movies could not be added to the table.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var title1 = "Star Wars";
var year1 = 1977;
var title2 = "Wizard of Oz";
var year2 = 1939;

Console.WriteLine($"Updating two movies with producer information: {title1} and
 {title2}.");
success = await PartiQLBatchMethods.GetBatch(tableName, title1, title2, year1,
 year2);
if (success)
{
    Console.WriteLine($"Successfully retrieved {title1} and {title2}.");
```

```
}
else
{
    Console.WriteLine("Select statement failed.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var producer1 = "LucasFilm";
var producer2 = "MGM";

Console.WriteLine($"Updating two movies with producer information: {title1} and
 {title2}.");
success = await PartiQLBatchMethods.UpdateBatch(tableName, producer1, title1,
 year1, producer2, title2, year2);
if (success)
{
    Console.WriteLine($"Successfully updated {title1} and {title2}.");
}
else
{
    Console.WriteLine("Update failed.");
}

WaitForEnter();

// Delete multiple movies by using the BatchExecute statement.
Console.WriteLine($"Now we will delete {title1} and {title2} from the table.");
success = await PartiQLBatchMethods.DeleteBatch(tableName, title1, year1, title2,
 year2);

if (success)
{
    Console.WriteLine($"Deleted {title1} and {title2}");
}
else
{
    Console.WriteLine($"could not delete {title1} or {title2}");
}

WaitForEnter();

// DNow that the PartiQL Batch scenario is complete, delete the movie table.
```

```
success = await DynamoDBMethods.DeleteTableAsync(tableName);

if (success)
{
    Console.WriteLine($"Successfully deleted {tableName}");
}
else
{
    Console.WriteLine($"Could not delete {tableName}");
}

/// <summary>
/// Displays the description of the application on the console.
/// </summary>
void DisplayInstructions()
{
    Console.Clear();
    Console.WriteLine();
    Console.Write(new string(' ', 24));
    Console.WriteLine("DynamoDB PartiQL Basics Example");
    Console.WriteLine(SepBar);
    Console.WriteLine("This demo application shows the basics of using Amazon
 DynamoDB with the AWS SDK for");
    Console.WriteLine(".NET version 3.7 and .NET 6.");
    Console.WriteLine(SepBar);
    Console.WriteLine("Creates a table by using the CreateTable method.");
    Console.WriteLine("Gets multiple movies by using a PartiQL SELECT
 statement.");
    Console.WriteLine("Updates multiple movies by using the ExecuteBatch
 method.");
    Console.WriteLine("Deletes multiple movies by using a PartiQL DELETE
 statement.");
    Console.WriteLine("Cleans up the resources created for the demo by deleting
 the table.");
    Console.WriteLine(SepBar);

    WaitForEnter();
}

/// <summary>
/// Simple method to wait for the <Enter> key to be pressed.
/// </summary>
void WaitForEnter()
{
```

```
    Console.WriteLine("\nPress <Enter> to continue.");
    Console.Write(SepBar);
    _ = Console.ReadLine();
}


        /// <summary>
        /// Gets movies from the movie table by
        /// using an Amazon DynamoDB PartiQL SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year1">The year of the first movie.</param>
        /// <param name="year2">The year of the second movie.</param>
        /// <returns>True if successful.</returns>
        public static async Task<bool> GetBatch(
            string tableName,
            string title1,
            string title2,
            int year1,
            int year2)
        {
            var getBatch = $"SELECT * FROM {tableName} WHERE title = ? AND year
 = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
```

```
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            if (response.Responses.Count > 0)
            {
                response.Responses.ForEach(r =>
                {
                    if (r.Item.Any())
                    {

Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
                    }
                });
                return true;
            }
            else
            {
                Console.WriteLine($"Couldn't find either {title1} or {title2}.");
                return false;
            }

        }

        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table into which the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
```

```
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE
{{'title': ?, 'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset +=
25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await
Client.BatchExecuteStatementAsync(new BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully
added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
```

```
                    // Clear the list of statements for the next batch.
                    statements.Clear();
                }
            }
            catch (AmazonDynamoDBException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        return success;
    }

    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the DynamoDB table.
    /// </summary>
    /// <param name="movieFileName">The full path to the JSON file.</param>
    /// <returns>A generic list of movie objects.</returns>
    public static List<Movie> ImportMovies(string movieFileName)
    {
        if (!File.Exists(movieFileName))
        {
            return null!;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();
        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        if (allMovies is not null)
        {
            // Return the first 250 entries.
            return allMovies.GetRange(0, 250);
        }
        else
        {
            return null!;
        }
    }

    /// <summary>
    /// Updates information for multiple movies.
```

```
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// movies to be updated.</param>
        /// <param name="producer1">The producer name for the first movie
        /// to update.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year that the first movie was released.</
param>
        /// <param name="producer2">The producer name for the second
        /// movie to update.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year that the second movie was released.</
param>
        /// <returns>A Boolean value that indicates the success of the update.</
returns>
        public static async Task<bool> UpdateBatch(
            string tableName,
            string producer1,
            string title1,
            int year1,
            string producer2,
            string title2,
            int year2)
        {

            string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title
 = ? AND year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer1 },
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
```

```
                    {
                        new AttributeValue { S = producer2 },
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
 BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }

        /// <summary>
        /// Deletes multiple movies using a PartiQL BatchExecuteAsync
        /// statement.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// moves that will be deleted.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year the first movie was released.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year the second movie was released.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeleteBatch(
            string tableName,
            string title1,
            int year1,
            string title2,
            int year2)
        {

            string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND
 year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
```

```
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        //  1. Create a table. (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::partiqlBatchExecuteScenario(clientConfig);

            // 7. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }

//! Scenario to modify and query a DynamoDB table using PartiQL batch statements.
/*!
  \sa partiqlBatchExecuteScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::partiqlBatchExecuteScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
                "On a scale of 1 - 10, how do you rate it? ",
                1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/
n) "));
    } while (doAgain == "y");
```

```
    std::cout << "Adding " << titles.size()
              << (titles.size() == 1 ? " movie " : " movies ")
              << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

            // Create attribute for the info map.
            Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute
= Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            ratingAttribute->SetN(ratings[i]);
            infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            plotAttribute->SetS(plots[i]);
            infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
            attributes.push_back(infoMapAttribute);
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
```

```
        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Retrieving the movie data with a batch \"SELECT\" statement."
              << std::endl;

    // 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
```

```
                request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    // 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

    for (size_t i = 0; i < titles.size(); ++i) {
        ratings[i] = askQuestionForFloatRange(
                Aws::String("\nLet's update your the movie, \"") + titles[i] +
                ".\nYou rated it  " + std::to_string(ratings[i])
                + ", what new rating would you give it? ", 1, 10);
    }

    std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";
```

```
        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Retrieving the updated movie data with a batch \"SELECT\"
statement."
              << std::endl;

    // 5. Get the updated data for multiple movies using "Select" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());
```

```
        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movies information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Deleting the movie data with a batch \"DELETE\" statement."
              << std::endl;
```

```
    // 6. Delete multiple movies using "Delete" statements.
 (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

 attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
 dynamoClient.BatchExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." <<
 std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
 dynamoClient.CreateTable(
                request);
```

```
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
```

```
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
```

```
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário que crie uma tabela e execute lotes de consultas do PartiQL.

```
import (
 "context"
 "fmt"
 "log"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunPartiQLBatchScenario shows you how to use the AWS SDK for Go
```

```
// to run batches of PartiQL statements to query a table that stores data about
 movies.
//
//   - Use batches of PartiQL statements to add, get, update, and delete data for
//     individual movies.
//
// This example creates an Amazon DynamoDB service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLBatchScenario(ctx context.Context, sdkConfig aws.Config, tableName
 string) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB PartiQL batch demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }
 runner := actions.PartiQLRunner{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }

 exists, err := tableBasics.TableExists(ctx)
 if err != nil {
  panic(err)
 }
 if !exists {
  log.Printf("Creating table %v...\n", tableName)
  _, err = tableBasics.CreateMovieTable(ctx)
  if err != nil {
   panic(err)
  } else {
   log.Printf("Created table %v.\n", tableName)
  }
```

```
} else {
 log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovies := []actions.Movie{{
 Title: "House PartiQL",
 Year:  currentYear - 5,
 Info: map[string]interface{}{
  "plot":   "Wacky high jinks result from querying a mysterious database.",
  "rating": 8.5}}, {
 Title: "House PartiQL 2",
 Year:  currentYear - 3,
 Info: map[string]interface{}{
  "plot":   "Moderate high jinks result from querying another mysterious
database.",
  "rating": 6.5}}, {
 Title: "House PartiQL 3",
 Year:  currentYear - 1,
 Info: map[string]interface{}{
  "plot":   "Tepid high jinks result from querying yet another mysterious
database.",
  "rating": 2.5},
},
}

log.Printf("Inserting a batch of movies into table '%v'.\n", tableName)
err = runner.AddMovieBatch(ctx, customMovies)
if err == nil {
 log.Printf("Added %v movies to the table.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting data for a batch of movies.")
movies, err := runner.GetMovieBatch(ctx, customMovies)
if err == nil {
 for _, movie := range movies {
  log.Println(movie)
 }
}
log.Println(strings.Repeat("-", 88))

newRatings := []float64{7.7, 4.4, 1.1}
```

```
 log.Println("Updating a batch of movies with new ratings.")
 err = runner.UpdateMovieBatch(ctx, customMovies, newRatings)
 if err == nil {
  log.Printf("Updated %v movies with new ratings.\n", len(customMovies))
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Getting projected data from the table to verify our update.")
 log.Println("Using a page size of 2 to demonstrate paging.")
 projections, err := runner.GetAllMovies(ctx, 2)
 if err == nil {
  log.Println("All movies:")
  for _, projection := range projections {
   log.Println(projection)
  }
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Deleting a batch of movies.")
 err = runner.DeleteMovieBatch(ctx, customMovies)
 if err == nil {
  log.Printf("Deleted %v movies.\n", len(customMovies))
 }

 err = tableBasics.DeleteTable(ctx)
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
```

```
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Crie uma estrutura e métodos que executem declarações PartiQL.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies
 to the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year, movie.Info})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(fmt.Sprintf(
    "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
 runner.TableName)),
   Parameters: params,
  }
 }
```

```
 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n",
 err)
 }
 return err
}



// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
 from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
 ([]Movie, error) {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 var outMovies []Movie
 if err != nil {
  log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
 } else {
  for _, response := range output.Responses {
   var movie Movie
```

```
    err = attributevalue.UnmarshalMap(response.Item, &movie)
    if err != nil {
     log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
    } else {
     outMovies = append(outMovies, movie)
    }
   }
  }
  return outMovies, err
}



// GetAllMovies runs a PartiQL SELECT statement to get all movies from the
 DynamoDB table.
// pageSize is not typically required and is used to show how to paginate the
 results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
 ([]map[string]interface{}, error) {
 var output []map[string]interface{}
 var response *dynamodb.ExecuteStatementOutput
 var err error
 var nextToken *string
 for moreData := true; moreData; {
  response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
   Statement: aws.String(
    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
   Limit:     aws.Int32(pageSize),
   NextToken: nextToken,
  })
  if err != nil {
   log.Printf("Couldn't get movies. Here's why: %v\n", err)
   moreData = false
  } else {
   var pageOutput []map[string]interface{}
   err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    log.Printf("Got a page of length %v.\n", len(response.Items))
    output = append(output, pageOutput...)
   }
```

```
    nextToken = response.NextToken
    moreData = nextToken != nil
   }
  }
  return output, err
}



// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the
 rating of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
 ratings []float64) error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{ratings[index],
 movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
 }
 return err
}



// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
 movies
// from the DynamoDB table.
```

```
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("Creating an Amazon DynamoDB table named " + tableName
            + " with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Adding multiple records into the " + tableName
            + " table using a batch command.");
        putRecordBatch(ddb);

        // Update multiple movies by using the BatchExecute statement.
        String title1 = "Star Wars";
        int year1 = 1977;
        String title2 = "Wizard of Oz";
        int year2 = 1939;

        System.out.println("Query two movies.");
        getBatch(ddb, tableName, title1, title2, year1, year2);

        System.out.println("Updating multiple records using a batch command.");
        updateTableItemBatch(ddb);

        System.out.println("Deleting multiple records using a batch command.");
        deleteItemBatch(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static boolean getBatch(DynamoDbClient ddb, String tableName, String
 title1, String title2, int year1, int year2) {
        String getBatch = "SELECT * FROM " + tableName + " WHERE title = ? AND
 year = ?";

        List<BatchStatementRequest> statements = new ArrayList<>();
        statements.add(BatchStatementRequest.builder()
```

```
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title1).build(),
                AttributeValue.builder().n(String.valueOf(year1)).build())
            .build());
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title2).build(),
                AttributeValue.builder().n(String.valueOf(year2)).build())
            .build());

        BatchExecuteStatementRequest batchExecuteStatementRequest =
BatchExecuteStatementRequest.builder()
            .statements(statements)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchExecuteStatementRequest);
            if (!response.responses().isEmpty()) {
                response.responses().forEach(r -> {
                    System.out.println(r.item().get("title") + "\\t" +
r.item().get("year"));
                });
                return true;
            } else {
                System.out.println("Couldn't find either " + title1 + " or " +
title2 + ".");
                return false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            return false;
        }
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());
```

```
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse = dbWaiter
                .waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

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

```
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n("1977")
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("Star Wars")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
 BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parameters)
                .build();

            // Set data for Movie 2.
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                .n("1939")
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("Wizard of Oz")
                .build();
```

```
            AttributeValue attMovie2B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie2.add(attMovie2);
            parametersMovie2.add(attMovie2A);
            parametersMovie2.add(attMovie2B);

            BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie2)
                .build();

            // Set data for Movie 3.
            List<AttributeValue> parametersMovie3 = new ArrayList<>();
            AttributeValue attMovie3 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie3A = AttributeValue.builder()
                .s("My Movie 3")
                .build();

            AttributeValue attMovie3B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie3.add(attMovie3);
            parametersMovie3.add(attMovie3A);
            parametersMovie3.add(attMovie3B);

            BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie3)
                .build();

            // Add all three movies to the list.
            List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
            myBatchStatementList.add(statementRequestMovie1);
            myBatchStatementList.add(statementRequestMovie2);
            myBatchStatementList.add(statementRequestMovie3);
```

```
            BatchExecuteStatementRequest batchRequest =
 BatchExecuteStatementRequest.builder()
                .statements(myBatchStatementList)
                .build();

            BatchExecuteStatementResponse response =
 ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
 response.toString());
            System.out.println("Added new movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "UPDATE MoviesPartiQBatch SET info = 'directors\":
[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Update three records.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
 BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Update record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
```

```
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Update record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);
        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
```

```
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Specify record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
```

```
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Specify record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);

        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch command.");
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
 {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse
 executeStatementRequest(DynamoDbClient ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }
}
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute instruções PartiQL em lote.

```
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DescribeTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);
const tableName = "Cities";

export const main = async (confirmAll = false) => {
  /**
   * Delete table if it exists.
   */
  try {
    await client.send(new DescribeTableCommand({ TableName: tableName }));
    // If no error was thrown, the table exists.
    const input = new ScenarioInput(
      "deleteTable",
      `A table named ${tableName} already exists. If you choose not to delete
this table, the scenario cannot continue. Delete it?`,
```

```
      { type: "confirm", confirmAll },
    );
    const deleteTable = await input.handle({}, { confirmAll });
    if (deleteTable) {
      await client.send(new DeleteTableCommand({ tableName }));
    } else {
      console.warn(
        "Scenario could not run. Either delete ${tableName} or provide a unique
 table name.",
      );
      return;
    }
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceNotFoundException"
    ) {
      // Do nothing. This means the table is not there.
    } else {
      throw caught;
    }
  }

  /**
   * Create a table.
   */

  log("Creating a table.");
  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "name",
        // 'S' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "S",
      },
```

```
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [{ AttributeName: "name", KeyType: "HASH" }],
  });
  await client.send(createTableCommand);
  log(`Table created: ${tableName}.`);

  /**
   * Wait until the table is active.
   */

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Insert items.
   */

  log("Inserting cities into the table.");
  const addItemsStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.insert.html
    Statements: [
      {
        Statement: `INSERT INTO ${tableName} value {'name':?, 'population':?}`,
        Parameters: ["Alachua", 10712],
      },
      {
        Statement: `INSERT INTO ${tableName} value {'name':?, 'population':?}`,
        Parameters: ["High Springs", 6415],
      },
    ],
  });
  await docClient.send(addItemsStatementCommand);
  log("Cities inserted.");

  /**
```

```
   * Select items.
   */

  log("Selecting cities from the table.");
  const selectItemsStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.select.html
    Statements: [
      {
        Statement: `SELECT * FROM ${tableName} WHERE name=?`,
        Parameters: ["Alachua"],
      },
      {
        Statement: `SELECT * FROM ${tableName} WHERE name=?`,
        Parameters: ["High Springs"],
      },
    ],
  });
  const selectItemResponse = await docClient.send(selectItemsStatementCommand);
  log(
    `Got cities: ${selectItemResponse.Responses.map(
      (r) => `${r.Item.name} (${r.Item.population})`,
    ).join(", ")}`,
  );

  /**
   * Update items.
   */

  log("Modifying the populations.");
  const updateItemStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.update.html
    Statements: [
      {
        Statement: `UPDATE ${tableName} SET population=? WHERE name=?`,
        Parameters: [10, "Alachua"],
      },
      {
        Statement: `UPDATE ${tableName} SET population=? WHERE name=?`,
        Parameters: [5, "High Springs"],
      },
    ],
  });
```

```
  await docClient.send(updateItemStatementCommand);
  log("Updated cities.");

  /**
   * Delete the items.
   */

  log("Deleting the cities.");
  const deleteItemStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.delete.html
    Statements: [
      {
        Statement: `DELETE FROM ${tableName} WHERE name=?`,
        Parameters: ["Alachua"],
      },
      {
        Statement: `DELETE FROM ${tableName} WHERE name=?`,
        Parameters: ["High Springs"],
      },
    ],
  });
  await docClient.send(deleteItemStatementCommand);
  log("Cities deleted.");

  /**
   * Delete the table.
   */

  log("Deleting the table.");
  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named
 id and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
```

```
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
 'title' : ?, 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
```

```
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
```

```
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
```

```
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    ddb.batchExecuteStatement(batchRequest)
```

```
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- Para obter detalhes da API, consulte a BatchExecuteStatementreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithPartiQLBatch
{
    public function run()
```

```
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo
using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
$tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
```

```
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQLBatch($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);
        echo "Movie added and updated.\n";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']}
        was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
```

```
        $service->updateItemByPartiQLBatch($statement, $parameters);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']
['S']}
        as a {$movie['Responses'][0]['Item']['rating']['N']}\n";

        $service->deleteItemByPartiQLBatch($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something
 lighter, in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were
 born:\n";
        $oops = "Oops! There were no movies released in that year (that we know
 of).\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
```

```
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQLBatch(string $statement, array
 $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
```

```
    }

    public function updateItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que possa executar lotes de instruções PartiQL.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because
 the
        resource transforms input and output from plain old Python objects
 (POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated
 with
                           each statement. This list must be in the same order as
 the
                           statements.
        :return: The responses returned from running the statements, if any.
        """
```

```
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[
                    {"Statement": statement, "Parameters": params}
                    for statement, params in zip(statements, param_list)
                ]
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the
 table "
                    "does not exist."
                )
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why:
 %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

Execute um cenário que crie uma tabela e execute consultas do PartiQL em lotes.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB PartiQL batch statement demo.")
    print("-" * 88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print("-" * 88)

    movie_data = [
```

```
        {
            "title": f"House PartiQL",
            "year": datetime.now().year - 5,
            "info": {
                "plot": "Wacky high jinks result from querying a mysterious
database.",
                "rating": Decimal("8.5"),
            },
        },
        {
            "title": f"House PartiQL 2",
            "year": datetime.now().year - 3,
            "info": {
                "plot": "Moderate high jinks result from querying another
mysterious database.",
                "rating": Decimal("6.5"),
            },
        },
        {
            "title": f"House PartiQL 3",
            "year": datetime.now().year - 1,
            "info": {
                "plot": "Tepid high jinks result from querying yet another
mysterious database.",
                "rating": Decimal("2.5"),
            },
        },
    ]

    print(f"Inserting a batch of movies into table '{table_name}.")
    statements = [
        f'INSERT INTO "{table_name}" ' f"VALUE {{'title': ?, 'year': ?,
'info': ?}}"
    ] * len(movie_data)
    params = [list(movie.values()) for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Getting data for a batch of movies.")
    statements = [f'SELECT * FROM "{table_name}" WHERE title=? AND year=?'] *
len(
        movie_data
    )
```

```
    params = [[movie["title"], movie["year"]] for movie in movie_data]
    output = wrapper.run_partiql(statements, params)
    for item in output["Responses"]:
        print(f"\n{item['Item']['title']}, {item['Item']['year']}")
        pprint(item["Item"])
    print("-" * 88)

    ratings = [Decimal("7.7"), Decimal("5.5"), Decimal("1.3")]
    print(f"Updating a batch of movies with new ratings.")
    statements = [
        f'UPDATE "{table_name}" SET info.rating=? ' f"WHERE title=? AND year=?"
    ] * len(movie_data)
    params = [
        [rating, movie["title"], movie["year"]]
        for rating, movie in zip(ratings, movie_data)
    ]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Getting projected data from the table to verify our update.")
    output = wrapper.dyn_resource.meta.client.execute_statement(
        Statement=f'SELECT title, info.rating FROM "{table_name}"'
    )
    pprint(output["Items"])
    print("-" * 88)

    print(f"Deleting a batch of movies from the table.")
    statements = [f'DELETE FROM "{table_name}" WHERE title=? AND year=?'] * len(
        movie_data
    )
    params = [[movie["title"], movie["year"]] for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print("-" * 88)

    print("\nThanks for watching!")
    print("-" * 88)
```

```
if __name__ == "__main__":
    try:
        dyn_res = boto3.resource("dynamodb")
        scaffold = Scaffold(dyn_res)
        movies = PartiQLBatchWrapper(dyn_res)
        run_scenario(scaffold, movies, "doc-example-table-partiql-movies")
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- Para obter detalhes da API, consulte a BatchExecuteStatementReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário que crie uma tabela e execute consultas do PartiQL em lotes.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**4)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLBatch.new(table_name)

new_step(1, 'Create a new DynamoDB table if none already exists.')
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end

new_step(2, 'Populate DynamoDB table with movie data.')
download_file = 'moviedata.json'
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
```

```
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(3, 'Select a batch of items from the movies table.')
  puts "Let's select some popular movies for side-by-side comparison."
  response = sdk.batch_execute_select([['Mean Girls', 2004], ['Goodfellas',
 1977], ['The Prancing of the Lambs', 2005]])
  puts("Items selected: #{response['responses'].length}\n")
  print "\nDone!\n".green

  new_step(4, 'Delete a batch of items from the movies table.')
  sdk.batch_execute_write([['Mean Girls', 2004], ['Goodfellas', 1977], ['The
 Prancing of the Lambs', 2005]])
  print "\nDone!\n".green

  new_step(5, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- Para obter detalhes da API, consulte BatchExecuteStatementa Referência AWS SDK para Ruby da API.

Consulte uma tabela do DynamoDB usando partiQL e um AWS SDK

Os exemplos de código a seguir mostram como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace PartiQL_Basics_Scenario
{
    public class PartiQLMethods
    {
        private static readonly AmazonDynamoDBClient Client = new
 AmazonDynamoDBClient();


        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table where the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
 movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
```

```
                string insertBatch = $"INSERT INTO {tableName} VALUE
{{'title': ?, 'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset +=
25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await
Client.BatchExecuteStatementAsync(new BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully
added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
```

```
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
            {
                return null!;
            }
        }



        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
```

```
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }



        /// <summary>
        /// Retrieve multiple movies by year using a SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="year">The year the movies were released.</param>
        /// <returns></returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { N = year.ToString() },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
```

```
        }


        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
```

```
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
```

```
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Displays the list of movies returned from a database query.
        /// </summary>
        /// <param name="items">The list of movie information to display.</param>
        private static void DisplayMovies(List<Dictionary<string,
 AttributeValue>> items)
        {
            if (items.Count > 0)
            {
                Console.WriteLine($"Found {items.Count} movies.");
                items.ForEach(item =>
 Console.WriteLine($"{item["year"].N}\t{item["title"].S}"));
            }
            else
            {
                Console.WriteLine($"Didn't find a movie that matched the supplied
 criteria.");
            }
        }


    }
}



        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
 GetSingleMovie(string tableName, string movieTitle)
```

```
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }



        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });
```

```
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
```

```
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        //  1. Create a table. (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::partiqlExecuteScenario(clientConfig);
```

```
            // 7. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }

//! Scenario to modify and query a DynamoDB table using single PartiQL
 statements.
/*!
  \sa partiqlExecuteScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::partiqlExecuteScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate
 it? ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
```

```
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
              << std::endl;

    //  3. Get the data for the movie using a "Select" statement.
(ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
```

```
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            // Print the retrieved movie information.
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were
retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }

    //  4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";
```

```
        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update a movie: "
                      << outcome.GetError().GetMessage();
            return false;
        }
    }

    std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

    //  5. Get the updated data for the movie using a "Select" statement.
(ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
```

```
            return false;
        }
        else {
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were
retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }

    std::cout << "Deleting the movie" << std::endl;

    // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
```

```
    }

    std::cout << "Movie successfully deleted." << std::endl;
    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
```

```
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." <<
 std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
 dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
```

```
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário que crie uma tabela e execute consultas do PartiQL.

```
import (
```

```
 "context"
 "fmt"
 "log"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunPartiQLSingleScenario shows you how to use the AWS SDK for Go
// to use PartiQL to query a table that stores data about movies.
//
// * Use PartiQL statements to add, get, update, and delete data for individual
 movies.
//
// This example creates an Amazon DynamoDB service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLSingleScenario(ctx context.Context, sdkConfig aws.Config,
 tableName string) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB PartiQL single action demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }
 runner := actions.PartiQLRunner{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }

 exists, err := tableBasics.TableExists(ctx)
```

```
if err != nil {
 panic(err)
}
if !exists {
 log.Printf("Creating table %v...\n", tableName)
 _, err = tableBasics.CreateMovieTable(ctx)
 if err != nil {
  panic(err)
 } else {
  log.Printf("Created table %v.\n", tableName)
 }
} else {
 log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovie := actions.Movie{
 Title: "24 Hour PartiQL People",
 Year:  currentYear,
 Info: map[string]interface{}{
  "plot":   "A group of data developers discover a new query language they can't
stop using.",
  "rating": 9.9,
 },
}

log.Printf("Inserting movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
err = runner.AddMovie(ctx, customMovie)
if err == nil {
 log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting data for movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
movie, err := runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
if err == nil {
 log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

newRating := 6.6
```

```
 log.Printf("Updating movie '%v' with a rating of %v.", customMovie.Title,
 newRating)
 err = runner.UpdateMovie(ctx, customMovie, newRating)
 if err == nil {
  log.Printf("Updated %v with a new rating.\n", customMovie.Title)
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Getting data again to verify the update.")
 movie, err = runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
 if err == nil {
  log.Println(movie)
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Deleting movie '%v'.\n", customMovie.Title)
 err = runner.DeleteMovie(ctx, customMovie)
 if err == nil {
  log.Printf("Deleted %v.\n", customMovie.Title)
 }

 err = tableBasics.DeleteTable(ctx)
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
```

```
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Crie uma estrutura e métodos que executem declarações PartiQL.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
 }
 return err
}
```

```
// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB
 table by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 var movie Movie
 params, err := attributevalue.MarshalList([]interface{}{title, year})
 if err != nil {
  panic(err)
 }
 response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Items[0], &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}



// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie
 that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
 float64) error {
 params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
```

```
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
 }
 return err
}



// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the
 DynamoDB table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        String fileName = "../../../resources/sample_files/movies.json";
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(
            "******* Creating an Amazon DynamoDB table named MoviesPartiQ with a
 key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("Getting data from the MoviesPartiQ table.");
        getItem(ddb);

        System.out.println("Putting a record into the MoviesPartiQ table.");
        putRecord(ddb);

        System.out.println("Updating a record.");
        updateTableItem(ddb);

        System.out.println("Querying the movies released in 2013.");
        queryTable(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
```

```
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //Scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();
```

```
            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        List<AttributeValue> parameters = new ArrayList<>();
        while (iter.hasNext()) {

            // Add 200 movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf(year))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s(title)
```

```
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s(info)
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            // Insert the movie into the Amazon DynamoDB table.
            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added Movie " + title);

            parameters.remove(att1);
            parameters.remove(att2);
            parameters.remove(att3);
            t++;
        }
    }

    public static void getItem(DynamoDbClient ddb) {

        String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and
title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n("2012")
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The Perks of Being a Wallflower")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
 'title' : ?, 'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2020"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added new movie.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItem(DynamoDbClient ddb) {

        String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":
[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
```

```
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The East")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
 {

        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse
 executeStatementRequest(DynamoDbClient ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }

    private static void processResults(ExecuteStatementResponse
 executeStatementResult) {
        System.out.println("ExecuteStatement successful: " +
 executeStatementResult.toString());
    }
}
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute instruções PartiQL individuais.

```
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DescribeTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);
const tableName = "SingleOriginCoffees";

export const main = async (confirmAll = false) => {
  /**
   * Delete table if it exists.
   */
  try {
    await client.send(new DescribeTableCommand({ TableName: tableName }));
    // If no error was thrown, the table exists.
    const input = new ScenarioInput(
      "deleteTable",
      `A table named ${tableName} already exists. If you choose not to delete
this table, the scenario cannot continue. Delete it?`,
```

```
      { type: "confirm", confirmAll },
    );
    const deleteTable = await input.handle({});
    if (deleteTable) {
      await client.send(new DeleteTableCommand({ tableName }));
    } else {
      console.warn(
        "Scenario could not run. Either delete ${tableName} or provide a unique
 table name.",
      );
      return;
    }
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceNotFoundException"
    ) {
      // Do nothing. This means the table is not there.
    } else {
      throw caught;
    }
  }

  /**
   * Create a table.
   */

  log("Creating a table.");
  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "varietal",
        // 'S' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "S",
      },
```

```
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [{ AttributeName: "varietal", KeyType: "HASH" }],
  });
  await client.send(createTableCommand);
  log(`Table created: ${tableName}.`);

  /**
   * Wait until the table is active.
   */

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Insert an item.
   */

  log("Inserting a coffee into the table.");
  const addItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.insert.html
    Statement: `INSERT INTO ${tableName} value {'varietal':?, 'profile':?}`,
    Parameters: ["arabica", ["chocolate", "floral"]],
  });
  await client.send(addItemStatementCommand);
  log("Coffee inserted.");

  /**
   * Select an item.
   */

  log("Selecting the coffee from the table.");
  const selectItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.select.html
    Statement: `SELECT * FROM ${tableName} WHERE varietal=?`,
```

```
    Parameters: ["arabica"],
  });
  const selectItemResponse = await docClient.send(selectItemStatementCommand);
  log(`Got coffee: ${JSON.stringify(selectItemResponse.Items[0])}`);

  /**
   * Update the item.
   */

  log("Add a flavor profile to the coffee.");
  const updateItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.update.html
    Statement: `UPDATE ${tableName} SET profile=list_append(profile, ?) WHERE
 varietal=?`,
    Parameters: [["fruity"], "arabica"],
  });
  await client.send(updateItemStatementCommand);
  log("Updated coffee");

  /**
   * Delete the item.
   */

  log("Deleting the coffee.");
  const deleteItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.delete.html
    Statement: `DELETE FROM ${tableName} WHERE varietal=?`,
    Parameters: ["arabica"],
  });
  await docClient.send(deleteItemStatementCommand);
  log("Coffee deleted.");

  /**
   * Delete the table.
   */

  log("Deleting the table.");
  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQ"
    val fileName = "../../../resources/sample_files/movies.json"
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key
 named id and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("******* Getting data from the MoviesPartiQ table.")
    getMoviePartiQL(ddb)

    println("******* Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("******* Updating a record.")
    updateTableItemPartiQL(ddb)

    println("******* Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("******* Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}

suspend fun createTablePartiQL(
```

```
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}
```

```
suspend fun loadDataPartiQL(
    ddb: DynamoDbClient,
    fileName: String,
) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {
        if (t == 200) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}
```

```
suspend fun putRecordPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
 Cooper\",\"Ernest B. Schoedsack\' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
```

```
    statementVal: String,
    parametersVal: List<AttributeValue>,
): ExecuteStatementResponse {
    val request =
        ExecuteStatementRequest {
            statement = statementVal
            parameters = parametersVal
        }

    return ddb.executeStatement(request)
}
```

- Para obter detalhes da API, consulte a ExecuteStatementreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\testable_readline;
use function AwsUtilities\loadMovieData;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
```

```
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo
 using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQL($statement, $parameters);
```

```
        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
        echo "Movie added and updated.\n";



        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
 {$movie['Items'][0]['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Items'][0]['title']
['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
```

```
        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
{$movie['Items'][0]['rating']['N']}\n";

        $service->deleteItemByPartiQL($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something
lighter, in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were
born:\n";
        $oops = "Oops! There were no movies released in that year (that we know
of).\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
```

```
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
```

```
            'Parameters' => $parameters,
        ]);
    }
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que possa executar instruções PartiQL.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
```

```
    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because
the
        resource transforms input and output from plain old Python objects
(POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                       statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not
exist.",
                    statement,
                )
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

Execute um cenário que crie uma tabela e execute consultas do PartiQL.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB PartiQL single statement demo.")
    print("-" * 88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print("-" * 88)

    title = "24 Hour PartiQL People"
    year = datetime.now().year
    plot = "A group of data developers discover a new query language they can't
 stop using."
    rating = Decimal("9.9")

    print(f"Inserting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f"INSERT INTO \"{table_name}\" VALUE {{'title': ?, 'year': ?,
 'info': ?}}",
        [title, year, {"plot": plot, "rating": rating}],
    )
    print("Success!")
    print("-" * 88)

    print(f"Getting data for movie '{title}' released in {year}.")
    output = wrapper.run_partiql(
        f'SELECT * FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    for item in output["Items"]:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output["Items"])
    print("-" * 88)

    rating = Decimal("2.4")
    print(f"Updating movie '{title}' with a rating of {float(rating)}.")
    wrapper.run_partiql(
        f'UPDATE "{table_name}" SET info.rating=? WHERE title=? AND year=?',
        [rating, title, year],
    )
    print("Success!")
    print("-" * 88)
```

```
    print(f"Getting data again to verify our update.")
    output = wrapper.run_partiql(
        f'SELECT * FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    for item in output["Items"]:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output["Items"])
    print("-" * 88)

    print(f"Deleting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f'DELETE FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    print("Success!")
    print("-" * 88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print("-" * 88)

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        dyn_res = boto3.resource("dynamodb")
        scaffold = Scaffold(dyn_res)
        movies = PartiQLWrapper(dyn_res)
        run_scenario(scaffold, movies, "doc-example-table-partiql-movies")
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- Para obter detalhes da API, consulte a ExecuteStatementReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário que crie uma tabela e execute consultas do PartiQL.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**8)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLSingle.new(table_name)

new_step(1, 'Create a new DynamoDB table if none already exists.')
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end

new_step(2, 'Populate DynamoDB table with movie data.')
download_file = 'moviedata.json'
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
scaffold.write_batch(movie_data)
puts("Records added: #{movie_data.length}.")
print "Done!\n".green

new_step(3, 'Select a single item from the movies table.')
response = sdk.select_item_by_title('Star Wars')
puts("Items selected for title 'Star Wars': #{response.items.length}\n")
print response.items.first.to_s.yellow
print "\n\nDone!\n".green

new_step(4, 'Update a single item from the movies table.')
puts "Let's correct the rating on The Big Lebowski to 10.0."
sdk.update_rating_by_title('The Big Lebowski', 1998, 10.0)
print "\nDone!\n".green
```

```
  new_step(5, 'Delete a single item from the movies table.')
  puts "Let's delete The Silence of the Lambs because it's just too scary."
  sdk.delete_item_by_title('The Silence of the Lambs', 1991)
  print "\nDone!\n".green

  new_step(6, 'Insert a new item into the movies table.')
  puts "Let's create a less-scary movie called The Prancing of the Lambs."
  sdk.insert_item('The Prancing of the Lambs', 2005, 'A movie about happy
 livestock.', 5.0)
  print "\nDone!\n".green

  new_step(7, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_table(
    client: &Client,
    table: &str,
    key: &str,
) -> Result<(), SdkError<CreateTableError>> {
    let ad = AttributeDefinition::builder()
        .attribute_name(key)
        .attribute_type(ScalarAttributeType::S)
        .build()
        .expect("creating AttributeDefinition");
```

```
    let ks = KeySchemaElement::builder()
        .attribute_name(key)
        .key_type(KeyType::Hash)
        .build()
        .expect("creating KeySchemaElement");

    match client
        .create_table()
        .table_name(table)
        .key_schema(ks)
        .attribute_definitions(ad)
        .billing_mode(BillingMode::PayPerRequest)
        .send()
        .await
    {
        Ok(_) => Ok(()),
        Err(e) => Err(e),
    }
}

async fn add_item(client: &Client, item: Item) -> Result<(),
 SdkError<ExecuteStatementError>> {
    match client
        .execute_statement()
        .statement(format!(
            r#"INSERT INTO "{}" VALUE {{
                "{}": ?,
                "acount_type": ?,
                "age": ?,
                "first_name": ?,
                "last_name": ?
        }} "#,
            item.table, item.key
        ))
        .set_parameters(Some(vec![
            AttributeValue::S(item.utype),
            AttributeValue::S(item.age),
            AttributeValue::S(item.first_name),
            AttributeValue::S(item.last_name),
        ]))
        .send()
        .await
    {
```

```
        Ok(_) => Ok(()),
        Err(e) => Err(e),
    }
}

async fn query_item(client: &Client, item: Item) -> bool {
    match client
        .execute_statement()
        .statement(format!(
            r#"SELECT * FROM "{}" WHERE "{}" = ?"#,
            item.table, item.key
        ))
        .set_parameters(Some(vec![AttributeValue::S(item.value)]))
        .send()
        .await
    {
        Ok(resp) => {
            if !resp.items().is_empty() {
                println!("Found a matching entry in the table:");
                println!("{:?}", resp.items.unwrap_or_default().pop());
                true
            } else {
                println!("Did not find a match.");
                false
            }
        }
        Err(e) => {
            println!("Got an error querying table:");
            println!("{}", e);
            process::exit(1);
        }
    }
}

async fn remove_item(client: &Client, table: &str, key: &str, value: String) ->
 Result<(), Error> {
    client
        .execute_statement()
        .statement(format!(r#"DELETE FROM "{table}" WHERE "{key}" = ?"#))
        .set_parameters(Some(vec![AttributeValue::S(value)]))
        .send()
        .await?;

    println!("Deleted item.");
```

```
    Ok(())
}

async fn remove_table(client: &Client, table: &str) -> Result<(), Error> {
    client.delete_table().table_name(table).send().await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a ExecuteStatementreferência da API AWS SDK for Rust.

Consulte uma tabela do DynamoDB usando um índice secundário global com um AWS SDK

O código de exemplo a seguir mostra como consultar uma tabela usando um índice secundário global.

- Consulte uma tabela do DynamoDB usando a respectiva chave primária.
- Consulte um índice secundário global (GSI) para obter padrões de acesso alternativos.
- Compare consultas de tabela e consultas de GSI.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB usando sua chave primária e um Índice Secundário Global (GSI) com. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
```

```
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
 response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
 table", e);
        }
    }
```

```
    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
```

```
            System.out.println("Query on GSI successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI",
e);
        }
    }
```

Compare a consulta direta de uma tabela com a consulta de um GSI com. AWS SDK for Java 2.x

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <basePartitionKeyName> <basePartitionKeyValue>
<gsiName> <gsiPartitionKeyName> <gsiPartitionKeyValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    basePartitionKeyName - The name of the base table partition
key attribute.
                    basePartitionKeyValue - The value of the base table partition
key to query.
                    gsiName - The name of the Global Secondary Index.
                    gsiPartitionKeyName - The name of the GSI partition key
attribute.
                    gsiPartitionKeyValue - The value of the GSI partition key to
query.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        final String tableName = args[0];
        final String basePartitionKeyName = args[1];
        final String basePartitionKeyValue = args[2];
        final String gsiName = args[3];
        final String gsiPartitionKeyName = args[4];
        final String gsiPartitionKeyValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        try (DynamoDbClient ddb =
DynamoDbClient.builder().region(region).build()) {
            final QueryTableAndGSI queryHelper = new QueryTableAndGSI(ddb);

            // Query the base table
            System.out.println("Querying base table where " +
basePartitionKeyName + " = " + basePartitionKeyValue);
            final QueryResponse tableResponse =
                queryHelper.queryTable(tableName, basePartitionKeyName,
basePartitionKeyValue);

            System.out.println("Found " + tableResponse.count() + " items in base
table:");
            tableResponse.items().forEach(item -> System.out.println(item));

            // Query the GSI
            System.out.println(
                "\nQuerying GSI '" + gsiName + "' where " + gsiPartitionKeyName +
" = " + gsiPartitionKeyValue);
            final QueryResponse gsiResponse =
                queryHelper.queryGlobalSecondaryIndex(tableName, gsiName,
gsiPartitionKeyName, gsiPartitionKeyValue);

            System.out.println("Found " + gsiResponse.count() + " items in
GSI:");
            gsiResponse.items().forEach(item -> System.out.println(item));

            // Explain the differences between querying a table and a GSI
            System.out.println("\nKey differences between querying a table and a
GSI:");
            System.out.println("1. When querying a GSI, you must specify the
indexName parameter");
```

```
            System.out.println("2. GSIs may not contain all attributes from the
 base table (projection)");
            System.out.println("3. GSIs consume read capacity units from the
 GSI's capacity, not the base table's");
            System.out.println("4. GSIs may have eventually consistent data
 (cannot use ConsistentRead=true)");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table or index not found: " + e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB usando a chave primária com. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table using the primary key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} userId - The user ID to query by (partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryTable(
  config,
```

```
  tableName,
  userId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the base table
    const input = {
      TableName: tableName,
      KeyConditionExpression: "user_id = :userId",
      ExpressionAttributeValues: {
        ":userId": { S: userId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying table: ${error}`);
    throw error;
  }
}
```

Consulte um Índice Secundário Global (GSI) do DynamoDB com. AWS SDK para JavaScript

```
/**
 * Queries a DynamoDB Global Secondary Index (GSI)
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} indexName - The name of the GSI to query
 * @param {string} gameId - The game ID to query by (GSI partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryGSI(
  config,
  tableName,
  indexName,
  gameId
) {
```

```
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the GSI
    const input = {
      TableName: tableName,
      IndexName: indexName,
      KeyConditionExpression: "game_id = :gameId",
      ExpressionAttributeValues: {
        ":gameId": { S: gameId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying GSI: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB usando sua chave primária e um Índice Secundário Global (GSI) com. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_table(table_name, partition_key_name, partition_key_value):
    """
    Query a DynamoDB table using its primary key.

    Args:
        table_name (str): The name of the DynamoDB table.
```

```
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the table's primary key
    response =
 table.query(KeyConditionExpression=Key(partition_key_name).eq(partition_key_value))

    return response


def query_gsi(table_name, index_name, partition_key_name, partition_key_value):
    """
    Query a Global Secondary Index (GSI) on a DynamoDB table.

    Args:
        table_name (str): The name of the DynamoDB table.
        index_name (str): The name of the Global Secondary Index.
        partition_key_name (str): The name of the GSI's partition key attribute.
        partition_key_value (str): The value of the GSI's partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the GSI
    response = table.query(
        IndexName=index_name,
 KeyConditionExpression=Key(partition_key_name).eq(partition_key_value)
    )

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela do DynamoDB usando uma condição begins_with com um AWS SDK

Os exemplos de código a seguir mostram como consultar uma tabela usando uma condição begins_with.

- Use a função begins_with em uma expressão de condição de chave.
- Filtre itens com base em um padrão de prefixo na chave de classificação.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB usando uma condição begins_with na chave de classificação por meio do AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithBeginsWithCondition(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String sortKeyName,
        final String sortKeyPrefix) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
```

```
        CodeSampleUtils.validateStringParameter("Sort key name", sortKeyName);
        CodeSampleUtils.validateStringParameter("Sort key prefix",
sortKeyPrefix);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, sortKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_SK_PREFIX,
            AttributeValue.builder().s(sortKeyPrefix).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with begins_with condition successful.
Found {0} items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with begins_with condition",
e);
            throw e;
        }
    }
```

Demonstre o uso de begins_with com diferentes comprimentos de prefixo por meio do AWS SDK for Java 2.x.

```
    public static void main(String[] args) {
        try {
            CodeSampleUtils.BeginsWithQueryConfig config =
CodeSampleUtils.BeginsWithQueryConfig.fromArgs(args);
            LOGGER.log(Level.INFO, "Querying items where {0} = {1} and {2} begins
with ''{3}''", new Object[] {
                config.getPartitionKeyName(),
                config.getPartitionKeyValue(),
                config.getSortKeyName(),
                config.getSortKeyPrefix()
            });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new BeginsWithQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withSortKeyName(config.getSortKeyName())
                .withSortKeyPrefix(config.getSortKeyPrefix())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> LOGGER.info(item.toString()));

            // Demonstrate with a different prefix
            if (!config.getSortKeyPrefix().isEmpty()) {
                String shorterPrefix = config.getSortKeyPrefix()
                    .substring(0, Math.max(1,
config.getSortKeyPrefix().length() / 2));
                LOGGER.log(Level.INFO, "\nNow querying with a shorter prefix:
''{0}''", shorterPrefix);

                final QueryResponse response2 = new BeginsWithQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withSortKeyName(config.getSortKeyName())
                    .withSortKeyPrefix(shorterPrefix)
```

```
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with shorter prefix:",
 response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }
        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            printUsage();
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found", e);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error", e);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error", e);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB usando uma condição begins_with na chave de classificação por meio do AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items where the sort key begins with a specific
 prefix
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key
 * @param {string} prefix - The prefix to match at the beginning of the sort key
 * @returns {Promise<Object>} - The query response
 */
```

```
async function queryWithBeginsWith(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  prefix
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND begins_with(#sk, :prefix)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":prefix": { S: prefix }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with begins_with: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB usando uma condição begins_with na chave de classificação por meio do AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_begins_with(
    table_name, partition_key_name, partition_key_value, sort_key_name, prefix
):
    """
    Query a DynamoDB table with a begins_with condition on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute.
        prefix (str): The prefix to match at the beginning of the sort key.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query with a begins_with condition on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) & Key(
        sort_key_name
    ).begins_with(prefix)
    response = table.query(KeyConditionExpression=key_condition)

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela do DynamoDB usando um intervalo de datas na chave de classificação com um AWS SDK

Os exemplos de código a seguir mostram como consultar uma tabela usando um intervalo de datas na chave de classificação.

- Consulte itens em um intervalo de datas específico.
- Use operadores de comparação em chaves de classificação formatadas por data.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB para itens dentro de um intervalo de datas com. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
```

```
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start
date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
```

```
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

Demonstra como consultar uma tabela do DynamoDB com filtragem de intervalo de datas.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<dateKeyName> <startDate> <endDate> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    dateKeyName - The name of the date attribute to filter on.
                    startDate - The start date for the range query (YYYY-MM-DD).
                    endDate - The end date for the range query (YYYY-MM-DD).
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        try {
            // Parse command line arguments into a config object
            CodeSampleUtils.DateRangeQueryConfig config =
CodeSampleUtils.DateRangeQueryConfig.fromArgs(args);

            LOGGER.log(
                Level.INFO, "Querying items from {0} to {1}", new Object[]
{config.getStartDate(), config.getEndDate()
                });
```

```
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DateRangeQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withDateKeyName(config.getDateKeyName())
                .withStartDate(config.getStartDate())
                .withEndDate(config.getEndDate())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> {
                LOGGER.info(item.toString());

                // Extract and display the date attribute for clarity
                if (item.containsKey(config.getDateKeyName())) {
                    LOGGER.log(
                        Level.INFO,
                        "  Date attribute: {0}",
                        item.get(config.getDateKeyName()).s());
                }
            });

            // Demonstrate with a different date range
            LocalDate narrowerStartDate = config.getStartDate().plusDays(1);
            LocalDate narrowerEndDate = config.getEndDate().minusDays(1);

            if (!narrowerStartDate.isAfter(narrowerEndDate)) {
                LOGGER.log(Level.INFO, "\nNow querying with a narrower date
range: {0} to {1}", new Object[] {
                    narrowerStartDate, narrowerEndDate
                });

                final QueryResponse response2 = new DateRangeQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withDateKeyName(config.getDateKeyName())
                    .withStartDate(narrowerStartDate)
                    .withEndDate(narrowerEndDate)
                    .withRegion(config.getRegion())
```

```
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with narrower date
range:", response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }

            LOGGER.info("\nNote: When storing dates in DynamoDB:");
            LOGGER.info("1. Use ISO format (YYYY-MM-DD) for lexicographical
ordering");
            LOGGER.info("2. Use the BETWEEN operator for inclusive date range
queries");
            LOGGER.info("3. Consider using ISO-8601 format for timestamps with
time components");

        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error: {0}", e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error: {0}", e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB para itens dentro de um intervalo de datas com. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
```

```
 * Queries a DynamoDB table for items within a specific date range on the sort
 key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key (must be a date/time
 attribute)
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRangeOnSortKey(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: '#pk = :pkValue AND #sk BETWEEN :startDate
 AND :endDate',
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
```

```
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range on sort key: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB para itens dentro de um intervalo de datas com. AWS SDK para Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_date_range(
    table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
):
    """
    Query a DynamoDB table with a date range on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        start_date (datetime): The start date for the query range.
        end_date (datetime): The end date for the query range.
```

```
    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date values as ISO 8601 strings
    # DynamoDB works well with ISO format for date values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key using BETWEEN operator
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response


def query_with_date_range_by_month(
    table_name, partition_key_name, partition_key_value, sort_key_name, year,
 month
):
    """
    Query a DynamoDB table for a specific month's data.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        year (int): The year to query.
```

```
        month (int): The month to query (1-12).

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Calculate the start and end dates for the specified month
    if month == 12:
        next_year = year + 1
        next_month = 1
    else:
        next_year = year
        next_month = month + 1

    start_date = datetime(year, month, 1)
    end_date = datetime(next_year, next_month, 1) - timedelta(microseconds=1)

    # Format the date values as ISO 8601 strings
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(KeyConditionExpression=key_condition)

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela do DynamoDB com uma expressão de filtro complexa com um AWS SDK

Os exemplos de código a seguir mostram como consultar uma tabela com uma expressão de filtro complexa.

- Aplique expressões de filtro complexas aos resultados da consulta.
- Combine várias condições usando operadores lógicos.
- Filtre itens com base em atributos não chave.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB com uma expressão de filtro complexa usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {
```

```
        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());
```

```
        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com uma expressão de filtro complexa usando. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with a complex filter expression
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number|string} minViews - Minimum number of views for filtering
 * @param {number|string} minReplies - Minimum number of replies for filtering
 * @param {string} requiredTag - Tag that must be present in the item's tags set
 * @returns {Promise<Object>} - The query response
 */
async function queryWithComplexFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  minViews,
```

```
  minReplies,
  requiredTag
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      FilterExpression: "views >= :minViews AND replies >= :minReplies AND
 contains(tags, :tag)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":minViews": { N: minViews.toString() },
        ":minReplies": { N: minReplies.toString() },
        ":tag": { S: requiredTag }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with complex filter: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com uma expressão de filtro complexa usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_complex_filter(
    table_name,
    partition_key_name,
    partition_key_value,
    min_rating=None,
    status_list=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        min_rating (float, optional): Minimum rating value for filtering.
        status_list (list, optional): List of status values to include.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize the filter expression and expression attribute values
    filter_expression = None
    expression_attribute_values = {}

    # Build the filter expression based on provided parameters
    if min_rating is not None:
        filter_expression = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if status_list and len(status_list) > 0:
```

```
        status_condition = None
        for i, status in enumerate(status_list):
            status_value_name = f":status{i}"
            expression_attribute_values[status_value_name] = status

            if status_condition is None:
                status_condition = Attr("status").eq(status)
            else:
                status_condition = status_condition | Attr("status").eq(status)

        if filter_expression is None:
            filter_expression = status_condition
        else:
            filter_expression = filter_expression & status_condition

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if filter_expression is None:
            filter_expression = price_condition
        else:
            filter_expression = filter_expression & price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] =
 expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response


def query_with_complex_filter_and_or(
    table_name,
    partition_key_name,
    partition_key_value,
    category=None,
    min_rating=None,
```

```
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression using AND and OR
 operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        category (str, optional): Category value for filtering.
        min_rating (float, optional): Minimum rating value for filtering.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build a complex filter expression with AND and OR operators
    filter_expression = None
    expression_attribute_values = {}

    # Build the category condition
    if category:
        filter_expression = Attr("category").eq(category)
        expression_attribute_values[":category"] = category

    # Build the rating and price condition (rating >= min_rating OR price <=
 max_price)
    rating_price_condition = None

    if min_rating is not None:
        rating_price_condition = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price
```

```
        if rating_price_condition is None:
            rating_price_condition = price_condition
        else:
            rating_price_condition = rating_price_condition | price_condition

    # Combine the conditions
    if rating_price_condition:
        if filter_expression is None:
            filter_expression = rating_price_condition
        else:
            filter_expression = filter_expression & rating_price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] =
expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela do DynamoDB com uma expressão de filtro dinâmico com um AWS SDK

Os exemplos de código a seguir mostram como consultar uma tabela com uma expressão de filtro dinâmica.

- Crie expressões de filtro dinamicamente no tempo de execução.
- Crie condições de filtro com base na entrada do usuário ou no estado da aplicação.
- Adicione ou remova critérios de filtro condicionalmente.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB com uma expressão de filtro construída dinamicamente usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
 QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName,
 partitionKeyName, partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
```

```
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " +
e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }
```

Demonstra como usar expressões de filtro dinâmico com AWS SDK for Java 2.x.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
```

```
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " +
filterAttrName + " = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items
with multiple filters:");
            multiFilterResponse.items().forEach(item ->
System.out.println(item));

        } catch (IllegalArgumentException e) {
```

```
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com uma expressão de filtro construída dinamicamente usando. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

async function queryWithDynamicFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  sortKeyValue,
  filterParams = {}
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize filter expression components
    let filterExpressions = [];
    const expressionAttributeValues = {
      ":pkValue": { S: partitionKeyValue },
```

```
      ":skValue": { S: sortKeyValue }
    };
    const expressionAttributeNames = {
      "#pk": partitionKeyName,
      "#sk": sortKeyName
    };

    // Add status filter if provided
    if (filterParams.status) {
      filterExpressions.push("status = :status");
      expressionAttributeValues[":status"] = { S: filterParams.status };
    }

    // Add minimum views filter if provided
    if (filterParams.minViews !== undefined) {
      filterExpressions.push("views >= :minViews");
      expressionAttributeValues[":minViews"] = { N:
 filterParams.minViews.toString() };
    }

    // Add author filter if provided
    if (filterParams.author) {
      filterExpressions.push("author = :author");
      expressionAttributeValues[":author"] = { S: filterParams.author };
    }

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND #sk = :skValue"
    };

    // Add filter expression if any filters were provided
    if (filterExpressions.length > 0) {
      input.FilterExpression = filterExpressions.join(" AND ");
    }

    // Add expression attribute names and values
    input.ExpressionAttributeNames = expressionAttributeNames;
    input.ExpressionAttributeValues = expressionAttributeValues;

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
```

```
  } catch (error) {
    console.error(`Error querying with dynamic filter: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com uma expressão de filtro construída dinamicamente usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_dynamic_filter(
    table_name, partition_key_name, partition_key_value, filter_conditions=None
):
    """
    Query a DynamoDB table with a dynamically constructed filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_conditions (dict, optional): A dictionary of filter conditions
 where
            keys are attribute names and values are dictionaries with 'operator'
 and 'value'.
            Example: {'rating': {'operator': '>=', 'value': 4}, 'status':
 {'operator': '=', 'value': 'active'}}

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize variables for the filter expression and attribute values
    filter_expression = None
    expression_attribute_values = {":pk_val": partition_key_value}

    # Dynamically build the filter expression if filter conditions are provided
    if filter_conditions:
        for attr_name, condition in filter_conditions.items():
            operator = condition.get("operator")
            value = condition.get("value")
            attr_value_name = f":{attr_name}"
            expression_attribute_values[attr_value_name] = value

            # Create the appropriate filter expression based on the operator
            current_condition = None
            if operator == "=":
                current_condition = Attr(attr_name).eq(value)
            elif operator == "!=":
                current_condition = Attr(attr_name).ne(value)
            elif operator == ">":
                current_condition = Attr(attr_name).gt(value)
            elif operator == ">=":
                current_condition = Attr(attr_name).gte(value)
            elif operator == "<":
                current_condition = Attr(attr_name).lt(value)
            elif operator == "<=":
                current_condition = Attr(attr_name).lte(value)
            elif operator == "contains":
                current_condition = Attr(attr_name).contains(value)
            elif operator == "begins_with":
                current_condition = Attr(attr_name).begins_with(value)

            # Combine with existing filter expression using AND
            if current_condition:
                if filter_expression is None:
                    filter_expression = current_condition
                else:
                    filter_expression = filter_expression & current_condition

    # Perform the query with the dynamically built filter expression
    query_params = {"KeyConditionExpression": key_condition}
```

```
    if filter_expression:
        query_params["FilterExpression"] = filter_expression

    response = table.query(**query_params)
    return response
```

Demonstra como usar expressões de filtro dinâmico com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the query_with_dynamic_filter function."""
    # Example parameters
    table_name = "Products"
    partition_key_name = "Category"
    partition_key_value = "Electronics"

    # Define dynamic filter conditions based on user input or runtime conditions
    user_min_rating = 4  # This could come from user input
    user_status_filter = "active"  # This could come from user input

    filter_conditions = {}

    # Only add conditions that are actually specified
    if user_min_rating is not None:
        filter_conditions["rating"] = {"operator": ">=", "value":
 user_min_rating}

    if user_status_filter:
        filter_conditions["status"] = {"operator": "=", "value":
 user_status_filter}

    print(
        f"Querying products in category '{partition_key_value}' with filter
 conditions: {filter_conditions}"
    )

    # Execute the query with dynamic filter
    response = query_with_dynamic_filter(
        table_name, partition_key_name, partition_key_value, filter_conditions
    )
```

```
    # Process the results
    items = response.get("Items", [])
    print(f"Found {len(items)} items")

    for item in items:
        print(f"Product: {item}")
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela do DynamoDB com uma expressão de filtro e limite com uma AWS SDK

Os exemplos de código a seguir mostram como consultar uma tabela com uma expressão de filtro e limite.

- Aplique expressões de filtro aos resultados da consulta com um limite nos itens avaliados.
- Entenda como o limite afeta os resultados da consulta filtrada.
- Controle o número máximo de itens processados em uma consulta.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB com uma expressão de filtro e limite usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
 filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
 filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
 filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
```

```
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
 {0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
 filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
 e.getMessage());
            throw e;
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com uma expressão de filtro e limite usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_filter_and_limit(
    table_name,
    partition_key_name,
    partition_key_value,
    filter_attribute=None,
    filter_value=None,
```

```
    limit=10,
):
    """
    Query a DynamoDB table with a filter expression and limit the number of
 results.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute (str, optional): The attribute name to filter on.
        filter_value (any, optional): The value to compare against in the filter.
        limit (int, optional): The maximum number of items to evaluate. Defaults
 to 10.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition, "Limit": limit}

    # Add the filter expression if filter attributes are provided
    if filter_attribute and filter_value is not None:
        query_params["FilterExpression"] =
 Attr(filter_attribute).gt(filter_value)
        query_params["ExpressionAttributeValues"] = {":filter_value":
 filter_value}

    # Execute the query
    response = table.query(**query_params)
    return response
```

Demonstra como usar expressões de filtro com limites em AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the query_with_filter_and_limit function."""
    # Example parameters
    table_name = "ProductReviews"
    partition_key_name = "ProductId"
    partition_key_value = "P123456"
    filter_attribute = "Rating"
    filter_value = 3  # Filter for ratings > 3
    limit = 5

    print(f"Querying reviews for product '{partition_key_value}' with rating >
 {filter_value}")
    print(f"Limiting to {limit} evaluated items")

    # Execute the query with filter and limit
    response = query_with_filter_and_limit(
        table_name, partition_key_name, partition_key_value, filter_attribute,
 filter_value, limit
    )

    # Process the results
    items = response.get("Items", [])
    print(f"\nReturned {len(items)} items that passed the filter")

    for item in items:
        print(f"Review: {item}")

    # Explain the difference between Limit and actual results
    explain_limit_vs_results(response)

    # Check if there are more results
    if "LastEvaluatedKey" in response:
        print("\nThere are more results available. Use the LastEvaluatedKey for
 pagination.")
    else:
        print("\nAll matching results have been retrieved.")
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela do DynamoDB com atributos aninhados usando um AWS SDK

Os exemplos de código a seguir mostram como consultar uma tabela com atributos aninhados.

- Acesse e filtre por atributos aninhados nos itens do DynamoDB.
- Use expressões de caminho de documento para se referir a elementos aninhados.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB com atributos aninhados usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithNestedAttributes(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String nestedPath,
        final String nestedAttr,
        final String nestedValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Nested path", nestedPath);
        CodeSampleUtils.validateStringParameter("Nested attribute", nestedAttr);
        CodeSampleUtils.validateStringParameter("Nested value", nestedValue);

        // Split the nested path into components
        final String[] pathComponents = nestedPath.split("\\.");

        // Create expression attribute names for the column names
```

```
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Build the nested attribute reference using document path notation
        final StringBuilder nestedAttributeRef = new StringBuilder();
        for (int i = 0; i < pathComponents.length; i++) {
            final String aliasName = "#n" + i;
            expressionAttributeNames.put(aliasName, pathComponents[i]);

            if (i > 0) {
                nestedAttributeRef.append(".");
            }
            nestedAttributeRef.append(aliasName);
        }

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_NESTED,
            AttributeValue.builder().s(nestedValue).build());

        // Create the filter expression using the nested attribute reference
        final String filterExpression = nestedAttributeRef + " = :nestedValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query with nested attribute filter successful.
 Found " + response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
```

```
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with nested attribute filter: " +
e.getMessage());
            throw e;
        }
    }
```

Demonstra como consultar uma tabela do DynamoDB com atributos aninhados.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<nestedPath> <nestedAttr> <nestedValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    nestedPath - The path to the nested map attribute (e.g.,
"address").
                    nestedAttr - The name of the nested attribute (e.g., "city").
                    nestedValue - The value to filter by (e.g., "Seattle").
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String nestedPath = args[3];
        final String nestedAttr = args[4];
        final String nestedValue = args[5];
```

```
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        System.out.println("Querying items where " + partitionKeyName + " = " +
partitionKeyValue + " and " + nestedPath
            + "." + nestedAttr + " = " + nestedValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new NestedAttributeQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withNestedPath(nestedPath)
                .withNestedAttribute(nestedAttr)
                .withNestedValue(nestedValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> {
                System.out.println(item);

                // Extract and display the nested attribute for clarity
                if (item.containsKey(nestedPath) && item.get(nestedPath).hasM())
{
                    Map<String, AttributeValue> nestedMap =
item.get(nestedPath).m();
                    if (nestedMap.containsKey(nestedAttr)) {
                        System.out.println("  Nested attribute " + nestedPath +
"." + nestedAttr + ": "
                                + formatAttributeValue(nestedMap.get(nestedAttr)));
                    }
                }
            });

            System.out.println("\nNote: When working with nested attributes in
DynamoDB:");
            System.out.println("1. Use dot notation in filter expressions to
access nested attributes");
            System.out.println("2. Use expression attribute names for each
component of the path");
```

```
            System.out.println("3. Check if the nested attribute exists before
 accessing it");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com atributos aninhados usando. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table filtering on a nested attribute
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} productId - The product ID to query by (partition key)
 * @param {string} category - The category to filter by (nested attribute)
 * @returns {Promise<Object>} - The query response
 */
async function queryWithNestedAttribute(
  config,
  tableName,
  productId,
```

```
  category
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "product_id = :productId",
      FilterExpression: "details.category = :category",
      ExpressionAttributeValues: {
        ":productId": { S: productId },
        ":category": { S: category }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with nested attribute: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com atributos aninhados usando. AWS SDK para Python (Boto3)

```
from typing import Any, Dict, List

import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_nested_attributes(
```

```
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    nested_path: str,
    comparison_operator: str,
    comparison_value: Any,
) -> Dict[str, Any]:
    """
    Query a DynamoDB table and filter by nested attributes.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        nested_path (str): The path to the nested attribute (e.g.,
 'specs.weight').
        comparison_operator (str): The comparison operator to use ('=', '!=',
 '<', '<=', '>', '>=').
        comparison_value (any): The value to compare against.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build the filter expression based on the nested attribute path and
 comparison operator
    filter_expression = None
    if comparison_operator == "=":
        filter_expression = Attr(nested_path).eq(comparison_value)
    elif comparison_operator == "!=":
        filter_expression = Attr(nested_path).ne(comparison_value)
    elif comparison_operator == "<":
        filter_expression = Attr(nested_path).lt(comparison_value)
    elif comparison_operator == "<=":
        filter_expression = Attr(nested_path).lte(comparison_value)
    elif comparison_operator == ">":
        filter_expression = Attr(nested_path).gt(comparison_value)
    elif comparison_operator == ">=":
```

```
        filter_expression = Attr(nested_path).gte(comparison_value)
    elif comparison_operator == "contains":
        filter_expression = Attr(nested_path).contains(comparison_value)
    elif comparison_operator == "begins_with":
        filter_expression = Attr(nested_path).begins_with(comparison_value)

    # Execute the query with the filter expression
    response = table.query(KeyConditionExpression=key_condition,
 FilterExpression=filter_expression)

    return response


def query_with_multiple_nested_attributes(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    nested_conditions: List[Dict[str, Any]],
) -> Dict[str, Any]:
    """
    Query a DynamoDB table and filter by multiple nested attributes.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        nested_conditions (list): A list of dictionaries, each containing:
            - path (str): The path to the nested attribute
            - operator (str): The comparison operator
            - value (any): The value to compare against

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build the combined filter expression for all nested attributes
    combined_filter = None
```

```
    for condition in nested_conditions:
        if not isinstance(condition, dict):
            continue
        path = condition.get("path", "")
        operator = condition.get("operator", "")
        value = condition.get("value")

        if not path or not operator:
            continue

        # Build the individual filter expression
        current_filter = None
        if operator == "=":
            current_filter = Attr(path).eq(value)
        elif operator == "!=":
            current_filter = Attr(path).ne(value)
        elif operator == "<":
            current_filter = Attr(path).lt(value)
        elif operator == "<=":
            current_filter = Attr(path).lte(value)
        elif operator == ">":
            current_filter = Attr(path).gt(value)
        elif operator == ">=":
            current_filter = Attr(path).gte(value)
        elif operator == "contains":
            current_filter = Attr(path).contains(value)
        elif operator == "begins_with":
            current_filter = Attr(path).begins_with(value)

        # Combine with the existing filter using AND
        if current_filter:
            if combined_filter is None:
                combined_filter = current_filter
            else:
                combined_filter = combined_filter & current_filter

    # Execute the query with the combined filter expression
    response = table.query(KeyConditionExpression=key_condition,
FilterExpression=combined_filter)

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela do DynamoDB com paginação usando um AWS SDK

Os exemplos de código a seguir mostram como consultar uma tabela com paginação.

- Implemente a paginação para os resultados da consulta do DynamoDB.
- Use o LastEvaluatedKey para recuperar as páginas subsequentes.
- Controle o número de itens por página com o parâmetro Limit.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB com paginação usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
```

```
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
```

```
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total
of " + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " +
e.getMessage());
            throw e;
        }
    }
```

Demonstra como consultar uma tabela do DynamoDB com paginação.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> [pageSize]
[region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    pageSize (optional) - The maximum number of items to return
per page. (Default: 10)
```

```
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final int pageSize = args.length > 3 ? Integer.parseInt(args[3]) : 10;
        final Region region = args.length > 4 ? Region.of(args[4]) :
Region.US_EAST_1;

        System.out.println("Querying items with pagination (page size: " +
pageSize + ")");

        try {
            // Using the builder pattern to create and execute the query
            final List<Map<String, AttributeValue>> allItems = new
PaginationQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withPageSize(pageSize)
                .withRegion(region)
                .executeWithPagination();

            // Process the results
            System.out.println("\nSummary: Retrieved a total of " +
allItems.size() + " items");

            // Display the first few items as a sample
            final int sampleSize = Math.min(5, allItems.size());
            if (sampleSize > 0) {
                System.out.println("\nSample of retrieved items (first " +
sampleSize + "):");
                for (int i = 0; i < sampleSize; i++) {
                    System.out.println(allItems.get(i));
                }

                if (allItems.size() > sampleSize) {
```

```
                    System.out.println("... and " + (allItems.size() -
 sampleSize) + " more items");
                }
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com paginação usando. AWS SDK para JavaScript

```
/**
 * Example demonstrating how to handle large query result sets in DynamoDB using
 pagination
 *
 * This example shows:
 * - How to use pagination to handle large result sets
 * - How to use LastEvaluatedKey to retrieve the next page of results
 * - How to construct subsequent query requests using ExclusiveStartKey
 */
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with pagination to handle large result sets
 *
```

```
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number} pageSize - Number of items per page
 * @returns {Promise<Array>} - All items from the query
 */
async function queryWithPagination(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  pageSize = 25
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize variables for pagination
    let lastEvaluatedKey = undefined;
    const allItems = [];
    let pageCount = 0;

    // Loop until all pages are retrieved
    do {
      // Construct the query input
      const input = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pkValue",
        Limit: pageSize,
        ExpressionAttributeNames: {
          "#pk": partitionKeyName
        },
        ExpressionAttributeValues: {
          ":pkValue": { S: partitionKeyValue }
        }
      };

      // Add ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
      if (lastEvaluatedKey) {
        input.ExclusiveStartKey = lastEvaluatedKey;
      }
```

```
      // Execute the query
      const command = new QueryCommand(input);
      const response = await client.send(command);

      // Process the current page of results
      pageCount++;
      console.log(`Processing page ${pageCount} with ${response.Items.length}
 items`);

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems.push(...response.Items);
      }

      // Get the LastEvaluatedKey for the next page
      lastEvaluatedKey = response.LastEvaluatedKey;

    } while (lastEvaluatedKey); // Continue until there are no more pages

    console.log(`Query complete. Retrieved ${allItems.length} items in
 ${pageCount} pages.`);
    return allItems;
  } catch (error) {
    console.error(`Error querying with pagination: ${error}`);
    throw error;
  }
}

/**
 * Example usage:
 *
 * // Query all items in the "AWS DynamoDB" forum with pagination
 * const allItems = await queryWithPagination(
 *   { region: "us-west-2" },
 *   "ForumThreads",
 *   "ForumName",
 *   "AWS DynamoDB",
 *   25 // 25 items per page
 * );
 *
 * console.log(`Total items retrieved: ${allItems.length}`);
 *
 * // Notes on pagination:
 * // - LastEvaluatedKey contains the primary key of the last evaluated item
```

```
 * // - When LastEvaluatedKey is undefined/null, there are no more items to
 retrieve
 * // - ExclusiveStartKey tells DynamoDB where to start the next page
 * // - Pagination helps manage memory usage for large result sets
 * // - Each page requires a separate network request to DynamoDB
 */

module.exports = { queryWithPagination };
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com paginação usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_pagination(
    table_name, partition_key_name, partition_key_value, page_size=25,
 max_pages=None
):
    """
    Query a DynamoDB table with pagination to handle large result sets.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page.
 Defaults to 25.
        max_pages (int, optional): The maximum number of pages to retrieve. If
 None, retrieves all pages.

    Returns:
        list: All items retrieved from the query across all pages.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
```

```
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_count = 0
    all_items = []

    # Paginate through the results
    while True:
        # Check if we've reached the maximum number of pages
        if max_pages is not None and page_count >= max_pages:
            break

        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Process the current page of results
        items = response.get("Items", [])
        all_items.extend(items)

        # Update pagination tracking
        page_count += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if not last_evaluated_key:
            break

    return all_items
```

```
def query_with_pagination_generator(
    table_name, partition_key_name, partition_key_value, page_size=25
):
    """
    Query a DynamoDB table with pagination using a generator to handle large
 result sets.
    This approach is memory-efficient as it yields one page at a time.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page.
 Defaults to 25.

    Yields:
        tuple: A tuple containing (items, page_number, last_page) where:
            - items is a list of items for the current page
            - page_number is the current page number (starting from 1)
            - last_page is a boolean indicating if this is the last page
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_number = 0

    # Paginate through the results
    while True:
        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key
```

```
        # Execute the query
        response = table.query(**query_params)

        # Get the current page of results
        items = response.get("Items", [])
        page_number += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # Determine if this is the last page
        is_last_page = last_evaluated_key is None

        # Yield the current page of results
        yield (items, page_number, is_last_page)

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if is_last_page:
            break
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela do DynamoDB com leituras altamente consistentes usando um AWS SDK

Os exemplos de código a seguir mostram como consultar uma tabela com leituras altamente consistentes.

- Configure o nível de consistência das consultas do DynamoDB.
- Use leituras altamente consistentes para obter os dados mais atualizados.
- Entenda as vantagens e desvantagens entre consistência final e consistência forte.

Java

SDK para Java 2.x

Consulte uma tabela do DynamoDB com consistência de leitura configurável usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
```

```
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
 response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com consistência de leitura configurável usando. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with configurable read consistency
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {boolean} useConsistentRead - Whether to use strongly consistent reads
 * @returns {Promise<Object>} - The query response
 */
async function queryWithConsistentRead(
  config,
```

```
  tableName,
  partitionKeyName,
  partitionKeyValue,
  useConsistentRead = false
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue }
      },
      ConsistentRead: useConsistentRead
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with consistent read: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com a opção de leituras altamente consistentes usando. AWS SDK para Python (Boto3)

```
import time
```

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_consistent_read(
    table_name,
    partition_key_name,
    partition_key_value,
    sort_key_name=None,
    sort_key_value=None,
    consistent_read=True,
):
    """
    Query a DynamoDB table with the option for strongly consistent reads.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str, optional): The name of the sort key attribute.
        sort_key_value (str, optional): The value of the sort key to query.
        consistent_read (bool, optional): Whether to use strongly consistent
 reads. Defaults to True.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    if sort_key_name and sort_key_value:
        key_condition = key_condition & Key(sort_key_name).eq(sort_key_value)

    # Perform the query with the consistent read option
    response = table.query(KeyConditionExpression=key_condition,
 ConsistentRead=consistent_read)

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte dados do DynamoDB usando instruções PARTIQL SELECT com um AWS SDK

O exemplo de código a seguir mostra como consultar dados usando declarações SELECT do PartiQL.

JavaScript

SDK para JavaScript (v3)

Consulte itens de uma tabela do DynamoDB usando instruções PARTIQL SELECT com. AWS SDK para JavaScript

```
/**
 * This example demonstrates how to query items from a DynamoDB table using
 PartiQL.
 * It shows different ways to select data with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Select all items from a DynamoDB table using PartiQL.
 * Note: This should be used with caution on large tables.
 *
 * @param tableName - The name of the DynamoDB table
 * @returns The response from the ExecuteStatementCommand
 */
export const selectAllItems = async (tableName: string) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
```

```
    Statement: `SELECT * FROM "${tableName}"`,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select an item by its primary key using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemByPartitionKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${partitionKeyName} = ?`,
    Parameters: [partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving item:", err);
    throw err;
  }
};
```

```
/**
 * Select an item by its composite key (partition key + sort key) using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemByCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${sortKeyName} = ?`,
    Parameters: [partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving item:", err);
    throw err;
  }
};

/**
 * Select items using a filter condition with PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param filterAttribute - The attribute to filter on
 * @param filterValue - The value to filter by
 * @returns The response from the ExecuteStatementCommand
 */
```

```
export const selectItemsWithFilter = async (
  tableName: string,
  filterAttribute: string,
  filterValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${filterAttribute} = ?`,
    Parameters: [filterValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select items using a begins_with function for prefix matching.
 * This is useful for querying hierarchical data.
 *
 * @param tableName - The name of the DynamoDB table
 * @param attributeName - The attribute to check for prefix
 * @param prefix - The prefix to match
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsByPrefix = async (
  tableName: string,
  attributeName: string,
  prefix: string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE
 begins_with(${attributeName}, ?)`,
    Parameters: [prefix],
```

```
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select items using a between condition for range queries.
 *
 * @param tableName - The name of the DynamoDB table
 * @param attributeName - The attribute to check for range
 * @param startValue - The start value of the range
 * @param endValue - The end value of the range
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsByRange = async (
  tableName: string,
  attributeName: string,
  startValue: number | string,
  endValue: number | string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${attributeName} BETWEEN ?
 AND ?`,
    Parameters: [startValue, endValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
```

```
};

/**
 * Example usage showing how to select items with different index types
 */
export const selectExamples = async () => {
  // Select all items from a table (use with caution on large tables)
  await selectAllItems("UsersTable");

  // Select by partition key (simple primary key)
  await selectItemByPartitionKey("UsersTable", "userId", "user123");

  // Select by composite key (partition key + sort key)
  await selectItemByCompositeKey("OrdersTable", "orderId", "order456",
 "productId", "prod789");

  // Select with a filter condition (can use any attribute)
  await selectItemsWithFilter("UsersTable", "userType", "premium");

  // Select items with a prefix (useful for hierarchical data)
  await selectItemsByPrefix("ProductsTable", "category", "electronics");

  // Select items within a range (useful for numeric or date ranges)
  await selectItemsByRange("OrdersTable", "orderDate", "2023-01-01",
 "2023-12-31");
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchExecuteStatement
  - ExecuteStatement

Consulte uma tabela do DynamoDB para itens TTL usando um AWS SDK

Os exemplos de código a seguir mostram como consultar itens com TTL.

Java

SDK para Java 2.x

Consulte a expressão filtrada para reunir itens TTL em uma tabela do DynamoDB usando o. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Map;
import java.util.Optional;

        final QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final QueryResponse response = ddb.query(request);
            System.out.println("Query successful. Found " + response.count() + "
 items that have not expired yet.");

            // Print each item
            response.items().forEach(item -> {
                System.out.println("Item: " + item);
            });

            return 0;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
```

```
            throw e;
        }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte a expressão filtrada para reunir itens TTL em uma tabela do DynamoDB usando o. AWS SDK para JavaScript

```
import { DynamoDBClient, QueryCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const queryFiltered = async (tableName, primaryKey, region = 'us-east-1')
 => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);

    const params = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pk",
        FilterExpression: "#ea > :ea",
        ExpressionAttributeNames: {
            "#pk": "primaryKey",
            "#ea": "expireAt"
        },
        ExpressionAttributeValues: marshall({
            ":pk": primaryKey,
            ":ea": currentTime
        })
    };

    try {
        const { Items } = await client.send(new QueryCommand(params));
        Items.forEach(item => {
            console.log(unmarshall(item))
```

```
        });
        return Items;
    } catch (err) {
        console.error(`Error querying items: ${err}`);
        throw err;
    }
}

// Example usage (commented out for testing)
// queryFiltered('your-table-name', 'your-partition-key-value');
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte a expressão filtrada para reunir itens TTL em uma tabela do DynamoDB usando o. AWS SDK para Python (Boto3)

```
from datetime import datetime

import boto3


def query_dynamodb_items(table_name, partition_key):
    """

    :param table_name: Name of the DynamoDB table
    :param partition_key:
    :return:
    """
    try:
        # Initialize a DynamoDB resource
        dynamodb = boto3.resource("dynamodb", region_name="us-east-1")

        # Specify your table
        table = dynamodb.Table(table_name)

        # Get the current time in epoch format
        current_time = int(datetime.now().timestamp())
```

```
        # Perform the query operation with a filter expression to exclude expired
items
        # response = table.query(
        #
KeyConditionExpression=boto3.dynamodb.conditions.Key('partitionKey').eq(partition_key),
        #
FilterExpression=boto3.dynamodb.conditions.Attr('expireAt').gt(current_time)
        # )
        response = table.query(

KeyConditionExpression=dynamodb.conditions.Key("partitionKey").eq(partition_key),

FilterExpression=dynamodb.conditions.Attr("expireAt").gt(current_time),
        )

        # Print the items that are not expired
        for item in response["Items"]:
            print(item)

    except Exception as e:
        print(f"Error querying items: {e}")


# Call the function with your values
query_dynamodb_items("Music", "your-partition-key-value")
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte tabelas do DynamoDB usando padrões de data e hora com um AWS SDK

Os exemplos de código a seguir mostram como consultar tabelas usando padrões de data e hora.

- Armazene e consulte date/time valores no DynamoDB.
- Implemente consultas de intervalo de datas usando chaves de classificação.
- Formate strings de data para obter uma consulta eficaz.

Java

SDK para Java 2.x

Consulte usando intervalos de datas em chaves de classificação com AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start
 date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
```

```
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

Consulte usando variáveis de data e hora com. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final String startDate,
        final String endDate) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateDateRangeParameters(dateKeyName, startDate,
 endDate);
        CodeSampleUtils.validateDateFormat("Start date", startDate);
        CodeSampleUtils.validateDateFormat("End date", endDate);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate", AttributeValue.builder().s(startDate).build());
        expressionAttributeValues.put(
```

```
            ":endDate", AttributeValue.builder().s(endDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
 items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with date range: " +
 e.getMessage());
            throw e;
        }
    }
```

Consulte dentro dos intervalos de datas nos carimbos de data/hora de época do Unix com o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;
```

```
    public QueryResponse queryWithDateTimeEpoch(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final long startEpoch,
        final long endEpoch) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        CodeSampleUtils.validateEpochTimestamp("Start epoch", startEpoch);
        CodeSampleUtils.validateEpochTimestamp("End epoch", endEpoch);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate",
AttributeValue.builder().n(String.valueOf(startEpoch)).build());
        expressionAttributeValues.put(
            ":endDate",
AttributeValue.builder().n(String.valueOf(endEpoch)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
```

```
            System.out.println("Query successful. Found " + response.count() + "
 items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with epoch timestamps: " +
 e.getMessage());
            throw e;
        }
    }
```

Consulte dentro de intervalos de datas usando LocalDateTime objetos com AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeLocalDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDateTime startDateTime,
        final LocalDateTime endDateTime) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        if (startDateTime == null || endDateTime == null) {
```

```
            throw new IllegalArgumentException("Start and end LocalDateTime must
 not be null");
        }

        // Convert LocalDateTime to ISO-8601 strings in UTC with the correct
 format
        final String startDate =
 startDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);
        final String endDate =
 endDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);

        return queryWithDateTime(tableName, partitionKeyName, partitionKeyValue,
 dateKeyName, startDate, endDate);
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

Consulte usando intervalos de datas em chaves de classificação com AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range on the sort
 key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key (must be a date/time
 attribute)
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRangeOnSortKey(
  config,
```

```
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: '#pk = :pkValue AND #sk BETWEEN :startDate
 AND :endDate',
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range on sort key: ${error}`);
    throw error;
  }
}
```

Consulte usando variáveis de data e hora com. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} dateKeyName - The name of the date attribute to filter on
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRange(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  dateKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: `#pk = :pkValue AND #dateAttr BETWEEN :startDate
 AND :endDate`,
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#dateAttr": dateKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
```

```
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Python

SDK para Python (Boto3)

Consulte usando intervalos de datas em chaves de classificação com AWS SDK para Python (Boto3).

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_date_range(
    table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
):
    """
    Query a DynamoDB table with a date range on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        start_date (datetime): The start date for the query range.
```

```
        end_date (datetime): The end date for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date values as ISO 8601 strings
    # DynamoDB works well with ISO format for date values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key using BETWEEN operator
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response


def query_with_date_range_by_month(
    table_name, partition_key_name, partition_key_value, sort_key_name, year,
 month
):
    """
    Query a DynamoDB table for a specific month's data.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
```

```
        sort_key_name (str): The name of the sort key attribute (containing date
values).
        year (int): The year to query.
        month (int): The month to query (1-12).

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Calculate the start and end dates for the specified month
    if month == 12:
        next_year = year + 1
        next_month = 1
    else:
        next_year = year
        next_month = month + 1

    start_date = datetime(year, month, 1)
    end_date = datetime(next_year, next_month, 1) - timedelta(microseconds=1)

    # Format the date values as ISO 8601 strings
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(KeyConditionExpression=key_condition)

    return response
```

Consulte usando variáveis de data e hora com. AWS SDK para Python (Boto3)

```
from datetime import datetime, timedelta
```

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_datetime(
    table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
):
    """
    Query a DynamoDB table with a date range filter on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date/
time values).
        start_date (datetime): The start date/time for the query range.
        end_date (datetime): The end date/time for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date/time values as ISO 8601 strings
    # DynamoDB works well with ISO format for date/time values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
```

```
            ":end_date": end_date_str,
        },
    )

    return response



def example_usage():
    """Example of how to use the query_with_datetime function."""
    # Example parameters
    table_name = "Events"
    partition_key_name = "EventType"
    partition_key_value = "UserLogin"
    sort_key_name = "Timestamp"

    # Create date/time variables for the query
    end_date = datetime.now()
    start_date = end_date - timedelta(days=7)  # Query events from the last 7
 days

    print(f"Querying events from {start_date.isoformat()} to
 {end_date.isoformat()}")

    # Execute the query
    response = query_with_datetime(
        table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
    )

    # Process the results
    items = response.get("Items", [])
    print(f"Found {len(items)} items")

    for item in items:
        print(f"Event: {item}")
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Salve EXIF e outras informações da imagem usando um AWS SDK

O exemplo de código a seguir mostra como:

- Obter informações de EXIF de um arquivo JPG, JPEG ou PNG.
- Fazer upload do arquivo de imagem para um bucket do Amazon S3.
- Usar o Amazon Rekognition para identificar os três principais atributos (rótulos) no arquivo.
- Adicionar as informações de EXIF e rótulo a uma tabela do Amazon DynamoDB na região.

Rust

SDK para Rust

Obtenha informações de EXIF de um arquivo JPG, JPEG ou PNG, faça upload do arquivo de imagem para um bucket do Amazon S3, use o Amazon Rekognition para identificar os três principais atributos (rótulos no Amazon Rekognition) no arquivo e adicione as informações de EXIF e de rótulo a uma tabela do Amazon DynamoDB na região.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Configure o controle de Attribute-Based acesso para o DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como implementar o Controle de Attribute-Based Acesso (ABAC) para o DynamoDB.

- Crie uma política do IAM para o ABAC.
- Crie tabelas com etiquetas que representem diferentes departamentos.
- Liste e filtre tabelas com base nas etiquetas.

Bash

AWS CLI com script Bash

Crie uma política do IAM para o ABAC.

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
```

```
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

Crie tabelas com etiquetas que representem diferentes departamentos.

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

Liste e filtre tabelas com base nas etiquetas.

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
```

```
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text |
 xargs -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn"
 --output text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreatePolicy
  - CreateTable
  - ListTables

Entenda a ordem da expressão de atualização no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como funciona a ordem da expressão de atualização.

- Saiba como o DynamoDB processa expressões de atualização.
- Entenda como funciona a ordem das operações nas expressões de atualização.
- Evite resultados inesperados entendendo a avaliação da expressão.

Java

SDK para Java 2.x

Demonstre a ordem da expressão de atualização usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Demonstrates the effect of update expression order.
     *
     * <p>This method shows how the order of operations in an update expression
     * affects the result of the update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the results of different update orders
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> demonstrateUpdateOrder(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
 AttributeValue> key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Initialize the item with a counter
            UpdateItemRequest initRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :zero, OldCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
```

```
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            dynamoDbClient.updateItem(initRequest);

            // Example 1: SET first, then ADD
            UpdateItemRequest setFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value ADD
OldCounter :increment")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse setFirstResponse =
dynamoDbClient.updateItem(setFirstRequest);
            results.put("setFirstResponse", setFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 2: ADD first, then SET
            UpdateItemRequest addFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("ADD Counter :increment SET OldCounter
= :value")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse addFirstResponse =
dynamoDbClient.updateItem(addFirstRequest);
            results.put("addFirstResponse", addFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 3: SET with multiple attributes
```

```
            UpdateItemRequest multiSetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value, OldCounter = Counter")
                .expressionAttributeValues(
                    Map.of(":value", AttributeValue.builder().n("10").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse multiSetResponse =
dynamoDbClient.updateItem(multiSetRequest);
            results.put("multiSetResponse", multiSetResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 4: SET with expression using the same attribute
            UpdateItemRequest selfReferenceRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = Counter + :increment, OldCounter
= Counter")
                .expressionAttributeValues(
                    Map.of(":increment",
AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse selfReferenceResponse =
dynamoDbClient.updateItem(selfReferenceRequest);
            results.put("selfReferenceResponse", selfReferenceResponse);

            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Updates an item with SET first, then REMOVE.
```

```
     *
     * <p>This method demonstrates updating an item with SET operation first,
     * followed by a REMOVE operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithSetFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue REMOVE #removeAttr")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with SET first: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }
```

```
    /**
     * Updates an item with REMOVE first, then SET.
     *
     * <p>This method demonstrates updating an item with REMOVE operation first,
     * followed by a SET operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithRemoveFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #removeAttr SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with REMOVE first: " +
e.getMessage())
                .cause(e)
```

```
                .build();
        }
    }

    /**
     * Updates an item with all operation types in a specific order.
     *
     * <p>This method demonstrates using all operation types (SET, REMOVE, ADD,
DELETE)
     * in a specific order in a single update expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithAllOperationTypes(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #stringAttr = :stringVal, #mapAttr.#nestedAttr
= :nestedVal " + "REMOVE #oldAttr "
                + "ADD #counterAttr :increment "
                + "DELETE #stringSetAttr :stringSetVal")
            .expressionAttributeNames(Map.of(
                "#stringAttr", "StringAttribute",
                "#mapAttr", "MapAttribute",
                "#nestedAttr", "NestedAttribute",
                "#oldAttr", "OldAttribute",
                "#counterAttr", "CounterAttribute",
                "#stringSetAttr", "StringSetAttribute"))
            .expressionAttributeValues(Map.of(
                ":stringVal", AttributeValue.builder().s("New Value").build(),
                ":nestedVal", AttributeValue.builder().s("Nested Value").build(),
                ":increment", AttributeValue.builder().n("1").build(),
                ":stringSetVal", AttributeValue.builder().ss("Value1").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();
```

```
        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with all operation types: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Gets the current state of an item.
     *
     * <p>Helper method to retrieve the current state of an item.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @return The item or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getItem(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key) {

        // Define the get parameters
        GetItemRequest request =
            GetItemRequest.builder().tableName(tableName).key(key).build();

        // Perform the get operation
        try {
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the item if it exists, otherwise null
            return response.item();
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get item: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Exemplo de uso da ordem de expressão de atualização com AWS SDK for Java 2.x.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating update expression order in DynamoDB");

       try {
           // Example 1: Demonstrate update order effects
           System.out.println("\nExample 1: Demonstrating update order
effects");
           Map<String, Object> orderResults =
demonstrateUpdateOrder(dynamoDbClient, tableName, key);

           if ((boolean) orderResults.get("success")) {
               System.out.println("SET first, then ADD:");
               System.out.println("  " + orderResults.get("setFirstResponse"));

               System.out.println("ADD first, then SET:");
               System.out.println("  " + orderResults.get("addFirstResponse"));

               System.out.println("SET with multiple attributes:");
               System.out.println("  " + orderResults.get("multiSetResponse"));

               System.out.println("SET with self-reference:");
               System.out.println("  " +
orderResults.get("selfReferenceResponse"));
           } else {
               System.out.println("Error: " + orderResults.get("error"));
           }

           // Example 2: Update with SET first, then REMOVE
           System.out.println("\nExample 2: Update with SET first, then
REMOVE");
           UpdateItemResponse setFirstResponse = updateWithSetFirst(
               dynamoDbClient,
               tableName,
               key,
```

```
                "Status",
                AttributeValue.builder().s("Active").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
setFirstResponse.attributes());

            // Example 3: Update with REMOVE first, then SET
            System.out.println("\nExample 3: Update with REMOVE first, then
SET");
            UpdateItemResponse removeFirstResponse = updateWithRemoveFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Inactive").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
removeFirstResponse.attributes());

            // Example 4: Update with all operation types
            System.out.println("\nExample 4: Update with all operation types");
            UpdateItemResponse allOpsResponse =
updateWithAllOperationTypes(dynamoDbClient, tableName, key);

            System.out.println("Updated attributes: " +
allOpsResponse.attributes());

            // Example 5: Get the current state of the item
            System.out.println("\nExample 5: Current state of the item");
            Map<String, AttributeValue> item = getItem(dynamoDbClient, tableName,
key);

            if (item != null) {
                System.out.println("Item: " + item);
            } else {
                System.out.println("Item not found");
            }

            // Explain update expression order
            System.out.println("\nKey points about update expression order in
DynamoDB:");
```

```
            System.out.println("1. Update expressions are processed in this
 order: SET, REMOVE, ADD, DELETE");
            System.out.println("2. Within each clause, operations are processed
 from left to right");
            System.out.println("3. SET operations use the item state before any
 updates in the expression");
            System.out.println("4. When an attribute is referenced multiple
 times, the first operation wins");
            System.out.println("5. To reference a new value, split the update
 into multiple operations");
            System.out.println("6. The order of clauses in the expression doesn't
 change the evaluation order");
            System.out.println("7. For complex updates, consider using multiple
 separate update operations");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Demonstre a ordem da expressão de atualização usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update an item with multiple actions in a single update expression.
 *
```

```
 * This function demonstrates how to use multiple actions in a single update
 expression
 * and how DynamoDB processes these actions.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {string} updateExpression - The update expression with multiple actions
 * @param {Object} [expressionAttributeNames] - Expression attribute name
 placeholders
 * @param {Object} [expressionAttributeValues] - Expression attribute value
 placeholders
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleActions(
  config,
  tableName,
  key,
  updateExpression,
  expressionAttributeNames,
  expressionAttributeValues
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Prepare the update parameters
  const updateParams = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ReturnValues: "UPDATED_NEW"
  };

  // Add expression attribute names if provided
  if (expressionAttributeNames) {
    updateParams.ExpressionAttributeNames = expressionAttributeNames;
  }

  // Add expression attribute values if provided
  if (expressionAttributeValues) {
    updateParams.ExpressionAttributeValues = expressionAttributeValues;
  }
```

```
  // Execute the update
  const response = await docClient.send(new UpdateCommand(updateParams));

  return response;
}

/**
 * Demonstrate that variables hold copies of existing values before
 modifications.
 *
 * This function creates an item with initial values, then updates it with an
 expression
 * that uses the values of attributes before they are modified in the same
 expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateValueCopying(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = { ...key, a: 1, b: 2, c: 3 };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));
```

```
  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with an expression that uses values before they are
 modified
  // This expression removes 'a', then sets 'b' to the value of 'a', and 'c' to
 the value of 'b'
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: "REMOVE a SET b = a, c = b",
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}

/**
 * Demonstrate the order in which different action types are processed.
 *
 * This function creates an item with initial values, then updates it with an
 expression
 * that includes multiple action types (SET, REMOVE, ADD, DELETE) to show the
 order
 * in which they are processed.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
```

```
async function demonstrateActionOrder(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = {
    ...key,
    counter: 10,
    set_attr: new Set(["A", "B", "C"]),
    to_remove: "This will be removed",
    to_modify: "Original value"
  };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with multiple action types
  // The actions will be processed in this order: REMOVE, SET, ADD, DELETE
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: "REMOVE to_remove SET to_modify = :new_value ADD
 counter :increment DELETE set_attr :elements",
    ExpressionAttributeValues: {
      ":new_value": "Updated value",
      ":increment": 5,
      ":elements": new Set(["B"])
    },
    ReturnValues: "UPDATED_NEW"
```

```
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}

/**
 * Update multiple attributes with a single SET action.
 *
 * This function demonstrates how to update multiple attributes in a single SET
 action,
 * which is more efficient than using multiple separate update operations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {Object} attributes - The attributes to update and their new values
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleSetActions(
  config,
  tableName,
  key,
  attributes
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = "SET ";
  const expressionAttributeValues = {};
```

```
  // Add each attribute to the update expression
  Object.entries(attributes).forEach(([attrName, attrValue], index) => {
    const valuePlaceholder = `:val${index}`;

    if (index > 0) {
      updateExpression += ", ";
    }
    updateExpression += `${attrName} = ${valuePlaceholder}`;

    expressionAttributeValues[valuePlaceholder] = attrValue;
  });

  // Execute the update
  const response = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  }));

  return response;
}

/**
 * Update an attribute with a value from another attribute or a default value.
 *
 * This function demonstrates how to use if_not_exists to conditionally copy a
 value
 * from one attribute to another, or use a default value if the source doesn't
 exist.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {string} sourceAttribute - The attribute to copy the value from
 * @param {string} targetAttribute - The attribute to update
 * @param {any} defaultValue - The default value to use if the source attribute
 doesn't exist
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithConditionalValueCopying(
  config,
```

```
  tableName,
  key,
  sourceAttribute,
  targetAttribute,
  defaultValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Use if_not_exists to conditionally copy the value
  const response = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${targetAttribute} =
 if_not_exists(${sourceAttribute}, :default)`,
    ExpressionAttributeValues: {
      ":default": defaultValue
    },
    ReturnValues: "UPDATED_NEW"
  }));

  return response;
}

/**
 * Demonstrate complex update expressions with multiple operations on the same
 attribute.
 *
 * This function shows how DynamoDB processes multiple operations on the same
 attribute
 * in a single update expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateMultipleOperationsOnSameAttribute(
  config,
  tableName,
  key
) {
```

```
// Initialize the DynamoDB client
const client = new DynamoDBClient(config);
const docClient = DynamoDBDocumentClient.from(client);

// Step 1: Create an item with initial values
const initialItem = {
  ...key,
  counter: 10,
  list_attr: [1, 2, 3],
  map_attr: {
    nested1: "value1",
    nested2: "value2"
  }
};

await docClient.send(new PutCommand({
  TableName: tableName,
  Item: initialItem
}));

// Step 2: Get the item to verify initial state
const responseBefore = await docClient.send(new GetCommand({
  TableName: tableName,
  Key: key
}));

const itemBefore = responseBefore.Item || {};

// Step 3: Update the item with multiple operations on the same attributes
const updateResponse = await docClient.send(new UpdateCommand({
  TableName: tableName,
  Key: key,
  UpdateExpression: `
    SET counter = counter + :inc1,
        counter = counter + :inc2,
        map_attr.nested1 = :new_val1,
        map_attr.nested3 = :new_val3,
        list_attr[0] = list_attr[1],
        list_attr[1] = list_attr[2]
  `,
  ExpressionAttributeValues: {
    ":inc1": 5,
    ":inc2": 3,
    ":new_val1": "updated_value1",
```

```
      ":new_val3": "new_value3"
    },
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}
```

Exemplo de uso da ordem de expressão de atualização com AWS SDK para JavaScript.

```
/**
 * Example of how to use update expression order of operations in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "OrderProcessing";

  console.log("Demonstrating update expression order of operations in DynamoDB");

  try {
    // Example 1: Demonstrating value copying in update expressions
    console.log("\nExample 1: Demonstrating value copying in update
 expressions");
    const results1 = await demonstrateValueCopying(
      config,
      tableName,
      { OrderId: "order123" }
    );
```

```
   console.log("Initial state:", JSON.stringify(results1.initialState, null,
2));
   console.log("Update response:", JSON.stringify(results1.updateResponse, null,
2));
   console.log("Final state:", JSON.stringify(results1.finalState, null, 2));

   console.log("\nExplanation:");
   console.log("1. The initial state had a=1, b=2, c=3");
   console.log("2. The update expression 'REMOVE a SET b = a, c = b' did the
following:");
   console.log("   - Copied the value of 'a' (which was 1) to be used for 'b'");
   console.log("   - Copied the value of 'b' (which was 2) to be used for 'c'");
   console.log("   - Removed the attribute 'a'");
   console.log("3. The final state has b=1, c=2, and 'a' is removed");
   console.log("4. This demonstrates that DynamoDB uses the values of attributes
as they were BEFORE any modifications");

   // Example 2: Demonstrating the order of different action types
   console.log("\nExample 2: Demonstrating the order of different action
types");
   const results2 = await demonstrateActionOrder(
     config,
     tableName,
     { OrderId: "order456" }
   );

   console.log("Initial state:", JSON.stringify(results2.initialState, null,
2));
   console.log("Update response:", JSON.stringify(results2.updateResponse, null,
2));
   console.log("Final state:", JSON.stringify(results2.finalState, null, 2));

   console.log("\nExplanation:");
   console.log("1. The update expression contained multiple action types:
REMOVE, SET, ADD, DELETE");
   console.log("2. DynamoDB processes these actions in this order: REMOVE, SET,
ADD, DELETE");
   console.log("3. First, 'to_remove' was removed");
   console.log("4. Then, 'to_modify' was set to a new value");
   console.log("5. Next, 'counter' was incremented by 5");
   console.log("6. Finally, 'B' was removed from the set attribute");

   // Example 3: Updating multiple attributes in a single SET action
```

```
   console.log("\nExample 3: Updating multiple attributes in a single SET
action");
   const response3 = await updateWithMultipleSetActions(
     config,
     tableName,
     { OrderId: "order789" },
     {
       Status: "Shipped",
       ShippingDate: "2025-05-28",
       TrackingNumber: "1Z999AA10123456784"
     }
   );

   console.log("Multiple attributes updated successfully:",
JSON.stringify(response3.Attributes, null, 2));

   // Example 4: Conditional value copying with if_not_exists
   console.log("\nExample 4: Conditional value copying with if_not_exists");
   const response4 = await updateWithConditionalValueCopying(
     config,
     tableName,
     { OrderId: "order101" },
     "PreferredShippingMethod",
     "ShippingMethod",
     "Standard"
   );

   console.log("Conditional value copying result:",
JSON.stringify(response4.Attributes, null, 2));

   // Example 5: Multiple operations on the same attribute
   console.log("\nExample 5: Multiple operations on the same attribute");
   const results5 = await demonstrateMultipleOperationsOnSameAttribute(
     config,
     tableName,
     { OrderId: "order202" }
   );

   console.log("Initial state:", JSON.stringify(results5.initialState, null,
2));
   console.log("Update response:", JSON.stringify(results5.updateResponse, null,
2));
   console.log("Final state:", JSON.stringify(results5.finalState, null, 2));
```

```
    console.log("\nExplanation:");
    console.log("1. The counter was incremented twice (first by 5, then by 3) for
 a total of +8");
    console.log("2. The map attribute had one value updated and a new nested
 attribute added");
    console.log("3. The list attribute had values shifted (value at index 1 moved
 to index 0, value at index 2 moved to index 1)");
    console.log("4. All operations within the SET action are processed from left
 to right");

    // Key points about update expression order of operations
    console.log("\nKey Points About Update Expression Order of Operations:");
    console.log("1. Variables in expressions hold copies of attribute values as
 they existed BEFORE any modifications");
    console.log("2. Multiple actions in an update expression are processed in
 this order: REMOVE, SET, ADD, DELETE");
    console.log("3. Within each action type, operations are processed from left
 to right");
    console.log("4. You can reference the same attribute multiple times in an
 expression");
    console.log("5. You can use if_not_exists() to conditionally set values based
 on attribute existence");
    console.log("6. Using a single update expression with multiple actions is
 more efficient than multiple separate updates");
    console.log("7. The update expression is atomic - either all actions succeed
 or none do");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Demonstre a ordem da expressão de atualização usando AWS SDK para Python (Boto3).

```
import boto3
```

```
import json
from typing import Any, Dict, Optional


def update_with_multiple_actions(
    table_name: str,
    key: Dict[str, Any],
    update_expression: str,
    expression_attribute_names: Optional[Dict[str, str]] = None,
    expression_attribute_values: Optional[Dict[str, Any]] = None,
) -> Dict[str, Any]:
    """
    Update an item with multiple actions in a single update expression.

    This function demonstrates how to use multiple actions in a single update
 expression
    and how DynamoDB processes these actions.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        update_expression (str): The update expression with multiple actions.
        expression_attribute_names (Optional[Dict[str, str]]): Expression
 attribute name placeholders.
        expression_attribute_values (Optional[Dict[str, Any]]): Expression
 attribute value placeholders.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Prepare the update parameters
    update_params = {
        "Key": key,
        "UpdateExpression": update_expression,
        "ReturnValues": "UPDATED_NEW",
    }

    # Add expression attribute names if provided
    if expression_attribute_names:
```

```
        update_params["ExpressionAttributeNames"] = expression_attribute_names

    # Add expression attribute values if provided
    if expression_attribute_values:
        update_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the update
    response = table.update_item(**update_params)

    return response


def demonstrate_value_copying(table_name: str, key: Dict[str, Any]) -> Dict[str,
 Any]:
    """
    Demonstrate that variables hold copies of existing values before
 modifications.

    This function creates an item with initial values, then updates it with an
 expression
    that uses the values of attributes before they are modified in the same
 expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to create and update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Step 1: Create an item with initial values
    initial_item = key.copy()
    initial_item.update({"a": 1, "b": 2, "c": 3})

    table.put_item(Item=initial_item)

    # Step 2: Get the item to verify initial state
    response_before = table.get_item(Key=key)
    item_before = response_before.get("Item", {})
```

```
    # Step 3: Update the item with an expression that uses values before they are
 modified
    # This expression removes 'a', then sets 'b' to the value of 'a', and 'c' to
 the value of 'b'
    update_response = table.update_item(
        Key=key, UpdateExpression="REMOVE a SET b = a, c = b",
 ReturnValues="UPDATED_NEW"
    )

    # Step 4: Get the item to verify final state
    response_after = table.get_item(Key=key)
    item_after = response_after.get("Item", {})

    # Return the results
    return {
        "initial_state": item_before,
        "update_response": update_response,
        "final_state": item_after,
    }


def demonstrate_action_order(table_name: str, key: Dict[str, Any]) -> Dict[str,
 Any]:
    """
    Demonstrate the order in which different action types are processed.

    This function creates an item with initial values, then updates it with an
 expression
    that includes multiple action types (SET, REMOVE, ADD, DELETE) to show the
 order
    in which they are processed.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to create and update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Step 1: Create an item with initial values
```

```
    initial_item = key.copy()
    initial_item.update(
        {
            "counter": 10,
            "set_attr": set(["A", "B", "C"]),
            "to_remove": "This will be removed",
            "to_modify": "Original value",
        }
    )

    table.put_item(Item=initial_item)

    # Step 2: Get the item to verify initial state
    response_before = table.get_item(Key=key)
    item_before = response_before.get("Item", {})

    # Step 3: Update the item with multiple action types
    # The actions will be processed in this order: REMOVE, SET, ADD, DELETE
    update_response = table.update_item(
        Key=key,
        UpdateExpression="REMOVE to_remove SET to_modify = :new_value ADD
 counter :increment DELETE set_attr :elements",
        ExpressionAttributeValues={
            ":new_value": "Updated value",
            ":increment": 5,
            ":elements": set(["B"]),
        },
        ReturnValues="UPDATED_NEW",
    )

    # Step 4: Get the item to verify final state
    response_after = table.get_item(Key=key)
    item_after = response_after.get("Item", {})

    # Return the results
    return {
        "initial_state": item_before,
        "update_response": update_response,
        "final_state": item_after,
    }


def update_with_multiple_set_actions(
    table_name: str, key: Dict[str, Any], attributes: Dict[str, Any]
```

```
) -> Dict[str, Any]:
    """
    Update multiple attributes with a single SET action.

    This function demonstrates how to update multiple attributes in a single SET
 action,
    which is more efficient than using multiple separate update operations.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        attributes (Dict[str, Any]): The attributes to update and their new
 values.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the update expression and expression attribute values
    update_expression = "SET "
    expression_attribute_values = {}

    # Add each attribute to the update expression
    for i, (attr_name, attr_value) in enumerate(attributes.items()):
        value_placeholder = f":val{i}"

        if i > 0:
            update_expression += ", "
        update_expression += f"{attr_name} = {value_placeholder}"

        expression_attribute_values[value_placeholder] = attr_value

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues="UPDATED_NEW",
    )
```

```
    return response


def update_with_conditional_value_copying(
    table_name: str,
    key: Dict[str, Any],
    source_attribute: str,
    target_attribute: str,
    default_value: Any,
) -> Dict[str, Any]:
    """
    Update an attribute with a value from another attribute or a default value.

    This function demonstrates how to use if_not_exists to conditionally copy a
 value
    from one attribute to another, or use a default value if the source doesn't
 exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        source_attribute (str): The attribute to copy the value from.
        target_attribute (str): The attribute to update.
        default_value (Any): The default value to use if the source attribute
 doesn't exist.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use if_not_exists to conditionally copy the value
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {target_attribute} =
 if_not_exists({source_attribute}, :default)",
        ExpressionAttributeValues={":default": default_value},
        ReturnValues="UPDATED_NEW",
    )

    return response
```

Exemplo de uso da ordem de expressão de atualização com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use update expression order of operations in
 DynamoDB."""
    # Example parameters
    table_name = "OrderProcessing"
    key = {"OrderId": "order123"}

    print("Example 1: Demonstrating value copying in update expressions")
    try:
        results = demonstrate_value_copying(table_name=table_name, key=key)

        print(f"Initial state: {json.dumps(results['initial_state'],
 default=str)}")
        print(f"Update response: {json.dumps(results['update_response'],
 default=str)}")
        print(f"Final state: {json.dumps(results['final_state'], default=str)}")

        print("\nExplanation:")
        print("1. The initial state had a=1, b=2, c=3")
        print("2. The update expression 'REMOVE a SET b = a, c = b' did the
 following:")
        print("   - Copied the value of 'a' (which was 1) to be used for 'b'")
        print("   - Copied the value of 'b' (which was 2) to be used for 'c'")
        print("   - Removed the attribute 'a'")
        print("3. The final state has b=1, c=2, and 'a' is removed")
        print(
            "4. This demonstrates that DynamoDB uses the values of attributes as
 they were BEFORE any modifications"
        )
    except Exception as e:
        print(f"Error demonstrating value copying: {e}")

    print("\nExample 2: Demonstrating the order of different action types")
    try:
        results = demonstrate_action_order(table_name=table_name, key={"OrderId":
 "order456"})
```

```
        print(f"Initial state: {json.dumps(results['initial_state'],
 default=str)}")
        print(f"Update response: {json.dumps(results['update_response'],
 default=str)}")
        print(f"Final state: {json.dumps(results['final_state'], default=str)}")

        print("\nExplanation:")
        print("1. The update expression contained multiple action types: REMOVE,
 SET, ADD, DELETE")
        print("2. DynamoDB processes these actions in this order: REMOVE, SET,
 ADD, DELETE")
        print("3. First, 'to_remove' was removed")
        print("4. Then, 'to_modify' was set to a new value")
        print("5. Next, 'counter' was incremented by 5")
        print("6. Finally, 'B' was removed from the set attribute")
    except Exception as e:
        print(f"Error demonstrating action order: {e}")

    print("\nExample 3: Updating multiple attributes in a single SET action")
    try:
        response = update_with_multiple_set_actions(
            table_name=table_name,
            key={"OrderId": "order789"},
            attributes={
                "Status": "Shipped",
                "ShippingDate": "2025-05-14",
                "TrackingNumber": "1Z999AA10123456784",
            },
        )

        print(
            f"Multiple attributes updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error updating multiple attributes: {e}")

    print("\nExample 4: Conditional value copying with if_not_exists")
    try:
        response = update_with_conditional_value_copying(
            table_name=table_name,
            key={"OrderId": "order101"},
            source_attribute="PreferredShippingMethod",
            target_attribute="ShippingMethod",
```

```
            default_value="Standard",
        )

        print(
            f"Conditional value copying result:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error with conditional value copying: {e}")

    print("\nKey Points About Update Expression Order of Operations:")
    print(
        "1. Variables in expressions hold copies of attribute values as they
existed BEFORE any modifications"
    )
    print(
        "2. Multiple actions in an update expression are processed in this order:
REMOVE, SET, ADD, DELETE"
    )
    print("3. Within each action type, operations are processed from left to
right")
    print("4. You can reference the same attribute multiple times in an
expression")
    print("5. You can use if_not_exists() to conditionally set values based on
attribute existence")
    print(
        "6. Using a single update expression with multiple actions is more
efficient than multiple separate updates"
    )
    print("7. The update expression is atomic - either all actions succeed or
none do")
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Atualize uma configuração de tabela do DynamoDB com taxa de transferência quente usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar a configuração de throughput a quente de uma tabela.

Java

SDK para Java 2.x

Atualize a configuração de throughput a quente em uma tabela existente do DynamoDB usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndexUpdate;
import
 software.amazon.awssdk.services.dynamodb.model.UpdateGlobalSecondaryIndexAction;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long
 readUnitsPerSecond, final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Updates a DynamoDB table with warm throughput settings for both the table
 and a global secondary index.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to update
     * @param tableReadUnitsPerSecond Read units per second for the table
     * @param tableWriteUnitsPerSecond Write units per second for the table
     * @param globalSecondaryIndexName The name of the global secondary index to
 update
     * @param globalSecondaryIndexReadUnitsPerSecond Read units per second for
 the GSI
     * @param globalSecondaryIndexWriteUnitsPerSecond Write units per second for
 the GSI
```

```
     */
    public static void updateDynamoDBTable(
        final DynamoDbClient ddb,
        final String tableName,
        final Long tableReadUnitsPerSecond,
        final Long tableWriteUnitsPerSecond,
        final String globalSecondaryIndexName,
        final Long globalSecondaryIndexReadUnitsPerSecond,
        final Long globalSecondaryIndexWriteUnitsPerSecond) {

        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableReadUnitsPerSecond,
tableWriteUnitsPerSecond);
        final WarmThroughput gsiWarmThroughput =
            buildWarmThroughput(globalSecondaryIndexReadUnitsPerSecond,
globalSecondaryIndexWriteUnitsPerSecond);

        final GlobalSecondaryIndexUpdate globalSecondaryIndexUpdate =
GlobalSecondaryIndexUpdate.builder()
            .update(UpdateGlobalSecondaryIndexAction.builder()
                .indexName(globalSecondaryIndexName)
                .warmThroughput(gsiWarmThroughput)
                .build())
            .build();

        final UpdateTableRequest request = UpdateTableRequest.builder()
            .tableName(tableName)
            .globalSecondaryIndexUpdates(globalSecondaryIndexUpdate)
            .warmThroughput(tableWarmThroughput)
            .build();

        try {
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }

        System.out.println(SUCCESS_MESSAGE);
    }
```

- Para obter detalhes da API, consulte UpdateTablea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Atualize a configuração de throughput a quente em uma tabela existente do DynamoDB usando o AWS SDK para JavaScript.

```
import { DynamoDBClient, UpdateTableCommand } from "@aws-sdk/client-dynamodb";

export async function updateDynamoDBTableWarmThroughput(
  tableName,
  tableReadUnits,
  tableWriteUnits,
  gsiName,
  gsiReadUnits,
  gsiWriteUnits,
  region = "us-east-1"
) {
  try {
    const ddbClient = new DynamoDBClient({ region: region });

    // Construct the update table request
    const updateTableRequest = {
      TableName: tableName,
      GlobalSecondaryIndexUpdates: [
        {
            Update: {
                IndexName: gsiName,
                WarmThroughput: {
                    ReadUnitsPerSecond: gsiReadUnits,
                    WriteUnitsPerSecond: gsiWriteUnits,
                },
            },
        },
      ],
      WarmThroughput: {
          ReadUnitsPerSecond: tableReadUnits,
          WriteUnitsPerSecond: tableWriteUnits,
      },
    };
```

```
    const command = new UpdateTableCommand(updateTableRequest);
    const response = await ddbClient.send(command);
    console.log(`Table updated successfully! Response:
 ${JSON.stringify(response)}`);
    return response;
  } catch (error) {
    console.error(`Error updating table: ${error}`);
    throw error;
  }
}

// Example usage (commented out for testing)
/*
updateDynamoDBTableWarmThroughput(
  'example-table',
  5, 5,
  'example-index',
  2, 2
);
*/
```

- Para obter detalhes da API, consulte UpdateTablea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Atualize a configuração de throughput a quente em uma tabela existente do DynamoDB usando o AWS SDK para Python (Boto3).

```
from boto3 import client
from botocore.exceptions import ClientError


def update_dynamodb_table_warm_throughput(
    table_name,
    table_read_units,
    table_write_units,
```

```
    gsi_name,
    gsi_read_units,
    gsi_write_units,
    region_name="us-east-1",
):
    """
    Updates the warm throughput of a DynamoDB table and a global secondary index.

    :param table_name: The name of the table to update.
    :param table_read_units: The new read units per second for the table's warm
 throughput.
    :param table_write_units: The new write units per second for the table's warm
 throughput.
    :param gsi_name: The name of the global secondary index to update.
    :param gsi_read_units: The new read units per second for the GSI's warm
 throughput.
    :param gsi_write_units: The new write units per second for the GSI's warm
 throughput.
    :param region_name: The AWS Region name to target. defaults to us-east-1
    :return: The response from the update_table operation
    """
    try:
        ddb = client("dynamodb", region_name=region_name)

        # Update the table's warm throughput
        table_warm_throughput = {
            "ReadUnitsPerSecond": table_read_units,
            "WriteUnitsPerSecond": table_write_units,
        }

        # Update the global secondary index's warm throughput
        gsi_warm_throughput = {
            "ReadUnitsPerSecond": gsi_read_units,
            "WriteUnitsPerSecond": gsi_write_units,
        }

        # Construct the global secondary index update
        global_secondary_index_update = [
            {"Update": {"IndexName": gsi_name, "WarmThroughput":
 gsi_warm_throughput}}
        ]

        # Construct the update table request
        update_table_request = {
```

```
            "TableName": table_name,
            "GlobalSecondaryIndexUpdates": global_secondary_index_update,
            "WarmThroughput": table_warm_throughput,
        }

        # Update the table
        response = ddb.update_table(**update_table_request)
        print("Table updated successfully!")
        return response  # Make sure to return the response
    except ClientError as e:
        print(f"Error updating table: {e}")
        raise e
```

- Para obter detalhes da API, consulte a UpdateTableReferência da API AWS SDK for Python (Boto3).

Atualizar um item do DynamoDB com um TTL usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar a TTL de um item.

Java

SDK para Java 2.x

Atualize a TTL em um item do DynamoDB existente em uma tabela.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

    public UpdateItemResponse updateItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
 sortKeyValue) {
        // Get current time in epoch second format
```

```
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = new HashMap<>();
        keyMap.put(PRIMARY_KEY_ATTR,
AttributeValue.builder().s(primaryKeyValue).build());
        keyMap.put(SORT_KEY_ATTR,
AttributeValue.builder().s(sortKeyValue).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":c",
AttributeValue.builder().n(String.valueOf(currentTime)).build());
        expressionAttributeValues.put(
            ":e",
AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final UpdateItemResponse response =
dynamoDbClient.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

```
import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const updateItem = async (tableName, partitionKey, sortKey, region = 'us-
east-1') => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);
    const expireAt = Math.floor((Date.now() + 90 * 24 * 60 * 60 * 1000) / 1000);

    const params = {
        TableName: tableName,
        Key: marshall({
            partitionKey: partitionKey,
            sortKey: sortKey
        }),
        UpdateExpression: "SET updatedAt = :c, expireAt = :e",
        ExpressionAttributeValues: marshall({
            ":c": currentTime,
            ":e": expireAt
        }),
    };

    try {
        const data = await client.send(new UpdateItemCommand(params));
        const responseData = unmarshall(data.Attributes);
        console.log("Item updated successfully: %s", responseData);
        return responseData;
    } catch (err) {
        console.error("Error updating item:", err);
        throw err;
    }
}
```

```
// Example usage (commented out for testing)
// updateItem('your-table-name', 'your-partition-key-value', 'your-sort-key-
value');
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

```
from datetime import datetime, timedelta

import boto3


def update_dynamodb_item(table_name, region, primary_key, sort_key):
    """
    Update an existing DynamoDB item with a TTL.
    :param table_name: Name of the DynamoDB table
    :param region: AWS Region of the table - example `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :return: Void (nothing)
    """
    try:
        # Create the DynamoDB resource.
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Get the current time in epoch second format
        current_time = int(datetime.now().timestamp())

        # Calculate the expireAt time (90 days from now) in epoch second format
        expire_at = int((datetime.now() + timedelta(days=90)).timestamp())

        table.update_item(
            Key={"partitionKey": primary_key, "sortKey": sort_key},
            UpdateExpression="set updatedAt=:c, expireAt=:e",
```

```
            ExpressionAttributeValues={":c": current_time, ":e": expire_at},
        )

        print("Item updated successfully.")
    except Exception as e:
        print(f"Error updating item: {e}")


# Replace with your own values
update_dynamodb_item(
    "your-table-name", "us-west-2", "your-partition-key-value", "your-sort-key-
value"
)
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Atualize os dados do DynamoDB usando instruções PARTIQL UPDATE com um AWS SDK

O exemplo de código a seguir mostra como atualizar dados usando declarações UPDATE do PartiQL.

JavaScript

SDK para JavaScript (v3)

Atualize itens em uma tabela do DynamoDB usando instruções PARTIQL UPDATE com. AWS SDK para JavaScript

```
/**
 * This example demonstrates how to update items in a DynamoDB table using
 PartiQL.
 * It shows different ways to update documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
```

```
} from "@aws-sdk/lib-dynamodb";

/**
 * Update a single attribute of an item using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const updateSingleAttribute = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeName: string,
  attributeValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ?`,
    Parameters: [attributeValue, partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update multiple attributes of an item using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
```

```
 * @param attributeUpdates - Object containing attribute names and their new
 values
 * @returns The response from the ExecuteStatementCommand
 */
export const updateMultipleAttributes = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeUpdates: Record<string, any>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create SET clause for each attribute
  const setClause = Object.keys(attributeUpdates)
    .map((attr, index) => `${attr} = ?`)
    .join(", ");

  // Create parameters array with attribute values followed by the partition key
 value
  const parameters = [...Object.values(attributeUpdates), partitionKeyValue];

  const params = {
    Statement: `UPDATE "${tableName}" SET ${setClause} WHERE ${partitionKeyName}
 = ?`,
    Parameters: parameters,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update an item identified by a composite key (partition key + sort key) using
 PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
```

```
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const updateItemWithCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number,
  attributeName: string,
  attributeValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ? AND ${sortKeyName} = ?`,
    Parameters: [attributeValue, partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update an item with a condition to ensure the update only happens if a
 condition is met.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
```

```
 * @param conditionAttribute - The attribute to check in the condition
 * @param conditionValue - The value to compare against in the condition
 * @returns The response from the ExecuteStatementCommand
 */
export const updateItemWithCondition = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeName: string,
  attributeValue: any,
  conditionAttribute: string,
  conditionValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ? AND ${conditionAttribute} = ?`,
    Parameters: [attributeValue, partitionKeyValue, conditionValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated with condition successfully");
    return data;
  } catch (err) {
    console.error("Error updating item with condition:", err);
    throw err;
  }
};

/**
 * Batch update multiple items using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param updates - Array of objects containing key and update information
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchUpdateItems = async (
  tableName: string,
  updates: Array<{
    partitionKeyName: string;
    partitionKeyValue: string | number;
```

```
    attributeName: string;
    attributeValue: any;
  }>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each update
  const statements = updates.map((update) => {
    return {
      Statement: `UPDATE "${tableName}" SET ${update.attributeName} = ? WHERE
 ${update.partitionKeyName} = ?`,
      Parameters: [update.attributeValue, update.partitionKeyValue],
    };
  });

  const params = {
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items batch updated successfully");
    return data;
  } catch (err) {
    console.error("Error batch updating items:", err);
    throw err;
  }
};

/**
 * Example usage showing how to update items with different index types
 */
export const updateExamples = async () => {
  // Update a single attribute using a simple primary key
  await updateSingleAttribute("UsersTable", "userId", "user123", "email",
 "newemail@example.com");

  // Update multiple attributes at once
  await updateMultipleAttributes("UsersTable", "userId", "user123", {
    email: "newemail@example.com",
    name: "John Smith",
    lastLogin: new Date().toISOString(),
  });
```

```
  // Update an item with a composite key (partition key + sort key)
  await updateItemWithCompositeKey(
    "OrdersTable",
    "orderId",
    "order456",
    "productId",
    "prod789",
    "quantity",
    5
  );

  // Update with a condition
  await updateItemWithCondition(
    "UsersTable",
    "userId",
    "user123",
    "userStatus",
    "active",
    "userType",
    "premium"
  );

  // Batch update multiple items
  await batchUpdateItems("UsersTable", [
    {
      partitionKeyName: "userId",
      partitionKeyValue: "user123",
      attributeName: "lastLogin",
      attributeValue: new Date().toISOString(),
    },
    {
      partitionKeyName: "userId",
      partitionKeyValue: "user456",
      attributeName: "lastLogin",
      attributeValue: new Date().toISOString(),
    },
  ]);
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .

- BatchExecuteStatement
- ExecuteStatement

Usar o API Gateway para invocar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Lambda Java. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma

mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3).

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.
- Use o pacote Requests para enviar solicitações à API REST.
- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

- Amazon SNS

Usar Step Functions para invocar funções do Lambda

O exemplo de código a seguir mostra como criar uma máquina de AWS Step Functions estado que invoca AWS Lambda funções em sequência.

Java

SDK para Java 2.x

Mostra como criar um fluxo de trabalho AWS sem servidor usando AWS Step Functions e. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Use um modelo de documento para o DynamoDB usando um AWS SDK

O exemplo de código a seguir mostra como realizar operações de criação, leitura, atualização e exclusão (CRUD) e em lote usando um modelo de documento para o DynamoDB e um SDK. AWS

Para obter mais informações, consulte o modelo de documento.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute operações CRUD usando um modelo de documento.

```
/// <summary>
/// Performs CRUD operations on an Amazon DynamoDB table.
/// </summary>
public class MidlevelItemCRUD
{
    public static async Task Main()
    {
        var tableName = "ProductCatalog";
        var sampleBookId = 555;

        var client = new AmazonDynamoDBClient();
        var productCatalog = LoadTable(client, tableName);

        await CreateBookItem(productCatalog, sampleBookId);
        RetrieveBook(productCatalog, sampleBookId);

        // Couple of sample updates.
        UpdateMultipleAttributes(productCatalog, sampleBookId);
        UpdateBookPriceConditionally(productCatalog, sampleBookId);

        // Delete.
        await DeleteBook(productCatalog, sampleBookId);
    }

    /// <summary>
    /// Loads the contents of a DynamoDB table.
    /// </summary>
    /// <param name="client">An initialized DynamoDB client object.</param>
    /// <param name="tableName">The name of the table to load.</param>
    /// <returns>A DynamoDB table object.</returns>
    public static Table LoadTable(IAmazonDynamoDB client, string tableName)
    {
        Table productCatalog = Table.LoadTable(client, tableName);
        return productCatalog;
    }

    /// <summary>
    /// Creates an example book item and adds it to the DynamoDB table
    /// ProductCatalog.
    /// </summary>
    /// <param name="productCatalog">A DynamoDB table object.</param>
```

```
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async Task CreateBookItem(Table productCatalog, int
sampleBookId)
        {
            Console.WriteLine("\n*** Executing CreateBookItem() ***");
            var book = new Document
            {
                ["Id"] = sampleBookId,
                ["Title"] = "Book " + sampleBookId,
                ["Price"] = 19.99,
                ["ISBN"] = "111-1111111111",
                ["Authors"] = new List<string> { "Author 1", "Author 2", "Author
3" },
                ["PageCount"] = 500,
                ["Dimensions"] = "8.5x11x.5",
                ["InPublication"] = new DynamoDBBool(true),
                ["InStock"] = new DynamoDBBool(false),
                ["QuantityOnHand"] = 0,
            };

            // Adds the book to the ProductCatalog table.
            await productCatalog.PutItemAsync(book);
        }

        /// <summary>
        /// Retrieves an item, a book, from the DynamoDB ProductCatalog table.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void RetrieveBook(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing RetrieveBook() ***");

            // Optional configuration.
            var config = new GetItemOperationConfig
            {
                AttributesToGet = new List<string> { "Id", "ISBN", "Title",
"Authors", "Price" },
                ConsistentRead = true,
            };
```

```
            Document document = await productCatalog.GetItemAsync(sampleBookId,
config);
            Console.WriteLine("RetrieveBook: Printing book retrieved...");
            PrintDocument(document);
        }

        /// <summary>
        /// Updates multiple attributes for a book and writes the changes to the
        /// DynamoDB table ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void UpdateMultipleAttributes(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\nUpdating multiple attributes....");
            int partitionKey = sampleBookId;

            var book = new Document
            {
                ["Id"] = partitionKey,

                // List of attribute updates.
                // The following replaces the existing authors list.
                ["Authors"] = new List<string> { "Author x", "Author y" },
                ["newAttribute"] = "New Value",
                ["ISBN"] = null, // Remove it.
            };

            // Optional parameters.
            var config = new UpdateItemOperationConfig
            {
                // Gets updated item in response.
                ReturnValues = ReturnValues.AllNewAttributes,
            };

            Document updatedBook = await productCatalog.UpdateItemAsync(book,
config);
            Console.WriteLine("UpdateMultipleAttributes: Printing item after
updates ...");
            PrintDocument(updatedBook);
```

```
        }

        /// <summary>
        /// Updates a book item if it meets the specified criteria.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void UpdateBookPriceConditionally(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing UpdateBookPriceConditionally()
***");

            int partitionKey = sampleBookId;

            var book = new Document
            {
                ["Id"] = partitionKey,
                ["Price"] = 29.99,
            };

            // For conditional price update, creating a condition expression.
            var expr = new Expression
            {
                ExpressionStatement = "Price = :val",
            };
            expr.ExpressionAttributeValues[":val"] = 19.00;

            // Optional parameters.
            var config = new UpdateItemOperationConfig
            {
                ConditionalExpression = expr,
                ReturnValues = ReturnValues.AllNewAttributes,
            };

            Document updatedBook = await productCatalog.UpdateItemAsync(book,
config);
            Console.WriteLine("UpdateBookPriceConditionally: Printing item whose
price was conditionally updated");
            PrintDocument(updatedBook);
        }
```

```
        /// <summary>
        /// Deletes the book with the supplied Id value from the DynamoDB table
        /// ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async Task DeleteBook(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing DeleteBook() ***");

            // Optional configuration.
            var config = new DeleteItemOperationConfig
            {
                // Returns the deleted item.
                ReturnValues = ReturnValues.AllOldAttributes,
            };
            Document document = await
productCatalog.DeleteItemAsync(sampleBookId, config);
            Console.WriteLine("DeleteBook: Printing deleted just deleted...");

            PrintDocument(document);
        }

        /// <summary>
        /// Prints the information for the supplied DynamoDB document.
        /// </summary>
        /// <param name="updatedDocument">A DynamoDB document object.</param>
        public static void PrintDocument(Document updatedDocument)
        {
            if (updatedDocument is null)
            {
                return;
            }

            foreach (var attribute in updatedDocument.GetAttributeNames())
            {
                string stringValue = null;
                var value = updatedDocument[attribute];

                if (value is null)
                {
```

```
                    continue;
                }

                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                        select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}", attribute,
stringValue);
            }
        }
    }
```

Execute operações de gravação em lote usando um modelo de documento.

```
    /// <summary>
    /// Shows how to use mid-level Amazon DynamoDB API calls to perform batch
    /// operations.
    /// </summary>
    public class MidLevelBatchWriteItem
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            await SingleTableBatchWrite(client);
            await MultiTableBatchWrite(client);
        }

        /// <summary>
        /// Perform a batch operation on a single DynamoDB table.
        /// </summary>
```

```
        /// <param name="client">An initialized DynamoDB object.</param>
        public static async Task SingleTableBatchWrite(IAmazonDynamoDB client)
        {
            Table productCatalog = Table.LoadTable(client, "ProductCatalog");
            var batchWrite = productCatalog.CreateBatchWrite();

            var book1 = new Document
            {
                ["Id"] = 902,
                ["Title"] = "My book1 in batch write using .NET helper classes",
                ["ISBN"] = "902-11-11-1111",
                ["Price"] = 10,
                ["ProductCategory"] = "Book",
                ["Authors"] = new List<string> { "Author 1", "Author 2", "Author
3" },
                ["Dimensions"] = "8.5x11x.5",
                ["InStock"] = new DynamoDBBool(true),
                ["QuantityOnHand"] = new DynamoDBNull(), // Quantity is unknown
at this time.
            };

            batchWrite.AddDocumentToPut(book1);

            // Specify delete item using overload that takes PK.
            batchWrite.AddKeyToDelete(12345);
            Console.WriteLine("Performing batch write in
SingleTableBatchWrite()");
            await batchWrite.ExecuteAsync();
        }

        /// <summary>
        /// Perform a batch operation involving multiple DynamoDB tables.
        /// </summary>
        /// <param name="client">An initialized DynamoDB client object.</param>
        public static async Task MultiTableBatchWrite(IAmazonDynamoDB client)
        {
            // Specify item to add in the Forum table.
            Table forum = Table.LoadTable(client, "Forum");
            var forumBatchWrite = forum.CreateBatchWrite();

            var forum1 = new Document
            {
                ["Name"] = "Test BatchWrite Forum",
                ["Threads"] = 0,
```

```
            };
            forumBatchWrite.AddDocumentToPut(forum1);

            // Specify item to add in the Thread table.
            Table thread = Table.LoadTable(client, "Thread");
            var threadBatchWrite = thread.CreateBatchWrite();

            var thread1 = new Document
            {
                ["ForumName"] = "S3 forum",
                ["Subject"] = "My sample question",
                ["Message"] = "Message text",
                ["KeywordTags"] = new List<string> { "S3", "Bucket" },
            };
            threadBatchWrite.AddDocumentToPut(thread1);

            // Specify item to delete from the Thread table.
            threadBatchWrite.AddKeyToDelete("someForumName", "someSubject");

            // Create multi-table batch.
            var superBatch = new MultiTableDocumentBatchWrite();
            superBatch.AddBatch(forumBatchWrite);
            superBatch.AddBatch(threadBatchWrite);
            Console.WriteLine("Performing batch write in
 MultiTableBatchWrite()");

            // Execute the batch.
            await superBatch.ExecuteAsync();
        }
    }
```

Verifique uma tabela usando um modelo de documento.

```
    /// <summary>
    /// Shows how to use mid-level Amazon DynamoDB API calls to scan a DynamoDB
    /// table for values.
    /// </summary>
    public class MidLevelScanOnly
    {
        public static async Task Main()
```

```
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            Table productCatalogTable = Table.LoadTable(client,
"ProductCatalog");

            await FindProductsWithNegativePrice(productCatalogTable);
            await FindProductsWithNegativePriceWithConfig(productCatalogTable);
        }

        /// <summary>
        /// Retrieves any products that have a negative price in a DynamoDB
table.
        /// </summary>
        /// <param name="productCatalogTable">A DynamoDB table object.</param>
        public static async Task FindProductsWithNegativePrice(
          Table productCatalogTable)
        {
            // Assume there is a price error. So we scan to find items priced <
0.
            var scanFilter = new ScanFilter();
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

            Search search = productCatalogTable.Scan(scanFilter);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindProductsWithNegativePrice:
printing ............");

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Finds any items in the ProductCatalog table using a DynamoDB
        /// configuration object.
        /// </summary>
        /// <param name="productCatalogTable">A DynamoDB table object.</param>
```

```
        public static async Task FindProductsWithNegativePriceWithConfig(
          Table productCatalogTable)
        {
            // Assume there is a price error. So we scan to find items priced <
0.
            var scanFilter = new ScanFilter();
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

            var config = new ScanOperationConfig()
            {
                Filter = scanFilter,
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string> { "Title", "Id" },
            };

            Search search = productCatalogTable.Scan(config);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindProductsWithNegativePriceWithConfig:
printing ............");

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Displays the details of the passed DynamoDB document object on the
        /// console.
        /// </summary>
        /// <param name="document">A DynamoDB document object.</param>
        public static void PrintDocument(Document document)
        {
            Console.WriteLine();
            foreach (var attribute in document.GetAttributeNames())
            {
                string stringValue = null;
                var value = document[attribute];
                if (value is Primitive)
```

```
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                        select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}");
            }
        }
    }
```

Consulte e verifique uma tabela usando um modelo de documento.

```
    /// <summary>
    /// Shows how to perform mid-level query procedures on an Amazon DynamoDB
    /// table.
    /// </summary>
    public class MidLevelQueryAndScan
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            // Query examples.
            Table replyTable = Table.LoadTable(client, "Reply");
            string forumName = "Amazon DynamoDB";
            string threadSubject = "DynamoDB Thread 2";

            await FindRepliesInLast15Days(replyTable);
            await FindRepliesInLast15DaysWithConfig(replyTable, forumName,
threadSubject);
            await FindRepliesPostedWithinTimePeriod(replyTable, forumName,
threadSubject);

            // Get Example.
```

```
            Table productCatalogTable = Table.LoadTable(client,
"ProductCatalog");
            int productId = 101;

            await GetProduct(productCatalogTable, productId);
        }

        /// <summary>
        /// Retrieves information about a product from the DynamoDB table
        /// ProductCatalog based on the product ID and displays the information
        /// on the console.
        /// </summary>
        /// <param name="tableName">The name of the table from which to retrieve
        /// product information.</param>
        /// <param name="productId">The ID of the product to retrieve.</param>
        public static async Task GetProduct(Table tableName, int productId)
        {
            Console.WriteLine("*** Executing GetProduct() ***");
            Document productDocument = await tableName.GetItemAsync(productId);
            if (productDocument != null)
            {
                PrintDocument(productDocument);
            }
            else
            {
                Console.WriteLine("Error: product " + productId + " does not
exist");
            }
        }

        /// <summary>
        /// Retrieves replies from the passed DynamoDB table object.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        public static async Task FindRepliesInLast15Days(
          Table table)
        {
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
            var filter = new QueryFilter("Id", QueryOperator.Equal, "Id");
            filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);

            // Use Query overloads that take the minimum required query
parameters.
```

```
            Search search = table.Query(filter);

            do
            {
                var documentSet = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesInLast15Days:
printing ............");

                foreach (var document in documentSet)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Retrieve replies made during a specific time period.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadSubject">The subject of the thread, which we are
        /// searching for replies.</param>
        public static async Task FindRepliesPostedWithinTimePeriod(
          Table table,
          string forumName,
          string threadSubject)
        {
            DateTime startDate = DateTime.UtcNow.Subtract(new TimeSpan(21, 0, 0,
0));
            DateTime endDate = DateTime.UtcNow.Subtract(new TimeSpan(1, 0, 0,
0));

            var filter = new QueryFilter("Id", QueryOperator.Equal, forumName +
"#" + threadSubject);
            filter.AddCondition("ReplyDateTime", QueryOperator.Between,
startDate, endDate);

            var config = new QueryOperationConfig()
            {
                Limit = 2, // 2 items/page.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
```

```
        {
          "Message",
          "ReplyDateTime",
          "PostedBy",
        },
                ConsistentRead = true,
                Filter = filter,
            };

            Search search = table.Query(config);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesPostedWithinTimePeriod: printing
replies posted within dates: {0} and {1} ............", startDate, endDate);

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Perform a query for replies made in the last 15 days using a DynamoDB
        /// QueryOperationConfig object.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadName">The bane of the thread that we are searching
        /// for replies.</param>
        public static async Task FindRepliesInLast15DaysWithConfig(
          Table table,
          string forumName,
          string threadName)
        {
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
            var filter = new QueryFilter("Id", QueryOperator.Equal, forumName +
"#" + threadName);
            filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);
```

```
            var config = new QueryOperationConfig()
            {
                Filter = filter,

                // Optional parameters.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
                {
                  "Message",
                  "ReplyDateTime",
                  "PostedBy",
                },
                ConsistentRead = true,
            };

            Search search = table.Query(config);

            do
            {
                var documentSet = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesInLast15DaysWithConfig:
printing ............");

                foreach (var document in documentSet)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Displays the contents of the passed DynamoDB document on the console.
        /// </summary>
        /// <param name="document">A DynamoDB document to display.</param>
        public static void PrintDocument(Document document)
        {
            Console.WriteLine();
            foreach (var attribute in document.GetAttributeNames())
            {
                string stringValue = null;
                var value = document[attribute];
```

```
                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                    select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}");
            }
        }
    }
```

Use um modelo de persistência de objetos de alto nível para o DynamoDB usando um AWS SDK

O exemplo de código a seguir mostra como realizar operações de criação, leitura, atualização e exclusão (CRUD) e em lote usando um modelo de persistência de objetos para o DynamoDB e um SDK. AWS

Para obter mais informações, consulte Modelo de persistência de objetos.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Realize operações CRUD usando um modelo de persistência de objetos de alto nível.

```
    /// <summary>
    /// Shows how to perform high-level CRUD operations on an Amazon DynamoDB
    /// table.
    /// </summary>
    public class HighLevelItemCrud
    {
        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);
            await PerformCRUDOperations(context);
        }

        public static async Task PerformCRUDOperations(IDynamoDBContext context)
        {
            int bookId = 1001; // Some unique value.
            Book myBook = new Book
            {
                Id = bookId,
                Title = "object persistence-AWS SDK for.NET SDK-Book 1001",
                Isbn = "111-1111111001",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
            };

            // Save the book to the ProductCatalog table.
            await context.SaveAsync(myBook);

            // Retrieve the book from the ProductCatalog table.
            Book bookRetrieved = await context.LoadAsync<Book>(bookId);

            // Update some properties.
            bookRetrieved.Isbn = "222-2222221001";

            // Update existing authors list with the following values.
            bookRetrieved.BookAuthors = new List<string> { " Author 1", "Author
x" };
            await context.SaveAsync(bookRetrieved);

            // Retrieve the updated book. This time, add the optional
            // ConsistentRead parameter using DynamoDBContextConfig object.
            await context.LoadAsync<Book>(bookId, new DynamoDBContextConfig
            {
                ConsistentRead = true,
            });
```

```
            // Delete the book.
            await context.DeleteAsync<Book>(bookId);

            // Try to retrieve deleted book. It should return null.
            Book deletedBook = await context.LoadAsync<Book>(bookId, new
DynamoDBContextConfig
            {
                ConsistentRead = true,
            });

            if (deletedBook == null)
            {
                Console.WriteLine("Book is deleted");
            }
        }
    }
```

Realize operações de gravação em lote usando um modelo de persistência de objetos de alto nível.

```
    /// <summary>
    /// Performs high-level batch write operations to an Amazon DynamoDB table.
    /// This example was written using the AWS SDK for .NET version 3.7 and .NET
    /// Core 5.0.
    /// </summary>
    public class HighLevelBatchWriteItem
    {
        public static async Task SingleTableBatchWrite(IDynamoDBContext context)
        {
            Book book1 = new Book
            {
                Id = 902,
                InPublication = true,
                Isbn = "902-11-11-1111",
                PageCount = "100",
                Price = 10,
                ProductCategory = "Book",
                Title = "My book3 in batch write",
            };
```

```
            Book book2 = new Book
            {
                Id = 903,
                InPublication = true,
                Isbn = "903-11-11-1111",
                PageCount = "200",
                Price = 10,
                ProductCategory = "Book",
                Title = "My book4 in batch write",
            };

            var bookBatch = context.CreateBatchWrite<Book>();
            bookBatch.AddPutItems(new List<Book> { book1, book2 });

            Console.WriteLine("Adding two books to ProductCatalog table.");
            await bookBatch.ExecuteAsync();
        }

        public static async Task MultiTableBatchWrite(IDynamoDBContext context)
        {
            // New Forum item.
            Forum newForum = new Forum
            {
                Name = "Test BatchWrite Forum",
                Threads = 0,
            };
            var forumBatch = context.CreateBatchWrite<Forum>();
            forumBatch.AddPutItem(newForum);

            // New Thread item.
            Thread newThread = new Thread
            {
                ForumName = "S3 forum",
                Subject = "My sample question",
                KeywordTags = new List<string> { "S3", "Bucket" },
                Message = "Message text",
            };

            DynamoDBOperationConfig config = new DynamoDBOperationConfig();
            config.SkipVersionCheck = true;
            var threadBatch = context.CreateBatchWrite<Thread>(config);
            threadBatch.AddPutItem(newThread);
```

```
            threadBatch.AddDeleteKey("some partition key value", "some sort key
value");

            var superBatch = new MultiTableBatchWrite(forumBatch, threadBatch);

            Console.WriteLine("Performing batch write in
MultiTableBatchWrite().");
            await superBatch.ExecuteAsync();
        }

        public static async Task Main()
        {
            AmazonDynamoDBClient client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);

            await SingleTableBatchWrite(context);
            await MultiTableBatchWrite(context);
        }
    }
```

Mapeie dados arbitrários em uma tabela usando um modelo de persistência de objetos de alto nível.

```
    /// <summary>
    /// Shows how to map arbitrary data to an Amazon DynamoDB table.
    /// </summary>
    public class HighLevelMappingArbitraryData
    {
        /// <summary>
        /// Creates a book, adds it to the DynamoDB ProductCatalog table,
retrieves
        /// the new book from the table, updates the dimensions and writes the
        /// changed item back to the table.
        /// </summary>
        /// <param name="context">The DynamoDB context object used to write and
        /// read data from the table.</param>
        public static async Task AddRetrieveUpdateBook(IDynamoDBContext context)
        {
            // Create a book.
            DimensionType myBookDimensions = new DimensionType()
```

```
            {
                Length = 8M,
                Height = 11M,
                Thickness = 0.5M,
            };

            Book myBook = new Book
            {
                Id = 501,
                Title = "AWS SDK for .NET Object Persistence Model Handling
 Arbitrary Data",
                Isbn = "999-9999999999",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
                Dimensions = myBookDimensions,
            };

            // Add the book to the DynamoDB table ProductCatalog.
            await context.SaveAsync(myBook);

            // Retrieve the book.
            Book bookRetrieved = await context.LoadAsync<Book>(501);

            // Update the book dimensions property.
            bookRetrieved.Dimensions.Height += 1;
            bookRetrieved.Dimensions.Length += 1;
            bookRetrieved.Dimensions.Thickness += 0.2M;

            // Write the changed item to the table.
            await context.SaveAsync(bookRetrieved);
        }

        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);
            await AddRetrieveUpdateBook(context);
        }
    }
```

Consulte e verifique uma tabela usando um modelo de persistência de objetos de alto nível.

```
    /// <summary>
    /// Shows how to perform high-level query and scan operations to Amazon
    /// DynamoDB tables.
    /// </summary>
    public class HighLevelQueryAndScan
    {
        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();

            DynamoDBContext context = new DynamoDBContext(client);

            // Get an item.
            await GetBook(context, 101);

            // Sample forum and thread to test queries.
            string forumName = "Amazon DynamoDB";
            string threadSubject = "DynamoDB Thread 1";

            // Sample queries.
            await FindRepliesInLast15Days(context, forumName, threadSubject);
            await FindRepliesPostedWithinTimePeriod(context, forumName,
threadSubject);

            // Scan table.
            await FindProductsPricedLessThanZero(context);
        }

        public static async Task GetBook(IDynamoDBContext context, int productId)
        {
            Book bookItem = await context.LoadAsync<Book>(productId);

            Console.WriteLine("\nGetBook: Printing result.....");
            Console.WriteLine($"Title: {bookItem.Title} \n ISBN:{bookItem.Isbn}
\n No. of pages: {bookItem.PageCount}");
        }

        /// <summary>
        /// Queries a DynamoDB table to find replies posted within the last 15
days.
        /// </summary>
```

```
        /// <param name="context">The DynamoDB context used to perform the
query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadSubject">The thread object containing the query
parameters.</param>
        public static async Task FindRepliesInLast15Days(
          IDynamoDBContext context,
          string forumName,
          string threadSubject)
        {
            string replyId = $"{forumName} #{threadSubject}";
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);

            List<object> times = new List<object>();
            times.Add(twoWeeksAgoDate);

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc = new ScanCondition("PostedBy", ScanOperator.GreaterThan,
times.ToArray());
            scs.Add(sc);

            var cfg = new DynamoDBOperationConfig
            {
                QueryFilter = scs,
            };

            AsyncSearch<Reply> response = context.QueryAsync<Reply>(replyId,
cfg);
            IEnumerable<Reply> latestReplies = await
response.GetRemainingAsync();

            Console.WriteLine("\nReplies in last 15 days:");

            foreach (Reply r in latestReplies)
            {

Console.WriteLine($"{r.Id}\t{r.PostedBy}\t{r.Message}\t{r.ReplyDateTime}");
            }
        }

        /// <summary>
        /// Queries for replies posted within a specific time period.
        /// </summary>
```

```
        /// <param name="context">The DynamoDB context used to perform the
query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadSubject">Information about the subject that we're
        /// interested in.</param>
        public static async Task FindRepliesPostedWithinTimePeriod(
          IDynamoDBContext context,
          string forumName,
          string threadSubject)
        {
            string forumId = forumName + "#" + threadSubject;
            Console.WriteLine("\nReplies posted within time period:");

            DateTime startDate = DateTime.UtcNow - TimeSpan.FromDays(30);
            DateTime endDate = DateTime.UtcNow - TimeSpan.FromDays(1);

            List<object> times = new List<object>();
            times.Add(startDate);
            times.Add(endDate);

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc = new ScanCondition("LastPostedBy", ScanOperator.Between,
times.ToArray());
            scs.Add(sc);

            var cfg = new DynamoDBOperationConfig
            {
                QueryFilter = scs,
            };

            AsyncSearch<Reply> response = context.QueryAsync<Reply>(forumId,
cfg);
            IEnumerable<Reply> repliesInAPeriod = await
response.GetRemainingAsync();

            foreach (Reply r in repliesInAPeriod)
            {

Console.WriteLine("{r.Id}\t{r.PostedBy}\t{r.Message}\t{r.ReplyDateTime}");
            }
        }

        /// <summary>
```

```
        /// Queries the DynamoDB ProductCatalog table for products costing less
        /// than zero.
        /// </summary>
        /// <param name="context">The DynamoDB context object used to perform the
        /// query.</param>
        public static async Task FindProductsPricedLessThanZero(IDynamoDBContext
context)
        {
            int price = 0;

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc1 = new ScanCondition("Price", ScanOperator.LessThan, price);
            var sc2 = new ScanCondition("ProductCategory", ScanOperator.Equal,
"Book");
            scs.Add(sc1);
            scs.Add(sc2);

            AsyncSearch<Book> response = context.ScanAsync<Book>(scs);

            IEnumerable<Book> itemsWithWrongPrice = await
response.GetRemainingAsync();

            Console.WriteLine("\nFindProductsPricedLessThanZero: Printing
result.....");

            foreach (Book r in itemsWithWrongPrice)
            {
                Console.WriteLine($"{r.Id}\t{r.Title}\t{r.Price}\t{r.Isbn}");
            }
        }
    }
```

Use operações de contador atômico no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como usar operações de contador atômico no DynamoDB.

- Incremente os contadores atomicamente usando as operações ADD e SET.
- Incremente seguramente contadores que talvez não existam.
- Implemente um bloqueio positivo para operações de contador.

Java

SDK para Java 2.x

Demonstre as operações do contador atômico usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Increments a counter using the ADD operation.
     *
     * <p>This method demonstrates how to use the ADD operation to atomically
     * increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithAdd(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #counterName :increment")
```

```
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter using the SET operation.
     *
     * <p>This method demonstrates how to use the SET operation with an
expression
     * to increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
```

```
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter safely, handling the case where the counter doesn't
exist yet.
     *
     * <p>This method demonstrates how to use if_not_exists to safely increment a
counter
     * that may not exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName =
if_not_exists(#counterName, :zero) + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build(),
                ":zero", AttributeValue.builder().n("0").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();
```

```
        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Decrements a counter safely, ensuring it doesn't go below zero.
     *
     * <p>This method demonstrates how to use a condition expression to safely
     * decrement a counter without going below zero.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param decrementValue The value to decrement by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation or if
the counter would go below zero
     */
    public static UpdateItemResponse decrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int decrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName - :decrement")
            .conditionExpression("#counterName >= :decrement")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":decrement",

AttributeValue.builder().n(String.valueOf(decrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }
```

```
    /**
     * Compares the ADD and SET approaches for incrementing counters.
     *
     * <p>This method demonstrates the differences between using ADD and SET
     * for incrementing counters in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the comparison results
     */
    public static Map<String, Object> compareAddVsSet(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Reset counters to ensure a fair comparison
            UpdateItemRequest resetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET AddCounter = :zero, SetCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .build();

            dynamoDbClient.updateItem(resetRequest);

            // Increment with ADD
            long addStartTime = System.nanoTime();
            UpdateItemResponse addResponse =
incrementCounterWithAdd(dynamoDbClient, tableName, key, "AddCounter", 1);
            long addEndTime = System.nanoTime();
            long addDuration = addEndTime - addStartTime;

            // Increment with SET
            long setStartTime = System.nanoTime();
            UpdateItemResponse setResponse =
incrementCounterWithSet(dynamoDbClient, tableName, key, "SetCounter", 1);
            long setEndTime = System.nanoTime();
            long setDuration = setEndTime - setStartTime;
```

```
            // Record results
            results.put("addResponse", addResponse);
            results.put("setResponse", setResponse);
            results.put("addDuration", addDuration);
            results.put("setDuration", setDuration);
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Gets the current value of a counter attribute.
     *
     * <p>Helper method to retrieve the current value of a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param counterName The name of the counter attribute
     * @return The counter value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Integer getCounterValue(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String counterName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(counterName)
            .build();

        // Perform the get operation
        GetItemResponse response = dynamoDbClient.getItem(request);

        // Return the counter value if it exists, otherwise null
        if (response.item() != null && response.item().containsKey(counterName))
{
```

```
            return Integer.parseInt(response.item().get(counterName).n());
        }

        return null;
    }
```

Exemplo de uso de operações de contador atômico com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating atomic counter operations in
 DynamoDB");

        try {
            // Example 1: Increment a counter using ADD
            System.out.println("\nExample 1: Incrementing a counter using ADD");
            UpdateItemResponse addResponse =
 incrementCounterWithAdd(dynamoDbClient, tableName, key, "ViewCount", 1);

            System.out.println("Updated counter: " + addResponse.attributes());

            // Example 2: Increment a counter using SET
            System.out.println("\nExample 2: Incrementing a counter using SET");
            UpdateItemResponse setResponse =
 incrementCounterWithSet(dynamoDbClient, tableName, key, "LikeCount", 1);

            System.out.println("Updated counter: " + setResponse.attributes());

            // Example 3: Increment a counter safely
            System.out.println("\nExample 3: Incrementing a counter safely");
            UpdateItemResponse safeResponse =
 incrementCounterSafely(dynamoDbClient, tableName, key, "ShareCount", 1);

            System.out.println("Updated counter: " + safeResponse.attributes());

            // Example 4: Decrement a counter safely
            System.out.println("\nExample 4: Decrementing a counter safely");
            try {
```

```
                UpdateItemResponse decrementResponse =
                    decrementCounterSafely(dynamoDbClient, tableName, key,
"InventoryCount", 1);

                System.out.println("Updated counter: " +
decrementResponse.attributes());
            } catch (DynamoDbException e) {
                if (e.getMessage().contains("ConditionalCheckFailed")) {
                    System.out.println("Cannot decrement counter below zero");
                } else {
                    throw e;
                }
            }

            // Example 5: Compare ADD vs SET
            System.out.println("\nExample 5: Comparing ADD vs SET");
            Map<String, Object> comparison = compareAddVsSet(dynamoDbClient,
tableName, key);

            if ((boolean) comparison.get("success")) {
                System.out.println("ADD duration: " +
comparison.get("addDuration") + " ns");
                System.out.println("SET duration: " +
comparison.get("setDuration") + " ns");
                System.out.println("ADD response: " +
comparison.get("addResponse"));
                System.out.println("SET response: " +
comparison.get("setResponse"));
            } else {
                System.out.println("Comparison failed: " +
comparison.get("error"));
            }

            // Explain atomic counter operations
            System.out.println("\nKey points about DynamoDB atomic counter
operations:");
            System.out.println("1. Both ADD and SET can be used for atomic
counters");
            System.out.println("2. ADD is more concise for simple increments");
            System.out.println("3. SET with an expression is more flexible for
complex operations");
            System.out.println("4. Use if_not_exists to handle the case where the
counter doesn't exist yet");
```

```
            System.out.println("5. Use condition expressions to prevent counters
 from going below zero");
            System.out.println("6. Atomic operations are guaranteed to be
 isolated from other writes");
            System.out.println("7. ADD can only be used with number and set data
 types");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

Demonstre as operações do contador atômico usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Increment a counter using the ADD operation.
 *
 * This function demonstrates using the ADD operation for atomic increments.
 * The ADD operation is atomic and is the recommended way to increment counters.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @returns {Promise<Object>} - The response from DynamoDB
 */
```

```
async function incrementCounterWithAdd(
  config,
  tableName,
  key,
  counterName,
  incrementValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using ADD
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `ADD ${counterName} :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter using the SET operation with an expression.
 *
 * This function demonstrates using the SET operation with an expression for
 increments.
 * While this approach works, it's less idiomatic for simple increments than
 using ADD.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterWithSet(
```

```
  config,
  tableName,
  key,
  counterName,
  incrementValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with an expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = ${counterName} + :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter safely, handling the case where the counter might not
 exist.
 *
 * This function demonstrates using the if_not_exists function with SET to safely
 * increment a counter that might not exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @param {number} defaultValue - The default value if the counter doesn't exist
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterSafely(
  config,
```

```
  tableName,
  key,
  counterName,
  incrementValue,
  defaultValue = 0
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with if_not_exists
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} =
 if_not_exists(${counterName}, :default) + :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue,
      ":default": defaultValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter with optimistic locking to prevent race conditions.
 *
 * This function demonstrates using a condition expression to implement
 optimistic
 * locking, which prevents race conditions when multiple processes try to update
 * the same counter.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @param {number} expectedValue - The expected current value of the counter
 * @returns {Promise<Object>} - The response from DynamoDB
```

```
 */
async function incrementCounterWithLocking(
  config,
  tableName,
  key,
  counterName,
  incrementValue,
  expectedValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = ${counterName} + :increment`,
    ConditionExpression: `${counterName} = :expected`,
    ExpressionAttributeValues: {
      ":increment": incrementValue,
      ":expected": expectedValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return {
      success: true,
      data: response
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        error: "Optimistic locking failed: the counter value has changed"
      };
    }
    // Re-throw other errors
    throw error;
  }
```

```
}

/**
 * Get the current value of a counter.
 *
 * Helper function to retrieve the current value of a counter attribute.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @param {string} counterName - The name of the counter attribute
 * @returns {Promise<number|null>} - The current counter value or null if not
 found
 */
async function getCounterValue(
  config,
  tableName,
  key,
  counterName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the counter value if it exists, otherwise null
  return response.Item && counterName in response.Item
    ? response.Item[counterName]
    : null;
}
```

Exemplo de uso de operações de contador atômico com AWS SDK para JavaScript.

```
/**
```

```
 * Example of how to use the atomic counter operations.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };
  const counterName = "ViewCount";
  const incrementValue = 1;

  console.log("Demonstrating different approaches to increment counters in
 DynamoDB");

  try {
    // Example 1: Using ADD operation (recommended for simple increments)
    console.log("\nExample 1: Incrementing counter with ADD operation");
    const response1 = await incrementCounterWithAdd(
      config,
      tableName,
      key,
      counterName,
      incrementValue
    );

    console.log(`Counter incremented to: ${response1.Attributes[counterName]}`);

    // Example 2: Using SET operation with an expression
    console.log("\nExample 2: Incrementing counter with SET operation");
    const response2 = await incrementCounterWithSet(
      config,
      tableName,
      key,
      counterName,
      incrementValue
    );

    console.log(`Counter incremented to: ${response2.Attributes[counterName]}`);

    // Example 3: Safely incrementing a counter that might not exist
    console.log("\nExample 3: Safely incrementing counter that might not exist");
    const newKey = { ProductId: "P67890" };
    const response3 = await incrementCounterSafely(
      config,
      tableName,
```

```
      newKey,
      counterName,
      incrementValue,
      0
    );

    console.log(`Counter initialized and incremented to:
${response3.Attributes[counterName]}`);

    // Example 4: Incrementing with optimistic locking
    console.log("\nExample 4: Incrementing with optimistic locking");

    // First, get the current counter value
    const currentValue = await getCounterValue(config, tableName, key,
counterName);
    console.log(`Current counter value: ${currentValue}`);

    // Then, try to increment with optimistic locking
    const response4 = await incrementCounterWithLocking(
      config,
      tableName,
      key,
      counterName,
      incrementValue,
      currentValue
    );

    if (response4.success) {
      console.log(`Counter successfully incremented to:
${response4.data.Attributes[counterName]}`);
    } else {
      console.log(response4.error);
    }

    // Explain the differences between ADD and SET
    console.log("\nKey differences between ADD and SET for counter operations:");
    console.log("1. ADD is more concise and idiomatic for simple increments");
    console.log("2. SET with expressions is more flexible for complex
operations");
    console.log("3. Both operations are atomic and safe for concurrent updates");
    console.log("4. SET with if_not_exists is required when the attribute might
not exist");
    console.log("5. Optimistic locking can be added to either approach for
additional safety");
```

```
  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte UpdateItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

Demonstre as operações do contador atômico usando AWS SDK para Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, Union


def increment_counter_with_add(
    table_name: str, key: Dict[str, Any], counter_name: str, increment_value: int
 = 1
) -> Dict[str, Any]:
    """
    Increment a counter attribute using the ADD operation.

    This function demonstrates the atomic ADD operation, which is ideal for
    incrementing counters without the risk of race conditions.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to
 1.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
```

```
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the ADD operation to atomically increment the counter
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #counter :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def increment_counter_with_set(
    table_name: str, key: Dict[str, Any], counter_name: str, increment_value: int
 = 1
) -> Dict[str, Any]:
    """
    Increment a counter attribute using the SET operation with an expression.

    This function demonstrates using SET with an expression to increment a
 counter.
    While this works, it's generally recommended to use ADD for simple
 increments.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to
 1.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the SET operation with an expression to increment the counter
    response = table.update_item(
```

```
        Key=key,
        UpdateExpression="SET #counter = #counter + :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def increment_counter_safely(
    table_name: str,
    key: Dict[str, Any],
    counter_name: str,
    increment_value: int = 1,
    initial_value: int = 0,
) -> Dict[str, Any]:
    """
    Increment a counter attribute safely, handling the case where it might not
 exist.

    This function demonstrates a best practice for incrementing counters by using
    the if_not_exists function to handle the case where the counter doesn't exist
 yet.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to
 1.
        initial_value (int, optional): The initial value if the counter doesn't
 exist. Defaults to 0.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use SET with if_not_exists to safely increment the counter
    response = table.update_item(
```

```
        Key=key,
        UpdateExpression="SET #counter = if_not_exists(#counter, :initial)
 + :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value, ":initial":
 initial_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def atomic_conditional_increment(
    table_name: str,
    key: Dict[str, Any],
    counter_name: str,
    condition_attribute: str,
    condition_value: Any,
    increment_value: int = 1,
) -> Union[Dict[str, Any], None]:
    """
    Atomically increment a counter only if a condition is met.

    This function demonstrates combining atomic counter operations with
    conditional expressions for more complex update scenarios.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.
        increment_value (int, optional): The value to increment by. Defaults to
 1.

    Returns:
        Optional[Dict[str, Any]]: The response from DynamoDB if successful, None
 if condition failed.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
```

```
        # Use ADD with a condition expression
        response = table.update_item(
            Key=key,
            UpdateExpression="ADD #counter :increment",
            ConditionExpression="#condition = :value",
            ExpressionAttributeNames={"#counter": counter_name, "#condition":
 condition_attribute},
            ExpressionAttributeValues={":increment": increment_value, ":value":
 condition_value},
            ReturnValues="UPDATED_NEW",
        )
        return response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return None
        else:
            # Other error occurred
            raise
```

Exemplo de uso de operações de contador atômico com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the atomic counter operations functions."""
    # Example parameters
    table_name = "GameScores"
    key = {"UserId": "user123", "GameId": "game456"}
    counter_name = "Score"

    print("Example 1: Incrementing a counter with ADD operation")
    try:
        response = increment_counter_with_add(
            table_name=table_name, key=key, counter_name=counter_name,
 increment_value=10
        )
        print(
            f"Counter incremented successfully. New value:
 {response.get('Attributes', {}).get(counter_name)}"
        )
    except Exception as e:
```

```
        print(f"Error incrementing counter with ADD: {e}")

    print("\nExample 2: Incrementing a counter with SET operation")
    try:
        response = increment_counter_with_set(
            table_name=table_name, key=key, counter_name=counter_name,
increment_value=5
        )
        print(
            f"Counter incremented successfully. New value:
{response.get('Attributes', {}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error incrementing counter with SET: {e}")

    print("\nExample 3: Safely incrementing a counter that might not exist")
    try:
        new_key = {"UserId": "newuser789", "GameId": "game456"}
        response = increment_counter_safely(
            table_name=table_name,
            key=new_key,
            counter_name=counter_name,
            increment_value=15,
            initial_value=100,
        )
        print(
            f"Counter safely incremented. New value: {response.get('Attributes',
{}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error safely incrementing counter: {e}")

    print("\nExample 4: Conditional counter increment")
    try:
        # Fix for mypy: Handle the case where response might be None
        result = atomic_conditional_increment(
            table_name=table_name,
            key=key,
            counter_name="Achievements",
            condition_attribute="Level",
            condition_value=5,
            increment_value=1,
        )
```

```
        if result is not None:
            print(
                f"Conditional increment succeeded. New value:
{result.get('Attributes', {}).get('Achievements')}"
            )
        else:
            print("Conditional increment failed because condition was not met.")
        if response:
            print(
                f"Conditional increment succeeded. New value:
{response.get('Attributes', {}).get('Achievements')}"
            )
        else:
            print("Conditional increment failed because condition was not met.")
    except Exception as e:
        print(f"Error with conditional increment: {e}")

    print("\nComparison of ADD vs SET for counter operations:")
    print("1. ADD is specifically designed for atomic numeric increments and set
operations")
    print("2. SET with an expression can be used for more complex calculations")
    print("3. Both operations are atomic, preventing race conditions")
    print("4. ADD is more concise for simple increments")
    print("5. SET with if_not_exists() is recommended when the attribute might
not exist")
    print("6. For counters, ADD is generally preferred for clarity and
simplicity")
```

- Para obter detalhes da API, consulte a UpdateItemReferência da API AWS SDK for Python (Boto3).

Use operações condicionais no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como usar operações condicionais no DynamoDB.

- Implemente gravações condicionais para evitar a substituição de dados.
- Use expressões condicionais para impor regras de negócios.
- Lide com falhas de verificação condicional com tranquilidade.

Java

SDK para Java 2.x

Demonstre operações condicionais usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Performs a conditional update on an item.
     *
     * <p>This method demonstrates how to use a condition expression to update an
 item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdate(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue,
```

```
        String updateAttribute,
        AttributeValue updateValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of(
                    "#condAttr", conditionAttribute,
                    "#updateAttr", updateAttribute))
                .expressionAttributeValues(Map.of(
                    ":condVal", conditionValue,
                    ":updateVal", updateValue))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and update was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, update was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
```

```
    }

    /**
     * Performs a conditional delete on an item.
     *
     * <p>This method demonstrates how to use a condition expression to delete an
item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to delete
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalDelete(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the delete parameters
            DeleteItemRequest request = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of("#condAttr",
conditionAttribute))
                .expressionAttributeValues(Map.of(":condVal", conditionValue))
                .returnValues(ReturnValue.ALL_OLD)
                .build();

            // Perform the delete operation
            DeleteItemResponse response = dynamoDbClient.deleteItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and delete was performed");
            result.put("attributes", response.attributes());
```

```
        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, delete was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Demonstrates optimistic locking using a version attribute.
     *
     * <p>This method shows how to implement optimistic locking by using a
version
     * attribute that is incremented with each update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param versionAttribute The name of the version attribute
     * @return Map containing the operation result
     */
    public static Map<String, Object> optimisticLockingExample(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String versionAttribute) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Get the current version of the item
            GetItemRequest getRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .projectionExpression(versionAttribute)
                .build();
```

```
            GetItemResponse getResponse = dynamoDbClient.getItem(getRequest);

            // Check if the item exists
            if (getResponse.item() == null || !
getResponse.item().containsKey(versionAttribute)) {
                // Item doesn't exist or doesn't have a version attribute
                // Initialize with version 1
                UpdateItemRequest initRequest = UpdateItemRequest.builder()
                    .tableName(tableName)
                    .key(key)
                    .updateExpression("SET #verAttr = :newVer, #dataAttr
 = :data")
                    .expressionAttributeNames(Map.of("#verAttr",
 versionAttribute, "#dataAttr", "Data"))
                    .expressionAttributeValues(Map.of(
                        ":newVer", AttributeValue.builder().n("1").build(),
                        ":data", AttributeValue.builder().s("Initial
 data").build()))
                    .returnValues(ReturnValue.UPDATED_NEW)
                    .build();

                UpdateItemResponse initResponse =
 dynamoDbClient.updateItem(initRequest);

                result.put("operation", "initialize");
                result.put("success", true);
                result.put("attributes", initResponse.attributes());

                return result;
            }

            // Get the current version number
            int currentVersion =
                Integer.parseInt(getResponse.item().get(versionAttribute).n());
            int newVersion = currentVersion + 1;

            // Update the item with a condition on the version
            UpdateItemRequest updateRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #verAttr = :newVer, #dataAttr = :newData")
                .conditionExpression("#verAttr = :curVer")
```

```
                .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
"#dataAttr", "Data"))
                .expressionAttributeValues(Map.of(
                    ":curVer",
                        AttributeValue.builder()
                            .n(String.valueOf(currentVersion))
                            .build(),
                    ":newVer",

AttributeValue.builder().n(String.valueOf(newVersion)).build(),
                    ":newData",
                        AttributeValue.builder()
                            .s("Updated data at version " + newVersion)
                            .build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse updateResponse =
dynamoDbClient.updateItem(updateRequest);

            // Record success result
            result.put("operation", "update");
            result.put("success", true);
            result.put("oldVersion", currentVersion);
            result.put("newVersion", newVersion);
            result.put("attributes", updateResponse.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to version mismatch
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Version mismatch, another process may have
updated the item");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
```

```
    }

    /**
     * Performs a conditional update with multiple conditions.
     *
     * <p>This method demonstrates how to use multiple conditions in a condition
expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditions Map of attribute names to values for conditions
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdateWithMultipleConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        Map<String, AttributeValue> conditions,
        String updateAttribute,
        AttributeValue updateValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Build the condition expression and attribute names/values
            StringBuilder conditionExpression = new StringBuilder();
            Map<String, String> expressionAttributeNames = new HashMap<>();
            Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();

            // Add update attribute
            expressionAttributeNames.put("#updateAttr", updateAttribute);
            expressionAttributeValues.put(":updateVal", updateValue);

            // Add conditions
            int i = 0;
            for (Map.Entry<String, AttributeValue> condition :
conditions.entrySet()) {
                String attrName = condition.getKey();
                AttributeValue attrValue = condition.getValue();
```

```
                String nameKey = "#cond" + i;
                String valueKey = ":val" + i;

                expressionAttributeNames.put(nameKey, attrName);
                expressionAttributeValues.put(valueKey, attrValue);

                // Add AND between conditions (except for the first one)
                if (i > 0) {
                    conditionExpression.append(" AND ");
                }

                conditionExpression.append(nameKey).append(" =
").append(valueKey);
                i++;
            }

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression(conditionExpression.toString())
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "All conditions were met and update was
performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "One or more conditions were not met, update
was not performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
```

```
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }
```

Exemplo de uso de operações condicionais com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating conditional operations in DynamoDB");

        try {
            // Example 1: Conditional update
            System.out.println("\nExample 1: Conditional update");
            Map<String, Object> updateResult = conditionalUpdate(
                dynamoDbClient,
                tableName,
                key,
                "InStock",
                AttributeValue.builder().bool(true).build(),
                "Status",
                AttributeValue.builder().s("Available").build());

            System.out.println("Update result: " + updateResult.get("message"));
            if ((boolean) updateResult.get("success")) {
                System.out.println("Updated attributes: " +
 updateResult.get("attributes"));
            }

            // Example 2: Conditional delete
            System.out.println("\nExample 2: Conditional delete");
            Map<String, Object> deleteResult = conditionalDelete(
                dynamoDbClient,
                tableName,
```

```
                key,
                "Status",
                AttributeValue.builder().s("Discontinued").build());

            System.out.println("Delete result: " + deleteResult.get("message"));
            if ((boolean) deleteResult.get("success")) {
                System.out.println("Deleted item: " +
deleteResult.get("attributes"));
            }

            // Example 3: Optimistic locking
            System.out.println("\nExample 3: Optimistic locking");
            Map<String, Object> lockingResult =
optimisticLockingExample(dynamoDbClient, tableName, key, "Version");

            System.out.println("Optimistic locking result:");
            System.out.println("  Operation: " + lockingResult.get("operation"));
            System.out.println("  Success: " + lockingResult.get("success"));
            if (lockingResult.get("operation").equals("update") && (boolean)
lockingResult.get("success")) {
                System.out.println("  Old version: " +
lockingResult.get("oldVersion"));
                System.out.println("  New version: " +
lockingResult.get("newVersion"));
            }
            System.out.println("  Attributes: " +
lockingResult.get("attributes"));

            // Example 4: Multiple conditions
            System.out.println("\nExample 4: Multiple conditions");
            Map<String, AttributeValue> conditions = new HashMap<>();
            conditions.put("Price",
AttributeValue.builder().n("199.99").build());
            conditions.put("Category",
AttributeValue.builder().s("Electronics").build());

            Map<String, Object> multiConditionResult =
conditionalUpdateWithMultipleConditions(
                dynamoDbClient,
                tableName,
                key,
                conditions,
                "OnSale",
                AttributeValue.builder().bool(true).build());
```

```
            System.out.println("Multiple conditions result: " +
multiConditionResult.get("message"));
            if ((boolean) multiConditionResult.get("success")) {
                System.out.println("Updated attributes: " +
multiConditionResult.get("attributes"));
            }

            // Explain conditional operations
            System.out.println("\nKey points about DynamoDB conditional
operations:");
            System.out.println("1. Conditional operations only succeed if the
condition is met");
            System.out.println("2. ConditionalCheckFailedException is thrown when
the condition fails");
            System.out.println("3. No changes are made to the item if the
condition fails");
            System.out.println("4. Conditions can be used with update, delete,
and put operations");
            System.out.println("5. Multiple conditions can be combined with AND
and OR");
            System.out.println("6. Optimistic locking can be implemented using a
version attribute");
            System.out.println(
                "7. Conditional operations consume the same amount of write
capacity whether they succeed or fail");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteItem
  - PutItem
  - UpdateItem

JavaScript

SDK para JavaScript (v3)

Demonstre operações condicionais usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  DeleteCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Perform a conditional update operation.
 *
 * This function demonstrates how to update an item only if a condition is met.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} conditionAttribute - The attribute to check in the condition
 * @param {any} conditionValue - The value to compare against
 * @param {string} updateAttribute - The attribute to update
 * @param {any} updateValue - The new value to set
 * @returns {Promise<Object>} - Result of the operation
 */
async function conditionalUpdate(
  config,
  tableName,
  key,
  conditionAttribute,
  conditionValue,
  updateAttribute,
  updateValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters with a condition expression
  const params = {
```

```
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${updateAttribute} = :value`,
    ConditionExpression: `${conditionAttribute} = :condition`,
    ExpressionAttributeValues: {
      ":value": updateValue,
      ":condition": conditionValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Condition was met and update was performed",
      updatedAttributes: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Condition was not met, update was not performed",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Perform a conditional delete operation.
 *
 * This function demonstrates how to delete an item only if a condition is met.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to delete
 * @param {string} conditionAttribute - The attribute to check in the condition
```

```
 * @param {any} conditionValue - The value to compare against
 * @returns {Promise<Object>} - Result of the operation
 */
async function conditionalDelete(
  config,
  tableName,
  key,
  conditionAttribute,
  conditionValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the delete parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    ConditionExpression: `${conditionAttribute} = :condition`,
    ExpressionAttributeValues: {
      ":condition": conditionValue
    },
    ReturnValues: "ALL_OLD"
  };

  try {
    // Perform the delete operation
    const response = await docClient.send(new DeleteCommand(params));

    return {
      success: true,
      message: "Condition was met and item was deleted",
      deletedItem: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Condition was not met, item was not deleted",
        error: "ConditionalCheckFailedException"
      };
    }
```

```
    // Re-throw other errors
    throw error;
  }
}

/**
 * Implement optimistic locking with a version number.
 *
 * This function demonstrates how to use a version number for optimistic locking
 * to prevent race conditions when multiple processes update the same item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {Object} updates - The attributes to update
 * @param {number} expectedVersion - The expected current version number
 * @returns {Promise<Object>} - Result of the operation
 */
async function updateWithOptimisticLocking(
  config,
  tableName,
  key,
  updates,
  expectedVersion
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression
  const updateExpressions = [];
  const expressionAttributeValues = {
    ":expectedVersion": expectedVersion,
    ":newVersion": expectedVersion + 1
  };

  // Add each update to the expression
  Object.entries(updates).forEach(([attribute, value], index) => {
    updateExpressions.push(`${attribute} = :val${index}`);
    expressionAttributeValues[`:val${index}`] = value;
  });

  // Add the version update
  updateExpressions.push("version = :newVersion");
```

```
  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${updateExpressions.join(", ")}`,
    ConditionExpression: "version = :expectedVersion",
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Update succeeded with optimistic locking",
      newVersion: expectedVersion + 1,
      updatedAttributes: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Optimistic locking failed: the item was modified by another
 process",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Implement a conditional write that creates an item only if it doesn't exist.
 *
 * This function demonstrates how to use attribute_not_exists to create an item
 * only if it doesn't already exist (similar to an "INSERT IF NOT EXISTS"
 operation).
 *
```

```
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} item - The item to create
 * @returns {Promise<Object>} - Result of the operation
 */
async function createIfNotExists(
  config,
  tableName,
  item
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Extract the primary key attributes
  const keyAttributes = Object.keys(item).filter(attr =>
    attr === "id" || attr === "ID" || attr === "Id" ||
    attr.endsWith("Id") || attr.endsWith("ID") ||
    attr.endsWith("Key")
  );

  if (keyAttributes.length === 0) {
    throw new Error("Could not determine primary key attributes");
  }

  // Create a condition expression that checks if the item doesn't exist
  const conditionExpression = `attribute_not_exists(${keyAttributes[0]})`;

  // Define the put parameters with a condition expression
  const params = {
    TableName: tableName,
    Item: item,
    ConditionExpression: conditionExpression
  };

  try {
    // Perform the put operation
    await docClient.send(new PutCommand(params));

    return {
      success: true,
      message: "Item was created because it didn't exist",
      item
    };
```

```
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Item already exists, creation was skipped",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
```

```
  return response.Item || null;
}
```

Exemplo de uso de operações condicionais com AWS SDK para JavaScript.

```
/**
 * Example of how to use conditional operations.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating conditional operations in DynamoDB");

  try {
    // Example 1: Conditional update based on attribute value
    console.log("\nExample 1: Conditional update based on attribute value");
    const updateResult = await conditionalUpdate(
      config,
      tableName,
      key,
      "Category",
      "Electronics",
      "Price",
      299.99
    );

    console.log(`Result: ${updateResult.message}`);
    if (updateResult.success) {
      console.log("Updated attributes:", updateResult.updatedAttributes);
    }

    // Example 2: Conditional delete based on attribute value
    console.log("\nExample 2: Conditional delete based on attribute value");
    const deleteResult = await conditionalDelete(
      config,
      tableName,
      key,
      "InStock",
      false
```

```
    );

    console.log(`Result: ${deleteResult.message}`);
    if (deleteResult.success) {
      console.log("Deleted item:", deleteResult.deletedItem);
    }

    // Example 3: Optimistic locking with version number
    console.log("\nExample 3: Optimistic locking with version number");

    // First, get the current item to check its version
    const currentItem = await getItem(config, tableName, { ProductId:
 "P67890" });
    const currentVersion = currentItem ? (currentItem.version || 0) : 0;

    console.log(`Current version: ${currentVersion}`);

    // Then, update with optimistic locking
    const lockingResult = await updateWithOptimisticLocking(
      config,
      tableName,
      { ProductId: "P67890" },
      {
        Name: "Updated Product Name",
        Description: "This is an updated description"
      },
      currentVersion
    );

    console.log(`Result: ${lockingResult.message}`);
    if (lockingResult.success) {
      console.log(`New version: ${lockingResult.newVersion}`);
      console.log("Updated attributes:", lockingResult.updatedAttributes);
    }

    // Example 4: Create item only if it doesn't exist
    console.log("\nExample 4: Create item only if it doesn't exist");
    const createResult = await createIfNotExists(
      config,
      tableName,
      {
        ProductId: "P99999",
        Name: "New Product",
        Category: "Accessories",
```

```
        Price: 19.99,
        InStock: true
      }
    );

    console.log(`Result: ${createResult.message}`);
    if (createResult.success) {
      console.log("Created item:", createResult.item);
    }

    // Explain conditional operations
    console.log("\nKey points about conditional operations:");
    console.log("1. Conditional operations only succeed if the condition is
 met");
    console.log("2. ConditionalCheckFailedException indicates the condition
 wasn't met");
    console.log("3. Optimistic locking prevents race conditions in concurrent
 updates");
    console.log("4. attribute_exists and attribute_not_exists are useful for
 checking if attributes are present");
    console.log("5. Conditional operations are atomic - they either succeed
 completely or fail completely");
    console.log("6. You can use any valid comparison operators and functions in
 condition expressions");
    console.log("7. Conditional operations don't consume write capacity if the
 condition fails");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteItem
  - PutItem
  - UpdateItem

Python

SDK para Python (Boto3)

Demonstre operações condicionais usando AWS SDK para Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, Optional, Tuple, Union


def conditional_update(
    table_name: str,
    key: Dict[str, Any],
    condition_attribute: str,
    condition_value: Any,
    update_attribute: str,
    update_value: Any,
) -> Tuple[bool, Optional[Dict[str, Any]]]:
    """
    Update an item only if a condition is met.

    This function demonstrates how to perform a conditional update operation
    and determine if the condition was met.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.

    Returns:
        Tuple[bool, Optional[Dict[str, Any]]]: A tuple containing:
            - A boolean indicating if the update succeeded
            - The response from DynamoDB if successful, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        # Perform the conditional update
```

```
        response = table.update_item(
            Key=key,
            UpdateExpression="SET #update_attr = :update_val",
            ConditionExpression="#cond_attr = :cond_val",
            ExpressionAttributeNames={
                "#update_attr": update_attribute,
                "#cond_attr": condition_attribute,
            },
            ExpressionAttributeValues={":update_val": update_value, ":cond_val":
 condition_value},
            ReturnValues="UPDATED_NEW",
        )
        # Update succeeded, condition was met
        return True, response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return False, None
        else:
            # Other error occurred
            raise


def conditional_delete(
    table_name: str, key: Dict[str, Any], condition_attribute: str,
 condition_value: Any
) -> bool:
    """
    Delete an item only if a condition is met.

    This function demonstrates how to perform a conditional delete operation
    and determine if the condition was met.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to delete.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.

    Returns:
        bool: True if the delete succeeded (condition was met), False otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
```

```
    table = dynamodb.Table(table_name)

    try:
        # Perform the conditional delete
        table.delete_item(
            Key=key,
            ConditionExpression="#attr = :val",
            ExpressionAttributeNames={"#attr": condition_attribute},
            ExpressionAttributeValues={":val": condition_value},
        )
        # Delete succeeded, condition was met
        return True
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return False
        else:
            # Other error occurred
            raise


def optimistic_locking_update(
    table_name: str,
    key: Dict[str, Any],
    version_attribute: str,
    update_attribute: str,
    update_value: Any,
) -> Tuple[bool, Optional[Dict[str, Any]]]:
    """
    Update an item using optimistic locking with a version attribute.

    This function demonstrates how to implement optimistic locking using
    a version attribute that is incremented with each update.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        version_attribute (str): The name of the version attribute.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.

    Returns:
        Tuple[bool, Optional[Dict[str, Any]]]: A tuple containing:
            - A boolean indicating if the update succeeded
```

```
            - The response from DynamoDB if successful, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # First, get the current version
    try:
        response = table.get_item(
            Key=key,
            ProjectionExpression=f"#{version_attribute}",
            ExpressionAttributeNames={f"#{version_attribute}":
version_attribute},
        )

        item = response.get("Item", {})
        current_version = item.get(version_attribute, 0)

        # Now, try to update with a condition on the version
        try:
            update_response = table.update_item(
                Key=key,
                UpdateExpression=f"SET #{update_attribute} = :update_val,
#{version_attribute} = :new_version",
                ConditionExpression=f"#{version_attribute} = :current_version",
                ExpressionAttributeNames={
                    f"#{update_attribute}": update_attribute,
                    f"#{version_attribute}": version_attribute,
                },
                ExpressionAttributeValues={
                    ":update_val": update_value,
                    ":current_version": current_version,
                    ":new_version": current_version + 1,
                },
                ReturnValues="UPDATED_NEW",
            )
            # Update succeeded
            return True, update_response
        except ClientError as e:
            if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
                # Version has changed, optimistic locking failed
                return False, None
            else:
                # Other error occurred
```

```
                raise
    except ClientError:
        # Error getting the item
        raise


def conditional_check_and_update(
    table_name: str,
    key: Dict[str, Any],
    check_attribute: str,
    check_value: Any,
    update_attribute: str,
    update_value: Any,
    create_if_not_exists: bool = False,
) -> Union[Dict[str, Any], None]:
    """
    Check if an attribute has a specific value and update another attribute if it
 does.

    This function demonstrates a more complex conditional update that can also
    create the item if it doesn't exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        check_attribute (str): The attribute to check in the condition.
        check_value (Any): The value to compare against.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.
        create_if_not_exists (bool, optional): Whether to create the item if it
 doesn't exist.

    Returns:
        Union[Dict[str, Any], None]: The response from DynamoDB if successful,
 None otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        if create_if_not_exists:
            # Use attribute_not_exists to create the item if it doesn't exist
```

```
            condition_expression = "attribute_not_exists(#pk) OR #check_attr
= :check_val"
            update_expression = "SET #update_attr = :update_val, #check_attr =
if_not_exists(#check_attr, :check_val)"

            # Get the partition key name from the key dictionary
            pk_name = next(iter(key))

            expression_attribute_names = {
                "#pk": pk_name,
                "#check_attr": check_attribute,
                "#update_attr": update_attribute,
            }
        else:
            # Only update if the check attribute has the expected value
            condition_expression = "#check_attr = :check_val"
            update_expression = "SET #update_attr = :update_val"

            expression_attribute_names = {
                "#check_attr": check_attribute,
                "#update_attr": update_attribute,
            }

        # Perform the conditional update
        response = table.update_item(
            Key=key,
            UpdateExpression=update_expression,
            ConditionExpression=condition_expression,
            ExpressionAttributeNames=expression_attribute_names,
            ExpressionAttributeValues={":check_val": check_value, ":update_val":
update_value},
            ReturnValues="UPDATED_NEW",
        )
        return response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return None
        else:
            # Other error occurred
            raise
```

Exemplo de uso de operações condicionais com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the conditional operations functions."""
    # Example parameters
    table_name = "Products"
    key = {"ProductId": "prod123"}

    print("Example 1: Conditional Update")
    try:
        # Update the price only if the current stock is greater than 10
        success, response = conditional_update(
            table_name=table_name,
            key=key,
            condition_attribute="Stock",
            condition_value=10,
            update_attribute="Price",
            update_value=99.99,
        )

        if success:
            # Fix for mypy: Handle the case where response might be None
            attributes = {} if response is None else response.get("Attributes",
 {})
            print(f"Update succeeded! New values: {attributes}")
        else:
            print("Update failed because the condition was not met.")
    except Exception as e:
        print(f"Error during conditional update: {e}")

    print("\nExample 2: Conditional Delete")
    try:
        # Delete the product only if it's discontinued
        success = conditional_delete(
            table_name=table_name,
            key=key,
            condition_attribute="Status",
            condition_value="Discontinued",
        )

        if success:
```

```
            print("Delete succeeded! The item was deleted.")
        else:
            print("Delete failed because the condition was not met.")
    except Exception as e:
        print(f"Error during conditional delete: {e}")

    print("\nExample 3: Optimistic Locking")
    try:
        # Update with optimistic locking using a version attribute
        success, response = optimistic_locking_update(
            table_name=table_name,
            key=key,
            version_attribute="Version",
            update_attribute="Description",
            update_value="Updated product description",
        )

        if success:
            # Fix for mypy: Handle the case where response might be None
            attributes = {} if response is None else response.get("Attributes",
{})
            print(f"Optimistic locking update succeeded! New values:
{attributes}")
        else:
            print("Optimistic locking update failed because the version has
changed.")
    except Exception as e:
        print(f"Error during optimistic locking update: {e}")

    print("\nExample 4: Conditional Check and Update")
    try:
        # Update the featured status if the product is in stock
        response = conditional_check_and_update(
            table_name=table_name,
            key=key,
            check_attribute="InStock",
            check_value=True,
            update_attribute="Featured",
            update_value=True,
            create_if_not_exists=True,
        )

        if response:
            print(
```

```
                f"Conditional check and update succeeded! New values:
{response.get('Attributes', {})}"
            )
        else:
            print("Conditional check and update failed because the condition was
not met.")
    except Exception as e:
        print(f"Error during conditional check and update: {e}")

    print("\nUnderstanding Conditional Operations in DynamoDB:")
    print("1. Conditional operations help maintain data integrity")
    print("2. They prevent race conditions in concurrent environments")
    print("3. Failed conditions result in ConditionalCheckFailedException")
    print("4. No DynamoDB capacity is consumed when conditions fail")
    print("5. Optimistic locking is a common pattern using version attributes")
    print("6. Conditions can be combined with logical operators (AND, OR, NOT)")
    print("7. Conditions can use comparison operators (=, <>, <, <=, >, >=)")
    print(
        "8. attribute_exists() and attribute_not_exists() are useful for checking
attribute presence"
    )
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DeleteItem
  - PutItem
  - UpdateItem

Use nomes de atributos de expressão no DynamoDB com um AWS SDK

Os exemplos de código a seguir mostram como usar nomes de atributo de expressão no DynamoDB.

- Trabalhe com palavras reservadas nas expressões do DynamoDB.
- Use espaços reservados para nomes de atributo de expressão.
- Manipule caracteres especiais em nomes de atributo.

Java

SDK para Java 2.x

Demonstre nomes de atributos de expressão usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates an attribute that is a reserved word in DynamoDB.
     *
     * <p>This method demonstrates how to use expression attribute names to
 update
     * attributes that are reserved words in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param reservedWordAttribute The reserved word attribute to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
```

```
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", reservedWordAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates an attribute that contains special characters.
     *
     * <p>This method demonstrates how to use expression attribute names to
update
     * attributes that contain special characters.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param specialCharAttribute The attribute with special characters to
update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateSpecialCharacterAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String specialCharAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", specialCharAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
```

```
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Queries items using an attribute that is a reserved word.
     *
     * <p>This method demonstrates how to use expression attribute names in a
query
     * when the attribute is a reserved word.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key
     * @param reservedWordAttribute The reserved word attribute to filter on
     * @param value The value to compare against
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse queryWithReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the query parameters using expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attr", reservedWordAttribute);

        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);
        expressionAttributeValues.put(":value", value);

        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attr = :value")
            .expressionAttributeNames(expressionAttributeNames)
```

```
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Updates a nested attribute with a path that contains reserved words.
     *
     * <p>This method demonstrates how to use expression attribute names to
update
     * nested attributes where the path contains reserved words.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributePath The path to the nested attribute as an array
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateNestedReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> attributePath,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < attributePath.size(); i++) {
            expressionAttributeNames.put("#attr" + i, attributePath.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < attributePath.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }
```

```
        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Scans a table with multiple attribute name placeholders.
     *
     * <p>This method demonstrates how to use multiple expression attribute names
     * in a complex filter expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param filters Object mapping attribute names to filter values
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithMultipleAttributeNames(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> filters) {

        // Create expression attribute names and values
        Map<String, String> expressionAttributeNames = new HashMap<>();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        StringBuilder filterExpression = new StringBuilder();

        // Build the filter expression
        int index = 0;
        for (Map.Entry<String, AttributeValue> entry : filters.entrySet()) {
            String attrName = entry.getKey();
            AttributeValue attrValue = entry.getValue();

            String nameKey = "#attr" + index;
            String valueKey = ":val" + index;
```

```
            expressionAttributeNames.put(nameKey, attrName);
            expressionAttributeValues.put(valueKey, attrValue);

            // Add AND between conditions (except for the first one)
            if (index > 0) {
                filterExpression.append(" AND ");
            }

            filterExpression.append(nameKey).append(" = ").append(valueKey);
            index++;
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression(filterExpression.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }
```

Exemplo de uso de nomes de atributos de expressão com AWS SDK for Java 2.x.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating expression attribute names in
DynamoDB");

       try {
           // Example 1: Update an attribute that is a reserved word
           System.out.println("\nExample 1: Updating an attribute that is a
reserved word");
           UpdateItemResponse response1 = updateReservedWordAttribute(
               dynamoDbClient,
```

```
                tableName,
                key,
                "Size", // "SIZE" is a reserved word in DynamoDB
                AttributeValue.builder().s("Large").build());

            System.out.println("Updated attribute: " + response1.attributes());

            // Example 2: Update an attribute with special characters
            System.out.println("\nExample 2: Updating an attribute with special
characters");
            UpdateItemResponse response2 = updateSpecialCharacterAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Product-Type", // Contains a hyphen, which is a special
character
                AttributeValue.builder().s("Electronics").build());

            System.out.println("Updated attribute: " + response2.attributes());

            // Example 3: Query with a reserved word attribute
            System.out.println("\nExample 3: Querying with a reserved word
attribute");
            QueryResponse response3 = queryWithReservedWordAttribute(
                dynamoDbClient,
                tableName,
                "Category",
                AttributeValue.builder().s("Electronics").build(),
                "Count", // "COUNT" is a reserved word in DynamoDB
                AttributeValue.builder().n("10").build());

            System.out.println("Found " + response3.count() + " items");

            // Example 4: Update a nested attribute with reserved words in the
path
            System.out.println("\nExample 4: Updating a nested attribute with
reserved words in the path");
            UpdateItemResponse response4 = updateNestedReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                Arrays.asList("Dimensions", "Size", "Height"), // "SIZE" is a
reserved word
                AttributeValue.builder().n("30").build());
```

```
            System.out.println("Updated nested attribute: " +
 response4.attributes());

            // Example 5: Scan with multiple attribute name placeholders
            System.out.println("\nExample 5: Scanning with multiple attribute
 name placeholders");
            Map<String, AttributeValue> filters = new HashMap<>();
            filters.put("Size", AttributeValue.builder().s("Large").build());
            filters.put("Count", AttributeValue.builder().n("10").build());
            filters.put(
                "Product-Type",
 AttributeValue.builder().s("Electronics").build());

            ScanResponse response5 =
 scanWithMultipleAttributeNames(dynamoDbClient, tableName, filters);

            System.out.println("Found " + response5.count() + " items");

            // Show some common reserved words
            System.out.println("\nSome common DynamoDB reserved words:");
            List<String> commonReservedWords = getDynamoDBReservedWords();
            System.out.println(String.join(", ", commonReservedWords));

            // Explain expression attribute names
            System.out.println("\nKey points about expression attribute names:");
            System.out.println("1. Use expression attribute names (#name) for
 reserved words");
            System.out.println("2. Use expression attribute names for attributes
 with special characters");
            System.out.println(
                "3. Special characters include: spaces, hyphens, dots, and other
 non-alphanumeric characters");
            System.out.println("4. Expression attribute names are required for
 nested attributes with reserved words");
            System.out.println("5. You can use multiple expression attribute
 names in a single expression");
            System.out.println("6. Expression attribute names are case-
sensitive");
            System.out.println("7. Expression attribute names are only used in
 expressions, not in the actual data");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
```

```
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - Consulta
  - UpdateItem

JavaScript

SDK para JavaScript (v3)

Demonstre nomes de atributos de expressão usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  QueryCommand,
  ScanCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update an attribute that is a reserved word in DynamoDB.
 *
 * This function demonstrates how to use expression attribute names to update
 * attributes that are reserved words in DynamoDB.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} reservedWordAttribute - The reserved word attribute to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateReservedWordAttribute(
  config,
  tableName,
  key,
```

```
  reservedWordAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using expression attribute names
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: "SET #attr = :value",
    ExpressionAttributeNames: {
      "#attr": reservedWordAttribute
    },
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update an attribute that contains special characters.
 *
 * This function demonstrates how to use expression attribute names to update
 * attributes that contain special characters.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} specialCharAttribute - The attribute with special characters
 to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateSpecialCharacterAttribute(
  config,
  tableName,
```

```
  key,
  specialCharAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using expression attribute names
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: "SET #attr = :value",
    ExpressionAttributeNames: {
      "#attr": specialCharAttribute
    },
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Query items using an attribute that is a reserved word.
 *
 * This function demonstrates how to use expression attribute names in a query
 * when the attribute is a reserved word.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key attribute
 * @param {any} partitionKeyValue - The value of the partition key
 * @param {string} reservedWordAttribute - The reserved word attribute to filter
 on
 * @param {any} value - The value to compare against
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function queryWithReservedWordAttribute(
```

```
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  reservedWordAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the query parameters using expression attribute names
  const params = {
    TableName: tableName,
    KeyConditionExpression: "#pkName = :pkValue",
    FilterExpression: "#attr = :value",
    ExpressionAttributeNames: {
      "#pkName": partitionKeyName,
      "#attr": reservedWordAttribute
    },
    ExpressionAttributeValues: {
      ":pkValue": partitionKeyValue,
      ":value": value
    }
  };

  // Perform the query operation
  const response = await docClient.send(new QueryCommand(params));

  return response;
}

/**
 * Update a nested attribute with a path that contains reserved words.
 *
 * This function demonstrates how to use expression attribute names to update
 * nested attributes where the path contains reserved words.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string[]} attributePath - The path to the nested attribute as an array
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
```

```
 */
async function updateNestedReservedWordAttribute(
  config,
  tableName,
  key,
  attributePath,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create expression attribute names for each part of the path
  const expressionAttributeNames = {};
  for (let i = 0; i < attributePath.length; i++) {
    expressionAttributeNames[`#attr${i}`] = attributePath[i];
  }

  // Build the attribute path using the expression attribute names
  const attributePathExpression = attributePath
    .map((_, i) => `#attr${i}`)
    .join(".");

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${attributePathExpression} = :value`,
    ExpressionAttributeNames: expressionAttributeNames,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Scan a table with multiple attribute name placeholders.
 *
```

```
 * This function demonstrates how to use multiple expression attribute names
 * in a complex filter expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} filters - Object mapping attribute names to filter values
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function scanWithMultipleAttributeNames(
  config,
  tableName,
  filters
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create expression attribute names and values
  const expressionAttributeNames = {};
  const expressionAttributeValues = {};
  const filterConditions = [];

  // Build the filter expression
  Object.entries(filters).forEach(([attrName, value], index) => {
    const nameKey = `#attr${index}`;
    const valueKey = `:val${index}`;

    expressionAttributeNames[nameKey] = attrName;
    expressionAttributeValues[valueKey] = value;
    filterConditions.push(`${nameKey} = ${valueKey}`);
  });

  // Join the filter conditions with AND
  const filterExpression = filterConditions.join(" AND ");

  // Define the scan parameters
  const params = {
    TableName: tableName,
    FilterExpression: filterExpression,
    ExpressionAttributeNames: expressionAttributeNames,
    ExpressionAttributeValues: expressionAttributeValues
  };

  // Perform the scan operation
```

```
  const response = await docClient.send(new ScanCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Exemplo de uso de nomes de atributos de expressão com AWS SDK para JavaScript.

```
/**
 * Example of how to use expression attribute names.
 */
```

```
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating expression attribute names in DynamoDB");

  try {
    // Example 1: Update an attribute that is a reserved word
    console.log("\nExample 1: Updating an attribute that is a reserved word");
    const response1 = await updateReservedWordAttribute(
      config,
      tableName,
      key,
      "Size", // "SIZE" is a reserved word in DynamoDB
      "Large"
    );

    console.log("Updated attribute:", response1.Attributes);

    // Example 2: Update an attribute with special characters
    console.log("\nExample 2: Updating an attribute with special characters");
    const response2 = await updateSpecialCharacterAttribute(
      config,
      tableName,
      key,
      "Product-Type", // Contains a hyphen, which is a special character
      "Electronics"
    );

    console.log("Updated attribute:", response2.Attributes);

    // Example 3: Query with a reserved word attribute
    console.log("\nExample 3: Querying with a reserved word attribute");
    const response3 = await queryWithReservedWordAttribute(
      config,
      tableName,
      "Category",
      "Electronics",
      "Count", // "COUNT" is a reserved word in DynamoDB
      10
    );
```

```
    console.log(`Found ${response3.Items.length} items`);

    // Example 4: Update a nested attribute with reserved words in the path
    console.log("\nExample 4: Updating a nested attribute with reserved words in
the path");
    const response4 = await updateNestedReservedWordAttribute(
      config,
      tableName,
      key,
      ["Dimensions", "Size", "Height"], // "SIZE" is a reserved word
      30
    );

    console.log("Updated nested attribute:", response4.Attributes);

    // Example 5: Scan with multiple attribute name placeholders
    console.log("\nExample 5: Scanning with multiple attribute name
placeholders");
    const response5 = await scanWithMultipleAttributeNames(
      config,
      tableName,
      {
        "Size": "Large",
        "Count": 10,
        "Product-Type": "Electronics"
      }
    );

    console.log(`Found ${response5.Items.length} items`);

    // Get the final state of the item
    console.log("\nFinal state of the item:");
    const item = await getItem(config, tableName, key);
    console.log(JSON.stringify(item, null, 2));

    // Show some common reserved words
    console.log("\nSome common DynamoDB reserved words:");
    const commonReservedWords = [
      "ABORT", "ABSOLUTE", "ACTION", "ADD", "ALL", "ALTER", "AND", "ANY", "AS",
      "ASC", "BETWEEN", "BY", "CASE", "CAST", "COLUMN", "CONNECT", "COUNT",
      "CREATE", "CURRENT", "DATE", "DELETE", "DESC", "DROP", "ELSE", "EXISTS",
      "FOR", "FROM", "GRANT", "GROUP", "HAVING", "IN", "INDEX", "INSERT", "INTO",
      "IS", "JOIN", "KEY", "LEVEL", "LIKE", "LIMIT", "LOCAL", "MAX", "MIN",
"NAME",
```

```
      "NOT", "NULL", "OF", "ON", "OR", "ORDER", "OUTER", "REPLACE", "RETURN",
      "SELECT", "SET", "SIZE", "TABLE", "THEN", "TO", "UPDATE", "USER", "VALUES",
      "VIEW", "WHERE"
    ];
    console.log(commonReservedWords.join(", "));

    // Explain expression attribute names
    console.log("\nKey points about expression attribute names:");
    console.log("1. Use expression attribute names (#name) for reserved words");
    console.log("2. Use expression attribute names for attributes with special
 characters");
    console.log("3. Special characters include: spaces, hyphens, dots, and other
 non-alphanumeric characters");
    console.log("4. Expression attribute names are required for nested attributes
 with reserved words");
    console.log("5. You can use multiple expression attribute names in a single
 expression");
    console.log("6. Expression attribute names are case-sensitive");
    console.log("7. Expression attribute names are only used in expressions, not
 in the actual data");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - Consulta
  - UpdateItem

Python

SDK para Python (Boto3)

Demonstre nomes de atributos de expressão usando AWS SDK para Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, List
```

```
def use_reserved_word_attribute(
    table_name: str, key: Dict[str, Any], reserved_word: str, value: Any
) -> Dict[str, Any]:
    """
    Update an attribute whose name is a DynamoDB reserved word.

    This function demonstrates how to use expression attribute names to work with
    attributes that have names that are DynamoDB reserved words.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        reserved_word (str): The reserved word to use as an attribute name.
        value (Any): The value to set for the attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names to handle the reserved word
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #reserved_attr = :value",
        ExpressionAttributeNames={"#reserved_attr": reserved_word},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def use_special_character_attribute(
    table_name: str, key: Dict[str, Any], attribute_with_special_chars: str,
 value: Any
) -> Dict[str, Any]:
    """
    Update an attribute whose name contains special characters.

    This function demonstrates how to use expression attribute names to work with
```

```
    attributes that have names containing special characters like spaces, dots,
 or hyphens.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        attribute_with_special_chars (str): The attribute name with special
 characters.
        value (Any): The value to set for the attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names to handle special characters
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #special_attr = :value",
        ExpressionAttributeNames={"#special_attr": attribute_with_special_chars},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def query_with_attribute_names(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    filter_attribute_name: str,
    filter_value: Any,
) -> Dict[str, Any]:
    """
    Query a table using expression attribute names for both key and filter
 attributes.

    This function demonstrates how to use expression attribute names in a query
 operation
    for both the key condition expression and filter expression.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute_name (str): The name of the attribute to filter on.
        filter_value (Any): The value to compare against in the filter.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names for both key condition and filter
    response = table.query(
        KeyConditionExpression="#pk = :pk_val",
        FilterExpression="#filter_attr = :filter_val",
        ExpressionAttributeNames={"#pk": partition_key_name, "#filter_attr":
 filter_attribute_name},
        ExpressionAttributeValues={":pk_val": partition_key_value, ":filter_val":
 filter_value},
    )

    return response


def update_nested_attribute_with_dots(
    table_name: str, key: Dict[str, Any], path_with_dots: str, value: Any
) -> Dict[str, Any]:
    """
    Update a nested attribute using a path with dot notation.

    This function demonstrates how to use expression attribute names to work with
    nested attributes specified using dot notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path_with_dots (str): The path to the nested attribute using dot notation
 (e.g., "a.b.c").
        value (Any): The value to set for the nested attribute.
```

```
    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Split the path into components
    path_parts = path_with_dots.split(".")

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        name_placeholder = f"#attr{i}"
        expression_attribute_names[name_placeholder] = part

        if i == 0:
            path_expression = name_placeholder
        else:
            path_expression += f".{name_placeholder}"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def demonstrate_attribute_name_requirements(table_name: str, key: Dict[str, Any])
 -> Dict[str, Any]:
    """
```

```
    Demonstrate the requirements and allowed characters for attribute names.

    This function shows examples of valid and invalid attribute names and how to
    handle them using expression attribute names.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Examples of attribute names with different characteristics
    examples = {
        "valid_standard": "NormalAttribute",  # Standard attribute name (no
placeholder needed)
        "valid_with_underscore": "Normal_Attribute",  # Underscore is allowed
        "valid_with_number": "Attribute123",  # Numbers are allowed
        "reserved_word": "Timestamp",  # Reserved word (requires placeholder)
        "starts_with_number": "123Attribute",  # Starts with number (valid but
may need placeholder in some contexts)
        "with_space": "Attribute Name",  # Contains space (requires placeholder)
        "with_dot": "Attribute.Name",  # Contains dot (requires placeholder)
        "with_hyphen": "Attribute-Name",  # Contains hyphen (requires
placeholder)
        "with_special_chars": "Attribute#$%",  # Contains special characters
(requires placeholder)
    }

    results = {}

    # Try to update each attribute type
    for example_type, attr_name in examples.items():
        try:
            # For attributes that don't need placeholders, try direct reference
            if example_type in ["valid_standard", "valid_with_underscore",
"valid_with_number"]:
                try:
                    # Try without expression attribute names first
                    response = table.update_item(
```

```
                        Key=key,
                        UpdateExpression=f"SET {attr_name} = :value",
                        ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                        ReturnValues="UPDATED_NEW",
                    )
                    results[example_type] = {
                        "attribute_name": attr_name,
                        "success": True,
                        "needed_placeholder": False,
                        "response": response,
                    }
                except ClientError:
                    # If direct reference fails, try with placeholder
                    response = table.update_item(
                        Key=key,
                        UpdateExpression="SET #attr = :value",
                        ExpressionAttributeNames={"#attr": attr_name},
                        ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                        ReturnValues="UPDATED_NEW",
                    )
                    results[example_type] = {
                        "attribute_name": attr_name,
                        "success": True,
                        "needed_placeholder": True,
                        "response": response,
                    }
            else:
                # For attributes that definitely need placeholders
                response = table.update_item(
                    Key=key,
                    UpdateExpression="SET #attr = :value",
                    ExpressionAttributeNames={"#attr": attr_name},
                    ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                    ReturnValues="UPDATED_NEW",
                )
                results[example_type] = {
                    "attribute_name": attr_name,
                    "success": True,
                    "needed_placeholder": True,
                    "response": response,
                }
```

```
        except ClientError as e:
            results[example_type] = {"attribute_name": attr_name, "success":
 False, "error": str(e)}

    return results
```

Exemplo de uso de nomes de atributos de expressão com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use expression attribute names in DynamoDB."""
    # Example parameters
    table_name = "Products"
    key = {"ProductId": "prod123"}

    print("Example 1: Using a reserved word as an attribute name")
    try:
        response = use_reserved_word_attribute(
            table_name=table_name, key=key, reserved_word="Timestamp",
 value="2025-05-14T12:00:00Z"
        )
        print(f"Reserved word attribute updated successfully:
 {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating reserved word attribute: {e}")

    print("\nExample 2: Using an attribute name with special characters")
    try:
        response = use_special_character_attribute(
            table_name=table_name,
            key=key,
            attribute_with_special_chars="Product Info",
            value="Special product information",
        )
        print(f"Special character attribute updated successfully:
 {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating special character attribute: {e}")

    print("\nExample 3: Querying with expression attribute names")
    try:
```

```
        response = query_with_attribute_names(
            table_name=table_name,
            partition_key_name="Category",
            partition_key_value="Electronics",
            filter_attribute_name="Price",
            filter_value=500,
        )
        print(
            f"Query with expression attribute names returned
{len(response.get('Items', []))} items"
        )
    except Exception as e:
        print(f"Error querying with expression attribute names: {e}")

    print("\nExample 4: Updating a nested attribute with dot notation")
    try:
        response = update_nested_attribute_with_dots(
            table_name=table_name,
            key=key,
            path_with_dots="Product.Details.Specifications",
            value={"Weight": "2.5 kg", "Dimensions": "30x20x10 cm"},
        )
        print(f"Nested attribute updated successfully:
{response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating nested attribute: {e}")

    print("\nExample 5: Demonstrating attribute name requirements")
    try:
        results = demonstrate_attribute_name_requirements(table_name=table_name,
key=key)

        print("Attribute Name Requirements Results:")
        for example_type, result in results.items():
            if result.get("success", False):
                needed_placeholder = result.get("needed_placeholder", True)
                print(
                    f"  - {example_type}: '{result['attribute_name']}' -
{'Requires' if needed_placeholder else 'Does not require'} placeholder"
                )
            else:
                print(
                    f"  - {example_type}: '{result['attribute_name']}' - Failed:
{result.get('error', 'Unknown error')}"
```

```
                )
    except Exception as e:
        print(f"Error demonstrating attribute name requirements: {e}")

    print("\nCommon DynamoDB Reserved Words (sample):")
    reserved_words = get_common_reserved_words()
    print(", ".join(reserved_words[:20]) + "... (and many more)")

    print("\nWhen to Use Expression Attribute Names:")
    print("1. When the attribute name is a DynamoDB reserved word")
    print("2. When the attribute name contains special characters (spaces, dots,
hyphens)")
    print("3. When the attribute name begins with a number")
    print("4. When working with nested attributes using dot notation")
    print("5. When you need to reference the same attribute multiple times in an
expression")

    print("\nExpression Attribute Name Requirements:")
    print("1. Must begin with a pound sign (#)")
    print("2. After the pound sign, must contain at least one character")
    print("3. Can contain alphanumeric characters and underscore (_)")
    print("4. Are case-sensitive")
    print("5. Must be unique within a single expression")

    print("\nAttribute Name Requirements in DynamoDB:")
    print("1. Can begin with a-z, A-Z, or 0-9")
    print("2. Can contain a-z, A-Z, 0-9, underscore (_), dash (-), and dot (.)")
    print("3. Are case-sensitive")
    print("4. No length restrictions, but practical limits apply")
    print("5. Cannot be a DynamoDB reserved word if used directly in
expressions")
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - Consulta
  - UpdateItem

Usar eventos programados para chamar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

Java

SDK para Java 2.x

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função Lambda usando a API de tempo de execução do JavaScript Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento programado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função Lambda o alvo.
- Concede permissão para permitir a EventBridge invocação da função Lambda.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações programadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda

- Amazon SNS

Trabalhe com índices secundários locais do DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como criar e consultar tabelas com índices secundários locais.

- Crie uma tabela com um índice secundário local (LSI).
- Crie uma tabela com vários LSIs com diferentes tipos de projeção.
- Consulte dados usando LSIs.

Bash

AWS CLI com script Bash

Crie uma tabela com um índice secundário local.

```
# Create a table with a Local Secondary Index
aws dynamodb create-table \
    --table-name CustomerOrders \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
        AttributeName=OrderID,AttributeType=S \
        AttributeName=OrderDate,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
        AttributeName=OrderID,KeyType=RANGE \
    --local-secondary-indexes \
        "IndexName=OrderDateIndex,\
        KeySchema=[{AttributeName=CustomerID,KeyType=HASH},
{AttributeName=OrderDate,KeyType=RANGE}],\
        Projection={ProjectionType=ALL}" \
    --billing-mode PAY_PER_REQUEST
```

Crie uma tabela com vários LSIs.

```
# Create a table with multiple Local Secondary Indexes
aws dynamodb create-table \
    --table-name CustomerDetails \
```

```
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
        AttributeName=Name,AttributeType=S \
        AttributeName=Email,AttributeType=S \
        AttributeName=RegistrationDate,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
        AttributeName=Name,KeyType=RANGE \
    --local-secondary-indexes \
        "[
            {
                \"IndexName\": \"EmailIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"CustomerID\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"Email\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"INCLUDE\",
\"NonKeyAttributes\":[\"Address\",\"Phone\"]}
            },
            {
                \"IndexName\": \"RegistrationIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"CustomerID\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"RegistrationDate\",\"KeyType\":\"RANGE
\"}
                ],
                \"Projection\": {\"ProjectionType\":\"KEYS_ONLY\"}
            }
        ]" \
    --billing-mode PAY_PER_REQUEST
```

Consulte dados usando LSIs.

```
# Query the OrderDateIndex LSI
aws dynamodb query \
    --table-name CustomerOrders \
    --index-name OrderDateIndex \
    --key-condition-expression "CustomerID = :custId AND OrderDate BETWEEN :date1
 AND :date2" \
    --expression-attribute-values '{
        ":custId": {"S": "C1"},
        ":date1": {"S": "2023-01-01"},
```

```
        ":date2": {"S": "2023-02-01"}
    }'

# Query with a filter expression
aws dynamodb query \
    --table-name CustomerOrders \
    --index-name OrderDateIndex \
    --key-condition-expression "CustomerID = :custId" \
    --filter-expression "Amount > :amount" \
    --expression-attribute-values '{
        ":custId": {"S": "C1"},
        ":amount": {"N": "150"}
    }'
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - Consulta

Trabalhe com o DynamoDB Streams e use Time-to-Live AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como gerenciar os recursos e os streams do DynamoDB. Time-to-Live

- Crie uma tabela com a opção de fluxos habilitada.
- Descreva os fluxos.
- Crie uma função do Lambda para processar os fluxos.
- Habilite a vida útil (TTL) em uma tabela.
- Adicione itens com atributos de TTL.
- Descreva as configurações de TTL.

Bash

AWS CLI com script Bash

Crie uma tabela com a opção de fluxos habilitada.

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

Descreva os fluxos.

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)

echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

Crie uma função do Lambda para os fluxos.

```
# Step 1: Create an IAM role for the Lambda function
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
echo "    --starting-position LATEST"
```

Habilite a vida útil (TTL) em uma tabela.

```
# Create a table for TTL demonstration
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo
```

```
# Enable TTL on the table
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

Adicione itens com atributos de TTL.

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

Descreva as configurações de TTL.

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
    --table-name TTLDemo
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole

- CreateTable
- DescribeTable
- DescribeTimeToLive
- PutItem
- UpdateTimeToLive

Trabalhe com tabelas globais do DynamoDB e replicação multirregional com consistência eventual (MREC) usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como gerenciar tabelas globais do DynamoDB com consistência final de replicação mutirregional (MREC).

- Crie uma tabela com replicação mutirregional (MREC).
- Insira e obtenha itens de tabelas-réplica.
- Remova as réplicas uma por uma.
- Exclua a tabela para fazer uma limpeza.

Bash

AWS CLI com script Bash

Crie uma tabela com replicação mutirregional.

```
# Step 1: Create a new table (MusicTable) in US East (Ohio), with DynamoDB
 Streams enabled (NEW_AND_OLD_IMAGES)
aws dynamodb create-table \
    --table-name MusicTable \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES \
    --region us-east-2

# Step 2: Create an identical MusicTable table in US East (N. Virginia)
```

```
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Create": {
        "RegionName": "us-east-1"
      }
    }
  ]
}' \
--region us-east-2

# Step 3: Create a table in Europe (Ireland)
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Create": {
        "RegionName": "eu-west-1"
      }
    }
  ]
}' \
--region us-east-2
```

Descreva a tabela multirregional.

```
# Step 4: View the list of replicas created using describe-table
aws dynamodb describe-table \
    --table-name MusicTable \
    --region us-east-2 \
    --query 'Table.
{TableName:TableName,TableStatus:TableStatus,MultiRegionConsistency:MultiRegionConsistenc
{Region:RegionName,Status:ReplicaStatus}}'
```

Insira itens em uma tabela-réplica.

```
# Step 5: To verify that replication is working, add a new item to the Music
 table in US East (Ohio)
```

```
aws dynamodb put-item \
    --table-name MusicTable \
    --item '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region us-east-2
```

Obtenha itens de tabelas-réplica.

```
# Step 6: Wait for a few seconds, and then check to see whether the item has
 been
# successfully replicated to US East (N. Virginia) and Europe (Ireland)
aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region us-east-1

aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region eu-west-1
```

Remova as réplicas.

```
# Step 7: Delete the replica table in Europe (Ireland) Region
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Delete": {
        "RegionName": "eu-west-1"
      }
    }
  ]
}' \
--region us-east-2

# Delete the replica table in US East (N. Virginia) Region
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
```

```
    {
      "Delete": {
        "RegionName": "us-east-1"
      }
    }
  ]
}' \
--region us-east-2
```

Exclua a tabela para fazer uma limpeza.

```
# Clean up: Delete the primary table
aws dynamodb delete-table --table-name MusicTable --region us-east-2

echo "Global table demonstration complete."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateTable

Trabalhe com a marcação de recursos do DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como gerenciar tags para recursos do DynamoDB.

- Crie uma tabela com tags.
- Listar as etiquetas de um recurso.
- Adicione tags a um recurso.
- Remova as tags de um recurso.
- Filtre tabelas por tags.

Bash

AWS CLI com script Bash

Crie uma tabela com tags.

```
# Create a table with tags
aws dynamodb create-table \
    --table-name TaggedTable \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Environment,Value=Production \
        Key=Project,Value=Analytics \
        Key=Owner,Value=DataTeam
```

Listar as etiquetas de um recurso.

```
# Get the table ARN
TABLE_ARN=$(aws dynamodb describe-table \
    --table-name TaggedTable \
    --query "Table.TableArn" \
    --output text)

# List tags for the table
aws dynamodb list-tags-of-resource \
    --resource-arn $TABLE_ARN
```

Adicione tags a um recurso.

```
# Add tags to an existing table
aws dynamodb tag-resource \
    --resource-arn $TABLE_ARN \
    --tags \
        Key=CostCenter,Value=12345 \
        Key=BackupSchedule,Value=Daily
```

Remova as tags de um recurso.

```
# Remove tags from a table
aws dynamodb untag-resource \
    --resource-arn $TABLE_ARN \
    --tag-keys Owner BackupSchedule
```

Filtre tabelas por tags.

```
# Create another table with different tags
aws dynamodb create-table \
    --table-name AnotherTaggedTable \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Environment,Value=Development \
        Key=Project,Value=Testing

# Wait for table to become active
aws dynamodb wait table-exists --table-name AnotherTaggedTable

# List all tables
echo "All tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nFiltering tables by Environment=Production tag:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text |
 xargs -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn"
 --output text)

# Find tables with specific tag
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
```

```
done
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - ListTagsOfResource
  - TagResource
  - UntagResource

Trabalhe com a criptografia de tabelas do DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como gerenciar opções de criptografia para tabelas do DynamoDB.

- Crie uma tabela com a criptografia padrão.
- Crie uma tabela com uma chave gerenciada pelo cliente (CMK).
- Atualize as configurações de criptografia da tabela.
- Descreva a criptografia da tabela.

Bash

AWS CLI com script Bash

Crie uma tabela com a criptografia padrão.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Crie uma tabela com uma chave gerenciada pelo cliente (CMK).

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Atualize a criptografia da tabela.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Descreva a criptografia da tabela.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateKey
  - CreateTable
  - DescribeTable

- UpdateTable

Exemplos de tecnologia sem servidor para o DynamoDB

Os exemplos de código a seguir mostram como usar o DynamoDB com AWS SDKs.

Exemplos

- Invocar uma função do Lambda em um gatilho do DynamoDB
- Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

Invocar uma função do Lambda em um gatilho do DynamoDB

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
```

```
namespace AWSLambda_DDB;

public class Function
{
    public void FunctionHandler(DynamoDBEvent dynamoEvent, ILambdaContext
 context)
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");

        foreach (var record in dynamoEvent.Records)
        {
            context.Logger.LogInformation($"Event ID: {record.EventID}");
            context.Logger.LogInformation($"Event Name: {record.EventName}");

            context.Logger.LogInformation(JsonSerializer.Serialize(record));
        }

        context.Logger.LogInformation("Stream processing complete.");
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
```

```
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-lambda-go/events"
 "fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string,
 error) {
 if len(event.Records) == 0 {
  return nil, fmt.Errorf("received empty event")
 }

 for _, record := range event.Records {
   LogDynamoDBRecord(record)
 }

 message := fmt.Sprintf("Records processed: %d", len(event.Records))
 return &message, nil
}

func main() {
 lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
 fmt.Println(record.EventID)
 fmt.Println(record.EventName)
 fmt.Printf("%+v\n", record.Change)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do DynamoDB com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
```

```
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new
 GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " +
 GSON.toJson(record.getDynamodb()));
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do DynamoDB com o uso do Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
```

```
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
};

const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Consumindo um evento do DynamoDB com o uso do Lambda. TypeScript

```
export const handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
}
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do DynamoDB com o Lambda usando PHP.

```
<?php
```

```
# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be
 marked as failed
        }
```

```
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Python.

```
import json

def lambda_handler(event, context):
    print(json.dumps(event, indent=2))

    for record in event['Records']:
        log_dynamodb_record(record)

def log_dynamodb_record(record):
    print(record['eventID'])
    print(record['eventName'])
    print(f"DynamoDB Record: {json.dumps(record['dynamodb'])}")
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<Event>) ->Result<(), Error> {

    let records = &event.payload.records;
    tracing::info!("event payload: {:?}",records);
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_dynamo_dbrecord(record);
    }

    tracing::info!("Dynamo db records processed");

    // Prepare the response
    Ok(())

}

fn log_dynamo_dbrecord(record: &EventRecord)-> Result<(), Error>{
    tracing::info!("EventId: {}", record.event_id);
    tracing::info!("EventName: {}", record.event_name);
    tracing::info!("DynamoDB Record: {:?}", record.change );
    Ok(())
```

```
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

Os exemplos de código a seguir mostram como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
```

```
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace AWSLambda_DDB;

public class Function
{
    public StreamsEventResponse FunctionHandler(DynamoDBEvent dynamoEvent,
 ILambdaContext context)

    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>();
        StreamsEventResponse streamsEventResponse = new StreamsEventResponse();

        foreach (var record in dynamoEvent.Records)
        {
            try
            {
                var sequenceNumber = record.Dynamodb.SequenceNumber;
                context.Logger.LogInformation(sequenceNumber);
            }
            catch (Exception ex)
            {
                context.Logger.LogError(ex.Message);
                batchItemFailures.Add(new StreamsEventResponse.BatchItemFailure()
 { ItemIdentifier = record.Dynamodb.SequenceNumber });
            }
        }

        if (batchItemFailures.Count > 0)
        {
            streamsEventResponse.BatchItemFailures = batchItemFailures;
        }

        context.Logger.LogInformation("Stream processing complete.");
        return streamsEventResponse;
```

```
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
 ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
 BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent)
 (*BatchResult, error) {
 var batchItemFailures []BatchItemFailure
 curRecordSequenceNumber := ""

 for _, record := range event.Records {
  // Process your record
  curRecordSequenceNumber = record.Change.SequenceNumber
 }
```

```
 if curRecordSequenceNumber != "" {
  batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
 curRecordSequenceNumber})
 }

 batchResult := BatchResult{
  BatchItemFailures: batchItemFailures,
 }

 return &batchResult, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;
```

```
public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context
 context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatando falhas de itens em lote do DynamoDB com o uso do Lambda. JavaScript

```
export const handler = async (event) => {
  const records = event.Records;
  let curRecordSequenceNumber = "";

  for (const record of records) {
    try {
      // Process your record
      curRecordSequenceNumber = record.dynamodb.SequenceNumber;
    } catch (e) {
      // Return failed record's sequence number
      return { batchItemFailures: [{ itemIdentifier:
 curRecordSequenceNumber }] };
    }
  }

  return { batchItemFailures: [] };
};
```

Relatando falhas de itens em lote do DynamoDB com o uso do Lambda. TypeScript

```
import {
  DynamoDBBatchResponse,
  DynamoDBBatchItemFailure,
  DynamoDBStreamEvent,
} from "aws-lambda";

export const handler = async (
  event: DynamoDBStreamEvent
): Promise<DynamoDBBatchResponse> => {
```

```
  const batchItemFailures: DynamoDBBatchItemFailure[] = [];
  let curRecordSequenceNumber;

  for (const record of event.Records) {
    curRecordSequenceNumber = record.dynamodb?.SequenceNumber;

    if (curRecordSequenceNumber) {
      batchItemFailures.push({
        itemIdentifier: curRecordSequenceNumber,
      });
    }
  }

  return { batchItemFailures: batchItemFailures };
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do DynamoDB com o Lambda usando PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
```

```
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
```

```
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["dynamodb"]["SequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Rust.

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord, StreamRecord},
    streams::{DynamoDbBatchItemFailure, DynamoDbEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Process the stream record
fn process_record(record: &EventRecord) -> Result<(), Error> {
    let stream_record: &StreamRecord = &record.change;

    // process your stream record here...
    tracing::info!("Data: {:?}", stream_record);

    Ok(())
}

/// Main Lambda handler here...
async fn function_handler(event: LambdaEvent<Event>) ->
 Result<DynamoDbEventResponse, Error> {
    let mut response = DynamoDbEventResponse {
        batch_item_failures: vec![],
    };

    let records = &event.payload.records;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in records {
        tracing::info!("EventId: {}", record.event_id);

        // Couldn't find a sequence number
        if record.change.sequence_number.is_none() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: Some("".to_string()),
            });
            return Ok(response);
        }

        // Process your record here...
```

```
        if process_record(record).is_err() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: record.change.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
            return Ok(response);
        }
    }

    tracing::info!("Successfully processed {} record(s)", records.len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

AWS contribuições da comunidade para o DynamoDB

AWS as contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes em todo o mundo AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Exemplos

- Compilar e testar uma aplicação com tecnologia sem servidor

Compilar e testar uma aplicação com tecnologia sem servidor

Os exemplos de código a seguir mostram como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

.NET

SDK para .NET

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK .NET.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Go

SDK para Go V2

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Go.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Java

SDK para Java 2.x

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Java.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Rust

SDK para Rust

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Rust.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos de código para o Amazon EBS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Elastic Block Store com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário do Amazon EBS: mais informações sobre o Amazon EBS.
- Referência da API Amazon EBS: detalhes sobre todas as ações disponíveis do Amazon EBS.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon EBS AWS SDKs
  - Ações para o Amazon EBS usando AWS SDKs
    - Use CompleteSnapshotcom um AWS SDK
    - Use PutSnapshotBlockcom um AWS SDK
    - Use StartSnapshotcom um AWS SDK

Exemplos básicos para o uso do Amazon EBS AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Elastic Block Store com AWS SDKs.

Exemplos

- Ações para o Amazon EBS usando AWS SDKs
  - Use CompleteSnapshotcom um AWS SDK
  - Use PutSnapshotBlockcom um AWS SDK
  - Use StartSnapshotcom um AWS SDK

Ações para o Amazon EBS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon EBS com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Elastic Block Store.

Exemplos

- Use CompleteSnapshotcom um AWS SDK
- Use PutSnapshotBlockcom um AWS SDK
- Use StartSnapshotcom um AWS SDK

Use **`CompleteSnapshot`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CompleteSnapshot`.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn finish(client: &Client, id: &str) -> Result<(), Error> {
    client
        .complete_snapshot()
        .changed_blocks_count(2)
        .snapshot_id(id)
        .send()
        .await?;

    println!("Snapshot ID {}", id);
    println!("The state is 'completed' when all of the modified blocks have been
 transferred to Amazon S3.");
    println!("Use the get-snapshot-state code example to get the state of the
 snapshot.");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CompleteSnapshotreferência da API AWS SDK for Rust.

Use **`PutSnapshotBlock`**com um AWS SDK

O código de exemplo a seguir mostra como usar `PutSnapshotBlock`.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn add_block(
    client: &Client,
    id: &str,
    idx: usize,
    block: Vec<u8>,
    checksum: &str,
) -> Result<(), Error> {
    client
        .put_snapshot_block()
        .snapshot_id(id)
        .block_index(idx as i32)
        .block_data(ByteStream::from(block))
        .checksum(checksum)
        .checksum_algorithm(ChecksumAlgorithm::ChecksumAlgorithmSha256)
        .data_length(EBS_BLOCK_SIZE as i32)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a PutSnapshotBlockreferência da API AWS SDK for Rust.

Use **`StartSnapshot`**com um AWS SDK

O código de exemplo a seguir mostra como usar `StartSnapshot`.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn start(client: &Client, description: &str) -> Result<String, Error> {
    let snapshot = client
        .start_snapshot()
        .description(description)
        .encrypted(false)
        .volume_size(1)
        .send()
        .await?;

    Ok(snapshot.snapshot_id.unwrap())
}
```

- Para obter detalhes da API, consulte a StartSnapshotreferência da API AWS SDK for Rust.

Exemplos de código para o Amazon EC2 usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Elastic Compute Cloud com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon EC2: mais informações sobre o Amazon EC2.
- Referência da API Amazon EC2: detalhes sobre todas as ações disponíveis do Amazon EC2.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon EC2 usando AWS SDKs
  - Olá, Amazon EC2
  - Aprenda as noções básicas do Amazon EC2 com um AWS SDK
  - Ações para o Amazon EC2 usando AWS SDKs
    - Usar AcceptVpcPeeringConnection com uma CLI
    - Use AllocateAddresscom um AWS SDK ou CLI
    - Usar AllocateHosts com uma CLI
    - Usar AssignPrivateIpAddresses com uma CLI
    - Use AssociateAddresscom um AWS SDK ou CLI
    - Usar AssociateDhcpOptions com uma CLI
    - Usar AssociateRouteTable com uma CLI
    - Usar AttachInternetGateway com uma CLI
    - Usar AttachNetworkInterface com uma CLI
    - Usar AttachVolume com uma CLI
    - Usar AttachVpnGateway com uma CLI
    - Usar AuthorizeSecurityGroupEgress com uma CLI
    - Use AuthorizeSecurityGroupIngresscom um AWS SDK ou CLI
    - Usar CancelCapacityReservation com uma CLI
    - Usar CancelImportTask com uma CLI

- Usar CancelSpotFleetRequests com uma CLI
- Usar CancelSpotInstanceRequests com uma CLI
- Usar ConfirmProductInstance com uma CLI
- Usar CopyImage com uma CLI
- Usar CopySnapshot com uma CLI
- Usar CreateCapacityReservation com uma CLI
- Usar CreateCustomerGateway com uma CLI
- Usar CreateDhcpOptions com uma CLI
- Usar CreateFlowLogs com uma CLI
- Usar CreateImage com uma CLI
- Usar CreateInstanceExportTask com uma CLI
- Usar CreateInternetGateway com uma CLI
- Use CreateKeyPaircom um AWS SDK ou CLI
- Use CreateLaunchTemplatecom um AWS SDK ou CLI
- Usar CreateNetworkAcl com uma CLI
- Usar CreateNetworkAclEntry com uma CLI
- Usar CreateNetworkInterface com uma CLI
- Usar CreatePlacementGroup com uma CLI
- Usar CreateRoute com uma CLI
- Use CreateRouteTablecom um AWS SDK ou CLI
- Use CreateSecurityGroupcom um AWS SDK ou CLI
- Usar CreateSnapshot com uma CLI
- Usar CreateSpotDatafeedSubscription com uma CLI
- Use CreateSubnetcom um AWS SDK ou CLI
- Use CreateTagscom um AWS SDK ou CLI
- Usar CreateVolume com uma CLI
- Use CreateVpccom um AWS SDK ou CLI
- Use CreateVpcEndpointcom um AWS SDK ou CLI
- Usar CreateVpnConnection com uma CLI

- Usar CreateVpnConnectionRoute com uma CLI

- Usar CreateVpnGateway com uma CLI
- Usar DeleteCustomerGateway com uma CLI
- Usar DeleteDhcpOptions com uma CLI
- Usar DeleteFlowLogs com uma CLI
- Usar DeleteInternetGateway com uma CLI
- Use DeleteKeyPaircom um AWS SDK ou CLI
- Use DeleteLaunchTemplatecom um AWS SDK ou CLI
- Usar DeleteNetworkAcl com uma CLI
- Usar DeleteNetworkAclEntry com uma CLI
- Usar DeleteNetworkInterface com uma CLI
- Usar DeletePlacementGroup com uma CLI
- Usar DeleteRoute com uma CLI
- Usar DeleteRouteTable com uma CLI
- Use DeleteSecurityGroupcom um AWS SDK ou CLI
- Use DeleteSnapshotcom um AWS SDK ou CLI
- Usar DeleteSpotDatafeedSubscription com uma CLI
- Usar DeleteSubnet com uma CLI
- Usar DeleteTags com uma CLI
- Usar DeleteVolume com uma CLI
- Use DeleteVpccom um AWS SDK ou CLI
- Use DeleteVpcEndpointscom um AWS SDK ou CLI
- Usar DeleteVpnConnection com uma CLI
- Usar DeleteVpnConnectionRoute com uma CLI
- Usar DeleteVpnGateway com uma CLI
- Usar DeregisterImage com uma CLI
- Usar DescribeAccountAttributes com uma CLI
- Use DescribeAddressescom um AWS SDK ou CLI
- Use DescribeAvailabilityZonescom um AWS SDK ou CLI
- Usar DescribeBundleTasks com uma CLI

- Usar DescribeCapacityReservations com uma CLI

- Usar DescribeCustomerGateways com uma CLI
- Usar DescribeDhcpOptions com uma CLI
- Usar DescribeFlowLogs com uma CLI
- Usar DescribeHostReservationOfferings com uma CLI
- Usar DescribeHosts com uma CLI
- Use DescribeIamInstanceProfileAssociationscom um AWS SDK ou CLI
- Usar DescribeIdFormat com uma CLI
- Usar DescribeIdentityIdFormat com uma CLI
- Usar DescribeImageAttribute com uma CLI
- Use DescribeImagescom um AWS SDK ou CLI
- Usar DescribeImportImageTasks com uma CLI
- Usar DescribeImportSnapshotTasks com uma CLI
- Usar DescribeInstanceAttribute com uma CLI
- Use DescribeInstanceStatuscom um AWS SDK ou CLI
- Use DescribeInstanceTypescom um AWS SDK ou CLI
- Use DescribeInstancescom um AWS SDK ou CLI
- Usar DescribeInternetGateways com uma CLI
- Use DescribeKeyPairscom um AWS SDK ou CLI
- Usar DescribeNetworkAcls com uma CLI
- Usar DescribeNetworkInterfaceAttribute com uma CLI
- Usar DescribeNetworkInterfaces com uma CLI
- Usar DescribePlacementGroups com uma CLI
- Usar DescribePrefixLists com uma CLI
- Use DescribeRegionscom um AWS SDK ou CLI
- Use DescribeRouteTablescom um AWS SDK ou CLI
- Usar DescribeScheduledInstanceAvailability com uma CLI
- Usar DescribeScheduledInstances com uma CLI
- Use DescribeSecurityGroupscom um AWS SDK ou CLI
- Usar DescribeSnapshotAttribute com uma CLI

- Use DescribeSnapshotscom um AWS SDK ou CLI

- Usar DescribeSpotDatafeedSubscription com uma CLI
- Usar DescribeSpotFleetInstances com uma CLI
- Usar DescribeSpotFleetRequestHistory com uma CLI
- Usar DescribeSpotFleetRequests com uma CLI
- Usar DescribeSpotInstanceRequests com uma CLI
- Usar DescribeSpotPriceHistory com uma CLI
- Use DescribeSubnetscom um AWS SDK ou CLI
- Usar DescribeTags com uma CLI
- Usar DescribeVolumeAttribute com uma CLI
- Usar DescribeVolumeStatus com uma CLI
- Usar DescribeVolumes com uma CLI
- Usar DescribeVpcAttribute com uma CLI
- Usar DescribeVpcClassicLink com uma CLI
- Usar DescribeVpcClassicLinkDnsSupport com uma CLI
- Usar DescribeVpcEndpointServices com uma CLI
- Usar DescribeVpcEndpoints com uma CLI
- Use DescribeVpcscom um AWS SDK ou CLI
- Usar DescribeVpnConnections com uma CLI
- Usar DescribeVpnGateways com uma CLI
- Usar DetachInternetGateway com uma CLI
- Usar DetachNetworkInterface com uma CLI
- Usar DetachVolume com uma CLI
- Usar DetachVpnGateway com uma CLI
- Usar DisableVgwRoutePropagation com uma CLI
- Usar DisableVpcClassicLink com uma CLI
- Usar DisableVpcClassicLinkDnsSupport com uma CLI
- Use DisassociateAddresscom um AWS SDK ou CLI
- Usar DisassociateRouteTable com uma CLI
- Usar EnableVgwRoutePropagation com uma CLI

- Usar EnableVolumeIo com uma CLI

- Usar EnableVpcClassicLink com uma CLI
- Usar EnableVpcClassicLinkDnsSupport com uma CLI
- Usar GetConsoleOutput com uma CLI
- Usar GetHostReservationPurchasePreview com uma CLI
- Use GetPasswordDatacom um AWS SDK ou CLI
- Usar ImportImage com uma CLI
- Usar ImportKeyPair com uma CLI
- Usar ImportSnapshot com uma CLI
- Usar ModifyCapacityReservation com uma CLI
- Usar ModifyHosts com uma CLI
- Usar ModifyIdFormat com uma CLI
- Usar ModifyImageAttribute com uma CLI
- Usar ModifyInstanceAttribute com uma CLI
- Usar ModifyInstanceCreditSpecification com uma CLI
- Usar ModifyNetworkInterfaceAttribute com uma CLI
- Usar ModifyReservedInstances com uma CLI
- Usar ModifySnapshotAttribute com uma CLI
- Usar ModifySpotFleetRequest com uma CLI
- Usar ModifySubnetAttribute com uma CLI
- Usar ModifyVolumeAttribute com uma CLI
- Usar ModifyVpcAttribute com uma CLI
- Use MonitorInstancescom um AWS SDK ou CLI
- Usar MoveAddressToVpc com uma CLI
- Usar PurchaseHostReservation com uma CLI
- Usar PurchaseScheduledInstances com uma CLI
- Use RebootInstancescom um AWS SDK ou CLI
- Usar RegisterImage com uma CLI
- Usar RejectVpcPeeringConnection com uma CLI
- Use ReleaseAddresscom um AWS SDK ou CLI

- Usar ReleaseHosts com uma CLI

- Use ReplaceIamInstanceProfileAssociationcom um AWS SDK ou CLI
- Usar ReplaceNetworkAclAssociation com uma CLI
- Usar ReplaceNetworkAclEntry com uma CLI
- Usar ReplaceRoute com uma CLI
- Usar ReplaceRouteTableAssociation com uma CLI
- Usar ReportInstanceStatus com uma CLI
- Usar RequestSpotFleet com uma CLI
- Usar RequestSpotInstances com uma CLI
- Usar ResetImageAttribute com uma CLI
- Usar ResetInstanceAttribute com uma CLI
- Usar ResetNetworkInterfaceAttribute com uma CLI
- Usar ResetSnapshotAttribute com uma CLI
- Usar RevokeSecurityGroupEgress com uma CLI
- Usar RevokeSecurityGroupIngress com uma CLI
- Use RunInstancescom um AWS SDK ou CLI
- Usar RunScheduledInstances com uma CLI
- Use StartInstancescom um AWS SDK ou CLI
- Use StopInstancescom um AWS SDK ou CLI
- Use TerminateInstancescom um AWS SDK ou CLI
- Usar UnassignPrivateIpAddresses com uma CLI
- Use UnmonitorInstancescom um AWS SDK ou CLI
- Usar UpdateSecurityGroupRuleDescriptionsIngress com uma CLI

- Cenários para o Amazon EC2 usando AWS SDKs
  - Crie e gerencie um serviço resiliente usando um AWS SDK
  - Configurar a conectividade do serviço de contêiner
  - Introdução às redes privadas virtuais
  - Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
  - Como criar um serviço de contêiner para instâncias de máquinas virtuais
  - Criação de uma instância de banco de dados relacional

  - Criação e gerenciamento de uma malha de rede de serviços

- Criação e gerenciamento de volumes de armazenamento em bloco
- Comece com conexões de rede dedicadas usando a CLI
- Comece a comprar no mercado de software usando a CLI
- Introdução aos clusters de processamento de big data
- Introdução aos bancos de dados de documentos
- Introdução aos bancos de dados de grafos
- Introdução ao armazenamento em cache em memória
- Introdução ao balanceamento de carga
- Conceitos básicos de streaming gerenciado
- Introdução aos gateways de trânsito de rede
- Introdução à transmissão de transporte de vídeo
- Introdução às máquinas virtuais
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Comece a usar o gerenciamento de endereços IP usando a CLI
- Crie uma rede privada virtual com sub-redes privadas e gateways de conversão de endereços de rede usando a CLI
- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco
- Trabalho com conexões de emparelhamento de rede

Exemplos básicos para o Amazon EC2 usando AWS SDKs

Os exemplos de código a seguir mostram como usar os elementos básicos do Amazon Elastic Compute Cloud com AWS SDKs.

Exemplos

- Olá, Amazon EC2
- Aprenda as noções básicas do Amazon EC2 com um AWS SDK
- Ações para o Amazon EC2 usando AWS SDKs
  - Usar AcceptVpcPeeringConnection com uma CLI
  - Use AllocateAddresscom um AWS SDK ou CLI
  - Usar AllocateHosts com uma CLI

- Usar AssignPrivateIpAddresses com uma CLI
- Use AssociateAddresscom um AWS SDK ou CLI
- Usar AssociateDhcpOptions com uma CLI
- Usar AssociateRouteTable com uma CLI
- Usar AttachInternetGateway com uma CLI
- Usar AttachNetworkInterface com uma CLI
- Usar AttachVolume com uma CLI
- Usar AttachVpnGateway com uma CLI
- Usar AuthorizeSecurityGroupEgress com uma CLI
- Use AuthorizeSecurityGroupIngresscom um AWS SDK ou CLI
- Usar CancelCapacityReservation com uma CLI
- Usar CancelImportTask com uma CLI
- Usar CancelSpotFleetRequests com uma CLI
- Usar CancelSpotInstanceRequests com uma CLI
- Usar ConfirmProductInstance com uma CLI
- Usar CopyImage com uma CLI
- Usar CopySnapshot com uma CLI
- Usar CreateCapacityReservation com uma CLI
- Usar CreateCustomerGateway com uma CLI
- Usar CreateDhcpOptions com uma CLI
- Usar CreateFlowLogs com uma CLI
- Usar CreateImage com uma CLI
- Usar CreateInstanceExportTask com uma CLI
- Usar CreateInternetGateway com uma CLI
- Use CreateKeyPaircom um AWS SDK ou CLI
- Use CreateLaunchTemplatecom um AWS SDK ou CLI
- Usar CreateNetworkAcl com uma CLI
- Usar CreateNetworkAclEntry com uma CLI
- Usar CreateNetworkInterface com uma CLI

- Usar CreatePlacementGroup com uma CLI

- Usar CreateRoute com uma CLI
- Use CreateRouteTablecom um AWS SDK ou CLI
- Use CreateSecurityGroupcom um AWS SDK ou CLI
- Usar CreateSnapshot com uma CLI
- Usar CreateSpotDatafeedSubscription com uma CLI
- Use CreateSubnetcom um AWS SDK ou CLI
- Use CreateTagscom um AWS SDK ou CLI
- Usar CreateVolume com uma CLI
- Use CreateVpccom um AWS SDK ou CLI
- Use CreateVpcEndpointcom um AWS SDK ou CLI
- Usar CreateVpnConnection com uma CLI
- Usar CreateVpnConnectionRoute com uma CLI
- Usar CreateVpnGateway com uma CLI
- Usar DeleteCustomerGateway com uma CLI
- Usar DeleteDhcpOptions com uma CLI
- Usar DeleteFlowLogs com uma CLI
- Usar DeleteInternetGateway com uma CLI
- Use DeleteKeyPaircom um AWS SDK ou CLI
- Use DeleteLaunchTemplatecom um AWS SDK ou CLI
- Usar DeleteNetworkAcl com uma CLI
- Usar DeleteNetworkAclEntry com uma CLI
- Usar DeleteNetworkInterface com uma CLI
- Usar DeletePlacementGroup com uma CLI
- Usar DeleteRoute com uma CLI
- Usar DeleteRouteTable com uma CLI
- Use DeleteSecurityGroupcom um AWS SDK ou CLI
- Use DeleteSnapshotcom um AWS SDK ou CLI
- Usar DeleteSpotDatafeedSubscription com uma CLI
- Usar DeleteSubnet com uma CLI

- Usar DeleteTags com uma CLI

- Usar DeleteVolume com uma CLI
- Use DeleteVpccom um AWS SDK ou CLI
- Use DeleteVpcEndpointscom um AWS SDK ou CLI
- Usar DeleteVpnConnection com uma CLI
- Usar DeleteVpnConnectionRoute com uma CLI
- Usar DeleteVpnGateway com uma CLI
- Usar DeregisterImage com uma CLI
- Usar DescribeAccountAttributes com uma CLI
- Use DescribeAddressescom um AWS SDK ou CLI
- Use DescribeAvailabilityZonescom um AWS SDK ou CLI
- Usar DescribeBundleTasks com uma CLI
- Usar DescribeCapacityReservations com uma CLI
- Usar DescribeCustomerGateways com uma CLI
- Usar DescribeDhcpOptions com uma CLI
- Usar DescribeFlowLogs com uma CLI
- Usar DescribeHostReservationOfferings com uma CLI
- Usar DescribeHosts com uma CLI
- Use DescribeIamInstanceProfileAssociationscom um AWS SDK ou CLI
- Usar DescribeIdFormat com uma CLI
- Usar DescribeIdentityIdFormat com uma CLI
- Usar DescribeImageAttribute com uma CLI
- Use DescribeImagescom um AWS SDK ou CLI
- Usar DescribeImportImageTasks com uma CLI
- Usar DescribeImportSnapshotTasks com uma CLI
- Usar DescribeInstanceAttribute com uma CLI
- Use DescribeInstanceStatuscom um AWS SDK ou CLI
- Use DescribeInstanceTypescom um AWS SDK ou CLI
- Use DescribeInstancescom um AWS SDK ou CLI
- Usar DescribeInternetGateways com uma CLI

- Use DescribeKeyPairscom um AWS SDK ou CLI

- Usar DescribeNetworkAcls com uma CLI
- Usar DescribeNetworkInterfaceAttribute com uma CLI
- Usar DescribeNetworkInterfaces com uma CLI
- Usar DescribePlacementGroups com uma CLI
- Usar DescribePrefixLists com uma CLI
- Use DescribeRegionscom um AWS SDK ou CLI
- Use DescribeRouteTablescom um AWS SDK ou CLI
- Usar DescribeScheduledInstanceAvailability com uma CLI
- Usar DescribeScheduledInstances com uma CLI
- Use DescribeSecurityGroupscom um AWS SDK ou CLI
- Usar DescribeSnapshotAttribute com uma CLI
- Use DescribeSnapshotscom um AWS SDK ou CLI
- Usar DescribeSpotDatafeedSubscription com uma CLI
- Usar DescribeSpotFleetInstances com uma CLI
- Usar DescribeSpotFleetRequestHistory com uma CLI
- Usar DescribeSpotFleetRequests com uma CLI
- Usar DescribeSpotInstanceRequests com uma CLI
- Usar DescribeSpotPriceHistory com uma CLI
- Use DescribeSubnetscom um AWS SDK ou CLI
- Usar DescribeTags com uma CLI
- Usar DescribeVolumeAttribute com uma CLI
- Usar DescribeVolumeStatus com uma CLI
- Usar DescribeVolumes com uma CLI
- Usar DescribeVpcAttribute com uma CLI
- Usar DescribeVpcClassicLink com uma CLI
- Usar DescribeVpcClassicLinkDnsSupport com uma CLI
- Usar DescribeVpcEndpointServices com uma CLI
- Usar DescribeVpcEndpoints com uma CLI
- Use DescribeVpcscom um AWS SDK ou CLI

- Usar DescribeVpnConnections com uma CLI

- Usar DescribeVpnGateways com uma CLI
- Usar DetachInternetGateway com uma CLI
- Usar DetachNetworkInterface com uma CLI
- Usar DetachVolume com uma CLI
- Usar DetachVpnGateway com uma CLI
- Usar DisableVgwRoutePropagation com uma CLI
- Usar DisableVpcClassicLink com uma CLI
- Usar DisableVpcClassicLinkDnsSupport com uma CLI
- Use DisassociateAddresscom um AWS SDK ou CLI
- Usar DisassociateRouteTable com uma CLI
- Usar EnableVgwRoutePropagation com uma CLI
- Usar EnableVolumeIo com uma CLI
- Usar EnableVpcClassicLink com uma CLI
- Usar EnableVpcClassicLinkDnsSupport com uma CLI
- Usar GetConsoleOutput com uma CLI
- Usar GetHostReservationPurchasePreview com uma CLI
- Use GetPasswordDatacom um AWS SDK ou CLI
- Usar ImportImage com uma CLI
- Usar ImportKeyPair com uma CLI
- Usar ImportSnapshot com uma CLI
- Usar ModifyCapacityReservation com uma CLI
- Usar ModifyHosts com uma CLI
- Usar ModifyIdFormat com uma CLI
- Usar ModifyImageAttribute com uma CLI
- Usar ModifyInstanceAttribute com uma CLI
- Usar ModifyInstanceCreditSpecification com uma CLI
- Usar ModifyNetworkInterfaceAttribute com uma CLI
- Usar ModifyReservedInstances com uma CLI
- Usar ModifySnapshotAttribute com uma CLI

- Usar ModifySpotFleetRequest com uma CLI

- Usar ModifySubnetAttribute com uma CLI
- Usar ModifyVolumeAttribute com uma CLI
- Usar ModifyVpcAttribute com uma CLI
- Use MonitorInstancescom um AWS SDK ou CLI
- Usar MoveAddressToVpc com uma CLI
- Usar PurchaseHostReservation com uma CLI
- Usar PurchaseScheduledInstances com uma CLI
- Use RebootInstancescom um AWS SDK ou CLI
- Usar RegisterImage com uma CLI
- Usar RejectVpcPeeringConnection com uma CLI
- Use ReleaseAddresscom um AWS SDK ou CLI
- Usar ReleaseHosts com uma CLI
- Use ReplaceIamInstanceProfileAssociationcom um AWS SDK ou CLI
- Usar ReplaceNetworkAclAssociation com uma CLI
- Usar ReplaceNetworkAclEntry com uma CLI
- Usar ReplaceRoute com uma CLI
- Usar ReplaceRouteTableAssociation com uma CLI
- Usar ReportInstanceStatus com uma CLI
- Usar RequestSpotFleet com uma CLI
- Usar RequestSpotInstances com uma CLI
- Usar ResetImageAttribute com uma CLI
- Usar ResetInstanceAttribute com uma CLI
- Usar ResetNetworkInterfaceAttribute com uma CLI
- Usar ResetSnapshotAttribute com uma CLI
- Usar RevokeSecurityGroupEgress com uma CLI
- Usar RevokeSecurityGroupIngress com uma CLI
- Use RunInstancescom um AWS SDK ou CLI
- Usar RunScheduledInstances com uma CLI
- Use StartInstancescom um AWS SDK ou CLI

- Use StopInstancescom um AWS SDK ou CLI

- Use TerminateInstancescom um AWS SDK ou CLI
- Usar UnassignPrivateIpAddresses com uma CLI
- Use UnmonitorInstancescom um AWS SDK ou CLI
- Usar UpdateSecurityGroupRuleDescriptionsIngress com uma CLI

Olá, Amazon EC2

Os exemplos de código a seguir mostram como começar a usar o Amazon EC2.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace EC2Actions;

public class HelloEc2
{
    /// <summary>
    /// HelloEc2 lists the existing security groups for the default users.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Elastic Compute Cloud (Amazon
 EC2).
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                .AddTransient<EC2Wrapper>()
            )
            .Build();
```

```
        // Now the client is available for injection.
        var ec2Client = host.Services.GetRequiredService<IAmazonEC2>();

        try
        {
            // Retrieve information for up to 10 Amazon EC2 security groups.
            var request = new DescribeSecurityGroupsRequest { MaxResults = 10 };
            var securityGroups = new List<SecurityGroup>();

            var paginatorForSecurityGroups =
                ec2Client.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
 paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            // Now print the security groups returned by the call to
            // DescribeSecurityGroupsAsync.
            Console.WriteLine("Welcome to the EC2 Hello Service example. " +
                              "\nLet's list your Security Groups:");
            securityGroups.ForEach(group =>
            {
                Console.WriteLine(
                    $"Security group: {group.GroupName} ID: {group.GroupId}");
            });
        }
        catch (AmazonEC2Exception ex)
        {
            Console.WriteLine($"An Amazon EC2 service error occurred while
 listing security groups. {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing security groups.
 {ex.Message}");
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS ec2)

# Set this project's name.
project("hello_ec2")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_ec2.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo fonte hello_ec2.cpp.

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <iomanip>
#include <iostream>

/*
 *  A "Hello EC2" starter application which initializes an Amazon Elastic Compute
 Cloud (Amazon EC2) client and describes
 *  the Amazon EC2 instances.
 *
 *  main function
 *
 *  Usage: 'hello_ec2'
 *
 */

int main(int argc, char **argv) {
    (void)argc;
    (void)argv;

    Aws::SDKOptions options;
```

```
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::EC2::EC2Client ec2Client(clientConfig);
        Aws::EC2::Model::DescribeInstancesRequest request;
        bool header = false;
        bool done = false;
        while (!done) {
            Aws::EC2::Model::DescribeInstancesOutcome outcome =
 ec2Client.DescribeInstances(request);
            if (outcome.IsSuccess()) {
                if (!header) {
                    std::cout << std::left <<
                              std::setw(48) << "Name" <<
                              std::setw(20) << "ID" <<
                              std::setw(25) << "Ami" <<
                              std::setw(15) << "Type" <<
                              std::setw(15) << "State" <<
                              std::setw(15) << "Monitoring" << std::endl;
                    header = true;
                }

                const std::vector<Aws::EC2::Model::Reservation> &reservations =
                        outcome.GetResult().GetReservations();

                for (const auto &reservation: reservations) {
                    const std::vector<Aws::EC2::Model::Instance> &instances =
                            reservation.GetInstances();
                    for (const auto &instance: instances) {
                        Aws::String instanceStateString =

 Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                        instance.GetState().GetName());

                        Aws::String typeString =

 Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                                        instance.GetInstanceType());
```

```
                        Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                                        instance.GetMonitoring().GetState());
                        Aws::String name = "Unknown";

                        const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                        auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                                                     [](const
Aws::EC2::Model::Tag &tag) {
                                                         return tag.GetKey() ==
"Name";
                                                     });
                        if (nameIter != tags.cend()) {
                            name = nameIter->GetValue();
                        }
                        std::cout <<
                                  std::setw(48) << name <<
                                  std::setw(20) << instance.GetInstanceId() <<
                                  std::setw(25) << instance.GetImageId() <<
                                  std::setw(15) << typeString <<
                                  std::setw(15) << instanceStateString <<
                                  std::setw(15) << monitorString << std::endl;
                    }
                }

                if (!outcome.GetResult().GetNextToken().empty()) {
                    request.SetNextToken(outcome.GetResult().GetNextToken());
                } else {
                    done = true;
                }
            } else {
                std::cerr << "Failed to describe EC2 instances:" <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
```

```
    return result;
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
```

```
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the
name: " + groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

// Call DescribeSecurityGroups and display the result.
export const main = async () => {
  const client = new EC2Client();
  try {
    const { SecurityGroups } = await client.send(
      new DescribeSecurityGroupsCommand({}),
```

```
    );

    const securityGroupList = SecurityGroups.slice(0, 9)
      .map((sg) => ` • ${sg.GroupId}: ${sg.GroupName}`)
      .join("\n");

    console.log(
      "Hello, Amazon EC2! Let's list up to 10 of your security groups:",
    );
    console.log(securityGroupList);
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }
```

```
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSecurityGroupsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def hello_ec2(ec2_client):
    """
    Use the AWS SDK for Python (Boto3) to list the security groups in your
 account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ec2_client: A Boto3 EC2 client. This client provides low-level
                       access to AWS EC2 services.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    try:
        paginator = ec2_client.get_paginator("describe_security_groups")
        response_iterator = paginator.paginate(PaginationConfig={'MaxItems': 10})
 # List only 10 security groups.
        logging.basicConfig(level=logging.INFO) # Enable logging.
        for page in response_iterator:
            for sg in page["SecurityGroups"]:
```

```
                logger.info(f"\t{sg['GroupId']}: {sg['GroupName']}")
    except ClientError as err:
        logger.error("Failed to list security groups.")
        if err.response["Error"]["Code"] == "AccessDeniedException":
            logger.error("You do not have permission to list security groups.")
        raise


if __name__ == "__main__":
    hello_ec2(boto3.client("ec2"))
```

- Para obter detalhes da API, consulte a DescribeSecurityGroupsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'
require 'logger'

# EC2Manager is a class responsible for managing EC2 operations
# such as listing all EC2 instances in the current AWS account.
class EC2Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all EC2 instances in the current AWS account.
  def list_instances
    @logger.info('Listing instances')
```

```
    instances = fetch_instances

    if instances.empty?
      @logger.info('You have no instances')
    else
      print_instances(instances)
    end
  end

  private

  # Fetches all EC2 instances using pagination.
  #
  # @return [Array<Aws::EC2::Types::Instance>] List of EC2 instances.
  def fetch_instances
    paginator = @client.describe_instances
    instances = []

    paginator.each_page do |page|
      page.reservations.each do |reservation|
        reservation.instances.each do |instance|
          instances << instance
        end
      end
    end

    instances
  end

  # Prints details of the given EC2 instances.
  #
  # @param instances [Array<Aws::EC2::Types::Instance>] List of EC2 instances to
 print.
  def print_instances(instances)
    instances.each do |instance|
      @logger.info("Instance ID: #{instance.instance_id}")
      @logger.info("Instance Type: #{instance.instance_type}")
      @logger.info("Public IP: #{instance.public_ip_address}")
      @logger.info("Public DNS Name: #{instance.public_dns_name}")
      @logger.info("\n")
    end
  end
end
```

```
if $PROGRAM_NAME == __FILE__
  ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
  manager = EC2Manager.new(ec2_client)
  manager.list_instances
end
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids:
 Vec<String>) {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
```

```
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSecurityGroupsreferência da API AWS SDK for Rust.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "hello-ec2",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
```

```
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "hello-ec2",
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to perform a simple
// operation using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation

import AWSEC2

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
```

```
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "hello-ec2",
        abstract: """
        Demonstrates a simple operation using Amazon EC2.
        """,
        discussion: """
        An example showing how to make a call to Amazon EC2 using the AWS SDK for
Swift.
        """
    )

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
    /// - Returns: An array of strings giving the names of every security
    ///   group the user is a member of.
    func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
        let pages = ec2Client.describeSecurityGroupsPaginated(
            input: DescribeSecurityGroupsInput()
        )

        var groupNames: [String] = []

        do {
            for try await page in pages {
                guard let groups = page.securityGroups else {
                    print("*** Error: No groups returned.")
                    continue
                }
```

```
                for group in groups {
                    groupNames.append(group.groupName ?? "<unknown>")
                }
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }

        return groupNames
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
 awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let groupNames = await getSecurityGroupNames(ec2Client: ec2Client)

        print("Found \(groupNames.count) security group(s):")

        for group in groupNames {
            print("    \(group)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa referência da API AWS SDK for Swift.

Aprenda as noções básicas do Amazon EC2 com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário em um prompt de comando.

```
/// <summary>
/// Show Amazon Elastic Compute Cloud (Amazon EC2) Basics actions.
/// </summary>
public class EC2Basics
{
    public static ILogger<EC2Basics> _logger = null!;
    public static EC2Wrapper _ec2Wrapper = null!;
    public static SsmWrapper _ssmWrapper = null!;
    public static UiMethods _uiMethods = null!;

    public static string associationId = null!;
    public static string allocationId = null!;
    public static string instanceId = null!;
```

```
    public static string keyPairName = null!;
    public static string groupName = null!;
    public static string tempFileName = null!;
    public static string secGroupId = null!;
    public static bool isInteractive = true;

    /// <summary>
    /// Perform the actions defined for the Amazon EC2 Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 and Amazon Simple Systems
        // Management (Amazon SSM) Service.
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddTransient<EC2Wrapper>()
                    .AddTransient<SsmWrapper>()
            )
            .Build();

        SetUpServices(host);

        var uniqueName = Guid.NewGuid().ToString();
        keyPairName = "mvp-example-key-pair" + uniqueName;
        groupName = "ec2-scenario-group" + uniqueName;
        var groupDescription = "A security group created for the EC2 Basics
scenario.";

        try
        {
            // Start the scenario.
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(isInteractive);

            // Create the key pair.
            _uiMethods.DisplayTitle("Create RSA key pair");
            Console.Write("Let's create an RSA key pair that you can be use to
");
            Console.WriteLine("securely connect to your EC2 instance.");
```

```
            var keyPair = await _ec2Wrapper.CreateKeyPair(keyPairName);

            // Save key pair information to a temporary file.
            tempFileName = _ec2Wrapper.SaveKeyPair(keyPair);

            Console.WriteLine(
                $"Created the key pair: {keyPair.KeyName} and saved it to:
{tempFileName}");
            string? answer = "";
            if (isInteractive)
            {
                do
                {
                    Console.Write("Would you like to list your existing key
pairs? ");
                    answer = Console.ReadLine();
                } while (answer!.ToLower() != "y" && answer.ToLower() != "n");
            }

            if (!isInteractive || answer == "y")
            {
                // List existing key pairs.
                _uiMethods.DisplayTitle("Existing key pairs");

                // Passing an empty string to the DescribeKeyPairs method will
return
                // a list of all existing key pairs.
                var keyPairs = await _ec2Wrapper.DescribeKeyPairs("");
                keyPairs.ForEach(kp =>
                {
                    Console.WriteLine(
                        $"{kp.KeyName} created at: {kp.CreateTime} Fingerprint:
{kp.KeyFingerprint}");
                });
            }

            _uiMethods.PressEnter(isInteractive);

            // Create the security group.
            Console.WriteLine(
                "Let's create a security group to manage access to your
instance.");
            secGroupId = await _ec2Wrapper.CreateSecurityGroup(groupName,
groupDescription);
```

```
            Console.WriteLine(
                "Let's add rules to allow all HTTP and HTTPS inbound traffic and
to allow SSH only from your current IP address.");

            _uiMethods.DisplayTitle("Security group information");
            var secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);

            Console.WriteLine($"Created security group {groupName} in your
default VPC.");
            secGroups.ForEach(group =>
            {
                _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
            });
            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we'll authorize the security group we just created so that
it can");
            Console.WriteLine("access the EC2 instances you create.");
            await _ec2Wrapper.AuthorizeSecurityGroupIngress(groupName);

            secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);
            Console.WriteLine($"Now let's look at the permissions again.");
            secGroups.ForEach(group =>
            {
                _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
            });
            _uiMethods.PressEnter(isInteractive);

            // Get list of available Amazon Linux 2 Amazon Machine Images (AMIs).
            var parameters =
                await _ssmWrapper.GetParametersByPath(
                    "/aws/service/ami-amazon-linux-latest");

            List<string> imageIds = parameters.Select(param =>
param.Value).ToList();

            var images = await _ec2Wrapper.DescribeImages(imageIds);

            var i = 1;
            images.ForEach(image =>
            {
                Console.WriteLine($"\t{i++}\t{image.Description}");
            });
```

```
            int choice = 1;
            bool validNumber = false;
            if (isInteractive)
            {
                do
                {
                    Console.Write("Please select an image: ");
                    var selImage = Console.ReadLine();
                    validNumber = int.TryParse(selImage, out choice);
                } while (!validNumber);
            }

            var selectedImage = images[choice - 1];

            // Display available instance types.
            _uiMethods.DisplayTitle("Instance Types");
            var instanceTypes =
                await
 _ec2Wrapper.DescribeInstanceTypes(selectedImage.Architecture);

            i = 1;
            instanceTypes.ForEach(instanceType =>
            {
                Console.WriteLine($"\t{i++}\t{instanceType.InstanceType}");
            });
            if (isInteractive)
            {
                do
                {
                    Console.Write("Please select an instance type: ");
                    var selImage = Console.ReadLine();
                    validNumber = int.TryParse(selImage, out choice);
                } while (!validNumber);
            }

            var selectedInstanceType = instanceTypes[choice - 1].InstanceType;

            // Create an EC2 instance.
            _uiMethods.DisplayTitle("Creating an EC2 Instance");
            instanceId = await _ec2Wrapper.RunInstances(selectedImage.ImageId,
                selectedInstanceType, keyPairName, secGroupId);

            _uiMethods.PressEnter(isInteractive);
```

```
            var instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine(
                "\nYou can use SSH to connect to your instance. For example:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we'll stop the instance and then start it again to see
what's changed.");

            await _ec2Wrapper.StopInstances(instanceId);

            Console.WriteLine("Now let's start it up again.");
            await _ec2Wrapper.StartInstances(instanceId);

            Console.WriteLine("\nLet's see what changed.");

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine("\nNotice the change in the SSH information:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we will stop the instance again. Then we will create and
associate an");
            Console.WriteLine("Elastic IP address to use with our instance.");

            await _ec2Wrapper.StopInstances(instanceId);
            _uiMethods.PressEnter(isInteractive);

            _uiMethods.DisplayTitle("Allocate Elastic IP address");
            Console.WriteLine(
```

```
                "You can allocate an Elastic IP address and associate it
with your instance\nto keep a consistent IP address even when your instance
restarts.");
            var allocationResponse = await _ec2Wrapper.AllocateAddress();
            allocationId = allocationResponse.AllocationId;
            Console.WriteLine(
                "Now we will associate the Elastic IP address with our
instance.");
            associationId = await _ec2Wrapper.AssociateAddress(allocationId,
instanceId);

            // Start the instance again.
            Console.WriteLine("Now let's start the instance again.");
            await _ec2Wrapper.StartInstances(instanceId);

            Console.WriteLine("\nLet's see what changed.");

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("Instance information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine("\nHere is the SSH information:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            Console.WriteLine("Let's stop and start the instance again.");
            _uiMethods.PressEnter(isInteractive);

            await _ec2Wrapper.StopInstances(instanceId);

            Console.WriteLine("\nThe instance has stopped.");

            Console.WriteLine("Now let's start it up again.");
            await _ec2Wrapper.StartInstances(instanceId);

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);
            Console.WriteLine("Note that the IP address did not change this
time.");
            _uiMethods.PressEnter(isInteractive);

            await Cleanup();
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, starting
cleanup.");
            await Cleanup();
        }

        _uiMethods.DisplayTitle("EC2 Basics Scenario completed.");
        _uiMethods.PressEnter(isInteractive);
    }

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host"></param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<EC2Basics>(loggerFactory);

        // Now the client is available for injection.
        _ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        _ssmWrapper = host.Services.GetRequiredService<SsmWrapper>();
        _uiMethods = new UiMethods();
    }

    /// <summary>
    /// Clean up any resources from the scenario.
    /// </summary>
    /// <returns></returns>
    public static async Task Cleanup()
    {
        _uiMethods.DisplayTitle("Clean up resources");
        Console.WriteLine("Now let's clean up the resources we created.");

        Console.WriteLine("Disassociate the Elastic IP address and release it.");
        // Disassociate the Elastic IP address.
        await _ec2Wrapper.DisassociateIp(associationId);

        // Delete the Elastic IP address.
        await _ec2Wrapper.ReleaseAddress(allocationId);
```

```
        // Terminate the instance.
        Console.WriteLine("Terminating the instance we created.");
        await _ec2Wrapper.TerminateInstances(instanceId);

        // Delete the security group.
        Console.WriteLine($"Deleting the Security Group: {groupName}.");
        await _ec2Wrapper.DeleteSecurityGroup(secGroupId);

        // Delete the RSA key pair.
        Console.WriteLine($"Deleting the key pair: {keyPairName}");
        await _ec2Wrapper.DeleteKeyPair(keyPairName);
        Console.WriteLine("Deleting the temporary file with the key
 information.");
        _ec2Wrapper.DeleteTempFile(tempFileName);
        _uiMethods.PressEnter(isInteractive);
    }
}
```

Defina uma classe que envolva as ações do EC2.

```
/// <summary>
/// Methods of this class perform Amazon Elastic Compute Cloud (Amazon EC2).
/// </summary>
public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEC2;
    private readonly ILogger<EC2Wrapper> _logger;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EC2 client.</param>
    /// <param name="logger">The injected logger.</param>
    public EC2Wrapper(IAmazonEC2 amazonService, ILogger<EC2Wrapper> logger)
    {
        _amazonEC2 = amazonService;
        _logger = logger;
    }

    /// <summary>
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
```

```
    // instance. By using an Elastic IP address, you can keep the public IP
 address
    // constant even when you restart the associated instance.
    /// </summary>
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation
 ID {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-
ip-addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
 exceeded. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Associates an Elastic IP address with an instance. When this association
 is
    /// created, the Elastic IP's public IP address is immediately used as the
 public
    /// IP address of the associated instance.
    /// </summary>
```

```
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
 instanceId)
    {
        try
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to associate address.
 {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer ingress to EC2 instances associated
    /// with the virtual private cloud (VPC) security group.
    /// </summary>
```

```
    /// <param name="groupName">The name of the security group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
    {
        try
        {
            // Get the IP address for the local computer.
            var ipAddress = await GetIpAddress();
            Console.WriteLine($"Your IP address is: {ipAddress}");
            var ipRanges =
                new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
            var permission = new IpPermission
            {
                Ipv4Ranges = ipRanges,
                IpProtocol = "tcp",
                FromPort = 22,
                ToPort = 22
            };
            var permissions = new List<IpPermission> { permission };
            var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
                new AuthorizeSecurityGroupIngressRequest(groupName,
permissions));
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
            {
                _logger.LogError(
                    $"The ingress rule already exists. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while authorizing ingress.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer for ingress to
```

```
    /// the Amazon EC2 SecurityGroup.
    /// </summary>
    /// <returns>The IPv4 address of the computer running the scenario.</returns>
    private static async Task<string> GetIpAddress()
    {
        var httpClient = new HttpClient();
        var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

        // The IP address is returned with a new line
        // character on the end. Trim off the whitespace and
        // return the value to the caller.
        return ipString.Trim();
    }

    /// <summary>
    /// Create an Amazon EC2 key pair with a specified name.
    /// </summary>
    /// <param name="keyPairName">The name for the new key pair.</param>
    /// <returns>The Amazon EC2 key pair created.</returns>
    public async Task<KeyPair?> CreateKeyPair(string keyPairName)
    {
        try
        {
            var request = new CreateKeyPairRequest { KeyName = keyPairName, };

            var response = await _amazonEC2.CreateKeyPairAsync(request);

            var kp = response.KeyPair;
            // Return the key pair so it can be saved if needed.

            // Wait until the key pair exists.
            int retries = 5;
            while (retries-- > 0)
            {
                Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
                var keyPairs = await DescribeKeyPairs(keyPairName);
                if (keyPairs.Any())
                {
                    return kp;
                }

                Thread.Sleep(5000);
                retries--;
```

```
            }
            _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
            throw new DoesNotExistException("KeyPair not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} already exists.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the key pair.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Save KeyPair information to a temporary file.
    /// </summary>
    /// <param name="keyPair">The name of the key pair.</param>
    /// <returns>The full path to the temporary file.</returns>
    public string SaveKeyPair(KeyPair keyPair)
    {
        var tempPath = Path.GetTempPath();
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");

        // Save the key pair to a file in a temporary folder.
        using var stream = new FileStream(pemFileName, FileMode.Create);
        using var writer = new StreamWriter(stream);
        writer.WriteLine(keyPair.KeyMaterial);

        return pemFileName;
    }

    /// <summary>
```

```
    /// Create an Amazon EC2 security group with a specified name and
 description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
    public async Task<string> CreateSecurityGroup(string groupName, string
 groupDescription)
    {
        try
        {
            var response = await _amazonEC2.CreateSecurityGroupAsync(
                new CreateSecurityGroupRequest(groupName, groupDescription));

            // Wait until the security group exists.
            int retries = 5;
            while (retries-- > 0)
            {
                var groups = await DescribeSecurityGroups(response.GroupId);
                if (groups.Any())
                {
                    return response.GroupId;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
            throw new DoesNotExistException("security group not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
 {ec2Exception.Message}");
            }
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
```

```
                $"An error occurred while creating the security group.:
 {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Create a new Amazon EC2 VPC.
    /// </summary>
    /// <param name="cidrBlock">The CIDR block for the new security group.</
param>
    /// <returns>The VPC Id of the new VPC.</returns>
    public async Task<string?> CreateVPC(string cidrBlock)
    {

        try
        {
            var response = await _amazonEC2.CreateVpcAsync(new CreateVpcRequest
            {
                CidrBlock = cidrBlock,
            });

            Vpc vpc = response.Vpc;
            Console.WriteLine($"Created VPC with ID: {vpc.VpcId}.");
            return vpc.VpcId;
        }
        catch (AmazonEC2Exception ex)
        {
            Console.WriteLine($"Couldn't create VPC because: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyPair(string keyPairName)
    {
        try
        {
```

```
            await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError($"KeyPair {keyPairName} does not exist and
cannot be deleted. Please verify the key pair name and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete the temporary file where the key pair information was saved.
    /// </summary>
    /// <param name="tempFileName">The path to the temporary file.</param>
    public void DeleteTempFile(string tempFileName)
    {
        if (File.Exists(tempFileName))
        {
            File.Delete(tempFileName);
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 security group.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteSecurityGroup(string groupId)
    {
        try
        {
            var response =
```

```
                await _amazonEC2.DeleteSecurityGroupAsync(
                    new DeleteSecurityGroupRequest { GroupId = groupId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"Security Group {groupId} does not exist and cannot be
deleted. Please verify the ID and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 VPC.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteVpc(string vpcId)
    {
        var request = new DeleteVpcRequest
        {
            VpcId = vpcId,
        };

        var response = await _amazonEC2.DeleteVpcAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Get information about existing Amazon EC2 images.
    /// </summary>
    /// <returns>A list of image information.</returns>
    public async Task<List<Image>> DescribeImages(List<string>? imageIds)
```

```
    {
        var request = new DescribeImagesRequest();
        if (imageIds is not null)
        {
            // If the imageIds list is not null, add the list
            // to the request object.
            request.ImageIds = imageIds;
        }

        var response = await _amazonEC2.DescribeImagesAsync(request);
        return response.Images;
    }

    /// <summary>
    /// Display the information returned by DescribeImages.
    /// </summary>
    /// <param name="images">The list of image information to display.</param>
    public void DisplayImageInfo(List<Image> images)
    {
        images.ForEach(image =>
        {
            Console.WriteLine($"{image.Name} Created on: {image.CreationDate}");
        });

    }

    /// <summary>
    /// Get information about an Amazon EC2 instance.
    /// </summary>
    /// <param name="instanceId">The instance Id of the EC2 instance.</param>
    /// <returns>An EC2 instance.</returns>
    public async Task<Instance> DescribeInstance(string instanceId)
    {
        var response = await _amazonEC2.DescribeInstancesAsync(
            new DescribeInstancesRequest { InstanceIds = new List<string>
{ instanceId } });
        return response.Reservations[0].Instances[0];
    }

    /// <summary>
    /// Display EC2 instance information.
    /// </summary>
    /// <param name="instance">The instance Id of the EC2 instance.</param>
    public void DisplayInstanceInformation(Instance instance)
```

```
    {
        Console.WriteLine($"ID: {instance.InstanceId}");
        Console.WriteLine($"Image ID: {instance.ImageId}");
        Console.WriteLine($"{instance.InstanceType}");
        Console.WriteLine($"Key Name: {instance.KeyName}");
        Console.WriteLine($"VPC ID: {instance.VpcId}");
        Console.WriteLine($"Public IP: {instance.PublicIpAddress}");
        Console.WriteLine($"State: {instance.State.Name}");
    }

    /// <summary>
    /// Get information about EC2 instances with a particular state.
    /// </summary>
    /// <param name="tagName">The name of the tag to filter on.</param>
    /// <param name="tagValue">The value of the tag to look for.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> GetInstancesWithState(string state)
    {
        try
        {
            // Filters the results of the instance list.
            var filters = new List<Filter>
            {
                new Filter
                {
                    Name = $"instance-state-name",
                    Values = new List<string> { state, },
                },
            };
            var request = new DescribeInstancesRequest { Filters = filters, };

            Console.WriteLine($"\nShowing instances with state {state}");
            var paginator = _amazonEC2.Paginators.DescribeInstances(request);

            await foreach (var response in paginator.Responses)
            {
                foreach (var reservation in response.Reservations)
                {
                    foreach (var instance in reservation.Instances)
                    {
                        Console.Write($"Instance ID: {instance.InstanceId} ");
                        Console.WriteLine($"\tCurrent State:
{instance.State.Name}");
                    }
```

```
                }
            }

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Invalid parameter value for filtering instances.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list instances because: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Describe the instance types available.
    /// </summary>
    /// <returns>A list of instance type information.</returns>
    public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
    {
        try
        {
            var request = new DescribeInstanceTypesRequest();

            var filters = new List<Filter>
            {
                new Filter("processor-info.supported-architecture",
                    new List<string> { architecture.ToString() })
            };
            filters.Add(new Filter("instance-type", new() { "*.micro",
"*.small" }));

            request.Filters = filters;
            var instanceTypes = new List<InstanceTypeInfo>();
```

```
            var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
            await foreach (var instanceType in paginator.InstanceTypes)
            {
                instanceTypes.Add(instanceType);
            }

            return instanceTypes;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Parameters are invalid. Ensure architecture and size
strings conform to DescribeInstanceTypes API reference.");
            }

            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get information about an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair.</param>
    /// <returns>A list of key pair information.</returns>
    public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
    {
        try
        {
            var request = new DescribeKeyPairsRequest();
            if (!string.IsNullOrEmpty(keyPairName))
            {
                request = new DescribeKeyPairsRequest
                {
                    KeyNames = new List<string> { keyPairName }
                };
            }
```

```
            var response = await _amazonEC2.DescribeKeyPairsAsync(request);
            return response.KeyPairs.ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while describing the key pair.:
{ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Retrieve information for one or all Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        try
        {
            var securityGroups = new List<SecurityGroup>();
            var request = new DescribeSecurityGroupsRequest();

            if (!string.IsNullOrEmpty(groupId))
            {
                var groupIds = new List<string> { groupId };
                request.GroupIds = groupIds;
            }

            var paginatorForSecurityGroups =
```

```
                _amazonEC2.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            return securityGroups;

        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"A security group {groupId} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while listing security groups.
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Display the information returned by the call to
    /// DescribeSecurityGroupsAsync.
    /// </summary>
    /// <param name="securityGroup">A list of security group information.</param>
    public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
    {
        Console.WriteLine($"{securityGroup.GroupName}");
        Console.WriteLine("Ingress permissions:");
        securityGroup.IpPermissions.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");
```

```
            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
        Console.WriteLine("Egress permissions:");
        securityGroup.IpPermissionsEgress.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
    }


    /// <summary>
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
```

```
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId =
associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)
    {
        try
        {
            var request = new RebootInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.RebootInstancesAsync(request);

            // Wait for the instance to be running.
            Console.Write("Waiting for the instance to start.");
```

```
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
 {ec2Exception.Message}");
            }
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while rebooting the instance
 {ec2InstanceId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Release an Elastic IP address. After the Elastic IP address is released,
    /// it can no longer be used.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
            var request = new ReleaseAddressRequest { AllocationId =
 allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
```

```
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while releasing the AllocationId
{allocationId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
    /// <param name="groupId">The Id of the Amazon EC2 security group that will
be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
    {
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
                MinCount = 1,
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
```

```
            var instanceId = response.Reservation.Instances[0].InstanceId;

            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Start an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
    /// to start.</param>
    /// <returns>Async task.</returns>
    public async Task StartInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StartInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StartInstancesAsync(request);

            Console.Write("Waiting for instance to start. ");
```

```
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while starting the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Stop an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance to
    /// stop.</param>
    /// <returns>Async task.</returns>
    public async Task StopInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StopInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StopInstancesAsync(request);
            Console.Write("Waiting for the instance to stop.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);

            Console.WriteLine("\nThe instance has stopped.");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
```

```
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while stopping the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Terminate an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance
    /// to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
    {
        try
        {
            var request = new TerminateInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId }
            };

            var response = await _amazonEC2.TerminateInstancesAsync(request);
            Console.Write("Waiting for the instance to terminate.");
            await WaitForInstanceState(ec2InstanceId,
InstanceStateName.Terminated);

            Console.WriteLine($"\nThe instance {ec2InstanceId} has been
terminated.");
            return response.TerminatingInstances;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
```

```
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while terminating the instance.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
```

```
        } while (!hasState);

        return hasState;
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
###############################################################################
# function get_started_with_ec2_instances
#
# Runs an interactive scenario that shows how to get started using EC2 instances.
#
#     "EC2 access" permissions are needed to run this code.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function get_started_with_ec2_instances() {
  # Requires version 4 for mapfile.
  local required_version=4.0

  # Get the current Bash version
  # Check if BASH_VERSION is set
  local current_version
  if [[ -n "$BASH_VERSION" ]]; then
    # Convert BASH_VERSION to a number for comparison
    current_version=$BASH_VERSION
  else
    # Get the current Bash version using the bash command
    current_version=$(bash --version | head -n 1 | awk '{ print $4 }')
  fi

  # Convert version strings to numbers for comparison
  local required_version_num current_version_num
  required_version_num=$(echo "$required_version" | awk -F. '{ print ($1 * 10000)
 + ($2 * 100) + $3 }')
```

```
 current_version_num=$(echo "$current_version" | awk -F. '{ print ($1 * 10000) +
($2 * 100) + $3 }')

 # Compare versions
 if ((current_version_num < required_version_num)); then
   echo "Error: This script requires Bash version $required_version or higher."
   echo "Your current Bash version is number is $current_version."
   exit 1
 fi

 {
   if [ "$EC2_OPERATIONS_SOURCED" != "True" ]; then

     source ./ec2_operations.sh
   fi
 }

 echo_repeat "*" 88
 echo "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started with
instances demo."
 echo_repeat "*" 88
 echo

 echo "Let's create an RSA key pair that you can be use to securely connect to "
 echo "your EC2 instance."

 echo -n "Enter a unique name for your key: "
 get_input
 local key_name
 key_name=$get_input_result

 local temp_dir
 temp_dir=$(mktemp -d)
 local key_file_name="$temp_dir/${key_name}.pem"

 if ec2_create_keypair -n "${key_name}" -f "${key_file_name}"; then
   echo "Created a key pair $key_name and saved the private key to
$key_file_name"
   echo
 else
   errecho "The key pair failed to create. This demo will exit."
   return 1
 fi
```

```
chmod 400 "${key_file_name}"

if yes_no_input "Do you want to list some of your key pairs? (y/n) "; then
  local keys_and_fingerprints
  keys_and_fingerprints="$(ec2_describe_key_pairs)" && {
    local image_name_and_id
    while IFS=$'\n' read -r image_name_and_id; do
      local entries
      IFS=$'\t' read -ra entries <<<"$image_name_and_id"
      echo "Found rsa key ${entries[0]} with fingerprint:"
      echo "     ${entries[1]}"
    done <<<"$keys_and_fingerprints"

  }
fi

echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's create a security group to manage access to your instance."
echo -n "Enter a unique name for your security group: "
get_input
local security_group_name
security_group_name=$get_input_result
local security_group_id
security_group_id=$(ec2_create_security_group -n "$security_group_name" \
  -d "Security group for EC2 instance") || {
  errecho "The security failed to create. This demo will exit."
  clean_up "$key_name" "$key_file_name"
  return 1
}

echo "Security group created with ID $security_group_id"
echo

local public_ip
public_ip=$(curl -s http://checkip.amazonaws.com)

echo "Let's add a rule to allow SSH only from your current IP address."
echo "Your public IP address is $public_ip"
echo -n "press return to add this rule to your security group."
get_input
```

```
  if ! ec2_authorize_security_group_ingress -g "$security_group_id" -i
 "$public_ip" -p tcp -f 22 -t 22; then
    errecho "The security group rules failed to update. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  fi

  echo "Security group rules updated"

  local security_group_description
  security_group_description="$(ec2_describe_security_groups -g
 "${security_group_id}")" || {
    errecho "Failed to describe security groups. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  mapfile -t parameters <<<"$security_group_description"
  IFS=$'\t' read -ra entries <<<"${parameters[0]}"
  echo "Security group: ${entries[0]}"
  echo "    ID: ${entries[1]}"
  echo "    VPC: ${entries[2]}"
  echo "Inbound permissions:"
  IFS=$'\t' read -ra entries <<<"${parameters[1]}"
  echo "    IpProtocol: ${entries[0]}"
  echo "    FromPort: ${entries[1]}"
  echo "    ToPort: ${entries[2]}"
  echo "    CidrIp: ${parameters[2]}"

  local parameters
  parameters="$(ssm_get_parameters_by_path -p "/aws/service/ami-amazon-linux-
latest")" || {
    errecho "Failed to get parameters. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1

  }

  local image_ids=""
  mapfile -t parameters <<<"$parameters"
  for image_name_and_id in "${parameters[@]}"; do
    IFS=$'\t' read -ra values <<<"$image_name_and_id"
    if [[ "${values[0]}" == *"amzn2"* ]]; then
      image_ids+="${values[1]} "
```

```
    fi
  done

  local images
  images="$(ec2_describe_images -i "$image_ids")" || {
    errecho "Failed to describe images. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1

  }

  new_line_and_tab_to_list "$images"
  local images=("${list_result[@]}")

  # Get the size of the array
  local images_count=${#images[@]}

  if ((images_count == 0)); then
    errecho "No images found. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  fi

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's create an instance from an Amazon Linux 2 AMI. Here are some
 options:"
  for ((i = 0; i < images_count; i += 3)); do
    echo "$(((i / 3) + 1)) - ${images[$i]}"
  done

  integer_input "Please enter the number of the AMI you want to use: " 1
 "$((images_count / 3))"
  local choice=$get_input_result
  choice=$(((choice - 1) * 3))

  echo "Great choice."
  echo

  local architecture=${images[$((choice + 1))]}
  local image_id=${images[$((choice + 2))]}
  echo "Here are some instance types that support the ${architecture}
 architecture of the image:"
```

```
  response="$(ec2_describe_instance_types -a "${architecture}" -t
 "*.micro,*.small")" || {
    errecho "Failed to describe instance types. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  local instance_types
  mapfile -t instance_types <<<"$response"

  # Get the size of the array
  local instance_types_count=${#instance_types[@]}

  echo "Here are some options:"
  for ((i = 0; i < instance_types_count; i++)); do
    echo "$((i + 1)) - ${instance_types[$i]}"
  done

  integer_input "Which one do you want to use? " 1 "${#instance_types[@]}
"
  choice=$get_input_result
  local instance_type=${instance_types[$((choice - 1))]}
  echo "Another great choice."
  echo

  echo "Creating your instance and waiting for it to start..."
  local instance_id
  instance_id=$(ec2_run_instances -i "$image_id" -t "$instance_type" -k
 "$key_name" -s "$security_group_id") || {
    errecho "Failed to run instance. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  ec2_wait_for_instance_running -i "$instance_id"
  echo "Your instance is ready:"
  echo

  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  echo
  print_instance_details "${instance_details}"
```

```
  local public_ip
  public_ip=$(echo "${instance_details}" | awk '{print $6}')
  echo
  echo "You can use SSH to connect to your instance"
  echo "If the connection attempt times out, you might have to manually update
 the SSH ingress rule"
  echo "for your IP address in the AWS Management Console."
  connect_to_instance "$key_file_name" "$public_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's stop and start your instance to see what changes."
  echo "Stopping your instance and waiting until it's stopped..."
  ec2_stop_instances -i "$instance_id"
  ec2_wait_for_instance_stopped -i "$instance_id"

  echo "Your instance is stopped. Restarting..."

  ec2_start_instances -i "$instance_id"
  ec2_wait_for_instance_running -i "$instance_id"

  echo "Your instance is running again."
  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  print_instance_details "${instance_details}"

  public_ip=$(echo "${instance_details}" | awk '{print $6}')

  echo "Every time your instance is restarted, its public IP address changes"
  connect_to_instance "$key_file_name" "$public_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "You can allocate an Elastic IP address and associate it with your
 instance"
```

```
  echo "to keep a consistent IP address even when your instance restarts."

  local result
  result=$(ec2_allocate_address -d vpc) || {
    errecho "Failed to allocate an address. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
    return 1
  }

  local elastic_ip allocation_id
  elastic_ip=$(echo "$result" | awk '{print $1}')
  allocation_id=$(echo "$result" | awk '{print $2}')

  echo "Allocated static Elastic IP address: $elastic_ip"

  local association_id
  association_id=$(ec2_associate_address -i "$instance_id" -a "$allocation_id")
 || {
    errecho "Failed to associate an address. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
 "$allocation_id"
    return 1
  }

  echo "Associated your Elastic IP with your instance."
  echo "You can now use SSH to connect to your instance by using the Elastic IP."
  connect_to_instance "$key_file_name" "$elastic_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's stop and start your instance to see what changes."
  echo "Stopping your instance and waiting until it's stopped..."
  ec2_stop_instances -i "$instance_id"
  ec2_wait_for_instance_stopped -i "$instance_id"

  echo "Your instance is stopped. Restarting..."

  ec2_start_instances -i "$instance_id"
  ec2_wait_for_instance_running -i "$instance_id"
```

```
  echo "Your instance is running again."
  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  print_instance_details "${instance_details}"

  echo "Because you have associated an Elastic IP with your instance, you can"
  echo "connect by using a consistent IP address after the instance restarts."
  connect_to_instance "$key_file_name" "$elastic_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  if yes_no_input "Do you want to delete the resources created in this demo: (y/
n) "; then
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id" \
      "$allocation_id" "$association_id"
  else
    echo "The following resources were not deleted."
    echo "Key pair: $key_name"
    echo "Key file: $key_file_name"
    echo "Security group: $security_group_id"
    echo "Instance: $instance_id"
    echo "Elastic IP address: $elastic_ip"
  fi
}

###############################################################################
# function clean_up
#
# This function cleans up the created resources.
#     $1 - The name of the ec2 key pair to delete.
#     $2 - The name of the key file to delete.
#     $3 - The ID of the security group to delete.
#     $4 - The ID of the instance to terminate.
#     $5 - The ID of the elastic IP address to release.
#     $6 - The ID of the elastic IP address to disassociate.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
```

```
###############################################################################
function clean_up() {
  local result=0
  local key_pair_name=$1
  local key_file_name=$2
  local security_group_id=$3
  local instance_id=$4
  local allocation_id=$5
  local association_id=$6

  if [ -n "$association_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_disassociate_address -a "$association_id"); then
      echo "Disassociated elastic IP address with ID $association_id"
    else
      errecho "The elastic IP address disassociation failed."
      result=1
    fi
  fi

  if [ -n "$allocation_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_release_address -a "$allocation_id"); then
      echo "Released elastic IP address with ID $allocation_id"
    else
      errecho "The elastic IP address release failed."
      result=1
    fi
  fi

  if [ -n "$instance_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_terminate_instances -i "$instance_id"); then
      echo "Started terminating instance with ID $instance_id"

      ec2_wait_for_instance_terminated -i "$instance_id"
    else
      errecho "The instance terminate failed."
      result=1
    fi
  fi

  if [ -n "$security_group_id" ]; then
    # bashsupport disable=BP2002
```

```
    if (ec2_delete_security_group -i "$security_group_id"); then
      echo "Deleted security group with ID $security_group_id"
    else
      errecho "The security group delete failed."
      result=1
    fi
  fi

  if [ -n "$key_pair_name" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_keypair -n "$key_pair_name"); then
      echo "Deleted key pair named $key_pair_name"
    else
      errecho "The key pair delete failed."
      result=1
    fi
  fi

  if [ -n "$key_file_name" ]; then
    rm -f "$key_file_name"
  fi

  return $result
}

###############################################################################
# function ssm_get_parameters_by_path
#
# This function retrieves one or more parameters from the AWS Systems Manager
 Parameter Store
# by specifying a parameter path.
#
# Parameters:
#       -p parameter_path - The path of the parameter(s) to retrieve.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ssm_get_parameters_by_path() {
  local parameter_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
```

```
  function usage() {
    echo "function ssm_get_parameters_by_path"
    echo "Retrieves one or more parameters from the AWS Systems Manager Parameter
 Store by specifying a parameter path."
    echo "  -p parameter_path - The path of the parameter(s) to retrieve."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "p:h" option; do
    case "${option}" in
      p) parameter_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$parameter_path" ]]; then
    errecho "ERROR: You must provide a parameter path with the -p parameter."
    usage
    return 1
  fi

  response=$(aws ssm get-parameters-by-path \
    --path "$parameter_path" \
    --query "Parameters[*].[Name, Value]" \
    --output text) || {
    aws_cli_error_log $?
    errecho "ERROR: AWS reports get-parameters-by-path operation failed.
$response"
    return 1
  }

  echo "$response"

  return 0
```

```
}

##############################################################################
# function print_instance_details
#
# This function prints the details of an Amazon Elastic Compute Cloud (Amazon
 EC2) instance.
#
# Parameters:
#       instance_details - The instance details in the format "InstanceId ImageId
 InstanceType KeyName VpcId PublicIpAddress State.Name".
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function print_instance_details() {
  local instance_details="$1"

  if [[ -z "${instance_details}" ]]; then
    echo "Error: Missing required instance details argument."
    return 1
  fi

  local instance_id image_id instance_type key_name vpc_id public_ip state
  instance_id=$(echo "${instance_details}" | awk '{print $1}')
  image_id=$(echo "${instance_details}" | awk '{print $2}')
  instance_type=$(echo "${instance_details}" | awk '{print $3}')
  key_name=$(echo "${instance_details}" | awk '{print $4}')
  vpc_id=$(echo "${instance_details}" | awk '{print $5}')
  public_ip=$(echo "${instance_details}" | awk '{print $6}')
  state=$(echo "${instance_details}" | awk '{print $7}')

  echo "    ID: ${instance_id}"
  echo "    Image ID: ${image_id}"
  echo "    Instance type: ${instance_type}"
  echo "    Key name: ${key_name}"
  echo "    VPC ID: ${vpc_id}"
  echo "    Public IP: ${public_ip}"
  echo "    State: ${state}"

  return 0
}
```

```
##############################################################################
# function connect_to_instance
#
# This function displays the public IP address of an Amazon Elastic Compute Cloud
 (Amazon EC2) instance and prompts the user to connect to the instance via SSH.
#
# Parameters:
#       $1 - The name of the key file used to connect to the instance.
#       $2 - The public IP address of the instance.
#
# Returns:
#       None
##############################################################################
function connect_to_instance() {
  local key_file_name="$1"
  local public_ip="$2"

  # Validate the input parameters
  if [[ -z "$key_file_name" ]]; then
    echo "ERROR: You must provide a key file name as the first argument." >&2
    return 1
  fi

  if [[ -z "$public_ip" ]]; then
    echo "ERROR: You must provide a public IP address as the second argument."
 >&2
    return 1
  fi

  # Display the public IP address and connection command
  echo "To connect, run the following command:"
  echo "    ssh -i ${key_file_name} ec2-user@${public_ip}"

  # Prompt the user to connect to the instance
  if yes_no_input "Do you want to connect now? (y/n) "; then
    echo "After you have connected, you can return to this example by typing
 'exit'"
    ssh -i "${key_file_name}" ec2-user@"${public_ip}"
  fi
}

##############################################################################
# function get_input
#
```

```
# This function gets user input from the command line.
#
# Outputs:
#   User input to stdout.
#
# Returns:
#       0
##############################################################################
function get_input() {

  if [ -z "${mock_input+x}" ]; then
    read -r get_input_result
  else

    if [ "$mock_input_array_index" -lt ${#mock_input_array[@]} ]; then
      get_input_result="${mock_input_array[$mock_input_array_index]}"
      # bashsupport disable=BP2001
      # shellcheck disable=SC2206
      ((mock_input_array_index++))
      echo -n "$get_input_result"
    else
      echo "MOCK_INPUT_ARRAY has no more elements" 1>&2
      return 1
    fi
  fi

  return 0
}

##############################################################################
# function yes_no_input
#
# This function requests a yes/no answer from the user, following to a prompt.
#
# Parameters:
#       $1 - The prompt.
#
# Returns:
#       0 - If yes.
#       1 - If no.
##############################################################################
function yes_no_input() {
  if [ -z "$1" ]; then
    echo "Internal error yes_no_input"
```

```
    return 1
  fi

  local index=0
  local response="N"
  while [[ $index -lt 10 ]]; do
    index=$((index + 1))
    echo -n "$1"
    if ! get_input; then
      return 1
    fi
    response=$(echo "$get_input_result" | tr '[:upper:]' '[:lower:]')
    if [ "$response" = "y" ] || [ "$response" = "n" ]; then
      break
    else
      echo -e "\nPlease enter or 'y' or 'n'."
    fi
  done

  echo

  if [ "$response" = "y" ]; then
    return 0
  else
    return 1
  fi
}

##############################################################################
# function integer_input
#
# This function prompts the user to enter an integer within a specified range
# and validates the input.
#
# Parameters:
#       $1 - The prompt message to display to the user.
#       $2 - The minimum value of the accepted range.
#       $3 - The maximum value of the accepted range.
#
# Returns:
#       The valid integer input from the user.
#       If the input is invalid or out of range, the function will continue
#       prompting the user until a valid input is provided.
##############################################################################
```

```
function integer_input() {
  local prompt="$1"
  local min_value="$2"
  local max_value="$3"
  local input=""

  while true; do
    # Display the prompt message and wait for user input
    echo -n "$prompt"

    if ! get_input; then
      return 1
    fi

    input="$get_input_result"

    # Check if the input is a valid integer
    if [[ "$input" =~ ^-?[0-9]+$ ]]; then
      # Check if the input is within the specified range
      if ((input >= min_value && input <= max_value)); then
        return 0
      else
        echo "Error: Input, $input, must be between $min_value and $max_value."
      fi
    else
      echo "Error: Invalid input- $input. Please enter an integer."
    fi
  done
}
##############################################################################
# function new_line_and_tab_to_list
#
# This function takes a string input containing newlines and tabs, and
# converts it into a list (array) of elements.
#
# Parameters:
#       $1 - The input string containing newlines and tabs.
#
# Returns:
#       The resulting list (array) is stored in the global variable
#       'list_result'.
##############################################################################
function new_line_and_tab_to_list() {
  local input=$1
```

```
  export list_result

  list_result=()
  mapfile -t lines <<<"$input"
  local line
  for line in "${lines[@]}"; do
    IFS=$'\t' read -ra parameters <<<"$line"
    list_result+=("${parameters[@]}")
  done
}

###############################################################################
# function echo_repeat
#
# This function prints a string 'n' times to stdout.
#
# Parameters:
#       $1 - The string.
#       $2 - Number of times to print the string.
#
# Outputs:
#   String 'n' times to stdout.
#
# Returns:
#       0
###############################################################################
function echo_repeat() {
  local end=$2
  for ((i = 0; i < end; i++)); do
    echo -n "$1"
  done
  echo
}
```

As funções do DynamoDB usadas nesse cenário.

```
###############################################################################
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
 2048-bit RSA key pair
# and writes it to a file.
```

```
#
# Parameters:
#       -n key_pair_name - A key pair name.
#       -f file_path - File to store the key pair.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_create_keypair() {
  local key_pair_name file_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_create_keypair"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-
bit RSA key pair"
    echo " and writes it to a file."
    echo "  -n key_pair_name - A key pair name."
    echo "  -f file_path - File to store the key pair."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:f:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      f) file_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key name with the -n parameter."
```

```
    usage
    return 1
  fi

  if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
  fi

  response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  }

  if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
  fi

  return 0
}

###############################################################################
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 key pairs.
#
# Parameters:
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_key_pairs() {
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
```

```
    echo "function ec2_describe_key_pairs"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security
 group.
#
```

```
# Parameters:
#       -n security_group_name - The name of the security group.
#       -d security_group_description - The description of the security group.
#
# Returns:
#       The ID of the created security group, or an error message if the
 operation fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_create_security_group() {
  local security_group_name security_group_description response

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -n security_group_name - The name of the security group."
    echo "  -d security_group_description - The description of the security
 group."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "n:d:h" option; do
    case "${option}" in
      n) security_group_name="${OPTARG}" ;;
      d) security_group_description="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
```

```
  if [[ -z "$security_group_name" ]]; then
    errecho "ERROR: You must provide a security group name with the -n
 parameter."
    return 1
  fi

  if [[ -z "$security_group_description" ]]; then
    errecho "ERROR: You must provide a security group description with the -d
 parameter."
    return 1
  fi

  # Create the security group
  response=$(aws ec2 create-security-group \
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups.
#
# Parameters:
#       -g security_group_id - The ID of the security group to describe
 (optional).
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_security_groups() {
  local security_group_id response
```

```
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_security_groups"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups."
    echo "  -g security_group_id - The ID of the security group to describe
 (optional)."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"

  if [[ -n "$security_group_id" ]]; then
    response=$(aws ec2 describe-security-groups --group-ids "$security_group_id"
 --query "${query}" --output text)
  else
    response=$(aws ec2 describe-security-groups --query "${query}" --output text)
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports describe-security-groups operation failed.
$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
 (Amazon EC2) security group.
#
# Parameters:
#       -g security_group_id - The ID of the security group.
#       -i ip_address - The IP address or CIDR block to authorize.
#       -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#       -f from_port - The start of the port range to authorize.
#       -t to_port - The end of the port range to authorize.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_authorize_security_group_ingress() {
  local security_group_id ip_address protocol from_port to_port response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
 EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
  }
```

```
  # Retrieve the calling parameters.
  while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      i) ip_address="${OPTARG}" ;;
      p) protocol="${OPTARG}" ;;
      f) from_port="${OPTARG}" ;;
      t) to_port="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
  fi

  if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
 parameter."
    usage
    return 1
  fi

  if [[ -z "$protocol" ]]; then
    errecho "ERROR: You must provide a protocol with the -p parameter."
    usage
    return 1
  fi

  if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
```

```
  fi

  if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
  fi

  response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation
 failed.$response"
    return 1
  }

  return 0
}

################################################################################
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images.
#
# Parameters:
#       -i image_ids - A space-separated  list of image IDs (optional).
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function ec2_describe_images() {
  local image_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_images"
```

```
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) image_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
  fi

  response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

```
##############################################################################
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g.,
 x86_64)
# -t, --type INSTANCE_TYPE         Comma-separated list of instance types (e.g.,
 t2.micro)
# -h, --help                       Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
##############################################################################
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""

  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-
t|--type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture
 (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE         Comma-separated list of instance
 types (e.g., t2.micro)"
    echo "  -h, --help                       Show this help message"
  }

  while [[ $# -gt 0 ]]; do
    case "$1" in
      -a | --architecture)
        architecture="$2"
        shift 2
        ;;
      -t | --type)
        instance_types="$2"
        shift 2
        ;;
      -h | --help)
        usage
        return 0
```

```
      ;;
    *)
      echo "Unknown argument: $1"
      return 1
      ;;
  esac
done

if [[ -z "$architecture" ]]; then
  errecho "Error: Architecture not specified."
  usage
  return 1
fi

if [[ -z "$instance_types" ]]; then
  errecho "Error: Instance type not specified."
  usage
  return 1
fi

local tmp_json_file="temp_ec2.json"
echo -n '[
  {
    "Name": "processor-info.supported-architecture",
    "Values": [' >"$tmp_json_file"

local items
IFS=',' read -ra items <<<"$architecture"
local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
  echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
  if [[ $i -lt $((array_size - 1)) ]]; then
    echo -n ',' >>"$tmp_json_file"
  fi
done
echo -n ']},
  {
  "Name": "instance-type",
    "Values": [' >>"$tmp_json_file"
IFS=',' read -ra items <<<"$instance_types"
local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
```

```
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done

  echo -n ']}]' >>"$tmp_json_file"

  local response
  response=$(aws ec2 describe-instance-types --filters file://"$tmp_json_file" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

  local error_code=$?

  rm "$tmp_json_file"

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
    return 1
  fi

  echo "$response"
  return 0
}

###############################################################################
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#       -t instance_type - The instance type to use (e.g., t2.micro).
#       -k key_pair_name - The name of the key pair to use.
#       -s security_group_id - The ID of the security group to use.
#       -c count - The number of instances to launch (default: 1).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function ec2_run_instances() {
  local image_id instance_type key_pair_name security_group_id count response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_run_instances"
    echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
    echo "  -k key_pair_name - The name of the key pair to use."
    echo "  -s security_group_id - The ID of the security group to use."
    echo "  -c count - The number of instances to launch (default: 1)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:t:k:s:c:h" option; do
    case "${option}" in
      i) image_id="${OPTARG}" ;;
      t) instance_type="${OPTARG}" ;;
      k) key_pair_name="${OPTARG}" ;;
      s) security_group_id="${OPTARG}" ;;
      c) count="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$image_id" ]]; then
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
 parameter."
    usage
    return 1
```

```
  fi

  if [[ -z "$instance_type" ]]; then
    errecho "ERROR: You must provide an instance type with the -t parameter."
    usage
    return 1
  fi

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -k parameter."
    usage
    return 1
  fi

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -s parameter."
    usage
    return 1
  fi

  if [[ -z "$count" ]]; then
    count=1
  fi

  response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

################################################################################
# function ec2_describe_instances
```

```
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID of the instance to describe (optional).
#       -q query - The query to filter the response (optional).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_describe_instances() {
  local instance_id query response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_instances"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
    echo "  -q query - The query to filter the response (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:q:h" option; do
    case "${option}" in
      i) instance_id="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
  fi

  local query_arg=""
  if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
  else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function ec2_stop_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_stop_instances"
    echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
```

```
    return 1
  }

  return 0
}

##############################################################################
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_start_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 start-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports start-instances operation failed with $response."
    return 1
  }

  return 0
}

###############################################################################
# function ec2_allocate_address
#
# This function allocates an Elastic IP address for use with Amazon Elastic
 Compute Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#       -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard').
#
# Returns:
#       The allocated Elastic IP address, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_allocate_address() {
```

```
  local domain response

  # Function to display usage information
  function usage() {
    echo "function ec2_allocate_address"
    echo "Allocates an Elastic IP address for use with Amazon Elastic Compute
 Cloud (Amazon EC2) instances in a specific AWS Region."
    echo "  -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard')."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "d:h" option; do
    case "${option}" in
      d) domain="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$domain" ]]; then
    errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc'
 or 'standard')."
    return 1
  fi

  if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
    errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
    return 1
  fi

  # Allocate the Elastic IP address
  response=$(aws ec2 allocate-address \
    --domain "$domain" \
```

```
    --query "[PublicIp,AllocationId]" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to
 associate.
#       -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_associate_address() {
  local allocation_id instance_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 associate."
    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic
 IP address with."
    echo ""
  }
```

```
  # Parse the command-line arguments
  while getopts "a:i:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      i) instance_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
    return 1
  fi

  # Associate the Elastic IP address
  response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

```
##############################################################################
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic
 Compute Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a association_id - The association ID that represents the association of
 the Elastic IP address with an instance.
#
# And:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the
 association of the Elastic IP address with an instance."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) association_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_release_address
#
# This function releases an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to
 release.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_release_address() {
  local allocation_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_release_address"
    echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 release."
```

```
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 release-address \
    --allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
```

```
#       -i instance_ids - A space-separated list of instance IDs.
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_terminate_instances() {
  local instance_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_terminate_instances"
    echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_ids - A space-separated list of instance IDs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Check if instance ID is provided
  if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
  fi
```

```
  # shellcheck disable=SC2086
  response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security
 group.
#
# Parameters:
#       -i security_group_id - The ID of the security group to delete.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_security_group() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) security_group_id="${OPTARG}" ;;
      h)
```

```
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-security-group --group-id "$security_group_id" --
output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_keypair() {
  local key_pair_name response

  local option OPTARG # Required to use getopts command in a function.
```

```
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-key-pair \
    --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas nesse cenário.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário em um prompt de comando.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ssm.model.Parameter;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets additional information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI’s
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
```

```
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(EC2Scenario.class);
    public static void main(String[] args) throws InterruptedException,
 UnknownHostException {

        logger.info("""
            Usage:
               <keyName> <fileName> <groupName> <groupDesc>

            Where:
               keyName -  A key pair name (for example, TestKeyPair).\s
               fileName -  A file name where the key information is written to.\s
               groupName - The name of the security group.\s
               groupDesc - The description of the security group.\s
            """);

        Scanner scanner = new Scanner(System.in);
        EC2Actions ec2Actions = new EC2Actions();

        String keyName = "TestKeyPair7" ;
        String fileName = "ec2Key.pem";
        String groupName = "TestSecGroup7" ;
        String groupDesc = "Test Group" ;
        String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
        InetAddress localAddress = InetAddress.getLocalHost();
        String myIpAddress = localAddress.getHostAddress();

        logger.info("""
            Amazon Elastic Compute Cloud (EC2) is a web service that provides
 secure, resizable compute
            capacity in the cloud. It allows developers and organizations to
 easily launch and manage
            virtual server instances, known as EC2 instances, to run their
 applications.
```

```
            EC2 provides a wide range of instance types, each with different
 compute, memory,
            and storage capabilities, to meet the diverse needs of various
 workloads. Developers
            can choose the appropriate instance type based on their application's
 requirements,
            such as high-performance computing, memory-intensive tasks, or GPU-
accelerated workloads.

            The `Ec2AsyncClient` interface in the AWS SDK for Java 2.x provides a
 set of methods to
            programmatically interact with the Amazon EC2 service. This allows
 developers to
            automate the provisioning, management, and monitoring of EC2
 instances as part of their
            application deployment pipelines. With EC2, teams can focus on
 building and deploying
            their applications without having to worry about the underlying
 infrastructure
            required to host and manage physical servers.

            This scenario walks you through how to perform key operations for
 this service.
            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an RSA key pair and save the private key material
 as a .pem file.");
        logger.info("""
            An RSA key pair for Amazon EC2 is a security mechanism used to
 authenticate and secure
            access to your EC2 instances. It consists of a public key and a
 private key,
            which are generated as a pair.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateKeyPairResponse> future =
 ec2Actions.createKeyPairAsync(keyName, fileName);
            CreateKeyPairResponse response = future.join();
```

```
            logger.info("Key Pair successfully created. Key Fingerprint: " +
response.keyFingerprint());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.getMessage().contains("already exists")) {
                    // Key pair already exists.
                    logger.info("The key pair '" + keyName + "' already exists.
Moving on...");
                } else {
                    logger.info("EC2 error occurred: Error message: {}, Error
code {}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " +
(rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();
            DescribeKeyPairsResponse keyPairsResponse = future.join();
            keyPairsResponse.keyPairs().forEach(keyPair -> logger.info(
                "Found key pair with name {} and fingerprint {}",
                keyPair.keyName(),
                keyPair.keyFingerprint()));

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Error message: {}, Error code
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
```

```
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a security group.");
        logger.info("""
            An AWS EC2 Security Group is a virtual firewall that controls the
            inbound and outbound traffic to an EC2 instance. It acts as a first
line
            of defense for your EC2 instances, allowing you to specify the rules
that
            govern the network traffic entering and leaving your instances.
           """);
        waitForInputToContinue(scanner);
        String groupId = "";
        try {
            CompletableFuture<String> future =
ec2Actions.createSecurityGroupAsync(groupName, groupDesc, vpcId, myIpAddress);
            future.join();
            logger.info("Created security group") ;

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {
                    logger.info("The Security Group already exists. Moving
on...");
                } else {
                    logger.error("An unexpected error occurred: {}",
ec2Ex.awsErrorDetails().errorMessage());
                    return;
                }
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Display security group information for the new security
group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeSecurityGroupArnByNameAsync(groupName);
            groupId = future.join();
            logger.info("The security group Id is "+groupId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                String errorCode = ec2Ex.awsErrorDetails().errorCode();
                if ("InvalidGroup.NotFound".equals(errorCode)) {
                    logger.info("Security group '{}' does not exist. Error Code:
{}", groupName, errorCode);
                } else {
                    logger.info("EC2 error occurred: Message {}, Error Code: {}",
ec2Ex.getMessage(), errorCode);
                }
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get a list of Amazon Linux 2 AMIs and select one with
amzn2 in the name.");
        logger.info("""
            An Amazon EC2 AMI (Amazon Machine Image) is a pre-configured virtual
machine image that
            serves as a template for launching EC2 instances. It contains all the
necessary software and
            configurations required to run an application or operating system on
an EC2 instance.
            """);
        waitForInputToContinue(scanner);
```

```
        String instanceAMI="";
        try {
            CompletableFuture<GetParametersByPathResponse> future =
ec2Actions.getParaValuesAsync();
            GetParametersByPathResponse pathResponse = future.join();
            List<Parameter> parameterList = pathResponse.parameters();
            for (Parameter para : parameterList) {
                if (filterName(para.name())) {
                    instanceAMI = para.value();
                    break;
                }
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        logger.info("The AMI value with amzn2 is: {}", instanceAMI);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
        logger.info("""
           An AMI value represents a specific version of a virtual machine (VM)
or server image.
           It uniquely identifies a particular version of an EC2 instance,
including its operating system,
           pre-installed software, and any custom configurations. This allows you
to consistently deploy the same
           VM image across your infrastructure.

            """);
        waitForInputToContinue(scanner);
        String amiValue;
        try {
```

```
            CompletableFuture<String> future =
ec2Actions.describeImageAsync(instanceAMI);
            amiValue = future.join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Retrieves an instance type available in the current AWS
region.");
        waitForInputToContinue(scanner);
        String instanceType;
        try {
            CompletableFuture<String> future =
ec2Actions.getInstanceTypesAsync();
            instanceType = future.join();
            if (!instanceType.isEmpty()) {
                logger.info("Found instance type: " + instanceType);
            } else {
                logger.info("Desired instance type not found.");
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
```

```
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an Amazon EC2 instance using the key pair, the
 instance type, the security group, and the EC2 AMI value.");
        logger.info("Once the EC2 instance is created, it is placed into a
 running state.");
        waitForInputToContinue(scanner);
        String newInstanceId;
        try {
            CompletableFuture<String> future =
 ec2Actions.runInstanceAsync(instanceType, keyName, groupName, amiValue);
            newInstanceId = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                switch (ec2Ex.awsErrorDetails().errorCode()) {
                    case "InvalidParameterValue":
                        logger.info("EC2 error occurred: Message {}, Error Code:
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InsufficientInstanceCapacity":
                        // Handle insufficient instance capacity.
                        logger.info("Insufficient instance capacity: {}, {}",
 ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InvalidGroup.NotFound":
                        // Handle security group not found.
                        logger.info("Security group not found: {},{}",
 ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    default:
                        logger.info("EC2 error occurred: {} (Code: {}",
 ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                }
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
 cause.getMessage() : rt.getMessage()));
```

```
                return;
            }
        }
        logger.info("The instance Id is " + newInstanceId);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Display information about the running instance. ");

        waitForInputToContinue(scanner);
        String publicIp;
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP {}", publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }

        }
        logger.info("You can SSH to the instance using this command:");
        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Stop the instance using a waiter (this may take a few
mins).");
        // Remove the 2nd one
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.stopInstanceAsync(newInstanceId);
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Start the instance using a waiter (this may take a few
mins).");
        try {
            CompletableFuture<Void> future =
ec2Actions.startInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Allocate an Elastic IP address and associate it with the
instance.");
        logger.info("""
```

```
            An Elastic IP address is a static public IP address that you can
associate with your EC2 instance.
            This allows you to have a fixed, predictable IP address that remains
the same even if your instance
            is stopped, terminated, or replaced.
            This is particularly useful for applications or services that need to
be accessed consistently from a
            known IP address.

            An EC2 Allocation ID (also known as a Reserved Instance Allocation
ID) is a unique identifier associated with a Reserved Instance (RI) that you
have purchased in AWS.

            When you purchase a Reserved Instance, AWS assigns a unique
Allocation ID to it.
            This Allocation ID is used to track and identify the specific RI you
have purchased,
            and it is important for managing and monitoring your Reserved
Instances.

            """);

        waitForInputToContinue(scanner);
        String allocationId;
        try {
            CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
            allocationId = future.join();
            logger.info("Successfully allocated address with ID: "
+allocationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        logger.info("The allocation Id value is " + allocationId);
        waitForInputToContinue(scanner);
        String associationId;
```

```
        try {
            CompletableFuture<String> future =
ec2Actions.associateAddressAsync(newInstanceId, allocationId);
            associationId = future.join();
            logger.info("Successfully associated address with ID: "
+associationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Describe the instance again. Note that the public IP
address has changed");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP: " + publicIp);
            logger.info("You can SSH to the instance using this command:");
            logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Disassociate and release the Elastic IP address.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
            future.join();
            logger.info("Address successfully disassociated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
            future.join(); // Wait for the operation to complete
            logger.info("Elastic IP address successfully released.");
        } catch (RuntimeException rte) {
            logger.info("An unexpected error occurred: {}", rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Terminate the instance and use a waiter (this may take a
few mins).");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
ec2Actions.terminateEC2Async(newInstanceId);
```

```
            future.join();
            logger.info("EC2 instance successfully terminated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Delete the security group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
            future.join();
            logger.info("Security group successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Delete the key.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<DeleteKeyPairResponse> future =
ec2Actions.deleteKeysAsync(keyName);
            future.join();
            logger.info("Successfully deleted key pair named " + keyName);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon EC2 scenario.");
        logger.info(DASHES);
    }
    public static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
```

```
    }
}
```

Defina uma classe que envolva as ações do EC2.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
import
 software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.InstanceTypeInfo;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.IpRange;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
```

```
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Vpc;
import software.amazon.awssdk.services.ec2.paginators.DescribeImagesPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesPublisher;
import
 software.amazon.awssdk.services.ec2.paginators.DescribeSecurityGroupsPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeVpcsPublisher;
import software.amazon.awssdk.services.ec2.waiters.Ec2AsyncWaiter;
import software.amazon.awssdk.services.ssm.SsmAsyncClient;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathRequest;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesResponse;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.concurrent.atomic.AtomicReference;

public class EC2Actions {
    private static final Logger logger =
 LoggerFactory.getLogger(EC2Actions.class);
    private static Ec2AsyncClient ec2AsyncClient;

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static Ec2AsyncClient getAsyncClient() {
        if (ec2AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
```

```
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
              .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .build();

           ec2AsyncClient = Ec2AsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return ec2AsyncClient;
   }

   /**
    * Deletes a key pair asynchronously.
    *
    * @param keyPair the name of the key pair to delete
    * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
    *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
    *         that provides the result of the key pair deletion operation.
    */
   public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String
keyPair) {
       DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
           .keyName(keyPair)
```

```
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " +
keyPair, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }

    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is
deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with
Id " + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }

    /**
     * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
```

```
    *
    * @param instanceId the ID of the EC2 instance to terminate
    * @return a {@link CompletableFuture} that completes when the instance has
been terminated
    * @throws RuntimeException if there is no response from the AWS SDK or if
there is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
           System.out.println("Going to terminate an EC2 instance and use a
waiter to wait for it to be in terminated state");
           return getAsyncClient().waiter()
               .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
               .thenApply(waiterResponse -> null);
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
       });
   }

   /**
    * Releases an Elastic IP address asynchronously.
    *
    * @param allocId the allocation ID of the Elastic IP address to be released
    * @return a {@link CompletableFuture} representing the asynchronous
operation of releasing the Elastic IP address
    */
   public CompletableFuture<ReleaseAddressResponse>
releaseEC2AddressAsync(String allocId) {
       ReleaseAddressRequest request = ReleaseAddressRequest.builder()
           .allocationId(allocId)
           .build();
```

```
        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP
address", ex);
            }
        });

        return response;
    }

    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous
operation of disassociating the address. The
     *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
     *         finished.
     * @throws RuntimeException if the disassociation of the address fails.
     */
    public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }
```

```
    /**
     * Associates an Elastic IP address with an EC2 instance asynchronously.
     *
     * @param instanceId    the ID of the EC2 instance to associate the Elastic
IP address with
     * @param allocationId  the allocation ID of the Elastic IP address to
associate
     * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
     *         or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<String> associateAddressAsync(String instanceId,
String allocationId) {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
        return responseFuture.thenApply(response -> {
            if (response.associationId() != null) {
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after
associating address.");
            }
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }

    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
```

```
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }

    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a
string describing the state of the EC2 instance
     */
    public CompletableFuture<String> describeEC2InstancesAsync(String
newInstanceId) {
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
        AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.reservations().stream()
                .flatMap(reservation -> reservation.instances().stream())
                .filter(instance -> instance.instanceId().equals(newInstanceId))
                .findFirst()
                .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
        }).thenApply(v -> {
            String publicIpAddress = publicIpAddressRef.get();
            if (publicIpAddress == null) {
                throw new RuntimeException("Instance with ID " + newInstanceId +
" not found.");
            }
            return publicIpAddress;
```

```
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
        });
    }

    /**
     * Runs an EC2 instance asynchronously.
     *
     * @param instanceType The instance type to use for the EC2 instance.
     * @param keyName The name of the key pair to associate with the EC2
instance.
     * @param groupName The name of the security group to associate with the EC2
instance.
     * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
     * @return A {@link CompletableFuture} that completes with the ID of the
started EC2 instance.
     * @throws RuntimeException If there is an error running the EC2 instance.
     */
    public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
        return responseFuture.thenCompose(response -> {
            String instanceIdVal = response.instances().get(0).instanceId();
            System.out.println("Going to start an EC2 instance and use a waiter
to wait for it to be in running state");
            return getAsyncClient().waiter()
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
```

```
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it
returns a
     * {@link CompletableFuture} that resolves to the instance type string for
the "t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge"
instance type is not
     * found, an empty string is returned.
     * </p>
     *
     * @return a {@link CompletableFuture} that resolves to the instance type
string for the
     * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
     */
    public CompletableFuture<String> getInstanceTypesAsync() {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
                for (InstanceTypeInfo type : instanceTypes) {
                    logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                    logger.info("Network information is " +
type.networkInfo().toString());
```

```
                    logger.info("Instance type is " +
type.instanceType().toString());
                }
            } else {
                throw (RuntimeException) ex;
            }
        });

        return response.thenApply(resp -> {
            for (InstanceTypeInfo type : resp.instanceTypes()) {
                String instanceType = type.instanceType().toString();
                if (instanceType.equals("t2.2xlarge")) {
                    return instanceType;
                }
            }
            return "";
        });
    }

    /**
     * Asynchronously describes an AWS EC2 image with the specified image ID.
     *
     * @param imageId the ID of the image to be described
     * @return a {@link CompletableFuture} that, when completed, contains the ID
of the described image
     * @throws RuntimeException if no images are found with the provided image
ID, or if an error occurs during the AWS API call
     */
    public CompletableFuture<String> describeImageAsync(String imageId) {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(imageId)
            .build();

        AtomicReference<String> imageIdRef = new AtomicReference<>();
        DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
        return paginator.subscribe(response -> {
            response.images().stream()
                .filter(image -> image.imageId().equals(imageId))
                .findFirst()
                .ifPresent(image -> {
                    logger.info("The description of the image is " +
image.description());
                    logger.info("The name of the image is " + image.name());
```

```
                    imageIdRef.set(image.imageId());
                });
        }).thenApply(v -> {
            String id = imageIdRef.get();
            if (id == null) {
                throw new RuntimeException("No images found with the provided
image ID.");
            }
            return id;
        }).exceptionally(ex -> {
            logger.info("Failed to describe image: " + ex.getMessage());
            throw new RuntimeException("Failed to describe image", ex);
        });
    }

    /**
     * Retrieves the parameter values asynchronously using the AWS Systems
Manager (SSM) API.
     *
     * @return a {@link CompletableFuture} that holds the response from the SSM
API call to get parameters by path
     */
    public CompletableFuture<GetParametersByPathResponse> getParaValuesAsync() {
        SsmAsyncClient ssmClient = SsmAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        // Create a CompletableFuture to hold the final result.
        CompletableFuture<GetParametersByPathResponse> responseFuture = new
CompletableFuture<>();
        ssmClient.getParametersByPath(parameterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    responseFuture.completeExceptionally(new
RuntimeException("Failed to get parameters by path", exception));
                } else {
                    responseFuture.complete(response);
                }
            });
```

```
        return responseFuture;
    }


    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupNames(groupName)
            .build();

        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the
name: " + groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

```
   /**
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP
address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security
group
    * @param myIpAddress  the IP address from which to allow inbound traffic
(e.g., "192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security
group or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
                   .build();

               IpPermission ipPerm = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(80)
                   .fromPort(80)
                   .ipRanges(ipRange)
                   .build();

               IpPermission ipPerm2 = IpPermission.builder()
                   .ipProtocol("tcp")
```

```
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
```

```
        });

        return responseFuture;
    }

    /**
     * Creates a new key pair asynchronously.
     *
     * @param keyName the name of the key pair to create
     * @param fileName the name of the file to write the key material to
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of creating the key pair and writing the key material to a file
     */
    public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
        responseFuture.whenComplete((response, exception) -> {
            if (response != null) {
                try {
                    BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                    writer.write(response.keyMaterial());
                    writer.close();
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Describes the first default VPC asynchronously and using a paginator.
```

```
     *
     * @return a {@link CompletableFuture} that, when completed, contains the
first default VPC found.\
     */
    public CompletableFuture<Vpc> describeFirstEC2VpcAsync() {
        Filter myFilter = Filter.builder()
            .name("is-default")
            .values("true")
            .build();

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
            .filters(myFilter)
            .build();

        DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
        AtomicReference<Vpc> vpcRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.vpcs().stream()
                .findFirst()
                .ifPresent(vpcRef::set);
        }).thenApply(v -> {
            Vpc vpc = vpcRef.get();
            if (vpc == null) {
                throw new RuntimeException("Default VPC not found");
            }
            return vpc;
        }).exceptionally(ex -> {
            logger.info("Failed to describe VPCs: " + ex.getMessage());
            throw new RuntimeException("Failed to describe VPCs", ex);
        });
    }

    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for
the instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has
been stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
```

```
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new
RuntimeException("No instances were stopped. Please check the instance ID: " +
instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed
to stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }

    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
```

```
     * @return a {@link CompletableFuture} that completes when the instance has
been started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed
to start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AllocateAddress
  - AssociateAddress

- AuthorizeSecurityGroupIngress
- CreateKeyPair
- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo contém uma lista de ações comuns usadas com o EC2. As etapas são construídas com um framework de cenário que simplifica a execução de um exemplo interativo. Para ver o contexto completo, visite o GitHub repositório.

```
import { tmpdir } from "node:os";
import { writeFile, mkdtemp, rm } from "node:fs/promises";
```

```
import { join } from "node:path";
import { get } from "node:http";

import {
  AllocateAddressCommand,
  AssociateAddressCommand,
  AuthorizeSecurityGroupIngressCommand,
  CreateKeyPairCommand,
  CreateSecurityGroupCommand,
  DeleteKeyPairCommand,
  DeleteSecurityGroupCommand,
  DisassociateAddressCommand,
  paginateDescribeImages,
  paginateDescribeInstances,
  paginateDescribeInstanceTypes,
  ReleaseAddressCommand,
  RunInstancesCommand,
  StartInstancesCommand,
  StopInstancesCommand,
  TerminateInstancesCommand,
  waitUntilInstanceStatusOk,
  waitUntilInstanceStopped,
  waitUntilInstanceTerminated,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

import { paginateGetParametersByPath, SSMClient } from "@aws-sdk/client-ssm";

/**
 * @typedef {{
 *   ec2Client: import('@aws-sdk/client-ec2').EC2Client,
 *   errors: Error[],
 *   keyPairId?: string,
 *   tmpDirectory?: string,
 *   securityGroupId?: string,
 *   ipAddress?: string,
 *   images?: import('@aws-sdk/client-ec2').Image[],
 *   image?: import('@aws-sdk/client-ec2').Image,
 *   instanceTypes?: import('@aws-sdk/client-ec2').InstanceTypeInfo[],
```

```
 *   instanceId?: string,
 *   instanceIpAddress?: string,
 *   allocationId?: string,
 *   allocatedIpAddress?: string,
 *   associationId?: string,
 * }} State
 */

/**
 * A skip function provided to the `skipWhen` of a Step when you want
 * to ignore that step if any errors have occurred.
 * @param {State} state
 */
const skipWhenErrors = (state) => state.errors.length > 0;

const MAX_WAITER_TIME_IN_SECONDS = 60 * 8;

export const confirm = new ScenarioInput("confirmContinue", "Continue?", {
  type: "confirm",
  skipWhen: skipWhenErrors,
});

export const exitOnNoConfirm = new ScenarioAction(
  "exitOnConfirmContinueFalse",
  (/** @type { { earlyExit: boolean } & Record<string, any>} */ state) => {
    if (!state[confirm.name]) {
      state.earlyExit = true;
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

export const greeting = new ScenarioOutput(
  "greeting",
  `

Welcome to the Amazon EC2 basic usage scenario.

Before you launch an instances, you'll need to provide a few things:
 - A key pair - This is for SSH access to your EC2 instance. You only need to
 provide the name.
```

```
 - A security group - This is used for configuring access to your instance.
 Again, only the name is needed.
 - An IP address - Your public IP address will be fetched.
 - An Amazon Machine Image (AMI)
 - A compatible instance type`,
  { header: true, preformatted: true, skipWhen: skipWhenErrors },
);

export const provideKeyPairName = new ScenarioInput(
  "keyPairName",
  "Provide a name for a new key pair.",
  { type: "input", default: "ec2-example-key-pair", skipWhen: skipWhenErrors },
);

export const createKeyPair = new ScenarioAction(
  "createKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Create a key pair in Amazon EC2.
      const { KeyMaterial, KeyPairId } = await state.ec2Client.send(
        // A unique name for the key pair. Up to 255 ASCII characters.
        new CreateKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );

      state.keyPairId = KeyPairId;

      // Save the private key in a temporary location.
      state.tmpDirectory = await mkdtemp(join(tmpdir(), "ec2-scenario-tmp"));
      await writeFile(
        `${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem`,
        KeyMaterial,
        {
          mode: 0o400,
        },
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidKeyPair.Duplicate"
      ) {
        caught.message = `${caught.message}. Try another key name.`;
      }

      state.errors.push(caught);
```

```
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logKeyPair = new ScenarioOutput(
  "logKeyPair",
  (/** @type {State} */ state) =>
    `Created the key pair ${state[provideKeyPairName.name]}.`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteKeyPair = new ScenarioInput(
  "confirmDeleteKeyPair",
  "Do you want to delete the key pair?",
  {
    type: "confirm",
    // Don't do anything when a key pair was never created.
    skipWhen: (/** @type {State} */ state) => !state.keyPairId,
  },
);

export const maybeDeleteKeyPair = new ScenarioAction(
  "deleteKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Delete a key pair by name from EC2
      await state.ec2Client.send(
        new DeleteKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        // Occurs when a required parameter (e.g. KeyName) is undefined.
        caught.name === "MissingParameter"
      ) {
        caught.message = `${caught.message}. Did you provide the required value?
`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no key pair to delete or the user chooses
```

```
    // to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.keyPairId || !state[confirmDeleteKeyPair.name],
  },
);

export const provideSecurityGroupName = new ScenarioInput(
  "securityGroupName",
  "Provide a name for a new security group.",
  { type: "input", default: "ec2-scenario-sg", skipWhen: skipWhenErrors },
);

export const createSecurityGroup = new ScenarioAction(
  "createSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Create a new security group that will be used to configure ingress/
egress for
      // an EC2 instance.
      const { GroupId } = await state.ec2Client.send(
        new CreateSecurityGroupCommand({
          GroupName: state[provideSecurityGroupName.name],
          Description: "A security group for the Amazon EC2 example.",
        }),
      );
      state.securityGroupId = GroupId;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidGroup.Duplicate") {
        caught.message = `${caught.message}. Please provide a different name for
 your security group.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroup = new ScenarioOutput(
  "logSecurityGroup",
  (/** @type {State} */ state) =>
    `Created the security group ${state.securityGroupId}.`,
  { skipWhen: skipWhenErrors },
);
```

```
export const confirmDeleteSecurityGroup = new ScenarioInput(
  "confirmDeleteSecurityGroup",
  "Do you want to delete the security group?",
  {
    type: "confirm",
    // Don't do anything when a security group was never created.
    skipWhen: (/** @type {State} */ state) => !state.securityGroupId,
  },
);

export const maybeDeleteSecurityGroup = new ScenarioAction(
  "deleteSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Delete the security group if the 'skipWhen' condition below is not met.
      await state.ec2Client.send(
        new DeleteSecurityGroupCommand({
          GroupId: state.securityGroupId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no security group to delete
    // or the user chooses to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.securityGroupId || !state[confirmDeleteSecurityGroup.name],
  },
);

export const authorizeSecurityGroupIngress = new ScenarioAction(
  "authorizeSecurity",
  async (/** @type {State} */ state) => {
    try {
      // Get the public IP address of the machine running this example.
```

```
      const ipAddress = await new Promise((res, rej) => {
        get("http://checkip.amazonaws.com", (response) => {
          let data = "";
          response.on("data", (chunk) => {
            data += chunk;
          });
          response.on("end", () => res(data.trim()));
        }).on("error", (err) => {
          rej(err);
        });
      });
      state.ipAddress = ipAddress;
      // Allow ingress from the IP address above to the security group.
      // This will allow you to SSH into the EC2 instance.
      const command = new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.securityGroupId,
        IpPermissions: [
          {
            IpProtocol: "tcp",
            FromPort: 22,
            ToPort: 22,
            IpRanges: [{ CidrIp: `${ipAddress}/32` }],
          },
        ],
      });

      await state.ec2Client.send(command);
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroupIngress = new ScenarioOutput(
  "logSecurityGroupIngress",
  (/** @type {State} */ state) =>
```

```
    `Allowed SSH access from your public IP: ${state.ipAddress}.`,
  { skipWhen: skipWhenErrors },
);

export const getImages = new ScenarioAction(
  "images",
  async (/** @type {State} */ state) => {
    const AMIs = [];
    // Some AWS services publish information about common artifacts as AWS
 Systems Manager (SSM)
    // public parameters. For example, the Amazon Elastic Compute Cloud (Amazon
 EC2)
    // service publishes information about Amazon Machine Images (AMIs) as public
 parameters.

    // Create the paginator for getting images. Actions that return multiple
 pages of
    // results have paginators to simplify those calls.
    const getParametersByPathPaginator = paginateGetParametersByPath(
      {
        // Not storing this client in state since it's only used once.
        client: new SSMClient({}),
      },
      {
        // The path to the public list of the latest amazon-linux instances.
        Path: "/aws/service/ami-amazon-linux-latest",
      },
    );

    try {
      for await (const page of getParametersByPathPaginator) {
        for (const param of page.Parameters) {
          // Filter by Amazon Linux 2
          if (param.Name.includes("amzn2")) {
            AMIs.push(param.Value);
          }
        }
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidFilterValue") {
        caught.message = `${caught.message} Please provide a valid filter value
 for paginateGetParametersByPath.`;
      }
      state.errors.push(caught);
```

```
      return;
    }

    const imageDetails = [];
    const describeImagesPaginator = paginateDescribeImages(
      { client: state.ec2Client },
      // The images found from the call to SSM.
      { ImageIds: AMIs },
    );

    try {
      // Get more details for the images found above.
      for await (const page of describeImagesPaginator) {
        imageDetails.push(...(page.Images || []));
      }

      // Store the image details for later use.
      state.images = imageDetails;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidAMIID.NotFound") {
        caught.message = `${caught.message}. Please provide a valid image id.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const provideImage = new ScenarioInput(
  "image",
  "Select one of the following images.",
  {
    type: "select",
    choices: (/** @type { State } */ state) =>
      state.images.map((image) => ({
        name: `${image.Description}`,
        value: image,
      })),
    default: (/** @type { State } */ state) => state.images[0],
    skipWhen: skipWhenErrors,
  },
);
```

```
export const getCompatibleInstanceTypes = new ScenarioAction(
  "getCompatibleInstanceTypes",
  async (/** @type {State} */ state) => {
    // Get more details about instance types that match the architecture of
    // the provided image.
    const paginator = paginateDescribeInstanceTypes(
      { client: state.ec2Client, pageSize: 25 },
      {
        Filters: [
          {
            Name: "processor-info.supported-architecture",
            // The value selected from provideImage()
            Values: [state.image.Architecture],
          },
          // Filter for smaller, less expensive, types.
          { Name: "instance-type", Values: ["*.micro", "*.small"] },
        ],
      },
    );

    const instanceTypes = [];

    try {
      for await (const page of paginator) {
        if (page.InstanceTypes.length) {
          instanceTypes.push(...(page.InstanceTypes || []));
        }
      }

      if (!instanceTypes.length) {
        state.errors.push(
          "No instance types matched the instance type filters.",
        );
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check the provided values and
 try again.`;
      }

      state.errors.push(caught);
    }

    state.instanceTypes = instanceTypes;
```

```
  },
  { skipWhen: skipWhenErrors },
);

export const provideInstanceType = new ScenarioInput(
  "instanceType",
  "Select an instance type.",
  {
    choices: (/** @type {State} */ state) =>
      state.instanceTypes.map((instanceType) => ({
        name: `${instanceType.InstanceType} - Memory:
${instanceType.MemoryInfo.SizeInMiB}`,
        value: instanceType.InstanceType,
      })),
    type: "select",
    default: (/** @type {State} */ state) =>
      state.instanceTypes[0].InstanceType,
    skipWhen: skipWhenErrors,
  },
);

export const runInstance = new ScenarioAction(
  "runInstance",
  async (/** @type { State } */ state) => {
    const { Instances } = await state.ec2Client.send(
      new RunInstancesCommand({
        KeyName: state[provideKeyPairName.name],
        SecurityGroupIds: [state.securityGroupId],
        ImageId: state.image.ImageId,
        InstanceType: state[provideInstanceType.name],
        // Availability Zones have capacity limitations that may impact your
 ability to launch instances.
        // The `RunInstances` operation will only succeed if it can allocate at
 least the `MinCount` of instances.
        // However, EC2 will attempt to launch up to the `MaxCount` of instances,
 even if the full request cannot be satisfied.
        // If you need a specific number of instances, use `MinCount` and
 `MaxCount` set to the same value.
        // If you want to launch up to a certain number of instances, use
 `MaxCount` and let EC2 provision as many as possible.
        // If you require a minimum number of instances, but do not want to
 exceed a maximum, use both `MinCount` and `MaxCount`.
        MinCount: 1,
        MaxCount: 1,
```

```
      }),
    );

    state.instanceId = Instances[0].InstanceId;

    try {
      // Poll `DescribeInstanceStatus` until status is "ok".
      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [Instances[0].InstanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
 the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logRunInstance = new ScenarioOutput(
  "logRunInstance",
  "The next step is to run your EC2 instance for the first time. This can take a
 few minutes.",
  { header: true, skipWhen: skipWhenErrors },
);

export const describeInstance = new ScenarioAction(
  "describeInstance",
  async (/** @type { State } */ state) => {
    /** @type { import("@aws-sdk/client-ec2").Instance[] } */
    const instances = [];

    try {
      const paginator = paginateDescribeInstances(
        {
          client: state.ec2Client,
        },
```

```
        {
          // Only get our created instance.
          InstanceIds: [state.instanceId],
        },
      );

      for await (const page of paginator) {
        for (const reservation of page.Reservations) {
          instances.push(...reservation.Instances);
        }
      }
      if (instances.length !== 1) {
        throw new Error(`Instance ${state.instanceId} not found.`);
      }

      // The only info we need is the IP address for SSH purposes.
      state.instanceIpAddress = instances[0].PublicIpAddress;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check provided values and try
 again.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSSHConnectionInfo = new ScenarioOutput(
  "logSSHConnectionInfo",
  (/** @type { State } */ state) =>
    `You can now SSH into your instance using the following command:
ssh -i ${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem ec2-user@
${state.instanceIpAddress}`,
  { preformatted: true, skipWhen: skipWhenErrors },
);

export const logStopInstance = new ScenarioOutput(
  "logStopInstance",
  "Stopping your EC2 instance.",
  { skipWhen: skipWhenErrors },
);
```

```
export const stopInstance = new ScenarioAction(
  "stopInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StopInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStopped(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
 the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  // Don't try to stop an instance that doesn't exist.
  { skipWhen: (/** @type { State } */ state) => !state.instanceId },
);

export const logIpAddressBehavior = new ScenarioOutput(
  "logIpAddressBehavior",
  [
    "When you run an instance, by default it's assigned an IP address.",
    "That IP address is not static. It will change every time the instance is
 restarted.",
    "The next step is to stop and restart your instance to demonstrate this
 behavior.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const logStartInstance = new ScenarioOutput(
  "logStartInstance",
```

```
  (/** @type { State } */ state) => `Starting instance ${state.instanceId}`,
  { skipWhen: skipWhenErrors },
);

export const startInstance = new ScenarioAction(
  "startInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StartInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
 the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logIpAllocation = new ScenarioOutput(
  "logIpAllocation",
  [
    "It is possible to have a static IP address.",
    "To demonstrate this, an IP will be allocated and associated to your EC2
 instance.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const allocateIp = new ScenarioAction(
```

```
  "allocateIp",
  async (/** @type { State } */ state) => {
    try {
      // An Elastic IP address is allocated to your AWS account, and is yours
 until you release it.
      const { AllocationId, PublicIp } = await state.ec2Client.send(
        new AllocateAddressCommand({}),
      );
      state.allocationId = AllocationId;
      state.allocatedIpAddress = PublicIp;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "MissingParameter") {
        caught.message = `${caught.message}. Did you provide these values?`;
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const associateIp = new ScenarioAction(
  "associateIp",
  async (/** @type { State } */ state) => {
    try {
      // Associate an allocated IP address to an EC2 instance. An IP address can
 be allocated
      // with the AllocateAddress action.
      const { AssociationId } = await state.ec2Client.send(
        new AssociateAddressCommand({
          AllocationId: state.allocationId,
          InstanceId: state.instanceId,
        }),
      );
      state.associationId = AssociationId;
      // Update the IP address that is being tracked to match
      // the one just associated.
      state.instanceIpAddress = state.allocatedIpAddress;
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
        caught.message = `${caught.message}. Did you provide the ID of a valid
 Elastic IP address AllocationId?`;
```

```
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logStaticIpProof = new ScenarioOutput(
  "logStaticIpProof",
  "The IP address should remain the same even after stopping and starting the
 instance.",
  { header: true, skipWhen: skipWhenErrors },
);

export const logCleanUp = new ScenarioOutput(
  "logCleanUp",
  "That's it! You can choose to clean up the resources now, or clean them up on
 your own later.",
  { header: true, skipWhen: skipWhenErrors },
);

export const confirmDisassociateAddress = new ScenarioInput(
  "confirmDisassociateAddress",
  "Do you want to disassociate and release the static IP address created
 earlier?",
  {
    type: "confirm",
    skipWhen: (/** @type { State } */ state) => !state.associationId,
  },
);

export const maybeDisassociateAddress = new ScenarioAction(
  "maybeDisassociateAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new DisassociateAddressCommand({
          AssociationId: state.associationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAssociationID.NotFound"
```

```
      ) {
        caught.message = `${caught.message}. Please provide a valid association
 ID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.associationId,
  },
);

export const maybeReleaseAddress = new ScenarioAction(
  "maybeReleaseAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new ReleaseAddressCommand({
          AllocationId: state.allocationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
        caught.message = `${caught.message}. Please provide a valid
 AllocationID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.allocationId,
  },
);

export const confirmTerminateInstance = new ScenarioInput(
  "confirmTerminateInstance",
  "Do you want to terminate the instance?",
  // Don't do anything when an instance was never run.
  {
```

```
    skipWhen: (/** @type { State } */ state) => !state.instanceId,
    type: "confirm",
  },
);

export const maybeTerminateInstance = new ScenarioAction(
  "terminateInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new TerminateInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );
      await waitUntilInstanceTerminated(
        { client: state.ec2Client },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
 the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no instance to terminate or the
    // use chooses not to terminate.
    skipWhen: (/** @type { State } */ state) =>
      !state.instanceId || !state[confirmTerminateInstance.name],
  },
);

export const deleteTemporaryDirectory = new ScenarioAction(
  "deleteTemporaryDirectory",
  async (/** @type { State } */ state) => {
    try {
      await rm(state.tmpDirectory, { recursive: true });
    } catch (caught) {
      state.errors.push(caught);
    }
  },
```

```
);

export const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    preformatted: true,
    header: true,
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances

- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks:

 1. Creates an RSA key pair and saves the private key data as a .pem file.
 2. Lists key pairs.
 3. Creates a security group for the default VPC.
 4. Displays security group information.
 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 6. Gets more information about the image.
 7. Gets a list of instance types that are compatible with the selected AMI's
 architecture.
 8. Creates an instance with the key pair, security group, AMI, and an instance
 type.
 9. Displays information about the instance.
 10. Stops the instance and waits for it to stop.
 11. Starts the instance and waits for it to start.
 12. Allocates an Elastic IP address and associates it with the instance.
 13. Displays SSH connection info for the instance.
```

```
 14. Disassociates and deletes the Elastic IP address.
 15. Terminates the instance.
 16. Deletes the security group.
 17. Deletes the key pair.
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
 Console.
            myIpAddress - The IP address of your development machine.

"""

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }

    val keyName = args[0]
    val fileName = args[1]
    val groupName = args[2]
    val groupDesc = args[3]
    val vpcId = args[4]
    val myIpAddress = args[5]
    var newInstanceId: String? = ""

    println(DASHES)
    println("Welcome to the Amazon EC2 example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an RSA key pair and save the private key material as
 a .pem file.")
```

```
    createKeyPairSc(keyName, fileName)
    println(DASHES)

    println(DASHES)
    println("2. List key pairs.")
    describeEC2KeysSc()
    println(DASHES)

    println(DASHES)
    println("3. Create a security group.")
    val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId,
myIpAddress)
    println(DASHES)

    println(DASHES)
    println("4. Display security group info for the newly created security
group.")
    describeSecurityGroupsSc(groupId.toString())
    println(DASHES)

    println(DASHES)
    println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in
the name.")
    val instanceId = getParaValuesSc()
    if (instanceId == "") {
        println("The instance Id value isn't valid.")
        exitProcess(0)
    }
    println("The instance Id is $instanceId.")
    println(DASHES)

    println(DASHES)
    println("6. Get more information about an amzn2 image and return the AMI
value.")
    val amiValue = instanceId?.let { describeImageSc(it) }
    if (instanceId == "") {
        println("The instance Id value is invalid.")
        exitProcess(0)
    }
    println("The AMI value is $amiValue.")
    println(DASHES)

    println(DASHES)
    println("7. Get a list of instance types.")
```

```
    val instanceType = getInstanceTypesSc()
    println(DASHES)

    println(DASHES)
    println("8. Create an instance.")
    if (amiValue != null) {
        newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
        println("The instance Id is $newInstanceId")
    }
    println(DASHES)

    println(DASHES)
    println("9. Display information about the running instance. ")
    var ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("10.  Stop the instance.")
    if (newInstanceId != null) {
        stopInstanceSc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("11.  Start the instance.")
    if (newInstanceId != null) {
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
 instance.")
    val allocationId = allocateAddressSc()
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
    println("The associate Id value is $associationId")
    println(DASHES)
```

```
    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("14. Disassociate and release the Elastic IP address.")
    disassociateAddressSc(associationId)
    releaseEC2AddressSc(allocationId)
    println(DASHES)

    println(DASHES)
    println("15. Terminate the instance and use a waiter.")
    if (newInstanceId != null) {
        terminateEC2Sc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("16. Delete the security group.")
    if (groupId != null) {
        deleteEC2SecGroupSc(groupId)
    }
    println(DASHES)

    println(DASHES)
    println("17. Delete the key pair.")
    deleteKeysSc(keyName)
    println(DASHES)

    println(DASHES)
    println("You successfully completed the Amazon EC2 scenario.")
    println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
```

```
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitUntilInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
```

```
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
```

```
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
                    ?.get(0)
                    ?.instances
```

```
                    ?.get(0)
                    ?.state
                    ?.name
                    ?. value
            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
 ${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
 ${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
                    println("Instance state is
 ${response.reservations!!.get(0).instances?.get(0)?.state}")
                    pubAddress =
                        response.reservations!!
                            .get(0)
                            .instances
                            ?.get(0)
                            ?.publicIpAddress
                            .toString()
                    println("Instance address is $pubAddress")
                    isRunning = true
                }
            }
        }
    }
    return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
            imageId = amiIdVal
        }
```

```
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }
```

```
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
 ${response.images?.get(0)?.description}")
        println("The name of the first image is
 ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient.fromEnvironment { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
```

```
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() +
 " and group VPC " + group.vpcId)
        }
    }
}

suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
```

```
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group
 $groupNameVal")
        return resp.groupId
    }
}

suspend fun describeEC2KeysSc() {
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class EC2InstanceScenario:
    """
    A scenario that demonstrates how to use Boto3 to manage Amazon EC2 resources.
    Covers creating a key pair, security group, launching an instance,
 associating
    an Elastic IP, and cleaning up resources.
    """

    def __init__(
        self,
        inst_wrapper: EC2InstanceWrapper,
        key_wrapper: KeyPairWrapper,
        sg_wrapper: SecurityGroupWrapper,
        eip_wrapper: ElasticIpWrapper,
        ssm_client: boto3.client,
        remote_exec: bool = False,
    ):
        """
        Initializes the EC2InstanceScenario with the necessary AWS service
 wrappers.

        :param inst_wrapper: Wrapper for EC2 instance operations.
        :param key_wrapper: Wrapper for key pair operations.
        :param sg_wrapper: Wrapper for security group operations.
        :param eip_wrapper: Wrapper for Elastic IP operations.
        :param ssm_client: Boto3 client for accessing SSM to retrieve AMIs.
        :param remote_exec: Flag to indicate if the scenario is running in a
 remote execution
                            environment. Defaults to False. If True, the script
 won't prompt
                            for user interaction.
        """
        self.inst_wrapper = inst_wrapper
        self.key_wrapper = key_wrapper
        self.sg_wrapper = sg_wrapper
        self.eip_wrapper = eip_wrapper
        self.ssm_client = ssm_client
        self.remote_exec = remote_exec

    def create_and_list_key_pairs(self) -> None:
        """
```

```
        Creates an RSA key pair for SSH access to the EC2 instance and lists
available key pairs.
        """
        console.print("**Step 1: Create a Secure Key Pair**", style="bold cyan")
        console.print(
            "Let's create a secure RSA key pair for connecting to your EC2
instance."
        )
        key_name = f"MyUniqueKeyPair-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Key Pair Name**: {key_name}")

        # Create the key pair and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Key Pair") as bar:
            self.key_wrapper.create(key_name)
            time.sleep(0.4)  # Simulate the delay in key creation
            bar()

        console.print(f"- **Private Key Saved to**:
{self.key_wrapper.key_file_path}\n")

        # List key pairs (simulated) and show a progress bar.
        list_keys = True
        if list_keys:
            console.print("- Listing your key pairs...")
            start_time = time.time()
            with alive_bar(100, title="Listing Key Pairs") as bar:
                while time.time() - start_time < 2:
                    time.sleep(0.2)
                    bar(10)
                self.key_wrapper.list(5)
                if time.time() - start_time > 2:
                    console.print(
                        "Taking longer than expected! Please wait...",
                        style="bold yellow",
                    )

    def create_security_group(self) -> None:
        """
        Creates a security group that controls access to the EC2 instance and
adds a rule
        to allow SSH access from the user's current public IP address.
        """
        console.print("**Step 2: Create a Security Group**", style="bold cyan")
        console.print(
```

```
            "Security groups manage access to your instance. Let's create one."
        )
        sg_name = f"MySecurityGroup-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Security Group Name**: {sg_name}")

        # Create the security group and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Security Group") as bar:
            self.sg_wrapper.create(
                sg_name, "Security group for example: get started with
instances."
            )
            time.sleep(0.5)
            bar()

        console.print(f"- **Security Group ID**:
{self.sg_wrapper.security_group}\n")

        # Get the current public IP to set up SSH access.
        ip_response = urllib.request.urlopen("http://checkip.amazonaws.com")
        current_ip_address = ip_response.read().decode("utf-8").strip()
        console.print(
            "Let's add a rule to allow SSH only from your current IP address."
        )
        console.print(f"- **Your Public IP Address**: {current_ip_address}")
        console.print("- Automatically adding SSH rule...")

        # Update security group rules to allow SSH and simulate with a progress
bar.
        with alive_bar(1, title="Updating Security Group Rules") as bar:
            response = self.sg_wrapper.authorize_ingress(current_ip_address)
            time.sleep(0.4)
            if response and response.get("Return"):
                console.print("- **Security Group Rules Updated**.")
            else:
                console.print(
                    "- **Error**: Couldn't update security group rules.",
                    style="bold red",
                )
            bar()

        self.sg_wrapper.describe(self.sg_wrapper.security_group)

    def create_instance(self) -> None:
        """
```

```
        Launches an EC2 instance using an Amazon Linux 2 AMI and the created key
 pair
        and security group. Displays instance details and SSH connection
 information.
        """
        # Retrieve Amazon Linux 2 AMIs from SSM.
        ami_paginator = self.ssm_client.get_paginator("get_parameters_by_path")
        ami_options = []
        for page in ami_paginator.paginate(Path="/aws/service/ami-amazon-linux-
latest"):
            ami_options += page["Parameters"]
        amzn2_images = self.inst_wrapper.get_images(
            [opt["Value"] for opt in ami_options if "amzn2" in opt["Name"]]
        )
        console.print("\n**Step 3: Launch Your Instance**", style="bold cyan")
        console.print(
            "Let's create an instance from an Amazon Linux 2 AMI. Here are some
 options:"
        )
        image_choice = 0
        console.print(f"- Selected AMI: {amzn2_images[image_choice]
['ImageId']}\n")

        # Display instance types compatible with the selected AMI
        inst_types = self.inst_wrapper.get_instance_types(
            amzn2_images[image_choice]["Architecture"]
        )
        inst_type_choice = 0
        console.print(
            f"- Selected instance type: {inst_types[inst_type_choice]
['InstanceType']}\n"
        )

        console.print("Creating your instance and waiting for it to start...")
        with alive_bar(1, title="Creating Instance") as bar:
            self.inst_wrapper.create(
                amzn2_images[image_choice]["ImageId"],
                inst_types[inst_type_choice]["InstanceType"],
                self.key_wrapper.key_pair["KeyName"],
                [self.sg_wrapper.security_group],
            )
            time.sleep(21)
            bar()
```

```
        console.print(f"**Success! Your instance is ready:**\n", style="bold
green")
        self.inst_wrapper.display()

        console.print(
            "You can use SSH to connect to your instance. "
            "If the connection attempt times out, you might have to manually
update "
            "the SSH ingress rule for your IP address in the AWS Management
Console."
        )
        self._display_ssh_info()

    def _display_ssh_info(self) -> None:
        """
        Displays SSH connection information for the user to connect to the EC2
instance.
        Handles the case where the instance does or does not have an associated
public IP address.
        """
        if (
            not self.eip_wrapper.elastic_ips
            or not self.eip_wrapper.elastic_ips[0].allocation_id
        ):
            if self.inst_wrapper.instances:
                instance = self.inst_wrapper.instances[0]
                instance_id = instance["InstanceId"]

                waiter =
self.inst_wrapper.ec2_client.get_waiter("instance_running")
                console.print(
                    "Waiting for the instance to be in a running state with a
public IP...",
                    style="bold cyan",
                )

                with alive_bar(1, title="Waiting for Instance to Start") as bar:
                    waiter.wait(InstanceIds=[instance_id])
                    time.sleep(20)
                    bar()

                instance = self.inst_wrapper.ec2_client.describe_instances(
                    InstanceIds=[instance_id]
                )["Reservations"][0]["Instances"][0]
```

```
                public_ip = instance.get("PublicIpAddress")
                if public_ip:
                    console.print(
                        "\nTo connect via SSH, open another command prompt and
 run the following command:",
                        style="bold cyan",
                    )
                    console.print(
                        f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{public_ip}"
                    )
                else:
                    console.print(
                        "Instance does not have a public IP address assigned.",
                        style="bold red",
                    )
            else:
                console.print(
                    "No instance available to retrieve public IP address.",
                    style="bold red",
                )
        else:
            elastic_ip = self.eip_wrapper.elastic_ips[0]
            elastic_ip_address = elastic_ip.public_ip
            console.print(
                f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{elastic_ip_address}"
            )

        if not self.remote_exec:
            console.print("\nOpen a new terminal tab to try the above SSH
 command.")
            input("Press Enter to continue...")

    def associate_elastic_ip(self) -> None:
        """
        Allocates an Elastic IP address and associates it with the EC2 instance.
        Displays the Elastic IP address and SSH connection information.
        """
        console.print("\n**Step 4: Allocate an Elastic IP Address**", style="bold
 cyan")
        console.print(
```

```
            "You can allocate an Elastic IP address and associate it with your
 instance\n"
            "to keep a consistent IP address even when your instance restarts."
        )

        with alive_bar(1, title="Allocating Elastic IP") as bar:
            elastic_ip = self.eip_wrapper.allocate()
            time.sleep(0.5)
            bar()

        console.print(
            f"- **Allocated Static Elastic IP Address**: {elastic_ip.public_ip}."
        )

        with alive_bar(1, title="Associating Elastic IP") as bar:
            self.eip_wrapper.associate(
                elastic_ip.allocation_id, self.inst_wrapper.instances[0]
["InstanceId"]
            )
            time.sleep(2)
            bar()

        console.print(f"- **Associated Elastic IP with Your Instance**.")
        console.print(
            "You can now use SSH to connect to your instance by using the Elastic
 IP."
        )
        self._display_ssh_info()

    def stop_and_start_instance(self) -> None:
        """
        Stops and restarts the EC2 instance. Displays instance state and explains
        changes that occur when the instance is restarted, such as the potential
 change
        in the public IP address unless an Elastic IP is associated.
        """
        console.print("\n**Step 5: Stop and Start Your Instance**", style="bold
 cyan")
        console.print("Let's stop and start your instance to see what changes.")
        console.print("- **Stopping your instance and waiting until it's
 stopped...**")

        with alive_bar(1, title="Stopping Instance") as bar:
            self.inst_wrapper.stop()
```

```
            time.sleep(360)
            bar()

        console.print("- **Your instance is stopped. Restarting...**")

        with alive_bar(1, title="Starting Instance") as bar:
            self.inst_wrapper.start()
            time.sleep(20)
            bar()

        console.print("**Your instance is running.**", style="bold green")
        self.inst_wrapper.display()

        elastic_ip = (
            self.eip_wrapper.elastic_ips[0] if self.eip_wrapper.elastic_ips else
None
        )

        if elastic_ip is None or elastic_ip.allocation_id is None:
            console.print(
                "- **Note**: Every time your instance is restarted, its public IP
address changes."
            )
        else:
            console.print(
                f"Because you have associated an Elastic IP with your instance,
you can \n"
                f"connect by using a consistent IP address after the instance
restarts: {elastic_ip.public_ip}"
            )

        self._display_ssh_info()

    def cleanup(self) -> None:
        """
        Cleans up all the resources created during the scenario, including
disassociating
        and releasing the Elastic IP, terminating the instance, deleting the
security
        group, and deleting the key pair.
        """
        console.print("\n**Step 6: Clean Up Resources**", style="bold cyan")
        console.print("Cleaning up resources:")
```

```
        for elastic_ip in self.eip_wrapper.elastic_ips:
            console.print(f"- **Elastic IP**: {elastic_ip.public_ip}")

            with alive_bar(1, title="Disassociating Elastic IP") as bar:
                self.eip_wrapper.disassociate(elastic_ip.allocation_id)
                time.sleep(2)
                bar()

            console.print("\t- **Disassociated Elastic IP from the Instance**")

            with alive_bar(1, title="Releasing Elastic IP") as bar:
                self.eip_wrapper.release(elastic_ip.allocation_id)
                time.sleep(1)
                bar()

            console.print("\t- **Released Elastic IP**")

        console.print(f"- **Instance**: {self.inst_wrapper.instances[0]
['InstanceId']}")

        with alive_bar(1, title="Terminating Instance") as bar:
            self.inst_wrapper.terminate()
            time.sleep(380)
            bar()

        console.print("\t- **Terminated Instance**")

        console.print(f"- **Security Group**: {self.sg_wrapper.security_group}")

        with alive_bar(1, title="Deleting Security Group") as bar:
            self.sg_wrapper.delete(self.sg_wrapper.security_group)
            time.sleep(1)
            bar()

        console.print("\t- **Deleted Security Group**")

        console.print(f"- **Key Pair**: {self.key_wrapper.key_pair['KeyName']}")

        with alive_bar(1, title="Deleting Key Pair") as bar:
            self.key_wrapper.delete(self.key_wrapper.key_pair["KeyName"])
            time.sleep(0.4)
            bar()

        console.print("\t- **Deleted Key Pair**")
```

```
    def run_scenario(self) -> None:
        """
        Executes the entire EC2 instance scenario: creates key pairs, security
 groups,
        launches an instance, associates an Elastic IP, and cleans up all
 resources.
        """
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
 %(message)s")

        console.print("-" * 88)
        console.print(
            "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
 with instances demo.",
            style="bold magenta",
        )
        console.print("-" * 88)

        self.create_and_list_key_pairs()
        self.create_security_group()
        self.create_instance()
        self.stop_and_start_instance()
        self.associate_elastic_ip()
        self.stop_and_start_instance()
        self.cleanup()

        console.print("\nThanks for watching!", style="bold green")
        console.print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = EC2InstanceScenario(
            EC2InstanceWrapper.from_client(),
            KeyPairWrapper.from_client(),
            SecurityGroupWrapper.from_client(),
            ElasticIpWrapper.from_client(),
            boto3.client("ssm"),
        )
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que envolva as ações de pares de chaves.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
 client
        and a temporary directory for storing key files.
```

```
        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2
instance.
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that
represents the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
                )
            raise
        else:
            return self.key_pair


    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.
```

```
        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
                      all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}'
with fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise


    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
```

```
                    f"The key pair '{key_name}' does not exist and cannot be
 deleted. "
                    "Please verify the key pair name and try again."
                )
            raise
```

Defina uma classe que envolva as ações de grupos de segurança.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of
the current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to
create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose
a different name."
                )
            raise
        else:
            return self.security_group


    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                               to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                 response indicates whether the request succeeded or failed, or
None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        if self.security_group is None:
```

```
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"
                    f"in security group '{self.security_group}'."
                )
            raise
        else:
            return response


    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                  If None, an open search is performed to
describe all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security
group(s), such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
```

```
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

            for page in page_iterator:
                for security_group in page["SecurityGroups"]:
                    print(f"Security group: {security_group['GroupName']}")
                    print(f"\tID: {security_group['GroupId']}")
                    print(f"\tVPC: {security_group['VpcId']}")
                    if security_group["IpPermissions"]:
                        print("Inbound permissions:")
                        pp(security_group["IpPermissions"])

            return True
        except ClientError as err:
            logger.error("Failed to describe security group(s).")
            if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group {security_group_id} does not exist "
                    f"because the specified security group ID was not found."
                )
            raise


    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.

        :param security_group_id: The ID of the security group to delete.
Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an
AWS service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
'{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
'{security_group_id}'")
```

```
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
 because it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
 because it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise
```

Defina uma classe que envolva as ações de instâncias.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
```

```
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
the instance.
                                   If not specified, the default security group
of the VPC is used.
        :return: A list of dictionaries representing Boto3 Instance objects
representing the newly created instances.
        """
        try:
            instance_params = {
                "ImageId": image_id,
                "InstanceType": instance_type,
                "KeyName": key_pair_name,
            }
            if security_group_ids is not None:
```

```
                instance_params["SecurityGroupIds"] = security_group_ids

            response = self.ec2_client.run_instances(
                **instance_params, MinCount=1, MaxCount=1
            )
            instance = response["Instances"][0]
            self.instances.append(instance)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=[instance["InstanceId"]])
        except ClientError as err:
            params_str = "\n\t".join(
                f"{key}: {value}" for key, value in instance_params.items()
            )
            logger.error(
                f"Failed to complete instance creation request.\nRequest details:
{params_str}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InstanceLimitExceeded":
                logger.error(
                    (
                        f"Insufficient capacity for instance type
 '{instance_type}'. "
                        "Terminate unused instances or contact AWS Support for a
 limit increase."
                    )
                )
            if error_code == "InsufficientInstanceCapacity":
                logger.error(
                    (
                        f"Insufficient capacity for instance type
 '{instance_type}'. "
                        "Select a different instance type or launch in a
 different availability zone."
                    )
                )
            raise
        return self.instances


    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.
```

```
        :param state_filter: The instance state to include in the output. Only
 instances in this state
                             will be displayed. Default is 'running'. Example
 states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:
                            continue  # Skip this instance if it doesn't match
 the filter

                        # Create a formatted string with instance details
                        instance_info = (
                            f"• ID: {instance['InstanceId']}\n"
                            f"• Image ID: {instance['ImageId']}\n"
                            f"• Instance type: {instance['InstanceType']}\n"
                            f"• Key name: {instance['KeyName']}\n"
                            f"• VPC ID: {instance['VpcId']}\n"
                            f"• Public IP: {instance.get('PublicIpAddress', 'N/
A')}\n"
                            f"• State: {instance_state}"
                        )
                        print(instance_info)

        except ClientError as err:
            logger.error(
                f"Failed to display instance(s). : {' '.join(map(str,
 instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
```

```
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
                raise


    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
            raise


    def start(self) -> Optional[Dict[str, Any]]:
        """
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
```

```
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=instance_ids)
            return start_response
        except ClientError as err:
            logger.error(
                f"Failed to start instance(s): {','.join(map(str,
instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't start instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a stopped state before starting
them."
                )
            raise


    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no
instances to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
```

```
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response


    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str,
image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images


    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
"*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
```

```
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
API_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types.
 Default: 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support
 the specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture},
 {','.join(map(str, sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "
                    "Ensure architecture and size strings conform to
 DescribeInstanceTypes API reference."
                )
            raise
        else:
            return inst_types
```

Defina uma classe que envolva as ações de IP elástico.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
```

```
        return cls(ec2_client)


    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP
address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the
maximum limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
Support for an increase."
                )
            raise err
        return elastic_ip


    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association
is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
```

```
        :return: A response that contains the ID of the association, or None if
no Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance
ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id  # Track the instance associated with this Elastic
IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
                )
            raise
        return response


    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance.
When the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to
disassociate.
```

```
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(
                f"No association found for Elastic IP with allocation ID
{allocation_id}."
            )
            return

        try:
            # Retrieve the association ID before disassociating
            response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
            association_id = response["Addresses"][0].get("AssociationId")

            if association_id:

self.ec2_client.disassociate_address(AssociationId=association_id)
                elastic_ip.instance_id = None  # Remove the instance association
            else:
                logger.info(
                    f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
                )

        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
                logger.error(
                    f"Failed to disassociate Elastic IP {allocation_id} "
                    "because the specified association ID for the Elastic IP
address was not found. "
                    "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                    "resource before attempting to disassociate it."
                )
            raise


    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
```

```
        it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip)  # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address
"
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise


    @staticmethod
    def get_elastic_ip_by_allocation(
        elastic_ips: List["ElasticIpWrapper.ElasticIp"], allocation_id: str
    ) -> Optional["ElasticIpWrapper.ElasticIp"]:
        """
        Retrieves an Elastic IP object by its allocation ID from a given list of
Elastic IPs.

        :param elastic_ips: A list of ElasticIp objects.
        :param allocation_id: The allocation ID of the Elastic IP to retrieve.
        :return: The ElasticIp object associated with the allocation ID, or None
if not found.
        """
        return next(
```

```
            (ip for ip in elastic_ips if ip.allocation_id == allocation_id), None
        )
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A EC2InstanceScenario implementação contém lógica para executar o exemplo como um todo.

```
//! Scenario that uses the AWS SDK for Rust (the SDK) with Amazon Elastic Compute
 Cloud
//! (Amazon EC2) to do the following:
//!
//! * Create a key pair that is used to secure SSH communication between your
 computer and
//!   an EC2 instance.
//! * Create a security group that acts as a virtual firewall for your EC2
 instances to
//!   control incoming and outgoing traffic.
//! * Find an Amazon Machine Image (AMI) and a compatible instance type.
//! * Create an instance that is created from the instance type and AMI you
 select, and
//!   is configured to use the security group and key pair created in this
 example.
//! * Stop and restart the instance.
//! * Create an Elastic IP address and associate it as a consistent IP address
 for your instance.
//! * Connect to your instance with SSH, using both its public IP address and
 your Elastic IP
//!   address.
//! * Clean up all of the resources created by this example.

use std::net::Ipv4Addr;

use crate::{
    ec2::{EC2Error, EC2},
    getting_started::{key_pair::KeyPairManager, util::Util},
```

```
    ssm::SSM,
};
use aws_sdk_ssm::types::Parameter;

use super::{
    elastic_ip::ElasticIpManager, instance::InstanceManager,
 security_group::SecurityGroupManager,
    util::ScenarioImage,
};

pub struct Ec2InstanceScenario {
    ec2: EC2,
    ssm: SSM,
    util: Util,
    key_pair_manager: KeyPairManager,
    security_group_manager: SecurityGroupManager,
    instance_manager: InstanceManager,
    elastic_ip_manager: ElasticIpManager,
}

impl Ec2InstanceScenario {
    pub fn new(ec2: EC2, ssm: SSM, util: Util) -> Self {
        Ec2InstanceScenario {
            ec2,
            ssm,
            util,
            key_pair_manager: Default::default(),
            security_group_manager: Default::default(),
            instance_manager: Default::default(),
            elastic_ip_manager: Default::default(),
        }
    }

    pub async fn run(&mut self) -> Result<(), EC2Error> {
        self.create_and_list_key_pairs().await?;
        self.create_security_group().await?;
        self.create_instance().await?;
        self.stop_and_start_instance().await?;
        self.associate_elastic_ip().await?;
        self.stop_and_start_instance().await?;
        Ok(())
    }
```

```
    /// 1. Creates an RSA key pair and saves its private key data as a .pem file
in secure
    ///    temporary storage. The private key data is deleted after the example
completes.
    /// 2. Optionally, lists the first five key pairs for the current account.
    pub async fn create_and_list_key_pairs(&mut self) -> Result<(), EC2Error> {
        println!( "Let's create an RSA key pair that you can be use to securely
connect to your EC2 instance.");

        let key_name = self.util.prompt_key_name()?;

        self.key_pair_manager
            .create(&self.ec2, &self.util, key_name)
            .await?;

        println!(
            "Created a key pair {} and saved the private key to {:?}.",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .ok_or_else(|| EC2Error::new("No key name after creating key"))?,
            self.key_pair_manager
                .key_file_path()
                .ok_or_else(|| EC2Error::new("No key file after creating key"))?
        );

        if self.util.should_list_key_pairs()? {
            for pair in self.key_pair_manager.list(&self.ec2).await? {
                println!(
                    "Found {:?} key {} with fingerprint:\t{:?}",
                    pair.key_type(),
                    pair.key_name().unwrap_or("Unknown"),
                    pair.key_fingerprint()
                );
            }
        }

        Ok(())
    }

    /// 1. Creates a security group for the default VPC.
    /// 2. Adds an inbound rule to allow SSH. The SSH rule allows only
    ///    inbound traffic from the current computer’s public IPv4 address.
    /// 3. Displays information about the security group.
```

```
    ///
    /// This function uses <http://checkip.amazonaws.com> to get the current
public IP
    /// address of the computer that is running the example. This method works in
most
    /// cases. However, depending on how your computer connects to the internet,
you
    /// might have to manually add your public IP address to the security group
by using
    /// the AWS Management Console.
    pub async fn create_security_group(&mut self) -> Result<(), EC2Error> {
        println!("Let's create a security group to manage access to your
instance.");
        let group_name = self.util.prompt_security_group_name()?;

        self.security_group_manager
            .create(
                &self.ec2,
                &group_name,
                "Security group for example: get started with instances.",
            )
            .await?;

        println!(
            "Created security group {} in your default VPC {}.",
            self.security_group_manager.group_name(),
            self.security_group_manager
                .vpc_id()
                .unwrap_or("(unknown vpc)")
        );

        let check_ip = self.util.do_get("https://checkip.amazonaws.com").await?;
        let current_ip_address: Ipv4Addr = check_ip.trim().parse().map_err(|e| {
            EC2Error::new(format!(
                "Failed to convert response {} to IP Address: {e:?}",
                check_ip
            ))
        })?;

        println!("Your public IP address seems to be {current_ip_address}");
        if self.util.should_add_to_security_group() {
            match self
                .security_group_manager
                .authorize_ingress(&self.ec2, current_ip_address)
```

```
                .await
            {
                Ok(_) => println!("Security group rules updated"),
                Err(err) => eprintln!("Couldn't update security group rules:
{err:?}"),
            }
        }
        println!("{}", self.security_group_manager);

        Ok(())
    }

    /// 1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager.
Specifying the
    ///    '/aws/service/ami-amazon-linux-latest' path returns only the latest
AMIs.
    /// 2. Gets and displays information about the available AMIs and lets you
select one.
    /// 3. Gets a list of instance types that are compatible with the selected
AMI and
    ///    lets you select one.
    /// 4. Creates an instance with the previously created key pair and security
group,
    ///    and the selected AMI and instance type.
    /// 5. Waits for the instance to be running and then displays its
information.
    pub async fn create_instance(&mut self) -> Result<(), EC2Error> {
        let ami = self.find_image().await?;

        let instance_types = self
            .ec2
            .list_instance_types(&ami.0)
            .await
            .map_err(|e| e.add_message("Could not find instance types"))?;
        println!(
            "There are several instance types that support the {} architecture of
the image.",
            ami.0
                .architecture
                .as_ref()
                .ok_or_else(|| EC2Error::new(format!("Missing architecture in
{:?}", ami.0)))?
        );
        let instance_type = self.util.select_instance_type(instance_types)?;
```

```
        println!("Creating your instance and waiting for it to start...");
        self.instance_manager
            .create(
                &self.ec2,
                ami.0
                    .image_id()
                    .ok_or_else(|| EC2Error::new("Could not find image ID"))?,
                instance_type,
                self.key_pair_manager.key_pair(),
                self.security_group_manager
                    .security_group()
                    .map(|sg| vec![sg])
                    .ok_or_else(|| EC2Error::new("Could not find security
group"))?,
            )
            .await
            .map_err(|e| e.add_message("Scenario failed to create instance"))?;

        while let Err(err) = self
            .ec2
            .wait_for_instance_ready(self.instance_manager.instance_id(), None)
            .await
        {
            println!("{err}");
            if !self.util.should_continue_waiting() {
                return Err(err);
            }
        }

        println!("Your instance is ready:\n{}", self.instance_manager);

        self.display_ssh_info();

        Ok(())
    }

    async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
        let params: Vec<Parameter> = self
            .ssm
            .list_path("/aws/service/ami-amazon-linux-latest")
            .await
            .map_err(|e| e.add_message("Could not find parameters for available
images"))?
```

```
            .into_iter()
            .filter(|param| param.name().is_some_and(|name|
name.contains("amzn2")))
            .collect();
        let amzn2_images: Vec<ScenarioImage> = self
            .ec2
            .list_images(params)
            .await
            .map_err(|e| e.add_message("Could not find images"))?
            .into_iter()
            .map(ScenarioImage::from)
            .collect();
        println!("We will now create an instance from an Amazon Linux 2 AMI");
        let ami = self.util.select_scenario_image(amzn2_images)?;
        Ok(ami)
    }

    // 1. Stops the instance and waits for it to stop.
    // 2. Starts the instance and waits for it to start.
    // 3. Displays information about the instance.
    // 4. Displays an SSH connection string. When an Elastic IP address is
associated
    //    with the instance, the IP address stays consistent when the instance
stops
    //    and starts.
    pub async fn stop_and_start_instance(&self) -> Result<(), EC2Error> {
        println!("Let's stop and start your instance to see what changes.");
        println!("Stopping your instance and waiting until it's stopped...");
        self.instance_manager.stop(&self.ec2).await?;
        println!("Your instance is stopped. Restarting...");
        self.instance_manager.start(&self.ec2).await?;
        println!("Your instance is running.");
        println!("{}", self.instance_manager);
        if self.elastic_ip_manager.public_ip() == "0.0.0.0" {
            println!("Every time your instance is restarted, its public IP
address changes.");
        } else {
            println!(
                "Because you have associated an Elastic IP with your instance,
you can connect by using a consistent IP address after the instance restarts."
            );
        }
        self.display_ssh_info();
        Ok(())
```

```
    }

    /// 1. Allocates an Elastic IP address and associates it with the instance.
    /// 2. Displays an SSH connection string that uses the Elastic IP address.
    async fn associate_elastic_ip(&mut self) -> Result<(), EC2Error> {
        self.elastic_ip_manager.allocate(&self.ec2).await?;
        println!(
            "Allocated static Elastic IP address: {}",
            self.elastic_ip_manager.public_ip()
        );

        self.elastic_ip_manager
            .associate(&self.ec2, self.instance_manager.instance_id())
            .await?;
        println!("Associated your Elastic IP with your instance.");
        println!("You can now use SSH to connect to your instance by using the
Elastic IP.");
        self.display_ssh_info();
        Ok(())
    }

    /// Displays an SSH connection string that can be used to connect to a
running
    /// instance.
    fn display_ssh_info(&self) {
        let ip_addr = if self.elastic_ip_manager.has_allocation() {
            self.elastic_ip_manager.public_ip()
        } else {
            self.instance_manager.instance_ip()
        };
        let key_file_path = self.key_pair_manager.key_file_path().unwrap();
        println!("To connect, open another command prompt and run the following
command:");
        println!("\nssh -i {} ec2-user@{ip_addr}\n", key_file_path.display());
        let _ = self.util.enter_to_continue();
    }

    /// 1. Disassociate and delete the previously created Elastic IP.
    /// 2. Terminate the previously created instance.
    /// 3. Delete the previously created security group.
    /// 4. Delete the previously created key pair.
    pub async fn clean_up(self) {
        println!("Let's clean everything up. This example created these
resources:");
```

```
        println!(
            "\tKey pair: {}",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .unwrap_or("(unknown key pair)")
        );
        println!(
            "\tSecurity group: {}",
            self.security_group_manager.group_name()
        );
        println!(
            "\tInstance: {}",
            self.instance_manager.instance_display_name()
        );
        if self.util.should_clean_resources() {
            if let Err(err) = self.elastic_ip_manager.remove(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.instance_manager.delete(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.security_group_manager.delete(&self.ec2).await
 {
                eprintln!("{err}");
            }
            if let Err(err) = self.key_pair_manager.delete(&self.ec2,
 &self.util).await {
                eprintln!("{err}");
            }
        } else {
            println!("Ok, not cleaning up any resources!");
        }
    }
}

pub async fn run(mut scenario: Ec2InstanceScenario) {
    println!
("----------------------------------------------------------------------------");
    println!(
        "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
 with instances demo."
    );
```

```
    println!
("--------------------------------------------------------------------------------");

    if let Err(err) = scenario.run().await {
        eprintln!("There was an error running the scenario: {err}")
    }

    println!
("--------------------------------------------------------------------------------");

    scenario.clean_up().await;

    println!("Thanks for running!");
    println!
("--------------------------------------------------------------------------------");
}
```

O struct EC2Impl serve como um ponto de controle automático para testes e suas funções envolvem as chamadas do SDK do EC2.

```
use std::{net::Ipv4Addr, time::Duration};

use aws_sdk_ec2::{
    client::Waiters,
    error::ProvideErrorMetadata,
    operation::{
        allocate_address::AllocateAddressOutput,
 associate_address::AssociateAddressOutput,
    },
    types::{
        DomainType, Filter, Image, Instance, InstanceType, IpPermission, IpRange,
 KeyPairInfo,
        SecurityGroup, Tag,
    },
    Client as EC2Client,
};
use aws_sdk_ssm::types::Parameter;
use aws_smithy_runtime_api::client::waiters::error::WaiterError;

#[cfg(test)]
use mockall::automock;
```

```
#[cfg(not(test))]
pub use EC2Impl as EC2;

#[cfg(test)]
pub use MockEC2Impl as EC2;

#[derive(Clone)]
pub struct EC2Impl {
    pub client: EC2Client,
}

#[cfg_attr(test, automock)]
impl EC2Impl {
    pub fn new(client: EC2Client) -> Self {
        EC2Impl { client }
    }

    pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
 String), EC2Error> {
        tracing::info!("Creating key pair {name}");
        let output = self.client.create_key_pair().key_name(name).send().await?;
        let info = KeyPairInfo::builder()
            .set_key_name(output.key_name)
            .set_key_fingerprint(output.key_fingerprint)
            .set_key_pair_id(output.key_pair_id)
            .build();
        let material = output
            .key_material
            .ok_or_else(|| EC2Error::new("Create Key Pair has no key
 material"))?;
        Ok((info, material))
    }

    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }

    pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
        let key_name: String = key_name.into();
        tracing::info!("Deleting key pair {key_name}");
        self.client
            .delete_key_pair()
```

```
            .key_name(key_name)
            .send()
            .await?;
        Ok(())
    }

    pub async fn create_security_group(
        &self,
        name: &str,
        description: &str,
    ) -> Result<SecurityGroup, EC2Error> {
        tracing::info!("Creating security group {name}");
        let create_output = self
            .client
            .create_security_group()
            .group_name(name)
            .description(description)
            .send()
            .await
            .map_err(EC2Error::from)?;

        let group_id = create_output
            .group_id
            .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

        let group = self
            .describe_security_group(&group_id)
            .await?
            .ok_or_else(|| {
                EC2Error::new(format!("Could not find security group with id
{group_id}"))
            })?;

        tracing::info!("Created security group {name} as {group_id}");

        Ok(group)
    }

    /// Find a single security group, by ID. Returns Err if multiple groups are
found.
    pub async fn describe_security_group(
        &self,
        group_id: &str,
```

```
    ) -> Result<Option<SecurityGroup>, EC2Error> {
        let group_id: String = group_id.into();
        let describe_output = self
            .client
            .describe_security_groups()
            .group_ids(&group_id)
            .send()
            .await?;

        let mut groups = describe_output.security_groups.unwrap_or_default();

        match groups.len() {
            0 => Ok(None),
            1 => Ok(Some(groups.remove(0))),
            _ => Err(EC2Error::new(format!(
                "Expected single group for {group_id}"
            ))),
        }
    }

    /// Add an ingress rule to a security group explicitly allowing IPv4 address
    /// as {ip}/32 over TCP port 22.
    pub async fn authorize_security_group_ssh_ingress(
        &self,
        group_id: &str,
        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
                            .build()
                    })
                    .collect(),
```

```
            ))
            .send()
            .await?;
        Ok(())
    }

    pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Deleting security group {group_id}");
        self.client
            .delete_security_group()
            .group_id(group_id)
            .send()
            .await?;
        Ok(())
    }

    pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
        let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
        let output = self
            .client
            .describe_images()
            .set_image_ids(Some(image_ids))
            .send()
            .await?;

        let images = output.images.unwrap_or_default();
        if images.is_empty() {
            Err(EC2Error::new("No images for selected AMIs"))
        } else {
            Ok(images)
        }
    }

    /// List instance types that match an image's architecture and are free tier
eligible.
    pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
        let architecture = format!(
            "{}",
            image.architecture().ok_or_else(|| EC2Error::new(format!(
                "Image {:?} does not have a listed architecture",
                image.image_id()
```

```
            )))?
        );
        let free_tier_eligible_filter = Filter::builder()
            .name("free-tier-eligible")
            .values("false")
            .build();
        let supported_architecture_filter = Filter::builder()
            .name("processor-info.supported-architecture")
            .values(architecture)
            .build();
        let response = self
            .client
            .describe_instance_types()
            .filters(free_tier_eligible_filter)
            .filters(supported_architecture_filter)
            .send()
            .await?;

        Ok(response
            .instance_types
            .unwrap_or_default()
            .into_iter()
            .filter_map(|iti| iti.instance_type)
            .collect())
    }

    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
```

```
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }

    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
```

```
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
        let response = self
            .client
            .describe_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        let instance = response
            .reservations()
            .first()
            .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
            .instances()
            .first()
            .ok_or_else(|| {
                EC2Error::new(format!("No instances in reservation for
{instance_id}"))
            })?;

        Ok(instance.clone())
    }
```

```
    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }

    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }

    pub async fn reboot_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Rebooting instance {instance_id}");

        self.client
            .reboot_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        Ok(())
    }
```

```
    pub async fn wait_for_instance_stopped(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_stopped()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to stop.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn delete_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Deleting instance with id {instance_id}");
        self.stop_instance(instance_id).await?;
        self.client
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await?;
        self.wait_for_instance_terminated(instance_id).await?;
        tracing::info!("Terminated instance with id {instance_id}");
        Ok(())
    }

    async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
        self.client
            .wait_until_instance_terminated()
            .instance_ids(instance_id)
            .wait(Duration::from_secs(60))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
```

```
                    "Exceeded max time ({}s) waiting for instance to terminate.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
        self.client
            .allocate_address()
            .domain(DomainType::Vpc)
            .send()
            .await
            .map_err(EC2Error::from)
    }

    pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
        self.client
            .release_address()
            .allocation_id(allocation_id)
            .send()
            .await?;
        Ok(())
    }

    pub async fn associate_ip_address(
        &self,
        allocation_id: &str,
        instance_id: &str,
    ) -> Result<AssociateAddressOutput, EC2Error> {
        let response = self
            .client
            .associate_address()
            .allocation_id(allocation_id)
            .instance_id(instance_id)
            .send()
            .await?;
        Ok(response)
    }
```

```
    pub async fn disassociate_ip_address(&self, association_id: &str) ->
 Result<(), EC2Error> {
        self.client
            .disassociate_address()
            .association_id(association_id)
            .send()
            .await?;
        Ok(())
    }
}

#[derive(Debug)]
pub struct EC2Error(String);
impl EC2Error {
    pub fn new(value: impl Into<String>) -> Self {
        EC2Error(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        EC2Error(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: ProvideErrorMetadata> From<T> for EC2Error {
    fn from(value: T) -> Self {
        EC2Error(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for EC2Error {}

impl std::fmt::Display for EC2Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
```

```
    }
}
```

O struct SSM serve como um ponto de controle automático para testes e suas funções envolvem as chamadas do SDK do SSM.

```
use aws_sdk_ssm::{types::Parameter, Client};
use aws_smithy_async::future::pagination_stream::TryFlatMap;

use crate::ec2::EC2Error;

#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use SSMImpl as SSM;

#[cfg(test)]
pub use MockSSMImpl as SSM;

pub struct SSMImpl {
    inner: Client,
}

#[cfg_attr(test, automock)]
impl SSMImpl {
    pub fn new(inner: Client) -> Self {
        SSMImpl { inner }
    }

    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error>
 {
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
```

```
        .await;
        // Fail on the first error
        let params = maybe_params
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
    }
}
```

O cenário usa várias estruturas no estilo “Gerenciador” para lidar com o acesso aos recursos que são criados e excluídos em todo o cenário.

```
use aws_sdk_ec2::operation::{
    allocate_address::AllocateAddressOutput,
 associate_address::AssociateAddressOutput,
};

use crate::ec2::{EC2Error, EC2};

/// ElasticIpManager tracks the lifecycle of a public IP address, including its
/// allocation from the global pool and association with a specific instance.
#[derive(Debug, Default)]
pub struct ElasticIpManager {
    elastic_ip: Option<AllocateAddressOutput>,
    association: Option<AssociateAddressOutput>,
}

impl ElasticIpManager {
    pub fn has_allocation(&self) -> bool {
        self.elastic_ip.is_some()
    }

    pub fn public_ip(&self) -> &str {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(addr) = allocation.public_ip() {
                return addr;
            }
        }
        "0.0.0.0"
    }
```

```
    pub async fn allocate(&mut self, ec2: &EC2) -> Result<(), EC2Error> {
        let allocation = ec2.allocate_ip_address().await?;
        self.elastic_ip = Some(allocation);
        Ok(())
    }

    pub async fn associate(&mut self, ec2: &EC2, instance_id: &str) -> Result<(),
 EC2Error> {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                let association = ec2.associate_ip_address(allocation_id,
 instance_id).await?;
                self.association = Some(association);
                return Ok(());
            }
        }
        Err(EC2Error::new("No ip address allocation to associate"))
    }

    pub async fn remove(mut self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(association) = &self.association {
            if let Some(association_id) = association.association_id() {
                ec2.disassociate_ip_address(association_id).await?;
            }
        }
        self.association = None;
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                ec2.deallocate_ip_address(allocation_id).await?;
            }
        }
        self.elastic_ip = None;
        Ok(())
    }
}


use std::fmt::Display;

use aws_sdk_ec2::types::{Instance, InstanceType, KeyPairInfo, SecurityGroup};

use crate::ec2::{EC2Error, EC2};

/// InstanceManager wraps the lifecycle of an EC2 Instance.
```

```
#[derive(Debug, Default)]
pub struct InstanceManager {
    instance: Option<Instance>,
}

impl InstanceManager {
    pub fn instance_id(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(id) = instance.instance_id() {
                return id;
            }
        }
        "Unknown"
    }

    pub fn instance_name(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(tag) = instance.tags().iter().find(|e| e.key() ==
 Some("Name")) {
                if let Some(value) = tag.value() {
                    return value;
                }
            }
        }
        "Unknown"
    }

    pub fn instance_ip(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(public_ip_address) = instance.public_ip_address() {
                return public_ip_address;
            }
        }
        "0.0.0.0"
    }

    pub fn instance_display_name(&self) -> String {
        format!("{} ({})", self.instance_name(), self.instance_id())
    }

    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
```

```
        ec2: &EC2,
        image_id: &str,
        instance_type: InstanceType,
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }

    /// Start the managed EC2 instance, if present.
    pub async fn start(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.start_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    /// Stop the managed EC2 instance, if present.
    pub async fn stop(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.stop_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.reboot_instance(self.instance_id()).await?;
            ec2.wait_for_instance_stopped(self.instance_id(), None)
                .await?;
            ec2.wait_for_instance_ready(self.instance_id(), None)
                .await?;
        }
        Ok(())
    }

    /// Terminate and delete the managed EC2 instance, if present.
    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
```

```
            ec2.delete_instance(self.instance_id()).await?;
        }
        Ok(())
    }
}

impl Display for InstanceManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        if let Some(instance) = &self.instance {
            writeln!(f, "\tID: {}",
 instance.instance_id().unwrap_or("(Unknown)"))?;
            writeln!(
                f,
                "\tImage ID: {}",
                instance.image_id().unwrap_or("(Unknown)")
            )?;
            writeln!(
                f,
                "\tInstance type: {}",
                instance
                    .instance_type()
                    .map(|it| format!("{it}"))
                    .unwrap_or("(Unknown)".to_string())
            )?;
            writeln!(
                f,
                "\tKey name: {}",
                instance.key_name().unwrap_or("(Unknown)")
            )?;
            writeln!(f, "\tVPC ID: {}",
 instance.vpc_id().unwrap_or("(Unknown)"))?;
            writeln!(
                f,
                "\tPublic IP: {}",
                instance.public_ip_address().unwrap_or("(Unknown)")
            )?;
            let instance_state = instance
                .state
                .as_ref()
                .map(|is| {
                    is.name()
                        .map(|isn| format!("{isn}"))
                        .unwrap_or("(Unknown)".to_string())
                })
```

```
                .unwrap_or("(Unknown)".to_string());
            writeln!(f, "\tState: {instance_state}")?;
        } else {
            writeln!(f, "\tNo loaded instance")?;
        }
        Ok(())
    }
}


use std::{env, path::PathBuf};

use aws_sdk_ec2::types::KeyPairInfo;

use crate::ec2::{EC2Error, EC2};

use super::util::Util;

/// KeyPairManager tracks a KeyPairInfo and the path the private key has been
/// written to, if it's been created.
#[derive(Debug)]
pub struct KeyPairManager {
    key_pair: KeyPairInfo,
    key_file_path: Option<PathBuf>,
    key_file_dir: PathBuf,
}

impl KeyPairManager {
    pub fn new() -> Self {
        Self::default()
    }

    pub fn key_pair(&self) -> &KeyPairInfo {
        &self.key_pair
    }

    pub fn key_file_path(&self) -> Option<&PathBuf> {
        self.key_file_path.as_ref()
    }

    pub fn key_file_dir(&self) -> &PathBuf {
        &self.key_file_dir
    }
```

```
   /// Creates a key pair that can be used to securely connect to an EC2
instance.
   /// The returned key pair contains private key information that cannot be
retrieved
   /// again. The private key data is stored as a .pem file.
   ///
   /// :param key_name: The name of the key pair to create.
   pub async fn create(
       &mut self,
       ec2: &EC2,
       util: &Util,
       key_name: String,
   ) -> Result<KeyPairInfo, EC2Error> {
       let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
           self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
           e.add_message(format!("Couldn't create key {key_name}"))
       })?;

       let path = self.key_file_dir.join(format!("{key_name}.pem"));

       // Save the key_pair information immediately, so it can get cleaned up if
write_secure fails.
       self.key_file_path = Some(path.clone());
       self.key_pair = key_pair.clone();

       util.write_secure(&key_name, &path, material)?;

       Ok(key_pair)
   }

   pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
       if let Some(key_name) = self.key_pair.key_name() {
           ec2.delete_key_pair(key_name).await?;
           if let Some(key_path) = self.key_file_path() {
               if let Err(err) = util.remove(key_path) {
                   eprintln!("Failed to remove {key_path:?} ({err:?})");
               }
           }
       }
       Ok(())
   }
```

```
    pub async fn list(&self, ec2: &EC2) -> Result<Vec<KeyPairInfo>, EC2Error> {
        ec2.list_key_pair().await
    }
}

impl Default for KeyPairManager {
    fn default() -> Self {
        KeyPairManager {
            key_pair: KeyPairInfo::builder().build(),
            key_file_path: Default::default(),
            key_file_dir: env::temp_dir(),
        }
    }
}


use std::net::Ipv4Addr;

use aws_sdk_ec2::types::SecurityGroup;

use crate::ec2::{EC2Error, EC2};

/// SecurityGroupManager tracks the lifecycle of a SecurityGroup for an instance,
/// including adding a rule to allow SSH from a public IP address.
#[derive(Debug, Default)]
pub struct SecurityGroupManager {
    group_name: String,
    group_description: String,
    security_group: Option<SecurityGroup>,
}

impl SecurityGroupManager {
    pub async fn create(
        &mut self,
        ec2: &EC2,
        group_name: &str,
        group_description: &str,
    ) -> Result<(), EC2Error> {
        self.group_name = group_name.into();
        self.group_description = group_description.into();

        self.security_group = Some(
            ec2.create_security_group(group_name, group_description)
                .await
```

```
                .map_err(|e| e.add_message("Couldn't create security group"))?,
        );

        Ok(())
    }

    pub async fn authorize_ingress(&self, ec2: &EC2, ip_address: Ipv4Addr) ->
 Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.authorize_security_group_ssh_ingress(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
                vec![ip_address],
            )
            .await?;
        };

        Ok(())
    }

    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.delete_security_group(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
            )
            .await?;
        };

        Ok(())
    }

    pub fn group_name(&self) -> &str {
        &self.group_name
    }

    pub fn vpc_id(&self) -> Option<&str> {
        self.security_group.as_ref().and_then(|sg| sg.vpc_id())
    }

    pub fn security_group(&self) -> Option<&SecurityGroup> {
        self.security_group.as_ref()
    }
}
```

```
impl std::fmt::Display for SecurityGroupManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.security_group {
            Some(sg) => {
                writeln!(
                    f,
                    "Security group: {}",
                    sg.group_name().unwrap_or("(unknown group)")
                )?;
                writeln!(f, "\tID: {}", sg.group_id().unwrap_or("(unknown group
 id)"))?;
                writeln!(f, "\tVPC: {}", sg.vpc_id().unwrap_or("(unknown group
 vpc)"))?;
                if !sg.ip_permissions().is_empty() {
                    writeln!(f, "\tInbound Permissions:")?;
                    for permission in sg.ip_permissions() {
                        writeln!(f, "\t\t{permission:?}")?;
                    }
                }
                Ok(())
            }
            None => writeln!(f, "No security group loaded."),
        }
    }
}
```

O principal ponto de entrada para o cenário.

```
use ec2_code_examples::{
    ec2::EC2,
    getting_started::{
        scenario::{run, Ec2InstanceScenario},
        util::UtilImpl,
    },
    ssm::SSM,
};

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
```

```
    let sdk_config = aws_config::load_from_env().await;
    let ec2 = EC2::new(aws_sdk_ec2::Client::new(&sdk_config));
    let ssm = SSM::new(aws_sdk_ssm::Client::new(&sdk_config));
    let util = UtilImpl {};
    let scenario = Ec2InstanceScenario::new(ec2, ssm, util);
    run(scenario).await;
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "ec2-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "ec2-scenario",
```

```
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "AWSSSM", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation
import AWSEC2

// Allow waiters to be used.

import class SmithyWaitersAPI.Waiter
import struct SmithyWaitersAPI.WaiterOptions

import AWSSSM

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
```

```
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "ec2-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon EC2 using
 the
        AWS SDK for Swift.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ssmConfig = try await SSMClient.SSMClientConfiguration(region:
 awsRegion)
        let ssmClient = SSMClient(config: ssmConfig)

        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
 awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let example = Example(ec2Client: ec2Client, ssmClient: ssmClient)

        await example.run()
    }
}

class Example {
    let ec2Client: EC2Client
    let ssmClient: SSMClient

    // Storage for AWS EC2 properties.

    var keyName: String? = nil
    var securityGroupId: String? = nil
    var instanceId: String? = nil
    var allocationId: String? = nil
    var associationId: String? = nil

    init(ec2Client: EC2Client, ssmClient: SSMClient) {
```

```
        self.ec2Client = ec2Client
        self.ssmClient = ssmClient
    }

    /// The example's main body.
    func run() async {
        //=====================================================================
        // 1. Create an RSA key pair, saving the private key as a `.pem` file.
        //    Create a `defer` block that will delete the private key when the
        //    program exits.
        //=====================================================================

        print("Creating an RSA key pair...")

        keyName = self.tempName(prefix: "ExampleKeyName")
        let keyUrl = await self.createKeyPair(name: keyName!)

        guard let keyUrl else {
            print("*** Failed to create the key pair!")
            return
        }

        print("Created the private key at: \(keyUrl.absoluteString)")

        // Schedule deleting the private key file to occur automatically when
        // the program exits, no matter how it exits.

        defer {
            do {
                try FileManager.default.removeItem(at: keyUrl)
            } catch {
                print("*** Failed to delete the private key at
\(keyUrl.absoluteString)")
            }
        }

        //=====================================================================
        // 2. List the key pairs by calling `DescribeKeyPairs`.
        //=====================================================================

        print("Describing available key pairs...")
        await self.describeKeyPairs()

        //=====================================================================
```

```
        // 3. Create a security group for the default VPC, and add an inbound
        //    rule to allow SSH from the current computer's public IPv4
        //    address.
        //======================================================================

        print("Creating the security group...")

        let secGroupName = self.tempName(prefix: "ExampleSecurityGroup")
        let ipAddress = self.getMyIPAddress()

        guard let ipAddress else {
            print("*** Unable to get the device's IP address.")
            return
        }

        print("IP address is: \(ipAddress)")

        securityGroupId = await self.createSecurityGroup(
            name: secGroupName,
            description: "An example security group created using the AWS SDK for
Swift"
        )

        if securityGroupId == nil {
            await cleanUp()
            return
        }

        print("Created security group: \(securityGroupId ?? "<unknown>")")

        if !(await self.authorizeSecurityGroupIngress(groupId: securityGroupId!,
ipAddress: ipAddress)) {
            await cleanUp()
            return
        }

        //======================================================================
        // 4. Display security group information for the new security group
        //    using DescribeSecurityGroups.
        //======================================================================

        if !(await self.describeSecurityGroups(groupId: securityGroupId!)) {
            await cleanUp()
            return
```

```
        }

        //=====================================================================
        // 5. Get a list of Amazon Linux 2023 AMIs and pick one (SSM is the
        //    best practice), using path and then filter the list after the
        //    fact to include "al2023" in the Name field
        //    (ssm.GetParametersByPath). Paginate to get all images.
        //=====================================================================

        print("Searching available images for Amazon Linux 2023 images...")

        let options = await self.findAMIsMatchingFilter("al2023")

        //=====================================================================
        // 6. The information in the AMI options isn't great, so make a list
        //    of the image IDs (the "Value" field in the AMI options) and get
        //    more information about them from EC2. Display the Description
        //    field and select one of them (DescribeImages with ImageIds
        //    filter).
        //=====================================================================

        print("Images matching Amazon Linux 2023:")

        var imageIds: [String] = []
        for option in options {
            guard let id = option.value else {
                continue
            }
            imageIds.append(id)
        }

        let images = await self.describeImages(imageIds)

        // This is where you would normally let the user choose which AMI to
        // use. However, for this example, we're just going to use the first
        // one, whatever it is.

        let chosenImage = images[0]

        //=====================================================================
        // 7. Get a list of instance types that are compatible with the
        //    selected AMI's architecture (such as "x86_64") and are either
        //    small or micro. Select one (DescribeInstanceTypes).
        //=====================================================================
```

```
        print("Getting the instance types compatible with the selected image...")

        guard let arch = chosenImage.architecture else {
            print("*** The selected image doesn't have a valid architecture.")
            await cleanUp()
            return
        }

        let imageTypes = await self.getMatchingInstanceTypes(architecture: arch)

        for type in imageTypes {
            guard let instanceType = type.instanceType else {
                continue
            }
            print("    \(instanceType.rawValue)")
        }

        // This example selects the first returned instance type. A real-world
        // application would probably ask the user to select one here.

        let chosenInstanceType = imageTypes[0]

        //=====================================================================
        // 8. Create an instance with the key pair, security group, AMI, and
        //    instance type (RunInstances).
        //=====================================================================

        print("Creating an instance...")

        guard let imageId = chosenImage.imageId else {
            print("*** Cannot start image without a valid image ID.")
            await cleanUp()
            return
        }
        guard let instanceType = chosenInstanceType.instanceType else {
            print("*** Unable to start image without a valid image type.")
            await cleanUp()
            return
        }

        let instance = await self.runInstance(
            imageId: imageId,
            instanceType: instanceType,
```

```
            keyPairName: keyName!,
            securityGroups: [securityGroupId!]
        )

        guard let instance else {
            await cleanUp()
            return
        }

        instanceId = instance.instanceId
        if instanceId == nil {
            print("*** Instance is missing an ID. Canceling.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 9. Wait for the instance to be ready and then display its
        //    information (DescribeInstances).
        //=====================================================================

        print("Waiting a few seconds to let the instance come up...")

        do {
            try await Task.sleep(for: .seconds(20))
        } catch {
            print("*** Error pausing the task.")
        }
        print("Success! Your new instance is ready:")

        //=====================================================================
        // 10. Display SSH connection info for the instance.
        //=====================================================================

        var runningInstance = await self.describeInstance(instanceId:
 instanceId!)

        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
 {
            print("\nYou can SSH to this instance using the following command:")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }
```

```
        //=====================================================================
        // 11. Stop the instance and wait for it to stop (StopInstances).
        //=====================================================================

        print("Stopping the instance...")

        if !(await self.stopInstance(instanceId: instanceId!, waitUntilStopped:
 true)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 12. Start the instance and wait for it to start (StartInstances).
        //=====================================================================

        print("Starting the instance again...")

        if !(await self.startInstance(instanceId: instanceId!, waitUntilStarted:
 true)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 13. Display SSH connection info for the instance. Note that it's
        //     changed.
        //=====================================================================

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
 {
            print("\nYou can SSH to this instance using the following command.")
            print("This is probably different from when the instance was running
 before.")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // 14. Allocate an elastic IP and associate it with the instance
        //     (AllocateAddress and AssociateAddress).
        //=====================================================================
```

```
        allocationId = await self.allocateAddress()

        if allocationId == nil {
            await cleanUp()
            return
        }

        associationId = await self.associateAddress(instanceId: instanceId!,
 allocationId: allocationId)

        if associationId == nil {
            await cleanUp()
            return
        }

        //=====================================================================
        // 15. Display SSH connection info for the connection. Note that the
        //     public IP is now the Elastic IP, which stays constant.
        //=====================================================================

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
 {
            print("\nYou can SSH to this instance using the following command.")
            print("This has changed again, and is now the Elastic IP.")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // Handle all cleanup tasks
        //=====================================================================

        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items:
    ///
    /// * Elastic IP allocation and association
    /// * Terminate the instance
    /// * Delete the security group
    /// * Delete the key pair
    func cleanUp() async {
        //=====================================================================
```

```
        // 16. Disassociate and delete the Elastic IP (DisassociateAddress and
        //     ReleaseAddress).
        //===========================================================================

        if associationId != nil {
            await self.disassociateAddress(associationId: associationId!)
        }

        if allocationId != nil {
            await self.releaseAddress(allocationId: allocationId!)
        }

        //===========================================================================
        // 17. Terminate the instance and wait for it to terminate
        //     (TerminateInstances).
        //===========================================================================

        if instanceId != nil {
            print("Terminating the instance...")
            _ = await self.terminateInstance(instanceId: instanceId!,
waitUntilTerminated: true)
        }

        //===========================================================================
        // 18. Delete the security group (DeleteSecurityGroup).
        //===========================================================================

        if securityGroupId != nil {
            print("Deleting the security group...")
            _ = await self.deleteSecurityGroup(groupId: securityGroupId!)
        }

        //===========================================================================
        // 19. Delete the key pair (DeleteKeyPair).
        //===========================================================================

        if keyName != nil {
            print("Deleting the key pair...")
            _ = await self.deleteKeyPair(keyPair: keyName!)
        }
    }

    /// Create a new RSA key pair and save the private key to a randomly-named
    /// file in the temporary directory.
```

```
    ///
    /// - Parameter name: The name of the key pair to create.
    ///
    /// - Returns: The URL of the newly created `.pem` file or `nil` if unable
    ///   to create the key pair.
    func createKeyPair(name: String) async -> URL? {
        do {
            let output = try await ec2Client.createKeyPair(
                input: CreateKeyPairInput(
                    keyName: name
                )
            )

            guard let keyMaterial = output.keyMaterial else {
                return nil
            }

            // Build the URL of the temporary private key file.

            let fileURL = URL.temporaryDirectory
                                .appendingPathComponent(name)
                                .appendingPathExtension("pem")

            do {
                try keyMaterial.write(to: fileURL, atomically: true, encoding:
 String.Encoding.utf8)
                return fileURL
            } catch {
                print("*** Failed to write the private key.")
                return nil
            }
        } catch {
            print("*** Unable to create the key pair.")
            return nil
        }
    }

    /// Describe the key pairs associated with the user by outputting each key
    /// pair's name and fingerprint.
    func describeKeyPairs() async {
        do {
            let output = try await ec2Client.describeKeyPairs(
                input: DescribeKeyPairsInput()
            )
```

```
            guard let keyPairs = output.keyPairs else {
                print("*** No key pairs list available.")
                return
            }

            for keyPair in keyPairs {
                print(keyPair.keyName ?? "<unknown>", ":",
keyPair.keyFingerprint ?? "<unknown>")
            }
        } catch {
            print("*** Error: Unable to obtain a key pair list.")
        }
    }

    /// Delete an EC2 key pair.
    ///
    /// - Parameter keyPair: The name of the key pair to delete.
    ///
    /// - Returns: `true` if the key pair is deleted successfully; otherwise
    ///   `false`.
    func deleteKeyPair(keyPair: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteKeyPair(
                input: DeleteKeyPairInput(
                    keyName: keyPair
                )
            )

            return true
        } catch {
            print("*** Error deleting the key pair:
\(error.localizedDescription)")
            return false
        }
    }

    /// Return a list of AMI names that contain the specified string.
    ///
    /// - Parameter filter: A string that must be contained in all returned
    ///   AMI names.
    ///
    /// - Returns: An array of the parameters matching the specified substring.
```

```
    func findAMIsMatchingFilter(_ filter: String) async ->
[SSMClientTypes.Parameter] {
        var parameterList: [SSMClientTypes.Parameter] = []
        var matchingAMIs: [SSMClientTypes.Parameter] = []

        do {
            let pages = ssmClient.getParametersByPathPaginated(
                input: GetParametersByPathInput(
                    path: "/aws/service/ami-amazon-linux-latest"
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return matchingAMIs
                }

                for parameter in parameters {
                    parameterList.append(parameter)
                }
            }

            print("Found \(parameterList.count) images total:")
            for parameter in parameterList {
                guard let name = parameter.name else {
                    continue
                }
                print("    \(name)")

                if name.contains(filter) {
                    matchingAMIs.append(parameter)
                }
            }
        } catch {
            return matchingAMIs
        }

        return matchingAMIs
    }

    /// Return a list of instance types matching the specified architecture
    /// and instance sizes.
    ///
    /// - Parameters:
```

```
    ///   - architecture: The architecture of the instance types to return, as
    ///     a member of `EC2ClientTypes.ArchitectureValues`.
    ///   - sizes: An array of one or more strings identifying sizes of
    ///     instance type to accept.
    ///
    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues
 = EC2ClientTypes.ArchitectureValues.x8664,
                          sizes: [String] = ["*.micro", "*.small"]) async
                          -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )

        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )

            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }
```

```
    /// Get the latest information about the specified instance and output it
    /// to the screen, returning the instance details to the caller.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to provide details about.
    ///   - stateFilter: The state to require the instance to be in.
    ///
    /// - Returns: The instance's details as an `EC2ClientTypes.Instance` object.
    func describeInstance(instanceId: String,
                          stateFilter: EC2ClientTypes.InstanceStateName? =
EC2ClientTypes.InstanceStateName.running) async
                          -> EC2ClientTypes.Instance? {
        do {
            let pages = ec2Client.describeInstancesPaginated(
                input: DescribeInstancesInput(
                    instanceIds: [instanceId]
                )
            )

            for try await page in pages {
                guard let reservations = page.reservations else {
                    continue
                }

                for reservation in reservations {
                    guard let instances = reservation.instances else {
                        continue
                    }

                    for instance in instances {
                        guard let state = instance.state else {
                            print("*** Instance is missing its state...")
                            continue
                        }
                        let instanceState = state.name

                        if stateFilter != nil && (instanceState != stateFilter) {
                            continue
                        }

                        let instanceTypeName: String
                        if instance.instanceType == nil {
                            instanceTypeName = "<N/A>"
                        } else {
```

```
                            instanceTypeName = instance.instanceType?.rawValue ??
 "<N/A>"
                        }

                        let instanceStateName: String
                        if instanceState == nil {
                            instanceStateName = "<N/A>"
                        } else {
                            instanceStateName = instanceState?.rawValue ?? "<N/
A>"
                        }

                        print("""
                        Instance: \(instance.instanceId ?? "<N/A>")
                                • Image ID: \(instance.imageId ?? "<N/A>")
                                • Instance type: \(instanceTypeName)
                                • Key name: \(instance.keyName ?? "<N/A>")
                                • VPC ID: \(instance.vpcId ?? "<N/A>")
                                • Public IP: \(instance.publicIpAddress ?? "N/A")
                                • State: \(instanceStateName)
                        """)

                        return instance
                    }
                }
            }
        } catch {
            print("*** Error retrieving instance information to display:
 \(error.localizedDescription)")
            return nil
        }

        return nil
    }

    /// Stop the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to stop.
    ///   - waitUntilStopped: If `true`, execution waits until the instance
    ///     has stopped. Otherwise, execution continues and the instance stops
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully stopped (or is left to
```

```
    ///   stop asynchronously). `false` if the instance doesn't stop.
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async -
> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance:
 \(error.localizedDescription)")
            return false
        }
    }

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance
```

```
    ///     has started. Otherwise, execution continues and the instance starts
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully started (or is left to
    ///   start asynchronously). `false` if the instance doesn't start.
    func startInstance(instanceId: String, waitUntilStarted: Bool = false) async
-> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.startInstances(
                input: StartInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStarted {
                print("Waiting for the instance to start...")

                let waitOptions = WaiterOptions(maxWaitTime: 60.0)
                let output = try await ec2Client.waitUntilInstanceRunning(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to start the instance:
\(error.localizedDescription)")
            return false
        }
    }

    /// Terminate the specified instance.
    ///
```

```
    /// - Parameters:
    ///   - instanceId: The instance to terminate.
    ///   - waitUntilTerminated: Whether or not to wait until the instance is
    ///     terminated before returning.
    ///
    /// - Returns: `true` if terminated successfully. `false` if not or if an
    ///   error occurs.
    func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
 async -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.terminateInstances(
                input: TerminateInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilTerminated {
                print("Waiting for the instance to terminate...")

                let waitOptions = WaiterOptions(maxWaitTime: 600.0)
                let output = try await ec2Client.waitUntilInstanceTerminated(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to terminate the instance:
 \(error.localizedDescription)")
            return false
        }
    }
```

```
    /// Return an array of `EC2ClientTypes.Image` objects describing all of
    /// the images in the specified array.
    ///
    /// - Parameter idList: A list of image ID strings indicating the images
    ///   to return details about.
    ///
    /// - Returns: An array of the images.
    func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
        do {
            let output = try await ec2Client.describeImages(
                input: DescribeImagesInput(
                    imageIds: idList
                )
            )

            guard let images = output.images else {
                print("*** No images found.")
                return []
            }

            for image in images {
                guard let id = image.imageId else {
                    continue
                }
                print("   \(id): \(image.description ?? "<no description>")")
            }

            return images
        } catch {
            print("*** Error getting image descriptions:
\(error.localizedDescription)")
            return []
        }
    }

    /// Create and return a new EC2 instance.
    ///
    /// - Parameters:
    ///   - imageId: The image ID of the AMI to use when creating the instance.
    ///   - instanceType: The type of instance to create.
    ///   - keyPairName: The RSA key pair's name to use to secure the instance.
    ///   - securityGroups: The security group or groups to add the instance
    ///     to.
```

```
    ///
    /// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
    func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
                     keyPairName: String, securityGroups: [String]?) async ->
EC2ClientTypes.Instance? {
        do {
            let output = try await ec2Client.runInstances(
                input: RunInstancesInput(
                    imageId: imageId,
                    instanceType: instanceType,
                    keyName: keyPairName,
                    maxCount: 1,
                    minCount: 1,
                    securityGroupIds: securityGroups
                )
            )

            guard let instances = output.instances else {
                print("*** Unable to create the instance.")
                return nil
            }

            return instances[0]
        } catch {
            print("*** Error creating the instance:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Return the device's external IP address.
    ///
    /// - Returns: A string containing the device's IP address.
    func getMyIPAddress() -> String? {
        guard let url = URL(string: "http://checkip.amazonaws.com") else {
            print("Couldn't create the URL")
            return nil
        }

        do {
            print("Getting the IP address...")
            return try String(contentsOf: url, encoding:
String.Encoding.utf8).trim()
        } catch {
```

```
            print("*** Unable to get your public IP address.")
            return nil
        }
    }

    /// Create a new security group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the group to create.
    ///   - groupDescription: A description of the new security group.
    ///
    /// - Returns: The ID string of the new security group.
    func createSecurityGroup(name groupName: String, description
groupDescription: String) async -> String? {
        do {
            let output = try await ec2Client.createSecurityGroup(
                input: CreateSecurityGroupInput(
                    description: groupDescription,
                    groupName: groupName
                )
            )

            return output.groupId
        } catch {
            print("*** Error creating the security group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Authorize ingress of connections for the security group.
    ///
    /// - Parameters:
    ///   - groupId: The group ID of the security group to authorize access for.
    ///   - ipAddress: The IP address of the device to grant access to.
    ///
    /// - Returns: `true` if access is successfully granted; otherwise `false`.
    func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async
-> Bool {
        let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
        let httpPermission = EC2ClientTypes.IpPermission(
            fromPort: 80,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
```

```
            toPort: 80
        )

        let sshPermission = EC2ClientTypes.IpPermission(
            fromPort: 22,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 22
        )

        do {
            _ = try await ec2Client.authorizeSecurityGroupIngress(
                input: AuthorizeSecurityGroupIngressInput(
                    groupId: groupId,
                    ipPermissions: [httpPermission, sshPermission]
                )
            )

            return true
        } catch {
            print("*** Error authorizing ingress for the security group:
\(error.localizedDescription)")
            return false
        }
    }

    func describeSecurityGroups(groupId: String) async -> Bool {
        do {
            let output = try await ec2Client.describeSecurityGroups(
                input: DescribeSecurityGroupsInput(
                    groupIds: [groupId]
                )
            )

            guard let securityGroups = output.securityGroups else {
                print("No security groups found.")
                return true
            }

            for group in securityGroups {
                print("Group \(group.groupId ?? "<unknown>") found with VPC
\(group.vpcId ?? "<unknown>")")
            }
            return true
```

```
        } catch {
            print("*** Error getting security group details:
\(error.localizedDescription)")
            return false
        }
    }

    /// Delete a security group.
    ///
    /// - Parameter groupId: The ID of the security group to delete.
    ///
    /// - Returns: `true` on successful deletion; `false` on error.
    func deleteSecurityGroup(groupId: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteSecurityGroup(
                input: DeleteSecurityGroupInput(
                    groupId: groupId
                )
            )

            return true
        } catch {
            print("*** Error deleting the security group:
\(error.localizedDescription)")
            return false
        }
    }

    /// Allocate an Elastic IP address.
    ///
    /// - Returns: A string containing the ID of the Elastic IP.
    func allocateAddress() async -> String? {
        do {
            let output = try await ec2Client.allocateAddress(
                input: AllocateAddressInput(
                    domain: EC2ClientTypes.DomainType.vpc
                )
            )

            guard let allocationId = output.allocationId else {
                return nil
            }

            return allocationId
```

```
        } catch {
            print("*** Unable to allocate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Associate the specified allocated Elastic IP to a given instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )

            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
            )
        } catch {
```

```
            print("*** Unable to disassociate the IP address:
 \(error.localizedDescription)")
        }
    }

    /// Release an allocated Elastic IP.
    ///
    /// - Parameter allocationId: The allocation ID of the Elastic IP to
    ///   release.
    func releaseAddress(allocationId: String?) async {
        do {
            _ = try await ec2Client.releaseAddress(
                input: ReleaseAddressInput(
                    allocationId: allocationId
                )
            )
        } catch {
            print("*** Unable to release the IP address:
 \(error.localizedDescription)")
        }
    }

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())
```

```
        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Ações para o Amazon EC2 usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon EC2 com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon EC2 e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon EC2 usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Elastic Compute Cloud.

Exemplos

- Usar AcceptVpcPeeringConnection com uma CLI
- Use AllocateAddresscom um AWS SDK ou CLI
- Usar AllocateHosts com uma CLI
- Usar AssignPrivateIpAddresses com uma CLI
- Use AssociateAddresscom um AWS SDK ou CLI
- Usar AssociateDhcpOptions com uma CLI
- Usar AssociateRouteTable com uma CLI
- Usar AttachInternetGateway com uma CLI
- Usar AttachNetworkInterface com uma CLI
- Usar AttachVolume com uma CLI
- Usar AttachVpnGateway com uma CLI
- Usar AuthorizeSecurityGroupEgress com uma CLI
- Use AuthorizeSecurityGroupIngresscom um AWS SDK ou CLI
- Usar CancelCapacityReservation com uma CLI
- Usar CancelImportTask com uma CLI
- Usar CancelSpotFleetRequests com uma CLI
- Usar CancelSpotInstanceRequests com uma CLI
- Usar ConfirmProductInstance com uma CLI
- Usar CopyImage com uma CLI
- Usar CopySnapshot com uma CLI

- Usar CreateCapacityReservation com uma CLI
- Usar CreateCustomerGateway com uma CLI
- Usar CreateDhcpOptions com uma CLI
- Usar CreateFlowLogs com uma CLI
- Usar CreateImage com uma CLI
- Usar CreateInstanceExportTask com uma CLI
- Usar CreateInternetGateway com uma CLI
- Use CreateKeyPaircom um AWS SDK ou CLI
- Use CreateLaunchTemplatecom um AWS SDK ou CLI
- Usar CreateNetworkAcl com uma CLI
- Usar CreateNetworkAclEntry com uma CLI
- Usar CreateNetworkInterface com uma CLI
- Usar CreatePlacementGroup com uma CLI
- Usar CreateRoute com uma CLI
- Use CreateRouteTablecom um AWS SDK ou CLI
- Use CreateSecurityGroupcom um AWS SDK ou CLI
- Usar CreateSnapshot com uma CLI
- Usar CreateSpotDatafeedSubscription com uma CLI
- Use CreateSubnetcom um AWS SDK ou CLI
- Use CreateTagscom um AWS SDK ou CLI
- Usar CreateVolume com uma CLI
- Use CreateVpccom um AWS SDK ou CLI
- Use CreateVpcEndpointcom um AWS SDK ou CLI
- Usar CreateVpnConnection com uma CLI
- Usar CreateVpnConnectionRoute com uma CLI
- Usar CreateVpnGateway com uma CLI
- Usar DeleteCustomerGateway com uma CLI
- Usar DeleteDhcpOptions com uma CLI
- Usar DeleteFlowLogs com uma CLI
- Usar DeleteInternetGateway com uma CLI

- Use DeleteKeyPaircom um AWS SDK ou CLI
- Use DeleteLaunchTemplatecom um AWS SDK ou CLI
- Usar DeleteNetworkAcl com uma CLI
- Usar DeleteNetworkAclEntry com uma CLI
- Usar DeleteNetworkInterface com uma CLI
- Usar DeletePlacementGroup com uma CLI
- Usar DeleteRoute com uma CLI
- Usar DeleteRouteTable com uma CLI
- Use DeleteSecurityGroupcom um AWS SDK ou CLI
- Use DeleteSnapshotcom um AWS SDK ou CLI
- Usar DeleteSpotDatafeedSubscription com uma CLI
- Usar DeleteSubnet com uma CLI
- Usar DeleteTags com uma CLI
- Usar DeleteVolume com uma CLI
- Use DeleteVpccom um AWS SDK ou CLI
- Use DeleteVpcEndpointscom um AWS SDK ou CLI
- Usar DeleteVpnConnection com uma CLI
- Usar DeleteVpnConnectionRoute com uma CLI
- Usar DeleteVpnGateway com uma CLI
- Usar DeregisterImage com uma CLI
- Usar DescribeAccountAttributes com uma CLI
- Use DescribeAddressescom um AWS SDK ou CLI
- Use DescribeAvailabilityZonescom um AWS SDK ou CLI
- Usar DescribeBundleTasks com uma CLI
- Usar DescribeCapacityReservations com uma CLI
- Usar DescribeCustomerGateways com uma CLI
- Usar DescribeDhcpOptions com uma CLI
- Usar DescribeFlowLogs com uma CLI
- Usar DescribeHostReservationOfferings com uma CLI
- Usar DescribeHosts com uma CLI

- Use DescribeIamInstanceProfileAssociationscom um AWS SDK ou CLI
- Usar DescribeIdFormat com uma CLI
- Usar DescribeIdentityIdFormat com uma CLI
- Usar DescribeImageAttribute com uma CLI
- Use DescribeImagescom um AWS SDK ou CLI
- Usar DescribeImportImageTasks com uma CLI
- Usar DescribeImportSnapshotTasks com uma CLI
- Usar DescribeInstanceAttribute com uma CLI
- Use DescribeInstanceStatuscom um AWS SDK ou CLI
- Use DescribeInstanceTypescom um AWS SDK ou CLI
- Use DescribeInstancescom um AWS SDK ou CLI
- Usar DescribeInternetGateways com uma CLI
- Use DescribeKeyPairscom um AWS SDK ou CLI
- Usar DescribeNetworkAcls com uma CLI
- Usar DescribeNetworkInterfaceAttribute com uma CLI
- Usar DescribeNetworkInterfaces com uma CLI
- Usar DescribePlacementGroups com uma CLI
- Usar DescribePrefixLists com uma CLI
- Use DescribeRegionscom um AWS SDK ou CLI
- Use DescribeRouteTablescom um AWS SDK ou CLI
- Usar DescribeScheduledInstanceAvailability com uma CLI
- Usar DescribeScheduledInstances com uma CLI
- Use DescribeSecurityGroupscom um AWS SDK ou CLI
- Usar DescribeSnapshotAttribute com uma CLI
- Use DescribeSnapshotscom um AWS SDK ou CLI
- Usar DescribeSpotDatafeedSubscription com uma CLI
- Usar DescribeSpotFleetInstances com uma CLI
- Usar DescribeSpotFleetRequestHistory com uma CLI
- Usar DescribeSpotFleetRequests com uma CLI
- Usar DescribeSpotInstanceRequests com uma CLI

- Usar DescribeSpotPriceHistory com uma CLI
- Use DescribeSubnetscom um AWS SDK ou CLI
- Usar DescribeTags com uma CLI
- Usar DescribeVolumeAttribute com uma CLI
- Usar DescribeVolumeStatus com uma CLI
- Usar DescribeVolumes com uma CLI
- Usar DescribeVpcAttribute com uma CLI
- Usar DescribeVpcClassicLink com uma CLI
- Usar DescribeVpcClassicLinkDnsSupport com uma CLI
- Usar DescribeVpcEndpointServices com uma CLI
- Usar DescribeVpcEndpoints com uma CLI
- Use DescribeVpcscom um AWS SDK ou CLI
- Usar DescribeVpnConnections com uma CLI
- Usar DescribeVpnGateways com uma CLI
- Usar DetachInternetGateway com uma CLI
- Usar DetachNetworkInterface com uma CLI
- Usar DetachVolume com uma CLI
- Usar DetachVpnGateway com uma CLI
- Usar DisableVgwRoutePropagation com uma CLI
- Usar DisableVpcClassicLink com uma CLI
- Usar DisableVpcClassicLinkDnsSupport com uma CLI
- Use DisassociateAddresscom um AWS SDK ou CLI
- Usar DisassociateRouteTable com uma CLI
- Usar EnableVgwRoutePropagation com uma CLI
- Usar EnableVolumeIo com uma CLI
- Usar EnableVpcClassicLink com uma CLI
- Usar EnableVpcClassicLinkDnsSupport com uma CLI
- Usar GetConsoleOutput com uma CLI
- Usar GetHostReservationPurchasePreview com uma CLI
- Use GetPasswordDatacom um AWS SDK ou CLI

- Usar ImportImage com uma CLI
- Usar ImportKeyPair com uma CLI
- Usar ImportSnapshot com uma CLI
- Usar ModifyCapacityReservation com uma CLI
- Usar ModifyHosts com uma CLI
- Usar ModifyIdFormat com uma CLI
- Usar ModifyImageAttribute com uma CLI
- Usar ModifyInstanceAttribute com uma CLI
- Usar ModifyInstanceCreditSpecification com uma CLI
- Usar ModifyNetworkInterfaceAttribute com uma CLI
- Usar ModifyReservedInstances com uma CLI
- Usar ModifySnapshotAttribute com uma CLI
- Usar ModifySpotFleetRequest com uma CLI
- Usar ModifySubnetAttribute com uma CLI
- Usar ModifyVolumeAttribute com uma CLI
- Usar ModifyVpcAttribute com uma CLI
- Use MonitorInstancescom um AWS SDK ou CLI
- Usar MoveAddressToVpc com uma CLI
- Usar PurchaseHostReservation com uma CLI
- Usar PurchaseScheduledInstances com uma CLI
- Use RebootInstancescom um AWS SDK ou CLI
- Usar RegisterImage com uma CLI
- Usar RejectVpcPeeringConnection com uma CLI
- Use ReleaseAddresscom um AWS SDK ou CLI
- Usar ReleaseHosts com uma CLI
- Use ReplaceIamInstanceProfileAssociationcom um AWS SDK ou CLI
- Usar ReplaceNetworkAclAssociation com uma CLI
- Usar ReplaceNetworkAclEntry com uma CLI
- Usar ReplaceRoute com uma CLI
- Usar ReplaceRouteTableAssociation com uma CLI

- Usar ReportInstanceStatus com uma CLI
- Usar RequestSpotFleet com uma CLI
- Usar RequestSpotInstances com uma CLI
- Usar ResetImageAttribute com uma CLI
- Usar ResetInstanceAttribute com uma CLI
- Usar ResetNetworkInterfaceAttribute com uma CLI
- Usar ResetSnapshotAttribute com uma CLI
- Usar RevokeSecurityGroupEgress com uma CLI
- Usar RevokeSecurityGroupIngress com uma CLI
- Use RunInstancescom um AWS SDK ou CLI
- Usar RunScheduledInstances com uma CLI
- Use StartInstancescom um AWS SDK ou CLI
- Use StopInstancescom um AWS SDK ou CLI
- Use TerminateInstancescom um AWS SDK ou CLI
- Usar UnassignPrivateIpAddresses com uma CLI
- Use UnmonitorInstancescom um AWS SDK ou CLI
- Usar UpdateSecurityGroupRuleDescriptionsIngress com uma CLI

Usar **`AcceptVpcPeeringConnection`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AcceptVpcPeeringConnection`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Aceitar uma conexão de emparelhamento da VPC

Este exemplo aceita a solicitação de conexão de emparelhamento da VPC especificada.

Comando:

```
aws ec2 accept-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

Saída:

```
{
  "VpcPeeringConnection": {
    "Status": {
      "Message": "Provisioning",
      "Code": "provisioning"
    },
    "Tags": [],
    "AccepterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-44455566",
      "CidrBlock": "10.0.1.0/28"
    },
    "VpcPeeringConnectionId": "pcx-1a2b3c4d",
    "RequesterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-111abc45",
      "CidrBlock": "10.0.0.0/28"
    }
  }
}
```

- Para obter detalhes da API, consulte AcceptVpcPeeringConnectionna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo aprova o pcx-1dfad234b56ff78be solicitado VpcPeeringConnectionId

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

Saída:

```
AccepterVpcInfo        : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime         : 1/1/0001 12:00:00 AM
RequesterVpcInfo       : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status                 : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                   : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- Para obter detalhes da API, consulte AcceptVpcPeeringConnectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo aprova o pcx-1dfad234b56ff78be solicitado VpcPeeringConnectionId

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

Saída:

```
AccepterVpcInfo        : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime         : 1/1/0001 12:00:00 AM
RequesterVpcInfo       : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status                 : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                   : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- Para obter detalhes da API, consulte AcceptVpcPeeringConnectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`AllocateAddress`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AllocateAddress`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma rede privada virtual básica
- Introdução às máquinas virtuais

- Rede privada virtual com servidores privados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
    // instance. By using an Elastic IP address, you can keep the public IP
 address
    // constant even when you restart the associated instance.
    /// </summary>
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation
 ID {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-
ip-addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
 exceeded. {ec2Exception.Message}");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
 {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte AllocateAddressa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_allocate_address
#
# This function allocates an Elastic IP address for use with Amazon Elastic
 Compute Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#       -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard').
#
# Returns:
#       The allocated Elastic IP address, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
```

```
function ec2_allocate_address() {
  local domain response

  # Function to display usage information
  function usage() {
    echo "function ec2_allocate_address"
    echo "Allocates an Elastic IP address for use with Amazon Elastic Compute
 Cloud (Amazon EC2) instances in a specific AWS Region."
    echo "  -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard')."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "d:h" option; do
    case "${option}" in
      d) domain="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$domain" ]]; then
    errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc'
 or 'standard')."
    return 1
  fi

  if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
    errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
    return 1
  fi

  # Allocate the Elastic IP address
  response=$(aws ec2 allocate-address \
```

```
    --domain "$domain" \
    --query "[PublicIp,AllocationId]" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
```

```
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte AllocateAddressna Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Allocate an Elastic IP address and associate it with an Amazon Elastic
 Compute Cloud
//! (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param[out] publicIPAddress: String to return the public IP address.
  \param[out] allocationID: String to return the allocation ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::EC2::allocateAndAssociateAddress(const Aws::String &instanceId,
 Aws::String &publicIPAddress,
                                              Aws::String &allocationID,
                                              const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AllocateAddressRequest request;
    request.SetDomain(Aws::EC2::Model::DomainType::vpc);

    const Aws::EC2::Model::AllocateAddressOutcome outcome =
            ec2Client.AllocateAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to allocate Elastic IP address:" <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    const Aws::EC2::Model::AllocateAddressResponse &response =
 outcome.GetResult();
    allocationID = response.GetAllocationId();
    publicIPAddress = response.GetPublicIp();


    return true;
}
```

- Para obter detalhes da API, consulte AllocateAddressa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: alocar um endereço IP elástico do conjunto de endereços da Amazon

O exemplo `allocate-address` a seguir aloca um endereço IP elástico. O Amazon EC2 seleciona o endereço do conjunto de endereços da Amazon.

```
aws ec2 allocate-address
```

Saída:

```
{
    "PublicIp": "70.224.234.241",
    "AllocationId": "eipalloc-01435ba59eEXAMPLE",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-west-2",
    "Domain": "vpc"
}
```

Para obter mais informações, consulte Endereços IP elásticos no Guia do usuário do Amazon EC2.

Exemplo 2: alocar um endereço IP elástico e associá-lo a um grupo de borda de rede

O exemplo `allocate-address` a seguir aloca um endereço IP elástico e o associa ao grupo de borda de rede especificado.

```
aws ec2 allocate-address \
    --network-border-group us-west-2-lax-1
```

Saída:

```
{
    "PublicIp": "70.224.234.241",
    "AllocationId": "eipalloc-e03dd489ceEXAMPLE",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-west-2-lax-1",
    "Domain": "vpc"
}
```

Para obter mais informações, consulte Endereços IP elásticos no Guia do usuário do Amazon EC2.

Exemplo 3: alocar um endereço IP elástico de um conjunto de endereços de sua propriedade

O exemplo `allocate-address` a seguir aloca um endereço IP elástico de um conjunto de endereços que você trouxe para a conta da Amazon Web Services. O Amazon EC2 seleciona o endereço do conjunto de endereços.

```
aws ec2 allocate-address \
```

```
    --public-ipv4-pool ipv4pool-ec2-1234567890abcdef0
```

Saída:

```
{
    "AllocationId": "eipalloc-02463d08ceEXAMPLE",
    "NetworkBorderGroup": "us-west-2",
    "CustomerOwnedIp": "18.218.95.81",
    "CustomerOwnedIpv4Pool": "ipv4pool-ec2-1234567890abcdef0",
    "Domain": "vpc"
    "NetworkBorderGroup": "us-west-2",
}
```

Para obter mais informações, consulte Endereços IP elásticos no Guia do usuário do Amazon EC2.

Exemplo 4: alocar um endereço IP elástico de um grupo de IPAM

O exemplo de `allocate-address` a seguir aloca um endereço IP elástico /32 específico de um grupo do Gerenciador de endereços IP (IPAM) da Amazon VPC.

```
aws ec2 allocate-address \
    --region us-east-1 \
    --ipam-pool-id ipam-pool-1234567890abcdef0 \
    --address 192.0.2.0
```

Saída:

```
{
    "PublicIp": "192.0.2.0",
    "AllocationId": "eipalloc-abcdef01234567890",
    "PublicIpv4Pool": "ipam-pool-1234567890abcdef0",
    "NetworkBorderGroup": "us-east-1",
    "Domain": "vpc"
}
```

Para obter mais informações, consulte Allocate sequential Elastic IP addresses from an IPAM pool no Guia do usuário do IPAM da Amazon VPC.

- Para obter detalhes da API, consulte AllocateAddressna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }
```

- Para obter detalhes da API, consulte AllocateAddressa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { AllocateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Allocates an Elastic IP address to your AWS account.
 */
export const main = async () => {
  const client = new EC2Client({});
  const command = new AllocateAddressCommand({});

  try {
    const { AllocationId, PublicIp } = await client.send(command);
    console.log("A new IP address has been allocated to your account:");
    console.log(`ID: ${AllocationId} Public IP: ${PublicIp}`);
    console.log(
      "You can view your IP addresses in the AWS Management Console for Amazon
 EC2. Look under Network & Security > Elastic IPs",
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
import { fileURLToPath } from "node:url";
// Call function if run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte AllocateAddressa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

- Para obter detalhes da API, consulte a AllocateAddressreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo aloca um endereço IP elástico para usar com uma instância em uma VPC.

```
New-EC2Address -Domain Vpc
```

Saída:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

Exemplo 2: Este exemplo aloca um endereço IP elástico para usar com uma instância em. EC2-Classic

```
New-EC2Address
```

Saída:

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- Para obter detalhes da API, consulte AllocateAddressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo aloca um endereço IP elástico para usar com uma instância em uma VPC.

```
New-EC2Address -Domain Vpc
```

Saída:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

Exemplo 2: Este exemplo aloca um endereço IP elástico para usar com uma instância em. EC2-Classic

```
New-EC2Address
```

Saída:

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- Para obter detalhes da API, consulte AllocateAddressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
```

```
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP
 address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the
 maximum limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
```

```
                allocation_id=response["AllocationId"],
 public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
 Support for an increase."
                )
            raise err
        return elastic_ip
```

- Para obter detalhes da API, consulte a AllocateAddressReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end
```

- Para obter detalhes da API, consulte AllocateAddressa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
       self.client
           .allocate_address()
           .domain(DomainType::Vpc)
           .send()
           .await
           .map_err(EC2Error::from)
   }
```

- Para obter detalhes da API, consulte a AllocateAddressreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).   " oo_result
 is returned for testing purposes. "
        MESSAGE 'Allocated an Elastic IP address.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AllocateAddressreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Allocate an Elastic IP address.
    ///
    /// - Returns: A string containing the ID of the Elastic IP.
    func allocateAddress() async -> String? {
        do {
            let output = try await ec2Client.allocateAddress(
                input: AllocateAddressInput(
                    domain: EC2ClientTypes.DomainType.vpc
                )
            )

            guard let allocationId = output.allocationId else {
                return nil
            }
```

```
            return allocationId
        } catch {
            print("*** Unable to allocate the IP address:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte AllocateAddressa referência da API AWS SDK for Swift.

Usar **`AllocateHosts`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AllocateHosts`.

CLI

AWS CLI

Exemplo 1: alocar um Host Dedicado

O exemplo `allocate-hosts` a seguir aloca um único Host Dedicado na Zona de Disponibilidade `eu-west-1a`, no qual você pode executar instâncias `m5.large`. Por padrão, o Host Dedicado aceita somente a execução da instância de destino e não oferece suporte à recuperação do host.

```
aws ec2 allocate-hosts \
    --instance-type m5.large \
    --availability-zone eu-west-1a \
    --quantity 1
```

Saída:

```
{
    "HostIds": [
        "h-07879acf49EXAMPLE"
    ]
}
```

Exemplo 2: alocar um Host Dedicado com posicionamento automático e recuperação de host ativados

O exemplo `allocate-hosts` a seguir aloca um único Host Dedicado na Zona de Disponibilidade `eu-west-1a` com posicionamento automático e recuperação de host ativados.

```
aws ec2 allocate-hosts \
    --instance-type m5.large \
    --availability-zone eu-west-1a \
    --auto-placement on \
    --host-recovery on \
    --quantity 1
```

Saída:

```
{
     "HostIds": [
         "h-07879acf49EXAMPLE"
     ]
}
```

Exemplo 3: alocar um Host Dedicado com tags

O exemplo `allocate-hosts` a seguir aloca um único Host Dedicado e aplica uma tag com uma chave chamada `purpose` e um valor de `production`.

```
aws ec2 allocate-hosts \
    --instance-type m5.large \
    --availability-zone eu-west-1a \
    --quantity 1 \
    --tag-specifications 'ResourceType=dedicated-
host,Tags={Key=purpose,Value=production}'
```

Saída:

```
{
    "HostIds": [
        "h-07879acf49EXAMPLE"
    ]
}
```

Para obter mais informações, consulte Alocar um host dedicado no Manual do Usuário do Amazon EC2.

- Para obter detalhes da API, consulte AllocateHostsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo aloca um host dedicado à sua conta para o tipo de instância e zona de disponibilidade específicos

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType
 m4.xlarge -Quantity 1
```

Saída:

```
h-01e23f4cd567890f3
```

- Para obter detalhes da API, consulte AllocateHostsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo aloca um host dedicado à sua conta para o tipo de instância e zona de disponibilidade específicos

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType
 m4.xlarge -Quantity 1
```

Saída:

```
h-01e23f4cd567890f3
```

- Para obter detalhes da API, consulte AllocateHostsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AssignPrivateIpAddresses`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AssignPrivateIpAddresses`.

CLI

AWS CLI

Atribuir um endereço IP privado secundário a uma interface de rede

Este exemplo atribui os endereços IP privados secundários especificados à interface de rede especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
private-ip-addresses 10.0.0.82
```

Atribuir endereços IP privados secundários que o Amazon EC2 seleciona a uma interface de rede

Este exemplo atribui dois endereços IP privados secundários à interface de rede especificada. O Amazon EC2 atribui automaticamente esses endereços IP a partir dos endereços IP disponíveis no intervalo de blocos CIDR da sub-rede à qual a interface de rede está associada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
secondary-private-ip-address-count 2
```

- Para obter detalhes da API, consulte AssignPrivateIpAddressesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atribui os endereços IP privados secundários especificados à interface de rede especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

Exemplo 2: esse exemplo cria dois endereços IP privados secundários e os atribui à interface de rede especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- Para obter detalhes da API, consulte AssignPrivateIpAddressesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atribui os endereços IP privados secundários especificados à interface de rede especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

Exemplo 2: esse exemplo cria dois endereços IP privados secundários e os atribui à interface de rede especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- Para obter detalhes da API, consulte AssignPrivateIpAddressesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`AssociateAddress`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AssociateAddress`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução às máquinas virtuais

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Associates an Elastic IP address with an instance. When this association
 is
    /// created, the Elastic IP's public IP address is immediately used as the
 public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
 instanceId)
    {
        try
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
```

```
                    $"InstanceId is invalid, unable to associate address.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte AssociateAddressa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to
 associate.
#       -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with.
#
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_associate_address() {
  local allocation_id instance_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 associate."
    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic
 IP address with."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:i:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      i) instance_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi
```

```
  if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
    return 1
  fi

  # Associate the Elastic IP address
  response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
```

```
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte AssociateAddressna Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

//! Associate an Elastic IP address with an EC2 instance.
```

```
/*!
  \param instanceId: An EC2 instance ID.
  \param allocationId: An Elastic IP allocation ID.
  \param[out] associationID: String to receive the association ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: True if the address was associated with the instance; otherwise,
 false.
 */
bool AwsDoc::EC2::associateAddress(const Aws::String &instanceId, const
 Aws::String &allocationId,
                                    Aws::String &associationID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AssociateAddressRequest request;
    request.SetInstanceId(instanceId);
    request.SetAllocationId(allocationId);

    Aws::EC2::Model::AssociateAddressOutcome outcome =
 ec2Client.AssociateAddress(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to associate address " << allocationId <<
                  " with instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully associated address " << allocationId <<
                  " with instance " << instanceId << std::endl;
        associationID = outcome.GetResult().GetAssociationId();
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AssociateAddressa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: associar um endereço IP elástico a uma instância

O exemplo de `associate-address` a seguir associa um endereço IP elástico à instância do EC2 especificada.

```
aws ec2 associate-address \
    --instance-id i-0b263919b6498b123 \
    --allocation-id eipalloc-64d5890a
```

Saída:

```
{
    "AssociationId": "eipassoc-2bebb745"
}
```

Exemplo 2: associar um endereço IP elástico a uma interface de rede

O exemplo de `associate-address` a seguir associa o endereço IP elástico especificado à interface de rede determinada.

```
aws ec2 associate-address
    --allocation-id eipalloc-64d5890a \
    --network-interface-id eni-1a2b3c4d
```

Saída:

```
{
    "AssociationId": "eipassoc-2bebb745"
}
```

Exemplo 3: associar um endereço IP elástico a um endereço IP privado

O exemplo de `associate-address` a seguir associa o endereço IP elástico especificado ao endereço IP privado determinado na interface de rede indicada.

```
aws ec2 associate-address \
    --allocation-id eipalloc-64d5890a \
    --network-interface-id eni-1a2b3c4d \
```

```
    --private-ip-address 10.0.0.85
```

Saída:

```
{
    "AssociationId": "eipassoc-2bebb745"
}
```

Para obter mais informações, consulte Endereços IP elásticos no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte AssociateAddressna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic
IP address with
    * @param allocationId  the allocation ID of the Elastic IP address to
associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId,
String allocationId) {
       AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
```

```
            .build();

        CompletableFuture<AssociateAddressResponse> responseFuture =
 getAsyncClient().associateAddress(associateRequest);
        return responseFuture.thenApply(response -> {
            if (response.associationId() != null) {
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after
 associating address.");
            }
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }
```

- Para obter detalhes da API, consulte AssociateAddressa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { AssociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Associates an Elastic IP address, or carrier IP address (for instances that
 are in subnets in Wavelength Zones)
 * with an instance or a network interface.
 * @param {{ instanceId: string, allocationId: string }} options
 */
export const main = async ({ instanceId, allocationId }) => {
```

```
  const client = new EC2Client({});
  const command = new AssociateAddressCommand({
    // You need to allocate an Elastic IP address before associating it with an
 instance.
    // You can do that with the AllocateAddressCommand.
    AllocationId: allocationId,
    // You need to create an EC2 instance before an IP address can be associated
 with it.
    // You can do that with the RunInstancesCommand.
    InstanceId: instanceId,
  });

  try {
    const { AssociationId } = await client.send(command);
    console.log(
      `Address with allocation ID ${allocationId} is now associated with instance
 ${instanceId}.`,
      `The association ID is ${AssociationId}.`,
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Did you provide the ID of a valid Elastic IP address
 AllocationId?`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte AssociateAddressa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- Para obter detalhes da API, consulte a AssociateAddressreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo associa o endereço IP elástico especificado à instância especificada em uma VPC.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Saída:

```
eipassoc-12345678
```

Exemplo 2: Este exemplo associa o endereço IP elástico especificado à instância especificada em EC2-Classic.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- Para obter detalhes da API, consulte AssociateAddressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa o endereço IP elástico especificado à instância especificada em uma VPC.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Saída:

```
eipassoc-12345678
```

Exemplo 2: Este exemplo associa o endereço IP elástico especificado à instância especificada em EC2-Classic.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- Para obter detalhes da API, consulte AssociateAddressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association
is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
        :return: A response that contains the ID of the association, or None if
no Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance
ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id  # Track the instance associated with this Elastic
IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
```

```
                )
            raise
        return response
```

- Para obter detalhes da API, consulte a AssociateAddressReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
```

```
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  'Error'
end
```

- Para obter detalhes da API, consulte AssociateAddressa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn associate_ip_address(
        &self,
        allocation_id: &str,
        instance_id: &str,
    ) -> Result<AssociateAddressOutput, EC2Error> {
        let response = self
            .client
            .associate_address()
            .allocation_id(allocation_id)
            .instance_id(instance_id)
            .send()
            .await?;
        Ok(response)
    }
```

- Para obter detalhes da API, consulte a AssociateAddressreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->associateaddress(                        " oo_result
 is returned for testing purposes. "
            iv_allocationid = iv_allocation_id
            iv_instanceid = iv_instance_id ).
        MESSAGE 'Associated an Elastic IP address with an EC2 instance.' TYPE
 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AssociateAddressreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Associate the specified allocated Elastic IP to a given instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
 String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )

            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte AssociateAddressa referência da API AWS SDK for Swift.

Usar **`AssociateDhcpOptions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AssociateDhcpOptions`.

CLI

AWS CLI

Associar um conjunto de opções DHCP a uma VPC

Este exemplo associa o conjunto de opções de DHCP especificado à VPC especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 associate-dhcp-options --dhcp-options-id dopt-d9070ebb --vpc-id vpc-
a01106c2
```

Associar o conjunto de opções DHCP padrão associado a uma VPC

Este exemplo associa o conjunto padrão de opções de DHCP à VPC especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 associate-dhcp-options --dhcp-options-id default --vpc-id vpc-a01106c2
```

- Para obter detalhes da API, consulte AssociateDhcpOptionsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo associa o conjunto de opções de DHCP especificado à VPC especificada.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Exemplo 2: esse exemplo associa o conjunto padrão de opções de DHCP à VPC especificada.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte AssociateDhcpOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa o conjunto de opções de DHCP especificado à VPC especificada.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Exemplo 2: esse exemplo associa o conjunto padrão de opções de DHCP à VPC especificada.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte AssociateDhcpOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AssociateRouteTable`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AssociateRouteTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com servidores privados
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Associar uma tabela de rotas a uma sub-rede

Este exemplo associa a tabela de rotas especificada à sub-rede especificada.

Comando:

```
aws ec2 associate-route-table --route-table-id rtb-22574640 --subnet-
id subnet-9d4a7b6c
```

Saída:

```
{
    "AssociationId": "rtbassoc-781d0d1a"
}
```

- Para obter detalhes da API, consulte AssociateRouteTablena Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo associa a tabela de rotas especificada à sub-rede especificada.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Saída:

```
rtbassoc-12345678
```

- Para obter detalhes da API, consulte AssociateRouteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa a tabela de rotas especificada à sub-rede especificada.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Saída:

```
rtbassoc-12345678
```

- Para obter detalhes da API, consulte AssociateRouteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AttachInternetGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AttachInternetGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com servidores privados

CLI

AWS CLI

Anexar um gateway da internet à VPC

O exemplo `attach-internet-gateway` a seguir anexa o gateway da internet especificado à VPC específica.

```
aws ec2 attach-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE \
    --vpc-id vpc-0a60eb65b4EXAMPLE
```

Este comando não produz saída.

Para obter mais informações, consulte Gateways da Internet no Guia do usuário da Amazon VPC.

- Para obter detalhes da API, consulte AttachInternetGatewayna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo anexa o gateway da internet especificado à VPC especificada.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Exemplo 2: esse exemplo cria uma VPC e um gateway da internet e, depois, conecta o gateway da internet à VPC.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- Para obter detalhes da API, consulte AttachInternetGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo anexa o gateway da internet especificado à VPC especificada.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Exemplo 2: esse exemplo cria uma VPC e um gateway da internet e, depois, conecta o gateway da internet à VPC.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- Para obter detalhes da API, consulte AttachInternetGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AttachNetworkInterface`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AttachNetworkInterface`.

CLI

AWS CLI

Exemplo 1: anexar uma interface de rede a uma instância

O exemplo `attach-network-interface` a seguir anexa a interface de rede especificada à instância especificada.

```
aws ec2 attach-network-interface \
    --network-interface-id eni-0dc56a8d4640ad10a \
    --instance-id i-1234567890abcdef0 \
    --device-index 1
```

Saída:

```
{
    "AttachmentId": "eni-attach-01a8fc87363f07cf9"
}
```

Para obter mais informações, consulte Interfaces de rede elástica no Guia do usuário do Amazon EC2.

Exemplo 2: associar uma interface de rede a uma instância com vários cartões de rede

O exemplo `attach-network-interface` a seguir anexa a interface de rede especificada à instância e ao cartão de rede especificados.

```
aws ec2 attach-network-interface \
    --network-interface-id eni-07483b1897541ad83 \
    --instance-id i-01234567890abcdef \
    --network-card-index 1 \
    --device-index 1
```

Saída:

```
{
    "AttachmentId": "eni-attach-0fbd7ee87a88cd06c"
}
```

Para obter mais informações, consulte Interfaces de rede elástica no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte AttachNetworkInterfacena Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo anexa a interface de rede especificada à instância especificada.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

Saída:

```
eni-attach-1a2b3c4d
```

- Para obter detalhes da API, consulte AttachNetworkInterfaceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo anexa a interface de rede especificada à instância especificada.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

Saída:

```
eni-attach-1a2b3c4d
```

- Para obter detalhes da API, consulte AttachNetworkInterfaceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AttachVolume`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AttachVolume`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação e gerenciamento de volumes de armazenamento em bloco

CLI

AWS CLI

Anexar um volume a uma instância

Este exemplo de comando anexa um volume (`vol-1234567890abcdef0`) a uma instância (`i-01474ef662b89480`) como `/dev/sdf`.

Comando:

```
aws ec2 attach-volume --volume-id vol-1234567890abcdef0 --instance-
id i-01474ef662b89480 --device /dev/sdf
```

Saída:

```
{
    "AttachTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "InstanceId": "i-01474ef662b89480",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "attaching",
    "Device": "/dev/sdf"
}
```

- Para obter detalhes da API, consulte AttachVolumena Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo anexa o volume especificado à instância especificada e o expõe com o nome do dispositivo especificado.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Saída:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- Para obter detalhes da API, consulte AttachVolumeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo anexa o volume especificado à instância especificada e o expõe com o nome do dispositivo especificado.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Saída:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- Para obter detalhes da API, consulte AttachVolumeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AttachVpnGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AttachVpnGateway`.

CLI

AWS CLI

Anexar um gateway privado virtual à VPC

O exemplo `attach-vpn-gateway` a seguir anexa o gateway privado virtual especificado à VPC especificada.

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id vgw-9a4cacf3 \
    --vpc-id vpc-a01106c2
```

Saída:

```
{
    "VpcAttachment": {
        "State": "attaching",
        "VpcId": "vpc-a01106c2"
    }
}
```

- Para obter detalhes da API, consulte AttachVpnGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo anexa o gateway privado virtual especificado à VPC especificada.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Saída:

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- Para obter detalhes da API, consulte AttachVpnGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo anexa o gateway privado virtual especificado à VPC especificada.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Saída:

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- Para obter detalhes da API, consulte AttachVpnGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AuthorizeSecurityGroupEgress`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AuthorizeSecurityGroupEgress`.

CLI

AWS CLI

Exemplo 1: adicionar uma regra que permita tráfego de saída para um intervalo de endereços específico

O exemplo de `authorize-security-group-egress` a seguir adiciona uma regra que concede acesso aos intervalos de endereços especificados na porta TCP 80.

```
aws ec2 authorize-security-group-egress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=10.0.0.0/16}]'
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0b15794cdb17bf29c",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": true,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "10.0.0.0/16"
        }
    ]
}
```

Exemplo 2: adicionar uma regra que permita tráfego de saída a um grupo de segurança específico

O exemplo de `authorize-security-group-egress` a seguir adiciona uma regra que concede acesso ao grupo de segurança especificado na porta TCP 80.

```
aws ec2 authorize-security-group-egress \
    --group-id sg-1234567890abcdef0 \
```

```
    --ip-permissions
 'IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs=[{GroupId=sg-0aad1c26bbeec5c22}]'
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0b5dd815afcea9cc3",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": true,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "ReferencedGroupInfo": {
                "GroupId": "sg-0aad1c26bbeec5c22",
                "UserId": "123456789012"
            }
        }
    ]
}
```

Para obter mais informações, consulte Grupos de segurança no Manual do usuário da Amazon VPC.

- Para obter detalhes da API, consulte AuthorizeSecurityGroupEgressem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo define uma regra de saída para o grupo de segurança especificado para EC2-VPC. A regra concede acesso ao intervalo de endereços IP especificados na porta TCP 80. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: esse exemplo concede acesso ao grupo de segurança de origem especificado na porta TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupEgressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo define uma regra de saída para o grupo de segurança especificado para EC2-VPC. A regra concede acesso ao intervalo de endereços IP especificados na porta TCP 80. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
```

```
$ip.IpRanges.Add("203.0.113.0/24")

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: esse exemplo concede acesso ao grupo de segurança de origem especificado na porta TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupEgressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`AuthorizeSecurityGroupIngress`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AuthorizeSecurityGroupIngress`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Configurar a conectividade do serviço de contêiner
- Criar uma rede privada virtual básica
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criar uma instância de banco de dados relacional
- Criação e gerenciamento de volumes de armazenamento em bloco
- Conceitos básicos de compras do mercado de software
- Introdução aos bancos de dados de documentos
- Introdução aos bancos de dados de grafos
- Introdução ao armazenamento em cache em memória
- Introdução ao balanceamento de carga
- Conceitos básicos de streaming gerenciado

- Introdução às máquinas virtuais
- Rede privada virtual com servidores privados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Authorize the local computer ingress to EC2 instances associated
    /// with the virtual private cloud (VPC) security group.
    /// </summary>
    /// <param name="groupName">The name of the security group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
    {
        try
        {
            // Get the IP address for the local computer.
            var ipAddress = await GetIpAddress();
            Console.WriteLine($"Your IP address is: {ipAddress}");
            var ipRanges =
                new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
            var permission = new IpPermission
            {
                Ipv4Ranges = ipRanges,
                IpProtocol = "tcp",
                FromPort = 22,
                ToPort = 22
            };
            var permissions = new List<IpPermission> { permission };
            var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
                new AuthorizeSecurityGroupIngressRequest(groupName,
permissions));
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
            {
                _logger.LogError(
                    $"The ingress rule already exists. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while authorizing ingress.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer for ingress to
    /// the Amazon EC2 SecurityGroup.
    /// </summary>
    /// <returns>The IPv4 address of the computer running the scenario.</returns>
    private static async Task<string> GetIpAddress()
    {
        var httpClient = new HttpClient();
        var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

        // The IP address is returned with a new line
        // character on the end. Trim off the whitespace and
        // return the value to the caller.
        return ipString.Trim();
    }
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
 (Amazon EC2) security group.
#
# Parameters:
#       -g security_group_id - The ID of the security group.
#       -i ip_address - The IP address or CIDR block to authorize.
#       -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#       -f from_port - The start of the port range to authorize.
#       -t to_port - The end of the port range to authorize.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_authorize_security_group_ingress() {
  local security_group_id ip_address protocol from_port to_port response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
 EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
```

```
  }

  # Retrieve the calling parameters.
  while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      i) ip_address="${OPTARG}" ;;
      p) protocol="${OPTARG}" ;;
      f) from_port="${OPTARG}" ;;
      t) to_port="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
  fi

  if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
 parameter."
    usage
    return 1
  fi

  if [[ -z "$protocol" ]]; then
    errecho "ERROR: You must provide a protocol with the -p parameter."
    usage
    return 1
  fi

  if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
```

```
    usage
    return 1
  fi

  if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
  fi

  response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation
 failed.$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
```

```
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Authorize ingress to an Amazon Elastic Compute Cloud (Amazon EC2) group.
/*!
  \param groupID: The EC2 group ID.
  \param clientConfiguration: The ClientConfiguration object.
  \return bool: True if the operation was successful, false otherwise.
 */
bool
AwsDoc::EC2::authorizeSecurityGroupIngress(const Aws::String &groupID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest
 authorizeSecurityGroupIngressRequest;
    authorizeSecurityGroupIngressRequest.SetGroupId(groupID);
    buildSampleIngressRule(authorizeSecurityGroupIngressRequest);

    Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome
 authorizeSecurityGroupIngressOutcome =

 ec2Client.AuthorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);

    if (authorizeSecurityGroupIngressOutcome.IsSuccess()) {
        std::cout << "Successfully authorized security group ingress." <<
 std::endl;
    } else {
        std::cerr << "Error authorizing security group ingress: "
                  << authorizeSecurityGroupIngressOutcome.GetError().GetMessage()
 << std::endl;
    }

    return authorizeSecurityGroupIngressOutcome.IsSuccess();
}
```

Função utilitária para criar uma regra de entrada.

```
//! Build a sample ingress rule.
/*!
  \param authorize_request: An 'AuthorizeSecurityGroupIngressRequest' instance.
  \return void:
 */
void buildSampleIngressRule(
```

```
        Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest &authorize_request)
 {
    Aws::String ingressIPRange = "203.0.113.0/24";  // Configure this for your
 allowed IP range.
    Aws::EC2::Model::IpRange ip_range;
    ip_range.SetCidrIp(ingressIPRange);

    Aws::EC2::Model::IpPermission permission1;
    permission1.SetIpProtocol("tcp");
    permission1.SetToPort(80);
    permission1.SetFromPort(80);
    permission1.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission1);

    Aws::EC2::Model::IpPermission permission2;
    permission2.SetIpProtocol("tcp");
    permission2.SetToPort(22);
    permission2.SetFromPort(22);
    permission2.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission2);
}
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: adicionar uma regra que permita tráfego SSH de entrada

O exemplo `authorize-security-group-ingress` a seguir adiciona uma regra que permite o tráfego de entrada na porta TCP 22 (SSH).

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --protocol tcp \
    --port 22 \
    --cidr 203.0.113.0/24
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-01afa97ef3e1bedfc",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "203.0.113.0/24"
        }
    ]
}
```

Exemplo 2: adicionar uma regra que permita tráfego HTTP de entrada de outro grupo de segurança

O exemplo `authorize-security-group-ingress` a seguir adiciona uma regra que permite o acesso de entrada na porta TCP 80 do grupo de segurança de origem `sg-1a2b3c4d`. O grupo de origem deve estar na mesma VPC ou em uma VPC de par (requer uma conexão de emparelhamento da VPC). O tráfego de entrada é permitido com base nos endereços IP privados das instâncias associadas ao grupo de segurança de origem (e não ao endereço IP público ou ao endereço IP elástico).

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --protocol tcp \
    --port 80 \
    --source-group sg-1a2b3c4d
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-01f4be99110f638a7",
```

```
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "ReferencedGroupInfo": {
                "GroupId": "sg-1a2b3c4d",
                "UserId": "123456789012"
            }
        }
    ]
}
```

Exemplo 3: adicionar várias regras na mesma chamada

O `authorize-security-group-ingress` exemplo a seguir usa o `ip-permissions` parâmetro para adicionar duas regras de entrada, uma que permite o acesso de entrada na porta TCP 3389 (RDP) e a outra que ativa. ping/ICMP

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=3389,ToPort=3389,IpRanges=[{CidrIp=172.31.0.0/16}]'
 'IpProtocol=icmp,FromPort=-1,ToPort=-1,IpRanges=[{CidrIp=172.31.0.0/16}]'
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-00e06e5d3690f29f3",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 3389,
            "ToPort": 3389,
            "CidrIpv4": "172.31.0.0/16"
        },
        {
```

```
            "SecurityGroupRuleId": "sgr-0a133dd4493944b87",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": -1,
            "ToPort": -1,
            "CidrIpv4": "172.31.0.0/16"
        }
    ]
}
```

Exemplo 4: adicionar uma regra para o tráfego ICMP

O exemplo `authorize-security-group-ingress` a seguir usa o parâmetro `ip-permissions` para adicionar uma regra de entrada que permite a mensagem do ICMP `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` (Tipo 3, Código 4) de qualquer lugar.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=icmp,FromPort=3,ToPort=4,IpRanges=[{CidrIp=0.0.0.0/0}]'
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0de3811019069b787",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "icmp",
            "FromPort": 3,
            "ToPort": 4,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
}
```

Exemplo 5: adicionar uma regra para o tráfego IPv6

O exemplo `authorize-security-group-ingress` a seguir usa o parâmetro `ip-permissions` para adicionar uma regra de entrada que permite o acesso SSH (porta 22) do intervalo IPv6 `2001:db8:1234:1a00::/64`.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=22,ToPort=22,Ipv6Ranges=[{CidrIpv6=2001:db8:1234:1a00::/64}]'
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0455bc68b60805563",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv6": "2001:db8:1234:1a00::/64"
        }
    ]
}
```

Exemplo 6: adicionar uma regra para o tráfego ICMPv6

O exemplo `authorize-security-group-ingress` a seguir usa o parâmetro `ip-permissions` para adicionar uma regra de entrada que permite o tráfego ICMPv6 de qualquer lugar.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions 'IpProtocol=icmpv6,Ipv6Ranges=[{CidrIpv6=::/0}]'
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-04b612d9363ab6327",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "icmpv6",
            "FromPort": -1,
            "ToPort": -1,
            "CidrIpv6": "::/0"
        }
    ]
}
```

Exemplo 7: adicionar uma regra com uma descrição

O exemplo `authorize-security-group-ingress` a seguir usa o parâmetro `ip-permissions` para adicionar uma regra de entrada que permite o tráfego RDP do intervalo de endereços IPv4 especificado. A regra inclui uma descrição para ajudar a identificá-lo posteriormente.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=3389,ToPort=3389,IpRanges=[{CidrIp=203.0.113.0/24,Description='
 office'}]'
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0397bbcc01e974db3",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 3389,
```

```
            "ToPort": 3389,
            "CidrIpv4": "203.0.113.0/24",
            "Description": "RDP access from NY office"
        }
    ]
}
```

Exemplo 8: adicionar uma regra de entrada que use uma lista de prefixos

O exemplo `authorize-security-group-ingress` a seguir usa o parâmetro `ip-permissions` para adicionar uma regra de entrada que permite todo o tráfego para os intervalos CIDR na lista de prefixos especificada.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-04a351bfe432d4e71 \
    --ip-permissions
 'IpProtocol=all,PrefixListIds=[{PrefixListId=pl-002dc3ec097de1514}]'
```

Saída:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-09c74b32f677c6c7c",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "-1",
            "FromPort": -1,
            "ToPort": -1,
            "PrefixListId": "pl-0721453c7ac4ec009"
        }
    ]
}
```

Para obter mais informações, consulte Grupos de segurança no Manual do usuário da Amazon VPC.

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP
address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security
group
    * @param myIpAddress  the IP address from which to allow inbound traffic
(e.g., "192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security
group or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
```

```
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  AuthorizeSecurityGroupIngressCommand,
  EC2Client,
} from "@aws-sdk/client-ec2";

/**
 * Adds the specified inbound (ingress) rules to a security group.
 * @param {{ groupId: string, ipAddress: string }} options
 */
export const main = async ({ groupId, ipAddress }) => {
  const client = new EC2Client({});
  const command = new AuthorizeSecurityGroupIngressCommand({
    // Use a group ID from the AWS console or
    // the DescribeSecurityGroupsCommand.
    GroupId: groupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        // The IP address to authorize.
        // For more information on this notation, see
        // https://en.wikipedia.org/wiki/Classless_Inter-
Domain_Routing#CIDR_notation
        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
      },
    ],
  });

  try {
    const { SecurityGroupRules } = await client.send(command);
    console.log(JSON.stringify(SecurityGroupRules, null, 2));
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
```

```
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group
 $groupNameVal")
        return resp.groupId
    }
}
```

- Para obter detalhes da API, consulte a AuthorizeSecurityGroupIngressreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo define regras de entrada para um grupo de segurança para EC2-VPC. Essas regras concedem acesso a um endereço IP específico para SSH (porta 22) e RDC (porta 3389). Observe que você deve identificar grupos de segurança para EC2-VPC

usar o ID do grupo de segurança, não o nome do grupo de segurança. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar os IpPermission objetos.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Exemplo 3: Este exemplo define regras de entrada para um grupo de segurança para EC2-Classic. Essas regras concedem acesso a um endereço IP específico para SSH (porta 22) e RDC (porta 3389). A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
 @( $ip1, $ip2 )
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar os IpPermission objetos.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
 @( $ip1, $ip2 )
```

Exemplo 5: esse exemplo concede acesso à porta TCP 8081 do grupo de segurança de origem especificado (sg-1a2b3c4d) ao grupo de segurança especificado (sg-12345678).

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

Exemplo 6: Este exemplo adiciona o CIDR 5.5.5. 5/32 às regras de entrada do grupo de segurança sg-1234abcd para tráfego da porta TCP 22 com uma descrição.

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo define regras de entrada para um grupo de segurança para EC2-VPC. Essas regras concedem acesso a um endereço IP específico para SSH (porta 22) e RDC (porta 3389). Observe que você deve identificar grupos de segurança para EC2-VPC usar o ID do grupo de segurança, não o nome do grupo de segurança. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar os IpPermission objetos.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Exemplo 3: Este exemplo define regras de entrada para um grupo de segurança para EC2-Classic. Essas regras concedem acesso a um endereço IP específico para SSH (porta 22) e RDC (porta 3389). A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }
```

```
Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
 @( $ip1, $ip2 )
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar os IpPermission objetos.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
 @( $ip1, $ip2 )
```

Exemplo 5: esse exemplo concede acesso à porta TCP 8081 do grupo de segurança de origem especificado (sg-1a2b3c4d) ao grupo de segurança especificado (sg-12345678).

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

Exemplo 6: Este exemplo adiciona o CIDR 5.5.5. 5/32 às regras de entrada do grupo de segurança sg-1234abcd para tráfego da porta TCP 22 com uma descrição.

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
```

```
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.
```

```
        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                               to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                 response indicates whether the request succeeded or failed, or
None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"
                    f"in security group '{self.security_group}'."
```

```
                )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a AuthorizeSecurityGroupIngressReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// Add an ingress rule to a security group explicitly allowing IPv4 address
    /// as {ip}/32 over TCP port 22.
    pub async fn authorize_security_group_ssh_ingress(
        &self,
        group_id: &str,
        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
```

```
                            .build()
                    })
                    .collect(),
            ))
            .send()
            .await?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a AuthorizeSecurityGroupIngressreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create IP permissions for SSH access (port 22)
    " iv_cidr_ip = '192.0.2.0/24'
    DATA lt_ip_permissions TYPE /aws1/cl_ec2ippermission=>tt_ippermissionlist.
    DATA(lo_ip_permission) = NEW /aws1/cl_ec2ippermission(
      iv_ipprotocol = 'tcp'
      iv_fromport = 22
      iv_toport = 22
      it_ipranges = VALUE /aws1/cl_ec2iprange=>tt_iprangelist(
        ( NEW /aws1/cl_ec2iprange( iv_cidrip = iv_cidr_ip ) )
      )
    ).
    APPEND lo_ip_permission TO lt_ip_permissions.

    TRY.
        oo_result = lo_ec2->authsecuritygroupingress(             " oo_result is
returned for testing purposes. "
          iv_groupid = iv_group_id
          it_ippermissions = lt_ip_permissions ).
```

```
        MESSAGE 'Authorized ingress rule for security group.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AuthorizeSecurityGroupIngressreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Authorize ingress of connections for the security group.
    ///
    /// - Parameters:
    ///   - groupId: The group ID of the security group to authorize access for.
    ///   - ipAddress: The IP address of the device to grant access to.
    ///
    /// - Returns: `true` if access is successfully granted; otherwise `false`.
    func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async
 -> Bool {
        let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
        let httpPermission = EC2ClientTypes.IpPermission(
            fromPort: 80,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 80
        )

        let sshPermission = EC2ClientTypes.IpPermission(
```

```
            fromPort: 22,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 22
        )

        do {
            _ = try await ec2Client.authorizeSecurityGroupIngress(
                input: AuthorizeSecurityGroupIngressInput(
                    groupId: groupId,
                    ipPermissions: [httpPermission, sshPermission]
                )
            )

            return true
        } catch {
            print("*** Error authorizing ingress for the security group:
\(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngressa referência da API AWS SDK for Swift.

Usar **`CancelCapacityReservation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CancelCapacityReservation`.

CLI

AWS CLI

Cancelar uma reserva de capacidade

O exemplo `cancel-capacity-reservation` a seguir cancela a reserva de capacidade especificada.

```
aws ec2 cancel-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE
```

Saída:

```
{
    "Return": true
}
```

Consulte mais informações em Cancelar uma Reserva de capacidade no Manual do usuário do Amazon EC2.

- Para obter detalhes da API, consulte CancelCapacityReservationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela a reserva de capacidade cr-0c1f2345db6f7cdba

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation
 (CancelCapacityReservation)" on target "cr-0c1f2345db6f7cdba".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
True
```

- Para obter detalhes da API, consulte CancelCapacityReservationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a reserva de capacidade cr-0c1f2345db6f7cdba

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation
 (CancelCapacityReservation)" on target "cr-0c1f2345db6f7cdba".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
True
```

- Para obter detalhes da API, consulte CancelCapacityReservationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CancelImportTask`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CancelImportTask`.

CLI

AWS CLI

Cancelar uma tarefa de importação

O exemplo `cancel-import-task` a seguir cancela a tarefa de importação de imagem especificada.

```
aws ec2 cancel-import-task \
    --import-task-id import-ami-1234567890abcdef0
```

Saída:

```
{
    "ImportTaskId": "import-ami-1234567890abcdef0",
    "PreviousState": "active",
    "State": "deleting"
}
```

- Para obter detalhes da API, consulte CancelImportTaskem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela a tarefa de importação especificada (importação de snapshot ou de imagem). Se necessário, um motivo pode ser fornecido usando o parâmetro **-CancelReason**.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- Para obter detalhes da API, consulte CancelImportTaskem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a tarefa de importação especificada (importação de snapshot ou de imagem). Se necessário, um motivo pode ser fornecido usando o parâmetro **-CancelReason**.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- Para obter detalhes da API, consulte CancelImportTaskem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **CancelSpotFleetRequests** com uma CLI

Os exemplos de código a seguir mostram como usar o `CancelSpotFleetRequests`.

CLI

AWS CLI

Exemplo 1: cancelar uma solicitação de frota Spot e encerrar as instâncias associadas

O `cancel-spot-fleet-requests` exemplo a seguir cancela uma solicitação de frota spot e encerra as On-Demand instâncias e instâncias spot associadas.

```
aws ec2 cancel-spot-fleet-requests \
    --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
    --terminate-instances
```

Saída:

```
{
    "SuccessfulFleetRequests": [
        {
            "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
            "CurrentSpotFleetRequestState": "cancelled_terminating",
            "PreviousSpotFleetRequestState": "active"
        }
    ],
    "UnsuccessfulFleetRequests": []
}
```

Exemplo 2: cancelar uma solicitação de frota Spot sem encerrar as instâncias associadas

O `cancel-spot-fleet-requests` exemplo a seguir cancela uma solicitação de frota spot sem encerrar as On-Demand instâncias e instâncias spot associadas.

```
aws ec2 cancel-spot-fleet-requests \
    --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
    --no-terminate-instances
```

Saída:

```
{
    "SuccessfulFleetRequests": [
        {
            "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
            "CurrentSpotFleetRequestState": "cancelled_running",
            "PreviousSpotFleetRequestState": "active"
        }
    ],
    "UnsuccessfulFleetRequests": []
}
```

Consulte mais informações em Cancelar uma solicitação de frota spot no Manual do usuário do Amazon EC2.

- Para obter detalhes da API, consulte CancelSpotFleetRequestsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela a solicitação de frota spot especificada e encerra as instâncias spot associadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Exemplo 2: esse exemplo cancela a solicitação de frota spot especificada sem encerrar as instâncias spot associadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- Para obter detalhes da API, consulte CancelSpotFleetRequestsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a solicitação de frota spot especificada e encerra as instâncias spot associadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Exemplo 2: esse exemplo cancela a solicitação de frota spot especificada sem encerrar as instâncias spot associadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- Para obter detalhes da API, consulte CancelSpotFleetRequestsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CancelSpotInstanceRequests`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CancelSpotInstanceRequests`.

CLI

AWS CLI

Cancelar solicitações de Instância Spot

Este exemplo de comando cancela uma solicitação de Instância Spot.

Comando:

```
aws ec2 cancel-spot-instance-requests --spot-instance-request-ids sir-08b93456
```

Saída:

```
{
    "CancelledSpotInstanceRequests": [
        {
            "State": "cancelled",
            "SpotInstanceRequestId": "sir-08b93456"
        }
    ]
}
```

- Para obter detalhes da API, consulte CancelSpotInstanceRequestsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela a solicitação de instância spot especificada.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Saída:

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- Para obter detalhes da API, consulte CancelSpotInstanceRequestsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a solicitação de instância spot especificada.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Saída:

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- Para obter detalhes da API, consulte CancelSpotInstanceRequestsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ConfirmProductInstance`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ConfirmProductInstance`.

CLI

AWS CLI

Confirmar a instância do produto

Esse exemplo determina se o código do produto especificado está associado à instância especificada.

Comando:

```
aws ec2 confirm-product-instance --product-code 774F4FF8 --instance-
id i-1234567890abcdef0
```

Saída:

```
{
  "OwnerId": "123456789012"
}
```

- Para obter detalhes da API, consulte ConfirmProductInstanceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo determina se o código do produto especificado está associado à instância especificada.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- Para obter detalhes da API, consulte ConfirmProductInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo determina se o código do produto especificado está associado à instância especificada.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- Para obter detalhes da API, consulte ConfirmProductInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CopyImage`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CopyImage`.

CLI

AWS CLI

Exemplo 1: copiar uma AMI para outra região

O comando de exemplo `copy-image` a seguir copia a AMI especificada da região `us-west-2` para a região `us-east-1` e adiciona uma breve descrição.

```
aws ec2 copy-image \
    --region us-east-1 \
    --name ami-name \
```

```
    --source-region us-west-2 \
    --source-image-id ami-066877671789bd71b \
    --description "This is my copied image."
```

Saída:

```
{
    "ImageId": "ami-0123456789abcdefg"
}
```

Para obter mais informações, consulte Copiar uma AMI no Guia do usuário do Amazon EC2.

Exemplo 2: copiar uma AMI para outra região e criptografar o snapshot de apoio

O comando `copy-image` a seguir copia a AMI especificada da região `us-west-2` para a região atual e criptografa o snapshot de backup usando a chave KMS especificada.

```
aws ec2 copy-image \
    --source-region us-west-2 \
    --name ami-name \
    --source-image-id ami-066877671789bd71b \
    --encrypted \
    --kms-key-id alias/my-kms-key
```

Saída:

```
{
    "ImageId": "ami-0123456789abcdefg"
}
```

Para obter mais informações, consulte Copiar uma AMI no Guia do usuário do Amazon EC2.

Exemplo 3: incluir as tags de AMI definidas pelo usuário ao copiar uma AMI

O comando `copy-image` a seguir usa o parâmetro `--copy-image-tags` para copiar as tags de AMI definidas pelo usuário ao copiar a AMI.

```
aws ec2 copy-image \
    --region us-east-1 \
    --name ami-name \
    --source-region us-west-2 \
```

```
    --source-image-id ami-066877671789bd71b \
    --description "This is my copied image."
    --copy-image-tags
```

Saída:

```
{
    "ImageId": "ami-0123456789abcdefg"
}
```

Para obter mais informações, consulte Copiar uma AMI no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte CopyImageem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo copia a AMI especificada na região “UE (Irlanda)” para a região “Oeste dos EUA (Oregon)”. Se -Region não for especificada, a região padrão atual será usada como a região de destino.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-
west-2 -Name "Copy of ami-12345678"
```

Saída:

```
ami-87654321
```

- Para obter detalhes da API, consulte CopyImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo copia a AMI especificada na região “UE (Irlanda)” para a região “Oeste dos EUA (Oregon)”. Se -Region não for especificada, a região padrão atual será usada como a região de destino.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-
west-2 -Name "Copy of ami-12345678"
```

Saída:

```
ami-87654321
```

- Para obter detalhes da API, consulte CopyImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CopySnapshot`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CopySnapshot`.

CLI

AWS CLI

Exemplo 1: copiar um snapshot para outra região

O comando de exemplo `copy-snapshot` a seguir copia o snapshot especificado da região `us-west-2` para a região `us-east-1` e adiciona uma breve descrição.

```
aws ec2 copy-snapshot \
    --region us-east-1 \
    --source-region us-west-2 \
    --source-snapshot-id snap-066877671789bd71b \
    --description 'This is my copied snapshot.'
```

Saída:

```
{
    "SnapshotId": "snap-066877671789bd71b"
}
```

Exemplo 2: copiar um snapshot não criptografado e criptografar o novo snapshot

O comando `copy-snapshot` a seguir copia o snapshot não criptografado especificado da região `us-west-2` para a região atual e criptografa o novo snapshot usando a chave KMS especificada.

```
aws ec2 copy-snapshot \
    --source-region us-west-2 \
    --source-snapshot-id snap-066877671789bd71b \
```

```
    --encrypted \
    --kms-key-id alias/my-kms-key
```

Saída:

```
{
    "SnapshotId": "snap-066877671789bd71b"
}
```

Consulte mais informações em Copy an Amazon EBS snapshot no Guia do usuário do Amazon EBS.

- Para obter detalhes da API, consulte CopySnapshotem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo copia a AMI especificada na região “UE (Irlanda)” para a região “Oeste dos EUA (Oregon)”.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region
 us-west-2
```

Exemplo 2: se você definir uma região padrão e omitir o parâmetro Região, a região de destino padrão será a região padrão.

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- Para obter detalhes da API, consulte CopySnapshotem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo copia a AMI especificada na região “UE (Irlanda)” para a região “Oeste dos EUA (Oregon)”.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region
 us-west-2
```

Exemplo 2: se você definir uma região padrão e omitir o parâmetro Região, a região de destino padrão será a região padrão.

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- Para obter detalhes da API, consulte CopySnapshotem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateCapacityReservation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateCapacityReservation`.

CLI

AWS CLI

Exemplo 1: criar uma reserva de capacidade

O `create-capacity-reservation` exemplo a seguir cria uma reserva de capacidade na zona de `eu-west-1a` disponibilidade, na qual você pode executar três `t2.medium` instâncias executando um sistema Linux/Unix operacional. Por padrão, a reserva de capacidade é criada com critérios de correspondência de instâncias abertas e sem suporte para armazenamento temporário, e permanece ativa até que você a cancele manualmente.

```
aws ec2 create-capacity-reservation \
    --availability-zone eu-west-1a \
    --instance-type t2.medium \
    --instance-platform Linux/UNIX \
    --instance-count 3
```

Saída:

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
        "EndDateType": "unlimited",
        "AvailabilityZone": "eu-west-1a",
        "InstanceMatchCriteria": "open",
        "EphemeralStorage": false,
```

```
        "CreateDate": "2019-08-16T09:27:35.000Z",
        "AvailableInstanceCount": 3,
        "InstancePlatform": "Linux/UNIX",
        "TotalInstanceCount": 3,
        "State": "active",
        "Tenancy": "default",
        "EbsOptimized": false,
        "InstanceType": "t2.medium"
    }
}
```

Exemplo 2: Para criar uma reserva de capacidade que termine automaticamente em um determinado date/time

O `create-capacity-reservation` exemplo a seguir cria uma reserva de capacidade na zona de `eu-west-1a` disponibilidade, na qual você pode executar três `m5.large` instâncias executando um sistema Linux/Unix operacional. Essa reserva de capacidade termina automaticamente em 08/31 /2019 às 23:59:59.

```
aws ec2 create-capacity-reservation \
    --availability-zone eu-west-1a \
    --instance-type m5.large \
    --instance-platform Linux/UNIX \
    --instance-count 3 \
    --end-date-type limited \
    --end-date 2019-08-31T23:59:59Z
```

Saída:

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
        "EndDateType": "limited",
        "AvailabilityZone": "eu-west-1a",
        "EndDate": "2019-08-31T23:59:59.000Z",
        "InstanceMatchCriteria": "open",
        "EphemeralStorage": false,
        "CreateDate": "2019-08-16T10:15:53.000Z",
        "AvailableInstanceCount": 3,
        "InstancePlatform": "Linux/UNIX",
        "TotalInstanceCount": 3,
        "State": "active",
```

```
        "Tenancy": "default",
        "EbsOptimized": false,
        "InstanceType": "m5.large"
    }
}
```

Exemplo 3: criar uma reserva de capacidade que aceite somente execuções direcionadas de instância

O exemplo `create-capacity-reservation` a seguir cria uma reserva de capacidade que aceita somente lançamentos de instâncias direcionadas.

```
aws ec2 create-capacity-reservation \
    --availability-zone eu-west-1a \
    --instance-type m5.large \
    --instance-platform Linux/UNIX \
    --instance-count 3 \
    --instance-match-criteria targeted
```

Saída:

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
        "EndDateType": "unlimited",
        "AvailabilityZone": "eu-west-1a",
        "InstanceMatchCriteria": "targeted",
        "EphemeralStorage": false,
        "CreateDate": "2019-08-16T10:21:57.000Z",
        "AvailableInstanceCount": 3,
        "InstancePlatform": "Linux/UNIX",
        "TotalInstanceCount": 3,
        "State": "active",
        "Tenancy": "default",
        "EbsOptimized": false,
        "InstanceType": "m5.large"
    }
}
```

Consulte mais informações em Criar uma Reserva de capacidade no Manual do usuário do Amazon EC2.

- Para obter detalhes da API, consulte CreateCapacityReservationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma reserva de capacidade com os atributos especificados.

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

Saída:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
State                  : active
Tags                   : {}
Tenancy                : default
TotalInstanceCount     : 2
```

- Para obter detalhes da API, consulte CreateCapacityReservationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma reserva de capacidade com os atributos especificados.

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

Saída:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
State                  : active
Tags                   : {}
Tenancy                : default
TotalInstanceCount     : 2
```

- Para obter detalhes da API, consulte CreateCapacityReservationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateCustomerGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateCustomerGateway`.

CLI

AWS CLI

Criar um gateway do cliente

Este exemplo cria um gateway do cliente com o endereço IP especificado para sua interface externa.

Comando:

```
aws ec2 create-customer-gateway --type ipsec.1 --public-ip 12.1.2.3 --bgp-
asn 65534
```

Saída:

```
{
    "CustomerGateway": {
```

```
        "CustomerGatewayId": "cgw-0e11f167",
        "IpAddress": "12.1.2.3",
        "State": "available",
        "Type": "ipsec.1",
        "BgpAsn": "65534"
    }
}
```

- Para obter detalhes da API, consulte CreateCustomerGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria o gateway do cliente especificado.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

Saída:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- Para obter detalhes da API, consulte CreateCustomerGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria o gateway do cliente especificado.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

Saída:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
```

```
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- Para obter detalhes da API, consulte CreateCustomerGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateDhcpOptions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateDhcpOptions`.

CLI

AWS CLI

Criar um conjunto de opções DHCP

O exemplo `create-dhcp-options` a seguir cria um conjunto de opções DHCP que especifica o nome do domínio, os servidores de nomes de domínio e o tipo de nó NetBIOS.

```
aws ec2 create-dhcp-options \
    --dhcp-configuration \
        "Key=domain-name-servers,Values=10.2.5.1,10.2.5.2" \
        "Key=domain-name,Values=example.com" \
        "Key=netbios-node-type,Values=2"
```

Saída:

```
{
    "DhcpOptions": {
        "DhcpConfigurations": [
            {
                "Key": "domain-name",
                "Values": [
                    {
                        "Value": "example.com"
                    }
                ]
            },
            {
                "Key": "domain-name-servers",
```

```
                "Values": [
                    {
                        "Value": "10.2.5.1"
                    },
                    {
                        "Value": "10.2.5.2"
                    }
                ]
            },
            {
                "Key": "netbios-node-type",
                "Values": [
                    {
                        "Value": "2"
                    }
                ]
            }
        ],
        "DhcpOptionsId": "dopt-06d52773eff4c55f3"
    }
}
```

- Para obter detalhes da API, consulte CreateDhcpOptionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria o conjunto especificado de opções DHCP. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

Saída:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada opção DHCP.

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @("10.0.0.101","10.0.0.102")

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

Saída:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- Para obter detalhes da API, consulte CreateDhcpOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria o conjunto especificado de opções DHCP. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

Saída:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada opção DHCP.

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
```

```
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @("10.0.0.101","10.0.0.102")

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

Saída:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- Para obter detalhes da API, consulte CreateDhcpOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateFlowLogs`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateFlowLogs`.

CLI

AWS CLI

Exemplo 1: criar um log de fluxo

O exemplo `create-flow-logs` a seguir cria um log de fluxo que captura todo o tráfego rejeitado para a interface de rede especificada. Os registros de fluxo são entregues a um grupo de CloudWatch registros em Logs usando as permissões na função do IAM especificada.

```
aws ec2 create-flow-logs \
    --resource-type NetworkInterface \
    --resource-ids eni-11223344556677889 \
    --traffic-type REJECT \
    --log-group-name my-flow-logs \
    --deliver-logs-permission-arn arn:aws:iam::123456789101:role/publishFlowLogs
```

Saída:

```
{
    "ClientToken": "so0eNA2uSHUNlHI0S2cJ305GuIX1CezaRdGtexample",
    "FlowLogIds": [
        "fl-12345678901234567"
    ],
    "Unsuccessful": []
}
```

Para obter mais informações, consulte Logs de fluxo da VPC no Guia do usuário do Amazon Virtual Private Cloud.

Exemplo 2: criar um log de fluxo com um formato personalizado

O exemplo `create-flow-logs` a seguir cria um log de fluxo que captura todo o tráfego da VPC especificada e fornece os logs de fluxo a um bucket do Amazon S3. O parâmetro `--log-format` especifica um formato personalizado para os registros de log de fluxo. Para executar esse comando no Windows, altere as aspas simples (') para aspas duplas (").

```
aws ec2 create-flow-logs \
    --resource-type VPC \
    --resource-ids vpc-00112233344556677 \
    --traffic-type ALL \
    --log-destination-type s3 \
    --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \
    --log-format '${version} ${vpc-id} ${subnet-id} ${instance-id} ${srcaddr}
 ${dstaddr} ${srcport} ${dstport} ${protocol} ${tcp-flags} ${type} ${pkt-srcaddr}
 ${pkt-dstaddr}'
```

Para obter mais informações, consulte Logs de fluxo da VPC no Guia do usuário do Amazon Virtual Private Cloud.

Exemplo 3: criar um log de fluxo com um intervalo máximo de agregação de um minuto

O exemplo `create-flow-logs` a seguir cria um log de fluxo que captura todo o tráfego da VPC especificada e fornece os logs de fluxo a um bucket do Amazon S3. O parâmetro `--max-aggregation-interval` especifica um intervalo de agregação máximo de 60 segundos (um minuto).

```
aws ec2 create-flow-logs \
    --resource-type VPC \
    --resource-ids vpc-00112233344556677 \
```

```
    --traffic-type ALL \
    --log-destination-type s3 \
    --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \
    --max-aggregation-interval 60
```

Para obter mais informações, consulte Logs de fluxo da VPC no Guia do usuário do Amazon Virtual Private Cloud.

- Para obter detalhes da API, consulte CreateFlowLogsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um log de fluxo do EC2 para a sub-rede subnet-1d234567 para o cloud-watch-log chamado “subnet1-log” para todo o tráfego “REJECT” usando as permissões do perfil “Admin”

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Saída:

```
ClientToken                                  FlowLogIds            Unsuccessful
-----------                                  ----------            ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- Para obter detalhes da API, consulte CreateFlowLogsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um log de fluxo do EC2 para a sub-rede subnet-1d234567 para o cloud-watch-log chamado “subnet1-log” para todo o tráfego “REJECT” usando as permissões do perfil “Admin”

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Saída:

```
ClientToken                                 FlowLogIds             Unsuccessful
-----------                                 ----------             ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- Para obter detalhes da API, consulte CreateFlowLogsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateImage`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateImage`.

CLI

AWS CLI

Exemplo 1: Para criar uma AMI a partir de uma EBS-backed instância da Amazon

O exemplo `create-image` a seguir cria uma AMI da instância especificada.

```
aws ec2 create-image \
    --instance-id i-1234567890abcdef0 \
    --name "My server" \
    --description "An AMI for my server"
```

Saída:

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

Para obter mais informações sobre especificar um mapeamento de dispositivos de blocos para sua AMI, consulte Especificar um mapeamento de dispositivos de blocos para AMI no Guia do usuário do Amazon EC2.

Exemplo 2: Para criar uma AMI a partir de uma EBS-backed instância da Amazon sem reinicializar

O exemplo `create-image` a seguir cria uma AMI e define o parâmetro --no-reboot, para que a instância não seja reinicializada antes da criação da imagem.

```
aws ec2 create-image \
    --instance-id i-1234567890abcdef0 \
    --name "My server" \
    --no-reboot
```

Saída:

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

Para obter mais informações sobre especificar um mapeamento de dispositivos de blocos para sua AMI, consulte Especificar um mapeamento de dispositivos de blocos para AMI no Guia do usuário do Amazon EC2.

Exemplo 3: aplicar tag em uma AMI e snapshots na criação

O exemplo `create-image` a seguir cria uma AMI e aplica uma tag na AMI e nos snapshots com a mesma tag `cost-center=cc123`

```
aws ec2 create-image \
    --instance-id i-1234567890abcdef0 \
    --name "My server" \
    --tag-specifications "ResourceType=image,Tags=[{Key=cost-
center,Value=cc123}]" "ResourceType=snapshot,Tags=[{Key=cost-
center,Value=cc123}]"
```

Saída:

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

Para obter mais informações sobre como aplicar tags em seus recursos na criação, consulte Adicionar tags na criação de recursos no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte CreateImageem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma AMI com o nome e a descrição especificados com base na instância especificada. O Amazon EC2 tenta desligar a instância de forma limpa antes de criar a imagem e reinicia a instância após a conclusão.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

Exemplo 2: esse exemplo cria uma AMI com o nome e a descrição especificados com base na instância especificada. O Amazon EC2 cria a imagem sem desligar e reiniciar a instância; portanto, a integridade do sistema de arquivos na imagem criada não pode ser garantida.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

Exemplo 3: esse exemplo cria uma AMI com três volumes. O primeiro volume é baseado em um snapshot do Amazon EBS. O segundo volume é um volume vazio de 100 GiB do Amazon EBS. O terceiro volume é um volume de armazenamento de instância. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
 "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
$ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
sdc";VirtualName="ephemeral0"})
```

- Para obter detalhes da API, consulte CreateImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma AMI com o nome e a descrição especificados com base na instância especificada. O Amazon EC2 tenta desligar a instância de forma limpa antes de criar a imagem e reinicia a instância após a conclusão.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

Exemplo 2: esse exemplo cria uma AMI com o nome e a descrição especificados com base na instância especificada. O Amazon EC2 cria a imagem sem desligar e reiniciar a instância; portanto, a integridade do sistema de arquivos na imagem criada não pode ser garantida.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

Exemplo 3: esse exemplo cria uma AMI com três volumes. O primeiro volume é baseado em um snapshot do Amazon EBS. O segundo volume é um volume vazio de 100 GiB do Amazon EBS. O terceiro volume é um volume de armazenamento de instância. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
 "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
$ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
sdc";VirtualName="ephemeral0"})
```

- Para obter detalhes da API, consulte CreateImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateInstanceExportTask`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateInstanceExportTask`.

CLI

AWS CLI

Exportar uma instância

Esse comando de exemplo cria uma tarefa para exportar a instância i-1234567890abcdef0 para o bucket myexportbucket do Amazon S3.

Comando:

```
aws ec2 create-instance-export-task --description "RHEL5 instance" --
instance-id i-1234567890abcdef0 --target-environment vmware --export-to-s3-
task DiskImageFormat=vmdk,ContainerFormat=ova,S3Bucket=myexportbucket,S3Prefix=RHEL5
```

Saída:

```
{
    "ExportTask": {
        "State": "active",
        "InstanceExportDetails": {
            "InstanceId": "i-1234567890abcdef0",
            "TargetEnvironment": "vmware"
        },
        "ExportToS3Task": {
            "S3Bucket": "myexportbucket",
            "S3Key": "RHEL5export-i-fh8sjjsq.ova",
            "DiskImageFormat": "vmdk",
            "ContainerFormat": "ova"
        },
        "Description": "RHEL5 instance",
        "ExportTaskId": "export-i-fh8sjjsq"
    }
}
```

- Para obter detalhes da API, consulte CreateInstanceExportTaskem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exporta uma instância interrompida, **`i-0800b00a00EXAMPLE`**, como um disco rígido virtual (VHD) para o bucket do S3 **`testbucket-export-instances-2019`**. O ambiente de destino é**`Microsoft`**, e o parâmetro region é adicionado porque a instância está na **`us-east-1`** região, enquanto a AWS região padrão do usuário não é us-east-1. Para obter o status da tarefa de exportação, copie o valor **`ExportTaskId`** dos resultados desse comando e execute **`Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.`**

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"
 -TargetEnvironment Microsoft -Region us-east-1
```

Saída:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- Para obter detalhes da API, consulte CreateInstanceExportTaskem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exporta uma instância interrompida, **`i-0800b00a00EXAMPLE`**, como um disco rígido virtual (VHD) para o bucket do S3 **`testbucket-export-instances-2019`**. O ambiente de destino é**`Microsoft`**, e o parâmetro region é adicionado porque a instância está na **`us-east-1`** região, enquanto a AWS região padrão do usuário não é us-east-1. Para obter o status da tarefa de exportação, copie o valor **`ExportTaskId`** dos resultados desse comando e execute **`Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.`**

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"
 -TargetEnvironment Microsoft -Region us-east-1
```

Saída:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- Para obter detalhes da API, consulte CreateInstanceExportTaskem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateInternetGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateInternetGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com servidores privados

CLI

AWS CLI

Criar um gateway da Internet

O exemplo `create-internet-gateway` a seguir cria um gateway da Internet com a tag `Name=my-igw`.

```
aws ec2 create-internet-gateway \
    --tag-specifications ResourceType=internet-gateway,Tags=[{Key=Name,Value=my-igw}]
```

Saída:

```
{
    "InternetGateway": {
        "Attachments": [],
        "InternetGatewayId": "igw-0d0fb496b3994d755",
        "OwnerId": "123456789012",
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-igw"
            }
        ]
    }
}
```

Para obter mais informações, consulte Gateways da Internet no Guia do usuário da Amazon VPC.

- Para obter detalhes da API, consulte CreateInternetGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um gateway da internet.

```
New-EC2InternetGateway
```

Saída:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- Para obter detalhes da API, consulte CreateInternetGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um gateway da internet.

```
New-EC2InternetGateway
```

Saída:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- Para obter detalhes da API, consulte CreateInternetGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateKeyPair`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateKeyPair`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma rede privada virtual básica
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Conceitos básicos de compras do mercado de software
- Introdução aos clusters de processamento de big data
- Conceitos básicos de streaming gerenciado
- Introdução às máquinas virtuais

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create an Amazon EC2 key pair with a specified name.
/// </summary>
/// <param name="keyPairName">The name for the new key pair.</param>
/// <returns>The Amazon EC2 key pair created.</returns>
public async Task<KeyPair?> CreateKeyPair(string keyPairName)
{
    try
    {
        var request = new CreateKeyPairRequest { KeyName = keyPairName, };

        var response = await _amazonEC2.CreateKeyPairAsync(request);

        var kp = response.KeyPair;
        // Return the key pair so it can be saved if needed.

        // Wait until the key pair exists.
        int retries = 5;
```

```
            while (retries-- > 0)
            {
                Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
                var keyPairs = await DescribeKeyPairs(keyPairName);
                if (keyPairs.Any())
                {
                    return kp;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
            throw new DoesNotExistException("KeyPair not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} already exists.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the key pair.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Save KeyPair information to a temporary file.
    /// </summary>
    /// <param name="keyPair">The name of the key pair.</param>
    /// <returns>The full path to the temporary file.</returns>
    public string SaveKeyPair(KeyPair keyPair)
    {
        var tempPath = Path.GetTempPath();
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");
```

```
        // Save the key pair to a file in a temporary folder.
        using var stream = new FileStream(pemFileName, FileMode.Create);
        using var writer = new StreamWriter(stream);
        writer.WriteLine(keyPair.KeyMaterial);

        return pemFileName;
    }
```

- Para obter detalhes da API, consulte CreateKeyPaira Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#       -f file_path - File to store the key pair.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_create_keypair() {
  local key_pair_name file_path response
  local option OPTARG # Required to use getopts command in a function.
```

```
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_create_keypair"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-
bit RSA key pair"
    echo " and writes it to a file."
    echo "  -n key_pair_name - A key pair name."
    echo "  -f file_path - File to store the key pair."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:f:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      f) file_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
  fi

  response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
```

```
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  }

  if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
```

```
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte CreateKeyPairem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
  \param keyPairName: A name for a key pair.
  \param keyFilePath: File path where the credentials are stored. Ignored if it
 is an empty string;
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createKeyPair(const Aws::String &keyPairName, const Aws::String
 &keyFilePath,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(keyPairName);

    Aws::EC2::Model::CreateKeyPairOutcome outcome =
 ec2Client.CreateKeyPair(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create key pair - "  << keyPairName << ". " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully created key pair named " <<
                  keyPairName << std::endl;
        if (!keyFilePath.empty()) {
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
                      keyFilePath << std::endl;
        }

    }

    return outcome.IsSuccess();

}
```

- Para obter detalhes da API, consulte CreateKeyPair a Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar um par de chaves

Este exemplo cria um par de chaves denominado `MyKeyPair`.

Comando:

```
aws ec2 create-key-pair --key-name MyKeyPair
```

A saída é uma versão ASCII da chave privada e da impressão digital da chave. Você precisa salvar a chave em um arquivo.

Para obter mais informações, consulte Using Key Pairs no Guia do usuário da AWS Command Line Interface.

- Para obter detalhes da API, consulte CreateKeyPairem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new key pair asynchronously.
    *
    * @param keyName the name of the key pair to create
    * @param fileName the name of the file to write the key material to
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of creating the key pair and writing the key material to a file
    */
   public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
       CreateKeyPairRequest request = CreateKeyPairRequest.builder()
           .keyName(keyName)
           .build();

       CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
       responseFuture.whenComplete((response, exception) -> {
           if (response != null) {
               try {
                   BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                   writer.write(response.keyMaterial());
                   writer.close();
```

```
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
 file: " + e.getMessage(), e);
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
 exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- Para obter detalhes da API, consulte CreateKeyPaira Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates an ED25519 or 2048-bit RSA key pair with the specified name and in the
 specified PEM or PPK format.
 * Amazon EC2 stores the public key and displays the private key for you to save
 to a file.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new CreateKeyPairCommand({
    KeyName: keyName,
  });
```

```
  try {
    const { KeyMaterial, KeyName } = await client.send(command);
    console.log(KeyName);
    console.log(KeyMaterial);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidKeyPair.Duplicate") {
      console.warn(`${caught.message}. Try another key name.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateKeyPaira Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- Para obter detalhes da API, consulte a CreateKeyPairreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria um par de chaves e captura a chave privada PEM-encoded RSA em um arquivo com o nome especificado. Quando você estiver usando PowerShell, a codificação deve ser definida como ascii para gerar uma chave válida. Para obter mais informações, consulte Criar, exibir e excluir pares de chaves do Amazon EC2 (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) no Guia do usuário da interface de linha de AWS comando.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- Para obter detalhes da API, consulte CreateKeyPairem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um par de chaves e captura a chave privada PEM-encoded RSA em um arquivo com o nome especificado. Quando você estiver usando PowerShell, a codificação deve ser definida como ascii para gerar uma chave válida. Para obter mais informações, consulte Criar, exibir e excluir pares de chaves do Amazon EC2 (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) no Guia do usuário da interface de linha de AWS comando.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- Para obter detalhes da API, consulte CreateKeyPairem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir
```

```
    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2
instance.
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that
represents the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
                )
            raise
        else:
```

```
            return self.key_pair
```

- Para obter detalhes da API, consulte a CreateKeyPairReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
```

```
  puts "Private key file saved locally as '#{filename}'."
  true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  false
end

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts 'No key pairs found.'
  else
    puts 'Key pair names:'
    result.key_pairs.each do |key_pair|
      puts key_pair.key_name
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
```

```
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end

# Example usage:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'Displaying existing key pair names before creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Creating key pair...'
  unless key_pair_created?(ec2_client, key_pair_name)
    puts 'Stopping program.'
    exit 1
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after creating this key pair...'
  describe_key_pairs(ec2_client)
```

```
  puts '-' * 10
  puts 'Deleting key pair...'
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts 'Stopping program. You must delete the key pair yourself.'
    exit 1
  end
  puts 'Key pair deleted.'

  puts '-' * 10
  puts 'Now that the key pair is deleted, ' \
    'also deleting the related private key pair file...'
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.delete(filename)
  if File.exist?(filename)
    puts "Could not delete file at '#{filename}'. You must delete it yourself."
  else
    puts 'File deleted.'
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after deleting this key pair...'
  describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateKeyPaira Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Implementação do Rust que chama o create_key_pair do cliente do EC2 e extrai o material retornado.

```
   pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
String), EC2Error> {
       tracing::info!("Creating key pair {name}");
       let output = self.client.create_key_pair().key_name(name).send().await?;
       let info = KeyPairInfo::builder()
           .set_key_name(output.key_name)
           .set_key_fingerprint(output.key_fingerprint)
           .set_key_pair_id(output.key_pair_id)
           .build();
       let material = output
           .key_material
           .ok_or_else(|| EC2Error::new("Create Key Pair has no key
material"))?;
       Ok((info, material))
   }
```

Uma função que chama create_key impl e salva com segurança a chave privada do PEM.

```
   /// Creates a key pair that can be used to securely connect to an EC2
instance.
   /// The returned key pair contains private key information that cannot be
retrieved
   /// again. The private key data is stored as a .pem file.
   ///
   /// :param key_name: The name of the key pair to create.
   pub async fn create(
       &mut self,
       ec2: &EC2,
       util: &Util,
       key_name: String,
   ) -> Result<KeyPairInfo, EC2Error> {
       let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
           self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
           e.add_message(format!("Couldn't create key {key_name}"))
       })?;

       let path = self.key_file_dir.join(format!("{key_name}.pem"));
```

```
        // Save the key_pair information immediately, so it can get cleaned up if
 write_secure fails.
        self.key_file_path = Some(path.clone());
        self.key_pair = key_pair.clone();

        util.write_secure(&key_name, &path, material)?;

        Ok(key_pair)
    }
```

- Para obter detalhes da API, consulte a CreateKeyPairreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).
                 " oo_result is returned for testing purposes. "
        MESSAGE 'Amazon EC2 key pair created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateKeyPairreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Create a new RSA key pair and save the private key to a randomly-named
    /// file in the temporary directory.
    ///
    /// - Parameter name: The name of the key pair to create.
    ///
    /// - Returns: The URL of the newly created `.pem` file or `nil` if unable
    ///   to create the key pair.
    func createKeyPair(name: String) async -> URL? {
        do {
            let output = try await ec2Client.createKeyPair(
                input: CreateKeyPairInput(
                    keyName: name
                )
            )

            guard let keyMaterial = output.keyMaterial else {
                return nil
            }

            // Build the URL of the temporary private key file.

            let fileURL = URL.temporaryDirectory
                                .appendingPathComponent(name)
                                .appendingPathExtension("pem")

            do {
                try keyMaterial.write(to: fileURL, atomically: true, encoding:
 String.Encoding.utf8)
                return fileURL
            } catch {
```

```
                print("*** Failed to write the private key.")
                return nil
            }
        } catch {
            print("*** Unable to create the key pair.")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte CreateKeyPaira referência da API AWS SDK for Swift.

Use **`CreateLaunchTemplate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateLaunchTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Rede privada virtual com servidores privados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
   /// The launch template specifies a Bash script in its user data field that
runs after
   /// the instance is started. This script installs the Python packages and
starts a Python
   /// web server on the instance.
```

```
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
```

```
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
 "InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
 {_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
 {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateLaunchTemplatea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: criar um modelo de lançamento

O exemplo `create-launch-template` a seguir cria um modelo de execução que especifica a sub-rede na qual executar a instância, atribui um endereço IP público e um endereço IPv6 à instância e cria uma tag para a instância.

```
aws ec2 create-launch-template \
    --launch-template-name TemplateForWebServer \
    --version-description WebVersion1 \
    --launch-template-data '{"NetworkInterfaces":
[{"AssociatePublicIpAddress":true,"DeviceIndex":0,"Ipv6AddressCount":1,"SubnetId":"subnet
[{"ResourceType":"instance","Tags":[{"Key":"purpose","Value":"webserver"}]}]}'
```

Saída:

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-01238c059e3466abc",
        "LaunchTemplateName": "TemplateForWebServer",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2019-01-27T09:13:24.000Z"
    }
}
```

Para obter mais informações, consulte Execução de uma instância em um modelo de execução no Guia do usuário do Amazon Elastic Compute Cloud. Para obter informações sobre JSON-formatted parâmetros de cotação, consulte Citar cadeias de caracteres no Guia do usuário da interface de linha de AWS comando.

Exemplo 2: para criar um modelo de execução para o Amazon EC2 Auto Scaling

O exemplo `create-launch-template` a seguir cria um modelo de execução com várias tags e um mapeamento de dispositivos de blocos para especificar um volume adicional do EBS quando uma instância é executada. Especifique um valor para `Groups` que corresponda aos grupos de segurança da VPC na qual o seu grupo do Auto Scaling executará as instâncias. Especifique a VPC e as sub-redes como propriedades do grupo do Auto Scaling.

```
aws ec2 create-launch-template \
    --launch-template-name TemplateForAutoScaling \
    --version-description AutoScalingVersion1 \
    --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
["sg-7c227019,sg-903004f8"],"DeleteOnTermination":true}],"ImageId":"ami-
b42209de","InstanceType":"m4.large","TagSpecifications":
[{"ResourceType":"instance","Tags":[{"Key":"environment","Value":"production"},
{"Key":"purpose","Value":"webserver"}]},{"ResourceType":"volume","Tags":
[{"Key":"environment","Value":"production"},{"Key":"cost-
center","Value":"cc123"}]}],"BlockDeviceMappings":[{"DeviceName":"/dev/
sda1","Ebs":{"VolumeSize":100}}]}' --region us-east-1
```

Saída:

```
{
```

```
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-0123c79c33a54e0abc",
        "LaunchTemplateName": "TemplateForAutoScaling",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2019-04-30T18:16:06.000Z"
    }
}
```

Para obter mais informações, consulte Como criar um modelo de execução para um grupo do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling. Para obter informações sobre JSON-formatted parâmetros de cotação, consulte Citar cadeias de caracteres no Guia do usuário da interface de linha de AWS comando.

Exemplo 3: criar um modelo de execução que especifique a criptografia dos volumes do EBS

O exemplo `create-launch-template` a seguir cria um modelo de execução que inclui volumes criptografados do EBS criados de um snapshot não criptografado. Ele também coloca tags nos volumes durante a criação. Se a criptografia por padrão estiver desabilitada, você deve especificar a opção `"Encrypted"` conforme mostrado no exemplo a seguir. Se você usar a opção `"KmsKeyId"` para especificar uma CMK gerenciada pelo cliente, também deverá especificar a opção `"Encrypted"` mesmo que a criptografia por padrão esteja habilitada.

```
aws ec2 create-launch-template \
  --launch-template-name TemplateForEncryption \
  --launch-template-data file://config.json
```

Conteúdo de `config.json`:

```
{
    "BlockDeviceMappings":[
        {
            "DeviceName":"/dev/sda1",
            "Ebs":{
                "VolumeType":"gp2",
                "DeleteOnTermination":true,
                "SnapshotId":"snap-066877671789bd71b",
                "Encrypted":true,
```

```
                "KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/abcd1234-
a123-456a-a12b-a123b4cd56ef"
            }
        }
    ],
    "ImageId":"ami-00068cd7555f543d5",
    "InstanceType":"c5.large",
    "TagSpecifications":[
        {
            "ResourceType":"volume",
            "Tags":[
                {
                    "Key":"encrypted",
                    "Value":"yes"
                }
            ]
        }
    ]
}
```

Saída:

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-0d5bd51bcf8530abc",
        "LaunchTemplateName": "TemplateForEncryption",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2020-01-07T19:08:36.000Z"
    }
}
```

Para obter mais informações, consulte Restoring an Amazon EBS Volume from a Snapshot and Encryption by Default no Guia do usuário do Amazon Elastic Compute Cloud.

- Para obter detalhes da API, consulte CreateLaunchTemplateem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const ssmClient = new SSMClient({});
const { Parameter } = await ssmClient.send(
  new GetParameterCommand({
    Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
  }),
);
const ec2Client = new EC2Client({});
await ec2Client.send(
  new CreateLaunchTemplateCommand({
    LaunchTemplateName: NAMES.launchTemplateName,
    LaunchTemplateData: {
      InstanceType: "t3.micro",
      ImageId: Parameter.Value,
      IamInstanceProfile: { Name: NAMES.instanceProfileName },
      UserData: readFileSync(
        join(RESOURCES_PATH, "server_startup_script.sh"),
      ).toString("base64"),
      KeyName: NAMES.keyPairName,
    },
  }),
```

- Para obter detalhes da API, consulte CreateLaunchTemplatea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo cria um modelo de execução que inclui um perfil de instância que concede permissões específicas à instância e um script Bash de dados do usuário que é executado na instância após sua inicialização.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
```

```
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
```

```
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
```

```
                    f"Launch template {self.launch_template_name} already exists,
 nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template
```

- Para obter detalhes da API, consulte a CreateLaunchTemplateReferência da API AWS SDK for Python (Boto3).

Usar **`CreateNetworkAcl`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateNetworkAcl`.

CLI

AWS CLI

Criar uma ACL de rede

Este exemplo cria uma rede ACL para a VPC especificada.

Comando:

```
aws ec2 create-network-acl --vpc-id vpc-a01106c2
```

Saída:

```
{
    "NetworkAcl": {
        "Associations": [],
        "NetworkAclId": "acl-5fb85d36",
        "VpcId": "vpc-a01106c2",
        "Tags": [],
        "Entries": [
            {
                "CidrBlock": "0.0.0.0/0",
                "RuleNumber": 32767,
                "Protocol": "-1",
                "Egress": true,
                "RuleAction": "deny"
```

```
            },
            {
                "CidrBlock": "0.0.0.0/0",
                "RuleNumber": 32767,
                "Protocol": "-1",
                "Egress": false,
                "RuleAction": "deny"
            }
        ],
        "IsDefault": false
    }
}
```

- Para obter detalhes da API, consulte CreateNetworkAclem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma ACL de rede para a VPC especificada.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Saída:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- Para obter detalhes da API, consulte CreateNetworkAclem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma ACL de rede para a VPC especificada.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Saída:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- Para obter detalhes da API, consulte CreateNetworkAclem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateNetworkAclEntry`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateNetworkAclEntry`.

CLI

AWS CLI

Criar uma entrada de ACL de rede

Este exemplo cria uma entrada para a rede ACL especificada. A regra permite o tráfego de entrada de qualquer endereço IPv4 (0.0.0). 0/0) na porta UDP 53 (DNS) em qualquer sub-rede associada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 create-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 100 --protocol udp --port-range From=53,To=53 --cidr-block 0.0.0.0/0 --
rule-action allow
```

Este exemplo cria uma regra para a rede ACL especificada que permite o tráfego de entrada de qualquer endereço IPv6 (:: /0) na porta TCP 80 (HTTP).

Comando:

```
aws ec2 create-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 120 --protocol tcp --port-range From=80,To=80 --ipv6-cidr-block ::/0 --
rule-action allow
```

- Para obter detalhes da API, consulte CreateNetworkAclEntryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma entrada para a ACL de rede especificada. A regra permite tráfego de entrada de qualquer lugar (0.0.0). 0/0) na porta UDP 53 (DNS) em qualquer sub-rede associada.

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -
RuleAction allow
```

- Para obter detalhes da API, consulte CreateNetworkAclEntryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma entrada para a ACL de rede especificada. A regra permite tráfego de entrada de qualquer lugar (0.0.0). 0/0) na porta UDP 53 (DNS) em qualquer sub-rede associada.

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -
RuleAction allow
```

- Para obter detalhes da API, consulte CreateNetworkAclEntryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateNetworkInterface`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateNetworkInterface`.

CLI

AWS CLI

Exemplo 1: especificar um endereço IPv4 para uma interface de rede

O exemplo `create-network-interface` a seguir cria uma interface de rede para a sub-rede especificada com o endereço IPv4 primário especificado.

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my network interface" \
    --groups sg-09dfba7ed20cda78b \
    --private-ip-address 10.0.8.17
```

Saída:

```
{
    "NetworkInterface": {
        "AvailabilityZone": "us-west-2a",
        "Description": "my network interface",
        "Groups": [
            {
                "GroupName": "my-security-group",
                "GroupId": "sg-09dfba7ed20cda78b"
            }
        ],
        "InterfaceType": "interface",
        "Ipv6Addresses": [],
        "MacAddress": "06:6a:0f:9a:49:37",
        "NetworkInterfaceId": "eni-0492b355f0cf3b3f8",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.17",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateDnsName": "ip-10-0-8-17.us-west-2.compute.internal",
                "PrivateIpAddress": "10.0.8.17"
            }
        ],
        "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-00a24d0d67acf6333",
        "TagSet": [],
        "VpcId": "vpc-02723a0feeeb9d57b"
    }
```

```
}
```

Exemplo 2: criar uma interface de rede com um endereço IPv4 e um endereço IPv6

O exemplo `create-network-interface` a seguir cria uma interface de rede para a sub-rede especificada com um endereço IPv4 e um endereço IPv6 selecionados pelo Amazon EC2.

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my dual stack network interface" \
    --ipv6-address-count 1 \
    --groups sg-09dfba7ed20cda78b
```

Saída:

```
{
    "NetworkInterface": {
        "AvailabilityZone": "us-west-2a",
        "Description": "my dual stack network interface",
        "Groups": [
            {
                "GroupName": "my-security-group",
                "GroupId": "sg-09dfba7ed20cda78b"
            }
        ],
        "InterfaceType": "interface",
        "Ipv6Addresses": [
            {
                "Ipv6Address": "2600:1f13:cfe:3650:a1dc:237c:393a:4ba7",
                "IsPrimaryIpv6": false
            }
        ],
        "MacAddress": "06:b8:68:d2:b2:2d",
        "NetworkInterfaceId": "eni-05da417453f9a84bf",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.18",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
                "PrivateIpAddress": "10.0.8.18"
```

```
            }
        ],
        "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-00a24d0d67acf6333",
        "TagSet": [],
        "VpcId": "vpc-02723a0feeeb9d57b",
        "Ipv6Address": "2600:1f13:cfe:3650:a1dc:237c:393a:4ba7"
    }
}
```

Exemplo 3: criar uma interface de rede com opções de configuração de rastreamento de conexão

O exemplo `create-network-interface` a seguir cria uma interface de rede e configura os tempos limite de rastreamento de conexão ociosa.

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --groups sg-02e57dbcfe0331c1b \
    --connection-tracking-specification TcpEstablishedTimeout=86400,UdpTimeout=60
```

Saída:

```
{
    "NetworkInterface": {
        "AvailabilityZone": "us-west-2a",
        "ConnectionTrackingConfiguration": {
            "TcpEstablishedTimeout": 86400,
            "UdpTimeout": 60
        },
        "Description": "",
        "Groups": [
            {
                "GroupName": "my-security-group",
                "GroupId": "sg-02e57dbcfe0331c1b"
            }
        ],
        "InterfaceType": "interface",
        "Ipv6Addresses": [],
        "MacAddress": "06:4c:53:de:6d:91",
```

```
        "NetworkInterfaceId": "eni-0c133586e08903d0b",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.94",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",
                "PrivateIpAddress": "10.0.8.94"
            }
        ],
        "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-00a24d0d67acf6333",
        "TagSet": [],
        "VpcId": "vpc-02723a0feeeb9d57b"
    }
}
```

Exemplo 4: criar um adaptador Fabric elástico

O exemplo `create-network-interface` a seguir cria um EFA:

```
aws ec2 create-network-interface \
    --interface-type efa \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my efa" \
    --groups sg-02e57dbcfe0331c1b
```

Saída:

```
{
    "NetworkInterface": {
        "AvailabilityZone": "us-west-2a",
        "Description": "my efa",
        "Groups": [
            {
                "GroupName": "my-efa-sg",
                "GroupId": "sg-02e57dbcfe0331c1b"
            }
        ],
```

```
        "InterfaceType": "efa",
        "Ipv6Addresses": [],
        "MacAddress": "06:d7:a4:f7:4d:57",
        "NetworkInterfaceId": "eni-034acc2885e862b65",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.180",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
                "PrivateIpAddress": "10.0.8.180"
            }
        ],
        "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-00a24d0d67acf6333",
        "TagSet": [],
        "VpcId": "vpc-02723a0feeeb9d57b"
    }
}
```

Para obter mais informações, consulte Interfaces de rede elástica no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte CreateNetworkInterfaceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria a interface de rede especificada.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Saída:

```
Association        :
```

```
Attachment         :
AvailabilityZone   : us-west-2c
Description        : my network interface
Groups             : {my-security-group}
MacAddress         : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.17
PrivateIpAddresses : {}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : pending
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

- Para obter detalhes da API, consulte CreateNetworkInterfaceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria a interface de rede especificada.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Saída:

```
Association        :
Attachment         :
AvailabilityZone   : us-west-2c
Description        : my network interface
Groups             : {my-security-group}
MacAddress         : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.17
PrivateIpAddresses : {}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
```

```
Status              : pending
SubnetId            : subnet-1a2b3c4d
TagSet              : {}
VpcId               : vpc-12345678
```

- Para obter detalhes da API, consulte CreateNetworkInterfaceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreatePlacementGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreatePlacementGroup`.

CLI

AWS CLI

Criar um grupo de posicionamento

Esse exemplo de comando cria um grupo de posicionamento com o nome especificado.

Comando:

```
aws ec2 create-placement-group --group-name my-cluster --strategy cluster
```

Criar um grupo com posicionamento em partições

Esse exemplo de comando cria um grupo com posicionamento em partições chamado `HDFS-Group-A` com cinco partições.

Comando:

```
aws ec2 create-placement-group --group-name HDFS-Group-A --strategy partition --
partition-count 5
```

- Para obter detalhes da API, consulte CreatePlacementGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um grupo de posicionamento com o nome especificado.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- Para obter detalhes da API, consulte CreatePlacementGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um grupo de posicionamento com o nome especificado.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- Para obter detalhes da API, consulte CreatePlacementGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateRoute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateRoute`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Introdução aos gateways de trânsito de rede
- Rede privada virtual com servidores privados
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Criar uma rota

Este exemplo cria uma rota para a tabela de rotas especificada. A rota faz a correspondência de todo o tráfego IPv4 (`0.0.0.0/0`) e o encaminha para o gateway da internet especificado. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 create-route --route-table-id rtb-22574640 --destination-cidr-
block 0.0.0.0/0 --gateway-id igw-c0a643a9
```

Este exemplo de comando cria uma rota na tabela de rotas rtb-g8ff4ea2. A rota corresponde ao tráfego do bloco CIDR IPv4 10.0.0. 0/16 e o encaminha para a conexão de emparelhamento VPC, pcx-111aaa22. Essa rota permite que o tráfego seja direcionado para a VPC de mesmo nível na conexão de emparelhamento da VPC. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 create-route --route-table-id rtb-g8ff4ea2 --destination-cidr-
block 10.0.0.0/16 --vpc-peering-connection-id pcx-1a2b3c4d
```

Este exemplo cria uma rota na tabela de rotas especificada que corresponde a todo tráfego IPv6 (`::/0`) e a roteia para o Gateway da Internet apenas de saída especificado.

Comando:

```
aws ec2 create-route --route-table-id rtb-dce620b8 --destination-ipv6-cidr-
block ::/0 --egress-only-internet-gateway-id eigw-01eadbd45ecd7943f
```

- Para obter detalhes da API, consulte CreateRouteem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria a rota especificada para a tabela de rotas especificada. A rota faz a correspondência de todo o tráfego e o encaminha para o gateway da internet especificado.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -
GatewayId igw-1a2b3c4d
```

Saída:

```
True
```

- Para obter detalhes da API, consulte CreateRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria a rota especificada para a tabela de rotas especificada. A rota faz a correspondência de todo o tráfego e o encaminha para o gateway da internet especificado.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -
GatewayId igw-1a2b3c4d
```

Saída:

```
True
```

- Para obter detalhes da API, consulte CreateRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateRouteTable`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateRouteTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com servidores privados
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Para criar uma tabela de rotas

Este exemplo cria uma tabela de rotas para a VPC especificada.

Comando:

```
aws ec2 create-route-table --vpc-id vpc-a01106c2
```

Saída:

```
{
    "RouteTable": {
        "Associations": [],
        "RouteTableId": "rtb-22574640",
        "VpcId": "vpc-a01106c2",
        "PropagatingVgws": [],
        "Tags": [],
        "Routes": [
            {
                "GatewayId": "local",
                "DestinationCidrBlock": "10.0.0.0/16",
                "State": "active"
            }
        ]
    }
}
```

- Para obter detalhes da API, consulte CreateRouteTableem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma tabela de rotas para a VPC especificada.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Saída:

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- Para obter detalhes da API, consulte CreateRouteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma tabela de rotas para a VPC especificada.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Saída:

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- Para obter detalhes da API, consulte CreateRouteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
```

```
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts 'Added tags to route table.'
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts 'Created route with destination CIDR block ' \
```

```
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
      'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
      'TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
      'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
      "'0.0.0.0/0' my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-0b6f769731EXAMPLE'
    subnet_id = 'subnet-03d9303b57EXAMPLE'
    gateway_id = 'igw-06ca90c011EXAMPLE'
    destination_cidr_block = '0.0.0.0/0'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
```

```
    subnet_id = ARGV[1]
    gateway_id = ARGV[2]
    destination_cidr_block = ARGV[3]
    tag_key = ARGV[4]
    tag_value = ARGV[5]
    region = ARGV[6]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if route_table_created_and_associated?(
    ec2_resource,
    vpc_id,
    subnet_id,
    gateway_id,
    destination_cidr_block,
    tag_key,
    tag_value
  )
    puts 'Route table created and associated.'
  else
    puts 'Route table not created or not associated.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateRouteTablea Referência AWS SDK para Ruby da API.

Use **`CreateSecurityGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateSecurityGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Configurar a conectividade do serviço de contêiner
- Criar uma rede privada virtual básica

- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criar uma instância de banco de dados relacional
- Criação e gerenciamento de volumes de armazenamento em bloco
- Conceitos básicos de compras do mercado de software
- Introdução aos bancos de dados de grafos
- Introdução ao balanceamento de carga
- Conceitos básicos de streaming gerenciado
- Introdução às máquinas virtuais
- Rede privada virtual com servidores privados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an Amazon EC2 security group with a specified name and
 description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
    public async Task<string> CreateSecurityGroup(string groupName, string
 groupDescription)
    {
        try
        {
            var response = await _amazonEC2.CreateSecurityGroupAsync(
                new CreateSecurityGroupRequest(groupName, groupDescription));
```

```
            // Wait until the security group exists.
            int retries = 5;
            while (retries-- > 0)
            {
                var groups = await DescribeSecurityGroups(response.GroupId);
                if (groups.Any())
                {
                    return response.GroupId;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
            throw new DoesNotExistException("security group not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
            }
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the security group.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateSecurityGroupa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security
 group.
#
# Parameters:
#       -n security_group_name - The name of the security group.
#       -d security_group_description - The description of the security group.
#
# Returns:
#       The ID of the created security group, or an error message if the
 operation fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_create_security_group() {
  local security_group_name security_group_description response

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -n security_group_name - The name of the security group."
    echo "  -d security_group_description - The description of the security
 group."
    echo ""
  }

  # Parse the command-line arguments
```

```
  while getopts "n:d:h" option; do
    case "${option}" in
      n) security_group_name="${OPTARG}" ;;
      d) security_group_description="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$security_group_name" ]]; then
    errecho "ERROR: You must provide a security group name with the -n
parameter."
    return 1
  fi

  if [[ -z "$security_group_description" ]]; then
    errecho "ERROR: You must provide a security group description with the -d
parameter."
    return 1
  fi

  # Create the security group
  response=$(aws ec2 create-security-group \
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
```

```
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
```

```
  fi

  return 0
}
```

- Para obter detalhes da API, consulte CreateSecurityGroupem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create a security group.
/*!
  \param groupName: A security group name.
  \param description: A description.
  \param vpcID: A virtual private cloud (VPC) ID.
  \param[out] groupIDResult: A string to receive the group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createSecurityGroup(const Aws::String &groupName,
                                      const Aws::String &description,
                                      const Aws::String &vpcID,
                                      Aws::String &groupIDResult,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(groupName);
    request.SetDescription(description);
    request.SetVpcId(vpcID);
```

```
    const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
            ec2Client.CreateSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create security group:" <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    std::cout << "Successfully created security group named " << groupName <<
              std::endl;


    groupIDResult = outcome.GetResult().GetGroupId();

    return true;
}
```

- Para obter detalhes da API, consulte CreateSecurityGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar um grupo de segurança para EC2-Classic

Este exemplo cria um grupo de segurança chamado `MySecurityGroup`.

Comando:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My
 security group"
```

Saída:

```
{
    "GroupId": "sg-903004f8"
}
```

Para criar um grupo de segurança para EC2-VPC

Este exemplo cria um grupo de segurança chamado `MySecurityGroup` para a VPC especificada.

Comando:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My
 security group" --vpc-id vpc-1a2b3c4d
```

Saída:

```
{
    "GroupId": "sg-903004f8"
}
```

Para obter mais informações, consulte Using Security Groups no Guia do usuário da AWS Command Line Interface.

- Para obter detalhes da API, consulte CreateSecurityGroupem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new security group asynchronously with the specified group name,
 description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP
 address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
```

```
    * @param vpcId        the ID of the VPC in which to create the security
group
    * @param myIpAddress  the IP address from which to allow inbound traffic
(e.g., "192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security
group or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
                   .build();

               IpPermission ipPerm = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(80)
                   .fromPort(80)
                   .ipRanges(ipRange)
                   .build();

               IpPermission ipPerm2 = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(22)
                   .fromPort(22)
                   .ipRanges(ipRange)
                   .build();

               AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                   .groupName(groupName)
                   .ipPermissions(ipPerm, ipPerm2)
```

```
                    .build();

                return
 getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
 exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security
 group: " + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateSecurityGroupa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates a security group.
 * @param {{ groupName: string, description: string }} options
 */
export const main = async ({ groupName, description }) => {
  const client = new EC2Client({});
```

```
  const command = new CreateSecurityGroupCommand({
    // Up to 255 characters in length. Cannot start with sg-.
    GroupName: groupName,
    // Up to 255 characters in length.
    Description: description,
  });

  try {
    const { GroupId } = await client.send(command);
    console.log(GroupId);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateSecurityGroupa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
```

```
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "0.0.0.0/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group
 $groupNameVal")
        return resp.groupId
    }
}
```

- Para obter detalhes da API, consulte a CreateSecurityGroupreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um grupo de segurança para a VPC especificada.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group" -VpcId vpc-12345678
```

Saída:

```
sg-12345678
```

Exemplo 2: Este exemplo cria um grupo de segurança para EC2-Classic.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group"
```

Saída:

```
sg-45678901
```

- Para obter detalhes da API, consulte CreateSecurityGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um grupo de segurança para a VPC especificada.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group" -VpcId vpc-12345678
```

Saída:

```
sg-12345678
```

Exemplo 2: Este exemplo cria um grupo de segurança para EC2-Classic.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group"
```

Saída:

```
sg-45678901
```

- Para obter detalhes da API, consulte CreateSecurityGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.
```

```
        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of
the current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to
create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose
a different name."
                )
            raise
        else:
            return self.security_group
```

- Para obter detalhes da API, consulte a CreateSecurityGroupReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(ec2_client, group_name, description, vpc_id)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
```

```
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
```

```
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol
 " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  false
end

# Refactored method to simplify complexity for describing security group
 permissions
def format_port_information(perm)
  from_port_str = perm.from_port == '-1' || perm.from_port == -1 ? 'All' :
 perm.from_port.to_s
  to_port_str = perm.to_port == '-1' || perm.to_port == -1 ? 'All' :
 perm.to_port.to_s
  { from_port: from_port_str, to_port: to_port_str }
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_group_permissions(perm)
  ports = format_port_information(perm)

  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"
  print ", From: #{ports[:from_port]}, To: #{ports[:to_port]}"

  print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}" if perm.key?
(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?

  print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}" if perm.key?(:ip_ranges) &&
 perm.ip_ranges.count.positive?
  print "\n"
end

# Displays information about available security groups in
```

```
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      display_group_details(sg)
    end
  else
    puts 'No security groups found.'
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Helper method to display the details of security groups
def display_group_details(sg)
  puts '-' * (sg.group_name.length + 13)
  puts "Name:        #{sg.group_name}"
  puts "Description: #{sg.description}"
  puts "Group ID:    #{sg.group_id}"
  puts "Owner ID:    #{sg.owner_id}"
  puts "VPC ID:      #{sg.vpc_id}"

  display_group_tags(sg.tags) if sg.tags.count.positive?
  display_group_permissions(sg)
end

def display_group_tags(tags)
  puts 'Tags:'
  tags.each do |tag|
    puts "  Key: #{tag.key}, Value: #{tag.value}"
  end
end

def display_group_permissions(sg)
  if sg.ip_permissions.count.positive?
    puts 'Inbound rules:'
    sg.ip_permissions.each do |p|
      describe_security_group_permissions(p)
    end
  end

  return if sg.ip_permissions_egress.empty?
```

```
  puts 'Outbound rules:'
  sg.ip_permissions_egress.each do |p|
    describe_security_group_permissions(p)
  end
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  false
end

# Example usage with refactored run_me to reduce complexity
def run_me
  group_name, description, vpc_id, ip_protocol_http, from_port_http,
 to_port_http, \
  cidr_ip_range_http, ip_protocol_ssh, from_port_ssh, to_port_ssh, \
  cidr_ip_range_ssh, region = process_arguments
  ec2_client = Aws::EC2::Client.new(region: region)

  security_group_id = attempt_create_security_group(ec2_client, group_name,
 description, vpc_id)
  security_group_exists = security_group_id != 'Error'

  if security_group_exists
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_http,
 from_port_http, to_port_http, cidr_ip_range_http)
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_ssh,
 from_port_ssh, to_port_ssh, cidr_ip_range_ssh)
  end

  describe_security_groups(ec2_client)
  attempt_delete_security_group(ec2_client, security_group_id) if
 security_group_exists
end

def process_arguments
  if ARGV[0] == '--help' || ARGV[0] == '-h'
```

```
    display_help
    exit 1
  elsif ARGV.count.zero?
    default_values
  else
    ARGV
  end
end

def attempt_create_security_group(ec2_client, group_name, description, vpc_id)
  puts 'Attempting to create security group...'
  security_group_id = create_security_group(ec2_client, group_name, description,
 vpc_id)
  puts 'Could not create security group. Skipping this step.' if
 security_group_id == 'Error'
  security_group_id
end

def add_inbound_rules(ec2_client, security_group_id, ip_protocol, from_port,
 to_port, cidr_ip_range)
  puts 'Attempting to add inbound rules to security group...'
  return if security_group_ingress_authorized?(ec2_client, security_group_id,
 ip_protocol, from_port, to_port,
                                               cidr_ip_range)

  puts 'Could not add inbound rule to security group. Skipping this step.'
end

def attempt_delete_security_group(ec2_client, security_group_id)
  puts "\nAttempting to delete security group..."
  return if security_group_deleted?(ec2_client, security_group_id)

  puts 'Could not delete security group. You must delete it yourself.'
end

def display_help
  puts 'Usage:   ruby ec2-ruby-example-security-group.rb ' \
    'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
    'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
    'CIDR_IP_RANGE_2 REGION'
  puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
    "my-security-group 'This is my security group.' vpc-6713dfEX " \
    "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
end
```

```
def default_values
  [
    'my-security-group', 'This is my security group.', 'vpc-6713dfEX', 'tcp',
 '80', '80',
    '0.0.0.0/0', 'tcp', '22', '22', '0.0.0.0/0', 'us-west-2'
  ]
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateSecurityGroupa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn create_security_group(
        &self,
        name: &str,
        description: &str,
    ) -> Result<SecurityGroup, EC2Error> {
        tracing::info!("Creating security group {name}");
        let create_output = self
            .client
            .create_security_group()
            .group_name(name)
            .description(description)
            .send()
            .await
            .map_err(EC2Error::from)?;

        let group_id = create_output
```

```
            .group_id
            .ok_or_else(|| EC2Error::new("Missing security group id after
 creation"))?;

        let group = self
            .describe_security_group(&group_id)
            .await?
            .ok_or_else(|| {
                EC2Error::new(format!("Could not find security group with id
 {group_id}"))
            })?;

        tracing::info!("Created security group {name} as {group_id}");

        Ok(group)
    }
```

- Para obter detalhes da API, consulte a CreateSecurityGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->createsecuritygroup(                 " oo_result is
 returned for testing purposes. "
          iv_description = 'Security group example'
          iv_groupname = iv_security_group_name
          iv_vpcid = iv_vpc_id ).
        MESSAGE 'Security group created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
```

```
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateSecurityGroupreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Create a new security group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the group to create.
    ///   - groupDescription: A description of the new security group.
    ///
    /// - Returns: The ID string of the new security group.
    func createSecurityGroup(name groupName: String, description
 groupDescription: String) async -> String? {
        do {
            let output = try await ec2Client.createSecurityGroup(
                input: CreateSecurityGroupInput(
                    description: groupDescription,
                    groupName: groupName
                )
            )

            return output.groupId
        } catch {
            print("*** Error creating the security group:
 \(error.localizedDescription)")
            return nil
```

```
        }
    }
```

- Para obter detalhes da API, consulte CreateSecurityGroupa referência da API AWS SDK for Swift.

Usar **`CreateSnapshot`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateSnapshot`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

CLI

AWS CLI

Para criar um snapshot

Esse exemplo de comando cria um snapshot do volume com um ID do volume `vol-1234567890abcdef0` e uma breve descrição para identificar o snapshot.

Comando:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description "This is
 my root volume snapshot"
```

Saída:

```
{
    "Description": "This is my root volume snapshot",
    "Tags": [],
    "Encrypted": false,
    "VolumeId": "vol-1234567890abcdef0",
    "State": "pending",
    "VolumeSize": 8,
    "StartTime": "2018-02-28T21:06:01.000Z",
    "Progress": "",
```

```
    "OwnerId": "012345678910",
    "SnapshotId": "snap-066877671789bd71b"
}
```

Para criar um snapshot com tags

Esse exemplo de comando cria um snapshot e aplica duas tags: purpose=prod e costcenter=123.

Comando:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0
 --description 'Prod backup' --tag-specifications
 'ResourceType=snapshot,Tags=[{Key=purpose,Value=prod},
{Key=costcenter,Value=123}]'
```

Saída:

```
{
    "Description": "Prod backup",
    "Tags": [
        {
            "Value": "prod",
            "Key": "purpose"
        },
        {
            "Value": "123",
            "Key": "costcenter"
        }
     ],
     "Encrypted": false,
     "VolumeId": "vol-1234567890abcdef0",
     "State": "pending",
     "VolumeSize": 8,
     "StartTime": "2018-02-28T21:06:06.000Z",
     "Progress": "",
     "OwnerId": "012345678910",
     "SnapshotId": "snap-09ed24a70bc19bbe4"
 }
```

- Para obter detalhes da API, consulte CreateSnapshotem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um snapshot do volume especificado.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Saída:

```
DataEncryptionKeyId :
Description         : This is a test
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            :
SnapshotId          : snap-12345678
StartTime           : 12/22/2015 1:28:42 AM
State               : pending
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 20
```

- Para obter detalhes da API, consulte CreateSnapshotem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um snapshot do volume especificado.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Saída:

```
DataEncryptionKeyId :
Description         : This is a test
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            :
```

```
SnapshotId          : snap-12345678
StartTime           : 12/22/2015 1:28:42 AM
State               : pending
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 20
```

- Para obter detalhes da API, consulte CreateSnapshotem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateSpotDatafeedSubscription`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateSpotDatafeedSubscription`.

CLI

AWS CLI

Criar um feed de dados da instância spot

O exemplo `create-spot-datafeed-subscription` a seguir cria um feed de dados da Instância Spot.

```
aws ec2 create-spot-datafeed-subscription \
    --bucket amzn-s3-demo-bucket \
    --prefix spot-data-feed
```

Saída:

```
{
    "SpotDatafeedSubscription": {
        "Bucket": "amzn-s3-demo-bucket",
        "OwnerId": "123456789012",
        "Prefix": "spot-data-feed",
        "State": "Active"
    }
}
```

O feed de dados é armazenado no bucket do Amazon S3 que você especificar. Os nomes de arquivo desse feed de dados têm o formato a seguir.

```
amzn-s3-demo-bucket.s3.amazonaws.com/spot-data-feed/123456789012.YYYY-MM-DD-
HH.n.abcd1234.gz
```

Consulte mais informações em Feed de dados da instância spot no Manual do usuário do Amazon EC2.

- Para obter detalhes da API, consulte CreateSpotDatafeedSubscriptionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um feed de dados da instância Spot.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Saída:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Para obter detalhes da API, consulte CreateSpotDatafeedSubscriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um feed de dados da instância Spot.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Saída:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Para obter detalhes da API, consulte CreateSpotDatafeedSubscriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateSubnet`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateSubnet`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Introdução aos gateways de trânsito de rede
- Rede privada virtual com servidores privados
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Exemplo 1: criar uma sub-rede somente com um bloco CIDR IPv4

O exemplo `create-subnet` a seguir cria uma sub-rede na VPC especificada com o bloco CIDR IPv4 especificado.

```
aws ec2 create-subnet \
    --vpc-id vpc-081ec835f3EXAMPLE \
    --cidr-block 10.0.0.0/24 \
    --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-only-
subnet}]
```

Saída:

```
{
    "Subnet": {
        "AvailabilityZone": "us-west-2a",
        "AvailabilityZoneId": "usw2-az2",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "10.0.0.0/24",
        "DefaultForAz": false,
```

```
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-0e99b93155EXAMPLE",
        "VpcId": "vpc-081ec835f3EXAMPLE",
        "OwnerId": "123456789012",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-ipv4-only-subnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-0e99b93155EXAMPLE"
    }
}
```

Exemplo 2: para criar uma sub-rede com blocos CIDR IPv4 e IPv6

O exemplo `create-subnet` a seguir cria uma sub-rede na VPC especificada com os blocos CIDR IPv4 e IPv6 especificados.

```
aws ec2 create-subnet \
    --vpc-id vpc-081ec835f3EXAMPLE \
    --cidr-block 10.0.0.0/24 \
    --ipv6-cidr-block 2600:1f16:cfe:3660::/64 \
    --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-ipv6-
subnet}]
```

Saída:

```
{
    "Subnet": {
        "AvailabilityZone": "us-west-2a",
        "AvailabilityZoneId": "usw2-az2",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "10.0.0.0/24",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-0736441d38EXAMPLE",
        "VpcId": "vpc-081ec835f3EXAMPLE",
```

```
        "OwnerId": "123456789012",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [
            {
                "AssociationId": "subnet-cidr-assoc-06c5f904499fcc623",
                "Ipv6CidrBlock": "2600:1f13:cfe:3660::/64",
                "Ipv6CidrBlockState": {
                    "State": "associating"
                }
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-ipv4-ipv6-subnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-0736441d38EXAMPLE"
    }
}
```

Exemplo 3: criar uma sub-rede somente com um bloco CIDR IPv6

O exemplo `create-subnet` a seguir cria uma sub-rede na VPC especificada com o bloco CIDR IPv6 especificado.

```
aws ec2 create-subnet \
    --vpc-id vpc-081ec835f3EXAMPLE \
    --ipv6-native \
    --ipv6-cidr-block 2600:1f16:115:200::/64 \
    --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv6-only-
subnet}]
```

Saída:

```
{
    "Subnet": {
        "AvailabilityZone": "us-west-2a",
        "AvailabilityZoneId": "usw2-az2",
        "AvailableIpAddressCount": 0,
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
```

```
        "State": "available",
        "SubnetId": "subnet-03f720e7deEXAMPLE",
        "VpcId": "vpc-081ec835f3EXAMPLE",
        "OwnerId": "123456789012",
        "AssignIpv6AddressOnCreation": true,
        "Ipv6CidrBlockAssociationSet": [
            {
                "AssociationId": "subnet-cidr-assoc-01ef639edde556709",
                "Ipv6CidrBlock": "2600:1f13:cfe:3660::/64",
                "Ipv6CidrBlockState": {
                    "State": "associating"
                }
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-ipv6-only-subnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-03f720e7deEXAMPLE"
    }
}
```

Para obter mais informações, consulte VPCs e sub-redes no Manual do usuário da Amazon VPC.

- Para obter detalhes da API, consulte CreateSubnetem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma sub-rede com o CIDR especificado.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Saída:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
```

```
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : pending
SubnetId                : subnet-1a2b3c4d
Tag                     : {}
VpcId                   : vpc-12345678
```

- Para obter detalhes da API, consulte CreateSubnetem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma sub-rede com o CIDR especificado.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Saída:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : pending
SubnetId                : subnet-1a2b3c4d
Tag                     : {}
VpcId                   : vpc-12345678
```

- Para obter detalhes da API, consulte CreateSubnetem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
```

```
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

# Example usage:
def run_me
  vpc_id = ''
  cidr_block = ''
  availability_zone = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-subnet.rb ' \
      'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
      'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    cidr_block = '10.0.0.0/24'
    availability_zone = 'us-west-2a'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
```

```
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
    tag_key = ARGV[3]
    tag_value = ARGV[4]
    region = ARGV[5]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if subnet_created_and_tagged?(
    ec2_resource,
    vpc_id,
    cidr_block,
    availability_zone,
    tag_key,
    tag_value
  )
    puts 'Subnet created and tagged.'
  else
    puts 'Subnet not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateSubneta Referência AWS SDK para Ruby da API.

Use **`CreateTags`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateTags`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Add or overwrite only the specified tags for the specified Amazon Elastic
 Compute Cloud (Amazon EC2) resource or resources.
/*!
  \param resources: The resources for the tags.
  \param tags: Vector of tags.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createTags(const Aws::Vector<Aws::String> &resources,
                             const Aws::Vector<Aws::EC2::Model::Tag> &tags,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateTagsRequest createTagsRequest;
    createTagsRequest.SetResources(resources);
    createTagsRequest.SetTags(tags);

    Aws::EC2::Model::CreateTagsOutcome outcome =
 ec2Client.CreateTags(createTagsRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created tags for resources" << std::endl;
    } else {
        std::cerr << "Failed to create tags for resources, " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTagsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: adicionar uma tag a um recurso

O exemplo `create-tags` a seguir adiciona a tag `Stack=production` à imagem especificada ou substitui uma tag existente para a AMI na qual a chave de tag é `Stack`.

```
aws ec2 create-tags \
    --resources ami-1234567890abcdef0 \
    --tags Key=Stack,Value=production
```

Este comando não produz saída

Exemplo 2: adicionar tags a vários recursos

O exemplo `create-tags` a seguir adiciona (ou substitui) duas tags para uma AMI e uma instância. Uma das tags tem uma chave (`webserver`), mas nenhum valor (o valor é definido como uma string vazia). A outra tag tem uma chave (`stack`) e um valor (`Production`).

```
aws ec2 create-tags \
    --resources ami-1a2b3c4d i-1234567890abcdef0 \
    --tags Key=webserver,Value=   Key=stack,Value=Production
```

Este comando não produz saída

Exemplo 3: adicionar tags contendo caracteres especiais

Os exemplos de `create-tags` a seguir adicionam a tag `[Group]=test` a uma instância. Os colchetes ([e]) são caracteres especiais e devem ser recuados. Os exemplos a seguir também usam o caractere de continuação de linha apropriado para cada ambiente.

Se você estiver usando o Windows, coloque o elemento com caracteres especiais entre aspas duplas (“) e preceda cada caractere de aspas duplas com uma barra invertida (\) da maneira a seguir.

```
aws ec2 create-tags ^
    --resources i-1234567890abcdef0 ^
```

```
    --tags Key=\"[Group]\",Value=test
```

Se você estiver usando o Windows PowerShell, coloque o elemento no valor que tem caracteres especiais com aspas duplas (“), preceda cada caractere de aspas duplas com uma barra invertida (\) e, em seguida, coloque toda a estrutura de chave e valor entre aspas simples (') da seguinte maneira.

```
aws ec2 create-tags `
    --resources i-1234567890abcdef0 `
    --tags 'Key=\"[Group]\",Value=test'
```

Se você estiver usando Linux ou OS X, coloque o elemento com caracteres especiais entre aspas duplas (“) e toda a estrutura de chave e valor entre aspas simples (‘) da maneira a seguir.

```
aws ec2 create-tags \
    --resources i-1234567890abcdef0 \
    --tags 'Key="[Group]",Value=test'
```

Para ter mais informações, consulte Marcar com tag os recursos do Amazon EC2 no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte CreateTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona uma única tag ao recurso especificado. A chave da tag é 'myTag' e o valor da tag é TagValue 'meu'. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Exemplo 2: esse exemplo atualiza ou adiciona as tags especificadas ao recurso especificado. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Para obter detalhes da API, consulte CreateTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona uma única tag ao recurso especificado. A chave da tag é 'myTag' e o valor da tag é TagValue 'meu'. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Exemplo 2: esse exemplo atualiza ou adiciona as tags especificadas ao recurso especificado. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Para obter detalhes da API, consulte CreateTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo aplica a tag Name depois de criar uma instância.

```
    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
```

```
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }
```

- Para obter detalhes da API, consulte a CreateTagsreferência da API AWS SDK for Rust.

Usar **`CreateVolume`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateVolume`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criação e gerenciamento de volumes de armazenamento em bloco
- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

CLI

AWS CLI

Criar um volume SSD de uso geral (gp2) vazio

O exemplo `create-volume` a seguir cria um volume SSD de uso geral (gp2) de 80 GiB na zona de disponibilidade especificada. Observe que a região atual deve ser `us-east-1`, ou você pode adicionar o parâmetro `--region` para especificar a região para o comando.

```
aws ec2 create-volume \
    --volume-type gp2 \
    --size 80 \
    --availability-zone us-east-1a
```

Saída:

```
{
    "AvailabilityZone": "us-east-1a",
    "Tags": [],
    "Encrypted": false,
    "VolumeType": "gp2",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "creating",
    "Iops": 240,
    "SnapshotId": "",
    "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "Size": 80
}
```

Se você não especificar um tipo de volume, o tipo de volume padrão será gp2.

```
aws ec2 create-volume \
    --size 80 \
    --availability-zone us-east-1a
```

Exemplo 2: criar um volume SSD (io1) de IOPS provisionadas de um snapshot

O exemplo `create-volume` a seguir cria um volume SSD de IOPS provisionadas (io1) com 1.000 IOPS provisionadas na Zona de Disponibilidade especificada usando o snapshot especificado.

```
aws ec2 create-volume \
    --volume-type io1 \
    --iops 1000 \
    --snapshot-id snap-066877671789bd71b \
    --availability-zone us-east-1a
```

Saída:

```
{
    "AvailabilityZone": "us-east-1a",
    "Tags": [],
    "Encrypted": false,
    "VolumeType": "io1",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "creating",
    "Iops": 1000,
    "SnapshotId": "snap-066877671789bd71b",
    "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "Size": 500
}
```

Exemplo 3: criar um volume criptografado

O exemplo `create-volume` a seguir cria um volume criptografado usando a CMK padrão para criptografia do EBS. Se a criptografia por padrão estiver desabilitada, você deve especificar o parâmetro `--encrypted` da seguinte maneira.

```
aws ec2 create-volume \
    --size 80 \
    --encrypted \
    --availability-zone us-east-1a
```

Saída:

```
{
    "AvailabilityZone": "us-east-1a",
    "Tags": [],
    "Encrypted": true,
    "VolumeType": "gp2",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "creating",
```

```
    "Iops": 240,
    "SnapshotId": "",
    "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "Size": 80
}
```

Se a criptografia estiver habilitada por padrão, o comando de exemplo a seguir criará um volume criptografado, mesmo sem o parâmetro `--encrypted`.

```
aws ec2 create-volume \
    --size 80 \
    --availability-zone us-east-1a
```

Se você usar o parâmetro `--kms-key-id` para especificar uma CMK gerenciada pelo cliente, você deverá especificar o parâmetro `--encrypted` mesmo que a criptografia por padrão esteja habilitada.

```
aws ec2 create-volume \
    --volume-type gp2 \
    --size 80 \
    --encrypted \
    --kms-key-id 0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE \
    --availability-zone us-east-1a
```

Exemplo 4: criar um volume com tags

O exemplo `create-volume` a seguir cria um volume e adiciona duas tags.

```
aws ec2 create-volume \
    --availability-zone us-east-1a \
    --volume-type gp2 \
    --size 80 \
    --tag-specifications
 'ResourceType=volume,Tags=[{Key=purpose,Value=production},{Key=cost-
center,Value=cc123}]'
```

- Para obter detalhes da API, consulte CreateVolumeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria o volume especificado.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2a
CreateTime       : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size             : 50
SnapshotId       :
State            : creating
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
```

Exemplo 2: esse exemplo de solicitação cria um volume e aplica uma tag com uma chave de pilha e um valor de produção.

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- Para obter detalhes da API, consulte CreateVolumeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria o volume especificado.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2a
CreateTime       : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size             : 50
SnapshotId       :
State            : creating
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
```

Exemplo 2: esse exemplo de solicitação cria um volume e aplica uma tag com uma chave de pilha e um valor de produção.

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- Para obter detalhes da API, consulte CreateVolumeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateVpc`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateVpc`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Introdução aos gateways de trânsito de rede
- Rede privada virtual com gerenciamento de endereços IP

- Rede privada virtual com servidores privados
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Exemplo 1: criar uma VPC

O exemplo `create-vpc` a seguir cria uma VPC com o bloco CIDR IPv4 especificado e uma tag de nome.

```
aws ec2 create-vpc \
    --cidr-block 10.0.0.0/16 \
    --tag-specifications ResourceType=vpc,Tags=[{Key=Name,Value=MyVpc}]
```

Saída:

```
{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-5EXAMPLE",
        "State": "pending",
        "VpcId": "vpc-0a60eb65b4EXAMPLE",
        "OwnerId": "123456789012",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-07501b79ecEXAMPLE",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Name",
                "Value": MyVpc"
            }
```

```
        ]
    }
}
```

Exemplo 2: para criar uma VPC com locação dedicada

O exemplo `create-vpc` a seguir cria uma VPC com o bloco CIDR IPv4 especificado e locação dedicada.

```
aws ec2 create-vpc \
    --cidr-block 10.0.0.0/16 \
    --instance-tenancy dedicated
```

Saída:

```
{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-19edf471",
        "State": "pending",
        "VpcId": "vpc-0a53287fa4EXAMPLE",
        "OwnerId": "111122223333",
        "InstanceTenancy": "dedicated",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-00b24cc1c2EXAMPLE",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false
    }
}
```

Exemplo 3: criar uma VPC com um bloco CIDR IPv6

O `create-vpc` exemplo a seguir cria uma VPC com um bloco CIDR Amazon-provided IPv6.

```
aws ec2 create-vpc \
```

```
    --cidr-block 10.0.0.0/16 \
    --amazon-provided-ipv6-cidr-block
```

Saída:

```
{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-dEXAMPLE",
        "State": "pending",
        "VpcId": "vpc-0fc5e3406bEXAMPLE",
        "OwnerId": "123456789012",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-068432c60bEXAMPLE",
                "Ipv6CidrBlock": "",
                "Ipv6CidrBlockState": {
                    "State": "associating"
                },
                "Ipv6Pool": "Amazon",
                "NetworkBorderGroup": "us-west-2"
            }
        ],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0669f8f9f5EXAMPLE",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false
    }
}
```

Exemplo 4: criar uma VPC com um CIDR de um grupo do IPAM

O exemplo `create-vpc` a seguir cria uma VPC com um CIDR de um conjunto do gerenciador de endereços IP (IPAM) da Amazon VPC.

Linux e macOS:

```
aws ec2 create-vpc \
    --ipv4-ipam-pool-id ipam-pool-0533048da7d823723 \
    --tag-specifications
 ResourceType=vpc,Tags='[{Key=Environment,Value="Preprod"},
{Key=Owner,Value="Build Team"}]'
```

Windows:

```
aws ec2 create-vpc ^
    --ipv4-ipam-pool-id ipam-pool-0533048da7d823723 ^
    --tag-specifications
 ResourceType=vpc,Tags=[{Key=Environment,Value="Preprod"},{Key=Owner,Value="Build
 Team"}]
```

Saída:

```
{
    "Vpc": {
        "CidrBlock": "10.0.1.0/24",
        "DhcpOptionsId": "dopt-2afccf50",
        "State": "pending",
        "VpcId": "vpc-010e1791024eb0af9",
        "OwnerId": "123456789012",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0a77de1d803226d4b",
                "CidrBlock": "10.0.1.0/24",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Environment",
                "Value": "Preprod"
            },
            {
                "Key": "Owner",
```

```
                "Value": "Build Team"
            }
        ]
    }
}
```

Para obter mais informações, consulte Criar uma VPC que usa um CIDR de um conjunto do IPAM no Guia do usuário do IPAM da Amazon VPC.

- Para obter detalhes da API, consulte CreateVpcem Referência de AWS CLI Comandos.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param string $cidr
     * @return array
     */
    public function createVpc(string $cidr): array
    {
        try {
            $result = $this->ec2Client->createVpc([
                "CidrBlock" => $cidr,
            ]);
            return $result['Vpc'];
        }catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte CreateVpca Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma VPC com o CIDR especificado. A Amazon VPC também cria o seguinte para a VPC: um conjunto de opções DHCP padrão, uma tabela de rotas principal e uma ACL de rede padrão.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Saída:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- Para obter detalhes da API, consulte CreateVpcem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma VPC com o CIDR especificado. A Amazon VPC também cria o seguinte para a VPC: um conjunto de opções DHCP padrão, uma tabela de rotas principal e uma ACL de rede padrão.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Saída:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
```

```
VpcId           : vpc-12345678
```

- Para obter detalhes da API, consulte CreateVpcem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def create(self, cidr_block: str) -> str:
        """
        Creates a new Amazon VPC with the specified CIDR block.

        :param cidr_block: The CIDR block for the new VPC, such as '10.0.0.0/16'.
        :return: The ID of the new VPC.
        """
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr_block)
            vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[vpc_id])
            return vpc_id
        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateVpcReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
```

```
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
```

```
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-vpc.rb ' \
      'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = '10.0.0.0/24'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,
    cidr_block,
    tag_key,
    tag_value
  )
    puts 'VPC created and tagged.'
  else
    puts 'VPC not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateVpca Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_cidr_block = '10.0.0.0/16'
    TRY.
        oo_result = lo_ec2->createvpc( iv_cidrblock = iv_cidr_block ).
  " oo_result is returned for testing purposes. "
        DATA(lv_vpc_id) = oo_result->get_vpc( )->get_vpcid( ).
        MESSAGE 'Created VPC.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateVpcreferência da API AWS SDK for SAP ABAP.

Use **`CreateVpcEndpoint`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateVpcEndpoint`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Rede privada virtual com servidores privados

CLI

AWS CLI

Exemplo 1: criar um endpoint de gateway

O exemplo `create-vpc-endpoint` a seguir cria um endpoint da VPC de gateway entre a VPC `vpc-1a2b3c4d` e o Amazon S3 na região `us-east-1` e associa a tabela de rotas `rtb-11aa22bb` ao endpoint.

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --service-name com.amazonaws.us-east-1.s3 \
    --route-table-ids rtb-11aa22bb
```

Saída:

```
{
    "VpcEndpoint": {
        "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":
\"\",\"Effect\":\"Allow\",\"Principal\":\"\*\",\"Action\":\"\*\",\"Resource\":
\"\*\"}]}",
        "VpcId": "vpc-1a2b3c4d",
        "State": "available",
        "ServiceName": "com.amazonaws.us-east-1.s3",
        "RouteTableIds": [
            "rtb-11aa22bb"
        ],
        "VpcEndpointId": "vpc-1a2b3c4d",
        "CreationTimestamp": "2015-05-15T09:40:50Z"
    }
}
```

Para obter mais informações, consulte Criar um endpoint de gateway no Guia do AWS PrivateLink usuário.

Exemplo 2: criar um endpoint de interface

O exemplo `create-vpc-endpoint` a seguir cria uma interface de endpoint da VPC entre a VPC `vpc-1a2b3c4d` e o Amazon S3 na região `us-east-1`. O comando cria o endpoint na sub-rede `subnet-1a2b3c4d`, o associa ao grupo de segurança `sg-1a2b3c4d` e adiciona uma tag com uma chave de “Serviço” e um valor de “S3".

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --vpc-endpoint-type Interface \
    --service-name com.amazonaws.us-east-1.s3 \
```

```
    --subnet-ids subnet-7b16de0c \
    --security-group-id sg-1a2b3c4d \
    --tag-specifications ResourceType=vpc-endpoint,Tags=[{Key=service,Value=S3}]
```

Saída:

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-1a2b3c4d5e6f1a2b3",
        "VpcEndpointType": "Interface",
        "VpcId": "vpc-1a2b3c4d",
        "ServiceName": "com.amazonaws.us-east-1.s3",
        "State": "pending",
        "RouteTableIds": [],
        "SubnetIds": [
            "subnet-1a2b3c4d"
        ],
        "Groups": [
            {
                "GroupId": "sg-1a2b3c4d",
                "GroupName": "default"
            }
        ],
        "PrivateDnsEnabled": false,
        "RequesterManaged": false,
        "NetworkInterfaceIds": [
            "eni-0b16f0581c8ac6877"
        ],
        "DnsEntries": [
            {
                "DnsName": "*.vpce-1a2b3c4d5e6f1a2b3-9hnenorg.s3.us-
east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
            },
            {
                "DnsName": "*.vpce-1a2b3c4d5e6f1a2b3-9hnenorg-us-east-1c.s3.us-
east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
            }
        ],
        "CreationTimestamp": "2021-03-05T14:46:16.030000+00:00",
        "Tags": [
            {
```

```
                "Key": "service",
                "Value": "S3"
            }
        ],
        "OwnerId": "123456789012"
    }
}
```

Para obter mais informações, consulte Criar uma interface VPC endpoint no Guia do AWS PrivateLink usuário.

Exemplo 3: criar um endpoint do balanceador de carga de gateway

O exemplo `create-vpc-endpoint` a seguir cria um endpoint do balanceador de carga de Gateway entre a VPC `vpc-111122223333aabbc` e um serviço configurado usando o balanceador de carga de Gateway.

```
aws ec2 create-vpc-endpoint \
    --service-name com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123 \
    --vpc-endpoint-type GatewayLoadBalancer \
    --vpc-id vpc-111122223333aabbc \
    --subnet-ids subnet-0011aabbcc2233445
```

Saída:

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-aabbaabbaabbaabba",
        "VpcEndpointType": "GatewayLoadBalancer",
        "VpcId": "vpc-111122223333aabbc",
        "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123",
        "State": "pending",
        "SubnetIds": [
            "subnet-0011aabbcc2233445"
        ],
        "RequesterManaged": false,
        "NetworkInterfaceIds": [
            "eni-0101012020303040 5"
        ],
        "CreationTimestamp": "2020-11-11T08:06:03.522Z",
        "OwnerId": "123456789012"
    }
```

```
}
```

Para obter mais informações, consulte os endpoints do Gateway Load Balancer no Guia do AWS PrivateLink usuário.

Exemplo 4: como criar um endpoint de recurso

O exemplo de `create-vpc-endpoint` a seguir cria um endpoint de recurso.

```
aws ec2 create-vpc-endpoint \
    --vpc-endpoint-type Resource \
    --vpc-id vpc-111122223333aabbc \
    --subnet-ids subnet-0011aabbcc2233445 \
    --resource-configuration-arn arn:aws:vpc-lattice-us-
east-1:123456789012:resourceconfiguration/rcfg-0123abcde98765432
```

Saída:

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-00939a7ed9EXAMPLE",
        "VpcEndpointType": "Resource",
        "VpcId": "vpc-111122223333aabbc",
        "State": "Pending",
        "SubnetIds": [
            "subnet-0011aabbcc2233445"
        ],
        "Groups": [
            {
                "GroupId": "sg-03e2f15fbfc09b000",
                "GroupName": "default"
            }
        ],
        "IpAddressType": "IPV4",
        "PrivateDnsEnabled": false,
        "CreationTimestamp": "2025-02-06T23:38:49.525000+00:00",
        "Tags": [],
        "OwnerId": "123456789012",
        "ResourceConfigurationArn": "arn:aws:vpc-lattice:us-
east-1:123456789012:resourceconfiguration/rcfg-0123abcde98765432"
    }
}
```

Para obter mais informações, consulte Endpoints de recursos no Guia do AWS PrivateLink usuário.

Exemplo 5: como criar um endpoint de rede de serviço

O exemplo de `create-vpc-endpoint` a seguir cria um endpoint de rede de serviço.

```
aws ec2 create-vpc-endpoint \
    --vpc-endpoint-type ServiceNetwork \
    --vpc-id vpc-111122223333aabbc \
    --subnet-ids subnet-0011aabbcc2233445 \
    --service-network-arn arn:aws:vpc-lattice:us-
east-1:123456789012:servicenetwork/sn-0101abcd5432abcd0 \
    --security-group-ids sg-0123456789012abcd
```

Saída:

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-0f00567fa8EXAMPLE",
        "VpcEndpointType": "ServiceNetwork",
        "VpcId": "vpc-111122223333aabbc",
        "State": "Pending",
        "SubnetIds": [
            "subnet-0011aabbcc2233445"
        ],
        "Groups": [
            {
                "GroupId": "sg-0123456789012abcd",
                "GroupName": "my-security-group"
            }
        ],
        "IpAddressType": "IPV4",
        "PrivateDnsEnabled": false,
        "CreationTimestamp": "2025-02-06T23:44:20.449000+00:00",
        "Tags": [],
        "OwnerId": "123456789012",
        "ServiceNetworkArn": "arn:aws:vpc-lattice:us-
east-1:123456789012:servicenetwork/sn-0101abcd5432abcd0"
    }
}
```

Para obter mais informações, consulte Endpoints da rede de serviços no Guia do AWS PrivateLink usuário.

- Para obter detalhes da API, consulte CreateVpcEndpointem Referência de AWS CLI Comandos.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param string $serviceName
     * @param string $vpcId
     * @param array $routeTableIds
     * @return array
     */
    public function createVpcEndpoint(string $serviceName, string $vpcId, array
 $routeTableIds): array
    {
        try {
            $result = $this->ec2Client->createVpcEndpoint([
                'ServiceName' => $serviceName,
                'VpcId' => $vpcId,
                'RouteTableIds' => $routeTableIds,
            ]);

            return $result["VpcEndpoint"];
        } catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte CreateVpcEndpointa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: nesse exemplo, crie um endpoint da VPC para o serviço com.amazonaws.eu-west-1.s3 na VPC vpc-0fc1ff23f45b678eb

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Saída:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- Para obter detalhes da API, consulte CreateVpcEndpointem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: nesse exemplo, crie um endpoint da VPC para o serviço com.amazonaws.eu-west-1.s3 na VPC vpc-0fc1ff23f45b678eb

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Saída:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- Para obter detalhes da API, consulte CreateVpcEndpointem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create_vpc_endpoint(
        self, vpc_id: str, service_name: str, route_table_ids: list[str]
    ) -> Dict[str, Any]:
        """
        Creates a new VPC endpoint for the specified service and associates it
 with the specified route tables.
```

```
        :param vpc_id: The ID of the VPC to create the endpoint in.
        :param service_name: The name of the service to create the endpoint for.
        :param route_table_ids: A list of IDs of the route tables to associate
with the endpoint.
        :return: A dictionary representing the newly created VPC endpoint.
        """
        try:
            response = self.ec2_client.create_vpc_endpoint(
                VpcId=vpc_id,
                ServiceName=service_name,
                RouteTableIds=route_table_ids,
            )
            return response["VpcEndpoint"]
        except ClientError as err:
            logger.error(
                "Couldn't create VPC endpoint for service %s. Here's why: %s:
%s",
                service_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateVpcEndpointReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_vpc_id = 'vpc-abc123'
    " iv_service_name = 'com.amazonaws.region.service'
```

```
    TRY.
        oo_result = lo_ec2->createvpcendpoint(             " oo_result is
 returned for testing purposes. "
          iv_vpcid = iv_vpc_id
          iv_servicename = iv_service_name
          it_routetableids = it_route_table_ids ).
        DATA(lv_vpc_endpoint_id) = oo_result->get_vpcendpoint( )-
>get_vpcendpointid( ).
        MESSAGE 'Created VPC endpoint.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateVpcEndpointreferência da API AWS SDK for SAP ABAP.

Usar **`CreateVpnConnection`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateVpnConnection`.

CLI

AWS CLI

Exemplo 1: criar uma conexão VPN com roteamento dinâmico

O exemplo `create-vpn-connection` a seguir cria uma conexão VPN entre o gateway privado virtual especificado e o gateway do cliente especificado e aplica tags à conexão VPN. A saída inclui as informações de configuração do dispositivo de gateway do cliente, no formato XML.

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --customer-gateway-id cgw-001122334455aabbc \
    --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
    --tag-specification 'ResourceType=vpn-connection,Tags=[{Key=Name,Value=BGP-
VPN}]'
```

Saída:

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "...configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
        "Category": "VPN",
        "State": "pending",
        "VpnConnectionId": "vpn-123123123123abcab",
        "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
        "Options": {
            "EnableAcceleration": false,
            "StaticRoutesOnly": false,
            "LocalIpv4NetworkCidr": "0.0.0.0/0",
            "RemoteIpv4NetworkCidr": "0.0.0.0/0",
            "TunnelInsideIpVersion": "ipv4",
            "TunnelOptions": [
                {},
                {}
            ]
        },
        "Routes": [],
        "Tags": [
             {
                "Key": "Name",
                "Value": "BGP-VPN"
            }
        ]
    }
}
```

Para obter mais informações, consulte Como a AWS Site-to-Site VPN funciona no Guia do usuário da AWS Site-to-Site VPN.

Exemplo 2: criar uma conexão VPN com roteamento estático

O exemplo `create-vpn-connection` a seguir cria uma conexão VPN entre o gateway privado virtual especificado e o gateway do cliente especificado. As opções especificam o roteamento estático. A saída inclui as informações de configuração do dispositivo de gateway do cliente, no formato XML.

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --customer-gateway-id cgw-001122334455aabbc \
```

```
    --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
    --options "{\"StaticRoutesOnly\":true}"
```

Saída:

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "..configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
        "Category": "VPN",
        "State": "pending",
        "VpnConnectionId": "vpn-123123123123abcab",
        "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
        "Options": {
            "EnableAcceleration": false,
            "StaticRoutesOnly": true,
            "LocalIpv4NetworkCidr": "0.0.0.0/0",
            "RemoteIpv4NetworkCidr": "0.0.0.0/0",
            "TunnelInsideIpVersion": "ipv4",
            "TunnelOptions": [
                {},
                {}
            ]
        },
        "Routes": [],
        "Tags": []
    }
}
```

Para obter mais informações, consulte Como a AWS Site-to-Site VPN funciona no Guia do usuário da AWS Site-to-Site VPN.

Exemplo 3: criar uma conexão VPN e especificar sua própria chave interna CIDR e pré-compartilhada

O exemplo `create-vpn-connection` a seguir cria uma conexão VPN e especifica o bloco CIDR do endereço IP interno e uma chave personalizada pré-compartilhada para cada túnel. Os valores especificados são retornados nas informações `CustomerGatewayConfiguration`.

```
aws ec2 create-vpn-connection \
```

```
    --type ipsec.1 \
    --customer-gateway-id cgw-001122334455aabbc \
    --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
    --options
 TunnelOptions='[{TunnelInsideCidr=169.254.12.0/30,PreSharedKey=ExamplePreSharedKey1},
{TunnelInsideCidr=169.254.13.0/30,PreSharedKey=ExamplePreSharedKey2}]'
```

Saída:

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "..configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
        "Category": "VPN",
        "State": "pending",
        "VpnConnectionId": "vpn-123123123123abcab",
        "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
        "Options": {
            "EnableAcceleration": false,
            "StaticRoutesOnly": false,
            "LocalIpv4NetworkCidr": "0.0.0.0/0",
            "RemoteIpv4NetworkCidr": "0.0.0.0/0",
            "TunnelInsideIpVersion": "ipv4",
            "TunnelOptions": [
                {
                    "OutsideIpAddress": "203.0.113.3",
                    "TunnelInsideCidr": "169.254.12.0/30",
                    "PreSharedKey": "ExamplePreSharedKey1"
                },
                {
                    "OutsideIpAddress": "203.0.113.5",
                    "TunnelInsideCidr": "169.254.13.0/30",
                    "PreSharedKey": "ExamplePreSharedKey2"
                }
            ]
        },
        "Routes": [],
        "Tags": []
    }
}
```

Para obter mais informações, consulte Como a AWS Site-to-Site VPN funciona no Guia do usuário da AWS Site-to-Site VPN.

Exemplo 4: criar uma conexão VPN que ofereça suporte ao tráfego IPv6

O exemplo `create-vpn-connection` a seguir cria uma conexão VPN que oferece suporte ao tráfego IPv6 entre o gateway de trânsito especificado e o gateway do cliente especificado. As opções de túnel para ambos os túneis especificam que AWS deve iniciar a negociação IKE.

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --transit-gateway-id tgw-12312312312312312 \
    --customer-gateway-id cgw-001122334455aabbc \
    --options TunnelInsideIpVersion=ipv6,TunnelOptions=[{StartupAction=start},
{StartupAction=start}]
```

Saída:

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "..configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
        "Category": "VPN",
        "State": "pending",
        "VpnConnectionId": "vpn-11111111122222222",
        "TransitGatewayId": "tgw-12312312312312312",
        "Options": {
            "EnableAcceleration": false,
            "StaticRoutesOnly": false,
            "LocalIpv6NetworkCidr": "::/0",
            "RemoteIpv6NetworkCidr": "::/0",
            "TunnelInsideIpVersion": "ipv6",
            "TunnelOptions": [
                {
                    "OutsideIpAddress": "203.0.113.3",
                    "StartupAction": "start"
                },
                {
                    "OutsideIpAddress": "203.0.113.5",
                    "StartupAction": "start"
                }
            ]
        },
        "Routes": [],
```

```
        "Tags": []
    }
}
```

Para obter mais informações, consulte Como a AWS Site-to-Site VPN funciona no Guia do usuário da AWS Site-to-Site VPN.

- Para obter detalhes da API, consulte CreateVpnConnectionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma conexão VPN entre o gateway privado virtual especificado e o gateway do cliente especificado. A saída inclui as informações de configuração de que o administrador de rede precisa no formato XML.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

Saída:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo cria a conexão VPN e captura a configuração em um arquivo com o nome especificado.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

Exemplo 3: esse exemplo cria uma conexão VPN, com roteamento estático, entre o gateway privado virtual especificado e o gateway do cliente especificado.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Para obter detalhes da API, consulte CreateVpnConnectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma conexão VPN entre o gateway privado virtual especificado e o gateway do cliente especificado. A saída inclui as informações de configuração de que o administrador de rede precisa no formato XML.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

Saída:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo cria a conexão VPN e captura a configuração em um arquivo com o nome especificado.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

Exemplo 3: esse exemplo cria uma conexão VPN, com roteamento estático, entre o gateway privado virtual especificado e o gateway do cliente especificado.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Para obter detalhes da API, consulte CreateVpnConnectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateVpnConnectionRoute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateVpnConnectionRoute`.

CLI

AWS CLI

Criar uma rota estática para uma conexão VPN

Este exemplo cria uma rota estática para a VPN especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 create-vpn-connection-route --vpn-connection-id vpn-40f41529 --
destination-cidr-block 11.12.0.0/16
```

- Para obter detalhes da API, consulte CreateVpnConnectionRouteem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria a rota estática especificada para a conexão VPN especificada.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- Para obter detalhes da API, consulte CreateVpnConnectionRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria a rota estática especificada para a conexão VPN especificada.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- Para obter detalhes da API, consulte CreateVpnConnectionRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateVpnGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateVpnGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de conexões de rede dedicadas

CLI

AWS CLI

Criar um gateway privado virtual

Este exemplo cria um gateway privado virtual.

Comando:

```
aws ec2 create-vpn-gateway --type ipsec.1
```

Saída:

```
{
    "VpnGateway": {
        "AmazonSideAsn": 64512,
        "State": "available",
        "Type": "ipsec.1",
        "VpnGatewayId": "vgw-9a4cacf3",
        "VpcAttachments": []
    }
```

```
}
```

Para criar um gateway privado virtual com um Amazon-side ASN específico

Esse exemplo cria um gateway privado virtual e especifica o Número de sistema autônomo (ASN) para o lado da Amazon da sessão BGP.

Comando:

```
aws ec2 create-vpn-gateway --type ipsec.1 --amazon-side-asn 65001
```

Saída:

```
{
    "VpnGateway": {
        "AmazonSideAsn": 65001,
        "State": "available",
        "Type": "ipsec.1",
        "VpnGatewayId": "vgw-9a4cacf3",
        "VpcAttachments": []
    }
}
```

- Para obter detalhes da API, consulte CreateVpnGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria o gateway privado virtual especificado.

```
New-EC2VpnGateway -Type ipsec.1
```

Saída:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
```

```
VpnGatewayId     : vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte CreateVpnGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria o gateway privado virtual especificado.

```
New-EC2VpnGateway -Type ipsec.1
```

Saída:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte CreateVpnGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteCustomerGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteCustomerGateway`.

CLI

AWS CLI

Excluir um gateway do cliente

Este exemplo exclui o gateway do cliente especificado. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-customer-gateway --customer-gateway-id cgw-0e11f167
```

- Para obter detalhes da API, consulte DeleteCustomerGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o gateway do cliente especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on
 Target "cgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteCustomerGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o gateway do cliente especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on
 Target "cgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteCustomerGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteDhcpOptions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteDhcpOptions`.

CLI

AWS CLI

Excluir um conjunto de opções DHCP

Este exemplo exclui o conjunto de opções DHCP especificado. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-dhcp-options --dhcp-options-id dopt-d9070ebb
```

- Para obter detalhes da API, consulte DeleteDhcpOptionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o conjunto de opções DHCP especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
 "dopt-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteDhcpOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o conjunto de opções DHCP especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
 "dopt-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteDhcpOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteFlowLogs`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteFlowLogs`.

CLI

AWS CLI

Excluir um log de fluxo

O exemplo `delete-flow-logs` a seguir exclui o log de fluxo especificado.

```
aws ec2 delete-flow-logs --flow-log-id fl-11223344556677889
```

Saída:

```
{
    "Unsuccessful": []
}
```

- Para obter detalhes da API, consulte DeleteFlowLogsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo remove o FlowLogId fl-01a2b3456a789c01 fornecido

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
 "fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteFlowLogsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove o FlowLogId fl-01a2b3456a789c01 fornecido

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
 "fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteFlowLogsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteInternetGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteInternetGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com servidores privados

CLI

AWS CLI

Excluir um gateway da internet

O exemplo `delete-internet-gateway` a seguir exclui o gateway da internet especificado.

```
aws ec2 delete-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE
```

Este comando não produz saída.

Para obter mais informações, consulte Gateways da Internet no Guia do usuário da Amazon VPC.

- Para obter detalhes da API, consulte DeleteInternetGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o gateway da internet especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on
 Target "igw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteInternetGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o gateway da internet especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on
 Target "igw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteInternetGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteKeyPair`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteKeyPair`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma rede privada virtual básica
- Como criar um serviço de contêiner para instâncias de máquinas virtuais

- Conceitos básicos de compras do mercado de software
- Conceitos básicos de streaming gerenciado
- Introdução às máquinas virtuais

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyPair(string keyPairName)
    {
        try
        {
            await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError($"KeyPair {keyPairName} does not exist and
cannot be deleted. Please verify the key pair name and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
```

```
            return false;
        }
    }

    /// <summary>
    /// Delete the temporary file where the key pair information was saved.
    /// </summary>
    /// <param name="tempFileName">The path to the temporary file.</param>
    public void DeleteTempFile(string tempFileName)
    {
        if (File.Exists(tempFileName))
        {
            File.Delete(tempFileName);
        }
    }
```

- Para obter detalhes da API, consulte DeleteKeyPaira Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#
# And:
#       0 - If successful.
#       1 - If it fails.
```

```
###############################################################################
function ec2_delete_keypair() {
  local key_pair_name response

  local option OPTARG # Required to use getopts command in a function.
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-key-pair \
    --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
  }

  return 0
```

```
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
```

```
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteKeyPairem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
  \param keyPairName: A name for a key pair.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::EC2::deleteKeyPair(const Aws::String &keyPairName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteKeyPairRequest request;

    request.SetKeyName(keyPairName);
    const Aws::EC2::Model::DeleteKeyPairOutcome outcome =
 ec2Client.DeleteKeyPair(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete key pair " << keyPairName <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted key pair named " << keyPairName <<
                  std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteKeyPaira Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir um par de chaves

O exemplo `delete-key-pair` a seguir exclui o par de chaves especificado.

```
aws ec2 delete-key-pair \
    --key-name my-key-pair
```

Saída:

```
{
    "Return": true,
    "KeyPairId": "key-03c8d3aceb53b507"
}
```

Para obter mais informações, consulte Criar e excluir pares de chaves no Guia do usuário da interface da linha de comando AWS .

- Para obter detalhes da API, consulte DeleteKeyPairem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a key pair asynchronously.
    *
    * @param keyPair the name of the key pair to delete
    * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
    *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
    *         that provides the result of the key pair deletion operation.
    */
   public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String
keyPair) {
       DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
           .keyName(keyPair)
           .build();

       // Initiate the asynchronous request to delete the key pair.
       CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete key pair: " +
keyPair, ex);
           } else if (resp == null) {
               throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
           }
       });
   }
```

- Para obter detalhes da API, consulte DeleteKeyPair a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes the specified key pair, by removing the public key from Amazon EC2.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new DeleteKeyPairCommand({
    KeyName: keyName,
  });

  try {
    await client.send(command);
    console.log("Successfully deleted key pair.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide the required value?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteKeyPair a Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- Para obter detalhes da API, consulte a DeleteKeyPairreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o par de chaves especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteKeyPairem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o par de chaves especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteKeyPairem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """
```

```
    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
```

```
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
                    f"The key pair '{key_name}' does not exist and cannot be
deleted. "
                    "Please verify the key pair name and try again."
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteKeyPairReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envolva delete_key que também remove a chave PEM privada de apoio.

```
    pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
        if let Some(key_name) = self.key_pair.key_name() {
            ec2.delete_key_pair(key_name).await?;
            if let Some(key_path) = self.key_file_path() {
```

```
                if let Err(err) = util.remove(key_path) {
                    eprintln!("Failed to remove {key_path:?} ({err:?})");
                }
            }
        }
        Ok(())
    }
```

```
    pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
        let key_name: String = key_name.into();
        tracing::info!("Deleting key pair {key_name}");
        self.client
            .delete_key_pair()
            .key_name(key_name)
            .send()
            .await?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a DeleteKeyPairreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->deletekeypair( iv_keyname = iv_key_name ).
        MESSAGE 'Amazon EC2 key pair deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteKeyPairreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Delete an EC2 key pair.
    ///
    /// - Parameter keyPair: The name of the key pair to delete.
    ///
    /// - Returns: `true` if the key pair is deleted successfully; otherwise
    ///   `false`.
    func deleteKeyPair(keyPair: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteKeyPair(
                input: DeleteKeyPairInput(
                    keyName: keyPair
                )
            )

            return true
        } catch {
            print("*** Error deleting the key pair:
 \(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte DeleteKeyPaira referência da API AWS SDK for Swift.

Use **`DeleteLaunchTemplate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteLaunchTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Rede privada virtual com servidores privados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
```

```
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DeleteLaunchTemplatea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para excluir um modelo de execução

Este exemplo exclui o modelo de execução especificado.

Comando:

```
aws ec2 delete-launch-template --launch-template-id lt-0abcd290751193123
```

Saída:

```
{
  "LaunchTemplate": {
      "LatestVersionNumber": 2,
      "LaunchTemplateId": "lt-0abcd290751193123",
      "LaunchTemplateName": "TestTemplate",
      "DefaultVersionNumber": 2,
      "CreatedBy": "arn:aws:iam::123456789012:root",
      "CreateTime": "2017-11-23T16:46:25.000Z"
  }
}
```

- Para obter detalhes da API, consulte DeleteLaunchTemplateem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
```

- Para obter detalhes da API, consulte DeleteLaunchTemplatea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
```

```
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
```

```
    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DeleteLaunchTemplateReferência da API AWS SDK for Python (Boto3).

Usar **`DeleteNetworkAcl`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteNetworkAcl`.

CLI

AWS CLI

Excluir uma ACL de rede

Este exemplo exclui a rede ACL especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-network-acl --network-acl-id acl-5fb85d36
```

- Para obter detalhes da API, consulte DeleteNetworkAclem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a ACL de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
 "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteNetworkAclem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a ACL de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Saída:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
 "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteNetworkAclem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteNetworkAclEntry`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteNetworkAclEntry`.

CLI

AWS CLI

Excluir uma entrada de ACL de rede

Este exemplo exclui a regra de entrada número 100 da ACL de rede especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 100
```

- Para obter detalhes da API, consulte DeleteNetworkAclEntryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a regra especificada da ACL de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on
 Target "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteNetworkAclEntryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a regra especificada da ACL de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on
 Target "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteNetworkAclEntryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteNetworkInterface`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteNetworkInterface`.

CLI

AWS CLI

Excluir uma interface de rede

Este exemplo exclui a interface de rede especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-network-interface --network-interface-id eni-e5aa89a3
```

- Para obter detalhes da API, consulte DeleteNetworkInterfaceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a interface de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on
 Target "eni-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteNetworkInterfaceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a interface de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on
 Target "eni-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteNetworkInterfaceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeletePlacementGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeletePlacementGroup`.

CLI

AWS CLI

Excluir um grupo de posicionamento

Esse exemplo de comando exclui o grupo de posicionamento especificado.

Comando:

```
aws ec2 delete-placement-group --group-name my-cluster
```

- Para obter detalhes da API, consulte DeletePlacementGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o grupo de posicionamento especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
 "my-placement-group".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeletePlacementGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o grupo de posicionamento especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
 "my-placement-group".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeletePlacementGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteRoute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteRoute`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução aos gateways de trânsito de rede

CLI

AWS CLI

Para excluir uma rota

Este exemplo exclui a rota especificada da tabela de rotas especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-route --route-table-id rtb-22574640 --destination-cidr-
block 0.0.0.0/0
```

- Para obter detalhes da API, consulte DeleteRouteem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a rota especificada da tabela de rotas especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a rota especificada da tabela de rotas especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteRouteTable`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteRouteTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com servidores privados
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Excluir uma tabela de rotas

Este exemplo exclui a tabela de rotas especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-route-table --route-table-id rtb-22574640
```

- Para obter detalhes da API, consulte DeleteRouteTableem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a tabela de rotas especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
 "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteRouteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a tabela de rotas especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
 "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteRouteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteSecurityGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteSecurityGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Configurar a conectividade do serviço de contêiner
- Criar uma rede privada virtual básica
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criar uma instância de banco de dados relacional
- Criação e gerenciamento de volumes de armazenamento em bloco
- Conceitos básicos de compras do mercado de software
- Introdução aos bancos de dados de grafos
- Introdução ao balanceamento de carga
- Conceitos básicos de streaming gerenciado
- Introdução às máquinas virtuais
- Rede privada virtual com servidores privados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
```

```
    /// Delete an Amazon EC2 security group.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteSecurityGroup(string groupId)
    {
        try
        {
            var response =
                await _amazonEC2.DeleteSecurityGroupAsync(
                    new DeleteSecurityGroupRequest { GroupId = groupId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"Security Group {groupId} does not exist and cannot be
deleted. Please verify the ID and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteSecurityGroupa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security
 group.
#
# Parameters:
#       -i security_group_id - The ID of the security group to delete.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_security_group() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
```

```
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-security-group --group-id "$security_group_id" --
output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
```

```
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteSecurityGroupem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete a security group.
/*!
  \param securityGroupID: A security group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::deleteSecurityGroup(const Aws::String &securityGroupID,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteSecurityGroupRequest request;

    request.SetGroupId(securityGroupID);
    Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =
 ec2Client.DeleteSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete security group " << securityGroupID <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted security group " << securityGroupID <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteSecurityGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

[EC2-Classic] Para excluir um grupo de segurança

Este exemplo exclui o grupo de segurança chamado `MySecurityGroup`. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-security-group --group-name MySecurityGroup
```

[EC2-VPC] Para excluir um grupo de segurança

Este exemplo exclui o grupo de segurança com o ID `sg-903004f8`. Observe que você não pode referenciar um grupo de segurança EC2-VPC pelo nome. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-security-group --group-id sg-903004f8
```

Para obter mais informações, consulte Using Security Groups no Guia do usuário da AWS Command Line Interface.

- Para obter detalhes da API, consulte DeleteSecurityGroupem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is
deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();
```

```
        CompletableFuture<DeleteSecurityGroupResponse> response =
 getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with
 Id " + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
 security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }
```

- Para obter detalhes da API, consulte DeleteSecurityGroupa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes a security group.
 * @param {{ groupId: string }} options
 */
export const main = async ({ groupId }) => {
  const client = new EC2Client({});
  const command = new DeleteSecurityGroupCommand({
    GroupId: groupId,
  });

  try {
    await client.send(command);
```

```
    console.log("Security group deleted successfully.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteSecurityGroupa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteSecurityGroupreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui o grupo de segurança especificado para EC2-VPC. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
 "sg-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Este exemplo exclui o grupo de segurança especificado para EC2-Classic.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Para obter detalhes da API, consulte DeleteSecurityGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui o grupo de segurança especificado para EC2-VPC. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
 "sg-12345678".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Este exemplo exclui o grupo de segurança especificado para EC2-Classic.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Para obter detalhes da API, consulte DeleteSecurityGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
```

```
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.

        :param security_group_id: The ID of the security group to delete.
Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an
AWS service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
'{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
'{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
because it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
because it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
```

```
                    "\n\t- Removed from VPC's as a default group"
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteSecurityGroupReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
       tracing::info!("Deleting security group {group_id}");
       self.client
           .delete_security_group()
           .group_id(group_id)
           .send()
           .await?;
       Ok(())
   }
```

- Para obter detalhes da API, consulte a DeleteSecurityGroupreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).
        MESSAGE 'Security group deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteSecurityGroupreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Delete a security group.
    ///
    /// - Parameter groupId: The ID of the security group to delete.
    ///
```

```
    /// - Returns: `true` on successful deletion; `false` on error.
    func deleteSecurityGroup(groupId: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteSecurityGroup(
                input: DeleteSecurityGroupInput(
                    groupId: groupId
                )
            )

            return true
        } catch {
            print("*** Error deleting the security group:
\(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte DeleteSecurityGroupa referência da API AWS SDK for Swift.

Use **`DeleteSnapshot`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteSnapshot`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

CLI

AWS CLI

Para excluir um snapshot

Este exemplo de comando exclui um snapshot com o ID de snapshot de `snap-1234567890abcdef0`. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-snapshot --snapshot-id snap-1234567890abcdef0
```

- Para obter detalhes da API, consulte DeleteSnapshotem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o snapshot especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
 "snap-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteSnapshotem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o snapshot especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Saída:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
 "snap-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteSnapshotem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteSnapshotreferência da API AWS SDK for Rust.

Usar **`DeleteSpotDatafeedSubscription`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteSpotDatafeedSubscription`.

CLI

AWS CLI

Cancelar uma assinatura de feed de dados da Instância Spot

Este exemplo de comando exclui uma assinatura de feed de dados Spot para a conta. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-spot-datafeed-subscription
```

- Para obter detalhes da API, consulte DeleteSpotDatafeedSubscriptionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o feed de dados da instância spot. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2SpotDatafeedSubscription
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
 (DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteSpotDatafeedSubscriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o feed de dados da instância spot. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2SpotDatafeedSubscription
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
 (DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteSpotDatafeedSubscriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteSubnet`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteSubnet`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com servidores privados
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Excluir uma sub-rede

Este exemplo exclui a sub-rede especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-subnet --subnet-id subnet-9d4a7b6c
```

- Para obter detalhes da API, consulte DeleteSubnetem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a sub-rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target
 "subnet-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteSubnetem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a sub-rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target
 "subnet-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteSubnetem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteTags`.

CLI

AWS CLI

Exemplo 1: excluir uma tag de um recurso

O exemplo `delete-tags` a seguir exclui a tag `Stack=Test` da imagem especificada. Quando você especifica um valor e um nome de chave, a tag é excluída somente se o valor da tag corresponder ao valor especificado.

```
aws ec2 delete-tags \
    --resources ami-1234567890abcdef0 \
    --tags Key=Stack,Value=Test
```

É opcional especificar o valor de uma tag. O exemplo `delete-tags` a seguir exclui a tag com o nome de chave `purpose` da instância especificada, independentemente do valor da tag.

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 \
    --tags Key=purpose
```

Se você especificar a string vazia como o valor da tag, a tag será excluída somente se o valor da tag for a string vazia. O exemplo `delete-tags` a seguir especifica a string vazia como o valor da tag a ser excluída.

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 \
    --tags Key=Name,Value=
```

Exemplo 2: excluir uma tag de vários recursos

O exemplo `delete-tags` a seguir exclui a tag ``Purpose=Test`` de uma instância e de uma AMI. Conforme mostrado no exemplo anterior, você pode omitir o valor da tag do comando.

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 ami-1234567890abcdef0 \
```

```
    --tags Key=Purpose
```

- Para obter detalhes da API, consulte DeleteTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a tag especificada do recurso especificado, independentemente do valor da tag. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Exemplo 2: esse exemplo exclui a tag especificada do recurso especificado, mas somente se o valor da tag corresponder. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -
Force
```

Exemplo 3: esse exemplo exclui a tag especificada do recurso especificado, independentemente do valor da tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Exemplo 4: esse exemplo exclui a tag especificada do recurso especificado, mas somente se o valor da tag corresponder.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Para obter detalhes da API, consulte DeleteTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a tag especificada do recurso especificado, independentemente do valor da tag. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Exemplo 2: esse exemplo exclui a tag especificada do recurso especificado, mas somente se o valor da tag corresponder. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -
Force
```

Exemplo 3: esse exemplo exclui a tag especificada do recurso especificado, independentemente do valor da tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Exemplo 4: esse exemplo exclui a tag especificada do recurso especificado, mas somente se o valor da tag corresponder.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Para obter detalhes da API, consulte DeleteTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteVolume`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteVolume`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criação e gerenciamento de volumes de armazenamento em bloco
- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

CLI

AWS CLI

Excluir um volume

Este exemplo de comando exclui um volume disponível com o ID do volume de `vol-049df61146c4d7901`. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-volume --volume-id vol-049df61146c4d7901
```

- Para obter detalhes da API, consulte DeleteVolumeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo desvincula o volume especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target
 "vol-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVolumeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desvincula o volume especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target
 "vol-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVolumeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteVpc`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteVpc`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com gerenciamento de endereços IP
- Rede privada virtual com servidores privados
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Para excluir uma VPC

Este exemplo exclui a VPC especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-vpc --vpc-id vpc-a01106c2
```

- Para obter detalhes da API, consulte DeleteVpcem Referência de AWS CLI Comandos.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param string $vpcId
     * @return void
     */
    public function deleteVpc(string $vpcId)
    {
        try {
            $this->ec2Client->deleteVpc([
                "VpcId" => $vpcId,
            ]);
        }catch(Ec2Exception $caught){
            echo "There was a problem deleting the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
```

```
    }
```

- Para obter detalhes da API, consulte DeleteVpca Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a VPC especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVpcem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a VPC especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVpcem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def delete(self, vpc_id: str) -> None:
        """
        Deletes the specified VPC.

        :param vpc_id: The ID of the VPC to delete.
        """
        try:
            self.ec2_client.delete_vpc(VpcId=vpc_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete VPC %s. Here's why: %s: %s",
                vpc_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteVpcReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->deletevpc( iv_vpcid = iv_vpc_id ).
        MESSAGE 'Deleted VPC.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteVpcreferência da API AWS SDK for SAP ABAP.

Use **`DeleteVpcEndpoints`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteVpcEndpoints`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Rede privada virtual com servidores privados

CLI

AWS CLI

Para excluir um endpoint

Este exemplo exclui os endpoints vpce-aa22bb33 e vpce-1a2b3c4d. Se o comando for parcialmente bem-sucedido ou malsucedido, uma lista de itens malsucedidos será retornada. Se houver sucesso, a lista retornada ficará vazia.

Comando:

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-aa22bb33 vpce-1a2b3c4d
```

Saída:

```
{
  "Unsuccessful": []
}
```

- Para obter detalhes da API, consulte DeleteVpcEndpointsem Referência de AWS CLI Comandos.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param string $vpcEndpointId
     * @return void
     */
    public function deleteVpcEndpoint(string $vpcEndpointId)
    {
        try {
            $this->ec2Client->deleteVpcEndpoints([
                "VpcEndpointIds" => [$vpcEndpointId],
            ]);
        }catch (Ec2Exception $caught){
            echo "There was a problem deleting the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DeleteVpcEndpointsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete_vpc_endpoints(self, vpc_endpoint_ids: list[str]) -> None:
        """
        Deletes the specified VPC endpoints.

        :param vpc_endpoint_ids: A list of IDs of the VPC endpoints to delete.
        """
```

```
        try:
            self.ec2_client.delete_vpc_endpoints(VpcEndpointIds=vpc_endpoint_ids)
        except ClientError as err:
            logger.error(
                "Couldn't delete VPC endpoints %s. Here's why: %s: %s",
                vpc_endpoint_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteVpcEndpointsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->deletevpcendpoints( it_vpcendpointids = it_vpc_endpoint_ids ).
        MESSAGE 'Deleted VPC endpoint(s).' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteVpcEndpointsreferência da API AWS SDK for SAP ABAP.

Usar **`DeleteVpnConnection`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteVpnConnection`.

CLI

AWS CLI

Excluir uma conexão VPN

Este exemplo exclui a conexão VPN especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-vpn-connection --vpn-connection-id vpn-40f41529
```

- Para obter detalhes da API, consulte DeleteVpnConnectionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui conexão VPN especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
 "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVpnConnectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui conexão VPN especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
 "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVpnConnectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteVpnConnectionRoute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteVpnConnectionRoute`.

CLI

AWS CLI

Excluir uma rota estática de uma conexão VPN

Este exemplo exclui a rota estática especificada da conexão VPN especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-vpn-connection-route --vpn-connection-id vpn-40f41529 --
destination-cidr-block 11.12.0.0/16
```

- Para obter detalhes da API, consulte DeleteVpnConnectionRouteem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a rota estática especificada da conexão VPN especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on
 Target "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVpnConnectionRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a rota estática especificada da conexão VPN especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on
 Target "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVpnConnectionRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteVpnGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteVpnGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de conexões de rede dedicadas

CLI

AWS CLI

Excluir um gateway privado virtual

Este exemplo exclui o gateway privado virtual especificado. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 delete-vpn-gateway --vpn-gateway-id vgw-9a4cacf3
```

- Para obter detalhes da API, consulte DeleteVpnGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o gateway privado virtual especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
 "vgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVpnGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o gateway privado virtual especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
 "vgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteVpnGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeregisterImage`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterImage`.

CLI

AWS CLI

Cancelar o registro de uma AMI

Este exemplo cancela o registro da AMI especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 deregister-image --image-id ami-4fa54026
```

- Para obter detalhes da API, consulte DeregisterImageem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela o registro da AMI especificada.

```
Unregister-EC2Image -ImageId ami-12345678
```

- Para obter detalhes da API, consulte DeregisterImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro da AMI especificada.

```
Unregister-EC2Image -ImageId ami-12345678
```

- Para obter detalhes da API, consulte DeregisterImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar `DescribeAccountAttributes` com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAccountAttributes`.

CLI

AWS CLI

Para descrever todos os atributos da sua AWS conta

Este exemplo descreve os atributos da sua AWS conta.

Comando:

```
aws ec2 describe-account-attributes
```

Saída:

```
{
    "AccountAttributes": [
        {
            "AttributeName": "vpc-max-security-groups-per-interface",
            "AttributeValues": [
                {
                    "AttributeValue": "5"
                }
            ]
        },
        {
            "AttributeName": "max-instances",
            "AttributeValues": [
                {
                    "AttributeValue": "20"
                }
            ]
        },
        {
            "AttributeName": "supported-platforms",
            "AttributeValues": [
                {
                    "AttributeValue": "EC2"
                },
                {
                    "AttributeValue": "VPC"
                }
            ]
        },
        {
            "AttributeName": "default-vpc",
            "AttributeValues": [
                {
                    "AttributeValue": "none"
                }
            ]
        },
        {
            "AttributeName": "max-elastic-ips",
```

```
            "AttributeValues": [
                {
                    "AttributeValue": "5"
                }
            ]
        },
        {
            "AttributeName": "vpc-max-elastic-ips",
            "AttributeValues": [
                {
                    "AttributeValue": "5"
                }
            ]
        }
    ]
}
```

Para descrever um único atributo para sua AWS conta

Este exemplo descreve o `supported-platforms` atributo da sua AWS conta.

Comando:

```
aws ec2 describe-account-attributes --attribute-names supported-platforms
```

Saída:

```
{
    "AccountAttributes": [
        {
            "AttributeName": "supported-platforms",
            "AttributeValues": [
                {
                    "AttributeValue": "EC2"
                },
                {
                    "AttributeValue": "VPC"
                }
            ]
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeAccountAttributesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve se você pode executar instâncias EC2-VPC na EC2-Classic e na região, ou somente em EC2-VPC.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Saída:

```
AttributeValue
--------------
EC2
VPC
```

Exemplo 2: esse exemplo descreve a VPC padrão ou indica “nenhuma” caso você não tenha uma VPC padrão na região.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Saída:

```
AttributeValue
--------------
vpc-12345678
```

Exemplo 3: Este exemplo descreve o número máximo de On-Demand instâncias que você pode executar.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Saída:

```
AttributeValue
```

```
--------------
20
```

- Para obter detalhes da API, consulte DescribeAccountAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve se você pode executar instâncias EC2-VPC na EC2-Classic e na região, ou somente em EC2-VPC.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Saída:

```
AttributeValue
--------------
EC2
VPC
```

Exemplo 2: esse exemplo descreve a VPC padrão ou indica “nenhuma” caso você não tenha uma VPC padrão na região.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Saída:

```
AttributeValue
--------------
vpc-12345678
```

Exemplo 3: Este exemplo descreve o número máximo de On-Demand instâncias que você pode executar.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Saída:

```
AttributeValue
```

```
--------------
20
```

- Para obter detalhes da API, consulte DescribeAccountAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeAddresses`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAddresses`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Elastic IP addresses.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeAddresses(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAddressesRequest request;
    Aws::EC2::Model::DescribeAddressesOutcome outcome =
 ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
                  std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
                  std::setw(30) << "Allocation ID" << std::setw(25) <<
                  "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
 outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
```

```
                    Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                            address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                      address.GetInstanceId() << std::setw(15) <<
                      address.GetPublicIp() << std::setw(10) << domainString <<
                      std::setw(30) << address.GetAllocationId() << std::setw(25)
                      << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeAddressesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: recuperar detalhes de todos os seus endereços IP elásticos

O exemplo `describe addresses` a seguir mostra os detalhes dos seus endereços IP elásticos.

```
aws ec2 describe-addresses
```

Saída:

```
{
    "Addresses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PublicIp": "198.51.100.0",
            "PublicIpv4Pool": "amazon",
            "Domain": "standard"
        },
```

```
        {
            "Domain": "vpc",
            "PublicIpv4Pool": "amazon",
            "InstanceId": "i-1234567890abcdef0",
            "NetworkInterfaceId": "eni-12345678",
            "AssociationId": "eipassoc-12345678",
            "NetworkInterfaceOwnerId": "123456789012",
            "PublicIp": "203.0.113.0",
            "AllocationId": "eipalloc-12345678",
            "PrivateIpAddress": "10.0.1.241"
        }
    ]
}
```

Exemplo 2: Para recuperar detalhes dos seus endereços IP elásticos EC2-VPC

O exemplo `describe-addresses` a seguir mostra os detalhes dos seus endereços IP elásticos para usar com instâncias em uma VPC.

```
aws ec2 describe-addresses \
    --filters "Name=domain,Values=vpc"
```

Saída:

```
{
    "Addresses": [
        {
            "Domain": "vpc",
            "PublicIpv4Pool": "amazon",
            "InstanceId": "i-1234567890abcdef0",
            "NetworkInterfaceId": "eni-12345678",
            "AssociationId": "eipassoc-12345678",
            "NetworkInterfaceOwnerId": "123456789012",
            "PublicIp": "203.0.113.0",
            "AllocationId": "eipalloc-12345678",
            "PrivateIpAddress": "10.0.1.241"
        }
    ]
}
```

Exemplo 3: recuperar detalhes sobre um endereço IP elástico especificado pelo ID de alocação

O `describe-addresses` exemplo a seguir exibe detalhes sobre o endereço IP elástico com o ID de alocação especificado, que está associado a uma instância em EC2-VPC.

```
aws ec2 describe-addresses \
    --allocation-ids eipalloc-282d9641
```

Saída:

```
{
    "Addresses": [
        {
            "Domain": "vpc",
            "PublicIpv4Pool": "amazon",
            "InstanceId": "i-1234567890abcdef0",
            "NetworkInterfaceId": "eni-1a2b3c4d",
            "AssociationId": "eipassoc-123abc12",
            "NetworkInterfaceOwnerId": "1234567891012",
            "PublicIp": "203.0.113.25",
            "AllocationId": "eipalloc-282d9641",
            "PrivateIpAddress": "10.251.50.12"
        }
    ]
}
```

Exemplo 4: recuperar detalhes sobre um endereço IP elástico especificado pelo endereço IP privado de VPC

O `describe-addresses` exemplo a seguir exibe detalhes sobre o endereço IP elástico associado a um determinado endereço IP privado em EC2-VPC.

```
aws ec2 describe-addresses \
    --filters "Name=private-ip-address,Values=10.251.50.12"
```

Exemplo 5: Para recuperar detalhes sobre endereços IP elásticos em EC2-Classic

O `describe-addresses` exemplo a seguir exibe detalhes sobre seus endereços IP elásticos para uso em EC2-Classic.

```
aws ec2 describe-addresses \
    --filters "Name=domain,Values=standard"
```

Saída:

```
{
    "Addresses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PublicIp": "203.0.110.25",
            "PublicIpv4Pool": "amazon",
            "Domain": "standard"
        }
    ]
}
```

Exemplo 6: recuperar detalhes sobre um endereço IP elástico especificado pelo endereço IP público

O `describe-addresses` exemplo a seguir exibe detalhes sobre o endereço IP elástico com o valor`203.0.110.25`, que está associado a uma instância em EC2-Classic.

```
aws ec2 describe-addresses \
    --public-ips 203.0.110.25
```

Saída:

```
{
    "Addresses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PublicIp": "203.0.110.25",
            "PublicIpv4Pool": "amazon",
            "Domain": "standard"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeAddressesem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeAddressesCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified Elastic IP addresses or all of your Elastic IP
 addresses.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new DescribeAddressesCommand({
    // You can omit this property to show all addresses.
    AllocationIds: [allocationId],
  });

  try {
    const { Addresses } = await client.send(command);
    const addressList = Addresses.map((address) => ` • ${address.PublicIp}`);
    console.log("Elastic IP addresses:");
    console.log(addressList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationId.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeAddressesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Saída:

```
AllocationId            : eipalloc-12345678
AssociationId           : eipassoc-12345678
Domain                  : vpc
InstanceId              : i-87654321
NetworkInterfaceId      : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress        : 10.0.2.172
PublicIp                : 198.51.100.2
```

Exemplo 2: esse exemplo descreve os endereços IP elásticos para instâncias em uma VPC. Essa sintaxe requer a PowerShell versão 3 ou posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Exemplo 3: Este exemplo descreve o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Saída:

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
```

```
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

Exemplo 4: Este exemplo descreve seus endereços IP elásticos para instâncias em EC2-Classic. Essa sintaxe requer a PowerShell versão 3 ou posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Exemplo 5: esse exemplo descreve todos os endereços IP elásticos.

```
Get-EC2Address
```

Exemplo 6: esse exemplo retorna o IP público e privado para o ID da instância fornecido no filtro

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Saída:

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

Exemplo 7: esse exemplo recupera todos os IPs elásticos com seu ID de alocação, ID de associação e IDs de instância

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Saída:

```
InstanceId          AssociationId              AllocationId
 PublicIp
----------          -------------              ------------
 --------
```

```
                                                    eipalloc-012e3b456789e1fad
 17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
 17.212.124.77
                                                    eipalloc-012345678eeabcfad
 17.212.225.7
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
 37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
 37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
 37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
 37.215.222.270
```

Exemplo 8: esse exemplo busca uma lista de endereços IP do EC2 que correspondem à chave de tag “Category” com o valor “Prod”

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Saída:

```
AllocationId            : eipalloc-0123f456f81a01b58
AssociationId           : eipassoc-0d1b23a456d103810
CustomerOwnedIp         :
CustomerOwnedIpv4Pool   :
Domain                  : vpc
InstanceId              : i-012e3cb4df567e1aa
NetworkBorderGroup      : eu-west-1
NetworkInterfaceId      : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress        : 192.168.1.84
PublicIp                : 34.250.81.29
PublicIpv4Pool          : amazon
Tags                    : {Category, Name}
```

- Para obter detalhes da API, consulte DescribeAddressesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Saída:

```
AllocationId            : eipalloc-12345678
AssociationId           : eipassoc-12345678
Domain                  : vpc
InstanceId              : i-87654321
NetworkInterfaceId      : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress        : 10.0.2.172
PublicIp                : 198.51.100.2
```

Exemplo 2: esse exemplo descreve os endereços IP elásticos para instâncias em uma VPC. Essa sintaxe requer a PowerShell versão 3 ou posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Exemplo 3: Este exemplo descreve o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Saída:

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

Exemplo 4: Este exemplo descreve seus endereços IP elásticos para instâncias em EC2-Classic. Essa sintaxe requer a PowerShell versão 3 ou posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Exemplo 5: esse exemplo descreve todos os endereços IP elásticos.

```
Get-EC2Address
```

Exemplo 6: esse exemplo retorna o IP público e privado para o ID da instância fornecido no filtro

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Saída:

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

Exemplo 7: esse exemplo recupera todos os IPs elásticos com seu ID de alocação, ID de associação e IDs de instância

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Saída:

```
InstanceId          AssociationId              AllocationId
 PublicIp
----------          -------------              ------------
 --------
                                               eipalloc-012e3b456789e1fad
 17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
 17.212.124.77
                                               eipalloc-012345678eeabcfad
 17.212.225.7
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
 37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
 37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
 37.210.82.27
```

```
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
 37.215.222.270
```

Exemplo 8: esse exemplo busca uma lista de endereços IP do EC2 que correspondem à chave de tag “Category” com o valor “Prod”

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Saída:

```
AllocationId            : eipalloc-0123f456f81a01b58
AssociationId           : eipassoc-0d1b23a456d103810
CustomerOwnedIp         :
CustomerOwnedIpv4Pool   :
Domain                  : vpc
InstanceId              : i-012e3cb4df567e1aa
NetworkBorderGroup      : eu-west-1
NetworkInterfaceId      : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress        : 192.168.1.84
PublicIp                : 34.250.81.29
PublicIpv4Pool          : amazon
Tags                    : {Category, Name}
```

- Para obter detalhes da API, consulte DescribeAddressesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeaddresses( ).                       "
oo_result is returned for testing purposes. "
```

```
        DATA(lt_addresses) = oo_result->get_addresses( ).
        MESSAGE 'Retrieved information about Elastic IP addresses.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAddressesreferência da API AWS SDK for SAP ABAP.

Use **`DescribeAvailabilityZones`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAvailabilityZones`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Criar uma rede privada virtual básica
- Criação e gerenciamento de volumes de armazenamento em bloco
- Introdução aos bancos de dados de grafos
- Conceitos básicos de streaming gerenciado
- Introdução aos gateways de trânsito de rede
- Introdução à transmissão de transporte de vídeo
- Rede privada virtual com servidores privados
- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeAvailabilityZonesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! DescribeAvailabilityZones
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
int AwsDoc::EC2::describeAvailabilityZones(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
 ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "ZoneName" <<
                  std::setw(20) << "State" <<
                  std::setw(32) << "Region" << std::endl;

        const auto &zones =
                outcome.GetResult().GetAvailabilityZones();

        for (const auto &zone: zones) {
            Aws::String stateString =

 Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                            zone.GetState());
            std::cout << std::left <<
                      std::setw(32) << zone.GetZoneName() <<
                      std::setw(20) << stateString <<
                      std::setw(32) << zone.GetRegionName() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe availability zones:" <<
                  outcome.GetError().GetMessage() << std::endl;

    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeAvailabilityZonesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para descrever suas zonas de disponibilidade

O exemplo `describe-availability-zones` a seguir exibe detalhes das zonas de disponibilidade disponíveis para você. A resposta inclui zonas de disponibilidade somente para a região atual. Neste exemplo, ela usa a região padrão dos perfis do `us-west-2` (Oregon).

```
aws ec2 describe-availability-zones
```

Saída:

```
{
    "AvailabilityZones": [
        {
            "State": "available",
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2a",
            "ZoneId": "usw2-az1",
            "GroupName": "us-west-2",
            "NetworkBorderGroup": "us-west-2"
        },
        {
            "State": "available",
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2b",
            "ZoneId": "usw2-az2",
            "GroupName": "us-west-2",
            "NetworkBorderGroup": "us-west-2"
        },
        {
            "State": "available",
```

```
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2c",
            "ZoneId": "usw2-az3",
            "GroupName": "us-west-2",
            "NetworkBorderGroup": "us-west-2"
        },
        {
            "State": "available",
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2d",
            "ZoneId": "usw2-az4",
            "GroupName": "us-west-2",
            "NetworkBorderGroup": "us-west-2"
        },
        {
            "State": "available",
            "OptInStatus": "opted-in",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2-lax-1a",
            "ZoneId": "usw2-lax1-az1",
            "GroupName": "us-west-2-lax-1",
            "NetworkBorderGroup": "us-west-2-lax-1"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeAvailabilityZonesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve as zonas de disponibilidade da região atual que estão disponíveis para você.

```
Get-EC2AvailabilityZone
```

Saída:

```
Messages    RegionName    State        ZoneName
--------    ----------    -----        --------
{}          us-west-2     available    us-west-2a
{}          us-west-2     available    us-west-2b
{}          us-west-2     available    us-west-2c
```

Exemplo 2: esse exemplo descreve todas as zonas de disponibilidade que estão em estado de comprometimento. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- Para obter detalhes da API, consulte DescribeAvailabilityZonesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as zonas de disponibilidade da região atual que estão disponíveis para você.

```
Get-EC2AvailabilityZone
```

Saída:

```
Messages    RegionName    State        ZoneName
--------    ----------    -----        --------
{}          us-west-2     available    us-west-2a
{}          us-west-2     available    us-west-2b
{}          us-west-2     available    us-west-2c
```

Exemplo 2: esse exemplo descreve todas as zonas de disponibilidade que estão em estado de comprometimento. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- Para obter detalhes da API, consulte DescribeAvailabilityZonesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
```

```
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.
```

```
        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones
```

- Para obter detalhes da API, consulte a DescribeAvailabilityZonesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeavailabilityzones( ).
 " oo_result is returned for testing purposes. "
        DATA(lt_zones) = oo_result->get_availabilityzones( ).
        MESSAGE 'Retrieved information about Availability Zones.' TYPE 'I'.

      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAvailabilityZonesreferência da API AWS SDK for SAP ABAP.

Usar **`DescribeBundleTasks`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeBundleTasks`.

CLI

AWS CLI

Descrever suas tarefas de empacotamento

Este exemplo descreve todas as tarefas de empacotamento.

Comando:

```
aws ec2 describe-bundle-tasks
```

Saída:

```
{
  "BundleTasks": [
    {
      "UpdateTime": "2015-09-15T13:26:54.000Z",
      "InstanceId": "i-1234567890abcdef0",
      "Storage": {
        "S3": {
            "Prefix": "winami",
            "Bucket": "bundletasks"
        }
      },
      "State": "bundling",
      "StartTime": "2015-09-15T13:24:35.000Z",
      "Progress": "3%",
      "BundleId": "bun-2a4e041c"
    }
  ]
}
```

- Para obter detalhes da API, consulte DescribeBundleTasksem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a tarefa de empacotamento especificada.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Exemplo 2: esse exemplo descreve as tarefas do pacote cujo estado é “concluído” ou “com falha”.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- Para obter detalhes da API, consulte DescribeBundleTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a tarefa de empacotamento especificada.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Exemplo 2: esse exemplo descreve as tarefas do pacote cujo estado é “concluído” ou “com falha”.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- Para obter detalhes da API, consulte DescribeBundleTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeCapacityReservations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeCapacityReservations`.

CLI

AWS CLI

Exemplo 1: descrever uma ou mais de suas reservas de capacidade

O `describe-capacity-reservations` exemplo a seguir exibe detalhes sobre todas as suas reservas de capacidade na AWS região atual.

```
aws ec2 describe-capacity-reservations
```

Saída:

```
{
    "CapacityReservations": [
        {
            "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
            "OwnerId": "123456789111",
            "CapacityReservationArn": "arn:aws:ec2:us-
east-1:123456789111:capacity-reservation/cr-1234abcd56EXAMPLE",
            "AvailabilityZoneId": "use1-az2",
            "InstanceType": "c5.large",
            "InstancePlatform": "Linux/UNIX",
            "AvailabilityZone": "us-east-1a",
            "Tenancy": "default",
            "TotalInstanceCount": 1,
            "AvailableInstanceCount": 1,
            "EbsOptimized": true,
            "EphemeralStorage": false,
            "State": "active",
            "StartDate": "2024-10-23T15:00:24+00:00",
            "EndDateType": "unlimited",
            "InstanceMatchCriteria": "open",
            "CreateDate": "2024-10-23T15:00:24+00:00",
            "Tags": [],
            "CapacityAllocations": []
        },
        {
            "CapacityReservationId": "cr-abcdEXAMPLE9876ef ",
            "OwnerId": "123456789111",
            "CapacityReservationArn": "arn:aws:ec2:us-
east-1:123456789111:capacity-reservation/cr-abcdEXAMPLE9876ef",
            "AvailabilityZoneId": "use1-az2",
```

```
            "InstanceType": "c4.large",
            "InstancePlatform": "Linux/UNIX",
            "AvailabilityZone": "us-east-1a",
            "Tenancy": "default",
            "TotalInstanceCount": 1,
            "AvailableInstanceCount": 1,
            "EbsOptimized": true,
            "EphemeralStorage": false,
            "State": "cancelled",
            "StartDate": "2024-10-23T15:01:03+00:00",
            "EndDateType": "unlimited",
            "InstanceMatchCriteria": "open",
            "CreateDate": "2024-10-23T15:01:02+00:00",
            "Tags": [],
            "CapacityAllocations": []
        }
    ]
}
```

Exemplo 2: descrever uma ou mais de suas reservas de capacidade

O exemplo `describe-capacity-reservations` a seguir exibe os detalhes da reserva de capacidade especificada.

```
aws ec2 describe-capacity-reservations \
    --capacity-reservation-ids cr-1234abcd56EXAMPLE
```

Saída:

```
{
    "CapacityReservations": [
        {
            "CapacityReservationId": "cr-abcdEXAMPLE9876ef ",
            "OwnerId": "123456789111",
            "CapacityReservationArn": "arn:aws:ec2:us-
east-1:123456789111:capacity-reservation/cr-abcdEXAMPLE9876ef",
            "AvailabilityZoneId": "use1-az2",
            "InstanceType": "c4.large",
            "InstancePlatform": "Linux/UNIX",
            "AvailabilityZone": "us-east-1a",
            "Tenancy": "default",
            "TotalInstanceCount": 1,
```

```
            "AvailableInstanceCount": 1,
            "EbsOptimized": true,
            "EphemeralStorage": false,
            "State": "active",
            "StartDate": "2024-10-23T15:01:03+00:00",
            "EndDateType": "unlimited",
            "InstanceMatchCriteria": "open",
            "CreateDate": "2024-10-23T15:01:02+00:00",
            "Tags": [],
            "CapacityAllocations": []
        }
    ]
}
```

Para obter mais informações, consulte Visualizar uma reserva de capacidade no Guia do usuário do Amazon Elastic Compute Cloud para instâncias Linux.

- Para obter detalhes da API, consulte DescribeCapacityReservationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descrever uma ou mais de suas reservas de capacidade para a região

```
Get-EC2CapacityReservation -Region eu-west-1
```

Saída:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
```

```
State                   : active
Tags                    : {}
Tenancy                 : default
TotalInstanceCount      : 2
```

- Para obter detalhes da API, consulte DescribeCapacityReservationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descrever uma ou mais de suas reservas de capacidade para a região

```
Get-EC2CapacityReservation -Region eu-west-1
```

Saída:

```
AvailabilityZone        : eu-west-1b
AvailableInstanceCount  : 2
CapacityReservationId   : cr-0c1f2345db6f7cdba
CreateDate              : 3/28/2019 9:29:41 AM
EbsOptimized            : True
EndDate                 : 1/1/0001 12:00:00 AM
EndDateType             : unlimited
EphemeralStorage        : False
InstanceMatchCriteria   : open
InstancePlatform        : Windows
InstanceType            : m4.xlarge
State                   : active
Tags                    : {}
Tenancy                 : default
TotalInstanceCount      : 2
```

- Para obter detalhes da API, consulte DescribeCapacityReservationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeCustomerGateways`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeCustomerGateways`.

CLI

AWS CLI

Descrever os gateways do cliente

Este exemplo descreve os gateways do cliente.

Comando:

```
aws ec2 describe-customer-gateways
```

Saída:

```
{
    "CustomerGateways": [
        {
            "CustomerGatewayId": "cgw-b4dc3961",
            "IpAddress": "203.0.113.12",
            "State": "available",
            "Type": "ipsec.1",
            "BgpAsn": "65000"
        },
        {
            "CustomerGatewayId": "cgw-0e11f167",
            "IpAddress": "12.1.2.3",
            "State": "available",
            "Type": "ipsec.1",
            "BgpAsn": "65534"
        }
    ]
}
```

Descrever um gateway do cliente específico

Este exemplo descreve o gateway do cliente especificado.

Comando:

```
aws ec2 describe-customer-gateways --customer-gateway-ids cgw-0e11f167
```

Saída:

```
{
    "CustomerGateways": [
        {
            "CustomerGatewayId": "cgw-0e11f167",
            "IpAddress": "12.1.2.3",
            "State": "available",
            "Type": "ipsec.1",
            "BgpAsn": "65534"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeCustomerGatewaysem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o gateway do cliente especificado.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Saída:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

Exemplo 2: esse exemplo descreve qualquer gateway do cliente cujo estado seja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Exemplo 3: esse exemplo descreve todos os gateways do cliente.

```
Get-EC2CustomerGateway
```

- Para obter detalhes da API, consulte DescribeCustomerGatewaysem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o gateway do cliente especificado.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Saída:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

Exemplo 2: esse exemplo descreve qualquer gateway do cliente cujo estado seja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Exemplo 3: esse exemplo descreve todos os gateways do cliente.

```
Get-EC2CustomerGateway
```

- Para obter detalhes da API, consulte DescribeCustomerGatewaysem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeDhcpOptions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeDhcpOptions`.

CLI

AWS CLI

Exemplo 1: descrever as opções de DHCP

O exemplo `describe-dhcp-options` a seguir recupera detalhes das suas opções DHCP.

```
aws ec2 describe-dhcp-options
```

Saída:

```
{
    "DhcpOptions": [
        {
            "DhcpConfigurations": [
                {
                    "Key": "domain-name",
                    "Values": [
                        {
                            "Value": "us-east-2.compute.internal"
                        }
                    ]
                },
                {
                    "Key": "domain-name-servers",
                    "Values": [
                        {
                            "Value": "AmazonProvidedDNS"
                        }
                    ]
                }
            ],
            "DhcpOptionsId": "dopt-19edf471",
            "OwnerId": "111122223333"
        },
        {
            "DhcpConfigurations": [
                {
                    "Key": "domain-name",
                    "Values": [
                        {
                            "Value": "us-east-2.compute.internal"
```

```
                        }
                    ]
                },
                {
                    "Key": "domain-name-servers",
                    "Values": [
                        {
                            "Value": "AmazonProvidedDNS"
                        }
                    ]
                }
            ],
            "DhcpOptionsId": "dopt-fEXAMPLE",
            "OwnerId": "111122223333"
        }
    ]
}
```

Para obter mais informações, consulte Trabalhar com conjuntos de opções DHCP no Guia do usuário da AWS VPC.

Exemplo 2: descrever suas opções de DHCP e filtrar o resultado

O exemplo `describe-dhcp-options` a seguir descreve as opções de DHCP e usa um filtro para retornar somente as opções de DHCP que têm `example.com` como o servidor de nomes de domínio. O exemplo usa o parâmetro `--query` para exibir somente as informações de configuração e o ID no resultado.

```
aws ec2 describe-dhcp-options \
    --filters Name=key,Values=domain-name-servers Name=value,Values=example.com \
    --query "DhcpOptions[*].[DhcpConfigurations,DhcpOptionsId]"
```

Saída:

```
[
    [
        [
            {
                "Key": "domain-name",
                "Values": [
                    {
                        "Value": "example.com"
```

```
                    }
                ]
            },
            {
                "Key": "domain-name-servers",
                "Values": [
                    {
                        "Value": "172.16.16.16"
                    }
                ]
            }
        ],
        "dopt-001122334455667ab"
    ]
]
```

Para obter mais informações, consulte Trabalhar com conjuntos de opções DHCP no Guia do usuário da AWS VPC.

- Para obter detalhes da API, consulte DescribeDhcpOptionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista seus conjuntos de opções DHCP.

```
Get-EC2DhcpOption
```

Saída:

```
DhcpConfigurations                    DhcpOptionsId    Tag
------------------                    -------------    ---
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
{domain-name-servers}                 dopt-3a4b5c6d    {}
```

Exemplo 2: esse exemplo obtém detalhes de configuração para o conjunto de opções DHCP especificado.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Saída:

```
Key                     Values
---                     ------
domain-name             {abc.local}
domain-name-servers     {10.0.0.101, 10.0.0.102}
```

- Para obter detalhes da API, consulte DescribeDhcpOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista seus conjuntos de opções DHCP.

```
Get-EC2DhcpOption
```

Saída:

```
DhcpConfigurations                    DhcpOptionsId    Tag
------------------                    -------------    ---
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
{domain-name-servers}                 dopt-3a4b5c6d    {}
```

Exemplo 2: esse exemplo obtém detalhes de configuração para o conjunto de opções DHCP especificado.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Saída:

```
Key                     Values
---                     ------
domain-name             {abc.local}
domain-name-servers     {10.0.0.101, 10.0.0.102}
```

- Para obter detalhes da API, consulte DescribeDhcpOptionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeFlowLogs`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeFlowLogs`.

CLI

AWS CLI

Exemplo 1: descrever todos os seus logs de fluxo

O exemplo `describe-flow-logs` a seguir exibe detalhes de todos os logs de fluxo.

```
aws ec2 describe-flow-logs
```

Saída:

```
{
    "FlowLogs": [
        {
            "CreationTime": "2018-02-21T13:22:12.644Z",
            "DeliverLogsPermissionArn": "arn:aws:iam::123456789012:role/flow-
logs-role",
            "DeliverLogsStatus": "SUCCESS",
            "FlowLogId": "fl-aabbccdd112233445",
            "MaxAggregationInterval": 600,
            "FlowLogStatus": "ACTIVE",
            "LogGroupName": "FlowLogGroup",
            "ResourceId": "subnet-12345678901234567",
            "TrafficType": "ALL",
            "LogDestinationType": "cloud-watch-logs",
            "LogFormat": "${version} ${account-id} ${interface-id} ${srcaddr}
 ${dstaddr} ${srcport} ${dstport} ${protocol} ${packets} ${bytes} ${start} ${end}
 ${action} ${log-status}"
        },
        {
            "CreationTime": "2020-02-04T15:22:29.986Z",
            "DeliverLogsStatus": "SUCCESS",
            "FlowLogId": "fl-01234567890123456",
            "MaxAggregationInterval": 60,
            "FlowLogStatus": "ACTIVE",
            "ResourceId": "vpc-00112233445566778",
            "TrafficType": "ACCEPT",
            "LogDestinationType": "s3",
```

```
            "LogDestination": "arn:aws:s3:::my-flow-log-bucket/custom",
            "LogFormat": "${version} ${vpc-id} ${subnet-id} ${instance-id}
 ${interface-id} ${account-id} ${type} ${srcaddr} ${dstaddr} ${srcport}
 ${dstport} ${pkt-srcaddr} ${pkt-dstaddr} ${protocol} ${bytes} ${packets}
 ${start} ${end} ${action} ${tcp-flags} ${log-status}"
        }
    ]
}
```

Exemplo 2: descrever um subconjunto dos seus logs de fluxo

O `describe-flow-logs` exemplo a seguir usa um filtro para exibir detalhes somente dos registros de fluxo que estão no grupo de registros especificado no Amazon CloudWatch Logs.

```
aws ec2 describe-flow-logs \
    --filter "Name=log-group-name,Values=MyFlowLogs"
```

- Para obter detalhes da API, consulte DescribeFlowLogsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve um ou mais logs de fluxo com o tipo de destino de log “s3”

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

Saída:

```
CreationTime             : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage  :
DeliverLogsPermissionArn :
DeliverLogsStatus        : SUCCESS
FlowLogId                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType       : s3
LogGroupName             :
ResourceId               : eni-01d2dda3456b7e890
```

```
TrafficType              : ALL
```

- Para obter detalhes da API, consulte DescribeFlowLogsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve um ou mais logs de fluxo com o tipo de destino de log “s3”

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

Saída:

```
CreationTime             : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage  :
DeliverLogsPermissionArn :
DeliverLogsStatus        : SUCCESS
FlowLogId                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType       : s3
LogGroupName             :
ResourceId               : eni-01d2dda3456b7e890
TrafficType              : ALL
```

- Para obter detalhes da API, consulte DescribeFlowLogsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeHostReservationOfferings`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeHostReservationOfferings`.

CLI

AWS CLI

Descrever as ofertas de reserva de host dedicado

Este exemplo descreve as Reservas de Host Dedicado para a família de instâncias M4 que estão disponíveis para compra.

Comando:

```
aws ec2 describe-host-reservation-offerings --filter Name=instance-
family,Values=m4
```

Saída:

```
{
  "OfferingSet": [
      {
          "HourlyPrice": "1.499",
          "OfferingId": "hro-03f707bf363b6b324",
          "InstanceFamily": "m4",
          "PaymentOption": "NoUpfront",
          "UpfrontPrice": "0.000",
          "Duration": 31536000
      },
      {
          "HourlyPrice": "1.045",
          "OfferingId": "hro-0ef9181cabdef7a02",
          "InstanceFamily": "m4",
          "PaymentOption": "NoUpfront",
          "UpfrontPrice": "0.000",
          "Duration": 94608000
      },
      {
          "HourlyPrice": "0.714",
          "OfferingId": "hro-04567a15500b92a51",
          "InstanceFamily": "m4",
          "PaymentOption": "PartialUpfront",
          "UpfrontPrice": "6254.000",
          "Duration": 31536000
      },
      {
          "HourlyPrice": "0.484",
          "OfferingId": "hro-0d5d7a9d23ed7fbfe",
          "InstanceFamily": "m4",
          "PaymentOption": "PartialUpfront",
          "UpfrontPrice": "12720.000",
          "Duration": 94608000
      },
      {
          "HourlyPrice": "0.000",
          "OfferingId": "hro-05da4108ca998c2e5",
          "InstanceFamily": "m4",
```

```
            "PaymentOption": "AllUpfront",
            "UpfrontPrice": "23913.000",
            "Duration": 94608000
        },
        {
            "HourlyPrice": "0.000",
            "OfferingId": "hro-0a9f9be3b95a3dc8f",
            "InstanceFamily": "m4",
            "PaymentOption": "AllUpfront",
            "UpfrontPrice": "12257.000",
            "Duration": 31536000
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeHostReservationOfferingsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve as reservas de host dedicado que estão disponíveis para compra para o determinado filtro 'instance-family', onde está PaymentOption " NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

Saída:

```
CurrencyCode   :
Duration       : 94608000
HourlyPrice    : 1.307
InstanceFamily : m4
OfferingId     : hro-0c1f234567890d9ab
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000

CurrencyCode   :
Duration       : 31536000
HourlyPrice    : 1.830
InstanceFamily : m4
```

```
OfferingId     : hro-04ad12aaaf34b5a67
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000
```

- Para obter detalhes da API, consulte DescribeHostReservationOfferingsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve as reservas de host dedicado que estão disponíveis para compra para o determinado filtro 'instance-family', onde está PaymentOption " NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

Saída:

```
CurrencyCode   :
Duration       : 94608000
HourlyPrice    : 1.307
InstanceFamily : m4
OfferingId     : hro-0c1f234567890d9ab
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000

CurrencyCode   :
Duration       : 31536000
HourlyPrice    : 1.830
InstanceFamily : m4
OfferingId     : hro-04ad12aaaf34b5a67
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000
```

- Para obter detalhes da API, consulte DescribeHostReservationOfferingsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeHosts`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeHosts`.

CLI

AWS CLI

Visualizar os detalhes de um Host dedicado

O `describe-hosts` exemplo a seguir exibe detalhes dos hosts `available` dedicados em sua AWS conta.

```
aws ec2 describe-hosts --filter "Name=state,Values=available"
```

Saída:

```
{
    "Hosts": [
        {
            "HostId": "h-07879acf49EXAMPLE",
            "Tags": [
                {
                    "Value": "production",
                    "Key": "purpose"
                }
            ],
            "HostProperties": {
                "Cores": 48,
                "TotalVCpus": 96,
                "InstanceType": "m5.large",
                "Sockets": 2
            },
            "Instances": [],
            "State": "available",
            "AvailabilityZone": "eu-west-1a",
            "AvailableCapacity": {
                "AvailableInstanceCapacity": [
                    {
                        "AvailableCapacity": 48,
                        "InstanceType": "m5.large",
                        "TotalCapacity": 48
                    }
                ],
                "AvailableVCpus": 96
            },
            "HostRecovery": "on",
```

```
            "AllocationTime": "2019-08-19T08:57:44.000Z",
            "AutoPlacement": "off"
        }
    ]
}
```

Para obter mais informações, consulte Visualizar Hosts Dedicados no Guia do usuário do Amazon Elastic Compute Cloud para instâncias do Linux.

- Para obter detalhes da API, consulte DescribeHostsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo retorna os detalhes do host do EC2.

```
Get-EC2Host
```

Saída:

```
AllocationTime    : 3/23/2019 4:55:22 PM
AutoPlacement     : off
AvailabilityZone  : eu-west-1b
AvailableCapacity : Amazon.EC2.Model.AvailableCapacity
ClientToken       :
HostId            : h-01e23f4cd567890f1
HostProperties    : Amazon.EC2.Model.HostProperties
HostReservationId :
Instances         : {}
ReleaseTime       : 1/1/0001 12:00:00 AM
State             : available
Tags              : {}
```

Exemplo 2: Este exemplo consulta o host AvailableInstanceCapacity h-01e23f4cd567899f1

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Saída:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- Para obter detalhes da API, consulte DescribeHostsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna os detalhes do host do EC2.

```
Get-EC2Host
```

Saída:

```
AllocationTime    : 3/23/2019 4:55:22 PM
AutoPlacement     : off
AvailabilityZone  : eu-west-1b
AvailableCapacity : Amazon.EC2.Model.AvailableCapacity
ClientToken       :
HostId            : h-01e23f4cd567890f1
HostProperties    : Amazon.EC2.Model.HostProperties
HostReservationId :
Instances         : {}
ReleaseTime       : 1/1/0001 12:00:00 AM
State             : available
Tags              : {}
```

Exemplo 2: Este exemplo consulta o host AvailableInstanceCapacity h-01e23f4cd567899f1

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Saída:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- Para obter detalhes da API, consulte DescribeHostsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeIamInstanceProfileAssociations`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeIamInstanceProfileAssociations`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeIamInstanceProfileAssociationsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para descrever as associações do perfil de instância do IAM

Este exemplo descreve todas as suas associações de perfil de instância do IAM.

Comando:

```
aws ec2 describe-iam-instance-profile-associations
```

Saída:

```
{
  "IamInstanceProfileAssociations": [
      {
          "InstanceId": "i-09eb09efa73ec1dee",
          "State": "associated",
          "AssociationId": "iip-assoc-0db249b1f25fa24b8",
          "IamInstanceProfile": {
              "Id": "AIPAJVQN4F5WVLGCJDRGM",
```

```
                "Arn": "arn:aws:iam::123456789012:instance-profile/admin-role"
            }
        },
        {
            "InstanceId": "i-0402909a2f4dffd14",
            "State": "associating",
            "AssociationId": "iip-assoc-0d1ec06278d29f44a",
            "IamInstanceProfile": {
                "Id": "AGJAJVQN4F5WVLGCJABCM",
                "Arn": "arn:aws:iam::123456789012:instance-profile/user1-role"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeIamInstanceProfileAssociationsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const ec2Client = new EC2Client({});
const { IamInstanceProfileAssociations } = await ec2Client.send(
  new DescribeIamInstanceProfileAssociationsCommand({
    Filters: [
      { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
    ],
  }),
);
```

- Para obter detalhes da API, consulte DescribeIamInstanceProfileAssociationsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```

```
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
```

```
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DescribeIamInstanceProfileAssociationsReferência da API AWS SDK for Python (Boto3).

Usar **`DescribeIdFormat`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeIdFormat`.

CLI

AWS CLI

Exemplo 1: descrever o formato de ID de um recurso

O exemplo `describe-id-format` a seguir descreve o formato de ID para grupos de segurança.

```
aws ec2 describe-id-format \
    --resource security-group
```

No exemplo de saída a seguir, o valor `Deadline` indica que o prazo para esse tipo de recurso mudar permanentemente do formato de ID curto para o formato de ID longo expirou às 00:00 UTC de 15 de agosto de 2018.

```
{
    "Statuses": [
        {
            "Deadline": "2018-08-15T00:00:00.000Z",
            "Resource": "security-group",
            "UseLongIds": true
        }
    ]
}
```

Exemplo 2: descrever o formato de ID de todos os recursos

O exemplo `describe-id-format` a seguir descreve o formato de ID para todos os tipos de recursos. Todos os tipos de recursos compatíveis com o formato de ID curto foram alterados para o formato de ID longo.

```
aws ec2 describe-id-format
```

- Para obter detalhes da API, consulte DescribeIdFormatem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o formato de ID para o tipo de recurso especificado.

```
Get-EC2IdFormat -Resource instance
```

Saída:

```
Resource       UseLongIds
--------       ----------
instance       False
```

Exemplo 2: esse exemplo descreve os formatos de ID para todos os tipos de recursos que oferecem suporte a IDs mais longos.

```
Get-EC2IdFormat
```

Saída:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- Para obter detalhes da API, consulte DescribeIdFormatem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o formato de ID para o tipo de recurso especificado.

```
Get-EC2IdFormat -Resource instance
```

Saída:

```
Resource       UseLongIds
--------       ----------
instance       False
```

Exemplo 2: esse exemplo descreve os formatos de ID para todos os tipos de recursos que oferecem suporte a IDs mais longos.

```
Get-EC2IdFormat
```

Saída:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- Para obter detalhes da API, consulte DescribeIdFormatem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeIdentityIdFormat`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeIdentityIdFormat`.

CLI

AWS CLI

Descrever o formato de ID de um perfil do IAM

O `describe-identity-id-format` exemplo a seguir descreve o formato de ID recebido pelas instâncias criadas pela função do IAM `EC2Role` em sua AWS conta.

```
aws ec2 describe-identity-id-format \
    --principal-arn arn:aws:iam::123456789012:role/my-iam-role \
```

```
    --resource instance
```

O resultado a seguir indica que as instâncias criadas por essa função recebem IDs no formato de ID longo.

```
{
    "Statuses": [
        {
            "Deadline": "2016-12-15T00:00:00Z",
            "Resource": "instance",
            "UseLongIds": true
        }
    ]
}
```

Descrever o formato de ID de um usuário do IAM

O `describe-identity-id-format` exemplo a seguir descreve o formato de ID recebido pelos snapshots criados pelo usuário do IAM `AdminUser` em sua AWS conta.

```
aws ec2 describe-identity-id-format \
    --principal-arn arn:aws:iam::123456789012:user/AdminUser \
    --resource snapshot
```

O resultado indica que os snapshots criados por esse usuário recebem IDs no formato de ID longo.

```
{
    "Statuses": [
        {
            "Deadline": "2016-12-15T00:00:00Z",
            "Resource": "snapshot",
            "UseLongIds": true
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeIdentityIdFormatem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo retorna o formato de ID do recurso “image” para a função especificada

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image
```

Saída:

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- Para obter detalhes da API, consulte DescribeIdentityIdFormatem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o formato de ID do recurso “image” para a função especificada

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image
```

Saída:

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- Para obter detalhes da API, consulte DescribeIdentityIdFormatem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeImageAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeImageAttribute`.

CLI

AWS CLI

Descrever as permissões de lançamento de uma AMI

Este exemplo descreve as permissões de lançamento da AMI especificada.

Comando:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --
attribute launchPermission
```

Saída:

```
{
    "LaunchPermissions": [
        {
            "UserId": "123456789012"
        }
    ],
    "ImageId": "ami-5731123e",
}
```

Descrever os códigos de produto de uma AMI

Este exemplo descreve os códigos de produto para a AMI especificada. Observe que essa AMI não tem códigos de produto.

Comando:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute productCodes
```

Saída:

```
{
    "ProductCodes": [],
    "ImageId": "ami-5731123e",
}
```

- Para obter detalhes da API, consulte DescribeImageAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém a descrição da AMI especificada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Saída:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Exemplo 2: esse exemplo obtém as permissões de lançamento da AMI especificada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Saída:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Exemplo 3: esse exemplo testa se as redes aprimoradas estão habilitadas.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Saída:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     : simple
```

- Para obter detalhes da API, consulte DescribeImageAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém a descrição da AMI especificada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Saída:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Exemplo 2: esse exemplo obtém as permissões de lançamento da AMI especificada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Saída:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
```

```
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Exemplo 3: esse exemplo testa se as redes aprimoradas estão habilitadas.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Saída:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     : simple
```

- Para obter detalhes da API, consulte DescribeImageAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeImages`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeImages`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma rede privada virtual básica
- Criação e gerenciamento de volumes de armazenamento em bloco
- Conceitos básicos de compras do mercado de software
- Conceitos básicos de streaming gerenciado
- Introdução às máquinas virtuais
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Rede privada virtual com servidores privados

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images.
#
# Parameters:
#       -i image_ids - A space-separated  list of image IDs (optional).
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_images() {
  local image_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_images"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) image_ids="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
  fi

  response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeImagesem Referência de AWS CLI Comandos.

CLI

AWS CLI

Exemplo 1: descrever uma AMI

O exemplo `describe-images` a seguir descreve a AMI especificada na região especificada.

```
aws ec2 describe-images \
    --region us-east-1 \
    --image-ids ami-1234567890EXAMPLE
```

Saída:

```
{
    "Images": [
        {
            "VirtualizationType": "hvm",
            "Description": "Provided by Red Hat, Inc.",
            "PlatformDetails": "Red Hat Enterprise Linux",
            "EnaSupport": true,
            "Hypervisor": "xen",
            "State": "available",
            "SriovNetSupport": "simple",
            "ImageId": "ami-1234567890EXAMPLE",
            "UsageOperation": "RunInstances:0010",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "SnapshotId": "snap-111222333444aaabb",
                        "DeleteOnTermination": true,
                        "VolumeType": "gp2",
                        "VolumeSize": 10,
                        "Encrypted": false
                    }
                }
            ],
            "Architecture": "x86_64",
            "ImageLocation": "123456789012/RHEL-8.0.0_HVM-20190618-x86_64-1-
Hourly2-GP2",
            "RootDeviceType": "ebs",
            "OwnerId": "123456789012",
```

```
            "RootDeviceName": "/dev/sda1",
            "CreationDate": "2019-05-10T13:17:12.000Z",
            "Public": true,
            "ImageType": "machine",
            "Name": "RHEL-8.0.0_HVM-20190618-x86_64-1-Hourly2-GP2"
        }
    ]
}
```

Para obter mais informações, consulte Imagens de máquina da Amazon (AMIs) no Guia do usuário do Amazon EC2.

Exemplo 2: descrever AMIs com base em filtros

O exemplo `describe-images` a seguir descreve as AMIs do Windows fornecidas pela Amazon com o Amazon EBS.

```
aws ec2 describe-images \
    --owners amazon \
    --filters "Name=platform,Values=windows" "Name=root-device-type,Values=ebs"
```

Para obter um exemplo da saída de `describe-images`, consulte o Exemplo 1.

Para obter mais exemplos do uso de filtros, consulte Listar e filtrar seus recursos no Guia do usuário do Amazon EC2.

Exemplo 3: descrever AMIs com base em tags

O exemplo `describe-images` a seguir descreve todas as AMIs que têm a tag `Type=Custom`. O exemplo usa o parâmetro `--query` para exibir somente os IDs da AMI.

```
aws ec2 describe-images \
    --filters "Name=tag:Type,Values=Custom" \
    --query 'Images[*].[ImageId]' \
    --output text
```

Saída:

```
ami-1234567890EXAMPLE
ami-0abcdef1234567890
```

Para obter mais exemplos do uso de filtros de tags, consulte Trabalhando com tags no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeImagesem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, paginateDescribeImages } from "@aws-sdk/client-ec2";

/**
 * Describes the specified images (AMIs, AKIs, and ARIs) available to you or all
 of the images available to you.
 * @param {{ architecture: string, pageSize: number }} options
 */
export const main = async ({ architecture, pageSize }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the base command.
  const paginator = paginateDescribeImages(
    // Without limiting the page size, this call can take a long time. pageSize
 is just sugar for
    // the MaxResults property in the base command.
    { client, pageSize },
    {
      // There are almost 70,000 images available. Be specific with your
 filtering
      // to increase efficiency.
      // See https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/
client-ec2/interfaces/describeimagescommandinput.html#filters
      Filters: [{ Name: "architecture", Values: [architecture] }],
    },
  );
```

```
  /**
   * @type {import('@aws-sdk/client-ec2').Image[]}
   */
  const images = [];
  let recordsScanned = 0;

  try {
    for await (const page of paginator) {
      recordsScanned += pageSize;
      if (page.Images.length) {
        images.push(...page.Images);
        break;
      }
      console.log(
        `No matching image found yet. Searched ${recordsScanned} records.`,
      );
    }

    if (images.length) {
      console.log(
        `Found ${images.length} images:\n\n${images.map((image) =>
 image.Name).join("\n")}\n`,
      );
    } else {
      console.log(
        `No matching images found. Searched ${recordsScanned} records.\n`,
      );
    }

    return images;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
      return [];
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte DescribeImagesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a AMI especificada.

```
Get-EC2Image -ImageId ami-12345678
```

Saída:

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/xvda}
CreationDate        : 2014-10-20T00:56:28.000Z
Description         : My image
Hypervisor          : xen
ImageId             : ami-12345678
ImageLocation       : 123456789012/my-image
ImageOwnerAlias     :
ImageType           : machine
KernelId            :
Name                : my-image
OwnerId             : 123456789012
Platform            :
ProductCodes        : {}
Public              : False
RamdiskId           :
RootDeviceName      : /dev/xvda
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {Name}
VirtualizationType  : hvm
```

Exemplo 2: esse exemplo descreve as AMIs que você possui.

```
Get-EC2Image -owner self
```

Exemplo 3: esse exemplo descreve as AMIs públicas que executam o Microsoft Windows Server.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Exemplo 4: esse exemplo descreve todas as AMIs públicas na região “us-west-2”.

```
Get-EC2Image -Region us-west-2
```

- Para obter detalhes da API, consulte DescribeImagesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a AMI especificada.

```
Get-EC2Image -ImageId ami-12345678
```

Saída:

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/xvda}
CreationDate        : 2014-10-20T00:56:28.000Z
Description         : My image
Hypervisor          : xen
ImageId             : ami-12345678
ImageLocation       : 123456789012/my-image
ImageOwnerAlias     :
ImageType           : machine
KernelId            :
Name                : my-image
OwnerId             : 123456789012
Platform            :
ProductCodes        : {}
Public              : False
RamdiskId           :
RootDeviceName      : /dev/xvda
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {Name}
VirtualizationType  : hvm
```

Exemplo 2: esse exemplo descreve as AMIs que você possui.

```
Get-EC2Image -owner self
```

Exemplo 3: esse exemplo descreve as AMIs públicas que executam o Microsoft Windows Server.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Exemplo 4: esse exemplo descreve todas as AMIs públicas na região “us-west-2”.

```
Get-EC2Image -Region us-west-2
```

- Para obter detalhes da API, consulte DescribeImagesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
```

```
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str,
image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images
```

- Para obter detalhes da API, consulte a DescribeImagesReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
       let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
       let output = self
           .client
           .describe_images()
           .set_image_ids(Some(image_ids))
           .send()
           .await?;

       let images = output.images.unwrap_or_default();
       if images.is_empty() {
           Err(EC2Error::new("No images for selected AMIs"))
       } else {
           Ok(images)
       }
   }
```

Usar a função list_images com o SSM para limitar com base em seu ambiente. Para obter mais detalhes sobre o SSM, consulte https://docs.aws.amazon.com/systems-manager/latest/userguide/example_ssm_GetParameters_section.html.

```
   async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
       let params: Vec<Parameter> = self
           .ssm
           .list_path("/aws/service/ami-amazon-linux-latest")
           .await
           .map_err(|e| e.add_message("Could not find parameters for available
images"))?
           .into_iter()
```

```
            .filter(|param| param.name().is_some_and(|name|
 name.contains("amzn2")))
            .collect();
        let amzn2_images: Vec<ScenarioImage> = self
            .ec2
            .list_images(params)
            .await
            .map_err(|e| e.add_message("Could not find images"))?
            .into_iter()
            .map(ScenarioImage::from)
            .collect();
        println!("We will now create an instance from an Amazon Linux 2 AMI");
        let ami = self.util.select_scenario_image(amzn2_images)?;
        Ok(ami)
    }
```

- Para obter detalhes da API, consulte a DescribeImagesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeimages( it_imageids = it_image_ids ).
     " oo_result is returned for testing purposes. "
        DATA(lt_images) = oo_result->get_images( ).
        MESSAGE 'Retrieved information about Amazon Machine Images (AMIs).' TYPE
 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeImagesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Return an array of `EC2ClientTypes.Image` objects describing all of
    /// the images in the specified array.
    ///
    /// - Parameter idList: A list of image ID strings indicating the images
    ///   to return details about.
    ///
    /// - Returns: An array of the images.
    func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
        do {
            let output = try await ec2Client.describeImages(
                input: DescribeImagesInput(
                    imageIds: idList
                )
            )

            guard let images = output.images else {
                print("*** No images found.")
                return []
            }

            for image in images {
                guard let id = image.imageId else {
                    continue
                }
```

```
                print("   \(id): \(image.description ?? "<no description>")")
            }

            return images
        } catch {
            print("*** Error getting image descriptions:
\(error.localizedDescription)")
            return []
        }
    }
```

- Para obter detalhes da API, consulte DescribeImagesa referência da API AWS SDK for Swift.

Usar **`DescribeImportImageTasks`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeImportImageTasks`.

CLI

AWS CLI

Monitorar uma tarefa de importação de imagem

O exemplo `describe-import-image-tasks` a seguir verifica o status da tarefa de importação de imagem especificada.

```
aws ec2 describe-import-image-tasks \
    --import-task-ids import-ami-1234567890abcdef0
```

Resultado para uma tarefa de importação de imagem que está em andamento.

```
{
    "ImportImageTasks": [
        {
            "ImportTaskId": "import-ami-1234567890abcdef0",
            "Progress": "28",
            "SnapshotDetails": [
                {
                    "DiskImageSize": 705638400.0,
```

```
                    "Format": "ova",
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "my-import-bucket",
                        "S3Key": "vms/my-server-vm.ova"
                    }
                }
            ],
            "Status": "active",
            "StatusMessage": "converting"
        }
    ]
}
```

Resultado para uma tarefa de importação de imagem concluída. O ID da AMI resultante é fornecido por `ImageId`.

```
{
    "ImportImageTasks": [
        {
            "ImportTaskId": "import-ami-1234567890abcdef0",
            "ImageId": "ami-1234567890abcdef0",
            "SnapshotDetails": [
                {
                    "DiskImageSize": 705638400.0,
                    "Format": "ova",
                    "SnapshotId": "snap-1234567890abcdef0"
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "my-import-bucket",
                        "S3Key": "vms/my-server-vm.ova"
                    }
                }
            ],
            "Status": "completed"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeImportImageTasksem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a tarefa de importação de imagem especificada.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Saída:

```
Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

Exemplo 2: esse exemplo descreve todas as tarefas de importação de imagens.

```
Get-EC2ImportImageTask
```

Saída:

```
Architecture    :
Description     : Windows Image 1
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {}
Status          : deleted
StatusMessage   : User initiated task cancelation

Architecture    : x86_64
```

```
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

- Para obter detalhes da API, consulte DescribeImportImageTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a tarefa de importação de imagem especificada.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Saída:

```
Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

Exemplo 2: esse exemplo descreve todas as tarefas de importação de imagens.

```
Get-EC2ImportImageTask
```

Saída:

```
Architecture    :
Description     : Windows Image 1
```

```
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {}
Status          : deleted
StatusMessage   : User initiated task cancelation

Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

- Para obter detalhes da API, consulte DescribeImportImageTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeImportSnapshotTasks`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeImportSnapshotTasks`.

CLI

AWS CLI

Monitorar uma tarefa de importação de snapshots

O exemplo `describe-import-snapshot-tasks` a seguir verifica o status da tarefa de importação instantânea especificada.

```
aws ec2 describe-import-snapshot-tasks \
    --import-task-ids import-snap-1234567890abcdef0
```

Resultado de uma tarefa de captura de snapshot de importação que está em andamento:

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "My server VMDK",
            "ImportTaskId": "import-snap-1234567890abcdef0",
            "SnapshotTaskDetail": {
                "Description": "My server VMDK",
                "DiskImageSize": "705638400.0",
                "Format": "VMDK",
                "Progress": "42",
                "Status": "active",
                "StatusMessage": "downloading/converting",
                "UserBucket": {
                    "S3Bucket": "my-import-bucket",
                    "S3Key": "vms/my-server-vm.vmdk"
                }
            }
        }
    ]
}
```

Resultado para uma tarefa de importação de snapshot concluída. O ID do snapshot resultante é fornecido por `SnapshotId`.

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "My server VMDK",
            "ImportTaskId": "import-snap-1234567890abcdef0",
            "SnapshotTaskDetail": {
                "Description": "My server VMDK",
                "DiskImageSize": "705638400.0",
                "Format": "VMDK",
                "SnapshotId": "snap-1234567890abcdef0"
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "my-import-bucket",
                    "S3Key": "vms/my-server-vm.vmdk"
                }
            }
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte DescribeImportSnapshotTasksem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a tarefa de importação de snapshot especificada.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Saída:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

Exemplo 2: esse exemplo descreve todas as tarefas de importação de snapshots.

```
Get-EC2ImportSnapshotTask
```

Saída:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2     import-snap-hgfedcba
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Para obter detalhes da API, consulte DescribeImportSnapshotTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a tarefa de importação de snapshot especificada.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Saída:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

Exemplo 2: esse exemplo descreve todas as tarefas de importação de snapshots.

```
Get-EC2ImportSnapshotTask
```

Saída:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2     import-snap-hgfedcba
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Para obter detalhes da API, consulte DescribeImportSnapshotTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeInstanceAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstanceAttribute`.

CLI

AWS CLI

Descrever o tipo de instância

Este exemplo descreve o tipo de instância da instância especificada.

Comando:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --
attribute instanceType
```

Saída:

```
{
    "InstanceId": "i-1234567890abcdef0"
    "InstanceType": {
        "Value": "t1.micro"
    }
}
```

Para descrever o ApiTermination atributo de desativação

Este exemplo descreve o atributo `disableApiTermination` da instância especificada.

Comando:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --
attribute disableApiTermination
```

Saída:

```
{
"InstanceId": "i-1234567890abcdef0"
    "DisableApiTermination": {
        "Value": "false"
    }
}
```

Descrever um mapeamento de dispositivos de blocos para uma instância

Este exemplo descreve o atributo `blockDeviceMapping` da instância especificada.

Comando:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --
attribute blockDeviceMapping
```

Saída:

```
{
    "InstanceId": "i-1234567890abcdef0"
    "BlockDeviceMappings": [
        {
            "DeviceName": "/dev/sda1",
            "Ebs": {
                "Status": "attached",
                "DeleteOnTermination": true,
                "VolumeId": "vol-049df61146c4d7901",
                "AttachTime": "2013-05-17T22:42:34.000Z"
            }
        },
        {
            "DeviceName": "/dev/sdf",
            "Ebs": {
                "Status": "attached",
                "DeleteOnTermination": false,
                "VolumeId": "vol-049df61146c4d7901",
                "AttachTime": "2013-09-10T23:07:00.000Z"
            }
        }
    ],
}
```

- Para obter detalhes da API, consulte DescribeInstanceAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o tipo de instância da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Saída:

```
InstanceType                     : t2.micro
```

Exemplo 2: esse exemplo descreve se a rede avançada está habilitada para a instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Saída:

```
SriovNetSupport                  : simple
```

Exemplo 3: esse exemplo descreve os grupos de segurança para a instância especificada.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Saída:

```
GroupId
-------
sg-12345678
sg-45678901
```

Exemplo 4: esse exemplo descreve se a otimização do EBS está habilitada para a instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Saída:

```
EbsOptimized                     : False
```

Exemplo 5: Esse exemplo descreve o atributo “desabilitarApiTermination” da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Saída:

```
DisableApiTermination             : False
```

Exemplo 6: Esse exemplo descreve o atributo “instânciaInitiatedShutdownBehavior” da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

Saída:

```
InstanceInitiatedShutdownBehavior : stop
```

- Para obter detalhes da API, consulte DescribeInstanceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o tipo de instância da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Saída:

```
InstanceType                      : t2.micro
```

Exemplo 2: esse exemplo descreve se a rede avançada está habilitada para a instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Saída:

```
SriovNetSupport                   : simple
```

Exemplo 3: esse exemplo descreve os grupos de segurança para a instância especificada.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Saída:

```
GroupId
-------
sg-12345678
sg-45678901
```

Exemplo 4: esse exemplo descreve se a otimização do EBS está habilitada para a instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Saída:

```
EbsOptimized                      : False
```

Exemplo 5: Esse exemplo descreve o atributo “desabilitarApiTermination” da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Saída:

```
DisableApiTermination             : False
```

Exemplo 6: Esse exemplo descreve o atributo “instânciaInitiatedShutdownBehavior” da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

Saída:

```
InstanceInitiatedShutdownBehavior : stop
```

- Para obter detalhes da API, consulte DescribeInstanceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeInstanceStatus`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstanceStatus`.

CLI

AWS CLI

Para descrever o status de uma instância

O exemplo `describe-instance-status` a seguir descreve o status atual da instância especificada.

```
aws ec2 describe-instance-status \
    --instance-ids i-1234567890abcdef0
```

Saída:

```
{
    "InstanceStatuses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "InstanceState": {
                "Code": 16,
                "Name": "running"
            },
            "AvailabilityZone": "us-east-1d",
            "SystemStatus": {
                "Status": "ok",
                "Details": [
                    {
                        "Status": "passed",
                        "Name": "reachability"
                    }
                ]
            },
            "InstanceStatus": {
                "Status": "ok",
                "Details": [
```

```
                    {
                        "Status": "passed",
                        "Name": "reachability"
                    }
                ]
            }
        }
    ]
}
```

Para obter mais informações, consulte Monitorar o status das instâncias no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeInstanceStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o status da instância especificada.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Saída:

```
AvailabilityZone : us-west-2a
Events           : {}
InstanceId       : i-12345678
InstanceState    : Amazon.EC2.Model.InstanceState
Status           : Amazon.EC2.Model.InstanceStatusSummary
SystemStatus     : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

Saída:

```
Code    Name
----    ----
```

```
16      running
```

```
$status.Status
```

Saída:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

Saída:

```
Details           Status
-------           ------
{reachability}    ok
```

- Para obter detalhes da API, consulte DescribeInstanceStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o status da instância especificada.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Saída:

```
AvailabilityZone : us-west-2a
Events           : {}
InstanceId       : i-12345678
InstanceState    : Amazon.EC2.Model.InstanceState
Status           : Amazon.EC2.Model.InstanceStatusSummary
SystemStatus     : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

Saída:

```
Code     Name
----     ----
16       running
```

```
$status.Status
```

Saída:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

Saída:

```
Details           Status
-------           ------
{reachability}    ok
```

- Para obter detalhes da API, consulte DescribeInstanceStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_all_events(client: &Client) -> Result<(), Error> {
    let resp = client.describe_regions().send().await.unwrap();
```

```
    for region in resp.regions.unwrap_or_default() {
        let reg: &'static str =
 Box::leak(Box::from(region.region_name().unwrap()));
        let region_provider =
 RegionProviderChain::default_provider().or_else(reg);
        let config = aws_config::from_env().region(region_provider).load().await;
        let new_client = Client::new(&config);

        let resp = new_client.describe_instance_status().send().await;

        println!("Instances in region {}:", reg);
        println!();

        for status in resp.unwrap().instance_statuses() {
            println!(
                "  Events scheduled for instance ID: {}",
                status.instance_id().unwrap_or_default()
            );
            for event in status.events() {
                println!("    Event ID:     {}",
 event.instance_event_id().unwrap());
                println!("    Description:  {}", event.description().unwrap());
                println!("    Event code:   {}", event.code().unwrap().as_ref());
                println!();
            }
        }
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeInstanceStatusreferência da API AWS SDK for Rust.

Use **`DescribeInstanceTypes`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstanceTypes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Describe the instance types available.
   /// </summary>
   /// <returns>A list of instance type information.</returns>
   public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
   {
       try
       {
           var request = new DescribeInstanceTypesRequest();

           var filters = new List<Filter>
           {
               new Filter("processor-info.supported-architecture",
                   new List<string> { architecture.ToString() })
           };
           filters.Add(new Filter("instance-type", new() { "*.micro",
"*.small" }));

           request.Filters = filters;
           var instanceTypes = new List<InstanceTypeInfo>();

           var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
           await foreach (var instanceType in paginator.InstanceTypes)
           {
               instanceTypes.Add(instanceType);
           }

           return instanceTypes;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Parameters are invalid. Ensure architecture and size
 strings conform to DescribeInstanceTypes API reference.");
            }

            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
 {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeInstanceTypesa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g.,
 x86_64)
# -t, --type INSTANCE_TYPE         Comma-separated list of instance types (e.g.,
 t2.micro)
```

```
# -h, --help                      Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
##############################################################################
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""

  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-
t|--type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture
 (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE         Comma-separated list of instance
 types (e.g., t2.micro)"
    echo "  -h, --help                       Show this help message"
  }

  while [[ $# -gt 0 ]]; do
    case "$1" in
      -a | --architecture)
        architecture="$2"
        shift 2
        ;;
      -t | --type)
        instance_types="$2"
        shift 2
        ;;
      -h | --help)
        usage
        return 0
        ;;
      *)
        echo "Unknown argument: $1"
        return 1
        ;;
    esac
  done

  if [[ -z "$architecture" ]]; then
    errecho "Error: Architecture not specified."
    usage
```

```
    return 1
  fi

  if [[ -z "$instance_types" ]]; then
    errecho "Error: Instance type not specified."
    usage
    return 1
  fi

  local tmp_json_file="temp_ec2.json"
  echo -n '[
    {
      "Name": "processor-info.supported-architecture",
      "Values": [' >"$tmp_json_file"

  local items
  IFS=',' read -ra items <<<"$architecture"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done
  echo -n ']},
    {
    "Name": "instance-type",
      "Values": [' >>"$tmp_json_file"
  IFS=',' read -ra items <<<"$instance_types"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done

  echo -n ']}]' >>"$tmp_json_file"

  local response
  response=$(aws ec2 describe-instance-types --filters file://"$tmp_json_file" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)
```

```
  local error_code=$?

  rm "$tmp_json_file"

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
    return 1
  fi

  echo "$response"
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
```

```
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeInstanceTypesem Referência de AWS CLI Comandos.

CLI

AWS CLI

Exemplo 1: descrever um tipo de instância

O exemplo `describe-instance-types` a seguir exibe os detalhes do tipo da instância especificado.

```
aws ec2 describe-instance-types \
    --instance-types t2.micro
```

Saída:

```
{
    "InstanceTypes": [
        {
            "InstanceType": "t2.micro",
            "CurrentGeneration": true,
            "FreeTierEligible": true,
```

```
            "SupportedUsageClasses": [
                "on-demand",
                "spot"
            ],
            "SupportedRootDeviceTypes": [
                "ebs"
            ],
            "BareMetal": false,
            "Hypervisor": "xen",
            "ProcessorInfo": {
                "SupportedArchitectures": [
                    "i386",
                    "x86_64"
                ],
                "SustainedClockSpeedInGhz": 2.5
            },
            "VCpuInfo": {
                "DefaultVCpus": 1,
                "DefaultCores": 1,
                "DefaultThreadsPerCore": 1,
                "ValidCores": [
                    1
                ],
                "ValidThreadsPerCore": [
                    1
                ]
            },
            "MemoryInfo": {
                "SizeInMiB": 1024
            },
            "InstanceStorageSupported": false,
            "EbsInfo": {
                "EbsOptimizedSupport": "unsupported",
                "EncryptionSupport": "supported"
            },
            "NetworkInfo": {
                "NetworkPerformance": "Low to Moderate",
                "MaximumNetworkInterfaces": 2,
                "Ipv4AddressesPerInterface": 2,
                "Ipv6AddressesPerInterface": 2,
                "Ipv6Supported": true,
                "EnaSupport": "unsupported"
            },
            "PlacementGroupInfo": {
```

```
                "SupportedStrategies": [
                    "partition",
                    "spread"
                ]
            },
            "HibernationSupported": false,
            "BurstablePerformanceSupported": true,
            "DedicatedHostsSupported": false,
            "AutoRecoverySupported": true
        }
    ]
}
```

Para obter mais informações, consulte Tipos de instância no Guia do usuário do Amazon Elastic Compute Cloud para Instâncias Linux.

Exemplo 2: filtrar os tipos de instância disponíveis

Você pode especificar um filtro para definir o escopo dos resultados para os tipos de instância que tenham uma característica específica. O exemplo `describe-instance-types` a seguir lista os tipos de instância compatíveis com hibernação.

```
aws ec2 describe-instance-types \
    --filters Name=hibernation-supported,Values=true --query
 'InstanceTypes[*].InstanceType'
```

Saída:

```
[
    "m5.8xlarge",
    "r3.large",
    "c3.8xlarge",
    "r5.large",
    "m4.4xlarge",
    "c4.large",
    "m5.xlarge",
    "m4.xlarge",
    "c3.large",
    "c4.8xlarge",
    "c4.4xlarge",
    "c5.xlarge",
    "c5.12xlarge",
```

```
    "r5.4xlarge",
    "c5.4xlarge"
]
```

Para obter mais informações, consulte Tipos de instância no Guia do usuário do Amazon Elastic Compute Cloud para Instâncias Linux.

- Para obter detalhes da API, consulte DescribeInstanceTypesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it
returns a
     * {@link CompletableFuture} that resolves to the instance type string for
the "t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge"
instance type is not
     * found, an empty string is returned.
     * </p>
     *
     * @return a {@link CompletableFuture} that resolves to the instance type
string for the
     * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
```

```
     */
    public CompletableFuture<String> getInstanceTypesAsync() {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
                for (InstanceTypeInfo type : instanceTypes) {
                    logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                    logger.info("Network information is " +
type.networkInfo().toString());
                    logger.info("Instance type is " +
type.instanceType().toString());
                }
            } else {
                throw (RuntimeException) ex;
            }
        });

        return response.thenApply(resp -> {
            for (InstanceTypeInfo type : resp.instanceTypes()) {
                String instanceType = type.instanceType().toString();
                if (instanceType.equals("t2.2xlarge")) {
                    return instanceType;
                }
            }
            return "";
        });
    }
```

- Para obter detalhes da API, consulte DescribeInstanceTypesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, paginateDescribeInstanceTypes } from "@aws-sdk/client-ec2";

/**
 * Describes the specified instance types. By default, all instance types for the
 * current Region are described. Alternatively, you can filter the results.
 * @param {{ pageSize: string, supportedArch: string[], freeTier: boolean }}
 options
 */
export const main = async ({ pageSize, supportedArch, freeTier }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the underlying command.
  const paginator = paginateDescribeInstanceTypes(
    // Without limiting the page size, this call can take a long time. pageSize
 is just sugar for
    // the MaxResults property in the underlying command.
    { client, pageSize },
    {
      Filters: [
        {
          Name: "processor-info.supported-architecture",
          Values: supportedArch,
        },
        { Name: "free-tier-eligible", Values: [freeTier ? "true" : "false"] },
      ],
    },
  );

  try {
    /**
     * @type {import('@aws-sdk/client-ec2').InstanceTypeInfo[]}
```

```
     */
    const instanceTypes = [];

    for await (const page of paginator) {
      if (page.InstanceTypes.length) {
        instanceTypes.push(...page.InstanceTypes);

        // When we have at least 1 result, we can stop.
        if (instanceTypes.length >= 1) {
          break;
        }
      }
    }
    console.log(
      `Memory size in MiB for matching instance types:\n\n
${instanceTypes.map((it) => `${it.InstanceType}: ${it.MemoryInfo.SizeInMiB}
 MiB`).join("\n")}`,
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
      return [];
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte DescribeInstanceTypesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}
```

- Para obter detalhes da API, consulte a DescribeInstanceTypesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
 "*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
API_DescribeInstanceTypes.html
        for a list of allowable parameters.
```

```
        :param architecture: The architecture supported by instance types.
Default: 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support
the specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture},
{','.join(map(str, sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "
                    "Ensure architecture and size strings conform to
DescribeInstanceTypes API reference."
                )
            raise
        else:
            return inst_types
```

- Para obter detalhes da API, consulte a DescribeInstanceTypesReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// List instance types that match an image's architecture and are free tier
eligible.
   pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
       let architecture = format!(
           "{}",
           image.architecture().ok_or_else(|| EC2Error::new(format!(
               "Image {:?} does not have a listed architecture",
               image.image_id()
           )))?
       );
       let free_tier_eligible_filter = Filter::builder()
           .name("free-tier-eligible")
           .values("false")
           .build();
       let supported_architecture_filter = Filter::builder()
           .name("processor-info.supported-architecture")
           .values(architecture)
           .build();
       let response = self
           .client
           .describe_instance_types()
           .filters(free_tier_eligible_filter)
           .filters(supported_architecture_filter)
           .send()
           .await?;

       Ok(response
           .instance_types
           .unwrap_or_default()
           .into_iter()
           .filter_map(|iti| iti.instance_type)
```

```
            .collect())
    }
```

- Para obter detalhes da API, consulte a DescribeInstanceTypesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Create filters for architecture and instance type patterns
" iv_architecture = 'x86_64'
DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
APPEND NEW /aws1/cl_ec2filter(
  iv_name = 'processor-info.supported-architecture'
  it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
    ( NEW /aws1/cl_ec2valuestringlist_w( iv_architecture ) )
  )
) TO lt_filters.
" Filter for instance type patterns like '*.micro', '*.small'
APPEND NEW /aws1/cl_ec2filter(
  iv_name = 'instance-type'
  it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
    ( NEW /aws1/cl_ec2valuestringlist_w( '*.micro' ) )
    ( NEW /aws1/cl_ec2valuestringlist_w( '*.small' ) )
  )
) TO lt_filters.

TRY.
    oo_result = lo_ec2->describeinstancetypes( it_filters = lt_filters ).
     " oo_result is returned for testing purposes. "
    DATA(lt_instance_types) = oo_result->get_instancetypes( ).
    MESSAGE 'Retrieved information about EC2 instance types.' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeInstanceTypesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Return a list of instance types matching the specified architecture
    /// and instance sizes.
    ///
    /// - Parameters:
    ///   - architecture: The architecture of the instance types to return, as
    ///     a member of `EC2ClientTypes.ArchitectureValues`.
    ///   - sizes: An array of one or more strings identifying sizes of
    ///     instance type to accept.
    ///
    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues
 = EC2ClientTypes.ArchitectureValues.x8664,
                          sizes: [String] = ["*.micro", "*.small"]) async
                          -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
```

```
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )

        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )

            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }
```

- Para obter detalhes da API, consulte DescribeInstanceTypesa referência da API AWS SDK for Swift.

Use **`DescribeInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar e gerenciar um serviço resiliente
- Criar uma rede privada virtual básica

- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criação e gerenciamento de volumes de armazenamento em bloco
- Conceitos básicos de compras do mercado de software
- Introdução ao balanceamento de carga
- Conceitos básicos de streaming gerenciado
- Introdução às máquinas virtuais

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get information about EC2 instances with a particular state.
/// </summary>
/// <param name="tagName">The name of the tag to filter on.</param>
/// <param name="tagValue">The value of the tag to look for.</param>
/// <returns>True if successful.</returns>
public async Task<bool> GetInstancesWithState(string state)
{
    try
    {
        // Filters the results of the instance list.
        var filters = new List<Filter>
        {
            new Filter
            {
                Name = $"instance-state-name",
                Values = new List<string> { state, },
            },
        };
        var request = new DescribeInstancesRequest { Filters = filters, };

        Console.WriteLine($"\nShowing instances with state {state}");
```

```
            var paginator = _amazonEC2.Paginators.DescribeInstances(request);

            await foreach (var response in paginator.Responses)
            {
                foreach (var reservation in response.Reservations)
                {
                    foreach (var instance in reservation.Instances)
                    {
                        Console.Write($"Instance ID: {instance.InstanceId} ");
                        Console.WriteLine($"\tCurrent State:
{instance.State.Name}");
                    }
                }
            }

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Invalid parameter value for filtering instances.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list instances because: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DescribeInstancesa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_describe_instances
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID of the instance to describe (optional).
#       -q query - The query to filter the response (optional).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_instances() {
  local instance_id query response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_instances"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
    echo "  -q query - The query to filter the response (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:q:h" option; do
```

```
    case "${option}" in
      i) instance_id="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
  fi

  local query_arg=""
  if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
  else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
```

```
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
```

```
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeInstancesem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instances associated
 with an account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeInstances(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
 ec2Client.DescribeInstances(request);
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<
                          std::setw(48) << "Name" <<
                          std::setw(20) << "ID" <<
                          std::setw(25) << "Ami" <<
                          std::setw(15) << "Type" <<
```

```
                        std::setw(15) << "State" <<
                        std::setw(15) << "Monitoring" << std::endl;
                header = true;
            }

            const std::vector<Aws::EC2::Model::Reservation> &reservations =
                    outcome.GetResult().GetReservations();

            for (const auto &reservation: reservations) {
                const std::vector<Aws::EC2::Model::Instance> &instances =
                        reservation.GetInstances();
                for (const auto &instance: instances) {
                    Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                    instance.GetState().GetName());

                    Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                                    instance.GetInstanceType());

                    Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                                    instance.GetMonitoring().GetState());
                    Aws::String name = "Unknown";

                    const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                    auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                                                 [](const Aws::EC2::Model::Tag
&tag) {
                                                     return tag.GetKey() ==
"Name";
                                                 });
                    if (nameIter != tags.cend()) {
                        name = nameIter->GetValue();
                    }
                    std::cout <<
                              std::setw(48) << name <<
                              std::setw(20) << instance.GetInstanceId() <<
                              std::setw(25) << instance.GetImageId() <<
                              std::setw(15) << typeString <<
```

```
                                std::setw(15) << instanceStateString <<
                                std::setw(15) << monitorString << std::endl;
                    }
                }

                if (!outcome.GetResult().GetNextToken().empty()) {
                    request.SetNextToken(outcome.GetResult().GetNextToken());
                } else {
                    done = true;
                }
            } else {
                std::cerr << "Failed to describe EC2 instances:" <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }

        return true;
    }
```

- Para obter detalhes da API, consulte DescribeInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: descrever uma instância

O exemplo `describe-instances` a seguir descreve a instância especificada.

```
aws ec2 describe-instances \
    --instance-ids i-1234567890abcdef0
```

Saída:

```
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
```

```
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-0abcdef1234567890",
                    "InstanceId": "i-1234567890abcdef0",
                    "InstanceType": "t3.nano",
                    "KeyName": "my-key-pair",
                    "LaunchTime": "2022-11-15T10:48:59+00:00",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-east-2a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
                    "PrivateIpAddress": "10-0-0-157",
                    "ProductCodes": [],
                    "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
                    "PublicIpAddress": "34.253.223.13",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "StateTransitionReason": "",
                    "SubnetId": "subnet-04a636d18e83cfacb",
                    "VpcId": "vpc-1234567890abcdef0",
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/xvda",
                            "Ebs": {
                                "AttachTime": "2022-11-15T10:49:00+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-02e6ccdca7de29cf2"
                            }
                        }
                    ],
                    "ClientToken": "1234abcd-1234-abcd-1234-d46a8903e9bc",
                    "EbsOptimized": true,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
```

```
                    "IamInstanceProfile": {
                        "Arn": "arn:aws:iam::111111111111:instance-profile/
AmazonSSMRoleForInstancesQuickSetup",
                        "Id": "11111111111111111111"
                    },
                    "NetworkInterfaces": [
                        {
                            "Association": {
                                "IpOwnerId": "amazon",
                                "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
                                "PublicIp": "34.253.223.13"
                            },
                            "Attachment": {
                                "AttachTime": "2022-11-15T10:48:59+00:00",
                                "AttachmentId": "eni-attach-1234567890abcdefg",
                                "DeleteOnTermination": true,
                                "DeviceIndex": 0,
                                "Status": "attached",
                                "NetworkCardIndex": 0
                            },
                            "Description": "",
                            "Groups": [
                                {
                                    "GroupName": "launch-wizard-146",
                                    "GroupId": "sg-1234567890abcdefg"
                                }
                            ],
                            "Ipv6Addresses": [],
                            "MacAddress": "00:11:22:33:44:55",
                            "NetworkInterfaceId": "eni-1234567890abcdefg",
                            "OwnerId": "104024344472",
                            "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
                            "PrivateIpAddress": "10-0-0-157",
                            "PrivateIpAddresses": [
                                {
                                    "Association": {
                                        "IpOwnerId": "amazon",
                                        "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
                                        "PublicIp": "34.253.223.13"
                                    },
                                    "Primary": true,
```

```
                                    "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
                                    "PrivateIpAddress": "10-0-0-157"
                                }
                            ],
                            "SourceDestCheck": true,
                            "Status": "in-use",
                            "SubnetId": "subnet-1234567890abcdefg",
                            "VpcId": "vpc-1234567890abcdefg",
                            "InterfaceType": "interface"
                        }
                    ],
                    "RootDeviceName": "/dev/xvda",
                    "RootDeviceType": "ebs",
                    "SecurityGroups": [
                        {
                            "GroupName": "launch-wizard-146",
                            "GroupId": "sg-1234567890abcdefg"
                        }
                    ],
                    "SourceDestCheck": true,
                    "Tags": [
                        {
                            "Key": "Name",
                            "Value": "my-instance"
                        }
                    ],
                    "VirtualizationType": "hvm",
                    "CpuOptions": {
                        "CoreCount": 1,
                        "ThreadsPerCore": 2
                    },
                    "CapacityReservationSpecification": {
                        "CapacityReservationPreference": "open"
                    },
                    "HibernationOptions": {
                        "Configured": false
                    },
                    "MetadataOptions": {
                        "State": "applied",
                        "HttpTokens": "optional",
                        "HttpPutResponseHopLimit": 1,
                        "HttpEndpoint": "enabled",
                        "HttpProtocolIpv6": "disabled",
```

```
                        "InstanceMetadataTags": "enabled"
                    },
                    "EnclaveOptions": {
                        "Enabled": false
                    },
                    "PlatformDetails": "Linux/UNIX",
                    "UsageOperation": "RunInstances",
                    "UsageOperationUpdateTime": "2022-11-15T10:48:59+00:00",
                    "PrivateDnsNameOptions": {
                        "HostnameType": "ip-name",
                        "EnableResourceNameDnsARecord": true,
                        "EnableResourceNameDnsAAAARecord": false
                    },
                    "MaintenanceOptions": {
                        "AutoRecovery": "default"
                    }
                }
            ],
            "OwnerId": "111111111111",
            "ReservationId": "r-1234567890abcdefg"
        }
    ]
}
```

Exemplo 2: para filtrar instâncias com o tipo especificado

O exemplo `describe-instances` a seguir usa filtros para definir o escopo dos resultados para instâncias do tipo especificado.

```
aws ec2 describe-instances \
    --filters Name=instance-type,Values=m5.large
```

Para obter um exemplo da saída, consulte o Exemplo 1.

Para obter mais informações, consulte Listar e filtrar usando a CLI no Guia do usuário do Amazon EC2.

Exemplo 3: filtrar instâncias com o tipo e a zona de disponibilidade especificados

O exemplo `describe-instances` a seguir usa vários filtros para definir o escopo dos resultados para instâncias com o tipo especificado que também estão na zona de disponibilidade especificada.

```
aws ec2 describe-instances \
    --filters Name=instance-type,Values=t2.micro,t3.micro Name=availability-
zone,Values=us-east-2c
```

Para obter um exemplo da saída, consulte o Exemplo 1.

Exemplo 4: filtrar instâncias com o tipo e a zona de disponibilidade especificados usando um arquivo JSON

O exemplo `describe-instances` a seguir usa um arquivo de entrada JSON para realizar a mesma filtragem do exemplo anterior. Quando os filtros ficam mais complicados, pode ser mais fácil especificá-los em um arquivo JSON.

```
aws ec2 describe-instances \
    --filters file://filters.json
```

Conteúdo de `filters.json`:

```
[
    {
        "Name": "instance-type",
        "Values": ["t2.micro", "t3.micro"]
    },
    {
        "Name": "availability-zone",
        "Values": ["us-east-2c"]
    }
]
```

Para obter um exemplo da saída, consulte o Exemplo 1.

Exemplo 5: filtrar instâncias com a tag Proprietário especificada

O exemplo `describe-instances` a seguir usa filtros de tag para definir o escopo dos resultados para instâncias que têm uma tag com a chave de tag especificada (Proprietário), independentemente do valor da tag.

```
aws ec2 describe-instances \
    --filters "Name=tag-key,Values=Owner"
```

Para obter um exemplo da saída, consulte o Exemplo 1.

Exemplo 6: filtrar instâncias com o valor especificado my-team da tag

O exemplo `describe-instances` a seguir usa filtros de tag para definir o escopo dos resultados para instâncias que têm uma tag com o valor especificado da tag (my-team), independentemente da chave da tag.

```
aws ec2 describe-instances \
    --filters "Name=tag-value,Values=my-team"
```

Para obter um exemplo da saída, consulte o Exemplo 1.

Exemplo 7: filtrar instâncias com a tag Proprietário e o valor my-team especificados

O exemplo `describe-instances` a seguir usa filtros de tag para definir o escopo dos resultados para instâncias que têm a tag especificada (Owner=my-team).

```
aws ec2 describe-instances \
    --filters "Name=tag:Owner,Values=my-team"
```

Para obter um exemplo da saída, consulte o Exemplo 1.

Exemplo 8: exibir somente IDs de instância e sub-rede de todas as instâncias

Os exemplos `describe-instances` a seguir usam o parâmetro `--query` para exibir somente os IDs de instância e sub-rede de todas as instâncias, no formato JSON.

Linux e macOS:

```
aws ec2 describe-instances \
    --query 'Reservations[*].Instances[*].{Instance:InstanceId,Subnet:SubnetId}'
 \
    --output json
```

Windows:

```
aws ec2 describe-instances ^
    --query "Reservations[*].Instances[*].
{Instance:InstanceId,Subnet:SubnetId}" ^
```

```
    --output json
```

Saída:

```
[
    {
        "Instance": "i-057750d42936e468a",
        "Subnet": "subnet-069beee9b12030077"
    },
    {
        "Instance": "i-001efd250faaa6ffa",
        "Subnet": "subnet-0b715c6b7db68927a"
    },
    {
        "Instance": "i-027552a73f021f3bd",
        "Subnet": "subnet-0250c25a1f4e15235"
    }
    ...
]
```

Exemplo 9: filtrar instâncias do tipo especificado e exibir somente os IDs de instância

O exemplo `describe-instances` a seguir usa filtros para definir o escopo dos resultados para instâncias do tipo especificado e o parâmetro `--query` para exibir somente os IDs da instância.

```
aws ec2 describe-instances \
    --filters "Name=instance-type,Values=t2.micro" \
    --query "Reservations[*].Instances[*].[InstanceId]" \
    --output text
```

Saída:

```
i-031c0dc19de2fb70c
i-00d8bff789a736b75
i-0b715c6b7db68927a
i-0626d4edd54f1286d
i-00b8ae04f9f99908e
i-0fc71c25d2374130c
```

Exemplo 10: filtrar instâncias do tipo especificado e exibir somente os IDs de instância, a zona de disponibilidade e o valor especificado da tag

Os exemplos `describe-instances` a seguir exibem o ID da instância, a zona de disponibilidade e o valor da tag `Name` para instâncias que têm uma tag com o nome `tag-key`, em formato de tabela.

Linux e macOS:

```
aws ec2 describe-instances \
    --filters Name=tag-key,Values=Name \
    --query 'Reservations[*].Instances[*].
{Instance:InstanceId,AZ:Placement.AvailabilityZone,Name:Tags[?Key==`Name`]|
[0].Value}' \
    --output table
```

Windows:

```
aws ec2 describe-instances ^
    --filters Name=tag-key,Values=Name ^
    --query "Reservations[*].Instances[*].
{Instance:InstanceId,AZ:Placement.AvailabilityZone,Name:Tags[?Key=='Name']|
[0].Value}" ^
    --output table
```

Saída:

```
-------------------------------------------------------------
|                     DescribeInstances                     |
+--------------+-----------------------+--------------------+
|      AZ      |       Instance        |        Name        |
+--------------+-----------------------+--------------------+
|  us-east-2b  |  i-057750d42936e468a  |  my-prod-server    |
|  us-east-2a  |  i-001efd250faaa6ffa  |  test-server-1     |
|  us-east-2a  |  i-027552a73f021f3bd  |  test-server-2     |
+--------------+-----------------------+--------------------+
```

Exemplo 11: descrever instâncias em um grupo com posicionamento em partições

O exemplo `describe-instances` a seguir descreve a instância especificada. A saída inclui as informações de posicionamento da instância, o que contém o nome do grupo de posicionamento e o número da partição da instância.

```
aws ec2 describe-instances \
```

```
    --instance-ids i-0123a456700123456 \
    --query "Reservations[*].Instances[*].Placement"
```

Saída:

```
[
    [
        {
            "AvailabilityZone": "us-east-1c",
            "GroupName": "HDFS-Group-A",
            "PartitionNumber": 3,
            "Tenancy": "default"
        }

    ]
]
```

Para obter mais informações, consulte Descrever instâncias em um grupo de posicionamento no Guia do usuário do Amazon EC2.

Exemplo 12: filtrar instâncias com o grupo de posicionamento e o número de partição especificados

O exemplo `describe-instances` a seguir filtra os resultados somente para as instâncias com o grupo de posicionamento e o número de partição especificados.

```
aws ec2 describe-instances \
    --filters "Name=placement-group-name,Values=HDFS-Group-A" "Name=placement-
partition-number,Values=7"
```

A seguir, são mostradas somente as informações relevantes da saída.

```
"Instances": [
    {
        "InstanceId": "i-0123a456700123456",
        "InstanceType": "r4.large",
        "Placement": {
            "AvailabilityZone": "us-east-1c",
            "GroupName": "HDFS-Group-A",
            "PartitionNumber": 7,
            "Tenancy": "default"
```

```
        }
    },
    {
        "InstanceId": "i-9876a543210987654",
        "InstanceType": "r4.large",
        "Placement": {
            "AvailabilityZone": "us-east-1c",
            "GroupName": "HDFS-Group-A",
            "PartitionNumber": 7,
            "Tenancy": "default"
        }
    ],
```

Para obter mais informações, consulte Descrever instâncias em um grupo de posicionamento no Guia do usuário do Amazon EC2.

Exemplo 13: filtrar instâncias configuradas para permitir o acesso às tags dos metadados da instância

O exemplo `describe-instances` a seguir filtra os resultados somente para as instâncias que estão configuradas para permitir o acesso às tags de instância nos metadados da instância.

```
aws ec2 describe-instances \
    --filters "Name=metadata-options.instance-metadata-tags,Values=enabled" \
    --query "Reservations[*].Instances[*].InstanceId" \
    --output text
```

A saída esperada é mostrada a seguir.

```
i-1234567890abcdefg
i-abcdefg1234567890
i-11111111aaaaaaaaa
i-aaaaaaaa111111111
```

Para obter mais informações, consulte Work with instance tags in instance metadata no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeInstancesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the state of an EC2 instance.
    * The paginator helps you iterate over multiple pages of results.
    *
    * @param newInstanceId the ID of the EC2 instance to describe
    * @return a {@link CompletableFuture} that, when completed, contains a
string describing the state of the EC2 instance
    */
   public CompletableFuture<String> describeEC2InstancesAsync(String
newInstanceId) {
       DescribeInstancesRequest request = DescribeInstancesRequest.builder()
           .instanceIds(newInstanceId)
           .build();

       DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
       AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.reservations().stream()
               .flatMap(reservation -> reservation.instances().stream())
               .filter(instance -> instance.instanceId().equals(newInstanceId))
               .findFirst()
               .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
       }).thenApply(v -> {
           String publicIpAddress = publicIpAddressRef.get();
           if (publicIpAddress == null) {
               throw new RuntimeException("Instance with ID " + newInstanceId +
" not found.");
           }
           return publicIpAddress;
       }).exceptionally(ex -> {
```

```
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
        });
    }
```

- Para obter detalhes da API, consulte DescribeInstancesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, paginateDescribeInstances } from "@aws-sdk/client-ec2";

/**
 * List all of your EC2 instances running with the provided architecture that
 * were launched in the past month.
 * @param {{ pageSize: string, architectures: string[] }} options
 */
export const main = async ({ pageSize, architectures }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});
  const d = new Date();
  const year = d.getFullYear();
  const month = `0${d.getMonth() + 1}`.slice(-2);
  const launchTimePattern = `${year}-${month}-*`;

  const paginator = paginateDescribeInstances(
    {
      client,
      pageSize,
    },
    {
      Filters: [
```

```
        { Name: "architecture", Values: architectures },
        { Name: "instance-state-name", Values: ["running"] },
        {
          Name: "launch-time",
          Values: [launchTimePattern],
        },
      ],
    },
  );

  try {
    /**
     * @type {import('@aws-sdk/client-ec2').Instance[]}
     */
    const instanceList = [];
    for await (const page of paginator) {
      const { Reservations } = page;
      for (const reservation of Reservations) {
        instanceList.push(...reservation.Instances);
      }
    }
    console.log(
      `Running instances launched this month:\n\n${instanceList.map((instance) =>
 instance.InstanceId).join("\n")}`,
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeInstancesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstances(request)
        response.reservations?.forEach { reservation ->
            reservation.instances?.forEach { instance ->
                println("Instance Id is ${instance.instanceId}")
                println("Image id is ${instance.imageId}")
                println("Instance type is ${instance.instanceType}")
                println("Instance state name is ${instance.state?.name}")
                println("monitoring information is
 ${instance.monitoring?.state}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeInstancesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a instância especificada.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Saída:

```
AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           : TleEy1448154045270
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    : Amazon.EC2.Model.IamInstanceProfile
ImageId               : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle     :
InstanceType          : t2.micro
KernelId              :
KeyName               : my-key-pair
LaunchTime            : 12/4/2015 4:44:40 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {ip-10-0-2-172.us-west-2.compute.internal}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress      : 10.0.2.172
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       :
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {default}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-12345678
Tags                  : {Name}
VirtualizationType    : hvm
VpcId                 : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as instâncias na região atual, agrupadas por reserva. Para ver os detalhes da instância, expanda a coleção de instâncias em cada objeto de reserva.

```
Get-EC2Instance
```

Saída:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Exemplo 3: esse exemplo ilustra o uso de um filtro para consultar instâncias do EC2 em uma sub-rede específica de uma VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-01af...82cf180e19 t2.medium    Windows  10.0.0.98                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0374...7e9d5b0c45 t2.xlarge    Windows  10.0.0.53                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

Exemplo 4: esse exemplo ilustra o uso de um filtro com vários valores para consultar instâncias do EC2 que estão em execução e que foram interrompidas

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

Exemplo 5: Este exemplo ilustra o uso de um filtro com vários valores para consultar instâncias do EC2 que estão em execução e paradas e o uso do Select-Object cmdlet para escolher valores específicos para saída.

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

$SelectParams = @{
    Property = @(
```

```
        "InstanceID", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_.Tags.Key.IndexOf("Name")].Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress Name         State
----------          ------------ -------- ---------------- ----         -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7         ec2-name-01  running
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8         ec2-name-02  stopped
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10        ec2-name-03  running
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11        ec2-name-04  stopped
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5         ec2-name-05  running
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6         ec2-name-06  stopped
```

- Para obter detalhes da API, consulte DescribeInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a instância especificada.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Saída:

```
AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           : TleEy1448154045270
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    : Amazon.EC2.Model.IamInstanceProfile
ImageId               : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle     :
InstanceType          : t2.micro
KernelId              :
```

```
KeyName               : my-key-pair
LaunchTime            : 12/4/2015 4:44:40 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {ip-10-0-2-172.us-west-2.compute.internal}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress      : 10.0.2.172
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       :
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {default}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-12345678
Tags                  : {Name}
VirtualizationType    : hvm
VpcId                 : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as instâncias na região atual, agrupadas por reserva. Para ver os detalhes da instância, expanda a coleção de instâncias em cada objeto de reserva.

```
Get-EC2Instance
```

Saída:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
```

```
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Exemplo 3: esse exemplo ilustra o uso de um filtro para consultar instâncias do EC2 em uma sub-rede específica de uma VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-01af...82cf180e19 t2.medium    Windows  10.0.0.98                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0374...7e9d5b0c45 t2.xlarge    Windows  10.0.0.53                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

Exemplo 4: esse exemplo ilustra o uso de um filtro com vários valores para consultar instâncias do EC2 que estão em execução e que foram interrompidas

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

```
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

Exemplo 5: Este exemplo ilustra o uso de um filtro com vários valores para consultar instâncias do EC2 que estão em execução e paradas e o uso do Select-Object cmdlet para escolher valores específicos para saída.

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

$SelectParams = @{
    Property = @(
        "InstanceID", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_.Tags.Key.IndexOf("Name")].Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress Name         State
----------          ------------ -------- ---------------- ----         -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7         ec2-name-01  running
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8         ec2-name-02  stopped
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10        ec2-name-03  running
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11        ec2-name-04  stopped
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5         ec2-name-05  running
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6         ec2-name-06  stopped
```

Exemplo 6: Este exemplo valida as permissões para obter instâncias do EC2 usando o DryRun parâmetro sem realmente buscá-las. Nota: se isso for feito com sucesso, será gerada uma exceção, que é o comportamento esperado.

```
Get-EC2Tag -DryRun $true
```

Saída:

```
Get-EC2Instance: Request would have succeeded, but DryRun flag is set.
```

- Para obter detalhes da API, consulte DescribeInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
```

```
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
instances in this state
                             will be displayed. Default is 'running'. Example
states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:
                            continue  # Skip this instance if it doesn't match
the filter
```

```
                        # Create a formatted string with instance details
                        instance_info = (
                            f"• ID: {instance['InstanceId']}\n"
                            f"• Image ID: {instance['ImageId']}\n"
                            f"• Instance type: {instance['InstanceType']}\n"
                            f"• Key name: {instance['KeyName']}\n"
                            f"• VPC ID: {instance['VpcId']}\n"
                            f"• Public IP: {instance.get('PublicIpAddress', 'N/
A')}\n"
                            f"• State: {instance_state}"
                        )
                        print(instance_info)

        except ClientError as err:
            logger.error(
                f"Failed to display instance(s). : {' '.join(map(str,
 instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
                raise
```

- Para obter detalhes da API, consulte a DescribeInstancesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte DescribeInstancesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recupere detalhes de uma instância do EC2.

```
   pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
       let response = self
           .client
           .describe_instances()
           .instance_ids(instance_id)
           .send()
           .await?;

       let instance = response
           .reservations()
           .first()
           .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
           .instances()
           .first()
           .ok_or_else(|| {
               EC2Error::new(format!("No instances in reservation for
{instance_id}"))
           })?;

       Ok(instance.clone())
   }
```

Depois de criar uma instância do EC2, recupere e armazene seus detalhes.

```
    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        image_id: &str,
        instance_type: InstanceType,
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }
```

- Para obter detalhes da API, consulte a DescribeInstancesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeinstances( ).                       "
oo_result is returned for testing purposes. "

        " Retrieving details of EC2 instances. "
        DATA: lv_istance_id    TYPE /aws1/ec2string,
              lv_status        TYPE /aws1/ec2instancestatename,
              lv_instance_type TYPE /aws1/ec2instancetype,
              lv_image_id      TYPE /aws1/ec2string.
```

```
        LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
          LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
            lv_istance_id = lo_instance->get_instanceid( ).
            lv_status = lo_instance->get_state( )->get_name( ).
            lv_instance_type = lo_instance->get_instancetype( ).
            lv_image_id = lo_instance->get_imageid( ).
          ENDLOOP.
        ENDLOOP.
        MESSAGE 'Retrieved information about EC2 instances.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeInstancesreferência da API AWS SDK for SAP ABAP.

Usar **`DescribeInternetGateways`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeInternetGateways`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar uma rede privada virtual básica

CLI

AWS CLI

Descrever um gateway da internet

O exemplo `describe-internet-gateways` a seguir descreve o gateway da internet especificado.

```
aws ec2 describe-internet-gateways \
    --internet-gateway-ids igw-0d0fb496b3EXAMPLE
```

Saída:

```
{
    "InternetGateways": [
        {
            "Attachments": [
                {
                    "State": "available",
                    "VpcId": "vpc-0a60eb65b4EXAMPLE"
                }
            ],
            "InternetGatewayId": "igw-0d0fb496b3EXAMPLE",
            "OwnerId": "123456789012",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "my-igw"
                }
            ]
        }
    ]
}
```

Para obter mais informações, consulte Gateways da Internet no Guia do usuário da Amazon VPC.

- Para obter detalhes da API, consulte DescribeInternetGatewaysem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o gateway da internet especificado.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Saída:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

Exemplo 2: esse exemplo descreve todos os gateways da internet.

```
Get-EC2InternetGateway
```

Saída:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- Para obter detalhes da API, consulte DescribeInternetGatewaysem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o gateway da internet especificado.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Saída:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

Exemplo 2: esse exemplo descreve todos os gateways da internet.

```
Get-EC2InternetGateway
```

Saída:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- Para obter detalhes da API, consulte DescribeInternetGatewaysem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeKeyPairs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeKeyPairs`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução aos clusters de processamento de big data
- Conceitos básicos de streaming gerenciado

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get information about an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair.</param>
/// <returns>A list of key pair information.</returns>
public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
{
    try
    {
        var request = new DescribeKeyPairsRequest();
        if (!string.IsNullOrEmpty(keyPairName))
        {
            request = new DescribeKeyPairsRequest
            {
                KeyNames = new List<string> { keyPairName }
            };
        }

        var response = await _amazonEC2.DescribeKeyPairsAsync(request);
        return response.KeyPairs.ToList();
```

```
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while describing the key pair.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeKeyPairsa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 key pairs.
#
```

```
# Parameters:
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_key_pairs() {
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_key_pairs"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
  }
```

```
  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
```

```
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeKeyPairsem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instance key pairs.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeKeyPairs(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeKeyPairsRequest request;

    Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
 ec2Client.DescribeKeyPairs(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "Name" <<
                  std::setw(64) << "Fingerprint" << std::endl;

        const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
```

```
                outcome.GetResult().GetKeyPairs();
        for (const auto &key_pair: key_pairs) {
            std::cout << std::left <<
                      std::setw(32) << key_pair.GetKeyName() <<
                      std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe key pairs:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeKeyPairsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exibir um par de chaves

O exemplo `describe-key-pairs` a seguir mostra as informações do par de chaves especificado.

```
aws ec2 describe-key-pairs \
    --key-names my-key-pair
```

Saída:

```
{
    "KeyPairs": [
        {
            "KeyPairId": "key-0b94643da6EXAMPLE",
            "KeyFingerprint":
 "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f",
            "KeyName": "my-key-pair",
            "KeyType": "rsa",
            "Tags": [],
```

```
            "CreateTime": "2022-05-27T21:51:16.000Z"
        }
    ]
}
```

Para obter mais informações, consulte Descrever chaves públicas no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeKeyPairsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the key pairs associated with the current AWS
account.
    *
    * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
    * information about the key pairs.
    */
   public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
       CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
       responseFuture.whenComplete((response, exception) -> {
           if (exception != null) {
             throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
           }
       });

       return responseFuture;
   }
```

- Para obter detalhes da API, consulte DescribeKeyPairsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeKeyPairsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * List all key pairs in the current AWS account.
 * @param {{ dryRun: boolean }}
 */
export const main = async ({ dryRun }) => {
  const client = new EC2Client({});
  const command = new DescribeKeyPairsCommand({ DryRun: dryRun });

  try {
    const { KeyPairs } = await client.send(command);
    const keyPairList = KeyPairs.map(
      (kp) => ` • ${kp.KeyPairId}: ${kp.KeyName}`,
    ).join("\n");
    console.log("The following key pairs were found in your account:");
    console.log(keyPairList);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeKeyPairsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEC2Keys() {
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeKeyPairsreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o par de chaves especificado.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Saída:

```
KeyFingerprint                                          KeyName
--------------                                          -------
```

```
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f my-key-pair
```

Exemplo 2: esse exemplo descreve todos os pares de chaves.

```
Get-EC2KeyPair
```

- Para obter detalhes da API, consulte DescribeKeyPairsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o par de chaves especificado.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Saída:

```
KeyFingerprint                                          KeyName
--------------                                          -------
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f my-key-pair
```

Exemplo 2: esse exemplo descreve todos os pares de chaves.

```
Get-EC2KeyPair
```

- Para obter detalhes da API, consulte DescribeKeyPairsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPairWrapper:
    """
```

```
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def list(self, limit: Optional[int] = None) -> None:
```

```
        """
        Displays a list of key pairs for the current account.

        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
                      all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}'
with fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise
```

- Para obter detalhes da API, consulte a DescribeKeyPairsReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }
```

- Para obter detalhes da API, consulte a DescribeKeyPairsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describekeypairs( ).                        "
 oo_result is returned for testing purposes. "
        DATA(lt_key_pairs) = oo_result->get_keypairs( ).
        MESSAGE 'Retrieved information about key pairs.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeKeyPairsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Describe the key pairs associated with the user by outputting each key
    /// pair's name and fingerprint.
    func describeKeyPairs() async {
        do {
            let output = try await ec2Client.describeKeyPairs(
                input: DescribeKeyPairsInput()
            )

            guard let keyPairs = output.keyPairs else {
                print("*** No key pairs list available.")
                return
            }

            for keyPair in keyPairs {
                print(keyPair.keyName ?? "<unknown>", ":",
 keyPair.keyFingerprint ?? "<unknown>")
            }
        } catch {
            print("*** Error: Unable to obtain a key pair list.")
        }
    }
```

- Para obter detalhes da API, consulte DescribeKeyPairsa referência da API AWS SDK for Swift.

Usar **`DescribeNetworkAcls`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeNetworkAcls`.

CLI

AWS CLI

Descrever suas ACLs de rede

O exemplo `describe-network-acls` a seguir recupera detalhes das suas ACLs de rede.

```
aws ec2 describe-network-acls
```

Saída:

```
{
    "NetworkAcls": [
        {
            "Associations": [
                {
                    "NetworkAclAssociationId": "aclassoc-0c1679dc41EXAMPLE",
                    "NetworkAclId": "acl-0ea1f54ca7EXAMPLE",
                    "SubnetId": "subnet-0931fc2fa5EXAMPLE"
                }
            ],
            "Entries": [
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": true,
                    "Protocol": "-1",
                    "RuleAction": "allow",
                    "RuleNumber": 100
                },
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": true,
                    "Protocol": "-1",
                    "RuleAction": "deny",
                    "RuleNumber": 32767
                },
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": false,
                    "Protocol": "-1",
                    "RuleAction": "allow",
                    "RuleNumber": 100
```

```
                },
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": false,
                    "Protocol": "-1",
                    "RuleAction": "deny",
                    "RuleNumber": 32767
                }
            ],
            "IsDefault": true,
            "NetworkAclId": "acl-0ea1f54ca7EXAMPLE",
            "Tags": [],
            "VpcId": "vpc-06e4ab6c6cEXAMPLE",
            "OwnerId": "111122223333"
        },
        {
            "Associations": [],
            "Entries": [
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": true,
                    "Protocol": "-1",
                    "RuleAction": "allow",
                    "RuleNumber": 100
                },
                {
                    "Egress": true,
                    "Ipv6CidrBlock": "::/0",
                    "Protocol": "-1",
                    "RuleAction": "allow",
                    "RuleNumber": 101
                },
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": true,
                    "Protocol": "-1",
                    "RuleAction": "deny",
                    "RuleNumber": 32767
                },
                {
                    "Egress": true,
                    "Ipv6CidrBlock": "::/0",
                    "Protocol": "-1",
                    "RuleAction": "deny",
```

```
                "RuleNumber": 32768
            },
            {
                "CidrBlock": "0.0.0.0/0",
                "Egress": false,
                "Protocol": "-1",
                "RuleAction": "allow",
                "RuleNumber": 100
            },
            {
                "Egress": false,
                "Ipv6CidrBlock": "::/0",
                "Protocol": "-1",
                "RuleAction": "allow",
                "RuleNumber": 101
            },
            {
                "CidrBlock": "0.0.0.0/0",
                "Egress": false,
                "Protocol": "-1",
                "RuleAction": "deny",
                "RuleNumber": 32767
            },
            {
                "Egress": false,
                "Ipv6CidrBlock": "::/0",
                "Protocol": "-1",
                "RuleAction": "deny",
                "RuleNumber": 32768
            }
        ],
        "IsDefault": true,
        "NetworkAclId": "acl-0e2a78e4e2EXAMPLE",
        "Tags": [],
        "VpcId": "vpc-03914afb3eEXAMPLE",
        "OwnerId": "111122223333"
    }
  ]
}
```

Para obter mais informações, consulte ACLs da rede no Guia do usuário da AWS VPC.

- Para obter detalhes da API, consulte DescribeNetworkAclsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a ACL de rede especificada.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Saída:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Exemplo 2: esse exemplo descreve duas regras para a ACL de rede especificada.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Saída:

```
CidrBlock    : 0.0.0.0/0
Egress       : True
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767
```

Exemplo 3: esse exemplo descreve todas as ACLs de rede.

```
Get-EC2NetworkAcl
```

- Para obter detalhes da API, consulte DescribeNetworkAclsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a ACL de rede especificada.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Saída:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Exemplo 2: esse exemplo descreve duas regras para a ACL de rede especificada.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Saída:

```
CidrBlock    : 0.0.0.0/0
Egress       : True
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
```

```
RuleNumber    : 32767
```

Exemplo 3: esse exemplo descreve todas as ACLs de rede.

```
Get-EC2NetworkAcl
```

- Para obter detalhes da API, consulte DescribeNetworkAclsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeNetworkInterfaceAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeNetworkInterfaceAttribute`.

CLI

AWS CLI

Descrever o atributo de anexo de uma interface de rede

Este exemplo de comando descreve o atributo `attachment` da interface de rede especificada.

Comando:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute attachment
```

Saída:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Attachment": {
      "Status": "attached",
      "DeviceIndex": 0,
      "AttachTime": "2015-05-21T20:02:20.000Z",
      "InstanceId": "i-1234567890abcdef0",
      "DeleteOnTermination": true,
      "AttachmentId": "eni-attach-43348162",
      "InstanceOwnerId": "123456789012"
  }
}
```

Descrever o atributo de descrição de uma interface de rede

Este exemplo de comando descreve o atributo `description` da interface de rede especificada.

Comando:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute description
```

Saída:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Description": {
      "Value": "My description"
  }
}
```

Descrever o atributo groupSet de uma interface de rede

Este exemplo de comando descreve o atributo `groupSet` da interface de rede especificada.

Comando:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute groupSet
```

Saída:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Groups": [
      {
          "GroupName": "my-security-group",
          "GroupId": "sg-903004f8"
      }
  ]
}
```

Para descrever o DestCheck atributo de origem de uma interface de rede

Este exemplo de comando descreve o atributo `sourceDestCheck` da interface de rede especificada.

Comando:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute sourceDestCheck
```

Saída:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "SourceDestCheck": {
      "Value": true
  }
}
```

- Para obter detalhes da API, consulte DescribeNetworkInterfaceAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

Saída:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Exemplo 2: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

Saída:

```
Description        : My description
```

Exemplo 3: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

Saída:

```
Groups             : {my-security-group}
```

Exemplo 4: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

Saída:

```
SourceDestCheck    : True
```

- Para obter detalhes da API, consulte DescribeNetworkInterfaceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

Saída:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Exemplo 2: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

Saída:

```
Description        : My description
```

Exemplo 3: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

Saída:

```
Groups             : {my-security-group}
```

Exemplo 4: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

Saída:

```
SourceDestCheck    : True
```

- Para obter detalhes da API, consulte DescribeNetworkInterfaceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeNetworkInterfaces`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeNetworkInterfaces`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

CLI

AWS CLI

Descrever suas interfaces de rede

Este exemplo descreve todas as interfaces de rede.

Comando:

```
aws ec2 describe-network-interfaces
```

Saída:

```
{
  "NetworkInterfaces": [
      {
          "Status": "in-use",
          "MacAddress": "02:2f:8f:b0:cf:75",
          "SourceDestCheck": true,
          "VpcId": "vpc-a01106c2",
          "Description": "my network interface",
          "Association": {
              "PublicIp": "203.0.113.12",
              "AssociationId": "eipassoc-0fbb766a",
              "PublicDnsName": "ec2-203-0-113-12.compute-1.amazonaws.com",
              "IpOwnerId": "123456789012"
          },
          "NetworkInterfaceId": "eni-e5aa89a3",
          "PrivateIpAddresses": [
              {
                  "PrivateDnsName": "ip-10-0-1-17.ec2.internal",
                  "Association": {
                      "PublicIp": "203.0.113.12",
                      "AssociationId": "eipassoc-0fbb766a",
                      "PublicDnsName":
 "ec2-203-0-113-12.compute-1.amazonaws.com",
                      "IpOwnerId": "123456789012"
                  },
                  "Primary": true,
                  "PrivateIpAddress": "10.0.1.17"
              }
          ],
          "RequesterManaged": false,
          "Ipv6Addresses": [],
          "PrivateDnsName": "ip-10-0-1-17.ec2.internal",
          "AvailabilityZone": "us-east-1d",
          "Attachment": {
              "Status": "attached",
              "DeviceIndex": 1,
              "AttachTime": "2013-11-30T23:36:42.000Z",
```

```
            "InstanceId": "i-1234567890abcdef0",
            "DeleteOnTermination": false,
            "AttachmentId": "eni-attach-66c4350a",
            "InstanceOwnerId": "123456789012"
        },
        "Groups": [
            {
                "GroupName": "default",
                "GroupId": "sg-8637d3e3"
            }
        ],
        "SubnetId": "subnet-b61f49f0",
        "OwnerId": "123456789012",
        "TagSet": [],
        "PrivateIpAddress": "10.0.1.17"
    },
    {
        "Status": "in-use",
        "MacAddress": "02:58:f5:ef:4b:06",
        "SourceDestCheck": true,
        "VpcId": "vpc-a01106c2",
        "Description": "Primary network interface",
        "Association": {
            "PublicIp": "198.51.100.0",
            "IpOwnerId": "amazon"
        },
        "NetworkInterfaceId": "eni-f9ba99bf",
        "PrivateIpAddresses": [
            {
                "Association": {
                    "PublicIp": "198.51.100.0",
                    "IpOwnerId": "amazon"
                },
                "Primary": true,
                "PrivateIpAddress": "10.0.1.149"
            }
        ],
        "RequesterManaged": false,
        "Ipv6Addresses": [],
        "AvailabilityZone": "us-east-1d",
        "Attachment": {
            "Status": "attached",
            "DeviceIndex": 0,
            "AttachTime": "2013-11-30T23:35:33.000Z",
```

```
                "InstanceId": "i-0598c7d356eba48d7",
                "DeleteOnTermination": true,
                "AttachmentId": "eni-attach-1b9db777",
                "InstanceOwnerId": "123456789012"
            },
            "Groups": [
                {
                    "GroupName": "default",
                    "GroupId": "sg-8637d3e3"
                }
            ],
            "SubnetId": "subnet-b61f49f0",
            "OwnerId": "123456789012",
            "TagSet": [],
            "PrivateIpAddress": "10.0.1.149"
        }
    ]
}
```

Este exemplo descreve interfaces de rede que têm uma tag com a chave `Purpose` e o valor `Prod`.

Comando:

```
aws ec2 describe-network-interfaces --filters Name=tag:Purpose,Values=Prod
```

Saída:

```
{
    "NetworkInterfaces": [
        {
            "Status": "available",
            "MacAddress": "12:2c:bd:f9:bf:17",
            "SourceDestCheck": true,
            "VpcId": "vpc-8941ebec",
            "Description": "ProdENI",
            "NetworkInterfaceId": "eni-b9a5ac93",
            "PrivateIpAddresses": [
                {
                    "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
                    "Primary": true,
                    "PrivateIpAddress": "10.0.1.55"
```

```
                },
                {
                    "PrivateDnsName": "ip-10-0-1-117.ec2.internal",
                    "Primary": false,
                    "PrivateIpAddress": "10.0.1.117"
                }
            ],
            "RequesterManaged": false,
            "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
            "AvailabilityZone": "us-east-1d",
            "Ipv6Addresses": [],
            "Groups": [
                {
                    "GroupName": "MySG",
                    "GroupId": "sg-905002f5"
                }
            ],
            "SubnetId": "subnet-31d6c219",
            "OwnerId": "123456789012",
            "TagSet": [
                {
                    "Value": "Prod",
                    "Key": "Purpose"
                }
            ],
            "PrivateIpAddress": "10.0.1.55"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeNetworkInterfacesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Saída:

```
Association        :
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone   : us-west-2c
Description        :
Groups             : {my-security-group}
MacAddress         : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : in-use
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as interfaces de rede.

```
Get-EC2NetworkInterface
```

- Para obter detalhes da API, consulte DescribeNetworkInterfacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Saída:

```
Association        :
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone   : us-west-2c
Description        :
Groups             : {my-security-group}
MacAddress         : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
```

```
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : in-use
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as interfaces de rede.

```
Get-EC2NetworkInterface
```

- Para obter detalhes da API, consulte DescribeNetworkInterfacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribePlacementGroups`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribePlacementGroups`.

CLI

AWS CLI

Descrever todos os seus grupos de posicionamento

Este exemplo de comando descreve todos os grupos de posicionamento.

Comando:

```
aws ec2 describe-placement-groups
```

Saída:

```
{
    "PlacementGroups": [
        {
```

```
            "GroupName": "my-cluster",
            "State": "available",
            "Strategy": "cluster"
        },
        ...
    ]
}
```

- Para obter detalhes da API, consulte DescribePlacementGroupsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve o grupo de posicionamento especificado.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Saída:

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- Para obter detalhes da API, consulte DescribePlacementGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o grupo de posicionamento especificado.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Saída:

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- Para obter detalhes da API, consulte DescribePlacementGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribePrefixLists`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribePrefixLists`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Rede privada virtual com servidores privados

CLI

AWS CLI

Descrever listas de prefixo

Este exemplo lista todas as listas de prefixos disponíveis para a região.

Comando:

```
aws ec2 describe-prefix-lists
```

Saída:

```
{
  "PrefixLists": [
    {
      "PrefixListName": "com.amazonaws.us-east-1.s3",
      "Cidrs": [
        "54.231.0.0/17"
      ],
      "PrefixListId": "pl-63a5400a"
    }
  ]
}
```

- Para obter detalhes da API, consulte DescribePrefixListsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo busca o disponível Serviços da AWS em um formato de lista de prefixos para a região

```
Get-EC2PrefixList
```

Saída:

```
Cidrs                                          PrefixListId PrefixListName
-----                                          ------------ --------------
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23} pl-6fa54006  com.amazonaws.eu-
west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}               pl-6da54004  com.amazonaws.eu-
west-1.s3
```

- Para obter detalhes da API, consulte DescribePrefixListsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo busca o disponível Serviços da AWS em um formato de lista de prefixos para a região

```
Get-EC2PrefixList
```

Saída:

```
Cidrs                                          PrefixListId PrefixListName
-----                                          ------------ --------------
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23} pl-6fa54006  com.amazonaws.eu-
west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}               pl-6da54004  com.amazonaws.eu-
west-1.s3
```

- Para obter detalhes da API, consulte DescribePrefixListsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeRegions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeRegions`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) Regions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeRegions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::DescribeRegionsRequest request;
    Aws::EC2::Model::DescribeRegionsOutcome outcome =
 ec2Client.DescribeRegions(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "RegionName" <<
                  std::setw(64) << "Endpoint" << std::endl;

        const auto &regions = outcome.GetResult().GetRegions();
        for (const auto &region: regions) {
            std::cout << std::left <<
                      std::setw(32) << region.GetRegionName() <<
                      std::setw(64) << region.GetEndpoint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe regions:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

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

```
    return outcome.IsSuccess();

}
```

- Para obter detalhes da API, consulte DescribeRegionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: descrever todas as suas regiões habilitadas

O exemplo `describe-regions` a seguir descreve todas as regiões habilitadas para a sua conta.

```
aws ec2 describe-regions
```

Saída:

```
{
    "Regions": [
        {
            "Endpoint": "ec2.eu-north-1.amazonaws.com",
            "RegionName": "eu-north-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-south-1.amazonaws.com",
            "RegionName": "ap-south-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-3.amazonaws.com",
            "RegionName": "eu-west-3",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-2.amazonaws.com",
            "RegionName": "eu-west-2",
```

```
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-1.amazonaws.com",
            "RegionName": "eu-west-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-3.amazonaws.com",
            "RegionName": "ap-northeast-3",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
            "RegionName": "ap-northeast-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-1.amazonaws.com",
            "RegionName": "ap-northeast-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.sa-east-1.amazonaws.com",
            "RegionName": "sa-east-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ca-central-1.amazonaws.com",
            "RegionName": "ca-central-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-southeast-1.amazonaws.com",
            "RegionName": "ap-southeast-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-southeast-2.amazonaws.com",
            "RegionName": "ap-southeast-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-central-1.amazonaws.com",
```

```
            "RegionName": "eu-central-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-east-1.amazonaws.com",
            "RegionName": "us-east-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-east-2.amazonaws.com",
            "RegionName": "us-east-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-west-1.amazonaws.com",
            "RegionName": "us-west-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-west-2.amazonaws.com",
            "RegionName": "us-west-2",
            "OptInStatus": "opt-in-not-required"
        }
    ]
}
```

Para obter mais informações, consulte Regiões e zonas no Guia do usuário do Amazon EC2.

Exemplo 2: para descrever as regiões habilitadas com um endpoint cujo nome contém uma string específica

O exemplo `describe-regions` a seguir descreve todas as regiões que você habilitou e que têm a string “us” no endpoint.

```
aws ec2 describe-regions \
    --filters "Name=endpoint,Values=*us*"
```

Saída:

```
{
    "Regions": [
        {
            "Endpoint": "ec2.us-east-1.amazonaws.com",
```

```
            "RegionName": "us-east-1"
        },
        {
            "Endpoint": "ec2.us-east-2.amazonaws.com",
            "RegionName": "us-east-2"
        },
        {
            "Endpoint": "ec2.us-west-1.amazonaws.com",
            "RegionName": "us-west-1"
        },
        {
            "Endpoint": "ec2.us-west-2.amazonaws.com",
            "RegionName": "us-west-2"
        }
    ]
}
```

Para obter mais informações, consulte Regiões e zonas no Guia do usuário do Amazon EC2.

Exemplo 3: descrever todas as regiões

O exemplo `describe-regions` a seguir descreve todas as regiões disponíveis, incluindo as que estão desabilitadas.

```
aws ec2 describe-regions \
    --all-regions
```

Saída:

```
{
    "Regions": [
        {
            "Endpoint": "ec2.eu-north-1.amazonaws.com",
            "RegionName": "eu-north-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-south-1.amazonaws.com",
            "RegionName": "ap-south-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-3.amazonaws.com",
```

```
            "RegionName": "eu-west-3",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-2.amazonaws.com",
            "RegionName": "eu-west-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-1.amazonaws.com",
            "RegionName": "eu-west-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-3.amazonaws.com",
            "RegionName": "ap-northeast-3",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.me-south-1.amazonaws.com",
            "RegionName": "me-south-1",
            "OptInStatus": "not-opted-in"
        },
        {
            "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
            "RegionName": "ap-northeast-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-1.amazonaws.com",
            "RegionName": "ap-northeast-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.sa-east-1.amazonaws.com",
            "RegionName": "sa-east-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ca-central-1.amazonaws.com",
            "RegionName": "ca-central-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
```

```
            "Endpoint": "ec2.ap-east-1.amazonaws.com",
            "RegionName": "ap-east-1",
            "OptInStatus": "not-opted-in"
        },
        {
            "Endpoint": "ec2.ap-southeast-1.amazonaws.com",
            "RegionName": "ap-southeast-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-southeast-2.amazonaws.com",
            "RegionName": "ap-southeast-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-central-1.amazonaws.com",
            "RegionName": "eu-central-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-east-1.amazonaws.com",
            "RegionName": "us-east-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-east-2.amazonaws.com",
            "RegionName": "us-east-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-west-1.amazonaws.com",
            "RegionName": "us-west-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-west-2.amazonaws.com",
            "RegionName": "us-west-2",
            "OptInStatus": "opt-in-not-required"
        }
    ]
}
```

Para obter mais informações, consulte Regiões e zonas no Guia do usuário do Amazon EC2.

Exemplo 4: listar somente os nomes das regiões

O exemplo `describe-regions` a seguir usa o parâmetro `--query` para filtrar a saída e retornar somente os nomes das regiões como texto.

```
aws ec2 describe-regions \
    --all-regions \
    --query "Regions[].{Name:RegionName}" \
    --output text
```

Saída:

```
eu-north-1
ap-south-1
eu-west-3
eu-west-2
eu-west-1
ap-northeast-3
ap-northeast-2
me-south-1
ap-northeast-1
sa-east-1
ca-central-1
ap-east-1
ap-southeast-1
ap-southeast-2
eu-central-1
us-east-1
us-east-2
us-west-1
us-west-2
```

Para obter mais informações, consulte Regiões e zonas no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeRegionsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeRegionsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * List all available AWS regions.
 * @param {{ regionNames: string[], includeOptInRegions: boolean }} options
 */
export const main = async ({ regionNames, includeOptInRegions }) => {
  const client = new EC2Client({});
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions is true, even the regions that require opt-in will be
 returned.
    AllRegions: includeOptInRegions,
    // You can omit the Filters property if you want to get all regions.
    Filters: regionNames?.length
      ? [
          {
            Name: "region-name",
            // You can specify multiple values for a filter.
            // You can also use '*' as a wildcard. This will return all
            // of the regions that start with `us-east-`.
            Values: regionNames,
          },
        ]
      : undefined,
  });

  try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeRegionsa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve as regiões que estão disponíveis para você.

```
Get-EC2Region
```

Saída:

```
Endpoint                          RegionName
--------                          ----------
ec2.eu-west-1.amazonaws.com       eu-west-1
ec2.ap-southeast-1.amazonaws.com  ap-southeast-1
ec2.ap-southeast-2.amazonaws.com  ap-southeast-2
ec2.eu-central-1.amazonaws.com    eu-central-1
ec2.ap-northeast-1.amazonaws.com  ap-northeast-1
ec2.us-east-1.amazonaws.com       us-east-1
ec2.sa-east-1.amazonaws.com       sa-east-1
ec2.us-west-1.amazonaws.com       us-west-1
ec2.us-west-2.amazonaws.com       us-west-2
```

- Para obter detalhes da API, consulte DescribeRegionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as regiões que estão disponíveis para você.

```
Get-EC2Region
```

Saída:

```
Endpoint                            RegionName
--------                            ----------
ec2.eu-west-1.amazonaws.com         eu-west-1
ec2.ap-southeast-1.amazonaws.com    ap-southeast-1
ec2.ap-southeast-2.amazonaws.com    ap-southeast-2
ec2.eu-central-1.amazonaws.com      eu-central-1
ec2.ap-northeast-1.amazonaws.com    ap-northeast-1
ec2.us-east-1.amazonaws.com         us-east-1
ec2.sa-east-1.amazonaws.com         sa-east-1
ec2.us-west-1.amazonaws.com         us-west-1
ec2.us-west-2.amazonaws.com         us-west-2
```

- Para obter detalhes da API, consulte DescribeRegionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
```

```
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print "  Endpoint\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print ' ' * (max_region_string_length - region.region_name.length)
    print '  '
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print '  Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print "  State\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_zone_string_length
  print '  '
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
```

```
  result.availability_zones.each do |zone|
    print zone.region_name
    print ' ' * (max_region_string_length - zone.region_name.length)
    print '  '
    print zone.zone_name
    print ' ' * (max_zone_string_length - zone.zone_name.length)
    print '  '
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts '  Messages for this zone:'
      zone.messages.each do |message|
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'AWS Regions for Amazon EC2 that are available to you:'
  list_regions_endpoints(ec2_client)
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS
 Region '#{region}':"
```

```
  list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte DescribeRegionsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_regions(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_regions().send().await?;

    println!("Regions:");
    for region in rsp.regions() {
        println!("  {}", region.region_name().unwrap());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeRegionsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeregions( ).                        "
 oo_result is returned for testing purposes. "
        DATA(lt_regions) = oo_result->get_regions( ).
        MESSAGE 'Retrieved information about Regions.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeRegionsreferência da API AWS SDK for SAP ABAP.

Use **`DescribeRouteTables`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeRouteTables`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos gateways de trânsito de rede

CLI

AWS CLI

Descrever suas tabelas de rotas

O exemplo `describe-route-tables` a seguir recupera detalhes das tabelas de rotas

```
aws ec2 describe-route-tables
```

Saída:

```
{
    "RouteTables": [
        {
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-0df3f54e06EXAMPLE",
                    "RouteTableId": "rtb-09ba434c1bEXAMPLE"
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-09ba434c1bEXAMPLE",
            "Routes": [
                {
                    "DestinationCidrBlock": "10.0.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "NatGatewayId": "nat-06c018cbd8EXAMPLE",
                    "Origin": "CreateRoute",
                    "State": "blackhole"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-0065acced4EXAMPLE",
            "OwnerId": "111122223333"
        },
        {
```

```
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-9EXAMPLE",
                    "RouteTableId": "rtb-a1eec7de"
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-a1eec7de",
            "Routes": [
                {
                    "DestinationCidrBlock": "172.31.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "GatewayId": "igw-fEXAMPLE",
                    "Origin": "CreateRoute",
                    "State": "active"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-3EXAMPLE",
            "OwnerId": "111122223333"
        },
        {
            "Associations": [
                {
                    "Main": false,
                    "RouteTableAssociationId": "rtbassoc-0b100c28b2EXAMPLE",
                    "RouteTableId": "rtb-07a98f76e5EXAMPLE",
                    "SubnetId": "subnet-0d3d002af8EXAMPLE"
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-07a98f76e5EXAMPLE",
            "Routes": [
                {
                    "DestinationCidrBlock": "10.0.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
```

```
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "GatewayId": "igw-06cf664d80EXAMPLE",
                    "Origin": "CreateRoute",
                    "State": "active"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-0065acced4EXAMPLE",
            "OwnerId": "111122223333"
        }
    ]
}
```

Para obter mais informações, consulte Trabalhar com tabelas de rotas no Guia do usuário da AWS VPC.

- Para obter detalhes da API, consulte DescribeRouteTablesem Referência de AWS CLI Comandos.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param array $routeTableIds
     * @param array $filters
     * @return array
     */
    public function describeRouteTables(array $routeTableIds = [], array $filters
 = []): array
    {
        $parameters = [];
        if($routeTableIds){
```

```
            $parameters['RouteTableIds'] = $routeTableIds;
        }
        if($filters){
            $parameters['Filters'] = $filters;
        }
        try {
            $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
 $parameters);
            $contents = [];
            foreach ($paginator as $result) {
                foreach ($result['RouteTables'] as $object) {
                    $contents[] = $object['RouteTableId'];
                }
            }
        }catch (Ec2Exception $caught){
            echo "There was a problem paginating the results of
 DescribeRouteTables: {$caught->getAwsErrorMessage()}\n";
            throw $caught;
        }
        return $contents;
    }
```

- Para obter detalhes da API, consulte DescribeRouteTablesa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve todas as tabelas de rotas.

```
Get-EC2RouteTable
```

Saída:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
```

```
NetworkInterfaceId      :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId  :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId  :
```

Exemplo 2: esse exemplo retorna os detalhes da tabela de rotas especificada.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Exemplo 3: esse exemplo descreve as tabelas de rotas da VPC especificada.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Saída:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- Para obter detalhes da API, consulte DescribeRouteTablesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve todas as tabelas de rotas.

```
Get-EC2RouteTable
```

Saída:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId  :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId  :
```

Exemplo 2: esse exemplo retorna os detalhes da tabela de rotas especificada.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Exemplo 3: esse exemplo descreve as tabelas de rotas da VPC especificada.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Saída:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- Para obter detalhes da API, consulte DescribeRouteTablesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def describe_route_tables(self, vpc_ids: list[str]) -> None:
        """
        Displays information about the route tables in the specified VPC.

        :param vpc_ids: A list of VPC IDs.
        """
```

```
        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": vpc_ids}]
            )
            pp(response["RouteTables"])
        except ClientError as err:
            logger.error(
                "Couldn't describe route tables for VPCs %s. Here's why: %s: %s",
                vpc_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeRouteTablesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create filter for VPC ID
    " iv_vpc_id = 'vpc-abc123'
    DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
    APPEND NEW /aws1/cl_ec2filter(
      iv_name = 'vpc-id'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( iv_vpc_id ) )
      )
    ) TO lt_filters.

    TRY.
```

```
        oo_result = lo_ec2->describeroutetables( it_filters = lt_filters ).
       " oo_result is returned for testing purposes. "
        DATA(lt_route_tables) = oo_result->get_routetables( ).
        MESSAGE 'Retrieved information about route tables.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeRouteTablesreferência da API AWS SDK for SAP ABAP.

Usar **`DescribeScheduledInstanceAvailability`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeScheduledInstanceAvailability`.

CLI

AWS CLI

Descrever um cronograma disponível

Este exemplo descreve uma programação que ocorre toda semana no domingo, começando na data especificada.

Comando:

```
aws ec2 describe-scheduled-instance-availability --
recurrence Frequency=Weekly,Interval=1,OccurrenceDays=[1] --first-slot-start-
time-range EarliestTime=2016-01-31T00:00:00Z,LatestTime=2016-01-31T04:00:00Z
```

Saída:

```
{
  "ScheduledInstanceAvailabilitySet": [
    {
        "AvailabilityZone": "us-west-2b",
        "TotalScheduledInstanceHours": 1219,
```

```
        "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
        "MinTermDurationInDays": 366,
        "AvailableInstanceCount": 20,
        "Recurrence": {
            "OccurrenceDaySet": [
                1
            ],
            "Interval": 1,
            "Frequency": "Weekly",
            "OccurrenceRelativeToEnd": false
        },
        "Platform": "Linux/UNIX",
        "FirstSlotStartTime": "2016-01-31T00:00:00Z",
        "MaxTermDurationInDays": 366,
        "SlotDurationInHours": 23,
        "NetworkPlatform": "EC2-VPC",
        "InstanceType": "c4.large",
        "HourlyPrice": "0.095"
    },
    ...
  ]
}
```

Para restringir os resultados, você pode adicionar filtros que especificam o sistema operacional, a rede e o tipo de instância.

Comando:

--filters nome=plataforma, valores= nome=plataforma de rede, valores= nome=tipo de instância, Linux/UNIX valores=C4.large EC2-VPC

- Para obter detalhes da API, consulte DescribeScheduledInstanceAvailabilityem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve uma programação que ocorre toda semana no domingo, começando na data especificada.

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency
 Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -
```

```
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

Saída:

```
AvailabilityZone            : us-west-2b
AvailableInstanceCount      : 20
FirstSlotStartTime          : 1/31/2016 8:00:00 AM
HourlyPrice                 : 0.095
InstanceType                : c4.large
MaxTermDurationInDays       : 366
MinTermDurationInDays       : 366
NetworkPlatform             : EC2-VPC
Platform                    : Linux/UNIX
PurchaseToken               : eyJ2IjoiMSIsInMiOjEsImMiOi...
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
SlotDurationInHours         : 23
TotalScheduledInstanceHours : 1219

...
```

Exemplo 2: para restringir os resultados, você pode adicionar filtros para critérios como sistema operacional, rede e tipo de instância.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Para obter detalhes da API, consulte DescribeScheduledInstanceAvailabilityem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve uma programação que ocorre toda semana no domingo, começando na data especificada.

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency
 Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

Saída:

```
AvailabilityZone            : us-west-2b
AvailableInstanceCount      : 20
FirstSlotStartTime          : 1/31/2016 8:00:00 AM
HourlyPrice                 : 0.095
InstanceType                : c4.large
MaxTermDurationInDays       : 366
MinTermDurationInDays       : 366
NetworkPlatform             : EC2-VPC
Platform                    : Linux/UNIX
PurchaseToken               : eyJ2IjoiMSIsInMiOjEsImMiOi...
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
SlotDurationInHours         : 23
TotalScheduledInstanceHours : 1219

...
```

Exemplo 2: para restringir os resultados, você pode adicionar filtros para critérios como sistema operacional, rede e tipo de instância.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Para obter detalhes da API, consulte DescribeScheduledInstanceAvailabilityem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeScheduledInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeScheduledInstances`.

CLI

AWS CLI

Descrever suas instâncias agendadas

Este exemplo descreve a Instância Agendada especificada.

Comando:

```
aws ec2 describe-scheduled-instances --scheduled-instance-
ids sci-1234-1234-1234-1234-123456789012
```

Saída:

```
{
  "ScheduledInstanceSet": [
      {
          "AvailabilityZone": "us-west-2b",
          "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
          "HourlyPrice": "0.095",
          "CreateDate": "2016-01-25T21:43:38.612Z",
          "Recurrence": {
              "OccurrenceDaySet": [
                  1
              ],
              "Interval": 1,
              "Frequency": "Weekly",
              "OccurrenceRelativeToEnd": false,
              "OccurrenceUnit": ""
          },
          "Platform": "Linux/UNIX",
          "TermEndDate": "2017-01-31T09:00:00Z",
          "InstanceCount": 1,
          "SlotDurationInHours": 32,
          "TermStartDate": "2016-01-31T09:00:00Z",
          "NetworkPlatform": "EC2-VPC",
          "TotalScheduledInstanceHours": 1696,
          "NextSlotStartTime": "2016-01-31T09:00:00Z",
          "InstanceType": "c4.large"
      }
  ]
}
```

Este exemplo descreve todas as suas instâncias agendadas.

Comando:

```
aws ec2 describe-scheduled-instances
```

- Para obter detalhes da API, consulte DescribeScheduledInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a Instância Programada especificada.

```
Get-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012
```

Saída:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

Exemplo 2: esse exemplo descreve todas as suas Instâncias Programadas.

```
Get-EC2ScheduledInstance
```

- Para obter detalhes da API, consulte DescribeScheduledInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a Instância Programada especificada.

```
Get-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012
```

Saída:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

Exemplo 2: esse exemplo descreve todas as suas Instâncias Programadas.

```
Get-EC2ScheduledInstance
```

- Para obter detalhes da API, consulte DescribeScheduledInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeSecurityGroups`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeSecurityGroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma rede privada virtual básica
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Criação e gerenciamento de uma malha de rede de serviços
- Introdução ao armazenamento em cache em memória
- Conceitos básicos de streaming gerenciado

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve information for one or all Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        try
        {
            var securityGroups = new List<SecurityGroup>();
            var request = new DescribeSecurityGroupsRequest();

            if (!string.IsNullOrEmpty(groupId))
            {
                var groupIds = new List<string> { groupId };
                request.GroupIds = groupIds;
            }

            var paginatorForSecurityGroups =
                _amazonEC2.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            return securityGroups;

        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"A security group {groupId} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while listing security groups.
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Display the information returned by the call to
    /// DescribeSecurityGroupsAsync.
    /// </summary>
    /// <param name="securityGroup">A list of security group information.</param>
    public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
    {
        Console.WriteLine($"{securityGroup.GroupName}");
        Console.WriteLine("Ingress permissions:");
        securityGroup.IpPermissions.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
```

```
        });
        Console.WriteLine("Egress permissions:");
        securityGroup.IpPermissionsEgress.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
    }
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_describe_security_groups
#
```

```
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups.
#
# Parameters:
#       -g security_group_id - The ID of the security group to describe
 (optional).
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_describe_security_groups() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_security_groups"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups."
    echo "  -g security_group_id - The ID of the security group to describe
 (optional)."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"
```

```
  if [[ -n "$security_group_id" ]]; then
    response=$(aws ec2 describe-security-groups --group-ids "$security_group_id"
 --query "${query}" --output text)
  else
    response=$(aws ec2 describe-security-groups --query "${query}" --output text)
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports describe-security-groups operation failed.
$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
```

```
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) security groups, or a
 specific group.
```

```
/*!
  \param groupID: A group ID, ignored if empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeSecurityGroups(const Aws::String &groupID,
                                         const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeSecurityGroupsRequest request;

    if (!groupID.empty()) {
        request.AddGroupIds(groupID);
    }

    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
 ec2Client.DescribeSecurityGroups(request);
        if (outcome.IsSuccess()) {
            std::cout << std::left <<
                      std::setw(32) << "Name" <<
                      std::setw(30) << "GroupId" <<
                      std::setw(30) << "VpcId" <<
                      std::setw(64) << "Description" << std::endl;

            const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
                    outcome.GetResult().GetSecurityGroups();

            for (const auto &securityGroup: securityGroups) {
                std::cout << std::left <<
                          std::setw(32) << securityGroup.GetGroupName() <<
                          std::setw(30) << securityGroup.GetGroupId() <<
                          std::setw(30) << securityGroup.GetVpcId() <<
                          std::setw(64) << securityGroup.GetDescription() <<
                          std::endl;
            }
        } else {
            std::cerr << "Failed to describe security groups:" <<
                      outcome.GetError().GetMessage() << std::endl;
```

```
            return false;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return true;
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: descrever um grupo de segurança

O exemplo `describe-security-groups` a seguir descreve o grupo de segurança especificado.

```
aws ec2 describe-security-groups \
    --group-ids sg-903004f8
```

Saída:

```
{
    "SecurityGroups": [
        {
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "UserIdGroupPairs": [],
                    "PrefixListIds": []
                }
```

```
            ],
            "Description": "My security group",
            "Tags": [
                {
                    "Value": "SG1",
                    "Key": "Name"
                }
            ],
            "IpPermissions": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [],
                    "UserIdGroupPairs": [
                        {
                            "UserId": "123456789012",
                            "GroupId": "sg-903004f8"
                        }
                    ],
                    "PrefixListIds": []
                },
                {
                    "PrefixListIds": [],
                    "FromPort": 22,
                    "IpRanges": [
                        {
                            "Description": "Access from NY office",
                            "CidrIp": "203.0.113.0/24"
                        }
                    ],
                    "ToPort": 22,
                    "IpProtocol": "tcp",
                    "UserIdGroupPairs": []
                    }
            ],
            "GroupName": "MySecurityGroup",
            "VpcId": "vpc-1a2b3c4d",
            "OwnerId": "123456789012",
            "GroupId": "sg-903004f8",
        }
    ]
}
```

Exemplo 2: descrever grupos de segurança com regras específicas

O exemplo `describe-security-groups` a seguir usa filtros para definir os resultados para grupos de segurança que tenham uma regra que permita tráfego SSH (porta 22) e uma regra que permita tráfego de todos os endereços (`0.0.0.0/0`). O exemplo usa o parâmetro `--query` para exibir somente os nomes dos grupos de segurança. Os grupos de segurança devem corresponder a todos os filtros para serem retornados nos resultados; no entanto, uma única regra não precisa corresponder a todos os filtros. Por exemplo, a saída retorna um grupo de segurança com uma regra que permite o tráfego SSH de um endereço IP específico e outra regra que permite o tráfego HTTP de todos os endereços.

```
aws ec2 describe-security-groups \
    --filters Name=ip-permission.from-port,Values=22 Name=ip-permission.to-
port,Values=22 Name=ip-permission.cidr,Values='0.0.0.0/0' \
    --query "SecurityGroups[*].[GroupName]" \
    --output text
```

Saída:

```
default
my-security-group
web-servers
launch-wizard-1
```

Exemplo 3: descrever grupos de segurança com base em tags

O exemplo a seguir `describe-security-groups` usa filtros para definir o escopo dos resultados para grupos de segurança que incluem `test` no nome do grupo de segurança e contêm a tag `Test=To-delete`. O exemplo usa o parâmetro `--query` para exibir somente os nomes e os IDs dos grupos de segurança.

```
aws ec2 describe-security-groups \
    --filters Name=group-name,Values=*test* Name=tag:Test,Values=To-delete \
    --query "SecurityGroups[*].{Name:GroupName,ID:GroupId}"
```

Saída:

```
[
    {
        "Name": "testfornewinstance",
        "ID": "sg-33bb22aa"
    },
```

```
    {
        "Name": "newgrouptest",
        "ID": "sg-1a2b3c4d"
    }
]
```

Para obter mais exemplos do uso de filtros de tags, consulte Trabalhando com tags no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeSecurityGroupsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
```

```
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
 securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
 groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the
 name: " + groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified security groups or all of your security groups.
 * @param {{ groupIds: string[] }} options
 */
```

```
export const main = async ({ groupIds = [] }) => {
  const client = new EC2Client({});
  const command = new DescribeSecurityGroupsCommand({
    GroupIds: groupIds,
  });

  try {
    const { SecurityGroups } = await client.send(command);
    const sgList = SecurityGroups.map(
      (sg) => `• ${sg.GroupName} (${sg.GroupId}): ${sg.Description}`,
    ).join("\n");
    if (sgList.length) {
      console.log(`Security groups:\n${sgList}`);
    } else {
      console.log("No security groups found.");
    }
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else if (
      caught instanceof Error &&
      caught.name === "InvalidGroup.NotFound"
    ) {
      console.warn(caught.message);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSecurityGroupsreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o grupo de segurança especificado para uma VPC. Ao trabalhar com grupos de segurança pertencentes a uma VPC, você deve usar o ID do grupo de segurança (- GroupId parâmetro), não o nome (- GroupName parâmetro), para referenciar o grupo.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Saída:

```
Description         : default VPC security group
GroupId             : sg-12345678
GroupName           : default
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId             : 123456789012
Tags                : {}
VpcId               : vpc-12345678
```

Exemplo 2: Este exemplo descreve o grupo de segurança especificado para EC2-Classic. Ao trabalhar com grupos de segurança, EC2-Classic você pode usar o nome do grupo (-GroupName parâmetro) ou o ID do grupo (- GroupId parâmetro) para referenciar o grupo de segurança.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Saída:

```
Description         : my security group
GroupId             : sg-45678901
GroupName           : my-security-group
IpPermissions       : {Amazon.EC2.Model.IpPermission,
 Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
OwnerId             : 123456789012
Tags                : {}
VpcId               :
```

Exemplo 3: esse exemplo recupera todos os grupos de segurança da vpc-0fc1ff23456b789eb

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o grupo de segurança especificado para uma VPC. Ao trabalhar com grupos de segurança pertencentes a uma VPC, você deve usar o ID do grupo

de segurança (- GroupId parâmetro), não o nome (- GroupName parâmetro), para referenciar o grupo.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Saída:

```
Description         : default VPC security group
GroupId             : sg-12345678
GroupName           : default
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId             : 123456789012
Tags                : {}
VpcId               : vpc-12345678
```

Exemplo 2: Este exemplo descreve o grupo de segurança especificado para EC2-Classic. Ao trabalhar com grupos de segurança, EC2-Classic você pode usar o nome do grupo (- GroupName parâmetro) ou o ID do grupo (- GroupId parâmetro) para referenciar o grupo de segurança.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Saída:

```
Description         : my security group
GroupId             : sg-45678901
GroupName           : my-security-group
IpPermissions       : {Amazon.EC2.Model.IpPermission,
 Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
OwnerId             : 123456789012
Tags                : {}
VpcId               :
```

Exemplo 3: esse exemplo recupera todos os grupos de segurança da vpc-0fc1ff23456b789eb

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
```

```
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                  If None, an open search is performed to
describe all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security
group(s), such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

            for page in page_iterator:
                for security_group in page["SecurityGroups"]:
                    print(f"Security group: {security_group['GroupName']}")
                    print(f"\tID: {security_group['GroupId']}")
                    print(f"\tVPC: {security_group['VpcId']}")
                    if security_group["IpPermissions"]:
                        print("Inbound permissions:")
                        pp(security_group["IpPermissions"])

            return True
        except ClientError as err:
            logger.error("Failed to describe security group(s).")
            if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group {security_group_id} does not exist "
                    f"because the specified security group ID was not found."
                )
            raise
```

- Para obter detalhes da API, consulte a DescribeSecurityGroupsReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids:
 Vec<String>) {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSecurityGroupsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA lt_group_ids TYPE /aws1/
cl_ec2groupidstrlist_w=>tt_groupidstringlist.
        APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO
 lt_group_ids.
        oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).
         " oo_result is returned for testing purposes. "
        DATA(lt_security_groups) = oo_result->get_securitygroups( ).
        MESSAGE 'Retrieved information about security groups.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeSecurityGroupsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Usar paginação com `describeSecurityGroupsPaginated()`.

```
import AWSEC2

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
    /// - Returns: An array of strings giving the names of every security
    ///   group the user is a member of.
    func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
        let pages = ec2Client.describeSecurityGroupsPaginated(
            input: DescribeSecurityGroupsInput()
        )

        var groupNames: [String] = []

        do {
            for try await page in pages {
                guard let groups = page.securityGroups else {
                    print("*** Error: No groups returned.")
                    continue
                }

                for group in groups {
                    groupNames.append(group.groupName ?? "<unknown>")
                }
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }
```

```
        return groupNames
    }
```

Sem paginação.

```
import AWSEC2

    func describeSecurityGroups(groupId: String) async -> Bool {
        do {
            let output = try await ec2Client.describeSecurityGroups(
                input: DescribeSecurityGroupsInput(
                    groupIds: [groupId]
                )
            )

            guard let securityGroups = output.securityGroups else {
                print("No security groups found.")
                return true
            }

            for group in securityGroups {
                print("Group \(group.groupId ?? "<unknown>") found with VPC
 \(group.vpcId ?? "<unknown>")")
            }
            return true
        } catch {
            print("*** Error getting security group details:
 \(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte DescribeSecurityGroupsa referência da API AWS SDK for Swift.

Usar **`DescribeSnapshotAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeSnapshotAttribute`.

CLI

AWS CLI

Descrever os atributos de um snapshot

O exemplo `describe-snapshot-attribute` a seguir lista as contas com as quais um snapshot é compartilhado.

```
aws ec2 describe-snapshot-attribute \
    --snapshot-id snap-01234567890abcedf \
    --attribute createVolumePermission
```

Saída:

```
{
    "SnapshotId": "snap-01234567890abcedf",
    "CreateVolumePermissions": [
        {
            "UserId": "123456789012"
        }
    ]
}
```

Para obter mais informações, consulte Compartilhar um snapshot do Amazon EBS no Guia do usuário do Amazon Elastic Compute Cloud.

- Para obter detalhes da API, consulte DescribeSnapshotAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o atributo especificado do snapshot especificado.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Saída:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
```

```
{}                          {}              snap-12345678
```

Exemplo 2: esse exemplo descreve o atributo especificado do snapshot especificado.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

Saída:

```
Group    UserId
-----    ------
all
```

- Para obter detalhes da API, consulte DescribeSnapshotAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o atributo especificado do snapshot especificado.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Saída:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

Exemplo 2: esse exemplo descreve o atributo especificado do snapshot especificado.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

Saída:

```
Group    UserId
-----    ------
all
```

- Para obter detalhes da API, consulte DescribeSnapshotAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeSnapshots`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeSnapshots`.

CLI

AWS CLI

Exemplo 1: descrever um snapshot

O exemplo `describe-snapshots` a seguir descreve o snapshot especificado.

```
aws ec2 describe-snapshots \
    --snapshot-ids snap-1234567890abcdef0
```

Saída:

```
{
    "Snapshots": [
        {
            "Description": "This is my snapshot",
            "Encrypted": false,
            "VolumeId": "vol-049df61146c4d7901",
            "State": "completed",
            "VolumeSize": 8,
            "StartTime": "2019-02-28T21:28:32.000Z",
            "Progress": "100%",
            "OwnerId": "012345678910",
            "SnapshotId": "snap-01234567890abcdef",
            "Tags": [
                {
                    "Key": "Stack",
                    "Value": "test"
                }
            ]
        }
    ]
}
```

Para obter mais informações, consulte Snapshots do Amazon EBS no Guia do usuário do Amazon EC2.

Exemplo 2: descrever snapshots com base em filtros

O `describe-snapshots` exemplo a seguir usa filtros para definir o escopo dos resultados para instantâneos pertencentes à sua AWS conta que estão no `pending` estado. O exemplo usa o parâmetro `--query` para exibir somente os IDs do snapshot e a hora em que o snapshot foi iniciado.

```
aws ec2 describe-snapshots \
    --owner-ids self \
    --filters Name=status,Values=pending \
    --query "Snapshots[*].{ID:SnapshotId,Time:StartTime}"
```

Saída:

```
[
    {
        "ID": "snap-1234567890abcdef0",
        "Time": "2019-08-04T12:48:18.000Z"
    },
    {
        "ID": "snap-066877671789bd71b",
        "Time": "2019-08-04T02:45:16.000Z
    },
    ...
]
```

O exemplo `describe-snapshots` a seguir usa filtros para definir o escopo dos resultados para snapshots criados no volume especificado. O exemplo usa o parâmetro `--query` para exibir somente os IDs do snapshot.

```
aws ec2 describe-snapshots \
    --filters Name=volume-id,Values=049df61146c4d7901 \
    --query "Snapshots[*].[SnapshotId]" \
    --output text
```

Saída:

```
snap-1234567890abcdef0
snap-08637175a712c3fb9
...
```

Para obter mais exemplos do uso de filtros, consulte Listar e filtrar seus recursos no Guia do usuário do Amazon EC2.

Exemplo 3: descrever snapshots com base em tags

O exemplo `describe-snapshots` a seguir usa filtros de tag para definir o escopo dos resultados para snapshots que tenham a tag `Stack=Prod`.

```
aws ec2 describe-snapshots \
    --filters Name=tag:Stack,Values=prod
```

Para obter um exemplo da saída de `describe-snapshots`, consulte o Exemplo 1.

Para obter mais exemplos do uso de filtros de tags, consulte Trabalhando com tags no Guia do usuário do Amazon EC2.

Exemplo 4: descrever snapshots com base na idade

O `describe-snapshots` exemplo a seguir usa expressões JMESPath para descrever todos os instantâneos criados pela sua AWS conta antes da data especificada. Ele exibe somente os IDs do snapshot.

```
aws ec2 describe-snapshots \
    --owner-ids 012345678910 \
    --query "Snapshots[?(StartTime<='2020-03-31')].[SnapshotId]"
```

Para obter mais exemplos do uso de filtros, consulte Listar e filtrar seus recursos no Guia do usuário do Amazon EC2.

Exemplo 5: visualizar somente snapshots arquivados

O exemplo `describe-snapshots` a seguir lista apenas os snapshots armazenados no nível de arquivamento.

```
aws ec2 describe-snapshots \
    --filters "Name=storage-tier,Values=archive"
```

Saída:

```
{
    "Snapshots": [
        {
            "Description": "Snap A",
            "Encrypted": false,
```

```
            "VolumeId": "vol-01234567890aaaaaa",
            "State": "completed",
            "VolumeSize": 8,
            "StartTime": "2021-09-07T21:00:00.000Z",
            "Progress": "100%",
            "OwnerId": "123456789012",
            "SnapshotId": "snap-01234567890aaaaaa",
            "StorageTier": "archive",
            "Tags": []
        },
    ]
}
```

Para obter mais informações, consulte View archived snapshots no Guia do usuário do Amazon Elastic Compute Cloud.

- Para obter detalhes da API, consulte DescribeSnapshotsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o snapshot especificado.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Saída:

```
DataEncryptionKeyId :
Description         : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from
 vol-12345678
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            : 100%
SnapshotId          : snap-12345678
StartTime           : 10/23/2014 6:01:28 AM
State               : completed
StateMessage        :
Tags                : {}
```

```
VolumeId            : vol-12345678
VolumeSize          : 8
```

Exemplo 2: esse exemplo descreve os snapshots que têm uma tag “Name”.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Exemplo 3: Este exemplo descreve os instantâneos que têm uma tag 'Nome' com o valor 'TestValue'.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

Exemplo 4: esse exemplo descreve todos os snapshots.

```
Get-EC2Snapshot -Owner self
```

- Para obter detalhes da API, consulte DescribeSnapshotsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o snapshot especificado.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Saída:

```
DataEncryptionKeyId :
Description         : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from
 vol-12345678
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            : 100%
SnapshotId          : snap-12345678
StartTime           : 10/23/2014 6:01:28 AM
State               : completed
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
```

```
VolumeSize          : 8
```

Exemplo 2: esse exemplo descreve os snapshots que têm uma tag “Name”.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Exemplo 3: Este exemplo descreve os instantâneos que têm uma tag 'Nome' com o valor 'TestValue'.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

Exemplo 4: esse exemplo descreve todos os snapshots.

```
Get-EC2Snapshot -Owner self
```

- Para obter detalhes da API, consulte DescribeSnapshotsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Mostra o estado de um snapshot.

```
async fn show_state(client: &Client, id: &str) -> Result<(), Error> {
    let resp = client
        .describe_snapshots()
        .filters(Filter::builder().name("snapshot-id").values(id).build())
        .send()
        .await?;

    println!(
        "State: {}",
```

```
        resp.snapshots().first().unwrap().state().unwrap().as_ref()
    );

    Ok(())
}
```

```
async fn show_snapshots(client: &Client) -> Result<(), Error> {
    // "self" represents your account ID.
    // You can list the snapshots for any account by replacing
    // "self" with that account ID.
    let resp = client.describe_snapshots().owner_ids("self").send().await?;
    let snapshots = resp.snapshots();
    let length = snapshots.len();

    for snapshot in snapshots {
        println!(
            "ID:          {}",
            snapshot.snapshot_id().unwrap_or_default()
        );
        println!(
            "Description: {}",
            snapshot.description().unwrap_or_default()
        );
        println!("State:       {}", snapshot.state().unwrap().as_ref());
        println!();
    }

    println!();
    println!("Found {} snapshot(s)", length);
    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeSnapshotsreferência da API AWS SDK for Rust.

Usar **`DescribeSpotDatafeedSubscription`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeSpotDatafeedSubscription`.

CLI

AWS CLI

Descrever a assinatura do feed de dados da Instância Spot para uma conta

Este exemplo de comando descreve o feed de dados da conta.

Comando:

```
aws ec2 describe-spot-datafeed-subscription
```

Saída:

```
{
    "SpotDatafeedSubscription": {
        "OwnerId": "123456789012",
        "Prefix": "spotdata",
        "Bucket": "amzn-s3-demo-bucket",
        "State": "Active"
    }
}
```

- Para obter detalhes da API, consulte DescribeSpotDatafeedSubscriptionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o feed de dados da instância Spot.

```
Get-EC2SpotDatafeedSubscription
```

Saída:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Para obter detalhes da API, consulte DescribeSpotDatafeedSubscriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o feed de dados da instância Spot.

```
Get-EC2SpotDatafeedSubscription
```

Saída:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Para obter detalhes da API, consulte DescribeSpotDatafeedSubscriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeSpotFleetInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeSpotFleetInstances`.

CLI

AWS CLI

Descrever as Instâncias Spot associadas a uma Frota Spot

Este exemplo de comando lista as Instâncias Spot associadas à Frota Spot especificada.

Comando:

```
aws ec2 describe-spot-fleet-instances --spot-fleet-request-id sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Saída:

```
{
  "ActiveInstances": [
      {
```

```
            "InstanceId": "i-1234567890abcdef0",
            "InstanceType": "m3.medium",
            "SpotInstanceRequestId": "sir-08b93456"
        },
        ...
    ],
    "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

- Para obter detalhes da API, consulte DescribeSpotFleetInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve as instâncias associadas à solicitação de frota spot especificada.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Saída:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- Para obter detalhes da API, consulte DescribeSpotFleetInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as instâncias associadas à solicitação de frota spot especificada.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Saída:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- Para obter detalhes da API, consulte DescribeSpotFleetInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeSpotFleetRequestHistory`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeSpotFleetRequestHistory`.

CLI

AWS CLI

Descrever o histórico da frota Spot

Este comando de exemplo retorna o histórico da frota Spot especificada a partir da hora especificada.

Comando:

```
aws ec2 describe-spot-fleet-request-history --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE --start-time 2015-05-26T00:00:00Z
```

O exemplo de saída a seguir mostra os lançamentos bem-sucedidos de duas Instâncias Spot para a frota Spot.

Saída:

```
{
  "HistoryRecords": [
      {
          "Timestamp": "2015-05-26T23:17:20.697Z",
          "EventInformation": {
              "EventSubType": "submitted"
          },
          "EventType": "fleetRequestChange"
      },
      {
          "Timestamp": "2015-05-26T23:17:20.873Z",
```

```
            "EventInformation": {
                "EventSubType": "active"
            },
            "EventType": "fleetRequestChange"
        },
        {
            "Timestamp": "2015-05-26T23:21:21.712Z",
            "EventInformation": {
                "InstanceId": "i-1234567890abcdef0",
                "EventSubType": "launched"
            },
            "EventType": "instanceChange"
        },
        {
            "Timestamp": "2015-05-26T23:21:21.816Z",
            "EventInformation": {
                "InstanceId": "i-1234567890abcdef1",
                "EventSubType": "launched"
            },
            "EventType": "instanceChange"
        }
    ],
    "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
    "NextToken": "CpHNsscimcV5oH7bSbub03CI2Qms5+ypNpNm
+53MNlR0YcXAkp0xFlfKf91yVxSExmbtma3awYxMFzNA663ZskT0AHtJ6TCb2Z8bQC2EnZgyELbymtWPfpZ1ZbauV
+P+TfGlWxWWB/Vr5dk5d4LfdgA/DRAHUrYgxzrEXAMPLE=",
    "StartTime": "2015-05-26T00:00:00Z"
}
```

- Para obter detalhes da API, consulte DescribeSpotFleetRequestHistoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o histórico de solicitação de frota spot especificada.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Saída:

```
HistoryRecords     : {Amazon.EC2.Model.HistoryRecord,
 Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime  : 12/26/2015 8:29:11 AM
NextToken          :
SpotFleetRequestId : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime          : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

Saída:

```
EventInformation                    EventType             Timestamp
----------------                    ---------             ---------
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:05 AM
```

- Para obter detalhes da API, consulte DescribeSpotFleetRequestHistoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o histórico de solicitação de frota spot especificada.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Saída:

```
HistoryRecords     : {Amazon.EC2.Model.HistoryRecord,
 Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime  : 12/26/2015 8:29:11 AM
NextToken          :
SpotFleetRequestId : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime          : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

Saída:

```
EventInformation                     EventType             Timestamp
----------------                     ---------             ---------
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:05 AM
```

- Para obter detalhes da API, consulte DescribeSpotFleetRequestHistoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeSpotFleetRequests`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeSpotFleetRequests`.

CLI

AWS CLI

Descrever suas solicitações de frota Spot

Este exemplo descreve todas as suas solicitações de frota Spot.

Comando:

```
aws ec2 describe-spot-fleet-requests
```

Saída:

```
{
  "SpotFleetRequestConfigs": [
      {
          "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
          "SpotFleetRequestConfig": {
              "TargetCapacity": 20,
              "LaunchSpecifications": [
                  {
                      "EbsOptimized": false,
                      "NetworkInterfaces": [
                          {
```

```
                            "SubnetId": "subnet-a61dafcf",
                            "DeviceIndex": 0,
                            "DeleteOnTermination": false,
                            "AssociatePublicIpAddress": true,
                            "SecondaryPrivateIpAddressCount": 0
                        }
                    ],
                    "InstanceType": "cc2.8xlarge",
                    "ImageId": "ami-1a2b3c4d"
                },
                {
                    "EbsOptimized": false,
                    "NetworkInterfaces": [
                        {
                            "SubnetId": "subnet-a61dafcf",
                            "DeviceIndex": 0,
                            "DeleteOnTermination": false,
                            "AssociatePublicIpAddress": true,
                            "SecondaryPrivateIpAddressCount": 0
                        }
                    ],
                    "InstanceType": "r3.8xlarge",
                    "ImageId": "ami-1a2b3c4d"
                }
            ],
            "SpotPrice": "0.05",
            "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
        },
        "SpotFleetRequestState": "active"
    },
    {
        "SpotFleetRequestId": "sfr-306341ed-9739-402e-881b-ce47bEXAMPLE",
        "SpotFleetRequestConfig": {
            "TargetCapacity": 20,
            "LaunchSpecifications": [
                {
                    "EbsOptimized": false,
                    "NetworkInterfaces": [
                        {
                            "SubnetId": "subnet-6e7f829e",
                            "DeviceIndex": 0,
                            "DeleteOnTermination": false,
                            "AssociatePublicIpAddress": true,
                            "SecondaryPrivateIpAddressCount": 0
```

```
                        }
                    ],
                    "InstanceType": "m3.medium",
                    "ImageId": "ami-1a2b3c4d"
                }
            ],
            "SpotPrice": "0.05",
            "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
        },
        "SpotFleetRequestState": "active"
    }
  ]
}
```

Descrever uma solicitação de frota spot

Este exemplo descreve a solicitação de frota Spot especificada.

Comando:

```
aws ec2 describe-spot-fleet-requests --spot-fleet-request-ids sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Saída:

```
{
  "SpotFleetRequestConfigs": [
      {
          "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
          "SpotFleetRequestConfig": {
              "TargetCapacity": 20,
              "LaunchSpecifications": [
                  {
                      "EbsOptimized": false,
                      "NetworkInterfaces": [
                          {
                              "SubnetId": "subnet-a61dafcf",
                              "DeviceIndex": 0,
                              "DeleteOnTermination": false,
                              "AssociatePublicIpAddress": true,
                              "SecondaryPrivateIpAddressCount": 0
                          }
                      ],
```

```
                        "InstanceType": "cc2.8xlarge",
                        "ImageId": "ami-1a2b3c4d"
                    },
                    {
                        "EbsOptimized": false,
                        "NetworkInterfaces": [
                            {
                                "SubnetId": "subnet-a61dafcf",
                                "DeviceIndex": 0,
                                "DeleteOnTermination": false,
                                "AssociatePublicIpAddress": true,
                                "SecondaryPrivateIpAddressCount": 0
                            }
                        ],
                        "InstanceType": "r3.8xlarge",
                        "ImageId": "ami-1a2b3c4d"
                    }
                ],
                "SpotPrice": "0.05",
                "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
            },
            "SpotFleetRequestState": "active"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeSpotFleetRequestsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a solicitação de frota spot especificada.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE | format-list
```

Saída:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
```

```
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Exemplo 2: esse exemplo descreve todas as suas solicitações de frota spot.

```
Get-EC2SpotFleetRequest
```

- Para obter detalhes da API, consulte DescribeSpotFleetRequestsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a solicitação de frota spot especificada.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE | format-list
```

Saída:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Exemplo 2: esse exemplo descreve todas as suas solicitações de frota spot.

```
Get-EC2SpotFleetRequest
```

- Para obter detalhes da API, consulte DescribeSpotFleetRequestsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeSpotInstanceRequests`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeSpotInstanceRequests`.

CLI

AWS CLI

Exemplo 1: descrever uma solicitação de Instância Spot

O exemplo `describe-spot-instance-requests` a seguir descreve a Instância Spot especificada.

```
aws ec2 describe-spot-instance-requests \
    --spot-instance-request-ids sir-08b93456
```

Saída:

```
{
    "SpotInstanceRequests": [
        {
            "CreateTime": "2018-04-30T18:14:55.000Z",
            "InstanceId": "i-1234567890abcdef1",
            "LaunchSpecification": {
                "InstanceType": "t2.micro",
                "ImageId": "ami-003634241a8fcdec0",
                "KeyName": "my-key-pair",
                "SecurityGroups": [
                    {
                        "GroupName": "default",
                        "GroupId": "sg-e38f24a7"
                    }
                ],
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sda1",
                        "Ebs": {
                            "DeleteOnTermination": true,
                            "SnapshotId": "snap-0e54a519c999adbbd",
                            "VolumeSize": 8,
                            "VolumeType": "standard",
                            "Encrypted": false
                        }
                    }
                ],
                "NetworkInterfaces": [
                    {
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "SubnetId": "subnet-049df61146c4d7901"
                    }
                ],
                "Placement": {
```

```
                    "AvailabilityZone": "us-east-2b",
                    "Tenancy": "default"
                },
                "Monitoring": {
                    "Enabled": false
                }
            },
            "LaunchedAvailabilityZone": "us-east-2b",
            "ProductDescription": "Linux/UNIX",
            "SpotInstanceRequestId": "sir-08b93456",
            "SpotPrice": "0.010000"
            "State": "active",
            "Status": {
                "Code": "fulfilled",
                "Message": "Your Spot request is fulfilled.",
                "UpdateTime": "2018-04-30T18:16:21.000Z"
            },
            "Tags": [],
            "Type": "one-time",
            "InstanceInterruptionBehavior": "terminate"
        }
    ]
}
```

Exemplo 2: descrever solicitações de Instância Spot com base em filtros

O exemplo `describe-spot-instance-requests` a seguir usa filtros para definir o escopo dos resultados para solicitações de Instâncias Spot com o tipo de instância especificado que também estão na Zona de Disponibilidade especificada. O exemplo usa o parâmetro `--query` para exibir somente os IDs da instância.

```
aws ec2 describe-spot-instance-requests \
    --filters Name=launch.instance-type,Values=m3.medium Name=launched-
availability-zone,Values=us-east-2a \
    --query "SpotInstanceRequests[*].[InstanceId]" \
    --output text
```

Saída:

```
i-057750d42936e468a
i-001efd250faaa6ffa
i-027552a73f021f3bd
```

```
...
```

Para obter mais exemplos do uso de filtros, consulte Listar e filtrar seus recursos no Guia do usuário do Amazon Elastic Computer Cloud.

Exemplo 3: descrever solicitações de Instância Spot com base em tags

O exemplo `describe-spot-instance-requests` a seguir usa filtros de tag para definir o escopo dos resultados das solicitações de Instância Spot que tenham a tag `cost-center=cc123`.

```
aws ec2 describe-spot-instance-requests \
    --filters Name=tag:cost-center,Values=cc123
```

Para obter um exemplo da saída de `describe-spot-instance-requests`, consulte o Exemplo 1.

Para obter mais exemplos do uso de filtros de tags, consulte Trabalhando com tags no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeSpotInstanceRequestsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a solicitação de instância spot especificada.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Saída:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 4/8/2015 2:51:33 PM
Fault                    :
InstanceId               : i-12345678
LaunchedAvailabilityZone : us-west-2b
LaunchGroup              :
```

```
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.020000
State                    : active
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {Name}
Type                     : one-time
```

Exemplo 2: esse exemplo descreve todas as solicitações de instância spot.

```
Get-EC2SpotInstanceRequest
```

- Para obter detalhes da API, consulte DescribeSpotInstanceRequestsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a solicitação de instância spot especificada.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Saída:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 4/8/2015 2:51:33 PM
Fault                    :
InstanceId               : i-12345678
LaunchedAvailabilityZone : us-west-2b
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.020000
State                    : active
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {Name}
Type                     : one-time
```

Exemplo 2: esse exemplo descreve todas as solicitações de instância spot.

```
Get-EC2SpotInstanceRequest
```

- Para obter detalhes da API, consulte DescribeSpotInstanceRequestsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeSpotPriceHistory`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeSpotPriceHistory`.

CLI

AWS CLI

Descrever o histórico de preço Spot

Esse exemplo de comando retorna o histórico de preços spot para instâncias m1.xlarge em um determinado dia de janeiro.

Comando:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --start-
time 2014-01-06T07:08:09 --end-time 2014-01-06T08:09:10
```

Saída:

```
{
  "SpotPriceHistory": [
          {
              "Timestamp": "2014-01-06T07:10:55.000Z",
              "ProductDescription": "SUSE Linux",
              "InstanceType": "m1.xlarge",
              "SpotPrice": "0.087000",
              "AvailabilityZone": "us-west-1b"
          },
          {
              "Timestamp": "2014-01-06T07:10:55.000Z",
              "ProductDescription": "SUSE Linux",
              "InstanceType": "m1.xlarge",
              "SpotPrice": "0.087000",
              "AvailabilityZone": "us-west-1c"
          },
          {
```

```
                "Timestamp": "2014-01-06T05:42:36.000Z",
                "ProductDescription": "SUSE Linux (Amazon VPC)",
                "InstanceType": "m1.xlarge",
                "SpotPrice": "0.087000",
                "AvailabilityZone": "us-west-1a"
        },
        ...
}
```

Para descrever o histórico de preços spot da Linux/UNIX Amazon VPC

Esse exemplo de comando retorna o histórico de preços spot para instâncias m1.xlarge da Amazon Linux/UNIX VPC de um determinado dia de janeiro.

Comando:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --product-
description "Linux/UNIX (Amazon VPC)" --start-time 2014-01-06T07:08:09 --end-
time 2014-01-06T08:09:10
```

Saída:

```
{
  "SpotPriceHistory": [
      {
          "Timestamp": "2014-01-06T04:32:53.000Z",
          "ProductDescription": "Linux/UNIX (Amazon VPC)",
          "InstanceType": "m1.xlarge",
          "SpotPrice": "0.080000",
          "AvailabilityZone": "us-west-1a"
      },
      {
          "Timestamp": "2014-01-05T11:28:26.000Z",
          "ProductDescription": "Linux/UNIX (Amazon VPC)",
          "InstanceType": "m1.xlarge",
          "SpotPrice": "0.080000",
          "AvailabilityZone": "us-west-1c"
      }
  ]
}
```

- Para obter detalhes da API, consulte DescribeSpotPriceHistoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém as últimas 10 entradas no histórico de preços spot do tipo de instância especificado e da zona de disponibilidade. Observe que o valor especificado para o AvailabilityZone parâmetro - deve ser válido para o valor da região fornecido ao parâmetro -Region do cmdlet (não mostrado no exemplo) ou definido como padrão no shell. Este exemplo de comando pressupõe que uma região padrão de “us-west-2” tenha sido definida no ambiente.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Saída:

```
AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:39:49 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:38:29 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 6:57:13 AM
...
```

- Para obter detalhes da API, consulte DescribeSpotPriceHistoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém as últimas 10 entradas no histórico de preços spot do tipo de instância especificado e da zona de disponibilidade. Observe que o valor especificado para

o AvailabilityZone parâmetro - deve ser válido para o valor da região fornecido ao parâmetro -Region do cmdlet (não mostrado no exemplo) ou definido como padrão no shell. Este exemplo de comando pressupõe que uma região padrão de “us-west-2” tenha sido definida no ambiente.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Saída:

```
AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:39:49 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:38:29 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 6:57:13 AM
...
```

- Para obter detalhes da API, consulte DescribeSpotPriceHistoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeSubnets`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeSubnets`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Configurar a conectividade do serviço de contêiner

- Criar uma rede privada virtual básica
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criar uma instância de banco de dados relacional
- Criação e gerenciamento de volumes de armazenamento em bloco
- Introdução aos bancos de dados de documentos
- Introdução ao balanceamento de carga
- Conceitos básicos de streaming gerenciado
- Introdução aos gateways de trânsito de rede

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
```

```
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeSubnetsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: descrever todas as suas sub-redes

O exemplo `describe-subnets` a seguir mostra os detalhes das suas sub-redes.

```
aws ec2 describe-subnets
```

Saída:

```
{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1d",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 4089,
            "CidrBlock": "172.31.80.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": false,
            "MapCustomerOwnedIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-0bb1c79de3EXAMPLE",
            "VpcId": "vpc-0ee975135dEXAMPLE",
            "OwnerId": "111122223333",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "CustomerOwnedIpv4Pool:": 'pool-2EXAMPLE',
            "SubnetArn": "arn:aws:ec2:us-east-2:111122223333:subnet/
subnet-0bb1c79de3EXAMPLE",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            }
        },
        {
            "AvailabilityZone": "us-east-1d",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 4089,
            "CidrBlock": "172.31.80.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-8EXAMPLE",
            "VpcId": "vpc-3EXAMPLE",
            "OwnerId": "1111222233333",
```

```
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "MySubnet"
                }
            ],
            "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-8EXAMPLE",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            }
        }
    ]
}
```

Para obter mais informações, consulte Working with VPCs and Subnets no Guia do usuário da VPC da AWS .

Exemplo 2: descrever as sub-redes de uma VPC específica

O exemplo de `describe-subnets` a seguir usa um filtro para recuperar detalhes das sub-redes da VPC especificada.

```
aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=vpc-3EXAMPLE"
```

Saída:

```
{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1d",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 4089,
            "CidrBlock": "172.31.80.0/20",
            "DefaultForAz": true,
```

```
            "MapPublicIpOnLaunch": true,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-8EXAMPLE",
            "VpcId": "vpc-3EXAMPLE",
            "OwnerId": "1111222233333",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "MySubnet"
                }
            ],
            "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-8EXAMPLE",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            }
        }
    ]
}
```

Para obter mais informações, consulte Working with VPCs and Subnets no Guia do usuário da VPC da AWS .

Exemplo 3: descrever as sub-redes com uma tag específica

O exemplo `describe-subnets` a seguir usa um filtro para recuperar os detalhes dessas sub-redes com a tag `CostCenter=123` e o parâmetro `--query` para exibir os IDs de sub-redes com essa tag.

```
aws ec2 describe-subnets \
    --filters "Name=tag:CostCenter,Values=123" \
    --query "Subnets[*].SubnetId" \
    --output text
```

Saída:

```
subnet-0987a87c8b37348ef
subnet-02a95061c45f372ee
subnet-03f720e7de2788d73
```

Para obter mais informações, consulte Working with VPCs and Subnets no Guia do usuário da Amazon VPC.

- Para obter detalhes da API, consulte DescribeSubnetsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new EC2Client({});
const { Subnets } = await client.send(
  new DescribeSubnetsCommand({
    Filters: [
      { Name: "vpc-id", Values: [state.defaultVpc] },
      { Name: "availability-zone", Values: state.availabilityZoneNames },
      { Name: "default-for-az", Values: ["true"] },
    ],
  }),
);
```

- Para obter detalhes da API, consulte DescribeSubnetsa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a sub-rede especificada.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Saída:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : available
SubnetId                : subnet-1a2b3c4d
Tags                    : {}
VpcId                   : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as sub-redes.

```
Get-EC2Subnet
```

- Para obter detalhes da API, consulte DescribeSubnetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a sub-rede especificada.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Saída:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : available
SubnetId                : subnet-1a2b3c4d
Tags                    : {}
VpcId                   : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as sub-redes.

```
Get-EC2Subnet
```

- Para obter detalhes da API, consulte DescribeSubnetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
```

```
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
```

```
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DescribeSubnetsReferência da API AWS SDK for Python (Boto3).

Usar **`DescribeTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTags`.

CLI

AWS CLI

Exemplo 1: descrever todas as tags de um único recurso

O exemplo `describe-tags` a seguir descreve as tags da instância especificada.

```
aws ec2 describe-tags \
```

```
    --filters "Name=resource-id,Values=i-1234567890abcdef8"
```

Saída:

```
{
    "Tags": [
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Beta Server",
            "Key": "Name"
        }
    ]
}
```

Exemplo 2: descrever todas as tags de um tipo de recurso

O exemplo `describe-tags` a seguir descreve as tags para os volumes.

```
aws ec2 describe-tags \
    --filters "Name=resource-type,Values=volume"
```

Saída:

```
{
    "Tags": [
        {
            "ResourceType": "volume",
            "ResourceId": "vol-1234567890abcdef0",
            "Value": "Project1",
            "Key": "Purpose"
        },
        {
            "ResourceType": "volume",
            "ResourceId": "vol-049df61146c4d7901",
            "Value": "Logs",
```

```
            "Key": "Purpose"
        }
    ]
}
```

Exemplo 3: descrever todas as tags

O exemplo `describe-tags` a seguir descreve as tags de todos os recursos.

```
aws ec2 describe-tags
```

Exemplo 4: descrever as tags dos recursos com base em uma chave de tag

O exemplo `describe-tags` a seguir descreve as tags dos seus recursos que têm uma tag com a chave `Stack`.

```
aws ec2 describe-tags \
    --filters Name=key,Values=Stack
```

Saída:

```
{
    "Tags": [
        {
            "ResourceType": "volume",
            "ResourceId": "vol-027552a73f021f3b",
            "Value": "Production",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        }
    ]
}
```

Exemplo 5: descrever as tags de seus recursos com base na chave e no valor da tag

O exemplo `describe-tags` a seguir descreve as tags dos seus recursos que têm a tag `Stack=Test`.

```
aws ec2 describe-tags \
    --filters Name=key,Values=Stack Name=value,Values=Test
```

Saída:

```
{
    "Tags": [
        {
            "ResourceType": "image",
            "ResourceId": "ami-3ac336533f021f3bd",
            "Value": "Test",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        }
    ]
}
```

O exemplo `describe-tags` a seguir usa uma sintaxe alternativa para descrever recursos com a tag `Stack=Test`.

```
aws ec2 describe-tags \
    --filters "Name=tag:Stack,Values=Test"
```

O exemplo `describe-tags` a seguir descreve as tags de todas as suas instâncias que têm uma tag com a chave `Purpose` e sem valor.

```
aws ec2 describe-tags \
    --filters "Name=resource-
type,Values=instance" "Name=key,Values=Purpose" "Name=value,Values="
```

Saída:

```
{
    "Tags": [
        {
            "ResourceType": "instance",
```

```
            "ResourceId": "i-1234567890abcdef5",
            "Value": null,
            "Key": "Purpose"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo busca as tags do tipo de recurso “image”

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Saída:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

Exemplo 2: esse exemplo busca todas as tags de todos os recursos e as agrupa por tipo de recurso

```
Get-EC2Tag | Group-Object resourcetype
```

Saída:

```
Count Name                      Group
----- ----                      -----
    9 subnet                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   53 instance                  {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    3 route-table               {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
```

```
    5 security-group            {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   30 volume                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    1 internet-gateway          {Amazon.EC2.Model.TagDescription}
    3 network-interface         {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    4 elastic-ip                {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    1 dhcp-options              {Amazon.EC2.Model.TagDescription}
    2 image                     {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    3 vpc                       {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
```

Exemplo 3: esse exemplo exibe todos os recursos com a tag “auto-delete” com o valor “no” para a região em questão

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Saída:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
auto-delete i-0f1bce234d5dd678b   instance     no
auto-delete vol-01d234aa5678901a2 volume       no
auto-delete vol-01234bfb5def6f7b8 volume       no
auto-delete vol-01ccb23f4c5e67890 volume       no
```

Exemplo 4: este exemplo obtém todos os recursos com a tag “exclusão automática” com valor “nenhum” e filtros adicionais no próximo canal para analisar somente os tipos de recursos de “instância” e, eventualmente, cria a tag “ThisInstance” para cada recurso da instância, com o valor sendo o próprio ID da instância

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
 | Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -
ResourceId $_.ResourceId -Tag @{Key="ThisInstance";Value=$_.ResourceId}}
```

Exemplo 5: esse exemplo busca tags para todos os recursos da instância, bem como para as chaves “Name”, e as exibe em formato de tabela

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Saída:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo busca as tags do tipo de recurso “image”

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Saída:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

Exemplo 2: esse exemplo busca todas as tags de todos os recursos e as agrupa por tipo de recurso

```
Get-EC2Tag | Group-Object resourcetype
```

Saída:

```
Count Name                      Group
----- ----                      -----
```

```
    9 subnet                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   53 instance                  {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    3 route-table               {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    5 security-group            {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   30 volume                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    1 internet-gateway          {Amazon.EC2.Model.TagDescription}
    3 network-interface         {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    4 elastic-ip                {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    1 dhcp-options              {Amazon.EC2.Model.TagDescription}
    2 image                     {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    3 vpc                       {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
```

Exemplo 3: esse exemplo exibe todos os recursos com a tag “auto-delete” com o valor “no” para a região em questão

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Saída:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
auto-delete i-0f1bce234d5dd678b   instance     no
auto-delete vol-01d234aa5678901a2 volume       no
auto-delete vol-01234bfb5def6f7b8 volume       no
auto-delete vol-01ccb23f4c5e67890 volume       no
```

Exemplo 4: este exemplo obtém todos os recursos com a tag “exclusão automática” com valor “nenhum” e filtros adicionais no próximo canal para analisar somente os tipos de recursos de

“instância” e, eventualmente, cria a tag “ThisInstance” para cada recurso da instância, com o valor sendo o próprio ID da instância

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
 | Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -
ResourceId $_.ResourceId -Tag @{Key="ThisInstance";Value=$_.ResourceId}}
```

Exemplo 5: esse exemplo busca tags para todos os recursos da instância, bem como para as chaves “Name”, e as exibe em formato de tabela

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Saída:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

Exemplo 6: Este exemplo valida as permissões para obter tags do EC2 usando o DryRun parâmetro sem realmente buscá-las. Nota: se isso for feito com sucesso, será gerada uma exceção, que é o comportamento esperado.

```
Get-EC2Tag -DryRun $true
```

Saída:

```
Get-EC2Tag: Request would have succeeded, but DryRun flag is set.
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVolumeAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVolumeAttribute`.

CLI

AWS CLI

Descrever um atributo de volume

Este exemplo de comando descreve o atributo `autoEnableIo` do volume com o ID `vol-049df61146c4d7901`.

Comando:

```
aws ec2 describe-volume-attribute --volume-id vol-049df61146c4d7901 --
attribute autoEnableIO
```

Saída:

```
{
    "AutoEnableIO": {
        "Value": false
    },
    "VolumeId": "vol-049df61146c4d7901"
}
```

- Para obter detalhes da API, consulte DescribeVolumeAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o atributo especificado do volume especificado.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Saída:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- Para obter detalhes da API, consulte DescribeVolumeAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o atributo especificado do volume especificado.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Saída:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- Para obter detalhes da API, consulte DescribeVolumeAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVolumeStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVolumeStatus`.

CLI

AWS CLI

Descrever o status de um único volume

Este exemplo de comando descreve o status do volume `vol-1234567890abcdef0`.

Comando:

```
aws ec2 describe-volume-status --volume-ids vol-1234567890abcdef0
```

Saída:

```
{
    "VolumeStatuses": [
        {
            "VolumeStatus": {
```

```
                "Status": "ok",
                "Details": [
                    {
                        "Status": "passed",
                        "Name": "io-enabled"
                    },
                    {
                        "Status": "not-applicable",
                        "Name": "io-performance"
                    }
                ]
            },
            "AvailabilityZone": "us-east-1a",
            "VolumeId": "vol-1234567890abcdef0",
            "Actions": [],
            "Events": []
        }
    ]
}
```

Descrever o status dos volumes danificados

Este exemplo de comando descreve o status de todos os volumes que estão danificados. Neste exemplo de saída, não há volumes comprometidos.

Comando:

```
aws ec2 describe-volume-status --filters Name=volume-
status.status,Values=impaired
```

Saída:

```
{
    "VolumeStatuses": []
}
```

Se você tiver um volume com um status de falha em uma verificação de status (o status é comprometido), consulte Trabalhar com um volume comprometido no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeVolumeStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o status do volume especificado.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Saída:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Saída:

```
Details                        Status
-------                        ------
{io-enabled, io-performance}   ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Saída:

```
Name                           Status
----                           ------
io-enabled                     passed
io-performance                 not-applicable
```

- Para obter detalhes da API, consulte DescribeVolumeStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o status do volume especificado.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Saída:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Saída:

```
Details                         Status
-------                         ------
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Saída:

```
Name                            Status
----                            ------
io-enabled                      passed
io-performance                  not-applicable
```

- Para obter detalhes da API, consulte DescribeVolumeStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVolumes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVolumes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criação e gerenciamento de volumes de armazenamento em bloco
- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

CLI

AWS CLI

Exemplo 1: descrever um volume

O exemplo `describe-volumes` a seguir descreve os volumes especificados na região atual.

```
aws ec2 describe-volumes \
    --volume-ids vol-049df61146c4d7901 vol-1234567890abcdef0
```

Saída:

```
{
    "Volumes": [
        {
            "AvailabilityZone": "us-east-1a",
            "Attachments": [
                {
                    "AttachTime": "2013-12-18T22:35:00.000Z",
                    "InstanceId": "i-1234567890abcdef0",
                    "VolumeId": "vol-049df61146c4d7901",
                    "State": "attached",
                    "DeleteOnTermination": true,
                    "Device": "/dev/sda1"
                }
            ],
            "Encrypted": true,
            "KmsKeyId": "arn:aws:kms:us-east-2a:123456789012:key/8c5b2c63-
b9bc-45a3-a87a-5513eEXAMPLE,
            "VolumeType": "gp2",
            "VolumeId": "vol-049df61146c4d7901",
            "State": "in-use",
            "Iops": 100,
            "SnapshotId": "snap-1234567890abcdef0",
            "CreateTime": "2019-12-18T22:35:00.084Z",
            "Size": 8
        },
        {
            "AvailabilityZone": "us-east-1a",
            "Attachments": [],
            "Encrypted": false,
            "VolumeType": "gp2",
```

```
            "VolumeId": "vol-1234567890abcdef0",
            "State": "available",
            "Iops": 300,
            "SnapshotId": "",
            "CreateTime": "2020-02-27T00:02:41.791Z",
            "Size": 100
        }
    ]
}
```

Exemplo 2: descrever volumes anexados a uma instância específica

O exemplo `describe-volumes` a seguir descreve todos os volumes anexados à instância especificada e definidos para serem excluídos quando a instância for encerrada.

```
aws ec2 describe-volumes \
    --region us-east-1 \
    --filters Name=attachment.instance-
id,Values=i-1234567890abcdef0 Name=attachment.delete-on-termination,Values=true
```

Para obter um exemplo da saída de `describe-volumes`, consulte o Exemplo 1.

Exemplo 3: descrever os volumes disponíveis em uma Zona de Disponibilidade específica

O exemplo `describe-volumes` a seguir descreve todos os volumes que têm um status `available` e estão na Zona de Disponibilidade especificada.

```
aws ec2 describe-volumes \
    --filters Name=status,Values=available Name=availability-zone,Values=us-
east-1a
```

Para obter um exemplo da saída de `describe-volumes`, consulte o Exemplo 1.

Exemplo 4: descrever volumes com base em tags

O exemplo `describe-volumes` a seguir descreve todos os volumes que têm a chave de tag `Name` e um valor que começa com `Test`. A saída é, então, filtrada com uma consulta que exibe somente as tags e IDs dos volumes.

```
aws ec2 describe-volumes \
    --filters Name=tag:Name,Values=Test* \
    --query "Volumes[*].{ID:VolumeId,Tag:Tags}"
```

Saída:

```
[
    {
       "Tag": [
           {
               "Value": "Test2",
               "Key": "Name"
           }
       ],
       "ID": "vol-1234567890abcdef0"
   },
   {
       "Tag": [
           {
               "Value": "Test1",
               "Key": "Name"
           }
       ],
       "ID": "vol-049df61146c4d7901"
    }
]
```

Para obter mais exemplos do uso de filtros de tags, consulte Trabalhando com tags no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DescribeVolumesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve volume do EBS especificado.

```
Get-EC2Volume -VolumeId vol-12345678
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 7/17/2015 4:35:19 PM
```

```
Encrypted        : False
Iops             : 90
KmsKeyId         :
Size             : 30
SnapshotId       : snap-12345678
State            : in-use
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : standard
```

Exemplo 2: esse exemplo descreve os volumes do EBS que têm o status “available”.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 12/21/2015 2:31:29 PM
Encrypted        : False
Iops             : 60
KmsKeyId         :
Size             : 20
SnapshotId       : snap-12345678
State            : available
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
...
```

Exemplo 3: esse exemplo descreve todos os volumes do EBS.

```
Get-EC2Volume
```

- Para obter detalhes da API, consulte DescribeVolumesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve volume do EBS especificado.

```
Get-EC2Volume -VolumeId vol-12345678
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 7/17/2015 4:35:19 PM
Encrypted        : False
Iops             : 90
KmsKeyId         :
Size             : 30
SnapshotId       : snap-12345678
State            : in-use
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : standard
```

Exemplo 2: esse exemplo descreve os volumes do EBS que têm o status “available”.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 12/21/2015 2:31:29 PM
Encrypted        : False
Iops             : 60
KmsKeyId         :
Size             : 20
SnapshotId       : snap-12345678
State            : available
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
...
```

Exemplo 3: esse exemplo descreve todos os volumes do EBS.

```
Get-EC2Volume
```

- Para obter detalhes da API, consulte DescribeVolumesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVpcAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVpcAttribute`.

CLI

AWS CLI

Para descrever o DnsSupport atributo enable

Este exemplo descreve o atributo `enableDnsSupport`. Esse atributo indica se a VPC conta com resolução de DNS habilitada. Se este atributo é `true`, o servidor de DNS da Amazon resolve os nomes de hosts DNS de suas instâncias para os endereços IP correspondentes; caso contrário, ele não resolve.

Comando:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute enableDnsSupport
```

Saída:

```
{
    "VpcId": "vpc-a01106c2",
    "EnableDnsSupport": {
        "Value": true
    }
}
```

Para descrever o DnsHostnames atributo enable

Este exemplo descreve o atributo `enableDnsHostnames`. Esse atributo indica se as instâncias executadas na VPC obtêm nomes de host DNS. Se esse atributo é `true`, as instâncias na VPC obtêm os nomes de hosts DNS; caso contrário, isso não ocorrerá.

Comando:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --
attribute enableDnsHostnames
```

Saída:

```
{
```

```
    "VpcId": "vpc-a01106c2",
    "EnableDnsHostnames": {
        "Value": true
    }
}
```

- Para obter detalhes da API, consulte DescribeVpcAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve o atributo DnsSupport 'habilitar'.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Saída:

```
EnableDnsSupport
----------------
True
```

Exemplo 2: Este exemplo descreve o atributo DnsHostnames 'habilitar'.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Saída:

```
EnableDnsHostnames
------------------
True
```

- Para obter detalhes da API, consulte DescribeVpcAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o atributo DnsSupport 'habilitar'.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Saída:

```
EnableDnsSupport
----------------
True
```

Exemplo 2: Este exemplo descreve o atributo DnsHostnames 'habilitar'.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Saída:

```
EnableDnsHostnames
------------------
True
```

- Para obter detalhes da API, consulte DescribeVpcAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVpcClassicLink`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVpcClassicLink`.

CLI

AWS CLI

Para descrever o ClassicLink status de suas VPCs

Este exemplo lista o ClassicLink status de vpc-88888888.

Comando:

```
aws ec2 describe-vpc-classic-link --vpc-id vpc-88888888
```

Saída:

```
{
  "Vpcs": [
    {
      "ClassicLinkEnabled": true,
```

```
      "VpcId": "vpc-88888888",
      "Tags": [
        {
          "Value": "classiclinkvpc",
          "Key": "Name"
        }
      ]
    }
  ]
}
```

Este exemplo lista somente VPCs habilitadas para o Classiclink (o valor do filtro `is-classic-link-enabled` está definido como `true`).

Comando:

```
aws ec2 describe-vpc-classic-link --filter "Name=is-classic-link-enabled,Values=true"
```

- Para obter detalhes da API, consulte DescribeVpcClassicLinkem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: O exemplo acima retorna todas as VPCs com seu ClassicLinkEnabled estado para a região

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Saída:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Para obter detalhes da API, consulte DescribeVpcClassicLinkem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: O exemplo acima retorna todas as VPCs com seu ClassicLinkEnabled estado para a região

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Saída:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Para obter detalhes da API, consulte DescribeVpcClassicLinkem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVpcClassicLinkDnsSupport`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Para descrever o suporte ClassicLink de DNS para suas VPCs

Este exemplo descreve o status de suporte de ClassicLink DNS de todas as suas VPCs.

Comando:

```
aws ec2 describe-vpc-classic-link-dns-support
```

Saída:

```
{
```

```
  "Vpcs": [
    {
      "VpcId": "vpc-88888888",
      "ClassicLinkDnsSupported": true
    },
    {
      "VpcId": "vpc-1a2b3c4d",
      "ClassicLinkDnsSupported": false
    }
  ]
}
```

- Para obter detalhes da API, consulte DescribeVpcClassicLinkDnsSupportem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve o status de suporte de ClassicLink DNS das VPCs para a região eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Saída:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Para obter detalhes da API, consulte DescribeVpcClassicLinkDnsSupportem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o status de suporte de ClassicLink DNS das VPCs para a região eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Saída:

```
ClassicLinkDnsSupported VpcId
------------------------ -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Para obter detalhes da API, consulte DescribeVpcClassicLinkDnsSupportem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVpcEndpointServices`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVpcEndpointServices`.

CLI

AWS CLI

Exemplo 1: descrever todos os serviços de endpoint da VPC

O `describe-vpc-endpoint-services` exemplo a seguir lista todos os serviços de VPC endpoint para uma região. AWS

```
aws ec2 describe-vpc-endpoint-services
```

Saída:

```
{
    "ServiceDetails": [
        {
            "ServiceType": [
                {
                    "ServiceType": "Gateway"
                }
            ],
            "AcceptanceRequired": false,
            "ServiceName": "com.amazonaws.us-east-1.dynamodb",
            "VpcEndpointPolicySupported": true,
            "Owner": "amazon",
            "AvailabilityZones": [
                "us-east-1a",
                "us-east-1b",
                "us-east-1c",
                "us-east-1d",
```

```
                "us-east-1e",
                "us-east-1f"
            ],
            "BaseEndpointDnsNames": [
                "dynamodb.us-east-1.amazonaws.com"
            ]
        },
        {
            "ServiceType": [
                {
                    "ServiceType": "Interface"
                }
            ],
            "PrivateDnsName": "ec2.us-east-1.amazonaws.com",
            "ServiceName": "com.amazonaws.us-east-1.ec2",
            "VpcEndpointPolicySupported": false,
            "Owner": "amazon",
            "AvailabilityZones": [
                "us-east-1a",
                "us-east-1b",
                "us-east-1c",
                "us-east-1d",
                "us-east-1e",
                "us-east-1f"
            ],
            "AcceptanceRequired": false,
            "BaseEndpointDnsNames": [
                "ec2.us-east-1.vpce.amazonaws.com"
            ]
        },
        {
            "ServiceType": [
                {
                    "ServiceType": "Interface"
                }
            ],
            "PrivateDnsName": "ssm.us-east-1.amazonaws.com",
            "ServiceName": "com.amazonaws.us-east-1.ssm",
            "VpcEndpointPolicySupported": true,
            "Owner": "amazon",
            "AvailabilityZones": [
                "us-east-1a",
                "us-east-1b",
                "us-east-1c",
```

```
                "us-east-1d",
                "us-east-1e"
            ],
            "AcceptanceRequired": false,
            "BaseEndpointDnsNames": [
                "ssm.us-east-1.vpce.amazonaws.com"
            ]
        }
    ],
    "ServiceNames": [
        "com.amazonaws.us-east-1.dynamodb",
        "com.amazonaws.us-east-1.ec2",
        "com.amazonaws.us-east-1.ec2messages",
        "com.amazonaws.us-east-1.elasticloadbalancing",
        "com.amazonaws.us-east-1.kinesis-streams",
        "com.amazonaws.us-east-1.s3",
        "com.amazonaws.us-east-1.ssm"
    ]
}
```

Exemplo 2: descrever os detalhes sobre um serviço de endpoint

O exemplo de `describe-vpc-endpoint-services` a seguir lista os detalhes do serviço de endpoint da interface do Amazon S3.

```
aws ec2 describe-vpc-endpoint-services \
    --filter 'Name=service-type,Values=Interface' Name=service-
name,Values=com.amazonaws.us-east-1.s3
```

Saída:

```
{
    "ServiceDetails": [
        {
            "ServiceName": "com.amazonaws.us-east-1.s3",
            "ServiceId": "vpce-svc-081d84efcdEXAMPLE",
            "ServiceType": [
                {
                    "ServiceType": "Interface"
                }
            ],
            "AvailabilityZones": [
                "us-east-1a",
```

```
                "us-east-1b",
                "us-east-1c",
                "us-east-1d",
                "us-east-1e",
            "us-east-1f"
            ],
            "Owner": "amazon",
            "BaseEndpointDnsNames": [
                "s3.us-east-1.vpce.amazonaws.com"
            ],
            "VpcEndpointPolicySupported": true,
            "AcceptanceRequired": false,
            "ManagesVpcEndpoints": false,
            "Tags": []
        }
    ],
    "ServiceNames": [
        "com.amazonaws.us-east-1.s3"
    ]
}
```

Para obter mais informações, consulte Exibir nomes AWS de serviços disponíveis no Guia AWS PrivateLink do usuário.

- Para obter detalhes da API, consulte DescribeVpcEndpointServicesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o serviço de endpoint da VPC do EC2 com o filtro fornecido, nesse caso com.amazonaws.eu-west-1.ecs. Além disso, ele também expande a ServiceDetails propriedade e exibe os detalhes

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

Saída:

```
AcceptanceRequired         : False
```

```
AvailabilityZones           : {eu-west-1a, eu-west-1b, eu-west-1c}
BaseEndpointDnsNames        : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                       : amazon
PrivateDnsName              : ecs.eu-west-1.amazonaws.com
ServiceName                 : com.amazonaws.eu-west-1.ecs
ServiceType                 : {Amazon.EC2.Model.ServiceTypeDetail}
VpcEndpointPolicySupported : False
```

Exemplo 2: Este exemplo recupera todos os serviços do EC2 VPC Endpoint e retorna o “ssm” correspondente ServiceNames

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

Saída:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- Para obter detalhes da API, consulte DescribeVpcEndpointServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o serviço de endpoint da VPC do EC2 com o filtro fornecido, nesse caso com.amazonaws.eu-west-1.ecs. Além disso, ele também expande a ServiceDetails propriedade e exibe os detalhes

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

Saída:

```
AcceptanceRequired          : False
AvailabilityZones           : {eu-west-1a, eu-west-1b, eu-west-1c}
BaseEndpointDnsNames        : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                       : amazon
PrivateDnsName              : ecs.eu-west-1.amazonaws.com
ServiceName                 : com.amazonaws.eu-west-1.ecs
ServiceType                 : {Amazon.EC2.Model.ServiceTypeDetail}
```

```
VpcEndpointPolicySupported : False
```

Exemplo 2: Este exemplo recupera todos os serviços do EC2 VPC Endpoint e retorna o “ssm” correspondente ServiceNames

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

Saída:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- Para obter detalhes da API, consulte DescribeVpcEndpointServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVpcEndpoints`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVpcEndpoints`.

CLI

AWS CLI

Descrever seus endpoint da VPC

O exemplo `describe-vpc-endpoints` a seguir exibe detalhes de todos os seus endpoints da VPC.

```
aws ec2 describe-vpc-endpoints
```

Saída:

```
{
    "VpcEndpoints": [
        {
            "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":
[{\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"*\",\"Resource\":\"*
\"}]}",
            "VpcId": "vpc-aabb1122",
            "NetworkInterfaceIds": [],
```

```
            "SubnetIds": [],
            "PrivateDnsEnabled": true,
            "State": "available",
            "ServiceName": "com.amazonaws.us-east-1.dynamodb",
            "RouteTableIds": [
                "rtb-3d560345"
            ],
            "Groups": [],
            "VpcEndpointId": "vpce-032a826a",
            "VpcEndpointType": "Gateway",
            "CreationTimestamp": "2017-09-05T20:41:28Z",
            "DnsEntries": [],
            "OwnerId": "123456789012"
        },
        {
            "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\":
\"*\", \n      \"Effect\": \"Allow\", \n      \"Principal\": \"*\", \n
\"Resource\": \"*\"\n    }\n  ]\n}",
            "VpcId": "vpc-1a2b3c4d",
            "NetworkInterfaceIds": [
                "eni-2ec2b084",
                "eni-1b4a65cf"
            ],
            "SubnetIds": [
                "subnet-d6fcaa8d",
                "subnet-7b16de0c"
            ],
            "PrivateDnsEnabled": false,
            "State": "available",
            "ServiceName": "com.amazonaws.us-east-1.elasticloadbalancing",
            "RouteTableIds": [],
            "Groups": [
                {
                    "GroupName": "default",
                    "GroupId": "sg-54e8bf31"
                }
            ],
            "VpcEndpointId": "vpce-0f89a33420c1931d7",
            "VpcEndpointType": "Interface",
            "CreationTimestamp": "2017-09-05T17:55:27.583Z",
            "DnsEntries": [
                {
                    "HostedZoneId": "Z7HUB22UULQXV",
```

```
                    "DnsName": "vpce-0f89a33420c1931d7-
bluzidnv.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
                },
                {
                    "HostedZoneId": "Z7HUB22UULQXV",
                    "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1b.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
                },
                {
                    "HostedZoneId": "Z7HUB22UULQXV",
                    "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1a.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
                }
            ],
            "OwnerId": "123456789012"
        },
        {
            "VpcEndpointId": "vpce-aabbaabbaabbaabba",
            "VpcEndpointType": "GatewayLoadBalancer",
            "VpcId": "vpc-111122223333aabbc",
            "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-
svc-123123a1c43abc123",
            "State": "available",
            "SubnetIds": [
                "subnet-0011aabbcc2233445"
            ],
            "RequesterManaged": false,
            "NetworkInterfaceIds": [
                "eni-01010120203030405"
            ],
            "CreationTimestamp": "2020-11-11T08:06:03.522Z",
            "Tags": [],
            "OwnerId": "123456789012"
        }
    ]
}
```

Para obter mais informações, consulte Conceitos no Guia AWS PrivateLink do usuário.

- Para obter detalhes da API, consulte DescribeVpcEndpointsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve um ou mais dos endpoints da VPC para a região eu-west-1. Em seguida, ele canaliza a saída para o próximo comando, que seleciona a VpcEndpointId propriedade e retorna o ID da VPC da matriz como matriz de seqüências

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
 VpcEndpointId
```

Saída:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Exemplo 2: Este exemplo descreve todos os endpoints vpc da região eu-west-1 e seleciona VpcEndpointId,, e as propriedades para apresentá-los em formato VpcId tabular ServiceName PrivateDnsEnabled

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Saída:

```
VpcEndpointId          VpcId                 ServiceName
 PrivateDnsEnabled
-------------          -----                 -----------
 -----------------
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
             True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
             True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
             True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
             True
```

Exemplo 3: esse exemplo exporta o documento de política do endpoint da VPC vpce-01a2ab3f4f5cc6f7d em um arquivo json

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
 Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Para obter detalhes da API, consulte DescribeVpcEndpointsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve um ou mais dos endpoints da VPC para a região eu-west-1. Em seguida, ele canaliza a saída para o próximo comando, que seleciona a VpcEndpointId propriedade e retorna o ID da VPC da matriz como matriz de seqüências

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
 VpcEndpointId
```

Saída:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Exemplo 2: Este exemplo descreve todos os endpoints vpc da região eu-west-1 e seleciona VpcEndpointId,, e as propriedades para apresentá-los em formato VpcId tabular ServiceName PrivateDnsEnabled

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Saída:

```
VpcEndpointId          VpcId                 ServiceName
 PrivateDnsEnabled
-------------          -----                 -----------
 -----------------
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
             True
```

```
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
             True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
             True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
             True
```

Exemplo 3: esse exemplo exporta o documento de política do endpoint da VPC vpce-01a2ab3f4f5cc6f7d em um arquivo json

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
 Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Para obter detalhes da API, consulte DescribeVpcEndpointsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeVpcs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeVpcs`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Configurar a conectividade do serviço de contêiner
- Criar uma rede privada virtual básica
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criar uma instância de banco de dados relacional
- Criação e gerenciamento de uma malha de rede de serviços
- Criação e gerenciamento de volumes de armazenamento em bloco
- Introdução aos bancos de dados de documentos
- Introdução ao balanceamento de carga
- Conceitos básicos de streaming gerenciado
- Introdução aos gateways de trânsito de rede
- Rede privada virtual com gerenciamento de endereços IP

- Trabalho com conexões de emparelhamento de rede

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeVpcsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: descrever todas as suas VPCs

O exemplo `describe-vpcs` a seguir recupera detalhes das suas VPCs.

```
aws ec2 describe-vpcs
```

Saída:

```
{
    "Vpcs": [
        {
            "CidrBlock": "30.1.0.0/16",
            "DhcpOptionsId": "dopt-19edf471",
            "State": "available",
            "VpcId": "vpc-0e9801d129EXAMPLE",
            "OwnerId": "111122223333",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-062c64cfafEXAMPLE",
                    "CidrBlock": "30.1.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
```

```
                {
                    "Key": "Name",
                    "Value": "Not Shared"
                }
            ]
        },
        {
            "CidrBlock": "10.0.0.0/16",
            "DhcpOptionsId": "dopt-19edf471",
            "State": "available",
            "VpcId": "vpc-06e4ab6c6cEXAMPLE",
            "OwnerId": "222222222222",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
                    "CidrBlock": "10.0.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Shared VPC"
                }
            ]
        }
    ]
}
```

Exemplo 2: descrever uma VPC especificada

O exemplo de `describe-vpcs` a seguir recupera detalhes da VPC especificada.

```
aws ec2 describe-vpcs \
    --vpc-ids vpc-06e4ab6c6cEXAMPLE
```

Saída:

```
{
```

```
    "Vpcs": [
        {
            "CidrBlock": "10.0.0.0/16",
            "DhcpOptionsId": "dopt-19edf471",
            "State": "available",
            "VpcId": "vpc-06e4ab6c6cEXAMPLE",
            "OwnerId": "111122223333",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
                    "CidrBlock": "10.0.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Shared VPC"
                }
            ]
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeVpcsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
```

```
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
```

- Para obter detalhes da API, consulte DescribeVpcsa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a VPC especificada.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Saída:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

Exemplo 2: esse exemplo descreve a VPC padrão (só pode haver uma por região). Se sua conta for compatível EC2-Classic com essa região, não há VPC padrão.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

Saída:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
```

```
VpcId           : vpc-45678901
```

Exemplo 3: Este exemplo descreve as VPCs que correspondem ao filtro especificado (ou seja, têm um CIDR que corresponde ao valor '10.0.0). 0/16'e estão no estado' disponível ').

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Exemplo 4: esse exemplo descreve todas as VPCs.

```
Get-EC2Vpc
```

- Para obter detalhes da API, consulte DescribeVpcsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a VPC especificada.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Saída:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

Exemplo 2: esse exemplo descreve a VPC padrão (só pode haver uma por região). Se sua conta for compatível EC2-Classic com essa região, não há VPC padrão.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

Saída:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
```

```
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

Exemplo 3: Este exemplo descreve as VPCs que correspondem ao filtro especificado (ou seja, têm um CIDR que corresponde ao valor '10.0.0). 0/16'e estão no estado' disponível ').

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Exemplo 4: esse exemplo descreve todas as VPCs.

```
Get-EC2Vpc
```

- Para obter detalhes da API, consulte DescribeVpcsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
```

```
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.
```

```
        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
 "
                    "Ensure that your AWS IAM user or role has the correct
 permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
 parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

- Para obter detalhes da API, consulte a DescribeVpcsReferência da API AWS SDK for Python (Boto3).

Usar **`DescribeVpnConnections`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVpnConnections`.

CLI

AWS CLI

Exemplo 1: descrever as conexões VPN

O `describe-vpn-connections` exemplo a seguir descreve todas as suas conexões Site-to-Site VPN.

```
aws ec2 describe-vpn-connections
```

Saída:

```
{
    "VpnConnections": [
        {
            "CustomerGatewayConfiguration": "...configuration information...",
            "CustomerGatewayId": "cgw-01234567abcde1234",
            "Category": "VPN",
            "State": "available",
            "Type": "ipsec.1",
            "VpnConnectionId": "vpn-1122334455aabbccd",
            "TransitGatewayId": "tgw-00112233445566aab",
            "Options": {
                "EnableAcceleration": false,
                "StaticRoutesOnly": true,
                "LocalIpv4NetworkCidr": "0.0.0.0/0",
                "RemoteIpv4NetworkCidr": "0.0.0.0/0",
                "TunnelInsideIpVersion": "ipv4"
            },
            "Routes": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "CanadaVPN"
                }
            ],
            "VgwTelemetry": [
                {
                    "AcceptedRouteCount": 0,
                    "LastStatusChange": "2020-07-29T10:35:11.000Z",
                    "OutsideIpAddress": "203.0.113.3",
                    "Status": "DOWN",
```

```
                    "StatusMessage": ""
                },
                {
                    "AcceptedRouteCount": 0,
                    "LastStatusChange": "2020-09-02T09:09:33.000Z",
                    "OutsideIpAddress": "203.0.113.5",
                    "Status": "UP",
                    "StatusMessage": ""
                }
            ]
        }
    ]
}
```

Para obter mais informações, consulte Como a AWS Site-to-Site VPN funciona no Guia do usuário da AWS Site-to-Site VPN.

Exemplo 2: descrever as conexões VPN disponíveis

O `describe-vpn-connections` exemplo a seguir descreve suas conexões Site-to-Site VPN com um estado de`available`.

```
aws ec2 describe-vpn-connections \
    --filters "Name=state,Values=available"
```

Para obter mais informações, consulte Como a AWS Site-to-Site VPN funciona no Guia do usuário da AWS Site-to-Site VPN.

- Para obter detalhes da API, consulte DescribeVpnConnectionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve conexão VPN especificada.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Saída:

```
CustomerGatewayConfiguration : [XML document]
```

```
CustomerGatewayId            : cgw-1a2b3c4d
Options                      : Amazon.EC2.Model.VpnConnectionOptions
Routes                       : {Amazon.EC2.Model.VpnStaticRoute}
State                        : available
Tags                         : {}
Type                         : ipsec.1
VgwTelemetry                 : {Amazon.EC2.Model.VgwTelemetry,
 Amazon.EC2.Model.VgwTelemetry}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo descreve qualquer conexão VPN cujo estado esteja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Exemplo 3: esse exemplo descreve todas as conexões VPN.

```
Get-EC2VpnConnection
```

- Para obter detalhes da API, consulte DescribeVpnConnectionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve conexão VPN especificada.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Saída:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      : Amazon.EC2.Model.VpnConnectionOptions
Routes                       : {Amazon.EC2.Model.VpnStaticRoute}
State                        : available
Tags                         : {}
```

```
Type                          : ipsec.1
VgwTelemetry                  : {Amazon.EC2.Model.VgwTelemetry,
 Amazon.EC2.Model.VgwTelemetry}
VpnConnectionId               : vpn-12345678
VpnGatewayId                  : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo descreve qualquer conexão VPN cujo estado esteja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Exemplo 3: esse exemplo descreve todas as conexões VPN.

```
Get-EC2VpnConnection
```

- Para obter detalhes da API, consulte DescribeVpnConnectionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeVpnGateways`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeVpnGateways`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de conexões de rede dedicadas

CLI

AWS CLI

Descrever seus gateways privados virtuais

Este exemplo descreve seus gateways privados virtuais.

Comando:

```
aws ec2 describe-vpn-gateways
```

Saída:

```
{
    "VpnGateways": [
        {
            "State": "available",
            "Type": "ipsec.1",
            "VpnGatewayId": "vgw-f211f09b",
            "VpcAttachments": [
                {
                    "State": "attached",
                    "VpcId": "vpc-98eb5ef5"
                }
            ]
        },
        {
            "State": "available",
            "Type": "ipsec.1",
            "VpnGatewayId": "vgw-9a4cacf3",
            "VpcAttachments": [
                {
                    "State": "attaching",
                    "VpcId": "vpc-a01106c2"
                }
            ]
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeVpnGatewaysem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o gateway privado virtual especificado.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Saída:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo descreve qualquer gateway privado virtual cujo estado seja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Exemplo 3: esse exemplo descreve todos os seus gateways privados virtuais.

```
Get-EC2VpnGateway
```

- Para obter detalhes da API, consulte DescribeVpnGatewaysem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o gateway privado virtual especificado.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Saída:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo descreve qualquer gateway privado virtual cujo estado seja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Exemplo 3: esse exemplo descreve todos os seus gateways privados virtuais.

```
Get-EC2VpnGateway
```

- Para obter detalhes da API, consulte DescribeVpnGatewaysem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DetachInternetGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DetachInternetGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos
- Rede privada virtual com servidores privados

CLI

AWS CLI

Separar um gateway da Internet da VPC

O exemplo `detach-internet-gateway` a seguir separa o gateway da internet especificado da VPC específica.

```
aws ec2 detach-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE \
    --vpc-id vpc-0a60eb65b4EXAMPLE
```

Este comando não produz saída.

Para obter mais informações, consulte Gateways da Internet no Guia do usuário da Amazon VPC.

- Para obter detalhes da API, consulte DetachInternetGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo desanexa o gateway da internet especificado da VPC especificada.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte DetachInternetGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desanexa o gateway da internet especificado da VPC especificada.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte DetachInternetGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DetachNetworkInterface`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DetachNetworkInterface`.

CLI

AWS CLI

Desanexar uma interface de rede de uma instância

Este exemplo desvincula a interface de rede especificada da instância especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 detach-network-interface --attachment-id eni-attach-66c4350a
```

- Para obter detalhes da API, consulte DetachNetworkInterfaceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove o anexo especificado entre uma interface de rede e uma instância.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Para obter detalhes da API, consulte DetachNetworkInterfaceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o anexo especificado entre uma interface de rede e uma instância.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Para obter detalhes da API, consulte DetachNetworkInterfaceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DetachVolume`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DetachVolume`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criação e gerenciamento de volumes de armazenamento em bloco
- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

CLI

AWS CLI

Desanexar um volume de uma instância

Este exemplo de comando separa o volume (`vol-049df61146c4d7901`) da instância à qual ele está conectado.

Comando:

```
aws ec2 detach-volume --volume-id vol-1234567890abcdef0
```

Saída:

```
{
    "AttachTime": "2014-02-27T19:23:06.000Z",
    "InstanceId": "i-1234567890abcdef0",
    "VolumeId": "vol-049df61146c4d7901",
    "State": "detaching",
    "Device": "/dev/sdb"
}
```

- Para obter detalhes da API, consulte DetachVolumeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo desvincula o volume especificado.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Saída:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

Exemplo 2: você também pode especificar o ID da instância e o nome do dispositivo para garantir que esteja desanexando o volume correto.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- Para obter detalhes da API, consulte DetachVolumeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desvincula o volume especificado.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Saída:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

Exemplo 2: você também pode especificar o ID da instância e o nome do dispositivo para garantir que esteja desanexando o volume correto.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- Para obter detalhes da API, consulte DetachVolumeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DetachVpnGateway`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DetachVpnGateway`.

CLI

AWS CLI

Desanexar um gateway privado virtual da VPC

Este exemplo desvincula o gateway privado virtual especificado da VPC especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 detach-vpn-gateway --vpn-gateway-id vgw-9a4cacf3 --vpc-id vpc-a01106c2
```

- Para obter detalhes da API, consulte DetachVpnGatewayem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo desvincula o gateway privado virtual especificado da VPC especificada.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte DetachVpnGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desvincula o gateway privado virtual especificado da VPC especificada.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- Para obter detalhes da API, consulte DetachVpnGatewayem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DisableVgwRoutePropagation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DisableVgwRoutePropagation`.

CLI

AWS CLI

Desabilitar a propagação de rotas

Esse exemplo desabilita o gateway privado virtual especificado de propagar rotas estáticas para a tabela de rotas especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 disable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-
id vgw-9a4cacf3
```

- Para obter detalhes da API, consulte DisableVgwRoutePropagationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo impede que o VGW propague automaticamente as rotas para a tabela de rotas especificada.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte DisableVgwRoutePropagationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo impede que o VGW propague automaticamente as rotas para a tabela de rotas especificada.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte DisableVgwRoutePropagationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DisableVpcClassicLink`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DisableVpcClassicLink`.

CLI

AWS CLI

Para desativar ClassicLink para uma VPC

Este exemplo desativa ClassicLink para vpc-8888888.

Comando:

```
aws ec2 disable-vpc-classic-link --vpc-id vpc-88888888
```

Saída:

```
{
  "Return": true
}
```

- Para obter detalhes da API, consulte DisableVpcClassicLinkem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo é desativado EC2VpcClassicLink para o vpc-01e23c4a5d6db78e9. Ele retorna Verdadeiro ou Falso

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Para obter detalhes da API, consulte DisableVpcClassicLinkem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo é desativado EC2VpcClassicLink para o vpc-01e23c4a5d6db78e9. Ele retorna Verdadeiro ou Falso

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Para obter detalhes da API, consulte DisableVpcClassicLinkem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar `DisableVpcClassicLinkDnsSupport` com uma CLI

Os exemplos de código a seguir mostram como usar o `DisableVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Para desativar o suporte ClassicLink de DNS para uma VPC

Este exemplo desativa o suporte de ClassicLink DNS para. `vpc-88888888`

Comando:

```
aws ec2 disable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

Saída:

```
{
  "Return": true
}
```

- Para obter detalhes da API, consulte DisableVpcClassicLinkDnsSupportem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo desativa o suporte de ClassicLink DNS para o vpc-0b12d3456a7e8910d

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Para obter detalhes da API, consulte DisableVpcClassicLinkDnsSupportem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo desativa o suporte de ClassicLink DNS para o vpc-0b12d3456a7e8910d

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Para obter detalhes da API, consulte DisableVpcClassicLinkDnsSupportem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DisassociateAddress`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DisassociateAddress`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução às máquinas virtuais

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId =
associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
```

```
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while disassociating the Elastic IP.:
 {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DisassociateAddressa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic
 Compute Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a association_id - The association ID that represents the association of
 the Elastic IP address with an instance.
#
# And:
#       0 - If successful.
#       1 - If it fails.
#
```

```
###############################################################################
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the
 association of the Elastic IP address with an instance."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) association_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
  }
```

```
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
```

```
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DisassociateAddressem Referência de AWS CLI Comandos.

CLI

AWS CLI

Para desassociar um endereço IP elástico em EC2-Classic

Este exemplo dissocia um endereço IP elástico de uma instância em. EC2-Classic Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 disassociate-address --public-ip 198.51.100.0
```

Para desassociar um endereço IP elástico em EC2-VPC

Este exemplo desassocia um endereço IP elástico de uma instância em uma VPC. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 disassociate-address --association-id eipassoc-2bebb745
```

- Para obter detalhes da API, consulte DisassociateAddressem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Disassociates an Elastic IP address from an instance asynchronously.
    *
    * @param associationId The ID of the association you want to disassociate.
    * @return a {@link CompletableFuture} representing the asynchronous
operation of disassociating the address. The
    *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
    *         finished.
    * @throws RuntimeException if the disassociation of the address fails.
    */
   public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
       Ec2AsyncClient ec2 = getAsyncClient();
       DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
           .associationId(associationId)
           .build();

       // Disassociate the address asynchronously.
       CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              throw new RuntimeException("Failed to disassociate address", ex);
           }
       });

       return response;
   }
```

- Para obter detalhes da API, consulte DisassociateAddressa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DisassociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Disassociate an Elastic IP address from an instance.
 * @param {{ associationId: string }} options
 */
export const main = async ({ associationId }) => {
  const client = new EC2Client({});
  const command = new DisassociateAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AssociationId: associationId,
  });

  try {
    await client.send(command);
    console.log("Successfully disassociated address");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAssociationID.NotFound"
    ) {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DisassociateAddressa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- Para obter detalhes da API, consulte a DisassociateAddressreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo desassocia o endereço IP elástico especificado da instância especificada em uma VPC.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Exemplo 2: Este exemplo dissocia o endereço IP elástico especificado da instância especificada em. EC2-Classic

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Para obter detalhes da API, consulte DisassociateAddressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desassocia o endereço IP elástico especificado da instância especificada em uma VPC.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Exemplo 2: Este exemplo dissocia o endereço IP elástico especificado da instância especificada em. EC2-Classic

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Para obter detalhes da API, consulte DisassociateAddressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""
```

```
        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance.
 When the
        association is removed, the instance is assigned a new public IP address.
```

```
        :param allocation_id: The allocation ID of the Elastic IP to
disassociate.
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(
                f"No association found for Elastic IP with allocation ID
{allocation_id}."
            )
            return

        try:
            # Retrieve the association ID before disassociating
            response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
            association_id = response["Addresses"][0].get("AssociationId")

            if association_id:

self.ec2_client.disassociate_address(AssociationId=association_id)
                elastic_ip.instance_id = None  # Remove the instance association
            else:
                logger.info(
                    f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
                )

        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
                logger.error(
                    f"Failed to disassociate Elastic IP {allocation_id} "
                    "because the specified association ID for the Elastic IP
address was not found. "
                    "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                    "resource before attempting to disassociate it."
                )
            raise
```

- Para obter detalhes da API, consulte a DisassociateAddressReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn disassociate_ip_address(&self, association_id: &str) ->
Result<(), EC2Error> {
       self.client
           .disassociate_address()
           .association_id(association_id)
           .send()
           .await?;
       Ok(())
   }
```

- Para obter detalhes da API, consulte a DisassociateAddressreferência da API AWS SDK for Rust.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2
```

```
    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
            )
        } catch {
            print("*** Unable to disassociate the IP address:
\(error.localizedDescription)")
        }
    }
```

- Para obter detalhes da API, consulte DisassociateAddressa referência da API AWS SDK for Swift.

Usar **`DisassociateRouteTable`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DisassociateRouteTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Trabalho com conexões de emparelhamento de rede

CLI

AWS CLI

Desassociar uma tabela de rotas

Este exemplo dissocia a tabela de rotas especificada da sub-rede especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 disassociate-route-table --association-id rtbassoc-781d0d1a
```

- Para obter detalhes da API, consulte DisassociateRouteTableem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a associação especificada entre uma tabela de rotas e uma sub-rede.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Para obter detalhes da API, consulte DisassociateRouteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a associação especificada entre uma tabela de rotas e uma sub-rede.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Para obter detalhes da API, consulte DisassociateRouteTableem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`EnableVgwRoutePropagation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EnableVgwRoutePropagation`.

CLI

AWS CLI

Habilitar a propagação de rotas

Esse exemplo habilita o gateway privado virtual especificado para propagar rotas estáticas para a tabela de rotas especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 enable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-
id vgw-9a4cacf3
```

- Para obter detalhes da API, consulte EnableVgwRoutePropagationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo habilita o VGW especificado a propagar automaticamente as rotas para a tabela de rotas especificada.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte EnableVgwRoutePropagationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o VGW especificado a propagar automaticamente as rotas para a tabela de rotas especificada.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte EnableVgwRoutePropagationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`EnableVolumeIo`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EnableVolumeIo`.

CLI

AWS CLI

I/O Para habilitar um volume

Este exemplo ativa I/O o volume`vol-1234567890abcdef0`.

Comando:

```
aws ec2 enable-volume-io --volume-id vol-1234567890abcdef0
```

Saída:

```
{
  "Return": true
}
```

- Para obter detalhes da API, consulte EnableVolumeIoem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo habilita I/O operações para o volume especificado, se I/O as operações estiverem desativadas.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Para obter detalhes da API, consulte EnableVolumeIoem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo habilita I/O operações para o volume especificado, se I/O as operações estiverem desativadas.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Para obter detalhes da API, consulte EnableVolumeIoem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`EnableVpcClassicLink`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EnableVpcClassicLink`.

CLI

AWS CLI

Para habilitar uma VPC para ClassicLink

Este exemplo habilita o vpc-8888888 para. ClassicLink

Comando:

```
aws ec2 enable-vpc-classic-link --vpc-id vpc-88888888
```

Saída:

```
{
  "Return": true
}
```

- Para obter detalhes da API, consulte EnableVpcClassicLinkem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo habilita a VPC vpc-0123456b789b0d12f para ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Saída:

```
True
```

- Para obter detalhes da API, consulte EnableVpcClassicLinkem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo habilita a VPC vpc-0123456b789b0d12f para ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Saída:

```
True
```

- Para obter detalhes da API, consulte EnableVpcClassicLinkem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`EnableVpcClassicLinkDnsSupport`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EnableVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Para habilitar o suporte de ClassicLink DNS para uma VPC

Este exemplo habilita o suporte de ClassicLink DNS para`vpc-88888888`.

Comando:

```
aws ec2 enable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

Saída:

```
{
  "Return": true
}
```

- Para obter detalhes da API, consulte EnableVpcClassicLinkDnsSupportem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo permite que o vpc-0b12d3456a7e8910d ofereça suporte à resolução de nome de host DNS para ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Para obter detalhes da API, consulte EnableVpcClassicLinkDnsSupportem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo permite que o vpc-0b12d3456a7e8910d ofereça suporte à resolução de nome de host DNS para ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Para obter detalhes da API, consulte EnableVpcClassicLinkDnsSupportem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetConsoleOutput`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetConsoleOutput`.

CLI

AWS CLI

Exemplo 1: obter a saída do console

O exemplo `get-console-output` a seguir obtém a saída do console para a instância Linux especificada.

```
aws ec2 get-console-output \
    --instance-id i-1234567890abcdef0
```

Saída:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-07-25T21:23:53.000Z",
    "Output": "..."
}
```

Para obter mais informações, consulte Saída de console da instância no Guia do usuário Amazon EC2.

Exemplo 2: obter a saída mais recente do console

O exemplo `get-console-output` a seguir obtém a saída mais recente do console para a instância Linux especificada.

```
aws ec2 get-console-output \
    --instance-id i-1234567890abcdef0 \
    --latest \
    --output text
```

Saída:

```
i-1234567890abcdef0 [    0.000000] Command line: root=LABEL=/ console=tty1
 console=ttyS0 selinux=0 nvme_core.io_timeout=4294967295
[    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point
 registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
...
Cloud-init v. 0.7.6 finished at Wed, 09 May 2018 19:01:13 +0000. Datasource
 DataSourceEc2.  Up 21.50 seconds
Amazon Linux AMI release 2018.03
Kernel 4.14.26-46.32.amzn1.x
```

Para obter mais informações, consulte Saída de console da instância no Guia do usuário Amazon EC2.

- Para obter detalhes da API, consulte GetConsoleOutputem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém a saída do console para a instância Linux especificada. A saída do console é codificada.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Saída:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Exemplo 2: esse exemplo armazena a saída codificada do console em uma variável e, depois, a decodifica.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded
```

- Para obter detalhes da API, consulte GetConsoleOutputem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém a saída do console para a instância Linux especificada. A saída do console é codificada.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Saída:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Exemplo 2: esse exemplo armazena a saída codificada do console em uma variável e, depois, a decodifica.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded
```

- Para obter detalhes da API, consulte GetConsoleOutputem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetHostReservationPurchasePreview`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetHostReservationPurchasePreview`.

CLI

AWS CLI

Obter uma prévia da compra de uma reserva de host dedicado

Este exemplo fornece uma prévia dos custos de uma Reserva de Host Dedicado especificada para o Host Dedicado especificado em sua conta.

Comando:

```
aws ec2 get-host-reservation-purchase-preview --offering-id hro-03f707bf363b6b324
 --host-id-set h-013abcd2a00cbd123
```

Saída:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
      {
          "HourlyPrice": "1.499",
          "InstanceFamily": "m4",
          "PaymentOption": "NoUpfront",
          "HostIdSet": [
              "h-013abcd2a00cbd123"
          ],
          "UpfrontPrice": "0.000",
          "Duration": 31536000
      }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- Para obter detalhes da API, consulte GetHostReservationPurchasePreviewem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo permite visualizar uma compra de reserva com configurações que correspondem às do seu host dedicado h-01e23f4cd567890f1

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

Saída:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- Para obter detalhes da API, consulte GetHostReservationPurchasePreviewem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo permite visualizar uma compra de reserva com configurações que correspondem às do seu host dedicado h-01e23f4cd567890f1

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

Saída:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- Para obter detalhes da API, consulte GetHostReservationPurchasePreviewem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetPasswordData`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetPasswordData`.

CLI

AWS CLI

Obter a senha criptografada

Este exemplo obtém a senha criptografada.

Comando:

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0
```

Saída:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-08-07T22:18:38.000Z",
    "PasswordData": "gSlJFq+VpcZXqy+iktxMF6NyxQ4qCrT4+gaOuNOenX1MmgXPTj7XEXAMPLE
UQ+YeFfb+L1U4C4AKv652Ux1iRB3CPTYP7WmU3TUnhsuBd+p6LVk7T2lKUml6OXbk6WPW1VYYm/TRPB1
e1DQ7PY4an/DgZT4mwcpRFigzhniQgDDeO1InvSDcwoUTwNs0Y1S8ouri2W4n5GNlriM3Q0AnNVelVz/
53TkDtxbNoU606M1gK9zUWSxqEgwvbV2j8c5rP0WCuaMWSFl4ziDu4bd7q+4RSyi8NUsVWnKZ4aEZffu
DPGzKrF5yLlf3etP2L4ZR6CvG7K1hx7VKOQVN32Dajw=="
}
```

Obter a senha descriptografada

Este exemplo obtém a senha descriptografada.

Comando:

```
aws ec2 get-password-data --instance-id  i-1234567890abcdef0 --priv-launch-key C:
\Keys\MyKeyPair.pem
```

Saída:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-08-30T23:18:05.000Z",
    "PasswordData": "&ViJ652e*u"
}
```

- Para obter detalhes da API, consulte GetPasswordDataem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

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

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

                Usage:
                   <instanceId>

                Where:
                   instanceId - An instance id value that you can obtain from the
 AWS Management Console.\s
             """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String instanceId = args[0];
```

```
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
instanceId);
            future.join();
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Fetches the password data for the specified EC2 instance asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the request
     * @param instanceId instanceId the ID of the EC2 instance for which you want
to fetch the password data
     * @return a {@link CompletableFuture} that completes when the password data
has been fetched
     * @throws RuntimeException if there was a failure in fetching the password
data
     */
    public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
ec2AsyncClient, String instanceId) {
        GetPasswordDataRequest getPasswordDataRequest =
GetPasswordDataRequest.builder()
            .instanceId(instanceId)
            .build();


        CompletableFuture<GetPasswordDataResponse> response =
ec2AsyncClient.getPasswordData(getPasswordDataRequest);
        response.whenComplete((getPasswordDataResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get password data for
instance: " + instanceId, ex);
            } else if (getPasswordDataResponse == null ||
getPasswordDataResponse.passwordData().isEmpty()) {
                throw new RuntimeException("No password data found for instance:
" + instanceId);
            } else {
```

```
                String encryptedPasswordData =
 getPasswordDataResponse.passwordData();
                System.out.println("Encrypted Password Data: " +
 encryptedPasswordData);
            }
        });

        return response.thenApply(resp -> null);
    }
}
```

- Para obter detalhes da API, consulte GetPasswordDataa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descriptografa a senha que o Amazon EC2 atribuiu à conta do administrador para a instância especificada do Windows. Quando um arquivo pem foi especificado, a configuração da opção -Decrypt é automaticamente assumida.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Saída:

```
mYZ(PA9?C)Q
```

Exemplo 2: (PowerShell somente para Windows) inspeciona a instância para determinar o nome do par de chaves usado para iniciar a instância e, em seguida, tenta encontrar os dados do par de chaves correspondente no repositório de configuração do AWS Toolkit for Visual Studio. Se os dados do par de chaves forem encontrados, a senha será descriptografada.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Saída:

```
mYZ(PA9?C)Q
```

Exemplo 3: retorna os dados da senha criptografada para a instância.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Saída:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Para obter detalhes da API, consulte GetPasswordDataem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descriptografa a senha que o Amazon EC2 atribuiu à conta do administrador para a instância especificada do Windows. Quando um arquivo pem foi especificado, a configuração da opção -Decrypt é automaticamente assumida.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Saída:

```
mYZ(PA9?C)Q
```

Exemplo 2: (PowerShell somente para Windows) inspeciona a instância para determinar o nome do par de chaves usado para iniciar a instância e, em seguida, tenta encontrar os dados do par de chaves correspondente no repositório de configuração do AWS Toolkit for Visual Studio. Se os dados do par de chaves forem encontrados, a senha será descriptografada.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Saída:

```
mYZ(PA9?C)Q
```

Exemplo 3: retorna os dados da senha criptografada para a instância.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Saída:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Para obter detalhes da API, consulte GetPasswordDataem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ImportImage`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ImportImage`.

CLI

AWS CLI

Importar um arquivo de imagem da VM como uma AMI

O exemplo `import-image` a seguir importa o OVA especificado.

```
aws ec2 import-image \
  --disk-containers Format=ova,UserBucket="{S3Bucket=my-import-bucket,S3Key=vms/
my-server-vm.ova}"
```

Saída:

```
{
    "ImportTaskId": "import-ami-1234567890abcdef0",
    "Progress": "2",
    "SnapshotDetails": [
        {
            "DiskImageSize": 0.0,
            "Format": "ova",
            "UserBucket": {
                "S3Bucket": "my-import-bucket",
                "S3Key": "vms/my-server-vm.ova"
            }
        }
    ],
    "Status": "active",
    "StatusMessage": "pending"
}
```

- Para obter detalhes da API, consulte ImportImageem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo importa uma imagem de máquina virtual de disco único do bucket do Amazon S3 especificado para o Amazon EC2 com um token de idempotência. O exemplo exige que exista um perfil de serviço de importação de VM com o nome padrão “vmimport”, com uma política que permita que o Amazon EC2 acesse o bucket especificado, conforme explicado no tópico Pré-requisitos de importação da VM. Para usar um perfil personalizado, especifique o nome do perfil usando o parâmetro **-RoleName**.

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

Saída:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- Para obter detalhes da API, consulte ImportImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo importa uma imagem de máquina virtual de disco único do bucket do Amazon S3 especificado para o Amazon EC2 com um token de idempotência. O exemplo exige que exista um perfil de serviço de importação de VM com o nome padrão “vmimport”, com uma política que permita que o Amazon EC2 acesse o bucket especificado, conforme explicado no tópico Pré-requisitos de importação da VM. Para usar um perfil personalizado, especifique o nome do perfil usando o parâmetro **-RoleName**.

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

Saída:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- Para obter detalhes da API, consulte ImportImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ImportKeyPair`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ImportKeyPair`.

CLI

AWS CLI

Importar uma chave pública

Primeiro, gere um par de chaves com uma ferramenta de sua escolha. Por exemplo, use este comando ssh-keygen:

Comando:

```
ssh-keygen -t rsa -C "my-key" -f ~/.ssh/my-key
```

Saída:

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/my-key.
Your public key has been saved in /home/ec2-user/.ssh/my-key.pub.
...
```

Este exemplo de comando importa a chave pública especificada.

Comando:

```
aws ec2 import-key-pair --key-name "my-key" --public-key-material fileb://~/.ssh/
my-key.pub
```

Saída:

```
{
  "KeyName": "my-key",
  "KeyFingerprint": "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca"
}
```

- Para obter detalhes da API, consulte ImportKeyPairem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo importa uma chave pública para o EC2. A primeira linha armazena o conteúdo do arquivo de chave pública (*.pub) na variável **$publickey**. Em seguida, o exemplo converte o formato UTF8 do arquivo de chave pública em uma Base64-encoded string e armazena a string convertida na variável. **$pkbase64** Na última linha, a chave pública convertida é importada para o EC2. O cmdlet retorna a impressão digital e o nome da chave como resultados.

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
$pkbase64 =
 [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

Saída:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- Para obter detalhes da API, consulte ImportKeyPairem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo importa uma chave pública para o EC2. A primeira linha armazena o conteúdo do arquivo de chave pública (*.pub) na variável **$publickey**. Em seguida, o exemplo converte o formato UTF8 do arquivo de chave pública em uma Base64-encoded string e armazena a string convertida na variável. **$pkbase64** Na última linha, a chave pública convertida é importada para o EC2. O cmdlet retorna a impressão digital e o nome da chave como resultados.

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
$pkbase64 =
 [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

Saída:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- Para obter detalhes da API, consulte ImportKeyPairem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ImportSnapshot`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ImportSnapshot`.

CLI

AWS CLI

Importar um snapshot

O exemplo `import-snapshot` a seguir importa o disco especificado como um snapshot.

```
aws ec2 import-snapshot \
    --description "My server VMDK" \
    --disk-container Format=VMDK,UserBucket={'S3Bucket=my-import-
bucket,S3Key=vms/my-server-vm.vmdk'}
```

Saída:

```
{
    "Description": "My server VMDK",
    "ImportTaskId": "import-snap-1234567890abcdef0",
    "SnapshotTaskDetail": {
        "Description": "My server VMDK",
        "DiskImageSize": "0.0",
        "Format": "VMDK",
        "Progress": "3",
        "Status": "active",
        "StatusMessage": "pending"
        "UserBucket": {
            "S3Bucket": "my-import-bucket",
            "S3Key": "vms/my-server-vm.vmdk"
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte ImportSnapshotem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo importa uma imagem de disco de VM no formato “VMDK” para um snapshot do Amazon EBS. O exemplo requer uma função de serviço de importação de VM com o nome padrão 'vmimport', com uma política que permita que o Amazon EC2 acesse o bucket especificado, conforme explicado no tópico http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html. em Para usar uma função personalizada, especifique **`VM Import Prequisites`** o nome da função usando o parâmetro. **`-RoleName`**

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

Saída:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Para obter detalhes da API, consulte ImportSnapshotem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo importa uma imagem de disco de VM no formato “VMDK” para um snapshot do Amazon EBS. O exemplo requer uma função de serviço de importação de VM com o nome padrão 'vmimport', com uma política que permita que o Amazon EC2 acesse o bucket especificado, conforme explicado no tópico http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html. em Para usar uma função personalizada, especifique **`VM Import Prequisites`** o nome da função usando o parâmetro. **`-RoleName`**

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

Saída:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Para obter detalhes da API, consulte ImportSnapshotem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyCapacityReservation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyCapacityReservation`.

CLI

AWS CLI

Exemplo 1: alterar o número de instâncias reservadas por uma reserva de capacidade existente

O exemplo `modify-capacity-reservation` a seguir altera o número de instâncias para as quais a reserva de capacidade reserva a capacidade.

```
aws ec2 modify-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE \
    --instance-count 5
```

Saída:

```
{
    "Return": true
}
```

Consulte mais informações em Modify a Capacity Reservation no Manual do usuário do Amazon EC2.

Exemplo 2: alterar a data e a hora de término de uma reserva de capacidade existente

O exemplo `modify-capacity-reservation` a seguir modifica uma reserva de capacidade existente para terminar na data e na hora especificadas.

```
aws ec2 modify-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE \
    --end-date-type limited \
    --end-date 2019-08-31T23:59:59Z
```

Consulte mais informações em Modify a Capacity Reservation no Manual do usuário do Amazon EC2.

- Para obter detalhes da API, consulte ModifyCapacityReservationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo modifica o CapacityReservationId cr-0c1f2345db6f7cdba alterando a contagem de instâncias para 1

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Saída:

```
True
```

- Para obter detalhes da API, consulte ModifyCapacityReservationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo modifica o CapacityReservationId cr-0c1f2345db6f7cdba alterando a contagem de instâncias para 1

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Saída:

```
True
```

- Para obter detalhes da API, consulte ModifyCapacityReservationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyHosts`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyHosts`.

CLI

AWS CLI

Exemplo 1: habilitar o posicionamento automático para um Host Dedicado

O exemplo `modify-hosts` a seguir habilita o posicionamento automático de um Host dedicado, para que ele aceite todas as execuções de instâncias não direcionadas que correspondam à configuração do tipo de instância.

```
aws ec2 modify-hosts \
    --host-id h-06c2f189b4EXAMPLE \
    --auto-placement on
```

Saída:

```
{
    "Successful": [
        "h-06c2f189b4EXAMPLE"
    ],
    "Unsuccessful": []
}
```

Consulte mais informações em Modificar a configuração de posicionamento automático para um host dedicado no Manual do usuário do Amazon EC2.

Exemplo 2: habilitar a recuperação de host para um host dedicado

O exemplo `modify-hosts` a seguir habilita a recuperação de host para o Host Dedicado especificado.

```
aws ec2 modify-hosts \
    --host-id h-06c2f189b4EXAMPLE \
    --host-recovery on
```

Saída:

```
{
    "Successful": [
        "h-06c2f189b4EXAMPLE"
    ],
    "Unsuccessful": []
}
```

Consulte mais informações em Modificar a configuração de posicionamento automático para um host dedicado no Manual do usuário do Amazon EC2.

- Para obter detalhes da API, consulte ModifyHostsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo modifica as AutoPlacement configurações para desativadas para o host dedicado h-01e23f4cd567890f3

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Saída:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- Para obter detalhes da API, consulte ModifyHostsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo modifica as AutoPlacement configurações para desativadas para o host dedicado h-01e23f4cd567890f3

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Saída:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- Para obter detalhes da API, consulte ModifyHostsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyIdFormat`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyIdFormat`.

CLI

AWS CLI

Habilitar o formato de ID mais longo para um recurso

O exemplo `modify-id-format` a seguir ativa o formato de ID mais longo para o tipo de recurso `instance`.

```
aws ec2 modify-id-format \
    --resource instance \
    --use-long-ids
```

Desativar o formato de ID mais longo para um recurso

O exemplo `modify-id-format` a seguir desativa o formato de ID mais longo para o tipo de recurso `instance`.

```
aws ec2 modify-id-format \
    --resource instance \
    --no-use-long-ids
```

O exemplo `modify-id-format` a seguir habilita o formato mais longo de ID para todos os tipos de recursos compatíveis que estejam no período de inclusão.

```
aws ec2 modify-id-format \
    --resource all-current \
    --use-long-ids
```

- Para obter detalhes da API, consulte ModifyIdFormatem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo habilita o formato de ID mais longo para o tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Exemplo 2: esse exemplo desativa o formato de ID mais longo para o tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Para obter detalhes da API, consulte ModifyIdFormatem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o formato de ID mais longo para o tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Exemplo 2: esse exemplo desativa o formato de ID mais longo para o tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Para obter detalhes da API, consulte ModifyIdFormatem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyImageAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyImageAttribute`.

CLI

AWS CLI

Exemplo 1: tornar uma AMI pública

O exemplo `modify-instance-attribute` a seguir torna pública a AMI especificada.

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Add=[{Group=all}]"
```

Este comando não produz saída.

Exemplo 2: tornar uma AMI privada

O exemplo `modify-instance-attribute` a seguir torna a AMI especificada privada.

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Remove=[{Group=all}]"
```

Este comando não produz saída.

Exemplo 3: para conceder permissão de lançamento a uma AWS conta

O `modify-instance-attribute` exemplo a seguir concede permissões de lançamento à AWS conta especificada.

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Add=[{UserId=123456789012}]"
```

Este comando não produz saída.

Exemplo 4: Para remover a permissão de lançamento de uma AWS conta

O `modify-instance-attribute` exemplo a seguir remove as permissões de lançamento da AWS conta especificada.

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Remove=[{UserId=123456789012}]"
```

- Para obter detalhes da API, consulte ModifyImageAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza a descrição da AMI especificada.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Exemplo 2: Esse exemplo torna a AMI pública (por exemplo, para que qualquer Conta da AWS pessoa possa usá-la).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

Exemplo 3: esse exemplo torna a AMI privada (por exemplo, para que somente você, como proprietário, possa usá-la).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

Exemplo 4: Este exemplo concede permissão de lançamento ao especificado Conta da AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

Exemplo 5: Este exemplo remove a permissão de lançamento do especificado Conta da AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- Para obter detalhes da API, consulte ModifyImageAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza a descrição da AMI especificada.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Exemplo 2: Esse exemplo torna a AMI pública (por exemplo, para que qualquer Conta da AWS pessoa possa usá-la).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

Exemplo 3: esse exemplo torna a AMI privada (por exemplo, para que somente você, como proprietário, possa usá-la).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

Exemplo 4: Este exemplo concede permissão de lançamento ao especificado Conta da AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

Exemplo 5: Este exemplo remove a permissão de lançamento do especificado Conta da AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- Para obter detalhes da API, consulte ModifyImageAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyInstanceAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyInstanceAttribute`.

CLI

AWS CLI

Exemplo 1: modificar o tipo de instância

O exemplo `modify-instance-attribute` a seguir modifica o tipo de instância da instância especificada. A instância deve estar no estado `stopped`.

```
aws ec2 modify-instance-attribute \
    --instance-id i-1234567890abcdef0 \
    --instance-type "{\"Value\": \"m1.small\"}"
```

Este comando não produz saída.

Exemplo 2: habilitar a rede aprimorada em uma instância

O exemplo `modify-instance-attribute` a seguir ativa a rede aprimorada para a instância especificada. A instância deve estar no estado `stopped`.

```
aws ec2 modify-instance-attribute \
    --instance-id i-1234567890abcdef0 \
    --sriov-net-support simple
```

Este comando não produz saída.

Exemplo 3: Para modificar o DestCheck atributo de origem

O exemplo `modify-instance-attribute` a seguir define o atributo `sourceDestCheck` para a instância especificada para `true`. A instância deve estar em uma VPC.

```
aws ec2 modify-instance-attribute --instance-id i-1234567890abcdef0 --source-
dest-check "{\"Value\": true}"
```

Este comando não produz saída.

Exemplo 4: Para modificar o OnTermination atributo delete do volume raiz

O `modify-instance-attribute` exemplo a seguir define o `deleteOnTermination` atributo para o volume raiz da EBS-backed instância da Amazon especificada como`false`. Por padrão, esse atributo é `true` para o volume raiz.

Comando:

```
aws ec2 modify-instance-attribute \
  --instance-id i-1234567890abcdef0 \
  --block-device-mappings "[{\"DeviceName\": \"/dev/sda1\",\"Ebs\":
{\"DeleteOnTermination\":false}}]"
```

Este comando não produz saída.

Exemplo 5: modificar os dados do usuário anexados a uma instância

O `modify-instance-attribute` exemplo a seguir adiciona o conteúdo do arquivo `UserData.txt` como UserData o da instância especificada.

Conteúdo do arquivo `UserData.txt` original:

```
#!/bin/bash
yum update -y
service httpd start
```

```
chkconfig httpd on
```

O conteúdo do arquivo deve ser codificado em base64. O primeiro comando converte o arquivo de texto em base64 e o salva como um novo arquivo.

Linux/macOS versão do comando:

```
base64 UserData.txt > UserData.base64.txt
```

Este comando não produz saída.

Versão Windows do comando:

```
certutil -encode UserData.txt tmp.b64 && findstr /v /c:- tmp.b64 >
 UserData.base64.txt
```

Saída:

```
Input Length = 67
Output Length = 152
CertUtil: -encode command completed successfully.
```

Agora você pode referenciar esse arquivo no comando CLI a seguir:

```
aws ec2 modify-instance-attribute \
    --instance-id=i-09b5a14dbca622e76 \
    --attribute userData --value file://UserData.base64.txt
```

Este comando não produz saída.

Para obter mais informações, consulte Dados do usuário e a AWS CLI no Guia do usuário do EC2.

- Para obter detalhes da API, consulte ModifyInstanceAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica o tipo de instância da instância especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Exemplo 2: Este exemplo habilita a rede aprimorada para a instância especificada, especificando “simple” como o valor do parâmetro de suporte de rede de I/O virtualização (SR-IOV) raiz única, -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Exemplo 3: esse exemplo modifica os grupos de segurança da instância especificada. A instância deve estar em uma VPC. Você deve especificar o ID de cada grupo de segurança, não o nome.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

Exemplo 4: Esse exemplo permite a I/O otimização do EBS para a instância especificada. Esse recurso não está disponível com todos os tipos de instâncias. Cobranças de uso adicionais se aplicam ao usar uma EBS-optimized instância.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Exemplo 5: Este exemplo permite source/destination verificar a instância especificada. Para que a instância execute a NAT, o valor deve ser “false”.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Exemplo 6: esse exemplo desabilita o encerramento da instância especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Exemplo 7: esse exemplo altera a instância especificada para que ela seja encerrada quando o desligamento for iniciado por meio da instância.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -
InstanceInitiatedShutdownBehavior terminate
```

- Para obter detalhes da API, consulte ModifyInstanceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica o tipo de instância da instância especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Exemplo 2: Este exemplo habilita a rede aprimorada para a instância especificada, especificando “simple” como o valor do parâmetro de suporte de rede de I/O virtualização (SR-IOV) raiz única, -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Exemplo 3: esse exemplo modifica os grupos de segurança da instância especificada. A instância deve estar em uma VPC. Você deve especificar o ID de cada grupo de segurança, não o nome.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

Exemplo 4: Esse exemplo permite a I/O otimização do EBS para a instância especificada. Esse recurso não está disponível com todos os tipos de instâncias. Cobranças de uso adicionais se aplicam ao usar uma EBS-optimized instância.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Exemplo 5: Este exemplo permite source/destination verificar a instância especificada. Para que a instância execute a NAT, o valor deve ser “false”.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Exemplo 6: esse exemplo desabilita o encerramento da instância especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Exemplo 7: esse exemplo altera a instância especificada para que ela seja encerrada quando o desligamento for iniciado por meio da instância.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -
InstanceInitiatedShutdownBehavior terminate
```

- Para obter detalhes da API, consulte ModifyInstanceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyInstanceCreditSpecification`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyInstanceCreditSpecification`.

CLI

AWS CLI

Modificar a opção de crédito para uso de CPU de uma instância

Este exemplo modifica a opção de crédito para uso da CPU da instância especificada na região especificada para “ilimitado”. As opções de crédito válidas são “padrão” e “ilimitado”.

Comando:

```
aws ec2 modify-instance-credit-specification --instance-credit-
specification "InstanceId=i-1234567890abcdef0,CpuCredits=unlimited"
```

Saída:

```
{
  "SuccessfulInstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef0"
    }
  ],
  "UnsuccessfulInstanceCreditSpecifications": []
}
```

- Para obter detalhes da API, consulte ModifyInstanceCreditSpecificationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: isso habilita créditos ilimitados de T2, por exemplo, i-01234567890abcdef.

```
$Credit = New-Object -TypeName
 Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Para obter detalhes da API, consulte ModifyInstanceCreditSpecificationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: isso habilita créditos ilimitados de T2, por exemplo, i-01234567890abcdef.

```
$Credit = New-Object -TypeName
 Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Para obter detalhes da API, consulte ModifyInstanceCreditSpecificationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyNetworkInterfaceAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyNetworkInterfaceAttribute`.

CLI

AWS CLI

Modificar o atributo de anexo de uma interface de rede

Este exemplo de comando modifica o atributo `attachment` da interface de rede especificada.

Comando:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
attachment AttachmentId=eni-attach-43348162,DeleteOnTermination=false
```

Modificar o atributo de descrição de uma interface de rede

Este exemplo de comando modifica o atributo `description` da interface de rede especificada.

Comando:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
description "My description"
```

Modificar o atributo groupSet de uma interface de rede

Este exemplo de comando modifica o atributo `groupSet` da interface de rede especificada.

Comando:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
groups sg-903004f8 sg-1a2b3c4d
```

Para modificar o DestCheck atributo de origem de uma interface de rede

Este exemplo de comando modifica o atributo `sourceDestCheck` da interface de rede especificada.

Comando:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
no-source-dest-check
```

- Para obter detalhes da API, consulte ModifyNetworkInterfaceAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica a interface de rede especificada para que o anexo especificado seja excluído no encerramento.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Exemplo 2: esse exemplo modifica a descrição da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description
 "my description"
```

Exemplo 3: esse exemplo modifica o grupo de segurança da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

Exemplo 4: Este exemplo desativa a source/destination verificação da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck $false
```

- Para obter detalhes da API, consulte ModifyNetworkInterfaceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica a interface de rede especificada para que o anexo especificado seja excluído no encerramento.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Exemplo 2: esse exemplo modifica a descrição da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description
 "my description"
```

Exemplo 3: esse exemplo modifica o grupo de segurança da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

Exemplo 4: Este exemplo desativa a source/destination verificação da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck $false
```

- Para obter detalhes da API, consulte ModifyNetworkInterfaceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyReservedInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyReservedInstances`.

CLI

AWS CLI

Modificar instâncias reservadas

Esse exemplo de comando move uma instância reservada para outra zona de disponibilidade na mesma região.

Comando:

```
aws ec2 modify-reserved-instances --reserved-instances-ids b847fa93-e282-4f55-b59a-1342f5bd7c02 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-Classic,InstanceCount=10
```

Saída:

```
{
  "ReservedInstancesModificationId": "rimod-d3ed4335-b1d3-4de6-ab31-0f13aaf46687"
}
```

Modificar a plataforma de rede das Instâncias Reservadas

Este exemplo de comando converte instâncias EC2-Classic reservadas em. EC2-VPC

Comando:

```
aws ec2 modify-reserved-instances --reserved-instances-ids f127bd27-edb7-44c9-a0eb-0d7e09259af0 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-VPC,InstanceCount=5
```

Saída:

```
{
```

```
  "ReservedInstancesModificationId": "rimod-82fa9020-668f-4fb6-945d-61537009d291"
}
```

Para obter mais informações, consulte Modificar Instâncias Reservadas no Guia do usuário do Amazon EC2.

Modificar o tamanho de instância de instâncias reservadas

Este exemplo de comando modifica uma instância reservada que tem 10 instâncias m1.small em us-west-1c para que 8 Linux/UNIX instâncias m1.small se tornem 2 instâncias m1.large e as 2 m1.small restantes se tornem 1 instância m1.medium na mesma zona de disponibilidade. Comando:

```
aws ec2 modify-reserved-instances --reserved-instances-
ids 1ba8e2e3-3556-4264-949e-63ee671405a9 --target-
configurations AvailabilityZone=us-west-1c,Platform=EC2-
Classic,InstanceCount=2,InstanceType=m1.large AvailabilityZone=us-
west-1c,Platform=EC2-Classic,InstanceCount=1,InstanceType=m1.medium
```

Saída:

```
{
    "ReservedInstancesModificationId": "rimod-acc5f240-080d-4717-
b3e3-1c6b11fa00b6"
}
```

Para obter mais informações, consulte Modificar o tamanho da instância de suas reservas no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte ModifyReservedInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica a zona de disponibilidade, a contagem de instâncias e a plataforma das instâncias reservadas especificadas.

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
```

```
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"

Edit-EC2ReservedInstance `
-ReservedInstancesId @("FE32132D-70D5-4795-B400-AE435EXAMPLE",
 "0CC556F3-7AB8-4C00-B0E5-98666EXAMPLE") `
-TargetConfiguration $config
```

- Para obter detalhes da API, consulte ModifyReservedInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica a zona de disponibilidade, a contagem de instâncias e a plataforma das instâncias reservadas especificadas.

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"

Edit-EC2ReservedInstance `
-ReservedInstancesId @("FE32132D-70D5-4795-B400-AE435EXAMPLE",
 "0CC556F3-7AB8-4C00-B0E5-98666EXAMPLE") `
-TargetConfiguration $config
```

- Para obter detalhes da API, consulte ModifyReservedInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifySnapshotAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifySnapshotAttribute`.

CLI

AWS CLI

Exemplo 1: modificar um atributo de snapshot

O exemplo `modify-snapshot-attribute` a seguir atualiza o atributo `createVolumePermission` do snapshot especificado, removendo as permissões de volume do usuário especificado.

```
aws ec2 modify-snapshot-attribute \
    --snapshot-id snap-1234567890abcdef0 \
    --attribute createVolumePermission \
    --operation-type remove \
    --user-ids 123456789012
```

Exemplo 2: tornar público um snapshot

O exemplo `modify-snapshot-attribute` a seguir torna público o snapshot especificado.

```
aws ec2 modify-snapshot-attribute \
    --snapshot-id snap-1234567890abcdef0 \
    --attribute createVolumePermission \
    --operation-type add \
    --group-names all
```

- Para obter detalhes da API, consulte ModifySnapshotAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo torna público o snapshot especificado definindo seu CreateVolumePermission atributo.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- Para obter detalhes da API, consulte ModifySnapshotAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo torna público o snapshot especificado definindo seu CreateVolumePermission atributo.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- Para obter detalhes da API, consulte ModifySnapshotAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifySpotFleetRequest`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifySpotFleetRequest`.

CLI

AWS CLI

Modificar uma solicitação de frota spot

Este comando de exemplo atualiza a capacidade de destino da solicitação de frota Spot especificada.

Comando:

```
aws ec2 modify-spot-fleet-request --target-capacity 20 --spot-fleet-request-
id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Saída:

```
{
    "Return": true
}
```

Este exemplo de comando diminui a capacidade de destino da frota Spot especificada sem encerrar qualquer instância Spot como resultado.

Comando:

```
aws ec2 modify-spot-fleet-request --target-capacity 10 --excess-capacity-
termination-policy NoTermination --spot-fleet-request-id sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Saída:

```
{
    "Return": true
```

```
}
```

- Para obter detalhes da API, consulte ModifySpotFleetRequestem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza a capacidade de destino da solicitação de frota spot especificada.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Saída:

```
True
```

- Para obter detalhes da API, consulte ModifySpotFleetRequestem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza a capacidade de destino da solicitação de frota spot especificada.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Saída:

```
True
```

- Para obter detalhes da API, consulte ModifySpotFleetRequestem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifySubnetAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifySubnetAttribute`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar uma rede privada virtual básica

CLI

AWS CLI

Alterar o comportamento de endereçamento IPv4 público de uma sub-rede

Este exemplo modifica a subnet-1a2b3c4d para especificar que todas as instâncias executadas nessa sub-rede recebam um endereço IPv4 público. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --map-public-ip-on-launch
```

Alterar o comportamento de endereçamento IPv6 de uma sub-rede

Este exemplo modifica a subnet-1a2b3c4d para especificar que todas as instâncias executadas nessa sub-rede recebam um endereço IPv6 do intervalo da sub-rede.

Comando:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --assign-ipv6-address-on-creation
```

Para obter mais informações, consulte Endereço IP na VPC no Guia do usuário da Nuvem privada virtual AWS .

- Para obter detalhes da API, consulte ModifySubnetAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo habilita o endereçamento IP público para a sub-rede especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Exemplo 2: esse exemplo desabilita o endereçamento IP público para a sub-rede especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Para obter detalhes da API, consulte ModifySubnetAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o endereçamento IP público para a sub-rede especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Exemplo 2: esse exemplo desabilita o endereçamento IP público para a sub-rede especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Para obter detalhes da API, consulte ModifySubnetAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyVolumeAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyVolumeAttribute`.

CLI

AWS CLI

Modificar um atributo de volume

Este exemplo define o atributo `autoEnableIo` do volume com o ID `vol-1234567890abcdef0` como `true`. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 modify-volume-attribute --volume-id vol-1234567890abcdef0 --auto-enable-
io
```

- Para obter detalhes da API, consulte ModifyVolumeAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo modifica o atributo especificado do volume especificado. I/O as operações do volume são retomadas automaticamente após serem suspensas devido a dados potencialmente inconsistentes.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Para obter detalhes da API, consulte ModifyVolumeAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo modifica o atributo especificado do volume especificado. I/O as operações do volume são retomadas automaticamente após serem suspensas devido a dados potencialmente inconsistentes.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Para obter detalhes da API, consulte ModifyVolumeAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyVpcAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyVpcAttribute`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma rede privada virtual básica
- Introdução aos bancos de dados de grafos

CLI

AWS CLI

Para modificar o DnsSupport atributo enable

Este exemplo modifica o atributo `enableDnsSupport`. Esse atributo indica se a VPC conta com resolução de DNS habilitada. Se este atributo é `true`, o servidor de DNS da Amazon resolve os nomes de hosts DNS de suas instâncias para os endereços IP correspondentes; caso contrário, ele não resolve. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-support "{\"Value
\":false}"
```

Para modificar o DnsHostnames atributo enable

Este exemplo modifica o atributo `enableDnsHostnames`. Esse atributo indica se instâncias executadas na VPC obtêm nomes de host DNS. Se esse atributo é `true`, as instâncias na VPC obtêm os nomes de hosts DNS; caso contrário, isso não ocorrerá. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-hostnames
 "{\"Value\":false}"
```

- Para obter detalhes da API, consulte ModifyVpcAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo habilita o suporte a nomes de host DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Exemplo 2: esse exemplo desabilita o suporte a nomes de host DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Exemplo 3: esse exemplo habilita o suporte à resolução DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Exemplo 4: esse exemplo desabilita o suporte à resolução DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Para obter detalhes da API, consulte ModifyVpcAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o suporte a nomes de host DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Exemplo 2: esse exemplo desabilita o suporte a nomes de host DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Exemplo 3: esse exemplo habilita o suporte à resolução DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Exemplo 4: esse exemplo desabilita o suporte à resolução DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Para obter detalhes da API, consulte ModifyVpcAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`MonitorInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `MonitorInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Enable detailed monitoring for an Amazon Elastic Compute Cloud (Amazon EC2)
 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::enableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
 ec2Client.MonitorInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to enable monitoring on instance. A dry run
 should trigger an error."
                <<
                std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
                  instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                  std::endl;
        return false;
```

```
    }

    request.SetDryRun(false);
    Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
 ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
                  instanceId << ": " <<
                  monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully enabled monitoring on instance " <<
                  instanceId << std::endl;
    }

    return monitorInstancesOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte MonitorInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para habilitar o monitoramento detalhado de uma instância

Este exemplo de comando habilita o monitoramento detalhado da instância especificada.

Comando:

```
aws ec2 monitor-instances --instance-ids i-1234567890abcdef0
```

Saída:

```
{
  "InstanceMonitorings": [
      {
          "InstanceId": "i-1234567890abcdef0",
          "Monitoring": {
              "State": "pending"
          }
      }
```

```
  ]
}
```

- Para obter detalhes da API, consulte MonitorInstancesem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, MonitorInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Turn on detailed monitoring for the selected instance.
 * By default, metrics are sent to Amazon CloudWatch every 5 minutes.
 * For a cost you can enable detailed monitoring which sends metrics every
 minute.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new MonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instancesBeingMonitored = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
 ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instancesBeingMonitored.join("\n"));
  } catch (caught) {
```

```
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte MonitorInstancesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo de comando habilita o monitoramento detalhado da instância especificada.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Saída:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Para obter detalhes da API, consulte MonitorInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo de comando habilita o monitoramento detalhado da instância especificada.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Saída:

```
InstanceId    Monitoring
----------    ----------
```

```
i-12345678    Amazon.EC2.Model.Monitoring
```

- Para obter detalhes da API, consulte MonitorInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to
 monitor the instance without actually making the request. "
        lo_ec2->monitorinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to monitor this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to enable detailed monitoring completed.' TYPE 'I'.
          " DryRun is set to false to enable detailed monitoring. "
          lo_ec2->monitorinstances(
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Detailed monitoring enabled.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to monitor this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
```

```
          MESSAGE 'Dry run to enable detailed monitoring failed. User does not
 have the permissions to monitor the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a MonitorInstancesreferência da API AWS SDK for SAP ABAP.

Usar **`MoveAddressToVpc`** com uma CLI

Os exemplos de código a seguir mostram como usar o `MoveAddressToVpc`.

CLI

AWS CLI

Para mover um endereço para EC2-VPC

Este exemplo move o endereço IP elástico 54.123.4.56 para a plataforma. EC2-VPC

Comando:

```
aws ec2 move-address-to-vpc --public-ip 54.123.4.56
```

Saída:

```
{
  "Status": "MoveInProgress"
}
```

- Para obter detalhes da API, consulte MoveAddressToVpcem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo move uma instância do EC2 com um endereço IP público de 12.345.67.89 para a EC2-VPC plataforma na região Leste dos EUA (Norte da Virgínia).

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Exemplo 2: Este exemplo canaliza os resultados de um Get-EC2Instance comando para o Move-EC2AddressToVpc cmdlet. O Get-EC2Instance comando obtém uma instância especificada pelo ID da instância e retorna a propriedade de endereço IP público da instância.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- Para obter detalhes da API, consulte MoveAddressToVpcem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo move uma instância do EC2 com um endereço IP público de 12.345.67.89 para a EC2-VPC plataforma na região Leste dos EUA (Norte da Virgínia).

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Exemplo 2: Este exemplo canaliza os resultados de um Get-EC2Instance comando para o Move-EC2AddressToVpc cmdlet. O Get-EC2Instance comando obtém uma instância especificada pelo ID da instância e retorna a propriedade de endereço IP público da instância.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- Para obter detalhes da API, consulte MoveAddressToVpcem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PurchaseHostReservation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PurchaseHostReservation`.

CLI

AWS CLI

Comprar uma reserva de host dedicado

Este exemplo compra a oferta de Reserva de Host Dedicado especificada para o Host Dedicado especificado em sua conta.

Comando:

```
aws ec2 purchase-host-reservation --offering-id hro-03f707bf363b6b324 --host-id-set h-013abcd2a00cbd123
```

Saída:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
      {
          "HourlyPrice": "1.499",
          "InstanceFamily": "m4",
          "PaymentOption": "NoUpfront",
          "HostIdSet": [
              "h-013abcd2a00cbd123"
          ],
          "HostReservationId": "hr-0d418a3a4ffc669ae",
          "UpfrontPrice": "0.000",
          "Duration": 31536000
      }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- Para obter detalhes da API, consulte PurchaseHostReservationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo compra a oferta de reserva hro-0c1f23456789d0ab com configurações que correspondem às do seu host dedicado h-01e23f4cd567890f1

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

Saída:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- Para obter detalhes da API, consulte PurchaseHostReservationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo compra a oferta de reserva hro-0c1f23456789d0ab com configurações que correspondem às do seu host dedicado h-01e23f4cd567890f1

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

Saída:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- Para obter detalhes da API, consulte PurchaseHostReservationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PurchaseScheduledInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PurchaseScheduledInstances`.

CLI

AWS CLI

Comprar uma instância programada

Este exemplo compra uma Instância Programada.

Comando:

```
aws ec2 purchase-scheduled-instances --purchase-requests file://purchase-
request.json
```

Purchase-request.json:

```
[
    {
        "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
        "InstanceCount": 1
    }
]
```

Saída:

```
{
  "ScheduledInstanceSet": [
      {
          "AvailabilityZone": "us-west-2b",
          "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
          "HourlyPrice": "0.095",
          "CreateDate": "2016-01-25T21:43:38.612Z",
          "Recurrence": {
              "OccurrenceDaySet": [
                  1
              ],
              "Interval": 1,
              "Frequency": "Weekly",
              "OccurrenceRelativeToEnd": false,
```

```
                "OccurrenceUnit": ""
            },
            "Platform": "Linux/UNIX",
            "TermEndDate": "2017-01-31T09:00:00Z",
            "InstanceCount": 1,
            "SlotDurationInHours": 32,
            "TermStartDate": "2016-01-31T09:00:00Z",
            "NetworkPlatform": "EC2-VPC",
            "TotalScheduledInstanceHours": 1696,
            "NextSlotStartTime": "2016-01-31T09:00:00Z",
            "InstanceType": "c4.large"
        }
    ]
}
```

- Para obter detalhes da API, consulte PurchaseScheduledInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo: esse exemplo compra uma Instância Programada.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Saída:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
```

```
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

- Para obter detalhes da API, consulte PurchaseScheduledInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo: esse exemplo compra uma Instância Programada.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Saída:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

- Para obter detalhes da API, consulte PurchaseScheduledInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`RebootInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RebootInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Reinicialize uma instância pelo ID.

```
    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)
    {
        try
        {
            var request = new RebootInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.RebootInstancesAsync(request);

            // Wait for the instance to be running.
            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
```

```
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
{ec2Exception.Message}");
            }
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while rebooting the instance
{ec2InstanceId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);
```

```
        return hasState;
    }
```

Substitua o perfil de uma instância, reinicialize e reinicie um servidor Web.

```
    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
```

```
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
```

```
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte RebootInstancesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::rebootInstance(const Aws::String &instanceId,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);
```

```
    Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
 ec2Client.RebootInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to reboot on instance. A dry run should
 trigger an error."
                <<
                std::endl;
        return false;
    } else if (dry_run_outcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to reboot instance " << instanceId << ": "
                  << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::RebootInstancesOutcome outcome =
 ec2Client.RebootInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to reboot instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully rebooted instance " << instanceId <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte RebootInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para reinicializar uma instância do Amazon EC2

Este exemplo reinicia a instância especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 reboot-instances --instance-ids i-1234567890abcdef5
```

Para obter mais informações, consulte Reinicializar a instância no Guia do usuário do Amazon Elastic Compute Cloud.

- Para obter detalhes da API, consulte RebootInstancesem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, RebootInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Requests a reboot of the specified instances. This operation is asynchronous;
 * it only queues a request to reboot the specified instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new RebootInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    await client.send(command);
    console.log("Instance rebooted successfully.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
```

```
      console.warn(
        `${caught.message}. Please provide the InstanceId of a valid instance to
 reboot.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte RebootInstancesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo reinicia a instância especificada.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Para obter detalhes da API, consulte RebootInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo reinicia a instância especificada.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Para obter detalhes da API, consulte RebootInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```

```
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
```

```
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a RebootInstancesReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.reboot_instance(self.instance_id()).await?;
            ec2.wait_for_instance_stopped(self.instance_id(), None)
                .await?;
            ec2.wait_for_instance_ready(self.instance_id(), None)
                .await?;
        }
        Ok(())
    }
```

```
    pub async fn reboot_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Rebooting instance {instance_id}");

        self.client
            .reboot_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        Ok(())
    }
```

Os waiters, por exemplo, devem estar nos estados interrompido e pronto, usando a API Waiters. O uso da API Waiters requer “use aws_sdk_ec2::client::Waiters” no arquivo rust.

```
/// Wait for an instance to be ready and status ok (default wait 60 seconds)
pub async fn wait_for_instance_ready(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_status_ok()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to start.",
                exceeded.max_wait().as_secs()
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}

pub async fn wait_for_instance_stopped(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_stopped()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to stop.",
                exceeded.max_wait().as_secs(),
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}
```

- Para obter detalhes da API, consulte a RebootInstancesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to
 reboot the instance without actually making the request. "
        lo_ec2->rebootinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to reboot this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to reboot instance completed.' TYPE 'I'.
          " DryRun is set to false to make a reboot request. "
          lo_ec2->rebootinstances(
             it_instanceids = lt_instance_ids
             iv_dryrun = abap_false ).
          MESSAGE 'Instance rebooted.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to reboot this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
```

```
          MESSAGE 'Dry run to reboot instance failed. User does not have
 permissions to reboot the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RebootInstancesreferência da API AWS SDK for SAP ABAP.

Usar **`RegisterImage`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterImage`.

CLI

AWS CLI

Exemplo 1: registrar uma AMI usando um arquivo de manifesto

O exemplo `register-image` a seguir registra uma AMI usando o arquivo de manifesto especificado no Amazon S3.

```
aws ec2 register-image \
    --name my-image \
    --image-location amzn-s3-demo-bucket/myimage/image.manifest.xml
```

Saída:

```
{
    "ImageId": "ami-1234567890EXAMPLE"
}
```

Para obter mais informações, consulte Imagens de máquina da Amazon (AMIs) no Guia do usuário do Amazon EC2.

Exemplo 2: registrar uma AMI usando um snapshot de um dispositivo raiz

O exemplo `register-image` a seguir registra uma AMI usando o snapshot especificado de um volume raiz do EBS como um dispositivo `/dev/xvda`. O mapeamento de dispositivos de blocos também inclui um volume EBS vazio de 100 GiB como dispositivo `/dev/xvdf`.

```
aws ec2 register-image \
    --name my-image \
    --root-device-name /dev/xvda \
    --block-device-mappings DeviceName=/dev/
xvda,Ebs={SnapshotId=snap-0db2cf683925d191f} DeviceName=/dev/
xvdf,Ebs={VolumeSize=100}
```

Saída:

```
{
    "ImageId": "ami-1a2b3c4d5eEXAMPLE"
}
```

Para obter mais informações, consulte Imagens de máquina da Amazon (AMIs) no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte RegisterImageem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra uma AMI usando o arquivo de manifesto especificado no Amazon S3.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- Para obter detalhes da API, consulte RegisterImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma AMI usando o arquivo de manifesto especificado no Amazon S3.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- Para obter detalhes da API, consulte RegisterImageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RejectVpcPeeringConnection`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RejectVpcPeeringConnection`.

CLI

AWS CLI

Rejeitar uma conexão de emparelhamento da VPC

Este exemplo rejeita a solicitação de conexão de emparelhamento da VPC especificada.

Comando:

```
aws ec2 reject-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

Saída:

```
{
    "Return": true
}
```

- Para obter detalhes da API, consulte RejectVpcPeeringConnectionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: O exemplo acima nega a solicitação de ID de solicitação VpcPeering pcx-01a2b3ce45fe67eb8

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Para obter detalhes da API, consulte RejectVpcPeeringConnectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: O exemplo acima nega a solicitação de ID de solicitação VpcPeering pcx-01a2b3ce45fe67eb8

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Para obter detalhes da API, consulte RejectVpcPeeringConnectionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ReleaseAddress`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ReleaseAddress`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma rede privada virtual básica
- Introdução às máquinas virtuais
- Rede privada virtual com servidores privados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Release an Elastic IP address. After the Elastic IP address is released,
    /// it can no longer be used.
```

```
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
            var request = new ReleaseAddressRequest { AllocationId =
 allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
 {ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while releasing the AllocationId
 {allocationId}.: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte ReleaseAddressa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_release_address
#
# This function releases an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to
 release.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_release_address() {
  local allocation_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_release_address"
    echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 release."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 release-address \
    --allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
```

```
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte ReleaseAddressem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Release an Elastic IP address.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::releaseAddress(const Aws::String &allocationID,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2(clientConfiguration);

    Aws::EC2::Model::ReleaseAddressRequest request;
    request.SetAllocationId(allocationID);

    Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to release Elastic IP address " <<
                  allocationID << ":" << outcome.GetError().GetMessage() <<
                  std::endl;
    } else {
        std::cout << "Successfully released Elastic IP address " <<
                  allocationID << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ReleaseAddressa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para liberar um endereço IP elástico para EC2-Classic

Este exemplo libera um endereço IP elástico para uso com instâncias em EC2-Classic. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 release-address --public-ip 198.51.100.0
```

Para liberar um endereço IP elástico para EC2-VPC

Este exemplo libera um endereço IP elástico para usar com instâncias em uma VPC. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 release-address --allocation-id eipalloc-64d5890a
```

- Para obter detalhes da API, consulte ReleaseAddressem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Releases an Elastic IP address asynchronously.
     *
     * @param allocId the allocation ID of the Elastic IP address to be released
```

```
     * @return a {@link CompletableFuture} representing the asynchronous
 operation of releasing the Elastic IP address
     */
    public CompletableFuture<ReleaseAddressResponse>
 releaseEC2AddressAsync(String allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
 getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP
 address", ex);
            }
        });

        return response;
    }
```

- Para obter detalhes da API, consulte ReleaseAddressa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ReleaseAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Release an Elastic IP address.
 * @param {{ allocationId: string }} options
 */
```

```
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new ReleaseAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AllocationId: allocationId,
  });

  try {
    await client.send(command);
    console.log("Successfully released address.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationID.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte ReleaseAddressa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
```

```
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- Para obter detalhes da API, consulte a ReleaseAddressreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo libera o endereço IP elástico especificado para instâncias em uma VPC.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Exemplo 2: Este exemplo libera o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Para obter detalhes da API, consulte ReleaseAddressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo libera o endereço IP elástico especificado para instâncias em uma VPC.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Exemplo 2: Este exemplo libera o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Para obter detalhes da API, consulte ReleaseAddressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
```

```
                            access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip)  # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address
"
```

```
                    "and ensure it is allocated to your account in the correct
 region "
                    "before attempting to release it."
                )
            raise
```

- Para obter detalhes da API, consulte a ReleaseAddressReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  true
```

```
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  false
end
```

- Para obter detalhes da API, consulte ReleaseAddressa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
       self.client
           .release_address()
           .allocation_id(allocation_id)
           .send()
           .await?;
       Ok(())
   }
```

- Para obter detalhes da API, consulte a ReleaseAddressreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).
        MESSAGE 'Elastic IP address released.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ReleaseAddressreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Release an allocated Elastic IP.
    ///
    /// - Parameter allocationId: The allocation ID of the Elastic IP to
    ///   release.
    func releaseAddress(allocationId: String?) async {
```

```
        do {
            _ = try await ec2Client.releaseAddress(
                input: ReleaseAddressInput(
                    allocationId: allocationId
                )
            )
        } catch {
            print("*** Unable to release the IP address:
\(error.localizedDescription)")
        }
    }
```

- Para obter detalhes da API, consulte ReleaseAddressa referência da API AWS SDK for Swift.

Usar **`ReleaseHosts`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ReleaseHosts`.

CLI

AWS CLI

Liberar um host dedicado da sua conta

Liberar um host dedicado da sua conta. As instâncias que estão no host devem ser interrompidas ou encerradas antes que o host possa ser liberado.

Comando:

```
aws ec2 release-hosts --host-id=h-0029d6e3cacf1b3da
```

Saída:

```
{
    "Successful":  [
        "h-0029d6e3cacf1b3da"
         ],
  "Unsuccessful": []

 }
```

- Para obter detalhes da API, consulte ReleaseHostsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo libera o ID de host h-0badafd1dcb2f3456 fornecido

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
 "h-0badafd1dcb2f3456".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Successful            Unsuccessful
----------            ------------
{h-0badafd1dcb2f3456} {}
```

- Para obter detalhes da API, consulte ReleaseHostsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo libera o ID de host h-0badafd1dcb2f3456 fornecido

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
 "h-0badafd1dcb2f3456".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

```
Successful            Unsuccessful
----------            ------------
{h-0badafd1dcb2f3456} {}
```

- Para obter detalhes da API, consulte ReleaseHostsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ReplaceIamInstanceProfileAssociation`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ReplaceIamInstanceProfileAssociation`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e gerenciar um serviço resiliente

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
   /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
   /// used to restart the Python web server.
   /// </summary>
   /// <param name="instanceId">The Id of the instance to update.</param>
   /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
   /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
   /// <returns>Async task.</returns>
```

```
   public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
   {
       try
       {
           await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
               new ReplaceIamInstanceProfileAssociationRequest()
               {
                   AssociationId = associationId,
                   IamInstanceProfile = new IamInstanceProfileSpecification()
                   {
                       Name = credsProfileName
                   }
               });
           // Allow time before resetting.
           Thread.Sleep(25000);

           await _amazonEc2.RebootInstancesAsync(
               new RebootInstancesRequest(new List<string>() { instanceId }));
           Thread.Sleep(25000);
           var instanceReady = false;
           var retries = 5;
           while (retries-- > 0 && !instanceReady)
           {
               var instancesPaginator =
                   _amazonSsm.Paginators.DescribeInstanceInformation(
                       new DescribeInstanceInformationRequest());
               // Get the entire list using the paginator.
               await foreach (var instance in
instancesPaginator.InstanceInformationList)
               {
                   instanceReady = instance.InstanceId == instanceId;
                   if (instanceReady)
                   {
                       break;
                   }
               }
           }
           Console.WriteLine("Waiting for instance to be running.");
           await WaitForInstanceState(instanceId, InstanceStateName.Running);
           Console.WriteLine("Instance ready.");
           Console.WriteLine($"Sending restart command to instance
{instanceId}");
           await _amazonSsm.SendCommandAsync(
```

```
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ReplaceIamInstanceProfileAssociationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para substituir um perfil de instância do IAM de uma instância

Este exemplo substitui o perfil de instância do IAM representado pela associação `iip-assoc-060bae234aac2e7fa` pelo perfil de instância do IAM chamado `AdminRole`.

```
aws ec2 replace-iam-instance-profile-association \
    --iam-instance-profile Name=AdminRole \
    --association-id iip-assoc-060bae234aac2e7fa
```

Saída:

```
{
    "IamInstanceProfileAssociation": {
        "InstanceId": "i-087711ddaf98f9489",
        "State": "associating",
        "AssociationId": "iip-assoc-0b215292fab192820",
        "IamInstanceProfile": {
            "Id": "AIPAJLNLDX3AMYZNWYYAY",
            "Arn": "arn:aws:iam::123456789012:instance-profile/AdminRole"
        }
    }
}
```

- Para obter detalhes da API, consulte ReplaceIamInstanceProfileAssociationem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
```

```
        ),
      );
```

- Para obter detalhes da API, consulte ReplaceIamInstanceProfileAssociationa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo substitui o perfil de instância de uma instância em execução, reinicia a instância e envia um comando para a instância após ela iniciar.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
```

```
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.
```

```
        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
```

```
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
 is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a ReplaceIamInstanceProfileAssociationReferência da API AWS SDK for Python (Boto3).

Usar **`ReplaceNetworkAclAssociation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ReplaceNetworkAclAssociation`.

CLI

AWS CLI

Substituir a ACL de rede associada a uma sub-rede

Este exemplo associa a ACL de rede especificada à sub-rede da associação de ACL de rede especificada.

Comando:

```
aws ec2 replace-network-acl-association --association-id aclassoc-e5b95c8c --
network-acl-id acl-5fb85d36
```

Saída:

```
{
    "NewAssociationId": "aclassoc-3999875b"
}
```

- Para obter detalhes da API, consulte ReplaceNetworkAclAssociationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo associa a ACL de rede especificada à sub-rede da associação de ACL de rede especificada.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Saída:

```
aclassoc-87654321
```

- Para obter detalhes da API, consulte ReplaceNetworkAclAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa a ACL de rede especificada à sub-rede da associação de ACL de rede especificada.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Saída:

```
aclassoc-87654321
```

- Para obter detalhes da API, consulte ReplaceNetworkAclAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ReplaceNetworkAclEntry`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ReplaceNetworkAclEntry`.

CLI

AWS CLI

Substituir uma entrada de ACL de rede

Este exemplo substitui uma entrada para a rede ACL especificada. A nova regra 100 permite tráfego de entrada a partir de 203.0.113. 12/24 na porta UDP 53 (DNS) em qualquer sub-rede associada.

Comando:

```
aws ec2 replace-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 100 --protocol udp --port-range From=53,To=53 --cidr-block 203.0.113.12/24 --rule-action allow
```

- Para obter detalhes da API, consulte ReplaceNetworkAclEntryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo substitui a entrada especificada da ACL de rede especificada. A nova regra permite tráfego de entrada do endereço especificado para qualquer sub-rede associada.

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -RuleAction allow
```

- Para obter detalhes da API, consulte ReplaceNetworkAclEntryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui a entrada especificada da ACL de rede especificada. A nova regra permite tráfego de entrada do endereço especificado para qualquer sub-rede associada.

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -RuleAction allow
```

- Para obter detalhes da API, consulte ReplaceNetworkAclEntryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **ReplaceRoute** com uma CLI

Os exemplos de código a seguir mostram como usar o `ReplaceRoute`.

CLI

AWS CLI

Substituir uma rota

Este exemplo substitui a rota especificada na tabela de rotas especificada. A nova rota corresponde ao CIDR especificado e envia o tráfego para o gateway privado virtual especificado. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 replace-route --route-table-id rtb-22574640 --destination-cidr-
block 10.0.0.0/16 --gateway-id vgw-9a4cacf3
```

- Para obter detalhes da API, consulte ReplaceRouteem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo substitui a rota especificada para a tabela de rotas especificada. A nova rota envia o tráfego especificado ao gateway privado virtual especificado.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -
GatewayId vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte ReplaceRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui a rota especificada para a tabela de rotas especificada. A nova rota envia o tráfego especificado ao gateway privado virtual especificado.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -
GatewayId vgw-1a2b3c4d
```

- Para obter detalhes da API, consulte ReplaceRouteem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ReplaceRouteTableAssociation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ReplaceRouteTableAssociation`.

CLI

AWS CLI

Substituir uma tabela de rotas associada a uma sub-rede

Este exemplo associa a tabela de rotas especificada à sub-rede para a associação da tabela de rotas especificada.

Comando:

```
aws ec2 replace-route-table-association --association-id rtbassoc-781d0d1a --
route-table-id rtb-22574640
```

Saída:

```
{
    "NewAssociationId": "rtbassoc-3a1f0f58"
}
```

- Para obter detalhes da API, consulte ReplaceRouteTableAssociationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo associa a tabela de rotas especificada à sub-rede para a associação da tabela de rotas especificada.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

Saída:

```
rtbassoc-87654321
```

- Para obter detalhes da API, consulte ReplaceRouteTableAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa a tabela de rotas especificada à sub-rede para a associação da tabela de rotas especificada.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

Saída:

```
rtbassoc-87654321
```

- Para obter detalhes da API, consulte ReplaceRouteTableAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ReportInstanceStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ReportInstanceStatus`.

CLI

AWS CLI

Reportar feedback de status de uma instância

Este exemplo de comando reporta um feedback de status para a instância especificada.

Comando:

```
aws ec2 report-instance-status --instances i-1234567890abcdef0 --status impaired
 --reason-codes unresponsive
```

- Para obter detalhes da API, consulte ReportInstanceStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo informa o feedback de status para a instância especificada.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- Para obter detalhes da API, consulte ReportInstanceStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo informa o feedback de status para a instância especificada.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- Para obter detalhes da API, consulte ReportInstanceStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RequestSpotFleet`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RequestSpotFleet`.

CLI

AWS CLI

Solicitar uma frota Spot na sub-rede com o menor preço

Esse exemplo de comando cria uma solicitação de frota Spot com duas especificações de lançamento que diferem somente por sub-rede. A frota Spot lança as instâncias na sub-rede especificada de menor preço. Se as instâncias forem executadas em uma VPC padrão, elas receberão um endereço IP público por padrão. Se as instâncias forem executadas em uma VPC não padrão, elas não receberão um endereço IP público por padrão.

Observe que você não pode especificar sub-redes diferentes da mesma zona de disponibilidade em uma solicitação de frota spot.

Comando:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

Config.json:

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
      {
          "ImageId": "ami-1a2b3c4d",
          "KeyName": "my-key-pair",
          "SecurityGroups": [
              {
                  "GroupId": "sg-1a2b3c4d"
              }
          ],
          "InstanceType": "m3.medium",
          "SubnetId": "subnet-1a2b3c4d, subnet-3c4d5e6f",
          "IamInstanceProfile": {
              "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
          }
      }
  ]
}
```

Saída:

```
{
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

Solicitar uma frota Spot na zona de disponibilidade com o menor preço

Esse exemplo de comando cria uma solicitação de frota Spot com duas especificações de lançamento que diferem somente por Zona de Disponibilidade. A frota Spot lança as instâncias na Zona de Disponibilidade especificada de menor preço. Se sua conta suportar EC2-VPC apenas, o Amazon EC2 executa as instâncias spot na sub-rede padrão da zona de disponibilidade. Se sua conta oferecer suporte EC2-Classic, o Amazon EC2 executa as instâncias EC2-Classic na zona de disponibilidade.

Comando:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

Config.json:

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
      {
          "ImageId": "ami-1a2b3c4d",
          "KeyName": "my-key-pair",
          "SecurityGroups": [
              {
                  "GroupId": "sg-1a2b3c4d"
              }
          ],
          "InstanceType": "m3.medium",
          "Placement": {
              "AvailabilityZone": "us-west-2a, us-west-2b"
          },
          "IamInstanceProfile": {
              "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
          }
      }
  ]
}
```

Lançar instâncias spot em uma sub-rede e atribuir a elas endereços IP públicos

Este comando de exemplo atribui endereços IP públicos a instâncias lançadas em uma VPC não padrão. Observe que, ao especificar uma interface de rede, você deve incluir o ID da sub-rede e o ID do grupo de segurança usando a interface de rede.

Comando:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

Config.json:

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
      {
          "ImageId": "ami-1a2b3c4d",
          "KeyName": "my-key-pair",
          "InstanceType": "m3.medium",
          "NetworkInterfaces": [
              {
                  "DeviceIndex": 0,
                  "SubnetId": "subnet-1a2b3c4d",
                  "Groups": [ "sg-1a2b3c4d" ],
                  "AssociatePublicIpAddress": true
              }
          ],
          "IamInstanceProfile": {
              "Arn": "arn:aws:iam::880185128111:instance-profile/my-iam-role"
          }
      }
  ]
}
```

Solicitar uma frota Spot usando a estratégia de alocação diversificada

Esse exemplo de comando cria uma solicitação de frota Spot que executa 30 instâncias usando a estratégia de alocação diversificada. As especificações de lançamento variam de acordo com o tipo de instância. A frota spot distribui as instâncias pelas especificações de execução, de modo que haja 10 instâncias de cada tipo.

Comando:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

Config.json:

```
{
  "SpotPrice": "0.70",
  "TargetCapacity": 30,
  "AllocationStrategy": "diversified",
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
```

```
  "LaunchSpecifications": [
      {
          "ImageId": "ami-1a2b3c4d",
          "InstanceType": "c4.2xlarge",
          "SubnetId": "subnet-1a2b3c4d"
      },
      {
          "ImageId": "ami-1a2b3c4d",
          "InstanceType": "m3.2xlarge",
          "SubnetId": "subnet-1a2b3c4d"
      },
      {
          "ImageId": "ami-1a2b3c4d",
          "InstanceType": "r3.2xlarge",
          "SubnetId": "subnet-1a2b3c4d"
      }
  ]
}
```

Para obter mais informações, consulte Solicitações de frota Spot no Guia do usuário do Amazon Elastic Compute Cloud.

- Para obter detalhes da API, consulte RequestSpotFleetem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma solicitação de frota spot na zona de disponibilidade com o menor preço para o tipo de instância especificado. Se sua conta suportar EC2-VPC apenas, a frota spot executa as instâncias na zona de disponibilidade de menor preço que tem uma sub-rede padrão. Se sua conta suportar EC2-Classic, a frota spot executa as instâncias EC2-Classic na zona de disponibilidade de menor preço. Observe que o preço pago não excederá o preço spot especificado para a solicitação.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
```

```
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-
fleet-role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- Para obter detalhes da API, consulte RequestSpotFleetem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma solicitação de frota spot na zona de disponibilidade com o menor preço para o tipo de instância especificado. Se sua conta suportar EC2-VPC apenas, a frota spot executa as instâncias na zona de disponibilidade de menor preço que tem uma sub-rede padrão. Se sua conta suportar EC2-Classic, a frota spot executa as instâncias EC2-Classic na zona de disponibilidade de menor preço. Observe que o preço pago não excederá o preço spot especificado para a solicitação.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-
fleet-role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- Para obter detalhes da API, consulte RequestSpotFleetem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RequestSpotInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RequestSpotInstances`.

CLI

AWS CLI

Solicitar Instâncias spot

Este exemplo de comando cria uma solicitação única de Instância Spot para cinco instâncias na Zona de Disponibilidade especificada. Se sua conta suportar EC2-VPC apenas, o Amazon EC2 executa as instâncias na sub-rede padrão da zona de disponibilidade especificada. Se sua conta oferecer suporte EC2-Classic, o Amazon EC2 executa as instâncias EC2-Classic na zona de disponibilidade especificada.

Comando:

```
aws ec2 request-spot-instances --spot-price "0.03" --instance-count 5 --
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```
{
  "ImageId": "ami-1a2b3c4d",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "Placement": {
    "AvailabilityZone": "us-west-2a"
  },
  "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

Saída:

```
{
  "SpotInstanceRequests": [
      {
          "Status": {
              "UpdateTime": "2014-03-25T20:54:21.000Z",
              "Code": "pending-evaluation",
              "Message": "Your Spot request has been submitted for review, and is
 pending evaluation."
          },
          "ProductDescription": "Linux/UNIX",
          "SpotInstanceRequestId": "sir-df6f405d",
          "State": "open",
          "LaunchSpecification": {
              "Placement": {
```

```
                    "AvailabilityZone": "us-west-2a"
                },
                "ImageId": "ami-1a2b3c4d",
                "KeyName": "my-key-pair",
                "SecurityGroups": [
                    {
                        "GroupName": "my-security-group",
                        "GroupId": "sg-1a2b3c4d"
                    }
                ],
                "Monitoring": {
                    "Enabled": false
                },
                "IamInstanceProfile": {
                    "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
                },
                "InstanceType": "m3.medium"
            },
            "Type": "one-time",
            "CreateTime": "2014-03-25T20:54:20.000Z",
            "SpotPrice": "0.050000"
        },
        ...
    ]
}
```

Este exemplo de comando cria uma solicitação única de Instância Spot para cinco instâncias na sub-rede especificada. O Amazon EC2 executa as instâncias na sub-rede especificada. Se a VPC não for padrão, as instâncias não receberão um endereço IP público por padrão.

Comando:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 5 --
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```
{
  "ImageId": "ami-1a2b3c4d",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "SubnetId": "subnet-1a2b3c4d",
```

```
  "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

Saída:

```
{
  "SpotInstanceRequests": [
      {
          "Status": {
             "UpdateTime": "2014-03-25T22:21:58.000Z",
             "Code": "pending-evaluation",
             "Message": "Your Spot request has been submitted for review, and is
 pending evaluation."
          },
          "ProductDescription": "Linux/UNIX",
          "SpotInstanceRequestId": "sir-df6f405d",
          "State": "open",
          "LaunchSpecification": {
             "Placement": {
                 "AvailabilityZone": "us-west-2a"
             }
             "ImageId": "ami-1a2b3c4d"
             "SecurityGroups": [
                 {
                     "GroupName": "my-security-group",
                     "GroupID": "sg-1a2b3c4d"
                 }
             ]
             "SubnetId": "subnet-1a2b3c4d",
             "Monitoring": {
                 "Enabled": false
             },
             "IamInstanceProfile": {
                 "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
             },
             "InstanceType": "m3.medium",
          },
          "Type": "one-time",
          "CreateTime": "2014-03-25T22:21:58.000Z",
          "SpotPrice": "0.050000"
      },
```

```
      ...
  ]
}
```

Este exemplo atribui um endereço IP público às Instâncias Spot que você lança em uma VPC não padrão. Observe que, ao especificar uma interface de rede, você deve incluir o ID da sub-rede e o ID do grupo de segurança usando a interface de rede.

Comando:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 1 --
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```
{
  "ImageId": "ami-1a2b3c4d",
  "KeyName": "my-key-pair",
  "InstanceType": "m3.medium",
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "SubnetId": "subnet-1a2b3c4d",
      "Groups": [ "sg-1a2b3c4d" ],
      "AssociatePublicIpAddress": true
    }
  ],
  "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

- Para obter detalhes da API, consulte RequestSpotInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo solicita uma instância spot única na sub-rede especificada. Observe que o grupo de segurança deve ser criado para a VPC que contém a sub-rede especificada e

deve ser especificado por ID usando a interface de rede. Ao especificar uma interface de rede, você deve incluir o ID da sub-rede usando a interface de rede.

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
-LaunchSpecification_NetworkInterface $n
```

Saída:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 12/26/2015 7:44:10 AM
Fault                    :
InstanceId               :
LaunchedAvailabilityZone :
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.050000
State                    : open
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {}
Type                     : one-time
```

- Para obter detalhes da API, consulte RequestSpotInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo solicita uma instância spot única na sub-rede especificada. Observe que o grupo de segurança deve ser criado para a VPC que contém a sub-rede especificada e deve ser especificado por ID usando a interface de rede. Ao especificar uma interface de rede, você deve incluir o ID da sub-rede usando a interface de rede.

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
-LaunchSpecification_NetworkInterface $n
```

Saída:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 12/26/2015 7:44:10 AM
Fault                    :
InstanceId               :
LaunchedAvailabilityZone :
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.050000
State                    : open
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {}
Type                     : one-time
```

- Para obter detalhes da API, consulte RequestSpotInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ResetImageAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ResetImageAttribute`.

CLI

AWS CLI

Redefinir o atributo launchPermission

Este exemplo redefine o atributo `launchPermission` da AMI especificada para seu valor padrão. Por padrão, as AMIs são privadas. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 reset-image-attribute --image-id ami-5731123e --
attribute launchPermission
```

- Para obter detalhes da API, consulte ResetImageAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo redefine o atributo “launchPermission” para seu valor padrão. Por padrão, as AMIs são privadas.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Para obter detalhes da API, consulte ResetImageAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo redefine o atributo “launchPermission” para seu valor padrão. Por padrão, as AMIs são privadas.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Para obter detalhes da API, consulte ResetImageAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ResetInstanceAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ResetInstanceAttribute`.

CLI

AWS CLI

Para redefinir o DestCheck atributo de origem

Este exemplo redefine o atributo `sourceDestCheck` da instância especificada. A instância deve estar em uma VPC. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --
attribute sourceDestCheck
```

Redefinir o atributo kernel

Este exemplo redefine o atributo `kernel` da instância especificada. A instância deve estar no estado `stopped`. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --
attribute kernel
```

Redefinir o atributo ramdisk

Este exemplo redefine o atributo `ramdisk` da instância especificada. A instância deve estar no estado `stopped`. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --
attribute ramdisk
```

- Para obter detalhes da API, consulte ResetInstanceAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo redefine o atributo NetSupport 'sriov para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Exemplo 2: esse exemplo redefine o atributo “ebsOptimized” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Exemplo 3: Esse exemplo redefine o atributo “sourceDestCheck” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Exemplo 4: Esse exemplo redefine o atributo “disableApiTermination” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 disableApiTermination
```

Exemplo 5: Esse exemplo redefine o atributo “instanceInitiatedShutdownBehavior” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- Para obter detalhes da API, consulte ResetInstanceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo redefine o atributo NetSupport 'sriov para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Exemplo 2: esse exemplo redefine o atributo “ebsOptimized” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Exemplo 3: Esse exemplo redefine o atributo “sourceDestCheck” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Exemplo 4: Esse exemplo redefine o atributo “disableApiTermination” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 disableApiTermination
```

Exemplo 5: Esse exemplo redefine o atributo “instanceInitiatedShutdownBehavior” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- Para obter detalhes da API, consulte ResetInstanceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ResetNetworkInterfaceAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ResetNetworkInterfaceAttribute`.

CLI

AWS CLI

Redefinir um atributo da interface de rede

O `reset-network-interface-attribute` exemplo a seguir redefine o valor do atributo de source/destination verificação para`true`.

```
aws ec2 reset-network-interface-attribute \
    --network-interface-id eni-686ea200 \
    --source-dest-check sourceDestCheck
```

Este comando não produz saída.

- Para obter detalhes da API, consulte ResetNetworkInterfaceAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo redefine a source/destination verificação da interface de rede especificada.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck
```

- Para obter detalhes da API, consulte ResetNetworkInterfaceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo redefine a source/destination verificação da interface de rede especificada.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck
```

- Para obter detalhes da API, consulte ResetNetworkInterfaceAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ResetSnapshotAttribute`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ResetSnapshotAttribute`.

CLI

AWS CLI

Redefinir um atributo de snapshot

Este exemplo redefine as permissões de criação de volume para o snapshot `snap-1234567890abcdef0`. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 reset-snapshot-attribute --snapshot-id snap-1234567890abcdef0 --
attribute createVolumePermission
```

- Para obter detalhes da API, consulte ResetSnapshotAttributeem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo redefine o atributo especificado do snapshot especificado.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- Para obter detalhes da API, consulte ResetSnapshotAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo redefine o atributo especificado do snapshot especificado.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- Para obter detalhes da API, consulte ResetSnapshotAttributeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RevokeSecurityGroupEgress`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RevokeSecurityGroupEgress`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de streaming gerenciado

CLI

AWS CLI

Exemplo 1: remover a regra que permite tráfego de saída para um intervalo de endereços específico

O comando de exemplo `revoke-security-group-egress` a seguir remove a regra que concede acesso aos intervalos de endereços especificados na porta TCP 80.

```
aws ec2 revoke-security-group-egress \
    --group-id sg-026c12253ce15eff7 \
    --ip-
permissions [{IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=10.0.0.0/16}]
```

Este comando não produz saída.

Para obter mais informações, consulte Grupos de segurança no Manual do usuário da Amazon EC2.

Exemplo 2: remover a regra que permite tráfego de saída para um grupo de segurança específico

O comando de exemplo `revoke-security-group-egress` a seguir remove a regra que concede acesso ao grupo de segurança especificado na porta TCP 80.

```
aws ec2 revoke-security-group-egress \
    --group-id sg-026c12253ce15eff7 \
    --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort":
 443,"UserIdGroupPairs": [{"GroupId": "sg-06df23a01ff2df86d"}]}]'
```

Este comando não produz saída.

Para obter mais informações, consulte Grupos de segurança no Manual do usuário da Amazon EC2.

- Para obter detalhes da API, consulte RevokeSecurityGroupEgressem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo remove a regra do grupo de segurança especificado para EC2-VPC. Isso revoga o acesso ao intervalo de endereços IP especificado na porta TCP 80. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
```

```
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: esse exemplo revoga o acesso ao grupo de segurança de origem especificado na porta TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Para obter detalhes da API, consulte RevokeSecurityGroupEgressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove a regra do grupo de segurança especificado para EC2-VPC. Isso revoga o acesso ao intervalo de endereços IP especificado na porta TCP 80. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
```

```
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: esse exemplo revoga o acesso ao grupo de segurança de origem especificado na porta TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Para obter detalhes da API, consulte RevokeSecurityGroupEgressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RevokeSecurityGroupIngress`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RevokeSecurityGroupIngress`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Introdução aos bancos de dados de documentos
- Introdução ao armazenamento em cache em memória
- Conceitos básicos de streaming gerenciado

CLI

AWS CLI

Exemplo 1: remover uma regra de um grupo de segurança

O exemplo `revoke-security-group-ingress` a seguir remove o acesso à porta TCP 22 para o intervalo de endereços `203.0.113.0/24` do grupo de segurança especificado para uma VPC padrão.

```
aws ec2 revoke-security-group-ingress \
    --group-name mySecurityGroup
    --protocol tcp \
    --port 22 \
    --cidr 203.0.113.0/24
```

Se for bem-sucedido, esse comando não gerará nenhum resultado.

Para obter mais informações, consulte Grupos de segurança no Manual do usuário da Amazon EC2.

Exemplo 2: remover uma regra usando o conjunto de permissões de IP

O exemplo `revoke-security-group-ingress` a seguir usa o parâmetro `ip-permissions` para remover uma regra de entrada que permite a mensagem do ICMP `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` (Tipo 3, Código 4).

```
aws ec2 revoke-security-group-ingress \
    --group-id sg-026c12253ce15eff7 \
    --ip-
permissions IpProtocol=icmp,FromPort=3,ToPort=4,IpRanges=[{CidrIp=0.0.0.0/0}]
```

Se for bem-sucedido, esse comando não gerará nenhum resultado.

Para obter mais informações, consulte Grupos de segurança no Manual do usuário da Amazon EC2.

- Para obter detalhes da API, consulte RevokeSecurityGroupIngressem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo revoga o acesso à porta TCP 22 do intervalo de endereços especificado para o grupo de segurança especificado para. EC2-VPC Observe que você deve identificar grupos de segurança para EC2-VPC usar o ID do grupo de segurança, não o nome do grupo de segurança. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: Este exemplo revoga o acesso à porta TCP 22 do intervalo de endereços especificado para o grupo de segurança especificado para. EC2-Classic A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- Para obter detalhes da API, consulte RevokeSecurityGroupIngressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo revoga o acesso à porta TCP 22 do intervalo de endereços especificado para o grupo de segurança especificado para. EC2-VPC Observe que você deve identificar grupos de segurança para EC2-VPC usar o ID do grupo de segurança, não o nome do grupo de segurança. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }
```

```
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: Este exemplo revoga o acesso à porta TCP 22 do intervalo de endereços especificado para o grupo de segurança especificado para. EC2-Classic A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- Para obter detalhes da API, consulte RevokeSecurityGroupIngressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`RunInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RunInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma rede privada virtual básica
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criação e gerenciamento de volumes de armazenamento em bloco
- Conceitos básicos de compras do mercado de software
- Conceitos básicos de streaming gerenciado
- Introdução às máquinas virtuais
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
    /// <param name="groupId">The Id of the Amazon EC2 security group that will
be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
    {
```

```
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
                MinCount = 1,
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
            var instanceId = response.Reservation.Instances[0].InstanceId;

            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte RunInstancesa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#       -t instance_type - The instance type to use (e.g., t2.micro).
#       -k key_pair_name - The name of the key pair to use.
#       -s security_group_id - The ID of the security group to use.
#       -c count - The number of instances to launch (default: 1).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_run_instances() {
  local image_id instance_type key_pair_name security_group_id count response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_run_instances"
    echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
    echo "  -k key_pair_name - The name of the key pair to use."
    echo "  -s security_group_id - The ID of the security group to use."
    echo "  -c count - The number of instances to launch (default: 1)."
```

```
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:t:k:s:c:h" option; do
    case "${option}" in
      i) image_id="${OPTARG}" ;;
      t) instance_type="${OPTARG}" ;;
      k) key_pair_name="${OPTARG}" ;;
      s) security_group_id="${OPTARG}" ;;
      c) count="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$image_id" ]]; then
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
 parameter."
    usage
    return 1
  fi

  if [[ -z "$instance_type" ]]; then
    errecho "ERROR: You must provide an instance type with the -t parameter."
    usage
    return 1
  fi

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -k parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -s parameter."
    usage
    return 1
  fi

  if [[ -z "$count" ]]; then
    count=1
  fi

  response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
```

```
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte RunInstancesem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceName: A name for the EC2 instance.
  \param amiId: An Amazon Machine Image (AMI) identifier.
  \param[out] instanceID: String to return the instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::runInstance(const Aws::String &instanceName,
                              const Aws::String &amiId,
                              Aws::String &instanceID,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RunInstancesRequest runRequest;
    runRequest.SetImageId(amiId);
    runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    runRequest.SetMinCount(1);
    runRequest.SetMaxCount(1);

    Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
            runRequest);
    if (!runOutcome.IsSuccess()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
                  " based on ami " << amiId << ":" <<
                  runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const Aws::Vector<Aws::EC2::Model::Instance> &instances =
 runOutcome.GetResult().GetInstances();
    if (instances.empty()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
                  " based on ami " << amiId << ":" <<
                  runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    instanceID = instances[0].GetInstanceId();

    return true;
}
```

- Para obter detalhes da API, consulte RunInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: executar uma instância em uma sub-rede padrão

O exemplo `run-instances` a seguir executa uma única instância do tipo `t2.micro` na sub-rede padrão da região atual e a associa à sub-rede padrão da VPC padrão da região. O par de chaves é opcional se você não planeja se conectar à instância usando SSH (Linux) ou RDP (Windows).

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --key-name MyKeyPair
```

Saída:

```
{
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0abcdef1234567890",
            "InstanceId": "i-1231231230abcdef0",
            "InstanceType": "t2.micro",
            "KeyName": "MyKeyPair",
            "LaunchTime": "2018-05-10T08:05:20.000Z",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-2a",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
            "PrivateIpAddress": "10.0.0.157",
```

```
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-04a636d18e83cfacb",
            "VpcId": "vpc-1234567890abcdef0",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "",
            "EbsOptimized": false,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2018-05-10T08:05:20.000Z",
                        "AttachmentId": "eni-attach-0e325c07e928a0405",
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "Status": "attaching"
                    },
                    "Description": "",
                    "Groups": [
                        {
                            "GroupName": "MySecurityGroup",
                            "GroupId": "sg-0598c7d356eba48d7"
                        }
                    ],
                    "Ipv6Addresses": [],
                    "MacAddress": "0a:ab:58:e0:67:e2",
                    "NetworkInterfaceId": "eni-0c0a29997760baee7",
                    "OwnerId": "123456789012",
                    "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
                    "PrivateIpAddress": "10.0.0.157",
                    "PrivateIpAddresses": [
                        {
                            "Primary": true,
                            "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
                            "PrivateIpAddress": "10.0.0.157"
                        }
                    ],
```

```
                    "SourceDestCheck": true,
                    "Status": "in-use",
                    "SubnetId": "subnet-04a636d18e83cfacb",
                    "VpcId": "vpc-1234567890abcdef0",
                    "InterfaceType": "interface"
                }
            ],
            "RootDeviceName": "/dev/xvda",
            "RootDeviceType": "ebs",
            "SecurityGroups": [
                {
                    "GroupName": "MySecurityGroup",
                    "GroupId": "sg-0598c7d356eba48d7"
                }
            ],
            "SourceDestCheck": true,
            "StateReason": {
                "Code": "pending",
                "Message": "pending"
            },
            "Tags": [],
            "VirtualizationType": "hvm",
            "CpuOptions": {
                "CoreCount": 1,
                "ThreadsPerCore": 1
            },
            "CapacityReservationSpecification": {
                "CapacityReservationPreference": "open"
            },
            "MetadataOptions": {
                "State": "pending",
                "HttpTokens": "optional",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "enabled"
            }
        }
    ],
    "OwnerId": "123456789012",
    "ReservationId": "r-02a3f596d91211712"
}
```

Exemplo 2: para executar uma instância em uma sub-rede não padrão e adicionar um endereço IP público

O exemplo `run-instances` a seguir solicita um endereço IP público para uma instância que você está executando em uma sub-rede não padrão. A instância está associada ao grupo de segurança especificado.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --subnet-id subnet-08fc749671b2d077c \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --associate-public-ip-address \
    --key-name MyKeyPair
```

Para obter um exemplo da saída de `run-instances`, consulte o Exemplo 1.

Exemplo 3: executar uma instância com volumes adicionais

O exemplo `run-instances` a seguir usa um mapeamento de dispositivos de blocos, especificado em mapping.json, para anexar volumes adicionais na execução. Um mapeamento de dispositivos de blocos pode especificar volumes do EBS, volumes de armazenamento de instância ou tanto volumes do EBS quanto volumes de armazenamento de instância.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --subnet-id subnet-08fc749671b2d077c \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --key-name MyKeyPair \
    --block-device-mappings file://mapping.json
```

Conteúdo de `mapping.json`. Este exemplo adiciona `/dev/sdh` como um volume vazio do EBS com um tamanho de 100 GiB.

```
[
    {
        "DeviceName": "/dev/sdh",
        "Ebs": {
            "VolumeSize": 100
        }
    }
]
```

Conteúdo de `mapping.json`. Este exemplo adiciona `ephemeral1` como um volume de armazenamento de instância.

```
[
    {
        "DeviceName": "/dev/sdc",
        "VirtualName": "ephemeral1"
    }
]
```

Para obter um exemplo da saída de `run-instances`, consulte o Exemplo 1.

Para obter mais informações sobre mapeamentos de dispositivos de blocos, consulte Mapeamento de dispositivos de blocos no Guia do usuário do Amazon EC2.

Exemplo 4: executar uma instância e adicionar tags na criação

O exemplo `run-instances` a seguir adiciona uma tag com uma chave de `webserver` e um valor de `production` à instância. O comando também aplica uma tag com uma chave de `cost-center` e um valor de `cc123` a qualquer volume do EBS criado (neste caso, o volume do dispositivo raiz).

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --count 1 \
    --subnet-id subnet-08fc749671b2d077c \
    --key-name MyKeyPair \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --tag-specifications
 'ResourceType=instance,Tags=[{Key=webserver,Value=production}]'
 'ResourceType=volume,Tags=[{Key=cost-center,Value=cc123}]'
```

Para obter um exemplo da saída de `run-instances`, consulte o Exemplo 1.

Exemplo 5: executar uma instância com dados do usuário

O exemplo `run-instances` a seguir passa dados do usuário em um arquivo chamado `my_script.txt` que contém um script de configuração para a sua instância. O script é executado na inicialização.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --count 1 \
    --subnet-id subnet-08fc749671b2d077c \
    --key-name MyKeyPair \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --user-data file://my_script.txt
```

Para obter um exemplo da saída de `run-instances`, consulte o Exemplo 1.

Para obter mais informações sobre dados do usuário da instância, consulte Trabalho com dados do usuário da instância no Guia do usuário do Amazon EC2.

Exemplo 6: executar uma instância de desempenho expansível

O exemplo `run-instances` a seguir executa uma instância t2.micro com a opção de crédito `unlimited`. Ao executar uma instância T2, se você não especificar a `--credit-specification`, o padrão é a opção de crédito `standard`. Ao executar uma instância T3, o padrão é a opção de crédito `unlimited`.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --count 1 \
    --subnet-id subnet-08fc749671b2d077c \
    --key-name MyKeyPair \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --credit-specification CpuCredits=unlimited
```

Para obter um exemplo da saída de `run-instances`, consulte o Exemplo 1.

Para obter mais informações sobre instâncias de desempenho expansível, consulte Instâncias de desempenho expansível no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte RunInstancesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2
instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the
started EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
       RunInstancesRequest runRequest = RunInstancesRequest.builder()
           .instanceType(instanceType)
           .keyName(keyName)
           .securityGroups(groupName)
           .maxCount(1)
           .minCount(1)
           .imageId(amiId)
           .build();

       CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
       return responseFuture.thenCompose(response -> {
           String instanceIdVal = response.instances().get(0).instanceId();
           System.out.println("Going to start an EC2 instance and use a waiter
to wait for it to be in running state");
           return getAsyncClient().waiter()
```

```
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
 throwable.getMessage(), throwable);
        });
    }
```

- Para obter detalhes da API, consulte RunInstancesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, RunInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Create new EC2 instances.
 * @param {{
 *  keyName: string,
 *  securityGroupIds: string[],
 *  imageId: string,
 *  instanceType: import('@aws-sdk/client-ec2')._InstanceType,
 *  minCount?: number,
 *  maxCount?: number }} options
 */
export const main = async ({
  keyName,
  securityGroupIds,
  imageId,
```

```
  instanceType,
  minCount = "1",
  maxCount = "1",
}) => {
  const client = new EC2Client({});
  minCount = Number.parseInt(minCount);
  maxCount = Number.parseInt(maxCount);
  const command = new RunInstancesCommand({
    // Your key pair name.
    KeyName: keyName,
    // Your security group.
    SecurityGroupIds: securityGroupIds,
    // An Amazon Machine Image (AMI). There are multiple ways to search for AMIs.
 For more information, see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html
    ImageId: imageId,
    // An instance type describing the resources provided to your instance. There
 are multiple
    // ways to search for instance types. For more information see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-
discovery.html
    InstanceType: instanceType,
    // Availability Zones have capacity limitations that may impact your ability
 to launch instances.
    // The `RunInstances` operation will only succeed if it can allocate at least
 the `MinCount` of instances.
    // However, EC2 will attempt to launch up to the `MaxCount` of instances,
 even if the full request cannot be satisfied.
    // If you need a specific number of instances, use `MinCount` and `MaxCount`
 set to the same value.
    // If you want to launch up to a certain number of instances, use `MaxCount`
 and let EC2 provision as many as possible.
    // If you require a minimum number of instances, but do not want to exceed a
 maximum, use both `MinCount` and `MaxCount`.
    MinCount: minCount,
    MaxCount: maxCount,
  });

  try {
    const { Instances } = await client.send(command);
    const instanceList = Instances.map(
      (instance) => `• ${instance.InstanceId}`,
    ).join("\n");
    console.log(`Launched instances:\n${instanceList}`);
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceCountExceeded") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte RunInstancesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
```

```
                value = name
            }

        val requestTags =
            CreateTagsRequest {
                resources = listOf(instanceId.toString())
                tags = listOf(tag)
            }
        ec2.createTags(requestTags)
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiId")
        return instanceId
    }
}
```

- Para obter detalhes da API, consulte a RunInstancesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo executa uma única instância da AMI especificada em uma VPC padrão EC2-Classic ou em uma VPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Exemplo 2: esse exemplo executa uma única instância da AMI especificada em uma VPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

Exemplo 3: para adicionar um volume do EBS ou um volume de armazenamento de instância, defina um mapeamento de dispositivos de blocos e adicione-o ao comando. Esse exemplo adiciona um volume de armazenamento de instância.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
```

```
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Exemplo 4: Para especificar uma das AMIs atuais do Windows, obtenha sua ID de AMI usando Get-EC2ImageByName. Esse exemplo executa uma instância da AMI base atual para Windows Server 2016.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

Exemplo 5: executa uma instância no ambiente de host dedicado especificado.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

Exemplo 6: essa solicitação executa duas instâncias e aplica uma tag com uma chave de servidor da web e um valor de produção às instâncias. A solicitação também aplica uma tag com uma chave de centro de custos e um valor de cc123 aos volumes criados (nesse caso, o volume do dispositivo raiz de cada instância).

```
$tag1 = @{ Key="webserver"; Value="production" }
$tag2 = @{ Key="cost-center"; Value="cc123" }

$tagspec1 = new-object Amazon.EC2.Model.TagSpecification
$tagspec1.ResourceType = "instance"
$tagspec1.Tags.Add($tag1)

$tagspec2 = new-object Amazon.EC2.Model.TagSpecification
$tagspec2.ResourceType = "volume"
$tagspec2.Tags.Add($tag2)

New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,
$tagspec2
```

- Para obter detalhes da API, consulte RunInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo executa uma única instância da AMI especificada em uma VPC padrão EC2-Classic ou em uma VPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Exemplo 2: esse exemplo executa uma única instância da AMI especificada em uma VPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

Exemplo 3: para adicionar um volume do EBS ou um volume de armazenamento de instância, defina um mapeamento de dispositivos de blocos e adicione-o ao comando. Esse exemplo adiciona um volume de armazenamento de instância.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Exemplo 4: Para especificar uma das AMIs atuais do Windows, obtenha sua ID de AMI usando Get-SSMLatestEC2Image. Esse exemplo executa uma instância da AMI base atual para Windows Server 2016.

```
$ami = (Get-SSMLatestEC2Image -Path 'ami-windows-latest' -ImageName
 'WINDOWS*2016*English*Core*BASE').Value

New-EC2Instance -ImageId $ami ...
```

Exemplo 5: executa uma instância no ambiente de host dedicado especificado.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

Exemplo 6: essa solicitação executa duas instâncias e aplica uma tag com uma chave de servidor da web e um valor de produção às instâncias. A solicitação também aplica uma tag com uma chave de centro de custos e um valor de cc123 aos volumes criados (nesse caso, o volume do dispositivo raiz de cada instância).

```
$tag1 = @{ Key="webserver"; Value="production" }
$tag2 = @{ Key="cost-center"; Value="cc123" }

$tagspec1 = new-object Amazon.EC2.Model.TagSpecification
$tagspec1.ResourceType = "instance"
$tagspec1.Tags.Add($tag1)

$tagspec2 = new-object Amazon.EC2.Model.TagSpecification
$tagspec2.ResourceType = "volume"
$tagspec2.Tags.Add($tag2)

New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,
$tagspec2
```

Exemplo 7: Este exemplo valida as permissões para iniciar uma instância do EC2 usando o DryRun parâmetro sem realmente criar a instância. Nota: se isso for feito com sucesso, será gerada uma exceção, que é o comportamento esperado.

```
New-EC2Instance -ImageId 'ami-12345678' -InstanceType 't2.micro' -KeyName 'my-
key-pair' -Region 'us-west-2' -DryRun $true
```

- Para obter detalhes da API, consulte RunInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.
```

```
        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
 instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
 the instance.
                                       If not specified, the default security group
 of the VPC is used.
        :return: A list of dictionaries representing Boto3 Instance objects
 representing the newly created instances.
        """
        try:
            instance_params = {
                "ImageId": image_id,
                "InstanceType": instance_type,
                "KeyName": key_pair_name,
            }
            if security_group_ids is not None:
                instance_params["SecurityGroupIds"] = security_group_ids

            response = self.ec2_client.run_instances(
                **instance_params, MinCount=1, MaxCount=1
            )
            instance = response["Instances"][0]
            self.instances.append(instance)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=[instance["InstanceId"]])
        except ClientError as err:
            params_str = "\n\t".join(
                f"{key}: {value}" for key, value in instance_params.items()
            )
            logger.error(
                f"Failed to complete instance creation request.\nRequest details:
{params_str}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InstanceLimitExceeded":
                logger.error(
                    (
                        f"Insufficient capacity for instance type
 '{instance_type}'. "
```

```
                        "Terminate unused instances or contact AWS Support for a
limit increase."
                    )
                )
            if error_code == "InsufficientInstanceCapacity":
                logger.error(
                    (
                        f"Insufficient capacity for instance type
'{instance_type}'. "
                        "Select a different instance type or launch in a
different availability zone."
                    )
                )
            raise
        return self.instances
```

- Para obter detalhes da API, consulte a RunInstancesReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
```

```
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }
```

```
        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }
```

- Para obter detalhes da API, consulte a RunInstancesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create tags for resource created during instance launch. "
    DATA lt_tagspecifications TYPE /aws1/
cl_ec2tagspecification=>tt_tagspecificationlist.
    DATA ls_tagspecifications LIKE LINE OF lt_tagspecifications.
    ls_tagspecifications = NEW /aws1/cl_ec2tagspecification(
      iv_resourcetype = 'instance'
      it_tags = VALUE /aws1/cl_ec2tag=>tt_taglist(
        ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
      ) ).
    APPEND ls_tagspecifications TO lt_tagspecifications.

    TRY.
        " Create/launch Amazon Elastic Compute Cloud (Amazon EC2) instance. "
        oo_result = lo_ec2->runinstances(                           " oo_result
 is returned for testing purposes. "
          iv_imageid = iv_ami_id
          iv_instancetype = 't3.micro'
          iv_maxcount = 1
          iv_mincount = 1
          it_tagspecifications = lt_tagspecifications
          iv_subnetid = iv_subnet_id ).
        MESSAGE 'EC2 instance created.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RunInstancesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Create and return a new EC2 instance.
    ///
    /// - Parameters:
    ///   - imageId: The image ID of the AMI to use when creating the instance.
    ///   - instanceType: The type of instance to create.
    ///   - keyPairName: The RSA key pair's name to use to secure the instance.
    ///   - securityGroups: The security group or groups to add the instance
    ///     to.
    ///
    /// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
    func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
                     keyPairName: String, securityGroups: [String]?) async ->
 EC2ClientTypes.Instance? {
        do {
            let output = try await ec2Client.runInstances(
                input: RunInstancesInput(
                    imageId: imageId,
                    instanceType: instanceType,
                    keyName: keyPairName,
```

```
                    maxCount: 1,
                    minCount: 1,
                    securityGroupIds: securityGroups
                )
            )

            guard let instances = output.instances else {
                print("*** Unable to create the instance.")
                return nil
            }

            return instances[0]
        } catch {
            print("*** Error creating the instance:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte RunInstancesa referência da API AWS SDK for Swift.

Usar **`RunScheduledInstances`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RunScheduledInstances`.

CLI

AWS CLI

Executar uma instância programada

Este exemplo inicia a Instância Programada especificada em uma VPC.

Comando:

```
aws ec2 run-scheduled-instances --scheduled-instance-
id sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-
specification file://launch-specification.json
```

Launch-specification.json:

```
{
```

```
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "InstanceType": "c4.large",
  "NetworkInterfaces": [
    {
        "DeviceIndex": 0,
        "SubnetId": "subnet-12345678",
        "AssociatePublicIpAddress": true,
        "Groups": ["sg-12345678"]
    }
  ],
  "IamInstanceProfile": {
      "Name": "my-iam-role"
  }
}
```

Saída:

```
{
  "InstanceIdSet": [
      "i-1234567890abcdef0"
  ]
}
```

Este exemplo inicia a instância agendada especificada em EC2-Classic.

Comando:

```
aws ec2 run-scheduled-instances --scheduled-instance-
id sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-
specification file://launch-specification.json
```

Launch-specification.json:

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": ["sg-12345678"],
  "InstanceType": "c4.large",
  "Placement": {
    "AvailabilityZone": "us-west-2b"
  }
  "IamInstanceProfile": {
```

```
        "Name": "my-iam-role"
    }
}
```

Saída:

```
{
    "InstanceIdSet": [
        "i-1234567890abcdef0"
    ]
}
```

- Para obter detalhes da API, consulte RunScheduledInstancesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo inicia a instância programada especificada.

```
New-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012 -InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678`
-LaunchSpecification_SecurityGroupId sg-12345678
```

- Para obter detalhes da API, consulte RunScheduledInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo inicia a instância programada especificada.

```
New-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012 -InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678`
```

```
-LaunchSpecification_SecurityGroupId sg-12345678
```

- Para obter detalhes da API, consulte RunScheduledInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`StartInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução às máquinas virtuais

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Start an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
/// to start.</param>
/// <returns>Async task.</returns>
public async Task StartInstances(string ec2InstanceId)
{
    try
    {
        var request = new StartInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };
```

```
            await _amazonEC2.StartInstancesAsync(request);

            Console.Write("Waiting for instance to start. ");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while starting the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);
```

```
            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Para obter detalhes da API, consulte StartInstancesa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_start_instances() {
```

```
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 start-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports start-instances operation failed with $response."
    return 1
  }
```

```
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
```

```
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte StartInstancesem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::startInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest startRequest;
    startRequest.AddInstanceIds(instanceId);
    startRequest.SetDryRun(true);

    Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
 ec2Client.StartInstances(startRequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to start instance. A dry run should trigger an
 error."
                << std::endl;
```

```
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
    Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
 ec2Client.StartInstances(startRequest);

    if (!startInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
                  startInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully started instance " << instanceId <<
                  std::endl;
    }

    return startInstancesOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte StartInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para iniciar uma instância do Amazon EC2

Este exemplo inicia a EBS-backed instância especificada da Amazon.

Comando:

```
aws ec2 start-instances --instance-ids i-1234567890abcdef0
```

Saída:

```
{
```

```
    "StartingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
}
```

Para obter mais informações, consulte Interromper e iniciar sua instância no Guia do usuário do Amazon Elastic Compute Cloud.

- Para obter detalhes da API, consulte StartInstancesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
 "running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has
 been started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();
```

```
        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed
to start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
```

- Para obter detalhes da API, consulte StartInstancesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, StartInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Starts an Amazon EBS-backed instance that you've previously stopped.
 * @param {{ instanceIds }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StartInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StartingInstances } = await client.send(command);
    const instanceIdList = StartingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Starting instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte StartInstancesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- Para obter detalhes da API, consulte a StartInstancesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo inicia a instância especificada.

```
Start-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

Exemplo 2: esse exemplo inicia as instâncias especificadas.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Exemplo 3: esse exemplo inicia o conjunto de instâncias que estão atualmente paradas. Os objetos Instance retornados por Get-EC2Instance são canalizados para Start-EC2Instance. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";
 Values="stopped"}).Instances | Start-EC2Instance
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro para o parâmetro Filter.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- Para obter detalhes da API, consulte StartInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo inicia a instância especificada.

```
Start-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

Exemplo 2: esse exemplo inicia as instâncias especificadas.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Exemplo 3: esse exemplo inicia o conjunto de instâncias que estão atualmente paradas. Os objetos Instance retornados por Get-EC2Instance são canalizados para Start-EC2Instance. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";
 Values="stopped"}).Instances | Start-EC2Instance
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro para o parâmetro Filter.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

Exemplo 5: Este exemplo valida as permissões para iniciar uma instância do EC2 usando o DryRun parâmetro sem realmente iniciar a instância. Nota: se isso for feito com sucesso, será gerada uma exceção, que é o comportamento esperado.

```
Start-EC2Instance -InstanceId 'i-0abcdef123456' -Region 'us-west-1' -DryRun $true
```

- Para obter detalhes da API, consulte StartInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
```

```
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def start(self) -> Optional[Dict[str, Any]]:
        """
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
```

```
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=instance_ids)
            return start_response
        except ClientError as err:
            logger.error(
                f"Failed to start instance(s): {','.join(map(str,
instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't start instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a stopped state before starting
them."
                )
            raise
```

- Para obter detalhes da API, consulte a StartInstancesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Attempts to start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
```

```
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was started; otherwise, false.
# @example
#   exit 1 unless instance_started?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'pending'
      puts 'Error starting instance: the instance is pending. Try again later.'
      return false
    when 'running'
      puts 'The instance is already running.'
      return true
    when 'terminated'
      puts 'Error starting instance: ' \
        'the instance is terminated, so you cannot start it.'
      return false
    end
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance started.'
  true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
```

```
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to start instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_started?(ec2_client, instance_id)

  puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte StartInstancesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicie uma instância do EC2 por ID de instância.

```
    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }
```

Aguarde até que uma instância esteja pronta e com status OK, usando a API Waiters. O uso da API Waiters requer “use aws_sdk_ec2::client::Waiters” no arquivo rust.

```
    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a StartInstancesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to
 start the instance without actually making the request. "
        lo_ec2->startinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to start this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to start instance completed.' TYPE 'I'.
          " DryRun is set to false to start instance. "
          oo_result = lo_ec2->startinstances(           " oo_result is returned
 for testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Successfully started the EC2 instance.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to start this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to start instance failed. User does not have
 permissions to start the instance.' TYPE 'E'.
        ELSE.
```

```
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartInstancesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance
    ///     has started. Otherwise, execution continues and the instance starts
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully started (or is left to
    ///   start asynchronously). `false` if the instance doesn't start.
    func startInstance(instanceId: String, waitUntilStarted: Bool = false) async
 -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.startInstances(
                input: StartInstancesInput(
                    instanceIds: instanceList
                )
```

```
            )

            if waitUntilStarted {
                print("Waiting for the instance to start...")

                let waitOptions = WaiterOptions(maxWaitTime: 60.0)
                let output = try await ec2Client.waitUntilInstanceRunning(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to start the instance:
\(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte StartInstancesa referência da API AWS SDK for Swift.

Use **`StopInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StopInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução às máquinas virtuais

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Stop an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance to
    /// stop.</param>
    /// <returns>Async task.</returns>
    public async Task StopInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StopInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StopInstancesAsync(request);
            Console.Write("Waiting for the instance to stop.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);

            Console.WriteLine("\nThe instance has stopped.");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
            }

            throw;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while stopping the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Para obter detalhes da API, consulte StopInstancesa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_stop_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_stop_instances"
    echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
```

```
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte StopInstancesem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Stop an EC2 instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::stopInstance(const Aws::String &instanceId,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
 ec2Client.StopInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to stop instance. A dry run should trigger an
 error."
                << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::StopInstancesOutcome outcome =
 ec2Client.StopInstances(request);
```

```
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully stopped instance " << instanceId <<
                  std::endl;
    }

    return outcome.IsSuccess();
}

void PrintUsage() {
    std::cout << "Usage: run_start_stop_instance <instance_id> <start|stop>" <<
              std::endl;
}
```

- Para obter detalhes da API, consulte StopInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: interromper uma instância do Amazon EC2

O `stop-instances` exemplo a seguir interrompe a EBS-backed instância especificada da Amazon.

```
aws ec2 stop-instances \
    --instance-ids i-1234567890abcdef0
```

Saída:

```
{
    "StoppingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
```

```
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Para obter mais informações, consulte Interromper e iniciar sua instância no Guia do usuário do Amazon Elastic Compute Cloud.

Exemplo 2: hibernar uma instância do Amazon EC2

O `stop-instances` exemplo a seguir hiberna a EBS-backed instância da Amazon se ela estiver habilitada para hibernação e atender aos pré-requisitos de hibernação. Depois de colocar a instância em hibernação, ela é interrompida.

```
aws ec2 stop-instances \
    --instance-ids i-1234567890abcdef0 \
    --hibernate
```

Saída:

```
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-1234567890abcdef0",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Para obter mais informações, consulte Hibernate sua instância On-Demand Linux no Amazon Elastic Cloud Compute User Guide.

- Para obter detalhes da API, consulte StopInstancesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Stops the EC2 instance with the specified ID asynchronously and waits for
the instance to stop.
    *
    * @param instanceId the ID of the EC2 instance to stop
    * @return a {@link CompletableFuture} that completes when the instance has
been stopped, or exceptionally if an error occurs
    */
   public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
       StopInstancesRequest stopRequest = StopInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
           .client(getAsyncClient())
           .build();

       CompletableFuture<Void> resultFuture = new CompletableFuture<>();
       logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
       getAsyncClient().stopInstances(stopRequest)
           .thenCompose(response -> {
               if (response.stoppingInstances().isEmpty()) {
                   return CompletableFuture.failedFuture(new
RuntimeException("No instances were stopped. Please check the instance ID: " +
instanceId));
               }
```

```
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
 throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed
 to stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }
```

- Para obter detalhes da API, consulte StopInstancesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, StopInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Stop one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
```

```
  const command = new StopInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte StopInstancesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
```

```
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

- Para obter detalhes da API, consulte a StopInstancesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo interrompe a instância especificada.

```
Stop-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para obter detalhes da API, consulte StopInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo interrompe a instância especificada.

```
Stop-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para obter detalhes da API, consulte StopInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
```

```
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no
instances to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response
```

- Para obter detalhes da API, consulte a StopInstancesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'stopping'
      puts 'The instance is already stopping.'
      return true
    when 'stopped'
      puts 'The instance is already stopped.'
      return true
    when 'terminated'
```

```
      puts 'Error stopping instance: ' \
        'the instance is terminated, so you cannot stop it.'
      return false
    end
  end

  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_stopped?(ec2_client, instance_id)
```

```
  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte StopInstancesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }
```

Aguarde até que uma instância esteja no estado interrompido, usando a API Waiters. O uso da API Waiters requer “use aws_sdk_ec2::client::Waiters” no arquivo rust.

```
    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
 {
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }
```

- Para obter detalhes da API, consulte a StopInstancesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to stop
 the instance without actually making the request. "
        lo_ec2->stopinstances(
```

```
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to stop this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to stop instance completed.' TYPE 'I'.
          " DryRun is set to false to stop instance. "
          oo_result = lo_ec2->stopinstances(           " oo_result is returned
 for testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Successfully stopped the EC2 instance.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to stop this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to stop instance failed. User does not have
 permissions to stop the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StopInstancesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2
```

```
    /// Stop the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to stop.
    ///   - waitUntilStopped: If `true`, execution waits until the instance
    ///     has stopped. Otherwise, execution continues and the instance stops
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully stopped (or is left to
    ///   stop asynchronously). `false` if the instance doesn't stop.
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async -
> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance:
 \(error.localizedDescription)")
```

```
            return false
        }
    }
```

- Para obter detalhes da API, consulte StopInstancesa referência da API AWS SDK for Swift.

Use **`TerminateInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TerminateInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma rede privada virtual básica
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criação e gerenciamento de volumes de armazenamento em bloco
- Conceitos básicos de compras do mercado de software
- Conceitos básicos de streaming gerenciado
- Introdução às máquinas virtuais
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Terminate an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance
    /// to terminate.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
    {
        try
        {
            var request = new TerminateInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId }
            };

            var response = await _amazonEC2.TerminateInstancesAsync(request);
            Console.Write("Waiting for the instance to terminate.");
            await WaitForInstanceState(ec2InstanceId,
InstanceStateName.Terminated);

            Console.WriteLine($"\nThe instance {ec2InstanceId} has been
terminated.");
            return response.TerminatingInstances;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while terminating the instance.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
```

```
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Para obter detalhes da API, consulte TerminateInstancesa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
#       -i instance_ids - A space-separated list of instance IDs.
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_terminate_instances() {
  local instance_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_terminate_instances"
    echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_ids - A space-separated list of instance IDs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  # Check if instance ID is provided
  if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
```

```
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte TerminateInstancesem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
```

```
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::terminateInstances(const Aws::String &instanceID,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::TerminateInstancesRequest request;
    request.SetInstanceIds({instanceID});

    Aws::EC2::Model::TerminateInstancesOutcome outcome =
            ec2Client.TerminateInstances(request);
    if (outcome.IsSuccess()) {
        std::cout << "Ec2 instance '" << instanceID <<
                  "' was terminated." << std::endl;
    } else {
        std::cerr << "Failed to terminate ec2 instance " << instanceID <<
                  ", " <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte TerminateInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para encerrar uma instância do Amazon EC2

Este exemplo encerra a instância especificada.

Comando:

```
aws ec2 terminate-instances --instance-ids i-1234567890abcdef0
```

Saída:

```
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Para obter mais informações, consulte Using Amazon EC2 Instances no Guia do usuário da AWS Command Line Interface.

- Para obter detalhes da API, consulte TerminateInstancesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Terminates an EC2 instance asynchronously and waits for it to reach the
 terminated state.
     *
     * @param instanceId the ID of the EC2 instance to terminate
     * @return a {@link CompletableFuture} that completes when the instance has
 been terminated
```

```
    * @throws RuntimeException if there is no response from the AWS SDK or if
there is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
           System.out.println("Going to terminate an EC2 instance and use a
waiter to wait for it to be in terminated state");
           return getAsyncClient().waiter()
               .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
               .thenApply(waiterResponse -> null);
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
       });
   }
```

- Para obter detalhes da API, consulte TerminateInstancesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, TerminateInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Terminate one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new TerminateInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { TerminatingInstances } = await client.send(command);
    const instanceList = TerminatingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Terminating instances:");
    console.log(instanceList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte TerminateInstancesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is
 ${instance.instanceId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a TerminateInstancesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo encerra a instância especificada (a instância pode estar em execução ou no estado “interrompido”). O cmdlet solicitará uma confirmação antes de continuar. Use a opção -Force para suprimir a solicitação.

```
Remove-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para obter detalhes da API, consulte TerminateInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo encerra a instância especificada (a instância pode estar em execução ou no estado “interrompido”). O cmdlet solicitará uma confirmação antes de continuar. Use a opção -Force para suprimir a solicitação.

```
Remove-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para obter detalhes da API, consulte TerminateInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
```

```
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")
```

```
        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
            raise
```

- Para obter detalhes da API, consulte a TerminateInstancesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
```

```
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
     response.instance_statuses[0].instance_state.name == 'terminated'

    puts 'The instance is already terminated.'
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
  true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)
```

```
  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_terminated?(ec2_client, instance_id)

  puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte TerminateInstancesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn delete_instance(&self, instance_id: &str) -> Result<(),
 EC2Error> {
        tracing::info!("Deleting instance with id {instance_id}");
        self.stop_instance(instance_id).await?;
        self.client
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await?;
        self.wait_for_instance_terminated(instance_id).await?;
        tracing::info!("Terminated instance with id {instance_id}");
        Ok(())
    }
```

Aguarde até que uma instância esteja no estado encerrado, usando a API Waiters. O uso da API Waiters requer “use aws_sdk_ec2::client::Waiters” no arquivo rust.

```
    async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
 EC2Error> {
        self.client
            .wait_until_instance_terminated()
            .instance_ids(instance_id)
            .wait(Duration::from_secs(60))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to terminate.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a TerminateInstancesreferência da API AWS SDK for Rust.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Terminate the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to terminate.
    ///   - waitUntilTerminated: Whether or not to wait until the instance is
    ///     terminated before returning.
    ///
    /// - Returns: `true` if terminated successfully. `false` if not or if an
```

```
    ///   error occurs.
    func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
 async -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.terminateInstances(
                input: TerminateInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilTerminated {
                print("Waiting for the instance to terminate...")

                let waitOptions = WaiterOptions(maxWaitTime: 600.0)
                let output = try await ec2Client.waitUntilInstanceTerminated(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to terminate the instance:
 \(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte TerminateInstancesa referência da API AWS SDK for Swift.

Usar **`UnassignPrivateIpAddresses`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UnassignPrivateIpAddresses`.

CLI

AWS CLI

Cancelar a atribuição de um endereço IP privado secundário de uma interface de rede

Este exemplo cancela a atribuição do endereço IP privado especificado da interface de rede especificada. Se o comando for bem-sucedido, nenhuma saída será retornada.

Comando:

```
aws ec2 unassign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
private-ip-addresses 10.0.0.82
```

- Para obter detalhes da API, consulte UnassignPrivateIpAddressesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela a atribuição do endereço IP privado especificado da interface de rede especificada.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- Para obter detalhes da API, consulte UnassignPrivateIpAddressesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a atribuição do endereço IP privado especificado da interface de rede especificada.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- Para obter detalhes da API, consulte UnassignPrivateIpAddressesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`UnmonitorInstances`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UnmonitorInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Disable monitoring for an EC2 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::disableMonitoring(const Aws::String &instanceId,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instanceId);
    unrequest.SetDryRun(true);

    Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
 ec2Client.UnmonitorInstances(unrequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
```

```
                << "Failed dry run to disable monitoring on instance. A dry run
 should trigger an error."
                <<
                std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
                  instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                  std::endl;
        return false;
    }

    unrequest.SetDryRun(false);
    Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
 ec2Client.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to disable monitoring on instance " << instanceId
                  << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
                  std::endl;
    } else {
        std::cout << "Successfully disable monitoring on instance " <<
                  instanceId << std::endl;
    }

    return unmonitorInstancesOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UnmonitorInstancesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para desabilitar o monitoramento detalhado de uma instância

Este exemplo de comando desabilita o monitoramento detalhado da instância especificada.

Comando:

```
aws ec2 unmonitor-instances --instance-ids i-1234567890abcdef0
```

Saída:

```
{
  "InstanceMonitorings": [
      {
          "InstanceId": "i-1234567890abcdef0",
          "Monitoring": {
              "State": "disabling"
          }
      }
  ]
}
```

- Para obter detalhes da API, consulte UnmonitorInstancesem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, UnmonitorInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Turn off detailed monitoring for the selected instance.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new UnmonitorInstancesCommand({
    InstanceIds: instanceIds,
```

```
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instanceMonitoringsList = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
 ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instanceMonitoringsList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte UnmonitorInstancesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo de comando desabilita o monitoramento detalhado da instância especificada.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Saída:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Para obter detalhes da API, consulte UnmonitorInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo de comando desabilita o monitoramento detalhado da instância especificada.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Saída:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Para obter detalhes da API, consulte UnmonitorInstancesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateSecurityGroupRuleDescriptionsIngress`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateSecurityGroupRuleDescriptionsIngress`.

CLI

AWS CLI

Exemplo 1: atualizar a descrição de uma regra de grupo de segurança de entrada com uma fonte CIDR

O exemplo `update-security-group-rule-descriptions-ingress` a seguir atualiza a descrição da regra do grupo de segurança para a porta especificada e o intervalo de endereços IPv4. A descrição '`SSH access from ABC office`' substitui qualquer descrição existente para a regra.

```
aws ec2 update-security-group-rule-descriptions-ingress \
    --group-id sg-02f0d35a850ba727f \
    --ip-permissions
 IpProtocol=tcp,FromPort=22,ToPort=22,IpRanges='[{CidrIp=203.0.113.0/16,Description="SSH
 access from corpnet"}]'
```

Saída:

```
{
    "Return": true
}
```

Consulte mais informações em Regras de grupo de segurança no Guia do usuário do Amazon EC2.

Exemplo 2: atualizar a descrição de uma regra de grupo de segurança de entrada com uma fonte de lista de prefixos

O exemplo `update-security-group-rule-descriptions-ingress` a seguir atualiza a descrição da regra do grupo de segurança para a porta especificada e a lista de prefixos. A descrição '`SSH access from ABC office`' substitui qualquer descrição existente para a regra.

```
aws ec2 update-security-group-rule-descriptions-ingress \
    --group-id sg-02f0d35a850ba727f \
    --ip-permissions
 IpProtocol=tcp,FromPort=22,ToPort=22,PrefixListIds='[{PrefixListId=pl-12345678,Descripti
 access from corpnet"}]'
```

Saída:

```
{
    "Return": true
}
```

Consulte mais informações em Regras de grupo de segurança no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte UpdateSecurityGroupRuleDescriptionsIngressem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: atualiza a descrição de uma regra de grupo de segurança de entrada.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
 $existingInboundRule.GroupId -SecurityGroupRuleDescription
 $ruleWithUpdatedDescription
```

Exemplo 2: remove a descrição de uma regra de grupo de segurança de entrada (entrada) existente (omitindo o parâmetro na solicitação).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
 $existingInboundRule.GroupId -SecurityGroupRuleDescription
 $ruleWithoutDescription
```

- Para obter detalhes da API, consulte UpdateSecurityGroupRuleDescriptionsIngressem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: atualiza a descrição de uma regra de grupo de segurança de entrada.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
 $existingInboundRule.GroupId -SecurityGroupRuleDescription
 $ruleWithUpdatedDescription
```

Exemplo 2: remove a descrição de uma regra de grupo de segurança de entrada (entrada) existente (omitindo o parâmetro na solicitação).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
 $existingInboundRule.GroupId -SecurityGroupRuleDescription
 $ruleWithoutDescription
```

- Para obter detalhes da API, consulte UpdateSecurityGroupRuleDescriptionsIngressem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para o Amazon EC2 usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon EC2 com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon EC2 ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie e gerencie um serviço resiliente usando um AWS SDK
- Configurar a conectividade do serviço de contêiner
- Introdução às redes privadas virtuais
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Criação de uma instância de banco de dados relacional
- Criação e gerenciamento de uma malha de rede de serviços

- Criação e gerenciamento de volumes de armazenamento em bloco
- Comece com conexões de rede dedicadas usando a CLI
- Comece a comprar no mercado de software usando a CLI
- Introdução aos clusters de processamento de big data
- Introdução aos bancos de dados de documentos
- Introdução aos bancos de dados de grafos
- Introdução ao armazenamento em cache em memória
- Introdução ao balanceamento de carga
- Conceitos básicos de streaming gerenciado
- Introdução aos gateways de trânsito de rede
- Introdução à transmissão de transporte de vídeo
- Introdução às máquinas virtuais
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Comece a usar o gerenciamento de endereços IP usando a CLI
- Crie uma rede privada virtual com sub-redes privadas e gateways de conversão de endereços de rede usando a CLI
- Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco
- Trabalho com conexões de emparelhamento de rede

Crie e gerencie um serviço resiliente usando um AWS SDK

Os exemplos de código a seguir mostram como criar um serviço da Web com carga balanceada que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.

- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de saúde atualizando AWS Systems Manager os parâmetros.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
```

```
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }
```

```
    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");
```

```
        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
```

```
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
```

```
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
```

```
                    "\nFor this example to work, the default security group for
your default VPC must\n"
                    + "allows access from this computer. You can either add it
automatically from this\n"
                    + "example or add it yourself using the AWS Management
Console.\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
```

```
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
```

```
                        $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
                        "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
```

```
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
```

```
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");
```

```
        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
```

```
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
```

```
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
 a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
 granted to
```

```
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
 the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
```

```
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
```

```
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
```

```
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
 Scaling.
    /// The launch template specifies a Bash script in its user data field that
 runs after
```

```
   /// the instance is started. This script installs the Python packages and
starts a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
```

```
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
 zones.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
```

```
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
```

```
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
```

```
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
```

```
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
```

```
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
```

```
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);
```

```
            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
```

```
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
```

```
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
```

```
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
calling computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
    /// If you do, be sure to remove public access when you're done.
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
```

```
                    Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                        "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
```

```
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
```

```
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }
```

```
    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
```

```
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
```

```
    {
        var createResponse = await
_amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
```

```
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
```

```
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
```

```
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
 _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
```

```
                    new DeleteTargetGroupRequest() { TargetGroupArn =
 targetArn });
                Console.WriteLine($"Deleted load balancing target group
 {groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
```

```
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
```

```
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
_amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
```

```
    /// <param name="recommendationsPath">The path of the recommendations data.</
 param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
  recommendationsPath)
    {
        var recommendationsText = await
  File.ReadAllTextAsync(recommendationsPath);
        var records =

  JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
```

```
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones

- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
```

```
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
```

```
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
```

```
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
```

```
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }
```

```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
```

```
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns
a static response.
               The service still reports as healthy because health checks are
still shallow.
               """);
       demoChoices(loadBalancer);

       System.out.println("Let's reinstate the recommendation service.");
       paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

       System.out.println("""
               Let's also substitute bad credentials for one of the instances in
the target group so that it can't
               access the DynamoDB recommendation table. We will get an instance
id value.
               """);

       LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
       AutoScaler autoScaler = new AutoScaler();

       // Create a new instance profile based on badCredsProfileName.
       templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
       String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
       System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

       String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
```

```
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
```

```
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load
balancing rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance
is running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
```

```
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
```

```
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
balancer targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }
```

```
    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
```

```
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
```

```
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                                // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
```

```
            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
```

```
   public void deleteInstanceProfile(String roleName, String profileName) {
       try {
           software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                   .builder()
                   .instanceProfileName(profileName)
                   .build();

           GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
           String name = response.instanceProfile().instanceProfileName();
           System.out.println(name);

           RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .roleName(roleName)
                   .build();

           getIAMClient().removeRoleFromInstanceProfile(profileRequest);
           DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .build();

           getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
           System.out.println("Deleted instance profile " + profileName);

           DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                   .roleName(roleName)
                   .build();

           // List attached role policies.
           ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                   .listAttachedRolePolicies(role -> role.roleName(roleName));
           List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
           for (AttachedPolicy attachedPolicy : attachedPolicies) {
               DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                       .roleName(roleName)
                       .policyArn(attachedPolicy.policyArn())
                       .build();
```

```
                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
```

```
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
```

```
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();
```

```
getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
           System.out.println("Attached load balancer to " + asGroupName);

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

   // Creates an EC2 Auto Scaling group with the specified size.
   public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

       // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
               .builder()
               .build();

       DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
       List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
               .collect(Collectors.toList());

       String availabilityZones = String.join(",", availabilityZoneNames);
       LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
               .launchTemplateName(templateName)
               .version("$Default")
               .build();

       String[] zones = availabilityZones.split(",");
       CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
               .launchTemplate(specification)
               .availabilityZones(zones)
               .maxSize(groupSize)
```

```
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
```

```
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
```

```
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
 DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();
```

```
                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();
```

```
        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
```

```
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
```

```
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load
balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
```

```
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
       CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
               .healthCheckPath("/healthcheck")
               .healthCheckTimeoutSeconds(5)
               .port(port)
               .vpcId(vpcId)
               .name(targetGroupName)
               .protocol(protocol)
               .build();

       CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
       String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
       String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
       System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
       return targetGroupArn;
   }

   /*
    * Creates an Elastic Load Balancing load balancer that uses the specified
    * subnets
    * and forwards requests to the specified target group.
    */
   public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
```

```
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();
```

```
            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;
```

```
        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
```

```
                                        .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                        .readCapacityUnits(5L)
                                        .writeCapacityUnits(5L)
                                        .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
```

```
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth

- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
```

```
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Criar etapas para implantar todos os recursos.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
```

```
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
```

```
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
```

```
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
```

```
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
```

```
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
```

```
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
```

```
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
```

```
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
```

```
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
```

```
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
```

```
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
```

```
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
```

```
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
```

```
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Criar etapas para executar a demonstração.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
```

```
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
```

```
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
```

```
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
```

```
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
```

```
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
```

```
    await createSsmOnlyInstanceProfile();
    const autoScalingClient = new AutoScalingClient({});
    const { AutoScalingGroups } = await autoScalingClient.send(
      new DescribeAutoScalingGroupsCommand({
        AutoScalingGroupNames: [NAMES.autoScalingGroupName],
      }),
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
```

```
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
```

```
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
```

```
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
```

```
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
```

```
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Criar etapas para destruir todos os recursos.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
```

```
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
```

```
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
```

```
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
```

```
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
```

```
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
```

```
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
```

```
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
```

```
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
```

```
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
```

```
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
```

```
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
```

```
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
```

```
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
```

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
```

```
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")
```

```
        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
```

```
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
```

```
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
```

```
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
```

```
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
```

```
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
```

```
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
    for action in actions:
        if action == "deploy":
```

```
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
```

```
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
```

```
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
```

```
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
 attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
 {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
 created by
        this class. An instance's associated profile defines a role that is
 assumed by the
        instance. The role has attached policies that specify the AWS permissions
 granted to
        clients that run on the instance.
```

```
        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
```

```
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
```

```
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
```

```
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
```

```
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
```

```
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
```

```
            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
```

```
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
```

```
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
```

```
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
```

```
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
```

```
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
"
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
```

```
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
 from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
 you
        must instead specify a prefix list ID. You can also temporarily open the
 port to
        any IP address while running this example. If you do, be sure to remove
 public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
 that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")
```

```
            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
```

```
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
```

```
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""
```

```
    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
```

```
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )
```

```
    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
```

```
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
```

```
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
```

```
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
```

```
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
```

```
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
 found. "
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
 parameters.
```

```
        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
```

```
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
self.table_name)
            else:
```

```
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Crie uma classe que envolva as ações do Systems Manager.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
```

```
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate

- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação

- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
```

```
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
```

```
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
```

```
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
```

```
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"
```

```
    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
```

```
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
```

```
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
```

```
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
```

```
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
```

```
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
```

```
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
```

```
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
```

```
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure
```

```
    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
dependency issues."
    fi
```

```
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNamespaces
  - ListServices
  - ListTaskDefinitions

- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Introdução às redes privadas virtuais

O exemplo de código a seguir mostra como:

- Criar uma VPC.
- Criar sub-redes.
- Configurar a conectividade com a internet.
- Criar um gateway NAT.
- Definir as configurações da sub-rede.
- Criar grupos de segurança.
- Verificar a configuração de sua VPC.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC Creation Script
# This script creates a VPC with public and private subnets, internet gateway,
 NAT gateway, and security groups

# Set up logging
LOG_FILE="vpc_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
# Function to handle errors
handle_error() {
  echo "ERROR: $1"
  echo "Resources created before error:"
  for resource in "${CREATED_RESOURCES[@]}"
  do
    echo "- $resource"
  done

  echo "Attempting to clean up resources..."
  cleanup_resources
  exit 1
}

# Function to clean up resources
cleanup_resources() {
  echo "Cleaning up resources in reverse order..."

  # Reverse the array to delete in reverse order of creation
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--))
  do
    resource="${CREATED_RESOURCES[$i]}"
    resource_type=$(echo "$resource" | cut -d':' -f1)
    resource_id=$(echo "$resource" | cut -d':' -f2)

    case "$resource_type" in
      "INSTANCE")
        echo "Terminating EC2 instance: $resource_id"
        aws ec2 terminate-instances --instance-ids "$resource_id" || echo "Failed
 to terminate instance: $resource_id"
        # Wait for instance to terminate
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$resource_id" || echo
 "Failed to wait for instance termination: $resource_id"
        ;;
      "KEY_PAIR")
        echo "Deleting key pair: $resource_id"
        aws ec2 delete-key-pair --key-name "$resource_id" || echo "Failed to
 delete key pair: $resource_id"
        # Remove the .pem file if it exists
        if [ -f "${resource_id}.pem" ]; then
          rm -f "${resource_id}.pem"
        fi
```

```
        ;;
      "NAT_GATEWAY")
        echo "Deleting NAT Gateway: $resource_id"
        aws ec2 delete-nat-gateway --nat-gateway-id "$resource_id" || echo
 "Failed to delete NAT Gateway: $resource_id"
        # NAT Gateway deletion takes time, wait for it to complete
        echo "Waiting for NAT Gateway to be deleted..."
        aws ec2 wait nat-gateway-deleted --nat-gateway-ids "$resource_id" || echo
 "Failed to wait for NAT Gateway deletion: $resource_id"
        ;;
      "EIP")
        echo "Releasing Elastic IP: $resource_id"
        aws ec2 release-address --allocation-id "$resource_id" || echo "Failed to
 release Elastic IP: $resource_id"
        ;;
      "ROUTE_TABLE_ASSOCIATION")
        echo "Disassociating Route Table: $resource_id"
        aws ec2 disassociate-route-table --association-id "$resource_id" || echo
 "Failed to disassociate Route Table: $resource_id"
        ;;
      "ROUTE_TABLE")
        echo "Deleting Route Table: $resource_id"
        aws ec2 delete-route-table --route-table-id "$resource_id" || echo
 "Failed to delete Route Table: $resource_id"
        ;;
      "INTERNET_GATEWAY")
        echo "Detaching Internet Gateway: $resource_id from VPC: $VPC_ID"
        aws ec2 detach-internet-gateway --internet-gateway-id "$resource_id" --
vpc-id "$VPC_ID" || echo "Failed to detach Internet Gateway: $resource_id"
        echo "Deleting Internet Gateway: $resource_id"
        aws ec2 delete-internet-gateway --internet-gateway-id "$resource_id" ||
 echo "Failed to delete Internet Gateway: $resource_id"
        ;;
      "SECURITY_GROUP")
        echo "Deleting Security Group: $resource_id"
        aws ec2 delete-security-group --group-id "$resource_id" || echo "Failed
 to delete Security Group: $resource_id"
        ;;
      "SUBNET")
        echo "Deleting Subnet: $resource_id"
        aws ec2 delete-subnet --subnet-id "$resource_id" || echo "Failed to
 delete Subnet: $resource_id"
        ;;
      "VPC")
```

```
        echo "Deleting VPC: $resource_id"
        aws ec2 delete-vpc --vpc-id "$resource_id" || echo "Failed to delete VPC:
 $resource_id"
        ;;
    esac
  done
}

# Initialize array to track created resources
CREATED_RESOURCES=()

echo "Starting VPC creation script at $(date)"

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
aws configure list || handle_error "AWS CLI is not properly configured"

# Verify identity and permissions
echo "Verifying identity and permissions..."
if ! aws sts get-caller-identity; then
  echo "ERROR: Unable to verify AWS identity. This could be due to:"
  echo "  - Expired credentials"
  echo "  - Missing or invalid AWS credentials"
  echo "  - Insufficient permissions"
  echo ""
  echo "Please run 'aws configure' to update your credentials or check your IAM
 permissions."
  exit 1
fi

# Create VPC
echo "Creating VPC with CIDR block 10.0.0.0/16..."
VPC_ID=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 'ResourceType=vpc,Tags=[{Key=Name,Value=MyVPC},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'Vpc.VpcId' --output text)

if [ -z "$VPC_ID" ]; then
  handle_error "Failed to create VPC"
fi

CREATED_RESOURCES+=("VPC:$VPC_ID")
echo "VPC created with ID: $VPC_ID"

# Enable DNS support and hostnames
```

```
echo "Enabling DNS support and hostnames for VPC..."
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-support ||
 handle_error "Failed to enable DNS support"
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-hostnames ||
 handle_error "Failed to enable DNS hostnames"

# Get available Availability Zones
echo "Getting available Availability Zones..."
AZ1=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[0].ZoneName'
 --output text)
AZ2=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[1].ZoneName'
 --output text)

if [ -z "$AZ1" ] || [ -z "$AZ2" ]; then
  handle_error "Failed to get Availability Zones"
fi

echo "Using Availability Zones: $AZ1 and $AZ2"

# Create public subnets
echo "Creating public subnet in $AZ1..."
PUBLIC_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.0.0/24 \
  --availability-zone "$AZ1" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ1},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ1" ]; then
  handle_error "Failed to create public subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ1")
echo "Public subnet created in $AZ1 with ID: $PUBLIC_SUBNET_AZ1"

echo "Creating public subnet in $AZ2..."
PUBLIC_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.1.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ2},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
```

```
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ2" ]; then
  handle_error "Failed to create public subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ2")
echo "Public subnet created in $AZ2 with ID: $PUBLIC_SUBNET_AZ2"

# Create private subnets
echo "Creating private subnet in $AZ1..."
PRIVATE_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.2.0/24 \
  --availability-zone "$AZ1" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ1},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PRIVATE_SUBNET_AZ1" ]; then
  handle_error "Failed to create private subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ1")
echo "Private subnet created in $AZ1 with ID: $PRIVATE_SUBNET_AZ1"

echo "Creating private subnet in $AZ2..."
PRIVATE_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.3.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ2},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PRIVATE_SUBNET_AZ2" ]; then
  handle_error "Failed to create private subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ2")
echo "Private subnet created in $AZ2 with ID: $PRIVATE_SUBNET_AZ2"
```

```
# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_ID=$(aws ec2 create-internet-gateway \
  --tag-specifications 'ResourceType=internet-
gateway,Tags=[{Key=Name,Value=MyIGW},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' \
  --query 'InternetGateway.InternetGatewayId' \
  --output text)

if [ -z "$IGW_ID" ]; then
  handle_error "Failed to create Internet Gateway"
fi

CREATED_RESOURCES+=("INTERNET_GATEWAY:$IGW_ID")
echo "Internet Gateway created with ID: $IGW_ID"

# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
 "$VPC_ID" || handle_error "Failed to attach Internet Gateway to VPC"

# Create public route table
echo "Creating public route table..."
PUBLIC_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Public-
RT},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PUBLIC_RT" ]; then
  handle_error "Failed to create public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PUBLIC_RT")
echo "Public route table created with ID: $PUBLIC_RT"

# Add route to Internet Gateway
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT" --destination-cidr-block
 0.0.0.0/0 --gateway-id "$IGW_ID" || handle_error "Failed to add route to
 Internet Gateway"
```

```
# Associate public subnets with public route table
echo "Associating public subnet in $AZ1 with public route table..."
PUBLIC_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT"
 --subnet-id "$PUBLIC_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_1" ]; then
  handle_error "Failed to associate public subnet in AZ1 with public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_1")

echo "Associating public subnet in $AZ2 with public route table..."
PUBLIC_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT"
 --subnet-id "$PUBLIC_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_2" ]; then
  handle_error "Failed to associate public subnet in AZ2 with public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_2")

# Create private route table
echo "Creating private route table..."
PRIVATE_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Private-
RT},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PRIVATE_RT" ]; then
  handle_error "Failed to create private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PRIVATE_RT")
echo "Private route table created with ID: $PRIVATE_RT"

# Associate private subnets with private route table
echo "Associating private subnet in $AZ1 with private route table..."
PRIVATE_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT"
 --subnet-id "$PRIVATE_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_1" ]; then
```

```
  handle_error "Failed to associate private subnet in AZ1 with private route
 table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_1")

echo "Associating private subnet in $AZ2 with private route table..."
PRIVATE_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT"
 --subnet-id "$PRIVATE_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_2" ]; then
  handle_error "Failed to associate private subnet in AZ2 with private route
 table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_2")

# Allocate Elastic IP for NAT Gateway
echo "Allocating Elastic IP for NAT Gateway..."
EIP_ALLOC=$(aws ec2 allocate-address --domain vpc --tag-specifications
 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'AllocationId' --output text)

if [ -z "$EIP_ALLOC" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

CREATED_RESOURCES+=("EIP:$EIP_ALLOC")
echo "Elastic IP allocated with ID: $EIP_ALLOC"

# Create NAT Gateway
echo "Creating NAT Gateway in public subnet in $AZ1..."
NAT_GW=$(aws ec2 create-nat-gateway \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --allocation-id "$EIP_ALLOC" \
  --tag-specifications
 'ResourceType=natgateway,Tags=[{Key=Name,Value=MyNATGateway},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'NatGateway.NatGatewayId' \
  --output text)

if [ -z "$NAT_GW" ]; then
  handle_error "Failed to create NAT Gateway"
fi
```

```
CREATED_RESOURCES+=("NAT_GATEWAY:$NAT_GW")
echo "NAT Gateway created with ID: $NAT_GW"

# Wait for NAT Gateway to be available
echo "Waiting for NAT Gateway to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW" || handle_error
 "NAT Gateway did not become available"

# Add route to NAT Gateway in private route table
echo "Adding route to NAT Gateway in private route table..."
aws ec2 create-route --route-table-id "$PRIVATE_RT" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW" || handle_error "Failed to add route to NAT
 Gateway"

# Enable auto-assign public IP for instances in public subnets
echo "Enabling auto-assign public IP for instances in public subnet in $AZ1..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ1" --map-public-
ip-on-launch || handle_error "Failed to enable auto-assign public IP for public
 subnet in AZ1"

echo "Enabling auto-assign public IP for instances in public subnet in $AZ2..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ2" --map-public-
ip-on-launch || handle_error "Failed to enable auto-assign public IP for public
 subnet in AZ2"

# Create security group for web servers
echo "Creating security group for web servers..."
WEB_SG=$(aws ec2 create-security-group \
  --group-name "WebServerSG-$(date +%s)" \
  --description "Security group for web servers" \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'GroupId' \
  --output text)

if [ -z "$WEB_SG" ]; then
  handle_error "Failed to create security group for web servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$WEB_SG")
echo "Security group for web servers created with ID: $WEB_SG"
```

```
# Allow HTTP and HTTPS traffic
echo "Allowing HTTP traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --
port 80 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTP traffic"

echo "Allowing HTTPS traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --
port 443 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTPS traffic"

# Note: In a production environment, you should restrict the source IP ranges for
 security
echo "NOTE: In a production environment, you should restrict the source IP ranges
 for HTTP and HTTPS traffic"

# Create security group for database servers
echo "Creating security group for database servers..."
DB_SG=$(aws ec2 create-security-group \
  --group-name "DBServerSG-$(date +%s)" \
  --description "Security group for database servers" \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'GroupId' \
  --output text)

if [ -z "$DB_SG" ]; then
  handle_error "Failed to create security group for database servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$DB_SG")
echo "Security group for database servers created with ID: $DB_SG"

# Allow MySQL/Aurora traffic from web servers only
echo "Allowing MySQL/Aurora traffic from web servers to database servers..."
aws ec2 authorize-security-group-ingress --group-id "$DB_SG" --protocol tcp --
port 3306 --source-group "$WEB_SG" || handle_error "Failed to allow MySQL/Aurora
 traffic"

# Verify VPC configuration
echo "Verifying VPC configuration..."
echo "VPC:"
aws ec2 describe-vpcs --vpc-id "$VPC_ID" || handle_error "Failed to describe VPC"

echo "Subnets:"
```

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" || handle_error
 "Failed to describe subnets"

echo "Route tables:"
aws ec2 describe-route-tables --filters "Name=vpc-id,Values=$VPC_ID" ||
 handle_error "Failed to describe route tables"

echo "Internet gateway:"
aws ec2 describe-internet-gateways --filters "Name=attachment.vpc-id,Values=
$VPC_ID" || handle_error "Failed to describe Internet Gateway"

echo "NAT gateway:"
aws ec2 describe-nat-gateways --filter "Name=vpc-id,Values=$VPC_ID" ||
 handle_error "Failed to describe NAT Gateway"

echo "Security groups:"
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$VPC_ID" ||
 handle_error "Failed to describe security groups"

echo ""
# Summary of created resources
echo "VPC creation completed successfully!"
echo "Summary of created resources:"
echo "- VPC: $VPC_ID"
echo "- Public Subnet in $AZ1: $PUBLIC_SUBNET_AZ1"
echo "- Public Subnet in $AZ2: $PUBLIC_SUBNET_AZ2"
echo "- Private Subnet in $AZ1: $PRIVATE_SUBNET_AZ1"
echo "- Private Subnet in $AZ2: $PRIVATE_SUBNET_AZ2"
echo "- Internet Gateway: $IGW_ID"
echo "- Public Route Table: $PUBLIC_RT"
echo "- Private Route Table: $PRIVATE_RT"
echo "- Elastic IP: $EIP_ALLOC"
echo "- NAT Gateway: $NAT_GW"
echo "- Web Servers Security Group: $WEB_SG"
echo "- Database Servers Security Group: $DB_SG"

# Deploy EC2 instances
echo ""
echo "Deploying EC2 instances..."

# Create key pair for SSH access
KEY_NAME="vpc-tutorial-key-$(date +%s)"
echo "Creating key pair $KEY_NAME..."
```

```
aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-specifications
 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'KeyMaterial' --output text >
 "${KEY_NAME}.pem" || handle_error "Failed to create key pair"
chmod 400 "${KEY_NAME}.pem"
echo "Key pair saved to ${KEY_NAME}.pem"
CREATED_RESOURCES+=("KEY_PAIR:$KEY_NAME")

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" --output text) ||
 handle_error "Failed to get AMI"
echo "Using AMI: $AMI_ID"

# Launch web server in public subnet
echo "Launching web server in public subnet..."
WEB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$WEB_SG" \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --associate-public-ip-address \
  --user-data '#!/bin/bash
    yum update -y
    yum install -y httpd
    systemctl start httpd
    systemctl enable httpd
    echo "<h1>Hello from $(hostname -f) in the public subnet</h1>" > /var/www/
html/index.html' \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=WebServer},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Instances[0].InstanceId' \
  --output text) || handle_error "Failed to launch web server"
echo "Web server instance created with ID: $WEB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$WEB_INSTANCE")

# Wait for web server to be running
echo "Waiting for web server to be running..."
aws ec2 wait instance-running --instance-ids "$WEB_INSTANCE"
```

```
# Get web server public IP
WEB_PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$WEB_INSTANCE" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' --output text)
echo "Web server public IP: $WEB_PUBLIC_IP"
echo "You can access the web server at: http://$WEB_PUBLIC_IP"

# Launch database server in private subnet
echo "Launching database server in private subnet..."
DB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$DB_SG" \
  --subnet-id "$PRIVATE_SUBNET_AZ1" \
  --user-data '#!/bin/bash
    yum update -y
    yum install -y mariadb-server
    systemctl start mariadb
    systemctl enable mariadb' \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=DBServer},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Instances[0].InstanceId' \
  --output text) || handle_error "Failed to launch database server"
echo "Database server instance created with ID: $DB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$DB_INSTANCE")

# Wait for database server to be running
echo "Waiting for database server to be running..."
aws ec2 wait instance-running --instance-ids "$DB_INSTANCE"

# Get database server private IP
DB_PRIVATE_IP=$(aws ec2 describe-instances --instance-ids "$DB_INSTANCE" \
  --query 'Reservations[0].Instances[0].PrivateIpAddress' --output text)
echo "Database server private IP: $DB_PRIVATE_IP"

echo "EC2 instances deployed successfully!"
echo "- Web Server (Public): $WEB_INSTANCE ($WEB_PUBLIC_IP)"
echo "- Database Server (Private): $DB_INSTANCE ($DB_PRIVATE_IP)"
echo ""
echo "Note: To connect to the web server: ssh -i ${KEY_NAME}.pem ec2-user@
$WEB_PUBLIC_IP"
```

```
echo "To connect to the database server, you must first connect to the web
 server, then use it as a bastion host."
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  echo "Cleaning up resources..."
  cleanup_resources
  echo "All resources have been cleaned up."
else
  echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateInternetGateway
  - CreateKeyPair
  - CreateNatGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - DeleteInternetGateway
  - DeleteKeyPair
  - DeleteNatGateway
  - DeleteRouteTable
  - DeleteSecurityGroup

- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeImages
- DescribeInstances
- DescribeInternetGateways
- DescribeNatGateways
- DescribeRouteTables
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachInternetGateway
- DisassociateRouteTable
- ModifySubnetAttribute
- ModifyVpcAttribute
- ReleaseAddress
- RunInstances
- TerminateInstances

Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um cluster.
- Criar uma definição de tarefa.
- Criar o serviço.
- Fazer a limpeza.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"
```

```
    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "=========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "=========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
```

```
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
```

```
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
```

```
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
 then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME'" "Delete ECS service"
            else
```

```
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
" ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
```

```
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}
```

```
# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
```

```
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
```

```
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
```

```
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
```

```
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"
```

```
# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
```

```
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices
  - DescribeSubnets

- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
```

```
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
```

```
    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
```

```
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
$TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
```

```
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
file for details."
    fi
```

```
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
```

```
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
```

```
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}
```

```
# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role
```

```
        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
```

```
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
```

```
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
```

```
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."
```

```
    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
```

```
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10
```

```
        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "==========================================="
    echo "Resources created:"
```

```
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup

- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Criação de uma instância de banco de dados relacional

O exemplo de código a seguir mostra como:

- Configurar componentes de rede.
- Criar um grupo de sub-redes de banco de dados
- Criar uma instância de banco de dados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/
latest/UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
```

```
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier
 "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ];
 then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ];
 then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
```

```
if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found
 ${#SUBNET_IDS[@]} subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP
 ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
```

```
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
```

```
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id
 "$SECRET_NAME" --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -
d'"' -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -
d'"' -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi
```

```
echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."
```

```
    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue

- Wait

Criação e gerenciamento de uma malha de rede de serviços

O exemplo de código a seguir mostra como:

- Criar uma rede de serviços.
- Criar um serviço.
- Listar as VPCs disponíveis.
- Listar grupos de segurança para a VPC selecionada.
- Listar associações de serviço.
- Listar associações de VPC.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC Lattice Service Network Tutorial Script
# This script demonstrates how to create and manage a VPC Lattice service network

set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="vpc-lattice-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting VPC Lattice tutorial script at $(date)" > "$LOG_FILE"

# Function to log commands and their output
```

```
log_command() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    if [ "$1" -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        exit "$1"
    fi
}

# Function to validate AWS CLI is available
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" | tee -a
 "$LOG_FILE"
        exit 1
    fi
}

# Function to validate input parameters
validate_input() {
    local input="$1"
    local param_name="$2"

    if [[ -z "$input" ]]; then
        echo "ERROR: $param_name is empty" | tee -a "$LOG_FILE"
        return 1
    fi

    # Validate against common injection patterns
    if [[ "$input" =~ [\;\$\`\|\&\<\>\(\)\{\}] ]]; then
        echo "ERROR: $param_name contains invalid characters" | tee -a
 "$LOG_FILE"
        return 1
    fi

    return 0
}
```

```
# Function to wait for a resource to be in the desired state
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
    local desired_status="$3"
    local command="$4"
    local max_attempts=30
    local attempt=1
    local status=""

    validate_input "$resource_type" "resource_type" || return 1
    validate_input "$resource_id" "resource_id" || return 1
    validate_input "$desired_status" "desired_status" || return 1

    echo "Waiting for $resource_type $resource_id to be in state
 $desired_status..." | tee -a "$LOG_FILE"

    while [ "$attempt" -le "$max_attempts" ]; do
        echo "Attempt $attempt of $max_attempts..." >> "$LOG_FILE"

        # Run the command to get the status and capture the output
        status_output=$(eval "$command" 2>&1) || true
        echo "$status_output" >> "$LOG_FILE"

        # For service networks, they do not have a status field in the output
        # We'll consider them active if we can retrieve them
        if [[ "$resource_type" == "Service Network" ]]; then
            if [[ "$status_output" == *"$resource_id"* ]]; then
                echo "$resource_type $resource_id is now active" | tee -a
 "$LOG_FILE"
                return 0
            fi
        else
            # For other resources, extract the status field
            status=$(echo "$status_output" | grep -i "status" | awk -F'"' '{print
 $4}' || true)
            echo "Current status: $status" >> "$LOG_FILE"

            if [[ "$status" == "$desired_status" ]]; then
                echo "$resource_type $resource_id is now in state
 $desired_status" | tee -a "$LOG_FILE"
                return 0
            elif [[ "$status" == *"FAIL"* ]]; then
```

```
                echo "ERROR: $resource_type $resource_id failed to reach desired
 state. Current status: $status" | tee -a "$LOG_FILE"
                return 1
            fi
        fi

        echo "Waiting for status change... (attempt $attempt/$max_attempts)" >>
 "$LOG_FILE"
        sleep 10
        ((attempt++))
    done

    echo "ERROR: Timed out waiting for $resource_type $resource_id to reach state
 $desired_status" | tee -a "$LOG_FILE"
    return 1
}

# Cleanup function for trap
cleanup() {
    local exit_code=$?
    echo "Script interrupted or failed. Cleaning up..." | tee -a "$LOG_FILE"
    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Check prerequisites
check_aws_cli

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
SERVICE_NETWORK_NAME="lattice-network-${RANDOM_ID}"
SERVICE_NAME="lattice-service-${RANDOM_ID}"

# Store created resources for cleanup
declare -a CREATED_RESOURCES

echo "=== VPC Lattice Service Network Tutorial ===" | tee -a "$LOG_FILE"
echo "Random ID for this session: ${RANDOM_ID}" | tee -a "$LOG_FILE"

# Step 1: Create a VPC Lattice service network
echo -e "\n=== Step 1: Creating a VPC Lattice service network ===" | tee -a
 "$LOG_FILE"
echo "Creating service network: $SERVICE_NETWORK_NAME" | tee -a "$LOG_FILE"
```

```
SERVICE_NETWORK_OUTPUT=$(log_command "aws vpc-lattice create-service-
network --name '$SERVICE_NETWORK_NAME' --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-vpc-lattice-gs --output json")
check_error $?

# Extract the service network ID using jq for safety
SERVICE_NETWORK_ID=$(echo "$SERVICE_NETWORK_OUTPUT" | jq -r '.id // empty' 2>/
dev/null || true)
if [ -z "$SERVICE_NETWORK_ID" ]; then
    echo "ERROR: Failed to extract service network ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_NETWORK_ID" "SERVICE_NETWORK_ID" || exit 1

echo "Service network created with ID: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Network: $SERVICE_NETWORK_ID")

# Wait for the service network to be active
wait_for_resource "Service Network" "$SERVICE_NETWORK_ID" "ACTIVE" "aws vpc-
lattice get-service-network --service-network-identifier '$SERVICE_NETWORK_ID' --
output json"
check_error $?

# Step 2: Create a VPC Lattice service
echo -e "\n=== Step 2: Creating a VPC Lattice service ===" | tee -a "$LOG_FILE"
echo "Creating service: $SERVICE_NAME" | tee -a "$LOG_FILE"

SERVICE_OUTPUT=$(log_command "aws vpc-lattice create-service --name
 '$SERVICE_NAME' --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-
vpc-lattice-gs --output json")
check_error $?

# Extract the service ID using jq for safety
SERVICE_ID=$(echo "$SERVICE_OUTPUT" | jq -r '.id // empty' 2>/dev/null || true)
if [ -z "$SERVICE_ID" ]; then
    echo "ERROR: Failed to extract service ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ID" "SERVICE_ID" || exit 1

echo "Service created with ID: $SERVICE_ID" | tee -a "$LOG_FILE"
```

```
CREATED_RESOURCES+=("Service: $SERVICE_ID")

# Wait for the service to be active
wait_for_resource "Service" "$SERVICE_ID" "ACTIVE" "aws vpc-lattice get-service
 --service-identifier '$SERVICE_ID' --output json"
check_error $?

# Step 3: Associate the service with the service network
echo -e "\n=== Step 3: Associating service with service network ===" | tee -a
 "$LOG_FILE"

SERVICE_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
service-association --service-identifier '$SERVICE_ID' --service-network-
identifier '$SERVICE_NETWORK_ID' --output json")
check_error $?

# Extract the service association ID using jq for safety
SERVICE_ASSOC_ID=$(echo "$SERVICE_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
if [ -z "$SERVICE_ASSOC_ID" ]; then
    echo "ERROR: Failed to extract service association ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ASSOC_ID" "SERVICE_ASSOC_ID" || exit 1

echo "Service association created with ID: $SERVICE_ASSOC_ID" | tee -a
 "$LOG_FILE"
CREATED_RESOURCES+=("Service Association: $SERVICE_ASSOC_ID")

# Wait for the service association to be active
wait_for_resource "Service Association" "$SERVICE_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-service-association --service-network-service-
association-identifier '$SERVICE_ASSOC_ID' --output json"
check_error $?

# Step 4: List available VPCs to associate with the service network
echo -e "\n=== Step 4: Listing available VPCs ===" | tee -a "$LOG_FILE"

VPC_LIST=$(log_command "aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId,Tags[?Key==
\`Name\`].Value|[0]]' --output text")
check_error $?

echo "Available VPCs:" | tee -a "$LOG_FILE"
```

```
echo "$VPC_LIST" | tee -a "$LOG_FILE"

# Step 5: Auto-select first available VPC
echo -e "\n=== Step 5: Associate a VPC with the service network ===" | tee -a
 "$LOG_FILE"

VPC_ID=$(echo "$VPC_LIST" | head -n 1 | awk '{print $1}')

if [ -z "$VPC_ID" ]; then
    echo "WARNING: No VPC ID found" | tee -a "$LOG_FILE"
    echo "Skipping VPC association step" | tee -a "$LOG_FILE"
else
    validate_input "$VPC_ID" "VPC_ID" || {
        echo "ERROR: VPC_ID validation failed"
        exit 1
    }

    echo "Auto-selected VPC: $VPC_ID" | tee -a "$LOG_FILE"

    # Step 6: List security groups for the selected VPC
    echo -e "\n=== Step 6: Listing security groups for VPC $VPC_ID ===" | tee -a
 "$LOG_FILE"

    SG_LIST=$(log_command "aws ec2 describe-security-groups --filters Name=vpc-
id,Values='$VPC_ID' --query 'SecurityGroups[*].[GroupId,GroupName]' --output
 text")
    check_error $?

    echo "Available Security Groups for VPC $VPC_ID:" | tee -a "$LOG_FILE"
    echo "$SG_LIST" | tee -a "$LOG_FILE"

    # Step 7: Auto-select first available security group
    echo -e "\n=== Step 7: Select a security group for the VPC association ===" |
 tee -a "$LOG_FILE"

    SG_ID=$(echo "$SG_LIST" | head -n 1 | awk '{print $1}')

    if [ -z "$SG_ID" ]; then
        echo "WARNING: No Security Group ID found" | tee -a "$LOG_FILE"
        echo "Skipping VPC association step" | tee -a "$LOG_FILE"
    else
        validate_input "$SG_ID" "SG_ID" || {
            echo "ERROR: SG_ID validation failed"
            exit 1
```

```
        }

        echo "Auto-selected Security Group: $SG_ID" | tee -a "$LOG_FILE"

        # Step 8: Associate the VPC with the service network
        echo -e "\n=== Step 8: Associating VPC with service network ===" | tee -a
 "$LOG_FILE"

        VPC_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
vpc-association --vpc-identifier '$VPC_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --security-group-ids '$SG_ID' --output json")
        check_error $?

        # Extract the VPC association ID using jq for safety
        VPC_ASSOC_ID=$(echo "$VPC_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
        if [ -z "$VPC_ASSOC_ID" ]; then
            echo "ERROR: Failed to extract VPC association ID" | tee -a
 "$LOG_FILE"
        else
            validate_input "$VPC_ASSOC_ID" "VPC_ASSOC_ID" || exit 1

            echo "VPC association created with ID: $VPC_ASSOC_ID" | tee -a
 "$LOG_FILE"
            CREATED_RESOURCES+=("VPC Association: $VPC_ASSOC_ID")

            # Wait for the VPC association to be active
            wait_for_resource "VPC Association" "$VPC_ASSOC_ID" "ACTIVE"
 "aws vpc-lattice get-service-network-vpc-association --service-network-vpc-
association-identifier '$VPC_ASSOC_ID' --output json"
            check_error $?
        fi
    fi
fi

# Step 9: Display information about the created resources
echo -e "\n=== Step 9: Displaying information about created resources ===" | tee
 -a "$LOG_FILE"

echo "Service Network Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json"

echo "Service Details:" | tee -a "$LOG_FILE"
```

```
log_command "aws vpc-lattice get-service --service-identifier '$SERVICE_ID' --
output json"

echo "Service Network Service Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-service-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

echo "Service Network VPC Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-vpc-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

# Step 10: Cleanup - Auto-confirm
echo -e "\n=== Step 10: Resource Cleanup ===" | tee -a "$LOG_FILE"
echo "Resources created in this tutorial:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
done

echo "Starting cleanup process..." | tee -a "$LOG_FILE"

# Delete resources in reverse order

# Delete VPC association if it was created
if [[ -n "${VPC_ASSOC_ID:-}" ]]; then
    echo "Deleting VPC association: $VPC_ASSOC_ID" | tee -a "$LOG_FILE"
    log_command "aws vpc-lattice delete-service-network-vpc-association --
service-network-vpc-association-identifier '$VPC_ASSOC_ID'" || true

    # Wait for the VPC association to be deleted
    echo "Waiting for VPC association to be deleted..." | tee -a "$LOG_FILE"
    sleep 30
fi

# Delete service association
echo "Deleting service association: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network-service-association --
service-network-service-association-identifier '$SERVICE_ASSOC_ID'" || true

# Wait for the service association to be deleted
echo "Waiting for service association to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service
echo "Deleting service: $SERVICE_ID" | tee -a "$LOG_FILE"
```

```
log_command "aws vpc-lattice delete-service --service-identifier '$SERVICE_ID'"
 || true

# Wait for the service to be deleted
echo "Waiting for service to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service network
echo "Deleting service network: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID'" || true

echo "Cleanup completed successfully!" | tee -a "$LOG_FILE"

echo -e "\n=== Tutorial completed! ===" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateService
  - CreateServiceNetwork
  - CreateServiceNetworkServiceAssociation
  - CreateServiceNetworkVpcAssociation
  - DeleteService
  - DeleteServiceNetwork
  - DeleteServiceNetworkServiceAssociation
  - DeleteServiceNetworkVpcAssociation
  - DescribeSecurityGroups
  - DescribeVpcs
  - GetService
  - GetServiceNetwork
  - GetServiceNetworkServiceAssociation
  - GetServiceNetworkVpcAssociation
  - ListServiceNetworkServiceAssociations
  - ListServiceNetworkVpcAssociations

Criação e gerenciamento de volumes de armazenamento em bloco

O exemplo de código a seguir mostra como:

- Criar um volume do EBS.
- Verificar o status do volume.
- Criar uma instância do EC2 (opcional).
- Anexar um volume a uma instância do EC2.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create and manage Amazon EBS volumes
# This script demonstrates how to create an EBS volume and attach it to an EC2
 instance
# It can also create a test EC2 instance if needed

# Set up logging
LOG_FILE="ebs-volume-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting EBS volume creation script at $(date)"
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "$VOLUME_ID" ]; then
        echo "- EBS Volume: $VOLUME_ID"
```

```
    fi
    if [ -n "$INSTANCE_ID" ]; then
        echo "- EC2 Instance: $INSTANCE_ID"
    fi
    if [ -n "$SG_ID" ]; then
        echo "- Security Group: $SG_ID"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Do you want to clean up created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
    else
        echo "Resources were not cleaned up. You will need to delete them
 manually."
    fi

    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "$VOLUME_ID" ] && [ "$ATTACHED" = true ]; then
        echo "Detaching volume $VOLUME_ID..."
        aws ec2 detach-volume --volume-id "$VOLUME_ID"

        # Wait for volume to be detached
        echo "Waiting for volume to be detached..."
        aws ec2 wait volume-available --volume-ids "$VOLUME_ID"
    fi

    if [ -n "$VOLUME_ID" ]; then
        echo "Deleting volume $VOLUME_ID..."
        aws ec2 delete-volume --volume-id "$VOLUME_ID"
    fi

    if [ -n "$INSTANCE_ID" ] && [ "$CREATED_INSTANCE" = true ]; then
```

```
        echo "Terminating instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
    fi

    # Clean up security group if created
    if [ -n "$SG_ID" ] && [ "$CREATED_INSTANCE" = true ]; then
        echo "Deleting security group $SG_ID..."
        # Wait a bit for instance termination to complete
        sleep 10
        aws ec2 delete-security-group --group-id "$SG_ID" 2>/dev/null || echo
 "Security group may have dependencies, delete manually if needed"
    fi

    echo "Cleanup completed."
}

# Function to get available instance type
get_available_instance_type() {
    local region=$1

    # Try instance types in order of preference (cheapest first)
    local instance_types=("t3.nano" "t3.micro" "t2.micro" "t2.nano")

    for instance_type in "${instance_types[@]}"; do
        local available=$(aws ec2 describe-instance-type-offerings \
            --region "$region" \
            --filters "Name=instance-type,Values=$instance_type" \
            --query "length(InstanceTypeOfferings)" \
            --output text)

        if [ "$available" -gt 0 ]; then
            echo "$instance_type"
            return 0
        fi
    done

    # If none of the preferred types are available, get any available type
    local fallback_type=$(aws ec2 describe-instance-type-offerings \
        --region "$region" \
        --query "InstanceTypeOfferings[0].InstanceType" \
        --output text)
```

```
    if [ "$fallback_type" != "None" ] && [ -n "$fallback_type" ]; then
        echo "$fallback_type"
        return 0
    fi

    return 1
}

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 "AvailabilityZones[0].RegionName" --output text)
fi

echo "Using region: $REGION"

# Get available Availability Zones
echo "Retrieving available Availability Zones..."
AZ=$(aws ec2 describe-availability-zones --filters "Name=state,Values=available"
 --query "AvailabilityZones[0].ZoneName" --output text)
if [ -z "$AZ" ]; then
    handle_error "Failed to retrieve Availability Zones"
fi
echo "Using Availability Zone: $AZ"

# Create a gp3 volume
echo "Creating a 10 GiB gp3 volume in $AZ..."
VOLUME_ID=$(aws ec2 create-volume \
    --volume-type gp3 \
    --size 10 \
    --availability-zone "$AZ" \
    --tag-specifications
 'ResourceType=volume,Tags=[{Key=Name,Value=EBSTutorialVolume},
{Key=Purpose,Value=Tutorial},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-gs-volumes}]' \
    --query 'VolumeId' \
    --output text)

if [ -z "$VOLUME_ID" ]; then
    handle_error "Failed to create EBS volume"
fi

echo "Volume created with ID: $VOLUME_ID"
```

```
# Wait for volume to become available
echo "Waiting for volume to become available..."
aws ec2 wait volume-available --volume-ids "$VOLUME_ID"
if [ $? -ne 0 ]; then
    handle_error "Volume did not become available"
fi

# Check volume details
echo "Retrieving volume details..."
aws ec2 describe-volumes --volume-ids "$VOLUME_ID"

# Ask if user wants to attach the volume to an instance
echo ""
echo "==========================================="
echo "VOLUME ATTACHMENT"
echo "==========================================="
echo "Do you want to attach this volume to an EC2 instance? (y/n): "
read -r ATTACH_CHOICE

ATTACHED=false
CREATED_INSTANCE=false
INSTANCE_ID=""
SG_ID=""

if [[ "$ATTACH_CHOICE" =~ ^[Yy]$ ]]; then
    # List available instances in the same AZ
    echo "Retrieving EC2 instances in $AZ..."
    INSTANCES_COUNT=$(aws ec2 describe-instances \
        --filters "Name=availability-zone,Values=$AZ" "Name=instance-state-
name,Values=running" \
        --query "length(Reservations[].Instances[])" \
        --output text)

    # Check if there are any running instances in the AZ
    if [ "$INSTANCES_COUNT" -eq 0 ]; then
        echo "No running instances found in $AZ."
        echo ""
        echo "Would you like to create a test EC2 instance? (y/n): "
        read -r CREATE_INSTANCE_CHOICE

        if [[ "$CREATE_INSTANCE_CHOICE" =~ ^[Yy]$ ]]; then
            # Get available instance type
            echo "Finding available instance type for region $REGION..."
```

```
            INSTANCE_TYPE=$(get_available_instance_type "$REGION")
            if [ $? -ne 0 ] || [ -z "$INSTANCE_TYPE" ]; then
                handle_error "No suitable instance type found in region $REGION"
            fi
            echo "Using instance type: $INSTANCE_TYPE"

            # Get the latest Amazon Linux 2 AMI
            echo "Finding the latest Amazon Linux 2 AMI..."
            AMI_ID=$(aws ec2 describe-images \
                --owners amazon \
                --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
"Name=state,Values=available" \
                --query "sort_by(Images, &CreationDate)[-1].ImageId" \
                --output text)

            if [ -z "$AMI_ID" ]; then
                handle_error "Failed to find a suitable AMI"
            fi

            echo "Using AMI: $AMI_ID"

            # Check if a default VPC exists
            DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
                --filters "Name=isDefault,Values=true" \
                --query "Vpcs[0].VpcId" \
                --output text)

            if [ "$DEFAULT_VPC_ID" = "None" ] || [ -z "$DEFAULT_VPC_ID" ]; then
                handle_error "No default VPC found. Please create a VPC and
subnet before running this script."
            fi

            # Get a subnet in the selected AZ
            SUBNET_ID=$(aws ec2 describe-subnets \
                --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID"
"Name=availability-zone,Values=$AZ" \
                --query "Subnets[0].SubnetId" \
                --output text)

            if [ -z "$SUBNET_ID" ] || [ "$SUBNET_ID" = "None" ]; then
                handle_error "No subnet found in $AZ. Please create a subnet
before running this script."
            fi
```

```
            echo "Using subnet: $SUBNET_ID"

            # Create a security group that allows SSH
            SG_NAME="EBSTutorialSG-$(date +%s)"
            SG_ID=$(aws ec2 create-security-group \
                --group-name "$SG_NAME" \
                --description "Security group for EBS tutorial" \
                --vpc-id "$DEFAULT_VPC_ID" \
                --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=ebs-gs-volumes}]' \
                --query "GroupId" \
                --output text)

            if [ -z "$SG_ID" ]; then
                handle_error "Failed to create security group"
            fi

            echo "Created security group: $SG_ID"

            # Add a rule to allow SSH
            aws ec2 authorize-security-group-ingress \
                --group-id "$SG_ID" \
                --protocol tcp \
                --port 22 \
                --cidr 0.0.0.0/0

            echo "Added SSH rule to security group"

            # Create the instance
            echo "Creating EC2 instance in $AZ with instance type
 $INSTANCE_TYPE..."
            INSTANCE_ID=$(aws ec2 run-instances \
                --image-id "$AMI_ID" \
                --instance-type "$INSTANCE_TYPE" \
                --subnet-id "$SUBNET_ID" \
                --security-group-ids "$SG_ID" \
                --tag-specifications
 'ResourceType=instance,Tags=[{Key=Name,Value=EBSTutorialInstance},
{Key=Purpose,Value=Tutorial},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-gs-volumes}]' \
                --query "Instances[0].InstanceId" \
                --output text)

            if [ -z "$INSTANCE_ID" ]; then
```

```
                handle_error "Failed to create EC2 instance"
            fi

            CREATED_INSTANCE=true
            echo "Instance created with ID: $INSTANCE_ID"

            # Wait for the instance to be running
            echo "Waiting for instance to be running..."
            aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

            # Wait a bit more for the instance to initialize
            echo "Waiting for instance initialization (30 seconds)..."
            sleep 30
        else
            echo "Skipping instance creation and volume attachment."
            INSTANCE_ID=""
        fi
    else
        # Display available instances
        echo "Available instances in $AZ:"
        aws ec2 describe-instances \
            --filters "Name=availability-zone,Values=$AZ" "Name=instance-state-
name,Values=running" \
            --query "Reservations[*].Instances[*].[InstanceId,Tags[?
Key=='Name'].Value|[0],InstanceType]" \
            --output table

        # Ask for instance ID
        echo ""
        echo "Enter the instance ID to attach the volume to (or press Enter to
 skip): "
        read -r INSTANCE_ID
    fi

    if [ -n "$INSTANCE_ID" ]; then
        # Attach volume to the instance
        echo "Attaching volume $VOLUME_ID to instance $INSTANCE_ID..."
        ATTACH_RESULT=$(aws ec2 attach-volume \
            --volume-id "$VOLUME_ID" \
            --instance-id "$INSTANCE_ID" \
            --device "/dev/sdf" \
            --query 'State' \
            --output text)
```

```
        if [ $? -ne 0 ] || [ -z "$ATTACH_RESULT" ]; then
            handle_error "Failed to attach volume to instance"
        fi

        ATTACHED=true
        echo "Volume attached successfully. Device: /dev/sdf"

        # Verify attachment
        echo "Verifying attachment..."
        aws ec2 describe-volumes \
            --volume-ids "$VOLUME_ID" \
            --query "Volumes[0].Attachments"
    else
        echo "Skipping volume attachment."
    fi
else
    echo "Skipping volume attachment."
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "Created resources:"
echo "- EBS Volume: $VOLUME_ID"
if [ "$ATTACHED" = true ]; then
    echo "  - Attached to: $INSTANCE_ID as /dev/sdf"
fi
if [ "$CREATED_INSTANCE" = true ]; then
    echo "- EC2 Instance: $INSTANCE_ID (type: $INSTANCE_TYPE)"
fi
if [ -n "$SG_ID" ]; then
    echo "- Security Group: $SG_ID"
fi

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
else
    echo ""
    echo "Resources were not cleaned up. You can manually delete them later."
    if [ -n "$VOLUME_ID" ]; then
        if [ "$ATTACHED" = true ]; then
            echo "To detach the volume:"
            echo "  aws ec2 detach-volume --volume-id $VOLUME_ID"
        fi
        echo "To delete the volume:"
        echo "  aws ec2 delete-volume --volume-id $VOLUME_ID"
    fi
    if [ "$CREATED_INSTANCE" = true ]; then
        echo "To terminate the instance:"
        echo "  aws ec2 terminate-instances --instance-ids $INSTANCE_ID"
    fi
    if [ -n "$SG_ID" ]; then
        echo "To delete the security group (after instance termination):"
        echo "  aws ec2 delete-security-group --group-id $SG_ID"
    fi
fi

echo ""
echo "Script completed at $(date)"
echo "================================================"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachVolume
  - AuthorizeSecurityGroupIngress
  - CreateSecurityGroup
  - CreateVolume
  - DeleteSecurityGroup
  - DeleteVolume
  - DescribeAvailabilityZones
  - DescribeImages
  - DescribeInstanceTypeOfferings
  - DescribeInstances
  - DescribeSubnets

- DescribeVolumes
- DescribeVpcs
- DetachVolume
- RunInstances
- TerminateInstances
- Wait

Comece com conexões de rede dedicadas usando a CLI

O exemplo de código a seguir mostra como:

- Use ec2 CreateVpnGateway
- Use ec2 DeleteVpnGateway
- Use ec2 DescribeVpnGateways
- Use a conexão direta CreateConnection
- Use a conexão direta CreatePrivateVirtualInterface
- Use a conexão direta DeleteConnection

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Direct Connect Connection Management Script - Version 6
# This script demonstrates how to create and manage AWS Direct Connect
 connections using the AWS CLI
# This version includes fixes for user input handling and better error reporting

# Set up logging
```

```
LOG_FILE="directconnect-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting AWS Direct Connect script v6"

# Function to check for errors in command output
check_error() {
    local output=$1
    local command=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $command"
        echo "Output: $output"
        cleanup_resources
        exit 1
    fi
}

# Function to wait for VGW to be available
wait_for_vgw() {
    local vgw_id=$1
    local max_attempts=30
    local attempt=1

    echo "Waiting for virtual private gateway $vgw_id to become available..."

    while [ $attempt -le $max_attempts ]; do
        VGW_STATE=$(aws ec2 describe-vpn-gateways --vpn-gateway-ids "$vgw_id" --
query 'VpnGateways[0].State' --output text)

        if [ "$VGW_STATE" == "available" ]; then
            echo "Virtual private gateway is now available"
            return 0
        elif [ "$VGW_STATE" == "failed" ]; then
            echo "Virtual private gateway failed to become available"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: VGW state is $VGW_STATE, waiting 10
 seconds..."
        sleep 10
        attempt=$((attempt + 1))
    done
```

```
    echo "Timeout waiting for VGW to become available"
    return 1
}

# Function to wait for connection to be available
wait_for_connection() {
    local connection_id=$1
    local max_attempts=60
    local attempt=1

    echo "Waiting for connection $connection_id to become available..."
    echo "Note: This can take 30+ minutes in production as AWS provisions the
 physical connection"

    while [ $attempt -le $max_attempts ]; do
        CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$connection_id" --query 'connections[0].connectionState' --output text)

        if [ "$CONNECTION_STATE" == "available" ]; then
            echo "Connection is now available"
            return 0
        elif [ "$CONNECTION_STATE" == "rejected" ] || [ "$CONNECTION_STATE" ==
 "deleted" ]; then
            echo "Connection failed with state: $CONNECTION_STATE"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: Connection state is
 $CONNECTION_STATE, waiting 30 seconds..."
        sleep 30
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for connection to become available"
    return 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete virtual interfaces if they exist
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "Deleting private virtual interface: $PRIVATE_VIF_ID"
```

```
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PRIVATE_VIF_ID"
    fi

    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "Deleting public virtual interface: $PUBLIC_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PUBLIC_VIF_ID"
    fi

    # Delete connection if it exists
    if [ -n "$CONNECTION_ID" ]; then
        echo "Deleting connection: $CONNECTION_ID"
        aws directconnect delete-connection --connection-id "$CONNECTION_ID"
    fi

    # Delete VGW if it exists
    if [ -n "$VGW_ID" ]; then
        echo "Deleting virtual private gateway: $VGW_ID"
        aws ec2 delete-vpn-gateway --vpn-gateway-id "$VGW_ID"
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 6)
CONNECTION_NAME="DxConn-${RANDOM_ID}"

# Step 1: List available Direct Connect locations
echo "Listing available Direct Connect locations..."
LOCATIONS_OUTPUT=$(aws directconnect describe-locations)
check_error "$LOCATIONS_OUTPUT" "describe-locations"
echo "$LOCATIONS_OUTPUT"

# Extract the first location code for demonstration purposes
LOCATION_CODE=$(aws directconnect describe-locations --query
 'locations[0].locationCode' --output text)
if [ -z "$LOCATION_CODE" ] || [ "$LOCATION_CODE" == "None" ]; then
    echo "Error: Could not extract location code from the output."
    exit 1
fi
echo "Using location: $LOCATION_CODE"

# Step 2: Create a dedicated connection
```

```
echo "Creating a dedicated connection at location $LOCATION_CODE with bandwidth
 1Gbps..."
CONNECTION_OUTPUT=$(aws directconnect create-connection \
    --location "$LOCATION_CODE" \
    --bandwidth "1Gbps" \
    --connection-name "$CONNECTION_NAME" \
    --tags key=project,value=doc-smith key=tutorial,value=aws-direct-connect-gs)
check_error "$CONNECTION_OUTPUT" "create-connection"
echo "$CONNECTION_OUTPUT"

# Extract connection ID directly from the output
CONNECTION_ID=$(echo "$CONNECTION_OUTPUT" | grep -o '"connectionId": "[^"]*' |
 cut -d'"' -f4)
if [ -z "$CONNECTION_ID" ]; then
    echo "Error: Could not extract connection ID from the output."
    exit 1
fi
echo "Connection created with ID: $CONNECTION_ID"

# Step 3: Describe the connection
echo "Retrieving connection details..."
DESCRIBE_OUTPUT=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID")
check_error "$DESCRIBE_OUTPUT" "describe-connections"
echo "$DESCRIBE_OUTPUT"

# Step 4: Update the connection name
NEW_CONNECTION_NAME="${CONNECTION_NAME}-updated"
echo "Updating connection name to $NEW_CONNECTION_NAME..."
UPDATE_OUTPUT=$(aws directconnect update-connection \
    --connection-id "$CONNECTION_ID" \
    --connection-name "$NEW_CONNECTION_NAME")
check_error "$UPDATE_OUTPUT" "update-connection"
echo "$UPDATE_OUTPUT"

# Step 5: Check if we can download the LOA-CFA
# Note: In a real scenario, the LOA-CFA might not be immediately available
echo "Attempting to download the LOA-CFA (this may not be available yet)..."
LOA_OUTPUT=$(aws directconnect describe-loa --connection-id "$CONNECTION_ID"
 2>&1)
if echo "$LOA_OUTPUT" | grep -i "error" > /dev/null; then
    echo "LOA-CFA not available yet. This is expected for newly created
 connections."
```

```
    echo "The LOA-CFA will be available once AWS begins provisioning your
 connection."
else
    LOA_CONTENT=$(echo "$LOA_OUTPUT" | grep -o '"loaContent": "[^"]*' | cut -d'"'
 -f4)
    echo "$LOA_CONTENT" | base64 --decode > "loa-cfa-${CONNECTION_ID}.pdf"
    echo "LOA-CFA downloaded to loa-cfa-${CONNECTION_ID}.pdf"
fi

# Step 6: Create a virtual private gateway (required for private virtual
 interface)
echo "Creating a virtual private gateway..."
VGW_OUTPUT=$(aws ec2 create-vpn-gateway --type ipsec.1 \
    --tag-specifications 'ResourceType=vpn-gateway,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=aws-direct-connect-gs}]')
check_error "$VGW_OUTPUT" "create-vpn-gateway"
echo "$VGW_OUTPUT"

# Extract VGW ID directly from the output
VGW_ID=$(echo "$VGW_OUTPUT" | grep -o '"VpnGatewayId": "[^"]*' | cut -d'"' -f4)
if [ -z "$VGW_ID" ]; then
    echo "Error: Could not extract VPN gateway ID from the output."
    exit 1
fi
echo "Virtual private gateway created with ID: $VGW_ID"

# Wait for VGW to become available
if ! wait_for_vgw "$VGW_ID"; then
    echo "Failed to wait for VGW to become available. Skipping virtual interface
 creation."
    VIF_CREATION_SKIPPED=true
else
    VIF_CREATION_SKIPPED=false
fi

# Step 7: Create a private virtual interface (only if VGW is available)
if [ "$VIF_CREATION_SKIPPED" = false ]; then
    echo "Creating a private virtual interface..."
    PRIVATE_VIF_OUTPUT=$(aws directconnect create-private-virtual-interface \
        --connection-id "$CONNECTION_ID" \
        --new-private-virtual-interface '{
            "virtualInterfaceName": "PrivateVIF-'"$RANDOM_ID"'",
            "vlan": 100,
            "asn": 65000,
```

```
            "authKey": "'"$RANDOM_ID"'key",
            "amazonAddress": "192.168.1.1/30",
            "customerAddress": "192.168.1.2/30",
            "addressFamily": "ipv4",
            "virtualGatewayId": "'"$VGW_ID"'"
        }' 2>&1)

    if echo "$PRIVATE_VIF_OUTPUT" | grep -i "error" > /dev/null; then
        echo "Could not create private virtual interface. This is expected if the
 connection is not yet available."
        echo "Error: $PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=""
    else
        echo "$PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=$(echo "$PRIVATE_VIF_OUTPUT" | grep -o
 '"virtualInterfaceId": "[^"]*' | cut -d'"' -f4)
        echo "Private virtual interface created with ID: $PRIVATE_VIF_ID"
    fi
else
    echo "Skipping private virtual interface creation due to VGW not being
 available"
    PRIVATE_VIF_ID=""
fi

# Step 8: Check connection state and provide guidance for public virtual
 interface
CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID" --query 'connections[0].connectionState' --output text)
echo "Current connection state: $CONNECTION_STATE"

if [ "$CONNECTION_STATE" != "available" ]; then
    echo ""
    echo "============================================"
    echo "CONNECTION NOT YET AVAILABLE"
    echo "============================================"
    echo "The connection is in '$CONNECTION_STATE' state."
    echo "In production, you would:"
    echo "1. Wait for AWS to provision the connection (can take 30+ minutes)"
    echo "2. Download the LOA-CFA when available"
    echo "3. Provide the LOA-CFA to your network provider for cross-connect"
    echo "4. Create virtual interfaces once connection is 'available'"
    echo ""

    # Ask if user wants to wait for connection to become available
```

```
    echo ""
    echo "============================================"
    echo "CONNECTION WAIT CONFIRMATION"
    echo "============================================"
    echo -n "Do you want to wait for the connection to become available? (y/n): "
    read -r WAIT_CHOICE

    if [[ "$WAIT_CHOICE" =~ ^[Yy]$ ]]; then
        if wait_for_connection "$CONNECTION_ID"; then
            echo "Connection is now available! You could now create virtual
 interfaces."
        else
            echo "Connection did not become available within the timeout period."
        fi
    else
        echo "Skipping wait for connection availability."
    fi
else
    echo "Connection is available! Virtual interfaces can be created."
fi

# Step 9: List all virtual interfaces
echo "Listing all virtual interfaces..."
VIF_LIST_OUTPUT=$(aws directconnect describe-virtual-interfaces)
check_error "$VIF_LIST_OUTPUT" "describe-virtual-interfaces"
echo "$VIF_LIST_OUTPUT"

# Step 10: Display important information about production usage
echo ""
echo "============================================"
echo "IMPORTANT PRODUCTION NOTES"
echo "============================================"
echo "1. Direct Connect connections take time to be provisioned by AWS"
echo "2. You cannot create virtual interfaces until the connection is
 'available'"
echo "3. For public virtual interfaces, you must own the public IP addresses"
echo "4. LOA-CFA (Letter of Authorization) is needed for cross-connect at the
 facility"
echo "5. This demo creates resources that incur costs (~\$300/month for 1Gbps)"
echo "6. Always test connectivity before putting into production"
echo ""

# Step 11: Ask user if they want to clean up resources
echo ""
```

```
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo -n "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually delete them later."
    echo ""
    echo "Created resources:"
    echo "- Connection ID: $CONNECTION_ID"
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "- Private Virtual Interface ID: $PRIVATE_VIF_ID"
    fi
    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "- Public Virtual Interface ID: $PUBLIC_VIF_ID"
    fi
    echo "- Virtual Private Gateway ID: $VGW_ID"
    echo ""
    echo "Manual cleanup commands:"
    echo "aws directconnect delete-connection --connection-id $CONNECTION_ID"
    echo "aws ec2 delete-vpn-gateway --vpn-gateway-id $VGW_ID"
    echo ""
    echo "Remember: Direct Connect resources incur ongoing costs!"
fi

echo "$(date): Script completed"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateConnection
  - CreatePrivateVirtualInterface
  - CreateVpnGateway
  - DeleteConnection
  - DeleteVirtualInterface
  - DeleteVpnGateway
  - DescribeConnections

- DescribeLoa
- DescribeLocations
- DescribeVirtualInterfaces
- DescribeVpnGateways
- UpdateConnection

Comece a comprar no mercado de software usando a CLI

O exemplo de código a seguir mostra como:

- Use ec2 AuthorizeSecurityGroupIngress
- Use ec2 CreateKeyPair
- Use ec2 CreateSecurityGroup

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Marketplace Buyer Getting Started Script
# This script demonstrates how to search for products in AWS Marketplace,
# launch an EC2 instance with a product AMI, and manage subscriptions.

set -euo pipefail

# Setup logging with secure permissions
LOG_FILE="marketplace-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "====================================================="
echo "AWS Marketplace Buyer Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. List available products in AWS Marketplace"
echo "2. Create resources needed to launch an EC2 instance"
echo "3. Launch an EC2 instance with an Amazon Linux 2 AMI"
echo "4. Show how to manage and terminate the instance"
echo "====================================================="
echo ""

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it first."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured. Please configure them
 first."
    exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
    echo "ERROR: jq is not installed. Please install jq for safe JSON parsing."
    exit 1
fi

# Function to safely extract JSON values using jq
extract_json_value() {
    local json=$1
    local query=$2

    echo "$json" | jq -r "$query" 2>/dev/null || {
        echo "ERROR: Failed to parse JSON with query: $query" >&2
        return 1
    }
}

# Function to validate AWS permissions
validate_aws_permissions() {
    echo "Validating AWS permissions..."
```

```
    local identity
    identity=$(aws sts get-caller-identity --output json)
    local account_id
    account_id=$(extract_json_value "$identity" '.Account') || return 1
    local arn
    arn=$(extract_json_value "$identity" '.Arn') || return 1

    echo "AWS Account ID: $account_id"
    echo "AWS Principal ARN: $arn"
    echo "Note: This script requires EC2 permissions for key pair, security
 group, and instance management."
    echo ""
}

# Function to clean up resources
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${INSTANCE_ID:-}" ]; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" > /dev/null 2>&1 || true

        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" 2>/dev/null || true
        echo "Instance terminated successfully."
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting before deleting security group..."
        sleep 5
        echo "Deleting security group: $SECURITY_GROUP_ID"
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID" --region
 "$AWS_REGION" > /dev/null 2>&1 || true
        echo "Security group deleted."
    fi

    if [ -n "${KEY_NAME:-}" ]; then
        echo "Deleting key pair: $KEY_NAME"
```

```
        aws ec2 delete-key-pair --key-name "$KEY_NAME" --region "$AWS_REGION" > /
dev/null 2>&1 || true

        # Remove the local key file if it exists with secure deletion
        if [ -f "${KEY_NAME}.pem" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "${KEY_NAME}.pem" 2>/dev/null || rm -f
 "${KEY_NAME}.pem"
            else
                rm -f "${KEY_NAME}.pem"
            fi
            echo "Local key file securely deleted."
        fi
    fi

    echo "Cleanup completed."
}

# Set trap to ensure cleanup on script exit
trap cleanup_resources EXIT

# Get the current AWS region
AWS_REGION=$(aws configure get region || echo "us-east-1")
if [ -z "$AWS_REGION" ] || [ "$AWS_REGION" = "None" ]; then
    AWS_REGION="us-east-1"
fi
echo "Using AWS Region: $AWS_REGION"
echo ""

# Validate permissions
validate_aws_permissions

# Generate random identifier for resource names using cryptographically secure
 method
RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
KEY_NAME="marketplace-key-${RANDOM_ID}"
SECURITY_GROUP_NAME="marketplace-sg-${RANDOM_ID}"

# Initialize variables to track created resources
INSTANCE_ID=""
SECURITY_GROUP_ID=""
AMI_ID=""

# Step 1: List available products in AWS Marketplace
```

```
echo "Listing available products in AWS Marketplace..."
echo "Note: In a real scenario, you would use marketplace-catalog commands to
 list and search for products."
echo "However, this requires specific permissions and product knowledge."
echo ""
echo "For this tutorial, we'll use a public Amazon Linux 2 AMI instead of an
 actual marketplace product."
echo "This is because subscribing to marketplace products requires accepting
 terms via the console."
echo ""

# Step 2: Create a key pair for SSH access
echo "Creating key pair: $KEY_NAME"
KEY_OUTPUT=$(aws ec2 create-key-pair \
  --key-name "$KEY_NAME" \
  --region "$AWS_REGION" \
  --query 'KeyMaterial' \
  --output text) || {
    echo "ERROR: Failed to create key pair" >&2
    exit 1
}

# Securely save the key with restricted permissions
if ! echo "$KEY_OUTPUT" > "${KEY_NAME}.pem" 2>/dev/null; then
    echo "ERROR: Failed to write key file ${KEY_NAME}.pem" >&2
    exit 1
fi
chmod 600 "${KEY_NAME}.pem" || {
    echo "ERROR: Failed to set permissions on key file" >&2
    rm -f "${KEY_NAME}.pem"
    exit 1
}
echo "Key pair created and saved to ${KEY_NAME}.pem with secure permissions
 (600)"

# Step 3: Create a security group
echo "Creating security group: $SECURITY_GROUP_NAME"
SG_OUTPUT=$(aws ec2 create-security-group \
  --group-name "$SECURITY_GROUP_NAME" \
  --description "Security group for AWS Marketplace tutorial" \
  --region "$AWS_REGION" \
  --output json) || {
    echo "ERROR: Failed to create security group" >&2
    exit 1
```

```
}

# Extract security group ID using jq for safe parsing
SECURITY_GROUP_ID=$(extract_json_value "$SG_OUTPUT" '.GroupId') || exit 1
if [ -z "$SECURITY_GROUP_ID" ] || [ "$SECURITY_GROUP_ID" = "null" ]; then
    echo "ERROR: Could not extract security group ID" >&2
    exit 1
fi
echo "Security group created with ID: $SECURITY_GROUP_ID"

# Add inbound rules for SSH and HTTP in parallel for better performance
echo "Configuring security group rules..."
{
    aws ec2 authorize-security-group-ingress \
      --group-id "$SECURITY_GROUP_ID" \
      --protocol tcp \
      --port 22 \
      --cidr 0.0.0.0/0 \
      --region "$AWS_REGION" > /dev/null 2>&1
} &
SSH_PID=$!

{
    aws ec2 authorize-security-group-ingress \
      --group-id "$SECURITY_GROUP_ID" \
      --protocol tcp \
      --port 80 \
      --cidr 0.0.0.0/0 \
      --region "$AWS_REGION" > /dev/null 2>&1
} &
HTTP_PID=$!

# Wait for both operations to complete
wait $SSH_PID || {
    echo "ERROR: Failed to add SSH ingress rule" >&2
    exit 1
}
wait $HTTP_PID || {
    echo "ERROR: Failed to add HTTP ingress rule" >&2
    exit 1
}

echo "Security group configured with SSH and HTTP access."
```

```
echo "WARNING: SSH rule allows access from any IP (0.0.0.0/0). Restrict this in
 production."
echo "WARNING: In a production environment, you should restrict access to
 specific IP ranges."
echo ""

# Step 4: Get the latest Amazon Linux 2 AMI ID - Use pagination to optimize costs
echo "Getting the latest Amazon Linux 2 AMI ID..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-2.0.*-x86_64-gp2"
 "Name=state,Values=available" \
  --region "$AWS_REGION" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text \
  --max-results 50) || {
    echo "ERROR: Failed to describe images" >&2
    exit 1
}

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    echo "ERROR: Could not find a suitable AMI ID" >&2
    exit 1
fi

echo "Using AMI ID: $AMI_ID"
echo "Note: In a real marketplace scenario, you would use the AMI ID from your
 subscribed product."
echo ""

# Step 5: Launch an EC2 instance with cost optimization
echo "Launching EC2 instance with the AMI..."
echo "Using t2.micro (eligible for AWS Free Tier if applicable)"
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --count 1 \
  --region "$AWS_REGION" \
  --monitoring Enabled=false \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=marketplace-buyer-gs}]' \
  --output json) || {
```

```
    echo "ERROR: Failed to launch instance" >&2
    exit 1
}

# Extract instance ID using jq for safe parsing
INSTANCE_ID=$(extract_json_value "$INSTANCE_OUTPUT" '.Instances[0].InstanceId')
 || exit 1
if [ -z "$INSTANCE_ID" ] || [ "$INSTANCE_ID" = "null" ]; then
    echo "ERROR: Could not extract instance ID" >&2
    exit 1
fi
echo "Instance launched with ID: $INSTANCE_ID"

# Wait for the instance to be running
echo "Waiting for instance to be in running state..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" || {
    echo "ERROR: Instance failed to reach running state" >&2
    exit 1
}
echo "Instance is now running."
echo ""

# Step 6: Get instance details
echo "Getting instance details..."
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --region "$AWS_REGION" \
  --output json) || {
    echo "ERROR: Failed to describe instance" >&2
    exit 1
}

echo "Instance details:"
extract_json_value "$INSTANCE_DETAILS" '.Reservations[0].Instances[0] |
 {InstanceId, State: .State.Name, PublicDnsName, PrivateIpAddress, LaunchTime,
 InstanceType}' || exit 1

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Key Pair: $KEY_NAME"
```

```
echo "Security Group: $SECURITY_GROUP_NAME (ID: $SECURITY_GROUP_ID)"
echo "EC2 Instance: $INSTANCE_ID"
echo "Instance Type: t2.micro (cost-optimized)"
echo "AMI ID: $AMI_ID"
echo "Region: $AWS_REGION"
echo ""
echo "COST OPTIMIZATION NOTES:"
echo "- t2.micro instances are eligible for AWS Free Tier (750 hours/month for 12
 months)"
echo "- Detailed monitoring is disabled to reduce costs"
echo "- Consider using Spot Instances for non-production workloads"
echo "- Review AWS Pricing Calculator: https://calculator.aws/"
echo ""
echo "To connect to your instance (once it's fully initialized):"
echo "ssh -i ${KEY_NAME}.pem ec2-user@<public-dns-name>"
echo "Replace <public-dns-name> with the PublicDnsName from the instance details
 above."
echo ""

# Auto-confirm cleanup of resources
echo "====================================================="
echo "CLEANUP CONFIRMATION"
echo "====================================================="
echo "Cleaning up all created resources..."

echo ""
echo "Script completed. See $LOG_FILE for the complete log."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - RunInstances
  - TerminateInstances
  - Wait

Introdução aos clusters de processamento de big data

O exemplo de código a seguir mostra como:

- Criar um par de chaves do EC2.
- Configurar o armazenamento e preparar uma aplicação.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
```

```
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
 true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
 then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi
```

```
    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
```

```
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
 find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
 's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
```

```
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red
 violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
```

```
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
 data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
 text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
```

```
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
```

```
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout
 period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi
```

```
# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
 "$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
 files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
 | head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
 | head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
```

```
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles

- CreateKeyPair
- DescribeCluster
- DescribeKeyPairs
- DescribeStep
- Ls
- Mb
- Rb
- Rm
- ssh
- TerminateClusters
- Wait

Introdução aos bancos de dados de documentos

O exemplo de código a seguir mostra como:

- Criar um grupo de sub-redes de banco de dados
- Criar um cluster do DocumentDB.
- Criar uma instância do DocumentDB.
- Configurar a segurança e a conectividade.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
```

```
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

###############################################################################
# Configuration
###############################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

###############################################################################
# Logging
###############################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""

###############################################################################
# Region pre-check
###############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
```

```
echo "Using region: $REGION"
echo ""

##############################################################################
# Error handler
##############################################################################
handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

##############################################################################
# Wait function
##############################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
```

```
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
```

```
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound
\|not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
```

```
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG
 ingress rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed
 to delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete
 cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}
```

```
##############################################################################
# Step 1: Generate password and store in Secrets Manager
##############################################################################
echo "============================================"
echo "Step 1: Create master password in Secrets Manager"
echo "============================================"
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "============================================"
```

```
echo "Step 2: Find default VPC and subnets"
echo "============================================"
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi
```

```
# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

################################################################################
# Step 3: Create subnet group
################################################################################
echo "============================================"
echo "Step 3: Create DocumentDB subnet group"
echo "============================================"
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

################################################################################
# Step 4: Create DocumentDB cluster
################################################################################
echo "============================================"
echo "Step 4: Create DocumentDB cluster"
echo "============================================"
```

```
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "============================================"
echo "Step 5: Create DocumentDB instance"
echo "============================================"
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
```

```
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

##############################################################################
# Step 6: Get cluster endpoint and security group
##############################################################################
echo "==========================================="
echo "Step 6: Get cluster endpoint and security group"
echo "==========================================="
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
```

```
# Step 7: Add security group ingress for port 27017 from user's IP
##############################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null
 || true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
```

```
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "==========================================="
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "==========================================="
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

################################################################################
# Step 9: Display connection information
################################################################################
```

```
echo "==========================================="
echo "CONNECTION INFORMATION"
echo "==========================================="
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH}
 \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString
 --output text)"
echo ""

###############################################################################
# Step 10: Cleanup
###############################################################################
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources
```

```
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - RevokeSecurityGroupIngress
  - Wait

Introdução aos bancos de dados de grafos

O exemplo de código a seguir mostra como:

- Criar uma VPC para um banco de dados do Neptune.
- Criar sub-redes em várias zonas de disponibilidade.
- Configurar a segurança do banco de dados do Neptune.
- Criar um grupo de sub-redes do Neptune.
- Criar uma instância e um cluster de banco de dados do Neptune.
- Adicionar dados a um banco de dados de grafos.
- Consultar um banco de dados de grafos.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Neptune Getting Started Script
# This script creates an Amazon Neptune database cluster and demonstrates basic
 operations

# Set up logging
LOG_FILE="neptune-setup.log"
echo "Starting Neptune setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
```

```
    echo "Error encountered. Cleaning up resources..." | tee -a "$LOG_FILE"

    # Only attempt to delete resources that were successfully created
    if [[ -n "$DB_INSTANCE_ID" ]]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"
        log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"
    fi

    if [[ -n "$DB_CLUSTER_ID" ]]; then
        echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --skip-final-snapshot"
    fi

    if [[ -n "$DB_SUBNET_GROUP" ]]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"
    fi

    if [[ -n "$SECURITY_GROUP_ID" ]]; then
        echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
    fi

    if [[ -n "$SUBNET_IDS" ]]; then
        for SUBNET_ID in $SUBNET_IDS; do
            echo "Deleting subnet $SUBNET_ID..." | tee -a "$LOG_FILE"
            log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET_ID"
        done
    fi

    if [[ -n "$VPC_ID" ]]; then
        echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"
    fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
VPC_NAME="neptune-vpc-$RANDOM_ID"
```

```
DB_SUBNET_GROUP="neptune-subnet-group-$RANDOM_ID"
DB_CLUSTER_ID="neptune-cluster-$RANDOM_ID"
DB_INSTANCE_ID="neptune-instance-$RANDOM_ID"
SG_NAME="neptune-sg-$RANDOM_ID"

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "VPC Name: $VPC_NAME" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "DB Cluster ID: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "DB Instance ID: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "Security Group Name: $SG_NAME" | tee -a "$LOG_FILE"

# Step 1: Create VPC
echo "Creating VPC..." | tee -a "$LOG_FILE"
VPC_OUTPUT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=$VPC_NAME},{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-neptune-gs}]" --output json)
check_error "$VPC_OUTPUT" $? "Failed to create VPC"

VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID" | tee -a "$LOG_FILE"

# Enable DNS support for the VPC
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support"
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames"

# Step 2: Create Internet Gateway and attach to VPC
echo "Creating Internet Gateway..." | tee -a "$LOG_FILE"
IGW_OUTPUT=$(aws ec2 create-internet-gateway --tag-specifications
 'ResourceType=internet-gateway,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$IGW_OUTPUT" $? "Failed to create Internet Gateway"

IGW_ID=$(echo "$IGW_OUTPUT" | grep -o '"InternetGatewayId": "[^"]*' | cut -d'"' -
f4)
echo "Internet Gateway created with ID: $IGW_ID" | tee -a "$LOG_FILE"

log_cmd "aws ec2 attach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"

# Step 3: Create subnets in different AZs
echo "Creating subnets..." | tee -a "$LOG_FILE"

# Get available AZs
```

```
AZ_OUTPUT=$(aws ec2 describe-availability-zones --output json)
check_error "$AZ_OUTPUT" $? "Failed to get availability zones"

# Extract first 3 AZ names
AZ1=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -1)
AZ2=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -2
 | tail -1)
AZ3=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -3
 | tail -1)

# Create 3 subnets in different AZs
SUBNET1_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.1.0/24 --availability-zone $AZ1 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET1_OUTPUT" $? "Failed to create subnet 1"
SUBNET1_ID=$(echo "$SUBNET1_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET2_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.2.0/24 --availability-zone $AZ2 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET2_OUTPUT" $? "Failed to create subnet 2"
SUBNET2_ID=$(echo "$SUBNET2_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET3_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.3.0/24 --availability-zone $AZ3 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET3_OUTPUT" $? "Failed to create subnet 3"
SUBNET3_ID=$(echo "$SUBNET3_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET_IDS="$SUBNET1_ID $SUBNET2_ID $SUBNET3_ID"
echo "Created subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a
 "$LOG_FILE"

# Step 4: Create route table and add route to Internet Gateway
echo "Creating route table..." | tee -a "$LOG_FILE"
ROUTE_TABLE_OUTPUT=$(aws ec2 create-route-table --vpc-id $VPC_ID --tag-
specifications 'ResourceType=route-table,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
```

```
check_error "$ROUTE_TABLE_OUTPUT" $? "Failed to create route table"

ROUTE_TABLE_ID=$(echo "$ROUTE_TABLE_OUTPUT" | grep -o '"RouteTableId": "[^"]*' |
 cut -d'"' -f4)
echo "Route table created with ID: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"

# Add route to Internet Gateway
log_cmd "aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-
cidr-block 0.0.0.0/0 --gateway-id $IGW_ID"

# Associate route table with subnets
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET1_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET2_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET3_ID"

# Step 5: Create security group
echo "Creating security group..." | tee -a "$LOG_FILE"
SG_OUTPUT=$(aws ec2 create-security-group --group-name $SG_NAME --description
 "Security group for Neptune" --vpc-id $VPC_ID --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SG_OUTPUT" $? "Failed to create security group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
echo "Security group created with ID: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"

# Add inbound rule for Neptune port (8182)
# Note: In production, you should restrict this to specific IP ranges
echo "Adding security group rule for Neptune port 8182..." | tee -a "$LOG_FILE"
log_cmd "aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID
 --protocol tcp --port 8182 --cidr 10.0.0.0/16"

# Step 6: Create DB subnet group
echo "Creating DB subnet group..." | tee -a "$LOG_FILE"
DB_SUBNET_GROUP_OUTPUT=$(aws neptune create-db-subnet-group --db-subnet-group-
name $DB_SUBNET_GROUP --db-subnet-group-description "Subnet group for Neptune" --
subnet-ids $SUBNET1_ID $SUBNET2_ID $SUBNET3_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_SUBNET_GROUP_OUTPUT" $? "Failed to create DB subnet group"
echo "DB subnet group created: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
```

```
# Step 7: Create Neptune DB cluster
echo "Creating Neptune DB cluster..." | tee -a "$LOG_FILE"
DB_CLUSTER_OUTPUT=$(aws neptune create-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --engine neptune --vpc-security-group-ids $SECURITY_GROUP_ID
 --db-subnet-group-name $DB_SUBNET_GROUP --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_CLUSTER_OUTPUT" $? "Failed to create Neptune DB cluster"
echo "Neptune DB cluster created: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"

# Step 8: Create Neptune DB instance
echo "Creating Neptune DB instance..." | tee -a "$LOG_FILE"
DB_INSTANCE_OUTPUT=$(aws neptune create-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --db-instance-class db.r5.large --engine neptune --db-
cluster-identifier $DB_CLUSTER_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_INSTANCE_OUTPUT" $? "Failed to create Neptune DB instance"
echo "Neptune DB instance created: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"

# Step 9: Wait for the DB instance to become available
echo "Waiting for Neptune DB instance to become available..." | tee -a
 "$LOG_FILE"
log_cmd "aws neptune wait db-instance-available --db-instance-identifier
 $DB_INSTANCE_ID"

# Step 10: Get the Neptune endpoint
echo "Getting Neptune endpoint..." | tee -a "$LOG_FILE"
ENDPOINT_OUTPUT=$(aws neptune describe-db-clusters --db-cluster-identifier
 $DB_CLUSTER_ID --output json)
check_error "$ENDPOINT_OUTPUT" $? "Failed to get Neptune endpoint"

NEPTUNE_ENDPOINT=$(echo "$ENDPOINT_OUTPUT" | grep -o '"Endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Neptune endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"

# Step 11: Display information about how to connect to Neptune
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "NEPTUNE SETUP COMPLETE" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Neptune Endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"
echo "Port: 8182" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
```

```
echo "To query your Neptune database using Gremlin, you can use curl:" | tee -a
 "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.V().limit(1)\"}' https://
$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To add data to your graph:" | tee -a "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.addV(\\\"person\\\").property(\\\"name
\\\", \\\"Howard\\\")\"}' https://$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Note: You may need to configure your client machine to access the Neptune
 instance within the VPC." | tee -a "$LOG_FILE"
echo "For production use, consider using an EC2 instance in the same VPC or
 setting up a bastion host." | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 12: List all created resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "RESOURCES CREATED" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "VPC: $VPC_ID" | tee -a "$LOG_FILE"
echo "Internet Gateway: $IGW_ID" | tee -a "$LOG_FILE"
echo "Subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"
echo "Route Table: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"
echo "Security Group: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "Neptune DB Cluster: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "Neptune DB Instance: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 13: Ask if user wants to clean up resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Do you want to clean up all created resources? (y/n): " | tee -a
 "$LOG_FILE"
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..." | tee -a "$LOG_FILE"

    # Delete DB instance
```

```
    echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"

    # Wait for DB instance to be deleted
    echo "Waiting for DB instance to be deleted..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"

    # Delete DB cluster
    echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-cluster --db-cluster-identifier $DB_CLUSTER_ID
 --skip-final-snapshot"

    # Wait for DB cluster to be deleted (no specific wait command for this, so
 we'll sleep)
    echo "Waiting for DB cluster to be deleted..." | tee -a "$LOG_FILE"
    sleep 60

    # Delete DB subnet group
    echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"

    # Delete security group
    echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"

    # Detach and delete internet gateway
    echo "Detaching and deleting internet gateway $IGW_ID..." | tee -a
 "$LOG_FILE"
    log_cmd "aws ec2 detach-internet-gateway --internet-gateway-id $IGW_ID --vpc-
id $VPC_ID"
    log_cmd "aws ec2 delete-internet-gateway --internet-gateway-id $IGW_ID"

    # Delete subnets
    echo "Deleting subnets..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET1_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET2_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET3_ID"

    # Delete route table
    echo "Deleting route table $ROUTE_TABLE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-route-table --route-table-id $ROUTE_TABLE_ID"
```

```
    # Delete VPC
    echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"

    echo "Cleanup complete!" | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can delete them manually later."
 | tee -a "$LOG_FILE"
    echo "See the list of resources above for reference." | tee -a "$LOG_FILE"
fi

echo "Script completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateInternetGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteInternetGateway
  - DeleteRouteTable
  - DeleteSecurityGroup
  - DeleteSubnet

- DeleteVpc
- DescribeAvailabilityZones
- DescribeDbClusters
- DetachInternetGateway
- ModifyVpcAttribute
- Wait

Introdução ao armazenamento em cache em memória

O exemplo de código a seguir mostra como:

- Configurar grupo de segurança para ElastiCache acesso
- Criar um cache sem servidor do Valkey.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ElastiCache Getting Started Script
# This script creates a Valkey serverless cache, configures security groups,
# and demonstrates how to connect to and use the cache.

set -uo pipefail

# Set up logging
LOG_FILE="elasticache_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "Starting ElastiCache tutorial script. Logging to $LOG_FILE"
echo "============================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${CACHE_NAME:-}" ]; then
        echo "- ElastiCache serverless cache: $CACHE_NAME"
    fi
    if [ -n "${SG_RULE_6379:-}" ] || [ -n "${SG_RULE_6380:-}" ]; then
        echo "- Security group rules for ports 6379 and 6380"
    fi
    echo "Please clean up these resources manually."
    exit 1
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CACHE_NAME="valkey-cache-${RANDOM_ID}"

echo "Using cache name: $CACHE_NAME"

# Step 1: Set up security group for ElastiCache access
echo "Step 1: Setting up security group for ElastiCache access..."

# Get default security group ID
echo "Getting default security group ID..."
SG_ID=$(aws ec2 describe-security-groups \
  --filters Name=group-name,Values=default \
  --query "SecurityGroups[0].GroupId" \
  --output text 2>/dev/null || echo "")

if [[ -z "$SG_ID" || "$SG_ID" == "None" ]]; then
```

```
    handle_error "Failed to get default security group ID"
fi

echo "Default security group ID: $SG_ID"

# Add inbound rule for port 6379
echo "Adding inbound rule for port 6379..."
SG_RULE_6379=""
if SG_RULE_6379=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6379 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379"
 == *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    fi
else
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379"
 == *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    else
        handle_error "Failed to add security group rule for port 6379:
 $SG_RULE_6379"
    fi
fi

# Add inbound rule for port 6380
echo "Adding inbound rule for port 6380..."
SG_RULE_6380=""
if SG_RULE_6380=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6380 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380"
 == *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
```

```
        SG_RULE_6380="existing"
    fi
else
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380"
 == *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    else
        handle_error "Failed to add security group rule for port 6380:
 $SG_RULE_6380"
    fi
fi

echo "Security group rules added successfully."
echo ""
echo "##  SECURITY WARNING: The security group rules created allow access from
 any IP address (0.0.0.0/0)."
echo "This is NOT RECOMMENDED for production environments. For production,"
echo "you should restrict access to specific IP ranges or security groups."
echo "Update the CIDR blocks in this script before using in production."
echo ""

# Step 2: Create a Valkey serverless cache
echo "Step 2: Creating Valkey serverless cache..."
if ! CREATE_RESULT=$(aws elasticache create-serverless-cache \
  --serverless-cache-name "$CACHE_NAME" \
  --engine valkey \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-elasticache-gs
 2>&1); then
    handle_error "Failed to create serverless cache: $CREATE_RESULT"
fi

echo "Cache creation initiated. Waiting for cache to become available..."

# Step 3: Check the status of the cache creation
echo "Step 3: Checking cache status..."

# Wait for the cache to become active
MAX_ATTEMPTS=30
ATTEMPT=1
CACHE_STATUS=""

while [[ $ATTEMPT -le $MAX_ATTEMPTS ]]; do
    echo "Checking cache status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."
```

```
    if ! DESCRIBE_RESULT=$(aws elasticache describe-serverless-caches \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        handle_error "Failed to describe serverless cache: $DESCRIBE_RESULT"
    fi

    # Extract status using jq for reliable JSON parsing
    if command -v jq &> /dev/null; then
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | jq -r
 '.ServerlessCaches[0].Status // "UNKNOWN"' 2>/dev/null || echo "")
    else
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | grep -o '"Status": "[^"]*"' |
 awk -F'"' '{print $4}' | head -n 1)
    fi

    echo "Current status: $CACHE_STATUS"

    if [[ "${CACHE_STATUS,,}" == "available" ]]; then
        echo "Cache is now available!"
        break
    elif [[ "${CACHE_STATUS,,}" == "create-failed" ]]; then
        handle_error "Cache creation failed. Please check the AWS console for
 details."
    fi

    if [[ $ATTEMPT -lt $MAX_ATTEMPTS ]]; then
        echo "Waiting 30 seconds..."
        sleep 30
    fi

    ((ATTEMPT++))
done

if [[ "${CACHE_STATUS,,}" != "available" ]]; then
    handle_error "Cache did not become available within the expected time. Last
 status: $CACHE_STATUS"
fi

# Step 4: Find your cache endpoint
echo "Step 4: Getting cache endpoint..."
if ! ENDPOINT=$(aws elasticache describe-serverless-caches \
  --serverless-cache-name "$CACHE_NAME" \
  --query "ServerlessCaches[0].Endpoint.Address" \
  --output text 2>&1); then
```

```
    handle_error "Failed to get cache endpoint: $ENDPOINT"
fi

if [[ -z "$ENDPOINT" || "$ENDPOINT" == "None" ]]; then
    handle_error "Failed to get cache endpoint"
fi

echo "Cache endpoint: $ENDPOINT"

# Step 5: Instructions for connecting to the cache
echo ""
echo "============================================================"
echo "Your Valkey serverless cache has been successfully created!"
echo "Cache Name: $CACHE_NAME"
echo "Endpoint: $ENDPOINT"
echo "============================================================"
echo ""
echo "To connect to your cache from an EC2 instance, follow these steps:"
echo ""
echo "1. Install valkey-cli on your EC2 instance:"
echo "   sudo amazon-linux-extras install epel -y"
echo "   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y"
echo "   wget https://github.com/valkey-io/valkey/archive/refs/tags/8.0.0.tar.gz"
echo "   tar xvzf 8.0.0.tar.gz"
echo "   cd valkey-8.0.0"
echo "   make BUILD_TLS=yes"
echo ""
echo "2. Connect to your cache using valkey-cli:"
echo "   src/valkey-cli -h $ENDPOINT --tls -p 6379"
echo ""
echo "3. Once connected, you can run commands like:"
echo "   set mykey \"Hello ElastiCache\""
echo "   get mykey"
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- ElastiCache serverless cache: $CACHE_NAME"
if [[ "${SG_RULE_6379:-}" != "existing" ]] || [[ "${SG_RULE_6380:-}" !=
 "existing" ]]; then
```

```
    echo "- Security group rules for ports 6379 and 6380"
fi
echo ""
echo "Proceeding with cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Step 7: Delete the cache
    echo "Deleting serverless cache $CACHE_NAME..."
    if ! DELETE_RESULT=$(aws elasticache delete-serverless-cache \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        echo "WARNING: Failed to delete serverless cache: $DELETE_RESULT"
        echo "Please delete the cache manually from the AWS console."
    else
        echo "Cache deletion initiated. This may take several minutes to
 complete."
    fi

    # Only attempt to remove security group rules if we created them
    if [[ "${SG_RULE_6379:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6379..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6379 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6379"
        fi
    fi

    if [[ "${SG_RULE_6380:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6380..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6380 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6380"
        fi
    fi
```

```
    echo "Cleanup completed."
fi

echo ""
echo "Script completed. See $LOG_FILE for the full log."
echo "============================================================"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateServerlessCache
  - DeleteServerlessCache
  - DescribeSecurityGroups
  - DescribeServerlessCaches
  - RevokeSecurityGroupIngress

Introdução ao balanceamento de carga

O exemplo de código a seguir mostra como:

- Criar um Application Load Balancer.
- Criar um grupo de destino.
- Criar um receptor.
- Verificar sua configuração.
- Adicionar um receptor HTTPS (opcional).
- Adicionar roteamento baseado em caminho (opcional).
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Elastic Load Balancing Getting Started Script - v2
# This script creates an Application Load Balancer with HTTP listener and target
 group

set -euo pipefail

# Set up logging
LOG_FILE="elb-script-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Elastic Load Balancing setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check AWS CLI command success
check_command() {
    local output="$1"
    if [[ -z "$output" ]] || [[ "$output" == "None" ]]; then
        handle_error "AWS CLI command returned empty or invalid output"
    fi
}

# Function to validate ARN format
validate_arn() {
    local arn="$1"
    if [[ ! "$arn" =~ ^arn:aws:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$ ]]; then
        handle_error "Invalid ARN format: $arn"
    fi
}

# Function to validate security group ID
validate_security_group_id() {
    local sg_id="$1"
    if [[ ! "$sg_id" =~ ^sg-[a-f0-9]{8,17}$ ]]; then
```

```
        handle_error "Invalid security group ID format: $sg_id"
    fi
}

# Function to validate VPC ID
validate_vpc_id() {
    local vpc_id="$1"
    if [[ ! "$vpc_id" =~ ^vpc-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid VPC ID format: $vpc_id"
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    if [ -n "${LISTENER_ARN:-}" ]; then
        echo "Deleting listener: $LISTENER_ARN"
        aws elbv2 delete-listener --listener-arn "$LISTENER_ARN" 2>/dev/null ||
 true
    fi

    if [ -n "${LOAD_BALANCER_ARN:-}" ]; then
        echo "Deleting load balancer: $LOAD_BALANCER_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LOAD_BALANCER_ARN"
 2>/dev/null || true

        # Wait for load balancer to be deleted before deleting target group
        echo "Waiting for load balancer to be deleted..."
        aws elbv2 wait load-balancers-deleted --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null || true
    fi

    if [ -n "${TARGET_GROUP_ARN:-}" ]; then
        echo "Deleting target group: $TARGET_GROUP_ARN"
        aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN" 2>/
dev/null || true
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting 30 seconds before deleting security group to ensure all
 dependencies are removed..."
        sleep 30
```

```
        echo "Deleting security group: $SECURITY_GROUP_ID"
        local sg_delete_output
        sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)

        local retry_count=0
        local max_retries=5
        while echo "$sg_delete_output" | grep -i "DependencyViolation" > /dev/
null && [ $retry_count -lt $max_retries ]; do
            retry_count=$((retry_count+1))
            echo "Security group still has dependencies. Retrying in 30
 seconds... (Attempt $retry_count of $max_retries)"
            sleep 30
            sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)
        done

        if echo "$sg_delete_output" | grep -i "error" > /dev/null; then
            echo "WARNING: Could not delete security group: $SECURITY_GROUP_ID"
 >&2
            echo "You may need to delete it manually using: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        else
            echo "Security group deleted successfully."
        fi
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
RESOURCE_PREFIX="elb-demo-${RANDOM_ID}"

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Step 1: Verify AWS CLI support for Elastic Load Balancing
echo "Verifying AWS CLI support for Elastic Load Balancing..."
if ! aws elbv2 help > /dev/null 2>&1; then
    handle_error "AWS CLI does not support elbv2 commands. Please update your AWS
 CLI."
fi
```

```
# Step 2: Get VPC ID and subnet information
echo "Retrieving VPC information..."
VPC_INFO=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 "Vpcs[0].VpcId" --output text 2>/dev/null || echo "")
check_command "$VPC_INFO"
VPC_ID="$VPC_INFO"
validate_vpc_id "$VPC_ID"
echo "Using VPC: $VPC_ID"

# Get two subnets from different Availability Zones
echo "Retrieving subnet information..."
SUBNET_INFO=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[0:2].SubnetId" --output text 2>/dev/null || echo "")
check_command "$SUBNET_INFO"

# Convert space-separated list to array
read -r -a SUBNETS <<< "$SUBNET_INFO"
if [ ${#SUBNETS[@]} -lt 2 ]; then
    handle_error "Need at least 2 subnets in different Availability Zones. Found:
 ${#SUBNETS[@]}"
fi

echo "Using subnets: ${SUBNETS[0]} and ${SUBNETS[1]}"

# Step 3: Create a security group for the load balancer
echo "Creating security group for the load balancer..."
SG_INFO=$(aws ec2 create-security-group \
    --group-name "${RESOURCE_PREFIX}-sg" \
    --description "Security group for ELB demo" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=elastic-load-
balancing-gs}]' \
    --query "GroupId" --output text 2>/dev/null || echo "")
check_command "$SG_INFO"
SECURITY_GROUP_ID="$SG_INFO"
validate_security_group_id "$SECURITY_GROUP_ID"
echo "Created security group: $SECURITY_GROUP_ID"

# Add inbound rule to allow HTTP traffic
echo "Adding inbound rule to allow HTTP traffic..."
aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
```

```
    --port 80 \
    --cidr "0.0.0.0/0" > /dev/null 2>&1 || handle_error "Failed to authorize
 security group ingress"

echo "WARNING: Security group allows HTTP from 0.0.0.0/0. In production, restrict
 to specific IP addresses." >&2

# Step 4: Create the load balancer
echo "Creating Application Load Balancer..."
LB_INFO=$(aws elbv2 create-load-balancer \
    --name "${RESOURCE_PREFIX}-lb" \
    --subnets "${SUBNETS[0]}" "${SUBNETS[1]}" \
    --security-groups "$SECURITY_GROUP_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "LoadBalancers[0].LoadBalancerArn" --output text 2>/dev/null || echo
 "")
check_command "$LB_INFO"
LOAD_BALANCER_ARN="$LB_INFO"
validate_arn "$LOAD_BALANCER_ARN"
echo "Created load balancer: $LOAD_BALANCER_ARN"

# Wait for the load balancer to be active
echo "Waiting for load balancer to become active..."
if ! aws elbv2 wait load-balancer-available --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null; then
    handle_error "Load balancer did not reach active state within timeout period"
fi

# Step 5: Create a target group
echo "Creating target group..."
TG_INFO=$(aws elbv2 create-target-group \
    --name "${RESOURCE_PREFIX}-targets" \
    --protocol HTTP \
    --port 80 \
    --vpc-id "$VPC_ID" \
    --target-type instance \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "TargetGroups[0].TargetGroupArn" --output text 2>/dev/null || echo
 "")
check_command "$TG_INFO"
TARGET_GROUP_ARN="$TG_INFO"
validate_arn "$TARGET_GROUP_ARN"
```

```
echo "Created target group: $TARGET_GROUP_ARN"

# Step 6: Find EC2 instances to register as targets
echo "Looking for available EC2 instances to register as targets..."
INSTANCES=$(aws ec2 describe-instances \
    --filters "Name=vpc-id,Values=$VPC_ID" "Name=instance-state-
name,Values=running" \
    --query "Reservations[*].Instances[*].InstanceId" --output text 2>/dev/null
 || echo "")

# Convert space-separated list to array
read -r -a INSTANCE_IDS <<< "$INSTANCES"

if [ ${#INSTANCE_IDS[@]} -eq 0 ]; then
    echo "No running instances found in VPC $VPC_ID."
    echo "You will need to register targets manually after launching instances."
else
    # Step 7: Register targets with the target group (up to 2 instances)
    echo "Registering targets with the target group..."
    target_args=()
    for i in "${!INSTANCE_IDS[@]}"; do
        if [ "$i" -lt 2 ]; then
            target_args+=("Id=${INSTANCE_IDS[$i]}")
        fi
    done

    if [ ${#target_args[@]} -gt 0 ]; then
        if aws elbv2 register-targets \
            --target-group-arn "$TARGET_GROUP_ARN" \
            --targets "${target_args[@]}" 2>/dev/null; then
            echo "Registered instances: ${target_args[*]}"
        else
            handle_error "Failed to register targets"
        fi
    fi
fi

# Step 8: Create a listener
echo "Creating HTTP listener..."
LISTENER_INFO=$(aws elbv2 create-listener \
    --load-balancer-arn "$LOAD_BALANCER_ARN" \
    --protocol HTTP \
    --port 80 \
    --default-actions Type=forward,TargetGroupArn="$TARGET_GROUP_ARN" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "Listeners[0].ListenerArn" --output text 2>/dev/null || echo "")
check_command "$LISTENER_INFO"
LISTENER_ARN="$LISTENER_INFO"
validate_arn "$LISTENER_ARN"
echo "Created listener: $LISTENER_ARN"

# Step 9: Verify target health
echo "Verifying target health..."
aws elbv2 describe-target-health --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/
null || true

# Display load balancer DNS name
LB_DNS=$(aws elbv2 describe-load-balancers \
    --load-balancer-arns "$LOAD_BALANCER_ARN" \
    --query "LoadBalancers[0].DNSName" --output text 2>/dev/null || echo "")
check_command "$LB_DNS"

echo ""
echo "================================================"
echo "SETUP COMPLETE"
echo "================================================"
echo "Load Balancer DNS Name: $LB_DNS"
echo ""
echo "Resources created:"
echo "- Load Balancer: $LOAD_BALANCER_ARN"
echo "- Target Group: $TARGET_GROUP_ARN"
echo "- Listener: $LISTENER_ARN"
echo "- Security Group: $SECURITY_GROUP_ID"
echo ""

# Prompt for cleanup confirmation
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
read -p "Do you want to clean up all created resources? (y/n): " -r
 CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources have been preserved."
    echo "To clean up later, run the following commands:"
    echo "aws elbv2 delete-listener --listener-arn $LISTENER_ARN"
    echo "aws elbv2 delete-load-balancer --load-balancer-arn $LOAD_BALANCER_ARN"
    echo "aws elbv2 wait load-balancers-deleted --load-balancer-arns
 $LOAD_BALANCER_ARN"
    echo "aws elbv2 delete-target-group --target-group-arn $TARGET_GROUP_ARN"
    echo "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateListener
  - CreateLoadBalancer
  - CreateSecurityGroup
  - CreateTargetGroup
  - DeleteListener
  - DeleteLoadBalancer
  - DeleteSecurityGroup
  - DeleteTargetGroup
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetHealth
  - DescribeVpcs
  - Help
  - RegisterTargets
  - Wait

Conceitos básicos de streaming gerenciado

O exemplo de código a seguir mostra como:

- Criar um cluster do Amazon MSK.
- Criar permissões do IAM para acessar o MSK.
- Criar uma máquina cliente
- Obter os agentes de bootstrap.
- Configurar a máquina cliente.
- Crie um tópico e produce/consume dados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
```

```
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
```

```
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
```

```
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
```

```
   echo "Cleaning up resources..."

   # Delete EC2 instance if it exists
   if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
       echo "Terminating EC2 instance: $INSTANCE_ID"
       aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
to terminate instance"
       echo "Waiting for instance to terminate..."
       aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
"Failed to wait for instance termination"
   fi

   # Delete MSK cluster first (to remove dependencies on security group)
   if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
       echo "Deleting MSK cluster: $CLUSTER_ARN"
       aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
delete cluster"

       # Wait a bit for the cluster deletion to start
       echo "Waiting 30 seconds for cluster deletion to begin..."
       sleep 30
   fi

   # Remove security group references before attempting deletion
   if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
"$CLIENT_SG_ID"; then
       remove_security_group_references "$CLIENT_SG_ID"

       echo "Deleting security group: $CLIENT_SG_ID"
       # Try multiple times with longer delays to ensure dependencies are
removed
       for i in {1..10}; do
           if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
               echo "Security group deleted successfully"
               break
           fi
           echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
           sleep 30
       done
   fi

   # Delete key pair if it exists
   if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
```

```
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
```

```
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
```

```
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
```

```
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)
```

```
# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
```

```
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
```

```
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
```

```
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)
```

```
if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi
```

```
echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
```

```
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
```

```
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi
```

```
echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
```

```
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
```

```
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
```

```
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi
```

```
# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
```

```
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
```

```
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole

- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Introdução aos gateways de trânsito de rede

O exemplo de código a seguir mostra como:

- Criar um gateway de trânsito.
- Anexar as VPCs ao gateway de trânsito.
- Adicionar rotas entre o gateway de trânsito e as VPCs.
- Testar o gateway de trânsito.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon VPC Transit Gateway CLI Script
# This script demonstrates how to create a transit gateway and connect two VPCs
# Modified to work with older AWS CLI versions that don't support transit gateway
 wait commands
# Security improved: Added input validation, error handling, and credential
 checks

# Error handling
set -euo pipefail
LOG_FILE="transit-gateway-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Security: Check for required AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
  echo "ERROR: AWS credentials not configured or invalid. Please configure AWS
 credentials."
  exit 1
fi
```

```
# Function to validate AWS CLI output
validate_aws_output() {
  local output=$1
  local context=$2

  if [ -z "$output" ] || [ "$output" = "None" ]; then
    echo "ERROR: Failed to retrieve $context from AWS API"
    return 1
  fi
}

# Function to validate CIDR blocks
validate_cidr() {
  local cidr=$1
  if ! [[ "$cidr" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}$ ]]; then
    echo "ERROR: Invalid CIDR block format: $cidr"
    return 1
  fi
}

# Function to wait for transit gateway to be available
wait_for_tgw() {
  local tgw_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway $tgw_id to become available..."

  while [ $attempt -lt $max_attempts ]; do
    status=$(aws ec2 describe-transit-gateways \
      --transit-gateway-ids "$tgw_id" \
      --query "TransitGateways[0].State" \
      --output text 2>/dev/null || echo "failed")

    echo "Current status: $status"

    if [ "$status" = "available" ]; then
      echo "Transit Gateway is now available"
      return 0
    fi

    if [ "$status" = "failed" ]; then
      echo "ERROR: Transit Gateway creation failed"
```

```
      return 1
    fi

    echo "Waiting for transit gateway to become available. Current state:
 $status"
    sleep 10
    ((attempt++))
  done

  echo "ERROR: Timeout waiting for transit gateway to become available"
  return 1
}

# Function to wait for transit gateway attachment to be available
wait_for_tgw_attachment() {
  local attachment_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway Attachment $attachment_id to become
 available..."

  while [ $attempt -lt $max_attempts ]; do
    status=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --transit-gateway-attachment-ids "$attachment_id" \
      --query "TransitGatewayVpcAttachments[0].State" \
      --output text 2>/dev/null || echo "failed")

    echo "Current status: $status"

    if [ "$status" = "available" ]; then
      echo "Transit Gateway Attachment is now available"
      return 0
    fi

    if [ "$status" = "failed" ]; then
      echo "ERROR: Transit Gateway Attachment creation failed"
      return 1
    fi

    echo "Waiting for transit gateway attachment to become available. Current
 state: $status"
    sleep 10
    ((attempt++))
```

```
  done

  echo "ERROR: Timeout waiting for transit gateway attachment to become
 available"
  return 1
}

# Function to wait for transit gateway attachment to be deleted
wait_for_tgw_attachment_deleted() {
  local attachment_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway Attachment $attachment_id to be deleted..."

  while [ $attempt -lt $max_attempts ]; do
    count=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --filters "Name=transit-gateway-attachment-id,Values=$attachment_id" \
      --query "length(TransitGatewayVpcAttachments)" \
      --output text 2>/dev/null || echo "0")

    if [ "$count" = "0" ]; then
      echo "Transit Gateway Attachment has been deleted"
      return 0
    fi

    status=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --transit-gateway-attachment-ids "$attachment_id" \
      --query "TransitGatewayVpcAttachments[0].State" \
      --output text 2>/dev/null || echo "deleted")

    if [ "$status" = "deleted" ] || [ "$status" = "deleting" ]; then
      echo "Transit Gateway Attachment is being deleted. Current state: $status"
    fi

    echo "Waiting for transit gateway attachment to be deleted. Current state:
 $status"
    sleep 10
    ((attempt++))
  done

  echo "WARNING: Timeout waiting for transit gateway attachment to be deleted"
  return 0
}
```

```
# Function to clean up resources
cleanup() {
  local exit_code=$?
  echo "Error occurred (exit code: $exit_code). Cleaning up resources..."

  # Delete resources in reverse order
  if [ -n "${TGW_ATTACHMENT_1_ID:-}" ]; then
    echo "Deleting Transit Gateway VPC Attachment 1: $TGW_ATTACHMENT_1_ID"
    aws ec2 delete-transit-gateway-vpc-attachment \
      --transit-gateway-attachment-id "$TGW_ATTACHMENT_1_ID" &>/dev/null || true
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
  fi

  if [ -n "${TGW_ATTACHMENT_2_ID:-}" ]; then
    echo "Deleting Transit Gateway VPC Attachment 2: $TGW_ATTACHMENT_2_ID"
    aws ec2 delete-transit-gateway-vpc-attachment \
      --transit-gateway-attachment-id "$TGW_ATTACHMENT_2_ID" &>/dev/null || true
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true
  fi

  if [ -n "${TGW_ID:-}" ]; then
    echo "Deleting Transit Gateway: $TGW_ID"
    aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" &>/dev/null ||
 true
  fi

  exit "$exit_code"
}

# Set up trap for error handling
trap cleanup EXIT

echo "=== Amazon VPC Transit Gateway Tutorial ==="
echo "This script will create a transit gateway and connect two VPCs"
echo ""

# Get a valid availability zone dynamically
echo "Getting available AZ in current region..."
AZ=$(aws ec2 describe-availability-zones \
  --query "AvailabilityZones[0].ZoneName" \
  --output text)
validate_aws_output "$AZ" "availability zone" || exit 1
echo "Using availability zone: $AZ"
```

```
# Check if VPCs exist
echo "Checking for existing VPCs..."
VPC1_ID=$(aws ec2 describe-vpcs \
  --filters "Name=tag:Name,Values=VPC1" \
  --query "Vpcs[0].VpcId" \
  --output text)
VPC2_ID=$(aws ec2 describe-vpcs \
  --filters "Name=tag:Name,Values=VPC2" \
  --query "Vpcs[0].VpcId" \
  --output text)

if [ "$VPC1_ID" = "None" ] || [ -z "$VPC1_ID" ]; then
  echo "Creating VPC1..."
  VPC1_ID=$(aws ec2 create-vpc \
    --cidr-block 10.1.0.0/16 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
    --query Vpc.VpcId \
    --output text)
  validate_aws_output "$VPC1_ID" "VPC1" || exit 1
  echo "Created VPC1: $VPC1_ID"

  # Create a subnet in VPC1
  echo "Creating subnet in VPC1..."
  SUBNET1_ID=$(aws ec2 create-subnet \
    --vpc-id "$VPC1_ID" \
    --cidr-block 10.1.0.0/24 \
    --availability-zone "$AZ" \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-
Subnet},{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
    --query Subnet.SubnetId \
    --output text)
  validate_aws_output "$SUBNET1_ID" "VPC1 subnet" || exit 1
  echo "Created subnet in VPC1: $SUBNET1_ID"
else
  echo "Using existing VPC1: $VPC1_ID"
  SUBNET1_ID=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$VPC1_ID" \
    --query "Subnets[0].SubnetId" \
    --output text)
  if [ "$SUBNET1_ID" = "None" ] || [ -z "$SUBNET1_ID" ]; then
```

```
    echo "Creating subnet in VPC1..."
    SUBNET1_ID=$(aws ec2 create-subnet \
      --vpc-id "$VPC1_ID" \
      --cidr-block 10.1.0.0/24 \
      --availability-zone "$AZ" \
      --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-
Subnet},{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
      --query Subnet.SubnetId \
      --output text)
    validate_aws_output "$SUBNET1_ID" "VPC1 subnet" || exit 1
    echo "Created subnet in VPC1: $SUBNET1_ID"
  else
    echo "Using existing subnet in VPC1: $SUBNET1_ID"
  fi
fi

if [ "$VPC2_ID" = "None" ] || [ -z "$VPC2_ID" ]; then
  echo "Creating VPC2..."
  VPC2_ID=$(aws ec2 create-vpc \
    --cidr-block 10.2.0.0/16 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
    --query Vpc.VpcId \
    --output text)
  validate_aws_output "$VPC2_ID" "VPC2" || exit 1
  echo "Created VPC2: $VPC2_ID"

  # Create a subnet in VPC2
  echo "Creating subnet in VPC2..."
  SUBNET2_ID=$(aws ec2 create-subnet \
    --vpc-id "$VPC2_ID" \
    --cidr-block 10.2.0.0/24 \
    --availability-zone "$AZ" \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-
Subnet},{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
    --query Subnet.SubnetId \
    --output text)
  validate_aws_output "$SUBNET2_ID" "VPC2 subnet" || exit 1
  echo "Created subnet in VPC2: $SUBNET2_ID"
else
  echo "Using existing VPC2: $VPC2_ID"
```

```
  SUBNET2_ID=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$VPC2_ID" \
    --query "Subnets[0].SubnetId" \
    --output text)
  if [ "$SUBNET2_ID" = "None" ] || [ -z "$SUBNET2_ID" ]; then
    echo "Creating subnet in VPC2..."
    SUBNET2_ID=$(aws ec2 create-subnet \
      --vpc-id "$VPC2_ID" \
      --cidr-block 10.2.0.0/24 \
      --availability-zone "$AZ" \
      --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-
Subnet},{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
      --query Subnet.SubnetId \
      --output text)
    validate_aws_output "$SUBNET2_ID" "VPC2 subnet" || exit 1
    echo "Created subnet in VPC2: $SUBNET2_ID"
  else
    echo "Using existing subnet in VPC2: $SUBNET2_ID"
  fi
fi

# Get route tables for each VPC
RTB1_ID=$(aws ec2 describe-route-tables \
  --filters "Name=vpc-id,Values=$VPC1_ID" \
  --query "RouteTables[0].RouteTableId" \
  --output text)
validate_aws_output "$RTB1_ID" "VPC1 route table" || exit 1

RTB2_ID=$(aws ec2 describe-route-tables \
  --filters "Name=vpc-id,Values=$VPC2_ID" \
  --query "RouteTables[0].RouteTableId" \
  --output text)
validate_aws_output "$RTB2_ID" "VPC2 route table" || exit 1

echo "Route table for VPC1: $RTB1_ID"
echo "Route table for VPC2: $RTB2_ID"

# Step 1: Create the transit gateway
echo "Creating Transit Gateway..."
TGW_ID=$(aws ec2 create-transit-gateway \
  --description "My Transit Gateway" \
```

```
  --options
 "AmazonSideAsn=64512,AutoAcceptSharedAttachments=disable,DefaultRouteTableAssociation=en
 \
  --tag-specifications 'ResourceType=transit-
gateway,Tags=[{Key=Name,Value=MyTransitGateway},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGateway.TransitGatewayId \
  --output text)
validate_aws_output "$TGW_ID" "Transit Gateway" || exit 1
echo "Created Transit Gateway: $TGW_ID"

# Wait for the transit gateway to become available
wait_for_tgw "$TGW_ID" || exit 1

# Step 2: Attach VPCs to the transit gateway
echo "Attaching VPC1 to Transit Gateway..."
TGW_ATTACHMENT_1_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC1_ID" \
  --subnet-ids "$SUBNET1_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC1-Attachment},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)
validate_aws_output "$TGW_ATTACHMENT_1_ID" "Transit Gateway VPC1 Attachment" ||
 exit 1
echo "Created Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"

echo "Attaching VPC2 to Transit Gateway..."
TGW_ATTACHMENT_2_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC2_ID" \
  --subnet-ids "$SUBNET2_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC2-Attachment},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)
validate_aws_output "$TGW_ATTACHMENT_2_ID" "Transit Gateway VPC2 Attachment" ||
 exit 1
echo "Created Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"

# Wait for the attachments to become available
```

```
wait_for_tgw_attachment "$TGW_ATTACHMENT_1_ID" || exit 1
wait_for_tgw_attachment "$TGW_ATTACHMENT_2_ID" || exit 1

# Step 3: Add routes between the transit gateway and VPCs
echo "Adding route from VPC1 to VPC2 via Transit Gateway..."
validate_cidr "10.2.0.0/16" || exit 1
aws ec2 create-route \
  --route-table-id "$RTB1_ID" \
  --destination-cidr-block 10.2.0.0/16 \
  --transit-gateway-id "$TGW_ID" || exit 1

echo "Adding route from VPC2 to VPC1 via Transit Gateway..."
validate_cidr "10.1.0.0/16" || exit 1
aws ec2 create-route \
  --route-table-id "$RTB2_ID" \
  --destination-cidr-block 10.1.0.0/16 \
  --transit-gateway-id "$TGW_ID" || exit 1

echo "Routes added successfully"

# Step 4: Display information for testing
echo ""
echo "=== Transit Gateway Setup Complete ==="
echo "Transit Gateway ID: $TGW_ID"
echo "VPC1 ID: $VPC1_ID"
echo "VPC2 ID: $VPC2_ID"
echo ""
echo "To test connectivity:"
echo "1. Launch an EC2 instance in each VPC"
echo "2. Configure security groups to allow ICMP traffic"
echo "3. Connect to one instance and ping the other instance's private IP"
echo ""

echo "Viewing created resources..."

echo ""
echo "=== Resources Created ==="
echo "Transit Gateway: $TGW_ID"
echo "VPC1: $VPC1_ID"
echo "VPC2: $VPC2_ID"
echo "Subnet in VPC1: $SUBNET1_ID"
echo "Subnet in VPC2: $SUBNET2_ID"
echo "Transit Gateway Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
echo "Transit Gateway Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
```

```
echo ""

echo "Starting cleanup..."

# Delete routes
echo "Deleting routes..."
aws ec2 delete-route --route-table-id "$RTB1_ID" --destination-cidr-block
 10.2.0.0/16 || true
aws ec2 delete-route --route-table-id "$RTB2_ID" --destination-cidr-block
 10.1.0.0/16 || true

# Delete transit gateway attachments
echo "Deleting Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
aws ec2 delete-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id "$TGW_ATTACHMENT_1_ID" || true

echo "Deleting Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
aws ec2 delete-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id "$TGW_ATTACHMENT_2_ID" || true

# Wait for attachments to be deleted
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true

# Delete transit gateway
echo "Deleting Transit Gateway: $TGW_ID"
aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" || true

echo "Cleanup completed successfully"

echo "Tutorial completed. See $LOG_FILE for detailed logs."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateRoute
  - CreateSubnet
  - CreateTransitGateway
  - CreateTransitGatewayVpcAttachment
  - CreateVpc
  - DeleteRoute
  - DeleteTransitGateway

- DeleteTransitGatewayVpcAttachment
- DescribeAvailabilityZones
- DescribeRouteTables
- DescribeSubnets
- DescribeTransitGatewayAttachments
- DescribeTransitGatewayVpcAttachments
- DescribeTransitGateways
- DescribeVpcs

Introdução à transmissão de transporte de vídeo

O exemplo de código a seguir mostra como:

- Verifique o acesso ao Elemental MediaConnect
- Criar um fluxo.
- Adicionar uma saída.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Elemental MediaConnect Getting Started Tutorial Script
# This script creates a MediaConnect flow, adds an output, grants an entitlement,
# and then cleans up the resources.

set -euo pipefail
```

```
# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="mediaconnect-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Elemental MediaConnect tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Tags for all resources
TAGS_ARRAY=("Key=project,Value=doc-smith" "Key=tutorial,Value=aws-elemental-
mediaconnect-gs")

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI is available
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        handle_error "AWS CLI is not installed or not in PATH"
    fi

    # Security: Verify AWS CLI version is recent
    local aws_version
    aws_version=$(aws --version 2>&1 | head -1)
    echo "AWS CLI version: $aws_version"

    if ! aws sts get-caller-identity &> /dev/null; then
        handle_error "AWS credentials are not configured or invalid"
    fi

    # Security: Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null)
    if [ -z "$account_id" ]; then
```

```
        handle_error "Failed to retrieve AWS account ID"
    fi
    echo "AWS Account ID: $account_id"
}

# Function to safely extract JSON values using jq (preferred) or fallback
extract_json_value() {
    local json_output="$1"
    local key="$2"

    if [ -z "$json_output" ]; then
        return 1
    fi

    # Security: Use jq if available for safer JSON parsing
    if command -v jq &> /dev/null; then
        echo "$json_output" | jq -r ".${key} // empty" 2>/dev/null || return 1
    else
        # Fallback with additional validation
        if ! echo "$json_output" | grep -q "\"$key\""; then
            return 1
        fi
        echo "$json_output" | grep -o "\"$key\": \"[^\"]*" | head -1 | cut -d'"'
 -f4
    fi
}

# Function to tag MediaConnect resource
tag_mediaconnect_resource() {
    local resource_arn="$1"
    echo "Tagging resource: $resource_arn"

    for tag in "${TAGS_ARRAY[@]}"; do
        if ! aws mediaconnect tag-resource --resource-arn "$resource_arn" --tags
 "$tag" 2>&1; then
            echo "WARNING: Failed to apply tag $tag to resource"
        fi
    done
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."
```

```
    if [ -n "${FLOW_ARN:-}" ]; then
        # Security: Validate ARN format before using it
        if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-
zA-Z0-9:-]+$ ]]; then
            echo "WARNING: Invalid Flow ARN format, skipping cleanup: $FLOW_ARN"
            return 1
        fi

        # Check flow status before attempting to stop
        echo "Checking flow status..."
        local flow_status_output
        if flow_status_output=$(aws mediaconnect describe-flow --flow-arn
 "$FLOW_ARN" --query "Flow.Status" --output text 2>&1); then
            echo "Current flow status: $flow_status_output"

            if [ "$flow_status_output" == "ACTIVE" ] || [ "$flow_status_output"
 == "UPDATING" ]; then
                echo "Stopping flow: $FLOW_ARN"
                if aws mediaconnect stop-flow --flow-arn "$FLOW_ARN" 2>&1; then
                    # Wait for flow to stop before deleting
                    echo "Waiting for flow to stop..."
                    sleep 10
                else
                    echo "WARNING: Failed to stop flow. Attempting to delete
 anyway..."
                fi
            else
                echo "Flow is not in ACTIVE or UPDATING state, skipping stop
 operation."
            fi

            # Delete the flow
            echo "Deleting flow: $FLOW_ARN"
            if aws mediaconnect delete-flow --flow-arn "$FLOW_ARN" 2>&1; then
                echo "Flow deleted successfully"
            else
                echo "WARNING: Failed to delete flow. You may need to manually
 delete it from the AWS console."
            fi
        else
            echo "WARNING: Could not check flow status"
        fi
    fi
}
```

```
# Set trap to cleanup on script exit
trap cleanup_resources EXIT

# Validate AWS CLI setup
validate_aws_cli

# Get the current AWS region
aws_region=""
if aws_region=$(aws configure get region 2>/dev/null); then
    if [ -z "$aws_region" ]; then
        handle_error "Failed to get AWS region. Please make sure AWS CLI is
 configured."
    fi
else
    handle_error "Failed to retrieve AWS region configuration"
fi

# Security: Validate region format
if [[ ! "$aws_region" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $aws_region"
fi

AWS_REGION="$aws_region"
echo "Using AWS Region: $AWS_REGION"

# Get available availability zones in the current region
echo "Getting available availability zones in region $AWS_REGION..."
az_output=""
if az_output=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --query
 "AvailabilityZones[0].ZoneName" --output text 2>&1); then
    AVAILABILITY_ZONE="$az_output"
    if [ -z "$AVAILABILITY_ZONE" ]; then
        handle_error "Failed to retrieve availability zones"
    fi

    # Security: Validate AZ format
    if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9][a-z]$ ]]; then
        handle_error "Invalid availability zone format: $AVAILABILITY_ZONE"
    fi

    echo "Using availability zone: $AVAILABILITY_ZONE"
else
    handle_error "Failed to get availability zones"
```

```
fi

# Generate a unique suffix for resource names
SUFFIX=$(date +%s | cut -c 6-10)
FLOW_NAME="AwardsNYCShow-${SUFFIX}"
SOURCE_NAME="AwardsNYCSource-${SUFFIX}"
OUTPUT_NAME="AwardsNYCOutput-${SUFFIX}"
ENTITLEMENT_NAME="PhillyTeam-${SUFFIX}"

echo "Using the following resource names:"
echo "Flow name: $FLOW_NAME"
echo "Source name: $SOURCE_NAME"
echo "Output name: $OUTPUT_NAME"
echo "Entitlement name: $ENTITLEMENT_NAME"

# Step 1: Verify access to MediaConnect
echo "Step 1: Verifying access to AWS Elemental MediaConnect..."
list_flows_output=""
if list_flows_output=$(aws mediaconnect list-flows 2>&1); then
    echo "$list_flows_output"
else
    handle_error "Failed to list flows. Please check your AWS credentials and
 permissions."
fi

# Step 2: Create a flow
echo "Step 2: Creating a flow..."
create_flow_output=""
if create_flow_output=$(aws mediaconnect create-flow \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "$FLOW_NAME" \
    --source "Name=$SOURCE_NAME,Protocol=zixi-
push,WhitelistCidr=10.24.34.0/23,StreamId=ZixiAwardsNYCFeed" 2>&1); then
    echo "$create_flow_output"
else
    handle_error "Failed to create flow"
fi

# Extract the flow ARN from the output
FLOW_ARN=$(echo "$create_flow_output" | jq -r '.Flow.FlowArn // empty' 2>/dev/
null)
if [ -z "$FLOW_ARN" ]; then
    FLOW_ARN=$(echo "$create_flow_output" | grep -o '"FlowArn": "[^"]*' | head -1
 | cut -d'"' -f4)
```

```
fi
if [ -z "$FLOW_ARN" ]; then
    handle_error "Failed to extract flow ARN from output"
fi
echo "Flow ARN: $FLOW_ARN"

# Validate flow ARN format
if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-
Z0-9:-]+$ ]]; then
    handle_error "Invalid Flow ARN format: $FLOW_ARN"
fi

# Tag the flow
tag_mediaconnect_resource "$FLOW_ARN"

# Step 3: Add an output
echo "Step 3: Adding an output to the flow..."
add_output_output=""
if add_output_output=$(aws mediaconnect add-flow-outputs \
    --flow-arn "$FLOW_ARN" \
    --outputs "Name=$OUTPUT_NAME,Protocol=zixi-
push,Destination=198.51.100.11,Port=1024,StreamId=ZixiAwardsOutput" 2>&1); then
    echo "$add_output_output"
else
    handle_error "Failed to add output to flow"
fi

# Extract the output ARN
output_arn=""
output_arn=$(echo "$add_output_output" | jq -r ".Output.OutputArn // empty" 2>/
dev/null)
if [ -z "$output_arn" ]; then output_arn=$(echo "$add_output_output" | grep -o
 '"OutputArn": "[^"]*' | head -1 | cut -d'"' -f4); fi
if [ -z "$output_arn" ]; then
    echo "WARNING: Failed to extract output ARN from output"
else
    OUTPUT_ARN="$output_arn"
    echo "Output ARN: $OUTPUT_ARN"
    tag_mediaconnect_resource "$OUTPUT_ARN"
fi

# Step 4: Grant an entitlement
echo "Step 4: Granting an entitlement..."
grant_entitlement_output=""
```

```
if grant_entitlement_output=$(aws mediaconnect grant-flow-entitlements \
    --flow-arn "$FLOW_ARN" \
    --entitlements "Name=$ENTITLEMENT_NAME,Subscribers=222233334444" 2>&1); then
    echo "$grant_entitlement_output"
else
    handle_error "Failed to grant entitlement"
fi

# Extract the entitlement ARN
entitlement_arn=""
entitlement_arn=$(echo "$grant_entitlement_output" | jq -r
 '.Entitlement.EntitlementArn // empty' 2>/dev/null)
if [ -z "$entitlement_arn" ]; then
    entitlement_arn=$(echo "$grant_entitlement_output" | grep -o
 '"EntitlementArn": "[^"]*' | head -1 | cut -d'"' -f4)
fi
if [ -z "$entitlement_arn" ]; then
    echo "WARNING: Failed to extract entitlement ARN from output"
else
    ENTITLEMENT_ARN="$entitlement_arn"
    echo "Entitlement ARN: $ENTITLEMENT_ARN"
    tag_mediaconnect_resource "$ENTITLEMENT_ARN"
fi

# Step 5: List entitlements to share with affiliates
echo "Step 5: Listing entitlements for the flow..."
describe_flow_output=""
if describe_flow_output=$(aws mediaconnect describe-flow --flow-arn "$FLOW_ARN"
 --query "Flow.Entitlements" 2>&1); then
    echo "Entitlements for the flow:"
    echo "$describe_flow_output"
else
    handle_error "Failed to describe flow"
fi

# Display information to share with affiliates
echo ""
echo "Information to share with your Philadelphia affiliate:"
echo "Entitlement ARN: ${ENTITLEMENT_ARN:-N/A}"
echo "AWS Region: $AWS_REGION"

# Display resource summary
echo ""
echo "============================================"
```

```
echo "RESOURCE SUMMARY"
echo "=========================================="
echo "The following resources were created:"
echo "1. Flow: $FLOW_NAME (ARN: $FLOW_ARN)"
echo "2. Output: $OUTPUT_NAME (ARN: ${OUTPUT_ARN:-N/A})"
echo "3. Entitlement: $ENTITLEMENT_NAME (ARN: ${ENTITLEMENT_ARN:-N/A})"
echo ""
echo "=========================================="
echo "CLEANUP CONFIRMATION"
echo "=========================================="
echo "Automatically cleaning up all created resources..."

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddFlowOutputs
  - CreateFlow
  - DeleteFlow
  - DescribeAvailabilityZones
  - DescribeFlow
  - GrantFlowEntitlements
  - ListFlows
  - StopFlow

Introdução às máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um par de chaves.
- Criar um grupo de segurança.
- Interromper e iniciar uma instância.
- Testar a persistência de um IP elástico.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EC2 Basics Tutorial Script - Revised
# This script demonstrates the basics of working with EC2 instances using AWS CLI
# Updated to use Amazon Linux 2023 and enhanced security settings

# Set up logging
LOG_FILE="ec2_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to log messages
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

# Function to handle errors
handle_error() {
  log "ERROR: $1"
  log "Cleaning up resources..."
  cleanup
  exit 1
}

# Function to clean up resources
cleanup() {
  log "Resources created:"

  if [ -n "$ASSOCIATION_ID" ]; then
    log "- Elastic IP Association: $ASSOCIATION_ID"
  fi

  if [ -n "$ALLOCATION_ID" ]; then
```

```
    log "- Elastic IP Allocation: $ALLOCATION_ID (IP: $ELASTIC_IP)"
  fi

  if [ -n "$INSTANCE_ID" ]; then
    log "- EC2 Instance: $INSTANCE_ID"
  fi

  if [ -n "$SECURITY_GROUP_ID" ]; then
    log "- Security Group: $SECURITY_GROUP_ID"
  fi

  if [ -n "$KEY_NAME" ]; then
    log "- Key Pair: $KEY_NAME (File: $KEY_FILE)"
  fi

  REPLY=y

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    log "Starting cleanup..."

    # Track cleanup failures
    CLEANUP_FAILURES=0

    # Disassociate Elastic IP if it exists
    if [ -n "$ASSOCIATION_ID" ]; then
      log "Disassociating Elastic IP..."
      if ! aws ec2 disassociate-address --association-id "$ASSOCIATION_ID"; then
        log "Failed to disassociate Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Release Elastic IP if it exists
    if [ -n "$ALLOCATION_ID" ]; then
      log "Releasing Elastic IP..."
      if ! aws ec2 release-address --allocation-id "$ALLOCATION_ID"; then
        log "Failed to release Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Terminate instance if it exists
    if [ -n "$INSTANCE_ID" ]; then
      log "Terminating instance $INSTANCE_ID..."
```

```
      if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" > /dev/null;
then
        log "Failed to terminate instance"
        ((CLEANUP_FAILURES++))
      else
        log "Waiting for instance to terminate..."
        if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"; then
          log "Failed while waiting for instance to terminate"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Delete security group if it exists
    if [ -n "$SECURITY_GROUP_ID" ]; then
      log "Deleting security group..."
      if ! aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"; then
        log "Failed to delete security group"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ]; then
      log "Deleting key pair..."
      if ! aws ec2 delete-key-pair --key-name "$KEY_NAME"; then
        log "Failed to delete key pair"
        ((CLEANUP_FAILURES++))
      fi

      # Remove key file
      if [ -f "$KEY_FILE" ]; then
        log "Removing key file..."
        if ! rm -f "$KEY_FILE"; then
          log "Failed to remove key file"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Report cleanup status
    if [ $CLEANUP_FAILURES -eq 0 ]; then
      log "Cleanup completed successfully."
    else
```

```
      log "WARNING: Cleanup completed with $CLEANUP_FAILURES failures. Some
 resources may not have been deleted properly."
    fi
  else
    log "Resources were not deleted."
  fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
KEY_NAME="ec2-tutorial-key-$RANDOM_ID"
SG_NAME="ec2-tutorial-sg-$RANDOM_ID"

# Create a directory for the key file
KEY_DIR=$(mktemp -d)
KEY_FILE="$KEY_DIR/$KEY_NAME.pem"

log "Starting EC2 basics tutorial script"
log "Random identifier: $RANDOM_ID"
log "Key name: $KEY_NAME"
log "Security group name: $SG_NAME"

# Step 1: Create a key pair
log "Creating key pair..."
KEY_RESULT=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial'
 --output text)

if [ $? -ne 0 ] || [ -z "$KEY_RESULT" ]; then
  handle_error "Failed to create key pair"
fi

echo "$KEY_RESULT" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
log "Created key pair and saved to $KEY_FILE"

# Step 2: Create a security group
log "Creating security group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --group-name "$SG_NAME" \
  --description "Security group for EC2 tutorial" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query "GroupId" \
  --output text)
```

```
if [ $? -ne 0 ] || [ -z "$SECURITY_GROUP_ID" ]; then
  handle_error "Failed to create security group"
fi

log "Created security group: $SECURITY_GROUP_ID"

# Get current public IP address for SSH access
MY_IP=$(curl -s http://checkip.amazonaws.com)
if [ $? -ne 0 ] || [ -z "$MY_IP" ]; then
  handle_error "Failed to get current IP address"
fi

log "Adding SSH ingress rule for IP $MY_IP..."
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr "$MY_IP/32" > /dev/null

if [ $? -ne 0 ]; then
  handle_error "Failed to add security group ingress rule"
fi

log "Added SSH ingress rule for IP $MY_IP"

# Step 3: Find an Amazon Linux 2023 AMI (updated from AL2)
log "Finding latest Amazon Linux 2023 AMI..."
AMI_ID=$(aws ssm get-parameters-by-path \
  --path "/aws/service/ami-amazon-linux-latest" \
  --query "Parameters[?contains(Name, 'al2023-ami-kernel-default-x86_64')].Value"
 \
  --output text | head -1)

if [ $? -ne 0 ] || [ -z "$AMI_ID" ]; then
  handle_error "Failed to find Amazon Linux 2023 AMI"
fi

log "Selected AMI: $AMI_ID"

# Get the architecture of the AMI
log "Getting AMI architecture..."
AMI_ARCH=$(aws ec2 describe-images \
  --image-ids "$AMI_ID" \
```

```
  --query "Images[0].Architecture" \
  --output text)

if [ $? -ne 0 ] || [ -z "$AMI_ARCH" ]; then
  handle_error "Failed to get AMI architecture"
fi

log "AMI architecture: $AMI_ARCH"

# Find a compatible instance type
log "Finding compatible instance type..."
# Directly use t2.micro for simplicity
INSTANCE_TYPE="t2.micro"
log "Using instance type: $INSTANCE_TYPE"

# Step 4: Launch an EC2 instance with enhanced security
log "Launching EC2 instance with IMDSv2 and encryption enabled..."
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --metadata-options "HttpTokens=required,HttpEndpoint=enabled" \
  --block-device-mappings "DeviceName=/dev/xvda,Ebs={Encrypted=true}" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]'
 'ResourceType=volume,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' \
  --count 1 \
  --query 'Instances[0].InstanceId' \
  --output text)

if [ $? -ne 0 ] || [ -z "$INSTANCE_ID" ]; then
  handle_error "Failed to launch EC2 instance"
fi

log "Launched instance $INSTANCE_ID. Waiting for it to start..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi
```

```
# Get instance details
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].
{ID:InstanceId,Type:InstanceType,State:State.Name,PublicIP:PublicIpAddress}' \
  --output json)

if [ $? -ne 0 ]; then
  handle_error "Failed to get instance details"
fi

log "Instance details: $INSTANCE_DETAILS"

# Get the public IP address
PUBLIC_IP=$(echo "$INSTANCE_DETAILS" | grep -oP '"PublicIP": "\K[^"]+')
if [ -z "$PUBLIC_IP" ]; then
  handle_error "Failed to get instance public IP"
fi

log "Instance public IP: $PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$PUBLIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 6: Stop and Start the Instance
log "Stopping instance $INSTANCE_ID..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi
```

```
log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get the new public IP address
NEW_PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$NEW_PUBLIC_IP" ]; then
  handle_error "Failed to get new public IP"
fi

log "Instance restarted with new public IP: $NEW_PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$NEW_PUBLIC_IP"

# Step 7: Allocate and Associate an Elastic IP Address
log "Allocating Elastic IP address..."
ALLOCATION_RESULT=$(aws ec2 allocate-address \
  --domain vpc \
  --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query '[PublicIp,AllocationId]' \
  --output text)

if [ $? -ne 0 ] || [ -z "$ALLOCATION_RESULT" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

ELASTIC_IP=$(echo "$ALLOCATION_RESULT" | awk '{print $1}')
ALLOCATION_ID=$(echo "$ALLOCATION_RESULT" | awk '{print $2}')

log "Allocated Elastic IP: $ELASTIC_IP with ID: $ALLOCATION_ID"

log "Associating Elastic IP with instance..."
ASSOCIATION_ID=$(aws ec2 associate-address \
  --instance-id "$INSTANCE_ID" \
  --allocation-id "$ALLOCATION_ID" \
  --query "AssociationId" \
  --output text)
```

```
if [ $? -ne 0 ] || [ -z "$ASSOCIATION_ID" ]; then
  handle_error "Failed to associate Elastic IP"
fi

log "Associated Elastic IP with instance. Association ID: $ASSOCIATION_ID"
log "To connect to your instance using the Elastic IP, run: ssh -i $KEY_FILE ec2-
user@$ELASTIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 8: Test the Elastic IP by Stopping and Starting the Instance
log "Stopping instance $INSTANCE_ID to test Elastic IP persistence..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Verify the Elastic IP is still associated
CURRENT_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$CURRENT_IP" ]; then
  handle_error "Failed to get current public IP"
```

```
fi

log "Current public IP address: $CURRENT_IP"
log "Elastic IP address: $ELASTIC_IP"

if [ "$CURRENT_IP" = "$ELASTIC_IP" ]; then
  log "Success! The Elastic IP is still associated with your instance."
else
  log "Something went wrong. The Elastic IP is not associated with your
 instance."
fi

log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$ELASTIC_IP"

# Step 9: Clean up resources
log "Tutorial completed successfully!"
cleanup

exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - DisassociateAddress
  - GetParametersByPath
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances

- Wait

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
```

```
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
```

```
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()
```

```
echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
```

```
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
```

```
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"
```

```
# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
```

```
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
```

```
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
```

```
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
```

```
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
```

```
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
```

```
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"
```

```
# Terminate EC2 instance
echo "Terminating EC2 instance $INSTANCE_ID..."
TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
echo "Waiting for instance to terminate..."
aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

# Clean up IAM resources
echo "Removing role from instance profile..."
REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

echo "Deleting instance profile..."
DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

echo "Deleting FIS role policy..."
DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME")
check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

echo "Detaching policy from EC2 role..."
DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

echo "Deleting FIS role..."
DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

echo "Deleting EC2 role..."
DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
  --role-name "$EC2_ROLE_NAME")
check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

# Clean up temporary files
echo "Cleaning up temporary files..."
```

```
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy

- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Comece a usar o gerenciamento de endereços IP usando a CLI

O exemplo de código a seguir mostra como:

- Definir e configurar o Gerenciador de endereços IP (IPAM) da Amazon VPC usando a CLI.
- Criar um IPAM com regiões de operação (p. ex., us-east-1 e us-west-2).
- Recuperar o ID do escopo privado do IPAM.
- Criar uma estrutura hierárquica de grupos de IPv4 (grupos de nível superior, regionais e de desenvolvimento).
- Provisione blocos CIDR para cada pool (por exemplo, 10.0.0). 0/8, 10.0.0. 0/16, 10.0.0. 0/24).
- Criar uma VPC usando um CIDR alocado em um grupo do IPAM.
- Verificar as alocações do grupo do IPAM e a criação da VPC.
- Solucionar problemas comuns, como erros de permissão, falhas na alocação de CIDR e violações de dependências.
- Limpar os recursos do IPAM (VPC, grupos, CIDRs e IPAM) para evitar cobranças desnecessárias.
- Explorar as próximas etapas para recursos avançados do IPAM.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# IPAM Getting Started CLI Script - Version 7
# This script creates an IPAM, creates a hierarchy of IP address pools, and
 allocates a CIDR to a VPC
# Fixed to correctly identify the private scope ID, wait for resources to be
 available, add locale to development pool,
# use the correct parameter names for VPC creation, and wait for CIDR
 provisioning to complete

# Set up logging
LOG_FILE="ipam_script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting IPAM setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED:"
    echo "============================================"

    if [ -n "$VPC_ID" ]; then
        echo "VPC: $VPC_ID"
    fi

    if [ -n "$DEV_POOL_ID" ]; then
        echo "Development Pool: $DEV_POOL_ID"
    fi

    if [ -n "$REGIONAL_POOL_ID" ]; then
        echo "Regional Pool: $REGIONAL_POOL_ID"
```

```
    fi

    if [ -n "$TOP_POOL_ID" ]; then
        echo "Top-level Pool: $TOP_POOL_ID"
    fi

    if [ -n "$IPAM_ID" ]; then
        echo "IPAM: $IPAM_ID"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."

        # Delete resources in reverse order of creation to handle dependencies

        if [ -n "$VPC_ID" ]; then
            echo "Deleting VPC: $VPC_ID"
            aws ec2 delete-vpc --vpc-id "$VPC_ID" || echo "Failed to delete VPC"
            echo "Waiting for VPC to be deleted..."
            sleep 10
        fi

        if [ -n "$DEV_POOL_ID" ]; then
            echo "Deleting Development Pool: $DEV_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$DEV_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Development Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$DEV_POOL_ID"
 --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$DEV_POOL_ID" || echo
 "Failed to delete Development Pool"
            echo "Waiting for Development Pool to be deleted..."
            sleep 10
```

```
        fi

        if [ -n "$REGIONAL_POOL_ID" ]; then
            echo "Deleting Regional Pool: $REGIONAL_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id
 "$REGIONAL_POOL_ID" --query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Regional Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id
 "$REGIONAL_POOL_ID" --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$REGIONAL_POOL_ID" || echo
 "Failed to delete Regional Pool"
            echo "Waiting for Regional Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$TOP_POOL_ID" ]; then
            echo "Deleting Top-level Pool: $TOP_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$TOP_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Top-level Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$TOP_POOL_ID"
 --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$TOP_POOL_ID" || echo
 "Failed to delete Top-level Pool"
            echo "Waiting for Top-level Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$IPAM_ID" ]; then
            echo "Deleting IPAM: $IPAM_ID"
            aws ec2 delete-ipam --ipam-id "$IPAM_ID" || echo "Failed to delete
 IPAM"
        fi

        echo "Cleanup completed."
    else
```

```
        echo "Cleanup skipped. Resources will remain in your account."
    fi
}

# Function to wait for a pool to be in the 'create-complete' state
wait_for_pool() {
    local pool_id=$1
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for pool $pool_id to be available..."

    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 describe-ipam-pools --ipam-pool-ids "$pool_id" --query
 'IpamPools[0].State' --output text)

        if [ "$state" = "create-complete" ]; then
            echo "Pool $pool_id is now available (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: Pool $pool_id is in state: $state.
 Waiting..."
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for pool $pool_id to be available"
    return 1
}

# Function to wait for a CIDR to be fully provisioned
wait_for_cidr_provisioning() {
    local pool_id=$1
    local cidr=$2
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for CIDR $cidr to be fully provisioned in pool $pool_id..."

    while [ $attempt -le $max_attempts ]; do
```

```
        state=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$pool_id" --query
 "IpamPoolCidrs[?Cidr=='$cidr'].State" --output text)

        if [ "$state" = "provisioned" ]; then
            echo "CIDR $cidr is now fully provisioned (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: CIDR $cidr is in state: $state.
 Waiting..."
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for CIDR $cidr to be provisioned"
    return 1
}

# Step 1: Create an IPAM
echo "Creating IPAM..."
IPAM_RESULT=$(aws ec2 create-ipam \
    --description "My IPAM" \
    --operating-regions RegionName=us-east-1 RegionName=us-west-2 \
    --tag-specifications 'ResourceType=ipam,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create IPAM"
fi

IPAM_ID=$(echo "$IPAM_RESULT" | grep -o '"IpamId": "[^"]*' | cut -d'"' -f4)
echo "IPAM created with ID: $IPAM_ID"

# Wait for IPAM to be created and available
echo "Waiting for IPAM to be available..."
sleep 20

# Step 2: Get the IPAM Scope ID - FIXED to correctly identify the private scope
echo "Getting IPAM Scope ID..."
SCOPE_RESULT=$(aws ec2 describe-ipams --ipam-id "$IPAM_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to get IPAM details"
fi
```

```
# Extract the private scope ID directly from the IPAM details
PRIVATE_SCOPE_ID=$(echo "$SCOPE_RESULT" | grep -o '"PrivateDefaultScopeId":
 "[^"]*' | cut -d'"' -f4)
echo "Private Scope ID: $PRIVATE_SCOPE_ID"

if [ -z "$PRIVATE_SCOPE_ID" ]; then
    handle_error "Failed to get Private Scope ID"
fi

# Step 3: Create a Top-Level IPv4 Pool
echo "Creating Top-level IPv4 Pool..."
TOP_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --address-family ipv4 \
    --description "Top-level pool" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Top-level Pool"
fi

TOP_POOL_ID=$(echo "$TOP_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -
d'"' -f4)
echo "Top-level Pool created with ID: $TOP_POOL_ID"

# Wait for the top-level pool to be available
if ! wait_for_pool "$TOP_POOL_ID"; then
    handle_error "Top-level Pool did not become available in time"
fi

# Provision CIDR to the top-level pool
echo "Provisioning CIDR to Top-level Pool..."
TOP_POOL_CIDR="10.0.0.0/8"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$TOP_POOL_ID" \
    --cidr "$TOP_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Top-level Pool"
fi

echo "$PROVISION_RESULT"
```

```
# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$TOP_POOL_ID" "$TOP_POOL_CIDR"; then
    handle_error "CIDR provisioning to Top-level Pool did not complete in time"
fi

# Step 4: Create a Regional IPv4 Pool
echo "Creating Regional IPv4 Pool..."
REGIONAL_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$TOP_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Regional pool in us-east-1" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Regional Pool"
fi

REGIONAL_POOL_ID=$(echo "$REGIONAL_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*'
 | cut -d'"' -f4)
echo "Regional Pool created with ID: $REGIONAL_POOL_ID"

# Wait for the regional pool to be available
if ! wait_for_pool "$REGIONAL_POOL_ID"; then
    handle_error "Regional Pool did not become available in time"
fi

# Provision CIDR to the regional pool
echo "Provisioning CIDR to Regional Pool..."
REGIONAL_POOL_CIDR="10.0.0.0/16"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$REGIONAL_POOL_ID" \
    --cidr "$REGIONAL_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Regional Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
```

```
if ! wait_for_cidr_provisioning "$REGIONAL_POOL_ID" "$REGIONAL_POOL_CIDR"; then
    handle_error "CIDR provisioning to Regional Pool did not complete in time"
fi

# Step 5: Create a Development IPv4 Pool - FIXED to include locale
echo "Creating Development IPv4 Pool..."
DEV_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$REGIONAL_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Development pool" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Development Pool"
fi

DEV_POOL_ID=$(echo "$DEV_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -
d'"' -f4)
echo "Development Pool created with ID: $DEV_POOL_ID"

# Wait for the development pool to be available
if ! wait_for_pool "$DEV_POOL_ID"; then
    handle_error "Development Pool did not become available in time"
fi

# Provision CIDR to the development pool
echo "Provisioning CIDR to Development Pool..."
DEV_POOL_CIDR="10.0.0.0/24"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$DEV_POOL_ID" \
    --cidr "$DEV_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Development Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$DEV_POOL_ID" "$DEV_POOL_CIDR"; then
    handle_error "CIDR provisioning to Development Pool did not complete in time"
```

```
fi

# Step 6: Create a VPC Using an IPAM Pool CIDR - FIXED to use the correct
 parameter names and a smaller netmask length
echo "Creating VPC using IPAM Pool CIDR..."
VPC_RESULT=$(aws ec2 create-vpc \
    --ipv4-ipam-pool-id "$DEV_POOL_ID" \
    --ipv4-netmask-length 26 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=IPAM-VPC},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create VPC"
fi

VPC_ID=$(echo "$VPC_RESULT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID"

# Step 7: Verify the IPAM Pool Allocation
echo "Verifying IPAM Pool Allocation..."
ALLOCATION_RESULT=$(aws ec2 get-ipam-pool-allocations \
    --ipam-pool-id "$DEV_POOL_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to verify IPAM Pool Allocation"
fi

echo "IPAM Pool Allocation verified:"
echo "$ALLOCATION_RESULT" | grep -A 5 "Allocations"

echo ""
echo "IPAM setup completed successfully!"
echo ""

# Prompt for cleanup
cleanup_resources

echo "Script completed at $(date)"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateIpam

- CreateIpamPool
- CreateVpc
- DeleteIpam
- DeleteIpamPool
- DeleteVpc
- DeprovisionIpamPoolCidr
- DescribeIpamPools
- DescribeIpams
- DescribeVpcs
- GetIpamPoolAllocations
- GetIpamPoolCidrs
- ProvisionIpamPoolCidr

Crie uma rede privada virtual com sub-redes privadas e gateways de conversão de endereços de rede usando a CLI

O exemplo de código a seguir mostra como:

- Criar uma VPC com sub-redes privadas e gateways NAT usando a CLI.
- Configurar os componentes necessários, como VPC, sub-redes, tabelas de rotas e gateways NAT.
- Configurar grupos de segurança e perfis do IAM para acesso e segurança adequados.
- Usar os comandos da CLI para automatizar a criação e a configuração desses recursos.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC with Private Subnets and NAT Gateways (IMDSv2 Compliant Version)
# This script creates a VPC with public and private subnets in two Availability
 Zones,
# NAT gateways, an internet gateway, route tables, a VPC endpoint for S3,
# security groups, a launch template, an Auto Scaling group, and an Application
 Load Balancer.

# Set up logging
LOG_FILE="vpc-private-subnets-nat.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Cleanup function to delete all created resources
cleanup_resources() {
  echo "Cleaning up resources..."

  # Delete Auto Scaling group if it exists
  if [ -n "${ASG_NAME:-}" ]; then
    echo "Deleting Auto Scaling group: $ASG_NAME"
    aws autoscaling delete-auto-scaling-group --auto-scaling-group-name
 "$ASG_NAME" --force-delete
    echo "Waiting for Auto Scaling group to be deleted..."
    aws autoscaling wait auto-scaling-groups-deleted --auto-scaling-group-names
 "$ASG_NAME"
  fi

  # Delete load balancer if it exists
  if [ -n "${LB_ARN:-}" ]; then
    echo "Deleting load balancer: $LB_ARN"
    aws elbv2 delete-load-balancer --load-balancer-arn "$LB_ARN"
    # Wait for load balancer to be deleted
    sleep 30
  fi

  # Delete target group if it exists
  if [ -n "${TARGET_GROUP_ARN:-}" ]; then
    echo "Deleting target group: $TARGET_GROUP_ARN"
    aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN"
  fi

  # Delete launch template if it exists
  if [ -n "${LAUNCH_TEMPLATE_NAME:-}" ]; then
    echo "Deleting launch template: $LAUNCH_TEMPLATE_NAME"
```

```
  aws ec2 delete-launch-template --launch-template-name "$LAUNCH_TEMPLATE_NAME"
fi

# Delete NAT Gateways if they exist
if [ -n "${NAT_GW1_ID:-}" ]; then
  echo "Deleting NAT Gateway 1: $NAT_GW1_ID"
  aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW1_ID"
fi

if [ -n "${NAT_GW2_ID:-}" ]; then
  echo "Deleting NAT Gateway 2: $NAT_GW2_ID"
  aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW2_ID"
fi

# Wait for NAT Gateways to be deleted
if [ -n "${NAT_GW1_ID:-}" ] || [ -n "${NAT_GW2_ID:-}" ]; then
  echo "Waiting for NAT Gateways to be deleted..."
  sleep 60
fi

# Release Elastic IPs if they exist
if [ -n "${EIP1_ALLOC_ID:-}" ]; then
  echo "Releasing Elastic IP 1: $EIP1_ALLOC_ID"
  aws ec2 release-address --allocation-id "$EIP1_ALLOC_ID"
fi

if [ -n "${EIP2_ALLOC_ID:-}" ]; then
  echo "Releasing Elastic IP 2: $EIP2_ALLOC_ID"
  aws ec2 release-address --allocation-id "$EIP2_ALLOC_ID"
fi

# Delete VPC endpoint if it exists
if [ -n "${VPC_ENDPOINT_ID:-}" ]; then
  echo "Deleting VPC endpoint: $VPC_ENDPOINT_ID"
  aws ec2 delete-vpc-endpoints --vpc-endpoint-ids "$VPC_ENDPOINT_ID"
fi

# Delete security groups if they exist
if [ -n "${APP_SG_ID:-}" ]; then
  echo "Deleting application security group: $APP_SG_ID"
  aws ec2 delete-security-group --group-id "$APP_SG_ID"
fi

if [ -n "${LB_SG_ID:-}" ]; then
```

```
    echo "Deleting load balancer security group: $LB_SG_ID"
    aws ec2 delete-security-group --group-id "$LB_SG_ID"
  fi

  # Detach and delete Internet Gateway if it exists
  if [ -n "${IGW_ID:-}" ] && [ -n "${VPC_ID:-}" ]; then
    echo "Detaching Internet Gateway: $IGW_ID from VPC: $VPC_ID"
    aws ec2 detach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
 "$VPC_ID"
    echo "Deleting Internet Gateway: $IGW_ID"
    aws ec2 delete-internet-gateway --internet-gateway-id "$IGW_ID"
  fi

  # Delete route table associations and route tables if they exist
  if [ -n "${PUBLIC_RT_ASSOC1_ID:-}" ]; then
    echo "Disassociating public route table from subnet 1: $PUBLIC_RT_ASSOC1_ID"
    aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC1_ID"
  fi

  if [ -n "${PUBLIC_RT_ASSOC2_ID:-}" ]; then
    echo "Disassociating public route table from subnet 2: $PUBLIC_RT_ASSOC2_ID"
    aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC2_ID"
  fi

  if [ -n "${PRIVATE_RT1_ASSOC_ID:-}" ]; then
    echo "Disassociating private route table 1: $PRIVATE_RT1_ASSOC_ID"
    aws ec2 disassociate-route-table --association-id "$PRIVATE_RT1_ASSOC_ID"
  fi

  if [ -n "${PRIVATE_RT2_ASSOC_ID:-}" ]; then
    echo "Disassociating private route table 2: $PRIVATE_RT2_ASSOC_ID"
    aws ec2 disassociate-route-table --association-id "$PRIVATE_RT2_ASSOC_ID"
  fi

  if [ -n "${PUBLIC_RT_ID:-}" ]; then
    echo "Deleting public route table: $PUBLIC_RT_ID"
    aws ec2 delete-route-table --route-table-id "$PUBLIC_RT_ID"
  fi

  if [ -n "${PRIVATE_RT1_ID:-}" ]; then
    echo "Deleting private route table 1: $PRIVATE_RT1_ID"
    aws ec2 delete-route-table --route-table-id "$PRIVATE_RT1_ID"
  fi
```

```
  if [ -n "${PRIVATE_RT2_ID:-}" ]; then
    echo "Deleting private route table 2: $PRIVATE_RT2_ID"
    aws ec2 delete-route-table --route-table-id "$PRIVATE_RT2_ID"
  fi

  # Delete subnets if they exist
  if [ -n "${PUBLIC_SUBNET1_ID:-}" ]; then
    echo "Deleting public subnet 1: $PUBLIC_SUBNET1_ID"
    aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET1_ID"
  fi

  if [ -n "${PUBLIC_SUBNET2_ID:-}" ]; then
    echo "Deleting public subnet 2: $PUBLIC_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET2_ID"
  fi

  if [ -n "${PRIVATE_SUBNET1_ID:-}" ]; then
    echo "Deleting private subnet 1: $PRIVATE_SUBNET1_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET1_ID"
  fi

  if [ -n "${PRIVATE_SUBNET2_ID:-}" ]; then
    echo "Deleting private subnet 2: $PRIVATE_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET2_ID"
  fi

  # Delete VPC if it exists
  if [ -n "${VPC_ID:-}" ]; then
    echo "Deleting VPC: $VPC_ID"
    aws ec2 delete-vpc --vpc-id "$VPC_ID"
  fi

  echo "Cleanup completed."
}

# Error handling function
handle_error() {
  echo "ERROR: $1"
  echo "Attempting to clean up resources..."
  cleanup_resources
  exit 1
}

# Function to check command success
```

```
check_command() {
  if [ $? -ne 0 ]; then
    handle_error "$1"
  fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Create VPC
echo "Creating VPC..."
VPC_RESULT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=ProductionVPC-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create VPC"

VPC_ID=$(echo "$VPC_RESULT" | jq -r '.Vpc.VpcId')
echo "VPC created with ID: $VPC_ID"

# Get Availability Zones
echo "Getting Availability Zones..."
AZ_RESULT=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0:2].ZoneName' --output text)
check_command "Failed to get Availability Zones"

# Convert space-separated output to array
read -r -a AZS <<< "$AZ_RESULT"
AZ1=${AZS[0]}
AZ2=${AZS[1]}
echo "Using Availability Zones: $AZ1 and $AZ2"

# Create subnets
echo "Creating subnets..."
PUBLIC_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.0.0/24 --availability-zone "$AZ1" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create public subnet 1"
PUBLIC_SUBNET1_ID=$(echo "$PUBLIC_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.1.0/24 --availability-zone "$AZ1" --tag-specifications
```

```
 "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private subnet 1"
PRIVATE_SUBNET1_ID=$(echo "$PRIVATE_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PUBLIC_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.2.0/24 --availability-zone "$AZ2" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create public subnet 2"
PUBLIC_SUBNET2_ID=$(echo "$PUBLIC_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.3.0/24 --availability-zone "$AZ2" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private subnet 2"
PRIVATE_SUBNET2_ID=$(echo "$PRIVATE_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

echo "Subnets created with IDs:"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_RESULT=$(aws ec2 create-internet-gateway --tag-specifications
 "ResourceType=internet-gateway,Tags=[{Key=Name,Value=ProductionIGW-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create Internet Gateway"
IGW_ID=$(echo "$IGW_RESULT" | jq -r '.InternetGateway.InternetGatewayId')
echo "Internet Gateway created with ID: $IGW_ID"

# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
 "$VPC_ID"
check_command "Failed to attach Internet Gateway to VPC"

# Create route tables
echo "Creating route tables..."
PUBLIC_RT_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PublicRouteTable-
```

```
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-
servers-gs}]")
check_command "Failed to create public route table"
PUBLIC_RT_ID=$(echo "$PUBLIC_RT_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT1_RESULT=$(aws ec2 create-route-table --vpc-
id "$VPC_ID" --tag-specifications "ResourceType=route-
table,Tags=[{Key=Name,Value=PrivateRouteTable1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private route table 1"
PRIVATE_RT1_ID=$(echo "$PRIVATE_RT1_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT2_RESULT=$(aws ec2 create-route-table --vpc-
id "$VPC_ID" --tag-specifications "ResourceType=route-
table,Tags=[{Key=Name,Value=PrivateRouteTable2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private route table 2"
PRIVATE_RT2_ID=$(echo "$PRIVATE_RT2_RESULT" | jq -r '.RouteTable.RouteTableId')

echo "Route tables created with IDs:"
echo "Public Route Table: $PUBLIC_RT_ID"
echo "Private Route Table 1: $PRIVATE_RT1_ID"
echo "Private Route Table 2: $PRIVATE_RT2_ID"

# Add route to Internet Gateway in public route table
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT_ID" --destination-cidr-block
 0.0.0.0/0 --gateway-id "$IGW_ID"
check_command "Failed to add route to Internet Gateway"

# Associate subnets with route tables
echo "Associating subnets with route tables..."
PUBLIC_RT_ASSOC1_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET1_ID")
check_command "Failed to associate public subnet 1 with route table"
PUBLIC_RT_ASSOC1_ID=$(echo "$PUBLIC_RT_ASSOC1_RESULT" | jq -r '.AssociationId')

PUBLIC_RT_ASSOC2_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET2_ID")
check_command "Failed to associate public subnet 2 with route table"
PUBLIC_RT_ASSOC2_ID=$(echo "$PUBLIC_RT_ASSOC2_RESULT" | jq -r '.AssociationId')

PRIVATE_RT1_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PRIVATE_RT1_ID" --subnet-id "$PRIVATE_SUBNET1_ID")
```

```
check_command "Failed to associate private subnet 1 with route table"
PRIVATE_RT1_ASSOC_ID=$(echo "$PRIVATE_RT1_ASSOC_RESULT" | jq -r '.AssociationId')

PRIVATE_RT2_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PRIVATE_RT2_ID" --subnet-id "$PRIVATE_SUBNET2_ID")
check_command "Failed to associate private subnet 2 with route table"
PRIVATE_RT2_ASSOC_ID=$(echo "$PRIVATE_RT2_ASSOC_RESULT" | jq -r '.AssociationId')

echo "Route table associations created with IDs:"
echo "Public Subnet 1 Association: $PUBLIC_RT_ASSOC1_ID"
echo "Public Subnet 2 Association: $PUBLIC_RT_ASSOC2_ID"
echo "Private Subnet 1 Association: $PRIVATE_RT1_ASSOC_ID"
echo "Private Subnet 2 Association: $PRIVATE_RT2_ASSOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateways..."

# Allocate Elastic IPs for NAT Gateways
echo "Allocating Elastic IPs for NAT Gateways..."
EIP1_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
 "ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT1-EIP-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to allocate Elastic IP 1"
EIP1_ALLOC_ID=$(echo "$EIP1_RESULT" | jq -r '.AllocationId')

EIP2_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
 "ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT2-EIP-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to allocate Elastic IP 2"
EIP2_ALLOC_ID=$(echo "$EIP2_RESULT" | jq -r '.AllocationId')

echo "Elastic IPs allocated with IDs:"
echo "EIP 1 Allocation ID: $EIP1_ALLOC_ID"
echo "EIP 2 Allocation ID: $EIP2_ALLOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateway in public subnet 1..."
NAT_GW1_RESULT=$(aws ec2 create-nat-gateway --subnet-id
 "$PUBLIC_SUBNET1_ID" --allocation-id "$EIP1_ALLOC_ID" --tag-specifications
 "ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create NAT Gateway 1"
NAT_GW1_ID=$(echo "$NAT_GW1_RESULT" | jq -r '.NatGateway.NatGatewayId')
```

```
echo "Creating NAT Gateway in public subnet 2..."
NAT_GW2_RESULT=$(aws ec2 create-nat-gateway --subnet-id
 "$PUBLIC_SUBNET2_ID" --allocation-id "$EIP2_ALLOC_ID" --tag-specifications
 "ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create NAT Gateway 2"
NAT_GW2_ID=$(echo "$NAT_GW2_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "NAT Gateways created with IDs:"
echo "NAT Gateway 1: $NAT_GW1_ID"
echo "NAT Gateway 2: $NAT_GW2_ID"

# Wait for NAT Gateways to be available
echo "Waiting for NAT Gateways to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW1_ID"
check_command "NAT Gateway 1 did not become available"
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW2_ID"
check_command "NAT Gateway 2 did not become available"
echo "NAT Gateways are now available"

# Add routes to NAT Gateways in private route tables
echo "Adding routes to NAT Gateways in private route tables..."
aws ec2 create-route --route-table-id "$PRIVATE_RT1_ID" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW1_ID"
check_command "Failed to add route to NAT Gateway 1"

aws ec2 create-route --route-table-id "$PRIVATE_RT2_ID" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW2_ID"
check_command "Failed to add route to NAT Gateway 2"

# Create VPC Endpoint for S3
echo "Creating VPC Endpoint for S3..."
S3_PREFIX_LIST_ID=$(aws ec2 describe-prefix-lists --filters "Name=prefix-
list-name,Values=com.amazonaws.$(aws configure get region).s3" --query
 'PrefixLists[0].PrefixListId' --output text)
check_command "Failed to get S3 prefix list ID"

VPC_ENDPOINT_RESULT=$(aws ec2 create-vpc-endpoint --vpc-id "$VPC_ID" --
service-name "com.amazonaws.$(aws configure get region).s3" --route-table-ids
 "$PRIVATE_RT1_ID" "$PRIVATE_RT2_ID" --tag-specifications "ResourceType=vpc-
endpoint,Tags=[{Key=Name,Value=S3-Endpoint-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create VPC endpoint for S3"
```

```
VPC_ENDPOINT_ID=$(echo "$VPC_ENDPOINT_RESULT" | jq -r
 '.VpcEndpoint.VpcEndpointId')
echo "VPC Endpoint created with ID: $VPC_ENDPOINT_ID"

# Create security groups
echo "Creating security groups..."
LB_SG_RESULT=$(aws ec2 create-security-group --group-name "LoadBalancerSG-
$RANDOM_ID" --description "Security group for the load balancer"
 --vpc-id "$VPC_ID" --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=LoadBalancerSG-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create load balancer security group"
LB_SG_ID=$(echo "$LB_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from anywhere to the load balancer
aws ec2 authorize-security-group-ingress --group-id "$LB_SG_ID" --protocol tcp --
port 80 --cidr 0.0.0.0/0
check_command "Failed to authorize ingress to load balancer security group"

APP_SG_RESULT=$(aws ec2 create-security-group --group-name "AppServerSG-
$RANDOM_ID" --description "Security group for the application servers"
 --vpc-id "$VPC_ID" --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=AppServerSG-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create application server security group"
APP_SG_ID=$(echo "$APP_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from the load balancer security group to the
 application servers
aws ec2 authorize-security-group-ingress --group-id "$APP_SG_ID" --protocol tcp
 --port 80 --source-group "$LB_SG_ID"
check_command "Failed to authorize ingress to application server security group"

echo "Security groups created with IDs:"
echo "Load Balancer Security Group: $LB_SG_ID"
echo "Application Server Security Group: $APP_SG_ID"

# Create a launch template
echo "Creating launch template..."

# Create user data script with IMDSv2 support
cat > user-data.sh << 'EOF'
#!/bin/bash
yum update -y
```

```
yum install -y httpd
systemctl start httpd
systemctl enable httpd

# Use IMDSv2 with session token
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600")
AZ=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/
meta-data/placement/availability-zone)
HOSTNAME=$(hostname -f)

echo "<h1>Hello from $HOSTNAME in $AZ</h1>" > /var/www/html/index.html
EOF

# Encode user data
USER_DATA=$(base64 -w 0 user-data.sh)

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon --filters
 "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2" "Name=state,Values=available" --
query 'sort_by(Images, &CreationDate)[-1].ImageId' --output text)
check_command "Failed to get latest Amazon Linux 2 AMI"
echo "Using AMI: $AMI_ID"

# Create launch template with IMDSv2 required
LAUNCH_TEMPLATE_NAME="AppServerTemplate-$RANDOM_ID"
echo "Creating launch template: $LAUNCH_TEMPLATE_NAME"

aws ec2 create-launch-template \
  --launch-template-name "$LAUNCH_TEMPLATE_NAME" \
  --version-description "Initial version" \
  --tag-specifications "ResourceType=launch-template,Tags=[{Key=Name,Value=
$LAUNCH_TEMPLATE_NAME},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-
private-servers-gs}]" \
  --launch-template-data "{
    \"NetworkInterfaces\": [{
      \"DeviceIndex\": 0,
      \"Groups\": [\"$APP_SG_ID\"],
      \"DeleteOnTermination\": true
    }],
    \"ImageId\": \"$AMI_ID\",
    \"InstanceType\": \"t3.micro\",
    \"UserData\": \"$USER_DATA\",
```

```
    \"MetadataOptions\": {
      \"HttpTokens\": \"required\",
      \"HttpEndpoint\": \"enabled\"
    },
    \"TagSpecifications\": [{
      \"ResourceType\": \"instance\",
      \"Tags\": [{
        \"Key\": \"Name\",
        \"Value\": \"AppServer-$RANDOM_ID\"
      }]
    }]
  }"
check_command "Failed to create launch template"

# Create target group
echo "Creating target group..."
TARGET_GROUP_NAME="AppTargetGroup-$RANDOM_ID"
TARGET_GROUP_RESULT=$(aws elbv2 create-target-group \
  --name "$TARGET_GROUP_NAME" \
  --protocol HTTP \
  --port 80 \
  --vpc-id "$VPC_ID" \
  --target-type instance \
  --health-check-protocol HTTP \
  --health-check-path "/" \
  --health-check-port traffic-port \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=vpc-private-servers-gs)
check_command "Failed to create target group"
TARGET_GROUP_ARN=$(echo "$TARGET_GROUP_RESULT" | jq -r
 '.TargetGroups[0].TargetGroupArn')
echo "Target group created with ARN: $TARGET_GROUP_ARN"

# Create load balancer
echo "Creating load balancer..."
LB_NAME="AppLoadBalancer-$RANDOM_ID"
LB_RESULT=$(aws elbv2 create-load-balancer \
  --name "$LB_NAME" \
  --subnets "$PUBLIC_SUBNET1_ID" "$PUBLIC_SUBNET2_ID" \
  --security-groups "$LB_SG_ID" \
  --tags "Key=Name,Value=$LB_NAME" Key=project,Value=doc-smith
 Key=tutorial,Value=vpc-private-servers-gs)
check_command "Failed to create load balancer"
LB_ARN=$(echo "$LB_RESULT" | jq -r '.LoadBalancers[0].LoadBalancerArn')
echo "Load balancer created with ARN: $LB_ARN"
```

```
# Wait for load balancer to be active
echo "Waiting for load balancer to be active..."
aws elbv2 wait load-balancer-available --load-balancer-arns "$LB_ARN"
check_command "Load balancer did not become available"

# Create listener
echo "Creating listener..."
LISTENER_RESULT=$(aws elbv2 create-listener \
  --load-balancer-arn "$LB_ARN" \
  --protocol HTTP \
  --port 80 \
  --default-actions "Type=forward,TargetGroupArn=$TARGET_GROUP_ARN")
check_command "Failed to create listener"
LISTENER_ARN=$(echo "$LISTENER_RESULT" | jq -r '.Listeners[0].ListenerArn')
echo "Listener created with ARN: $LISTENER_ARN"

# Create Auto Scaling group
echo "Creating Auto Scaling group..."
ASG_NAME="AppAutoScalingGroup-$RANDOM_ID"
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name "$ASG_NAME" \
  --launch-template "LaunchTemplateName=$LAUNCH_TEMPLATE_NAME,Version=\$Latest" \
  --min-size 2 \
  --max-size 4 \
  --desired-capacity 2 \
  --vpc-zone-identifier "$PRIVATE_SUBNET1_ID,$PRIVATE_SUBNET2_ID" \
  --target-group-arns "$TARGET_GROUP_ARN" \
  --health-check-type ELB \
  --health-check-grace-period 300 \
  --tags "Key=Name,Value=AppServer-$RANDOM_ID,PropagateAtLaunch=true"
 Key=project,Value=doc-smith,PropagateAtLaunch=true Key=tutorial,Value=vpc-
private-servers-gs,PropagateAtLaunch=true
check_command "Failed to create Auto Scaling group"
echo "Auto Scaling group created with name: $ASG_NAME"

# Get load balancer DNS name
LB_DNS_NAME=$(aws elbv2 describe-load-balancers --load-balancer-arns "$LB_ARN" --
query 'LoadBalancers[0].DNSName' --output text)
check_command "Failed to get load balancer DNS name"

echo ""
echo "============================================"
echo "DEPLOYMENT COMPLETE"
```

```
echo "============================================"
echo "VPC ID: $VPC_ID"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID (AZ: $AZ1)"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID (AZ: $AZ1)"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID (AZ: $AZ2)"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID (AZ: $AZ2)"
echo "NAT Gateway 1: $NAT_GW1_ID"
echo "NAT Gateway 2: $NAT_GW2_ID"
echo "Load Balancer: $LB_NAME"
echo "Auto Scaling Group: $ASG_NAME"
echo ""
echo "Your application will be available at: http://$LB_DNS_NAME"
echo "It may take a few minutes for the instances to launch and pass health
 checks."
echo ""

# Add health check monitoring
echo "============================================"
echo "MONITORING INSTANCE HEALTH AND LOAD BALANCER"
echo "============================================"
echo "Waiting for instances to launch and pass health checks..."
echo "This may take 3-5 minutes. Checking every 30 seconds..."

# Monitor instance health and load balancer accessibility
MAX_ATTEMPTS=10
ATTEMPT=1
HEALTHY_INSTANCES=0

while [ $ATTEMPT -le $MAX_ATTEMPTS ] && [ $HEALTHY_INSTANCES -lt 2 ]; do
  echo "Check attempt $ATTEMPT of $MAX_ATTEMPTS..."

  # Check Auto Scaling group instances
  echo "Checking Auto Scaling group instances..."
  ASG_INSTANCES=$(aws autoscaling describe-auto-scaling-groups --auto-
scaling-group-names "$ASG_NAME" --query 'AutoScalingGroups[0].Instances[*].
[InstanceId,HealthStatus]' --output json)
  echo "ASG Instances status:"
  echo "$ASG_INSTANCES" | jq -r '.[] | "Instance: \(.[0]), Health: \(.[1])"'

  # Check target group health
  echo "Checking target group health..."
  TARGET_HEALTH=$(aws elbv2 describe-target-health --target-group-arn
 "$TARGET_GROUP_ARN" --output json)
  echo "Target health status:"
```

```
 echo "$TARGET_HEALTH" | jq -r '.TargetHealthDescriptions[] | "Instance:
\(.Target.Id), State: \(.TargetHealth.State), Reason: \(.TargetHealth.Reason //
"N/A"), Description: \(.TargetHealth.Description // "N/A")"'

 # Count healthy instances
 HEALTHY_INSTANCES=$(echo "$TARGET_HEALTH" | jq -r '[.TargetHealthDescriptions[]
| select(.TargetHealth.State=="healthy")] | length')
 echo "Number of healthy instances: $HEALTHY_INSTANCES of 2 expected"

 # Check if we have healthy instances
 if [ $HEALTHY_INSTANCES -ge 2 ]; then
   echo "All instances are healthy!"

   # Test load balancer accessibility
   echo "Testing load balancer accessibility..."
   HTTP_STATUS=$(curl -s -o /dev/null -w "%{http_code}" "http://$LB_DNS_NAME")

   if [ "$HTTP_STATUS" = "200" ]; then
     echo "Load balancer is accessible! HTTP Status: $HTTP_STATUS"
     echo "You can access your application at: http://$LB_DNS_NAME"

     # Try to get the content to verify IMDSv2 is working
     echo "Fetching content to verify IMDSv2 functionality..."
     CONTENT=$(curl -s "http://$LB_DNS_NAME")
     echo "Response from server:"
     echo "$CONTENT"

     # Check if the content contains the expected pattern
     if [[ "$CONTENT" == *"Hello from"* && "$CONTENT" == *"in"* ]]; then
       echo "IMDSv2 is working correctly! The instance was able to access
metadata using the token-based approach."
     else
       echo "Warning: Content doesn't match expected pattern. IMDSv2
functionality could not be verified."
     fi

     break
   else
     echo "Load balancer returned HTTP status: $HTTP_STATUS"
     echo "Will try again in 30 seconds..."
   fi
 else
   echo "Waiting for instances to become healthy..."
   echo "Will check again in 30 seconds..."
```

```
  fi

  ATTEMPT=$((ATTEMPT+1))

  if [ $ATTEMPT -le $MAX_ATTEMPTS ]; then
    sleep 30
  fi
done

if [ $HEALTHY_INSTANCES -lt 2 ]; then
  echo "Warning: Not all instances are healthy after maximum attempts."
  echo "You may need to wait longer or check for configuration issues."
fi

echo "To test your application, run:"
echo "curl http://$LB_DNS_NAME"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  cleanup_resources
  echo "All resources have been deleted."
else
  echo "Resources will not be deleted. You can manually delete them later."
  echo "To delete resources, run this script again and choose to clean up."
fi
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateInternetGateway
  - CreateLaunchTemplate
  - CreateNatGateway

- CreateRoute
- CreateRouteTable
- CreateSecurityGroup
- CreateSubnet
- CreateVpc
- CreateVpcEndpoint
- DeleteAutoScalingGroup
- DeleteInternetGateway
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteNatGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteTargetGroup
- DeleteVpc
- DeleteVpcEndpoints
- DescribeAvailabilityZones
- DescribeImages
- DescribePrefixLists
- DetachInternetGateway
- ReleaseAddress

Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

O exemplo de código a seguir mostra como:

- Habilitar a criptografia do Amazon EBS por padrão.
- Criar snapshots do EBS.
- Criar e inicializar um volume por meio de um snapshot.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script for EBS operations: encryption, snapshots, and volume initialization
# This script demonstrates:
# 1. Enabling EBS encryption by default
# 2. Creating an EBS snapshot
# 3. Creating a volume from a snapshot
# Cost optimizations:
# - Reduced volume size from 1 GiB to 100 MiB for testing
# - Changed volume type to gp3 with cost-optimized IOPS/throughput
# - Added early cleanup to minimize storage duration
# - Removed unnecessary API calls for KMS key retrieval

set -euo pipefail

# Security: Restrict file permissions for log files
umask 0077

# Setup logging with secure permissions
LOG_FILE="ebs-operations-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting EBS operations script at $(date)"
echo "All operations will be logged to $LOG_FILE"

# Function to check command status
check_status() {
    local exit_code=$?
    if [ $exit_code -ne 0 ]; then
        echo "ERROR: $1 failed with exit code $exit_code. Exiting."
        cleanup_resources
        exit 1
```

```
    fi
}

# Function to cleanup resources with retry logic
cleanup_resources() {
    echo "Attempting to clean up resources..."
    local retry_count=0
    local max_retries=3

    if [ -n "${NEW_VOLUME_ID:-}" ]; then
        echo "Deleting new volume $NEW_VOLUME_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-volume --volume-id "$NEW_VOLUME_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted new volume $NEW_VOLUME_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $NEW_VOLUME_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete new volume $NEW_VOLUME_ID
 after $max_retries attempts"
                fi
            fi
        done
    fi

    if [ -n "${VOLUME_ID:-}" ]; then
        echo "Deleting original volume $VOLUME_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-volume --volume-id "$VOLUME_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted original volume $VOLUME_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $VOLUME_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete original volume $VOLUME_ID
 after $max_retries attempts"
```

```
                fi
            fi
        done
    fi

    if [ -n "${SNAPSHOT_ID:-}" ]; then
        echo "Deleting snapshot $SNAPSHOT_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-snapshot --snapshot-id "$SNAPSHOT_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted snapshot $SNAPSHOT_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $SNAPSHOT_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete snapshot $SNAPSHOT_ID after
 $max_retries attempts"
                fi
            fi
        done
    fi

    if [ "${ENCRYPTION_MODIFIED:-false}" = true ]; then
        echo "Restoring original encryption setting..."
        if [ "${ORIGINAL_ENCRYPTION:-}" = "False" ]; then
            aws ec2 disable-ebs-encryption-by-default --region "$AWS_REGION" 2>/
dev/null || echo "WARNING: Could not restore encryption setting"
        fi
    fi

    echo "Cleanup completed."
}

# Set trap for cleanup on exit
trap cleanup_resources EXIT

# Track created resources
VOLUME_ID=""
NEW_VOLUME_ID=""
SNAPSHOT_ID=""
ENCRYPTION_MODIFIED=false
```

```
ORIGINAL_ENCRYPTION=""

# Input validation function
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        exit 1
    fi

    # Verify AWS credentials are configured
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS credentials are not properly configured"
        exit 1
    fi
}

validate_aws_cli

# Security: Validate AWS region format
validate_region() {
    local region="$1"
    if [[ ! "$region" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
        echo "ERROR: Invalid AWS region format: $region"
        exit 1
    fi
}

# Get the current AWS region
AWS_REGION="${AWS_REGION:-$(aws configure get region)}"
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config. Using default: $AWS_REGION"
fi

validate_region "$AWS_REGION"
echo "Using AWS region: $AWS_REGION"

# Security: Validate volume ID format before use
validate_volume_id() {
    local volume_id="$1"
    if [[ ! "$volume_id" =~ ^vol-[a-z0-9]{17}$ ]]; then
        echo "ERROR: Invalid volume ID format: $volume_id"
        exit 1
    fi
```

```
}

# Security: Validate snapshot ID format before use
validate_snapshot_id() {
    local snapshot_id="$1"
    if [[ ! "$snapshot_id" =~ ^snap-[a-z0-9]{17}$ ]]; then
        echo "ERROR: Invalid snapshot ID format: $snapshot_id"
        exit 1
    fi
}

# Get availability zones in the region with caching
AVAILABILITY_ZONE=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --
query 'AvailabilityZones[0].ZoneName' --output text)
check_status "Getting availability zone"

# Security: Validate AZ format
if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}[a-z]$ ]]; then
    echo "ERROR: Invalid availability zone format: $AVAILABILITY_ZONE"
    exit 1
fi
echo "Using availability zone: $AVAILABILITY_ZONE"

# Step 1: Check and enable EBS encryption by default
echo "Step 1: Checking current EBS encryption by default setting..."
ORIGINAL_ENCRYPTION=$(aws ec2 get-ebs-encryption-by-default --region
 "$AWS_REGION" --query 'EbsEncryptionByDefault' --output text)
check_status "Checking encryption status"
echo "Current encryption by default setting: $ORIGINAL_ENCRYPTION"

if [ "$ORIGINAL_ENCRYPTION" = "False" ]; then
    echo "Enabling EBS encryption by default..."
    aws ec2 enable-ebs-encryption-by-default --region "$AWS_REGION"
    check_status "Enabling encryption by default"
    ENCRYPTION_MODIFIED=true
    echo "Updated encryption by default setting: True"
else
    echo "EBS encryption by default is already enabled."
fi

# Step 2: Create a test volume for snapshot with minimal size for cost
 optimization
echo "Step 2: Creating a test volume (1 GiB for testing)..."
```

```
VOLUME_ID=$(aws ec2 create-volume --region "$AWS_REGION" --availability-zone
 "$AVAILABILITY_ZONE" --size 1 --volume-type gp3 --iops 3000 --throughput 125
 --tag-specifications 'ResourceType=volume,Tags=[{Key=Name,Value=ebs-tutorial-
volume},{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-
smith},{Key=tutorial,Value=ebs-intermediate}]' --query 'VolumeId' --output text)
check_status "Creating test volume"

# Security: Validate volume ID
validate_volume_id "$VOLUME_ID"
echo "Created test volume: $VOLUME_ID"

# Wait for volume to become available with timeout
echo "Waiting for volume to become available..."
timeout 300 aws ec2 wait volume-available --region "$AWS_REGION" --volume-ids
 "$VOLUME_ID" || {
    echo "ERROR: Volume did not become available within timeout"
    exit 1
}
check_status "Waiting for volume"

# Step 3: Create a snapshot of the volume
echo "Step 3: Creating snapshot of the volume..."
SNAPSHOT_ID=$(aws ec2 create-snapshot --region "$AWS_REGION" --volume-id
 "$VOLUME_ID" --description "Snapshot for EBS tutorial - $(date +%Y-%m-%d)" --
tag-specifications 'ResourceType=snapshot,Tags=[{Key=Name,Value=ebs-tutorial-
snapshot},{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-
smith},{Key=tutorial,Value=ebs-intermediate}]' --query 'SnapshotId' --output
 text)
check_status "Creating snapshot"

# Security: Validate snapshot ID
validate_snapshot_id "$SNAPSHOT_ID"
echo "Created snapshot: $SNAPSHOT_ID"

# Wait for snapshot to complete with progress indication and timeout
echo "Waiting for snapshot to complete (this may take several minutes)..."
timeout 1800 aws ec2 wait snapshot-completed --region "$AWS_REGION" --snapshot-
ids "$SNAPSHOT_ID" || {
    echo "ERROR: Snapshot did not complete within timeout"
    exit 1
}
check_status "Waiting for snapshot"
echo "Snapshot completed."
```

```
# Step 4: Create a new volume from the snapshot
echo "Step 4: Creating a new volume from the snapshot..."
NEW_VOLUME_ID=$(aws ec2 create-volume --region "$AWS_REGION" --
snapshot-id "$SNAPSHOT_ID" --availability-zone "$AVAILABILITY_ZONE"
 --volume-type gp3 --iops 3000 --throughput 125 --tag-specifications
 'ResourceType=volume,Tags=[{Key=Name,Value=ebs-tutorial-volume-from-snapshot},
{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-intermediate}]' --query 'VolumeId' --output text)
check_status "Creating new volume from snapshot"

# Security: Validate new volume ID
validate_volume_id "$NEW_VOLUME_ID"
echo "Created new volume from snapshot: $NEW_VOLUME_ID"

# Wait for new volume to become available with timeout
echo "Waiting for new volume to become available..."
timeout 300 aws ec2 wait volume-available --region "$AWS_REGION" --volume-ids
 "$NEW_VOLUME_ID" || {
    echo "ERROR: New volume did not become available within timeout"
    exit 1
}
check_status "Waiting for new volume"

# Display created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "Original Volume: $VOLUME_ID"
echo "Snapshot: $SNAPSHOT_ID"
echo "New Volume: $NEW_VOLUME_ID"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup process to minimize storage costs..."

# Delete the new volume immediately to reduce storage duration
echo "Deleting new volume $NEW_VOLUME_ID..."
aws ec2 delete-volume --region "$AWS_REGION" --volume-id "$NEW_VOLUME_ID"
check_status "Deleting new volume"
```

```
# Delete the original volume
echo "Deleting original volume $VOLUME_ID..."
aws ec2 delete-volume --region "$AWS_REGION" --volume-id "$VOLUME_ID"
check_status "Deleting original volume"

# Delete the snapshot
echo "Deleting snapshot $SNAPSHOT_ID..."
aws ec2 delete-snapshot --region "$AWS_REGION" --snapshot-id "$SNAPSHOT_ID"
check_status "Deleting snapshot"

# Restore original encryption setting if modified
if [ "${ENCRYPTION_MODIFIED:-false}" = true ]; then
    echo "Restoring original encryption setting..."
    if [ "${ORIGINAL_ENCRYPTION:-}" = "False" ]; then
        aws ec2 disable-ebs-encryption-by-default --region "$AWS_REGION"
        check_status "Disabling encryption by default"
    fi
fi

echo "Cleanup completed successfully."

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateSnapshot
  - CreateVolume
  - DeleteSnapshot
  - DeleteVolume
  - DescribeAvailabilityZones
  - DescribeVolumes
  - DetachVolume
  - DisableEbsEncryptionByDefault
  - EnableEbsEncryptionByDefault
  - GetEbsDefaultKmsKeyId
  - GetEbsEncryptionByDefault
  - Wait

Trabalho com conexões de emparelhamento de rede

O exemplo de código a seguir mostra como:

- Criar VPCs para emparelhamento.
- Criar uma conexão de emparelhamento da VPC.
- Atualizar tabelas de rotas.
- Verificar a conexão de emparelhamento da VPC.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC Peering Connection Script - Version 6 (Security Enhanced)
# This script establishes a VPC peering connection between two VPCs,
# creates subnets if needed, and configures the necessary route tables.
# It will use existing VPCs if available, or create new ones if needed.

set -euo pipefail

# Security: Set strict umask
umask 0077

# Initialize log file with restricted permissions
LOG_FILE="./vpc-peering-script-v6.log"
touch "$LOG_FILE"
chmod 0600 "$LOG_FILE"
echo "Starting VPC Peering script at $(date)" > "$LOG_FILE"

# Configuration
declare -r AWS_REGION="${AWS_REGION:-us-east-1}"
```

```
declare -r MAX_RETRIES=3
declare -r RETRY_DELAY=5

# Validate script is run from secure location
if [[ "$LOG_FILE" != /* ]] && [[ "$LOG_FILE" != ./* ]]; then
    echo "ERROR: Log file path must be absolute or relative starting with ./" >&2
    exit 1
fi

# Function to sanitize variable for safe command execution
sanitize_var() {
    local var="$1"
    if [[ ! "$var" =~ ^[a-zA-Z0-9_/.-]+$ ]]; then
        echo "ERROR: Invalid characters in variable: $var" | tee -a "$LOG_FILE"
        return 1
    fi
    echo "$var"
    return 0
}

# Function to escape string for safe use in commands
escape_string() {
    local string="$1"
    printf '%s\n' "$string" | sed -e 's/[\/&]/\\&/g'
}

# Function to log commands and their output securely
log_cmd() {
    local cmd="$1"

    # Validate command doesn't contain suspicious patterns
    if [[ "$cmd" =~ (\$\(|\`|;.*rm|;.*mv|;.*cp) ]]; then
        echo "ERROR: Suspicious command pattern detected" | tee -a "$LOG_FILE"
        return 1
    fi

    echo "$(date): COMMAND: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    local exit_code="$1"
```

```
    local error_msg="${2:-Command failed}"
    if [ "$exit_code" -ne 0 ]; then
        echo "ERROR: $error_msg (exit code: $exit_code)" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        cleanup_on_error
        exit "$exit_code"
    fi
}

# Function to validate AWS CLI is available and configured
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
        exit 1
    fi

    # Check AWS CLI version
    local aws_version
    aws_version=$(aws --version 2>&1 | cut -d' ' -f1 | cut -d'/' -f2)
    echo "AWS CLI version: $aws_version" >> "$LOG_FILE"

    if ! aws sts get-caller-identity --region "$AWS_REGION" &>/dev/null; then
        echo "ERROR: AWS CLI is not properly configured or credentials are
 invalid" | tee -a "$LOG_FILE"
        exit 1
    fi

    # Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null)
    if [[ ! "$account_id" =~ ^[0-9]{12}$ ]]; then
        echo "ERROR: Invalid AWS account ID" | tee -a "$LOG_FILE"
        exit 1
    fi
    echo "Authenticated as AWS Account: $account_id" | tee -a "$LOG_FILE"
}

# Function to validate CIDR blocks
validate_cidr() {
    local cidr="$1"
    if ! [[ "$cidr" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}$ ]]; then
        echo "ERROR: Invalid CIDR block format: $cidr" | tee -a "$LOG_FILE"
        return 1
```

```
    fi

    # Additional validation for IP octets
    local ip_part="${cidr%/*}"
    local mask_part="${cidr#*/}"

    IFS='.' read -r -a octets <<< "$ip_part"
    for octet in "${octets[@]}"; do
        if (( octet > 255 )); then
            echo "ERROR: Invalid octet value in CIDR: $cidr" | tee -a "$LOG_FILE"
            return 1
        fi
    done

    if (( mask_part > 32 || mask_part < 0 )); then
        echo "ERROR: Invalid CIDR mask value: $mask_part" | tee -a "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 "$LOG_FILE"

    # List created resources
    echo "Resources created:" | tee -a "$LOG_FILE"
    for resource in "${CREATED_RESOURCES[@]:-}"; do
        echo "- $resource" | tee -a "$LOG_FILE"
    done

    # Clean up in reverse order with retry logic
    for ((i=${#CLEANUP_COMMANDS[@]}-1; i>=0; i--)); do
        echo "Executing cleanup: ${CLEANUP_COMMANDS[$i]}" >> "$LOG_FILE"
        local retry_count=0
        while [ $retry_count -lt $MAX_RETRIES ]; do
            if eval "${CLEANUP_COMMANDS[$i]}" 2>&1 >> "$LOG_FILE"; then
                break
            else
                retry_count=$((retry_count + 1))
                if [ $retry_count -lt $MAX_RETRIES ]; then
```

```
                    echo "Cleanup command failed, retrying in ${RETRY_DELAY}s..."
 >> "$LOG_FILE"
                    sleep "$RETRY_DELAY"
                fi
            fi
        done
    done
}

# Array to store created resources and cleanup commands
declare -a CREATED_RESOURCES=()
declare -a CLEANUP_COMMANDS=()

# Trap errors and cleanup
trap cleanup_on_error EXIT

echo "Setting up VPC peering connection..."

# Validate AWS CLI
validate_aws_cli

# Check for existing VPCs
echo "Checking for existing VPCs..."
EXISTING_VPCS=$(aws ec2 describe-vpcs --region "$AWS_REGION" --query 'Vpcs[?
State==`available`].[VpcId,CidrBlock]' --output text 2>/dev/null || echo "")

if [ -z "$EXISTING_VPCS" ]; then
    echo "No existing VPCs found. Creating new VPCs..."
    CREATE_VPCS=true
else
    echo "Found existing VPCs:"
    echo "$EXISTING_VPCS"
    echo ""
    echo "Using existing VPCs..."
    CREATE_VPCS=false
    # Get the first two available VPCs
    VPC1_INFO=$(echo "$EXISTING_VPCS" | head -n 1)
    VPC2_INFO=$(echo "$EXISTING_VPCS" | head -n 2 | tail -n 1)

    if [ -z "$VPC2_INFO" ]; then
        echo "Only one VPC found. Creating a second VPC..."
        VPC1_ID=$(echo "$VPC1_INFO" | awk '{print $1}')
        VPC1_CIDR=$(echo "$VPC1_INFO" | awk '{print $2}')
```

```
        # Sanitize extracted values
        VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID
 format"
        VPC1_CIDR=$(sanitize_var "$VPC1_CIDR") || check_error 1 "Invalid
 VPC1_CIDR format"

        validate_cidr "$VPC1_CIDR" || check_error 1 "Invalid VPC1 CIDR"
        CREATE_VPC2_ONLY=true
    else
        VPC1_ID=$(echo "$VPC1_INFO" | awk '{print $1}')
        VPC1_CIDR=$(echo "$VPC1_INFO" | awk '{print $2}')
        VPC2_ID=$(echo "$VPC2_INFO" | awk '{print $1}')
        VPC2_CIDR=$(echo "$VPC2_INFO" | awk '{print $2}')

        # Sanitize extracted values
        VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID
 format"
        VPC1_CIDR=$(sanitize_var "$VPC1_CIDR") || check_error 1 "Invalid
 VPC1_CIDR format"
        VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID
 format"
        VPC2_CIDR=$(sanitize_var "$VPC2_CIDR") || check_error 1 "Invalid
 VPC2_CIDR format"

        validate_cidr "$VPC1_CIDR" || check_error 1 "Invalid VPC1 CIDR"
        validate_cidr "$VPC2_CIDR" || check_error 1 "Invalid VPC2 CIDR"
        CREATE_VPC2_ONLY=false
    fi
fi

# Create VPCs if needed
if [ "$CREATE_VPCS" = true ]; then
    echo "Creating VPC1..."
    VPC1_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.1.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC1"
    VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID
 returned"
    VPC1_CIDR="10.1.0.0/16"
    CREATED_RESOURCES+=("VPC1: $VPC1_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC1_ID'")
```

```
    echo "VPC1 created with ID: $VPC1_ID"

    echo "Creating VPC2..."
    VPC2_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.2.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC2"
    VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID
 returned"
    VPC2_CIDR="10.2.0.0/16"
    CREATED_RESOURCES+=("VPC2: $VPC2_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC2_ID'")
    echo "VPC2 created with ID: $VPC2_ID"

    # Wait for VPCs to be available
    echo "Waiting for VPCs to be available..."
    log_cmd "aws ec2 wait vpc-available --region '$AWS_REGION' --vpc-ids
 '$VPC1_ID' '$VPC2_ID'"
    check_error $? "Timeout waiting for VPCs to become available"

elif [ "$CREATE_VPC2_ONLY" = true ]; then
    echo "Creating VPC2..."
    VPC2_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.2.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC2"
    VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID
 returned"
    VPC2_CIDR="10.2.0.0/16"
    CREATED_RESOURCES+=("VPC2: $VPC2_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC2_ID'")
    echo "VPC2 created with ID: $VPC2_ID"

    # Wait for VPC2 to be available
    echo "Waiting for VPC2 to be available..."
    log_cmd "aws ec2 wait vpc-available --region '$AWS_REGION' --vpc-ids
 '$VPC2_ID'"
    check_error $? "Timeout waiting for VPC2 to become available"
fi
```

```
echo "Using the following VPCs:"
echo "VPC1: $VPC1_ID ($VPC1_CIDR)"
echo "VPC2: $VPC2_ID ($VPC2_CIDR)"

# Verify the VPCs exist and are available
echo "Verifying VPCs..."
log_cmd "aws ec2 describe-vpcs --region '$AWS_REGION' --vpc-ids '$VPC1_ID'
 '$VPC2_ID' --query 'Vpcs[*].[VpcId,State,CidrBlock]' --output table"
check_error $? "Failed to verify VPCs"

# Determine subnet CIDR blocks based on VPC CIDR blocks
VPC1_SUBNET_CIDR=$(echo "$VPC1_CIDR" | sed 's/0\.0\/16/1.0\/24/')
VPC2_SUBNET_CIDR=$(echo "$VPC2_CIDR" | sed 's/0\.0\/16/1.0\/24/')

# Sanitize subnet CIDR blocks
VPC1_SUBNET_CIDR=$(sanitize_var "$VPC1_SUBNET_CIDR") || check_error 1 "Invalid
 VPC1_SUBNET_CIDR format"
VPC2_SUBNET_CIDR=$(sanitize_var "$VPC2_SUBNET_CIDR") || check_error 1 "Invalid
 VPC2_SUBNET_CIDR format"

validate_cidr "$VPC1_SUBNET_CIDR" || check_error 1 "Invalid subnet CIDR for VPC1"
validate_cidr "$VPC2_SUBNET_CIDR" || check_error 1 "Invalid subnet CIDR for VPC2"

# Create subnets in both VPCs
echo "Creating subnet in VPC1..."
SUBNET1_ID=$(log_cmd "aws ec2 create-subnet --region '$AWS_REGION' --
vpc-id '$VPC1_ID' --cidr-block '$VPC1_SUBNET_CIDR' --tag-specifications
 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Peering-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'Subnet.SubnetId' --output text")
check_error $? "Failed to create subnet in VPC1"
SUBNET1_ID=$(sanitize_var "$SUBNET1_ID") || check_error 1 "Invalid SUBNET1_ID
 returned"
CREATED_RESOURCES+=("Subnet in VPC1: $SUBNET1_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET1_ID'")
echo "Subnet created in VPC1 with ID: $SUBNET1_ID (CIDR: $VPC1_SUBNET_CIDR)"

echo "Creating subnet in VPC2..."
SUBNET2_ID=$(log_cmd "aws ec2 create-subnet --region '$AWS_REGION' --
vpc-id '$VPC2_ID' --cidr-block '$VPC2_SUBNET_CIDR' --tag-specifications
 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Peering-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'Subnet.SubnetId' --output text")
```

```
check_error $? "Failed to create subnet in VPC2"
SUBNET2_ID=$(sanitize_var "$SUBNET2_ID") || check_error 1 "Invalid SUBNET2_ID
 returned"
CREATED_RESOURCES+=("Subnet in VPC2: $SUBNET2_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET2_ID'")
echo "Subnet created in VPC2 with ID: $SUBNET2_ID (CIDR: $VPC2_SUBNET_CIDR)"

# Create a VPC peering connection
echo "Creating VPC peering connection..."
PEERING_ID=$(log_cmd "aws ec2 create-vpc-peering-connection --region
 '$AWS_REGION' --vpc-id '$VPC1_ID' --peer-vpc-id '$VPC2_ID' --tag-specifications
 'ResourceType=vpc-peering-connection,Tags=[{Key=Name,Value=VPC1-VPC2-Peering},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'VpcPeeringConnection.VpcPeeringConnectionId' --output text")
check_error $? "Failed to create VPC peering connection"
PEERING_ID=$(sanitize_var "$PEERING_ID") || check_error 1 "Invalid PEERING_ID
 returned"
CREATED_RESOURCES+=("VPC Peering Connection: $PEERING_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-vpc-peering-connection --region '$AWS_REGION'
 --vpc-peering-connection-id '$PEERING_ID'")
echo "VPC Peering Connection created with ID: $PEERING_ID"

# Accept the VPC peering connection
echo "Accepting VPC peering connection..."
log_cmd "aws ec2 accept-vpc-peering-connection --region '$AWS_REGION' --vpc-
peering-connection-id '$PEERING_ID'"
check_error $? "Failed to accept VPC peering connection"
echo "VPC Peering Connection accepted"

# Wait for the peering connection to become active
echo "Waiting for peering connection to become active..."
log_cmd "aws ec2 wait vpc-peering-connection-exists --region '$AWS_REGION' --vpc-
peering-connection-ids '$PEERING_ID'"
check_error $? "Timeout waiting for peering connection to become active"

# Create a route table for VPC1
echo "Creating route table for VPC1..."
RTB1_ID=$(log_cmd "aws ec2 create-route-table --region '$AWS_REGION'
 --vpc-id '$VPC1_ID' --tag-specifications 'ResourceType=route-
table,Tags=[{Key=Name,Value=VPC1-RouteTable},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-peering}]' --query 'RouteTable.RouteTableId' --output
 text")
check_error $? "Failed to create route table for VPC1"
```

```
RTB1_ID=$(sanitize_var "$RTB1_ID") || check_error 1 "Invalid RTB1_ID returned"
CREATED_RESOURCES+=("Route Table for VPC1: $RTB1_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-route-table --region '$AWS_REGION' --route-
table-id '$RTB1_ID'")
echo "Route table created for VPC1 with ID: $RTB1_ID"

# Create a route from VPC1 to VPC2
echo "Creating route from VPC1 to VPC2..."
log_cmd "aws ec2 create-route --region '$AWS_REGION' --route-table-id '$RTB1_ID'
 --destination-cidr-block '$VPC2_CIDR' --vpc-peering-connection-id '$PEERING_ID'"
check_error $? "Failed to create route from VPC1 to VPC2"
echo "Route created from VPC1 to VPC2"

# Associate the route table with the subnet in VPC1
echo "Associating route table with subnet in VPC1..."
RTB1_ASSOC_ID=$(log_cmd "aws ec2 associate-route-table --region '$AWS_REGION' --
route-table-id '$RTB1_ID' --subnet-id '$SUBNET1_ID' --query 'AssociationId' --
output text")
check_error $? "Failed to associate route table with subnet in VPC1"
RTB1_ASSOC_ID=$(sanitize_var "$RTB1_ASSOC_ID") || check_error 1 "Invalid
 RTB1_ASSOC_ID returned"
CREATED_RESOURCES+=("Route Table Association for VPC1: $RTB1_ASSOC_ID")
CLEANUP_COMMANDS+=("aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB1_ASSOC_ID'")
echo "Route table associated with subnet in VPC1"

# Create a route table for VPC2
echo "Creating route table for VPC2..."
RTB2_ID=$(log_cmd "aws ec2 create-route-table --region '$AWS_REGION'
 --vpc-id '$VPC2_ID' --tag-specifications 'ResourceType=route-
table,Tags=[{Key=Name,Value=VPC2-RouteTable},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-peering}]' --query 'RouteTable.RouteTableId' --output
 text")
check_error $? "Failed to create route table for VPC2"
RTB2_ID=$(sanitize_var "$RTB2_ID") || check_error 1 "Invalid RTB2_ID returned"
CREATED_RESOURCES+=("Route Table for VPC2: $RTB2_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-route-table --region '$AWS_REGION' --route-
table-id '$RTB2_ID'")
echo "Route table created for VPC2 with ID: $RTB2_ID"

# Create a route from VPC2 to VPC1
echo "Creating route from VPC2 to VPC1..."
log_cmd "aws ec2 create-route --region '$AWS_REGION' --route-table-id '$RTB2_ID'
 --destination-cidr-block '$VPC1_CIDR' --vpc-peering-connection-id '$PEERING_ID'"
```

```
check_error $? "Failed to create route from VPC2 to VPC1"
echo "Route created from VPC2 to VPC1"

# Associate the route table with the subnet in VPC2
echo "Associating route table with subnet in VPC2..."
RTB2_ASSOC_ID=$(log_cmd "aws ec2 associate-route-table --region '$AWS_REGION' --
route-table-id '$RTB2_ID' --subnet-id '$SUBNET2_ID' --query 'AssociationId' --
output text")
check_error $? "Failed to associate route table with subnet in VPC2"
RTB2_ASSOC_ID=$(sanitize_var "$RTB2_ASSOC_ID") || check_error 1 "Invalid
 RTB2_ASSOC_ID returned"
CREATED_RESOURCES+=("Route Table Association for VPC2: $RTB2_ASSOC_ID")
CLEANUP_COMMANDS+=("aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB2_ASSOC_ID'")
echo "Route table associated with subnet in VPC2"

# Verify the VPC peering connection
echo "Verifying VPC peering connection..."
log_cmd "aws ec2 describe-vpc-peering-connections --region '$AWS_REGION' --
vpc-peering-connection-ids '$PEERING_ID' --query 'VpcPeeringConnections[0].
[VpcPeeringConnectionId,Status.Code,AccepterVpcInfo.VpcId,RequesterVpcInfo.VpcId]'
 --output table"
check_error $? "Failed to verify VPC peering connection"
echo "VPC peering connection verified"

# Display summary of created resources
echo ""
echo "==============================================="
echo "SUMMARY OF CREATED RESOURCES"
echo "==============================================="
echo "VPC1 ID: $VPC1_ID"
echo "VPC1 CIDR: $VPC1_CIDR"
echo "Subnet1 ID: $SUBNET1_ID (CIDR: $VPC1_SUBNET_CIDR)"
echo "VPC2 ID: $VPC2_ID"
echo "VPC2 CIDR: $VPC2_CIDR"
echo "Subnet2 ID: $SUBNET2_ID (CIDR: $VPC2_SUBNET_CIDR)"
echo "Peering Connection ID: $PEERING_ID"
echo "Route Table 1 ID: $RTB1_ID"
echo "Route Table 1 Association ID: $RTB1_ASSOC_ID"
echo "Route Table 2 ID: $RTB2_ID"
echo "Route Table 2 Association ID: $RTB2_ASSOC_ID"
echo ""
echo "Created resources:"
for resource in "${CREATED_RESOURCES[@]}"; do
```

```
    echo "- $resource"
done
echo "================================================"
echo ""

# Test connectivity (optional)
echo "================================================"
echo "CONNECTIVITY TEST"
echo "================================================"
echo "To test connectivity between VPCs, you would need to:"
echo "1. Launch EC2 instances in each subnet"
echo "2. Configure security groups to allow traffic"
echo "3. Test ping or other network connectivity"
echo ""

# Automatic cleanup
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Auto-confirming cleanup of all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Clean up in reverse order
    echo "Disassociating route table from subnet in VPC2..."
    log_cmd "aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB2_ASSOC_ID'" || true

    echo "Disassociating route table from subnet in VPC1..."
    log_cmd "aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB1_ASSOC_ID'" || true

    echo "Deleting route table for VPC2..."
    log_cmd "aws ec2 delete-route-table --region '$AWS_REGION' --route-table-id
 '$RTB2_ID'" || true

    echo "Deleting route table for VPC1..."
    log_cmd "aws ec2 delete-route-table --region '$AWS_REGION' --route-table-id
 '$RTB1_ID'" || true

    echo "Deleting VPC peering connection..."
```

```
    log_cmd "aws ec2 delete-vpc-peering-connection --region '$AWS_REGION' --vpc-
peering-connection-id '$PEERING_ID'" || true

    echo "Deleting subnet in VPC2..."
    log_cmd "aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET2_ID'" || true

    echo "Deleting subnet in VPC1..."
    log_cmd "aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET1_ID'" || true

    # Delete VPCs if they were created by this script
    if [ "$CREATE_VPCS" = true ]; then
        echo "Deleting VPC2..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC2_ID'"
 || true

        echo "Deleting VPC1..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC1_ID'"
 || true
    elif [ "$CREATE_VPC2_ONLY" = true ]; then
        echo "Deleting VPC2..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC2_ID'"
 || true
    fi

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo ""
    echo "To manually clean up later, you can delete resources in this order:"
    echo "1. Route table associations"
    echo "2. Route tables"
    echo "3. VPC peering connection"
    echo "4. Subnets"
    if [ "$CREATE_VPCS" = true ] || [ "$CREATE_VPC2_ONLY" = true ]; then
        echo "5. VPCs (if created by this script)"
    fi
fi

echo "Script execution completed. See $LOG_FILE for detailed logs."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AcceptVpcPeeringConnection
- AssociateRouteTable
- CreateRoute
- CreateRouteTable
- CreateSubnet
- CreateVpc
- CreateVpcPeeringConnection
- DeleteRouteTable
- DeleteSubnet
- DeleteVpc
- DeleteVpcPeeringConnection
- DescribeVpcPeeringConnections
- DescribeVpcs
- DisassociateRouteTable
- Wait

Exemplos de código para o Amazon ECR usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Elastic Container Registry com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon ECR: mais informações sobre o Amazon ECR.
- Referência da API Amazon ECR: detalhes sobre todas as ações disponíveis do Amazon ECR.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon ECR AWS SDKs
  - Hello Amazon ECR
  - Aprenda as noções básicas do Amazon ECR com um AWS SDK
  - Ações para o Amazon ECR usando AWS SDKs
    - Use CreateRepositorycom um AWS SDK ou CLI
    - Use DeleteRepositorycom um AWS SDK ou CLI
    - Use DescribeImagescom um AWS SDK ou CLI
    - Use DescribeRepositoriescom um AWS SDK ou CLI
    - Use GetAuthorizationTokencom um AWS SDK ou CLI
    - Use GetRepositoryPolicycom um AWS SDK ou CLI
    - Use ListImagescom um AWS SDK ou CLI
    - Use PushImageCmdcom um AWS SDK
    - Use PutLifeCyclePolicycom um AWS SDK ou CLI
    - Use SetRepositoryPolicycom um AWS SDK ou CLI
    - Use StartLifecyclePolicyPreviewcom um AWS SDK ou CLI
- Cenários para o Amazon ECR usando AWS SDKs
  - Conceitos básicos de registros de contêineres

Exemplos básicos para o uso do Amazon ECR AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Elastic Container Registry com SDKs da AWS .

Exemplos

- Hello Amazon ECR
- Aprenda as noções básicas do Amazon ECR com um AWS SDK

- Ações para o Amazon ECR usando AWS SDKs
  - Use CreateRepositorycom um AWS SDK ou CLI
  - Use DeleteRepositorycom um AWS SDK ou CLI
  - Use DescribeImagescom um AWS SDK ou CLI
  - Use DescribeRepositoriescom um AWS SDK ou CLI
  - Use GetAuthorizationTokencom um AWS SDK ou CLI
  - Use GetRepositoryPolicycom um AWS SDK ou CLI
  - Use ListImagescom um AWS SDK ou CLI
  - Use PushImageCmdcom um AWS SDK
  - Use PutLifeCyclePolicycom um AWS SDK ou CLI
  - Use SetRepositoryPolicycom um AWS SDK ou CLI
  - Use StartLifecyclePolicyPreviewcom um AWS SDK ou CLI

Hello Amazon ECR

Os exemplos de código a seguir mostram como começar a usar o Amazon ECR.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

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

```
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();

        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Consulte detalhes da API em listImages na Referência de APIs do AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
```

```
        }
    }
}
```

- Consulte detalhes da API em listImages na Referência de APIs do AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )
```

```
    image_names: [str] = []
    for page in page_iterator:
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Consulte detalhes da API em listImages na Referência de API do AWS SDK para Python (Boto3).

Aprenda as noções básicas do Amazon ECR com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um repositório do Amazon ECR.
- Defina políticas de repositório.
- Recupere URIs de repositórios.
- Obtenha tokens de autorização do Amazon ECR.
- Defina políticas de ciclo de vida para repositórios do Amazon ECR.
- Envie por push uma imagem do Docker para um repositório do Amazon ECR.
- Verifique a existência de uma imagem em um repositório do Amazon ECR.
- Liste os repositórios do Amazon ECR da conta e verifique os detalhes sobre eles.

- Exclua repositórios do Amazon ECR.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon ECR.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact
 with the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text.
 Without a local image,
 * this Java program will not successfully run. For more information including
 how to create the local
 * image, see:
 *
```

```
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static void main(String[] args) {
        final String usage = """
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions
 to access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed
 Docker container registry
             service provided by AWS. It allows developers and organizations to
 securely
             store, manage, and deploy Docker container images.
             ECR provides a simple and scalable way to manage container images
 throughout their lifecycle,
             from building and testing to production deployment.\s

             The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides
 a set of methods to
             programmatically interact with the Amazon ECR service. This allows
 developers to
             automate the storage, retrieval, and management of container images
 as part of their application
```

```
            deployment pipelines. With ECR, teams can focus on building and
 deploying their
            applications without having to worry about the underlying
 infrastructure required to
            host and manage a container registry.

           This scenario walks you through how to perform key operations for
 this service.
           Let's get started...

           You have two choices:
           1 - Run the entire program.
           2 - Delete an existing Amazon ECR repository named echo-text (created
 from a previous execution of
           this program that did not complete).
           """);

       while (true) {
           String input = scanner.nextLine();
           if (input.trim().equalsIgnoreCase("1")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else if (input.trim().equalsIgnoreCase("2")) {
               String repoName = "echo-text";
               ecrActions.deleteECRRepository(repoName);
               return;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
       }

       waitForInputToContinue(scanner);
       System.out.println(DASHES);

       System.out.println("""
          1. Create an ECR repository.

          The first task is to ensure we have a local Docker image named echo-
text.
          If this image exists, then an Amazon ECR repository is created.

          An ECR repository is a private Docker container repository provided
```

```
          by Amazon Web Services (AWS). It is a managed service that makes it
easy
          to store, manage, and deploy Docker container images.\s
          """ );

       // Ensure that a local docker image named echo-text exists.
       boolean doesExist = ecrActions.isEchoTextImagePresent();
       String repoName;
       if (!doesExist){
           System.out.println("The local image named echo-text does not exist");
           return;
       } else {
           localImageName = "echo-text";
           repoName = "echo-text";
       }

       try {
           String repoArn = ecrActions.createECRRepository(repoName);
           System.out.println("The ARN of the ECR repository is " + repoArn);

       } catch (IllegalArgumentException e) {
           System.err.println("Invalid repository name: " + e.getMessage());
           return;
       } catch (RuntimeException e) {
           System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
           e.printStackTrace();
           return;
       }
       waitForInputToContinue(scanner);

       System.out.println(DASHES);
       System.out.println("""
       2. Set an ECR repository policy.

       Setting an ECR repository policy using the `setRepositoryPolicy` function
is crucial for maintaining
       the security and integrity of your container images. The repository
policy allows you to
       define specific rules and restrictions for accessing and managing the
images stored within your ECR
       repository.
       """);
       waitForInputToContinue(scanner);
```

```
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.
```

```
        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the
authorization token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime
to pull the
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);
```

```
        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker
images stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images,
ensuring that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
```

```
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message
indicating that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker
image to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
```

```
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the
 image in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
 CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS
 CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=
%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository
 and have Amazon ECR delete all of its contents
        on your behalf.
        """);
```

```
        System.out.println("Would you like to delete the Amazon ECR Repository?
 (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " +
 e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("An error occurred while deleting the Docker
 image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
          }
       }
   }
}
```

Uma classe de wrapper para os métodos do SDK do Amazon ECR.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
```

```
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<CreateRepositoryResponse> response =
 getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
 successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
```

```
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return
describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }

    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();
```

```
        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " +
repoName + " repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }



    private static DockerClient getDockerClient() {
        String osName = System.getProperty("os.name");
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
            DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build();
        }
        return dockerClient;
    }

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
```

```
    private static EcrAsyncClient getAsyncClient() {

        /*
         The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
         and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
         It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
         provide a non-blocking, event-driven approach to HTTP requests and
responses.
         */
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(50)  // Adjust as needed.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
policy.
     */
```

```
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
 StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
 getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
 successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
```

```
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
ex.getCause());
                }
```

```
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
     * @throws CompletionException if the asynchronous operation completes
 exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request =
 DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
 getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
 asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
 cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
```

```
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
     * If the operation is successful, the method prints the token to the
console.
     * If an exception occurs, the method handles the exception and prints the
error message.
     *
     * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
     * @throws RuntimeException if there is an unexpected error during the
operation.
     */
    public void getAuthToken() {
        CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
                AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
```

```
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }

    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     * @throws EcrException if an AWS error occurs while getting the repository
policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
```

```
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }

    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
     * @throws EcrException                      if there is an unexpected error
setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
        /*
          This example policy document grants the specified AWS principal the
permission to perform the
          `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
          to retrieve Docker images from the ECR repository.
         */
        String policyDocumentTemplate = """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "%s"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }
             """;

        String policyDocument = String.format(policyDocumentTemplate, iamRole);
```

```
        SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .policyText(policyDocument)
            .build();

        CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy set successfully.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof RepositoryPolicyNotFoundException) {
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new
String(Base64.getDecoder().decode(token));
```

```
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
```

```
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon ECR.

```
import java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a
 local image,
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
```

```
        """
        Usage: <iamRoleARN> <accountId>

        Where:
           iamRoleARN - The IAM role ARN that has the necessary permissions to
access and manage the Amazon ECR repository.
           accountId - Your AWS account number.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        return
    }

    var iamRole = args[0]
    var localImageName: String
    var accountId = args[1]
    val ecrActions = ECRActions()
    val scanner = Scanner(System.`in`)

    println(
        """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
        service provided by AWS. It allows developers and organizations to
securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
        from building and testing to production deployment.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
developers to
        automate the storage, retrieval, and management of container images as
part of their application
        deployment pipelines. With ECR, teams can focus on building and deploying
their
        applications without having to worry about the underlying infrastructure
required to
        host and manage a container registry.
```

```
        This scenario walks you through how to perform key operations for this
 service.
        Let's get started...

         You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
 from a previous execution of
            this program that did not complete).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }

    waitForInputToContinue(scanner)
    println(DASHES)
    println(
        """
        1. Create an ECR repository.

        The first task is to ensure we have a local Docker image named echo-
text.
        If this image exists, then an Amazon ECR repository is created.

        An ECR repository is a private Docker container repository provided
        by Amazon Web Services (AWS). It is a managed service that makes it easy
        to store, manage, and deploy Docker container images.

        """.trimIndent(),
```

```
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function
is crucial for maintaining
        the security and integrity of your container images. The repository
policy allows you to
        define specific rules and restrictions for accessing and managing the
images stored within your ECR
        repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
```

```
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.


        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.getAuthToken()
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime
to pull the
```

```
        correct container image from the ECR repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
    println("The repository URI is $repositoryURI")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        6. Set an ECR Lifecycle Policy.

        An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
        These policies allow you to automatically remove old or unused Docker
images from your repositories,
        freeing up storage space and reducing costs.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val pol = ecrActions.setLifeCyclePolicy(repoName)
    println(pol)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
```

```
        repository name and image tag, and then pushes the image to the ECR
 repository using the Docker client.
        If the push operation is successful, the method prints a message
 indicating that the image was pushed to ECR.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("9. As an optional step, you can interact with the image in Amazon
 ECR by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
 image? (y/n)")
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
        1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
 you need to authenticate with the registry. You can do this using the AWS CLI:

            aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

        2. Describe the image using this command:

           aws ecr describe-images --repository-name $repoName --image-ids
 imageTag=$localImageName

        3. Run the Docker container and view the output using this command:

           docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
```

```
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository
 and have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }
}
```

Uma classe de wrapper para os métodos do SDK do Amazon ECR.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
```

```
     *
     * @param repoName the name of the repository for which to set the lifecycle
policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
            ]
            }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
```

```
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
```

```
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
```

```
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }


    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
```

```
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeResponse =
ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
        }
    }

    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
```

```
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
```

```
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
```

```
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)

            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
            val registryURL: String =
 repoData?.repositoryUri?.split("/")?.get(0) ?: ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateRepository
  - DeleteRepository

- DescribeImages
- DescribeRepositories
- GetAuthorizationToken
- GetRepositoryPolicy
- SetRepositoryPolicy
- StartLifecyclePolicyPreview

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class ECRGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to perform basic operations
 using
    Amazon ECR.
    """

    def __init__(
        self,
        ecr_wrapper: ECRWrapper,
        docker_client: docker.DockerClient,
    ):
        self.ecr_wrapper = ecr_wrapper
        self.docker_client = docker_client
        self.tag = "echo-text"
        self.repository_name = "ecr-basics"
        self.docker_image = None
        self.full_tag_name = None
        self.repository = None
```

```
    def run(self, role_arn: str) -> None:
        """
        Runs the scenario.
        """
        print(
            """
The Amazon Elastic Container Registry (ECR) is a fully-managed Docker container
 registry
service provided by AWS. It allows developers and organizations to securely
store, manage, and deploy Docker container images.
ECR provides a simple and scalable way to manage container images throughout
 their lifecycle,
from building and testing to production deployment.

The `ECRWrapper' class is a wrapper for the Boto3 'ecr' client. The 'ecr' client
 provides a set of methods to
programmatically interact with the Amazon ECR service. This allows developers to
automate the storage, retrieval, and management of container images as part of
 their application
deployment pipelines. With ECR, teams can focus on building and deploying their
applications without having to worry about the underlying infrastructure required
 to
host and manage a container registry.

This scenario walks you through how to perform key operations for this service.
Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(
            f"""
* Create an ECR repository.

An ECR repository is a private Docker container repository provided
by Amazon Web Services (AWS). It is a managed service that makes it easy
to store, manage, and deploy Docker container images.
        """
        )
        print(f"Creating a repository named {self.repository_name}")
        self.repository =
 self.ecr_wrapper.create_repository(self.repository_name)
        print(f"The ARN of the ECR repository is
 {self.repository['repositoryArn']}")
```

```
        repository_uri = self.repository["repositoryUri"]
        press_enter_to_continue()
        print_dashes()

        print(
            f"""
* Build a Docker image.

Create a local Docker image if it does not already exist.
A Python Docker client is used to execute Docker commands.
You must have Docker installed and running.
            """
        )
        print(f"Building a docker image from 'docker_files/Dockerfile'")
        self.full_tag_name = f"{repository_uri}:{self.tag}"
        self.docker_image = self.docker_client.images.build(
            path="docker_files", tag=self.full_tag_name
        )[0]
        print(f"Docker image {self.full_tag_name} successfully built.")
        press_enter_to_continue()
        print_dashes()

        if role_arn is None:
            print(
                """
* Because an IAM role ARN was not provided, a role policy will not be set for
 this repository.
            """
            )
        else:
            print(
                """
* Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is
 crucial for maintaining
the security and integrity of your container images. The repository policy allows
 you to
define specific rules and restrictions for accessing and managing the images
 stored within your ECR
repository.
        """
            )
```

```
            self.grant_role_download_access(role_arn)
            print(f"Download access granted to the IAM role ARN {role_arn}")
            press_enter_to_continue()
            print_dashes()

            print(
                """
* Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
            """
            )

            policy_text =
 self.ecr_wrapper.get_repository_policy(self.repository_name)
            print("Policy Text:")
            print(f"{policy_text}")
            press_enter_to_continue()
            print_dashes()

        print(
            """
* Retrieve an ECR authorization token.

You need an authorization token to securely access and interact with the Amazon
 ECR registry.
The `get_authorization_token` method of the `ecr` client is responsible for
 securely accessing
and interacting with an Amazon ECR repository. This operation is responsible for
 obtaining a
valid authorization token, which is required to authenticate your requests to the
 ECR service.

Without a valid authorization token, you would not be able to perform any
 operations on the
ECR repository, such as pushing, pulling, or managing your Docker images.
        """
        )

        authorization_token = self.ecr_wrapper.get_authorization_token()
        print("Authorization token retrieved.")
        press_enter_to_continue()
        print_dashes()
        print(
```

```
            """
* Get the ECR Repository URI.

The URI  of an Amazon ECR repository is important. When you want to deploy a
 container image to
a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)
or Amazon Elastic Container Service (ECS), you need to specify the full image
 URI,
which includes the ECR repository URI. This allows the container runtime to pull
 the
correct container image from the ECR repository.
        """
        )
        repository_descriptions = self.ecr_wrapper.describe_repositories(
            [self.repository_name]
        )
        repository_uri = repository_descriptions[0]["repositoryUri"]
        print(f"Repository URI found: {repository_uri}")
        press_enter_to_continue()
        print_dashes()

        print(
            """
* Set an ECR Lifecycle Policy.

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored
 in your ECR repositories.
These policies allow you to automatically remove old or unused Docker images from
 your repositories,
freeing up storage space and reducing costs.

This example policy helps to maintain the size and efficiency of the container
 registry
by automatically removing older and potentially unused images, ensuring that the
storage is optimized and the registry remains up-to-date.
            """
        )
        press_enter_to_continue()
        self.put_expiration_policy()
        print(f"An expiration policy was added to the repository.")
        print_dashes()

        print(
            """
```

```
* Push a docker image to the Amazon ECR Repository.

The Docker client uses the authorization token is used to authenticate the when
 pushing the image to the
ECR repository.
        """
        )
        decoded_authorization =
 base64.b64decode(authorization_token).decode("utf-8")
        username, password = decoded_authorization.split(":")

        resp = self.docker_client.api.push(
            repository=repository_uri,
            auth_config={"username": username, "password": password},
            tag=self.tag,
            stream=True,
            decode=True,
        )
        for line in resp:
            print(line)

        print_dashes()

        print("* Verify if the image is in the ECR Repository.")
        image_descriptions = self.ecr_wrapper.describe_images(
            self.repository_name, [self.tag]
        )
        if len(image_descriptions) > 0:
            print("Image found in ECR Repository.")
        else:
            print("Image not found in ECR Repository.")
        press_enter_to_continue()
        print_dashes()

        print(
            "* As an optional step, you can interact with the image in Amazon ECR
 by using the CLI."
        )
        if q.ask(
            "Would you like to view instructions on how to use the CLI to run the
 image? (y/n)",
            q.is_yesno,
        ):
            print(
```

```
                f"""
1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you
 need to authenticate with the registry. You can do this using the AWS CLI:

    aws ecr get-login-password --region us-east-1 | docker login --username AWS
 --password-stdin {repository_uri.split("/")[0]}

2. Describe the image using this command:

   aws ecr describe-images --repository-name {self.repository_name} --image-ids
 imageTag={self.tag}

3. Run the Docker container and view the output using this command:

   docker run --rm {self.full_tag_name}
"""
            )

        self.cleanup(True)

    def cleanup(self, ask: bool):
        """
        Deletes the resources created in this scenario.
        :param ask: If True, prompts the user to confirm before deleting the
 resources.
        """
        if self.repository is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the ECR repository
 '{self.repository_name}? (y/n) "
            )
        ):
            print(f"Deleting the ECR repository '{self.repository_name}'.")
            self.ecr_wrapper.delete_repository(self.repository_name)

        if self.full_tag_name is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the local Docker image
 '{self.full_tag_name}? (y/n) "
            )
        ):
            print(f"Deleting the docker image '{self.full_tag_name}'.")
```

```
            self.docker_client.images.remove(self.full_tag_name)

    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )


    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
```

```
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )



if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Run Amazon ECR getting started scenario."
    )
    parser.add_argument(
        "--iam-role-arn",
        type=str,
        default=None,
        help="an optional IAM role ARN that will be granted access to download
 images from a repository.",
        required=False,
    )
    parser.add_argument(
        "--no-art",
        action="store_true",
        help="accessibility setting that suppresses art in the console output.",
    )
    args = parser.parse_args()
    no_art = args.no_art
    iam_role_arn = args.iam_role_arn
    demo = None
    a_docker_client = None
    try:
        a_docker_client = docker.from_env()
        if not a_docker_client.ping():
            raise docker.errors.DockerException("Docker is not running.")
    except docker.errors.DockerException as err:
        logging.error(
            """
        The Python Docker client could not be created.
        Do you have Docker installed and running?
        Here is the error message:
        %s
        """,
            err,
        )
```

```
        sys.exit("Error with Docker.")
    try:
        an_ecr_wrapper = ECRWrapper.from_client()
        demo = ECRGettingStarted(an_ecr_wrapper, a_docker_client)
        demo.run(iam_role_arn)

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup(False)
```

Classe ECRWrapper que encapsula ações do Amazon ECR.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
 self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] ==
 "RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
```

```
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

```
    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.
```

```
        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
```

```
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Ações para o Amazon ECR usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon ECR com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API Amazon ECR e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para o Amazon ECR usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do Amazon Elastic Container Registry.

Exemplos

- Use CreateRepositorycom um AWS SDK ou CLI
- Use DeleteRepositorycom um AWS SDK ou CLI
- Use DescribeImagescom um AWS SDK ou CLI

- Use DescribeRepositoriescom um AWS SDK ou CLI
- Use GetAuthorizationTokencom um AWS SDK ou CLI
- Use GetRepositoryPolicycom um AWS SDK ou CLI
- Use ListImagescom um AWS SDK ou CLI
- Use PushImageCmdcom um AWS SDK
- Use PutLifeCyclePolicycom um AWS SDK ou CLI
- Use SetRepositoryPolicycom um AWS SDK ou CLI
- Use StartLifecyclePolicyPreviewcom um AWS SDK ou CLI

Use **`CreateRepository`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateRepository`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de registros de contêineres

CLI

AWS CLI

Exemplo 1: criar um repositório

O exemplo `create-repository` a seguir cria um repositório dentro do namespace especificado no registro padrão de uma conta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo
```

Saída:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
```

```
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo"
    }
}
```

Para obter mais informações, consulte Creating a repository no Guia do usuário do Amazon ECR.

Exemplo 2: criar um repositório configurado com imutabilidade da tag de imagem

O exemplo `create-repository` a seguir cria um repositório configurado para imutabilidade de tags no registro padrão de uma conta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-tag-mutability IMMUTABLE
```

Saída:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageTagMutability": "IMMUTABLE"
    }
}
```

Para obter mais informações, consulte Mutabilidade da tag de imagem no Guia do usuário do Amazon ECR.

Exemplo 3: criar um repositório configurado com uma configuração de digitalização

O exemplo de `create-repository` a seguir cria um repositório configurado para realizar uma verificação de vulnerabilidade no envio de imagens por push no registro padrão de uma conta.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-scanning-configuration scanOnPush=true
```

Saída:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageScanningConfiguration": {
            "scanOnPush": true
        }
    }
}
```

Para obter mais informações, consulte Verificação de Imagens no Guia do Usuário do Amazon ECR.

- Para obter detalhes da API, consulte CreateRepositoryem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
```

```
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return
describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateRepositorya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
    * @throws RepositoryAlreadyExistsException if the repository exists.
    * @throws EcrException         if an error occurs while creating the
repository.
    */
   suspend fun createECRRepository(repoName: String?): String? {
       val request =
           CreateRepositoryRequest {
               repositoryName = repoName
           }

       return try {
           EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
               val response = ecrClient.createRepository(request)
               response.repository?.repositoryArn
           }
       } catch (e: RepositoryAlreadyExistsException) {
           println("Repository already exists: $repoName")
           repoName?.let { getRepoARN(it) }
       } catch (e: EcrException) {
```

```
            println("An error occurred: ${e.message}")
            null
        }
    }
```

- Para obter detalhes da API, consulte a CreateRepositoryreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
```

```
        """
        try:
            response =
self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a CreateRepositoryReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_repository_name = 'my-repository'
        oo_result = lo_ecr->createrepository(
          iv_repositoryname = iv_repository_name ).
```

```
        DATA(lv_repository_uri) = oo_result->get_repository( )-
>get_repositoryuri( ).
        MESSAGE |Repository created with URI: { lv_repository_uri }| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositoryalrexex.
        " If repository already exists, retrieve it
        DATA lt_repo_names TYPE /aws1/
cl_ecrrepositorynamels00=>tt_repositorynamelist.
        APPEND NEW /aws1/cl_ecrrepositorynamels00( iv_value =
 iv_repository_name ) TO lt_repo_names.
        DATA(lo_describe_result) = lo_ecr-
>describerepositories( it_repositorynames = lt_repo_names ).
        DATA(lt_repos) = lo_describe_result->get_repositories( ).
        IF lines( lt_repos ) > 0.
          READ TABLE lt_repos INDEX 1 INTO DATA(lo_repo).
          oo_result = NEW /aws1/cl_ecrcrerepositoryrsp( io_repository =
 lo_repo ).
          MESSAGE |Repository { iv_repository_name } already exists.| TYPE 'I'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateRepositoryreferência da API AWS SDK for SAP ABAP.

Use **`DeleteRepository`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteRepository`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de registros de contêineres

CLI

AWS CLI

Para excluir um repositório

A força de comando do exemplo de `delete-repository` a seguir exclui o repositório especificado no registro padrão de uma conta. O sinalizador `--force` será obrigatório se o repositório contiver imagens.

```
aws ecr delete-repository \
    --repository-name ubuntu \
    --force
```

Saída:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
    }
}
```

Para obter mais informações, consulte Excluir um repositório no Guia do usuário do Amazon ECR.

- Para obter detalhes da API, consulte DeleteRepositoryem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
```

```
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " +
repoName + " repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }
```

- Para obter detalhes da API, consulte DeleteRepositorya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- Para obter detalhes da API, consulte a DeleteRepositoryreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteRepositoryReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_repository_name = 'my-repository'
    lo_ecr->deleterepository(
      iv_repositoryname = iv_repository_name
      iv_force = abap_true ).
    MESSAGE |Repository { iv_repository_name } deleted.| TYPE 'I'.
  CATCH /aws1/cx_ecrrepositorynotfndex.
    MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteRepositoryreferência da API AWS SDK for SAP ABAP.

Use **`DescribeImages`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeImages`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para descrever uma imagem em um repositório

O exemplo de `describe-images` a seguir exibe os detalhes sobre uma imagem no repositório `cluster-autoscaler` com a tag `v1.13.6`.

```
aws ecr describe-images \
    --repository-name cluster-autoscaler \
    --image-ids imageTag=v1.13.6
```

Saída:

```
{
    "imageDetails": [
        {
            "registryId": "012345678910",
            "repositoryName": "cluster-autoscaler",
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTags": [
                "v1.13.6"
            ],
            "imageSizeInBytes": 48318255,
            "imagePushedAt": 1565128275.0
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeImagesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               } else {
                   throw new RuntimeException("Unexpected error: " +
ex.getCause());
```

```
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Para obter detalhes da API, consulte DescribeImagesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
```

```
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a DescribeImagesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeImagesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       " iv_repository_name = 'my-repository'
       " it_image_ids = VALUE #( ( NEW /aws1/cl_ecrimageidentifier( iv_imagetag
= 'latest' ) ) )
       IF it_image_ids IS NOT INITIAL.
         oo_result = lo_ecr->describeimages(
           iv_repositoryname = iv_repository_name
           it_imageids = it_image_ids ).
       ELSE.
         oo_result = lo_ecr->describeimages(
           iv_repositoryname = iv_repository_name ).
       ENDIF.
       DATA(lt_image_details) = oo_result->get_imagedetails( ).
       MESSAGE |Found { lines( lt_image_details ) } images in repository.| TYPE
'I'.
     CATCH /aws1/cx_ecrrepositorynotfndex.
       MESSAGE 'Repository not found.' TYPE 'I'.
     CATCH /aws1/cx_ecrimagenotfoundex.
       MESSAGE 'Image not found.' TYPE 'I'.
     CATCH /aws1/cx_ecrinvalidparameterex.
       MESSAGE 'Invalid parameter provided.' TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeImagesreferência da API AWS SDK for SAP ABAP.

Use **`DescribeRepositories`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeRepositories`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como descrever os repositórios em um registro

Este exemplo descreve os repositórios no registro padrão de uma conta.

Comando:

```
aws ecr describe-repositories
```

Saída:

```
{
    "repositories": [
        {
            "registryId": "012345678910",
            "repositoryName": "ubuntu",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/
ubuntu"
        },
        {
            "registryId": "012345678910",
            "repositoryName": "test",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/test"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeRepositoriesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the repository URI for the specified repository name.
    *
    * @param repoName the name of the repository to retrieve the URI for.
    * @return the repository URI for the specified repository name.
    * @throws EcrException        if there is an error retrieving the repository
information.
    * @throws CompletionException if the asynchronous operation completes
exceptionally.
    */
   public void getRepositoryURI(String repoName) {
       DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
           .repositoryNames(repoName)
           .build();

       CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
       response.whenComplete((describeRepositoriesResponse, ex) -> {
           if (ex != null) {
               Throwable cause = ex.getCause();
               if (cause instanceof InterruptedException) {
                   Thread.currentThread().interrupt();
                   String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " +
cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
```

```
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }
```

- Para obter detalhes da API, consulte DescribeRepositoriesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
```

```
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
 ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
 describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- Para obter detalhes da API, consulte a DescribeRepositoriesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
```

```
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeRepositoriesReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(),
 aws_sdk_ecr::Error> {
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!("  ARN:  {}", repo.repository_arn().unwrap());
        println!("  Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeRepositoriesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " it_repository_names = VALUE #( ( NEW /aws1/
cl_ecrrepositorynamels00( iv_value = 'my-repository' ) ) )
        oo_result = lo_ecr->describerepositories(
          it_repositorynames = it_repository_names ).
        DATA(lt_repositories) = oo_result->get_repositories( ).
        MESSAGE |Found { lines( lt_repositories ) } repositories.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeRepositoriesreferência da API AWS SDK for SAP ABAP.

Use **`GetAuthorizationToken`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetAuthorizationToken`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para obter um token de autorização para o registro padrão

O comando do exemplo de `get-authorization-token` a seguir obtém um token de autorização para o registro padrão.

```
aws ecr get-authorization-token
```

Saída:

```
{
    "authorizationData": [
        {
```

```
            "authorizationToken": "QVdTOkN...",
            "expiresAt": 1448875853.241,
            "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetAuthorizationTokenem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the
console.
    * If an exception occurs, the method handles the exception and prints the
error message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
```

```
                AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }
```

- Para obter detalhes da API, consulte GetAuthorizationTokena Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
```

```
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a GetAuthorizationTokenreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)
```

```
    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a GetAuthorizationTokenReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ecr->getauthorizationtoken( ).
        DATA(lt_auth_data) = oo_result->get_authorizationdata( ).
        IF lines( lt_auth_data ) > 0.
          READ TABLE lt_auth_data INDEX 1 INTO DATA(lo_auth_data).
          DATA(lv_token) = lo_auth_data->get_authorizationtoken( ).
          MESSAGE 'Authorization token retrieved.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_ecrserverexception.
        MESSAGE 'Server exception occurred.' TYPE 'I'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetAuthorizationTokenreferência da API AWS SDK for SAP ABAP.

Use **`GetRepositoryPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetRepositoryPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para recuperar a política de repositório de um repositório

O exemplo de `get-repository-policy` a seguir exibe os detalhes sobre a política de repositório para o repositório `cluster-autoscaler`.

```
aws ecr get-repository-policy \
    --repository-name cluster-autoscaler
```

Saída:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Para obter detalhes da API, consulte GetRepositoryPolicyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex);
               }
           }
```

```
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }
```

- Para obter detalhes da API, consulte GetRepositoryPolicya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
```

```
    }
```

- Para obter detalhes da API, consulte a GetRepositoryPolicyreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
```

```
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a GetRepositoryPolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_repository_name = 'my-repository'
        oo_result = lo_ecr->getrepositorypolicy(
          iv_repositoryname = iv_repository_name ).
        DATA(lv_policy_text) = oo_result->get_policytext( ).
        MESSAGE 'Repository policy retrieved.' TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrrepositoryplynot00.
```

```
        MESSAGE 'Repository policy not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetRepositoryPolicyreferência da API AWS SDK for SAP ABAP.

Use **`ListImages`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListImages`.

CLI

AWS CLI

Para listar as imagens em um repositório

O exemplo de `list-images` a seguir exibe uma lista das imagens presentes no repositório `cluster-autoscaler`.

```
aws ecr list-images \
    --repository-name cluster-autoscaler
```

Saída:

```
{
    "imageIds": [
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.8"
        },
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.7"
        },
        {
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTag": "v1.13.6"
```

```
        }
    ]
}
```

- Para obter detalhes da API, consulte ListImagesem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_images(
    client: &aws_sdk_ecr::Client,
    repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
    let rsp = client
        .list_images()
        .repository_name(repository)
        .send()
        .await?;

    let images = rsp.image_ids();

    println!("found {} images", images.len());

    for image in images {
        println!(
            "image: {}:{}",
            image.image_tag().unwrap(),
            image.image_digest().unwrap()
        );
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListImagesreferência da API AWS SDK for Rust.

Use **PushImageCmd**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PushImageCmd`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName  the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   public void pushDockerImage(String repoName, String imageName) {
       System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
       CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
           .thenApply(response -> {
               String token =
response.authorizationData().get(0).authorizationToken();
               String decodedToken = new
String(Base64.getDecoder().decode(token));
               String password = decodedToken.substring(4);

               DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
               Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
               assert repoData != null;
               String registryURL = repoData.repositoryUri().split("/")[0];
```

```
                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- Para obter detalhes da API, consulte PushImageCmda Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   suspend fun pushDockerImage(
       repoName: String,
       imageName: String,
   ) {
       println("Pushing $imageName to $repoName will take a few seconds")
       val authConfig = getAuthConfig(repoName)

       EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
           val desRequest =
               DescribeRepositoriesRequest {
                   repositoryNames = listOf(repoName)
               }

           val describeRepoResponse = ecrClient.describeRepositories(desRequest)
           val repoData =
               describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                   ?: throw RuntimeException("Repository not found: $repoName")

           val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
           if (tagImageCmd != null) {
               tagImageCmd.exec()
           }
```

```
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

- Para obter detalhes da API, consulte a PushImageCmdreferência da API AWS SDK for Kotlin.

Use **`PutLifeCyclePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutLifeCyclePolicy`.

CLI

AWS CLI

Para criar uma política de ciclo de vida

O exemplo `put-lifecycle-policy` a seguir cria uma política de ciclo de vida para o repositório especificado no registro padrão de uma conta.

```
aws ecr put-lifecycle-policy \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Conteúdo de `policy.json`:

```
{
```

```
    "rules": [
        {
            "rulePriority": 1,
            "description": "Expire images older than 14 days",
            "selection": {
                "tagStatus": "untagged",
                "countType": "sinceImagePushed",
                "countUnit": "days",
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Saída:

```
{
   "registryId": "<aws_account_id>",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\"rules\":[{\"rulePriority\":1,\"description\":
\"Expire images older than 14 days\",\"selection\":{\"tagStatus\":\"untagged\",
\"countType\":\"sinceImagePushed\",\"countUnit\":\"days\",\"countNumber\":14},
\"action\":{\"type\":\"expire\"}}]}"
}
```

Para obter mais informações, consulte Políticas de ciclo de vida no Guia do usuário do Amazon ECR.

- Para obter detalhes da API, consulte PutLifeCyclePolicyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
 str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
 policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
```

```
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

Exemplo que coloca uma política de data de validade.

```
    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Para obter detalhes da API, consulte a PutLifeCyclePolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_repository_name = 'my-repository'
        " iv_lifecycle_policy_text = '{"rules":
[{"rulePriority":1,"description":"Expire images older than 14 days",...}]}'
        lo_ecr->putlifecyclepolicy(
          iv_repositoryname = iv_repository_name
          iv_lifecyclepolicytext = iv_lifecycle_policy_text ).
        MESSAGE |Lifecycle policy set for repository { iv_repository_name }.|
 TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrvalidationex.
        MESSAGE 'Invalid lifecycle policy format.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutLifeCyclePolicyreferência da API AWS SDK for SAP ABAP.

Use **`SetRepositoryPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetRepositoryPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para definir a política de repositório para um repositório

O exemplo `set-repository-policy` a seguir anexa uma política de repositório contida em um arquivo ao repositório `cluster-autoscaler`.

```
aws ecr set-repository-policy \
    --repository-name cluster-autoscaler \
    --policy-text file://my-policy.json
```

Conteúdo de `my-policy.json`:

```
{
    "Version":"2012-10-17",
    "Statement" : [
        {
            "Sid" : "allow public pull",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        }
    ]
}
```

Saída:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Para obter detalhes da API, consulte SetRepositoryPolicyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Sets the repository policy for the specified ECR repository.
    *
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
    * @throws EcrException                      if there is an unexpected error
setting the repository policy.
    */
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
           {
            "Version":"2012-10-17",
            "Statement" : [ {
              "Sid" : "new statement",
              "Effect" : "Allow",
              "Principal" : {
                "AWS" : "%s"
              },
              "Action" : "ecr:BatchGetImage"
```

```
              } ]
            }
             """;

        String policyDocument = String.format(policyDocumentTemplate, iamRole);
        SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .policyText(policyDocument)
            .build();

        CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy set successfully.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof RepositoryPolicyNotFoundException) {
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }
```

- Para obter detalhes da API, consulte SetRepositoryPolicya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
    val policyDocumentTemplate =
        """
         {
          "Version":"2012-10-17",
          "Statement" : [ {
            "Sid" : "new statement",
            "Effect" : "Allow",
            "Principal" : {
              "AWS" : "$iamRole"
            },
            "Action" : "ecr:BatchGetImage"
          } ]
        }

        """.trimIndent()
    val setRepositoryPolicyRequest =
        SetRepositoryPolicyRequest {
            repositoryName = repoName
            policyText = policyDocumentTemplate
        }
```

```
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
 ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a SetRepositoryPolicyreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
```

```
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

Exemplo que concede acesso de download a um perfil do IAM.

```
    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
```

```
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Para obter detalhes da API, consulte a SetRepositoryPolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_repository_name = 'my-repository'
    " iv_policy_text = '{"Version":"2012-10-17",       "Statement":[...]}'
    lo_ecr->setrepositorypolicy(
      iv_repositoryname = iv_repository_name
      iv_policytext = iv_policy_text ).
    MESSAGE |Policy set for repository { iv_repository_name }.| TYPE 'I'.
  CATCH /aws1/cx_ecrrepositorynotfndex.
    MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a SetRepositoryPolicyreferência da API AWS SDK for SAP ABAP.

Use **StartLifecyclePolicyPreview**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartLifecyclePolicyPreview`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para criar uma pré-visualização da política de ciclo de vida

O exemplo de `start-lifecycle-policy-preview` a seguir cria uma pré-visualização prévia da política de ciclo de vida definida por um arquivo JSON para o repositório especificado.

```
aws ecr start-lifecycle-policy-preview \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Conteúdo de `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
               "countNumber": 14
           },
           "action": {
               "type": "expire"
           }
       }
   ]
```

```
}
```

Saída:

```
{
   "registryId": "012345678910",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\n    \"rules\": [\n        {\n
 \"rulePriority\": 1,\n            \"description\": \"Expire images older than 14
 days\",\n            \"selection\": {\n                \"tagStatus\": \"untagged
\",\n                \"countType\": \"sinceImagePushed\",\n
 \"countUnit\": \"days\",\n                \"countNumber\": 14\n            },\n
           \"action\": {\n                \"type\": \"expire\"\n            }\n
      }\n    ]\n}\n",
   "status": "IN_PROGRESS"
}
```

- Para obter detalhes da API, consulte StartLifecyclePolicyPreviewem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
```

```
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Para obter detalhes da API, consulte StartLifecyclePolicyPreviewa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
```

```
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a StartLifecyclePolicyPreviewreferência da API AWS SDK for Kotlin.

Cenários para o Amazon ECR usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon ECR com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon ECR ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos de registros de contêineres

Conceitos básicos de registros de contêineres

O exemplo de código a seguir mostra como:

- Criar uma imagem do Docker.
- Criar um repositório do Amazon ECR.
- Excluir recursos.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "==================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "==================================================="

# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-
cliv2.html for installation instructions."
    exit 1
fi
```

```
# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure'
 to set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running
 this script."
    echo "Visit https://docs.docker.com/get-docker/ for installation
 instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try
 again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"

    echo "==================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "==================================================="
```

```
    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "==================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force ||
 echo "Failed to delete repository, it may not exist or may have already been
 deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local
 image, it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove
 tagged image, it may not exist or may have already been deleted."
    fi

    echo "Cleanup completed."
    echo "==================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
```

```
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build
 Docker image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list
 Docker images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"

# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials
 are configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"
```

```
# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --
tags key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-
registry-gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username
 AWS --password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" ||
 handle_error "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"

# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
```

```
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image
 to ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged
 image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "====================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "====================================================="

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]];
 then
```

```
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-
ids imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "==================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - BatchDeleteImage
  - CreateRepository
  - DeleteRepository
  - GetCallerIdentity

- GetLoginPassword

Exemplos de código para o Amazon ECS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Elastic Container Service com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon ECS: mais informações sobre o Amazon ECS.
- Referência da API Amazon ECS: detalhes sobre todas as ações disponíveis do Amazon ECS.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon ECS usando AWS SDKs
  - Olá, Amazon ECS
  - Ações para o Amazon ECS usando AWS SDKs
    - Use CreateClustercom um AWS SDK ou CLI
    - Use CreateServicecom um AWS SDK ou CLI
    - Use DeleteClustercom um AWS SDK ou CLI
    - Use DeleteServicecom um AWS SDK ou CLI
    - Use DescribeClusterscom um AWS SDK ou CLI
    - Usar DescribeServices com uma CLI
    - Use DescribeTaskscom um AWS SDK ou CLI

    - Use ListClusterscom um AWS SDK ou CLI
    - Use ListServicescom um AWS SDK ou CLI
    - Use ListTaskscom um AWS SDK ou CLI
    - Usar UpdateClusterSettings com uma CLI
    - Use UpdateServicecom um AWS SDK ou CLI
- Cenários para o Amazon ECS usando AWS SDKs
  - Configurar a conectividade do serviço de contêiner
  - Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
  - Como criar um serviço de contêiner para instâncias de máquinas virtuais
  - Obtenha informações de ARN para clusters, serviços e tarefas do Amazon ECS usando um AWS SDK

Exemplos básicos para o Amazon ECS usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Elastic Container Service com AWS SDKs.

Exemplos

- Olá, Amazon ECS
- Ações para o Amazon ECS usando AWS SDKs
  - Use CreateClustercom um AWS SDK ou CLI
  - Use CreateServicecom um AWS SDK ou CLI
  - Use DeleteClustercom um AWS SDK ou CLI
  - Use DeleteServicecom um AWS SDK ou CLI
  - Use DescribeClusterscom um AWS SDK ou CLI
  - Usar DescribeServices com uma CLI
  - Use DescribeTaskscom um AWS SDK ou CLI
  - Use ListClusterscom um AWS SDK ou CLI
  - Use ListServicescom um AWS SDK ou CLI
  - Use ListTaskscom um AWS SDK ou CLI
  - Usar UpdateClusterSettings com uma CLI
  - Use UpdateServicecom um AWS SDK ou CLI

Olá, Amazon ECS

O exemplo de código a seguir mostra como começar a usar o Amazon ECS.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.ECS;
using Amazon.ECS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace ECSActions;

/// <summary>
/// A class that introduces the Amazon ECS Client by listing the
/// cluster ARNs for the account.
/// </summary>
public class HelloECS
{
    static async System.Threading.Tasks.Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon ECS client.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonECS>()
            )
            .Build();

        var amazonECSClient = host.Services.GetRequiredService<IAmazonECS>();

        Console.WriteLine($"Hello Amazon ECS! Following are some cluster ARNS
 available in the your account");
        Console.WriteLine();

        var clusters = new List<string>();

        var clustersPaginator = amazonECSClient.Paginators.ListClusters(new
 ListClustersRequest());

        await foreach (var response in clustersPaginator.Responses)
        {
            clusters.AddRange(response.ClusterArns);
        }

        if (clusters.Count > 0)
        {
            clusters.ForEach(cluster =>
            {
                Console.WriteLine($"\tARN: {cluster}");
                Console.WriteLine($"Cluster Name: {cluster.Split("/").Last()}");
                Console.WriteLine();
            });
        }
        else
        {
            Console.WriteLine("No clusters were found.");
        }

    }
}
```

- Para obter detalhes da API, consulte ListClustersa Referência AWS SDK para .NET da API.

Ações para o Amazon ECS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon ECS com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon ECS e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon ECS usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Elastic Container Service.

Exemplos

- Use CreateClustercom um AWS SDK ou CLI
- Use CreateServicecom um AWS SDK ou CLI
- Use DeleteClustercom um AWS SDK ou CLI
- Use DeleteServicecom um AWS SDK ou CLI
- Use DescribeClusterscom um AWS SDK ou CLI
- Usar DescribeServices com uma CLI
- Use DescribeTaskscom um AWS SDK ou CLI
- Use ListClusterscom um AWS SDK ou CLI
- Use ListServicescom um AWS SDK ou CLI
- Use ListTaskscom um AWS SDK ou CLI
- Usar UpdateClusterSettings com uma CLI
- Use UpdateServicecom um AWS SDK ou CLI

Use **`CreateCluster`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

- Como criar um serviço de contêiner para instâncias de máquinas virtuais

CLI

AWS CLI

Exemplo 1: como criar um cluster

O `create-cluster` exemplo a seguir cria um cluster chamado `MyCluster` e ativa o CloudWatch Container Insights com observabilidade aprimorada.

```
aws ecs create-cluster \
    --cluster-name MyCluster \
    --settings name=containerInsights,value=enhanced
```

Saída:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "enhanced"
            }
        ],
        "tags": []
    }
}
```

Para obter mais informações, consulte Creating a Cluster no Guia do desenvolvedor do Amazon ECS.

Exemplo 2: como criar um cluster usando provedores de capacidade

O exemplo de `create-cluster` a seguir cria um cluster e associa dois provedores de capacidade existentes a ele. O comando `create-capacity-provider` é usado para criar um provedor de capacidade. Embora recomendado, não é necessário especificar uma estratégia de provedor de capacidade. Neste exemplo, criamos um cluster chamado `MyCluster` e associamos os provedores de capacidade `MyCapacityProvider1` e `MyCapacityProvider2` a ele. Uma estratégia de provedor de capacidade padrão é especificada e distribui a tarefa igualmente em ambos os provedores de capacidade.

```
aws ecs create-cluster \
    --cluster-name MyCluster \
    --capacity-providers MyCapacityProvider1 MyCapacityProvider2 \
    --default-capacity-provider-
strategy capacityProvider=MyCapacityProvider1,weight=1 capacityProvider=MyCapacityProvide
```

Saída:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "clusterName": "MyCluster",
        "status": "PROVISIONING",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "enabled"
            }
        ],
        "capacityProviders": [
            "MyCapacityProvider1",
            "MyCapacityProvider2"
        ],
        "defaultCapacityProviderStrategy": [
            {
                "capacityProvider": "MyCapacityProvider1",
                "weight": 1,
                "base": 0
            },
```

```
            {
                "capacityProvider": "MyCapacityProvider2",
                "weight": 1,
                "base": 0
            }
        ],
        "attachments": [
           {
               "id": "0fb0c8f4-6edd-4de1-9b09-17e470ee1918",
               "type": "asp",
               "status": "PRECREATED",
               "details": [
                   {
                       "name": "capacityProviderName",
                       "value": "MyCapacityProvider1"
                   },
                   {
                       "name": "scalingPlanName",
                       "value": "ECSManagedAutoScalingPlan-a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111"
                   }
               ]
            },
            {
                "id": "ae592060-2382-4663-9476-b015c685593c",
                "type": "asp",
                "status": "PRECREATED",
                "details": [
                    {
                        "name": "capacityProviderName",
                        "value": "MyCapacityProvider2"
                    },
                    {
                        "name": "scalingPlanName",
                        "value": "ECSManagedAutoScalingPlan-a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222"
                    }
                ]
            }
        ],
        "attachmentsStatus": "UPDATE_IN_PROGRESS"
    }
}
```

Para obter mais informações, consulte provedores de capacidade de clusters no Guia do desenvolvedor do Amazon ECS.

Exemplo 3: como criar um cluster com várias tags

O exemplo `create-cluster` a seguir cria um cluster com várias tags. Para obter mais informações sobre como adicionar tags usando a sintaxe abreviada, consulte Usando a sintaxe abreviada com a interface de AWS linha de comando no Guia do usuário da CLI.AWS

```
aws ecs create-cluster \
    --cluster-name MyCluster \
    --tags key=key1,value=value1 key=key2,value=value2
```

Saída:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [
            {
                "key": "key1",
                "value": "value1"
            },
            {
                "key": "key2",
                "value": "value2"
            }
        ]
     }
 }
```

Para obter mais informações, consulte Creating a Cluster no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte CreateClusterem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <clusterName>\s

                Where:
                   clusterName - The name of the ECS cluster to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String clusterName = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        String clusterArn = createGivenCluster(ecsClient, clusterName);
        System.out.println("The cluster ARN is " + clusterArn);
        ecsClient.close();
    }

    public static String createGivenCluster(EcsClient ecsClient, String
 clusterName) {
        try {
            ExecuteCommandConfiguration commandConfiguration =
 ExecuteCommandConfiguration.builder()
                    .logging(ExecuteCommandLogging.DEFAULT)
                    .build();

            ClusterConfiguration clusterConfiguration =
 ClusterConfiguration.builder()
                    .executeCommandConfiguration(commandConfiguration)
                    .build();

            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
                    .clusterName(clusterName)
                    .configuration(clusterConfiguration)
                    .build();

            CreateClusterResponse response =
 ecsClient.createCluster(clusterRequest);
            return response.cluster().clusterArn();

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

- Para obter detalhes da API, consulte CreateCluster a Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet cria um cluster do Amazon ECS.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Saída:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/
LAB-ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Para obter detalhes da API, consulte CreateCluster em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cria um cluster do Amazon ECS.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Saída:

```
ActiveServicesCount                : 0
Attachments                        : {}
AttachmentsStatus                  :
CapacityProviders                  : {}
ClusterArn                         : arn:aws:ecs:us-west-2:012345678912:cluster/
LAB-ECS-CL
ClusterName                        : LAB-ECS-CL
DefaultCapacityProviderStrategy    : {}
PendingTasksCount                  : 0
RegisteredContainerInstancesCount  : 0
RunningTasksCount                  : 0
Settings                           : {containerInsights}
Statistics                         : {}
Status                             : ACTIVE
Tags                               : {}
```

- Para obter detalhes da API, consulte CreateClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_cluster(client: &aws_sdk_ecs::Client, name: &str) -> Result<(),
 aws_sdk_ecs::Error> {
    let cluster = client.create_cluster().cluster_name(name).send().await?;
    println!("cluster created: {:?}", cluster);

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateClusterreferência da API AWS SDK for Rust.

Use **`CreateService`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateService`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais

CLI

AWS CLI

Exemplo 1: como criar um serviço com uma tarefa do Fargate

O exemplo de `create-service` a seguir demonstra como criar um serviço usando uma tarefa do Fargate.

```
aws ecs create-service \
    --cluster MyCluster \
    --service-name MyService \
    --task-definition sample-fargate:1 \
    --desired-count 2 \
    --launch-type FARGATE \
    --platform-version LATEST \
    --network-configuration
 'awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPubli
 \
    --tags key=key1,value=value1 key=key2,value=value2 key=key3,value=value3
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/
MyService",
        "serviceName": "MyService",
          "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
```

```
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
sample-fargate:1",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "id": "ecs-svc/1234567890123456789",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/sample-fargate:1",
                "desiredCount": 2,
                "pendingCount": 0,
                "runningCount": 0,
                "createdAt": 1557119253.821,
                "updatedAt": 1557119253.821,
                "launchType": "FARGATE",
                "platformVersion": "1.3.0",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                }
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
        "createdAt": 1557119253.821,
        "placementConstraints": [],
        "placementStrategy": [],
```

```
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321"
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "schedulingStrategy": "REPLICA",
        "tags": [
            {
                "key": "key1",
                "value": "value1"
            },
            {
                "key": "key2",
                "value": "value2"
            },
            {
                "key": "key3",
                "value": "value3"
            }
        ],
        "enableECSManagedTags": false,
        "propagateTags": "NONE"
    }
}
```

Para obter mais informações, consulte Creating a Service no Guia do desenvolvedor do Amazon ECS.

Exemplo 2: como criar um serviço usando o tipo de inicialização do EC2

O exemplo de `create-service` a seguir mostra como criar um serviço chamado `ecs-simple-service` com uma tarefa que usa o tipo de inicialização do EC2. O serviço usa a definição de tarefa `sleep360` e mantem uma instanciação da tarefa.

```
aws ecs create-service \
    --cluster MyCluster \
    --service-name ecs-simple-service \
```

```
    --task-definition sleep360:2 \
    --desired-count 1
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/ecs-
simple-service",
        "serviceName": "ecs-simple-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "EC2",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
sleep360:2",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "id": "ecs-svc/1234567890123456789",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/sleep360:2",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "createdAt": 1557206498.798,
                "updatedAt": 1557206498.798,
                "launchType": "EC2"
            }
        ],
        "events": [],
        "createdAt": 1557206498.798,
        "placementConstraints": [],
        "placementStrategy": [],
        "schedulingStrategy": "REPLICA",
```

```
        "enableECSManagedTags": false,
        "propagateTags": "NONE"
    }
}
```

Para obter mais informações, consulte Creating a Service no Guia do desenvolvedor do Amazon ECS.

Exemplo 3: como criar um serviço que usa um controlador de implantação externo

O exemplo de `create-service` a seguir cria um serviço que usa um controlador de implantação externo.

```
aws ecs create-service \
    --cluster MyCluster \
    --service-name MyService \
    --deployment-controller type=EXTERNAL \
    --desired-count 1
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/
MyService",
        "serviceName": "MyService",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "EC2",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "taskSets": [],
        "deployments": [],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
```

```
        "createdAt": 1557128207.101,
        "placementConstraints": [],
        "placementStrategy": [],
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "EXTERNAL"
        },
        "enableECSManagedTags": false,
        "propagateTags": "NONE"
    }
}
```

Para obter mais informações, consulte Creating a Service no Guia do desenvolvedor do Amazon ECS.

Exemplo 4: como criar um serviço por trás de um balanceador de carga

O exemplo de `create-service` a seguir mostra como criar um serviço que fique por trás de um balanceador de carga. É necessário ter um balanceador de carga configurado na mesma região da instância de contêiner. Este exemplo usa a opção `--cli-input-json` e um arquivo de entrada JSON chamado `ecs-simple-service-elb.json` com o conteúdo a seguir.

```
aws ecs create-service \
    --cluster MyCluster \
    --service-name ecs-simple-service-elb \
    --cli-input-json file://ecs-simple-service-elb.json
```

Conteúdo de `ecs-simple-service-elb.json`:

```
 {
    "serviceName": "ecs-simple-service-elb",
    "taskDefinition": "ecs-demo",
    "loadBalancers": [
        {
            "loadBalancerName": "EC2Contai-EcsElast-123456789012",
            "containerName": "simple-demo",
            "containerPort": 80
        }
    ],
    "desiredCount": 10,
    "role": "ecsServiceRole"
```

```
}
```

Saída:

```
{
    "service": {
        "status": "ACTIVE",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
ecs-demo:1",
        "pendingCount": 0,
        "loadBalancers": [
            {
                "containerName": "ecs-demo",
                "containerPort": 80,
                "loadBalancerName": "EC2Contai-EcsElast-123456789012"
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/ecsServiceRole",
        "desiredCount": 10,
        "serviceName": "ecs-simple-service-elb",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/ecs-simple-
service-elb",
        "deployments": [
            {
                "status": "PRIMARY",
                "pendingCount": 0,
                "createdAt": 1428100239.123,
                "desiredCount": 10,
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/ecs-demo:1",
                "updatedAt": 1428100239.123,
                "id": "ecs-svc/1234567890123456789",
                "runningCount": 0
            }
        ],
        "events": [],
        "runningCount": 0
    }
}
```

Para obter mais informações, consulte Uso do balanceamento de carga para distribuir o tráfego de serviço do Amazon ECS no Guia do desenvolvedor do Amazon ECS.

Exemplo 5: como configurar volumes do Amazon EBS ao criar o serviço

O exemplo de `create-service` a seguir mostra como criar volumes do EBS para cada tarefa gerenciada pelo serviço. Você deve ter uma função de infraestrutura do Amazon ECS configurada com a política `AmazonECSInfrastructureRolePolicyForVolumes` gerenciada anexada. É necessário especificar uma definição de tarefa com o mesmo nome de volume da solicitação `create-service`. Este exemplo usa a opção `--cli-input-json` e um arquivo de entrada JSON chamado `ecs-simple-service-ebs.json` com o conteúdo a seguir.

```
aws ecs create-service \
    --cli-input-json file://ecs-simple-service-ebs.json
```

Conteúdo de `ecs-simple-service-ebs.json`:

```
{
    "cluster": "mycluster",
    "taskDefinition": "mytaskdef",
    "serviceName": "ecs-simple-service-ebs",
    "desiredCount": 2,
    "launchType": "FARGATE",
    "networkConfiguration":{
        "awsvpcConfiguration":{
            "assignPublicIp": "ENABLED",
            "securityGroups": ["sg-12344321"],
            "subnets":["subnet-12344321"]
        }
    },
    "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
                "roleArn":"arn:aws:iam::123456789012:role/ecsInfrastructureRole",
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "iops": 3000,
                "throughput": 125,
                "filesystemType": "ext4"
            }
        }
   ]
```

```
}
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/mycluster/ecs-
simple-service-ebs",
        "serviceName": "ecs-simple-service-ebs",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/mycluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "EC2",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
mytaskdef:3",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": false,
                "rollback": false
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "id": "ecs-svc/7851020056849183687",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/mytaskdef:3",
                "desiredCount": 0,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": "2025-01-21T11:32:38.034000-06:00",
                "updatedAt": "2025-01-21T11:32:38.034000-06:00",
                "launchType": "EC2",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
```

```
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "DISABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/7851020056849183687
 in progress.",
                "volumeConfigurations": [
                    {
                        "name": "myEBSVolume",
                        "managedEBSVolume": {
                            "volumeType": "gp3",
                            "sizeInGiB": 100,
                            "iops": 3000,
                            "throughput": 125,
                            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRole",
                            "filesystemType": "ext4"
                        }
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
        "createdAt": "2025-01-21T11:32:38.034000-06:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321"
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "DISABLED"
            }
        },
```

```
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:user/AIDACKCEVSQ6C2EXAMPLE",
        "enableECSManagedTags": false,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "DISABLED"
    }
}
```

Consulte mais informações em Uso de volumes do Amazon EBS com o Amazon ECS no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte CreateServiceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateService {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                  <clusterName> <serviceName> <securityGroups>
 <subnets> <taskDefinition>

                                Where:
                                  clusterName - The name of the ECS cluster.
                                  serviceName - The name of the ECS service to
 create.
                                  securityGroups - The name of the security
 group.
                                  subnets - The name of the subnet.
                                  taskDefinition - The name of the task
 definition.
                                """;

                if (args.length != 5) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String clusterName = args[0];
                String serviceName = args[1];
                String securityGroups = args[2];
                String subnets = args[3];
                String taskDefinition = args[4];
                Region region = Region.US_EAST_1;
                EcsClient ecsClient = EcsClient.builder()
                                .region(region)
                                .build();

                String serviceArn = createNewService(ecsClient, clusterName,
 serviceName, securityGroups, subnets,
                                taskDefinition);
                System.out.println("The ARN of the service is " + serviceArn);
                ecsClient.close();
        }
```

```
        public static String createNewService(EcsClient ecsClient,
                        String clusterName,
                        String serviceName,
                        String securityGroups,
                        String subnets,
                        String taskDefinition) {

                try {
                        AwsVpcConfiguration vpcConfiguration =
 AwsVpcConfiguration.builder()
                                        .securityGroups(securityGroups)
                                        .subnets(subnets)
                                        .build();

                        NetworkConfiguration configuration =
 NetworkConfiguration.builder()
                                        .awsvpcConfiguration(vpcConfiguration)
                                        .build();

                        CreateServiceRequest serviceRequest =
 CreateServiceRequest.builder()
                                        .cluster(clusterName)
                                        .networkConfiguration(configuration)
                                        .desiredCount(1)
                                        .launchType(LaunchType.FARGATE)
                                        .serviceName(serviceName)
                                        .taskDefinition(taskDefinition)
                                        .build();

                        CreateServiceResponse response =
 ecsClient.createService(serviceRequest);
                        return response.service().serviceArn();

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

- Para obter detalhes da API, consulte CreateServicea Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo de comando cria um serviço no cluster padrão chamado “ecs-simple-service”. O serviço usa a definição de tarefa “ecs-demo” e mantém 10 instanciações dessa tarefa.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Exemplo 2: esse exemplo de comando cria um serviço por trás de um balanceador de carga no cluster padrão chamado “ecs-simple-service”. O serviço usa a definição de tarefa “ecs-demo” e mantém 10 instanciações dessa tarefa.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- Para obter detalhes da API, consulte CreateServiceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo de comando cria um serviço no cluster padrão chamado “ecs-simple-service”. O serviço usa a definição de tarefa “ecs-demo” e mantém 10 instanciações dessa tarefa.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Exemplo 2: esse exemplo de comando cria um serviço por trás de um balanceador de carga no cluster padrão chamado “ecs-simple-service”. O serviço usa a definição de tarefa “ecs-demo” e mantém 10 instanciações dessa tarefa.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- Para obter detalhes da API, consulte CreateServiceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteCluster`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais

CLI

AWS CLI

Como excluir um cluster vazio

O exemplo de `delete-cluster` a seguir exclui o cluster vazio especificado.

```
aws ecs delete-cluster --cluster MyCluster
```

Saída:

```
{
```

```
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "status": "INACTIVE",
        "clusterName": "MyCluster",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0
        "statistics": [],
        "tags": []
    }
}
```

Para obter mais informações, consulte Deleting a Cluster no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte DeleteClusterem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet exclui o cluster do ECS especificado. Cancele o registro de todas as instâncias de contêiner desse cluster antes de excluí-las.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ECSCluster (DeleteCluster)" on target "LAB-ECS".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui o cluster do ECS especificado. Cancele o registro de todas as instâncias de contêiner desse cluster antes de excluí-las.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ECSCluster (DeleteCluster)" on target "LAB-ECS".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn remove_cluster(
    client: &aws_sdk_ecs::Client,
    name: &str,
) -> Result<(), aws_sdk_ecs::Error> {
    let cluster_deleted = client.delete_cluster().cluster(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteClusterreferência da API AWS SDK for Rust.

Use **`DeleteService`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteService`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais

CLI

AWS CLI

Como excluir um serviço

O exemplo de `ecs delete-service` a seguir exclui o serviço especificado de um cluster. Inclua o parâmetro `--force` para excluir um serviço mesmo que ele não tenha sido reduzido a zero tarefas.

```
aws ecs delete-service --cluster MyCluster --service MyService1 --force
```

Para obter mais informações, consulte Deleting a Service no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte DeleteServiceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
```

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

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

                Usage:
                  <clusterName> <serviceArn>\s

                Where:
                  clusterName - The name of the ECS cluster.
                  serviceArn - The ARN of the ECS service.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        deleteSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void deleteSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .build();
```

```
            ecsClient.deleteService(serviceRequest);
            System.out.println("The Service was successfully deleted");

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

- Para obter detalhes da API, consulte DeleteServicea Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui o serviço chamado “my-http-service” no cluster padrão. O serviço deve ter uma contagem desejada e uma contagem contínua de 0 antes que você possa excluí-lo. A confirmação será solicitada antes que o comando continue. Para ignorar a solicitação confirmação, adicione a opção -Force.

```
Remove-ECSService -Service my-http-service
```

Exemplo 2: exclui o serviço chamado “my-http-service” no cluster nomeado.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Para obter detalhes da API, consulte DeleteServiceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui o serviço chamado “my-http-service” no cluster padrão. O serviço deve ter uma contagem desejada e uma contagem contínua de 0 antes que você possa excluí-lo. A confirmação será solicitada antes que o comando continue. Para ignorar a solicitação confirmação, adicione a opção -Force.

```
Remove-ECSService -Service my-http-service
```

Exemplo 2: exclui o serviço chamado “my-http-service” no cluster nomeado.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Para obter detalhes da API, consulte DeleteServiceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeClusters`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeClusters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Configurar a conectividade do serviço de contêiner

CLI

AWS CLI

Exemplo 1: como descrever um cluster

O exemplo de `describe-clusters` a seguir recupera detalhes do cluster especificado.

```
aws ecs describe-clusters \
    --cluster default
```

Saída:

```
{
    "clusters": [
        {
            "status": "ACTIVE",
            "clusterName": "default",
            "registeredContainerInstancesCount": 0,
            "pendingTasksCount": 0,
            "runningTasksCount": 0,
            "activeServicesCount": 1,
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/default"
        }
    ],
```

```
    "failures": []
}
```

Para obter mais informações, consulte Clusters do Amazon ECS no Guia do desenvolvedor do Amazon ECS.

Exemplo 2: como descrever um cluster com a opção de anexo

O exemplo de `describe-clusters` a seguir especifica a opção de ANEXOS. Ele recupera detalhes sobre o cluster especificado e uma lista dos recursos anexados ao cluster no formato de anexos. Ao usar um provedor de capacidade com um cluster, os recursos, sejam AutoScaling planos ou políticas de escalabilidade, serão representados como ASP ou as_policy ATTACHMENTS.

```
aws ecs describe-clusters \
    --include ATTACHMENTS \
    --clusters sampleCluster
```

Saída:

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:af-south-1:123456789222:cluster/
sampleCluster",
            "clusterName": "sampleCluster",
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 0,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [],
            "settings": [],
            "capacityProviders": [
                "sampleCapacityProvider"
            ],
            "defaultCapacityProviderStrategy": [],
            "attachments": [
                {
                    "id": "a1b2c3d4-5678-901b-cdef-EXAMPLE22222",
                    "type": "as_policy",
```

```
                    "status": "CREATED",
                    "details": [
                        {
                            "name": "capacityProviderName",
                            "value": "sampleCapacityProvider"
                        },
                        {
                            "name": "scalingPolicyName",
                            "value": "ECSManagedAutoScalingPolicy-3048e262-
fe39-4eaf-826d-6f975d303188"
                        }
                    ]
                }
            ],
            "attachmentsStatus": "UPDATE_COMPLETE"
        }
    ],
    "failures": []
}
```

Para obter mais informações, consulte Clusters do Amazon ECS no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte DescribeClustersem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> \s

                Where:
                  clusterArn - The ARN of the ECS cluster to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
            DescribeClustersRequest clustersRequest =
 DescribeClustersRequest.builder()
                    .clusters(clusterArn)
                    .build();
```

```
            DescribeClustersResponse response =
 ecsClient.describeClusters(clustersRequest);
            List<Cluster> clusters = response.clusters();
            for (Cluster cluster : clusters) {
                System.out.println("The cluster name is " +
 cluster.clusterName());
            }

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

- Para obter detalhes da API, consulte DescribeClustersa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet descreve um ou mais de seus clusters do ECS.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Saída:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- Para obter detalhes da API, consulte DescribeClustersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet descreve um ou mais de seus clusters do ECS.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Saída:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- Para obter detalhes da API, consulte DescribeClustersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_clusters(client: &aws_sdk_ecs::Client) -> Result<(),
 aws_sdk_ecs::Error> {
    let resp = client.list_clusters().send().await?;

    let cluster_arns = resp.cluster_arns();
    println!("Found {} clusters:", cluster_arns.len());

    let clusters = client
        .describe_clusters()
        .set_clusters(Some(cluster_arns.into()))
        .send()
        .await?;

    for cluster in clusters.clusters() {
        println!("  ARN:  {}", cluster.cluster_arn().unwrap());
        println!("  Name: {}", cluster.cluster_name().unwrap());
```

```
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeClustersreferência da API AWS SDK for Rust.

Usar **`DescribeServices`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeServices`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais

CLI

AWS CLI

Para descrever um serviço

O exemplo `describe-services` a seguir recupera detalhes do serviço `my-http-service` no cluster padrão.

```
aws ecs describe-services --services my-http-service
```

Saída:

```
{
    "services": [
        {
            "status": "ACTIVE",
            "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/amazon-ecs-sample:1",
```

```
            "pendingCount": 0,
            "loadBalancers": [],
            "desiredCount": 10,
            "createdAt": 1466801808.595,
            "serviceName": "my-http-service",
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/default",
            "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/my-http-
service",
            "deployments": [
                {
                    "status": "PRIMARY",
                    "pendingCount": 0,
                    "createdAt": 1466801808.595,
                    "desiredCount": 10,
                    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/amazon-ecs-sample:1",
                    "updatedAt": 1428326312.703,
                    "id": "ecs-svc/1234567890123456789",
                    "runningCount": 10
                }
            ],
            "events": [
                {
                    "message": "(service my-http-service) has reached a steady
 state.",
                    "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
                    "createdAt": 1466801812.435
                }
            ],
            "runningCount": 10
        }
    ],
    "failures": []
}
```

Para obter mais informações, consulte Serviços no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte DescribeServicesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra como recuperar detalhes de um serviço específico do cluster padrão.

```
Get-ECSService -Service my-hhtp-service
```

Exemplo 2: esse exemplo mostra como recuperar detalhes de um serviço específico em execução no cluster nomeado.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- Para obter detalhes da API, consulte DescribeServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como recuperar detalhes de um serviço específico do cluster padrão.

```
Get-ECSService -Service my-hhtp-service
```

Exemplo 2: esse exemplo mostra como recuperar detalhes de um serviço específico em execução no cluster nomeado.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- Para obter detalhes da API, consulte DescribeServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeTasks`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeTasks`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

- Como criar um serviço de contêiner para instâncias de máquinas virtuais

CLI

AWS CLI

Exemplo 1: como descrever uma única tarefa

O exemplo de `describe-tasks` a seguir recupera os detalhes de uma tarefa em um cluster. É possível especificar a tarefa usando o ID ou o ARN completo da tarefa. Este exemplo usa o ARN completo da tarefa.

```
aws ecs describe-tasks \
    --cluster MyCluster \
    --tasks arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/4d590253bb114126b7afa7b58EXAMPLE
```

Saída:

```
{
    "tasks": [
        {
            "attachments": [],
            "attributes": [
                {
                    "name": "ecs.cpu-architecture",
                    "value": "x86_64"
                }
            ],
            "availabilityZone": "us-east-1b",
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
            "connectivity": "CONNECTED",
            "connectivityAt": "2021-08-11T12:21:26.681000-04:00",
            "containerInstanceArn": "arn:aws:ecs:us-
east-1:123456789012:container-instance/test/025c7e2c5e054a6790a29fc1fEXAMPLE",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:123456789012:container/MyCluster/4d590253bb114126b7afa7b58eea9221/
a992d1cc-ea46-474a-b6e8-24688EXAMPLE",
                    "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/4d590253bb114126b7afa7b58EXAMPLE",
```

```
                    "name": "simple-app",
                    "image": "httpd:2.4",
                    "runtimeId":
 "91251eed27db90006ad67b1a08187290869f216557717dd5c39b37c94EXAMPLE",
                    "lastStatus": "RUNNING",
                    "networkBindings": [
                        {
                            "bindIP": "0.0.0.0",
                            "containerPort": 80,
                            "hostPort": 80,
                            "protocol": "tcp"
                        }
                    ],
                    "networkInterfaces": [],
                    "healthStatus": "UNKNOWN",
                    "cpu": "10",
                    "memory": "300"
                }
            ],
            "cpu": "10",
            "createdAt": "2021-08-11T12:21:26.681000-04:00",
            "desiredStatus": "RUNNING",
            "enableExecuteCommand": false,
            "group": "service:testupdate",
            "healthStatus": "UNKNOWN",
            "lastStatus": "RUNNING",
            "launchType": "EC2",
            "memory": "300",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "simple-app"
                    }
                ],
                "inferenceAcceleratorOverrides": []
            },
            "pullStartedAt": "2021-08-11T12:21:28.234000-04:00",
            "pullStoppedAt": "2021-08-11T12:21:33.793000-04:00",
            "startedAt": "2021-08-11T12:21:34.945000-04:00",
            "startedBy": "ecs-svc/968695068243EXAMPLE",
            "tags": [],
            "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/4d590253bb114126b7afa7b58eea9221",
```

```
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-
definition/console-sample-app-static2:1",
            "version": 2
        }
    ],
    "failures": []
}
```

Para obter mais informações, consulte Definições de tarefa do Amazon ECS no Guia do desenvolvedor do Amazon ECS.

Exemplo 2: como descrever várias tarefas

O exemplo de `describe-tasks` a seguir recupera os detalhes de várias tarefas de um cluster. É possível especificar a tarefa usando o ID ou o ARN completo da tarefa. Este exemplo usa os IDs completos das tarefas.

```
aws ecs describe-tasks \
    --cluster MyCluster \
    --tasks "74de0355a10a4f979ac495c14EXAMPLE" "d789e94343414c25b9f6bd59eEXAMPLE"
```

Saída:

```
{
    "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d51EXAMPLE",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnet-0d0eab1bb3EXAMPLE"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
                        {
                            "name": "macAddress",
                            "value": "0e:89:76:28:07:b3"
```

```
                        },
                        {
                            "name": "privateDnsName",
                            "value": "ip-10-0-1-184.ec2.internal"
                        },
                        {
                            "name": "privateIPv4Address",
                            "value": "10.0.1.184"
                        }
                    ]
                }
            ],
            "attributes": [
                {
                    "name": "ecs.cpu-architecture",
                    "value": "x86_64"
                }
            ],
            "availabilityZone": "us-east-1b",
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
            "connectivity": "CONNECTED",
            "connectivityAt": "2021-12-20T12:13:37.875000-05:00",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:123456789012:container/MyCluster/74de0355a10a4f979ac495c14EXAMPLE/
aad3ba00-83b3-4dac-84d4-11f8cEXAMPLE",
                    "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/74de0355a10a4f979ac495c14EXAMPLE",
                    "name": "web",
                    "image": "nginx",
                    "runtimeId": "74de0355a10a4f979ac495c14EXAMPLE-265927825",
                    "lastStatus": "RUNNING",
                    "networkBindings": [],
                    "networkInterfaces": [
                        {
                            "attachmentId": "d9e7735a-16aa-4128-bc7a-
b2d51EXAMPLE",
                            "privateIpv4Address": "10.0.1.184"
                        }
                    ],
                    "healthStatus": "UNKNOWN",
                    "cpu": "99",
                    "memory": "100"
```

```
                }
            ],
            "cpu": "256",
            "createdAt": "2021-12-20T12:13:20.226000-05:00",
            "desiredStatus": "RUNNING",
            "enableExecuteCommand": false,
            "group": "service:tdsevicetag",
            "healthStatus": "UNKNOWN",
            "lastStatus": "RUNNING",
            "launchType": "FARGATE",
            "memory": "512",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "web"
                    }
                ],
                "inferenceAcceleratorOverrides": []
            },
            "platformVersion": "1.4.0",
            "platformFamily": "Linux",
            "pullStartedAt": "2021-12-20T12:13:42.665000-05:00",
            "pullStoppedAt": "2021-12-20T12:13:46.543000-05:00",
            "startedAt": "2021-12-20T12:13:48.086000-05:00",
            "startedBy": "ecs-svc/988401040018EXAMPLE",
            "tags": [],
            "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/74de0355a10a4f979ac495c14EXAMPLE",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-
definition/webserver:2",
            "version": 3,
            "ephemeralStorage": {
            "sizeInGiB": 20
            }
        },
        {
            "attachments": [
                {
                    "id": "214eb5a9-45cd-4bf8-87bc-57fefEXAMPLE",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
```

```
                            "value": "subnet-0d0eab1bb3EXAMPLE"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-064c7766daEXAMPLE"
                        },
                        {
                            "name": "macAddress",
                            "value": "0e:76:83:01:17:a9"
                        },
                        {
                            "name": "privateDnsName",
                            "value": "ip-10-0-1-41.ec2.internal"
                        },
                        {
                            "name": "privateIPv4Address",
                            "value": "10.0.1.41"
                        }
                    ]
                }
            ],
            "attributes": [
                {
                    "name": "ecs.cpu-architecture",
                    "value": "x86_64"
                }
            ],
            "availabilityZone": "us-east-1b",
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
            "connectivity": "CONNECTED",
            "connectivityAt": "2021-12-20T12:13:35.243000-05:00",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:123456789012:container/MyCluster/
d789e94343414c25b9f6bd59eEXAMPLE/9afef792-609b-43a5-bb6a-3efdbEXAMPLE",
                    "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/d789e94343414c25b9f6bd59eEXAMPLE",
                    "name": "web",
                    "image": "nginx",
                    "runtimeId": "d789e94343414c25b9f6bd59eEXAMPLE-265927825",
                    "lastStatus": "RUNNING",
                    "networkBindings": [],
                    "networkInterfaces": [
```

```
                        {
                            "attachmentId":
 "214eb5a9-45cd-4bf8-87bc-57fefEXAMPLE",
                            "privateIpv4Address": "10.0.1.41"
                        }
                    ],
                    "healthStatus": "UNKNOWN",
                    "cpu": "99",
                    "memory": "100"
                }
            ],
            "cpu": "256",
            "createdAt": "2021-12-20T12:13:20.226000-05:00",
            "desiredStatus": "RUNNING",
            "enableExecuteCommand": false,
            "group": "service:tdsevicetag",
            "healthStatus": "UNKNOWN",
            "lastStatus": "RUNNING",
            "launchType": "FARGATE",
            "memory": "512",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "web"
                    }
                ],
                "inferenceAcceleratorOverrides": []
            },
            "platformVersion": "1.4.0",
            "platformFamily": "Linux",
            "pullStartedAt": "2021-12-20T12:13:44.611000-05:00",
            "pullStoppedAt": "2021-12-20T12:13:48.251000-05:00",
            "startedAt": "2021-12-20T12:13:49.326000-05:00",
            "startedBy": "ecs-svc/988401040018EXAMPLE",
            "tags": [],
            "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/MyCluster/
d789e94343414c25b9f6bd59eEXAMPLE",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-
definition/webserver:2",
            "version": 3,
            "ephemeralStorage": {
                "sizeInGiB": 20
            }
        }
```

```
    ],
    "failures": []
}
```

Para obter mais informações, consulte Definições de tarefa do Amazon ECS no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte DescribeTasksem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> <taskId>\s
```

```
                Where:
                  clusterArn - The ARN of an ECS cluster.
                  taskId - The task Id value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        String taskId = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        getAllTasks(ecsClient, clusterArn, taskId);
        ecsClient.close();
    }

    public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
 taskId) {
        try {
            DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
                    .cluster(clusterArn)
                    .tasks(taskId)
                    .build();

            DescribeTasksResponse response =
 ecsClient.describeTasks(tasksRequest);
            List<Task> tasks = response.tasks();
            for (Task task : tasks) {
                System.out.println("The task ARN is " +
 task.taskDefinitionArn());
            }

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

- Para obter detalhes da API, consulte DescribeTasksa Referência AWS SDK for Java 2.x da API.

Use **`ListClusters`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListClusters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Obter informações de ARN para clusters, serviços e tarefas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List cluster ARNs available.
    /// </summary>
    /// <returns>The ARN list of clusters.</returns>
    public async Task<List<string>> GetClusterARNSAsync()
    {

        Console.WriteLine("Getting a list of all the clusters in your AWS
account...");
        List<string> clusterArnList = new List<string>();
        // Get a list of all the clusters in your AWS account
        try
        {

            var listClustersResponse = _ecsClient.Paginators.ListClusters(new
ListClustersRequest
            {
```

```
            });

            var clusterArns = listClustersResponse.ClusterArns;

            // Print the ARNs of the clusters
            await foreach (var clusterArn in clusterArns)
            {
                clusterArnList.Add(clusterArn);
            }

            if (clusterArnList.Count == 0)
            {
                _logger.LogWarning("No clusters found in your AWS account.");
            }
            return clusterArnList;
        }
        catch (Exception e)
        {
            _logger.LogError($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
            throw new Exception($"An error occurred while getting a list of all
the clusters in your AWS account. {e.InnerException}");
        }
    }
```

- Para obter detalhes da API, consulte ListClustersa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os clusters disponíveis

O exemplo de `list-clusters` a seguir lista todos os clusters disponíveis.

```
aws ecs list-clusters
```

Saída:

```
{
    "clusterArns": [
        "arn:aws:ecs:us-west-2:123456789012:cluster/MyECSCluster1",
```

```
        "arn:aws:ecs:us-west-2:123456789012:cluster/AnotherECSCluster"
    ]
}
```

Para obter mais informações, consulte Clusters do Amazon ECS no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte ListClustersem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

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

public class ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();
```

```
        listAllClusters(ecsClient);
        ecsClient.close();
    }

    public static void listAllClusters(EcsClient ecsClient) {
        try {
            ListClustersResponse response = ecsClient.listClusters();
            List<String> clusters = response.clusterArns();
            for (String cluster : clusters) {
                System.out.println("The cluster arn is " + cluster);
            }

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

- Para obter detalhes da API, consulte ListClustersa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet retorna uma lista de clusters do ECS existentes.

```
Get-ECSClusterList
```

Saída:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Para obter detalhes da API, consulte ListClustersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet retorna uma lista de clusters do ECS existentes.

```
Get-ECSClusterList
```

Saída:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Para obter detalhes da API, consulte ListClustersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListServices`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListServices`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Obter informações de ARN para clusters, serviços e tarefas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List service ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of services in given cluster.</returns>
    public async Task<List<string>> GetServiceARNSAsync(string clusterARN)
    {
        List<string> serviceArns = new List<string>();
```

```
        var request = new ListServicesRequest
        {
            Cluster = clusterARN
        };
        // Call the ListServices API operation and get the list of service ARNs
        var serviceList = _ecsClient.Paginators.ListServices(request);

        await foreach (var serviceARN in serviceList.ServiceArns)
        {
            if (serviceARN is null)
                continue;

            serviceArns.Add(serviceARN);
        }

        if (serviceArns.Count == 0)
        {
            _logger.LogWarning($"No services found in cluster {clusterARN} .");
        }

        return serviceArns;
    }
```

- Para obter detalhes da API, consulte ListServicesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os serviços em um cluster

O exemplo de `list-services` a seguir mostra como listar os serviços em execução em um cluster.

```
aws ecs list-services --cluster MyCluster
```

Saída:

```
{
    "serviceArns": [
        "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/MyService"
```

```
    ]
}
```

Para obter mais informações, consulte Serviços no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte ListServicesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todos os serviços em execução no cluster padrão.

```
Get-ECSClusterService
```

Exemplo 2: esse exemplo lista todos os serviços em execução no cluster especificado.

```
Get-ECSClusterService -Cluster myCluster
```

- Para obter detalhes da API, consulte ListServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os serviços em execução no cluster padrão.

```
Get-ECSClusterService
```

Exemplo 2: esse exemplo lista todos os serviços em execução no cluster especificado.

```
Get-ECSClusterService -Cluster myCluster
```

- Para obter detalhes da API, consulte ListServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListTasks`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTasks`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Obter informações de ARN para clusters, serviços e tarefas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// List task ARNs available.
/// </summary>
/// <param name="clusterARN">The arn of the ECS cluster.</param>
/// <returns>The ARN list of tasks in given cluster.</returns>
public async Task<List<string>> GetTaskARNsAsync(string clusterARN)
{
    // Set up the request to describe the tasks in the service
    var listTasksRequest = new ListTasksRequest
    {
        Cluster = clusterARN
    };
    List<string> taskArns = new List<string>();

    // Call the ListTasks API operation and get the list of task ARNs
    var tasks = _ecsClient.Paginators.ListTasks(listTasksRequest);

    await foreach (var task in tasks.TaskArns)
    {
        if (task is null)
            continue;


        taskArns.Add(task);
    }
```

```
        if (taskArns.Count == 0)
        {
            _logger.LogWarning("No tasks found in cluster: " + clusterARN);
        }

        return taskArns;
    }
```

- Para obter detalhes da API, consulte ListTasksa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: listar as tarefas em um cluster

O exemplo de `list-tasks` a seguir lista todas as tarefas de um cluster.

```
aws ecs list-tasks --cluster default
```

Saída:

```
{
    "taskArns": [
        "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-11111EXAMPLE",
        "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE"
    ]
}
```

Exemplo 2: como listar as tarefas de uma instância de contêiner específica

O exemplo de `list-tasks` a seguir lista as tarefas de uma instância de contêiner, usando o UUID da instância de contêiner como filtro.

```
aws ecs list-tasks --cluster default --container-instance a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE
```

Saída:

```
{
    "taskArns": [
        "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-44444EXAMPLE"
    ]
}
```

Para obter mais informações, consulte Definições de tarefa do Amazon ECS no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte ListTasksem Referência de AWS CLI Comandos.

Usar **`UpdateClusterSettings`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateClusterSettings`.

CLI

AWS CLI

Para modificar as configurações do seu cluster

O `update-cluster-settings` exemplo a seguir habilita o CloudWatch Container Insights com observabilidade aprimorada para o `MyCluster` cluster.

```
aws ecs update-cluster-settings \
    --cluster MyCluster \
    --settings name=containerInsights,value=enhanced
```

Saída:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-esat-1:123456789012:cluster/MyCluster",
        "clusterName": "default",
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
```

```
        "tags": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "enhanced"
            }
        ]
    }
}
```

Para obter mais informações, consulte Modificar configurações da conta no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte UpdateClusterSettingsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet modifica as configurações a serem usadas em um cluster do ECS.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting
 @{Name="containerInsights"; Value="disabled"}
```

Saída:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/
LAB-ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Para obter detalhes da API, consulte UpdateClusterSettingsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet modifica as configurações a serem usadas em um cluster do ECS.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting
 @{Name="containerInsights"; Value="disabled"}
```

Saída:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/
LAB-ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Para obter detalhes da API, consulte UpdateClusterSettingsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`UpdateService`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateService`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais

CLI

AWS CLI

Exemplo 1: como alterar a definição de tarefa usada em um serviço

O exemplo de `update-service` a seguir atualiza o serviço `my-http-service` para usar a definição de tarefa `amazon-ecs-sample`.

```
aws ecs update-service \
    --cluster test \
    --service my-http-service \
    --task-definition amazon-ecs-sample
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/test/my-http-
service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/test",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 2,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "1.4.0",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
amazon-ecs-sample:2",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
```

```
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/7419115625193919142",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/amazon-ecs-sample:2",
                "desiredCount": 0,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": "2025-02-21T13:26:02.734000-06:00",
                "updatedAt": "2025-02-21T13:26:02.734000-06:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/7419115625193919142
 in progress."
            },
            {
                "id": "ecs-svc/1709597507655421668",
                "status": "ACTIVE",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/old-amazon-ecs-sample:4",
                "desiredCount": 2,
                "pendingCount": 0,
                "runningCount": 2,
                "failedTasks": 0,
                "createdAt": "2025-01-24T11:13:07.621000-06:00",
                "updatedAt": "2025-02-02T16:11:30.838000-06:00",
```

```
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                             "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/1709597507655421668
 completed."
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [
            {
                "id": "e40b4d1c-80d9-4834-aaf3-6a268e530e17",
                "createdAt": "2025-02-21T10:31:26.037000-06:00",
                "message": "(my-http-service) has reached a steady state."
            },
            {
                "id": "6ac069ad-fc8b-4e49-a35d-b5574a964c8e",
                "createdAt": "2025-02-21T04:31:22.703000-06:00",
                "message": "(my-http-service) has reached a steady state."
            },
            {
                "id": "265f7d37-dfd1-4880-a846-ec486f341919",
                "createdAt": "2025-02-20T22:31:22.514000-06:00",
                "message": "(my-http-service) has reached a steady state."
            }
        ],
        "createdAt": "2024-10-30T17:12:43.218000-05:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
```

```
                    "subnet-12344321",
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/AIDACKCEVSQ6C2EXAMPLE",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "DISABLED"
    }
}
```

Consulte mais informações em Atualização de um serviço do Amazon ECS usando o console no Guia do desenvolvedor do Amazon ECS.

Exemplo 2: alterar o número de tarefas em um serviço

O exemplo de `update-service` a seguir atualiza a contagem de tarefas desejada do serviço `my-http-service` para 2.

```
aws ecs update-service \
    --cluster MyCluster \
    --service my-http-service \
    --desired-count 2
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:123456789012:service/MyCluster/my-
http-service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
        "loadBalancers": [],
```

```
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 1,
        "pendingCount": 0,
        "capacityProviderStrategy": [
            {
                "capacityProvider": "FARGATE",
                "weight": 1,
                "base": 0
            }
        ],
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-east-1:123456789012:task-definition/
MyTaskDefinition",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/1976744184940610707",
                "status": "PRIMARY",
                "taskkDefinition": "arn:aws:ecs:us-east-1:123456789012:task-
definition/MyTaskDefinition",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 1,
                "failedTasks": 0,
                "createdAt": "2024-12-03T16:24:25.225000-05:00",
                "updatedAt": "2024-12-03T16:25:15.837000-05:00",
                "capacityProviderStrategy": [
                    {
                        "capacityProvider": "FARGATE",
```

```
                        "weight": 1,
                        "base": 0
                    }
                ],
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-0d0eab1bb38d5ca64",
                            "subnet-0db5010045995c2d5"
                        ],
                        "securityGroups": [
                            "sg-02556bf85a191f59a"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/1976744184940610707
 completed."
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [
            {
                "id": "f27350b9-4b2a-4e2e-b72e-a4b68380de45",
                "createdAt": "2024-12-30T13:24:07.345000-05:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "e764ec63-f53f-45e3-9af2-d99f922d2957",
                "createdAt": "2024-12-30T12:32:21.600000-05:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "28444756-c2fa-47f8-bd60-93a8e05f3991",
                "createdAt": "2024-12-08T19:26:10.367000-05:00",
                "message": "(service my-http-service) has reached a steady
 state."
            }
```

```
        ],
        "createdAt": "2024-12-03T16:24:25.225000-05:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-0d0eab1bb38d5ca64",
                    "subnet-0db5010045995c2d5"
                ],
                "securityGroups": [
                    "sg-02556bf85a191f59a"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/Admin",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "ENABLED"
    }
}
```

Consulte mais informações em Atualização de um serviço do Amazon ECS usando o console no Guia do desenvolvedor do Amazon ECS.

Exemplo 3: como configurar volumes do Amazon EBS para anexação na atualização do serviço

O exemplo de `update-service` a seguir atualiza o serviço `my-http-service` para usar volumes do Amazon EBS. Você deve ter uma função de infraestrutura do Amazon ECS configurada com a política `AmazonECSInfrastructureRolePolicyForVolumes` gerenciada anexada. Você também deve especificar uma definição de tarefa com o mesmo nome de volume da solicitação `update-service` e com `configuredAtLaunch` definida como `true`. Este exemplo usa a opção `--cli-input-json` e um arquivo de entrada JSON chamado `ebs.json`.

```
aws ecs update-service \
    --cli-input-json file://ebs.json
```

Conteúdo de `ebs.json`:

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "service": "my-http-service",
   "desiredCount": 2,
   "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
                "roleArn":"arn:aws:iam::123456789012:role/ecsInfrastructureRole",
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "iops": 3000,
                "throughput": 125,
                "filesystemType": "ext4"
            }
        }
   ]
}
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/mycluster/my-
http-service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/mycluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 2,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
```

```
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
mytaskdef:1",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/2420458347226626275",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/mytaskdef:1",
                "desiredCount": 0,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": "2025-02-21T15:07:20.519000-06:00",
                "updatedAt": "2025-02-21T15:07:20.519000-06:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321",
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/2420458347226626275
 in progress.",
```

```
                "volumeConfigurations": [
                    {
                        "name": "ebs-volume",
                        "managedEBSVolume": {
                            "volumeType": "gp3",
                            "sizeInGiB": 100,
                            "iops": 3000,
                            "throughput": 125,
                            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRole",
                            "filesystemType": "ext4"
                        }
                    }
                ]
            },
            {
                "id": "ecs-svc/5191625155316533644",
                "status": "ACTIVE",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/mytaskdef:2",
                "desiredCount": 2,
                "pendingCount": 0,
                "runningCount": 2,
                "failedTasks": 0,
                "createdAt": "2025-02-21T14:54:48.862000-06:00",
                "updatedAt": "2025-02-21T14:57:22.502000-06:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/5191625155316533644
 completed."
            }
```

```
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [
            {
                "id": "b5823113-c2c5-458e-9649-8c2ed38f23a5",
                "createdAt": "2025-02-21T14:57:22.508000-06:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "b05a48e8-da35-4074-80aa-37ceb3167357",
                "createdAt": "2025-02-21T14:57:22.507000-06:00",
                "message": "(service my-http-service) (deployment ecs-
svc/5191625155316533644) deployment completed."
            },
            {
                "id": "a10cd55d-4ba6-4cea-a655-5a5d32ada8a0",
                "createdAt": "2025-02-21T14:55:32.833000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 fb9c8df512684aec92f3c57dc3f22361)."
            },
        ],
        "createdAt": "2025-02-21T14:54:48.862000-06:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321"
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/AIDACKCEVSQ6C2EXAMPLE",
        "enableECSManagedTags": true,
```

```
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "ENABLED"
    }
}
```

Consulte mais informações em Uso de volumes do Amazon EBS com o Amazon ECS no Guia do desenvolvedor do Amazon ECS.

Exemplo 4: como atualizar um serviço para não usar mais volumes do Amazon EBS

O exemplo de `update-service` a seguir atualiza o serviço `my-http-service` para deixar de usar volumes do Amazon EBS. Você deve especificar uma revisão de definição de tarefa com `configuredAtLaunch` definida como `false`.

```
aws ecs update-service \
    --cluster mycluster \
    --task-definition mytaskdef \
    --service my-http-service \
    --desired-count 2 \
    --volume-configurations "[]"
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/mycluster/my-
http-service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/mycluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 2,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
mytaskdef:3",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
```

```
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/7522791612543716777",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/mytaskdef:3",
                "desiredCount": 0,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": "2025-02-21T15:25:38.598000-06:00",
                "updatedAt": "2025-02-21T15:25:38.598000-06:00",
                    "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/7522791612543716777
 in progress."
            },
            {
                "id": "ecs-svc/2420458347226626275",
                "status": "ACTIVE",
```

```
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/myoldtaskdef:1",
                "desiredCount": 2,
                "pendingCount": 0,
                "runningCount": 2,
                "failedTasks": 0,
                "createdAt": "2025-02-21T15:07:20.519000-06:00",
                "updatedAt": "2025-02-21T15:10:59.955000-06:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/2420458347226626275
 completed.",
                "volumeConfigurations": [
                    {
                        "name": "ebs-volume",
                        "managedEBSVolume": {
                            "volumeType": "gp3",
                            "sizeInGiB": 100,
                            "iops": 3000,
                            "throughput": 125,
                            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRole",
                            "filesystemType": "ext4"
                        }
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [
```

```
            {
                "id": "4f2c3ca1-7800-4048-ba57-bba210ada2ad",
                "createdAt": "2025-02-21T15:10:59.959000-06:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "4b36a593-2d40-4ed6-8be8-b9b699eb6198",
                "createdAt": "2025-02-21T15:10:59.958000-06:00",
                "message": "(service my-http-service) (deployment ecs-
svc/2420458347226626275) deployment completed."
            },
            {
                "id": "88380089-14e2-4ef0-8dbb-a33991683371",
                "createdAt": "2025-02-21T15:09:39.055000-06:00",
                "message": "(service my-http-service) has stopped 1 running
 tasks: (task fb9c8df512684aec92f3c57dc3f22361)."
            },
            {
                "id": "97d84243-d52f-4255-89bb-9311391c61f6",
                "createdAt": "2025-02-21T15:08:57.653000-06:00",
                "message": "(service my-http-service) has stopped 1 running
 tasks: (task 33eff090ad2c40539daa837e6503a9bc)."
            },
            {
                "id": "672ece6c-e2d0-4021-b5da-eefb14001687",
                "createdAt": "2025-02-21T15:08:15.631000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 996c02a66ff24f3190a4a8e0c841740f)."
            },
            {
                "id": "a3cf9bea-9be6-4175-ac28-4c68360986eb",
                "createdAt": "2025-02-21T15:07:36.931000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 d5d23c39f89e46cf9a647b9cc6572feb)."
            },
            {
                "id": "b5823113-c2c5-458e-9649-8c2ed38f23a5",
                "createdAt": "2025-02-21T14:57:22.508000-06:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "b05a48e8-da35-4074-80aa-37ceb3167357",
```

```
                "createdAt": "2025-02-21T14:57:22.507000-06:00",
                "message": "(service my-http-service) (deployment ecs-
svc/5191625155316533644) deployment completed."
            },
            {
                "id": "a10cd55d-4ba6-4cea-a655-5a5d32ada8a0",
                "createdAt": "2025-02-21T14:55:32.833000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 fb9c8df512684aec92f3c57dc3f22361)."
            },
            {
                "id": "42da91fa-e26d-42ef-88c3-bb5965c56b2f",
                "createdAt": "2025-02-21T14:55:02.703000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 33eff090ad2c40539daa837e6503a9bc)."
            }
        ],
        "createdAt": "2025-02-21T14:54:48.862000-06:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321"
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/AIDACKCEVSQ6C2EXAMPLE",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "ENABLED"
    }
}
```

Consulte mais informações em Uso de volumes do Amazon EBS com o Amazon ECS no Guia do desenvolvedor do Amazon ECS.

Exemplo 5: como ativar o rebalanceamento da zona de disponibilidade para um serviço

O exemplo de `update-service` a seguir ativa o rebalanceamento da zona de disponibilidade para o serviço `my-http-service`.

```
aws ecs update-service \
    --cluster MyCluster \
    --service my-http-service \
    --availability-zone-rebalancing ENABLED
```

Saída:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:123456789012:service/MyCluster/my-
http-service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 1,
        "pendingCount": 0,
        "capacityProviderStrategy": [
            {
                "capacityProvider": "FARGATE",
                "weight": 1,
                "base": 0
            }
        ],
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-east-1:123456789012:task-definition/
MyTaskDefinition",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
```

```
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/1976744184940610707",
                "status": "PRIMARY",
                "taskkDefinition": "arn:aws:ecs:us-east-1:123456789012:task-
definition/MyTaskDefinition",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 1,
                "failedTasks": 0,
                "createdAt": "2024-12-03T16:24:25.225000-05:00",
                "updatedAt": "2024-12-03T16:25:15.837000-05:00",
                "capacityProviderStrategy": [
                    {
                        "capacityProvider": "FARGATE",
                        "weight": 1,
                        "base": 0
                    }
                ],
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-0d0eab1bb38d5ca64",
                            "subnet-0db5010045995c2d5"
                        ],
                        "securityGroups": [
                            "sg-02556bf85a191f59a"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/1976744184940610707
 completed."
```

```
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
        "createdAt": "2024-12-03T16:24:25.225000-05:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-0d0eab1bb38d5ca64",
                    "subnet-0db5010045995c2d5"
                ],
                "securityGroups": [
                    "sg-02556bf85a191f59a"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/Admin",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "ENABLED"
    }
}
```

Consulte mais informações em Atualização de um serviço do Amazon ECS usando o console no Guia do desenvolvedor do Amazon ECS.

- Para obter detalhes da API, consulte UpdateServiceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

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

public class UpdateService {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                   <clusterName> <serviceArn>\s

                Where:
                  clusterName - The cluster name.
                  serviceArn - The service ARN value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        updateSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void updateSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .desiredCount(0)
                    .build();

            ecsClient.updateService(serviceRequest);
            System.out.println("The service was modified");

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

- Para obter detalhes da API, consulte UpdateServicea Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo de comando atualiza o serviço “my-http-service” para usar a definição de tarefa “amazon-ecs-sample”.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Exemplo 2: esse exemplo de comando atualiza a contagem desejada do serviço “my-http-service” para 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Para obter detalhes da API, consulte UpdateServiceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo de comando atualiza o serviço “my-http-service” para usar a definição de tarefa “amazon-ecs-sample”.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Exemplo 2: esse exemplo de comando atualiza a contagem desejada do serviço “my-http-service” para 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Para obter detalhes da API, consulte UpdateServiceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para o Amazon ECS usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon ECS com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon ECS ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Obtenha informações de ARN para clusters, serviços e tarefas do Amazon ECS usando um AWS SDK

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
```

```
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"
```

```
    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
```

```
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
```

```
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
```

```
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
```

```
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
```

```
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
```

```
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF
```

```
    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
```

```
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}
```

```
# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
```

```
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
```

```
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources
```

```
    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace

- DeleteRole
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeClusters
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um cluster.
- Criar uma definição de tarefa.
- Criar o serviço.
- Fazer a limpeza.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"
```

```
    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
```

```
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
```

```
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
```

```
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
 then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME'" "Delete ECS service"
            else
```

```
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
" ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
```

```
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}
```

```
# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
```

```
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
```

```
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
```

```
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
```

```
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"
```

```
# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
```

```
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices
  - DescribeSubnets

- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
```

```
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
```

```
    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
```

```
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
$TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
```

```
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
file for details."
    fi
```

```
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
```

```
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
```

```
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}
```

```
# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role
```

```
        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
```

```
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
```

```
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
```

```
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."
```

```
    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
```

```
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10
```

```
        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "==========================================="
    echo "Resources created:"
```

```
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup

- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Obtenha informações de ARN para clusters, serviços e tarefas do Amazon ECS usando um AWS SDK

O exemplo de código a seguir mostra como:

- Obter uma lista de todos os clusters.
- Obter serviços para um cluster.
- Obter tarefas para um cluster.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
using Amazon.ECS;
using ECSActions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace ECSScenario;

public class ECSScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.


    This .NET example performs the following tasks:
        1. List ECS Cluster ARNs.
        2. List services in every cluster
        3. List Task ARNs in every cluster.
    */

    private static ILogger logger = null!;
    private static ECSWrapper _ecsWrapper = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
       .Build();

       ILoggerFactory loggerFactory = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       });

       logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
           .CreateLogger<ECSScenario>();

       var loggerECSWarpper = LoggerFactory.Create(builder =>
{ builder.AddConsole(); })
         .CreateLogger<ECSWrapper>();

       var amazonECSClient = new AmazonECSClient();

       _ecsWrapper = new ECSWrapper(amazonECSClient, loggerECSWarpper);

       Console.WriteLine(new string('-', 80));
       Console.WriteLine("Welcome to the Amazon ECS example scenario.");
       Console.WriteLine(new string('-', 80));

       try
       {
           await ListClusterARNs();
           await ListServiceARNs();
           await ListTaskARNs();

       }
       catch (Exception ex)
       {
           logger.LogError(ex, "There was a problem executing the scenario.");
       }
   }

   /// <summary>
   /// List ECS Cluster ARNs
   /// </summary>
   private static async Task ListClusterARNs()
   {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. List Cluster ARNs from ECS.");
        var arns = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var arn in arns)
        {
            Console.WriteLine($"Cluster arn: {arn}");
            Console.WriteLine($"Cluster name: {arn.Split("/").Last()}");
        }

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// List services in every cluster
    /// </summary>
    private static async Task ListServiceARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List Service ARNs in every cluster.");
        var clusterARNs = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var clusterARN in clusterARNs)
        {
            Console.WriteLine($"Getting services for cluster name:
{clusterARN.Split("/").Last()}");
            Console.WriteLine(new string('.', 5));


            var serviceARNs = await _ecsWrapper.GetServiceARNSAsync(clusterARN);

            foreach (var serviceARN in serviceARNs)
            {
                Console.WriteLine($"Service arn: {serviceARN}");
                Console.WriteLine($"Service name:
{serviceARN.Split("/").Last()}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// List tasks in every cluster
    /// </summary>
    private static async Task ListTaskARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. List Task ARNs in every cluster.");
        var clusterARNs = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var clusterARN in clusterARNs)
        {
            Console.WriteLine($"Getting tasks for cluster name:
 {clusterARN.Split("/").Last()}");
            Console.WriteLine(new string('.', 5));

            var taskARNs = await _ecsWrapper.GetTaskARNsAsync(clusterARN);

            foreach (var taskARN in taskARNs)
            {
                Console.WriteLine($"Task arn: {taskARN}");
            }
        }
        Console.WriteLine(new string('-', 80));
    }
}
```

Métodos de encapsulamento que são chamados pelo cenário para gerenciar as ações do Amazon ECS.

```
using Amazon.ECS;
using Amazon.ECS.Model;
using Microsoft.Extensions.Logging;

namespace ECSActions;

public class ECSWrapper
{
    private readonly AmazonECSClient _ecsClient;
    private readonly ILogger<ECSWrapper> _logger;

    /// <summary>
    /// Constructor for the ECS wrapper.
```

```
    /// </summary>
    /// <param name="ecsClient">The injected ECS client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public ECSWrapper(AmazonECSClient ecsClient, ILogger<ECSWrapper> logger)

    {
        _logger = logger;
        _ecsClient = ecsClient;
    }

    /// <summary>
    /// List cluster ARNs available.
    /// </summary>
    /// <returns>The ARN list of clusters.</returns>
    public async Task<List<string>> GetClusterARNSAsync()
    {

        Console.WriteLine("Getting a list of all the clusters in your AWS
account...");
        List<string> clusterArnList = new List<string>();
        // Get a list of all the clusters in your AWS account
        try
        {

            var listClustersResponse = _ecsClient.Paginators.ListClusters(new
ListClustersRequest
            {
            });

            var clusterArns = listClustersResponse.ClusterArns;

            // Print the ARNs of the clusters
            await foreach (var clusterArn in clusterArns)
            {
                clusterArnList.Add(clusterArn);
            }

            if (clusterArnList.Count == 0)
            {
                _logger.LogWarning("No clusters found in your AWS account.");
            }
            return clusterArnList;
        }
        catch (Exception e)
```

```
        {
            _logger.LogError($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
            throw new Exception($"An error occurred while getting a list of all
the clusters in your AWS account. {e.InnerException}");
        }
    }

    /// <summary>
    /// List service ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of services in given cluster.</returns>
    public async Task<List<string>> GetServiceARNSAsync(string clusterARN)
    {
        List<string> serviceArns = new List<string>();

        var request = new ListServicesRequest
        {
            Cluster = clusterARN
        };
        // Call the ListServices API operation and get the list of service ARNs
        var serviceList = _ecsClient.Paginators.ListServices(request);

        await foreach (var serviceARN in serviceList.ServiceArns)
        {
            if (serviceARN is null)
                continue;

            serviceArns.Add(serviceARN);
        }

        if (serviceArns.Count == 0)
        {
            _logger.LogWarning($"No services found in cluster {clusterARN} .");
        }

        return serviceArns;
    }

    /// <summary>
    /// List task ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
```

```
    /// <returns>The ARN list of tasks in given cluster.</returns>
    public async Task<List<string>> GetTaskARNsAsync(string clusterARN)
    {
        // Set up the request to describe the tasks in the service
        var listTasksRequest = new ListTasksRequest
        {
            Cluster = clusterARN
        };
        List<string> taskArns = new List<string>();

        // Call the ListTasks API operation and get the list of task ARNs
        var tasks = _ecsClient.Paginators.ListTasks(listTasksRequest);

        await foreach (var task in tasks.TaskArns)
        {
            if (task is null)
                continue;


            taskArns.Add(task);
        }

        if (taskArns.Count == 0)
        {
            _logger.LogWarning("No tasks found in cluster: " + clusterARN);
        }

        return taskArns;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - ListClusters
  - ListServices
  - ListTasks

Exemplos de código para o Amazon EFS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Elastic File System com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Amazon EFS: mais informações sobre o Amazon EFS.
- Referência de API do Amazon EFS: detalhes sobre todas as ações disponíveis do Amazon EFS.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon EFS usando AWS SDKs
  - Ações para o Amazon EFS usando AWS SDKs
    - Usar CreateFileSystem com uma CLI
    - Usar CreateMountTarget com uma CLI
    - Usar DeleteFileSystem com uma CLI
    - Usar DeleteMountTarget com uma CLI
    - Usar DescribeFileSystems com uma CLI
    - Usar DescribeMountTargets com uma CLI
    - Usar DescribeTags com uma CLI

Exemplos básicos para o Amazon EFS usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Elastic File System com AWS SDKs.

Exemplos

- Ações para o Amazon EFS usando AWS SDKs
  - Usar CreateFileSystem com uma CLI
  - Usar CreateMountTarget com uma CLI
  - Usar DeleteFileSystem com uma CLI
  - Usar DeleteMountTarget com uma CLI
  - Usar DescribeFileSystems com uma CLI
  - Usar DescribeMountTargets com uma CLI
  - Usar DescribeTags com uma CLI

Ações para o Amazon EFS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon EFS com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Elastic File System.

Exemplos

- Usar CreateFileSystem com uma CLI
- Usar CreateMountTarget com uma CLI
- Usar DeleteFileSystem com uma CLI
- Usar DeleteMountTarget com uma CLI
- Usar DescribeFileSystems com uma CLI
- Usar DescribeMountTargets com uma CLI
- Usar DescribeTags com uma CLI

Usar **`CreateFileSystem`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateFileSystem`.

CLI

AWS CLI

Para criar um sistema de arquivos criptografado

O exemplo `create-file-system` a seguir cria um sistema de arquivos criptografado usando a CMK padrão. Ele também adiciona a tag `Name=my-file-system`.

```
aws efs create-file-system \
    --performance-mode generalPurpose \
    --throughput-mode bursting \
    --encrypted \
    --tags Key=Name,Value=my-file-system
```

Saída:

```
{
    "OwnerId": "123456789012",
    "CreationToken": "console-d7f56c5f-e433-41ca-8307-9d9c0example",
    "FileSystemId": "fs-c7a0456e",
    "FileSystemArn": "arn:aws:elasticfilesystem:us-west-2:123456789012:file-
system/fs-48499b4d",
    "CreationTime": 1595286880.0,
    "LifeCycleState": "creating",
    "Name": "my-file-system",
    "NumberOfMountTargets": 0,
    "SizeInBytes": {
        "Value": 0,
        "ValueInIA": 0,
        "ValueInStandard": 0
    },
    "PerformanceMode": "generalPurpose",
    "Encrypted": true,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/a59b3472-e62c-42e4-
adcf-30d92example",
    "ThroughputMode": "bursting",
    "Tags": [
        {
            "Key": "Name",
            "Value": "my-file-system"
        }
    ]
}
```

Para ter mais informações, consulte Criar sistemas de arquivos do Amazon EFS no Guia do usuário do Amazon Elastic File System.

- Para obter detalhes da API, consulte CreateFileSystemem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria um sistema de arquivos vazio. O token usado para garantir a criação idempotente será gerado automaticamente e poderá ser acessado pelo membro **`CreationToken`** do objeto retornado.

```
New-EFSFileSystem
```

Saída:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : creating
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize
```

Exemplo 2: cria um sistema de arquivos vazio usando um token personalizado para garantir uma criação idempotente.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Para obter detalhes da API, consulte CreateFileSystemem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria um sistema de arquivos vazio. O token usado para garantir a criação idempotente será gerado automaticamente e poderá ser acessado pelo membro **`CreationToken`** do objeto retornado.

```
New-EFSFileSystem
```

Saída:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : creating
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize
```

Exemplo 2: cria um sistema de arquivos vazio usando um token personalizado para garantir uma criação idempotente.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Para obter detalhes da API, consulte CreateFileSystemem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateMountTarget`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateMountTarget`.

CLI

AWS CLI

Para criar um destino de montagem

O exemplo `create-mount-target` a seguir cria um destino de montagem para o sistema de arquivos especificado.

```
aws efs create-mount-target \
    --file-system-id fs-c7a0456e \
    --subnet-id subnet-02bf4c428bexample \
    --security-groups sg-068f739363example
```

Saída:

```
{
    "OwnerId": "123456789012",
    "MountTargetId": "fsmt-f9a14450",
    "FileSystemId": "fs-c7a0456e",
```

```
    "SubnetId": "subnet-02bf4c428bexample",
    "LifeCycleState": "creating",
    "IpAddress": "10.0.1.24",
    "NetworkInterfaceId": "eni-02d542216aexample",
    "AvailabilityZoneId": "use2-az2",
    "AvailabilityZoneName": "us-east-2b",
    "VpcId": "vpc-0123456789abcdef0"
}
```

Para obter mais informações, consulte Criar destinos de montagem no Guia do usuário do Amazon Elastic File System.

- Para obter detalhes da API, consulte CreateMountTargetem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria um destino de montagem para um sistema de arquivos. A sub-rede especificada será usada para determinar a nuvem privada virtual (VPC) na qual o destino de montagem será criado e o endereço IP que será atribuído automaticamente (do intervalo de endereços da sub-rede). O endereço IP atribuído pode ser usado para, em seguida, montar esse sistema de arquivos em uma instância do Amazon EC2. Como nenhum grupo de segurança foi especificado, a interface de rede criada para o destino está associada ao grupo de segurança padrão para a VPC da sub-rede.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Saída:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Exemplo 2: cria um destino de montagem para o sistema de arquivos especificado com endereço IP atribuído automaticamente. A interface de rede criada para o destino de

montagem está associada aos grupos de segurança especificados (até cinco, no formato “sg-xxxxxxxx”, podem ser especificados).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Exemplo 3: cria um destino de montagem para o sistema de arquivos especificado com o endereço IP especificado.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- Para obter detalhes da API, consulte CreateMountTargetem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria um destino de montagem para um sistema de arquivos. A sub-rede especificada será usada para determinar a nuvem privada virtual (VPC) na qual o destino de montagem será criado e o endereço IP que será atribuído automaticamente (do intervalo de endereços da sub-rede). O endereço IP atribuído pode ser usado para, em seguida, montar esse sistema de arquivos em uma instância do Amazon EC2. Como nenhum grupo de segurança foi especificado, a interface de rede criada para o destino está associada ao grupo de segurança padrão para a VPC da sub-rede.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Saída:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Exemplo 2: cria um destino de montagem para o sistema de arquivos especificado com endereço IP atribuído automaticamente. A interface de rede criada para o destino de

montagem está associada aos grupos de segurança especificados (até cinco, no formato “sg-xxxxxxxx”, podem ser especificados).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Exemplo 3: cria um destino de montagem para o sistema de arquivos especificado com o endereço IP especificado.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- Para obter detalhes da API, consulte CreateMountTargetem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteFileSystem`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteFileSystem`.

CLI

AWS CLI

Para excluir um sistema de arquivos

O exemplo `delete-file-system` a seguir exclui o sistema de arquivos especificado.

```
aws efs delete-file-system \
    --file-system-id fs-c7a0456e
```

Este comando não produz saída.

Para ter mais informações, consulte Excluir um sistema de arquivos do Amazon EFS no Guia do usuário do Amazon Elastic File System.

- Para obter detalhes da API, consulte DeleteFileSystemem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui o sistema de arquivos especificado que não está mais em uso (se o sistema de arquivos tiver destinos de montagem, eles deverão ser removidos primeiro). É solicitada uma confirmação antes que o cmdlet continue. Para ignorar a confirmação, use a opção **-Force**.

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Para obter detalhes da API, consulte DeleteFileSystemem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui o sistema de arquivos especificado que não está mais em uso (se o sistema de arquivos tiver destinos de montagem, eles deverão ser removidos primeiro). É solicitada uma confirmação antes que o cmdlet continue. Para ignorar a confirmação, use a opção **-Force**.

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Para obter detalhes da API, consulte DeleteFileSystemem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **DeleteMountTarget** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteMountTarget`.

CLI

AWS CLI

Para excluir um destino de montagem

O exemplo `delete-mount-target` a seguir exclui o destino de montagem especificado.

```
aws efs delete-mount-target \
    --mount-target-id fsmt-f9a14450
```

Este comando não produz saída.

Para obter mais informações, consulte Criar destinos de montagem no Guia do usuário do Amazon Elastic File System.

- Para obter detalhes da API, consulte DeleteMountTargetem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui o destino de montagem especificado. A confirmação será solicitada antes que a operação continue. Para suprimir o prompt, use a opção **`-Force`**. Observe que essa operação interrompe à força qualquer montagem do sistema de arquivos por meio do destino. Convém considerar a desmontagem do sistema de arquivos antes de executar esse comando, se possível.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Para obter detalhes da API, consulte DeleteMountTargetem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui o destino de montagem especificado. A confirmação será solicitada antes que a operação continue. Para suprimir o prompt, use a opção **`-Force`**. Observe que essa operação interrompe à força qualquer montagem do sistema de arquivos por meio do destino. Convém considerar a desmontagem do sistema de arquivos antes de executar esse comando, se possível.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Para obter detalhes da API, consulte DeleteMountTargetem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeFileSystems`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeFileSystems`.

CLI

AWS CLI

Para descrever um sistema de arquivos

O exemplo `describe-file-systems` a seguir descreve o sistema de arquivos especificado.

```
aws efs describe-file-systems \
    --file-system-id fs-c7a0456e
```

Saída:

```
{
    "FileSystems": [
        {
            "OwnerId": "123456789012",
            "CreationToken": "console-d7f56c5f-e433-41ca-8307-9d9c0example",
            "FileSystemId": "fs-c7a0456e",
            "FileSystemArn": "arn:aws:elasticfilesystem:us-
west-2:123456789012:file-system/fs-48499b4d",
            "CreationTime": 1595286880.0,
            "LifeCycleState": "available",
            "Name": "my-file-system",
            "NumberOfMountTargets": 3,
            "SizeInBytes": {
                "Value": 6144,
                "Timestamp": 1600991437.0,
                "ValueInIA": 0,
                "ValueInStandard": 6144
            },
            "PerformanceMode": "generalPurpose",
            "Encrypted": true,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/a59b3472-
e62c-42e4-adcf-30d92example",
            "ThroughputMode": "bursting",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "my-file-system"
                }
            ]
        }
```

```
    ]
}
```

Para ter mais informações, consulte Gerenciar sistemas de arquivos do Amazon EFS no Guia do usuário do Amazon Elastic File System.

- Para obter detalhes da API, consulte DescribeFileSystemsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna a coleção de todos os sistemas de arquivos de propriedade da conta do chamador na região.

```
Get-EFSFileSystem
```

Saída:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : available
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime         : 5/26/2015 4:06:23 PM
CreationToken        : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-4d3c2b1a
...
```

Exemplo 2: retorna os detalhes do sistema de arquivos especificado.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Exemplo 3: retorna os detalhes de um sistema de arquivos usando o token de criação de idempotência que foi especificado no momento em que o sistema de arquivos foi criado.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Para obter detalhes da API, consulte DescribeFileSystemsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de todos os sistemas de arquivos de propriedade da conta do chamador na região.

```
Get-EFSFileSystem
```

Saída:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : available
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime         : 5/26/2015 4:06:23 PM
CreationToken        : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-4d3c2b1a
...
```

Exemplo 2: retorna os detalhes do sistema de arquivos especificado.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Exemplo 3: retorna os detalhes de um sistema de arquivos usando o token de criação de idempotência que foi especificado no momento em que o sistema de arquivos foi criado.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Para obter detalhes da API, consulte DescribeFileSystemsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeMountTargets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeMountTargets`.

CLI

AWS CLI

Para descrever um destino de montagem

O exemplo `describe-mount-targets` a seguir descreve o destino de montagem especificado.

```
aws efs describe-mount-targets \
    --mount-target-id fsmt-f9a14450
```

Saída:

```
{
    "MountTargets": [
        {
            "OwnerId": "123456789012",
            "MountTargetId": "fsmt-f9a14450",
            "FileSystemId": "fs-c7a0456e",
            "SubnetId": "subnet-02bf4c428bexample",
            "LifeCycleState": "creating",
            "IpAddress": "10.0.1.24",
            "NetworkInterfaceId": "eni-02d542216aexample",
            "AvailabilityZoneId": "use2-az2",
            "AvailabilityZoneName": "us-east-2b",
            "VpcId": "vpc-0123456789abcdef0"
        }
    ]
}
```

Para obter mais informações, consulte Criar destinos de montagem no Guia do usuário do Amazon Elastic File System.

- Para obter detalhes da API, consulte DescribeMountTargetsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna a coleção de destinos de montagem associados ao sistema de arquivos especificado.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Saída:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- Para obter detalhes da API, consulte DescribeMountTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de destinos de montagem associados ao sistema de arquivos especificado.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Saída:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- Para obter detalhes da API, consulte DescribeMountTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTags`.

CLI

AWS CLI

Para descrever as tags de um sistema de arquivos

O exemplo `describe-tags` a seguir descreve as tags para o sistema de arquivos especificado.

```
aws efs describe-tags \
    --file-system-id fs-c7a0456e
```

Saída:

```
{
    "Tags": [
        {
            "Key": "Name",
            "Value": "my-file-system"
        },
        {
            "Key": "Department",
            "Value": "Business Intelligence"
        }
    ]
}
```

Para obter mais informações, consulte Gerenciar tags do sistema de arquivos no Guia do usuário do Amazon Elastic File System.

- Para obter detalhes da API, consulte DescribeTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna a coleção de tags atualmente associadas ao sistema de arquivos especificado.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Saída:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de tags atualmente associadas ao sistema de arquivos especificado.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Saída:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para o Amazon EKS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Elastic Kubernetes Service com AWS um kit de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon EKS: mais informações sobre o Amazon EKS.
- Referência da API Amazon EKS: detalhes sobre todas as ações disponíveis do Amazon EKS.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon EKS AWS SDKs
  - Ações para o Amazon EKS usando AWS SDKs
    - Use CreateClustercom um AWS SDK ou CLI
    - Usar CreateFargateProfile com uma CLI
    - Usar CreateNodegroup com uma CLI
    - Use DeleteClustercom um AWS SDK ou CLI
    - Usar DeleteFargateProfile com uma CLI
    - Usar DeleteNodegroup com uma CLI
    - Usar DescribeCluster com uma CLI
    - Usar DescribeFargateProfile com uma CLI
    - Usar DescribeNodegroup com uma CLI
    - Usar DescribeUpdate com uma CLI
    - Usar ListClusters com uma CLI
    - Usar ListFargateProfiles com uma CLI
    - Usar ListNodegroups com uma CLI
    - Usar ListTagsForResource com uma CLI
    - Usar ListUpdates com uma CLI
    - Usar TagResource com uma CLI
    - Usar UntagResource com uma CLI

    - Usar UpdateClusterConfig com uma CLI
    - Usar UpdateClusterVersion com uma CLI
- Cenários para o Amazon EKS usando AWS SDKs
  - Conceitos básicos de clusters gerenciados de kubernetes

Exemplos básicos para o uso do Amazon EKS AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Elastic Kubernetes Service com AWS SDKs.

Exemplos

- Ações para o Amazon EKS usando AWS SDKs
  - Use CreateClustercom um AWS SDK ou CLI
  - Usar CreateFargateProfile com uma CLI
  - Usar CreateNodegroup com uma CLI
  - Use DeleteClustercom um AWS SDK ou CLI
  - Usar DeleteFargateProfile com uma CLI
  - Usar DeleteNodegroup com uma CLI
  - Usar DescribeCluster com uma CLI
  - Usar DescribeFargateProfile com uma CLI
  - Usar DescribeNodegroup com uma CLI
  - Usar DescribeUpdate com uma CLI
  - Usar ListClusters com uma CLI
  - Usar ListFargateProfiles com uma CLI
  - Usar ListNodegroups com uma CLI
  - Usar ListTagsForResource com uma CLI
  - Usar ListUpdates com uma CLI
  - Usar TagResource com uma CLI
  - Usar UntagResource com uma CLI
  - Usar UpdateClusterConfig com uma CLI
  - Usar UpdateClusterVersion com uma CLI

Ações para o Amazon EKS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon EKS com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API Amazon EKS e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para o Amazon EKS usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Elastic Kubernetes Service.

Exemplos

- Use CreateClustercom um AWS SDK ou CLI
- Usar CreateFargateProfile com uma CLI
- Usar CreateNodegroup com uma CLI
- Use DeleteClustercom um AWS SDK ou CLI
- Usar DeleteFargateProfile com uma CLI
- Usar DeleteNodegroup com uma CLI
- Usar DescribeCluster com uma CLI
- Usar DescribeFargateProfile com uma CLI
- Usar DescribeNodegroup com uma CLI
- Usar DescribeUpdate com uma CLI
- Usar ListClusters com uma CLI
- Usar ListFargateProfiles com uma CLI
- Usar ListNodegroups com uma CLI
- Usar ListTagsForResource com uma CLI
- Usar ListUpdates com uma CLI
- Usar TagResource com uma CLI
- Usar UntagResource com uma CLI
- Usar UpdateClusterConfig com uma CLI
- Usar UpdateClusterVersion com uma CLI

Use **`CreateCluster`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Como criar um cluster

Este comando de exemplo cria o cluster `prod` na sua região padrão.

Comando:

```
aws eks create-cluster --name prod \
--role-arn arn:aws:iam::012345678910:role/eks-service-role-
AWSServiceRoleForAmazonEKS-J7ONKE3BQ4PI \
--resources-vpc-config subnetIds=subnet-6782e71e,subnet-
e7e761ac,securityGroupIds=sg-6979fe18
```

Saída:

```
{
    "cluster": {
        "name": "prod",
        "arn": "arn:aws:eks:us-west-2:012345678910:cluster/prod",
        "createdAt": 1527808069.147,
        "version": "1.10",
        "roleArn": "arn:aws:iam::012345678910:role/eks-service-role-
AWSServiceRoleForAmazonEKS-J7ONKE3BQ4PI",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-6782e71e",
                "subnet-e7e761ac"
            ],
            "securityGroupIds": [
                "sg-6979fe18"
            ],
```

```
            "vpcId": "vpc-950809ec"
        },
        "status": "CREATING",
        "certificateAuthority": {}
    }
}
```

Como criar um cluster com acesso ao endpoint privado e registro em log habilitado

Este comando de exemplo cria o cluster `example` na região padrão com acesso ao endpoint público desabilitado e acesso ao endpoint privado e todos os tipos de registro em log habilitados.

Comando:

```
aws eks create-cluster --name example --kubernetes-version 1.12 \
--role-arn arn:aws:iam::012345678910:role/example-cluster-
ServiceRole-1XWBQWYSFRE2Q \
--resources-vpc-
config subnetIds=subnet-0a188dccd2f9a632f,subnet-09290d93da4278664,subnet-0f21dd86e0e9113
 \
--logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

Saída:

```
{
    "cluster": {
        "name": "example",
        "arn": "arn:aws:eks:us-west-2:012345678910:cluster/example",
        "createdAt": 1565804921.901,
        "version": "1.12",
        "roleArn": "arn:aws:iam::012345678910:role/example-cluster-
ServiceRole-1XWBQWYSFRE2Q",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0a188dccd2f9a632f",
                "subnet-09290d93da4278664",
                "subnet-0f21dd86e0e91134a",
                "subnet-0173dead68481a583",
                "subnet-051f70a57ed6fcab6",
                "subnet-01322339c5c7de9b4"
            ],
```

```
            "securityGroupIds": [
                "sg-0c5b580845a031c10"
            ],
            "vpcId": "vpc-0f622c01f68d4afec",
            "endpointPublicAccess": false,
            "endpointPrivateAccess": true
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "status": "CREATING",
        "certificateAuthority": {},
        "platformVersion": "eks.3"
    }
}
```

- Para obter detalhes da API, consulte CreateClusterem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um cluster chamado “prod”.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Saída:

```
Arn                  : arn:aws:eks:us-west-2:012345678901:cluster/prod
```

```
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/10/2018 9:25:31 PM
Endpoint             :
Name                 : prod
PlatformVersion      : eks.3
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678901:role/eks-service-role
Status               : CREATING
Version              : 1.10
```

- Para obter detalhes da API, consulte CreateClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um cluster chamado “prod”.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Saída:

```
Arn                  : arn:aws:eks:us-west-2:012345678901:cluster/prod
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/10/2018 9:25:31 PM
Endpoint             :
Name                 : prod
PlatformVersion      : eks.3
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678901:role/eks-service-role
Status               : CREATING
Version              : 1.10
```

- Para obter detalhes da API, consulte CreateClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
    arn: &str,
    subnet_ids: Vec<String>,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster = client
        .create_cluster()
        .name(name)
        .role_arn(arn)
        .resources_vpc_config(
            VpcConfigRequest::builder()
                .set_subnet_ids(Some(subnet_ids))
                .build(),
        )
        .send()
        .await?;
    println!("cluster created: {:?}", cluster);

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateClusterreferência da API AWS SDK for Rust.

Usar **`CreateFargateProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateFargateProfile`.

CLI

AWS CLI

Exemplo 1: criar um perfil do EKS Fargate para um seletor com um namespace

O exemplo `create-fargate-profile` a seguir cria um perfil do EKS Fargate para um seletor com um namespace.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "default"}]'
```

Saída:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/a2c72bca-318e-abe8-8ed1-27c6d4892e9e",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T12:38:47.368000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default"
            }
        ],
        "status": "CREATING",
        "tags": {}
    }
}
```

Para obter mais informações, consulte Perfil do AWS Fargate – Criar um perfil do Fargate no Guia do usuário do Amazon EKS.

Exemplo 2: criar um perfil do EKS Fargate para um seletor com um namespace e rótulos

O exemplo `create-fargate-profile` a seguir cria um perfil do EKS Fargate para um seletor com um namespace e rótulos.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "default", "labels": {"labelname1":
 "labelvalue1"}}]'
```

Saída:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/88c72bc7-e8a4-fa34-44e4-2f1397224bb3",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T12:33:48.125000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default",
                "labels": {
                    "labelname1": "labelvalue1"
                }
            }
        ],
        "status": "CREATING",
        "tags": {}
    }
}
```

Para obter mais informações, consulte Perfil do AWS Fargate – Criar um perfil do Fargate no Guia do usuário do Amazon EKS.

Exemplo 3: criar um perfil do EKS Fargate para um seletor com um namespace e rótulos, junto com IDs de sub-redes nas quais iniciar um pod.

O exemplo `create-fargate-profile` a seguir cria o perfil do EKS Fargate para um seletor com um namespace e rótulos, junto com IDs de sub-redes nas quais iniciar um pod.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "default", "labels": {"labelname1":
 "labelvalue1"}}]' \
    --subnets '["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d",
 "subnet-0e2907431c9988b72"]'
```

Saída:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/e8c72bc8-e87b-5eb6-57cb-ed4fe57577e3",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T12:35:58.640000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default",
                "labels": {
                    "labelname1": "labelvalue1"
                }
            }
        ],
        "status": "CREATING",
        "tags": {}
    }
}
```

Para obter mais informações, consulte Perfil do AWS Fargate – Criar um perfil do Fargate no Guia do usuário do Amazon EKS.

Exemplo 4: criar um perfil do EKS Fargate para um seletor com vários namespaces e rótulos, junto com IDs de sub-redes nas quais iniciar um pod

O exemplo `create-fargate-profile` a seguir cria um perfil do EKS Fargate para um seletor com vários namespaces e rótulos, junto com IDs de sub-redes nas quais iniciar um pod.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "default1", "labels": {"labelname1":
 "labelvalue1", "labelname2": "labelvalue2"}}, {"namespace": "default2",
 "labels": {"labelname1": "labelvalue1", "labelname2": "labelvalue2"}}]' \
    --subnets '["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d",
 "subnet-0e2907431c9988b72"]' \
    --tags '{"eks-fargate-profile-key-1": "value-1" , "eks-fargate-profile-
key-2": "value-2"}'
```

Saída:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/4cc72bbf-b766-8ee6-8d29-e62748feb3cd",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T12:15:55.271000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default1",
                "labels": {
                    "labelname2": "labelvalue2",
                    "labelname1": "labelvalue1"
```

```
                }
            },
            {
                "namespace": "default2",
                "labels": {
                    "labelname2": "labelvalue2",
                    "labelname1": "labelvalue1"
                }
            }
        ],
        "status": "CREATING",
        "tags": {
            "eks-fargate-profile-key-2": "value-2",
            "eks-fargate-profile-key-1": "value-1"
        }
    }
}
```

Para obter mais informações, consulte Perfil do AWS Fargate – Criar um perfil do Fargate no Guia do usuário do Amazon EKS.

Exemplo 5: criar um perfil do EKS Fargate com um seletor curinga para namespaces e rótulos, junto com IDs de sub-redes nas quais iniciar um pod

O exemplo `create-fargate-profile` a seguir cria um perfil do EKS Fargate para um seletor com vários namespaces e rótulos, junto com IDs de sub-redes nas quais iniciar um pod.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "prod*", "labels": {"labelname*?": "*value1"}},
 {"namespace": "*dev*", "labels": {"labelname*?": "*value*"}}]' \
    --subnets '["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d",
 "subnet-0e2907431c9988b72"]' \
    --tags '{"eks-fargate-profile-key-1": "value-1" , "eks-fargate-profile-
key-2": "value-2"}'
```

Saída:

```
{
```

```
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/e8c72bd6-5966-0bfe-b77b-1802893e5a6f",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T13:05:20.550000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "prod*",
                "labels": {
                    "labelname*?": "*value1"
                }
            },
            {
                "namespace": "*dev*",
                "labels": {
                    "labelname*?": "*value*"
                }
            }
        ],
        "status": "CREATING",
        "tags": {
            "eks-fargate-profile-key-2": "value-2",
            "eks-fargate-profile-key-1": "value-1"
        }
    }
}
```

Para obter mais informações, consulte Perfil do AWS Fargate – Criar um perfil do Fargate no Guia do usuário do Amazon EKS.

- Para obter detalhes da API, consulte CreateFargateProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse cmdlet cria um perfil AWS Fargate para seu cluster Amazon EKS. Você deve ter pelo menos um perfil do Fargate em um cluster para poder programar pods na infraestrutura do Fargate.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName
 TEST -Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae"
 -PodExecutionRoleArn arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole -Selector @{Namespace="default"}
```

Saída:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:38:21 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName  : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : CREATING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes da API, consulte CreateFargateProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet cria um perfil AWS Fargate para seu cluster Amazon EKS. Você deve ter pelo menos um perfil do Fargate em um cluster para poder programar pods na infraestrutura do Fargate.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName
 TEST -Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae"
 -PodExecutionRoleArn arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole -Selector @{Namespace="default"}
```

Saída:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:38:21 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName  : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : CREATING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes da API, consulte CreateFargateProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateNodegroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateNodegroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Exemplo 1: criar um grupo de nós gerenciado para um cluster do Amazon EKS

O exemplo `create-nodegroup` a seguir cria um grupo de nós gerenciados para um cluster do Amazon EKS.

```
aws eks create-nodegroup \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup \
    --node-role arn:aws:iam::111122223333:role/role-name \
    --
subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d" "subnet-09d912bb63ef21b9a"
 \
    --scaling-config minSize=1,maxSize=3,desiredSize=1 \
```

```
    --region us-east-2
```

Saída:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/bac7550f-b8b8-5fbb-4f3e-7502a931119e",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-04T13:19:32.260000-04:00",
        "modifiedAt": "2024-04-04T13:19:32.260000-04:00",
        "status": "CREATING",
        "capacityType": "ON_DEMAND",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 3,
            "desiredSize": 1
        },
        "instanceTypes": [
            "t3.medium"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72, subnet-04ad87f71c6e5ab4d,
 subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "diskSize": 20,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 1
        },
        "tags": {}
    }
}
```

Para obter mais informações, consulte Criar um grupo de nós gerenciados no Guia do usuário do Amazon EKS.

Exemplo 2: criar um grupo de nós gerenciado para um cluster do Amazon EKS com tipos de instância e tamanho de disco personalizados

O exemplo `create-nodegroup` a seguir cria um grupo de nós gerenciados para um cluster do Amazon EKS com instance-types e disk-size personalizados.

```
aws eks create-nodegroup \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup \
    --node-role arn:aws:iam::111122223333:role/role-name \
    --
subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d" "subnet-09d912bb63ef21b9a"
 \
    --scaling-config minSize=1,maxSize=3,desiredSize=1 \
    --capacity-type ON_DEMAND \
    --instance-types 'm5.large' \
    --disk-size 50 \
    --region us-east-2
```

Saída:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/c0c7551b-e4f9-73d9-992c-a450fdb82322",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-04T13:46:07.595000-04:00",
        "modifiedAt": "2024-04-04T13:46:07.595000-04:00",
        "status": "CREATING",
        "capacityType": "ON_DEMAND",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 3,
            "desiredSize": 1
        },
        "instanceTypes": [
            "m5.large"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72",
```

```
            "subnet-04ad87f71c6e5ab4d",
            "subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "diskSize": 50,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 1
        },
        "tags": {}
    }
}
```

Para obter mais informações, consulte Criar um grupo de nós gerenciados no Guia do usuário do Amazon EKS.

Exemplo 3: criar um grupo de nós gerenciados para um cluster do Amazon EKS com instance-types, disk-size, ami-type, capacity-type, update-config, rótulos, taints e tags personalizados.

O exemplo `create-nodegroup` a seguir cria um grupo de nós gerenciados para um cluster do Amazon EKS com instance-types, disk-size, ami-type, capacity-type, update-config, rótulos, taints e tags personalizados.

```
aws eks create-nodegroup  \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup \
    --node-role arn:aws:iam::111122223333:role/role-name \
    --
subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d" "subnet-09d912bb63ef21b9a"
 \
    --scaling-config minSize=1,maxSize=5,desiredSize=4 \
    --instance-types 't3.large' \
    --disk-size 50 \
    --ami-type AL2_x86_64 \
    --capacity-type SPOT \
    --update-config maxUnavailable=2 \
    --labels '{"my-eks-nodegroup-label-1": "value-1" , "my-eks-nodegroup-
label-2": "value-2"}' \
    --taints '{"key": "taint-key-1" , "value": "taint-value-1", "effect":
 "NO_EXECUTE"}' \
```

```
    --tags '{"my-eks-nodegroup-key-1": "value-1" , "my-eks-nodegroup-key-2":
 "value-2"}'
```

Saída:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/88c75524-97af-0cb9-a9c5-7c0423ab5314",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-04T14:05:07.940000-04:00",
        "modifiedAt": "2024-04-04T14:05:07.940000-04:00",
        "status": "CREATING",
        "capacityType": "SPOT",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 5,
            "desiredSize": 4
        },
        "instanceTypes": [
            "t3.large"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "labels": {
            "my-eks-nodegroup-label-2": "value-2",
            "my-eks-nodegroup-label-1": "value-1"
        },
        "taints": [
            {
                "key": "taint-key-1",
                "value": "taint-value-1",
                "effect": "NO_EXECUTE"
            }
        ],
```

```
        "diskSize": 50,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 2
        },
        "tags": {
            "my-eks-nodegroup-key-1": "value-1",
            "my-eks-nodegroup-key-2": "value-2"
        }
    }
}
```

Para obter mais informações, consulte Criar um grupo de nós gerenciados no Guia do usuário do Amazon EKS.

- Para obter detalhes da API, consulte CreateNodegroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet cria um grupo gerenciado de nós de processamento para um cluster do Amazon EKS. Você pode criar um grupo de nós para o cluster somente se for igual à versão atual de Kubernetes para o cluster. Todos os grupos de nós são criados com a versão de lançamento mais recente da AMI para a respectiva versão secundária de Kubernetes do cluster.

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
 -DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
 -NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
 "subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"
```

Saída:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
```

```
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes da API, consulte CreateNodegroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cria um grupo gerenciado de nós de processamento para um cluster do Amazon EKS. Você pode criar um grupo de nós para o cluster somente se for igual à versão atual de Kubernetes para o cluster. Todos os grupos de nós são criados com a versão de lançamento mais recente da AMI para a respectiva versão secundária de Kubernetes do cluster.

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
 -DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
 -NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
 "subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"
```

Saída:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
```

```
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes da API, consulte CreateNodegroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteCluster`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Excluir um ambiente de gerenciamento de cluster do Amazon EKS

O exemplo `delete-cluster` a seguir exclui um ambiente de gerenciamento de cluster do Amazon EKS.

```
aws eks delete-cluster \
    --name my-eks-cluster
```

Saída:

```
{
```

```
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.27",
        "endpoint": "https://DALSJ343KE23J3RN45653DSKJTT647TYD.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "0.0.0.0/0"
            ]
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
```

```
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
DALSJ343KE23J3RN45653DSKJTT647TYD"
            }
        },
        "status": "DELETING",
        "certificateAuthority": {
            "data": "XXX_CA_DATA_XXX"
        },
        "platformVersion": "eks.16",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
}
```

Para obter mais informações, consulte Excluir um cluster do Amazon EKS no Guia do usuário do Amazon EKS.

- Para obter detalhes da API, consulte DeleteClusterem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet exclui um ambiente de gerenciamento de cluster do Amazon EKS.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSCluster (DeleteCluster)" on target "DEV-KUBE-
CL".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 9:33:25 AM
Endpoint             : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : DEV-KUBE-CL
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : DELETING
Tags                 : {}
Version              : 1.14
```

- Para obter detalhes da API, consulte DeleteClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui um ambiente de gerenciamento de cluster do Amazon EKS.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSCluster (DeleteCluster)" on target "DEV-KUBE-
CL".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 9:33:25 AM
Endpoint             : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : DEV-KUBE-CL
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : DELETING
Tags                 : {}
Version              : 1.14
```

- Para obter detalhes da API, consulte DeleteClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn remove_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster_deleted = client.delete_cluster().name(name).send().await?;
```

```
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteClusterreferência da API AWS SDK for Rust.

Usar **`DeleteFargateProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteFargateProfile`.

CLI

AWS CLI

Exemplo 1: criar um perfil do EKS Fargate para um seletor com um namespace

O exemplo `delete-fargate-profile` a seguir cria um perfil do EKS Fargate para um seletor com um namespace.

```
aws eks delete-fargate-profile \
    --cluster-name my-eks-cluster \
    --fargate-profile-name my-fargate-profile
```

Saída:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/1ac72bb3-3fc6-2631-f1e1-98bff53bed62",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T11:48:39.975000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
```

```
                "namespace": "default",
                "labels": {
                    "foo": "bar"
                }
            }
        ],
        "status": "DELETING",
        "tags": {}
    }
}
```

Para obter mais informações, consulte Perfil do AWS Fargate – Excluir um Fargate no Guia do usuário do Amazon EKS.

- Para obter detalhes da API, consulte DeleteFargateProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse cmdlet exclui um perfil do AWS Fargate. Quando você exclui um perfil do Fargate, qualquer pod em execução no Fargate que foi criado com o perfil é excluído.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSFargateProfile (DeleteFargateProfile)" on
 target "EKSFargate".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
```

```
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : DELETING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes da API, consulte DeleteFargateProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet exclui um perfil do AWS Fargate. Quando você exclui um perfil do Fargate, qualquer pod em execução no Fargate que foi criado com o perfil é excluído.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSFargateProfile (DeleteFargateProfile)" on
 target "EKSFargate".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : DELETING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes da API, consulte DeleteFargateProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteNodegroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteNodegroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Exemplo 1: excluir um grupo de nós gerenciado em um cluster do Amazon EKS

O exemplo `delete-nodegroup` a seguir exclui um grupo de nós gerenciado em um cluster do Amazon EKS.

```
aws eks delete-nodegroup \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup
```

Saída:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/1ec75f5f-0e21-dcc0-b46e-f9c442685cd8",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-08T13:25:15.033000-04:00",
        "modifiedAt": "2024-04-08T13:25:31.252000-04:00",
        "status": "DELETING",
        "capacityType": "SPOT",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 5,
            "desiredSize": 4
        },
        "instanceTypes": [
            "t3.large"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72",
```

```
            "subnet-04ad87f71c6e5ab4d",
            "subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "labels": {
            "my-eks-nodegroup-label-2": "value-2",
            "my-eks-nodegroup-label-1": "value-1"
        },
        "taints": [
            {
                "key": "taint-key-1",
                "value": "taint-value-1",
                "effect": "NO_EXECUTE"
            }
        ],
        "diskSize": 50,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 2
        },
        "tags": {
            "my-eks-nodegroup-key-1": "value-1",
            "my-eks-nodegroup-key-2": "value-2"
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteNodegroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet exclui um grupo de nós do Amazon EKS para um cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSNodegroup (DeleteNodegroup)" on target
 "ProdEKSNodeGroup".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 11:01:16 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      : Amazon.EKS.Model.NodegroupResources
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : DELETING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes da API, consulte DeleteNodegroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui um grupo de nós do Amazon EKS para um cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSNodegroup (DeleteNodegroup)" on target
 "ProdEKSNodeGroup".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 11:01:16 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      : Amazon.EKS.Model.NodegroupResources
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : DELETING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes da API, consulte DeleteNodegroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeCluster`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Descrever o complemento do EKS em execução ativa em seu cluster do Amazon EKS

O exemplo `describe-cluster` a seguir executa ativamente o complemento do Amazon EKS em seu cluster do Amazon EKS.

```
aws eks describe-cluster \
    --name my-eks-cluster
```

Saída:

```
{
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.26",
        "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "22.19.18.2/32"
            ]
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
```

```
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
            }
        },
        "status": "ACTIVE",
        "certificateAuthority": {
            "data": "CA_DATA_STRING..."
        },
        "platformVersion": "eks.14",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "health": {
            "issues": []
        },
        "accessConfig": {
```

```
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeClusterem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet retorna informações descritivas sobre um cluster do Amazon EKS.

```
Get-EKSCluster -Name "PROD"
```

Saída:

```
Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/PROD
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 6:46:17 AM
Endpoint             : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : PROD
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : ACTIVE
Tags                 : {}
Version              : 1.14
```

- Para obter detalhes da API, consulte DescribeClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet retorna informações descritivas sobre um cluster do Amazon EKS.

```
Get-EKSCluster -Name "PROD"
```

Saída:

```
Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/PROD
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 6:46:17 AM
Endpoint             : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : PROD
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : ACTIVE
Tags                 : {}
Version              : 1.14
```

- Para obter detalhes da API, consulte DescribeClusterem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeFargateProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeFargateProfile`.

CLI

AWS CLI

Descrever um perfil do Fargate

O exemplo `describe-fargate-profile` a seguir descreve um perfil do Fargate.

```
aws eks describe-fargate-profile \
    --cluster-name my-eks-cluster \
    --fargate-profile-name my-fargate-profile
```

Saída:

```
{
    "fargateProfile": {
```

```
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/96c766ce-43d2-f9c9-954c-647334391198",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-04-11T10:42:52.486000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-
cluster-farga-FargatePodExecutionRole-1htfAaJdJUEO",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "prod*",
                "labels": {
                    "labelname*?": "*value1"
                }
            },
            {
                "namespace": "*dev*",
                "labels": {
                    "labelname*?": "*value*"
                }
            }
        ],
        "status": "ACTIVE",
        "tags": {
            "eks-fargate-profile-key-2": "value-2",
            "eks-fargate-profile-key-1": "value-1"
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeFargateProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse cmdlet retorna informações descritivas sobre um perfil do AWS Fargate.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Saída:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : ACTIVE
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes da API, consulte DescribeFargateProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet retorna informações descritivas sobre um perfil do AWS Fargate.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Saída:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : ACTIVE
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes da API, consulte DescribeFargateProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeNodegroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeNodegroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Descrever um grupo de nós gerenciado para um cluster do Amazon EKS

O exemplo `describe-nodegroup` a seguir descreve um grupo de nós gerenciado em um cluster do Amazon EKS.

```
aws eks describe-nodegroup \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup
```

Saída:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/a8c75f2f-df78-a72f-4063-4b69af3de5b1",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-08T11:42:10.555000-04:00",
        "modifiedAt": "2024-04-08T11:44:12.402000-04:00",
        "status": "ACTIVE",
        "capacityType": "ON_DEMAND",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 3,
            "desiredSize": 1
        },
        "instanceTypes": [
```

```
            "t3.medium"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "labels": {},
        "resources": {
            "autoScalingGroups": [
                {
                    "name": "eks-my-eks-nodegroup-a8c75f2f-df78-
a72f-4063-4b69af3de5b1"
                }
            ]
        },
        "diskSize": 20,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 1
        },
        "tags": {}
    }
}
```

- Para obter detalhes da API, consulte DescribeNodegroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet retorna informações descritivas sobre um grupo de nós do Amazon EKS.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Saída:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes da API, consulte DescribeNodegroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet retorna informações descritivas sobre um grupo de nós do Amazon EKS.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Saída:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
```

```
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes da API, consulte DescribeNodegroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeUpdate`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeUpdate`.

CLI

AWS CLI

Exemplo 1: descrever uma atualização para um cluster

O exemplo `describe-update` a seguir descreve uma atualização para um cluster chamado.

```
aws eks describe-update \
    --name my-eks-cluster \
    --update-id 10bddb13-a71b-425a-b0a6-71cd03e59161
```

Saída:

```
{
    "update": {
        "id": "10bddb13-a71b-425a-b0a6-71cd03e59161",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "false"
            },
```

```
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            }
        ],
        "createdAt": "2024-03-14T10:01:26.297000-04:00",
        "errors": []
    }
}
```

Para obter mais informações, consulte Atualizar uma versão do Kubernetes do cluster do Amazon EKS no Guia do usuário do Amazon EKS.

Exemplo 2: descrever uma atualização para um cluster

O exemplo `describe-update` a seguir descreve uma atualização para um cluster chamado.

```
aws eks describe-update \
    --name my-eks-cluster \
    --update-id e4994991-4c0f-475a-a040-427e6da52966
```

Saída:

```
{
    "update": {
        "id": "e4994991-4c0f-475a-a040-427e6da52966",
        "status": "Successful",
        "type": "AssociateEncryptionConfig",
        "params": [
            {
                "type": "EncryptionConfig",
                "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
            }
        ],
        "createdAt": "2024-03-14T11:01:26.297000-04:00",
        "errors": []
    }
}
```

Para obter mais informações, consulte Atualizar uma versão do Kubernetes do cluster do Amazon EKS no Guia do usuário do Amazon EKS.

Exemplo 3: descrever uma atualização para um cluster

O exemplo `describe-update` a seguir descreve uma atualização para um cluster chamado.

```
aws eks describe-update \
    --name my-eks-cluster \
    --update-id b5f0ba18-9a87-4450-b5a0-825e6e84496f
```

Saída:

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "Successful",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.29"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
        "createdAt": "2024-03-14T12:05:26.297000-04:00",
        "errors": []
    }
}
```

Para obter mais informações, consulte Atualizar uma versão do Kubernetes do cluster do Amazon EKS no Guia do usuário do Amazon EKS.

- Para obter detalhes da API, consulte DescribeUpdateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet retorna informações descritivas sobre uma atualização no cluster do Amazon EKS ou grupo de nós gerenciados associado.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Saída:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Para obter detalhes da API, consulte DescribeUpdateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet retorna informações descritivas sobre uma atualização no cluster do Amazon EKS ou grupo de nós gerenciados associado.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Saída:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Para obter detalhes da API, consulte DescribeUpdateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListClusters`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListClusters`.

CLI

AWS CLI

Para listar todos os complementos instalados em seu cluster do Amazon EKS chamado `my-eks-cluster`

O exemplo `list-clusters` a seguir lista todos os complementos instalados em seu cluster do Amazon EKS chamado my-eks-cluster.

```
aws eks list-clusters
```

Saída:

```
{
    "clusters": [
        "prod",
        "qa",
        "stage",
        "my-eks-cluster"
    ]
}
```

- Para obter detalhes da API, consulte ListClustersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse cmdlet lista os clusters do Amazon EKS Conta da AWS em sua região especificada.

```
Get-EKSClusterList
```

Saída:

```
PROD
```

- Para obter detalhes da API, consulte ListClustersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet lista os clusters do Amazon EKS Conta da AWS em sua região especificada.

```
Get-EKSClusterList
```

Saída:

```
 PROD
```

- Para obter detalhes da API, consulte ListClustersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListFargateProfiles`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListFargateProfiles`.

CLI

AWS CLI

Para listar todos os perfis de fargate em seu cluster do Amazon EKS chamado `my-eks-cluster`

O exemplo `list-fargate-profiles` a seguir lista todos os perfis do Fargate em seu cluster do Amazon EKS chamado my-eks-cluster.

```
aws eks list-fargate-profiles \
    --cluster-name my-eks-cluster
```

Saída:

```
{
    "fargateProfileNames": [
        "my-fargate-profile"
    ]
}
```

- Para obter detalhes da API, consulte ListFargateProfilesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse cmdlet lista os perfis do AWS Fargate associados ao cluster especificado no seu Conta da AWS na região especificada.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Saída:

```
EKSFargate
EKSFargateProfile
```

- Para obter detalhes da API, consulte ListFargateProfilesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet lista os perfis do AWS Fargate associados ao cluster especificado no seu Conta da AWS na região especificada.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Saída:

```
EKSFargate
EKSFargateProfile
```

- Para obter detalhes da API, consulte ListFargateProfilesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListNodegroups`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListNodegroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de clusters gerenciados de kubernetes

CLI

AWS CLI

Listar todos os grupos de nós em um cluster do Amazon EKS

O exemplo `list-nodegroups` a seguir lista todos os grupos de nós gerenciados em um cluster do Amazon EKS.

```
aws eks list-nodegroups \
    --cluster-name my-eks-cluster
```

Saída:

```
{
    "nodegroups": [
        "my-eks-managed-node-group",
        "my-eks-nodegroup"
    ]
}
```

- Para obter detalhes da API, consulte ListNodegroupsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse cmdlet lista os grupos de nós do Amazon EKS associados ao cluster especificado em você Conta da AWS na região especificada.

```
Get-EKSNodegroupList -ClusterName PROD
```

Saída:

```
ProdEKSNodeGroup
```

- Para obter detalhes da API, consulte ListNodegroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet lista os grupos de nós do Amazon EKS associados ao cluster especificado em você Conta da AWS na região especificada.

```
Get-EKSNodegroupList -ClusterName PROD
```

Saída:

```
ProdEKSNodeGroup
```

- Para obter detalhes da API, consulte ListNodegroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListTagsForResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListTagsForResource`.

CLI

AWS CLI

Exemplo 1: listar todas as tags de um ARN do cluster do Amazon EKS

O exemplo `list-tags-for-resource` a seguir lista todas as tags de um ARN do cluster do Amazon EKS.

```
aws eks list-tags-for-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster
```

Saída:

```
{
    "tags": {
```

```
        "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
        "alpha.eksctl.io/cluster-name": "my-eks-cluster",
        "karpenter.sh/discovery": "my-eks-cluster",
        "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
        "auto-delete": "no",
        "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
        "EKS-Cluster-Name": "my-eks-cluster",
        "alpha.eksctl.io/cluster-oidc-enabled": "true",
        "aws:cloudformation:logical-id": "ControlPlane",
        "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
        "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
    }
}
```

Exemplo 2: listar todas as tags de um ARN do grupo de nós do Amazon EKS

O exemplo `list-tags-for-resource` a seguir lista todas as tags de um ARN do grupo de nós do Amazon EKS.

```
aws eks list-tags-for-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/
my-eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c
```

Saída:

```
{
    "tags": {
        "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-nodegroup-my-eks-
managed-node-group",
        "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-nodegroup-my-eks-managed-node-
group/eaa20310-e219-11ee-b851-0ab9ad8228ff",
        "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
        "EKS-Cluster-Name": "my-eks-cluster",
        "alpha.eksctl.io/nodegroup-type": "managed",
        "NodeGroup Name 1": "my-eks-managed-node-group",
        "k8s.io/cluster-autoscaler/enabled": "true",
        "nodegroup-role": "worker",
        "alpha.eksctl.io/cluster-name": "my-eks-cluster",
        "alpha.eksctl.io/nodegroup-name": "my-eks-managed-node-group",
```

```
        "karpenter.sh/discovery": "my-eks-cluster",
        "NodeGroup Name 2": "AmazonLinux-Linux-Managed-NG-v1-26-v1",
        "auto-delete": "no",
        "k8s.io/cluster-autoscaler/my-eks-cluster": "owned",
        "aws:cloudformation:logical-id": "ManagedNodeGroup",
        "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z"
    }
}
```

Exemplo 3: listar todas as tags de um ARN do perfil do Amazon EKS Fargate

O exemplo `list-tags-for-resource` a seguir lista todas as tags de um ARN do perfil do Amazon EKS Fargate.

```
aws eks list-tags-for-resource  \
    --resource-arn arn:aws:eks:us-east-2:111122223333:fargateprofile/my-eks-
cluster/my-fargate-profile/d6c76780-e541-0725-c816-36754cab734b
```

Saída:

```
{
    "tags": {
        "eks-fargate-profile-key-2": "value-2",
        "eks-fargate-profile-key-1": "value-1"
    }
}
```

Exemplo 4: Para listar todas as tags de um Add-on ARN do Amazon EKS

O `list-tags-for-resource` exemplo a seguir lista todas as tags de um Add-on ARN do Amazon EKS.

```
aws eks list-tags-for-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/vpc-
cni/0ec71efc-98dd-3203-60b0-4b939b2a5e5f
```

Saída:

```
{
```

```
    "tags": {
        "eks-addon-key-2": "value-2",
        "eks-addon-key-1": "value-1"
    }
}
```

Exemplo 5: listar todas as tags de um ARN provedor de identidade OIDC do Amazon EKS

O exemplo `list-tags-for-resource` a seguir lista todas as tags de um ARN do provedor de identidade OIDC do Amazon EKS.

```
aws eks list-tags-for-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:identityproviderconfig/my-eks-cluster/oidc/my-identity-provider/8ac76722-78e4-cec1-ed76-d49eea058622
```

Saída:

```
{
    "tags": {
        "my-identity-provider": "test"
    }
}
```

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet lista as tags para um recurso do Amazon EKS.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Saída:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet lista as tags para um recurso do Amazon EKS.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Saída:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListUpdates`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListUpdates`.

CLI

AWS CLI

Para listar as atualizações de um cluster

Este comando de exemplo lista as atualizações atuais de um cluster chamado `example` na sua região padrão.

Comando:

```
aws eks list-updates --name example
```

Saída:

```
{
    "updateIds": [
        "10bddb13-a71b-425a-b0a6-71cd03e59161"
```

```
    ]
}
```

- Para obter detalhes da API, consulte ListUpdatesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse cmdlet lista as atualizações associadas a um cluster ou grupo de nós gerenciados do Amazon EKS em seu Conta da AWS, na região especificada.

```
Get-EKSUpdateList -Name "PROD"
```

Saída:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Para obter detalhes da API, consulte ListUpdatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet lista as atualizações associadas a um cluster ou grupo de nós gerenciados do Amazon EKS em seu Conta da AWS, na região especificada.

```
Get-EKSUpdateList -Name "PROD"
```

Saída:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Para obter detalhes da API, consulte ListUpdatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`TagResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `TagResource`.

CLI

AWS CLI

Exemplo 1: adicionar as tags especificadas a um cluster do Amazon EKS

O exemplo `tag-resource` a seguir adiciona as tags especificadas a um cluster do Amazon EKS.

```
aws eks tag-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster \
    --tag 'my-eks-cluster-test-1=test-value-1,my-eks-cluster-dev-1=dev-value-2'
```

Este comando não produz saída.

Exemplo 2: adicionar as tags especificadas a um grupo de nós do Amazon EKS

O exemplo `tag-resource` a seguir adiciona as tags especificadas a um grupo de nós do Amazon EKS.

```
aws eks tag-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/
my-eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c \
    --tag 'my-eks-nodegroup-test-1=test-value-1,my-eks-nodegroup-dev-1=dev-
value-2'
```

Este comando não produz saída.

- Para obter detalhes da API, consulte TagResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet associa as tags especificadas a um recurso com o resourceArn especificado.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -Tag @{Name = "EKSPRODCLUSTER"}
```

- Para obter detalhes da API, consulte TagResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet associa as tags especificadas a um recurso com o resourceArn especificado.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -Tag @{Name = "EKSPRODCLUSTER"}
```

- Para obter detalhes da API, consulte TagResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UntagResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UntagResource`.

CLI

AWS CLI

Exemplo 1: excluir as tags especificadas de um cluster do Amazon EKS

O exemplo `untag-resource` a seguir exclui as tags especificadas de um cluster do Amazon EKS.

```
aws eks untag-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster \
    --tag-keys "my-eks-cluster-test-1" "my-eks-cluster-dev-1"
```

Este comando não produz saída.

Exemplo 2: excluir as tags especificadas de um grupo de nós do Amazon EKS

O exemplo `untag-resource` a seguir exclui as tags especificadas de um grupo de nós do Amazon EKS.

```
aws eks untag-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/
my-eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c \
    --tag-keys "my-eks-nodegroup-test-1" "my-eks-nodegroup-dev-1"
```

Este comando não produz saída.

- Para obter detalhes da API, consulte UntagResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet exclui tags especificadas de um recurso do EKS.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/
PROD" -TagKey "Name"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
 "arn:aws:eks:us-west-2:012345678912:cluster/PROD".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte UntagResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui tags especificadas de um recurso do EKS.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/
PROD" -TagKey "Name"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
 "arn:aws:eks:us-west-2:012345678912:cluster/PROD".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte UntagResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateClusterConfig`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateClusterConfig`.

CLI

AWS CLI

Para atualizar o acesso ao endpoint do cluster

Este comando de exemplo atualiza um cluster para desativar o acesso público ao endpoint e habilitar o acesso ao endpoint privado.

Comando:

```
aws eks update-cluster-config --name example \
--resources-vpc-config endpointPublicAccess=false,endpointPrivateAccess=true
```

Saída:

```
{
    "update": {
        "id": "ec883c93-2e9e-407c-a22f-8f6fa6e67d4f",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "false"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            }
        ],
        "createdAt": 1565806986.506,
        "errors": []
    }
```

```
}
```

Para habilitar o log para um cluster

Este comando de exemplo habilita todos os tipos de registro do ambiente de gerenciamento de cluster para um cluster chamado `example`.

Comando:

```
aws eks update-cluster-config --name example \
--logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

Saída:

```
{
    "update": {
        "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1565807210.37,
        "errors": []
    }
}
```

- Para obter detalhes da API, consulte UpdateClusterConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: atualiza uma configuração de cluster do Amazon EKS. O cluster continua funcionando durante a atualização.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Saída:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Para obter detalhes da API, consulte UpdateClusterConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: atualiza uma configuração de cluster do Amazon EKS. O cluster continua funcionando durante a atualização.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Saída:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Para obter detalhes da API, consulte UpdateClusterConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateClusterVersion`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateClusterVersion`.

CLI

AWS CLI

Para atualizar um cluster do Amazon EKS chamado `my-eks-cluster` para a versão especificada do Kubernetes

O exemplo `update-cluster-version` a seguir atualiza um cluster do Amazon EKS para a versão especificada do Kubernetes.

```
aws eks update-cluster-version \
    --name my-eks-cluster \
    --kubernetes-version 1.27
```

Saída:

```
{
    "update": {
        "id": "e4091a28-ea14-48fd-a8c7-975aeb469e8a",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.27"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.16"
            }
        ],
        "createdAt": "2024-04-12T16:56:01.082000-04:00",
        "errors": []
    }
}
```

Para obter mais informações, consulte Atualizar uma versão do Kubernetes do cluster do Amazon EKS no Guia do usuário do Amazon EKS.

- Para obter detalhes da API, consulte UpdateClusterVersionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse cmdlet atualiza um cluster do Amazon EKS para a versão especificada do Kubernetes. O cluster continua funcionando durante a atualização.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Saída:

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- Para obter detalhes da API, consulte UpdateClusterVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet atualiza um cluster do Amazon EKS para a versão especificada do Kubernetes. O cluster continua funcionando durante a atualização.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Saída:

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- Para obter detalhes da API, consulte UpdateClusterVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para o Amazon EKS usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon EKS com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon EKS ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos de clusters gerenciados de kubernetes

Conceitos básicos de clusters gerenciados de kubernetes

O exemplo de código a seguir mostra como:

- Criar uma VPC para um cluster do EKS.
- Criar perfis do IAM para um cluster do EKS.
- Criação do seu cluster de EKS
- Configurar o kubectl para se comunicar com o cluster.
- Crie um grupo de nós gerenciados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
```

```
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
```

```
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")
```

```
echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"
```

```
# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"
```

```
# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
```

```
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"
```

```
NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
```

```
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
```

```
done
echo "==========================================="

# Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup

- DeleteRole
- DeleteStack
- DescribeCluster
- DescribeNodegroup
- DescribeStacks
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

Exemplos de código para o Elastic Beanstalk usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Elastic Beanstalk com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Elastic Beanstalk: mais informações sobre o Elastic Beanstalk.
- Referência de API do Elastic Beanstalk: detalhes sobre todas as ações disponíveis do Elastic Beanstalk.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de uso do Elastic Beanstalk AWS SDKs
  - Ações para o Elastic Beanstalk usando AWS SDKs

- Use DescribeApplicationscom um AWS SDK ou CLI
- Use ListAvailableSolutionStackscom um AWS SDK ou CLI
- Use UpdateApplicationcom um AWS SDK ou CLI

Exemplos básicos de uso do Elastic Beanstalk AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Elastic Beanstalk com AWS SDKs.

Exemplos

- Ações para o Elastic Beanstalk usando AWS SDKs
  - Use DescribeApplicationscom um AWS SDK ou CLI
  - Use ListAvailableSolutionStackscom um AWS SDK ou CLI
  - Use UpdateApplicationcom um AWS SDK ou CLI

Ações para o Elastic Beanstalk usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Elastic AWS Beanstalk com SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Elastic Beanstalk.

Exemplos

- Use DescribeApplicationscom um AWS SDK ou CLI
- Use ListAvailableSolutionStackscom um AWS SDK ou CLI
- Use UpdateApplicationcom um AWS SDK ou CLI

Use **`DescribeApplications`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeApplications`.

CLI

AWS CLI

Visualizar uma lista de aplicações

O comando a seguir recupera informações sobre aplicações na região atual:

```
aws elasticbeanstalk describe-applications
```

Saída:

```
{
    "Applications": [
        {
            "ApplicationName": "ruby",
            "ConfigurationTemplates": [],
            "DateUpdated": "2015-08-13T21:05:44.376Z",
            "Versions": [
                "Sample Application"
            ],
            "DateCreated": "2015-08-13T21:05:44.376Z"
        },
        {
            "ApplicationName": "pythonsample",
            "Description": "Application created from the EB CLI using \"eb init
\"",
            "Versions": [
                "Sample Application"
            ],
            "DateCreated": "2015-08-13T19:05:43.637Z",
            "ConfigurationTemplates": [],
            "DateUpdated": "2015-08-13T19:05:43.637Z"
        },
        {
            "ApplicationName": "nodejs-example",
            "ConfigurationTemplates": [],
            "DateUpdated": "2015-08-06T17:50:02.486Z",
            "Versions": [
                "add elasticache",
                "First Release"
            ],
            "DateCreated": "2015-08-06T17:50:02.486Z"
```

```
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeApplicationsem Referência de AWS CLI Comandos.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Class to manage Elastic Beanstalk applications
class ElasticBeanstalkManager
  def initialize(eb_client, logger: Logger.new($stdout))
    @eb_client = eb_client
    @logger = logger
  end

  # Lists applications and their environments
  def list_applications
    @eb_client.describe_applications.applications.each do |application|
      log_application_details(application)
      list_environments(application.application_name)
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Elastic Beanstalk Service Error: #{e.message}")
  end

  private

  # Logs application details
  def log_application_details(application)
    @logger.info("Name:        #{application.application_name}")
    @logger.info("Description: #{application.description}")
  end
```

```
  # Lists and logs details of environments for a given application
  def list_environments(application_name)
    @eb_client.describe_environments(application_name:
 application_name).environments.each do |env|
      @logger.info("  Environment:  #{env.environment_name}")
      @logger.info("    URL:        #{env.cname}")
      @logger.info("    Health:     #{env.health}")
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Error listing environments for application
 #{application_name}: #{e.message}")
  end
end
```

- Para obter detalhes da API, consulte DescribeApplicationsa Referência AWS SDK para Ruby da API.

Use **`ListAvailableSolutionStacks`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListAvailableSolutionStacks`.

CLI

AWS CLI

Visualizar as pilhas de soluções

O comando a seguir lista as pilhas de soluções para todas as configurações de plataforma disponíveis atualmente e todas as que você usou no passado:

```
aws elasticbeanstalk list-available-solution-stacks
```

Resultado (abreviado):

```
{
    "SolutionStacks": [
        "64bit Amazon Linux 2015.03 v2.0.0 running Node.js",
        "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.6",
        "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.5",
        "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.4",
```

```
        "64bit Amazon Linux 2015.03 v2.0.0 running Python 3.4",
        "64bit Amazon Linux 2015.03 v2.0.0 running Python 2.7",
        "64bit Amazon Linux 2015.03 v2.0.0 running Python",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.2 (Puma)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.2 (Passenger
 Standalone)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.1 (Puma)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.1 (Passenger
 Standalone)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.0 (Puma)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.0 (Passenger
 Standalone)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 1.9.3",
        "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8 Java 8",
        "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 7 Java 7",
        "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 7 Java 6",
        "64bit Windows Server Core 2012 R2 running IIS 8.5",
        "64bit Windows Server 2012 R2 running IIS 8.5",
        "64bit Windows Server 2012 running IIS 8",
        "64bit Windows Server 2008 R2 running IIS 7.5",
        "64bit Amazon Linux 2015.03 v2.0.0 running Docker 1.6.2",
        "64bit Amazon Linux 2015.03 v2.0.0 running Multi-container Docker 1.6.2
 (Generic)",
        "64bit Debian jessie v2.0.0 running GlassFish 4.1 Java 8 (Preconfigured -
 Docker)",
        "64bit Debian jessie v2.0.0 running GlassFish 4.0 Java 7 (Preconfigured -
 Docker)",
        "64bit Debian jessie v2.0.0 running Go 1.4 (Preconfigured - Docker)",
        "64bit Debian jessie v2.0.0 running Go 1.3 (Preconfigured - Docker)",
        "64bit Debian jessie v2.0.0 running Python 3.4 (Preconfigured - Docker)",
    ],
    "SolutionStackDetails": [
        {
            "PermittedFileTypes": [
                "zip"
            ],
            "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running
 Node.js"
        },
        ...
    ]
}
```

- Para obter detalhes da API, consulte ListAvailableSolutionStacksem Referência de AWS CLI Comandos.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Manages listing of AWS Elastic Beanstalk solution stacks
# @param [Aws::ElasticBeanstalk::Client] eb_client
# @param [String] filter - Returns subset of results based on match
# @param [Logger] logger
class StackLister
  # Initialize with AWS Elastic Beanstalk client
  def initialize(eb_client, filter, logger: Logger.new($stdout))
    @eb_client = eb_client
    @filter = filter.downcase
    @logger = logger
  end

  # Lists and logs Elastic Beanstalk solution stacks
  def list_stacks
    stacks = @eb_client.list_available_solution_stacks.solution_stacks
    orig_length = stacks.length
    filtered_length = 0

    stacks.each do |stack|
      if @filter.empty? || stack.downcase.include?(@filter)
        @logger.info(stack)
        filtered_length += 1
      end
    end

    log_summary(filtered_length, orig_length)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Error listing solution stacks: #{e.message}")
```

```
  end

  private

  # Logs summary of listed stacks
  def log_summary(filtered_length, orig_length)
    if @filter.empty?
      @logger.info("Showed #{orig_length} stack(s)")
    else
      @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
    end
  end
end
```

- Para obter detalhes da API, consulte ListAvailableSolutionStacksa Referência AWS SDK para Ruby da API.

Use **`UpdateApplication`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateApplication`.

CLI

AWS CLI

Alterar a descrição de uma aplicação

O comando a seguir atualiza a descrição de uma aplicação chamada `my-app`:

```
aws elasticbeanstalk update-application --application-name my-app --
description "my Elastic Beanstalk application"
```

Saída:

```
{
    "Application": {
        "ApplicationName": "my-app",
        "Description": "my Elastic Beanstalk application",
        "Versions": [
            "2fba-stage-150819_234450",
            "bf07-stage-150820_214945",
```

```
            "93f8",
            "fd7c-stage-150820_000431",
            "22a0-stage-150819_185942"
        ],
        "DateCreated": "2015-08-13T19:15:50.449Z",
        "ConfigurationTemplates": [],
        "DateUpdated": "2015-08-20T22:34:56.195Z"
    }
}
```

- Para obter detalhes da API, consulte UpdateApplicationem Referência de AWS CLI Comandos.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Manages deployment of Rails applications to AWS Elastic Beanstalk
class RailsAppDeployer
  def initialize(eb_client, s3_client, app_name, logger: Logger.new($stdout))
    @eb_client = eb_client
    @s3_client = s3_client
    @app_name = app_name
    @logger = logger
  end

  # Deploys the latest application version to Elastic Beanstalk
  def deploy
    create_storage_location
    zip_file_name = create_zip_file
    upload_zip_to_s3(zip_file_name)
    create_and_deploy_new_application_version(zip_file_name)
  end

  private
```

```
  # Creates a new S3 storage location for the application
  def create_storage_location
    resp = @eb_client.create_storage_location
    @logger.info("Created storage location in bucket #{resp.s3_bucket}")
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create storage location: #{e.message}")
  end

  # Creates a ZIP file of the application using git
  def create_zip_file
    zip_file_basename = SecureRandom.urlsafe_base64
    zip_file_name = "#{zip_file_basename}.zip"
    `git archive --format=zip -o #{zip_file_name} HEAD`
    zip_file_name
  end

  # Uploads the ZIP file to the S3 bucket
  def upload_zip_to_s3(zip_file_name)
    zip_contents = File.read(zip_file_name)
    key = "#{@app_name}/#{zip_file_name}"
    @s3_client.put_object(body: zip_contents, bucket: fetch_bucket_name, key:
key)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to upload ZIP file to S3: #{e.message}")
  end

  # Fetches the S3 bucket name from Elastic Beanstalk application versions
  def fetch_bucket_name
    app_versions = @eb_client.describe_application_versions(application_name:
@app_name)
    av = app_versions.application_versions.first
    av.source_bundle.s3_bucket
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch bucket name: #{e.message}")
    raise
  end

  # Creates a new application version and deploys it
  def create_and_deploy_new_application_version(zip_file_name)
    version_label = File.basename(zip_file_name, '.zip')
    @eb_client.create_application_version(
      process: false,
      application_name: @app_name,
```

```
      version_label: version_label,
      source_bundle: {
        s3_bucket: fetch_bucket_name,
        s3_key: "#{@app_name}/#{zip_file_name}"
      },
      description: "Updated #{Time.now.strftime('%d/%m/%Y')}"
    )
    update_environment(version_label)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create or deploy application version: #{e.message}")
  end

  # Updates the environment to the new application version
  def update_environment(version_label)
    env_name = fetch_environment_name
    @eb_client.update_environment(
      environment_name: env_name,
      version_label: version_label
    )
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to update environment: #{e.message}")
  end

  # Fetches the environment name of the application
  def fetch_environment_name
    envs = @eb_client.describe_environments(application_name: @app_name)
    envs.environments.first.environment_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch environment name: #{e.message}")
    raise
  end
end
```

- Para obter detalhes da API, consulte UpdateApplicationa Referência AWS SDK para Ruby da API.

Exemplos de código para o Elastic Load Balancing - Versão 1 usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Elastic Load Balancing — Versão 1 com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário do Elastic Load Balancing Versão 1: mais informações sobre o Elastic Load Balancing Versão 1.
- Referência de API do Elastic Load Balancing Versão 1: detalhes sobre todas as ações disponíveis do Elastic Load Balancing Versão 1.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Elastic Load Balancing - Versão 1 usando AWS SDKs
  - Ações para o Elastic Load Balancing - Versão 1 usando AWS SDKs
    - Usar AddTags com uma CLI
    - Usar ApplySecurityGroupsToLoadBalancer com uma CLI
    - Usar AttachLoadBalancerToSubnets com uma CLI
    - Usar ConfigureHealthCheck com uma CLI
    - Usar CreateAppCookieStickinessPolicy com uma CLI
    - Usar CreateLbCookieStickinessPolicy com uma CLI
    - Usar CreateLoadBalancer com uma CLI
    - Usar CreateLoadBalancerListeners com uma CLI
    - Usar CreateLoadBalancerPolicy com uma CLI
    - Usar DeleteLoadBalancer com uma CLI

- Usar DeleteLoadBalancerListeners com uma CLI
- Usar DeleteLoadBalancerPolicy com uma CLI
- Usar DeregisterInstancesFromLoadBalancer com uma CLI
- Usar DescribeInstanceHealth com uma CLI
- Usar DescribeLoadBalancerAttributes com uma CLI
- Usar DescribeLoadBalancerPolicies com uma CLI
- Usar DescribeLoadBalancerPolicyTypes com uma CLI
- Usar DescribeLoadBalancers com uma CLI
- Usar DescribeTags com uma CLI
- Usar DetachLoadBalancerFromSubnets com uma CLI
- Usar DisableAvailabilityZonesForLoadBalancer com uma CLI
- Usar EnableAvailabilityZonesForLoadBalancer com uma CLI
- Usar ModifyLoadBalancerAttributes com uma CLI
- Usar RegisterInstancesWithLoadBalancer com uma CLI
- Usar RemoveTags com uma CLI
- Usar SetLoadBalancerListenerSslCertificate com uma CLI
- Usar SetLoadBalancerPoliciesForBackendServer com uma CLI
- Usar SetLoadBalancerPoliciesOfListener com uma CLI

Exemplos básicos do Elastic Load Balancing - Versão 1 usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Elastic Load Balancing Versão 1 com os AWS SDKs.

Exemplos

- Ações para o Elastic Load Balancing - Versão 1 usando AWS SDKs
  - Usar AddTags com uma CLI
  - Usar ApplySecurityGroupsToLoadBalancer com uma CLI
  - Usar AttachLoadBalancerToSubnets com uma CLI
  - Usar ConfigureHealthCheck com uma CLI
  - Usar CreateAppCookieStickinessPolicy com uma CLI
  - Usar CreateLbCookieStickinessPolicy com uma CLI

- Usar CreateLoadBalancer com uma CLI
- Usar CreateLoadBalancerListeners com uma CLI
- Usar CreateLoadBalancerPolicy com uma CLI
- Usar DeleteLoadBalancer com uma CLI
- Usar DeleteLoadBalancerListeners com uma CLI
- Usar DeleteLoadBalancerPolicy com uma CLI
- Usar DeregisterInstancesFromLoadBalancer com uma CLI
- Usar DescribeInstanceHealth com uma CLI
- Usar DescribeLoadBalancerAttributes com uma CLI
- Usar DescribeLoadBalancerPolicies com uma CLI
- Usar DescribeLoadBalancerPolicyTypes com uma CLI
- Usar DescribeLoadBalancers com uma CLI
- Usar DescribeTags com uma CLI
- Usar DetachLoadBalancerFromSubnets com uma CLI
- Usar DisableAvailabilityZonesForLoadBalancer com uma CLI
- Usar EnableAvailabilityZonesForLoadBalancer com uma CLI
- Usar ModifyLoadBalancerAttributes com uma CLI
- Usar RegisterInstancesWithLoadBalancer com uma CLI
- Usar RemoveTags com uma CLI
- Usar SetLoadBalancerListenerSslCertificate com uma CLI
- Usar SetLoadBalancerPoliciesForBackendServer com uma CLI
- Usar SetLoadBalancerPoliciesOfListener com uma CLI

Ações para o Elastic Load Balancing - Versão 1 usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Elastic Load Balancing — Versão 1 com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Elastic Load Balancing Versão 1.

Exemplos

- Usar AddTags com uma CLI
- Usar ApplySecurityGroupsToLoadBalancer com uma CLI
- Usar AttachLoadBalancerToSubnets com uma CLI
- Usar ConfigureHealthCheck com uma CLI
- Usar CreateAppCookieStickinessPolicy com uma CLI
- Usar CreateLbCookieStickinessPolicy com uma CLI
- Usar CreateLoadBalancer com uma CLI
- Usar CreateLoadBalancerListeners com uma CLI
- Usar CreateLoadBalancerPolicy com uma CLI
- Usar DeleteLoadBalancer com uma CLI
- Usar DeleteLoadBalancerListeners com uma CLI
- Usar DeleteLoadBalancerPolicy com uma CLI
- Usar DeregisterInstancesFromLoadBalancer com uma CLI
- Usar DescribeInstanceHealth com uma CLI
- Usar DescribeLoadBalancerAttributes com uma CLI
- Usar DescribeLoadBalancerPolicies com uma CLI
- Usar DescribeLoadBalancerPolicyTypes com uma CLI
- Usar DescribeLoadBalancers com uma CLI
- Usar DescribeTags com uma CLI
- Usar DetachLoadBalancerFromSubnets com uma CLI
- Usar DisableAvailabilityZonesForLoadBalancer com uma CLI
- Usar EnableAvailabilityZonesForLoadBalancer com uma CLI
- Usar ModifyLoadBalancerAttributes com uma CLI
- Usar RegisterInstancesWithLoadBalancer com uma CLI
- Usar RemoveTags com uma CLI
- Usar SetLoadBalancerListenerSslCertificate com uma CLI
- Usar SetLoadBalancerPoliciesForBackendServer com uma CLI
- Usar SetLoadBalancerPoliciesOfListener com uma CLI

Usar **AddTags** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddTags`.

CLI

AWS CLI

Adicionar uma tag ao balanceador de carga

Este exemplo adiciona tags ao balanceador de carga especificado.

Comando:

```
aws elb add-tags --load-balancer-name my-load-balancer --
tags "Key=project,Value=lima" "Key=department,Value=digital-media"
```

- Para obter detalhes da API, consulte AddTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona as tags especificadas ao balanceador de carga especificado. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar uma tag para o parâmetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Para obter detalhes da API, consulte AddTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona as tags especificadas ao balanceador de carga especificado. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar uma tag para o parâmetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Para obter detalhes da API, consulte AddTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ApplySecurityGroupsToLoadBalancer`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ApplySecurityGroupsToLoadBalancer`.

CLI

AWS CLI

Associar um grupo de segurança a um balanceador de carga em uma VPC

Este exemplo associa um grupo de segurança ao balanceador de carga especificado em uma VPC.

Comando:

```
aws elb apply-security-groups-to-load-balancer --load-balancer-name my-load-
balancer --security-groups sg-fc448899
```

Saída:

```
{
```

```
  "SecurityGroups": [
      "sg-fc448899"
  ]
}
```

- Para obter detalhes da API, consulte ApplySecurityGroupsToLoadBalancerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo substitui o grupo de segurança atual do balanceador de carga especificado pelo grupo de segurança especificado.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Saída:

```
sg-87654321
```

Exemplo 2: para manter o grupo de segurança atual e especificar um grupo de segurança adicional, especifique os grupos de segurança existentes e os novos.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Saída:

```
sg-12345678
sg-87654321
```

- Para obter detalhes da API, consulte ApplySecurityGroupsToLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui o grupo de segurança atual do balanceador de carga especificado pelo grupo de segurança especificado.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Saída:

```
sg-87654321
```

Exemplo 2: para manter o grupo de segurança atual e especificar um grupo de segurança adicional, especifique os grupos de segurança existentes e os novos.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Saída:

```
sg-12345678
sg-87654321
```

- Para obter detalhes da API, consulte ApplySecurityGroupsToLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AttachLoadBalancerToSubnets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AttachLoadBalancerToSubnets`.

CLI

AWS CLI

Anexar sub-redes a um balanceador de carga

Este exemplo adiciona a sub-rede especificada ao conjunto de sub-redes configuradas para o balanceador de carga especificado.

Comando:

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-load-balancer --
subnets subnet-0ecac448
```

Saída:

```
{
   "Subnets": [
       "subnet-15aaab61",
       "subnet-0ecac448"
   ]
}
```

- Para obter detalhes da API, consulte AttachLoadBalancerToSubnetsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona a sub-rede especificada ao conjunto de sub-redes configuradas para o balanceador de carga especificado. A saída inclui a lista completa de sub-redes.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Saída:

```
subnet-12345678
subnet-87654321
```

- Para obter detalhes da API, consulte AttachLoadBalancerToSubnetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a sub-rede especificada ao conjunto de sub-redes configuradas para o balanceador de carga especificado. A saída inclui a lista completa de sub-redes.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Saída:

```
subnet-12345678
subnet-87654321
```

- Para obter detalhes da API, consulte AttachLoadBalancerToSubnetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ConfigureHealthCheck`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ConfigureHealthCheck`.

CLI

AWS CLI

Especificar as configurações de verificação de integridade para suas instâncias EC2 de backend

Este exemplo especifica as configurações de verificação de integridade usadas para avaliar a integridade de suas instâncias de backend do EC2.

Comando:

```
aws elb configure-health-check --load-balancer-name my-load-balancer --health-check Target=HTTP:80/
png,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

Saída:

```
{
   "HealthCheck": {
       "HealthyThreshold": 2,
       "Interval": 30,
       "Target": "HTTP:80/png",
       "Timeout": 3,
       "UnhealthyThreshold": 2
   }
}
```

- Para obter detalhes da API, consulte ConfigureHealthCheckem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo define as configurações de verificação de integridade do balanceador de carga especificado.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Saída:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- Para obter detalhes da API, consulte ConfigureHealthCheckem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo define as configurações de verificação de integridade do balanceador de carga especificado.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Saída:

```
HealthyThreshold   : 2
Interval           : 30
```

```
Target              : HTTP:80/ping
Timeout             : 3
UnhealthyThreshold : 2
```

- Para obter detalhes da API, consulte ConfigureHealthCheckem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateAppCookieStickinessPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateAppCookieStickinessPolicy`.

CLI

AWS CLI

Gerar uma política de aderência para seu balanceador de carga HTTPS

Este exemplo gera uma políticas de aderência com os tempos de vida de sessão persistente que seguem as políticas de um cookie gerado pela aplicação.

Comando:

```
aws elb create-app-cookie-stickiness-policy --load-balancer-name my-load-balancer
 --policy-name my-app-cookie-policy --cookie-name my-app-cookie
```

- Para obter detalhes da API, consulte CreateAppCookieStickinessPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma política de permanência com os tempos de vida de sessão persistente do cookie gerado pela aplicação.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-app-cookie-policy -CookieName my-app-cookie
```

- Para obter detalhes da API, consulte CreateAppCookieStickinessPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma política de permanência com os tempos de vida de sessão persistente do cookie gerado pela aplicação.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-app-cookie-policy -CookieName my-app-cookie
```

- Para obter detalhes da API, consulte CreateAppCookieStickinessPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateLbCookieStickinessPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateLbCookieStickinessPolicy`.

CLI

AWS CLI

Gerar uma política de aderência baseada em duração para seu balanceador de carga HTTPS

Este exemplo gera uma política de aderência com tempos de vida de sessão persistente controlados por um período de expiração especificado.

Comando:

```
aws elb create-lb-cookie-stickiness-policy --load-balancer-name my-load-balancer
 --policy-name my-duration-cookie-policy --cookie-expiration-period 60
```

- Para obter detalhes da API, consulte CreateLbCookieStickinessPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo gera uma política de persistência com tempos de duração de sessão persistente controlados por um período de validade especificado (em segundos).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-duration-cookie-policy -CookieExpirationPeriod 60
```

Exemplo 2: esse exemplo cria uma política de persistência em que o tempo de vida da sessão persistente é controlado pelo tempo de vida do navegador (usuário-agente).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-duration-cookie-policy
```

- Para obter detalhes da API, consulte CreateLbCookieStickinessPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo gera uma política de persistência com tempos de duração de sessão persistente controlados por um período de validade especificado (em segundos).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-duration-cookie-policy -CookieExpirationPeriod 60
```

Exemplo 2: esse exemplo cria uma política de persistência em que o tempo de vida da sessão persistente é controlado pelo tempo de vida do navegador (usuário-agente).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-duration-cookie-policy
```

- Para obter detalhes da API, consulte CreateLbCookieStickinessPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateLoadBalancer`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateLoadBalancer`.

CLI

AWS CLI

Criar um balanceador de carga HTTP

Este exemplo cria um balanceador de carga com um receptor HTTP em uma VPC.

Comando:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
 --subnets subnet-15aaab61 --security-groups sg-a61988c3
```

Saída:

```
{
    "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com"
}
```

Este exemplo cria um balanceador de carga com um ouvinte HTTP em. EC2-Classic

Comando:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
 --availability-zones us-west-2a us-west-2b
```

Saída:

```
{
    "DNSName": "my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

Criar um balanceador de carga HTTPS

Este exemplo cria um balanceador de carga com um receptor HTTPS em uma VPC.

Comando:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" "Prot
certificate/my-server-cert" --subnets subnet-15aaab61 --security-groups sg-
a61988c3
```

Saída:

```
{
    "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com"
}
```

Este exemplo cria um balanceador de carga com um ouvinte HTTPS em. EC2-Classic

Comando:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" "Prot
certificate/my-server-cert" --availability-zones us-west-2a us-west-2b
```

Saída:

```
{
    "DNSName": "my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

Criar um balanceador de carga interno

Este exemplo cria um balanceador de carga interno com um receptor HTTP em uma VPC.

Comando:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
 --scheme internal --subnets subnet-a85db0df --security-groups sg-a61988c3
```

Saída:

```
{
    "DNSName": "internal-my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

- Para obter detalhes da API, consulte CreateLoadBalancerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um balanceador de carga com um receptor HTTP em uma VPC.

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Exemplo 2: Este exemplo cria um balanceador de carga com um ouvinte HTTP em. EC2-Classic

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone
 us-west-2a` -Listener $httpListener
```

Saída:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Exemplo 3: esse exemplo cria um balanceador de carga com um receptor HTTPS.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-
west-2a -Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- Para obter detalhes da API, consulte CreateLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um balanceador de carga com um receptor HTTP em uma VPC.

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Exemplo 2: Este exemplo cria um balanceador de carga com um ouvinte HTTP em. EC2-Classic

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone
 us-west-2a` -Listener $httpListener
```

Saída:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Exemplo 3: esse exemplo cria um balanceador de carga com um receptor HTTPS.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-
west-2a -Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- Para obter detalhes da API, consulte CreateLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateLoadBalancerListeners`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateLoadBalancerListeners`.

CLI

AWS CLI

Criar receptores HTTP para um balanceador de carga

Este exemplo cria um receptor para seu balanceador de carga na porta 80 usando o protocolo HTTP.

Comando:

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
```

Criar receptores HTTPS para um balanceador de carga

Este exemplo cria um receptor para seu balanceador de carga na porta 443 usando o protocolo HTTPS.

Comando:

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --
listeners "Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80"
```

- Para obter detalhes da API, consulte CreateLoadBalancerListenersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona um receptor HTTPS ao balanceador de carga especificado.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
```

```
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
 $httpsListener
```

- Para obter detalhes da API, consulte CreateLoadBalancerListenersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona um receptor HTTPS ao balanceador de carga especificado.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
 $httpsListener
```

- Para obter detalhes da API, consulte CreateLoadBalancerListenersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateLoadBalancerPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateLoadBalancerPolicy`.

CLI

AWS CLI

Criar uma política que habilite o Protocolo de Proxy em um balanceador de carga

Este exemplo cria uma política que habilita o Protocolo de Proxy em um balanceador de carga especificado.

Comando:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-balancer --
policy-name my-ProxyProtocol-policy --policy-type-name ProxyProtocolPolicyType --
policy-attributes AttributeName=ProxyProtocol,AttributeValue=true
```

Criar uma política de negociação SSL usando a política de segurança recomendada

Este exemplo cria uma política de negociação SSL para o balanceador de carga HTTPS especificado usando a política de segurança recomendada.

Comando:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-
balancer --policy-name my-SSLNegotiation-policy --policy-type-
name SSLNegotiationPolicyType --policy-attributes AttributeName=Reference-
Security-Policy,AttributeValue=ELBSecurityPolicy-2015-03
```

Criar uma política de negociação SSL usando uma política de segurança personalizada

Este exemplo cria uma política de negociação SSL para seu balanceador de carga HTTPS usando uma política de segurança personalizada, ativando os protocolos e as cifras.

Comando:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-
balancer --policy-name my-SSLNegotiation-policy --policy-type-
name SSLNegotiationPolicyType --policy-attributes AttributeName=Protocol-
SSLv3,AttributeValue=true AttributeName=Protocol-
TLSv1.1,AttributeValue=true AttributeName=DHE-RSA-AES256-
SHA256,AttributeValue=true AttributeName=Server-Defined-Cipher-
Order,AttributeValue=true
```

Criar uma política de chave pública

Este exemplo cria uma política de chave pública.

Comando:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-balancer --
policy-name my-PublicKey-policy --policy-type-name PublicKeyPolicyType --policy-
attributes AttributeName=PublicKey,AttributeValue=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgK
+dS74kj//c6x7ROtusUaeQCTgIUkayttRDWchuqo1pHC1u
+n5xxXnBBe2ejbb2WRsKIQ5rXEeixsjFpFsojpSQKkzhVGI6mJVZBJDVKSHmswnwLBdofLhzvllpovBPTHe
+o4haAWvDBALJU0pkSI1FecPHcs2hwxf14zHoXy1e2k36A64nXW43wtfx5qcVSIxtCEOjnYRg7RPvybaGfQ
+v6Iaxb/+7J5kEvZhTFQId+bSiJImF1FSUT1W1xwzBZPUbcUkkXDj45vC2s3Z8E
+Lk7a3uZhvsQHLZnrfuWjBWGWvZ/MhZYgEXAMPLE
```

Criar uma política de autenticação de servidor de backend

Este exemplo cria uma política de autenticação de servidor de backend que permite a autenticação em sua instância de backend usando uma política de chave pública.

Comando:

```
aws elb create-load-balancer-policy --load-balancer-name my-
load-balancer --policy-name my-authentication-policy --policy-
type-name BackendServerAuthenticationPolicyType --policy-
attributes AttributeName=PublicKeyPolicyName,AttributeValue=my-PublicKey-policy
```

- Para obter detalhes da API, consulte CreateLoadBalancerPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma política de protocolo de proxy para um balanceador de carga especificado.

```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -
Property @{
         AttributeName="ProxyProtocol"
         AttributeValue="True"
    }
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
 $attribute
```

- Para obter detalhes da API, consulte CreateLoadBalancerPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma política de protocolo de proxy para um balanceador de carga especificado.

```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -
Property @{
```

```
        AttributeName="ProxyProtocol"
        AttributeValue="True"
    }
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
 $attribute
```

- Para obter detalhes da API, consulte CreateLoadBalancerPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteLoadBalancer`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteLoadBalancer`.

CLI

AWS CLI

Como excluir um balanceador de carga

Este exemplo exclui o balanceador de carga especificado.

Comando:

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

- Para obter detalhes da API, consulte DeleteLoadBalancerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancer (DeleteLoadBalancer)" on Target "my-
load-balancer".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancer (DeleteLoadBalancer)" on Target "my-
load-balancer".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteLoadBalancerListeners`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteLoadBalancerListeners`.

CLI

AWS CLI

Excluir um receptor do balanceador de carga

Este exemplo exclui o receptor da porta especificada do balanceador de carga especificado.

Comando:

```
aws elb delete-load-balancer-listeners --load-balancer-name my-load-balancer --
load-balancer-ports 80
```

- Para obter detalhes da API, consulte DeleteLoadBalancerListenersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o receptor da porta 80 do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 80
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerListener
 (DeleteLoadBalancerListeners)" on Target "80".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteLoadBalancerListenersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o receptor da porta 80 do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 80
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerListener
 (DeleteLoadBalancerListeners)" on Target "80".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteLoadBalancerListenersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteLoadBalancerPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteLoadBalancerPolicy`.

CLI

AWS CLI

Excluir uma política do seu balanceador de carga

Este exemplo exclui a política especificada do balanceador de carga especificado. A política não deve ser habilitada em qualquer receptor.

Comando:

```
aws elb delete-load-balancer-policy --load-balancer-name my-load-balancer --
policy-name my-duration-cookie-policy
```

- Para obter detalhes da API, consulte DeleteLoadBalancerPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a política especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerPolicy (DeleteLoadBalancerPolicy)" on
 Target "my-duration-cookie-policy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteLoadBalancerPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a política especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerPolicy (DeleteLoadBalancerPolicy)" on
 Target "my-duration-cookie-policy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes da API, consulte DeleteLoadBalancerPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeregisterInstancesFromLoadBalancer`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterInstancesFromLoadBalancer`.

CLI

AWS CLI

Cancelar o registro de instâncias do balanceador de carga

Este exemplo cancela o registro das instâncias especificadas do balanceador de carga especificado.

Comando:

```
aws elb deregister-instances-from-load-balancer --load-balancer-name my-load-balancer --instances i-d6f6fae3
```

Saída:

```
{
    "Instances": [
        {
            "InstanceId": "i-207d9717"
        },
        {
            "InstanceId": "i-afefb49b"
        }
    ]
}
```

- Para obter detalhes da API, consulte DeregisterInstancesFromLoadBalancerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a instância do EC2 especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBInstanceFromLoadBalancer
 (DeregisterInstancesFromLoadBalancer)" on Target
"Amazon.ElasticLoadBalancing.Model.Instance".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

InstanceId
----------
i-87654321
```

- Para obter detalhes da API, consulte DeregisterInstancesFromLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a instância do EC2 especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBInstanceFromLoadBalancer
 (DeregisterInstancesFromLoadBalancer)" on Target
"Amazon.ElasticLoadBalancing.Model.Instance".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

InstanceId
----------
i-87654321
```

- Para obter detalhes da API, consulte DeregisterInstancesFromLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeInstanceHealth`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstanceHealth`.

CLI

AWS CLI

Descrever a integridade das instâncias de um balanceador de carga

Este exemplo descreve a integridade das instâncias do balanceador de carga especificado.

Comando:

```
aws elb describe-instance-health --load-balancer-name my-load-balancer
```

Saída:

```
{
  "InstanceStates": [
      {
          "InstanceId": "i-207d9717",
          "ReasonCode": "N/A",
          "State": "InService",
          "Description": "N/A"
      },
      {
          "InstanceId": "i-afefb49b",
          "ReasonCode": "N/A",
          "State": "InService",
          "Description": "N/A"
      }
  ]
}
```

Descrever a integridade de uma instância para um balanceador de carga

Este exemplo descreve a integridade da instância especificada para o balanceador de carga especificado.

Comando:

```
aws elb describe-instance-health --load-balancer-name my-load-balancer --
instances i-7299c809
```

Veja a seguir um exemplo de resposta para uma instância que está sendo registrada.

Saída:

```
{
  "InstanceStates": [
      {
          "InstanceId": "i-7299c809",
          "ReasonCode": "ELB",
          "State": "OutOfService",
          "Description": "Instance registration is still in progress."
    }
  ]
}
```

Veja a seguir um exemplo de resposta para uma instância que não está íntegra.

Saída:

```
{
  "InstanceStates": [
      {
          "InstanceId": "i-7299c809",
          "ReasonCode": "Instance",
          "State": "OutOfService",
          "Description": "Instance has failed at least the UnhealthyThreshold
 number of health checks consecutively."
      }
  ]
}
```

- Para obter detalhes da API, consulte DescribeInstanceHealthem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: descreve o estado das instâncias registradas com o balanceador de carga especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Saída:

```
Description                   InstanceId                    ReasonCode
         State
-----------                   ----------                    ----------
         -----
N/A                           i-87654321                    N/A
         InService
Instance has failed at lea... i-12345678                    Instance
         OutOfService
```

Exemplo 2: descreve o estado da instância especificada registrada com o balanceador de carga especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Exemplo 3: esse exemplo exibe a descrição completa do estado da instância especificada.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Saída:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- Para obter detalhes da API, consulte DescribeInstanceHealthem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: descreve o estado das instâncias registradas com o balanceador de carga especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Saída:

```
Description                   InstanceId                    ReasonCode
         State
-----------                   ----------                    ----------
         -----
N/A                           i-87654321                    N/A
         InService
Instance has failed at lea... i-12345678                    Instance
         OutOfService
```

Exemplo 2: descreve o estado da instância especificada registrada com o balanceador de carga especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Exemplo 3: esse exemplo exibe a descrição completa do estado da instância especificada.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Saída:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- Para obter detalhes da API, consulte DescribeInstanceHealthem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLoadBalancerAttributes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLoadBalancerAttributes`.

CLI

AWS CLI

Descrever os atributos de um balanceador de carga

Este exemplo descreve os atributos do balanceador de carga especificado.

Comando:

```
aws elb describe-load-balancer-attributes --load-balancer-name my-load-balancer
```

Saída:

```
{
  "LoadBalancerAttributes": {
      "ConnectionDraining": {
          "Enabled": false,
          "Timeout": 300
      },
      "CrossZoneLoadBalancing": {
          "Enabled": true
      },
      "ConnectionSettings": {
          "IdleTimeout": 30
      },
      "AccessLog": {
          "Enabled": false
    }
  }
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancerAttributesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve os atributos para o balanceador de carga especificado.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Saída:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Para obter detalhes da API, consulte DescribeLoadBalancerAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve os atributos para o balanceador de carga especificado.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Saída:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Para obter detalhes da API, consulte DescribeLoadBalancerAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLoadBalancerPolicies`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLoadBalancerPolicies`.

CLI

AWS CLI

Descrever todas as políticas associadas a um balanceador de carga

Este exemplo descreve todas as políticas associadas ao balanceador de carga especificado.

Comando:

```
aws elb describe-load-balancer-policies --load-balancer-name my-load-balancer
```

Saída:

```
{
  "PolicyDescriptions": [
    {
      "PolicyAttributeDescriptions": [
        {
          "AttributeName": "ProxyProtocol",
          "AttributeValue": "true"
        }
      ],
      "PolicyName": "my-ProxyProtocol-policy",
      "PolicyTypeName": "ProxyProtocolPolicyType"
    },
    {
        "PolicyAttributeDescriptions": [
            {
                "AttributeName": "CookieName",
                "AttributeValue": "my-app-cookie"
            }
        ],
        "PolicyName": "my-app-cookie-policy",
        "PolicyTypeName": "AppCookieStickinessPolicyType"
    },
    {
      "PolicyAttributeDescriptions": [
        {
          "AttributeName": "CookieExpirationPeriod",
          "AttributeValue": "60"
        }
      ],
      "PolicyName": "my-duration-cookie-policy",
      "PolicyTypeName": "LBCookieStickinessPolicyType"
    },
    .
    .
    .
  ]
}
```

Descrever uma política específica associada a um balanceador de carga

Este exemplo descreve a política especificada associada ao balanceador de carga especificado.

Comando:

```
aws elb describe-load-balancer-policies --load-balancer-name my-load-balancer --
policy-name my-authentication-policy
```

Saída:

```
{
  "PolicyDescriptions": [
      {
          "PolicyAttributeDescriptions": [
              {
                  "AttributeName": "PublicKeyPolicyName",
                  "AttributeValue": "my-PublicKey-policy"
              }
          ],
          "PolicyName": "my-authentication-policy",
          "PolicyTypeName": "BackendServerAuthenticationPolicyType"
      }
  ]
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancerPoliciesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve as políticas associadas ao balanceador de carga especificado.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Saída:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Exemplo 2: esse exemplo descreve os atributos da política especificada.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Saída:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Exemplo 3: esse exemplo descreve as políticas predefinidas, incluindo os exemplos de políticas. Os nomes das políticas de amostra têm o prefixo ELBSample-.

```
Get-ELBLoadBalancerPolicy
```

Saída:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-05
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
 SSLNegotiationPolicyType
```

```
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
 SSLNegotiationPolicyType
```

- Para obter detalhes da API, consulte DescribeLoadBalancerPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as políticas associadas ao balanceador de carga especificado.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Saída:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Exemplo 2: esse exemplo descreve os atributos da política especificada.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Saída:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Exemplo 3: esse exemplo descreve as políticas predefinidas, incluindo os exemplos de políticas. Os nomes das políticas de amostra têm o prefixo ELBSample-.

```
Get-ELBLoadBalancerPolicy
```

Saída:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-05
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
 SSLNegotiationPolicyType
```

- Para obter detalhes da API, consulte DescribeLoadBalancerPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLoadBalancerPolicyTypes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLoadBalancerPolicyTypes`.

CLI

AWS CLI

Descrever os tipos de políticas de balanceador de carga definidos pelo Elastic Load Balancing

Este exemplo descreve os tipos de política do balanceador de carga que você pode usar para criar configurações de política para seu balanceador de carga.

Comando:

```
aws elb describe-load-balancer-policy-types
```

Saída:

```
{
  "PolicyTypeDescriptions": [
      {
          "PolicyAttributeTypeDescriptions": [
              {
                  "Cardinality": "ONE",
                  "AttributeName": "ProxyProtocol",
                  "AttributeType": "Boolean"
              }
          ],
          "PolicyTypeName": "ProxyProtocolPolicyType",
          "Description": "Policy that controls whether to include the IP address
 and port of the originating request for TCP messages. This policy operates on
 TCP/SSL listeners only"
      },
      {
          "PolicyAttributeTypeDescriptions": [
              {
                  "Cardinality": "ONE",
                  "AttributeName": "PublicKey",
                  "AttributeType": "String"
              }
          ],
          "PolicyTypeName": "PublicKeyPolicyType",
          "Description": "Policy containing a list of public keys to
 accept when authenticating the back-end server(s). This policy cannot be
 applied directly to back-end servers or listeners but must be part of a
 BackendServerAuthenticationPolicyType."
      },
      {
          "PolicyAttributeTypeDescriptions": [
              {
                  "Cardinality": "ONE",
                  "AttributeName": "CookieName",
                  "AttributeType": "String"
              }
          ],
          "PolicyTypeName": "AppCookieStickinessPolicyType",
```

```
        "Description": "Stickiness policy with session lifetimes controlled by
the lifetime of the application-generated cookie. This policy can be associated
only with HTTP/HTTPS listeners."
    },
    {
        "PolicyAttributeTypeDescriptions": [
            {
                "Cardinality": "ZERO_OR_ONE",
                "AttributeName": "CookieExpirationPeriod",
                "AttributeType": "Long"
            }
        ],
        "PolicyTypeName": "LBCookieStickinessPolicyType",
        "Description": "Stickiness policy with session lifetimes controlled by
the browser (user-agent) or a specified expiration period. This policy can be
associated only with HTTP/HTTPS listeners."
    },
    {
        "PolicyAttributeTypeDescriptions": [
            .
            .
            .
        ],
        "PolicyTypeName": "SSLNegotiationPolicyType",
        "Description": "Listener policy that defines the ciphers and protocols
that will be accepted by the load balancer. This policy can be associated only
with HTTPS/SSL listeners."
    },
    {
        "PolicyAttributeTypeDescriptions": [
            {
                "Cardinality": "ONE_OR_MORE",
                "AttributeName": "PublicKeyPolicyName",
                "AttributeType": "PolicyName"
            }
        ],
        "PolicyTypeName": "BackendServerAuthenticationPolicyType",
        "Description": "Policy that controls authentication to back-end
server(s) and contains one or more policies, such as an instance of a
PublicKeyPolicyType. This policy can be associated only with back-end servers
that are using HTTPS/SSL."
    }
 ]
```

```
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancerPolicyTypesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém os tipos de políticas compatíveis com o Elastic Load Balancing.

```
Get-ELBLoadBalancerPolicyType
```

Saída:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Stickiness policy with session lifet... {CookieExpirationPeriod}
 LBCookieStickinessPolicyType
Policy that controls authentication ... {PublicKeyPolicyName}
 BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
 SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
 PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
 AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Exemplo 2: esse exemplo descreve o tipo de política especificado.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Saída:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
```

```
-----------                              -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Exemplo 3: esse exemplo exibe a descrição completa do tipo de política especificado.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Saída:

```
Policy that controls whether to include the IP address and port of the
 originating request for TCP messages.
This policy operates on TCP/SSL listeners only
```

- Para obter detalhes da API, consulte DescribeLoadBalancerPolicyTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os tipos de políticas compatíveis com o Elastic Load Balancing.

```
Get-ELBLoadBalancerPolicyType
```

Saída:

```
Description                              PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                              -------------------------------
 --------------
Stickiness policy with session lifet... {CookieExpirationPeriod}
 LBCookieStickinessPolicyType
Policy that controls authentication ... {PublicKeyPolicyName}
 BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
 SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
 PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
 AppCookieStickinessPolicyType
```

```
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Exemplo 2: esse exemplo descreve o tipo de política especificado.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Saída:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Exemplo 3: esse exemplo exibe a descrição completa do tipo de política especificado.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Saída:

```
Policy that controls whether to include the IP address and port of the
 originating request for TCP messages.
This policy operates on TCP/SSL listeners only
```

- Para obter detalhes da API, consulte DescribeLoadBalancerPolicyTypesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLoadBalancers`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLoadBalancers`.

CLI

AWS CLI

Descrever os balanceadores de carga

Este exemplo descreve todos os balanceadores de carga.

Comando:

```
aws elb describe-load-balancers
```

Descrever um dos balanceadores de carga

Este exemplo descreve o balanceador de carga especificado.

Comando:

```
aws elb describe-load-balancers --load-balancer-name my-load-balancer
```

Veja a seguir um exemplo de resposta para um balanceador de carga HTTPS em uma VPC.

Saída:

```
{
  "LoadBalancerDescriptions": [
    {
      "Subnets": [
          "subnet-15aaab61"
      ],
      "CanonicalHostedZoneNameID": "Z3DZXE0EXAMPLE",
      "CanonicalHostedZoneName": "my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com",
      "ListenerDescriptions": [
          {
              "Listener": {
                  "InstancePort": 80,
                  "LoadBalancerPort": 80,
                  "Protocol": "HTTP",
                  "InstanceProtocol": "HTTP"
              },
              "PolicyNames": []
          },
          {
              "Listener": {
                  "InstancePort": 443,
                  "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-cert",
                  "LoadBalancerPort": 443,
                  "Protocol": "HTTPS",
                  "InstanceProtocol": "HTTPS"
```

```
            },
            "PolicyNames": [
                "ELBSecurityPolicy-2015-03"
            ]
        }
    ],
    "HealthCheck": {
        "HealthyThreshold": 2,
        "Interval": 30,
        "Target": "HTTP:80/png",
        "Timeout": 3,
        "UnhealthyThreshold": 2
    },
    "VPCId": "vpc-a01106c2",
    "BackendServerDescriptions": [
        {
            "InstancePort": 80,
            "PolicyNames": [
                "my-ProxyProtocol-policy"
            ]
        }
    ],
    "Instances": [
        {
            "InstanceId": "i-207d9717"
        },
        {
            "InstanceId": "i-afefb49b"
        }
    ],
    "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com",
    "SecurityGroups": [
        "sg-a61988c3"
    ],
    "Policies": {
        "LBCookieStickinessPolicies": [
            {
                "PolicyName": "my-duration-cookie-policy",
                "CookieExpirationPeriod": 60
            }
        ],
        "AppCookieStickinessPolicies": [],
        "OtherPolicies": [
            "my-PublicKey-policy",
```

```
                "my-authentication-policy",
                "my-SSLNegotiation-policy",
                "my-ProxyProtocol-policy",
                "ELBSecurityPolicy-2015-03"
            ]
        },
        "LoadBalancerName": "my-load-balancer",
        "CreatedTime": "2015-03-19T03:24:02.650Z",
        "AvailabilityZones": [
            "us-west-2a"
        ],
        "Scheme": "internet-facing",
        "SourceSecurityGroup": {
            "OwnerAlias": "123456789012",
            "GroupName": "my-elb-sg"
        }
      }
    ]
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista os nomes de seus balanceadores de carga.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Saída:

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Exemplo 2: esse exemplo descreve o balanceador de carga especificado.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Saída:

```
AvailabilityZones         : {us-west-2a, us-west-2b}
BackendServerDescriptions :
 {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName   : my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime               : 4/11/2015 12:12:45 PM
DNSName                   : my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com
HealthCheck               : Amazon.ElasticLoadBalancing.Model.HealthCheck
Instances                 : {i-207d9717, i-afefb49b}
ListenerDescriptions      :
 {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
LoadBalancerName          : my-load-balancer
Policies                  : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                    : internet-facing
SecurityGroups            : {sg-a61988c3}
SourceSecurityGroup       : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                   : {subnet-15aaab61}
VPCId                     : vpc-a01106c2
```

Exemplo 3: Este exemplo descreve todos os seus balanceadores de carga na AWS região atual.

```
Get-ELBLoadBalancer
```

Exemplo 4: esse exemplo descreve todos os balanceadores de carga em todas as Regiões da AWS disponíveis.

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Para obter detalhes da API, consulte DescribeLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os nomes de seus balanceadores de carga.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Saída:

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Exemplo 2: esse exemplo descreve o balanceador de carga especificado.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Saída:

```
AvailabilityZones         : {us-west-2a, us-west-2b}
BackendServerDescriptions :
 {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName   : my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime               : 4/11/2015 12:12:45 PM
DNSName                   : my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com
HealthCheck               : Amazon.ElasticLoadBalancing.Model.HealthCheck
Instances                 : {i-207d9717, i-afefb49b}
ListenerDescriptions      :
 {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
LoadBalancerName          : my-load-balancer
Policies                  : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                    : internet-facing
SecurityGroups            : {sg-a61988c3}
SourceSecurityGroup       : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                   : {subnet-15aaab61}
VPCId                     : vpc-a01106c2
```

Exemplo 3: Este exemplo descreve todos os seus balanceadores de carga na AWS região atual.

```
Get-ELBLoadBalancer
```

Exemplo 4: esse exemplo descreve todos os balanceadores de carga em todas as Regiões da AWS disponíveis.

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Para obter detalhes da API, consulte DescribeLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTags`.

CLI

AWS CLI

Descrever as tags atribuídas a um balanceador de carga

Este exemplo descreve as tags atribuídas ao balanceador de carga especificado.

Comando:

```
aws elb describe-tags --load-balancer-name my-load-balancer
```

Saída:

```
{
  "TagDescriptions": [
      {
          "Tags": [
              {
                  "Value": "lima",
                  "Key": "project"
              },
              {
                  "Value": "digital-media",
                  "Key": "department"
              }
          ],
          "LoadBalancerName": "my-load-balancer"
      }
  ]
```

```
}
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista as tags para os balanceadores de carga especificados.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Saída:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

Exemplo 2: esse exemplo descreve as tags para o balanceador de carga especificado.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Saída:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as tags para os balanceadores de carga especificados.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Saída:

```
LoadBalancerName            Tags
----------------            ----
my-load-balancer            {project, department}
my-internal-load-balancer   {project, department}
```

Exemplo 2: esse exemplo descreve as tags para o balanceador de carga especificado.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Saída:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DetachLoadBalancerFromSubnets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DetachLoadBalancerFromSubnets`.

CLI

AWS CLI

Separar os balanceadores de carga das sub-redes

Este exemplo separa o balanceador de carga especificado da sub-rede especificada.

Comando:

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-load-balancer
 --subnets subnet-0ecac448
```

Saída:

```
{
```

```
    "Subnets": [
        "subnet-15aaab61"
    ]
}
```

- Para obter detalhes da API, consulte DetachLoadBalancerFromSubnetsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona a sub-rede especificada ao conjunto de sub-redes configuradas para o balanceador de carga especificado. A saída inclui as sub-redes restantes.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Saída:

```
subnet-87654321
```

- Para obter detalhes da API, consulte DetachLoadBalancerFromSubnetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a sub-rede especificada ao conjunto de sub-redes configuradas para o balanceador de carga especificado. A saída inclui as sub-redes restantes.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Saída:

```
subnet-87654321
```

- Para obter detalhes da API, consulte DetachLoadBalancerFromSubnetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DisableAvailabilityZonesForLoadBalancer`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DisableAvailabilityZonesForLoadBalancer`.

CLI

AWS CLI

Desativar as Zonas de Disponibilidade para um balanceador de carga

Este exemplo remove as Zonas de Disponibilidade especificadas do conjunto de Zonas de Disponibilidade do balanceador de carga especificado.

Comando:

```
aws elb disable-availability-zones-for-load-balancer --load-balancer-name my-load-balancer --availability-zones us-west-2a
```

Saída:

```
{
    "AvailabilityZones": [
        "us-west-2b"
    ]
}
```

- Para obter detalhes da API, consulte DisableAvailabilityZonesForLoadBalancerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona a Zona de Disponibilidade especificada ao balanceador de carga especificado. A saída inclui as zonas de disponibilidade restantes.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-west-2a
```

Saída:

```
us-west-2b
```

- Para obter detalhes da API, consulte DisableAvailabilityZonesForLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a Zona de Disponibilidade especificada ao balanceador de carga especificado. A saída inclui as zonas de disponibilidade restantes.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Saída:

```
us-west-2b
```

- Para obter detalhes da API, consulte DisableAvailabilityZonesForLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`EnableAvailabilityZonesForLoadBalancer`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EnableAvailabilityZonesForLoadBalancer`.

CLI

AWS CLI

Habilitar Zonas de Disponibilidade para um balanceador de carga

Este exemplo adiciona a Zona de Disponibilidade especificada ao balanceador de carga especificado.

Comando:

```
aws elb enable-availability-zones-for-load-balancer --load-balancer-name my-load-
balancer --availability-zones us-west-2b
```

Saída:

```
{
    "AvailabilityZones": [
        "us-west-2a",
        "us-west-2b"
    ]
}
```

- Para obter detalhes da API, consulte EnableAvailabilityZonesForLoadBalancerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona a Zona de Disponibilidade especificada ao balanceador de carga especificado. O resultado inclui a lista completa de zonas de disponibilidade.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Saída:

```
us-west-2a
us-west-2b
```

- Para obter detalhes da API, consulte EnableAvailabilityZonesForLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a Zona de Disponibilidade especificada ao balanceador de carga especificado. O resultado inclui a lista completa de zonas de disponibilidade.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Saída:

```
us-west-2a
us-west-2b
```

- Para obter detalhes da API, consulte EnableAvailabilityZonesForLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyLoadBalancerAttributes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyLoadBalancerAttributes`.

CLI

AWS CLI

Modificar os atributos de um balanceador de carga

Este exemplo modifica o atributo `CrossZoneLoadBalancing` do balanceador de carga especificado.

Comando:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer --
load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":true}}"
```

Saída:

```
{
    "LoadBalancerAttributes": {
        "CrossZoneLoadBalancing": {
            "Enabled": true
        }
    },
    "LoadBalancerName": "my-load-balancer"
}
```

Este exemplo modifica o atributo `ConnectionDraining` do balanceador de carga especificado.

Comando:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer
 --load-balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":true,\"Timeout
\":300}}"
```

Saída:

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": true,
            "Timeout": 300
        }
    },
    "LoadBalancerName": "my-load-balancer"
}
```

- Para obter detalhes da API, consulte ModifyLoadBalancerAttributesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo habilita o balanceamento de carga entre zonas para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Exemplo 2: esse exemplo desativa a drenagem da conexão para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Exemplo 3: esse exemplo habilita os logs de acesso para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Para obter detalhes da API, consulte ModifyLoadBalancerAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o balanceamento de carga entre zonas para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Exemplo 2: esse exemplo desativa a drenagem da conexão para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Exemplo 3: esse exemplo habilita os logs de acesso para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Para obter detalhes da API, consulte ModifyLoadBalancerAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RegisterInstancesWithLoadBalancer`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterInstancesWithLoadBalancer`.

CLI

AWS CLI

Registrar instâncias com um balanceador de carga

Este exemplo registra a instância especificada com o balanceador de carga especificado.

Comando:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-load-
balancer --instances i-d6f6fae3
```

Saída:

```
{
   "Instances": [
       {
           "InstanceId": "i-d6f6fae3"
       },
       {
           "InstanceId": "i-207d9717"
       },
       {
           "InstanceId": "i-afefb49b"
       }
   ]
}
```

- Para obter detalhes da API, consulte RegisterInstancesWithLoadBalancerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra a instância do EC2 especificada com o balanceador de carga especificado.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Saída:

```
InstanceId
----------
i-12345678
i-87654321
```

- Para obter detalhes da API, consulte RegisterInstancesWithLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra a instância do EC2 especificada com o balanceador de carga especificado.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Saída:

```
InstanceId
----------
i-12345678
i-87654321
```

- Para obter detalhes da API, consulte RegisterInstancesWithLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RemoveTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemoveTags`.

CLI

AWS CLI

Remover tags de um balanceador de carga

Este exemplo remove uma tag do balanceador de carga especificado.

Comando:

```
aws elb remove-tags --load-balancer-name my-load-balancer --tags project
```

- Para obter detalhes da API, consulte RemoveTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a tag especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELBResourceTag (RemoveTags)" on target
 "Amazon.ElasticLoadBalancing.Model.TagKeyOnly".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Com o Powershell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Para obter detalhes da API, consulte RemoveTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELBResourceTag (RemoveTags)" on target
 "Amazon.ElasticLoadBalancing.Model.TagKeyOnly".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Com o Powershell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Para obter detalhes da API, consulte RemoveTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SetLoadBalancerListenerSslCertificate`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetLoadBalancerListenerSslCertificate`.

CLI

AWS CLI

Atualizar o certificado SSL para um balanceador de carga HTTPS

Este exemplo substitui o certificado SSL existente para o balanceador de carga HTTPS especificado.

Comando:

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-
name my-load-balancer --load-balancer-port 443 --ssl-certificate-
id arn:aws:iam::123456789012:server-certificate/new-server-cert
```

- Para obter detalhes da API, consulte SetLoadBalancerListenerSslCertificateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo substitui o certificado que encerra as conexões SSL para o receptor especificado.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-
cert"
```

- Para obter detalhes da API, consulte SetLoadBalancerListenerSslCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui o certificado que encerra as conexões SSL para o receptor especificado.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-
cert"
```

- Para obter detalhes da API, consulte SetLoadBalancerListenerSslCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SetLoadBalancerPoliciesForBackendServer`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetLoadBalancerPoliciesForBackendServer`.

CLI

AWS CLI

Substituir as políticas associadas a uma porta para uma instância de backend

Este exemplo substitui as políticas atualmente associadas à porta especificada.

Comando:

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-load-balancer --instance-port 80 --policy-names my-ProxyProtocol-policy
```

Remover todas as políticas atualmente associadas a uma porta na sua instância de backend

Este exemplo remove todas as políticas associadas à porta especificada.

Comando:

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-load-balancer --instance-port 80 --policy-names []
```

Para confirmar que as políticas foram removidas, use o comando `describe-load-balancer-policies`.

- Para obter detalhes da API, consulte SetLoadBalancerPoliciesForBackendServerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo substitui as políticas da porta especificada pela política especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Exemplo 2: esse exemplo remove todas as políticas associadas à porta especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -InstancePort 80
```

- Para obter detalhes da API, consulte SetLoadBalancerPoliciesForBackendServerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui as políticas da porta especificada pela política especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Exemplo 2: esse exemplo remove todas as políticas associadas à porta especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- Para obter detalhes da API, consulte SetLoadBalancerPoliciesForBackendServerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SetLoadBalancerPoliciesOfListener`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetLoadBalancerPoliciesOfListener`.

CLI

AWS CLI

Substituir as políticas associadas a um receptor

Este exemplo substitui as políticas atualmente associadas ao receptor especificado.

Comando:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-load-
balancer --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

Remover todas as políticas associadas ao receptor

Este exemplo remove todas as políticas atualmente associadas ao receptor especificado.

Comando:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-load-
balancer --load-balancer-port 443 --policy-names []
```

Para confirmar se as políticas foram removidas do balanceador de carga, use o comando `describe-load-balancer-policies`.

- Para obter detalhes da API, consulte SetLoadBalancerPoliciesOfListenerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo substitui as políticas do receptor especificado pela política especificada.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Exemplo 2: esse exemplo remove todas as políticas associadas ao receptor especificado.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- Para obter detalhes da API, consulte SetLoadBalancerPoliciesOfListenerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui as políticas do receptor especificado pela política especificada.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Exemplo 2: esse exemplo remove todas as políticas associadas ao receptor especificado.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- Para obter detalhes da API, consulte SetLoadBalancerPoliciesOfListenerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para o Elastic Load Balancing - Versão 2 usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Elastic Load Balancing - Versão 2 com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Elastic Load Balancing Versão 2: mais informações sobre o Elastic Load Balancing Versão 2.
- Referência de API do Elastic Load Balancing Versão 2: detalhes sobre todas as ações disponíveis do Elastic Load Balancing Versão 2.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Elastic Load Balancing - Versão 2 usando AWS SDKs
  - Olá, Elastic Load Balancing
  - Ações para o Elastic Load Balancing - Versão 2 usando AWS SDKs
    - Usar AddListenerCertificates com uma CLI
    - Usar AddTags com uma CLI
    - Use CreateListenercom um AWS SDK ou CLI
    - Use CreateLoadBalancercom um AWS SDK ou CLI
    - Usar CreateRule com uma CLI
    - Use CreateTargetGroupcom um AWS SDK ou CLI
    - Usar DeleteListener com uma CLI
    - Use DeleteLoadBalancercom um AWS SDK ou CLI
    - Usar DeleteRule com uma CLI
    - Use DeleteTargetGroupcom um AWS SDK ou CLI
    - Usar DeregisterTargets com uma CLI
    - Usar DescribeAccountLimits com uma CLI

- Usar DescribeListenerCertificates com uma CLI
- Usar DescribeListeners com uma CLI
- Usar DescribeLoadBalancerAttributes com uma CLI
- Use DescribeLoadBalancerscom um AWS SDK ou CLI
- Usar DescribeRules com uma CLI
- Usar DescribeSslPolicies com uma CLI
- Usar DescribeTags com uma CLI
- Usar DescribeTargetGroupAttributes com uma CLI
- Use DescribeTargetGroupscom um AWS SDK ou CLI
- Use DescribeTargetHealthcom um AWS SDK ou CLI
- Usar ModifyListener com uma CLI
- Usar ModifyLoadBalancerAttributes com uma CLI
- Usar ModifyRule com uma CLI
- Usar ModifyTargetGroup com uma CLI
- Usar ModifyTargetGroupAttributes com uma CLI
- Usar RegisterTargets com uma CLI
- Usar RemoveListenerCertificates com uma CLI
- Usar RemoveTags com uma CLI
- Usar SetIpAddressType com uma CLI
- Usar SetRulePriorities com uma CLI
- Usar SetSecurityGroups com uma CLI
- Usar SetSubnets com uma CLI

- Cenários para o Elastic Load Balancing - Versão 2 usando AWS SDKs
  - Crie e gerencie um serviço resiliente usando um AWS SDK
  - Introdução ao balanceamento de carga

Exemplos básicos do Elastic Load Balancing - Versão 2 usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Elastic Load Balancing Versão 2 com os AWS SDKs.

Exemplos

- Olá, Elastic Load Balancing
- Ações para o Elastic Load Balancing - Versão 2 usando AWS SDKs
  - Usar AddListenerCertificates com uma CLI
  - Usar AddTags com uma CLI
  - Use CreateListenercom um AWS SDK ou CLI
  - Use CreateLoadBalancercom um AWS SDK ou CLI
  - Usar CreateRule com uma CLI
  - Use CreateTargetGroupcom um AWS SDK ou CLI
  - Usar DeleteListener com uma CLI
  - Use DeleteLoadBalancercom um AWS SDK ou CLI
  - Usar DeleteRule com uma CLI
  - Use DeleteTargetGroupcom um AWS SDK ou CLI
  - Usar DeregisterTargets com uma CLI
  - Usar DescribeAccountLimits com uma CLI
  - Usar DescribeListenerCertificates com uma CLI
  - Usar DescribeListeners com uma CLI
  - Usar DescribeLoadBalancerAttributes com uma CLI
  - Use DescribeLoadBalancerscom um AWS SDK ou CLI
  - Usar DescribeRules com uma CLI
  - Usar DescribeSslPolicies com uma CLI
  - Usar DescribeTags com uma CLI
  - Usar DescribeTargetGroupAttributes com uma CLI
  - Use DescribeTargetGroupscom um AWS SDK ou CLI
  - Use DescribeTargetHealthcom um AWS SDK ou CLI
  - Usar ModifyListener com uma CLI
  - Usar ModifyLoadBalancerAttributes com uma CLI
  - Usar ModifyRule com uma CLI
  - Usar ModifyTargetGroup com uma CLI
  - Usar ModifyTargetGroupAttributes com uma CLI

  - Usar RegisterTargets com uma CLI

- Usar RemoveListenerCertificates com uma CLI
- Usar RemoveTags com uma CLI
- Usar SetIpAddressType com uma CLI
- Usar SetRulePriorities com uma CLI
- Usar SetSecurityGroups com uma CLI
- Usar SetSubnets com uma CLI

Olá, Elastic Load Balancing

Os exemplos de código a seguir mostram como começar a usar o Elastic Load Balancing .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class HelloLoadBalancer {

        public static void main(String[] args) {
                ElasticLoadBalancingV2Client loadBalancingV2Client =
 ElasticLoadBalancingV2Client.builder()
                                .region(Region.US_EAST_1)
                                .build();

                DescribeLoadBalancersResponse loadBalancersResponse =
 loadBalancingV2Client
                                .describeLoadBalancers(r -> r.pageSize(10));
                List<LoadBalancer> loadBalancerList =
 loadBalancersResponse.loadBalancers();
                for (LoadBalancer lb : loadBalancerList)
                        System.out.println("Load Balancer DNS name = " +
 lb.dnsName());
        }
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancersa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  ElasticLoadBalancingV2Client,
  DescribeLoadBalancersCommand,
} from "@aws-sdk/client-elastic-load-balancing-v2";

export async function main() {
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
    new DescribeLoadBalancersCommand({}),
  );
  const loadBalancersList = LoadBalancers.map(
    (lb) => `• ${lb.LoadBalancerName}: ${lb.DNSName}`,
  ).join("\n");
  console.log(
    "Hello, Elastic Load Balancing! Let's list some of your load balancers:\n",
    loadBalancersList,
  );
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancersa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_elbv2(elbv2_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Elastic Load Balancing V2
 client and list
    up to ten of the load balancers for your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param elbv2_client: A Boto3 Elastic Load Balancing V2 client object.
    """
    print("Hello, Elastic Load Balancing! Let's list some of your load
 balancers:")
    load_balancers = elbv2_client.describe_load_balancers(PageSize=10).get(
        "LoadBalancers", []
    )
    if load_balancers:
        for lb in load_balancers:
            print(f"\t{lb['LoadBalancerName']}: {lb['DNSName']}")
    else:
        print("Your account doesn't have any load balancers.")


if __name__ == "__main__":
    hello_elbv2(boto3.client("elbv2"))
```

- Para obter detalhes da API, consulte a DescribeLoadBalancersReferência da API AWS SDK for Python (Boto3).

Ações para o Elastic Load Balancing - Versão 2 usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Elastic Load Balancing — Versão 2 com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Elastic Load Balancing Versão 2 e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Elastic Load Balancing - Versão 2 usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Elastic Load Balancing Versão 2.

Exemplos

- Usar AddListenerCertificates com uma CLI
- Usar AddTags com uma CLI
- Use CreateListenercom um AWS SDK ou CLI
- Use CreateLoadBalancercom um AWS SDK ou CLI
- Usar CreateRule com uma CLI
- Use CreateTargetGroupcom um AWS SDK ou CLI
- Usar DeleteListener com uma CLI
- Use DeleteLoadBalancercom um AWS SDK ou CLI
- Usar DeleteRule com uma CLI
- Use DeleteTargetGroupcom um AWS SDK ou CLI
- Usar DeregisterTargets com uma CLI
- Usar DescribeAccountLimits com uma CLI
- Usar DescribeListenerCertificates com uma CLI
- Usar DescribeListeners com uma CLI
- Usar DescribeLoadBalancerAttributes com uma CLI
- Use DescribeLoadBalancerscom um AWS SDK ou CLI
- Usar DescribeRules com uma CLI

- Usar DescribeSslPolicies com uma CLI
- Usar DescribeTags com uma CLI
- Usar DescribeTargetGroupAttributes com uma CLI
- Use DescribeTargetGroupscom um AWS SDK ou CLI
- Use DescribeTargetHealthcom um AWS SDK ou CLI
- Usar ModifyListener com uma CLI
- Usar ModifyLoadBalancerAttributes com uma CLI
- Usar ModifyRule com uma CLI
- Usar ModifyTargetGroup com uma CLI
- Usar ModifyTargetGroupAttributes com uma CLI
- Usar RegisterTargets com uma CLI
- Usar RemoveListenerCertificates com uma CLI
- Usar RemoveTags com uma CLI
- Usar SetIpAddressType com uma CLI
- Usar SetRulePriorities com uma CLI
- Usar SetSecurityGroups com uma CLI
- Usar SetSubnets com uma CLI

Usar **`AddListenerCertificates`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddListenerCertificates`.

CLI

AWS CLI

Adicionar um certificado a um receptor seguro

Esse exemplo adiciona o certificado especificado ao receptor seguro especificado.

Comando:

```
aws elbv2 add-listener-certificates --listener-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 --
```

```
certificates CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705
```

Saída:

```
{
  "Certificates": [
      {
          "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705",
          "IsDefault": false
      }
  ]
}
```

- Para obter detalhes da API, consulte AddListenerCertificatesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona um certificado adicional ao receptor especificado.

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
 -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'}
```

Saída:

```
CertificateArn
   IsDefault
--------------
   ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-
b1d7-5217feba1d97 False
```

- Para obter detalhes da API, consulte AddListenerCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona um certificado adicional ao receptor especificado.

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
 -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'}
```

Saída:

```
CertificateArn
   IsDefault
--------------
   ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-
b1d7-5217feba1d97 False
```

- Para obter detalhes da API, consulte AddListenerCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AddTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddTags`.

CLI

AWS CLI

Adicionar tags ao balanceador de carga

O exemplo `add-tags` a seguir adiciona as tags `project` e `department` ao balanceador de carga especificado.

```
aws elbv2 add-tags \
    --resource-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --tags "Key=project,Value=lima" "Key=department,Value=digital-media"
```

- Para obter detalhes da API, consulte AddTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona uma nova tag ao recurso **`AWS.Tools.ElasticLoadBalancingV2`** especificado.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- Para obter detalhes da API, consulte AddTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona uma nova tag ao recurso **`AWS.Tools.ElasticLoadBalancingV2`** especificado.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- Para obter detalhes da API, consulte AddTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateListener`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateListener`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Introdução ao balanceamento de carga

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
   /// and forwards requests to the specified target group.
   /// </summary>
   /// <param name="name">The name for the new load balancer.</param>
   /// <param name="subnetIds">Subnets for the load balancer.</param>
   /// <param name="targetGroup">Target group for forwarded requests.</param>
   /// <returns>The new LoadBalancer object.</returns>
   public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
   {
       var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
           new CreateLoadBalancerRequest()
           {
               Name = name,
               Subnets = subnetIds
           });
       var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

       // Wait for load balancer to be available.
       var loadBalancerReady = false;
       while (!loadBalancerReady)
       {
           try
           {
               var describeResponse =
                   await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                       new DescribeLoadBalancersRequest()
                       {
```

```
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }
```

- Para obter detalhes da API, consulte CreateListenera Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: como criar um receptor HTTP

O exemplo de `create-listener` a seguir cria um receptor HTTP para o Application Load Balancer especificado. Esse receptor encaminha as solicitações para o grupo de destino especificado.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --protocol HTTP \
    --port 80 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Para obter mais informações, consulte Tutorial: Criar um Application Load Balancer usando a AWS CLI no Guia do usuário para Application Load Balancers.

Exemplo 2: como criar um receptor HTTPS

O exemplo de `create-listener` a seguir cria um receptor HTTPS para o Application Load Balancer especificado. Esse receptor encaminha as solicitações para o grupo de destino especificado. É necessário especificar um certificado SSL para um receptor HTTPS. Você pode criar e gerenciar certificados usando o AWS Certificate Manager (ACM). Como alternativa, você pode criar um certificado usando SSL/TLS ferramentas, obter o certificado assinado por uma autoridade de certificação (CA) e fazer o upload do certificado para AWS Identity and Access Management (IAM).

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --protocol HTTPS \
    --port 443 \
    --certificates CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557 \
    --ssl-policy ELBSecurityPolicy-2016-08 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Para obter mais informações, consulte Adicionar um receptor HTTPS no Guia do usuário de Application Load Balancers.

Exemplo 3: como criar um receptor TCP

O exemplo de `create-listener` a seguir cria um receptor TCP para o Network Load Balancer especificado. Esse receptor encaminha as solicitações para o grupo de destino especificado.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e \
    --protocol TCP \
    --port 80 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-targets/b6bba954d1361c78
```

Para obter mais informações, consulte Tutorial: Criar um Network Load Balancer usando a AWS CLI no Guia do usuário para Network Load Balancers.

Exemplo 4: como criar um receptor TLS

O exemplo de `create-listener` a seguir cria um receptor TLS para o Network Load Balancer especificado. Esse receptor encaminha as solicitações para o grupo de destino especificado. É necessário especificar um certificado SSL para um listener TLS.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --protocol TLS \
    --port 443 \
    --certificates CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557 \
    --ssl-policy ELBSecurityPolicy-2016-08 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Para obter mais informações, consulte Listeners TLS para o Network Load Balancer no Guia do usuário dos Network Load Balancers.

Exemplo 5: como criar um receptor UDP

O exemplo de `create-listener` a seguir cria um receptor UDP para o Network Load Balancer especificado. Esse receptor encaminha as solicitações para o grupo de destino especificado.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e \
    --protocol UDP \
    --port 53 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-targets/b6bba954d1361c78
```

Para obter mais informações, consulte Tutorial: Criar um Network Load Balancer usando a AWS CLI no Guia do usuário para Network Load Balancers.

Exemplo 6: como criar um receptor para o gateway e encaminhamento especificados

O exemplo de `create-listener` a seguir cria um receptor para o Gateway Load Balancer especificado. Esse receptor encaminha as solicitações para o grupo de destino especificado.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
east-1:850631746142:loadbalancer/gwy/my-gateway-load-balancer/e0f9b3d5c7f7d3d6 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
east-1:850631746142:targetgroup/my-glb-targets/007ca469fae3bb1615
```

Saída:

```
{
    "Listeners": [
        {
            "ListenerArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:listener/gwy/my-agw-lb-example2/e0f9b3d5c7f7d3d6/
afc127db15f925de",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:loadbalancer/gwy/my-agw-lb-example2/e0f9b3d5c7f7d3d6",
            "DefaultActions": [
                {
                    "Type": "forward",
```

```
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:targetgroup/test-tg-agw-2/007ca469fae3bb1615",
                    "ForwardConfig": {
                        "TargetGroups": [
                            {
                                "TargetGroupArn":
 "arn:aws:elasticloadbalancing:us-east-1:850631746142:targetgroup/test-tg-
agw-2/007ca469fae3bb1615"
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

Para obter mais informações, consulte Introdução aos balanceadores de carga de gateway usando a AWS CLI no Guia do usuário para balanceadores de carga de gateway.

- Para obter detalhes da API, consulte CreateListenerem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
 targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
```

```
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
```

```
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateListenera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
const { Listeners } = await client.send(
  new CreateListenerCommand({
    LoadBalancerArn: state.loadBalancerArn,
    Protocol: state.targetGroupProtocol,
    Port: state.targetGroupPort,
    DefaultActions: [
      { Type: "forward", TargetGroupArn: state.targetGroupArn },
```

```
        ],
      }),
    );
```

- Para obter detalhes da API, consulte CreateListenera Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um receptor do ALB com a ação padrão “Encaminhar” para enviar tráfego ao grupo de destino especificado.

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  ForwardConfig = @{
    TargetGroups = @(
      @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
    )
    TargetGroupStickinessConfig = @{
      DurationSeconds = 900
      Enabled = $true
    }
  }
  Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -
Protocol "HTTP" -DefaultAction $defaultAction
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
Port            : 8001
```

```
Protocol        : HTTP
SslPolicy       :
```

- Para obter detalhes da API, consulte CreateListenerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um receptor do ALB com a ação padrão “Encaminhar” para enviar tráfego ao grupo de destino especificado.

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  ForwardConfig = @{
    TargetGroups = @(
      @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
    )
    TargetGroupStickinessConfig = @{
      DurationSeconds = 900
      Enabled = $true
    }
  }
  Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -
Protocol "HTTP" -DefaultAction $defaultAction
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
Port            : 8001
Protocol        : HTTP
SslPolicy       :
```

- Para obter detalhes da API, consulte CreateListenerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
 to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
 listener for.
        :param target_group: An existing target group that is added as a listener
 to the
                             load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
```

```
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a CreateListenerReferência da API AWS SDK for Python (Boto3).

Use **`CreateLoadBalancer`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateLoadBalancer`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Introdução ao balanceamento de carga

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
   /// and forwards requests to the specified target group.
   /// </summary>
   /// <param name="name">The name for the new load balancer.</param>
   /// <param name="subnetIds">Subnets for the load balancer.</param>
   /// <param name="targetGroup">Target group for forwarded requests.</param>
   /// <returns>The new LoadBalancer object.</returns>
   public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
   {
       var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
           new CreateLoadBalancerRequest()
```

```
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
```

```
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }
```

- Para obter detalhes da API, consulte CreateLoadBalancera Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: Para criar um Internet-facing balanceador de carga

O `create-load-balancer` exemplo a seguir cria um Internet-facing Application Load Balancer e ativa as zonas de disponibilidade para as sub-redes especificadas.

```
aws elbv2 create-load-balancer \
    --name my-load-balancer \
    --subnets subnet-b7d581c0 subnet-8360a9e7
```

Saída:

```
{
    "LoadBalancers": [
        {
            "Type": "application",
            "Scheme": "internet-facing",
            "IpAddressType": "ipv4",
            "VpcId": "vpc-3ac0fb5f",
            "AvailabilityZones": [
                {
                    "ZoneName": "us-west-2a",
                    "SubnetId": "subnet-8360a9e7"
                },
                {
                    "ZoneName": "us-west-2b",
                    "SubnetId": "subnet-b7d581c0"
                }
```

```
            ],
            "CreatedTime": "2017-08-25T21:26:12.920Z",
            "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
            "DNSName": "my-load-balancer-424835706.us-west-2.elb.amazonaws.com",
            "SecurityGroups": [
                "sg-5943793c"
            ],
            "LoadBalancerName": "my-load-balancer",
            "State": {
                "Code": "provisioning"
            },
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
        }
    ]
}
```

Para obter mais informações, consulte Tutorial: Criar um Application Load Balancer usando a AWS CLI no Guia do usuário para Application Load Balancers.

Exemplo 2: como criar um balanceador de carga interno

O exemplo de `create-load-balancer` a seguir cria um Application Load Balancer interno e habilita as zonas de disponibilidade para as sub-redes especificadas.

```
aws elbv2 create-load-balancer \
    --name my-internal-load-balancer \
    --scheme internal \
    --subnets subnet-b7d581c0 subnet-8360a9e7
```

Saída:

```
{
    "LoadBalancers": [
        {
            "Type": "application",
            "Scheme": "internal",
            "IpAddressType": "ipv4",
            "VpcId": "vpc-3ac0fb5f",
            "AvailabilityZones": [
                {
                    "ZoneName": "us-west-2a",
```

```
                    "SubnetId": "subnet-8360a9e7"
                },
                {
                    "ZoneName": "us-west-2b",
                    "SubnetId": "subnet-b7d581c0"
                }
            ],
            "CreatedTime": "2016-03-25T21:29:48.850Z",
            "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
            "DNSName": "internal-my-internal-load-balancer-1529930873.us-
west-2.elb.amazonaws.com",
            "SecurityGroups": [
                "sg-5943793c"
            ],
            "LoadBalancerName": "my-internal-load-balancer",
            "State": {
                "Code": "provisioning"
            },
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-internal-load-balancer/5b49b8d4303115c2"
        }
    ]
}
```

Para obter mais informações, consulte Tutorial: Criar um Application Load Balancer usando a AWS CLI no Guia do usuário para Application Load Balancers.

Exemplo 3: como criar um Network Load Balancer

O `create-load-balancer` exemplo a seguir cria um Internet-facing Network Load Balancer e ativa a Zona de Disponibilidade para a sub-rede especificada. Ele usa o mapeamento de sub-redes para associar o endereço IP do Elastic específico à interface de rede usada pelos nós do balanceador de carga da zona de disponibilidade.

```
aws elbv2 create-load-balancer \
    --name my-network-load-balancer \
    --type network \
    --subnet-mappings SubnetId=subnet-b7d581c0,AllocationId=eipalloc-64d5890a
```

Saída:

```
{
```

```
    "LoadBalancers": [
        {
            "Type": "network",
            "Scheme": "internet-facing",
            "IpAddressType": "ipv4",
            "VpcId": "vpc-3ac0fb5f",
            "AvailabilityZones": [
                {
                    "LoadBalancerAddresses": [
                        {
                            "IpAddress": "35.161.207.171",
                            "AllocationId": "eipalloc-64d5890a"
                        }
                    ],
                    "ZoneName": "us-west-2b",
                    "SubnetId": "subnet-5264e837"
                }
            ],
            "CreatedTime": "2017-10-15T22:41:25.657Z",
            "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
            "DNSName": "my-network-load-balancer-5d1b75f4f1cee11e.elb.us-
west-2.amazonaws.com",
            "LoadBalancerName": "my-network-load-balancer",
            "State": {
                "Code": "provisioning"
            },
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e"
        }
    ]
}
```

Para obter mais informações, consulte Tutorial: Criar um Network Load Balancer usando a AWS CLI no Guia do usuário para Network Load Balancers.

Exemplo 4: como criar um Gateway Load Balancer

O exemplo de `create-load-balancer` a seguir cria um Gateway Load Balancer e habilita as zonas de disponibilidade para as sub-redes especificadas.

```
aws elbv2 create-load-balancer \
    --name my-gateway-load-balancer \
    --type gateway \
```

```
    --subnets subnet-dc83f691 subnet-a62583f9
```

Saída:

```
{
    "LoadBalancers": [
        {
            "Type": "gateway",
            "VpcId": "vpc-838475fe",
            "AvailabilityZones": [
                {
                    "ZoneName": "us-east-1b",
                    "SubnetId": "subnet-a62583f9"
                },
            {
                    "ZoneName": "us-east-1a",
                    "SubnetId": "subnet-dc83f691"
                }
            ],
            "CreatedTime": "2021-07-14T19:33:43.324000+00:00",
            "LoadBalancerName": "my-gateway-load-balancer",
            "State": {
                "Code": "provisioning"
            },
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:loadbalancer/gwy/my-gateway-load-balancer/dfbb5a7d32cdee79"
        }
    ]
}
```

Para obter mais informações, consulte Introdução aos balanceadores de carga de gateway usando a AWS CLI no Guia do usuário para balanceadores de carga de gateway.

- Para obter detalhes da API, consulte CreateLoadBalancerem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /*
    * Creates an Elastic Load Balancing load balancer that uses the specified
    * subnets
    * and forwards requests to the specified target group.
    */
   public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                   .subnets(subnetIdStrings)
                   .name(lbName)
                   .scheme("internet-facing")
                   .build();

           // Create and wait for the load balancer to become available.
           CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
           String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(lbARN)
                   .build();
```

```
            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateLoadBalancera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
const { LoadBalancers } = await client.send(
  new CreateLoadBalancerCommand({
    Name: NAMES.loadBalancerName,
    Subnets: state.subnets,
  }),
);
state.loadBalancerDns = LoadBalancers[0].DNSName;
state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
await waitUntilLoadBalancerAvailable(
  { client },
  { Names: [NAMES.loadBalancerName] },
);
```

- Para obter detalhes da API, consulte CreateLoadBalancera Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um Application Load Balancer voltado para a internet com duas sub-redes.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Saída:

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
Scheme                : internet-facing
SecurityGroups        : {sg-07c3414abb8811cbd}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : application
VpcId                 : vpc-2cfd7000
```

- Para obter detalhes da API, consulte CreateLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um Application Load Balancer voltado para a internet com duas sub-redes.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Saída:

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
```

```
Scheme                  : internet-facing
SecurityGroups          : {sg-07c3414abb8811cbd}
State                   : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                    : application
VpcId                   : vpc-2cfd7000
```

- Para obter detalhes da API, consulte CreateLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
 subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
```

```
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer
```

- Para obter detalhes da API, consulte a CreateLoadBalancerReferência da API AWS SDK for Python (Boto3).

Usar **`CreateRule`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateRule`.

CLI

AWS CLI

Exemplo 1: criar uma regra usando uma condição de caminho e uma ação de avanço

O exemplo `create-rule` a seguir cria uma regra que encaminha solicitações para o grupo de destino especificado, se o URL contiver o padrão especificado.

```
aws elbv2 create-rule \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --priority 5 \
    --conditions file://conditions-pattern.json
    --actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Conteúdo de `conditions-pattern.json`:

```
[
    {
        "Field": "path-pattern",
        "PathPatternConfig": {
            "Values": ["/images/*"]
        }
    }
]
```

Exemplo 2: criar uma regra usando uma condição de host e uma resposta fixa

O exemplo `create-rule` a seguir cria uma regra que fornece uma resposta fixa se o nome do host no cabeçalho do host corresponder ao nome do host especificado.

```
aws elbv2 create-rule \
```

```
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --priority 10 \
    --conditions file://conditions-host.json \
    --actions file://actions-fixed-response.json
```

Conteúdo de `conditions-host.json`

```
[
  {
      "Field": "host-header",
      "HostHeaderConfig": {
          "Values": ["*.example.com"]
      }
  }
]
```

Conteúdo de `actions-fixed-response.json`

```
[
    {
        "Type": "fixed-response",
        "FixedResponseConfig": {
            "MessageBody": "Hello world",
            "StatusCode": "200",
            "ContentType": "text/plain"
        }
    }
]
```

Exemplo 3: criar uma regra usando uma condição de endereço IP de origem, uma ação de autenticação e uma ação de encaminhamento

O exemplo `create-rule` a seguir cria uma regra que autentica o usuário se o endereço IP de origem corresponder ao endereço IP especificado e encaminha a solicitação para o grupo de destino especificado se a autenticação for bem-sucedida.

```
aws elbv2 create-rule \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --priority 20 \
    --conditions file://conditions-source-ip.json \
```

```
    --actions file://actions-authenticate.json
```

Conteúdo de `conditions-source-ip.json`

```
[
    {
        "Field": "source-ip",
        "SourceIpConfig": {
            "Values": ["192.0.2.0/24", "198.51.100.10/32"]
        }
    }
]
```

Conteúdo de `actions-authenticate.json`

```
[
    {
        "Type": "authenticate-oidc",
        "AuthenticateOidcConfig": {
            "Issuer": "https://idp-issuer.com",
            "AuthorizationEndpoint": "https://authorization-endpoint.com",
            "TokenEndpoint": "https://token-endpoint.com",
            "UserInfoEndpoint": "https://user-info-endpoint.com",
            "ClientId": "abcdefghijklmnopqrstuvwxyz123456789",
            "ClientSecret": "12345678901234567890123456789 0",
            "SessionCookieName": "my-cookie",
            "SessionTimeout": 3600,
            "Scope": "email",
            "AuthenticationRequestExtraParams": {
                "display": "page",
                "prompt": "login"
            },
            "OnUnauthenticatedRequest": "deny"
        },
        "Order": 1
    },
    {
        "Type": "forward",
        "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
east-1:880185128111:targetgroup/cli-test/642a97ecb0e0f26b",
        "Order": 2
    }
```

```
]
```

- Para obter detalhes da API, consulte CreateRuleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma regra de receptor com ação de resposta fixa com base no valor do cabeçalho do cliente para o receptor especificado.

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
  "httpHeaderConfig" = @{
    "HttpHeaderName" = "customHeader"
    "Values" = "header2","header1"
  }
  "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -
Action $newRuleAction -Condition $newRuleCondition -Priority 10
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Para obter detalhes da API, consulte CreateRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma regra de receptor com ação de resposta fixa com base no valor do cabeçalho do cliente para o receptor especificado.

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
  "httpHeaderConfig" = @{
    "HttpHeaderName" = "customHeader"
    "Values" = "header2","header1"
  }
  "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -
Action $newRuleAction -Condition $newRuleCondition -Priority 10
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Para obter detalhes da API, consulte CreateRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateTargetGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateTargetGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Introdução ao balanceamento de carga

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
```

```
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }
```

- Para obter detalhes da API, consulte CreateTargetGroupa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: criar um grupo de destino para o Application Load Balancer

O exemplo de `create-target-group` a seguir cria um grupo de destino para um Application Load Balancer no qual você registra destinos por ID de instância (o tipo do destino é `instance`). Esse grupo de destino usa o protocolo HTTP, a porta 80 e as configurações padrão de verificação de integridade de um grupo de destino HTTP.

```
aws elbv2 create-target-group \
    --name my-targets \
    --protocol HTTP \
    --port 80 \
    --target-type instance \
    --vpc-id vpc-3ac0fb5f
```

Saída:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
            "TargetGroupName": "my-targets",
            "Protocol": "HTTP",
            "Port": 80,
            "VpcId": "vpc-3ac0fb5f",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200"
            },
            "TargetType": "instance",
            "ProtocolVersion": "HTTP1",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Para obter mais informações, consulte Criar um grupo de destino no Guia do usuário dos Application Load Balancers.

Exemplo 2: criar um grupo de destino para rotear o tráfego de um Application Load Balancer para uma função do Lambda

O exemplo de `create-target-group` a seguir cria um grupo de destino para um Application Load Balancer, no qual o destino é uma função do Lambda (o tipo do destino é `lambda`). Por padrão, as verificações de integridade estão desabilitadas para esse grupo de destino.

```
aws elbv2 create-target-group \
    --name my-lambda-target \
    --target-type lambda
```

Saída:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-lambda-target/a3003e085dbb8ddc",
            "TargetGroupName": "my-lambda-target",
            "HealthCheckEnabled": false,
            "HealthCheckIntervalSeconds": 35,
            "HealthCheckTimeoutSeconds": 30,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200"
            },
            "TargetType": "lambda",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Para obter mais informações, consulte Lambda functions as targets no Guia do usuário de Application Load Balancers.

Exemplo 3: criar um grupo de destino para o Network Load Balancer

O exemplo de `create-target-group` a seguir cria um grupo de destino para um Network Load Balancer no qual você registra destinos por endereço IP (o tipo do destino é `ip`). Esse grupo de destino usa o protocolo TCP, a porta 80 e as configurações padrão de verificação de integridade de um grupo de destino TCP.

```
aws elbv2 create-target-group \
    --name my-ip-targets \
    --protocol TCP \
    --port 80 \
    --target-type ip \
    --vpc-id vpc-3ac0fb5f
```

Saída:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-ip-targets/b6bba954d1361c78",
            "TargetGroupName": "my-ip-targets",
            "Protocol": "TCP",
            "Port": 80,
            "VpcId": "vpc-3ac0fb5f",
            "HealthCheckEnabled": true,
            "HealthCheckProtocol": "TCP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 10,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "TargetType": "ip",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Para obter mais informações, consulte Criar um grupo de destino no Guia do usuário do Network Load Balancer.

Exemplo 4: criar um grupo de destino para rotear o tráfego de um Network Load Balancer para um Application Load Balancer

O exemplo `create-target-group` a seguir cria um grupo de destino para um Network Load Balancer no qual você registra um Application Load Balancer como destino (o tipo do destino é `alb`).

aws elbv2 create-target-group --name my-alb-target --protocol TCP --port 80 --target-type alb --vpc-id vpc-3ac0fb5f

Saída:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target/a3003e085dbb8ddc",
```

```
            "TargetGroupName": "my-alb-target",
            "Protocol": "TCP",
            "Port": 80,
            "VpcId": "vpc-838475fe",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 6,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200-399"
            },
            "TargetType": "alb",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Para obter mais informações, consulte Criar um grupo de destino com um Application Load Balancer como destino no Guia do usuário do Network Load Balancer.

Exemplo 5: criar um grupo de destino para o Balanceador de Carga de Gateway

O exemplo `create-target-group` a seguir cria um grupo destino para um Balanceador de Carga de Gateway, no qual o destino é uma instância e o protocolo do grupo de destino é `GENEVE`.

```
aws elbv2 create-target-group \
    --name my-glb-targetgroup \
    --protocol GENEVE \
    --port 6081 \
    --target-type instance \
    --vpc-id vpc-838475fe
```

Saída:

```
{
    "TargetGroups": [
        {
```

```
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-glb-targetgroup/00c3d57eacd6f40b6f",
            "TargetGroupName": "my-glb-targetgroup",
            "Protocol": "GENEVE",
            "Port": 6081,
            "VpcId": "vpc-838475fe",
            "HealthCheckProtocol": "TCP",
            "HealthCheckPort": "80",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 10,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "TargetType": "instance"
        }
    ]
}
```

Para obter mais informações, consulte Criar um grupo-alvo < https://docs.aws.amazon.com/elasticloadbalancing/latest/gateway/create-target-group.html >`__ no Guia do usuário do Gateway Load Balancer.

- Para obter detalhes da API, consulte CreateTargetGroupem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
 instance
     * health is checked.
```

```
     */
    public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }
```

- Para obter detalhes da API, consulte CreateTargetGroupa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
```

```
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
```

- Para obter detalhes da API, consulte CreateTargetGroupa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um grupo de destino com os parâmetros fornecidos.

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

Saída:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /index.html
HealthCheckPort            : 80
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 2
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
```

```
TargetGroupName             : NewTargetGroup
TargetType                  : instance
UnhealthyThresholdCount     : 5
VpcId                       : vpc-2cfd7000
```

- Para obter detalhes da API, consulte CreateTargetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um grupo de destino com os parâmetros fornecidos.

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

Saída:

```
HealthCheckEnabled          : True
HealthCheckIntervalSeconds : 30
HealthCheckPath             : /index.html
HealthCheckPort             : 80
HealthCheckProtocol         : HTTP
HealthCheckTimeoutSeconds   : 5
HealthyThresholdCount       : 2
LoadBalancerArns            : {}
Matcher                     : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                        : 80
Protocol                    : HTTP
TargetGroupArn              : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
TargetGroupName             : NewTargetGroup
TargetType                  : instance
UnhealthyThresholdCount     : 5
VpcId                       : vpc-2cfd7000
```

- Para obter detalhes da API, consulte CreateTargetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
 specifies how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
 times and
        lower thresholds. In production, you might want to decrease the
 sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
```

```
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a CreateTargetGroupReferência da API AWS SDK for Python (Boto3).

Usar **`DeleteListener`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteListener`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução ao balanceamento de carga

CLI

AWS CLI

Excluir um receptor

O exemplo `delete-listener` a seguir exclui o receptor especificado.

```
aws elbv2 delete-listener \
    --listener-arn arn:aws:elasticloadbalancing:ua-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

- Para obter detalhes da API, consulte DeleteListenerem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o receptor especificado.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

Exemplo 2: esse exemplo remove o receptor especificado do balanceador de carga.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte DeleteListenerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o receptor especificado.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

Exemplo 2: esse exemplo remove o receptor especificado do balanceador de carga.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte DeleteListenerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteLoadBalancer`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteLoadBalancer`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Introdução ao balanceamento de carga

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
```

```
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }
```

- Para obter detalhes da API, consulte DeleteLoadBalancera Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir um balanceador de carga

O exemplo `delete-load-balancer` a seguir exclui o balanceador de carga especificado.

```
aws elbv2 delete-load-balancer \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

- Para obter detalhes da API, consulte DeleteLoadBalancerna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Deletes a load balancer.
   public void deleteLoadBalancer(String lbName) {
       try {
           // Use a waiter to delete the Load Balancer.
           DescribeLoadBalancersResponse res = getLoadBalancerClient()
                   .describeLoadBalancers(describe -> describe.names(lbName));
           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                   .build();

           getLoadBalancerClient().deleteLoadBalancer(
                   builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancersDeleted(request);
           waiterResponse.matched().response().ifPresent(System.out::println);

       } catch (ElasticLoadBalancingV2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
       System.out.println(lbName + " was deleted.");
   }
```

- Para obter detalhes da API, consulte DeleteLoadBalancera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
await client.send(
  new DeleteLoadBalancerCommand({
    LoadBalancerArn: loadBalancer.LoadBalancerArn,
  }),
);
await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
  const lb = await findLoadBalancer(NAMES.loadBalancerName);
  if (lb) {
    throw new Error("Load balancer still exists.");
  }
});
```

- Para obter detalhes da API, consulte DeleteLoadBalancera Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui o balanceador de carga especificado.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte DeleteLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o balanceador de carga especificado.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte DeleteLoadBalancerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
```

```
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DeleteLoadBalancerReferência da API AWS SDK for Python (Boto3).

Usar **`DeleteRule`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteRule`.

CLI

AWS CLI

Para excluir uma regra

O exemplo de `delete-rule` a seguir exclui a regra especificada.

```
aws elbv2 delete-rule \
    --rule-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/1291d13826f405c3
```

- Para obter detalhes da API, consulte DeleteRulena Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a regra especificada do receptor

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte DeleteRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a regra especificada do receptor

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte DeleteRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteTargetGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteTargetGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Introdução ao balanceamento de carga

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

- Para obter detalhes da API, consulte DeleteTargetGroupa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir um grupo de destino

O exemplo de `delete-target-group` a seguir exclui o grupo de destino especificado.

```
aws elbv2 delete-target-group \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir o balanceador de carga no Guia do Application Load Balancer Guide.

- Para obter detalhes da API, consulte DeleteTargetGroupna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
 describe.names(targetGroupName));
```

```
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

- Para obter detalhes da API, consulte DeleteTargetGroupa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
try {
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
    client.send(
      new DeleteTargetGroupCommand({
        TargetGroupArn: TargetGroups[0].TargetGroupArn,
      }),
    ),
  );
} catch (e) {
  state.deleteLoadBalancerTargetGroupError = e;
}
```

- Para obter detalhes da API, consulte DeleteTargetGroupa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove o grupo de destino especificado.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testssssss/4e0b6076bc6483a7'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
 target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testssssss/4e0b6076bc6483a7".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte DeleteTargetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o grupo de destino especificado.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testssssss/4e0b6076bc6483a7'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
 target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testssssss/4e0b6076bc6483a7".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte DeleteTargetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
 self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
 target_group_name)
```

```
            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
```

```
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
 * delay} seconds."
        )
```

- Para obter detalhes da API, consulte a DeleteTargetGroupReferência da API AWS SDK for Python (Boto3).

Usar **`DeregisterTargets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterTargets`.

CLI

AWS CLI

Exemplo 1: cancelar o registro de um destino de um grupo de destino

O exemplo `deregister-targets` a seguir remove a instância especificada do grupo de destino especificado.

```
aws elbv2 deregister-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \
    --targets Id=i-1234567890abcdef0
```

Exemplo 2: cancelar o registro de um destino registrado usando substituições de porta

O exemplo `deregister-targets` a seguir remove uma instância de um grupo de destino que foi registrado usando substituições de porta.

```
aws elbv2 deregister-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-internal-targets/3bb63f11dfb0faf9 \
    --targets Id=i-1234567890abcdef0,Port=80 Id=i-1234567890abcdef0,Port=766
```

- Para obter detalhes da API, consulte DeregisterTargetsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela o registro da instância “i-0672a4c4cdeae3111” do grupo de destino especificado.

```
$targetDescription = New-Object
 Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- Para obter detalhes da API, consulte DeregisterTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro da instância “i-0672a4c4cdeae3111” do grupo de destino especificado.

```
$targetDescription = New-Object
 Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- Para obter detalhes da API, consulte DeregisterTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAccountLimits`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAccountLimits`.

CLI

AWS CLI

Descrever os limites do Balanceamento de Carga Elastic

O `describe-account-limits` exemplo a seguir mostra os limites do Elastic Load Balancing para sua AWS conta na região atual.

```
aws elbv2 describe-account-limits
```

Saída:

```
{
    "Limits": [
        {
            "Name": "target-groups",
            "Max": "3000"
        },
        {
            "Name": "targets-per-application-load-balancer",
            "Max": "1000"
        },
        {
            "Name": "listeners-per-application-load-balancer",
            "Max": "50"
        },
        {
            "Name": "rules-per-application-load-balancer",
            "Max": "100"
        },
        {
            "Name": "network-load-balancers",
            "Max": "50"
        },
        {
            "Name": "targets-per-network-load-balancer",
            "Max": "3000"
        },
        {
            "Name": "targets-per-availability-zone-per-network-load-balancer",
            "Max": "500"
        },
        {
            "Name": "listeners-per-network-load-balancer",
            "Max": "50"
        },
        {
            "Name": "condition-values-per-alb-rule",
```

```
            "Max": "5"
        },
        {
            "Name": "condition-wildcards-per-alb-rule",
            "Max": "5"
        },
        {
            "Name": "target-groups-per-application-load-balancer",
            "Max": "100"
        },
        {
            "Name": "target-groups-per-action-on-application-load-balancer",
            "Max": "5"
        },
        {
            "Name": "target-groups-per-action-on-network-load-balancer",
            "Max": "1"
        },
        {
            "Name": "certificates-per-application-load-balancer",
            "Max": "25"
        },
        {
            "Name": "certificates-per-network-load-balancer",
            "Max": "25"
        },
        {
            "Name": "targets-per-target-group",
            "Max": "1000"
        },
        {
            "Name": "target-id-registrations-per-application-load-balancer",
            "Max": "1000"
        },
        {
            "Name": "network-load-balancer-enis-per-vpc",
            "Max": "1200"
        },
        {
            "Name": "application-load-balancers",
            "Max": "50"
        },
        {
            "Name": "gateway-load-balancers",
```

```
            "Max": "100"
        },
        {
            "Name": "gateway-load-balancers-per-vpc",
            "Max": "100"
        },
        {
            "Name": "geneve-target-groups",
            "Max": "100"
        },
        {
            "Name": "targets-per-availability-zone-per-gateway-load-balancer",
            "Max": "300"
        }
    ]
}
```

Para obter mais informações, consulte Quotas na Referência geral da AWS .

- Para obter detalhes da API, consulte DescribeAccountLimitsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse comando lista os limites da conta ELB2 para determinada região.

```
Get-ELB2AccountLimit
```

Saída:

```
Max  Name
---  ----
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
```

```
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- Para obter detalhes da API, consulte DescribeAccountLimitsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse comando lista os limites da conta ELB2 para determinada região.

```
Get-ELB2AccountLimit
```

Saída:

```
Max  Name
---  ----
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- Para obter detalhes da API, consulte DescribeAccountLimitsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeListenerCertificates`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeListenerCertificates`.

CLI

AWS CLI

Descrever os certificados para um receptor seguro

Esse exemplo descreve os certificados do receptor seguro especificado.

Comando:

```
aws elbv2 describe-listener-certificates --listener-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

Saída:

```
{
  "Certificates": [
      {
          "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705",
          "IsDefault": false
      },
      {
          "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557",
          "IsDefault": false
      },
      {
          "CertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
fe59da96-6f58-4a22-8eed-6d0d50477e1d",
          "IsDefault": true
      }
  ]
}
```

- Para obter detalhes da API, consulte DescribeListenerCertificatesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o certificado do receptor especificado.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
CertificateArn
   IsDefault
--------------
   ---------
arn:aws:acm:us-
east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c True
```

- Para obter detalhes da API, consulte DescribeListenerCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o certificado do receptor especificado.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
CertificateArn
   IsDefault
--------------
   ---------
arn:aws:acm:us-
east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c True
```

- Para obter detalhes da API, consulte DescribeListenerCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeListeners`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeListeners`.

CLI

AWS CLI

Descrever um receptor

Este exemplo descreve o receptor especificado.

Comando:

```
aws elbv2 describe-listeners --listener-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2
```

Saída:

```
{
  "Listeners": [
      {
          "Port": 80,
          "Protocol": "HTTP",
          "DefaultActions": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                  "Type": "forward"
              }
          ],
          "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
          "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2"
      }
  ]
}
```

Descrever os receptores de um balanceador de carga

Este exemplo descreve os receptores do balanceador de carga especificado.

Comando:

```
aws elbv2 describe-listeners --load-balancer-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

Saída:

```
{
  "Listeners": [
      {
          "Port": 443,
          "Protocol": "HTTPS",
          "DefaultActions": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                  "Type": "forward"
              }
          ],
          "SslPolicy": "ELBSecurityPolicy-2015-05",
          "Certificates": [
              {
                  "CertificateArn": "arn:aws:iam::123456789012:server-
certificate/my-server-cert"
              }
          ],
          "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
          "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/0467ef3c8400ae65"
      },
      {
          "Port": 80,
          "Protocol": "HTTP",
          "DefaultActions": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                  "Type": "forward"
              }
```

```
          ],
          "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
          "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2"
      }
  ]
}
```

- Para obter detalhes da API, consulte DescribeListenersna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo descreve os ouvintes do especificado ALB/NLB.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port            : 80
Protocol        : HTTP
SslPolicy       :

Certificates    : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port            : 443
Protocol        : HTTPS
```

```
SslPolicy       : ELBSecurityPolicy-2016-08
```

- Para obter detalhes da API, consulte DescribeListenersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve os ouvintes do especificado ALB/NLB.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port            : 80
Protocol        : HTTP
SslPolicy       :

Certificates    : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port            : 443
Protocol        : HTTPS
SslPolicy       : ELBSecurityPolicy-2016-08
```

- Para obter detalhes da API, consulte DescribeListenersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeLoadBalancerAttributes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeLoadBalancerAttributes`.

CLI

AWS CLI

Descrever os atributos do balanceador de carga

O exemplo `describe-load-balancer-attributes` a seguir exibe os atributos do balanceador de carga especificado.

```
aws elbv2 describe-load-balancer-attributes \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

O exemplo de saída a seguir mostra os atributos de um Application Load Balancer.

```
{
    "Attributes": [
        {
            "Value": "false",
            "Key": "access_logs.s3.enabled"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.bucket"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.prefix"
        },
        {
            "Value": "60",
            "Key": "idle_timeout.timeout_seconds"
        },
        {
            "Value": "false",
            "Key": "deletion_protection.enabled"
        },
        {
            "Value": "true",
            "Key": "routing.http2.enabled"
        }
    ]
}
```

O exemplo de saída a seguir inclui os atributos de um Network Load Balancer.

```
{
    "Attributes": [
        {
            "Value": "false",
            "Key": "access_logs.s3.enabled"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.bucket"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.prefix"
        },
        {
            "Value": "false",
            "Key": "deletion_protection.enabled"
        },
        {
            "Value": "false",
            "Key": "load_balancing.cross_zone.enabled"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancerAttributesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse comando descreve os atributos de determinado balanceador de carga.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Saída:

```
Key                                Value
```

```
---                                 -----
access_logs.s3.enabled              false
load_balancing.cross_zone.enabled true
access_logs.s3.prefix
deletion_protection.enabled         false
access_logs.s3.bucket
```

- Para obter detalhes da API, consulte DescribeLoadBalancerAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse comando descreve os atributos de determinado balanceador de carga.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Saída:

```
Key                                 Value
---                                 -----
access_logs.s3.enabled              false
load_balancing.cross_zone.enabled true
access_logs.s3.prefix
deletion_protection.enabled         false
access_logs.s3.bucket
```

- Para obter detalhes da API, consulte DescribeLoadBalancerAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeLoadBalancers`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeLoadBalancers`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Introdução ao balanceamento de carga

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }
```

- Para obter detalhes da API, consulte DescribeLoadBalancersa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como descrever um balanceador de carga

Este exemplo descreve o balanceador de carga especificado.

Comando:

```
aws elbv2 describe-load-balancers --load-balancer-
arns arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-
load-balancer/50dc6c495c0c9188
```

Saída:

```
{
  "LoadBalancers": [
      {
          "Type": "application",
          "Scheme": "internet-facing",
          "IpAddressType": "ipv4",
          "VpcId": "vpc-3ac0fb5f",
          "AvailabilityZones": [
              {
                  "ZoneName": "us-west-2a",
                  "SubnetId": "subnet-8360a9e7"
              },
              {
                  "ZoneName": "us-west-2b",
                  "SubnetId": "subnet-b7d581c0"
              }
          ],
          "CreatedTime": "2016-03-25T21:26:12.920Z",
          "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
          "DNSName": "my-load-balancer-424835706.us-west-2.elb.amazonaws.com",
          "SecurityGroups": [
              "sg-5943793c"
          ],
          "LoadBalancerName": "my-load-balancer",
          "State": {
              "Code": "active"
          },
          "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
      }
  ]
}
```

Como descrever todos os balanceadores de carga

Este exemplo descreve todos os balanceadores de carga.

Comando:

```
aws elbv2 describe-load-balancers
```

- Para obter detalhes da API, consulte DescribeLoadBalancersna Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  ElasticLoadBalancingV2Client,
  DescribeLoadBalancersCommand,
} from "@aws-sdk/client-elastic-load-balancing-v2";

export async function main() {
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
    new DescribeLoadBalancersCommand({}),
  );
  const loadBalancersList = LoadBalancers.map(
    (lb) => `• ${lb.LoadBalancerName}: ${lb.DNSName}`,
  ).join("\n");
  console.log(
    "Hello, Elastic Load Balancing! Let's list some of your load balancers:\n",
    loadBalancersList,
  );
}
```

```
// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancersa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe todos os balanceadores de carga de determinada região.

```
Get-ELB2LoadBalancer
```

Saída:

```
AvailabilityZones     : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
CreatedTime           : 6/22/18 11:21:50 AM
DNSName               : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName      : test-elb1234567890
Scheme                : internet-facing
SecurityGroups        : {}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : network
VpcId                 : vpc-2cf00000
```

- Para obter detalhes da API, consulte DescribeLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe todos os balanceadores de carga de determinada região.

```
Get-ELB2LoadBalancer
```

Saída:

```
AvailabilityZones     : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
CreatedTime           : 6/22/18 11:21:50 AM
DNSName               : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName      : test-elb1234567890
Scheme                : internet-facing
SecurityGroups        : {}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : network
VpcId                 : vpc-2cf00000
```

- Para obter detalhes da API, consulte DescribeLoadBalancersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client
```

```
    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DescribeLoadBalancersReferência da API AWS SDK for Python (Boto3).

Usar **`DescribeRules`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeRules`.

CLI

AWS CLI

Exemplo 1: descrever uma regra

O exemplo `describe-rules` a seguir exibe os detalhes da regra especificada.

```
aws elbv2 describe-rules \
    --rule-arns arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-
rule/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/9683b2d02a6cabee
```

Exemplo 2: descrever as regras para um receptor

O exemplo `describe-rules` a seguir exibe os detalhes da regra para o receptor especificado. A saída inclui a regra padrão e quaisquer outras regras que você tenha adicionado.

```
aws elbv2 describe-rules \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

- Para obter detalhes da API, consulte DescribeRulesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve as regras do receptor para o ARN do receptor especificado.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 1
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 2
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba
```

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {}
IsDefault  : True
Priority   : default
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf
```

- Para obter detalhes da API, consulte DescribeRulesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as regras do receptor para o ARN do receptor especificado.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 1
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 2
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {}
IsDefault  : True
Priority   : default
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf
```

- Para obter detalhes da API, consulte DescribeRulesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeSslPolicies`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeSslPolicies`.

CLI

AWS CLI

Exemplo 1: listar as políticas usadas para negociação de SSL por tipo de balanceador de carga

O exemplo `describe-ssl-policies` a seguir exibe os nomes das políticas que você pode usar para negociação de SSL com um Application Load Balancer. O exemplo usa o parâmetro `--query` para exibir somente os nomes das políticas.

```
aws elbv2 describe-ssl-policies \
    --load-balancer-type application \
    --query SslPolicies[*].Name
```

Saída:

```
[
    "ELBSecurityPolicy-2016-08",
    "ELBSecurityPolicy-TLS13-1-2-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Res-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06",
    "ELBSecurityPolicy-TLS13-1-1-2021-06",
    "ELBSecurityPolicy-TLS13-1-0-2021-06",
    "ELBSecurityPolicy-TLS13-1-3-2021-06",
    "ELBSecurityPolicy-TLS-1-2-2017-01",
    "ELBSecurityPolicy-TLS-1-1-2017-01",
    "ELBSecurityPolicy-TLS-1-2-Ext-2018-06",
    "ELBSecurityPolicy-FS-2018-06",
    "ELBSecurityPolicy-2015-05",
    "ELBSecurityPolicy-TLS-1-0-2015-04",
    "ELBSecurityPolicy-FS-1-2-Res-2019-08",
    "ELBSecurityPolicy-FS-1-1-2019-08",
    "ELBSecurityPolicy-FS-1-2-2019-08",
```

```
    "ELBSecurityPolicy-FS-1-2-Res-2020-10"
]
```

Exemplo 2: listar as políticas que oferecem suporte a um protocolo específico

O exemplo `describe-ssl-policies` a seguir exibe os nomes das políticas que oferecem suporte ao protocolo TLS 1.3. O exemplo usa o parâmetro `--query` para exibir somente os nomes das políticas.

```
aws elbv2 describe-ssl-policies \
    --load-balancer-type application \
    --query SslPolicies[?contains(SslProtocols,'TLSv1.3')].Name
```

Saída:

```
[
    "ELBSecurityPolicy-TLS13-1-2-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Res-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06",
    "ELBSecurityPolicy-TLS13-1-1-2021-06",
    "ELBSecurityPolicy-TLS13-1-0-2021-06",
    "ELBSecurityPolicy-TLS13-1-3-2021-06"
]
```

Exemplo 3: exibir as cifras de uma política

O exemplo `describe-ssl-policies` a seguir exibe os nomes das cifras da política especificada. O exemplo usa o parâmetro `--query` para exibir somente os nomes das cifras. A primeira cifra da lista tem prioridade 1 e as cifras restantes estão em ordem de prioridade.

```
aws elbv2 describe-ssl-policies \
    --names ELBSecurityPolicy-TLS13-1-2-2021-06 \
    --query SslPolicies[*].Ciphers[*].Name
```

Saída:

```
[
    "TLS_AES_128_GCM_SHA256",
    "TLS_AES_256_GCM_SHA384",
    "TLS_CHACHA20_POLY1305_SHA256",
```

```
    "ECDHE-ECDSA-AES128-GCM-SHA256",
    "ECDHE-RSA-AES128-GCM-SHA256",
    "ECDHE-ECDSA-AES128-SHA256",
    "ECDHE-RSA-AES128-SHA256",
    "ECDHE-ECDSA-AES256-GCM-SHA384",
    "ECDHE-RSA-AES256-GCM-SHA384",
    "ECDHE-ECDSA-AES256-SHA384",
    "ECDHE-RSA-AES256-SHA384"
]
```

Para obter mais informações, consulte Políticas de segurança no Guia do usuário dos Application Load Balancers.

- Para obter detalhes da API, consulte DescribeSslPoliciesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo lista todas as políticas de ouvinte disponíveis para ElasticLoadBalancingV2.

```
Get-ELB2SSLPolicy
```

Saída:

```
Ciphers
                                        Name
 SslProtocols
-------
                                        ----
 ------------
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2016-08            {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-2017-01     {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01     {TLSv1.1,
 TLSv1.2}
```

```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06          {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05             {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04     {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08  {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08      {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08      {TLSv1.2}
```

- Para obter detalhes da API, consulte DescribeSslPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista todas as políticas de ouvinte disponíveis para ElasticLoadBalancingV2.

```
Get-ELB2SSLPolicy
```

Saída:

```
Ciphers
                                        Name
 SslProtocols
-------
                                        ----
 ------------
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2016-08             {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-2017-01     {TLSv1.2}
```

```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01     {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06         {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05            {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04     {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08  {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08      {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08      {TLSv1.2}
```

- Para obter detalhes da API, consulte DescribeSslPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTags`.

CLI

AWS CLI

Descrever as tags atribuídas a um balanceador de carga

Este exemplo descreve as tags atribuídas ao balanceador de carga especificado.

Comando:

```
aws elbv2 describe-tags --resource-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

Saída:

```
{
  "TagDescriptions": [
      {
          "ResourceArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
          "Tags": [
              {
                  "Value": "lima",
                  "Key": "project"
              },
              {
                  "Value": "digital-media",
                  "Key": "department"
              }
          ]
      }
  ]
}
```

- Para obter detalhes da API, consulte DescribeTagsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista as tags para o recurso especificado.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
ResourceArn
              Tags
-----------
              ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as tags para o recurso especificado.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
ResourceArn
              Tags
-----------
              ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTargetGroupAttributes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTargetGroupAttributes`.

CLI

AWS CLI

Descrever os atributos do grupo de destino

O exemplo `describe-target-group-attributes` a seguir exibe os atributos do grupo de destino especificado.

```
aws elbv2 describe-target-group-attributes \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

A saída inclui os atributos se o protocolo for HTTP ou HTTPS e o tipo de destino for `instance` ou `ip`.

```
{
```

```
    "Attributes": [
        {
            "Value": "false",
            "Key": "stickiness.enabled"
        },
        {
            "Value": "300",
            "Key": "deregistration_delay.timeout_seconds"
        },
        {
            "Value": "lb_cookie",
            "Key": "stickiness.type"
        },
        {
            "Value": "86400",
            "Key": "stickiness.lb_cookie.duration_seconds"
        },
        {
            "Value": "0",
            "Key": "slow_start.duration_seconds"
        }
    ]
}
```

A saída a seguir inclui os atributos se o protocolo for HTTP ou HTTPS e o tipo de destino for `lambda`.

```
{
    "Attributes": [
        {
            "Value": "false",
            "Key": "lambda.multi_value_headers.enabled"
        }
    ]
}
```

A saída a seguir inclui os atributos se o protocolo for TCP, TLS, UDP ou TCP_UDP.

```
{
    "Attributes": [
        {
            "Value": "false",
            "Key": "proxy_protocol_v2.enabled"
```

```
        },
        {
            "Value": "300",
            "Key": "deregistration_delay.timeout_seconds"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeTargetGroupAttributesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve os atributos do grupo de destino especificado.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
Key                                  Value
---                                  -----
stickiness.enabled                   false
deregistration_delay.timeout_seconds 300
stickiness.type                      lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds          0
load_balancing.algorithm.type        round_robin
```

- Para obter detalhes da API, consulte DescribeTargetGroupAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve os atributos do grupo de destino especificado.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
Key                                  Value
---                                  -----
stickiness.enabled                   false
deregistration_delay.timeout_seconds 300
stickiness.type                      lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds          0
load_balancing.algorithm.type        round_robin
```

- Para obter detalhes da API, consulte DescribeTargetGroupAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeTargetGroups`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeTargetGroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e gerenciar um serviço resiliente

CLI

AWS CLI

Exemplo 1: como descrever um grupo de destino

O exemplo de `describe-target-groups` a seguir exibe os detalhes do grupo de destino especificado.

```
aws elbv2 describe-target-groups \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Saída:

```
{
    "TargetGroups": [
        {
```

```
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
            "TargetGroupName": "my-targets",
            "Protocol": "HTTP",
            "Port": 80,
            "VpcId": "vpc-3ac0fb5f",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200"
            },
            "LoadBalancerArns": [
                "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
            ],
            "TargetType": "instance",
            "ProtocolVersion": "HTTP1",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Exemplo 2: como descrever todos os grupos de destino de um balanceador de carga

O exemplo de `describe-target-groups` a seguir exibe os detalhes de todos os grupos de destino do balanceador de carga especificado. O exemplo usa o parâmetro `--query` para exibir somente os nomes dos grupo de destino.

```
aws elbv2 describe-target-groups \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --query TargetGroups[*].TargetGroupName
```

Saída:

```
[
```

```
    "my-instance-targets",
    "my-ip-targets",
    "my-lambda-target"
]
```

Para ter mais informações, consulte Grupos de destino no Guia do Application Load Balancer.

- Para obter detalhes da API, consulte DescribeTargetGroupsna Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
const { TargetGroups } = await client.send(
  new DescribeTargetGroupsCommand({
    Names: [NAMES.loadBalancerTargetGroupName],
  }),
);
```

- Para obter detalhes da API, consulte DescribeTargetGroupsa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve o grupo de destino especificado.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /
HealthCheckPort            : traffic-port
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- Para obter detalhes da API, consulte DescribeTargetGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o grupo de destino especificado.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /
HealthCheckPort            : traffic-port
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
```

```
Port                        : 80
Protocol                    : HTTP
TargetGroupArn              : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName             : test-tg
TargetType                  : instance
UnhealthyThresholdCount     : 2
VpcId                       : vpc-2cfd7000
```

- Para obter detalhes da API, consulte DescribeTargetGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeTargetHealth`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeTargetHealth`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Introdução ao balanceamento de carga

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
```

```
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }
```

- Para obter detalhes da API, consulte DescribeTargetHealtha Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: como descrever a integridade dos destinos de um grupo de destino

O exemplo de `describe-target-health` a seguir exibe os detalhes de integridade dos destinos do grupo de destino especificado. Os destinos estão íntegros.

```
aws elbv2 describe-target-health \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Saída:

```
{
    "TargetHealthDescriptions": [
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-ceddcd4d",
                "Port": 80
            },
            "TargetHealth": {
                "State": "healthy"
            }
        },
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-0f76fade",
                "Port": 80
            },
            "TargetHealth": {
                "State": "healthy"
            }
        }
    ]
}
```

Exemplo 2: como descrever a integridade de um destino

O exemplo de `describe-target-health` a seguir exibe os detalhes de integridade do destino especificado. Este destino é íntegro.

```
aws elbv2 describe-target-health \
    --targets Id=i-0f76fade,Port=80 \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Saída:

```
{
    "TargetHealthDescriptions": [
        {
            "HealthCheckPort": "80",
```

```
            "Target": {
                "Id": "i-0f76fade",
                "Port": 80
            },
            "TargetHealth": {
                "State": "healthy"
            }
        }
    ]
}
```

O exemplo de saída a seguir é de um destino cujo grupo de destino não está especificado em uma ação de um receptor. Esse destino não pode receber tráfego do balanceador de carga.

```
{
    "TargetHealthDescriptions": [
    {
        "HealthCheckPort": "80",
        "Target": {
            "Id": "i-0f76fade",
            "Port": 80
        },
            "TargetHealth": {
                "State": "unused",
                "Reason": "Target.NotInUse",
                "Description": "Target group is not configured to receive traffic
 from the load balancer"
            }
        }
    ]
}
```

O exemplo de saída a seguir é de um destino cujo grupo de destino foi especificado em uma ação de um receptor. O destino ainda está sendo registrado.

```
{
    "TargetHealthDescriptions": [
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-0f76fade",
                "Port": 80
```

```
            },
            "TargetHealth": {
                "State": "initial",
                "Reason": "Elb.RegistrationInProgress",
                "Description": "Target registration is in progress"
            }
        }
    ]
}
```

O exemplo de saída a seguir é de um destino não íntegro.

```
{
    "TargetHealthDescriptions": [
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-0f76fade",
                "Port": 80
            },
            "TargetHealth": {
                "State": "unhealthy",
                "Reason": "Target.Timeout",
                "Description": "Connection to target timed out"
            }
        }
    ]
}
```

O exemplo de saída a seguir é de um destino que é uma função do Lambda e com verificações de integridade desabilitadas.

```
{
    "TargetHealthDescriptions": [
        {
            "Target": {
                "Id": "arn:aws:lambda:us-west-2:123456789012:function:my-
function",
                "AvailabilityZone": "all",
            },
            "TargetHealth": {
                "State": "unavailable",
                "Reason": "Target.HealthCheckDisabled",
```

```
                "Description": "Health checks are not enabled for this target"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeTargetHealthna Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Checks the health of the instances in the target group.
   public List<TargetHealthDescription> checkTargetHealth(String
targetGroupName) {
       DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
               .names(targetGroupName)
               .build();

       DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

       DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()

.targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
               .build();

       DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
       return healthResponse.targetHealthDescriptions();
   }
```

- Para obter detalhes da API, consulte DescribeTargetHealtha Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const { TargetHealthDescriptions } = await client.send(
  new DescribeTargetHealthCommand({
    TargetGroupArn: TargetGroups[0].TargetGroupArn,
  }),
);
```

- Para obter detalhes da API, consulte DescribeTargetHealtha Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo retorna o status de integridade dos destinos presentes no grupo de destino especificado.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
HealthCheckPort Target
 TargetHealth
--------------- ------
 ------------
```

```
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- Para obter detalhes da API, consulte DescribeTargetHealthem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o status de integridade dos destinos presentes no grupo de destino especificado.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
HealthCheckPort Target
 TargetHealth
--------------- ------
 ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- Para obter detalhes da API, consulte DescribeTargetHealthem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
```

```
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
                    "Ensure the load balancer exists, is in the correct AWS
region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
correct region, "
                    "and ensure it has not been deleted or created in a different
account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

- Para obter detalhes da API, consulte a DescribeTargetHealthReferência da API AWS SDK for Python (Boto3).

Usar **`ModifyListener`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyListener`.

CLI

AWS CLI

Exemplo 1: alterar a ação padrão para uma ação de encaminhamento

O exemplo `modify-listener` a seguir altera a ação padrão para uma ação de `forward` para o receptor especificado.

```
aws elbv2 modify-listener \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-new-targets/2453ed029918f21f
```

Saída:

```
{
    "Listeners": [
        {
            "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
            "Protocol": "HTTP",
            "Port": 80,
            "DefaultActions": [
                {
                    "Type": "forward",
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-new-targets/2453ed029918f21f"
                }
            ]
```

```
        }
    ]
}
```

Exemplo 2: alterar a ação padrão para uma ação de redirecionamento

O exemplo `modify-listener` a seguir altera a ação padrão para uma ação de `redirect` para o receptor especificado.

```
aws elbv2 modify-listener \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --default-actions Type=redirect,
 RedirectConfig='{Protocol=HTTPS,StatusCode=HTTP_302}'
```

Saída:

```
{
    "Listeners": [
        {
            "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
            "Protocol": "HTTP",
            "Port": 80,
            "DefaultActions": [
                {
                    "Type": "redirect",
                    "RedirectConfig": {
                        "Protocol": "HTTPS",
                        "Port": "#{port}",
                        "Host": "#{host}",
                        "Path": "/#{path}",
                        "Query": "#{query}",
                        "StatusCode": "HTTP_302",
                    }
                }
            ]
        }
    ]
```

```
}
```

Exemplo 3: alterar o certificado do servidor

O exemplo `modify-listener` a seguir altera o certificado do servidor para o receptor HTTPS especificado.

```
aws elbv2 modify-listener \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65 \
    --certificates CertificateArn=arn:aws:iam::123456789012:server-certificate/
my-new-server-cert
```

Saída:

```
{
    "Listeners": [
        {
            "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/0467ef3c8400ae65",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
            "Protocol": "HTTPS",
            "Port": 443,
            "DefaultActions": [
                {
                    "Type": "forward",
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067"
                }
            ],
            "SslPolicy": "ELBSecurityPolicy-2015-05",
            "Certificates": [
                {
                    "CertificateArn": "arn:aws:iam::123456789012:server-
certificate/my-new-server-cert"
                }
            ],
        }
    ]
}
```

Para ter mais informações, consulte Listener rules no Guia do usuário de Application Load Balancers.

- Para obter detalhes da API, consulte ModifyListenerna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica a ação padrão do receptor especificado para resposta fixa.

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port
 8080 -DefaultAction $newDefaultAction
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
Port            : 8080
Protocol        : HTTP
SslPolicy       :
```

- Para obter detalhes da API, consulte ModifyListenerem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica a ação padrão do receptor especificado para resposta fixa.

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port
 8080 -DefaultAction $newDefaultAction
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
Port            : 8080
Protocol        : HTTP
SslPolicy       :
```

- Para obter detalhes da API, consulte ModifyListenerem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyLoadBalancerAttributes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyLoadBalancerAttributes`.

CLI

AWS CLI

Ativar a proteção contra exclusão

Este exemplo habilita a proteção contra exclusão do balanceador de carga especificado.

Comando:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --attributes Key=deletion_protection.enabled,Value=true
```

Saída:

```
{
  "Attributes": [
      {
          "Value": "true",
          "Key": "deletion_protection.enabled"
      },
      {
          "Value": "false",
          "Key": "access_logs.s3.enabled"
      },
      {
          "Value": "60",
          "Key": "idle_timeout.timeout_seconds"
      },
      {
          "Value": "",
          "Key": "access_logs.s3.prefix"
      },
      {
          "Value": "",
          "Key": "access_logs.s3.bucket"
      }
  ]
}
```

Alterar o tempo limite de inatividade

Este exemplo altera o valor do tempo limite de inatividade do balanceador de carga especificado.

Comando:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --attributes Key=idle_timeout.timeout_seconds,Value=30
```

Saída:

```
{
  "Attributes": [
      {
          "Value": "30",
          "Key": "idle_timeout.timeout_seconds"
      },
      {
          "Value": "false",
          "Key": "access_logs.s3.enabled"
      },
      {
          "Value": "",
          "Key": "access_logs.s3.prefix"
      },
      {
          "Value": "true",
          "Key": "deletion_protection.enabled"
      },
      {
          "Value": "",
          "Key": "access_logs.s3.bucket"
      }
  ]
}
```

Habilitar logs de acesso

Este exemplo habilita os logs de acesso para o balanceador de carga especificado. Observe que o bucket do S3 deve existir na mesma região que o balanceador de carga e deve conter uma política anexada que conceda acesso ao serviço Elastic Load Balancing.

Comando:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --
attributes Key=access_logs.s3.enabled,Value=true Key=access_logs.s3.bucket,Value=my-
loadbalancer-logs Key=access_logs.s3.prefix,Value=myapp
```

Saída:

```
{
  "Attributes": [
      {
          "Value": "true",
          "Key": "access_logs.s3.enabled"
      },
      {
          "Value": "my-load-balancer-logs",
          "Key": "access_logs.s3.bucket"
      },
      {
          "Value": "myapp",
          "Key": "access_logs.s3.prefix"
      },
      {
          "Value": "60",
          "Key": "idle_timeout.timeout_seconds"
      },
      {
          "Value": "false",
          "Key": "deletion_protection.enabled"
      }
  ]
}
```

- Para obter detalhes da API, consulte ModifyLoadBalancerAttributesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica os atributos do balanceador de carga especificado.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key
 = 'deletion_protection.enabled'; Value = 'true'}
```

Saída:

```
Key                                                 Value
---                                                 -----
```

```
deletion_protection.enabled                     true
access_logs.s3.enabled                          false
access_logs.s3.bucket
access_logs.s3.prefix
idle_timeout.timeout_seconds                    60
routing.http2.enabled                           true
routing.http.drop_invalid_header_fields.enabled false
```

- Para obter detalhes da API, consulte ModifyLoadBalancerAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica os atributos do balanceador de carga especificado.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key
 = 'deletion_protection.enabled'; Value = 'true'}
```

Saída:

```
Key                                             Value
---                                             -----
deletion_protection.enabled                     true
access_logs.s3.enabled                          false
access_logs.s3.bucket
access_logs.s3.prefix
idle_timeout.timeout_seconds                    60
routing.http2.enabled                           true
routing.http.drop_invalid_header_fields.enabled false
```

- Para obter detalhes da API, consulte ModifyLoadBalancerAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyRule`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyRule`.

CLI

AWS CLI

Modificar uma regra

O exemplo `modify-rule` a seguir atualiza as ações e condições da regra especificada.

```
aws elbv2 modify-rule \
  --actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \
  --conditions Field=path-pattern,Values='/images/*'
  --rule-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/9683b2d02a6cabee
```

Saída:

```
{
    "Rules": [
        {
            "Priority": "10",
            "Conditions": [
                {
                    "Field": "path-pattern",
                    "Values": [
                        "/images/*"
                    ]
                }
            ],
            "RuleArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2/9683b2d02a6cabee",
            "IsDefault": false,
            "Actions": [
                {
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                    "Type": "forward"
                }
            ]
        }
    ]
}
```

- Para obter detalhes da API, consulte ModifyRulena Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica as configurações especificadas da regra do receptor.

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
    "Values" = "/login1","/login2","/login3"
  }
  "Field" = "path-pattern"
}

Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/
f4f51dfaa033a8cc' -Condition $newRuleCondition
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Para obter detalhes da API, consulte ModifyRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica as configurações especificadas da regra do receptor.

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
    "Values" = "/login1","/login2","/login3"
  }
  "Field" = "path-pattern"
}

Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/
f4f51dfaa033a8cc' -Condition $newRuleCondition
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Para obter detalhes da API, consulte ModifyRuleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyTargetGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyTargetGroup`.

CLI

AWS CLI

Modificar as configurações de verificação de integridade de um grupo de destino

O exemplo `modify-target-group` a seguir altera a configuração das verificações de integridade usadas para avaliar a integridade dos destinos para o grupo de destino especificado. Observe que, devido à forma como a CLI analisa as vírgulas, você deve delimitar o intervalo da opção `--matcher` com aspas simples em vez de aspas duplas.

```
aws elbv2 modify-target-group \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-https-targets/2453ed029918f21f \
    --health-check-protocol HTTPS \
    --health-check-port 443 \
    --matcher HttpCode='200,299'
```

Saída:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-https-targets/2453ed029918f21f",
```

```
            "TargetGroupName": "my-https-targets",
            "Protocol": "HTTPS",
            "Port": 443,
            "VpcId": "vpc-3ac0fb5f",
            "HealthCheckProtocol": "HTTPS",
            "HealthCheckPort": "443",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "Matcher": {
                "HttpCode": "200,299"
            },
            "LoadBalancerArns": [
                "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
            ],
            "TargetType": "instance",
            "ProtocolVersion": "HTTP1",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Para ter mais informações, consulte Grupos de destino no Guia do Application Load Balancer.

- Para obter detalhes da API, consulte ModifyTargetGroupna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica as propriedades do grupo de destino especificado.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -
HealthCheckIntervalSecond 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Saída:

```
HealthCheckEnabled         : True
```

```
HealthCheckIntervalSeconds : 60
HealthCheckPath            : /index.html
HealthCheckPort            : 8080
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- Para obter detalhes da API, consulte ModifyTargetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica as propriedades do grupo de destino especificado.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -
HealthCheckIntervalSecond 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Saída:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 60
HealthCheckPath            : /index.html
HealthCheckPort            : 8080
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
```

```
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- Para obter detalhes da API, consulte ModifyTargetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyTargetGroupAttributes`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyTargetGroupAttributes`.

CLI

AWS CLI

Modificar o tempo limite de atraso do cancelamento de registro

Este exemplo define o tempo limite do atraso de cancelamento de registro para o valor especificado para o grupo de destino especificado.

Comando:

```
aws elbv2 modify-target-group-attributes --target-group-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067 --
attributes Key=deregistration_delay.timeout_seconds,Value=600
```

Saída:

```
{
  "Attributes": [
      {
          "Value": "false",
          "Key": "stickiness.enabled"
      },
      {
          "Value": "600",
          "Key": "deregistration_delay.timeout_seconds"
      },
      {
          "Value": "lb_cookie",
```

```
            "Key": "stickiness.type"
        },
        {
            "Value": "86400",
            "Key": "stickiness.lb_cookie.duration_seconds"
        }
    ]
}
```

- Para obter detalhes da API, consulte ModifyTargetGroupAttributesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica o atributo deregistration_delay do grupo de destino especificado.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Saída:

```
Key                                  Value
---                                  -----
stickiness.enabled                   false
deregistration_delay.timeout_seconds 600
stickiness.type                      lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds          0
load_balancing.algorithm.type        round_robin
```

- Para obter detalhes da API, consulte ModifyTargetGroupAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica o atributo deregistration_delay do grupo de destino especificado.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Saída:

```
Key                                 Value
---                                 -----
stickiness.enabled                  false
deregistration_delay.timeout_seconds  600
stickiness.type                     lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds         0
load_balancing.algorithm.type       round_robin
```

- Para obter detalhes da API, consulte ModifyTargetGroupAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RegisterTargets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterTargets`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução ao balanceamento de carga

CLI

AWS CLI

Exemplo 1: registrar destinos com um grupo de destino por ID de instância

O exemplo `register-targets` a seguir registra as instâncias especificadas com um grupo de destino. O grupo de destino deve ter um tipo de destino de `instance`.

```
aws elbv2 register-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \
    --targets Id=i-1234567890abcdef0 Id=i-0abcdef1234567890
```

Exemplo 2: registrar destinos em um grupo de destino usando substituições de porta

O exemplo `register-targets` a seguir registra a instância especificada com um grupo de destino usando várias portas. Isso permite que você registre contêineres na mesma instância que os destinos no grupo de destino.

```
aws elbv2 register-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-internal-targets/3bb63f11dfb0faf9 \
    --targets Id=i-0598c7d356eba48d7,Port=80 Id=i-0598c7d356eba48d7,Port=766
```

Exemplo 3: registrar destinos com um grupo de destino por endereço IP

O exemplo `register-targets` a seguir registra os endereços IP especificados com um grupo de destino. O grupo de destino deve ter um tipo de destino de `ip`.

```
aws elbv2 register-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-ip-targets/8518e899d173178f \
    --targets Id=10.0.1.15 Id=10.0.1.23
```

Exemplo 4: registrar uma função do Lambda como um destino

O exemplo `register-targets` a seguir registra os endereços IP especificados com um grupo de destino. O grupo de destino deve ter um tipo de destino de `lambda`. O Elastic Load Balancing precisa ter permissão para invocar cada função do Lambda.

```
aws elbv2 register-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-ip-targets/8518e899d173178f \
    --targets Id=arn:aws:lambda:us-west-2:123456789012:function:my-function
```

- Para obter detalhes da API, consulte RegisterTargetsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra a instância “i-0672a4c4cdeae3111” com o grupo de destino especificado.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id
 = 'i-0672a4c4cdeae3111'}
```

- Para obter detalhes da API, consulte RegisterTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra a instância “i-0672a4c4cdeae3111” com o grupo de destino especificado.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id
 = 'i-0672a4c4cdeae3111'}
```

- Para obter detalhes da API, consulte RegisterTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RemoveListenerCertificates`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemoveListenerCertificates`.

CLI

AWS CLI

Remover um certificado de um receptor seguro

Este exemplo remove os certificados especificados do receptor seguro especificado.

Comando:

```
aws elbv2 remove-listener-certificates --listener-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 --
certificates CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705
```

- Para obter detalhes da API, consulte RemoveListenerCertificatesna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove o certificado especificado do grupo de destino especificado.

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -
ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
 (RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte RemoveListenerCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o certificado especificado do grupo de destino especificado.

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -
ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
 (RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte RemoveListenerCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RemoveTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemoveTags`.

CLI

AWS CLI

Remover tags de um balanceador de carga

O exemplo `remove-tags` a seguir remove as tags `project` e `department` do balanceador de carga especificado.

```
aws elbv2 remove-tags \
    --resource-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --tag-keys project department
```

- Para obter detalhes da API, consulte RemoveTagsna Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a tag para a chave especificada.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte RemoveTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag para a chave especificada.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes da API, consulte RemoveTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SetIpAddressType`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetIpAddressType`.

CLI

AWS CLI

Definir o tipo de endereço de um balanceador de carga

Este exemplo define o tipo de endereço do balanceador de carga especificado como `dualstack`. As sub-redes do balanceador de carga devem conter blocos CIDR IPv6 associados.

Comando:

```
aws elbv2 set-ip-address-type --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --ip-address-type dualstack
```

Saída:

```
{
    "IpAddressType": "dualstack"
}
```

- Para obter detalhes da API, consulte SetIpAddressTypena Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo altera o tipo de endereço IP do balanceador de carga de 'IPv4' para ". DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Saída:

```
Value
-----
dualstack
```

- Para obter detalhes da API, consulte SetIpAddressTypeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo altera o tipo de endereço IP do balanceador de carga de 'IPv4' para ". DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Saída:

```
Value
-----
dualstack
```

- Para obter detalhes da API, consulte SetIpAddressTypeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SetRulePriorities`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetRulePriorities`.

CLI

AWS CLI

Definir a prioridade da regra

Este exemplo define a prioridade da regra especificada.

Comando:

```
aws elbv2 set-rule-priorities --rule-
priorities RuleArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2/1291d13826f405c3,Priority=5
```

Saída:

```
{
  "Rules": [
      {
          "Priority": "5",
          "Conditions": [
              {
                  "Field": "path-pattern",
```

```
                    "Values": [
                        "/img/*"
                    ]
                }
            ],
            "RuleArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2/1291d13826f405c3",
            "IsDefault": false,
            "Actions": [
                {
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                    "Type": "forward"
                }
            ]
        }
    ]
}
```

- Para obter detalhes da API, consulte SetRulePrioritiesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera a prioridade da regra de receptor especificada.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 11
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- Para obter detalhes da API, consulte SetRulePrioritiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera a prioridade da regra de receptor especificada.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 11
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- Para obter detalhes da API, consulte SetRulePrioritiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SetSecurityGroups`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetSecurityGroups`.

CLI

AWS CLI

Associar um grupo de segurança a um balanceador de carga

Este exemplo associa o grupo de segurança especificado ao balanceador de carga especificado.

Comando:

```
aws elbv2 set-security-groups --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --security-groups sg-5943793c
```

Saída:

```
{
  "SecurityGroupIds": [
      "sg-5943793c"
  ]
}
```

- Para obter detalhes da API, consulte SetSecurityGroupsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona o grupo de segurança “sg-07c3414abb8811cbd” ao balanceador de carga especificado.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Saída:

```
sg-07c3414abb8811cbd
```

- Para obter detalhes da API, consulte SetSecurityGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona o grupo de segurança “sg-07c3414abb8811cbd” ao balanceador de carga especificado.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Saída:

```
sg-07c3414abb8811cbd
```

- Para obter detalhes da API, consulte SetSecurityGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SetSubnets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SetSubnets`.

CLI

AWS CLI

Habilitar Zonas de Disponibilidade para um balanceador de carga

Este exemplo habilita a Zona de Disponibilidade da sub-rede especificada para o balanceador de carga especificado.

Comando:

```
aws elbv2 set-subnets --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 --
subnets subnet-8360a9e7 subnet-b7d581c0
```

Saída:

```
{
  "AvailabilityZones": [
      {
          "SubnetId": "subnet-8360a9e7",
          "ZoneName": "us-west-2a"
      },
      {
          "SubnetId": "subnet-b7d581c0",
          "ZoneName": "us-west-2b"
      }
  ]
}
```

- Para obter detalhes da API, consulte SetSubnetsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo modifica as sub-redes do balanceador de carga especificado.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Saída:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- Para obter detalhes da API, consulte SetSubnetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica as sub-redes do balanceador de carga especificado.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Saída:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- Para obter detalhes da API, consulte SetSubnetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para o Elastic Load Balancing - Versão 2 usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Elastic Load Balancing — Versão 2 com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Elastic Load Balancing Versão 2 ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie e gerencie um serviço resiliente usando um AWS SDK
- Introdução ao balanceamento de carga

Crie e gerencie um serviço resiliente usando um AWS SDK

Os exemplos de código a seguir mostram como criar um serviço da Web com carga balanceada que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de saúde atualizando AWS Systems Manager os parâmetros.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
```

```
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
```

```
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
```

```
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
```

```
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");
```

```
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
```

```
                            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
```

```
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");
```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
```

```
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
 a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
 granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
```

```
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
```

```
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
```

```
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
 List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
```

```
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
```

```
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
```

```
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
```

```
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
```

```
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
  calling computer.
```

```
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                       "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
```

```
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
```

```
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
```

```
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
```

```
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
```

```
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
```

```
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
```

```
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
```

```
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
```

```
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
```

```
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
```

```
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs

- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";
```

```
    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your
 account, we can clean up all AWS resources
                that were created for this demo.
```

```
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
```

```
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

       System.out.println("Press Enter when you're ready.");
       in.nextLine();
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Creating and populating a DynamoDB table named " +
tableName);
       Database database = new Database();
       database.createTable(tableName, fileName);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("""
               Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
               This script starts a Python web server defined in the `server.py`
script. The web server
               listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
               For demo purposes, this server is run as the root user. In
production, the best practice is to
               run a web server, such as Apache, with least-privileged
credentials.

               The template also defines an IAM policy that each instance uses
to assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
```

```
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
```

```
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
       autoScaler.terminateInstance(badInstanceId);

       System.out.println("""
               Even while the instance is terminating and the new instance is
starting, sending a GET
               request to the web service continues to get a successful
recommendation response because
               the load balancer routes requests to the healthy instances. After
the replacement instance
               starts and reports as healthy, it is included in the load
balancing rotation.
               Note that terminating and replacing an instance typically takes
several minutes, during which time you
               can see the changing health check status until the new instance
is running and healthy.
               """);

       demoChoices(loadBalancer);
       System.out.println(
               "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       demoChoices(loadBalancer);
       paramHelper.reset();
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
```

```
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());
```

```
                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
 balancer targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
```

```
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
```

```
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
```

```
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();
```

```
            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
```

```
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
```

```
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

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

```
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }
```

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

```
   // Verify this computer can successfully send a GET request to the load
balancer
   // endpoint.
   public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
       boolean success = false;
       int retries = 3;
       CloseableHttpClient httpClient = HttpClients.createDefault();

       // Create an HTTP GET request to the ELB.
       HttpGet httpGet = new HttpGet("http://" + elbDnsName);
       try {
           while ((!success) && (retries > 0)) {
               // Execute the request and get the response.
               HttpResponse response = httpClient.execute(httpGet);
               int statusCode = response.getStatusLine().getStatusCode();
               System.out.println("HTTP Status Code: " + statusCode);
               if (statusCode == 200) {
                   success = true;
               } else {
                   retries--;
                   System.out.println("Got connection error from load balancer
endpoint, retrying...");
                   TimeUnit.SECONDS.sleep(15);
               }
           }

       } catch (org.apache.http.conn.HttpHostConnectException e) {
           System.out.println(e.getMessage());
       }

       System.out.println("Status.." + success);
       return success;
   }

   /*
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
```

```
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
```

```
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
        return false;
```

```
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();
```

```
            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
```

```
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
```

```
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
```

```
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Criar etapas para implantar todos os recursos.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
```

```
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
```

```
    }),
    new ScenarioAction(
      "handleConfirmDeployment",
      (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
      "creatingTable",
      MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
      const client = new DynamoDBClient({});
      await client.send(
        new CreateTableCommand({
          TableName: NAMES.tableName,
          ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
          },
          AttributeDefinitions: [
            {
              AttributeName: "MediaType",
              AttributeType: "S",
            },
            {
              AttributeName: "ItemId",
              AttributeType: "N",
            },
          ],
          KeySchema: [
            {
              AttributeName: "MediaType",
              KeyType: "HASH",
            },
            {
              AttributeName: "ItemId",
              KeyType: "RANGE",
            },
          ],
        }),
      );
      await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
    }),
    new ScenarioOutput(
      "createdTable",
```

```
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
```

```
    new ScenarioOutput(
      "createdKeyPair",
      MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
    ),
    new ScenarioOutput(
      "creatingInstancePolicy",
      MESSAGES.creatingInstancePolicy.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      ),
    ),
    new ScenarioAction("createInstancePolicy", async (state) => {
      const client = new IAMClient({});
      const {
        Policy: { Arn },
      } = await client.send(
        new CreatePolicyCommand({
          PolicyName: NAMES.instancePolicyName,
          PolicyDocument: readFileSync(
            join(RESOURCES_PATH, "instance_policy.json"),
          ),
        }),
      );
      state.instancePolicyArn = Arn;
    }),
    new ScenarioOutput("createdInstancePolicy", (state) =>
      MESSAGES.createdInstancePolicy
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
    ),
    new ScenarioOutput(
      "creatingInstanceRole",
      MESSAGES.creatingInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      ),
    ),
    new ScenarioAction("createInstanceRole", () => {
      const client = new IAMClient({});
      return client.send(
        new CreateRoleCommand({
          RoleName: NAMES.instanceRoleName,
          AssumeRolePolicyDocument: readFileSync(
            join(ROOT, "assume-role-policy.json"),
```

```
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
```

```
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
```

```
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
```

```
        LaunchTemplate: {
          LaunchTemplateName: NAMES.launchTemplateName,
          Version: "$Default",
        },
        MinSize: 3,
        MaxSize: 3,
      }),
    ),
  );
}),
new ScenarioOutput(
  "createdAutoScalingGroup",
  /**
   * @param {{ availabilityZoneNames: string[] }} state
   */
  (state) =>
    MESSAGES.createdAutoScalingGroup
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
      .replace(
        "${AVAILABILITY_ZONE_NAMES}",
        state.availabilityZoneNames.join(", "),
      ),
),
new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
  type: "confirm",
}),
new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
new ScenarioAction("getVpc", async (state) => {
  const client = new EC2Client({});
  const { Vpcs } = await client.send(
    new DescribeVpcsCommand({
      Filters: [{ Name: "is-default", Values: ["true"] }],
    }),
  );
  state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
  MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
  const client = new EC2Client({});
  const { Subnets } = await client.send(
```

```
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
```

```
    }),
    new ScenarioOutput(
      "createdLoadBalancerTargetGroup",
      MESSAGES.createdLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      ),
    ),
    new ScenarioOutput(
      "creatingLoadBalancer",
      MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
    ),
    new ScenarioAction("createLoadBalancer", async (state) => {
      const client = new ElasticLoadBalancingV2Client({});
      const { LoadBalancers } = await client.send(
        new CreateLoadBalancerCommand({
          Name: NAMES.loadBalancerName,
          Subnets: state.subnets,
        }),
      );
      state.loadBalancerDns = LoadBalancers[0].DNSName;
      state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
      await waitUntilLoadBalancerAvailable(
        { client },
        { Names: [NAMES.loadBalancerName] },
      );
    }),
    new ScenarioOutput("createdLoadBalancer", (state) =>
      MESSAGES.createdLoadBalancer
        .replace("${LB_NAME}", NAMES.loadBalancerName)
        .replace("${DNS_NAME}", state.loadBalancerDns),
    ),
    new ScenarioOutput(
      "creatingListener",
      MESSAGES.creatingLoadBalancerListener
        .replace("${LB_NAME}", NAMES.loadBalancerName)
        .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
    ),
    new ScenarioAction("createListener", async (state) => {
      const client = new ElasticLoadBalancingV2Client({});
      const { Listeners } = await client.send(
        new CreateListenerCommand({
          LoadBalancerArn: state.loadBalancerArn,
          Protocol: state.targetGroupProtocol,
```

```
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
```

```
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
```

```
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
```

```
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Criar etapas para executar a demonstração.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
```

```
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
```

```
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
```

```
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
```

```
    ...statusSteps,
    new ScenarioInput(
      "brokenDependencyConfirmation",
      MESSAGES.demoBrokenDependencyConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("brokenDependency", async (state) => {
      if (!state.brokenDependencyConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        state.badTableName = `fake-table-${Date.now()}`;
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmTableNameKey,
            Value: state.badTableName,
            Overwrite: true,
            Type: "String",
          }),
        );
      }
    }),
    new ScenarioOutput("testBrokenDependency", (state) =>
      MESSAGES.demoTestBrokenDependency.replace(
        "${TABLE_NAME}",
        state.badTableName,
      ),
    ),
    ...statusSteps,
    new ScenarioInput(
      "staticResponseConfirmation",
      MESSAGES.demoStaticResponseConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("staticResponse", async (state) => {
      if (!state.staticResponseConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmFailureResponseKey,
            Value: "static",
            Overwrite: true,
```

```
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
```

```
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
        InstanceIds: [state.targetInstance.InstanceId],
        DocumentName: "AWS-RunShellScript",
        Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
```

```
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
```

```
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
```

```
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
```

```
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
```

```
    }),
  );

  return InstanceProfile;
}
```

Criar etapas para destruir todos os recursos.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
```

```
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
```

```
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
```

```
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
```

```
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
```

```
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
```

```
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
```

```
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
```

```
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
```

```
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
```

```
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
```

```
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
 state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
```

```
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
```

```
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
 DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
```

```
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)
```

```
        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
 from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
```

```
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
```

```
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()
```

```
        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()
```

```
        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
```

```
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
```

```
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
```

```
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
 definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
 {policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
```

```
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
```

```
        :param aws_managed_policies: A tuple of AWS-managed policy names to
attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
{role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
```

```
                                        use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn
```

```
    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.
```

```
        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
```

```
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
```

```
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
```

```
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
```

```
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
```

```
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
```

```
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
```

```
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
```

```
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
```

```
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
```

```
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
 "
                    "Ensure that your AWS IAM user or role has the correct
 permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
 parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

```
    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
```

```
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
```

```
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
```

```
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
 {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
```

```
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
```

```
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
```

```
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.
```

```
        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer
```

```
    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
```

```
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
```

```
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
```

```
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
 parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
```

```
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
```

```
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Crie uma classe que envolva as ações do Systems Manager.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
```

```
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer

- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Introdução ao balanceamento de carga

O exemplo de código a seguir mostra como:

- Criar um Application Load Balancer.
- Criar um grupo de destino.
- Criar um receptor.
- Verificar sua configuração.
- Adicionar um receptor HTTPS (opcional).
- Adicionar roteamento baseado em caminho (opcional).
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Elastic Load Balancing Getting Started Script - v2
# This script creates an Application Load Balancer with HTTP listener and target
 group

set -euo pipefail

# Set up logging
LOG_FILE="elb-script-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Elastic Load Balancing setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check AWS CLI command success
check_command() {
    local output="$1"
    if [[ -z "$output" ]] || [[ "$output" == "None" ]]; then
        handle_error "AWS CLI command returned empty or invalid output"
    fi
}

# Function to validate ARN format
```

```
validate_arn() {
    local arn="$1"
    if [[ ! "$arn" =~ ^arn:aws:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$ ]]; then
        handle_error "Invalid ARN format: $arn"
    fi
}

# Function to validate security group ID
validate_security_group_id() {
    local sg_id="$1"
    if [[ ! "$sg_id" =~ ^sg-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid security group ID format: $sg_id"
    fi
}

# Function to validate VPC ID
validate_vpc_id() {
    local vpc_id="$1"
    if [[ ! "$vpc_id" =~ ^vpc-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid VPC ID format: $vpc_id"
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    if [ -n "${LISTENER_ARN:-}" ]; then
        echo "Deleting listener: $LISTENER_ARN"
        aws elbv2 delete-listener --listener-arn "$LISTENER_ARN" 2>/dev/null ||
 true
    fi

    if [ -n "${LOAD_BALANCER_ARN:-}" ]; then
        echo "Deleting load balancer: $LOAD_BALANCER_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LOAD_BALANCER_ARN"
 2>/dev/null || true

        # Wait for load balancer to be deleted before deleting target group
        echo "Waiting for load balancer to be deleted..."
        aws elbv2 wait load-balancers-deleted --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null || true
    fi
```

```
    if [ -n "${TARGET_GROUP_ARN:-}" ]; then
        echo "Deleting target group: $TARGET_GROUP_ARN"
        aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN" 2>/
dev/null || true
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting 30 seconds before deleting security group to ensure all
 dependencies are removed..."
        sleep 30

        echo "Deleting security group: $SECURITY_GROUP_ID"
        local sg_delete_output
        sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)

        local retry_count=0
        local max_retries=5
        while echo "$sg_delete_output" | grep -i "DependencyViolation" > /dev/
null && [ $retry_count -lt $max_retries ]; do
            retry_count=$((retry_count+1))
            echo "Security group still has dependencies. Retrying in 30
 seconds... (Attempt $retry_count of $max_retries)"
            sleep 30
            sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)
        done

        if echo "$sg_delete_output" | grep -i "error" > /dev/null; then
            echo "WARNING: Could not delete security group: $SECURITY_GROUP_ID"
 >&2
            echo "You may need to delete it manually using: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        else
            echo "Security group deleted successfully."
        fi
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
RESOURCE_PREFIX="elb-demo-${RANDOM_ID}"

# Verify AWS CLI is available
```

```
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Step 1: Verify AWS CLI support for Elastic Load Balancing
echo "Verifying AWS CLI support for Elastic Load Balancing..."
if ! aws elbv2 help > /dev/null 2>&1; then
    handle_error "AWS CLI does not support elbv2 commands. Please update your AWS
 CLI."
fi

# Step 2: Get VPC ID and subnet information
echo "Retrieving VPC information..."
VPC_INFO=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 "Vpcs[0].VpcId" --output text 2>/dev/null || echo "")
check_command "$VPC_INFO"
VPC_ID="$VPC_INFO"
validate_vpc_id "$VPC_ID"
echo "Using VPC: $VPC_ID"

# Get two subnets from different Availability Zones
echo "Retrieving subnet information..."
SUBNET_INFO=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[0:2].SubnetId" --output text 2>/dev/null || echo "")
check_command "$SUBNET_INFO"

# Convert space-separated list to array
read -r -a SUBNETS <<< "$SUBNET_INFO"
if [ ${#SUBNETS[@]} -lt 2 ]; then
    handle_error "Need at least 2 subnets in different Availability Zones. Found:
 ${#SUBNETS[@]}"
fi

echo "Using subnets: ${SUBNETS[0]} and ${SUBNETS[1]}"

# Step 3: Create a security group for the load balancer
echo "Creating security group for the load balancer..."
SG_INFO=$(aws ec2 create-security-group \
    --group-name "${RESOURCE_PREFIX}-sg" \
    --description "Security group for ELB demo" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=elastic-load-
balancing-gs}]' \
```

```
    --query "GroupId" --output text 2>/dev/null || echo "")
check_command "$SG_INFO"
SECURITY_GROUP_ID="$SG_INFO"
validate_security_group_id "$SECURITY_GROUP_ID"
echo "Created security group: $SECURITY_GROUP_ID"

# Add inbound rule to allow HTTP traffic
echo "Adding inbound rule to allow HTTP traffic..."
aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 80 \
    --cidr "0.0.0.0/0" > /dev/null 2>&1 || handle_error "Failed to authorize
 security group ingress"

echo "WARNING: Security group allows HTTP from 0.0.0.0/0. In production, restrict
 to specific IP addresses." >&2

# Step 4: Create the load balancer
echo "Creating Application Load Balancer..."
LB_INFO=$(aws elbv2 create-load-balancer \
    --name "${RESOURCE_PREFIX}-lb" \
    --subnets "${SUBNETS[0]}" "${SUBNETS[1]}" \
    --security-groups "$SECURITY_GROUP_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "LoadBalancers[0].LoadBalancerArn" --output text 2>/dev/null || echo
 "")
check_command "$LB_INFO"
LOAD_BALANCER_ARN="$LB_INFO"
validate_arn "$LOAD_BALANCER_ARN"
echo "Created load balancer: $LOAD_BALANCER_ARN"

# Wait for the load balancer to be active
echo "Waiting for load balancer to become active..."
if ! aws elbv2 wait load-balancer-available --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null; then
    handle_error "Load balancer did not reach active state within timeout period"
fi

# Step 5: Create a target group
echo "Creating target group..."
TG_INFO=$(aws elbv2 create-target-group \
    --name "${RESOURCE_PREFIX}-targets" \
```

```
    --protocol HTTP \
    --port 80 \
    --vpc-id "$VPC_ID" \
    --target-type instance \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "TargetGroups[0].TargetGroupArn" --output text 2>/dev/null || echo
 "")
check_command "$TG_INFO"
TARGET_GROUP_ARN="$TG_INFO"
validate_arn "$TARGET_GROUP_ARN"
echo "Created target group: $TARGET_GROUP_ARN"

# Step 6: Find EC2 instances to register as targets
echo "Looking for available EC2 instances to register as targets..."
INSTANCES=$(aws ec2 describe-instances \
    --filters "Name=vpc-id,Values=$VPC_ID" "Name=instance-state-
name,Values=running" \
    --query "Reservations[*].Instances[*].InstanceId" --output text 2>/dev/null
 || echo "")

# Convert space-separated list to array
read -r -a INSTANCE_IDS <<< "$INSTANCES"

if [ ${#INSTANCE_IDS[@]} -eq 0 ]; then
    echo "No running instances found in VPC $VPC_ID."
    echo "You will need to register targets manually after launching instances."
else
    # Step 7: Register targets with the target group (up to 2 instances)
    echo "Registering targets with the target group..."
    target_args=()
    for i in "${!INSTANCE_IDS[@]}"; do
        if [ "$i" -lt 2 ]; then
            target_args+=("Id=${INSTANCE_IDS[$i]}")
        fi
    done

    if [ ${#target_args[@]} -gt 0 ]; then
        if aws elbv2 register-targets \
            --target-group-arn "$TARGET_GROUP_ARN" \
            --targets "${target_args[@]}" 2>/dev/null; then
            echo "Registered instances: ${target_args[*]}"
        else
            handle_error "Failed to register targets"
```

```
        fi
    fi
fi

# Step 8: Create a listener
echo "Creating HTTP listener..."
LISTENER_INFO=$(aws elbv2 create-listener \
    --load-balancer-arn "$LOAD_BALANCER_ARN" \
    --protocol HTTP \
    --port 80 \
    --default-actions Type=forward,TargetGroupArn="$TARGET_GROUP_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "Listeners[0].ListenerArn" --output text 2>/dev/null || echo "")
check_command "$LISTENER_INFO"
LISTENER_ARN="$LISTENER_INFO"
validate_arn "$LISTENER_ARN"
echo "Created listener: $LISTENER_ARN"

# Step 9: Verify target health
echo "Verifying target health..."
aws elbv2 describe-target-health --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/
null || true

# Display load balancer DNS name
LB_DNS=$(aws elbv2 describe-load-balancers \
    --load-balancer-arns "$LOAD_BALANCER_ARN" \
    --query "LoadBalancers[0].DNSName" --output text 2>/dev/null || echo "")
check_command "$LB_DNS"

echo ""
echo "================================================"
echo "SETUP COMPLETE"
echo "================================================"
echo "Load Balancer DNS Name: $LB_DNS"
echo ""
echo "Resources created:"
echo "- Load Balancer: $LOAD_BALANCER_ARN"
echo "- Target Group: $TARGET_GROUP_ARN"
echo "- Listener: $LISTENER_ARN"
echo "- Security Group: $SECURITY_GROUP_ID"
echo ""

# Prompt for cleanup confirmation
```

```
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
read -p "Do you want to clean up all created resources? (y/n): " -r
 CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
    echo "To clean up later, run the following commands:"
    echo "aws elbv2 delete-listener --listener-arn $LISTENER_ARN"
    echo "aws elbv2 delete-load-balancer --load-balancer-arn $LOAD_BALANCER_ARN"
    echo "aws elbv2 wait load-balancers-deleted --load-balancer-arns
 $LOAD_BALANCER_ARN"
    echo "aws elbv2 delete-target-group --target-group-arn $TARGET_GROUP_ARN"
    echo "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateListener
  - CreateLoadBalancer
  - CreateSecurityGroup
  - CreateTargetGroup
  - DeleteListener
  - DeleteLoadBalancer
  - DeleteSecurityGroup
  - DeleteTargetGroup
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetHealth

- DescribeVpcs
- Help
- RegisterTargets
- Wait

Exemplos de código para ElastiCache usar AWS SDKs

Os exemplos de código a seguir mostram como usar a Amazon ElastiCache com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- ElastiCache Guia do usuário — Mais informações sobre ElastiCache.
- ElastiCache Referência da API — Detalhes sobre todas as ElastiCache ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para ElastiCache usar AWS SDKs
  - Introdução ao armazenamento em cache em memória

Cenários para ElastiCache usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns ElastiCache com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas ElastiCache ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução ao armazenamento em cache em memória

Introdução ao armazenamento em cache em memória

O exemplo de código a seguir mostra como:

- Configurar grupo de segurança para ElastiCache acesso
- Criar um cache sem servidor do Valkey.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ElastiCache Getting Started Script
# This script creates a Valkey serverless cache, configures security groups,
# and demonstrates how to connect to and use the cache.

set -uo pipefail

# Set up logging
LOG_FILE="elasticache_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting ElastiCache tutorial script. Logging to $LOG_FILE"
echo "============================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
```

```
    echo "Resources created:"
    if [ -n "${CACHE_NAME:-}" ]; then
        echo "- ElastiCache serverless cache: $CACHE_NAME"
    fi
    if [ -n "${SG_RULE_6379:-}" ] || [ -n "${SG_RULE_6380:-}" ]; then
        echo "- Security group rules for ports 6379 and 6380"
    fi
    echo "Please clean up these resources manually."
    exit 1
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CACHE_NAME="valkey-cache-${RANDOM_ID}"

echo "Using cache name: $CACHE_NAME"

# Step 1: Set up security group for ElastiCache access
echo "Step 1: Setting up security group for ElastiCache access..."

# Get default security group ID
echo "Getting default security group ID..."
SG_ID=$(aws ec2 describe-security-groups \
  --filters Name=group-name,Values=default \
  --query "SecurityGroups[0].GroupId" \
  --output text 2>/dev/null || echo "")

if [[ -z "$SG_ID" || "$SG_ID" == "None" ]]; then
    handle_error "Failed to get default security group ID"
fi

echo "Default security group ID: $SG_ID"

# Add inbound rule for port 6379
```

```
echo "Adding inbound rule for port 6379..."
SG_RULE_6379=""
if SG_RULE_6379=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6379 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379"
 == *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    fi
else
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379"
 == *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    else
        handle_error "Failed to add security group rule for port 6379:
 $SG_RULE_6379"
    fi
fi

# Add inbound rule for port 6380
echo "Adding inbound rule for port 6380..."
SG_RULE_6380=""
if SG_RULE_6380=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6380 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380"
 == *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    fi
else
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380"
 == *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
```

```
        SG_RULE_6380="existing"
    else
        handle_error "Failed to add security group rule for port 6380:
 $SG_RULE_6380"
    fi
fi

echo "Security group rules added successfully."
echo ""
echo "##  SECURITY WARNING: The security group rules created allow access from
 any IP address (0.0.0.0/0)."
echo "This is NOT RECOMMENDED for production environments. For production,"
echo "you should restrict access to specific IP ranges or security groups."
echo "Update the CIDR blocks in this script before using in production."
echo ""

# Step 2: Create a Valkey serverless cache
echo "Step 2: Creating Valkey serverless cache..."
if ! CREATE_RESULT=$(aws elasticache create-serverless-cache \
  --serverless-cache-name "$CACHE_NAME" \
  --engine valkey \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-elasticache-gs
 2>&1); then
    handle_error "Failed to create serverless cache: $CREATE_RESULT"
fi

echo "Cache creation initiated. Waiting for cache to become available..."

# Step 3: Check the status of the cache creation
echo "Step 3: Checking cache status..."

# Wait for the cache to become active
MAX_ATTEMPTS=30
ATTEMPT=1
CACHE_STATUS=""

while [[ $ATTEMPT -le $MAX_ATTEMPTS ]]; do
    echo "Checking cache status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    if ! DESCRIBE_RESULT=$(aws elasticache describe-serverless-caches \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        handle_error "Failed to describe serverless cache: $DESCRIBE_RESULT"
    fi
```

```
    # Extract status using jq for reliable JSON parsing
    if command -v jq &> /dev/null; then
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | jq -r
 '.ServerlessCaches[0].Status // "UNKNOWN"' 2>/dev/null || echo "")
    else
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | grep -o '"Status": "[^"]*"' |
 awk -F'"' '{print $4}' | head -n 1)
    fi

    echo "Current status: $CACHE_STATUS"

    if [[ "${CACHE_STATUS,,}" == "available" ]]; then
        echo "Cache is now available!"
        break
    elif [[ "${CACHE_STATUS,,}" == "create-failed" ]]; then
        handle_error "Cache creation failed. Please check the AWS console for
 details."
    fi

    if [[ $ATTEMPT -lt $MAX_ATTEMPTS ]]; then
        echo "Waiting 30 seconds..."
        sleep 30
    fi

    ((ATTEMPT++))
done

if [[ "${CACHE_STATUS,,}" != "available" ]]; then
    handle_error "Cache did not become available within the expected time. Last
 status: $CACHE_STATUS"
fi

# Step 4: Find your cache endpoint
echo "Step 4: Getting cache endpoint..."
if ! ENDPOINT=$(aws elasticache describe-serverless-caches \
  --serverless-cache-name "$CACHE_NAME" \
  --query "ServerlessCaches[0].Endpoint.Address" \
  --output text 2>&1); then
    handle_error "Failed to get cache endpoint: $ENDPOINT"
fi

if [[ -z "$ENDPOINT" || "$ENDPOINT" == "None" ]]; then
    handle_error "Failed to get cache endpoint"
fi
```

```
echo "Cache endpoint: $ENDPOINT"

# Step 5: Instructions for connecting to the cache
echo ""
echo "============================================================"
echo "Your Valkey serverless cache has been successfully created!"
echo "Cache Name: $CACHE_NAME"
echo "Endpoint: $ENDPOINT"
echo "============================================================"
echo ""
echo "To connect to your cache from an EC2 instance, follow these steps:"
echo ""
echo "1. Install valkey-cli on your EC2 instance:"
echo "   sudo amazon-linux-extras install epel -y"
echo "   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y"
echo "   wget https://github.com/valkey-io/valkey/archive/refs/tags/8.0.0.tar.gz"
echo "   tar xvzf 8.0.0.tar.gz"
echo "   cd valkey-8.0.0"
echo "   make BUILD_TLS=yes"
echo ""
echo "2. Connect to your cache using valkey-cli:"
echo "   src/valkey-cli -h $ENDPOINT --tls -p 6379"
echo ""
echo "3. Once connected, you can run commands like:"
echo "   set mykey \"Hello ElastiCache\""
echo "   get mykey"
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- ElastiCache serverless cache: $CACHE_NAME"
if [[ "${SG_RULE_6379:-}" != "existing" ]] || [[ "${SG_RULE_6380:-}" !=
 "existing" ]]; then
    echo "- Security group rules for ports 6379 and 6380"
fi
echo ""
echo "Proceeding with cleanup..."

CLEANUP_CHOICE="y"
```

```
if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Step 7: Delete the cache
    echo "Deleting serverless cache $CACHE_NAME..."
    if ! DELETE_RESULT=$(aws elasticache delete-serverless-cache \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        echo "WARNING: Failed to delete serverless cache: $DELETE_RESULT"
        echo "Please delete the cache manually from the AWS console."
    else
        echo "Cache deletion initiated. This may take several minutes to
 complete."
    fi

    # Only attempt to remove security group rules if we created them
    if [[ "${SG_RULE_6379:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6379..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6379 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6379"
        fi
    fi

    if [[ "${SG_RULE_6380:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6380..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6380 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6380"
        fi
    fi

    echo "Cleanup completed."
fi

echo ""
echo "Script completed. See $LOG_FILE for the full log."
```

```
echo "============================================================"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateServerlessCache
  - DeleteServerlessCache
  - DescribeSecurityGroups
  - DescribeServerlessCaches
  - RevokeSecurityGroupIngress

Exemplos de código para MediaStore usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Elemental MediaStore com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- MediaStore Guia do usuário — Mais informações sobre MediaStore.
- MediaStore Referência da API — Detalhes sobre todas as MediaStore ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de MediaStore uso AWS SDKs
  - Ações para MediaStore usar AWS SDKs
    - Use CreateContainercom um AWS SDK ou CLI
    - Use DeleteContainercom um AWS SDK ou CLI
    - Use DeleteObjectcom um AWS SDK

- Use DescribeContainercom um AWS SDK ou CLI
- Use GetObjectcom um AWS SDK ou CLI
- Use ListContainerscom um AWS SDK ou CLI
- Use PutObjectcom um AWS SDK ou CLI

Exemplos básicos de MediaStore uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Elemental MediaStore com AWS SDKs.

Exemplos

- Ações para MediaStore usar AWS SDKs
  - Use CreateContainercom um AWS SDK ou CLI
  - Use DeleteContainercom um AWS SDK ou CLI
  - Use DeleteObjectcom um AWS SDK
  - Use DescribeContainercom um AWS SDK ou CLI
  - Use GetObjectcom um AWS SDK ou CLI
  - Use ListContainerscom um AWS SDK ou CLI
  - Use PutObjectcom um AWS SDK ou CLI

Ações para MediaStore usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar MediaStore ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Elemental MediaStore.

Exemplos

- Use CreateContainercom um AWS SDK ou CLI
- Use DeleteContainercom um AWS SDK ou CLI
- Use DeleteObjectcom um AWS SDK
- Use DescribeContainercom um AWS SDK ou CLI

- Use GetObjectcom um AWS SDK ou CLI
- Use ListContainerscom um AWS SDK ou CLI
- Use PutObjectcom um AWS SDK ou CLI

Use **`CreateContainer`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateContainer`.

CLI

AWS CLI

Para criar um contêiner

O exemplo `create-container` a seguir cria um contêiner novo e vazio.

```
aws mediastore create-container --container-name ExampleContainer
```

Saída:

```
{
    "Container": {
        "AccessLoggingEnabled": false,
        "CreationTime": 1563557265,
        "Name": "ExampleContainer",
        "Status": "CREATING",
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
    }
}
```

Para obter mais informações, consulte Criação de um contêiner no Guia do MediaStore usuário do AWS Elemental.

- Para obter detalhes da API, consulte CreateContainerem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

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

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateContainera Referência AWS SDK for Java 2.x da API.

Use **`DeleteContainer`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteContainer`.

CLI

AWS CLI

Para excluir um contêiner

O exemplo `delete-container` a seguir exclui o contêiner especificado. Você só pode excluir um contêiner se ele não tiver objetos.

```
aws mediastore delete-container \
    --container-name=ExampleLiveDemo
```

Este comando não produz saída.

Para obter mais informações, consulte Excluindo um contêiner no Guia do MediaStore usuário do AWS Elemental.

- Para obter detalhes da API, consulte DeleteContainerem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

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

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();
```

```
            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteContainera Referência AWS SDK for Java 2.x da API.

Use **`DeleteObject`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteObject`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
```

```
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName>

                Where:
                   completePath - The path (including the container) of the item
 to delete.
                   containerName - The name of the container.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
```

```
                .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData,
 String completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest =
 DeleteObjectRequest.builder()
                    .path(completePath)
                    .build();

            mediaStoreData.deleteObject(deleteObjectRequest);

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

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- Para obter detalhes da API, consulte DeleteObjecta Referência AWS SDK for Java 2.x da API.

Use **`DescribeContainer`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeContainer`.

CLI

AWS CLI

Como visualizar os detalhes de um contêiner

O exemplo `describe-container` a seguir exibe os detalhes do contêiner especificado.

```
aws mediastore describe-container \
    --container-name ExampleContainer
```

Saída:

```
{
    "Container": {
        "CreationTime": 1563558086,
        "AccessLoggingEnabled": false,
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
        "Status": "ACTIVE",
        "Name": "ExampleContainer",
        "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com"
    }
}
```

Para obter mais informações, consulte Visualizando os detalhes de um contêiner no Guia do MediaStore usuário do AWS Elemental.

- Para obter detalhes da API, consulte DescribeContainerem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

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

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

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
 containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
 containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
 DescribeContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            DescribeContainerResponse containerResponse =
 mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
 containerResponse.container().name());
            System.out.println("The container ARN is " +
 containerResponse.container().arn());
            return containerResponse.container().status().toString();

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

- Para obter detalhes da API, consulte DescribeContainera Referência AWS SDK for Java 2.x da API.

Use **`GetObject`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetObject`.

CLI

AWS CLI

Como fazer download de um objeto

O exemplo `get-object` a seguir faz o download de um objeto no endpoint especificado.

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path=/folder_name/README.md README.md
```

Saída:

```
{
    "ContentLength": "2307346",
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",
    "StatusCode": 200
}
```

Para fazer o download de parte de um objeto

O exemplo `get-object` a seguir faz o download de um objeto no endpoint especificado.

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path /folder_name/README.md \
    --range="bytes=0-100" README2.md
```

Saída:

```
{
    "StatusCode": 206,
    "ContentRange": "bytes 0-100/2307346",
    "ContentLength": "101",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentType": "image/jpeg",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

Para obter mais informações, consulte Baixar um objeto no Guia do MediaStore usuário do AWS Elemental.

- Para obter detalhes da API, consulte GetObjectem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

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

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

                Usage:    <completePath> <containerName> <savePath>

                Where:
                   completePath - The path of the object in the container (for
 example, Videos5/sampleVideo.mp4).
                   containerName - The name of the container.
                   savePath - The path on the local drive where the file is
 saved, including the file name (for example, C:/AWS/myvid.mp4).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }

    public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
 completePath, String savePath) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .path(completePath)
                    .build();

            // Write out the data to a file.
```

```
            ResponseInputStream<GetObjectResponse> data =
 mediaStoreData.getObject(objectRequest);
            byte[] buffer = new byte[data.available()];
            data.read(buffer);

            File targetFile = new File(savePath);
            OutputStream outStream = new FileOutputStream(targetFile);
            outStream.write(buffer);
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK for Java 2.x da API.

Use **`ListContainers`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListContainers`.

CLI

AWS CLI

Como visualizar uma lista de contêineres

O exemplo `list-containers` a seguir da lista todos os contêineres associados à sua conta.

```
aws mediastore list-containers
```

Saída:

```
{
    "Containers": [
        {
            "CreationTime": 1505317931,
            "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
            "AccessLoggingEnabled": false,
            "Name": "ExampleLiveDemo"
        },
        {
            "CreationTime": 1506528818,
            "Endpoint": "https://fffggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
            "AccessLoggingEnabled": false,
            "Name": "ExampleContainer"
        }
    ]
}
```

Para obter mais informações, consulte Visualizando uma lista de contêineres no Guia do MediaStore usuário do AWS Elemental.

- Para obter detalhes da API, consulte ListContainersem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

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

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
```

```
            ListContainersResponse containersResponse =
 mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

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

- Para obter detalhes da API, consulte ListContainersa Referência AWS SDK for Java 2.x da API.

Use **PutObject**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutObject`.

CLI

AWS CLI

Como fazer upload de um objeto

O exemplo `put-object` a seguir faz upload de um objeto no contêiner especificado. É possível especificar um caminho de pasta no qual o objeto será salvo dentro do contêiner. Se a pasta já existir, o AWS Elemental MediaStore armazena o objeto na pasta. Se a pasta não existir, o serviço a cria e, em seguida, armazena o objeto na pasta.

```
aws mediastore-data put-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --body README.md \
    --path /folder_name/README.md \
    --cache-control "max-age=6, public" \
    --content-type binary/octet-stream
```

Saída:

```
{
    "ContentSHA256":
 "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",
    "StorageClass": "TEMPORAL",
    "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"
}
```

Para obter mais informações, consulte Carregando um objeto no Guia do MediaStore usuário do AWS Elemental.

- Para obter detalhes da API, consulte PutObjectem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = """

                To run this example, supply the name of a container, a file
 location to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .path(completePath)
                    .contentType("video/mp4")
```

```
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

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

    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK for Java 2.x da API.

Exemplos de código para o Amazon EMR usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon EMR com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia de gerenciamento do Amazon EMR: mais informações sobre o Amazon EMR.
- Referência da API Amazon EMR: detalhes sobre todas as ações disponíveis do Amazon EMR.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon EMR usando AWS SDKs
  - Ações para o Amazon EMR usando AWS SDKs
    - Use AddJobFlowStepscom um AWS SDK
    - Use DescribeClustercom um AWS SDK ou CLI
    - Use DescribeStepcom um AWS SDK ou CLI
    - Use ListStepscom um AWS SDK ou CLI
    - Use RunJobFlowcom um AWS SDK
    - Use TerminateJobFlowscom um AWS SDK
- Cenários para o Amazon EMR usando AWS SDKs
  - Crie um cluster Amazon EMR de curta duração e execute uma etapa usando um AWS SDK
  - Introdução aos clusters de processamento de big data
  - Execute um script de shell para instalar bibliotecas em instâncias do Amazon EMR usando um AWS SDK

Exemplos básicos para o Amazon EMR usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon EMR com AWS SDKs.

Exemplos

- Ações para o Amazon EMR usando AWS SDKs

- Use AddJobFlowStepscom um AWS SDK
- Use DescribeClustercom um AWS SDK ou CLI
- Use DescribeStepcom um AWS SDK ou CLI
- Use ListStepscom um AWS SDK ou CLI
- Use RunJobFlowcom um AWS SDK
- Use TerminateJobFlowscom um AWS SDK

Ações para o Amazon EMR usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon EMR com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon EMR e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon EMR usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon EMR.

Exemplos

- Use AddJobFlowStepscom um AWS SDK
- Use DescribeClustercom um AWS SDK ou CLI
- Use DescribeStepcom um AWS SDK ou CLI
- Use ListStepscom um AWS SDK ou CLI
- Use RunJobFlowcom um AWS SDK
- Use TerminateJobFlowscom um AWS SDK

Use **`AddJobFlowSteps`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `AddJobFlowSteps`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione uma etapa do Spark, que será executada pelo cluster assim que for adicionada.

```
def add_step(cluster_id, name, script_uri, script_args, emr_client):
    """
    Adds a job step to the specified cluster. This example adds a Spark
    step, which is run by the cluster as soon as it is added.

    :param cluster_id: The ID of the cluster.
    :param name: The name of the step.
    :param script_uri: The URI where the Python script is stored.
    :param script_args: Arguments to pass to the Python script.
    :param emr_client: The Boto3 EMR client object.
    :return: The ID of the newly added step.
    """
    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id,
            Steps=[
                {
                    "Name": name,
                    "ActionOnFailure": "CONTINUE",
                    "HadoopJarStep": {
                        "Jar": "command-runner.jar",
                        "Args": [
                            "spark-submit",
                            "--deploy-mode",
                            "cluster",
                            script_uri,
                            *script_args,
                        ],
                    },
                }
            ],
```

```
        )
        step_id = response["StepIds"][0]
        logger.info("Started step with ID %s", step_id)
    except ClientError:
        logger.exception("Couldn't start step %s with URI %s.", name, script_uri)
        raise
    else:
        return step_id
```

Execute um comando do Amazon EMR File System (EMRFS) como uma etapa de trabalho em um cluster. Isso pode ser usado para automatizar comandos do EMRFS em um cluster em vez de executar comandos manualmente por meio de uma conexão SSH.

```
import boto3
from botocore.exceptions import ClientError


def add_emrfs_step(command, bucket_url, cluster_id, emr_client):
    """
    Add an EMRFS command as a job flow step to an existing cluster.

    :param command: The EMRFS command to run.
    :param bucket_url: The URL of a bucket that contains tracking metadata.
    :param cluster_id: The ID of the cluster to update.
    :param emr_client: The Boto3 Amazon EMR client object.
    :return: The ID of the added job flow step. Status can be tracked by calling
             the emr_client.describe_step() function.
    """
    job_flow_step = {
        "Name": "Example EMRFS Command Step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
            "Jar": "command-runner.jar",
            "Args": ["/usr/bin/emrfs", command, bucket_url],
        },
    }

    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id, Steps=[job_flow_step]
```

```
        )
        step_id = response["StepIds"][0]
        print(f"Added step {step_id} to cluster {cluster_id}.")
    except ClientError:
        print(f"Couldn't add a step to cluster {cluster_id}.")
        raise
    else:
        return step_id


def usage_demo():
    emr_client = boto3.client("emr")
    # Assumes the first waiting cluster has EMRFS enabled and has created
 metadata
    # with the default name of 'EmrFSMetadata'.
    cluster = emr_client.list_clusters(ClusterStates=["WAITING"])["Clusters"][0]
    add_emrfs_step(
        "sync", "s3://elasticmapreduce/samples/cloudfront", cluster["Id"],
 emr_client
    )


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a AddJobFlowStepsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Build args list for Spark submit
```

```
      DATA lt_args TYPE /aws1/cl_emrxmlstringlist_w=>tt_xmlstringlist.
      APPEND NEW /aws1/cl_emrxmlstringlist_w( 'spark-submit' ) TO lt_args.
      APPEND NEW /aws1/cl_emrxmlstringlist_w( '--deploy-mode' ) TO lt_args.
      APPEND NEW /aws1/cl_emrxmlstringlist_w( 'cluster' ) TO lt_args.
      APPEND NEW /aws1/cl_emrxmlstringlist_w( iv_script_uri ) TO lt_args.
      APPEND LINES OF it_script_args TO lt_args.

      " Create step configuration
      DATA(lo_hadoop_jar_step) = NEW /aws1/cl_emrhadoopjarstepcfg(
        iv_jar = 'command-runner.jar'
        it_args = lt_args
      ).

      DATA(lo_step_config) = NEW /aws1/cl_emrstepconfig(
        iv_name = iv_name
        iv_actiononfailure = 'CONTINUE'
        io_hadoopjarstep = lo_hadoop_jar_step
      ).

      DATA lt_steps TYPE /aws1/cl_emrstepconfig=>tt_stepconfiglist.
      APPEND lo_step_config TO lt_steps.

      DATA(lo_result) = lo_emr->addjobflowsteps(
        iv_jobflowid = iv_cluster_id
        it_steps = lt_steps
      ).

      " Get first step ID
      DATA(lt_step_ids) = lo_result->get_stepids( ).
      READ TABLE lt_step_ids INDEX 1 INTO DATA(lo_step_id_obj).
      IF sy-subrc = 0.
        ov_step_id = lo_step_id_obj->get_value( ).
        MESSAGE |Step added with ID { ov_step_id }| TYPE 'I'.
      ENDIF.
    CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
      DATA(lv_error) = lo_internal_error->if_message~get_text( ).
      MESSAGE lv_error TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a AddJobFlowStepsreferência da API AWS SDK for SAP ABAP.

Use **`DescribeCluster`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução aos clusters de processamento de big data

CLI

AWS CLI

Comando:

```
aws emr describe-cluster --cluster-id j-XXXXXXXX
```

Saída:

```
For release-label based uniform instance groups cluster:

        {
            "Cluster": {
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1436475075.199,
                        "CreationDateTime": 1436474656.563,
                    },
                    "State": "WAITING",
                    "StateChangeReason": {
                        "Message": "Waiting for steps to run"
                    }
                },
                "Ec2InstanceAttributes": {
                    "ServiceAccessSecurityGroup": "sg-xxxxxxxx",
                    "EmrManagedMasterSecurityGroup": "sg-xxxxxxxx",
                    "IamInstanceProfile": "EMR_EC2_DefaultRole",
                    "Ec2KeyName": "myKey",
                    "Ec2AvailabilityZone": "us-east-1c",
                    "EmrManagedSlaveSecurityGroup": "sg-yyyyyyyyy"
                },
                "Name": "My Cluster",
                "ServiceRole": "EMR_DefaultRole",
```

```
                "Tags": [],
                "TerminationProtected": true,
                "UnhealthyNodeReplacement": true,
                "ReleaseLabel": "emr-4.0.0",
                "NormalizedInstanceHours": 96,
                "InstanceGroups": [
                    {
                        "RequestedInstanceCount": 2,
                        "Status": {
                            "Timeline": {
                                "ReadyDateTime": 1436475074.245,
                                "CreationDateTime": 1436474656.564,
                                "EndDateTime": 1436638158.387
                            },
                            "State": "RUNNING",
                            "StateChangeReason": {
                                "Message": "",
                            }
                        },
                        "Name": "CORE",
                        "InstanceGroupType": "CORE",
                        "Id": "ig-YYYYYYY",
                        "Configurations": [],
                        "InstanceType": "m3.large",
                        "Market": "ON_DEMAND",
                        "RunningInstanceCount": 2
                    },
                    {
                        "RequestedInstanceCount": 1,
                        "Status": {
                            "Timeline": {
                                "ReadyDateTime": 1436475074.245,
                                "CreationDateTime": 1436474656.564,
                                "EndDateTime": 1436638158.387
                            },
                            "State": "RUNNING",
                            "StateChangeReason": {
                                "Message": "",
                            }
                        },
                        "Name": "MASTER",
                        "InstanceGroupType": "MASTER",
                        "Id": "ig-XXXXXXXXX",
                        "Configurations": [],
```

```
                        "InstanceType": "m3.large",
                        "Market": "ON_DEMAND",
                        "RunningInstanceCount": 1
                    }
                ],
                "Applications": [
                    {
                        "Name": "Hadoop"
                    }
                ],
                "VisibleToAllUsers": true,
                "BootstrapActions": [],
                "MasterPublicDnsName":
 "ec2-54-147-144-78.compute-1.amazonaws.com",
                "AutoTerminate": false,
                "Id": "j-XXXXXXXX",
                "Configurations": [
                    {
                        "Properties": {
                            "fs.s3.consistent.retryPeriodSeconds": "20",
                            "fs.s3.enableServerSideEncryption": "true",
                            "fs.s3.consistent": "false",
                            "fs.s3.consistent.retryCount": "2"
                        },
                        "Classification": "emrfs-site"
                    }
                ]
            }
        }


For release-label based instance fleet cluster:
{
    "Cluster": {
        "Status": {
            "Timeline": {
                "ReadyDateTime": 1487897289.705,
                "CreationDateTime": 1487896933.942
            },
            "State": "WAITING",
            "StateChangeReason": {
                "Message": "Waiting for steps to run"
            }
        },
```

```
        "Ec2InstanceAttributes": {
            "EmrManagedMasterSecurityGroup": "sg-xxxxx",
            "RequestedEc2AvailabilityZones": [],
            "RequestedEc2SubnetIds": [],
            "IamInstanceProfile": "EMR_EC2_DefaultRole",
            "Ec2AvailabilityZone": "us-east-1a",
            "EmrManagedSlaveSecurityGroup": "sg-xxxxx"
        },
        "Name": "My Cluster",
        "ServiceRole": "EMR_DefaultRole",
        "Tags": [],
        "TerminationProtected": false,
        "UnhealthyNodeReplacement": false,
        "ReleaseLabel": "emr-5.2.0",
        "NormalizedInstanceHours": 472,
        "InstanceCollectionType": "INSTANCE_FLEET",
        "InstanceFleets": [
            {
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1487897212.74,
                        "CreationDateTime": 1487896933.948
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "ProvisionedSpotCapacity": 1,
                "Name": "MASTER",
                "InstanceFleetType": "MASTER",
                "LaunchSpecifications": {
                    "SpotSpecification": {
                        "TimeoutDurationMinutes": 60,
                        "TimeoutAction": "TERMINATE_CLUSTER"
                    }
                },
                "TargetSpotCapacity": 1,
                "ProvisionedOnDemandCapacity": 0,
                "InstanceTypeSpecifications": [
                    {
                        "BidPrice": "0.5",
                        "InstanceType": "m3.xlarge",
                        "WeightedCapacity": 1
```

```
                    }
                ],
                "Id": "if-xxxxxxx",
                "TargetOnDemandCapacity": 0
            }
        ],
        "Applications": [
            {
                "Version": "2.7.3",
                "Name": "Hadoop"
            }
        ],
        "ScaleDownBehavior": "TERMINATE_AT_INSTANCE_HOUR",
        "VisibleToAllUsers": true,
        "BootstrapActions": [],
        "MasterPublicDnsName": "ec2-xxx-xx-xxx-xx.compute-1.amazonaws.com",
        "AutoTerminate": false,
        "Id": "j-xxxxx",
        "Configurations": []
    }
}

For ami based uniform instance group cluster:

    {
        "Cluster": {
            "Status": {
                "Timeline": {
                    "ReadyDateTime": 1399400564.432,
                    "CreationDateTime": 1399400268.62
                },
                "State": "WAITING",
                "StateChangeReason": {
                    "Message": "Waiting for steps to run"
                }
            },
            "Ec2InstanceAttributes": {
                "IamInstanceProfile": "EMR_EC2_DefaultRole",
                "Ec2AvailabilityZone": "us-east-1c"
            },
            "Name": "My Cluster",
            "Tags": [],
            "TerminationProtected": true,
            "UnhealthyNodeReplacement": true,
```

```
            "RunningAmiVersion": "2.5.4",
            "InstanceGroups": [
                {
                    "RequestedInstanceCount": 1,
                    "Status": {
                        "Timeline": {
                            "ReadyDateTime": 1399400558.848,
                            "CreationDateTime": 1399400268.621
                        },
                        "State": "RUNNING",
                        "StateChangeReason": {
                            "Message": ""
                        }
                    },
                    "Name": "Master instance group",
                    "InstanceGroupType": "MASTER",
                    "InstanceType": "m1.small",
                    "Id": "ig-ABCD",
                    "Market": "ON_DEMAND",
                    "RunningInstanceCount": 1
                },
                {
                    "RequestedInstanceCount": 2,
                    "Status": {
                        "Timeline": {
                            "ReadyDateTime": 1399400564.439,
                            "CreationDateTime": 1399400268.621
                        },
                        "State": "RUNNING",
                        "StateChangeReason": {
                            "Message": ""
                        }
                    },
                    "Name": "Core instance group",
                    "InstanceGroupType": "CORE",
                    "InstanceType": "m1.small",
                    "Id": "ig-DEF",
                    "Market": "ON_DEMAND",
                    "RunningInstanceCount": 2
                }
            ],
            "Applications": [
                {
                    "Version": "1.0.3",
```

```
                    "Name": "hadoop"
                }
            ],
            "BootstrapActions": [],
            "VisibleToAllUsers": false,
            "RequestedAmiVersion": "2.4.2",
            "LogUri": "s3://myLogUri/",
            "AutoTerminate": false,
            "Id": "j-XXXXXXXX"
        }
    }
```

- Para obter detalhes da API, consulte DescribeClusterem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def describe_cluster(cluster_id, emr_client):
    """
    Gets detailed information about a cluster.

    :param cluster_id: The ID of the cluster to describe.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved cluster information.
    """
    try:
        response = emr_client.describe_cluster(ClusterId=cluster_id)
        cluster = response["Cluster"]
        logger.info("Got data for cluster %s.", cluster["Name"])
    except ClientError:
        logger.exception("Couldn't get data for cluster %s.", cluster_id)
        raise
    else:
```

```
        return cluster
```

- Para obter detalhes da API, consulte a DescribeClusterReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_emr->describecluster(
      iv_clusterid = iv_cluster_id
    ).
    DATA(lo_cluster) = oo_result->get_cluster( ).
    DATA(lv_cluster_name) = lo_cluster->get_name( ).
    MESSAGE |Retrieved cluster information for { lv_cluster_name }| TYPE 'I'.
  CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
  CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
    lv_error = lo_invalid_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeClusterreferência da API AWS SDK for SAP ABAP.

Use **`DescribeStep`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeStep`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução aos clusters de processamento de big data

CLI

AWS CLI

O seguinte comando descreve uma etapa com o ID de etapa `s-3LZC0QUT43AM` em um cluster com o ID de cluster `j-3SD91U2E1L2QX`:

```
aws emr describe-step --cluster-id j-3SD91U2E1L2QX --step-id s-3LZC0QUT43AM
```

Saída:

```
{
    "Step": {
        "Status": {
            "Timeline": {
                "EndDateTime": 1433200470.481,
                "CreationDateTime": 1433199926.597,
                "StartDateTime": 1433200404.959
            },
            "State": "COMPLETED",
            "StateChangeReason": {}
        },
        "Config": {
            "Args": [
                "s3://us-west-2.elasticmapreduce/libs/hive/hive-script",
                "--base-path",
                "s3://us-west-2.elasticmapreduce/libs/hive/",
                "--install-hive",
                "--hive-versions",
                "0.13.1"
            ],
            "Jar": "s3://us-west-2.elasticmapreduce/libs/script-runner/script-
runner.jar",
            "Properties": {}
        },
        "Id": "s-3LZC0QUT43AM",
        "ActionOnFailure": "TERMINATE_CLUSTER",
```

```
        "Name": "Setup hive"
    }
}
```

- Para obter detalhes da API, consulte DescribeStepem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def describe_step(cluster_id, step_id, emr_client):
    """
    Gets detailed information about the specified step, including the current
 state of
    the step.

    :param cluster_id: The ID of the cluster.
    :param step_id: The ID of the step.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved information about the specified step.
    """
    try:
        response = emr_client.describe_step(ClusterId=cluster_id, StepId=step_id)
        step = response["Step"]
        logger.info("Got data for step %s.", step_id)
    except ClientError:
        logger.exception("Couldn't get data for step %s.", step_id)
        raise
    else:
        return step
```

- Para obter detalhes da API, consulte a DescribeStepReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_emr->describestep(
      iv_clusterid = iv_cluster_id
      iv_stepid = iv_step_id
    ).
    DATA(lo_step) = oo_result->get_step( ).
    DATA(lv_step_name) = lo_step->get_name( ).
    MESSAGE |Retrieved step information for { lv_step_name }| TYPE 'I'.
  CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
  CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
    lv_error = lo_invalid_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeStepreferência da API AWS SDK for SAP ABAP.

Use **`ListSteps`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListSteps`.

CLI

AWS CLI

O seguinte comando lista todas as etapas em um cluster com o ID de cluster `j-3SD91U2E1L2QX`:

```
aws emr list-steps --cluster-id j-3SD91U2E1L2QX
```

- Para obter detalhes da API, consulte ListStepsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_steps(cluster_id, emr_client):
    """
    Gets a list of steps for the specified cluster. In this example, all steps
 are
    returned, including completed and failed steps.

    :param cluster_id: The ID of the cluster.
    :param emr_client: The Boto3 EMR client object.
    :return: The list of steps for the specified cluster.
    """
    try:
        response = emr_client.list_steps(ClusterId=cluster_id)
        steps = response["Steps"]
        logger.info("Got %s steps for cluster %s.", len(steps), cluster_id)
    except ClientError:
        logger.exception("Couldn't get steps for cluster %s.", cluster_id)
        raise
    else:
        return steps
```

- Para obter detalhes da API, consulte a ListStepsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_emr->liststeps(
      iv_clusterid = iv_cluster_id
    ).
    DATA(lt_steps) = oo_result->get_steps( ).
    DATA(lv_step_count) = lines( lt_steps ).
    MESSAGE |Retrieved { lv_step_count } steps for cluster| TYPE 'I'.
  CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
  CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
    lv_error = lo_invalid_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListStepsreferência da API AWS SDK for SAP ABAP.

Use **`RunJobFlow`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `RunJobFlow`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def run_job_flow(
    name,
    log_uri,
    keep_alive,
    applications,
    job_flow_role,
    service_role,
    security_groups,
    steps,
    emr_client,
):
    """
    Runs a job flow with the specified steps. A job flow creates a cluster of
    instances and adds steps to be run on the cluster. Steps added to the cluster
    are run as soon as the cluster is ready.

    This example uses the 'emr-5.30.1' release. A list of recent releases can be
    found here:
        https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-release-
components.html.

    :param name: The name of the cluster.
    :param log_uri: The URI where logs are stored. This can be an Amazon S3
 bucket URL,
                    such as 's3://my-log-bucket'.
    :param keep_alive: When True, the cluster is put into a Waiting state after
 all
                       steps are run. When False, the cluster terminates itself
 when
                       the step queue is empty.
    :param applications: The applications to install on each instance in the
 cluster,
```

```
                        such as Hive or Spark.
   :param job_flow_role: The IAM role assumed by the cluster.
   :param service_role: The IAM role assumed by the service.
   :param security_groups: The security groups to assign to the cluster
instances.
                           Amazon EMR adds all needed rules to these groups, so
                           they can be empty if you require only the default
rules.
   :param steps: The job flow steps to add to the cluster. These are run in
order
                when the cluster is ready.
   :param emr_client: The Boto3 EMR client object.
   :return: The ID of the newly created cluster.
   """
   try:
       response = emr_client.run_job_flow(
           Name=name,
           LogUri=log_uri,
           ReleaseLabel="emr-5.30.1",
           Instances={
               "MasterInstanceType": "m5.xlarge",
               "SlaveInstanceType": "m5.xlarge",
               "InstanceCount": 3,
               "KeepJobFlowAliveWhenNoSteps": keep_alive,
               "EmrManagedMasterSecurityGroup": security_groups["manager"].id,
               "EmrManagedSlaveSecurityGroup": security_groups["worker"].id,
           },
           Steps=[
               {
                   "Name": step["name"],
                   "ActionOnFailure": "CONTINUE",
                   "HadoopJarStep": {
                       "Jar": "command-runner.jar",
                       "Args": [
                           "spark-submit",
                           "--deploy-mode",
                           "cluster",
                           step["script_uri"],
                           *step["script_args"],
                       ],
                   },
               }
               for step in steps
           ],
```

```
            Applications=[{"Name": app} for app in applications],
            JobFlowRole=job_flow_role.name,
            ServiceRole=service_role.name,
            EbsRootVolumeSize=10,
            VisibleToAllUsers=True,
        )
        cluster_id = response["JobFlowId"]
        logger.info("Created cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't create cluster.")
        raise
    else:
        return cluster_id
```

- Para obter detalhes da API, consulte a RunJobFlowReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create instances configuration
        DATA(lo_instances) = NEW /aws1/cl_emrjobflowinstsconfig(
          iv_masterinstancetype = 'm5.xlarge'
          iv_slaveinstancetype = 'm5.xlarge'
          iv_instancecount = 3
          iv_keepjobflowalivewhennos00 = iv_keep_alive
          iv_emrmanagedmastersecgroup = iv_primary_sec_grp
          iv_emrmanagedslavesecgroup = iv_secondary_sec_grp
        ).
```

```
        DATA(lo_result) = lo_emr->runjobflow(
          iv_name = iv_name
          iv_loguri = iv_log_uri
          iv_releaselabel = 'emr-5.30.1'
          io_instances = lo_instances
          it_steps = it_steps
          it_applications = it_applications
          iv_jobflowrole = iv_job_flow_role
          iv_servicerole = iv_service_role
          iv_ebsrootvolumesize = 10
          iv_visibletoallusers = abap_true
        ).

        ov_cluster_id = lo_result->get_jobflowid( ).
        MESSAGE 'EMR cluster created successfully.' TYPE 'I'.
      CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
      CATCH /aws1/cx_emrclientexc INTO DATA(lo_client_error).
        lv_error = lo_client_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RunJobFlowreferência da API AWS SDK for SAP ABAP.

Use **`TerminateJobFlows`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `TerminateJobFlows`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def terminate_cluster(cluster_id, emr_client):
    """
    Terminates a cluster. This terminates all instances in the cluster and cannot
    be undone. Any data not saved elsewhere, such as in an Amazon S3 bucket, is
 lost.

    :param cluster_id: The ID of the cluster to terminate.
    :param emr_client: The Boto3 EMR client object.
    """
    try:
        emr_client.terminate_job_flows(JobFlowIds=[cluster_id])
        logger.info("Terminated cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't terminate cluster %s.", cluster_id)
        raise
```

- Para obter detalhes da API, consulte a TerminateJobFlowsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA lt_cluster_ids TYPE /aws1/cl_emrxmlstringlist_w=>tt_xmlstringlist.
        APPEND NEW /aws1/cl_emrxmlstringlist_w( iv_cluster_id ) TO
 lt_cluster_ids.

        lo_emr->terminatejobflows(
          it_jobflowids = lt_cluster_ids
        ).
        MESSAGE 'EMR cluster terminated successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a TerminateJobFlowsreferência da API AWS SDK for SAP ABAP.

Cenários para o Amazon EMR usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon EMR com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções no Amazon EMR ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie um cluster Amazon EMR de curta duração e execute uma etapa usando um AWS SDK
- Introdução aos clusters de processamento de big data
- Execute um script de shell para instalar bibliotecas em instâncias do Amazon EMR usando um AWS SDK

Crie um cluster Amazon EMR de curta duração e execute uma etapa usando um AWS SDK

O exemplo de código a seguir mostra como criar um cluster do Amazon EMR de curta duração que executa uma etapa e termina automaticamente após a conclusão dessa etapa.

Python

SDK para Python (Boto3).

Crie um cluster do Amazon EMR de curta duração que estime o valor de pi usando o Apache Spark para paralelizar um grande número de cálculos. O trabalho grava a saída em logs do

Amazon EMR e em um bucket do Amazon Simple Storage Service (Amazon S3). O cluster é encerrado automaticamente após a conclusão do trabalho.

- Crie um bucket do Amazon S3 e carregue um script de trabalho.
- Crie funções AWS Identity and Access Management (IAM).
- Crie grupos de segurança do Amazon Elastic Compute Cloud (Amazon EC2).
- Criar um cluster de curta duração e executar uma única etapa do trabalho.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon EMR

Introdução aos clusters de processamento de big data

O exemplo de código a seguir mostra como:

- Criar um par de chaves do EC2.
- Configurar o armazenamento e preparar uma aplicação.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail
```

```
# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
 true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi
```

```
    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
 then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
```

```
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession
```

```
def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
 find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
 's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red
 violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
```

```
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
 data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."
```

```
# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
 text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
```

```
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout
 period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."
```

```
# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
 "$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
```

```
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
 files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
 | head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
 | head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi
```

```
# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb
  - Rb
  - Rm
  - ssh
  - TerminateClusters
  - Wait

Execute um script de shell para instalar bibliotecas em instâncias do Amazon EMR usando um AWS SDK

O exemplo de código a seguir mostra como usar AWS Systems Manager para executar um script de shell em instâncias do Amazon EMR que instalam bibliotecas adicionais. Dessa forma, é possível automatizar o gerenciamento de instâncias em vez de executar comandos manualmente por meio de uma conexão SSH.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import argparse
import time
import boto3


def install_libraries_on_core_nodes(cluster_id, script_path, emr_client,
 ssm_client):
    """
    Copies and runs a shell script on the core nodes in the cluster.

    :param cluster_id: The ID of the cluster.
    :param script_path: The path to the script, typically an Amazon S3 object
 URL.
    :param emr_client: The Boto3 Amazon EMR client.
    :param ssm_client: The Boto3 AWS Systems Manager client.
    """
    core_nodes = emr_client.list_instances(
        ClusterId=cluster_id, InstanceGroupTypes=["CORE"]
    )["Instances"]
    core_instance_ids = [node["Ec2InstanceId"] for node in core_nodes]
    print(f"Found core instances: {core_instance_ids}.")

    commands = [
        # Copy the shell script from Amazon S3 to each node instance.
        f"aws s3 cp {script_path} /home/hadoop",
        # Run the shell script to install libraries on each node instance.
        "bash /home/hadoop/install_libraries.sh",
    ]
    for command in commands:
        print(f"Sending '{command}' to core instances...")
        command_id = ssm_client.send_command(
            InstanceIds=core_instance_ids,
```

```
            DocumentName="AWS-RunShellScript",
            Parameters={"commands": [command]},
            TimeoutSeconds=3600,
        )["Command"]["CommandId"]
        while True:
            # Verify the previous step succeeded before running the next step.
            cmd_result = ssm_client.list_commands(CommandId=command_id)
["Commands"][0]
            if cmd_result["StatusDetails"] == "Success":
                print(f"Command succeeded.")
                break
            elif cmd_result["StatusDetails"] in ["Pending", "InProgress"]:
                print(f"Command status is {cmd_result['StatusDetails']},
 waiting...")
                time.sleep(10)
            else:
                print(f"Command status is {cmd_result['StatusDetails']},
 quitting.")
                raise RuntimeError(
                    f"Command {command} failed to run. "
                    f"Details: {cmd_result['StatusDetails']}"
                )


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("cluster_id", help="The ID of the cluster.")
    parser.add_argument("script_path", help="The path to the script in Amazon
 S3.")
    args = parser.parse_args()

    emr_client = boto3.client("emr")
    ssm_client = boto3.client("ssm")

    install_libraries_on_core_nodes(
        args.cluster_id, args.script_path, emr_client, ssm_client
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ListInstancesReferência da API AWS SDK for Python (Boto3).

Exemplos de código para AWS Entity Resolution usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Entity Resolution com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS Entity Resolution Guia do usuário — Mais informações sobre AWS Entity Resolution.
- AWS Entity Resolution Referência da API — Detalhes sobre todas as AWS Entity Resolution ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Entity Resolution usar AWS SDKs
  - Hello AWS Entity Resolution
  - Aprenda as noções básicas do AWS Entity Resolution com um AWS SDK
  - Ações para AWS Entity Resolution usar AWS SDKs
    - Use CreateMatchingWorkflowcom um AWS SDK
    - Use CreateSchemaMappingcom um AWS SDK
    - Use DeleteMatchingWorkflowcom um AWS SDK
    - Use DeleteSchemaMappingcom um AWS SDK
    - Use GetMatchingJobcom um AWS SDK

- Use GetSchemaMappingcom um AWS SDK
- Use ListSchemaMappingscom um AWS SDK
- Use StartMatchingJobcom um AWS SDK
- Use TagResourcecom um AWS SDK

Exemplos básicos para AWS Entity Resolution usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Entity Resolution com AWS SDKs.

Exemplos

- Hello AWS Entity Resolution
- Aprenda as noções básicas do AWS Entity Resolution com um AWS SDK
- Ações para AWS Entity Resolution usar AWS SDKs
  - Use CreateMatchingWorkflowcom um AWS SDK
  - Use CreateSchemaMappingcom um AWS SDK
  - Use DeleteMatchingWorkflowcom um AWS SDK
  - Use DeleteSchemaMappingcom um AWS SDK
  - Use GetMatchingJobcom um AWS SDK
  - Use GetSchemaMappingcom um AWS SDK
  - Use ListSchemaMappingscom um AWS SDK
  - Use StartMatchingJobcom um AWS SDK
  - Use TagResourcecom um AWS SDK

Hello AWS Entity Resolution

O exemplo de código a seguir mostra como começar a usar o AWS Entity Resolution.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

    private static final Logger logger =
 LoggerFactory.getLogger(HelloEntityResoultion.class);

    private static EntityResolutionAsyncClient entityResolutionAsyncClient;
    public static void main(String[] args) {
        listMatchingWorkflows();
    }

    public static EntityResolutionAsyncClient getResolutionAsyncClient() {
        if (entityResolutionAsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */
```

```
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return entityResolutionAsyncClient;
    }

    /**
     * Lists all matching workflows using an asynchronous paginator.
     * <p>
     * This method requests a paginated list of matching workflows from the
     * AWS Entity Resolution service and logs the names of the retrieved
workflows.
     * It uses an asynchronous approach with a paginator and waits for the
operation
     * to complete using {@code CompletableFuture#join()}.
     * </p>
     */
    public static void listMatchingWorkflows() {
        ListMatchingWorkflowsRequest request =
ListMatchingWorkflowsRequest.builder().build();

        ListMatchingWorkflowsPublisher paginator =
            getResolutionAsyncClient().listMatchingWorkflowsPaginator(request);

        // Iterate through the paginated results asynchronously
```

```
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.workflowSummaries().forEach(workflow ->
                logger.info("Matching Workflow Name: " + workflow.workflowName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- Para obter detalhes da API, consulte ListMatchingWorkflowsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  EntityResolutionClient,
  ListMatchingWorkflowsCommand,
} from "@aws-sdk/client-entityresolution";

export const main = async () => {
  const region = "eu-west-1";
  const erClient = new EntityResolutionClient({ region: region });
  try {
    const command = new ListMatchingWorkflowsCommand({});
    const response = await erClient.send(command);
    const workflowSummaries = response.workflowSummaries;
    for (const workflowSummary of workflowSummaries) {
      console.log(`Attribute name: ${workflowSummaries[0].workflowName} `);
    }
```

```
    if (workflowSummaries.length === 0) {
      console.log("No matching workflows found.");
    }
  } catch (error) {
    console.error(
      `An error occurred in listing the workflow summaries: ${error.message} \n
 Exiting program.`,
    );
    return;
  }
};
```

- Para obter detalhes da API, consulte ListMatchingWorkflowsa Referência AWS SDK para JavaScript da API.

Aprenda as noções básicas do AWS Entity Resolution com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie mapeamento de esquemas.
- Crie um AWS Entity Resolution fluxo de trabalho.
- Inicie o trabalho correspondente para o fluxo de trabalho.
- Obtenha detalhes do trabalho correspondente.
- Obtenha o mapeamento de esquemas.
- Liste todos os mapeamentos de esquemas.
- Marque o recurso de mapeamento de esquemas.
- Exclua os AWS Entity Resolution ativos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS Entity Resolution recursos.

```
public class EntityResScenario {
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResScenario.class);
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final String STACK_NAME = "EntityResolutionCdkStack2";
    private static final String ENTITY_RESOLUTION_ROLE_ARN_KEY =
 "EntityResolutionRoleArn";
    private static final String GLUE_DATA_BUCKET_NAME_KEY = "GlueDataBucketName";
    private static final String JSON_GLUE_TABLE_ARN_KEY = "JsonErGlueTableArn";
    private static final String CSV_GLUE_TABLE_ARN_KEY = "CsvErGlueTableArn";
    private static String glueBucketName;
    private static String workflowName = "workflow-" + UUID.randomUUID();

    private static String jsonSchemaMappingName = "jsonschema-" +
 UUID.randomUUID();
    private static String jsonSchemaMappingArn = null;
    private static String csvSchemaMappingName = "csv-" + UUID.randomUUID();
    private static String roleARN;
    private static String csvGlueTableArn;
    private static String jsonGlueTableArn;
    private static Scanner scanner = new Scanner(System.in);

    private static EntityResActions actions = new EntityResActions();

    public static void main(String[] args) throws InterruptedException {

        logger.info("Welcome to the AWS Entity Resolution Scenario.");
        logger.info("""
            AWS Entity Resolution is a fully-managed machine learning service
 provided by
```

```
            Amazon Web Services (AWS) that helps organizations extract, link,
and
            organize information from multiple data sources. It leverages
natural
            language processing and deep learning models to identify and resolve
            entities, such as people, places, organizations, and products,
            across structured and unstructured data.

            With Entity Resolution, customers can build robust data integration
            pipelines to combine and reconcile data from multiple systems,
databases,
            and documents. The service can handle ambiguous, incomplete, or
conflicting
            information, and provide a unified view of entities and their
relationships.
            This can be particularly valuable in applications such as customer
360,
            fraud detection, supply chain management, and knowledge management,
where
            accurate entity identification is crucial.

            The `EntityResolutionAsyncClient` interface in the AWS SDK for Java
2.x
            provides a set of methods to programmatically interact with the AWS
Entity
            Resolution service. This allows developers to automate the entity
extraction,
            linking, and deduplication process as part of their data processing
workflows.
            With Entity Resolution, organizations can unlock the value of their
data,
            improve decision-making, and enhance customer experiences by having a
reliable,
            comprehensive view of their key entities.
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("""
            To prepare the AWS resources needed for this scenario application,
the next step uploads
```

```
            a CloudFormation template whose resulting stack creates the following
resources:
            - An AWS Glue Data Catalog table
            - An AWS IAM role
            - An AWS S3 bucket
            - An AWS Entity Resolution Schema

            It can take a couple minutes for the Stack to finish creating the
resources.
            """);
        waitForInputToContinue(scanner);
        logger.info("Generating resources...");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);
        Map<String, String> outputsMap =
CloudFormationHelper.getStackOutputsAsync(STACK_NAME).join();
        roleARN = outputsMap.get(ENTITY_RESOLUTION_ROLE_ARN_KEY);
        glueBucketName = outputsMap.get(GLUE_DATA_BUCKET_NAME_KEY);
        csvGlueTableArn = outputsMap.get(CSV_GLUE_TABLE_ARN_KEY);
        jsonGlueTableArn = outputsMap.get(JSON_GLUE_TABLE_ARN_KEY);
        logger.info(DASHES);
        waitForInputToContinue(scanner);

        try {
            runScenario();

        } catch (Exception ce) {
            Throwable cause = ce.getCause();
            logger.error("An exception happened: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
        }
    }

    private static void runScenario() throws InterruptedException {
        /*
         This JSON is a valid input for the AWS Entity Resolution service.
         The JSON represents an array of three objects, each containing an "id",
"name", and "email"
         property. This format aligns with the expected input structure for the
         Entity Resolution service.
         */
        String json = """
            {"id":"1","name":"Jane Doe","email":"jane.doe@example.com"}
            {"id":"2","name":"John Doe","email":"john.doe@example.com"}
            {"id":"3","name":"Jorge Souza","email":"jorge_souza@example.com"}
```

```
            """;
        logger.info("Upload the following JSON objects to the {} S3 bucket.",
glueBucketName);
        logger.info(json);
        String csv = """
            id,name,email,phone
            1,Jane B.,Doe,jane.doe@example.com,555-876-9846
            2,John Doe Jr.,john.doe@example.com,555-654-3210
            3,María García,maría_garcia@company.com,555-567-1234
            4,Mary Major,mary_major@company.com,555-222-3333
            """;
        logger.info("Upload the following CSV data to the {} S3 bucket.",
glueBucketName);
        logger.info(csv);
        waitForInputToContinue(scanner);
        try {
            actions.uploadInputData(glueBucketName, json, csv);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to upload input data: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ResourceNotFoundException) {
                logger.error("Failed to upload input data as the resource was not
found: {}", cause.getMessage(), cause);
            }
            return;
        }
        logger.info("The JSON and CSV objects have been uploaded to the S3
bucket.");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create Schema Mapping");
        logger.info("""
            Entity Resolution schema mapping aligns and integrates data from
            multiple sources by identifying and matching corresponding entities
            like customers or products. It unifies schemas, resolves conflicts,
            and uses machine learning to link related entities, enabling a
```

```
            consolidated, accurate view for improved data quality and decision-
making.

            In this example, the schema mapping lines up with the fields in the
 JSON and CSV objects. That is,
            it contains these fields: id, name, and email.
            """);
        try {
            CreateSchemaMappingResponse response =
 actions.createSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingName = response.schemaName();
            logger.info("The JSON schema mapping name is " +
 jsonSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to create JSON schema mapping: {}",
 ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
 cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating schema mapping:
 {}", cause.getMessage(), cause);
            }
            return;
        }

        try {
            CreateSchemaMappingResponse response =
 actions.createSchemaMappingAsync(csvSchemaMappingName).join();
            csvSchemaMappingName = response.schemaName();
            logger.info("The CSV schema mapping name is " +
 csvSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause == null) {
                logger.error("Failed to create CSV schema mapping: {}",
 ce.getMessage(), ce);
            }
```

```
            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating CSV schema mapping:
{}", cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Entity Resolution Workflow. ");
        logger.info("""
            An Entity Resolution matching workflow identifies and links records
            across datasets that represent the same real-world entity, such as
            customers or products. Using techniques like schema mapping,
            data profiling, and machine learning algorithms,
            it evaluates attributes like names or emails to detect duplicates
            or relationships, even with variations or inconsistencies.
            The workflow outputs consolidated, de-duplicated data.

            We will use the machine learning-based matching technique.
            """);
        waitForInputToContinue(scanner);
        try {
            String workflowArn = actions.createMatchingWorkflowAsync(
                roleARN, workflowName, glueBucketName, jsonGlueTableArn,
                jsonSchemaMappingName, csvGlueTableArn,
csvSchemaMappingName).join();

            logger.info("The workflow ARN is: " + workflowArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("An unexpected error occurred: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ValidationException) {
                logger.error("Validation error: {}", cause.getMessage(), cause);
            } else if (cause instanceof ConflictException) {
```

```
                logger.error("Workflow conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error: {}", cause.getMessage(), cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        logger.info("3. Start the matching job of the " + workflowName + "
workflow.");
        waitForInputToContinue(scanner);
        String jobId = null;
        try {
            jobId = actions.startMatchingJobAsync(workflowName).join();
            logger.info("The matching job was successfully started.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("Job conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error while starting the job: {}",
ce.getMessage(), ce);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. While the matching job is running, let's look at other
API methods. First, let's get details for job " + jobId);
        waitForInputToContinue(scanner);
        try {
            actions.getMatchingJobAsync(jobId, workflowName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The matching job not found: {}",
cause.getMessage(), cause);
            } else {
```

```
                logger.error("Failed to start matching job: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get the schema mapping for the JSON data.");
        waitForInputToContinue(scanner);
        try {
            GetSchemaMappingResponse response =
actions.getSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingArn = response.schemaArn();
            logger.info("Schema mapping ARN is " + jsonSchemaMappingArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("Schema mapping not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Error retrieving the specific schema mapping: " +
ce.getCause().getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List Schema Mappings.");
        try {
            actions.ListSchemaMappings();
        } catch (CompletionException ce) {
            logger.error("Error retrieving schema mappings: " +
ce.getCause().getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Tag the {} resource.", jsonSchemaMappingName);
        logger.info("""
```

```
            Tags can help you organize and categorize your Entity Resolution
resources.
            You can also use them to scope user permissions by granting a user
permission
            to access or change only resources with certain tag values.
            In Entity Resolution, SchemaMapping and MatchingWorkflow can be
tagged. For this example,
            the SchemaMapping is tagged.
                """);
        try {
            actions.tagEntityResource(jsonSchemaMappingArn).join();
        } catch (CompletionException ce) {
            logger.error("Error tagging the resource: " +
ce.getCause().getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. View the results of the AWS Entity Resolution
Workflow.");
        logger.info("""
            You cannot view the result of the workflow that is in a running
state.
            In order to view the results, you need to wait for the workflow that
we started in step 3 to complete.

            If you choose not to wait, you cannot view the results. You can
perform
            this task manually in the AWS Management Console.

            This can take up to 30 mins (y/n).
            """);
        String viewAns = scanner.nextLine().trim();
        boolean isComplete = false;
        if (viewAns.equalsIgnoreCase("y")) {
            logger.info("You selected to view the Entity Resolution Workflow
results.");
            countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
            isComplete = true;
            try {
```

```
                JobMetrics metrics = actions.getJobInfo(workflowName,
jobId).join();
                logger.info("Number of input records: {}",
metrics.inputRecords());
                logger.info("Number of match ids: {}", metrics.matchIDs());
                logger.info("Number of records not processed: {}",
metrics.recordsNotProcessed());
                logger.info("Number of total records processed: {}",
metrics.totalRecordsProcessed());
                logger.info("The following represents the output data generated
by the Entity Resolution workflow based on the JSON and CSV input data. The
output data is stored in the {} bucket.", glueBucketName);
                actions.printData(glueBucketName);

                logger.info("""

                    Note that each of the last 2 records are considered a match
even though the 'name' differs between the records;
                    For example 'John Doe Jr.' compared to 'John Doe'.
                    The confidence level is a value between 0 and 1, where 1
indicates a perfect match.

                    """);

            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The job not found: {}", cause.getMessage(),
cause);
                } else {
                    logger.error("Error retrieving job information: " +
ce.getCause().getMessage());
                }
                return;
            }
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Do you want to delete the resources, including the
workflow? (y/n)");
        logger.info("""
```

```
            You cannot delete the workflow that is in a running state.
            In order to delete the workflow, you need to wait for the workflow to
complete.

            You can delete the workflow manually in the AWS Management Console at
a later time.

            If you already waited for the workflow to complete in the previous
step,
            the workflow is completed and you can delete it.

            If the workflow is not completed, this can take up to 30 mins (y/n).
            """);
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                if (!isComplete) {
                    countdownWithWorkflowCheck(actions, 1800, jobId,
workflowName);
                }
                actions.deleteMatchingWorkflowAsync(workflowName).join();
                logger.info("Workflow deleted successfully!");
            } catch (CompletionException ce) {
                logger.info("Error deleting the workflow: {} ", ce.getMessage());
                return;
            }

            try {
                // Delete both schema mappings.
                actions.deleteSchemaMappingAsync(jsonSchemaMappingName).join();
                actions.deleteSchemaMappingAsync(csvSchemaMappingName).join();
                logger.info("Both schema mappings were deleted successfully!");
            } catch (CompletionException ce) {
                logger.error("Error deleting schema mapping: {}",
ce.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            logger.info(DASHES);
            logger.info("""
                Now we delete the CloudFormation stack, which deletes
                the resources that were created at the beginning of this
scenario.
```

```
                """);
            waitForInputToContinue(scanner);
            logger.info(DASHES);
            try {
                deleteCloudFormationStack();
            } catch (RuntimeException e) {
                logger.error("Failed to delete the stack: {}", e.getMessage());
                return;
            }

        } else {
            logger.info("You can delete the AWS resources in the AWS Management
Console.");
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the AWS Entity Resolution scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdownWithWorkflowCheck(EntityResActions actions, int
totalSeconds, String jobId, String workflowName) throws InterruptedException {
        int secondsElapsed = 0;
```

```
        while (true) {
            // Calculate display minutes and seconds.
            int remainingTime = totalSeconds - secondsElapsed;
            int displayMinutes = remainingTime / 60;
            int displaySeconds = remainingTime % 60;

            // Print the countdown.
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
            secondsElapsed++;

            // Check workflow status every 60 seconds.
            if (secondsElapsed % 60 == 0 || remainingTime <= 0) {
                GetMatchingJobResponse response =
actions.checkWorkflowStatusCompleteAsync(jobId, workflowName).join();
                if (response != null &&
"SUCCEEDED".equalsIgnoreCase(String.valueOf(response.status()))) {
                    logger.info(""); // Move to the next line after countdown.
                    logger.info("Countdown complete: Workflow is in Completed
state!");
                    break; // Break out of the loop if the status is "SUCCEEDED"
                }
            }

            // If countdown reaches zero, reset it for continuous countdown.
            if (remainingTime <= 0) {
                secondsElapsed = 0;
            }
        }
    }

    private static void deleteCloudFormationStack() {
        try {
            CloudFormationHelper.emptyS3Bucket(glueBucketName);
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
            logger.info("Resources deleted successfully!");
        } catch (CloudFormationException e) {
            throw new RuntimeException("Failed to delete CloudFormation stack: "
+ e.getMessage(), e);
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to empty S3 bucket: " +
e.getMessage(), e);
        }
    }
```

```
}
```

Uma classe wrapper para métodos do AWS Entity Resolution SDK.

```
public class EntityResActions {

    private static final String PREFIX = "eroutput/";
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResActions.class);

    private static EntityResolutionAsyncClient entityResolutionAsyncClient;

    private static S3AsyncClient s3AsyncClient;

    public static EntityResolutionAsyncClient getResolutionAsyncClient() {
        if (entityResolutionAsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
 call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
```

```
                .build();

            entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return entityResolutionAsyncClient;
    }

    public static S3AsyncClient getS3AsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
```

```
        }
        return s3AsyncClient;
    }

    /**
     * Deletes the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
     * @return a {@link CompletableFuture} that completes when the schema mapping
is deleted successfully,
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
        DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().deleteSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully deleted the schema mapping, log the success
message.
                    logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema
mapping: " + schemaName, exception);
                }
            });
    }
```

```
    /**
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
     * retrieve the schema mappings, and prints the name of each schema mapping
to the console.
     */
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();

        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }

    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has
been deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
    public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
        DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().deleteMatchingWorkflow(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
```

```
                    logger.info("{} was deleted", workflowName );
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous
creation of the schema mapping
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.U

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeTy

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttribute
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.U
```

```
SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeTy

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttribute

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).
            );
        }

        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema
mapping already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema
mapping: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Retrieves the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to retrieve the mapping for
     * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
```

```
     * is complete
     * @throws RuntimeException if the schema mapping retrieval fails
     */
    public CompletableFuture<GetSchemaMappingResponse>
getSchemaMappingAsync(String schemaName) {
        GetSchemaMappingRequest mappingRequest =
GetSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema
mapping was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job
information is available or an exception occurs
```

```
    */
   public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
       GetMatchingJobRequest request = GetMatchingJobRequest.builder()
           .jobId(jobId)
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().getMatchingJob(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully fetched the matching job details, log the job
status.
                   logger.info("Job status: " + response.status());
                   logger.info("Job details: " + response.toString());
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested job could
not be found.", cause);
                   }

                   // Wrap other exceptions in a CompletionException with the
message.
                   throw new CompletionException("Error fetching matching job: "
+ exception.getMessage(), exception);
               }
           });
   }


   /**
    * Starts a matching job asynchronously for the specified workflow name.
    *
    * @param workflowName the name of the workflow for which to start the
matching job
    * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
    * string if the operation fails
```

```
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already
running. Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }

    /**
     * Checks the status of a workflow asynchronously.
     *
     * @param jobId        the ID of the job to check
     * @param workflowName the name of the workflow to check
     * @return a CompletableFuture that resolves to a boolean value indicating
whether the workflow has completed
     * successfully
     */
    public CompletableFuture<GetMatchingJobResponse>
checkWorkflowStatusCompleteAsync(String jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
```

```
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Process the response and log the job status.
                    logger.info("Job status: " + response.status());
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while checking job status.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested resource was
not found while checking the job status.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to check job status: "
+ exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
     *
     * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
     * @param workflowName           the name of the workflow to be created
     * @param outputBucket           the S3 bucket path where the workflow
output will be stored
     * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to
be used as the input source
     * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
     * @return a CompletableFuture that, when completed, will return the ARN of
the created workflow
     */
```

```
    public CompletableFuture<String> createMatchingWorkflowAsync(
        String roleARN
        , String workflowName
        , String outputBucket
        , String jsonGlueTableArn
        , String jsonErSchemaMappingName
        , String csvGlueTableArn
        , String csvErSchemaMappingName) {

        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();
```

```
        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please
check input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }

    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
```

```
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was
not found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: "
+ exception.getMessage(), exception);
                }
            });
    }

    public CompletableFuture<JobMetrics> getJobInfo(String workflowName, String
jobId) {
        return getResolutionAsyncClient().getMatchingJob(b -> b
                .workflowName(workflowName)
                .jobId(jobId))
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Job metrics fetched successfully for jobId: " +
jobId);
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("Invalid request: Job id
was not found.", cause);
```

```
                    }
                    throw new CompletionException("Failed to fetch job info: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.metrics()); // Extract job metrics
    }

    /**
     * Uploads data to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the data to
     * @param jsonData   the JSON data to be uploaded
     * @param csvData    the CSV data to be uploaded
     * @return a {@link CompletableFuture} representing both asynchronous
operation of uploading the data
     * @throws RuntimeException if an error occurs during the file upload
     */

    public void uploadInputData(String bucketName, String jsonData, String
csvData) {
        // Upload JSON data.
        String jsonKey = "jsonData/data.json";
        PutObjectRequest jsonUploadRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(jsonKey)
            .contentType("application/json")
            .build();

        CompletableFuture<PutObjectResponse> jsonUploadResponse
= getS3AsyncClient().putObject(jsonUploadRequest,
AsyncRequestBody.fromString(jsonData));

        // Upload CSV data.
        String csvKey = "csvData/data.csv";
        PutObjectRequest csvUploadRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(csvKey)
            .contentType("text/csv")
            .build();
        CompletableFuture<PutObjectResponse> csvUploadResponse
= getS3AsyncClient().putObject(csvUploadRequest,
AsyncRequestBody.fromString(csvData));
```

```
        CompletableFuture.allOf(jsonUploadResponse, csvUploadResponse)
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    // Wrap an AWS exception.
                    throw new CompletionException("Failed to upload files", ex);
                }
            }).join();

    }

    /**
     * Finds the latest file in the S3 bucket that starts with "run-" in any
depth of subfolders
     */
    private CompletableFuture<String> findLatestMatchingFile(String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(PREFIX) // Searches within the given folder
            .build();

        return getS3AsyncClient().listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .filter(key -> key.matches(".*?/run-[0-9a-zA-Z\\-]+")) // Matches
files like run-XXXXX in any subfolder
                .max(String::compareTo) // Gets the latest file
                .orElse(null))
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Latest matching file found: " + result);
                    } else {
                        logger.info("No matching files found.");
                    }
                } else {
                    throw new CompletionException("Failed to find latest matching
file: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prints the data located in the file in the S3 bucket that starts with
"run-" in any depth of subfolders
```

```
     */
    public void printData(String bucketName) {
        try {
            // Find the latest file with "run-" prefix in any depth of
subfolders.
            String s3Key = findLatestMatchingFile(bucketName).join();
            if (s3Key == null) {
                logger.error("No matching files found in S3.");
                return;
            }

            logger.info("Downloading file: " + s3Key);

            // Read CSV file as String.
            String csvContent = readCSVFromS3Async(bucketName, s3Key).join();
            if (csvContent.isEmpty()) {
                logger.error("File is empty.");
                return;
            }

            // Process CSV content.
            List<String[]> records = parseCSV(csvContent);
            printTable(records);

        } catch (RuntimeException | IOException | CsvException e) {
            logger.error("Error processing CSV file from S3: " + e.getMessage());
            e.printStackTrace();
        }
    }

    /**
     * Reads a CSV file from S3 and returns it as a String.
     */
    private static CompletableFuture<String> readCSVFromS3Async(String
bucketName, String s3Key) {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(s3Key)
            .build();

        // Initiating the asynchronous request to get the file as bytes
        return getS3AsyncClient().getObject(getObjectRequest,
AsyncResponseTransformer.toBytes())
```

```
            .thenApply(responseBytes -> responseBytes.asUtf8String()) // Convert
bytes to UTF-8 string
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Failed to read CSV from S3: "
+ exception.getMessage(), exception);
                } else {
                    logger.info("Successfully fetched CSV file content from
S3.");
                }
            });
    }

    /**
     * Parses CSV content from a String into a list of records.
     */
    private static List<String[]> parseCSV(String csvContent) throws IOException,
CsvException {
        try (CSVReader csvReader = new CSVReader(new StringReader(csvContent))) {
            return csvReader.readAll();
        }
    }

    /**
     * Prints the given CSV data in a formatted table
     */
    private static void printTable(List<String[]> records) {
        if (records.isEmpty()) {
            System.out.println("No records found.");
            return;
        }

        String[] headers = records.get(0);
        List<String[]> rows = records.subList(1, records.size());

        // Determine column widths dynamically based on longest content
        int[] columnWidths = new int[headers.length];
        for (int i = 0; i < headers.length; i++) {
            final int columnIndex = i;
            int maxWidth = Math.max(headers[i].length(), rows.stream()
                .map(row -> row.length > columnIndex ?
row[columnIndex].length() : 0)
                .max(Integer::compareTo)
                .orElse(0));
```

```
            columnWidths[i] = Math.min(maxWidth, 25); // Limit max width for
 better readability
        }

        // Enable ANSI Console for colored output
        AnsiConsole.systemInstall();

        // Print table header
        System.out.println(ansi().fgYellow().a("=== CSV Data from S3
 ===").reset());
        printRow(headers, columnWidths, true);

        // Print rows
        rows.forEach(row -> printRow(row, columnWidths, false));

        // Restore console to normal
        AnsiConsole.systemUninstall();
    }

    private static void printRow(String[] row, int[] columnWidths, boolean
 isHeader) {
        String border = IntStream.range(0, columnWidths.length)
            .mapToObj(i -> "-".repeat(columnWidths[i] + 2))
            .collect(Collectors.joining("+", "+", "+"));

        if (isHeader) {
            System.out.println(border);
        }

        System.out.print("|");
        for (int i = 0; i < columnWidths.length; i++) {
            String cell = (i < row.length && row[i] != null) ? row[i] : "";
            System.out.printf(" %-" + columnWidths[i] + "s |", isHeader ?
 ansi().fgBrightBlue().a(cell).reset() : cell);
        }
        System.out.println();

        if (isHeader) {
            System.out.println(border);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateMatchingWorkflow
  - CreateSchemaMapping
  - DeleteMatchingWorkflow
  - DeleteSchemaMapping
  - GetMatchingJob
  - GetSchemaMapping
  - ListMatchingWorkflows
  - ListSchemaMappings
  - StartMatchingJob
  - TagResource

JavaScript

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS Entity Resolution recursos.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
```

```
  waitUntilStackExists,
  waitUntilStackCreateComplete,
} from "@aws-sdk/client-cloudformation";
import {
  EntityResolutionClient,
  CreateSchemaMappingCommand,
  CreateMatchingWorkflowCommand,
  GetMatchingJobCommand,
  StartMatchingJobCommand,
  GetSchemaMappingCommand,
  ListSchemaMappingsCommand,
  TagResourceCommand,
  DeleteMatchingWorkflowCommand,
  DeleteSchemaMappingCommand,
  ConflictException,
  ValidationException,
} from "@aws-sdk/client-entityresolution";
import {
  DeleteObjectsCommand,
  DeleteBucketCommand,
  PutObjectCommand,
  S3Client,
  ListObjectsCommand,
} from "@aws-sdk/client-s3";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { readFile } from "node:fs/promises";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = `${data.inputs.entityResolutionStack}`;

/*The inputs for this example can be edited in the ../input.json.*/
import data from "../inputs.json" with { type: "json" };
const skipWhenErrors = (state) => state.errors.length > 0;
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
/* v8 ignore next 3 */
```

```
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  verbose: "false",
  skipWhen: skipWhenErrors,
});

const region = "eu-west-1";

const entityResolutionClient = new EntityResolutionClient({ region: region });
const cloudFormationClient = new CloudFormationClient({ region: region });
const s3Client = new S3Client({ region: region });

const greet = new ScenarioOutput(
  "greet",
  "AWS Entity Resolution is a fully-managed machine learning service provided by
 " +
    "Amazon Web Services (AWS) that helps organizations extract, link, and " +
    "organize information from multiple data sources. It leverages natural " +
    "language processing and deep learning models to identify and resolve " +
    "entities, such as people, places, organizations, and products, " +
    "across structured and unstructured data.\n" +
    "\n" +
    "With Entity Resolution, customers can build robust data integration " +
    "pipelines to combine and reconcile data from multiple systems, databases, "
 +
    "and documents. The service can handle ambiguous, incomplete, or conflicting
 " +
    "information, and provide a unified view of entities and their relationships.
 " +
    "This can be particularly valuable in applications such as customer 360, " +
    "fraud detection, supply chain management, and knowledge management, where "
 +
    "accurate entity identification is crucial.\n" +
    "\n" +
    "The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x " +
    "provides a set of methods to programmatically interact with the AWS Entity "
 +
    "Resolution service. This allows developers to automate the entity
 extraction, " +
    "linking, and deduplication process as part of their data processing
 workflows. " +
    "With Entity Resolution, organizations can unlock the value of their data, "
 +
```

```
    "improve decision-making, and enhance customer experiences by having a
 reliable, " +
    "comprehensive view of their key entities.",

  { header: true },
);
const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "To prepare the AWS resources needed for this scenario application, the next
 step uploads " +
    "a CloudFormation template whose resulting stack creates the following
 resources:\n" +
    "- An AWS Glue Data Catalog table \n" +
    "- An AWS IAM role \n" +
    "- An AWS S3 bucket \n" +
    "- An AWS Entity Resolution Schema \n" +
    "It can take a couple minutes for the Stack to finish creating the
 resources.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../resources/cfn/entity-resolution-basics/entity-
resolution-basics-template.yml`,
        "utf8",
      );
      await cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: cloudFormationClient },
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: cloudFormationClient },
        { StackName: stackName },
      );
```

```
      const stack = await cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );

      state.entityResolutionRole = stack.Stacks[0].Outputs[1];
      state.jsonGlueTable = stack.Stacks[0].Outputs[2];
      state.CSVGlueTable = stack.Stacks[0].Outputs[3];
      state.glueDataBucket = stack.Stacks[0].Outputs[0];
      state.stackName = stack.StackName;
      console.log(state.glueDataBucket);
      console.log(
        `The  ARN of the EntityResolution Role is
 ${state.entityResolutionRole.OutputValue}`,
      );
      console.log(
        `The ARN of the Json Glue Table is ${state.jsonGlueTable.OutputValue}`,
      );
      console.log(
        `The ARN of the CSV Glue Table is ${state.CSVGlueTable.OutputValue}`,
      );
      console.log(
        `The name of the Glue Data Bucket is
 ${state.glueDataBucket.OutputValue}\n`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
    try {
      console.log(
        `Uploading the following JSON in ../data.json to the
 ${state.glueDataBucket.OutputValue} S3 bucket...`,
      );
      const bucketName = state.glueDataBucket.OutputValue;

      const putObjectParams = {
        Bucket: bucketName,
        Key: "jsonData/data.json",
        Body: await readFileSync(
          `${__dirname}/../../../../javascriptv3/example_code/entityresolution/
data.json`,
        ),
```

```
      };
      const command = new PutObjectCommand(putObjectParams);
      const response = await s3Client.send(command);
      console.log(
        `../data.json file data uploaded to the
 ${state.glueDataBucket.OutputValue} S3 bucket.\n`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
    try {
      console.log(
        `Uploading the CSV data in ../data.csv to the
 ${state.glueDataBucket.OutputValue} S3 bucket...`,
      );

      const bucketName = state.glueDataBucket.OutputValue;
      const putObjectParams = {
        Bucket: bucketName,
        Key: "csvData/data.csv",
        Body: await readFileSync(
          `${__dirname}/../../../../javascriptv3/example_code/entityresolution/
data.csv`,
        ),
      };
      const command = new PutObjectCommand(putObjectParams);
      const response = await s3Client.send(command);
      console.log(
        `../data.csv file data uploaded to the
 ${state.glueDataBucket.OutputValue} S3 bucket.`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateSchemaMapping = new ScenarioOutput(
  "displayCreateSchemaMapping",
  "1. Create Schema Mapping" +
    "Entity Resolution schema mapping aligns and integrates data from " +
    "multiple sources by identifying and matching corresponding entities " +
```

```
    "like customers or products. It unifies schemas, resolves conflicts, " +
    "and uses machine learning to link related entities, enabling a " +
    "consolidated, accurate view for improved data quality and decision-making."
 +
    "\n" +
    "In this example, the schema mapping lines up with the fields in the JSON and
 CSV objects. That is, " +
    " it contains these fields: id, name, and email. ",
);

const sdkCreateSchemaMapping = new ScenarioAction(
  "sdkCreateSchemaMapping",
  async (/** @type {State} */ state) => {
    const createSchemaMappingParamsJson = {
      schemaName: `${data.inputs.schemaNameJson}`,
      mappedInputFields: [
        {
          fieldName: "id",
          type: "UNIQUE_ID",
        },
        {
          fieldName: "name",
          type: "NAME",
        },
        {
          fieldName: "email",
          type: "EMAIL_ADDRESS",
        },
      ],
    };
    const createSchemaMappingParamsCSV = {
      schemaName: `${data.inputs.schemaNameCSV}`,
      mappedInputFields: [
        {
          fieldName: "id",
          type: "UNIQUE_ID",
        },
        {
          fieldName: "name",
          type: "NAME",
        },
        {
          fieldName: "email",
          type: "EMAIL_ADDRESS",
```

```
        },
        {
          fieldName: "phone",
          type: "PROVIDER_ID",
          subType: "STRING",
        },
      ],
    };
    try {
      const command = new CreateSchemaMappingCommand(
        createSchemaMappingParamsJson,
      );
      const response = await entityResolutionClient.send(command);
      state.schemaNameJson = response.schemaName;
      state.schemaArn = response.schemaArn;
      state.idOutputAttribute = response.mappedInputFields[0].fieldName;
      state.nameOutputAttribute = response.mappedInputFields[1].fieldName;
      state.emailOutputAttribute = response.mappedInputFields[2].fieldName;

      console.log("The JSON schema mapping name is ", state.schemaNameJson);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema mapping already exists: ${caught.message} \n Exiting
program.`,
        );
        return;
      }
    }
    try {
      const command = new CreateSchemaMappingCommand(
        createSchemaMappingParamsCSV,
      );
      const response = await entityResolutionClient.send(command);
      state.schemaNameCSV = response.schemaName;
      state.phoneOutputAttribute = response.mappedInputFields[3].fieldName;
      console.log("The CSV schema mapping name is ", state.schemaNameCSV);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An unexpected error occurred while creating the geofence collection:
${caught.message} \n Exiting program.`,
        );
        return;
```

```
      }
    }
  },
);
const displayCreateMatchingWorkflow = new ScenarioOutput(
  "displayCreateMatchingWorkflow",
  "2. Create an AWS Entity Resolution Workflow. " +
    "An Entity Resolution matching workflow identifies and links records " +
    "across datasets that represent the same real-world entity, such as " +
    "customers or products. Using techniques like schema mapping, " +
    "data profiling, and machine learning algorithms, " +
    "it evaluates attributes like names or emails to detect duplicates " +
    "or relationships, even with variations or inconsistencies. " +
    "The workflow outputs consolidated, de-duplicated data." +
    "\n" +
    "We will use the machine learning-based matching technique.",
);

const sdkCreateMatchingWorkflow = new ScenarioAction(
  "sdkCreateMatchingWorkflow",
  async (/** @type {State} */ state) => {
    const createMatchingWorkflowParams = {
      roleArn: `${state.entityResolutionRole.OutputValue}`,
      workflowName: `${data.inputs.workflowName}`,
      description: "Created by using the AWS SDK for JavaScript (v3).",
      inputSourceConfig: [
        {
          inputSourceARN: `${state.jsonGlueTable.OutputValue}`,
          schemaName: `${data.inputs.schemaNameJson}`,
          applyNormalization: false,
        },
        {
          inputSourceARN: `${state.CSVGlueTable.OutputValue}`,
          schemaName: `${data.inputs.schemaNameCSV}`,
          applyNormalization: false,
        },
      ],
      outputSourceConfig: [
        {
          outputS3Path: `s3://${state.glueDataBucket.OutputValue}/eroutput`,
          output: [
            {
              name: state.idOutputAttribute,
            },
```

```
            {
              name: state.nameOutputAttribute,
            },
            {
              name: state.emailOutputAttribute,
            },
            {
              name: state.phoneOutputAttribute,
            },
          ],
          applyNormalization: false,
        },
      ],
      resolutionTechniques: { resolutionType: "ML_MATCHING" },
    };
    try {
      const command = new CreateMatchingWorkflowCommand(
        createMatchingWorkflowParams,
      );
      const response = await entityResolutionClient.send(command);
      state.workflowArn = response.workflowArn;
      console.log(
        `Workflow created successfully.\n The workflow ARN is:
 ${response.workflowArn}`,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The matching workflow already exists: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
      if (caught instanceof ValidationException) {
        console.error(
          `There was a validation exception: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
    }
  },
);
const displayMatchingJobOfWorkflow = new ScenarioOutput(
```

```
  "displayMatchingJobOfWorkflow",
  "3. Start the matching job of the workflow",
);

const sdkMatchingJobOfWorkflow = new ScenarioAction(
  "sdk",
  async (/** @type {State} */ state) => {
    const matchingJobOfWorkflowParams = {
      workflowName: `${data.inputs.workflowName}`,
    };
    try {
      const command = new StartMatchingJobCommand(matchingJobOfWorkflowParams);
      const response = await entityResolutionClient.send(command);
      state.jobID = response.jobId;
      console.log(`Job ID: ${state.jobID} \n
The matching job was successfully started.`);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The matching workflow already exists: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
    }
  },
);

const displayGetDetailsforJob = new ScenarioOutput(
  "displayGetDetailsforJob",
  `4. While the matching job is running, let's look at other API methods. First,
 let's get details for the job `,
);

const sdkGetDetailsforJob = new ScenarioAction(
  "sdkGetDetailsforJob",
  async (/** @type {State} */ state) => {
    const getDetailsforJobParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${state.jobID}`,
    };
    try {
      const command = new GetMatchingJobCommand(getDetailsforJobParams);
      const response = await entityResolutionClient.send(command);
```

```
      state.Status = response.status;
      state.response = response;
      console.log(`Job status: ${state.Status} `);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayGetSchemaMappingJson = new ScenarioOutput(
  "displayGetSchemaMappingJson",
  "5. Get the schema mapping for the JSON data.",
);

const sdkGetSchemaMappingJson = new ScenarioAction(
  "sdkGetSchemaMappingJson",
  async (/** @type {State} */ state) => {
    const getSchemaMappingJsonParams = {
      schemaName: `${data.inputs.schemaNameJson}`,
    };
    try {
      const command = new GetSchemaMappingCommand(getSchemaMappingJsonParams);
      const response = await entityResolutionClient.send(command);
      console.log("Schema·mapping·ARN·is:·", response.schemaArn);
      const resultMappings = response.mappedInputFields;
      const noOfResultMappings = resultMappings.length;
      for (let i = 0; i < noOfResultMappings; i++) {
        console.log(
          `Attribute name: ${resultMappings[i].fieldName} `,
          `Attribute type: ${resultMappings[i].type}`,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayListSchemaMappings = new ScenarioOutput(
  "displayListSchemaMappings",
  "6. List Schema Mappings.",
);
```

```
const sdkListSchemaMappings = new ScenarioAction(
  "sdkListSchemaMappings",
  async (/** @type {State} */ state) => {
    try {
      const command = new ListSchemaMappingsCommand({});
      const response = await entityResolutionClient.send(command);
      const noOfSchemas = response.schemaList.length;
      for (let i = 0; i < noOfSchemas; i++) {
        console.log(
          `Schema Mapping Name: ${response.schemaList[i].schemaName} `,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayTagTheJsonSchema = new ScenarioOutput(
  "display",
  "7. Tag the resource. \n" +
    "Tags can help you organize and categorize your Entity Resolution resources.
 " +
    "You can also use them to scope user permissions by granting a user
 permission " +
    "to access or change only resources with certain tag values. " +
    "In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged. For
 this example, " +
    "the SchemaMapping is tagged.",
);

const sdkTagTheJsonSchema = new ScenarioAction(
  "sdkGetSchemaMappingJson",
  async (/** @type {State} */ state) => {
    const tagResourceCommandParams = {
      resourceArn: state.schemaArn,
      tags: {
        tag1: "tag1Value",
        tag2: "tag2Value",
      },
    };
    try {
```

```
      const command = new TagResourceCommand(tagResourceCommandParams);
      const response = await entityResolutionClient.send(command);
      console.log("Successfully tagged the resource.");
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayGetJobInfo = new ScenarioOutput(
  "displayGetJobInfo",
  "8. View the results of the AWS Entity Resolution Workflow.\n " +
    "Please perform this task manually in the AWS Management Console. ",
);

const displayDeleteResources = new ScenarioOutput(
  "displayDeleteResources",
  "9. Delete the resources \n" +
    "You cannot delete a workflow that is in a running state. So this will take
 ~30 minutes.\n" +
    "If you don't want to delete the resources, simply exit this application.",
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    console.log(
      "You selected to delete the resources. This will take about 30 minutes.",
    );
    await wait(1800);
    const bucketName = state.glueDataBucket.OutputValue;
    try {
      const emptyBucket = async ({ bucketName }) => {
        const listObjectsCommand = new ListObjectsCommand({
          Bucket: bucketName,
        });
        const { Contents } = await s3Client.send(listObjectsCommand);
        const keys = Contents.map((c) => c.Key);

        const deleteObjectsCommand = new DeleteObjectsCommand({
          Bucket: bucketName,
          Delete: { Objects: keys.map((key) => ({ Key: key })) },
        });
```

```
        await s3Client.send(deleteObjectsCommand);
        console.log(`Bucket ${bucketName} emptied successfully.\n`);
      };
      await emptyBucket({ bucketName });
    } catch (error) {
      console.log("error ", error);
    }
    try {
      const deleteBucket = async ({ bucketName }) => {
        const command = new DeleteBucketCommand({ Bucket: bucketName });
        await s3Client.send(command);
        console.log(`Bucket ${bucketName} deleted successfully.\n`);
      };
      await deleteBucket({ bucketName });
    } catch (error) {
      console.log("error ", error);
    }
    try {
      console.log(
        "Now we will delete the CloudFormation stack, which deletes the resources
that were created at the beginning of the scenario.",
      );
      const deleteStackParams = { StackName: `${state.stackName}` };
      const command = new DeleteStackCommand(deleteStackParams);
      const response = await cloudFormationClient.send(command);
      console.log("CloudFormation stack deleted successfully.");
    } catch (error) {
      console.log("error ", error);
    }
    try {
      const deleteWorkflowParams = {
        workflowName: `${data.inputs.workflowName}`,
      };
      const command = new DeleteMatchingWorkflowCommand(deleteWorkflowParams);
      const response = await entityResolutionClient.send(command);
      console.log("Workflow deleted successfully!");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `Job associated with workflow ${data.inputs.workflowName} is still
running, so can't be deleted.
          Neither can schemas ${data.inputs.schemaNameJson} and
${data.inputs.schemaNameCSV} associated with it. Please confirm this workflow is
finished in the AWS Management Console, then delete it manually.`,
```

```
        );
        throw caught;
      }
    }
    try {
      const deleteJSONschemaMapping = {
        schemaName: `${data.inputs.schemaNameJson}`,
      };
      const command = new DeleteSchemaMappingCommand(deleteJSONschemaMapping);
      const response = await entityResolutionClient.send(command);
      console.log("Schema mapping deleted successfully. ");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema ${data.inputs.schemaNameJson} can't be deleted because it
is associated with workflow
           ${data.inputs.workflowName}, which is still running. Please confirm
this workflow is finished in the AWS Management Console, then delete it
manually.`,
        );
        throw caught;
      }
    }
    try {
      const deleteCSVschemaMapping = {
        schemaName: `${data.inputs.schemaNameCSV}`,
      };
      const command = new DeleteSchemaMappingCommand(deleteCSVschemaMapping);
      const response = await entityResolutionClient.send(command);
      console.log("Schema mapping deleted successfully.");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema ${data.inputs.schemaNameCSV} can't be deleted because it is
associated with workflow ${data.inputs.workflowName}, which is still running.
Please confirm this workflow is finished in the AWS Management Console, then
delete it manually.`,
        );
        throw caught;
      }
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
```

```
      state.confirmDeleteResources === "",
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Location Service Use demo. We hope you "
 +
    "learned something new, or got some inspiration for your own apps today!" +
    " For more Amazon Location Services examples in different programming
 languages, have a look at: " +
    "https://docs.aws.amazon.com/code-library/latest/ug/
location_code_examples.html",
);

const myScenario = new Scenario("Entity Resolution Basics Scenario", [
  greet,
  pressEnter,
  displayBuildCloudFormationStack,
  sdkBuildCloudFormationStack,
  pressEnter,
  displayCreateSchemaMapping,
  sdkCreateSchemaMapping,
  pressEnter,
  displayCreateMatchingWorkflow,
  sdkCreateMatchingWorkflow,
  pressEnter,
  displayMatchingJobOfWorkflow,
  sdkMatchingJobOfWorkflow,
  pressEnter,
  displayGetDetailsforJob,
  sdkGetDetailsforJob,
  pressEnter,
  displayGetSchemaMappingJson,
  sdkGetSchemaMappingJson,
  pressEnter,
  displayListSchemaMappings,
  sdkListSchemaMappings,
  pressEnter,
  displayTagTheJsonSchema,
  sdkTagTheJsonSchema,
  pressEnter,
  displayGetJobInfo,
  pressEnter,
```

```
  displayDeleteResources,
  pressEnter,
  sdkDeleteResources,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateMatchingWorkflow
  - CreateSchemaMapping
  - DeleteMatchingWorkflow
  - DeleteSchemaMapping
  - GetMatchingJob
  - GetSchemaMapping
  - ListMatchingWorkflows
  - ListSchemaMappings
  - StartMatchingJob
  - TagResource

Ações para AWS Entity Resolution usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Entity Resolution ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Entity Resolution.

Exemplos

- Use CreateMatchingWorkflowcom um AWS SDK
- Use CreateSchemaMappingcom um AWS SDK
- Use DeleteMatchingWorkflowcom um AWS SDK
- Use DeleteSchemaMappingcom um AWS SDK
- Use GetMatchingJobcom um AWS SDK
- Use GetSchemaMappingcom um AWS SDK
- Use ListSchemaMappingscom um AWS SDK
- Use StartMatchingJobcom um AWS SDK
- Use TagResourcecom um AWS SDK

Use **`CreateMatchingWorkflow`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateMatchingWorkflow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
    *
    * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
    * @param workflowName           the name of the workflow to be created
    * @param outputBucket           the S3 bucket path where the workflow
output will be stored
    * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to
be used as the input source
    * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
    * @return a CompletableFuture that, when completed, will return the ARN of
the created workflow
    */
   public CompletableFuture<String> createMatchingWorkflowAsync(
       String roleARN
       , String workflowName
       , String outputBucket
       , String jsonGlueTableArn
       , String jsonErSchemaMappingName
       , String csvGlueTableArn
       , String csvErSchemaMappingName) {

       InputSource jsonInputSource = InputSource.builder()
           .inputSourceARN(jsonGlueTableArn)
           .schemaName(jsonErSchemaMappingName)
           .applyNormalization(false)
           .build();

       InputSource csvInputSource = InputSource.builder()
```

```
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
```

```
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please
check input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }
```

- Para obter detalhes da API, consulte CreateMatchingWorkflowa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  CreateMatchingWorkflowCommand,
```

```
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const createMatchingWorkflowParams = {
    roleArn: `${data.inputs.roleArn}`,
    workflowName: `${data.inputs.workflowName}`,
    description: "Created by using the AWS SDK for JavaScript (v3).",
    inputSourceConfig: [
      {
        inputSourceARN: `${data.inputs.JSONinputSourceARN}`,
        schemaName: `${data.inputs.schemaNameJson}`,
        applyNormalization: false,
      },
      {
        inputSourceARN: `${data.inputs.CSVinputSourceARN}`,
        schemaName: `${data.inputs.schemaNameCSV}`,
        applyNormalization: false,
      },
    ],
    outputSourceConfig: [
      {
        outputS3Path: `s3://${data.inputs.myBucketName}/eroutput`,
        output: [
          {
            name: "id",
          },
          {
            name: "name",
          },
          {
            name: "email",
          },
          {
            name: "phone",
          },
        ],
        applyNormalization: false,
      },
    ],
```

```
    resolutionTechniques: { resolutionType: "ML_MATCHING" },
  };
  try {
    const command = new CreateMatchingWorkflowCommand(
      createMatchingWorkflowParams,
    );
    const response = await erClient.send(command);

    console.log(
      `Workflow created successfully.\n The workflow ARN is:
 ${response.workflowArn}`,
    );
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte CreateMatchingWorkflowa Referência AWS SDK para JavaScript da API.

Use **`CreateSchemaMapping`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateSchemaMapping`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to create
    * @return a {@link CompletableFuture} that represents the asynchronous
creation of the schema mapping
    */
   public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
       List<SchemaInputAttribute> schemaAttributes = null;
       if (schemaName.startsWith("json")) {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.U

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeTy

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttribute
           );
       } else {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.U

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeTy

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttribute

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).
           );
       }

       CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .mappedInputFields(schemaAttributes)
           .build();

       return getResolutionAsyncClient().createSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("[{}] schema mapping Created Successfully!",
schemaName);
               } else {
```

```
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
 while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema
 mapping already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema
 mapping: " + exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateSchemaMappinga Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  CreateSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
```

```
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const createSchemaMappingParamsJson = {
    schemaName: `${data.inputs.schemaNameJson}`,
    mappedInputFields: [
      {
        fieldName: "id",
        type: "UNIQUE_ID",
      },
      {
        fieldName: "name",
        type: "NAME",
      },
      {
        fieldName: "email",
        type: "EMAIL_ADDRESS",
      },
    ],
  };
  const createSchemaMappingParamsCSV = {
    schemaName: `${data.inputs.schemaNameCSV}`,
    mappedInputFields: [
      {
        fieldName: "id",
        type: "UNIQUE_ID",
      },
      {
        fieldName: "name",
        type: "NAME",
      },
      {
        fieldName: "email",
        type: "EMAIL_ADDRESS",
      },
      {
        fieldName: "phone",
        type: "PROVIDER_ID",
        subType: "STRING",
      },
    ],
```

```
  };
  try {
    const command = new CreateSchemaMappingCommand(
      createSchemaMappingParamsJson,
    );
    const response = await erClient.send(command);
    console.log("The JSON schema mapping name is ", response.schemaName);
  } catch (error) {
    console.log("error ", error.message);
  }
};
```

- Para obter detalhes da API, consulte CreateSchemaMappinga Referência AWS SDK para JavaScript da API.

Use **`DeleteMatchingWorkflow`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteMatchingWorkflow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
```

```
    * @return a {@link CompletableFuture} that completes when the workflow has
been deleted
    * @throws RuntimeException if the deletion of the workflow fails
    */
   public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
       DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().deleteMatchingWorkflow(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("{} was deleted", workflowName );
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The workflow to delete was
not found.", cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
               }
           });
   }
```

- Para obter detalhes da API, consulte DeleteMatchingWorkflowa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  DeleteMatchingWorkflowCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  try {
    const deleteWorkflowParams = {
      workflowName: `${data.inputs.workflowName}`,
    };
    const command = new DeleteMatchingWorkflowCommand(deleteWorkflowParams);
    const response = await erClient.send(command);
    console.log("Workflow deleted successfully!", response);
  } catch (error) {
    console.log("error ", error);
  }
};
```

- Para obter detalhes da API, consulte DeleteMatchingWorkflowa Referência AWS SDK para JavaScript da API.

Use **`DeleteSchemaMapping`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteSchemaMapping`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes the schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to delete
    * @return a {@link CompletableFuture} that completes when the schema mapping
is deleted successfully,
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
       DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().deleteSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully deleted the schema mapping, log the success
message.
                   logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
               } else {
                   // Ensure exception is not null before accessing its cause.
```

```
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
 while deleting the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The schema mapping was not
 found to delete: " + schemaName, cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema
 mapping: " + schemaName, exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteSchemaMappinga Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  DeleteSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
```

```
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const deleteSchemaMapping = {
    schemaName: `${data.inputs.schemaNameJson}`,
  };
  try {
    const command = new DeleteSchemaMappingCommand(deleteSchemaMapping);
    const response = await erClient.send(command);
    console.log("Schema mapping deleted successfully. ", response);
  } catch (error) {
    console.log("error ", error);
  }
};
```

- Para obter detalhes da API, consulte DeleteSchemaMappinga Referência AWS SDK para JavaScript da API.

Use **`GetMatchingJob`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetMatchingJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
    *
    * @param jobId        the ID of the job to retrieve
    * @param workflowName the name of the workflow associated with the job
    * @return a {@link CompletableFuture} that completes when the job
information is available or an exception occurs
    */
   public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
       GetMatchingJobRequest request = GetMatchingJobRequest.builder()
           .jobId(jobId)
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().getMatchingJob(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully fetched the matching job details, log the job
status.
                   logger.info("Job status: " + response.status());
                   logger.info("Job details: " + response.toString());
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested job could
not be found.", cause);
                   }

                   // Wrap other exceptions in a CompletionException with the
message.
                   throw new CompletionException("Error fetching matching job: "
+ exception.getMessage(), exception);
               }
           });
   }
```

- Para obter detalhes da API, consulte GetMatchingJoba Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  GetMatchingJobCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  async function getInfo() {
    const getJobInfoParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${data.inputs.jobId}`,
    };
    try {
      const command = new GetMatchingJobCommand(getJobInfoParams);
      const response = await erClient.send(command);
      console.log(`Job status: ${response.status}`);
    } catch (error) {
      console.log("error ", error.message);
    }
  }
```

```
};
```

- Para obter detalhes da API, consulte GetMatchingJoba Referência AWS SDK para JavaScript da API.

Use **`GetSchemaMapping`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetSchemaMapping`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to retrieve the mapping for
    * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
    * is complete
    * @throws RuntimeException if the schema mapping retrieval fails
    */
   public CompletableFuture<GetSchemaMappingResponse>
getSchemaMappingAsync(String schemaName) {
       GetSchemaMappingRequest mappingRequest =
GetSchemaMappingRequest.builder()
           .schemaName(schemaName)
```

```
            .build();

        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema
mapping was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte GetSchemaMappinga Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  GetSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const getSchemaMappingJsonParams = {
    schemaName: `${data.inputs.schemaNameJson}`,
  };
  try {
    const command = new GetSchemaMappingCommand(getSchemaMappingJsonParams);
    const response = await erClient.send(command);
    console.log(response);
    console.log(
      `Schema mapping for the JSON data:\n ${response.mappedInputFields[0]}`,
    );
    console.log("Schema mapping ARN is: ", response.schemaArn);
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte GetSchemaMappinga Referência AWS SDK para JavaScript da API.

Use **`ListSchemaMappings`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListSchemaMappings`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
    * retrieve the schema mappings, and prints the name of each schema mapping
to the console.
    */
   public void ListSchemaMappings() {
       ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
           .build();

       ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

       // Iterate through the pages of results
       CompletableFuture<Void> future = paginator.subscribe(response -> {
           response.schemaList().forEach(schemaMapping ->
               logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
           );
       });

       // Wait for the asynchronous operation to complete
       future.join();
   }
```

- Para obter detalhes da API, consulte ListSchemaMappingsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  ListSchemaMappingsCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  async function getInfo() {
    const listSchemaMappingsParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${data.inputs.jobId}`,
    };
    try {
      const command = new ListSchemaMappingsCommand(listSchemaMappingsParams);
      const response = await erClient.send(command);
      const noOfSchemas = response.schemaList.length;
      for (let i = 0; i < noOfSchemas; i++) {
        console.log(
          `Schema Mapping Name: ${response.schemaList[i].schemaName} `,
        );
```

```
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  }
```

- Para obter detalhes da API, consulte ListSchemaMappingsa Referência AWS SDK para JavaScript da API.

Use **`StartMatchingJob`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartMatchingJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the
matching job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
```

```
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already
running. Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }
```

- Para obter detalhes da API, consulte StartMatchingJoba Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";
import {
  StartMatchingJobCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const matchingJobOfWorkflowParams = {
    workflowName: `${data.inputs.workflowName}`,
  };
  try {
    const command = new StartMatchingJobCommand(matchingJobOfWorkflowParams);
    const response = await erClient.send(command);
    console.log(`Job ID: ${response.jobID} \n
The matching job was successfully started.`);
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte StartMatchingJoba Referência AWS SDK para JavaScript da API.

Use **`TagResource`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `TagResource`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
                } else {
                    if (exception == null) {
```

```
                        throw new CompletionException("An unknown error occurred
 while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was
 not found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: "
 + exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  TagResourceCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
```

```
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const tagResourceCommandParams = {
    resourceArn: `${data.inputs.schemaArn}`,
    tags: {
      tag1: "tag1Value",
      tag2: "tag2Value",
    },
  };
  try {
    const command = new TagResourceCommand(tagResourceCommandParams);
    const response = await erClient.send(command);
    console.log("Successfully tagged the resource.");
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK para JavaScript da API.

Exemplos de código para o OpenSearch serviço usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon OpenSearch Service com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- OpenSearch Guia do usuário do serviço — Mais informações sobre o OpenSearch serviço.
- OpenSearch Referência da API de serviço — detalhes sobre todas as ações OpenSearch de serviço disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de uso OpenSearch de serviços AWS SDKs
  - Olá, OpenSearch serviço
  - Aprenda as principais operações do Amazon OpenSearch Service usando um AWS SDK
  - Ações para o OpenSearch serviço usando AWS SDKs
    - Use AddTagscom um AWS SDK
    - Use ChangeProgresscom um AWS SDK
    - Use CreateDomaincom um AWS SDK
    - Use DeleteDomaincom um AWS SDK
    - Use DescribeDomaincom um AWS SDK
    - Use ListDomainNamescom um AWS SDK
    - Use ListTagscom um AWS SDK
    - Use UpdateDomainConfigcom um AWS SDK
- Cenários para uso OpenSearch de serviços AWS SDKs
  - Conceitos básicos dos mecanismos de pesquisa e analytics

Exemplos básicos de uso OpenSearch de serviços AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon OpenSearch Service com AWS SDKs.

Exemplos

- Olá, OpenSearch serviço

- Aprenda as principais operações do Amazon OpenSearch Service usando um AWS SDK
- Ações para o OpenSearch serviço usando AWS SDKs
  - Use AddTagscom um AWS SDK
  - Use ChangeProgresscom um AWS SDK
  - Use CreateDomaincom um AWS SDK
  - Use DeleteDomaincom um AWS SDK
  - Use DescribeDomaincom um AWS SDK
  - Use ListDomainNamescom um AWS SDK
  - Use ListTagscom um AWS SDK
  - Use UpdateDomainConfigcom um AWS SDK

Olá, OpenSearch serviço

O exemplo de código a seguir mostra como começar a usar o OpenSearch Service.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.ListVersionsRequest;
import java.util.List;
import java.util.concurrent.CompletableFuture;

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

```
 */
public class HelloOpenSearch {
    public static void main(String[] args) {
        try {
            CompletableFuture<Void> future = listVersionsAsync();
            future.join();
            System.out.println("Versions listed successfully.");
        } catch (RuntimeException e) {
            System.err.println("Error occurred while listing versions: " +
 e.getMessage());
        }
    }

    private static OpenSearchAsyncClient getAsyncClient() {
        return OpenSearchAsyncClient.builder().build();
    }

    public static CompletableFuture<Void> listVersionsAsync() {
        ListVersionsRequest request = ListVersionsRequest.builder()
            .maxResults(10)
            .build();

        return getAsyncClient().listVersions(request).thenAccept(response -> {
            List<String> versionList = response.versions();
            for (String version : versionList) {
                System.out.println("Version info: " + version);
            }
        }).exceptionally(ex -> {
            // Handle the exception, or propagate it as a RuntimeException
            throw new RuntimeException("Failed to list versions", ex);
        });
    }
}
```

- Para obter detalhes da API, consulte ListVersionsa Referência AWS SDK for Java 2.x da API.

Aprenda as principais operações do Amazon OpenSearch Service usando um AWS SDK

O exemplo de código a seguir mostra como:

- Crie um domínio OpenSearch de serviço.
- Fornece informações detalhadas sobre um domínio OpenSearch de serviço específico.
- Lista todos os domínios OpenSearch de serviço pertencentes à conta.
- Espera até que o status de alteração do domínio do OpenSearch Serviço atinja um estado concluído.
- Modifica a configuração de um domínio de OpenSearch serviço existente.
- Adicione uma tag ao domínio do OpenSearch serviço.
- Lista as tags associadas a um domínio OpenSearch de serviço.
- Remove as tags de um domínio OpenSearch de serviço.
- Exclui o domínio do OpenSearch serviço.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos OpenSearch do Serviço.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.opensearch.model.*;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

public class OpenSearchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchScenario.class);
    static Scanner scanner = new Scanner(System.in);
```

```
    static OpenSearchActions openSearchActions = new OpenSearchActions();

    public static void main(String[] args) throws Throwable {
        logger.info("""
            Welcome to the Amazon OpenSearch Service Basics Scenario.

            Use the Amazon OpenSearch Service API to create, configure, and
 manage OpenSearch Service domains.

            The operations exposed by the AWS OpenSearch Service client are
 focused on managing the OpenSearch Service domains
            and their configurations, not the data within the domains (such as
 indexing or querying documents).
            For document management, you typically interact directly with the
 OpenSearch REST API or use other libraries,
            such as the OpenSearch Java client (https://opensearch.org/docs/
latest/clients/java/).

            Let's get started...
        """);
        waitForInputToContinue(scanner);
        try {
            runScenario();
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
```

```
    private static void runScenario() throws Throwable {
        String currentTimestamp = String.valueOf(System.currentTimeMillis());
        String domainName = "test-domain-" + currentTimestamp;

        logger.info(DASHES);
        logger.info("1. Create an Amazon OpenSearch domain");
        logger.info("""
            An Amazon OpenSearch domain is a managed instance of the OpenSearch
engine,
            which is an open-source search and analytics engine derived from
Elasticsearch.
            An OpenSearch domain is essentially a cluster of compute resources
and storage that hosts
            one or more OpenSearch indexes, enabling you to perform full-text
searches, data analysis, and
            visualizations.

            In this step, we'll initiate the creation of the domain. We'll check
on the progress in a later step.
        """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
openSearchActions.createNewDomainAsync(domainName);
            String domainId = future.join();
            logger.info("Domain successfully created with ID: {}", domainId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause != null) {
                if (cause instanceof OpenSearchException openSearchEx) {
                    logger.error("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
                } else {
                    logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("2. Describe the Amazon OpenSearch domain");
        logger.info("In this step, we get back the Domain ARN which is used in an
upcoming step.");
        waitForInputToContinue(scanner);

        String arn = "";
        try {
            CompletableFuture<String> future =
openSearchActions.describeDomainAsync(domainName);
            arn = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("3. List the domains in your account");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<List<DomainInfo>> future =
openSearchActions.listAllDomainsAsync();
            List<DomainInfo> domainInfoList = future.join();
            for (DomainInfo domain : domainInfoList) {
                logger.info("Domain name is: " + domain.domainName());
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
```

```
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we check on the change status of the domain that we
initiated in Step 1.
            Until we reach a COMPLETED state, we stay in a loop by sending a
DescribeDomainChangeProgressRequest.

            The time it takes for a change to an OpenSearch domain to reach a
completed state can range
            from a few minutes to several hours. In this case the change is
creating a new domain that we initiated in Step 1.
            The time varies depending on the complexity of the change and the
current load on
            the OpenSearch service. In general, simple changes, such as scaling
the number of data nodes or
            updating the OpenSearch version, may take 10-30 minutes.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
ResourceNotFoundException)) {
                cause = cause.getCause();
            }
```

```
            if (cause instanceof ResourceNotFoundException
resourceNotFoundException) {
                logger.info("The specific AWS resource
was not found: Error message: {}, Error code {}",
resourceNotFoundException.awsErrorDetails().errorMessage(),
resourceNotFoundException.awsErrorDetails().errorCode());

                if (cause instanceof OpenSearchException ex) {
                    logger.info("An OpenSearch error occurred: Error message: " +
ex.getMessage());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Modify the domain");
        logger.info("""
            You can change your OpenSearch domain's settings, like the number of
instances, without starting over from scratch.
            This makes it easy to adjust your domain as your needs change,
allowing you to scale up or
            down quickly without recreating everything.

            We modify the domain in this step by changing the number of
instances.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<UpdateDomainConfigResponse> future =
openSearchActions.updateSpecificDomainAsync(domainName);
            UpdateDomainConfigResponse updateResponse = future.join();
            logger.info("Domain update status: " +
updateResponse.domainConfig().changeProgressDetails().configChangeStatusAsString());
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
```

```
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we poll the status until the domain's change status
reaches a completed state.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException ex) {
                logger.info("EC2 error occurred: Error message: "
+ex.getMessage());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("7. Tag the Domain");
        logger.info("""
            Tags let you assign arbitrary information to an Amazon OpenSearch
Service domain so you can
            categorize and filter on that information. A tag is a key-value pair
that you define and
```

```
            associate with an OpenSearch Service domain. You can use these tags
to track costs by grouping
            expenses for similarly tagged resources.

            In this scenario, we create tags with keys "service" and "instances".
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<AddTagsResponse> future =
openSearchActions.addDomainTagsAsync(arn);
            future.join();
            logger.info("Domain tags added successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                if (cause != null) {
                    if (cause instanceof OpenSearchException) {
                        logger.error("OpenSearch error occurred: Error message: "
+ cause.getMessage(), cause);
                    } else {
                        logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                    }
                } else {
                    logger.error("An unexpected error occurred: " +
rt.getMessage(), rt);
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info("8. List Domain tags");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<ListTagsResponse> future =
openSearchActions.listDomainTagsAsync(arn);
            ListTagsResponse listTagsResponse = future.join();
            listTagsResponse.tagList().forEach(tag -> logger.info("Tag Key: " +
tag.key() + ", Tag Value: " + tag.value()));
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Delete the domain");
        logger.info("""
            In this step, we'll delete the Amazon OpenSearch domain that we
created in Step 1.
            Deleting a domain will remove all data and configuration for that
domain.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<DeleteDomainResponse> future =
openSearchActions.deleteSpecificDomainAsync(domainName);
            future.join();
            logger.info("Domain successfully deleted.");
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
 OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
 {}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
 openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("Scenario complete!");
    }
 }
```

Uma classe wrapper para métodos do OpenSearch Service SDK.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.AddTagsRequest;
import software.amazon.awssdk.services.opensearch.model.AddTagsResponse;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressRequest;
```

```
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.DomainStatus;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsResponse;
import
 software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.Tag;
import
 software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import
 software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class OpenSearchActions {
    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchActions.class);
    private static OpenSearchAsyncClient openSearchClientAsyncClient;
    private static OpenSearchAsyncClient getAsyncClient() {
        if (openSearchClientAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();
```

```
            openSearchClientAsyncClient = OpenSearchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return openSearchClientAsyncClient;
    }

    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
```

```
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }

    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has
been deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse>
deleteSpecificDomainAsync(String domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }

    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the
domain
     * @throws RuntimeException if the domain description fails
     */
```

```
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }

    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list
of {@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain
names on success
```

```
            });
    }

    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }

    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress
for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();
```

```
        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
            long startTime = System.currentTimeMillis();

            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response =
getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check
domain progress", ex);
                            }
                            return resp;
                        }).join();

                    String state =
response.changeProgressStatus().statusAsString();  // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds =
(System.currentTimeMillis() - startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " +
state + " | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
```

```
    }

    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String
domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain:
" + domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }


    /**
```

```
     * Asynchronously lists the tags associated with the specified Amazon
 Resource Name (ARN).
     * @param arn the Amazon Resource Name (ARN) of the resource for which to
 list the tags
     * @return a {@link CompletableFuture} that, when completed, will contain a
 list of the tags associated with the
     * specified ARN
     * @throws RuntimeException if there is an error listing the tags
     */
    public CompletableFuture<ListTagsResponse> listDomainTagsAsync(String arn) {
        ListTagsRequest tagsRequest = ListTagsRequest.builder()
            .arn(arn)
            .build();

        return getAsyncClient().listTags(tagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list domain tags",
 exception);
                }

                List<Tag> tagList = response.tagList();
                for (Tag tag : tagList) {
                    logger.info("Tag key is " + tag.key());
                    logger.info("Tag value is " + tag.value());
                }
            });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AddTags
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - DescribeDomainChangeProgress
  - ListDomainNames
  - ListTags

- UpdateDomainConfig

Ações para o OpenSearch serviço usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações OpenSearch de serviço individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API de OpenSearch serviço e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para uso OpenSearch de serviços AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon OpenSearch Service API Reference.

Exemplos

- Use AddTagscom um AWS SDK
- Use ChangeProgresscom um AWS SDK
- Use CreateDomaincom um AWS SDK
- Use DeleteDomaincom um AWS SDK
- Use DescribeDomaincom um AWS SDK
- Use ListDomainNamescom um AWS SDK
- Use ListTagscom um AWS SDK
- Use UpdateDomainConfigcom um AWS SDK

Use **`AddTags`**com um AWS SDK

O código de exemplo a seguir mostra como usar `AddTags`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça as principais operações do OpenSearch serviço

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously adds tags to an Amazon OpenSearch Service domain.
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String
domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();

       return getAsyncClient().addTags(addTagsRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
```

```
                    throw new RuntimeException("Failed to add tags to the domain:
" + domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- Para obter detalhes da API, consulte AddTagsa Referência AWS SDK for Java 2.x da API.

Use **ChangeProgress**com um AWS SDK

O código de exemplo a seguir mostra como usar `ChangeProgress`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress
for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
```

```
            long startTime = System.currentTimeMillis();

            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response =
getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check
domain progress", ex);
                            }
                            return resp;
                        }).join();

                    String state =
response.changeProgressStatus().statusAsString();  // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds =
(System.currentTimeMillis() - startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " +
state + " | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }
```

- Para obter detalhes da API, consulte ChangeProgressa Referência AWS SDK for Java 2.x da API.

Use **`CreateDomain`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDomain`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos dos mecanismos de pesquisa e analytics
- Conheça as principais operações do OpenSearch serviço

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
```

```
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }
```

- Para obter detalhes da API, consulte CreateDomaina Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }

    val ebsOptionsOb =
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptionsOb =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
            clusterConfig = clusterConfigOb
            ebsOptions = ebsOptionsOb
            nodeToNodeEncryptionOptions = encryptionOptionsOb
        }
```

```
    println("Sending domain creation request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient
 ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- Para obter detalhes da API, consulte a CreateDomainreferência da API AWS SDK for Kotlin.

Use **`DeleteDomain`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteDomain`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos dos mecanismos de pesquisa e analytics
- Conheça as principais operações do OpenSearch serviço

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has
 been deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
```

```
    public CompletableFuture<DeleteDomainResponse>
 deleteSpecificDomainAsync(String domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
 domainName, exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteDomaina Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient
 ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteDomainreferência da API AWS SDK for Kotlin.

Use **`DescribeDomain`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeDomain`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos dos mecanismos de pesquisa e analytics
- Conheça as principais operações do OpenSearch serviço

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the
domain
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
```

```
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }
```

- Para obter detalhes da API, consulte DescribeDomaina Referência AWS SDK for Java 2.x da API.

Use **`ListDomainNames`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListDomainNames`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça as principais operações do OpenSearch serviço

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list
 of {@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
 exception);
                }
                return response.domainNames();  // Return the list of domain
 names on success
            });
    }
```

- Para obter detalhes da API, consulte ListDomainNamesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllDomains() {
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient
 ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
```

```
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- Para obter detalhes da API, consulte a ListDomainNamesreferência da API AWS SDK for Kotlin.

Use **`ListTags`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListTags`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça as principais operações do OpenSearch serviço

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously adds tags to an Amazon OpenSearch Service domain.
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String
domainARN) {
       Tag tag1 = Tag.builder()
```

```
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain:
" + domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- Para obter detalhes da API, consulte ListTagsa Referência AWS SDK for Java 2.x da API.

Use **`UpdateDomainConfig`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `UpdateDomainConfig`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça as principais operações do OpenSearch serviço

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates the configuration of a specific domain asynchronously.
    * @param domainName the name of the domain to update
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
    */
   public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
       ClusterConfig clusterConfig = ClusterConfig.builder()
           .instanceCount(3)
           .build();

       UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
           .domainName(domainName)
           .clusterConfig(clusterConfig)
           .build();

       return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to update the domain
configuration", exception);
               }
               // Handle success if needed (e.g., logging or additional actions)
           });
   }
```

- Para obter detalhes da API, consulte UpdateDomainConfiga Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient
 ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
}
```

- Para obter detalhes da API, consulte a UpdateDomainConfigreferência da API AWS SDK for Kotlin.

Cenários para uso OpenSearch de serviços AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no OpenSearch Service with AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no OpenSearch Serviço ou combinadas com outras Serviços da AWS. Cada cenário

inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos dos mecanismos de pesquisa e analytics

Conceitos básicos dos mecanismos de pesquisa e analytics

O exemplo de código a seguir mostra como:

- Criar um domínio OpenSearch de serviço
- Carregar dados em seu domínio.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md

# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
```

```
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Track if domain was successfully created
DOMAIN_CREATED=false
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [[ "$DOMAIN_CREATED" == "true" ]]; then
        echo "Checking if domain $DOMAIN_NAME exists before attempting to
 delete..."

        # Check if domain exists before trying to delete
```

```
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/
null 2>&1; then
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
}

# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
```

```
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
# This policy restricts access to specific actions and should be further
 restricted to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for
 completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack
 creation failed"

DOMAIN_CREATED=true
echo "Domain created successfully via CloudFormation."

# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
```

```
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120

# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo
 "No"; fi)"

# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
```

```
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel",
 "Malahide, Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas",
 "Stellate, Nick", "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese",
 "Spielbauer, Bruce", "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett,
 Charlie"], "title": "U.S. Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955,
 "actor": ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal",
 "Backus, Jim", "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen,
 Corey", "Birch, Paul", "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh,
 Nelson", "Williams, Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira",
 "McMahon, David", "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"

    # Test 1: Basic authentication with root endpoint
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
```

```
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)

    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }

    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

        if is_success_code "$HEALTH_HTTP_CODE"; then
            echo "# Cluster health authentication test successful! (HTTP
 $HEALTH_HTTP_CODE)"
```

```
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
            echo "Cluster health authentication also failed with HTTP code:
 $HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
 then
                echo "Error: Request is being treated as anonymous
 (authentication not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential
 backoff..."

            # Retry authentication test with exponential backoff
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT
+1))/$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

                echo "Retry result (HTTP $HTTP_CODE):"
                echo "$AUTH_TEST_RESULT"

                if is_success_code "$HTTP_CODE"; then
```

```
                    echo "# Authentication test successful after retry! (HTTP
$HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:"
| cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
(HTTP $HEALTH_HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding
with data operations."

        # Upload single document (matches tutorial exactly)
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
```

```
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
 '"result"'; then
            echo "# Single document uploaded successfully! (HTTP
 $UPLOAD_HTTP_CODE)"
        else
            echo "# Warning: Single document upload may have failed (HTTP
 $UPLOAD_HTTP_CODE)"
        fi

        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
 '"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
 $BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
        echo "Waiting for documents to be indexed..."
        sleep 5
```

```
        # Step 3: Search Documents (matches tutorial exactly)
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
 '"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "==========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "==========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."
```

```
        # Debug domain configuration
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-
name "$DOMAIN_NAME" --query 'DomainStatus.{AdvancedSecurityOptions:
 AdvancedSecurityOptions, AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user
 works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi
```

```
# Display OpenSearch Dashboards URL (matches tutorial)
echo ""
echo "============================================"
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================"
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
```

```
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
echo "Cleaning up temporary files..."

# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/
_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateDomain
  - DeleteDomain

- DescribeDomain
- GetCallerIdentity

Exemplos de código para EventBridge usar AWS SDKs

Os exemplos de código a seguir mostram como usar a Amazon EventBridge com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- EventBridge Guia do usuário — Mais informações sobre EventBridge.
- EventBridge Referência da API — Detalhes sobre todas as EventBridge ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de EventBridge uso AWS SDKs
  - Olá EventBridge
  - Aprenda o básico sobre EventBridge com um AWS SDK
  - Ações para EventBridge usar AWS SDKs
    - Use DeleteRulecom um AWS SDK ou CLI
    - Use DescribeRulecom um AWS SDK ou CLI
    - Use DisableRulecom um AWS SDK ou CLI

    - Use EnableRulecom um AWS SDK ou CLI
    - Use ListRuleNamesByTargetcom um AWS SDK ou CLI
    - Use ListRulescom um AWS SDK ou CLI
    - Use ListTargetsByRulecom um AWS SDK ou CLI
    - Use PutEventscom um AWS SDK ou CLI
    - Use PutRulecom um AWS SDK ou CLI
    - Use PutTargetscom um AWS SDK ou CLI
    - Use RemoveTargetscom um AWS SDK ou CLI
- Cenários para EventBridge usar AWS SDKs
  - Crie e acione uma regra na Amazon EventBridge usando um AWS SDK
  - Envie notificações de eventos do S3 para a Amazon EventBridge usando um AWS SDK
  - Usar eventos programados para chamar uma função do Lambda

Exemplos básicos de EventBridge uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico da Amazon EventBridge com AWS SDKs.

Exemplos

- Olá EventBridge
- Aprenda o básico sobre EventBridge com um AWS SDK
- Ações para EventBridge usar AWS SDKs
  - Use DeleteRulecom um AWS SDK ou CLI
  - Use DescribeRulecom um AWS SDK ou CLI
  - Use DisableRulecom um AWS SDK ou CLI
  - Use EnableRulecom um AWS SDK ou CLI
  - Use ListRuleNamesByTargetcom um AWS SDK ou CLI
  - Use ListRulescom um AWS SDK ou CLI
  - Use ListTargetsByRulecom um AWS SDK ou CLI
  - Use PutEventscom um AWS SDK ou CLI
  - Use PutRulecom um AWS SDK ou CLI
  - Use PutTargetscom um AWS SDK ou CLI

- Use RemoveTargetscom um AWS SDK ou CLI

Olá EventBridge

O exemplo de código a seguir mostra como começar a usar o EventBridge.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.EventBridge;
using Amazon.EventBridge.Model;

namespace EventBridgeActions;

public static class HelloEventBridge
{
    static async Task Main(string[] args)
    {
        var eventBridgeClient = new AmazonEventBridgeClient();

        Console.WriteLine($"Hello Amazon EventBridge! Following are some of your
 EventBuses:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five event buses.
        var response = await eventBridgeClient.ListEventBusesAsync(
            new ListEventBusesRequest()
            {
                Limit = 5
            });

        foreach (var eventBus in response.EventBuses)
```

```
        {
            Console.WriteLine($"\tEventBus: {eventBus.Name}");
            Console.WriteLine($"\tArn: {eventBus.Arn}");
            Console.WriteLine($"\tPolicy: {eventBus.Policy}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte ListEventBusesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        listBuses(eventBrClient);
```

```
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                    .limit(10)
                    .build();

            ListEventBusesResponse response =
 eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
            for (EventBus bus : buses) {
                System.out.println("The name of the event bus is: " +
 bus.name());
                System.out.println("The ARN of the event bus is: " + bus.arn());
            }

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

- Para obter detalhes da API, consulte ListEventBusesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
```

```
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request =
        ListEventBusesRequest {
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-west-2" }.use
 { eventBrClient ->
        val response: ListEventBusesResponse =
 eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListEventBusesreferência da API AWS SDK for Kotlin.

Aprenda o básico sobre EventBridge com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar uma regra e adicionar um destino a ela.
- Habilitar e desabilitar regras.
- Listar e atualizar regras e destinos.
- Envie eventos e, em seguida, limpe os recursos.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
public class EventBridgeScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks with Amazon EventBridge:
    - Create a rule.
    - Add a target to a rule.
    - Enable and disable rules.
    - List rules and targets.
    - Update rules and targets.
    - Send events.
    - Delete the rule.
    */

    private static ILogger logger = null!;
    private static EventBridgeWrapper _eventBridgeWrapper = null!;
    private static IConfiguration _configuration = null!;

    private static IAmazonIdentityManagementService? _iamClient = null!;
    private static IAmazonSimpleNotificationService? _snsClient = null!;
    private static IAmazonS3 _s3Client = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonEventBridge>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddAWSService<IAmazonS3>()
            .AddAWSService<IAmazonSimpleNotificationService>()
            .AddTransient<EventBridgeWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<EventBridgeScenario>();

        ServicesSetup(host);

        string topicArn = "";
        string roleArn = "";

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            roleArn = await CreateRole();

            await CreateBucketWithEventBridgeEvents();

            await AddEventRule(roleArn);

            await ListEventRules();

            topicArn = await CreateSnsTopic();
```

```
            var email = await SubscribeToSnsTopic(topicArn);

            await AddSnsTarget(topicArn);

            await ListTargets();

            await ListRulesForTarget(topicArn);

            await UploadS3File(_s3Client);

            await ChangeRuleState(false);

            await GetRuleState();

            await UpdateSnsEventRule(topicArn);

            await ChangeRuleState(true);

            await UploadS3File(_s3Client);

            await UpdateToCustomRule(topicArn);

            await TriggerCustomRule(email);

            await CleanupResources(topicArn);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources(topicArn);
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon EventBridge example scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
```

```
        _eventBridgeWrapper =
host.Services.GetRequiredService<EventBridgeWrapper>();
        _snsClient =
host.Services.GetRequiredService<IAmazonSimpleNotificationService>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    }

    /// <summary>
    /// Create a role to be used by EventBridge.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateRole()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating a role to use with EventBridge and attaching
managed policy AmazonEventBridgeFullAccess.");
        Console.WriteLine(new string('-', 80));

        var roleName = _configuration["roleName"];

        var assumeRolePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                "\"Effect\": \"Allow\"," +
                                "\"Principal\": {" +
                                $"\"Service\": \"events.amazonaws.com\"" +
                                "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                                "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = roleName
            });

        await _iamClient.AttachRolePolicyAsync(
            new AttachRolePolicyRequest()
            {
```

```
                PolicyArn = "arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess",
                RoleName = roleName
            });
        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Create an Amazon Simple Storage Service (Amazon S3) bucket with
 EventBridge events enabled.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateBucketWithEventBridgeEvents()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an S3 bucket with EventBridge events
 enabled.");

        var testBucketName = _configuration["testBucketName"];

        var bucketExists = await
 Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            testBucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = testBucketName,
                UseClientRegion = true
            });
        }

        await _s3Client.PutBucketNotificationAsync(new
 PutBucketNotificationRequest()
        {
            BucketName = testBucketName,
            EventBridgeConfiguration = new EventBridgeConfiguration()
        });

        Console.WriteLine($"\tAdded bucket {testBucketName} with EventBridge
 events enabled.");
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and upload a file to an S3 bucket to trigger an event.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UploadS3File(IAmazonS3 s3Client)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Uploading a file to the test bucket. This will trigger
 a subscription email.");

        var testBucketName = _configuration["testBucketName"];

        var fileName = $"example_upload_{DateTime.UtcNow.Ticks}.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for testing uploads.");
        }

        await s3Client.PutObjectAsync(new PutObjectRequest()
        {
            FilePath = fileName,
            BucketName = testBucketName
        });

        Console.WriteLine($"\tPress Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an Amazon Simple Notification Service (Amazon SNS) topic to use as
 an EventBridge target.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> CreateSnsTopic()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "Creating an Amazon Simple Notification Service (Amazon SNS) topic
for email subscriptions.");

        var topicName = _configuration["topicName"];

        string topicPolicy = "{" +
                             "\"Version\": \"2012-10-17\"," +
                             "\"Statement\": [{" +
                             "\"Sid\": \"EventBridgePublishTopic\"," +
                             "\"Effect\": \"Allow\"," +
                             "\"Principal\": {" +
                             $"\"Service\": \"events.amazonaws.com\"" +
                             "}," +
                             "\"Resource\": \"*\"," +
                             "\"Action\": \"sns:Publish\"" +
                             "}]" +
                             "}";

        var topicAttributes = new Dictionary<string, string>()
        {
            { "Policy", topicPolicy }
        };

        var topicResponse = await _snsClient!.CreateTopicAsync(new
CreateTopicRequest()
        {
            Name = topicName,
            Attributes = topicAttributes

        });

        Console.WriteLine($"\tAdded topic {topicName} for email subscriptions.");

        Console.WriteLine(new string('-', 80));

        return topicResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a user email to an SNS topic.
    /// </summary>
```

```
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>The user's email.</returns>
    private static async Task<string> SubscribeToSnsTopic(string topicArn)
    {
        Console.WriteLine(new string('-', 80));


        string email = "";
        while (string.IsNullOrEmpty(email))
        {
            Console.WriteLine("Enter your email to subscribe to the Amazon SNS
topic:");
            email = Console.ReadLine()!;
        }

        var subscriptions = new List<string>();
        var paginatedSubscriptions =
_snsClient!.Paginators.ListSubscriptionsByTopic(
            new ListSubscriptionsByTopicRequest()
            {
                TopicArn = topicArn
            });

        // Get the entire list using the paginator.
        await foreach (var subscription in paginatedSubscriptions.Subscriptions)
        {
            subscriptions.Add(subscription.Endpoint);
        }

        if (subscriptions.Contains(email))
        {
            Console.WriteLine($"\tYour email is already subscribed.");
            Console.WriteLine(new string('-', 80));
            return email;
        }

        await _snsClient.SubscribeAsync(new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "email",
            Endpoint = email
        });
```

```
        Console.WriteLine($"Use the link in the email you received to confirm
your subscription, then press Enter to continue.");

        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
        return email;
    }

    /// <summary>
    /// Add a rule which triggers when a file is uploaded to an S3 bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role used by EventBridge.</param>
    /// <returns>Async task.</returns>
    private static async Task AddEventRule(string roleArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an EventBridge event that sends an email when
an Amazon S3 object is created.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await _eventBridgeWrapper.PutS3UploadRule(roleArn, eventRuleName,
testBucketName);
        Console.WriteLine($"\tAdded event rule {eventRuleName} for bucket
{testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an SNS target to the rule.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task AddSnsTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Adding a target to the rule to that sends an email
when the rule is triggered.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];
```

```
        var topicName = _configuration["topicName"];
        await _eventBridgeWrapper.AddSnsTargetToRule(eventRuleName, topicArn);
        Console.WriteLine($"\tAdded event rule {eventRuleName} with Amazon SNS
target {topicName} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the event rules on the default event bus.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListEventRules()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Current event rules:");

        var rules = await _eventBridgeWrapper.ListAllRulesForEventBus();
        rules.ForEach(r => Console.WriteLine($"\tRule: {r.Name} Description:
{r.Description} State: {r.State}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the event target to use a transform.
    /// </summary>
    /// <param name="topicArn">The SNS topic ARN target to update.</param>
    /// <returns>Async task.</returns>
    private static async Task UpdateSnsEventRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Let's update the event target with a transform.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await
_eventBridgeWrapper.UpdateS3UploadRuleTargetWithTransform(eventRuleName,
topicArn);
        Console.WriteLine($"\tUpdated event rule {eventRuleName} with Amazon SNS
target {topicArn} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Update the rule to use a custom event pattern.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UpdateToCustomRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Updating the event pattern to be triggered by a custom
event instead.");

        var eventRuleName = _configuration["eventRuleName"];

        await _eventBridgeWrapper.UpdateCustomEventPattern(eventRuleName);

        Console.WriteLine($"\tUpdated event rule {eventRuleName} to custom
pattern.");
        await
_eventBridgeWrapper.UpdateCustomRuleTargetWithTransform(eventRuleName,
            topicArn);

        Console.WriteLine($"\tUpdated event target {topicArn}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Send rule events for a custom rule using the user's email address.
    /// </summary>
    /// <param name="email">The email address to include.</param>
    /// <returns>Async task.</returns>
    private static async Task TriggerCustomRule(string email)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Sending an event to trigger the rule. This will
trigger a subscription email.");

        await _eventBridgeWrapper.PutCustomEmailEvent(email);

        Console.WriteLine($"\tEvents have been sent. Press Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// List all of the targets for a rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListTargets()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the targets for a particular rule.");

        var eventRuleName = _configuration["eventRuleName"];
        var targets = await
_eventBridgeWrapper.ListAllTargetsOnRule(eventRuleName);
        targets.ForEach(t => Console.WriteLine($"\tTarget: {t.Arn} Id: {t.Id}
Input: {t.Input}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List all of the rules for a particular target.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task ListRulesForTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the rules for a particular target.");

        var rules = await _eventBridgeWrapper.ListAllRuleNamesByTarget(topicArn);
        rules.ForEach(r => Console.WriteLine($"\tRule: {r}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Enable or disable a particular rule.
    /// </summary>
    /// <param name="isEnabled">True to enable the rule, otherwise false.</param>
    /// <returns>Async task.</returns>
    private static async Task ChangeRuleState(bool isEnabled)
    {
        Console.WriteLine(new string('-', 80));
```

```
        var eventRuleName = _configuration["eventRuleName"];

        if (!isEnabled)
        {
            Console.WriteLine($"Disabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.DisableRuleByName(eventRuleName);
        }
        else
        {
            Console.WriteLine($"Enabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.EnableRuleByName(eventRuleName);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the current state of the rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetRuleState()
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        var state = await
_eventBridgeWrapper.GetRuleStateByRuleName(eventRuleName);
        Console.WriteLine($"Rule {eventRuleName} is in current state {state}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic to clean up.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        var eventRuleName = _configuration["eventRuleName"];
```

```
        if (GetYesNoResponse($"\tDelete all targets and event rule
{eventRuleName}? (y/n)"))
        {
            Console.WriteLine($"\tRemoving all targets from the event rule.");
            await _eventBridgeWrapper.RemoveAllTargetsFromRule(eventRuleName);

            Console.WriteLine($"\tDeleting event rule.");
            await _eventBridgeWrapper.DeleteRuleByName(eventRuleName);
        }

        var topicName = _configuration["topicName"];
        if (GetYesNoResponse($"\tDelete Amazon SNS subscription topic
{topicName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting topic.");
            await _snsClient!.DeleteTopicAsync(new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        }

        var bucketName = _configuration["testBucketName"];
        if (GetYesNoResponse($"\tDelete Amazon S3 bucket {bucketName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.S3Objects
                    .Select(o => new KeyVersion { Key = o.Key }).ToList()
            });
            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }
```

```
        var roleName = _configuration["roleName"];
        if (GetYesNoResponse($"\tDelete role {roleName}? (y/n)"))
        {
            Console.WriteLine($"\tDetaching policy and deleting role.");

            await _iamClient!.DetachRolePolicyAsync(new DetachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = "arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess",
            });

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = roleName
            });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Crie uma classe que envolva as EventBridge operações.

```
/// <summary>
```

```
/// Wrapper for Amazon EventBridge operations.
/// </summary>
public class EventBridgeWrapper
{
    private readonly IAmazonEventBridge _amazonEventBridge;
    private readonly ILogger<EventBridgeWrapper> _logger;

    /// <summary>
    /// Constructor for the EventBridge wrapper.
    /// </summary>
    /// <param name="amazonEventBridge">The injected EventBridge client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public EventBridgeWrapper(IAmazonEventBridge amazonEventBridge,
 ILogger<EventBridgeWrapper> logger)

    {
        _amazonEventBridge = amazonEventBridge;
        _logger = logger;
    }

    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
 uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
 eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }

    /// <summary>
    /// Enable a particular rule on an event bus.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> EnableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
            new EnableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable a particular rule on an event bus.
    /// </summary
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
            new DisableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty,
 uses the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
 null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
```

```
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
```

```
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await
 _amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created
 in a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]"
 +
                                        "}" +
                                    "}" +
                                "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });
```

```
        return response.RuleArn;
    }

    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
```

```
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
 {e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Update a custom rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
 default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateCustomRuleTargetWithTransform(string
 ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputTemplate = "\"Notification: sample event was received.
\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
```

```
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }

    /// <summary>
    /// Update a rule to use a custom defined event pattern.
```

```
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }

    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string
targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };
```

```
        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });

        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);
```

```
        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage},
 code {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteRuleByName(string ruleName)
    {
        var response = await _amazonEventBridge.DeleteRuleAsync(
            new DeleteRuleRequest()
            {
                Name = ruleName
            });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - DeleteRule
  - DescribeRule

- DisableRule
- EnableRule
- ListRuleNamesByTarget
- ListRules
- ListTargetsByRule
- PutEvents
- PutRule
- PutTargets

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
```

```
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
 *
 */
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

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

                Usage:
                    <roleName> <bucketName> <topicName> <eventRuleName>

                Where:
                    roleName - The name of the role to create.
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
 bucket name to create.
                    topicName - The name of the Amazon Simple Notification
 Service (Amazon SNS) topic to create.
                    eventRuleName - The Amazon EventBridge rule name to create.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
```

```
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        SnsClient snsClient = SnsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EventBridge example
 scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out
                .println("1. Create an AWS Identity and Access Management (IAM)
 role to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the
user subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to
the specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
                "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
```

```
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email
when an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to
the S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule " + eventRuleName + " to use a
custom pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn,
eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("The Amazon EventBridge example scenario has
successfully completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient,
SnsClient snsClient, S3Client s3Client,
            IamClient iam, String topicArn, String eventRuleName, String
bucketName, String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
                .policyArn(policyArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from
role " + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3Client.listObjects(listObjects);
```

```
        List<S3Object> objects = res.contents();
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

        for (S3Object myValue : objects) {
            toDelete.add(ObjectIdentifier.builder()
                    .key(myValue.key())
                    .build());
        }

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(Delete.builder()
                        .objects(toDelete).build())
                .build();

        s3Client.deleteObjects(dor);

        // Delete the S3 bucket.
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("You have deleted the bucket and the objects");
    }

    // Delete the SNS topic.
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

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

```
   public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
       DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
               .name(ruleName)
               .build();

       eventBrClient.deleteRule(ruleRequest);
       System.out.println("Successfully deleted the rule");
   }

   public static void deleteTargetsFromRule(EventBridgeClient eventBrClient,
String eventRuleName) {
       // First, get all targets that will be deleted.
       ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
               .rule(eventRuleName)
               .build();

       ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
       List<Target> allTargets = response.targets();

       // Get all targets and delete them.
       for (Target myTarget : allTargets) {
           RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                   .rule(eventRuleName)
                   .ids(myTarget.id())
                   .build();

           eventBrClient.removeTargets(removeTargetsRequest);
           System.out.println("Successfully removed the target");
       }
   }

   public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
       String json = "{" +
               "\"UserEmail\": \"" + email + "\"," +
               "\"Message\": \"This event was generated by example code.\"," +
               "\"UtcTime\": \"Now.\"" +
               "}";

       PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
               .source("ExampleSource")
```

```
                .detail(json)
                .detailType("ExampleType")
                .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
        String customEventsPattern = "{" +
                "\"source\": [\"ExampleSource\"]," +
                "\"detail-type\": [\"ExampleType\"]" +
```

```
                "}";

        PutRuleRequest request = PutRuleRequest.builder()
                .name(ruleName)
                .description("Custom test rule")
                .eventPattern(customEventsPattern)
                .build();

        eventBrClient.putRule(request);
    }

    // Update an Amazon S3 object created rule with a transform on the target.
    public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
 topicArn, String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        Map<String, String> myMap = new HashMap<>();
        myMap.put("bucket", "$.detail.bucket.name");
        myMap.put("time", "$.time");

        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\"")
                .inputPathsMap(myMap)
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);

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

```
    }

    public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        try {
            DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

            DescribeRuleResponse response =
eventBrClient.describeRule(ruleRequest);
            System.out.println("The state of the rule is " +
response.stateAsString());

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

    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

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

```
   // Create and upload a file to an S3 bucket to trigger an event.
   public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
       // Create a unique file name.
       String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
       String fileName = "TextFile" + fileSuffix + ".txt";

       File myFile = new File(fileName);
       FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
       BufferedWriter bw = new BufferedWriter(fw);
       bw.write("This is a sample file for testing uploads.");
       bw.close();

       try {
           PutObjectRequest putOb = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(fileName)
                   .build();

           s3Client.putObject(putOb, RequestBody.fromFile(myFile));

       } catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
       ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
               .targetArn(topicArn)
               .build();

       ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
       List<String> rules = response.ruleNames();
       for (String rule : rules) {
           System.out.println("The rule name is " + rule);
       }
   }
```

```
   public static void listTargets(EventBridgeClient eventBrClient, String
ruleName) {
       ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
               .rule(ruleName)
               .build();

       ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
       List<Target> targetsList = res.targets();
       for (Target target: targetsList) {
           System.out.println("Target ARN: "+target.arn());
       }
   }

   // Add a rule which triggers an SNS target when a file is uploaded to an S3
   // bucket.
   public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
           String topicName, String eventRuleName, String bucketName) {
       String targetID = java.util.UUID.randomUUID().toString();
       Target myTarget = Target.builder()
               .id(targetID)
               .arn(topicArn)
               .build();

       List<Target> targets = new ArrayList<>();
       targets.add(myTarget);
       PutTargetsRequest request = PutTargetsRequest.builder()
               .eventBusName(null)
               .targets(targets)
               .rule(ruleName)
               .build();

       eventBrClient.putTargets(request);
       System.out.println("Added event rule " + eventRuleName + " with Amazon
SNS target " + topicName + " for bucket "
               + bucketName + ".");
   }

   public static void subEmail(SnsClient snsClient, String topicArn, String
email) {
       try {
           SubscribeRequest request = SubscribeRequest.builder()
                   .protocol("email")
```

```
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
"\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

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

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " +
rule.stateAsString());
            }

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

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
```

```
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
    }

    // Create a new event rule that triggers when an Amazon S3 object is created
in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String
roleArn, String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
```

```
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

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

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String
bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
```

```
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
```

```
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/*
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
 2. Creates an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
 events enabled.
 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 4. Lists rules on the event bus.
 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and lets
 the user subscribe to it.
 6. Adds a target to the rule that sends an email to the specified topic.
 7. Creates an EventBridge event that sends an email when an Amazon S3 object is
 created.
 8. Lists targets.
 9. Lists the rules for the same target.
 10. Triggers the rule by uploading a file to the S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints the state of the rule.
 13. Adds a transform to the rule to change the text of the email.
 14. Enables a specific rule.
 15. Triggers the updated rule by uploading a file to the S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleName> <bucketName> <topicName> <eventRuleName>

    Where:
        roleName - The name of the role to create.
        bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name to
 create.
        topicName - The name of the Amazon Simple Notification Service (Amazon
 SNS) topic to create.
        eventRuleName - The Amazon EventBridge rule name to create.
    """
    val polJSON =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
```

```
   println("1. Create an AWS Identity and Access Management (IAM) role to use
with Amazon EventBridge.")
   val roleArn = createIAMRole(roleName, polJSON)
   println(DASHES)

   println(DASHES)
   println("2. Create an S3 bucket with EventBridge events enabled.")
   if (checkBucket(bucketName)) {
       println("$bucketName already exists. Ending this scenario.")
       exitProcess(1)
   }

   createBucket(bucketName)
   delay(3000)
   setBucketNotification(bucketName)
   println(DASHES)

   println(DASHES)
   println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
   delay(10000)
   addEventRule(roleArn, bucketName, eventRuleName)
   println(DASHES)

   println(DASHES)
   println("4. List rules on the event bus.")
   listRules()
   println(DASHES)

   println(DASHES)
   println("5. Create a new SNS topic for testing and let the user subscribe to
the topic.")
   val topicArn = createSnsTopic(topicName)
   println(DASHES)

   println(DASHES)
   println("6. Add a target to the rule that sends an email to the specified
topic.")
   println("Enter your email to subscribe to the Amazon SNS topic:")
   val email = sc.nextLine()
   subEmail(topicArn, email)
   println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
   sc.nextLine()
```

```
    println(DASHES)

    println(DASHES)
    println("7. Create an EventBridge event that sends an email when an Amazon S3
 object is created.")
    addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName,
 bucketName)
    println(DASHES)

    println(DASHES)
    println("8. List targets.")
    listTargets(eventRuleName)
    println(DASHES)

    println(DASHES)
    println(" 9. List the rules for the same target.")
    listTargetRules(topicArn)
    println(DASHES)

    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
```

```
    println(DASHES)

    println(DASHES)
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
    println("17. Send an event to trigger the rule. This will trigger a
 subscription email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("18. Clean up resources.")
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully
 completed.")
    println(DASHES)
}

suspend fun cleanupResources(
    topicArn: String?,
```

```
    eventRuleName: String?,
    bucketName: String?,
    roleName: String?,
) {
    println("Removing all targets from the event rule.")
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest =
        DetachRolePolicyRequest {
            policyArn = policyArnVal
            roleName = roleNameVal
        }
    IamClient.fromEnvironment { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")

        // Delete the role.
        val roleRequest =
            DeleteRoleRequest {
                roleName = roleNameVal
            }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
```

```
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
```

```
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
    val json =
```

```
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
```

```
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern =
        "{" +
            "\"source\": [\"ExampleSource\"]," +
            "\"detail-type\": [\"ExampleType\"]" +
            "}"
    val request =
        PutRuleRequest {
            name = ruleName
            description = "Custom test rule"
            eventPattern = customEventsPattern
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb =
        InputTransformer {
            inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
```

```
            inputPathsMap = myMap
        }
    val targetOb =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(targetOb)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
```

```
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb =
        PutObjectRequest {
            bucket = bucketName
            key = fileName
            body = myFile.asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
```

```
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3
 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
```

```
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target
 $topicName for bucket $bucketName.")
    }
}

suspend fun subEmail(
    topicArnVal: String?,
    email: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Sid": "EventBridgePublishTopic",
                "Effect": "Allow",
                "Principal": {
```

```
                    "Service": "events.amazonaws.com"
                },
                "Resource": "*",
                "Action": "sns:Publish"
            }
        ]
    }
    """.trimIndent()

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
    }
}

suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
```

```
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """
    {
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
    val eventBridgeConfig =
        EventBridgeConfiguration {
        }

    val configuration =
        NotificationConfiguration {
            eventBridgeConfiguration = eventBridgeConfig
        }

    val configurationRequest =
```

```
        PutBucketNotificationConfigurationRequest {
            bucket = bucketName
            notificationConfiguration = configuration
            skipDestinationValidation = true
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest =
            HeadBucketRequest {
                bucket = bucketName
            }

        S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
```

```
}

suspend fun createIAMRole(
    rolenameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    val rolePolicyRequest =
        AttachRolePolicyRequest {
            roleName = rolenameVal
            policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
        }

    IamClient.fromEnvironment { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Ações para EventBridge usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar EventBridge ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a EventBridge API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para EventBridge usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon EventBridge API Reference.

Exemplos

- Use DeleteRulecom um AWS SDK ou CLI
- Use DescribeRulecom um AWS SDK ou CLI
- Use DisableRulecom um AWS SDK ou CLI
- Use EnableRulecom um AWS SDK ou CLI
- Use ListRuleNamesByTargetcom um AWS SDK ou CLI
- Use ListRulescom um AWS SDK ou CLI
- Use ListTargetsByRulecom um AWS SDK ou CLI
- Use PutEventscom um AWS SDK ou CLI
- Use PutRulecom um AWS SDK ou CLI
- Use PutTargetscom um AWS SDK ou CLI
- Use RemoveTargetscom um AWS SDK ou CLI

Use **`DeleteRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteRule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma regra pelo nome.

```
/// <summary>
/// Delete an event rule by name.
/// </summary>
/// <param name="ruleName">The name of the event rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteRuleByName(string ruleName)
{
    var response = await _amazonEventBridge.DeleteRuleAsync(
        new DeleteRuleRequest()
        {
            Name = ruleName
        });

    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DeleteRulea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para excluir uma regra de CloudWatch eventos

Este exemplo exclui a regra chamada EC2InstanceStateChanges:

```
aws events delete-rule --name "EC2InstanceStateChanges"
```

- Para obter detalhes da API, consulte DeleteRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
       DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
               .name(ruleName)
               .build();

       eventBrClient.deleteRule(ruleRequest);
       System.out.println("Successfully deleted the rule");
   }
```

- Para obter detalhes da API, consulte DeleteRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
```

```
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- Para obter detalhes da API, consulte a DeleteRulereferência da API AWS SDK for Kotlin.

Use **`DescribeRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeRule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obter o estado de uma regra usando a descrição da regra.

```
    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
 uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
```

```
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
 eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }
```

- Para obter detalhes da API, consulte DescribeRulea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para exibir informações sobre uma regra de CloudWatch Eventos

Este exemplo exibe informações sobre a regra chamada DailyLambdaFunction:

```
aws events describe-rule --name "DailyLambdaFunction"
```

- Para obter detalhes da API, consulte DescribeRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void checkRule(EventBridgeClient eventBrClient, String
 eventRuleName) {
```

```
        try {
            DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

            DescribeRuleResponse response =
 eventBrClient.describeRule(ruleRequest);
            System.out.println("The state of the rule is " +
 response.stateAsString());

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

- Para obter detalhes da API, consulte DescribeRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
```

```
}
```

- Para obter detalhes da API, consulte a DescribeRulereferência da API AWS SDK for Kotlin.

Use **`DisableRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DisableRule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Desabilitar uma regra pelo nome da regra.

```
/// <summary>
/// Disable a particular rule on an event bus.
/// </summary
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DisableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
        new DisableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DisableRulea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para desativar uma regra de CloudWatch eventos

Este exemplo desativa a regra chamada DailyLambdaFunction. A regra não é excluída:

```
aws events disable-rule --name "DailyLambdaFunction"
```

- Para obter detalhes da API, consulte DisableRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Desabilitar uma regra usando o nome da regra.

```
    public static void changeRuleState(EventBridgeClient eventBrClient, String
 eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
```

```
                eventBrClient.enableRule(ruleRequest);
            }

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

- Para obter detalhes da API, consulte DisableRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
```

```
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- Para obter detalhes da API, consulte a DisableRulereferência da API AWS SDK for Kotlin.

Use **`EnableRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `EnableRule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Habilitar uma regra pelo nome da regra.

```
    /// <summary>
    /// Enable a particular rule on an event bus.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
            new EnableRuleRequest()
```

```
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte EnableRulea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para habilitar uma regra de CloudWatch Eventos

Este exemplo ativa a regra chamada DailyLambdaFunction, que havia sido desativada anteriormente:

```
aws events enable-rule --name "DailyLambdaFunction"
```

- Para obter detalhes da API, consulte EnableRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Habilitar uma regra usando o nome da regra.

```
   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
```

```
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

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

- Para obter detalhes da API, consulte EnableRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
```

```
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- Para obter detalhes da API, consulte a EnableRulereferência da API AWS SDK for Kotlin.

Use **`ListRuleNamesByTarget`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListRuleNamesByTarget`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todos os nomes das regras usando o destino.

```
    /// <summary>
```

```
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await
_amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- Para obter detalhes da API, consulte ListRuleNamesByTargeta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como exibir todas as regras que têm um destino especificado

Este exemplo exibe todas as regras que têm a função Lambda chamada "MyFunctionName" como destino:

```
aws events list-rule-names-by-target --target-arn "arn:aws:lambda:us-
east-1:123456789012:function:MyFunctionName"
```

- Para obter detalhes da API, consulte ListRuleNamesByTargetem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar todos os nomes das regras usando o destino.

```
   public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
       ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
               .targetArn(topicArn)
               .build();

       ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
       List<String> rules = response.ruleNames();
       for (String rule : rules) {
           System.out.println("The rule name is " + rule);
       }
   }
```

- Para obter detalhes da API, consulte ListRuleNamesByTargeta Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListRuleNamesByTargetreferência da API AWS SDK for Kotlin.

Use **`ListRules`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListRules`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todas as regras para um barramento de eventos.

```
    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty,
 uses the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
 null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- Para obter detalhes da API, consulte ListRulesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para exibir uma lista de todas as regras de CloudWatch eventos

Este exemplo exibe todas as regras de CloudWatch eventos na região:

```
aws events list-rules
```

Para exibir uma lista de regras de CloudWatch eventos começando com uma determinada string.

Este exemplo exibe todas as regras de CloudWatch eventos na região que têm um nome começando com “Diário”:

```
aws events list-rules --name-prefix "Daily"
```

- Para obter detalhes da API, consulte ListRulesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Habilitar uma regra usando o nome da regra.

```
    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
 rule.description());
                System.out.println("The rule state is : " +
 rule.stateAsString());
            }

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

- Para obter detalhes da API, consulte ListRulesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListRulesreferência da API AWS SDK for Kotlin.

Use **`ListTargetsByRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTargetsByRule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar todos os destinos para uma regra usando o nome da regra.

```
/// <summary>
/// List all of the targets matching a rule by name.
/// </summary>
/// <param name="ruleName">The name of the rule.</param>
/// <returns>The list of targets.</returns>
public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
{
    var results = new List<Target>();
    var request = new ListTargetsByRuleRequest()
    {
        Rule = ruleName
    };
    ListTargetsByRuleResponse response;
    do
    {
        response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
        results.AddRange(response.Targets);
        request.NextToken = response.NextToken;

    } while (response.NextToken is not null);

    return results;
}
```

- Para obter detalhes da API, consulte ListTargetsByRulea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para exibir todos os alvos de uma regra de CloudWatch Eventos

Este exemplo exibe todos os alvos da regra chamada DailyLambdaFunction:

```
aws events list-targets-by-rule --rule  "DailyLambdaFunction"
```

- Para obter detalhes da API, consulte ListTargetsByRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todos os destinos de uma regra usando o nome da regra.

```
   public static void listTargets(EventBridgeClient eventBrClient, String
ruleName) {
       ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
               .rule(ruleName)
               .build();

       ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
       List<Target> targetsList = res.targets();
       for (Target target: targetsList) {
           System.out.println("Target ARN: "+target.arn());
       }
   }
```

- Para obter detalhes da API, consulte ListTargetsByRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListTargetsByRulereferência da API AWS SDK for Kotlin.

Use **`PutEvents`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutEvents`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos

- Criar e acionar uma regra

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie um evento que corresponda a um padrão personalizado para uma regra.

```
    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });
```

```
        return response.FailedEntryCount == 0;
    }
```

- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Enviar o evento.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
        event_entry.SetDetail(MakeDetails(event_key, event_value));
        event_entry.SetDetailType("sampleSubmitted");
        event_entry.AddResources(resource_arn);
        event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

        Aws::CloudWatchEvents::Model::PutEventsRequest request;
        request.AddEntries(event_entry);

        auto outcome = cwe.PutEvents(request);
        if (!outcome.IsSuccess())
```

```
        {
            std::cout << "Failed to post CloudWatch event: " <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully posted CloudWatch event" << std::endl;
        }
```

- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para enviar um evento personalizado para CloudWatch Eventos

Este exemplo envia um evento personalizado para CloudWatch Eventos. O evento está contido no arquivo putevents.json:

```
aws events put-events --entries file://putevents.json
```

Veja a seguir o conteúdo do arquivo putevent.json:

```
[
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value1\", \"key2\": \"value2\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType"
  },
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value3\", \"key2\": \"value4\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
```

```
    "DetailType": "myDetailType"
   }
]
```

- Para obter detalhes da API, consulte PutEventsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
       String json = "{" +
               "\"UserEmail\": \"" + email + "\"," +
               "\"Message\": \"This event was generated by example code.\"," +
               "\"UtcTime\": \"Now.\"" +
               "}";

       PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
               .source("ExampleSource")
               .detail(json)
               .detailType("ExampleType")
               .build();

       PutEventsRequest eventsRequest = PutEventsRequest.builder()
               .entries(entry)
               .build();

       eventBrClient.putEvents(eventsRequest);
   }
```

- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import {
  EventBridgeClient,
  PutEventsCommand,
} from "@aws-sdk/client-eventbridge";

export const putEvents = async (
  source = "eventbridge.integration.test",
  detailType = "greeting",
  resources = [],
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutEventsCommand({
      Entries: [
        {
          Detail: JSON.stringify({ greeting: "Hello there." }),
          DetailType: detailType,
          Resources: resources,
          Source: source,
        },
      ],
    }),
  );

  console.log("PutEvents response:");
  console.log(response);
  // PutEvents response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
```

```
  //     requestId: '3d0df73d-dcea-4a23-ae0d-f5556a3ac109',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Entries: [ { EventId: '51620841-5af4-6402-d9bc-b77734991eb5' } ],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};
```

```
ebevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
```

```
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- Para obter detalhes da API, consulte a PutEventsreferência da API AWS SDK for Kotlin.

Use **`PutRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutRule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar e acionar uma regra
- Envie notificações de eventos para EventBridge

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma regra que seja acionada quando um objeto é adicionado a um bucket do Amazon Simple Storage Service.

```
    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created
 in a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
```

```
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]"
 +
                                        "}" +
                                    "}" +
                                "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }
```

Crie uma regra que utilize um padrão personalizado.

```
    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
```

```
                                    "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }
```

- Para obter detalhes da API, consulte PutRulea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Crie a regra.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;
        Aws::CloudWatchEvents::Model::PutRuleRequest request;
        request.SetName(rule_name);
```

```
        request.SetRoleArn(role_arn);
        request.SetScheduleExpression("rate(5 minutes)");
        request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

        auto outcome = cwe.PutRule(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events rule " <<
                rule_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch events rule " <<
                rule_name << " with resulting Arn " <<
                outcome.GetResult().GetRuleArn() << std::endl;
        }
```

- Para obter detalhes da API, consulte PutRulea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar regras de CloudWatch eventos

Este exemplo cria uma regra que é acionada todos os dias, às 9h UTC. Se você usar put-targets para adicionar uma função do Lambda como destino dessa regra, poderá executar a função do Lambda todos os dias no horário especificado:

```
aws events put-rule --name "DailyLambdaFunction" --schedule-expression "cron(0 9
 * * ? *)"
```

Este exemplo cria uma regra que é acionada quando qualquer instância do EC2 na região muda de estado:

```
aws events put-rule --name "EC2InstanceStateChanges" --event-pattern "{\"source
\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change Notification\"]}"
 --role-arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

Este exemplo cria uma regra que é acionada quando qualquer instância do EC2 na região é interrompida ou encerrada:

```
aws events put-rule --name "EC2InstanceStateChangeStopOrTerminate" --event-
pattern "{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change
 Notification\"],\"detail\":{\"state\":[\"stopped\",\"terminated\"]}}" --role-
arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

- Para obter detalhes da API, consulte PutRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma regra agendada

```
   public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
       try {
           PutRuleRequest ruleRequest = PutRuleRequest.builder()
                   .name(ruleName)
                   .eventBusName("default")
                   .scheduleExpression(cronExpression)
                   .state("ENABLED")
                   .description("A test rule that runs on a schedule created by
the Java API")
                   .build();

           PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
           System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

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

```
        }
    }
```

Crie uma regra que seja acionada quando um objeto é adicionado a um bucket do Amazon Simple Storage Service.

```
    // Create a new event rule that triggers when an Amazon S3 object is created
in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String
roleArn, String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

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

- Para obter detalhes da API, consulte PutRulea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { EventBridgeClient, PutRuleCommand } from "@aws-sdk/client-eventbridge";

export const putRule = async (
  ruleName = "some-rule",
  source = "some-source",
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutRuleCommand({
      Name: ruleName,
      EventPattern: JSON.stringify({ source: [source] }),
      State: "ENABLED",
      EventBusName: "default",
    }),
  );

  console.log("PutRule response:");
  console.log(response);
  // PutRule response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd7292ced-1544-421b-842f-596326bc7072',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   RuleArn: 'arn:aws:events:us-east-1:xxxxxxxxxxxx:rule/
EventBridgeTestRule-1696280037720'
```

```
  // }
  return response;
};
```

- Para obter detalhes da API, consulte PutRulea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

ebevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- Para obter detalhes da API, consulte PutRulea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma regra agendada

```
suspend fun createScRule(
    ruleName: String?,
    cronExpression: String?,
) {
    val ruleRequest =
        PutRuleRequest {
            name = ruleName
            eventBusName = "default"
            scheduleExpression = cronExpression
            state = RuleState.Enabled
            description = "A test rule that runs on a schedule created by the
 Kotlin API"
        }

    EventBridgeClient.fromEnvironment { region = "us-west-2" }.use
 { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

Crie uma regra que seja acionada quando um objeto é adicionado a um bucket do Amazon Simple Storage Service.

```
// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
```

```
    eventRuleName: String?,
) {
    val pattern = """
    {
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- Para obter detalhes da API, consulte a PutRulereferência da API AWS SDK for Kotlin.

Use **`PutTargets`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutTargets`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Envie notificações de eventos para EventBridge

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione um tópico do Amazon SNS como um destino para uma regra.

```
    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string
targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };

        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
```

```
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }
```

Adicione um transformador de entrada a um destino para uma regra.

```
    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
```

```
                    InputTemplate = "\"Notification: an object was uploaded to
 bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
 {e.ErrorCode}");
            });
        }
        return targetID;
    }
```

- Para obter detalhes da API, consulte PutTargetsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
```

```
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Adicione o destino.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::Target target;
        target.SetArn(lambda_arn);
        target.SetId(target_id);

        Aws::CloudWatchEvents::Model::PutTargetsRequest request;
        request.SetRule(rule_name);
        request.AddTargets(target);

        auto putTargetsOutcome = cwe.PutTargets(request);
        if (!putTargetsOutcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout <<
                "Successfully created CloudWatch events target for rule "
                << rule_name << std::endl;
        }
```

- Para obter detalhes da API, consulte PutTargetsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para adicionar alvos às regras de CloudWatch eventos

Este exemplo adiciona uma função do Lambda como o destino de uma regra:

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:MyFunctionName"
```

Este exemplo define um fluxo do Amazon Kinesis como o destino. Desta forma, os eventos capturados pela regra são retransmitidos para o fluxo:

```
aws events put-targets --rule EC2InstanceStateChanges --targets
 "Id"="1","Arn"="arn:aws:kinesis:us-east-1:123456789012:stream/
MyStream","RoleArn"="arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

Este exemplo define dois fluxos do Amazon Kinesis como destinos de uma regra:

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="Target1","Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/
MyStream1","RoleArn"="arn:aws:iam::379642911888:role/ MyRoleToAccessLambda"
  "Id"="Target2"," Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/
MyStream2","RoleArn"="arn:aws:iam::379642911888:role/MyRoleToAccessLambda"
```

- Para obter detalhes da API, consulte PutTargetsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione um tópico do Amazon SNS como um destino para uma regra.

```
    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
 ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
```

```
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon
SNS target " + topicName + " for bucket "
                + bucketName + ".");
    }
```

Adicione um transformador de entrada a um destino para uma regra.

```
    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
```

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

- Para obter detalhes da API, consulte PutTargetsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import {
  EventBridgeClient,
  PutTargetsCommand,
} from "@aws-sdk/client-eventbridge";

export const putTarget = async (
  existingRuleName = "some-rule",
  targetArn = "arn:aws:lambda:us-east-1:000000000000:function:test-func",
  uniqueId = Date.now().toString(),
) => {
  const client = new EventBridgeClient({});
  const response = await client.send(
    new PutTargetsCommand({
      Rule: existingRuleName,
      Targets: [
        {
          Arn: targetArn,
          Id: uniqueId,
        },
      ],
    }),
```

```
  );

  console.log("PutTargets response:");
  console.log(response);
  // PutTargets response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'f5b23b9a-2c17-45c1-ad5c-f926c3692e3d',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   FailedEntries: [],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- Para obter detalhes da API, consulte PutTargetsa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
```

```
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myEventBridgeTarget",
    },
  ],
};

ebevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter detalhes da API, consulte PutTargetsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3
 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
```

```
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target
 $topicName for bucket $bucketName.")
    }
}
```

Adicione um transformador de entrada a um destino para uma regra.

```
suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }
```

```
    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- Para obter detalhes da API, consulte a PutTargetsreferência da API AWS SDK for Kotlin.

Use **`RemoveTargets`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RemoveTargets`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Remover todos os destinos de uma regra usando o nome da regra.

```
    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
```

```
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);

        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage},
code {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte RemoveTargetsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como remover um destino de um evento

Este exemplo remove o stream do Amazon Kinesis chamado MyStream 1 de ser um alvo da regra. DailyLambdaFunction Quando DailyLambdaFunction foi criado, esse fluxo foi definido como um destino com um ID de Target1:

```
aws events remove-targets --rule "DailyLambdaFunction" --ids "Target1"
```

- Para obter detalhes da API, consulte RemoveTargetsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Remover todos os destinos de uma regra usando o nome da regra.

```
   public static void deleteTargetsFromRule(EventBridgeClient eventBrClient,
String eventRuleName) {
       // First, get all targets that will be deleted.
       ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
               .rule(eventRuleName)
               .build();

       ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
       List<Target> allTargets = response.targets();

       // Get all targets and delete them.
       for (Target myTarget : allTargets) {
           RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                   .rule(eventRuleName)
                   .ids(myTarget.id())
                   .build();
```

```
            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }
```

- Para obter detalhes da API, consulte RemoveTargetsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
```

```
                }
            }
        }
    }
```

- Para obter detalhes da API, consulte a RemoveTargetsreferência da API AWS SDK for Kotlin.

Cenários para EventBridge usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns EventBridge com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas EventBridge ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie e acione uma regra na Amazon EventBridge usando um AWS SDK
- Envie notificações de eventos do S3 para a Amazon EventBridge usando um AWS SDK
- Usar eventos programados para chamar uma função do Lambda

Crie e acione uma regra na Amazon EventBridge usando um AWS SDK

O exemplo de código a seguir mostra como criar e acionar uma regra na Amazon EventBridge.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Chame as funções na ordem correta.

```
require 'aws-sdk-sns'
require 'aws-sdk-iam'
require 'aws-sdk-cloudwatchevents'
require 'aws-sdk-ec2'
require 'aws-sdk-cloudwatch'
require 'aws-sdk-cloudwatchlogs'
require 'securerandom'
```

Verifica se o tópico do Amazon Simple Notification Service (Amazon SNS) existe dentre aqueles fornecidos para essa função.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  false
end
```

Verifica se o tópico especificado existe dentre aqueles disponíveis para o chamador no Amazon SNS.

```
# Checks whether the specified topic exists among those available to the
```

```
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts 'Topic found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.topics.count.positive?

      if topic_found?(response.topics, topic_arn)
        puts 'Topic found.'
        return true
      end
    end
  end
  puts 'Topic not found.'
  false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  false
end
```

Crie um tópico no Amazon SNS e, em seguida, assine um endereço de e-mail para receber notificações sobre esse tópico.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
```

```
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: 'email',
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts 'Subscription created with ARN ' \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    'and confirm the subscription to start receiving notification emails.'
  topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  'Error'
end
```

Verifique se a função especificada AWS Identity and Access Management (IAM) existe entre as fornecidas para essa função.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
```

```
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  false
end
```

Verificar se o perfil especificado existe dentre aqueles disponíveis para o chamador no IAM.

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts 'Role found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.roles.count.positive?

      if role_found?(response.roles, role_arn)
        puts 'Role found.'
        return true
      end
```

```
    end
  end
  puts 'Role not found.'
  false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  false
end
```

Criar um perfil do IAM.

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'
          },
          'Action': 'sts:AssumeRole'
        }
      ]
    }.to_json,
    path: '/',
    role_name: role_name
```

```
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts 'Adding access policy to role...'
  iam_client.put_role_policy(
    policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': 'CloudWatchEventsFullAccess',
          'Effect': 'Allow',
          'Resource': '*',
          'Action': 'events:*'
        },
        {
          'Sid': 'IAMPassRoleForCloudWatchEvents',
          'Effect': 'Allow',
          'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
          'Action': 'iam:PassRole'
        }
      ]
    }.to_json,
    policy_name: 'CloudWatchEventsPolicy',
    role_name: role_name
  )
  puts 'Access policy added to role.'
  response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts 'If the role was created, you must add the access policy ' \
    'to the role yourself, or delete the role yourself and try again.'
  'Error'
end
```

Verifica se a EventBridge regra especificada existe entre as fornecidas para essa função.

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
```

```
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  false
end
```

Verifica se a regra especificada existe entre as disponíveis para o chamador. EventBridge

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts 'Rule found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.rules.count.positive?

      if rule_found?(response.rules, rule_name)
        puts 'Rule found.'
        return true
```

```
      end
    end
  end
  puts 'Rule not found.'
  false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  false
end
```

Crie uma regra em EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
```

```
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        'aws.ec2'
      ],
      'detail-type': [
        'EC2 Instance State-change Notification'
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: 'ENABLED',
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
     put_targets_response.failed_entry_count.positive?
```

```
    puts 'Error(s) adding target to rule:'
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    false
  else
    true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts 'If the rule was created, you must add the target ' \
    'to the rule yourself, or delete the rule yourself and try again.'
  false
end
```

Verifique se o grupo de registros especificado existe entre aqueles disponíveis para o chamador no Amazon CloudWatch Logs.

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts 'Log group found.'
        return true
      end
    end
```

```
  end
  puts 'Log group not found.'
  false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  false
end
```

Crie um grupo de CloudWatch registros em Registros.

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts 'Log group created.'
  true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  false
end
```

Grave um evento em um stream de CloudWatch registros em Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
```

```
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '49542672486831074009579604567656788214806863282469607 3EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  event[:sequence_token] = sequence_token unless sequence_token.empty?

  response = cloudwatchlogs_client.put_log_events(event)
  puts 'Message logged.'
  response.next_sequence_token
rescue StandardError => e
```

```
  puts "Message not logged: #{e.message}"
end
```

Reinicie uma instância do Amazon Elastic Compute Cloud (Amazon EC2) e adicione informações sobre a atividade relacionada a um stream de log no Logs. CloudWatch

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
```

```
  sequence_token = ''

  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    'This might take a few minutes...'
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts 'Attempting to restart the instance. This might take a few minutes...'
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance restarted.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  true
rescue StandardError => e
  puts 'Error creating log stream or stopping or restarting the instance: ' \
    "#{e.message}"
  log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  false
end
```

Exibir informações sobre a atividade de uma regra em EventBridge.

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts 'Attempting to display rule activity...'
  response = cloudwatch_client.get_metric_statistics(
    namespace: 'AWS/Events',
    metric_name: 'Invocations',
    dimensions: [
      {
        name: 'RuleName',
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ['Sum'],
```

```
    unit: 'Count'
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      'specified time period.'
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Exibir informações de registro de todos os fluxos de registros em um grupo de CloudWatch registros de registros.

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts 'Attempting to display log stream data for the log group ' \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: 'LastEventTime',
    descending: true
  )
```

```
  if describe_log_streams_response.key?(:log_streams) &&
     describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts '-' * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
         get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts 'No log messages for this log stream.'
      end
    end
  end
rescue StandardError => e
  puts 'Error getting information about the log streams or their messages: ' \
    "#{e.message}"
end
```

Exiba um lembrete para o chamador limpar manualmente todos AWS os recursos associados dos quais ele não precisa mais.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log
 group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
```

```
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts '-' * 10
  puts 'Some of the following AWS resources might still exist in your account.'
  puts 'If you no longer want to use this code example, then to clean up'
  puts 'your AWS account and avoid unexpected costs, you might want to'
  puts 'manually delete any of the following resources if they exist:'
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - PutEvents
  - PutRule

Envie notificações de eventos do S3 para a Amazon EventBridge usando um AWS SDK

O exemplo de código a seguir mostra como habilitar um bucket para enviar notificações de eventos do S3 EventBridge e rotear notificações para um tópico do Amazon SNS e uma fila do Amazon SQS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
```

```
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Usar eventos programados para chamar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

Java

SDK para Java 2.x

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função Lambda usando a API de tempo de execução do JavaScript Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento programado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função Lambda o alvo.
- Concede permissão para permitir a EventBridge invocação da função Lambda.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações programadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda

- Amazon SNS

Exemplos de código para o EventBridge Scheduler usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon EventBridge Scheduler com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- EventBridge Guia do usuário do Scheduler — Mais informações sobre o EventBridge Scheduler.
- EventBridge Referência da API do Scheduler — Detalhes sobre todas as EventBridge ações disponíveis do Scheduler.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do EventBridge Scheduler AWS SDKs
  - Olá, EventBridge Agendador
  - Ações para o EventBridge Scheduler usando AWS SDKs
    - Use CreateSchedulecom um AWS SDK
    - Use CreateScheduleGroupcom um AWS SDK
    - Use DeleteSchedulecom um AWS SDK
    - Use DeleteScheduleGroupcom um AWS SDK
- Cenários para o uso do EventBridge Scheduler AWS SDKs
  - Um cenário completo de eventos EventBridge agendados do Scheduler usando um AWS SDK

Exemplos básicos para o uso do EventBridge Scheduler AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon EventBridge Scheduler com AWS SDKs.

Exemplos

- Olá, EventBridge Agendador
- Ações para o EventBridge Scheduler usando AWS SDKs
  - Use CreateSchedulecom um AWS SDK
  - Use CreateScheduleGroupcom um AWS SDK
  - Use DeleteSchedulecom um AWS SDK
  - Use DeleteScheduleGroupcom um AWS SDK

Olá, EventBridge Agendador

Os exemplos de código a seguir mostram como começar a usar o EventBridge Scheduler.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static class HelloScheduler
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the EventBridge Scheduler service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonScheduler>()
            ).Build();
```

```
        // Now the client is available for injection.
        var schedulerClient =
 host.Services.GetRequiredService<IAmazonScheduler>();

        // You can use await and any of the async methods to get a response, or a
 paginator to list schedules or groups.
        var results = new List<ScheduleSummary>();
        var paginateSchedules = schedulerClient.Paginators.ListSchedules(
            new ListSchedulesRequest());
        Console.WriteLine(
            $"Hello AWS Scheduler! Let's list schedules in your account.");
        // Get the entire list using the paginator.
        await foreach (var schedule in paginateSchedules.Schedules)
        {
            results.Add(schedule);
        }
        Console.WriteLine($"\tTotal of {results.Count} schedule(s) available.");
        results.ForEach(s => Console.WriteLine($"\tSchedule: {s.Name}"));
    }
}
```

- Para obter detalhes da API, consulte ListSchedulesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ListSchedulesRequest;
import software.amazon.awssdk.services.scheduler.model.ScheduleSummary;
import
 software.amazon.awssdk.services.scheduler.paginators.ListSchedulesPublisher;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloScheduler {

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

    /**
     * Lists all the schedules available.
     * <p>
     * This method uses the {@link SchedulerAsyncClient} to make an asynchronous
 request to
     * list all the schedules available. The method uses the {@link
 ListSchedulesPublisher}
     * to fetch the schedules in a paginated manner, and then processes the
 responses
     * asynchronously.
     */
    public static void listSchedulesAsync() {
        SchedulerAsyncClient schedulerAsyncClient =
 SchedulerAsyncClient.create();

        // Build the request to list schedules
        ListSchedulesRequest listSchedulesRequest =
 ListSchedulesRequest.builder().build();

        // Use the paginator to fetch all schedules asynchronously.
        ListSchedulesPublisher paginator =
 schedulerAsyncClient.listSchedulesPaginator(listSchedulesRequest);
        List<ScheduleSummary> results = new ArrayList<>();

        // Subscribe to the paginator to process the response asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schedules().forEach(schedule -> {
                results.add(schedule);
                System.out.printf("Schedule: %s%n", schedule.name());
            });
        });

        // Wait for the asynchronous operation to complete.
        future.join();
```

```
        // After all schedules are fetched, print the total count.
        System.out.printf("Total of %d schedule(s) available.%n",
 results.size());
    }
}
```

- Para obter detalhes da API, consulte ListSchedulesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_scheduler(scheduler_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EventBridge Scheduler
    client and list the schedules in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param scheduler_client: A Boto3 Amazon EventBridge Scheduler Client object.
 This object wraps
                             the low-level Amazon EventBridge Scheduler service
 API.
    """
    print("Hello, Amazon EventBridge Scheduler! Let's list some of your
 schedules:\n")
    paginator = scheduler_client.get_paginator("list_schedules")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})
```

```
    schedule_names: [str] = []
    for page in page_iterator:
        for schedule in page["Schedules"]:
            schedule_names.append(schedule["Name"])

    print(f"{len(schedule_names)} schedule(s) retrieved.")
    for schedule_name in schedule_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    hello_scheduler(boto3.client("scheduler"))
```

- Para obter detalhes da API, consulte a ListSchedulesReferência da API AWS SDK for Python (Boto3).

Ações para o EventBridge Scheduler usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do EventBridge Scheduler com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API EventBridge Scheduler e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para o uso do EventBridge Scheduler AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon EventBridge Scheduler API Reference.

Exemplos

- Use CreateSchedulecom um AWS SDK
- Use CreateScheduleGroupcom um AWS SDK
- Use DeleteSchedulecom um AWS SDK
- Use DeleteScheduleGroupcom um AWS SDK

Use **`CreateSchedule`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateSchedule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Eventos agendados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a new schedule in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule.</param>
    /// <param name="scheduleExpression">The schedule expression that defines
when the schedule should run.</param>
    /// <param name="scheduleGroupName">The name of the schedule group to which
the schedule should be added.</param>
    /// <param name="deleteAfterCompletion">Indicates whether to delete the
schedule after completion.</param>
    /// <param name="useFlexibleTimeWindow">Indicates whether to use a flexible
time window for the schedule.</param>
    /// <param name="targetArn">ARN of the event target.</param>
    /// <param name="roleArn">Execution Role ARN.</param>
    /// <returns>True if the schedule was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleAsync(
            string name,
            string scheduleExpression,
            string scheduleGroupName,
            string targetArn,
            string roleArn,
            string input,
            bool deleteAfterCompletion = false,
            bool useFlexibleTimeWindow = false)
    {
```

```
        try
        {
            int hoursToRun = 1;
            int flexibleTimeWindowMinutes = 10;

            var request = new CreateScheduleRequest
            {
                Name = name,
                ScheduleExpression = scheduleExpression,
                GroupName = scheduleGroupName,
                Target = new Target { Arn = targetArn, RoleArn = roleArn, Input =
input },
                ActionAfterCompletion = deleteAfterCompletion
                    ? ActionAfterCompletion.DELETE
                    : ActionAfterCompletion.NONE,
                StartDate = DateTime.UtcNow, // Ignored for one-time schedules.
                EndDate =
                    DateTime.UtcNow
                        .AddHours(hoursToRun) // Ignored for one-time schedules.
            };
            // Allow a flexible time window if the caller specifies it.
            request.FlexibleTimeWindow = new FlexibleTimeWindow
            {
                Mode = useFlexibleTimeWindow
                    ? FlexibleTimeWindowMode.FLEXIBLE
                    : FlexibleTimeWindowMode.OFF,
                MaximumWindowInMinutes = useFlexibleTimeWindow
                    ? flexibleTimeWindowMinutes
                    : null
            };

            var response = await _amazonScheduler.CreateScheduleAsync(request);

            Console.WriteLine($"Successfully created schedule '{name}' " +
                              $"in schedule group '{scheduleGroupName}':
{response.ScheduleArn}.");
            return true;
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule '{name}' due to a
conflict. {ex.Message}");
            return false;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating schedule '{name}'
" +
                              $"in schedule group '{scheduleGroupName}':
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateSchedulea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new schedule for a target task.
     *
     * @param name                  the name of the schedule
     * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
     * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
     * @param targetArn             the Amazon Resource Name (ARN) of the target
task
     * @param roleArn               the ARN of the IAM role to be used for the
schedule
     * @param input                 the input data for the target task
     * @param deleteAfterCompletion whether to delete the schedule after it's
executed
```

```
     * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
     * @return true if the schedule was successfully created, false otherwise
     */
    public CompletableFuture<Boolean> createScheduleAsync(
        String name,
        String scheduleExpression,
        String scheduleGroupName,
        String targetArn,
        String roleArn,
        String input,
        boolean deleteAfterCompletion,
        boolean useFlexibleTimeWindow) {

        int hoursToRun = 1;
        int flexibleTimeWindowMinutes = 10;

        Target target = Target.builder()
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
```

```
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group
{}, The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while
creating the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception
occurred while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: "
+ ex.getMessage(), ex);
                    }
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateScheduleea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SchedulerWrapper:
```

```
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
client.

        :return: An instance of SchedulerWrapper initialized with the default
EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule(
        self,
        name: str,
        schedule_expression: str,
        schedule_group_name: str,
        target_arn: str,
        role_arn: str,
        input: str,
        delete_after_completion: bool = False,
        use_flexible_time_window: bool = False,
    ) -> str:
        """
        Creates a new schedule with the specified parameters.

        :param name: The name of the schedule.
        :param schedule_expression: The expression that defines when the schedule
runs.
        :param schedule_group_name: The name of the schedule group.
        :param target_arn: The Amazon Resource Name (ARN) of the target.
        :param role_arn: The Amazon Resource Name (ARN) of the execution IAM
role.
        :param input: The input for the target.
        :param delete_after_completion: Whether to delete the schedule after it
completes.
        :param use_flexible_time_window: Whether to use a flexible time window.

        :return The ARN of the created schedule.
        """
```

```
        try:
            hours_to_run = 1
            flexible_time_window_minutes = 10
            parameters = {
                "Name": name,
                "ScheduleExpression": schedule_expression,
                "GroupName": schedule_group_name,
                "Target": {"Arn": target_arn, "RoleArn": role_arn, "Input":
 input},
                "StartDate": datetime.now(timezone.utc),
                "EndDate": datetime.now(timezone.utc) +
 timedelta(hours=hours_to_run),
            }

            if delete_after_completion:
                parameters["ActionAfterCompletion"] = "DELETE"

            if use_flexible_time_window:
                parameters["FlexibleTimeWindow"] = {
                    "Mode": "FLEXIBLE",
                    "MaximumWindowInMinutes": flexible_time_window_minutes,
                }
            else:
                parameters["FlexibleTimeWindow"] = {"Mode": "OFF"}

            response = self.scheduler_client.create_schedule(**parameters)
            return response["ScheduleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule: %s", err.response["Error"]
["Message"]
                )
            raise
```

- Para obter detalhes da API, consulte a CreateScheduleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Constants for time calculations
    DATA lv_start_date TYPE /aws1/scdstartdate.
    DATA lv_end_date TYPE /aws1/scdenddate.
    DATA lv_start_timestamp TYPE timestamp.
    DATA lv_end_timestamp TYPE timestamp.
    DATA lv_hours_to_run TYPE i VALUE 1.

    " Get current timestamp
    GET TIME STAMP FIELD lv_start_timestamp.

    " Add 1 hour to the current timestamp using CL_ABAP_TSTMP
    lv_end_timestamp = cl_abap_tstmp=>add(
      tstmp = lv_start_timestamp
      secs = lv_hours_to_run * 3600 ).

    " Convert timestamps to decimal format for AWS API
    lv_start_date = lv_start_timestamp.
    lv_end_date = lv_end_timestamp.

    " Prepare flexible time window configuration
    DATA lo_flexible_time_window TYPE REF TO /aws1/cl_scdflexibletimewindow.
    IF iv_use_flexible_time_win = abap_true.
      " iv_use_flexible_time_win = ABAP_TRUE
      " Example: Set MaximumWindowInMinutes to 10 for flexible window
      lo_flexible_time_window = NEW /aws1/cl_scdflexibletimewindow(
        iv_mode = 'FLEXIBLE'
        iv_maximumwindowinminutes = 10 ).
```

```
        ELSE.
          lo_flexible_time_window = NEW /aws1/cl_scdflexibletimewindow(
            iv_mode = 'OFF' ).
        ENDIF.

        " Prepare target configuration
        " Example iv_target_arn = 'arn:aws:sqs:us-east-1:123456789012:my-queue'
        " Example iv_role_arn = 'arn:aws:iam::123456789012:role/SchedulerRole'
        " Example iv_input = '{"message": "Hello from EventBridge Scheduler"}'
        DATA(lo_target) = NEW /aws1/cl_scdtarget(
          iv_arn = iv_target_arn
          iv_rolearn = iv_role_arn
          iv_input = iv_input ).

        " Set action after completion if needed
        DATA lv_action_after_completion TYPE /aws1/scdactionaftercompletion.
        IF iv_delete_after_completion = abap_true.
          " iv_delete_after_completion = ABAP_TRUE
          lv_action_after_completion = 'DELETE'.
        ELSE.
          lv_action_after_completion = 'NONE'.
        ENDIF.

        " Create the schedule
        " Example iv_name = 'my-schedule'
        " Example iv_schedule_expression = 'rate(15 minutes)'
        " Example iv_schedule_group_name = 'my-schedule-group'
        DATA(lo_result) = lo_scd->createschedule(
          iv_name = iv_name
          iv_scheduleexpression = iv_schedule_expression
          iv_groupname = iv_schedule_group_name
          io_target = lo_target
          io_flexibletimewindow = lo_flexible_time_window
          iv_startdate = lv_start_date
          iv_enddate = lv_end_date
          iv_actionaftercompletion = lv_action_after_completion ).

        ov_schedule_arn = lo_result->get_schedulearn( ).
        MESSAGE 'Schedule created successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdconflictexception INTO DATA(lo_conflict_ex).
        DATA(lv_error) = |Conflict creating schedule: { lo_conflict_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error creating schedule: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateScheduleReferência da API AWS SDK for SAP ABAP.

Use **`CreateScheduleGroup`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateScheduleGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Eventos agendados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a new schedule group in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group.</param>
    /// <returns>True if the schedule group was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleGroupAsync(string name)
    {
        try
        {
            var request = new CreateScheduleGroupRequest { Name = name };
```

```
            var response = await
_amazonScheduler.CreateScheduleGroupAsync(request);

            Console.WriteLine($"Successfully created schedule group '{name}':
{response.ScheduleGroupArn}.");
            return true;

        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule group '{name}' due to a
conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating schedule group '{name}':
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateScheduleGroupa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new schedule group.
```

```
     *
     * @param name the name of the schedule group to be created
     * @return a {@link CompletableFuture} representing the asynchronous
operation of creating the schedule group
     */
    public CompletableFuture<CreateScheduleGroupResponse>
createScheduleGroup(String name) {
        CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
            .name(name)
            .build();

        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
                } else {
                    throw new CompletionException("Failed to create schedule
group: " + name, ex);
                }
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }
```

- Para obter detalhes da API, consulte CreateScheduleGroupa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule_group(self, name: str) -> str:
        """
        Creates a new schedule group with the specified name and description.

        :param name: The name of the schedule group.
        :param description: The description of the schedule group.

        :return: The ARN of the created schedule group.
        """
        try:
            response = self.scheduler_client.create_schedule_group(Name=name)
            return response["ScheduleGroupArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
```

```
                    "Failed to create schedule group '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule group: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a CreateScheduleGroupReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example iv_name = 'my-schedule-group'
        DATA(lo_result) = lo_scd->createschedulegroup(
          iv_name = iv_name ).

        ov_schedule_group_arn = lo_result->get_schedulegrouparn( ).
        MESSAGE 'Schedule group created successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdconflictexception INTO DATA(lo_conflict_ex).
        DATA(lv_error) = |Conflict creating schedule group: { lo_conflict_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error creating schedule group: { lo_generic_ex-
>if_message~get_text( ) }|.
```

```
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateScheduleGroupreferência da API AWS SDK for SAP ABAP.

Use **`DeleteSchedule`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteSchedule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Eventos agendados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an existing schedule from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule to delete.</param>
    /// <param name="groupName">The group name of the schedule to delete.</param>
    /// <returns>True if the schedule was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleAsync(string name, string groupName)
    {
        try
        {
            var request = new DeleteScheduleRequest
            {
                Name = name,
                GroupName = groupName
```

```
            };

            await _amazonScheduler.DeleteScheduleAsync(request);

            Console.WriteLine($"Successfully deleted schedule with name
'{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule with ID '{name}' because the resource
was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule with ID '{name}':
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteSchedulea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
```

```
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the
schedule, except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while
deleting schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.",
ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
```

- Para obter detalhes da API, consulte DeleteSchedulea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def delete_schedule(self, name: str, schedule_group_name: str) -> None:
        """
        Deletes the schedule with the specified name and schedule group.

        :param name: The name of the schedule.
        :param schedule_group_name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule(
                Name=name, GroupName=schedule_group_name
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
```

```
                    "Failed to delete schedule with ID '%s' because the resource
 was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule: %s", err.response["Error"]
["Message"]
                )
                raise
```

- Para obter detalhes da API, consulte a DeleteScheduleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example iv_name = 'my-schedule'
        " Example iv_schedule_group_name = 'my-schedule-group'
        lo_scd->deleteschedule(
          iv_name = iv_name
          iv_groupname = iv_schedule_group_name ).
        MESSAGE 'Schedule deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdresourcenotfoundex INTO DATA(lo_not_found_ex).
        DATA(lv_error) = |Schedule not found: { lo_not_found_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
```

```
        DATA(lv_generic_error) = |Error deleting schedule: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteSchedulereferência da API AWS SDK for SAP ABAP.

Use **`DeleteScheduleGroup`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteScheduleGroup`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an existing schedule group from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group to delete.</param>
    /// <returns>True if the schedule group was deleted successfully, false
 otherwise.</returns>
    public async Task<bool> DeleteScheduleGroupAsync(string name)
    {
        try
        {
            var request = new DeleteScheduleGroupRequest { Name = name };

            await _amazonScheduler.DeleteScheduleGroupAsync(request);

            Console.WriteLine($"Successfully deleted schedule group '{name}'.");
            return true;

        }
```

```
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule group '{name}' because the resource
was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule group '{name}':
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteScheduleGroupa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
```

```
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not
 found: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
 group: " + ex.getMessage(), ex);
                    }
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteScheduleGroupa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
```

```
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
client.

        :return: An instance of SchedulerWrapper initialized with the default
EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def delete_schedule_group(self, name: str) -> None:
        """
        Deletes the schedule group with the specified name.

        :param name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule_group(Name=name)
            logger.info("Schedule group %s deleted successfully.", name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule group with ID '%s' because the
resource was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule group: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a DeleteScheduleGroupReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example iv_name = 'my-schedule-group'
        lo_scd->deleteschedulegroup(
          iv_name = iv_name ).
        MESSAGE 'Schedule group deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdresourcenotfoundex INTO DATA(lo_not_found_ex).
        DATA(lv_error) = |Schedule group not found: { lo_not_found_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error deleting schedule group: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteScheduleGroupreferência da API AWS SDK for SAP ABAP.

Cenários para o uso do EventBridge Scheduler AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no EventBridge Scheduler com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no EventBridge Scheduler ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Um cenário completo de eventos EventBridge agendados do Scheduler usando um AWS SDK

Um cenário completo de eventos EventBridge agendados do Scheduler usando um AWS SDK

Os exemplos de código a seguir mostram como:

- Implante uma CloudFormation pilha com os recursos necessários.
- Crie um grupo de EventBridge agendamento do Scheduler.
- Crie uma EventBridge agenda única do Scheduler com uma janela de horário flexível.
- Crie uma programação recorrente do EventBridge Scheduler com uma taxa especificada.
- Exclua o EventBridge agendador, a agenda e o grupo de agendamentos.
- Limpe os recursos e exclua a pilha.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário.

```
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.Scheduler;
using Amazon.Scheduler.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
using SchedulerActions;
```

```
using Exception = System.Exception;

namespace SchedulerScenario;

public class SchedulerWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the Amazon
 EventBridge Scheduler workflow:

    1. Prepare the Application:
       - Prompt the user for an email address to use for the subscription for the
 SNS topic subscription.
       - Prompt the user for a name for the Cloud Formation stack.
       - Deploy the Cloud Formation template in resources/cfn_template.yaml for
 resource creation.
       - Store the outputs of the stack into variables for use in the scenario.
       - Create a schedule group for all schedules.

    2. Create one-time Schedule:
       - Create a one-time schedule to send an initial event.
       - Use a Flexible Time Window and set the schedule to delete after
 completion.
       - Wait for the user to receive the event email from SNS.

    3. Create a time-based schedule:
       - Prompt the user for how many X times per Y hours a recurring event
 should be scheduled.
       - Create the scheduled event for X times per hour for Y hours.
       - Wait for the user to receive the event email from SNS.
       - Delete the schedule when the user is finished.

    4. Clean up:
       - Prompt the user for y/n answer if they want to destroy the stack and
 clean up all resources.
       - Delete the schedule group.
       - Destroy the Cloud Formation stack and wait until the stack has been
 removed.
    */

    public static ILogger<SchedulerWorkflow> _logger = null!;
    public static SchedulerWrapper _schedulerWrapper = null!;
```

```
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _roleArn = null!;
    private static string _snsTopicArn = null!;

    public static bool _interactive = true;
    private static string _stackName = "default-scheduler-scenario-stack-name";
    private static string _scheduleGroupName = "scenario-schedules-group";
    private static string _stackResourcePath = "../../../../../../scenarios/
features/eventbridge_scheduler/resources/cfn_template.yaml";

    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonScheduler>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<SchedulerWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<SchedulerWorkflow>();

            _schedulerWrapper =
 host.Services.GetRequiredService<SchedulerWrapper>();
            _amazonCloudFormation =
 host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge Scheduler
 Scenario.");
        Console.WriteLine(new string('-', 80));

        try
```

```
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await CreateOneTimeSchedule();
                Console.WriteLine(new string('-', 80));

                Console.WriteLine(new string('-', 80));
                await CreateRecurringSchedule();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("Amazon EventBridge Scheduler scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        try
        {
            // Prompt the user for an email address to use for the subscription.
            Console.WriteLine("\nThis example creates resources in a
CloudFormation stack, including an SNS topic" +
```

```
                            "\nthat will be subscribed to the EventBridge Scheduler
events. " +
                            "\n\nYou will need to confirm the subscription in order
to receive event emails. ");

            var emailAddress = PromptUserForEmail();

            // Prompt the user for a name for the CloudFormation stack
            _stackName = PromptUserForStackName();

            // Deploy the CloudFormation stack
            var deploySuccess = await DeployCloudFormationStack(_stackName,
emailAddress);

            if (deploySuccess)
            {
                // Create a schedule group for all schedules
                await
_schedulerWrapper.CreateScheduleGroupAsync(_scheduleGroupName);

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }
        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="email">The email to use for the subscription.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
string email)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");
```

```
        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = { Capability.CAPABILITY_NAMED_IAM }
            };

            // If an email is provided, set the parameter.
            if (!string.IsNullOrWhiteSpace(email))
            {
                request.Parameters = new List<Parameter>()
                {
                    new() { ParameterKey = "email", ParameterValue = email }
                };
            }

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                // Wait for the stack to be in CREATE_COMPLETE state
                bool stackCreated = await
WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    // Retrieve the output values
                    var success = await GetStackOutputs(response.StackId);
                    return success;
                }
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
```

```
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName, email);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="client">The CloudFormation client.</param>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 10;
        const int retryDelay = 30000; // 30 seconds.

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
```

```
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    Console.WriteLine("CloudFormation stack creation complete.");
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                        describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    Console.WriteLine("CloudFormation stack creation failed.");
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Retrieves the output values from the CloudFormation stack.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    private static async Task<bool> GetStackOutputs(string stackId)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest { StackName =
stackId };

            var describeStacksResponse =
                await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
```

```
                var stack = describeStacksResponse.Stacks[0];
                _roleArn = GetStackOutputValue(stack, "RoleARN");
                _snsTopicArn = GetStackOutputValue(stack, "SNStopicARN");
                return true;
            }
            else
            {
                _logger.LogError($"No stack found for stack outputs: {stackId}");
                return false;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(
                ex, $"Failed to retrieve CloudFormation stack outputs:
 {stackId}");
            return false;
        }
    }

    /// <summary>
    /// Get an output value by key from a CloudFormation stack.
    /// </summary>
    /// <param name="stack">The CloudFormation stack.</param>
    /// <param name="outputKey">The key of the output.</param>
    /// <returns>The value as a string.</returns>
    private static string GetStackOutputValue(Stack stack, string outputKey)
    {
        var output = stack.Outputs.First(o => o.OutputKey == outputKey);
        var outputValue = output.OutputValue;
        Console.WriteLine($"Stack output {outputKey}: {outputValue}");
        return outputValue;
    }

    /// <summary>
    /// Creates a one-time schedule to send an initial event.
    /// </summary>
    /// <returns>True if the one-time schedule was created successfully.</
returns>
    public static async Task<bool> CreateOneTimeSchedule()
    {
        var scheduleName =
            PromptUserForResourceName("Enter a name for the one-time schedule:");
```

```
        Console.WriteLine($"Creating a one-time schedule named '{scheduleName}' "
+
                          $"\nto send an initial event in 1 minute with a
flexible time window...");
        try
        {
            // Create a one-time schedule with a flexible time
            // window set to delete after completion.
            // You may also set a timezone instead of using UTC.
            var scheduledTime = DateTime.UtcNow.AddMinutes(1).ToString("s");

            var createSuccess = await _schedulerWrapper.CreateScheduleAsync(
                scheduleName,
                $"at({scheduledTime})",
                _scheduleGroupName,
                _snsTopicArn,
                _roleArn,
                $"One time scheduled event test from schedule {scheduleName}.",
                true,
                useFlexibleTimeWindow: true);

            Console.WriteLine($"Subscription email will receive an email from
this event.");
            Console.WriteLine($"You must confirm your subscription to receive
event emails.");

            Console.WriteLine($"One-time schedule '{scheduleName}' created
successfully.");
            return createSuccess;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(ex, $"The target with ARN '{_snsTopicArn}' was not
found.");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the one-time
schedule '{scheduleName}'.");
            return false;
        }
    }
```

```
    /// <summary>
    /// Create a recurring schedule to send events at a specified rate in
 minutes.
    /// </summary>
    /// <returns>True if the recurring schedule was created successfully.</
returns>
    public static async Task<bool> CreateRecurringSchedule()
    {
        Console.WriteLine("Creating a recurring schedule to send events for one
 hour...");

        try
        {
            // Prompt the user for a schedule name.
            var scheduleName =
                PromptUserForResourceName("Enter a name for the recurring
 schedule: ");

            // Prompt the user for the schedule rate (in minutes).
            var scheduleRateInMinutes =
                PromptUserForInteger("Enter the desired schedule rate (in
 minutes): ");

            // Create the recurring schedule.
            var createSuccess = await _schedulerWrapper.CreateScheduleAsync(
                scheduleName,
                $"rate({scheduleRateInMinutes} minutes)",
                _scheduleGroupName,
                _snsTopicArn,
                _roleArn,
                $"Recurrent event test from schedule {scheduleName}.");

            Console.WriteLine($"Subscription email will receive an email from
 this event.");
            Console.WriteLine($"You must confirm your subscription to receive
 event emails.");

            // Delete the schedule when the user is finished.
            if (!_interactive || GetYesNoResponse($"Are you ready to delete the
 '{scheduleName}' schedule? (y/n)"))
            {
                await _schedulerWrapper.DeleteScheduleAsync(scheduleName,
 _scheduleGroupName);
            }
```

```
            return createSuccess;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(ex, $"The target with ARN '{_snsTopicArn}' was not
found.");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while creating the recurring
schedule.");
            return false;
        }
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    /// <returns>True if the cleanup was successful.</returns>
    public static async Task<bool> Cleanup()
    {
        // Prompt the user to confirm cleanup.
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete all resources created by this scenario? (y/n)
");
        if (cleanup)
        {
            try
            {
                // Delete the schedule group.
                var groupDeleteSuccess = await
_schedulerWrapper.DeleteScheduleGroupAsync(_scheduleGroupName);

                // Destroy the CloudFormation stack and wait for it to be
removed.
                var stackDeleteSuccess = await
DeleteCloudFormationStack(_stackName, false);

                return groupDeleteSuccess && stackDeleteSuccess;
            }
            catch (Exception ex)
            {
```

```
                _logger.LogError(ex,
                    "An error occurred while cleaning up the resources.");
                return false;
            }
        }
        _logger.LogInformation("EventBridge Scheduler scenario is complete.");
        return true;
    }

    /// <summary>
    /// Delete the resources in the stack and wait for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the stack.</param>
    /// <param name="forceDelete">True to force delete the stack.</param>
    /// <returns>True if successful.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
 bool forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{_stackName}' is being deleted.
 This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(_stackName, forceDelete);

        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{_stackName}' has been
 deleted.");
            return true;
        }
        else
        {
            _logger.LogError($"Failed to delete CloudFormation stack
 '{_stackName}'.");
```

```
            return false;
        }
    }

    /// <summary>
    /// Wait for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the stack.</param>
    /// <param name="forceDelete">True to force delete the stack.</param>
    /// <returns>True if successful.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 10;
        const int retryDelay = 30000; // 30 seconds

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
describeStacksResponse.Stacks[0].StackStatus == StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus
== StackStatus.DELETE_FAILED)
                {
                    // Try one time to force delete.
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
```

```
                // Stack does not exist, so it has been successfully deleted.
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
 be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack
 '{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompt the user for a valid email address.
    /// </summary>
    /// <returns>The valid email address.</returns>
    private static string PromptUserForEmail()
    {
        if (_interactive)
        {
            Console.WriteLine("Enter an email address to use for event
 subscriptions: ");

            string email = Console.ReadLine()!;

            if (!IsValidEmail(email))
```

```
            {
                Console.WriteLine("Invalid email address. Please try again.");
                return PromptUserForEmail();
            }
            return email;
        }
        // Used when running without user prompts.
        return "";
    }

    /// <summary>
    /// Prompt the user for a non-empty stack name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static string PromptUserForStackName()
    {
        Console.WriteLine("Enter a name for the AWS Cloud Formation Stack: ");
        if (_interactive)
        {
            string stackName = Console.ReadLine()!;
            var regex = "[a-zA-Z][-a-zA-Z0-9]|arn:[-a-zA-Z0-9:/._+]";
            if (!Regex.IsMatch(stackName, regex))
            {
                Console.WriteLine(
                    $"Invalid stack name. Please use a name that matches the
pattern {regex}.");
                return PromptUserForStackName();
            }

            return stackName;
        }
        // Used when running without user prompts.
        return _stackName;
    }

    /// <summary>
    /// Prompt the user for a non-empty resource name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static string PromptUserForResourceName(string prompt)
    {
        if (_interactive)
        {
            Console.WriteLine(prompt);
```

```
            string resourceName = Console.ReadLine()!;
            var regex = "[0-9a-zA-Z-_.]+";
            if (!Regex.IsMatch(resourceName, regex))
            {
                Console.WriteLine($"Invalid resource name. Please use a name that
matches the pattern {regex}.");
                return PromptUserForResourceName(prompt);
            }
            return resourceName!;
        }
        // Used when running without user prompts.
        return "resource-" + Guid.NewGuid();
    }

    /// <summary>
    /// Prompt the user for a non-empty resource name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static int PromptUserForInteger(string prompt)
    {
        if (_interactive)
        {
            Console.WriteLine(prompt);
            string stringResponse = Console.ReadLine()!;
            if (string.IsNullOrWhiteSpace(stringResponse) ||
                !Int32.TryParse(stringResponse, out var intResponse))
            {
                Console.WriteLine($"Invalid integer. ");
                return PromptUserForInteger(prompt);
            }
            return intResponse!;
        }
        // Used when running without user prompts.
        return 1;
    }

    /// <summary>
    /// Use System Mail to check for a valid email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email address.</returns>
    private static bool IsValidEmail(string email)
    {
        try
```

```
        {
            var mailAddress = new System.Net.Mail.MailAddress(email);
            return mailAddress.Address == email;
        }
        catch
        {
            // Invalid emails will cause an exception, return false.
            return false;
        }
    }
}
```

Wrapper para operações de serviço.

```
using Amazon.Scheduler;
using Amazon.Scheduler.Model;
using Microsoft.Extensions.Logging;

namespace SchedulerActions;

/// <summary>
/// Wrapper class for Amazon EventBridge Scheduler operations.
/// </summary>
public class SchedulerWrapper
{
    private readonly IAmazonScheduler _amazonScheduler;
    private readonly ILogger<SchedulerWrapper> _logger;

    /// <summary>
    /// Constructor for the SchedulerWrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EventBridge Scheduler
 client.</param>
    /// <param name="logger">The injected logger.</param>
    public SchedulerWrapper(IAmazonScheduler amazonScheduler,
 ILogger<SchedulerWrapper> logger)
    {
        _amazonScheduler = amazonScheduler;
        _logger = logger;
    }

    /// <summary>
```

```
    /// Creates a new schedule in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule.</param>
    /// <param name="scheduleExpression">The schedule expression that defines
when the schedule should run.</param>
    /// <param name="scheduleGroupName">The name of the schedule group to which
the schedule should be added.</param>
    /// <param name="deleteAfterCompletion">Indicates whether to delete the
schedule after completion.</param>
    /// <param name="useFlexibleTimeWindow">Indicates whether to use a flexible
time window for the schedule.</param>
    /// <param name="targetArn">ARN of the event target.</param>
    /// <param name="roleArn">Execution Role ARN.</param>
    /// <returns>True if the schedule was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleAsync(
            string name,
            string scheduleExpression,
            string scheduleGroupName,
            string targetArn,
            string roleArn,
            string input,
            bool deleteAfterCompletion = false,
            bool useFlexibleTimeWindow = false)
    {
        try
        {
            int hoursToRun = 1;
            int flexibleTimeWindowMinutes = 10;

            var request = new CreateScheduleRequest
            {
                Name = name,
                ScheduleExpression = scheduleExpression,
                GroupName = scheduleGroupName,
                Target = new Target { Arn = targetArn, RoleArn = roleArn, Input =
input },
                ActionAfterCompletion = deleteAfterCompletion
                    ? ActionAfterCompletion.DELETE
                    : ActionAfterCompletion.NONE,
                StartDate = DateTime.UtcNow, // Ignored for one-time schedules.
                EndDate =
                    DateTime.UtcNow
                        .AddHours(hoursToRun) // Ignored for one-time schedules.
```

```
            };
            // Allow a flexible time window if the caller specifies it.
            request.FlexibleTimeWindow = new FlexibleTimeWindow
            {
                Mode = useFlexibleTimeWindow
                    ? FlexibleTimeWindowMode.FLEXIBLE
                    : FlexibleTimeWindowMode.OFF,
                MaximumWindowInMinutes = useFlexibleTimeWindow
                    ? flexibleTimeWindowMinutes
                    : null
            };

            var response = await _amazonScheduler.CreateScheduleAsync(request);

            Console.WriteLine($"Successfully created schedule '{name}' " +
                              $"in schedule group '{scheduleGroupName}':
{response.ScheduleArn}.");
            return true;
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule '{name}' due to a
conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating schedule '{name}'
" +
                             $"in schedule group '{scheduleGroupName}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Creates a new schedule group in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group.</param>
    /// <returns>True if the schedule group was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleGroupAsync(string name)
    {
```

```
        try
        {
            var request = new CreateScheduleGroupRequest { Name = name };

            var response = await
_amazonScheduler.CreateScheduleGroupAsync(request);

            Console.WriteLine($"Successfully created schedule group '{name}':
{response.ScheduleGroupArn}.");
            return true;

        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule group '{name}' due to a
conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating schedule group '{name}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an existing schedule from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule to delete.</param>
    /// <param name="groupName">The group name of the schedule to delete.</param>
    /// <returns>True if the schedule was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleAsync(string name, string groupName)
    {
        try
        {
            var request = new DeleteScheduleRequest
            {
                Name = name,
                GroupName = groupName
            };
```

```
            await _amazonScheduler.DeleteScheduleAsync(request);

            Console.WriteLine($"Successfully deleted schedule with name
'{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule with ID '{name}' because the resource
was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule with ID '{name}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an existing schedule group from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group to delete.</param>
    /// <returns>True if the schedule group was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleGroupAsync(string name)
    {
        try
        {
            var request = new DeleteScheduleGroupRequest { Name = name };

            await _amazonScheduler.DeleteScheduleGroupAsync(request);

            Console.WriteLine($"Successfully deleted schedule group '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
```

```
            _logger.LogError(
                $"Failed to delete schedule group '{name}' because the resource
 was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule group '{name}':
 {ex.Message}");
            return false;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.scheduler.model.SchedulerException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
```

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * This Java code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:
 * <p>
 * 1. Prepare the Application:
 * - Prompt the user for an email address to use for the subscription for the SNS
 topic subscription.
 * - Deploy the Cloud Formation template in resources/cfn_template.yaml for
 resource creation.
 * - Store the outputs of the stack into variables for use in the workflow.
 * - Create a schedule group for all workflow schedules.
 * <p>
 * 2. Create one-time Schedule:
 * - Create a one-time schedule to send an initial event.
 * - Use a Flexible Time Window and set the schedule to delete after completion.
 * - Wait for the user to receive the event email from SNS.
 * <p>
 * 3. Create a time-based schedule:
 * - Prompt the user for how many X times per Y hours a recurring event should be
 scheduled.
 * - Create the scheduled event for X times per hour for Y hours.
 * - Wait for the user to receive the event email from SNS.
 * - Delete the schedule when the user is finished.
 * <p>
 * 4. Clean up:
 * - Prompt the user for y/n answer if they want to destroy the stack and clean
 up all resources.
 * - Delete the schedule group.
 * - Destroy the Cloud Formation stack and wait until the stack has been removed.
 */

public class EventbridgeSchedulerScenario {

    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerScenario.class);
    private static final Scanner scanner = new Scanner(System.in);
    private static String STACK_NAME = "workflow-stack-name";
```

```
    private static final String scheduleGroupName = "schedules-group";

    private static String recurringScheduleName = "";

    private static String oneTimeScheduleName = "";

    private static final EventbridgeSchedulerActions eventbridgeActions = new
EventbridgeSchedulerActions();

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");

    public static String roleArn = "";
    public static String snsTopicArn = "";

    public static void main(String[] args) {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon EventBridge Scheduler Workflow.");
        logger.info("""
            Amazon EventBridge Scheduler is a fully managed service that helps
you schedule and execute
            a wide range of tasks and events in the cloud. It's designed to
simplify the process of
            scheduling and managing recurring or one-time events, making it
easier for developers and
            businesses to automate various workflows and processes.

            One of the key features of Amazon EventBridge Scheduler is its
ability to schedule events
            based on a variety of triggers, including time-based schedules,
custom event patterns, or
            even integration with other AWS services. For example, you can use
EventBridge Scheduler
            to schedule a report generation task to run every weekday at 9 AM, or
to trigger a
            Lambda function when a specific Amazon S3 object is created.

            This flexibility allows you to build complex and dynamic event-driven
architectures
            that adapt to your business needs.

            Lets get started...
            """);
        waitForInputToContinue();
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Prepare the application.");
        waitForInputToContinue();
        try {
            boolean prepareSuccess = prepareApplication();
            logger.info(DASHES);

            if (prepareSuccess) {
                logger.info("2. Create one-time schedule.");
                logger.info("""
                    A one-time schedule in Amazon EventBridge Scheduler is an
event trigger that allows
                    you to schedule a one-time event to run at a specific date
and time. This is useful for
                    executing a specific task or workflow at a predetermined
time, without the need for recurring
                    or complex scheduling.
                    """);
                waitForInputToContinue();
                createOneTimeSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans = scanner.nextLine().trim();
                if (ans.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(oneTimeScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);

                logger.info("3. Create a recurring schedule.");
                logger.info("""
                    A recurring schedule is a feature that allows you to schedule
and manage the execution
                    of your serverless applications or workloads on a recurring
basis. For example,
                    with EventBridge Scheduler, you can create custom schedules
for your AWS Lambda functions,
                    AWS Step Functions, and other supported event sources,
enabling you to automate tasks and
                    workflows without the need for complex infrastructure
management.
                    """);
```

```
                waitForInputToContinue();
                createRecurringSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(recurringScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);
            }
        } catch (Exception ex) {
            logger.info("There was a problem with the workflow {}, initiating
cleanup...", ex.getMessage());
            cleanUp();
        }

        logger.info(DASHES);
        logger.info("4. Clean up the resources.");
        logger.info("Do you want to delete these AWS resources (y/n) ?");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp();
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        logger.info("Amazon EventBridge Scheduler workflow completed.");
        logger.info(DASHES);
    }

    /**
     * Cleans up the resources associated with the EventBridge scheduler.
     * If any errors occur during the cleanup process, the corresponding error
messages are logged.
     */
    public static void cleanUp() {
        logger.info("First, delete the schedule group.");
        logger.info("When the schedule group is deleted, schedules that are part
of that group are deleted.");
        waitForInputToContinue();
        try {

eventbridgeActions.deleteScheduleGroupAsync(scheduleGroupName).join();
```

```
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof SchedulerException schedulerException) {
                logger.error("Scheduler error occurred: Error message: {}, Error
code {}",
                    schedulerException.getMessage(),
schedulerException.awsErrorDetails().errorCode(), schedulerException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }

        logger.info("Destroy the CloudFormation stack");
        waitForInputToContinue();
        CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
    }

    /**
     * Prepares the application by creating resources in a CloudFormation stack,
including an SNS topic
     * that will be subscribed to the EventBridge Scheduler events. The user will
need to confirm the subscription
     * in order to receive event emails.
     *
     * @return true if the application preparation was successful, false
otherwise
     */
    public static boolean prepareApplication() {
        logger.info("""
            This example creates resources in a CloudFormation stack, including
an SNS topic
            that will be subscribed to the EventBridge Scheduler events.
            You will need to confirm the subscription in order to receive event
emails.
             """);

        String emailAddress = promptUserForEmail();
        logger.info("You entered {}", emailAddress);

        logger.info("Do you want to use a custom Stack name (y/n) ?");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
```

```
            String newStackName = scanner.nextLine();
            logger.info("You entered {} for the new stack name", newStackName);
            waitForInputToContinue();
            STACK_NAME = newStackName;
        }

        logger.info("Get the roleArn and snsTopicArn values using a
Cloudformation template.");
        waitForInputToContinue();
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME, emailAddress);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        roleArn = stackOutputs.get("RoleARN");
        snsTopicArn = stackOutputs.get("SNStopicARN");

        logger.info("The roleARN is {}", roleArn);
        logger.info("The snsTopicArn is {}", snsTopicArn);

        try {
            eventbridgeActions.createScheduleGroup(scheduleGroupName).join();
            logger.info("createScheduleGroupAsync completed successfully.");

        } catch (RuntimeException e) {
            logger.error("Error occurred: {} ", e.getMessage());
            return false;
        }
        logger.info("Application preparation complete.");
        return true;
    }

    /**
     * Waits for the user to enter 'c' followed by <ENTER> to continue the
program.
     * This method is used to pause the program execution and wait for user input
before
     * proceeding.
     */
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
```

```
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    /**
     * Prompts the user to enter an email address and validates the input.
     * If the provided email address is invalid, the method will prompt the user
to try again.
     *
     * @return the valid email address entered by the user
     */
    private static String promptUserForEmail() {
        logger.info("Enter an email address to use for event subscriptions: ");
        String email = scanner.nextLine();
        if (!isValidEmail(email)) {
            logger.info("Invalid email address. Please try again.");
            return promptUserForEmail();
        }
        return email;
    }

    /**
     * Checks if the given email address is valid.
     *
     * @param email the email address to be validated
     * @return {@code true} if the email address is valid, {@code false}
otherwise
     */
    private static boolean isValidEmail(String email) {
        try {
            InternetAddress emailAddress = new InternetAddress(email);
            emailAddress.validate();
            return true;

        } catch (AddressException e) {
            return false;
        }
    }
```

```
    /**
     * Creates a one-time schedule to send an initial event in 1 minute with a
 flexible time window.
     *
     * @return {@code true} if the schedule was created successfully, {@code
 false} otherwise
     */
    public static Boolean createOneTimeSchedule() {
        oneTimeScheduleName = promptUserForResourceName("Enter a name for the
 one-time schedule:");
        logger.info("Creating a one-time schedule named {} to send an initial
 event in 1 minute with a flexible time window...", oneTimeScheduleName);
        LocalDateTime scheduledTime = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd'T'HH:mm:ss");

        String scheduleExpression = "at(" + scheduledTime.format(formatter) +
 ")";
        return eventbridgeActions.createScheduleAsync(
            oneTimeScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "One time scheduled event test from schedule",
            true,
            true).join();
    }


    /**
     * Creates a recurring schedule to send events based on a specific time.
     *
     * @return A {@link CompletableFuture} that completes with a boolean value
 indicating the success or failure of the operation.
     */
    public static Boolean createRecurringSchedule() {
        logger.info("Creating a recurring schedule to send events for one
 hour...");
        recurringScheduleName = promptUserForResourceName("Enter a name for the
 recurring schedule:");

        // Prompt the user for the schedule rate (in minutes).
```

```
        int scheduleRateInMinutes = promptUserForInteger("Enter the desired
schedule rate (in minutes): ");
        String scheduleExpression = "rate(" + scheduleRateInMinutes + "
minutes)";
        return eventbridgeActions.createScheduleAsync(
            recurringScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "Recurrent event test from schedule " + recurringScheduleName,
            true,
            true).join();
    }

    /**
     * Prompts the user for a resource name and validates the input.
     *
     * @param prompt the message to display to the user when prompting for the
resource name
     * @return the valid resource name entered by the user
     */
    private static String promptUserForResourceName(String prompt) {
        logger.info(prompt);
        String resourceName = scanner.nextLine();
        String regex = "[0-9a-zA-Z-_.]+";
        if (!resourceName.matches(regex)) {
            logger.info("Invalid resource name. Please use a name that matches
the pattern " + regex + ".");
            return promptUserForResourceName(prompt);
        }
        return resourceName;
    }

    /**
     * Prompts the user for an integer input and returns the integer value.
     *
     * @param prompt the message to be displayed to the user when prompting for
input
     * @return the integer value entered by the user
     */
    private static int promptUserForInteger(String prompt) {
        logger.info(prompt);
        String stringResponse = scanner.nextLine();
```

```
        if (stringResponse == null || stringResponse.trim().isEmpty() || !
isInteger(stringResponse)) {
            logger.info("Invalid integer.");
            return promptUserForInteger(prompt);
        }
        return Integer.parseInt(stringResponse);
    }

    /**
     * Checks if the given string represents a valid integer.
     *
     * @param str the string to be checked
     * @return {@code true} if the string represents a valid integer, {@code
 false} otherwise
     */
    private static boolean isInteger(String str) {
        try {
            Integer.parseInt(str);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }
}
```

Wrapper para operações de serviço.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ActionAfterCompletion;
import software.amazon.awssdk.services.scheduler.model.ConflictException;
import
 software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupRequest;
import
 software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupResponse;
```

```
import software.amazon.awssdk.services.scheduler.model.CreateScheduleRequest;
import
 software.amazon.awssdk.services.scheduler.model.DeleteScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleResponse;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindow;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindowMode;
import software.amazon.awssdk.services.scheduler.model.ResourceNotFoundException;
import software.amazon.awssdk.services.scheduler.model.Target;

import java.time.Instant;
import java.util.concurrent.CompletableFuture;
import java.time.Duration;
import java.util.concurrent.CompletionException;

public class EventbridgeSchedulerActions {

    private static SchedulerAsyncClient schedulerClient;
    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerActions.class);

    public static SchedulerAsyncClient getAsyncClient() {
        if (schedulerClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
```

```
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            schedulerClient = SchedulerAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return schedulerClient;
    }


    /**
     * Creates a new schedule group.
     *
     * @param name the name of the schedule group to be created
     * @return a {@link CompletableFuture} representing the asynchronous
operation of creating the schedule group
     */
    public CompletableFuture<CreateScheduleGroupResponse>
createScheduleGroup(String name) {
        CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
            .name(name)
            .build();

        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
                } else {
                    throw new CompletionException("Failed to create schedule
group: " + name, ex);
                }
            } else if (response == null) {
```

```
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }


    /**
     * Creates a new schedule for a target task.
     *
     * @param name                  the name of the schedule
     * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
     * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
     * @param targetArn             the Amazon Resource Name (ARN) of the target
task
     * @param roleArn               the ARN of the IAM role to be used for the
schedule
     * @param input                 the input data for the target task
     * @param deleteAfterCompletion whether to delete the schedule after it's
executed
     * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
     * @return true if the schedule was successfully created, false otherwise
     */
    public CompletableFuture<Boolean> createScheduleAsync(
        String name,
        String scheduleExpression,
        String scheduleGroupName,
        String targetArn,
        String roleArn,
        String input,
        boolean deleteAfterCompletion,
        boolean useFlexibleTimeWindow) {

        int hoursToRun = 1;
        int flexibleTimeWindowMinutes = 10;
```

```
        Target target = Target.builder()
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group
{}, The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while
creating the schedule: {}", ex.getMessage());
```

```
                        throw new CompletionException("A conflict exception
occurred while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: "
+ ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not
found: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
```

```
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates
 whether the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the
 schedule, except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
 groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
 getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while
 deleting schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.",
 ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class SchedulerScenario:
    """
    A scenario that demonstrates how to use Boto3 to schedule and receive events
 using
    the Amazon EventBridge Scheduler.
    """

    def __init__(
        self,
        scheduler_wrapper: SchedulerWrapper,
        cloud_formation_resource: ServiceResource,
    ):
        self.eventbridge_scheduler = scheduler_wrapper
        self.cloud_formation_resource = cloud_formation_resource
        self.stack: ServiceResource = None
        self.schedule_group_name = None
        self.sns_topic_arn = None
        self.role_arn = None

    def run(self) -> None:
        """
        Runs the scenario.
        """

        print(DASHES)
        print("Welcome to the Amazon EventBridge Scheduler Workflow.")
        print(DASHES)

        print(DASHES)
        self.prepare_application()
        print(DASHES)
```

```
        print(DASHES)
        self.create_one_time_schedule()
        print(DASHES)

        print(DASHES)
        self.create_recurring_schedule()
        print(DASHES)

        print(DASHES)
        if q.ask(
            "Do you want to delete all resources created by this workflow? (y/n)
",
            q.is_yesno,
        ):
            self.cleanup()
        print(DASHES)

        print("Amazon EventBridge Scheduler workflow completed.")

    def prepare_application(self) -> None:
        """
        Prepares the application by prompting the user setup information,
deploying a CloudFormation stack and
        creating a schedule group.
        """
        print("Preparing the application...")
        print(
            "\nThis example creates resources in a CloudFormation stack,
including an SNS topic"
            + "\nthat will be subscribed to the EventBridge Scheduler events. "
            + "\n\nYou will need to confirm the subscription in order to receive
event emails. "
        )

        email_address = q.ask("Enter an email address to use for event
subscriptions: ")
        stack_name = q.ask("Enter a name for the AWS Cloud Formation Stack: ")

        template_file = SchedulerScenario.get_template_as_string()

        parameters = [{"ParameterKey": "email", "ParameterValue": email_address}]

        self.stack = self.deploy_cloudformation_stack(
```

```
            stack_name, template_file, parameters
        )
        outputs = self.stack.outputs
        for output in outputs:
            if output.get("OutputKey") == "RoleARN":
                self.role_arn = output.get("OutputValue")
            elif output.get("OutputKey") == "SNStopicARN":
                self.sns_topic_arn = output.get("OutputValue")

        if not self.sns_topic_arn or not self.role_arn:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
            'sns_topic_arn'={self.sns_topic_arn}, 'role_arn'={self.role_arn}
            """
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"Stack output RoleARN: {self.role_arn}")
        print(f"Stack output SNStopicARN: a")
        schedule_group_name = "scenario-schedules-group"
        schedule_group_arn = self.eventbridge_scheduler.create_schedule_group(
            schedule_group_name
        )
        print(
            f"Successfully created schedule group '{self.schedule_group_name}':
{schedule_group_arn}."
        )
        self.schedule_group_name = schedule_group_name
        print("Application preparation complete.")

    def create_one_time_schedule(self) -> None:
        """
        Creates a one-time schedule to send an initial event.
        """
        schedule_name = q.ask("Enter a name for the one-time schedule:")

        scheduled_time = datetime.now(timezone.utc) + timedelta(minutes=1)
        formatted_scheduled_time = scheduled_time.strftime("%Y-%m-%dT%H:%M:%S")

        print(
            f"Creating a one-time schedule named '{schedule_name}' "
            + f"\nto send an initial event in 1 minute with a flexible time
window..."
        )
```

```
        schedule_arn = self.eventbridge_scheduler.create_schedule(
            schedule_name,
            f"at({formatted_scheduled_time})",
            self.schedule_group_name,
            self.sns_topic_arn,
            self.role_arn,
            f"One time scheduled event test from schedule {schedule_name}.",
            delete_after_completion=True,
            use_flexible_time_window=True,
        )
        print(
            f"Successfully created schedule '{schedule_name}' in schedule group
 'scenario-schedules-group': {schedule_arn}."
        )
        print(f"Subscription email will receive an email from this event.")
        print(f"You must confirm your subscription to receive event emails.")
        print(f"One-time schedule '{schedule_name}' created successfully.")

    def create_recurring_schedule(self) -> None:
        """
        Create a recurring schedule to send events at a specified rate in
 minutes.
        """

        print("Creating a recurring schedule to send events for one hour...")
        schedule_name = q.ask("Enter a name for the recurring schedule: ")
        schedule_rate_in_minutes = q.ask(
            "Enter the desired schedule rate (in minutes): ", q.is_int
        )

        schedule_arn = self.eventbridge_scheduler.create_schedule(
            schedule_name,
            f"rate({schedule_rate_in_minutes} minutes)",
            self.schedule_group_name,
            self.sns_topic_arn,
            self.role_arn,
            f"Recurrent event test from schedule {schedule_name}.",
        )

        print(
            f"Successfully created schedule '{schedule_name}' in schedule group
 'scenario-schedules-group': {schedule_arn}."
        )
```

```
        print(f"Subscription email will receive an email from this event.")
        print(f"You must confirm your subscription to receive event emails.")

        if q.ask(
            f"Are you ready to delete the '{schedule_name}' schedule? (y/n)",
q.is_yesno
        ):
            self.eventbridge_scheduler.delete_schedule(
                schedule_name, self.schedule_group_name
            )

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str, parameters: [dict[str, str]]
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `cfn_template.yaml` AWS CloudFormation script
and are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :param parameters: The parameters for the CloudFormation stack.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=parameters,
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
```

```
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example
resources.
        """
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may take
a few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")

    def cleanup(self) -> None:
        """
        Deletes the CloudFormation stack and the resources created for the demo.
        """

        if self.schedule_group_name:
            schedule_group_name = self.schedule_group_name
            self.schedule_group_name = None
            self.eventbridge_scheduler.delete_schedule_group(schedule_group_name)
            print(f"Successfully deleted schedule group
'{schedule_group_name}'.")

        if self.stack is not None:
            stack = self.stack
            self.stack = None
            self.destroy_cloudformation_stack(stack)
        print("Stack deleted, demo complete.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory, "cfn_template.yaml")
```

```
        file = open(template_file_path, "r")
        return file.read()


if __name__ == "__main__":
    demo: SchedulerScenario = None
    try:
        scheduler_wrapper = SchedulerWrapper.from_client()
        cloud_formation_resource = resource("cloudformation")
        demo = SchedulerScenario(scheduler_wrapper, cloud_formation_resource)
        demo.run()

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup()
```

SchedulerWrapper classe que envolve as ações do Amazon EventBridge Scheduler.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule(
        self,
        name: str,
        schedule_expression: str,
        schedule_group_name: str,
```

```
        target_arn: str,
        role_arn: str,
        input: str,
        delete_after_completion: bool = False,
        use_flexible_time_window: bool = False,
    ) -> str:
        """
        Creates a new schedule with the specified parameters.

        :param name: The name of the schedule.
        :param schedule_expression: The expression that defines when the schedule
runs.
        :param schedule_group_name: The name of the schedule group.
        :param target_arn: The Amazon Resource Name (ARN) of the target.
        :param role_arn: The Amazon Resource Name (ARN) of the execution IAM
role.
        :param input: The input for the target.
        :param delete_after_completion: Whether to delete the schedule after it
completes.
        :param use_flexible_time_window: Whether to use a flexible time window.

        :return The ARN of the created schedule.
        """
        try:
            hours_to_run = 1
            flexible_time_window_minutes = 10
            parameters = {
                "Name": name,
                "ScheduleExpression": schedule_expression,
                "GroupName": schedule_group_name,
                "Target": {"Arn": target_arn, "RoleArn": role_arn, "Input":
input},
                "StartDate": datetime.now(timezone.utc),
                "EndDate": datetime.now(timezone.utc) +
timedelta(hours=hours_to_run),
            }

            if delete_after_completion:
                parameters["ActionAfterCompletion"] = "DELETE"

            if use_flexible_time_window:
                parameters["FlexibleTimeWindow"] = {
                    "Mode": "FLEXIBLE",
                    "MaximumWindowInMinutes": flexible_time_window_minutes,
```

```
                }
            else:
                parameters["FlexibleTimeWindow"] = {"Mode": "OFF"}

            response = self.scheduler_client.create_schedule(**parameters)
            return response["ScheduleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule: %s", err.response["Error"]
["Message"]
                )
            raise


    def delete_schedule(self, name: str, schedule_group_name: str) -> None:
        """
        Deletes the schedule with the specified name and schedule group.

        :param name: The name of the schedule.
        :param schedule_group_name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule(
                Name=name, GroupName=schedule_group_name
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule with ID '%s' because the resource
 was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule: %s", err.response["Error"]
["Message"]
```

```
                )
                raise


    def create_schedule_group(self, name: str) -> str:
        """
        Creates a new schedule group with the specified name and description.

        :param name: The name of the schedule group.
        :param description: The description of the schedule group.

        :return: The ARN of the created schedule group.
        """
        try:
            response = self.scheduler_client.create_schedule_group(Name=name)
            return response["ScheduleGroupArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule group '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule group: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def delete_schedule_group(self, name: str) -> None:
        """
        Deletes the schedule group with the specified name.

        :param name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule_group(Name=name)
            logger.info("Schedule group %s deleted successfully.", name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
```

```
                    "Failed to delete schedule group with ID '%s' because the
resource was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule group: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Exemplos de código para AWS FIS usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Fault Injection Service com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS FIS Guia do usuário — Mais informações sobre AWS FIS.
- AWS FIS Referência da API — Detalhes sobre todas as AWS FIS ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para AWS FIS usar AWS SDKs
  - Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

Cenários para AWS FIS usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS FIS com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS FIS ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
```

```
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true
```

```
        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
```

```
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
```

```
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
```

```
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
```

```
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
```

```
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"
```

```
    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
```

```
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
```

```
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile

- AttachRolePolicy
- CreateExperimentTemplate
- CreateInstanceProfile
- CreateRole
- DeleteAlarms
- DeleteExperimentTemplate
- DeleteInstanceProfile
- DeleteRole
- DeleteRolePolicy
- DescribeAlarms
- DescribeImages
- DetachRolePolicy
- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Exemplos de código para Forecast usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Forecast com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário do Forecast: mais informações sobre o Forecast.
- Referência da API Forecast: detalhes sobre todas as ações disponíveis do Forecast.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Forecast usando AWS SDKs
  - Ações para Forecast usando AWS SDKs
    - Use CreateDatasetcom um AWS SDK
    - Use CreateForecastcom um AWS SDK
    - Use DeleteDatasetcom um AWS SDK
    - Use DeleteForecastcom um AWS SDK
    - Use DescribeForecastcom um AWS SDK
    - Use ListDatasetGroupscom um AWS SDK
    - Use ListForecastscom um AWS SDK

Exemplos básicos de Forecast usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Forecast com AWS SDKs.

Exemplos

- Ações para Forecast usando AWS SDKs
  - Use CreateDatasetcom um AWS SDK
  - Use CreateForecastcom um AWS SDK
  - Use DeleteDatasetcom um AWS SDK

- Use DeleteForecastcom um AWS SDK
- Use DescribeForecastcom um AWS SDK
- Use ListDatasetGroupscom um AWS SDK
- Use ListForecastscom um AWS SDK

Ações para Forecast usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Forecast com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Forecast.

Exemplos

- Use CreateDatasetcom um AWS SDK
- Use CreateForecastcom um AWS SDK
- Use DeleteDatasetcom um AWS SDK
- Use DeleteForecastcom um AWS SDK
- Use DescribeForecastcom um AWS SDK
- Use ListDatasetGroupscom um AWS SDK
- Use ListForecastscom um AWS SDK

Use **`CreateDataset`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateDataset`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name>\s

                Where:
                    name - The name of the data set.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
```

```
        forecast.close();
    }

    public static String createForecastDataSet(ForecastClient forecast, String
 name) {
        try {
            Schema schema = Schema.builder()
                    .attributes(getSchema())
                    .build();

            CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
                    .datasetName(name)
                    .domain("CUSTOM")
                    .datasetType("RELATED_TIME_SERIES")
                    .dataFrequency("D")
                    .schema(schema)
                    .build();

            CreateDatasetResponse response =
 forecast.createDataset(datasetRequest);
            return response.datasetArn();

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

        return "";
    }

    // Create a SchemaAttribute list required to create a data set.
    private static List<SchemaAttribute> getSchema() {

        List<SchemaAttribute> schemaList = new ArrayList<>();
        SchemaAttribute att1 = SchemaAttribute.builder()
                .attributeName("item_id")
                .attributeType("string")
                .build();

        SchemaAttribute att2 = SchemaAttribute.builder()
                .attributeName("timestamp")
                .attributeType("timestamp")
                .build();
```

```
        SchemaAttribute att3 = SchemaAttribute.builder()
                .attributeName("target_value")
                .attributeType("float")
                .build();

        // Push the SchemaAttribute objects to the List.
        schemaList.add(att1);
        schemaList.add(att2);
        schemaList.add(att3);
        return schemaList;
    }
}
```

- Para obter detalhes da API, consulte CreateDatasetta Referência AWS SDK for Java 2.x da API.

Use **`CreateForecast`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateForecast`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name> <predictorArn>\s

                Where:
                    name - The name of the forecast.\s
                    predictorArn - The arn of the predictor to use.\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }

    public static String createNewForecast(ForecastClient forecast, String name,
 String predictorArn) {
        try {
            CreateForecastRequest forecastRequest =
 CreateForecastRequest.builder()
                    .forecastName(name)
                    .predictorArn(predictorArn)
                    .build();
```

```
            CreateForecastResponse response =
 forecast.createForecast(forecastRequest);
            return response.forecastArn();

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

- Para obter detalhes da API, consulte CreateForecasta Referência AWS SDK for Java 2.x da API.

Use **`DeleteDataset`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteDataset`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteDataset {

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

                Usage:
                    <datasetARN>\s

                Where:
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

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

```
    }
}
```

- Para obter detalhes da API, consulte DeleteDatasetta Referência AWS SDK for Java 2.x da API.

Use **`DeleteForecast`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteForecast`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

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

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

                Usage:
```

```
                    <datasetARN>\s

                Where:
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

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

- Para obter detalhes da API, consulte DeleteForecasta Referência AWS SDK for Java 2.x da API.

Use **`DescribeForecast`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeForecast`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <forecastarn>\s

                Where:
                    forecastarn - The arn of the forecast (for example,
 "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String forecastarn = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        describe(forecast, forecastarn);
        forecast.close();
    }

    public static void describe(ForecastClient forecast, String forecastarn) {
        try {
            DescribeForecastRequest request = DescribeForecastRequest.builder()
                    .forecastArn(forecastarn)
                    .build();

            DescribeForecastResponse response =
 forecast.describeForecast(request);
            System.out.println("The name of the forecast is " +
 response.forecastName());

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

- Para obter detalhes da API, consulte DescribeForecasta Referência AWS SDK for Java 2.x da API.

Use **`ListDatasetGroups`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListDatasetGroups`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                    .maxResults(10)
```

```
                    .build();

            ListDatasetGroupsResponse response =
 forecast.listDatasetGroups(group);
            List<DatasetGroupSummary> groups = response.datasetGroups();
            for (DatasetGroupSummary myGroup : groups) {
                System.out.println("The Data Set name is " +
 myGroup.datasetGroupName());
            }

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

- Para obter detalhes da API, consulte ListDatasetGroupsa Referência AWS SDK for Java 2.x da API.

Use **`ListForecasts`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListForecasts`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
```

```
import java.util.List;

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

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listAllForeCasts(forecast);
        forecast.close();
    }

    public static void listAllForeCasts(ForecastClient forecast) {
        try {
            ListForecastsRequest request = ListForecastsRequest.builder()
                    .maxResults(10)
                    .build();

            ListForecastsResponse response = forecast.listForecasts(request);
            List<ForecastSummary> forecasts = response.forecasts();
            for (ForecastSummary forecastSummary : forecasts) {
                System.out.println("The name of the forecast is " +
 forecastSummary.forecastName());
            }

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

- Para obter detalhes da API, consulte ListForecastsa Referência AWS SDK for Java 2.x da API.

Exemplos de código para o Amazon Glacier usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Glacier com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Glacier — Mais informações sobre o Amazon Glacier.
- Referência da API do Amazon Glacier — Detalhes sobre todas as ações disponíveis do Amazon Glacier.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Amazon Glacier usando AWS SDKs
  - Olá, Amazon Glacier
  - Ações para o Amazon Glacier usando AWS SDKs
    - Use AddTagsToVaultcom um AWS SDK ou CLI
    - Use CreateVaultcom um AWS SDK ou CLI
    - Use DeleteArchivecom um AWS SDK ou CLI
    - Use DeleteVaultcom um AWS SDK ou CLI
    - Use DeleteVaultNotificationscom um AWS SDK ou CLI
    - Use DescribeJobcom um AWS SDK ou CLI

    - Use DescribeVaultcom um AWS SDK ou CLI
    - Use GetJobOutputcom um AWS SDK ou CLI
    - Use GetVaultNotificationscom um AWS SDK ou CLI
    - Use InitiateJobcom um AWS SDK ou CLI
    - Use ListJobscom um AWS SDK ou CLI
    - Use ListTagsForVaultcom um AWS SDK ou CLI
    - Use ListVaultscom um AWS SDK ou CLI
    - Use SetVaultNotificationscom um AWS SDK ou CLI
    - Use UploadArchivecom um AWS SDK ou CLI
    - Use UploadMultipartPartcom um AWS SDK ou CLI
- Cenários para o uso do Amazon Glacier AWS SDKs
  - Arquive um arquivo no Amazon Glacier, receba notificações e inicie um trabalho usando um AWS SDK
  - Obtenha o conteúdo de arquivamento do Amazon Glacier e exclua o arquivo usando um AWS SDK

Exemplos básicos do Amazon Glacier usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Glacier com SDKs da AWS .

Exemplos

- Olá, Amazon Glacier
- Ações para o Amazon Glacier usando AWS SDKs
  - Use AddTagsToVaultcom um AWS SDK ou CLI
  - Use CreateVaultcom um AWS SDK ou CLI
  - Use DeleteArchivecom um AWS SDK ou CLI
  - Use DeleteVaultcom um AWS SDK ou CLI
  - Use DeleteVaultNotificationscom um AWS SDK ou CLI
  - Use DescribeJobcom um AWS SDK ou CLI
  - Use DescribeVaultcom um AWS SDK ou CLI
  - Use GetJobOutputcom um AWS SDK ou CLI

- Use GetVaultNotificationscom um AWS SDK ou CLI
- Use InitiateJobcom um AWS SDK ou CLI
- Use ListJobscom um AWS SDK ou CLI
- Use ListTagsForVaultcom um AWS SDK ou CLI
- Use ListVaultscom um AWS SDK ou CLI
- Use SetVaultNotificationscom um AWS SDK ou CLI
- Use UploadArchivecom um AWS SDK ou CLI
- Use UploadMultipartPartcom um AWS SDK ou CLI

Olá, Amazon Glacier

O exemplo de código a seguir mostra como começar a usar o Amazon Glacier.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.Glacier;
using Amazon.Glacier.Model;

namespace GlacierActions;

public static class HelloGlacier
{
    static async Task Main()
    {
        var glacierService = new AmazonGlacierClient();

        Console.WriteLine("Hello Amazon Glacier!");
        Console.WriteLine("Let's list your Glacier vaults:");
```

```
        // You can use await and any of the async methods to get a response.
        // Let's get the vaults using a paginator.
        var glacierVaultPaginator = glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });

        await foreach (var vault in glacierVaultPaginator.VaultList)
        {
            Console.WriteLine($"{vault.CreationDate}:{vault.VaultName}, ARN:
{vault.VaultARN}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListVaultsa Referência AWS SDK para .NET da API.

Ações para o Amazon Glacier usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Glacier com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Glacier e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Amazon Glacier AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Glacier.

Exemplos

- Use AddTagsToVaultcom um AWS SDK ou CLI
- Use CreateVaultcom um AWS SDK ou CLI
- Use DeleteArchivecom um AWS SDK ou CLI
- Use DeleteVaultcom um AWS SDK ou CLI
- Use DeleteVaultNotificationscom um AWS SDK ou CLI
- Use DescribeJobcom um AWS SDK ou CLI
- Use DescribeVaultcom um AWS SDK ou CLI
- Use GetJobOutputcom um AWS SDK ou CLI

- Use GetVaultNotificationscom um AWS SDK ou CLI
- Use InitiateJobcom um AWS SDK ou CLI
- Use ListJobscom um AWS SDK ou CLI
- Use ListTagsForVaultcom um AWS SDK ou CLI
- Use ListVaultscom um AWS SDK ou CLI
- Use SetVaultNotificationscom um AWS SDK ou CLI
- Use UploadArchivecom um AWS SDK ou CLI
- Use UploadMultipartPartcom um AWS SDK ou CLI

Use **`AddTagsToVault`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AddTagsToVault`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Add tags to the items in an Amazon S3 Glacier vault.
   /// </summary>
   /// <param name="vaultName">The name of the vault to add tags to.</param>
   /// <param name="key">The name of the object to tag.</param>
   /// <param name="value">The tag value to add.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> AddTagsToVaultAsync(string vaultName, string key,
string value)
   {
       var request = new AddTagsToVaultRequest
       {
           Tags = new Dictionary<string, string>
               {
                   { key, value },
```

```
                },
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.AddTagsToVaultAsync(request);
        return response.HttpStatusCode == HttpStatusCode.NoContent;
    }
```

- Para obter detalhes da API, consulte AddTagsToVaulta Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando adiciona duas tags ao cofre `my-vault`:

```
aws glacier add-tags-to-vault --account-id - --vault-name my-vault --
tags id=1234,date=july2015
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte AddTagsToVaultem Referência de AWS CLI Comandos.

Use **`CreateVault`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateVault`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Arquivar um arquivo, obter notificações e iniciar um trabalho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create an Amazon S3 Glacier vault.
/// </summary>
/// <param name="vaultName">The name of the vault to create.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateVaultAsync(string vaultName)
{
    var request = new CreateVaultRequest
    {
        // Setting the AccountId to "-" means that
        // the account associated with the current
        // account will be used.
        AccountId = "-",
        VaultName = vaultName,
    };

    var response = await _glacierService.CreateVaultAsync(request);

    Console.WriteLine($"Created {vaultName} at: {response.Location}");

    return response.HttpStatusCode == HttpStatusCode.Created;
}
```

- Para obter detalhes da API, consulte CreateVaulta Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando cria um cofre chamado `my-vault`:

```
aws glacier create-vault --vault-name my-vault --account-id -
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte CreateVaultem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateVault {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to create.
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String
 vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            CreateVaultResponse createVaultResult =
 glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
 createVaultResult.location());

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

- Para obter detalhes da API, consulte CreateVaulta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Crie o cofre.

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateVaulta Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
// Call Glacier to create the vault
glacier.createVault({ vaultName: "YOUR_VAULT_NAME" }, function (err) {
  if (!err) {
    console.log("Created vault!");
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateVaulta Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: criar novo cofre para a conta do usuário. Como nenhum valor foi fornecido ao AccountId parâmetro -, os cmdlets usam o padrão “-” indicando a conta atual.

```
New-GLCVault -VaultName myvault
```

Saída:

```
/01234567812/vaults/myvault
```

- Para obter detalhes da API, consulte CreateVaultem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: criar novo cofre para a conta do usuário. Como nenhum valor foi fornecido ao AccountId parâmetro -, os cmdlets usam o padrão "-" indicando a conta atual.

```
New-GLCVault -VaultName myvault
```

Saída:

```
/01234567812/vaults/myvault
```

- Para obter detalhes da API, consulte CreateVaultem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
```

```
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault
```

- Para obter detalhes da API, consulte a CreateVaultReferência da API AWS SDK for Python (Boto3).

Use **`DeleteArchive`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteArchive`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Obter o conteúdo do arquivo e excluir o arquivo

CLI

AWS CLI

Como excluir um arquivo de um cofre

O exemplo de `delete-archive` a seguir remove o arquivamento especificado de `example_vault`.

```
aws glacier delete-archive \
    --account-id 111122223333  \
    --vault-name example_vault \
    --archive-id Sc0u9ZP8yaWkmh-XGlIvAVprtLhaLCGnNwNl5I5x9HqPIkX5mjc0DrId3Ln-
Gi_k2HzmlIDZUz117KSdVMdMXLuFWi9PJUitxWO73edQ43eTlMWkH0pd9zVSAuV_XXZBVhKhyGhJ7w
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteArchiveem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
```

```
                    vaultName - The name of the vault that contains the archive to
 delete.
                    accountId - The account ID value.
                    archiveId - The archive ID value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String archiveId = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
        glacier.close();
    }

    public static void deleteGlacierArchive(GlacierClient glacier, String
 vaultName, String accountId,
            String archiveId) {
        try {
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                    .vaultName(vaultName)
                    .accountId(accountId)
                    .archiveId(archiveId)
                    .build();

            glacier.deleteArchive(delArcRequest);
            System.out.println("The archive was deleted.");

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

- Para obter detalhes da API, consulte DeleteArchivea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
 archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

- Para obter detalhes da API, consulte a DeleteArchiveReferência da API AWS SDK for Python (Boto3).

Use **`DeleteVault`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteVault`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Obter o conteúdo do arquivo e excluir o arquivo

CLI

AWS CLI

O seguinte comando cria um cofre chamado `my-vault`:

```
aws glacier delete-vault --vault-name my-vault --account-id -
```

Esse comando não produz nenhuma saída. O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte DeleteVaultem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteVault {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void deleteGlacierVault(GlacierClient glacier, String
 vaultName) {
        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");
```

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

- Para obter detalhes da API, consulte DeleteVaulta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
```

```
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteVaultReferência da API AWS SDK for Python (Boto3).

Use **`DeleteVaultNotifications`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteVaultNotifications`.

CLI

AWS CLI

Como remover as notificações do SNS de um cofre

O comando de `delete-vault-notifications` a seguir remove as notificações enviadas pelo Amazon Simple Notification Service (Amazon SNS) do cofre especificado.

```
aws glacier delete-vault-notifications \
    --account-id 111122223333 \
    --vault-name example_vault
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteVaultNotificationsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def stop_notifications(notification):
        """
        Stops notifications to the configured Amazon SNS topic.

        :param notification: The notification configuration to remove.
        """
        try:
            notification.delete()
            logger.info("Notifications stopped.")
        except ClientError:
            logger.exception("Couldn't stop notifications.")
            raise
```

- Para obter detalhes da API, consulte a DeleteVaultNotificationsReferência da API AWS SDK for Python (Boto3).

Use **`DescribeJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeJob`.

CLI

AWS CLI

O seguinte comando recupera informações sobre um trabalho de recuperação de inventário no cofre `my-vault`:

```
aws glacier describe-job --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW
```

Saída:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
    "Completed": false,
    "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "Action": "InventoryRetrieval",
    "CreationDate": "2015-07-17T20:23:41.616Z",
    "StatusCode": "InProgress"
}
```

O ID do trabalho pode ser encontrado na saída de `aws glacier initiate-job` e `aws glacier list-jobs`. O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte DescribeJobem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exibir detalhes do trabalho especificado. Quando o trabalho é concluído com êxito, o Read-GCJobOutput cmdlet pode ser usado para recuperar o conteúdo do trabalho (um arquivo ou uma lista de inventário) no sistema de arquivos local.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Saída:

```
Action                      : ArchiveRetrieval
ArchiveId                   : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash       : 79f3ea754c02f58...dc57bf4395b
```

```
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
JobDescription               :
JobId                        : op1x...JSbthM
JobOutputPath                :
OutputLocation               :
RetrievalByteRange           : 0-38034479
SelectParameters             :
SHA256TreeHash               : 79f3ea754c02f58...dc57bf4395b
SNSTopic                     :
StatusCode                   : InProgress
StatusMessage                :
Tier                         : Standard
VaultARN                     : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- Para obter detalhes da API, consulte DescribeJobem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exibir detalhes do trabalho especificado. Quando o trabalho é concluído com êxito, o Read-GCJobOutput cmdlet pode ser usado para recuperar o conteúdo do trabalho (um arquivo ou uma lista de inventário) no sistema de arquivos local.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Saída:

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
JobDescription               :
JobId                        : op1x...JSbthM
```

```
JobOutputPath                :
OutputLocation               :
RetrievalByteRange           : 0-38034479
SelectParameters             :
SHA256TreeHash               : 79f3ea754c02f58...dc57bf4395b
SNSTopic                     :
StatusCode                   : InProgress
StatusMessage                :
Tier                         : Standard
VaultARN                     : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- Para obter detalhes da API, consulte DescribeJobem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_status(job):
        """
        Gets the status of a job.

        :param job: The job to query.
        :return: The current status of the job.
```

```
        """
        try:
            job.load()
            logger.info(
                "Job %s is performing action %s and has status %s.",
                job.id,
                job.action,
                job.status_code,
            )
        except ClientError:
            logger.exception("Couldn't get status for job %s.", job.id)
            raise
        else:
            return job.status_code
```

- Para obter detalhes da API, consulte a DescribeJobReferência da API AWS SDK for Python (Boto3).

Use **`DescribeVault`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeVault`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Describe an Amazon S3 Glacier vault.
/// </summary>
/// <param name="vaultName">The name of the vault to describe.</param>
/// <returns>The Amazon Resource Name (ARN) of the vault.</returns>
public async Task<string> DescribeVaultAsync(string vaultName)
{
```

```
        var request = new DescribeVaultRequest
        {
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.DescribeVaultAsync(request);

        // Display the information about the vault.
        Console.WriteLine($"{response.VaultName}\tARN: {response.VaultARN}");
        Console.WriteLine($"Created on: {response.CreationDate}\tNumber
 of Archives: {response.NumberOfArchives}\tSize (in bytes):
 {response.SizeInBytes}");
        if (response.LastInventoryDate != DateTime.MinValue)
        {
            Console.WriteLine($"Last inventory: {response.LastInventoryDate}");
        }

        return response.VaultARN;
    }
```

- Para obter detalhes da API, consulte DescribeVaulta Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando recupera dados sobre o cofre `my-vault`:

```
aws glacier describe-vault --vault-name my-vault --account-id -
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte DescribeVaultem Referência de AWS CLI Comandos.

Use **`GetJobOutput`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetJobOutput`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Obter o conteúdo do arquivo e excluir o arquivo

CLI

AWS CLI

O seguinte comando salva a saída de um trabalho de inventário do cofre em um arquivo no diretório atual chamado `output.json`:

```
aws glacier get-job-output --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW output.json
```

O `job-id` está disponível na saída de `aws glacier list-jobs`. Observe que o nome do arquivo de saída é um argumento posicional que não é prefixado por um nome de opção. O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

Saída:

```
{
    "status": 200,
    "acceptRanges": "bytes",
    "contentType": "application/json"
}
```

`output.json`:

```
{"VaultARN":"arn:aws:glacier:us-west-2:0123456789012:vaults/
my-vault","InventoryDate":"2015-04-07T00:26:18Z","ArchiveList":
[{"ArchiveId":"kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw","ArchiveDescription":"multipart
 upload
 test","CreationDate":"2015-04-06T22:24:34Z","Size":3145728,"SHA256TreeHash":"9628195fcdb
```

- Para obter detalhes da API, consulte GetJobOutputem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: baixar o conteúdo do arquivo que foi agendado para recuperação na tarefa especificada e armazená-lo em um arquivo no disco. O download valida a soma de verificação para você, se houver uma disponível. Se desejado, toda a resposta, incluindo a soma de verificação, pode ser exibida especificando **`-Select '*'`**.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- Para obter detalhes da API, consulte GetJobOutputem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: baixar o conteúdo do arquivo que foi agendado para recuperação na tarefa especificada e armazená-lo em um arquivo no disco. O download valida a soma de verificação para você, se houver uma disponível. Se desejado, toda a resposta, incluindo a soma de verificação, pode ser exibida especificando **`-Select '*'`**.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- Para obter detalhes da API, consulte GetJobOutputem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
```

```
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes
```

- Para obter detalhes da API, consulte a GetJobOutputReferência da API AWS SDK for Python (Boto3).

Use **`GetVaultNotifications`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetVaultNotifications`.

CLI

AWS CLI

O seguinte comando captura uma descrição da configuração de notificação do cofre `my-vault`:

```
aws glacier get-vault-notifications --account-id - --vault-name my-vault
```

Saída:

```
{
    "vaultNotificationConfig": {
        "Events": [
            "InventoryRetrievalCompleted",
            "ArchiveRetrievalCompleted"
        ],
        "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault"
    }
}
```

Se nenhuma notificação tiver sido configurada para o cofre, será retornado um erro. O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte GetVaultNotificationsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
```

```
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_notification(vault):
        """
        Gets the currently notification configuration for a vault.

        :param vault: The vault to query.
        :return: The notification configuration for the specified vault.
        """
        try:
            notification = vault.Notification()
            logger.info(
                "Vault %s notifies %s on %s events.",
                vault.name,
                notification.sns_topic,
                notification.events,
            )
        except ClientError:
            logger.exception("Couldn't get notification data for %s.",
vault.name)
            raise
        else:
            return notification
```

- Para obter detalhes da API, consulte a GetVaultNotificationsReferência da API AWS SDK for Python (Boto3).

Use **InitiateJob**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `InitiateJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Arquivar um arquivo, obter notificações e iniciar um trabalho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recupere um arquivo de um cofre. Este exemplo usa a ArchiveTransferManager classe. Para obter detalhes da API, consulte ArchiveTransferManager.

```
    /// <summary>
    /// Download an archive from an Amazon S3 Glacier vault using the Archive
    /// Transfer Manager.
    /// </summary>
    /// <param name="vaultName">The name of the vault containing the object.</
param>
    /// <param name="archiveId">The Id of the archive to download.</param>
    /// <param name="localFilePath">The local directory where the file will
    /// be stored after download.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> DownloadArchiveWithArchiveManagerAsync(string
 vaultName, string archiveId, string localFilePath)
    {
        try
        {
            var manager = new ArchiveTransferManager(_glacierService);

            var options = new DownloadOptions
            {
                StreamTransferProgress = Progress!,
            };

            // Download an archive.
            Console.WriteLine("Initiating the archive retrieval job and then
 polling SQS queue for the archive to be available.");
            Console.WriteLine("When the archive is available, downloading will
 begin.");
            await manager.DownloadAsync(vaultName, archiveId, localFilePath,
 options);
```

```
            return true;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Event handler to track the progress of the Archive Transfer Manager.
    /// </summary>
    /// <param name="sender">The object that raised the event.</param>
    /// <param name="args">The argument values from the object that raised the
    /// event.</param>
    static void Progress(object sender, StreamTransferProgressArgs args)
    {
        if (args.PercentDone != _currentPercentage)
        {
            _currentPercentage = args.PercentDone;
            Console.WriteLine($"Downloaded {_currentPercentage}%");
        }
    }
```

- Para obter detalhes da API, consulte InitiateJoba Referência AWS SDK para .NET da API.

CLI

AWS CLI

O comando a seguir inicia um trabalho para obter um inventário de cofre `my-vault`:

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-parameters
 '{"Type": "inventory-retrieval"}'
```

Saída:

```
{
```

```
    "location": "/0123456789012/vaults/my-vault/jobs/
zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "jobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW"
}
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

O comando a seguir inicia um trabalho para recuperar um arquivo do cofre `my-vault`:

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-
parameters file://job-archive-retrieval.json
```

`job-archive-retrieval.json` é um arquivo JSON na pasta local que especifica o tipo de trabalho, a ID do arquivo e alguns parâmetros opcionais:

```
{
  "Type": "archive-retrieval",
  "ArchiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
  "Description": "Retrieve archive on 2015-07-17",
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-topic"
}
```

As IDs de arquivo estão disponíveis na saída de `aws glacier upload-archive` e `aws glacier get-job-output`.

Saída:

```
{
    "location": "/011685312445/vaults/mwunderl/jobs/l7IL5-
EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
    "jobId": "l7IL5-EkXy2O5uLYaFdAYOiEY9Ws95fClzIbk-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav"
}
```

Consulte Iniciar Tarefa na Referência de API do Amazon Glacier para ver detalhes sobre o formato dos parâmetros da tarefa.

- Para obter detalhes da API, consulte InitiateJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recupere um inventário do cofre.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <vaultName> <accountId> <path>

                Where:
                   vaultName - The name of the vault.
                   accountId - The account ID value.
                   path - The path where the file is written to.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String jobNum = createJob(glacier, vaultName, accountId);
        checkJob(glacier, jobNum, vaultName, accountId, path);
        glacier.close();
    }

    public static String createJob(GlacierClient glacier, String vaultName,
 String accountId) {
        try {
            JobParameters job = JobParameters.builder()
                    .type("inventory-retrieval")
                    .build();

            InitiateJobRequest initJob = InitiateJobRequest.builder()
                    .jobParameters(job)
                    .accountId(accountId)
                    .vaultName(vaultName)
                    .build();

            InitiateJobResponse response = glacier.initiateJob(initJob);
            System.out.println("The job ID is: " + response.jobId());
            System.out.println("The relative URI path of the job is: " +
 response.location());
```

```
            return response.jobId();

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

        }
        return "";
    }

    // Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
    // Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
 String account, String path) {
        try {
            boolean finished = false;
            String jobStatus;
            int yy = 0;

            while (!finished) {
                DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                        .jobId(jobId)
                        .accountId(account)
                        .vaultName(name)
                        .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

                if (jobStatus.compareTo("Succeeded") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + jobStatus);
                    Thread.sleep(1000);
                }
                yy++;
            }

            System.out.println("Job has Succeeded");
            GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
                    .jobId(jobId)
                    .vaultName(name)
                    .accountId(account)
                    .build();
```

```
            ResponseBytes<GetJobOutputResponse> objectBytes =
 glacier.getJobOutputAsBytes(jobOutputRequest);
            // Write the data to a local file.
            byte[] data = objectBytes.asByteArray();
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from a Glacier
 vault");
            os.close();

        } catch (GlacierException | InterruptedException | IOException e) {
            System.out.println(e.getMessage());
            System.exit(1);

        }
    }
}
```

- Para obter detalhes da API, consulte InitiateJoba Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: iniciar um trabalho para recuperar um arquivo do cofre especificado de propriedade do usuário. O status do trabalho pode ser verificado usando o Get-GLCJob cmdlet. Quando o trabalho é concluído com êxito, o Read-GCJobOutput cmdlet pode ser usado para recuperar o conteúdo do arquivamento no sistema de arquivos local.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Saída:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- Para obter detalhes da API, consulte InitiateJobem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: iniciar um trabalho para recuperar um arquivo do cofre especificado de propriedade do usuário. O status do trabalho pode ser verificado usando o Get-GLCJob cmdlet. Quando o trabalho é concluído com êxito, o Read-GCJobOutput cmdlet pode ser usado para recuperar o conteúdo do arquivamento no sistema de arquivos local.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Saída:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- Para obter detalhes da API, consulte InitiateJobem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recupere um inventário do cofre.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
```

```
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_inventory_retrieval(vault):
        """
        Initiates an inventory retrieval job. The inventory describes the
 contents
        of the vault. Standard retrievals typically complete within 3–5 hours.
        When the job completes, you can get the inventory by calling
 get_output().

        :param vault: The vault to inventory.
        :return: The inventory retrieval job.
        """
        try:
            job = vault.initiate_inventory_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on vault %s.", vault.name)
            raise
        else:
            return job
```

Recupere um arquivo de um cofre.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
```

```
        Initiates an archive retrieval job. Standard retrievals typically
complete
        within 3-5 hours. When the job completes, you can get the archive
contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job
```

- Para obter detalhes da API, consulte a InitiateJobReferência da API AWS SDK for Python (Boto3).

Use **`ListJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListJobs`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Arquivar um arquivo, obter notificações e iniciar um trabalho
- Obter o conteúdo do arquivo e excluir o arquivo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List Amazon S3 Glacier jobs.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list jobs for.</param>
    /// <returns>A list of Amazon S3 Glacier jobs.</returns>
    public async Task<List<GlacierJobDescription>> ListJobsAsync(string
vaultName)
    {
        var request = new ListJobsRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the current account.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListJobsAsync(request);

        return response.JobList;
    }
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando lista trabalhos em andamento e concluídos recentemente no cofre `my-vault`:

```
aws glacier list-jobs --account-id - --vault-name my-vault
```

Saída:

```
{
    "JobList": [
        {
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "RetrievalByteRange": "0-3145727",
            "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
            "Completed": false,
            "SHA256TreeHash":
 "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
            "JobId": "l7IL5-EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
            "ArchiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
            "JobDescription": "Retrieve archive on 2015-07-17",
            "ArchiveSizeInBytes": 3145728,
            "Action": "ArchiveRetrieval",
            "ArchiveSHA256TreeHash":
 "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
            "CreationDate": "2015-07-17T21:16:13.840Z",
            "StatusCode": "InProgress"
        },
        {
            "InventoryRetrievalParameters": {
                "Format": "JSON"
            },
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "Completed": false,
            "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
```

```
            "Action": "InventoryRetrieval",
            "CreationDate": "2015-07-17T20:23:41.616Z",
            "StatusCode": ""InProgress""
        }
    ]
}
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte ListJobsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
```

```
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list
```

- Para obter detalhes da API, consulte a ListJobsReferência da API AWS SDK for Python (Boto3).

Use **`ListTagsForVault`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTagsForVault`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List tags for an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list tags for.</param>
    /// <returns>A dictionary listing the tags attached to each object in the
    /// vault and its tags.</returns>
    public async Task<Dictionary<string, string>> ListTagsForVaultAsync(string
 vaultName)
    {
        var request = new ListTagsForVaultRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListTagsForVaultAsync(request);

        return response.Tags;
    }
```

- Para obter detalhes da API, consulte ListTagsForVaulta Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando lista as tags aplicadas ao cofre `my-vault`:

```
aws glacier list-tags-for-vault --account-id - --vault-name my-vault
```

Saída:

```
{
    "Tags": {
        "date": "july2015",
```

```
        "id": "1234"
    }
}
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte ListTagsForVaultem Referência de AWS CLI Comandos.

Use **`ListVaults`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListVaults`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Arquivar um arquivo, obter notificações e iniciar um trabalho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the Amazon S3 Glacier vaults associated with the current account.
    /// </summary>
    /// <returns>A list containing information about each vault.</returns>
    public async Task<List<DescribeVaultOutput>> ListVaultsAsync()
    {
        var glacierVaultPaginator = _glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });
        var vaultList = new List<DescribeVaultOutput>();

        await foreach (var vault in glacierVaultPaginator.VaultList)
```

```
        {
            vaultList.Add(vault);
        }

        return vaultList;
    }
```

- Para obter detalhes da API, consulte ListVaultsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando lista os cofres na conta e na região padrão:

```
aws glacier list-vaults --account-id -
```

Saída:

```
{
    "VaultList": [
        {
            "SizeInBytes": 3178496,
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "LastInventoryDate": "2015-04-07T00:26:19.028Z",
            "VaultName": "my-vault",
            "NumberOfArchives": 1,
            "CreationDate": "2015-04-06T21:23:45.708Z"
        }
    ]
}
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte ListVaultsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
```

```
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v : vaultList) {
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }

                // Check for further results.
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }

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

- Para obter detalhes da API, consulte ListVaultsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

- Para obter detalhes da API, consulte a ListVaultsReferência da API AWS SDK for Python (Boto3).

Use **`SetVaultNotifications`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetVaultNotifications`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Arquivar um arquivo, obter notificações e iniciar um trabalho

CLI

AWS CLI

O seguinte comando configura as notificações do SNS para o cofre `my-vault`:

```
aws glacier set-vault-notifications --account-id - --vault-name my-vault --vault-notification-config file://notificationconfig.json
```

`notificationconfig.json` é um arquivo JSON na pasta atual que especifica um tópico do SNS e os eventos a serem publicados:

```
{
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
  "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

- Para obter detalhes da API, consulte SetVaultNotificationsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
```

```
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
vault.name
            )
            raise
        else:
            return notification
```

- Para obter detalhes da API, consulte a SetVaultNotificationsReferência da API AWS SDK for Python (Boto3).

Use **`UploadArchive`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UploadArchive`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Arquivar um arquivo, obter notificações e iniciar um trabalho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Upload an object to an Amazon S3 Glacier vault.
   /// </summary>
   /// <param name="vaultName">The name of the Amazon S3 Glacier vault to upload
   /// the archive to.</param>
   /// <param name="archiveFilePath">The file path of the archive to upload to
the vault.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<string> UploadArchiveWithArchiveManager(string vaultName,
string archiveFilePath)
   {
       try
       {
           var manager = new ArchiveTransferManager(_glacierService);
```

```
            // Upload an archive.
            var response = await manager.UploadAsync(vaultName, "upload archive
 test", archiveFilePath);
            return response.ArchiveId;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return string.Empty;
        }
    }
```

- Para obter detalhes da API, consulte UploadArchivea Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando faz upload um arquivo na pasta atual com o nome `archive.zip` para um cofre chamado `my-vault`:

```
aws glacier upload-archive --account-id - --vault-name my-vault --
body archive.zip
```

Saída:

```
{
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum":
 "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
    "location": "/0123456789012/vaults/my-vault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

Para recuperar um arquivamento carregado, inicie um trabalho de recuperação com o comando aws glacier initiate-job.

- Para obter detalhes da API, consulte UploadArchiveem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

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

```
    static final int ONE_MB = 1024 * 1024;

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

                Usage:   <strPath> <vaultName>\s

                Where:
                   strPath - The path to the archive to upload (for example, C:\
\AWS\\test.pdf).
                   vaultName - The name of the vault.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
 vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest,
 path);
            return res.archiveId();
```

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

    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s",
inputFile, ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws
IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
```

```
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
```

```
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
```

```
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

- Para obter detalhes da API, consulte UploadArchivea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Faça upload do arquivo.

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
```

```
// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
  try {
    const data = await glacierClient.send(new UploadArchiveCommand(params));
    console.log("Archive ID", data.archiveId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error uploading archive!", err);
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UploadArchivea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object and buffer
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = { vaultName: "YOUR_VAULT_NAME", body: buffer };
// Call Glacier to upload the archive.
glacier.uploadArchive(params, function (err, data) {
  if (err) {
```

```
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UploadArchivea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: carregar um único arquivo no cofre especificado, exibindo o ID do arquivo e a soma de verificação computada.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Saída:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Exemplo 2: carregar o conteúdo de uma hierarquia de pastas no cofre especificado na conta do usuário. Para cada arquivo carregado, o cmdlet emite o nome do arquivo, o ID do arquivo correspondente e a soma de verificação computada do arquivo.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Saída:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
```

```
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- Para obter detalhes da API, consulte UploadArchiveem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: carregar um único arquivo no cofre especificado, exibindo o ID do arquivo e a soma de verificação computada.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Saída:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Exemplo 2: carregar o conteúdo de uma hierarquia de pastas no cofre especificado na conta do usuário. Para cada arquivo carregado, o cmdlet emite o nome do arquivo, o ID do arquivo correspondente e a soma de verificação computada do arquivo.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Saída:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- Para obter detalhes da API, consulte UploadArchiveem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
```

```
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive
```

- Para obter detalhes da API, consulte a UploadArchiveReferência da API AWS SDK for Python (Boto3).

Use **`UploadMultipartPart`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UploadMultipartPart`.

CLI

AWS CLI

O comando a seguir carrega a primeira parte de 1 MiB (1024 x 1024 bytes) de um arquivamento:

```
aws glacier upload-multipart-part --body part1 --range 'bytes
 0-1048575/*' --account-id - --vault-name my-vault --upload-
id 19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
OssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ
```

O Amazon Glacier exige um argumento de ID de conta ao realizar operações, mas você pode usar um hífen para especificar a conta em uso.

O parâmetro de corpo segue um caminho para um arquivo da parte no sistema de arquivos local. O parâmetro de intervalo usa um intervalo de conteúdo HTTP indicando os bytes que a peça ocupa no arquivamento concluído. O ID de upload é retornado pelo comando `aws glacier initiate-multipart-upload` e também pode ser coletado usando `aws glacier list-multipart-uploads`.

Para obter mais informações sobre uploads de várias partes para o Amazon Glacier usando a AWS CLI, consulte Como usar o Amazon Glacier no Guia do usuário da CLI.AWS

- Para obter detalhes da API, consulte UploadMultipartPartem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um carregamento fracionado com pedaços de 1 megabyte de um objeto Buffer.

```
// Create a new service object and some supporting variables
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" }),
  vaultName = "YOUR_VAULT_NAME",
  buffer = new Buffer(2.5 * 1024 * 1024), // 2.5MB buffer
  partSize = 1024 * 1024, // 1MB chunks,
  numPartsLeft = Math.ceil(buffer.length / partSize),
  startTime = new Date(),
  params = { vaultName: vaultName, partSize: partSize.toString() };

// Compute the complete SHA-256 tree hash so we can pass it
// to completeMultipartUpload request at the end
var treeHash = glacier.computeChecksums(buffer).treeHash;

// Initiate the multipart upload
console.log("Initiating upload to", vaultName);
// Call Glacier to initiate the upload.
glacier.initiateMultipartUpload(params, function (mpErr, multipart) {
  if (mpErr) {
    console.log("Error!", mpErr.stack);
    return;
  }
  console.log("Got upload ID", multipart.uploadId);

  // Grab each partSize chunk and upload it as a part
  for (var i = 0; i < buffer.length; i += partSize) {
    var end = Math.min(i + partSize, buffer.length),
      partParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        range: "bytes " + i + "-" + (end - 1) + "/*",
        body: buffer.slice(i, end),
```

```
      };

    // Send a single part
    console.log("Uploading part", i, "=", partParams.range);
    glacier.uploadMultipartPart(partParams, function (multiErr, mData) {
      if (multiErr) return;
      console.log("Completed part", this.request.params.range);
      if (--numPartsLeft > 0) return; // complete only when all parts uploaded

      var doneParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        archiveSize: buffer.length.toString(),
        checksum: treeHash, // the computed tree hash
      };

      console.log("Completing upload...");
      glacier.completeMultipartUpload(doneParams, function (err, data) {
        if (err) {
          console.log("An error occurred while uploading the archive");
          console.log(err);
        } else {
          var delta = (new Date() - startTime) / 1000;
          console.log("Completed upload in", delta, "seconds");
          console.log("Archive ID:", data.archiveId);
          console.log("Checksum:  ", data.checksum);
        }
      });
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UploadMultipartPart a Referência AWS SDK para JavaScript da API.

Cenários para o uso do Amazon Glacier AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Glacier com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon Glacier ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Arquive um arquivo no Amazon Glacier, receba notificações e inicie um trabalho usando um AWS SDK
- Obtenha o conteúdo de arquivamento do Amazon Glacier e exclua o arquivo usando um AWS SDK

Arquive um arquivo no Amazon Glacier, receba notificações e inicie um trabalho usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um cofre do Amazon Glacier.
- Configurar o cofre para publicar notificações em um tópico do Amazon SNS.
- Fazer upload de um arquivo para o cofre.
- Iniciar um trabalho de recuperação de arquivo.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva operações do Amazon Glacier.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
```

```
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically
complete
        within 3-5 hours. When the job completes, you can get the archive
contents
        by calling get_output().
```

```
        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
```

```
        else:
            return job_list


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
vault.name
            )
            raise
        else:
            return notification
```

Chame funções na classe wrapper para criar um cofre e fazer upload de um arquivo, depois configure o cofre para publicar notificações e iniciar um trabalho para recuperar o arquivo.

```
def upload_demo(glacier, vault_name, topic_arn):
    """
    Shows how to:
    * Create a vault.
    * Configure the vault to publish notifications to an Amazon SNS topic.
    * Upload an archive.
    * Start a job to retrieve the archive.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to create.
    :param topic_arn: The ARN of an Amazon SNS topic that receives notification
 of
                      Amazon S3 Glacier events.
    """
    print(f"\nCreating vault {vault_name}.")
    vault = glacier.create_vault(vault_name)
    print("\nList of vaults in your account:")
    glacier.list_vaults()
    print(f"\nUploading glacier_basics.py to {vault.name}.")
    with open("glacier_basics.py", "rb") as upload_file:
        archive = glacier.upload_archive(vault, "glacier_basics.py", upload_file)
    print(
        "\nStarting an archive retrieval request to get the file back from the "
        "vault."
    )
    glacier.initiate_archive_retrieval(archive)
    print("\nListing in progress jobs:")
    glacier.list_jobs(vault, "in_progress")
    print(
        "\nBecause Amazon S3 Glacier is intended for infrequent retrieval, an "
        "archive request with Standard retrieval typically completes within 3-5 "
        "hours."
    )
    if topic_arn:
        notification = glacier.set_notifications(vault, topic_arn)
        print(
            f"\nVault {vault.name} is configured to notify the "
            f"{notification.sns_topic} topic when {notification.events} "
            f"events occur. You can subscribe to this topic to receive "
            f"a message when the archive retrieval completes.\n"
```

```
        )
    else:
        print(
            f"\nVault {vault.name} is not configured to notify an Amazon SNS
topic "
            f"when the archive retrieval completes so wait a few hours."
        )
    print("\nRetrieve your job output by running this script with the --retrieve
flag.")
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateVault
  - InitiateJob
  - ListJobs
  - ListVaults
  - SetVaultNotifications
  - UploadArchive

Obtenha o conteúdo de arquivamento do Amazon Glacier e exclua o arquivo usando um AWS SDK

O exemplo de código a seguir mostra como:

- Listar trabalhos para um cofre do Amazon Glacier e obter o status do trabalho.
- Obter a saída de um trabalho de recuperação de arquivo concluído.
- Excluir um arquivo.
- Excluir um cofre.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva operações do Amazon Glacier.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
```

```
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
```

```
        else:
            return out_bytes


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
 archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

Chame funções na classe wrapper para obter o conteúdo de arquivo de um trabalho concluído e, em seguida, exclua o arquivo.

```
def retrieve_demo(glacier, vault_name):
    """
```

```
    Shows how to:
    * List jobs for a vault and get job status.
    * Get the output of a completed archive retrieval job.
    * Delete an archive.
    * Delete a vault.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to query for jobs.
    """
    vault = glacier.glacier_resource.Vault("-", vault_name)
    try:
        vault.load()
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            print(
                f"\nVault {vault_name} doesn't exist. You must first run this
script "
                f"with the --upload flag to create the vault."
            )
            return
        else:
            raise

    print(f"\nGetting completed jobs for {vault.name}.")
    jobs = glacier.list_jobs(vault, "completed")
    if not jobs:
        print("\nNo completed jobs found. Give it some time and try again
later.")
        return

    retrieval_job = None
    for job in jobs:
        if job.action == "ArchiveRetrieval" and job.status_code == "Succeeded":
            retrieval_job = job
            break
    if retrieval_job is None:
        print(
            "\nNo ArchiveRetrieval jobs found. Give it some time and try again "
            "later."
        )
        return

    print(f"\nGetting output from job {retrieval_job.id}.")
    archive_bytes = glacier.get_job_output(retrieval_job)
```

```
    archive_str = archive_bytes.decode("utf-8")
    print("\nGot archive data. Printing the first 10 lines.")
    print(os.linesep.join(archive_str.split(os.linesep)[:10]))

    print(f"\nDeleting the archive from {vault.name}.")
    archive = glacier.glacier_resource.Archive(
        "-", vault.name, retrieval_job.archive_id
    )
    glacier.delete_archive(archive)

    print(f"\nDeleting {vault.name}.")
    glacier.delete_vault(vault)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DeleteArchive
  - DeleteVault
  - GetJobOutput
  - ListJobs

Exemplos de código para AWS Glue usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Glue com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS Glue Guia do desenvolvedor — Mais informações sobre AWS Glue.
- AWS Glue Referência da API — Detalhes sobre todas as AWS Glue ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Glue usar AWS SDKs
  - Hello AWS Glue
  - Aprenda as noções básicas do AWS Glue com um AWS SDK
  - Ações para AWS Glue usar AWS SDKs
    - Use CreateCrawlercom um AWS SDK
    - Use CreateJobcom um AWS SDK ou CLI
    - Use DeleteCrawlercom um AWS SDK
    - Use DeleteDatabasecom um AWS SDK
    - Use DeleteJobcom um AWS SDK ou CLI
    - Use DeleteTablecom um AWS SDK
    - Use GetCrawlercom um AWS SDK
    - Use GetDatabasecom um AWS SDK
    - Use GetDatabasescom um AWS SDK ou CLI
    - Use GetJobcom um AWS SDK ou CLI
    - Use GetJobRuncom um AWS SDK ou CLI
    - Use GetJobRunscom um AWS SDK ou CLI
    - Use GetTablescom um AWS SDK ou CLI
    - Use ListJobscom um AWS SDK
    - Use StartCrawlercom um AWS SDK ou CLI
    - Use StartJobRuncom um AWS SDK ou CLI
  - Cenários para AWS Glue usar AWS SDKs

- Conceitos básicos do catálogo de dados

Exemplos básicos para AWS Glue usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Glue com AWS SDKs.

Exemplos

- Hello AWS Glue
- Aprenda as noções básicas do AWS Glue com um AWS SDK
- Ações para AWS Glue usar AWS SDKs
  - Use CreateCrawlercom um AWS SDK
  - Use CreateJobcom um AWS SDK ou CLI
  - Use DeleteCrawlercom um AWS SDK
  - Use DeleteDatabasecom um AWS SDK
  - Use DeleteJobcom um AWS SDK ou CLI
  - Use DeleteTablecom um AWS SDK
  - Use GetCrawlercom um AWS SDK
  - Use GetDatabasecom um AWS SDK
  - Use GetDatabasescom um AWS SDK ou CLI
  - Use GetJobcom um AWS SDK ou CLI
  - Use GetJobRuncom um AWS SDK ou CLI
  - Use GetJobRunscom um AWS SDK ou CLI
  - Use GetTablescom um AWS SDK ou CLI
  - Use ListJobscom um AWS SDK
  - Use StartCrawlercom um AWS SDK ou CLI
  - Use StartJobRuncom um AWS SDK ou CLI

Hello AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace GlueActions;

public class HelloGlue
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonGlue>()
                .AddTransient<GlueWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloGlue>();
        var glueClient = host.Services.GetRequiredService<IAmazonGlue>();

        var request = new ListJobsRequest();

        var jobNames = new List<string>();

        do
        {
```

```
            var response = await glueClient.ListJobsAsync(request);
            jobNames.AddRange(response.JobNames);
            request.NextToken = response.NextToken;
        }
        while (request.NextToken is not null);

        Console.Clear();
        Console.WriteLine("Hello, Glue. Let's list your existing Glue Jobs:");
        if (jobNames.Count == 0)
        {
            Console.WriteLine("You don't have any AWS Glue jobs.");
        }
        else
        {
            jobNames.ForEach(Console.WriteLine);
        }
    }
}
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
```

```
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_glue.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_glue.cpp.

```
#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
```

```
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 *  A "Hello Glue" starter application which initializes an AWS Glue client and
 lists the
 *  AWS Glue job definitions.
 *
 *  main function
 *
 *  Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken;  // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }

            Aws::Glue::Model::ListJobsOutcome listRunsOutcome =
 glueClient.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());
```

```
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Your account has " << jobs.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < jobs.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << jobs[i] << std::endl;
        }
    }
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;
```

```
public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ListJobsCommand, GlueClient } from "@aws-sdk/client-glue";

const client = new GlueClient({});

export const main = async () => {
  const command = new ListJobsCommand({});

  const { JobNames } = await client.send(command);
```

```
  const formattedJobNames = JobNames.join("\n");
  console.log("Job names: ");
  console.log(formattedJobNames);
  return JobNames;
};
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
from botocore.exceptions import ClientError


def hello_glue():
    """
    Lists the job definitions in your AWS Glue account, using the AWS SDK for
 Python (Boto3).
    """
    try:
        # Create the Glue client
        glue = boto3.client("glue")

        # List the jobs, limiting the results to 10 per page
        paginator = glue.get_paginator("get_jobs")
        response_iterator = paginator.paginate(
            PaginationConfig={"MaxItems": 10, "PageSize": 10}
        )

        # Print the job names
```

```
        print("Here are the jobs in your account:")
        for page in response_iterator:
            for job in page["Jobs"]:
                print(f"\t{job['Name']}")

    except ClientError as e:
        print(f"Error: {e}")


if __name__ == "__main__":
    hello_glue()
```

- Para obter detalhes da API, consulte a ListJobsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-glue'
require 'logger'

# GlueManager is a class responsible for managing AWS Glue operations
# such as listing all Glue jobs in the current AWS account.
class GlueManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all Glue jobs in the current AWS account.
  def list_jobs
```

```
    @logger.info('Here are the Glue jobs in your account:')

    paginator = @client.get_jobs(max_results: 10)
    jobs = []

    paginator.each_page do |page|
      jobs.concat(page.jobs)
    end

    if jobs.empty?
      @logger.info("You don't have any Glue jobs.")
    else
      jobs.each do |job|
        @logger.info("- #{job.name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  glue_client = Aws::Glue::Client.new
  manager = GlueManager.new(glue_client)
  manager.list_jobs
end
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
```

```
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- Para obter detalhes da API, consulte a ListJobsreferência da API AWS SDK for Rust.

Aprenda as noções básicas do AWS Glue com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as AWS Glue funções usadas no cenário.

```
using System.Net;
```

```
namespace GlueActions;

public class GlueWrapper
{
    private readonly IAmazonGlue _amazonGlue;

    /// <summary>
    /// Constructor for the AWS Glue actions wrapper.
    /// </summary>
    /// <param name="amazonGlue"></param>
    public GlueWrapper(IAmazonGlue amazonGlue)
    {
        _amazonGlue = amazonGlue;
    }

    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
 executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
 (Amazon S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
```

```
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };
```

```
        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
 DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
```

```
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };
```

```
        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
 found.");
        return null;
    }


    /// <summary>
    /// Get information about the state of an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A value describing the state of the crawler.</returns>
    public async Task<CrawlerState> GetCrawlerStateAsync(string crawlerName)
    {
        var response = await _amazonGlue.GetCrawlerAsync(
            new GetCrawlerRequest { Name = crawlerName });
        return response.Crawler.State;
    }


    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }
```

```
    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }


    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }
```

```
    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }


    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
 ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }


    /// <summary>
```

```
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }

}
```

Crie uma classe que execute o cenário.

```
global using Amazon.Glue;
global using GlueActions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.Glue.Model;
using Amazon.S3;
using Amazon.S3.Model;

namespace GlueBasics;

public class GlueBasics
{
    private static ILogger logger = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonGlue>()
            .AddTransient<GlueWrapper>()
            .AddTransient<UiWrapper>()
            )
            .Build();
```

```
        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
        .CreateLogger<GlueBasics>();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // These values are stored in settings.json
        // Once you have run the CDK script to deploy the resources,
        // edit the file to set "BucketName", "RoleName", and "ScriptURL"
        // to the appropriate values. Also set "CrawlerName" to the name
        // you want to give the crawler when it is created.
        string bucketName = _configuration["BucketName"]!;
        string bucketUrl = _configuration["BucketUrl"]!;
        string crawlerName = _configuration["CrawlerName"]!;
        string roleName = _configuration["RoleName"]!;
        string sourceData = _configuration["SourceData"]!;
        string dbName = _configuration["DbName"]!;
        string cron = _configuration["Cron"]!;
        string scriptUrl = _configuration["ScriptURL"]!;
        string jobName = _configuration["JobName"]!;

        var wrapper = host.Services.GetRequiredService<GlueWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UiWrapper>();

        uiWrapper.DisplayOverview();
        uiWrapper.PressEnter();

        // Create the crawler and wait for it to be ready.
        uiWrapper.DisplayTitle("Create AWS Glue crawler");
        Console.WriteLine("Let's begin by creating the AWS Glue crawler.");

        var crawlerDescription = "Crawler created for the AWS Glue Basics
scenario.";
        var crawlerCreated = await wrapper.CreateCrawlerAsync(crawlerName,
crawlerDescription, roleName, cron, sourceData, dbName);
        if (crawlerCreated)
        {
            Console.WriteLine($"The crawler: {crawlerName} has been created. Now
let's wait until it's ready.");
```

```
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't create crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.DisplayTitle("Start AWS Glue crawler");
        Console.WriteLine("Now let's wait until the crawler has successfully
started.");
        var crawlerStarted = await wrapper.StartCrawlerAsync(crawlerName);
        if (crawlerStarted)
        {
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't start the crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.PressEnter();

        Console.WriteLine($"\nLet's take a look at the database: {dbName}");
        var database = await wrapper.GetDatabaseAsync(dbName);

        if (database != null)
        {
            uiWrapper.DisplayTitle($"{database.Name} Details");
```

```
            Console.WriteLine($"{database.Name} created on
{database.CreateTime}");
            Console.WriteLine(database.Description);
        }

        uiWrapper.PressEnter();

        var tables = await wrapper.GetTablesAsync(dbName);
        if (tables.Count > 0)
        {
            tables.ForEach(table =>
            {
                Console.WriteLine($"{table.Name}\tCreated:
{table.CreateTime}\tUpdated: {table.UpdateTime}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create AWS Glue job");
        Console.WriteLine("Creating a new AWS Glue job.");
        var description = "An AWS Glue job created using the AWS SDK for .NET";
        await wrapper.CreateJobAsync(dbName, tables[0].Name, bucketUrl, jobName,
roleName, description, scriptUrl);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Starting AWS Glue job");
        Console.WriteLine("Starting the new AWS Glue job...");
        var jobRunId = await wrapper.StartJobRunAsync(jobName, dbName,
tables[0].Name, bucketName);
        var jobRunComplete = false;
        var jobRun = new JobRun();
        do
        {
            jobRun = await wrapper.GetJobRunAsync(jobName, jobRunId);
            if (jobRun.JobRunState == "SUCCEEDED" || jobRun.JobRunState ==
"STOPPED" ||
                jobRun.JobRunState == "FAILED" || jobRun.JobRunState ==
"TIMEOUT")
            {
                jobRunComplete = true;
            }
        } while (!jobRunComplete);
```

```
        uiWrapper.DisplayTitle($"Data in {bucketName}");

        // Get the list of data stored in the S3 bucket.
        var s3Client = new AmazonS3Client();

        var response = await s3Client.ListObjectsAsync(new ListObjectsRequest
{ BucketName = bucketName });
        response.S3Objects.ForEach(s3Object =>
        {
            Console.WriteLine(s3Object.Key);
        });

        uiWrapper.DisplayTitle("AWS Glue jobs");
        var jobNames = await wrapper.ListJobsAsync();
        jobNames.ForEach(jobName =>
        {
            Console.WriteLine(jobName);
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Get AWS Glue job run information");
        Console.WriteLine("Getting information about the AWS Glue job.");
        var jobRuns = await wrapper.GetJobRunsAsync(jobName);

        jobRuns.ForEach(jobRun =>
        {

Console.WriteLine($"{jobRun.JobName}\t{jobRun.JobRunState}\t{jobRun.CompletedOn}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Deleting resources");
        Console.WriteLine("Deleting the AWS Glue job used by the example.");
        await wrapper.DeleteJobAsync(jobName);

        Console.WriteLine("Deleting the tables from the database.");
        tables.ForEach(async table =>
        {
            await wrapper.DeleteTableAsync(dbName, table.Name);
        });
```

```
        Console.WriteLine("Deleting the database.");
        await wrapper.DeleteDatabaseAsync(dbName);

        Console.WriteLine("Deleting the AWS Glue crawler.");
        await wrapper.DeleteCrawlerAsync(crawlerName);

        Console.WriteLine("The AWS Glue scenario has completed.");
        uiWrapper.PressEnter();
    }
}


namespace GlueBasics;

public class UiWrapper
{
    public readonly string SepBar = new string('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Amazon Glue: get started with crawlers and jobs");

        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t 1. Create a crawler, pass it the IAM role and the
 URL to the public S3 bucket that contains the source data");
        Console.WriteLine("\t 2. Start the crawler.");
        Console.WriteLine("\t 3. Get the database created by the crawler and the
 tables in the database.");
        Console.WriteLine("\t 4. Create a job.");
        Console.WriteLine("\t 5. Start a job run.");
        Console.WriteLine("\t 6. Wait for the job run to complete.");
        Console.WriteLine("\t 7. Show the data stored in the bucket.");
        Console.WriteLine("\t 8. List jobs for the account.");
        Console.WriteLine("\t 9. Get job run details for the job that was run.");
        Console.WriteLine("\t10. Delete the demo job.");
        Console.WriteLine("\t11. Delete the database and tables created for the
 demo.");
        Console.WriteLine("\t12. Delete the crawler.");
    }
```

```
    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPlease press <Enter> to continue. ");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to center on the screen.</param>
    /// <returns>The string padded to make it center on the screen.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase

- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Scenario which demonstrates using AWS Glue to add a crawler and run a job.
/*!
 \\sa runGettingStartedWithGlueScenario()
 \param bucketName: An S3 bucket created in the setup.
 \param roleName: An AWS Identity and Access Management (IAM) role created in the
 setup.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String
 &bucketName,
                                                      const Aws::String &roleName,
```

```
                                                        const
Aws::Client::ClientConfiguration &clientConfig) {
   Aws::Glue::GlueClient client(clientConfig);

   Aws::String roleArn;
   if (!getRoleArn(roleName, roleArn, clientConfig)) {
       std::cerr << "Error getting role ARN for role." << std::endl;
       return false;
   }

   // 1. Upload the job script to the S3 bucket.
   {
       std::cout << "Uploading the job script '"
                 << AwsDoc::Glue::PYTHON_SCRIPT
                 << "'." << std::endl;

       if (!AwsDoc::Glue::uploadFile(bucketName,
                                     AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                     AwsDoc::Glue::PYTHON_SCRIPT,
                                     clientConfig)) {
           std::cerr << "Error uploading the job file." << std::endl;
           return false;
       }
   }

   // 2. Create a crawler.
   {
       Aws::Glue::Model::S3Target s3Target;
       s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
       Aws::Glue::Model::CrawlerTargets crawlerTargets;
       crawlerTargets.AddS3Targets(s3Target);

       Aws::Glue::Model::CreateCrawlerRequest request;
       request.SetTargets(crawlerTargets);
       request.SetName(CRAWLER_NAME);
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
       request.SetRole(roleArn);

       Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully created the crawler." << std::endl;
```

```
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
            return false;
        }
    }

    // 3. Get a crawler.
    {
        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 4. Start a crawler.
    {
        Aws::Glue::Model::StartCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);
```

```
        if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                    outcome.GetError().GetErrorType())) {
            if (!outcome.IsSuccess()) {
                std::cout << "Crawler was already started." << std::endl;
            }
            else {
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                        crawlerState)
                                << ". After " << iterations
                                << " seconds elapsed."
                                << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                                << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
```

```
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 5. Get a database.
    {
        Aws::Glue::Model::GetDatabaseRequest request;
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }
```

```
    // 6. Get tables.
    Aws::String tableName;
    {
        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        std::vector<Aws::Glue::Model::Table> all_tables;
        Aws::String nextToken; // Used for pagination.
        do {
            Aws::Glue::Model::GetTablesOutcome outcome =
client.GetTables(request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
                all_tables.insert(all_tables.end(), tables.begin(),
tables.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting the tables. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " <<
all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex -
1].Jsonize().View().WriteReadable()
```

```
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
    }

    // 7. Create a job.
    {
        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 8. Start a job run.
    {
        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
```

```
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
```

```
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                        jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                         clientConfig);
            return false;
        }
    }

    // 9. List the output data stored in the S3 bucket.
    {
        Aws::S3::S3Client s3Client;
        Aws::S3::Model::ListObjectsV2Request request;
        request.SetBucket(bucketName);
        request.SetPrefix(OUTPUT_FILE_PREFIX);
```

```
        Aws::String continuationToken; // Used for pagination.
        std::vector<Aws::S3::Model::Object> allObjects;
        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome =
s3Client.ListObjectsV2(
                    request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::S3::Model::Object> &objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
                continuationToken =
outcome.GetResult().GetNextContinuationToken();
            }
            else {
                std::cerr << "Error listing objects. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!continuationToken.empty());

        std::cout << "Data from your job is in " << allObjects.size() <<
                  " files in the S3 bucket, " << bucketName << "." << std::endl;

        for (size_t i = 0; i < allObjects.size(); ++i) {
            std::cout << "    " << i + 1 << ". " << allObjects[i].GetKey()
                      << std::endl;
        }

        int objectIndex = askQuestionForIntRange(
                std::string(
                        "Enter the number of a block to download it and see the
first ") +
                std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
                " lines of JSON output in the block: ", 1,
                static_cast<int>(allObjects.size()));

        Aws::String objectKey = allObjects[objectIndex - 1].GetKey();
```

```
        std::stringstream stringStream;
        if (getObjectFromBucket(bucketName, objectKey, stringStream,
                                clientConfig)) {
            for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream; +
+i) {
                std::string line;
                std::getline(stringStream, line);
                std::cout << "    " << line << std::endl;
            }
        }
        else {
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                         clientConfig);
            return false;
        }
    }

    // 10. List all the jobs.
    Aws::String jobName;
    {
        Aws::Glue::Model::ListJobsRequest listJobsRequest;

        Aws::String nextToken;
        std::vector<Aws::String> allJobNames;

        do {
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                allJobNames.insert(allJobNames.end(), jobNames.begin(),
 jobNames.end());
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
```

```
                          << std::endl;
            }
        } while (!nextToken.empty());
        std::cout << "Your account has " << allJobNames.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < allJobNames.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobNames[i] << std::endl;
        }
        int jobIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobNames.size()) +
                " to see the list of runs for a job: ",
                1, static_cast<int>(allJobNames.size()));

        jobName = allJobNames[jobIndex - 1];
    }

    // 11. Get the job runs for a job.
    Aws::String jobRunID;
    if (!jobName.empty()) {
        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome =
client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(),
jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
```

```
                        << std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "There are " << allJobRuns.size() << " runs in the job '"
                  <<
                  jobName << "'." << std::endl;

        for (size_t i = 0; i < allJobRuns.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobRuns[i].GetJobName()
                      << std::endl;
        }

        int runIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobRuns.size()) +
                " to see details for a run: ",
                1, static_cast<int>(allJobRuns.size()));
        jobRunID = allJobRuns[runIndex - 1].GetId();
    }

    // 12. Get a single job run.
    if (!jobRunID.empty()) {
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
    }
```

```
    return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                        clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
 \\sa deleteAssets()
 \param crawler: Name of an AWS Glue crawler.
 \param database: The name of an AWS Glue database.
 \param job: The name of an AWS Glue job.
 \param bucketName: The name of an S3 bucket.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
 &database,
                                const Aws::String &job, const Aws::String
 &bucketName,
                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::Glue::GlueClient client(clientConfig);
    bool result = true;

    // 13. Delete a job.
    if (!job.empty()) {
        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }
```

```
    // 14. Delete a database.
    if (!database.empty()) {
        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    // 15. Delete a crawler.
    if (!crawler.empty()) {
        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
 client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    // 16. Delete the job script and run data from the S3 bucket.
    result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
                                                       clientConfig);
    return result;
}

//! Routine which uploads a file to an S3 bucket.
```

```
/*!
 \\sa uploadFile()
 \param bucketName: An S3 bucket created in the setup.
 \param filePath: The path of the file to upload.
 \param fileName The name for the uploaded file.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                         const Aws::String &filePath,
                         const Aws::String &fileName,
                         const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          filePath.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << filePath << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Added object '" << filePath << "' to bucket '"
                  << bucketName << "'." << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
 \\sa deleteAllObjectsInS3Bucket()
 \param bucketName: The S3 bucket name.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                              const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsV2Request listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    bool result = true;
    do {
        if (!continuationToken.empty()) {
            listObjectsRequest.SetContinuationToken(continuationToken);
        }

        Aws::S3::Model::ListObjectsV2Outcome listObjectsOutcome =
 client.ListObjectsV2(
                listObjectsRequest);

        if (listObjectsOutcome.IsSuccess()) {
            const std::vector<Aws::S3::Model::Object> &objects =
 listObjectsOutcome.GetResult().GetContents();
            if (!objects.empty()) {
                Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
                deleteObjectsRequest.SetBucket(bucketName);

                std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
                for (const Aws::S3::Model::Object &object: objects) {
                    objectIdentifiers.push_back(
                            Aws::S3::Model::ObjectIdentifier().WithKey(
                                    object.GetKey()));
                }
                Aws::S3::Model::Delete objectsDelete;
                objectsDelete.SetObjects(objectIdentifiers);
                objectsDelete.SetQuiet(true);
```

```
                deleteObjectsRequest.SetDelete(objectsDelete);

                Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                        client.DeleteObjects(deleteObjectsRequest);

                if (!deleteObjectsOutcome.IsSuccess()) {
                    std::cerr << "Error deleting objects. " <<
                              deleteObjectsOutcome.GetError().GetMessage() <<
 std::endl;
                    result = false;
                    break;
                }
                else {
                    std::cout << "Successfully deleted the objects." <<
 std::endl;

                }
            }
            else {
                std::cout << "No objects to delete in '" << bucketName << "'."
                          << std::endl;
            }

            continuationToken =
 listObjectsOutcome.GetResult().GetNextContinuationToken();
        }
        else {
            std::cerr << "Error listing objects. "
                      << listObjectsOutcome.GetError().GetMessage() << std::endl;
            result = false;
            break;
        }
    } while (!continuationToken.empty());

    return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
 \\sa getObjectFromBucket()
 \param bucketName: The S3 bucket name.
 \param objectKey: The object's name.
 \param objectStream: A stream to receive the retrieved data.
 \param clientConfig: AWS client configuration.
```

```
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);


    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." <<
 std::endl;
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
    }
    else {
        std::cerr << "Error retrieving object. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable

- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

```
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

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

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri> <bucketNameSc>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3
 permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
 that contains data (for example, s3://<bucket name>/read).
                cron - A cron expression used to specify the schedule  (i.e.,
 cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
                jobName - The name you assign to this job definition.
                scriptLocation - The Amazon S3 path to a script that runs a job.
                locationUri - The location of the database (you can find this
 file in resources folder).
                bucketNameSc - The Amazon S3 bucket name used when creating a job
                """;

        if (args.length != 9) {
            System.out.println(usage);
```

```
            return;
        }
        Scanner scanner = new Scanner(System.in);
        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println("""
            AWS Glue is a fully managed extract, transform, and load (ETL)
service provided by Amazon
            Web Services (AWS). It is designed to simplify the process of
building, running, and maintaining
            ETL pipelines, which are essential for data integration and data
warehousing tasks.

            One of the key features of AWS Glue is its ability to automatically
discover and catalog data
            stored in various sources, such as Amazon S3, Amazon RDS, Amazon
Redshift, and other databases.
            This cataloging process creates a central metadata repository, known
as the AWS Glue Data Catalog,
            which provides a unified view of an organization's data assets. This
metadata can then be used to
            create ETL jobs, which can be scheduled and run on-demand or on a
regular basis.

            Lets get started.

            """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        try {
            createDatabase(glueClient, dbName, locationUri);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().equals("Database already
exists.")) {
                System.out.println("Database " + dbName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                return;
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        try {
            createGlueCrawler(glueClient, iam, s3Path, cron, dbName,
crawlerName);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().contains("already exists")) {
                System.out.println("Crawler " + crawlerName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        try {
            getSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        try {
            startSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        try {
            getSpecificDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName;
        try {
            myTableName = getGlueTables(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        try {
            createJob(glueClient, jobName, iam, scriptLocation);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        try {
            startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        try {
            getAllJobs(glueClient);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        try {
            getJobRuns(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        try {
            deleteJob(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            return;
        }
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        try {
            deleteDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        try {
            deleteSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }


    /**
     * Creates a Glue database with the specified name and location URI.
     *
     * @param glueClient  The Glue client to use for the database creation.
     * @param dbName      The name of the database to create.
     * @param locationUri The location URI for the database.
     */
    public static void createDatabase(GlueClient glueClient, String dbName,
 String locationUri) {
```

```
        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
                .locationUri(locationUri)
                .build();

            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                .databaseInput(input)
                .build();

            glueClient.createDatabase(request);
            System.out.println(dbName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue crawler using the AWS Glue Java API.
     *
     * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
     * @param iam         the IAM role that the crawler will use to access the
data source
     * @param s3Path      the S3 path that the crawler will scan for data
     * @param cron        the cron expression that defines the crawler's schedule
     * @param dbName      the name of the AWS Glue database where the crawler
will store the metadata
     * @param crawlerName the name of the crawler to be created
     */
    public static void createGlueCrawler(GlueClient glueClient,
                                         String iam,
                                         String s3Path,
                                         String cron,
                                         String dbName,
                                         String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();
```

```
            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to
be in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue
service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) throws InterruptedException {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response =
glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
```

```
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }

    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact
 with the service
     * @param databaseName the name of the database to retrieve
     * @throws GlueException if there is an error retrieving the database from
 the AWS Glue service
     */
```

```
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " +
createDate);

       } catch (GlueException e) {
           throw e;
       }
   }


   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table
names from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
               .databaseName(dbName)
               .build();

           GetTablesResponse response = glueClient.getTables(tableRequest);
```

```
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }


    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    public static void startJob(GlueClient glueClient, String jobName, String
 inputDatabase, String inputTable,
                                String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
                .jobName(jobName)
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
```

```
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    public static void createJob(GlueClient glueClient, String jobName, String
iam, String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            throw e;
        }
    }
```

```
    /**
     * Retrieves and prints information about all the jobs in the Glue data
catalog.
     *
     * @param glueClient the Glue client used to interact with the AWS Glue
service
     */
    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                .maxResults(10)
                .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the job runs for a given Glue job and prints the status of the
job runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
```

```
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
```

```
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact
with the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName)
{
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
```

```
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
 DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateCrawler
  - CreateJob

- DeleteCrawler
- DeleteDatabase
- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie e execute um rastreador que rastreia um bucket público do Amazon Simple Storage Service (Amazon S3) e gera um banco de dados de metadados que descreve os dados encontrados. CSV-formatted

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
```

```
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const crawlerExists = async ({ getCrawler }, crawlerName) => {
  try {
    await getCrawler(crawlerName);
    return true;
  } catch {
    return false;
  }
};

/**
 * @param {{ createCrawler: import('../../../actions/create-
crawler.js').createCrawler}} actions
 */
const makeCreateCrawlerStep = (actions) => async (context) => {
  if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
```

```
    log("Crawler already exists. Skipping creation.");
  } else {
    await actions.createCrawler(
      process.env.CRAWLER_NAME,
      process.env.ROLE_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_PREFIX,
      process.env.S3_TARGET_PATH,
    );

    log("Crawler created successfully.", { type: "success" });
  }

  return { ...context };
};

/**
 * @param {(name: string) => Promise<import('@aws-sdk/client-
glue').GetCrawlerCommandOutput>} getCrawler
 * @param {string} crawlerName
 */
const waitForCrawler = async (getCrawler, crawlerName) => {
  const waitTimeInSeconds = 30;
  const { Crawler } = await getCrawler(crawlerName);

  if (!Crawler) {
    throw new Error(`Crawler with name ${crawlerName} not found.`);
  }

  if (Crawler.State === "READY") {
    return;
  }

  log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
  await wait(waitTimeInSeconds);
  return waitForCrawler(getCrawler, crawlerName);
};

const makeStartCrawlerStep =
  ({ startCrawler, getCrawler }) =>
  async (context) => {
    log("Starting crawler.");
    await startCrawler(process.env.CRAWLER_NAME);
    log("Crawler started.", { type: "success" });
```

```
    log("Waiting for crawler to finish running. This can take a while.");
    await waitForCrawler(getCrawler, process.env.CRAWLER_NAME);
    log("Crawler ready.", { type: "success" });

    return { ...context };
  };
```

Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};

const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
  };

/**
```

```
 * @param {{ getTables: () => Promise<import('@aws-sdk/client-
glue').GetTablesCommandOutput}} config
 */
const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
  };
```

Crie e execute um trabalho que extraia dados CSV do bucket de origem do Amazon S3, transforme-os removendo e renomeando campos e carregue a saída JSON-formatted em outro bucket do Amazon S3.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
```

```
    },
  });

  return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY,
    );
    log("Job created.", { type: "success" });

    return { ...context };
  };

/**
 * @param {(name: string, runId: string) => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput> }  getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "TIMEOUT":
    case "STOPPED":
    case "ERROR":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`,
      );
    case "SUCCEEDED":
```

```
      return;
    default:
      break;
  }

  log(
    `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...`,
  );
  await wait(waitTimeInSeconds);
  return waitForJobRun(getJobRun, jobName, jobRunId);
};

/**
 * @param {{ prompter: { prompt: () => Promise<{ shouldOpen: boolean }>} }}
 context
 */
const promptToOpen = async (context) => {
  const { shouldOpen } = await context.prompter.prompt({
    name: "shouldOpen",
    type: "confirm",
    message: "Open the output bucket in your browser?",
  });

  if (shouldOpen) {
    return open(
      `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME} to
 view the output.`,
    );
  }
};

const makeStartJobRunStep =
  ({ startJobRun, getJobRun }) =>
  async (context) => {
    log("Starting job.");
    const { JobRunId } = await startJobRun(
      process.env.JOB_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_NAME,
      process.env.BUCKET_NAME,
    );
    log("Job started.", { type: "success" });

    log("Waiting for job to finish running. This can take a while.");
```

```
    await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
    log("Job run succeeded.", { type: "success" });

    await promptToOpen(context);

    return { ...context };
  };
```

Liste informações sobre execuções de tarefas e visualize alguns dos dados transformados.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};

/**
 * @typedef {{ prompter: { prompt: () => Promise<{jobName: string}> } }} Context
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput>} getJobRun
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-
glue').GetJobRunsCommandOutput} getJobRuns
 */
```

```
/**
 *
 * @param {getJobRun} getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
  const { JobRun } = await getJobRun(jobName, jobRunId);
  log(JobRun, { type: "object" });
};

/**
 *
 * @param {{getJobRuns: getJobRuns, getJobRun: getJobRun }} funcs
 */
const makePickJobRunStep =
  ({ getJobRuns, getJobRun }) =>
  async (/** @type { Context } */ context) => {
    if (context.selectedJobName) {
      const { JobRuns } = await getJobRuns(context.selectedJobName);

      const { jobRunId } = await context.prompter.prompt({
        name: "jobRunId",
        type: "list",
        message: "Select a job run to see details.",
        choices: JobRuns.map((run) => run.Id),
      });

      logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
    }

    return { ...context };
  };
```

Exclua todos os recursos criados pela demonstração.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
```

```
  });

  return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};

const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};

const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};

/**
 *
 * @param {import('../../../actions/delete-job.js').deleteJob} deleteJobFn
 * @param {string[]} jobNames
 * @param {{ prompter: { prompt: () => Promise<any> }}} context
 */
const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
  /**
```

```
    * @type {{ selectedJobNames: string[] }}
    */
  const { selectedJobNames } = await context.prompter.prompt({
    name: "selectedJobNames",
    type: "checkbox",
    message: "Let's clean up jobs. Select jobs to delete.",
    choices: jobNames,
  });

  if (selectedJobNames.length === 0) {
    log("No jobs selected.");
  } else {
    log("Deleting jobs.");
    await Promise.all(
      selectedJobNames.map((n) => deleteJobFn(n).catch(console.error)),
    );
    log("Jobs deleted.", { type: "success" });
  }
};

/**
 * @param {{
 *   listJobs: import('../../../actions/list-jobs.js').listJobs,
 *   deleteJob: import('../../../actions/delete-job.js').deleteJob
 * }} config
 */
const makeCleanUpJobsStep =
  ({ listJobs, deleteJob }) =>
  async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
      await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
  };

/**
 * @param {import('../../../actions/delete-table.js').deleteTable} deleteTable
 * @param {string} databaseName
 * @param {string[]} tableNames
 */
const deleteTables = (deleteTable, databaseName, tableNames) =>
  Promise.all(
```

```
    tableNames.map((tableName) =>
      deleteTable(databaseName, tableName).catch(console.error),
    ),
  );

/**
 * @param {{
 *  getTables: import('../../../actions/get-tables.js').getTables,
 *  deleteTable: import('../../../actions/delete-table.js').deleteTable
 * }} config
 */
const makeCleanUpTablesStep =
  ({ getTables, deleteTable }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}}} context
   */
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
      () => ({ TableList: null }),
    );

    if (TableList && TableList.length > 0) {
      /**
       * @type {{ tableNames: string[] }}
       */
      const { tableNames } = await context.prompter.prompt({
        name: "tableNames",
        type: "checkbox",
        message: "Let's clean up tables. Select tables to delete.",
        choices: TableList.map((t) => t.Name),
      });

      if (tableNames.length === 0) {
        log("No tables selected.");
      } else {
        log("Deleting tables.");
        await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
        log("Tables deleted.", { type: "success" });
      }
    }

    return { ...context };
  };
```

```
/**
 * @param {import('../../../actions/delete-database.js').deleteDatabase}
 deleteDatabase
 * @param {string[]} databaseNames
 */
const deleteDatabases = (deleteDatabase, databaseNames) =>
  Promise.all(
    databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error)),
  );

/**
 * @param {{
 *   getDatabases: import('../../../actions/get-databases.js').getDatabases
 *   deleteDatabase: import('../../../actions/delete-database.js').deleteDatabase
 * }} config
 */
const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}} context
   */
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      /** @type {{ dbNames: string[] }} */
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
        message: "Let's clean up databases. Select databases to delete.",
        choices: DatabaseList.map((db) => db.Name),
      });

      if (dbNames.length === 0) {
        log("No databases selected.");
      } else {
        log("Deleting databases.");
        await deleteDatabases(deleteDatabase, dbNames);
        log("Databases deleted.", { type: "success" });
      }
    }

    return { ...context };
  };
```

```
const cleanUpCrawlerStep = async (context) => {
  log("Deleting crawler.");

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log("Crawler is already deleted.");
    } else {
      throw err;
    }
  }

  return { ...context };
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler

- StartJobRun

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service
 (Amazon S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
```

```
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
```

```
    }
}

suspend fun createCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
```

```
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
```

```
    }
}

suspend fun startJob(jobNameVal: String?) {
    val runRequest =
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
```

```
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
            maxResults = 10
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
```

```
        DeleteDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables

- ListJobs
- StartCrawler
- StartJobRun

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
```

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);
```

```
        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }
```

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;
```

```
class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path):
 Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
 $databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

```
    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
 '3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables,
 $outputBucketUrl): Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
```

```
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
```

```
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase

- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as AWS Glue funções usadas no cenário.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
```

```
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created
by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
```

```
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
that can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
```

```
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_job_run(self, name, input_database, input_table,
output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
the
```

```
            # Python ETL script to determine the location of input and output
data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]


    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
```

```
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]


    def delete_job(self, job_name):
        """
```

```
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Crie uma classe que execute o cenário.

```
class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job
 and use
    them to transform data from CSV to JSON format.
    """

    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
```

```
        :param glue_service_role: An AWS Identity and Access Management (IAM)
role
                                  that AWS Glue can assume to gain access to the
                                  resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output
data
                            from AWS Glue job runs.
        """
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
    def wait(seconds, tick=12):
        """
        Waits for a specified number of seconds, while also displaying an
animated
        spinner.

        :param seconds: The number of seconds to wait.
        :param tick: The number of frames per second used to animate the spinner.
        """
        progress = "|/-\\"
        waited = 0
        while waited < seconds:
            for frame in range(tick):
                sys.stdout.write(f"\r{progress[frame % len(progress)]}")
                sys.stdout.flush()
                time.sleep(1 / tick)
            waited += 1

    def upload_job_script(self, job_script):
        """
        Uploads a Python ETL script to an S3 bucket. The script is used by the
AWS Glue
        job to transform data.

        :param job_script: The relative path to the job script.
        """
        try:
            self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
            print(f"Uploaded job script '{job_script}' to the example bucket.")
        except S3UploadFailedError as err:
            logger.error("Couldn't upload job script. Here's why: %s", err)
```

```
            raise

    def run(self, crawler_name, db_name, db_prefix, data_source, job_script,
job_name):
        """
        Runs the scenario. This is an interactive experience that runs at a
command
        prompt and asks you for input throughout.

        :param crawler_name: The name of the crawler used in the scenario. If the
                             crawler does not exist, it is created.
        :param db_name: The name to give the metadata database created by the
crawler.
        :param db_prefix: The prefix to give tables added to the database by the
                          crawler.
        :param data_source: The location of the data source that is targeted by
the
                            crawler and extracted during job runs.
        :param job_script: The job script that is used to transform data during
job
                           runs.
        :param job_name: The name to give the job definition that is created
during the
                         scenario.
        """
        wrapper = GlueWrapper(self.glue_client)
        print(f"Checking for crawler {crawler_name}.")
        crawler = wrapper.get_crawler(crawler_name)
        if crawler is None:
            print(f"Creating crawler {crawler_name}.")
            wrapper.create_crawler(
                crawler_name,
                self.glue_service_role.arn,
                db_name,
                db_prefix,
                data_source,
            )
            print(f"Created crawler {crawler_name}.")
            crawler = wrapper.get_crawler(crawler_name)
        pprint(crawler)
        print("-" * 88)

        print(
```

```
            f"When you run the crawler, it crawls data stored in {data_source}
and "
            f"creates a metadata database in the AWS Glue Data Catalog that
describes "
            f"the data in the data source."
        )
        print("In this example, the source data is in CSV format.")
        ready = False
        while not ready:
            ready = Question.ask_question(
                "Ready to start the crawler? (y/n) ", Question.is_yesno
            )
        wrapper.start_crawler(crawler_name)
        print("Let's wait for the crawler to run. This typically takes a few
minutes.")
        crawler_state = None
        while crawler_state != "READY":
            self.wait(10)
            crawler = wrapper.get_crawler(crawler_name)
            crawler_state = crawler["State"]
            print(f"Crawler is {crawler['State']}.")
        print("-" * 88)

        database = wrapper.get_database(db_name)
        print(f"The crawler created database {db_name}:")
        pprint(database)
        print(f"The database contains these tables:")
        tables = wrapper.get_tables(db_name)
        for index, table in enumerate(tables):
            print(f"\t{index + 1}. {table['Name']}")
        table_index = Question.ask_question(
            f"Enter the number of a table to see more detail: ",
            Question.is_int,
            Question.in_range(1, len(tables)),
        )
        pprint(tables[table_index - 1])
        print("-" * 88)

        print(f"Creating job definition {job_name}.")
        wrapper.create_job(
            job_name,
            "Getting started example job.",
            self.glue_service_role.arn,
            f"s3://{self.glue_bucket.name}/{job_script}",
```

```
        )
        print("Created job definition.")
        print(
            f"When you run the job, it extracts data from {data_source},
transforms it "
            f"by using the {job_script} script, and loads the output into "
            f"S3 bucket {self.glue_bucket.name}."
        )
        print(
            "In this example, the data is transformed from CSV to JSON, and only
a few "
            "fields are included in the output."
        )
        job_run_status = None
        if Question.ask_question(f"Ready to run? (y/n) ", Question.is_yesno):
            job_run_id = wrapper.start_job_run(
                job_name, db_name, tables[0]["Name"], self.glue_bucket.name
            )
            print(f"Job {job_name} started. Let's wait for it to run.")
            while job_run_status not in ["SUCCEEDED", "STOPPED", "FAILED",
"TIMEOUT"]:
                self.wait(10)
                job_run = wrapper.get_job_run(job_name, job_run_id)
                job_run_status = job_run["JobRunState"]
                print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
        print("-" * 88)

        if job_run_status == "SUCCEEDED":
            print(
                f"Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':"
            )
            try:
                keys = [
                    obj.key for obj in
self.glue_bucket.objects.filter(Prefix="run-")
                ]
                for index, key in enumerate(keys):
                    print(f"\t{index + 1}: {key}")
                lines = 4
                key_index = Question.ask_question(
                    f"Enter the number of a block to download it and see the
first {lines} "
                    f"lines of JSON output in the block: ",
```

```
                    Question.is_int,
                    Question.in_range(1, len(keys)),
                )
                job_data = io.BytesIO()
                self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)
                job_data.seek(0)
                for _ in range(lines):
                    print(job_data.readline().decode("utf-8"))
            except ClientError as err:
                logger.error(
                    "Couldn't get job run data. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
            print("-" * 88)

        job_names = wrapper.list_jobs()
        if job_names:
            print(f"Your account has {len(job_names)} jobs defined:")
            for index, job_name in enumerate(job_names):
                print(f"\t{index + 1}. {job_name}")
            job_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_names)} to see the list
of runs for "
                f"a job: ",
                Question.is_int,
                Question.in_range(1, len(job_names)),
            )
            job_runs = wrapper.get_job_runs(job_names[job_index - 1])
            if job_runs:
                print(f"Found {len(job_runs)} runs for job {job_names[job_index -
1]}:")
                for index, job_run in enumerate(job_runs):
                    print(
                        f"\t{index + 1}. {job_run['JobRunState']} on "
                        f"{job_run['CompletedOn']:%Y-%m-%d %H:%M:%S}"
                    )
                run_index = Question.ask_question(
                    f"Enter a number between 1 and {len(job_runs)} to see details
for a run: ",
                    Question.is_int,
                    Question.in_range(1, len(job_runs)),
                )
```

```
                pprint(job_runs[run_index - 1])
            else:
                print(f"No runs found for job {job_names[job_index - 1]}")
        else:
            print("Your account doesn't have any jobs defined.")
        print("-" * 88)

        print(
            f"Let's clean up. During this example we created job definition
 '{job_name}'."
        )
        if Question.ask_question(
            "Do you want to delete the definition and all runs? (y/n) ",
            Question.is_yesno,
        ):
            wrapper.delete_job(job_name)
            print(f"Job definition '{job_name}' deleted.")
        tables = wrapper.get_tables(db_name)
        print(f"We also created database '{db_name}' that contains these
 tables:")
        for table in tables:
            print(f"\t{table['Name']}")
        if Question.ask_question(
            "Do you want to delete the tables and the database? (y/n) ",
            Question.is_yesno,
        ):
            for table in tables:
                wrapper.delete_table(db_name, table["Name"])
                print(f"Deleted table {table['Name']}.")
            wrapper.delete_database(db_name)
            print(f"Deleted database {db_name}.")
        print(f"We also created crawler '{crawler_name}'.")
        if Question.ask_question(
            "Do you want to delete the crawler? (y/n) ", Question.is_yesno
        ):
            wrapper.delete_crawler(crawler_name)
            print(f"Deleted crawler {crawler_name}.")
        print("-" * 88)


def parse_args(args):
    """
    Parse command line arguments.
```

```
    :param args: The command line arguments.
    :return: The parsed arguments.
    """
    parser = argparse.ArgumentParser(
        description="Runs the AWS Glue getting started with crawlers and jobs
 scenario. "
        "Before you run this scenario, set up scaffold resources by running "
        "'python scaffold.py deploy'."
    )
    parser.add_argument(
        "role_name",
        help="The name of an IAM role that AWS Glue can assume. This role must
 grant access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole "
        "managed policy.",
    )
    parser.add_argument(
        "bucket_name",
        help="The name of an S3 bucket that AWS Glue can access to get the job
 script and "
        "put job results.",
    )
    parser.add_argument(
        "--job_script",
        default="flight_etl_job_script.py",
        help="The name of the job script file that is used in the scenario.",
    )
    return parser.parse_args(args)


def main():
    args = parse_args(sys.argv[1:])
    try:
        print("-" * 88)
        print(
            "Welcome to the AWS Glue getting started with crawlers and jobs
 scenario."
        )
        print("-" * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client("glue"),
            boto3.resource("iam").Role(args.role_name),
            boto3.resource("s3").Bucket(args.bucket_name),
        )
```

```
        scenario.upload_job_script(args.job_script)
        scenario.run(
            "doc-example-crawler",
            "doc-example-database",
            "doc-example-",
            "s3://crawler-public-us-east-1/flight/2016/csv",
            args.job_script,
            "doc-example-job",
        )
        print("-" * 88)
        print(
            "To destroy scaffold resources, including the IAM role and S3 bucket
 "
            "used in this scenario, run 'python scaffold.py destroy'."
        )
        print("\nThanks for watching!")
        print("-" * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")
```

Crie um script ETL que seja usado AWS Glue para extrair, transformar e carregar dados durante a execução do trabalho.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
```

```
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
```

```
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as AWS Glue funções usadas no cenário.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
 if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
```

```
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
```

```
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
```

```
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
 job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
```

```
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
```

```
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
 file to.
  # @return [void]
  def upload_job_script(file_path, bucket_resource)
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
                                          body: file,
                                          bucket: bucket_resource.name,
                                          key: file_path
                                        })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end
end
```

Crie uma classe que execute o cenário.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
  end

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
  end
```

```
  private

  def setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    new_step(1, 'Create a crawler')
    crawler = wrapper.get_crawler(crawler_name)
    unless crawler
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
 db_prefix, data_source)
      puts "Successfully created #{crawler_name}."
    end
    wrapper.start_crawler(crawler_name)
    monitor_crawler(wrapper, crawler_name)
  end

  def monitor_crawler(wrapper, crawler_name)
    new_step(2, 'Monitor Crawler')
    crawler_state = nil
    until crawler_state == 'READY'
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]['state']
      print "Crawler status: #{crawler_state}".yellow
    end
  end

  def query_database(wrapper, _crawler_name, db_name)
    new_step(3, 'Query the database.')
    wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    puts "Database contains tables: #{wrapper.get_tables(db_name).map { |t|
 t['name'] }}"
  end

  def create_and_run_job(wrapper, job_script, job_name, db_name)
    new_step(4, 'Create and run job.')
    wrapper.upload_job_script(job_script, @glue_bucket)
    wrapper.create_job(job_name, 'ETL Job', @glue_service_role.arn, "s3://
#{@glue_bucket.name}/#{job_script}")
    run_job(wrapper, job_name, db_name)
  end

  def run_job(wrapper, job_name, db_name)
```

```
    new_step(5, 'Run the job.')
    wrapper.start_job_run(job_name, db_name, wrapper.get_tables(db_name)[0]
['name'], @glue_bucket.name)
    job_run_status = nil
    until %w[SUCCEEDED FAILED STOPPED].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]['job_run_state']
      print "Job #{job_name} status: #{job_run_status}".yellow
    end
  end
end

def main
  banner('../../helpers/banner.txt')
  puts 'Starting AWS Glue demo...'

  # Load resource names from YAML.
  resource_names = YAML.load_file('resource_names.yaml')

  # Setup services and resources.
  iam_role = Aws::IAM::Resource.new(region: 'us-
east-1').role(resource_names['glue_service_role'])
  s3_bucket = Aws::S3::Resource.new(region: 'us-
east-1').bucket(resource_names['glue_bucket'])

  # Instantiate scenario and run.
  scenario = GlueCrawlerJobScenario.new(Aws::Glue::Client.new(region: 'us-
east-1'), iam_role, s3_bucket, @logger)
  random_suffix = rand(10**4)
  scenario.run("crawler-#{random_suffix}", "db-#{random_suffix}", "prefix-
#{random_suffix}-", 's3://data_source',
               'job_script.py', "job-#{random_suffix}")

  puts 'Demo complete.'
end
```

Crie um script ETL que seja usado AWS Glue para extrair, transformar e carregar dados durante a execução do trabalho.

```
import sys
from awsglue.transforms import *
```

```
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
```

```
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun

- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie e execute um rastreador que rastreia um bucket público do Amazon Simple Storage Service (Amazon S3) e gera um banco de dados de metadados que descreve os dados encontrados. CSV-formatted

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
```

```
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;

        let start_crawler =
 glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
 database".into()))?;

        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        let tables = tables.table_list();
```

Crie e execute um trabalho que extraia dados CSV do bucket de origem do Amazon S3, transforme-os removendo e renomeando campos e carregue a saída JSON-formatted em outro bucket do Amazon S3.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;

        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
```

```
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

Exclua todos os recursos criados pela demonstração.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }

        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.

- CreateCrawler
- CreateJob
- DeleteCrawler
- DeleteDatabase
- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription
```

```
let package = Package(
    name: "glue-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "glue-scenario",
            dependencies: [
                .product(name: "AWSGlue", package: "aws-sdk-swift"),
                .product(name: "AWSS3", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo de código Swift, `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to demonstrate
// creating and using crawlers and jobs using AWS Glue.
//
// 0. Upload the Python job script to Amazon S3 so it can be used when
```

```
//    calling `startJobRun()` later.
// 1. Create a crawler, pass it the IAM role and the URL of the public Amazon
//    S3 bucket that contains the source data:
//    s3://crawler-public-us-east-1/flight/2016/csv.
// 2. Start the crawler. This takes time, so after starting it, use a loop
//    that calls `getCrawler()` until the state is "READY".
// 3. Get the database created by the crawler, and the tables in the
//    database. Display them to the user.
// 4. Create a job. Pass it the IAM role and the URL to a Python ETL script
//    previously uploaded to the user's S3 bucket.
// 5. Start a job run, passing the following custom arguments. These are
//    expected by the ETL script, so must exactly match.
//    * `--input_database: <name of the database created by the crawler>`
//    * `--input_table: <name of the table created by the crawler>`
//    * `--output_bucket_url: <URL to the scaffold bucket created for the
//      user>`
// 6. Loop and get the job run until it returns one of the following states:
//    "SUCCEEDED", "STOPPED", "FAILED", or "TIMEOUT".
// 7. Output data is stored in a group of files in the user's S3 bucket.
//    Either direct the user to their location or download a file and display
//    the results inline.
// 8. List the jobs for the user's account.
// 9. Get job run details for a job run.
// 10. Delete the demo job.
// 11. Delete the database and tables created by the example.
// 12. Delete the crawler created by the example.

import ArgumentParser
import AWSS3
import Foundation
import Smithy

import AWSClientRuntime
import AWSGlue

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS IAM role to use for AWS Glue calls.")
    var role: String

    @Option(help: "The Amazon S3 bucket to use for this example.")
    var bucket: String

    @Option(help: "The Amazon S3 URL of the data to crawl.")
    var s3url: String = "s3://crawler-public-us-east-1/flight/2016/csv"
```

```
    @Option(help: "The Python script to run as a job with AWS Glue.")
    var script: String = "./flight_etl_job_script.py"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(help: "A prefix string to use when naming tables.")
    var tablePrefix = "swift-glue-basics-table"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "glue-scenario",
        abstract: """
        Demonstrates various features of AWS Glue.
        """,
        discussion: """
        An example showing how to use AWS Glue to create, run, and monitor
        crawlers and jobs.
        """
    )

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
```

```
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Upload a file to an Amazon S3 bucket.
    ///
    /// - Parameters:
    ///   - s3Client: The S3 client to use when uploading the file.
    ///   - path: The local path of the source file to upload.
    ///   - toBucket: The name of the S3 bucket into which to upload the file.
    ///   - key: The key (name) to give the file in the S3 bucket.
    ///
    /// - Returns: `true` if the file is uploaded successfully, otherwise
`false`.
    func uploadFile(s3Client: S3Client, path: String, toBucket: String, key:
String) async -> Bool {
        do {
            let fileData: Data = try Data(contentsOf: URL(fileURLWithPath: path))
            let dataStream = ByteStream.data(fileData)
            _ = try await s3Client.putObject(
                input: PutObjectInput(
                    body: dataStream,
                    bucket: toBucket,
                    key: key
                )
            )
        } catch {
            print("*** An unexpected error occurred uploading the script to the
Amazon S3 bucket \"\(bucket)\".")
            return false
        }

        return true
    }

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
```

```
   ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
   ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
   ///   - databaseName: The name of an AWS Glue database to operate on.
   ///
   /// - Returns: `true` if the crawler is created successfully, otherwise
`false`.
   func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole:
String,
                      s3Path: String, cronSchedule: String, databaseName:
String) async -> Bool {
       let s3Target = GlueClientTypes.S3Target(path: s3url)
       let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])

       do {
           _ = try await glueClient.createCrawler(
               input: CreateCrawlerInput(
                   databaseName: databaseName,
                   description: "Created by the AWS SDK for Swift Scenario
Example for AWS Glue.",
                   name: crawlerName,
                   role: iamRole,
                   schedule: cronSchedule,
                   tablePrefix: tablePrefix,
                   targets: targetList
               )
           )
       } catch _ as AlreadyExistsException {
           print("*** A crawler named \"\(crawlerName)\" already exists.")
           return false
       } catch _ as OperationTimeoutException {
           print("*** The attempt to create the AWS Glue crawler timed out.")
           return false
       } catch {
           print("*** An unexpected error occurred creating the AWS Glue
crawler: \(error.localizedDescription)")
           return false
       }

       return true
   }

   /// Delete an AWS Glue crawler.
   ///
   /// - Parameters:
```

```
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
        return true
    }

    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise
`false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
```

```
   func getCrawlerState(glueClient: GlueClient, name: String) async ->
GlueClientTypes.CrawlerState {
       do {
           let output = try await glueClient.getCrawler(
               input: GetCrawlerInput(name: name)
           )

           // If the crawler or its state is `nil`, report that the crawler
           // is stopping. This may not be what you want for your
           // application but it works for this one!

           guard let crawler = output.crawler else {
               return GlueClientTypes.CrawlerState.stopping
           }
           guard let state = crawler.state else {
               return GlueClientTypes.CrawlerState.stopping
           }
           return state
       } catch {
           return GlueClientTypes.CrawlerState.stopping
       }
   }

   /// Wait until the specified crawler is ready to run.
   ///
   /// - Parameters:
   ///   - glueClient: The AWS Glue client to use.
   ///   - name: The name of the crawler to wait for.
   ///
   /// - Returns: `true` if the crawler is ready, `false` if the client is
   ///   stopping (and will therefore never be ready).
   func waitUntilCrawlerReady(glueClient: GlueClient, name: String) async ->
Bool {
       while true {
           let state = await getCrawlerState(glueClient: glueClient, name: name)

           if state == .ready {
               return true
           } else if state == .stopping {
               return false
           }

           // Wait four seconds before trying again.
```

```
            do {
                try await Task.sleep(for: .seconds(4))
            } catch {
                print("*** Error pausing the task.")
            }
        }
    }

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
                    description: "Created by the AWS SDK for Swift Glue basic
scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }
```

```
/// Return a list of the AWS Glue jobs listed on the user's account.
///
/// - Parameters:
///   - glueClient: The AWS Glue client to use.
///   - maxJobs: The maximum number of jobs to return (default: 100).
///
/// - Returns: An array of strings listing the names of all available AWS
///   Glue jobs.
func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
    var jobList: [String] = []
    var nextToken: String?

    repeat {
        do {
            let output = try await glueClient.listJobs(
                input: ListJobsInput(
                    maxResults: maxJobs,
                    nextToken: nextToken
                )
            )

            guard let jobs = output.jobNames else {
                return jobList
            }

            jobList = jobList + jobs
            nextToken = output.nextToken
        } catch {
            return jobList
        }
    } while (nextToken != nil)

    return jobList
}

/// Delete an AWS Glue job.
///
/// - Parameters:
///   - glueClient: The AWS Glue client to use.
///   - jobName: The name of the job to delete.
///
/// - Returns: `true` if the job is successfully deleted, otherwise `false`.
func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
    do {
```

```
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }

    /// Create an AWS Glue database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name to give the new database.
    ///   - location: The URL of the source data to use with AWS Glue.
    ///
    /// - Returns: `true` if the database is created successfully, otherwise
`false`.
    func createDatabase(glueClient: GlueClient, name databaseName: String,
location: String) async -> Bool {
        let databaseInput = GlueClientTypes.DatabaseInput(
            description: "Created by the AWS SDK for Swift Glue basic scenario
example.",
            locationUri: location,
            name: databaseName
        )

        do {
            _ = try await glueClient.createDatabase(
                input: CreateDatabaseInput(
                    databaseInput: databaseInput
                )
            )
        } catch {
            return false
        }

        return true
    }

    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
```

```
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
    ///   specified database, or `nil` if an error occurs or the database
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async
-> [GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
```

```
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }

    /// Delete the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient:
glueClient, databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables. If there's only one table, use
            // `deleteTable()`, otherwise, use `batchDeleteTable()`. You can
            // use `batchDeleteTable()` for a single table, but this
            // demonstrates the use of `deleteTable()`.
```

```
            if tableNames.count == 1 {
                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
            } else {
                do {
                    print("    Deleting tables...")
                    _ = try await glueClient.batchDeleteTable(
                        input: BatchDeleteTableInput(
                            databaseName: databaseName,
                            tablesToDelete: tableNames
                        )
                    )
                } catch {
                    print("*** Unable to delete the tables.")
                }
            }
        }

        // Delete the database itself.

        do {
            print("    Deleting the database itself...")
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
        } catch {
            print("*** Unable to delete the database.")
            return false
        }
        return true
    }

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
```

```
   ///   - glueClient: The AWS Glue client to use.
   ///   - jobName: The name of the job to run.
   ///   - databaseName: The name of the AWS Glue database to run the job
against.
   ///   - tableName: The name of the table in the database to run the job
against.
   ///   - outputURL: The AWS S3 URI of the bucket location into which to
   ///     write the resulting output.
   ///
   /// - Returns: `true` if the job run is started successfully, otherwise
`false`.
   func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
String,
                     tableName: String, outputURL: String) async -> String? {
       do {
           let output = try await glueClient.startJobRun(
               input: StartJobRunInput(
                   arguments: [
                       "--input_database": databaseName,
                       "--input_table": tableName,
                       "--output_bucket_url": outputURL
                   ],
                   jobName: jobName,
                   numberOfWorkers: 10,
                   workerType: .g1x
               )
           )

           guard let id = output.jobRunId else {
               return nil
           }

           return id
       } catch {
           return nil
       }
   }

   /// Return a list of the job runs for the specified job.
   ///
   /// - Parameters:
   ///   - glueClient: The AWS Glue client to use.
   ///   - jobName: The name of the job for which to return its job runs.
   ///   - maxResults: The maximum number of job runs to return (default:
```

```
    ///     1000).
    ///
    /// - Returns: An array of `GlueClientTypes.JobRun` objects describing
    ///   each job run.
    func getJobRuns(glueClient: GlueClient, name jobName: String, maxResults:
 Int? = nil) async -> [GlueClientTypes.JobRun] {
        do {
            let output = try await glueClient.getJobRuns(
                input: GetJobRunsInput(
                    jobName: jobName,
                    maxResults: maxResults
                )
            )

            guard let jobRuns = output.jobRuns else {
                print("*** No job runs found.")
                return []
            }

            return jobRuns
        } catch is EntityNotFoundException {
            print("*** The specified job name, \(jobName), doesn't exist.")
            return []
        } catch {
            print("*** Unexpected error getting job runs:")
            dump(error)
            return []
        }
    }

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String)
 async -> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
```

```
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // A name to give the Python script upon upload to the Amazon S3
        // bucket.
        let scriptName = "jobscript.py"

        // Schedule string in `cron` format, as described here:
        // https://docs.aws.amazon.com/glue/latest/dg/monitor-data-warehouse-
schedule.html
        let cron = "cron(15 12 * * ? *)"

        let glueConfig = try await GlueClient.GlueClientConfiguration(region:
 awsRegion)
        let glueClient = GlueClient(config: glueConfig)

        let s3Config = try await S3Client.S3ClientConfiguration(region:
 awsRegion)
        let s3Client = S3Client(config: s3Config)

        // Create random names for things that need them.

        let crawlerName = tempName(prefix: "swift-glue-basics-crawler")
        let databaseName = tempName(prefix: "swift-glue-basics-db")

        // Create a name for the AWS Glue job.

        let jobName = tempName(prefix: "scenario-job")

        // The URL of the Python script on S3.

        let scriptURL = "s3://\(bucket)/\(scriptName)"

        print("Welcome to the AWS SDK for Swift basic scenario for AWS Glue!")
```

```
        //=====================================================================
        // 0. Upload the Python script to the target bucket so it's available
        //    for use by the Amazon Glue service.
        //=====================================================================

        print("Uploading the Python script: \(script) as key \(scriptName)")
        print("Destination bucket: \(bucket)")
        if !(await uploadFile(s3Client: s3Client, path: script, toBucket: bucket,
key: scriptName)) {
            return
        }

        //=====================================================================
        // 1. Create the database and crawler using the randomized names
        //    generated previously.
        //=====================================================================

        print("Creating database \"\(databaseName)\"...")
        if !(await createDatabase(glueClient: glueClient, name: databaseName,
location: s3url)) {
            print("*** Unable to create the database.")
            return
        }

        print("Creating crawler \"\(crawlerName)\"...")
        if !(await createCrawler(glueClient: glueClient, crawlerName:
crawlerName,
                                 iamRole: role, s3Path: s3url, cronSchedule:
cron,
                                 databaseName: databaseName)) {
            return
        }

        //=====================================================================
        // 2. Start the crawler, then wait for it to be ready.
        //=====================================================================

        print("Starting the crawler and waiting until it's ready...")
        if !(await startCrawler(glueClient: glueClient, name: crawlerName)) {
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }
```

```
        if !(await waitUntilCrawlerReady(glueClient: glueClient, name:
crawlerName)) {
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
        }

        //=====================================================================
        // 3. Get the database and table created by the crawler.
        //=====================================================================

        print("Getting the crawler's database...")
        let database = await getDatabase(glueClient: glueClient, name:
databaseName)

        guard let database else {
            print("*** Unable to get the database.")
            return
        }
        print("Database URI: \(database.locationUri ?? "<unknown>")")

        let tableList = await getTablesInDatabase(glueClient: glueClient,
databaseName: databaseName)

        print("Found \(tableList.count) table(s):")
        for table in tableList {
            print("  \(table.name ?? "<unnamed>")")
        }

        if tableList.count != 1 {
            print("*** Incorrect number of tables found. There should only be
one.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        guard let tableName = tableList[0].name else {
            print("*** Table is unnamed.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }
```

```
        //=====================================================================
        // 4. Create a job.
        //=====================================================================

        print("Creating a job...")
        if !(await createJob(glueClient: glueClient, name: jobName, role: role,
                             scriptLocation: scriptURL)) {
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //=====================================================================
        // 5. Start a job run.
        //=====================================================================

        print("Starting the job...")

        // Construct the Amazon S3 URL for the job run's output. This is in
        // the bucket specified on the command line, with a folder name that's
        // unique for this job run.

        let timeStamp = Date().timeIntervalSince1970
        let jobPath = "\(jobName)-\(Int(timeStamp))"
        let outputURL = "s3://\(bucket)/\(jobPath)"

        // Start the job run.

        let jobRunID = await startJobRun(glueClient: glueClient, name: jobName,
                                         databaseName: databaseName,
                                         tableName: tableName,
                                         outputURL: outputURL)

        guard let jobRunID else {
            print("*** Job run ID is invalid.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //=====================================================================
```

```
        // 6. Wait for the job run to indicate that the run is complete.
        //===================================================================

        print("Waiting for job run to end...")

        var jobRunFinished = false
        var jobRunState: GlueClientTypes.JobRunState

        repeat {
            let jobRun = await getJobRun(glueClient: glueClient, name: jobName,
 id: jobRunID)
            guard let jobRun else {
                print("*** Unable to get the job run.")
                _ = await deleteJob(glueClient: glueClient, name: jobName)
                _ = await deleteDatabase(glueClient: glueClient, name:
 databaseName, withTables: true)
                _ = await deleteCrawler(glueClient: glueClient, name:
 crawlerName)
                return
            }
            jobRunState = jobRun.jobRunState ?? .failed

            //
===================================================================
            // 7. Output where to find the data if the job run was successful.
            //    If the job run failed for any reason, output an appropriate
            //    error message.
            //
===================================================================

            switch jobRunState {
                case .succeeded:
                    print("Job run succeeded. JSON files are in the Amazon S3
 path:")
                    print("    \(outputURL)")
                    jobRunFinished = true
                case .stopped:
                    jobRunFinished = true
                case .error:
                    print("*** Error: Job run ended in an error.
 \(jobRun.errorMessage ?? "")")
                    jobRunFinished = true
                case .failed:
```

```
                    print("*** Error: Job run failed. \(jobRun.errorMessage ??
 "")")
                    jobRunFinished = true
                case .timeout:
                    print("*** Warning: Job run timed out.")
                    jobRunFinished = true
                default:
                    do {
                        try await Task.sleep(for: .milliseconds(250))
                    } catch {
                        print("*** Error pausing the task.")
                    }
            }
        } while jobRunFinished != true

        //=====================================================================
        // 7.5. List the job runs for this job, showing each job run's ID and
        // its execution time.
        //=====================================================================

        print("Getting all job runs for the job \(jobName):")
        let jobRuns = await getJobRuns(glueClient: glueClient, name: jobName)

        if jobRuns.count == 0 {
            print("    <no job runs found>")
        } else {
            print("Found \(jobRuns.count) job runs... listing execution times:")
            for jobRun in jobRuns {
                print("    \(jobRun.id ?? "<unnamed>"): \(jobRun.executionTime)
 seconds")
            }
        }

        //=====================================================================
        // 8. List the jobs for the user's account.
        //=====================================================================

        print("\nThe account has the following jobs:")
        let jobs = await listJobs(glueClient: glueClient)

        if jobs.count == 0 {
            print("    <no jobs found>")
        } else {
            for job in jobs {
```

```
                print("    \(job)")
            }
        }

        //=====================================================================
        // 9. Get the job run details for a job run.
        //=====================================================================

        print("Information about the job run:")
        let jobRun = await getJobRun(glueClient: glueClient, name: jobName, id:
jobRunID)

        guard let jobRun else {
            print("*** Unable to retrieve the job run.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        let startDate = jobRun.startedOn ?? Date(timeIntervalSince1970: 0)
        let endDate = jobRun.completedOn ?? Date(timeIntervalSince1970: 0)
        let dateFormatter: DateFormatter = DateFormatter()
        dateFormatter.dateStyle = .long
        dateFormatter.timeStyle = .long

        print("    Started at: \(dateFormatter.string(from: startDate))")
        print("  Completed at: \(dateFormatter.string(from: endDate))")

        //=====================================================================
        // 10. Delete the job.
        //=====================================================================

        print("\nDeleting the job...")
        _ = await deleteJob(glueClient: glueClient, name: jobName)

        //=====================================================================
        // 11. Delete the database and tables created by this example.
        //=====================================================================

        print("Deleting the database...")
        _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
```

```
        //=====================================================================
        // 12. Delete the crawler.
        //=====================================================================

        print("Deleting the crawler...")
        if !(await deleteCrawler(glueClient: glueClient, name: crawlerName)) {
            return
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases

- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Ações para AWS Glue usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Glue ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a AWS Glue API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para AWS Glue usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Glue.

Exemplos

- Use CreateCrawlercom um AWS SDK
- Use CreateJobcom um AWS SDK ou CLI
- Use DeleteCrawlercom um AWS SDK
- Use DeleteDatabasecom um AWS SDK
- Use DeleteJobcom um AWS SDK ou CLI
- Use DeleteTablecom um AWS SDK
- Use GetCrawlercom um AWS SDK
- Use GetDatabasecom um AWS SDK
- Use GetDatabasescom um AWS SDK ou CLI
- Use GetJobcom um AWS SDK ou CLI
- Use GetJobRuncom um AWS SDK ou CLI

- Use GetJobRunscom um AWS SDK ou CLI
- Use GetTablescom um AWS SDK ou CLI
- Use ListJobscom um AWS SDK
- Use StartCrawlercom um AWS SDK ou CLI
- Use StartJobRuncom um AWS SDK ou CLI

Use **`CreateCrawler`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateCrawler`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
(Amazon S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
```

```
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte CreateCrawlera Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::S3Target s3Target;
       s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
       Aws::Glue::Model::CrawlerTargets crawlerTargets;
       crawlerTargets.AddS3Targets(s3Target);

       Aws::Glue::Model::CreateCrawlerRequest request;
       request.SetTargets(crawlerTargets);
       request.SetName(CRAWLER_NAME);
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
       request.SetRole(roleArn);

       Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully created the crawler." << std::endl;
       }
       else {
           std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
           return false;
```

```
        }
```

- Para obter detalhes da API, consulte CreateCrawlera Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the
data source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler
will store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   public static void createGlueCrawler(GlueClient glueClient,
                                        String iam,
                                        String s3Path,
                                        String cron,
                                        String dbName,
                                        String crawlerName) {

       try {
           S3Target s3Target = S3Target.builder()
               .path(s3Path)
               .build();
```

```
            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CreateCrawlera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});
```

```
  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte CreateCrawlera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
```

```
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- Para obter detalhes da API, consulte a CreateCrawlerreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
```

```
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path):
Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- Para obter detalhes da API, consulte CreateCrawlera Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
```

```
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created
by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateCrawlerReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
 the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
```

```
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte CreateCrawlera Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
```

```
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;
```

- Para obter detalhes da API, consulte a CreateCrawlerreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/AWSGlueServiceRole-Test'
        " iv_database_name = 'my-database'
        " iv_table_prefix = 'test_'
        " iv_s3_target = 's3://example-bucket/data/'

        DATA(lt_s3_targets) = VALUE /aws1/cl_glus3target=>tt_s3targetlist(
          ( NEW /aws1/cl_glus3target( iv_path = iv_s3_target ) ) ).

        DATA(lo_targets) = NEW /aws1/cl_glucrawlertargets(
          it_s3targets = lt_s3_targets ).

        lo_glu->createcrawler(
          iv_name = iv_crawler_name
          iv_role = iv_role_arn
```

```
          iv_databasename = iv_database_name
          iv_tableprefix = iv_table_prefix
          io_targets = lo_targets ).
        MESSAGE 'Crawler created successfully.' TYPE 'I'.
      CATCH /aws1/cx_glualreadyexistsex.
        MESSAGE 'Crawler already exists.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateCrawlerreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
    ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
```

```
   ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
   ///   - databaseName: The name of an AWS Glue database to operate on.
   ///
   /// - Returns: `true` if the crawler is created successfully, otherwise
`false`.
   func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole:
String,
                      s3Path: String, cronSchedule: String, databaseName:
String) async -> Bool {
       let s3Target = GlueClientTypes.S3Target(path: s3url)
       let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])

       do {
           _ = try await glueClient.createCrawler(
               input: CreateCrawlerInput(
                   databaseName: databaseName,
                   description: "Created by the AWS SDK for Swift Scenario
Example for AWS Glue.",
                   name: crawlerName,
                   role: iamRole,
                   schedule: cronSchedule,
                   tablePrefix: tablePrefix,
                   targets: targetList
               )
           )
       } catch _ as AlreadyExistsException {
           print("*** A crawler named \"\(crawlerName)\" already exists.")
           return false
       } catch _ as OperationTimeoutException {
           print("*** The attempt to create the AWS Glue crawler timed out.")
           return false
       } catch {
           print("*** An unexpected error occurred creating the AWS Glue
crawler: \(error.localizedDescription)")
           return false
       }

       return true
   }
```

- Para obter detalhes da API, consulte CreateCrawlera referência da API AWS SDK for Swift.

Use **`CreateJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
```

```
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::CreateJobRequest request;
```

```
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar um trabalho para transformar dados

O exemplo de `create-job` a seguir cria um trabalho de streaming que executa um script armazenado no S3.

```
aws glue create-job \
    --name my-testing-job \
    --role AWSGlueServiceRoleDefault \
    --command '{ \
        "Name": "gluestreaming", \
        "ScriptLocation": "s3://amzn-s3-demo-bucket/folder/" \
```

```
    }' \
    --region us-east-1 \
    --output json \
    --default-arguments '{ \
        "--job-language":"scala", \
        "--class":"GlueApp" \
    }' \
    --profile my-profile \
    --endpoint https://glue.us-east-1.amazonaws.com
```

Conteúdo de `test_script.scala`:

```
import com.amazonaws.services.glue.ChoiceOption
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.ResolveSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
    def main(sysArgs: Array[String]) {
        val spark: SparkContext = new SparkContext()
        val glueContext: GlueContext = new GlueContext(spark)
        // @params: [JOB_NAME]
        val args = GlueArgParser.getResolvedOptions(sysArgs,
 Seq("JOB_NAME").toArray)
        Job.init(args("JOB_NAME"), glueContext, args.asJava)
        // @type: DataSource
        // @args: [database = "tempdb", table_name = "s3-source",
 transformation_ctx = "datasource0"]
        // @return: datasource0
        // @inputs: []
        val datasource0 = glueContext.getCatalogSource(database = "tempdb",
 tableName = "s3-source", redshiftTmpDir = "", transformationContext =
 "datasource0").getDynamicFrame()
        // @type: ApplyMapping
        // @args: [mapping = [("sensorid", "int", "sensorid", "int"),
 ("currenttemperature", "int", "currenttemperature", "int"), ("status", "string",
 "status", "string")], transformation_ctx = "applymapping1"]
```

```
        // @return: applymapping1
        // @inputs: [frame = datasource0]
        val applymapping1 = datasource0.applyMapping(mappings = Seq(("sensorid",
 "int", "sensorid", "int"), ("currenttemperature", "int", "currenttemperature",
 "int"), ("status", "string", "status", "string")), caseSensitive = false,
 transformationContext = "applymapping1")
        // @type: SelectFields
        // @args: [paths = ["sensorid", "currenttemperature", "status"],
 transformation_ctx = "selectfields2"]
        // @return: selectfields2
        // @inputs: [frame = applymapping1]
        val selectfields2 = applymapping1.selectFields(paths = Seq("sensorid",
 "currenttemperature", "status"), transformationContext = "selectfields2")
        // @type: ResolveChoice
        // @args: [choice = "MATCH_CATALOG", database = "tempdb", table_name =
 "my-s3-sink", transformation_ctx = "resolvechoice3"]
        // @return: resolvechoice3
        // @inputs: [frame = selectfields2]
        val resolvechoice3 = selectfields2.resolveChoice(choiceOption =
 Some(ChoiceOption("MATCH_CATALOG")), database = Some("tempdb"), tableName =
 Some("my-s3-sink"), transformationContext = "resolvechoice3")
        // @type: DataSink
        // @args: [database = "tempdb", table_name = "my-s3-sink",
 transformation_ctx = "datasink4"]
        // @return: datasink4
        // @inputs: [frame = resolvechoice3]
        val datasink4 = glueContext.getCatalogSink(database = "tempdb",
 tableName = "my-s3-sink", redshiftTmpDir = "", transformationContext =
 "datasink4").writeDynamicFrame(resolvechoice3)
        Job.commit()
    }
}
```

Saída:

```
{
    "Name": "my-testing-job"
}
```

Para obter mais informações, consulte Authoring Jobs in AWS Glue no Glue Developer Guide.AWS

- Para obter detalhes da API, consulte CreateJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new AWS Glue job.
    *
    * @param glueClient     the AWS Glue client to use for the operation
    * @param jobName        the name of the job to create
    * @param iam            the IAM role to associate with the job
    * @param scriptLocation the location of the script to be used by the job
    * @throws GlueException if there is an error creating the job
    */
   public static void createJob(GlueClient glueClient, String jobName, String
iam, String scriptLocation) {
       try {
           JobCommand command = JobCommand.builder()
               .pythonVersion("3")
               .name("glueetl")
               .scriptLocation(scriptLocation)
               .build();

           CreateJobRequest jobRequest = CreateJobRequest.builder()
               .description("A Job created by using the AWS SDK for Java V2")
               .glueVersion("2.0")
               .workerType(WorkerType.G_1_X)
               .numberOfWorkers(10)
               .name(jobName)
               .role(iam)
               .command(command)
               .build();

           glueClient.createJob(jobRequest);
           System.out.println(jobName + " was successfully created.");
```

```
        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo cria uma nova tarefa no AWS Glue. O valor do nome do comando é sempre**`glueetl`**. AWS O Glue é compatível com a execução de scripts de tarefas escritos em Python ou Scala. Neste exemplo, o script de trabalho (MyTestGlueJob.py) é escrito em Python. Os parâmetros do Python são especificados na **`$DefArgs`** variável e, em seguida, passados para o PowerShell comando no **`DefaultArguments`** parâmetro, que aceita uma tabela de hash. Os parâmetros na **`$JobParams`** variável vêm da CreateJob API, documentados no tópico Jobs (https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) da referência da API AWS Glue.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/
MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
```

```
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- Para obter detalhes da API, consulte CreateJobem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo cria uma nova tarefa no AWS Glue. O valor do nome do comando é sempre**`glueetl`**. AWS O Glue é compatível com a execução de scripts de tarefas escritos em Python ou Scala. Neste exemplo, o script de trabalho (MyTestGlueJob.py) é escrito em Python. Os parâmetros do Python são especificados na **`$DefArgs`** variável e, em seguida, passados para o PowerShell comando no **`DefaultArguments`** parâmetro, que aceita uma tabela de hash. Os parâmetros na **`$JobParams`** variável vêm da CreateJob API, documentados no tópico Jobs (https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) da referência da API AWS Glue.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/
MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
```

```
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- Para obter detalhes da API, consulte CreateJobem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
 that can
```

```
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateJobReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
```

```
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;
```

- Para obter detalhes da API, consulte a CreateJobreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        " iv_description = 'ETL job for data transformation'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/AWSGlueServiceRole-Test'
        " iv_script_location = 's3://example-bucket/scripts/my-script.py'

        DATA(lo_command) = NEW /aws1/cl_glujobcommand(
          iv_name = 'glueetl'
          iv_scriptlocation = iv_script_location
          iv_pythonversion = '3' ).

        lo_glu->createjob(
          iv_name = iv_job_name
          iv_description = iv_description
          iv_role = iv_role_arn
          io_command = lo_command
          iv_glueversion = '3.0' ).
        MESSAGE 'Job created successfully.' TYPE 'I'.
      CATCH /aws1/cx_glualreadyexistsex.
        MESSAGE 'Job already exists.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateJobreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
```

```
                    description: "Created by the AWS SDK for Swift Glue basic
scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }
```

- Para obter detalhes da API, consulte CreateJoba referência da API AWS SDK for Swift.

Use **`DeleteCrawler`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteCrawler`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
```

```
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
```

```
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
```

- Para obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a specific AWS Glue crawler.
    *
    * @param glueClient  the AWS Glue client object
    * @param crawlerName the name of the crawler to be deleted
    * @throws GlueException if an error occurs during the deletion process
    */
   public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
       try {
           DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
               .name(crawlerName)
               .build();

           glueClient.deleteCrawler(deleteCrawlerRequest);
           System.out.println(crawlerName + " was deleted");

       } catch (GlueException e) {
           throw e;
       }
```

```
    }
```

- Para obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
```

```
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteCrawlerReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
```

```
#
# @param name [String] The name of the crawler to delete.
# @return [void]
def delete_crawler(name)
  @glue_client.delete_crawler(name: name)
rescue Aws::Glue::Errors::ServiceError => e
  @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
  raise
end
```

- Para obter detalhes da API, consulte DeleteCrawlera Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteCrawlerreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        lo_glu->deletecrawler( iv_name = iv_crawler_name ).
        MESSAGE 'Crawler deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_glucrawlerrunningex.
        MESSAGE 'Crawler is currently running.' TYPE 'E'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Crawler does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluschdrtransingex.
        MESSAGE 'Scheduler is transitioning.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCrawlerreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
```

```
import AWSGlue

    /// Delete an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
        return true
    }
```

- Para obter detalhes da API, consulte DeleteCrawlera referência da API AWS SDK for Swift.

Use **`DeleteDatabase`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteDatabase`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos do catálogo de dados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Para obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact
with the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
```

```
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName)
 {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteDatabaseReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte DeleteDatabasea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_database()
            .name(self.database())
            .send()
            .await
```

```
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteDatabasereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_database_name = 'my-database'
    lo_glu->deletedatabase( iv_name = iv_database_name ).
    MESSAGE 'Database deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Database does not exist.' TYPE 'E'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDatabasereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Delete the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient:
 glueClient, databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables. If there's only one table, use
            // `deleteTable()`, otherwise, use `batchDeleteTable()`. You can
            // use `batchDeleteTable()` for a single table, but this
```

```
            // demonstrates the use of `deleteTable()`.

            if tableNames.count == 1 {
                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
            } else {
                do {
                    print("    Deleting tables...")
                    _ = try await glueClient.batchDeleteTable(
                        input: BatchDeleteTableInput(
                            databaseName: databaseName,
                            tablesToDelete: tableNames
                        )
                    )
                } catch {
                    print("*** Unable to delete the tables.")
                }
            }
        }

        // Delete the database itself.

        do {
            print("    Deleting the database itself...")
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
        } catch {
            print("*** Unable to delete the database.")
            return false
        }
        return true
    }
```

- Para obter detalhes da API, consulte DeleteDatabasea referência da API AWS SDK for Swift.

Use **`DeleteJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteJoba Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Para obter detalhes da API, consulte DeleteJoba Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir um trabalho

O exemplo de `delete-job` a seguir exclui um trabalho que não é mais necessário.

```
aws glue delete-job \
    --job-name my-testing-job
```

Saída:

```
{
    "JobName": "my-testing-job"
}
```

Para obter mais informações, consulte Trabalhando com trabalhos no AWS Glue Console no AWS Glue Developer Guide.

- Para obter detalhes da API, consulte DeleteJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();
```

```
            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteJoba Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteJoba Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteJoba Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
```

```
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteJobReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
```

```
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte DeleteJoba Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteJobreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        lo_glu->deletejob( iv_jobname = iv_job_name ).
        MESSAGE 'Job deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteJobreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
```

```
import AWSGlue

    /// Delete an AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to delete.
    ///
    /// - Returns: `true` if the job is successfully deleted, otherwise `false`.
    func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
        do {
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }
```

- Para obter detalhes da API, consulte DeleteJoba referência da API AWS SDK for Swift.

Use **`DeleteTable`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos do catálogo de dados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
 { Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para .NET da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
```

```
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTableReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }
```

- Para obter detalhes da API, consulte a DeleteTablereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_database_name = 'my-database'
        " iv_table_name = 'my-table'
        lo_glu->deletetable(
          iv_databasename = iv_database_name
          iv_name = iv_table_name ).
        MESSAGE 'Table deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
```

```
        MESSAGE 'Table or database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTablereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
```

- Para obter detalhes da API, consulte DeleteTablea referência da API AWS SDK for Swift.

Use **`GetCrawler`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetCrawler`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get information about an AWS Glue crawler.
/// </summary>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Crawler object describing the crawler.</returns>
public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
{
    var crawlerRequest = new GetCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        var databaseName = response.Crawler.DatabaseName;
        Console.WriteLine($"{crawlerName} has the database {databaseName}");
        return response.Crawler;
    }
```

```
        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }
```

- Para obter detalhes da API, consulte GetCrawlera Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                            crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
```

```
                        << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte GetCrawlera Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves a specific crawler from the AWS Glue service and waits for it to
be in the "READY" state.
    *
    * @param glueClient  the AWS Glue client used to interact with the Glue
service
    * @param crawlerName the name of the crawler to be retrieved
    */
   public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) throws InterruptedException {
       try {
           GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
               .name(crawlerName)
               .build();

           boolean ready = false;
           while (!ready) {
               GetCrawlerResponse response =
glueClient.getCrawler(crawlerRequest);
               String status = response.crawler().stateAsString();
               if (status.compareTo("READY") == 0) {
                   ready = true;
               }
```

```
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte GetCrawlera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetCrawlera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- Para obter detalhes da API, consulte a GetCrawlerreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
```

```
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- Para obter detalhes da API, consulte GetCrawlera Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
```

```
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler
```

- Para obter detalhes da API, consulte a GetCrawlerReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
```

```
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte GetCrawlera Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
            let tmp_crawler = glue
                .get_crawler()
                .name(self.crawler())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a GetCrawlerreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        oo_result = lo_glu->getcrawler( iv_name = iv_crawler_name ).
        DATA(lo_crawler) = oo_result->get_crawler( ).
        MESSAGE 'Crawler information retrieved.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Crawler does not exist.' TYPE 'I'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetCrawlerreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Get the state of the specified AWS Glue crawler.
```

```
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
    func getCrawlerState(glueClient: GlueClient, name: String) async ->
GlueClientTypes.CrawlerState {
        do {
            let output = try await glueClient.getCrawler(
                input: GetCrawlerInput(name: name)
            )

            // If the crawler or its state is `nil`, report that the crawler
            // is stopping. This may not be what you want for your
            // application but it works for this one!

            guard let crawler = output.crawler else {
                return GlueClientTypes.CrawlerState.stopping
            }
            guard let state = crawler.state else {
                return GlueClientTypes.CrawlerState.stopping
            }
            return state
        } catch {
            return GlueClientTypes.CrawlerState.stopping
        }
    }
```

- Para obter detalhes da API, consulte GetCrawlera referência da API AWS SDK for Swift.

Use **`GetDatabase`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetDatabase`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos do catálogo de dados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }
```

- Para obter detalhes da API, consulte GetDatabasea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetDatabaseRequest request;
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte GetDatabasea Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the specific database from the AWS Glue service.
    *
    * @param glueClient   an instance of the AWS Glue client used to interact
with the service
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from
the AWS Glue service
    */
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " +
createDate);

       } catch (GlueException e) {
           throw e;
       }
   }
```

- Para obter detalhes da API, consulte GetDatabasea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetDatabasea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }
```

```
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- Para obter detalhes da API, consulte a GetDatabasereferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $databaseName = "doc-example-database-$uniqid";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

- Para obter detalhes da API, consulte GetDatabasea Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

- Para obter detalhes da API, consulte a GetDatabaseReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte GetDatabasea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;
```

- Para obter detalhes da API, consulte a GetDatabasereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        " iv_database_name = 'my-database'
        oo_result = lo_glu->getdatabase( iv_name = iv_database_name ).
        DATA(lo_database) = oo_result->get_database( ).
        MESSAGE 'Database information retrieved.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetDatabasereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
```

```
    ///   specified database, or `nil` if an error occurs or the database
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }
```

- Para obter detalhes da API, consulte GetDatabasea referência da API AWS SDK for Swift.

Use **`GetDatabases`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetDatabases`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para listar as definições de alguns ou de todos os bancos de dados no AWS Glue Data Catalog

O exemplo de `get-databases` a seguir retorna informações sobre os bancos de dados no Catálogo de Dados.

```
aws glue get-databases
```

Saída:

```
{
    "DatabaseList": [
        {
            "Name": "default",
            "Description": "Default Hive database",
            "LocationUri": "file:/spark-warehouse",
            "CreateTime": 1602084052.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "flights-db",
            "CreateTime": 1587072847.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "legislators",
            "CreateTime": 1601415625.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
```

```
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "tempdb",
            "CreateTime": 1601498566.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        }
    ]
}
```

Para obter mais informações, consulte Definir um banco de dados no seu Catálogo de Dados no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte GetDatabasesem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getDatabases = () => {
  const client = new GlueClient({});
```

```
  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetDatabasesa Referência AWS SDK para JavaScript da API.

Use **`GetJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para recuperar informações sobre um trabalho

O exemplo de `get-job` a seguir recupera informações sobre um trabalho.

```
aws glue get-job \
    --job-name my-testing-job
```

Saída:

```
{
    "Job": {
        "Name": "my-testing-job",
        "Role": "Glue_DefaultRole",
        "CreatedOn": 1602805698.167,
        "LastModifiedOn": 1602805698.167,
        "ExecutionProperty": {
            "MaxConcurrentRuns": 1
        },
        "Command": {
            "Name": "gluestreaming",
```

```
            "ScriptLocation": "s3://janetst-bucket-01/Scripts/test_script.scala",
            "PythonVersion": "2"
        },
        "DefaultArguments": {
            "--class": "GlueApp",
            "--job-language": "scala"
        },
        "MaxRetries": 0,
        "AllocatedCapacity": 10,
        "MaxCapacity": 10.0,
        "GlueVersion": "1.0"
    }
}
```

Para obter mais informações, consulte Trabalhos no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte GetJobem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getJob = (jobName) => {
  const client = new GlueClient({});

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJoba Referência AWS SDK para JavaScript da API.

Use **`GetJobRun`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetJobRun`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }
```

- Para obter detalhes da API, consulte GetJobRuna Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte GetJobRuna Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para obter informações sobre uma execução de trabalho

O exemplo `get-job-run` a seguir recupera informações sobre uma execução de trabalho.

```
aws glue get-job-run \
    --job-name "Combine legistators data" \
    --run-id jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e
```

Saída:

```
{
    "JobRun": {
        "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
        "Attempt": 0,
        "JobName": "Combine legistators data",
        "StartedOn": 1602873931.255,
        "LastModifiedOn": 1602874075.985,
        "CompletedOn": 1602874075.985,
        "JobRunState": "SUCCEEDED",
        "Arguments": {
            "--enable-continuous-cloudwatch-log": "true",
            "--enable-metrics": "",
            "--enable-spark-ui": "true",
            "--job-bookmark-option": "job-bookmark-enable",
            "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
        },
        "PredecessorRuns": [],
        "AllocatedCapacity": 10,
        "ExecutionTime": 117,
        "Timeout": 2880,
        "MaxCapacity": 10.0,
        "WorkerType": "G.1X",
        "NumberOfWorkers": 10,
        "LogGroupName": "/aws-glue/jobs",
        "GlueVersion": "2.0"
    }
}
```

Para obter mais informações, consulte Execuções de trabalhos no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte GetJobRunem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJobRuna Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
 $outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
 ['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
 Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- Para obter detalhes da API, consulte GetJobRuna Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]
```

- Para obter detalhes da API, consulte a GetJobRunReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte GetJobRuna Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let get_job_run = || async {
            Ok::<JobRun, GlueMvpError>(
                glue.get_job_run()
                    .job_name(self.job())
                    .run_id(job_run_id.to_string())
                    .send()
                    .await
                    .map_err(GlueMvpError::from_glue_sdk)?
                    .job_run()
                    .ok_or_else(|| GlueMvpError::Unknown("Failed to get
job_run".into()))?
                    .to_owned(),
            )
        };

        let mut job_run = get_job_run().await?;
        let mut state =
job_run.job_run_state().unwrap_or(&unknown_state).to_owned();

        while matches!(
            state,
            JobRunState::Starting | JobRunState::Stopping | JobRunState::Running
        ) {
            info!(?state, "Waiting for job to finish");
            tokio::time::sleep(self.wait_delay).await;

            job_run = get_job_run().await?;
            state = job_run.job_run_state().unwrap_or(&unknown_state).to_owned();
        }
```

- Para obter detalhes da API, consulte a GetJobRunreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_job_name = 'my-etl-job'
    " iv_run_id = 'jr_abcd1234567890abcdef1234567890abcdef12345678'
    oo_result = lo_glu->getjobrun(
      iv_jobname = iv_job_name
      iv_runid = iv_run_id ).
    DATA(lo_job_run) = oo_result->get_jobrun( ).
    MESSAGE 'Job run information retrieved.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Job or job run does not exist.' TYPE 'E'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetJobRunreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String)
 async -> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
    }
```

- Para obter detalhes da API, consulte GetJobRuna referência da API AWS SDK for Swift.

Use **`GetJobRuns`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetJobRuns`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
```

```
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }
```

- Para obter detalhes da API, consulte GetJobRunsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
       getJobRunsRequest.SetJobName(jobName);

       Aws::String nextToken; // Used for pagination.
       std::vector<Aws::Glue::Model::JobRun> allJobRuns;
       do {
           if (!nextToken.empty()) {
               getJobRunsRequest.SetNextToken(nextToken);
           }
           Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome =
client.GetJobRuns(
                   getJobRunsRequest);
```

```
            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
 jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(),
 jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());
```

- Para obter detalhes da API, consulte GetJobRunsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para obter informações sobre todas as execuções de um trabalho

O exemplo `get-job-runs` a seguir recupera informações sobre execuções de trabalho para um trabalho.

```
aws glue get-job-runs \
    --job-name "my-testing-job"
```

Saída:

```
{
    "JobRuns": [
        {
            "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
            "Attempt": 0,
            "JobName": "my-testing-job",
            "StartedOn": 1602873931.255,
```

```
            "LastModifiedOn": 1602874075.985,
            "CompletedOn": 1602874075.985,
            "JobRunState": "SUCCEEDED",
            "Arguments": {
                "--enable-continuous-cloudwatch-log": "true",
                "--enable-metrics": "",
                "--enable-spark-ui": "true",
                "--job-bookmark-option": "job-bookmark-enable",
                "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
            },
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 117,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_2",
            "Attempt": 2,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "JobName": "my-testing-job",
            "StartedOn": 1602811168.496,
            "LastModifiedOn": 1602811282.39,
            "CompletedOn": 1602811282.39,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
                Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                Request ID: 021AAB703DB20A2D;
                S3 Extended Request ID: teZk24Y09TkXzBvMPG502L5VJBhe9DJuWA9/
TXtuGOqfByajkfL/Tlqt5JBGdEGpigAqzdMDM/U=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 110,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
```

```
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "Attempt": 1,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f",
            "JobName": "my-testing-job",
            "StartedOn": 1602811020.518,
            "LastModifiedOn": 1602811138.364,
            "CompletedOn": 1602811138.364,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
                Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                Request ID: 2671D37856AE7ABB;
                S3 Extended Request ID: RLJCJw20brV
+PpC6GpORahyF2fp9flB5SSb2bTGPnUSPVizLXRl1PN3QZldb+v1o9qRVktNYbW8=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 113,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        }
    ]
}
```

Para obter mais informações, consulte Execuções de trabalhos no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte GetJobRunsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the job runs for a given Glue job and prints the status of the
job runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;
```

```
                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
 jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
 jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte GetJobRunsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });
```

```
  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJobRunsa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
 Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- Para obter detalhes da API, consulte GetJobRunsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]
```

- Para obter detalhes da API, consulte a GetJobRunsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte GetJobRunsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_job_name = 'my-etl-job'
    oo_result = lo_glu->getjobruns( iv_jobname = iv_job_name ).
    DATA(lt_job_runs) = oo_result->get_jobruns( ).
    MESSAGE 'Job runs retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Job does not exist.' TYPE 'E'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetJobRunsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Return a list of the job runs for the specified job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job for which to return its job runs.
    ///   - maxResults: The maximum number of job runs to return (default:
    ///     1000).
    ///
    /// - Returns: An array of `GlueClientTypes.JobRun` objects describing
    ///   each job run.
    func getJobRuns(glueClient: GlueClient, name jobName: String, maxResults:
 Int? = nil) async -> [GlueClientTypes.JobRun] {
        do {
            let output = try await glueClient.getJobRuns(
                input: GetJobRunsInput(
                    jobName: jobName,
                    maxResults: maxResults
                )
            )

            guard let jobRuns = output.jobRuns else {
                print("*** No job runs found.")
                return []
            }

            return jobRuns
        } catch is EntityNotFoundException {
            print("*** The specified job name, \(jobName), doesn't exist.")
            return []
        } catch {
            print("*** Unexpected error getting job runs:")
            dump(error)
            return []
        }
    }
```

- Para obter detalhes da API, consulte GetJobRunsa referência da API AWS SDK for Swift.

Use **`GetTables`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetTables`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get a list of tables for an AWS Glue database.
/// </summary>
/// <param name="dbName">The name of the database.</param>
/// <returns>A list of Table objects.</returns>
public async Task<List<Table>> GetTablesAsync(string dbName)
{
    var request = new GetTablesRequest { DatabaseName = dbName };
    var tables = new List<Table>();

    // Get a paginator for listing the tables.
    var tablePaginator = _amazonGlue.Paginators.GetTables(request);

    await foreach (var response in tablePaginator.Responses)
    {
        tables.AddRange(response.TableList);
    }

    return tables;
}
```

- Para obter detalhes da API, consulte GetTablesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetTablesRequest request;
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       std::vector<Aws::Glue::Model::Table> all_tables;
       Aws::String nextToken; // Used for pagination.
       do {
           Aws::Glue::Model::GetTablesOutcome outcome =
client.GetTables(request);

           if (outcome.IsSuccess()) {
               const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
               all_tables.insert(all_tables.end(), tables.begin(),
tables.end());
               nextToken = outcome.GetResult().GetNextToken();
           }
           else {
               std::cerr << "Error getting the tables. "
                         << outcome.GetError().GetMessage()
                         << std::endl;
               deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                            clientConfig);
               return false;
           }
```

```
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " <<
all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex -
1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
```

- Para obter detalhes da API, consulte GetTablesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar as definições de algumas ou de todas as tabelas no banco de dados especificado

O exemplo de `get-tables` a seguir retorna informações sobre as tabelas no banco de dados especificado.

```
aws glue get-tables --database-name 'tempdb'
```

Saída:

```
{
    "TableList": [
        {
```

```
            "Name": "my-s3-sink",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730539.0,
            "UpdateTime": 1602730539.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/test-s3-output/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
        {
            "Name": "s3-source",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730658.0,
            "UpdateTime": 1602730658.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
```

```
                {
                    "Name": "sensorid",
                    "Type": "int"
                },
                {
                    "Name": "currenttemperature",
                    "Type": "int"
                },
                {
                    "Name": "status",
                    "Type": "string"
                }
            ],
            "Location": "s3://janetst-bucket-01/",
            "Compressed": false,
            "NumberOfBuckets": 0,
            "SortColumns": [],
            "StoredAsSubDirectories": false
        },
        "Parameters": {
            "classification": "json"
        },
        "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
        "IsRegisteredWithLakeFormation": false,
        "CatalogId": "007436865787"
    },
    {
        "Name": "test-kinesis-input",
        "DatabaseName": "tempdb",
        "CreateTime": 1601507001.0,
        "UpdateTime": 1601507001.0,
        "Retention": 0,
        "StorageDescriptor": {
            "Columns": [
                {
                    "Name": "sensorid",
                    "Type": "int"
                },
                {
                    "Name": "currenttemperature",
                    "Type": "int"
                },
                {
                    "Name": "status",
```

```
                        "Type": "string"
                    }
                ],
                "Location": "my-testing-stream",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "Parameters": {
                    "kinesisUrl": "https://kinesis.us-east-1.amazonaws.com",
                    "streamName": "my-testing-stream",
                    "typeOfData": "kinesis"
                },
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        }
    ]
}
```

Para obter mais informações, consulte Definindo tabelas no catálogo de dados do AWS Glue no AWS Glue Developer Guide.

- Para obter detalhes da API, consulte GetTablesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the names of the tables in the specified Glue database.
     *
     * @param glueClient the Glue client to use for the operation
     * @param dbName     the name of the Glue database to retrieve the table
names from
     * @return the name of the first table retrieved, or an empty string if no
tables were found
     */
    public static String getGlueTables(GlueClient glueClient, String dbName) {
        String myTableName = "";
        try {
            GetTablesRequest tableRequest = GetTablesRequest.builder()
                .databaseName(dbName)
                .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }
```

- Para obter detalhes da API, consulte GetTablesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetTablesa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
```

```
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Para obter detalhes da API, consulte GetTablesa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]
```

- Para obter detalhes da API, consulte a GetTablesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
```

```
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte GetTablesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

- Para obter detalhes da API, consulte a GetTablesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_database_name = 'my-database'
        oo_result = lo_glu->gettables( iv_databasename = iv_database_name ).
        DATA(lt_tables) = oo_result->get_tablelist( ).
        MESSAGE 'Tables retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetTablesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
```

```
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async
-> [GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }
```

- Para obter detalhes da API, consulte GetTablesa referência da API AWS SDK for Swift.

Use **`ListJobs`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListJobs`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::ListJobsRequest listJobsRequest;

       Aws::String nextToken;
       std::vector<Aws::String> allJobNames;

       do {
           if (!nextToken.empty()) {
               listJobsRequest.SetNextToken(nextToken);
           }
           Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                   listJobsRequest);

           if (listRunsOutcome.IsSuccess()) {
               const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
               allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
               nextToken = listRunsOutcome.GetResult().GetNextToken();
           }
           else {
               std::cerr << "Error listing jobs. "
                         << listRunsOutcome.GetError().GetMessage()
                         << std::endl;
           }
       } while (!nextToken.empty());
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const listJobs = () => {
  const client = new GlueClient({});

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
```

```
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
 Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]
```

- Para obter detalhes da API, consulte a ListJobsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
```

```
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- Para obter detalhes da API, consulte a ListJobsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_glu->listjobs( ).
    DATA(lt_job_names) = oo_result->get_jobnames( ).
    MESSAGE 'Job list retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'No jobs found.' TYPE 'I'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListJobsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Return a list of the AWS Glue jobs listed on the user's account.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - maxJobs: The maximum number of jobs to return (default: 100).
    ///
    /// - Returns: An array of strings listing the names of all available AWS
    ///   Glue jobs.
    func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
        var jobList: [String] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.listJobs(
                    input: ListJobsInput(
                        maxResults: maxJobs,
                        nextToken: nextToken
                    )
                )

                guard let jobs = output.jobNames else {
                    return jobList
                }

                jobList = jobList + jobs
                nextToken = output.nextToken
            } catch {
                return jobList
            }
        } while (nextToken != nil)

        return jobList
    }
```

- Para obter detalhes da API, consulte ListJobsa referência da API AWS SDK for Swift.

Use **`StartCrawler`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartCrawler`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Start an AWS Glue crawler.
/// </summary>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StartCrawlerAsync(string crawlerName)
{
    var crawlerRequest = new StartCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte StartCrawlera Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::StartCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


       if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                   outcome.GetError().GetErrorType())) {
           if (!outcome.IsSuccess()) {
               std::cout << "Crawler was already started." << std::endl;
           }
           else {
               std::cout << "Successfully started crawler." << std::endl;
           }

           std::cout << "This may take a while to run." << std::endl;

           Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
           int iterations = 0;
           while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
               std::this_thread::sleep_for(std::chrono::seconds(1));
               ++iterations;
               if ((iterations % 10) == 0) { // Log status every 10 seconds.
```

```
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                    crawlerState)
                            << ". After " << iterations
                            << " seconds elapsed."
                            << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                            << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte StartCrawlera Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para iniciar um crawler

O exemplo de `start-crawler` a seguir inicia um crawler.

```
aws glue start-crawler --name my-crawler
```

Saída:

```
None
```

Para obter mais informações, consulte Definição de crawlers no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte StartCrawlerem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
```

```
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte StartCrawlera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte StartCrawlera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- Para obter detalhes da API, consulte a StartCrawlerreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);
```

```
    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Para obter detalhes da API, consulte StartCrawlera Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a StartCrawlerReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
```

```
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte StartCrawlera Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let start_crawler =
glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

- Para obter detalhes da API, consulte a StartCrawlerreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        lo_glu->startcrawler( iv_name = iv_crawler_name ).
        MESSAGE 'Crawler started successfully.' TYPE 'I'.
      CATCH /aws1/cx_glucrawlerrunningex.
        MESSAGE 'Crawler is already running.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Crawler does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartCrawlerreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue
```

```
    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise
`false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }
```

- Para obter detalhes da API, consulte StartCrawlera referência da API AWS SDK for Swift.

Use **`StartJobRun`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartJobRun`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }
```

- Para obter detalhes da API, consulte StartJobRuna Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();
```

```
                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                          jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
```

```
                        << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                           clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte StartJobRuna Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para iniciar a execução de um trabalho

O exemplo de `start-job-run` a seguir inicia um trabalho.

```
aws glue start-job-run \
    --job-name my-job
```

Saída:

```
{
    "JobRunId":
 "jr_22208b1f44eb5376a60569d4b21dd20fcb8621e1a366b4e7b2494af764b82ded"
}
```

Para obter mais informações, consulte Criação de trabalhos no Guia do desenvolvedor do AWS Glue.

- Para obter detalhes da API, consulte StartJobRunem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Starts a job run in AWS Glue.
    *
    * @param glueClient    the AWS Glue client to use for the job run
    * @param jobName       the name of the Glue job to run
    * @param inputDatabase the name of the input database
    * @param inputTable    the name of the input table
    * @param outBucket     the URL of the output S3 bucket
    * @throws GlueException if there is an error starting the job run
    */
   public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                               String outBucket) {
       try {
           Map<String, String> myMap = new HashMap<>();
           myMap.put("--input_database", inputDatabase);
           myMap.put("--input_table", inputTable);
           myMap.put("--output_bucket_url", outBucket);

           StartJobRunRequest runRequest = StartJobRunRequest.builder()
               .workerType(WorkerType.G_1_X)
               .numberOfWorkers(10)
               .arguments(myMap)
               .jobName(jobName)
               .build();

           StartJobRunResponse response = glueClient.startJobRun(runRequest);
           System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

       } catch (GlueException e) {
```

```
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte StartJobRuna Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte StartJobRuna Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl): Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte StartJobRuna Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_job_run(self, name, input_database, input_table,
 output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
 tables
                               that describe the source data. This is typically
 created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
 the
            # Python ETL script to determine the location of input and output
 data.
```

```
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]
```

- Para obter detalhes da API, consulte a StartJobRunReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
```

```
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
 job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte StartJobRuna Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

- Para obter detalhes da API, consulte a StartJobRunreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        " iv_input_database = 'my-database'
        " iv_input_table = 'my-table'
        " iv_output_bucket_url = 's3://example-output-bucket/'
```

```
        DATA lt_arguments TYPE /aws1/cl_glugenericmap_w=>tt_genericmap.
        lt_arguments = VALUE #(
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--input_database'
            value = NEW /aws1/cl_glugenericmap_w( iv_value =
iv_input_database ) ) )
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--input_table'
            value = NEW /aws1/cl_glugenericmap_w( iv_value = iv_input_table ) ) )
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--output_bucket_url'
            value = NEW /aws1/cl_glugenericmap_w( iv_value =
iv_output_bucket_url ) ) ) ).

        DATA(oo_result) = lo_glu->startjobrun(
          iv_jobname = iv_job_name
          it_arguments = lt_arguments ).
        ov_job_run_id = oo_result->get_jobrunid( ).
        MESSAGE 'Job run started successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluconcurrentrunsex00.
        MESSAGE 'Maximum concurrent runs exceeded.' TYPE 'E'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Job does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartJobRunreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to run.
    ///   - databaseName: The name of the AWS Glue database to run the job
 against.
    ///   - tableName: The name of the table in the database to run the job
 against.
    ///   - outputURL: The AWS S3 URI of the bucket location into which to
    ///     write the resulting output.
    ///
    /// - Returns: `true` if the job run is started successfully, otherwise
 `false`.
    func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
 String,
                     tableName: String, outputURL: String) async -> String? {
        do {
            let output = try await glueClient.startJobRun(
                input: StartJobRunInput(
                    arguments: [
                        "--input_database": databaseName,
                        "--input_table": tableName,
                        "--output_bucket_url": outputURL
                    ],
                    jobName: jobName,
                    numberOfWorkers: 10,
                    workerType: .g1x
                )
```

```
            )

            guard let id = output.jobRunId else {
                return nil
            }

            return id
        } catch {
            return nil
        }
    }
```

- Para obter detalhes da API, consulte StartJobRuna referência da API AWS SDK for Swift.

Cenários para AWS Glue usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS Glue com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS Glue ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos do catálogo de dados

Conceitos básicos do catálogo de dados

O exemplo de código a seguir mostra como:

- Criar um banco de dados.
- Criar uma tabela.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Glue Data Catalog Tutorial Script
# This script demonstrates how to create and manage AWS Glue Data Catalog
 resources using the AWS CLI
# Cost improvements: Reduced API calls, optimized queries, eliminated redundant
 operations
# Reliability improvements: Enhanced error handling, input validation, resource
 tracking

set -euo pipefail

# Setup logging
LOG_FILE="glue-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Glue Data Catalog tutorial script at $(date)"
echo "All operations will be logged to $LOG_FILE"

# Generate a unique identifier for resource names
UNIQUE_ID=$(openssl rand -hex 4)
DB_NAME="tutorial-db-${UNIQUE_ID}"
TABLE_NAME="flights-data-${UNIQUE_ID}"
TABLE_INPUT_FILE="table-input-${UNIQUE_ID}.json"

# Track created resources
declare -a CREATED_RESOURCES=()

# Set default region if not provided
AWS_REGION="${AWS_REGION:-us-east-1}"

# Flag to track if database was successfully created
```

```
DATABASE_CREATED=false

# Trap to ensure cleanup on exit
trap cleanup_resources EXIT

# Function to check command status
check_status() {
    if [ $? -ne 0 ]; then
        echo "ERROR: $1 failed." >&2
        exit 1
    fi
}

# Function to cleanup resources
cleanup_resources() {
    local exit_code=$?
    echo "Attempting to clean up resources..."

    # Delete resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource=${CREATED_RESOURCES[$i]}
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_name=$(echo "$resource" | cut -d':' -f2)

        echo "Deleting $resource_type: $resource_name"

        case $resource_type in
            "table")
                if [ "$DATABASE_CREATED" = true ]; then
                    aws glue delete-table \
                        --database-name "$DB_NAME" \
                        --name "$resource_name" \
                        --region "$AWS_REGION" \
                        2>/dev/null || echo "Warning: Failed to delete table
 $resource_name"
                fi
                ;;
            "database")
                aws glue delete-database \
                    --name "$resource_name" \
                    --region "$AWS_REGION" \
                    2>/dev/null || echo "Warning: Failed to delete database
 $resource_name"
                ;;
```

```
            *)
                echo "Unknown resource type: $resource_type" >&2
                ;;
        esac
    done

    # Clean up temporary files securely
    if [ -f "$TABLE_INPUT_FILE" ]; then
        if command -v shred &> /dev/null; then
            shred -vfz -n 3 "$TABLE_INPUT_FILE" 2>/dev/null || rm -f
 "$TABLE_INPUT_FILE"
        else
            rm -f "$TABLE_INPUT_FILE"
        fi
    fi

    echo "Cleanup completed."
    exit $exit_code
}

# Function to validate prerequisites
validate_prerequisites() {
    # Validate AWS CLI is available
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" >&2
        exit 1
    fi

    # Validate AWS CLI version
    local AWS_CLI_VERSION
    AWS_CLI_VERSION=$(aws --version 2>&1 | cut -d' ' -f1 | cut -d'/' -f2 | cut -
d'.' -f1)
    if [ "$AWS_CLI_VERSION" -lt 1 ]; then
        echo "ERROR: AWS CLI is required" >&2
        exit 1
    fi

    # Validate jq is available for JSON validation
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH" >&2
        exit 1
    fi
```

```
    # Validate AWS credentials and get account identity in single call (cost
 optimization)
    local CALLER_IDENTITY
    CALLER_IDENTITY=$(aws sts get-caller-identity --region "$AWS_REGION" --query
 'Account' --output text 2>/dev/null) || {
        echo "ERROR: Failed to get AWS caller identity. Check credentials and
 permissions." >&2
        exit 1
    }

    if [ -z "$CALLER_IDENTITY" ] || [ "$CALLER_IDENTITY" == "None" ]; then
        echo "ERROR: Unable to determine AWS account identity" >&2
        exit 1
    fi
    echo "Using AWS Account: $CALLER_IDENTITY"
    echo "Using Region: $AWS_REGION"
}

# Function to create database with verification
create_database() {
    echo "Step 1: Creating a database named $DB_NAME"

    if ! aws glue create-database \
        --database-input "Name=$DB_NAME,Description=Database for AWS Glue
 tutorial" \
        --region "$AWS_REGION" \
        --output json > /dev/null 2>&1; then
        echo "ERROR: Failed to create database $DB_NAME" >&2
        exit 1
    fi

    ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
    aws glue tag-resource \
        --resource-arn "arn:aws:glue:${AWS_REGION}:${ACCOUNT_ID}:database/
${DB_NAME}" \
        --tags-to-add Key=project,Value=doc-smith Key=tutorial,Value=glue-gs \
        --region "$AWS_REGION" \
        2>/dev/null || true

    DATABASE_CREATED=true
    CREATED_RESOURCES+=("database:$DB_NAME")
    echo "Database $DB_NAME created successfully."
}
```

```
# Function to prepare table input JSON
prepare_table_input() {
    # Create a temporary JSON file for table input with restricted permissions
    if ! touch "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Failed to create temporary file $TABLE_INPUT_FILE" >&2
        exit 1
    fi

    if ! chmod 600 "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Failed to set permissions on $TABLE_INPUT_FILE" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    cat > "$TABLE_INPUT_FILE" << 'EOF'
{
  "Name": "TABLE_NAME_PLACEHOLDER",
  "StorageDescriptor": {
    "Columns": [
      {
        "Name": "year",
        "Type": "bigint"
      },
      {
        "Name": "quarter",
        "Type": "bigint"
      }
    ],
    "Location": "s3://crawler-public-us-west-2/flight/2016/csv",
    "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
    "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
    "Compressed": false,
    "NumberOfBuckets": -1,
    "SerdeInfo": {
      "SerializationLibrary":
 "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
      "Parameters": {
        "field.delim": ",",
        "serialization.format": ","
      }
    }
  },
  "PartitionKeys": [
    {
```

```
      "Name": "mon",
      "Type": "string"
    }
  ],
  "TableType": "EXTERNAL_TABLE",
  "Parameters": {
    "EXTERNAL": "TRUE",
    "classification": "csv",
    "columnsOrdered": "true",
    "compressionType": "none",
    "delimiter": ",",
    "skip.header.line.count": "1",
    "typeOfData": "file"
  }
}
EOF

    # Replace placeholder with actual table name
    if ! sed -i "s/TABLE_NAME_PLACEHOLDER/$TABLE_NAME/g" "$TABLE_INPUT_FILE" 2>/
dev/null; then
        echo "ERROR: Failed to substitute table name in JSON file" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    # Validate JSON syntax before using it
    if ! jq empty "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Invalid JSON in table input file" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi
}

# Function to create table
create_table() {
    echo "Step 2: Creating a table named $TABLE_NAME in database $DB_NAME"

    prepare_table_input

    local TABLE_ARN
    if ! aws glue create-table \
        --database-name "$DB_NAME" \
        --table-input "file://${TABLE_INPUT_FILE}" \
        --region "$AWS_REGION" \
```

```
        --output json > /dev/null 2>&1; then
        echo "ERROR: Failed to create table $TABLE_NAME" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    aws glue tag-resource \
        --resource-arn "arn:aws:glue:${AWS_REGION}:${ACCOUNT_ID}:table/
${DB_NAME}/${TABLE_NAME}" \
        --tags-to-add Key=project,Value=doc-smith Key=tutorial,Value=glue-gs \
        --region "$AWS_REGION" \
        2>/dev/null || true

    CREATED_RESOURCES+=("table:$TABLE_NAME")
    echo "Table $TABLE_NAME created successfully."
}

# Function to get and display table details
display_table_details() {
    echo "Step 3: Getting details of table $TABLE_NAME"

    if ! aws glue get-table \
        --database-name "$DB_NAME" \
        --name "$TABLE_NAME" \
        --region "$AWS_REGION" \
        --output json; then
        echo "ERROR: Failed to retrieve table details" >&2
        exit 1
    fi
}

# Function to display summary
display_summary() {
    echo ""
    echo "==========================================="
    echo "RESOURCES CREATED"
    echo "==========================================="
    echo "Database: $DB_NAME"
    echo "Table: $TABLE_NAME"
    echo "==========================================="
}

# Main execution flow
validate_prerequisites
```

```
create_database
create_table
display_table_details
display_summary

echo ""
echo "=========================================="
echo "CLEANUP CONFIRMATION"
echo "=========================================="
echo "Starting cleanup process..."

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateDatabase
  - CreateTable
  - DeleteDatabase
  - DeleteTable
  - GetDatabase
  - GetTable

Exemplos de código para AWS Health usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Health com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS Health Guia do desenvolvedor — Mais informações sobre AWS Health.
- AWS Health Referência da API — Detalhes sobre todas as AWS Health ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Health usar AWS SDKs
  - Ações para AWS Health usar AWS SDKs
    - Usar DescribeEvents com uma CLI

Exemplos básicos para AWS Health usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Health com AWS SDKs.

Exemplos

- Ações para AWS Health usar AWS SDKs
  - Usar DescribeEvents com uma CLI

Ações para AWS Health usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Health ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Health.

Exemplos

- Usar DescribeEvents com uma CLI

Usar **`DescribeEvents`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeEvents`.

CLI

AWS CLI

Exemplo 1: Para listar eventos de AWS Health

O `describe-events` exemplo a seguir lista eventos recentes AWS de Health.

```
aws health describe-events \
```

```
    --region us-east-1
```

Saída:

```
{
    "events": [
        {
            "arn": "arn:aws:health:us-west-1::event/ECS/
AWS_ECS_OPERATIONAL_ISSUE/AWS_ECS_OPERATIONAL_ISSUE_KWQPY_EXAMPLE111",
            "service": "ECS",
            "eventTypeCode": "AWS_ECS_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-west-1",
            "startTime": 1589077890.53,
            "endTime": 1589086345.597,
            "lastUpdatedTime": 1589086345.905,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:global::event/BILLING/
AWS_BILLING_NOTIFICATION/AWS_BILLING_NOTIFICATION_6ce1d874-e995-40e2-99cd-
EXAMPLE1118b",
            "service": "BILLING",
            "eventTypeCode": "AWS_BILLING_NOTIFICATION",
            "eventTypeCategory": "accountNotification",
            "region": "global",
            "startTime": 1588356000.0,
            "lastUpdatedTime": 1588356524.358,
            "statusCode": "open",
            "eventScopeCode": "ACCOUNT_SPECIFIC"
        },
        {
            "arn": "arn:aws:health:us-west-2::event/
CLOUDFORMATION/AWS_CLOUDFORMATION_OPERATIONAL_ISSUE/
AWS_CLOUDFORMATION_OPERATIONAL_ISSUE_OHTWY_EXAMPLE111",
            "service": "CLOUDFORMATION",
            "eventTypeCode": "AWS_CLOUDFORMATION_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-west-2",
            "startTime": 1588279630.761,
            "endTime": 1588284650.0,
            "lastUpdatedTime": 1588284691.941,
```

```
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:ap-northeast-1::event/LAMBDA/
AWS_LAMBDA_OPERATIONAL_ISSUE/AWS_LAMBDA_OPERATIONAL_ISSUE_JZDND_EXAMPLE111",
            "service": "LAMBDA",
            "eventTypeCode": "AWS_LAMBDA_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "ap-northeast-1",
            "startTime": 1587379534.08,
            "endTime": 1587391771.0,
            "lastUpdatedTime": 1587395689.316,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:us-east-1::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_COBXJ_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-east-1",
            "startTime": 1586473044.284,
            "endTime": 1586479706.091,
            "lastUpdatedTime": 1586479706.153,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:global::event/SECURITY/
AWS_SECURITY_NOTIFICATION/AWS_SECURITY_NOTIFICATION_42007387-8129-42da-8c88-
EXAMPLE11139",
            "service": "SECURITY",
            "eventTypeCode": "AWS_SECURITY_NOTIFICATION",
            "eventTypeCategory": "accountNotification",
            "region": "global",
            "startTime": 1585674000.0,
            "lastUpdatedTime": 1585674004.132,
            "statusCode": "open",
            "eventScopeCode": "PUBLIC"
        },
        {
```

```
            "arn": "arn:aws:health:global::event/CLOUDFRONT/
AWS_CLOUDFRONT_OPERATIONAL_ISSUE/
AWS_CLOUDFRONT_OPERATIONAL_ISSUE_FRQXG_EXAMPLE111",
            "service": "CLOUDFRONT",
            "eventTypeCode": "AWS_CLOUDFRONT_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "global",
            "startTime": 1585610898.589,
            "endTime": 1585617671.0,
            "lastUpdatedTime": 1585620638.869,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:us-east-1::event/SES/
AWS_SES_OPERATIONAL_ISSUE/AWS_SES_OPERATIONAL_ISSUE_URNDF_EXAMPLE111",
            "service": "SES",
            "eventTypeCode": "AWS_SES_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-east-1",
            "startTime": 1585342008.46,
            "endTime": 1585344017.0,
            "lastUpdatedTime": 1585344355.989,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:global::event/IAM/
AWS_IAM_OPERATIONAL_NOTIFICATION/
AWS_IAM_OPERATIONAL_NOTIFICATION_b6771c34-6ecd-4aea-9d3e-EXAMPLE1117e",
            "service": "IAM",
            "eventTypeCode": "AWS_IAM_OPERATIONAL_NOTIFICATION",
            "eventTypeCategory": "accountNotification",
            "region": "global",
            "startTime": 1584978300.0,
            "lastUpdatedTime": 1584978553.572,
            "statusCode": "open",
            "eventScopeCode": "ACCOUNT_SPECIFIC"
        },
        {
            "arn": "arn:aws:health:ap-southeast-2::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_HNGHE_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
```

```
            "eventTypeCategory": "issue",
            "region": "ap-southeast-2",
            "startTime": 1583881487.483,
            "endTime": 1583885056.785,
            "lastUpdatedTime": 1583885057.052,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        }
    ]
}
```

Para obter mais informações, consulte Introdução ao AWS Personal Health Dashboard no AWS Health User Guide.

Exemplo 2: Para listar eventos de AWS Health por serviço e código de status do evento

O `describe-events` exemplo a seguir lista eventos de AWS saúde para o Amazon Elastic Compute Cloud (Amazon EC2) em que o status do evento está fechado.

```
aws health describe-events \
    --filter "services=EC2,eventStatusCodes=closed"
```

Saída:

```
{
    "events": [
        {
            "arn": "arn:aws:health:us-east-1::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_VKTXI_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-east-1",
            "startTime": 1587462325.096,
            "endTime": 1587464204.774,
            "lastUpdatedTime": 1587464204.865,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:us-east-1::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_COBXJ_EXAMPLE111",
            "service": "EC2",
```

```
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-east-1",
            "startTime": 1586473044.284,
            "endTime": 1586479706.091,
            "lastUpdatedTime": 1586479706.153,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:ap-southeast-2::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_HNGHE_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "ap-southeast-2",
            "startTime": 1583881487.483,
            "endTime": 1583885056.785,
            "lastUpdatedTime": 1583885057.052,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        }
    ]
}
```

Para obter mais informações, consulte Introdução ao AWS Personal Health Dashboard no AWS Health User Guide.

- Para obter detalhes da API, consulte DescribeEventsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse comando retorna eventos do AWS Personal Health Dashboard. O usuário adiciona o parâmetro -Region para ver os eventos disponíveis para o serviço na região Leste dos EUA (Norte da Virgínia), mas o Filter_Region parâmetro - filtra os eventos registrados nas regiões UE (Londres) e Oeste dos EUA (Oregon) (eu-west-2 e us-west-2). O Filter_StartTime parâmetro - filtra por um intervalo de vezes em que os eventos podem começar, enquanto o Filter_EndTime parâmetro - filtra por um intervalo de vezes em que os eventos podem terminar. O resultado é um evento de manutenção programada para o RDS que começa

dentro do Filter_StartTime intervalo especificado e termina dentro do Filter_EndTime intervalo programado.

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

Saída:

```
Arn               : arn:aws:health:us-west-2::event/RDS/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321
AvailabilityZone  :
EndTime           : 3/21/2019 2:00:00 PM
EventTypeCategory : scheduledChange
EventTypeCode     : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime   : 2/28/2019 2:26:07 PM
Region            : us-west-2
Service           : RDS
StartTime         : 3/14/2019 2:00:00 PM
StatusCode        : open
```

- Para obter detalhes da API, consulte DescribeEventsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna eventos do AWS Personal Health Dashboard. O usuário adiciona o parâmetro -Region para ver os eventos disponíveis para o serviço na região Leste dos EUA (Norte da Virgínia), mas o Filter_Region parâmetro - filtra os eventos registrados nas regiões UE (Londres) e Oeste dos EUA (Oregon) (eu-west-2 e us-west-2). O Filter_StartTime parâmetro - filtra por um intervalo de vezes em que os eventos podem começar, enquanto o Filter_EndTime parâmetro - filtra por um intervalo de vezes em que os eventos podem terminar. O resultado é um evento de manutenção programada para o RDS que começa dentro do Filter_StartTime intervalo especificado e termina dentro do Filter_EndTime intervalo programado.

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

Saída:

```
Arn               : arn:aws:health:us-west-2::event/RDS/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321
AvailabilityZone  :
EndTime           : 3/21/2019 2:00:00 PM
EventTypeCategory : scheduledChange
EventTypeCode     : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime   : 2/28/2019 2:26:07 PM
Region            : us-west-2
Service           : RDS
StartTime         : 3/14/2019 2:00:00 PM
StatusCode        : open
```

- Para obter detalhes da API, consulte DescribeEventsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para HealthImaging usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS HealthImaging com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- HealthImaging Guia do desenvolvedor — Mais informações sobre HealthImaging.
- HealthImaging Referência da API — Detalhes sobre todas as HealthImaging ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de HealthImaging uso AWS SDKs
  - Olá HealthImaging
  - Ações para HealthImaging usar AWS SDKs
    - Use CopyImageSetcom um AWS SDK ou CLI
    - Use CreateDatastorecom um AWS SDK ou CLI
    - Use DeleteDatastorecom um AWS SDK ou CLI
    - Use DeleteImageSetcom um AWS SDK ou CLI
    - Use GetDICOMImportJobcom um AWS SDK ou CLI
    - Use GetDatastorecom um AWS SDK ou CLI
    - Use GetImageFramecom um AWS SDK ou CLI
    - Use GetImageSetcom um AWS SDK ou CLI
    - Use GetImageSetMetadatacom um AWS SDK ou CLI
    - Use ListDICOMImportJobscom um AWS SDK ou CLI
    - Use ListDatastorescom um AWS SDK ou CLI
    - Use ListImageSetVersionscom um AWS SDK ou CLI
    - Use ListTagsForResourcecom um AWS SDK ou CLI
    - Use SearchImageSetscom um AWS SDK ou CLI
    - Use StartDICOMImportJobcom um AWS SDK ou CLI
    - Use TagResourcecom um AWS SDK ou CLI
    - Use UntagResourcecom um AWS SDK ou CLI
    - Use UpdateImageSetMetadatacom um AWS SDK ou CLI
- Cenários para HealthImaging usar AWS SDKs
  - Comece com conjuntos de HealthImaging imagens e molduras de imagem usando um AWS SDK
  - Marcando um armazenamento HealthImaging de dados usando um AWS SDK
  - Marcar um conjunto HealthImaging de imagens usando um AWS SDK

Exemplos básicos de HealthImaging uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS HealthImaging com AWS SDKs.

Exemplos

- Olá HealthImaging
- Ações para HealthImaging usar AWS SDKs
  - Use CopyImageSetcom um AWS SDK ou CLI
  - Use CreateDatastorecom um AWS SDK ou CLI
  - Use DeleteDatastorecom um AWS SDK ou CLI
  - Use DeleteImageSetcom um AWS SDK ou CLI
  - Use GetDICOMImportJobcom um AWS SDK ou CLI
  - Use GetDatastorecom um AWS SDK ou CLI
  - Use GetImageFramecom um AWS SDK ou CLI
  - Use GetImageSetcom um AWS SDK ou CLI
  - Use GetImageSetMetadatacom um AWS SDK ou CLI
  - Use ListDICOMImportJobscom um AWS SDK ou CLI
  - Use ListDatastorescom um AWS SDK ou CLI
  - Use ListImageSetVersionscom um AWS SDK ou CLI
  - Use ListTagsForResourcecom um AWS SDK ou CLI
  - Use SearchImageSetscom um AWS SDK ou CLI
  - Use StartDICOMImportJobcom um AWS SDK ou CLI
  - Use TagResourcecom um AWS SDK ou CLI
  - Use UntagResourcecom um AWS SDK ou CLI
  - Use UpdateImageSetMetadatacom um AWS SDK ou CLI

Olá HealthImaging

O exemplo de código a seguir mostra como começar a usar o HealthImaging.

C++

SDK para C++

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS medical-imaging)

# Set this project's name.
project("hello_health-imaging")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executable location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()
```

```
add_executable(${PROJECT_NAME}
        hello_health_imaging.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_health_imaging.cpp.

```
#include <aws/core/Aws.h>
#include <aws/medical-imaging/MedicalImagingClient.h>
#include <aws/medical-imaging/model/ListDatastoresRequest.h>

#include <iostream>

/*
 *  A "Hello HealthImaging" starter application which initializes an AWS
 HealthImaging (HealthImaging) client
 *  and lists the HealthImaging data stores in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_health-imaging'
 *
 */
#include <aws/core/auth/AWSCredentialsProviderChain.h>
#include <aws/core/platform/Environment.h>

int main(int argc, char **argv) {
    (void) argc;
    (void) argv;
    Aws::SDKOptions options;
    //   Optional: change the log level for debugging.
    //   options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;

    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::MedicalImaging::MedicalImagingClient
 medicalImagingClient(clientConfig);
```

```
        Aws::MedicalImaging::Model::ListDatastoresRequest listDatastoresRequest;

        Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
allDataStoreSummaries;
        Aws::String nextToken; // Used for paginated results.
        do {
            if (!nextToken.empty()) {
                listDatastoresRequest.SetNextToken(nextToken);
            }
            Aws::MedicalImaging::Model::ListDatastoresOutcome
listDatastoresOutcome =
                    medicalImagingClient.ListDatastores(listDatastoresRequest);
            if (listDatastoresOutcome.IsSuccess()) {
                const Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
&dataStoreSummaries =

listDatastoresOutcome.GetResult().GetDatastoreSummaries();
                allDataStoreSummaries.insert(allDataStoreSummaries.cend(),
                                             dataStoreSummaries.cbegin(),
                                             dataStoreSummaries.cend());
                nextToken = listDatastoresOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "ListDatastores error: "
                          << listDatastoresOutcome.GetError().GetMessage() <<
std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allDataStoreSummaries.size() << " HealthImaging data "
                  << ((allDataStoreSummaries.size() == 1) ?
                      "store was retrieved." : "stores were retrieved.") <<
std::endl;

        for (auto const &dataStoreSummary: allDataStoreSummaries) {
            std::cout << "  Datastore: " << dataStoreSummary.GetDatastoreName()
                      << std::endl;
            std::cout << "  Datastore ID: " << dataStoreSummary.GetDatastoreId()
                      << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
```

```
    return 0;
}
```

- Para obter detalhes da API, consulte ListDatastoresa Referência AWS SDK para C++ da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import {
  ListDatastoresCommand,
  MedicalImagingClient,
} from "@aws-sdk/client-medical-imaging";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new MedicalImagingClient({});

export const helloMedicalImaging = async () => {
  const command = new ListDatastoresCommand({});

  const { datastoreSummaries } = await client.send(command);
  console.log("Datastores: ");
  console.log(datastoreSummaries.map((item) => item.datastoreName).join("\n"));
  return datastoreSummaries;
};
```

- Para obter detalhes da API, consulte ListDatastoresa Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def hello_medical_imaging(medical_imaging_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS HealthImaging
    client and list the data stores in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param medical_imaging_client: A Boto3 AWS HealthImaging Client object.
    """
    print("Hello, Amazon Health Imaging! Let's list some of your data stores:\n")
    try:
        paginator = medical_imaging_client.get_paginator("list_datastores")
        page_iterator = paginator.paginate()
        datastore_summaries = []
        for page in page_iterator:
            datastore_summaries.extend(page["datastoreSummaries"])
        print("\tData Stores:")
        for ds in datastore_summaries:
            print(f"\t\tDatastore: {ds['datastoreName']} ID {ds['datastoreId']}")
    except ClientError as err:
        logger.error(
            "Couldn't list data stores. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
```

```
        raise


if __name__ == "__main__":
    hello_medical_imaging(boto3.client("medical-imaging"))
```

- Para obter detalhes da API, consulte a ListDatastoresReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Ações para HealthImaging usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar HealthImaging ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a HealthImaging API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para HealthImaging usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS HealthImaging.

Exemplos

- Use CopyImageSetcom um AWS SDK ou CLI
- Use CreateDatastorecom um AWS SDK ou CLI
- Use DeleteDatastorecom um AWS SDK ou CLI
- Use DeleteImageSetcom um AWS SDK ou CLI
- Use GetDICOMImportJobcom um AWS SDK ou CLI
- Use GetDatastorecom um AWS SDK ou CLI
- Use GetImageFramecom um AWS SDK ou CLI
- Use GetImageSetcom um AWS SDK ou CLI

- Use GetImageSetMetadatacom um AWS SDK ou CLI
- Use ListDICOMImportJobscom um AWS SDK ou CLI
- Use ListDatastorescom um AWS SDK ou CLI
- Use ListImageSetVersionscom um AWS SDK ou CLI
- Use ListTagsForResourcecom um AWS SDK ou CLI
- Use SearchImageSetscom um AWS SDK ou CLI
- Use StartDICOMImportJobcom um AWS SDK ou CLI
- Use TagResourcecom um AWS SDK ou CLI
- Use UntagResourcecom um AWS SDK ou CLI
- Use UpdateImageSetMetadatacom um AWS SDK ou CLI

Use **`CopyImageSet`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CopyImageSet`.

CLI

AWS CLI

Exemplo 1: para copiar um conjunto de imagens sem um destino.

O exemplo `copy-image-set` a seguir cria uma cópia duplicada de um conjunto de imagens sem um destino.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" } }'
```

Saída:

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042357.432,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
```

```
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042357.432,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Exemplo 2: para copiar um conjunto de imagens com um destino.

O exemplo `copy-image-set` a seguir cria uma cópia duplicada de um conjunto de imagens com um destino.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" },
 "destinationImageSet": { "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
 "latestVersionId": "1"} }'
```

Saída:

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
```

```
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Exemplo 3: copiar um subconjunto de instâncias de um conjunto de imagens de origem para um conjunto de imagens de destino.

O exemplo `copy-image-set` a seguir copia uma instância DICOM do conjunto de imagens de origem para o conjunto de imagens de destino. O parâmetro force é fornecido para substituir inconsistências nos atributos dos níveis Patient, Study e Series.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet":
 {"latestVersionId": "1","DICOMCopies": {"copiableAttributes":
 "{\"SchemaVersion\":\"1.1\",\"Study\":{\"Series\":
{\"1.3.6.1.4.1.5962.99.1.3673257865.2104868982.1369432891697.3666.0\":
{\"Instances\":
{\"1.3.6.1.4.1.5962.99.1.3673257865.2104868982.1369432891697.3669.0\":
{}}}}}}"}},"destinationImageSet": {"imageSetId":
 "b9eb50d8ee682eb9fcf4acbf92f62bb7","latestVersionId": "1"}}' \
    --force
```

Saída:

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9eb50d8ee682eb9fcf4acbf92f62bb7",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
```

```
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Para obter mais informações, consulte Copiar um conjunto de imagens no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte CopyImageSetem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   /**
    * Copy an AWS HealthImaging image set.
    *
    * @param medicalImagingClient  - The AWS HealthImaging client object.
    * @param datastoreId           - The datastore ID.
    * @param imageSetId            - The image set ID.
    * @param latestVersionId       - The version ID.
    * @param destinationImageSetId - The optional destination image set ID,
ignored if null.
    * @param destinationVersionId  - The optional destination version ID,
ignored if null.
    * @param force                 - The force flag.
    * @param subsets               - The optional subsets to copy, ignored if
null.
    * @return                      - The image set ID of the copy.
    * @throws MedicalImagingException - Base exception for all service
exceptions thrown by AWS HealthImaging.
    */
   public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
                                            String datastoreId,
                                            String imageSetId,
                                            String latestVersionId,
                                            String destinationImageSetId,
                                            String destinationVersionId,
                                            boolean force,
                                            Vector<String> subsets) {

       try {
```

```
            CopySourceImageSetInformation.Builder copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
                    .latestVersionId(latestVersionId);

            // Optionally copy a subset of image instances.
            if (subsets != null) {
                String subsetInstanceToCopy =
getCopiableAttributesJSON(imageSetId, subsets);

copySourceImageSetInformation.dicomCopies(MetadataCopies.builder()
                        .copiableAttributes(subsetInstanceToCopy)
                        .build());
            }

            CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                    .sourceImageSet(copySourceImageSetInformation.build());

            // Optionally designate a destination image set.
            if (destinationImageSetId != null) {
                copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                        .imageSetId(destinationImageSetId)
                        .latestVersionId(destinationVersionId)
                        .build());
            }

            CopyImageSetRequest copyImageSetRequest =
CopyImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .force(force)
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Função utilitária para criar atributos copiáveis.

```
   /**
    * Create a JSON string of copiable image instances.
    *
    * @param imageSetId - The image set ID.
    * @param subsets    - The subsets to copy.
    * @return A JSON string of copiable image instances.
    */
   private static String getCopiableAttributesJSON(String imageSetId,
Vector<String> subsets) {
       StringBuilder subsetInstanceToCopy = new StringBuilder(
               """
                       {
                         "SchemaVersion": 1.1,
                         "Study": {
                           "Series": {
                               "
                                """
       );

       subsetInstanceToCopy.append(imageSetId);

       subsetInstanceToCopy.append(
               """
                               ": {
                               "Instances": {
                       """
       );

       for (String subset : subsets) {
           subsetInstanceToCopy.append('"' + subset + "\": {},");
       }
       subsetInstanceToCopy.deleteCharAt(subsetInstanceToCopy.length() - 1);
       subsetInstanceToCopy.append("""
                        }
                      }
                    }
                  }
               }
```

```
                """);
        return subsetInstanceToCopy.toString();
    }
```

- Para obter detalhes da API, consulte CopyImageSeta Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

Função de utilitário para copiar um conjunto de imagens.

```
import { CopyImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The source image set ID.
 * @param {string} sourceVersionId - The source version ID.
 * @param {string} destinationImageSetId - The optional ID of the destination
 image set.
 * @param {string} destinationVersionId - The optional version ID of the
 destination image set.
 * @param {boolean} force - Force the copy action.
 * @param {[string]} copySubsets - A subset of instance IDs to copy.
 */
export const copyImageSet = async (
  datastoreId = "xxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
  sourceVersionId = "1",
  destinationImageSetId = "",
  destinationVersionId = "",
  force = false,
  copySubsets = [],
```

```
) => {
  try {
    const params = {
      datastoreId: datastoreId,
      sourceImageSetId: imageSetId,
      copyImageSetInformation: {
        sourceImageSet: { latestVersionId: sourceVersionId },
      },
      force: force,
    };
    if (destinationImageSetId !== "" && destinationVersionId !== "") {
      params.copyImageSetInformation.destinationImageSet = {
        imageSetId: destinationImageSetId,
        latestVersionId: destinationVersionId,
      };
    }

    if (copySubsets.length > 0) {
      let copySubsetsJson;
      copySubsetsJson = {
        SchemaVersion: 1.1,
        Study: {
          Series: {
            imageSetId: {
              Instances: {},
            },
          },
        },
      };

      for (let i = 0; i < copySubsets.length; i++) {
        copySubsetsJson.Study.Series.imageSetId.Instances[copySubsets[i]] = {};
      }

      params.copyImageSetInformation.dicomCopies = copySubsetsJson;
    }

    const response = await medicalImagingClient.send(
      new CopyImageSetCommand(params),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //         httpStatusCode: 200,
```

```
    //         requestId: 'd9b219ce-cc48-4a44-a5b2-c5c3068f1ee8',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    //      },
    //       datastoreId: 'xxxxxxxxxxxxxx',
    //       destinationImageSetProperties: {
    //             createdAt: 2023-09-27T19:46:21.824Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxxx:datastore/xxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING',
    //             latestVersionId: '1',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //       },
    //       sourceImageSetProperties: {
    //             createdAt: 2023-09-22T14:49:26.427Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxxx:datastore/xxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING_WITH_READ_ONLY_ACCESS',
    //             latestVersionId: '4',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //      }
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Copiar um conjunto de imagens sem um destino.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
  );
```

Copiar um conjunto de imagens com um destino.

```
await copyImageSet(
  "12345678901234567890123456789012",
  "12345678901234567890123456789012",
  "1",
  "12345678901234567890123456789012",
  "1",
  false,
);
```

Copie um subconjunto de um conjunto de imagens com um destino e force a cópia.

```
await copyImageSet(
  "12345678901234567890123456789012",
  "12345678901234567890123456789012",
  "1",
  "12345678901234567890123456789012",
  "1",
  true,
  ["12345678901234567890123456789012", "11223344556677889900112233445566"],
);
```

- Para obter detalhes da API, consulte CopyImageSeta Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

Função de utilitário para copiar um conjunto de imagens.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def copy_image_set(
        self,
        datastore_id,
        image_set_id,
        version_id,
        destination_image_set_id=None,
        destination_version_id=None,
        force=False,
        subsets=[],
    ):
        """
        Copy an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param destination_image_set_id: The ID of the optional destination image
 set.
        :param destination_version_id: The ID of the optional destination image
 set version.
        :param force: Force the copy.
        :param subsets: The optional subsets to copy. For example:
 ["12345678901234567890123456789012"].
        :return: The copied image set ID.
        """
        try:
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }
            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
```

```
                }
            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"]
[
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }
            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't copy image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return copy_results["destinationImageSetProperties"]["imageSetId"]
```

Copiar um conjunto de imagens sem um destino.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
```

```
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Copiar um conjunto de imagens com um destino.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Copiar um subconjunto de um conjunto de imagens.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"]
[
                        subset
                    ] = {}
```

```
                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a CopyImageSetReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_source_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_source_version_id = '1'
        " iv_destination_image_set_id =
'12345678901234567890123456789012345678 90' (optional)
        " iv_destination_version_id = '1' (optional)
        " iv_force = abap_false
        DATA(lo_source_info) = NEW /aws1/cl_migcpsrcimagesetinf00(
          iv_latestversionid = iv_source_version_id ).
        DATA(lo_copy_info) = NEW /aws1/cl_migcpimagesetinfmtion(
```

```
          io_sourceimageset = lo_source_info ).
        IF iv_destination_image_set_id IS NOT INITIAL AND
           iv_destination_version_id IS NOT INITIAL.
          DATA(lo_dest_info) = NEW /aws1/cl_migcopydstimageset(
            iv_imagesetid = iv_destination_image_set_id
            iv_latestversionid = iv_destination_version_id ).
          lo_copy_info = NEW /aws1/cl_migcpimagesetinfmtion(
            io_sourceimageset = lo_source_info
            io_destinationimageset = lo_dest_info ).
        ENDIF.
        oo_result = lo_mig->copyimageset(
          iv_datastoreid = iv_datastore_id
          iv_sourceimagesetid = iv_source_image_set_id
          io_copyimagesetinformation = lo_copy_info
          iv_force = iv_force ).
        DATA(lo_dest_props) = oo_result->get_dstimagesetproperties( ).
        DATA(lv_new_id) = lo_dest_props->get_imagesetid( ).
        MESSAGE |Image set copied with new ID: { lv_new_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CopyImageSetreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`CreateDatastore`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateDatastore`.

Bash

AWS CLI com script Bash

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function imaging_create_datastore
#
# This function creates an AWS HealthImaging data store for importing DICOM P10
 files.
#
# Parameters:
#       -n data_store_name - The name of the data store.
#
# Returns:
#       The datastore ID.
#    And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function imaging_create_datastore() {
  local datastore_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_create_datastore"
    echo "Creates an AWS HealthImaging data store for importing DICOM P10 files."
    echo "  -n data_store_name - The name of the data store."
    echo ""
  }
```

```
  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) datastore_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_name" ]]; then
    errecho "ERROR: You must provide a data store name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging create-datastore \
    --datastore-name "$datastore_name" \
    --output text \
    --query 'datastoreId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging create-datastore operation
 failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte CreateDatastoreem Referência de AWS CLI Comandos.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Exemplo 1: criar um datastore

O exemplo de código `create-datastore` a seguir cria um armazenamento de dados com o nome de `my-datastore`. Quando você cria um armazenamento de dados sem especificar um`--lossless-storage-format`, o AWS HealthImaging padrão é HTJ2K (High Throughput JPEG 2000).

```
aws medical-imaging create-datastore \
    --datastore-name "my-datastore"
```

Saída:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "CREATING"
}
```

Exemplo 2: criar um datastore com o formato de armazenamento JPEG 2000 Lossless

Um datastore configurado com o formato de armazenamento JPEG 2000 Lossless transcodificará e persistirá quadros de imagem sem perdas no formato JPEG 2000. Os quadros de imagem podem então ser recuperados no JPEG 2000 Lossless sem transcodificação. O exemplo de código `create-datastore` a seguir cria um datastore configurado para o formato de armazenamento JPEG 2000 Lossless com o nome `my-datastore`.

```
aws medical-imaging create-datastore \
```

```
    --datastore-name "my-datastore" \
    --lossless-storage-format JPEG_2000_LOSSLESS
```

Saída:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "CREATING"
}
```

Para obter mais informações, consulte Criação de um armazenamento de dados no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte CreateDatastoreem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
           String datastoreName) {
       try {
           CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
                   .datastoreName(datastoreName)
                   .build();
           CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
           return response.datastoreId();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return "";
   }
```

- Para obter detalhes da API, consulte CreateDatastorea Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { CreateDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreName - The name of the data store to create.
 */
export const createDatastore = async (datastoreName = "DATASTORE_NAME") => {
  const response = await medicalImagingClient.send(
    new CreateDatastoreCommand({ datastoreName: datastoreName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: 'a71cd65f-2382-49bf-b682-f9209d8d399b',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //    datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //    datastoreStatus: 'CREATING'
  // }
  return response;
};
```

- Para obter detalhes da API, consulte CreateDatastorea Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def create_datastore(self, name):
        """
        Create a data store.

        :param name: The name of the data store to create.
        :return: The data store ID.
        """
        try:
            data_store =
 self.health_imaging_client.create_datastore(datastoreName=name)
        except ClientError as err:
            logger.error(
                "Couldn't create data store %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreId"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a CreateDatastoreReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
   TRY.
       " iv_datastore_name = 'my-datastore-name'
       oo_result = lo_mig->createdatastore( iv_datastorename =
iv_datastore_name ).
       DATA(lv_datastore_id) = oo_result->get_datastoreid( ).
       MESSAGE 'Data store created.' TYPE 'I'.
     CATCH /aws1/cx_migaccessdeniedex.
       MESSAGE 'Access denied.' TYPE 'I'.
     CATCH /aws1/cx_migconflictexception.
       MESSAGE 'Conflict. Data store may already exist.' TYPE 'I'.
     CATCH /aws1/cx_miginternalserverex.
       MESSAGE 'Internal server error.' TYPE 'I'.
     CATCH /aws1/cx_migservicequotaexcdex.
       MESSAGE 'Service quota exceeded.' TYPE 'I'.
     CATCH /aws1/cx_migthrottlingex.
       MESSAGE 'Request throttled.' TYPE 'I'.
     CATCH /aws1/cx_migvalidationex.
       MESSAGE 'Validation error.' TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDatastorereferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`DeleteDatastore`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteDatastore`.

Bash

AWS CLI com script Bash

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function imaging_delete_datastore
#
# This function deletes an AWS HealthImaging data store.
#
# Parameters:
#       -i datastore_id - The ID of the data store.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function imaging_delete_datastore() {
  local datastore_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_delete_datastore"
    echo "Deletes an AWS HealthImaging data store."
```

```
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging delete-datastore \
    --datastore-id "$datastore_id")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging delete-datastore operation
 failed.$response"
    return 1
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteDatastoreem Referência de AWS CLI Comandos.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Para excluir um armazenamento de dados

O exemplo de código `delete-datastore` a seguir exclui um armazenamento de dados.

```
aws medical-imaging delete-datastore \
    --datastore-id "12345678901234567890123456789012"
```

Saída:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "DELETING"
}
```

Para obter mais informações, consulte Excluindo um armazenamento de dados no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte DeleteDatastoreem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
    public static void deleteMedicalImagingDatastore(MedicalImagingClient
 medicalImagingClient,
            String datastoreID) {
        try {
```

```
            DeleteDatastoreRequest datastoreRequest =
 DeleteDatastoreRequest.builder()
                    .datastoreId(datastoreID)
                    .build();
            medicalImagingClient.deleteDatastore(datastoreRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteDatastorea Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { DeleteDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store to delete.
 */
export const deleteDatastore = async (datastoreId = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new DeleteDatastoreCommand({ datastoreId }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //           httpStatusCode: 200,
  //           requestId: 'f5beb409-678d-48c9-9173-9a001ee1ebb1',
  //           extendedRequestId: undefined,
  //           cfId: undefined,
```

```
  //           attempts: 1,
  //           totalRetryDelay: 0
  //        },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     datastoreStatus: 'DELETING'
  // }

  return response;
};
```

- Para obter detalhes da API, consulte DeleteDatastorea Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_datastore(self, datastore_id):
        """
        Delete a data store.

        :param datastore_id: The ID of the data store.
        """
        try:
            self.health_imaging_client.delete_datastore(datastoreId=datastore_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete data store %s. Here's why: %s: %s",
                datastore_id,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a DeleteDatastoreReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678 90'
    oo_result = lo_mig->deletedatastore( iv_datastoreid = iv_datastore_id ).
    MESSAGE 'Data store deleted.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict. Data store may contain resources.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Data store not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDatastorereferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`DeleteImageSet`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteImageSet`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Começar a usar conjuntos de imagens e quadros de imagem

C++

SDK para C++

```
//! Routine which deletes an AWS HealthImaging image set.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::deleteImageSet(
        const Aws::String &dataStoreID, const Aws::String &imageSetID,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::DeleteImageSetRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    Aws::MedicalImaging::Model::DeleteImageSetOutcome outcome =
 client.DeleteImageSet(
            request);
    if (outcome.IsSuccess()) {
```

```
        std::cout << "Successfully deleted image set " << imageSetID
                  << " from data store " << dataStoreID << std::endl;
    }
    else {
        std::cerr << "Error deleting image set " << imageSetID << " from data
 store "
                  << dataStoreID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteImageSeta Referência AWS SDK para C++ da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Para excluir um conjunto de imagens

O exemplo de código `delete-image-set` a seguir exclui um conjunto de imagens.

```
aws medical-imaging delete-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

Saída:

```
{
    "imageSetWorkflowStatus": "DELETING",
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "datastoreId": "12345678901234567890123456789012"
```

```
}
```

Para obter mais informações, consulte Excluir um conjunto de imagens no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte DeleteImageSetem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String imagesetId) {
       try {
           DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId)
                   .build();

           medicalImagingClient.deleteImageSet(deleteImageSetRequest);

           System.out.println("The image set was deleted.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DeleteImageSeta Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { DeleteImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store ID.
 * @param {string} imageSetId - The image set ID.
 */
export const deleteImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new DeleteImageSetCommand({
      datastoreId: datastoreId,
      imageSetId: imageSetId,
    }),
  );
  console.log(response);
  // {
  //    '$metadata': {
  //        httpStatusCode: 200,
  //        requestId: '6267bbd2-eaa5-4a50-8ee8-8fddf535cf73',
  //        extendedRequestId: undefined,
  //        cfId: undefined,
  //        attempts: 1,
  //        totalRetryDelay: 0
  //     },
  //     datastoreId: 'xxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'LOCKED',
  //     imageSetWorkflowStatus: 'DELETING'
  // }
  return response;
};
```

- Para obter detalhes da API, consulte DeleteImageSeta Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The delete results.
        """
        try:
            delete_results = self.health_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delete_results
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
```

```
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a DeleteImageSetReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->deleteimageset(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id ).
        MESSAGE 'Image set deleted.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteImageSetreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`GetDICOMImportJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetDICOMImportJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Começar a usar conjuntos de imagens e quadros de imagem

C++

SDK para C++

```
//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome;
}
```

- Para obter detalhes da API, consulte GetDICOMImportJoba Referência AWS SDK para C++ da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Para obter as propriedades de um trabalho de importação dicom

O exemplo de código `get-dicom-import-job` a seguir obtém as propriedades de um trabalho de importação dicom.

```
aws medical-imaging get-dicom-import-job \
    --datastore-id "12345678901234567890123456789012" \
    --job-id "09876543210987654321098765432109"
```

Saída:

```
{
    "jobProperties": {
        "jobId": "09876543210987654321098765432109",
        "jobName": "my-job",
        "jobStatus": "COMPLETED",
        "datastoreId": "12345678901234567890123456789012",
        "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
        "endedAt": "2022-08-12T11:29:42.285000+00:00",
        "submittedAt": "2022-08-12T11:28:11.152000+00:00",
        "inputS3Uri": "s3://medical-imaging-dicom-input/dicom_input/",
```

```
        "outputS3Uri": "s3://medical-imaging-output/
job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/"
    }
}
```

Para obter mais informações, consulte Obter propriedades do trabalho de importação no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte GetDICOMImportJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String jobId) {

       try {
           GetDicomImportJobRequest getDicomImportJobRequest =
GetDicomImportJobRequest.builder()
                   .datastoreId(datastoreId)
                   .jobId(jobId)
                   .build();
           GetDicomImportJobResponse response =
medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
           return response.jobProperties();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Para obter detalhes da API, consulte GetDICOMImportJoba Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { GetDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} jobId - The ID of the import job.
 */
export const getDICOMImportJob = async (
  datastoreId = "xxxxxxxxxxxxxxxxxxxx",
  jobId = "xxxxxxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new GetDICOMImportJobCommand({ datastoreId: datastoreId, jobId: jobId }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: 'a2637936-78ea-44e7-98b8-7a87d95dfaee',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobProperties: {
  //         dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-19T17:29:21.753Z,
  //             inputS3Uri: 's3://healthimaging-source/CTStudy/',
  //             jobId: ''xxxxxxxxxxxxxxxxxxxxxxxx'',
  //             jobName: 'job_1',
  //             jobStatus: 'COMPLETED',
```

```
  //             outputS3Uri: 's3://health-imaging-dest/
ouput_ct/'xxxxxxxxxxxxxxxxxxxxxxxx'-DicomImport-'xxxxxxxxxxxxxxxxxxxxxxxx'/',
  //             submittedAt: 2023-09-19T17:27:25.143Z
  //     }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte GetDICOMImportJoba Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_dicom_import_job(self, datastore_id, job_id):
        """
        Get the properties of a DICOM import job.

        :param datastore_id: The ID of the data store.
        :param job_id: The ID of the job.
        :return: The job properties.
        """
        try:
            job = self.health_imaging_client.get_dicom_import_job(
                jobId=job_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't get DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobProperties"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetDICOMImportJobReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678 90'
    " iv_job_id = '12345678901234567890123456789012'
    oo_result = lo_mig->getdicomimportjob(
      iv_datastoreid = iv_datastore_id
      iv_jobid = iv_job_id ).
    DATA(lo_job_props) = oo_result->get_jobproperties( ).
    DATA(lv_job_status) = lo_job_props->get_jobstatus( ).
    MESSAGE |Job status: { lv_job_status }.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
```

```
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Job not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetDICOMImportJobreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`GetDatastore`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetDatastore`.

Bash

AWS CLI com script Bash

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function imaging_get_datastore
#
# Get a data store's properties.
#
```

```
# Parameters:
#       -i data_store_id - The ID of the data store.
#
# Returns:
#       [datastore_name, datastore_id, datastore_status, datastore_arn,
 created_at, updated_at]
#    And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function imaging_get_datastore() {
  local datastore_id option OPTARG # Required to use getopts command in a
 function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_get_datastore"
    echo "Gets a data store's properties."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi
```

```
  local response

  response=$(
    aws medical-imaging get-datastore \
      --datastore-id "$datastore_id" \
      --output text \
      --query "[ datastoreProperties.datastoreName,
 datastoreProperties.datastoreId, datastoreProperties.datastoreStatus,
 datastoreProperties.datastoreArn,  datastoreProperties.createdAt,
 datastoreProperties.updatedAt]"
  )
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte GetDatastoreem Referência de AWS CLI Comandos.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Exemplo 1: obter as propriedades de um datastore

O exemplo de código `get-datastore` a seguir obtém as propriedades de um armazenamento de dados.

```
aws medical-imaging get-datastore \
    --datastore-id 12345678901234567890123456789012
```

Saída:

```
{
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "losslessStorageFormat": "HTJ2K"
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
}
```

Exemplo 2: obter as propriedades do datastore configurado para JPEG 2000

O exemplo de código `get-datastore` a seguir obtém as propriedades de um datastore configurado para o formato de armazenamento JPEG 2000 Lossless.

```
aws medical-imaging get-datastore \
    --datastore-id 12345678901234567890123456789012
```

Saída:

```
{
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "losslessStorageFormat": "JPEG_2000_LOSSLESS",
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
}
```

Para obter mais informações, consulte Obter propriedades do armazenamento de dados no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte GetDatastoreem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   public static DatastoreProperties
getMedicalImageDatastore(MedicalImagingClient medicalImagingClient,
           String datastoreID) {
       try {
           GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
                   .datastoreId(datastoreID)
                   .build();
           GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
           return response.datastoreProperties();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Para obter detalhes da API, consulte GetDatastorea Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { GetDatastoreCommand } from "@aws-sdk/client-medical-imaging";
```

```
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreID - The ID of the data store.
 */
export const getDatastore = async (datastoreID = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new GetDatastoreCommand({ datastoreId: datastoreID }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '55ea7d2e-222c-4a6a-871e-4f591f40cadb',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //   datastoreProperties: {
  //        createdAt: 2023-08-04T18:50:36.239Z,
  //         datastoreArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxx:datastore/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreName: 'my_datastore',
  //         datastoreStatus: 'ACTIVE',
  //         updatedAt: 2023-08-04T18:50:36.239Z
  //   }
  // }
  return response.datastoreProperties;
};
```

- Para obter detalhes da API, consulte GetDatastorea Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_datastore_properties(self, datastore_id):
        """
        Get the properties of a data store.

        :param datastore_id: The ID of the data store.
        :return: The data store properties.
        """
        try:
            data_store = self.health_imaging_client.get_datastore(
                datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get data store %s. Here's why: %s: %s",
                id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreProperties"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetDatastoreReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678 90'
    oo_result = lo_mig->getdatastore( iv_datastoreid = iv_datastore_id ).
    DATA(lo_properties) = oo_result->get_datastoreproperties( ).
    DATA(lv_name) = lo_properties->get_datastorename( ).
    DATA(lv_status) = lo_properties->get_datastorestatus( ).
    MESSAGE 'Data store properties retrieved.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Data store not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetDatastorereferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`GetImageFrame`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetImageFrame`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Começar a usar conjuntos de imagens e quadros de imagem

C++

SDK para C++

```
//! Routine which downloads an AWS HealthImaging image frame.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param frameID: The image frame ID.
  \param jphFile: File to store the downloaded frame.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageFrame(const Aws::String &dataStoreID,
                                            const Aws::String &imageSetID,
                                            const Aws::String &frameID,
                                            const Aws::String &jphFile,
                                            const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);

    Aws::MedicalImaging::Model::GetImageFrameRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);

    Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
    imageFrameInformation.SetImageFrameId(frameID);
    request.SetImageFrameInformation(imageFrameInformation);

    Aws::MedicalImaging::Model::GetImageFrameOutcome outcome =
 client.GetImageFrame(
            request);

    if (outcome.IsSuccess()) {
```

```
        std::cout << "Successfully retrieved image frame." << std::endl;
        auto &buffer = outcome.GetResult().GetImageFrameBlob();

        std::ofstream outfile(jphFile, std::ios::binary);
        outfile << buffer.rdbuf();
    }
    else {
        std::cout << "Error retrieving image frame." <<
 outcome.GetError().GetMessage()
                  << std::endl;

    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetImageFramea Referência AWS SDK para C++ da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Para obter dados de pixels do conjunto de imagens

O exemplo de código `get-image-frame` a seguir obtém um quadro de imagem.

```
aws medical-imaging get-image-frame \
    --datastore-id "12345678901234567890123456789012" \
    --image-set-id "98765412345612345678907890789012" \
    --image-frame-information imageFrameId=3abf5d5d7ae72f80a0ec81b2c0de3ef4 \
    imageframe.jph
```

Observação: esse exemplo de código não inclui a saída porque a GetImageFrame ação retorna um fluxo de dados de pixels para o arquivo imageframe.jph. Para obter informações

sobre decodificação e visualização de quadros de imagem, consulte as bibliotecas de decodificação HTJ2K.

Para obter mais informações, consulte Obter dados de pixels do conjunto de imagens no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte GetImageFrameem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
        public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
                        String destinationPath,
                        String datastoreId,
                        String imagesetId,
                        String imageFrameId) {

                try {
                        GetImageFrameRequest getImageSetMetadataRequest =
GetImageFrameRequest.builder()
                                        .datastoreId(datastoreId)
                                        .imageSetId(imagesetId)

.imageFrameInformation(ImageFrameInformation.builder()

.imageFrameId(imageFrameId)
                                                        .build())
                                        .build();

medicalImagingClient.getImageFrame(getImageSetMetadataRequest,

FileSystems.getDefault().getPath(destinationPath));

                        System.out.println("Image frame downloaded to " +
destinationPath);
                } catch (MedicalImagingException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
```

- Para obter detalhes da API, consulte GetImageFramea Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { GetImageFrameCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} imageFrameFileName - The name of the file for the HTJ2K-
encoded image frame.
 * @param {string} datastoreID - The data store's ID.
 * @param {string} imageSetID - The image set's ID.
 * @param {string} imageFrameID - The image frame's ID.
 */
export const getImageFrame = async (
  imageFrameFileName = "image.jph",
  datastoreID = "DATASTORE_ID",
  imageSetID = "IMAGE_SET_ID",
  imageFrameID = "IMAGE_FRAME_ID",
) => {
  const response = await medicalImagingClient.send(
    new GetImageFrameCommand({
      datastoreId: datastoreID,
      imageSetId: imageSetID,
      imageFrameInformation: { imageFrameId: imageFrameID },
    }),
  );
  const buffer = await response.imageFrameBlob.transformToByteArray();
  writeFileSync(imageFrameFileName, buffer);

  console.log(response);
```

```
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'e4ab42a5-25a3-4377-873f-374ecf4380e1',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     contentType: 'application/octet-stream',
  //     imageFrameBlob: <ref *1> IncomingMessage {}
  // }
  return response;
};
```

- Para obter detalhes da API, consulte GetImageFramea Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K
 encoded pixel data.
```

```
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.health_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetImageFrameReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '1234567890123456789012345678901234567890'
        " iv_image_set_id = '1234567890123456789012345678901234567890'
        " iv_image_frame_id = '1234567890123456789012345678901234567890'
        oo_result = lo_mig->getimageframe(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          io_imageframeinformation = NEW /aws1/cl_migimageframeinfmtion(
            iv_imageframeid = iv_image_frame_id ) ).
        DATA(lv_frame_blob) = oo_result->get_imageframeblob( ).
        MESSAGE 'Image frame retrieved.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image frame not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetImageFramereferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`GetImageSet`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetImageSet`.

CLI

AWS CLI

Para obter as propriedades do conjunto de imagens

O exemplo de código `get-image-set` a seguir obtém as propriedades de um conjunto de imagens.

```
aws medical-imaging get-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 18f88ac7870584f58d56256646b4d92b \
    --version-id 1
```

Saída:

```
{
    "versionId": "1",
    "imageSetWorkflowStatus": "COPIED",
    "updatedAt": 1680027253.471,
    "imageSetId": "18f88ac7870584f58d56256646b4d92b",
    "imageSetState": "ACTIVE",
    "createdAt": 1679592510.753,
    "datastoreId": "12345678901234567890123456789012"
}
```

Para obter mais informações, consulte Como obter propriedades do conjunto de imagens no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte GetImageSetem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
    public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
 medicalImagingClient,
            String datastoreId,
            String imagesetId,
            String versionId) {
        try {
```

```
            GetImageSetRequest.Builder getImageSetRequestBuilder =
GetImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId);

            if (versionId != null) {
                getImageSetRequestBuilder =
getImageSetRequestBuilder.versionId(versionId);
            }

            return
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Para obter detalhes da API, consulte GetImageSeta Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { GetImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 * @param {string} imageSetVersion - The optional version of the image set.
 *
```

```
 */
export const getImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxx",
  imageSetVersion = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imageSetId };
  if (imageSetVersion !== "") {
    params.imageSetVersion = imageSetVersion;
  }
  const response = await medicalImagingClient.send(
    new GetImageSetCommand(params),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '0615c161-410d-4d06-9d8c-6e1241bb0a5a',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     createdAt: 2023-09-22T14:49:26.427Z,
  //     datastoreId: 'xxxxxxxxxxxxxxx',
  //     imageSetArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'ACTIVE',
  //     imageSetWorkflowStatus: 'CREATED',
  //     updatedAt: 2023-09-22T14:49:26.427Z,
  //     versionId: '1'
  // }

  return response;
};
```

- Para obter detalhes da API, consulte GetImageSeta Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return image_set
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetImageSetReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_state) = oo_result->get_imagesetstate( ).
        MESSAGE |Image set retrieved with state: { lv_state }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
```

```
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetImageSetreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`GetImageSetMetadata`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetImageSetMetadata`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Começar a usar conjuntos de imagens e quadros de imagem

C++

SDK para C++

Função de utilitário para obter metadados do conjunto de imagens.

```
//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
```

```
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped
 json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String
 &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Obter metadados do conjunto de imagens sem versão.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
 "", outputFilePath, clientConfig))
        {
```

```
            std::cout << "Successfully retrieved image set metadata." <<
std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

Obter metadados do conjunto de imagens com versão.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
versionID, outputFilePath, clientConfig))
        {
            std::cout << "Successfully retrieved image set metadata." <<
std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

- Para obter detalhes da API, consulte GetImageSetMetadataa Referência AWS SDK para C ++ da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Exemplo 1: para obter os metadados de um conjunto de imagens sem versão

O exemplo de código `get-image-set-metadata` a seguir obtém metadados para um conjunto de imagens sem especificar uma versão.

Observação: `outfile` é um parâmetro obrigatório

```
aws medical-imaging get-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    studymetadata.json.gz
```

Os metadados retornados são compactados com o gzip e armazenados no arquivo studymetadata.json.gz. Para visualizar o conteúdo do objeto JSON retornado, você deve primeiro descompactá-lo.

Saída:

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

Exemplo 2: para obter os metadados de um conjunto de imagens com versão

O exemplo de código `get-image-set-metadata` a seguir obtém metadados para um conjunto de imagens com uma versão especificada.

Observação: `outfile` é um parâmetro obrigatório

```
aws medical-imaging get-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --version-id 1 \
    studymetadata.json.gz
```

Os metadados retornados são compactados com o gzip e armazenados no arquivo studymetadata.json.gz. Para visualizar o conteúdo do objeto JSON retornado, você deve primeiro descompactá-lo.

Saída:

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

Para obter mais informações, consulte Obter metadados do conjunto de imagens no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte GetImageSetMetadataem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
           String destinationPath,
           String datastoreId,
           String imagesetId,
           String versionId) {

       try {
           GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder
= GetImageSetMetadataRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId);

           if (versionId != null) {
               getImageSetMetadataRequestBuilder =
getImageSetMetadataRequestBuilder.versionId(versionId);
           }


medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
                   FileSystems.getDefault().getPath(destinationPath));

           System.out.println("Metadata downloaded to " + destinationPath);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte GetImageSetMetadataa Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

Função de utilitário para obter metadados do conjunto de imagens.

```
import { GetImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
import { writeFileSync } from "node:fs";

/**
 * @param {string} metadataFileName - The name of the file for the gzipped
 metadata.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imagesetId - The ID of the image set.
 * @param {string} versionID - The optional version ID of the image set.
 */
export const getImageSetMetadata = async (
  metadataFileName = "metadata.json.gzip",
  datastoreId = "xxxxxxxxxxxxxx",
  imagesetId = "xxxxxxxxxxxxxx",
  versionID = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imagesetId };

  if (versionID) {
    params.versionID = versionID;
  }

  const response = await medicalImagingClient.send(
    new GetImageSetMetadataCommand(params),
  );
  const buffer = await response.imageSetMetadataBlob.transformToByteArray();
  writeFileSync(metadataFileName, buffer);

  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '5219b274-30ff-4986-8cab-48753de3a599',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
```

```
  // },
  //     contentType: 'application/json',
  //     contentEncoding: 'gzip',
  //     imageSetMetadataBlob: <ref *1> IncomingMessage {}
  // }

  return response;
};
```

Obter metadados do conjunto de imagens sem versão.

```
  try {
    await getImageSetMetadata(
      "metadata.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

Obter metadados do conjunto de imagens com versão.

```
  try {
    await getImageSetMetadata(
      "metadata2.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
      "1",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

- Para obter detalhes da API, consulte GetImageSetMetadataa Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

Função de utilitário para obter metadados do conjunto de imagens.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """
        try:
            if version_id:
                image_set_metadata =
 self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:

                image_set_metadata =
 self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            print(image_set_metadata)
```

```
            with open(metadata_file, "wb") as f:
                for chunk in
image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Obter metadados do conjunto de imagens sem versão.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
```

Obter metadados do conjunto de imagens com versão.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetImageSetMetadataReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimagesetmetadata(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimagesetmetadata(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_metadata_blob) = oo_result->get_imagesetmetadatablob( ).
        MESSAGE 'Image set metadata retrieved.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetImageSetMetadatareferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`ListDICOMImportJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListDICOMImportJobs`.

CLI

AWS CLI

Para listar trabalhos de importação dicom

O exemplo de código `list-dicom-import-jobs` a seguir lista os trabalhos de importação dicom.

```
aws medical-imaging list-dicom-import-jobs \
    --datastore-id "12345678901234567890123456789012"
```

Saída:

```
{
    "jobSummaries": [
        {
            "jobId": "09876543210987654321098765432109",
            "jobName": "my-job",
            "jobStatus": "COMPLETED",
            "datastoreId": "12345678901234567890123456789012",
            "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
            "endedAt": "2022-08-12T11:21:56.504000+00:00",
            "submittedAt": "2022-08-12T11:20:21.734000+00:00"
        }
```

```
    ]
}
```

Para obter mais informações, consulte Listar trabalhos de importação no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte ListDICOMImportJobsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
           String datastoreId) {

       try {
           ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                   .datastoreId(datastoreId)
                   .build();
           ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
           return response.jobSummaries();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return new ArrayList<>();
   }
```

- Para obter detalhes da API, consulte ListDICOMImportJobsa Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { paginateListDICOMImportJobs } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 */
export const listDICOMImportJobs = async (
  datastoreId = "xxxxxxxxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId: datastoreId };
  const paginator = paginateListDICOMImportJobs(paginatorConfig, commandParams);

  const jobSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    jobSummaries.push(...page.jobSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '3c20c66e-0797-446a-a1d8-91b742fd15a0',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobSummaries: [
  //         {
  //             dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/
dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-22T14:49:51.351Z,
  //             jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
```

```
  //             jobName: 'test-1',
  //             jobStatus: 'COMPLETED',
  //             submittedAt: 2023-09-22T14:48:45.767Z
  // }
  // ]}

  return jobSummaries;
};
```

- Para obter detalhes da API, consulte ListDICOMImportJobsa Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_dicom_import_jobs(self, datastore_id):
        """
        List the DICOM import jobs.

        :param datastore_id: The ID of the data store.
        :return: The list of jobs.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_dicom_import_jobs"
            )
            page_iterator = paginator.paginate(datastoreId=datastore_id)
            job_summaries = []
            for page in page_iterator:
```

```
                job_summaries.extend(page["jobSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list DICOM import jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job_summaries
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a ListDICOMImportJobsReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->listdicomimportjobs( iv_datastoreid =
iv_datastore_id ).
        DATA(lt_jobs) = oo_result->get_jobsummaries( ).
        DATA(lv_count) = lines( lt_jobs ).
        MESSAGE |Found { lv_count } DICOM import jobs.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
```

```
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListDICOMImportJobsreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`ListDatastores`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListDatastores`.

Bash

AWS CLI com script Bash

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function imaging_list_datastores
#
# List the HealthImaging data stores in the account.
```

```
#
# Returns:
#       [[datastore_name, datastore_id, datastore_status]]
#    And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_list_datastores() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_list_datastores"
    echo "Lists the AWS HealthImaging data stores in the account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response
  response=$(aws medical-imaging list-datastores \
    --output text \
    --query "datastoreSummaries[*][datastoreName, datastoreId, datastoreStatus]")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi
```

```
  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte ListDatastoresem Referência de AWS CLI Comandos.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Para listar armazenamentos de dados

O exemplo de código `list-datastores` a seguir lista os armazenamentos de dados disponíveis.

```
aws medical-imaging list-datastores
```

Saída:

```
{
    "datastoreSummaries": [
        {
            "datastoreId": "12345678901234567890123456789012",
            "datastoreName": "TestDatastore123",
            "datastoreStatus": "ACTIVE",
            "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
            "createdAt": "2022-11-15T23:33:09.643000+00:00",
            "updatedAt": "2022-11-15T23:33:09.643000+00:00"
        }
    ]
}
```

Para obter mais informações, consulte Listar armazenamentos de dados no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte ListDatastoresem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
       try {
           ListDatastoresRequest datastoreRequest =
ListDatastoresRequest.builder()
                   .build();
           ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
           List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

           responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

           return datastoreSummaries;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Para obter detalhes da API, consulte ListDatastoresa Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { paginateListDatastores } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

export const listDatastores = async () => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {};
  const paginator = paginateListDatastores(paginatorConfig, commandParams);

  /**
   * @type {import("@aws-sdk/client-medical-imaging").DatastoreSummary[]}
   */
  const datastoreSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    datastoreSummaries.push(...page.datastoreSummaries);
    console.log(page);
  }
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '6aa99231-d9c2-4716-a46e-edb830116fa3',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //   },
  //   datastoreSummaries: [
  //     {
  //       createdAt: 2023-08-04T18:49:54.429Z,
  //       datastoreArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreName: 'my_datastore',
  //       datastoreStatus: 'ACTIVE',
  //       updatedAt: 2023-08-04T18:49:54.429Z
```

```
  //     }
  //     ...
  //   ]
  // }

  return datastoreSummaries;
};
```

- Para obter detalhes da API, consulte ListDatastoresa Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_datastores(self):
        """
        List the data stores.

        :return: The list of data stores.
        """
        try:
            paginator =
 self.health_imaging_client.get_paginator("list_datastores")
            page_iterator = paginator.paginate()
            datastore_summaries = []
            for page in page_iterator:
                datastore_summaries.extend(page["datastoreSummaries"])
        except ClientError as err:
            logger.error(
```

```
                "Couldn't list data stores. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return datastore_summaries
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a ListDatastoresReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
TRY.
    oo_result = lo_mig->listdatastores( ).
    DATA(lt_datastores) = oo_result->get_datastoresummaries( ).
    DATA(lv_count) = lines( lt_datastores ).
    MESSAGE |Found { lv_count } data stores.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListDatastoresreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`ListImageSetVersions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListImageSetVersions`.

CLI

AWS CLI

Para listar as versões de um conjunto de imagens

O exemplo de código `list-image-set-versions` a seguir lista o histórico de versões de um conjunto de imagens.

```
aws medical-imaging list-image-set-versions \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

Saída:

```
{
    "imageSetPropertiesList": [
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "4",
            "updatedAt": 1680029436.304,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
```

```
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "3",
            "updatedAt": 1680029163.325,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "COPY_FAILED",
            "versionId": "2",
            "updatedAt": 1680027455.944,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "message": "INVALID_REQUEST:  Series of SourceImageSet and
 DestinationImageSet don't match.",
            "createdAt": 1680027126.436
        },
        {
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "versionId": "1",
            "ImageSetWorkflowStatus": "COPIED",
            "createdAt": 1680027126.436
        }
    ]
}
```

Para obter mais informações, consulte Listar as versões do conjunto de imagens no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte ListImageSetVersionsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
    public static List<ImageSetProperties>
 listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
            String datastoreId,
            String imagesetId) {
        try {
```

```
            ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId)
                    .build();

            ListImageSetVersionsIterable responses = medicalImagingClient
                    .listImageSetVersionsPaginator(getImageSetRequest);
            List<ImageSetProperties> imageSetProperties = new ArrayList<>();
            responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

            return imageSetProperties;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Para obter detalhes da API, consulte ListImageSetVersionsa Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { paginateListImageSetVersions } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 */
```

```
export const listImageSetVersions = async (
  datastoreId = "xxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId, imageSetId };
  const paginator = paginateListImageSetVersions(
    paginatorConfig,
    commandParams,
  );

  const imageSetPropertiesList = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetPropertiesList.push(...page.imageSetPropertiesList);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '74590b37-a002-4827-83f2-3c590279c742',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetPropertiesList: [
  //         {
  //             ImageSetWorkflowStatus: 'CREATED',
  //             createdAt: 2023-09-22T14:49:26.427Z,
  //             imageSetId: 'xxxxxxxxxxxxxxxxxxxxxxx',
  //             imageSetState: 'ACTIVE',
  //             versionId: '1'
  //         }]
  // }
  return imageSetPropertiesList;
};
```

- Para obter detalhes da API, consulte ListImageSetVersionsa Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_image_set_versions(self, datastore_id, image_set_id):
        """
        List the image set versions.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The list of image set versions.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_image_set_versions"
            )
            page_iterator = paginator.paginate(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
            image_set_properties_list = []
            for page in page_iterator:
                image_set_properties_list.extend(page["imageSetPropertiesList"])
        except ClientError as err:
            logger.error(
                "Couldn't list image set versions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return image_set_properties_list
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a ListImageSetVersionsReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
TRY.
    " iv_datastore_id = '1234567890123456789012345678901234567890'
    " iv_image_set_id = '1234567890123456789012345678901234567890'
    oo_result = lo_mig->listimagesetversions(
      iv_datastoreid = iv_datastore_id
      iv_imagesetid = iv_image_set_id ).
    DATA(lt_versions) = oo_result->get_imagesetpropertieslist( ).
    DATA(lv_count) = lines( lt_versions ).
    MESSAGE |Found { lv_count } image set versions.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Image set not found.' TYPE 'I'.
```

```
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListImageSetVersionsreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`ListTagsForResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTagsForResource`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Marcar um datastore
- Marcar um conjunto de imagens

CLI

AWS CLI

Exemplo 1: para listar as tags de recurso de um armazenamento de dados

O exemplo de código `list-tags-for-resource` a seguir lista as tags de um armazenamento de dados.

```
aws medical-imaging list-tags-for-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"
```

Saída:

```
{
    "tags":{
        "Deployment":"Development"
    }
}
```

Exemplo 2: para listar tags de recurso de um conjunto de imagens

O exemplo de código `list-tags-for-resource` a seguir lista as tags de um conjunto de imagens.

```
aws medical-imaging list-tags-for-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b"
```

Saída:

```
{
    "tags":{
        "Deployment":"Development"
    }
}
```

Para obter mais informações, consulte Como marcar recursos AWS HealthImaging no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
```

```
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Para obter detalhes da API, consulte ListTagsForResourcea Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
```

```
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte ListTagsForResourcea Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
```

```
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a ListTagsForResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        oo_result = lo_mig->listtagsforresource( iv_resourcearn =
 iv_resource_arn ).
        DATA(lt_tags) = oo_result->get_tags( ).
        DATA(lv_count) = lines( lt_tags ).
        MESSAGE |Found { lv_count } tags for resource.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
```

```
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTagsForResourcereferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`SearchImageSets`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SearchImageSets`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Começar a usar conjuntos de imagens e quadros de imagem

C++

SDK para C++

A função de utilitário para pesquisar conjuntos de imagens.

```
//! Routine which searches for image sets based on defined input attributes.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param searchCriteria: A search criteria instance.
  \param imageSetResults: Vector to receive the image set IDs.
  \param clientConfig: Aws client configuration.
```

```
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::searchImageSets(const Aws::String &dataStoreID,
                                              const
 Aws::MedicalImaging::Model::SearchCriteria &searchCriteria,
                                              Aws::Vector<Aws::String>
 &imageSetResults,
                                              const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::SearchImageSetsRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetSearchCriteria(searchCriteria);

    Aws::String nextToken; // Used for paginated results.
    bool result = true;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::MedicalImaging::Model::SearchImageSetsOutcome outcome =
 client.SearchImageSets(
                request);
        if (outcome.IsSuccess()) {
            for (auto &imageSetMetadataSummary:
 outcome.GetResult().GetImageSetsMetadataSummaries()) {

 imageSetResults.push_back(imageSetMetadataSummary.GetImageSetId());
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            result = false;
        }
    } while (!nextToken.empty());

    return result;
}
```

Caso de uso nº 1: operador EQUAL.

```
        Aws::Vector<Aws::String> imageIDsForPatientID;
        Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
        Aws::Vector<Aws::MedicalImaging::Model::SearchFilter>
patientIDSearchFilters = {

Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Oper

.WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(pat
        };

        searchCriteriaEqualsPatientID.SetFilters(patientIDSearchFilters);
        bool result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

searchCriteriaEqualsPatientID,

imageIDsForPatientID,
                                                                clientConfig);
        if (result) {
            std::cout << imageIDsForPatientID.size() << " image sets found for
the patient with ID '"
            <<  patientID << "'." << std::endl;
            for (auto &imageSetResult : imageIDsForPatientID) {
                std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
            }
        }
```

Caso de uso #2: BETWEEN operador usando DICOMStudyDate DICOMStudyTime e.

```
         Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAnd
        .WithDICOMStudyDate("19990101")
        .WithDICOMStudyTime("000000.000"));

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTi
```

```
 .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeSt
%m%d"))
        .WithDICOMStudyTime("000000.000"));

       Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
       useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

 useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

       Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;
       useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

       Aws::Vector<Aws::String> usesCase2Results;
       result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                         useCase2SearchCriteria,
                                                         usesCase2Results,
                                                         clientConfig);
       if (result) {
           std::cout << usesCase2Results.size() << " image sets found for
 between 1999/01/01 and present."
                     <<  std::endl;
           for (auto &imageSetResult : usesCase2Results) {
               std::cout << "  Image set with ID '" << imageSetResult <<
 std::endl;
           }
       }
```

Caso de uso nº 3: operador BETWEEN usando o createdAt. Os estudos de tempo foram previamente persistidos.

```
       Aws::MedicalImaging::Model::SearchByAttributeValue useCase3StartDate;

 useCase3StartDate.SetCreatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::Da

       Aws::MedicalImaging::Model::SearchByAttributeValue useCase3EndDate;

 useCase3EndDate.SetCreatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

       Aws::MedicalImaging::Model::SearchFilter useCase3SearchFilter;
       useCase3SearchFilter.SetValues({useCase3StartDate, useCase3EndDate});
```

```
useCase3SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

       Aws::MedicalImaging::Model::SearchCriteria useCase3SearchCriteria;
       useCase3SearchCriteria.SetFilters({useCase3SearchFilter});

       Aws::Vector<Aws::String> usesCase3Results;
       result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                         useCase3SearchCriteria,
                                                         usesCase3Results,
                                                         clientConfig);
       if (result) {
           std::cout << usesCase3Results.size() << " image sets found for
created between 2023/11/30 and present."
                     <<  std::endl;
           for (auto &imageSetResult : usesCase3Results) {
               std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
           }
       }
```

Caso de uso #4: operador EQUAL em DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifique a resposta em ordem ASC no campo updatedAt.

```
       Aws::MedicalImaging::Model::SearchByAttributeValue useCase4StartDate;

useCase4StartDate.SetUpdatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::Da

       Aws::MedicalImaging::Model::SearchByAttributeValue useCase4EndDate;

useCase4EndDate.SetUpdatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

       Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterBetween;
       useCase4SearchFilterBetween.SetValues({useCase4StartDate,
useCase4EndDate});

useCase4SearchFilterBetween.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

       Aws::MedicalImaging::Model::SearchByAttributeValue seriesInstanceUID;
       seriesInstanceUID.SetDICOMSeriesInstanceUID(dicomSeriesInstanceUID);

       Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterEqual;
```

```
        useCase4SearchFilterEqual.SetValues({seriesInstanceUID});

useCase4SearchFilterEqual.SetOperator(Aws::MedicalImaging::Model::Operator::EQUAL);

        Aws::MedicalImaging::Model::SearchCriteria useCase4SearchCriteria;
        useCase4SearchCriteria.SetFilters({useCase4SearchFilterBetween,
useCase4SearchFilterEqual});

        Aws::MedicalImaging::Model::Sort useCase4Sort;

useCase4Sort.SetSortField(Aws::MedicalImaging::Model::SortField::updatedAt);
        useCase4Sort.SetSortOrder(Aws::MedicalImaging::Model::SortOrder::ASC);

        useCase4SearchCriteria.SetSort(useCase4Sort);

        Aws::Vector<Aws::String> usesCase4Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase4SearchCriteria,
                                                          usesCase4Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase4Results.size() << " image sets found for EQUAL
operator "
            << "on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort
response\n"
            <<  "in ASC order on updatedAt field." <<  std::endl;
            for (auto &imageSetResult : usesCase4Results) {
                std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
            }
        }
```

- Para obter detalhes da API, consulte SearchImageSetsa Referência AWS SDK para C++ da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Exemplo 1: para pesquisar conjuntos de imagens com um operador EQUAL

O exemplo de código `search-image-sets` a seguir usa o operador EQUAL para pesquisar conjuntos de imagens com base em um valor específico.

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria file://search-criteria.json
```

Conteúdo de `search-criteria.json`

```
{
    "filters": [{
        "values": [{"DICOMPatientId" : "SUBJECT08701"}],
        "operator": "EQUAL"
    }]
}
```

Saída:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
             "DICOMPatientSex": "F",
             "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
             "DICOMPatientBirthDate": "19201120",
             "DICOMStudyDescription": "UNKNOWN",
             "DICOMPatientId": "SUBJECT08701",
             "DICOMPatientName": "Melissa844 Huel628",
             "DICOMNumberOfStudyRelatedInstances": 1,
             "DICOMStudyTime": "140728",
             "DICOMNumberOfStudyRelatedSeries": 1
            },
```

```
        "updatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Exemplo 2: Para pesquisar conjuntos de imagens com um operador BETWEEN usando DICOMStudyDate e DICOMStudyTime

O exemplo de código `search-image-sets` a seguir pesquisa conjuntos de imagens com estudos DICOM gerados entre 1º de janeiro de 1990 (00h) e 1º de janeiro de 2023 (00h).

Nota: DICOMStudyTime é opcional. Se não estiver presente, 00h (início do dia) é o valor da hora para as datas fornecidas para filtragem.

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria file://search-criteria.json
```

Conteúdo de `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "DICOMStudyDateAndTime": {
                "DICOMStudyDate": "19900101",
                "DICOMStudyTime": "000000"
            }
        },
        {
            "DICOMStudyDateAndTime": {
                "DICOMStudyDate": "20230101",
                "DICOMStudyTime": "000000"
            }
        }],
        "operator": "BETWEEN"
    }]
}
```

Saída:

```
{
    "imageSetsMetadataSummaries": [{
```

```
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "updatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Exemplo 3: para pesquisar conjuntos de imagens com um operador BETWEEN usando createdAt (os estudos de tempo persistiam anteriormente)

O exemplo de `search-image-sets` código a seguir pesquisa conjuntos de imagens com estudos DICOM persistentes HealthImaging entre os intervalos de tempo no fuso horário UTC.

Nota: Forneça createdAt no formato de exemplo (“1985-04-12 “T23:20:50.52Z).

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria  file://search-criteria.json
```

Conteúdo de `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "createdAt": "1985-04-12T23:20:50.52Z"
        },
        {
            "createdAt": "2022-04-12T23:20:50.52Z"
        }],
```

```
        "operator": "BETWEEN"
    }]
}
```

Saída:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Exemplo 4: Para pesquisar conjuntos de imagens com um operador EQUAL em e BETWEEN em updatedAt DICOMSeriesInstanceUID e classificar a resposta em ordem ASC no campo updatedAt

O exemplo de `search-image-sets` código a seguir pesquisa conjuntos de imagens com um operador EQUAL em DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifica a resposta em ordem ASC no campo updatedAt.

Nota: Forneça updatedAt no formato de exemplo (“1985-04-12 “T23:20:50.52Z).

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria  file://search-criteria.json
```

Conteúdo de `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "updatedAt": "2024-03-11T15:00:05.074000-07:00"
        }, {
            "updatedAt": "2024-03-11T16:00:05.074000-07:00"
        }],
        "operator": "BETWEEN"
    }, {
        "values": [{
            "DICOMSeriesInstanceUID": "1.2.840.99999999.84710745.943275268089"
        }],
        "operator": "EQUAL"
    }],
    "sort": {
        "sortField": "updatedAt",
        "sortOrder": "ASC"
    }
}
```

Saída:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
```

```
    }]
}
```

Para obter mais informações, consulte Pesquisar conjuntos de imagens no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte SearchImageSetsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

A função de utilitário para pesquisar conjuntos de imagens.

```
    public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
            MedicalImagingClient medicalImagingClient,
            String datastoreId, SearchCriteria searchCriteria) {
        try {
            SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
                    .datastoreId(datastoreId)
                    .searchCriteria(searchCriteria)
                    .build();
            SearchImageSetsIterable responses = medicalImagingClient
                    .searchImageSetsPaginator(datastoreRequest);
            List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();

            responses.stream().forEach(response -> imageSetsMetadataSummaries
                    .addAll(response.imageSetsMetadataSummaries()));

            return imageSetsMetadataSummaries;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

Caso de uso nº 1: operador EQUAL.

```
        List<SearchFilter> searchFilters =
 Collections.singletonList(SearchFilter.builder()
                .operator(Operator.EQUAL)
                .values(SearchByAttributeValue.builder()
                        .dicomPatientId(patientId)
                        .build())
                .build());

        SearchCriteria searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
 searchMedicalImagingImageSets(
                medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets for patient " + patientId + " are:
\n"
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Caso de uso #2: BETWEEN operador usando DICOMStudyDate DICOMStudyTime e.

```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

 .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate("19990101")
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build(),
                        SearchByAttributeValue.builder()

 .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate((LocalDate.now()
                                                .format(formatter)))
                                        .dicomStudyTime("000000.000")
                                        .build())
```

```
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println(
                    "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
                            +
                            imageSetsMetadataSummaries);
            System.out.println();
        }
```

Caso de uso nº 3: operador BETWEEN usando o createdAt. Os estudos de tempo foram previamente persistidos.

```
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

.createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                                .build(),
                        SearchByAttributeValue.builder()
                                .createdAt(Instant.now())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();
        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with BETWEEN operator
using createdAt are:\n "
```

```
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Caso de uso #4: operador EQUAL em DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifique a resposta em ordem ASC no campo updatedAt.

```
        Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
        Instant endDate = Instant.now();

        searchFilters = Arrays.asList(
                SearchFilter.builder()
                        .operator(Operator.EQUAL)
                        .values(SearchByAttributeValue.builder()
                                .dicomSeriesInstanceUID(seriesInstanceUID)
                                .build())
                        .build(),
                SearchFilter.builder()
                        .operator(Operator.BETWEEN)
                        .values(

SearchByAttributeValue.builder().updatedAt(startDate).build(),

SearchByAttributeValue.builder().updatedAt(endDate).build()
                        ).build());

        Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .sort(sort)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
                    "in ASC order on updatedAt field are:\n "
                    + imageSetsMetadataSummaries);
```

```
            System.out.println();
        }
```

- Para obter detalhes da API, consulte SearchImageSetsa Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

A função de utilitário para pesquisar conjuntos de imagens.

```
import { paginateSearchImageSets } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store's ID.
 * @param { import('@aws-sdk/client-medical-imaging').SearchFilter[] } filters -
 The search criteria filters.
 * @param { import('@aws-sdk/client-medical-imaging').Sort } sort - The search
 criteria sort.
 */
export const searchImageSets = async (
  datastoreId = "xxxxxxxx",
  searchCriteria = {},
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {
    datastoreId: datastoreId,
    searchCriteria: searchCriteria,
  };
```

```
  const paginator = paginateSearchImageSets(paginatorConfig, commandParams);

  const imageSetsMetadataSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetsMetadataSummaries.push(...page.imageSetsMetadataSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'f009ea9c-84ca-4749-b5b6-7164f00a5ada',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetsMetadataSummaries: [
  //         {
  //             DICOMTags: [Object],
  //             createdAt: "2023-09-19T16:59:40.551Z",
  //             imageSetId: '7f75e1b5c0f40eac2b24cf712f485f50',
  //             updatedAt: "2023-09-19T16:59:40.551Z",
  //             version: 1
  //         }]
  // }

  return imageSetsMetadataSummaries;
};
```

Caso de uso nº 1: operador EQUAL.

```
  const datastoreId = "12345678901234567890123456789012";

  try {
    const searchCriteria = {
      filters: [
        {
          values: [{ DICOMPatientId: "1234567" }],
          operator: "EQUAL",
```

```
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Caso de uso #2: BETWEEN operador usando DICOMStudyDate DICOMStudyTime e.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "19900101",
              DICOMStudyTime: "000000",
            },
          },
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "20230901",
              DICOMStudyTime: "000000",
            },
          },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Caso de uso nº 3: operador BETWEEN usando o createdAt. Os estudos de tempo foram previamente persistidos.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { createdAt: new Date("1985-04-12T23:20:50.52Z") },
          { createdAt: new Date() },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Caso de uso #4: operador EQUAL em DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifique a resposta em ordem ASC no campo updatedAt.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { updatedAt: new Date("1985-04-12T23:20:50.52Z") },
          { updatedAt: new Date() },
        ],
        operator: "BETWEEN",
      },
      {
        values: [
          {
            DICOMSeriesInstanceUID:
```

```
                "1.1.123.123456.1.12.1.1234567890.1234.12345678.123",
            },
          ],
          operator: "EQUAL",
        },
      ],
      sort: {
        sortOrder: "ASC",
        sortField: "updatedAt",
      },
    };

    await searchImageSets(datastoreId, searchCriteria);
  } catch (err) {
    console.error(err);
  }
```

- Para obter detalhes da API, consulte SearchImageSetsa Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

A função de utilitário para pesquisar conjuntos de imagens.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.
```

```
        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
[{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
self.health_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries
```

Caso de uso nº 1: operador EQUAL.

```
        search_filter = {
            "filters": [
                {"operator": "EQUAL", "values": [{"DICOMPatientId": patient_id}]}
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(f"Image sets found with EQUAL operator\n{image_sets}")
```

Caso de uso #2: BETWEEN operador usando DICOMStudyDate DICOMStudyTime e.

```
        search_filter = {
            "filters": [
```

```
                {
                    "operator": "BETWEEN",
                    "values": [
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "19900101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "20230101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                    ],
                }
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with BETWEEN operator using DICOMStudyDate and
DICOMStudyTime\n{image_sets}"
        )
```

Caso de uso nº 3: operador BETWEEN usando o createdAt. Os estudos de tempo foram previamente persistidos.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "createdAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "createdAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
```

```
                    ],
                    "operator": "BETWEEN",
                }
            ]
        }

        recent_image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with with BETWEEN operator using createdAt
\n{recent_image_sets}"
        )
```

Caso de uso #4: operador EQUAL em DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifique a resposta em ordem ASC no campo updatedAt.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "updatedAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "updatedAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                },
                {
                    "values": [{"DICOMSeriesInstanceUID": series_instance_uid}],
                    "operator": "EQUAL",
                },
            ],
            "sort": {
                "sortOrder": "ASC",
                "sortField": "updatedAt",
            },
        }
```

```
        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            "Image sets found with EQUAL operator on DICOMSeriesInstanceUID and
BETWEEN on updatedAt and"
        )
        print(f"sort response in ASC order on updatedAt field\n{image_sets}")
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a SearchImageSetsReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '1234567890123456789012345678901234567890'
        oo_result = lo_mig->searchimagesets(
          iv_datastoreid = iv_datastore_id
          io_searchcriteria = io_search_criteria ).
        DATA(lt_imagesets) = oo_result->get_imagesetsmetadatasums( ).
        DATA(lv_count) = lines( lt_imagesets ).
        MESSAGE |Found { lv_count } image sets.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
```

```
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SearchImageSetsreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`StartDICOMImportJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartDICOMImportJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Começar a usar conjuntos de imagens e quadros de imagem

C++

SDK para C++

```
//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM
 files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
```

```
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory +
 "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte StartDICOMImportJoba Referência AWS SDK para C++ da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

CLI

AWS CLI

Para iniciar um trabalho de importação dicom

O exemplo de código `start-dicom-import-job` a seguir inicia um trabalho de importação dicom.

```
aws medical-imaging start-dicom-import-job \
    --job-name "my-job" \
    --datastore-id "12345678901234567890123456789012" \
    --input-s3-uri "s3://medical-imaging-dicom-input/dicom_input/" \
    --output-s3-uri "s3://medical-imaging-output/job_output/" \
    --data-access-role-arn "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole"
```

Saída:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "jobId": "09876543210987654321098765432109",
    "jobStatus": "SUBMITTED",
    "submittedAt": "2022-08-12T11:28:11.152000+00:00"
}
```

Para obter mais informações, consulte Iniciando um trabalho de importação no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte StartDICOMImportJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   public static String startDicomImportJob(MedicalImagingClient
medicalImagingClient,
           String jobName,
           String datastoreId,
           String dataAccessRoleArn,
           String inputS3Uri,
           String outputS3Uri) {

       try {
           StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
                   .jobName(jobName)
                   .datastoreId(datastoreId)
                   .dataAccessRoleArn(dataAccessRoleArn)
                   .inputS3Uri(inputS3Uri)
                   .outputS3Uri(outputS3Uri)
                   .build();
           StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
           return response.jobId();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return "";
   }
```

- Para obter detalhes da API, consulte StartDICOMImportJoba Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { StartDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} jobName - The name of the import job.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} dataAccessRoleArn - The Amazon Resource Name (ARN) of the role
 that grants permission.
 * @param {string} inputS3Uri - The URI of the S3 bucket containing the input
 files.
 * @param {string} outputS3Uri - The URI of the S3 bucket where the output files
 are stored.
 */
export const startDicomImportJob = async (
  jobName = "test-1",
  datastoreId = "12345678901234567890123456789012",
  dataAccessRoleArn = "arn:aws:iam::xxxxxxxxxxxx:role/ImportJobDataAccessRole",
  inputS3Uri = "s3://medical-imaging-dicom-input/dicom_input/",
  outputS3Uri = "s3://medical-imaging-output/job_output/",
) => {
  const response = await medicalImagingClient.send(
    new StartDICOMImportJobCommand({
      jobName: jobName,
      datastoreId: datastoreId,
      dataAccessRoleArn: dataAccessRoleArn,
      inputS3Uri: inputS3Uri,
      outputS3Uri: outputS3Uri,
    }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '6e81d191-d46b-4e48-a08a-cdcc7e11eb79',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
```

```
  //     jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobStatus: 'SUBMITTED',
  //     submittedAt: 2023-09-22T14:48:45.767Z
  // }
  return response;
};
```

- Para obter detalhes da API, consulte StartDICOMImportJoba Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def start_dicom_import_job(
        self, job_name, datastore_id, role_arn, input_s3_uri, output_s3_uri
    ):
        """
        Start a DICOM import job.

        :param job_name: The name of the job.
        :param datastore_id: The ID of the data store.
        :param role_arn: The Amazon Resource Name (ARN) of the role to use for
 the job.
        :param input_s3_uri: The S3 bucket input prefix path containing the DICOM
 files.
        :param output_s3_uri: The S3 bucket output prefix path for the result.
        :return: The job ID.
        """
        try:
```

```
            job = self.health_imaging_client.start_dicom_import_job(
                jobName=job_name,
                datastoreId=datastore_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_s3_uri,
                outputS3Uri=output_s3_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a StartDICOMImportJobReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_job_name = 'import-job-1'
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/ImportJobRole'
```

```
      " iv_input_s3_uri = 's3://my-bucket/input/'
      " iv_output_s3_uri = 's3://my-bucket/output/'
      oo_result = lo_mig->startdicomimportjob(
        iv_jobname = iv_job_name
        iv_datastoreid = iv_datastore_id
        iv_dataaccessrolearn = iv_role_arn
        iv_inputs3uri = iv_input_s3_uri
        iv_outputs3uri = iv_output_s3_uri ).
      DATA(lv_job_id) = oo_result->get_jobid( ).
      MESSAGE |DICOM import job started with ID: { lv_job_id }.| TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Resource not found.' TYPE 'I'.
    CATCH /aws1/cx_migservicequotaexcdex.
      MESSAGE 'Service quota exceeded.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a StartDICOMImportJobreferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`TagResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TagResource`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Marcar um datastore
- Marcar um conjunto de imagens

CLI

AWS CLI

Exemplo 1: para marcar um armazenamento de dados

Os exemplos de código `tag-resource` a seguir marcam um armazenamento de dados.

```
aws medical-imaging tag-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
  --tags '{"Deployment":"Development"}'
```

Este comando não produz saída.

Exemplo 2: para marcar um conjunto de imagens

Os exemplos de código `tag-resource` a seguir marcam um conjunto de imagens.

```
aws medical-imaging tag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
    --tags '{"Deployment":"Development"}'
```

Este comando não produz saída.

Para obter mais informações, consulte Como marcar recursos AWS HealthImaging no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte TagResourceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
```

```
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
```

```
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.
```

```
        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a TagResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->tagresource(
          iv_resourcearn = iv_resource_arn
```

```
          it_tags = it_tags ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a TagResourcereferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`UntagResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UntagResource`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Marcar um datastore
- Marcar um conjunto de imagens

CLI

AWS CLI

Exemplo 1: para desmarcar um armazenamento de dados

O exemplo de código `untag-resource` a seguir desmarca um armazenamento de dados.

```
aws medical-imaging untag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
    --tag-keys '["Deployment"]'
```

Este comando não produz saída.

Exemplo 2: para desmarcar um conjunto de imagens

O exemplo de código `untag-resource` a seguir desmarca um conjunto de imagens.

```
aws medical-imaging untag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
    --tag-keys '["Deployment"]'
```

Este comando não produz saída.

Para obter mais informações, consulte Como marcar recursos AWS HealthImaging no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte UntagResourceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);
```

```
            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte UntagResourcea Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
```

```
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte UntagResourcea Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a UntagResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
```

```
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UntagResourcereferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use **`UpdateImageSetMetadata`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateImageSetMetadata`.

CLI

AWS CLI

Exemplo 1: inserir ou atualizar um atributo nos metadados do conjunto de imagens

O exemplo `update-image-set-metadata` a seguir insere ou atualiza um atributo nos metadados do conjunto de imagens.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Conteúdo de `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":
{\"PatientName\":\"MX^MX\"}}}"
```

```
    }
}
```

Saída:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Exemplo 2: remover um atributo dos metadados do conjunto de imagens

O exemplo `update-image-set-metadata` a seguir remove um atributo dos metadados do conjunto de imagens.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Conteúdo de `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "removableAttributes": "{\"SchemaVersion\":1.1,\"Study\":{\"DICOM\":
{\"StudyDescription\":\"CHEST\"}}}"
    }
}
```

Saída:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
```

```
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Exemplo 3: remover uma instância dos metadados de um conjunto de imagens

O exemplo `update-image-set-metadata` a seguir remove uma instância dos metadados de um conjunto de imagens.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json \
    --force
```

Conteúdo de `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "removableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances\":
 {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {}}}}}}"
    }
}
```

Saída:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Exemplo 4: reverter um conjunto de imagens para uma versão anterior

O `update-image-set-metadata` exemplo a seguir mostra como reverter um conjunto de imagens para uma versão anterior. CopyImageSet e UpdateImageSetMetadata as ações criam novas versões dos conjuntos de imagens.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 3 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates '{"revertToVersionId": "1"}'
```

Saída:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "latestVersionId": "4",
    "imageSetState": "LOCKED",
    "imageSetWorkflowStatus": "UPDATING",
    "createdAt": 1680027126.436,
    "updatedAt": 1680042257.908
}
```

Exemplo 5: adicionar um elemento de dados DICOM privado a uma instância

O exemplo `update-image-set-metadata` a seguir mostra como adicionar um elemento privado a uma instância especificada em um conjunto de imagens. O padrão DICOM permite elementos de dados privados para comunicação de informações que não podem estar contidas em elementos de dados padrão. Você pode criar, atualizar e excluir elementos de dados privados com a UpdateImageSetMetadata ação.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --force \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Conteúdo de `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"DICOM\":
 {\"001910F9\": \"97\"},\"DICOMVRs\": {\"001910F9\": \"DS\"}}}}}}"
    }
}
```

Saída:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Exemplo 6: atualizar um elemento de dados DICOM privado a uma instância

O exemplo `update-image-set-metadata` a seguir mostra como atualizar o valor de um elemento privado que pertence a uma instância em um conjunto de imagens.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --force \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Conteúdo de `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances
```

```
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"DICOM\":
 {\"00091001\": \"GE_GENESIS_DD\"}}}}}}}"
    }
}
```

Saída:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Exemplo 7: atualizar um SOPInstanceUID com o parâmetro force

O exemplo `update-image-set-metadata` a seguir mostra como atualizar um SOPInstanceUID usando o parâmetro force para substituir as restrições de metadados DICOM.

```
aws medical-imaging update-image-set-metadata \
        --datastore-id 12345678901234567890123456789012 \
        --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
        --latest-version-id 1 \
        --cli-binary-format raw-in-base64-out \
        --force \
        --update-image-set-metadata-updates file://metadata-updates.json
```

Conteúdo de `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\":1.1,\"Study\":{\"Series
\":{\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3656.0\":
{\"Instances\":
{\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3659.0\":{\"DICOM\":
{\"SOPInstanceUID\":
\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3659.9\"}}}}}}}"
    }
```

```
}
```

Saída:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Para obter mais informações, consulte Atualização dos metadados do conjunto de imagens no Guia do AWS HealthImaging desenvolvedor.

- Para obter detalhes da API, consulte UpdateImageSetMetadataem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

```
   /**
    * Update the metadata of an AWS HealthImaging image set.
    *
    * @param medicalImagingClient - The AWS HealthImaging client object.
    * @param datastoreId          - The datastore ID.
    * @param imageSetId           - The image set ID.
    * @param versionId            - The version ID.
    * @param metadataUpdates      - A MetadataUpdates object containing the
updates.
    * @param force                - The force flag.
    * @throws MedicalImagingException - Base exception for all service
exceptions thrown by AWS HealthImaging.
    */
   public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imageSetId,
```

```
                                                        String versionId,
                                                        MetadataUpdates
metadataUpdates,
                                                        boolean force) {
        try {
            UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                    .builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imageSetId)
                    .latestVersionId(versionId)
                    .updateImageSetMetadataUpdates(metadataUpdates)
                    .force(force)
                    .build();

            UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

            System.out.println("The image set metadata was updated" + response);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Caso de uso 1: insira ou atualize um atributo.

```
                final String insertAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .updatableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(insertAttributes
```

```
.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataInsertUpdates, force);
```

Caso de uso 2: remova um atributo.

```
                final String removeAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeAttributes

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Caso de uso 3: remova uma instância.

```
                final String removeInstance = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
```

```
"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                              }
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeInstance

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Caso de uso 4: reverta para uma versão anterior.

```
                // In this case, revert to previous version.
                String revertVersionId =
Integer.toString(Integer.parseInt(versionid) - 1);
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .revertToVersionId(revertVersionId)
                        .build();
                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

- Para obter detalhes da API, consulte UpdateImageSetMetadataa Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

```
import { UpdateImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the HealthImaging data store.
 * @param {string} imageSetId - The ID of the HealthImaging image set.
 * @param {string} latestVersionId - The ID of the HealthImaging image set
 version.
 * @param {{}} updateMetadata - The metadata to update.
 * @param {boolean} force - Force the update.
 */
export const updateImageSetMetadata = async (
  datastoreId = "xxxxxxxxxx",
  imageSetId = "xxxxxxxxxx",
  latestVersionId = "1",
  updateMetadata = "{}",
  force = false,
) => {
  try {
    const response = await medicalImagingClient.send(
      new UpdateImageSetMetadataCommand({
        datastoreId: datastoreId,
        imageSetId: imageSetId,
        latestVersionId: latestVersionId,
        updateImageSetMetadataUpdates: updateMetadata,
        force: force,
      }),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //     httpStatusCode: 200,
```

```
    //         requestId: '7966e869-e311-4bff-92ec-56a61d3003ea',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    // },
    //     createdAt: 2023-09-22T14:49:26.427Z,
    //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetState: 'LOCKED',
    //     imageSetWorkflowStatus: 'UPDATING',
    //     latestVersionId: '4',
    //     updatedAt: 2023-09-27T19:41:43.494Z
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Caso de uso 1: insira ou atualize um atributo e force a atualização.

```
    const insertAttributes = JSON.stringify({
      SchemaVersion: 1.1,
      Study: {
        DICOM: {
          StudyDescription: "CT CHEST",
        },
      },
    });

    const updateMetadata = {
      DICOMUpdates: {
        updatableAttributes: new TextEncoder().encode(insertAttributes),
      },
    };

    await updateImageSetMetadata(
      datastoreID,
      imageSetID,
      versionID,
      updateMetadata,
```

```
    true,
  );
```

Caso de uso 2: remova um atributo.

```
  // Attribute key and value must match the existing attribute.
  const remove_attribute = JSON.stringify({
    SchemaVersion: 1.1,
    Study: {
      DICOM: {
        StudyDescription: "CT CHEST",
      },
    },
  });

  const updateMetadata = {
    DICOMUpdates: {
      removableAttributes: new TextEncoder().encode(remove_attribute),
    },
  };

  await updateImageSetMetadata(
    datastoreID,
    imageSetID,
    versionID,
    updateMetadata,
  );
```

Caso de uso 3: remova uma instância.

```
  const remove_instance = JSON.stringify({
    SchemaVersion: 1.1,
    Study: {
      Series: {
        "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
          Instances: {
            "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {},
          },
        },
      },
    },
```

```
});

const updateMetadata = {
  DICOMUpdates: {
    removableAttributes: new TextEncoder().encode(remove_instance),
  },
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

Caso de uso 4: reverta para uma versão anterior.

```
const updateMetadata = {
  revertToVersionId: "1",
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

- Para obter detalhes da API, consulte UpdateImageSetMetadataa Referência AWS SDK para JavaScript da API.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def update_image_set_metadata(
        self, datastore_id, image_set_id, version_id, metadata, force=False
    ):
        """
        Update the metadata of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param metadata: The image set metadata as a dictionary.
            For example {"DICOMUpdates": {"updatableAttributes":
            "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientName\":
\"Garcia^Gloria\"}}}"}}
        :param: force: Force the update.
        :return: The updated image set metadata.
        """
        try:
            updated_metadata =
 self.health_imaging_client.update_image_set_metadata(
                imageSetId=image_set_id,
                datastoreId=datastore_id,
                latestVersionId=version_id,
                updateImageSetMetadataUpdates=metadata,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update image set metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return updated_metadata
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

Caso de uso 1: insira ou atualize um atributo.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"updatableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Caso de uso 2: remova um atributo.

```
            # Attribute key and value must match the existing attribute.
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Caso de uso 3: remova uma instância.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "Series": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                            }
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Caso de uso 4: reverta para uma versão anterior.

```
            metadata = {"revertToVersionId": "1"}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

- Para obter detalhes da API, consulte a UpdateImageSetMetadataReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

```
   TRY.
       " iv_datastore_id = '1234567890123456789012345678901234567890'
       " iv_image_set_id = '1234567890123456789012345678901234567890'
       " iv_latest_version_id = '1'
       " iv_force = abap_false
       oo_result = lo_mig->updateimagesetmetadata(
         iv_datastoreid = iv_datastore_id
         iv_imagesetid = iv_image_set_id
         iv_latestversionid = iv_latest_version_id
         io_updateimagesetmetupdates = io_metadata_updates
         iv_force = iv_force ).
       DATA(lv_new_version) = oo_result->get_latestversionid( ).
       MESSAGE |Image set metadata updated to version: { lv_new_version }.| TYPE
'I'.
     CATCH /aws1/cx_migaccessdeniedex.
       MESSAGE 'Access denied.' TYPE 'I'.
     CATCH /aws1/cx_migconflictexception.
       MESSAGE 'Conflict error.' TYPE 'I'.
     CATCH /aws1/cx_miginternalserverex.
       MESSAGE 'Internal server error.' TYPE 'I'.
     CATCH /aws1/cx_migresourcenotfoundex.
       MESSAGE 'Image set not found.' TYPE 'I'.
     CATCH /aws1/cx_migservicequotaexcdex.
       MESSAGE 'Service quota exceeded.' TYPE 'I'.
     CATCH /aws1/cx_migthrottlingex.
       MESSAGE 'Request throttled.' TYPE 'I'.
     CATCH /aws1/cx_migvalidationex.
       MESSAGE 'Validation error.' TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateImageSetMetadatareferência da API AWS SDK for SAP ABAP.

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cenários para HealthImaging usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns HealthImaging com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas HealthImaging ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Comece com conjuntos de HealthImaging imagens e molduras de imagem usando um AWS SDK
- Marcando um armazenamento HealthImaging de dados usando um AWS SDK
- Marcar um conjunto HealthImaging de imagens usando um AWS SDK

Comece com conjuntos de HealthImaging imagens e molduras de imagem usando um AWS SDK

Os exemplos de código a seguir mostram como importar arquivos DICOM e baixar molduras de imagem em HealthImaging.

A implementação é estruturada como uma aplicação da linha de comando.

- Configure recursos para uma importação DICOM.
- Importe arquivos DICOM para um armazenamento de dados.
- Recupere os IDs do conjunto de imagens para o trabalho de importação.
- Recupere os IDs do quadro de imagem para o conjunto de imagens.
- Baixe, decodifique e verifique os quadros de imagem.
- Limpe recursos.

C++

SDK para C++

Crie uma CloudFormation pilha com os recursos necessários.

```
   Aws::String inputBucketName;
   Aws::String outputBucketName;
   Aws::String dataStoreId;
   Aws::String roleArn;
   Aws::String stackName;

   if (askYesNoQuestion(
           "Would you like to let this workflow create the resources for you?
(y/n) ")) {
       stackName = askQuestion(
               "Enter a name for the AWS CloudFormation stack to create. ");
       Aws::String dataStoreName = askQuestion(
               "Enter a name for the HealthImaging datastore to create. ");

       Aws::Map<Aws::String, Aws::String> outputs = createCloudFormationStack(
               stackName,
               dataStoreName,
               clientConfiguration);

       if (!retrieveOutputs(outputs, dataStoreId, inputBucketName,
outputBucketName,
                            roleArn)) {
           return false;
       }

       std::cout << "The following resources have been created." << std::endl;
       std::cout << "A HealthImaging datastore with ID: " << dataStoreId << "."
                 << std::endl;
       std::cout << "An Amazon S3 input bucket named: " << inputBucketName <<
"."
                 << std::endl;
       std::cout << "An Amazon S3 output bucket named: " << outputBucketName <<
"."
                 << std::endl;
       std::cout << "An IAM role with the ARN: " << roleArn << "." << std::endl;
       askQuestion("Enter return to continue.", alwaysTrueTest);
   }
   else {
       std::cout << "You have chosen to use preexisting resources:" <<
std::endl;
       dataStoreId = askQuestion(
               "Enter the data store ID of the HealthImaging datastore you wish
to use: ");
```

```
        inputBucketName = askQuestion(
                "Enter the name of the S3 input bucket you wish to use: ");
        outputBucketName = askQuestion(
                "Enter the name of the S3 output bucket you wish to use: ");
        roleArn = askQuestion(
                "Enter the ARN for the IAM role with the proper permissions to
import a DICOM series: ");
    }
```

Copie arquivos DICOM para o bucket de importação do Amazon S3.

```
    std::cout
            << "This workflow uses DICOM files from the National Cancer Institute
Imaging Data\n"
            << "Commons (IDC) Collections." << std::endl;
    std::cout << "Here is the link to their website." << std::endl;
    std::cout << "https://registry.opendata.aws/nci-imaging-data-commons/" <<
std::endl;
    std::cout << "We will use DICOM files stored in an S3 bucket managed by the
IDC."
              << std::endl;
    std::cout
            << "First one of the DICOM folders in the IDC collection must be
copied to your\n"
               "input S3 bucket."
            << std::endl;
    std::cout << "You have the choice of one of the following "
              << IDC_ImageChoices.size() << " folders to copy." << std::endl;

    int index = 1;
    for (auto &idcChoice: IDC_ImageChoices) {
        std::cout << index << " - " << idcChoice.mDescription << std::endl;
        index++;
    }
    int choice = askQuestionForIntRange("Choose DICOM files to import: ", 1, 4);

    Aws::String fromDirectory = IDC_ImageChoices[choice - 1].mDirectory;
    Aws::String inputDirectory = "input";

    std::cout << "The files in the directory '" << fromDirectory << "' in the
bucket '"
              << IDC_S3_BucketName << "' will be copied " << std::endl;
```

```
    std::cout << "to the folder '" << inputDirectory << "/" << fromDirectory
              << "' in the bucket '" << inputBucketName << "'." << std::endl;
    askQuestion("Enter return to start the copy.", alwaysTrueTest);

    if (!AwsDoc::Medical_Imaging::copySeriesBetweenBuckets(
            IDC_S3_BucketName,
            fromDirectory,
            inputBucketName,
            inputDirectory, clientConfiguration)) {
        std::cerr << "This workflow will exit because of an error." << std::endl;
        cleanup(stackName, dataStoreId, clientConfiguration);
        return false;
    }
```

Importe os arquivos DICOM para o armazenamento de dados do Amazon S3.

```
bool AwsDoc::Medical_Imaging::startDicomImport(const Aws::String &dataStoreID,
                                               const Aws::String
 &inputBucketName,
                                               const Aws::String &inputDirectory,
                                               const Aws::String
 &outputBucketName,
                                               const Aws::String
 &outputDirectory,
                                               const Aws::String &roleArn,
                                               Aws::String &importJobId,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = false;
    if (startDICOMImportJob(dataStoreID, inputBucketName, inputDirectory,
                            outputBucketName, outputDirectory, roleArn,
 importJobId,
                            clientConfiguration)) {
        std::cout << "DICOM import job started with job ID " << importJobId <<
 "."
                  << std::endl;
        result = waitImportJobCompleted(dataStoreID, importJobId,
 clientConfiguration);
        if (result) {
            std::cout << "DICOM import job completed." << std::endl;

        }
```

```
    }

    return result;
}

//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM
 files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory +
 "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
```

```
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}


//! Routine which waits for a DICOM import job to complete.
/*!
 * @param dataStoreID: The HealthImaging data store ID.
 * @param importJobId: The import job ID.
 * @param clientConfiguration : Aws client configuration.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::waitImportJobCompleted(const Aws::String
 &datastoreID,
                                                      const Aws::String
 &importJobId,
                                                      const
 Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::JobStatus jobStatus =
 Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS;
    while (jobStatus == Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS) {
        std::this_thread::sleep_for(std::chrono::seconds(1));

        Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
 getDicomImportJobOutcome = getDICOMImportJob(
                datastoreID, importJobId,
                clientConfiguration);

        if (getDicomImportJobOutcome.IsSuccess()) {
            jobStatus =
 getDicomImportJobOutcome.GetResult().GetJobProperties().GetJobStatus();

            std::cout << "DICOM import job status: " <<

 Aws::MedicalImaging::Model::JobStatusMapper::GetNameForJobStatus(
                              jobStatus) << std::endl;
        }
        else {
```

```
            std::cerr << "Failed to get import job status because "
                      << getDicomImportJobOutcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    }

    return jobStatus == Aws::MedicalImaging::Model::JobStatus::COMPLETED;
}

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

Obtenha conjuntos de imagens criados pelo trabalho de importação DICOM.

```
bool
AwsDoc::Medical_Imaging::getImageSetsForDicomImportJob(const Aws::String
 &datastoreID,
```

```
                                                            const Aws::String
&importJobId,
                                                            Aws::Vector<Aws::String>
&imageSets,
                                                            const
Aws::Client::ClientConfiguration &clientConfiguration) {
   Aws::MedicalImaging::Model::GetDICOMImportJobOutcome getDicomImportJobOutcome
= getDICOMImportJob(
           datastoreID, importJobId, clientConfiguration);
   bool result = false;
   if (getDicomImportJobOutcome.IsSuccess()) {
       auto outputURI =
getDicomImportJobOutcome.GetResult().GetJobProperties().GetOutputS3Uri();
       Aws::Http::URI uri(outputURI);
       const Aws::String &bucket = uri.GetAuthority();
       Aws::String key = uri.GetPath();

       Aws::S3::S3Client s3Client(clientConfiguration);
       Aws::S3::Model::GetObjectRequest objectRequest;
       objectRequest.SetBucket(bucket);
       objectRequest.SetKey(key + "/" + IMPORT_JOB_MANIFEST_FILE_NAME);

       auto getObjectOutcome = s3Client.GetObject(objectRequest);
       if (getObjectOutcome.IsSuccess()) {
           auto &data = getObjectOutcome.GetResult().GetBody();

           std::stringstream stringStream;
           stringStream << data.rdbuf();

           try {
               // Use JMESPath to extract the image set IDs.
               // https://jmespath.org/specification.html
               std::string jmesPathExpression =
"jobSummary.imageSetsSummary[].imageSetId";
               jsoncons::json doc = jsoncons::json::parse(stringStream.str());

               jsoncons::json imageSetsJson = jsoncons::jmespath::search(doc,

jmesPathExpression);\
               for (auto &imageSet: imageSetsJson.array_range()) {
                   imageSets.push_back(imageSet.as_string());
               }

               result = true;
```

```
            }
            catch (const std::exception &e) {
                std::cerr << e.what() << '\n';
            }

        }
        else {
            std::cerr << "Failed to get object because "
                      << getObjectOutcome.GetError().GetMessage() << std::endl;
        }

    }
    else {
        std::cerr << "Failed to get import job status because "
                  << getDicomImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return result;
}
```

Obtenha informações sobre os quadros de imagem de conjuntos de imagens.

```
bool AwsDoc::Medical_Imaging::getImageFramesForImageSet(const Aws::String
 &dataStoreID,
                                                         const Aws::String
 &imageSetID,
                                                         const Aws::String
 &outDirectory,

 Aws::Vector<ImageFrameInfo> &imageFrames,
                                                         const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::String fileName = outDirectory + "/" + imageSetID +
 "_metadata.json.gzip";
    bool result = false;
    if (getImageSetMetadata(dataStoreID, imageSetID, "", // Empty string for
 version ID.
                            fileName, clientConfiguration)) {
        try {
            std::string metadataGZip;
            {
```

```
                std::ifstream inFileStream(fileName.c_str(), std::ios::binary);
                if (!inFileStream) {
                    throw std::runtime_error("Failed to open file " + fileName);
                }

                std::stringstream stringStream;
                stringStream << inFileStream.rdbuf();
                metadataGZip = stringStream.str();
            }
            std::string metadataJson = gzip::decompress(metadataGZip.data(),
                                                        metadataGZip.size());
            // Use JMESPath to extract the image set IDs.
            // https://jmespath.org/specification.html
            jsoncons::json doc = jsoncons::json::parse(metadataJson);
            std::string jmesPathExpression = "Study.Series.*.Instances[].*[]";
            jsoncons::json instances = jsoncons::jmespath::search(doc,

jmesPathExpression);
            for (auto &instance: instances.array_range()) {
                jmesPathExpression = "DICOM.RescaleSlope";
                std::string rescaleSlope = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();
                jmesPathExpression = "DICOM.RescaleIntercept";
                std::string rescaleIntercept =
jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();

                jmesPathExpression = "ImageFrames[][]";
                jsoncons::json imageFramesJson =
jsoncons::jmespath::search(instance,

jmesPathExpression);

                for (auto &imageFrame: imageFramesJson.array_range()) {
                    ImageFrameInfo imageFrameIDs;
                    imageFrameIDs.mImageSetId = imageSetID;
                    imageFrameIDs.mImageFrameId = imageFrame.find(
                            "ID")->value().as_string();
                    imageFrameIDs.mRescaleIntercept = rescaleIntercept;
                    imageFrameIDs.mRescaleSlope = rescaleSlope;
                    imageFrameIDs.MinPixelValue = imageFrame.find(
                            "MinPixelValue")->value().as_string();
```

```
                    imageFrameIDs.MaxPixelValue = imageFrame.find(
                            "MaxPixelValue")->value().as_string();

                    jmesPathExpression =
 "max_by(PixelDataChecksumFromBaseToFullResolution, &Width).Checksum";
                    jsoncons::json checksumJson =
 jsoncons::jmespath::search(imageFrame,

 jmesPathExpression);
                    imageFrameIDs.mFullResolutionChecksum =
 checksumJson.as_integer<uint32_t>();

                    imageFrames.emplace_back(imageFrameIDs);
                }
            }

            result = true;
        }
        catch (const std::exception &e) {
            std::cerr << "getImageFramesForImageSet failed because " << e.what()
                      << std::endl;
        }
    }

    return result;
}

//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped
 json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String
 &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
```

```
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Baixe, decodifique e verifique os quadros de imagem.

```
bool AwsDoc::Medical_Imaging::downloadDecodeAndCheckImageFrames(
        const Aws::String &dataStoreID,
        const Aws::Vector<ImageFrameInfo> &imageFrames,
        const Aws::String &outDirectory,
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::Client::ClientConfiguration clientConfiguration1(clientConfiguration);
    clientConfiguration1.executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(
            clientConfiguration1);

    Aws::Utils::Threading::Semaphore semaphore(0, 1);
    std::atomic<size_t> count(imageFrames.size());

    bool result = true;
```

```
    for (auto &imageFrame: imageFrames) {
        Aws::MedicalImaging::Model::GetImageFrameRequest getImageFrameRequest;
        getImageFrameRequest.SetDatastoreId(dataStoreID);
        getImageFrameRequest.SetImageSetId(imageFrame.mImageSetId);

        Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
        imageFrameInformation.SetImageFrameId(imageFrame.mImageFrameId);
        getImageFrameRequest.SetImageFrameInformation(imageFrameInformation);

        auto getImageFrameAsyncLambda = [&semaphore, &result, &count, imageFrame,
outDirectory](
                const Aws::MedicalImaging::MedicalImagingClient *client,
                const Aws::MedicalImaging::Model::GetImageFrameRequest &request,
                Aws::MedicalImaging::Model::GetImageFrameOutcome outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext>
&context) {

                if (!handleGetImageFrameResult(outcome, outDirectory,
imageFrame)) {
                    std::cerr << "Failed to download and convert image frame: "
                              << imageFrame.mImageFrameId << " from image set: "
                              << imageFrame.mImageSetId << std::endl;
                    result = false;
                }

                count--;
                if (count <= 0) {

                    semaphore.ReleaseAll();
                }
        }; // End of 'getImageFrameAsyncLambda' lambda.

        medicalImagingClient.GetImageFrameAsync(getImageFrameRequest,
                                                getImageFrameAsyncLambda);
    }

    if (count > 0) {
        semaphore.WaitOne();
    }

    if (result) {
        std::cout << imageFrames.size() << " image files were downloaded."
                  << std::endl;
    }
```

```
    return result;
}

bool AwsDoc::Medical_Imaging::decodeJPHFileAndValidateWithChecksum(
        const Aws::String &jphFile,
        uint32_t crc32Checksum) {
    opj_image_t *outputImage = jphImageToOpjBitmap(jphFile);
    if (!outputImage) {
        return false;
    }

    bool result = true;
    if (!verifyChecksumForImage(outputImage, crc32Checksum)) {
        std::cerr << "The checksum for the image does not match the expected
 value."
                  << std::endl;
        std::cerr << "File :" << jphFile << std::endl;
        result = false;
    }

    opj_image_destroy(outputImage);

    return result;
}

opj_image *
AwsDoc::Medical_Imaging::jphImageToOpjBitmap(const Aws::String &jphFile) {
    opj_stream_t *inFileStream = nullptr;
    opj_codec_t *decompressorCodec = nullptr;
    opj_image_t *outputImage = nullptr;
    try {
        std::shared_ptr<opj_dparameters> decodeParameters =
 std::make_shared<opj_dparameters>();
        memset(decodeParameters.get(), 0, sizeof(opj_dparameters));

        opj_set_default_decoder_parameters(decodeParameters.get());

        decodeParameters->decod_format = 1; // JP2 image format.
        decodeParameters->cod_format = 2; // BMP image format.

        std::strncpy(decodeParameters->infile, jphFile.c_str(),
                     OPJ_PATH_LEN);
```

```
        inFileStream = opj_stream_create_default_file_stream(
                decodeParameters->infile, true);
        if (!inFileStream) {
            throw std::runtime_error(
                    "Unable to create input file stream for file '" + jphFile +
"'.");
        }

        decompressorCodec = opj_create_decompress(OPJ_CODEC_JP2);
        if (!decompressorCodec) {
            throw std::runtime_error("Failed to create decompression codec.");
        }

        int decodeMessageLevel = 1;
        if (!setupCodecLogging(decompressorCodec, &decodeMessageLevel)) {
            std::cerr << "Failed to setup codec logging." << std::endl;
        }

        if (!opj_setup_decoder(decompressorCodec, decodeParameters.get())) {
            throw std::runtime_error("Failed to setup decompression codec.");
        }
        if (!opj_codec_set_threads(decompressorCodec, 4)) {
            throw std::runtime_error("Failed to set decompression codec
threads.");
        }

        if (!opj_read_header(inFileStream, decompressorCodec, &outputImage)) {
            throw std::runtime_error("Failed to read header.");
        }

        if (!opj_decode(decompressorCodec, inFileStream,
                        outputImage)) {
            throw std::runtime_error("Failed to decode.");
        }

        if (DEBUGGING) {
            std::cout << "image width : " << outputImage->x1 - outputImage->x0
                      << std::endl;
            std::cout << "image height : " << outputImage->y1 - outputImage->y0
                      << std::endl;
            std::cout << "number of channels: " << outputImage->numcomps
                      << std::endl;
            std::cout << "colorspace : " << outputImage->color_space <<
std::endl;
```

```
        }

    } catch (const std::exception &e) {
        std::cerr << e.what() << std::endl;
        if (outputImage) {
            opj_image_destroy(outputImage);
            outputImage = nullptr;
        }
    }
    if (inFileStream) {
        opj_stream_destroy(inFileStream);
    }
    if (decompressorCodec) {
        opj_destroy_codec(decompressorCodec);
    }

    return outputImage;
}

//! Template function which converts a planar image bitmap to an interleaved
 image bitmap and
//! then verifies the checksum of the bitmap.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
template<class myType>
bool verifyChecksumForImageForType(opj_image_t *image, uint32_t crc32Checksum) {
    uint32_t width = image->x1 - image->x0;
    uint32_t height = image->y1 - image->y0;
    uint32_t numOfChannels = image->numcomps;

    // Buffer for interleaved bitmap.
    std::vector<myType> buffer(width * height * numOfChannels);

    // Convert planar bitmap to interleaved bitmap.
    for (uint32_t channel = 0; channel < numOfChannels; channel++) {
        for (uint32_t row = 0; row < height; row++) {
            uint32_t fromRowStart = row / image->comps[channel].dy * width /
                                    image->comps[channel].dx;
            uint32_t toIndex = (row * width) * numOfChannels + channel;

            for (uint32_t col = 0; col < width; col++) {
```

```
                uint32_t fromIndex = fromRowStart + col / image-
>comps[channel].dx;

                buffer[toIndex] = static_cast<myType>(image-
>comps[channel].data[fromIndex]);

                toIndex += numOfChannels;
            }
        }
    }

    // Verify checksum.
    boost::crc_32_type crc32;
    crc32.process_bytes(reinterpret_cast<char *>(buffer.data()),
                        buffer.size() * sizeof(myType));

    bool result = crc32.checksum() == crc32Checksum;
    if (!result) {
        std::cerr << "verifyChecksumForImage, checksum mismatch, expected - "
                  << crc32Checksum << ", actual - " << crc32.checksum()
                  << std::endl;
    }

    return result;
}

//! Routine which verifies the checksum of an OpenJPEG image struct.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::verifyChecksumForImage(opj_image_t *image,
                                                     uint32_t crc32Checksum) {
    uint32_t channels = image->numcomps;
    bool result = false;
    if (0 < channels) {
        // Assume the precision is the same for all channels.
        uint32_t precision = image->comps[0].prec;
        bool signedData = image->comps[0].sgnd;
        uint32_t bytes = (precision + 7) / 8;

        if (signedData) {
            switch (bytes) {
```

```
                case 1 :
                    result = verifyChecksumForImageForType<int8_t>(image,

crc32Checksum);
                    break;
                case 2 :
                    result = verifyChecksumForImageForType<int16_t>(image,

crc32Checksum);
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<int32_t>(image,

crc32Checksum);
                    break;
                default:
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
signed bytes - "
                            << bytes << std::endl;
                    break;
            }
        }
        else {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<uint8_t>(image,

crc32Checksum);
                    break;
                case 2 :
                    result = verifyChecksumForImageForType<uint16_t>(image,

crc32Checksum);
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<uint32_t>(image,

crc32Checksum);
                    break;
                default:
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
unsigned bytes - "
```

```
                            << bytes << std::endl;
                    break;
            }
        }

        if (!result) {
            std::cerr << "verifyChecksumForImage, error bytes " << bytes
                      << " signed "
                      << signedData << std::endl;
        }
    }
    else {
        std::cerr << "'verifyChecksumForImage', no channels in the image."
                  << std::endl;
    }
    return result;
}
```

Limpe recursos.

```
bool AwsDoc::Medical_Imaging::cleanup(const Aws::String &stackName,
                                      const Aws::String &dataStoreId,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    bool result = true;

    if (!stackName.empty() && askYesNoQuestion(
            "Would you like to delete the stack " + stackName + "? (y/n)")) {
        std::cout << "Deleting the image sets in the stack." << std::endl;
        result &= emptyDatastore(dataStoreId, clientConfiguration);
        printAsterisksLine();
        std::cout << "Deleting the stack." << std::endl;
        result &= deleteStack(stackName, clientConfiguration);
    }
    return result;
}

bool AwsDoc::Medical_Imaging::emptyDatastore(const Aws::String &datastoreID,
                                             const
 Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::SearchCriteria emptyCriteria;
```

```
    Aws::Vector<Aws::String> imageSetIDs;
    bool result = false;
    if (searchImageSets(datastoreID, emptyCriteria, imageSetIDs,
                        clientConfiguration)) {
        result = true;
        for (auto &imageSetID: imageSetIDs) {
            result &= deleteImageSet(datastoreID, imageSetID,
 clientConfiguration);
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

Orquestre etapas (index.js).

```
import {
  parseScenarioArgs,
  Scenario,
```

```
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  saveState,
  loadState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import {
  createStack,
  deployStack,
  getAccountId,
  getDatastoreName,
  getStackName,
  outputState,
  waitForStackCreation,
} from "./deploy-steps.js";
import {
  doCopy,
  selectDataset,
  copyDataset,
  outputCopiedObjects,
} from "./dataset-steps.js";
import {
  doImport,
  outputImportJobStatus,
  startDICOMImport,
  waitForImportJobCompletion,
} from "./import-steps.js";
import {
  getManifestFile,
  outputImageSetIds,
  parseManifestFile,
} from "./image-set-steps.js";
import {
  getImageSetMetadata,
  outputImageFrameIds,
} from "./image-frame-steps.js";
import { decodeAndVerifyImages, doVerify } from "./verify-steps.js";
import {
  confirmCleanup,
  deleteImageSets,
  deleteStack,
} from "./clean-up-steps.js";

const context = {};
```

```
const scenarios = {
  deploy: new Scenario(
    "Deploy Resources",
    [
      deployStack,
      getStackName,
      getDatastoreName,
      getAccountId,
      createStack,
      waitForStackCreation,
      outputState,
      saveState,
    ],
    context,
  ),
  demo: new Scenario(
    "Run Demo",
    [
      loadState,
      doCopy,
      selectDataset,
      copyDataset,
      outputCopiedObjects,
      doImport,
      startDICOMImport,
      waitForImportJobCompletion,
      outputImportJobStatus,
      getManifestFile,
      parseManifestFile,
      outputImageSetIds,
      getImageSetMetadata,
      outputImageFrameIds,
      doVerify,
      decodeAndVerifyImages,
      saveState,
    ],
    context,
  ),
  destroy: new Scenario(
    "Clean Up Resources",
    [loadState, confirmCleanup, deleteImageSets, deleteStack],
    context,
  ),
```

```
};

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Health Imaging Workflow",
    description:
      "Work with DICOM images using an AWS Health Imaging data store.",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
  });
}
```

Implante recursos (deploy-steps.js).

```
import fs from "node:fs/promises";
import path from "node:path";

import {
  CloudFormationClient,
  CreateStackCommand,
  DescribeStacksCommand,
} from "@aws-sdk/client-cloudformation";
import { STSClient, GetCallerIdentityCommand } from "@aws-sdk/client-sts";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const cfnClient = new CloudFormationClient({});
const stsClient = new STSClient({});

const __dirname = path.dirname(new URL(import.meta.url).pathname);
const cfnTemplatePath = path.join(
  __dirname,
  "../../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml",
```

```
);

export const deployStack = new ScenarioInput(
  "deployStack",
  "Do you want to deploy the CloudFormation stack?",
  { type: "confirm" },
);

export const getStackName = new ScenarioInput(
  "getStackName",
  "Enter a name for the CloudFormation stack:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getDatastoreName = new ScenarioInput(
  "getDatastoreName",
  "Enter a name for the HealthImaging datastore:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getAccountId = new ScenarioAction(
  "getAccountId",
  async (/** @type {{}} */ state) => {
    const command = new GetCallerIdentityCommand({});
    const response = await stsClient.send(command);
    state.accountId = response.Account;
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const createStack = new ScenarioAction(
  "createStack",
  async (/** @type {{}} */ state) => {
    const stackName = state.getStackName;
    const datastoreName = state.getDatastoreName;
    const accountId = state.accountId;

    const command = new CreateStackCommand({
      StackName: stackName,
      TemplateBody: await fs.readFile(cfnTemplatePath, "utf8"),
      Capabilities: ["CAPABILITY_IAM"],
      Parameters: [
```

```
        {
          ParameterKey: "datastoreName",
          ParameterValue: datastoreName,
        },
        {
          ParameterKey: "userAccountID",
          ParameterValue: accountId,
        },
      ],
    });

    const response = await cfnClient.send(command);
    state.stackId = response.StackId;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const waitForStackCreation = new ScenarioAction(
  "waitForStackCreation",
  async (/** @type {{}} */ state) => {
    const command = new DescribeStacksCommand({
      StackName: state.stackId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await cfnClient.send(command);
      const stack = response.Stacks?.find(
        (s) => s.StackName === state.getStackName,
      );
      if (!stack || stack.StackStatus === "CREATE_IN_PROGRESS") {
        throw new Error("Stack creation is still in progress");
      }
      if (stack.StackStatus === "CREATE_COMPLETE") {
        state.stackOutputs = stack.Outputs?.reduce((acc, output) => {
          acc[output.OutputKey] = output.OutputValue;
          return acc;
        }, {});
      } else {
        throw new Error(
          `Stack creation failed with status: ${stack.StackStatus}`,
        );
      }
    });
  },
```

```
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const outputState = new ScenarioOutput(
  "outputState",
  (/** @type {{}} */ state) => {
    /**
     * @type {{ stackOutputs: { DatastoreID: string, BucketName: string, RoleArn:
 string }}}
     */
    const { stackOutputs } = state;
    return `Stack creation completed. Output values:
Datastore ID: ${stackOutputs?.DatastoreID}
Bucket Name: ${stackOutputs?.BucketName}
Role ARN: ${stackOutputs?.RoleArn}
    `;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);
```

Copie arquivos DICOM (dataset-steps.js).

```
import {
  S3Client,
  CopyObjectCommand,
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const s3Client = new S3Client({});

const datasetOptions = [
  {
    name: "CT of chest (2 images)",
    value: "00029d25-fb18-4d42-aaa5-a0897d1ac8f7",
```

```
  },
  {
    name: "CT of pelvis (57 images)",
    value: "00025d30-ef8f-4135-a35a-d83eff264fc1",
  },
  {
    name: "MRI of head (192 images)",
    value: "0002d261-8a5d-4e63-8e2e-0cbfac87b904",
  },
  {
    name: "MRI of breast (92 images)",
    value: "0002dd07-0b7f-4a68-a655-44461ca34096",
  },
];

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   doCopy: boolean
 * }}} State
 */

export const selectDataset = new ScenarioInput(
  "selectDataset",
  (state) => {
    if (!state.doCopy) {
      process.exit(0);
    }
    return "Select a DICOM dataset to import:";
  },
  {
    type: "select",
    choices: datasetOptions,
  },
);

export const doCopy = new ScenarioInput(
  "doCopy",
  "Do you want to copy images from the public dataset into your bucket?",
  {
    type: "confirm",
  },
);
```

```
export const copyDataset = new ScenarioAction(
  "copyDataset",
  async (/** @type { State } */ state) => {
    const inputBucket = state.stackOutputs.BucketName;
    const inputPrefix = "input/";
    const selectedDatasetId = state.selectDataset;

    const sourceBucket = "idc-open-data";
    const sourcePrefix = `${selectedDatasetId}`;

    const listObjectsCommand = new ListObjectsV2Command({
      Bucket: sourceBucket,
      Prefix: sourcePrefix,
    });

    const objects = await s3Client.send(listObjectsCommand);

    const copyPromises = objects.Contents.map((object) => {
      const sourceKey = object.Key;
      const destinationKey = `${inputPrefix}${sourceKey
        .split("/")
        .slice(1)
        .join("/")}`;

      const copyCommand = new CopyObjectCommand({
        Bucket: inputBucket,
        CopySource: `/${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });

      return s3Client.send(copyCommand);
    });

    const results = await Promise.all(copyPromises);
    state.copiedObjects = results.length;
  },
);

export const outputCopiedObjects = new ScenarioOutput(
  "outputCopiedObjects",
  (state) => `${state.copiedObjects} DICOM files were copied.`,
);
```

Inicie a importação para o datastore (import-steps.js).

```
import {
  MedicalImagingClient,
  StartDICOMImportJobCommand,
  GetDICOMImportJobCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioOutput,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }}} State
 */

export const doImport = new ScenarioInput(
  "doImport",
  "Do you want to import DICOM images into your datastore?",
  {
    type: "confirm",
    default: true,
  },
);

export const startDICOMImport = new ScenarioAction(
  "startDICOMImport",
  async (/** @type {State} */ state) => {
    if (!state.doImport) {
      process.exit(0);
    }
    const medicalImagingClient = new MedicalImagingClient({});
    const inputS3Uri = `s3://${state.stackOutputs.BucketName}/input/`;
    const outputS3Uri = `s3://${state.stackOutputs.BucketName}/output/`;
```

```
    const command = new StartDICOMImportJobCommand({
      dataAccessRoleArn: state.stackOutputs.RoleArn,
      datastoreId: state.stackOutputs.DatastoreID,
      inputS3Uri,
      outputS3Uri,
    });

    const response = await medicalImagingClient.send(command);
    state.importJobId = response.jobId;
  },
);

export const waitForImportJobCompletion = new ScenarioAction(
  "waitForImportJobCompletion",
  async (/** @type {State} */ state) => {
    const medicalImagingClient = new MedicalImagingClient({});
    const command = new GetDICOMImportJobCommand({
      datastoreId: state.stackOutputs.DatastoreID,
      jobId: state.importJobId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await medicalImagingClient.send(command);
      const jobStatus = response.jobProperties?.jobStatus;
      if (!jobStatus || jobStatus === "IN_PROGRESS") {
        throw new Error("Import job is still in progress");
      }
      if (jobStatus === "COMPLETED") {
        state.importJobOutputS3Uri = response.jobProperties.outputS3Uri;
      } else {
        throw new Error(`Import job failed with status: ${jobStatus}`);
      }
    });
  },
);

export const outputImportJobStatus = new ScenarioOutput(
  "outputImportJobStatus",
  (state) =>
    `DICOM import job completed. Output location: ${state.importJobOutputS3Uri}`,
);
```

Obtenha IDs do conjunto de imagens (image-set-steps.js -).

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, importJobId: string,
 * importJobOutputS3Uri: string,
 * imageSetIds: string[],
 * manifestContent: { jobSummary: { imageSetsSummary: { imageSetId: string }
[] } }
 * }} State
 */

const s3Client = new S3Client({});

export const getManifestFile = new ScenarioAction(
  "getManifestFile",
  async (/** @type {State} */ state) => {
    const bucket = state.stackOutputs.BucketName;
    const prefix = `output/${state.stackOutputs.DatastoreID}-DicomImport-
${state.importJobId}/`;
    const key = `${prefix}job-output-manifest.json`;

    const command = new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    });

    const response = await s3Client.send(command);
    const manifestContent = await response.Body.transformToString();
    state.manifestContent = JSON.parse(manifestContent);
  },
);

export const parseManifestFile = new ScenarioAction(
```

```
  "parseManifestFile",
  (/** @type {State} */ state) => {
    const imageSetIds =
      state.manifestContent.jobSummary.imageSetsSummary.reduce((ids, next) => {
        return Object.assign({}, ids, {
          [next.imageSetId]: next.imageSetId,
        });
      }, {});
    state.imageSetIds = Object.keys(imageSetIds);
  },
);

export const outputImageSetIds = new ScenarioOutput(
  "outputImageSetIds",
  (/** @type {State} */ state) =>
    `The image sets created by this import job are: \n${state.imageSetIds
      .map((id) => `Image set: ${id}`)
      .join("\n")}`,
);
```

Obtenha IDs do quadro de imagens (image-frame-steps.js).

```
import {
  MedicalImagingClient,
  GetImageSetMetadataCommand,
} from "@aws-sdk/client-medical-imaging";
import { gunzip } from "node:zlib";
import { promisify } from "node:util";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const gunzipAsync = promisify(gunzip);

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */
```

```
/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */
```

```
/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetIds: string[] }} State
 */

const medicalImagingClient = new MedicalImagingClient({});

export const getImageSetMetadata = new ScenarioAction(
  "getImageSetMetadata",
  async (/** @type {State} */ state) => {
    const outputMetadata = [];

    for (const imageSetId of state.imageSetIds) {
      const command = new GetImageSetMetadataCommand({
        datastoreId: state.stackOutputs.DatastoreID,
        imageSetId,
      });

      const response = await medicalImagingClient.send(command);
      const compressedMetadataBlob =
        await response.imageSetMetadataBlob.transformToByteArray();
      const decompressedMetadata = await gunzipAsync(compressedMetadataBlob);
      const imageSetMetadata = JSON.parse(decompressedMetadata.toString());

      outputMetadata.push(imageSetMetadata);
    }

    state.imageSetMetadata = outputMetadata;
  },
);

export const outputImageFrameIds = new ScenarioOutput(
  "outputImageFrameIds",
  (/** @type {State & { imageSetMetadata: ImageSetMetadata[] }} */ state) => {
    let output = "";

    for (const metadata of state.imageSetMetadata) {
      const imageSetId = metadata.ImageSetID;
      /** @type {DICOMMetadata[]} */
      const instances = Object.values(metadata.Study.Series).flatMap(
        (series) => {
```

```
          return Object.values(series.Instances);
        },
      );
      const imageFrameIds = instances.flatMap((instance) =>
        instance.ImageFrames.map((frame) => frame.ID),
      );

      output += `Image set ID: ${imageSetId}\nImage frame IDs:\n
${imageFrameIds.join(
        "\n",
      )}\n\n`;
    }

    return output;
  },
);
```

Verifique os quadros de imagens (verify-steps.js). A biblioteca AWS HealthImaging Pixel Data Verification foi usada para verificação.

```
import { spawn } from "node:child_process";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
```

```
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

export const doVerify = new ScenarioInput(
```

```
  "doVerify",
  "Do you want to verify the imported images?",
  {
    type: "confirm",
    default: true,
  },
);

export const decodeAndVerifyImages = new ScenarioAction(
  "decodeAndVerifyImages",
  async (/** @type {State} */ state) => {
    if (!state.doVerify) {
      process.exit(0);
    }
    const verificationTool = "./pixel-data-verification/index.js";

    for (const metadata of state.imageSetMetadata) {
      const datastoreId = state.stackOutputs.DatastoreID;
      const imageSetId = metadata.ImageSetID;

      for (const [seriesInstanceUid, series] of Object.entries(
        metadata.Study.Series,
      )) {
        for (const [sopInstanceUid, _] of Object.entries(series.Instances)) {
          console.log(
            `Verifying image set ${imageSetId} with series ${seriesInstanceUid}
 and sop ${sopInstanceUid}`,
          );
          const child = spawn(
            "node",
            [
              verificationTool,
              datastoreId,
              imageSetId,
              seriesInstanceUid,
              sopInstanceUid,
            ],
            { stdio: "inherit" },
          );

          await new Promise((resolve, reject) => {
            child.on("exit", (code) => {
              if (code === 0) {
                resolve();
```

```
              } else {
                reject(
                  new Error(
                    `Verification tool exited with code ${code} for image set
 ${imageSetId}`,
                  ),
                );
              }
            });
          });
        }
      }
    }
  },
);
```

Destrua recursos (clean-up-steps.js).

```
import {
  CloudFormationClient,
  DeleteStackCommand,
} from "@aws-sdk/client-cloudformation";
import {
  MedicalImagingClient,
  DeleteImageSetCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
```

```
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
```

```
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

const cfnClient = new CloudFormationClient({});
const medicalImagingClient = new MedicalImagingClient({});

export const confirmCleanup = new ScenarioInput(
  "confirmCleanup",
  "Do you want to delete the created resources?",
  { type: "confirm" },
);

export const deleteImageSets = new ScenarioAction(
  "deleteImageSets",
  async (/** @type {State} */ state) => {
    const datastoreId = state.stackOutputs.DatastoreID;

    for (const metadata of state.imageSetMetadata) {
      const command = new DeleteImageSetCommand({
        datastoreId,
        imageSetId: metadata.ImageSetID,
      });

      try {
        await medicalImagingClient.send(command);
        console.log(`Successfully deleted image set ${metadata.ImageSetID}`);
      } catch (e) {
        if (e instanceof Error) {
          if (e.name === "ConflictException") {
            console.log(`Image set ${metadata.ImageSetID} already deleted`);
          }
        }
      }
    }
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);

export const deleteStack = new ScenarioAction(
  "deleteStack",
  async (/** @type {State} */ state) => {
```

```
    const stackName = state.getStackName;

    const command = new DeleteStackCommand({
      StackName: stackName,
    });

    await cfnClient.send(command);
    console.log(`Stack ${stackName} deletion initiated`);
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

Crie uma CloudFormation pilha com os recursos necessários.

```
    def deploy(self):
        """
        Deploys prerequisite resources used by the scenario. The resources are
```

```
        defined in the associated `setup.yaml` AWS CloudFormation script and are
 deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.
        """

        print("\t\tLet's deploy the stack for resource creation.")
        stack_name = q.ask("\t\tEnter a name for the stack: ", q.non_empty)

        data_store_name = q.ask(
            "\t\tEnter a name for the Health Imaging Data Store: ", q.non_empty
        )

        account_id = boto3.client("sts").get_caller_identity()["Account"]

        with open(
            "../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml"
        ) as setup_file:
            setup_template = setup_file.read()
        print(f"\t\tCreating {stack_name}.")
        stack = self.cf_resource.create_stack(
            StackName=stack_name,
            TemplateBody=setup_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=[
                {
                    "ParameterKey": "datastoreName",
                    "ParameterValue": data_store_name,
                },
                {
                    "ParameterKey": "userAccountID",
                    "ParameterValue": account_id,
                },
            ],
        )
        print("\t\tWaiting for stack to deploy. This typically takes a minute or
 two.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_create_complete")
        waiter.wait(StackName=stack.name)
        stack.load()
        print(f"\t\tStack status: {stack.stack_status}")

        outputs_dictionary = {
```

```
            output["OutputKey"]: output["OutputValue"] for output in
stack.outputs
        }
        self.input_bucket_name = outputs_dictionary["BucketName"]
        self.output_bucket_name = outputs_dictionary["BucketName"]
        self.role_arn = outputs_dictionary["RoleArn"]
        self.data_store_id = outputs_dictionary["DatastoreID"]
        return stack
```

Copie arquivos DICOM para o bucket de importação do Amazon S3.

```
    def copy_single_object(self, key, source_bucket, target_bucket,
target_directory):
        """
        Copies a single object from a source to a target bucket.

        :param key: The key of the object to copy.
        :param source_bucket: The source bucket for the copy.
        :param target_bucket: The target bucket for the copy.
        :param target_directory: The target directory for the copy.
        """
        new_key = target_directory + "/" + key
        copy_source = {"Bucket": source_bucket, "Key": key}
        self.s3_client.copy_object(
            CopySource=copy_source, Bucket=target_bucket, Key=new_key
        )
        print(f"\n\t\tCopying {key}.")

    def copy_images(
        self, source_bucket, source_directory, target_bucket, target_directory
    ):
        """
        Copies the images from the source to the target bucket using multiple
threads.

        :param source_bucket: The source bucket for the images.
        :param source_directory: Directory within the source bucket.
        :param target_bucket: The target bucket for the images.
        :param target_directory: Directory within the target bucket.
        """
```

```
        # Get list of all objects in source bucket.
        list_response = self.s3_client.list_objects_v2(
            Bucket=source_bucket, Prefix=source_directory
        )
        objs = list_response["Contents"]
        keys = [obj["Key"] for obj in objs]

        # Copy the objects in the bucket.
        for key in keys:
            self.copy_single_object(key, source_bucket, target_bucket,
 target_directory)

        print("\t\tDone copying all objects.")
```

Importe os arquivos DICOM para o armazenamento de dados do Amazon S3.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def start_dicom_import_job(
        self,
        data_store_id,
        input_bucket_name,
        input_directory,
```

```
        output_bucket_name,
        output_directory,
        role_arn,
    ):
        """
        Routine which starts a HealthImaging import job.

        :param data_store_id: The HealthImaging data store ID.
        :param input_bucket_name: The name of the Amazon S3 bucket containing the
DICOM files.
        :param input_directory: The directory in the S3 bucket containing the
DICOM files.
        :param output_bucket_name: The name of the S3 bucket for the output.
        :param output_directory: The directory in the S3 bucket to store the
output.
        :param role_arn: The ARN of the IAM role with permissions for the import.
        :return: The job ID of the import.
        """

        input_uri = f"s3://{input_bucket_name}/{input_directory}/"
        output_uri = f"s3://{output_bucket_name}/{output_directory}/"
        try:
            job = self.medical_imaging_client.start_dicom_import_job(
                jobName="examplejob",
                datastoreId=data_store_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_uri,
                outputS3Uri=output_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

Obtenha conjuntos de imagens criados pelo trabalho de importação DICOM.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_sets_for_dicom_import_job(self, datastore_id, import_job_id):
        """
        Retrieves the image sets created for an import job.

        :param datastore_id: The HealthImaging data store ID
        :param import_job_id: The import job ID
        :return: List of image set IDs
        """

        import_job = self.medical_imaging_client.get_dicom_import_job(
            datastoreId=datastore_id, jobId=import_job_id
        )

        output_uri = import_job["jobProperties"]["outputS3Uri"]

        bucket = output_uri.split("/")[2]
        key = "/".join(output_uri.split("/")[3:])

        # Try to get the manifest.
        retries = 3
        while retries > 0:
            try:
                obj = self.s3_client.get_object(
```

```
                    Bucket=bucket, Key=key + "job-output-manifest.json"
                )
                body = obj["Body"]
                break
            except ClientError as error:
                retries = retries - 1
                time.sleep(3)
        try:
            data = json.load(body)
            expression =
jmespath.compile("jobSummary.imageSetsSummary[].imageSetId")
            image_sets = expression.search(data)
        except json.decoder.JSONDecodeError as error:
            image_sets = import_job["jobProperties"]

        return image_sets


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return image_set
```

Obtenha informações sobre os quadros de imagem de conjuntos de imagens.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_frames_for_image_set(self, datastore_id, image_set_id,
 out_directory):
        """
        Get the image frames for an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param out_directory: The directory to save the file.
        :return: The image frames.
        """
        image_frames = []
        file_name = os.path.join(out_directory,
 f"{image_set_id}_metadata.json.gzip")
        file_name = file_name.replace("/", "\\\\")
        self.get_image_set_metadata(file_name, datastore_id, image_set_id)
```

```
        try:
            with gzip.open(file_name, "rb") as f_in:
                doc = json.load(f_in)
            instances = jmespath.search("Study.Series.*.Instances[].*[]", doc)
            for instance in instances:
                rescale_slope = jmespath.search("DICOM.RescaleSlope", instance)
                rescale_intercept = jmespath.search("DICOM.RescaleIntercept",
instance)
                image_frames_json = jmespath.search("ImageFrames[][]", instance)
                for image_frame in image_frames_json:
                    checksum_json = jmespath.search(
                        "max_by(PixelDataChecksumFromBaseToFullResolution,
&Width)",
                        image_frame,
                    )
                    image_frame_info = {
                        "imageSetId": image_set_id,
                        "imageFrameId": image_frame["ID"],
                        "rescaleIntercept": rescale_intercept,
                        "rescaleSlope": rescale_slope,
                        "minPixelValue": image_frame["MinPixelValue"],
                        "maxPixelValue": image_frame["MaxPixelValue"],
                        "fullResolutionChecksum": checksum_json["Checksum"],
                    }
                    image_frames.append(image_frame_info)
            return image_frames
        except TypeError:
            return {}
        except ClientError as err:
            logger.error(
                "Couldn't get image frames for image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        return image_frames


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.
```

```
        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """

        try:
            if version_id:
                image_set_metadata =
 self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set_metadata =
 self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            with open(metadata_file, "wb") as f:
                for chunk in
 image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Baixe, decodifique e verifique os quadros de imagem.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
```

```
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K
encoded pixel data.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.medical_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def download_decode_and_check_image_frames(
        self, data_store_id, image_frames, out_directory
    ):
        """
        Downloads image frames, decodes them, and uses the checksum to validate
        the decoded images.

        :param data_store_id: The HealthImaging data store ID.
        :param image_frames: A list of dicts containing image frame information.
        :param out_directory: A directory for the downloaded images.
        :return: True if the function succeeded; otherwise, False.
        """
        total_result = True
        for image_frame in image_frames:
            image_file_path = f"{out_directory}/
image_{image_frame['imageFrameId']}.jph"
            self.get_pixel_data(
                image_file_path,
                data_store_id,
                image_frame["imageSetId"],
                image_frame["imageFrameId"],
            )

            image_array = self.jph_image_to_opj_bitmap(image_file_path)
            crc32_checksum = image_frame["fullResolutionChecksum"]
            # Verify checksum.
            crc32_calculated = zlib.crc32(image_array)
            image_result = crc32_checksum == crc32_calculated
            print(
                f"\t\tImage checksum verified for {image_frame['imageFrameId']}:
 {image_result }"
            )
            total_result = total_result and image_result
        return total_result

    @staticmethod
    def jph_image_to_opj_bitmap(jph_file):
        """
        Decode the image to a bitmap using an OPENJPEG library.
        :param jph_file: The file to decode.
        :return: The decoded bitmap as an array.
        """
        # Use format 2 for the JPH file.
        params = openjpeg.utils.get_parameters(jph_file, 2)
```

```
        print(f"\n\t\tImage parameters for {jph_file}: \n\t\t{params}")

        image_array = openjpeg.utils.decode(jph_file, 2)

        return image_array
```

Limpe recursos.

```
    def destroy(self, stack):
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example
resources.
        """

        print(f"\t\tCleaning up resources and {stack.name}.")
        data_store_id = None
        for oput in stack.outputs:
            if oput["OutputKey"] == "DatastoreID":
                data_store_id = oput["OutputValue"]
        if data_store_id is not None:
            print(f"\t\tDeleting image sets in data store {data_store_id}.")
            image_sets = self.medical_imaging_wrapper.search_image_sets(
                data_store_id, {}
            )
            image_set_ids = [image_set["imageSetId"] for image_set in image_sets]

            for image_set_id in image_set_ids:
                self.medical_imaging_wrapper.delete_image_set(
                    data_store_id, image_set_id
                )
                print(f"\t\tDeleted image set with id : {image_set_id}")

        print(f"\t\tDeleting {stack.name}.")
        stack.delete()
        print("\t\tWaiting for stack removal. This may take a few minutes.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_delete_complete")
        waiter.wait(StackName=stack.name)
```

```
        print("\t\tStack delete complete.")




class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
 [{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
 self.medical_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
```

```
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        """
        try:
            delete_results = self.medical_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Marcando um armazenamento HealthImaging de dados usando um AWS SDK

Os exemplos de código a seguir mostram como marcar um armazenamento HealthImaging de dados.

Java

SDK para Java 2.x

Marcar um datastore.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                ImmutableMap.of("Deployment", "Development"));
```

A função de utilitário para marcar um recurso.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
```

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

Listar tags para um datastore.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                datastoreArn);
                if (result != null) {
                        System.out.println("Tags for resource: " +
 result.tags());
                }
```

A função de utilitário para listar as tags de um recurso.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

Desmarcar um datastore.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                Collections.singletonList("Deployment"));
```

A função de utilitário para desmarcar um recurso.

```
   public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
           String resourceArn,
           Collection<String> tagKeys) {
       try {
           UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .tagKeys(tagKeys)
                   .build();

           medicalImagingClient.untagResource(untagResourceRequest);

           System.out.println("Tags have been removed from the resource.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

Marcar um datastore.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(datastoreArn, tags);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para marcar um recurso.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
```

```
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

Listar tags para um datastore.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(datastoreArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para listar as tags de um recurso.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
```

```
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

Desmarcar um datastore.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(datastoreArn, keys);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para desmarcar um recurso.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
```

```
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //       httpStatusCode: 204,
  //        requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //        extendedRequestId: undefined,
  //        cfId: undefined,
  //        attempts: 1,
  //        totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

Marcar um datastore.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.tag_resource(data_store_arn, {"Deployment":
 "Development"})
```

A função de utilitário para marcar um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Listar tags para um datastore.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.list_tags_for_resource(data_store_arn)
```

A função de utilitário para listar as tags de um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

Desmarcar um datastore.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.untag_resource(data_store_arn, ["Deployment"])
```

A função de utilitário para desmarcar um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - ListTagsForResource
  - TagResource

- UntagResource

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Marcar um conjunto HealthImaging de imagens usando um AWS SDK

Os exemplos de código a seguir mostram como marcar um conjunto de HealthImaging imagens.

Java

SDK para Java 2.x

Marcar um conjunto de imagens

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                ImmutableMap.of("Deployment", "Development"));
```

A função de utilitário para marcar um recurso.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);
```

```
            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Listar tags para um conjunto de imagens

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                imageSetArn);
                if (result != null) {
                        System.out.println("Tags for resource: " +
 result.tags());
                }
```

A função de utilitário para listar as tags de um recurso.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
```

```
    }
```

Desmarcar um conjunto de imagens

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                Collections.singletonList("Deployment"));
```

A função de utilitário para desmarcar um recurso.

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - ListTagsForResource
  - TagResource

- UntagResource

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

JavaScript

SDK para JavaScript (v3)

Marcar um conjunto de imagens

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(imagesetArn, tags);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para marcar um recurso.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
```

```
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

Listar tags para um conjunto de imagens

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(imagesetArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para listar as tags de um recurso.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
```

```
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

Desmarcar um conjunto de imagens

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(imagesetArn, keys);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para desmarcar um recurso.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
```

```
/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //       httpStatusCode: 204,
  //        requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //        extendedRequestId: undefined,
  //        cfId: undefined,
  //        attempts: 1,
  //        totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Python

SDK para Python (Boto3)

Marcar um conjunto de imagens

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.tag_resource(image_set_arn, {"Deployment":
 "Development"})
```

A função de utilitário para marcar um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Listar tags para um conjunto de imagens

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.list_tags_for_resource(image_set_arn)
```

A função de utilitário para listar as tags de um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

Desmarcar um conjunto de imagens

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.untag_resource(image_set_arn, ["Deployment"])
```

A função de utilitário para desmarcar um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exemplos de código para HealthLake usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS HealthLake com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- HealthLake Guia do desenvolvedor — Mais informações sobre HealthLake.
- HealthLake Referência da API — Detalhes sobre todas as HealthLake ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de HealthLake uso AWS SDKs
  - Ações para HealthLake usar AWS SDKs
    - Use createFhirDatastore com um AWS SDK ou CLI
    - Use DeleteFhirDatastore com um AWS SDK ou CLI

- Use DescribeFhirDatastore com um AWS SDK ou CLI
- Use DescribeFHIRExportJobcom um AWS SDK ou CLI
- Use DescribeFHIRImportJobcom um AWS SDK ou CLI
- Use ListFhirDatastores com um AWS SDK ou CLI
- Use ListFHIRExportJobscom um AWS SDK ou CLI
- Use ListFHIRImportJobscom um AWS SDK ou CLI
- Use ListTagsForResourcecom um AWS SDK ou CLI
- Use StartFHIRExportJobcom um AWS SDK ou CLI
- Use StartFHIRImportJobcom um AWS SDK ou CLI
- Use TagResourcecom um AWS SDK ou CLI
- Use UntagResourcecom um AWS SDK ou CLI

Exemplos básicos de HealthLake uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS HealthLake com AWS SDKs.

Exemplos

- Ações para HealthLake usar AWS SDKs
  - Use createFhirDatastore com um AWS SDK ou CLI
  - Use DeleteFhirDatastore com um AWS SDK ou CLI
  - Use DescribeFhirDatastore com um AWS SDK ou CLI
  - Use DescribeFHIRExportJobcom um AWS SDK ou CLI
  - Use DescribeFHIRImportJobcom um AWS SDK ou CLI
  - Use ListFhirDatastores com um AWS SDK ou CLI
  - Use ListFHIRExportJobscom um AWS SDK ou CLI
  - Use ListFHIRImportJobscom um AWS SDK ou CLI
  - Use ListTagsForResourcecom um AWS SDK ou CLI
  - Use StartFHIRExportJobcom um AWS SDK ou CLI
  - Use StartFHIRImportJobcom um AWS SDK ou CLI
  - Use TagResourcecom um AWS SDK ou CLI
  - Use UntagResourcecom um AWS SDK ou CLI

Ações para HealthLake usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar HealthLake ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS HealthLake.

Exemplos

- Use createFhirDatastore com um AWS SDK ou CLI
- Use DeleteFhirDatastore com um AWS SDK ou CLI
- Use DescribeFhirDatastore com um AWS SDK ou CLI
- Use DescribeFHIRExportJobcom um AWS SDK ou CLI
- Use DescribeFHIRImportJobcom um AWS SDK ou CLI
- Use ListFhirDatastores com um AWS SDK ou CLI
- Use ListFHIRExportJobscom um AWS SDK ou CLI
- Use ListFHIRImportJobscom um AWS SDK ou CLI
- Use ListTagsForResourcecom um AWS SDK ou CLI
- Use StartFHIRExportJobcom um AWS SDK ou CLI
- Use StartFHIRImportJobcom um AWS SDK ou CLI
- Use TagResourcecom um AWS SDK ou CLI
- Use UntagResourcecom um AWS SDK ou CLI

Use **`createFhirDatastore com um`** AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateFHIRDatastore`.

CLI

AWS CLI

Exemplo 1: Criar um armazenamento SigV4-enabled HealthLake de dados

O `create-fhir-datastore` exemplo a seguir demonstra como criar um novo armazenamento de dados em AWS HealthLake.

```
aws healthlake create-fhir-datastore \
    --datastore-type-version R4 \
    --datastore-name "FhirTestDatastore"
```

Saída:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

Exemplo 2: Criar um SMART no armazenamento FHIR-enabled HealthLake de dados

O `create-fhir-datastore` exemplo a seguir demonstra como criar um novo SMART no armazenamento de FHIR-enabled dados em AWS HealthLake.

```
aws healthlake create-fhir-datastore \
    --datastore-name "your-data-store-name" \
    --datastore-type-version R4 \
    --preload-data-config PreloadDataType="SYNTHEA" \
    --sse-configuration '{ "KmsEncryptionConfig": {  "CmkType":
 "CUSTOMER_MANAGED_KMS_KEY", "KmsKeyId": "arn:aws:kms:us-east-1:your-account-
id:key/your-key-id" } }' \
    --identity-provider-configuration  file://
identity_provider_configuration.json
```

Conteúdo de `identity_provider_configuration.json`:

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR_V1",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
```

```
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\"]}"
}
```

Saída:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

Para obter mais informações, consulte Criação e monitoramento de um armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Consulte detalhes da API em CreateFHIRDatastore na Referência de comandos da AWS CLI .

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)
```

```
    def create_fhir_datastore(
        self,
        datastore_name: str,
        sse_configuration: dict[str, any] = None,
        identity_provider_configuration: dict[str, any] = None,
    ) -> dict[str, str]:
        """
        Creates a new HealthLake data store.
        When creating a SMART on FHIR data store, the following parameters are
required:
        - sse_configuration: The server-side encryption configuration for a SMART
on FHIR-enabled data store.
        - identity_provider_configuration: The identity provider configuration
for a SMART on FHIR-enabled data store.

        :param datastore_name: The name of the data store.
        :param sse_configuration: The server-side encryption configuration for a
SMART on FHIR-enabled data store.
        :param identity_provider_configuration: The identity provider
configuration for a SMART on FHIR-enabled data store.
        :return: A dictionary containing the data store information.
        """
        try:
            parameters = {"DatastoreName": datastore_name,
"DatastoreTypeVersion": "R4"}
            if (
                sse_configuration is not None
                and identity_provider_configuration is not None
            ):
                # Creating a SMART on FHIR-enabled data store
                parameters["SseConfiguration"] = sse_configuration
                parameters[
                    "IdentityProviderConfiguration"
                ] = identity_provider_configuration

            response =
self.health_lake_client.create_fhir_datastore(**parameters)
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't create data store %s. Here's why %s",
                datastore_name,
                err.response["Error"]["Message"],
```

```
            )
            raise
```

O código a seguir mostra um exemplo de parâmetros para um SMART no armazenamento FHIR-enabled HealthLake de dados.

```
            sse_configuration = {
                "KmsEncryptionConfig": {"CmkType": "AWS_OWNED_KMS_KEY"}
            }
            # TODO: Update the metadata to match your environment.
            metadata = {
                "issuer": "https://ehr.example.com",
                "jwks_uri": "https://ehr.example.com/.well-known/jwks.json",
                "authorization_endpoint": "https://ehr.example.com/auth/
authorize",
                "token_endpoint": "https://ehr.token.com/auth/token",
                "token_endpoint_auth_methods_supported": [
                    "client_secret_basic",
                    "foo",
                ],
                "grant_types_supported": ["client_credential", "foo"],
                "registration_endpoint": "https://ehr.example.com/auth/register",
                "scopes_supported": ["openId", "profile", "launch"],
                "response_types_supported": ["code"],
                "management_endpoint": "https://ehr.example.com/user/manage",
                "introspection_endpoint": "https://ehr.example.com/user/
introspect",
                "revocation_endpoint": "https://ehr.example.com/user/revoke",
                "code_challenge_methods_supported": ["S256"],
                "capabilities": [
                    "launch-ehr",
                    "sso-openid-connect",
                    "client-public",
                ],
            }
            # TODO: Update the IdpLambdaArn.
            identity_provider_configuration = {
                "AuthorizationStrategy": "SMART_ON_FHIR_V1",
                "FineGrainedAuthorizationEnabled": True,
                "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-
id:function:your-lambda-name",
```

```
                "Metadata": json.dumps(metadata),
            }
            data_store = self.create_fhir_datastore(
                datastore_name, sse_configuration,
identity_provider_configuration
            )
```

- Consulte detalhes da API em CreateFHIRDatastore na Referência de API do AWS SDK para Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_name = 'MyHealthLakeDataStore'
        oo_result = lo_hll->createfhirdatastore(
          iv_datastorename = iv_datastore_name
          iv_datastoretypeversion = 'R4'
        ).
        MESSAGE 'Data store created successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllinternalserverex INTO DATA(lo_internal_ex).
```

```
        lv_error = |Internal server error: { lo_internal_ex->av_err_code }-
{ lo_internal_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte CreateFhirDatastore na AWS referência da API SDK for SAP ABAP.

Use **`DeleteFhirDatastore com um`** AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteFHIRDatastore`.

CLI

AWS CLI

Como excluir um datastore FHIR

O `delete-fhir-datastore` exemplo a seguir demonstra como excluir um armazenamento de dados e todo o seu conteúdo em AWS HealthLake.

```
aws healthlake delete-fhir-datastore \
    --datastore-id (Data store ID)
```

Saída:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "DELETING",
    "DatastoreId": "(Data store ID)"
}
```

Para obter mais informações, consulte Criação e monitoramento de um armazenamento de dados FHIR < https://docs.aws.amazon.com/healthlake/latest/devguide/working-with-FHIR-healthlake.html > no Guia do AWS HealthLake desenvolvedor.

- Consulte detalhes da API em DeleteFHIRDatastore na Referência de comandos da AWS CLI .

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def delete_fhir_datastore(self, datastore_id: str) -> None:
        """
        Deletes a HealthLake data store.
        :param datastore_id: The data store ID.
        """
        try:

self.health_lake_client.delete_fhir_datastore(DatastoreId=datastore_id)
        except ClientError as err:
            logger.exception(
                "Couldn't delete data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Consulte detalhes da API em DeleteFHIRDatastore na Referência de API do AWS SDK para Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->deletefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        MESSAGE 'Data store deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        DATA(lv_error) = |Access denied: { lo_access_ex->av_err_code }-
{ lo_access_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
      CATCH /aws1/cx_hllconflictexception INTO DATA(lo_conflict_ex).
        lv_error = |Conflict error: { lo_conflict_ex->av_err_code }-
{ lo_conflict_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte DeleteFhirDatastore na AWS referência da API SDK for SAP ABAP.

Use **`DescribeFhirDatastore`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeFHIRDatastore`.

CLI

AWS CLI

Como descrever um datastore FHIR

O `describe-fhir-datastore` exemplo a seguir demonstra como encontrar as propriedades de um armazenamento de dados em AWS HealthLake.

```
aws healthlake describe-fhir-datastore \
    --datastore-id "1f2f459836ac6c513ce899f9e4f66a59"
```

Saída:

```
{
    "DatastoreProperties": {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
```

```
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
}
```

Para obter mais informações, consulte Criação e monitoramento de um armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Consulte detalhes da API em DescribeFHIRDatastore na Referência de comandos da AWS CLI .

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_datastore(self, datastore_id: str) -> dict[str, any]:
        """
        Describes a HealthLake data store.
        :param datastore_id: The data store ID.
        :return: The data store description.
        """
        try:
            response = self.health_lake_client.describe_fhir_datastore(
                DatastoreId=datastore_id
            )
```

```
            return response["DatastoreProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Consulte detalhes da API em DescribeFHIRDatastore na Referência de API do AWS SDK para Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lo_datastore_properties) = oo_result->get_datastoreproperties( ).
        IF lo_datastore_properties IS BOUND.
          DATA(lv_datastore_name) = lo_datastore_properties-
>get_datastorename( ).
          DATA(lv_datastore_status) = lo_datastore_properties-
>get_datastorestatus( ).
```

```
          MESSAGE 'Data store described successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte DescribeFhirDatastore na AWS referência da API SDK for SAP ABAP.

Use **`DescribeFHIRExportJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeFHIRExportJob`.

CLI

AWS CLI

Descrever um serviço de exportação do FHIR

O `describe-fhir-export-job` exemplo a seguir mostra como encontrar as propriedades de um trabalho de exportação FHIR em. AWS HealthLake

```
aws healthlake describe-fhir-export-job \
    --datastore-id (Data store ID) \
    --job-id 9b9a51943afaedd0a8c0c26c49135a31
```

Saída:

```
{
    "ExportJobProperties": {
        "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
        "JobStatus": "IN_PROGRESS",
        "JobId": "9009813e9d69ba7cf79bcb3468780f16",
```

```
        "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
        "EndTime": "2024-11-20T11:34:01.636000-05:00",
        "OutputDataConfig": {
            "S3Configuration": {
            "S3Uri": "s3:///(Bucket Name)/(Prefix Name)/",
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"
        }

        },
        "DatastoreId": "(Data store ID)"
    }
}
```

Para obter mais informações, consulte Exportação de arquivos de um armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Para obter detalhes da API, consulte DescribeFHIRExportJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_export_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake export job.
        :param datastore_id: The data store ID.
```

```
        :param job_id: The export job ID.
        :return: The export job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_export_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ExportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe export job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeFHIRExportJobReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirexportjob(
```

```
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_export_job_properties) = oo_result->get_exportjobproperties( ).
        IF lo_export_job_properties IS BOUND.
          DATA(lv_job_status) = lo_export_job_properties->get_jobstatus( ).
          MESSAGE |Export job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeFHIRExportJobreferência da API AWS SDK for SAP ABAP.

Use **`DescribeFHIRImportJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeFHIRImportJob`.

CLI

AWS CLI

Descrever um serviço de importação do FHIR

O `describe-fhir-import-job` exemplo a seguir mostra como aprender as propriedades de uma tarefa de importação FHIR usando. AWS HealthLake

```
aws healthlake describe-fhir-import-job \
    --datastore-id (Data store ID) \
    --job-id c145fbb27b192af392f8ce6e7838e34f
```

Saída:

```
{
    "ImportJobProperties": {
    "InputDataConfig": {
        "S3Uri": "s3:///(Bucket Name)/(Prefix Name)/"
        { "arrayitem2": 2 }
    },
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Data store ID)"
    }
}
```

Para obter mais informações, consulte Importação de arquivos para um armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Para obter detalhes da API, consulte DescribeFHIRImportJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_import_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
```

```
        Describes a HealthLake import job.
        :param datastore_id: The data store ID.
        :param job_id: The import job ID.
        :return: The import job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_import_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ImportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe import job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeFHIRImportJobReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
```

```
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirimportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_import_job_properties) = oo_result->get_importjobproperties( ).
        IF lo_import_job_properties IS BOUND.
          DATA(lv_job_status) = lo_import_job_properties->get_jobstatus( ).
          MESSAGE |Import job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeFHIRImportJobreferência da API AWS SDK for SAP ABAP.

Use **`ListFhirDatastores`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFHIRDatastores`.

CLI

AWS CLI

Como listar datastores FHIR

O `list-fhir-datastores` exemplo a seguir mostra como usar o comando e como os usuários podem filtrar os resultados com base no status do armazenamento de dados em AWS HealthLake.

```
aws healthlake list-fhir-datastores \
    --filter DatastoreStatus=ACTIVE
```

Saída:

```
{
    "DatastorePropertiesList": [
    {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
    ]
}
```

Para obter mais informações, consulte Criação e monitoramento de um armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Consulte detalhes da API em ListFHIRDatastores na Referência de comandos da AWS CLI .

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_datastores(self) -> list[dict[str, any]]:
        """
        Lists all HealthLake data stores.
        :return: A list of data store descriptions.
        """
        try:
            next_token = None
            datastores = []

            # Loop through paginated results.
            while True:
                parameters = {}
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
 self.health_lake_client.list_fhir_datastores(**parameters)
                datastores.extend(response["DatastorePropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break

            return datastores
        except ClientError as err:
            logger.exception(
                "Couldn't list data stores. Here's why %s", err.response["Error"]
["Message"]
```

```
            )
            raise
```

- Consulte detalhes da API em ListFHIRDatastores na Referência de API do AWS SDK para Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_hll->listfhirdatastores( ).
        DATA(lt_datastores) = oo_result->get_datastorepropertieslist( ).
        DATA(lv_datastore_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_datastore_count } data store(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte ListFhirDatastores no AWS SDK para referência da API SAP ABAP.

Use **ListFHIRExportJobs**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFHIRExportJobs`.

CLI

AWS CLI

Listar todos os serviços de exportação do FHIR

O exemplo `list-fhir-export-jobs` a seguir mostra como usar o comando para exibir uma lista de serviços de exportação associados a uma conta.

```
aws healthlake list-fhir-export-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z)\
    --submitted-after (DATE like 2020-10-13T19:00:00Z )\
    --job-name "FHIR-EXPORT" \
    --job-status SUBMITTED  \
    --max-results (Integer between 1 and 500)
```

Saída:

```
{
    "ExportJobPropertiesList": [
        {
            "ExportJobProperties": {
                "OutputDataConfig": {
                    "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                    "S3Configuration": {
                        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                        "KmsKeyId": "(KmsKey Id)"
                    }
                },
                "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role
 Name)",
                "JobStatus": "COMPLETED",
                "JobId": "c145fbb27b192af392f8ce6e7838e34f",
                "JobName": "FHIR-EXPORT",
```

```
                "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
                "EndTime": "2024-11-20T11:34:01.636000-05:00",
                "DatastoreId": "(Data store ID)"
            }
        }
    ]
}
```

Para obter mais informações, consulte Exportação de arquivos de um armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Para obter detalhes da API, consulte ListFHIRExportJobsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_export_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake export jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The export job name.
```

```
        :param job_status: The export job status.
        :param submitted_before: The export job submitted before the specified
date.
        :param submitted_after: The export job submitted after the specified
date.
        :return: A list of export jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
                jobs.extend(response["ExportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListFHIRExportJobsReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_export_jobs) = oo_result->get_exportjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_export_jobs ).
        MESSAGE |Found { lv_job_count } export job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListFHIRExportJobsreferência da API AWS SDK for SAP ABAP.

Use **`ListFHIRImportJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFHIRImportJobs`.

CLI

AWS CLI

Listar todos os serviços de importação do FHIR

O exemplo `list-fhir-import-jobs` a seguir mostra como usar o comando para visualizar uma lista de todos os serviços de importação associados a uma conta.

```
aws healthlake list-fhir-import-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z) \
    --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
    --job-name "FHIR-IMPORT" \
    --job-status SUBMITTED  \
    -max-results (Integer between 1 and 500)
```

Saída:

```
{
    "ImportJobPropertiesList": [
        {
            "JobId": "c0fddbf76f238297632d4aebdbfc9ddf",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2024-11-20T10:08:46.813000-05:00",
            "EndTime": "2024-11-20T10:10:09.093000-05:00",
            "DatastoreId": "(Data store ID)",
            "InputDataConfig": {
                "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
            },
            "JobOutputDataConfig": {
                "S3Configuration": {
```

```
                    "S3Uri": "s3://(Bucket Name)/
import/6407b9ae4c2def3cb6f1a46a0c599ec0-FHIR_IMPORT-
c0fddbf76f238297632d4aebdbfc9ddf/",
                    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/b7f645cb-
e564-4981-8672-9e012d1ff1a0"
                }
            },
            "JobProgressReport": {
                "TotalNumberOfScannedFiles": 1,
                "TotalSizeOfScannedFilesInMB": 0.001798,
                "TotalNumberOfImportedFiles": 1,
                "TotalNumberOfResourcesScanned": 1,
                "TotalNumberOfResourcesImported": 1,
                "TotalNumberOfResourcesWithCustomerError": 0,
                "TotalNumberOfFilesReadWithCustomerError": 0,
                "Throughput": 0.0
            },
            "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)"
        }
    ]
}
```

Para obter mais informações, consulte Importação de arquivos para o armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Para obter detalhes da API, consulte ListFHIRImportJobsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)
```

```
    def list_fhir_import_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake import jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The import job name.
        :param job_status: The import job status.
        :param submitted_before: The import job submitted before the specified
date.
        :param submitted_after: The import job submitted after the specified
date.
        :return: A list of import jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_import_jobs(**parameters)
                jobs.extend(response["ImportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
```

```
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list import jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListFHIRImportJobsReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
```

```
        DATA(lt_import_jobs) = oo_result->get_importjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_import_jobs ).
        MESSAGE |Found { lv_job_count } import job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListFHIRImportJobsreferência da API AWS SDK for SAP ABAP.

Use **`ListTagsForResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTagsForResource`.

CLI

AWS CLI

Como listar tags para um datastore

O exemplo de `list-tags-for-resource` a seguir lista as tags associadas ao datastore especificado.

```
aws healthlake list-tags-for-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe"
```

Saída:

```
{
    "tags": {
        "key": "value",
```

```
        "key1": "value1"
    }
}
```

Para obter mais informações, consulte Como marcar recursos AWS HealthLake no Guia do AWS HealthLake desenvolvedor.

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
        """
        Lists the tags for a HealthLake resource.
        :param resource_arn: The resource ARN.
        :return: The tags for the resource.
        """
        try:
            response = self.health_lake_client.list_tags_for_resource(
                ResourceARN=resource_arn
            )
            return response["Tags"]
        except ClientError as err:
            logger.exception(
                "Couldn't list tags for resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- Para obter detalhes da API, consulte a ListTagsForResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        DATA(lo_result) = lo_hll->listtagsforresource(
          iv_resourcearn = iv_resource_arn
        ).
        ot_tags = lo_result->get_tags( ).
        DATA(lv_tag_count) = lines( ot_tags ).
        MESSAGE |Found { lv_tag_count } tag(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
```

```
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTagsForResourcereferência da API AWS SDK for SAP ABAP.

Use **`StartFHIRExportJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartFHIRExportJob`.

CLI

AWS CLI

Iniciar um serviço de exportação do FHIR

O `start-fhir-export-job` exemplo a seguir mostra como iniciar um trabalho de exportação do FHIR usando o. AWS HealthLake

```
aws healthlake start-fhir-export-job \
    --output-data-config '{"S3Configuration": {"S3Uri":"s3://(Bucket Name)/
(Prefix Name)/","KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"}}' \
    --datastore-id (Data store ID) \
    --data-access-role-arn arn:aws:iam::(AWS Account ID):role/(Role Name)
```

Saída:

```
{
    "DatastoreId": "(Data store ID)",
    "JobStatus": "SUBMITTED",
    "JobId": "9b9a51943afaedd0a8c0c26c49135a31"
}
```

Para obter mais informações, consulte Exportação de arquivos de um armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Para obter detalhes da API, consulte StartFHIRExportJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_export_job(
        self,
        job_name: str,
        datastore_id: str,
        output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake export job.
        :param job_name: The export job name.
        :param datastore_id: The data store ID.
        :param output_s3_uri: The output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The export job.
        """
        try:
            response = self.health_lake_client.start_fhir_export_job(
                OutputDataConfig={
                    "S3Configuration": {"S3Uri": output_s3_uri, "KmsKeyId":
kms_key_id}
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
                JobName=job_name,
            )
```

```
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start export job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a StartFHIRExportJobReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_job_name = 'MyExportJob'
        " iv_output_s3_uri = 's3://my-bucket/export/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeExportRole'
        oo_result = lo_hll->startfhirexportjob(
          iv_jobname = iv_job_name
          io_outputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
```

```
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Export job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartFHIRExportJobreferência da API AWS SDK for SAP ABAP.

Use **`StartFHIRImportJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartFHIRImportJob`.

CLI

AWS CLI

Iniciar um trabalho de importação do FHIR

O `start-fhir-import-job` exemplo a seguir mostra como iniciar um trabalho de importação do FHIR usando o. AWS HealthLake

```
aws healthlake start-fhir-import-job \
    --input-data-config S3Uri="s3://(Bucket Name)/(Prefix Name)/" \
    --job-output-data-config '{"S3Configuration": {"S3Uri":"s3://(Bucket Name)/
(Prefix Name)/","KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"}}' \
    --datastore-id (Data store ID) \
    --data-access-role-arn "arn:aws:iam::(AWS Account ID):role/(Role Name)"
```

Saída:

```
{
    "DatastoreId": "(Data store ID)",
    "JobStatus": "SUBMITTED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f"
}
```

Para obter mais informações, consulte Importação de arquivos para um armazenamento de dados FHIR no Guia do AWS HealthLake desenvolvedor.

- Para obter detalhes da API, consulte StartFHIRImportJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_import_job(
        self,
```

```
        job_name: str,
        datastore_id: str,
        input_s3_uri: str,
        job_output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake import job.
        :param job_name: The import job name.
        :param datastore_id: The data store ID.
        :param input_s3_uri: The input S3 URI.
        :param job_output_s3_uri: The job output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The import job.
        """
        try:
            response = self.health_lake_client.start_fhir_import_job(
                JobName=job_name,
                InputDataConfig={"S3Uri": input_s3_uri},
                JobOutputDataConfig={
                    "S3Configuration": {
                        "S3Uri": job_output_s3_uri,
                        "KmsKeyId": kms_key_id,
                    }
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
            )
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start import job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a StartFHIRImportJobReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_job_name = 'MyImportJob'
        " iv_input_s3_uri = 's3://my-bucket/import/data.ndjson'
        " iv_job_output_s3_uri = 's3://my-bucket/import/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeImportRole'
        oo_result = lo_hll->startfhirimportjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_hllinputdataconfig( iv_s3uri =
 iv_input_s3_uri )
          io_joboutputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Import job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
```

```
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartFHIRImportJobreferência da API AWS SDK for SAP ABAP.

Use **`TagResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TagResource`.

CLI

AWS CLI

Como adicionar uma tag ao datastore

O exemplo de `tag-resource` a seguir mostra como adicionar uma tag a um datastore.

```
aws healthlake tag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe" \
    --tags '[{"Key": "key1", "Value": "value1"}]'
```

Este comando não produz saída.

Para obter mais informações, consulte Adicionar uma tag a um armazenamento de dados no Guia do AWS HealthLake desenvolvedor. .

- Para obter detalhes da API, consulte TagResourceem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def tag_resource(self, resource_arn: str, tags: list[dict[str, str]]) ->
None:
        """
        Tags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tags: The tags to add to the resource.
        """
        try:
            self.health_lake_client.tag_resource(ResourceARN=resource_arn,
Tags=tags)
        except ClientError as err:
            logger.exception(
                "Couldn't tag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a TagResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags
        ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a TagResourcereferência da API AWS SDK for SAP ABAP.

Use **`UntagResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UntagResource`.

CLI

AWS CLI

Como remover tags de um datastore.

O exemplo de `untag-resource` a seguir mostra como remover tags de um datastore.

```
aws healthlake untag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
b91723d65c6fdeb1d26543a49d2ed1fa" \
    --tag-keys '["key1"]'
```

Este comando não produz saída.

Para obter mais informações, consulte Remoção de tags de um armazenamento de dados no Guia do AWS HealthLake desenvolvedor.

- Para obter detalhes da API, consulte UntagResourceem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
```

```
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tag_keys: The tag keys to remove from the resource.
        """
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UntagResourceReferência da API AWS SDK for Python (Boto3).

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
```

```
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UntagResourcereferência da API AWS SDK for SAP ABAP.

Exemplos de código para o IAM usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Identity and Access Management com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do IAM: mais informações sobre o IAM.
- Referência da API IAM: detalhes sobre todas as ações disponíveis do IAM.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do IAM AWS SDKs
  - Olá, IAM
  - Aprenda as noções básicas do IAM com um AWS SDK
  - Ações para o IAM usando AWS SDKs
    - Usar AddClientIdToOpenIdConnectProvider com uma CLI
    - Usar AddRoleToInstanceProfile com uma CLI
    - Usar AddUserToGroup com uma CLI
    - Usar AttachGroupPolicy com uma CLI
    - Use AttachRolePolicycom um AWS SDK ou CLI
    - Use AttachUserPolicycom um AWS SDK ou CLI
    - Usar ChangePassword com uma CLI
    - Use CreateAccessKeycom um AWS SDK ou CLI
    - Use CreateAccountAliascom um AWS SDK ou CLI
    - Use CreateGroupcom um AWS SDK ou CLI
    - Use CreateInstanceProfilecom um AWS SDK ou CLI
    - Usar CreateLoginProfile com uma CLI
    - Usar CreateOpenIdConnectProvider com uma CLI
    - Use CreatePolicycom um AWS SDK ou CLI
    - Use CreatePolicyVersioncom um AWS SDK ou CLI
    - Use CreateRolecom um AWS SDK ou CLI
    - Use createSAMLProvider com um AWS SDK ou CLI
    - Use CreateServiceLinkedRolecom um AWS SDK ou CLI
    - Use CreateUsercom um AWS SDK ou CLI
    - Usar CreateVirtualMfaDevice com uma CLI
    - Usar DeactivateMfaDevice com uma CLI

- Use DeleteAccessKeycom um AWS SDK ou CLI
- Use DeleteAccountAliascom um AWS SDK ou CLI
- Usar DeleteAccountPasswordPolicy com uma CLI
- Use DeleteGroupcom um AWS SDK ou CLI
- Usar DeleteGroupPolicy com uma CLI
- Use DeleteInstanceProfilecom um AWS SDK ou CLI
- Usar DeleteLoginProfile com uma CLI
- Usar DeleteOpenIdConnectProvider com uma CLI
- Use DeletePolicycom um AWS SDK ou CLI
- Use DeletePolicyVersioncom um AWS SDK ou CLI
- Use DeleteRolecom um AWS SDK ou CLI
- Usar DeleteRolePermissionsBoundary com uma CLI
- Use DeleteRolePolicycom um AWS SDK ou CLI
- Use DeleteSAMLProvider com um AWS SDK ou CLI
- Use DeleteServerCertificatecom um AWS SDK ou CLI
- Use DeleteServiceLinkedRolecom um AWS SDK ou CLI
- Usar DeleteSigningCertificate com uma CLI
- Use DeleteUsercom um AWS SDK ou CLI
- Usar DeleteUserPermissionsBoundary com uma CLI
- Use DeleteUserPolicycom um AWS SDK ou CLI
- Usar DeleteVirtualMfaDevice com uma CLI
- Usar DetachGroupPolicy com uma CLI
- Use DetachRolePolicycom um AWS SDK ou CLI
- Use DetachUserPolicycom um AWS SDK ou CLI
- Usar EnableMfaDevice com uma CLI
- Use GenerateCredentialReportcom um AWS SDK ou CLI
- Usar GenerateServiceLastAccessedDetails com uma CLI
- Use GetAccessKeyLastUsedcom um AWS SDK ou CLI
- Use GetAccountAuthorizationDetailscom um AWS SDK ou CLI

- Use GetAccountPasswordPolicycom um AWS SDK ou CLI

- Use GetAccountSummarycom um AWS SDK ou CLI
- Usar GetContextKeysForCustomPolicy com uma CLI
- Usar GetContextKeysForPrincipalPolicy com uma CLI
- Use GetCredentialReportcom um AWS SDK ou CLI
- Usar GetGroup com uma CLI
- Usar GetGroupPolicy com uma CLI
- Usar GetInstanceProfile com uma CLI
- Usar GetLoginProfile com uma CLI
- Usar GetOpenIdConnectProvider com uma CLI
- Use GetPolicycom um AWS SDK ou CLI
- Use GetPolicyVersioncom um AWS SDK ou CLI
- Use GetRolecom um AWS SDK ou CLI
- Usar GetRolePolicy com uma CLI
- Usar GetSamlProvider com uma CLI
- Use GetServerCertificatecom um AWS SDK ou CLI
- Usar GetServiceLastAccessedDetails com uma CLI
- Usar GetServiceLastAccessedDetailsWithEntities com uma CLI
- Use GetServiceLinkedRoleDeletionStatuscom um AWS SDK ou CLI
- Use GetUsercom um AWS SDK ou CLI
- Usar GetUserPolicy com uma CLI
- Use ListAccessKeyscom um AWS SDK ou CLI
- Use ListAccountAliasescom um AWS SDK ou CLI
- Usar ListAttachedGroupPolicies com uma CLI
- Use ListAttachedRolePoliciescom um AWS SDK ou CLI
- Usar ListAttachedUserPolicies com uma CLI
- Usar ListEntitiesForPolicy com uma CLI
- Usar ListGroupPolicies com uma CLI
- Use ListGroupscom um AWS SDK ou CLI
- Usar ListGroupsForUser com uma CLI

- Usar ListInstanceProfiles com uma CLI

- Usar ListInstanceProfilesForRole com uma CLI
- Usar ListMfaDevices com uma CLI
- Usar ListOpenIdConnectProviders com uma CLI
- Use ListPoliciescom um AWS SDK ou CLI
- Use ListPolicyVersionscom um AWS SDK ou CLI
- Use ListRolePoliciescom um AWS SDK ou CLI
- Usar ListRoleTags com uma CLI
- Use ListRolescom um AWS SDK ou CLI
- Use ListSamlProviders com um AWS SDK ou CLI
- Use ListServerCertificatescom um AWS SDK ou CLI
- Usar ListSigningCertificates com uma CLI
- Use ListUserPoliciescom um AWS SDK ou CLI
- Usar ListUserTags com uma CLI
- Use ListUserscom um AWS SDK ou CLI
- Usar ListVirtualMfaDevices com uma CLI
- Usar PutGroupPolicy com uma CLI
- Usar PutRolePermissionsBoundary com uma CLI
- Use PutRolePolicycom um AWS SDK ou CLI
- Usar PutUserPermissionsBoundary com uma CLI
- Use PutUserPolicycom um AWS SDK ou CLI
- Usar RemoveClientIdFromOpenIdConnectProvider com uma CLI
- Usar RemoveRoleFromInstanceProfile com uma CLI
- Usar RemoveUserFromGroup com uma CLI
- Usar ResyncMfaDevice com uma CLI
- Use SetDefaultPolicyVersioncom um AWS SDK ou CLI
- Usar TagRole com uma CLI
- Usar TagUser com uma CLI
- Usar UntagRole com uma CLI
- Usar UntagUser com uma CLI

- Use UpdateAccessKeycom um AWS SDK ou CLI

- Usar UpdateAccountPasswordPolicy com uma CLI
- Usar UpdateAssumeRolePolicy com uma CLI
- Usar UpdateGroup com uma CLI
- Usar UpdateLoginProfile com uma CLI
- Usar UpdateOpenIdConnectProviderThumbprint com uma CLI
- Usar UpdateRole com uma CLI
- Usar UpdateRoleDescription com uma CLI
- Usar UpdateSamlProvider com uma CLI
- Use UpdateServerCertificatecom um AWS SDK ou CLI
- Usar UpdateSigningCertificate com uma CLI
- Use UpdateUsercom um AWS SDK ou CLI
- Use UploadServerCertificatecom um AWS SDK ou CLI
- Usar UploadSigningCertificate com uma CLI

- Cenários para o uso do IAM AWS SDKs
  - Crie e gerencie um serviço resiliente usando um AWS SDK
  - Configurar a conectividade do serviço de contêiner
  - Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
  - Criar uma API REST com integração de proxy de função
  - Crie usuários do IAM somente para leitura e leitura e gravação usando um AWS SDK
  - Como criar um serviço de contêiner para instâncias de máquinas virtuais
  - Como criar um espaço de trabalho de monitoramento gerenciado
  - Como criar um painel de monitoramento com o nome da função como uma variável
  - Como criar sua primeira função com tecnologia sem servidor
  - Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI
  - Conceitos básicos do gerenciamento de configurações
  - Conceitos básicos da proteção de dispositivos da Internet das Coisas
  - Conceitos básicos de arquivos de atributos de machine learning
  - Conceitos básicos de clusters gerenciados de kubernetes
  - Conceitos básicos de streaming gerenciado

  - Conceitos básicos de clusters de data warehouse provisionados

- Conceitos básicos da orquestração de fluxo de trabalaho
- Gerencie as chaves de acesso do IAM usando um AWS SDK
- Gerencie políticas do IAM usando um AWS SDK
- Gerencie as funções do IAM usando um AWS SDK
- Gerencie sua conta do IAM usando um AWS SDK
- Como mover segredos codificados para o armazenamento seguro de segredos
- Permite que AWS Compute Optimizer Recurso de automação para aplicar ações recomendadas
- Política para permitir a automação em toda a sua organização
- Política para permitir a automação na sua conta
- Política para conceder acesso total ao Compute Optimizer Automation para uma conta gerencial de uma organização
- Política para conceder acesso total à Automação do Compute Optimizer para uso autônomo AWS contas
- Política para conceder acesso somente leitura ao Compute Optimizer Automation para uma conta gerencial de uma organização
- Política para conceder acesso somente de leitura à Automação do Compute Optimizer para uso autônomo AWS contas
- Política para conceder permissões de perfil vinculado ao serviço para o Compute Optimization Automation
- Reverta uma versão da política do IAM usando um AWS SDK
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Configure o controle de Attribute-Based acesso para o DynamoDB usando AWS Command Line Interface v2 da
- Como configurar o gerenciamento de sistemas
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor
- Trabalhe com o DynamoDB Streams e use Time-to-Live AWS Command Line Interface v2 da
- Trabalhe com a API IAM Policy Builder usando um AWS SDK

Exemplos básicos para o uso do IAM AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Identity and Access Management com AWS SDKs.

Exemplos

- Olá, IAM
- Aprenda as noções básicas do IAM com um AWS SDK
- Ações para o IAM usando AWS SDKs
  - Usar AddClientIdToOpenIdConnectProvider com uma CLI
  - Usar AddRoleToInstanceProfile com uma CLI
  - Usar AddUserToGroup com uma CLI
  - Usar AttachGroupPolicy com uma CLI
  - Use AttachRolePolicycom um AWS SDK ou CLI
  - Use AttachUserPolicycom um AWS SDK ou CLI
  - Usar ChangePassword com uma CLI
  - Use CreateAccessKeycom um AWS SDK ou CLI
  - Use CreateAccountAliascom um AWS SDK ou CLI
  - Use CreateGroupcom um AWS SDK ou CLI
  - Use CreateInstanceProfilecom um AWS SDK ou CLI
  - Usar CreateLoginProfile com uma CLI
  - Usar CreateOpenIdConnectProvider com uma CLI
  - Use CreatePolicycom um AWS SDK ou CLI
  - Use CreatePolicyVersioncom um AWS SDK ou CLI
  - Use CreateRolecom um AWS SDK ou CLI
  - Use createSAMLProvider com um AWS SDK ou CLI
  - Use CreateServiceLinkedRolecom um AWS SDK ou CLI
  - Use CreateUsercom um AWS SDK ou CLI
  - Usar CreateVirtualMfaDevice com uma CLI
  - Usar DeactivateMfaDevice com uma CLI
  - Use DeleteAccessKeycom um AWS SDK ou CLI
  - Use DeleteAccountAliascom um AWS SDK ou CLI

- Usar DeleteAccountPasswordPolicy com uma CLI
- Use DeleteGroupcom um AWS SDK ou CLI
- Usar DeleteGroupPolicy com uma CLI
- Use DeleteInstanceProfilecom um AWS SDK ou CLI
- Usar DeleteLoginProfile com uma CLI
- Usar DeleteOpenIdConnectProvider com uma CLI
- Use DeletePolicycom um AWS SDK ou CLI
- Use DeletePolicyVersioncom um AWS SDK ou CLI
- Use DeleteRolecom um AWS SDK ou CLI
- Usar DeleteRolePermissionsBoundary com uma CLI
- Use DeleteRolePolicycom um AWS SDK ou CLI
- Use DeleteSAMLProvider com um AWS SDK ou CLI
- Use DeleteServerCertificatecom um AWS SDK ou CLI
- Use DeleteServiceLinkedRolecom um AWS SDK ou CLI
- Usar DeleteSigningCertificate com uma CLI
- Use DeleteUsercom um AWS SDK ou CLI
- Usar DeleteUserPermissionsBoundary com uma CLI
- Use DeleteUserPolicycom um AWS SDK ou CLI
- Usar DeleteVirtualMfaDevice com uma CLI
- Usar DetachGroupPolicy com uma CLI
- Use DetachRolePolicycom um AWS SDK ou CLI
- Use DetachUserPolicycom um AWS SDK ou CLI
- Usar EnableMfaDevice com uma CLI
- Use GenerateCredentialReportcom um AWS SDK ou CLI
- Usar GenerateServiceLastAccessedDetails com uma CLI
- Use GetAccessKeyLastUsedcom um AWS SDK ou CLI
- Use GetAccountAuthorizationDetailscom um AWS SDK ou CLI
- Use GetAccountPasswordPolicycom um AWS SDK ou CLI
- Use GetAccountSummarycom um AWS SDK ou CLI

- Usar GetContextKeysForCustomPolicy com uma CLI

- Usar GetContextKeysForPrincipalPolicy com uma CLI
- Use GetCredentialReportcom um AWS SDK ou CLI
- Usar GetGroup com uma CLI
- Usar GetGroupPolicy com uma CLI
- Usar GetInstanceProfile com uma CLI
- Usar GetLoginProfile com uma CLI
- Usar GetOpenIdConnectProvider com uma CLI
- Use GetPolicycom um AWS SDK ou CLI
- Use GetPolicyVersioncom um AWS SDK ou CLI
- Use GetRolecom um AWS SDK ou CLI
- Usar GetRolePolicy com uma CLI
- Usar GetSamlProvider com uma CLI
- Use GetServerCertificatecom um AWS SDK ou CLI
- Usar GetServiceLastAccessedDetails com uma CLI
- Usar GetServiceLastAccessedDetailsWithEntities com uma CLI
- Use GetServiceLinkedRoleDeletionStatuscom um AWS SDK ou CLI
- Use GetUsercom um AWS SDK ou CLI
- Usar GetUserPolicy com uma CLI
- Use ListAccessKeyscom um AWS SDK ou CLI
- Use ListAccountAliasescom um AWS SDK ou CLI
- Usar ListAttachedGroupPolicies com uma CLI
- Use ListAttachedRolePoliciescom um AWS SDK ou CLI
- Usar ListAttachedUserPolicies com uma CLI
- Usar ListEntitiesForPolicy com uma CLI
- Usar ListGroupPolicies com uma CLI
- Use ListGroupscom um AWS SDK ou CLI
- Usar ListGroupsForUser com uma CLI
- Usar ListInstanceProfiles com uma CLI
- Usar ListInstanceProfilesForRole com uma CLI

- Usar ListMfaDevices com uma CLI

- Usar ListOpenIdConnectProviders com uma CLI
- Use ListPoliciescom um AWS SDK ou CLI
- Use ListPolicyVersionscom um AWS SDK ou CLI
- Use ListRolePoliciescom um AWS SDK ou CLI
- Usar ListRoleTags com uma CLI
- Use ListRolescom um AWS SDK ou CLI
- Use ListSamlProviders com um AWS SDK ou CLI
- Use ListServerCertificatescom um AWS SDK ou CLI
- Usar ListSigningCertificates com uma CLI
- Use ListUserPoliciescom um AWS SDK ou CLI
- Usar ListUserTags com uma CLI
- Use ListUserscom um AWS SDK ou CLI
- Usar ListVirtualMfaDevices com uma CLI
- Usar PutGroupPolicy com uma CLI
- Usar PutRolePermissionsBoundary com uma CLI
- Use PutRolePolicycom um AWS SDK ou CLI
- Usar PutUserPermissionsBoundary com uma CLI
- Use PutUserPolicycom um AWS SDK ou CLI
- Usar RemoveClientIdFromOpenIdConnectProvider com uma CLI
- Usar RemoveRoleFromInstanceProfile com uma CLI
- Usar RemoveUserFromGroup com uma CLI
- Usar ResyncMfaDevice com uma CLI
- Use SetDefaultPolicyVersioncom um AWS SDK ou CLI
- Usar TagRole com uma CLI
- Usar TagUser com uma CLI
- Usar UntagRole com uma CLI
- Usar UntagUser com uma CLI
- Use UpdateAccessKeycom um AWS SDK ou CLI
- Usar UpdateAccountPasswordPolicy com uma CLI

- Usar UpdateAssumeRolePolicy com uma CLI

- Usar UpdateGroup com uma CLI
- Usar UpdateLoginProfile com uma CLI
- Usar UpdateOpenIdConnectProviderThumbprint com uma CLI
- Usar UpdateRole com uma CLI
- Usar UpdateRoleDescription com uma CLI
- Usar UpdateSamlProvider com uma CLI
- Use UpdateServerCertificatecom um AWS SDK ou CLI
- Usar UpdateSigningCertificate com uma CLI
- Use UpdateUsercom um AWS SDK ou CLI
- Use UploadServerCertificatecom um AWS SDK ou CLI
- Usar UploadSigningCertificate com uma CLI

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();
```

```
        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
            Console.WriteLine($"Created:
 {policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")
```

```
# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem iam.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
```

```
#include <iomanip>

/*
 *  A "Hello IAM" starter application which initializes an AWS Identity and
 Access Management (IAM) client
 *  and lists the IAM policies.
 *
 *  main function
 *
 *  Usage: 'hello_iam'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
                          std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                          std::setw(64) << "Description" << std::setw(12) <<
                          "CreateDate" << std::endl;
```

```
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                          policy.GetPolicyName() << std::setw(30) <<
                          policy.GetPolicyId() << std::setw(80) <<
policy.GetArn() <<
                          std::setw(64) << policy.GetDescription() <<
std::setw(12) <<
                          policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
<<
                          std::endl;
            }

            if (outcome.GetResult().GetIsTruncated()) {
                request.SetMarker(outcome.GetResult().GetMarker());
            } else {
                done = true;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access
 Management (IAM)
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 iamClient := iam.NewFromConfig(sdkConfig)
 const maxPols = 10
 fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
 result, err := iamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPols),
 })
 if err != nil {
  fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Policies) == 0 {
  fmt.Println("You don't have any policies!")
 } else {
  for _, policy := range result.Policies {
   fmt.Printf("\t%v\n", *policy.PolicyName)
  }
 }
```

```
}
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
```

```
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
  /**
   * In v3, the clients expose paginateOperationName APIs that are written using
 async generators so that you can use async iterators in a for await..of loop.
   * https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
   */
  const paginator = paginateListPolicies(
    { client, pageSize: 10 },
    // List only customer managed policies.
    { Scope: "Local" },
  );

  console.log("IAM policies defined in your account:");
  let policyCount = 0;
  for await (const page of paginator) {
```

```
    if (page.Policies) {
      for (const policy of page.Policies) {
        console.log(`${policy.PolicyName}`);
        policyCount++;
      }
    }
  }
  console.log(`Found ${policyCount} policies.`);
};
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def main():
    """
    Lists the managed policies in your AWS account using the AWS SDK for Python
 (Boto3).
    """
    iam = boto3.client("iam")

    try:
        # Get a paginator for the list_policies operation
        paginator = iam.get_paginator("list_policies")

        # Iterate through the pages of results
        for page in paginator.paginate(Scope="All", OnlyAttached=False):
```

```
            for policy in page["Policies"]:
                print(f"Policy name: {policy['PolicyName']}")
                print(f"  Policy ARN: {policy['Arn']}")
    except boto3.exceptions.BotoCoreError as e:
        print(f"Encountered an error while listing policies: {e}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ListPoliciesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-iam'
require 'logger'

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')
```

```
    paginator = @client.list_policies
    policies = []

    paginator.each_page do |page|
      policies.concat(page.policies)
    end

    if policies.empty?
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

De src/bin /hello.rs.

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
```

```
use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
struct HelloScenarioArgs {
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}
```

De src/iam -service-lib.rs.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
```

```
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- Para obter detalhes da API, consulte a ListPoliciesreferência da API AWS SDK for Rust.

Aprenda as noções básicas do IAM com um AWS SDK

Os exemplos de código a seguir mostram como criar um usuário e assumir um perfil.

> **Warning**
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

.NET

SDK para .NET

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
```

```
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
```

```
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }
```

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();
```

```
        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
```

```
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
```

```
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
 ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
 policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> PutUserPolicyAsync(string userName, string
 policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
 accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}
```

```
using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;
```

```
        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
```

```
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();

        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null
list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();
```

```
        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
```

```
            {
                Console.WriteLine($"{bucket.BucketName} created:
 {bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
 complete.");
        Console.WriteLine("Please wait while we clean up the resources we
 created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();

        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
```

```
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
```

```
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
{ BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
```

```
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
 (IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
 full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
 creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
```

```
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
```

```
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole

- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might
 be necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
```

```
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role
 to the user."
  echo_repeat "*" 88
  echo

  echo -n "Enter a name for a new IAM user: "
  get_input
  user_name=$get_input_result

  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}

  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
```

```
 echo "Creating a role named $iam_role_name with user $user_name as the
principal."

 local assume_role_policy_document="{
   \"Version\": \"2012-10-17\",
   \"Statement\": [{
       \"Effect\": \"Allow\",
       \"Principal\": {\"AWS\": \"$user_arn\"},
       \"Action\": \"sts:AssumeRole\"
       }]
   }"

 local role_arn
 role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Created IAM role named $iam_role_name"
 else
   errecho "The role failed to create. This demo will exit."
   clean_up "$user_name" "$key_name"
   return 1
 fi

 local policy_name
 policy_name=$(generate_random_name "test-policy")
 local policy_document="{
              \"Version\": \"2012-10-17\",
              \"Statement\": [{
                  \"Effect\": \"Allow\",
                  \"Action\": \"s3:ListAllMyBuckets\",
                  \"Resource\": \"arn:aws:s3:::*\"}]}"

 local policy_arn
 policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
 # shellcheck disable=SC2181
 if [[ ${?} == 0 ]]; then
   echo "Created  IAM policy named $policy_name"
 else
   errecho "The policy failed to create."
   clean_up "$user_name" "$key_name" "$iam_role_name"
   return 1
 fi
```

```
 if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
   echo "Attached policy $policy_arn to role $iam_role_name"
 else
   errecho "The policy failed to attach."
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
   return 1
 fi

 local assume_role_policy_document="{
              \"Version\": \"2012-10-17\",
              \"Statement\": [{
                  \"Effect\": \"Allow\",
                  \"Action\": \"sts:AssumeRole\",
                  \"Resource\": \"$role_arn\"}]}"

 local assume_role_policy_name
 assume_role_policy_name=$(generate_random_name "test-assume-role-")

 # shellcheck disable=SC2181
 local assume_role_policy_arn
 assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
 else
   errecho "The policy failed to create."
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
   return 1
 fi

 echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
 sleep 10
 echo_repeat "*" 88
 echo

 echo "Try to list buckets without the new user assuming the role."
 echo_repeat "*" 88
 echo

 # Set the environment variables for the created user.
```

```
 # bashsupport disable=BP2001
 export AWS_ACCESS_KEY_ID=$key_name
 # bashsupport disable=BP2001
 export AWS_SECRET_ACCESS_KEY=$key_secret

 local buckets
 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
   echo "There are $bucket_count buckets in the account. This should not have
happened."
 else
   errecho "Because the role with permissions has not been assumed, listing
buckets failed."
 fi

 echo
 echo_repeat "*" 88
 echo "Now assume the role $iam_role_name and list the buckets."
 echo_repeat "*" 88
 echo

 local credentials

 credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Assumed role $iam_role_name"
 else
   errecho "Failed to assume role."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 IFS=$'\t ' read -r -a credentials <<<"$credentials"

 export AWS_ACCESS_KEY_ID=${credentials[0]}
 export AWS_SECRET_ACCESS_KEY=${credentials[1]}
```

```
  # bashsupport disable=BP2001
  export AWS_SESSION_TOKEN=${credentials[2]}

  buckets=$(s3_list_buckets)

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets
 succeeded because of "
    echo "the assumed role."
  else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
 "$policy_arn" "$assume_role_policy_arn"
    return 1
  fi

  local result=0
  export AWS_ACCESS_KEY_ID=""
  export AWS_SECRET_ACCESS_KEY=""

  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

As funções do IAM usadas neste cenário.

```
##############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
##############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}

##############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
```

```
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
```

```
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
```

```
  echo "Creates an AWS Identity and Access Management (IAM) role."
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_json -- The assume role policy document."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
```

```
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
```

```
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
```

```
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
```

```
function usage() {
  echo "function iam_detach_role_policy"
  echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam detach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
```

```
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""
```

```
  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }
```

```
# Retrieve the calling parameters.
while getopts "u:k:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    k) access_key="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
```

```
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}

##############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy

- CreateAccessKey
- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
```

```
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                                Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
 std::endl;
        }
```

```
        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());
```

```
        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.

  request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");
```

```
        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " <<
policyName <<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);
```

```
        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
```

```
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
```

```
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
 role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
 s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
 LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
 std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}
```

```
bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
 client.DetachRolePolicy(
                    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
 std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
 client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
```

```
        }

    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/credentials"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/sts"
 "github.com/aws/smithy-go"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/iam/actions"
)

// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
//  1. Create a user who has no permissions.
//  2. Create a role that grants permission to list Amazon Simple Storage Service
//     (Amazon S3) buckets for the account.
//  3. Add a policy to let the user assume the role.
//  4. Try and fail to list buckets without permissions.
//  5. Assume the role and list S3 buckets using temporary credentials.
//  6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig      aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper  actions.PolicyWrapper
 roleWrapper    actions.RoleWrapper
 userWrapper    actions.UserWrapper
 questioner     demotools.IQuestioner
 helper         IScenarioHelper
 isTestRun      bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
```

```
 return AssumeRoleScenario{
  sdkConfig:      sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration
 to
// another configuration. This is used to attach the middleware stubber to
 clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
   log.Println(r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
 demo.")
 log.Println(strings.Repeat("-", 88))

 user := scenario.CreateUser(ctx)
 accessKey := scenario.CreateAccessKey(ctx, user)
 role := scenario.CreateRoleAndPolicies(ctx, user)
 noPermsConfig := scenario.ListBucketsWithoutPermissions(ctx, accessKey)
 scenario.ListBucketsWithAssumedRole(ctx, noPermsConfig, role)
 scenario.Cleanup(ctx, user, role)
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser(ctx context.Context) *types.User {
 log.Println("Let's create an example user with no permissions.")
 userName := scenario.questioner.Ask("Enter a name for the example user:",
 demotools.NotEmpty{})
 user, err := scenario.userWrapper.GetUser(ctx, userName)
 if err != nil {
  panic(err)
 }
 if user == nil {
  user, err = scenario.userWrapper.CreateUser(ctx, userName)
  if err != nil {
   panic(err)
  }
  log.Printf("Created user %v.\n", *user.UserName)
 } else {
  log.Printf("User %v already exists.\n", *user.UserName)
 }
 log.Println(strings.Repeat("-", 88))
 return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(ctx context.Context, user
 *types.User) *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(ctx, *user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
 buckets for
```

```
// the current account and attaches the policy to a newly created role. It also
 adds an
// inline policy to the specified user that grants the user permission to assume
 the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(ctx context.Context,
 user *types.User) *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err := scenario.roleWrapper.CreateRole(ctx,
 scenario.helper.GetName(), *user.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  ctx, scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"},
 "arn:aws:s3:::*")
 if err != nil {
  panic(err)
 }
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(ctx, *listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {
  panic(err)
 }
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(ctx, *user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}
```

```
// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
 access key
// credentials and tries to list buckets for the account. Because the user does
 not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(ctx
 context.Context, accessKey *types.AccessKey) *aws.Config {
 log.Println("Let's try to list buckets without permissions. This should return
 an AccessDenied error.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 noPermsConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
  ))
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

 s3Client := s3.NewFromConfig(noPermsConfig)
 _, err = s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n"
 +
     "the ListBuckets call was made without permissions.")
   default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
   }
  }
 } else {
  log.Println("Expected AccessDenied error when calling ListBuckets without
 permissions,\n" +
   "but the call succeeded. Continuing the example anyway...")
```

```
 }
 log.Println(strings.Repeat("-", 88))
 return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
//  1. Creates an AWS Security Token Service (AWS STS) client from the config
 created from
//     the user's access key credentials.
//  2. Gets temporary credentials by assuming the role that grants permission to
 list the
//     buckets.
//  3. Creates an Amazon S3 client from the temporary credentials.
//  4. Lists buckets for the account. Because the temporary credentials are
 generated by
//     assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(ctx
 context.Context, noPermsConfig *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and
 try again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(ctx, &sts.AssumeRoleInput{
  RoleArn:         role.Arn,
  RoleSessionName: aws.String("AssumeRoleExampleSession"),
  DurationSeconds: aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
 )
 if err != nil {
  panic(err)
 }
```

```
 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n"
 +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(ctx context.Context, user *types.User,
 role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
  policies, err := scenario.roleWrapper.ListAttachedRolePolicies(ctx,
 *role.RoleName)
  if err != nil {
   panic(err)
  }
  for _, policy := range policies {
   err = scenario.roleWrapper.DetachRolePolicy(ctx, *role.RoleName,
 *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   err = scenario.policyWrapper.DeletePolicy(ctx, *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   log.Printf("Detached policy %v from role %v and deleted the policy.\n",
    *policy.PolicyName, *role.RoleName)
  }
  err = scenario.roleWrapper.DeleteRole(ctx, *role.RoleName)
```

```
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted role %v.\n", *role.RoleName)

  userPols, err := scenario.userWrapper.ListUserPolicies(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, userPol := range userPols {
   err = scenario.userWrapper.DeleteUserPolicy(ctx, *user.UserName, userPol)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
  }
  keys, err := scenario.userWrapper.ListAccessKeys(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, key := range keys {
   err = scenario.userWrapper.DeleteAccessKey(ctx, *user.UserName,
 *key.AccessKeyId)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
 *user.UserName)
  }
  err = scenario.userWrapper.DeleteUser(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted user %v.\n", *user.UserName)
  log.Println(strings.Repeat("-", 88))
 }

}

// IScenarioHelper abstracts input and wait functions from a scenario so that
 they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
```

```
 Pause(secs int)
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper *ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}
```

Defina um struct que encapsule as ações de conta.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}
```

```
// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

Defina um struct que encapsule as ações de política.

```


import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
```

```
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
```

```
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

Defina um struct que encapsule as ações de perfil.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
```

```
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
```

```
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context,
 serviceName string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
```

```
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn
 string, roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
```

```
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx,
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error
 {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

Defina um struct que encapsule as ações de usuário.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
```

```
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
```

```
 return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
```

```
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx,
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
```

```
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error
 {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName
 string) (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
```

```
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser

- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações do usuário do IAM.

```
/*
  To run this Java V2 code example, set up your development environment,
 including your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3
 Service client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
```

```
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = """

                Usage:
                    <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

                Where:
                    username - The name of the IAM user to create.\s
                    policyName - The name of the policy to create.\s
                    roleName - The name of the role to create.\s
                    roleSessionName - The name of the session required for the
assumeRole operation.\s
                    bucketName - The name of the Amazon S3 bucket from which
objects are read.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        String policyName = args[1];
        String roleName = args[2];
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
```

```
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
```

```
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String
json) {

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());
            return response.role().arn();

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

```
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

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

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
```

```
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

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

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code
example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();
```

```
            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRole(IamClient iam, String roleName, String polArn)
{

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
```

```
                .policyArn(polArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(polArn)
                .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted " + polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

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

public static void deleteKey(IamClient iam, String username, String
accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                .accessKeyId(accessKey)
                .userName(username)
                .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
                " from user " + username);

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

```
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário e um perfil do IAM que conceda permissão para listar os buckets do Amazon S3. O usuário só tem direitos para assumir a função. Após assumir a função, use credenciais temporárias para listar os buckets para a conta.

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "iam_basic_test_username";
const policyName = "iam_basic_test_policy";
const roleName = "iam_basic_test_role";

/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
```

```
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    }
    throw caught;
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }
```

```
 // Create an access key. This key is used to authenticate the new user to
 // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
(AWS STS).
 // It's not best practice to use access keys. For more information, see
https://aws.amazon.com/iam/resources/best-practices/.
 const createAccessKeyResponse = await iamClient.send(
   new CreateAccessKeyCommand({ UserName: userName }),
 );

 if (
   !createAccessKeyResponse.AccessKey?.AccessKeyId ||
   !createAccessKeyResponse.AccessKey?.SecretAccessKey
 ) {
   throw new Error("Access key not created");
 }

 const {
   AccessKey: { AccessKeyId, SecretAccessKey },
 } = createAccessKeyResponse;

 let s3Client = new S3Client({
   credentials: {
     accessKeyId: AccessKeyId,
     secretAccessKey: SecretAccessKey,
   },
 });

 // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
 // thrown while the user and access keys are still stabilizing.
 await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
   try {
     return await listBuckets(s3Client);
   } catch (err) {
     if (err instanceof Error && err.name === "InvalidAccessKeyId") {
       throw err;
     }
   }
 });

 // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
 // is thrown while the user and access keys are still stabilizing.
 const { Role } = await retry(
   {
```

```
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
                  // Allow the previously created user to assume this role.
                  AWS: User.Arn,
                },
                Action: "sts:AssumeRole",
              },
            ],
          }),
          RoleName: roleName,
        }),
      ),
  );

  if (!Role) {
    throw new Error("Role not created");
  }

  // Create a policy that allows the user to list S3 buckets.
  const { Policy: listBucketPolicy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Action: ["s3:ListAllMyBuckets"],
            Resource: "*",
          },
        ],
      }),
      PolicyName: policyName,
    }),
  );
```

```
if (!listBucketPolicy) {
  throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
  new AttachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

// Assume the role.
const stsClient = new STSClient({
  credentials: {
    accessKeyId: AccessKeyId,
    secretAccessKey: SecretAccessKey,
  },
});

// Retry the assume role operation until it succeeds.
const { Credentials } = await retry(
  { intervalInMs: 2000, maxRetries: 60 },
  () =>
    stsClient.send(
      new AssumeRoleCommand({
        RoleArn: Role.Arn,
        RoleSessionName: `iamBasicScenarioSession-${Math.floor(
          Math.random() * 1000000,
        )}`,
        DurationSeconds: 900,
      }),
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
```

```
      sessionToken: Credentials.SessionToken,
    },
  });

  // List the S3 buckets again.
  // Retry the list buckets operation until it succeeds. AccessDenied might
  // be thrown while the role policy is still stabilizing.
  await retry({ intervalInMs: 2000, maxRetries: 120 }, () =>
    listBuckets(s3Client),
  );

  // Clean up.
  await iamClient.send(
    new DetachRolePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeletePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
    }),
  );

  await iamClient.send(
    new DeleteRoleCommand({
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeleteAccessKeyCommand({
      UserName: userName,
      AccessKeyId,
    }),
  );

  await iamClient.send(
    new DeleteUserCommand({
      UserName: userName,
    }),
  );
};
```

```
/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações do usuário do IAM.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
 read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
```

```
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:*"
                ],
                "Resource": "*"
            }
        ]
```

```
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
```

```
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient = StsClient.fromEnvironment { region = "us-east-1" }
```

```
    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials = StaticCredentialsProvider {
        accessKeyId = key
        secretAccessKey = secKey
        sessionToken = secToken
    }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 = S3Client.fromEnvironment {
        region = "us-east-1"
        credentialsProvider = staticCredentials
    }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
```

```
    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
```

```
    return jsonParser.parse(reader)
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
```

```
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("--------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("--------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
```

```
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
```

```
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário e um perfil do IAM que conceda permissão para listar os buckets do Amazon S3. O usuário só tem direitos para assumir a função. Após assumir a função, use credenciais temporárias para listar os buckets para a conta.

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, roles, and
 policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise
```

```
    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
```

```
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the
role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume
"
            f"the role."
        )
    except ClientError as error:
        print(
```

```
            f"Couldn't create an inline policy for user {user.name}. Here's why:
 "
            f"{error.response['Error']['Message']}"
        )
        raise

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
    Uses the temporary credentials from the role to list the buckets that are
 owned
    by the assumed role's account.
```

```
    :param user_key: The access key of a user that has permission to assume the
role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise
```

```
def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
```

```
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
 "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário e um perfil do IAM que conceda permissão para listar os buckets do Amazon S3. O usuário só tem direitos para assumir a função. Após assumir a função, use credenciais temporárias para listar os buckets para a conta.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts('Give AWS time to propagate resources...')
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info('Tried and failed to create demo user.')
    @logger.info("\t#{e.code}: #{e.message}")
```

```
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Principal: { 'AWS': user.arn },
        Action: 'sts:AssumeRole'
      }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
```

```
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account,
and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 's3:ListAllMyBuckets',
        Resource: 'arn:aws:s3:::*'
      }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
```

```
 # @return [Aws::IAM::UserPolicy] The newly created policy.
 def create_user_policy(policy_name, user, role)
   policy_document = {
     Version: '2012-10-17',
     Statement: [{
       Effect: 'Allow',
       Action: 'sts:AssumeRole',
       Resource: role.arn
     }]
   }.to_json
   @iam_client.put_user_policy(
     user_name: user.user_name,
     policy_name: policy_name,
     policy_document: policy_document
   )
   puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Creates an Amazon S3 resource with specified credentials. This is separated
into a
 # factory function so that it can be mocked for unit testing.
 #
 # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able
to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
   s3_resource.buckets.each do |bucket|
     @logger.info "\t#{bucket.name}"
```

```
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == 'AccessDenied'
      puts('Attempt to list buckets with no permissions: AccessDenied.')
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
  # This is separated into a factory function so that it can be mocked for unit
testing.
  #
  # @param key_id [String] The ID of the access key used by the STS client.
  # @param key_secret [String] The secret part of the access key used by the STS
client.
  def create_sts_client(key_id, key_secret)
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end

  # Deletes a role. If the role has policies attached, they are detached and
```

```
  # deleted before the role is deleted.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
 role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
 policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
 why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are
 deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
```

```
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
 role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - AttachRolePolicy

- CreateAccessKey
- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
 as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
 =
        initialize_variables().await;
```

```
    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
```

```
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}",
 "iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
```

```
    );

    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
            .await?;
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
    let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
        .await?;
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
        }
    }

    let sts_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    let sts_client: stsClient = stsClient::new(&sts_config);
    sleep(Duration::from_secs(10)).await;
    let assumed_role = sts_client
```

```
        .assume_role()
        .role_arn(assume_role_role.arn())
        .role_session_name(format!("iam_demo_assumerole_session_{uuid}"))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
```

```
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole

- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Ações para o IAM usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do IAM com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do IAM e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do IAM AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Identity and Access Management.

Exemplos

- Usar AddClientIdToOpenIdConnectProvider com uma CLI
- Usar AddRoleToInstanceProfile com uma CLI
- Usar AddUserToGroup com uma CLI
- Usar AttachGroupPolicy com uma CLI
- Use AttachRolePolicycom um AWS SDK ou CLI
- Use AttachUserPolicycom um AWS SDK ou CLI
- Usar ChangePassword com uma CLI
- Use CreateAccessKeycom um AWS SDK ou CLI
- Use CreateAccountAliascom um AWS SDK ou CLI
- Use CreateGroupcom um AWS SDK ou CLI
- Use CreateInstanceProfilecom um AWS SDK ou CLI
- Usar CreateLoginProfile com uma CLI
- Usar CreateOpenIdConnectProvider com uma CLI
- Use CreatePolicycom um AWS SDK ou CLI

- Use CreatePolicyVersioncom um AWS SDK ou CLI
- Use CreateRolecom um AWS SDK ou CLI
- Use createSAMLProvider com um AWS SDK ou CLI
- Use CreateServiceLinkedRolecom um AWS SDK ou CLI
- Use CreateUsercom um AWS SDK ou CLI
- Usar CreateVirtualMfaDevice com uma CLI
- Usar DeactivateMfaDevice com uma CLI
- Use DeleteAccessKeycom um AWS SDK ou CLI
- Use DeleteAccountAliascom um AWS SDK ou CLI
- Usar DeleteAccountPasswordPolicy com uma CLI
- Use DeleteGroupcom um AWS SDK ou CLI
- Usar DeleteGroupPolicy com uma CLI
- Use DeleteInstanceProfilecom um AWS SDK ou CLI
- Usar DeleteLoginProfile com uma CLI
- Usar DeleteOpenIdConnectProvider com uma CLI
- Use DeletePolicycom um AWS SDK ou CLI
- Use DeletePolicyVersioncom um AWS SDK ou CLI
- Use DeleteRolecom um AWS SDK ou CLI
- Usar DeleteRolePermissionsBoundary com uma CLI
- Use DeleteRolePolicycom um AWS SDK ou CLI
- Use DeleteSAMLProvider com um AWS SDK ou CLI
- Use DeleteServerCertificatecom um AWS SDK ou CLI
- Use DeleteServiceLinkedRolecom um AWS SDK ou CLI
- Usar DeleteSigningCertificate com uma CLI
- Use DeleteUsercom um AWS SDK ou CLI
- Usar DeleteUserPermissionsBoundary com uma CLI
- Use DeleteUserPolicycom um AWS SDK ou CLI
- Usar DeleteVirtualMfaDevice com uma CLI
- Usar DetachGroupPolicy com uma CLI
- Use DetachRolePolicycom um AWS SDK ou CLI

- Use DetachUserPolicycom um AWS SDK ou CLI
- Usar EnableMfaDevice com uma CLI
- Use GenerateCredentialReportcom um AWS SDK ou CLI
- Usar GenerateServiceLastAccessedDetails com uma CLI
- Use GetAccessKeyLastUsedcom um AWS SDK ou CLI
- Use GetAccountAuthorizationDetailscom um AWS SDK ou CLI
- Use GetAccountPasswordPolicycom um AWS SDK ou CLI
- Use GetAccountSummarycom um AWS SDK ou CLI
- Usar GetContextKeysForCustomPolicy com uma CLI
- Usar GetContextKeysForPrincipalPolicy com uma CLI
- Use GetCredentialReportcom um AWS SDK ou CLI
- Usar GetGroup com uma CLI
- Usar GetGroupPolicy com uma CLI
- Usar GetInstanceProfile com uma CLI
- Usar GetLoginProfile com uma CLI
- Usar GetOpenIdConnectProvider com uma CLI
- Use GetPolicycom um AWS SDK ou CLI
- Use GetPolicyVersioncom um AWS SDK ou CLI
- Use GetRolecom um AWS SDK ou CLI
- Usar GetRolePolicy com uma CLI
- Usar GetSamlProvider com uma CLI
- Use GetServerCertificatecom um AWS SDK ou CLI
- Usar GetServiceLastAccessedDetails com uma CLI
- Usar GetServiceLastAccessedDetailsWithEntities com uma CLI
- Use GetServiceLinkedRoleDeletionStatuscom um AWS SDK ou CLI
- Use GetUsercom um AWS SDK ou CLI
- Usar GetUserPolicy com uma CLI
- Use ListAccessKeyscom um AWS SDK ou CLI
- Use ListAccountAliasescom um AWS SDK ou CLI
- Usar ListAttachedGroupPolicies com uma CLI

- Use ListAttachedRolePoliciescom um AWS SDK ou CLI
- Usar ListAttachedUserPolicies com uma CLI
- Usar ListEntitiesForPolicy com uma CLI
- Usar ListGroupPolicies com uma CLI
- Use ListGroupscom um AWS SDK ou CLI
- Usar ListGroupsForUser com uma CLI
- Usar ListInstanceProfiles com uma CLI
- Usar ListInstanceProfilesForRole com uma CLI
- Usar ListMfaDevices com uma CLI
- Usar ListOpenIdConnectProviders com uma CLI
- Use ListPoliciescom um AWS SDK ou CLI
- Use ListPolicyVersionscom um AWS SDK ou CLI
- Use ListRolePoliciescom um AWS SDK ou CLI
- Usar ListRoleTags com uma CLI
- Use ListRolescom um AWS SDK ou CLI
- Use ListSamlProviders com um AWS SDK ou CLI
- Use ListServerCertificatescom um AWS SDK ou CLI
- Usar ListSigningCertificates com uma CLI
- Use ListUserPoliciescom um AWS SDK ou CLI
- Usar ListUserTags com uma CLI
- Use ListUserscom um AWS SDK ou CLI
- Usar ListVirtualMfaDevices com uma CLI
- Usar PutGroupPolicy com uma CLI
- Usar PutRolePermissionsBoundary com uma CLI
- Use PutRolePolicycom um AWS SDK ou CLI
- Usar PutUserPermissionsBoundary com uma CLI
- Use PutUserPolicycom um AWS SDK ou CLI
- Usar RemoveClientIdFromOpenIdConnectProvider com uma CLI
- Usar RemoveRoleFromInstanceProfile com uma CLI
- Usar RemoveUserFromGroup com uma CLI

- Usar ResyncMfaDevice com uma CLI
- Use SetDefaultPolicyVersioncom um AWS SDK ou CLI
- Usar TagRole com uma CLI
- Usar TagUser com uma CLI
- Usar UntagRole com uma CLI
- Usar UntagUser com uma CLI
- Use UpdateAccessKeycom um AWS SDK ou CLI
- Usar UpdateAccountPasswordPolicy com uma CLI
- Usar UpdateAssumeRolePolicy com uma CLI
- Usar UpdateGroup com uma CLI
- Usar UpdateLoginProfile com uma CLI
- Usar UpdateOpenIdConnectProviderThumbprint com uma CLI
- Usar UpdateRole com uma CLI
- Usar UpdateRoleDescription com uma CLI
- Usar UpdateSamlProvider com uma CLI
- Use UpdateServerCertificatecom um AWS SDK ou CLI
- Usar UpdateSigningCertificate com uma CLI
- Use UpdateUsercom um AWS SDK ou CLI
- Use UploadServerCertificatecom um AWS SDK ou CLI
- Usar UploadSigningCertificate com uma CLI

Usar **`AddClientIdToOpenIdConnectProvider`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddClientIdToOpenIdConnectProvider`.

CLI

AWS CLI

Para adicionar um ID de cliente (público) a um provedor Open-ID Connect (OIDC)

O comando `add-client-id-to-open-id-connect-provider` a seguir adiciona o ID do cliente `my-application-ID` ao provedor OIDC denominado `server.example.com`.

```
aws iam add-client-id-to-open-id-connect-provider \
    --client-id my-application-ID \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

Este comando não produz saída.

Para criar um provedor OIDC, use o comando `create-open-id-connect-provider`.

Para obter mais informações, consulte Criar provedores de identidade OpenID Connect (OIDC) no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte AddClientIdToOpenIdConnectProviderem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando adiciona o ID do cliente (ou público) **`my-application-ID`** ao provedor OIDC existente denominado **`server.example.com`**.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Para obter detalhes da API, consulte AddClientIdToOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando adiciona o ID do cliente (ou público) **`my-application-ID`** ao provedor OIDC existente denominado **`server.example.com`**.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Para obter detalhes da API, consulte AddClientIdToOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AddRoleToInstanceProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddRoleToInstanceProfile`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Conceitos básicos de streaming gerenciado
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

CLI

AWS CLI

Adicionar um perfil a um perfil de instância

O comando `add-role-to-instance-profile` a seguir adiciona o perfil denominado `S3Access` ao perfil de instância denominado `Webserver`.

```
aws iam add-role-to-instance-profile \
    --role-name S3Access \
    --instance-profile-name Webserver
```

Este comando não produz saída.

Para criar um perfil de instância, use o comando `create-instance-profile`.

Para obter mais informações, consulte Use um perfil do IAM para conceder permissões a aplicações em execução em instâncias do Amazon EC2 no AWS Guia do usuário do IAM.

- Para obter detalhes da API, consulte AddRoleToInstanceProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando adiciona o perfil denominado **`S3Access`** a um perfil de instância existente denominado **`webserver`**. Para criar o perfil de instância, use o comando **`New-IAMInstanceProfile`**. Depois de criar o perfil de instância e associá-lo a um perfil usando

esse comando, você pode anexá-lo a uma instância do EC2. Para isso, use o cmdlet **New-EC2Instance** com o parâmetro **InstanceProfile_Arn** ou **InstanceProfile-Name** para executar a nova instância.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName
 "webserver"
```

- Para obter detalhes da API, consulte AddRoleToInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando adiciona o perfil denominado **S3Access** a um perfil de instância existente denominado **webserver**. Para criar o perfil de instância, use o comando **New-IAMInstanceProfile**. Depois de criar o perfil de instância e associá-lo a um perfil usando esse comando, você pode anexá-lo a uma instância do EC2. Para isso, use o cmdlet **New-EC2Instance** com o parâmetro **InstanceProfile_Arn** ou **InstanceProfile-Name** para executar a nova instância.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName
 "webserver"
```

- Para obter detalhes da API, consulte AddRoleToInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **AddUserToGroup** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddUserToGroup`.

CLI

AWS CLI

Como adicionar um usuário a um grupo do IAM

O comando `add-user-to-group`, apresentado a seguir, adiciona um usuário do IAM denominado `Bob` ao grupo do IAM denominado `Admins`.

```
aws iam add-user-to-group \
    --user-name Bob \
    --group-name Admins
```

Este comando não produz saída.

Para obter mais informações, consulte Adicionar e remover usuários de um grupo de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte AddUserToGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando adiciona o usuário chamado **Bob** ao grupo denominado **Admins**.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Para obter detalhes da API, consulte AddUserToGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando adiciona o usuário chamado **Bob** ao grupo denominado **Admins**.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Para obter detalhes da API, consulte AddUserToGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`AttachGroupPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AttachGroupPolicy`.

CLI

AWS CLI

Anexar uma política gerenciada a um grupo do IAM

O `attach-group-policy` comando a seguir anexa a política AWS gerenciada nomeada `ReadOnlyAccess` ao grupo IAM chamado`Finance`.

```
aws iam attach-group-policy \
```

```
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --group-name Finance
```

Este comando não produz saída.

Para obter mais informações, consulte Políticas gerenciadas e em linha no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte AttachGroupPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo anexa a política gerenciada pelo cliente denominada **TesterPolicy** ao grupo do IAM **Testers**. Os usuários desse grupo são imediatamente afetados pelas permissões definidas na versão padrão dessa política.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Exemplo 2: Este exemplo anexa a política AWS gerenciada nomeada **AdministratorAccess** ao grupo **Admins** do IAM. Os usuários desse grupo são imediatamente afetados pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Para obter detalhes da API, consulte AttachGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo anexa a política gerenciada pelo cliente denominada **TesterPolicy** ao grupo do IAM **Testers**. Os usuários desse grupo são imediatamente afetados pelas permissões definidas na versão padrão dessa política.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Exemplo 2: Este exemplo anexa a política AWS gerenciada nomeada **`AdministratorAccess`** ao grupo **`Admins`** do IAM. Os usuários desse grupo são imediatamente afetados pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Para obter detalhes da API, consulte AttachGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`AttachRolePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AttachRolePolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Criar uma API REST com integração de proxy de função
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Como criar um espaço de trabalho de monitoramento gerenciado
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de arquivos de atributos de machine learning
- Conceitos básicos de clusters gerenciados de kubernetes
- Conceitos básicos de streaming gerenciado
- Conceitos básicos da orquestração de fluxo de trabalaho
- Gerenciar funções
- Como mover segredos codificados para o armazenamento seguro de segredos
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

- Como configurar o gerenciamento de sistemas
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor
- Trabalhe com Streams e Time-to-Live

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte AttachRolePolicya Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
```

```
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam attach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
  return 1
```

```
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte AttachRolePolicyem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                      roleName << ": " << list_outcome.GetError().GetMessage() <<
                      std::endl;
            return false;
        }

        const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
```

```
        if (std::any_of(policies.cbegin(), policies.cend(),
                        [=](const Aws::IAM::Model::AttachedPolicy &policy) {
                                return policy.GetPolicyArn() == policyArn;
                        })) {
            std::cout << "Policy " << policyArn <<
                      " is already attached to role " << roleName << std::endl;
            return true;
        }

        done = !list_outcome.GetResult().GetIsTruncated();
        list_request.SetMarker(list_outcome.GetResult().GetMarker());
    }

    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(roleName);
    request.SetPolicyArn(policyArn);

    Aws::IAM::Model::AttachRolePolicyOutcome outcome =
 iam.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to attach policy " << policyArn << " to role " <<
                  roleName << ": " << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        std::cout << "Successfully attached policy " << policyArn << " to role "
 <<
                  roleName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AttachRolePolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como anexar uma política gerenciada a um perfil do IAM

O `attach-role-policy` comando a seguir anexa a política AWS gerenciada nomeada `ReadOnlyAccess` à função do IAM chamada`ReadOnlyRole`.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --role-name ReadOnlyRole
```

Este comando não produz saída.

Para obter mais informações, consulte Políticas gerenciadas e em linha no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte AttachRolePolicyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
```

```
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn
 string, roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte AttachRolePolicya Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import
 software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn) {
```

```
        try {
            ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
 iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
 this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                    " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte AttachRolePolicya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Anexe a política.

```
import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte AttachRolePolicya Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    var params = {
      PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
      RoleName: process.argv[2],
    };
    iam.attachRolePolicy(params, function (err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
```

```
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte AttachRolePolicya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
```

```
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- Para obter detalhes da API, consulte a AttachRolePolicyreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- Para obter detalhes da API, consulte AttachRolePolicya Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo anexa a política AWS gerenciada nomeada **`SecurityAudit`** à função **`CoSecurityAuditors`** do IAM. Os usuários que assumem esse perfil são imediatamente afetados pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- Para obter detalhes da API, consulte AttachRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo anexa a política AWS gerenciada nomeada **`SecurityAudit`** à função **`CoSecurityAuditors`** do IAM. Os usuários que assumem esse perfil são imediatamente afetados pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- Para obter detalhes da API, consulte AttachRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Anexar uma política a uma função usando o objeto Policy do Boto3.

```
def attach_to_role(role_name, policy_arn):
```

```
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

Anexar uma política a uma função usando o objeto Role do Boto3.

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

- Para obter detalhes da API, consulte a AttachRolePolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte AttachRolePolicya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name())
        .policy_arn(policy.arn().unwrap_or_default())
        .send()
        .await
```

```
}
```

- Para obter detalhes da API, consulte a AttachRolePolicyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->attachrolepolicy(
          iv_rolename = iv_role_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy attached to role successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role or policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AttachRolePolicyreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func attachRolePolicy(role: String, policyArn: String) async throws {
        let input = AttachRolePolicyInput(
            policyArn: policyArn,
            roleName: role
        )
        do {
            _ = try await client.attachRolePolicy(input: input)
        } catch {
            print("ERROR: Attaching a role policy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte AttachRolePolicya referência da API AWS SDK for Swift.

Use **`AttachUserPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AttachUserPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar usuários somente leitura e leitura/gravação usando

CLI

AWS CLI

Como anexar uma política gerenciada a um usuário do IAM

O `attach-user-policy` comando a seguir anexa a política AWS gerenciada nomeada `AdministratorAccess` ao usuário do IAM chamado`Alice`.

```
aws iam attach-user-policy \
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
```

```
    --user-name Alice
```

Este comando não produz saída.

Para obter mais informações, consulte Políticas gerenciadas e em linha no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte AttachUserPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo anexa a política AWS gerenciada nomeada **AmazonCognitoPowerUser** ao usuário **Bob** do IAM. O usuário é imediatamente afetado pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- Para obter detalhes da API, consulte AttachUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo anexa a política AWS gerenciada nomeada **AmazonCognitoPowerUser** ao usuário **Bob** do IAM. O usuário é imediatamente afetado pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- Para obter detalhes da API, consulte AttachUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise
```

- Para obter detalhes da API, consulte a AttachUserPolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Attaches a policy to a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_user(user_name, policy_arn)
    @iam_client.attach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
  end
```

- Para obter detalhes da API, consulte AttachUserPolicya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;
```

```
    Ok(())
}
```

- Para obter detalhes da API, consulte a AttachUserPolicyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->attachuserpolicy(
          iv_username = iv_user_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy attached to user successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User or policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AttachUserPolicyreferência da API AWS SDK for SAP ABAP.

Usar **`ChangePassword`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ChangePassword`.

CLI

AWS CLI

Alterar a senha do usuário do IAM

Para alterar a senha do usuário do IAM, recomendamos usar o parâmetro `--cli-input-json` para transmitir um arquivo JSON que contém suas senhas antigas e novas. Com esse método, você pode usar senhas fortes com caracteres não alfanuméricos. Pode ser difícil usar senhas com caracteres não alfanuméricos quando elas são transmitidas como parâmetros da linha de comando. Para usar o parâmetro `--cli-input-json`, comece usando o comando `change-password` com o parâmetro `--generate-cli-skeleton`, como no exemplo a seguir.

```
aws iam change-password \
    --generate-cli-skeleton > change-password.json
```

O comando anterior cria um arquivo JSON chamado change-password.json que você pode usar para preencher senhas antigas e novas. Por exemplo, o perfil pode ter a aparência a seguir.

```
{
    "OldPassword": "3s0K_;xh4~8XXI",
    "NewPassword": "]35d/{pB9Fo9wJ"
}
```

Em seguida, para alterar a senha, use o comando `change-password` novamente, desta vez transmitindo o parâmetro `--cli-input-json` para especificar o arquivo JSON. O comando `change-password` a seguir usa o parâmetro `--cli-input-json` com um arquivo JSON chamado change-password.json.

```
aws iam change-password \
    --cli-input-json file://change-password.json
```

Este comando não produz saída.

Esse comando pode ser chamado somente por usuários do IAM. Se esse comando for chamado usando credenciais AWS da conta (raiz), o comando retornará um `InvalidUserType` erro.

Para obter mais informações, consulte Como um usuário do IAM altera a própria senha no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ChangePasswordem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando altera a senha do usuário que está executando o comando. Esse comando pode ser chamado somente por usuários do IAM. Se esse comando for chamado quando você estiver conectado com as credenciais da AWS conta (raiz), o comando retornará um erro. **`InvalidUserType`**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Para obter detalhes da API, consulte ChangePasswordem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando altera a senha do usuário que está executando o comando. Esse comando pode ser chamado somente por usuários do IAM. Se esse comando for chamado quando você estiver conectado com as credenciais da AWS conta (raiz), o comando retornará um erro. **`InvalidUserType`**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Para obter detalhes da API, consulte ChangePasswordem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateAccessKey`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateAccessKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos

- Criar usuários somente leitura e leitura/gravação usando
- Gerenciar chaves de acesso

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }
```

- Para obter detalhes da API, consulte CreateAccessKeya Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
```

```
  echo "  -u user_name   The name of the IAM user."
  echo "  [-f file_name]   Optional file name for the access key output."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:f:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    f) file_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

response=$(aws iam create-access-key \
  --user-name "$user_name" \
  --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-access-key operation failed.$response"
  return 1
fi

if [[ -n "$file_name" ]]; then
  echo "$response" >"$file_name"
fi
```

```
  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}
```

- Para obter detalhes da API, consulte CreateAccessKeyem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
    Aws::IAM::Model::CreateAccessKeyOutcome outcome =
 iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
        std::cout << "Successfully created access key for IAM user " <<
                  userName << std::endl << "  aws_access_key_id = " <<
                  accessKey.GetAccessKeyId() << std::endl <<
                  " aws_secret_access_key = " << accessKey.GetSecretAccessKey()
 <<
                  std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}
```

- Para obter detalhes da API, consulte CreateAccessKeya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como criar uma chave de acesso para um usuário do IAM

O comando `create-access-key`, apresentado a seguir, cria uma chave de acesso (ID da chave de acesso e chave de acesso secreta) para o usuário do IAM denominado Bob.

```
aws iam create-access-key \
    --user-name Bob
```

Saída:

```
{
    "AccessKey": {
        "UserName": "Bob",
        "Status": "Active",
        "CreateDate": "2015-03-09T18:39:23.411Z",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

Armazene a chave de acesso secreta em um local seguro. Se ela for perdida, não será possível recuperá-la e você deverá criar uma nova chave de acesso.

Para obter mais informações, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateAccessKeyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
```

```
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName
 string) (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}
```

- Para obter detalhes da API, consulte CreateAccessKeya Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <user>\s

                Where:
                   user - An AWS IAM user that you can obtain from the AWS
 Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

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

```
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateAccessKeya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a chave de acesso.

```
import { CreateAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 */
export const createAccessKey = (userName) => {
  const command = new CreateAccessKeyCommand({ UserName: userName });
  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateAccessKeya Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccessKey({ UserName: "IAM_USER_NAME" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateAccessKeya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- Para obter detalhes da API, consulte a CreateAccessKeyreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma chave de acesso e um par de chaves de acesso secreto e os atribui ao usuário **`David`**. Certifique-se de salvar os valores **`AccessKeyId`** e **`SecretAccessKey`** em um arquivo, pois este é o único momento em que você pode obter a **`SecretAccessKey`**. Não será possível recuperá-la depois. Caso perca a chave secreta, você deve criar um par de chaves de acesso.

```
New-IAMAccessKey -UserName David
```

Saída:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- Para obter detalhes da API, consulte CreateAccessKeyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma chave de acesso e um par de chaves de acesso secreto e os atribui ao usuário **`David`**. Certifique-se de salvar os valores **`AccessKeyId`** e **`SecretAccessKey`** em um arquivo, pois este é o único momento em que você pode obter a **`SecretAccessKey`**. Não será possível recuperá-la depois. Caso perca a chave secreta, você deve criar um par de chaves de acesso.

```
New-IAMAccessKey -UserName David
```

Saída:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- Para obter detalhes da API, consulte CreateAccessKeyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
```

```
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- Para obter detalhes da API, consulte a CreateAccessKeyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
```

```
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create
more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
```

```
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte CreateAccessKeya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
 Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;
```

```
    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
 loop {
        match client.create_access_key().user_name(user_name).send().await {
            Ok(inner_response) => {
                break Ok(inner_response);
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    Ok(response.unwrap().access_key.unwrap())
}
```

- Para obter detalhes da API, consulte a CreateAccessKeyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->createaccesskey(
          iv_username = iv_user_name ).
        MESSAGE 'Access key created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
```

```
        MESSAGE 'Maximum number of access keys reached.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAccessKeyreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createAccessKey(userName: String) async throws ->
 IAMClientTypes.AccessKey {
        let input = CreateAccessKeyInput(
            userName: userName
        )
        do {
            let output = try await iamClient.createAccessKey(input: input)
            guard let accessKey = output.accessKey else {
                throw ServiceHandlerError.keyError
            }
            return accessKey
        } catch {
            print("ERROR: createAccessKey:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateAccessKeya referência da API AWS SDK for Swift.

Use **`CreateAccountAlias`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateAccountAlias`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar sua conta

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
                  std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte CreateAccountAliasa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como criar um alias da conta

O `create-account-alias` comando a seguir cria o alias `examplecorp` para sua AWS conta.

```
aws iam create-account-alias \
    --account-alias examplecorp
```

Este comando não produz saída.

Para obter mais informações, consulte o ID AWS da sua conta e seu alias no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateAccountAliasem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to create (for example,
 myawsaccount).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request =
 CreateAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.createAccountAlias(request);
```

```
            System.out.println("Successfully created account alias: " + alias);

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

- Para obter detalhes da API, consulte CreateAccountAliasa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar o alias da conta.

```
import { CreateAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias - A unique name for the account alias.
 * @returns
 */
export const createAccountAlias = (alias) => {
  const command = new CreateAccountAliasCommand({
    AccountAlias: alias,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateAccountAliasa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateAccountAliasa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- Para obter detalhes da API, consulte a CreateAccountAliasreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo altera o alias da AWS conta para**`mycompanyaws`**. <accountidnumber>O endereço da página de login do usuário muda para O URL original usando https://mycompanyaws.signin.aws.amazon.com/console. o número de ID da sua conta em vez do alias (https://.signin.aws.amazon). com/console) continua funcionando. No entanto, todos os URLs baseados em alias definidos anteriormente param de funcionar.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obter detalhes da API, consulte CreateAccountAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo altera o alias da AWS conta para**mycompanyaws**. <accountidnumber>O endereço da página de login do usuário muda para O URL original usando https://mycompanyaws.signin.aws.amazon.com/console. o número de ID da sua conta em vez do alias (https://.signin.aws.amazon). com/console) continua funcionando. No entanto, todos os URLs baseados em alias definidos anteriormente param de funcionar.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obter detalhes da API, consulte CreateAccountAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

- Para obter detalhes da API, consulte a CreateAccountAliasReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, criar e excluir aliases da conta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end
```

```
  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte CreateAccountAliasa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->createaccountalias(
```

```
          iv_accountalias = iv_account_alias ).
        MESSAGE 'Account alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Account alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Account alias limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAccountAliasreferência da API AWS SDK for SAP ABAP.

Use **`CreateGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateGroup`.

CLI

AWS CLI

Para criar um grupo do IAM

O comando `create-group`, apresentado a seguir, cria um grupo do IAM denominado `Admins`.

```
aws iam create-group \
    --group-name Admins
```

Saída:

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-03-09T20:30:24.940Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    }
}
```

Para obter mais informações, consulte Criação de grupos de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateGroupem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const createGroup = async (groupName) => {
  const command = new CreateGroupCommand({ GroupName: groupName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte CreateGroupa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um grupo do IAM denominado **`Developers`**.

```
New-IAMGroup -GroupName Developers
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- Para obter detalhes da API, consulte CreateGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um grupo do IAM denominado **`Developers`**.

```
New-IAMGroup -GroupName Developers
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- Para obter detalhes da API, consulte CreateGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateInstanceProfile`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateInstanceProfile`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Conceitos básicos de streaming gerenciado
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create a policy, role, and profile that is associated with instances with
a specified name.
   /// An instance's associated profile defines a role that is assumed by the
   /// instance.The role has attached policies that specify the AWS permissions
granted to
   /// clients that run on the instance.
   /// </summary>
   /// <param name="policyName">Name to use for the policy.</param>
   /// <param name="roleName">Name to use for the role.</param>
   /// <param name="profileName">Name to use for the profile.</param>
   /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
   /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
   /// <returns>The Arn of the profile.</returns>
   public async Task<string> CreateInstanceProfileWithName(
       string policyName,
       string roleName,
       string profileName,
       string ssmOnlyPolicyFile,
       List<string>? awsManagedPolicies = null)
   {

       var assumeRoleDoc = "{" +
                                   "\"Version\": \"2012-10-17\"," +
                                   "\"Statement\": [{" +
                                        "\"Effect\": \"Allow\"," +
                                        "\"Principal\": {" +
                                        "\"Service\": [" +
                                            "\"ec2.amazonaws.com\"" +
                                        "]" +
                                        "}," +
```

```
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
```

```
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
```

```
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }
```

- Para obter detalhes da API, consulte CreateInstanceProfilea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar um perfil de instância

O comando `create-instance-profile`, apresentado a seguir, cria um perfil de instância denominado `Webserver`.

```
aws iam create-instance-profile \
    --instance-profile-name Webserver
```

Saída:

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJMBYC7DLSPEXAMPLE",
        "Roles": [],
        "CreateDate": "2015-03-09T20:33:19.626Z",
        "InstanceProfileName": "Webserver",
        "Path": "/",
```

```
        "Arn": "arn:aws:iam::123456789012:instance-profile/Webserver"
    }
}
```

Para adicionar um perfil a um perfil de instância, use o comando `add-role-to-instance-profile`.

Para obter mais informações, consulte Uso de um perfil do IAM para conceder permissões a aplicações em execução em instâncias do Amazon EC2 no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateInstanceProfileem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
```

- Para obter detalhes da API, consulte CreateInstanceProfilea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um perfil de instância do IAM denominado **`ProfileForDevEC2Instance`**. Você deve executar o comando **`Add-IAMRoleToInstanceProfile`** separadamente para associar o perfil de instância a um perfil do IAM existente que fornece permissões à instância. Por fim, anexe o perfil de instância a uma instância do EC2 ao executá-la. Para isso, use o cmdlet **`New-EC2Instance`** com o parâmetro **`InstanceProfile_Arn`** ou **`InstanceProfile_Name`**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- Para obter detalhes da API, consulte CreateInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um perfil de instância do IAM denominado **`ProfileForDevEC2Instance`**. Você deve executar o comando **`Add-IAMRoleToInstanceProfile`** separadamente para associar o perfil de instância a um perfil do IAM existente que fornece permissões à instância. Por fim, anexe o perfil de instância a uma instância do EC2 ao executá-la. Para isso, use o cmdlet **`New-EC2Instance`** com o parâmetro **`InstanceProfile_Arn`** ou **`InstanceProfile_Name`**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
```

```
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- Para obter detalhes da API, consulte CreateInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo cria uma política, um perfil e um perfil de instância e vincula uns aos outros.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
```

```
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
```

```
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
```

```
            log.info("Created profile %s and added role %s.", profile_name,
 role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
 profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn
```

- Para obter detalhes da API, consulte a CreateInstanceProfileReferência da API AWS SDK for Python (Boto3).

Usar **`CreateLoginProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateLoginProfile`.

CLI

AWS CLI

Criar uma senha para um usuário do IAM

Para criar uma senha de um usuário do IAM, recomendamos usar o parâmetro `--cli-input-json` para transmitir um arquivo JSON que contém a senha. Usando esse método, você pode criar uma senha forte com caracteres não alfanuméricos. Pode ser difícil criar uma senha com caracteres não alfanuméricos ao transmiti-la como parâmetro da linha de comando.

Para usar o parâmetro `--cli-input-json`, comece usando o comando `create-login-profile` com o parâmetro `--generate-cli-skeleton`, como no exemplo a seguir.

```
aws iam create-login-profile \
    --generate-cli-skeleton > create-login-profile.json
```

O comando anterior cria um arquivo JSON chamado create-login-profile.json que pode ser usado para preencher as informações de um comando `create-login-profile` subsequente. Por exemplo:

```
{
    "UserName": "Bob",
    "Password": "&1-3a6u:RA0djs",
    "PasswordResetRequired": true
}
```

Em seguida, para criar uma senha de um usuário do IAM, use o comando `create-login-profile` novamente, desta vez transmitindo o parâmetro `--cli-input-json` a fim de especificar o arquivo JSON. O comando `create-login-profile` a seguir usa o parâmetro `--cli-input-json` com um arquivo JSON chamado create-login-profile.json.

```
aws iam create-login-profile \
    --cli-input-json file://create-login-profile.json
```

Saída:

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2015-03-10T20:55:40.274Z",
        "PasswordResetRequired": true
    }
}
```

Se a nova senha violar a política de senha da conta, o comando retornará um erro `PasswordPolicyViolation`.

Para alterar a senha de um usuário que já tem uma, use `update-login-profile`. Para definir uma política de senha da conta, use o comando `update-account-password-policy`.

Se a política de senha da conta permitir, os usuários do IAM poderão alterar suas próprias senhas usando o comando `change-password`.

Para obter mais informações, consulte Gerenciamento de senhas de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateLoginProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma senha (temporária) para o usuário do IAM chamado Bob e define a sinalização que exige que o usuário altere a senha na próxima vez que **Bob** fizer login.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Saída:

```
CreateDate                    PasswordResetRequired               UserName
----------                    ---------------------               --------
4/14/2015 12:26:30 PM         True                                Bob
```

- Para obter detalhes da API, consulte CreateLoginProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma senha (temporária) para o usuário do IAM chamado Bob e define a sinalização que exige que o usuário altere a senha na próxima vez que **Bob** fizer login.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Saída:

```
CreateDate                    PasswordResetRequired               UserName
----------                    ---------------------               --------
4/14/2015 12:26:30 PM         True                                Bob
```

- Para obter detalhes da API, consulte CreateLoginProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateOpenIdConnectProvider`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateOpenIdConnectProvider`.

CLI

AWS CLI

Criar um provedor OpenID Connect (OIDC)

Para criar um provedor OpenID Connect (OIDC), recomendamos usar o parâmetro `--cli-input-json` para transmitir um arquivo JSON que contém os parâmetros necessários. Ao criar um provedor OIDC, você deve transmitir o URL do provedor, e o URL deve começar com `https://`. Pode ser difícil transmitir o URL como parâmetro de linha de comando, porque os caracteres de dois pontos (:) e barra (/) têm um significado especial em alguns ambientes de linha de comando. Usar o parâmetro `--cli-input-json` contorna essa limitação.

Para usar o parâmetro `--cli-input-json`, comece usando o comando `create-open-id-connect-provider` com o parâmetro `--generate-cli-skeleton`, como no exemplo a seguir.

```
aws iam create-open-id-connect-provider \
    --generate-cli-skeleton > create-open-id-connect-provider.json
```

O comando anterior cria um arquivo JSON chamado create-open-id-connect-provider.json que você pode usar para preencher as informações de um comando `create-open-id-connect-provider` subsequente. Por exemplo:

```
{
    "Url": "https://server.example.com",
    "ClientIDList": [
        "example-application-ID"
    ],
    "ThumbprintList": [
        "c3768084dfb3d2b68b7897bf5f565da8eEXAMPLE"
    ]
}
```

Se a descoberta do seu provedor OIDC e os endpoints do JWKS usarem certificados diferentes, inclua impressões digitais de ambos os endpoints no `ThumbprintList`.

Em seguida, para criar o provedor OpenID Connect (OIDC), use o comando `create-open-id-connect-provider` novamente, desta vez transmitindo o parâmetro `--cli-input-json` a fim de especificar o arquivo JSON. O comando `create-open-id-connect-provider` a seguir usa o parâmetro `--cli-input-json` com um arquivo JSON chamado create-open-id-connect-provider.json.

```
aws iam create-open-id-connect-provider \
    --cli-input-json file://create-open-id-connect-provider.json
```

Saída:

```
{
    "OpenIDConnectProviderArn": "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
}
```

Para obter mais informações sobre provedores OIDC, consulte Criar provedores de identidade OpenID Connect (OIDC) no Guia do usuário do AWS IAM.

Para obter mais informações sobre como conseguir impressões digitais de um provedor OIDC, consulte Obter a impressão digital para um provedor de identidade OpenID Connect no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateOpenIdConnectProviderem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um provedor OIDC do IAM associado ao serviço do provedor compatível com OIDC encontrado no URL **`https://example.oidcprovider.com`** e no ID do cliente **`my-testapp-1`**. O provedor OIDC fornece a impressão digital. Para autenticar a impressão digital, siga as etapas em http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList
 my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Saída:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para obter detalhes da API, consulte CreateOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um provedor OIDC do IAM associado ao serviço do provedor compatível com OIDC encontrado no URL **`https://example.oidcprovider.com`** e no ID do cliente **`my-testapp-1`**. O provedor OIDC fornece a impressão digital. Para autenticar a impressão digital, siga as etapas em http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList
 my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Saída:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para obter detalhes da API, consulte CreateOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreatePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreatePolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar usuários somente leitura e leitura/gravação usando
- Como criar um espaço de trabalho de monitoramento gerenciado
- Conceitos básicos de streaming gerenciado
- Conceitos básicos da orquestração de fluxo de trabalaho

- Políticas gerenciadas
- Configurar o controle de Attribute-Based acesso
- Como configurar o gerenciamento de sistemas
- Trabalhar com a API IAM Policy Builder

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
```

```
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) policy_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
  errecho "ERROR: You must provide a policy name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-policy \
  --policy-name "$policy_name" \
  --policy-document "$policy_document" \
  --output text \
  --query Policy.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-policy operation failed.\n$response"
```

```
    return 1
  fi

  echo "$response"
}
```

- Para obter detalhes da API, consulte CreatePolicyem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                      const Aws::String &rsrcArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
                  std::endl;
    }

    return result;
```

```
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
                 << "  \"Version\": \"2012-10-17\","
                 << "  \"Statement\": ["
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": \"logs:CreateLogGroup\","
                 << "        \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    },"
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": ["
                 << "            \"dynamodb:DeleteItem\","
                 << "            \"dynamodb:GetItem\","
                 << "            \"dynamodb:PutItem\","
                 << "            \"dynamodb:Scan\","
                 << "            \"dynamodb:UpdateItem\""
                 << "       ],"
                 << "       \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    }"
                 << "   ]"
                 << "}";

    return stringStream.str();
}
```

- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como criar uma política gerenciada pelo cliente

O comando apresentado a seguir cria uma política gerenciada pelo cliente denominada `my-policy`. O arquivo `policy.json` é um documento JSON na pasta atual que concede acesso somente leitura à pasta `shared` em um bucket do Amazon S3 denominado `amzn-s3-demo-bucket`.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json
```

Conteúdo de policy.json:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/shared/*"
            ]
        }
    ]
}
```

Saída:

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "CreateDate": "2015-06-01T19:31:18.620Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::0123456789012:policy/my-policy",
        "UpdateDate": "2015-06-01T19:31:18.620Z"
    }
```

```
}
```

Para obter mais informações sobre o uso de arquivos como entrada para parâmetros de string, consulte Especificar valores de parâmetros para a AWS CLI no Guia do usuário da AWS CLI.

Exemplo 2: como criar uma política gerenciada pelo cliente com uma descrição

O comando a seguir cria uma política gerenciada pelo cliente denominada `my-policy` com uma descrição imutável.

O arquivo `policy.json` é um documento JSON na pasta atual que concede acesso a todas as ações Put, List e Get para um bucket do Amazon S3 denominado `amzn-s3-demo-bucket`.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --description "This policy grants access to all Put, Get, and List actions
 for amzn-s3-demo-bucket"
```

Conteúdo de policy.json:

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

Saída:

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T22:38:47+00:00",
        "UpdateDate": "2023-05-24T22:38:47+00:00"
    }
}
```

Para obter mais informações sobre Idenity-based políticas, consulte Identity-based políticas e políticas baseadas em recursos no Guia do usuário do AWS IAM.

Exemplo 3: para criar uma política gerenciada pelo cliente com tags

O comando apresentado a seguir cria uma política gerenciada pelo cliente, denominada `my-policy`, com etiquetas. Este exemplo usa o `--tags` parâmetro com as seguintes JSON-formatted tags:`'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'`. Como alternativa, o parâmetro `--tags` pode ser usado com tags no formato abreviado: `'Key=Department,Value=Accounting Key=Location,Value=Seattle'`.

O arquivo `policy.json` é um documento JSON na pasta atual que concede acesso a todas as ações Put, List e Get para um bucket do Amazon S3 denominado `amzn-s3-demo-bucket`.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

Conteúdo de policy.json:

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

Saída:

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::12345678012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T23:16:39+00:00",
        "UpdateDate": "2023-05-24T23:16:39+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
                "Key": "Location",
                "Value": "Seattle"
            {
        ]
    }
}
```

Para obter mais informações sobre as políticas de marcação, consulte Marcar políticas gerenciadas pelo cliente no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreatePolicyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
```

```
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

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

    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
```

```
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static void main(String[] args) {

        final String usage = """
                Usage:
                    CreatePolicy <policyName>\s

                Where:
                    policyName - A unique policy name.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: "
+ result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument)
                    .build();

            CreatePolicyResponse response = iam.createPolicy(request);
```

```
            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);

 waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

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

- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a política .

```
import { CreatePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
```

```
 * @param {string} policyName
 */
export const createPolicy = (policyName) => {
  const command = new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "*",
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });
```

```
var myManagedPolicy = {
  Version: "2012-10-17",
  Statement: [
    {
      Effect: "Allow",
      Action: "logs:CreateLogGroup",
      Resource: "RESOURCE_ARN",
    },
    {
      Effect: "Allow",
      Action: [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem",
      ],
      Resource: "RESOURCE_ARN",
    },
  ],
};

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "myDynamoDBPolicy",
};

iam.createPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "dynamodb:DeleteItem",
                    "dynamodb:GetItem",
                    "dynamodb:PutItem",
                    "dynamodb:Scan",
                    "dynamodb:UpdateItem"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- Para obter detalhes da API, consulte a CreatePolicyreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    /**
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    public function createPolicy(string $policyName, string $policyDocument)
    {
        $result = $this->customWaiter(function () use ($policyName,
 $policyDocument) {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria uma nova política do IAM na AWS conta atual chamada **MySamplePolicy** O arquivo **MySamplePolicy.json** fornece o conteúdo da política. Observe que você deve usar o parâmetro switch **-Raw** para processar com êxito o arquivo de política JSON.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Saída:

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- Para obter detalhes da API, consulte CreatePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma nova política do IAM na AWS conta atual chamada **MySamplePolicy** O arquivo **MySamplePolicy.json** fornece o conteúdo da política. Observe que você deve usar o parâmetro switch **-Raw** para processar com êxito o arquivo de política JSON.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Saída:

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- Para obter detalhes da API, consulte CreatePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
```

```
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- Para obter detalhes da API, consulte a CreatePolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
```

```
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
```

```
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- Para obter detalhes da API, consulte a CreatePolicyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->createpolicy(
          iv_policyname = iv_policy_name
          iv_policydocument = iv_policy_document
          iv_description = iv_description ).
        MESSAGE 'Policy created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Policy already exists.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreatePolicyreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createPolicy(name: String, policyDocument: String) async throws -
> IAMClientTypes.Policy {
        let input = CreatePolicyInput(
            policyDocument: policyDocument,
            policyName: name
        )
        do {
            let output = try await iamClient.createPolicy(input: input)
            guard let policy = output.policy else {
```

```
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            print("ERROR: createPolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreatePolicya referência da API AWS SDK for Swift.

Use **`CreatePolicyVersion`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreatePolicyVersion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Políticas gerenciadas

CLI

AWS CLI

Como criar uma nova versão de uma política gerenciada

Este exemplo cria uma nova versão `v2` da política do IAM cujo ARN é `arn:aws:iam::123456789012:policy/MyPolicy` e a torna a versão padrão.

```
aws iam create-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --policy-document file://NewPolicyVersion.json \
    --set-as-default
```

Saída:

```
{
    "PolicyVersion": {
        "CreateDate": "2015-06-16T18:56:03.721Z",
```

```
        "VersionId": "v2",
        "IsDefaultVersion": true
    }
}
```

Para obter mais informações, consulte Versionamento de políticas do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreatePolicyVersionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma versão "v2" da política do IAM cujo ARN é **`arn:aws:iam::123456789012:policy/MyPolicy`** e a torna a versão padrão. O arquivo **`NewPolicyVersion.json`** fornece o conteúdo da política. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o arquivo de política JSON.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Saída:

```
CreateDate                          Document                 IsDefaultVersion
           VersionId
----------                          --------                 ----------------
           ---------
4/15/2015 10:54:54 AM                                        True
           v2
```

- Para obter detalhes da API, consulte CreatePolicyVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma versão "v2" da política do IAM cujo ARN é **`arn:aws:iam::123456789012:policy/MyPolicy`** e a torna a versão padrão. O arquivo **`NewPolicyVersion.json`** fornece o conteúdo da política. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o arquivo de política JSON.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Saída:

```
CreateDate                          Document                 IsDefaultVersion
           VersionId
----------                          --------                 ----------------
           ---------
4/15/2015 10:54:54 AM                                        True
           v2
```

- Para obter detalhes da API, consulte CreatePolicyVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
```

```
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version
```

- Para obter detalhes da API, consulte a CreatePolicyVersionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->createpolicyversion(
          iv_policyarn = iv_policy_arn
          iv_policydocument = iv_policy_document
          iv_setasdefault = iv_set_as_default ).
        MESSAGE 'Policy version created successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy version limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreatePolicyVersionreferência da API AWS SDK for SAP ABAP.

Use **`CreateRole`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateRole`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Criar uma API REST com integração de proxy de função
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Como criar um espaço de trabalho de monitoramento gerenciado
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Conceitos básicos de data warehouses de tecnologia sem servidor
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de arquivos de atributos de machine learning
- Conceitos básicos de clusters gerenciados de kubernetes
- Conceitos básicos de streaming gerenciado
- Conceitos básicos de clusters de data warehouse provisionados
- Conceitos básicos da orquestração de fluxo de trabalaho

- Gerenciar funções
- Como mover segredos codificados para o armazenamento seguro de segredos
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Como configurar o gerenciamento de sistemas
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor
- Trabalhe com Streams e Time-to-Live

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create a new IAM role.
   /// </summary>
   /// <param name="roleName">The name of the IAM role.</param>
   /// <param name="rolePolicyDocument">The name of the IAM policy document
   /// for the new role.</param>
   /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
   public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
   {
       var request = new CreateRoleRequest
       {
           RoleName = roleName,
           AssumeRolePolicyDocument = rolePolicyDocument,
       };

       var response = await _IAMService.CreateRoleAsync(request);
       return response.Role.Arn;
   }
```

- Para obter detalhes da API, consulte CreateRolea Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
```

```
  echo "function iam_create_user_access_key"
  echo "Creates an AWS Identity and Access Management (IAM) role."
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_json -- The assume role policy document."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte CreateRoleem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::createIamRole(
        const Aws::String &roleName,
        const Aws::String &policy,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
```

```
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateRolea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como criar um perfil do IAM

O comando `create-role`, apresentado a seguir, cria um perfil, denominado `Test-Role`, e anexa uma política de confiança a ele.

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

Saída:

```
{
    "Role": {
        "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "CreateDate": "2013-06-07T20:43:32.821Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

A política de confiança é definida como um documento JSON no Test-Role-Trust-Policy.jsonarquivo. (O nome e a extensão do arquivo não têm significado.) A política de confiança deve especificar uma entidade principal.

Para anexar uma política de permissões a um perfil, use o comando `put-role-policy`.

Para obter mais informações, consulte Criação de perfis do IAM no Guia do usuário do AWS IAM.

Exemplo 2: como criar um perfil do IAM com a duração máxima da sessão especificada

O comando `create-role`, apresentado a seguir, cria um perfil denominado `Test-Role` e define a duração máxima da sessão como 7.200 segundos (duas horas).

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --max-session-duration 7200
```

Saída:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:role/Test-Role",
        "CreateDate": "2023-05-24T23:50:25+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::12345678012:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

Para obter mais informações, consulte Modificar a duração máxima da sessão (AWS API) de uma função no Guia do usuário do AWS IAM.

Exemplo 3: como criar um perfil do IAM com etiquetas

O comando apresentado a seguir cria um perfil do IAM `Test-Role` com etiquetas. Este exemplo usa o sinalizador de `--tags` parâmetro com as seguintes JSON-formatted tags:`'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'`. Como alternativa, o sinalizador `--tags` pode ser usado com etiquetas no formato abreviado: `'Key=Department,Value=Accounting Key=Location,Value=Seattle'`.

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

Saída:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role",
        "CreateDate": "2023-05-25T23:29:41+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
```

```
                "Value": "Seattle"
            }
        ]
    }
}
```

Para obter mais informações, consulte Marcar perfis do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateRoleem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
```

```
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- Para obter detalhes da API, consulte CreateRolea Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
*   This example requires a trust policy document. For more information, see:
*   https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-
roles/
*
*
*  In addition, set up your development environment, including your credentials.
*
*  For information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = """
                Usage:
                    <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
```

```
                    fileLocation - The location of the JSON document that
represents the trust policy.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject)
readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(jsonObject.toJSONString())
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());

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

    public static Object readJsonSimpleDemo(String filename) throws Exception {
```

```
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
}
```

- Para obter detalhes da API, consulte CreateRolea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a função.

```
import { CreateRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const createRole = (roleName) => {
  const command = new CreateRoleCommand({
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            Service: "lambda.amazonaws.com",
          },
          Action: "sts:AssumeRole",
        },
```

```
      ],
    }),
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte CreateRolea Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
```

```
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- Para obter detalhes da API, consulte CreateRolea Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um perfil denominado **`MyNewRole`** e anexa a ele a política encontrada no arquivo **`NewRoleTrustPolicy.json`**. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o arquivo de política JSON. O documento de política exibido na saída é codificado em URL. Ele é decodificado nesse exemplo com o método .NET **`UrlDecode`**.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Saída:

```
Arn                       : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
```

```
                            %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                            %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                            %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                            %20%5D%0D%0A%7D
CreateDate                : 4/15/2015 11:04:23 AM
Path                      : /
RoleId                    : V5PAJI2KPN4EAEXAMPLE1
RoleName                  : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Para obter detalhes da API, consulte CreateRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um perfil denominado **MyNewRole** e anexa a ele a política encontrada no arquivo **NewRoleTrustPolicy.json**. Observe que você deve usar o parâmetro switch **-Raw** para processar com êxito o arquivo de política JSON. O documento de política exibido na saída é codificado em URL. Ele é decodificado nesse exemplo com o método .NET **UrlDecode**.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Saída:

```
Arn                       : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                           %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                           %20%5D%0D%0A%7D
CreateDate                : 4/15/2015 11:04:23 AM
Path                      : /
RoleId                    : V5PAJI2KPN4EAEXAMPLE1
RoleName                  : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Para obter detalhes da API, consulte CreateRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

- Para obter detalhes da API, consulte a CreateRoleReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Creates a role and attaches policies to it.
 #
 # @param role_name [String] The name of the role.
 # @param assume_role_policy_document [Hash] The trust relationship policy
document.
 # @param policy_arns [Array<String>] The ARNs of the policies to attach.
 # @return [String, nil] The ARN of the new role if successful, or nil if an
error occurred.
 def create_role(role_name, assume_role_policy_document, policy_arns)
   response = @iam_client.create_role(
     role_name: role_name,
     assume_role_policy_document: assume_role_policy_document.to_json
   )
   role_arn = response.role.arn

   policy_arns.each do |policy_arn|
     @iam_client.attach_role_policy(
       role_name: role_name,
       policy_arn: policy_arn
     )
   end

   role_arn
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error creating role: #{e.message}")
   nil
```

```
  end
```

- Para obter detalhes da API, consulte CreateRolea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    };

    Ok(response.role.unwrap())
}
```

- Para obter detalhes da API, consulte a CreateRolereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->createrole(
          iv_rolename = iv_role_name
          iv_assumerolepolicydocument = iv_assume_role_policy_document ).
        MESSAGE 'Role created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Role already exists.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Assume role policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Role limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateRolereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3
```

```
    public func createRole(name: String, policyDocument: String) async throws ->
 String {
        let input = CreateRoleInput(
            assumeRolePolicyDocument: policyDocument,
            roleName: name
        )
        do {
            let output = try await client.createRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            guard let id = role.roleId else {
                throw ServiceHandlerError.noSuchRole
            }
            return id
        } catch {
            print("ERROR: createRole:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateRolea referência da API AWS SDK for Swift.

Use **`createSAMLProvider com`** um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateSAMLProvider`.

CLI

AWS CLI

Como criar um provedor SAML

Este exemplo cria um novo provedor SAML no IAM denominado `MySAMLProvider`. Ele é descrito pelo documento de metadados do SAML, que encontra-se no arquivo `SAMLMetaData.xml`.

```
aws iam create-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
```

```
    --name MySAMLProvider
```

Saída:

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/MySAMLProvider"
}
```

Para obter mais informações, consulte Criação de provedores de identidade SAML do IAM no Guia do usuário do AWS IAM.

- Consulte detalhes da API em CreateSAMLProvider na Referência de comandos da AWS CLI .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
import * as path from "node:path";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";

const client = new IAMClient({});

/**
 * This sample document was generated using Auth0.
 * For more information on generating this document,
 see https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_providers_create_saml.html#samlstep1.
 */
const sampleMetadataDocument = readFileSync(
  path.join(
    dirnameFromMetaUrl(import.meta.url),
```

```
    "../../../../resources/sample_files/sample_saml_metadata.xml",
  ),
);

/**
 *
 * @param {*} providerName
 * @returns
 */
export const createSAMLProvider = async (providerName) => {
  const command = new CreateSAMLProviderCommand({
    Name: providerName,
    SAMLMetadataDocument: sampleMetadataDocument.toString(),
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Consulte detalhes da API em CreateSAMLProvider na Referência da API do AWS SDK para JavaScript .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma entidade provedora SAML no IAM. Ele é denominado **`MySAMLProvider`** e descrito pelo documento de metadados SAML encontrado no arquivo **`SAMLMetaData.xml`**, que foi baixado separadamente do site do provedor de serviços SAML.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Saída:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Para ver detalhes da API, consulte CreateSAMLProvider na Referência de cmdlets do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma entidade provedora SAML no IAM. Ele é denominado **`MySAMLProvider`** e descrito pelo documento de metadados SAML encontrado no arquivo **`SAMLMetaData.xml`**, que foi baixado separadamente do site do provedor de serviços SAML.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Saída:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Para obter detalhes sobre a API, consulte CreateSAMLProvider na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Use **`CreateServiceLinkedRole`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateServiceLinkedRole`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
```

```
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }
```

- Para obter detalhes da API, consulte CreateServiceLinkedRolea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar um perfil vinculado ao serviço

O `create-service-linked-role` exemplo a seguir cria uma função vinculada ao serviço para o AWS serviço especificado e anexa a descrição especificada.

```
aws iam create-service-linked-role \
    --aws-service-name lex.amazonaws.com \
    --description "My service-linked role to support Lex"
```

Saída:

```
{
    "Role": {
        "Path": "/aws-service-role/lex.amazonaws.com/",
        "RoleName": "AWSServiceRoleForLexBots",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::1234567890:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
        "CreateDate": "2019-04-17T20:34:14+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
```

```
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "lex.amazonaws.com"
                        ]
                    }
                }
            ]
        }
    }
}
```

Para obter mais informações, consulte Usar perfis vinculados ao serviço no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateServiceLinkedRoleem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context,
 serviceName string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- Para obter detalhes da API, consulte CreateServiceLinkedRolea Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma função vinculada ao serviço.

```
import {
  CreateServiceLinkedRoleCommand,
  GetRoleCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} serviceName
 */
export const createServiceLinkedRole = async (serviceName) => {
  const command = new CreateServiceLinkedRoleCommand({
    // For a list of AWS services that support service-linked roles,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-
services-that-work-with-iam.html.
    //
    // For a list of AWS service endpoints, see https://docs.aws.amazon.com/
general/latest/gr/aws-service-information.html.
    AWSServiceName: serviceName,
  });
  try {
    const response = await client.send(command);
    console.log(response);
    return response;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInputException" &&
      caught.message.includes(
```

```
        "Service role name AWSServiceRoleForElasticBeanstalk has been taken in
 this account",
      )
    ) {
      console.warn(caught.message);
      return client.send(
        new GetRoleCommand({ RoleName: "AWSServiceRoleForElasticBeanstalk" }),
      );
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte CreateServiceLinkedRolea Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' =>
 $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
```

```
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- Para obter detalhes da API, consulte CreateServiceLinkedRolea Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um perfil vinculado ao serviço para o serviço de ajuste de escala automático.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support
 autoscaling"
```

- Para obter detalhes da API, consulte CreateServiceLinkedRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um perfil vinculado ao serviço para o serviço de ajuste de escala automático.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support
 autoscaling"
```

- Para obter detalhes da API, consulte CreateServiceLinkedRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description
        )
        role = iam.Role(response["Role"]["RoleName"])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
 service_name)
        raise
    else:
        return role
```

- Para obter detalhes da API, consulte a CreateServiceLinkedRoleReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Creates a service-linked role
 #
 # @param service_name [String] The service name to create the role for.
 # @param description [String] The description of the service-linked role.
 # @param suffix [String] Suffix for customizing role name.
 # @return [String] The name of the created role
 def create_service_linked_role(service_name, description, suffix)
   response = @iam_client.create_service_linked_role(
     aws_service_name: service_name, description: description, custom_suffix:
suffix
   )
   role_name = response.role.role_name
   @logger.info("Created service-linked role #{role_name}.")
   role_name
 rescue Aws::Errors::ServiceError => e
   @logger.error("Couldn't create service-linked role for #{service_name}.
Here's why:")
   @logger.error("\t#{e.code}: #{e.message}")
   raise
 end
```

- Para obter detalhes da API, consulte CreateServiceLinkedRolea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput,
 SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a CreateServiceLinkedRolereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listpolicyversions(
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateServiceLinkedRolereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createServiceLinkedRole(service: String, suffix: String? = nil,
 description: String?)
                    async throws -> IAMClientTypes.Role {
        let input = CreateServiceLinkedRoleInput(
            awsServiceName: service,
            customSuffix: suffix,
            description: description
        )
        do {
            let output = try await client.createServiceLinkedRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
```

```
            return role
        } catch {
            print("ERROR: createServiceLinkedRole:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateServiceLinkedRolea referência da API AWS SDK for Swift.

Use **`CreateUser`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateUser`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar usuários somente leitura e leitura/gravação usando

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
```

```
        return response.User;
    }
```

- Para obter detalhes da API, consulte CreateUsera Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
```

```
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
```

```
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte CreateUserem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateUserRequest create_request;
    create_request.SetUserName(userName);

    auto create_outcome = iam.CreateUser(create_request);
    if (!create_outcome.IsSuccess()) {
        std::cerr << "Error creating IAM user " << userName << ":" <<
                  create_outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created IAM user " << userName << std::endl;
    }

    return create_outcome.IsSuccess();
```

- Para obter detalhes da API, consulte CreateUsera Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como criar um usuário do IAM

O comando `create-user`, apresentado a seguir, cria um usuário do IAM denominado Bob na conta atual.

```
aws iam create-user \
    --user-name Bob
```

Saída:

```
{
    "User": {
        "UserName": "Bob",
        "Path": "/",
        "CreateDate": "2023-06-08T03:20:41.270Z",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Bob"
    }
```

```
}
```

Para obter mais informações, consulte Como criar um usuário do IAM em sua AWS conta no Guia AWS do usuário do IAM.

Exemplo 2: como criar um usuário do IAM em um caminho especificado

O comando `create-user`, apresentado a seguir, cria um usuário do IAM denominado Bob no caminho especificado.

```
aws iam create-user \
    --user-name Bob \
    --path /division_abc/subdivision_xyz/
```

Saída:

```
{
    "User": {
        "Path": "/division_abc/subdivision_xyz/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/division_abc/subdivision_xyz/Bob",
        "CreateDate": "2023-05-24T18:20:17+00:00"
    }
}
```

Para obter mais informações, consulte Identificadores do IAM no Guia do usuário do AWS IAM.

Exemplo 3: como criar um usuário do IAM com etiquetas

O comando `create-user`, apresentado a seguir, cria um usuário do IAM denominado Bob com etiquetas. Este exemplo usa o sinalizador de `--tags` parâmetro com as seguintes JSON-formatted tags:`'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'`. Como alternativa, o sinalizador `--tags` pode ser usado com etiquetas no formato abreviado: `'Key=Department,Value=Accounting Key=Location,Value=Seattle'`.

```
aws iam create-user \
    --user-name Bob \
```

```
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

Saída:

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-25T17:14:21+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
            }
        ]
    }
}
```

Para obter mais informações, consulte Marcar usuários do IAM no Guia do usuário do AWS IAM.

Exemplo 3: como criar um usuário do IAM com um limite de permissões definido

O `create-user` comando a seguir cria um usuário do IAM chamado Bob com o limite de permissões de AmazonS3FullAccess.

```
aws iam create-user \
    --user-name Bob \
    --permissions-boundary arn:aws:iam::aws:policy/AmazonS3FullAccess
```

Saída:

```
{
    "User": {
        "Path": "/",
```

```
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-24T17:50:53+00:00",
        "PermissionsBoundary": {
        "PermissionsBoundaryType": "Policy",
        "PermissionsBoundaryArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
        }
    }
}
```

Para obter mais informações, consulte Limites de permissões para entidades do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateUserem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
```

```
 IamClient *iam.Client
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}
```

- Para obter detalhes da API, consulte CreateUsera Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
```

```
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username>\s

                Where:
                    username - The name of the user to create.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMUser(iam, username);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
```

```
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created.
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
 iamWaiter.waitUntilUserExists(userRequest);

 waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

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

- Para obter detalhes da API, consulte CreateUsera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o usuário.

```
import { CreateUserCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const createUser = (name) => {
  const command = new CreateUserCommand({ UserName: name });
  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateUsera Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    iam.createUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
```

```
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log(
      "User " + process.argv[2] + " already exists",
      data.User.UserId
    );
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateUsera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- Para obter detalhes da API, consulte a CreateUserreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);

        return $result['User'];
    }
```

- Para obter detalhes da API, consulte CreateUsera Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um usuário do IAM chamado **Bob**. Se Bob precisar entrar no AWS console, você deverá executar o comando separadamente **New-IAMLoginProfile** para criar um perfil de login com uma senha. Se Bob precisar executar AWS PowerShell comandos CLI multiplataforma ou AWS fazer chamadas de API, você deverá executar **New-IAMAccessKey** o comando separadamente para criar chaves de acesso.

```
New-IAMUser -UserName Bob
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- Para obter detalhes da API, consulte CreateUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um usuário do IAM chamado **Bob**. Se Bob precisar entrar no AWS console, você deverá executar o comando separadamente **New-IAMLoginProfile** para criar um perfil de login com uma senha. Se Bob precisar executar AWS PowerShell comandos CLI multiplataforma ou AWS fazer chamadas de API, você deverá executar **New-IAMAccessKey** o comando separadamente para criar chaves de acesso.

```
New-IAMUser -UserName Bob
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
```

```
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- Para obter detalhes da API, consulte CreateUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- Para obter detalhes da API, consulte a CreateUserReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Creates a user and their login profile
 #
 # @param user_name [String] The name of the user
 # @param initial_password [String] The initial password for the user
 # @return [String, nil] The ID of the user if created, or nil if an error
occurred
 def create_user(user_name, initial_password)
   response = @iam_client.create_user(user_name: user_name)
   @iam_client.wait_until(:user_exists, user_name: user_name)
   @iam_client.create_login_profile(
     user_name: user_name,
     password: initial_password,
     password_reset_required: true
   )
   @logger.info("User '#{user_name}' created successfully.")
   response.user.user_id
 rescue Aws::IAM::Errors::EntityAlreadyExists
   @logger.error("Error creating user '#{user_name}': user already exists.")
   nil
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error creating user '#{user_name}': #{e.message}")
   nil
 end
```

- Para obter detalhes da API, consulte CreateUsera Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
 iamError> {
    let response = client.create_user().user_name(user_name).send().await?;

    Ok(response.user.unwrap())
}
```

- Para obter detalhes da API, consulte a CreateUserreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->createuser(
          iv_username = iv_user_name ).
        MESSAGE 'User created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'User already exists.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Limit exceeded for IAM users.' TYPE 'E'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Entity does not exist.' TYPE 'E'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateUserreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createUser(name: String) async throws -> String {
        let input = CreateUserInput(
            userName: name
        )
        do {
            let output = try await client.createUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            guard let id = user.userId else {
                throw ServiceHandlerError.noSuchUser
            }
            return id
        } catch {
            print("ERROR: createUser:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateUsera referência da API AWS SDK for Swift.

Usar **`CreateVirtualMfaDevice`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateVirtualMfaDevice`.

CLI

AWS CLI

Criar um dispositivo de MFA virtual

Este exemplo cria um dispositivo de MFA virtual denominado `BobsMFADevice`. Ele cria um arquivo contendo informações de bootstrap denominadas `QRCode.png` e as coloca no diretório `C:/`. O método de bootstrap usado neste exemplo é `QRCodePNG`.

```
aws iam create-virtual-mfa-device \
    --virtual-mfa-device-name BobsMFADevice \
    --outfile C:/QRCode.png \
    --bootstrap-method QRCodePNG
```

Saída:

```
{
    "VirtualMFADevice": {
        "SerialNumber": "arn:aws:iam::210987654321:mfa/BobsMFADevice"
}
```

Para obter mais informações, consulte Uso de autenticação multifator (MFA) na AWS no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte CreateVirtualMfaDeviceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um dispositivo de MFA virtual. As linhas 2 e 3 extraem o valor de **`Base32StringSeed`** de que o programa de software de MFA virtual precisa para criar uma conta (como alternativa ao código QR). Depois de configurar o programa com o valor, obtenha dois códigos de autenticação sequencial do programa. Por fim, use o último comando

para vincular o dispositivo de MFA virtual ao usuário do IAM **Bob** e sincronizar a conta com os dois códigos de autenticação.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Saída:

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
 register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

Exemplo 2: este exemplo cria um dispositivo de MFA virtual. As linhas 2 e 3 extraem o valor de **QRCodePNG** e o gravam em um arquivo. Essa imagem pode ser digitalizada pelo programa de software de MFA virtual para criar uma conta (como alternativa à inserção manual Base32StringSeed do valor). Depois de criar a conta no programa de MFA virtual, obtenha dois códigos de autenticação sequencial e insira-os nos últimos comandos para vincular o dispositivo MFA virtual ao usuário do IAM **Bob** e sincronizar a conta.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
 QRCode.png
```

Saída:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Para obter detalhes da API, consulte CreateVirtualMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um dispositivo de MFA virtual. As linhas 2 e 3 extraem o valor de **`Base32StringSeed`** de que o programa de software de MFA virtual precisa para criar uma conta (como alternativa ao código QR). Depois de configurar o programa com o valor, obtenha dois códigos de autenticação sequencial do programa. Por fim, use o último comando para vincular o dispositivo de MFA virtual ao usuário do IAM **`Bob`** e sincronizar a conta com os dois códigos de autenticação.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Saída:

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
 register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

Exemplo 2: este exemplo cria um dispositivo de MFA virtual. As linhas 2 e 3 extraem o valor de **`QRCodePNG`** e o gravam em um arquivo. Essa imagem pode ser digitalizada pelo programa de software de MFA virtual para criar uma conta (como alternativa à inserção manual Base32StringSeed do valor). Depois de criar a conta no programa de MFA virtual, obtenha dois códigos de autenticação sequencial e insira-os nos últimos comandos para vincular o dispositivo MFA virtual ao usuário do IAM **`Bob`** e sincronizar a conta.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
 QRCode.png
```

Saída:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --
```

```
Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Para obter detalhes da API, consulte CreateVirtualMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeactivateMfaDevice`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeactivateMfaDevice`.

CLI

AWS CLI

Desativar um dispositivo de MFA

Esse comando desativa o dispositivo de MFA virtual com o ARN `arn:aws:iam::210987654321:mfa/BobsMFADevice` associado ao usuário `Bob`.

```
aws iam deactivate-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice
```

Este comando não produz saída.

Para obter mais informações, consulte Uso de autenticação multifator (MFA) na AWS no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeactivateMfaDeviceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando desabilita o dispositivo de MFA de hardware associado ao usuário **`Bob`** que tem o número de série **`123456789012`**.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Exemplo 2: este comando desativa o dispositivo de MFA virtual associado ao usuário **`David`** que tem o ARN **`arn:aws:iam::210987654321:mfa/David`**. Observe que o dispositivo de MFA virtual não é excluído da conta. O dispositivo virtual ainda está presente e aparece na saída do comando **`Get-IAMVirtualMFADevice`**. Antes de criar um dispositivo de MFA virtual para o mesmo usuário, você deve excluir o antigo usando o comando **`Remove-IAMVirtualMFADevice`**.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber
 "arn:aws:iam::210987654321:mfa/David"
```

- Para obter detalhes da API, consulte DeactivateMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando desabilita o dispositivo de MFA de hardware associado ao usuário **`Bob`** que tem o número de série **`123456789012`**.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Exemplo 2: este comando desativa o dispositivo de MFA virtual associado ao usuário **`David`** que tem o ARN **`arn:aws:iam::210987654321:mfa/David`**. Observe que o dispositivo de MFA virtual não é excluído da conta. O dispositivo virtual ainda está presente e aparece na saída do comando **`Get-IAMVirtualMFADevice`**. Antes de criar um dispositivo de MFA virtual para o mesmo usuário, você deve excluir o antigo usando o comando **`Remove-IAMVirtualMFADevice`**.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber
 "arn:aws:iam::210987654321:mfa/David"
```

- Para obter detalhes da API, consulte DeactivateMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteAccessKey`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteAccessKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar usuários somente leitura e leitura/gravação usando
- Gerenciar chaves de acesso

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Delete an IAM user's access key.
   /// </summary>
   /// <param name="accessKeyId">The Id for the IAM access key.</param>
   /// <param name="userName">The username of the user that owns the IAM
   /// access key.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
   {
       var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
       {
           AccessKeyId = accessKeyId,
           UserName = userName,
       });

       return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
   }
```

- Para obter detalhes da API, consulte DeleteAccessKeya Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
```

```
  echo "  -k access_key   The access key to delete."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:k:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    k) access_key="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}
```

- Para obter detalhes da API, consulte DeleteAccessKeyem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
```

```
                    << userName << ": " << outcome.GetError().GetMessage() <<
                    std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                    << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteAccessKeya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir uma chave de acesso para um usuário do IAM

O comando `delete-access-key`, apresentado a seguir, exclui a chave de acesso especificada (ID da chave de acesso e chave de acesso secreta) para o usuário do IAM denominado Bob.

```
aws iam delete-access-key \
    --access-key-id AKIDPMS9RO4H3FEXAMPLE \
    --user-name Bob
```

Este comando não produz saída.

Para listar as chaves de acesso definidas para um usuário do IAM, use o comando `list-access-keys`.

Para obter mais informações, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteAccessKeyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
```

```
}
```

- Para obter detalhes da API, consulte DeleteAccessKeya Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessKey>\s

                Where:
                    username - The name of the user.\s
```

```
                    accessKey - The access key ID for the secret access key you
 want to delete.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String
 accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

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

- Para obter detalhes da API, consulte DeleteAccessKeya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a chave de acesso.

```
import { DeleteAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const deleteAccessKey = (userName, accessKeyId) => {
  const command = new DeleteAccessKeyCommand({
    AccessKeyId: accessKeyId,
    UserName: userName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAccessKeya Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  UserName: "USER_NAME",
};

iam.deleteAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAccessKeya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAccessKeyreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui o par de chaves de AWS acesso com o ID **`AKIAIOSFODNN7EXAMPLE`** da chave do usuário chamado**`Bob`**.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Para obter detalhes da API, consulte DeleteAccessKeyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui o par de chaves de AWS acesso com o ID **AKIAIOSFODNN7EXAMPLE** da chave do usuário chamado**Bob**.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Para obter detalhes da API, consulte DeleteAccessKeyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteAccessKeyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
```

```
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create
more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
```

```
    false
  end
end
```

- Para obter detalhes da API, consulte DeleteAccessKeya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_access_key(
    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
            .delete_access_key()
            .user_name(user.user_name())
            .access_key_id(key.access_key_id())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(e) => {
                println!("Can't delete the access key: {:?}", e);
                sleep(Duration::from_secs(2)).await;
            }
        }
    }
    Ok(())
```

```
}
```

- Para obter detalhes da API, consulte a DeleteAccessKeyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->deleteaccesskey(
      iv_accesskeyid = iv_access_key_id
      iv_username = iv_user_name ).
    MESSAGE 'Access key deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Access key or user does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAccessKeyreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func deleteAccessKey(user: IAMClientTypes.User? = nil,
                                key: IAMClientTypes.AccessKey) async throws
    {
        let userName: String?

        if user != nil {
            userName = user!.userName
        } else {
            userName = nil
        }

        let input = DeleteAccessKeyInput(
            accessKeyId: key.accessKeyId,
            userName: userName
        )
        do {
            _ = try await iamClient.deleteAccessKey(input: input)
        } catch {
            print("ERROR: deleteAccessKey:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteAccessKeya referência da API AWS SDK for Swift.

Use **`DeleteAccountAlias`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteAccountAlias`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar sua conta

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteAccountAliasa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir um alias da conta

O comando `delete-account-alias`, apresentado a seguir, remove o alias `mycompany` para a conta atual.

```
aws iam delete-account-alias \
    --account-alias mycompany
```

Este comando não produz saída.

Para obter mais informações, consulte o ID AWS da sua conta e seu alias no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteAccountAliasem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request =
 DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte DeleteAccountAliasa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua o alias da conta.

```
import { DeleteAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias
 */
export const deleteAccountAlias = (alias) => {
  const command = new DeleteAccountAliasCommand({ AccountAlias: alias });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAccountAliasa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAccountAliasa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAccountAliasreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo remove o alias da conta do seu Conta da AWS. A página de login do usuário com o alias em https://mycompanyaws.signin.aws.amazon.com/console não funciona mais. Em vez disso, você deve usar o URL original com seu número de Conta da AWS identificação em https://<accountidnumber>.signin.aws.amazon. com/console.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obter detalhes da API, consulte DeleteAccountAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove o alias da conta do seu Conta da AWS. A página de login do usuário com o alias em https://mycompanyaws.signin.aws.amazon.com/console não funciona mais. Em vez disso, você deve usar o URL original com seu número de Conta da AWS identificação em https://<accountidnumber>.signin.aws.amazon. com/console.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obter detalhes da API, consulte DeleteAccountAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise
```

- Para obter detalhes da API, consulte a DeleteAccountAliasReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, criar e excluir aliases da conta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte DeleteAccountAliasa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->deleteaccountalias(
          iv_accountalias = iv_account_alias ).
        MESSAGE 'Account alias deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Account alias does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAccountAliasreferência da API AWS SDK for SAP ABAP.

Usar **`DeleteAccountPasswordPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteAccountPasswordPolicy`.

CLI

AWS CLI

Excluir a política de senha da conta atual

O comando `delete-account-password-policy` a seguir remove a política de senha da conta atual.

```
aws iam delete-account-password-policy
```

Este comando não produz saída.

Para obter mais informações, consulte Definição de uma política de senhas de contas para usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteAccountPasswordPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo exclui a política de senha do Conta da AWS e redefine todos os valores para seus padrões originais. Se uma política de senha não existir no momento, a seguinte mensagem de erro será exibida: A política de conta com nome PasswordPolicy não pode ser encontrada.

```
Remove-IAMAccountPasswordPolicy
```

- Para obter detalhes da API, consulte DeleteAccountPasswordPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a política de senha do Conta da AWS e redefine todos os valores para seus padrões originais. Se uma política de senha não existir no momento, a seguinte mensagem de erro será exibida: A política de conta com nome PasswordPolicy não pode ser encontrada.

```
Remove-IAMAccountPasswordPolicy
```

- Para obter detalhes da API, consulte DeleteAccountPasswordPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteGroup`.

CLI

AWS CLI

Como excluir um grupo do IAM

O comando `delete-group`, apresentado a seguir, exclui um grupo do IAM denominado `MyTestGroup`.

```
aws iam delete-group \
    --group-name MyTestGroup
```

Este comando não produz saída.

Para obter mais informações, consulte Exclusão de um grupo de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteGroupem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} groupName
 */
export const deleteGroup = async (groupName) => {
  const command = new DeleteGroupCommand({
    GroupName: groupName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte DeleteGroupa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o grupo do IAM denominado **`MyTestGroup`**. O primeiro comando remove todos os usuários do IAM que são membros do grupo, e o segundo exclui o grupo do IAM. Ambos os comandos funcionam sem nenhuma solicitação de confirmação.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Para obter detalhes da API, consulte DeleteGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o grupo do IAM denominado **`MyTestGroup`**. O primeiro comando remove todos os usuários do IAM que são membros do grupo, e o segundo exclui o grupo do IAM. Ambos os comandos funcionam sem nenhuma solicitação de confirmação.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
```

```
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Para obter detalhes da API, consulte DeleteGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteGroupPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteGroupPolicy`.

CLI

AWS CLI

Como excluir uma política de um grupo do IAM

O comando `delete-group-policy`, apresentado a seguir, exclui a política denominada `ExamplePolicy` do grupo denominado `Admins`.

```
aws iam delete-group-policy \
    --group-name Admins \
    --policy-name ExamplePolicy
```

Este comando não produz saída.

Para visualizar as políticas anexadas a um grupo, use o comando `list-group-policies`.

Para obter mais informações, consulte Gerenciamento de políticas do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteGroupPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo remove a política em linha denominada **`TesterPolicy`** do grupo do IAM **`Testers`**. Os usuários desse grupo perdem imediatamente as permissões definidas nessa política.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Para obter detalhes da API, consulte DeleteGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove a política em linha denominada **`TesterPolicy`** do grupo do IAM **`Testers`**. Os usuários desse grupo perdem imediatamente as permissões definidas nessa política.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Para obter detalhes da API, consulte DeleteGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteInstanceProfile`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteInstanceProfile`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e gerenciar um serviço resiliente
- Conceitos básicos de streaming gerenciado
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
```

```
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
```

```
        }
    }
```

- Para obter detalhes da API, consulte DeleteInstanceProfilea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir um perfil de instância

O comando `delete-instance-profile`, apresentado a seguir, exclui o perfil de instância denominado `ExampleInstanceProfile`.

```
aws iam delete-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

Este comando não produz saída.

Para obter mais informações, consulte Usar perfis de instância no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteInstanceProfileem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
      const client = new IAMClient({});
      await client.send(
```

```
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
```

- Para obter detalhes da API, consulte DeleteInstanceProfilea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o perfil de instância do EC2 denominado **`MyAppInstanceProfile`**. O primeiro comando desassocia todos os perfis do perfil de instância e, em seguida, o segundo comando exclui o perfil de instância.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
 Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Para obter detalhes da API, consulte DeleteInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o perfil de instância do EC2 denominado **`MyAppInstanceProfile`**. O primeiro comando desassocia todos os perfis do perfil de instância e, em seguida, o segundo comando exclui o perfil de instância.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
 Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Para obter detalhes da API, consulte DeleteInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo remove o perfil do perfil de instância, separa todas as políticas anexadas ao perfil e exclui todos os recursos.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
```

```
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
```

```
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
 profile_name
                )
```

- Para obter detalhes da API, consulte a DeleteInstanceProfileReferência da API AWS SDK for Python (Boto3).

Usar **`DeleteLoginProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteLoginProfile`.

CLI

AWS CLI

Excluir uma senha de um usuário do IAM

O comando `delete-login-profile` a seguir exclui a senha do usuário do IAM chamado `Bob`.

```
aws iam delete-login-profile \
    --user-name Bob
```

Este comando não produz saída.

Para obter mais informações, consulte Gerenciamento de senhas de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteLoginProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o perfil de login do usuário do IAM denominado **Bob**. Isso impede que o usuário faça login no AWS console. Isso não impede que o usuário execute nenhuma AWS chamada de CLI ou API usando chaves de AWS acesso que ainda possam estar anexadas à conta do usuário. PowerShell

```
Remove-IAMLoginProfile -UserName Bob
```

- Para obter detalhes da API, consulte DeleteLoginProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o perfil de login do usuário do IAM denominado **Bob**. Isso impede que o usuário faça login no AWS console. Isso não impede que o usuário execute nenhuma AWS chamada de CLI ou API usando chaves de AWS acesso que ainda possam estar anexadas à conta do usuário. PowerShell

```
Remove-IAMLoginProfile -UserName Bob
```

- Para obter detalhes da API, consulte DeleteLoginProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteOpenIdConnectProvider`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteOpenIdConnectProvider`.

CLI

AWS CLI

Excluir um provedor de identidade OpenID Connect do IAM

Este exemplo exclui o provedor OIDC do IAM que se conecta ao provedor `example.oidcprovider.com`.

```
aws iam delete-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

Este comando não produz saída.

Para obter mais informações, consulte Criar provedores de identidade OpenID Connect (OIDC) no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteOpenIdConnectProviderem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o provedor OIDC do IAM que se conecta ao provedor **`example.oidcprovider.com`**. Certifique-se de atualizar ou excluir quaisquer perfis que façam referência a esse provedor no elemento **`Principal`** da política de confiança do perfil.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para obter detalhes da API, consulte DeleteOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o provedor OIDC do IAM que se conecta ao provedor **`example.oidcprovider.com`**. Certifique-se de atualizar ou excluir quaisquer perfis que façam referência a esse provedor no elemento **`Principal`** da política de confiança do perfil.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para obter detalhes da API, consulte DeleteOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeletePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeletePolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar usuários somente leitura e leitura/gravação usando
- Como criar um espaço de trabalho de monitoramento gerenciado
- Conceitos básicos de streaming gerenciado
- Conceitos básicos da orquestração de fluxo de trabalaho
- Políticas gerenciadas
- Como configurar o gerenciamento de sistemas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeletePolicya Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
```

```
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
```

```
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}
```

- Para obter detalhes da API, consulte DeletePolicyem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
```

```
                    << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeletePolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir uma política do IAM

Este exemplo exclui a política cujo ARN é `arn:aws:iam::123456789012:policy/MySamplePolicy`.

```
aws iam delete-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Este comando não produz saída.

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeletePolicyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeletePolicya Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <policyARN>\s

                Where:
                    policyARN - A policy ARN value to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
```

```
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(policyARN)
                    .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte DeletePolicya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a política.

```
import { DeletePolicyCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const deletePolicy = (policyArn) => {
  const command = new DeletePolicyCommand({ PolicyArn: policyArn });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeletePolicya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- Para obter detalhes da API, consulte a DeletePolicyreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui a política cujo ARN é **`arn:aws:iam::123456789012:policy/MySamplePolicy`**. Antes de excluir a política, exclua primeiro todas as versões, exceto a padrão, executando **`Remove-IAMPolicyVersion`**. Você também deve desassociar a política de qualquer usuário, grupo ou perfil do IAM.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Exemplo 2: este exemplo exclui uma política excluindo primeiro todas as versões não padrão da política, desassociando-a de todas as entidades do IAM anexadas e, por fim, excluindo a própria política. A primeira linha recupera o objeto da política. A segunda linha recupera todas as versões da política que não estão marcadas como a versão padrão em uma compilação e depois exclui cada política na compilação. A terceira linha recupera todos os usuários, grupos e perfis do IAM aos quais a política está anexada. As linhas de quatro a seis desassociam a política de cada entidade anexada. A última linha usa esse comando para remover a política gerenciada e a versão padrão restante. O exemplo inclui o parâmetro switch **`-Force`** em qualquer linha que precise dele para suprimir solicitações de confirmação.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- Para obter detalhes da API, consulte DeletePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui a política cujo ARN é **`arn:aws:iam::123456789012:policy/MySamplePolicy`**. Antes de excluir a política, exclua primeiro todas as versões, exceto a padrão, executando **`Remove-`**

`IAMPolicyVersion`. Você também deve desassociar a política de qualquer usuário, grupo ou perfil do IAM.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Exemplo 2: este exemplo exclui uma política excluindo primeiro todas as versões não padrão da política, desassociando-a de todas as entidades do IAM anexadas e, por fim, excluindo a própria política. A primeira linha recupera o objeto da política. A segunda linha recupera todas as versões da política que não estão marcadas como a versão padrão em uma compilação e depois exclui cada política na compilação. A terceira linha recupera todos os usuários, grupos e perfis do IAM aos quais a política está anexada. As linhas de quatro a seis desassociam a política de cada entidade anexada. A última linha usa esse comando para remover a política gerenciada e a versão padrão restante. O exemplo inclui o parâmetro switch **`-Force`** em qualquer linha que precise dele para suprimir solicitações de confirmação.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- Para obter detalhes da API, consulte DeletePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_policy(policy_arn):
```

```
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

- Para obter detalhes da API, consulte a DeletePolicyReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
 iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeletePolicyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->deletepolicy( iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Policy cannot be deleted due to attachments.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeletePolicyreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func deletePolicy(policy: IAMClientTypes.Policy) async throws {
        let input = DeletePolicyInput(
            policyArn: policy.arn
```

```
        )
        do {
            _ = try await iamClient.deletePolicy(input: input)
        } catch {
            print("ERROR: deletePolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeletePolicya referência da API AWS SDK for Swift.

Use **`DeletePolicyVersion`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeletePolicyVersion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Políticas gerenciadas
- Reverter uma versão de política

CLI

AWS CLI

Para excluir uma versão de uma política gerenciada

Este exemplo exclui a versão identificada como v2 da política cujo ARN é `arn:aws:iam::123456789012:policy/MySamplePolicy`.

```
aws iam delete-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

Este comando não produz saída.

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeletePolicyVersionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui a versão identificada como **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MySamplePolicy**.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MySamplePolicy -VersionID v2
```

Exemplo 2: este exemplo exclui uma política excluindo primeiro todas as versões não padrão da política e depois excluindo a própria política. A primeira linha recupera o objeto da política. A segunda linha recupera todas as versões da política que não estão marcadas como padrão em uma compilação e, em seguida, usa esse comando para excluir cada política na compilação. A última linha remove a política em si, bem como a versão padrão restante. Observe que, para excluir com êxito uma política gerenciada, você também deve desassociar a política de qualquer usuário, grupo ou perfis usando os comandos **Unregister-IAMUserPolicy**, **Unregister-IAMGroupPolicy** e **Unregister-IAMRolePolicy**. Veja o exemplo do cmdlet **Remove-IAMPolicy**.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- Para obter detalhes da API, consulte DeletePolicyVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui a versão identificada como **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MySamplePolicy**.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MySamplePolicy -VersionID v2
```

Exemplo 2: este exemplo exclui uma política excluindo primeiro todas as versões não padrão da política e depois excluindo a própria política. A primeira linha recupera o objeto da política. A segunda linha recupera todas as versões da política que não estão marcadas como padrão em uma compilação e, em seguida, usa esse comando para excluir cada política na compilação. A última linha remove a política em si, bem como a versão padrão restante. Observe que, para excluir com êxito uma política gerenciada, você também deve desassociar a política de qualquer usuário, grupo ou perfis usando os comandos **`Unregister-IAMUserPolicy`**, **`Unregister-IAMGroupPolicy`** e **`Unregister-IAMRolePolicy`**. Veja o exemplo do cmdlet **`Remove-IAMPolicy`**.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- Para obter detalhes da API, consulte DeletePolicyVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->deletepolicyversion(
          iv_policyarn = iv_policy_arn
          iv_versionid = iv_version_id ).
        MESSAGE 'Policy version deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy or version does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Cannot delete default policy version.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeletePolicyVersionreferência da API AWS SDK for SAP ABAP.

Use **`DeleteRole`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteRole`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Configurar a conectividade do serviço de contêiner
- Criar uma API REST com integração de proxy de função
- Como criar um espaço de trabalho de monitoramento gerenciado
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Conceitos básicos de data warehouses de tecnologia sem servidor
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de arquivos de atributos de machine learning
- Conceitos básicos de clusters gerenciados de kubernetes
- Conceitos básicos de streaming gerenciado
- Conceitos básicos de clusters de data warehouse provisionados
- Conceitos básicos da orquestração de fluxo de trabalaho
- Gerenciar funções
- Como mover segredos codificados para o armazenamento seguro de segredos
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Como configurar o gerenciamento de sistemas
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteRolea Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
```

```
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
```

```
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}
```

- Para obter detalhes da API, consulte DeleteRoleem Referência de AWS CLI Comandos.

CLI

AWS CLI

Como excluir um perfil do IAM

O comando `delete-role`, apresentado a seguir, remove o perfil denominado `Test-Role`.

```
aws iam delete-role \
    --role-name Test-Role
```

Este comando não produz saída.

Antes de poder excluir um perfil, você deve removê-lo de qualquer perfil de instância (`remove-role-from-instance-profile`), desanexar quaisquer políticas gerenciadas (`detach-role-policy`) e excluir quaisquer políticas em linha anexadas a ele (`delete-role-policy`).

Para obter mais informações, consulte Criação de perfis do IAM e Usar perfis de instância no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteRoleem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error
 {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteRolea Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua o perfil.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteRolea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o perfil denominado **MyNewRole** da conta atual do IAM. Antes de excluir o perfil, use primeiro o comando **Unregister-IAMRolePolicy** para desassociar todas as políticas gerenciadas. As políticas em linha são excluídas com o perfil.

```
Remove-IAMRole -RoleName MyNewRole
```

Exemplo 2: este exemplo desassocia todas as políticas gerenciadas do perfil denominado **MyNewRole** e depois o exclui. A primeira linha recupera todas as políticas gerenciadas anexadas ao perfil como uma compilação e, em seguida, desassocia cada política da compilação do perfil. A segunda linha exclui o perfil em si. As políticas em linha são excluídas com o perfil.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- Para obter detalhes da API, consulte DeleteRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o perfil denominado **MyNewRole** da conta atual do IAM. Antes de excluir o perfil, use primeiro o comando **Unregister-IAMRolePolicy** para desassociar todas as políticas gerenciadas. As políticas em linha são excluídas com o perfil.

```
Remove-IAMRole -RoleName MyNewRole
```

Exemplo 2: este exemplo desassocia todas as políticas gerenciadas do perfil denominado **MyNewRole** e depois o exclui. A primeira linha recupera todas as políticas gerenciadas anexadas ao perfil como uma compilação e, em seguida, desassocia cada política da compilação do perfil. A segunda linha exclui o perfil em si. As políticas em linha são excluídas com o perfil.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- Para obter detalhes da API, consulte DeleteRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
```

```
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteRoleReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Deletes a role and its attached policies.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    # Detach and delete attached policies
    @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
      response.attached_policies.each do |policy|
        @iam_client.detach_role_policy({
                                        role_name: role_name,
                                        policy_arn: policy.policy_arn
                                      })
        # Check if the policy is a customer managed policy (not AWS managed)
        unless policy.policy_arn.include?('aws:policy/')
          @iam_client.delete_policy({ policy_arn: policy.policy_arn })
          @logger.info("Deleted customer managed policy #{policy.policy_name}.")
        end
      end
    end
```

```
    # Delete the role
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Deleted role #{role_name}.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
 why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteRolea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError>
 {
    let role = role.clone();
    while client
        .delete_role()
        .role_name(role.role_name())
        .send()
        .await
        .is_err()
    {
        sleep(Duration::from_secs(2)).await;
    }
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteRolereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->deleterole( iv_rolename = iv_role_name ).
        MESSAGE 'Role deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Role cannot be deleted due to attached resources.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteRolereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func deleteRole(role: IAMClientTypes.Role) async throws {
        let input = DeleteRoleInput(
```

```
            roleName: role.roleName
        )
        do {
            _ = try await iamClient.deleteRole(input: input)
        } catch {
            print("ERROR: deleteRole:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteRolea referência da API AWS SDK for Swift.

Usar **`DeleteRolePermissionsBoundary`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteRolePermissionsBoundary`.

CLI

AWS CLI

Excluir um limite de permissões de um perfil do IAM

O exemplo de `delete-role-permissions-boundary` a seguir exclui o limite de permissões do perfil do IAM especificado. Para aplicar um limite de permissões a um perfil, use o comando `put-role-permissions-boundary`.

```
aws iam delete-role-permissions-boundary \
    --role-name lambda-application-role
```

Este comando não produz saída.

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteRolePermissionsBoundaryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo mostra como remover o limite de permissões anexado a um perfil do IAM.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Para obter detalhes da API, consulte DeleteRolePermissionsBoundaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra como remover o limite de permissões anexado a um perfil do IAM.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Para obter detalhes da API, consulte DeleteRolePermissionsBoundaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteRolePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteRolePolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos de data warehouses de tecnologia sem servidor
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de clusters de data warehouse provisionados
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteRolePolicya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como remover uma política de um perfil do IAM

O comando `delete-role-policy`, apresentado a seguir, remove a política denominada `ExamplePolicy` do perfil denominado `Test-Role`.

```
aws iam delete-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

Este comando não produz saída.

Para obter mais informações, consulte Modificar um perfil no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteRolePolicyem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 */
export const deleteRolePolicy = (roleName, policyName) => {
  const command = new DeleteRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
  });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteRolePolicya Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui a política em linha **`S3AccessPolicy`** incorporada no perfil do IAM **`S3BackupRole`**.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Para obter detalhes da API, consulte DeleteRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui a política em linha **`S3AccessPolicy`** incorporada no perfil do IAM **`S3BackupRole`**.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Para obter detalhes da API, consulte DeleteRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteSAMLProvider com um`** AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteSAMLProvider`.

CLI

AWS CLI

Como excluir um provedor SAML

Este exemplo exclui o provedor SAML 2.0 do IAM cujo ARN é `arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`.

```
aws iam delete-saml-provider \
--saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider
```

Este comando não produz saída.

Para obter mais informações, consulte Criação de provedores de identidade SAML do IAM no Guia do usuário do AWS IAM.

- Consulte detalhes da API em DeleteSAMLProvider na Referência de comandos da AWS CLI .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} providerArn
 * @returns
 */
export const deleteSAMLProvider = async (providerArn) => {
  const command = new DeleteSAMLProviderCommand({
    SAMLProviderArn: providerArn,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Consulte detalhes da API em DeleteSAMLProvider na Referência da API do AWS SDK para JavaScript .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o provedor SAML 2.0 do IAM cujo ARN é **`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`**.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- Consulte detalhes da API em DeleteSAMLProvider na Referência de cmdlets do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o provedor SAML 2.0 do IAM cujo ARN é **`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`**.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- Para obter detalhes sobre a API, consulte DeleteSAMLProvider na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Use **`DeleteServerCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteServerCertificate`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
```

```
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " <<
 certificateName
                  << std::endl;
    }

    return result;
}
```

- Para obter detalhes da API, consulte DeleteServerCertificatea Referência AWS SDK para C ++ da API.

CLI

AWS CLI

Para excluir um certificado de servidor da sua AWS conta

O `delete-server-certificate` comando a seguir remove o certificado de servidor especificado da sua AWS conta.

```
aws iam delete-server-certificate \
```

```
    --server-certificate-name myUpdatedServerCertificate
```

Este comando não produz saída.

Para listar os certificados do servidor disponíveis na sua AWS conta, use o `list-server-certificates` comando.

Para obter mais informações, consulte Gerenciar certificados de servidor no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteServerCertificateem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir um certificado de servidor.

```
import { DeleteServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 */
export const deleteServerCertificate = (certName) => {
  const command = new DeleteServerCertificateCommand({
    ServerCertificateName: certName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteServerCertificatea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteServerCertificatea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o certificado do servidor denominado **`MyServerCert`**.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Para obter detalhes da API, consulte DeleteServerCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o certificado do servidor denominado **`MyServerCert`**.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Para obter detalhes da API, consulte DeleteServerCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, atualizar e excluir certificados de servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
```

```
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end
```

```
  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte DeleteServerCertificatea Referência AWS SDK para Ruby da API.

Use **`DeleteServiceLinkedRole`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteServiceLinkedRole`.

CLI

AWS CLI

Como excluir um perfil vinculado ao serviço

O exemplo para `delete-service-linked-role`, apresentado a seguir, exclui o perfil vinculado ao serviço especificado que não é mais necessário. A exclusão acontece de forma assíncrona. É possível verificar o status da exclusão e confirmar quando ela for concluída ao usar o comando `get-service-linked-role-deletion-status`.

```
aws iam delete-service-linked-role \
    --role-name AWSServiceRoleForLexBots
```

Saída:

```
{
    "DeletionTaskId": "task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE"
}
```

Para obter mais informações, consulte Usar perfis vinculados ao serviço no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteServiceLinkedRoleem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
```

```
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteServiceLinkedRolea Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteServiceLinkedRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteServiceLinkedRole = (roleName) => {
  const command = new DeleteServiceLinkedRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteServiceLinkedRolea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo excluiu o perfil vinculado ao serviço. Observe que, se o serviço ainda estiver usando esse perfil, esse comando resultará em uma falha.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Para obter detalhes da API, consulte DeleteServiceLinkedRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo excluiu o perfil vinculado ao serviço. Observe que, se o serviço ainda estiver usando esse perfil, esse comando resultará em uma falha.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Para obter detalhes da API, consulte DeleteServiceLinkedRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Deletes a service-linked role.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_service_linked_role(role_name)
    response = @iam_client.delete_service_linked_role(role_name: role_name)
```

```
    task_id = response.deletion_task_id
    check_deletion_status(role_name, task_id)
  rescue Aws::Errors::ServiceError => e
    handle_deletion_error(e, role_name)
  end

  private

  # Checks the deletion status of a service-linked role
  #
  # @param role_name [String] The name of the role being deleted
  # @param task_id [String] The task ID for the deletion process
  def check_deletion_status(role_name, task_id)
    loop do
      response = @iam_client.get_service_linked_role_deletion_status(
        deletion_task_id: task_id
      )
      status = response.status
      @logger.info("Deletion of #{role_name} #{status}.")
      break if %w[SUCCEEDED FAILED].include?(status)

      sleep(3)
    end
  end

  # Handles deletion error
  #
  # @param e [Aws::Errors::ServiceError] The error encountered during deletion
  # @param role_name [String] The name of the role attempted to delete
  def handle_deletion_error(e, role_name)
    return if e.code == 'NoSuchEntity'

    @logger.error("Couldn't delete #{role_name}. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteServiceLinkedRolea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteServiceLinkedRolereferência da API AWS SDK for Rust.

Usar **`DeleteSigningCertificate`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteSigningCertificate`.

CLI

AWS CLI

Excluir um certificado de assinatura de um usuário do IAM

O comando `delete-signing-certificate` a seguir exclui o certificado de assinatura especificado do usuário do IAM chamado Bob.

```
aws iam delete-signing-certificate \
```

```
    --user-name Bob \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE
```

Este comando não produz saída.

Para obter o ID de um certificado de assinatura, use o comando `list-signing-certificates`.

Para obter mais informações, consulte Manage signing certificates no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte DeleteSigningCertificateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o certificado de assinatura com o ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** do usuário do IAM chamado **`Bob`**.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Para obter detalhes da API, consulte DeleteSigningCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o certificado de assinatura com o ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** do usuário do IAM chamado **`Bob`**.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Para obter detalhes da API, consulte DeleteSigningCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteUser`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteUser`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar usuários somente leitura e leitura/gravação usando

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
```

```
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- Para obter detalhes da API, consulte DeleteUserem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteUserRequest request;
    request.SetUserName(userName);
    auto outcome = iam.DeleteUser(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting IAM user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;;
    }
```

```
    else {
        std::cout << "Successfully deleted IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
```

- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir um usuário do IAM

O comando `delete-user`, apresentado a seguir, remove o usuário do IAM denominado Bob da conta atual.

```
aws iam delete-user \
    --user-name Bob
```

Este comando não produz saída.

Para obter mais informações, consulte Exclusão de um usuário do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteUserem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
```

```
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error
 {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
```

```
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

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

- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua o usuário.

```
import { DeleteUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const deleteUser = (name) => {
  const command = new DeleteUserCommand({ UserName: name });
  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteUserreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o usuário do IAM chamado **Bob**.

```
Remove-IAMUser -UserName Bob
```

Exemplo 2: este exemplo exclui a usuária do IAM chamada **`Theresa`** com todos os elementos que devem ser excluídos primeiro.

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
 $pol.PolicyArn -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

```
# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- Para obter detalhes da API, consulte DeleteUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o usuário do IAM chamado **Bob**.

```
Remove-IAMUser -UserName Bob
```

Exemplo 2: este exemplo exclui a usuária do IAM chamada **Theresa** com todos os elementos que devem ser excluídos primeiro.

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
 $pol.PolicyArn -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }
```

```
# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- Para obter detalhes da API, consulte DeleteUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
```

```
        raise
```

- Para obter detalhes da API, consulte a DeleteUserReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
 end
```

- Para obter detalhes da API, consulte DeleteUsera Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
 SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<(), SdkError<DeleteUserError>> = loop {
        match client
            .delete_user()
            .user_name(user.user_name())
            .send()
            .await
        {
            Ok(_) => {
                break Ok(());
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    response
}
```

- Para obter detalhes da API, consulte a DeleteUserreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->deleteuser( iv_username = iv_user_name ).
        MESSAGE 'User deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'User cannot be deleted due to attached resources.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteUserreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func deleteUser(user: IAMClientTypes.User) async throws {
        let input = DeleteUserInput(
```

```
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUser(input: input)
        } catch {
            print("ERROR: deleteUser:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteUsera referência da API AWS SDK for Swift.

Usar **`DeleteUserPermissionsBoundary`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteUserPermissionsBoundary`.

CLI

AWS CLI

Excluir um limite de permissões de um usuário do IAM

O exemplo `delete-user-permissions-boundary` a seguir exclui o limite de permissões anexado ao usuário do IAM chamado `intern`. Para aplicar um limite de permissões a um usuário, use o comando `put-user-permissions-boundary`.

```
aws iam delete-user-permissions-boundary \
    --user-name intern
```

Este comando não produz saída.

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteUserPermissionsBoundaryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo mostra como remover o limite de permissões anexado a um usuário do IAM.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Para obter detalhes da API, consulte DeleteUserPermissionsBoundaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra como remover o limite de permissões anexado a um usuário do IAM.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Para obter detalhes da API, consulte DeleteUserPermissionsBoundaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteUserPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteUserPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
 userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
 DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteUserPolicya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como remover uma política de um usuário do IAM

O comando `delete-user-policy`, apresentado a seguir, remove a política especificada do usuário do IAM denominado Bob.

```
aws iam delete-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

Este comando não produz saída.

Para obter uma lista de políticas para um usuário do IAM, use o comando `list-user-policies`.

Para obter mais informações, consulte Como criar um usuário do IAM em sua AWS conta no Guia AWS do usuário do IAM.

- Para obter detalhes da API, consulte DeleteUserPolicyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
```

```
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteUserPolicya Referência AWS SDK para Go da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui a política em linha denominada **`AccessToEC2Policy`** incorporada no usuário do IAM chamado **`Bob`**.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Exemplo 2: este exemplo encontra todas as políticas em linha incorporadas na usuária do IAM chamada **`Theresa`** e depois as exclui.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- Para obter detalhes da API, consulte DeleteUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui a política em linha denominada **`AccessToEC2Policy`** incorporada no usuário do IAM chamado **`Bob`**.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Exemplo 2: este exemplo encontra todas as políticas em linha incorporadas na usuária do IAM chamada **`Theresa`** e depois as exclui.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- Para obter detalhes da API, consulte DeleteUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
 end
```

- Para obter detalhes da API, consulte DeleteUserPolicya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteUserPolicyreferência da API AWS SDK for Rust.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
```

```
import AWSS3


    func deleteUserPolicy(user: IAMClientTypes.User, policyName: String) async
 throws {
        let input = DeleteUserPolicyInput(
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUserPolicy(input: input)
        } catch {
            print("ERROR: deleteUserPolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteUserPolicya referência da API AWS SDK for Swift.

Usar **`DeleteVirtualMfaDevice`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteVirtualMfaDevice`.

CLI

AWS CLI

Remover um dispositivo de MFA virtual

O comando `delete-virtual-mfa-device` a seguir remove o dispositivo de MFA especificado da conta atual.

```
aws iam delete-virtual-mfa-device \
    --serial-number arn:aws:iam::123456789012:mfa/MFATest
```

Este comando não produz saída.

Para obter mais informações, consulte Desativar dispositivos de MFA no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DeleteVirtualMfaDeviceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o dispositivo de MFA virtual do IAM cujo ARN é **`arn:aws:iam::123456789012:mfa/bob`**.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Exemplo 2: este exemplo verifica se a usuária do IAM Theresa tem um dispositivo de MFA atribuído. Se for encontrado, o dispositivo é desabilitado para a usuária do IAM. Se o dispositivo for virtual, ele também é excluído.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

- Para obter detalhes da API, consulte DeleteVirtualMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o dispositivo de MFA virtual do IAM cujo ARN é **`arn:aws:iam::123456789012:mfa/bob`**.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Exemplo 2: este exemplo verifica se a usuária do IAM Theresa tem um dispositivo de MFA atribuído. Se for encontrado, o dispositivo é desabilitado para a usuária do IAM. Se o dispositivo for virtual, ele também é excluído.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
```

```
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

- Para obter detalhes da API, consulte DeleteVirtualMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DetachGroupPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DetachGroupPolicy`.

CLI

AWS CLI

Para desanexar uma política de um grupo

Este exemplo remove a política gerenciada com o ARN `arn:aws:iam::123456789012:policy/TesterAccessPolicy` do grupo denominado `Testers`.

```
aws iam detach-group-policy \
    --group-name Testers \
    --policy-arn arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Este comando não produz saída.

Para obter mais informações, consulte Gerenciar grupos de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DetachGroupPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo desassocia a política de grupo gerenciado cujo ARN é **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** do grupo denominado **`Testers`**.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao grupo denominado **`Testers`** e as desassocia do grupo.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- Para obter detalhes da API, consulte DetachGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo desassocia a política de grupo gerenciado cujo ARN é **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** do grupo denominado **`Testers`**.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao grupo denominado **`Testers`** e as desassocia do grupo.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- Para obter detalhes da API, consulte DetachGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DetachRolePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetachRolePolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Configurar a conectividade do serviço de contêiner
- Criar uma API REST com integração de proxy de função

- Como criar um espaço de trabalho de monitoramento gerenciado
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de arquivos de atributos de machine learning
- Conceitos básicos de clusters gerenciados de kubernetes
- Conceitos básicos de streaming gerenciado
- Conceitos básicos da orquestração de fluxo de trabalaho
- Gerenciar funções
- Como mover segredos codificados para o armazenamento seguro de segredos
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Como configurar o gerenciamento de sistemas
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
```

```
        var response = await _IAMService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DetachRolePolicya Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
```

```
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte DetachRolePolicyem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
    detachRequest.SetRoleName(roleName);
```

```
    detachRequest.SetPolicyArn(policyArn);

    auto detachOutcome = iam.DetachRolePolicy(detachRequest);
    if (!detachOutcome.IsSuccess()) {
        std::cerr << "Failed to detach policy " << policyArn << " from role "
                  << roleName << ": " << detachOutcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully detached policy " << policyArn << " from role
 "
                  << roleName << std::endl;
    }

    return detachOutcome.IsSuccess();
```

- Para obter detalhes da API, consulte DetachRolePolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como desanexar uma política de um perfil

Este exemplo remove a política gerenciada com o ARN `arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy` do perfil denominado `FedTesterRole`.

```
aws iam detach-role-policy \
    --role-name FedTesterRole \
    --policy-arn arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Este comando não produz saída.

Para obter mais informações, consulte Modificar um perfil no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DetachRolePolicyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
```

```
}
```

- Para obter detalhes da API, consulte DetachRolePolicya Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
```

```
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                    " from role " + roleName);

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

- Para obter detalhes da API, consulte DetachRolePolicya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Desanexe a política.

```
import { DetachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const detachRolePolicy = (policyArn, roleName) => {
  const command = new DetachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DetachRolePolicya Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        var params = {
          PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
          RoleName: process.argv[2],
        };
        iam.detachRolePolicy(params, function (err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
```

```
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DetachRolePolicya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DetachRolePolicyreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo desassocia a política de grupo gerenciado cujo ARN é **`arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy`** do perfil denominado **`FedTesterRole`**.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao perfil denominado **`FedTesterRole`** e as desassocia dele.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
 -Rolename FedTesterRole
```

- Para obter detalhes da API, consulte DetachRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo desassocia a política de grupo gerenciado cujo ARN é **`arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy`** do perfil denominado **`FedTesterRole`**.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao perfil denominado **`FedTesterRole`** e as desassocia dele.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
 -Rolename FedTesterRole
```

- Para obter detalhes da API, consulte DetachRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Desanexar uma política de uma função usando o objeto Policy do Boto3.

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

Desanexar uma política de uma função usando o objeto Role do Boto3.

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
```

```
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

- Para obter detalhes da API, consulte a DetachRolePolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
```

```
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
```

```
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte DetachRolePolicya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a DetachRolePolicyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        lo_iam->detachrolepolicy(
          iv_rolename = iv_role_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy detached from role successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role or policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetachRolePolicyreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func detachRolePolicy(policy: IAMClientTypes.Policy, role:
 IAMClientTypes.Role) async throws {
        let input = DetachRolePolicyInput(
            policyArn: policy.arn,
            roleName: role.roleName
        )

        do {
            _ = try await iamClient.detachRolePolicy(input: input)
        } catch {
            print("ERROR: detachRolePolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DetachRolePolicya referência da API AWS SDK for Swift.

Use **`DetachUserPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetachUserPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar usuários somente leitura e leitura/gravação usando

CLI

AWS CLI

Como desanexar uma política de um usuário

Este exemplo remove a política gerenciada com o ARN `arn:aws:iam::123456789012:policy/TesterPolicy` do usuário `Bob`.

```
aws iam detach-user-policy \
    --user-name Bob \
    --policy-arn arn:aws:iam::123456789012:policy/TesterPolicy
```

Este comando não produz saída.

Para obter mais informações, consulte Alteração de permissões de um usuário do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DetachUserPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo desassocia a política gerenciada cujo ARN é **`arn:aws:iam::123456789012:policy/TesterPolicy`** do usuário do IAM chamado **`Bob`**.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao usuário do IAM chamado **`Theresa`** e as desassocia dele.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- Para obter detalhes da API, consulte DetachUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo desassocia a política gerenciada cujo ARN é **`arn:aws:iam::123456789012:policy/TesterPolicy`** do usuário do IAM chamado **`Bob`**.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao usuário do IAM chamado **`Theresa`** e as desassocia dele.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- Para obter detalhes da API, consulte DetachUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

- Para obter detalhes da API, consulte a DetachUserPolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Detaches a policy from a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The ARN of the policy to detach
  # @return [Boolean] true if the policy was successfully detached, false
 otherwise
  def detach_user_policy(user_name, policy_arn)
    @iam_client.detach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
 successfully.")
    true
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error('Error detaching policy: Policy or user does not exist.')
    false
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from user '#{user_name}':
 #{e.message}")
    false
  end
```

- Para obter detalhes da API, consulte DetachUserPolicya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
```

```
) -> Result<(), iamError> {
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a DetachUserPolicyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->detachuserpolicy(
          iv_username = iv_user_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy detached from user successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User or policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetachUserPolicyreferência da API AWS SDK for SAP ABAP.

Usar **`EnableMfaDevice`** com uma CLI

Os exemplos de código a seguir mostram como usar o `EnableMfaDevice`.

CLI

AWS CLI

Habilitar um dispositivo de MFA

Depois de usar o comando `create-virtual-mfa-device` para criar um dispositivo de MFA virtual, você pode atribuir o dispositivo de MFA a um usuário. O exemplo de `enable-mfa-device` a seguir atribui o dispositivo de MFA com o número de série `arn:aws:iam::210987654321:mfa/BobsMFADevice` ao usuário Bob. O comando também sincroniza o dispositivo com AWS a inclusão dos dois primeiros códigos em sequência do dispositivo de MFA virtual.

```
aws iam enable-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 789012
```

Este comando não produz saída.

Para obter mais informações, consulte Habilitar um dispositivo de autenticação multifator (MFA) virtual no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte EnableMfaDeviceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando habilita o dispositivo de MFA de hardware com o número de série **`987654321098`** e associa o dispositivo ao usuário **`Bob`**. Ele inclui os dois primeiros códigos em sequência do dispositivo.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Exemplo 2: este exemplo cria e habilita um dispositivo de MFA virtual. O primeiro comando cria o dispositivo virtual e retorna a representação de objeto do dispositivo na variável **`$MFADevice`**. Você pode usar as propriedades **`.Base32StringSeed`** ou **`QRCodePng`** para configurar a aplicação de software do usuário. O comando final atribui o dispositivo ao usuário **`David`**, identificando o dispositivo pelo número de série. O comando também sincroniza o dispositivo com AWS a inclusão dos dois primeiros códigos em sequência do dispositivo de MFA virtual.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- Para obter detalhes da API, consulte EnableMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando habilita o dispositivo de MFA de hardware com o número de série **`987654321098`** e associa o dispositivo ao usuário **`Bob`**. Ele inclui os dois primeiros códigos em sequência do dispositivo.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Exemplo 2: este exemplo cria e habilita um dispositivo de MFA virtual. O primeiro comando cria o dispositivo virtual e retorna a representação de objeto do dispositivo na variável **`$MFADevice`**. Você pode usar as propriedades **`.Base32StringSeed`** ou **`QRCodePng`** para configurar a aplicação de software do usuário. O comando final atribui o dispositivo ao usuário **`David`**, identificando o dispositivo pelo número de série. O comando também sincroniza o dispositivo com AWS a inclusão dos dois primeiros códigos em sequência do dispositivo de MFA virtual.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
```

```
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- Para obter detalhes da API, consulte EnableMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **GenerateCredentialReport**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GenerateCredentialReport`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar sua conta

CLI

AWS CLI

Como gerar um relatório de credenciais

O exemplo a seguir tenta gerar um relatório de credenciais para a AWS conta.

```
aws iam generate-credential-report
```

Saída:

```
{
    "State":  "STARTED",
    "Description": "No report exists. Starting a new report generation task"
}
```

Para obter mais informações, consulte Obter relatórios de credenciais para sua AWS conta no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GenerateCredentialReportem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo solicita a geração de um novo relatório, que pode ser feito a cada quatro horas. Se o último relatório ainda for recente, o campo Estado será **`COMPLETE`**. Use **`Get-IAMCredentialReport`** para visualizar o relatório completo.

```
Request-IAMCredentialReport
```

Saída:

```
Description                                                  State
-----------                                                  -----
No report exists. Starting a new report generation task      STARTED
```

- Para obter detalhes da API, consulte GenerateCredentialReportem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo solicita a geração de um novo relatório, que pode ser feito a cada quatro horas. Se o último relatório ainda for recente, o campo Estado será **`COMPLETE`**. Use **`Get-IAMCredentialReport`** para visualizar o relatório completo.

```
Request-IAMCredentialReport
```

Saída:

```
Description                                                  State
-----------                                                  -----
No report exists. Starting a new report generation task      STARTED
```

- Para obter detalhes da API, consulte GenerateCredentialReportem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response
```

- Para obter detalhes da API, consulte a GenerateCredentialReportReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->generatecredentialreport( ).
    MESSAGE 'Credential report generation started.' TYPE 'I'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Report generation limit exceeded.' TYPE 'E'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when generating credential report.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GenerateCredentialReportreferência da API AWS SDK for SAP ABAP.

Usar **`GenerateServiceLastAccessedDetails`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GenerateServiceLastAccessedDetails`.

CLI

AWS CLI

Exemplo 1: para gerar um relatório de acesso ao serviço de uma política personalizada

O exemplo de `generate-service-last-accessed-details` a seguir inicia um trabalho em segundo plano para gerar um relatório que lista os serviços acessados pelos usuários do IAM e outras entidades com uma política personalizada denominada `intern-boundary`. Você pode exibir o relatório após a criação executando o comando `get-service-last-accessed-details`.

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::123456789012:policy/intern-boundary
```

Saída:

```
{
    "JobId": "2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc"
}
```

Exemplo 2: Para gerar um relatório de acesso ao serviço para a AdministratorAccess política AWS gerenciada

O `generate-service-last-accessed-details` exemplo a seguir inicia um trabalho em segundo plano para gerar um relatório que lista os serviços acessados pelos usuários do IAM e outras entidades com a `AdministratorAccess` política AWS gerenciada. Você pode exibir o relatório após a criação executando o comando `get-service-last-accessed-details`.

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::aws:policy/AdministratorAccess
```

Saída:

```
{
    "JobId": "78b6c2ba-d09e-6xmp-7039-ecde30b26916"
}
```

Para obter mais informações, consulte Como refinar as permissões no AWS uso das informações do último acesso no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GenerateServiceLastAccessedDetailsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo é um cmdlet equivalente da GenerateServiceLastAccessedDetails API. Isso fornece um ID de trabalho que pode ser usado em Get-IAMServiceLastAccessedDetail e Get-IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Para obter detalhes da API, consulte GenerateServiceLastAccessedDetailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo é um cmdlet equivalente da GenerateServiceLastAccessedDetails API. Isso fornece um ID de trabalho que pode ser usado em Get-IAMServiceLastAccessedDetail e Get-IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Para obter detalhes da API, consulte GenerateServiceLastAccessedDetailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetAccessKeyLastUsed`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetAccessKeyLastUsed`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar chaves de acesso

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
```

```
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome =
 iam.GetAccessKeyLastUsed(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
                  secretKeyID << ":" << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
                outcome.GetResult()
                        .GetAccessKeyLastUsed()
                        .GetLastUsedDate()
                        .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
                  lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetAccessKeyLastUsed a Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para recuperar informações sobre quando a chave de acesso especificada foi usada pela última vez

O exemplo apresentado a seguir recupera informações sobre quando a chave de acesso `ABCDEXAMPLE` foi usada pela última vez.

```
aws iam get-access-key-last-used \
    --access-key-id ABCDEXAMPLE
```

Saída:

```
{
    "UserName":  "Bob",
    "AccessKeyLastUsed": {
        "Region": "us-east-1",
        "ServiceName": "iam",
        "LastUsedDate": "2015-06-16T22:45:00Z"
    }
}
```

Para obter mais informações, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetAccessKeyLastUsedem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a chave de acesso.

```
import { GetAccessKeyLastUsedCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} accessKeyId
 */
export const getAccessKeyLastUsed = async (accessKeyId) => {
  const command = new GetAccessKeyLastUsedCommand({
    AccessKeyId: accessKeyId,
  });
```

```
  const response = await client.send(command);

  if (response.AccessKeyLastUsed?.LastUsedDate) {
    console.log(`
    ${accessKeyId} was last used by ${response.UserName} via
    the ${response.AccessKeyLastUsed.ServiceName} service on
    ${response.AccessKeyLastUsed.LastUsedDate.toISOString()}
    `);
  }

  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetAccessKeyLastUsedda Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getAccessKeyLastUsed(
  { AccessKeyId: "ACCESS_KEY_ID" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.AccessKeyLastUsed);
```

```
    }
  }
);
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetAccessKeyLastUseda Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna o nome de usuário proprietário e as informações do último uso da chave de acesso fornecida.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Para obter detalhes da API, consulte GetAccessKeyLastUsedem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna o nome de usuário proprietário e as informações do último uso da chave de acesso fornecida.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Para obter detalhes da API, consulte GetAccessKeyLastUsedem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- Para obter detalhes da API, consulte a GetAccessKeyLastUsedReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->getaccesskeylastused(
      iv_accesskeyid = iv_access_key_id ).
    MESSAGE 'Retrieved access key last used information.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Access key does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetAccessKeyLastUsedreferência da API AWS SDK for SAP ABAP.

Use **`GetAccountAuthorizationDetails`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetAccountAuthorizationDetails`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar sua conta

CLI

AWS CLI

Para listar os usuários, grupos, funções e políticas do IAM de uma AWS conta

O `get-account-authorization-details` comando a seguir retorna informações sobre todos os usuários, grupos, funções e políticas do IAM na AWS conta.

```
aws iam get-account-authorization-details
```

Saída:

```
{
    "RoleDetailList": [
        {
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "RoleId": "AROA1234567890EXAMPLE",
            "CreateDate": "2014-07-30T17:09:20Z",
            "InstanceProfileList": [
                {
                    "InstanceProfileId": "AIPA1234567890EXAMPLE",
                    "Roles": [
                        {
                            "AssumeRolePolicyDocument": {
                                "Version":"2012-10-17",
                                "Statement": [
                                    {
                                        "Sid": "",
                                        "Effect": "Allow",
                                        "Principal": {
                                            "Service": "ec2.amazonaws.com"
                                        },
                                        "Action": "sts:AssumeRole"
                                    }
                                ]
                            },
                            "RoleId": "AROA1234567890EXAMPLE",
                            "CreateDate": "2014-07-30T17:09:20Z",
                            "RoleName": "EC2role",
```

```
                            "Path": "/",
                            "Arn": "arn:aws:iam::123456789012:role/EC2role"
                        }
                    ],
                    "CreateDate": "2014-07-30T17:09:20Z",
                    "InstanceProfileName": "EC2role",
                    "Path": "/",
                    "Arn": "arn:aws:iam::123456789012:instance-profile/EC2role"
                }
            ],
            "RoleName": "EC2role",
            "Path": "/",
            "AttachedManagedPolicies": [
                {
                    "PolicyName": "AmazonS3FullAccess",
                    "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
                },
                {
                    "PolicyName": "AmazonDynamoDBFullAccess",
                    "PolicyArn": "arn:aws:iam::aws:policy/
AmazonDynamoDBFullAccess"
                }
            ],
            "RoleLastUsed": {
                "Region": "us-west-2",
                "LastUsedDate": "2019-11-13T17:30:00Z"
            },
            "RolePolicyList": [],
            "Arn": "arn:aws:iam::123456789012:role/EC2role"
        }
    ],
    "GroupDetailList": [
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": {
                "PolicyName": "AdministratorAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
            },
            "GroupName": "Admins",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "CreateDate": "2013-10-14T18:32:24Z",
            "GroupPolicyList": []
        },
```

```
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": {
                "PolicyName": "PowerUserAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
            },
            "GroupName": "Dev",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Dev",
            "CreateDate": "2013-10-14T18:33:55Z",
            "GroupPolicyList": []
        },
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": [],
            "GroupName": "Finance",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Finance",
            "CreateDate": "2013-10-14T18:57:48Z",
            "GroupPolicyList": [
                {
                    "PolicyName": "policygen-201310141157",
                    "PolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Action": "aws-portal:*",
                                "Sid": "Stmt1381777017000",
                                "Resource": "*",
                                "Effect": "Allow"
                            }
                        ]
                    }
                }
            ]
        }
    ],
    "UserDetailList": [
        {
            "UserName": "Alice",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:24Z",
```

```
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Alice"
        },
        {
            "UserName": "Bob",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:25Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [
                {
                    "PolicyName": "DenyBillingAndIAMPolicy",
                    "PolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": {
                            "Effect": "Deny",
                            "Action": [
                                "aws-portal:*",
                                "iam:*"
                            ],
                            "Resource": "*"
                        }
                    }
                }
            ],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        },
        {
            "UserName": "Charlie",
            "GroupList": [
                "Dev"
            ],
            "CreateDate": "2013-10-14T18:33:56Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Charlie"
```

```
        }
    ],
    "Policies": [
        {
            "PolicyName": "create-update-delete-set-managed-policies",
            "CreateDate": "2015-02-06T19:58:34Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-02-06T19:58:34Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
                        "Statement": {
                            "Effect": "Allow",
                            "Action": [
                                "iam:CreatePolicy",
                                "iam:CreatePolicyVersion",
                                "iam:DeletePolicy",
                                "iam:DeletePolicyVersion",
                                "iam:GetPolicy",
                                "iam:GetPolicyVersion",
                                "iam:ListPolicies",
                                "iam:ListPolicyVersions",
                                "iam:SetDefaultPolicyVersion"
                            ],
                            "Resource": "*"
                        }
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/create-update-delete-set-
managed-policies",
            "UpdateDate": "2015-02-06T19:58:34Z"
        },
        {
            "PolicyName": "S3-read-only-specific-bucket",
            "CreateDate": "2015-01-21T21:39:41Z",
            "AttachmentCount": 1,
```

```
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-01-21T21:39:41Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "s3:Get*",
                                    "s3:List*"
                                ],
                                "Resource": [
                                    "arn:aws:s3:::amzn-s3-demo-bucket",
                                    "arn:aws:s3:::amzn-s3-demo-bucket/*"
                                ]
                            }
                        ]
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/S3-read-only-specific-
bucket",
            "UpdateDate": "2015-01-21T23:39:41Z"
        },
        {
            "PolicyName": "AmazonEC2FullAccess",
            "CreateDate": "2015-02-06T18:40:15Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2014-10-30T20:59:46Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
```

```
                        "Statement": [
                            {
                                "Action": "ec2:*",
                                "Effect": "Allow",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "elasticloadbalancing:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "cloudwatch:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "autoscaling:*",
                                "Resource": "*"
                            }
                        ]
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess",
            "UpdateDate": "2015-02-06T18:40:15Z"
        }
    ],
    "Marker": "EXAMPLEkakv9BCuUNFDtxWSyfzetYwEx2ADc8dnzfvERF5S6YMvXKx41t6gCl/
eeaCX3Jo94/bKqezEAg8TEVS99EKFLxm3jtbpl25FDWEXAMPLE",
    "IsTruncated": true
}
```

Para obter mais informações, consulte Diretrizes de auditoria de segurança da AWS no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetAccountAuthorizationDetailsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo obtém detalhes de autorização sobre as identidades na AWS conta e exibe a lista de elementos do objeto retornado, incluindo usuários, grupos e funções. Por exemplo, a propriedade **`UserDetailList`** exibe detalhes sobre os usuários. Informações semelhantes estão disponíveis nas propriedades **`RoleDetailList`** e **`GroupDetailList`**.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Saída:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Saída:

```
Arn            : arn:aws:iam::123456789012:user/Administrator
CreateDate     : 10/16/2014 9:03:09 AM
GroupList      : {Administrators}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE1
UserName       : Administrator
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/Bob
CreateDate     : 4/6/2015 12:54:42 PM
GroupList      : {Developers}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE2
UserName       : bab
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/BackupToS3
```

```
CreateDate     : 1/27/2015 10:15:08 AM
GroupList      : {Backup}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE3
UserName       : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- Para obter detalhes da API, consulte GetAccountAuthorizationDetailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém detalhes de autorização sobre as identidades na AWS conta e exibe a lista de elementos do objeto retornado, incluindo usuários, grupos e funções. Por exemplo, a propriedade **`UserDetailList`** exibe detalhes sobre os usuários. Informações semelhantes estão disponíveis nas propriedades **`RoleDetailList`** e **`GroupDetailList`**.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Saída:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Saída:

```
Arn            : arn:aws:iam::123456789012:user/Administrator
CreateDate     : 10/16/2014 9:03:09 AM
GroupList      : {Administrators}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE1
UserName       : Administrator
UserPolicyList : {}
```

```
Arn             : arn:aws:iam::123456789012:user/Bob
CreateDate      : 4/6/2015 12:54:42 PM
GroupList       : {Developers}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE2
UserName        : bab
UserPolicyList : {}

Arn             : arn:aws:iam::123456789012:user/BackupToS3
CreateDate      : 1/27/2015 10:15:08 AM
GroupList       : {Backup}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE3
UserName        : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- Para obter detalhes da API, consulte GetAccountAuthorizationDetailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
```

```
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- Para obter detalhes da API, consulte a GetAccountAuthorizationDetailsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_iam->getaccountauthdetails( ).
       MESSAGE 'Retrieved account authorization details.' TYPE 'I'.
     CATCH /aws1/cx_iamservicefailureex.
       MESSAGE 'Service failure when getting account authorization details.'
TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetAccountAuthorizationDetailsreferência da API AWS SDK for SAP ABAP.

Use **`GetAccountPasswordPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetAccountPasswordPolicy`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como visualizar a política de senha da conta atual

O comando `get-account-password-policy`, apresentado a seguir, exibe detalhes sobre a política de senha para a conta atual.

```
aws iam get-account-password-policy
```

Saída:

```
{
```

```
    "PasswordPolicy": {
        "AllowUsersToChangePassword": false,
        "RequireLowercaseCharacters": false,
        "RequireUppercaseCharacters": false,
        "MinimumPasswordLength": 8,
        "RequireNumbers": true,
        "RequireSymbols": true
    }
}
```

Se nenhuma política de senha estiver definida para a conta, o comando retornará um erro `NoSuchEntity`.

Para obter mais informações, consulte Definição de uma política de senhas de contas para usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetAccountPasswordPolicyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
```

```
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicya Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política de senha da conta.

```
import {
```

```
  GetAccountPasswordPolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const getAccountPasswordPolicy = async () => {
  const command = new GetAccountPasswordPolicyCommand({});

  const response = await client.send(command);
  console.log(response.PasswordPolicy);
  return response;
};
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicya Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicya Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna detalhes sobre a política de senha da conta atual. Se nenhuma política de senha estiver definida na conta, o comando retorna um erro **`NoSuchEntity`**.

```
Get-IAMAccountPasswordPolicy
```

Saída:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre a política de senha da conta atual. Se nenhuma política de senha estiver definida na conta, o comando retorna um erro **`NoSuchEntity`**.

```
Get-IAMAccountPasswordPolicy
```

Saída:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
```

```
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(
            f"\tallow_users_to_change_password:
 {pw_policy.allow_users_to_change_password}"
        )
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
        print(f"\tpassword_reuse_prevention:
 {pw_policy.password_reuse_prevention}")
        print(
            f"\trequire_lowercase_characters:
 {pw_policy.require_lowercase_characters}"
        )
```

```
        print(f"\trequire_numbers: {pw_policy.require_numbers}")
        print(f"\trequire_symbols: {pw_policy.require_symbols}")
        print(
            f"\trequire_uppercase_characters:
 {pw_policy.require_uppercase_characters}"
        )
        printed = True
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchEntity":
            print("The account does not have a password policy set.")
        else:
            logger.exception("Couldn't get account password policy.")
            raise
    else:
        return printed
```

- Para obter detalhes da API, consulte a GetAccountPasswordPolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'IAMPolicyManager'
  end
```

```
  # Retrieves and logs the account password policy
  def print_account_password_policy
    response = @iam_client.get_account_password_policy
    @logger.info("The account password policy is:
 #{response.password_policy.to_h}")
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.info('The account does not have a password policy.')
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't print the account password policy. Error: #{e.code} -
 #{e.message}")
    raise
  end
end
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn get_account_password_policy(
    client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput,
 SdkError<GetAccountPasswordPolicyError>> {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a GetAccountPasswordPolicyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->getaccountpasswordpolicy( ).
    MESSAGE 'Retrieved account password policy.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'No password policy exists.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when getting password policy.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetAccountPasswordPolicyreferência da API AWS SDK for SAP ABAP.

Use **`GetAccountSummary`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetAccountSummary`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar sua conta

CLI

AWS CLI

Como obter informações sobre o uso da entidade do IAM e das cotas do IAM na conta atual

O comando `get-account-summary`, apresentado a seguir, retorna informações sobre o uso atual da entidade do IAM e das cotas atuais da entidade do IAM na conta.

```
aws iam get-account-summary
```

Saída:

```
{
    "SummaryMap": {
        "UsersQuota": 5000,
        "GroupsQuota": 100,
        "InstanceProfiles": 6,
        "SigningCertificatesPerUserQuota": 2,
        "AccountAccessKeysPresent": 0,
        "RolesQuota": 250,
        "RolePolicySizeQuota": 10240,
        "AccountSigningCertificatesPresent": 0,
        "Users": 27,
        "ServerCertificatesQuota": 20,
        "ServerCertificates": 0,
        "AssumeRolePolicySizeQuota": 2048,
        "Groups": 7,
        "MFADevicesInUse": 1,
        "Roles": 3,
        "AccountMFAEnabled": 1,
        "MFADevices": 3,
        "GroupsPerUserQuota": 10,
        "GroupPolicySizeQuota": 5120,
        "InstanceProfilesQuota": 100,
        "AccessKeysPerUserQuota": 2,
        "Providers": 0,
        "UserPolicySizeQuota": 2048
    }
}
```

Para obter mais informações sobre limitações de entidades, consulte as cotas do IAM e do AWS STS no Guia AWS do usuário do IAM.

- Para obter detalhes da API, consulte GetAccountSummaryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna informações sobre o uso atual da entidade do IAM e das cotas atuais da entidade do IAM na Conta da AWS.

```
Get-IAMAccountSummary
```

Saída:

```
Key                                     Value
Users                                   7
GroupPolicySizeQuota                    5120
PolicyVersionsInUseQuota                10000
ServerCertificatesQuota                 20
AccountSigningCertificatesPresent       0
AccountAccessKeysPresent                0
Groups                                  3
UsersQuota                              5000
RolePolicySizeQuota                     10240
UserPolicySizeQuota                     2048
GroupsPerUserQuota                      10
AssumeRolePolicySizeQuota               2048
AttachedPoliciesPerGroupQuota           2
Roles                                   9
VersionsPerPolicyQuota                  5
GroupsQuota                             100
PolicySizeQuota                         5120
Policies                                5
RolesQuota                              250
ServerCertificates                      0
AttachedPoliciesPerRoleQuota            2
MFADevicesInUse                         2
PoliciesQuota                           1000
AccountMFAEnabled                       1
Providers                               2
InstanceProfilesQuota                   100
MFADevices                              4
AccessKeysPerUserQuota                  2
AttachedPoliciesPerUserQuota            2
SigningCertificatesPerUserQuota         2
```

```
PolicyVersionsInUse                         4
InstanceProfiles                            1
...
```

- Para obter detalhes da API, consulte GetAccountSummaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna informações sobre o uso atual da entidade do IAM e das cotas atuais da entidade do IAM na Conta da AWS.

```
Get-IAMAccountSummary
```

Saída:

```


Key                                         Value
Users                                       7
GroupPolicySizeQuota                        5120
PolicyVersionsInUseQuota                    10000
ServerCertificatesQuota                     20
AccountSigningCertificatesPresent           0
AccountAccessKeysPresent                    0
Groups                                      3
UsersQuota                                  5000
RolePolicySizeQuota                         10240
UserPolicySizeQuota                         2048
GroupsPerUserQuota                          10
AssumeRolePolicySizeQuota                   2048
AttachedPoliciesPerGroupQuota               2
Roles                                       9
VersionsPerPolicyQuota                      5
GroupsQuota                                 100
PolicySizeQuota                             5120
Policies                                    5
RolesQuota                                  250
ServerCertificates                          0
AttachedPoliciesPerRoleQuota                2
MFADevicesInUse                             2
PoliciesQuota                               1000
AccountMFAEnabled                           1
Providers                                   2
```

```
InstanceProfilesQuota                      100
MFADevices                                 4
AccessKeysPerUserQuota                     2
AttachedPoliciesPerUserQuota               2
SigningCertificatesPerUserQuota            2
PolicyVersionsInUse                        4
InstanceProfiles                           1
...
```

- Para obter detalhes da API, consulte GetAccountSummaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- Para obter detalhes da API, consulte a GetAccountSummaryReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->getaccountsummary( ).
    MESSAGE 'Retrieved account summary.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when getting account summary.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetAccountSummaryreferência da API AWS SDK for SAP ABAP.

Usar **`GetContextKeysForCustomPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetContextKeysForCustomPolicy`.

CLI

AWS CLI

Exemplo 1: para listar as chaves de contexto referenciadas por uma ou mais políticas JSON personalizadas fornecidas como um parâmetro na linha de comando

O comando `get-context-keys-for-custom-policy` a seguir analisa cada política fornecida e lista as chaves de contexto usadas por essas políticas. Use esse comando para identificar quais valores de chave de contexto você deve fornecer para usar com êxito os comandos do simulador de políticas `simulate-custom-policy` e `simulate-custom-`

`policy`. Você também pode recuperar a lista de chaves de contexto utilizadas por todas as políticas associadas a um perfil ou usuário do IAM com o comando `get-context-keys-for-custom-policy`. Os valores de parâmetro que começam com `file://` instruem o comando a ler o arquivo e usar o conteúdo como o valor do parâmetro em vez do próprio nome do arquivo.

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/${aws:username}","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

Saída:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

Exemplo 2: para listar as chaves de contexto referenciadas por uma ou mais políticas JSON personalizadas fornecidas como entrada de arquivo

O comando `get-context-keys-for-custom-policy` a seguir é igual ao exemplo anterior, exceto que as políticas são fornecidas em um arquivo e não como um parâmetro. Como o comando espera uma lista JSON de strings e não uma lista de estruturas JSON, o arquivo deve ser estruturado da forma a seguir, embora você possa reduzi-lo em uma só.

```
[
    "Policy1",
    "Policy2"
]
```

Assim, por exemplo, um arquivo que contém a política do exemplo anterior deve ter a aparência a seguir. Você deve escapar cada aspas duplas incorporadas dentro da string de política precedendo-as com uma barra invertida ".

```
[ "{\"Version\": \"2012-10-17\", \"Statement\": {\"Effect\": \"Allow
\", \"Action\": \"dynamodb:*\", \"Resource\": \"arn:aws:dynamodb:us-
```

```
west-2:128716708097:table/${aws:username}\", \"Condition\": {\"DateGreaterThan\":
 {\"aws:CurrentTime\": \"2015-08-16T12:00:00Z\"}}}}" ]
```

Esse arquivo pode então ser enviado ao comando a seguir.

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list file://policyfile.json
```

Saída:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

Para obter mais informações, consulte Como usar o IAM Policy Simulator (AWS CLI AWS e API) no Guia AWS do usuário do IAM.

- Para obter detalhes da API, consulte GetContextKeysForCustomPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo busca todas as chaves de contexto presentes no JSON da política fornecida. A fim de produzir várias políticas, você pode fornecer uma lista de valores separados por vírgula.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- Para obter detalhes da API, consulte GetContextKeysForCustomPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo busca todas as chaves de contexto presentes no JSON da política fornecida. A fim de produzir várias políticas, você pode fornecer uma lista de valores separados por vírgula.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- Para obter detalhes da API, consulte GetContextKeysForCustomPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetContextKeysForPrincipalPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetContextKeysForPrincipalPolicy`.

CLI

AWS CLI

Para listar as chaves de contexto referenciadas por todas as políticas associadas a uma entidade principal do IAM

O comando `get-context-keys-for-principal-policy` a seguir recupera todas as políticas anexadas à usuária `saanvi` e aos grupos dos quais ela é membro. Em seguida, ele analisa cada uma delas e lista as chaves de contexto usadas por essas políticas. Utilize esse comando para identificar quais valores de chave de contexto você deve fornecer para usar com êxito os comandos `simulate-custom-policy` e `simulate-principal-policy`. Você também pode recuperar a lista de chaves de contexto usadas por uma política JSON arbitrária com o comando `get-context-keys-for-custom-policy`.

```
aws iam get-context-keys-for-principal-policy \
```

```
    --policy-source-arn arn:aws:iam::123456789012:user/saanvi
```

Saída:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

Para obter mais informações, consulte Como usar o IAM Policy Simulator (AWS CLI AWS e API) no Guia AWS do usuário do IAM.

- Para obter detalhes da API, consulte GetContextKeysForPrincipalPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo busca todas as chaves de contexto presentes no json de política fornecido e nas políticas anexadas à entidade do IAM (user/role etc.). Para - PolicyInputList você pode fornecer uma lista de vários valores como valores separados por vírgula.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- Para obter detalhes da API, consulte GetContextKeysForPrincipalPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo busca todas as chaves de contexto presentes no json de política fornecido e nas políticas anexadas à entidade do IAM (user/role etc.). Para - PolicyInputList você pode fornecer uma lista de vários valores como valores separados por vírgula.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- Para obter detalhes da API, consulte GetContextKeysForPrincipalPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetCredentialReport`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetCredentialReport`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar sua conta

CLI

AWS CLI

Como obter um relatório de credenciais

Este exemplo abre o relatório retornado e o envia ao pipeline como uma matriz de linhas de texto.

```
aws iam get-credential-report
```

Saída:

```
{
    "GeneratedTime":  "2015-06-17T19:11:50Z",
    "ReportFormat": "text/csv"
}
```

Para obter mais informações, consulte Obter relatórios de credenciais para sua AWS conta no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetCredentialReportem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo1: este exemplo abre o relatório retornado e o envia ao pipeline como uma matriz de linhas de texto. A primeira linha é o cabeçalho com nomes de colunas separados por vírgula. Cada linha sucessiva é a linha de detalhes de um usuário, com cada campo separado por vírgulas. Antes de visualizar o relatório, você deve gerá-lo com o cmdlet **`Request-IAMCredentialReport`**. Para recuperar o relatório como uma única string, use **`-Raw`** em vez de **`-AsTextArray`**. O alias **`-SplitLines`** também é aceito no switch **`-AsTextArray`**. Para obter a lista completa de colunas na saída, consulte a referência de API do serviço. Observe que, se não usar **`-AsTextArray`** ou **`-SplitLines`**, você deve extrair o texto da propriedade **`.Content`** usando a classe **`StreamReader`** .NET.

```
Request-IAMCredentialReport
```

Saída:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Saída:

```
user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,pa
```

```
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/
A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- Para obter detalhes da API, consulte GetCredentialReportem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo1: este exemplo abre o relatório retornado e o envia ao pipeline como uma matriz de linhas de texto. A primeira linha é o cabeçalho com nomes de colunas separados por vírgula. Cada linha sucessiva é a linha de detalhes de um usuário, com cada campo separado por vírgulas. Antes de visualizar o relatório, você deve gerá-lo com o cmdlet **`Request-IAMCredentialReport`**. Para recuperar o relatório como uma única string, use **`-Raw`** em vez de **`-AsTextArray`**. O alias **`-SplitLines`** também é aceito no switch **`-AsTextArray`**. Para obter a lista completa de colunas na saída, consulte a referência de API do serviço. Observe que, se não usar **`-AsTextArray`** ou **`-SplitLines`**, você deve extrair o texto da propriedade **`.Content`** usando a classe **`StreamReader`** .NET.

```
Request-IAMCredentialReport
```

Saída:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Saída:

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,pa
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015
A,false,N/A,false,N/A,false,N/A
```

```
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/
A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- Para obter detalhes da API, consulte GetCredentialReportem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]
```

- Para obter detalhes da API, consulte a GetCredentialReportReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->getcredentialreport( ).
    MESSAGE 'Retrieved credential report.' TYPE 'I'.
  CATCH /aws1/cx_iamcredrptnotpresen00.
    MESSAGE 'Credential report not present.' TYPE 'E'.
  CATCH /aws1/cx_iamcredrptexpiredex.
    MESSAGE 'Credential report expired.' TYPE 'E'.
  CATCH /aws1/cx_iamcredrptnotreadyex.
    MESSAGE 'Credential report not ready.' TYPE 'E'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when getting credential report.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetCredentialReportreferência da API AWS SDK for SAP ABAP.

Usar **`GetGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetGroup`.

CLI

AWS CLI

Obter um grupo do IAM

Este exemplo retorna detalhes sobre o grupo do IAM `Admins`.

```
aws iam get-group \
```

```
    --group-name Admins
```

Saída:

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-06-16T19:41:48Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    },
    "Users": []
}
```

Para obter mais informações, consulte Identidades do IAM (usuários, grupos de usuários e perfis) no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna detalhes sobre o grupo do IAM **`Testers`**, incluindo uma compilação de todos os usuários do IAM que pertencem ao grupo.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Saída:

```
Group                                     IsTruncated           Marker
     Users
-----                                     -----------           ------
     -----
Amazon.IdentityManagement.Model.Group     False
     {Theresa, David}
```

```
$results.Group
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- Para obter detalhes da API, consulte GetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre o grupo do IAM **`Testers`**, incluindo uma compilação de todos os usuários do IAM que pertencem ao grupo.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Saída:

```
Group                                     IsTruncated           Marker
     Users
```

```
-----                                    -----------          ------
     -----
Amazon.IdentityManagement.Model.Group     False
     {Theresa, David}
```

```
$results.Group
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- Para obter detalhes da API, consulte GetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetGroupPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetGroupPolicy`.

CLI

AWS CLI

Obter informações sobre uma política anexada a um grupo do IAM

O comando `get-group-policy` a seguir obtém informações sobre a política especificada anexada ao grupo denominado `Test-Group`.

```
aws iam get-group-policy \
    --group-name Test-Group \
    --policy-name S3-ReadOnly-Policy
```

Saída:

```
{
    "GroupName": "Test-Group",
    "PolicyDocument": {
        "Statement": [
            {
                "Action": [
                    "s3:Get*",
                    "s3:List*"
                ],
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    },
    "PolicyName": "S3-ReadOnly-Policy"
}
```

Para obter mais informações, consulte Gerenciamento de políticas do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetGroupPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna detalhes sobre a política em linha incorporada denominada **`PowerUserAccess-Testers`** do grupo **`Testers`**. A propriedade **`PolicyDocument`** é codificada em URL. Ela é decodificada neste exemplo com o método .NET **`UrlDecode`**.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-
Testers
$results
```

Saída:

```
GroupName     PolicyDocument
 PolicyName
---------     --------------
 ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- Para obter detalhes da API, consulte GetGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre a política em linha incorporada denominada **`PowerUserAccess-Testers`** do grupo **`Testers`**. A propriedade **`PolicyDocument`** é codificada em URL. Ela é decodificada neste exemplo com o método .NET **`UrlDecode`**.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-
Testers
$results
```

Saída:

```
GroupName     PolicyDocument
 PolicyName
---------     --------------
 ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- Para obter detalhes da API, consulte GetGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetInstanceProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetInstanceProfile`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de streaming gerenciado

CLI

AWS CLI

Obter informações sobre um perfil de instância

O comando `get-instance-profile` a seguir obtém informações sobre o perfil de instância denominado `ExampleInstanceProfile`.

```
aws iam get-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

Saída:

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AID2MAB8DPLSRHEXAMPLE",
        "Roles": [
            {
                "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                "RoleId": "AIDGPMS9RO4H3FEXAMPLE",
                "CreateDate": "2013-01-09T06:33:26Z",
                "RoleName": "Test-Role",
                "Path": "/",
                "Arn": "arn:aws:iam::336924118301:role/Test-Role"
            }
        ],
        "CreateDate": "2013-06-12T23:52:02Z",
        "InstanceProfileName": "ExampleInstanceProfile",
        "Path": "/",
        "Arn": "arn:aws:iam::336924118301:instance-profile/
ExampleInstanceProfile"
    }
}
```

Para obter mais informações, consulte Usar perfis de instância no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetInstanceProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo retorna detalhes do nome **`ec2instancerole`** do perfil da instância definido na AWS conta atual.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Para obter detalhes da API, consulte GetInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna detalhes do nome **`ec2instancerole`** do perfil da instância definido na AWS conta atual.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Para obter detalhes da API, consulte GetInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetLoginProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetLoginProfile`.

CLI

AWS CLI

Obter informações de senha de um usuário do IAM

O comando `get-login-profile` a seguir obtém informações sobre a senha do usuário do IAM chamado `Bob`.

```
aws iam get-login-profile \
    --user-name Bob
```

Saída:

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2012-09-21T23:03:39Z"
    }
}
```

O comando `get-login-profile` pode ser usado para verificar se um usuário do IAM tem uma senha. O comando retorna um erro `NoSuchEntity` se nenhuma senha for definida para o usuário.

Não é possível visualizar uma senha com esse comando. Se a senha for perdida, você pode redefini-la (`update-login-profile`) para o usuário. Como alternativa, você pode excluir o perfil de login (`delete-login-profile`) do usuário e criar um novo (`create-login-profile`).

Para obter mais informações, consulte Gerenciamento de senhas de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetLoginProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna a data de criação da senha e se uma redefinição de senha é necessária para o usuário do IAM **`David`**.

```
Get-IAMLoginProfile -UserName David
```

Saída:

```
CreateDate                  PasswordResetRequired                UserName
----------                  ---------------------                --------
12/10/2014 3:39:44 PM       False                                David
```

- Para obter detalhes da API, consulte GetLoginProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a data de criação da senha e se uma redefinição de senha é necessária para o usuário do IAM **`David`**.

```
Get-IAMLoginProfile -UserName David
```

Saída:

```
CreateDate                  PasswordResetRequired                UserName
----------                  ---------------------                --------
12/10/2014 3:39:44 PM       False                                David
```

- Para obter detalhes da API, consulte GetLoginProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetOpenIdConnectProvider`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetOpenIdConnectProvider`.

CLI

AWS CLI

Retornar informações sobre o provedor OpenID Connect especificado

Este exemplo retorna detalhes sobre o provedor OpenID Connect cujo ARN é `arn:aws:iam::123456789012:oidc-provider/server.example.com`.

```
aws iam get-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

Saída:

```
{
    "Url": "server.example.com"
        "CreateDate": "2015-06-16T19:41:48Z",
        "ThumbprintList": [
        "12345abcdefghijk67890lmnopqrst987example"
        ],
        "ClientIDList": [
        "example-application-ID"
        ]
}
```

Para obter mais informações, consulte Criar provedores de identidade OpenID Connect (OIDC) no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetOpenIdConnectProviderem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna detalhes sobre o provedor OpenID Connect cujo ARN é **`arn:aws:iam::123456789012:oidc-provider/accounts.google.com`**. A propriedade **`ClientIDList`** é uma compilação que contém todos os IDs de cliente definidos para esse provedor.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Saída:

```
ClientIDList         CreateDate                ThumbprintList
           Url
------------         ----------                --------------
           ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- Para obter detalhes da API, consulte GetOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre o provedor OpenID Connect cujo ARN é **`arn:aws:iam::123456789012:oidc-provider/accounts.google.com`**. A propriedade **`ClientIDList`** é uma compilação que contém todos os IDs de cliente definidos para esse provedor.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Saída:

```
ClientIDList         CreateDate                ThumbprintList
           Url
------------         ----------                --------------
           ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- Para obter detalhes da API, consulte GetOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos de streaming gerenciado
- Trabalhar com a API IAM Policy Builder

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }
```

- Para obter detalhes da API, consulte GetPolicya Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                            const Aws::Client::ClientConfiguration &clientConfig)
 {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
                  "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
                  policy.GetArn() << std::endl << "Description: " <<
                  policy.GetDescription() << std::endl << "CreateDate: " <<

 policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetPolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para recuperar informações sobre a política gerenciada especificada

Este exemplo retorna detalhes sobre a política gerenciada cujo ARN é `arn:aws:iam::123456789012:policy/MySamplePolicy`.

```
aws iam get-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Saída:

```
{
    "Policy": {
        "PolicyName": "MySamplePolicy",
        "CreateDate": "2015-06-17T19:23;32Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "Z27SI6FQMGNQ2EXAMPLE1",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:policy/MySamplePolicy",
        "UpdateDate": "2015-06-17T19:23:32Z"
    }
}
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetPolicyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- Para obter detalhes da API, consulte GetPolicya Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política.

```
import { GetPolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const getPolicy = (policyArn) => {
  const command = new GetPolicyCommand({
    PolicyArn: policyArn,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetPolicya Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  PolicyArn: "arn:aws:iam::aws:policy/AWSLambdaExecute",
};

iam.getPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetPolicya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
```

```
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- Para obter detalhes da API, consulte a GetPolicyreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- Para obter detalhes da API, consulte GetPolicya Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna detalhes sobre a política gerenciada cujo ARN é **`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Saída:

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- Para obter detalhes da API, consulte GetPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre a política gerenciada cujo ARN é **`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Saída:

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- Para obter detalhes da API, consulte GetPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- Para obter detalhes da API, consulte a GetPolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Fetches an IAM policy by its ARN
# @param policy_arn [String] the ARN of the IAM policy to retrieve
# @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
def get_policy(policy_arn)
  response = @iam_client.get_policy(policy_arn: policy_arn)
  policy = response.policy
  @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
  policy
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
  raise
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
  raise
end
```

- Para obter detalhes da API, consulte GetPolicya Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->getpolicy( iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetPolicyreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func getPolicy(arn: String) async throws -> IAMClientTypes.Policy {
        let input = GetPolicyInput(
            policyArn: arn
        )
        do {
            let output = try await client.getPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            print("ERROR: getPolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte GetPolicya referência da API AWS SDK for Swift.

Use **`GetPolicyVersion`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetPolicyVersion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Políticas gerenciadas
- Trabalhar com a API IAM Policy Builder

CLI

AWS CLI

Para recuperar informações sobre a versão especificada da política gerenciada especificada

Este exemplo retorna o documento da política para a versão v2 da política cujo ARN é `arn:aws:iam::123456789012:policy/MyManagedPolicy`.

```
aws iam get-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

Saída:

```
{
    "PolicyVersion": {
        "Document": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "iam:*",
                    "Resource": "*"
                }
            ]
```

```
        },
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2023-04-11T00:22:54+00:00"
    }
}
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetPolicyVersionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna o documento da política na versão **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MyManagedPolicy**. O documento da política na propriedade **Document** é codificado em URL, sendo decodificado neste exemplo com o método .NET **UrlDecode**.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Saída:

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
           v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version":"2012-10-17",
  "Statement":
```

```
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- Para obter detalhes da API, consulte GetPolicyVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o documento da política na versão **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MyManagedPolicy**. O documento da política na propriedade **Document** é codificado em URL, sendo decodificado neste exemplo com o método .NET **UrlDecode**.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Saída:

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
           v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version":"2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
```

```
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- Para obter detalhes da API, consulte GetPolicyVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
```

```
        return policy_statement
```

- Para obter detalhes da API, consulte a GetPolicyVersionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->getpolicyversion(
      iv_policyarn = iv_policy_arn
      iv_versionid = iv_version_id ).
    MESSAGE 'Retrieved policy version information.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy or version does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iaminvalidinputex.
    MESSAGE 'Invalid input provided.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetPolicyVersionreferência da API AWS SDK for SAP ABAP.

Use **`GetRole`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetRole`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Conceitos básicos de data warehouses de tecnologia sem servidor
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de arquivos de atributos de machine learning
- Conceitos básicos de clusters gerenciados de kubernetes
- Conceitos básicos de streaming gerenciado
- Conceitos básicos de clusters de data warehouse provisionados
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get information about an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to retrieve information
/// for.</param>
/// <returns>The IAM role that was retrieved.</returns>
public async Task<Role> GetRoleAsync(string roleName)
{
    var response = await _IAMService.GetRoleAsync(new GetRoleRequest
    {
        RoleName = roleName,
    });

    return response.Role;
}
```

- Para obter detalhes da API, consulte GetRolea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como obter informações sobre um perfil do IAM

O comando `get-role`, apresentado a seguir, obtém informações sobre o perfil denominado `Test-Role`.

```
aws iam get-role \
    --role-name Test-Role
```

Saída:

```
{
    "Role": {
        "Description": "Test Role",
        "AssumeRolePolicyDocument":"<URL-encoded-JSON>",
        "MaxSessionDuration": 3600,
        "RoleId": "AROA1234567890EXAMPLE",
        "CreateDate": "2019-11-13T16:45:56Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "RoleLastUsed": {
            "Region": "us-east-1",
            "LastUsedDate": "2019-11-13T17:14:00Z"
        },
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

O comando exibe a política de confiança anexada ao perfil. Para listar as políticas de permissões anexadas a um perfil, use o comando `list-role-policies`.

Para obter mais informações, consulte Criar perfis do IAM no AWS Guia do usuário do IAM.

- Para obter detalhes da API, consulte GetRoleem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
```

```
}
```

- Para obter detalhes da API, consulte GetRolea Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a função.

```
import { GetRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const getRole = (roleName) => {
  const command = new GetRoleCommand({
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetRolea Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- Para obter detalhes da API, consulte GetRolea Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna os detalhes do **`lamda_exec_role`**. Ele inclui o documento da política de confiança que especifica quem pode assumir esse perfil. O documento da política é codificado em URL e pode ser decodificado usando o método .NET **`UrlDecode`**. Neste exemplo, todos os espaços em branco da política original foram removidos antes de ela ser carregada na política. Para ver os documentos da política de permissões que determinam o que alguém que assume o perfil pode fazer, use **`Get-IAMRolePolicy`** para políticas em linha e **`Get-IAMPolicyVersion`** para políticas gerenciadas anexadas.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Saída:

```
Arn                       : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement
%22%3A%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate                : 4/2/2015 9:16:11 AM
Path                      : /
RoleId                    : 2YBIKAIBHNKB4EXAMPLE1
RoleName                  : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Saída:

```
{"Version":"2012-10-17",       "Statement":
[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Para obter detalhes da API, consulte GetRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna os detalhes do **`lamda_exec_role`**. Ele inclui o documento da política de confiança que especifica quem pode assumir esse perfil. O documento da política é codificado em URL e pode ser decodificado usando o método .NET **`UrlDecode`**. Neste exemplo, todos os espaços em branco da política original foram removidos antes de ela ser carregada na política. Para ver os documentos da política de permissões que determinam o que alguém que assume o perfil pode fazer, use **`Get-IAMRolePolicy`** para políticas em linha e **`Get-IAMPolicyVersion`** para políticas gerenciadas anexadas.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Saída:

```
Arn                       : arn:aws:iam::123456789012:role/lambda_exec_role
```

```
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement
%22%3A%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate               : 4/2/2015 9:16:11 AM
Path                     : /
RoleId                   : 2YBIKAIBHNKB4EXAMPLE1
RoleName                 : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Saída:

```
{"Version":"2012-10-17",       "Statement":
[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Para obter detalhes da API, consulte GetRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
```

```
    """
    try:
        role = iam.Role(role_name)
        role.load()  # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- Para obter detalhes da API, consulte a GetRoleReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                              role_name: name
                            }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
```

```
    role
  end
```

- Para obter detalhes da API, consulte GetRolea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- Para obter detalhes da API, consulte a GetRolereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        oo_result = lo_iam->getrole( iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved role information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetRolereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func getRole(name: String) async throws -> IAMClientTypes.Role {
        let input = GetRoleInput(
            roleName: name
        )
        do {
            let output = try await client.getRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            print("ERROR: getRole:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte GetRolea referência da API AWS SDK for Swift.

Usar **`GetRolePolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetRolePolicy`.

CLI

AWS CLI

Obter informações sobre uma política anexada a um perfil do IAM

O comando `get-role-policy` a seguir obtém informações sobre a política especificada anexada ao perfil denominado `Test-Role`.

```
aws iam get-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

Saída:

```
{
  "RoleName": "Test-Role",
  "PolicyDocument": {
      "Statement": [
          {
              "Action": [
                  "s3:ListBucket",
                  "s3:Put*",
                  "s3:Get*",
                  "s3:*MultipartUpload*"
              ],
              "Resource": "*",
              "Effect": "Allow",
              "Sid": "1"
          }
      ]
  }
  "PolicyName": "ExamplePolicy"
}
```

Para obter mais informações, consulte Criação de perfis do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetRolePolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna o documento da política de permissões da política denominada **`oneClick_lambda_exec_role_policy`** incorporada no perfil do IAM **`lamda_exec_role`**. O documento resultante da política é codificado em URL. Ela é decodificada neste exemplo com o método .NET **`UrlDecode`**.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Saída:

```
PolicyDocument                                           PolicyName
              UserName
--------------                                           ----------
              --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

Saída:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- Para obter detalhes da API, consulte GetRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o documento da política de permissões da política denominada **oneClick_lambda_exec_role_policy** incorporada no perfil do IAM **lamda_exec_role**. O documento resultante da política é codificado em URL. Ela é decodificada neste exemplo com o método .NET **UrlDecode**.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Saída:

```
PolicyDocument                                          PolicyName
              UserName
--------------                                          ----------
              --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

Saída:

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- Para obter detalhes da API, consulte GetRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetSamlProvider`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetSamlProvider`.

CLI

AWS CLI

Recuperar o metadocumento do provedor SAML

Este exemplo recupera os detalhes sobre o provedor SAML 2.0 cujo ARM é `arn:aws:iam::123456789012:saml-provider/SAMLADFS`. A resposta inclui o documento de metadados que você obteve do provedor de identidade para criar a entidade do provedor AWS SAML, bem como as datas de criação e expiração.

```
aws iam get-saml-provider \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

Saída:

```
{
    "SAMLMetadataDocument": "...SAMLMetadataDocument-XML...",
    "CreateDate": "2017-03-06T22:29:46+00:00",
    "ValidUntil": "2117-03-06T22:29:46.433000+00:00",
    "Tags": [
        {
            "Key": "DeptID",
            "Value": "123456"
        },
        {
            "Key": "Department",
            "Value": "Accounting"
        }
    ]
}
```

Para obter mais informações, consulte Criação de provedores de identidade SAML do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetSamlProviderem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo recupera os detalhes sobre o provedor SAML 2.0 cujo ARM é arn:aws:iam: :123456789012:saml-. provider/SAMLADFS A resposta inclui o documento de metadados que você obteve do provedor de identidade para criar a entidade do provedor AWS SAML, bem como as datas de criação e expiração.

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Saída:

```
CreateDate                  SAMLMetadataDocument
        ValidUntil
----------                  --------------------
        ----------
```

```
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...    12/23/2114 12:16:54 PM
```

- Para obter detalhes da API, consulte GetSamlProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo recupera os detalhes sobre o provedor SAML 2.0 cujo ARM é arn:aws:iam: :123456789012:saml-. provider/SAMLADFS A resposta inclui o documento de metadados que você obteve do provedor de identidade para criar a entidade do provedor AWS SAML, bem como as datas de criação e expiração.

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Saída:

```
CreateDate                 SAMLMetadataDocument
        ValidUntil
----------                 --------------------
        ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...    12/23/2114 12:16:54 PM
```

- Para obter detalhes da API, consulte GetSamlProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetServerCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetServerCertificate`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<

 certificate.GetServerCertificateMetadata().GetServerCertificateName()
                 << std::endl << "Body: " << certificate.GetCertificateBody() <<
                 std::endl << "Chain: " << certificate.GetCertificateChain() <<
                 std::endl;
    }

    return result;
}
```

- Para obter detalhes da API, consulte GetServerCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para obter detalhes sobre um certificado de servidor em sua AWS conta

O `get-server-certificate` comando a seguir recupera todos os detalhes sobre o certificado de servidor especificado em sua AWS conta.

```
aws iam get-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

Saída:


```
{
    "ServerCertificate": {
        "ServerCertificateMetadata": {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        "CertificateBody": "-----BEGIN CERTIFICATE-----
            MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
            VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
            b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
            BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
            MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
            VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
            b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
            YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
            21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
            rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
            Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
            nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
            FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
            NYiytVbZPQUQ5Yaxu2jXnimvrszlaEXAMPLE=-----END CERTIFICATE-----",
        "CertificateChain": "-----BEGIN CERTIFICATE-----\nMIICiTCCAfICCQD6md
            7oRw0uXOjANBgkqhkiG9w0BAqQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgT
            AldBMRAwDgYDVQQHEwdTZWF0drGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAs
```

```
            TC0lBTSBDb25zb2xlMRIwEAYDVsQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQ
            jb20wHhcNMTEwNDI1MjA0NTIxWhtcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
            MCVVMxCzAJBgNVBAgTAldBMRAwDgsYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
            WF6b24xFDASBgNVBAsTC0lBTSBDb2d5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
            HzAdBgkqhkiG9w0BCQEWEG5vb25lQGfFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
            BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIgWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
            k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8mh9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
            ITxOUSQv7c7ugFFDzQGBzZswY6786m86gjpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
            AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCku4nUhVVxYUntneD9+h8Mg9q6q+auN
            KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FlkbFFBjvSfpJIlJ00zbhNYS5f6Guo
            EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjS;TbNYiytVbZPQUQ5Yaxu2jXnimvw
            3rrszlaEWEG5vb25lQGFtsYXpvbiEXAMPLE=\n-----END CERTIFICATE-----"
    }
}
```

Para listar os certificados do servidor disponíveis na sua AWS conta, use o `list-server-certificates` comando.

Para obter mais informações, consulte Gerenciar certificados de servidor no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetServerCertificateem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um certificado do servidor.

```
import { GetServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
```

```
 *
 * @param {string} certName
 * @returns
 */
export const getServerCertificate = async (certName) => {
  const command = new GetServerCertificateCommand({
    ServerCertificateName: certName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetServerCertificatea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
```

```
      console.log("Success", data);
    }
  }
);
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetServerCertificate a Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera detalhes sobre o certificado do servidor denominado **`MyServerCertificate`**. Você pode encontrar os detalhes do certificado nas propriedades **`CertificateBody`** e **`ServerCertificateMetadata`**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Saída:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
```

```
                            21uUSfwfEvySWtC2XADZ4nB
+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes da API, consulte GetServerCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera detalhes sobre o certificado do servidor denominado **`MyServerCertificate`**. Você pode encontrar os detalhes do certificado nas propriedades **`CertificateBody`** e **`ServerCertificateMetadata`**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Saída:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                            21uUSfwfEvySWtC2XADZ4nB
+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
```

```
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes da API, consulte GetServerCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetServiceLastAccessedDetails`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetServiceLastAccessedDetails`.

CLI

AWS CLI

Recuperar um relatório de acesso ao serviço

O exemplo de `get-service-last-accessed-details` a seguir recupera um relatório gerado anteriormente que lista os serviços acessados pelas entidades do IAM. Para gerar um relatório, use o comando `generate-service-last-accessed-details`.

```
aws iam get-service-last-accessed-details \
    --job-id 2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc
```

Saída:

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:50:35.929Z",
    "ServicesLastAccessed": [
        ...
        {
            "ServiceName": "AWS Lambda",
            "LastAuthenticated": "2019-09-30T23:02:00Z",
            "ServiceNamespace": "lambda",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/admin",
            "TotalAuthenticatedEntities": 6
        },
    ]
}
```

Para obter mais informações, consulte Como refinar as permissões no AWS uso das informações do último acesso no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetServiceLastAccessedDetailsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo fornece detalhes do último serviço acessado pela entidade do IAM (usuário, grupo, perfil ou política) associada na chamada de solicitação.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Saída:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Para obter detalhes da API, consulte GetServiceLastAccessedDetailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo fornece detalhes do último serviço acessado pela entidade do IAM (usuário, grupo, perfil ou política) associada na chamada de solicitação.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Saída:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Para obter detalhes da API, consulte GetServiceLastAccessedDetailsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetServiceLastAccessedDetailsWithEntities`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetServiceLastAccessedDetailsWithEntities`.

CLI

AWS CLI

Recuperar um relatório de acesso ao serviço com detalhes de um serviço

O exemplo de `get-service-last-accessed-details-with-entities` a seguir recupera um relatório que contém detalhes sobre os usuários do IAM e outras entidades que acessaram o serviço especificado. Para gerar um relatório, use o comando `generate-service-last-accessed-details`. Para obter uma lista de serviços acessados com namespaces, use `get-service-last-accessed-details`.

```
aws iam get-service-last-accessed-details-with-entities \
    --job-id 78b6c2ba-d09e-6xmp-7039-ecde30b26916 \
    --service-namespace lambda
```

Saída:

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:55:41.756Z",
    "JobCompletionDate": "2019-10-01T03:55:42.533Z",
    "EntityDetailsList": [
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/admin",
                "Name": "admin",
                "Type": "USER",
                "Id": "AIDAIO2XMPLENQEXAMPLE",
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-30T23:02:00Z"
        },
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/developer",
                "Name": "developer",
```

```
                "Type": "USER",
                "Id": "AIDAIBEYXMPL2YEXAMPLE",
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-16T19:34:00Z"
        }
    ]
}
```

Para obter mais informações, consulte Como refinar as permissões no AWS uso das informações do último acesso no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetServiceLastAccessedDetailsWithEntiesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo fornece o carimbo de data e hora do último acesso do serviço na solicitação pela respectiva entidade do IAM.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Saída:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

Saída:

```
EntityInfo                                    LastAuthenticated
```

```
----------                                    -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Saída:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Para obter detalhes da API, consulte GetServiceLastAccessedDetailsWithEntiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo fornece o carimbo de data e hora do último acesso do serviço na solicitação pela respectiva entidade do IAM.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Saída:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

Saída:

```
EntityInfo                                    LastAuthenticated
```

```
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Saída:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Para obter detalhes da API, consulte GetServiceLastAccessedDetailsWithEntiticsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetServiceLinkedRoleDeletionStatus`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetServiceLinkedRoleDeletionStatus`.

CLI

AWS CLI

Como verificar o status de uma solicitação para excluir um perfil vinculado ao serviço

O exemplo de `get-service-linked-role-deletion-status`, apresentado a seguir, exibe o status de uma solicitação anterior para excluir um perfil vinculado ao serviço. A operação de exclusão ocorre de forma assíncrona. Ao fazer a solicitação, você obtém um valor `DeletionTaskId` fornecido como parâmetro para esse comando.

```
aws iam get-service-linked-role-deletion-status \
    --deletion-task-id task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE
```

Saída:

```
{
```

```
"Status": "SUCCEEDED"
}
```

Para obter mais informações, consulte Usar perfis vinculados ao serviço no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetServiceLinkedRoleDeletionStatusem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetServiceLinkedRoleDeletionStatusCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} deletionTaskId
 */
export const getServiceLinkedRoleDeletionStatus = (deletionTaskId) => {
  const command = new GetServiceLinkedRoleDeletionStatusCommand({
    DeletionTaskId: deletionTaskId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetServiceLinkedRoleDeletionStatusa Referência AWS SDK para JavaScript da API.

Use **`GetUser`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetUser`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }
```

- Para obter detalhes da API, consulte GetUsera Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
##############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi
```

```
    return 1 # 1 in Bash script means false.
  fi
}
```

- Para obter detalhes da API, consulte GetUserem Referência de AWS CLI Comandos.

CLI

AWS CLI

Como obter informações sobre um usuário do IAM

O comando `get-user`, apresentado a seguir, obtém informações sobre o usuário do IAM denominado `Paulo`.

```
aws iam get-user \
    --user-name Paulo
```

Saída:

```
{
    "User": {
        "UserName": "Paulo",
        "Path": "/",
        "CreateDate": "2019-09-21T23:03:13Z",
        "UserId": "AIDA123456789EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Paulo"
    }
}
```

Para obter mais informações, consulte Gerenciar usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetUserem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
```

```
    case *types.NoSuchEntityException:
     log.Printf("User %v does not exist.\n", userName)
     err = nil
    default:
     log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
    }
   }
  } else {
   user = result.User
  }
  return user, err
}
```

- Para obter detalhes da API, consulte GetUsera Referência AWS SDK para Go da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera detalhes sobre o usuário chamado **`David`**.

```
Get-IAMUser -UserName David
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

Exemplo 2: este exemplo recupera detalhes sobre o usuário do IAM atualmente conectado.

```
Get-IAMUser
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Bob
```

```
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- Para obter detalhes da API, consulte GetUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera detalhes sobre o usuário chamado **`David`**.

```
Get-IAMUser -UserName David
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

Exemplo 2: este exemplo recupera detalhes sobre o usuário do IAM atualmente conectado.

```
Get-IAMUser
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- Para obter detalhes da API, consulte GetUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Retrieves a user's details
 #
 # @param user_name [String] The name of the user to retrieve
 # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
 def get_user(user_name)
   response = @iam_client.get_user(user_name: user_name)
   response.user
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("User '#{user_name}' not found.")
   nil
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error retrieving user '#{user_name}': #{e.message}")
   nil
 end
```

- Para obter detalhes da API, consulte GetUsera Referência AWS SDK para Ruby da API.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
```

```
import AWSS3


    public func getUser(name: String? = nil) async throws -> IAMClientTypes.User
 {
        let input = GetUserInput(
            userName: name
        )
        do {
            let output = try await iamClient.getUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            return user
        } catch {
            print("ERROR: getUser:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte GetUsera referência da API AWS SDK for Swift.

Usar **`GetUserPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetUserPolicy`.

CLI

AWS CLI

Listar detalhes da política de um usuário do IAM

O comando `get-user-policy` a seguir lista os detalhes da política especificada anexada ao usuário do IAM chamado Bob.

```
aws iam get-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

Saída:

```
{
    "UserName": "Bob",
    "PolicyName": "ExamplePolicy",
    "PolicyDocument": {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Action": "*",
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    }
}
```

Para obter uma lista de políticas para um usuário do IAM, use o comando `list-user-policies`.

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetUserPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera os detalhes da política em linha denominada **`Davids_IAM_Admin_Policy`** incorporada no usuário do IAM chamado **`David`**. O documento de política é codificado em URL.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Saída:

```
PolicyDocument                                              PolicyName
       UserName
--------------                                              ----------
       --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
       David
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- Para obter detalhes da API, consulte GetUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera os detalhes da política em linha denominada **`Davids_IAM_Admin_Policy`** incorporada no usuário do IAM chamado **`David`**. O documento de política é codificado em URL.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Saída:

```
PolicyDocument                                          PolicyName
       UserName
--------------                                          ----------
       --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
       David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- Para obter detalhes da API, consulte GetUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListAccessKeys`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListAccessKeys`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar chaves de acesso

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
```

```
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#
# Returns:
#       access_key_ids
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_list_access_keys() {

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_access_keys"
    echo "Lists the AWS Identity and Access Management (IAM) access key IDs for
 the specified user."
    echo "  -u user_name   The name of the IAM user."
    echo ""
  }

  local user_name response
  local option OPTARG # Required to use getopts command in a function.
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
```

```
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam list-access-keys \
    --user-name "$user_name" \
    --output text \
    --query 'AccessKeyMetadata[].AccessKeyId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-access-keys operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte ListAccessKeysem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                 const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                      << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                      std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                    Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                            key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                      std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                      statusString << std::setw(20) <<
                      key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
 std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
```

```
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListAccessKeysa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar os IDs da chave de acesso para um usuário do IAM

O comando `list-access-keys`, apresentado a seguir, lista os IDs das chaves de acesso para o usuário do IAM denominado Bob.

```
aws iam list-access-keys \
    --user-name Bob
```

Saída:

```
{
    "AccessKeyMetadata": [
        {
            "UserName": "Bob",
            "Status": "Active",
            "CreateDate": "2013-06-04T18:17:34Z",
            "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
        },
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CreateDate": "2013-06-06T20:42:26Z",
            "AccessKeyId": "AKIAI44QH8DHBEXAMPLE"
        }
    ]
}
```

Não é possível listar as chaves de acesso secretas para os usuários do IAM. Se as chaves de acesso secretas forem perdidas, você deverá criar novas chaves de acesso usando o comando `create-access-keys`.

Para obter mais informações, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListAccessKeysem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListAccessKeys lists the access keys for the specified user.
```

```
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- Para obter detalhes da API, consulte ListAccessKeysa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user for which access keys are
 retrieved.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void listKeys(IamClient iam, String userName) {
        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

                if (newMarker == null) {
                    ListAccessKeysRequest request =
 ListAccessKeysRequest.builder()
```

```
                            .userName(userName)
                            .build();

                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request =
 ListAccessKeysRequest.builder()
                            .userName(userName)
                            .marker(newMarker)
                            .build();

                    response = iam.listAccessKeys(request);
                }

                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
 metadata.accessKeyId());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

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

- Para obter detalhes da API, consulte ListAccessKeysa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as chaves de acesso.

```
import { ListAccessKeysCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} userName
 */
export async function* listAccessKeys(userName) {
  const command = new ListAccessKeysCommand({
    MaxItems: 5,
    UserName: userName,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListAccessKeysCommandOutput |
 undefined}
   */
  let response = await client.send(command);

  while (response?.AccessKeyMetadata?.length) {
    for (const key of response.AccessKeyMetadata) {
      yield key;
    }

    if (response.IsTruncated) {
      response = await client.send(
```

```
        new ListAccessKeysCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListAccessKeysa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME",
};

iam.listAccessKeys(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
```

```
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListAccessKeysa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListAccessKeysreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando lista as chaves de acesso do usuário do IAM chamado **Bob**. Observe que não é possível listar as chaves de acesso secretas dos usuários do IAM. Se as chaves de acesso secretas forem perdidas, você deverá criar novas chaves de acesso com o cmdlet **New-IAMAccessKey**.

```
Get-IAMAccessKey -UserName "Bob"
```

Saída:

```
AccessKeyId                CreateDate                  Status
 UserName
-----------                ----------                  ------
 --------
AKIAIOSFODNN7EXAMPLE       12/3/2014 10:53:41 AM       Active              Bob
AKIAI44QH8DHBEXAMPLE       6/6/2013 8:42:26 PM         Inactive            Bob
```

- Para obter detalhes da API, consulte ListAccessKeysem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando lista as chaves de acesso do usuário do IAM chamado **Bob**. Observe que não é possível listar as chaves de acesso secretas dos usuários do IAM. Se as chaves de acesso secretas forem perdidas, você deverá criar novas chaves de acesso com o cmdlet **New-IAMAccessKey**.

```
Get-IAMAccessKey -UserName "Bob"
```

Saída:

```
AccessKeyId                CreateDate                  Status
 UserName
-----------                ----------                  ------
 --------
AKIAIOSFODNN7EXAMPLE       12/3/2014 10:53:41 AM       Active              Bob
AKIAI44QH8DHBEXAMPLE       6/6/2013 8:42:26 PM         Inactive            Bob
```

- Para obter detalhes da API, consulte ListAccessKeysem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- Para obter detalhes da API, consulte a ListAccessKeysReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
```

```
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create
more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
```

```
    false
  end
end
```

- Para obter detalhes da API, consulte ListAccessKeysa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listaccesskeys(
          iv_username = iv_user_name ).
        MESSAGE 'Retrieved access key list.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListAccessKeysreferência da API AWS SDK for SAP ABAP.

Use **`ListAccountAliases`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListAccountAliases`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar sua conta

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
                break;
            }
            std::cout << std::left << std::setw(32) << "Alias" << std::endl;
            header = true;
        }

        for (const auto &alias: aliases) {
            std::cout << std::left << std::setw(32) << alias << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
```

```
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListAccountAliasesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar os aliases da conta

O comando `list-account-aliases`, apresentado a seguir, lista os aliases para a conta atual.

```
aws iam list-account-aliases
```

Saída:

```
{
    "AccountAliases": [
    "mycompany"
    ]
}
```

Para obter mais informações, consulte o ID AWS da sua conta e seu alias no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListAccountAliasesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }
```

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

- Para obter detalhes da API, consulte ListAccountAliasesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os aliases de conta.

```
import { ListAccountAliasesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listAccountAliases() {
  const command = new ListAccountAliasesCommand({ MaxItems: 5 });

  let response = await client.send(command);

  while (response.AccountAliases?.length) {
    for (const alias of response.AccountAliases) {
      yield alias;
```

```
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccountAliasesCommand({
          Marker: response.Marker,
          MaxItems: 5,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListAccountAliasesa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listAccountAliases({ MaxItems: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
```

```
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListAccountAliasesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAliases() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListAccountAliasesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna o alias da conta da Conta da AWS.

```
Get-IAMAccountAlias
```

Saída:

```
ExampleCo
```

- Para obter detalhes da API, consulte ListAccountAliasesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o alias da conta da Conta da AWS.

```
Get-IAMAccountAlias
```

Saída:

```
ExampleCo
```

- Para obter detalhes da API, consulte ListAccountAliasesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
```

```
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

- Para obter detalhes da API, consulte a ListAccountAliasesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, criar e excluir aliases da conta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases
```

```
    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListAccountAliasesa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listaccountaliases( ).
        MESSAGE 'Retrieved account alias list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing account aliases.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListAccountAliasesreferência da API AWS SDK for SAP ABAP.

Usar **`ListAttachedGroupPolicies`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListAttachedGroupPolicies`.

CLI

AWS CLI

Para listar todas as políticas gerenciadas anexadas ao grupo especificado

Este exemplo retorna os nomes e ARNs das políticas gerenciadas que estão anexadas ao grupo do IAM nomeado `Admins` na AWS conta.

```
aws iam list-attached-group-policies \
    --group-name Admins
```

Saída:

```
{
```

```
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListAttachedGroupPoliciesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse comando retorna os nomes e ARNs das políticas gerenciadas que estão anexadas ao grupo do IAM nomeado **Admins** na AWS conta. Para ver a lista de políticas em linha incorporadas no grupo, use o comando **Get-IAMGroupPolicyList**.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Saída:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/SecurityAudit                    SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess              AdministratorAccess
```

- Para obter detalhes da API, consulte ListAttachedGroupPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna os nomes e ARNs das políticas gerenciadas que estão anexadas ao grupo do IAM nomeado **`Admins`** na AWS conta. Para ver a lista de políticas em linha incorporadas no grupo, use o comando **`Get-IAMGroupPolicyList`**.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Saída:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/SecurityAudit                    SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess              AdministratorAccess
```

- Para obter detalhes da API, consulte ListAttachedGroupPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListAttachedRolePolicies`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListAttachedRolePolicies`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
 ListAttachedRolePoliciesAsync(string roleName)
    {
```

```
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
 _IAMService.Paginators.ListAttachedRolePolicies(new
 ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }
```

- Para obter detalhes da API, consulte ListAttachedRolePoliciesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar todas as políticas gerenciadas anexadas ao perfil especificado

Esse comando retorna os nomes e ARNs das políticas gerenciadas anexadas à função do IAM nomeada `SecurityAuditRole` na AWS conta.

```
aws iam list-attached-role-policies \
    --role-name SecurityAuditRole
```

Saída:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListAttachedRolePoliciesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
```

```
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}
```

- Para obter detalhes da API, consulte ListAttachedRolePoliciesa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Lista as políticas que estão anexadas a uma função.

```
import {
  ListAttachedRolePoliciesCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
```

```
 * @param {string} roleName
 */
export async function* listAttachedRolePolicies(roleName) {
  const command = new ListAttachedRolePoliciesCommand({
    RoleName: roleName,
  });

  let response = await client.send(command);

  while (response.AttachedPolicies?.length) {
    for (const policy of response.AttachedPolicies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAttachedRolePoliciesCommand({
          RoleName: roleName,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListAttachedRolePoliciesa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker
 = "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- Para obter detalhes da API, consulte ListAttachedRolePoliciesa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna os nomes e os ARNs das políticas gerenciadas anexadas ao perfil do IAM denominado **`SecurityAuditRole`** na conta da AWS . Para ver a lista de políticas em linha incorporadas no perfil, use o comando **`Get-IAMRolePolicyList`**.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Saída:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/SecurityAudit                    SecurityAudit
```

- Para obter detalhes da API, consulte ListAttachedRolePoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os nomes e os ARNs das políticas gerenciadas anexadas ao perfil do IAM denominado **`SecurityAuditRole`** na conta da AWS . Para ver a lista de políticas em linha incorporadas no perfil, use o comando **`Get-IAMRolePolicyList`**.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Saída:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/SecurityAudit                    SecurityAudit
```

- Para obter detalhes da API, consulte ListAttachedRolePoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
```

```
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- Para obter detalhes da API, consulte a ListAttachedRolePoliciesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
```

```
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
```

```
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListAttachedRolePoliciesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_attached_role_policies(
```

```
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput,
 SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListAttachedRolePoliciesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listattachedrolepolicies(
          iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved attached policy list for role.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListAttachedRolePoliciesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3



    /// Returns a list of AWS Identity and Access Management (IAM) policies
    /// that are attached to the role.
    ///
    /// - Parameter role: The IAM role to return the policy list for.
    ///
    /// - Returns: An array of `IAMClientTypes.AttachedPolicy` objects
    ///   describing each managed policy that's attached to the role.
    public func listAttachedRolePolicies(role: String) async throws ->
 [IAMClientTypes.AttachedPolicy] {
        var policyList: [IAMClientTypes.AttachedPolicy] = []

        // Use "Paginated" to get all the attached role polices.
        // This lets the SDK handle the 'isTruncated' in
 "ListAttachedRolePoliciesOutput".
        let input = ListAttachedRolePoliciesInput(
            roleName: role
        )
        let output = client.listAttachedRolePoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let attachedPolicies = page.attachedPolicies else {
                    print("Error: no attached policies returned.")
```

```
                    continue
                }
                for attachedPolicy in attachedPolicies {
                    policyList.append(attachedPolicy)
                }
            }
        } catch {
            print("ERROR: listAttachedRolePolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- Para obter detalhes da API, consulte ListAttachedRolePoliciesa referência da API AWS SDK for Swift.

Usar **`ListAttachedUserPolicies`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListAttachedUserPolicies`.

CLI

AWS CLI

Para listar todas as políticas gerenciadas anexadas ao usuário especificado

Esse comando retorna os nomes e ARNs das políticas gerenciadas para o usuário do IAM nomeado Bob na AWS conta.

```
aws iam list-attached-user-policies \
    --user-name Bob
```

Saída:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
```

```
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListAttachedUserPoliciesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse comando retorna os nomes e ARNs das políticas gerenciadas para o usuário do IAM nomeado **Bob** na AWS conta. Para ver a lista de políticas em linha incorporadas no usuário do IAM, use o comando **Get-IAMUserPolicyList**.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Saída:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/TesterPolicy                     TesterPolicy
```

- Para obter detalhes da API, consulte ListAttachedUserPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna os nomes e ARNs das políticas gerenciadas para o usuário do IAM nomeado **Bob** na AWS conta. Para ver a lista de políticas em linha incorporadas no usuário do IAM, use o comando **Get-IAMUserPolicyList**.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Saída:

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/TesterPolicy                       TesterPolicy
```

- Para obter detalhes da API, consulte ListAttachedUserPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListEntitiesForPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListEntitiesForPolicy`.

CLI

AWS CLI

Para listar todos os usuários, grupos e perfis aos quais a política gerenciada especificada está anexada

Este exemplo retorna uma lista de grupos, perfis e usuários do IAM que têm a política `arn:aws:iam::123456789012:policy/TestPolicy` anexada.

```
aws iam list-entities-for-policy \
    --policy-arn arn:aws:iam::123456789012:policy/TestPolicy
```

Saída:

```
{
    "PolicyGroups": [
        {
            "GroupName": "Admins",
            "GroupId": "AGPACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyUsers": [
        {
            "UserName": "Alice",
            "UserId": "AIDACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyRoles": [
```

```
        {
            "RoleName": "DevRole",
            "RoleId": "AROADBQP57FF2AEXAMPLE"
        }
    ],
    "IsTruncated": false
}
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListEntitiesForPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna uma lista de grupos, perfis e usuários do IAM que têm a política **`arn:aws:iam::123456789012:policy/TestPolicy`** anexada.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Saída:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- Para obter detalhes da API, consulte ListEntitiesForPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna uma lista de grupos, perfis e usuários do IAM que têm a política **`arn:aws:iam::123456789012:policy/TestPolicy`** anexada.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Saída:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- Para obter detalhes da API, consulte ListEntitiesForPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListGroupPolicies`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListGroupPolicies`.

CLI

AWS CLI

Para listar todas as políticas em linha anexadas ao grupo especificado

O comando `list-group-policies` a seguir lista os nomes das políticas em linha anexadas ao grupo do IAM denominado `Admins` na conta atual.

```
aws iam list-group-policies \
    --group-name Admins
```

Saída:

```
{
    "PolicyNames": [
        "AdminRoot",
        "ExamplePolicy"
    ]
}
```

Para obter mais informações, consulte Gerenciamento de políticas do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListGroupPoliciesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna uma lista das políticas em linha incorporadas no grupo **`Testers`**. Para obter as políticas gerenciadas anexadas ao grupo, use o comando **`Get-IAMAttachedGroupPolicyList`**.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Saída:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Para obter detalhes da API, consulte ListGroupPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna uma lista das políticas em linha incorporadas no grupo **`Testers`**. Para obter as políticas gerenciadas anexadas ao grupo, use o comando **`Get-IAMAttachedGroupPolicyList`**.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Saída:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Para obter detalhes da API, consulte ListGroupPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListGroups`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListGroups`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

- Para obter detalhes da API, consulte ListGroupsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os grupos do IAM para a conta atual

O comando `list-groups`, apresentado a seguir, lista os grupos do IAM na conta atual.

```
aws iam list-groups
```

Saída:

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-06-04T20:27:27.972Z",
            "GroupId": "AIDACKCEVSQ6C2EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "GroupName": "Admins"
        },
        {
            "Path": "/",
            "CreateDate": "2013-04-16T20:30:42Z",
            "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/S3-Admins",
            "GroupName": "S3-Admins"
        }
    ]
}
```

Para obter mais informações, consulte Gerenciar grupos de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListGroupsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group
 actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
 IamClient *iam.Client
}



// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(ctx context.Context, maxGroups int32)
 ([]types.Group, error) {
 var groups []types.Group
 result, err := wrapper.IamClient.ListGroups(ctx, &iam.ListGroupsInput{
  MaxItems: aws.Int32(maxGroups),
 })
 if err != nil {
  log.Printf("Couldn't list groups. Here's why: %v\n", err)
 } else {
  groups = result.Groups
 }
 return groups, err
}
```

- Para obter detalhes da API, consulte ListGroupsa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os grupos.

```
import { ListGroupsCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listGroups() {
  const command = new ListGroupsCommand({
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.Groups?.length) {
    for (const group of response.Groups) {
      yield group;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListGroupsCommand({
          Marker: response.Marker,
          MaxItems: 10,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListGroupsa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- Para obter detalhes da API, consulte ListGroupsa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo retorna uma coleção de todos os grupos do IAM definidos no atual Conta da AWS.

```
Get-IAMGroupList
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- Para obter detalhes da API, consulte ListGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna uma coleção de todos os grupos do IAM definidos no atual Conta da AWS.

```
Get-IAMGroupList
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
```

```
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- Para obter detalhes da API, consulte ListGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
    """
    try:
        for group in iam.groups.limit(count):
            logger.info("Group: %s", group.name)
    except ClientError:
        logger.exception("Couldn't list groups for the account.")
        raise
```

- Para obter detalhes da API, consulte a ListGroupsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Para obter detalhes da API, consulte ListGroupsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListGroupsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listgroups( ).
        MESSAGE 'Retrieved group list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing groups.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListGroupsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listGroups() async throws -> [String] {
        var groupList: [String] = []

        // Use "Paginated" to get all the groups.
        // This lets the SDK handle the 'isTruncated' property in
 "ListGroupsOutput".
        let input = ListGroupsInput()

        let pages = client.listGroupsPaginated(input: input)
        do {
            for try await page in pages {
                guard let groups = page.groups else {
                    print("Error: no groups returned.")
                    continue
                }
```

```
                for group in groups {
                    if let name = group.groupName {
                        groupList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listGroups:", dump(error))
            throw error
        }
        return groupList
    }
```

- Para obter detalhes da API, consulte ListGroupsa referência da API AWS SDK for Swift.

Usar **`ListGroupsForUser`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListGroupsForUser`.

CLI

AWS CLI

Listar os grupos aos quais um usuário do IAM pertence

O comando `list-groups-for-user` a seguir exibe os grupos aos quais o usuário do IAM chamado Bob pertence.

```
aws iam list-groups-for-user \
    --user-name Bob
```

Saída:

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:18:08Z",
            "GroupId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admin",
```

```
            "GroupName": "Admin"
        },
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:37:28Z",
            "GroupId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/s3-Users",
            "GroupName": "s3-Users"
        }
    ]
}
```

Para obter mais informações, consulte Gerenciar grupos de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListGroupsForUserem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna a lista de grupos do IAM aos quais o usuário do IAM **`David`** pertence.

```
Get-IAMGroupForUser -UserName David
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
```

```
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- Para obter detalhes da API, consulte ListGroupsForUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a lista de grupos do IAM aos quais o usuário do IAM **`David`** pertence.

```
Get-IAMGroupForUser -UserName David
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- Para obter detalhes da API, consulte ListGroupsForUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListInstanceProfiles`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListInstanceProfiles`.

CLI

AWS CLI

Listar os perfis de instância da conta

O comando `list-instance-profiles` a seguir lista os perfis de instância associados à conta atual.

```
aws iam list-instance-profiles
```

Saída:

```
{
    "InstanceProfiles": [
        {
            "Path": "/",
            "InstanceProfileName": "example-dev-role",
            "InstanceProfileId": "AIPAIXEU4NUHUPEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-dev-role",
            "CreateDate": "2023-09-21T18:17:41+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-dev-role",
                    "RoleId": "AROAJ52OTH4H7LEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-dev-role",
                    "CreateDate": "2023-09-21T18:17:40+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        },
```

```
        {
            "Path": "/",
            "InstanceProfileName": "example-s3-role",
            "InstanceProfileId": "AIPAJVJVNRIQFREXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-s3-role",
            "CreateDate": "2023-09-21T18:18:50+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-s3-role",
                    "RoleId": "AROAINUBC5O7XLEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-s3-role",
                    "CreateDate": "2023-09-21T18:18:49+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

Para obter mais informações, consulte Usar perfis de instância no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListInstanceProfilesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo retorna uma coleção dos perfis de instância definidos na versão atual Conta da AWS.

```
Get-IAMInstanceProfileList
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Para obter detalhes da API, consulte ListInstanceProfilesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna uma coleção dos perfis de instância definidos na versão atual Conta da AWS.

```
Get-IAMInstanceProfileList
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Para obter detalhes da API, consulte ListInstanceProfilesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListInstanceProfilesForRole`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListInstanceProfilesForRole`.

CLI

AWS CLI

Listar os perfis de instância de um perfil do IAM

O comando `list-instance-profiles-for-role` a seguir lista os perfis de instância associados ao perfil `Test-Role`.

```
aws iam list-instance-profiles-for-role \
    --role-name Test-Role
```

Saída:

```
{
    "InstanceProfiles": [
        {
            "InstanceProfileId": "AIDGPMS9RO4H3FEXAMPLE",
            "Roles": [
                {
                    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                    "RoleId": "AIDACKCEVSQ6C2EXAMPLE",
                    "CreateDate": "2013-06-07T20:42:15Z",
                    "RoleName": "Test-Role",
                    "Path": "/",
                    "Arn": "arn:aws:iam::123456789012:role/Test-Role"
                }
            ],
            "CreateDate": "2013-06-07T21:05:24Z",
            "InstanceProfileName": "ExampleInstanceProfile",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:instance-profile/
ExampleInstanceProfile"
        }
    ]
}
```

Para obter mais informações, consulte Usar perfis de instância no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListInstanceProfilesForRoleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna detalhes do perfil de instância associado ao perfil **`ec2instancerole`**.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Saída:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- Para obter detalhes da API, consulte ListInstanceProfilesForRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes do perfil de instância associado ao perfil **`ec2instancerole`**.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Saída:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- Para obter detalhes da API, consulte ListInstanceProfilesForRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListMfaDevices`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListMfaDevices`.

CLI

AWS CLI

Listar todos os dispositivos de MFA de um usuário especificado

Este exemplo retorna detalhes sobre o dispositivo de MFA atribuído ao usuário do IAM Bob.

```
aws iam list-mfa-devices \
    --user-name Bob
```

Saída:

```
{
    "MFADevices": [
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Bob",
            "EnableDate": "2019-10-28T20:37:09+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "GAKT12345678",
            "EnableDate": "2023-02-18T21:44:42+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey1-7XNL7NFNLZ123456789EXAMPLE",
            "EnableDate": "2023-09-19T02:25:35+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey2-VDRQTDBBN5123456789EXAMPLE",
            "EnableDate": "2023-09-19T01:49:18+00:00"
        }
    ]
}
```

Para obter mais informações, consulte Uso de autenticação multifator (MFA) na AWS no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListMfaDevicesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna detalhes sobre o dispositivo de MFA atribuído ao usuário do IAM **`David`**. Neste exemplo, você percebe que é um dispositivo virtual porque o **`SerialNumber`** é um ARN em vez do número de série real de um dispositivo físico.

```
Get-IAMMFADevice -UserName David
```

Saída:

```
EnableDate                  SerialNumber                          UserName
----------                  ------------                          --------
4/8/2015 9:41:10 AM         arn:aws:iam::123456789012:mfa/David    David
```

- Para obter detalhes da API, consulte ListMfaDevicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre o dispositivo de MFA atribuído ao usuário do IAM **`David`**. Neste exemplo, você percebe que é um dispositivo virtual porque o **`SerialNumber`** é um ARN em vez do número de série real de um dispositivo físico.

```
Get-IAMMFADevice -UserName David
```

Saída:

```
EnableDate                  SerialNumber                          UserName
----------                  ------------                          --------
4/8/2015 9:41:10 AM         arn:aws:iam::123456789012:mfa/David    David
```

- Para obter detalhes da API, consulte ListMfaDevicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListOpenIdConnectProviders`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListOpenIdConnectProviders`.

CLI

AWS CLI

Para listar informações sobre os provedores do OpenID Connect na conta AWS

Este exemplo retorna uma lista de ARNS de todos os provedores do OpenID Connect definidos AWS na conta atual.

```
aws iam list-open-id-connect-providers
```

Saída:

```
{
    "OpenIDConnectProviderList": [
        {
            "Arn": "arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com"
        }
    ]
}
```

Para obter mais informações, consulte Criar provedores de identidade OpenID Connect (OIDC) no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListOpenIdConnectProvidersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna uma lista de ARNS de todos os provedores OpenID Connect definidos na Conta da AWS atual.

```
Get-IAMOpenIDConnectProviderList
```

Saída:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Para obter detalhes da API, consulte ListOpenIdConnectProvidersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna uma lista de ARNS de todos os provedores OpenID Connect definidos na Conta da AWS atual.

```
Get-IAMOpenIDConnectProviderList
```

Saída:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Para obter detalhes da API, consulte ListOpenIdConnectProvidersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListPolicies`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListPolicies`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Políticas gerenciadas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(64) << "Description" << std::setw(12) <<
                      "CreateDate" << std::endl;
            header = true;
        }

        const auto &policies = outcome.GetResult().GetPolicies();
        for (const auto &policy: policies) {
            std::cout << std::left << std::setw(55) <<
                      policy.GetPolicyName() << std::setw(30) <<
                      policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
                      std::setw(64) << policy.GetDescription() << std::setw(12)
 <<
                      policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
```

```
    return true;
}
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar as políticas gerenciadas que estão disponíveis para sua AWS conta

Este exemplo retorna uma coleção das duas primeiras políticas gerenciadas disponíveis na AWS conta atual.

```
aws iam list-policies \
    --max-items 3
```

Saída:

```
{
    "Policies": [
        {
            "PolicyName": "AWSCloudTrailAccessPolicy",
            "PolicyId": "ANPAXQE2B5PJ7YEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:policy/AWSCloudTrailAccessPolicy",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 0,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2019-09-04T17:43:42+00:00",
            "UpdateDate": "2019-09-04T17:43:42+00:00"
        },
        {
            "PolicyName": "AdministratorAccess",
            "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",
            "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 6,
            "PermissionsBoundaryUsageCount": 0,
```

```
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:46+00:00",
            "UpdateDate": "2015-02-06T18:39:46+00:00"
        },
        {
            "PolicyName": "PowerUserAccess",
            "PolicyId": "ANPAJYRXTHIB4FOVS3ZXS",
            "Arn": "arn:aws:iam::aws:policy/PowerUserAccess",
            "Path": "/",
            "DefaultVersionId": "v5",
            "AttachmentCount": 1,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:47+00:00",
            "UpdateDate": "2023-07-06T22:04:00+00:00"
        }
    ],
    "NextToken": "EXAMPLErZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiA4fQ=="
}
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListPoliciesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as políticas.

```
import { ListPoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listPolicies() {
  const command = new ListPoliciesCommand({
    MaxItems: 10,
    OnlyAttached: false,
    // List only the customer managed policies in your Amazon Web Services
 account.
    Scope: "Local",
  });

  let response = await client.send(command);

  while (response.Policies?.length) {
    for (const policy of response.Policies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListPoliciesCommand({
          Marker: response.Marker,
          MaxItems: 10,
          OnlyAttached: false,
          Scope: "Local",
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse exemplo retorna uma coleção das três primeiras políticas gerenciadas disponíveis na AWS conta atual. Como não **`-scope`** está especificado, ele usa como padrão **`all`** e inclui políticas AWS gerenciadas e gerenciadas pelo cliente.

```
Get-IAMPolicyList -MaxItem 3
```

Saída:

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
```

```
UpdateDate       : 2/11/2015 9:21:45 AM
```

Exemplo 2: Este exemplo retorna uma coleção das duas primeiras políticas gerenciadas pelo cliente disponíveis na AWS conta corrente. Ele usa **`-Scope local`** para limitar a saída somente às políticas gerenciadas pelo cliente.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Saída:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- Para obter detalhes da API, consulte ListPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna uma coleção das três primeiras políticas gerenciadas disponíveis na AWS conta atual. Como não **`-scope`** está especificado, ele usa como padrão **`all`** e inclui políticas AWS gerenciadas e gerenciadas pelo cliente.

```
Get-IAMPolicyList -MaxItem 3
```

Saída:

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

Exemplo 2: Este exemplo retorna uma coleção das duas primeiras políticas gerenciadas pelo cliente disponíveis na AWS conta corrente. Ele usa **`-Scope local`** para limitar a saída somente às políticas gerenciadas pelo cliente.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Saída:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- Para obter detalhes da API, consulte ListPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
```

```
                    'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- Para obter detalhes da API, consulte a ListPoliciesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end
```

```
  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
```

```
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- Para obter detalhes da API, consulte a ListPoliciesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listpolicies( iv_scope = iv_scope ).
        MESSAGE 'Retrieved policy list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policies.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListPoliciesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listPolicies() async throws -> [MyPolicyRecord] {
        var policyList: [MyPolicyRecord] = []

        // Use "Paginated" to get all the policies.
```

```
        // This lets the SDK handle the 'isTruncated' in "ListPoliciesOutput".
        let input = ListPoliciesInput()
        let output = client.listPoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let policies = page.policies else {
                    print("Error: no policies returned.")
                    continue
                }

                for policy in policies {
                    guard let name = policy.policyName,
                          let id = policy.policyId,
                          let arn = policy.arn
                    else {
                        throw ServiceHandlerError.noSuchPolicy
                    }
                    policyList.append(MyPolicyRecord(name: name, id: id, arn:
arn))
                }
            }
        } catch {
            print("ERROR: listPolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- Para obter detalhes da API, consulte ListPoliciesa referência da API AWS SDK for Swift.

Use **`ListPolicyVersions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListPolicyVersions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Políticas gerenciadas
- Reverter uma versão de política

CLI

AWS CLI

Listar informações sobre as versões da política gerenciada especificada

Este exemplo retorna a lista de versões disponíveis da política cujo ARN é `arn:aws:iam::123456789012:policy/MySamplePolicy`.

```
aws iam list-policy-versions \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Saída:

```
{
    "IsTruncated": false,
    "Versions": [
        {
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2015-06-02T23:19:44Z"
        },
        {
        "VersionId": "v1",
        "IsDefaultVersion": false,
        "CreateDate": "2015-06-02T22:30:47Z"
        }
    ]
}
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListPolicyVersionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna a lista de versões disponíveis da política cujo ARN é **`arn:aws:iam::123456789012:policy/MyManagedPolicy`**. Para obter o documento de

política de uma versão específica, use o comando **`Get-IAMPolicyVersion`** e especifique o **`VersionId`** do que você deseja.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy
```

Saída:

```
CreateDate                  Document                IsDefaultVersion
       VersionId
----------                  --------                ----------------
       ---------
2/12/2015 9:39:53 AM                                True
       v2
2/12/2015 9:39:09 AM                                False
       v1
```

- Para obter detalhes da API, consulte ListPolicyVersionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a lista de versões disponíveis da política cujo ARN é **`arn:aws:iam::123456789012:policy/MyManagedPolicy`**. Para obter o documento de política de uma versão específica, use o comando **`Get-IAMPolicyVersion`** e especifique o **`VersionId`** do que você deseja.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy
```

Saída:

```
CreateDate                  Document                IsDefaultVersion
       VersionId
----------                  --------                ----------------
       ---------
2/12/2015 9:39:53 AM                                True
       v2
2/12/2015 9:39:09 AM                                False
       v1
```

- Para obter detalhes da API, consulte ListPolicyVersionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->listpolicyversions(
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListPolicyVersionsreferência da API AWS SDK for SAP ABAP.

Use **`ListRolePolicies`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListRolePolicies`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos da proteção de dispositivos da Internet das Coisas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
```

- Para obter detalhes da API, consulte ListRolePoliciesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar as políticas anexadas a um perfil do IAM

O comando `list-role-policies`, apresentado a seguir, lista os nomes das políticas de permissões para o perfil do IAM especificado.

```
aws iam list-role-policies \
    --role-name Test-Role
```

Saída:

```
{
    "PolicyNames": [
        "ExamplePolicy"
    ]
}
```

Para visualizar a política de confiança anexada a um perfil, use o comando `get-role`. Para visualizar os detalhes de uma política de permissões, use o comando `get-role-policy`.

Para obter mais informações, consulte Criar perfis do IAM no AWS Guia do usuário do IAM.

- Para obter detalhes da API, consulte ListRolePoliciesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
```

```
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx,
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

- Para obter detalhes da API, consulte ListRolePoliciesa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as políticas.

```
import { ListRolePoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} roleName
 */
export async function* listRolePolicies(roleName) {
  const command = new ListRolePoliciesCommand({
    RoleName: roleName,
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.PolicyNames?.length) {
    for (const policyName of response.PolicyNames) {
      yield policyName;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolePoliciesCommand({
          RoleName: roleName,
          MaxItems: 10,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListRolePoliciesa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient-
>listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- Para obter detalhes da API, consulte ListRolePoliciesa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna a lista de nomes de políticas em linha incorporadas no perfil do IAM **`lamda_exec_role`**. Para ver os detalhes de uma política em linha, use o comando **`Get-IAMRolePolicy`**.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Saída:

```
oneClick_lambda_exec_role_policy
```

- Para obter detalhes da API, consulte ListRolePoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a lista de nomes de políticas em linha incorporadas no perfil do IAM **`lamda_exec_role`**. Para ver os detalhes de uma política em linha, use o comando **`Get-IAMRolePolicy`**.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Saída:

```
oneClick_lambda_exec_role_policy
```

- Para obter detalhes da API, consulte ListRolePoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- Para obter detalhes da API, consulte a ListRolePoliciesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

- Para obter detalhes da API, consulte ListRolePoliciesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListRolePoliciesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listrolepolicies(
          iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved inline policy list for role.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListRolePoliciesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listRolePolicies(role: String) async throws -> [String] {
        var policyList: [String] = []

        // Use "Paginated" to get all the role policies.
```

```
        // This lets the SDK handle the 'isTruncated' in
 "ListRolePoliciesOutput".
        let input = ListRolePoliciesInput(
            roleName: role
        )
        let pages = client.listRolePoliciesPaginated(input: input)

        do {
            for try await page in pages {
                guard let policies = page.policyNames else {
                    print("Error: no role policies returned.")
                    continue
                }

                for policy in policies {
                    policyList.append(policy)
                }
            }
        } catch {
            print("ERROR: listRolePolicies:", dump(error))
            throw error
        }
        return policyList
    }
```

- Para obter detalhes da API, consulte ListRolePoliciesa referência da API AWS SDK for Swift.

Usar **`ListRoleTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListRoleTags`.

CLI

AWS CLI

Listar as tags anexadas a um perfil

O comando `list-role-tags` a seguir recupera a lista de tags associadas ao perfil especificado.

```
aws iam list-role-tags \
```

```
    --role-name production-role
```

Saída:

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

Para obter mais informações, consulte Recursos de tags do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListRoleTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo busca a tag associada ao perfil.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Para obter detalhes da API, consulte ListRoleTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo busca a tag associada ao perfil.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Para obter detalhes da API, consulte ListRoleTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListRoles`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListRoles`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }
```

- Para obter detalhes da API, consulte ListRolesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os perfis do IAM para a conta atual

O comando `list-roles`, apresentado a seguir, lista os perfis do IAM para a conta atual.

```
aws iam list-roles
```

Saída:

```
{
    "Roles": [
        {
            "Path": "/",
            "RoleName": "ExampleRole",
            "RoleId": "AROAJ52OTH4H7LEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/ExampleRole",
            "CreateDate": "2017-09-12T19:23:36+00:00",
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        },
        {
            "Path": "/example_path/",
            "RoleName": "ExampleRoleWithPath",
            "RoleId": "AROAI4QRP7UFT7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/example_path/
ExampleRoleWithPath",
            "CreateDate": "2023-09-21T20:29:38+00:00",
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
```

```
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        }
    ]
}
```

Para obter mais informações, consulte Criação de perfis do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListRolesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
```

```
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}
```

- Para obter detalhes da API, consulte ListRolesa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os perfis.

```
import { ListRolesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
```

```
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listRoles() {
  const command = new ListRolesCommand({
    MaxItems: 10,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListRolesCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.Roles?.length) {
    for (const role of response.Roles) {
      yield role;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolesCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListRolesa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- Para obter detalhes da API, consulte ListRolesa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera uma lista de todos os perfis do IAM na Conta da AWS.

```
Get-IAMRoleList
```

- Para obter detalhes da API, consulte ListRolesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera uma lista de todos os perfis do IAM na Conta da AWS.

```
Get-IAMRoleList
```

- Para obter detalhes da API, consulte ListRolesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
```

```
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- Para obter detalhes da API, consulte a ListRolesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count

      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end

  role_names
rescue Aws::IAM::Errors::ServiceError => e
```

```
    @logger.error("Couldn't list roles for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte ListRolesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListRolesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listroles( ).
        MESSAGE 'Retrieved role list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing roles.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListRolesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listRoles() async throws -> [String] {
        var roleList: [String] = []

        // Use "Paginated" to get all the roles.
```

```
        // This lets the SDK handle the 'isTruncated' in "ListRolesOutput".
        let input = ListRolesInput()
        let pages = client.listRolesPaginated(input: input)

        do {
            for try await page in pages {
                guard let roles = page.roles else {
                    print("Error: no roles returned.")
                    continue
                }

                for role in roles {
                    if let name = role.roleName {
                        roleList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listRoles:", dump(error))
            throw error
        }
        return roleList
    }
```

- Para obter detalhes da API, consulte ListRolesa referência da API AWS SDK for Swift.

Use **`ListSamlProviders`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListSAMLProviders`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
 ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK para .NET .

CLI

AWS CLI

Para listar os provedores de SAML na conta AWS

Este exemplo recupera a lista de provedores de SAML 2.0 criados na conta atual AWS .

```
aws iam list-saml-providers
```

Saída:

```
{
    "SAMLProviderList": [
        {
            "Arn": "arn:aws:iam::123456789012:saml-provider/SAML-ADFS",
            "ValidUntil": "2015-06-05T22:45:14Z",
            "CreateDate": "2015-06-05T22:45:14Z"
        }
    ]
}
```

Para obter mais informações, consulte Criação de provedores de identidade SAML do IAM no Guia do usuário do AWS IAM.

- Consulte detalhes da API em ListSAMLProviders na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK para Go .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os provedores SAML.

```
import { ListSAMLProvidersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listSamlProviders = async () => {
  const command = new ListSAMLProvidersCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API do AWS SDK para JavaScript .

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK para PHP .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera a lista de provedores SAML 2.0 criados na Conta da AWS atual. Ele retorna o ARN, a data de criação e a data de expiração de cada provedor SAML.

```
Get-IAMSAMLProviderList
```

Saída:

```
Arn                                                 CreateDate
   ValidUntil
---                                                 ----------
   ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
   12/23/2114 12:16:54 PM
```

- Para ver detalhes da API, consulte ListSAMLProviders na Referência de cmdlets do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a lista de provedores SAML 2.0 criados na Conta da AWS atual. Ele retorna o ARN, a data de criação e a data de expiração de cada provedor SAML.

```
Get-IAMSAMLProviderList
```

Saída:

```
Arn                                                CreateDate
   ValidUntil
---                                                ----------
   ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
   12/23/2114 12:16:54 PM
```

- Para obter detalhes sobre a API, consulte ListSAMLProviders na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
```

```
            logger.info("Got SAML provider %s.", provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
```

```
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_saml_providers(
    client: &Client,
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {
    let response = client.list_saml_providers().send().await?;

    Ok(response)
}
```

- Consulte detalhes da API em ListSAMLProvider na referência da API do AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listsamlproviders( ).
        MESSAGE 'Retrieved SAML provider list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing SAML providers.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte ListSAMLProviders na Referência de APIs do AWS SDK para SAP ABAP.

Use **`ListServerCertificates`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListServerCertificates`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::listServerCertificates(
        const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
```

```
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(14) << "UploadDate" << std::setw(14) <<
                      "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
                outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                      certificate.GetServerCertificateName() << std::setw(30) <<
                      certificate.GetServerCertificateId() << std::setw(80) <<
                      certificate.GetArn() << std::setw(14) <<

certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::setw(14) <<

certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
```

```
    return true;
}
```

- Para obter detalhes da API, consulte ListServerCertificatesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar os certificados do servidor em sua AWS conta

O `list-server-certificates` comando a seguir lista todos os certificados de servidor armazenados e disponíveis para uso em sua AWS conta.

```
aws iam list-server-certificates
```

Saída:

```
{
    "ServerCertificateMetadataList": [
        {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        {
            "Path": "/cloudfront/",
            "ServerCertificateName": "MyTestCert",
            "ServerCertificateId": "ASCAEXAMPLE456EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyTestCert",
            "UploadDate": "2015-04-21T18:14:16+00:00",
            "Expiration": "2018-01-14T17:52:36+00:00"
        }
    ]
```

```
}
```

Para obter mais informações, consulte Gerenciar certificados de servidor no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListServerCertificatesem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os certificados.

```
import { ListServerCertificatesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listServerCertificates() {
  const command = new ListServerCertificatesCommand({});
  let response = await client.send(command);

  while (response.ServerCertificateMetadataList?.length) {
    for await (const cert of response.ServerCertificateMetadataList) {
      yield cert;
    }

    if (response.IsTruncated) {
      response = await client.send(new ListServerCertificatesCommand({}));
```

```
    } else {
      break;
    }
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListServerCertificatesa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listServerCertificates({}, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListServerCertificatesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera a lista de certificados de servidor enviados à Conta da AWS atual.

```
Get-IAMServerCertificateList
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes da API, consulte ListServerCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a lista de certificados de servidor enviados à Conta da AWS atual.

```
Get-IAMServerCertificateList
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes da API, consulte ListServerCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, atualizar e excluir certificados de servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
```

```
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListServerCertificatesa Referência AWS SDK para Ruby da API.

Usar **`ListSigningCertificates`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListSigningCertificates`.

CLI

AWS CLI

Listar os certificados de assinatura de um usuário do IAM

O comando `list-signing-certificates` a seguir lista os certificados de assinatura do usuário do IAM denominado Bob.

```
aws iam list-signing-certificates \
    --user-name Bob
```

Saída:

```
{
    "Certificates": [
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-
body>-----END CERTIFICATE-----",
            "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
            "UploadDate": "2013-06-06T21:40:08Z"
        }
    ]
}
```

Para obter mais informações, consulte Manage signing certificates no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte ListSigningCertificatesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera detalhes sobre o certificado de assinatura associado ao usuário chamado **Bob**.

```
Get-IAMSigningCertificate -UserName Bob
```

Saída:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Para obter detalhes da API, consulte ListSigningCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera detalhes sobre o certificado de assinatura associado ao usuário chamado **Bob**.

```
Get-IAMSigningCertificate -UserName Bob
```

Saída:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Para obter detalhes da API, consulte ListSigningCertificatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListUserPolicies`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListUserPolicies`.

CLI

AWS CLI

Para listar as políticas para um usuário do IAM

O comando `list-user-policies`, apresentado a seguir, lista as políticas anexadas ao usuário do IAM denominado `Bob`.

```
aws iam list-user-policies \
    --user-name Bob
```

Saída:

```
{
    "PolicyNames": [
        "ExamplePolicy",
        "TestPolicy"
    ]
}
```

Para obter mais informações, consulte Como criar um usuário do IAM em sua AWS conta no Guia AWS do usuário do IAM.

- Para obter detalhes da API, consulte ListUserPoliciesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx,
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

- Para obter detalhes da API, consulte ListUserPoliciesa Referência AWS SDK para Go da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera a lista de nomes das políticas em linha incorporadas no usuário do IAM chamado **`David`**.

```
Get-IAMUserPolicyList -UserName David
```

Saída:

```
Davids_IAM_Admin_Policy
```

- Para obter detalhes da API, consulte ListUserPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a lista de nomes das políticas em linha incorporadas no usuário do IAM chamado **`David`**.

```
Get-IAMUserPolicyList -UserName David
```

Saída:

```
Davids_IAM_Admin_Policy
```

- Para obter detalhes da API, consulte ListUserPoliciesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListUserTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListUserTags`.

CLI

AWS CLI

Listar as tags anexadas a um usuário

O comando `list-user-tags` a seguir recupera a lista de tags associadas ao usuário do IAM especificado.

```
aws iam list-user-tags \
    --user-name alice
```

Saída:

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

Para obter mais informações, consulte Recursos de tags do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListUserTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo busca a tag associada ao usuário.

```
Get-IAMUserTagList -UserName joe
```

- Para obter detalhes da API, consulte ListUserTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo busca a tag associada ao usuário.

```
Get-IAMUserTagList -UserName joe
```

- Para obter detalhes da API, consulte ListUserTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListUsers`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListUsers`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar usuários somente leitura e leitura/gravação usando

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
```

```
    }
```

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_list_users
#
# List the IAM users in the account.
#
# Returns:
#       The list of users names
#    And:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_list_users() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_users"
```

```
    echo "Lists the AWS Identity and Access Management (IAM) user in the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws iam list-users \
    --output text \
    --query "Users[].UserName")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-users operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte ListUsersem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig)
 {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
                      std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &users = outcome.GetResult().GetUsers();
        for (const auto &user: users) {
            std::cout << std::left << std::setw(32) << user.GetUserName() <<
                      std::setw(30) << user.GetUserId() << std::setw(64) <<
                      user.GetArn() << std::setw(20) <<
                      user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
                      << std::endl;
        }
```

```
        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar os usuários do IAM

O comando `list-users`, apresentado a seguir, lista os usuários do IAM na conta atual.

```
aws iam list-users
```

Saída:

```
{
    "Users": [
        {
            "UserName": "Adele",
            "Path": "/",
            "CreateDate": "2013-03-07T05:14:48Z",
            "UserId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Adele"
        },
        {
            "UserName": "Bob",
            "Path": "/",
            "CreateDate": "2012-09-21T23:03:13Z",
            "UserId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        }
    ]
```

```
}
```

Para obter mais informações, consulte Listagem de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListUsersem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
```

```
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}
```

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

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

```
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request =
 ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
 user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",

 permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
```

```
                }
            }

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

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os usuários.

```
import { ListUsersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listUsers = async () => {
  const command = new ListUsersCommand({ MaxItems: 10 });

  const response = await client.send(command);

  for (const { UserName, CreateDate } of response.Users) {
    console.log(`${UserName} created on: ${CreateDate}`);
  }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 10,
};

iam.listUsers(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllUsers() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a ListUsersreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();
```

```
    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo recupera uma coleção de usuários no atual Conta da AWS.

```
Get-IAMUserList
```

Saída:

```
      Arn              : arn:aws:iam::123456789012:user/Administrator
      CreateDate       : 10/16/2014 9:03:09 AM
      PasswordLastUsed : 3/4/2015 12:12:33 PM
      Path             : /
      UserId           : 7K3GJEANSKZF2EXAMPLE1
      UserName         : Administrator

      Arn              : arn:aws:iam::123456789012:user/Bob
      CreateDate       : 4/6/2015 12:54:42 PM
      PasswordLastUsed : 1/1/0001 12:00:00 AM
      Path             : /
      UserId           : L3EWNONDOM3YUEXAMPLE2
      UserName         : bab
```

```
      Arn              : arn:aws:iam::123456789012:user/David
      CreateDate       : 12/10/2014 3:39:27 PM
      PasswordLastUsed : 3/19/2015 8:44:04 AM
      Path             : /
      UserId           : Y4FKWQCXTA52QEXAMPLE3
      UserName         : David
```

- Para obter detalhes da API, consulte ListUsersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo recupera uma coleção de usuários no atual Conta da AWS.

```
Get-IAMUserList
```

Saída:

```
      Arn              : arn:aws:iam::123456789012:user/Administrator
      CreateDate       : 10/16/2014 9:03:09 AM
      PasswordLastUsed : 3/4/2015 12:12:33 PM
      Path             : /
      UserId           : 7K3GJEANSKZF2EXAMPLE1
      UserName         : Administrator

      Arn              : arn:aws:iam::123456789012:user/Bob
      CreateDate       : 4/6/2015 12:54:42 PM
      PasswordLastUsed : 1/1/0001 12:00:00 AM
      Path             : /
      UserId           : L3EWNONDOM3YUEXAMPLE2
      UserName         : bab

      Arn              : arn:aws:iam::123456789012:user/David
      CreateDate       : 12/10/2014 3:39:27 PM
      PasswordLastUsed : 3/19/2015 8:44:04 AM
      Path             : /
      UserId           : Y4FKWQCXTA52QEXAMPLE3
      UserName         : David
```

- Para obter detalhes da API, consulte ListUsersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- Para obter detalhes da API, consulte a ListUsersReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Lists all users in the AWS account
  #
  # @return [Array<Aws::IAM::Types::User>] An array of user objects
  def list_users
    users = []
    @iam_client.list_users.each_page do |page|
      page.users.each do |user|
        users << user
      end
    end
    users
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing users: #{e.message}")
    []
  end
```

- Para obter detalhes da API, consulte ListUsersa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
```

```
        .await?;
    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListUsersreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listusers( ).
        MESSAGE 'Retrieved user list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing users.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListUsersreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
```

```
import AWSS3


    public func listUsers() async throws -> [MyUserRecord] {
        var userList: [MyUserRecord] = []

        // Use "Paginated" to get all the users.
        // This lets the SDK handle the 'isTruncated' in "ListUsersOutput".
        let input = ListUsersInput()
        let output = client.listUsersPaginated(input: input)

        do {
            for try await page in output {
                guard let users = page.users else {
                    continue
                }
                for user in users {
                    if let id = user.userId, let name = user.userName {
                        userList.append(MyUserRecord(id: id, name: name))
                    }
                }
            }
        }
        catch {
            print("ERROR: listUsers:", dump(error))
            throw error
        }
       return userList
    }
```

- Para obter detalhes da API, consulte ListUsersa referência da API AWS SDK for Swift.

Usar **`ListVirtualMfaDevices`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListVirtualMfaDevices`.

CLI

AWS CLI

Listar dispositivos de MFA virtuais

O comando `list-virtual-mfa-devices` a seguir lista os dispositivos de MFA virtuais configurados na conta atual.

```
aws iam list-virtual-mfa-devices
```

Saída:

```
{
    "VirtualMFADevices": [
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/ExampleMFADevice"
        },
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Fred"
        }
    ]
}
```

Para obter mais informações, consulte Habilitar um dispositivo de autenticação multifator (MFA) virtual no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ListVirtualMfaDevicesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo recupera uma coleção dos dispositivos virtuais de MFA atribuídos aos usuários na AWS conta. A propriedade do **`User`** de cada um é um objeto com detalhes do usuário do IAM ao qual o dispositivo está atribuído.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Saída:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
```

```
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- Para obter detalhes da API, consulte ListVirtualMfaDevicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo recupera uma coleção dos dispositivos virtuais de MFA atribuídos aos usuários na AWS conta. A propriedade do **`User`** de cada um é um objeto com detalhes do usuário do IAM ao qual o dispositivo está atribuído.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Saída:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- Para obter detalhes da API, consulte ListVirtualMfaDevicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutGroupPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutGroupPolicy`.

CLI

AWS CLI

Como adicionar uma política a um grupo

O comando `put-group-policy`, apresentado a seguir, adiciona uma política ao grupo do IAM denominado `Admins`.

```
aws iam put-group-policy \
    --group-name Admins \
    --policy-document file://AdminPolicy.json \
    --policy-name AdminRoot
```

Este comando não produz saída.

A política é definida como um documento JSON no AdminPolicy.jsonarquivo. (O nome e a extensão do arquivo não têm significado.)

Para obter mais informações, consulte Gerenciamento de políticas do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte PutGroupPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma política em linha denominada **`AppTesterPolicy`** e a incorpora no grupo do IAM **`AppTesters`**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem no arquivo **`apptesterpolicy.json`**. Observe que você deve usar o parâmetro **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Para obter detalhes da API, consulte PutGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma política em linha denominada **`AppTesterPolicy`** e a incorpora no grupo do IAM **`AppTesters`**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem no arquivo **`apptesterpolicy.json`**. Observe que você deve usar o parâmetro **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Para obter detalhes da API, consulte PutGroupPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutRolePermissionsBoundary`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutRolePermissionsBoundary`.

CLI

AWS CLI

Exemplo 1: para aplicar um limite de permissões a um perfil do IAM com base em uma política personalizada

O exemplo de `put-role-permissions-boundary` a seguir aplica a política personalizada denominada `intern-boundary` como limite de permissões no perfil do IAM especificado.

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --role-name lambda-application-role
```

Este comando não produz saída.

Exemplo 2: Para aplicar um limite de permissões com base em uma política AWS gerenciada a uma função do IAM

O `put-role-permissions-boundary` exemplo a seguir aplica a `PowerUserAccess` política AWS gerenciada como limite de permissões para a função do IAM especificada.

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --role-name x-account-admin
```

Este comando não produz saída.

Para obter mais informações, consulte Modificar um perfil no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte PutRolePermissionsBoundaryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo mostra como definir o limite de permissões de um perfil do IAM. Você pode definir políticas AWS gerenciadas ou políticas personalizadas como limite de permissão.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Para obter detalhes da API, consulte PutRolePermissionsBoundaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra como definir o limite de permissões de um perfil do IAM. Você pode definir políticas AWS gerenciadas ou políticas personalizadas como limite de permissão.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Para obter detalhes da API, consulte PutRolePermissionsBoundaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutRolePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutRolePolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos de data warehouses de tecnologia sem servidor
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de clusters de data warehouse provisionados
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte PutRolePolicya Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::putRolePolicy(
        const Aws::String &roleName,
        const Aws::String &policyName,
        const Aws::String &policyDocument,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
 iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte PutRolePolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como anexar uma política de permissões a um perfil do IAM

O comando `put-role-policy`, apresentado a seguir, adiciona uma política de permissões ao perfil denominada `Test-Role`.

```
aws iam put-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

Este comando não produz saída.

A política é definida como um documento JSON no AdminPolicy.jsonarquivo. (O nome e a extensão do arquivo não têm significado.)

Para anexar uma política de confiança a um perfil, use o comando `update-assume-role-policy`.

Para obter mais informações, consulte Modificar um perfil no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte PutRolePolicyem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { PutRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const examplePolicyDocument = JSON.stringify({
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "VisualEditor0",
      Effect: "Allow",
      Action: [
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts",
      ],
      Resource: "arn:aws:s3:::amzn-s3-demo-bucket",
    },
    {
      Sid: "VisualEditor1",
      Effect: "Allow",
      Action: [
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:ListJobs",
        "s3:ListMultiRegionAccessPoints",
      ],
      Resource: "*",
    },
  ],
});

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 * @param {string} policyDocument
 */
export const putRolePolicy = async (roleName, policyName, policyDocument) => {
  const command = new PutRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
    PolicyDocument: policyDocument,
  });
```

```
  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte PutRolePolicya Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma política em linha denominada **FedTesterRolePolicy** e a incorpora no perfil do IAM **FedTesterRole**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem do arquivo **FedTesterPolicy.json**. Observe que você deve usar o parâmetro **-Raw** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Para obter detalhes da API, consulte PutRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma política em linha denominada **FedTesterRolePolicy** e a incorpora no perfil do IAM **FedTesterRole**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem do arquivo **FedTesterPolicy.json**. Observe que você deve usar o parâmetro **-Raw** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Para obter detalhes da API, consulte PutRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutUserPermissionsBoundary`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutUserPermissionsBoundary`.

CLI

AWS CLI

Exemplo 1: para aplicar um limite de permissões a um usuário do IAM com base em uma política personalizada

O exemplo de `put-user-permissions-boundary` a seguir aplica uma política personalizada denominada `intern-boundary` como limite de permissões no usuário do IAM especificado.

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --user-name intern
```

Este comando não produz saída.

Exemplo 2: Para aplicar um limite de permissões com base em uma política AWS gerenciada a um usuário do IAM

O `put-user-permissions-boundary` exemplo a seguir aplica a política AWS gerenciada nomeada `PowerUserAccess` como limite de permissões para o usuário do IAM especificado.

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --user-name developer
```

Este comando não produz saída.

Para saber mais, consulte Adicionar e remover permissões de identidade do IAM no AWS Guia do usuário do IAM.

- Para obter detalhes da API, consulte PutUserPermissionsBoundaryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo mostra como definir o limite de permissões do usuário. Você pode definir políticas AWS gerenciadas ou políticas personalizadas como limite de permissão.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Para obter detalhes da API, consulte PutUserPermissionsBoundaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra como definir o limite de permissões do usuário. Você pode definir políticas AWS gerenciadas ou políticas personalizadas como limite de permissão.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Para obter detalhes da API, consulte PutUserPermissionsBoundaryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutUserPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutUserPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como anexar uma política a um usuário do IAM

O comando `put-user-policy`, apresentado a seguir, anexa uma política ao usuário do IAM denominado `Bob`.

```
aws iam put-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

Este comando não produz saída.

A política é definida como um documento JSON no AdminPolicy.jsonarquivo. (O nome e a extensão do arquivo não têm significado.)

Para obter mais informações, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte PutUserPolicyem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
```

```
 IamClient *iam.Client
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

- Para obter detalhes da API, consulte PutUserPolicya Referência AWS SDK para Go da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma política em linha denominada **`EC2AccessPolicy`** e a incorpora no usuário do IAM **`Bob`**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem do arquivo **`EC2AccessPolicy.json`**. Observe que você deve usar o parâmetro **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
 (Get-Content -Raw EC2AccessPolicy.json)
```

- Para obter detalhes da API, consulte PutUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma política em linha denominada **`EC2AccessPolicy`** e a incorpora no usuário do IAM **`Bob`**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem do arquivo **`EC2AccessPolicy.json`**. Observe que você deve usar o parâmetro **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
 (Get-Content -Raw EC2AccessPolicy.json)
```

- Para obter detalhes da API, consulte PutUserPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Creates an inline policy for a specified user.
  # @param username [String] The name of the IAM user.
  # @param policy_name [String] The name of the policy to create.
  # @param policy_document [String] The JSON policy document.
  # @return [Boolean]
  def create_user_policy(username, policy_name, policy_document)
    @iam_client.put_user_policy({
                                  user_name: username,
                                  policy_name: policy_name,
                                  policy_document: policy_document
                                })
    @logger.info("Policy #{policy_name} created for user #{username}.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't create policy #{policy_name} for user #{username}.
 Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    false
  end
```

- Para obter detalhes da API, consulte PutUserPolicya Referência AWS SDK para Ruby da API.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    func putUserPolicy(policyDocument: String, policyName: String, user:
 IAMClientTypes.User) async throws {
        let input = PutUserPolicyInput(
```

```
            policyDocument: policyDocument,
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.putUserPolicy(input: input)
        } catch {
            print("ERROR: putUserPolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte PutUserPolicya referência da API AWS SDK for Swift.

Usar **`RemoveClientIdFromOpenIdConnectProvider`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemoveClientIdFromOpenIdConnectProvider`.

CLI

AWS CLI

Para remover o ID do cliente especificado da lista de IDs de clientes registrados do provedor OpenID Connect do IAM

Este exemplo remove o ID do cliente `My-TestApp-3` da lista de IDs de cliente associados ao provedor OIDC do IAM cujo ARN é `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`.

```
aws iam remove-client-id-from-open-id-connect-provider
    --client-id My-TestApp-3 \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

Este comando não produz saída.

Para obter mais informações, consulte Criar provedores de identidade OpenID Connect (OIDC) no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte RemoveClientIdFromOpenIdConnectProviderem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo remove o ID do cliente **My-TestApp-3** da lista de IDs de cliente associados ao provedor OIDC do IAM cujo ARN é **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- Para obter detalhes da API, consulte RemoveClientIdFromOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove o ID do cliente **My-TestApp-3** da lista de IDs de cliente associados ao provedor OIDC do IAM cujo ARN é **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- Para obter detalhes da API, consulte RemoveClientIdFromOpenIdConnectProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **RemoveRoleFromInstanceProfile** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemoveRoleFromInstanceProfile`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conceitos básicos de streaming gerenciado
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

CLI

AWS CLI

Para remover um cargo de um perfil de instância

O comando `remove-role-from-instance-profile` a seguir remove o perfil denominado `Test-Role` do perfil de instância denominado `ExampleInstanceProfile`.

```
aws iam remove-role-from-instance-profile \
    --instance-profile-name ExampleInstanceProfile \
    --role-name Test-Role
```

Para obter mais informações, consulte Usar perfis de instância no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte RemoveRoleFromInstanceProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o perfil denominado **`MyNewRole`** do perfil de instância do EC2 denominado **`MyNewRole`**. Um perfil de instância criado no console do IAM sempre tem o mesmo nome do perfil, como neste exemplo. Se você os criar na API ou na CLI, eles poderão ter nomes diferentes.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
 MyNewRole -Force
```

- Para obter detalhes da API, consulte RemoveRoleFromInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o perfil denominado **`MyNewRole`** do perfil de instância do EC2 denominado **`MyNewRole`**. Um perfil de instância criado no console do IAM sempre tem o mesmo nome do perfil, como neste exemplo. Se você os criar na API ou na CLI, eles poderão ter nomes diferentes.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
 MyNewRole -Force
```

- Para obter detalhes da API, consulte RemoveRoleFromInstanceProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **RemoveUserFromGroup** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemoveUserFromGroup`.

CLI

AWS CLI

Como remover um usuário de um grupo do IAM

O comando `remove-user-from-group`, apresentado a seguir, remove o usuário denominado `Bob` do grupo do IAM denominado `Admins`.

```
aws iam remove-user-from-group \
    --user-name Bob \
    --group-name Admins
```

Este comando não produz saída.

Para obter mais informações, consulte Adicionar e remover usuários de um grupo de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte RemoveUserFromGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo remove o usuário do IAM **Bob** do grupo **Testers**.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Exemplo 2: este exemplo encontra todos os grupos dos quais a usuária do IAM **Theresa** é membro e, em seguida, remove a **Theresa** desses grupos.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName Theresa -Force }
```

Exemplo 3: este exemplo mostra uma forma alternativa de remover o usuário do IAM **Bob** do grupo **Testers**.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -
GroupName Testers -Force
```

- Para obter detalhes da API, consulte RemoveUserFromGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove o usuário do IAM **Bob** do grupo **Testers**.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Exemplo 2: este exemplo encontra todos os grupos dos quais a usuária do IAM **Theresa** é membro e, em seguida, remove a **Theresa** desses grupos.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName Theresa -Force }
```

Exemplo 3: este exemplo mostra uma forma alternativa de remover o usuário do IAM **Bob** do grupo **Testers**.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -
GroupName Testers -Force
```

- Para obter detalhes da API, consulte RemoveUserFromGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **ResyncMfaDevice** com uma CLI

Os exemplos de código a seguir mostram como usar o `ResyncMfaDevice`.

CLI

AWS CLI

Para sincronizar um dispositivo de MFA

O exemplo de `resync-mfa-device` a seguir sincroniza o dispositivo de MFA associado ao usuário do IAM `Bob` e cujo ARN é `arn:aws:iam::123456789012:mfa/BobsMFADevice` com um programa autenticador que forneceu os dois códigos de autenticação.

```
aws iam resync-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 987654
```

Este comando não produz saída.

Para obter mais informações, consulte Uso de autenticação multifator (MFA) na AWS no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte ResyncMfaDeviceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo sincroniza o dispositivo de MFA associado ao usuário do IAM **`Bob`** e cujo ARN é **`arn:aws:iam::123456789012:mfa/bob`** com um programa autenticador que forneceu os dois códigos de autenticação.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Exemplo 2: este exemplo sincroniza o dispositivo de MFA do IAM associado à usuária do IAM **`Theresa`** com um dispositivo físico que tem o número de série **`ABCD12345678`** e que forneceu os dois códigos de autenticação.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- Para obter detalhes da API, consulte ResyncMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo sincroniza o dispositivo de MFA associado ao usuário do IAM **Bob** e cujo ARN é **arn:aws:iam::123456789012:mfa/bob** com um programa autenticador que forneceu os dois códigos de autenticação.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Exemplo 2: este exemplo sincroniza o dispositivo de MFA do IAM associado à usuária do IAM **Theresa** com um dispositivo físico que tem o número de série **ABCD12345678** e que forneceu os dois códigos de autenticação.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- Para obter detalhes da API, consulte ResyncMfaDeviceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **SetDefaultPolicyVersion**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetDefaultPolicyVersion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Políticas gerenciadas
- Reverter uma versão de política

CLI

AWS CLI

Para definir a versão especificada da política especificada como a versão da política padrão.

Este exemplo define a versão `v2` da política cujo ARN é `arn:aws:iam::123456789012:policy/MyPolicy` como versão ativa padrão.

```
aws iam set-default-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

Para obter mais informações, consulte Políticas e permissões no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte SetDefaultPolicyVersionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo define a versão **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MyPolicy** como versão ativa padrão.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy
 -VersionId v2
```

- Para obter detalhes da API, consulte SetDefaultPolicyVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo define a versão **v2** da política cujo ARN é **arn:aws:iam::123456789012:policy/MyPolicy** como versão ativa padrão.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy
 -VersionId v2
```

- Para obter detalhes da API, consulte SetDefaultPolicyVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->setdefaultpolicyversion(
      iv_policyarn = iv_policy_arn
      iv_versionid = iv_version_id ).
    MESSAGE 'Default policy version set successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy or version does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iaminvalidinputex.
    MESSAGE 'Invalid input provided.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a SetDefaultPolicyVersionreferência da API AWS SDK for SAP ABAP.

Usar **`TagRole`** com uma CLI

Os exemplos de código a seguir mostram como usar o `TagRole`.

CLI

AWS CLI

Adicionar uma tag a um perfil

O comando `tag-role` a seguir adiciona uma tag com o nome do Departamento ao perfil especificado.

```
aws iam tag-role --role-name my-role \
```

```
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

Este comando não produz saída.

Para obter mais informações, consulte Recursos de tags do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte TagRoleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo adiciona uma tag ao perfil no Identity Management Service

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para obter detalhes da API, consulte TagRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo adiciona uma tag ao perfil no Identity Management Service

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para obter detalhes da API, consulte TagRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`TagUser`** com uma CLI

Os exemplos de código a seguir mostram como usar o `TagUser`.

CLI

AWS CLI

Adicionar uma tag a um usuário

O comando `tag-user` a seguir adiciona uma tag com o Departamento associado ao usuário especificado.

```
aws iam tag-user \
    --user-name alice \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

Este comando não produz saída.

Para obter mais informações, consulte Recursos de tags do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte TagUserem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo adiciona uma tag ao usuário no Identity Management Service

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para obter detalhes da API, consulte TagUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo adiciona uma tag ao usuário no Identity Management Service

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para obter detalhes da API, consulte TagUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UntagRole`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UntagRole`.

CLI

AWS CLI

Remover uma tag de um perfil

O comando `untag-role` a seguir remove qualquer tag com o nome de chave 'Department' do perfil especificado.

```
aws iam untag-role \
    --role-name my-role \
    --tag-keys Department
```

Este comando não produz saída.

Para obter mais informações, consulte Recursos de tags do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UntagRoleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo remove a tag da função chamada "MyRoleName" com a chave de tag como “abac”. Para remover várias tags, forneça uma lista de chaves de tag separadas por vírgulas.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Para obter detalhes da API, consulte UntagRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove a tag da função chamada "MyRoleName" com a chave de tag como “abac”. Para remover várias tags, forneça uma lista de chaves de tag separadas por vírgulas.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Para obter detalhes da API, consulte UntagRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UntagUser`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UntagUser`.

CLI

AWS CLI

Remover uma tag de um usuário

O comando `untag-user` a seguir remove qualquer tag com o nome de chave 'Department' do usuário especificado.

```
aws iam untag-user \
    --user-name alice \
    --tag-keys Department
```

Este comando não produz saída.

Para obter mais informações, consulte Recursos de tags do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UntagUserem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo remove a tag do usuário denominado “joe” com a chave de tag como “abac” e “xyzw”. Para remover várias tags, forneça uma lista de chaves de tag separadas por vírgulas.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Para obter detalhes da API, consulte UntagUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove a tag do usuário denominado “joe” com a chave de tag como “abac” e “xyzw”. Para remover várias tags, forneça uma lista de chaves de tag separadas por vírgulas.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Para obter detalhes da API, consulte UntagUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **UpdateAccessKey**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateAccessKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar chaves de acesso

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iam_update_access_key
#
# This function can activate or deactivate an IAM access key for the specified
 IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to update.
#       -a            -- Activate the selected access key.
#       -d            -- Deactivate the selected access key.
#
# Example:
#       # To deactivate the selected access key for IAM user Bob
#       iam_update_access_key -u Bob -k AKIAIOSFODNN7EXAMPLE -d
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
###############################################################################
function iam_update_access_key() {
  local user_name access_key status response
  local option OPTARG # Required to use getopts command in a function.
  local activate_flag=false deactivate_flag=false

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_update_access_key"
    echo "Updates the status of an AWS Identity and Access Management (IAM)
 access key for the specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to update."
    echo "  -a              Activate the access key."
    echo "  -d              Deactivate the access key."
    echo ""
  }

  # Retrieve the calling parameters.
    while getopts "u:k:adh" option; do
      case "${option}" in
        u) user_name="${OPTARG}" ;;
        k) access_key="${OPTARG}" ;;
        a) activate_flag=true ;;
        d) deactivate_flag=true ;;
        h)
          usage
          return 0
          ;;
        \?)
          echo "Invalid parameter"
          usage
          return 1
          ;;
      esac
    done
    export OPTIND=1

   # Validate input parameters
    if [[ -z "$user_name" ]]; then
      errecho "ERROR: You must provide a username with the -u parameter."
      usage
      return 1
    fi
```

```
    if [[ -z "$access_key" ]]; then
      errecho "ERROR: You must provide an access key with the -k parameter."
      usage
      return 1
    fi

    # Ensure that only -a or -d is specified
    if [[ "$activate_flag" == true && "$deactivate_flag" == true ]]; then
      errecho "ERROR: You cannot specify both -a (activate) and -d (deactivate)
at the same time."
      usage
      return 1
    fi

    # If neither -a nor -d is provided, return an error
    if [[ "$activate_flag" == false && "$deactivate_flag" == false ]]; then
      errecho "ERROR: You must specify either -a (activate) or -d (deactivate)."
      usage
      return 1
    fi

    # Determine the status based on the flag
    if [[ "$activate_flag" == true ]]; then
      status="Active"
    elif [[ "$deactivate_flag" == true ]]; then
      status="Inactive"
    fi

    iecho "Parameters:\n"
    iecho "    Username:   $user_name"
    iecho "    Access key: $access_key"
    iecho "    New status: $status"
    iecho ""

    # Update the access key status
    response=$(aws iam update-access-key \
      --user-name "$user_name" \
      --access-key-id "$access_key" \
      --status "$status" 2>&1)

    local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
```

```
      aws_cli_error_log $error_code
      errecho "ERROR: AWS reports update-access-key operation failed.\n$response"
      return 1
    fi

    iecho "update-access-key response: $response"
    iecho

    return 0
}
```

- Para obter detalhes da API, consulte UpdateAccessKeyem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  Aws::IAM::Model::StatusType status,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
```

```
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateAccessKeya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como ativar ou desativar uma chave de acesso para um usuário do IAM

O comando `update-access-key`, apresentado a seguir, desativa a chave de acesso especificada (ID da chave de acesso e chave de acesso secreta) para o usuário do IAM denominado `Bob`.

```
aws iam update-access-key \
    --access-key-id AKIAIOSFODNN7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

Este comando não produz saída.

Desativar a chave significa que ela não pode ser usada para acesso programático a. AWS No entanto, a chave continua disponível e pode ser ativada novamente.

Para obter mais informações, consulte Gerenciamento de chaves de acesso de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateAccessKeyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

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

    private static StatusType statusType;

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

                Usage:
                    <username> <accessId> <status>\s

                Where:
                    username - The name of the user whose key you want to update.
\s
                    accessId - The access key ID of the secret access key you
 want to update.\s
                    status - The status you want to assign to the secret access
 key.\s
```

```
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
        iam.close();
    }

    public static void updateKey(IamClient iam, String username, String accessId,
String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                    .accessKeyId(accessId)
                    .userName(username)
                    .status(statusType)
                    .build();

            iam.updateAccessKey(request);
            System.out.printf("Successfully updated the status of access key %s
to" +
                    "status %s for user %s", accessId, status, username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte UpdateAccessKeya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize a chave de acesso.

```
import {
  UpdateAccessKeyCommand,
  IAMClient,
  StatusType,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const updateAccessKey = (userName, accessKeyId) => {
  const command = new UpdateAccessKeyCommand({
    AccessKeyId: accessKeyId,
    Status: StatusType.Inactive,
    UserName: userName,
  });

  return client.send(command);
```

```
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateAccessKeya Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  Status: "Active",
  UserName: "USER_NAME",
};

iam.updateAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

- Para obter detalhes da API, consulte UpdateAccessKeya Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo altera para **`Inactive`** o status da chave de acesso **`AKIAIOSFODNN7EXAMPLE`** do usuário do IAM denominado **`Bob`**.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status
 Inactive
```

- Para obter detalhes da API, consulte UpdateAccessKeyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo altera para **`Inactive`** o status da chave de acesso **`AKIAIOSFODNN7EXAMPLE`** do usuário do IAM denominado **`Bob`**.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status
 Inactive
```

- Para obter detalhes da API, consulte UpdateAccessKeyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def update_key(user_name, key_id, activate):
```

```
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
key_id
        )
        raise
```

- Para obter detalhes da API, consulte a UpdateAccessKeyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        lo_iam->updateaccesskey(
          iv_accesskeyid = iv_access_key_id
          iv_status = iv_status
          iv_username = iv_user_name ).
        MESSAGE 'Access key updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Access key or user does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateAccessKeyreferência da API AWS SDK for SAP ABAP.

Usar **`UpdateAccountPasswordPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateAccountPasswordPolicy`.

CLI

AWS CLI

Definir ou alterar a política de senha da conta atual

O comando `update-account-password-policy` a seguir define a política de senha para exigir um mínimo de oito caracteres e um ou mais números na senha.

```
aws iam update-account-password-policy \
    --minimum-password-length 8 \
    --require-numbers
```

Este comando não produz saída.

As alterações na política de senha de uma conta afetam todas as novas senhas criadas para usuários do IAM na conta. As alterações na política de senha não afetam as senhas existentes.

Para obter mais informações, consulte Definição de uma política de senhas de contas para usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateAccountPasswordPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza a política de senha da conta com as configurações especificadas. Observe que quaisquer parâmetros que não estejam incluídos no comando não são modificados. Em vez disso, eles são redefinidos para os valores padrão.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- Para obter detalhes da API, consulte UpdateAccountPasswordPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a política de senha da conta com as configurações especificadas. Observe que quaisquer parâmetros que não estejam incluídos no comando não são modificados. Em vez disso, eles são redefinidos para os valores padrão.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- Para obter detalhes da API, consulte UpdateAccountPasswordPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateAssumeRolePolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateAssumeRolePolicy`.

CLI

AWS CLI

Atualizar a política de confiança de um perfil do IAM

O comando `update-assume-role-policy` a seguir atualiza a política de confiança do perfil denominado `Test-Role`.

```
aws iam update-assume-role-policy \
    --role-name Test-Role \
    --policy-document file://Test-Role-Trust-Policy.json
```

Este comando não produz saída.

A política de confiança é definida como um documento JSON no Test-Role-Trust-Policy.jsonarquivo. (O nome e a extensão do arquivo não têm significado.) A política de confiança deve especificar uma entidade principal.

Para atualizar a política de permissões de um perfil, use o comando `put-role-policy`.

Para obter mais informações, consulte Criar perfis do IAM no AWS Guia do usuário do IAM.

- Para obter detalhes da API, consulte UpdateAssumeRolePolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza o perfil do IAM denominado **`ClientRole`** com uma nova política de confiança, cujo conteúdo vem do arquivo **`ClientRolePolicy.json`**. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- Para obter detalhes da API, consulte UpdateAssumeRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza o perfil do IAM denominado **`ClientRole`** com uma nova política de confiança, cujo conteúdo vem do arquivo **`ClientRolePolicy.json`**. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- Para obter detalhes da API, consulte UpdateAssumeRolePolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **UpdateGroup** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateGroup`.

CLI

AWS CLI

Para renomear um grupo do IAM

O comando `update-group` a seguir altera o nome do grupo do IAM `Test` para `Test-1`.

```
aws iam update-group \
    --group-name Test \
    --new-group-name Test-1
```

Este comando não produz saída.

Para obter mais informações, consulte Renomeação de um grupo de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo renomeia o grupo do IAM **`Testers`** para **`AppTesters`**.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Exemplo 2: este exemplo altera o caminho do grupo do IAM **`AppTesters`** para **`/Org1/Org2/`**. Isso altera o ARN do grupo para **`arn:aws:iam::123456789012:group/Org1/Org2/AppTesters`**.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Para obter detalhes da API, consulte UpdateGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo renomeia o grupo do IAM **`Testers`** para **`AppTesters`**.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Exemplo 2: este exemplo altera o caminho do grupo do IAM **`AppTesters`** para **`/Org1/Org2/`**. Isso altera o ARN do grupo para **`arn:aws:iam::123456789012:group/Org1/Org2/AppTesters`**.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Para obter detalhes da API, consulte UpdateGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateLoginProfile`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateLoginProfile`.

CLI

AWS CLI

Atualizar a senha de um usuário do IAM

O comando `update-login-profile` a seguir cria uma senha para o usuário do IAM chamado Bob.

```
aws iam update-login-profile \
    --user-name Bob \
    --password <password>
```

Este comando não produz saída.

Para definir uma política de senha da conta, use o comando `update-account-password-policy`. Se a nova senha violar a política de senha da conta, o comando retornará um erro `PasswordPolicyViolation`.

Se a política de senha da conta permitir, os usuários do IAM poderão alterar suas próprias senhas usando o comando `change-password`.

Armazene a senha em um lugar seguro. Se a senha for perdida, não será possível recuperá-la e você deverá criar uma nova usando o comando `create-login-profile`.

Para obter mais informações, consulte Gerenciamento de senhas de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateLoginProfileem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo define uma nova senha temporária para o usuário **Bob** do IAM e exige que a pessoa altere a senha na próxima vez que fizer login.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -
PasswordResetRequired $true
```

- Para obter detalhes da API, consulte UpdateLoginProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo define uma nova senha temporária para o usuário **Bob** do IAM e exige que a pessoa altere a senha na próxima vez que fizer login.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -
PasswordResetRequired $true
```

- Para obter detalhes da API, consulte UpdateLoginProfileem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateOpenIdConnectProviderThumbprint`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateOpenIdConnectProviderThumbprint`.

CLI

AWS CLI

Para substituir a lista existente de impressões digitais do certificado de servidor por uma nova

Este exemplo atualiza a lista de impressões digitais do certificado do provedor OIDC cujo ARN é `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`, a fim de usar uma nova impressão digital.

```
aws iam update-open-id-connect-provider-thumbprint \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com \
    --thumbprint-list 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

Se a descoberta do seu provedor OIDC e os endpoints do JWKS usarem certificados diferentes, inclua impressões digitais de ambos os endpoints no valor `--thumbprint-list`.

Este comando não produz saída.

Para obter mais informações, consulte Criar provedores de identidade OpenID Connect (OIDC) no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateOpenIdConnectProviderThumbprintem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza a lista de impressões digitais do certificado do provedor OIDC cujo ARN é **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`**, a fim de usar uma nova impressão digital. O provedor OIDC compartilha o novo valor quando o certificado associado ao provedor é alterado.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Para obter detalhes da API, consulte UpdateOpenIdConnectProviderThumbprintem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a lista de impressões digitais do certificado do provedor OIDC cujo ARN é **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`**, a fim de usar uma nova impressão digital. O provedor OIDC compartilha o novo valor quando o certificado associado ao provedor é alterado.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Para obter detalhes da API, consulte UpdateOpenIdConnectProviderThumbprintem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateRole`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateRole`.

CLI

AWS CLI

Alterar a descrição ou a duração da sessão de um perfil do IAM

O comando `update-role` a seguir altera a descrição do perfil do IAM `production-role` para `Main production role` e define a duração máxima da sessão como 12 horas.

```
aws iam update-role \
    --role-name production-role \
    --description 'Main production role' \
    --max-session-duration 43200
```

Este comando não produz saída.

Para obter mais informações, consulte Modificar um perfil no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateRoleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza a descrição do perfil e o valor máximo da duração da sessão (em segundos) para o qual a sessão de um perfil pode ser solicitada.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Para obter detalhes da API, consulte UpdateRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a descrição do perfil e o valor máximo da duração da sessão (em segundos) para o qual a sessão de um perfil pode ser solicitada.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Para obter detalhes da API, consulte UpdateRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateRoleDescription`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateRoleDescription`.

CLI

AWS CLI

Alterar a descrição de um perfil do IAM

O comando `update-role` a seguir altera a descrição do perfil do IAM `production-role` para `Main production role`.

```
aws iam update-role-description \
    --role-name production-role \
    --description 'Main production role'
```

Saída:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "production-role",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/production-role",
        "CreateDate": "2017-12-06T17:16:37+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole",
                    "Condition": {}
                }
            ]
        },
        "Description": "Main production role"
    }
}
```

Para obter mais informações, consulte Modificar um perfil no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateRoleDescriptionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza a descrição de um perfil do IAM na sua conta.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Para obter detalhes da API, consulte UpdateRoleDescriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a descrição de um perfil do IAM na sua conta.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Para obter detalhes da API, consulte UpdateRoleDescriptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateSamlProvider`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateSamlProvider`.

CLI

AWS CLI

Atualizar o documento de metadados de um provedor SAML existente

Este exemplo atualiza o provedor SAML no IAM cujo ARN é `arn:aws:iam::123456789012:saml-provider/SAMLADFS` com um novo documento de metadados SAML do arquivo `SAMLMetaData.xml`.

```
aws iam update-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

Saída:

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/SAMLADFS"
}
```

Para obter mais informações, consulte Criação de provedores de identidade SAML do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateSamlProviderem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza o provedor SAML no IAM cujo ARN é **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** com um novo documento de metadados SAML do arquivo **`SAMLMetaData.xml`**. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Para obter detalhes da API, consulte UpdateSamlProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza o provedor SAML no IAM cujo ARN é **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** com um novo documento de metadados SAML do arquivo **`SAMLMetaData.xml`**. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Para obter detalhes da API, consulte UpdateSamlProviderem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`UpdateServerCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateServerCertificate`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String
 &currentCertificateName,
                                          const Aws::String &newCertificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName <<
 ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                      << "' not found." << std::endl;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte UpdateServerCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para alterar o caminho ou o nome de um certificado de servidor em sua AWS conta

O comando `update-server-certificate`, apresentado a seguir, altera o nome do certificado de `myServerCertificate` para `myUpdatedServerCertificate`. Ele também altera o caminho para que `/cloudfront/` possa ser acessado pelo CloudFront serviço da Amazon. Este comando não produz saída. É possível visualizar os resultados da atualização ao executar o comando `list-server-certificates`.

```
aws-iam update-server-certificate \
    --server-certificate-name myServerCertificate \
    --new-server-certificate-name myUpdatedServerCertificate \
    --new-path /cloudfront/
```

Este comando não produz saída.

Para obter mais informações, consulte Gerenciar certificados de servidor no IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateServerCertificateem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize um certificado do servidor.

```
import { UpdateServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} currentName
 * @param {string} newName
 */
export const updateServerCertificate = (currentName, newName) => {
  const command = new UpdateServerCertificateCommand({
    ServerCertificateName: currentName,
    NewServerCertificateName: newName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateServerCertificatea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  ServerCertificateName: "CERTIFICATE_NAME",
  NewServerCertificateName: "NEW_CERTIFICATE_NAME",
};
```

```
iam.updateServerCertificate(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateServerCertificatea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo renomeia o certificado denominado **`MyServerCertificate`** para **`MyRenamedServerCertificate`**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Exemplo 2: Este exemplo move o certificado nomeado **`MyServerCertificate`** para o caminho/Org1/Org2/. Isso altera o ARN do recurso para **`arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate`**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- Para obter detalhes da API, consulte UpdateServerCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo renomeia o certificado denominado **`MyServerCertificate`** para **`MyRenamedServerCertificate`**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Exemplo 2: Este exemplo move o certificado nomeado **`MyServerCertificate`** para o caminho/Org1/Org2/. Isso altera o ARN do recurso para **`arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate`**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- Para obter detalhes da API, consulte UpdateServerCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, atualizar e excluir certificados de servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
```

```
                                                server_certificate_name: name,
                                                certificate_body: certificate_body,
                                                private_key: private_key
                                              })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
```

```
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte UpdateServerCertificatea Referência AWS SDK para Ruby da API.

Usar **`UpdateSigningCertificate`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateSigningCertificate`.

CLI

AWS CLI

Ativar ou desativar um certificado de assinatura de um usuário do IAM

O comando `update-signing-certificate` a seguir desativa o certificado de assinatura especificado do usuário do IAM chamado Bob.

```
aws iam update-signing-certificate \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

Para obter o ID de um certificado de assinatura, use o comando `list-signing-certificates`.

Para obter mais informações, consulte Manage signing certificates no Guia do usuário do Amazon EC2.

- Para obter detalhes da API, consulte UpdateSigningCertificateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza o certificado associado ao usuário do IAM chamado **Bob** e cujo ID do certificado é **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** para marcá-lo como inativo.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- Para obter detalhes da API, consulte UpdateSigningCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza o certificado associado ao usuário do IAM chamado **Bob** e cujo ID do certificado é **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** para marcá-lo como inativo.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- Para obter detalhes da API, consulte UpdateSigningCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **UpdateUser**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateUser`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar usuários somente leitura e leitura/gravação usando

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                  " successfully updated with new user name " << newUserName <<
                  std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName
 <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateUsera Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como alterar o nome de um usuário do IAM

O comando `update-user`, apresentado a seguir, altera o nome do usuário do IAM de `Bob` para `Robert`.

```
aws iam update-user \
    --user-name Bob \
    --new-user-name Robert
```

Este comando não produz saída.

Para obter mais informações, consulte Renomeação de um grupo de usuários do IAM no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte UpdateUserem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <curName> <newName>\s

                Where:
                    curName - The current user name.\s
                    newName - An updated user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String
 newName) {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s",
 newName);
```

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

- Para obter detalhes da API, consulte UpdateUsera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize o usuário.

```
import { UpdateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentUserName
 * @param {string} newUserName
 */
export const updateUser = (currentUserName, newUserName) => {
  const command = new UpdateUserCommand({
    UserName: currentUserName,
    NewUserName: newUserName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateUsera Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3],
};

iam.updateUser(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateUsera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- Para obter detalhes da API, consulte a UpdateUserreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo renomeia o usuário do IAM **`Bob`** para **`Robert`**.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Exemplo 2: este exemplo altera o caminho do usuário do IAM **`Bob`** para **`/Org1/Org2/`**, o que efetivamente altera o ARN do usuário para **`arn:aws:iam::123456789012:user/Org1/Org2/bob`**.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Para obter detalhes da API, consulte UpdateUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo renomeia o usuário do IAM **Bob** para **Robert**.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Exemplo 2: este exemplo altera o caminho do usuário do IAM **Bob** para **/Org1/Org2/**, o que efetivamente altera o ARN do usuário para **arn:aws:iam::123456789012:user/Org1/Org2/bob**.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Para obter detalhes da API, consulte UpdateUserem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
```

```
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

- Para obter detalhes da API, consulte a UpdateUserReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Updates an IAM user's name
 #
 # @param current_name [String] The current name of the user
 # @param new_name [String] The new name of the user
 def update_user_name(current_name, new_name)
   @iam_client.update_user(user_name: current_name, new_user_name: new_name)
   true
 rescue StandardError => e
   @logger.error("Error updating user name from '#{current_name}' to
'#{new_name}': #{e.message}")
   false
 end
```

- Para obter detalhes da API, consulte UpdateUsera Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->updateuser(
      iv_username = iv_user_name
      iv_newusername = iv_new_user_name ).
    MESSAGE 'User updated successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamentityalrdyexex.
    MESSAGE 'New user name already exists.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateUserreferência da API AWS SDK for SAP ABAP.

Use **`UploadServerCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UploadServerCertificate`.

CLI

AWS CLI

Para fazer upload de um certificado de servidor para sua AWS conta

O comando upload-server-certificate a seguir carrega um certificado de servidor em sua conta. AWS Neste exemplo, o certificado está no arquivo `public_key_cert_file.pem`, a chave privada associada está no arquivo `my_private_key.pem` e a cadeia de certificados fornecida pela autoridade de certificação (CA) está no arquivo `my_certificate_chain_file.pem`. Quando o arquivo terminar de ser carregado, ele estará disponível com o nome my ServerCertificate. Os parâmetros que começam com

`file://` informam ao comando para ler o conteúdo do arquivo e usá-lo como valor do parâmetro em vez do próprio nome do arquivo.

```
aws iam upload-server-certificate \
    --server-certificate-name myServerCertificate \
    --certificate-body file://public_key_cert_file.pem \
    --private-key file://my_private_key.pem \
    --certificate-chain file://my_certificate_chain_file.pem
```

Saída:

```
{
    "ServerCertificateMetadata": {
        "Path": "/",
        "ServerCertificateName": "myServerCertificate",
        "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
        "Arn": "arn:aws:iam::1234567989012:server-certificate/
myServerCertificate",
        "UploadDate": "2019-04-22T21:13:44+00:00",
        "Expiration": "2019-10-15T22:23:16+00:00"
    }
}
```

Para obter mais informações, consulte Creating, Uploading, and Deleting Server Certificates no guia Using IAM.

- Para obter detalhes da API, consulte UploadServerCertificateem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { UploadServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
```

```
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import * as path from "node:path";

const client = new IAMClient({});

const certMessage = `Generate a certificate and key with the following command,
 or the equivalent for your system.

openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout example.key -out example.crt -subj "/CN=example.com" \
-addext "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
`;

const getCertAndKey = () => {
  try {
    const cert = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.crt"),
    );
    const key = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.key"),
    );
    return { cert, key };
  } catch (err) {
    if (err.code === "ENOENT") {
      throw new Error(
        `Certificate and/or private key not found. ${certMessage}`,
      );
    }

    throw err;
  }
};

/**
 *
 * @param {string} certificateName
 */
export const uploadServerCertificate = (certificateName) => {
  const { cert, key } = getCertAndKey();
  const command = new UploadServerCertificateCommand({
    ServerCertificateName: certificateName,
    CertificateBody: cert.toString(),
    PrivateKey: key.toString(),
  });
```

```
  return client.send(command);
};
```

- Para obter detalhes da API, consulte UploadServerCertificate a Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo faz upload de um novo certificado de servidor na conta do IAM. Os arquivos contendo o corpo do certificado, a chave privada e (opcionalmente) a cadeia de certificação devem ser codificados em PEM. Observe que os parâmetros exigem o conteúdo real dos arquivos em vez dos nomes deles. Você deve usar o parâmetro switch **-Raw** para processar com êxito o conteúdo do arquivo.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes da API, consulte UploadServerCertificate em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo faz upload de um novo certificado de servidor na conta do IAM. Os arquivos contendo o corpo do certificado, a chave privada e (opcionalmente) a cadeia de certificação devem ser codificados em PEM. Observe que os parâmetros exigem o conteúdo real dos arquivos em vez dos nomes deles. Você deve usar o parâmetro switch **-Raw** para processar com êxito o conteúdo do arquivo.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes da API, consulte UploadServerCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UploadSigningCertificate`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UploadSigningCertificate`.

CLI

AWS CLI

Fazer upload de um certificado de assinatura de um usuário do IAM

O comando `upload-signing-certificate` a seguir faz upload de um certificado de assinatura do usuário do IAM chamado Bob.

```
aws iam upload-signing-certificate \
    --user-name Bob \
    --certificate-body file://certificate.pem
```

Saída:

```
{
    "Certificate": {
        "UserName": "Bob",
        "Status": "Active",
        "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-body>-----END
 CERTIFICATE-----",
        "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
```

```
        "UploadDate": "2013-06-06T21:40:08.121Z"
    }
}
```

O certificado está em um arquivo denominado certificate.pem no formato PEM.

Para obter mais informações, consulte Creating and Uploading a User Signing Certificate no guia Uso do IAM.

- Para obter detalhes da API, consulte UploadSigningCertificateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo carrega um novo certificado de X.509 assinatura e o associa ao usuário do IAM chamado. **`Bob`** O arquivo que contém o corpo do certificado é codificado em PEM. O parâmetro **`CertificateBody`** exige o conteúdo real do arquivo de certificado em vez do nome do arquivo. Você deve usar o parâmetro switch **`-Raw`** para processar o arquivo com êxito.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Saída:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
```

```
                    YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                    21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                    rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                    nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                    NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                    -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Para obter detalhes da API, consulte UploadSigningCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo carrega um novo certificado de X.509 assinatura e o associa ao usuário do IAM chamado. **Bob** O arquivo que contém o corpo do certificado é codificado em PEM. O parâmetro **CertificateBody** exige o conteúdo real do arquivo de certificado em vez do nome do arquivo. Você deve usar o parâmetro switch **-Raw** para processar o arquivo com êxito.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Saída:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
```

```
 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Para obter detalhes da API, consulte UploadSigningCertificateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para o uso do IAM AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no IAM com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções dentro do IAM ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie e gerencie um serviço resiliente usando um AWS SDK
- Configurar a conectividade do serviço de contêiner
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Criar uma API REST com integração de proxy de função
- Crie usuários do IAM somente para leitura e leitura e gravação usando um AWS SDK
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Como criar um espaço de trabalho de monitoramento gerenciado
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de arquivos de atributos de machine learning
- Conceitos básicos de clusters gerenciados de kubernetes
- Conceitos básicos de streaming gerenciado
- Conceitos básicos de clusters de data warehouse provisionados
- Conceitos básicos da orquestração de fluxo de trabalaho
- Gerencie as chaves de acesso do IAM usando um AWS SDK
- Gerencie políticas do IAM usando um AWS SDK
- Gerencie as funções do IAM usando um AWS SDK
- Gerencie sua conta do IAM usando um AWS SDK
- Como mover segredos codificados para o armazenamento seguro de segredos
- Permite que AWS Compute Optimizer Recurso de automação para aplicar ações recomendadas
- Política para permitir a automação em toda a sua organização
- Política para permitir a automação na sua conta
- Política para conceder acesso total ao Compute Optimizer Automation para uma conta gerencial de uma organização
- Política para conceder acesso total à Automação do Compute Optimizer para uso autônomo AWS contas

- Política para conceder acesso somente leitura ao Compute Optimizer Automation para uma conta gerencial de uma organização
- Política para conceder acesso somente de leitura à Automação do Compute Optimizer para uso autônomo AWS contas
- Política para conceder permissões de perfil vinculado ao serviço para o Compute Optimization Automation
- Reverta uma versão da política do IAM usando um AWS SDK
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Configure o controle de Attribute-Based acesso para o DynamoDB usando AWS Command Line Interface v2 da
- Como configurar o gerenciamento de sistemas
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor
- Trabalhe com o DynamoDB Streams e use Time-to-Live AWS Command Line Interface v2 da
- Trabalhe com a API IAM Policy Builder usando um AWS SDK

Crie e gerencie um serviço resiliente usando um AWS SDK

Os exemplos de código a seguir mostram como criar um serviço da Web com carga balanceada que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de saúde atualizando AWS Systems Manager os parâmetros.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
```

```
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
```

```
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
```

```
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
```

```
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");
```

```
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
```

```
                            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
```

```
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");
```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
```

```
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
```

```
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
```

```
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
```

```
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
 List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
```

```
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
```

```
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
```

```
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
```

```
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
```

```
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
  calling computer.
```

```
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                       "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
```

```
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
```

```
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
```

```
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
```

```
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
```

```
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
```

```
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
```

```
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
```

```
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
```

```
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
```

```
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
```

```
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs

- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";
```

```
    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your
 account, we can clean up all AWS resources
                that were created for this demo.
```

```
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
```

```
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

       System.out.println("Press Enter when you're ready.");
       in.nextLine();
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Creating and populating a DynamoDB table named " +
tableName);
       Database database = new Database();
       database.createTable(tableName, fileName);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("""
               Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
               This script starts a Python web server defined in the `server.py`
script. The web server
               listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
               For demo purposes, this server is run as the root user. In
production, the best practice is to
               run a web server, such as Apache, with least-privileged
credentials.

               The template also defines an IAM policy that each instance uses
to assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
```

```
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
```

```
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
       autoScaler.terminateInstance(badInstanceId);

       System.out.println("""
               Even while the instance is terminating and the new instance is
starting, sending a GET
               request to the web service continues to get a successful
recommendation response because
               the load balancer routes requests to the healthy instances. After
the replacement instance
               starts and reports as healthy, it is included in the load
balancing rotation.
               Note that terminating and replacing an instance typically takes
several minutes, during which time you
               can see the changing health check status until the new instance
is running and healthy.
               """);

       demoChoices(loadBalancer);
       System.out.println(
               "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       demoChoices(loadBalancer);
       paramHelper.reset();
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
```

```
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());
```

```
                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
 balancer targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
```

```
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
```

```
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
```

```
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();
```

```
            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
```

```
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
```

```
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

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

```
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }
```

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

```
   // Verify this computer can successfully send a GET request to the load
balancer
   // endpoint.
   public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
       boolean success = false;
       int retries = 3;
       CloseableHttpClient httpClient = HttpClients.createDefault();

       // Create an HTTP GET request to the ELB.
       HttpGet httpGet = new HttpGet("http://" + elbDnsName);
       try {
           while ((!success) && (retries > 0)) {
               // Execute the request and get the response.
               HttpResponse response = httpClient.execute(httpGet);
               int statusCode = response.getStatusLine().getStatusCode();
               System.out.println("HTTP Status Code: " + statusCode);
               if (statusCode == 200) {
                   success = true;
               } else {
                   retries--;
                   System.out.println("Got connection error from load balancer
endpoint, retrying...");
                   TimeUnit.SECONDS.sleep(15);
               }
           }

       } catch (org.apache.http.conn.HttpHostConnectException e) {
           System.out.println(e.getMessage());
       }

       System.out.println("Status.." + success);
       return success;
   }

   /*
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
```

```
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
```

```
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
        return false;
```

```
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();
```

```
            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
```

```
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
```

```
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
```

```
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Criar etapas para implantar todos os recursos.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
```

```
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
```

```
    }),
    new ScenarioAction(
      "handleConfirmDeployment",
      (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
      "creatingTable",
      MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
      const client = new DynamoDBClient({});
      await client.send(
        new CreateTableCommand({
          TableName: NAMES.tableName,
          ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
          },
          AttributeDefinitions: [
            {
              AttributeName: "MediaType",
              AttributeType: "S",
            },
            {
              AttributeName: "ItemId",
              AttributeType: "N",
            },
          ],
          KeySchema: [
            {
              AttributeName: "MediaType",
              KeyType: "HASH",
            },
            {
              AttributeName: "ItemId",
              KeyType: "RANGE",
            },
          ],
        }),
      );
      await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
    }),
    new ScenarioOutput(
      "createdTable",
```

```
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
```

```
    new ScenarioOutput(
      "createdKeyPair",
      MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
    ),
    new ScenarioOutput(
      "creatingInstancePolicy",
      MESSAGES.creatingInstancePolicy.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      ),
    ),
    new ScenarioAction("createInstancePolicy", async (state) => {
      const client = new IAMClient({});
      const {
        Policy: { Arn },
      } = await client.send(
        new CreatePolicyCommand({
          PolicyName: NAMES.instancePolicyName,
          PolicyDocument: readFileSync(
            join(RESOURCES_PATH, "instance_policy.json"),
          ),
        }),
      );
      state.instancePolicyArn = Arn;
    }),
    new ScenarioOutput("createdInstancePolicy", (state) =>
      MESSAGES.createdInstancePolicy
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
    ),
    new ScenarioOutput(
      "creatingInstanceRole",
      MESSAGES.creatingInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      ),
    ),
    new ScenarioAction("createInstanceRole", () => {
      const client = new IAMClient({});
      return client.send(
        new CreateRoleCommand({
          RoleName: NAMES.instanceRoleName,
          AssumeRolePolicyDocument: readFileSync(
            join(ROOT, "assume-role-policy.json"),
```

```
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
```

```
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
```

```
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
```

```
        LaunchTemplate: {
          LaunchTemplateName: NAMES.launchTemplateName,
          Version: "$Default",
        },
        MinSize: 3,
        MaxSize: 3,
      }),
    ),
  );
}),
new ScenarioOutput(
  "createdAutoScalingGroup",
  /**
   * @param {{ availabilityZoneNames: string[] }} state
   */
  (state) =>
    MESSAGES.createdAutoScalingGroup
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
      .replace(
        "${AVAILABILITY_ZONE_NAMES}",
        state.availabilityZoneNames.join(", "),
      ),
),
new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
  type: "confirm",
}),
new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
new ScenarioAction("getVpc", async (state) => {
  const client = new EC2Client({});
  const { Vpcs } = await client.send(
    new DescribeVpcsCommand({
      Filters: [{ Name: "is-default", Values: ["true"] }],
    }),
  );
  state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
  MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
  const client = new EC2Client({});
  const { Subnets } = await client.send(
```

```
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
```

```
    }),
    new ScenarioOutput(
      "createdLoadBalancerTargetGroup",
      MESSAGES.createdLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      ),
    ),
    new ScenarioOutput(
      "creatingLoadBalancer",
      MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
    ),
    new ScenarioAction("createLoadBalancer", async (state) => {
      const client = new ElasticLoadBalancingV2Client({});
      const { LoadBalancers } = await client.send(
        new CreateLoadBalancerCommand({
          Name: NAMES.loadBalancerName,
          Subnets: state.subnets,
        }),
      );
      state.loadBalancerDns = LoadBalancers[0].DNSName;
      state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
      await waitUntilLoadBalancerAvailable(
        { client },
        { Names: [NAMES.loadBalancerName] },
      );
    }),
    new ScenarioOutput("createdLoadBalancer", (state) =>
      MESSAGES.createdLoadBalancer
        .replace("${LB_NAME}", NAMES.loadBalancerName)
        .replace("${DNS_NAME}", state.loadBalancerDns),
    ),
    new ScenarioOutput(
      "creatingListener",
      MESSAGES.creatingLoadBalancerListener
        .replace("${LB_NAME}", NAMES.loadBalancerName)
        .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
    ),
    new ScenarioAction("createListener", async (state) => {
      const client = new ElasticLoadBalancingV2Client({});
      const { Listeners } = await client.send(
        new CreateListenerCommand({
          LoadBalancerArn: state.loadBalancerArn,
          Protocol: state.targetGroupProtocol,
```

```
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
```

```
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
```

```
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
```

```
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Criar etapas para executar a demonstração.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
```

```
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
```

```
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
```

```
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
```

```
    ...statusSteps,
    new ScenarioInput(
      "brokenDependencyConfirmation",
      MESSAGES.demoBrokenDependencyConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("brokenDependency", async (state) => {
      if (!state.brokenDependencyConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        state.badTableName = `fake-table-${Date.now()}`;
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmTableNameKey,
            Value: state.badTableName,
            Overwrite: true,
            Type: "String",
          }),
        );
      }
    }),
    new ScenarioOutput("testBrokenDependency", (state) =>
      MESSAGES.demoTestBrokenDependency.replace(
        "${TABLE_NAME}",
        state.badTableName,
      ),
    ),
    ...statusSteps,
    new ScenarioInput(
      "staticResponseConfirmation",
      MESSAGES.demoStaticResponseConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("staticResponse", async (state) => {
      if (!state.staticResponseConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmFailureResponseKey,
            Value: "static",
            Overwrite: true,
```

```
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
```

```
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
        InstanceIds: [state.targetInstance.InstanceId],
        DocumentName: "AWS-RunShellScript",
        Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
```

```
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
```

```
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
```

```
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
```

```
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
```

```
    }),
  );

  return InstanceProfile;
}
```

Criar etapas para destruir todos os recursos.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
```

```
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
```

```
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
```

```
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
```

```
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
```

```
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
```

```
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
```

```
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
```

```
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
```

```
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
```

```
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
```

```
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
```

```
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
```

```
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
 DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
```

```
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)
```

```
        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
 from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
```

```
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
```

```
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()
```

```
        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()
```

```
        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
```

```
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
```

```
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
```

```
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
 definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
 {policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
```

```
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
```

```
        :param aws_managed_policies: A tuple of AWS-managed policy names to
attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
{role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
```

```
                                        use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn
```

```
    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.
```

```
        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
```

```
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
```

```
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
```

```
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
```

```
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
```

```
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
```

```
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
```

```
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
```

```
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
```

```
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
```

```
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
 "
                    "Ensure that your AWS IAM user or role has the correct
 permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
 parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

```
    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
```

```
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
```

```
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
```

```
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
 {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
```

```
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
```

```
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
```

```
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.
```

```
        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer
```

```
    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
```

```
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
```

```
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
```

```
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
 parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
```

```
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
```

```
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Crie uma classe que envolva as ações do Systems Manager.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
```

```
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer

- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
```

```
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
```

```
    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"
```

```
    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true
```

```
    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
```

```
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
```

```
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."
```

```
    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"
```

```
    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
```

```
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"
```

```
    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi
```

```
    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
```

```
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
```

```
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups
```

```
    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
```

```
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNamespaces
  - ListServices
  - ListTaskDefinitions
  - ListTasks
  - RegisterTaskDefinition

- UpdateService
- Wait

Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um cluster.
- Criar uma definição de tarefa.
- Criar o serviço.
- Fazer a limpeza.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"
```

```
# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "==========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "==========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
```

```
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
```

```
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
```

```
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."
```

```
        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
'$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"
```

```
            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "============================================"
        echo "CLEANUP COMPLETED"
        echo "============================================"
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
```

```
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"
```

```
CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
```

```
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
```

```
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
```

```
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"
```

```
echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
```

```
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole

- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeNetworkInterfaces
- DescribeSecurityGroups
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Criar uma API REST com integração de proxy de função

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para execução do Lambda.
- Criar e implantar uma função do Lambda.
- Criar uma API REST
- Configurar a integração de proxy do Lambda.
- Implantar e testar a API
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
```

```
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
```

```
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
```

```
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"
```

```
# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2
```

```
aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Crie usuários do IAM somente para leitura e leitura e gravação usando um AWS SDK

Os exemplos de código a seguir mostram como criar usuários e anexar políticas a eles.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Criar dois usuários do IAM.
- Anexe uma política para um usuário obter e colocar objetos em um bucket do Amazon S3.
- Anexar uma política para o segundo usuário para obter objetos do bucket.
- Obter outras permissões para o bucket com base nas credenciais do usuário.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações do usuário do IAM.

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.
```

```
    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user



def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user



def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
```

```
        logger.exception("Couldn't get users.")
        raise
    else:
        return users



def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise



def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise



def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.
```

```
    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

Crie a funções que envolvam ações de política do IAM.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
```

```
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

Crie funções que envolvam ações de chave de acesso do IAM.

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Use as funções de wrapper para criar usuários com políticas diferentes e use as credenciais deles para acessar um bucket do Amazon S3.

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in
 an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
    The demo then shows how the users can perform only the actions they are
 permitted
    to perform.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print("-" * 88)
    print(
        "Users can have policies and roles attached to grant them specific "
        "permissions."
    )
    s3 = boto3.resource("s3")
    bucket = s3.create_bucket(
        Bucket=f"demo-iam-bucket-{time.time_ns()}",
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user("demo-iam-read-writer")
    user_reader = create_user("demo-iam-reader")
    print(f"Created two IAM users: {user_read_writer.name} and
 {user_reader.name}")
    update_user(user_read_writer.name, "demo-iam-creator")
    update_user(user_reader.name, "demo-iam-getter")
    users = list_users()
    user_read_writer = next(
        user for user in users if user.user_id == user_read_writer.user_id
    )
```

```
    user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
    print(
        f"Changed the names of the users to {user_read_writer.name} "
        f"and {user_reader.name}."
    )

    read_write_policy = policy_wrapper.create_policy(
        "demo-iam-read-write-policy",
        "Grants rights to create and get an object in the demo bucket.",
        ["s3:PutObject", "s3:GetObject"],
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(
        f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}"
    )
    print(read_write_policy.description)
    read_policy = policy_wrapper.create_policy(
        "demo-iam-read-policy",
        "Grants rights to get an object from the demo bucket.",
        "s3:GetObject",
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(f"Created policy {read_policy.policy_name} with ARN:
{read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to
{user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}.")
    user_reader_key = access_key_wrapper.create_key(user_reader.name)
    print(f"Created access key pair for {user_reader.name}.")

    s3_read_writer_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_read_writer_key.id,
        aws_secret_access_key=user_read_writer_key.secret,
    )
    demo_object_key = f"object-{time.time_ns()}"
```

```
    demo_object = None
    while demo_object is None:
        try:
            demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
                Key=demo_object_key, Body=b"AWS IAM demo object content!"
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise
    print(
        f"Put {demo_object_key} into {bucket.name} using "
        f"{user_read_writer.name}'s credentials."
    )

    read_writer_object = s3_read_writer_resource.Bucket(bucket.name).Object(
        demo_object_key
    )
    read_writer_content = read_writer_object.get()["Body"].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret,
    )
    demo_content = None
    while demo_content is None:
        try:
            demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
            demo_content = demo_object.get()["Body"].read()
            print(f"Got object {demo_object.key} using reader user's
credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
```

```
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("-" * 88)
            print(
                "Tried to delete the object using the reader user's credentials.
"
                "Got expected AccessDenied error because the reader is not "
                "allowed to delete objects."
            )
            print("-" * 88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
    policy_wrapper.delete_policy(read_policy.arn)
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(
        f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}."
    )

    bucket.objects.delete()
    bucket.delete()
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachUserPolicy

- CreateAccessKey
- CreatePolicy
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteUser
- DetachUserPolicy
- ListUsers
- UpdateUser

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
```

```
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
```

```
    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
```

```
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
$TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
```

```
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
file for details."
    fi
```

```
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
```

```
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
```

```
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}
```

```
# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role
```

```
        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
```

```
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
```

```
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
```

```
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."
```

```
    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
```

```
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10
```

```
        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "==========================================="
    echo "Resources created:"
```

```
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup

- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Como criar um espaço de trabalho de monitoramento gerenciado

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para seu espaço de trabalho.
- Criar um espaço de trabalho do Grafana.
- Configurar autenticação
- Definir configurações opcionais.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
```

```
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
```

```
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-
grafana-gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)
```

```
if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id
 "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -
f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi
```

```
# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML
 settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
```

```
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 |
 grep -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi
```

```
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Como criar um painel de monitoramento com o nome da função como uma variável

O exemplo de código a seguir mostra como:

- Crie um CloudWatch painel
- Adicionar widgets de métricas do Lambda com uma variável de nome de função.
- Verificar o painel.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}
```

```
# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
```

```
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi
```

```
    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
```

```
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
```

```
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi
```

```
# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
```

```
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi
```

```
# Prompt for cleanup with automatic yes
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
```

```
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Como criar sua primeira função com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Lambda.
- Criar código da função.
- Criar uma função do Lambda
- Testar uma função do Lambda.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

###############################################################################
# Setup
###############################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
```

```
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

##############################################################################
# Helper functions
##############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
```

```
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function
 deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
 not detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
```

```
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}
```

```
##############################################################################
# Region pre-check
##############################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

##############################################################################
# Runtime selection
##############################################################################

echo ""
echo "============================================"
echo "AWS Lambda - Create Your First Function"
echo "============================================"
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
```

```
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

##############################################################################
# Step 1: Create IAM execution role
```

```
##############################################################################

echo ""
echo "============================================="
echo "Step 1: Create IAM execution role"
echo "============================================="
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
```

```
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
```

```
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "==========================================="
echo "Step 3: Invoke the function"
echo "==========================================="
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
```

```
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

###############################################################################
# Step 4: View CloudWatch logs
###############################################################################

echo ""
echo "==========================================="
echo "Step 4: View CloudWatch Logs"
echo "==========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
```

```
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################

echo ""
echo "============================================"
echo "SUMMARY"
echo "============================================"
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "============================================"
echo "CLEANUP"
```

```
echo "=========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction
  - GetFunctionConfiguration
  - GetLogEvents
  - Invoke

Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI

O exemplo de código a seguir mostra como:

- Use o gerenciador de segredos CreateSecret
- Use o gerenciador de segredos DeleteSecret
- Use o gerenciador de segredos GetSecretValue
- Use redshift CreateNamespace
- Use redshift CreateWorkgroup
- Use redshift DeleteNamespace
- Use iam CreateRole

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq
 dependency)
# This script creates a Redshift Serverless environment, loads sample data, and
 runs queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
```

```
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}
```

```
# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}
```

```
# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status:
 $status. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be
 available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
```

```
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "============================================"
  echo "CLEANUP CONFIRMATION"
  echo "============================================"
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --
workgroup-name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
```

```
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --
namespace-name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
```

```
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin
 credentials for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"
```

```
# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless
 get-namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless
 get-workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
```

```
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5
```

```
# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
```

```
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
```

```
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
```

```
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID"
 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi
```

```
# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time
 total sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price
 desc) as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
```

```
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID"
 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
```

```
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole
  - GetSecretValue
  - GetWorkgroup
  - PutRolePolicy
  - UpdateNamespace

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.

- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
```

```
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
```

```
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
```

```
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
```

```
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
```

```
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
```

```
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"
```

```
    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
```

```
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Conceitos básicos da proteção de dispositivos da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM obrigatórios.
- Habiltiar verificações de auditoria do IoT Device Defender.
- Execute uma On-Demand auditoria
- Criar uma ação de mitigação.
- Aplicar ações de mitigação às descobertas.
- Configurar notificações do SNS (opcional).
- Habilitar o registro em log de IoT

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "==================================================="
echo "Starting script execution at $(date)"
```

```
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /
dev/null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null) || true
```

```
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn'
 --output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
```

```
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)
```

```
        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
```

```
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}
```

```
# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "==================================================="
echo "Step 1: Creating required IAM roles"
echo "==================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")
```

```
# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
```

```
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "====================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "====================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1)
 || true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"
```

```
# Step 3: Run an on-demand audit
echo ""
echo "====================================================="
echo "Step 3: Running an on-demand audit"
echo "====================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi
```

```
    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null ||
 echo "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi
```

```
# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "====================================================="
echo "Step 4: Creating a mitigation action"
echo "====================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting
 it first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction"
 2>&1 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi
```

```
MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1
 || true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "==================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "==================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)
```

```
    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1)
 || true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
```

```
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --
output json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
```

```
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS
 notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "====================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "====================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi
```

```
    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2
 methods, continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved
 but could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10
```

```
# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
```

```
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --
output json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null
 || echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}
```

```
cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies
  - ListTopics
  - PutRolePolicy
  - SetLoggingOptions

- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Conceitos básicos de arquivos de atributos de machine learning

O exemplo de código a seguir mostra como:

- Configurar permissões do IAM.
- Crie uma função SageMaker de execução
- Criar grupo de atributos.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
```

```
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."
```

```
    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
```

```
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
```

```
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
```

```
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
```

```
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
```

```
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
```

```
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
```

```
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
```

```
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
```

```
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
```

```
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi
```

```
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Conceitos básicos de clusters gerenciados de kubernetes

O exemplo de código a seguir mostra como:

- Criar uma VPC para um cluster do EKS.
- Criar perfis do IAM para um cluster do EKS.
- Criação do seu cluster de EKS
- Configurar o kubectl para se comunicar com o cluster.
- Crie um grupo de nós gerenciados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
```

```
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
```

```
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
```

```
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."
```

```
# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
```

```
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
```

```
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"
```

```
# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
```

```
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
```

```
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
```

```
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "==========================================="

# Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNodegroups
  - UpdateKubeconfig
  - Wait

Conceitos básicos de streaming gerenciado

O exemplo de código a seguir mostra como:

- Criar um cluster do Amazon MSK.
- Criar permissões do IAM para acessar o MSK.
- Criar uma máquina cliente
- Obter os agentes de bootstrap.
- Configurar a máquina cliente.
- Crie um tópico e produce/consume dados

- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
```

```
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
```

```
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
```

```
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
 to terminate instance"
        echo "Waiting for instance to terminate..."
```

```
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
"Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
"$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are
removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
```

```
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
 profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
 "$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
 delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
 "$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
```

```
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}
```

```
# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"
```

```
if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)
```

```
if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
```

```
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
```

```
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10
```

```
echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
```

```
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
```

```
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
```

```
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi
```

```
echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
```

```
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
```

```
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
```

```
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
```

```
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
```

```
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "================================================"
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
```

```
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "==============================================="
echo ""

# Ask if user wants to clean up resources
```

```
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair
  - DeletePolicy
  - DeleteRole
  - DeleteSecurityGroup
  - DescribeAvailabilityZones

- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Conceitos básicos de clusters de data warehouse provisionados

O exemplo de código a seguir mostra como:

- Criar um cluster do Redshift.
- Criar um perfil do IAM para acessar o S3.
- Criar tabelas e carregar dados.
- Executar consultas de exemplo.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."
```

```
    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-
final-cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/
dev/null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""

    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --
query 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
```

```
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id
 "$statement_id" --query 'Error' --output text 2>/dev/null || echo "Unknown
 error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus"
 \
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi
```

```
        echo "IAM role status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and
 try again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi

# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"
```

```
echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
```

```
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed
 to attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="
```

```
echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer
 not null, eventid integer not null, dateid smallint not null sortkey, qtysold
 smallint not null, pricepaid decimal(8,2), commission decimal(8,2), saletime
 timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint
 not null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi

echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
```

```
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';"
 \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt'
 DELIMITER '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
fi
```

```
# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid
 AND caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- CreateCluster
- CreateRole
- DeleteCluster
- DeleteRole
- DeleteRolePolicy
- DescribeClusters
- GetRole
- ModifyClusterIamRoles
- PutRolePolicy
- Wait

Conceitos básicos da orquestração de fluxo de trabalaho

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Step Functions.
- Criar sua primeira máquina de estado.
- Iniciar a execução da máquina de estado.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS
 Step Functions Getting Started tutorial
```

```
set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
done

# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT

echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
```

```
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be
 less reliable."
    echo "Consider installing jq for better error handling: brew install jq
 (macOS) or apt-get install jq (Ubuntu)"
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
export AWS_REGION="$CURRENT_REGION"

# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
```

```
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}

# Function to extract JSON field efficiently
extract_json_field() {
    local json="$1"
    local field="$2"
```

```
    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] ||
 [ "$wait_time" -gt 300 ]; then
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"

    if [[ "$USE_JQ" == "true" ]]; then
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
```

```
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."

    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
 $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for detach operations to complete
    wait 2>/dev/null || true

    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
```

```
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/
null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed
 to delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
        "input.json"
        "sentiment-input.json"
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
            fi
        fi
    done
}

# Security: Set trap to cleanup on script exit
```

```
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
```

```
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
```

```
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)

check_api_error "$ROLE_RESULT" "Create IAM role"
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"

# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
```

```
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")

if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)

if [ $? -ne 0 ]; then
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi
```

```
# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs
 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)

check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"

# Wait for execution to complete (the workflow has a 10-second wait state)
echo "Waiting for execution to complete (12 seconds)..."
```

```
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
```

```
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
```

```
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)

check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"

# Wait for execution to complete (the workflow has a 5-second wait state)
echo "Waiting for execution to complete (8 seconds)..."
```

```
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")

    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
    fi
    echo "Policy ARN: $POLICY_ARN"
```

```
    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
```

```
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
      },
      "Next": "SuccessState"
    },
```

```
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment
 analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"

    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)
```

```
    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3

    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in
 $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
```

```
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
fi
echo "==========================================="

# Cleanup
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy
  - StartExecution
  - UpdateStateMachine

Gerencie as chaves de acesso do IAM usando um AWS SDK

O exemplo de código a seguir mostra como gerenciar chaves de acesso.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Criar e listar chaves de acesso.
- Descobrir quando e como a chave de acesso foi usada pela última vez.
- Atualizar e excluir chaves de acesso.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações de chave de acesso do IAM.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
```

```
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys



def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
```

```
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response



def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
 key_id
        )
        raise



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.
```

```
    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Use a funções de wrapper para executar ações de chave de acesso para o usuário atual.

```
def usage_demo():
    """Shows how to create and manage access keys."""

    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep="\n")

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print("-" * 88)
    current_user_name = iam.CurrentUser().user_name
    print(
        f"This demo creates an access key for the current user "
        f"({current_user_name}), manipulates the key in a few ways, and then "
        f"deletes it."
    )
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
        print(
            "The current user already has the maximum of 2 access keys. To run "
            "this demo, either delete one of the access keys or use a user "
            "that has only 1 access key."
```

```
        )
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret
{new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)["AccessKeyLastUsed"]
        print(
            f"Key {all_keys[0].id} was last used to access
{last_use['ServiceName']} "
            f"on {last_use['LastUsedDate']}"
        )
        update_key(current_user_name, new_key.id, False)
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()
        delete_key(current_user_name, new_key.id)
        print_keys()
        print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAccessKey
  - DeleteAccessKey
  - GetAccessKeyLastUsed
  - ListAccessKeys
  - UpdateAccessKey

Gerencie políticas do IAM usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar e listar políticas.
- Criar e obter versões de políticas.
- Reverter uma política para uma versão anterior.
- Excluir políticas.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações de política do IAM.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
```

```
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies



def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
```

```
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version



def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
```

```
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement



def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.",
 rollback_version.version_id)
        default_version.delete()
```

```
        logger.info("Deleted original default version %s.",
 default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
 so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

Use a funções de wrapper para criar políticas, atualizar versões e obter informações sobre elas.

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print("-" * 88)
    print(
        "Policies let you define sets of permissions that can be attached to "
        "other IAM resources, like users and roles."
    )
    bucket_arn = f"arn:aws:s3:::amzn-s3-demo-bucket"
    policy = create_policy(
        "demo-iam-policy",
        "Policy for IAM demonstration.",
        ["s3:ListObjects"],
        bucket_arn,
    )
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies("Local")
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=", ")
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ["s3:PutObject"], bucket_arn, True
    )
    print(
        f"Added policy version {policy_version.version_id} to policy "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(
        f"Rolled back to version {rollback_version.version_id} for "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
    print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreatePolicy
  - CreatePolicyVersion
  - DeletePolicy
  - DeletePolicyVersion
  - GetPolicyVersion
  - ListPolicies
  - ListPolicyVersions
  - SetDefaultPolicyVersion

Gerencie as funções do IAM usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM.
- Anexar e separar políticas para um perfil.
- Excluir um perfil.

Python

SDK para Python (Boto3)

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar perfis que envolvam ações de perfil do IAM.

```
import json
import logging
import pprint

import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role



def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
```

```
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise



def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise



def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

Use a funções de wrapper para criar uma função, depois, anexe e desanexe uma política.

```
def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print("-" * 88)
    print(
        "Roles let you define sets of permissions and can be assumed by "
        "other entities, like users and services."
    )
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        "demo-iam-role", ["lambda.amazonaws.com",
 "batchoperations.s3.amazonaws.com"]
    )
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    attach_policy(role.name, policy_arn)
    print(f"Attached policy {policy_arn} to {role.name}.")
    print(f"Policies attached to role {role.name} are:")
    list_attached_policies(role.name)
    detach_policy(role.name, policy_arn)
    print(f"Detached policy {policy_arn} from {role.name}.")
    delete_role(role.name)
    print(f"Deleted {role.name}.")
    print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachRolePolicy

- CreateRole
- DeleteRole
- DetachRolePolicy

Gerencie sua conta do IAM usando um AWS SDK

O exemplo de código a seguir mostra como:

- Obter e atualizar o alias da conta.
- Gerar um relatório de usuários e credenciais.
- Obter um resumo da utilização da conta.
- Obter detalhes de todos os usuários, grupos, perfis e políticas em sua conta, incluindo as relações uns com os outros.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que envolvam ações de conta do IAM.

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def list_aliases():
    """
```

```
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]



def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise



def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
```

```
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise



def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response



def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
```

```
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]



def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map



def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
```

```
        return account_details
```

Chame funções de wrapper para alterar o alias da conta e obter relatórios sobre a conta.

```
def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print("-" * 88)
    print(
        "Setting an account alias lets you use the alias in your sign-in URL "
        "instead of your account number."
    )
    old_aliases = list_aliases()
    if len(old_aliases) > 0:
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f"alias-{index}-{time.time_ns()}"
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print("-" * 88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
    report_state = None
    while report_state != "COMPLETE":
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response["State"]
        if report_state != old_report_state:
```

```
            print(report_state, sep="")
        else:
            print(".", sep="")
        sys.stdout.flush()
        time.sleep(1)
    print()
    cred_report = get_credential_report()
    col_count = 3
    print(f"Got credentials report. Showing only the first {col_count} columns.")
    cred_lines = [
        line.split(",")[:col_count] for line in
 cred_report.decode("utf-8").split("\n")
    ]
    col_width = max([len(item) for line in cred_lines for item in line]) + 2
    for line in cred_report.decode("utf-8").split("\n"):
        print(
            "".join(element.ljust(col_width) for element in line.split(",")
[:col_count])
        )

    print("-" * 88)
    print("Let's get an account summary.")
    summary = get_summary()
    print("Here's your summary:")
    pprint.pprint(summary)

    print("-" * 88)
    print("Let's get authorization details!")
    details = get_authorization_details([])
    see_details = input("These are pretty long, do you want to see them (y/n)? ")
    if see_details.lower() == "y":
        pprint.pprint(details)

    print("-" * 88)
    pw_policy_created = None
    see_pw_policy = input("Want to see the password policy for the account (y/n)?
 ")
    if see_pw_policy.lower() == "y":
        while True:
            if print_password_policy():
                break
            else:
                answer = input(
                    "Do you want to create a default password policy (y/n)? "
```

```
                )
                if answer.lower() == "y":
                    pw_policy_created = iam.create_account_password_policy()
                else:
                    break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == "y":
            pw_policy_created.delete()
            print("Password policy deleted.")

    print("The SAML providers for your account are:")
    list_saml_providers(10)

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAccountAlias
  - DeleteAccountAlias
  - GenerateCredentialReport
  - GetAccountAuthorizationDetails
  - GetAccountSummary
  - GetCredentialReport
  - ListAccountAliases

Como mover segredos codificados para o armazenamento seguro de segredos

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Criar um segredo no Secrets Manager.
- Atualizar o código de uma aplicação.
- Atualizar o segredo.

- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
```

```
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-
delete-without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name
 "$RUNTIME_ROLE_NAME" --query 'PolicyNames[]' --output text 2>/dev/null || true);
 do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
```

```
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
```

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
```

```
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_u
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"
```

```
# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/
null || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
```

```
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').tok

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
```

```
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole

- DeleteSecret
- DetachRolePolicy
- GetCallerIdentity
- GetSecretValue
- PutResourcePolicy
- UpdateSecret

Permite que AWS Compute Optimizer Recurso de automação para aplicar ações recomendadas

O exemplo de código a seguir mostra como: Essa política baseada em permissões permite que o recurso de AWS Compute Optimizer automação aplique as ações recomendadas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "aco-automation.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Política para permitir a automação em toda a sua organização

O exemplo de código a seguir mostra como esta política baseada em permissões permite a automação em toda a sua organização

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:AssociateAccounts",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:DisassociateAccounts",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:ListAccounts",
            "Resource": "*"
        }
    ]
}
```

Política para permitir a automação na sua conta

O exemplo de código a seguir mostra como esta política baseada em permissões permite a automação na sua conta

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        }
    ]
}
```

Política para conceder acesso total ao Compute Optimizer Automation para uma conta gerencial de uma organização

O exemplo de código a seguir mostra como esta política baseada em permissões concede acesso total à automação do Compute Optimizer Automation para uma conta gerencial de uma organização

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
               "ec2:DescribeVolumes",
               "organizations:ListAccounts",
               "organizations:DescribeOrganization",
               "organizations:DescribeAccount",
               "organizations:EnableAWSServiceAccess",
               "organizations:ListDelegatedAdministrators",
               "organizations:RegisterDelegatedAdministrator",
               "organizations:DeregisterDelegatedAdministrator"
            ],
            "Resource": "*"
        }
    ]
}
```

Política para conceder acesso total à Automação do Compute Optimizer para uso autônomo AWS contas

O exemplo de código a seguir mostra como essa política baseada em permissões concede acesso total à Automação do Compute Optimizer para contas autônomas. AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
            "ec2:DescribeVolumes"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Política para conceder acesso somente leitura ao Compute Optimizer Automation para uma conta gerencial de uma organização

O exemplo de código a seguir mostra como esta política baseada em permissões concede acesso somente leitura ao Compute Optimizer Automation para uma conta gerencial de uma organização

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:GetEnrollmentConfiguration",
               "aco-automation:GetAutomationEvent",
               "aco-automation:GetAutomationRule",
               "aco-automation:ListAccounts",
               "aco-automation:ListAutomationEvents",
               "aco-automation:ListAutomationEventSteps",
               "aco-automation:ListAutomationEventSummaries",
               "aco-automation:ListAutomationRules",
               "aco-automation:ListAutomationRulePreview",
               "aco-automation:ListAutomationRulePreviewSummaries",
               "aco-automation:ListRecommendedActions",
               "aco-automation:ListRecommendedActionSummaries",
               "aco-automation:ListTagsForResource",
               "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

Política para conceder acesso somente de leitura à Automação do Compute Optimizer para uso autônomo AWS contas

O exemplo de código a seguir mostra como: Essa política baseada em permissões concede acesso somente de leitura à Automação do Compute Optimizer para contas autônomas. AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:GetEnrollmentConfiguration",
               "aco-automation:GetAutomationEvent",
               "aco-automation:GetAutomationRule",
               "aco-automation:ListAutomationEvents",
               "aco-automation:ListAutomationEventSteps",
               "aco-automation:ListAutomationEventSummaries",
               "aco-automation:ListAutomationRules",
               "aco-automation:ListAutomationRulePreview",
               "aco-automation:ListAutomationRulePreviewSummaries",
               "aco-automation:ListRecommendedActions",
               "aco-automation:ListRecommendedActionSummaries",
               "aco-automation:ListTagsForResource",
               "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

Política para conceder permissões de perfil vinculado ao serviço para o Compute Optimization Automation

O exemplo de código a seguir mostra como esta política baseada em permissões concede permissões de perfil vinculado ao serviço para o Compute Optimization Automation

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        }
    ]
}
```

Reverta uma versão da política do IAM usando um AWS SDK

O exemplo de código a seguir mostra como:

- Obter a lista de versões da política em ordem por data.
- Encontrar a versão da política padrão.
- Tornar a versão da política anterior a padrão.
- Excluir a versão padrão antiga.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
```

```
        logger.info("Set %s as the default version.",
rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - DeletePolicyVersion
  - ListPolicyVersions
  - SetDefaultPolicyVersion

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento

- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
```

```
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi
```

```
    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
```

```
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
```

```
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
```

```
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60
```

```
# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
```

```
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
```

```
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail
```

```
    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
```

```
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "==========================================="
```

```
# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
```

```
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances
  - StartExperiment
  - StopExperiment
  - TerminateInstances
  - Wait

Configure o controle de Attribute-Based acesso para o DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como implementar o Controle de Attribute-Based Acesso (ABAC) para o DynamoDB.

- Crie uma política do IAM para o ABAC.
- Crie tabelas com etiquetas que representem diferentes departamentos.
- Liste e filtre tabelas com base nas etiquetas.

Bash

AWS CLI com script Bash

Crie uma política do IAM para o ABAC.

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
```

```
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

Crie tabelas com etiquetas que representem diferentes departamentos.

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
```

```
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

Liste e filtre tabelas com base nas etiquetas.

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text |
 xargs -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn"
 --output text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreatePolicy
  - CreateTable
  - ListTables

Como configurar o gerenciamento de sistemas

O exemplo de código a seguir mostra como:

- Criar permissões do IAM para o Systems Manager.
- Criar um perfil do IAM para o Systems Manager
- Configurar o Systems Manager.
- Verificar a configuração.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
```

```
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
```

```
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id
 --policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --
manager-arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
```

```
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/
null || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and
 region."
echo "It will create IAM policies and roles, then enable Systems Manager
 features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
```

```
        "Effect": "Allow",
        "Action": [
          "ssm:DescribeAutomationExecutions",
          "ssm:GetAutomationExecution",
          "ssm:ListAssociations",
          "ssm:DescribeAssociation",
          "ssm:ListDocuments",
          "ssm:ListResourceDataSync",
          "ssm:DescribePatchBaselines",
          "ssm:GetPatchBaseline",
          "ssm:DescribeMaintenanceWindows",
          "ssm:DescribeMaintenanceWindowTasks"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
```

```
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
```

```
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
```

```
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
```

```
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
```

```
        "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
        "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
        "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
        "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
      ]
    }
  ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"
```

```
# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
```

```
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --
policy-arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
```

```
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few
 minutes)..."
sleep 30
```

```
# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-
arn $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration
 manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later." | tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AttachRolePolicy
- CreateConfigurationManager
- CreatePolicy
- CreateRole
- DeleteConfigurationManager
- DeletePolicy
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetConfigurationManager

Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar funções do Lambda para monitoramento.
- Crie um CloudWatch painel
- Adicionar uma variável de propriedade ao painel.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
```

```
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM
```

```
# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1
 >> "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/
dev/null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"
```

```
    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
```

```
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
```

```
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
```

```
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /
dev/null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /
dev/null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a
 "$LOG_FILE"

# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
```

```
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name
 "$DASHBOARD_NAME" --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch
 dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/" | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee
 -a "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
```

```
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME"
 --output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AttachRolePolicy
- CreateFunction
- CreateRole
- DeleteDashboards
- DeleteFunction
- DeleteRole
- DetachRolePolicy
- GetDashboard
- Invoke
- PutDashboard

Trabalhe com o DynamoDB Streams e use Time-to-Live AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como gerenciar os recursos e os streams do DynamoDB. Time-to-Live

- Crie uma tabela com a opção de fluxos habilitada.
- Descreva os fluxos.
- Crie uma função do Lambda para processar os fluxos.
- Habilite a vida útil (TTL) em uma tabela.
- Adicione itens com atributos de TTL.
- Descreva as configurações de TTL.

Bash

AWS CLI com script Bash

Crie uma tabela com a opção de fluxos habilitada.

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
```

```
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

Descreva os fluxos.

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)

echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

Crie uma função do Lambda para os fluxos.

```
# Step 1: Create an IAM role for the Lambda function
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
echo "    --starting-position LATEST"
```

Habilite a vida útil (TTL) em uma tabela.

```
# Create a table for TTL demonstration
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo

# Enable TTL on the table
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

Adicione itens com atributos de TTL.

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

Descreva as configurações de TTL.

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
    --table-name TTLDemo
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole
  - CreateTable
  - DescribeTable
  - DescribeTimeToLive
  - PutItem
  - UpdateTimeToLive

Trabalhe com a API IAM Policy Builder usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar políticas do IAM usando a API orientada por objetos.
- Usar a API IAM Policy Builder com o serviço do IAM.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Os exemplos usam as importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Crie uma política com base no tempo.

```
        public String timeBasedPolicyExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addResource(IamResource.ALL)
                                                .addCondition(b1 -> b1

.operator(IamConditionOperator.DATE_GREATER_THAN)

.key("aws:CurrentTime")

.value("2020-04-01T00:00:00Z"))
                                                .addCondition(b1 -> b1

.operator(IamConditionOperator.DATE_LESS_THAN)

.key("aws:CurrentTime")

.value("2020-06-30T23:59:59Z")))
                                .build();

                // Use an IamPolicyWriter to write out the JSON string to a more
readable
                // format.
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true)
                                .build());
        }
```

Crie uma política com várias condições.

```
        public String multipleConditionsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")

.addAction("dynamodb:BatchGetItem")
                                                .addAction("dynamodb:Query")
```

```
                                                        .addAction("dynamodb:PutItem")
                                                        .addAction("dynamodb:UpdateItem")
                                                        .addAction("dynamodb:DeleteItem")

  .addAction("dynamodb:BatchWriteItem")

  .addResource("arn:aws:dynamodb:*:*:table/table-name")

  .addConditions(IamConditionOperator.STRING_EQUALS

  .addPrefix("ForAllValues:"),

  "dynamodb:Attributes",
                                                                    List.of("column-
name1", "column-name2", "column-name3"))
                                                        .addCondition(b1 -> b1

  .operator(IamConditionOperator.STRING_EQUALS

  .addSuffix("IfExists"))

  .key("dynamodb:Select")

  .value("SPECIFIC_ATTRIBUTES")))
                                .build();

                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Use entidades principais em uma política.

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                        .effect(IamEffect.DENY)
                                                        .addAction("s3:*")
                                                        .addPrincipal(IamPrincipal.ALL)
                                                        .addResource("arn:aws:s3:::amzn-
s3-demo-bucket/*")
                                                        .addResource("arn:aws:s3:::amzn-
s3-demo-bucket")
```

```
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.ARN_NOT_EQUALS)

 .key("aws:PrincipalArn")

 .value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Permitir o acesso entre contas ao .

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)

 .addPrincipal(IamPrincipalType.AWS, "111122223333")
                                                .addAction("s3:PutObject")
                                                .addResource("arn:aws:s3:::amzn-
s3-demo-bucket/*")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                .key("s3:x-amz-
acl")
                                                                .value("bucket-
owner-full-control")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Crie e carregue uma `IamPolicy`.

```
        public String createAndUploadPolicyExample(IamClient iam, String
 accountID, String policyName) {
                // Build the policy.
```

```
                IamPolicy policy = IamPolicy.builder() // 'version' defaults to
 "2012-10-17".
                                .addStatement(IamStatement.builder()
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:PutItem")

 .addResource("arn:aws:dynamodb:us-east-1:" + accountID
                                                                    + ":table/
exampleTableName")
                                                .build())
                                .build();
                // Upload the policy.
                iam.createPolicy(r ->
 r.policyName(policyName).policyDocument(policy.toJson()));
                return
 policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

Baixe e trabalhe com uma `IamPolicy`.

```
        public String createNewBasedOnExistingPolicyExample(IamClient iam, String
 accountID, String policyName,
                        String newPolicyName) {

                String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
 policyName;
                GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
 r.policyArn(policyArn));

                String policyVersion =
 getPolicyResponse.policy().defaultVersionId();
                GetPolicyVersionResponse getPolicyVersionResponse = iam
                                .getPolicyVersion(r ->
 r.policyArn(policyArn).versionId(policyVersion));

                // Create an IamPolicy instance from the JSON string returned
 from IAM.
                String decodedPolicy =
 URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                                StandardCharsets.UTF_8);
                IamPolicy policy = IamPolicy.fromJson(decodedPolicy);
```

```
                /*
                 * All IamPolicy components are immutable, so use the copy method
that creates a
                 * new instance that
                 * can be altered in the same method call.
                 *
                 * Add the ability to get an item from DynamoDB as an additional
action.
                 */
                IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

                // Create a new statement that replaces the original statement.
                IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

                // Upload the new policy. IAM now has both policies.
                iam.createPolicy(r -> r.policyName(newPolicyName)
                                .policyDocument(newPolicy.toJson()));

                return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK for Java 2.x.
- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

Exemplos de código para o Amazon Inspector usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Inspector com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário do Amazon Inspector — Mais informações sobre o Amazon Inspector.
- Referência da API do Amazon Inspector — Detalhes sobre todas as ações disponíveis do Amazon Inspector.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon Inspector AWS SDKs
  - Hello
  - Aprenda as noções básicas do Amazon Inspector com um AWS SDK
  - Ações para o Amazon Inspector usando AWS SDKs
    - Use BatchGetAccountStatuscom um AWS SDK
    - Use BatchGetFindingDetailscom um AWS SDK
    - Use CreateFiltercom um AWS SDK
    - Use DeleteFiltercom um AWS SDK
    - Use Desativar com um AWS SDK
    - Use Habilitar com um AWS SDK
    - Use ListCoveragecom um AWS SDK
    - Use ListCoverageStatisticscom um AWS SDK
    - Use ListFilterscom um AWS SDK
    - Use ListFindingscom um AWS SDK ou CLI
    - Use ListUsageTotalscom um AWS SDK

Exemplos básicos para o uso do Amazon Inspector AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Inspector AWS com SDKs.

Exemplos

- Hello
- Aprenda as noções básicas do Amazon Inspector com um AWS SDK
- Ações para o Amazon Inspector usando AWS SDKs
  - Use BatchGetAccountStatuscom um AWS SDK
  - Use BatchGetFindingDetailscom um AWS SDK
  - Use CreateFiltercom um AWS SDK
  - Use DeleteFiltercom um AWS SDK
  - Use Desativar com um AWS SDK
  - Use Habilitar com um AWS SDK
  - Use ListCoveragecom um AWS SDK
  - Use ListCoverageStatisticscom um AWS SDK
  - Use ListFilterscom um AWS SDK
  - Use ListFindingscom um AWS SDK ou CLI
  - Use ListUsageTotalscom um AWS SDK

Hello

O exemplo de código a seguir mostra como começar a usar o .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloInspector {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloInspector.class);

    public static void main(String[] args) {
        logger.info("Hello Amazon Inspector!");

        try (Inspector2Client inspectorClient =
 Inspector2Client.builder().build()) {

            logger.info("Listing member accounts for this Inspector administrator
 account...");
            listMembers(inspectorClient);

            logger.info("The Hello Inspector example completed successfully.");

        } catch (Inspector2Exception e) {
            logger.error("Error: {}", e.getMessage());
            logger.info("Troubleshooting:");
            logger.info("1. Verify AWS credentials are configured");
            logger.info("2. Check IAM permissions for Inspector2");
            logger.info("3. Ensure Inspector2 is enabled in your account");
            logger.info("4. Verify you're using a supported region");
        }
    }

    /**
     * Lists all member accounts associated with the current Inspector
 administrator account.
     *
     * @param inspectorClient The Inspector2Client used to interact with AWS
 Inspector.
     */
    public static void listMembers(Inspector2Client inspectorClient) {
```

```
        try {
            ListMembersRequest request = ListMembersRequest.builder()
                    .maxResults(50) // optional: limit results
                    .build();

            ListMembersResponse response = inspectorClient.listMembers(request);
            List<Member> members = response.members();

            if (members == null || members.isEmpty()) {
                logger.info("No member accounts found for this Inspector
 administrator account.");
                return;
            }

            logger.info("Found {} member account(s):", members.size());
            for (Member member : members) {
                logger.info(" - Account ID: {}, Status: {}",
                        member.accountId(),
                        member.relationshipStatusAsString());
            }

        } catch (Inspector2Exception e) {
            logger.error("Failed to list members: {}",
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ListMembersa Referência AWS SDK for Java 2.x da API.

Aprenda as noções básicas do Amazon Inspector com um AWS SDK

O exemplo de código a seguir mostra como:

- Verifique o status da conta do Inspector.
- Verifique se o Inspector está ativado.
- Analise as descobertas de segurança.
- Verifique a cobertura do escaneamento.
- Crie um filtro de descobertas.

- Liste os filtros existentes.
- Verifique o uso e os custos.
- Obtenha estatísticas de cobertura.
- Exclua um filtro.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Executar um cenário interativo demonstrando os recursos de .

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

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorScenario.class);
    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        InspectorActions inspectorActions = new InspectorActions();
        logger.info("Amazon Inspector Basics Scenario");

        logger.info("""
```

```
                    Amazon Inspector is a security assessment service provided by
 Amazon Web Services (AWS) that helps
                    improve the security and compliance of applications deployed
 on AWS. It automatically assesses
                    applications for vulnerabilities or deviations from best
 practices. By leveraging Amazon Inspector,
                    users can gain insights into the overall security state of
 their application and identify potential
                    security risks.

                    This service operates by conducting both network and host-
based assessments, allowing it to detect a
                    wide range of security issues, including those related to
 operating systems, network configurations,
                    and application dependencies.
                    """);

        waitForInputToContinue();

        try {
            runScenario(inspectorActions);

            logger.info("");
            logger.info("Scenario completed successfully!");
            logger.info("");
            logger.info("What you learned:");
            logger.info("  - How to check Inspector account status");
            logger.info("  - How to enable Inspector");
            logger.info("  - How to list and analyze findings");
            logger.info("  - How to check coverage information");
            logger.info("  - How to create and manage filters");
            logger.info("  - How to track usage and costs");
            logger.info("  - How to clean up resources");
            logger.info("");

        } catch (Exception ex) {
            logger.error("Scenario failed due to unexpected error: {}",
 ex.getMessage(), ex);

        } finally {
            scanner.close();
            logger.info("Exiting...");
        }
    }
```

```
   /**
    * Runs the Inspector scenario in a step-by-step sequence.
    *
    * All InspectorActions methods are asynchronous and return
CompletableFutures.
    * Each step ends with .join(). Any async exception thrown during .join()
will bubble up
    *
    */
   public static void runScenario(InspectorActions actions) {
       String filterArn = null;
       boolean inspectorEnabled = false;

       try {
           // Step 1
           logger.info(DASHES);
           logger.info("Step 1: Checking Inspector account status...");
           String status = actions.getAccountStatusAsync().join();
           logger.info(status);
           waitForInputToContinue();

           // Step 2
           logger.info(DASHES);
           logger.info("Step 2: Enabling Inspector...");
           String message = actions.enableInspectorAsync(null).join();
           logger.info(message);
           inspectorEnabled = true;  // track that Inspector was enabled
           waitForInputToContinue();

           // Step 3
           logger.info(DASHES);
           logger.info("Step 3: Listing LOW severity findings...");

           // Call the service method
           List<String> allFindings =
actions.listLowSeverityFindingsAsync().join();

           if (!allFindings.isEmpty()) {
               // Only proceed if there are findings
               String lastArn = allFindings.get(allFindings.size() - 1);
               logger.info("Look up details on: {}", lastArn);
               waitForInputToContinue();
               String details = actions.getFindingDetailsAsync(lastArn).join();
```

```
                logger.info(details);
            } else {
                logger.info("No LOW severity findings found.");
            }

            waitForInputToContinue();

            // Step 4
            logger.info(DASHES);
            logger.info("Step 4: Listing coverage...");
            String coverage = actions.listCoverageAsync(5).join();
            logger.info(coverage);
            waitForInputToContinue();

            // Step 5
            logger.info(DASHES);
            logger.info("Step 5: Creating filter...");
            String filterName = "suppress-low-" + System.currentTimeMillis();
            filterArn = actions.createLowSeverityFilterAsync(filterName,
 "Suppress low severity findings").join();
            logger.info("Created filter: {}", filterArn);
            waitForInputToContinue();

            // Step 6
            logger.info(DASHES);
            logger.info("Step 6: Listing filters...");
            String filters = actions.listFiltersAsync(10).join();
            logger.info(filters);
            waitForInputToContinue();

            // Step 7
            logger.info(DASHES);
            logger.info("Step 7: Usage totals...");
            String usage = actions.listUsageTotalsAsync(null, 10).join();
            logger.info(usage);
            waitForInputToContinue();

            // Step 8
            logger.info(DASHES);
            logger.info("Step 8: Coverage statistics...");
            String stats = actions.listCoverageStatisticsAsync().join();
            logger.info(stats);
            waitForInputToContinue();
```

```
            // Step 9
            logger.info(DASHES);
            logger.info("Step 9: Delete filter?");
            logger.info("Filter ARN: {}", filterArn);
            logger.info("Delete the filter and disable Inspector? (y/n)");

            if (scanner.nextLine().trim().equalsIgnoreCase("y")) {
                actions.deleteFilterAsync(filterArn).join();
                logger.info("Filter deleted.");
                String disableMsg = actions.disableInspectorAsync(null).join();
                logger.info(disableMsg);
                inspectorEnabled = false; // track that Inspector was disabled
            }

            waitForInputToContinue();

        } catch (Exception ex) {
            logger.error("Scenario encountered an error: {}", ex.getMessage(),
ex);
            // Rethrow the exception
            throw ex;

        } finally {
            // Cleanup in case of an exception
            if (filterArn != null) {
                try {
                    actions.deleteFilterAsync(filterArn).join();
                    logger.info("Cleanup: Filter deleted.");
                } catch (Exception e) {
                    logger.warn("Failed to delete filter during cleanup: {}",
e.getMessage(), e);
                }
            }

            if (inspectorEnabled) {
                try {
                    actions.disableInspectorAsync(null).join();
                    logger.info("Cleanup: Inspector disabled.");
                } catch (Exception e) {
                    logger.warn("Failed to disable Inspector during cleanup: {}",
e.getMessage(), e);
                }
            }
        }
```

```
    }

    // Utility Method
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' to continue:");
            String input = scanner.nextLine().trim();
            if (input.equalsIgnoreCase("c")) break;
            logger.info("Invalid input, try again.");
        }
    }
}
```

Uma classe de wrapper para métodos do SDK de .

```
public class InspectorActions {
    private static Inspector2AsyncClient inspectorAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorActions.class);

    private static Inspector2AsyncClient getAsyncClient() {
        if (inspectorAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            inspectorAsyncClient = Inspector2AsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
```

```
        return inspectorAsyncClient;
    }


    /**
     * Enables AWS Inspector for the provided account(s) and default resource
types.
     *
     * @param accountIds Optional list of AWS account IDs.
     */
    public CompletableFuture<String> enableInspectorAsync(List<String>
accountIds) {

        // The resource types to enable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        EnableRequest.Builder requestBuilder = EnableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        EnableRequest request = requestBuilder.build();
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
```

```
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:
\n");

                    if (response.accounts() == null ||
 response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
 target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
 account.accountId() : "Unknown";
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status:
 ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }


    /**
     * Retrieves and prints the coverage statistics using a paginator.
     */
```

```
    public CompletableFuture<String> listCoverageStatisticsAsync() {
        ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
                .build();

        return getAsyncClient().listCoverageStatistics(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();

                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Validation error listing coverage
statistics: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }
```

```
                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
 ").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
 getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !
response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
```

```
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage
totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." :
summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }



    /**
     * Retrieves the account status using the Inspector2Client.
     */
    public CompletableFuture<String> getAccountStatusAsync() {
        BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
                .accountIds(Collections.emptyList())
                .build();
```

```
        return getAsyncClient().batchGetAccountStatus(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof AccessDeniedException) {
                            throw new CompletionException(
                                    "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {
```

```
                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID:
 ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !
= null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }
```

```
        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }


    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
     * @param description a descriptive string explaining the purpose of the
filter
     * @return a CompletableFuture that completes with the ARN of the created
filter
     * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
     */
```

```
    public CompletableFuture<String> createLowSeverityFilterAsync(
            String filterName,
            String description) {

        // Define a filter to match LOW severity findings.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        // Create filter criteria.
        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the filter creation request.
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .filterCriteria(filterCriteria)
                .action(FilterAction.SUPPRESS)
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }
```

```
                        // Unexpected async error
                        throw new CompletionException(
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }


    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
```

```
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead
 of a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
 findings", ex);
                });
    }



    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
 paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
 getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();
```

```
        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();

            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
 ").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
 ").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
```

```
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
                        "Validation error listing coverage: " +
cause.getMessage(), cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }


    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter:
" + cause, cause);
                    }
                    return null;
```

```
                });
    }

    /**
     * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
     *
     * @param findingArn The ARN of the finding to look up.
     * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
     * containing all available details for the finding.
     * @throws RuntimeException if the async call to Inspector2 fails.
     */
    public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
        BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
                .findingArns(findingArn)
                .build();

        return getAsyncClient().batchGetFindingDetails(request)
                .thenApply(response -> {
                    if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                        return String.format("No details found for ARN: ",
findingArn);
                    }

                    StringBuilder sb = new StringBuilder();
                    response.findingDetails().forEach(detail -> {
                        sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                                .append("Risk Score:
").append(detail.riskScore()).append("\n");

                        // ExploitObserved timings
                        if (detail.exploitObserved() != null) {
                            sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                    .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                        }

                        // Reference URLs
                        if (detail.hasReferenceUrls()) {
```

```
                            sb.append("Reference URLs:\n");
                            detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
```

```
                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
 ARN: " + findingArn, cause);
                });
    }


    /**
     * Asynchronously disables AWS Inspector for the specified accounts and
 resource types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
 Inspector;
     *                   may be {@code null} or empty to target the current
 account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to
 validation errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String>
 accountIds) {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);
```

```
        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
 account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:
\n");

                    if (response.accounts() == null ||
 response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
 target accounts.");
                        return summary.toString();
                    }
```

```
                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
 account.accountId() : "Unknown";
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status:
 ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - BatchGetAccountStatus
  - BatchGetFindingDetails
  - CreateFilter
  - DeleteFilter
  - Habilitar
  - ListCoverage
  - ListCoverageStatistics
  - ListFilters
  - ListFindings
  - ListUsageTotals

Ações para o Amazon Inspector usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Inspector com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Inspector.

Exemplos

- Use BatchGetAccountStatuscom um AWS SDK
- Use BatchGetFindingDetailscom um AWS SDK
- Use CreateFiltercom um AWS SDK
- Use DeleteFiltercom um AWS SDK
- Use Desativar com um AWS SDK
- Use Habilitar com um AWS SDK
- Use ListCoveragecom um AWS SDK
- Use ListCoverageStatisticscom um AWS SDK
- Use ListFilterscom um AWS SDK
- Use ListFindingscom um AWS SDK ou CLI
- Use ListUsageTotalscom um AWS SDK

Use **`BatchGetAccountStatus`**com um AWS SDK

O código de exemplo a seguir mostra como usar `BatchGetAccountStatus`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Retrieves the account status using the Inspector2Client.
 */
```

```
    public CompletableFuture<String> getAccountStatusAsync() {
        BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
                .accountIds(Collections.emptyList())
                .build();

        return getAsyncClient().batchGetAccountStatus(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof AccessDeniedException) {
                            throw new CompletionException(
                                    "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
```

```
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {

                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID:
 ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !
= null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }
```

- Para obter detalhes da API, consulte BatchGetAccountStatusa Referência AWS SDK for Java 2.x da API.

Use **`BatchGetFindingDetails`**com um AWS SDK

O código de exemplo a seguir mostra como usar `BatchGetFindingDetails`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }

                   StringBuilder sb = new StringBuilder();
                   response.findingDetails().forEach(detail -> {
```

```
                        sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                                .append("Risk Score:
").append(detail.riskScore()).append("\n");

                        // ExploitObserved timings
                        if (detail.exploitObserved() != null) {
                            sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                    .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                        }

                        // Reference URLs
                        if (detail.hasReferenceUrls()) {
                            sb.append("Reference URLs:\n");
                            detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
```

```
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
ARN: " + findingArn, cause);
                });
    }
```

- Para obter detalhes da API, consulte BatchGetFindingDetailsa Referência AWS SDK for Java 2.x da API.

Use **`CreateFilter`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateFilter`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
    *
    * @param filterName  the name of the filter to create
    * @param description a descriptive string explaining the purpose of the
filter
    * @return a CompletableFuture that completes with the ARN of the created
filter
    * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
    */
   public CompletableFuture<String> createLowSeverityFilterAsync(
           String filterName,
           String description) {

       // Define a filter to match LOW severity findings.
       StringFilter severityFilter = StringFilter.builder()
               .value(Severity.LOW.toString())
               .comparison(StringComparison.EQUALS)
               .build();

       // Create filter criteria.
       FilterCriteria filterCriteria = FilterCriteria.builder()
               .severity(Collections.singletonList(severityFilter))
               .build();

       // Build the filter creation request.
       CreateFilterRequest request = CreateFilterRequest.builder()
               .name(filterName)
               .filterCriteria(filterCriteria)
               .action(FilterAction.SUPPRESS)
```

```
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        // Unexpected async error
                        throw new CompletionException(
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }
```

- Para obter detalhes da API, consulte CreateFiltera Referência AWS SDK for Java 2.x da API.

Use **`DeleteFilter`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteFilter`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
```

```
                        }

                        throw new RuntimeException("Failed to delete the filter:
" + cause, cause);
                    }
                    return null;
                });
    }
```

- Para obter detalhes da API, consulte DeleteFiltera Referência AWS SDK for Java 2.x da API.

Use **`Desativar`** com um AWS SDK

O código de exemplo a seguir mostra como usar `Disable`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously disables AWS Inspector for the specified accounts and
resource types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
Inspector;
     *                   may be {@code null} or empty to target the current
account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to
validation errors,
     *                             service errors, or other exceptions
```

```
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String>
 accountIds) {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
 account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
```

```
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:
\n");

                    if (response.accounts() == null ||
 response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
 target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
 account.accountId() : "Unknown";
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status:
 ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- Para obter detalhes da API, consulte Desativar na Referência AWS SDK for Java 2.x da API.

Use **`Habilitar`** com um AWS SDK

O código de exemplo a seguir mostra como usar `Enable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Enables AWS Inspector for the provided account(s) and default resource
types.
    *
    * @param accountIds Optional list of AWS account IDs.
    */
   public CompletableFuture<String> enableInspectorAsync(List<String>
accountIds) {

       // The resource types to enable.
       List<ResourceScanType> resourceTypes = List.of(
               ResourceScanType.EC2,
               ResourceScanType.ECR,
               ResourceScanType.LAMBDA,
               ResourceScanType.LAMBDA_CODE
       );

       // Build the request.
       EnableRequest.Builder requestBuilder = EnableRequest.builder()
               .resourceTypes(resourceTypes);

       if (accountIds != null && !accountIds.isEmpty()) {
           requestBuilder.accountIds(accountIds);
       }
```

```
        EnableRequest request = requestBuilder.build();
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
 account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:
\n");

                    if (response.accounts() == null ||
 response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
 target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
 account.accountId() : "Unknown";
```

```
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status:
").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- Para obter detalhes da API, consulte Habilitar na Referência AWS SDK for Java 2.x da API.

Use **`ListCoverage`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListCoverage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
```

```
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
 getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();

            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
 ").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
 ").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }
```

```
        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
                        "Validation error listing coverage: " +
cause.getMessage(), cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }
```

- Para obter detalhes da API, consulte ListCoveragea Referência AWS SDK for Java 2.x da API.

Use **`ListCoverageStatistics`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListCoverageStatistics`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves and prints the coverage statistics using a paginator.
    */
   public CompletableFuture<String> listCoverageStatisticsAsync() {
       ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
               .build();

       return getAsyncClient().listCoverageStatistics(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();

                       if (cause instanceof ValidationException) {
                           throw new CompletionException(
                                   "Validation error listing coverage
statistics: %s".formatted(cause.getMessage()),
                                   cause
                           );
                       }

                       if (cause instanceof Inspector2Exception) {
                           Inspector2Exception e = (Inspector2Exception) cause;

                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                   e
                           );
                       }
```

```
                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }
```

- Para obter detalhes da API, consulte ListCoverageStatisticsa Referência AWS SDK for Java 2.x da API.

Use **`ListFilters`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListFilters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }

        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
```

```
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }
```

- Para obter detalhes da API, consulte ListFiltersa Referência AWS SDK for Java 2.x da API.

Use **`ListFindings`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFindings`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Listar as descobertas

O comando `list-findings` a seguir lista todas as descobertas geradas:

```
aws inspector list-findings
```

Saída:

```
{
        "findingArns": [
```

```
        "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
template/0-4r1V2mAw/run/0-MKkpXXPE/finding/0-HwPnsDm4",
        "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
template/0-4r1V2mAw/run/0-v5D6fI3v/finding/0-tyvmqBLy"
      ]
}
```

Para obter mais informações, consulte Descobertas do Amazon Inspector no Guia do usuário do Amazon Inspector.

- Para obter detalhes da API, consulte ListFindingsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();
```

```
        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead
 of a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
 findings", ex);
                });
    }
```

- Para obter detalhes da API, consulte ListFindingsa Referência AWS SDK for Java 2.x da API.

Use **`ListUsageTotals`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListUsageTotals`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Asynchronously lists Inspector2 usage totals using a paginator.
 *
 * @param accountIds optional list of account IDs
 * @param maxResults maximum results per page
 * @return CompletableFuture completed with formatted summary text
 */
public CompletableFuture<String> listUsageTotalsAsync(
        List<String> accountIds,
        int maxResults) {

    logger.info("Starting usage totals paginator…");

    ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
            .maxResults(maxResults);

    if (accountIds != null && !accountIds.isEmpty()) {
        builder.accountIds(accountIds);
    }

    ListUsageTotalsRequest request = builder.build();
```

```
        ListUsageTotalsPublisher paginator =
 getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !
response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
 usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage
 totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." :
 summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
 cause);
                });
    }
```

- Para obter detalhes da API, consulte ListUsageTotalsa Referência AWS SDK for Java 2.x da API.

Exemplos de código para AWS IoT usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS IoT com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS IoT Guia do desenvolvedor — Mais informações sobre AWS IoT.
- AWS IoT Referência da API — Detalhes sobre todas as AWS IoT ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS IoT usar AWS SDKs
  - Hello AWS IoT
  - Aprenda as noções básicas do AWS IoT com um AWS SDK
  - Ações para AWS IoT usar AWS SDKs
    - Use AttachThingPrincipalcom um AWS SDK ou CLI
    - Use CreateKeysAndCertificatecom um AWS SDK ou CLI

    - Use CreateThingcom um AWS SDK ou CLI
    - Use CreateTopicRulecom um AWS SDK ou CLI
    - Use DeleteCertificatecom um AWS SDK ou CLI
    - Use DeleteThingcom um AWS SDK ou CLI
    - Use DeleteTopicRulecom um AWS SDK ou CLI
    - Use DescribeEndpointcom um AWS SDK ou CLI
    - Use DescribeThingcom um AWS SDK ou CLI
    - Use DetachThingPrincipalcom um AWS SDK ou CLI
    - Use ListCertificatescom um AWS SDK ou CLI
    - Use ListThingscom um AWS SDK ou CLI
    - Use SearchIndexcom um AWS SDK ou CLI
    - Use UpdateIndexingConfigurationcom um AWS SDK ou CLI
    - Use UpdateThingcom um AWS SDK ou CLI
- Cenários para AWS IoT usar AWS SDKs
  - Conceitos básicos da proteção de dispositivos da Internet das Coisas
  - Conceitos básicos de mensagens da Internet das Coisas

Exemplos básicos para AWS IoT usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS IoT com AWS SDKs.

Exemplos

- Hello AWS IoT
- Aprenda as noções básicas do AWS IoT com um AWS SDK
- Ações para AWS IoT usar AWS SDKs
  - Use AttachThingPrincipalcom um AWS SDK ou CLI
  - Use CreateKeysAndCertificatecom um AWS SDK ou CLI
  - Use CreateThingcom um AWS SDK ou CLI
  - Use CreateTopicRulecom um AWS SDK ou CLI
  - Use DeleteCertificatecom um AWS SDK ou CLI
  - Use DeleteThingcom um AWS SDK ou CLI

- Use DeleteTopicRulecom um AWS SDK ou CLI
- Use DescribeEndpointcom um AWS SDK ou CLI
- Use DescribeThingcom um AWS SDK ou CLI
- Use DetachThingPrincipalcom um AWS SDK ou CLI
- Use ListCertificatescom um AWS SDK ou CLI
- Use ListThingscom um AWS SDK ou CLI
- Use SearchIndexcom um AWS SDK ou CLI
- Use UpdateIndexingConfigurationcom um AWS SDK ou CLI
- Use UpdateThingcom um AWS SDK ou CLI

Hello AWS IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Hello AWS IoT example.
/// </summary>
public class HelloIoT
{
    /// <summary>
    /// Main method to run the Hello IoT example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var iotClient = new AmazonIoTClient();
```

```
        try
        {
            Console.WriteLine("Hello AWS IoT! Let's list your IoT Things:");
            Console.WriteLine(new string('-', 80));

            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await iotClient.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
 next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await iotClient.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            if (things is { Count: > 0 })
            {
                Console.WriteLine($"Found {things.Count} IoT Things:");
                foreach (var thing in things)
                {
                    Console.WriteLine($"- Thing Name: {thing.ThingName}");
                    Console.WriteLine($"  Thing ARN: {thing.ThingArn}");
                    Console.WriteLine($"  Thing Type: {thing.ThingTypeName ?? "No
 type specified"}");
                    Console.WriteLine($"  Version: {thing.Version}");

                    if (thing.Attributes?.Count > 0)
```

```
                    {
                        Console.WriteLine("  Attributes:");
                        foreach (var attr in thing.Attributes)
                        {
                            Console.WriteLine($"    {attr.Key}: {attr.Value}");
                        }
                    }
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No IoT Things found in your account.");
                Console.WriteLine("You can create IoT Things using the IoT Basics
 scenario example.");
            }

            Console.WriteLine("Hello IoT completed successfully.");
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            Console.WriteLine($"Request throttled, please try again later:
 {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list Things. Here's why: {ex.Message}");
        }
    }
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK para .NET .

C++

SDK para C++

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)
```

```
# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_iot.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
 iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
 listThingsOutcome.GetResult().GetThings();
```

```
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                          << listThingsOutcome.GetError().GetMessage() <<
 std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK para C++ .

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Para detalhes da API, consulte listThings em Referência de API AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def hello_iot():
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT client and list
    up to 10 things in your AWS IoT account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    try:
        iot_client = boto3.client("iot")
        response = iot_client.list_things(maxResults=10)
        things = response.get("things", [])

        print("Hello, AWS IoT! Here are your things:")
        if things:
            for i, thing in enumerate(things, 1):
                print(f"{i}. {thing['thingName']}")
        else:
            print("No things found in your AWS IoT account.")
    except ClientError as e:
        if e.response["Error"]["Code"] == "UnauthorizedException":
            print("You don't have permission to access AWS IoT.")
        else:
            print(f"Couldn't access AWS IoT. Error: {e}")
    except NoCredentialsError:
        print("No AWS credentials found. Please configure your credentials.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

- Para obter detalhes da API, consulte listThings na Referência da API AWS SDK para Python (Boto3).

Aprenda as noções básicas do AWS IoT com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie qualquer AWS IoT coisa.
- Gerar um certificado de dispositivo.
- Atualize AWS IoT qualquer coisa com atributos.
- Exibir um endpoint exclusivo.
- Liste seus AWS IoT certificados.
- Atualize uma AWS IoT sombra.
- Gravar informações do estado.
- Cria uma regra.
- Listar suas regras.
- Pesquisar coisas usando o nome da coisa.
- Exclua qualquer AWS IoT coisa.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS IoT recursos.

```
/// <summary>
/// Scenario class for AWS IoT basics.
/// </summary>
public class IoTBasics
{
```

```
    public static bool IsInteractive = true;
    public static IoTWrapper? Wrapper = null;
    public static IAmazonCloudFormation? CloudFormationClient = null;
    public static ILogger<IoTBasics> logger = null!;
    private static IoTWrapper _iotWrapper = null!;
    private static IAmazonCloudFormation _amazonCloudFormation = null!;
    private static ILogger<IoTBasics> _logger = null!;

    private static string _stackName = "IoTBasicsStack";
    private static string _stackResourcePath = "../../../../../../scenarios/
basics/iot/iot_usecase/resources/cfn_template.yaml";

    /// <summary>
    /// Main method for the IoT Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIoT>(new AWSOptions() { Region =
 RegionEndpoint.USEast1 })
                    .AddAWSService<IAmazonCloudFormation>()
                        .AddTransient<IoTWrapper>()
                        .AddLogging(builder => builder.AddConsole())
                        .AddSingleton<IAmazonIotData>(sp =>
                        {
                            var iotService = sp.GetRequiredService<IAmazonIoT>();
                            var request = new DescribeEndpointRequest
                            {
                                EndpointType = "iot:Data-ATS"
                            };
                            var response =
 iotService.DescribeEndpointAsync(request).Result;
                            return new AmazonIotDataClient($"https://
{response.EndpointAddress}/");
                        })
            )
            .Build();

        logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<IoTBasics>();
```

```
        Wrapper = host.Services.GetRequiredService<IoTWrapper>();
        CloudFormationClient =
host.Services.GetRequiredService<IAmazonCloudFormation>();

        // Set the private fields for backwards compatibility
        _logger = logger;
        _iotWrapper = Wrapper;
        _amazonCloudFormation = CloudFormationClient;

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS IoT example scenario.");
        Console.WriteLine("This example program demonstrates various interactions
with the AWS Internet of Things (IoT) Core service.");
        Console.WriteLine();
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }
        Console.WriteLine(new string('-', 80));

        try
        {
            await RunScenarioAsync();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem running the scenario.");
            Console.WriteLine($"\nAn error occurred: {ex.Message}");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The AWS IoT scenario has successfully completed.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Run the IoT Basics scenario.
    /// </summary>
    /// <returns>A Task object.</returns>
    public static async Task RunScenarioAsync()
    {
        // Use static properties if available, otherwise use private fields
```

```
        var iotWrapper = Wrapper ?? _iotWrapper;
        var cloudFormationClient = CloudFormationClient ?? _amazonCloudFormation;
        var scenarioLogger = logger ?? _logger;

        await RunScenarioInternalAsync(iotWrapper, cloudFormationClient,
 scenarioLogger);
    }

    /// <summary>
    /// Internal method to run the IoT Basics scenario with injected
 dependencies.
    /// </summary>
    /// <param name="iotWrapper">The IoT wrapper instance.</param>
    /// <param name="cloudFormationClient">The CloudFormation client instance.</
param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    private static async Task RunScenarioInternalAsync(IoTWrapper iotWrapper,
 IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        string thingName = $"iot-thing-{Guid.NewGuid():N}";
        string certificateArn = "";
        string certificateId = "";
        string ruleName = $"iotruledefault";
        string snsTopicArn = "";

        try
        {
            // Step 1: Create an AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("1. Create an AWS IoT Thing.");
            Console.WriteLine("An AWS IoT Thing represents a virtual entity in
 the AWS IoT service that can be associated with a physical device.");
            Console.WriteLine();

            if (IsInteractive)
            {
                Console.Write("Enter Thing name: ");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    thingName = userInput;
            }
            else
            {
```

```
                Console.WriteLine($"Using default Thing name: {thingName}");
            }

            var thingArn = await iotWrapper.CreateThingAsync(thingName);
            Console.WriteLine($"{thingName} was successfully created. The ARN
value is {thingArn}");
            Console.WriteLine(new string('-', 80));

            // Step 1.1: List AWS IoT Things
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("2. List AWS IoT Things.");
            Console.WriteLine("Now let's list the IoT Things to see the Thing we
just created.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var things = await iotWrapper.ListThingsAsync();
            Console.WriteLine($"Found {things.Count} IoT Things:");
            foreach (var thing in things.Take(10)) // Show first 10 things
            {
                Console.WriteLine($"Thing Name: {thing.ThingName}");
                Console.WriteLine($"Thing ARN: {thing.ThingArn}");
                if (thing.Attributes != null && thing.Attributes.Any())
                {
                    Console.WriteLine("Attributes:");
                    foreach (var attr in thing.Attributes)
                    {
                        Console.WriteLine($"  {attr.Key}: {attr.Value}");
                    }
                }
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 2: Generate a Device Certificate
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("3. Generate a device certificate.");
            Console.WriteLine("A device certificate performs a role in securing
the communication between devices (Things) and the AWS IoT platform.");
```

```
            Console.WriteLine();

            var createCert = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to create a certificate for
{thingName}? (y/n)");
                createCert = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Creating certificate for {thingName}...");
            }

            if (createCert?.ToLower() == "y")
            {
                var certificateResult = await
iotWrapper.CreateKeysAndCertificateAsync();
                if (certificateResult.HasValue)
                {
                    var (certArn, certPem, certId) = certificateResult.Value;
                    certificateArn = certArn;
                    certificateId = certId;

                    Console.WriteLine($"\nCertificate:");
                    // Show only first few lines of certificate for brevity
                    var lines = certPem.Split('\n');
                    for (int i = 0; i < Math.Min(lines.Length, 5); i++)
                    {
                        Console.WriteLine(lines[i]);
                    }
                    if (lines.Length > 5)
                    {
                        Console.WriteLine("...");
                    }

                    Console.WriteLine($"\nCertificate ARN:");
                    Console.WriteLine(certificateArn);

                    // Step 3: Attach the Certificate to the AWS IoT Thing
                    Console.WriteLine("Attach the certificate to the AWS IoT
Thing.");
                    var attachResult = await
iotWrapper.AttachThingPrincipalAsync(thingName, certificateArn);
```

```
                    if (attachResult)
                    {
                        Console.WriteLine("Certificate attached to Thing
successfully.");
                    }
                    else
                    {
                        Console.WriteLine("Failed to attach certificate to
Thing.");
                    }

                    Console.WriteLine("Thing Details:");
                    Console.WriteLine($"Thing Name: {thingName}");
                    Console.WriteLine($"Thing ARN: {thingArn}");
                }
                else
                {
                    Console.WriteLine("Failed to create certificate.");
                }
            }
            Console.WriteLine(new string('-', 80));

            // Step 4: Update an AWS IoT Thing with Attributes
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("4. Update an AWS IoT Thing with Attributes.");
            Console.WriteLine("IoT Thing attributes, represented as key-value
pairs, offer a pivotal advantage in facilitating efficient data");
            Console.WriteLine("management and retrieval within the AWS IoT
ecosystem.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var attributes = new Dictionary<string, string>
            {
                { "Location", "Seattle" },
                { "DeviceType", "Sensor" },
                { "Firmware", "1.2.3" }
            };

            await iotWrapper.UpdateThingAsync(thingName, attributes);
```

```
            Console.WriteLine("Thing attributes updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 5: Return a unique endpoint specific to the Amazon Web
Services account
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("5. Return a unique endpoint specific to the Amazon
Web Services account.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var endpoint = await iotWrapper.DescribeEndpointAsync();
            if (endpoint != null)
            {
                var subdomain = endpoint.Split('.')[0];
                Console.WriteLine($"Extracted subdomain: {subdomain}");
                Console.WriteLine($"Full Endpoint URL: https://{endpoint}");
            }
            else
            {
                Console.WriteLine("Failed to retrieve endpoint.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 6: List your AWS IoT certificates
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("6. List your AWS IoT certificates");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var certificates = await iotWrapper.ListCertificatesAsync();
            foreach (var cert in certificates.Take(5)) // Show first 5
certificates
            {
                Console.WriteLine($"Cert id: {cert.CertificateId}");
                Console.WriteLine($"Cert Arn: {cert.CertificateArn}");
            }
```

```
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 7: Create an IoT shadow
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("7. Update an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowPayload = JsonSerializer.Serialize(new
            {
                state = new
                {
                    desired = new
                    {
                        temperature = 25,
                        humidity = 50
                    }
                }
            });

            await iotWrapper.UpdateThingShadowAsync(thingName, shadowPayload);
            Console.WriteLine("Thing Shadow updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 8: Write out the state information, in JSON format
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("8. Write out the state information, in JSON
format.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowData = await iotWrapper.GetThingShadowAsync(thingName);
            Console.WriteLine($"Received Shadow Data: {shadowData}");
            Console.WriteLine(new string('-', 80));
```

```
            // Step 9: Set up resources (SNS topic and IAM role) and create a
rule
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("9. Set up resources and create a rule");
            Console.WriteLine();

            // Deploy CloudFormation stack to create SNS topic and IAM role
            Console.WriteLine("Deploying CloudFormation stack to create SNS topic
and IAM role...");

            var deployStack = !IsInteractive || GetYesNoResponse("Would you like
to deploy the CloudFormation stack? (y/n) ");
            if (deployStack)
            {
                if (IsInteractive)
                {
                    Console.Write(
                        $"Enter stack resource file path (or press Enter for
default '{_stackResourcePath}'): ");
                    var userResourcePath = Console.ReadLine();
                    if (!string.IsNullOrEmpty(userResourcePath))
                        _stackResourcePath = userResourcePath;
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);

                if (deploySuccess)
                {
                    // Get stack outputs
                    var stackOutputs = await GetStackOutputs(_stackName,
cloudFormationClient, scenarioLogger);
                    if (stackOutputs != null)
                    {
                        snsTopicArn = stackOutputs["SNSTopicArn"];
                        string roleArn = stackOutputs["RoleArn"];

                        Console.WriteLine($"Successfully deployed stack. SNS
topic: {snsTopicArn}");
                        Console.WriteLine($"Successfully deployed stack. IAM
role: {roleArn}");
```

```
                        if (IsInteractive)
                        {
                            Console.Write($"Enter Rule name (press Enter for
default '{ruleName}'): ");
                            var userRuleName = Console.ReadLine();
                            if (!string.IsNullOrEmpty(userRuleName))
                                ruleName = userRuleName;
                        }
                        else
                        {
                            Console.WriteLine($"Using default rule name:
{ruleName}");
                        }

                        // Now create the IoT rule with the CloudFormation
outputs
                        var ruleResult = await
iotWrapper.CreateTopicRuleAsync(ruleName, snsTopicArn, roleArn);
                        if (ruleResult)
                        {
                            Console.WriteLine("IoT Rule created successfully.");
                        }
                        else
                        {
                            Console.WriteLine("Failed to create IoT rule.");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Failed to get stack outputs. Skipping
rule creation.");
                    }
                }
                else
                {
                    Console.WriteLine("Failed to deploy CloudFormation stack.
Skipping rule creation.");
                }
            }
            else
            {
                Console.WriteLine("Skipping CloudFormation stack deployment and
rule creation.");
            }
```

```
            Console.WriteLine(new string('-', 80));

            // Step 10: List your rules
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("10. List your rules.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var rules = await iotWrapper.ListTopicRulesAsync();
            Console.WriteLine("List of IoT Rules:");
            foreach (var rule in rules.Take(5)) // Show first 5 rules
            {
                Console.WriteLine($"Rule Name: {rule.RuleName}");
                Console.WriteLine($"Rule ARN: {rule.RuleArn}");
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 11: Search things using the Thing name
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("11. Search things using the Thing name.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var searchResults = await iotWrapper.SearchIndexAsync($"thingName:
{thingName}");
            if (searchResults.Any())
            {
                Console.WriteLine($"Thing id found using search is
 {searchResults.First().ThingId}");
            }
            else
            {
                Console.WriteLine($"No search results found for Thing:
 {thingName}");
            }
            Console.WriteLine(new string('-', 80));
```

```
            // Step 12: Cleanup - Detach and delete certificate
            if (!string.IsNullOrEmpty(certificateArn))
            {
                Console.WriteLine(new string('-', 80));
                var deleteCert = "y";
                if (IsInteractive)
                {
                    Console.Write($"Do you want to detach and delete the
certificate for {thingName}? (y/n)");
                    deleteCert = Console.ReadLine();
                }
                else
                {
                    Console.WriteLine($"Detaching and deleting certificate for
{thingName}...");
                }

                if (deleteCert?.ToLower() == "y")
                {
                    Console.WriteLine("12. You selected to detach and delete the
certificate.");
                    if (IsInteractive)
                    {
                        Console.WriteLine("Press Enter to continue...");
                        Console.ReadLine();
                    }

                    await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                    Console.WriteLine($"{certificateArn} was successfully removed
from {thingName}");

                    await iotWrapper.DeleteCertificateAsync(certificateId);
                    Console.WriteLine($"{certificateArn} was successfully
deleted.");
                }
                Console.WriteLine(new string('-', 80));
            }

            // Step 13: Delete the AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("13. Delete the AWS IoT Thing.");
            var deleteThing = "y";
```

```
            if (IsInteractive)
            {
                Console.Write($"Do you want to delete the IoT Thing? (y/n)");
                deleteThing = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Deleting IoT Thing {thingName}...");
            }

            if (deleteThing?.ToLower() == "y")
            {
                await iotWrapper.DeleteThingAsync(thingName);
                Console.WriteLine($"Deleted Thing {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 14: Clean up CloudFormation stack
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                Console.WriteLine(new string('-', 80));
                Console.WriteLine("14. Clean up CloudFormation stack.");
                Console.WriteLine("Deleting the CloudFormation stack and all
resources...");

                var cleanup = !IsInteractive || GetYesNoResponse("Do you want to
delete the CloudFormation stack and all resources? (y/n) ");
                if (cleanup)
                {
                    var ruleCleanupSuccess = await
iotWrapper.DeleteTopicRuleAsync(ruleName);

                    var stackCleanupSuccess = await
DeleteCloudFormationStack(_stackName, cloudFormationClient, scenarioLogger);
                    if (ruleCleanupSuccess && stackCleanupSuccess)
                    {
                        Console.WriteLine("Successfully cleaned up CloudFormation
stack and all resources.");
                    }
                    else
                    {
                        Console.WriteLine("Some cleanup operations failed. Check
the logs for details.");
                    }
```

```
                }
                else
                {
                    Console.WriteLine($"Resources will remain. Stack name:
 {_stackName}");
                }
                Console.WriteLine(new string('-', 80));
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "Error occurred during scenario
 execution.");

            // Cleanup on error
            if (!string.IsNullOrEmpty(certificateArn) && !
string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DetachThingPrincipalAsync(thingName,
 certificateArn);
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
                }
            }

            if (!string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DeleteThingAsync(thingName);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during Thing
 cleanup.");
                }
            }

            // Clean up CloudFormation stack on error
```

```
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                try
                {
                    await _iotWrapper.DeleteTopicRuleAsync(ruleName);
                    await DeleteCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during
CloudFormation stack cleanup.");
                }
            }

            throw;
        }
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = new List<string>
{ Capability.CAPABILITY_NAMED_IAM }
            };

            var response = await cloudFormationClient.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await
WaitForStackCompletion(response.StackId, cloudFormationClient, scenarioLogger);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created
successfully.");
                    return true;
                }
                else
                {
                    scenarioLogger.LogError($"CloudFormation stack creation
failed: {stackName}");
                    return false;
                }
            }
            else
            {
                scenarioLogger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            scenarioLogger.LogWarning($"CloudFormation stack '{stackName}'
already exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName,
cloudFormationClient, scenarioLogger);
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
```

```
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }
```

```
        scenarioLogger.LogError("Timed out waiting for CloudFormation stack
creation to complete.");
        return false;
    }

    /// <summary>
    /// Gets the outputs from the CloudFormation stack.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>A dictionary of stack outputs.</returns>
    private static async Task<Dictionary<string, string>?> GetStackOutputs(string
stackName, IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics>
scenarioLogger)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            var response = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (response.Stacks.Count > 0)
            {
                var outputs = new Dictionary<string, string>();
                foreach (var output in response.Stacks[0].Outputs)
                {
                    outputs[output.OutputKey] = output.OutputValue;
                }
                return outputs;
            }

            return null;
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"Failed to get stack outputs for
{stackName}");
            return null;
```

```
        }
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        try
        {
            var request = new DeleteStackRequest
            {
                StackName = stackName
            };

            await cloudFormationClient.DeleteStackAsync(request);
            Console.WriteLine($"CloudFormation stack '{stackName}' is being
deleted. This may take a few minutes.");

            bool stackDeleted = await WaitForStackDeletion(stackName,
cloudFormationClient, scenarioLogger);

            if (stackDeleted)
            {
                Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
                return true;
            }
            else
            {
                scenarioLogger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
                return false;
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deleting the
CloudFormation stack: {stackName}");
```

```
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }
```

```
            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
 be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError($"Timed out waiting for CloudFormation stack
 '{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (IsInteractive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
 Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = new System.Text.RegularExpressions.Regex("[a-zA-Z][-
a-zA-Z0-9]*");
                if (!regex.IsMatch(input))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
 {_stackName}");
                    return _stackName;
                }
                return input;
```

```
            }
        }
        return _stackName;
    }
}
```

Uma classe wrapper para métodos do AWS IoT SDK.

```
/// <summary>
/// Wrapper methods to use Amazon IoT Core with .NET.
/// </summary>
public class IoTWrapper
{
    private readonly IAmazonIoT _amazonIoT;
    private readonly IAmazonIotData _amazonIotData;
    private readonly ILogger<IoTWrapper> _logger;

    /// <summary>
    /// Constructor for the IoT wrapper.
    /// </summary>
    /// <param name="amazonIoT">The injected IoT client.</param>
    /// <param name="amazonIotData">The injected IoT Data client.</param>
    /// <param name="logger">The injected logger.</param>
    public IoTWrapper(IAmazonIoT amazonIoT, IAmazonIotData amazonIotData,
 ILogger<IoTWrapper> logger)
    {
        _amazonIoT = amazonIoT;
        _amazonIotData = amazonIotData;
        _logger = logger;
    }

    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
```

```
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
{response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or
null if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };

            var response = await
_amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
```

```
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
 {ex.Message}");
            return false;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
```

```
        }
    }

    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing
failed.</returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
```

```
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
{ex.Message}");
            return new List<Certificate>();
        }
    }

    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
shadowPayload)
    {
        try
        {
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };

            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
```

```
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</
returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
 snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
```

```
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteTopicRuleAsync(string ruleName)
    {
        try
        {
            var request = new DeleteTopicRuleRequest
            {
                RuleName = ruleName,
            };

            await _amazonIoT.DeleteTopicRuleAsync(request);
            _logger.LogInformation($"Deleted IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogWarning($"Rule {ruleName} not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }
```

```
    /// <summary>
    /// Lists all IoT topic rules.
    /// </summary>
    /// <returns>List of topic rules, or empty list if listing failed.</returns>
    public async Task<List<TopicRuleListItem>> ListTopicRulesAsync()
    {
        try
        {
            var request = new ListTopicRulesRequest();
            var response = await _amazonIoT.ListTopicRulesAsync(request);

            _logger.LogInformation($"Retrieved {response.Rules.Count} IoT
rules");
            return response.Rules;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list topic rules. Here's why:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
    }

    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
```

```
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
        try
```

```
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });

            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if
it's ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
```

```
                        _logger.LogInformation($"Search found
 {response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
 wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
 (ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
 wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready
 after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
 why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
```

```
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);
```

```
            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
```

```
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
```

```
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration

- UpdateThing

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie qualquer AWS IoT coisa.

```
    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }
```

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
```

```
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Gerar e anexar um certificado de dispositivo.

```
    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing?
 (y/n) ")) {
        Aws::String outputFolder;
        if (askYesNoQuestion(
                "Would you like to save the certificate and keys to file? (y/n)
 ")) {
            outputFolder = std::filesystem::current_path();
            outputFolder += "/device_keys_and_certificates";

            std::filesystem::create_directories(outputFolder);

            std::cout << "The certificate and keys will be saved to the folder: "
                      << outputFolder << std::endl;
        }

        if (!createKeysAndCertificate(outputFolder, certificateARN,
 certificateID,
                                      clientConfiguration)) {
            std::cerr << "Exiting because createKeysAndCertificate failed."
                      << std::endl;
            cleanup(thingName, "", "", "", "", false, clientConfiguration);
            return false;
        }

        std::cout << "\nNext, the certificate will be attached to the thing.\n"
                  << std::endl;
        if (!attachThingPrincipal(certificateARN, thingName,
 clientConfiguration)) {
            std::cerr << "Exiting because attachThingPrincipal failed." <<
 std::endl;
```

```
            cleanup(thingName, certificateARN, certificateID, "", "",
                    false,
                    clientConfiguration);
            return false;
        }
    }
```

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
```

```
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
```

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

    return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Execute várias operações na AWS IoT coisa.

```
    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
 "v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
```

```
        return false;
    }

    printAsterisksLine();

    std::cout << "Now an endpoint will be retrieved for your account.\n" <<
std::endl;
    std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
Locator that serves as the entry point\n"
    << "for communication between IoT devices and the AWS IoT service." <<
std::endl;

    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String endpoint;
    if (!describeEndpoint(endpoint, clientConfiguration)) {
        std::cerr << "Exiting because getEndpoint failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
    printAsterisksLine();

    std::cout << "Now the certificates in your account will be listed." <<
std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    if (!listCertificates(clientConfiguration)) {
        std::cerr << "Exiting because listCertificates failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
    std::cout << "A thing shadow refers to a feature that enables you to create a
virtual representation, or \"shadow,\"\n"
    << "of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between\n"
    << "the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow." << std::endl;
```

```
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
        std::cerr << "Exiting because updateThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now, the state information for the shadow will be retrieved.\n"
 << std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String shadowState;
    if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
        std::cerr << "Exiting because getThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

    printAsterisksLine();

    std::cout << "A rule with now be added to to the thing.\n" << std::endl;
    std::cout << "Any user who has permission to create rules will be able to
 access data processed by the rule." << std::endl;
    std::cout << "In this case, the rule will use an Simple Notification Service
 (SNS) topic and an IAM rule." << std::endl;
    std::cout << "These resources will be created using a CloudFormation
 template." << std::endl;
    std::cout << "Stack creation may take a few minutes." << std::endl;

    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    Aws::Map<Aws::String, Aws::String> outputs
 =createCloudFormationStack(STACK_NAME,clientConfiguration);
    if (outputs.empty()) {
        std::cerr << "Exiting because createCloudFormationStack failed." <<
 std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
```

```
        return false;
    }

    // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
    auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
    auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
    if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
        std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
        "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."
 << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM '";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += "'";

    printAsterisksLine();

    std::cout << "Now a rule will be created.\n" << std::endl;
    std::cout << "Rules are an administrator-level action. Any user who has
 permission\n"
                << "to create rules will be able to access data processed by the
 rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
 std::endl;
    std::cout << "For more information on IoT SQL, see https://
docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" <<
 std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
 clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }
```

```
    printAsterisksLine();

    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/
latest/developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
 account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
       << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/
latest/developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account?
 (y/n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

 thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGI

 thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnec
        // The ThingGroupIndexingConfiguration object is ignored if not set.
        Aws::IoT::Model::ThingGroupIndexingConfiguration
 thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
 thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                    ruleName, false,
```

```
                    clientConfiguration);
            return false;
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }
```

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
```

```
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
```

```
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
 iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
```

```
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
```

```
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome =
 iotClient.ListTopicRules(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
 outcome.GetResult();
            allRules.insert(allRules.end(),
                            result.GetRules().cbegin(),
                            result.GetRules().cend());
```

```
            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "ListTopicRules error: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
              << std::endl;
    for (auto &rule: allRules) {
        std::cout << "  Rule name: " << rule.GetRuleName() << ", rule ARN: "
                  << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
```

```
        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

Limpe recursos.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
 &certificateARN,
                     const Aws::String &certificateID, const Aws::String
 &stackName,
                     const Aws::String &ruleName, bool askForConfirmation,
                     const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
```

```
                                askYesNoQuestion("Delete the rule '" + ruleName +
                                                 "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
    }

    Aws::CloudFormation::CloudFormationClient
 cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
                                askYesNoQuestion(
                                        "Delete the CloudFormation stack '" +
 stackName +
                                        "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
                                    askYesNoQuestion("Delete the certificate '" +
                                                     certificateARN + "'? (y/n)
 "))) {
        result &= detachThingPrincipal(certificateARN, thingName,
 clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                                askYesNoQuestion("Delete the thing '" + thingName
 +
                                                 "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
```

```
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration

- UpdateThing

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS IoT recursos.

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
```

```
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series
of steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality
for creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS
IoT capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
```

```
        System.out.println(DASHES);

        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service
that can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName
+"? (y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a
pivotal advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
```

```
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon
Web Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a
virtual representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For
example, you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON
format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate
for " +thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
```

```
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
```

```
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe wrapper para métodos do AWS IoT SDK.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.Certificate;
import
 software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
```

```
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
```

```
        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
```

```
    * This method initiates an asynchronous request to create an IoT
certificate.
    * If the request is successful, it prints the certificate details and
returns the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String certificatePem = response.certificatePem();
               certificateArn[0] = response.certificateArn();

               // Print the details.
               System.out.println("\nCertificate:");
               System.out.println(certificatePem);
               System.out.println("\nCertificate ARN:");
               System.out.println(certificateArn[0]);

           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               }
           }
       });

       future.join();
       return certificateArn[0];
   }

   /**
    * Creates an IoT Thing with the specified name asynchronously.
    *
    * @param thingName The name of the IoT Thing to create.
    *
```

```
    * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
    * If the request is successful, it prints the name of the thing and its ARN
value.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               }
           }
       });

       future.join();
   }

   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
    * If the request is successful, it prints a confirmation message and
additional information about the Thing.
    * If an exception occurs, it prints the error message.
    */
```

```
   public void attachCertificateToThing(String thingName, String certificateArn)
{
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing
successfully.");

               // Print additional information about the Thing.
               describeThing(thingName);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               } else {
                   System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
                       attachResponse.sdkHttpResponse().statusCode());
               }
           }
       });

       future.join();
   }

   /**
    * Describes an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to describe an IoT Thing.
    * If the request is successful, it prints the Thing details.
```

```
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an
IoT Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
```

```
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
```

```
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }

    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);
```

```
        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a
Thing's shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }
```

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
                List<TopicRuleListItem> ruleList =
listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
```

```
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
```

```
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed
from " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
```

```
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
```

```
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
 getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") +
 1);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachThingPrincipal

- CreateKeysAndCertificate
- CreateThing
- CreateTopicRule
- DeleteCertificate
- DeleteThing
- DeleteTopicRule
- DescribeEndpoint
- DescribeThing
- DetachThingPrincipal
- ListCertificates
- ListThings
- SearchIndex
- UpdateIndexingConfiguration
- UpdateThing

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
```

```
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development
 environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-
getting-started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
```

```
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an
IoT thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that
can be associated with a physical device.
        """.trimIndent(),
    )
```

```
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
        A device certificate performs a role in securing the communication
between devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
```

```
   println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
   println(
       """
       An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
       """.trimIndent(),
   )
   print("Press Enter to continue...")
   scanner.nextLine()
   val endpointUrl = describeEndpoint()
   println(DASHES)

   println(DASHES)
   println("5. List your AWS IoT certificates")
   print("Press Enter to continue...")
   scanner.nextLine()
   if (certificateArn!!.isNotEmpty()) {
       listCertificates()
   } else {
       println("You did not create a certificates. Skipping this step.")
   }
   println(DASHES)

   println(DASHES)
   println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
   println(
       """
       A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
       of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
       the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow.

       """.trimIndent(),
   )
   print("Press Enter to continue...")
   scanner.nextLine()
   updateShawdowThing(thingName)
   println(DASHES)
```

```
    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName?
(y/n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
            println("11. You selected to detach amd delete the certificate.")
```

```
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
 delete.")
    }
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
```

```
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
```

```
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse =
 iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
```

```
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
```

```
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }
```

```
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe de invólucro de IoT para gerenciar as operações.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
```

```
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)


    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
```

```
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

```
    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response =
self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]


    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
```

```
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
```

```
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_topic_rules(self):
        """
        Lists AWS IoT topic rules.

        :return: The list of topic rules.
        """
        try:
            rules = []
            paginator = self.iot_client.get_paginator("list_topic_rules")
            for page in paginator.paginate():
                rules.extend(page["rules"])
            logger.info("Retrieved %s topic rules.", len(rules))
            return rules
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
```

```
            else:
                logger.error(
                    "Couldn't list topic rules. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])


    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
```

```
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise


    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
            response =
self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
```

```
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

Execute um cenário interativo demonstrando os fundamentos da IoT.

```
class IoTScenario:
    """Runs an interactive scenario that shows how to use AWS IoT."""

    is_interactive = True

    def __init__(self, iot_wrapper, iot_data_client, cfn_client,
 stack_name="IoTBasicsStack", template_path=None):
        """
        :param iot_wrapper: An instance of the IoTWrapper class.
        :param iot_data_client: A Boto3 IoT Data Plane client.
        :param cfn_client: A Boto3 CloudFormation client.
        :param stack_name: Name for the CloudFormation stack.
        :param template_path: Path to the CloudFormation template file.
        """
        self.iot_wrapper = iot_wrapper
        self.iot_data_client = iot_data_client
        self.cfn_client = cfn_client
        self.thing_name = None
        self.certificate_arn = None
        self.certificate_id = None
        self.rule_name = None
        self.stack_name = stack_name
        self.template_path = template_path or "../../../scenarios/basics/iot/
iot_usecase/resources/cfn_template.yaml"

    def _deploy_stack(self):
        """Deploy CloudFormation stack and return outputs."""
        with open(self.template_path, "r") as f:
            template_body = f.read()
```

```
        try:
            self.cfn_client.create_stack(
                StackName=self.stack_name,
                TemplateBody=template_body,
                Capabilities=["CAPABILITY_NAMED_IAM"]
            )

            waiter = self.cfn_client.get_waiter("stack_create_complete")
            waiter.wait(StackName=self.stack_name)

            response = self.cfn_client.describe_stacks(StackName=self.stack_name)
            outputs = {output["OutputKey"]: output["OutputValue"]
                       for output in response["Stacks"][0]["Outputs"]}
            return outputs["SNSTopicArn"], outputs["RoleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                response =
self.cfn_client.describe_stacks(StackName=self.stack_name)
                outputs = {output["OutputKey"]: output["OutputValue"]
                           for output in response["Stacks"][0]["Outputs"]}
                return outputs["SNSTopicArn"], outputs["RoleArn"]
            raise

    def _cleanup_stack(self):
        """Delete CloudFormation stack."""
        try:
            self.cfn_client.delete_stack(StackName=self.stack_name)
            waiter = self.cfn_client.get_waiter("stack_delete_complete")
            waiter.wait(StackName=self.stack_name)
            print("CloudFormation stack deleted successfully.")
        except ClientError as err:
            logger.error(f"Failed to delete stack: {err}")

    def run_scenario(self, thing_name, rule_name):
        """
        Runs the IoT basics scenario.

        :param thing_name: The name of the thing to create.
        :param rule_name: The name of the topic rule to create.
        """
        print("-" * 88)
        print("Welcome to the AWS IoT basics scenario!")
        print("-" * 88)
```

```
        print(
            "This scenario demonstrates how to interact with AWS IoT using the
AWS SDK for Python (Boto3).\n"
            "AWS IoT provides secure, bi-directional communication between
Internet-connected devices\n"
            "and the AWS cloud. You can manage device connections, process device
data, and build IoT applications.\n"
        )

        self.thing_name = thing_name
        self.rule_name = rule_name

        try:
            print("\nDeploying CloudFormation stack...")
            sns_topic_arn, role_arn = self._deploy_stack()
            print(f"Stack deployed. SNS Topic: {sns_topic_arn}")

            input("\nNext, we'll create an AWS IoT thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("1. Create an AWS IoT thing")
            print("-" * 88)
            response = self.iot_wrapper.create_thing(thing_name)
            print(f"Created thing: {response['thingName']}")
            print(f"Thing ARN: {response['thingArn']}")

            input("\nNext, we'll list things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("2. List things")
            print("-" * 88)
            things = self.iot_wrapper.list_things()
            print(f"Found {len(things)} thing(s) in your account")
            for thing in things[:5]:  # Show first 5
                print(f"  Thing name: {thing['thingName']}")

            input("\nNext, we'll generate a device certificate. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("3. Generate a device certificate")
            print("-" * 88)
            cert_response = self.iot_wrapper.create_keys_and_certificate()
            self.certificate_arn = cert_response["certificateArn"]
            self.certificate_id = cert_response["certificateId"]
```

```
            print(f"Created certificate: {self.certificate_id}")

            input("\nNext, we'll attach the certificate to the thing. Press Enter
to continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("4. Attach the certificate to the thing")
            print("-" * 88)
            self.iot_wrapper.attach_thing_principal(thing_name,
self.certificate_arn)
            print(f"Attached certificate to thing: {thing_name}")

            input("\nNext, we'll update the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("5. Update the thing shadow")
            print("-" * 88)
            shadow_state = {"state": {"reported": {"temperature": 25, "humidity":
50}}}
            self.iot_wrapper.update_thing_shadow(thing_name, shadow_state)
            print(f"Updated shadow for thing: {thing_name}")

            input("\nNext, we'll get the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("6. Get the thing shadow")
            print("-" * 88)
            shadow = self.iot_wrapper.get_thing_shadow(thing_name)
            print(f"Shadow state: {json.dumps(shadow['state'], indent=2)}")

            input("\nNext, we'll get the AWS IoT endpoint. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("7. Get the AWS IoT endpoint")
            print("-" * 88)
            endpoint = self.iot_wrapper.describe_endpoint()
            print(f"IoT endpoint: {endpoint}")

            input("\nNext, we'll list certificates. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("8. List certificates")
            print("-" * 88)
            certificates = self.iot_wrapper.list_certificates()
            print(f"Found {len(certificates)} certificate(s)")
```

```
            for cert in certificates:
                print(f"  Certificate ID: {cert['certificateId']}")
                print(f"  Certificate ARN: {cert['certificateArn']}")
                print(f"  Status: {cert['status']}")

            input("\nNext, we'll create a topic rule. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("9. Create a topic rule")
            print("-" * 88)
            self.iot_wrapper.create_topic_rule(
                rule_name, f"device/{thing_name}/data", sns_topic_arn, role_arn
            )
            print(f"Created topic rule: {rule_name}")

            input("\nNext, we'll list topic rules. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("10. List topic rules")
            print("-" * 88)
            rules = self.iot_wrapper.list_topic_rules()
            print(f"Found {len(rules)} topic rule(s)")
            for rule in rules:
                print(f"  Rule name: {rule['ruleName']}")
                print(f"  Rule ARN: {rule['ruleArn']}")

            input("\nNext, we'll configure thing indexing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("11. Configure thing indexing")
            print("-" * 88)
            self.iot_wrapper.update_indexing_configuration()
            print("Enabled thing indexing")
            print("Waiting for indexing to be ready...")
            time.sleep(10)

            input("\nNext, we'll search for things. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("12. Search for things")
            print("-" * 88)
            try:
                things = self.iot_wrapper.search_index(f"thingName:{thing_name}")
                if things:
```

```
                    print(f"Found {len(things)} thing(s) matching the query")
                    for thing in things:
                        print(f"  Thing name: {thing.get('thingName', 'N/A')}")
                        print(f"  Thing ID: {thing.get('thingId', 'N/A')}")
                else:
                    print("No things found. Indexing may take a few minutes.")
            except ClientError as err:
                if err.response["Error"]["Code"] in [
                    "IndexNotReadyException",
                    "InvalidRequestException",
                ]:
                    print("Search index not ready yet. This is expected.")
                else:
                    raise

        except ClientError as err:
            logger.error(
                "Scenario failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        finally:
            self._cleanup()

    def _cleanup(self):
        """Cleans up resources created during the scenario."""
        if not self.thing_name:
            return

        print("\n" + "-" * 88)
        print("Cleanup")
        print("-" * 88)

        if q.ask("Do you want to delete the resources? (y/n) ", q.is_yesno):
            try:
                if self.certificate_arn:
                    print(f"Detaching certificate from thing: {self.thing_name}")
                    self.iot_wrapper.detach_thing_principal(
                        self.thing_name, self.certificate_arn
                    )

                if self.certificate_id:
                    print(f"Deleting certificate: {self.certificate_id}")
```

```
                self.iot_wrapper.delete_certificate(self.certificate_id)

            if self.thing_name:
                print(f"Deleting thing: {self.thing_name}")
                self.iot_wrapper.delete_thing(self.thing_name)

            if self.rule_name:
                print(f"Deleting topic rule: {self.rule_name}")
                self.iot_wrapper.delete_topic_rule(self.rule_name)

            self._cleanup_stack()
            print("Resources deleted successfully.")
        except ClientError as err:
            logger.error(
                "Cleanup failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
    else:
        print("Resources will remain in your account.")

    print("\n" + "-" * 88)
    print("Thanks for using AWS IoT!")
    print("-" * 88)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing

- DetachThingPrincipal
- ListCertificates
- ListThings
- SearchIndex
- UpdateIndexingConfiguration
- UpdateThing

Ações para AWS IoT usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS IoT ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a AWS IoT API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para AWS IoT usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS IoT.

Exemplos

- Use AttachThingPrincipalcom um AWS SDK ou CLI
- Use CreateKeysAndCertificatecom um AWS SDK ou CLI
- Use CreateThingcom um AWS SDK ou CLI
- Use CreateTopicRulecom um AWS SDK ou CLI
- Use DeleteCertificatecom um AWS SDK ou CLI
- Use DeleteThingcom um AWS SDK ou CLI
- Use DeleteTopicRulecom um AWS SDK ou CLI
- Use DescribeEndpointcom um AWS SDK ou CLI
- Use DescribeThingcom um AWS SDK ou CLI
- Use DetachThingPrincipalcom um AWS SDK ou CLI
- Use ListCertificatescom um AWS SDK ou CLI
- Use ListThingscom um AWS SDK ou CLI

- Use SearchIndexcom um AWS SDK ou CLI
- Use UpdateIndexingConfigurationcom um AWS SDK ou CLI
- Use UpdateThingcom um AWS SDK ou CLI

Use **`AttachThingPrincipal`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AttachThingPrincipal`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de mensagens da Internet das Coisas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
```

```
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte AttachThingPrincipala Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AttachThingPrincipala Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para anexar um certificado a um item

O `attach-thing-principal` exemplo a seguir anexa um certificado à MyTemperatureSensor coisa. O certificado é identificado por um ARN. Você pode encontrar o ARN de um certificado no console de IoT AWS .

```
aws iot attach-thing-principal \
    --thing-name MyTemperatureSensor \
    --principal arn:aws:iot:us-
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

Este comando não produz saída.

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte AttachThingPrincipalem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
    * If the request is successful, it prints a confirmation message and
additional information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn)
{
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
```

```
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing
successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte AttachThingPrincipala Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun attachCertificateToThing(
```

```
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- Para obter detalhes da API, consulte a AttachThingPrincipalreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
```

```
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a AttachThingPrincipalReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       lo_iot->attachthingprincipal(
         iv_thingname = iv_thing_name
         iv_principal = iv_principal ).
       MESSAGE |Principal attached to IoT thing '{ iv_thing_name }'.| TYPE 'I'.
     CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
       MESSAGE |Resource not found when attaching principal to
'{ iv_thing_name }'.| TYPE 'I'.
       RAISE EXCEPTION lo_ex.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
       MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_svc_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a AttachThingPrincipalreferência da API AWS SDK for SAP ABAP.

Use **`CreateKeysAndCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateKeysAndCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de mensagens da Internet das Coisas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Creates a device certificate for AWS IoT.
   /// </summary>
   /// <returns>The certificate details including ARN and certificate PEM, or
null if creation failed.</returns>
   public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
   {
       try
       {
           var request = new CreateKeysAndCertificateRequest
           {
               SetAsActive = true
           };

           var response = await
_amazonIoT.CreateKeysAndCertificateAsync(request);
           _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
           return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
       }
       catch (Amazon.IoT.Model.ThrottlingException ex)
       {
           _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
           return null;
       }
       catch (Exception ex)
       {
           _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
```

```
            return null;
        }
    }
```

- Para obter detalhes da API, consulte CreateKeysAndCertificatea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
```

```
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateKeysAndCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar um par de chaves RSA e emitir um certificado X.509

O seguinte `create-keys-and-certificate` cria um par de chaves RSA de 2048 bits e emite X.509 um certificado usando a chave pública emitida. Como essa é a única vez que a AWS IoT fornece a chave privada para esse certificado, certifique-se de mantê-lo em um local seguro.

```
aws iot create-keys-and-certificate \
    --certificate-pem-outfile "myTest.cert.pem" \
    --public-key-outfile "myTest.public.key" \
    --private-key-outfile "myTest.private.key"
```

Saída:

```
{
```

```
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nMMEXAMPLEuuN/
dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y
+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQ
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB
\n-----END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omittted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

Para obter mais informações, consulte Criar e registrar um certificado de dispositivo de AWS IoT no Guia do desenvolvedor de AWS IoT.

- Para obter detalhes da API, consulte CreateKeysAndCertificateem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an IoT certificate asynchronously.
    *
    * @return The ARN of the created certificate.
    * <p>
    * This method initiates an asynchronous request to create an IoT
certificate.
    * If the request is successful, it prints the certificate details and
returns the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String certificatePem = response.certificatePem();
               certificateArn[0] = response.certificateArn();

               // Print the details.
               System.out.println("\nCertificate:");
               System.out.println(certificatePem);
               System.out.println("\nCertificate ARN:");
               System.out.println(certificateArn[0]);

           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
```

```
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }
```

- Para obter detalhes da API, consulte CreateKeysAndCertificatea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- Para obter detalhes da API, consulte a CreateKeysAndCertificatereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
 self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateKeysAndCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       oo_result = lo_iot->createkeysandcertificate( iv_setasactive =
abap_true ).
       MESSAGE |Certificate created: { oo_result->get_certificateid( ) }| TYPE
'I'.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateKeysAndCertificatereferência da API AWS SDK for SAP ABAP.

Use **`CreateThing`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateThing`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de mensagens da Internet das Coisas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
```

```
            _logger.LogInformation($"Created Thing {thingName} with ARN
 {response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
 {ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte CreateThinga Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
```

```
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateThinga Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: criar um registro de item no registro

O `create-thing` exemplo a seguir cria uma entrada para um dispositivo no registro de coisas da AWS IoT.

```
aws iot create-thing \
    --thing-name SampleIoTThing
```

Saída:

```
{
    "thingName": "SampleIoTThing",
    "thingArn": "arn:aws:iot:us-west-2: 123456789012:thing/SampleIoTThing",
    "thingId": " EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "
}
```

Exemplo 2: definir um item associado a um tipo de item

O exemplo `create-thing` a seguir cria um item com o tipo de item e seus atributos especificados.

```
aws iot create-thing \
    --thing-name "MyLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Saída:

```
{
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797"
}
```

Para obter mais informações, consulte Como gerenciar itens com o registro e Tipos de itens no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte CreateThingem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with
 the specified name.
     * If the request is successful, it prints the name of the thing and its ARN
 value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
```

```
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
 getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The
 ARN value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte CreateThinga Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
```

```
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Para obter detalhes da API, consulte a CreateThingreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.
```

```
        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateThingReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
```

```
        oo_result = lo_iot->creatething(
          iv_thingname = iv_thing_name ).
        MESSAGE |IoT thing created: { oo_result->get_thingname( ) } ARN:
 { oo_result->get_thingarn( ) }| TYPE 'I'.
      CATCH /aws1/cx_iotresrcalrdyexistsex.
        MESSAGE |IoT thing '{ iv_thing_name }' already exists.| TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateThingreferência da API AWS SDK for SAP ABAP.

Use **`CreateTopicRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateTopicRule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
```

```
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateTopicRulea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
```

```
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTopicRulea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar uma regra que envie um alerta do Amazon SNS

O exemplo `create-topic-rule` a seguir cria uma regra que envia uma mensagem do Amazon SNS quando as leituras do nível de umidade do solo, encontradas em uma sombra do dispositivo, estão baixas.

```
aws iot create-topic-rule \
    --rule-name "LowMoistureRule" \
    --topic-rule-payload file://plant-rule.json
```

O exemplo exige que o código JSON a seguir seja salvo em um arquivo chamado `plant-rule.json`:

```
{
```

```
    "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE
 state.reported.moisture = 'low'\n",
    "description": "Sends an alert whenever soil moisture level readings are too
 low.",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
            "sns": {
                "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyRPiLowMoistureTopic",
                "roleArn": "arn:aws:iam::123456789012:role/service-role/
MyRPiLowMoistureTopicRole",
                "messageFormat": "RAW"
            }
    }]
}
```

Este comando não produz saída.

Para obter mais informações, consulte Criação de uma regra de AWS IoT no Guia para desenvolvedores de AWS IoT.

- Para obter detalhes da API, consulte CreateTopicRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
 actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
```

```
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
```

```
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte CreateTopicRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
```

```
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- Para obter detalhes da API, consulte a CreateTopicRulereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client
```

```
    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateTopicRuleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Build the SNS action that will receive messages matching the rule.
    DATA lo_sns_action TYPE REF TO /aws1/cl_iotsnsaction.
    CREATE OBJECT lo_sns_action
      EXPORTING
        iv_targetarn = iv_sns_action_arn
        iv_rolearn   = iv_role_arn.

    DATA lo_action TYPE REF TO /aws1/cl_iotaction.
    CREATE OBJECT lo_action
      EXPORTING
        io_sns = lo_sns_action.

    DATA lt_actions TYPE /aws1/cl_iotaction=>tt_actionlist.
    APPEND lo_action TO lt_actions.

    " iv_topic = 'my/iot/topic' - The MQTT topic pattern to match
    DATA lo_payload TYPE REF TO /aws1/cl_iottopicrulepayload.
    CREATE OBJECT lo_payload
      EXPORTING
        iv_sql     = |SELECT * FROM '{ iv_topic }'|
        it_actions = lt_actions.

    lo_iot->createtopicrule(
      iv_rulename         = iv_rule_name
      io_topicrulepayload = lo_payload ).
    MESSAGE |IoT topic rule created: { iv_rule_name }| TYPE 'I'.
  CATCH /aws1/cx_iotresrcalrdyexistsex.
    MESSAGE |Topic rule '{ iv_rule_name }' already exists.| TYPE 'I'.
```

```
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
      MESSAGE lo_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTopicRulereferência da API AWS SDK for SAP ABAP.

Use **`DeleteCertificate`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteCertificate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de mensagens da Internet das Coisas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Deletes an IoT certificate.
/// </summary>
/// <param name="certificateId">The ID of the certificate to delete.</param>
/// <returns>True if successful, false otherwise.</returns>
public async Task<bool> DeleteCertificateAsync(string certificateId)
{
    try
    {
        // First, update the certificate to inactive state
        var updateRequest = new UpdateCertificateRequest
```

```
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteCertificatea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir um certificado de um dispositivo

O exemplo de `delete-certificate` a seguir exclui o certificado do dispositivo com o ID especificado.

```
aws iot delete-certificate \
    --certificate-
id c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbee1428d216d54d53ac9
```

Este comando não produz saída.

Para obter mais informações, consulte a DeleteCertificateReferência da API de AWS IoT.

- Para obter detalhes da API, consulte DeleteCertificateem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
```

```
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DeleteCertificate a Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteCertificatereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)
```

```
    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteCertificateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
```

```
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        " Certificates must be deactivated before they can be deleted.
        lo_iot->updatecertificate(
          iv_certificateid = iv_certificate_id
          iv_newstatus     = 'INACTIVE' ).
        lo_iot->deletecertificate( iv_certificateid = iv_certificate_id ).
        MESSAGE |Certificate deleted: { iv_certificate_id }| TYPE 'I'.
      CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
        MESSAGE |Certificate '{ iv_certificate_id }' not found.| TYPE 'I'.
        RAISE EXCEPTION lo_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
        MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_svc_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCertificatereferência da API AWS SDK for SAP ABAP.

Use **`DeleteThing`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteThing`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de mensagens da Internet das Coisas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteThinga Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteThinga Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como exibir informações detalhadas sobre um item

O `delete-thing` exemplo a seguir exclui algo do registro de AWS IoT da sua conta. AWS

era delete-thing --thing-name" FourthBulb

Este comando não produz saída.

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte DeleteThingem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DeleteThinga Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteThingreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteThingReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    lo_iot->deletething( iv_thingname = iv_thing_name ).
    MESSAGE |IoT thing deleted: { iv_thing_name }| TYPE 'I'.
  CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE |IoT thing '{ iv_thing_name }' not found.| TYPE 'I'.
    RAISE EXCEPTION lo_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
    MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_svc_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteThingreferência da API AWS SDK for SAP ABAP.

Use **`DeleteTopicRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteTopicRule`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTopicRulea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir uma regra

O exemplo de `delete-topic-rule` a seguir exclui a regra especificada.

```
aws iot delete-topic-rule \
    --rule-name "LowMoistureRule"
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir uma regra no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte DeleteTopicRuleem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""
```

```
    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTopicRuleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    lo_iot->deletetopicrule( iv_rulename = iv_rule_name ).
    MESSAGE |IoT topic rule deleted: { iv_rule_name }| TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTopicRulereferência da API AWS SDK for SAP ABAP.

Use **`DescribeEndpoint`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeEndpoint`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de mensagens da Internet das Coisas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte DescribeEndpointa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeEndpointa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: Para obter seu AWS endpoint atual

O `describe-endpoint` exemplo a seguir recupera o AWS endpoint padrão ao qual todos os comandos são aplicados.

```
aws iot describe-endpoint
```

Saída:

```
{
    "endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

Para obter mais informações, consulte o DescribeEndpointGuia do desenvolvedor de AWS IoT.

Exemplo 2: como obter o endpoint do ATS

O exemplo de `describe-endpoint` a seguir recupera o endpoint do Amazon Trust Services (ATS).

```
aws iot describe-endpoint \
    --endpoint-type iot:Data-ATS
```

Saída:

```
{
```

```
    "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
}
```

Para obter mais informações, consulte X.509 Certificados e AWS IoT no Guia do desenvolvedor de AWS IoT.

- Para obter detalhes da API, consulte DescribeEndpointem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";
```

```
                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- Para obter detalhes da API, consulte DescribeEndpointa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
```

```
        return fullEndpoint
    }
}
```

- Para obter detalhes da API, consulte a DescribeEndpointreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
```

```
        """
        try:
            response =
 self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]
```

- Para obter detalhes da API, consulte a DescribeEndpointReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error>
 {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;
```

```
    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeEndpointreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    " iv_endpoint_type = 'iot:Data-ATS' - Endpoint type for data operations
    TRY.
        DATA(lo_result) = lo_iot->describeendpoint( iv_endpointtype =
iv_endpoint_type ).
        ov_endpoint_address = lo_result->get_endpointaddress( ).
        MESSAGE |Endpoint address: { ov_endpoint_address }| TYPE 'I'.
      CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
        MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
        RAISE EXCEPTION lo_throttle_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeEndpointreferência da API AWS SDK for SAP ABAP.

Use **`DescribeThing`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeThing`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
 iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "'" <<
 std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
```

```
                  << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
 attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeThinga Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como exibir informações detalhadas sobre um item

O `describe-thing` exemplo a seguir exibe informações sobre uma coisa (dispositivo) definida no registro de AWS IoT da sua AWS conta.

como descrever a coisa --nome da coisa" MyLightBulb

Saída:

```
{
    "defaultClientId": "MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    },
```

```
    "version": 1
}
```

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte DescribeThingem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DescribeThinga Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
```

```
    }
}
```

- Para obter detalhes da API, consulte a DescribeThingreferência da API AWS SDK for Kotlin.

Use **`DetachThingPrincipal`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetachThingPrincipal`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de mensagens da Internet das Coisas

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
```

```
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
why: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DetachThingPrincipala Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DetachThingPrincipala Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para separar um certificate/principal de uma coisa

O exemplo `detach-thing-principal` a seguir remove um certificado que representa uma entidade principal do item especificado.

```
aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
```

Este comando não produz saída.

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte DetachThingPrincipalem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from
 an IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
 DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();
```

```
        CompletableFuture<DetachThingPrincipalResponse> future =
 getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed
 from " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DetachThingPrincipala Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
```

```
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DetachThingPrincipalreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def detach_thing_principal(self, thing_name, principal):
```

```
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DetachThingPrincipalReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
```

```
    TRY.
        lo_iot->detachthingprincipal(
          iv_thingname = iv_thing_name
          iv_principal = iv_principal ).
        MESSAGE |Principal detached from IoT thing '{ iv_thing_name }'.| TYPE
'I'.
      CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
        MESSAGE |Resource not found when detaching principal from
'{ iv_thing_name }'.| TYPE 'I'.
        RAISE EXCEPTION lo_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
        MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_svc_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetachThingPrincipalreferência da API AWS SDK for SAP ABAP.

Use **`ListCertificates`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListCertificates`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists all certificates associated with the account.
```

```
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing
failed.</returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
{ex.Message}");
            return new List<Certificate>();
        }
    }
```

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
```

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

    return true;
}
```

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: Para listar os certificados registrados em sua AWS conta

O exemplo de `list-certificates` a seguir lista os certificados registrados em sua conta. Se o limite de paginação for maior do que o padrão de 25, é possível usar o valor de resposta `nextMarker` desse comando e fornecê-lo ao próximo comando para obter o próximo lote de resultados. Repita até `nextMarker` retornar sem um valor.

```
aws iot list-certificates
```

Saída:

```
{
    "certificates": [
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
            "certificateId":
 "604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
            "status": "ACTIVE",
            "creationDate": 1556810537.617
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e
            "certificateId":
 "262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
            "status": "ACTIVE",
```

```
            "creationDate": 1546447050.885
        },
        {
            "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "certificateId":
 "b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "status": "ACTIVE",
            "creationDate": 1546292258.322
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42
            "certificateId":
 "7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
            "status": "ACTIVE",
            "creationDate": 1541457693.453
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3
            "certificateId":
 "54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
            "status": "ACTIVE",
            "creationDate": 1541113568.611
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e
            "certificateId":
 "4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
            "status": "ACTIVE",
            "creationDate": 1541022751.983
        }
    ]
}
```

- Para obter detalhes da API, consulte ListCertificatesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
```

```
    }
```

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListCertificatesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListCertificatesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Collect all certificates by following the pagination marker.
    DATA lv_marker TYPE /aws1/iotmarker.
    DATA lv_count  TYPE i.

    DO.
      oo_result = lo_iot->listcertificates( iv_marker = lv_marker ).
      lv_count  = lv_count + lines( oo_result->get_certificates( ) ).
      lv_marker = oo_result->get_nextmarker( ).
      IF lv_marker IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    MESSAGE |Retrieved { lv_count } IoT certificates.| TYPE 'I'.
  CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
    MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
    RAISE EXCEPTION lo_throttle_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListCertificatesreferência da API AWS SDK for SAP ABAP.

Use **`ListThings`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListThings`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;
```

```
            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }
```

- Para obter detalhes da API, consulte ListThingsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: listar todas os itens do registro

O `list-things` exemplo a seguir lista as coisas (dispositivos) definidas no registro de AWS IoT da sua AWS conta.

```
aws iot list-things
```

Saída:

```
{
    "things": [
        {
            "thingName": "ThirdBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/ThirdBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 2
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        },
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
        "thingName": "SampleIoTThing",
        "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
        "attributes": {},
        "version": 1
        }
```

```
    ]
}
```

Exemplo 2: como listar os itens definidos que possuem um atributo específico

O exemplo de `list-things` a seguir exibe uma lista dos itens que têm um atributo chamado `wattage`.

```
aws iot list-things \
    --attribute-name wattage
```

Saída:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        }
    ]
}
```

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte ListThingsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListThingsReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name:  {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type:  {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
```

```
        println!(
            "  ARN:   {}",
            thing.thing_arn.as_deref().unwrap_or_default()
        );
        println!();
    }

    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListThingsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        " Collect all things by following the pagination token.
        DATA lv_nexttoken TYPE /aws1/iotnexttoken.
        DATA lv_count     TYPE i.

        DO.
          oo_result    = lo_iot->listthings( iv_nexttoken = lv_nexttoken ).
          lv_count     = lv_count + lines( oo_result->get_things( ) ).
          lv_nexttoken = oo_result->get_nexttoken( ).
          IF lv_nexttoken IS INITIAL.
            EXIT.
          ENDIF.
        ENDDO.
```

```
        MESSAGE |Retrieved { lv_count } IoT things.| TYPE 'I'.
      CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
        MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
        RAISE EXCEPTION lo_throttle_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListThingsreferência da API AWS SDK for SAP ABAP.

Use **`SearchIndex`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SearchIndex`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
```

```
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
```

```
   /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
   private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
   {
       try
       {
           // Update indexing configuration to REGISTRY mode
           _logger.LogInformation("Setting up IoT search indexing
configuration...");
           await _amazonIoT.UpdateIndexingConfigurationAsync(
               new UpdateIndexingConfigurationRequest()
               {
                   ThingIndexingConfiguration = new ThingIndexingConfiguration()
                   {
                       ThingIndexingMode = ThingIndexingMode.REGISTRY
                   }
               });

           _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

           // Wait for the index to be set up - this can take some time
           const int maxRetries = 10;
           const int retryDelaySeconds = 10;

           for (int attempt = 1; attempt <= maxRetries; attempt++)
           {
               try
               {
                   _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                   await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                   // Try to get the current indexing configuration to see if
it's ready
                   var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                   if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                   {
                       // Try the search again
                       var request = new SearchIndexRequest
```

```
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready
after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }
```

- Para obter detalhes da API, consulte SearchIndexa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
```

```
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

- Para obter detalhes da API, consulte SearchIndexa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como consultar o índice do item

O exemplo de `search-index` a seguir consulta o índice `AWS_Things` em busca de itens que tenham um tipo de `LightBulb`.

```
aws iot search-index \
    --index-name "AWS_Things" \
    --query-string "thingTypeName:LightBulb"
```

Saída:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
```

```
            "thingId": "40da2e73-c6af-406e-b415-15acae538797",
            "thingTypeName": "LightBulb",
            "thingGroupNames": [
                "LightBulbs",
                "DeadBulbs"
            ],
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "ThirdBulb",
            "thingId": "615c8455-33d5-40e8-95fd-3ee8b24490af",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingId": "6dae0d3f-40c1-476a-80c4-1ed24ba6aa11",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        }
    ]
}
```

Para obter mais informações, consulte Gerenciar indexações de itens no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte SearchIndexem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte SearchIndexa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
```

```
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- Para obter detalhes da API, consulte a SearchIndexreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
```

```
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])
```

- Para obter detalhes da API, consulte a SearchIndexReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    " iv_query_string = 'thingName:MyThing*' - Fleet indexing query string
    TRY.
        oo_result = lo_iot->searchindex( iv_querystring = iv_query_string ).
        MESSAGE |Found { lines( oo_result->get_things( ) ) } IoT things matching
query.| TYPE 'I'.
      CATCH /aws1/cx_iotindexnotreadyex INTO DATA(lo_idx_ex).
```

```
        MESSAGE 'Fleet indexing is not ready. Enable indexing with
 UpdateIndexingConfiguration first.' TYPE 'I'.
        RAISE EXCEPTION lo_idx_ex.
      CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
        MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
        RAISE EXCEPTION lo_throttle_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SearchIndexreferência da API AWS SDK for SAP ABAP.

Use **`UpdateIndexingConfiguration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateIndexingConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update the indexing configuration.
/*!
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration
 object which is ignored if not set.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateIndexingConfiguration(
        const Aws::IoT::Model::ThingIndexingConfiguration
 &thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
 &thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {

 request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
 iotClient.UpdateIndexingConfiguration(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateIndexingConfigurationa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para habilitar a indexação de itens

O `update-indexing-configuration` exemplo a seguir permite que a indexação de coisas ofereça suporte à pesquisa de dados de registro, dados de sombra e status de conectividade de AWS coisas usando o índice _Things.

```
aws iot update-indexing-configuration
    --thing-indexing-
configuration thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

Este comando não produz saída.

Para obter mais informações, consulte Gerenciar indexações de itens no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte UpdateIndexingConfigurationem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
```

```
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateIndexingConfigurationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
```

```
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        DATA lo_idx_conf TYPE REF TO /aws1/cl_iotthingindexingconf.
        CREATE OBJECT lo_idx_conf
          EXPORTING
            iv_thingindexingmode = 'REGISTRY'.
        lo_iot->updateindexingconfiguration(
          io_thingindexingconf = lo_idx_conf ).
        MESSAGE 'IoT thing indexing configuration updated to REGISTRY mode.' TYPE
'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateIndexingConfigurationreferência da API AWS SDK for SAP ABAP.

Use **`UpdateThing`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateThing`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
 string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte UpdateThinga Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateThinga Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como associar um item a um tipo de item

O `update-thing` exemplo a seguir associa uma coisa no registro de AWS IoT a um tipo de coisa. Ao fazer a associação, devem ser fornecidos valores para os atributos definidos pelo tipo de item.

```
aws iot update-thing \
    --thing-name "MyOtherLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Esse comando não produz nenhuma saída. É possível usar o comando `describe-thing` para exibir o resultado.

Para obter mais informações, consulte Tipos de itens no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte UpdateThingem Referência de AWS CLI Comandos.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
```

```
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- Para obter detalhes da API, consulte a UpdateThingreferência da API AWS SDK for Kotlin.

Cenários para AWS IoT usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS IoT com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS IoT ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Conceitos básicos de mensagens da Internet das Coisas

Conceitos básicos da proteção de dispositivos da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM obrigatórios.
- Habiltiar verificações de auditoria do IoT Device Defender.
- Execute uma On-Demand auditoria
- Criar uma ação de mitigação.
- Aplicar ações de mitigação às descobertas.
- Configurar notificações do SNS (opcional).
- Habilitar o registro em log de IoT

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "====================================================="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
```

```
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /
dev/null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
```

```
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn'
 --output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
```

```
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
```

```
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
```

```
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
```

```
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "====================================================="
echo "Step 1: Creating required IAM roles"
echo "====================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
```

```
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "==================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "==================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1)
 || true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "==================================================="
```

```
echo "Step 3: Running an on-demand audit"
echo "====================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))
```

```
    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null ||
 echo "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
```

```
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "==================================================="
echo "Step 4: Creating a mitigation action"
echo "==================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting
 it first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction"
 2>&1 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
```

```
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1
 || true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "====================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "====================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
```

```
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1)
 || true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
```

```
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --
output json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi
```

```
    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS
 notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "====================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "====================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true
```

```
    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2
 methods, continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved
 but could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."
```

```
DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
```

```
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --
output json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null
 || echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
```

```
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies
  - ListTopics
  - PutRolePolicy
  - SetLoggingOptions
  - SetV2LoggingOptions
  - StartAuditMitigationActionsTask

- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Conceitos básicos de mensagens da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar recursos de IoT
- Configurar o dispositivo
- Executar os aplicativos de exemplo
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
}
```

```
# Function to check for errors
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 $LOG_FILE
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID
 --new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
}
```

```
# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "==================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a
 $LOG_FILE
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
    echo "AWS region not configured. Please run 'aws configure' to set your
 region." | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE

# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE
```

```
# Create IoT policy
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-
document file://iot-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?
```

```
# Create IoT thing
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn'
 --output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?

# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active
 --certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile
 $CERTS_DIR/public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
```

```
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
check_error $?

# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." |
 tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-
python-v2.git"
    check_error $?
else
```

```
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
fi

# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE

# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
```

```
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE

    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"

    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"
```

```
    # Delete policies
    echo "Deleting policies..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE" | tee -a $LOG_FILE
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachPolicy
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreatePolicy
  - CreateThing
  - DeleteCertificate
  - DeletePolicy
  - DeleteThing

- DescribeEndpoint
- DetachPolicy
- DetachThingPrincipal
- GetCallerIdentity
- UpdateCertificate

Exemplos de código para AWS IoT data usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS IoT data com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS IoT data Guia do desenvolvedor — Mais informações sobre AWS IoT data.
- AWS IoT data Referência da API — Detalhes sobre todas as AWS IoT data ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS IoT data usar AWS SDKs
  - Ações para AWS IoT data usar AWS SDKs
    - Use GetThingShadowcom um AWS SDK ou CLI
    - Use UpdateThingShadowcom um AWS SDK ou CLI

Exemplos básicos para AWS IoT data usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS IoT data com AWS SDKs.

Exemplos

- Ações para AWS IoT data usar AWS SDKs
  - Use GetThingShadowcom um AWS SDK ou CLI
  - Use UpdateThingShadowcom um AWS SDK ou CLI

Ações para AWS IoT data usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS IoT data ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS IoT data.

Exemplos

- Use GetThingShadowcom um AWS SDK ou CLI
- Use UpdateThingShadowcom um AWS SDK ou CLI

Use **`GetThingShadow`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetThingShadow`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</
returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
```

```
    {
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte GetThingShadowa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetThingShadowa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para obter um documento de sombra de item

O exemplo `get-thing-shadow` a seguir obtém o documento de sombra do item para o item IoT especificado.

```
aws iot-data get-thing-shadow \
```

```
    --thing-name MyRPi \
    output.txt
```

O comando não produz nenhuma saída na tela, mas o seguinte mostra o conteúdo de `output.txt`:

```
{
  "state":{
    "reported":{
    "moisture":"low"
    }
  },
  "metadata":{
    "reported":{
      "moisture":{
        "timestamp":1560269319
      }
    }
  },
  "version":1,"timestamp":1560269405
}
```

Para obter mais informações, consulte Fluxo de dados da sombra do dispositivo no Guia do desenvolvedor da AWS IoT.

- Para obter detalhes da API, consulte GetThingShadowem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
```

```
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a
Thing's shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte GetThingShadowa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}
```

- Para obter detalhes da API, consulte a GetThingShadowreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
            response =
 self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

- Para obter detalhes da API, consulte a GetThingShadowReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_iop->getthingshadow( iv_thingname = iv_thing_name ).

    " Convert xstring payload to JSON string
    DATA(lv_payload) = lo_result->get_payload( ).
    ov_shadow = /aws1/cl_rt_util=>xstring_to_string( lv_payload ).
    MESSAGE 'Thing shadow retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_iopresourcenotfoundex.
    MESSAGE 'Thing shadow not found.' TYPE 'E'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_exception).
    DATA(lv_error) = |{ lo_exception->get_text( ) }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetThingShadowreferência da API AWS SDK for SAP ABAP.

Use **`UpdateThingShadow`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateThingShadow`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
shadowPayload)
    {
        try
        {
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };

            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
{ex.Message}");
```

```
            return false;
        }
    }
```

- Para obter detalhes da API, consulte UpdateThingShadowa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
 iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
```

```
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateThingShadowa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para atualizar uma sombra de item

O exemplo `update-thing-shadow` a seguir modifica o estado atual da sombra do dispositivo para o item especificado e o salva no arquivo `output.txt`.

```
aws iot-data update-thing-shadow \
    --thing-name MyRPi \
    --payload "{"state":{"reported":{"moisture":"okay"}}}" \
    "output.txt"
```

O comando não produz nenhuma saída na tela, mas o seguinte mostra o conteúdo de `output.txt`:

```
{
    "state": {
        "reported": {
            "moisture": "okay"
        }
    },
    "metadata": {
        "reported": {
            "moisture": {
                "timestamp": 1560270036
```

```
            }
        }
    },
    "version": 2,
    "timestamp": 1560270036
}
```

Para obter mais informações, consulte Fluxo de dados da sombra do dispositivo no Guia do desenvolvedor da AWS IoT.

- Para obter detalhes da API, consulte UpdateThingShadowem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an
 IoT Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
 \"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument,
 StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
 UpdateThingShadowRequest.builder()
            .thingName(thingName)
```

```
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
 getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte UpdateThingShadowa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
```

```
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- Para obter detalhes da API, consulte a UpdateThingShadowreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
```

```
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateThingShadowReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Convert JSON string to xstring for payload
        DATA(lv_payload) = /aws1/
cl_rt_util=>string_to_xstring( iv_shadow_state ).

        lo_iop->updatethingshadow(
          iv_thingname = iv_thing_name
          iv_payload = lv_payload ).
        MESSAGE 'Thing shadow updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_iopresourcenotfoundex.
        MESSAGE 'Thing not found.' TYPE 'E'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_exception).
        DATA(lv_error) = |{ lo_exception->get_text( ) }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateThingShadowreferência da API AWS SDK for SAP ABAP.

Exemplos de código para AWS IoT SiteWise usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS IoT SiteWise com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS IoT SiteWise Guia do desenvolvedor — Mais informações sobre AWS IoT SiteWise.
- AWS IoT SiteWise Referência da API — Detalhes sobre todas as AWS IoT SiteWise ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS IoT SiteWise usar AWS SDKs
  - Hello AWS IoT SiteWise
  - Aprenda as noções básicas do AWS IoT SiteWise com um AWS SDK
  - Ações para AWS IoT SiteWise usar AWS SDKs
    - Use BatchPutAssetPropertyValuecom um AWS SDK ou CLI
    - Use CreateAssetcom um AWS SDK ou CLI
    - Use CreateAssetModelcom um AWS SDK ou CLI
    - Use CreateGatewaycom um AWS SDK ou CLI
    - Use DeleteAssetcom um AWS SDK ou CLI
    - Use DeleteAssetModelcom um AWS SDK ou CLI
    - Use DeleteGatewaycom um AWS SDK ou CLI
    - Use DescribeAssetModelcom um AWS SDK ou CLI
    - Use DescribeGatewaycom um AWS SDK ou CLI
    - Use GetAssetPropertyValuecom um AWS SDK ou CLI
    - Use ListAssetModelPropertiescom um AWS SDK
    - Use ListAssetModelscom um AWS SDK ou CLI

Exemplos básicos para AWS IoT SiteWise usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS IoT SiteWise com AWS SDKs.

Exemplos

- Hello AWS IoT SiteWise
- Aprenda as noções básicas do AWS IoT SiteWise com um AWS SDK
- Ações para AWS IoT SiteWise usar AWS SDKs
  - Use BatchPutAssetPropertyValuecom um AWS SDK ou CLI
  - Use CreateAssetcom um AWS SDK ou CLI
  - Use CreateAssetModelcom um AWS SDK ou CLI
  - Use CreateGatewaycom um AWS SDK ou CLI
  - Use DeleteAssetcom um AWS SDK ou CLI
  - Use DeleteAssetModelcom um AWS SDK ou CLI
  - Use DeleteGatewaycom um AWS SDK ou CLI
  - Use DescribeAssetModelcom um AWS SDK ou CLI
  - Use DescribeGatewaycom um AWS SDK ou CLI
  - Use GetAssetPropertyValuecom um AWS SDK ou CLI
  - Use ListAssetModelPropertiescom um AWS SDK
  - Use ListAssetModelscom um AWS SDK ou CLI

Hello AWS IoT SiteWise

O exemplo de código a seguir mostra como começar a usar o AWS IoT SiteWise.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class HelloSitewise {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
         fetchAssetModels();
    }

    /**
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
 IoTSiteWiseAsyncClient.create();
        ListAssetModelsRequest assetModelsRequest =
 ListAssetModelsRequest.builder()
            .assetModelTypes(AssetModelType.ASSET_MODEL)
            .build();

        // Asynchronous paginator - process paginated results.
        ListAssetModelsPublisher listModelsPaginator =
 siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
        CompletableFuture<Void> future = listModelsPaginator.subscribe(response -
> {
            response.assetModelSummaries().forEach(assetSummary ->
                logger.info("Asset Model Name: {} ", assetSummary.name())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- Para obter detalhes da API, consulte ListAssetModelsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  paginateListAssetModels,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new IoTSiteWiseClient();
  const listAssetModelsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListAssetModels({ client }, { maxResults: 5 });
    for await (const page of paginator) {
      listAssetModelsPaginated.push(...page.assetModelSummaries);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }
  for (const { name, creationDate } of listAssetModelsPaginated) {
    console.log(`${name} - ${creationDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte ListAssetModelsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_iot_sitewise(iot_sitewise_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT SiteWise
    client and list the asset models in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param iot_sitewise_client: A Boto3 AWS IoT SiteWise Client object. This
 object wraps
                                the low-level AWS IoT SiteWise service API.
    """
    print("Hello, AWS IoT SiteWise! Let's list some of your asset models:\n")
    paginator = iot_sitewise_client.get_paginator("list_asset_models")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

    asset_model_names: [str] = []
    for page in page_iterator:
        for asset_model in page["assetModelSummaries"]:
            asset_model_names.append(asset_model["name"])

    print(f"{len(asset_model_names)} asset model(s) retrieved.")
    for asset_model_name in asset_model_names:
        print(f"\t{asset_model_name}")
```

```
if __name__ == "__main__":
    hello_iot_sitewise(boto3.client("iotsitewise"))
```

- Para obter detalhes da API, consulte a ListAssetModelsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ios->listassetmodels( ). " oo_result is returned for
testing purposes. "
        DATA(lt_asset_models) = oo_result->get_assetmodelsummaries( ).
        MESSAGE 'Retrieved list of asset models.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list asset models.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListAssetModelsreferência da API AWS SDK for SAP ABAP.

Aprenda as noções básicas do AWS IoT SiteWise com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um modelo AWS IoT SiteWise de ativo.
- Crie um AWS IoT SiteWise ativo.
- Recuperar os valores de ID da propriedade.

- Envie dados para um AWS IoT SiteWise ativo.
- Recupere o valor da propriedade do AWS IoT SiteWise ativo.
- Crie um AWS IoT SiteWise portal.
- Crie um AWS IoT SiteWise gateway.
- Descreva o AWS IoT SiteWise Gateway.
- Exclua os AWS IoT SiteWise ativos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS IoT SiteWise recursos.

```
public class SitewiseScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static final String ROLES_STACK = "RoleSitewise";

    static SitewiseActions sitewiseActions = new SitewiseActions();

    public static void main(String[] args) throws Throwable {
        Scanner scanner = new Scanner(System.in);
        String assetModelName = "MyAssetModel1";
        String assetName = "MyAsset1" ;
        String gatewayName = "MyGateway1" ;
        String myThing =  "MyThing1" ;

        logger.info("""
```

```
            AWS IoT SiteWise is a fully managed software-as-a-service (SaaS)
that
            makes it easy to collect, store, organize, and monitor data from
industrial equipment and processes.
            It is designed to help industrial and manufacturing organizations
collect data from their equipment and
            processes, and use that data to make informed decisions about their
operations.

            One of the key features of AWS IoT SiteWise is its ability to connect
to a wide range of industrial
            equipment and systems, including programmable logic controllers
(PLCs), sensors, and other
            industrial devices. It can collect data from these devices and
organize it into a unified data model,
            making it easier to analyze and gain insights from the data. AWS IoT
SiteWise also provides tools for
            visualizing the data, setting up alarms and alerts, and generating
reports.

            Another key feature of AWS IoT SiteWise is its ability to scale to
handle large volumes of data.
            It can collect and store data from thousands of devices and process
millions of data points per second,
            making it suitable for large-scale industrial operations.
Additionally, AWS IoT SiteWise is designed
            to be secure and compliant, with features like role-based access
controls, data encryption,
            and integration with other AWS services for additional security and
compliance features.

            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(assetModelName, assetName, gatewayName, myThing);
        } catch (RuntimeException e) {
           logger.info(e.getMessage());
        }
    }
```

```
   public static void runScenario(String assetModelName, String assetName,
String gatewayName, String myThing) throws Throwable {
       logger.info("Use AWS CloudFormation to create an IAM role that is
required for this scenario.");
       CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
       Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
       String iamRole = stackOutputs.get("SitewiseRoleArn");
       logger.info("The ARN of the IAM role is {}",iamRole);
       logger.info(DASHES);

       logger.info(DASHES);
       logger.info("1. Create an AWS SiteWise Asset Model");
       logger.info("""
            An AWS IoT SiteWise Asset Model is a way to represent the physical
assets, such as equipment,
            processes, and systems, that exist in an industrial environment.
This model provides a structured and
            hierarchical representation of these assets, allowing users to
define the relationships and properties
            of each asset.

            This scenario creates two asset model properties: temperature and
humidity.
           """);
       waitForInputToContinue(scanner);
       String assetModelId = null;
       try {
           CreateAssetModelResponse response =
sitewiseActions.createAssetModelAsync(assetModelName).join();
           assetModelId = response.assetModelId();
           logger.info("Asset Model successfully created. Asset Model ID: {}. ",
assetModelId);
       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof ResourceAlreadyExistsException) {
               try {
                   assetModelId =
sitewiseActions.getAssetModelIdAsync(assetModelName).join();
                   logger.info("The Asset Model {} already exists. The id of the
existing model is {}. Moving on...", assetModelName, assetModelId);
               } catch (CompletionException cex) {
                   logger.error("Exception thrown acquiring the asset model id:
{}", cex.getCause().getCause(), cex);
```

```
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " +
cause.getMessage(), cause);
                return;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("2. Create an AWS IoT SiteWise Asset");
        logger.info("""
             The IoT SiteWise model that we just created defines the structure
and metadata for your physical assets.
             Now we create an asset from the asset model.

            """);
        logger.info("Let's wait 30 seconds for the asset to be ready.");
        countdown(30);
        waitForInputToContinue(scanner);
        String assetId;
        try {
            CreateAssetResponse response =
sitewiseActions.createAssetAsync(assetName, assetModelId).join();
            assetId = response.assetId();
            logger.info("Asset created with ID: {}", assetId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The asset model id was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Retrieve the property ID values");
        logger.info("""
```

```
            To send data to an asset, we need to get the property ID values. In
this scenario, we access the
            temperature and humidity property ID values.
           """);
       waitForInputToContinue(scanner);
       Map<String, String>  propertyIds = null;
       try {
           propertyIds = sitewiseActions.getPropertyIds(assetModelId).join();
       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof IoTSiteWiseException) {
               logger.error("IoTSiteWiseException occurred: {}",
cause.getMessage(), ce);
           } else {
               logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
           }
           return;
       }
       String humPropId =  propertyIds.get("Humidity");
       logger.info("The Humidity property Id is {}", humPropId);
       String tempPropId = propertyIds.get("Temperature");
       logger.info("The Temperature property Id is {}", tempPropId);

       waitForInputToContinue(scanner);
       logger.info(DASHES);

       logger.info(DASHES);
       logger.info("4. Send data to an AWS IoT SiteWise Asset");
       logger.info("""
           By sending data to an IoT SiteWise Asset, you can aggregate data
from
           multiple sources, normalize the data into a standard format, and
store it in a
           centralized location. This makes it easier to analyze and gain
insights from the data.

           In this example, we generate sample temperature and humidity data and
send it to the AWS IoT SiteWise asset.

           """);
       waitForInputToContinue(scanner);
       try {
```

```
            sitewiseActions.sendDataToSiteWiseAsync(assetId, tempPropId,
humPropId).join();
            logger.info("Data sent successfully.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The AWS resource was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Retrieve the value of the IoT SiteWise Asset property");
        logger.info("""
            IoT SiteWise is an AWS service that allows you to collect, process,
and analyze industrial data
            from connected equipment and sensors. One of the key benefits of
reading an IoT SiteWise property
            is the ability to gain valuable insights from your industrial data.

            """);
        waitForInputToContinue(scanner);
        try {
            Double assetVal = sitewiseActions.getAssetPropValueAsync(tempPropId,
assetId).join();
            logger.info("The property name is: {}", "Temperature");
            logger.info("The value of this property is: {}", assetVal);

            waitForInputToContinue(scanner);

            assetVal = sitewiseActions.getAssetPropValueAsync(humPropId,
assetId).join();
            logger.info("The property name is: {}", "Humidity");
            logger.info("The value of this property is: {}", assetVal);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
```

```
                    logger.info("The AWS resource was not found: {}",
cause.getMessage(), cause);
                } else {
                    logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
                }
                return;
            }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Create an IoT SiteWise Gateway");
        logger.info(
            """
                IoT SiteWise Gateway serves as the bridge between industrial
equipment, sensors, and the
                cloud-based IoT SiteWise service. It is responsible for securely
collecting, processing, and
                transmitting data from various industrial assets to the IoT
SiteWise platform,
                enabling real-time monitoring, analysis, and optimization of
industrial operations.

                """);
        waitForInputToContinue(scanner);
        String gatewayId = "";
        try {
            gatewayId = sitewiseActions.createGatewayAsync(gatewayName,
myThing).join();
            logger.info("Gateway creation completed successfully. id is {}",
gatewayId );
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException siteWiseEx) {
                logger.error("IoT SiteWise error occurred: Error message: {},
Error code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
```

```
        }
        logger.info(DASHES);
        logger.info(DASHES);

        logger.info("7. Describe the IoT SiteWise Gateway");
         waitForInputToContinue(scanner);
        try {
            sitewiseActions.describeGatewayAsync(gatewayId)
                .thenAccept(response -> {
                    logger.info("Gateway Name: {}", response.gatewayName());
                    logger.info("Gateway ARN: {}", response.gatewayArn());
                    logger.info("Gateway Platform: {}",
response.gatewayPlatform());
                    logger.info("Gateway Creation Date: {}",
response.creationDate());
                }).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the AWS IoT SiteWise Assets");
        logger.info(
            """
            Before you can delete the Asset Model, you must delete the assets.

            """);
        logger.info("Would you like to delete the IoT SiteWise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the SiteWise assets.");

            try {
```

```
                sitewiseActions.deleteGatewayAsync(gatewayId).join();
                logger.info("Gateway {} was deleted successfully.", gatewayId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException)
{
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }

            try {
                sitewiseActions.deleteAssetAsync(assetId).join();
                logger.info("Request to delete asset {} sent successfully",
assetId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException)
{
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            logger.info("Let's wait 1 minute for the asset to be deleted.");
            countdown(60);
            waitForInputToContinue(scanner);
            logger.info("Delete the AWS IoT SiteWise Asset Model");
            try {
                sitewiseActions.deleteAssetModelAsync(assetModelId).join();
                logger.info("Asset model deleted successfully.");
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException)
{
```

```
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            waitForInputToContinue(scanner);

        } else {
            logger.info("The resources will not be deleted.");
        }
        logger.info(DASHES);

        logger.info(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        logger.info("This concludes the AWS IoT SiteWise Scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int totalSeconds) throws InterruptedException {
        for (int i = totalSeconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
```

```
        }
        System.out.println(); // Move to the next line after countdown
        logger.info("Countdown complete!");
    }
}
```

Uma classe wrapper para métodos do AWS IoT SiteWise SDK.

```
public class SitewiseActions {

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseActions.class);

    private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

    private static IoTSiteWiseAsyncClient getAsyncClient() {
        if (ioTSiteWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ioTSiteWiseAsyncClient;
    }


    /**
     * Creates an asset model.
```

```
    *
    * @param name the name of the asset model to create.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse>
createAssetModelAsync(String name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
```

```
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }


    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String
assetName, String assetModelId) {
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
```

```
            });
    }

    /**
     * Sends data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
     * @param humidityPropId the ID of the humidity property.
     * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
     *         calling code can attach callbacks, then handle the result or
exception by calling
     *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
        Map<String, Double> sampleData = generateSampleData();
        long timestamp = Instant.now().toEpochMilli();

        TimeInNanos time = TimeInNanos.builder()
            .timeInSeconds(timestamp / 1000)
            .offsetInNanos((int) ((timestamp % 1000) * 1000000))
            .build();

        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
```

```
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Fetches the value of an asset property.
     *
     * @param propId  the ID of the asset property to fetch.
     * @param assetId the ID of the asset to fetch the property value for.
     * @return a {@link CompletableFuture} that represents a {@link Double}
result. The calling code can attach
     *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
```

```
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property
value: {}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }

   /**
    * Retrieves the property IDs associated with a specific asset model.
    *
    * @param assetModelId the ID of the asset model that defines the properties.
    * @return a {@link CompletableFuture} that represents a {@link Map} result
that associates the property name to the
    *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
    *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
```

```
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs:
{}.", throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }

    /**
     * Deletes an asset.
     *
     * @param assetId the ID of the asset to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String
assetId) {
        DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
            .assetId(assetId)
            .build();

        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
                }
            });
    }

    /**
     * Deletes an Asset Model with the specified ID.
     *
     * @param assetModelId the ID of the Asset Model to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<DeleteAssetModelResponse>
deleteAssetModelAsync(String assetModelId) {
        DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
            .assetModelId(assetModelId)
            .build();

        return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
                }
            });
    }

    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
```

```
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the asset model ID or null if the
    *         asset model cannot be found. The calling code can attach
callbacks, then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> getAssetModelIdAsync(String assetModelName)
{
       ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
       return getAsyncClient().listAssetModels(listAssetModelsRequest)
               .handle((listAssetModelsResponse, exception) -> {
                   if (exception != null) {
                       logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                       if (assetModelSummary.name().equals(assetModelName)) {
                           return assetModelSummary.id();
                       }
                   }
                   return null;
               });
   }


   /**
    * Creates a new IoT Sitewise gateway.
    *
    * @param gatewayName The name of the gateway to create.
    * @param myThing     The name of the core device thing to associate with the
gateway.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the gateways ID. The calling code
```

```
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> createGatewayAsync(String gatewayName,
String myThing) {
       GreengrassV2 gg = GreengrassV2.builder()
           .coreDeviceThingName(myThing)
           .build();

       GatewayPlatform platform = GatewayPlatform.builder()
           .greengrassV2(gg)
           .build();

       Map<String, String> tag = new HashMap<>();
       tag.put("Environment", "Production");

       CreateGatewayRequest createGatewayRequest =
CreateGatewayRequest.builder()
           .gatewayName(gatewayName)
           .gatewayPlatform(platform)
           .tags(tag)
           .build();

       return getAsyncClient().createGateway(createGatewayRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.error("Error creating the gateway.");
                   throw (CompletionException) exception;
               }
               logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
               return response.gatewayId();
           });
   }

   /**
```

```
    * Deletes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest =
DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
               }
           });
   }

   /**
    * Describes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to describe.
    * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
```

```
     *         it available to the calling code as a {@link CompletionException}.
 By calling
     *         {@link CompletionException#getCause()}, the calling code can
 access the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
 gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
 method: {}", exception.getCause().getMessage());
                }
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel

- DescribeGateway
- GetAssetPropertyValue
- ListAssetModelProperties
- ListAssetModels

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
  //} from "@aws-doc-sdk-examples/lib/scenario/index.js";
} from "../../libs/scenario/index.js";
import {
  IoTSiteWiseClient,
  CreateAssetModelCommand,
  CreateAssetCommand,
  ListAssetModelPropertiesCommand,
  BatchPutAssetPropertyValueCommand,
  GetAssetPropertyValueCommand,
  CreatePortalCommand,
  DescribePortalCommand,
  CreateGatewayCommand,
  DescribeGatewayCommand,
  DeletePortalCommand,
  DeleteGatewayCommand,
  DeleteAssetCommand,
  DeleteAssetModelCommand,
  DescribeAssetModelCommand,
} from "@aws-sdk/client-iotsitewise";
import {
```

```
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
  waitUntilStackDeleteComplete,
} from "@aws-sdk/client-cloudformation";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = "SiteWiseBasicsStack";

/**
 * @typedef {{
 *   iotSiteWiseClient: import('@aws-sdk/client-iotsitewise').IotSiteWiseClient,
 *   cloudFormationClient: import('@aws-sdk/client-
cloudformation').CloudFormationClient,
 *   stackName,
 *   stack,
 *   askToDeleteResources: true,
 *   asset: {assetName: "MyAsset1"},
 *   assetModel: {assetModelName: "MyAssetModel1"},
 *   portal: {portalName: "MyPortal1"},
 *   gateway: {gatewayName: "MyGateway1"},
 *   propertyIds: [],
 *   contactEmail: "user@mydomain.com",
 *   thing: "MyThing1",
 *   sampleData: { temperature: 23.5, humidity: 65.0}
 * }} State
 */

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});
```

```
const greet = new ScenarioOutput(
  "greet",
  `AWS IoT SiteWise is a fully managed industrial software-as-a-service (SaaS)
 that makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes. It is designed to help industrial and manufacturing
 organizations collect data from their equipment and processes, and use that data
 to make informed decisions about their operations.
One of the key features of AWS IoT SiteWise is its ability to connect to a
 wide range of industrial equipment and systems, including programmable logic
 controllers (PLCs), sensors, and other industrial devices. It can collect data
 from these devices and organize it into a unified data model, making it easier
 to analyze and gain insights from the data. AWS IoT SiteWise also provides tools
 for visualizing the data, setting up alarms and alerts, and generating reports.
Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data. It can collect and store data from thousands of devices and
 process millions of data points per second, making it suitable for large-scale
 industrial operations. Additionally, AWS IoT SiteWise is designed to be secure
 and compliant, with features like role-based access controls, data encryption,
 and integration with other AWS services for additional security and compliance
 features.

Let's get started...`,
  { header: true },
);

const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "This scenario uses AWS CloudFormation to create an IAM role that is required
 for this scenario. The stack will now be deployed.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../resources/cfn/iotsitewise_basics/SitewiseRoles-
template.yml`,
        "utf8",
      );
      await state.cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
```

```
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await state.cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );
      state.stack = stack.Stacks[0].Outputs[0];
      console.log(`The ARN of the IAM role is ${state.stack.OutputValue}`);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateAWSSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSSiteWiseAssetModel",
  `1. Create an AWS SiteWise Asset Model
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such
 as equipment, processes, and systems, that exist in an industrial environment.
 This model provides a structured and hierarchical representation of these
 assets, allowing users to define the relationships and properties of each asset.

This scenario creates two asset model properties: temperature and humidity.`,
);

const sdkCreateAWSSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    let assetModelResponse;
    try {
      assetModelResponse = await state.iotSiteWiseClient.send(
        new CreateAssetModelCommand({
```

```
          assetModelName: state.assetModel.assetModelName,
          assetModelProperties: [
            {
              name: "Temperature",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
            {
              name: "Humidity",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
          ],
        }),
      );
      state.assetModel.assetModelId = assetModelResponse.assetModelId;
      console.log(
        `Asset Model successfully created. Asset Model ID:
 ${state.assetModel.assetModelId}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceAlreadyExistsException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} already exists.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateAWSIoTSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSIoTSiteWiseAssetModel",
  `2. Create an AWS IoT SiteWise Asset
The IoT SiteWise model that we just created defines the structure and metadata
 for your physical assets. Now we create an asset from the asset model.

Let's wait 30 seconds for the asset to be ready.`,
```

```
);

const waitThirtySeconds = new ScenarioAction("waitThirtySeconds", async () => {
  await wait(30); // wait 30 seconds
  console.log("Time's up! Let's check the asset's status.");
});

const sdkCreateAWSIoTSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSIoTSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    try {
      const assetResponse = await state.iotSiteWiseClient.send(
        new CreateAssetCommand({
          assetModelId: state.assetModel.assetModelId,
          assetName: state.asset.assetName,
        }),
      );
      state.asset.assetId = assetResponse.assetId;
      console.log(`Asset created with ID: ${state.asset.assetId}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset ${state.assetModel.assetModelName} was not found.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrievePropertyId = new ScenarioOutput(
  "displayRetrievePropertyId",
  `3. Retrieve the property ID values

To send data to an asset, we need to get the property ID values. In this
 scenario, we access the temperature and humidity property ID values.`,
);

const sdkRetrievePropertyId = new ScenarioAction(
  "sdkRetrievePropertyId",
  async (state) => {
    try {
```

```
      const retrieveResponse = await state.iotSiteWiseClient.send(
        new ListAssetModelPropertiesCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
      for (const retrieveResponseKey in
 retrieveResponse.assetModelPropertySummaries) {
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Humidity"
        ) {
          state.propertyIds.Humidity =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Temperature"
        ) {
          state.propertyIds.Temperature =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
      }
      console.log(`The Humidity propertyId is ${state.propertyIds.Humidity}`);
      console.log(
        `The Temperature propertyId is ${state.propertyIds.Temperature}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem retrieving the properties: ${caught.message}`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displaySendDataToIoTSiteWiseAsset = new ScenarioOutput(
```

```
  "displaySendDataToIoTSiteWiseAsset",
  `4. Send data to an AWS IoT SiteWise Asset

By sending data to an IoT SiteWise Asset, you can aggregate data from multiple
 sources, normalize the data into a standard format, and store it in a
 centralized location. This makes it easier to analyze and gain insights from the
 data.

In this example, we generate sample temperature and humidity data and send it to
 the AWS IoT SiteWise asset.`,
);

const sdkSendDataToIoTSiteWiseAsset = new ScenarioAction(
  "sdkSendDataToIoTSiteWiseAsset",
  async (state) => {
    try {
      const sendResponse = await state.iotSiteWiseClient.send(
        new BatchPutAssetPropertyValueCommand({
          entries: [
            {
              entryId: "entry-3",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Humidity,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.humidity,
                  },
                  timestamp: {
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
            {
              entryId: "entry-4",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Temperature,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.temperature,
                  },
                  timestamp: {
```

```
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
          ],
        }),
      );
      console.log("The data was sent successfully.");
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrieveValueOfIoTSiteWiseAsset = new ScenarioOutput(
  "displayRetrieveValueOfIoTSiteWiseAsset",
  `5. Retrieve the value of the IoT SiteWise Asset property

IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data from connected equipment and sensors. One of the key benefits of
 reading an IoT SiteWise property is the ability to gain valuable insights from
 your industrial data.`,
);

const sdkRetrieveValueOfIoTSiteWiseAsset = new ScenarioAction(
  "sdkRetrieveValueOfIoTSiteWiseAsset",
  async (/** @type {State} */ state) => {
    try {
      const temperatureResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Temperature,
        }),
      );
      const humidityResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Humidity,
```

```
        }),
      );
      console.log(
        `The property value for Temperature is
 ${temperatureResponse.propertyValue.value.doubleValue}`,
      );
      console.log(
        `The property value for Humidity is
 ${humidityResponse.propertyValue.value.doubleValue}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateIoTSiteWisePortal = new ScenarioOutput(
  "displayCreateIoTSiteWisePortal",
  `6. Create an IoT SiteWise Portal

An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources, such as sensors, equipment, and control systems, into a centralized
 platform.`,
);

const sdkCreateIoTSiteWisePortal = new ScenarioAction(
  "sdkCreateIoTSiteWisePortal",
  async (/** @type {State} */ state) => {
    try {
      const createPortalResponse = await state.iotSiteWiseClient.send(
        new CreatePortalCommand({
          portalName: state.portal.portalName,
          portalContactEmail: state.contactEmail,
          roleArn: state.stack.OutputValue,
        }),
      );
      state.portal = { ...state.portal, ...createPortalResponse };
      await wait(5); // Allow the portal to properly propagate.
      console.log(
```

```
        `Portal created successfully. Portal ID
 ${createPortalResponse.portalId}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the Portal: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribePortal = new ScenarioOutput(
  "displayDescribePortal",
  `7. Describe the Portal

In this step, we get a description of the portal and display the portal URL.`,
);

const sdkDescribePortal = new ScenarioAction(
  "sdkDescribePortal",
  async (/** @type {State} */ state) => {
    try {
      const describePortalResponse = await state.iotSiteWiseClient.send(
        new DescribePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(`Portal URL: ${describePortalResponse.portalStartUrl}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);
```

```
const displayCreateIoTSiteWiseGateway = new ScenarioOutput(
  "displayCreateIoTSiteWiseGateway",
  `8. Create an IoT SiteWise Gateway

IoT SiteWise Gateway serves as the bridge between industrial equipment, sensors,
 and the cloud-based IoT SiteWise service. It is responsible for securely
 collecting, processing, and transmitting data from various industrial assets
 to the IoT SiteWise platform, enabling real-time monitoring, analysis, and
 optimization of industrial operations.`,
);

const sdkCreateIoTSiteWiseGateway = new ScenarioAction(
  "sdkCreateIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const createGatewayResponse = await state.iotSiteWiseClient.send(
        new CreateGatewayCommand({
          gatewayName: state.gateway.gatewayName,
          gatewayPlatform: {
            greengrassV2: {
              coreDeviceThingName: state.thing,
            },
          },
        }),
      );
      console.log(
        `Gateway creation completed successfully. ID is
 ${createGatewayResponse.gatewayId}`,
      );
      state.gateway.gatewayId = createGatewayResponse.gatewayId;
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the gateway: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribeIoTSiteWiseGateway = new ScenarioOutput(
```

```
  "displayDescribeIoTSiteWiseGateway",
  "9. Describe the IoT SiteWise Gateway",
);

const sdkDescribeIoTSiteWiseGateway = new ScenarioAction(
  "sdkDescribeIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const describeGatewayResponse = await state.iotSiteWiseClient.send(
        new DescribeGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
      console.log("Gateway creation completed successfully.");
      console.log(`Gateway Name: ${describeGatewayResponse.gatewayName}`);
      console.log(`Gateway ARN: ${describeGatewayResponse.gatewayArn}`);
      console.log(
        `Gateway Platform:
 ${Object.keys(describeGatewayResponse.gatewayPlatform)}`,
      );
      console.log(
        `Gateway Creation Date: ${describeGatewayResponse.creationDate}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
  "askToDeleteResources",
  `10. Delete the AWS IoT SiteWise Assets

Before you can delete the Asset Model, you must delete the assets.`,
  { type: "confirm" },
);

const displayConfirmDeleteResources = new ScenarioAction(
  "displayConfirmDeleteResources",
```

```
  async (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You selected to delete the SiteWise assets.";
    }
    return "The resources will not be deleted. Please delete them manually to
 avoid charges.";
  },
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    await wait(10); // Give the portal status time to catch up.
    try {
      await state.iotSiteWiseClient.send(
        new DeletePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(
        `Portal ${state.portal.portalName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
      } else {
        console.log(`When trying to delete the portal: ${caught.message}`);
      }
    }

    try {
      await state.iotSiteWiseClient.send(
        new DeleteGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
      console.log(
        `Gateway ${state.gateway.gatewayName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
      } else {
        console.log(`When trying to delete the gateway: ${caught.message}`);
```

```
      }
    }

    try {
      await state.iotSiteWiseClient.send(
        new DeleteAssetCommand({
          assetId: state.asset.assetId,
        }),
      );
      await wait(5); // Allow the delete to finish.
      console.log(`Asset ${state.asset.assetName} was deleted successfully.`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
      } else {
        console.log(`When deleting the asset: ${caught.message}`);
      }
    }

    await wait(30); // Allow asset deletion to finish.
    try {
      await state.iotSiteWiseClient.send(
        new DeleteAssetModelCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
      console.log(
        `Asset Model ${state.assetModel.assetModelName} was deleted
successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} was not found.`,
        );
      } else {
        console.log(`When deleting the asset model: ${caught.message}`);
      }
    }

    try {
      await state.cloudFormationClient.send(
        new DeleteStackCommand({
          StackName: stackName,
```

```
        }),
      );
      await waitUntilStackDeleteComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      console.log("The stack was deleted successfully.");
    } catch (caught) {
      console.log(
        `${caught.message}. The stack was NOT deleted. Please clean up the
 resources manually.`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the IoT Sitewise Basics scenario for the AWS Javascript SDK v3.
 Thank you!",
);

const myScenario = new Scenario(
  "IoTSiteWise Basics",
  [
    greet,
    pressEnter,
    displayBuildCloudFormationStack,
    sdkBuildCloudFormationStack,
    pressEnter,
    displayCreateAWSSiteWiseAssetModel,
    sdkCreateAWSSiteWiseAssetModel,
    displayCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    waitThirtySeconds,
    sdkCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    displayRetrievePropertyId,
    sdkRetrievePropertyId,
    pressEnter,
    displaySendDataToIoTSiteWiseAsset,
    sdkSendDataToIoTSiteWiseAsset,
    pressEnter,
```

```
    displayRetrieveValueOfIoTSiteWiseAsset,
    sdkRetrieveValueOfIoTSiteWiseAsset,
    pressEnter,
    displayCreateIoTSiteWisePortal,
    sdkCreateIoTSiteWisePortal,
    pressEnter,
    displayDescribePortal,
    sdkDescribePortal,
    pressEnter,
    displayCreateIoTSiteWiseGateway,
    sdkCreateIoTSiteWiseGateway,
    pressEnter,
    displayDescribeIoTSiteWiseGateway,
    sdkDescribeIoTSiteWiseGateway,
    pressEnter,
    askToDeleteResources,
    displayConfirmDeleteResources,
    sdkDeleteResources,
    goodbye,
  ],
  {
    iotSiteWiseClient: new IoTSiteWiseClient({}),
    cloudFormationClient: new CloudFormationClient({}),
    asset: { assetName: "MyAsset1" },
    assetModel: { assetModelName: "MyAssetModel1" },
    portal: { portalName: "MyPortal1" },
    gateway: { gatewayName: "MyGateway1" },
    propertyIds: [],
    contactEmail: "user@mydomain.com",
    thing: "MyThing1",
    sampleData: { temperature: 23.5, humidity: 65.0 },
  },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
```

```
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties
  - ListAssetModels

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class IoTSitewiseGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to manage IoT physical assets
 using
    the AWS IoT SiteWise.
    """

    def __init__(
        self,
        iot_sitewise_wrapper: IoTSitewiseWrapper,
        cloud_formation_resource: ServiceResource,
    ):
        self.iot_sitewise_wrapper = iot_sitewise_wrapper
        self.cloud_formation_resource = cloud_formation_resource
        self.stack = None
        self.asset_model_id = None
        self.asset_id = None
        self.portal_id = None
        self.gateway_id = None

    def run(self) -> None:
        """
        Runs the scenario.
        """
        print(
            """
AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes.
It is designed to help industrial and manufacturing organizations collect data
 from their equipment and
processes, and use that data to make informed decisions about their operations.

One of the key features of AWS IoT SiteWise is its ability to connect to a wide
 range of industrial
equipment and systems, including programmable logic controllers (PLCs), sensors,
 and other
industrial devices. It can collect data from these devices and organize it into a
 unified data model,
making it easier to analyze and gain insights from the data. AWS IoT SiteWise
 also provides tools for
visualizing the data, setting up alarms and alerts, and generating reports.
```

```
Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data.
It can collect and store data from thousands of devices and process millions of
 data points per second,
making it suitable for large-scale industrial operations. Additionally, AWS IoT
 SiteWise is designed
to be secure and compliant, with features like role-based access controls, data
 encryption,
and integration with other AWS services for additional security and compliance
 features.

Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(f"")
        print(
            f"Use AWS CloudFormation to create an IAM role that is required for
 this scenario."
        )
        template_file = IoTSitewiseGettingStarted.get_template_as_string()

        self.stack = self.deploy_cloudformation_stack(
            "python-iot-sitewise-basics", template_file
        )
        outputs = self.stack.outputs
        iam_role = None

        for output in outputs:
            if output.get("OutputKey") == "SitewiseRoleArn":
                iam_role = output.get("OutputValue")

        if iam_role is None:
            error_string = f"Failed to retrieve iam_role from CloudFormation
 stack."
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"The ARN of the IAM role is {iam_role}")
        print_dashes()
        print_dashes()
        print(f"1. Create an AWS SiteWise Asset Model")
        print(
```

```
            """
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such
 as equipment,
processes, and systems, that exist in an industrial environment. This model
 provides a structured and
hierarchical representation of these assets, allowing users to define the
 relationships and values
of each asset.

This scenario creates two asset model values: temperature and humidity.
        """
        )
        press_enter_to_continue()
        asset_model_name = "MyAssetModel1"
        temperature_property_name = "temperature"
        humidity_property_name = "humidity"
        try:
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
            ]
            self.asset_model_id = self.iot_sitewise_wrapper.create_asset_model(
                asset_model_name, properties
            )
            print(
                f"Asset Model successfully created. Asset Model ID:
 {self.asset_model_id}. "
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                self.asset_model_id =
 self.get_model_id_for_model_name(asset_model_name)
```

```
                print(
                    f"Asset Model {asset_model_name} already exists. Asset Model
 ID: {self.asset_model_id}. "
                )
            else:
                raise

        press_enter_to_continue()
        print_dashes()
        print(f"2. Create an AWS IoT SiteWise Asset")
        print(
            """
The IoT SiteWise model that we just created defines the structure and metadata
 for your physical assets.
Now we create an asset from the asset model.

        """
        )
        press_enter_to_continue()

        self.asset_id = self.iot_sitewise_wrapper.create_asset(
            "MyAsset1", self.asset_model_id
        )

        print(f"Asset created with ID: {self.asset_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()
        print(f"3. Retrieve the property ID values")
        print(
            """
To send data to an asset, we need to get the property ID values. In this
 scenario, we access the
temperature and humidity property ID values.
        """
        )
        press_enter_to_continue()
        property_ids = self.iot_sitewise_wrapper.list_asset_model_properties(
            self.asset_model_id
        )
        humidity_property_id = None
        temperature_property_id = None
        for property_id in property_ids:
            if property_id.get("name") == humidity_property_name:
```

```
                humidity_property_id = property_id.get("id")
            elif property_id.get("name") == temperature_property_name:
                temperature_property_id = property_id.get("id")
        if humidity_property_id is None or temperature_property_id is None:
            error_string = f"Failed to retrieve property IDs from Asset Model."
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"The Humidity property Id is {humidity_property_id}")
        print(f"The Temperature property Id is {temperature_property_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"4. Send data to an AWS IoT SiteWise Asset")
        print(
            """
By sending data to an IoT SiteWise Asset, you can aggregate data from
multiple sources, normalize the data into a standard format, and store it in a
centralized location. This makes it easier to analyze and gain insights from the
 data.

In this example, we generate sample temperature and humidity data and send it to
 the AWS IoT SiteWise asset.

        """
        )
        press_enter_to_continue()

        values = [
            {
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
        self.iot_sitewise_wrapper.batch_put_asset_property_value(self.asset_id,
 values)
        print(f"Data sent successfully.")
```

```
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"5. Retrieve the value of the IoT SiteWise Asset property")
        print(
            """
IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data
from connected equipment and sensors. One of the key benefits of reading an IoT
 SiteWise property
is the ability to gain valuable insights from your industrial data.

        """
        )
        press_enter_to_continue()

        property_value = self.iot_sitewise_wrapper.get_asset_property_value(
            self.asset_id, temperature_property_id
        )
        print(f"The property name is '{temperature_property_name}'.")

        print(
            f"The value of this property is: {property_value['value']
['doubleValue']}"
        )
        press_enter_to_continue()

        property_value = self.iot_sitewise_wrapper.get_asset_property_value(
            self.asset_id, humidity_property_id
        )
        print(f"The property name is '{humidity_property_name}'.")
        print(
            f"The value of this property is: {property_value['value']
['doubleValue']}"
        )
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"6. Create an IoT SiteWise Portal")
        print(
            """
```

```
An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources,
such as sensors, equipment, and control systems, into a centralized platform.
        """
        )

        press_enter_to_continue()
        contact_email = q.ask("Enter a contact email for the portal:",
 q.non_empty)
        print("Creating the portal. The portal may take a while to become
 active.")
        self.portal_id = self.iot_sitewise_wrapper.create_portal(
            "MyPortal1", iam_role, contact_email
        )
        print(f"Portal created successfully. Portal ID {self.portal_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"7. Describe the Portal")
        print(
            """
In this step, we get a description of the portal and display the portal URL.
        """
        )
        press_enter_to_continue()
        portal_description =
 self.iot_sitewise_wrapper.describe_portal(self.portal_id)
        print(f"Portal URL: {portal_description['portalStartUrl']}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"8. Create an IoT SiteWise Gateway")
        press_enter_to_continue()
        self.gateway_id = self.iot_sitewise_wrapper.create_gateway(
            "MyGateway1", "MyThing1"
        )
        print(f"Gateway creation completed successfully. id is
 {self.gateway_id}")
        print_dashes()
        print_dashes()
        print(f"9. Describe the IoT SiteWise Gateway")
        press_enter_to_continue()
```

```
        gateway_description = self.iot_sitewise_wrapper.describe_gateway(
            self.gateway_id
        )
        print(f"Gateway Name: {gateway_description['gatewayName']}")
        print(f"Gateway ARN: {gateway_description['gatewayArn']}")
        print(f"Gateway Platform:\n{gateway_description['gatewayPlatform']}")
        print(f"Gateway Creation Date: {gateway_description['gatewayArn']}")
        print_dashes()
        print_dashes()

        print(f"10. Delete the AWS IoT SiteWise Assets")
        if q.ask("Would you like to delete the IoT SiteWise Assets? (y/n)",
q.is_yesno):
            self.cleanup()
        else:
            print(f"The resources will not be deleted.")
        print_dashes()
        print_dashes()
        print(f"This concludes the AWS IoT SiteWise Scenario")

    def cleanup(self) -> None:
        """
        Deletes the CloudFormation stack and the resources created for the demo.
        """

        if self.gateway_id is not None:
            self.iot_sitewise_wrapper.delete_gateway(self.gateway_id)
            print(f"Deleted gateway with id {self.gateway_id}.")
            self.gateway_id = None
        if self.portal_id is not None:
            self.iot_sitewise_wrapper.delete_portal(self.portal_id)
            print(f"Deleted portal with id {self.portal_id}.")
            self.portal_id = None
        if self.asset_id is not None:
            self.iot_sitewise_wrapper.delete_asset(self.asset_id)
            print(f"Deleted asset with id {self.asset_id}.")
            self.iot_sitewise_wrapper.wait_asset_deleted(self.asset_id)
            self.asset_id = None
        if self.asset_model_id is not None:
            self.iot_sitewise_wrapper.delete_asset_model(self.asset_model_id)
            print(f"Deleted asset model with id {self.asset_model_id}.")
            self.asset_model_id = None
        if self.stack is not None:
```

```
            stack = self.stack
            self.stack = None
            self.destroy_cloudformation_stack(stack)

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `SitewiseRoles-template.yaml` AWS
CloudFormation script and are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example
resources.
        """
        print(
```

```
            f"CloudFormation stack '{stack.name}' is being deleted. This may take
 a few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        template_file_path = os.path.join(script_dir, "SitewiseRoles-
template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def get_model_id_for_model_name(self, model_name: str) -> str:
        """
        Returns the model ID for the given model name.

        :param model_name: The name of the model.
        :return: The model ID.
        """
        model_id = None
        asset_models = self.iot_sitewise_wrapper.list_asset_models()
        for asset_model in asset_models:
            if asset_model["name"] == model_name:
                model_id = asset_model["id"]
                break
        return model_id
```

IoTSitewiseWrapper classe que envolve AWS IoT SiteWise ações.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""
```

```
    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset_model(
        self, asset_model_name: str, properties: List[Dict[str, Any]]
    ) -> str:
        """
        Creates an AWS IoT SiteWise Asset Model.

        :param asset_model_name: The name of the asset model to create.
        :param properties: The property definitions of the asset model.
        :return: The ID of the created asset model.
        """
        try:
            response = self.iotsitewise_client.create_asset_model(
                assetModelName=asset_model_name,
                assetModelDescription="This is a sample asset model
description.",
                assetModelProperties=properties,
            )
            asset_model_id = response["assetModelId"]
            waiter = self.iotsitewise_client.get_waiter("asset_model_active")
```

```
            waiter.wait(assetModelId=asset_model_id)
            return asset_model_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Asset model %s already exists.", asset_model_name)
            else:
                logger.error(
                    "Error creating asset model %s. Here's why %s",
                    asset_model_name,
                    err.response["Error"]["Message"],
                )
            raise


    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
        :param asset_model_id: The ID of the asset model to associate with the
asset.
        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
            raise


    def list_asset_models(self) -> List[Dict[str, Any]]:
```

```
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
model.

        """
        try:
            asset_models = []
            paginator =
self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def list_asset_model_properties(self, asset_model_id: str) -> List[Dict[str,
Any]]:
        """
        Lists all AWS IoT SiteWise Asset Model Properties.

        :param asset_model_id: The ID of the asset model to list values for.
        :return: A list of dictionaries containing information about each asset
model property.
        """
        try:
            asset_model_properties = []
            paginator = self.iotsitewise_client.get_paginator(
                "list_asset_model_properties"
            )
            pages = paginator.paginate(assetModelId=asset_model_id)
            for page in pages:

asset_model_properties.extend(page["assetModelPropertySummaries"])
            return asset_model_properties
        except ClientError as err:
            logger.error(
```

```
                "Error listing asset model values. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def batch_put_asset_property_value(
        self, asset_id: str, values: List[Dict[str, str]]
    ) -> None:
        """
        Sends data to an AWS IoT SiteWise Asset.

        :param asset_id: The asset ID.
        :param values: A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        """
        try:
            entries = self.properties_to_values(asset_id, values)

 self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Asset %s does not exist.", asset_id)
            else:
                logger.error(
                    "Error sending data to asset. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise


    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
```

```
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
                "propertyId": value["propertyId"],
                "propertyValues": [
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
                ],
            }
            entries.append(entry)
        return entries


    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
        """
        Gets the value of an AWS IoT SiteWise Asset Property.
```

```
        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Asset %s or property %s does not exist.", asset_id,
property_id
                )
            else:
                logger.error(
                    "Error getting asset property value. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise


    def create_portal(
        self, portal_name: str, iam_role_arn: str, portal_contact_email: str
    ) -> str:
        """
        Creates an AWS IoT SiteWise Portal.

        :param portal_name: The name of the portal to create.
        :param iam_role_arn: The ARN of an IAM role.
        :param portal_contact_email: The contact email of the portal.
        :return: The ID of the created portal.
        """
        try:
            response = self.iotsitewise_client.create_portal(
                portalName=portal_name,
                roleArn=iam_role_arn,
                portalContactEmail=portal_contact_email,
            )
            portal_id = response["portalId"]
            waiter = self.iotsitewise_client.get_waiter("portal_active")
            waiter.wait(portalId=portal_id, WaiterConfig={"MaxAttempts": 40})
            return portal_id
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Portal %s already exists.", portal_name)
            else:
                logger.error(
                    "Error creating portal %s. Here's why %s",
                    portal_name,
                    err.response["Error"]["Message"],
                )
            raise


    def describe_portal(self, portal_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to describe.
        :return: A dictionary containing information about the portal.
        """
        try:
            response =
self.iotsitewise_client.describe_portal(portalId=portal_id)
            return response
        except ClientError as err:
            logger.error(
                "Error describing portal %s. Here's why %s",
                portal_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_gateway(self, gateway_name: str, my_thing: str) -> str:
        """
        Creates an AWS IoT SiteWise Gateway.

        :param gateway_name: The name of the gateway to create.
        :param my_thing: The core device thing name.
        :return: The ID of the created gateway.
        """
        try:
            response = self.iotsitewise_client.create_gateway(
                gatewayName=gateway_name,
                gatewayPlatform={
```

```
                    "greengrassV2": {"coreDeviceThingName": my_thing},
                },
                tags={"Environment": "Production"},
            )
            gateway_id = response["gatewayId"]
            return gateway_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Gateway %s already exists.", gateway_name)
            else:
                logger.error(
                    "Error creating gateway %s. Here's why %s",
                    gateway_name,
                    err.response["Error"]["Message"],
                )
            raise


    def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to describe.
        :return: A dictionary containing information about the gateway.
        """
        try:
            response =
self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error describing gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_gateway(self, gateway_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Gateway.
```

```
        :param gateway_id: The ID of the gateway to delete.
        """
        try:
            self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error deleting gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_portal(self, portal_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to delete.
        """
        try:
            self.iotsitewise_client.delete_portal(portalId=portal_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Portal %s does not exist.", portal_id)
            else:
                logger.error(
                    "Error deleting portal %s. Here's why %s",
                    portal_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_asset(self, asset_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset.

        :param asset_id: The ID of the asset to delete.
        """
        try:
```

```
            self.iotsitewise_client.delete_asset(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset %s. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.

        :param asset_model_id: The ID of the asset model to delete.
        """
        try:

self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise


    def wait_asset_deleted(self, asset_id: str) -> None:
        """
        Waits for an AWS IoT SiteWise Asset to be deleted.

        :param asset_id: The ID of the asset to wait for.
        """
        try:
            waiter = self.iotsitewise_client.get_waiter("asset_not_exists")
            waiter.wait(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error waiting for asset %s to be deleted. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties
  - ListAssetModels

Ações para AWS IoT SiteWise usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS IoT SiteWise ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS IoT SiteWise.

Exemplos

- Use BatchPutAssetPropertyValuecom um AWS SDK ou CLI
- Use CreateAssetcom um AWS SDK ou CLI
- Use CreateAssetModelcom um AWS SDK ou CLI
- Use CreateGatewaycom um AWS SDK ou CLI
- Use DeleteAssetcom um AWS SDK ou CLI

- Use DeleteAssetModelcom um AWS SDK ou CLI
- Use DeleteGatewaycom um AWS SDK ou CLI
- Use DescribeAssetModelcom um AWS SDK ou CLI
- Use DescribeGatewaycom um AWS SDK ou CLI
- Use GetAssetPropertyValuecom um AWS SDK ou CLI
- Use ListAssetModelPropertiescom um AWS SDK
- Use ListAssetModelscom um AWS SDK ou CLI

Use **`BatchPutAssetPropertyValue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `BatchPutAssetPropertyValue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para enviar dados às propriedades do ativo

O exemplo `batch-put-asset-property-value` a seguir envia dados de potência e temperatura para as propriedades do ativo identificadas pelos aliases da propriedade.

```
aws iotsitewise batch-put-asset-property-value \
    --cli-input-json file://batch-put-asset-property-value.json
```

Conteúdo de `batch-put-asset-property-value.json`:

```
{
    "entries": [
        {
            "entryId": "1575691200-company-windfarm-3-turbine-7-power",
            "propertyAlias": "company-windfarm-3-turbine-7-power",
            "propertyValues": [
```

```
                {
                    "value": {
                        "doubleValue": 4.92
                    },
                    "timestamp": {
                        "timeInSeconds": 1575691200
                    },
                    "quality": "GOOD"
                }
            ]
        },
        {
            "entryId": "1575691200-company-windfarm-3-turbine-7-temperature",
            "propertyAlias": "company-windfarm-3-turbine-7-temperature",
            "propertyValues": [
                {
                    "value": {
                        "integerValue": 38
                    },
                    "timestamp": {
                        "timeInSeconds": 1575691200
                    }
                }
            ]
        }
    ]
}
```

Saída:

```
{
    "errorEntries": []
}
```

Para obter mais informações, consulte Como ingerir dados usando a API de AWS SiteWise IoT no Guia do usuário de AWS IoT SiteWise .

- Para obter detalhes da API, consulte BatchPutAssetPropertyValueem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Sends data to the SiteWise service.
    *
    * @param assetId        the ID of the asset to which the data will be sent.
    * @param tempPropertyId the ID of the temperature property.
    * @param humidityPropId the ID of the humidity property.
    * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
    *         calling code can attach callbacks, then handle the result or
exception by calling
    *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();

       TimeInNanos time = TimeInNanos.builder()
           .timeInSeconds(timestamp / 1000)
           .offsetInNanos((int) ((timestamp % 1000) * 1000000))
           .build();
```

```
        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- Para obter detalhes da API, consulte BatchPutAssetPropertyValuea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  BatchPutAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Batch put asset property values.
 * @param {{ entries : array }}
 */
export const main = async ({ entries }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new BatchPutAssetPropertyValueCommand({
        entries: entries,
      }),
    );
    console.log("Asset properties batch put successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(`${caught.message}. A resource could not be found.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte BatchPutAssetPropertyValuea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
 AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)
```

```
    def batch_put_asset_property_value(
        self, asset_id: str, values: List[Dict[str, str]]
    ) -> None:
        """
        Sends data to an AWS IoT SiteWise Asset.

        :param asset_id: The asset ID.
        :param values: A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        """
        try:
            entries = self.properties_to_values(asset_id, values)

 self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Asset %s does not exist.", asset_id)
            else:
                logger.error(
                    "Error sending data to asset. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Uma função auxiliar para gerar o parâmetro de entradas com base em uma lista de valores.

```
    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
```

```
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
                "propertyId": value["propertyId"],
                "propertyValues": [
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
                ],
            }
            entries.append(entry)
        return entries
```

Aqui está um exemplo de uma lista de valores para passar para a função auxiliar.

```
        values = [
            {
```

```
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
```

- Para obter detalhes da API, consulte a BatchPutAssetPropertyValueReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ios->batchputassetpropertyvalue(
      it_entries = it_entries
    ).
    MESSAGE 'Data sent to IoT SiteWise asset successfully.' TYPE 'I'.
  CATCH /aws1/cx_iosresourcenotfoundex.
    MESSAGE 'Asset does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a BatchPutAssetPropertyValuereferência da API AWS SDK for SAP ABAP.

Use **`CreateAsset`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateAsset`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para criar um ativo

O exemplo `create-asset` a seguir cria um ativo de turbina eólica a partir de um modelo de ativo de turbina eólica.

```
aws iotsitewise create-asset \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
    --asset-name "Wind Turbine 1"
```

Saída:

```
{
    "assetId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "assetArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "assetStatus": {
        "state": "CREATING"
    }
}
```

Para obter mais informações, consulte Criação de ativos no Guia do SiteWise Usuário de AWS IoT.

- Para obter detalhes da API, consulte CreateAssetem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an asset with the specified name and asset model Id.
    *
    * @param assetName    the name of the asset to create.
    * @param assetModelId the Id of the asset model to associate with the asset.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<CreateAssetResponse> createAssetAsync(String
assetName, String assetModelId) {
       CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
           .assetModelId(assetModelId)
           .assetDescription("Created using the AWS SDK for Java")
           .assetName(assetName)
           .build();

       return getAsyncClient().createAsset(createAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
```

```
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateAsseta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  CreateAssetCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetName, assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateAssetCommand({
        assetName: assetName, // The name to give the Asset.
        assetModelId: assetModelId, // The ID of the asset model from which to
 create the asset.
      }),
    );
    console.log("Asset created successfully.");
    return result;
  } catch (caught) {
```

```
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset model could not be found. Please check the
 asset model id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateAsseta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
```

```
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
        :param asset_model_id: The ID of the asset model to associate with the
asset.
        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a CreateAssetReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ios->createasset(
      iv_assetname = iv_asset_name
      iv_assetmodelid = iv_asset_model_id
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'IoT SiteWise asset created' TYPE 'I'.
  CATCH /aws1/cx_iosresourcenotfoundex.
    MESSAGE 'Asset model does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAssetreferência da API AWS SDK for SAP ABAP.

Use **`CreateAssetModel`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateAssetModel`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para criar um modelo de ativo

O exemplo `create-asset-model` a seguir cria um modelo de ativo que define uma turbina eólica com as seguintes propriedades:

Número de série - O número de série de uma turbina eólica Energia gerada - O fluxo de dados de energia gerada de uma turbina eólica Temperatura C - O fluxo de dados de temperatura de uma turbina eólica em CelsiusTemperature F - Os pontos de dados de temperatura mapeados de Celsius a Fahrenheit

```
aws iotsitewise create-asset-model \
    --cli-input-json file://create-wind-turbine-model.json
```

Conteúdo de `create-wind-turbine-model.json`:

```
{
    "assetModelName": "Wind Turbine Model",
    "assetModelDescription": "Represents a wind turbine",
    "assetModelProperties": [
        {
            "name": "Serial Number",
            "dataType": "STRING",
            "type": {
                "attribute": {}
            }
        },
        {
            "name": "Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "measurement": {}
            }
        },
        {
            "name": "Temperature C",
            "dataType": "DOUBLE",
            "unit": "Celsius",
            "type": {
```

```
                "measurement": {}
            }
        },
        {
            "name": "Temperature F",
            "dataType": "DOUBLE",
            "unit": "Fahrenheit",
            "type": {
                "transform": {
                    "expression": "temp_c * 9 / 5 + 32",
                    "variables": [
                        {
                            "name": "temp_c",
                            "value": {
                                "propertyId": "Temperature C"
                            }
                        }
                    ]
                }
            }
        },
        {
            "name": "Total Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "metric": {
                    "expression": "sum(power)",
                    "variables": [
                        {
                            "name": "power",
                            "value": {
                                "propertyId": "Generated Power"
                            }
                        }
                    ],
                    "window": {
                        "tumbling": {
                            "interval": "1h"
                        }
                    }
                }
            }
        }
```

```
    ]
}
```

Saída:

```
{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelStatus": {
        "state": "CREATING"
    }
}
```

Para obter mais informações, consulte Definição de modelos de ativos no Guia do SiteWise usuário de AWS IoT.

- Para obter detalhes da API, consulte CreateAssetModelem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an asset model.
     *
     * @param name the name of the asset model to create.
     * @return a {@link CompletableFuture} that represents a {@link
 CreateAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by
 calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
```

```
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse>
createAssetModelAsync(String name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
           .build();

       return getAsyncClient().createAssetModel(createAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
               }
```

```
            });
    }
```

- Para obter detalhes da API, consulte CreateAssetModela Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  CreateAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset Model.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetModelName, assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateAssetModelCommand({
        assetModelName: assetModelName, // The name to give the Asset Model.
      }),
    );
    console.log("Asset model created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the asset model.`,
```

```
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateAssetModela Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.
```

```
        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset_model(
        self, asset_model_name: str, properties: List[Dict[str, Any]]
    ) -> str:
        """
        Creates an AWS IoT SiteWise Asset Model.

        :param asset_model_name: The name of the asset model to create.
        :param properties: The property definitions of the asset model.
        :return: The ID of the created asset model.
        """
        try:
            response = self.iotsitewise_client.create_asset_model(
                assetModelName=asset_model_name,
                assetModelDescription="This is a sample asset model
description.",
                assetModelProperties=properties,
            )
            asset_model_id = response["assetModelId"]
            waiter = self.iotsitewise_client.get_waiter("asset_model_active")
            waiter.wait(assetModelId=asset_model_id)
            return asset_model_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Asset model %s already exists.", asset_model_name)
            else:
                logger.error(
                    "Error creating asset model %s. Here's why %s",
                    asset_model_name,
                    err.response["Error"]["Message"],
                )
            raise
```

Aqui está um exemplo de uma lista de propriedades para passar para a função auxiliar.

```
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
            ]
```

- Para obter detalhes da API, consulte a CreateAssetModelReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ios->createassetmodel(
          iv_assetmodelname = iv_asset_model_name
          iv_assetmodeldescription = 'This is a sample asset model description.'
          it_assetmodelproperties = it_properties
        ). " oo_result is returned for testing purposes. "
        MESSAGE 'IoT SiteWise asset model created' TYPE 'I'.
      CATCH /aws1/cx_iosresrcalrdyexistsex.
        MESSAGE 'Asset model already exists.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAssetModelreferência da API AWS SDK for SAP ABAP.

Use **`CreateGateway`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para criar um gateway

O `create-gateway` exemplo a seguir cria um gateway que é executado no AWS IoT Greengrass.

```
aws iotsitewise create-gateway \
    --gateway-name ExampleCorpGateway \
    --gateway-platform greengrass={groupArn=arn:aws:greengrass:us-
west-2:123456789012:/greengrass/groups/a1b2c3d4-5678-90ab-cdef-1b1b1EXAMPLE}
```

Saída:

```
{
    "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
}
```

Para obter mais informações, consulte Configurando um gateway no Guia do usuário de AWS SiteWise IoT.

- Para obter detalhes da API, consulte CreateGatewayem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new IoT Sitewise gateway.
    *
    * @param gatewayName The name of the gateway to create.
    * @param myThing     The name of the core device thing to associate with the
gateway.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the gateways ID. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> createGatewayAsync(String gatewayName,
String myThing) {
       GreengrassV2 gg = GreengrassV2.builder()
           .coreDeviceThingName(myThing)
           .build();

       GatewayPlatform platform = GatewayPlatform.builder()
           .greengrassV2(gg)
           .build();

       Map<String, String> tag = new HashMap<>();
       tag.put("Environment", "Production");
```

```
        CreateGatewayRequest createGatewayRequest =
 CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
 response.gatewayArn());
                return response.gatewayId();
            });
    }
```

- Para obter detalhes da API, consulte CreateGatewaya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  CreateGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
```

```
 * Create a Gateway.
 * @param {{  }}
 */
export const main = async ({ gatewayName }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateGatewayCommand({
        gatewayName: gatewayName, // The name to give the created Gateway.
      }),
    );
    console.log("Gateway created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the Gateway.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateGatewaya Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""
```

```
    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_gateway(self, gateway_name: str, my_thing: str) -> str:
        """
        Creates an AWS IoT SiteWise Gateway.

        :param gateway_name: The name of the gateway to create.
        :param my_thing: The core device thing name.
        :return: The ID of the created gateway.
        """
        try:
            response = self.iotsitewise_client.create_gateway(
                gatewayName=gateway_name,
                gatewayPlatform={
                    "greengrassV2": {"coreDeviceThingName": my_thing},
                },
                tags={"Environment": "Production"},
            )
            gateway_id = response["gatewayId"]
            return gateway_id
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Gateway %s already exists.", gateway_name)
            else:
                logger.error(
                    "Error creating gateway %s. Here's why %s",
                    gateway_name,
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a CreateGatewayReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ios->creategateway(
          iv_gatewayname = iv_gateway_name
          io_gatewayplatform = NEW /aws1/cl_iosgatewayplatform(
            io_greengrassv2 = NEW /aws1/cl_iosgreengrassv2(
              iv_coredevicethingname = iv_core_device_thing_name
            )
          )
          it_tags = VALUE /aws1/cl_iostagmap_w=>tt_tagmap(
            (
              VALUE /aws1/cl_iostagmap_w=>ts_tagmap_maprow(
                key = 'Environment'
                value = NEW /aws1/cl_iostagmap_w( 'Production' )
              )
            )
          )
```

```
        ). " oo_result is returned for testing purposes. "
        MESSAGE 'IoT SiteWise gateway created' TYPE 'I'.
      CATCH /aws1/cx_iosresrcalrdyexistsex.
        MESSAGE 'Gateway already exists.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateGatewayreferência da API AWS SDK for SAP ABAP.

Use **`DeleteAsset`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteAsset`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para excluir um ativo

O exemplo `delete-asset` a seguir exclui um modelo de ativo de turbina eólica.

```
aws iotsitewise delete-asset \
    --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

Saída:

```
{
    "assetStatus": {
        "state": "DELETING"
    }
}
```

Para obter mais informações, consulte Excluindo ativos no Guia do Usuário de AWS SiteWise IoT.

- Para obter detalhes da API, consulte DeleteAssetem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes an asset.
    *
    * @param assetId the ID of the asset to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String
assetId) {
       DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
           .assetId(assetId)
           .build();

       return getAsyncClient().deleteAsset(deleteAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred deleting asset with id: {}",
assetId);
               }
           });
```

```
    }
```

- Para obter detalhes da API, consulte DeleteAsseta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DeleteAssetCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset.
 * @param {{ assetId : string }}
 */
export const main = async ({ assetId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteAssetCommand({
        assetId: assetId, // The model id to delete.
      }),
    );
    console.log("Asset deleted successfully.");
    return { assetDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset.`,
      );
```

```
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteAsseta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.
```

```
        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_asset(self, asset_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset.

        :param asset_id: The ID of the asset to delete.
        """
        try:
            self.iotsitewise_client.delete_asset(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset %s. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteAssetReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ios->deleteasset(
          iv_assetid = iv_asset_id
```

```
        ).
        MESSAGE 'IoT SiteWise asset deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to delete asset.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAssetreferência da API AWS SDK for SAP ABAP.

Use **`DeleteAssetModel`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteAssetModel`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para excluir um modelo de ativo

O exemplo `delete-asset-model` a seguir exclui um modelo de ativo de turbina eólica.

```
aws iotsitewise delete-asset-model \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

Saída:

```
{
    "assetModelStatus": {
        "state": "DELETING"
    }
}
```

Para obter mais informações, consulte Excluindo modelos de ativos no Guia do usuário de AWS SiteWise IoT.

- Para obter detalhes da API, consulte DeleteAssetModelem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes an Asset Model with the specified ID.
    *
    * @param assetModelId the ID of the Asset Model to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse>
deleteAssetModelAsync(String assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
```

```
                    logger.error("Failed to delete asset model with ID:{}.",
 exception.getMessage());
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteAssetModela Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DeleteAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteAssetModelCommand({
        assetModelId: assetModelId, // The model id to delete.
      }),
    );
    console.log("Asset model deleted successfully.");
    return { assetModelDeleted: true };
  } catch (caught) {
```

```
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset model.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteAssetModela Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
```

```
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.

        :param asset_model_id: The ID of the asset model to delete.
        """
        try:

self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteAssetModelReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ios->deleteassetmodel(
          iv_assetmodelid = iv_asset_model_id
        ).
        MESSAGE 'IoT SiteWise asset model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to delete asset model.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAssetModelreferência da API AWS SDK for SAP ABAP.

Use **`DeleteGateway`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para excluir um gateway

O exemplo `delete-gateway` a seguir exclui um gateway.

```
aws iotsitewise delete-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

Este comando não produz saída.

Para obter mais informações, consulte Como ingerir dados usando um gateway no Guia do usuário de AWS SiteWise IoT.

- Para obter detalhes da API, consulte DeleteGatewayem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest =
DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
               }
           });
   }
```

- Para obter detalhes da API, consulte DeleteGatewaya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DeleteGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Gateway deleted successfully.");
    return { gatewayDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the
 Gateway Id.`,
      );
```

```
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteGatewaya Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.
```

```
        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_gateway(self, gateway_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to delete.
        """
        try:
            self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error deleting gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteGatewayReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ios->deletegateway(
          iv_gatewayid = iv_gateway_id
        ).
        MESSAGE 'IoT SiteWise gateway deleted.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Gateway does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteGatewayreferência da API AWS SDK for SAP ABAP.

Use **`DescribeAssetModel`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAssetModel`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para descrever um modelo de ativo

O exemplo `describe-asset-model` a seguir descreve um modelo de ativo de parque eólico.

```
aws iotsitewise describe-asset-model \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

Saída:

```
{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetModelName": "Wind Farm Model",
```

```
    "assetModelDescription": "Represents a wind farm that comprises many wind
 turbines",
    "assetModelProperties": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",
            "name": "Total Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "metric": {
                    "expression": "sum(power)",
                    "variables": [
                        {
                            "name": "power",
                            "value": {
                                "propertyId": "a1b2c3d4-5678-90ab-
cdef-66666EXAMPLE",
                                "hierarchyId": "a1b2c3d4-5678-90ab-
cdef-77777EXAMPLE"
                            }
                        }
                    ],
                    "window": {
                        "tumbling": {
                            "interval": "1h"
                        }
                    }
                }
            }
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-88888EXAMPLE",
            "name": "Region",
            "dataType": "STRING",
            "type": {
                "attribute": {
                    "defaultValue": " "
                }
            }
        }
    ],
    "assetModelHierarchies": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",
```

```
            "name": "Wind Turbines",
            "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
        }
    ],
    "assetModelCreationDate": 1575671284.0,
    "assetModelLastUpdateDate": 1575671988.0,
    "assetModelStatus": {
        "state": "ACTIVE"
    }
}
```

Para obter mais informações, consulte Descrição de um modelo de ativo específico no Guia do SiteWise usuário de AWS IoT.

- Para obter detalhes da API, consulte DescribeAssetModelem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result
that associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
```

```
     *         {@link CompletionException#getCause()}, the calling code can
 access the original exception.
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
 assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
 ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
 AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs:
 {}.", throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
```

- Para obter detalhes da API, consulte DescribeAssetModela Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DescribeAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";
```

```
/**
 * Describe an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { assetModelDescription } = await client.send(
      new DescribeAssetModelCommand({
        assetModelId: assetModelId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset model information retrieved successfully.");
    return { assetModelDescription: assetModelDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset model could not be found. Please check the
 asset model id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeAssetModela Referência AWS SDK para JavaScript da API.

Use **`DescribeGateway`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeGateway`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para descrever um gateway

O exemplo `describe-gateway` a seguir descreve um gateway.

```
aws iotsitewise describe-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

Saída:

```
{
    "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayName": "ExampleCorpGateway",
    "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayPlatform": {
        "greengrass": {
            "groupArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/a1b2c3d4-5678-90ab-cdef-1b1b1EXAMPLE"
        }
    },
    "gatewayCapabilitySummaries": [
        {
            "capabilityNamespace": "iotsitewise:opcuacollector:1",
            "capabilitySyncStatus": "IN_SYNC"
        }
    ],
    "creationDate": 1588369971.457,
    "lastUpdateDate": 1588369971.457
}
```

Para obter mais informações, consulte Como ingerir dados usando um gateway no Guia do usuário de AWS SiteWise IoT.

- Para obter detalhes da API, consulte DescribeGatewayem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Describes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to describe.
    * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
       DescribeGatewayRequest request = DescribeGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().describeGateway(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
               }
           });
   }
```

- Para obter detalhes da API, consulte DescribeGatewaya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DescribeGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { gatewayDescription } = await client.send(
      new DescribeGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Gateway information retrieved successfully.");
    return { gatewayDescription: gatewayDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the
 Gateway Id.`,
      );
    } else {
      throw caught;
```

```
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeGatewaya Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                              access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
 AWS IoT SiteWise client.
```

```
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to describe.
        :return: A dictionary containing information about the gateway.
        """
        try:
            response =
self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error describing gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DescribeGatewayReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ios->describegateway(
          iv_gatewayid = iv_gateway_id
        ). " oo_result is returned for testing purposes. "
        MESSAGE 'Retrieved gateway description.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Gateway does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeGatewayreferência da API AWS SDK for SAP ABAP.

Use **`GetAssetPropertyValue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetAssetPropertyValue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para recuperar o valor atual de uma propriedade do ativo

O exemplo `get-asset-property-value` a seguir recupera a potência total atual de um ativo de turbina eólica.

```
aws iotsitewise get-asset-property-value \
    --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \
    --property-id a1b2c3d4-5678-90ab-cdef-66666EXAMPLE
```

Saída:

```
{
    "propertyValue": {
        "value": {
            "doubleValue": 6890.8677520453875
```

```
        },
        "timestamp": {
            "timeInSeconds": 1580853000,
            "offsetInNanos": 0
        },
        "quality": "GOOD"
    }
}
```

Para obter mais informações, consulte Consultando os valores atuais das propriedades do ativo no Guia do Usuário de AWS SiteWise IoT.

- Para obter detalhes da API, consulte GetAssetPropertyValueem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Fetches the value of an asset property.
     *
     * @param propId  the ID of the asset property to fetch.
     * @param assetId the ID of the asset to fetch the property value for.
     * @return a {@link CompletableFuture} that represents a {@link Double}
result. The calling code can attach
     *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
```

```
     */
    public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
 assetId) {
        GetAssetPropertyValueRequest assetPropertyValueRequest =
 GetAssetPropertyValueRequest.builder()
                .propertyId(propId)
                .assetId(assetId)
                .build();

        return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
                .handle((response, exception) -> {
                    if (exception != null) {
                        logger.error("Error occurred while fetching property
 value: {}.", exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    return response.propertyValue().value().doubleValue();
                });
    }
```

- Para obter detalhes da API, consulte GetAssetPropertyValuea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
```

```
 * Describe an asset property value.
 * @param {{ entryId : string }}
 */
export const main = async ({ entryId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new GetAssetPropertyValueCommand({
        entryId: entryId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset property information retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset property entry could not be found. Please
 check the entry id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte GetAssetPropertyValuea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""
```

```
    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
        """
        Gets the value of an AWS IoT SiteWise Asset Property.

        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
```

```
                    "Asset %s or property %s does not exist.", asset_id,
property_id
                )
            else:
                logger.error(
                    "Error getting asset property value. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a GetAssetPropertyValueReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ios->getassetpropertyvalue(
          iv_assetid = iv_asset_id
          iv_propertyid = iv_property_id
        ). " oo_result is returned for testing purposes. "
        MESSAGE 'Retrieved asset property value.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Asset or property does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetAssetPropertyValuereferência da API AWS SDK for SAP ABAP.

Use **`ListAssetModelProperties`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListAssetModelProperties`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ios->listassetmodelproperties(
      iv_assetmodelid = iv_asset_model_id
    ). " oo_result is returned for testing purposes. "
    DATA(lt_properties) = oo_result->get_assetmodelpropertysums( ).
    MESSAGE 'Retrieved list of asset model properties.' TYPE 'I'.
  CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to list asset model properties.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListAssetModelPropertiesreferência da API AWS SDK for SAP ABAP.

Use **`ListAssetModels`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListAssetModels`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para listar todos os modelos de ativos

O `list-asset-models` exemplo a seguir lista todos os modelos de ativos definidos em sua AWS conta na região atual.

```
aws iotsitewise list-asset-models
```

Saída:

```
{
    "assetModelSummaries": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
            "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
            "name": "Wind Farm Model",
            "description": "Represents a wind farm that comprises many wind
 turbines",
            "creationDate": 1575671284.0,
            "lastUpdateDate": 1575671988.0,
            "status": {
                "state": "ACTIVE"
            }
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "name": "Wind Turbine Model",
            "description": "Represents a wind turbine manufactured by Example
 Corp",
            "creationDate": 1575671207.0,
            "lastUpdateDate": 1575686273.0,
            "status": {
                "state": "ACTIVE"
            }
```

```
        }
    ]
}
```

Para obter mais informações, consulte Listar todos os modelos de ativos no Guia do SiteWise usuário de AWS IoT.

- Para obter detalhes da API, consulte ListAssetModelsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the asset model ID for the given asset model name.
    *
    * @param assetModelName the name of the asset model for the ID.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the asset model ID or null if the
    *         asset model cannot be found. The calling code can attach
callbacks, then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> getAssetModelIdAsync(String assetModelName)
{
       ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
```

```
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
 exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
 listAssetModelsResponse.assetModelSummaries()) {
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
                    return null;
                });
    }
```

- Para obter detalhes da API, consulte ListAssetModelsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  ListAssetModelsCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * List asset models.
 * @param {{ assetModelTypes : array }}
 */
```

```
export const main = async ({ assetModelTypes = [] }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new ListAssetModelsCommand({
        assetModelTypes: assetModelTypes, // The model types to list
      }),
    );
    console.log("Asset model types retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem listing the asset model types.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte ListAssetModelsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.
```

```
        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def list_asset_models(self) -> List[Dict[str, Any]]:
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
model.

        """
        try:
            asset_models = []
            paginator =
self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListAssetModelsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_ios->listassetmodels( ). " oo_result is returned for
testing purposes. "
       DATA(lt_asset_models) = oo_result->get_assetmodelsummaries( ).
       MESSAGE 'Retrieved list of asset models.' TYPE 'I'.
     CATCH /aws1/cx_rt_generic.
       MESSAGE 'Unable to list asset models.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListAssetModelsreferência da API AWS SDK for SAP ABAP.

Exemplos de código para Amazon Keyspaces usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Keyspaces (para Apache Cassandra) com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Amazon Keyspaces : mais informações sobre o Amazon Keyspaces.
- Referência da API Amazon Keyspaces: detalhes sobre todas as ações disponíveis do Amazon Keyspaces.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Amazon Keyspaces usando AWS SDKs
  - Olá, Amazon Keyspaces
  - Aprenda as noções básicas do Amazon Keyspaces com um AWS SDK
  - Ações para o Amazon Keyspaces usando AWS SDKs
    - Use CreateKeyspacecom um AWS SDK
    - Use CreateTablecom um AWS SDK
    - Use DeleteKeyspacecom um AWS SDK
    - Use DeleteTablecom um AWS SDK
    - Use GetKeyspacecom um AWS SDK
    - Use GetTablecom um AWS SDK
    - Use ListKeyspacescom um AWS SDK
    - Use ListTablescom um AWS SDK
    - Use RestoreTablecom um AWS SDK
    - Use UpdateTablecom um AWS SDK

Exemplos básicos de Amazon Keyspaces usando AWS SDKs

Os exemplos de código a seguir mostram como usar os elementos básicos do Amazon Keyspaces (para Apache Cassandra) com SDKs da AWS .

Exemplos

- Olá, Amazon Keyspaces
- Aprenda as noções básicas do Amazon Keyspaces com um AWS SDK
- Ações para o Amazon Keyspaces usando AWS SDKs
  - Use CreateKeyspacecom um AWS SDK
  - Use CreateTablecom um AWS SDK
  - Use DeleteKeyspacecom um AWS SDK
  - Use DeleteTablecom um AWS SDK
  - Use GetKeyspacecom um AWS SDK
  - Use GetTablecom um AWS SDK
  - Use ListKeyspacescom um AWS SDK
  - Use ListTablescom um AWS SDK
  - Use RestoreTablecom um AWS SDK
  - Use UpdateTablecom um AWS SDK

Olá, Amazon Keyspaces

Os exemplos de código a seguir mostram como começar a usar o Amazon Keyspaces.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace KeyspacesActions;
```

```
public class HelloKeyspaces
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Keyspaces (for Apache
 Cassandra).
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonKeyspaces>()
                .AddTransient<KeyspacesWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloKeyspaces>();

        var keyspacesClient =
 host.Services.GetRequiredService<IAmazonKeyspaces>();
        var keyspacesWrapper = new KeyspacesWrapper(keyspacesClient);

        Console.WriteLine("Hello, Amazon Keyspaces! Let's list your keyspaces:");
        await keyspacesWrapper.ListKeyspaces();
    }
}
```

- Para obter detalhes da API, consulte ListKeyspacesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
 ListKeyspacesRequest.builder()
                    .maxResults(10)
```

```
                    .build();

            ListKeyspacesResponse response =
 keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace : keyspaces) {
                System.out.println("The name of the keyspace is " +
 keyspace.keyspaceName());
            }

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

- Para obter detalhes da API, consulte ListKeyspacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/
```

```
suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListKeyspacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_keyspaces(keyspaces_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Keyspaces (for Apache
 Cassandra)
    client and list the keyspaces in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
```

```
    :param keyspaces_client: A Boto3 Amazon Keyspaces Client object. This object
 wraps
                             the low-level Amazon Keyspaces service API.
    """
    print("Hello, Amazon Keyspaces! Let's list some of your keyspaces:\n")
    for ks in keyspaces_client.list_keyspaces(maxResults=5).get("keyspaces", []):
        print(ks["keyspaceName"])
        print(f"\t{ks['resourceArn']}")


if __name__ == "__main__":
    hello_keyspaces(boto3.client("keyspaces"))
```

- Para obter detalhes da API, consulte a ListKeyspacesReferência da API AWS SDK for Python (Boto3).

Aprenda as noções básicas do Amazon Keyspaces com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um keyspace e uma tabela. O esquema da tabela contém dados do filme e tem a recuperação para um ponto no tempo.
- Conectar-se ao keyspace usando uma conexão TLS segura com autenticação SigV4.
- Consultar a tabela. Adicionar, recuperar e atualizar dados do filme.
- Atualizar a tabela. Adicionar uma coluna para rastrear os filmes assistidos.
- Restaure a tabela ao estado anterior e limpe os recursos.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
global using System.Security.Cryptography.X509Certificates;
```

```
global using Amazon.Keyspaces;
global using Amazon.Keyspaces.Model;
global using KeyspacesActions;
global using KeyspacesScenario;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using Newtonsoft.Json;


namespace KeyspacesBasics;

/// <summary>
/// Amazon Keyspaces (for Apache Cassandra) scenario. Shows some of the basic
/// actions performed with Amazon Keyspaces.
/// </summary>
public class KeyspacesBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonKeyspaces>()
            .AddTransient<KeyspacesWrapper>()
            .AddTransient<CassandraWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<KeyspacesBasics>();

        var configuration = new ConfigurationBuilder()
```

```
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var keyspacesWrapper =
host.Services.GetRequiredService<KeyspacesWrapper>();
        var uiMethods = new UiMethods();

        var keyspaceName = configuration["KeyspaceName"];
        var tableName = configuration["TableName"];

        bool success; // Used to track the results of some operations.

        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        // Create the keyspace.
        var keyspaceArn = await keyspacesWrapper.CreateKeyspace(keyspaceName);

        // Wait for the keyspace to be available. GetKeyspace results in a
        // resource not found error until it is ready for use.
        try
        {
            var getKeyspaceArn = "";
            Console.Write($"Created {keyspaceName}. Waiting for it to become
available. ");
            do
            {
                getKeyspaceArn = await
keyspacesWrapper.GetKeyspace(keyspaceName);
                Console.Write(". ");
            } while (getKeyspaceArn != keyspaceArn);
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine("Waiting for keyspace to be created.");
        }

        Console.WriteLine($"\nThe keyspace {keyspaceName} is ready for use.");

        uiMethods.PressEnter();
```

```
        // Create the table.
        // First define the schema.
        var allColumns = new List<ColumnDefinition>
        {
            new ColumnDefinition { Name = "title", Type = "text" },
            new ColumnDefinition { Name = "year", Type = "int" },
            new ColumnDefinition { Name = "release_date", Type = "timestamp" },
            new ColumnDefinition { Name = "plot", Type = "text" },
        };

        var partitionKeys = new List<PartitionKey>
        {
            new PartitionKey { Name = "year", },
            new PartitionKey { Name = "title" },
        };

        var tableSchema = new SchemaDefinition
        {
            AllColumns = allColumns,
            PartitionKeys = partitionKeys,
        };

        var tableArn = await keyspacesWrapper.CreateTable(keyspaceName,
tableSchema, tableName);

        // Wait for the table to be active.
        try
        {
            var resp = new GetTableResponse();
            Console.Write("Waiting for the new table to be active. ");
            do
            {
                try
                {
                    resp = await keyspacesWrapper.GetTable(keyspaceName,
tableName);
                    Console.Write(".");
                }
                catch (ResourceNotFoundException)
                {
                    Console.Write(".");
                }
            } while (resp.Status != TableStatus.ACTIVE);
```

```
            // Display the table's schema.
            Console.WriteLine($"\nTable {tableName} has been created in
{keyspaceName}");
            Console.WriteLine("Let's take a look at the schema.");
            uiMethods.DisplayTitle("All columns");
            resp.SchemaDefinition.AllColumns.ForEach(column =>
            {
                Console.WriteLine($"{column.Name,-40}\t{column.Type,-20}");
            });

            uiMethods.DisplayTitle("Cluster keys");
            resp.SchemaDefinition.ClusteringKeys.ForEach(clusterKey =>
            {

Console.WriteLine($"{clusterKey.Name,-40}\t{clusterKey.OrderBy,-20}");
            });

            uiMethods.DisplayTitle("Partition keys");
            resp.SchemaDefinition.PartitionKeys.ForEach(partitionKey =>
            {
                Console.WriteLine($"{partitionKey.Name}");
            });

            uiMethods.PressEnter();
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        // Access Apache Cassandra using the Cassandra drive for C#.
        var cassandraWrapper =
host.Services.GetRequiredService<CassandraWrapper>();
        var movieFilePath = configuration["MovieFile"];

        Console.WriteLine("Let's add some movies to the table we created.");
        var inserted = await cassandraWrapper.InsertIntoMovieTable(keyspaceName,
tableName, movieFilePath);

        uiMethods.PressEnter();

        Console.WriteLine("Added the following movies to the table:");
        var rows = await cassandraWrapper.GetMovies(keyspaceName, tableName);
        uiMethods.DisplayTitle("All Movies");
```

```
        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            var plot = row.GetValue<string>("plot");
            var release_date = row.GetValue<DateTime>("release_date");
            Console.WriteLine($"{release_date}\t{title}\t{year}\n{plot}");
            Console.WriteLine(uiMethods.SepBar);
        }

        // Update the table schema
        uiMethods.DisplayTitle("Update table schema");
        Console.WriteLine("Now we will update the table to add a boolean field
 called watched.");

        // First save the current time as a UTC Date so the original
        // table can be restored later.
        var timeChanged = DateTime.UtcNow;

        // Now update the schema.
        var resourceArn = await keyspacesWrapper.UpdateTable(keyspaceName,
 tableName);
        uiMethods.PressEnter();

        Console.WriteLine("Now let's mark some of the movies as watched.");

        // Pick some files to mark as watched.
        var movieToWatch = rows[2].GetValue<string>("title");
        var watchedMovieYear = rows[2].GetValue<int>("year");
        var changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
 tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[6].GetValue<string>("title");
        watchedMovieYear = rows[6].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
 tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[9].GetValue<string>("title");
        watchedMovieYear = rows[9].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
 tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[10].GetValue<string>("title");
```

```
        watchedMovieYear = rows[10].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[13].GetValue<string>("title");
        watchedMovieYear = rows[13].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        uiMethods.DisplayTitle("Watched movies");
        Console.WriteLine("These movies have been marked as watched:");
        rows = await cassandraWrapper.GetWatchedMovies(keyspaceName, tableName);
        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            Console.WriteLine($"{title,-40}\t{year,8}");
        }
        uiMethods.PressEnter();

        Console.WriteLine("We can restore the table to its previous state but
that can take up to 20 minutes to complete.");
        string answer;
        do
        {
            Console.WriteLine("Do you want to restore the table? (y/n)");
            answer = Console.ReadLine();
        } while (answer.ToLower() != "y" && answer.ToLower() != "n");

        if (answer == "y")
        {
            var restoredTableName = $"{tableName}_restored";
            var restoredTableArn = await keyspacesWrapper.RestoreTable(
                keyspaceName,
                tableName,
                restoredTableName,
                timeChanged);
            // Loop and call GetTable until the table is gone. Once it has been
            // deleted completely, GetTable will raise a
ResourceNotFoundException.
            bool wasRestored = false;

            try
            {
```

```
                do
                {
                    var resp = await keyspacesWrapper.GetTable(keyspaceName,
restoredTableName);
                    wasRestored = (resp.Status == TableStatus.ACTIVE);
                } while (!wasRestored);
            }
            catch (ResourceNotFoundException)
            {
                // If the restored table raised an error, it isn't
                // ready yet.
                Console.Write(".");
            }
        }

        uiMethods.DisplayTitle("Clean up resources.");

        // Delete the table.
        success = await keyspacesWrapper.DeleteTable(keyspaceName, tableName);

        Console.WriteLine($"Table {tableName} successfully deleted from
{keyspaceName}.");
        Console.WriteLine("Waiting for the table to be removed completely. ");

        // Loop and call GetTable until the table is gone. Once it has been
        // deleted completely, GetTable will raise a ResourceNotFoundException.
        bool wasDeleted = false;

        try
        {
            do
            {
                var resp = await keyspacesWrapper.GetTable(keyspaceName,
tableName);
            } while (!wasDeleted);
        }
        catch (ResourceNotFoundException ex)
        {
            wasDeleted = true;
            Console.WriteLine($"{ex.Message} indicates that the table has been
deleted.");
        }

        // Delete the keyspace.
```

```
        success = await keyspacesWrapper.DeleteKeyspace(keyspaceName);
        Console.WriteLine("The keyspace has been deleted and the demo is now
 complete.");
    }
}
```

```
namespace KeyspacesActions;

/// <summary>
/// Performs Amazon Keyspaces (for Apache Cassandra) actions.
/// </summary>
public class KeyspacesWrapper
{
    private readonly IAmazonKeyspaces _amazonKeyspaces;

    /// <summary>
    /// Constructor for the KeyspaceWrapper.
    /// </summary>
    /// <param name="amazonKeyspaces">An Amazon Keyspaces client object.</param>
    public KeyspacesWrapper(IAmazonKeyspaces amazonKeyspaces)
    {
        _amazonKeyspaces = amazonKeyspaces;
    }

    /// <summary>
    /// Create a new keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name for the new keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
    public async Task<string> CreateKeyspace(string keyspaceName)
    {
        var response =
            await _amazonKeyspaces.CreateKeyspaceAsync(
                new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }


    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
```

```
    /// <param name="keyspaceName">The keyspace where the table will be
created.</param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
PointInTimeRecoveryStatus.ENABLED }
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }


    /// <summary>
    /// Delete an existing keyspace.
    /// </summary>
    /// <param name="keyspaceName"></param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
            new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
```

```
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get data about a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
    public async Task<string> GetKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.GetKeyspaceAsync(
            new GetKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }


    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }


    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());
```

```
        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

 Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }



    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new
 ListTablesRequest { KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

 Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }



    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
```

```
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }


    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
        var newColumn = new ColumnDefinition { Name = "watched", Type =
 "boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }

}
```

```
using System.Net;
using Cassandra;

namespace KeyspacesScenario;

/// <summary>
/// Class to perform CRUD methods on an Amazon Keyspaces (for Apache Cassandra)
 database.
///
```

```
/// NOTE: This sample uses a plain text authenticator for example purposes only.
/// Recommended best practice is to use a SigV4 authentication plugin, if
 available.
/// </summary>
public class CassandraWrapper
{
    private readonly IConfiguration _configuration;
    private readonly string _localPathToFile;
    private const string _certLocation = "https://certs.secureserver.net/
repository/sf-class2-root.crt";
    private const string _certFileName = "sf-class2-root.crt";
    private readonly X509Certificate2Collection _certCollection;
    private X509Certificate2 _amazoncert;
    private Cluster _cluster;

    // User name and password for the service.
    private string _userName = null!;
    private string _pwd = null!;

    public CassandraWrapper()
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        _localPathToFile = Path.GetTempPath();

        // Get the Starfield digital certificate and save it locally.
        var client = new WebClient();
        client.DownloadFile(_certLocation, $"{_localPathToFile}/
{_certFileName}");

        //var httpClient = new HttpClient();
        //var httpResult = httpClient.Get(fileUrl);
        //using var resultStream = await httpResult.Content.ReadAsStreamAsync();
        //using var fileStream = File.Create(pathToSave);
        //resultStream.CopyTo(fileStream);

        _certCollection = new X509Certificate2Collection();
        _amazoncert = new X509Certificate2($"{_localPathToFile}/
{_certFileName}");
```

```
        // Get the user name and password stored in the configuration file.
        _userName = _configuration["UserName"]!;
        _pwd = _configuration["Password"]!;

        // For a list of Service Endpoints for Amazon Keyspaces, see:
        // https://docs.aws.amazon.com/keyspaces/latest/devguide/
programmatic.endpoints.html
        var awsEndpoint = _configuration["ServiceEndpoint"];

        _cluster = Cluster.Builder()
            .AddContactPoints(awsEndpoint)
            .WithPort(9142)
            .WithAuthProvider(new PlainTextAuthProvider(_userName, _pwd))
            .WithSSL(new SSLOptions().SetCertificateCollection(_certCollection))
            .WithQueryOptions(
                new QueryOptions()
                    .SetConsistencyLevel(ConsistencyLevel.LocalQuorum)
                    .SetSerialConsistencyLevel(ConsistencyLevel.LocalSerial))
            .Build();
    }

    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the Apache Cassandra table.
    /// </summary>
    /// <param name="movieFileName">The full path to the JSON file.</param>
    /// <returns>A list of movie objects.</returns>
    public List<Movie> ImportMoviesFromJson(string movieFileName, int numToImport
 = 0)
    {
        if (!File.Exists(movieFileName))
        {
            return null!;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();

        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        // If numToImport = 0, return all movies in the collection.
        if (numToImport == 0)
        {
```

```
            // Now return the entire list of movies.
            return allMovies;
        }
        else
        {
            // Now return the first numToImport entries.
            return allMovies.GetRange(0, numToImport);
        }
    }

    /// <summary>
    /// Insert movies into the movie table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="movieTableName">The Amazon Keyspaces table.</param>
    /// <param name="movieFilePath">The path to the resource file containing
    /// movie data to insert into the table.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> InsertIntoMovieTable(string keyspaceName, string
 movieTableName, string movieFilePath, int numToImport = 20)
    {
        // Get some movie data from the movies.json file
        var movies = ImportMoviesFromJson(movieFilePath, numToImport);

        var session = _cluster.Connect(keyspaceName);

        string insertCql;

        RowSet rs;

        // Now we insert the numToImport movies into the table.
        foreach (var movie in movies)
        {
            // Escape single quote characters in the plot.
            insertCql = $"INSERT INTO {keyspaceName}.{movieTableName}
 (title, year, release_date, plot) values($${movie.Title}$$, {movie.Year},
 '{movie.Info.Release_Date.ToString("yyyy-MM-dd")}', $${movie.Info.Plot}$$)";
            rs = await session.ExecuteAsync(new SimpleStatement(insertCql));
        }

        return true;
    }

    /// <summary>
```

```
    /// Gets all of the movies in the movies table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing movie data.</returns>
    public async Task<List<Row>> GetMovies(string keyspaceName, string tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT * FROM
{keyspaceName}.{tableName}"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }

    /// <summary>
    /// Mark a movie in the movie table as watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <param name="title">The title of the movie to mark as watched.</param>
    /// <param name="year">The year the movie was released.</param>
    /// <returns>A set of rows containing the changed data.</returns>
    public async Task<List<Row>> MarkMovieAsWatched(string keyspaceName, string
tableName, string title, int year)
    {
        var session = _cluster.Connect();
        string updateCql = $"UPDATE {keyspaceName}.{tableName} SET watched=true
WHERE title = $${title}$$ AND year = {year};";
        var rs = await session.ExecuteAsync(new SimpleStatement(updateCql));
        var rows = rs.GetRows().ToList();
        return rows;
    }
```

```
    /// <summary>
    /// Retrieve the movies in the movies table where watched is true.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing information about movies
    /// where watched is true.</returns>
    public async Task<List<Row>> GetWatchedMovies(string keyspaceName, string
 tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT
 title, year, plot FROM {keyspaceName}.{tableName} WHERE watched = true ALLOW
 FILTERING"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable

- ListKeyspaces
- ListTables
- RestoreTable
- UpdateTable

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
```

```
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
 * 8. Get all records from the Movie table.
 * 9. Get a specific Movie.
 * 10. Get a UTC timestamp for the current time.
 * 11. Update the table schema to add a ‘watched’ Boolean column.
 * 12. Update an item as watched.
 * 13. Query for items with watched = True.
 * 14. Restore the table back to the previous state using the timestamp.
 * 15. Check for completion of the restore action.
 * 16. Delete the table.
 * 17. Confirm that both tables are deleted.
 * 18. Delete the keyspace.
 */

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this
 file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException,
 IOException {
        String fileName = "<Replace with the JSON file that contains movie
 data>";
        String keyspaceName = "<Replace with the name of the keyspace to
 create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
        String tableNameRestore = "MovieRestore";
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        DriverConfigLoader loader =
 DriverConfigLoader.fromClasspath("application.conf");
```

```
        CqlSession session = CqlSession.builder()
                .withConfigLoader(loader)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Keyspaces example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a keyspace.");
        createKeySpace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        Thread.sleep(5000);
        System.out.println("2. Check for keyspace existence.");
        checkKeyspaceExistence(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
 enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into
 the Movie table.");
```

```
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
 column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state
 using the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
 minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

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

```
   public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
           throws InterruptedException {
       try {
           String status;
           GetTableResponse response;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           // Keep looping until table cannot be found and a
ResourceNotFoundException is
           // thrown.
           while (true) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println(". The table status is " + status);
               Thread.sleep(500);
           }

       } catch (ResourceNotFoundException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
       System.out.println("The table is deleted");
   }

   public static void deleteTable(KeyspacesClient keyClient, String
keyspaceName, String tableName) {
       try {
           DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           keyClient.deleteTable(tableRequest);

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

```
   public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           while (!tableStatus) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println("The table status is " + status);

               if (status.compareTo("ACTIVE") == 0) {
                   tableStatus = true;
               }
               Thread.sleep(500);
           }

           List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
           for (ColumnDefinition def : cols) {
               System.out.println("The column name is " + def.name());
               System.out.println("The column type is " + def.type());
           }

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

   public static void restoreTable(KeyspacesClient keyClient, String
keyspaceName, ZonedDateTime utc) {
       try {
           Instant myTime = utc.toInstant();
           RestoreTableRequest restoreTableRequest =
RestoreTableRequest.builder()
                   .restoreTimestamp(myTime)
                   .sourceTableName("Movie")
```

```
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

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

    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session
                .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
                + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year
= :k1;";
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
                .setString("k0", titleUpdate)
                .setInt("k1", yearUpdate)
                .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }
```

```
   public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           ColumnDefinition def = ColumnDefinition.builder()
                   .name("watched")
                   .type("boolean")
                   .build();

           UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                   .keyspaceName(keySpace)
                   .tableName(tableName)
                   .addColumns(def)
                   .build();

           keyClient.updateTable(tableRequest);

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

   public static void getSpecificMovie(CqlSession session, String keyspaceName)
{
       ResultSet resultSet = session.execute(
               "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title =
'The Family' ALLOW FILTERING ;");
       resultSet.forEach(item -> {
           System.out.println("The Movie title is " + item.getString("title"));
           System.out.println("The Movie year is " + item.getInt("year"));
           System.out.println("The plot is " + item.getString("plot"));
       });
   }

   // Get records from the Movie table.
   public static void getMovieData(CqlSession session, String keyspaceName) {
       ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
       resultSet.forEach(item -> {
           System.out.println("The Movie title is " + item.getString("title"));
           System.out.println("The Movie year is " + item.getInt("year"));
           System.out.println("The plot is " + item.getString("plot"));
       });
```

```
    }

    // Load data into the table.
    public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
        String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {

            // Add 20 movies to the table.
            if (t == 20)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String plot = currentNode.path("info").path("plot").toString();

            // Insert the data into the Amazon Keyspaces table.
            BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
            builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
            PreparedStatement preparedStatement = session.prepare(sqlStatement);
            builder.addStatement(preparedStatement.boundStatementBuilder()
                    .setString("k0", title)
                    .setInt("k1", year)
                    .setString("k2", plot)
                    .build());

            BatchStatement batchStatement = builder.build();
            session.execute(batchStatement);
            t++;
        }

        System.out.println("You have added " + t + " records successfully!");
    }
```

```
   public static void listTables(KeyspacesClient keyClient, String keyspaceName)
{
       try {
           ListTablesRequest tablesRequest = ListTablesRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
           listRes.stream()
                   .flatMap(r -> r.tables().stream())
                   .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                           " Table name: " + content.tableName()));

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

   public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           while (!tableStatus) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println(". The table status is " + status);

               if (status.compareTo("ACTIVE") == 0) {
                   tableStatus = true;
               }
               Thread.sleep(500);
           }
```

```
            List<ColumnDefinition> cols =
 response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

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

    public static void createTable(KeyspacesClient keyClient, String keySpace,
 String tableName) {
        try {
            // Set the columns.
            ColumnDefinition defTitle = ColumnDefinition.builder()
                    .name("title")
                    .type("text")
                    .build();

            ColumnDefinition defYear = ColumnDefinition.builder()
                    .name("year")
                    .type("int")
                    .build();

            ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                    .name("release_date")
                    .type("timestamp")
                    .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                    .name("plot")
                    .type("text")
                    .build();

            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
```

```
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

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

    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();
```

```
            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

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

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            GetKeyspaceResponse response =
keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

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

    public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest =
CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
response.resourceArn());
```

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
```

```
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This example uses a secure file format to hold certificate information for
 Kotlin applications. This is required to make a connection to Amazon Keyspaces.
 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html

 This Kotlin example performs the following tasks:

 1. Create a keyspace.
 2. Check for keyspace existence.
 3. List keyspaces using a paginator.
 4. Create a table with a simple movie data schema and enable point-in-time
 recovery.
 5. Check for the table to be in an Active state.
 6. List all tables in the keyspace.
 7. Use a Cassandra driver to insert some records into the Movie table.
 8. Get all records from the Movie table.
 9. Get a specific Movie.
 10. Get a UTC timestamp for the current time.
 11. Update the table schema to add a ‘watched’ Boolean column.
 12. Update an item as watched.
 13. Query for items with watched = True.
 14. Restore the table back to the previous state using the timestamp.
 15. Check for completion of the restore action.
 16. Delete the table.
 17. Confirm that both tables are deleted.
 18. Delete the keyspace.
 */

/*
   Usage:
     fileName - The name of the JSON file that contains movie data. (Get this
 file from the GitHub repo at resources/sample_file.)
     keyspaceName - The name of the keyspace to create.
 */
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
```

```
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
    println(DASHES)

    println(DASHES)
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("3. List keyspaces using a paginator.")
    listKeyspacesPaginator()
    println(DASHES)

    println(DASHES)
    println("4. Create a table with a simple movie data schema and enable point-
in-time recovery.")
    createTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("5. Check for the table to be in an Active state.")
    delay(6000)
    checkTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
```

```
    println("6. List all tables in the keyspace.")
    listTables(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("7. Use a Cassandra driver to insert some records into the Movie
 table.")
    delay(6000)
    loadData(session, fileName, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("8. Get all records from the Movie table.")
    getMovieData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("9. Get a specific Movie.")
    getSpecificMovie(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("10. Get a UTC timestamp for the current time.")
    val utc = ZonedDateTime.now(ZoneOffset.UTC)
    println("DATETIME = ${Date.from(utc.toInstant())}")
    println(DASHES)

    println(DASHES)
    println("11. Update the table schema to add a watched Boolean column.")
    updateTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("12. Update an item as watched.")
    delay(10000) // Wait 10 seconds for the update.
    updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
    println(DASHES)

    println(DASHES)
    println("13. Query for items with watched = True.")
    getWatchedData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
```

```
    println("14. Restore the table back to the previous state using the
 timestamp.")
    println("Note that the restore operation can take up to 20 minutes.")
    restoreTable(keyspaceName, utc)
    println(DASHES)

    println(DASHES)
    println("15. Check for completion of the restore action.")
    delay(5000)
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("17. Confirm that both tables are deleted.")
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

```
suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient
 ->
            // Keep looping until the table cannot be found and a
 ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
```

```
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }

        val cols = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
```

```
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".
\"MovieKotlin\" WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0
 AND year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
```

```
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\" WHERE
 title = 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
```

```
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".
\"MovieKotlin\";")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
 (:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
```

```
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )

        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
```

```
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? =
 response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }
```

```
    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
```

```
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse =
 keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class KeyspaceScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Keyspaces."""

    def __init__(self, ks_wrapper):
        """
        :param ks_wrapper: An object that wraps Amazon Keyspace actions.
        """
        self.ks_wrapper = ks_wrapper
```

```
    @demo_func
    def create_keyspace(self):
        """
        1. Creates a keyspace.
        2. Lists up to 10 keyspaces in your account.
        """
        print("Let's create a keyspace.")
        ks_name = q.ask(
            "Enter a name for your new keyspace.\nThe name can contain only
letters, "
            "numbers and underscores: ",
            q.non_empty,
        )
        if self.ks_wrapper.exists_keyspace(ks_name):
            print(f"A keyspace named {ks_name} exists.")
        else:
            ks_arn = self.ks_wrapper.create_keyspace(ks_name)
            ks_exists = False
            while not ks_exists:
                wait(3)
                ks_exists = self.ks_wrapper.exists_keyspace(ks_name)
            print(f"Created a new keyspace.\n\t{ks_arn}.")
        print("The first 10 keyspaces in your account are:\n")
        self.ks_wrapper.list_keyspaces(10)

    @demo_func
    def create_table(self):
        """
        1. Creates a table in the keyspace. The table is configured with a schema
to hold
           movie data and has point-in-time recovery enabled.
        2. Waits for the table to be in an active state.
        3. Displays schema information for the table.
        4. Lists tables in the keyspace.
        """
        print("Let's create a table for movies in your keyspace.")
        table_name = q.ask("Enter a name for your table: ", q.non_empty)
        table = self.ks_wrapper.get_table(table_name)
        if table is not None:
            print(
                f"A table named {table_name} already exists in keyspace "
                f"{self.ks_wrapper.ks_name}."
            )
        else:
```

```
            table_arn = self.ks_wrapper.create_table(table_name)
            print(f"Created table {table_name}:\n\t{table_arn}")
            table = {"status": None}
            print("Waiting for your table to be ready...")
            while table["status"] != "ACTIVE":
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
        print(f"Your table is {table['status']}. Its schema is:")
        pp(table["schemaDefinition"])
        print("\nThe tables in your keyspace are:\n")
        self.ks_wrapper.list_tables()

    @demo_func
    def ensure_tls_cert(self):
        """
        Ensures you have a TLS certificate available to use to secure the
connection
        to the keyspace. This function downloads a default certificate or lets
you
        specify your own.
        """
        print("To connect to your keyspace, you must have a TLS certificate.")
        print("Checking for TLS certificate...")
        cert_path = os.path.join(
            os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
        )
        if not os.path.exists(cert_path):
            cert_choice = q.ask(
                f"Press enter to download a certificate from
{QueryManager.CERT_URL} "
                f"or enter the full path to the certificate you want to use: "
            )
            if cert_choice:
                cert_path = cert_choice
            else:
                cert = requests.get(QueryManager.CERT_URL).text
                with open(cert_path, "w") as cert_file:
                    cert_file.write(cert)
        else:
            q.ask(f"Certificate {cert_path} found. Press Enter to continue.")
        print(
            f"Certificate {cert_path} will be used to secure the connection to
your keyspace."
        )
```

```
        return cert_path

    @demo_func
    def query_table(self, qm, movie_file):
        """
        1. Adds movies to the table from a sample movie data file.
        2. Gets a list of movies from the table and lets you select one.
        3. Displays more information about the selected movie.
        """
        qm.add_movies(self.ks_wrapper.table_name, movie_file)
        movies = qm.get_movies(self.ks_wrapper.table_name)
        print(f"Added {len(movies)} movies to the table:")
        sel = q.choose("Pick one to learn more about it: ", [m.title for m in
movies])
        movie_choice = qm.get_movie(
            self.ks_wrapper.table_name, movies[sel].title, movies[sel].year
        )
        print(movie_choice.title)
        print(f"\tReleased: {movie_choice.release_date}")
        print(f"\tPlot: {movie_choice.plot}")

    @demo_func
    def update_and_restore_table(self, qm):
        """
        1. Updates the table by adding a column to track watched movies.
        2. Marks some of the movies as watched.
        3. Gets the list of watched movies from the table.
        4. Restores to a movies_restored table at a previous point in time.
        5. Gets the list of movies from the restored table.
        """
        print("Let's add a column to record which movies you've watched.")
        pre_update_timestamp = datetime.utcnow()
        print(
            f"Recorded the current UTC time of {pre_update_timestamp} so we can
restore the table later."
        )
        self.ks_wrapper.update_table()
        print("Waiting for your table to update...")
        table = {"status": "UPDATING"}
        while table["status"] != "ACTIVE":
            wait(5)
            table = self.ks_wrapper.get_table(self.ks_wrapper.table_name)
        print("Column 'watched' added to table.")
        q.ask(
```

```
            "Let's mark some of the movies as watched. Press Enter when you're
ready.\n"
        )
        movies = qm.get_movies(self.ks_wrapper.table_name)
        for movie in movies[:10]:
            qm.watched_movie(self.ks_wrapper.table_name, movie.title, movie.year)
            print(f"Marked {movie.title} as watched.")
        movies = qm.get_movies(self.ks_wrapper.table_name, watched=True)
        print("-" * 88)
        print("The watched movies in our table are:\n")
        for movie in movies:
            print(movie.title)
        print("-" * 88)
        if q.ask(
            "Do you want to restore the table to the way it was before all of
these\n"
            "updates? Keep in mind, this can take up to 20 minutes. (y/n) ",
            q.is_yesno,
        ):
            starting_table_name = self.ks_wrapper.table_name
            table_name_restored =
self.ks_wrapper.restore_table(pre_update_timestamp)
            table = {"status": "RESTORING"}
            while table["status"] != "ACTIVE":
                wait(10)
                table = self.ks_wrapper.get_table(table_name_restored)
            print(
                f"Restored {starting_table_name} to {table_name_restored} "
                f"at a point in time of {pre_update_timestamp}."
            )
            movies = qm.get_movies(table_name_restored)
            print("Now the movies in our table are:")
            for movie in movies:
                print(movie.title)

    def cleanup(self, cert_path):
        """
        1. Deletes the table and waits for it to be removed.
        2. Deletes the keyspace.

        :param cert_path: The path of the TLS certificate used in the demo. If
the
                          certificate was downloaded during the demo, it is
removed.
```

```
        """
        if q.ask(
            f"Do you want to delete your {self.ks_wrapper.table_name} table and "
            f"{self.ks_wrapper.ks_name} keyspace? (y/n) ",
            q.is_yesno,
        ):
            table_name = self.ks_wrapper.table_name
            self.ks_wrapper.delete_table()
            table = self.ks_wrapper.get_table(table_name)
            print("Waiting for the table to be deleted.")
            while table is not None:
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
            print("Table deleted.")
            self.ks_wrapper.delete_keyspace()
            print(
                "Keyspace deleted. If you chose to restore your table during the
"
                "demo, the original table is also deleted."
            )
            if cert_path == os.path.join(
                os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
            ) and os.path.exists(cert_path):
                os.remove(cert_path)
                print("Removed certificate that was downloaded for this demo.")

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

        print("-" * 88)
        print("Welcome to the Amazon Keyspaces (for Apache Cassandra) demo.")
        print("-" * 88)

        self.create_keyspace()
        self.create_table()
        cert_file_path = self.ensure_tls_cert()
        # Use a context manager to ensure the connection to the keyspace is
closed.
        with QueryManager(
            cert_file_path, boto3.DEFAULT_SESSION, self.ks_wrapper.ks_name
        ) as qm:
            self.query_table(qm, "../../../resources/sample_files/movies.json")
            self.update_and_restore_table(qm)
```

```
        self.cleanup(cert_file_path)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = KeyspaceScenario(KeyspaceWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que envolva ações de keyspace e tabelas.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
```

```
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists


    def list_keyspaces(self, limit):
        """
```

```
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems":
limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response


    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
```

```
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.
```

```
        :param restore_timestamp: The point in time to restore the table. This
time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return restored_table_name


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Defina uma classe que crie uma conexão TLS com um keyspace, autentique com SigV4 e envie consultas CQL para uma tabela no keyspace.

```
class QueryManager:
    """
    Manages queries to an Amazon Keyspaces (for Apache Cassandra) keyspace.
    Queries are secured by TLS and authenticated by using the Signature V4
 (SigV4)
    AWS signing protocol. This is more secure than sending username and password
    with a plain-text authentication provider.

    This example downloads a default certificate to secure TLS, or lets you
 specify
    your own.

    This example uses a table of movie data to demonstrate basic queries.
    """

    DEFAULT_CERT_FILE = "sf-class2-root.crt"
    CERT_URL = f"https://certs.secureserver.net/repository/sf-class2-root.crt"

    def __init__(self, cert_file_path, boto_session, keyspace_name):
```

```
        """
        :param cert_file_path: The path and file name of the certificate used for
 TLS.
        :param boto_session: A Boto3 session. This is used to acquire your AWS
 credentials.
        :param keyspace_name: The name of the keyspace to connect.
        """
        self.cert_file_path = cert_file_path
        self.boto_session = boto_session
        self.ks_name = keyspace_name
        self.cluster = None
        self.session = None

    def __enter__(self):
        """
        Creates a session connection to the keyspace that is secured by TLS and
        authenticated by SigV4.
        """
        ssl_context = SSLContext(PROTOCOL_TLSv1_2)
        ssl_context.load_verify_locations(self.cert_file_path)
        ssl_context.verify_mode = CERT_REQUIRED
        auth_provider = SigV4AuthProvider(self.boto_session)
        contact_point = f"cassandra.
{self.boto_session.region_name}.amazonaws.com"
        exec_profile = ExecutionProfile(
            consistency_level=ConsistencyLevel.LOCAL_QUORUM,
            load_balancing_policy=DCAwareRoundRobinPolicy(),
        )
        self.cluster = Cluster(
            [contact_point],
            ssl_context=ssl_context,
            auth_provider=auth_provider,
            port=9142,
            execution_profiles={EXEC_PROFILE_DEFAULT: exec_profile},
            protocol_version=4,
        )
        self.cluster.__enter__()
        self.session = self.cluster.connect(self.ks_name)
        return self

    def __exit__(self, *args):
        """
        Exits the cluster. This shuts down all existing session connections.
        """
```

```
        self.cluster.__exit__(*args)

    def add_movies(self, table_name, movie_file_path):
        """
        Gets movies from a JSON file and adds them to a table in the keyspace.

        :param table_name: The name of the table.
        :param movie_file_path: The path and file name of a JSON file that
 contains movie data.
        """
        with open(movie_file_path, "r") as movie_file:
            movies = json.loads(movie_file.read())
        stmt = self.session.prepare(
            f"INSERT INTO {table_name} (year, title, release_date, plot) VALUES
 (?, ?, ?, ?);"
        )
        for movie in movies[:20]:
            self.session.execute(
                stmt,
                parameters=[
                    movie["year"],
                    movie["title"],
                    date.fromisoformat(movie["info"]
["release_date"].partition("T")[0]),
                    movie["info"]["plot"],
                ],
            )

    def get_movies(self, table_name, watched=None):
        """
        Gets the title and year of the full list of movies from the table.

        :param table_name: The name of the movie table.
        :param watched: When specified, the returned list of movies is filtered
 to
                        either movies that have been watched or movies that have
 not
                        been watched. Otherwise, all movies are returned.
        :return: A list of movies in the table.
        """
        if watched is None:
            stmt = SimpleStatement(f"SELECT title, year from {table_name}")
            params = None
        else:
```

```
            stmt = SimpleStatement(
                f"SELECT title, year from {table_name} WHERE watched = %s ALLOW
FILTERING"
            )
            params = [watched]
        return self.session.execute(stmt, parameters=params).all()

    def get_movie(self, table_name, title, year):
        """
        Gets a single movie from the table, by title and year.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        :return: The requested movie.
        """
        return self.session.execute(
            SimpleStatement(
                f"SELECT * from {table_name} WHERE title = %s AND year = %s"
            ),
            parameters=[title, year],
        ).one()

    def watched_movie(self, table_name, title, year):
        """
        Updates a movie as having been watched.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        """
        self.session.execute(
            SimpleStatement(
                f"UPDATE {table_name} SET watched=true WHERE title = %s AND year
= %s"
            ),
            parameters=[title, year],
        )
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Ações para o Amazon Keyspaces usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Keyspaces com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Keyspaces (para Apache Cassandra).

Exemplos

- Use CreateKeyspacecom um AWS SDK
- Use CreateTablecom um AWS SDK
- Use DeleteKeyspacecom um AWS SDK
- Use DeleteTablecom um AWS SDK
- Use GetKeyspacecom um AWS SDK
- Use GetTablecom um AWS SDK
- Use ListKeyspacescom um AWS SDK
- Use ListTablescom um AWS SDK
- Use RestoreTablecom um AWS SDK
- Use UpdateTablecom um AWS SDK

Use **`CreateKeyspace`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateKeyspace`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create a new keyspace.
/// </summary>
/// <param name="keyspaceName">The name for the new keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
public async Task<string> CreateKeyspace(string keyspaceName)
{
    var response =
        await _amazonKeyspaces.CreateKeyspaceAsync(
            new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.ResourceArn;
}
```

- Para obter detalhes da API, consulte CreateKeyspacea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           CreateKeyspaceRequest keyspaceRequest =
CreateKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
           System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

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

- Para obter detalhes da API, consulte CreateKeyspacea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Para obter detalhes da API, consulte a CreateKeyspacereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""
```

```
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn
```

- Para obter detalhes da API, consulte a CreateKeyspaceReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->createkeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysconflictexception.
        MESSAGE 'Keyspace already exists.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateKeyspacereferência da API AWS SDK for SAP ABAP.

Use **`CreateTable`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create a new Amazon Keyspaces table.
   /// </summary>
   /// <param name="keyspaceName">The keyspace where the table will be
created.</param>
   /// <param name="schema">The schema for the new table.</param>
   /// <param name="tableName">The name of the new table.</param>
   /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
   public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
schema, string tableName)
   {
       var request = new CreateTableRequest
       {
           KeyspaceName = keyspaceName,
           SchemaDefinition = schema,
           TableName = tableName,
           PointInTimeRecovery = new PointInTimeRecovery { Status =
PointInTimeRecoveryStatus.ENABLED }
       };

       var response = await _amazonKeyspaces.CreateTableAsync(request);
       return response.ResourceArn;
   }
```

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           // Set the columns.
           ColumnDefinition defTitle = ColumnDefinition.builder()
                   .name("title")
                   .type("text")
                   .build();

           ColumnDefinition defYear = ColumnDefinition.builder()
                   .name("year")
                   .type("int")
                   .build();

           ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                   .name("release_date")
                   .type("timestamp")
                   .build();

           ColumnDefinition defPlot = ColumnDefinition.builder()
                   .name("plot")
                   .type("text")
                   .build();

           List<ColumnDefinition> colList = new ArrayList<>();
           colList.add(defTitle);
           colList.add(defYear);
           colList.add(defReleaseDate);
           colList.add(defPlot);

           // Set the keys.
           PartitionKey yearKey = PartitionKey.builder()
```

```
                .name("year")
                .build();

        PartitionKey titleKey = PartitionKey.builder()
                .name("title")
                .build();

        List<PartitionKey> keyList = new ArrayList<>();
        keyList.add(yearKey);
        keyList.add(titleKey);

        SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                .partitionKeys(keyList)
                .allColumns(colList)
                .build();

        PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                .status(PointInTimeRecoveryStatus.ENABLED)
                .build();

        CreateTableRequest tableRequest = CreateTableRequest.builder()
                .keyspaceName(keySpace)
                .tableName(tableName)
                .schemaDefinition(schemaDefinition)
                .pointInTimeRecovery(timeRecovery)
                .build();

        CreateTableResponse response = keyClient.createTable(tableRequest);
        System.out.println("The table ARN is " + response.resourceArn());

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

- Para obter detalhes da API, consulte CreateTablea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
```

```
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- Para obter detalhes da API, consulte a CreateTablereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
```

```
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]
```

- Para obter detalhes da API, consulte a CreateTableReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Define schema with columns
```

```
        DATA(lt_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'title' iv_type =
 'text' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'year' iv_type =
 'int' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'release_date' iv_type =
 'timestamp' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'plot' iv_type =
 'text' ) )
        ).

        " Define partition keys
        DATA(lt_partition_keys) = VALUE /aws1/
cl_kyspartitionkey=>tt_partitionkeylist(
          ( NEW /aws1/cl_kyspartitionkey( iv_name = 'year' ) )
          ( NEW /aws1/cl_kyspartitionkey( iv_name = 'title' ) )
        ).

        " Create schema definition
        DATA(lo_schema) = NEW /aws1/cl_kysschemadefinition(
          it_allcolumns = lt_columns
          it_partitionkeys = lt_partition_keys ).

        " Enable point-in-time recovery
        DATA(lo_pitr) = NEW /aws1/cl_kyspointintimerec(
          iv_status = 'ENABLED' ).

        oo_result = lo_kys->createtable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name
          io_schemadefinition = lo_schema
          io_pointintimerecovery = lo_pitr ).
        MESSAGE 'Table created successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTablereferência da API AWS SDK for SAP ABAP.

Use **`DeleteKeyspace`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteKeyspace`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete an existing keyspace.
/// </summary>
/// <param name="keyspaceName"></param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyspace(string keyspaceName)
{
    var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
        new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DeleteKeyspacea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           keyClient.deleteKeyspace(deleteKeyspaceRequest);

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

- Para obter detalhes da API, consulte DeleteKeyspacea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- Para obter detalhes da API, consulte a DeleteKeyspacereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
```

```
        return cls(keyspaces_client)


    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteKeyspaceReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_kys->deletekeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
```

```
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteKeyspacereferência da API AWS SDK for SAP ABAP.

Use **`DeleteTable`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteTable(KeyspacesClient keyClient, String
keyspaceName, String tableName) {
       try {
           DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           keyClient.deleteTable(tableRequest);

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

- Para obter detalhes da API, consulte DeleteTablea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- Para obter detalhes da API, consulte a DeleteTablereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
```

```
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTableReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_kys->deletetable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name ).
        MESSAGE 'Table deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTablereferência da API AWS SDK for SAP ABAP.

Use **`GetKeyspace`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetKeyspace`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get data about a keyspace.
/// </summary>
/// <param name="keyspaceName">The name of the keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
public async Task<string> GetKeyspace(string keyspaceName)
{
    var response = await _amazonKeyspaces.GetKeyspaceAsync(
        new GetKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.ResourceArn;
}
```

- Para obter detalhes da API, consulte GetKeyspacea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
       try {
```

```
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            GetKeyspaceResponse response =
 keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

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

- Para obter detalhes da API, consulte GetKeyspacea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse =
 keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- Para obter detalhes da API, consulte a GetKeyspacereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
```

```
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists
```

- Para obter detalhes da API, consulte a GetKeyspaceReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->getkeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysresourcenotfoundex.
        MESSAGE 'Keyspace does not exist.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
```

```
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetKeyspacereferência da API AWS SDK for SAP ABAP.

Use **`GetTable`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }
```

- Para obter detalhes da API, consulte GetTablea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           while (!tableStatus) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println(". The table status is " + status);

               if (status.compareTo("ACTIVE") == 0) {
                   tableStatus = true;
               }
               Thread.sleep(500);
           }

           List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
           for (ColumnDefinition def : cols) {
               System.out.println("The column name is " + def.name());
```

```
                System.out.println("The column type is " + def.type());
            }

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

- Para obter detalhes da API, consulte GetTablea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
```

```
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? =
 response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a GetTablereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
```

```
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- Para obter detalhes da API, consulte a GetTableReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->gettable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name ).
        MESSAGE 'Table information retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysresourcenotfoundex.
        MESSAGE 'Table does not exist.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetTablereferência da API AWS SDK for SAP ABAP.

Use **`ListKeyspaces`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListKeyspaces`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }
```

- Para obter detalhes da API, consulte ListKeyspacesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
       try {
           ListKeyspacesRequest keyspacesRequest =
ListKeyspacesRequest.builder()
                   .maxResults(10)
                   .build();

           ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
           listRes.stream()
                   .flatMap(r -> r.keyspaces().stream())
                   .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

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

- Para obter detalhes da API, consulte ListKeyspacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListKeyspacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
```

```
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems":
limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListKeyspacesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->listkeyspaces(
          iv_maxresults = iv_max_results ).
        MESSAGE 'Keyspaces listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListKeyspacesreferência da API AWS SDK for SAP ABAP.

Use **`ListTables`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListTables`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new
ListTablesRequest { KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }
```

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listTables(KeyspacesClient keyClient, String keyspaceName)
 {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
 keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
 content.resourceArn() +
                            " Table name: " + content.tableName()));

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

- Para obter detalhes da API, consulte ListTablesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
```

```
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListTablesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_tables(self):
```

```
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListTablesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->listtables(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Tables listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTablesreferência da API AWS SDK for SAP ABAP.

Use **`RestoreTable`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `RestoreTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
```

```
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }
```

- Para obter detalhes da API, consulte RestoreTablea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void restoreTable(KeyspacesClient keyClient, String
keyspaceName, ZonedDateTime utc) {
       try {
           Instant myTime = utc.toInstant();
           RestoreTableRequest restoreTableRequest =
RestoreTableRequest.builder()
                   .restoreTimestamp(myTime)
                   .sourceTableName("Movie")
                   .targetKeyspaceName(keyspaceName)
                   .targetTableName("MovieRestore")
                   .sourceKeyspaceName(keyspaceName)
                   .build();

           RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
           System.out.println("The ARN of the restored table is " +
response.restoredTableARN());
```

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

- Para obter detalhes da API, consulte RestoreTablea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
```

```
}
```

- Para obter detalhes da API, consulte a RestoreTablereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This
 time
                                  must be in UTC format.
```

```
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return restored_table_name
```

- Para obter detalhes da API, consulte a RestoreTableReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->restoretable(
          iv_sourcekeyspacename = iv_source_keyspace_name
          iv_sourcetablename = iv_source_table_name
```

```
          iv_targetkeyspacename = iv_target_keyspace_name
          iv_targettablename = iv_target_table_name
          iv_restoretimestamp = iv_restore_timestamp ).
        MESSAGE 'Table restore initiated successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RestoreTablereferência da API AWS SDK for SAP ABAP.

Use **`UpdateTable`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `UpdateTable`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
```

```
        var newColumn = new ColumnDefinition { Name = "watched", Type =
"boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }
```

- Para obter detalhes da API, consulte UpdateTablea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();
```

```
            keyClient.updateTable(tableRequest);

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

- Para obter detalhes da API, consulte UpdateTablea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
```

```
    }
}
```

- Para obter detalhes da API, consulte a UpdateTablereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
```

```
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateTableReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Add a new column to track watched movies
        DATA(lt_add_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'watched' iv_type =
 'boolean' ) )
        ).

        oo_result = lo_kys->updatetable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name
```

```
          it_addcolumns = lt_add_columns ).
        MESSAGE 'Table updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateTablereferência da API AWS SDK for SAP ABAP.

Exemplos de código para Kinesis usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Kinesis com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Kinesis: mais informações sobre o Kinesis.
- Referência da API Kinesis: detalhes sobre todas as ações disponíveis do Kinesis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de uso do Kinesis AWS SDKs
  - Aprenda o básico do Kinesis com um AWS SDK
  - Ações para o Kinesis usando AWS SDKs
    - Use AddTagsToStreamcom um AWS SDK ou CLI

    - Use CreateStreamcom um AWS SDK ou CLI
    - Use DeleteStreamcom um AWS SDK ou CLI
    - Use DeregisterStreamConsumercom um AWS SDK ou CLI
    - Use DescribeStreamcom um AWS SDK ou CLI
    - Use GetRecordscom um AWS SDK ou CLI
    - Usar GetShardIterator com uma CLI
    - Use ListStreamConsumerscom um AWS SDK
    - Use ListStreamscom um AWS SDK ou CLI
    - Use ListTagsForStreamcom um AWS SDK ou CLI
    - Use PutRecordcom um AWS SDK ou CLI
    - Use PutRecordscom um AWS SDK ou CLI
    - Use RegisterStreamConsumercom um AWS SDK ou CLI
- Exemplos de tecnologia sem servidor para o Kinesis
  - Invocar uma função do Lambda em um trigger do Kinesis
  - Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

Exemplos básicos de uso do Kinesis AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Kinesis com AWS SDKs.

Exemplos

- Aprenda o básico do Kinesis com um AWS SDK
- Ações para o Kinesis usando AWS SDKs
  - Use AddTagsToStreamcom um AWS SDK ou CLI
  - Use CreateStreamcom um AWS SDK ou CLI
  - Use DeleteStreamcom um AWS SDK ou CLI
  - Use DeregisterStreamConsumercom um AWS SDK ou CLI
  - Use DescribeStreamcom um AWS SDK ou CLI
  - Use GetRecordscom um AWS SDK ou CLI
  - Usar GetShardIterator com uma CLI
  - Use ListStreamConsumerscom um AWS SDK

- Use ListStreamscom um AWS SDK ou CLI
- Use ListTagsForStreamcom um AWS SDK ou CLI
- Use PutRecordcom um AWS SDK ou CLI
- Use PutRecordscom um AWS SDK ou CLI
- Use RegisterStreamConsumercom um AWS SDK ou CLI

Aprenda o básico do Kinesis com um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um fluxo e inserir um registro nele.
- Criar um iterador de fragmento.
- Ler o registro e, em seguida, limpar os recursos.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA lo_stream_describe_result TYPE REF TO /aws1/cl_knsdescrstreamoutput.
DATA lo_stream_description TYPE REF TO /aws1/cl_knsstreamdescription.
DATA lo_sharditerator TYPE REF TO /aws1/cl_knsgetsharditerator01.
DATA lo_record_result TYPE REF TO /aws1/cl_knsputrecordoutput.

"Create stream."
TRY.
    lo_kns->createstream(
        iv_streamname = iv_stream_name
        iv_shardcount = iv_shard_count ).
    MESSAGE 'Stream created.' TYPE 'I'.
  CATCH /aws1/cx_knsinvalidargumentex.
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
```

```
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceeded
 exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourceinuseex.
        MESSAGE 'The request processing has failed because the resource is in
 use.' TYPE 'E'.
    ENDTRY.

    "Wait for stream to becomes active."
    lo_stream_describe_result = lo_kns->describestream( iv_streamname =
 iv_stream_name ).
    lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
    WHILE lo_stream_description->get_streamstatus( ) <> 'ACTIVE'.
      IF sy-index = 30.
        EXIT.               "maximum 5 minutes"
      ENDIF.
      WAIT UP TO 10 SECONDS.
      lo_stream_describe_result = lo_kns->describestream( iv_streamname =
 iv_stream_name ).
      lo_stream_description = lo_stream_describe_result-
>get_streamdescription( ).
    ENDWHILE.

    "Create record."
    TRY.
        lo_record_result = lo_kns->putrecord(
            iv_streamname = iv_stream_name
            iv_data       = iv_data
            iv_partitionkey = iv_partition_key ).
        MESSAGE 'Record created.' TYPE 'I'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_knskmsaccessdeniedex.
        MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
 'E'.
      CATCH /aws1/cx_knskmsdisabledex.
        MESSAGE 'KMS key used is disabled.' TYPE 'E'.
      CATCH /aws1/cx_knskmsinvalidstateex.
        MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
      CATCH /aws1/cx_knskmsnotfoundex.
        MESSAGE 'KMS key used is not found.' TYPE 'E'.
      CATCH /aws1/cx_knskmsoptinrequired.
        MESSAGE 'KMS key option is required.' TYPE 'E'.
      CATCH /aws1/cx_knskmsthrottlingex.
```

```
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request
quotas.' TYPE 'E'.
      CATCH /aws1/cx_knsprovthruputexcdex.
        MESSAGE 'The request rate for the stream is too high, or the requested
data is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.

    "Create a shard iterator in order to read the record."
    TRY.
        lo_sharditerator = lo_kns->getsharditerator(
          iv_shardid = lo_record_result->get_shardid( )
          iv_sharditeratortype = iv_sharditeratortype
          iv_streamname = iv_stream_name ).
        MESSAGE 'Shard iterator created.' TYPE 'I'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_knsprovthruputexcdex.
        MESSAGE 'The request rate for the stream is too high, or the requested
data is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcenotfound.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.

    "Read the record."
    TRY.
        oo_result = lo_kns->getrecords(                    " oo_result is
returned for testing purposes. "
            iv_sharditerator   = lo_sharditerator->get_sharditerator( ) ).
        MESSAGE 'Shard iterator created.' TYPE 'I'.
      CATCH /aws1/cx_knsexpirediteratorex.
        MESSAGE 'Iterator expired.' TYPE 'E'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_knskmsaccessdeniedex.
        MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
      CATCH /aws1/cx_knskmsdisabledex.
        MESSAGE 'KMS key used is disabled.' TYPE 'E'.
      CATCH /aws1/cx_knskmsinvalidstateex.
        MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
      CATCH /aws1/cx_knskmsnotfoundex.
        MESSAGE 'KMS key used is not found.' TYPE 'E'.
```

```
      CATCH /aws1/cx_knskmsoptinrequired.
        MESSAGE 'KMS key option is required.' TYPE 'E'.
      CATCH /aws1/cx_knskmsthrottlingex.
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request
quotas.' TYPE 'E'.
      CATCH /aws1/cx_knsprovthruputexcdex.
        MESSAGE 'The request rate for the stream is too high, or the requested
data is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.

    "Delete stream."
    TRY.
        lo_kns->deletestream(
            iv_streamname = iv_stream_name ).
        MESSAGE 'Stream deleted.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourceinuseex.
        MESSAGE 'The request processing has failed because the resource is in
use.' TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - CreateStream
  - DeleteStream
  - GetRecords
  - GetShardIterator
  - PutRecord

Ações para o Kinesis usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Kinesis com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Kinesis.

Exemplos

- Use AddTagsToStreamcom um AWS SDK ou CLI
- Use CreateStreamcom um AWS SDK ou CLI
- Use DeleteStreamcom um AWS SDK ou CLI
- Use DeregisterStreamConsumercom um AWS SDK ou CLI
- Use DescribeStreamcom um AWS SDK ou CLI
- Use GetRecordscom um AWS SDK ou CLI
- Usar GetShardIterator com uma CLI
- Use ListStreamConsumerscom um AWS SDK
- Use ListStreamscom um AWS SDK ou CLI
- Use ListTagsForStreamcom um AWS SDK ou CLI
- Use PutRecordcom um AWS SDK ou CLI
- Use PutRecordscom um AWS SDK ou CLI
- Use RegisterStreamConsumercom um AWS SDK ou CLI

Use **`AddTagsToStream`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AddTagsToStream`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
```

```
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to apply key/value pairs to an Amazon Kinesis
/// stream.
/// </summary>
public class TagStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        var tags = new Dictionary<string, string>
        {
            { "Project", "Sample Kinesis Project" },
            { "Application", "Sample Kinesis App" },
        };

        var success = await ApplyTagsToStreamAsync(client, streamName, tags);

        if (success)
        {
            Console.WriteLine($"Taggs successfully added to {streamName}.");
        }
        else
        {
            Console.WriteLine("Tags were not added to the stream.");
        }
    }

    /// <summary>
    /// Applies the set of tags to the named Kinesis stream.
    /// </summary>
    /// <param name="client">The initialized Kinesis client.</param>
    /// <param name="streamName">The name of the Kinesis stream to which
    /// the tags will be attached.</param>
    /// <param name="tags">A sictionary containing key/value pairs which
    /// will be used to create the Kinesis tags.</param>
    /// <returns>A Boolean value which represents the success or failure
    /// of AddTagsToStreamAsync.</returns>
    public static async Task<bool> ApplyTagsToStreamAsync(
        IAmazonKinesis client,
        string streamName,
```

```
            Dictionary<string, string> tags)
        {
            var request = new AddTagsToStreamRequest
            {
                StreamName = streamName,
                Tags = tags,
            };

            var response = await client.AddTagsToStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte AddTagsToStreama Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para adicionar tags a um fluxo de dados

O exemplo `add-tags-to-stream` a seguir atribui uma tag com a chave `samplekey` e o valor `example` ao fluxo especificado.

```
aws kinesis add-tags-to-stream \
    --stream-name samplestream \
    --tags samplekey=example
```

Este comando não produz saída.

Para obter mais informações, consulte Adicionar tags a fluxos no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte AddTagsToStreamem Referência de AWS CLI Comandos.

Use **`CreateStream`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateStream`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// This example shows how to create a new Amazon Kinesis stream.
/// </summary>
public class CreateStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        int shardCount = 1;

        var success = await CreateNewStreamAsync(client, streamName,
shardCount);
        if (success)
        {
            Console.WriteLine($"The stream, {streamName} successfully
created.");
```

```
            }
        }

        /// <summary>
        /// Creates a new Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client.</param>
        /// <param name="streamName">The name for the new stream.</param>
        /// <param name="shardCount">The number of shards the new stream will
        /// use. The throughput of the stream is a function of the number of
        /// shards; more shards are required for greater provisioned
        /// throughput.</param>
        /// <returns>A Boolean value indicating whether the stream was created.</
returns>
        public static async Task<bool> CreateNewStreamAsync(IAmazonKinesis
 client, string streamName, int shardCount)
        {
            var request = new CreateStreamRequest
            {
                StreamName = streamName,
                ShardCount = shardCount,
            };

            var response = await client.CreateStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte CreateStreama Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar um fluxo de dados

O exemplo de `create-stream` a seguir cria um fluxo de dados chamado samplestream com três fragmentos.

```
aws kinesis create-stream \
    --stream-name samplestream \
    --shard-count 3
```

Este comando não produz saída.

Para obter mais informações, consulte Criar um fluxo no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte CreateStreamem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = """

                Usage:
```

```
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();
        createStream(kinesisClient, streamName);
        System.out.println("Done");
        kinesisClient.close();
    }

    public static void createStream(KinesisClient kinesisClient, String
 streamName) {
        try {
            CreateStreamRequest streamReq = CreateStreamRequest.builder()
                    .streamName(streamName)
                    .shardCount(1)
                    .build();

            kinesisClient.createStream(streamReq);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateStreama Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria um fluxo. Por padrão, esse cmdlet não retorna nenhuma saída, então a PassThru opção - é adicionada para retornar o valor fornecido ao StreamName parâmetro - para uso posterior.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- Para obter detalhes da API, consulte CreateStreamem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria um fluxo.

```
New-KINStream -StreamName "mystream" -ShardCount 1
```

- Para obter detalhes da API, consulte CreateStreamem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
```

```
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def create(self, name, wait_until_exists=True):
        """
        Creates a stream.

        :param name: The name of the stream.
        :param wait_until_exists: When True, waits until the service reports that
                                  the stream exists, then queries for its
metadata.
        """
        try:
            self.kinesis_client.create_stream(StreamName=name, ShardCount=1)
            self.name = name
            logger.info("Created stream %s.", name)
            if wait_until_exists:
                logger.info("Waiting until exists.")
                self.stream_exists_waiter.wait(StreamName=name)
                self.describe(name)
        except ClientError:
            logger.exception("Couldn't create stream %s.", name)
            raise
```

- Para obter detalhes da API, consulte a CreateStreamReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_stream(client: &Client, stream: &str) -> Result<(), Error> {
```

```
    client
        .create_stream()
        .stream_name(stream)
        .shard_count(4)
        .send()
        .await?;

    println!("Created stream");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateStreamreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_kns->createstream(
           iv_streamname = iv_stream_name
           iv_shardcount = iv_shard_count ).
       MESSAGE 'Stream created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in
use.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateStreamreferência da API AWS SDK for SAP ABAP.

Use **`DeleteStream`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteStream`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to delete an Amazon Kinesis stream.
/// </summary>
public class DeleteStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string streamName = "AmazonKinesisStream";

        var success = await DeleteStreamAsync(client, streamName);

        if (success)
```

```
            {
                Console.WriteLine($"Stream, {streamName} successfully deleted.");
            }
            else
            {
                Console.WriteLine("Stream not deleted.");
            }
        }

        /// <summary>
        /// Deletes a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamName">The name of the string to delete.</param>
        /// <returns>A Boolean value representing the success of the operation.</
returns>
        public static async Task<bool> DeleteStreamAsync(IAmazonKinesis client,
 string streamName)
        {
            // If EnforceConsumerDeletion is true, any consumers
            // of this stream will also be deleted. If it is set
            // to false and this stream has any consumers, the
            // call will fail with a ResourceInUseException.
            var request = new DeleteStreamRequest
            {
                StreamName = streamName,
                EnforceConsumerDeletion = true,
            };

            var response = await client.DeleteStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte DeleteStreama Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir um fluxo de dados

O exemplo de `delete-stream` a seguir exclui o fluxo de dados especificado.

```
aws kinesis delete-stream \
    --stream-name samplestream
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir um fluxo no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte DeleteStreamem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

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

```
public class DeleteDataStream {

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

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }

    public static void deleteStream(KinesisClient kinesisClient, String
 streamName) {
        try {
            DeleteStreamRequest delStream = DeleteStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            kinesisClient.deleteStream(delStream);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteStreama Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui o fluxo especificado. A confirmação será solicitada antes que a operação continue. Para ignorar a solicitação de confirmação, use a opção -Force.

```
Remove-KINStream -StreamName "mystream"
```

- Para obter detalhes da API, consulte DeleteStreamem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui o fluxo especificado. A confirmação será solicitada antes que a operação continue. Para ignorar a solicitação de confirmação, use a opção -Force.

```
Remove-KINStream -StreamName "mystream"
```

- Para obter detalhes da API, consulte DeleteStreamem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""
```

```
    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def delete(self):
        """
        Deletes a stream.
        """
        try:
            self.kinesis_client.delete_stream(StreamName=self.name)
            self._clear()
            logger.info("Deleted stream %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete stream %s.", self.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteStreamReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn remove_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client.delete_stream().stream_name(stream).send().await?;

    println!("Deleted stream.");
```

```
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteStreamreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_kns->deletestream(
           iv_streamname = iv_stream_name ).
       MESSAGE 'Stream deleted.' TYPE 'I'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in
use.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteStreamreferência da API AWS SDK for SAP ABAP.

Use **`DeregisterStreamConsumer`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeregisterStreamConsumer`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

    /// <summary>
    /// Shows how to deregister a consumer from an Amazon Kinesis stream.
    /// </summary>
    public class DeregisterConsumer
    {
        public static async Task Main(string[] args)
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream";
            string consumerName = "CONSUMER_NAME";
            string consumerARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream/consumer/CONSUMER_NAME:000000000000";

            var success = await DeregisterConsumerAsync(client, streamARN,
 consumerARN, consumerName);

            if (success)
            {
                Console.WriteLine($"{consumerName} successfully deregistered.");
            }
            else
            {
                Console.WriteLine($"{consumerName} was not successfully
 deregistered.");
            }
```

```
        }

        /// <summary>
        /// Deregisters a consumer from a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamARN">The ARN of a Kinesis stream.</param>
        /// <param name="consumerARN">The ARN of the consumer.</param>
        /// <param name="consumerName">The name of the consumer.</param>
        /// <returns>A Boolean value representing the success of the operation.</
returns>
        public static async Task<bool> DeregisterConsumerAsync(
            IAmazonKinesis client,
            string streamARN,
            string consumerARN,
            string consumerName)
        {
            var request = new DeregisterStreamConsumerRequest
            {
                StreamARN = streamARN,
                ConsumerARN = consumerARN,
                ConsumerName = consumerName,
            };

            var response = await client.DeregisterStreamConsumerAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte DeregisterStreamConsumera Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como cancelar o registro de um consumidor de fluxo de dados

O exemplo `deregister-stream-consumer` a seguir cancela o registro do consumidor especificado do fluxo de dados especificado.

```
aws kinesis deregister-stream-consumer \
    --stream-arn arn:aws:kinesis:us-west-2:123456789012:stream/samplestream \
    --consumer-name KinesisConsumerApplication
```

Este comando não produz saída.

Para obter mais informações, consulte Desenvolvendo consumidores com recursos aprimorados Fan-Out usando a API do Kinesis Data Streams no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte DeregisterStreamConsumerem Referência de AWS CLI Comandos.

Use **`DescribeStream`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeStream`.

CLI

AWS CLI

Como descrever um fluxo de dados

O exemplo de `describe-stream` a seguir retorna detalhes sobre o fluxo de dados especificado.

```
aws kinesis describe-stream \
    --stream-name samplestream
```

Saída:

```
{
    "StreamDescription": {
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "StartingHashKey": "0",
                    "EndingHashKey": "113427455640312821154458202477256070484"
                },
                "SequenceNumberRange": {
```

```
                    "StartingSequenceNumber":
 "49600871682957036442365024926191073437251060580128653314"
                }
            },
            {
                "ShardId": "shardId-000000000001",
                "HashKeyRange": {
                    "StartingHashKey": "113427455640312821154458202477256070485",
                    "EndingHashKey": "226854911280625642308916404954512140969"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
 "49600871682979337187563555549332609155523708941634633746"
                }
            },
            {
                "ShardId": "shardId-000000000002",
                "HashKeyRange": {
                    "StartingHashKey": "226854911280625642308916404954512140970",
                    "EndingHashKey": "340282366920938463463374607431768211455"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
 "49600871683001637932762086172474144873796357303140614178"
                }
            }
        ],
        "StreamARN": "arn:aws:kinesis:us-west-2:123456789012:stream/
samplestream",
        "StreamName": "samplestream",
        "StreamStatus": "ACTIVE",
        "RetentionPeriodHours": 24,
        "EnhancedMonitoring": [
            {
                "ShardLevelMetrics": []
            }
        ],
        "EncryptionType": "NONE",
        "KeyId": null,
        "StreamCreationTimestamp": 1572297168.0
    }
}
```

Para obter mais informações, consulte Criar e gerenciar fluxos no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte DescribeStreamem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna detalhes do fluxo especificado.

```
Get-KINStream -StreamName "mystream"
```

Saída:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- Para obter detalhes da API, consulte DescribeStreamem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes do fluxo especificado.

```
Get-KINStream -StreamName "mystream"
```

Saída:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- Para obter detalhes da API, consulte DescribeStreamem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response["StreamDescription"]
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
            raise
```

```
        else:
            return self.details
```

- Para obter detalhes da API, consulte a DescribeStreamReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_stream(client: &Client, stream: &str) -> Result<(), Error> {
    let resp = client.describe_stream().stream_name(stream).send().await?;

    let desc = resp.stream_description.unwrap();

    println!("Stream description:");
    println!("  Name:              {}:", desc.stream_name());
    println!("  Status:            {:?}", desc.stream_status());
    println!("  Open shards:       {:?}", desc.shards.len());
    println!("  Retention (hours): {}", desc.retention_period_hours());
    println!("  Encryption:        {:?}", desc.encryption_type.unwrap());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeStreamreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_kns->describestream(
           iv_streamname = iv_stream_name ).
       DATA(lt_stream_description) = oo_result->get_streamdescription( ).
       MESSAGE 'Streams retrieved.' TYPE 'I'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourcenotfoundex.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeStreamreferência da API AWS SDK for SAP ABAP.

Use **`GetRecords`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetRecords`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como obter registros de um fragmento

O exemplo de `get-records` a seguir obtém registros de dados do fragmento de um fluxo de dados do Kinesis usando o iterador de fragmento especificado.

```
aws kinesis get-records \
    --shard-iterator AAAAAAAAAAF7/0mWD7IuHj1yGv/
TKuNgx2ukD5xipCY4cy4gU96orWwZwcSXh3K9tAmGYeOZyLZrvzzeOFVf9iN99hUPw/w/
b0YWYeehfNvnf1DYt5XpDJghLKr3DzgznkTmMymDP3R+3wRKeuEw6/kdxY2yKJH0veaiekaVc4N2VwK/
GvaGP2Hh9Fg7N++q0Adg6fIDQPt4p8RpavDbk+A4sL9SWGE1
```

Saída:

```
{
    "Records": [],
    "MillisBehindLatest": 80742000
}
```

Para obter mais informações, consulte Desenvolvimento de consumidores usando a API Kinesis Data Streams AWS com o SDK for Java no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte GetRecordsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
```

```
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetRecords {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to read from (for
 example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
 streamName) {
        String shardIterator;
```

```
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
        do {
            streamRes = kinesisClient.describeStream(describeStreamRequest);
            shards.addAll(streamRes.streamDescription().shards());

            if (shards.size() > 0) {
                lastShardId = shards.get(shards.size() - 1).shardId();
            }
        } while (streamRes.streamDescription().hasMoreShards());

        GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
                .streamName(streamName)
                .shardIteratorType("TRIM_HORIZON")
                .shardId(lastShardId)
                .build();

        GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
        shardIterator = shardIteratorResult.shardIterator();

        // Continuously read data records from shard.
        List<Record> records;

        // Create new GetRecordsRequest with existing shardIterator.
        // Set maximum records to return to 1000.
        GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
                .shardIterator(shardIterator)
                .limit(1000)
                .build();

        GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

        // Put result into record list. Result may be empty.
        records = result.records();

        // Print records
        for (Record record : records) {
```

```
            SdkBytes byteBuffer = record.data();
            System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
 String(byteBuffer.asByteArray()));
        }
    }
}
```

- Para obter detalhes da API, consulte GetRecordsa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra como retornar e extrair dados de uma série de um ou mais registros. O iterador fornecido Get-KINRecord determina a posição inicial dos registros a serem retornados, os quais, neste exemplo, são capturados em uma variável, $records. Cada registro individual pode então ser acessado indexando a coleção $records. Supondo que os dados no registro sejam texto UTF-8 codificado, o comando final mostra como você pode extrair os dados do MemoryStream objeto e devolvê-los como texto ao console.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Saída:

```
MillisBehindLatest NextShardIterator           Records
------------------ -----------------           -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Saída:

```
ApproximateArrivalTimestamp Data                   PartitionKey SequenceNumber
--------------------------- ----                   ------------ --------------
3/7/2016 5:14:33 PM         System.IO.MemoryStream Key1
 4955986459776...931586
```

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

Saída:

```
test data from string
```

- Para obter detalhes da API, consulte GetRecordsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como retornar e extrair dados de uma série de um ou mais registros. O iterador fornecido Get-KINRecord determina a posição inicial dos registros a serem retornados, os quais, neste exemplo, são capturados em uma variável, $records. Cada registro individual pode então ser acessado indexando a coleção $records. Supondo que os dados no registro sejam texto UTF-8 codificado, o comando final mostra como você pode extrair os dados do MemoryStream objeto e devolvê-los como texto ao console.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Saída:

```
MillisBehindLatest NextShardIterator            Records
------------------ -----------------            -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Saída:

```
ApproximateArrivalTimestamp Data                   PartitionKey SequenceNumber
--------------------------- ----                   ------------ --------------
3/7/2016 5:14:33 PM         System.IO.MemoryStream Key1
 4955986459776...931586
```

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

Saída:

```
test data from string
```

- Para obter detalhes da API, consulte GetRecordsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def get_records(self, max_records):
        """
        Gets records from the stream. This function is a generator that first
 gets
        a shard iterator for the stream, then uses the shard iterator to get
 records
        in batches from the stream. The shard iterator can be accessed through
 the
        'details' property, which is populated using the 'describe' function of
 this class.
        Each batch of records is yielded back to the caller until the specified
        maximum number of records has been retrieved.
```

```
        :param max_records: The maximum number of records to retrieve.
        :return: Yields the current batch of retrieved records.
        """
        try:
            response = self.kinesis_client.get_shard_iterator(
                StreamName=self.name,
                ShardId=self.details["Shards"][0]["ShardId"],
                ShardIteratorType="LATEST",
            )
            shard_iter = response["ShardIterator"]
            record_count = 0
            while record_count < max_records:
                response = self.kinesis_client.get_records(
                    ShardIterator=shard_iter, Limit=10
                )
                shard_iter = response["NextShardIterator"]
                records = response["Records"]
                logger.info("Got %s records.", len(records))
                record_count += len(records)
                yield records
        except ClientError:
            logger.exception("Couldn't get records from stream %s.", self.name)
            raise



    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response["StreamDescription"]
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
            raise
        else:
            return self.details
```

- Para obter detalhes da API, consulte a GetRecordsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_kns->getrecords(             " oo_result is returned for
testing purposes. "
           iv_sharditerator = iv_shard_iterator ).
       DATA(lt_records) = oo_result->get_records( ).
       MESSAGE 'Record retrieved.' TYPE 'I'.
     CATCH /aws1/cx_knsexpirediteratorex.
       MESSAGE 'Iterator expired.' TYPE 'E'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request
quotas.' TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
```

```
        MESSAGE 'The request rate for the stream is too high, or the requested
 data is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetRecordsreferência da API AWS SDK for SAP ABAP.

Usar **`GetShardIterator`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetShardIterator`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para obter um iterador de fragmentos

O exemplo `get-shard-iterator` a seguir usa o tipo iterador de fragmento `AT_SEQUENCE_NUMBER` e gera um iterador de fragmento para começar a ler os registros de dados exatamente da posição indicada pelo número de sequência especificado.

```
aws kinesis get-shard-iterator \
    --stream-name samplestream \
    --shard-id shardId-000000000001 \
    --shard-iterator-type LATEST
```

Saída:

```
{
    "ShardIterator": "AAAAAAAAAAFEvJjIYI+3jw/4aqgH9FifJ+n48XWTh/
IFIsbILP6o5eDueD39NXNBfpZ10WL5K6ADXk8w+5H+Qhd9cFA9k268CPXCz/kebq1TGYI7Vy
+lUkA9BuN3xvATxMBGxRY3zYK05gqgvaIRn94O8SqeEqwhigwZxNWxID3Ej7YYYcxQi8Q/fIrCjGAy/
n2r5Z9G864YpWDfN9upNNQAR/iiOWKs"
```

```
}
```

Para obter mais informações, consulte Desenvolvimento de consumidores usando a API Kinesis Data Streams AWS com o SDK for Java no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte GetShardIteratorem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna um iterador de fragmentos para o fragmento e a posição inicial especificados. Detalhes dos identificadores de fragmentos e dos números de sequência podem ser obtidos na saída do Get-KINStream cmdlet, fazendo referência à coleção Shards do objeto de fluxo retornado. O iterador retornado pode ser usado com o Get-KINRecord cmdlet para extrair registros de dados no fragmento.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Saída:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Para obter detalhes da API, consulte GetShardIteratorem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna um iterador de fragmentos para o fragmento e a posição inicial especificados. Detalhes dos identificadores de fragmentos e dos números de sequência podem ser obtidos na saída do Get-KINStream cmdlet, fazendo referência à coleção Shards do objeto de fluxo retornado. O iterador retornado pode ser usado com o Get-KINRecord cmdlet para extrair registros de dados no fragmento.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Saída:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Para obter detalhes da API, consulte GetShardIteratorem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListStreamConsumers`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListStreamConsumers`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// List the consumers of an Amazon Kinesis stream.
    /// </summary>
    public class ListConsumers
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";
            int maxResults = 10;
```

```
            var consumers = await ListConsumersAsync(client, streamARN,
 maxResults);

            if (consumers.Count > 0)
            {
                consumers
                    .ForEach(c => Console.WriteLine($"Name: {c.ConsumerName} ARN:
 {c.ConsumerARN}"));
            }
            else
            {
                Console.WriteLine("No consumers found.");
            }
        }

        /// <summary>
        /// Retrieve a list of the consumers for a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamARN">The ARN of the stream for which we want to
        /// retrieve a list of clients.</param>
        /// <param name="maxResults">The maximum number of results to return.</
param>
        /// <returns>A list of Consumer objects.</returns>
        public static async Task<List<Consumer>>
 ListConsumersAsync(IAmazonKinesis client, string streamARN, int maxResults)
        {
            var request = new ListStreamConsumersRequest
            {
                StreamARN = streamARN,
                MaxResults = maxResults,
            };

            var response = await client.ListStreamConsumersAsync(request);

            return response.Consumers;
        }
    }
```

- Para obter detalhes da API, consulte ListStreamConsumersa Referência AWS SDK para .NET da API.

Use **`ListStreams`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListStreams`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Retrieves and displays a list of existing Amazon Kinesis streams.
/// </summary>
public class ListStreams
{
    public static async Task Main(string[] args)
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        var response = await client.ListStreamsAsync(new
ListStreamsRequest());

        List<string> streamNames = response.StreamNames;

        if (streamNames.Count > 0)
        {
            streamNames
                .ForEach(s => Console.WriteLine($"Stream name: {s}"));
        }
        else
        {
            Console.WriteLine("No streams were found.");
        }
    }
```

```
    }
```

- Para obter detalhes da API, consulte ListStreamsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar fluxos de dados

O exemplo de `list-streams` a seguir lista todos os fluxos de dados ativos na conta e região atuais.

```
aws kinesis list-streams
```

Saída:

```
{
    "StreamNames": [
        "samplestream",
        "samplestream1"
    ]
}
```

Para obter mais informações, consulte Listar fluxos no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte ListStreamsem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_streams(client: &Client) -> Result<(), Error> {
    let resp = client.list_streams().send().await?;

    println!("Stream names:");

    let streams = resp.stream_names;
    for stream in &streams {
        println!("  {}", stream);
    }

    println!("Found {} stream(s)", streams.len());

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListStreamsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kns->liststreams(        " oo_result is returned for
 testing purposes. "
            "Set Limit to specify that a maximum of streams should be returned."
            iv_limit = iv_limit ).
        DATA(lt_streams) = oo_result->get_streamnames( ).
        MESSAGE 'Streams listed.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceed
 exception.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListStreamsreferência da API AWS SDK for SAP ABAP.

Use **`ListTagsForStream`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTagsForStream`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Shows how to list the tags that have been attached to an Amazon Kinesis
/// stream.
/// </summary>
public class ListTags
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string streamName = "AmazonKinesisStream";

        await ListTagsAsync(client, streamName);
    }

    /// <summary>
    /// List the tags attached to a Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client object.</param>
    /// <param name="streamName">The name of the Kinesis stream for which you
```

```
        /// wish to display tags.</param>
        public static async Task ListTagsAsync(IAmazonKinesis client, string
streamName)
        {
            var request = new ListTagsForStreamRequest
            {
                StreamName = streamName,
                Limit = 10,
            };

            var response = await client.ListTagsForStreamAsync(request);
            DisplayTags(response.Tags);

            while (response.HasMoreTags)
            {
                request.ExclusiveStartTagKey = response.Tags[response.Tags.Count
- 1].Key;
                response = await client.ListTagsForStreamAsync(request);
            }
        }

        /// <summary>
        /// Displays the items in a list of Kinesis tags.
        /// </summary>
        /// <param name="tags">A list of the Tag objects to be displayed.</param>
        public static void DisplayTags(List<Tag> tags)
        {
            tags
                .ForEach(t => Console.WriteLine($"Key: {t.Key} Value:
{t.Value}"));
        }
    }
```

- Para obter detalhes da API, consulte ListTagsForStreama Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar tags para um fluxo de dados

O exemplo `list-tags-for-stream` a seguir lista as tags anexadas ao fluxo de dados especificado.

```
aws kinesis list-tags-for-stream \
    --stream-name samplestream
```

Saída:

```
{
    "Tags": [
        {
            "Key": "samplekey",
            "Value": "example"
        }
    ],
    "HasMoreTags": false
}
```

Para obter mais informações, consulte Adicionar tags a fluxos no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte ListTagsForStreamem Referência de AWS CLI Comandos.

Use **PutRecord**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutRecord`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como gravar um registro em um fluxo de dados

O exemplo de `put-record` a seguir grava um único registro de dados no fluxo de dados especificado usando a chave de partição especificada.

```
aws kinesis put-record \
    --stream-name samplestream \
    --data sampledatarecord \
    --partition-key samplepartitionkey
```

Saída:

```
{
    "ShardId": "shardId-000000000009",
    "SequenceNumber": "49600902273357540915989931256901506243878407835297513618",
    "EncryptionType": "KMS"
}
```

Para obter mais informações, consulte Desenvolvimento de produtores usando a API Amazon Kinesis Data Streams AWS com o SDK for Java no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte PutRecordem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to which records
 are written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        // Ensure that the Kinesis Stream is valid.
        validateStream(kinesisClient, streamName);
        setStockData(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void setStockData(KinesisClient kinesisClient, String
 streamName) {
        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in
 between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x = 0; x < index; x++) {
```

```
                StockTrade trade = stockTradeGenerator.getRandomTrade();
                sendStockTrade(trade, kinesisClient, streamName);
                Thread.sleep(100);
            }

        } catch (KinesisException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }

    private static void sendStockTrade(StockTrade trade, KinesisClient
kinesisClient,
            String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON
serialization by
        // the Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
                .partitionKey(trade.getTickerSymbol()) // We use the ticker
symbol as the partition key, explained in
                                                       // the Supplemental
Information section below.
                .streamName(streamName)
                .data(SdkBytes.fromByteArray(bytes))
                .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String
streamName) {
```

```
        try {
            DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            DescribeStreamResponse describeStreamResponse =
 kinesisClient.describeStream(describeStreamRequest);

            if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
 {
                System.err.println("Stream " + streamName + " is not active.
 Please wait a few moments and try again.");
                System.exit(1);
            }

        } catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
 streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutRecorda Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: grava um registro contendo a string fornecida ao parâmetro -Text.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -
PartitionKey "Key1"
```

Exemplo 2: grava um registro contendo os dados contidos no arquivo especificado. O arquivo é tratado como uma sequência de bytes, portanto, se contiver texto, ele deverá ser gravado com qualquer codificação necessária antes de ser usado com esse cmdlet.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- Para obter detalhes da API, consulte PutRecordem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: grava um registro contendo a string fornecida ao parâmetro -Text.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -
PartitionKey "Key1"
```

Exemplo 2: grava um registro contendo os dados contidos no arquivo especificado. O arquivo é tratado como uma sequência de bytes, portanto, se contiver texto, ele deverá ser gravado com qualquer codificação necessária antes de ser usado com esse cmdlet.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- Para obter detalhes da API, consulte PutRecordem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
```

```
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def put_record(self, data, partition_key):
        """
        Puts data into the stream. The data is formatted as JSON before it is
passed
        to the stream.

        :param data: The data to put in the stream.
        :param partition_key: The partition key to use for the data.
        :return: Metadata about the record, including its shard ID and sequence
number.
        """
        try:
            response = self.kinesis_client.put_record(
                StreamName=self.name, Data=json.dumps(data),
PartitionKey=partition_key
            )
            logger.info("Put record in stream %s.", self.name)
        except ClientError:
            logger.exception("Couldn't put record in stream %s.", self.name)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a PutRecordReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn add_record(client: &Client, stream: &str, key: &str, data: &str) ->
 Result<(), Error> {
    let blob = Blob::new(data);

    client
        .put_record()
        .data(blob)
        .partition_key(key)
        .stream_name(stream)
        .send()
        .await?;

    println!("Put data into stream.");

    Ok(())
}
```

- Para obter detalhes da API, consulte a PutRecordreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_kns->putrecord(            " oo_result is returned for
testing purposes. "
           iv_streamname = iv_stream_name
           iv_data       = iv_data
           iv_partitionkey = iv_partition_key ).
       MESSAGE 'Record created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request
quotas.' TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested
data is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_knsresourcenotfoundex.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a PutRecordreferência da API AWS SDK for SAP ABAP.

Use **`PutRecords`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutRecords`.

CLI

AWS CLI

Para gravar vários registros em um fluxo de dados

O exemplo `put-records` a seguir grava um registro de dados usando a chave de partição especificada e outro registro de dados usando uma chave de partição diferente em uma única chamada.

```
aws kinesis put-records \
    --stream-name samplestream \
    --
records Data=blob1,PartitionKey=partitionkey1 Data=blob2,PartitionKey=partitionkey2
```

Saída:

```
{
    "FailedRecordCount": 0,
    "Records": [
        {
            "SequenceNumber":
 "49600883331171471519674795588238531498465399900093808706",
            "ShardId": "shardId-000000000004"
        },
        {
            "SequenceNumber":
 "49600902273357540915989931256902715169698037101720764562",
            "ShardId": "shardId-000000000009"
        }
    ],
    "EncryptionType": "KMS"
}
```

Para obter mais informações, consulte Desenvolvimento de produtores usando a API Amazon Kinesis Data Streams AWS com o SDK for Java no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte PutRecordsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { PutRecordsCommand, KinesisClient } from "@aws-sdk/client-kinesis";

/**
 * Put multiple records into a Kinesis stream.
 * @param {{ streamArn: string }} config
 */
export const main = async ({ streamArn }) => {
  const client = new KinesisClient({});
  try {
    await client.send(
      new PutRecordsCommand({
        StreamARN: streamArn,
        Records: [
          {
            Data: new Uint8Array(),
            /**
             * Determines which shard in the stream the data record is assigned
 to.
             * Partition keys are Unicode strings with a maximum length limit of
 256
             * characters for each key. Amazon Kinesis Data Streams uses the
 partition
             * key as input to a hash function that maps the partition key and
             * associated data to a specific shard.
             */
            PartitionKey: "TEST_KEY",
          },
          {
            Data: new Uint8Array(),
            PartitionKey: "TEST_KEY",
          },
```

```
        ],
      }),
    );
  } catch (caught) {
    if (caught instanceof Error) {
      //
    } else {
      throw caught;
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    streamArn: {
      type: "string",
      description: "The ARN of the stream.",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- Para obter detalhes da API, consulte PutRecordsa Referência AWS SDK para JavaScript da API.

Use **`RegisterStreamConsumer`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RegisterStreamConsumer`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

    /// <summary>
    /// This example shows how to register a consumer to an Amazon Kinesis
    /// stream.
    /// </summary>
    public class RegisterConsumer
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            string consumerName = "NEW_CONSUMER_NAME";
            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";

            var consumer = await RegisterConsumerAsync(client, consumerName,
 streamARN);

            if (consumer is not null)
            {
                Console.WriteLine($"{consumer.ConsumerName}");
            }
        }

        /// <summary>
        /// Registers the consumer to a Kinesis stream.
        /// </summary>
        /// <param name="client">The initialized Kinesis client object.</param>
```

```
        /// <param name="consumerName">A string representing the consumer.</
param>
        /// <param name="streamARN">The ARN of the stream.</param>
        /// <returns>A Consumer object that contains information about the
 consumer.</returns>
        public static async Task<Consumer> RegisterConsumerAsync(IAmazonKinesis
 client, string consumerName, string streamARN)
        {
            var request = new RegisterStreamConsumerRequest
            {
                ConsumerName = consumerName,
                StreamARN = streamARN,
            };

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

- Para obter detalhes da API, consulte RegisterStreamConsumera Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para registrar um consumidor de fluxo de dados

O exemplo `register-stream-consumer` a seguir registra um consumidor chamado `KinesisConsumerApplication` com o fluxo de dados especificado.

```
aws kinesis register-stream-consumer \
    --stream-arn arn:aws:kinesis:us-west-2:012345678912:stream/samplestream \
    --consumer-name KinesisConsumerApplication
```

Saída:

```
{
    "Consumer": {
        "ConsumerName": "KinesisConsumerApplication",
```

```
        "ConsumerARN": "arn:aws:kinesis:us-west-2: 123456789012:stream/
samplestream/consumer/KinesisConsumerApplication:1572383852",
        "ConsumerStatus": "CREATING",
        "ConsumerCreationTimestamp": 1572383852.0
    }
}
```

Para obter mais informações, consulte Desenvolvendo consumidores com recursos aprimorados Fan-Out usando a API do Kinesis Data Streams no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Para obter detalhes da API, consulte RegisterStreamConsumerem Referência de AWS CLI Comandos.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_kns->registerstreamconsumer(       " oo_result is returned
for testing purposes. "
           iv_streamarn = iv_stream_arn
           iv_consumername = iv_consumer_name ).
       MESSAGE 'Stream consumer registered.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a RegisterStreamConsumerreferência da API AWS SDK for SAP ABAP.

Exemplos de tecnologia sem servidor para o Kinesis

Os exemplos de código a seguir mostram como usar o Kinesis com AWS SDKs.

Exemplos

- Invocar uma função do Lambda em um trigger do Kinesis
- Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

Invocar uma função do Lambda em um trigger do Kinesis

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
```

```
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace KinesisIntegrationSampleCode;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task FunctionHandler(KinesisEvent evnt, ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return;
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                throw;
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
```

```
        return data;
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
 if len(kinesisEvent.Records) == 0 {
  log.Printf("empty Kinesis event received")
  return nil
 }

 for _, record := range kinesisEvent.Records {
  log.Printf("processed Kinesis event with EventId: %v", record.EventID)
  recordDataBytes := record.Kinesis.Data
  recordDataText := string(recordDataBytes)
  log.Printf("record data: %v", recordDataText)
  // TODO: Do interesting work based on the new data
 }
 log.Printf("successfully processed %v records", len(kinesisEvent.Records))
```

```
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
```

```
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+"
 records");
        return null;
    }

}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do Kinesis com o uso do Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      throw err;
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
```

```
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Consumindo um evento do Kinesis com o uso do Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      throw err;
    }
    logger.info(`Successfully processed ${event.Records.length} records.`);
  }
```

```
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be
 marked as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import base64
```

```
def lambda_handler(event, context):

    for record in event['Records']:
        try:
            print(f"Processed Kinesis Event - EventID: {record['eventID']}")
            record_data = base64.b64decode(record['kinesis']
['data']).decode('utf-8')
            print(f"Record Data: {record_data}")
            # TODO: Do interesting work based on the new data
        except Exception as e:
            print(f"An error occurred {e}")
            raise e
    print(f"Successfully processed {len(event['Records'])} records.")
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
```

```
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do Kinesis com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::kinesis::KinesisEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) -> Result<(), Error>
 {
    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    event.payload.records.iter().for_each(|record| {
        tracing::info!("EventId:
 {}",record.event_id.as_deref().unwrap_or_default());

        let record_data = std::str::from_utf8(&record.kinesis.data);
```

```
        match record_data {
            Ok(data) => {
                // log the record data
                tracing::info!("Data: {}", data);
            }
            Err(e) => {
                tracing::error!("Error: {}", e);
            }
        }
    });

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

Os exemplos de código a seguir mostram como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using System.Text.Json.Serialization;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace KinesisIntegration;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task<StreamsEventResponse> FunctionHandler(KinesisEvent evnt,
 ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return new StreamsEventResponse();
        }

        foreach (var record in evnt.Records)
        {
```

```
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                return new StreamsEventResponse
                {
                    BatchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>
                    {
                        new StreamsEventResponse.BatchItemFailure
 { ItemIdentifier = record.Kinesis.SequenceNumber }
                    }
                };
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
        return new StreamsEventResponse();
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}

public class StreamsEventResponse
{
    [JsonPropertyName("batchItemFailures")]
```

```
    public IList<BatchItemFailure> BatchItemFailures { get; set; }
    public class BatchItemFailure
    {
        [JsonPropertyName("itemIdentifier")]
        public string ItemIdentifier { get; set; }
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, record := range kinesisEvent.Records {
  curRecordSequenceNumber := ""

  // Process your record
  if /* Your record processing condition here */ {
   curRecordSequenceNumber = record.Kinesis.SequenceNumber
  }
```

```
  // Add a condition to check if the record processing failed
  if curRecordSequenceNumber != "" {
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": curRecordSequenceNumber})
  }
 }

 kinesisBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return kinesisBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
```

```
public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context
 context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord =
 kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Javascript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed
 item onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Relatando falhas de itens em lote do Kinesis com o uso do Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
  KinesisStreamBatchResponse,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<KinesisStreamBatchResponse> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  logger.info(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};
```

```
async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["kinesis"]["sequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de item em lote do Kinesis com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relate falhas de itens em lote do Kinesis com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::kinesis::KinesisEvent,
    kinesis::KinesisEventRecord,
    streams::{KinesisBatchItemFailure, KinesisEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) ->
 Result<KinesisEventResponse, Error> {
    let mut response = KinesisEventResponse {
        batch_item_failures: vec![],
    };

    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in &event.payload.records {
        tracing::info!(
            "EventId: {}",
            record.event_id.as_deref().unwrap_or_default()
        );

        let record_processing_result = process_record(record);

        if record_processing_result.is_err() {
            response.batch_item_failures.push(KinesisBatchItemFailure {
                item_identifier: record.kinesis.sequence_number.clone(),
```

```
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
            return Ok(response);
        }
    }

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(response)
}

fn process_record(record: &KinesisEventRecord) -> Result<(), Error> {
    let record_data = std::str::from_utf8(record.kinesis.data.as_slice());

    if let Some(err) = record_data.err() {
        tracing::error!("Error: {}", err);
        return Err(Error::from(err));
    }

    let record_data = record_data.unwrap_or_default();

    // do something interesting with the data
    tracing::info!("Data: {}", record_data);

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();
```

```
    run(service_fn(function_handler)).await
}
```

Exemplos de código para AWS KMS usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Key Management Service com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS KMS Guia do desenvolvedor — Mais informações sobre AWS KMS.
- AWS KMS Referência da API — Detalhes sobre todas as AWS KMS ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS KMS usar AWS SDKs
  - Hello AWS Key Management Service
  - Aprenda as noções básicas do AWS KMS com um AWS SDK
  - Ações para AWS KMS usar AWS SDKs
    - Use CreateAliascom um AWS SDK ou CLI
    - Use CreateGrantcom um AWS SDK ou CLI
    - Use CreateKeycom um AWS SDK ou CLI
    - Use Decrypt com um AWS SDK ou CLI

- Use DeleteAliascom um AWS SDK ou CLI
- Use DescribeKeycom um AWS SDK ou CLI
- Use DisableKeycom um AWS SDK ou CLI
- Use EnableKeycom um AWS SDK ou CLI
- Use EnableKeyRotationcom um AWS SDK ou CLI
- Use Encrypt com um AWS SDK ou CLI
- Use GenerateDataKeycom um AWS SDK ou CLI
- Use GenerateDataKeyWithoutPlaintextcom um AWS SDK ou CLI
- Use GenerateRandomcom um AWS SDK ou CLI
- Use GetKeyPolicycom um AWS SDK ou CLI
- Use ListAliasescom um AWS SDK ou CLI
- Use ListGrantscom um AWS SDK ou CLI
- Use ListKeyPoliciescom um AWS SDK ou CLI
- Use ListKeyscom um AWS SDK ou CLI
- Use PutKeyPolicycom um AWS SDK ou CLI
- Use ReEncryptcom um AWS SDK ou CLI
- Use RetireGrantcom um AWS SDK ou CLI
- Use RevokeGrantcom um AWS SDK ou CLI
- Use ScheduleKeyDeletioncom um AWS SDK ou CLI
- Use o Sign com um AWS SDK ou CLI
- Use TagResourcecom um AWS SDK ou CLI
- Use UpdateAliascom um AWS SDK ou CLI
- Use Verify com um AWS SDK ou CLI

- Cenários para AWS KMS usar AWS SDKs
  - Trabalhe com a criptografia de tabelas do DynamoDB usando AWS Command Line Interface v2 da

Exemplos básicos para AWS KMS usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Key Management Service com AWS SDKs.

Exemplos

- Hello AWS Key Management Service
- Aprenda as noções básicas do AWS KMS com um AWS SDK
- Ações para AWS KMS usar AWS SDKs
  - Use CreateAliascom um AWS SDK ou CLI
  - Use CreateGrantcom um AWS SDK ou CLI
  - Use CreateKeycom um AWS SDK ou CLI
  - Use Decrypt com um AWS SDK ou CLI
  - Use DeleteAliascom um AWS SDK ou CLI
  - Use DescribeKeycom um AWS SDK ou CLI
  - Use DisableKeycom um AWS SDK ou CLI
  - Use EnableKeycom um AWS SDK ou CLI
  - Use EnableKeyRotationcom um AWS SDK ou CLI
  - Use Encrypt com um AWS SDK ou CLI
  - Use GenerateDataKeycom um AWS SDK ou CLI
  - Use GenerateDataKeyWithoutPlaintextcom um AWS SDK ou CLI
  - Use GenerateRandomcom um AWS SDK ou CLI
  - Use GetKeyPolicycom um AWS SDK ou CLI
  - Use ListAliasescom um AWS SDK ou CLI
  - Use ListGrantscom um AWS SDK ou CLI
  - Use ListKeyPoliciescom um AWS SDK ou CLI
  - Use ListKeyscom um AWS SDK ou CLI
  - Use PutKeyPolicycom um AWS SDK ou CLI
  - Use ReEncryptcom um AWS SDK ou CLI
  - Use RetireGrantcom um AWS SDK ou CLI
  - Use RevokeGrantcom um AWS SDK ou CLI
  - Use ScheduleKeyDeletioncom um AWS SDK ou CLI
  - Use o Sign com um AWS SDK ou CLI
  - Use TagResourcecom um AWS SDK ou CLI

  - Use UpdateAliascom um AWS SDK ou CLI

- Use Verify com um AWS SDK ou CLI

Hello AWS Key Management Service

O exemplo de código a seguir mostra como começar a usar o AWS Key Management Service.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();
```

```
        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
 Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " +
 exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ListKeysa Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get
 the keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or
 to break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
}
```

- Para obter detalhes da API, consulte ListKeysa Referência AWS SDK para PHP da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_kms->listkeys( ).
    MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListKeysreferência da API AWS SDK for SAP ABAP.

Aprenda as noções básicas do AWS KMS com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar uma chave do KMS.
- Listar chaves KMS para sua conta e obter detalhes sobre elas.
- Habilitar e desabilitar chaves do KMS.
- Gerar uma chave de dados simétrica que possa ser usada para criptografia do lado do cliente.
- Gere uma chave assimétrica usada para assinar dados digitalmente.
- Marcar chaves com tags.
- Excluir chaves do KMS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário em um prompt de comando.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

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

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String accountId = "";
```

```
   private static final Logger logger =
LoggerFactory.getLogger(KMSScenario.class);

   static KMSActions kmsActions = new KMSActions();

   static Scanner scanner = new Scanner(System.in);

   static String aliasName = "alias/dev-encryption-key";

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

           Where:
              granteePrincipal - The principal (user, service account, or group)
to whom the grant or permission is being given.
           """;

       if (args.length != 1) {
           logger.info(usage);
           return;
       }
       String granteePrincipal = args[0];
       String policyName = "default";

       accountId = kmsActions.getAccountId();
       String keyDesc = "Created by the AWS KMS API";

       logger.info(DASHES);
       logger.info("""
           Welcome to the AWS Key Management SDK Basics scenario.

           This program demonstrates how to interact with AWS Key Management
using the AWS SDK for Java (v2).
           The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
           and manage AWS KMS keys and control their use across a wide range of
AWS services and applications.
           KMS provides a centralized and unified approach to managing
encryption keys, making it easier to meet your
           data protection and regulatory compliance requirements.

           This Basics scenario creates two key types:
```

```
            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create a symmetric KMS key\n");
        logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId;
        try {
            CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
            targetKeyId = futureKeyId.join();
            logger.info("A symmetric key was successfully created " +
targetKeyId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            2. Enable a KMS key

            By default, when the SDK creates an AWS key, it is enabled. The next
bit of code checks to
            determine if the key is enabled.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled;
        try {
            CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
            isEnabled = futureIsKeyEnabled.join();
            logger.info("Is the key enabled? {}", isEnabled);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
```

```
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt
data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
        SdkBytes encryptedData;
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
```

```
            """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            By default, when the SDK enables automatic rotation of a KMS key,
            KMS rotates the key material of the KMS key one year (approximately
365 days) from the enable date and every year
            thereafter.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            7. Create a grant

            A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
            It also can allow them to view a KMS key (DescribeKey) and create and
manage grants.
            When authorizing access to a KMS key, grants are considered along
with key policies and IAM policies.
            """);

        waitForInputToContinue(scanner);
        String grantId = null;
```

```
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
```

```
            The revocation of a grant immediately removes the permissions and
access that the grant had provided.
            This means that any principal (user, role, or service) that was
granted access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            Lets decrypt the data that was encrypted in an early step.
            The code uses the same key to decrypt the string that we encrypted
earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptedData = "";
        try {
```

```
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
            A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
            access to KMS keys. Every KMS key must have exactly one key policy.
The statements in the key policy
            determine who has permission to use the KMS key and how they can use
it.
            You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
            must have a key policy.

            By default, when you create a key by using the SDK, a policy is
created that
            gives the AWS account that owns the KMS key full access to the KMS
key.

            Let's try to replace the automatically created policy with the
following policy.

                "Version":"2012-10-17",
                "Statement": [{
                "Effect": "Allow",
```

```
                "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make
sure it exists.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             Signing your data with an AWS key can provide several benefits that
make it an attractive option
             for your data signing needs. By using an AWS KMS key, you can
leverage the
             security controls and compliance features provided by AWS,
             which can help you meet various regulatory requirements and enhance
the overall security posture
             of your organization.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
            boolean success = future.join();
            if (success) {
                logger.info("Sign and verify data operation succeeded.");
            } else {
                logger.error("Sign and verify data operation failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("14. Tag your symmetric KMS Key\n");
        logger.info("""
            By using tags, you can improve the overall management, security, and
governance of your
            KMS keys, making it easier to organize, track, and control access to
your encrypted data within
            your AWS environment
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
kmsActions.tagKMSKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }

    // Deletes KMS resources with user input.
    private static void requestDeleteResources(String aliasName, String
targetKeyId) {
        logger.info(DASHES);
        logger.info("15. Schedule the deletion of the KMS key\n");
        logger.info("""
            By default, KMS applies a waiting period of 30 days,
```

```
            but you can specify a waiting period of 7-30 days. When this
 operation is successful,
            the key state of the KMS key changes to PendingDeletion and the key
 can't be used in any
            cryptographic operations. It remains in this state for the duration
 of the waiting period.

            Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted,
            all data that was encrypted under the KMS key is unrecoverable.
            """);
        logger.info("Would you like to delete the Key Management resources? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS KMS resources.");
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
 kmsActions.deleteSpecificAliasAsync(aliasName);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
 code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
 rt.getMessage());
                }
            }
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
 kmsActions.disableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
 code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
```

```
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteKey(String targetKeyId) {
        try {
            CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
```

```
            }
        }

        try {
            CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteAliasName(String aliasName) {
        try {
            CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
```

```
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Defina uma classe que encapsule as ações do KMS.

```
public class KMSActions {
    private static final Logger logger =
 LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async
 client, with the following configurations:
     * <ul>
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
 creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
```

```
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
```

```
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key
state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
```

```
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and
key ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
```

```
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }

    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of
aliases has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }

    /**
     * Enables key rotation asynchronously for the specified key ID.
     *
     * @param keyId the ID of the key for which to enable key rotation
     * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
     * @throws RuntimeException if there was an error enabling key rotation,
either due to a KMS exception or an unexpected error
     */
    public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
```

```
        EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to enable key rotation: "
+ kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }

    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID
of the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
```

```
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }

    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null
if the operation succeeded, or will throw a {@link RuntimeException} if the
operation failed
     * @throws RuntimeException if there was an error listing the grants, either
due to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();
```

```
        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }

    /**
     * Revokes a grant for the specified AWS KMS key asynchronously.
     *
     * @param keyId   The ID or key ARN of the AWS KMS key.
     * @param grantId The identifier of the grant to be revoked.
     * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
     *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
     *         if the operation is successful, or with a {@code null} value if an
error occurs.
     */
    public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
```

```
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }


    /**
     * Asynchronously decrypts the given encrypted data using the specified key
ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
```

```
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId       the ID of the KMS key to replace the policy for
     * @param policyName  the name of the policy to be replaced
     * @param accountId   the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean
indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = """
    {
      "Version":"2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::%s:root"},
        "Action": "kms:*",
        "Resource": "*"
      }]
    }
    """.formatted(accountId);

        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
```

```
            .policyName(policyName)
            .policy(policy)
            .build();

        // First, get the current policy to check if it exists
        return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
            .thenCompose(response -> {
                logger.info("Current policy exists. Replacing it...");
                return getAsyncClient().putKeyPolicy(keyPolicyRequest);
            })
            .thenApply(putPolicyResponse -> {
                logger.info("The key policy has been replaced.");
                return true;
            })
            .exceptionally(throwable -> {
                if (throwable.getCause() instanceof LimitExceededException) {
                    logger.error("Cannot replace policy, as only one policy is
allowed per key.");
                    return false;
                }
                throw new RuntimeException("Error replacing policy", throwable);
            });
    }


    /**
     * Asynchronously retrieves the key policy for the specified key ID and
policy name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the
policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
```

```
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
     *     <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();
```

```
                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes =
signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging
operation is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
```

```
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key",
throwable);
            });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
            });
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS)
key.
```

```
    *
    * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
    * @return a CompletableFuture that, when completed, indicates that the key
has been disabled successfully
    */
   public CompletableFuture<Void> disableKeyAsync(String keyId) {
       DisableKeyRequest keyRequest = DisableKeyRequest.builder()
           .keyId(keyId)
           .build();

       return getAsyncClient().disableKey(keyRequest)
           .thenRun(() -> {
               logger.info("Key {} has been disabled successfully",keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
           });
   }

   /**
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
    * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted
using the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when
deleting KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion
is scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
           .build();
```

```
        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for
 key ID: " + keyId, throwable);
            });
    }


    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion

- Sign
- TagResource

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "\n";
        echo "-------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management Service using
 the AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service
 that allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys,
 making it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;
        echo "-------------------------------------\n";
        $this->pressEnter();

        $this->kmsClient = new KmsClient([]);
        // Initialize the KmsService class with the client. This allows you to
 override any defaults in the client before giving it to the service class.
```

```
        $this->kmsService = new KmsService($this->kmsClient);

        // 1. Create a symmetric KMS key.
        echo "\n";
        echo "1. Create a symmetric KMS key.\n";
        echo "First, we will create a symmetric KMS key that is used to encrypt
 and decrypt data by invoking createKey().\n";
        $this->pressEnter();

        $key = $this->kmsService->createKey();
        $this->resources['symmetricKey'] = $key['KeyId'];
        echo "Created a customer key with ARN {$key['Arn']}.\n";
        $this->pressEnter();

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code
 checks to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

        $keyInfo = $this->kmsService->describeKey($key['KeyId']);
        if(!$keyInfo['Enabled']){
            echo "The key was not enabled, so we will enable it.\n";
            $this->pressEnter();
            $this->kmsService->enableKey($key['KeyId']);
            echo "The key was successfully enabled.\n";
        }else{
            echo "The key was already enabled, so there was no need to enable it.
\n";
        }
        $this->pressEnter();

        // 3. Encrypt data using the symmetric KMS key.
        echo "\n";
        echo "3. Encrypt data using the symmetric KMS key.\n";
        echo "One of the main uses of symmetric keys is to encrypt and decrypt
 data.\n";
        echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
 SYMMETRIC_DEFAULT encryption algorithm.\n";
        $this->pressEnter();
        $text = "Hello, AWS KMS!";
        $encryption = $this->kmsService->encrypt($key['KeyId'], $text);
```

```
        echo "The plaintext data was successfully encrypted with the algorithm:
 {$encryption['EncryptionAlgorithm']}.\n";
        $this->pressEnter();

        // 4. Create an alias.
        echo "\n";
        echo "4. Create an alias.\n";
        $aliasInput = testable_readline("Please enter an alias prefixed with
 \"alias/\" or press enter to use a default value: ");
        if($aliasInput == ""){
            $aliasInput = "alias/dev-encryption-key";
        }
        $this->kmsService->createAlias($key['KeyId'], $aliasInput);
        $this->resources['alias'] = $aliasInput;
        echo "The alias \"$aliasInput\" was successfully created.\n";
        $this->pressEnter();

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
        echo "6. Enable automatic rotation of the KMS key.\n";
        echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
 from the enable date and every year
thereafter.";
        $this->pressEnter();
        $this->kmsService->enableKeyRotation($key['KeyId']);
        echo "The key's rotation was successfully set for key:
 {$key['KeyId']}\n";
        $this->pressEnter();
```

```
        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
 grants.
When authorizing access to a KMS key, grants are considered along with key
 policies and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon
 Web Services accounts, IAM users, IAM roles, federated users, and assumed
 role users. For help with the ARN syntax for a principal, see IAM ARNs in the
 Identity and Access Management User Guide. \nTo skip this step, press enter
 without any other values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
 $operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

        // 8. List grants for the KMS key.
        if($granteeARN){
            echo "8. List grants for the KMS key.\n\n";
            $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
            foreach($grantsPaginator as $page){
                foreach($page['Grants'] as $grant){
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

        // 9. Revoke the grant.
```

```
        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted
 earlier in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
 $encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are
 the primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in
 the key policy determine who has permission to use the KMS key and how they can
 use it.\n";
        echo " You can also use IAM policies and grants to control access to the
 KMS key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
```

```
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

        // 12. Retrieve the key policy.
        echo "\n";
        echo "12. Retrieve the key policy.\n";
        echo "Let's get some information about the new policy and print it to the
 screen.\n";
        $this->pressEnter();
        $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
        echo "We got the info! Here is the policy: \n";
        echo $policyInfo['Policy'] . "\n";
        $this->pressEnter();

        // 13. Create an asymmetric KMS key and sign data.
        echo "\n";
        echo "13. Create an asymmetric KMS key and sign data.\n";
        echo "Signing your data with an AWS key can provide several benefits that
 make it an attractive option for your data signing needs.\n";
        echo "By using an AWS KMS key, you can leverage the security controls and
 compliance features provided by AWS, which can help you meet various regulatory
 requirements and enhance the overall security posture of your organization.\n";
        echo "First we'll create the asymmetric key.\n";
        $this->pressEnter();
        $keySpec = "RSA_2048";
        $keyUsage = "SIGN_VERIFY";
        $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
        $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text,
 $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
 $sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
```

```
        $this->pressEnter();

        // 14. Tag the symmetric KMS key.
        echo "\n";
        echo "14. Tag the symmetric KMS key.\n";
        echo "By using tags, you can improve the overall management, security,
 and governance of your KMS keys, making it easier to organize, track, and
 control access to your encrypted data within your AWS environment.\n";
        echo "Let's tag our symmetric key as Environment->Production\n";
        $this->pressEnter();
        $this->kmsService->tagResource($key['KeyId'], [
            [
                'TagKey' => "Environment",
                'TagValue' => "Production",
            ],
        ]);
        echo "The key was successfully tagged!\n";
        $this->pressEnter();

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
 specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
 changes to PendingDeletion and the key can't be used in any cryptographic
 operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n
\n";

        echo "Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted, all data that was encrypted under the KMS
 key is unrecoverable.\n\n";

        $cleanUp = testable_readline("Would you like to delete the resources
 created during this scenario, including the keys? (y/n): ");
        if($cleanUp == "Y" || $cleanUp == "y"){
            $this->cleanUp();
        }

        echo
 "-------------------------------------------------------------------------------
\n";
        echo "This concludes the AWS Key Management SDK Basics scenario\n";
```

```
        echo
 "--------------------------------------------------------------------------------
\n";



namespace Kms;

use Aws\Kms\Exception\KmsException;
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

    /***
     * @param KmsClient|null $client
     * @param bool $verbose
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }


    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
 $description = "Created by the SDK for PHP")
    {
```

```
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
 more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
 = "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is
not enabled.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
```

```
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name
is not valid.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
```

```
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token
is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
```

```
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
```

```
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity
 or resource could not be found.\n";
            }
            throw $caught;
        }
    }


    /***
     * @param string $keyId
     * @return Result
     */
    public function getKeyPolicy(string $keyId)
    {
        try {
            return $this->client->getKeyPolicy([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $rotationPeriodInDays
     * @return void
     */
    public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
365)
    {
        try{
            $this->client->enableKeyRotation([
                'KeyId' => $keyId,
                'RotationPeriodInDays' => $rotationPeriodInDays,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature,
 string $signingAlgorithm)
    {
        try {
            $result = $this->client->verify([
                'KeyId' => $keyId,
                'Message' => $message,
                'Signature' => $signature,
                'SigningAlgorithm' => $signingAlgorithm,
            ]);
```

```
            return $result['SignatureValid'];
        }catch(KmsException $caught){
            echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KMSScenario:
    """Runs an interactive scenario that shows how to get started with KMS."""

    def __init__(
        self,
        key_manager: KeyManager,
        key_encryption: KeyEncrypt,
        alias_manager: AliasManager,
        grant_manager: GrantManager,
        key_policy: KeyPolicy,
    ):
        self.key_manager = key_manager
        self.key_encryption = key_encryption
        self.alias_manager = alias_manager
        self.grant_manager = grant_manager
        self.key_policy = key_policy
        self.key_id = ""
        self.alias_name = ""
        self.asymmetric_key_id = ""

    def kms_scenario(self):
        key_description = "Created by the AWS KMS API"

        print(DASHES)
        print(
            """
Welcome to the AWS Key Management SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management using the AWS
 SDK for Python (Boto3).
The AWS Key Management Service (KMS) is a secure and highly available service
 that allows you to create
```

```
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys,
 making it easier to meet your
data protection and regulatory compliance requirements.

This Basics scenario creates two key types:

- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...
        """
        )
        q.ask("Press Enter to continue...")

        print(DASHES)
        print(f"1. Create a symmetric KMS key\n")
        print(
            f"First, the program will creates a symmetric KMS key that you can
 used to encrypt and decrypt data."
        )
        q.ask("Press Enter to continue...")
        self.key_id = self.key_manager.create_key(key_description)["KeyId"]
        print(f"A symmetric key was successfully created {self.key_id}.")
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(
            """
2. Enable a KMS key

By default, when the SDK creates an AWS key, it is enabled. The next bit of code
 checks to
determine if the key is enabled.
        """
        )
        q.ask("Press Enter to continue...")
        is_enabled = self.is_key_enabled(self.key_id)
        print(f"Is the key enabled? {is_enabled}")
        if not is_enabled:
            self.key_manager.enable_key(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"3. Encrypt data using the symmetric KMS key")
```

```
        plain_text = "Hello, AWS KMS!"
        print(
            f"""
One of the main uses of symmetric keys is to encrypt and decrypt data.
Next, the code encrypts the string "{plain_text}" with the SYMMETRIC_DEFAULT
 encryption algorithm.
        """
        )
        q.ask("Press Enter to continue...")
        encrypted_text = self.key_encryption.encrypt(self.key_id, plain_text)
        print(DASHES)
        print(f"4. Create an alias")
        print(
            """
Now, the program will create an alias for the KMS key. An alias is a friendly
 name that you
can associate with a KMS key. The alias name should be prefixed with 'alias/'.
        """
        )
        alias_name = q.ask("Enter an alias name: ", q.non_empty)
        self.alias_manager.create_alias(self.key_id, alias_name)
        print(f"{alias_name} was successfully created.")
        self.alias_name = alias_name
        print(DASHES)
        print(f"5. List all of your aliases")
        q.ask("Press Enter to continue...")
        self.alias_manager.list_aliases(10)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"6. Enable automatic rotation of the KMS key")
        print(
            """

By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
 from the enable date and every year
thereafter.
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.enable_key_rotation(self.key_id)
        print(DASHES)
        print(f"Key rotation has been enabled for key with id {self.key_id}")
        print(
```

```
            """
7. Create a grant

A grant is a policy instrument that allows Amazon Web Services principals to use
 KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
 grants.
When authorizing access to a KMS key, grants are considered along with key
 policies and IAM policies.
        """
        )
        print(
            """
To create a grant you must specify a account_id. To specify the grantee
 account_id, use the Amazon Resource Name (ARN)
of an AWS account_id. Valid principals include AWS accounts, IAM users, IAM
 roles, federated users,
and assumed role users.
        """
        )
        account_id = q.ask(
            "Enter an account_id, or press enter to skip creating a grant... "
        )
        grant = None
        if account_id != "":
            grant = self.grant_manager.create_grant(
                self.key_id,
                account_id,
                [
                    "Encrypt",
                    "Decrypt",
                    "DescribeKey",
                ],
            )
            print(f"Grant created successfully with ID: {grant['GrantId']}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(DASHES)
        print(f"8. List grants for the KMS key")
        q.ask("Press Enter to continue...")
        self.grant_manager.list_grants(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
```

```
        print(f"9. Revoke the grant")
        print(
            """
The revocation of a grant immediately removes the permissions and access that the
 grant had provided.
This means that any account_id (user, role, or service) that was granted access
 to perform specific
KMS operations on a KMS key will no longer be able to perform those operations.
        """
        )
        q.ask("Press Enter to continue...")

        if grant is not None:
            self.grant_manager.revoke_grant(self.key_id, grant["GrantId"])
            print(f"Grant ID: {grant['GrantId']} was successfully revoked!")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"10. Decrypt the data\n")
        print(
            """
Lets decrypt the data that was encrypted in an early step.
The code uses the same key to decrypt the string that we encrypted earlier in the
 program.
        """
        )
        q.ask("Press Enter to continue...")
        decrypted_data = self.key_encryption.decrypt(self.key_id, encrypted_text)
        print(f"Data decrypted successfully for key ID: {self.key_id}")
        print(f"Decrypted data: {decrypted_data}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"11. Replace a key policy\n")
        print(
            """
A key policy is a resource policy for a KMS key. Key policies are the primary way
 to control
access to KMS keys. Every KMS key must have exactly one key policy. The
 statements in the key policy
determine who has permission to use the KMS key and how they can use it.
You can also use IAM policies and grants to control access to the KMS key, but
 every KMS key
must have a key policy.
```

```
By default, when you create a key by using the SDK, a policy is created that
gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.
{
"Version":"2012-10-17",
"Statement": [{
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::0000000000:root"},
"Action": "kms:*",
"Resource": "*"
}]
}
        """
        )
        account_id = q.ask("Enter your account ID or press enter to skip: ")
        if account_id != "":
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": f"arn:aws:iam::{account_id}:root"},
                        "Action": "kms:*",
                        "Resource": "*",
                    }
                ],
            }

            self.key_policy.set_new_policy(self.key_id, policy)
            print("Key policy replacement succeeded.")
            q.ask("Press Enter to continue...")
        else:
            print("Skipping replacing the key policy.")

        print(DASHES)
        print(f"12. Get the key policy\n")
        print(
            f"The next bit of code that runs gets the key policy to make sure it
 exists."
        )
        q.ask("Press Enter to continue...")
        policy = self.key_policy.get_policy(self.key_id)
```

```
        print(f"The key policy is: {policy}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"13. Create an asymmetric KMS key and sign your data\n")
        print(
            """
        Signing your data with an AWS key can provide several benefits that make
it an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage
the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
        """
        )
        q.ask("Press Enter to continue...")
        print(f"Sign and verify data operation succeeded.")
        self.asymmetric_key_id = self.key_manager.create_asymmetric_key()
        message = "Here is the message that will be digitally signed"
        signature = self.key_encryption.sign(self.asymmetric_key_id, message)
        if self.key_encryption.verify(self.asymmetric_key_id, message,
signature):
            print("Signature verification succeeded.")
        else:
            print("Signature verification failed.")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"14. Tag your symmetric KMS Key\n")
        print(
            """
        By using tags, you can improve the overall management, security, and
governance of your
        KMS keys, making it easier to organize, track, and control access to your
encrypted data within
        your AWS environment
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.tag_resource(self.key_id, "Environment", "Production")
        self.clean_up()
```

```
    def is_key_enabled(self, key_id: str) -> bool:
        """
        Check if the key is enabled or not.

        :param key_id: The key to check.
        :return: True if the key is enabled, otherwise False.
        """
        response = self.key_manager.describe_key(key_id)
        return response["Enabled"] is True

    def clean_up(self):
        """
        Delete resources created by this scenario.
        """
        if self.alias_name != "":
            print(f"Deleting the alias {self.alias_name}.")
            self.alias_manager.delete_alias(self.alias_name)
        window = 7  # The window in days for a scheduled deletion.
        if self.key_id != "":
            print(
                """
Warning:
Deleting a KMS key is a destructive and potentially dangerous operation. When a
 KMS key is deleted,
all data that was encrypted under the KMS key is unrecoverable.
                """
            )
            if q.ask(
                f"Do you want to delete the key with ID {self.key_id} (y/n)?",
                q.is_yesno,
            ):
                print(
                    f"The key {self.key_id} will be deleted with a window of
 {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.key_id, window)
                self.key_id = ""

        if self.asymmetric_key_id != "":
            if q.ask(
                f"Do you want to delete the asymmetric key with ID
 {self.asymmetric_key_id} (y/n)?",
                q.is_yesno,
```

```
            ):
                print(
                    f"The key {self.asymmetric_key_id} will be deleted with a
 window of {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.asymmetric_key_id, window)
                self.asymmetric_key_id = ""


if __name__ == "__main__":
    kms_scenario = None
    try:
        kms_client = boto3.client("kms")
        a_key_manager = KeyManager(kms_client)
        a_key_encrypt = KeyEncrypt(kms_client)
        an_alias_manager = AliasManager(kms_client)
        a_grant_manager = GrantManager(kms_client)
        a_key_policy = KeyPolicy(kms_client)
        kms_scenario = KMSScenario(
            key_manager=a_key_manager,
            key_encryption=a_key_encrypt,
            alias_manager=an_alias_manager,
            grant_manager=a_grant_manager,
            key_policy=a_key_policy,
        )
        kms_scenario.kms_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo!")
        if kms_scenario is not None:
            kms_scenario.clean_up()
```

Classe e métodos do wrapper para gerenciamento de chaves do KMS.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
```

```
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
```

```
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_asymmetric_key(self) -> str:
        """
        Creates an asymmetric key in AWS KMS for signing messages.

        :return: The ID of the created key.
        """
        try:
            key = self.kms_client.create_key(
                KeySpec="RSA_2048", KeyUsage="SIGN_VERIFY", Origin="AWS_KMS"
            )["KeyMetadata"]
            self.created_keys.append(key)
            return key["KeyId"]
        except ClientError as err:
            logger.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
```

```
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
```

```
            raise
```

Classe e métodos do wrapper para aliases de chaves do KMS.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
```

```
                raise


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)?
"
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
```

```
                err.response["Error"]["Message"],
            )
            raise
```

Classe e métodos do wrapper para criptografia de chaves do KMS.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
 Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
 {response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
```

```
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id,
CiphertextBlob=cipher_text)[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
```

```
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Classe e métodos do wrapper para concessões de chaves do KMS.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

```
    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

Classe e métodos do wrapper para políticas de chaves do KMS.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_new_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
 existing
        policy, so care is taken to add a statement to the existing list of
 statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: A new key policy. The key policy must allow the calling
 principal to make a subsequent
                       PutKeyPolicy request on the KMS key. This reduces the risk
 that the KMS key becomes unmanageable
        """

        try:
            self.kms_client.put_key_policy(KeyId=key_id,
 Policy=json.dumps(policy))
        except ClientError as err:
            logger.error(
                "Couldn't set policy for key %s. Here's why %s",
                key_id,
                err.response["Error"]["Message"],
```

```
            )
            raise



    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey

- EnableKey
- Encrypt
- GetKeyPolicy
- ListAliases
- ListGrants
- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

Ações para AWS KMS usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS KMS ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a AWS KMS API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para AWS KMS usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Key Management Service.

Exemplos

- Use CreateAliascom um AWS SDK ou CLI
- Use CreateGrantcom um AWS SDK ou CLI
- Use CreateKeycom um AWS SDK ou CLI
- Use Decrypt com um AWS SDK ou CLI
- Use DeleteAliascom um AWS SDK ou CLI
- Use DescribeKeycom um AWS SDK ou CLI
- Use DisableKeycom um AWS SDK ou CLI
- Use EnableKeycom um AWS SDK ou CLI

- Use EnableKeyRotationcom um AWS SDK ou CLI
- Use Encrypt com um AWS SDK ou CLI
- Use GenerateDataKeycom um AWS SDK ou CLI
- Use GenerateDataKeyWithoutPlaintextcom um AWS SDK ou CLI
- Use GenerateRandomcom um AWS SDK ou CLI
- Use GetKeyPolicycom um AWS SDK ou CLI
- Use ListAliasescom um AWS SDK ou CLI
- Use ListGrantscom um AWS SDK ou CLI
- Use ListKeyPoliciescom um AWS SDK ou CLI
- Use ListKeyscom um AWS SDK ou CLI
- Use PutKeyPolicycom um AWS SDK ou CLI
- Use ReEncryptcom um AWS SDK ou CLI
- Use RetireGrantcom um AWS SDK ou CLI
- Use RevokeGrantcom um AWS SDK ou CLI
- Use ScheduleKeyDeletioncom um AWS SDK ou CLI
- Use o Sign com um AWS SDK ou CLI
- Use TagResourcecom um AWS SDK ou CLI
- Use UpdateAliascom um AWS SDK ou CLI
- Use Verify com um AWS SDK ou CLI

Use **`CreateAlias`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateAlias`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Creates an alias for an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class CreateAlias
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The alias name must start with alias/ and can be
        // up to 256 alphanumeric characters long.
        var aliasName = "alias/ExampleAlias";

        // The value supplied as the TargetKeyId can be either
        // the key ID or key Amazon Resource Name (ARN) of the
        // AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new CreateAliasRequest
        {
            AliasName = aliasName,
            TargetKeyId = keyId,
        };

        var response = await client.CreateAliasAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                Console.WriteLine($"Alias, {aliasName}, successfully created.");
            }
            else
            {
                Console.WriteLine($"Could not create alias.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateAliasa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar um alias para uma chave do KMS

O comando `create-alias` a seguir cria um alias com o nome `example-alias` para a chave do KMS identificada pelo ID de chave `1234abcd-12ab-34cd-56ef-1234567890ab`.

Os nomes de alias devem começar com `alias/`. Não use nomes de alias que comecem com`alias/aws`; eles são reservados para uso por AWS.

```
aws kms create-alias \
    --alias-name alias/example-alias \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Esse comando não retorna resultados. Use o `list-aliases` comando para visualizar o novo alias.

Para obter mais informações, consulte Using aliases no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte CreateAliasem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
               if (exception instanceof ResourceExistsException) {
                   logger.info("Alias [{}] already exists. Moving on...",
aliasName);
               } else if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
               }
           }
```

```
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte CreateAliasa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- Para obter detalhes da API, consulte a CreateAliasreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name
is not valid.";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte CreateAliasa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
```

```
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a CreateAliasReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->createalias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_key_id
        ).
        MESSAGE 'Alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsalreadyexistsex.
        MESSAGE 'Alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidaliasnameex.
        MESSAGE 'Invalid alias name.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAliasreferência da API AWS SDK for SAP ABAP.

Use **`CreateGrant`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateGrant`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();

            // The identity that is given permission to perform the operations
            // specified in the grant.
            var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

            // The identifier of the AWS KMS key to which the grant applies. You
            // can use the key ID or the Amazon Resource Name (ARN) of the KMS
key.
            var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";

            var request = new CreateGrantRequest
            {
                GranteePrincipal = grantee,
                KeyId = keyId,

                // A list of operations that the grant allows.
```

```
                Operations = new List<string>
                {
                    "Encrypt",
                    "Decrypt",
                },
            };

            var response = await client.CreateGrantAsync(request);

            string grantId = response.GrantId; // The unique identifier of the
 grant.
            string grantToken = response.GrantToken; // The grant token.

            Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
        }
    }
```

- Para obter detalhes da API, consulte CreateGrant a Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar uma concessão

O exemplo de `create-grant` a seguir cria uma concessão que permite que o usuário `exampleUser` use o comando `decrypt` na chave do KMS `1234abcd-12ab-34cd-56ef-1234567890ab` de exemplo. A entidade principal descontinuada é o perfil `adminRole`. A concessão usa a restrição de concessão `EncryptionContextSubset` para conceder essa permissão apenas quando o contexto de criptografia na solicitação `decrypt` incluir o par de chave/valor `"Department": "IT"`.

```
aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::123456789012:user/exampleUser \
    --operations Decrypt \
    --constraints EncryptionContextSubset={Department=IT} \
    --retiring-principal arn:aws:iam::123456789012:role/adminRole
```

Saída:

```
{
    "GrantId":
 "1a2b3c4d2f5e69f440bae30eaec9570bb1fb7358824f9ddfa1aa5a0dab1a59b2",
    "GrantToken": "<grant token here>"
}
```

Use o `list-grants` comando para visualizar informações detalhadas sobre a concessão.

Para obter mais informações, consulte Concessões no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte CreateGrantem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
    *
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID
of the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
    */
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
       List<GrantOperation> grantPermissions = List.of(
           GrantOperation.ENCRYPT,
           GrantOperation.DECRYPT,
```

```
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }
```

- Para obter detalhes da API, consulte CreateGrant a Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- Para obter detalhes da API, consulte a CreateGrantreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /***
    * @param string $keyId
    * @param string $granteePrincipal
    * @param array $operations
    * @param array $grantTokens
    * @return Result
    */
   public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
   {
       $args = [
           'KeyId' => $keyId,
           'GranteePrincipal' => $granteePrincipal,
           'Operations' => $operations,
       ];
       if($grantTokens){
           $args['GrantTokens'] = $grantTokens;
       }
       try{
           return $this->client->createGrant($args);
       }catch(KmsException $caught){
           if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
               echo "The request was rejected because the specified grant token
is not valid.\n";
           }
           throw $caught;
       }
   }
```

- Para obter detalhes da API, consulte CreateGrant a Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
```

```
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateGrantReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grantee_principal = 'arn:aws:iam::123456789012:role/my-role'
        " it_operations contains 'Encrypt', 'Decrypt', 'GenerateDataKey'
        oo_result = lo_kms->creategrant(
          iv_keyid = iv_key_id
          iv_granteeprincipal = iv_grantee_principal
          it_operations = it_operations
        ).
        MESSAGE 'Grant created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
```

```
      MESSAGE 'Key not found.' TYPE 'E'.
    CATCH /aws1/cx_kmskmsinternalex.
      MESSAGE 'An internal error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateGrantreferência da API AWS SDK for SAP ABAP.

Use **`CreateKey`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Trabalhar com a criptografia de tabelas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to create a new AWS Key Management Service (AWS KMS)
/// key.
/// </summary>
public class CreateKey
{
```

```
        public static async Task Main()
        {
            // Note that if you need to create a Key in an AWS Region
            // other than the Region defined for the default user, you need to
            // pass the Region to the client constructor.
            var client = new AmazonKeyManagementServiceClient();

            // The call to CreateKeyAsync will create a symmetrical AWS KMS
            // key. For more information about symmetrical and asymmetrical
            // keys, see:
            //
            // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
            var response = await client.CreateKeyAsync(new CreateKeyRequest());

            // The KeyMetadata object contains information about the new AWS KMS
 key.
            KeyMetadata keyMetadata = response.KeyMetadata;

            if (keyMetadata is not null)
            {
                Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
 created.");
            }
            else
            {
                Console.WriteLine("Could not create KMS Key.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateKeya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: Para criar uma chave KMS gerenciada pelo cliente no AWS KMS

O exemplo de `create-key` a seguir cria uma chave do KMS de criptografia simétrica.

Não é necessário especificar parâmetros para criar a chave básica do KMS (uma chave de criptografia simétrica). Os valores padrão desses parâmetros criam uma chave de criptografia simétrica.

Como esse comando não especifica uma política de chave, a chave do KMS adota a política de chave padrão para chaves do KMS criadas programaticamente. Use o comando `get-key-policy` para visualizar a política de chave. Use o comando `put-key-policy` para alterar a política de chave.

```
aws kms create-key
```

O comando `create-key` retorna os metadados da chave, incluindo o ID da chave e o ARN da nova chave do KMS. Você pode usar esses valores para identificar a chave KMS em outras operações do AWS KMS. A saída não inclui as tags. Use `list-resource-tags command` para visualizar as tags de uma chave do KMS.

Saída:

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2017-07-05T14:04:55-07:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_KMS"
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

Observação: o comando `create-key` não permite especificar um alias. Use o comando `create-alias` para especificar um alias para a nova chave do KMS.

Para obter mais informações, consulte Creating keys no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 2: como criar uma chave RSA assimétrica do KMS para criptografia e decodificação

O exemplo de `create-key` a seguir cria uma chave do KMS que contém um par de chaves RSA assimétrico para criptografia e descriptografia. A especificação da chave e o uso da chave não podem ser alterados depois que a chave é criada.

```
aws kms create-key \
   --key-spec RSA_4096 \
   --key-usage ENCRYPT_DECRYPT
```

Saída:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2021-04-05T14:04:55-07:00",
        "CustomerMasterKeySpec": "RSA_4096",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "RSAES_OAEP_SHA_1",
            "RSAES_OAEP_SHA_256"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "RSA_4096",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_KMS"
    }
}
```

Para obter mais informações, consulte Chaves assimétricas no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 3: como criar uma chave do KMS de curva elíptica assimétrica para assinatura e verificação

Como criar uma chave do KMS assimétrica que contém um par de chaves com curva elíptica (ECC) assimétrica para assinatura e verificação. O parâmetro `--key-usage` é obrigatório, embora `SIGN_VERIFY` seja o único valor válido para chaves ECC do KMS. A especificação da chave e o uso da chave não podem ser alterados depois que a chave é criada.

```
aws kms create-key \
    --key-spec ECC_NIST_P521 \
    --key-usage SIGN_VERIFY
```

Saída:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "ECC_NIST_P521",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "ECC_NIST_P521",
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "MultiRegion": false,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ECDSA_SHA_512"
        ]
    }
}
```

Para obter mais informações, consulte Chaves assimétricas no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 4: Para criar uma chave ML-DSA KMS assimétrica para assinatura e verificação

Este exemplo cria uma chave de algoritmo de assinatura digital (ML-DSA) de estrutura modular para assinatura e verificação. O parâmetro de uso da chave é obrigatório, embora `SIGN_VERIFY` seja o único valor válido para ML-DSA chaves.

```
aws kms create-key \
    --key-spec ML_DSA_65 \
    --key-usage SIGN_VERIFY
```

Saída:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "ML_DSA_65",
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "MultiRegion": false,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ML_DSA_SHAKE_256"
        ]
    }
}
```

Para obter mais informações, consulte Chaves assimétricas no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 5: criar uma chave HMAC do KMS

O exemplo de `create-key` a seguir cria uma chave do KMS com HMAC de 384 bits. O valor `GENERATE_VERIFY_MAC` é obrigatório para o parâmetro `--key-usage`, mesmo que seja o único valor válido para chaves do KMS com HMAC.

```
aws kms create-key \
    --key-spec HMAC_384 \
    --key-usage GENERATE_VERIFY_MAC
```

Saída:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2022-04-05T14:04:55-07:00",
        "CustomerMasterKeySpec": "HMAC_384",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "HMAC_384",
        "KeyState": "Enabled",
        "KeyUsage": "GENERATE_VERIFY_MAC",
        "MacAlgorithms": [
            "HMAC_SHA_384"
        ],
        "MultiRegion": false,
        "Origin": "AWS_KMS"
    }
}
```

Para obter mais informações, consulte Chaves HMAC no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 6: criar uma chave do KMS primária multirregional

O exemplo de `create-key` a seguir cria uma chave primária de criptografia simétrica multirregional. Como os valores padrão de todos os parâmetros criam uma chave de criptografia simétrica, somente o parâmetro `--multi-region` é necessário para a chave do KMS. Na AWS CLI, para indicar que um parâmetro booleano é verdadeiro, basta especificar o nome do parâmetro.

```
aws kms create-key \
    --multi-region
```

Saída:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2021-09-02T016:15:21-09:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
          "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "mrk-1234abcd12ab34cd56ef12345678990ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": true,
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
                "Region": "us-west-2"
            },
            "ReplicaKeys": []
        },
        "Origin": "AWS_KMS"
    }
}
```

Para obter mais informações, consulte Chaves assimétricas no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 7: criar uma chave do KMS para material de chave importada

O exemplo de `create-key` a seguir cria uma chave do KMS sem material de chave. Quando a operação for concluída, você poderá importar seu próprio material de chave na chave do KMS. Defina o parâmetro `--origin` como `EXTERNAL` para criar a chave do KMS.

```
aws kms create-key \
    --origin EXTERNAL
```

Saída:

```
{
     "KeyMetadata": {
         "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
         "AWSAccountId": "111122223333",
         "CreationDate": "2019-12-02T07:48:55-07:00",
         "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
         "Description": "",
         "Enabled": false,
         "EncryptionAlgorithms": [
             "SYMMETRIC_DEFAULT"
         ],
         "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
         "KeyManager": "CUSTOMER",
         "KeySpec": "SYMMETRIC_DEFAULT",
         "KeyState": "PendingImport",
         "KeyUsage": "ENCRYPT_DECRYPT",
         "MultiRegion": false,
         "Origin": "EXTERNAL"
     }
 }
```

Para obter mais informações, consulte Importação de material de chaves em chaves AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 6: Para criar uma chave KMS em um armazenamento de chaves do AWS CloudHSM

O `create-key` exemplo a seguir cria uma chave KMS no armazenamento de chaves especificado do AWS CloudHSM. A operação cria a chave KMS e seus metadados no AWS KMS e cria o material da chave no cluster do CloudHSM associado ao AWS armazenamento de chaves personalizadas. Os parâmetros `--custom-key-store-id` e `--origin` são obrigatórios.

```
aws kms create-key \
    --origin AWS_CLOUDHSM \
    --custom-key-store-id cks-1234567890abcdef0
```

Saída:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CloudHsmClusterId": "cluster-1a23b4cdefg",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "CustomKeyStoreId": "cks-1234567890abcdef0",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_CLOUDHSM"
    }
}
```

Para obter mais informações, consulte AWS CloudHSM key stores no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 8: criar uma chave do KMS em um repositório de chaves externo

O exemplo de `create-key` a seguir cria uma chave do KMS no repositório de chaves externo especificado. Os parâmetros `--custom-key-store-id`, `--origin` e `--xks-key-id` são obrigatórios neste comando.

O parâmetro `--xks-key-id` especifica o ID de uma chave de criptografia simétrica existente no gerenciador de chaves externo. Essa chave serve como o material de chave externo para a chave do KMS. O valor do parâmetro `--origin` deve ser `EXTERNAL_KEY_STORE`. O parâmetro `custom-key-store-id` deve identificar um repositório de chaves externo conectado ao proxy do repositório de chaves externo.

```
aws kms create-key \
```

```
    --origin EXTERNAL_KEY_STORE \
    --custom-key-store-id cks-9876543210fedcba9 \
    --xks-key-id bb8562717f809024
```

Saída:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2022-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "CustomKeyStoreId": "cks-9876543210fedcba9",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "EXTERNAL_KEY_STORE",
        "XksKeyConfiguration": {
            "Id": "bb8562717f809024"
        }
    }
}
```

Para obter mais informações, consulte External key stores no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte CreateKeyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }
```

- Para obter detalhes da API, consulte CreateKeya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- Para obter detalhes da API, consulte a CreateKeyreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
```

```
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
 $description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
 more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte CreateKeya Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
```

```
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateKeyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: 'CreatedBy',
                               tag_value: 'ExampleUser'
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

- Para obter detalhes da API, consulte CreateKeya Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_key(client: &Client) -> Result<(), Error> {
    let resp = client.create_key().send().await?;

    let id = resp.key_metadata.as_ref().unwrap().key_id();

    println!("Key: {}", id);

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateKeyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_description = 'Created by the AWS SDK for SAP ABAP'
        oo_result = lo_kms->createkey( iv_description = iv_description ).
        MESSAGE 'KMS key created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
      CATCH /aws1/cx_kmslimitexceededex.
```

```
        MESSAGE 'Limit exceeded for KMS resources.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateKeyreferência da API AWS SDK for SAP ABAP.

Use **`Decrypt`** **`com`** um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Decrypt`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Exemplo 1: como descriptografar uma mensagem criptografada com uma chave simétrica do KMS (Linux e macOS)

O exemplo de `decrypt` comando a seguir demonstra a forma recomendada de descriptografar dados com a CLI. AWS Esta versão mostra como descriptografar dados com uma chave simétrica do KMS.

Forneça o texto cifrado em um arquivo. No valor do parâmetro `--ciphertext-blob`, use o prefixo `fileb://`, que instrui a CLI a ler os dados de um arquivo binário. Se o arquivo não estiver no diretório atual, digite o caminho completo para o arquivo. Para obter mais informações sobre a leitura dos valores dos parâmetros da AWS CLI de um arquivo, consulte Carregando parâmetros da AWS CLI de um arquivo < https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-parameters-file.html > no Guia do usuário da interface de linha de AWS comando e as melhores práticas para parâmetros de arquivos locais< https://aws.amazon.com/blogs/developer/best-practices-for-local-file-parameters/ > no blog da ferramenta de linha de AWS comando .Especifique a chave KMS para descriptografar o texto `--key-id` cifrado.O parâmetro não é necessário ao descriptografar com uma chave KMS simétrica. AWS O KMS pode obter o ID da chave KMS que foi usada para criptografar os dados dos metadados no texto cifrado. Porém, sempre é uma prática recomendada

especificar a chave do KMS que você está usando. Esta prática garante que você use a chave do KMS desejada e impede que você descriptografe um texto cifrado acidentalmente usando uma chave do KMS em que você não confia. Solicite a saída de texto simples como um valor de texto. O parâmetro `--query` instrui a CLI obter somente o valor do campo `Plaintext` da saída. O `--output` parâmetro retorna a saída como texto. Base64-decode o texto simples e salve-o em um arquivo. O exemplo a seguir canaliza (|) o valor do `Plaintext` parâmetro para o utilitário Base64, que o decodifica. Em seguida, ele redireciona (>) a saída decodificada para o arquivo `ExamplePlaintext`.

Antes de executar esse comando, substitua o ID de chave de exemplo por um ID de chave válido da sua AWS conta.

```
aws kms decrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --output text \
    --query Plaintext | base64 \
    --decode > ExamplePlaintextFile
```

Este comando não produz saída. A saída do comando `decrypt` é decodificada em base64 e salva em um arquivo.

Para obter mais informações, consulte Decrypt na Referência da API do AWS Key Management Service.

Exemplo 2: como descriptografar uma mensagem criptografada com uma chave simétrica do KMS (prompt de comando do Windows)

O exemplo a seguir é igual ao anterior, exceto que ele usa o `certutil` utilitário para Base64-decode os dados de texto simples. Esse procedimento requer dois comandos, conforme mostrado nos exemplos a seguir.

Antes de executar esse comando, substitua o ID de chave de exemplo por um ID de chave válido da sua AWS conta.

```
aws kms decrypt ^
    --ciphertext-blob fileb://ExampleEncryptedFile ^
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
    --output text ^
    --query Plaintext > ExamplePlaintextFile.base64
```

Execute o comando `certutil`.

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

Saída:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Para obter mais informações, consulte Decrypt na Referência da API do AWS Key Management Service.

Exemplo 3: como descriptografar uma mensagem criptografada com uma chave assimétrica do KMS (Linux e macOS)

O exemplo de comando `decrypt` a seguir mostra como descriptografar dados criptografados sob uma chave RSA assimétrica do KMS.

Ao usar uma chave assimétrica do KMS, o parâmetro `encryption-algorithm`, que especifica o algoritmo usado para criptografar o texto simples, é necessário.

Antes de executar esse comando, substitua o ID de chave de exemplo por um ID de chave válido da sua AWS conta.

```
aws kms decrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --encryption-algorithm RSAES_OAEP_SHA_256 \
    --output text \
    --query Plaintext | base64 \
    --decode > ExamplePlaintextFile
```

Este comando não produz saída. A saída do comando `decrypt` é decodificada em base64 e salva em um arquivo.

Para obter mais informações, consulte Chaves assimétricas no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

- Consulte detalhes da API em Decrypt na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously decrypts the given encrypted data using the specified key
ID.
    *
    * @param encryptedData The encrypted data to be decrypted.
    * @param keyId The ID of the key to be used for decryption.
    * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
    *         If an error occurs during the decryption process, the
CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
String.
    */
   public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
       DecryptRequest decryptRequest = DecryptRequest.builder()
           .ciphertextBlob(encryptedData)
           .keyId(keyId)
           .build();

       CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
       responseFuture.whenComplete((decryptResponse, exception) -> {
           if (exception == null) {
               logger.info("Data decrypted successfully for key ID: " + keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
```

```
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
 decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
```

```
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
 = "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
```

```
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
```

```
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id,
 CiphertextBlob=cipher_text)[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Decrypted blob

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593
blob_packed = [blob].pack('H*')
```

```
client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts 'Raw text: '
puts resp.plaintext
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(),
 Error> {
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(filename)
        .map(|input| {
            base64::decode(input).expect("Input file does not contain valid base
 64 characters.")
        })
        .map(Blob::new);

    let resp = client
        .decrypt()
        .key_id(key)
        .ciphertext_blob(data.unwrap())
        .send()
        .await?;

    let inner = resp.plaintext.unwrap();
    let bytes = inner.as_ref();
```

```
    let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to
 UTF-8");

    println!();
    println!("Decoded string:");
    println!("{}", s);

    Ok(())
}
```

- Consulte detalhes da API em Descriptografar na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->decrypt(
          iv_keyid = iv_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Text decrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a referência da API Decrypt in AWS SDK for SAP ABAP.

Use **`DeleteAlias`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteAlias`.

CLI

AWS CLI

Para excluir um alias do AWS KMS

O exemplo de `delete-alias` a seguir exclui o alias `alias/example-alias`. O nome de alias deve começar com alias/.

```
aws kms delete-alias \
    --alias-name alias/example-alias
```

Este comando não produz saída. Use o `list-aliases` comando para localizar o alias.

Para obter mais informações, consulte Deleting an alias no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte DeleteAliasem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
```

```
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
            });
    }
```

- Para obter detalhes da API, consulte DeleteAliasa Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
```

```
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DeleteAliasa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_alias(self, alias: str) -> None:
```

```
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteAliasReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_alias_name = 'alias/my-key-alias'
    lo_kms->deletealias( iv_aliasname = iv_alias_name ).
    MESSAGE 'Alias deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_kmsnotfoundexception.
    MESSAGE 'Alias not found.' TYPE 'E'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAliasreferência da API AWS SDK for SAP ABAP.

Use **`DescribeKey`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Retrieve information about an AWS Key Management Service (AWS KMS) key.
/// You can supply either the key Id or the key Amazon Resource Name (ARN)
/// to the DescribeKeyRequest KeyId property.
/// </summary>
public class DescribeKey
{
    public static async Task Main()
    {
        var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
        var request = new DescribeKeyRequest
        {
            KeyId = keyId,
        };
```

```
            var client = new AmazonKeyManagementServiceClient();

            var response = await client.DescribeKeyAsync(request);
            var metadata = response.KeyMetadata;

            Console.WriteLine($"{metadata.KeyId} created on:
 {metadata.CreationDate}");
            Console.WriteLine($"State: {metadata.KeyState}");
            Console.WriteLine($"{metadata.Description}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeKeya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: como localizar informações detalhadas sobre uma chave do KMS

O `describe-key` exemplo a seguir obtém informações detalhadas sobre a chave AWS gerenciada para o Amazon S3 na conta e região do exemplo. Você pode usar esse comando para encontrar detalhes sobre chaves AWS gerenciadas e chaves gerenciadas pelo cliente.

Use o parâmetro `key-id` para especificar a chave do KMS. Este exemplo usa um valor de nome de alias, mas você pode usar um ID de chave, o ARN da chave, o nome do alias ou o ARN do alias neste comando.

```
aws kms describe-key \
    --key-id alias/aws/s3
```

Saída:

```
{
    "KeyMetadata": {
        "AWSAccountId": "846764612917",
        "KeyId": "b8a9477d-836c-491f-857e-07937918959b",
```

```
        "Arn": "arn:aws:kms:us-west-2:846764612917:key/
b8a9477d-836c-491f-857e-07937918959b",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CreationDate": 2017-06-30T21:44:32.140000+00:00,
        "Enabled": true,
        "Description": "Default KMS key that protects my S3 objects when no other
 key is defined",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "AWS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

Para obter mais informações, consulte Viewing keys no Guia do desenvolvedor do AWS Key Management Service.

Exemplo 2: como obter detalhes sobre uma chave RSA assimétrica do KMS

O exemplo de `describe-key` a seguir obtém informações detalhadas sobre uma chave RSA assimétrica do KMS usada para assinatura e verificação.

```
aws kms describe-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Saída:

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2019-12-02T19:47:14.861000+00:00",
        "CustomerMasterKeySpec": "RSA_2048",
        "Enabled": false,
        "Description": "",
```

```
        "KeyState": "Disabled",
        "Origin": "AWS_KMS",
        "MultiRegion": false,
        "KeyManager": "CUSTOMER",
        "KeySpec": "RSA_2048",
        "KeyUsage": "SIGN_VERIFY",
        "SigningAlgorithms": [
            "RSASSA_PKCS1_V1_5_SHA_256",
            "RSASSA_PKCS1_V1_5_SHA_384",
            "RSASSA_PKCS1_V1_5_SHA_512",
            "RSASSA_PSS_SHA_256",
            "RSASSA_PSS_SHA_384",
            "RSASSA_PSS_SHA_512"
        ]
    }
}
```

Exemplo 3: como obter detalhes sobre uma chave de réplica multirregional

O exemplo de `describe-key` a seguir os obtém metadados de uma chave de réplica multirregional. Essa chave multirregional é uma chave de criptografia simétrica. A saída de um comando `describe-key` para qualquer chave multirregional retorna informações sobre a chave primária e todas as suas réplicas.

```
aws kms describe-key \
    --key-id arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab
```

Saída:

```
{
    "KeyMetadata": {
        "MultiRegion": true,
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "CreationDate": "2021-06-28T21:09:16.114000+00:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "Description": "",
        "Enabled": true,
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
```

```
        "KeyManager": "CUSTOMER",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "Origin": "AWS_KMS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "us-west-2"
            },
            "ReplicaKeys": [
                {
                    "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "eu-west-1"
                },
                {
                    "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "ap-northeast-1"
                },
                {
                    "Arn": "arn:aws:kms:sa-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "sa-east-1"
                }
            ]
        }
    }
}
```

Exemplo 4: como obter detalhes sobre uma chave do KMS com HMAC

O exemplo de `describe-key` a seguir obtém informações detalhadas sobre uma chave do KMS com HMAC.

```
aws kms describe-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Saída:

```
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2022-04-03T22:23:10.194000+00:00",
        "Enabled": true,
        "Description": "Test key",
        "KeyUsage": "GENERATE_VERIFY_MAC",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "HMAC_256",
        "MacAlgorithms": [
            "HMAC_SHA_256"
        ],
        "MultiRegion": false
    }
}
```

- Para obter detalhes da API, consulte DescribeKeyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
 whether the key is enabled or not
```

```
     *
     * @throws RuntimeException if an exception occurs while checking the key
state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- Para obter detalhes da API, consulte DescribeKeya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
```

```
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeKeyreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
```

```
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DescribeKeya Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """
```

```
        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeKeyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->describekey( iv_keyid = iv_key_id ).
        DATA(lo_key) = oo_result->get_keymetadata( ).
        MESSAGE 'Retrieved key information successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeKeyreferência da API AWS SDK for SAP ABAP.

Use **`DisableKey`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DisableKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Disable an AWS Key Management Service (AWS KMS) key and then retrieve
/// the key's status to show that it has been disabled.
/// </summary>
public class DisableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
```

```
            var request = new DisableKeyRequest
            {
                KeyId = keyId,
            };

            var response = await client.DisableKeyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                // Retrieve information about the key to show that it has now
                // been disabled.
                var describeResponse = await client.DescribeKeyAsync(new
 DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
 {describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- Para obter detalhes da API, consulte DisableKeya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como desativar uma chave do KMS temporariamente

O exemplo `disable-key` a seguir desabilita uma chave do KMS gerenciada pelo cliente. Use o `enable-key` comando para reabilitar a chave do KMS.

```
aws kms disable-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Este comando não produz saída.

Para obter mais informações, consulte Enabling and Disabling Keys no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte DisableKeyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously disables the specified AWS Key Management Service (KMS)
key.
    *
    * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
    * @return a CompletableFuture that, when completed, indicates that the key
has been disabled successfully
    */
   public CompletableFuture<Void> disableKeyAsync(String keyId) {
       DisableKeyRequest keyRequest = DisableKeyRequest.builder()
           .keyId(keyId)
           .build();

       return getAsyncClient().disableKey(keyRequest)
           .thenRun(() -> {
               logger.info("Key {} has been disabled successfully",keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
           });
   }
```

- Para obter detalhes da API, consulte DisableKeya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- Para obter detalhes da API, consulte a DisableKeyreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @return void
     */
```

```
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DisableKeya Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def disable_key(self, key_id: str) -> None:
        try:
            self.kms_client.disable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't disable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DisableKeyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->disablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DisableKeyreferência da API AWS SDK for SAP ABAP.

Use **`EnableKey`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `EnableKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Enable an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class EnableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to enable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new EnableKeyRequest
```

```
            {
                KeyId = keyId,
            };

            var response = await client.EnableKeyAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                // Retrieve information about the key to show that it has now
                // been enabled.
                var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- Para obter detalhes da API, consulte EnableKeya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para habilitar uma chave do KMS

O exemplo de `enable-key` a seguir habilita uma chave gerenciada pelo cliente. Você pode usar um comando como esse para habilitar uma chave do KMS que foi temporariamente desabilitada usando o comando `disable-key`. Você também pode usá-lo para habilitar uma chave do KMS que está desabilitada porque foi programada para exclusão, mas a exclusão foi cancelada.

Use o parâmetro `key-id` para especificar a chave do KMS. Este exemplo usa um valor de ID de chave, mas você pode usar um ID de chave ou o valor do ARN da chave neste comando.

Antes de executar esse comando, substitua o ID da chave de exemplo por um ID válido.

```
aws kms enable-key \
```

```
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Este comando não produz saída. Use o `describe-key` comando para verificar se a chave KMS está habilitada. Consulte os valores dos campos `KeyState` e `Enabled` na saída `describe-key`.

Para obter mais informações, consulte Enabling and Disabling Keys no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte EnableKeyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously enables the specified key.
    *
    * @param keyId the ID of the key to enable
    * @return a {@link CompletableFuture} that completes when the key has been
enabled
    */
   public CompletableFuture<Void> enableKeyAsync(String keyId) {
       EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key with ID [{}] has been enabled.", keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
```

```
                } else {
                    throw new RuntimeException("An unexpected error occurred
 while enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte EnableKeya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- Para obter detalhes da API, consulte a EnableKeyreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte EnableKeya Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key(self, key_id: str) -> None:
        """
        Enables a key. Gets the key state after each state change.

        :param key_id: The ARN or ID of the key to enable.
        """
        try:
            self.kms_client.enable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a EnableKeyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a EnableKeyreferência da API AWS SDK for SAP ABAP.

Use **`EnableKeyRotation`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `EnableKeyRotation`.

CLI

AWS CLI

Como ativar a rotação automática de uma chave do KMS

O exemplo de `enable-key-rotation` a seguir viabiliza a alternância automática de uma chave do KMS gerenciada pelo cliente com um período de alternância de 180 dias. A chave do KMS será alternada em um ano (aproximadamente 365 dias) a partir da data em que esse comando for concluído e todos os anos a partir de então.

O parâmetro `--key-id` identifica a chave do KMS. Este exemplo usa o valor do ARN da chave, mas é possível usar o ID da chave ou o ARN da chave do KMS. O parâmetro `--rotation-period-in-days` especifica o número de dias entre as datas de cada rotação. Especifique um valor entre 90 e 2.560 dias. Se nenhum valor for especificado, o valor padrão será de 365 dias.

```
aws kms enable-key-rotation \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --rotation-period-in-days 180
```

Este comando não produz saída. Use o `get-key-rotation-status` comando para verificar se a chave KMS está habilitada.

Para obter mais informações, consulte Rotacionar chaves no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte EnableKeyRotationem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []
```

```
    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a EnableKeyRotationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekeyrotation( iv_keyid = iv_key_id ).
        MESSAGE 'Key rotation enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsunsupportedopex.
        MESSAGE 'Operation not supported for this key.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a EnableKeyRotationreferência da API AWS SDK for SAP ABAP.

Use **`Encrypt`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Encrypt`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Exemplo 1: como criptografar o conteúdo de um arquivo no Linux ou no macOS

O `encrypt` comando a seguir demonstra a forma recomendada de criptografar dados com a CLI AWS .

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
```

```
    --query CiphertextBlob | base64 \
    --decode > ExampleEncryptedFile
```

O comando realiza diversas ações:

Usa o parâmetro `--plaintext` para indicar os dados a serem criptografados. Esse valor de parâmetro deve ser codificado em Base64. O valor do `plaintext` parâmetro deve ser codificado em base64 ou você deve usar o prefixo`fileb://`, que instrui a AWS CLI a ler dados binários do arquivo. Se o arquivo não estiver no diretório atual, digite o caminho completo para o arquivo. Por exemplo: `fileb:///var/tmp/ExamplePlaintextFile` ou `fileb://C:\Temp\ExamplePlaintextFile`. Para obter mais informações sobre como ler valores de parâmetros AWS CLI de um arquivo, consulte Carregando parâmetros de um arquivo no Guia dousuário da interface de linha de AWS comando e práticas recomendadas para parâmetros de arquivo locais na ferramenta de linha de AWS comando `--output` e parâmetros para controlar Blog.Uses a saída do comando. Esses `--query` parâmetros extraem os dados criptografados, chamados de texto cifrado, da saída do comando. Para obter mais informações sobre como controlar a saída, consulte Controlando o comando Saída no Guia do usuário da interface de linha de AWS comando. Usa o `base64` utilitário para decodificar a saída extraída em dados binários.O texto cifrado retornado por um comando bem-sucedido é um texto codificado em base64. `encrypt` Você deve decodificar esse texto antes de poder usar a AWS CLI para descriptografá-lo.Salva o texto cifrado binário em um arquivo.A parte final do comando > `ExampleEncryptedFile` () salva o texto cifrado binário em um arquivo para facilitar a decodificação. Para ver um exemplo de comando que usa a AWS CLI para descriptografar dados, consulte os exemplos de descriptografia.

Exemplo 2: Usando a AWS CLI para criptografar dados no Windows

Esse exemplo é igual ao exemplo anterior, mas usa a ferramenta `certutil` em vez do `base64`. Esse procedimento requer dois comandos, conforme mostrado no exemplo a seguir.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob > C:\Temp\ExampleEncryptedFile.base64

certutil -decode C:\Temp\ExampleEncryptedFile.base64 C:\Temp\ExampleEncryptedFile
```

Exemplo 3: criptografia com uma chave do KMS assimétrica

O comando `encrypt` a seguir mostra como criptografar texto simples com uma chave assimétrica do KMS. O parâmetro `--encryption-algorithm` é obrigatório. Como em todos os comandos da `encrypt` CLI, o `plaintext` parâmetro deve ser codificado em base64 ou você deve usar o `fileb://` prefixo, que instrui a AWS CLI a ler dados binários do arquivo.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --encryption-algorithm RSAES_OAEP_SHA_256 \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob | base64 \
    --decode > ExampleEncryptedFile
```

Este comando não produz saída.

- Consulte detalhes da API em Encrypt na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Encrypts the given text asynchronously using the specified KMS client and
key ID.
    *
    * @param keyId the ID of the KMS key to use for encryption
    * @param text the text to encrypt
    * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
    */
   public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
       SdkBytes myBytes = SdkBytes.fromUtf8String(text);
       EncryptRequest encryptRequest = EncryptRequest.builder()
           .keyId(keyId)
```

```
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
 getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
                logger.info("The string was encrypted with algorithm {}.",
 algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
```

```
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $text
```

```
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is
 not enabled.\n";
            }
            throw $caught;
        }
    }
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.
```

```
        :return: An instance of KeyEncrypt initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
{response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

text = '1234567890'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.encrypt({
                        key_id: keyId,
                        plaintext: text
                      })

# Display a readable version of the resulting encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn encrypt_string(
    verbose: bool,
    client: &Client,
    text: &str,
    key: &str,
    out_file: &str,
) -> Result<(), Error> {
    let blob = Blob::new(text.as_bytes());

    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    let mut ofile = File::create(out_file).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {:?}", out_file);
        println!("{}", s);
    }

    Ok(())
}
```

- Consulte detalhes da API em Criptografar na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_plaintext contains the data to encrypt
        oo_result = lo_kms->encrypt(
          iv_keyid = iv_key_id
          iv_plaintext = iv_plaintext
        ).
        MESSAGE 'Text encrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte Criptografar no AWS SDK para referência da API SAP ABAP.

Use **`GenerateDataKey`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GenerateDataKey`.

CLI

AWS CLI

Exemplo 1: como gerar uma chave de dados simétrica de 256 bits

O `generate-data-key` exemplo a seguir solicita uma chave de dados simétrica de 256 bits para uso fora do. AWS O comando retorna uma chave de dados em texto simples para uso e exclusão imediatos e uma cópia da chave de dados criptografada na chave do KMS especificada. A chave de dados criptografada pode ser armazenada com segurança junto com os dados criptografados.

Use o parâmetro `key-spec` com um valor de `AES_256` para solicitar uma chave de dados de 256 bits. Use o parâmetro `key-spec` com um valor de `AES_128` para solicitar uma chave de dados de 128 bits. Para todos os outros comprimentos de chave de dados, use o parâmetro `number-of-bytes`.

A chave do KMS especificada deve ser uma chave de criptografia simétrica do KMS, ou seja, uma chave do KMS com um valor de especificação de chave de SYMMETRIC_DEFAULT.

```
aws kms generate-data-key \
    --key-id alias/ExampleAlias \
    --key-spec AES_256
```

Saída:

```
{
    "Plaintext": "VdzKNHGzUAzJeRBVY+uUmofUGGiDzyB3+i9fVkh3piw=",
    "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
    "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
+YdhV8MrkBQPeac0ReRVNDt9qleAt+SHgIRF8P0H+7U="
}
```

A `Plaintext` (chave de dados em texto simples) e a `CiphertextBlob` (chave de dados criptografada) são retornadas no formato codificado em base64.

Para obter mais informações, consulte Data keys no Guia do desenvolvedor do AWS Key Management Service. Exemplo 2: como gerar uma chave de dados simétrica de 512 bits

O exemplo de `generate-data-key` a seguir solicita uma chave de dados simétrica de 512 bits para criptografia e descriptografia. O comando retorna uma chave de dados em texto simples para uso e exclusão imediatos e uma cópia da chave de dados criptografada na

chave do KMS especificada. A chave de dados criptografada pode ser armazenada com segurança junto com os dados criptografados.

Use o parâmetro `number-of-bytes` para solicitar um comprimento de chave diferente de 128 ou 256 bits. Para solicitar uma chave de dados de 512 bits, o exemplo a seguir usa o parâmetro `number-of-bytes` com um valor de 64 (bytes).

A chave do KMS especificada deve ser uma chave de criptografia simétrica do KMS, ou seja, uma chave do KMS com um valor de especificação de chave de SYMMETRIC_DEFAULT.

OBSERVAÇÃO: os valores na saída deste exemplo são truncados para exibição.

```
aws kms generate-data-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --number-of-bytes 64
```

Saída:

```
{
    "CiphertextBlob": "AQIBAHi6LtupRpdKl2aJTzkK6FbhOtQkMlQJJH3PdtHvS/y+hAEnX/
QQNmMwDfg2korNMEc8AAACaDCCAmQGCSqGSIb3DQEHBqCCAlUwggJRAgEAMIICSgYJKoZ...",
    "Plaintext": "ty8Lr0Bk6OF07M2BWt6qbFdNB
+G00ZLtf5MSEb4al3R2UKWGOp06njAwy2n72VRm2m7z/
Pm9Wpbvttz6a4lSo9hgPvKhZ5y6RTm4OovEXiVfBveyX3DQxDzRSwbKDPk/...",
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

`Plaintext` (chave de dados em texto simples) e `CiphertextBlob` (chave de dados criptografada) são retornadas no formato codificado em base64.

Para obter mais informações, consulte Data keys no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte GenerateDataKeyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def generate_data_key(self, key_id):
        """
        Generates a symmetric data key that can be used for client-side
 encryption.
        """
        answer = input(
            f"Do you want to generate a symmetric data key from key {key_id} (y/
n)? "
        )
        if answer.lower() == "y":
            try:
                data_key = self.kms_client.generate_data_key(
                    KeyId=key_id, KeySpec="AES_256"
                )
            except ClientError as err:
```

```
                logger.error(
                    "Couldn't generate a data key for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
            else:
                pprint(data_key)
```

- Para obter detalhes da API, consulte a GenerateDataKeyReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);
```

```
    Ok(())
}
```

- Para obter detalhes da API, consulte a GenerateDataKeyreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_keyspec = 'AES_256'
        oo_result = lo_kms->generatedatakey(
          iv_keyid = iv_key_id
          iv_keyspec = 'AES_256'
        ).
        MESSAGE 'Data key generated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GenerateDataKeyreferência da API AWS SDK for SAP ABAP.

Use **`GenerateDataKeyWithoutPlaintext`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GenerateDataKeyWithoutPlaintext`.

CLI

AWS CLI

Como gerar uma chave de dados simétrica de 256 bits sem uma chave de texto simples

O exemplo de `generate-data-key-without-plaintext` a seguir solicita uma cópia criptografada de uma chave de dados simétrica de 256 bits para uso externo à AWS. Você pode chamar o AWS KMS para descriptografar a chave de dados quando estiver pronto para usá-la.

Use o parâmetro `key-spec` com um valor de `AES_256` para solicitar uma chave de dados de 256 bits. Use o parâmetro `key-spec` com um valor de `AES_128` para solicitar uma chave de dados de 128 bits. Para todos os outros comprimentos de chave de dados, use o parâmetro `number-of-bytes`.

A chave do KMS especificada deve ser uma chave de criptografia simétrica do KMS, ou seja, uma chave do KMS com um valor de especificação de chave de SYMMETRIC_DEFAULT.

```
aws kms generate-data-key-without-plaintext \
    --key-id "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" \
    --key-spec AES_256
```

Saída:

```
{
    "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
    "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

A `CiphertextBlob` (chave de dados criptografada) é retornada em um formato codificado em base64.

Para obter mais informações, consulte Data keys no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte GenerateDataKeyWithoutPlaintextem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Para obter detalhes da API, consulte a GenerateDataKeyWithoutPlaintextreferência da API AWS SDK for Rust.

Use **`GenerateRandom`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GenerateRandom`.

CLI

AWS CLI

Exemplo 1: como gerar uma string de bytes aleatória de 256 bits (Linux ou macOS)

O exemplo de `generate-random` a seguir gera uma string de bytes aleatórios de 256 bits (32 bytes) codificada em base64. O exemplo decodifica a string de bytes e a salva no arquivo aleatório.

Ao executar esse comando, use o parâmetro `number-of-bytes` para especificar o tamanho do valor aleatório em bytes.

Você não especifica uma chave do KMS ao executar esse comando. A sequência de bytes aleatória não está relacionada a nenhuma chave do KMS.

Por padrão, o AWS KMS gera o número aleatório. No entanto, se você especificar um armazenamento de chaves personalizadas, a sequência de bytes aleatória será gerada no cluster do AWS CloudHSM associado ao armazenamento de chaves personalizadas.

Este exemplo usa os seguintes parâmetros e valores:

Ele usa o `--number-of-bytes` parâmetro necessário com um valor de `32` para solicitar uma string de 32 bytes (256 bits). Ele usa o `--output` parâmetro com um valor de para `text` direcionar a AWS CLI a retornar a saída como texto, em vez de usar o para extrair o valor da `Plaintext` propriedade da resposta.Ele canaliza (|) a saída do comando `--query parameter` para o `base64` utilitário, que decodifica a saída extraída. Ele JSON.It usa o operador de redirecionamento (**>**) para salvar string de bytes decodificada para o arquivo. Ele usa o operador de redirecionamento (**>** `ExampleRandom`) para salvar o texto cifrado binário em um arquivo.

```
aws kms generate-random \
    --number-of-bytes 32 \
    --output text \
    --query Plaintext | base64 --decode > ExampleRandom
```

Este comando não produz saída.

Para obter mais informações, consulte GenerateRandoma Referência da API do AWS Key Management Service.

Exemplo 2: como gerar um número aleatório de 256 bits (prompt de comando do Windows)

O exemplo a seguir usa o comando `generate-random` para gerar uma string aleatória de 256 bits (32 bytes) codificada em base64. O exemplo decodifica a string de bytes e a salva no arquivo aleatório. Esse exemplo é igual ao exemplo anterior, mas usa o utilitário `certutil` no Windows para decodificar em base64 a string de bytes aleatória antes de salvá-la em um arquivo.

Primeiro, gere uma string de bytes aleatória codificada em base64 e salve-a em um arquivo temporário (`ExampleRandom.base64`).

```
aws kms generate-random \
    --number-of-bytes 32 \
    --output text \
    --query Plaintext > ExampleRandom.base64
```

Como a saída do comando `generate-random` é salva em um arquivo, esse exemplo não produz nenhuma saída.

Use o comando `certutil -decode` para decodificar a string de bytes codificada em base64 no arquivo `ExampleRandom.base64`. Em seguida, ele salva a string de bytes decodificada no arquivo `ExampleRandom`.

```
certutil -decode ExampleRandom.base64 ExampleRandom
```

Saída:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Para obter mais informações, consulte GenerateRandoma Referência da API do AWS Key Management Service.

- Para obter detalhes da API, consulte GenerateRandomem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
        .number_of_bytes(length)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.plaintext.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Para obter detalhes da API, consulte a GenerateRandomreferência da API AWS SDK for Rust.

Use **`GetKeyPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetKeyPolicy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como copiar uma política de chave de uma chave do KMS para outra chave do KMS

O exemplo de `get-key-policy` a seguir obtém a política de chave de uma chave do KMS e a salva em um arquivo de texto. Em seguida, ele substitui a política de uma chave diferente do KMS usando o arquivo de texto como a entrada da política.

Como o parâmetro `--policy` de `put-key-policy` requer uma string, você deve usar a opção `--output text` para retornar a saída como uma string de texto em vez de JSON.

```
aws kms get-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --query Policy \
    --output text > policy.txt

aws kms put-key-policy \
    --policy-name default \
    --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --policy file://policy.txt
```

Este comando não produz saída.

Para obter mais informações, consulte PutKeyPolicya Referência da API AWS KMS.

- Para obter detalhes da API, consulte GetKeyPolicyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Para obter detalhes da API, consulte a GetKeyPolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->getkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
        ).
        MESSAGE 'Retrieved key policy successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetKeyPolicyreferência da API AWS SDK for SAP ABAP.

Use **`ListAliases`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListAliases`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// List the AWS Key Management Service (AWS KMS) aliases that have been
defined for
/// the keys in the same AWS Region as the default user. If you want to list
/// the aliases in a different Region, pass the Region to the client
/// constructor.
/// </summary>
public class ListAliases
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListAliasesRequest();
        var response = new ListAliasesResponse();

        do
        {
            response = await client.ListAliasesAsync(request);

            response.Aliases.ForEach(alias =>
            {
                Console.WriteLine($"Created: {alias.CreationDate} Last
Update: {alias.LastUpdatedDate} Name: {alias.AliasName}");
            });

            request.Marker = response.NextMarker;
        }
```

```
            while (response.Truncated);
        }
    }
```

- Para obter detalhes da API, consulte ListAliasesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: Para listar todos os aliases em uma AWS conta e região

O exemplo a seguir usa o `list-aliases` comando para listar todos os aliases na região padrão da AWS conta. O resultado inclui aliases associados às chaves KMS AWS gerenciadas e às chaves KMS gerenciadas pelo cliente.

```
aws kms list-aliases
```

Saída:

```
{
    "Aliases": [
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/testKey",
            "AliasName": "alias/testKey",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/FinanceDept",
            "AliasName": "alias/FinanceDept",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
            "AliasName": "alias/aws/dynamodb",
            "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
            "AliasName": "alias/aws/ebs",
```

```
            "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef"
        },
        ...
    ]
}
```

Exemplo 2: como listar todos os aliases de uma chave específica do KMS

O exemplo a seguir usa o comando `list-aliases` e seu parâmetro `key-id` para listar todos os aliases associados a uma determinada chave do KMS.

Cada alias é associado a apenas um chave do KMS, mas uma chave do KMS pode ter vários aliases. Esse comando é muito útil porque o console AWS KMS lista somente um alias para cada chave KMS. Você deve usar o comando `list-aliases` para localizar todos os aliases de uma chave do KMS.

Este exemplo usa o ID de chave da chave do KMS no parâmetro `--key-id`, mas você pode usar um ID de chave, o ARN da chave, o nome do alias ou o ARN do alias neste comando.

```
aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Saída:

```
{
    "Aliases": [
        {
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/oregon-test-
key",
            "AliasName": "alias/oregon-test-key"
        },
        {
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project121-
test",
            "AliasName": "alias/project121-test"
        }
    ]
}
```

Para obter mais informações, consulte Working with aliases no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte ListAliasesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously lists all the aliases in the current AWS account.
    *
    * @return a {@link CompletableFuture} that completes when the list of
aliases has been processed
    */
   public CompletableFuture<Object> listAllAliasesAsync() {
       ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
           .limit(15)
           .build();

       ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
       return paginator.subscribe(response -> {
               response.aliases().forEach(alias ->
                   logger.info("The alias name is: " + alias.aliasName())
               );
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               if (ex.getCause() instanceof KmsException) {
                   KmsException e = (KmsException) ex.getCause();
                   throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
               }
           });
   }
```

- Para obter detalhes da API, consulte ListAliasesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListAliasesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
 where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte ListAliasesa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AliasManager:
```

```
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)?
"
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListAliasesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_kms->listaliases( ).
    MESSAGE 'Retrieved KMS aliases list.' TYPE 'I'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListAliasesreferência da API AWS SDK for SAP ABAP.

Use **`ListGrants`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListGrants`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// List the AWS Key Management Service (AWS KMS) grants that are associated
with
/// a specific key.
/// </summary>
public class ListGrants
{
    public static async Task Main()
    {
        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListGrantsRequest
        {
            KeyId = keyId,
        };

        var response = new ListGrantsResponse();

        do
        {
            response = await client.ListGrantsAsync(request);

            response.Grants.ForEach(grant =>
            {
                Console.WriteLine($"{grant.GrantId}");
```

```
                });

                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- Para obter detalhes da API, consulte ListGrantsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para visualizar as concessões em uma chave AWS KMS

O `list-grants` exemplo a seguir exibe todas as concessões na chave KMS AWS gerenciada especificada para o Amazon DynamoDB em sua conta. Essa concessão permite que o DynamoDB use a chave do KMS em seu nome para criptografar uma tabela do DynamoDB antes de gravá-la no disco. Você pode usar um comando como esse para visualizar as concessões nas chaves KMS AWS gerenciadas e nas chaves KMS gerenciadas pelo cliente na AWS conta e na região.

Este comando usa o parâmetro `key-id` com um ID de chave para identificar a chave do KMS. Você pode usar um ID de chave ou o ARN da chave para identificar a chave do KMS. Para obter o ID da chave ou o ARN da chave de uma chave KMS AWS gerenciada, use o `list-keys` comando ou. `list-aliases`

```
aws kms list-grants \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

O resultado mostra que a concessão fornece ao Amazon DynamoDB permissão para usar a chave do KMS para operações criptográficas e permite que ele visualize detalhes sobre a chave do KMS (`DescribeKey`) e descontinue concessões (`RetireGrant`). A restrição `EncryptionContextSubset` limita a permissão às solicitações que incluem os pares de contexto de criptografia especificados. Como resultado, as permissões na concessão são efetivas somente na conta especificada e na tabela do DynamoDB.

```
{
    "Grants": [
        {
            "Constraints": {
                "EncryptionContextSubset": {
                    "aws:dynamodb:subscriberId": "123456789012",
                    "aws:dynamodb:tableName": "Services"
                }
            },
            "IssuingAccount": "arn:aws:iam::123456789012:root",
            "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "ReEncryptFrom",
                "ReEncryptTo",
                "RetireGrant",
                "DescribeKey"
            ],
            "GrantId":
 "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59",
            "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",
            "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",
            "CreationDate": "2021-05-13T18:32:45.144000+00:00"
        }
    ]
}
```

Para obter mais informações, consulte Concessões no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte ListGrantsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously displays the grant IDs for the specified key ID.
    *
    * @param keyId the ID of the AWS KMS key for which to list the grants
    * @return a {@link CompletableFuture} that, when completed, will be null
if the operation succeeded, or will throw a {@link RuntimeException} if the
operation failed
    * @throws RuntimeException if there was an error listing the grants, either
due to an {@link KmsException} or an unexpected error
    */
   public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
       ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
           .keyId(keyId)
           .limit(15)
           .build();

       ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
       return paginator.subscribe(response -> {
               response.grants().forEach(grant -> {
                   logger.info("The grant Id is: " + grant.grantId());
               });
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               Throwable cause = ex.getCause();
               if (cause instanceof KmsException) {
                   throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
```

```
                }
            });
    }
```

- Para obter detalhes da API, consulte ListGrantsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListGrantsreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity
or resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte ListGrantsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
```

```
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListGrantsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listgrants( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved grants list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListGrantsreferência da API AWS SDK for SAP ABAP.

Use **`ListKeyPolicies`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListKeyPolicies`.

CLI

AWS CLI

Como obter os nomes das políticas de chave de uma chave do KMS

O exemplo de `list-key-policies` a seguir obtém os nomes das políticas de chave de uma chave gerenciada pelo cliente na conta e região de exemplo. Você pode usar esse comando para encontrar os nomes das políticas de chaves para chaves AWS gerenciadas e chaves gerenciadas pelo cliente.

Como o único nome de política de chave válido é `default`, esse comando não é muito útil.

Use o parâmetro `key-id` para especificar a chave do KMS. Este exemplo usa um valor de ID de chave, mas você pode usar um ID de chave ou o ARN da chave neste comando.

```
aws kms list-key-policies \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Saída:

```
{
    "PolicyNames": [
    "default"
    ]
}
```

Para obter mais informações sobre as políticas de chaves do AWS KMS, consulte Usando políticas de chaves no AWS KMS no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte ListKeyPoliciesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously retrieves the key policy for the specified key ID and
policy name.
    *
    * @param keyId       the ID of the AWS KMS key for which to retrieve the
policy
    * @param policyName the name of the key policy to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
    */
   public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
       GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
           .keyId(keyId)
           .policyName(policyName)
           .build();

       return getAsyncClient().getKeyPolicy(policyRequest)
           .thenApply(response -> {
               String policy = response.policy();
               logger.info("The response is: " + policy);
               return policy;
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Failed to get key policy", ex);
           });
   }
```

- Para obter detalhes da API, consulte ListKeyPoliciesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)[
                "PolicyNames"
            ]
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- Para obter detalhes da API, consulte a ListKeyPoliciesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listkeypolicies( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved key policies list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListKeyPoliciesreferência da API AWS SDK for SAP ABAP.

Use **`ListKeys`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListKeys`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

   /// <summary>
   /// List the AWS Key Managements Service (AWS KMS) keys for the AWS Region
   /// of the default user. To list keys in another AWS Region, supply the
Region
   /// as a parameter to the client constructor.
   /// </summary>
   public class ListKeys
   {
       public static async Task Main()
       {
           var client = new AmazonKeyManagementServiceClient();
           var request = new ListKeysRequest();
           var response = new ListKeysResponse();

           do
           {
               response = await client.ListKeysAsync(request);

               response.Keys.ForEach(key =>
               {
                   Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
               });

               // Set the Marker property when response.Truncated is true
               // in order to get the next keys.
```

```
                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- Para obter detalhes da API, consulte ListKeysa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como obter chaves do KMS em uma conta e região

O exemplo `list-keys` a seguir obtém as chaves do KMS em uma conta e região. Esse comando retorna as chaves AWS gerenciadas e as chaves gerenciadas pelo cliente.

```
aws kms list-keys
```

Saída:

```
{
    "Keys": [
        {
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
            "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
        },
        {
            "KeyArn": "arn:aws:kms:us-
east-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
            "KeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        }
    ]
}
```

Para obter mais informações, consulte Viewing Keys no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte ListKeysem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
```

```
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
 Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " +
 exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ListKeysa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListKeysreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
```

```
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte ListKeysa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
```

```
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            key_paginator = self.kms_client.get_paginator("list_keys")
            for key_page in key_paginator.paginate(PaginationConfig={"PageSize":
10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page["Keys"])
                if key_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} keys (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s",
                err.response["Error"]["Message"],
            )
```

- Para obter detalhes da API, consulte a ListKeysReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListKeysreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kms->listkeys( ).
        MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListKeysreferência da API AWS SDK for SAP ABAP.

Use **`PutKeyPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutKeyPolicy`.

CLI

AWS CLI

Como alterar a política de chaves de uma chave do KMS

O exemplo de `put-key-policy` a seguir altera a política de chave de uma chave gerenciada pelo cliente.

Para começar, crie uma política de chave e salve-a em um arquivo JSON local. Neste exemplo, o arquivo é `key_policy.json`. Você também pode especificar a política de chave como o valor da string do parâmetro `policy`.

A primeira declaração nessa política de chaves dá à AWS conta permissão para usar políticas do IAM para controlar o acesso à chave KMS. A segunda instrução fornece permissão de usuário ao `test-user` para executar os comandos `describe-key` e `list-keys` na chave do KMS.

Conteúdo de `key_policy.json`:

```
{
    "Version":"2012-10-17",
    "Id" : "key-default-1",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
```

```
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:root"
            },
            "Action" : "kms:*",
            "Resource" : "*"
        },
        {
            "Sid" : "Allow Use of Key",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:user/test-user"
            },
            "Action" : [
                "kms:DescribeKey",
                "kms:ListKeys"
            ],
            "Resource" : "*"
        }
    ]
}
```

Este exemplo usa o ID de chave para identificar a chave do KMS, mas também é possível usar um ARN de chave. O comando usa o parâmetro `policy` para especificar a política de chave. Para indicar que a política está em um arquivo, ela usa o prefixo `file://` necessário. Esse prefixo é necessário para identificar arquivos em todos os sistemas operacionais compatíveis. Por fim, o comando usa o parâmetro `policy-name` com um valor `default`. Se não for especificado um nome de política, o valor padrão será `default`. O único valor válido é `default`.

```
aws kms put-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --policy file://key_policy.json
```

Esse comando não produz nenhuma saída. Use o comando `get-key-policy` para verificar se o comando está em vigor. O comando de exemplo a seguir recupera a política de chave da mesma chave do KMS. O parâmetro `output` com o valor `text` retorna um formato de texto fácil de ler.

```
aws kms get-key-policy \
```

```
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --output text
```

Saída:

```
{
    "Version":"2012-10-17",
    "Id" : "key-default-1",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:root"
            },
            "Action" : "kms:*",
            "Resource" : "*"
            },
            {
            "Sid" : "Allow Use of Key",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:user/test-user"
            },
            "Action" : [ "kms:Describe", "kms:List" ],
            "Resource" : "*"
        }
    ]
}
```

Para obter mais informações, consulte Changing a Key Policy no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte PutKeyPolicyem Referência de AWS CLI Comandos.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte PutKeyPolicya Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
 existing
        policy, so care is taken to add a statement to the existing list of
 statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: The existing policy of the key.
        :return: None
        """
        principal = input(
            "Enter the ARN of an IAM role to set as the principal on the policy:
 "
        )
```

```
        if key_id != "" and principal != "":
            # The updated policy replaces the existing policy. Add a new
statement to
            # the list along with the original policy statements.
            policy["Statement"].append(
                {
                    "Sid": "Allow access for ExampleRole",
                    "Effect": "Allow",
                    "Principal": {"AWS": principal},
                    "Action": [
                        "kms:Encrypt",
                        "kms:GenerateDataKey*",
                        "kms:Decrypt",
                        "kms:DescribeKey",
                        "kms:ReEncrypt*",
                    ],
                    "Resource": "*",
                }
            )
            try:
                self.kms_client.put_key_policy(KeyId=key_id,
Policy=json.dumps(policy))
            except ClientError as err:
                logger.error(
                    "Couldn't set policy for key %s. Here's why %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                print(f"Set policy for key {key_id}.")
        else:
            print("Skipping set policy demo.")
```

- Para obter detalhes da API, consulte a PutKeyPolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_policy = '{"Version":"2012-10-17",        "Statement": [...]}'
        lo_kms->putkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
          iv_policy = iv_policy
        ).
        MESSAGE 'Key policy updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsmalformedplydocex.
        MESSAGE 'Malformed policy document.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutKeyPolicyreferência da API AWS SDK for SAP ABAP.

Use **`ReEncrypt`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ReEncrypt`.

CLI

AWS CLI

Exemplo 1: como criptografar novamente uma mensagem criptografada com uma chave simétrica diferente do KMS (Linux e macOS).

O exemplo de `re-encrypt` comando a seguir demonstra a forma recomendada de recriptografar dados com a CLI. AWS

Forneça o texto cifrado em um arquivo. No valor do parâmetro `--ciphertext-blob`, use o prefixo `fileb://`, que instrui a CLI a ler os dados de um arquivo binário. Se o arquivo não estiver no diretório atual, digite o caminho completo para o arquivo. Para obter mais informações sobre a leitura dos valores dos parâmetros da AWS CLI de um arquivo, consulte Carregando parâmetros da AWS CLI de um arquivo no Guia do usuário da interface de linha de AWS comando e as melhores práticas para parâmetros de arquivos locais no blog da ferramenta de linha de AWS comando .Especifique a chave KMS de origem, que descriptografa o texto cifrado.O parâmetro não é necessário ao descriptografar com chaves KMS de criptografia simétrica. `--source-key-id` AWS O KMS pode obter a chave KMS usada para criptografar os dados dos metadados no blob de texto cifrado. Porém, sempre é uma prática recomendada especificar a chave do KMS que você está usando. Essa prática garante que você use a chave do KMS desejada e impede que você descriptografe um texto cifrado acidentalmente usando uma chave do KMS em que você não confia. Especifique a chave do KMS de destino, que criptografa os dados novamente. O parâmetro `--destination-key-id` é sempre obrigatório. Este exemplo usa um ARN de chave, mas você pode usar qualquer identificador de chave válido. Solicite a saída de texto simples como um valor de texto. O parâmetro `--query` informa à CLI para obter somente o valor do campo `Plaintext` da saída. O `--output` parâmetro retorna a saída como texto. Base64-decode o texto simples e salve-o em um arquivo. O exemplo a seguir canaliza (|) o valor do `Plaintext` parâmetro para o utilitário Base64, que o decodifica. Em seguida, ele redireciona (>) a saída decodificada para o arquivo `ExamplePlaintext`.

Antes de executar esse comando, substitua os exemplos de IDs de chave por identificadores de chave válidos da sua AWS conta.

```
aws kms re-encrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
```

```
    --query CiphertextBlob \
    --output text | base64 --decode > ExampleReEncryptedFile
```

Este comando não produz saída. A saída do comando `re-encrypt` é decodificada em base64 e salva em um arquivo.

Para obter mais informações, consulte ReEncrypta Referência da API do AWS Key Management Service.

Exemplo 2: como criptografar novamente uma mensagem criptografada com uma chave simétrica diferente do KMS (prompt de comando do Windows).

O exemplo de `re-encrypt` comando a seguir é igual ao anterior, exceto que ele usa o `certutil` utilitário para Base64-decode os dados em texto simples. Esse procedimento requer dois comandos, conforme mostrado nos exemplos a seguir.

Antes de executar esse comando, substitua o ID de chave de exemplo por um ID de chave válido da sua AWS conta.

```
aws kms re-encrypt ^
    --ciphertext-blob fileb://ExampleEncryptedFile ^
    --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
    --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 ^
    --query CiphertextBlob ^
    --output text > ExampleReEncryptedFile.base64
```

Em seguida, use o utilitário `certutil`.

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

Saída:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Para obter mais informações, consulte ReEncrypta Referência da API do AWS Key Management Service.

- Para obter detalhes da API, consulte ReEncryptem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def re_encrypt(self, source_key_id, cipher_text):
        """
        Takes ciphertext previously encrypted with one key and reencrypt it by
 using
        another key.

        :param source_key_id: The ARN or ID of the original key used to encrypt
 the
                              ciphertext.
        :param cipher_text: The encrypted ciphertext.
        :return: The ciphertext encrypted by the second key.
        """
        destination_key_id = input(
            f"Your ciphertext is currently encrypted with key {source_key_id}. "
            f"Enter another key ID or ARN to reencrypt it: "
        )
```

```
        if destination_key_id != "":
            try:
                cipher_text = self.kms_client.re_encrypt(
                    SourceKeyId=source_key_id,
                    DestinationKeyId=destination_key_id,
                    CiphertextBlob=cipher_text,
                )["CiphertextBlob"]
            except ClientError as err:
                logger.error(
                    "Couldn't reencrypt your ciphertext. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Reencrypted your ciphertext as: {cipher_text}")
                return cipher_text
        else:
            print("Skipping reencryption demo.")
```

- Para obter detalhes da API, consulte a ReEncryptReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593
sourceCiphertextBlob = [blob].pack('H*')
```

```
# Replace the fictitious key ARN with a valid key ID

destinationKeyId = 'arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.re_encrypt({
                           ciphertext_blob: sourceCiphertextBlob,
                           destination_key_id: destinationKeyId
                         })

# Display a readable version of the resulting re-encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Para obter detalhes da API, consulte ReEncrypta Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn reencrypt_string(
    verbose: bool,
    client: &Client,
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
```

```
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client
        .re_encrypt()
        .ciphertext_blob(data.unwrap())
        .source_key_id(first_key)
        .destination_key_id(new_key)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);
    let o = &output_file;

    let mut ofile = File::create(o).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {}:", output_file);
        println!("{}", s);
    } else {
        println!("Wrote base64-encoded output to {}", output_file);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ReEncryptreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_source_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_destination_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->reencrypt(
          iv_sourcekeyid = iv_source_key_id
          iv_destinationkeyid = iv_destination_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Ciphertext reencrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect source key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ReEncryptreferência da API AWS SDK for SAP ABAP.

Use **`RetireGrant`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RetireGrant`.

CLI

AWS CLI

Como descontinuar uma concessão em uma chave mestra do cliente

O exemplo de `retire-grant` a seguir exclui a concessão de uma chave do KMS.

O comando de exemplo a seguir especifica os parâmetros `grant-id` e `key-id`. O valor do parâmetro `key-id` deve ser o ARN de chave da chave do KMS.

```
aws kms retire-grant \
```

```
    --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Este comando não produz saída. Use o comando `list-grants` para confirmar a descontinuação da concessão.

Para obter mais informações, consulte Retiring and revoking grants no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte RetireGrantem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def retire_grant(self, grant):
        """
        Retires a grant so that it can no longer be used.
```

```
        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant["GrantToken"])
        except ClientError as err:
            logger.error(
                "Couldn't retire grant %s. Here's why: %s",
                grant["GrantId"],
                err.response["Error"]["Message"],
            )
        else:
            print(f"Grant {grant['GrantId']} retired.")
```

- Para obter detalhes da API, consulte a RetireGrantReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_grant_token = 'AQpAM2RhZ...'
    lo_kms->retiregrant( iv_granttoken = iv_grant_token ).
    MESSAGE 'Grant retired successfully.' TYPE 'I'.
  CATCH /aws1/cx_kmsnotfoundexception.
    MESSAGE 'Grant not found.' TYPE 'E'.
  CATCH /aws1/cx_kmsinvgranttokenex.
    MESSAGE 'Invalid grant token.' TYPE 'E'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a RetireGrantreferência da API AWS SDK for SAP ABAP.

Use **`RevokeGrant`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RevokeGrant`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como revogar uma concessão em uma chave mestra do cliente

O exemplo de `revoke-grant` a seguir exclui a concessão de uma chave do KMS. O comando de exemplo a seguir especifica os parâmetros `grant-id` e `key-id`. O valor do parâmetro `key-id` pode ser o ID ou o ARN de chave da chave do KMS.

```
aws kms revoke-grant \
    --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Este comando não produz saída. Use o comando `list-grants` para confirmar que a concessão foi revogada.

Para obter mais informações, consulte Retiring and revoking grants no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte RevokeGrantem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
           } else {
               if (exception instanceof KmsException kmsEx) {
                   if (kmsEx.getMessage().contains("Grant does not exist")) {
                       logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
                   } else {
```

```
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }
```

- Para obter detalhes da API, consulte RevokeGrant a Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte RevokeGrant a Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.
```

```
        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a RevokeGrantReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grant_id = '1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p'
        lo_kms->revokegrant(
          iv_keyid = iv_key_id
          iv_grantid = iv_grant_id
        ).
        MESSAGE 'Grant revoked successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Grant or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidgrantidex.
```

```
        MESSAGE 'Invalid grant ID.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RevokeGrantreferência da API AWS SDK for SAP ABAP.

Use **`ScheduleKeyDeletion`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ScheduleKeyDeletion`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Como programar a exclusão de uma chave do KMS gerenciada pelo cliente.

O exemplo de `schedule-key-deletion` a seguir programa a chave do KMS gerenciada pelo cliente especificada pala ser excluída em 15 dias.

O parâmetro `--key-id` identifica a chave do KMS. Este exemplo usa o valor do ARN da chave, mas você pode usar o ID da chave ou o ARN da chave do KMS. O parâmetro `--pending-window-in-days` especifica a duração do período de espera (7 a 30 dias). Por padrão, o período de espera é de 30 dias. Este exemplo especifica um valor de 15, que indica AWS a exclusão permanente da chave KMS 15 dias após a conclusão do comando.

```
aws kms schedule-key-deletion \
    --key-id arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --pending-window-in-days 15
```

A resposta inclui o ARN da chave, o estado da chave, o período de espera (`PendingWindowInDays`) e a data de exclusão no horário Unix. Para ver a data de exclusão

no horário local, use o console AWS KMS. As chaves do KMS no estado `PendingDeletion` não podem ser usadas em operações criptográficas.

```
{
    "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "DeletionDate": "2022-06-18T23:43:51.272000+00:00",
    "KeyState": "PendingDeletion",
    "PendingWindowInDays": 15
}
```

Para obter mais informações, consulte Deleting keys no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte ScheduleKeyDeletionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
     * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted
using the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when
deleting KMS keys.</p>
     *
     * @param keyId the ID of the KMS key to delete
```

```
     * @return a {@link CompletableFuture} that completes when the key deletion
is scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for
key ID: " + keyId, throwable);
            });
    }
```

- Para obter detalhes da API, consulte ScheduleKeyDeletiona Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
```

```
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte ScheduleKeyDeletiona Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.
```

```
        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ScheduleKeyDeletionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_pending_window_days = 7
        oo_result = lo_kms->schedulekeydeletion(
          iv_keyid = iv_key_id
          iv_pendingwindowindays = iv_pending_window_days
        ).
        MESSAGE 'Key scheduled for deletion.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ScheduleKeyDeletionreferência da API AWS SDK for SAP ABAP.

Use o **`Sign`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Sign`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Exemplo 1: gerar uma assinatura digital para uma mensagem

O exemplo de `sign` a seguir gera uma assinatura criptográfica para uma mensagem curta. A saída do comando inclui um campo `Signature` codificado em base 64 que pode ser verificado usando o comando `verify`.

É necessário especificar uma mensagem para assinar e um algoritmo de assinatura compatível com sua chave do KMS assimétrica. Para obter os algoritmos de assinatura para sua chave do KMS, use o comando `describe-key`.

Na AWS CLI v2, o valor do `message` parâmetro deve ser. Base64-encoded Ou você pode salvar a mensagem em um arquivo e usar o `fileb://` prefixo, que AWS instrui a CLI a ler dados binários do arquivo.

Antes de executar esse comando, substitua o ID de chave de exemplo por um ID de chave válido da sua AWS conta. O ID da chave deverá representar uma chave assimétrica do KMS com o uso da chave SIGN_VERIFY.

```
msg=(echo 'Hello World' | base64)

aws kms sign \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://UnsignedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256
```

Saída:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "Signature": "ABCDEFhpyVYyTxbafE74ccSvEJLJr3zuoV1Hfymz4qv+/
fxmxNLA7SE1SiF8lHw80fKZZ3bJ...",
    "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

Para obter mais informações sobre o uso de chaves KMS assimétricas no AWS KMS, consulte Chaves assimétricas no KMS no Guia do desenvolvedor do AWS Key Management Service.AWS

Exemplo 2: salvar uma assinatura digital em um arquivo (Linux e macOS)

O exemplo de `sign` a seguir gera uma assinatura criptográfica para uma mensagem curta armazenada em um arquivo local. O comando também obtém a `Signature` propriedade da resposta e Base64-decodes a salva no ExampleSignature arquivo. É possível usar o arquivo de assinatura em um comando `verify` que verifique a assinatura.

O `sign` comando requer uma Base64-encoded mensagem e um algoritmo de assinatura compatíveis com sua chave KMS assimétrica. Para obter os algoritmos de assinatura compatíveis com sua chave do KMS, use o comando `describe-key`.

Antes de executar esse comando, substitua o ID de chave de exemplo por um ID de chave válido da sua AWS conta. O ID da chave deve representar uma chave KMS assimétrica com o uso da chave SIGN_VERIFY.

```
echo 'hello world' | base64 > EncodedMessage

aws kms sign \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://EncodedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
    --output text \
    --query Signature | base64 --decode > ExampleSignature
```

Este comando não produz saída. Este exemplo extrai a propriedade `Signature` da saída e a salva em um arquivo.

Para obter mais informações sobre o uso de chaves KMS assimétricas no AWS KMS, consulte Chaves assimétricas no KMS no Guia do desenvolvedor do AWS Key Management Service.AWS

- Consulte detalhes da API em Sign na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously signs and verifies data using AWS KMS.
    *
    * <p>The method performs the following steps:
    * <ol>
    *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
    *     <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
    *     <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
    * </ol>
    *
    * @return a {@link CompletableFuture} that completes with the result of the
signature verification,
    *         {@code true} if the signature is valid, {@code false} otherwise.
    * @throws KmsException if any error occurs during the KMS operations.
    * @throws RuntimeException if an unexpected error occurs.
    */
   public CompletableFuture<Boolean> signVerifyDataAsync() {
       String signMessage = "Here is the message that will be digitally signed";

       // Create an AWS KMS key used to digitally sign data.
       CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
           .keySpec(KeySpec.RSA_2048)
           .keyUsage(KeyUsageType.SIGN_VERIFY)
           .origin(OriginType.AWS_KMS)
           .build();

       return getAsyncClient().createKey(createKeyRequest)
           .thenCompose(createKeyResponse -> {
               String keyId = createKeyResponse.keyMetadata().keyId();
```

```
                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes =
signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }
```

- Consulte detalhes da API em Sign na Referência da API do AWS SDK for Java 2.x .

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Consulte detalhes da API em Sign na Referência da API do AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
```

```
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Consulte detalhes da API em Sign na Referência API do AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the message to sign
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->sign(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signingalgorithm = iv_signing_algorithm
        ).
        MESSAGE 'Message signed successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidkeyusageex.
        MESSAGE 'Key cannot be used for signing.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a referência da API Sign in AWS SDK for SAP ABAP.

Use **`TagResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TagResource`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para adicionar uma tag a uma chave do KMS

O exemplo de `tag-resource` a seguir adiciona as tags `"Purpose":"Test"` e `"Dept":"IT"` a uma chave do KMS gerenciada pelo cliente. É possível usar tags como essas para identificar chaves do KMS e criar categorias de chaves do KMS para permissões e auditoria.

Use o parâmetro `key-id` para especificar a chave do KMS. Este exemplo usa um valor de ID de chave, mas você pode usar um ID de chave ou o ARN da chave neste comando.

```
aws kms tag-resource \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --tags TagKey='Purpose',TagValue='Test' TagKey='Dept',TagValue='IT'
```

Este comando não produz saída. Para visualizar as tags em uma chave AWS KMS KMS, use o `list-resource-tags` comando.

Para obter mais informações sobre o uso de tags no AWS KMS, consulte Como marcar chaves no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte TagResourceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging
operation is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key",
throwable);
            });
    }
```

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
```

```
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a TagResourceReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_tags TYPE /aws1/cl_kmstag=>tt_taglist.

    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_tag_key = 'Environment'
        " iv_tag_value = 'Production'
        APPEND NEW /aws1/cl_kmstag(
          iv_tagkey = iv_tag_key
          iv_tagvalue = iv_tag_value
        ) TO lt_tags.

        lo_kms->tagresource(
          iv_keyid = iv_key_id
          it_tags = lt_tags
        ).
        MESSAGE 'Tag added to KMS key successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmstagexception.
        MESSAGE 'Invalid tag format.' TYPE 'E'.
```

```
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a TagResourcereferência da API AWS SDK for SAP ABAP.

Use **`UpdateAlias`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateAlias`.

CLI

AWS CLI

Como associar um alias a uma chave do KMS diferente

O exemplo de `update-alias` a seguir associa o alias `alias/test-key` a uma chave do KMS diferente.

O parâmetro `--alias-name` especifica o alias. O valor do nome do alias deve começar com `alias/`. O parâmetro `--target-key-id` especifica a chave do KMS que deve ser associada ao alias. Não é necessário especificar a chave do KMS atual para o alias.

```
aws kms update-alias \
    --alias-name alias/test-key \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Este comando não produz saída. Use o `list-aliases` comando para localizar o alias.

Para obter mais informações, consulte Updating aliases no Guia do desenvolvedor do AWS Key Management Service.

- Para obter detalhes da API, consulte UpdateAliasem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def update_alias(self, alias, current_key_id):
        """
        Updates an alias by assigning it to another key.

        :param alias: The alias to reassign.
        :param current_key_id: The ARN or ID of the key currently associated with
 the alias.
        """
        new_key_id = input(
            f"Alias {alias} is currently associated with {current_key_id}. "
            f"Enter another key ID or ARN that you want to associate with
 {alias}: "
        )
        if new_key_id != "":
            try:
```

```
                self.kms_client.update_alias(AliasName=alias,
 TargetKeyId=new_key_id)
            except ClientError as err:
                logger.error(
                    "Couldn't associate alias %s with key %s. Here's why: %s",
                    alias,
                    new_key_id,
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Alias {alias} is now associated with key {new_key_id}.")
        else:
            print("Skipping alias update.")
```

- Para obter detalhes da API, consulte a UpdateAliasReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_target_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        lo_kms->updatealias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_target_key_id
        ).
        MESSAGE 'Alias updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Alias or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
```

```
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateAliasreferência da API AWS SDK for SAP ABAP.

Use **`Verify`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Verify`.

CLI

AWS CLI

Como verificar uma assinatura digital

O `verify` comando a seguir verifica uma assinatura criptográfica para uma mensagem curta. Base64-encoded O ID da chave, a mensagem, o tipo de mensagem e o algoritmo de assinatura devem ser os mesmos usados para assinar a mensagem.

Na AWS CLI v2, o valor do `message` parâmetro deve ser. Base64-encoded Ou você pode salvar a mensagem em um arquivo e usar o `fileb://` prefixo, que AWS instrui a CLI a ler dados binários do arquivo.

A assinatura que especificada não pode ter codificação base64. Para obter ajuda na decodificação da assinatura retornada pelo comando `sign`, consulte os exemplos do comando `sign`.

A saída do comando inclui um campo booleano `SignatureValid` que indica que a assinatura foi verificada. Se a validação da assinatura falhar, o comando `verify` também falhará.

Antes de executar esse comando, substitua o ID de chave de exemplo por um ID de chave válido da sua AWS conta.

```
aws kms verify \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://EncodedMessage \
    --message-type RAW \
```

```
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
    --signature fileb://ExampleSignature
```

Saída:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "SignatureValid": true,
    "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

Para obter mais informações sobre o uso de chaves KMS assimétricas no AWS KMS, consulte Usando chaves assimétricas no Guia do desenvolvedor do AWS Key Management Service.

- Consulte detalhes da API em Verify na Referência de comandos da AWS CLI .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
```

```
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Consulte detalhes da API em Verify na Referência da API do AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the original message
        " iv_signature contains the signature to verify
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->verify(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signature = iv_signature
          iv_signingalgorithm = iv_signing_algorithm
        ).
        DATA(lv_valid) = oo_result->get_signaturevalid( ).
        IF lv_valid = abap_true.
          MESSAGE 'Signature is valid.' TYPE 'I'.
        ELSE.
          MESSAGE 'Signature is invalid.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinvalidsigex.
        MESSAGE 'Invalid signature.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte Verificar no AWS SDK para referência da API SAP ABAP.

Cenários para AWS KMS usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS KMS com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS KMS ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Trabalhe com a criptografia de tabelas do DynamoDB usando AWS Command Line Interface v2 da

Trabalhe com a criptografia de tabelas do DynamoDB usando AWS Command Line Interface v2 da

O exemplo de código a seguir mostra como gerenciar opções de criptografia para tabelas do DynamoDB.

- Crie uma tabela com a criptografia padrão.
- Crie uma tabela com uma chave gerenciada pelo cliente (CMK).
- Atualize as configurações de criptografia da tabela.
- Descreva a criptografia da tabela.

Bash

AWS CLI com script Bash

Crie uma tabela com a criptografia padrão.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Crie uma tabela com uma chave gerenciada pelo cliente (CMK).

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Atualize a criptografia da tabela.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Descreva a criptografia da tabela.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- CreateKey
- CreateTable
- DescribeTable
- UpdateTable

Exemplos de código para Lambda usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Lambda com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS as contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Mais atributos

- Guia do desenvolvedor do Lambda: mais informações sobre o Lambda.
- Referência da API Lambda: detalhes sobre todas as ações disponíveis do Lambda.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de uso do Lambda AWS SDKs
  - Olá, Lambda
  - Aprenda os conceitos básicos do Lambda com um AWS SDK

- Ações para Lambda usando AWS SDKs
  - Usar CreateAlias com uma CLI
  - Use CreateFunctioncom um AWS SDK ou CLI
  - Usar DeleteAlias com uma CLI
  - Use DeleteFunctioncom um AWS SDK ou CLI
  - Usar DeleteFunctionConcurrency com uma CLI
  - Usar DeleteProvisionedConcurrencyConfig com uma CLI
  - Usar GetAccountSettings com uma CLI
  - Usar GetAlias com uma CLI
  - Use GetFunctioncom um AWS SDK ou CLI
  - Usar GetFunctionConcurrency com uma CLI
  - Usar GetFunctionConfiguration com uma CLI
  - Usar GetPolicy com uma CLI
  - Usar GetProvisionedConcurrencyConfig com uma CLI
  - Use Invoke com um AWS SDK ou CLI
  - Use ListFunctionscom um AWS SDK ou CLI
  - Usar ListProvisionedConcurrencyConfigs com uma CLI
  - Usar ListTags com uma CLI
  - Usar ListVersionsByFunction com uma CLI
  - Usar PublishVersion com uma CLI
  - Usar PutFunctionConcurrency com uma CLI
  - Usar PutProvisionedConcurrencyConfig com uma CLI
  - Usar RemovePermission com uma CLI
  - Usar TagResource com uma CLI
  - Usar UntagResource com uma CLI
  - Usar UpdateAlias com uma CLI
  - Use UpdateFunctionCodecom um AWS SDK ou CLI
  - Use UpdateFunctionConfigurationcom um AWS SDK ou CLI
- Cenários para o uso do Lambda AWS SDKs

- Confirme automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK
- Migre automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK
- Crie uma API REST do API Gateway para rastrear COVID-19 dados
- Criar uma API REST de biblioteca de empréstimos
- Criar uma aplicação de mensageiro com o Step Functions
- Criar uma API REST com integração de proxy de função
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Criar uma aplicação de chat websocket com o API Gateway
- Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Invocar uma função do Lambda em um navegador
- Como transformar dados para sua aplicação com o S3 Object Lambda
- Usar o API Gateway para invocar uma função do Lambda
- Usar Step Functions para invocar funções do Lambda
- Usar eventos programados para chamar uma função do Lambda
- Use a API do Amazon Neptune para desenvolver uma função do Lambda que consulta dados de gráficos
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor
- Grave dados de atividades personalizados com uma função Lambda após a autenticação do usuário do Amazon Cognito usando um AWS SDK

- Exemplos de tecnologia sem servidor para o Lambda
  - Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda
  - Invocar uma função do Lambda em um trigger do Kinesis
  - Invocar uma função do Lambda em um gatilho do DynamoDB
  - Invocar uma função do Lambda de um acionador do Amazon DocumentDB
  - Invocar uma função do Lambda em um gatinho do Amazon MSK

- Invocar uma função do Lambda em um acionador do Amazon S3
- Invocar uma função do Lambda em um acionador do Amazon SNS
- Invocar uma função do Lambda em um trigger do Amazon SQS
- Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis
- Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB
- Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS
- AWS contribuições da comunidade para o Lambda
  - Compilar e testar uma aplicação com tecnologia sem servidor

Exemplos básicos de uso do Lambda AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Lambda com AWS SDKs.

Exemplos

- Olá, Lambda
- Aprenda os conceitos básicos do Lambda com um AWS SDK
- Ações para Lambda usando AWS SDKs
  - Usar CreateAlias com uma CLI
  - Use CreateFunctioncom um AWS SDK ou CLI
  - Usar DeleteAlias com uma CLI
  - Use DeleteFunctioncom um AWS SDK ou CLI
  - Usar DeleteFunctionConcurrency com uma CLI
  - Usar DeleteProvisionedConcurrencyConfig com uma CLI
  - Usar GetAccountSettings com uma CLI
  - Usar GetAlias com uma CLI
  - Use GetFunctioncom um AWS SDK ou CLI
  - Usar GetFunctionConcurrency com uma CLI
  - Usar GetFunctionConfiguration com uma CLI
  - Usar GetPolicy com uma CLI
  - Usar GetProvisionedConcurrencyConfig com uma CLI
  - Use Invoke com um AWS SDK ou CLI

- Use ListFunctionscom um AWS SDK ou CLI
- Usar ListProvisionedConcurrencyConfigs com uma CLI
- Usar ListTags com uma CLI
- Usar ListVersionsByFunction com uma CLI
- Usar PublishVersion com uma CLI
- Usar PutFunctionConcurrency com uma CLI
- Usar PutProvisionedConcurrencyConfig com uma CLI
- Usar RemovePermission com uma CLI
- Usar TagResource com uma CLI
- Usar UntagResource com uma CLI
- Usar UpdateAlias com uma CLI
- Use UpdateFunctionCodecom um AWS SDK ou CLI
- Use UpdateFunctionConfigurationcom um AWS SDK ou CLI

Olá, Lambda

Os exemplos de código a seguir mostram como começar a usar o Lambda.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace LambdaActions;

using Amazon.Lambda;

public class HelloLambda
{
    static async Task Main(string[] args)
    {
```

```
        var lambdaClient = new AmazonLambdaClient();

        Console.WriteLine("Hello AWS Lambda");
        Console.WriteLine("Let's get started with AWS Lambda by listing your
 existing Lambda functions:");

        var response = await lambdaClient.ListFunctionsAsync();
        response.Functions.ForEach(function =>
        {

 Console.WriteLine($"{function.FunctionName}\t{function.Description}");
        });
    }
}
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS lambda)

# Set this project's name.
project("hello_lambda")

# Set the C++ standard to use to build this target.
```

```
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                       # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_lambda.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_lambda.cpp.

```
#include <aws/core/Aws.h>
#include <aws/lambda/LambdaClient.h>
#include <aws/lambda/model/ListFunctionsRequest.h>
#include <iostream>
```

```
/*
 *  A "Hello Lambda" starter application which initializes an AWS Lambda (Lambda)
 client and lists the Lambda functions.
 *
 *  main function
 *
 *  Usage: 'hello_lambda'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Lambda::LambdaClient lambdaClient(clientConfig);
        std::vector<Aws::String> functions;
        Aws::String marker; // Used for pagination.

        do {
            Aws::Lambda::Model::ListFunctionsRequest request;
            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::Lambda::Model::ListFunctionsOutcome outcome =
 lambdaClient.ListFunctions(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Lambda::Model::ListFunctionsResult
 &listFunctionsResult = outcome.GetResult();
                std::cout << listFunctionsResult.GetFunctions().size()
                          << " lambda functions were retrieved." << std::endl;

                for (const Aws::Lambda::Model::FunctionConfiguration
 &functionConfiguration: listFunctionsResult.GetFunctions()) {
                    functions.push_back(functionConfiguration.GetFunctionName());
```

```
                    std::cout << functions.size() << "  "
                              << functionConfiguration.GetDescription() <<
 std::endl;
                    std::cout << "   "
                              <<
 Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                      functionConfiguration.GetRuntime()) << ": "
                              << functionConfiguration.GetHandler()
                              << std::endl;
                }
                marker = listFunctionsResult.GetNextMarker();
            } else {
                std::cerr << "Error with Lambda::ListFunctions. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!marker.empty());
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main
```

```
import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
)

// main uses the AWS SDK for Go (v2) to create an AWS Lambda client and list up
 to 10
// functions in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 lambdaClient := lambda.NewFromConfig(sdkConfig)

 maxItems := 10
 fmt.Printf("Let's list up to %v functions for your account.\n", maxItems)
 result, err := lambdaClient.ListFunctions(ctx, &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 if err != nil {
  fmt.Printf("Couldn't list functions for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Functions) == 0 {
  fmt.Println("You don't have any functions!")
 } else {
  for _, function := range result.Functions {
   fmt.Printf("\t%v\n", *function.FunctionName)
  }
 }
}
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " +
config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { LambdaClient, paginateListFunctions } from "@aws-sdk/client-lambda";

const client = new LambdaClient({});

export const helloLambda = async () => {
  const paginator = paginateListFunctions({ client }, {});
  const functions = [];

  for await (const page of paginator) {
    const funcNames = page.Functions.map((f) => f.FunctionName);
    functions.push(...funcNames);
  }

  console.log("Functions:");
  console.log(functions.join("\n"));
  return functions;
};
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def main():
    """
    List the Lambda functions in your AWS account.
    """
    # Create the Lambda client
    lambda_client = boto3.client("lambda")

    # Use the paginator to list the functions
    paginator = lambda_client.get_paginator("list_functions")
    response_iterator = paginator.paginate()

    print("Here are the Lambda functions in your account:")
    for page in response_iterator:
        for function in page["Functions"]:
            print(f"  {function['FunctionName']}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ListFunctionsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-lambda'
```

```
# Creates an AWS Lambda client using the default credentials and configuration
def lambda_client
  Aws::Lambda::Client.new
end

# Lists the Lambda functions in your AWS account, paginating the results if
 necessary
def list_lambda_functions
  lambda = lambda_client

  # Use a pagination iterator to list all functions
  functions = []
  lambda.list_functions.each_page do |page|
    functions.concat(page.functions)
  end

  # Print the name and ARN of each function
  functions.each do |function|
    puts "Function name: #{function.function_name}"
    puts "Function ARN: #{function.function_arn}"
    puts
  end

  puts "Total functions: #{functions.count}"
end

list_lambda_functions if __FILE__ == $PROGRAM_NAME
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para Ruby da API.

Aprenda os conceitos básicos do Lambda com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um perfil do IAM e uma função do Lambda e carregue o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.

- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

.NET

SDK para .NET

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie métodos que executem ações do Lambda.

```
namespace LambdaActions;

using Amazon.Lambda;
using Amazon.Lambda.Model;

/// <summary>
/// A class that implements AWS Lambda methods.
/// </summary>
public class LambdaWrapper
{
    private readonly IAmazonLambda _lambdaService;

    /// <summary>
    /// Constructor for the LambdaWrapper class.
    /// </summary>
    /// <param name="lambdaService">An initialized Lambda service client.</param>
    public LambdaWrapper(IAmazonLambda lambdaService)
    {
        _lambdaService = lambdaService;
    }

    /// <summary>
    /// Creates a new Lambda function.
    /// </summary>
```

```
    /// <param name="functionName">The name of the function.</param>
    /// <param name="s3Bucket">The Amazon Simple Storage Service (Amazon S3)
    /// bucket where the zip file containing the code is located.</param>
    /// <param name="s3Key">The Amazon S3 key of the zip file.</param>
    /// <param name="role">The Amazon Resource Name (ARN) of a role with the
    /// appropriate Lambda permissions.</param>
    /// <param name="handler">The name of the handler function.</param>
    /// <returns>The Amazon Resource Name (ARN) of the newly created
    /// Lambda function.</returns>
    public async Task<string> CreateLambdaFunctionAsync(
        string functionName,
        string s3Bucket,
        string s3Key,
        string role,
        string handler)
    {
        // Defines the location for the function code.
        // S3Bucket - The S3 bucket where the file containing
        //            the source code is stored.
        // S3Key    - The name of the file containing the code.
        var functionCode = new FunctionCode
        {
            S3Bucket = s3Bucket,
            S3Key = s3Key,
        };

        var createFunctionRequest = new CreateFunctionRequest
        {
            FunctionName = functionName,
            Description = "Created by the Lambda .NET API",
            Code = functionCode,
            Handler = handler,
            Runtime = Runtime.Dotnet6,
            Role = role,
        };

        var reponse = await
_lambdaService.CreateFunctionAsync(createFunctionRequest);
        return reponse.FunctionArn;
    }


    /// <summary>
    /// Delete an AWS Lambda function.
```

```
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// delete.</param>
    /// <returns>A Boolean value that indicates the success of the action.</
returns>
    public async Task<bool> DeleteFunctionAsync(string functionName)
    {
        var request = new DeleteFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.DeleteFunctionAsync(request);

        // A return value of NoContent means that the request was processed.
        // In this case, the function was deleted, and the return value
        // is intentionally blank.
        return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
    }


    /// <summary>
    /// Gets information about a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function for
    /// which to retrieve information.</param>
    /// <returns>Async Task.</returns>
    public async Task<FunctionConfiguration> GetFunctionAsync(string
 functionName)
    {
        var functionRequest = new GetFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.GetFunctionAsync(functionRequest);
        return response.Configuration;
    }


    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
```

```
    /// invoke.</param
    /// <param name="parameters">The parameter values that will be passed to the
function.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
        };

        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue =
System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }


    /// <summary>
    /// Get a list of Lambda functions.
    /// </summary>
    /// <returns>A list of FunctionConfiguration objects.</returns>
    public async Task<List<FunctionConfiguration>> ListFunctionsAsync()
    {
        var functionList = new List<FunctionConfiguration>();

        var functionPaginator =
            _lambdaService.Paginators.ListFunctions(new ListFunctionsRequest());
        await foreach (var function in functionPaginator.Functions)
        {
            functionList.Add(function);
        }

        return functionList;
    }


    /// <summary>
    /// Update an existing Lambda function.
```

```
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to update.</
param>
    /// <param name="bucketName">The bucket where the zip file containing
    /// the Lambda function code is stored.</param>
    /// <param name="key">The key name of the source code file.</param>
    /// <returns>Async Task.</returns>
    public async Task UpdateFunctionCodeAsync(
        string functionName,
        string bucketName,
        string key)
    {
        var functionCodeRequest = new UpdateFunctionCodeRequest
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await
 _lambdaService.UpdateFunctionCodeAsync(functionCodeRequest);
        Console.WriteLine($"The Function was last modified at
 {response.LastModified}.");
    }


    /// <summary>
    /// Update the code of a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function to update.</param>
    /// <param name="functionHandler">The code that performs the function's
 actions.</param>
    /// <param name="environmentVariables">A dictionary of environment
 variables.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateFunctionConfigurationAsync(
        string functionName,
        string functionHandler,
        Dictionary<string, string> environmentVariables)
    {
        var request = new UpdateFunctionConfigurationRequest
        {
            Handler = functionHandler,
```

```
            FunctionName = functionName,
            Environment = new Amazon.Lambda.Model.Environment { Variables =
 environmentVariables },
        };

        var response = await
 _lambdaService.UpdateFunctionConfigurationAsync(request);

        Console.WriteLine(response.LastModified);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


}
```

Crie uma função que execute o cenário.

```
global using System.Threading.Tasks;
global using Amazon.IdentityManagement;
global using Amazon.Lambda;
global using LambdaActions;
global using LambdaScenarioCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


using Amazon.Lambda.Model;
using Microsoft.Extensions.Configuration;

namespace LambdaBasics;

public class LambdaBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
```

```
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonLambda>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<LambdaWrapper>()
            .AddTransient<LambdaRoleWrapper>()
            .AddTransient<UIWrapper>()
        )
        .Build();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();


        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<LambdaBasics>();

        var lambdaWrapper = host.Services.GetRequiredService<LambdaWrapper>();
        var lambdaRoleWrapper =
host.Services.GetRequiredService<LambdaRoleWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        string functionName = configuration["FunctionName"]!;
        string roleName = configuration["RoleName"]!;
        string policyDocument = "{" +
            " \"Version\": \"2012-10-17\"," +
            " \"Statement\": [ " +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Principal\": {" +
            "            \"Service\": \"lambda.amazonaws.com\" " +
            "    }," +
            "        \"Action\": \"sts:AssumeRole\" " +
```

```
            "    }" +
            "]" +
        "}";

        var incrementHandler = configuration["IncrementHandler"];
        var calculatorHandler = configuration["CalculatorHandler"];
        var bucketName = configuration["BucketName"];
        var incrementKey = configuration["IncrementKey"];
        var calculatorKey = configuration["CalculatorKey"];
        var policyArn = configuration["PolicyArn"];

        uiWrapper.DisplayLambdaBasicsOverview();

        // Create the policy to use with the AWS Lambda functions and then attach
the
        // policy to a new role.
        var roleArn = await lambdaRoleWrapper.CreateLambdaRoleAsync(roleName,
policyDocument);

        Console.WriteLine("Waiting for role to become active.");
        uiWrapper.WaitABit(15, "Wait until the role is active before trying to
use it.");

        // Attach the appropriate AWS Identity and Access Management (IAM) role
policy to the new role.
        var success = await
lambdaRoleWrapper.AttachLambdaRolePolicyAsync(policyArn, roleName);
        uiWrapper.WaitABit(10, "Allow time for the IAM policy to be attached to
the role.");

        // Create the Lambda function using a zip file stored in an Amazon Simple
Storage Service
        // (Amazon S3) bucket.
        uiWrapper.DisplayTitle("Create Lambda Function");
        Console.WriteLine($"Creating the AWS Lambda function: {functionName}.");
        var lambdaArn = await lambdaWrapper.CreateLambdaFunctionAsync(
            functionName,
            bucketName,
            incrementKey,
            roleArn,
            incrementHandler);

        Console.WriteLine("Waiting for the new function to be available.");
        Console.WriteLine($"The AWS Lambda ARN is {lambdaArn}");
```

```
        // Get the Lambda function.
        Console.WriteLine($"Getting the {functionName} AWS Lambda function.");
        FunctionConfiguration config;
        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.State != State.Active);

        Console.WriteLine($"\nThe function, {functionName} has been created.");
        Console.WriteLine($"The runtime of this Lambda function is
{config.Runtime}.");

        uiWrapper.PressEnter();

        // List the Lambda functions.
        uiWrapper.DisplayTitle("Listing all Lambda functions.");
        var functions = await lambdaWrapper.ListFunctionsAsync();
        DisplayFunctionList(functions);

        uiWrapper.DisplayTitle("Invoke increment function");
        Console.WriteLine("Now that it has been created, invoke the Lambda
increment function.");
        string? value;
        do
        {
            Console.Write("Enter a value to increment: ");
            value = Console.ReadLine();
        }
        while (string.IsNullOrEmpty(value));

        string functionParameters = "{" +
            "\"action\": \"increment\", " +
            "\"x\": \"" + value + "\"" +
        "}";
        var answer = await lambdaWrapper.InvokeFunctionAsync(functionName,
functionParameters);
        Console.WriteLine($"{value} + 1 = {answer}.");

        uiWrapper.DisplayTitle("Update function");
        Console.WriteLine("Now update the Lambda function code.");
```

```
        await lambdaWrapper.UpdateFunctionCodeAsync(functionName, bucketName,
calculatorKey);

        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

        await lambdaWrapper.UpdateFunctionConfigurationAsync(
            functionName,
            calculatorHandler,
            new Dictionary<string, string> { { "LOG_LEVEL", "DEBUG" } });

        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

        uiWrapper.DisplayTitle("Call updated function");
        Console.WriteLine("Now call the updated function...");

        bool done = false;

        do
        {
            string? opSelected;

            Console.WriteLine("Select the operation to perform:");
            Console.WriteLine("\t1. add");
            Console.WriteLine("\t2. subtract");
            Console.WriteLine("\t3. multiply");
            Console.WriteLine("\t4. divide");
            Console.WriteLine("\tOr enter \"q\" to quit.");
            Console.WriteLine("Enter the number (1, 2, 3, 4, or q) of the
operation you want to perform: ");
            do
            {
                Console.Write("Your choice? ");
                opSelected = Console.ReadLine();
            }
```

```
            while (opSelected == string.Empty);

            var operation = (opSelected) switch
            {
                "1" => "add",
                "2" => "subtract",
                "3" => "multiply",
                "4" => "divide",
                "q" => "quit",
                _ => "add",
            };

            if (operation == "quit")
            {
                done = true;
            }
            else
            {
                // Get two numbers and an action from the user.
                value = string.Empty;
                do
                {
                    Console.Write("Enter the first value: ");
                    value = Console.ReadLine();
                }
                while (value == string.Empty);

                string? value2;
                do
                {
                    Console.Write("Enter a second value: ");
                    value2 = Console.ReadLine();
                }
                while (value2 == string.Empty);

                functionParameters = "{" +
                    "\"action\": \"" + operation + "\", " +
                    "\"x\": \"" + value + "\"," +
                    "\"y\": \"" + value2 + "\"" +
                "}";

                answer = await lambdaWrapper.InvokeFunctionAsync(functionName,
functionParameters);
```

```
                Console.WriteLine($"The answer when we {operation} the two
numbers is: {answer}.");
            }

            uiWrapper.PressEnter();
        } while (!done);

        // Delete the function created earlier.

        uiWrapper.DisplayTitle("Clean up resources");
        // Detach the IAM policy from the IAM role.
        Console.WriteLine("First detach the IAM policy from the role.");
        success = await lambdaRoleWrapper.DetachLambdaRolePolicyAsync(policyArn,
roleName);
        uiWrapper.WaitABit(15, "Let's wait for the policy to be fully detached
from the role.");

        Console.WriteLine("Delete the AWS Lambda function.");
        success = await lambdaWrapper.DeleteFunctionAsync(functionName);
        if (success)
        {
            Console.WriteLine($"The {functionName} function was deleted.");
        }
        else
        {
            Console.WriteLine($"Could not remove the function {functionName}");
        }

        // Now delete the IAM role created for use with the functions
        // created by the application.
        Console.WriteLine("Now we can delete the role that we created.");
        success = await lambdaRoleWrapper.DeleteLambdaRoleAsync(roleName);
        if (success)
        {
            Console.WriteLine("The role has been successfully removed.");
        }
        else
        {
            Console.WriteLine("Couldn't delete the role.");
        }

        Console.WriteLine("The Lambda Scenario is now complete.");
        uiWrapper.PressEnter();
```

```
        // Displays a formatted list of existing functions returned by the
        // LambdaMethods.ListFunctions.
        void DisplayFunctionList(List<FunctionConfiguration> functions)
        {
            functions.ForEach(functionConfig =>
            {

 Console.WriteLine($"{functionConfig.FunctionName}\t{functionConfig.Description}");
            });
        }
    }
}


namespace LambdaActions;

using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

public class LambdaRoleWrapper
{
    private readonly IAmazonIdentityManagementService _lambdaRoleService;

    public LambdaRoleWrapper(IAmazonIdentityManagementService lambdaRoleService)
    {
        _lambdaRoleService = lambdaRoleService;
    }

    /// <summary>
    /// Attach an AWS Identity and Access Management (IAM) role policy to the
    /// IAM role to be assumed by the AWS Lambda functions created for the
 scenario.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
 policy.</param>
    /// <param name="roleName">The name of the IAM role to attach the IAM policy
 to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachLambdaRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _lambdaRoleService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest { PolicyArn = policyArn, RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }

    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to create.</param>
    /// <param name="policyDocument">The policy document for the new IAM role.</
param>
    /// <returns>A string representing the ARN for newly created role.</returns>
    public async Task<string> CreateLambdaRoleAsync(string roleName, string
 policyDocument)
    {
        var request = new CreateRoleRequest
        {
            AssumeRolePolicyDocument = policyDocument,
            RoleName = roleName,
        };

        var response = await _lambdaRoleService.CreateRoleAsync(request);
        return response.Role.Arn;
    }

    /// <summary>
    /// Deletes an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> DeleteLambdaRoleAsync(string roleName)
    {
        var request = new DeleteRoleRequest
        {
            RoleName = roleName,
        };

        var response = await _lambdaRoleService.DeleteRoleAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    public async Task<bool> DetachLambdaRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _lambdaRoleService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest { PolicyArn = policyArn, RoleName = roleName });
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}


namespace LambdaScenarioCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the AWS Lambda Basics scenario.
    /// </summary>
    public void DisplayLambdaBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to AWS Lambda Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an AWS Identity and Access Management
 (IAM) role that will be assumed by the functions we create.");
        Console.WriteLine("\t2. Attaches an IAM role policy that has Lambda
 permissions.");
        Console.WriteLine("\t3. Creates a Lambda function that increments the
 value passed to it.");
        Console.WriteLine("\t4. Calls the increment function and passes a
 value.");
        Console.WriteLine("\t5. Updates the code so that the function is a simple
 calculator.");
        Console.WriteLine("\t6. Calls the calculator function with the values
 entered.");
        Console.WriteLine("\t7. Deletes the Lambda function.");
        Console.WriteLine("\t7. Detaches the IAM role policy.");
        Console.WriteLine("\t8. Deletes the IAM role.");
        PressEnter();
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
```

```
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
```

```
    }
}
```

Defina um manipulador do Lambda que aumente um número.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace LambdaIncrement;

public class Function
{

    /// <summary>
    /// A simple function increments the integer parameter.
    /// </summary>
    /// <param name="input">A JSON string containing an action, which must be
    /// "increment" and a string representing the value to increment.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</
param>
    /// <returns>A string representing the incremented value of the parameter.</
returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
 context)
    {
        if (input["action"] == "increment")
        {
            int inputValue = Convert.ToInt32(input["x"]);
            return inputValue + 1;
        }
        else
        {
            return 0;
        }
    }
}
```

Defina um segundo manipulador do Lambda que faça operações aritméticas.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace LambdaCalculator;

public class Function
{

    /// <summary>
    /// A simple function that takes two number in string format and performs
    /// the requested arithmetic function.
    /// </summary>
    /// <param name="input">JSON data containing an action, and x and y values.
    /// Valid actions include: add, subtract, multiply, and divide.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</
param>
    /// <returns>A string representing the results of the calculation.</returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
 context)
    {
        var action = input["action"];
        int x = Convert.ToInt32(input["x"]);
        int y = Convert.ToInt32(input["y"]);
        int result;
        switch (action)
        {
            case "add":
                result = x + y;
                break;
            case "subtract":
                result = x - y;
                break;
            case "multiply":
```

```
                result = x * y;
                break;
            case "divide":
                if (y == 0)
                {
                    Console.Error.WriteLine("Divide by zero error.");
                    result = 0;
                }
                else
                    result = x / y;
                break;
            default:
                Console.Error.WriteLine($"{action} is not a valid operation.");
                result = 0;
                break;
        }
        return result;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get started with functions scenario.
/*!
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Lambda::getStartedWithFunctionsScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::Lambda::LambdaClient client(clientConfig);

    // 1. Create an AWS Identity and Access Management (IAM) role for Lambda
 function.
    Aws::String roleArn;
    if (!getIamRoleArn(roleArn, clientConfig)) {
        return false;
    }

    // 2. Create a Lambda function.
    int seconds = 0;
    do {
        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#if USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);

        // Assume the AWS Lambda function was built in Docker with same
 architecture
        // as this code.
#if  defined(__x86_64__)
```

```
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif
        request.SetRole(roleArn);
        request.SetHandler(LAMBDA_HANDLER_NAME);
        request.SetPublish(true);
        Aws::Lambda::Model::FunctionCode code;
        std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();

        code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
 buffer.str().c_str(),
                                               buffer.str().length()));
        request.SetCode(code);

        Aws::Lambda::Model::CreateFunctionOutcome outcome =
 client.CreateFunction(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda function was successfully created. " <<
 seconds
```

```
                      << " seconds elapsed." << std::endl;
            break;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::Lambda::LambdaErrors::INVALID_PARAMETER_VALUE &&
                 outcome.GetError().GetMessage().find("role") >= 0) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout
                        << "Waiting for the IAM role to become available as a
CreateFunction parameter. "
                        << seconds
                        << " seconds elapsed." << std::endl;

                std::cout << outcome.GetError().GetMessage() << std::endl;
            }
        }
        else {
            std::cerr << "Error with CreateFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (60 > seconds);

    std::cout << "The current Lambda function increments 1 by an input." <<
std::endl;

    // 3.  Invoke the Lambda function.
    {
        int increment = askQuestionForInt("Enter an increment integer: ");

        Aws::Lambda::Model::InvokeResult invokeResult;
        Aws::Utils::Json::JsonValue jsonPayload;
        jsonPayload.WithString("action", "increment");
        jsonPayload.WithInteger("number", increment);
        if (invokeLambdaFunction(jsonPayload, Aws::Lambda::Model::LogType::Tail,
                                 invokeResult, client)) {
            Aws::Utils::Json::JsonValue jsonValue(invokeResult.GetPayload());
            Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
                    jsonValue.View().GetAllObjects();
            auto iter = values.find("result");
```

```
            if (iter != values.end() && iter->second.IsIntegerType()) {
                {
                    std::cout << INCREMENT_RESUlT_PREFIX
                              << iter->second.AsInteger() << std::endl;
                }
            }
            else {
                std::cout << "There was an error in execution. Here is the log."
                          << std::endl;
                Aws::Utils::ByteBuffer buffer =
 Aws::Utils::HashingUtils::Base64Decode(
                        invokeResult.GetLogResult());
                std::cout << "With log " << buffer.GetUnderlyingData() <<
 std::endl;
            }
        }
    }

    std::cout
            << "The Lambda function will now be updated with new code. Press
 return to continue, ";
    Aws::String answer;
    std::getline(std::cin, answer);

    // 4.  Update the Lambda function code.
    {
        Aws::Lambda::Model::UpdateFunctionCodeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }
```

```
        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
                Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                        buffer.str().length()));
        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
client.UpdateFunctionCode(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." <<
std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
            << "This function uses an environment variable to control the logging
level."
            << std::endl;
    std::cout
            << "UpdateFunctionConfiguration will be used to set the LOG_LEVEL to
DEBUG."
            << std::endl;
    seconds = 0;

    // 5.  Update the Lambda function configuration.
    do {
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        Aws::Lambda::Model::Environment environment;
        environment.AddVariables("LOG_LEVEL", "DEBUG");
        request.SetEnvironment(environment);
```

```
        Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
 client.UpdateFunctionConfiguration(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda configuration was successfully updated."
                      << std::endl;
            break;
        }

            // RESOURCE_IN_USE: function code update not completed.
        else if (outcome.GetError().GetErrorType() !=
                 Aws::Lambda::LambdaErrors::RESOURCE_IN_USE) {
            if ((seconds % 10) == 0) { // Log status every 10 seconds.
                std::cout << "Lambda function update in progress . After " <<
 seconds
                          << " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (0 < seconds);

    if (0 > seconds) {
        std::cerr << "Function failed to become active." << std::endl;
    }
    else {
        std::cout << "Updated function active after " << seconds << " seconds."
                  << std::endl;
    }

    std::cout
            << "\nThe new code applies an arithmetic operator to two variables, x
 an y."
            << std::endl;
    std::vector<Aws::String> operators = {"plus", "minus", "times", "divided-
by"};
    for (size_t i = 0; i < operators.size(); ++i) {
        std::cout << "   " << i + 1 << " " << operators[i] << std::endl;
    }
```

```
   // 6.  Invoke the updated Lambda function.
   do {
       int operatorIndex = askQuestionForIntRange("Select an operator index 1 -
4 ", 1,
                                                  4);
       int x = askQuestionForInt("Enter an integer for the x value ");
       int y = askQuestionForInt("Enter an integer for the y value ");

       Aws::Utils::Json::JsonValue calculateJsonPayload;
       calculateJsonPayload.WithString("action", operators[operatorIndex - 1]);
       calculateJsonPayload.WithInteger("x", x);
       calculateJsonPayload.WithInteger("y", y);
       Aws::Lambda::Model::InvokeResult calculatedResult;
       if (invokeLambdaFunction(calculateJsonPayload,
                                Aws::Lambda::Model::LogType::Tail,
                                calculatedResult, client)) {
           Aws::Utils::Json::JsonValue jsonValue(calculatedResult.GetPayload());
           Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
                   jsonValue.View().GetAllObjects();
           auto iter = values.find("result");
           if (iter != values.end() && iter->second.IsIntegerType()) {
               std::cout << ARITHMETIC_RESUlT_PREFIX << x << " "
                         << operators[operatorIndex - 1] << " "
                         << y << " is " << iter->second.AsInteger() <<
std::endl;
           }
           else if (iter != values.end() && iter->second.IsFloatingPointType())
{
               std::cout << ARITHMETIC_RESUlT_PREFIX << x << " "
                         << operators[operatorIndex - 1] << " "
                         << y << " is " << iter->second.AsDouble() << std::endl;
           }
           else {
               std::cout << "There was an error in execution. Here is the log."
                         << std::endl;
               Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
                       calculatedResult.GetLogResult());
               std::cout << "With log " << buffer.GetUnderlyingData() <<
std::endl;
           }
       }
```

```
        answer = askQuestion("Would you like to try another operation? (y/n) ");
    } while (answer == "y");

    std::cout
            << "A list of the lambda functions will be retrieved. Press return to
continue, ";
    std::getline(std::cin, answer);

    // 7.  List the Lambda functions.

    std::vector<Aws::String> functions;
    Aws::String marker;

    do {
        Aws::Lambda::Model::ListFunctionsRequest request;
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
            std::cout << result.GetFunctions().size()
                      << " lambda functions were retrieved." << std::endl;

            for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
                functions.push_back(functionConfiguration.GetFunctionName());
                std::cout << functions.size() << "  "
                          << functionConfiguration.GetDescription() << std::endl;
                std::cout << "   "
                          <<
Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                  functionConfiguration.GetRuntime()) << ": "
                          << functionConfiguration.GetHandler()
                          << std::endl;
            }
            marker = result.GetNextMarker();
        }
        else {
            std::cerr << "Error with Lambda::ListFunctions. "
```

```
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    // 8.  Get a Lambda function.
    if (!functions.empty()) {
        std::stringstream question;
        question << "Choose a function to retrieve between 1 and " <<
functions.size()
                 << " ";
        int functionIndex = askQuestionForIntRange(question.str(), 1,

static_cast<int>(functions.size()));

        Aws::String functionName = functions[functionIndex - 1];

        Aws::Lambda::Model::GetFunctionRequest request;
        request.SetFunctionName(functionName);

        Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

        if (outcome.IsSuccess()) {
            std::cout << "Function retrieve.\n" <<

outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
                      << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::GetFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout << "The resources will be deleted. Press return to continue, ";
    std::getline(std::cin, answer);

    // 9.  Delete the Lambda function.
    bool result = deleteLambdaFunction(client);

    // 10. Delete the IAM role.
    return result && deleteIamRole(clientConfig);
```

```
}

//! Routine which invokes a Lambda function and returns the result.
/*!
 \param jsonPayload: Payload for invoke function.
 \param logType: Log type setting for invoke function.
 \param invokeResult: InvokeResult object to receive the result.
 \param client: Lambda client.
 \return bool: Successful completion.
 */
bool
AwsDoc::Lambda::invokeLambdaFunction(const Aws::Utils::Json::JsonValue
 &jsonPayload,
                                      Aws::Lambda::Model::LogType logType,
                                      Aws::Lambda::Model::InvokeResult
 &invokeResult,
                                      const Aws::Lambda::LambdaClient &client) {
    int seconds = 0;
    bool result = false;
    /*
     * In this example, the Invoke function can be called before recently created
 resources are
     * available.  The Invoke function is called repeatedly until the resources
 are
     * available.
     */
    do {
        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload =
 Aws::MakeShared<Aws::StringStream>(
                "FunctionTest");
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
            invokeResult = std::move(outcome.GetResult());
            result = true;
            break;
        }
```

```
            // ACCESS_DENIED: because the role is not available yet.
            // RESOURCE_CONFLICT: because the Lambda function is being created or
 updated.
        else if ((outcome.GetError().GetErrorType() ==
                  Aws::Lambda::LambdaErrors::ACCESS_DENIED) ||
                 (outcome.GetError().GetErrorType() ==
                  Aws::Lambda::LambdaErrors::RESOURCE_CONFLICT)) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout << "Waiting for the invoke api to be available, status
 " <<
                          ((outcome.GetError().GetErrorType() ==
                            Aws::Lambda::LambdaErrors::ACCESS_DENIED ?
                            "ACCESS_DENIED" : "RESOURCE_CONFLICT")) << ". " <<
 seconds
                          << " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error with Lambda::InvokeRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            break;
        }
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (seconds < 60);

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um cenário interativo para mostrar como começar a usar funções do Lambda.

```
import (
 "archive/zip"
 "bytes"
 "context"
 "encoding/base64"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "os"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 iamtypes "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/lambda/actions"
)

// GetStartedFunctionsScenario shows you how to use AWS Lambda to perform the
 following
// actions:
//
//  1. Create an AWS Identity and Access Management (IAM) role and Lambda
 function, then upload handler code.
//  2. Invoke the function with a single parameter and get results.
//  3. Update the function code and configure with an environment variable.
```

```
//  4. Invoke the function with new parameters and get results. Display the
 returned execution log.
//  5. List the functions for your account, then clean up resources.
type GetStartedFunctionsScenario struct {
 sdkConfig       aws.Config
 functionWrapper actions.FunctionWrapper
 questioner      demotools.IQuestioner
 helper          IScenarioHelper
 isTestRun       bool
}

// NewGetStartedFunctionsScenario constructs a GetStartedFunctionsScenario
 instance from a configuration.
// It uses the specified config to get a Lambda client and create wrappers for
 the actions
// used in the scenario.
func NewGetStartedFunctionsScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedFunctionsScenario {
 lambdaClient := lambda.NewFromConfig(sdkConfig)
 return GetStartedFunctionsScenario{
  sdkConfig:       sdkConfig,
  functionWrapper: actions.FunctionWrapper{LambdaClient: lambdaClient},
  questioner:      questioner,
  helper:          helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedFunctionsScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Lambda get started with functions demo.")
 log.Println(strings.Repeat("-", 88))

 role := scenario.GetOrCreateRole(ctx)
 funcName := scenario.CreateFunction(ctx, role)
 scenario.InvokeIncrement(ctx, funcName)
 scenario.UpdateFunction(ctx, funcName)
```

```
 scenario.InvokeCalculator(ctx, funcName)
 scenario.ListFunctions(ctx)
 scenario.Cleanup(ctx, role, funcName)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// GetOrCreateRole checks whether the specified role exists and returns it if it
 does.
// Otherwise, a role is created that specifies Lambda as a trusted principal.
// The AWSLambdaBasicExecutionRole managed policy is attached to the role and the
 role
// is returned.
func (scenario GetStartedFunctionsScenario) GetOrCreateRole(ctx context.Context)
 *iamtypes.Role {
 var role *iamtypes.Role
 iamClient := iam.NewFromConfig(scenario.sdkConfig)
 log.Println("First, we need an IAM role that Lambda can assume.")
 roleName := scenario.questioner.Ask("Enter a name for the role:",
 demotools.NotEmpty{})
 getOutput, err := iamClient.GetRole(ctx, &iam.GetRoleInput{
  RoleName: aws.String(roleName)})
 if err != nil {
  var noSuch *iamtypes.NoSuchEntityException
  if errors.As(err, &noSuch) {
   log.Printf("Role %v doesn't exist. Creating it....\n", roleName)
  } else {
   log.Panicf("Couldn't check whether role %v exists. Here's why: %v\n",
    roleName, err)
  }
 } else {
  role = getOutput.Role
  log.Printf("Found role %v.\n", *role.RoleName)
 }
 if role == nil {
  trustPolicy := PolicyDocument{
   Version: "2012-10-17",
   Statement: []PolicyStatement{{
    Effect:    "Allow",
    Principal: map[string]string{"Service": "lambda.amazonaws.com"},
    Action:    []string{"sts:AssumeRole"},
   }},
```

```
  }
  policyArn := "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
  createOutput, err := iamClient.CreateRole(ctx, &iam.CreateRoleInput{
   AssumeRolePolicyDocument: aws.String(trustPolicy.String()),
   RoleName:                 aws.String(roleName),
  })
  if err != nil {
   log.Panicf("Couldn't create role %v. Here's why: %v\n", roleName, err)
  }
  role = createOutput.Role
  _, err = iamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
   PolicyArn: aws.String(policyArn),
   RoleName:  aws.String(roleName),
  })
  if err != nil {
   log.Panicf("Couldn't attach a policy to role %v. Here's why: %v\n", roleName,
 err)
  }
  log.Printf("Created role %v.\n", *role.RoleName)
  log.Println("Let's give AWS a few seconds to propagate resources...")
  scenario.helper.Pause(10)
 }
 log.Println(strings.Repeat("-", 88))
 return role
}

// CreateFunction creates a Lambda function and uploads a handler written in
 Python.
// The code for the Python handler is packaged as a []byte in .zip format.
func (scenario GetStartedFunctionsScenario) CreateFunction(ctx context.Context,
 role *iamtypes.Role) string {
 log.Println("Let's create a function that increments a number.\n" +
  "The function uses the 'lambda_handler_basic.py' script found in the \n" +
  "'handlers' directory of this project.")
 funcName := scenario.questioner.Ask("Enter a name for the Lambda function:",
 demotools.NotEmpty{})
 zipPackage := scenario.helper.CreateDeploymentPackage("lambda_handler_basic.py",
 fmt.Sprintf("%v.py", funcName))
 log.Printf("Creating function %v and waiting for it to be ready.", funcName)
 funcState := scenario.functionWrapper.CreateFunction(ctx, funcName,
 fmt.Sprintf("%v.lambda_handler", funcName),
  role.Arn, zipPackage)
 log.Printf("Your function is %v.", funcState)
 log.Println(strings.Repeat("-", 88))
```

```
 return funcName
}

// InvokeIncrement invokes a Lambda function that increments a number. The
 function
// parameters are contained in a Go struct that is used to serialize the
 parameters to
// a JSON payload that is passed to the function.
// The result payload is deserialized into a Go struct that contains an int
 value.
func (scenario GetStartedFunctionsScenario) InvokeIncrement(ctx context.Context,
 funcName string) {
 parameters := actions.IncrementParameters{Action: "increment"}
 log.Println("Let's invoke our function. This function increments a number.")
 parameters.Number = scenario.questioner.AskInt("Enter a number to increment:",
 demotools.NotEmpty{})
 log.Printf("Invoking %v with %v...\n", funcName, parameters.Number)
 invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, parameters,
 false)
 var payload actions.LambdaResultInt
 err := json.Unmarshal(invokeOutput.Payload, &payload)
 if err != nil {
  log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
   funcName, err)
 }
 log.Printf("Invoking %v with %v returned %v.\n", funcName, parameters.Number,
 payload)
 log.Println(strings.Repeat("-", 88))
}

// UpdateFunction updates the code for a Lambda function by uploading a simple
 arithmetic
// calculator written in Python. The code for the Python handler is packaged as a
// []byte in .zip format.
// After the code is updated, the configuration is also updated with a new log
// level that instructs the handler to log additional information.
func (scenario GetStartedFunctionsScenario) UpdateFunction(ctx context.Context,
 funcName string) {
 log.Println("Let's update the function to an arithmetic calculator.\n" +
  "The function uses the 'lambda_handler_calculator.py' script found in the \n" +
  "'handlers' directory of this project.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 log.Println("Creating deployment package...")
```

```
 zipPackage :=
 scenario.helper.CreateDeploymentPackage("lambda_handler_calculator.py",
  fmt.Sprintf("%v.py", funcName))
 log.Println("...and updating the Lambda function and waiting for it to be
 ready.")
 funcState := scenario.functionWrapper.UpdateFunctionCode(ctx, funcName,
 zipPackage)
 log.Printf("Updated function %v. Its current state is %v.", funcName, funcState)
 log.Println("This function uses an environment variable to control logging
 level.")
 log.Println("Let's set it to DEBUG to get the most logging.")
 scenario.functionWrapper.UpdateFunctionConfiguration(ctx, funcName,
  map[string]string{"LOG_LEVEL": "DEBUG"})
 log.Println(strings.Repeat("-", 88))
}

// InvokeCalculator invokes the Lambda calculator function. The parameters are
 stored in a
// Go struct that is used to serialize the parameters to a JSON payload. That
 payload is then passed
// to the function.
// The result payload is deserialized to a Go struct that stores the result as
 either an
// int or float32, depending on the kind of operation that was specified.
func (scenario GetStartedFunctionsScenario) InvokeCalculator(ctx context.Context,
 funcName string) {
 wantInvoke := true
 choices := []string{"plus", "minus", "times", "divided-by"}
 for wantInvoke {
  choice := scenario.questioner.AskChoice("Select an arithmetic operation:\n",
 choices)
  x := scenario.questioner.AskInt("Enter a value for x:", demotools.NotEmpty{})
  y := scenario.questioner.AskInt("Enter a value for y:", demotools.NotEmpty{})
  log.Printf("Invoking %v %v %v...", x, choices[choice], y)
  calcParameters := actions.CalculatorParameters{
   Action: choices[choice],
   X:      x,
   Y:      y,
  }
  invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, calcParameters,
 true)
  var payload any
  if choice == 3 { // divide-by results in a float.
   payload = actions.LambdaResultFloat{}
```

```
  } else {
   payload = actions.LambdaResultInt{}
  }
  err := json.Unmarshal(invokeOutput.Payload, &payload)
  if err != nil {
   log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
    funcName, err)
  }
  log.Printf("Invoking %v with %v %v %v returned %v.\n", funcName,
   calcParameters.X, calcParameters.Action, calcParameters.Y, payload)
  scenario.questioner.Ask("Press Enter to see the logs from the call.")
  logRes, err := base64.StdEncoding.DecodeString(*invokeOutput.LogResult)
  if err != nil {
   log.Panicf("Couldn't decode log result. Here's why: %v\n", err)
  }
  log.Println(string(logRes))
  wantInvoke = scenario.questioner.AskBool("Do you want to calculate again? (y/
n)", "y")
 }
 log.Println(strings.Repeat("-", 88))
}

// ListFunctions lists up to the specified number of functions for your account.
func (scenario GetStartedFunctionsScenario) ListFunctions(ctx context.Context) {
 count := scenario.questioner.AskInt(
  "Let's list functions for your account. How many do you want to see?",
 demotools.NotEmpty{})
 functions := scenario.functionWrapper.ListFunctions(ctx, count)
 log.Printf("Found %v functions:", len(functions))
 for _, function := range functions {
  log.Printf("\t%v", *function.FunctionName)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup removes the IAM and Lambda resources created by the example.
func (scenario GetStartedFunctionsScenario) Cleanup(ctx context.Context, role
 *iamtypes.Role, funcName string) {
 if scenario.questioner.AskBool("Do you want to clean up resources created for
 this example? (y/n)",
  "y") {
  iamClient := iam.NewFromConfig(scenario.sdkConfig)
  policiesOutput, err := iamClient.ListAttachedRolePolicies(ctx,
   &iam.ListAttachedRolePoliciesInput{RoleName: role.RoleName})
```

```
  if err != nil {
   log.Panicf("Couldn't get policies attached to role %v. Here's why: %v\n",
    *role.RoleName, err)
  }
  for _, policy := range policiesOutput.AttachedPolicies {
   _, err = iamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
    PolicyArn: policy.PolicyArn, RoleName: role.RoleName,
   })
   if err != nil {
    log.Panicf("Couldn't detach policy %v from role %v. Here's why: %v\n",
     *policy.PolicyArn, *role.RoleName, err)
   }
  }
  _, err = iamClient.DeleteRole(ctx, &iam.DeleteRoleInput{RoleName:
 role.RoleName})
  if err != nil {
   log.Panicf("Couldn't delete role %v. Here's why: %v\n", *role.RoleName, err)
  }
  log.Printf("Deleted role %v.\n", *role.RoleName)

  scenario.functionWrapper.DeleteFunction(ctx, funcName)
  log.Printf("Deleted function %v.\n", funcName)
 } else {
  log.Println("Okay. Don't forget to delete the resources when you're done with
 them.")
 }
}

// IScenarioHelper abstracts I/O and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 CreateDeploymentPackage(sourceFile string, destinationFile string) *bytes.Buffer
}

// ScenarioHelper lets the caller specify the path to Lambda handler functions.
type ScenarioHelper struct {
 HandlerPath string
}

// Pause waits for the specified number of seconds.
func (helper *ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}
```

```
// CreateDeploymentPackage creates an AWS Lambda deployment package from a source
 file. The
// deployment package is stored in .zip format in a bytes.Buffer. The buffer can
 be
// used to pass a []byte to Lambda when creating the function.
// The specified destinationFile is the name to give the file when it's deployed
 to Lambda.
func (helper *ScenarioHelper) CreateDeploymentPackage(sourceFile string,
 destinationFile string) *bytes.Buffer {
 var err error
 buffer := &bytes.Buffer{}
 writer := zip.NewWriter(buffer)
 zFile, err := writer.Create(destinationFile)
 if err != nil {
  log.Panicf("Couldn't create destination archive %v. Here's why: %v\n",
 destinationFile, err)
 }
 sourceBody, err := os.ReadFile(fmt.Sprintf("%v/%v", helper.HandlerPath,
 sourceFile))
 if err != nil {
  log.Panicf("Couldn't read handler source file %v. Here's why: %v\n",
   sourceFile, err)
 } else {
  _, err = zFile.Write(sourceBody)
  if err != nil {
   log.Panicf("Couldn't write handler %v to zip archive. Here's why: %v\n",
    sourceFile, err)
  }
 }
 err = writer.Close()
 if err != nil {
  log.Panicf("Couldn't close zip writer. Here's why: %v\n", err)
 }
 return buffer
}
```

Crie uma estrutura que encapsule ações individuais do Lambda.

```
import (
```

```
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}


// GetFunction gets data about the Lambda function specified by functionName.
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName
 string) types.State {
 var state types.State
 funcOutput, err := wrapper.LambdaClient.GetFunction(ctx,
 &lambda.GetFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
 } else {
  state = funcOutput.Configuration.State
 }
 return state
}



// CreateFunction creates a new Lambda function from code contained in the
 zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
```

```
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
 until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
 string, handlerName string,
 iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
  Code:         &types.FunctionCode{ZipFile: zipPackage.Bytes()},
  FunctionName: aws.String(functionName),
  Role:         iamRoleArn,
  Handler:      aws.String(handlerName),
  Publish:      true,
  Runtime:      types.RuntimePython39,
 })
 if err != nil {
  var resConflict *types.ResourceConflictException
  if errors.As(err, &resConflict) {
   log.Printf("Function %v already exists.\n", functionName)
   state = types.StateActive
  } else {
   log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
  }
 } else {
  waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}



// UpdateFunctionCode updates the code for the Lambda function specified by
 functionName.
// The existing code for the Lambda function is entirely replaced by the code in
 the
```

```
// zipPackage buffer. After the update action is called, a
 lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context,
 functionName string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
 &lambda.UpdateFunctionCodeInput{
  FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
 })
 if err != nil {
  log.Panicf("Couldn't update code for function %v. Here's why: %v\n",
 functionName, err)
 } else {
  waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}



// UpdateFunctionConfiguration updates a map of environment variables configured
 for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
 functionName string, envVars map[string]string) {
 _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
 &lambda.UpdateFunctionConfigurationInput{
  FunctionName: aws.String(functionName),
  Environment:  &types.Environment{Variables: envVars},
 })
 if err != nil {
  log.Panicf("Couldn't update configuration for %v. Here's why: %v",
 functionName, err)
 }
}
```

```
// ListFunctions lists up to maxItems functions for the account. This function
 uses a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
 []types.FunctionConfiguration {
 var functions []types.FunctionConfiguration
 paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
 &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 for paginator.HasMorePages() && len(functions) < maxItems {
  pageOutput, err := paginator.NextPage(ctx)
  if err != nil {
   log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
  }
  functions = append(functions, pageOutput.Functions...)
 }
 return functions
}



// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
 string) {
 _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
 }
}



// Invoke invokes the Lambda function specified by functionName, passing the
 parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
 tells
// Lambda to include the last few log lines in the returned result.
```

```
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
 parameters any, getLog bool) *lambda.InvokeOutput {
 logType := types.LogTypeNone
 if getLog {
  logType = types.LogTypeTail
 }
 payload, err := json.Marshal(parameters)
 if err != nil {
  log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
 }
 invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
  FunctionName: aws.String(functionName),
  LogType:      logType,
  Payload:      payload,
 })
 if err != nil {
  log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
 }
 return invokeOutput
}



// IncrementParameters is used to serialize parameters to the increment Lambda
 handler.
type IncrementParameters struct {
 Action string `json:"action"`
 Number int    `json:"number"`
}

// CalculatorParameters is used to serialize parameters to the calculator Lambda
 handler.
type CalculatorParameters struct {
 Action string `json:"action"`
 X      int    `json:"x"`
 Y      int    `json:"y"`
}

// LambdaResultInt is used to deserialize an int result from a Lambda handler.
type LambdaResultInt struct {
 Result int `json:"result"`
}
```

```
// LambdaResultFloat is used to deserialize a float32 result from a Lambda
 handler.
type LambdaResultFloat struct {
 Result float32 `json:"result"`
}
```

Defina um manipulador do Lambda que aumente um número.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)


def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
 number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the
 function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get("action")
    if action == "increment":
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Defina um segundo manipulador do Lambda que faça operações aritméticas.

```
import logging
import os


logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
 function.
ACTIONS = {
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}


def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the
 numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the
 function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)

    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
```

```
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {"result": result}
    return response
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/*
 *  Lambda function names appear as:
 *
 *  arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 *  To find this value, look at the function in the AWS Management Console.
 *
 *  Before running this Java code example, set up your development environment,
 including your credentials.
```

```
 *
 *  For more information, see this documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 *  This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

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

            Usage:
                <functionName> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\s
                role - The AWS Identity and Access Management (IAM) service role
 that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
 example.Handler::handleRequest).\s
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
 name that contains the .zip or .jar used to update the Lambda function's code.\s
                key - The Amazon S3 key name that represents the .zip or .jar
 (for example, LambdaHello-1.0-SNAPSHOT.jar).
                """;

        if (args.length != 5) {
              System.out.println(usage);
              return;
        }
```

```
        String functionName = args[0];
        String role = args[1];
        String handler = args[2];
        String bucketName = args[3];
        String key = args[4];
        LambdaClient awsLambda = LambdaClient.builder()
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Lambda Basics scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS Lambda function.");
        String funArn = createLambdaFunction(awsLambda, functionName, key,
bucketName, role, handler);
        System.out.println("The AWS Lambda ARN is " + funArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get the " + functionName + " AWS Lambda
function.");
        getFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List all AWS Lambda functions.");
        listFunctions(awsLambda);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Invoke the Lambda function.");
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it
again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
 Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();
```

```
            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which
is used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response =
awsLambda.getFunction(functionRequest);
```

```
            System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " +
config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with
the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    public static void invokeFunction(LambdaClient awsLambda, String
functionName) {
        InvokeResponse res;
        try {
```

```
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the code for an AWS Lambda function.
     *
     * @param awsLambda  the AWS Lambda client
     * @param functionName the name of the Lambda function to update
     * @param bucketName the name of the S3 bucket where the function code is
located
     * @param key the key (file name) of the function code in the S3 bucket
     * @throws LambdaException if there is an error updating the function code
     */
    public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
                .functionName(functionName)
                .publish(true)
                .s3Bucket(bucketName)
                .s3Key(key)
                .build();
```

```
            UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
            GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionConfigurationResponse> waiterResponse =
waiter
                .waitUntilFunctionUpdated(getFunctionConfigRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The last modified value is " +
response.lastModified());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the configuration of an AWS Lambda function.
     *
     * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
     * @param functionName  the name of the AWS Lambda function to update
     * @param handler       the new handler for the AWS Lambda function
     *
     * @throws LambdaException if there is an error while updating the function
configuration
     */
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which
 is used to interact with the AWS Lambda service
     * @param functionName  the name of the Lambda function to be deleted
     *
     * @throws LambdaException if an error occurs while deleting the Lambda
 function
     */
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode

- UpdateFunctionConfiguration

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma função AWS Identity and Access Management (IAM) que conceda ao Lambda permissão para gravar em registros.

```
    logger.log(`Creating role (${NAME_ROLE_LAMBDA})...`);
    const response = await createRole(NAME_ROLE_LAMBDA);

import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

Crie uma função do Lambda e carregue o código de manipulador.

```
const createFunction = async (funcName, roleArn) => {
  const client = new LambdaClient({});
```

```
  const code = await readFile(`${dirname}../functions/${funcName}.zip`);

  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

Invoque essa função com um único parâmetro e receba resultados.

```
const invoke = async (funcName, payload) => {
  const client = new LambdaClient({});
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

Atualize o código de função e configure seu ambiente do Lambda usando uma variável de ambiente.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${newFunc}.zip`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
```

```
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};

const updateFunctionConfiguration = async (funcName) => {
  const client = new LambdaClient({});
  const config = readFileSync(`${dirname}../functions/config.json`).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  const result = await client.send(command);
  await waitForFunctionUpdated({ FunctionName: funcName });
  return result;
};
```

Liste as funções para a sua conta.

```
const listFunctions = () => {
  const client = new LambdaClient({});
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

Exclua o perfil do IAM e a função do Lambda.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
```

```
  return client.send(command);
};

/**
 * @param {string} funcName
 */
const deleteFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName>
 <key>
```

```
        Where:
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that
has AWS Lambda permissions.
            handler - The fully qualified method name (for example,
example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
name that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file
(for example, LambdaHello-1.0-SNAPSHOT.jar).
            """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val functionName = args[0]
    val role = args[1]
    val handler = args[2]
    val bucketName = args[3]
    val updatedBucketName = args[4]
    val key = args[5]

    println("Creating a Lambda function named $functionName.")
    val funArn = createScFunction(functionName, bucketName, key, handler, role)
    println("The AWS Lambda ARN is $funArn")

    // Get a specific Lambda function.
    println("Getting the $functionName AWS Lambda function.")
    getFunction(functionName)

    // List the Lambda functions.
    println("Listing all AWS Lambda functions.")
    listFunctionsSc()

    // Invoke the Lambda function.
    println("*** Invoke the Lambda function.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function code.
```

```
    println("*** Update the Lambda function code.")
    updateFunctionCode(functionName, updatedBucketName, key)

    // println("*** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    updateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
```

```
        return functionResponse.functionArn.toString()
    }
}

suspend fun getFunction(functionNameVal: String) {
    val functionRequest =
        GetFunctionRequest {
            functionName = functionNameVal
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
 ${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {
    val request =
        ListFunctionsRequest {
            maxItems = 10
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            payload = byteArray
            logType = LogType.Tail
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
```

```
        println("The function payload is
 ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(
    functionNameVal: String?,
    bucketName: String?,
    key: String?,
) {
    val functionCodeRequest =
        UpdateFunctionCodeRequest {
            functionName = functionNameVal
            publish = true
            s3Bucket = bucketName
            s3Key = key
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun updateFunctionConfiguration(
    functionNameVal: String?,
    handlerVal: String?,
) {
    val configurationRequest =
        UpdateFunctionConfigurationRequest {
            functionName = functionNameVal
            handler = handlerVal
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {
```

```
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;
```

```
class GettingStartedWithLambda
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $iamService = new IAMService();
        $s3client = new S3Client($clientArgs);
        $lambdaService = new LambdaService();

        echo "First, let's create a role to run our Lambda code.\n";
        $roleName = "test-lambda-role-$uniqid";
        $rolePolicyDocument = "{
            \"Version\": \"2012-10-17\",
            \"Statement\": [
                {
                    \"Effect\": \"Allow\",
                    \"Principal\": {
                        \"Service\": \"lambda.amazonaws.com\"
                    },
                    \"Action\": \"sts:AssumeRole\"
                }
            ]
        }";
        $role = $iamService->createRole($roleName, $rolePolicyDocument);
        echo "Created role {$role['RoleName']}.\n";

        $iamService->attachRolePolicy(
            $role['RoleName'],
            "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        );
        echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.
\n";
```

```
        echo "\nNow let's create an S3 bucket and upload our Lambda code there.
\n";
        $bucketName = "amzn-s3-demo-bucket-$uniqid";
        $s3client->createBucket([
            'Bucket' => $bucketName,
        ]);
        echo "Created bucket $bucketName.\n";

        $functionName = "doc_example_lambda_$uniqid";
        $codeBasic = __DIR__ . "/lambda_handler_basic.zip";
        $handler = "lambda_handler_basic";
        $file = file_get_contents($codeBasic);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "Uploaded the Lambda code.\n";

        $createLambdaFunction = $lambdaService->createFunction($functionName,
 $role, $bucketName, $handler);
        // Wait until the function has finished being created.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['State'] == "Pending");
        echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\n";

        sleep(1);

        echo "\nOk, let's invoke that Lambda code.\n";
        $basicParams = [
            'action' => 'increment',
            'number' => 3,
        ];
        /** @var Stream $invokeFunction */
        $invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
        $result = json_decode($invokeFunction->getContents())->result;
        echo "After invoking the Lambda code with the input of
 {$basicParams['number']} we received $result.\n";

        echo "\nSince that's working, let's update the Lambda code.\n";
```

```
        $codeCalculator = "lambda_handler_calculator.zip";
        $handlerCalculator = "lambda_handler_calculator";
        echo "First, put the new code into the S3 bucket.\n";
        $file = file_get_contents($codeCalculator);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "New code uploaded.\n";

        $lambdaService->updateFunctionCode($functionName, $bucketName,
 $functionName);
        // Wait for the Lambda code to finish updating.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "New Lambda code uploaded.\n";

        $environment = [
            'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
        ];
        $lambdaService->updateFunctionConfiguration($functionName,
 $handlerCalculator, $environment);
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for
 more information.\n";

        echo "Invoke the new code with some new data.\n";
        $calculatorParams = [
            'action' => 'plus',
            'x' => 5,
            'y' => 4,
        ];
        $invokeFunction = $lambdaService->invoke($functionName,
 $calculatorParams, "Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
```

```
        echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does
 equal $result.\n";
        echo "Here's the extra debug info: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nBut what happens if you try to divide by zero?\n";
        $divZeroParams = [
            'action' => 'divide',
            'x' => 5,
            'y' => 0,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
 "Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "You get a |$result| result.\n";
        echo "And an error message: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nHere's all the Lambda functions you have in this Region:\n";
        $listLambdaFunctions = $lambdaService->listFunctions(5);
        $allLambdaFunctions = $listLambdaFunctions['Functions'];
        $next = $listLambdaFunctions->get('NextMarker');
        while ($next != false) {
            $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
            $next = $listLambdaFunctions->get('NextMarker');
            $allLambdaFunctions = array_merge($allLambdaFunctions,
 $listLambdaFunctions['Functions']);
        }
        foreach ($allLambdaFunctions as $function) {
            echo "{$function['FunctionName']}\n";
        }

        echo "\n\nAnd don't forget to clean up your data!\n";

        $lambdaService->deleteFunction($functionName);
        echo "Deleted Lambda function.\n";
        $iamService->deleteRole($role['RoleName']);
        echo "Deleted Role.\n";
        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
```

```
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Deleted all objects from the S3 bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);
        echo "Deleted the bucket.\n";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina um manipulador do Lambda que aumente um número.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
 number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the
 function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get("action")
    if action == "increment":
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Defina um segundo manipulador do Lambda que faça operações aritméticas.

```
import logging
import os


logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
 function.
ACTIONS = {
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}
```

```
def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the
 numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the
 function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)

    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {"result": result}
    return response
```

Crie funções que envolvam ações do Lambda.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource
```

```
    @staticmethod
    def create_deployment_package(source_file, destination_file):
        """
        Creates a Lambda deployment package in .zip format in an in-memory
buffer. This
        buffer can be passed directly to Lambda when creating the function.

        :param source_file: The name of the file that contains the Lambda handler
                            function.
        :param destination_file: The name to give the file when it's deployed to
Lambda.
        :return: The deployment package.
        """
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, "w") as zipped:
            zipped.write(source_file, destination_file)
        buffer.seek(0)
        return buffer.read()

    def get_iam_role(self, iam_role_name):
        """
        Get an AWS Identity and Access Management (IAM) role.

        :param iam_role_name: The name of the role to retrieve.
        :return: The IAM role.
        """
        role = None
        try:
            temp_role = self.iam_resource.Role(iam_role_name)
            temp_role.load()
            role = temp_role
            logger.info("Got IAM role %s", role.name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                logger.info("IAM role %s does not exist.", iam_role_name)
            else:
                logger.error(
                    "Couldn't get IAM role %s. Here's why: %s: %s",
                    iam_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
                raise
        return role

    def create_iam_role_for_lambda(self, iam_role_name):
        """
        Creates an IAM role that grants the Lambda function basic permissions. If
 a
        role with the specified name already exists, it is used for the demo.

        :param iam_role_name: The name of the role to create.
        :return: The role and a value that indicates whether the role is newly
 created.
        """
        role = self.get_iam_role(iam_role_name)
        if role is not None:
            return role, False

        lambda_assume_role_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "lambda.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"

        try:
            role = self.iam_resource.create_role(
                RoleName=iam_role_name,
                AssumeRolePolicyDocument=json.dumps(lambda_assume_role_policy),
            )
            logger.info("Created role %s.", role.name)
            role.attach_policy(PolicyArn=policy_arn)
            logger.info("Attached basic execution policy to role %s.", role.name)
        except ClientError as error:
            if error.response["Error"]["Code"] == "EntityAlreadyExists":
                role = self.iam_resource.Role(iam_role_name)
                logger.warning("The role %s already exists. Using it.",
 iam_role_name)
            else:
```

```
                logger.exception(
                    "Couldn't create role %s or attach policy %s.",
                    iam_role_name,
                    policy_arn,
                )
                raise

        return role, True

    def get_function(self, function_name):
        """
        Gets data about a Lambda function.

        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response =
self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response


    def create_function(
        self, function_name, handler_name, iam_role, deployment_package
    ):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function.
This
                             must include the file name and the function name.
```

```
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the
function
                                   code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime="python3.9",
                Role=iam_role.arn,
                Handler=handler_name,
                Code={"ZipFile": deployment_package},
                Publish=True,
            )
            function_arn = response["FunctionArn"]
            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)
            logger.info(
                "Created function '%s' with ARN: '%s'.",
                function_name,
                response["FunctionArn"],
            )
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn


    def delete_function(self, function_name):
        """
        Deletes a Lambda function.

        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise
```

```
    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This
dict
                                is serialized to JSON before it is sent to
Lambda.
        :param get_log: When true, the last 4 KB of the execution log are
included in
                        the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType="Tail" if get_log else "None",
            )
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response


    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
contains
        the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes
in
                                   .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={"Variables": env_vars}
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def list_functions(self):
        """
        Lists the Lambda functions for the current account.
        """
        try:
            func_paginator = self.lambda_client.get_paginator("list_functions")
            for func_page in func_paginator.paginate():
                for func in func_page["Functions"]:
                    print(func["FunctionName"])
```

```
                    desc = func.get("Description")
                    if desc:
                        print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
        except ClientError as err:
            logger.error(
                "Couldn't list functions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Crie uma função que execute o cenário.

```
class UpdateFunctionWaiter(CustomWaiter):
    """A custom waiter that waits until a function is successfully updated."""

    def __init__(self, client):
        super().__init__(
            "UpdateSuccess",
            "GetFunction",
            "Configuration.LastUpdateStatus",
            {"Successful": WaitState.SUCCESS, "Failed": WaitState.FAILURE},
            client,
        )

    def wait(self, function_name):
        self._wait(FunctionName=function_name)


def run_scenario(lambda_client, iam_resource, basic_file, calculator_file,
 lambda_name):
    """
    Runs the scenario.

    :param lambda_client: A Boto3 Lambda client.
    :param iam_resource: A Boto3 IAM resource.
    :param basic_file: The name of the file that contains the basic Lambda
 handler.
```

```
    :param calculator_file: The name of the file that contains the calculator
Lambda handler.
    :param lambda_name: The name to give resources created for the scenario, such
as the
                        IAM role and the Lambda function.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the AWS Lambda getting started with functions demo.")
    print("-" * 88)

    wrapper = LambdaWrapper(lambda_client, iam_resource)

    print("Checking for IAM role for Lambda...")
    iam_role, should_wait = wrapper.create_iam_role_for_lambda(lambda_name)
    if should_wait:
        logger.info("Giving AWS time to create resources...")
        wait(10)

    print(f"Looking for function {lambda_name}...")
    function = wrapper.get_function(lambda_name)
    if function is None:
        print("Zipping the Python script into a deployment package...")
        deployment_package = wrapper.create_deployment_package(
            basic_file, f"{lambda_name}.py"
        )
        print(f"...and creating the {lambda_name} Lambda function.")
        wrapper.create_function(
            lambda_name, f"{lambda_name}.lambda_handler", iam_role,
deployment_package
        )
    else:
        print(f"Function {lambda_name} already exists.")
    print("-" * 88)

    print(f"Let's invoke {lambda_name}. This function increments a number.")
    action_params = {
        "action": "increment",
        "number": q.ask("Give me a number to increment: ", q.is_int),
    }
    print(f"Invoking {lambda_name}...")
    response = wrapper.invoke_function(lambda_name, action_params)
    print(
```

```
        f"Incrementing {action_params['number']} resulted in "
        f"{json.load(response['Payload'])}"
    )
    print("-" * 88)

    print(f"Let's update the function to an arithmetic calculator.")
    q.ask("Press Enter when you're ready.")
    print("Creating a new deployment package...")
    deployment_package = wrapper.create_deployment_package(
        calculator_file, f"{lambda_name}.py"
    )
    print(f"...and updating the {lambda_name} Lambda function.")
    update_waiter = UpdateFunctionWaiter(lambda_client)
    wrapper.update_function_code(lambda_name, deployment_package)
    update_waiter.wait(lambda_name)
    print(f"This function uses an environment variable to control logging
level.")
    print(f"Let's set it to DEBUG to get the most logging.")
    wrapper.update_function_configuration(
        lambda_name, {"LOG_LEVEL": logging.getLevelName(logging.DEBUG)}
    )

    actions = ["plus", "minus", "times", "divided-by"]
    want_invoke = True
    while want_invoke:
        print(f"Let's invoke {lambda_name}. You can invoke these actions:")
        for index, action in enumerate(actions):
            print(f"{index + 1}: {action}")
        action_params = {}
        action_index = q.ask(
            "Enter the number of the action you want to take: ",
            q.is_int,
            q.in_range(1, len(actions)),
        )
        action_params["action"] = actions[action_index - 1]
        print(f"You've chosen to invoke 'x {action_params['action']} y'.")
        action_params["x"] = q.ask("Enter a value for x: ", q.is_int)
        action_params["y"] = q.ask("Enter a value for y: ", q.is_int)
        print(f"Invoking {lambda_name}...")
        response = wrapper.invoke_function(lambda_name, action_params, True)
        print(
            f"Calculating {action_params['x']} {action_params['action']}
{action_params['y']} "
            f"resulted in {json.load(response['Payload'])}"
```

```
        )
        q.ask("Press Enter to see the logs from the call.")
        print(base64.b64decode(response["LogResult"]).decode())
        want_invoke = q.ask("That was fun. Shall we do it again? (y/n) ",
 q.is_yesno)
    print("-" * 88)

    if q.ask(
        "Do you want to list all of the functions in your account? (y/n) ",
 q.is_yesno
    ):
        wrapper.list_functions()
    print("-" * 88)

    if q.ask("Ready to delete the function and role? (y/n) ", q.is_yesno):
        for policy in iam_role.attached_policies.all():
            policy.detach_role(RoleName=iam_role.name)
        iam_role.delete()
        print(f"Deleted role {lambda_name}.")
        wrapper.delete_function(lambda_name)
        print(f"Deleted function {lambda_name}.")

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        run_scenario(
            boto3.client("lambda"),
            boto3.resource("iam"),
            "lambda_handler_basic.py",
            "lambda_handler_calculator.py",
            "doc_example_lambda_calculator",
        )
    except Exception:
        logging.exception("Something went wrong with the demo!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateFunction

- DeleteFunction
- GetFunction
- Invoke
- ListFunctions
- UpdateFunctionCode
- UpdateFunctionConfiguration

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure as permissões de pré-requisitos do IAM para uma função do Lambda capaz de gravar logs.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  case action
  when 'create'
    create_iam_role(iam_role_name)
  when 'destroy'
    destroy_iam_role(iam_role_name)
  else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
  end
end

private

def create_iam_role(iam_role_name)
```

```
    role_policy = {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Effect': 'Allow',
          'Principal': { 'Service': 'lambda.amazonaws.com' },
          'Action': 'sts:AssumeRole'
        }
      ]
    }
    role = @iam_client.create_role(
      role_name: iam_role_name,
      assume_role_policy_document: role_policy.to_json
    )
    @iam_client.attach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    wait_for_role_to_exist(iam_role_name)
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    sleep(10)
    [role, role_policy.to_json]
  end

  def destroy_iam_role(iam_role_name)
    @iam_client.detach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    @iam_client.delete_role(role_name: iam_role_name)
    @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
  end

  def wait_for_role_to_exist(iam_role_name)
    @iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
```

```
  end
```

Defina um manipulador do Lambda que incremente um número fornecido como um parâmetro de invocação.

```
require 'logger'

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
# @param event [Hash] Parameters sent when the function is invoked
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
  logger = Logger.new($stdout)
  log_level = ENV['LOG_LEVEL']
  logger.level = case log_level
                 when 'debug'
                   Logger::DEBUG
                 when 'info'
                   Logger::INFO
                 else
                   Logger::ERROR
                 end
  logger.debug('This is a debug log message.')
  logger.info('This is an info log message. Code executed successfully!')
  number = event['number'].to_i
  incremented_number = number + 1
  logger.info("You provided #{number.round} and it was incremented to
 #{incremented_number.round}")
  incremented_number.round.to_s
end
```

Compacte a função do Lambda em um pacote de implantação.

```
  # Creates a Lambda deployment package in .zip format.
  #
  # @param source_file: The name of the object, without suffix, for the Lambda
 file and zip.
  # @return: The deployment package.
```

```
  def create_deployment_package(source_file)
    Dir.chdir(File.dirname(__FILE__))
    if File.exist?('lambda_function.zip')
      File.delete('lambda_function.zip')
      @logger.debug('Deleting old zip: lambda_function.zip')
    end
    Zip::File.open('lambda_function.zip', create: true) do |zipfile|
      zipfile.add('lambda_function.rb', "#{source_file}.rb")
    end
    @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
    File.read('lambda_function.zip').to_s
  rescue StandardError => e
    @logger.error("There was an error creating deployment package:\n
#{e.message}")
  end
```

Crie uma nova função do Lambda.

```
  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
```

```
    @lambda_client.wait_until(:function_active_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
  end
```

Invoque a função do Lambda com parâmetros de runtime opcionais.

```
  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n
#{e.message}")
  end
```

Atualize a configuração da função do Lambda para injetar uma nova variável de ambiente.

```
  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                    function_name: function_name,
                                                    environment: {
                                                      variables: {
                                                        'LOG_LEVEL' => log_level
                                                      }
```

```
                                                    }
                                                  })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for
#{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
  end
```

Atualize o código da função do Lambda com um pacote de implantação diferente que contenha um código diferente.

```
  # Updates the code for a Lambda function by submitting a .zip archive that
contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for:
#{function_name}:\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
```

```
    @logger.error("Failed waiting for #{function_name} to update:\n
#{e.message}")
  end
```

Liste todas as funções do Lambda existentes usando o paginador integrado.

```
# Lists the Lambda functions for the current account.
def list_functions
  functions = []
  @lambda_client.list_functions.each do |response|
    response['functions'].each do |function|
      functions.append(function['function_name'])
    end
  end
  functions
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error listing functions:\n #{e.message}")
end
```

Exclua uma função do Lambda específica.

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name: function_name
  )
  print 'Done!'.green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - CreateFunction
  - DeleteFunction
  - GetFunction

- Invoke
- ListFunctions
- UpdateFunctionCode
- UpdateFunctionConfiguration

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O Cargo.toml com dependências usadas nesse cenário.

```
[package]
name = "lambda-code-examples"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-ec2 = { version = "1.3.0" }
aws-sdk-iam = { version = "1.3.0" }
aws-sdk-lambda = { version = "1.3.0" }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
tracing = "0.1.37"
serde_json = "1.0.94"
anyhow = "1.0.71"
uuid = { version = "1.3.3", features = ["v4"] }
```

```
lambda_runtime = "0.8.0"
serde = "1.0.164"
```

Uma coleção de utilitários que simplificam chamar o Lambda para este cenário. Esse arquivo é src/ations .rs na caixa.

```
use anyhow::anyhow;
use aws_sdk_iam::operation::{create_role::CreateRoleError,
 delete_role::DeleteRoleOutput};
use aws_sdk_lambda::{
    operation::{
        delete_function::DeleteFunctionOutput, get_function::GetFunctionOutput,
        invoke::InvokeOutput, list_functions::ListFunctionsOutput,
        update_function_code::UpdateFunctionCodeOutput,
        update_function_configuration::UpdateFunctionConfigurationOutput,
    },
    primitives::ByteStream,
    types::{Environment, FunctionCode, LastUpdateStatus, State},
};
use aws_sdk_s3::{
    error::ErrorMetadata,
    operation::{delete_bucket::DeleteBucketOutput,
 delete_object::DeleteObjectOutput},
    types::CreateBucketConfiguration,
};
use aws_smithy_types::Blob;
use serde::{ser::SerializeMap, Serialize};
use std::{fmt::Display, path::PathBuf, str::FromStr, time::Duration};
use tracing::{debug, info, warn};

/* Operation describes  */
#[derive(Clone, Copy, Debug, Serialize)]
pub enum Operation {
    #[serde(rename = "plus")]
    Plus,
    #[serde(rename = "minus")]
    Minus,
    #[serde(rename = "times")]
    Times,
    #[serde(rename = "divided-by")]
    DividedBy,
```

```
}

impl FromStr for Operation {
    type Err = anyhow::Error;

    fn from_str(s: &str) -> Result<Self, Self::Err> {
        match s {
            "plus" => Ok(Operation::Plus),
            "minus" => Ok(Operation::Minus),
            "times" => Ok(Operation::Times),
            "divided-by" => Ok(Operation::DividedBy),
            _ => Err(anyhow!("Unknown operation {s}")),
        }
    }
}

impl Display for Operation {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match self {
            Operation::Plus => write!(f, "plus"),
            Operation::Minus => write!(f, "minus"),
            Operation::Times => write!(f, "times"),
            Operation::DividedBy => write!(f, "divided-by"),
        }
    }
}

/**
 * InvokeArgs will be serialized as JSON and sent to the AWS Lambda handler.
 */
#[derive(Debug)]
pub enum InvokeArgs {
    Increment(i32),
    Arithmetic(Operation, i32, i32),
}

impl Serialize for InvokeArgs {
    fn serialize<S>(&self, serializer: S) -> Result<S::Ok, S::Error>
    where
        S: serde::Serializer,
    {
        match self {
            InvokeArgs::Increment(i) => serializer.serialize_i32(*i),
            InvokeArgs::Arithmetic(o, i, j) => {
```

```
                let mut map: S::SerializeMap =
 serializer.serialize_map(Some(3))?;
                map.serialize_key(&"op".to_string())?;
                map.serialize_value(&o.to_string())?;
                map.serialize_key(&"i".to_string())?;
                map.serialize_value(&i)?;
                map.serialize_key(&"j".to_string())?;
                map.serialize_value(&j)?;
                map.end()
            }
        }
    }
}

/** A policy document allowing Lambda to execute this function on the account's
 behalf. */
const ROLE_POLICY_DOCUMENT: &str = r#"{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "Service": "lambda.amazonaws.com" },
            "Action": "sts:AssumeRole"
        }
    ]
}"#;

/**
 * A LambdaManager gathers all the resources necessary to run the Lambda example
 scenario.
 * This includes instantiated aws_sdk clients and details of resource names.
 */
pub struct LambdaManager {
    iam_client: aws_sdk_iam::Client,
    lambda_client: aws_sdk_lambda::Client,
    s3_client: aws_sdk_s3::Client,
    lambda_name: String,
    role_name: String,
    bucket: String,
    own_bucket: bool,
}

// These unit type structs provide nominal typing on top of String parameters for
 LambdaManager::new
```

```
pub struct LambdaName(pub String);
pub struct RoleName(pub String);
pub struct Bucket(pub String);
pub struct OwnBucket(pub bool);

impl LambdaManager {
    pub fn new(
        iam_client: aws_sdk_iam::Client,
        lambda_client: aws_sdk_lambda::Client,
        s3_client: aws_sdk_s3::Client,
        lambda_name: LambdaName,
        role_name: RoleName,
        bucket: Bucket,
        own_bucket: OwnBucket,
    ) -> Self {
        Self {
            iam_client,
            lambda_client,
            s3_client,
            lambda_name: lambda_name.0,
            role_name: role_name.0,
            bucket: bucket.0,
            own_bucket: own_bucket.0,
        }
    }

    /**
     * Load the AWS configuration from the environment.
     * Look up lambda_name and bucket if none are given, or generate a random
 name if not present in the environment.
     * If the bucket name is provided, the caller needs to have created the
 bucket.
     * If the bucket name is generated, it will be created.
     */
    pub async fn load_from_env(lambda_name: Option<String>, bucket:
 Option<String>) -> Self {
        let sdk_config = aws_config::load_from_env().await;
        let lambda_name = LambdaName(lambda_name.unwrap_or_else(|| {
            std::env::var("LAMBDA_NAME").unwrap_or_else(|_|
 "rust_lambda_example".to_string())
        }));
        let role_name = RoleName(format!("{}_role", lambda_name.0));
        let (bucket, own_bucket) =
            match bucket {
```

```
                Some(bucket) => (Bucket(bucket), false),
                None => (
                    Bucket(std::env::var("LAMBDA_BUCKET").unwrap_or_else(|_| {
                        format!("rust-lambda-example-{}", uuid::Uuid::new_v4())
                    })),
                    true,
                ),
            };

        let s3_client = aws_sdk_s3::Client::new(&sdk_config);

        if own_bucket {
            info!("Creating bucket for demo: {}", bucket.0);
            s3_client
                .create_bucket()
                .bucket(bucket.0.clone())
                .create_bucket_configuration(
                    CreateBucketConfiguration::builder()

.location_constraint(aws_sdk_s3::types::BucketLocationConstraint::from(
                            sdk_config.region().unwrap().as_ref(),
                        ))
                        .build(),
                )
                .send()
                .await
                .unwrap();
        }

        Self::new(
            aws_sdk_iam::Client::new(&sdk_config),
            aws_sdk_lambda::Client::new(&sdk_config),
            s3_client,
            lambda_name,
            role_name,
            bucket,
            OwnBucket(own_bucket),
        )
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
```

```
     * The easiest way to create such a zip is to use `cargo lambda build --
output-format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }

    /**
     * Create a function, uploading from a zip file.
     */
    pub async fn create_function(&self, zip_file: PathBuf) -> Result<String,
 anyhow::Error> {
        let code = self.prepare_function(zip_file, None).await?;

        let key = code.s3_key().unwrap().to_string();

        let role = self.create_role().await.map_err(|e| anyhow!(e))?;

        info!("Created iam role, waiting 15s for it to become active");
        tokio::time::sleep(Duration::from_secs(15)).await;

        info!("Creating lambda function {}", self.lambda_name);
        let _ = self
```

```
            .lambda_client
            .create_function()
            .function_name(self.lambda_name.clone())
            .code(code)
            .role(role.arn())
            .runtime(aws_sdk_lambda::types::Runtime::Providedal2)
            .handler("_unused")
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        self.lambda_client
            .publish_version()
            .function_name(self.lambda_name.clone())
            .send()
            .await?;

        Ok(key)
    }

    /**
     * Create an IAM execution role for the managed Lambda function.
     * If the role already exists, use that instead.
     */
    async fn create_role(&self) -> Result<aws_sdk_iam::types::Role,
CreateRoleError> {
        info!("Creating execution role for function");
        let get_role = self
            .iam_client
            .get_role()
            .role_name(self.role_name.clone())
            .send()
            .await;
        if let Ok(get_role) = get_role {
            if let Some(role) = get_role.role {
                return Ok(role);
            }
        }

        let create_role = self
            .iam_client
            .create_role()
```

```
            .role_name(self.role_name.clone())
            .assume_role_policy_document(ROLE_POLICY_DOCUMENT)
            .send()
            .await;

        match create_role {
            Ok(create_role) => match create_role.role {
                Some(role) => Ok(role),
                None => Err(CreateRoleError::generic(
                    ErrorMetadata::builder()
                        .message("CreateRole returned empty success")
                        .build(),
                )),
            },
            Err(err) => Err(err.into_service_error()),
        }
    }

    /**
     * Poll `is_function_ready` with a 1-second delay. It returns when the
function is ready or when there's an error checking the function's state.
     */
    pub async fn wait_for_function_ready(&self) -> Result<(), anyhow::Error> {
        info!("Waiting for function");
        while !self.is_function_ready(None).await? {
            info!("Function is not ready, sleeping 1s");
            tokio::time::sleep(Duration::from_secs(1)).await;
        }
        Ok(())
    }

    /**
     * Check if a Lambda function is ready to be invoked.
     * A Lambda function is ready for this scenario when its state is active and
its LastUpdateStatus is Successful.
     * Additionally, if a sha256 is provided, the function must have that as its
current code hash.
     * Any missing properties or failed requests will be reported as an Err.
     */
    async fn is_function_ready(
        &self,
        expected_code_sha256: Option<&str>,
    ) -> Result<bool, anyhow::Error> {
        match self.get_function().await {
```

```
            Ok(func) => {
                if let Some(config) = func.configuration() {
                    if let Some(state) = config.state() {
                        info!(?state, "Checking if function is active");
                        if !matches!(state, State::Active) {
                            return Ok(false);
                        }
                    }
                    match config.last_update_status() {
                        Some(last_update_status) => {
                            info!(?last_update_status, "Checking if function is
ready");
                            match last_update_status {
                                LastUpdateStatus::Successful => {
                                    // continue
                                }
                                LastUpdateStatus::Failed |
LastUpdateStatus::InProgress => {
                                    return Ok(false);
                                }
                                unknown => {
                                    warn!(
                                        status_variant = unknown.as_str(),
                                        "LastUpdateStatus unknown"
                                    );
                                    return Err(anyhow!(
                                        "Unknown LastUpdateStatus, fn config is
{config:?}"
                                    ));
                                }
                            }
                        }
                        None => {
                            warn!("Missing last update status");
                            return Ok(false);
                        }
                    };
                    if expected_code_sha256.is_none() {
                        return Ok(true);
                    }
                    if let Some(code_sha256) = config.code_sha256() {
                        return Ok(code_sha256 ==
expected_code_sha256.unwrap_or_default());
                    }
```

```
                }
            }
            Err(e) => {
                warn!(?e, "Could not get function while waiting");
            }
        }
        Ok(false)
    }

    /** Get the Lambda function with this Manager's name. */
    pub async fn get_function(&self) -> Result<GetFunctionOutput, anyhow::Error>
{
        info!("Getting lambda function");
        self.lambda_client
            .get_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

    /** List all Lambda functions in the current Region. */
    pub async fn list_functions(&self) -> Result<ListFunctionsOutput,
anyhow::Error> {
        info!("Listing lambda functions");
        self.lambda_client
            .list_functions()
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

    /** Invoke the lambda function using calculator InvokeArgs. */
    pub async fn invoke(&self, args: InvokeArgs) -> Result<InvokeOutput,
anyhow::Error> {
        info!(?args, "Invoking {}", self.lambda_name);
        let payload = serde_json::to_string(&args)?;
        debug!(?payload, "Sending payload");
        self.lambda_client
            .invoke()
            .function_name(self.lambda_name.clone())
            .payload(Blob::new(payload))
            .send()
            .await
```

```
            .map_err(anyhow::Error::from)
    }

    /** Given a Path to a zip file, update the function's code and wait for the
update to finish. */
    pub async fn update_function_code(
        &self,
        zip_file: PathBuf,
        key: String,
    ) -> Result<UpdateFunctionCodeOutput, anyhow::Error> {
        let function_code = self.prepare_function(zip_file, Some(key)).await?;

        info!("Updating code for {}", self.lambda_name);
        let update = self
            .lambda_client
            .update_function_code()
            .function_name(self.lambda_name.clone())
            .s3_bucket(self.bucket.clone())
            .s3_key(function_code.s3_key().unwrap().to_string())
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(update)
    }

    /** Update the environment for a function. */
    pub async fn update_function_configuration(
        &self,
        environment: Environment,
    ) -> Result<UpdateFunctionConfigurationOutput, anyhow::Error> {
        info!(
            ?environment,
            "Updating environment for {}", self.lambda_name
        );
        let updated = self
            .lambda_client
            .update_function_configuration()
            .function_name(self.lambda_name.clone())
            .environment(environment)
            .send()
            .await
```

```
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(updated)
    }

    /** Delete a function and its role, and if possible or necessary, its
associated code object and bucket. */
    pub async fn delete_function(
        &self,
        location: Option<String>,
    ) -> (
        Result<DeleteFunctionOutput, anyhow::Error>,
        Result<DeleteRoleOutput, anyhow::Error>,
        Option<Result<DeleteObjectOutput, anyhow::Error>>,
    ) {
        info!("Deleting lambda function {}", self.lambda_name);
        let delete_function = self
            .lambda_client
            .delete_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        info!("Deleting iam role {}", self.role_name);
        let delete_role = self
            .iam_client
            .delete_role()
            .role_name(self.role_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        let delete_object: Option<Result<DeleteObjectOutput, anyhow::Error>> =
            if let Some(location) = location {
                info!("Deleting object {location}");
                Some(
                    self.s3_client
                        .delete_object()
                        .bucket(self.bucket.clone())
                        .key(location)
                        .send()
```

```
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                info!(?location, "Skipping delete object");
                None
            };

        (delete_function, delete_role, delete_object)
    }

    pub async fn cleanup(
        &self,
        location: Option<String>,
    ) -> (
        (
            Result<DeleteFunctionOutput, anyhow::Error>,
            Result<DeleteRoleOutput, anyhow::Error>,
            Option<Result<DeleteObjectOutput, anyhow::Error>>,
        ),
        Option<Result<DeleteBucketOutput, anyhow::Error>>,
    ) {
        let delete_function = self.delete_function(location).await;

        let delete_bucket = if self.own_bucket {
            info!("Deleting bucket {}", self.bucket);
            if delete_function.2.is_none() ||
delete_function.2.as_ref().unwrap().is_ok() {
                Some(
                    self.s3_client
                        .delete_bucket()
                        .bucket(self.bucket.clone())
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                None
            }
        } else {
            info!("No bucket to clean up");
            None
        };
```

```
        (delete_function, delete_bucket)
    }
}

/**
 * Testing occurs primarily as an integration test running the `scenario` bin
 successfully.
 * Each action relies deeply on the internal workings and state of Amazon Simple
 Storage Service (Amazon S3), Lambda, and IAM working together.
 * It is therefore infeasible to mock the clients to test the individual actions.
 */
#[cfg(test)]
mod test {
    use super::{InvokeArgs, Operation};
    use serde_json::json;

    /** Make sure that the JSON output of serializing InvokeArgs is what's
 expected by the calculator. */
    #[test]
    fn test_serialize() {
        assert_eq!(json!(InvokeArgs::Increment(5)), 5);
        assert_eq!(
            json!(InvokeArgs::Arithmetic(Operation::Plus, 5, 7)).to_string(),
            r#"{"op":"plus","i":5,"j":7}"#.to_string(),
        );
    }
}
```

Um binário para executar o cenário do início ao fim usando sinalizadores de linha de comando para controlar algum comportamento. Esse arquivo está em src/bin /scenario.rs na caixa.

```
/*
## Service actions

Service actions wrap the SDK call, taking a client and any specific parameters
 necessary for the call.

* CreateFunction
* GetFunction
* ListFunctions
* Invoke
```

```
* UpdateFunctionCode
* UpdateFunctionConfiguration
* DeleteFunction

## Scenario
A scenario runs at a command prompt and prints output to the user on the result
 of each service action. A scenario can run in one of two ways: straight through,
 printing out progress as it goes, or as an interactive question/answer script.

## Getting started with functions

Use an SDK to manage AWS Lambda functions: create a function, invoke it, update
 its code, invoke it again, view its output and logs, and delete it.

This scenario uses two Lambda handlers:
_Note: Handlers don't use AWS SDK API calls._

The increment handler is straightforward:

1. It accepts a number, increments it, and returns the new value.
2. It performs simple logging of the result.

The arithmetic handler is more complex:
1. It accepts a set of actions ['plus', 'minus', 'times', 'divided-by'] and two
 numbers, and returns the result of the calculation.
2. It uses an environment variable to control log level (such as DEBUG, INFO,
 WARNING, ERROR).
It logs a few things at different levels, such as:
    * DEBUG: Full event data.
    * INFO: The calculation result.
    * WARN~ING~: When a divide by zero error occurs.
    * This will be the typical `RUST_LOG` variable.


The steps of the scenario are:

1. Create an AWS Identity and Access Management (IAM) role that meets the
 following requirements:
    * Has an assume_role policy that grants 'lambda.amazonaws.com' the
 'sts:AssumeRole' action.
    * Attaches the 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' managed role.
    * _You must wait for ~10 seconds after the role is created before you can use
 it!_
```

```
2. Create a function (CreateFunction) for the increment handler by packaging it
 as a zip and doing one of the following:
    * Adding it with CreateFunction Code.ZipFile.
    * --or--
    * Uploading it to Amazon Simple Storage Service (Amazon S3) and adding it
 with CreateFunction Code.S3Bucket/S3Key.
    * _Note: Zipping the file does not have to be done in code._
    * If you have a waiter, use it to wait until the function is active.
 Otherwise, call GetFunction until State is Active.
3. Invoke the function with a number and print the result.
4. Update the function (UpdateFunctionCode) to the arithmetic handler by
 packaging it as a zip and doing one of the following:
    * Adding it with UpdateFunctionCode ZipFile.
    * --or--
    * Uploading it to Amazon S3 and adding it with UpdateFunctionCode S3Bucket/
S3Key.
5. Call GetFunction until Configuration.LastUpdateStatus is 'Successful' (or
 'Failed').
6. Update the environment variable by calling UpdateFunctionConfiguration and
 pass it a log level, such as:
    * Environment={'Variables': {'RUST_LOG': 'TRACE'}}
7. Invoke the function with an action from the list and a couple of values.
 Include LogType='Tail' to get logs in the result. Print the result of the
 calculation and the log.
8. [Optional] Invoke the function to provoke a divide-by-zero error and show the
 log result.
9. List all functions for the account, using pagination (ListFunctions).
10. Delete the function (DeleteFunction).
11. Delete the role.

Each step should use the function created in Service Actions to abstract calling
 the SDK.
 */

use aws_sdk_lambda::{operation::invoke::InvokeOutput, types::Environment};
use clap::Parser;
use std::{collections::HashMap, path::PathBuf};
use tracing::{debug, info, warn};
use tracing_subscriber::EnvFilter;

use lambda_code_examples::actions::{
    InvokeArgs::{Arithmetic, Increment},
    LambdaManager, Operation,
};
```

```
#[derive(Debug, Parser)]
pub struct Opt {
    /// The AWS Region.
    #[structopt(short, long)]
    pub region: Option<String>,

    // The bucket to use for the FunctionCode.
    #[structopt(short, long)]
    pub bucket: Option<String>,

    // The name of the Lambda function.
    #[structopt(short, long)]
    pub lambda_name: Option<String>,

    // The number to increment.
    #[structopt(short, long, default_value = "12")]
    pub inc: i32,

    // The left operand.
    #[structopt(long, default_value = "19")]
    pub num_a: i32,

    // The right operand.
    #[structopt(long, default_value = "23")]
    pub num_b: i32,

    // The arithmetic operation.
    #[structopt(short, long, default_value = "plus")]
    pub operation: Operation,

    #[structopt(long)]
    pub cleanup: Option<bool>,

    #[structopt(long)]
    pub no_cleanup: Option<bool>,
}

fn code_path(lambda: &str) -> PathBuf {
    PathBuf::from(format!("../target/lambda/{lambda}/bootstrap.zip"))
}

fn log_invoke_output(invoke: &InvokeOutput, message: &str) {
    if let Some(payload) = invoke.payload().cloned() {
```

```
        let payload = String::from_utf8(payload.into_inner());
        info!(?payload, message);
    } else {
        info!("Could not extract payload")
    }
    if let Some(logs) = invoke.log_result() {
        debug!(?logs, "Invoked function logs")
    } else {
        debug!("Invoked function had no logs")
    }
}

async fn main_block(
    opt: &Opt,
    manager: &LambdaManager,
    code_location: String,
) -> Result<(), anyhow::Error> {
    let invoke = manager.invoke(Increment(opt.inc)).await?;
    log_invoke_output(&invoke, "Invoked function configured as increment");

    let update_code = manager
        .update_function_code(code_path("arithmetic"), code_location.clone())
        .await?;

    let code_sha256 = update_code.code_sha256().unwrap_or("Unknown SHA");
    info!(?code_sha256, "Updated function code with arithmetic.zip");

    let arithmetic_args = Arithmetic(opt.operation, opt.num_a, opt.num_b);
    let invoke = manager.invoke(arithmetic_args).await?;
    log_invoke_output(&invoke, "Invoked function configured as arithmetic");

    let update = manager
        .update_function_configuration(
            Environment::builder()
                .set_variables(Some(HashMap::from([(
                    "RUST_LOG".to_string(),
                    "trace".to_string(),
                )])))
                .build(),
        )
        .await?;
    let updated_environment = update.environment();
    info!(?updated_environment, "Updated function configuration");
```

```
    let invoke = manager
        .invoke(Arithmetic(opt.operation, opt.num_a, opt.num_b))
        .await?;
    log_invoke_output(
        &invoke,
        "Invoked function configured as arithmetic with increased logging",
    );

    let invoke = manager
        .invoke(Arithmetic(Operation::DividedBy, opt.num_a, 0))
        .await?;
    log_invoke_output(
        &invoke,
        "Invoked function configured as arithmetic with divide by zero",
    );

    Ok::<(), anyhow::Error>(())
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt()
        .without_time()
        .with_file(true)
        .with_line_number(true)
        .with_env_filter(EnvFilter::from_default_env())
        .init();

    let opt = Opt::parse();
    let manager = LambdaManager::load_from_env(opt.lambda_name.clone(),
 opt.bucket.clone()).await;

    let key = match manager.create_function(code_path("increment")).await {
        Ok(init) => {
            info!(?init, "Created function, initially with increment.zip");
            let run_block = main_block(&opt, &manager, init.clone()).await;
            info!(?run_block, "Finished running example, cleaning up");
            Some(init)
        }
        Err(err) => {
            warn!(?err, "Error happened when initializing function");
            None
        }
    };
```

```
    if Some(false) == opt.cleanup || Some(true) == opt.no_cleanup {
        info!("Skipping cleanup")
    } else {
        let delete = manager.cleanup(key).await;
        info!(?delete, "Deleted function & cleaned up resources");
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        "Create an AWS Identity and Access Management (IAM) role that grants AWS
 Lambda permission to write to logs."
        DATA(lv_policy_document) = `{` &&
            `"Version":"2012-10-17",       ` &&
                  `"Statement": [` &&
                    `{` &&
```

```
                      `"Effect": "Allow",` &&
                      `"Action": [` &&
                        `"sts:AssumeRole"` &&
                      `],` &&
                      `"Principal": {` &&
                        `"Service": [` &&
                          `"lambda.amazonaws.com"` &&
                        `]` &&
                      `}` &&
                    `}` &&
                  `]` &&
                `}`.
        TRY.
            DATA(lo_create_role_output) = lo_iam->createrole(
                    iv_rolename = iv_role_name
                    iv_assumerolepolicydocument = lv_policy_document
                    iv_description = 'Grant lambda permission to write to
 logs' ).
            DATA(lv_role_arn) = lo_create_role_output->get_role( )->get_arn( ).
            MESSAGE 'IAM role created.' TYPE 'I'.
            WAIT UP TO 10 SECONDS.           " Make sure that the IAM role is
 ready for use. "
          CATCH /aws1/cx_iamentityalrdyexex.
            DATA(lo_role) = lo_iam->getrole( iv_rolename = iv_role_name ).
            lv_role_arn = lo_role->get_role( )->get_arn( ).
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iammalformedplydocex.
            MESSAGE 'Policy document in the request is malformed.' TYPE 'E'.
        ENDTRY.

        TRY.
            lo_iam->attachrolepolicy(
                iv_rolename  = iv_role_name
                iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' ).
            MESSAGE 'Attached policy to the IAM role.' TYPE 'I'.
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'E'.
          CATCH /aws1/cx_iamplynotattachableex.
            MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
```

```
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.

        " Create a Lambda function and upload handler code. "
        " Lambda function performs 'increment' action on a number. "
        TRY.
            lo_lmd->createfunction(
                 iv_functionname = iv_function_name
                 iv_runtime = `python3.9`
                 iv_role = lv_role_arn
                 iv_handler = iv_handler
                 io_code = io_initial_zip_file
                 iv_description = 'AWS Lambda code example' ).
            MESSAGE 'Lambda function created.' TYPE 'I'.
          CATCH /aws1/cx_lmdcodestorageexcdex.
            MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        " Verify the function is in Active state "
        WHILE lo_lmd->getfunction( iv_functionname = iv_function_name )-
>get_configuration( )->ask_state( ) <> 'Active'.
          IF sy-index = 10.
            EXIT.               " Maximum 10 seconds. "
          ENDIF.
          WAIT UP TO 1 SECONDS.
        ENDWHILE.

        "Invoke the function with a single parameter and get results."
        TRY.
            DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
              `{`  &&
                `"action": "increment",`  &&
                `"number": 10` &&
              `}` ).
            DATA(lo_initial_invoke_output) = lo_lmd->invoke(
                       iv_functionname = iv_function_name
                       iv_payload = lv_json ).
```

```
            ov_initial_invoke_payload = lo_initial_invoke_output->get_payload( ).
           " ov_initial_invoke_payload is returned for testing purposes. "
            DATA(lo_writer_json) = cl_sxml_string_writer=>create( type =
 if_sxml=>co_xt_json ).
            CALL TRANSFORMATION id SOURCE XML ov_initial_invoke_payload RESULT
 XML lo_writer_json.
            DATA(lv_result) = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
            MESSAGE 'Lambda function invoked.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
          CATCH /aws1/cx_lmdunsuppedmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
        ENDTRY.

        " Update the function code and configure its Lambda environment with an
 environment variable. "
        " Lambda function is updated to perform 'decrement' action also. "
        TRY.
            lo_lmd->updatefunctioncode(
                  iv_functionname = iv_function_name
                  iv_zipfile = io_updated_zip_file ).
            WAIT UP TO 10 SECONDS.            " Make sure that the update is
 completed. "
            MESSAGE 'Lambda function code updated.' TYPE 'I'.
          CATCH /aws1/cx_lmdcodestorageexcdex.
            MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        TRY.
            DATA lt_variables TYPE /aws1/
cl_lmdenvironmentvaria00=>tt_environmentvariables.
            DATA ls_variable LIKE LINE OF lt_variables.
            ls_variable-key = 'LOG_LEVEL'.
```

```
            ls_variable-value = NEW /aws1/cl_lmdenvironmentvaria00( iv_value =
 'info' ).
            INSERT ls_variable INTO TABLE lt_variables.

            lo_lmd->updatefunctionconfiguration(
                  iv_functionname = iv_function_name
                  io_environment = NEW /aws1/cl_lmdenvironment( it_variables =
 lt_variables ) ).
            WAIT UP TO 10 SECONDS.            " Make sure that the update is
 completed. "
            MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourceconflictex.
            MESSAGE 'Resource already exists or another operation is in
 progress.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        "Invoke the function with new parameters and get results. Display the
 execution log that's returned from the invocation."
        TRY.
            lv_json = /aws1/cl_rt_util=>string_to_xstring(
              `{`  &&
                `"action": "decrement",`  &&
                `"number": 10` &&
              `}` ).
            DATA(lo_updated_invoke_output) = lo_lmd->invoke(
                       iv_functionname = iv_function_name
                       iv_payload = lv_json ).
            ov_updated_invoke_payload = lo_updated_invoke_output->get_payload( ).
           " ov_updated_invoke_payload is returned for testing purposes. "
            lo_writer_json = cl_sxml_string_writer=>create( type =
 if_sxml=>co_xt_json ).
            CALL TRANSFORMATION id SOURCE XML ov_updated_invoke_payload RESULT
 XML lo_writer_json.
            lv_result = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
            MESSAGE 'Lambda function invoked.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
```

```
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
          CATCH /aws1/cx_lmdunsuppedmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
        ENDTRY.

        " List the functions for your account. "
        TRY.
            DATA(lo_list_output) = lo_lmd->listfunctions( ).
            DATA(lt_functions) = lo_list_output->get_functions( ).
            MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
        ENDTRY.

        " Delete the Lambda function. "
        TRY.
            lo_lmd->deletefunction( iv_functionname = iv_function_name ).
            MESSAGE 'Lambda function deleted.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'W'.
        ENDTRY.

        " Detach role policy. "
        TRY.
            lo_iam->detachrolepolicy(
                iv_rolename  = iv_role_name
                iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' ).
            MESSAGE 'Detached policy from the IAM role.' TYPE 'I'.
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'W'.
          CATCH /aws1/cx_iamplynotattachableex.
            MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.
```

```
        " Delete the IAM role. "
        TRY.
            lo_iam->deleterole( iv_rolename = iv_role_name ).
            MESSAGE 'IAM role deleted.' TYPE 'I'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'W'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
modifiable.' TYPE 'E'.
        ENDTRY.

      CATCH /aws1/cx_rt_service_generic INTO lo_exception.
        DATA(lv_error) = lo_exception->get_longtext( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a primeira função do Lambda, que simplesmente incrementa o valor especificado.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "increment",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/swift-server/swift-aws-lambda-runtime.git",
            branch: "main"),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "increment",
            dependencies: [
                .product(name: "AWSLambdaRuntime", package: "swift-aws-lambda-
runtime"),
            ],
            path: "Sources"
        )
    ]
)

import Foundation
import AWSLambdaRuntime

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
```

```
struct Request: Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The number to act upon.
    let number: Int
}

/// The contents of the response sent back to the client. This must be
/// `Encodable`.
struct Response: Encodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}


/// The Lambda function body.
///
/// - Parameters:
///   - event: The `Request` describing the request made by the
///     client.
///   - context: A `LambdaContext` describing the context in
///     which the lambda function is running.
///
/// - Returns: A `Response` object that will be encoded to JSON and sent
///   to the client by the Lambda runtime.
let incrementLambdaRuntime = LambdaRuntime {
        (event: Request, context: LambdaContext) -> Response in
    let action = event.action
    var answer: Int?

    if action != "increment" {
        context.logger.error("Unrecognized operation: \"\(action)\". The only
 supported action is \"increment\".")
    } else {
        answer = event.number + 1
        context.logger.info("The calculated answer is \(answer!).")
    }

    let response = Response(answer: answer)
    return response
}

// Run the Lambda runtime code.
```

```
try await incrementLambdaRuntime.run()
```

Defina a segunda função do Lambda, que executa uma operação aritmética em dois números.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "calculator",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/swift-server/swift-aws-lambda-runtime.git",
            branch: "main"),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "calculator",
            dependencies: [
                .product(name: "AWSLambdaRuntime", package: "swift-aws-lambda-
runtime"),
            ],
            path: "Sources"
        )
    ]
)

import Foundation
```

```
import AWSLambdaRuntime

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct Request: Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The first number to act upon.
    let x: Int
    /// The second number to act upon.
    let y: Int
}

/// A dictionary mapping operation names to closures that perform that
/// operation and return the result.
let actions = [
    "plus": { (x: Int, y: Int) -> Int in
        return x + y
    },
    "minus": { (x: Int, y: Int) -> Int in
        return x - y
    },
    "times": { (x: Int, y: Int) -> Int in
        return x * y
    },
    "divided-by": { (x: Int, y: Int) -> Int in
        return x / y
    }
]

/// The contents of the response sent back to the client. This must be
/// `Encodable`.
struct Response: Encodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}


/// The Lambda function's entry point. Called by the Lambda runtime.
///
/// - Parameters:
///   - event: The `Request` describing the request made by the
///     client.
```

```
///   - context: A `LambdaContext` describing the context in
///     which the lambda function is running.
///
/// - Returns: A `Response` object that will be encoded to JSON and sent
///   to the client by the Lambda runtime.
let calculatorLambdaRuntime = LambdaRuntime {
        (_ event: Request, context: LambdaContext) -> Response in
    let action = event.action
    var answer: Int?
    var actionFunc: ((Int, Int) -> Int)?

    // Get the closure to run to perform the calculation.

    actionFunc = await actions[action]

    guard let actionFunc else {
        context.logger.error("Unrecognized operation '\(action)\'")
        return Response(answer: nil)
    }

    // Perform the calculation and return the answer.

    answer = actionFunc(event.x, event.y)

    guard let answer else {
        context.logger.error("Error computing \(event.x) \(action) \(event.y)")
    }
    context.logger.info("\(event.x) \(action) \(event.y) = \(answer)")

    return Response(answer: answer)
}

try await calculatorLambdaRuntime.run()
```

Defina o programa principal que invocará as duas funções do Lambda.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.
```

```
import PackageDescription

let package = Package(
    name: "lambda-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "lambda-basics",
            dependencies: [
                .product(name: "AWSLambda", package: "aws-sdk-swift"),
                .product(name: "AWSIAM", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources"
        )
    ]
)

//
/// An example demonstrating a variety of important AWS Lambda functions.

import ArgumentParser
import AWSIAM
import SmithyWaitersAPI
import AWSClientRuntime
```

```
import AWSLambda
import Foundation

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct IncrementRequest: Encodable, Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The number to act upon.
    let number: Int
}

struct Response: Encodable, Decodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}

struct CalculatorRequest: Encodable, Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The first number to act upon.
    let x: Int
    /// The second number to act upon.
    let y: Int
}

let exampleName = "SwiftLambdaRoleExample"
let basicsFunctionName = "lambda-basics-function"

/// The ARN of the standard IAM policy for execution of Lambda functions.
let policyARN = "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"

struct ExampleCommand: ParsableCommand {
    // -MARK: Command arguments
    @Option(help: "Name of the IAM Role to use for the Lambda functions")
    var role = exampleName
    @Option(help: "Zip archive containing the 'increment' lambda function")
    var incpath: String
    @Option(help: "Zip archive containing the 'calculator' lambda function")
    var calcpath: String
    @Option(help: "Name of the Amazon S3 Region to use (default: us-east-1)")
    var region = "us-east-1"
```

```
    static var configuration = CommandConfiguration(
        commandName: "lambda-basics",
        abstract: """
        This example demonstrates several common operations using AWS Lambda.
        """,
        discussion: """
        """
    )

    /// Returns the specified IAM role object.
    ///
    /// - Parameters:
    ///   - iamClient: `IAMClient` to use when looking for the role.
    ///   - roleName: The name of the role to check.
    ///
    /// - Returns: The `IAMClientTypes.Role` representing the specified role.
    func getRole(iamClient: IAMClient, roleName: String) async throws
                 -> IAMClientTypes.Role {
        do {
            let roleOutput = try await iamClient.getRole(
                input: GetRoleInput(
                    roleName: roleName
                )
            )

            guard let role = roleOutput.role else {
                throw ExampleError.roleNotFound
            }
            return role
        } catch {
            throw ExampleError.roleNotFound
        }
    }

    /// Create the AWS IAM role that will be used to access AWS Lambda.
    ///
    /// - Parameters:
    ///   - iamClient: The AWS `IAMClient` to use.
    ///   - roleName: The name of the AWS IAM role to use for Lambda.
    ///
    /// - Throws: `ExampleError.roleCreateError`
    ///
    /// - Returns: The `IAMClientTypes.Role` struct that describes the new role.
```

```
    func createRoleForLambda(iamClient: IAMClient, roleName: String) async throws
-> IAMClientTypes.Role {
        let output = try await iamClient.createRole(
            input: CreateRoleInput(
                assumeRolePolicyDocument:
                """
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"Service": "lambda.amazonaws.com"},
                            "Action": "sts:AssumeRole"
                        }
                    ]
                }
                """,
                roleName: roleName
            )
        )

        guard let role = output.role else {
            throw ExampleError.roleCreateError
        }

        // Wait for the role to be ready for use.

        _ = try await iamClient.waitUntilRoleExists(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetRoleInput(roleName: roleName)
        )

        return role
    }

    /// Detect whether or not the AWS Lambda function with the specified name
    /// exists, by requesting its function information.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
```

```
    ///   - name: The name of the AWS Lambda function to find.
    ///
    /// - Returns: `true` if the Lambda function exists. Otherwise `false`.
    func doesLambdaFunctionExist(lambdaClient: LambdaClient, name: String) async
 -> Bool {
        do {
            _ = try await lambdaClient.getFunction(
                input: GetFunctionInput(functionName: name)
            )
        } catch {
            return false
        }

        return true
    }

    /// Create the specified AWS Lambda function.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to create.
    ///   - roleArn: The ARN of the role to apply to the function.
    ///   - path: The path of the Zip archive containing the function.
    ///
    /// - Returns: `true` if the AWS Lambda was successfully created; `false`
    ///   if it wasn't.
    func createFunction(lambdaClient: LambdaClient, functionName: String,
                                roleArn: String?, path: String) async throws ->
 Bool {
        do {
            // Read the Zip archive containing the AWS Lambda function.

            let zipUrl = URL(fileURLWithPath: path)
            let zipData = try Data(contentsOf: zipUrl)

            // Create the AWS Lambda function that runs the specified code,
            // using the name given on the command line. The Lambda function
            // will run using the Amazon Linux 2 runtime.

            _ = try await lambdaClient.createFunction(
                input: CreateFunctionInput(
                    code: LambdaClientTypes.FunctionCode(zipFile: zipData),
                    functionName: functionName,
                    handler: "handle",
```

```
                    role: roleArn,
                    runtime: .providedal2
                )
            )
        } catch {
            print("*** Error creating Lambda function:")
            dump(error)
            return false
        }

        // Wait for a while to be sure the function is done being created.

        let output = try await lambdaClient.waitUntilFunctionActiveV2(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetFunctionInput(functionName: functionName)
        )

        switch output.result {
            case .success:
                return true
            case .failure:
                return false
        }
    }

    /// Update the AWS Lambda function with new code to run when the function
    /// is invoked.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to update.
    ///   - path: The pathname of the Zip file containing the packaged Lambda
    ///     function.
    /// - Throws: `ExampleError.zipFileReadError`
    /// - Returns: `true` if the function's code is updated successfully.
    ///   Otherwise, returns `false`.
    func updateFunctionCode(lambdaClient: LambdaClient, functionName: String,
                            path: String) async throws -> Bool {
        let zipUrl = URL(fileURLWithPath: path)
        let zipData: Data
```

```
        // Read the function's Zip file.

        do {
            zipData = try Data(contentsOf: zipUrl)
        } catch {
            throw ExampleError.zipFileReadError
        }

        // Update the function's code and wait for the updated version to be
        // ready for use.

        do {
            _ = try await lambdaClient.updateFunctionCode(
                input: UpdateFunctionCodeInput(
                    functionName: functionName,
                    zipFile: zipData
                )
            )
        } catch {
            return false
        }

        let output = try await lambdaClient.waitUntilFunctionUpdatedV2(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetFunctionInput(
                functionName: functionName
            )
        )

        switch output.result {
            case .success:
                return true
            case .failure:
                return false
        }
    }

    /// Tell the server-side component to log debug output by setting its
    /// environment's `LOG_LEVEL` to `DEBUG`.
```

```
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to enable debug
    ///     logging for.
    ///
    /// - Throws: `ExampleError.environmentResponseMissingError`,
    ///   `ExampleError.updateFunctionConfigurationError`,
    ///   `ExampleError.environmentVariablesMissingError`,
    ///   `ExampleError.logLevelIncorrectError`,
    ///   `ExampleError.updateFunctionConfigurationError`
    func enableDebugLogging(lambdaClient: LambdaClient, functionName: String)
 async throws {
        let envVariables = [
            "LOG_LEVEL": "DEBUG"
        ]
        let environment = LambdaClientTypes.Environment(variables: envVariables)

        do {
            let output = try await lambdaClient.updateFunctionConfiguration(
                input: UpdateFunctionConfigurationInput(
                    environment: environment,
                    functionName: functionName
                )
            )

            guard let response = output.environment else {
                throw ExampleError.environmentResponseMissingError
            }

            if response.error != nil {
                throw ExampleError.updateFunctionConfigurationError
            }

            guard let retVariables = response.variables else {
                throw ExampleError.environmentVariablesMissingError
            }

            for envVar in retVariables {
                if envVar.key == "LOG_LEVEL" && envVar.value != "DEBUG" {
                    print("*** Log level is not set to DEBUG!")
                    throw ExampleError.logLevelIncorrectError
                }
            }
```

```
        } catch {
            throw ExampleError.updateFunctionConfigurationError
        }
    }

    /// Returns an array containing the names of all AWS Lambda functions
    /// available to the user.
    ///
    /// - Parameter lambdaClient: The `IAMClient` to use.
    ///
    /// - Throws: `ExampleError.listFunctionsError`
    ///
    /// - Returns: An array of lambda function name strings.
    func getFunctionNames(lambdaClient: LambdaClient) async throws -> [String] {
        let pages = lambdaClient.listFunctionsPaginated(
            input: ListFunctionsInput()
        )

        var functionNames: [String] = []

        for try await page in pages {
            guard let functions = page.functions else {
                throw ExampleError.listFunctionsError
            }

            for function in functions {
                functionNames.append(function.functionName ?? "<unknown>")
            }
        }

        return functionNames
    }

    /// Invoke the Lambda function to increment a value.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - number: The number to increment.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: An integer number containing the incremented value.
    func invokeIncrement(lambdaClient: LambdaClient, number: Int) async throws ->
 Int {
```

```
        do {
            let incRequest = IncrementRequest(action: "increment", number:
number)
            let incData = try! JSONEncoder().encode(incRequest)

            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: incData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }
    }

    /// Invoke the calculator Lambda function.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - action: Which arithmetic operation to perform: "plus", "minus",
    ///     "times", or "divided-by".
    ///   - x: The first number to use in the computation.
    ///   - y: The second number to use in the computation.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: The computed answer as an `Int`.
    func invokeCalculator(lambdaClient: LambdaClient, action: String, x: Int, y:
Int) async throws -> Int {
        do {
            let calcRequest = CalculatorRequest(action: action, x: x, y: y)
            let calcData = try! JSONEncoder().encode(calcRequest)
```

```
            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: calcData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }

    }

    /// Perform the example's tasks.
    func basics() async throws {
        let iamClient = try await IAMClient(
            config: IAMClient.IAMClientConfiguration(region: region)
        )

        let lambdaClient = try await LambdaClient(
            config: LambdaClient.LambdaClientConfiguration(region: region)
        )

        /// The IAM role to use for the example.
        var iamRole: IAMClientTypes.Role

        // Look for the specified role. If it already exists, use it. If not,
        // create it and attach the desired policy to it.

        do {
            iamRole = try await getRole(iamClient: iamClient, roleName: role)
        } catch ExampleError.roleNotFound {
            // The role wasn't found, so create it and attach the needed
```

```
            // policy.

            iamRole = try await createRoleForLambda(iamClient: iamClient,
roleName: role)

            do {
                _ = try await iamClient.attachRolePolicy(
                    input: AttachRolePolicyInput(policyArn: policyARN, roleName:
role)
                )
            } catch {
                throw ExampleError.policyError
            }
        }

        // Give the policy time to attach to the role.

        sleep(5)

        // Look to see if the function already exists. If it does, throw an
        // error.

        if await doesLambdaFunctionExist(lambdaClient: lambdaClient, name:
basicsFunctionName) {
            throw ExampleError.functionAlreadyExists
        }

        // Create, then invoke, the "increment" version of the calculator
        // function.

        print("Creating the increment Lambda function...")
        if try await createFunction(lambdaClient: lambdaClient, functionName:
basicsFunctionName,
                                    roleArn: iamRole.arn, path: incpath) {
            print("Running increment function calls...")
            for number in 0...4 {
                do {
                    let answer = try await invokeIncrement(lambdaClient:
lambdaClient, number: number)
                    print("Increment \(number) = \(answer)")
                } catch {
                    print("Error incrementing \(number): ",
error.localizedDescription)
                }
```

```
            }
        } else {
            print("*** Failed to create the increment function.")
        }

        // Enable debug logging.

        print("\nEnabling debug logging...")
        try await enableDebugLogging(lambdaClient: lambdaClient, functionName:
basicsFunctionName)

        // Change it to a basic arithmetic calculator. Then invoke it a few
        // times.

        print("\nReplacing the Lambda function with a calculator...")

        if try await updateFunctionCode(lambdaClient: lambdaClient, functionName:
basicsFunctionName,
                                        path: calcpath) {
            print("Running calculator function calls...")
            for x in [6, 10] {
                for y in [2, 4] {
                    for action in ["plus", "minus", "times", "divided-by"] {
                        do {
                            let answer = try await invokeCalculator(lambdaClient:
lambdaClient, action: action, x: x, y: y)
                            print("\(x) \(action) \(y) = \(answer)")
                        } catch {
                            print("Error calculating \(x) \(action) \(y): ",
error.localizedDescription)
                        }
                    }
                }
            }
        }

        // List all lambda functions.

        let functionNames = try await getFunctionNames(lambdaClient:
lambdaClient)

        if functionNames.count > 0 {
            print("\nAWS Lambda functions available on your account:")
            for name in functionNames {
```

```
                print("  \(name)")
            }
        }

        // Delete the lambda function.

        print("Deleting lambda function...")

        do {
            _ = try await lambdaClient.deleteFunction(
                input: DeleteFunctionInput(
                    functionName: "lambda-basics-function"
                )
            )
        } catch {
            print("Error: Unable to delete the function.")
        }

        // Detach the role from the policy, then delete the role.

        print("Deleting the AWS IAM role...")

        do {
            _ = try await iamClient.detachRolePolicy(
                input: DetachRolePolicyInput(
                    policyArn: policyARN,
                    roleName: role
                )
            )
            _ = try await iamClient.deleteRole(
                input: DeleteRoleInput(
                    roleName: role
                )
            )
        } catch {
            throw ExampleError.deleteRoleError
        }
    }
}

// -MARK: - Entry point

/// The program's asynchronous entry point.
@main
```

```
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.basics()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}


/// Errors thrown by the example's functions.
enum ExampleError: Error {
    /// An AWS Lambda function with the specified name already exists.
    case functionAlreadyExists
    /// The specified role doesn't exist.
    case roleNotFound
    /// Unable to create the role.
    case roleCreateError
    /// Unable to delete the role.
    case deleteRoleError
    /// Unable to attach a policy to the role.
    case policyError
    /// Unable to get the executable directory.
    case executableNotFound
    /// An error occurred creating a lambda function.
    case createLambdaError
    /// An error occurred invoking the lambda function.
    case invokeError
    /// No answer received from the invocation.
    case noAnswerReceived
    /// Unable to list the AWS Lambda functions.
    case listFunctionsError
    /// Unable to update the AWS Lambda function.
    case updateFunctionError
    /// Unable to update the function configuration.
    case updateFunctionConfigurationError
    /// The environment response is missing after an
    /// UpdateEnvironmentConfiguration attempt.
    case environmentResponseMissingError
    /// The environment variables are missing from the EnvironmentResponse and
```

```
    /// no errors occurred.
    case environmentVariablesMissingError
    /// The log level is incorrect after attempting to set it.
    case logLevelIncorrectError
    /// Unable to load the AWS Lambda function's Zip file.
    case zipFileReadError

    var errorDescription: String? {
        switch self {
        case .functionAlreadyExists:
            return "An AWS Lambda function with that name already exists."
        case .roleNotFound:
            return "The specified role doesn't exist."
        case .deleteRoleError:
            return "Unable to delete the AWS IAM role."
        case .roleCreateError:
            return "Unable to create the specified role."
        case .policyError:
            return "An error occurred attaching the policy to the role."
        case .executableNotFound:
            return "Unable to find the executable program directory."
        case .createLambdaError:
            return "An error occurred creating a lambda function."
        case .invokeError:
            return "An error occurred invoking a lambda function."
        case .noAnswerReceived:
            return "No answer received from the lambda function."
        case .listFunctionsError:
            return "Unable to list the AWS Lambda functions."
        case .updateFunctionError:
            return "Unable to update the AWS lambda function."
        case .updateFunctionConfigurationError:
            return "Unable to update the AWS lambda function configuration."
        case .environmentResponseMissingError:
            return "The environment is missing from the response after updating
the function configuration."
        case .environmentVariablesMissingError:
            return "While no error occurred, no environment variables were
returned following function configuration."
        case .logLevelIncorrectError:
            return "The log level is incorrect after attempting to set it to
DEBUG."
        case .zipFileReadError:
            return "Unable to read the AWS Lambda function."
```

```
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações para Lambda usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Lambda com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Lambda e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Lambda AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Lambda.

Exemplos

- Usar CreateAlias com uma CLI
- Use CreateFunctioncom um AWS SDK ou CLI
- Usar DeleteAlias com uma CLI
- Use DeleteFunctioncom um AWS SDK ou CLI
- Usar DeleteFunctionConcurrency com uma CLI
- Usar DeleteProvisionedConcurrencyConfig com uma CLI
- Usar GetAccountSettings com uma CLI

- Usar GetAlias com uma CLI
- Use GetFunctioncom um AWS SDK ou CLI
- Usar GetFunctionConcurrency com uma CLI
- Usar GetFunctionConfiguration com uma CLI
- Usar GetPolicy com uma CLI
- Usar GetProvisionedConcurrencyConfig com uma CLI
- Use Invoke com um AWS SDK ou CLI
- Use ListFunctionscom um AWS SDK ou CLI
- Usar ListProvisionedConcurrencyConfigs com uma CLI
- Usar ListTags com uma CLI
- Usar ListVersionsByFunction com uma CLI
- Usar PublishVersion com uma CLI
- Usar PutFunctionConcurrency com uma CLI
- Usar PutProvisionedConcurrencyConfig com uma CLI
- Usar RemovePermission com uma CLI
- Usar TagResource com uma CLI
- Usar UntagResource com uma CLI
- Usar UpdateAlias com uma CLI
- Use UpdateFunctionCodecom um AWS SDK ou CLI
- Use UpdateFunctionConfigurationcom um AWS SDK ou CLI

Usar **`CreateAlias`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateAlias`.

CLI

AWS CLI

Para criar um alias para uma função do Lambda

O seguinte exemplo de `create-alias` cria um alias chamado `LIVE` que aponta para a versão 1 da função `my-function` do Lambda.

```
aws lambda create-alias \
```

```
    --function-name my-function \
    --description "alias for live version of function" \
    --function-version 1 \
    --name LIVE
```

Saída:

```
{
    "FunctionVersion": "1",
    "Name": "LIVE",
    "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:LIVE",
    "RevisionId": "873282ed-4cd3-4dc8-a069-d0c647e470c6",
    "Description": "alias for live version of function"
}
```

Para obter mais informações, consulte Configurando aliases de funções do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte CreateAliasem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um novo alias do Lambda para versão e configuração de roteamento especificadas a fim de indicar o percentual de solicitações de invocação que ele receberá.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias
 for version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Para obter detalhes da API, consulte CreateAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um novo alias do Lambda para versão e configuração de roteamento especificadas a fim de indicar o percentual de solicitações de invocação que ele receberá.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias
 for version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Para obter detalhes da API, consulte CreateAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateFunction`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateFunction`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma API REST com integração de proxy de função
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a new Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function.</param>
    /// <param name="s3Bucket">The Amazon Simple Storage Service (Amazon S3)
    /// bucket where the zip file containing the code is located.</param>
```

```
    /// <param name="s3Key">The Amazon S3 key of the zip file.</param>
    /// <param name="role">The Amazon Resource Name (ARN) of a role with the
    /// appropriate Lambda permissions.</param>
    /// <param name="handler">The name of the handler function.</param>
    /// <returns>The Amazon Resource Name (ARN) of the newly created
    /// Lambda function.</returns>
    public async Task<string> CreateLambdaFunctionAsync(
        string functionName,
        string s3Bucket,
        string s3Key,
        string role,
        string handler)
    {
        // Defines the location for the function code.
        // S3Bucket - The S3 bucket where the file containing
        //            the source code is stored.
        // S3Key    - The name of the file containing the code.
        var functionCode = new FunctionCode
        {
            S3Bucket = s3Bucket,
            S3Key = s3Key,
        };

        var createFunctionRequest = new CreateFunctionRequest
        {
            FunctionName = functionName,
            Description = "Created by the Lambda .NET API",
            Code = functionCode,
            Handler = handler,
            Runtime = Runtime.Dotnet6,
            Role = role,
        };

        var reponse = await
_lambdaService.CreateFunctionAsync(createFunctionRequest);
        return reponse.FunctionArn;
    }
```

- Para obter detalhes da API, consulte CreateFunctiona Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
 (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#if USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);

        // Assume the AWS Lambda function was built in Docker with same
 architecture
        // as this code.
#if  defined(__x86_64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif
        request.SetRole(roleArn);
        request.SetHandler(LAMBDA_HANDLER_NAME);
        request.SetPublish(true);
        Aws::Lambda::Model::FunctionCode code;
```

```
        std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();

        code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
 buffer.str().c_str(),
                                               buffer.str().length()));
        request.SetCode(code);

        Aws::Lambda::Model::CreateFunctionOutcome outcome =
 client.CreateFunction(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda function was successfully created. " <<
 seconds
                      << " seconds elapsed." << std::endl;
            break;
        }

        else {
            std::cerr << "Error with CreateFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte CreateFunctiona Referência AWS SDK para C++ da API.

CLI

AWS CLI

Criar uma função do Lambda

O exemplo `create-function` a seguir cria uma função do Lambda denominada `my-function`.

```
aws lambda create-function \
    --function-name my-function \
    --runtime nodejs22.x \
    --zip-file fileb://my-function.zip \
    --handler my-function.handler \
    --role arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-
tges6bf4
```

Conteúdo de `my-function.zip`:

```
This file is a deployment package that contains your function code and any
 dependencies.
```

Saída:

```
{
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "PFn4S+er27qk+UuZSTKEQfNKG/XNn7QJs90mJgq6oH8=",
    "FunctionName": "my-function",
    "CodeSize": 308,
    "RevisionId": "873282ed-4cd3-4dc8-a069-d0c647e470c6",
    "MemorySize": 128,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
    "Version": "$LATEST",
    "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-
zgur6bf4",
    "Timeout": 3,
    "LastModified": "2025-10-14T22:26:11.234+0000",
```

```
    "Handler": "my-function.handler",
    "Runtime": "nodejs22.x",
    "Description": ""
}
```

Para obter mais informações, consulte Configurar memória para uma função do Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte CreateFunctionem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}
```

```
// CreateFunction creates a new Lambda function from code contained in the
 zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
 until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
 string, handlerName string,
 iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
  Code:         &types.FunctionCode{ZipFile: zipPackage.Bytes()},
  FunctionName: aws.String(functionName),
  Role:         iamRoleArn,
  Handler:      aws.String(handlerName),
  Publish:      true,
  Runtime:      types.RuntimePython39,
 })
 if err != nil {
  var resConflict *types.ResourceConflictException
  if errors.As(err, &resConflict) {
   log.Printf("Function %v already exists.\n", functionName)
   state = types.StateActive
  } else {
   log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
  }
 } else {
  waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}
```

- Para obter detalhes da API, consulte CreateFunctiona Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
 Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();
```

```
            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateFunctiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const createFunction = async (funcName, roleArn) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${funcName}.zip`);

  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte CreateFunctiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String? {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
```

```
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

- Para obter detalhes da API, consulte a CreateFunctionreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function createFunction($functionName, $role, $bucketName, $handler)
    {
        //This assumes the Lambda function is in an S3 bucket.
        return $this->customWaiter(function () use ($functionName, $role,
$bucketName, $handler) {
```

```
            return $this->lambdaClient->createFunction([
                'Code' => [
                    'S3Bucket' => $bucketName,
                    'S3Key' => $functionName,
                ],
                'FunctionName' => $functionName,
                'Role' => $role['Arn'],
                'Runtime' => 'python3.9',
                'Handler' => "$handler.lambda_handler",
            ]);
        });
    }
```

- Para obter detalhes da API, consulte CreateFunctiona Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria uma nova função C# (dotnetcore1.0 runtime) nomeada no MyFunction AWS Lambda, fornecendo os binários compilados para a função a partir de um arquivo zip no sistema de arquivos local (caminhos relativos ou absolutos podem ser usados). As funções Lambda do C# especificam o manipulador da função usando a AssemblyName designação::::. Namespace.ClassName MethodName É necessário substituir adequadamente o nome da montagem (sem o sufixo .dll), o namespace, o nome da classe e o nome do método da especificação do manipulador. A nova função terá as variáveis de ambiente “envvar1” e “envvar2” configuradas com base nos valores fornecidos.

```
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -ZipFilename .\MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

Saída:

```
CodeSha256       : /NgBMd...gq71I=
```

```
CodeSize         : 214784
DeadLetterConfig :
Description      : My C# Lambda Function
Environment      : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn      : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName     : MyFunction
Handler          : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn        :
LastModified     : 2016-12-29T23:50:14.207+0000
MemorySize       : 128
Role             : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime          : dotnetcore1.0
Timeout          : 3
Version          : $LATEST
VpcConfig        :
```

Exemplo 2: este exemplo é semelhante ao anterior, com a exceção de que os binários da função são carregados primeiramente em um bucket do Amazon S3 (que deve estar na mesma região da função do Lambda desejada) e o objeto resultante do S3 será referenciado ao criar a função.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -
File .\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName amzn-s3-demo-bucket `
        -Key MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- Para obter detalhes da API, consulte CreateFunctionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma nova função C# (dotnetcore1.0 runtime) nomeada no MyFunction AWS Lambda, fornecendo os binários compilados para a função a partir de um arquivo zip no sistema de arquivos local (caminhos relativos ou absolutos podem ser usados). As funções Lambda do C# especificam o manipulador da função usando a AssemblyName designação::::. Namespace.ClassName MethodName É necessário substituir adequadamente

o nome da montagem (sem o sufixo .dll), o namespace, o nome da classe e o nome do método da especificação do manipulador. A nova função terá as variáveis de ambiente “envvar1” e “envvar2” configuradas com base nos valores fornecidos.

```
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -ZipFilename .\MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

Saída:

```
CodeSha256       : /NgBMd...gq71I=
CodeSize         : 214784
DeadLetterConfig :
Description      : My C# Lambda Function
Environment      : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn      : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName     : MyFunction
Handler          : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn        :
LastModified     : 2016-12-29T23:50:14.207+0000
MemorySize       : 128
Role             : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime          : dotnetcore1.0
Timeout          : 3
Version          : $LATEST
VpcConfig        :
```

Exemplo 2: este exemplo é semelhante ao anterior, com a exceção de que os binários da função são carregados primeiramente em um bucket do Amazon S3 (que deve estar na mesma região da função do Lambda desejada) e o objeto resultante do S3 será referenciado ao criar a função.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -
File .\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName amzn-s3-demo-bucket `
        -Key MyFunctionBinaries.zip `
```

```
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- Para obter detalhes da API, consulte CreateFunctionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def create_function(
        self, function_name, handler_name, iam_role, deployment_package
    ):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function.
 This
                              must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the
 function
                                     code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
```

```
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime="python3.9",
                Role=iam_role.arn,
                Handler=handler_name,
                Code={"ZipFile": deployment_package},
                Publish=True,
            )
            function_arn = response["FunctionArn"]
            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)
            logger.info(
                "Created function '%s' with ARN: '%s'.",
                function_name,
                response["FunctionArn"],
            )
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn
```

- Para obter detalhes da API, consulte a CreateFunctionReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client
```

```
  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
 code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
 #{e.message}")
  end
```

- Para obter detalhes da API, consulte CreateFunctiona Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Create a function, uploading from a zip file.
     */
    pub async fn create_function(&self, zip_file: PathBuf) -> Result<String,
anyhow::Error> {
        let code = self.prepare_function(zip_file, None).await?;

        let key = code.s3_key().unwrap().to_string();

        let role = self.create_role().await.map_err(|e| anyhow!(e))?;

        info!("Created iam role, waiting 15s for it to become active");
        tokio::time::sleep(Duration::from_secs(15)).await;

        info!("Creating lambda function {}", self.lambda_name);
        let _ = self
            .lambda_client
            .create_function()
            .function_name(self.lambda_name.clone())
            .code(code)
            .role(role.arn())
            .runtime(aws_sdk_lambda::types::Runtime::Providedal2)
            .handler("_unused")
            .send()
            .await
            .map_err(anyhow::Error::from)?;
```

```
        self.wait_for_function_ready().await?;

        self.lambda_client
            .publish_version()
            .function_name(self.lambda_name.clone())
            .send()
            .await?;

        Ok(key)
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --
output-format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }
```

- Para obter detalhes da API, consulte a CreateFunctionreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_lmd->createfunction(
            iv_functionname = iv_function_name
            iv_runtime = `python3.9`
            iv_role = iv_role_arn
            iv_handler = iv_handler
            io_code = io_zip_file
            iv_description = 'AWS Lambda code example' ).
        MESSAGE 'Lambda function created.' TYPE 'I'.
      CATCH /aws1/cx_lmdcodesigningcfgno00.
        MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdcodestorageexcdex.
        MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
      CATCH /aws1/cx_lmdcodeverification00.
        MESSAGE 'Code signature failed one or more validation checks for
signature mismatch or expiration.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidcodesigex.
        MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateFunctionreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        do {
            // Read the Zip archive containing the AWS Lambda function.

            let zipUrl = URL(fileURLWithPath: path)
            let zipData = try Data(contentsOf: zipUrl)

            // Create the AWS Lambda function that runs the specified code,
            // using the name given on the command line. The Lambda function
            // will run using the Amazon Linux 2 runtime.

            _ = try await lambdaClient.createFunction(
                input: CreateFunctionInput(
                    code: LambdaClientTypes.FunctionCode(zipFile: zipData),
                    functionName: functionName,
                    handler: "handle",
                    role: roleArn,
                    runtime: .providedal2
                )
            )
        } catch {
            print("*** Error creating Lambda function:")
```

```
            dump(error)
            return false
        }
```

- Para obter detalhes da API, consulte CreateFunctiona referência da API AWS SDK for Swift.

Usar **`DeleteAlias`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteAlias`.

CLI

AWS CLI

Para excluir um alias de uma função do Lambda

O seguinte exemplo de `delete-alias` exclui o alias chamado `LIVE` da função `my-function` do Lambda.

```
aws lambda delete-alias \
    --function-name my-function \
    --name LIVE
```

Este comando não produz saída.

Para obter mais informações, consulte Configurando aliases de funções do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte DeleteAliasem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui o alias da função do Lambda mencionado no comando.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Para obter detalhes da API, consulte DeleteAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o alias da função do Lambda mencionado no comando.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Para obter detalhes da API, consulte DeleteAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteFunction`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteFunction`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar uma API REST com integração de proxy de função
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an AWS Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
```

```
    /// delete.</param>
    /// <returns>A Boolean value that indicates the success of the action.</
returns>
    public async Task<bool> DeleteFunctionAsync(string functionName)
    {
        var request = new DeleteFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.DeleteFunctionAsync(request);

        // A return value of NoContent means that the request was processed.
        // In this case, the function was deleted, and the return value
        // is intentionally blank.
        return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
    }
```

- Para obter detalhes da API, consulte DeleteFunctiona Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
 (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::DeleteFunctionRequest request;
```

```
    request.SetFunctionName(LAMBDA_NAME);

    Aws::Lambda::Model::DeleteFunctionOutcome outcome = client.DeleteFunction(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda function was successfully deleted." <<
 std::endl;
    }
    else {
        std::cerr << "Error with Lambda::DeleteFunction. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Para obter detalhes da API, consulte DeleteFunctiona Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: excluir uma função do Lambda pelo nome da função

O exemplo de `delete-function` a seguir exclui a função do Lambda denominada `my-function` ao especificar o nome da função.

```
aws lambda delete-function \
    --function-name my-function
```

Este comando não produz saída.

Exemplo 2: excluir uma função do Lambda pelo ARN da função

O exemplo `delete-function` a seguir exclui a função do Lambda denominada `my-function` ao especificar o ARN da função.

```
aws lambda delete-function \
    --function-name arn:aws:lambda:us-west-2:123456789012:function:my-function
```

Este comando não produz saída.

Exemplo 3: excluir uma função do Lambda pelo ARN parcial da função

O exemplo `delete-function` a seguir exclui a função do Lambda denominada `my-function` ao especificar o ARN parcial da função.

```
aws lambda delete-function \
    --function-name 123456789012:function:my-function
```

Este comando não produz saída.

Para obter mais informações, consulte Configurar opções da função do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte DeleteFunctionem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)
```

```
// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
 string) {
 _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
 }
}
```

- Para obter detalhes da API, consulte DeleteFunctiona Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which
 is used to interact with the AWS Lambda service
     * @param functionName  the name of the Lambda function to be deleted
     *
```

```
     * @throws LambdaException if an error occurs while deleting the Lambda
 function
     */
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteFunctiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * @param {string} funcName
 */
const deleteFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteFunctiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Para obter detalhes da API, consulte a DeleteFunctionreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteFunctiona Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo exclui uma versão específica de uma função do Lambda

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Para obter detalhes da API, consulte DeleteFunctionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui uma versão específica de uma função do Lambda

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Para obter detalhes da API, consulte DeleteFunctionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def delete_function(self, function_name):
        """
        Deletes a Lambda function.

        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise
```

- Para obter detalhes da API, consulte a DeleteFunctionReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
```

```
  @logger.level = Logger::WARN
end

# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name: function_name
  )
  print 'Done!'.green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- Para obter detalhes da API, consulte DeleteFunctiona Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /** Delete a function and its role, and if possible or necessary, its
associated code object and bucket. */
   pub async fn delete_function(
       &self,
       location: Option<String>,
   ) -> (
       Result<DeleteFunctionOutput, anyhow::Error>,
       Result<DeleteRoleOutput, anyhow::Error>,
       Option<Result<DeleteObjectOutput, anyhow::Error>>,
   ) {
       info!("Deleting lambda function {}", self.lambda_name);
       let delete_function = self
```

```
            .lambda_client
            .delete_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        info!("Deleting iam role {}", self.role_name);
        let delete_role = self
            .iam_client
            .delete_role()
            .role_name(self.role_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        let delete_object: Option<Result<DeleteObjectOutput, anyhow::Error>> =
            if let Some(location) = location {
                info!("Deleting object {location}");
                Some(
                    self.s3_client
                        .delete_object()
                        .bucket(self.bucket.clone())
                        .key(location)
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                info!(?location, "Skipping delete object");
                None
            };

        (delete_function, delete_role, delete_object)
    }
```

- Para obter detalhes da API, consulte a DeleteFunctionreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_lmd->deletefunction( iv_functionname = iv_function_name ).
       MESSAGE 'Lambda function deleted.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteFunctionreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        do {
            _ = try await lambdaClient.deleteFunction(
                input: DeleteFunctionInput(
                    functionName: "lambda-basics-function"
                )
            )
        } catch {
            print("Error: Unable to delete the function.")
        }
```

- Para obter detalhes da API, consulte DeleteFunctiona referência da API AWS SDK for Swift.

Usar **`DeleteFunctionConcurrency`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteFunctionConcurrency`.

CLI

AWS CLI

Para remover o limite reservado de execução simultânea de uma função

O seguinte exemplo de `delete-function-concurrency` exclui o limite reservado de execução simultânea da função `my-function`.

```
aws lambda delete-function-concurrency \
    --function-name  my-function
```

Este comando não produz saída.

Para obter mais informações, consulte Como reservar simultaneidade para uma função do Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte DeleteFunctionConcurrencyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo remove a simultaneidade de função da função do Lambda.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Para obter detalhes da API, consulte DeleteFunctionConcurrencyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove a simultaneidade de função da função do Lambda.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Para obter detalhes da API, consulte DeleteFunctionConcurrencyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteProvisionedConcurrencyConfig`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteProvisionedConcurrencyConfig`.

CLI

AWS CLI

Para excluir uma configuração de simultaneidade provisionada

O seguinte exemplo de `delete-provisioned-concurrency-config` exclui a configuração de simultaneidade provisionada para o alias `GREEN` da função especificada.

```
aws lambda delete-provisioned-concurrency-config \
    --function-name my-function \
    --qualifier GREEN
```

- Para obter detalhes da API, consulte DeleteProvisionedConcurrencyConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo remove a configuração de simultaneidade provisionada para um alias específico.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

- Para obter detalhes da API, consulte DeleteProvisionedConcurrencyConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove a configuração de simultaneidade provisionada para um alias específico.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

- Para obter detalhes da API, consulte DeleteProvisionedConcurrencyConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetAccountSettings`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetAccountSettings`.

CLI

AWS CLI

Para recuperar detalhes sobre sua conta em uma região AWS

O seguinte exemplo de `get-account-settings` mostra os limites do Lambda e as informações de uso da sua conta.

```
aws lambda get-account-settings
```

Saída:

```
{
    "AccountLimit": {
       "CodeSizeUnzipped": 262144000,
       "UnreservedConcurrentExecutions": 1000,
       "ConcurrentExecutions": 1000,
       "CodeSizeZipped": 52428800,
       "TotalCodeSize": 80530636800
    },
    "AccountUsage": {
       "FunctionCount": 4,
       "TotalCodeSize": 9426
    }
}
```

Para obter mais informações, consulte Limites do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte GetAccountSettingsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo é apresentado para comparar o limite da conta e o uso da conta

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Saída:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- Para obter detalhes da API, consulte GetAccountSettingsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo é apresentado para comparar o limite da conta e o uso da conta

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Saída:

```
TotalCodeSizeLimit TotalCodeSizeUsed
----------------- -----------------
       80530636800          15078795
```

- Para obter detalhes da API, consulte GetAccountSettingsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetAlias`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetAlias`.

CLI

AWS CLI

Para recuperar detalhes sobre um alias de função

O seguinte exemplo de `get-alias` mostra os detalhes do alias chamado `LIVE` na função `my-function` do Lambda.

```
aws lambda get-alias \
    --function-name my-function \
    --name LIVE
```

Saída:

```
{
    "FunctionVersion": "3",
    "Name": "LIVE",
    "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:LIVE",
    "RevisionId": "594f41fb-b85f-4c20-95c7-6ca5f2a92c93",
    "Description": "alias for live version of function"
}
```

Para obter mais informações, consulte Configurando aliases de funções do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte GetAliasem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera os pesos da configuração de roteamento para o alias de uma função do Lambda específica.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Saída:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- Para obter detalhes da API, consulte GetAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera os pesos da configuração de roteamento para o alias de uma função do Lambda específica.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Saída:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- Para obter detalhes da API, consulte GetAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetFunction`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetFunction`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Como criar sua primeira função com tecnologia sem servidor

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets information about a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function for
    /// which to retrieve information.</param>
    /// <returns>Async Task.</returns>
    public async Task<FunctionConfiguration> GetFunctionAsync(string
functionName)
    {
        var functionRequest = new GetFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.GetFunctionAsync(functionRequest);
        return response.Configuration;
    }
```

- Para obter detalhes da API, consulte GetFunctiona Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Lambda::LambdaClient client(clientConfig);

       Aws::Lambda::Model::GetFunctionRequest request;
       request.SetFunctionName(functionName);

       Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

       if (outcome.IsSuccess()) {
           std::cout << "Function retrieve.\n" <<

outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
                     << std::endl;
       }
       else {
           std::cerr << "Error with Lambda::GetFunction. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
       }
```

- Para obter detalhes da API, consulte GetFunctiona Referência AWS SDK para C++ da API.

CLI

AWS CLI

Recuperar informações sobre uma função

O exemplo `get-function` a seguir mostra informações sobre a função `my-function`.

```
aws lambda get-function \
    --function-name  my-function
```

Saída:

```
{
    "Concurrency": {
        "ReservedConcurrentExecutions": 100
    },
    "Code": {
        "RepositoryType": "S3",
        "Location": "https://awslambda-us-west-2-tasks.s3.us-
west-2.amazonaws.com/snapshots/123456789012/my-function..."
    },
    "Configuration": {
        "TracingConfig": {
            "Mode": "PassThrough"
        },
        "Version": "$LATEST",
        "CodeSha256": "5tT2qgzYUHoqwR616pZ2dpkn/0J1FrzJmlKidWaaCgk=",
        "FunctionName": "my-function",
        "VpcConfig": {
            "SubnetIds": [],
            "VpcId": "",
            "SecurityGroupIds": []
        },
        "MemorySize": 128,
        "RevisionId": "28f0fb31-5c5c-43d3-8955-03e76c5c1075",
        "CodeSize": 304,
        "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function",
        "Handler": "index.handler",
        "Role": "arn:aws:iam::123456789012:role/service-role/helloWorldPython-
role-uy3l9qyq",
        "Timeout": 3,
```

```
        "LastModified": "2025-09-24T18:20:35.054+0000",
        "Runtime": "nodejs22.x",
        "Description": ""
    }
}
```

Para obter mais informações, consulte Configurar memória para uma função do Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte GetFunctionem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}
```

```
// GetFunction gets data about the Lambda function specified by functionName.
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName
 string) types.State {
 var state types.State
 funcOutput, err := wrapper.LambdaClient.GetFunction(ctx,
 &lambda.GetFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
 } else {
  state = funcOutput.Configuration.State
 }
 return state
}
```

- Para obter detalhes da API, consulte GetFunctiona Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which
 is used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
 information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
```

```
                .functionName(functionName)
                .build();

            GetFunctionResponse response =
 awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
 response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte GetFunctiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new GetFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetFunctiona Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function getFunction($functionName)
    {
        return $this->lambdaClient->getFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Para obter detalhes da API, consulte GetFunctiona Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def get_function(self, function_name):
        """
        Gets data about a Lambda function.
```

```
        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response =
 self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- Para obter detalhes da API, consulte a GetFunctionReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
```

```
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
```

- Para obter detalhes da API, consulte GetFunctiona Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** Get the Lambda function with this Manager's name. */
    pub async fn get_function(&self) -> Result<GetFunctionOutput, anyhow::Error>
{
        info!("Getting lambda function");
        self.lambda_client
            .get_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
```

```
            .map_err(anyhow::Error::from)
    }
```

- Para obter detalhes da API, consulte a GetFunctionreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_lmd->getfunction( iv_functionname = iv_function_name ).
    " oo_result is returned for testing purposes. "
        MESSAGE 'Lambda function information retrieved.' TYPE 'I'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetFunctionreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Detect whether or not the AWS Lambda function with the specified name
    /// exists, by requesting its function information.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - name: The name of the AWS Lambda function to find.
    ///
    /// - Returns: `true` if the Lambda function exists. Otherwise `false`.
    func doesLambdaFunctionExist(lambdaClient: LambdaClient, name: String) async
 -> Bool {
        do {
            _ = try await lambdaClient.getFunction(
                input: GetFunctionInput(functionName: name)
            )
        } catch {
            return false
        }

        return true
    }
```

- Para obter detalhes da API, consulte GetFunctiona referência da API AWS SDK for Swift.

Usar **`GetFunctionConcurrency`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetFunctionConcurrency`.

CLI

AWS CLI

Para visualizar a configuração de simultaneidade reservada para uma função

O seguinte exemplo de `get-function-concurrency` recupera a configuração de simultaneidade reservada para a função especificada.

```
aws lambda get-function-concurrency \
    --function-name my-function
```

Saída:

```
{
    "ReservedConcurrentExecutions": 250
}
```

- Para obter detalhes da API, consulte GetFunctionConcurrencyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo obtém a simultaneidade reservada para a função do Lambda

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Saída:

```
ReservedConcurrentExecutions
----------------------------
100
```

- Para obter detalhes da API, consulte GetFunctionConcurrencyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo obtém a simultaneidade reservada para a função do Lambda

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Saída:

```
ReservedConcurrentExecutions
----------------------------
100
```

- Para obter detalhes da API, consulte GetFunctionConcurrencyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetFunctionConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetFunctionConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Como criar sua primeira função com tecnologia sem servidor

CLI

AWS CLI

Para recuperar as configurações específicas da versão de uma função do Lambda

O seguinte exemplo de `get-function-configuration` mostra as configurações da versão 2 da função `my-function`.

```
aws lambda get-function-configuration \
    --function-name  my-function:2
```

Saída:

```
{
    "FunctionName": "my-function",
    "LastModified": "2019-09-26T20:28:40.438+0000",
    "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",
    "MemorySize": 256,
```

```
    "Version": "2",
    "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-
uy3l9qyq",
    "Timeout": 3,
    "Runtime": "nodejs10.x",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "5tT2qgzYUHaqwR716pZ2dpkn/0J1FrzJmlKidWoaCgk=",
    "Description": "",
    "VpcConfig": {
        "SubnetIds": [],
        "VpcId": "",
        "SecurityGroupIds": []
    },
    "CodeSize": 304,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:2",
    "Handler": "index.handler"
}
```

Para obter mais informações, consulte Configurar opções da função do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte GetFunctionConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna a configuração específica de versão de uma função do Lambda.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Saída:

```
CodeSha256                  : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                    : 1426
DeadLetterConfig            : Amazon.Lambda.Model.DeadLetterConfig
```

```
Description                : Verson 3 to test Aliases
Environment                : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn                : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
                             :PowershellAlias
FunctionName               : MylambdaFunction123
Handler                    : lambda_function.launch_instance
KMSKeyArn                  :
LastModified               : 2019-12-25T09:52:59.872+0000
LastUpdateStatus           : Successful
LastUpdateStatusReason     :
LastUpdateStatusReasonCode :
Layers                     : {}
MasterArn                  :
MemorySize                 : 128
RevisionId                 : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                       : arn:aws:iam::123456789012:role/service-role/lambda
Runtime                    : python3.8
State                      : Active
StateReason                :
StateReasonCode            :
Timeout                    : 600
TracingConfig              : Amazon.Lambda.Model.TracingConfigResponse
Version                    : 4
VpcConfig                  : Amazon.Lambda.Model.VpcConfigDetail
```

- Para obter detalhes da API, consulte GetFunctionConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a configuração específica de versão de uma função do Lambda.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Saída:

```
CodeSha256                 : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                   : 1426
DeadLetterConfig           : Amazon.Lambda.Model.DeadLetterConfig
Description                : Verson 3 to test Aliases
```

```
Environment                 : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn                 : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
                              :PowershellAlias
FunctionName                : MylambdaFunction123
Handler                     : lambda_function.launch_instance
KMSKeyArn                   :
LastModified                : 2019-12-25T09:52:59.872+0000
LastUpdateStatus            : Successful
LastUpdateStatusReason      :
LastUpdateStatusReasonCode  :
Layers                      : {}
MasterArn                   :
MemorySize                  : 128
RevisionId                  : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                        : arn:aws:iam::123456789012:role/service-role/lambda
Runtime                     : python3.8
State                       : Active
StateReason                 :
StateReasonCode             :
Timeout                     : 600
TracingConfig               : Amazon.Lambda.Model.TracingConfigResponse
Version                     : 4
VpcConfig                   : Amazon.Lambda.Model.VpcConfigDetail
```

- Para obter detalhes da API, consulte GetFunctionConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetPolicy`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetPolicy`.

CLI

AWS CLI

Para recuperar a política do IAM baseada em recursos para uma função, versão ou alias

O seguinte exemplo de `get-policy` mostra informações de política sobre a função `my-function` do Lambda.

```
aws lambda get-policy \
```

```
    --function-name my-function
```

Saída:

```
{
    "Policy": {
        "Version":"2012-10-17",
        "Id":"default",
        "Statement":
        [
            {
                "Sid":"iot-events",
                "Effect":"Allow",
                "Principal": {"Service":"iotevents.amazonaws.com"},
                "Action":"lambda:InvokeFunction",
                "Resource":"arn:aws:lambda:us-west-2:123456789012:function:my-
function"
            }
        ]
    },
    "RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668"
}
```

Para obter mais informações, consulte Usando Resource-based políticas para AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte GetPolicyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo mostra a política de função da função do Lambda

```
Get-LMPolicy -FunctionName test -Select Policy
```

Saída:

```
{"Version":"2012-10-17",       "Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lamb
east-1:123456789102:function:test"}]}
```

- Para obter detalhes da API, consulte GetPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra a política de função da função do Lambda

```
Get-LMPolicy -FunctionName test -Select Policy
```

Saída:

```
{"Version":"2012-10-17",       "Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lamb
east-1:123456789102:function:test"}]}
```

- Para obter detalhes da API, consulte GetPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetProvisionedConcurrencyConfig`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetProvisionedConcurrencyConfig`.

CLI

AWS CLI

Para visualizar uma configuração de simultaneidade provisionada

O seguinte exemplo de `get-provisioned-concurrency-config` mostra detalhes da configuração de simultaneidade provisionada para o alias `BLUE` da função especificada.

```
aws lambda get-provisioned-concurrency-config \
    --function-name my-function \
    --qualifier BLUE
```

Saída:

```
{
    "RequestedProvisionedConcurrentExecutions": 100,
```

```
    "AvailableProvisionedConcurrentExecutions": 100,
    "AllocatedProvisionedConcurrentExecutions": 100,
    "Status": "READY",
    "LastModified": "2019-12-31T20:28:49+0000"
}
```

- Para obter detalhes da API, consulte GetProvisionedConcurrencyConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo obtém a configuração de simultaneidade provisionada para o alias especificado da função do Lambda.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Saída:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- Para obter detalhes da API, consulte GetProvisionedConcurrencyConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo obtém a configuração de simultaneidade provisionada para o alias especificado da função do Lambda.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Saída:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- Para obter detalhes da API, consulte GetProvisionedConcurrencyConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`Invoke`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Invoke`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// invoke.</param
```

```
    /// <param name="parameters">The parameter values that will be passed to the
function.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
        };

        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue =
System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);
```

```
        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload =
 Aws::MakeShared<Aws::StringStream>(
                "FunctionTest");
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
            invokeResult = std::move(outcome.GetResult());
            result = true;
            break;
        }

        else {
            std::cerr << "Error with Lambda::InvokeRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            break;
        }
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Exemplo 1: invocar uma função do Lambda de forma síncrona

O exemplo de `invoke` a seguir invoca a função `my-function` de forma síncrona. A `cli-binary-format` opção é necessária se você estiver usando a AWS CLI versão 2. Para obter mais informações, consulte Opções de linha de comando globais compatíveis da AWS CLI no Guia do usuário da AWS Command Line Interface.

```
aws lambda invoke \
    --function-name my-function \
    --cli-binary-format raw-in-base64-out \
    --payload '{ "name": "Bob" }' \
```

```
    response.json
```

Saída:

```
{
    "ExecutedVersion": "$LATEST",
    "StatusCode": 200
}
```

Consulte mais informações em Invocar uma função do Lambda de modo síncrono no Guia do desenvolvedor do AWS Lambda.

Exemplo 2: invocar uma função do Lambda de forma assíncrona

O exemplo `invoke` a seguir invoca a função `my-function` de forma assíncrona. A `cli-binary-format` opção é necessária se você estiver usando a AWS CLI versão 2. Para obter mais informações, consulte Opções de linha de comando globais compatíveis da AWS CLI no Guia do usuário da AWS Command Line Interface.

```
aws lambda invoke \
    --function-name my-function \
    --invocation-type Event \
    --cli-binary-format raw-in-base64-out \
    --payload '{ "name": "Bob" }' \
    response.json
```

Saída:

```
{
    "StatusCode": 202
}
```

Consulte mais informações em Invocar uma função do Lambda de modo assíncrono no Guia do desenvolvedor do AWS Lambda.

- Consulte detalhes da API em Invocar na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// Invoke invokes the Lambda function specified by functionName, passing the
 parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
 tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
 parameters any, getLog bool) *lambda.InvokeOutput {
 logType := types.LogTypeNone
 if getLog {
  logType = types.LogTypeTail
```

```
 }
 payload, err := json.Marshal(parameters)
 if err != nil {
  log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
 }
 invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
  FunctionName: aws.String(functionName),
  LogType:      logType,
  Payload:      payload,
 })
 if err != nil {
  log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
 }
 return invokeOutput
}
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Invokes a specific AWS Lambda function.
    *
    * @param awsLambda    an instance of {@link LambdaClient} to interact with
the AWS Lambda service
    * @param functionName the name of the AWS Lambda function to be invoked
    */
   public static void invokeFunction(LambdaClient awsLambda, String
functionName) {
       InvokeResponse res;
       try {
           // Need a SdkBytes instance for the payload.
```

```
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const invoke = async (funcName, payload) => {
  const client = new LambdaClient({});
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
```

```
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun invokeFunction(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            logType = LogType.Tail
            payload = byteArray
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
        println("The log result is ${res.logResult}")
    }
}
```

- Consulte detalhes da API em Invoke na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function invoke($functionName, $params, $logType = 'None')
    {
        return $this->lambdaClient->invoke([
            'FunctionName' => $functionName,
            'Payload' => json_encode($params),
            'LogType' => $logType,
        ]);
    }
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def invoke_function(self, function_name, function_params, get_log=False):
        """
```

```
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This
 dict
                                is serialized to JSON before it is sent to
 Lambda.
        :param get_log: When true, the last 4 KB of the execution log are
 included in
                        the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType="Tail" if get_log else "None",
            )
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
```

```
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n
#{e.message}")
  end
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** Invoke the lambda function using calculator InvokeArgs. */
    pub async fn invoke(&self, args: InvokeArgs) -> Result<InvokeOutput,
anyhow::Error> {
        info!(?args, "Invoking {}", self.lambda_name);
        let payload = serde_json::to_string(&args)?;
        debug!(?payload, "Sending payload");
        self.lambda_client
            .invoke()
            .function_name(self.lambda_name.clone())
```

```
            .payload(Blob::new(payload))
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

fn log_invoke_output(invoke: &InvokeOutput, message: &str) {
    if let Some(payload) = invoke.payload().cloned() {
        let payload = String::from_utf8(payload.into_inner());
        info!(?payload, message);
    } else {
        info!("Could not extract payload")
    }
    if let Some(logs) = invoke.log_result() {
        debug!(?logs, "Invoked function logs")
    } else {
        debug!("Invoked function had no logs")
    }
}
```

- Consulte detalhes da API em Invoke na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
          `{`  &&
            `"action": "increment",`  &&
            `"number": 10` &&
          `}` ).
        oo_result = lo_lmd->invoke(                  " oo_result is returned for
 testing purposes. "
                iv_functionname = iv_function_name
```

```
                  iv_payload = lv_json ).
        MESSAGE 'Lambda function invoked.' TYPE 'I'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvrequestcontex.
        MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidzipfileex.
        MESSAGE 'The deployment package could not be unzipped.' TYPE 'E'.
      CATCH /aws1/cx_lmdrequesttoolargeex.
        MESSAGE 'Invoke request body JSON input limit was exceeded by the request
 payload.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.'
 TYPE 'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
 TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
      CATCH /aws1/cx_lmdunsuppedmediatyp00.
        MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Invoke na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
```

```
import AWSLambda
import Foundation

    /// Invoke the Lambda function to increment a value.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - number: The number to increment.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: An integer number containing the incremented value.
    func invokeIncrement(lambdaClient: LambdaClient, number: Int) async throws ->
 Int {
        do {
            let incRequest = IncrementRequest(action: "increment", number:
 number)
            let incData = try! JSONEncoder().encode(incRequest)

            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: incData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
 from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }
    }
```

- Para obter detalhes sobre a API, consulte Invoke na Referência da API AWS SDK para Swift.

Use **`ListFunctions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFunctions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Como criar um painel de monitoramento com o nome da função como uma variável

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get a list of Lambda functions.
/// </summary>
/// <returns>A list of FunctionConfiguration objects.</returns>
public async Task<List<FunctionConfiguration>> ListFunctionsAsync()
{
    var functionList = new List<FunctionConfiguration>();

    var functionPaginator =
        _lambdaService.Paginators.ListFunctions(new ListFunctionsRequest());
    await foreach (var function in functionPaginator.Functions)
    {
        functionList.Add(function);
    }

    return functionList;
}
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    std::vector<Aws::String> functions;
    Aws::String marker;

    do {
        Aws::Lambda::Model::ListFunctionsRequest request;
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
            std::cout << result.GetFunctions().size()
                      << " lambda functions were retrieved." << std::endl;
```

```
            for (const Aws::Lambda::Model::FunctionConfiguration
 &functionConfiguration: result.GetFunctions()) {
                functions.push_back(functionConfiguration.GetFunctionName());
                std::cout << functions.size() << "  "
                          << functionConfiguration.GetDescription() << std::endl;
                std::cout << "   "
                          <<
 Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                  functionConfiguration.GetRuntime()) << ": "
                          << functionConfiguration.GetHandler()
                          << std::endl;
            }
            marker = result.GetNextMarker();
        }
        else {
            std::cerr << "Error with Lambda::ListFunctions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Recuperar uma lista de funções do Lambda

O exemplo de `list-functions` a seguir exibe uma lista de todas as funções do usuário atual.

```
aws lambda list-functions
```

Saída:

```
{
    "Functions": [
        {
            "TracingConfig": {
                "Mode": "PassThrough"
```

```
            },
            "Version": "$LATEST",
            "CodeSha256": "dBG9m8SGdmlEjw/JYXlhhvCrAv5TxvXsbL/RMr0fT/I=",
            "FunctionName": "helloworld",
            "MemorySize": 128,
            "RevisionId": "1718e831-badf-4253-9518-d0644210af7b",
            "CodeSize": 294,
            "FunctionArn": "arn:aws:lambda:us-
west-2:123456789012:function:helloworld",
            "Handler": "helloworld.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-
role-zgur6bf4",
            "Timeout": 3,
            "LastModified": "2025-09-23T18:32:33.857+0000",
            "Runtime": "nodejs22.x",
            "Description": ""
        },
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "sU0cJ2/hOZevwV/lTxCuQqK3gDZP3i8gUoqUUVRmY6E=",
            "FunctionName": "my-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 256,
            "RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668",
            "CodeSize": 266,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function",
            "Handler": "index.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/
helloWorldPython-role-uy3l9qyq",
            "Timeout": 3,
            "LastModified": "2025-10-01T16:47:28.490+0000",
            "Runtime": "nodejs22.x",
            "Description": ""
        },
        {
            "Layers": [
```

```
                {
                    "CodeSize": 41784542,
                    "Arn": "arn:aws:lambda:us-
west-2:420165488524:layer:AWSLambda-Python37-SciPy1x:2"
                },
                {
                    "CodeSize": 4121,
                    "Arn": "arn:aws:lambda:us-
west-2:123456789012:layer:pythonLayer:1"
                }
            ],
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "ZQukCqxtkqFgyF2cU41Avj99TKQ/hNihPtDtRcc08mI=",
            "FunctionName": "my-python-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 128,
            "RevisionId": "80b4eabc-acf7-4ea8-919a-e874c213707d",
            "CodeSize": 299,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
python-function",
            "Handler": "lambda_function.lambda_handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/my-python-
function-role-z5g7dr6n",
            "Timeout": 3,
            "LastModified": "2025-10-01T19:40:41.643+0000",
            "Runtime": "python3.11",
            "Description": ""
        }
    ]
}
```

Para obter mais informações, consulte Configurar memória para uma função do Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte ListFunctionsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// ListFunctions lists up to maxItems functions for the account. This function
 uses a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
 []types.FunctionConfiguration {
 var functions []types.FunctionConfiguration
 paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
 &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
```

```
 for paginator.HasMorePages() && len(functions) < maxItems {
  pageOutput, err := paginator.NextPage(ctx)
  if err != nil {
   log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
  }
  functions = append(functions, pageOutput.Functions...)
 }
 return functions
}
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const listFunctions = () => {
  const client = new LambdaClient({});
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function listFunctions($maxItems = 50, $marker = null)
    {
        if (is_null($marker)) {
            return $this->lambdaClient->listFunctions([
                'MaxItems' => $maxItems,
            ]);
        }

        return $this->lambdaClient->listFunctions([
            'Marker' => $marker,
            'MaxItems' => $maxItems,
        ]);
    }
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo mostra todas as funções do Lambda com tamanho de código classificado

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Saída:

```
FunctionName                                                    Runtime   Timeout
 CodeSize
------------                                                    -------   -------
 --------
test                                                            python2.7       3
   243
MylambdaFunction123                                             python3.8     600
   659
myfuncpython1                                                   python3.8     303
   675
```

- Para obter detalhes da API, consulte ListFunctionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra todas as funções do Lambda com tamanho de código classificado

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Saída:

```
FunctionName                                                    Runtime   Timeout
 CodeSize
------------                                                    -------   -------
 --------
test                                                            python2.7       3
   243
MylambdaFunction123                                             python3.8     600
   659
myfuncpython1                                                   python3.8     303
   675
```

- Para obter detalhes da API, consulte ListFunctionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def list_functions(self):
        """
        Lists the Lambda functions for the current account.
        """
        try:
            func_paginator = self.lambda_client.get_paginator("list_functions")
            for func_page in func_paginator.paginate():
                for func in func_page["Functions"]:
                    print(func["FunctionName"])
                    desc = func.get("Description")
                    if desc:
                        print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
        except ClientError as err:
            logger.error(
                "Couldn't list functions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListFunctionsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response['functions'].each do |function|
        functions.append(function['function_name'])
      end
    end
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error listing functions:\n #{e.message}")
  end
```

- Para obter detalhes da API, consulte ListFunctionsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /** List all Lambda functions in the current Region. */
   pub async fn list_functions(&self) -> Result<ListFunctionsOutput,
anyhow::Error> {
       info!("Listing lambda functions");
       self.lambda_client
           .list_functions()
           .send()
           .await
           .map_err(anyhow::Error::from)
   }
```

- Para obter detalhes da API, consulte a ListFunctionsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_lmd->listfunctions( ).       " oo_result is returned for
testing purposes. "
       DATA(lt_functions) = oo_result->get_functions( ).
       MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
```

```
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
 TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListFunctionsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Returns an array containing the names of all AWS Lambda functions
    /// available to the user.
    ///
    /// - Parameter lambdaClient: The `IAMClient` to use.
    ///
    /// - Throws: `ExampleError.listFunctionsError`
    ///
    /// - Returns: An array of lambda function name strings.
    func getFunctionNames(lambdaClient: LambdaClient) async throws -> [String] {
        let pages = lambdaClient.listFunctionsPaginated(
            input: ListFunctionsInput()
        )

        var functionNames: [String] = []
```

```
        for try await page in pages {
            guard let functions = page.functions else {
                throw ExampleError.listFunctionsError
            }

            for function in functions {
                functionNames.append(function.functionName ?? "<unknown>")
            }
        }

        return functionNames
    }
```

- Para obter detalhes da API, consulte ListFunctionsa referência da API AWS SDK for Swift.

Usar **`ListProvisionedConcurrencyConfigs`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListProvisionedConcurrencyConfigs`.

CLI

AWS CLI

Para obter uma lista de configurações de simultaneidade provisionada

O seguinte exemplo de `list-provisioned-concurrency-configs` lista as configurações de simultaneidade provisionada para a função especificada.

```
aws lambda list-provisioned-concurrency-configs \
    --function-name my-function
```

Saída:

```
{
    "ProvisionedConcurrencyConfigs": [
        {
            "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-
function:GREEN",
            "RequestedProvisionedConcurrentExecutions": 100,
            "AvailableProvisionedConcurrentExecutions": 100,
```

```
            "AllocatedProvisionedConcurrentExecutions": 100,
            "Status": "READY",
            "LastModified": "2019-12-31T20:29:00+0000"
        },
        {
            "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-
function:BLUE",
            "RequestedProvisionedConcurrentExecutions": 100,
            "AvailableProvisionedConcurrentExecutions": 100,
            "AllocatedProvisionedConcurrentExecutions": 100,
            "Status": "READY",
            "LastModified": "2019-12-31T20:28:49+0000"
        }
    ]
}
```

- Para obter detalhes da API, consulte ListProvisionedConcurrencyConfigsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera a lista de configurações de simultaneidade provisionada para uma função do Lambda.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- Para obter detalhes da API, consulte ListProvisionedConcurrencyConfigsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a lista de configurações de simultaneidade provisionada para uma função do Lambda.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- Para obter detalhes da API, consulte ListProvisionedConcurrencyConfigsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListTags`.

CLI

AWS CLI

Para recuperar a lista de tags para uma função do Lambda

O seguinte exemplo de `list-tags` mostra as tags anexadas à função `my-function` do Lambda.

```
aws lambda list-tags \
    --resource arn:aws:lambda:us-west-2:123456789012:function:my-function
```

Saída:

```
{
    "Tags": {
        "Category": "Web Tools",
        "Department": "Sales"
    }
}
```

Para obter mais informações, consulte Como marcar funções do Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte ListTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: recupera as tags e seus valores atualmente definidos na função especificada.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Saída:

```
Key         Value
```

```
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- Para obter detalhes da API, consulte ListTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: recupera as tags e seus valores atualmente definidos na função especificada.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Saída:

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- Para obter detalhes da API, consulte ListTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListVersionsByFunction`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListVersionsByFunction`.

CLI

AWS CLI

Para recuperar uma lista de versões de uma função

O seguinte exemplo de `list-versions-by-function` mostra a lista de versões da função `my-function` do Lambda.

```
aws lambda list-versions-by-function \
    --function-name my-function
```

Saída:

```
{
    "Versions": [
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "sU0cJ2/hOZevwV/lTxCuQqK3gDZP3i8gUoqUUVRmY6E=",
            "FunctionName": "my-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 256,
            "RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668",
            "CodeSize": 266,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:$LATEST",
            "Handler": "index.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/
helloWorldPython-role-uy3l9qyq",
            "Timeout": 3,
            "LastModified": "2019-10-01T16:47:28.490+0000",
            "Runtime": "nodejs10.x",
            "Description": ""
        },
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "1",
            "CodeSha256": "5tT2qgzYUHoqwR616pZ2dpkn/0J1FrzJmlKidWaaCgk=",
            "FunctionName": "my-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 256,
            "RevisionId": "949c8914-012e-4795-998c-e467121951b1",
            "CodeSize": 304,
```

```
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:1",
            "Handler": "index.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/
helloWorldPython-role-uy3l9qyq",
            "Timeout": 3,
            "LastModified": "2019-09-26T20:28:40.438+0000",
            "Runtime": "nodejs10.x",
            "Description": "new version"
        },
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "2",
            "CodeSha256": "sU0cJ2/hOZevwV/lTxCuQqK3gDZP3i8gUoqUUVRmY6E=",
            "FunctionName": "my-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 256,
            "RevisionId": "cd669f21-0f3d-4e1c-9566-948837f2e2ea",
            "CodeSize": 266,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:2",
            "Handler": "index.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/
helloWorldPython-role-uy3l9qyq",
            "Timeout": 3,
            "LastModified": "2019-10-01T16:47:28.490+0000",
            "Runtime": "nodejs10.x",
            "Description": "newer version"
        }
    ]
}
```

Para obter mais informações, consulte Configurando aliases de funções do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte ListVersionsByFunctionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna a lista de configurações específicas de versão para cada versão da função do Lambda.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Saída:

```
FunctionName        Runtime   MemorySize Timeout CodeSize LastModified
      RoleName
------------        -------   ---------- ------- -------- ------------
      --------
MylambdaFunction123 python3.8        128     600      659
 2020-01-10T03:20:56.390+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:19:02.238+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:39:36.779+0000 lambda
MylambdaFunction123 python3.8        128     600     1426
 2019-12-25T09:52:59.872+0000 lambda
```

- Para obter detalhes da API, consulte ListVersionsByFunctionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a lista de configurações específicas de versão para cada versão da função do Lambda.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Saída:

```
FunctionName        Runtime   MemorySize Timeout CodeSize LastModified
      RoleName
------------        -------   ---------- ------- -------- ------------
      --------
MylambdaFunction123 python3.8        128     600      659
 2020-01-10T03:20:56.390+0000 lambda
```

```
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:19:02.238+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:39:36.779+0000 lambda
MylambdaFunction123 python3.8        128     600     1426
 2019-12-25T09:52:59.872+0000 lambda
```

- Para obter detalhes da API, consulte ListVersionsByFunctionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PublishVersion`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PublishVersion`.

CLI

AWS CLI

Para publicar uma nova versão de uma função

O seguinte exemplo de `publish-version` publica uma nova versão da função `my-function` do Lambda.

```
aws lambda publish-version \
    --function-name my-function
```

Saída:

```
{
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "dBG9m8SGdmlEjw/JYXlhhvCrAv5TxvXsbL/RMr0fT/I=",
    "FunctionName": "my-function",
    "CodeSize": 294,
    "RevisionId": "f31d3d39-cc63-4520-97d4-43cd44c94c20",
    "MemorySize": 128,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:3",
    "Version": "2",
    "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-
zgur6bf4",
```

```
    "Timeout": 3,
    "LastModified": "2019-09-23T18:32:33.857+0000",
    "Handler": "my-function.handler",
    "Runtime": "nodejs10.x",
    "Description": ""
}
```

Para obter mais informações, consulte Configurando aliases de funções do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte PublishVersionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria uma versão para o snapshot existente do código da função do Lambda

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- Para obter detalhes da API, consulte PublishVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma versão para o snapshot existente do código da função do Lambda

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- Para obter detalhes da API, consulte PublishVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutFunctionConcurrency`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutFunctionConcurrency`.

CLI

AWS CLI

Para configurar um limite de simultaneidade reservado para uma função

O seguinte exemplo de `put-function-concurrency` configura 100 execuções simultâneas reservadas para a função `my-function`.

```
aws lambda put-function-concurrency \
    --function-name  my-function  \
    --reserved-concurrent-executions 100
```

Saída:

```
{
    "ReservedConcurrentExecutions": 100
}
```

Para obter mais informações, consulte Como reservar simultaneidade para uma função do Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte PutFunctionConcurrencyem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo aplica as configurações de simultaneidade para a função de maneira geral.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- Para obter detalhes da API, consulte PutFunctionConcurrencyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo aplica as configurações de simultaneidade para a função de maneira geral.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- Para obter detalhes da API, consulte PutFunctionConcurrencyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutProvisionedConcurrencyConfig`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutProvisionedConcurrencyConfig`.

CLI

AWS CLI

Para alocar a simultaneidade provisionada

O seguinte exemplo de `put-provisioned-concurrency-config` aloca 100 simultaneidades provisionadas para o alias `BLUE` da função especificada.

```
aws lambda put-provisioned-concurrency-config \
    --function-name my-function \
    --qualifier BLUE \
    --provisioned-concurrent-executions 100
```

Saída:

```
{
    "Requested ProvisionedConcurrentExecutions": 100,
    "Allocated ProvisionedConcurrentExecutions": 0,
    "Status": "IN_PROGRESS",
    "LastModified": "2019-11-21T19:32:12+0000"
}
```

- Para obter detalhes da API, consulte PutProvisionedConcurrencyConfigem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo adiciona uma configuração de simultaneidade provisionada ao alias de uma função

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Para obter detalhes da API, consulte PutProvisionedConcurrencyConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo adiciona uma configuração de simultaneidade provisionada ao alias de uma função

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Para obter detalhes da API, consulte PutProvisionedConcurrencyConfigem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RemovePermission`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemovePermission`.

CLI

AWS CLI

Para remover permissões de uma função do Lambda existente

O seguinte exemplo de `remove-permission` remove permissão para invocar uma função chamada `my-function`.

```
aws lambda remove-permission \
    --function-name my-function \
    --statement-id sns
```

Este comando não produz saída.

Para obter mais informações, consulte Usando Resource-based políticas para AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte RemovePermissionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo remove a política de função especificada StatementId de uma Função Lambda.

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId
 $policy[0].Sid
```

- Para obter detalhes da API, consulte RemovePermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove a política de função especificada StatementId de uma Função Lambda.

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId
 $policy[0].Sid
```

- Para obter detalhes da API, consulte RemovePermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`TagResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `TagResource`.

CLI

AWS CLI

Para adicionar tags a uma função do Lambda existente

O seguinte exemplo de `tag-resource` adiciona uma tag com o nome de chave `DEPARTMENT` e um valor de `Department A` à função do Lambda especificada.

```
aws lambda tag-resource \
    --resource arn:aws:lambda:us-west-2:123456789012:function:my-function \
    --tags "DEPARTMENT=Department A"
```

Este comando não produz saída.

Para obter mais informações, consulte Como marcar funções do Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte TagResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: adiciona as três tags (Washington, Oregon e Califórnia) e seus valores associados à função especificada identificada por seu ARN.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia";
 "Oregon" = "Salem"; "California" = "Sacramento" }
```

- Para obter detalhes da API, consulte TagResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: adiciona as três tags (Washington, Oregon e Califórnia) e seus valores associados à função especificada identificada por seu ARN.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia";
 "Oregon" = "Salem"; "California" = "Sacramento" }
```

- Para obter detalhes da API, consulte TagResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UntagResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UntagResource`.

CLI

AWS CLI

Para remover tags de uma função do Lambda existente

O seguinte exemplo de `untag-resource` remove a tag com o nome de chave `DEPARTMENT` da função `my-function` do Lambda.

```
aws lambda untag-resource \
    --resource arn:aws:lambda:us-west-2:123456789012:function:my-function \
    --tag-keys DEPARTMENT
```

Este comando não produz saída.

Para obter mais informações, consulte Como marcar funções do Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte UntagResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: remove as tags fornecidas de uma função. A menos que a opção -Force esteja especificada, o cmdlet solicitará a confirmação antes de continuar. Uma única chamada será feita para o serviço a fim de remover as tags.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey
 "Washington","Oregon","California"
```

Exemplo 2: remove as tags fornecidas de uma função. A menos que a opção -Force esteja especificada, o cmdlet solicitará a confirmação antes de continuar. Isso acontece porque a chamada para o serviço é feita pela tag fornecida.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Para obter detalhes da API, consulte UntagResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: remove as tags fornecidas de uma função. A menos que a opção -Force esteja especificada, o cmdlet solicitará a confirmação antes de continuar. Uma única chamada será feita para o serviço a fim de remover as tags.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey
 "Washington","Oregon","California"
```

Exemplo 2: remove as tags fornecidas de uma função. A menos que a opção -Force esteja especificada, o cmdlet solicitará a confirmação antes de continuar. Isso acontece porque a chamada para o serviço é feita pela tag fornecida.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Para obter detalhes da API, consulte UntagResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateAlias`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateAlias`.

CLI

AWS CLI

Para atualizar um alias de função

O seguinte exemplo de `update-alias` atualiza um alias chamado `LIVE` que aponta para a versão 3 da função `my-function` do Lambda.

```
aws lambda update-alias \
    --function-name my-function \
    --function-version 3 \
    --name LIVE
```

Saída:

```
{
    "FunctionVersion": "3",
    "Name": "LIVE",
    "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:LIVE",
    "RevisionId": "594f41fb-b85f-4c20-95c7-6ca5f2a92c93",
    "Description": "alias for live version of function"
}
```

Para obter mais informações, consulte Configurando aliases de funções do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte UpdateAliasem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza a configuração de um alias de função do Lambda existente. Ele atualiza o RoutingConfiguration valor para transferir 60% (0,6) do tráfego para a versão 1

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description
 " Alias for version 2" -FunctionVersion 2 -Name "newlabel1" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6}
```

- Para obter detalhes da API, consulte UpdateAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a configuração de um alias de função do Lambda existente. Ele atualiza o RoutingConfiguration valor para transferir 60% (0,6) do tráfego para a versão 1

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description
 " Alias for version 2" -FunctionVersion 2 -Name "newlabel1" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6}
```

- Para obter detalhes da API, consulte UpdateAliasem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`UpdateFunctionCode`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateFunctionCode`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update an existing Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to update.</
param>
    /// <param name="bucketName">The bucket where the zip file containing
    /// the Lambda function code is stored.</param>
    /// <param name="key">The key name of the source code file.</param>
    /// <returns>Async Task.</returns>
    public async Task UpdateFunctionCodeAsync(
        string functionName,
        string bucketName,
        string key)
    {
```

```
        var functionCodeRequest = new UpdateFunctionCodeRequest
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await
_lambdaService.UpdateFunctionCodeAsync(functionCodeRequest);
        Console.WriteLine($"The Function was last modified at
{response.LastModified}.");
    }
```

- Para obter detalhes da API, consulte UpdateFunctionCodea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::UpdateFunctionCodeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
```

```
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
                Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                       buffer.str().length()));
        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
 client.UpdateFunctionCode(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." <<
 std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte UpdateFunctionCodea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Atualizar o código de uma função do Lambda

O exemplo de `update-function-code` a seguir substitui o código da versão não publicada ($LATEST) da função `my-function` pelo conteúdo do arquivo zip especificado.

```
aws lambda update-function-code \
    --function-name  my-function \
    --zip-file fileb://my-function.zip
```

Saída:

```
{
    "FunctionName": "my-function",
    "LastModified": "2019-09-26T20:28:40.438+0000",
    "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",
    "MemorySize": 256,
    "Version": "$LATEST",
    "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-
uy3l9qyq",
    "Timeout": 3,
    "Runtime": "nodejs10.x",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "5tT2qgzYUHaqwR716pZ2dpkn/0J1FrzJmlKidWoaCgk=",
    "Description": "",
    "VpcConfig": {
        "SubnetIds": [],
        "VpcId": "",
        "SecurityGroupIds": []
    },
    "CodeSize": 304,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
    "Handler": "index.handler"
}
```

Para obter mais informações, consulte Configurar opções da função do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte UpdateFunctionCodeem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// UpdateFunctionCode updates the code for the Lambda function specified by
 functionName.
// The existing code for the Lambda function is entirely replaced by the code in
 the
// zipPackage buffer. After the update action is called, a
 lambda.FunctionUpdatedV2Waiter
```

```
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context,
 functionName string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
 &lambda.UpdateFunctionCodeInput{
  FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
 })
 if err != nil {
  log.Panicf("Couldn't update code for function %v. Here's why: %v\n",
 functionName, err)
 } else {
  waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}
```

- Para obter detalhes da API, consulte UpdateFunctionCodea Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Updates the code for an AWS Lambda function.
    *
    * @param awsLambda  the AWS Lambda client
    * @param functionName the name of the Lambda function to update
    * @param bucketName the name of the S3 bucket where the function code is
located
    * @param key the key (file name) of the function code in the S3 bucket
    * @throws LambdaException if there is an error updating the function code
    */
   public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
       try {
           LambdaWaiter waiter = awsLambda.waiter();
           UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
               .functionName(functionName)
               .publish(true)
               .s3Bucket(bucketName)
               .s3Key(key)
               .build();

           UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
           GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .build();

           WaiterResponse<GetFunctionConfigurationResponse> waiterResponse =
waiter
               .waitUntilFunctionUpdated(getFunctionConfigRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("The last modified value is " +
response.lastModified());

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte UpdateFunctionCodea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${newFunc}.zip`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte UpdateFunctionCodea Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
{
    return $this->lambdaClient->updateFunctionCode([
        'FunctionName' => $functionName,
        'S3Bucket' => $s3Bucket,
        'S3Key' => $s3Key,
    ]);
}
```

- Para obter detalhes da API, consulte UpdateFunctionCodea Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: atualiza a função chamada 'MyFunction' com o novo conteúdo contido no arquivo zip especificado. Para uma função do Lambda em C# .NET Core, o arquivo zip deve conter a montagem compilada.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Exemplo 2: este exemplo é semelhante ao anterior, mas usa um objeto do Amazon S3 contendo o código atualizado para atualizar a função.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -
Key UpdatedCode.zip
```

- Para obter detalhes da API, consulte UpdateFunctionCodeem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: atualiza a função chamada 'MyFunction' com o novo conteúdo contido no arquivo zip especificado. Para uma função do Lambda em C# .NET Core, o arquivo zip deve conter a montagem compilada.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Exemplo 2: este exemplo é semelhante ao anterior, mas usa um objeto do Amazon S3 contendo o código atualizado para atualizar a função.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -
Key UpdatedCode.zip
```

- Para obter detalhes da API, consulte UpdateFunctionCodeem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
 contains
        the code for the function.
```

```
        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes
 in
                                .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a UpdateFunctionCodeReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
```

```
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for:
#{function_name}:\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n
#{e.message}")
  end
```

- Para obter detalhes da API, consulte UpdateFunctionCodea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** Given a Path to a zip file, update the function's code and wait for the
 update to finish. */
    pub async fn update_function_code(
        &self,
        zip_file: PathBuf,
        key: String,
    ) -> Result<UpdateFunctionCodeOutput, anyhow::Error> {
        let function_code = self.prepare_function(zip_file, Some(key)).await?;

        info!("Updating code for {}", self.lambda_name);
        let update = self
            .lambda_client
            .update_function_code()
            .function_name(self.lambda_name.clone())
            .s3_bucket(self.bucket.clone())
            .s3_key(function_code.s3_key().unwrap().to_string())
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(update)
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --
output-format Zip`.
     */
    async fn prepare_function(
```

```
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }
```

- Para obter detalhes da API, consulte a UpdateFunctionCodereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_lmd->updatefunctioncode(     " oo_result is returned for
testing purposes. "
```

```
              iv_functionname = iv_function_name
              iv_zipfile = io_zip_file ).

       MESSAGE 'Lambda function code updated.' TYPE 'I'.
     CATCH /aws1/cx_lmdcodesigningcfgno00.
       MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodestorageexcdex.
       MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodeverification00.
       MESSAGE 'Code signature failed one or more validation checks for
signature mismatch or expiration.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvalidcodesigex.
       MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateFunctionCodereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        let zipUrl = URL(fileURLWithPath: path)
        let zipData: Data

        // Read the function's Zip file.

        do {
            zipData = try Data(contentsOf: zipUrl)
        } catch {
            throw ExampleError.zipFileReadError
        }

        // Update the function's code and wait for the updated version to be
        // ready for use.

        do {
            _ = try await lambdaClient.updateFunctionCode(
                input: UpdateFunctionCodeInput(
                    functionName: functionName,
                    zipFile: zipData
                )
            )
        } catch {
            return false
        }
```

- Para obter detalhes da API, consulte UpdateFunctionCodea referência da API AWS SDK for Swift.

Use **`UpdateFunctionConfiguration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateFunctionConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update the code of a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function to update.</param>
    /// <param name="functionHandler">The code that performs the function's
actions.</param>
    /// <param name="environmentVariables">A dictionary of environment
variables.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateFunctionConfigurationAsync(
        string functionName,
        string functionHandler,
        Dictionary<string, string> environmentVariables)
    {
        var request = new UpdateFunctionConfigurationRequest
        {
            Handler = functionHandler,
            FunctionName = functionName,
            Environment = new Amazon.Lambda.Model.Environment { Variables =
environmentVariables },
        };

        var response = await
_lambdaService.UpdateFunctionConfigurationAsync(request);

        Console.WriteLine(response.LastModified);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Lambda::LambdaClient client(clientConfig);

       Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
       request.SetFunctionName(LAMBDA_NAME);
       Aws::Lambda::Model::Environment environment;
       environment.AddVariables("LOG_LEVEL", "DEBUG");
       request.SetEnvironment(environment);

       Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
               request);

       if (outcome.IsSuccess()) {
           std::cout << "The lambda configuration was successfully updated."
                     << std::endl;
           break;
       }

       else {
           std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
       }
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Modificar a configuração de uma função

O exemplo de `update-function-configuration` a seguir modifica o tamanho da memória para 256 MB para a versão não publicada ($LATEST) da função `my-function`.

```
aws lambda update-function-configuration \
    --function-name  my-function \
    --memory-size 256
```

Saída:

```
{
    "FunctionName": "my-function",
    "LastModified": "2019-09-26T20:28:40.438+0000",
    "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",
    "MemorySize": 256,
    "Version": "$LATEST",
    "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-
uy3l9qyq",
    "Timeout": 3,
    "Runtime": "nodejs10.x",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "5tT2qgzYUHaqwR716pZ2dpkn/0J1FrzJmlKidWoaCgk=",
    "Description": "",
    "VpcConfig": {
        "SubnetIds": [],
        "VpcId": "",
        "SecurityGroupIds": []
    },
    "CodeSize": 304,
```

```
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
    "Handler": "index.handler"
}
```

Para obter mais informações, consulte Configurar opções da função do AWS Lambda no Guia do desenvolvedor do AWS Lambda.

- Para obter detalhes da API, consulte UpdateFunctionConfigurationem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}
```

```
// UpdateFunctionConfiguration updates a map of environment variables configured
 for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
 functionName string, envVars map[string]string) {
 _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
 &lambda.UpdateFunctionConfigurationInput{
  FunctionName: aws.String(functionName),
  Environment:  &types.Environment{Variables: envVars},
 })
 if err != nil {
  log.Panicf("Couldn't update configuration for %v. Here's why: %v",
 functionName, err)
 }
}
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Updates the configuration of an AWS Lambda function.
     *
     * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
 Lambda operation
     * @param functionName  the name of the AWS Lambda function to update
     * @param handler       the new handler for the AWS Lambda function
     *
     * @throws LambdaException if there is an error while updating the function
 configuration
```

```
     */
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
 functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
 UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const updateFunctionConfiguration = async (funcName) => {
  const client = new LambdaClient({});
  const config = readFileSync(`${dirname}../functions/config.json`).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  const result = await client.send(command);
```

```
  await waitForFunctionUpdated({ FunctionName: funcName });
  return result;
};
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public function updateFunctionConfiguration($functionName, $handler,
 $environment = '')
   {
       return $this->lambdaClient->updateFunctionConfiguration([
           'FunctionName' => $functionName,
           'Handler' => "$handler.lambda_handler",
           'Environment' => $environment,
       ]);
   }
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo atualiza a configuração da função do Lambda existente

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler
 "lambda_function.launch_instance" -Timeout 600 -Environment_Variable
```

```
 @{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:
 123456789101:MyfirstTopic
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a configuração da função do Lambda existente

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler
 "lambda_function.launch_instance" -Timeout 600 -Environment_Variable
 @{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:
 123456789101:MyfirstTopic
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
```

```
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={"Variables": env_vars}
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a UpdateFunctionConfigurationReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
```

```
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                    function_name: function_name,
                                                    environment: {
                                                      variables: {
                                                        'LOG_LEVEL' => log_level
                                                      }
                                                    }
                                                  })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for
#{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
  end
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** Update the environment for a function. */
    pub async fn update_function_configuration(
        &self,
        environment: Environment,
    ) -> Result<UpdateFunctionConfigurationOutput, anyhow::Error> {
        info!(
            ?environment,
            "Updating environment for {}", self.lambda_name
        );
        let updated = self
            .lambda_client
            .update_function_configuration()
            .function_name(self.lambda_name.clone())
            .environment(environment)
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(updated)
    }
```

- Para obter detalhes da API, consulte a UpdateFunctionConfigurationreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_lmd->updatefunctionconfiguration(     " oo_result is
returned for testing purposes. "
             iv_functionname = iv_function_name
```

```
              iv_runtime = iv_runtime
              iv_description  = 'Updated Lambda function'
              iv_memorysize  = iv_memory_size ).

        MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
      CATCH /aws1/cx_lmdcodesigningcfgno00.
        MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdcodeverification00.
        MESSAGE 'Code signature failed one or more validation checks for
signature mismatch or expiration.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidcodesigex.
        MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateFunctionConfigurationreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
```

```
import AWSLambda
import Foundation

    /// Tell the server-side component to log debug output by setting its
    /// environment's `LOG_LEVEL` to `DEBUG`.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to enable debug
    ///     logging for.
    ///
    /// - Throws: `ExampleError.environmentResponseMissingError`,
    ///   `ExampleError.updateFunctionConfigurationError`,
    ///   `ExampleError.environmentVariablesMissingError`,
    ///   `ExampleError.logLevelIncorrectError`,
    ///   `ExampleError.updateFunctionConfigurationError`
    func enableDebugLogging(lambdaClient: LambdaClient, functionName: String)
 async throws {
        let envVariables = [
            "LOG_LEVEL": "DEBUG"
        ]
        let environment = LambdaClientTypes.Environment(variables: envVariables)

        do {
            let output = try await lambdaClient.updateFunctionConfiguration(
                input: UpdateFunctionConfigurationInput(
                    environment: environment,
                    functionName: functionName
                )
            )

            guard let response = output.environment else {
                throw ExampleError.environmentResponseMissingError
            }

            if response.error != nil {
                throw ExampleError.updateFunctionConfigurationError
            }

            guard let retVariables = response.variables else {
                throw ExampleError.environmentVariablesMissingError
            }

            for envVar in retVariables {
```

```
                if envVar.key == "LOG_LEVEL" && envVar.value != "DEBUG" {
                    print("*** Log level is not set to DEBUG!")
                    throw ExampleError.logLevelIncorrectError
                }
            }
        } catch {
            throw ExampleError.updateFunctionConfigurationError
        }
    }
```

- Para obter detalhes da API, consulte UpdateFunctionConfigurationa referência da API AWS SDK for Swift.

Cenários para o uso do Lambda AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Lambda com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Lambda ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Confirme automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK
- Migre automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK
- Crie uma API REST do API Gateway para rastrear COVID-19 dados
- Criar uma API REST de biblioteca de empréstimos
- Criar uma aplicação de mensageiro com o Step Functions
- Criar uma API REST com integração de proxy de função
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Criar uma aplicação de chat websocket com o API Gateway
- Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio

- Como criar um painel de monitoramento com o nome da função como uma variável
- Como criar sua primeira função com tecnologia sem servidor
- Invocar uma função do Lambda em um navegador
- Como transformar dados para sua aplicação com o S3 Object Lambda
- Usar o API Gateway para invocar uma função do Lambda
- Usar Step Functions para invocar funções do Lambda
- Usar eventos programados para chamar uma função do Lambda
- Use a API do Amazon Neptune para desenvolver uma função do Lambda que consulta dados de gráficos
- Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor
- Grave dados de atividades personalizados com uma função Lambda após a autenticação do usuário do Amazon Cognito usando um AWS SDK

Confirme automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK

Os exemplos de código a seguir mostram como confirmar automaticamente usuários conhecidos do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PreSignUp`.
- Inscreva-se para ser um usuário no Amazon Cognito.
- A função do Lambda verifica uma tabela do DynamoDB e confirma automaticamente os usuários conhecidos.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
```

```
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]
```

```
 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string,
 userName string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}
```

```
// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `PreSignUp` com uma função do Lambda.

```
import (
```

```
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
```

```
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
```

```
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}
```

```
// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
```

```
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
```

```
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
```

```
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
```

```
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
```

```
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
```

```
    log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
   }
  }
  return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
```

```
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
```

```
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
```

```
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
```

```
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
```

```
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
```

```
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
```

```
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
  HandlerArn: nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure uma execução interativa de “Cenário”. Os exemplos JavaScript (v3) compartilham um executor de cenários para simplificar exemplos complexos. O código-fonte completo está ativado GitHub.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      UserEmail: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
```

```
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

Esse cenário demonstra a confirmação automática de um usuário conhecido. Ele orquestra as etapas do exemplo.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
```

```
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
```

```
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
```

```
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
```

```
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
    const signUp = (password) =>
      signUpUser({
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
```

```
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
    if (logStreamErr) {
      state.errors.push(logStreamErr);
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);
```

```
const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with:
 ${state.token.slice(0, 11)}`,
  { skipWhen: skipWhenErrors },
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
```

```
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
```

```
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

Essas são etapas compartilhadas com outros cenários.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
```

```
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
  { skipWhen: skipWhenErrors },
);
```

Um manipulador do gatilho `PreSignUp` com uma função do Lambda.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
```

```
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user
 '${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
 '${storedUserInfo.UserName}' does not match supplied UserName
 '${event.userName}'. Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
```

```
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

Módulo de ações de CloudWatch registros.

```
import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );
```

```
    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[]
 | null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Módulo de ações do Amazon Cognito.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
```

```
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password:
 string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
```

```
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
```

```
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

Módulo de ações do DynamoDB.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string,
 unknown>[] }} config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
```

```
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Migre automaticamente usuários conhecidos do Amazon Cognito com uma função Lambda usando um AWS SDK

O exemplo de código a seguir mostra como migrar automaticamente usuários conhecidas do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `MigrateUser`.
- Faça login no Amazon Cognito com um nome de usuário e e-mail que não estejam no grupo de usuários.
- A função do Lambda verifica uma tabela do DynamoDB e migra automaticamente os usuários conhecidos para o grupo de usuários.
- Realize um fluxo de senha esquecida para redefinir a senha para o usuário migrado.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
```

```
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
```

```
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string,
 user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
```

```
 "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
if !wantCode {
 log.Println("To complete this example and successfully migrate a user to
Cognito, you must enter an email\n" +
  "you own that can receive a confirmation code.")
 return
}
codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
user.UserName)
if err != nil {
 panic(err)
}
log.Printf("\nA confirmation code has been sent to %v.",
*codeDelivery.Destination)
code := runner.questioner.Ask("Check your email and enter it here:")

confirmed := false
password := runner.questioner.AskPassword("\nEnter a password that has at least
eight characters, uppercase, lowercase, numbers and symbols.\n"+
 "(the password will not display as you type):", 8)
for !confirmed {
 log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
 err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
user.UserName, password)
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   password = runner.questioner.AskPassword("\nEnter another password:", 8)
  } else {
   panic(err)
  }
 } else {
  confirmed = true
 }
}
log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
log.Println("Signing in with your username and password...")
authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
password)
if err != nil {
 panic(err)
}
log.Printf("Successfully signed in. Your access token starts with: %v...\n",
(*authResult.AccessToken)[:10])
```

```
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `MigrateUser` com uma função do Lambda.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
```

```
tableName := os.Getenv(TABLE_NAME)
user := UserInfo{
 UserName: event.UserName,
}
log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
if err != nil {
 log.Printf("Error building expression to query for user '%v'.\n",
user.UserName)
 return event, err
}
output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
 TableName:                 aws.String(tableName),
 FilterExpression:          expr.Filter(),
 ExpressionAttributeNames:  expr.Names(),
 ExpressionAttributeValues: expr.Values(),
})
if err != nil {
 log.Printf("Error looking up user '%v'.\n", user.UserName)
 return event, err
}
if len(output.Items) == 0 {
 log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
 return event, err
}

var users []UserInfo
err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
if err != nil {
 log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
 return event, err
}

user = users[0]
log.Printf("UserName '%v' found with email %v. User is migrated and must reset
password.\n", user.UserName, user.UserEmail)
event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
map[string]string{
 "email":          user.UserEmail,
 "email_verified": "true", // email_verified is required for the forgot password
flow.
}
```

```
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
```

```
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
```

```
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
```

```
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int
```

```
const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
```

```
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
```

```
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
```

```
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
```

```
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
```

```
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
```

```
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
```

```
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
```

```
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
```

```
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
```

```
    }
    log.Println("Deleted user.")
   }
   triggerList := make([]actions.TriggerInfo, len(resources.triggers))
   for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
  HandlerArn: nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Crie uma API REST do API Gateway para rastrear COVID-19 dados

O exemplo de código a seguir mostra como criar uma API REST que simula um sistema para rastrear casos diários nos Estados Unidos da América, usando dados fictícios. COVID-19

Python

SDK para Python (Boto3)

Mostra como usar o AWS Chalice com o AWS SDK para Python (Boto3) para criar uma API REST sem servidor que usa o Amazon API Gateway e o Amazon DynamoDB AWS Lambda. A API REST simula um sistema que rastreia casos diários nos Estados Unidos da América, usando dados fictícios. COVID-19 Aprenda como:

- Use o AWS Chalice para definir rotas nas funções do Lambda que são chamadas para lidar com solicitações REST que chegam por meio do API Gateway.
- Usar as funções do Lambda para recuperar e armazenar dados em uma tabela do DynamoDB para atender a solicitações REST.
- Defina a estrutura da tabela e os recursos da função de segurança em um AWS CloudFormation modelo.
- Use AWS Chalice e CloudFormation para empacotar e implantar todos os recursos necessários.
- Use CloudFormation para limpar todos os recursos criados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice para criar uma API REST apoiada por um banco de dados Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Use AWS Secrets Manager para gerenciar as credenciais do banco de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST sem servidor no Amazon API Gateway e. AWS Lambda
- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Criar uma aplicação de mensageiro com o Step Functions

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon

DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.
- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

Criar uma API REST com integração de proxy de função

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para execução do Lambda.
- Criar e implantar uma função do Lambda.
- Criar uma API REST
- Configurar a integração de proxy do Lambda.
- Implantar e testar a API
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
```

```
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
```

```
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
```

```
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"
```

```
# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2
```

```
aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação de chat websocket com o API Gateway

O exemplo de código a seguir mostra como criar uma aplicação de chat que é atendido por uma API de Websocket criada no Amazon API Gateway.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon API Gateway V2 para criar uma API de websocket que se integre ao Amazon AWS Lambda DynamoDB.

- Crie uma API de Websocket atendida pelo API Gateway.
- Defina um manipulador do Lambda que armazena conexões no DynamoDB e publica mensagens para outros participantes do chat.
- Conecte-se à aplicação de chat websocket e envie mensagens com o pacote Websockets.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio

Os exemplos de código a seguir mostram como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

.NET

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.

- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado nas funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
```

```
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
```

```
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
```

```
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });
```

```
  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Como criar um painel de monitoramento com o nome da função como uma variável

O exemplo de código a seguir mostra como:

- Crie um CloudWatch painel
- Adicionar widgets de métricas do Lambda com uma variável de nome de função.
- Verificar o painel.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
```

```
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
```

```
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

```
    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
```

```
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
```

```
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
```

```
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
```

```
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
```

```
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
```

```
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AttachRolePolicy
- CreateFunction
- CreateRole
- DeleteDashboards
- DeleteFunction
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetDashboard
- Invoke
- ListDashboards
- ListFunctions
- PutDashboard

Como criar sua primeira função com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Lambda.
- Criar código da função.
- Criar uma função do Lambda
- Testar uma função do Lambda.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

################################################################################
# Helper functions
################################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""
```

```
   for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
       local RESOURCE="${CREATED_RESOURCES[$i]}"
       local TYPE="${RESOURCE%%:*}"
       local NAME="${RESOURCE#*:}"

       case "$TYPE" in
           log-group)
               echo "Deleting CloudWatch log group: ${NAME}"
               aws logs delete-log-group \
                   --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete log group ${NAME}."
               ;;
           lambda-function)
               echo "Deleting Lambda function: ${NAME}"
               aws lambda delete-function \
                   --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete Lambda function ${NAME}."
               echo "  Waiting for function deletion to complete..."
               local DELETE_WAIT=0
               while aws lambda get-function --function-name "$NAME" > /dev/null
2>&1; do
                   sleep 2
                   DELETE_WAIT=$((DELETE_WAIT + 2))
                   if [ "$DELETE_WAIT" -ge 60 ]; then
                       echo "  WARNING: Timed out waiting for function
deletion."
                       break
                   fi
               done
               ;;
           iam-role-policy)
               local ROLE_PART="${NAME%%|*}"
               local POLICY_PART="${NAME#*|}"
               echo "Detaching policy from role: ${ROLE_PART}"
               aws iam detach-role-policy \
                   --role-name "$ROLE_PART" \
                   --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
not detach policy from role ${ROLE_PART}."
               ;;
           iam-role)
               echo "Deleting IAM role: ${NAME}"
               aws iam delete-role \
```

```
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
```

```
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

################################################################################
# Runtime selection
################################################################################

echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
```

```
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
```

```
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

################################################################################
# Step 1: Create IAM execution role
################################################################################

echo ""
echo "==========================================="
echo "Step 1: Create IAM execution role"
echo "==========================================="
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
```

```
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
```

```
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
```

```
##############################################################################

echo ""
echo "============================================="
echo "Step 3: Invoke the function"
echo "============================================="
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

##############################################################################
# Step 4: View CloudWatch logs
##############################################################################
```

```
echo ""
echo "==========================================="
echo "Step 4: View CloudWatch Logs"
echo "==========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")
```

```
aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################

echo ""
echo "============================================"
echo "SUMMARY"
echo "============================================"
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction

- GetFunctionConfiguration
- GetLogEvents
- Invoke

Invocar uma função do Lambda em um navegador

O exemplo de código a seguir mostra como invocar uma AWS Lambda função em um navegador.

JavaScript

SDK para JavaScript (v2)

Você pode criar um aplicativo baseado em navegador que usa uma AWS Lambda função para atualizar uma tabela do Amazon DynamoDB com as seleções do usuário.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda

SDK para JavaScript (v3)

Você pode criar um aplicativo baseado em navegador que usa uma AWS Lambda função para atualizar uma tabela do Amazon DynamoDB com as seleções do usuário. Este aplicativo usa AWS SDK para JavaScript v3.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda

Como transformar dados para sua aplicação com o S3 Object Lambda

O exemplo de código a seguir mostra como transformar dados para sua aplicação com o S3 Object Lambda.

.NET

SDK para .NET

Mostra como adicionar código personalizado a solicitações GET padrão do S3 para modificar o objeto solicitado e recuperado do S3 e possibilitar que o objeto atenda às necessidades do cliente ou aplicação solicitante.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Lambda
- Amazon S3

Usar o API Gateway para invocar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Lambda Java. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3).

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.
- Use o pacote Requests para enviar solicitações à API REST.

- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar Step Functions para invocar funções do Lambda

O exemplo de código a seguir mostra como criar uma máquina de AWS Step Functions estado que invoca AWS Lambda funções em sequência.

Java

SDK para Java 2.x

Mostra como criar um fluxo de trabalho AWS sem servidor usando AWS Step Functions e. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Usar eventos programados para chamar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

Java

SDK para Java 2.x

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função Lambda usando a API de tempo de execução do JavaScript Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento programado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função Lambda o alvo.
- Concede permissão para permitir a EventBridge invocação da função Lambda.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações programadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Use a API do Amazon Neptune para desenvolver uma função do Lambda que consulta dados de gráficos

O exemplo de código a seguir mostra como usar a API do Neptune para consultar dados de gráficos.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Neptune para criar uma função do Lambda que consulta dados de gráficos na VPC.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Lambda
- Neptune

Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar funções do Lambda para monitoramento.
- Crie um CloudWatch painel
- Adicionar uma variável de propriedade ao painel.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
```

```
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM
```

```
# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1
 >> "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/
dev/null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"
```

```
    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
```

```
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
```

```
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
```

```
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /
dev/null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /
dev/null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a
 "$LOG_FILE"

# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
```

```
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name
 "$DASHBOARD_NAME" --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch
 dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/" | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee
 -a "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
```

```
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME"
 --output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AttachRolePolicy
- CreateFunction
- CreateRole
- DeleteDashboards
- DeleteFunction
- DeleteRole
- DetachRolePolicy
- GetDashboard
- Invoke
- PutDashboard

Grave dados de atividades personalizados com uma função Lambda após a autenticação do usuário do Amazon Cognito usando um AWS SDK

O exemplo de código a seguir mostra como gravar dados de atividade personalizados com uma função do Lambda depois da autenticação do usuário do Amazon Cognito.

- Use as funções de administrador para adicionar um usuário a um grupo de usuários.
- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PostAuthentication`.
- Faça login do novo usuário no Amazon Cognito.
- A função Lambda grava informações personalizadas em CloudWatch Logs e em uma tabela do DynamoDB.
- Obtenha e veja dados personalizados da tabela do DynamoDB e, em seguida, limpe os recursos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
```

```
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
```

```
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string,
 userName string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
```

```
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `PostAuthentication` com uma função do Lambda.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}
```

```
// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
```

```
    TableName: aws.String(tableName),
   })
   if err != nil {
    log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
   } else {
    log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
   }
  }

  return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)
```

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}
```

```
// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
```

```
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
```

```
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
```

```
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
```

```
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
```

```
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
```

```
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
```

```
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}
```

```
// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
```

```
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Exemplos de tecnologia sem servidor para o Lambda

Os exemplos de código a seguir mostram como usar o Lambda com AWS SDKs.

Exemplos

- Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda
- Invocar uma função do Lambda em um trigger do Kinesis
- Invocar uma função do Lambda em um gatilho do DynamoDB
- Invocar uma função do Lambda de um acionador do Amazon DocumentDB
- Invocar uma função do Lambda em um gatinho do Amazon MSK
- Invocar uma função do Lambda em um acionador do Amazon S3
- Invocar uma função do Lambda em um acionador do Amazon SNS
- Invocar uma função do Lambda em um trigger do Amazon SQS
- Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis
- Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB
- Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

Os exemplos de código a seguir mostram como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando .NET.

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;
```

```
// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace aws_rds;

public class InputModel
{
    public string key1 { get; set; }
    public string key2 { get; set; }
}

public class Function
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
 FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
 $"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

 $"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +
```

```
$"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                $"Pwd={authToken};";


       try
       {
           await using var connection = new MySqlConnection(connectionString);
           await connection.OpenAsync();

           const string sql = "SELECT @param1 + @param2 AS Sum";

           await using var command = new MySqlCommand(sql, connection);
           command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
"0"));
           command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
"0"));

           await using var reader = await command.ExecuteReaderAsync();
           if (await reader.ReadAsync())
           {
               int result = reader.GetInt32("Sum");

               //Sample Response: {"statusCode":200,"body":"{\"message\":\"The
sum is: 45\"}","isBase64Encoded":false}
               return new APIGatewayProxyResponse
               {
                   StatusCode = 200,
                   Body = JsonSerializer.Serialize(new { message = $"The sum is:
{result}" })
               };
           }

       }
       catch (Exception ex)
       {
           Console.WriteLine($"Error: {ex.Message}");
       }

       return new APIGatewayProxyResponse
       {
           StatusCode = 500,
           Body = JsonSerializer.Serialize(new { error = "Internal server
error" })
```

```
        };
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Go.

```
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"

 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {
```

```
var dbName string = os.Getenv("DatabaseName")
var dbUser string = os.Getenv("DatabaseUser")
var dbHost string = os.Getenv("DBHost") // Add hostname without https
var dbPort int = os.Getenv("Port")      // Add port number
var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
var region string = os.Getenv("AWS_REGION")

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
 panic("configuration error: " + err.Error())
}

authenticationToken, err := auth.BuildAuthToken(
 context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
if err != nil {
 panic("failed to create authentication token: " + err.Error())
}

dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
 dbUser, authenticationToken, dbEndpoint, dbName,
)

db, err := sql.Open("mysql", dsn)
if err != nil {
 panic(err)
}

defer db.Close()

var sum int
err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
if err != nil {
 panic(err)
}
s := fmt.Sprint(sum)
message := fmt.Sprintf("The selected sum is: %s", s)

messageBytes, err := json.Marshal(message)
if err != nil {
 return nil, err
}

messageString := string(messageBytes)
return map[string]interface{}{
```

```
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements
 RequestHandler<APIGatewayProxyRequestEvent, APIGatewayProxyResponseEvent> {
```

```
    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new
 APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"),
 token);
                 PreparedStatement statement =
 connection.prepareStatement("SELECT ? + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }
```

```
    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response =
 rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectando-se a um banco de dados do Amazon RDS em uma função Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
```

```
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}
```

```
export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

Conectando-se a um banco de dados do Amazon RDS em uma função Lambda usando. TypeScript

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that
 the DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}
```

```
async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number;
 body: string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
    };
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];
```

```
        // Create RDS AuthTokenGenerator object
        $generator = new
AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }
```

```
    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Python.

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
```

```
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
```

```
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Rust.

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
```

```
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
        port = port,
        db_user = db_username
    );

    let signable_request =
```

```
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
```

```
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

Invocar uma função do Lambda em um trigger do Kinesis

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
```

```
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace KinesisIntegrationSampleCode;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task FunctionHandler(KinesisEvent evnt, ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return;
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                throw;
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
```

```
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
 if len(kinesisEvent.Records) == 0 {
  log.Printf("empty Kinesis event received")
  return nil
 }

 for _, record := range kinesisEvent.Records {
  log.Printf("processed Kinesis event with EventId: %v", record.EventID)
  recordDataBytes := record.Kinesis.Data
  recordDataText := string(recordDataBytes)
```

```
  log.Printf("record data: %v", recordDataText)
  // TODO: Do interesting work based on the new data
 }
 log.Printf("successfully processed %v records", len(kinesisEvent.Records))
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
```

```
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+"
 records");
        return null;
    }

}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do Kinesis com o uso do Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
```

```
      throw err;
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Consumindo um evento do Kinesis com o uso do Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
```

```
      throw err;
    }
    logger.info(`Successfully processed ${event.Records.length} records.`);
  }
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends KinesisHandler
{
```

```
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be
 marked as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import base64
def lambda_handler(event, context):

    for record in event['Records']:
        try:
            print(f"Processed Kinesis Event - EventID: {record['eventID']}")
            record_data = base64.b64decode(record['kinesis']
['data']).decode('utf-8')
            print(f"Record Data: {record_data}")
            # TODO: Do interesting work based on the new data
        except Exception as e:
            print(f"An error occurred {e}")
            raise e
    print(f"Successfully processed {len(event['Records'])} records.")
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
```

```
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do Kinesis com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::kinesis::KinesisEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) -> Result<(), Error>
 {
    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    event.payload.records.iter().for_each(|record| {
```

```
        tracing::info!("EventId:
 {}",record.event_id.as_deref().unwrap_or_default());

        let record_data = std::str::from_utf8(&record.kinesis.data);

        match record_data {
            Ok(data) => {
                // log the record data
                tracing::info!("Data: {}", data);
            }
            Err(e) => {
                tracing::error!("Error: {}", e);
            }
        }
    });

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Invocar uma função do Lambda em um gatilho do DynamoDB

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace AWSLambda_DDB;

public class Function
{
    public void FunctionHandler(DynamoDBEvent dynamoEvent, ILambdaContext
 context)
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");

        foreach (var record in dynamoEvent.Records)
        {
```

```
            context.Logger.LogInformation($"Event ID: {record.EventID}");
            context.Logger.LogInformation($"Event Name: {record.EventName}");

            context.Logger.LogInformation(JsonSerializer.Serialize(record));
        }

        context.Logger.LogInformation("Stream processing complete.");
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-lambda-go/events"
 "fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string,
 error) {
 if len(event.Records) == 0 {
  return nil, fmt.Errorf("received empty event")
 }

 for _, record := range event.Records {
   LogDynamoDBRecord(record)
```

```
 }

 message := fmt.Sprintf("Records processed: %d", len(event.Records))
 return &message, nil
}

func main() {
 lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
 fmt.Println(record.EventID)
 fmt.Println(record.EventName)
 fmt.Printf("%+v\n", record.Change)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do DynamoDB com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new
 GsonBuilder().setPrettyPrinting().create();

    @Override
```

```
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " +
 GSON.toJson(record.getDynamodb()));
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do DynamoDB com o uso do Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
};

const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Consumindo um evento do DynamoDB com o uso do Lambda. TypeScript

```
export const handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
}
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do DynamoDB com o Lambda usando PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;
```

```
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be
 marked as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Python.

```
import json

def lambda_handler(event, context):
    print(json.dumps(event, indent=2))

    for record in event['Records']:
        log_dynamodb_record(record)

def log_dynamodb_record(record):
    print(record['eventID'])
    print(record['eventName'])
    print(f"DynamoDB Record: {json.dumps(record['dynamodb'])}")
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
```

```
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<Event>) ->Result<(), Error> {

    let records = &event.payload.records;
    tracing::info!("event payload: {:?}",records);
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_dynamo_dbrecord(record);
    }

    tracing::info!("Dynamo db records processed");

    // Prepare the response
    Ok(())

}

fn log_dynamo_dbrecord(record: &EventRecord)-> Result<(), Error>{
    tracing::info!("EventId: {}", record.event_id);
    tracing::info!("EventName: {}", record.event_name);
    tracing::info!("DynamoDB Record: {:?}", record.change );
    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())
```

```
}
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Processando um evento do Amazon DocumentDB com o Lambda ao usar .NET.

```
using Amazon.Lambda.Core;
using System.Text.Json;
using System;
using System.Collections.Generic;
using System.Text.Json.Serialization;
//Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace LambdaDocDb;

public class Function
{

     /// <summary>
    /// Lambda function entry point to process Amazon DocumentDB events.
    /// </summary>
    /// <param name="event">The Amazon DocumentDB event.</param>
    /// <param name="context">The Lambda context object.</param>
```

```
    /// <returns>A string to indicate successful processing.</returns>
    public string FunctionHandler(Event evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Events)
        {
            ProcessDocumentDBEvent(record, context);
        }

        return "OK";
    }

     private void ProcessDocumentDBEvent(DocumentDBEventRecord record,
 ILambdaContext context)
    {

        var eventData = record.Event;
        var operationType = eventData.OperationType;
        var databaseName = eventData.Ns.Db;
        var collectionName = eventData.Ns.Coll;
        var fullDocument = JsonSerializer.Serialize(eventData.FullDocument, new
 JsonSerializerOptions { WriteIndented = true });

        context.Logger.LogLine($"Operation type: {operationType}");
        context.Logger.LogLine($"Database: {databaseName}");
        context.Logger.LogLine($"Collection: {collectionName}");
        context.Logger.LogLine($"Full document:\n{fullDocument}");
    }



    public class Event
    {
        [JsonPropertyName("eventSourceArn")]
        public string EventSourceArn { get; set; }

        [JsonPropertyName("events")]
        public List<DocumentDBEventRecord> Events { get; set; }

        [JsonPropertyName("eventSource")]
        public string EventSource { get; set; }
    }

    public class DocumentDBEventRecord
```

```
    {
        [JsonPropertyName("event")]
        public EventData Event { get; set; }
    }

    public class EventData
    {
        [JsonPropertyName("_id")]
        public IdData Id { get; set; }

        [JsonPropertyName("clusterTime")]
        public ClusterTime ClusterTime { get; set; }

        [JsonPropertyName("documentKey")]
        public DocumentKey DocumentKey { get; set; }

        [JsonPropertyName("fullDocument")]
        public Dictionary<string, object> FullDocument { get; set; }

        [JsonPropertyName("ns")]
        public Namespace Ns { get; set; }

        [JsonPropertyName("operationType")]
        public string OperationType { get; set; }
    }

    public class IdData
    {
        [JsonPropertyName("_data")]
        public string Data { get; set; }
    }

    public class ClusterTime
    {
        [JsonPropertyName("$timestamp")]
        public Timestamp Timestamp { get; set; }
    }

    public class Timestamp
    {
        [JsonPropertyName("t")]
        public long T { get; set; }

        [JsonPropertyName("i")]
```

```
        public int I { get; set; }
    }

    public class DocumentKey
    {
        [JsonPropertyName("_id")]
        public Id Id { get; set; }
    }

    public class Id
    {
        [JsonPropertyName("$oid")]
        public string Oid { get; set; }
    }

    public class Namespace
    {
        [JsonPropertyName("db")]
        public string Db { get; set; }

        [JsonPropertyName("coll")]
        public string Coll { get; set; }
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Go.

```
package main

import (
```

```
 "context"
 "encoding/json"
 "fmt"

 "github.com/aws/aws-lambda-go/lambda"
)

type Event struct {
 Events []Record `json:"events"`
}

type Record struct {
 Event struct {
  OperationType string `json:"operationType"`
  NS            struct {
   DB   string `json:"db"`
   Coll string `json:"coll"`
  } `json:"ns"`
  FullDocument interface{} `json:"fullDocument"`
 } `json:"event"`
}

func main() {
 lambda.Start(handler)
}

func handler(ctx context.Context, event Event) (string, error) {
 fmt.Println("Loading function")
 for _, record := range event.Events {
  logDocumentDBEvent(record)
 }

 return "OK", nil
}

func logDocumentDBEvent(record Record) {
 fmt.Printf("Operation type: %s\n", record.Event.OperationType)
 fmt.Printf("db: %s\n", record.Event.NS.DB)
 fmt.Printf("collection: %s\n", record.Event.NS.Coll)
 docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")
 fmt.Printf("Full document: %s\n", string(docBytes))
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
```

```
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do Amazon DocumentDB com o uso do Lambda. JavaScript

```
console.log('Loading function');
exports.handler = async (event, context) => {
    event.events.forEach(record => {
        logDocumentDBEvent(record);
    });
    return 'OK';
};

const logDocumentDBEvent = (record) => {
    console.log('Operation type: ' + record.event.operationType);
    console.log('db: ' + record.event.ns.db);
    console.log('collection: ' + record.event.ns.coll);
    console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

Consumindo um evento do Amazon DocumentDB com o Lambda usando TypeScript

```
import { DocumentDBEventRecord, DocumentDBEventSubscriptionContext } from 'aws-
lambda';

console.log('Loading function');

export const handler = async (
  event: DocumentDBEventSubscriptionContext,
  context: any
): Promise<string> => {
  event.events.forEach((record: DocumentDBEventRecord) => {
    logDocumentDBEvent(record);
  });
  return 'OK';
};

const logDocumentDBEvent = (record: DocumentDBEventRecord): void => {
  console.log('Operation type: ' + record.event.operationType);
  console.log('db: ' + record.event.ns.db);
  console.log('collection: ' + record.event.ns.coll);
  console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando PHP.

```
<?php

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

```
use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Python.

```
import json

def lambda_handler(event, context):
    for record in event.get('events', []):
        log_document_db_event(record)
    return 'OK'

def log_document_db_event(record):
    event_data = record.get('event', {})
    operation_type = event_data.get('operationType', 'Unknown')
    db = event_data.get('ns', {}).get('db', 'Unknown')
    collection = event_data.get('ns', {}).get('coll', 'Unknown')
    full_document = event_data.get('fullDocument', {})

    print(f"Operation type: {operation_type}")
    print(f"db: {db}")
    print(f"collection: {collection}")
    print("Full document:", json.dumps(full_document, indent=2))
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::documentdb::{DocumentDbEvent, DocumentDbInnerEvent},
   };
```

```
// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<DocumentDbEvent>) ->Result<(),
 Error> {

    tracing::info!("Event Source ARN: {:?}", event.payload.event_source_arn);
    tracing::info!("Event Source: {:?}", event.payload.event_source);

    let records = &event.payload.events;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_document_db_event(record);
    }

    tracing::info!("Document db records processed");

    // Prepare the response
    Ok(())

}

fn log_document_db_event(record: &DocumentDbInnerEvent)-> Result<(), Error>{
    tracing::info!("Change Event: {:?}", record.event);

    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
```

```
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento disparado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando .NET.

```
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KafkaEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
```

```
namespace MSKLambda;

public class Function
{


    /// <param name="input">The event for the Lambda function handler to
 process.</param>
    /// <param name="context">The ILambdaContext that provides methods for
 logging and describing the Lambda environment.</param>
    /// <returns></returns>
    public void FunctionHandler(KafkaEvent evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Records)
        {
            Console.WriteLine("Key:" + record.Key);
            foreach (var eventRecord in record.Value)
            {
                var valueBytes = eventRecord.Value.ToArray();
                var valueText = Encoding.UTF8.GetString(valueBytes);

                Console.WriteLine("Message:" + valueText);
            }
        }
    }


}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando Go.

```
package main

import (
 "encoding/base64"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
 for key, records := range event.Records {
  fmt.Println("Key:", key)

  for _, record := range records {
   fmt.Println("Record:", record)

   decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
   message := string(decodedValue)
   fmt.Println("Message:", message)
  }
 }
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do Amazon MSK com o uso do JavaScript Lambda.

```
exports.handler = async (event) => {
    // Iterate through keys
    for (let key in event.records) {
      console.log('Key: ', key)
      // Iterate through records
      event.records[key].map((record) => {
        console.log('Record: ', record)
        // Decode base64
        const msg = Buffer.from(record.value, 'base64').toString()
        console.log('Message:', msg)
      })
    }
}
```

Consumindo um evento do Amazon MSK com o uso do TypeScript Lambda.

```
import { MSKEvent, Context } from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "msk-handler-sample",
});

export const handler = async (
  event: MSKEvent,
  context: Context
): Promise<void> => {
  for (const [topic, topicRecords] of Object.entries(event.records)) {
    logger.info(`Processing key: ${topic}`);

    // Process each record in the partition
    for (const record of topicRecords) {
      try {
        // Decode the message value from base64
        const decodedMessage = Buffer.from(record.value, 'base64').toString();

        logger.info({
          message: decodedMessage
```

```
        });
      }
      catch (error) {
        logger.error('Error processing event', { error });
        throw error;
      }
    };
  }
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Python.

```
import base64

def lambda_handler(event, context):
    # Iterate through keys
    for key in event['records']:
        print('Key:', key)
        # Iterate through records
        for record in event['records'][key]:
            print('Record:', record)
            # Decode base64
            msg = base64.b64decode(record['value']).decode('utf-8')
            print('Message:', msg)
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Ruby.

```
require 'base64'
```

```
def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando o Rust.

```
use aws_lambda_events::event::kafka::KafkaEvent;
use lambda_runtime::{run, service_fn, tracing, Error, LambdaEvent};
use base64::prelude::*;
use serde_json::{Value};
use tracing::{info};

/// Pre-Requisites:
/// 1. Install Cargo Lambda - see https://www.cargo-lambda.info/guide/getting-
started.html
/// 2. Add packages tracing, tracing-subscriber, serde_json, base64
///
/// This is the main body for the function.
/// Write your code inside it.
```

```
/// There are some code example in the following URLs:
/// - https://github.com/awslabs/aws-lambda-rust-runtime/tree/main/examples
/// - https://github.com/aws-samples/serverless-rust-demo/

async fn function_handler(event: LambdaEvent<KafkaEvent>) -> Result<Value, Error>
 {

    let payload = event.payload.records;

    for (_name, records) in payload.iter() {

        for record in records {

         let record_text = record.value.as_ref().ok_or("Value is None")?;
         info!("Record: {}", &record_text);

         // perform Base64 decoding
         let record_bytes = BASE64_STANDARD.decode(record_text)?;
         let message = std::str::from_utf8(&record_bytes)?;

         info!("Message: {}", message);
        }

    }

    Ok(().into())
}

#[tokio::main]
async fn main() -> Result<(), Error> {

    // required to enable CloudWatch error logging by the runtime
    tracing::init_default_subscriber();
    info!("Setup CW subscriber!");

    run(service_fn(function_handler)).await
}
```

Invocar uma função do Lambda em um acionador do Amazon S3

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto em um bucket do S3. A função recupera o nome

do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }
```

```
        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request -
 {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
```

```
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
```

```
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do S3 com o uso do JavaScript Lambda.

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
```

```
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g,
 ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Consumindo um evento do S3 com o uso do TypeScript Lambda.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
```

```
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do S3 com o Lambda usando PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

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


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
```

```
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
 $bucket . '. Make sure they exist and your bucket is in the same region as this
 function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
 your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do S3 com o Lambda usando Ruby.

```
require 'json'
require 'uri'
```

```
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
```

```
/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
 name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
```

```
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

Invocar uma função do Lambda em um acionador do Amazon SNS

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
```

```
namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
```

```
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do SNS com o JavaScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
```

```
    throw err;
  }
}
```

Consumindo um evento do SNS com o TypeScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SNS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
```

```
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Invocar uma função do Lambda em um trigger do Amazon SQS

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }

        context.Logger.LogInformation("done");
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
```

```
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }

    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
 for _, record := range event.Records {
  err := processMessage(record)
  if err != nil {
   return err
  }
```

```
 }
 fmt.Println("done")
 return nil
}

func processMessage(record events.SQSMessage) error {
 fmt.Printf("Processed message %s\n", record.Body)
 // TODO: Do interesting work based on the new message
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
```

```
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }

    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento SQS com o JavaScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message) {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
```

```
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Consumindo um evento SQS com o TypeScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";

export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
 record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

Os exemplos de código a seguir mostram como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using System.Text.Json.Serialization;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace KinesisIntegration;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task<StreamsEventResponse> FunctionHandler(KinesisEvent evnt,
 ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return new StreamsEventResponse();
        }

        foreach (var record in evnt.Records)
        {
```

```
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                return new StreamsEventResponse
                {
                    BatchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>
                    {
                        new StreamsEventResponse.BatchItemFailure
 { ItemIdentifier = record.Kinesis.SequenceNumber }
                    }
                };
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
        return new StreamsEventResponse();
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}

public class StreamsEventResponse
{
    [JsonPropertyName("batchItemFailures")]
```

```
    public IList<BatchItemFailure> BatchItemFailures { get; set; }
    public class BatchItemFailure
    {
        [JsonPropertyName("itemIdentifier")]
        public string ItemIdentifier { get; set; }
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, record := range kinesisEvent.Records {
  curRecordSequenceNumber := ""

  // Process your record
  if /* Your record processing condition here */ {
   curRecordSequenceNumber = record.Kinesis.SequenceNumber
  }
```

```
  // Add a condition to check if the record processing failed
  if curRecordSequenceNumber != "" {
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": curRecordSequenceNumber})
  }
 }

 kinesisBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return kinesisBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
```

```
public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context
 context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord =
 kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Javascript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed
 item onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Relatando falhas de itens em lote do Kinesis com o uso do Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
  KinesisStreamBatchResponse,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<KinesisStreamBatchResponse> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  logger.info(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};
```

```
async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["kinesis"]["sequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de item em lote do Kinesis com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relate falhas de itens em lote do Kinesis com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::kinesis::KinesisEvent,
    kinesis::KinesisEventRecord,
    streams::{KinesisBatchItemFailure, KinesisEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) ->
 Result<KinesisEventResponse, Error> {
    let mut response = KinesisEventResponse {
        batch_item_failures: vec![],
    };

    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in &event.payload.records {
        tracing::info!(
            "EventId: {}",
            record.event_id.as_deref().unwrap_or_default()
        );

        let record_processing_result = process_record(record);

        if record_processing_result.is_err() {
            response.batch_item_failures.push(KinesisBatchItemFailure {
                item_identifier: record.kinesis.sequence_number.clone(),
```

```
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
            return Ok(response);
        }
    }

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(response)
}

fn process_record(record: &KinesisEventRecord) -> Result<(), Error> {
    let record_data = std::str::from_utf8(record.kinesis.data.as_slice());

    if let Some(err) = record_data.err() {
        tracing::error!("Error: {}", err);
        return Err(Error::from(err));
    }

    let record_data = record_data.unwrap_or_default();

    // do something interesting with the data
    tracing::info!("Data: {}", record_data);

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();
```

```
    run(service_fn(function_handler)).await
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

Os exemplos de código a seguir mostram como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace AWSLambda_DDB;

public class Function
{
    public StreamsEventResponse FunctionHandler(DynamoDBEvent dynamoEvent,
 ILambdaContext context)
```

```
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>();
        StreamsEventResponse streamsEventResponse = new StreamsEventResponse();

        foreach (var record in dynamoEvent.Records)
        {
            try
            {
                var sequenceNumber = record.Dynamodb.SequenceNumber;
                context.Logger.LogInformation(sequenceNumber);
            }
            catch (Exception ex)
            {
                context.Logger.LogError(ex.Message);
                batchItemFailures.Add(new StreamsEventResponse.BatchItemFailure()
 { ItemIdentifier = record.Dynamodb.SequenceNumber });
            }
        }

        if (batchItemFailures.Count > 0)
        {
            streamsEventResponse.BatchItemFailures = batchItemFailures;
        }

        context.Logger.LogInformation("Stream processing complete.");
        return streamsEventResponse;
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
 ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
 BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent)
 (*BatchResult, error) {
 var batchItemFailures []BatchItemFailure
 curRecordSequenceNumber := ""

 for _, record := range event.Records {
  // Process your record
  curRecordSequenceNumber = record.Change.SequenceNumber
 }

 if curRecordSequenceNumber != "" {
  batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
 curRecordSequenceNumber})
 }

 batchResult := BatchResult{
  BatchItemFailures: batchItemFailures,
 }

 return &batchResult, nil
}

func main() {
 lambda.Start(HandleRequest)
```

```
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context
 context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
```

```
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatando falhas de itens em lote do DynamoDB com o uso do Lambda. JavaScript

```
export const handler = async (event) => {
  const records = event.Records;
  let curRecordSequenceNumber = "";

  for (const record of records) {
    try {
      // Process your record
      curRecordSequenceNumber = record.dynamodb.SequenceNumber;
    } catch (e) {
      // Return failed record's sequence number
      return { batchItemFailures: [{ itemIdentifier:
 curRecordSequenceNumber }] };
```

```
    }
  }

  return { batchItemFailures: [] };
};
```

Relatando falhas de itens em lote do DynamoDB com o uso do Lambda. TypeScript

```
import {
  DynamoDBBatchResponse,
  DynamoDBBatchItemFailure,
  DynamoDBStreamEvent,
} from "aws-lambda";

export const handler = async (
  event: DynamoDBStreamEvent
): Promise<DynamoDBBatchResponse> => {
  const batchItemFailures: DynamoDBBatchItemFailure[] = [];
  let curRecordSequenceNumber;

  for (const record of event.Records) {
    curRecordSequenceNumber = record.dynamodb?.SequenceNumber;

    if (curRecordSequenceNumber) {
      batchItemFailures.push({
        itemIdentifier: curRecordSequenceNumber,
      });
    }
  }

  return { batchItemFailures: batchItemFailures };
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do DynamoDB com o Lambda usando PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
```

```
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["dynamodb"]["SequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
```

```
    end

    {"batchItemFailures" => []}
  end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Rust.

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord, StreamRecord},
    streams::{DynamoDbBatchItemFailure, DynamoDbEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Process the stream record
fn process_record(record: &EventRecord) -> Result<(), Error> {
    let stream_record: &StreamRecord = &record.change;

    // process your stream record here...
    tracing::info!("Data: {:?}", stream_record);

    Ok(())
}

/// Main Lambda handler here...
async fn function_handler(event: LambdaEvent<Event>) ->
 Result<DynamoDbEventResponse, Error> {
    let mut response = DynamoDbEventResponse {
        batch_item_failures: vec![],
    };

    let records = &event.payload.records;
```

```
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in records {
        tracing::info!("EventId: {}", record.event_id);

        // Couldn't find a sequence number
        if record.change.sequence_number.is_none() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: Some("".to_string()),
            });
            return Ok(response);
        }

        // Process your record here...
        if process_record(record).is_err() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: record.change.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
            return Ok(response);
        }
    }

    tracing::info!("Successfully processed {} record(s)", records.len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();
```

```
    run(service_fn(function_handler)).await
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

Os exemplos de código a seguir mostram como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
namespace sqsSample;

public class Function
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
 ILambdaContext context)
    {
```

```
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
 SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
        context.Logger.LogInformation($"Processed message {message.Body}");
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, message := range sqsEvent.Records {
  if len(message.Body) > 0 {
   // Your message processing condition here
   fmt.Printf("Successfully processed message: %s\n", message.Body)
  } else {
   // Message processing failed
   fmt.Printf("Failed to process message %s\n", message.MessageId)
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": message.MessageId})
  }
 }

 sqsBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return sqsBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
             }
         }
         return new SQSBatchResponse(batchItemFailures);
     }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatando falhas de itens em lote do SQS com o JavaScript Lambda usando.

```
// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
    console.log(`Processed message: ${record.body}`);
}
```

Relatando falhas de itens em lote do SQS com o TypeScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord }
 from 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
 Promise<SQSBatchResponse> => {
    const batchItemFailures: SQSBatchItemFailure[] = [];
```

```
    for (const record of event.Records) {
        try {
            await processMessageAsync(record);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }

    return {batchItemFailures: batchItemFailures};
};

async function processMessageAsync(record: SQSRecord): Promise<void> {
    if (record.body && record.body.includes("error")) {
        throw new Error('There is an error in the SQS Message.');
    }
    console.log(`Processed message ${record.body}`);
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

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

```
class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

def lambda_handler(event, context):
    if event:
        batch_item_failures = []
        sqs_batch_response = {}

        for record in event["Records"]:
            try:
                print(f"Processed message: {record['body']}")
            except Exception as e:
                batch_item_failures.append({"itemIdentifier":
 record['messageId']})

        sqs_batch_response["batchItemFailures"] = batch_item_failures
        return sqs_batch_response
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
```

```
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) ->
 Result<SqsBatchResponse, Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
        }
    }

    Ok(SqsBatchResponse {
        batch_item_failures,
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(function_handler)).await
}
```

AWS contribuições da comunidade para o Lambda

AWS as contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Exemplos

- Compilar e testar uma aplicação com tecnologia sem servidor

Compilar e testar uma aplicação com tecnologia sem servidor

Os exemplos de código a seguir mostram como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

.NET

SDK para .NET

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK .NET.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Go

SDK para Go V2

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Go.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Java

SDK para Java 2.x

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Java.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Rust

SDK para Rust

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Rust.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos de código para o Amazon Lex usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Lex com um kit AWS de desenvolvimento de software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Lex: mais informações sobre o Amazon Lex.
- Referência da API Amazon Lex: detalhes sobre todas as ações disponíveis do Amazon Lex.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para o uso do Amazon Lex AWS SDKs
  - Crie um chatbot do Amazon Lex para engajar os visitantes do site

Cenários para o uso do Amazon Lex AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Lex com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon Lex ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie um chatbot do Amazon Lex para engajar os visitantes do site

Crie um chatbot do Amazon Lex para engajar os visitantes do site

Os exemplos de código a seguir mostram como criar um chatbot para engajar os visitantes do seu site.

Java

SDK para Java 2.x

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o exemplo completo Criando um chatbot Amazon Lex no guia do AWS SDK para JavaScript desenvolvedor.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Exemplos de código para o Lightsail usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Lightsail com AWS um kit de desenvolvimento de software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Lightsail — Mais informações sobre o Lightsail.
- Referência da API Lightsail — Detalhes sobre todas as ações disponíveis do Lightsail.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para o uso do Lightsail AWS SDKs
  - Introdução aos servidores privados virtuais simples

Cenários para o uso do Lightsail AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no AWS Lightsail com SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Lightsail ou combinadas com outras. Serviços da AWS Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução aos servidores privados virtuais simples

Introdução aos servidores privados virtuais simples

O exemplo de código a seguir mostra como:

- Criar uma instância do
- Adicionar armazenamento à sua instância
- Criar um snapshot
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Lightsail Getting Started CLI Script
# This script demonstrates how to create and manage Lightsail resources using the
 AWS CLI

set -euo pipefail

# Set AWS region
export AWS_REGION="${AWS_REGION:-us-west-2}"
echo "Using AWS region: $AWS_REGION"

# Set up logging
LOG_FILE="lightsail-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Lightsail Getting Started script at $(date)"

# Error handling function
handle_error() {
  echo "ERROR: $1" >&2
  echo "Attempting to clean up resources..."
  cleanup_resources || true
  exit 1
}

# Function to check if a command succeeded
check_status() {
  if [ $? -ne 0 ]; then
    handle_error "$1"
  fi
}
```

```
# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
INSTANCE_NAME="LightsailInstance-${RANDOM_ID}"
DISK_NAME="LightsailDisk-${RANDOM_ID}"
SNAPSHOT_NAME="LightsailSnapshot-${RANDOM_ID}"

# Array to track created resources
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
  CREATED_RESOURCES+=("$1:$2")
  echo "Created $1: $2"
}

# Function to clean up resources
cleanup_resources() {
  echo "Resources created by this script:"
  for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "  $resource"
  done

  # Reverse the array to delete resources in reverse order
  for (( idx=${#CREATED_RESOURCES[@]}-1 ; idx>=0 ; idx-- )); do
    IFS=':' read -r type name <<< "${CREATED_RESOURCES[idx]}"

    case "$type" in
      "instance_snapshot")
        echo "Deleting instance snapshot: $name"
        aws lightsail delete-instance-snapshot --instance-snapshot-name "$name"
 --region "$AWS_REGION" 2>/dev/null || true
        ;;
      "disk_snapshot")
        echo "Deleting disk snapshot: $name"
        aws lightsail delete-disk-snapshot --disk-snapshot-name "$name" --region
 "$AWS_REGION" 2>/dev/null || true
        ;;
      "disk")
        echo "Detaching disk: $name"
        aws lightsail detach-disk --disk-name "$name" --region "$AWS_REGION" 2>/
dev/null || true
        sleep 10
        echo "Deleting disk: $name"
```

```
        aws lightsail delete-disk --disk-name "$name" --region "$AWS_REGION" 2>/
dev/null || true
        ;;
      "instance")
        echo "Deleting instance: $name"
        INSTANCE_STATE=$(aws lightsail get-instance-state --instance-name "$name"
 --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null || echo
 "unknown")
        if [ "$INSTANCE_STATE" == "pending" ]; then
          echo "Instance is in pending state. Waiting for it to be ready before
 deleting..."
          MAX_WAIT=30
          WAITED=0
          while [ "$INSTANCE_STATE" == "pending" ] && [ $WAITED -lt $MAX_WAIT ];
 do
            sleep 10
            WAITED=$((WAITED+1))
            INSTANCE_STATE=$(aws lightsail get-instance-state --instance-name
 "$name" --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null ||
 echo "unknown")
            echo "Instance state: $INSTANCE_STATE"
          done
        fi
        aws lightsail delete-instance --instance-name "$name" --region
 "$AWS_REGION" 2>/dev/null || true
        ;;
    esac
  done

  echo "Cleanup completed"
}

# Step 1: Verify AWS CLI configuration
echo "Step 1: Verifying AWS CLI configuration"
aws configure list
check_status "Failed to verify AWS CLI configuration"

# Step 2: Get available blueprints and bundles
echo "Step 2: Getting available blueprints and bundles"
echo "Available blueprints (showing first 5):"
aws lightsail get-blueprints --region "$AWS_REGION" --query 'blueprints[0:5].
[blueprintId,name]' --output table
check_status "Failed to get blueprints"
```

```
echo "Available bundles (showing first 5):"
aws lightsail get-bundles --region "$AWS_REGION" --query 'bundles[0:5].
[bundleId,name,price]' --output table
check_status "Failed to get bundles"

# Get available regions and availability zones
echo "Getting available regions and availability zones"
# Use a specific availability zone in us-west-2 region
AVAILABILITY_ZONE="us-west-2a"
echo "Using availability zone: $AVAILABILITY_ZONE"

# Step 3: Create a Lightsail instance
echo "Step 3: Creating Lightsail instance: $INSTANCE_NAME"
aws lightsail create-instances \
  --instance-names "$INSTANCE_NAME" \
  --availability-zone "$AVAILABILITY_ZONE" \
  --blueprint-id amazon_linux_2023 \
  --bundle-id nano_3_0 \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create Lightsail instance"
track_resource "instance" "$INSTANCE_NAME"

# Wait for the instance to be in a running state
echo "Waiting for instance to be in running state..."
MAX_ATTEMPTS=30
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  STATUS=$(aws lightsail get-instance-state --instance-name "$INSTANCE_NAME"
 --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null || echo
 "unknown")
  if [ "$STATUS" == "running" ]; then
    echo "Instance is now running"
    break
  fi
  echo "Instance status: $STATUS. Waiting..."
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  handle_error "Instance failed to reach running state after 5 minutes"
fi
```

```
# Get instance details
echo "Getting instance details"
INSTANCE_IP=$(aws lightsail get-instance --instance-name "$INSTANCE_NAME" --
region "$AWS_REGION" --query 'instance.publicIpAddress' --output text)
check_status "Failed to get instance IP address"
echo "Instance IP address: $INSTANCE_IP"

# Validate IP address format
if ! [[ "$INSTANCE_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
  handle_error "Invalid IP address format: $INSTANCE_IP"
fi

# Step 4: Download the default key pair
echo "Step 4: Downloading default key pair"
KEY_FILE="lightsail_key_${RANDOM_ID}.pem"
umask 077
aws lightsail download-default-key-pair --region "$AWS_REGION" --output text >
 "$KEY_FILE"
check_status "Failed to download key pair"

if [ ! -f "$KEY_FILE" ] || [ ! -s "$KEY_FILE" ]; then
  handle_error "Key pair file was not created or is empty"
fi

chmod 400 "$KEY_FILE"
check_status "Failed to set permissions on key pair"
echo "Key pair downloaded to $KEY_FILE"

echo "To connect to your instance, use:"
echo "ssh -i $KEY_FILE ec2-user@$INSTANCE_IP"

# Step 5: Create a block storage disk
echo "Step 5: Creating block storage disk: $DISK_NAME"
aws lightsail create-disk \
  --disk-name "$DISK_NAME" \
  --availability-zone "$AVAILABILITY_ZONE" \
  --size-in-gb 8 \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create disk"
track_resource "disk" "$DISK_NAME"

# Wait for the disk to be available using polling
```

```
echo "Waiting for disk to be available..."
MAX_ATTEMPTS=30
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  DISK_STATE=$(aws lightsail get-disk --disk-name "$DISK_NAME" --region
 "$AWS_REGION" --query 'disk.state' --output text 2>/dev/null || echo "unknown")
  if [ "$DISK_STATE" == "available" ]; then
    echo "Disk is now available"
    break
  fi
  echo "Disk status: $DISK_STATE. Waiting..."
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  handle_error "Disk failed to become available after 5 minutes"
fi

# Attach the disk to the instance
echo "Attaching disk to instance"
aws lightsail attach-disk \
  --disk-name "$DISK_NAME" \
  --instance-name "$INSTANCE_NAME" \
  --disk-path /dev/xvdf \
  --region "$AWS_REGION"
check_status "Failed to attach disk to instance"

echo "Disk attached. To format and mount the disk, connect to your instance and
 run:"
echo "sudo mkfs -t ext4 /dev/xvdf"
echo "sudo mkdir -p /mnt/my-data"
echo "sudo mount /dev/xvdf /mnt/my-data"
echo "sudo chown ec2-user:ec2-user /mnt/my-data"

# Step 6: Create a snapshot of the instance
echo "Step 6: Creating snapshot of the instance: $SNAPSHOT_NAME"
aws lightsail create-instance-snapshot \
  --instance-name "$INSTANCE_NAME" \
  --instance-snapshot-name "$SNAPSHOT_NAME" \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create instance snapshot"
track_resource "instance_snapshot" "$SNAPSHOT_NAME"
```

```
# Wait for the snapshot to complete using polling
echo "Waiting for snapshot to complete... (this may take several minutes)"
MAX_ATTEMPTS=60
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  SNAPSHOT_STATE=$(aws lightsail get-instance-snapshot --instance-snapshot-name
 "$SNAPSHOT_NAME" --region "$AWS_REGION" --query 'instanceSnapshot.state' --
output text 2>/dev/null || echo "unknown")
  if [ "$SNAPSHOT_STATE" == "available" ]; then
    echo "Snapshot creation completed"
    break
  fi
  echo "Snapshot status: $SNAPSHOT_STATE. Waiting... ($ATTEMPTS/$MAX_ATTEMPTS)"
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  echo "Warning: Snapshot creation is taking longer than expected but will
 continue in the background."
  echo "You can check its status later with: aws lightsail get-instance-snapshot
 --instance-snapshot-name $SNAPSHOT_NAME --region $AWS_REGION"
fi

# Step 7: Clean up resources
echo "Step 7: Clean up resources"
echo "The script has created the following resources:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
  echo "  $resource"
done

echo "Cleaning up these resources automatically..."
cleanup_resources

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachDisk
  - CreateDisk
  - CreateInstanceSnapshot
  - CreateInstances

- DeleteDisk
- DeleteInstance
- DeleteInstanceSnapshot
- DetachDisk
- DownloadDefaultKeyPair
- GetBlueprints
- GetBundles
- GetDisk
- GetInstance
- GetInstanceSnapshot
- GetInstanceState

Exemplos de código para Amazon Location usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Location Service com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Amazon Location: mais informações sobre o Amazon Location.
- Referência de API do Amazon Location: detalhes sobre todas as ações disponíveis do Amazon Location.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de localização na Amazon usando AWS SDKs
  - Olá, Amazon Location Service
  - Aprenda o básico sobre a localização na Amazon com um AWS SDK
  - Ações para Amazon Location usando AWS SDKs
    - Use BatchUpdateDevicePositioncom um AWS SDK
    - Use CalculateRoutecom um AWS SDK
    - Use CreateGeofenceCollectioncom um AWS SDK
    - Use CreateKeycom um AWS SDK
    - Use CreateMapcom um AWS SDK
    - Use CreateRouteCalculatorcom um AWS SDK
    - Use CreateTrackercom um AWS SDK
    - Use DeleteGeofenceCollectioncom um AWS SDK
    - Use DeleteKeycom um AWS SDK
    - Use DeleteMapcom um AWS SDK
    - Use DeleteRouteCalculatorcom um AWS SDK
    - Use DeleteTrackercom um AWS SDK
    - Use GetDevicePositioncom um AWS SDK
    - Use PutGeofencecom um AWS SDK

Exemplos básicos de localização na Amazon usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Location Service com AWS SDKs.

Exemplos

- Olá, Amazon Location Service
- Aprenda o básico sobre a localização na Amazon com um AWS SDK
- Ações para Amazon Location usando AWS SDKs
  - Use BatchUpdateDevicePositioncom um AWS SDK
  - Use CalculateRoutecom um AWS SDK
  - Use CreateGeofenceCollectioncom um AWS SDK
  - Use CreateKeycom um AWS SDK

- Use CreateMapcom um AWS SDK
- Use CreateRouteCalculatorcom um AWS SDK
- Use CreateTrackercom um AWS SDK
- Use DeleteGeofenceCollectioncom um AWS SDK
- Use DeleteKeycom um AWS SDK
- Use DeleteMapcom um AWS SDK
- Use DeleteRouteCalculatorcom um AWS SDK
- Use DeleteTrackercom um AWS SDK
- Use GetDevicePositioncom um AWS SDK
- Use PutGeofencecom um AWS SDK

Olá, Amazon Location Service

Os exemplos de código a seguir mostram como começar a usar o Amazon Location Service.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, you need to create a collection using the AWS Management
 * console. For information, see the following documentation.
 *
```

```
 * https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
public class HelloLocation {

    private static LocationAsyncClient locationAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(HelloLocation.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance.
    private static LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

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

            Usage:
                <collectionName>

            Where:
                collectionName - The Amazon location collection name.
            """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        listGeofences(collectionName);
    }

    /**
     * Lists geofences from a specified geofence collection asynchronously.
     *
     * @param collectionName The name of the geofence collection to list
 geofences from.
     * @return A {@link CompletableFuture} representing the result of the
 asynchronous operation.
     *         The future completes when all geofences have been processed and
 logged.
     */
    public static CompletableFuture<Void> listGeofences(String collectionName) {
        ListGeofencesRequest geofencesRequest = ListGeofencesRequest.builder()
                .collectionName(collectionName)
                .build();

        ListGeofencesPublisher paginator =
 getClient().listGeofencesPaginator(geofencesRequest);
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            if (response.entries().isEmpty()) {
                logger.info("No Geofences were found in the collection.");
            } else {
                response.entries().forEach(geofence ->
                        logger.info("Geofence ID: " + geofence.geofenceId())
                );
            }
        });
        return future;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .

- ListGeofenceCollections
- ListGeofences

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  LocationClient,
  ListGeofenceCollectionsCommand,
} from "@aws-sdk/client-location";

/**
 * Lists geofences from a specified geofence collection asynchronously.
 */
export const main = async () => {
  const region = "eu-west-1";
  const locationClient = new LocationClient({ region: region });
  const listGeofenceCollParams = {
    MaxResults: 100,
  };
  try {
    const command = new ListGeofenceCollectionsCommand(listGeofenceCollParams);
    const response = await locationClient.send(command);
    const geofenceEntries = response.Entries;
    if (geofenceEntries.length === 0) {
      console.log("No Geofences were found in the collection.");
    } else {
      for (const geofenceEntry of geofenceEntries) {
        console.log(`Geofence ID: ${geofenceEntry.CollectionName}`);
      }
    }
  } catch (error) {
    console.error(
```

```
      `A validation error occurred while creating geofence: ${error} \n Exiting
 program.`,
    );
    return;
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - ListGeofenceCollections
  - ListGeofences

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you need to create a collection using the AWS Management
console. For information, see the following documentation.

https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
suspend fun main(args: Array<String>) {
    val usage = """
```

```
        Usage:
            <colletionName>

        Where:
            colletionName - The Amazon location collection name.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }
    val colletionName = args[0]
    listGeofences(colletionName)
}

/**
 * Lists the geofences for the specified collection name.
 *
 * @param collectionName the name of the geofence collection
 */
suspend fun listGeofences(collectionName: String) {
    val request = ListGeofencesRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.listGeofences(request)
        val geofences = response.entries
        if (geofences.isNullOrEmpty()) {
            println("No Geofences found")
        } else {
            geofences.forEach { geofence ->
                println("Geofence ID: ${geofence.geofenceId}")
            }
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - ListGeofenceCollections

- ListGeofences

Aprenda o básico sobre a localização na Amazon com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar um mapa do Amazon Location.
- Criar uma chave de API do Amazon Location.
- Exibir o URL do mapa.
- Criar uma coleção de geocercas
- Armazenar uma geometria de geocerca.
- Criar um recurso de rastreador.
- Atualizar a posição de um dispositivo.
- Recuperar a atualização de posição mais recente de um dispositivo específico.
- Criar uma calculadora de rotas.
- Determinar a distância entre Seattle e Vancouver.
- Usar as APIs de nível superior do Amazon Location.
- Excluir os ativos do Amazon Location.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon Location Service.

```
/*
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */
public class LocationScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    private static final Logger logger =
 LoggerFactory.getLogger(LocationScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static LocationActions locationActions = new LocationActions();

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

            Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

            Where:
              mapName - The name of the map to be create (e.g., "AWSMap").
              keyName - The name of the API key to create (e.g., "AWSApiKey").
              collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
              geoId - The geographic identifier used for the geofence or map
 (e.g., "geoId").
              trackerName - The name of the tracker (e.g., "geoTracker").
              calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
              deviceId - The ID of the device (e.g., "iPhone-112356").
            """;

        if (args.length != 7) {
            logger.info(usage);
            return;
        }

        String mapName = args[0];
        String keyName = args[1];
        String collectionName = args[2];
        String geoId = args[3];
```

```
        String trackerName = args[4];
        String calculatorName = args[5];
        String deviceId = args[6];

        logger.info("""
            AWS Location Service is a fully managed service offered by Amazon Web
 Services (AWS) that
            provides location-based services for developers. This service
 simplifies
            the integration of location-based features into applications, making
 it
            easier to build and deploy location-aware applications.

            The AWS Location Service offers a range of location-based services,
            including:

            Maps: The service provides access to high-quality maps, satellite
 imagery,\s
            and geospatial data from various providers, allowing developers to\s
            easily embed maps into their applications.

            Tracking: The Location Service enables real-time tracking of mobile
 devices,\s
            assets, or other entities, allowing developers to build applications
\s
            that can monitor the location of people, vehicles, or other objects.

            Geocoding: The service provides the ability to convert addresses or\s
            location names into geographic coordinates (latitude and longitude),
\s
            and vice versa, enabling developers to integrate location-based
 search\s
            and routing functionality into their applications.
            """);
        waitForInputToContinue(scanner);
        try {
            runScenario(mapName, keyName, collectionName, geoId, trackerName,
 calculatorName, deviceId);
        } catch (RuntimeException e) {
            // Clean up AWS Resources.
            cleanUp(mapName, keyName, collectionName, trackerName,
 calculatorName);
            logger.info(e.getMessage());
        }
```

```
    }

    public static void runScenario(String mapName, String keyName, String
collectionName, String geoId, String trackerName, String calculatorName, String
deviceId) {
        logger.info(DASHES);
        logger.info("1. Create a map");
        logger.info("""
             An AWS Location map can enhance the user experience of your
             application by providing accurate and personalized location-based
             features. For example, you could use the geocoding capabilities to
             allow users to search for and locate businesses, landmarks, or
             other points of interest within a specific region.
            """);

        waitForInputToContinue(scanner);
        String mapArn;
        try {
            mapArn = locationActions.createMap(mapName).join();
            logger.info("The Map ARN is: {}", mapArn);  // Log success in calling
code
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request exceeded the maximum quota: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the
map.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Location API key");
        logger.info("""
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.
            """);
```

```
        try {
            String keyArn = locationActions.createKey(keyName, mapArn).join();
            logger.info("The API key was successfully created: {}", keyArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof AccessDeniedException) {
                logger.error("Request was denied: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the API
 key.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Display Map URL");
        logger.info("""
            In order to get the MAP URL, you need to get the API Key value.
            You can get the key value using the AWS Management Console under
            Location Services. This operation cannot be completed using the
            AWS SDK. For more information about getting the key value, see
            the AWS Location Documentation.
            """);
        String mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/"+mapName
+"/tiles/{z}/{x}/{y}?key={KeyValue}";
        logger.info("Embed this URL in your Web app: " + mapUrl);
        logger.info("");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a geofence collection, which manages and stores
 geofences.");
        waitForInputToContinue(scanner);
        try {
            String collectionArn =
 locationActions.createGeofenceCollection(collectionName).join();
            logger.info("The geofence collection was successfully created: {}",
 collectionArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
```

```
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the
geofence collection.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Store a geofence geometry in a given geofence
collection.");
        logger.info("""
            An AWS Location geofence is a virtual boundary that defines a
geographic area
            on a map. It is a useful feature for tracking the location of
            assets or monitoring the movement of objects within a specific
region.

            To define a geofence, you need to specify the coordinates of a
            polygon that represents the area of interest. The polygon must be
            defined in a counter-clockwise direction, meaning that the points of
            the polygon must be listed in a counter-clockwise order.

            This is a requirement for the AWS Location service to correctly
            interpret the geofence and ensure that the location data is
            accurately processed within the defined area.
            """);

        waitForInputToContinue(scanner);
        try {
            locationActions.putGeofence(collectionName, geoId).join();
            logger.info("Successfully created geofence: {}", geoId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("A validation error occurred while creating
geofence: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
```

```
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Create a tracker resource which lets you retrieve current
and historical location of devices..");
        waitForInputToContinue(scanner);
        try {
            String trackerArn =
locationActions.createTracker(trackerName).join();
            logger.info("Successfully created tracker. ARN: {}", trackerArn);  //
Log success
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred while creating the tracker:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Update the position of a device in the location tracking
system.");
        logger.info("""
            The AWS location service does not enforce a strict format for
deviceId, but it must:
              - Be a string (case-sensitive).
              - Be 1-100 characters long.
              - Contain only:
                - Alphanumeric characters (A-Z, a-z, 0-9)
                - Underscores (_)
                - Hyphens (-)
                - Be the same ID used when sending and retrieving positions.
            """);

        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<BatchUpdateDevicePositionResponse> future =
locationActions.updateDevicePosition(trackerName, deviceId);
            future.join();
            logger.info(deviceId + " was successfully updated in the location
tracking system.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. Retrieve the most recent position update for a specified
device..");
        waitForInputToContinue(scanner);
        try {
            GetDevicePositionResponse response =
locationActions.getDevicePosition(trackerName, deviceId).join();
            logger.info("Successfully fetched device position: {}",
response.position());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Create a route calculator.");
```

```
        waitForInputToContinue(scanner);
        try {
            CreateRouteCalculatorResponse response =
locationActions.createRouteCalculator(calculatorName).join();
            logger.info("Route calculator created successfully: {}",
response.calculatorArn());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.info("A conflict occurred: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("10. Determine the distance between Seattle and Vancouver
using the route calculator.");
        waitForInputToContinue(scanner);
        try {
            CalculateRouteResponse response =
locationActions.calcDistanceAsync(calculatorName).join();
            logger.info("Successfully calculated route. The distance in
kilometers is {}", response.summary().distance());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("11. Use the GeoPlacesAsyncClient to perform additional
operations.");
```

```
        logger.info("""
            This scenario will show use of the GeoPlacesClient that enables
            location search and geocoding capabilities for your applications.\s

            We are going to use this client to perform these AWS Location tasks:
             - Reverse Geocoding (reverseGeocode): Converts geographic
coordinates into addresses.
             - Place Search (searchText): Finds places based on search queries.
             - Nearby Search (searchNearby): Finds places near a specific
location.
            """);

        logger.info("First we will perform a Reverse Geocoding operation");
        waitForInputToContinue(scanner);
        try {
            locationActions.reverseGeocode().join();
            logger.info("Now we are going to perform a text search using coffee
shop.");
            waitForInputToContinue(scanner);
            locationActions.searchText("coffee shop").join();
            waitForInputToContinue(scanner);

            logger.info("Now we are going to perform a nearby Search.");
            //waitForInputToContinue(scanner);
            locationActions.searchNearBy().join();
            waitForInputToContinue(scanner);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                logger.error("A validation error occurred: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info("12. Delete the AWS Location Services resources.");
        logger.info("Would you like to delete the AWS Location Services
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
```

```
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp(mapName, keyName, collectionName, trackerName,
calculatorName);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(" This concludes the AWS Location Service scenario.");
        logger.info(DASHES);
    }

    /**
     * Cleans up resources by deleting the specified map, key, geofence
collection, tracker, and route calculator.
     *
     * @param mapName The name of the map to delete.
     * @param keyName The name of the key to delete.
     * @param collectionName The name of the geofence collection to delete.
     * @param trackerName The name of the tracker to delete.
     * @param calculatorName The name of the route calculator to delete.
     */
    private static void cleanUp(String mapName, String keyName, String
collectionName, String trackerName, String calculatorName) {
        try {
            locationActions.deleteMap(mapName).join();
            locationActions.deleteKey(keyName).join();
            locationActions.deleteGeofenceCollectionAsync(collectionName).join();
            locationActions.deleteTracker(trackerName).join();
            locationActions.deleteRouteCalculator(calculatorName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
    }
```

```
    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper para os métodos de SDK do Amazon Location Service.

```
public class LocationActions {

    private static LocationAsyncClient locationAsyncClient;

    private static GeoPlacesAsyncClient geoPlacesAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(LocationActions.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance is used throughout the application.
    private LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
```

```
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    private static GeoPlacesAsyncClient getGeoPlacesClient() {
        if (geoPlacesAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            geoPlacesAsyncClient = GeoPlacesAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return geoPlacesAsyncClient;
    }


    /**
     * Performs a nearby places search based on the provided geographic
 coordinates (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes
 successfully.
```

```
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error
occurred: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place
search", exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }


    /**
     * Searches for a place using the provided search query and prints the
detailed information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
```

```
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType ->
 {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
```

```
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error
occurred: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place
search", exception);
                });
    }



    /**
     * Performs reverse geocoding using the AWS Geo Places API.
     * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
     * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
     */
    public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        logger.info("Use latitude 37.7749 and longitude -122.4194");

        // AWS expects [longitude, latitude].
        List<Double> queryPosition = List.of(longitude, latitude);
        ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
            .queryPosition(queryPosition)
            .build();
        CompletableFuture<ReverseGeocodeResponse> futureResponse =
            getGeoPlacesClient().reverseGeocode(request);

        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: "
+ cause.getMessage(), cause);
                }
```

```
                throw new CompletionException("Error performing reverse
geocoding", exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }


    /**
     * Calculates the distance between two locations asynchronously.
     *
     * @param routeCalcName the name of the route calculator to use
     * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the
route
     */
    public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
        // Define coordinates for Seattle, WA and Vancouver, BC.
        List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
        List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

        CalculateRouteRequest request = CalculateRouteRequest.builder()
            .calculatorName(routeCalcName)
            .departurePosition(departurePosition)
            .destinationPosition(arrivePosition)
            .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
            .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
            .build();

        return getClient().calculateRoute(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
                }
```

```
            });
    }


    /**
     * Creates a new route calculator with the specified name and data source.
     *
     * @param routeCalcName the name of the route calculator to be created
     */
    public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
        String dataSource = "Esri"; // or "Here"
        CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
            .calculatorName(routeCalcName)
            .dataSource(dataSource)
            .build();

        return getClient().createRouteCalculator(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflict error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device
position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
```

```
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device
position: " + exception.getMessage(), exception);
                }
            });
    }


    /**
     * Updates the position of a device in the location tracking system.
     *
     * @param trackerName the name of the tracker associated with the device
     * @param deviceId    the unique identifier of the device
     * @throws RuntimeException if an error occurs while updating the device
position
     */
    public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
        double latitude = 37.7749;  // Example: San Francisco
        double longitude = -122.4194;

        DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
            .deviceId(deviceId)
            .sampleTime(Instant.now()) // Timestamp of position update.
            .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
            .build();

        BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
            .trackerName(trackerName)
            .updates(positionUpdate)
            .build();
```

```
        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: "
+ cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device
position: " + exception.getMessage(), exception);
                }
            }
        });
    }


    /**
     * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
     *
     * @param trackerName the name of the tracker to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
     */
    public CompletableFuture<String> createTracker(String trackerName) {
        CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
            .description("Created using the Java V2 SDK")
            .trackerName(trackerName)
            .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
            .build();

        return getClient().createTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
```

```
                }
            })
            .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
    }



    /**
     * Adds a new geofence to the specified collection.
     *
     * @param collectionName the name of the geofence collection to add the
geofence to
     * @param geoId          the unique identifier for the geofence
     */
    public CompletableFuture<PutGeofenceResponse> putGeofence(String
collectionName, String geoId) {
        // Define the geofence geometry (polygon).
        GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
            .polygon(List.of(
                List.of(
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();

        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
```

```
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new geofence collection.
     *
     * @param collectionName the name of the geofence collection to be created
     */
    public CompletableFuture<String> createGeofenceCollection(String
collectionName) {
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection
was not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the
ARN
    }




    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
```

```
    * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
    * @return a {@link CompletableFuture} that completes with the Amazon
Resource Name (ARN) of the created API key,
    * or {@code null} if the operation failed
    */
   public CompletableFuture<String> createKey(String keyName, String mapArn) {
       ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
           .allowActions("geo:GetMap*")
           .allowResources(mapArn)
           .build();

       CreateKeyRequest request = CreateKeyRequest.builder()
           .keyName(keyName)
           .restrictions(keyRestrictions)
           .noExpiry(true)
           .build();

       return getClient().createKey(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof AccessDeniedException) {
                       throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                   }
                   throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
   }



   /**
    * Creates a new map with the specified name and configuration.
    *
    * @param mapName the name of the map to be created
    * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
    * @throws CompletionException if an error occurs while creating the map,
such as exceeding the service quota
```

```
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }



    /**
     * Deletes a geofence collection asynchronously.
     *
     * @param collectionName the name of the geofence collection to be deleted
     * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
     */
    public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
        DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .build();

        return getClient().deleteGeofenceCollection(collectionRequest)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested geofence
collection was not found.", cause);
                    }
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }



    /**
     * Deletes the specified key from the key-value store.
     *
     * @param keyName the name of the key to be deleted
     * @return a {@link CompletableFuture} that completes when the key has been
deleted
     * @throws CompletionException if the key was not found or if an error
occurred during the deletion process
     */
    public CompletableFuture<Void> deleteKey(String keyName) {
        DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
            .keyName(keyName)
            .build();

        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
```

```
            })
            .thenApply(response -> null);
    }


    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion
is successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();

        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }


    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has
been deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
```

```
    *                               - if any other error occurs, a generic
CompletionException is thrown with the error message
    */
   public CompletableFuture<Void> deleteTracker(String trackerName) {
       DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
           .trackerName(trackerName)
           .build();

       return getClient().deleteTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The tracker was not
found.", cause);
                   }
                   throw new CompletionException("Failed to delete the tracker:
" + exception.getMessage(), exception);
               }
               logger.info("The tracker {} was deleted.", trackerName);
           })
           .thenApply(response -> null); // Ensures CompletableFuture<Void>
   }


   /**
    * Deletes a route calculator from the system.
    *
    * @param calcName the name of the route calculator to delete
    * @return a {@link CompletableFuture} that completes when the route
calculator has been deleted
    * @throws CompletionException if an error occurs while deleting the route
calculator
    *                               - If the route calculator was not found, a
{@link ResourceNotFoundException} will be thrown
    *                               - If any other error occurs, a generic {@link
CompletionException} will be thrown
    */
   public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
       DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
           .calculatorName(calcName)
           .build();
```

```
        return getClient().deleteRouteCalculator(calculatorRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was
 not found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
 calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/*
Before running this JavaScript code example, set up your development environment,
 including your credentials.
This demo illustrates how to use the AWS SDK for JavaScript (v3) to work with
 Amazon Location Service.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/getting-
started.html
*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

import {
  CreateMapCommand,
  CreateGeofenceCollectionCommand,
  PutGeofenceCommand,
  CreateTrackerCommand,
  BatchUpdateDevicePositionCommand,
  GetDevicePositionCommand,
  CreateRouteCalculatorCommand,
  CalculateRouteCommand,
  LocationClient,
  ConflictException,
  ResourceNotFoundException,
```

```
  DeleteGeofenceCollectionCommand,
  DeleteRouteCalculatorCommand,
  DeleteTrackerCommand,
  DeleteMapCommand,
} from "@aws-sdk/client-location";

import {
  GeoPlacesClient,
  ReverseGeocodeCommand,
  SearchNearbyCommand,
  SearchTextCommand,
  GetPlaceCommand,
  ValidationException,
} from "@aws-sdk/client-geo-places";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";

/*The inputs for this example can be edited in the ./input.json.*/
import data from "./inputs.json" with { type: "json" };

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
/* v8 ignore next 3 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
  verbose: "false",
});

const pressEnterConfirm = new ScenarioInput(
  "confirm",
  "Press Enter to continue",
  {
    type: "confirm",
    verbose: "false",
  },
);

const region = "eu-west-1";

const locationClient = new LocationClient({ region: region });
```

```
const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Location Use demo! \n" +
    "AWS Location Service is a fully managed service offered by Amazon Web
 Services (AWS) that " +
    "provides location-based services for developers. This service simplifies " +
    "the integration of location-based features into applications, making it " +
    "Maps: The service provides access to high-quality maps, satellite imagery, "
 +
    "and geospatial data from various providers, allowing developers to " +
    "easily embed maps into their applications:\n" +
    "Tracking: The Location Service enables real-time tracking of mobile devices,
 " +
    "assets, or other entities, allowing developers to build applications " +
    "that can monitor the location of people, vehicles, or other objects.\n" +
    "Geocoding: The service provides the ability to convert addresses or " +
    "location names into geographic coordinates (latitude and longitude), " +
    "and vice versa, enabling developers to integrate location-based search " +
    "and routing functionality into their applications. " +
    "Please define values ./inputs.json for each user-defined variable used in
 this app. Otherwise the default is used:\n" +
    "- mapName: The name of the map to be create (default is 'AWSMap').\n" +
    "- keyName: The name of the API key to create (default is ' AWSApiKey')\n" +
    "- collectionName: The name of the geofence collection (default is
 'AWSLocationCollection')\n" +
    "- geoId: The geographic identifier used for the geofence or map (default is
 'geoId')\n" +
    "- trackerName: The name of the tracker (default is 'geoTracker')\n" +
    "- calculatorName: The name of the route calculator (default is
 'AWSRouteCalc')\n" +
    "- deviceId: The ID of the device (default is 'iPhone-112356')",

  { header: true },
);
const displayCreateAMap = new ScenarioOutput(
  "displayCreateAMap",
  "1. Create a map\n" +
    "An AWS Location map can enhance the user experience of your " +
    " application by providing accurate and personalized location-based " +
    " features. For example, you could use the geocoding capabilities to " +
    " allow users to search for and locate businesses, landmarks, or " +
    " other points of interest within a specific region.",
);
```

```
const sdkCreateAMap = new ScenarioAction(
  "sdkCreateAMap",
  async (/** @type {State} */ state) => {
    const createMapParams = {
      MapName: `${data.inputs.mapName}`,
      Configuration: { style: "VectorEsriNavigation" },
    };
    try {
      const command = new CreateMapCommand(createMapParams);
      const response = await locationClient.send(command);
      state.MapName = response.MapName;
      console.log("Map created. Map ARN is: ", state.MapName);
    } catch (error) {
      console.error("Error creating map: ", error);
      throw error;
    }
  },
);

const displayMapUrl = new ScenarioOutput(
  "displayMapUrl",
  "2. Display Map URL\n" +
    "When you embed a map in a web app or website, the API key is " +
    "included in the map tile URL to authenticate requests. You can " +
    "restrict API keys to specific AWS Location operations (e.g., only " +
    "maps, not geocoding). API keys can expire, ensuring temporary " +
    "access control.\n" +
    "In order to get the MAP URL you need to create and get the API Key value. "
 +
    "You can create and get the key value using the AWS Management Console under
 " +
    "Location Services. These operations cannot be completed using the " +
    "AWS SDK. For more information about getting the key value, see " +
    "the AWS Location Documentation.",
);

const sdkDisplayMapUrl = new ScenarioAction(
  "sdkDisplayMapUrl",
  async (/** @type {State} */ state) => {
    const mapURL = `https://maps.geo.aws.amazon.com/maps/v0/maps/
${state.MapName}/tiles/{z}/{x}/{y}?key=API_KEY_VALUE`;
    state.mapURL = mapURL;
    console.log(
```

```
      `Replace \'API_KEY_VALUE\' in the following URL with the value for the API
 key you create and get from the AWS Management Console under Location Services.
 This is then the Map URL you can embed this URL in your Web app:\n
${state.mapURL}`,
    );
  },
);
const displayCreateGeoFenceColl = new ScenarioOutput(
  "displayCreateGeoFenceColl",
  "3. Create a geofence collection, which manages and stores geofences.",
);

const sdkCreateGeoFenceColl = new ScenarioAction(
  "sdkCreateGeoFenceColl",
  async (/** @type {State} */ state) => {
    // Creates a new geofence collection.
    const geoFenceCollParams = {
      CollectionName: `${data.inputs.collectionName}`,
    };
    try {
      const command = new CreateGeofenceCollectionCommand(geoFenceCollParams);
      const response = await locationClient.send(command);
      state.CollectionName = response.CollectionName;
      console.log(
        `The geofence collection was successfully created:
 ${state.CollectionName}`,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An unexpected error occurred while creating the geofence collection:
 ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayStoreGeometry = new ScenarioOutput(
  "displayStoreGeometry",
  "4. Store a geofence geometry in a given geofence collection. " +
    "An AWS Location geofence is a virtual boundary that defines a geographic
 area " +
    "on a map. It is a useful feature for tracking the location of " +
```

```
    "assets or monitoring the movement of objects within a specific region. " +
    "To define a geofence, you need to specify the coordinates of a " +
    "polygon that represents the area of interest. The polygon must be " +
    "defined in a counter-clockwise direction, meaning that the points of " +
    "the polygon must be listed in a counter-clockwise order. " +
    "This is a requirement for the AWS Location service to correctly " +
    "interpret the geofence and ensure that the location data is " +
    "accurately processed within the defined area.",
);

const sdkStoreGeometry = new ScenarioAction(
  "sdkStoreGeometry",
  async (/** @type {State} */ state) => {
    const geoFenceGeoParams = {
      CollectionName: `${data.inputs.collectionName}`,
      GeofenceId: `${data.inputs.geoId}`,
      Geometry: {
        Polygon: [
          [
            [-122.3381, 47.6101],
            [-122.3281, 47.6101],
            [-122.3281, 47.6201],
            [-122.3381, 47.6201],
            [-122.3381, 47.6101],
          ],
        ],
      },
    };
    try {
      const command = new PutGeofenceCommand(geoFenceGeoParams);
      const response = await locationClient.send(command);
      state.GeoFencId = response.GeofenceId;
      console.log("GeoFence created. GeoFence ID is: ", state.GeoFencId);
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `A validation error occurred while creating geofence: ${caught.message}
 \n Exiting program.`,
        );
        return;
      }
    }
  },
);
```

```
const displayCreateTracker = new ScenarioOutput(
  "displayCreateTracker",
  "5. Create a tracker resource which lets you retrieve current and historical
 location of devices.",
);

const sdkCreateTracker = new ScenarioAction(
  "sdkCreateTracker",
  async (/** @type {State} */ state) => {
    //Creates a new tracker resource in your AWS account, which you can use to
 track the location of devices.
    const createTrackerParams = {
      TrackerName: `${data.inputs.trackerName}`,
      Description: "Created using the JavaScript V3 SDK",
      PositionFiltering: "TimeBased",
    };
    try {
      const command = new CreateTrackerCommand(createTrackerParams);
      const response = await locationClient.send(command);
      state.trackerName = response.TrackerName;
      console.log("Tracker created. Tracker name is : ", state.trackerName);
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `A validation error occurred while creating geofence: ${caught.message}
 \n Exiting program.`,
        );
      } else {
        `An unexpected error error occurred: ${caught.message} \n Exiting
 program.`;
      }
      return;
    }
  },
);
const displayUpdatePosition = new ScenarioOutput(
  "displayUpdatePosition",
  "6. Update the position of a device in the location tracking system." +
    "The AWS Location Service does not enforce a strict format for deviceId, but
 it must:\n " +
    "- Be a string (case-sensitive).\n" +
    "- Be 1-100 characters long.\n" +
    "- Contain only: Alphanumeric characters (A-Z, a-z, 0-9); Underscores (_);
 Hyphens (-); and be the same ID used when sending and retrieving positions.",
```

```
);

const sdkUpdatePosition = new ScenarioAction(
  "sdkUpdatePosition",
  async (/** @type {State} */ state) => {
    // Updates the position of a device in the location tracking system.

    const updateDevicePosParams = {
      TrackerName: `${data.inputs.trackerName}`,
      Updates: [
        {
          DeviceId: `${data.inputs.deviceId}`,
          SampleTime: new Date(),
          Position: [-122.4194, 37.7749],
        },
      ],
    };
    try {
      const command = new BatchUpdateDevicePositionCommand(
        updateDevicePosParams,
      );
      const response = await locationClient.send(command);
      console.log(
        `Device with id ${data.inputs.deviceId} was successfully updated in the
 location tracking system. `,
      );
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `A validation error occurred while updating the device:
 ${caught.message} \n Exiting program.`,
        );
      }
    }
  },
);
const displayRetrievePosition = new ScenarioOutput(
  "displayRetrievePosition",
  "7. Retrieve the most recent position update for a specified device.",
);

const sdkRetrievePosition = new ScenarioAction(
  "sdkRetrievePosition",
  async (/** @type {State} */ state) => {
```

```
    const devicePositionParams = {
      TrackerName: `${data.inputs.trackerName}`,
      DeviceId: `${data.inputs.deviceId}`,
    };
    try {
      const command = new GetDevicePositionCommand(devicePositionParams);
      const response = await locationClient.send(command);
      state.position = response.Position;
      console.log("Successfully fetched device position: : ", state.position);
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `"The resource was not found: ${caught.message} \n Exiting program.`,
        );
      } else {
        `An unexpected error error occurred: ${caught.message} \n Exiting
 program.`;
      }
      return;
    }
  },
);
const displayCreateRouteCalc = new ScenarioOutput(
  "displayCreateRouteCalc",
  "8. Create a route calculator.",
);

const sdkCreateRouteCalc = new ScenarioAction(
  "sdkCreateRouteCalc",
  async (/** @type {State} */ state) => {
    const routeCalcParams = {
      CalculatorName: `${data.inputs.calculatorName}`,
      DataSource: "Esri",
    };
    try {
      // Creates a new route calculator with the specified name and data source.
      const command = new CreateRouteCalculatorCommand(routeCalcParams);
      const response = await locationClient.send(command);
      state.CalculatorName = response.CalculatorName;
      console.log(
        "Route calculator created successfully. Calculator name is: ",
        state.CalculatorName,
      );
    } catch (caught) {
```

```
      if (caught instanceof ConflictException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayDetermineDist = new ScenarioOutput(
  "displayDetermineDist",
  "9. Determine the distance between Seattle and Vancouver using the route
 calculator.",
);

const sdkDetermineDist = new ScenarioAction(
  "sdkDetermineDist",
  async (/** @type {State} */ state) => {
    // Calculates the distance between two locations asynchronously.
    const determineDist = {
      CalculatorName: `${data.inputs.calculatorName}`,
      DeparturePosition: [-122.3321, 47.6062],
      DestinationPosition: [-123.1216, 49.2827],
      TravelMode: "Car",
      DistanceUnit: "Kilometers",
    };
    try {
      const command = new CalculateRouteCommand(determineDist);
      const response = await locationClient.send(command);

      console.log(
        "Successfully calculated route. The distance in kilometers is : ",
        response.Summary.Distance,
      );
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `Failed to calculate route: ${caught.message} \n Exiting program.`,
        );
      }
      return;
    }
  },
);
```

```
const displayUseGeoPlacesClient = new ScenarioOutput(
  "displayUseGeoPlacesClient",
  "10. Use the GeoPlacesAsyncClient to perform additional operations. " +
    "This scenario will show use of the GeoPlacesClient that enables" +
    "location search and geocoding capabilities for your applications. " +
    "We are going to use this client to perform these AWS Location tasks: \n" +
    " - Reverse Geocoding (reverseGeocode): Converts geographic coordinates into
 addresses.\n " +
    " - Place Search (searchText): Finds places based on search queries.\n " +
    " - Nearby Search (searchNearby): Finds places near a specific location.\n "
 +
    "First we will perform a Reverse Geocoding operation",
);

const sdkUseGeoPlacesClient = new ScenarioAction(
  "sdkUseGeoPlacesClient",
  async (/** @type {State} */ state) => {
    const geoPlacesClient = new GeoPlacesClient({ region: region });

    const reverseGeoCodeParams = {
      QueryPosition: [-122.4194, 37.7749],
    };
    const searchTextParams = {
      QueryText: "coffee shop",
      BiasPosition: [-122.4194, 37.7749], //San Fransisco
    };
    const searchNearbyParams = {
      QueryPosition: [-122.4194, 37.7749],
      QueryRadius: Number("1000"),
    };
    try {
      /*   Performs reverse geocoding using the AWS Geo Places API.
     Reverse geocoding is the process of converting geographic coordinates
 (latitude and longitude) to a human-readable address.
     This method uses the latitude and longitude of San Francisco as the input,
 and prints the resulting address.*/

      console.log("Use latitude 37.7749 and longitude -122.4194.");
      const command = new ReverseGeocodeCommand(reverseGeoCodeParams);
      const response = await geoPlacesClient.send(command);
      console.log(
        "Successfully calculated route. The distance in kilometers is : ",
        response.ResultItems[0].Distance,
      );
```

```
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
    try {
      console.log(
        "Now we are going to perform a text search using coffee shop",
      );

      /*Searches for a place using the provided search query and prints the
detailed information of the first result.
 @param searchTextParams the search query to be used for the place search (ex,
coffee shop)*/

      const command = new SearchTextCommand(searchTextParams);
      const response = await geoPlacesClient.send(command);
      const placeId = response.ResultItems[0].PlaceId.toString();
      const getPlaceCommand = new GetPlaceCommand({
        PlaceId: placeId,
      });
      const getPlaceResponse = await geoPlacesClient.send(getPlaceCommand);
      console.log(
        `Detailed Place Information: \n Name and address:
${getPlaceResponse.Address.Label}`,
      );

      const foodTypes = getPlaceResponse.FoodTypes;
      if (foodTypes.length) {
        console.log("Food Types: ");
        for (const foodType of foodTypes) {
          console.log("- ", foodType.LocalizedName);
        }
      } else {
        console.log("No food types available.");
      }
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
```

```
        return;
      }
    }
    try {
      console.log("\nNow we are going to perform a nearby search.");
      const command = new SearchNearbyCommand(searchNearbyParams);
      const response = await geoPlacesClient.send(command);
      const resultItems = response.ResultItems;
      console.log("\nSuccessfully performed nearby search.");
      for (const resultItem of resultItems) {
        console.log("Name and address: ", resultItem.Address.Label);
        console.log("Distance: ", resultItem.Distance);
      }
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);

const displayDeleteResources = new ScenarioOutput(
  "displayDeleteResources",
  "11. Delete the AWS Location Services resources. " +
    "Would you like to delete the AWS Location Services resources? (y/n)",
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    const deleteGeofenceCollParams = {
      CollectionName: `${state.CollectionName}`,
    };
    const deleteRouteCalculatorParams = {
      CalculatorName: `${state.CalculatorName}`,
    };
    const deleteTrackerParams = { TrackerName: `${state.trackerName}` };
    const deleteMapParams = { MapName: `${state.MapName}` };
    try {
      const command = new DeleteMapCommand(deleteMapParams);
      const response = await locationClient.send(command);
```

```
      console.log("Map deleted.");
    } catch (error) {
      console.log("Error deleting map: ", error);
    }
    try {
      const command = new DeleteGeofenceCollectionCommand(
        deleteGeofenceCollParams,
      );
      const response = await locationClient.send(command);
      console.log("Geofence collection deleted.");
    } catch (error) {
      console.log("Error deleting geofence collection: ", error);
    }
    try {
      const command = new DeleteRouteCalculatorCommand(
        deleteRouteCalculatorParams,
      );
      const response = await locationClient.send(command);
      console.log("Route calculator deleted.");
    } catch (error) {
      console.log("Error deleting route calculator: ", error);
    }
    try {
      const command = new DeleteTrackerCommand(deleteTrackerParams);
      const response = await locationClient.send(command);
      console.log("Tracker deleted.");
    } catch (error) {
      console.log("Error deleting tracker: ", error);
    }
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Location Service Use demo. We hope you "
 +
    "learned something new, or got some inspiration for your own apps today!" +
    " For more Amazon Location Services examples in different programming
 languages, have a look at: " +
    "https://docs.aws.amazon.com/code-library/latest/ug/
location_code_examples.html",
);

const myScenario = new Scenario("Location Services Scenario", [
```

```
  greet,
  pressEnter,
  displayCreateAMap,
  sdkCreateAMap,
  pressEnter,
  displayMapUrl,
  sdkDisplayMapUrl,
  pressEnter,
  displayCreateGeoFenceColl,
  sdkCreateGeoFenceColl,
  pressEnter,
  displayStoreGeometry,
  sdkStoreGeometry,
  pressEnter,
  displayCreateTracker,
  sdkCreateTracker,
  pressEnter,
  displayUpdatePosition,
  sdkUpdatePosition,
  pressEnter,
  displayRetrievePosition,
  sdkRetrievePosition,
  pressEnter,
  displayCreateRouteCalc,
  sdkCreateRouteCalc,
  pressEnter,
  displayDetermineDist,
  sdkDetermineDist,
  pressEnter,
  displayUseGeoPlacesClient,
  sdkUseGeoPlacesClient,
  pressEnter,
  displayDeleteResources,
  pressEnterConfirm,
  sdkDeleteResources,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
```

```
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

        Where:
            mapName - The name of the map to create (e.g., "AWSMap").
            keyName - The name of the API key to create (e.g., "AWSApiKey").
            collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
            geoId - The geographic identifier used for the geofence or map (e.g.,
 "geoId").
            trackerName - The name of the tracker (e.g., "geoTracker").
            calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
            deviceId - The ID of the device (e.g., "iPhone-112356").
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(0)
```

```
    }

    val mapName = args[0]
    val keyName = args[1]
    val collectionName = args[2]
    val geoId = args[3]
    val trackerName = args[4]
    val calculatorName = args[5]
    val deviceId = args[6]

    println(
        """
    AWS Location Service is a fully managed service offered by Amazon Web
Services (AWS) that
    provides location-based services for developers. This service simplifies
    the integration of location-based features into applications, making it
    easier to build and deploy location-aware applications.

    The AWS Location Service offers a range of location-based services,
    including:

    - Maps: The service provides access to high-quality maps, satellite imagery,
      and geospatial data from various providers, allowing developers to
      easily embed maps into their applications.

    - Tracking: The Location Service enables real-time tracking of mobile
devices,
      assets, or other entities, allowing developers to build applications
      that can monitor the location of people, vehicles, or other objects.

    - Geocoding: The service provides the ability to convert addresses or
      location names into geographic coordinates (latitude and longitude),
      and vice versa, enabling developers to integrate location-based search
      and routing functionality into their applications.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println(DASHES)
    println("1. Create an AWS Location Service map")
    println(
        """
        An AWS Location map can enhance the user experience of your
        application by providing accurate and personalized location-based
```

```
        features. For example, you could use the geocoding capabilities to
        allow users to search for and locate businesses, landmarks, or
        other points of interest within a specific region.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    val mapArn = createMap(mapName)
    println("The Map ARN is: $mapArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    waitForInputToContinue(scanner)
    println("2. Create an AWS Location API key")
    println(
        """
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.

        """.trimIndent(),
    )
    val keyArn = createKey(keyName, mapArn)
    println("The Key ARN is: $keyArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("3. Display Map URL")
    println(
        """
        In order to get the MAP URL, you need to get the API Key value.
        You can get the key value using the AWS Management Console under
        Location Services. This operation cannot be completed using the
        AWS SDK. For more information about getting the key value, see
        the AWS Location Documentation.
        """.trimIndent(),
    )
    val mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/$mapName/tiles/
{z}/{x}/{y}?key={KeyValue}"
    println("Embed this URL in your Web app: $mapUrl")
```

```
    println("")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("4. Create a geofence collection, which manages and stores
geofences.")
    waitForInputToContinue(scanner)
    val collectionArn: String =
        createGeofenceCollection(collectionName)
    println("The geofence collection was successfully created: $collectionArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println("5. Store a geofence geometry in a given geofence collection.")
    println(
        """
        An AWS Location geofence is a virtual boundary that defines a geographic
area
        on a map. It is a useful feature for tracking the location of
        assets or monitoring the movement of objects within a specific region.

        To define a geofence, you need to specify the coordinates of a
        polygon that represents the area of interest. The polygon must be
        defined in a counter-clockwise direction, meaning that the points of
        the polygon must be listed in a counter-clockwise order.

        This is a requirement for the AWS Location service to correctly
        interpret the geofence and ensure that the location data is
        accurately processed within the defined area.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    putGeofence(collectionName, geoId)
    println("Successfully created geofence: $geoId")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Create a tracker resource which lets you retrieve current and
historical location of devices.")
    waitForInputToContinue(scanner)
    val trackerArn: String = createTracker(trackerName)
```

```
    println("Successfully created tracker. ARN: $trackerArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Update the position of a device in the location tracking
system.")
    println(
        """
        The AWS location service does not enforce a strict format for deviceId,
but it must:
            - Be a string (case-sensitive).
            - Be 1-100 characters long.
            - Contain only:
            - Alphanumeric characters (A-Z, a-z, 0-9)
            - Underscores (_)
            - Hyphens (-)
            - Be the same ID used when sending and retrieving positions.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    updateDevicePosition(trackerName, deviceId)
    println("$deviceId was successfully updated in the location tracking
system.")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("8. Retrieve the most recent position update for a specified
device.")
    waitForInputToContinue(scanner)
    val response = getDevicePosition(trackerName, deviceId)
    println("Successfully fetched device position: ${response.position}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("9. Create a route calculator.")
    waitForInputToContinue(scanner)
    val routeResponse = createRouteCalculator(calculatorName)
    println("Route calculator created successfully:
${routeResponse.calculatorArn}")
```

```
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("10. Determine the distance in kilometers between Seattle and
Vancouver using the route calculator.")
    waitForInputToContinue(scanner)
    val responseDis = calcDistance(calculatorName)
    println("Successfully calculated route. The distance in kilometers is
${responseDis.summary?.distance}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("11. Use the GeoPlacesClient to perform additional operations.")
    println(
        """
        This scenario will show use of the GeoPlacesClient that enables
        location search and geocoding capabilities for your applications.

        We are going to use this client to perform these AWS Location tasks:
            - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
            - Place Search (searchText): Finds places based on search queries.
            - Nearby Search (searchNearby): Finds places near a specific
location.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println("First we will perform a Reverse Geocoding operation")
    waitForInputToContinue(scanner)
    reverseGeocode()

    println("Now we are going to perform a text search using coffee shop.")
    waitForInputToContinue(scanner)
    searchText("coffee shop")
    waitForInputToContinue(scanner)

    println("Now we are going to perform a nearby Search.")
    waitForInputToContinue(scanner)
    searchNearby()
    waitForInputToContinue(scanner)
```

```
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS Location Services resources.")
    println("Would you like to delete the AWS Location Services resources? (y/
n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        deleteMap(mapName)
        deleteKey(keyName)
        deleteGeofenceCollection(collectionName)
        deleteTracker(trackerName)
        deleteRouteCalculator(calculatorName)
    } else {
        println("The AWS resources will not be deleted.")
    }
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println(" This concludes the AWS Location Service scenario.")
    println(DASHES)
}

/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}


/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
```

```
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}


/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence
 collection has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
```

```
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}


/**
 * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
 * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
 * The results are processed and printed once the search completes successfully.
 */
suspend fun searchNearby() {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    // Set up the request for searching nearby places.
    val request = SearchNearbyRequest {
        this.queryPosition = queryPosition
        this.queryRadius = 1000L
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchNearby(request)

        // Process the response and print the results.
        response.resultItems?.forEach { result ->
            println("Title: ${result.title}")
            println("Address: ${result.address?.label}")
```

```
            println("Distance: ${result.distance} meters")
            println("-------------------------")
        }
    }
}


/**
 * Searches for a place using the provided search query and prints the detailed
 information of the first result.
 *
 * @param searchQuery the search query to be used for the place search (ex,
 coffee shop)
 */
suspend fun searchText(searchQuery: String) {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    val request = SearchTextRequest {
        this.queryText = searchQuery
        this.biasPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchText(request)

        response.resultItems?.firstOrNull()?.let { result ->
            val placeId = result.placeId // Get Place ID
            println("Found Place with id: $placeId")

            // Fetch detailed info using getPlace.
            val getPlaceRequest = GetPlaceRequest {
                this.placeId = placeId
            }

            val placeResponse = client.getPlace(getPlaceRequest)

            // Print detailed place information.
            println("Detailed Place Information:")
            println("Title: ${placeResponse.title}")
            println("Address: ${placeResponse.address?.label}")

            // Print each food type (if any).
```

```
            placeResponse.foodTypes?.takeIf { it.isNotEmpty() }?.let {
                println("Food Types:")
                it.forEach { foodType ->
                    println("  - $foodType")
                }
            } ?: run {
                println("No food types available.")
            }

            println("-------------------------")
        }
    }
}

/**
 * Performs reverse geocoding using the AWS Geo Places API.
 * Reverse geocoding is the process of converting geographic coordinates
 (latitude and longitude) to a human-readable address.
 * This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.
 */
suspend fun reverseGeocode() {
    val latitude = 37.7749
    val longitude = -122.4194
    println("Use latitude 37.7749 and longitude -122.4194")

    // AWS expects [longitude, latitude].
    val queryPosition = listOf(longitude, latitude)
    val request = ReverseGeocodeRequest {
        this.queryPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.reverseGeocode(request)
        response.resultItems?.forEach { result ->
            println("The address is: ${result.address?.label}")
        }
    }
}


/**
 * Calculates the distance between two locations.
 *
```

```
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the
 route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters,
 Kilometers, Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}

/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}

/**
```

```
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}

/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude,
 latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the
 Amazon Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}

/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence
 to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
```

```
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}

/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}

/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
```

```
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}

/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}

fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            println("Invalid input. Please try again.")
        }
```

```
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

Ações para Amazon Location usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais de localização da Amazon com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Location Service.

Exemplos

- Use BatchUpdateDevicePositioncom um AWS SDK
- Use CalculateRoutecom um AWS SDK
- Use CreateGeofenceCollectioncom um AWS SDK

- Use CreateKeycom um AWS SDK
- Use CreateMapcom um AWS SDK
- Use CreateRouteCalculatorcom um AWS SDK
- Use CreateTrackercom um AWS SDK
- Use DeleteGeofenceCollectioncom um AWS SDK
- Use DeleteKeycom um AWS SDK
- Use DeleteMapcom um AWS SDK
- Use DeleteRouteCalculatorcom um AWS SDK
- Use DeleteTrackercom um AWS SDK
- Use GetDevicePositioncom um AWS SDK
- Use PutGeofencecom um AWS SDK

Use **`BatchUpdateDevicePosition`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `BatchUpdateDevicePosition`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Updates the position of a device in the location tracking system.
 *
```

```
     * @param trackerName the name of the tracker associated with the device
     * @param deviceId    the unique identifier of the device
     * @throws RuntimeException if an error occurs while updating the device
position
     */
    public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
        double latitude = 37.7749;  // Example: San Francisco
        double longitude = -122.4194;

        DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
            .deviceId(deviceId)
            .sampleTime(Instant.now()) // Timestamp of position update.
            .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
            .build();

        BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
            .trackerName(trackerName)
            .updates(positionUpdate)
            .build();

        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: "
+ cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device
position: " + exception.getMessage(), exception);
                }
            }
        });
    }
```

- Para obter detalhes da API, consulte BatchUpdateDevicePositiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  BatchUpdateDevicePositionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });
const updateDevicePosParams = {
  TrackerName: `${data.inputs.trackerName}`,
  Updates: [
    {
      DeviceId: `${data.inputs.deviceId}`,
      SampleTime: new Date(),
      Position: [-122.4194, 37.7749],
    },
  ],
};
export const main = async () => {
  try {
    const command = new BatchUpdateDevicePositionCommand(updateDevicePosParams);
    const response = await locationClient.send(command);
    //console.log("response ", response.Errors[0].Error);

    console.log(
      `Device with id ${data.inputs.deviceId} was successfully updated in the
 location tracking system. `,
      response,
    );
  } catch (error) {
    console.log("error ", error);
```

```
  }
};
```

- Para obter detalhes da API, consulte BatchUpdateDevicePositiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude,
 latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

- Para obter detalhes da API, consulte a BatchUpdateDevicePositionreferência da API AWS SDK for Kotlin.

Use **`CalculateRoute`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CalculateRoute`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Calculates the distance between two locations asynchronously.
    *
    * @param routeCalcName the name of the route calculator to use
    * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the
route
    */
   public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
       // Define coordinates for Seattle, WA and Vancouver, BC.
       List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
```

```
        List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

        CalculateRouteRequest request = CalculateRouteRequest.builder()
            .calculatorName(routeCalcName)
            .departurePosition(departurePosition)
            .destinationPosition(arrivePosition)
            .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
            .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
            .build();

        return getClient().calculateRoute(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
 found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
 exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte CalculateRoutea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  CalculateRouteCommand,
```

```
  ResourceNotFoundException,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });

export const main = async () => {
  const routeCalcParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
    DeparturePosition: [-122.3321, 47.6062],
    DestinationPosition: [-123.1216, 49.2827],
    TravelMode: "Car",
    DistanceUnit: "Kilometers",
  };
  try {
    const command = new CalculateRouteCommand(routeCalcParams);
    const response = await locationClient.send(command);

    console.log(
      "Successfully calculated route. The distance in kilometers is : ",
      response.Summary.Distance,
    );
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `An conflict occurred: ${caught.message} \n Exiting program.`,
      );
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte CalculateRoutea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the
 route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters,
 Kilometers, Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}
```

- Para obter detalhes da API, consulte a CalculateRoutereferência da API AWS SDK for Kotlin.

Use **`CreateGeofenceCollection`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateGeofenceCollection`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new geofence collection.
    *
    * @param collectionName the name of the geofence collection to be created
    */
   public CompletableFuture<String> createGeofenceCollection(String
collectionName) {
       CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .description("Created by using the AWS SDK for Java")
           .build();

       return getClient().createGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("The geofence collection
was not created due to ConflictException.", cause);
                   }
```

```
                    throw new CompletionException("Failed to create geofence
 collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the
 ARN
    }
```

- Para obter detalhes da API, consulte CreateGeofenceCollectiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  ConflictException,
  CreateGeofenceCollectionCommand,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const geoFenceCollParams = {
    CollectionName: `${data.inputs.collectionName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateGeofenceCollectionCommand(geoFenceCollParams);
```

```
    const response = await locationClient.send(command);
    console.log(
      "Collection created. Collection name is: ",
      response.CollectionName,
    );
  } catch (caught) {
    if (caught instanceof ConflictException) {
      console.error("A conflict occurred. Exiting program.");
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateGeofenceCollectiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
```

```
        return response.collectionArn
    }
}
```

- Para obter detalhes da API, consulte a CreateGeofenceCollectionreferência da API AWS SDK for Kotlin.

Use **`CreateKey`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new API key with the specified name and restrictions.
    *
    * @param keyName the name of the API key to be created
    * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
    * @return a {@link CompletableFuture} that completes with the Amazon
Resource Name (ARN) of the created API key,
    * or {@code null} if the operation failed
    */
   public CompletableFuture<String> createKey(String keyName, String mapArn) {
       ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
```

```
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }
```

- Para obter detalhes da API, consulte CreateKeya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new API key with the specified name and restrictions.
```

```
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}
```

- Para obter detalhes da API, consulte a CreateKeyreferência da API AWS SDK for Kotlin.

Use **`CreateMap`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateMap`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new map with the specified name and configuration.
    *
    * @param mapName the name of the map to be created
    * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
    * @throws CompletionException if an error occurs while creating the map,
such as exceeding the service quota
    */
   public CompletableFuture<String> createMap(String mapName) {
       MapConfiguration configuration = MapConfiguration.builder()
           .style("VectorEsriNavigation")
           .build();

       CreateMapRequest mapRequest = CreateMapRequest.builder()
           .mapName(mapName)
           .configuration(configuration)
           .description("A map created using the Java V2 API")
           .build();

       return getClient().createMap(mapRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ServiceQuotaExceededException) {
                       throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                   }
                   throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
               }
```

```
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }
```

- Para obter detalhes da API, consulte CreateMapa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import { CreateMapCommand, LocationClient } from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const CreateMapCommandInput = {
    MapName: `${data.inputs.mapName}`,
    Configuration: { style: "VectorEsriNavigation" },
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateMapCommand(CreateMapCommandInput);
    const response = await locationClient.send(command);
    console.log("Map created. Map ARN is : ", response.MapArn);
  } catch (error) {
    console.error("Error creating map: ", error);
    throw error;
  }
};
```

- Para obter detalhes da API, consulte CreateMapa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}
```

- Para obter detalhes da API, consulte a CreateMapreferência da API AWS SDK for Kotlin.

Use **`CreateRouteCalculator`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateRouteCalculator`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new route calculator with the specified name and data source.
    *
    * @param routeCalcName the name of the route calculator to be created
    */
   public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
       String dataSource = "Esri"; // or "Here"
       CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
           .calculatorName(routeCalcName)
           .dataSource(dataSource)
           .build();

       return getClient().createRouteCalculator(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("A conflict error occurred:
" + cause.getMessage(), cause);
                   }
```

```
                    throw new CompletionException("Failed to create route
 calculator: " + exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateRouteCalculatora Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  ConflictException,
  CreateRouteCalculatorCommand,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });

export const main = async () => {
  const routeCalcParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
    DataSource: "Esri",
  };
  try {
    const command = new CreateRouteCalculatorCommand(routeCalcParams);
    const response = await locationClient.send(command);

    console.log(
```

```
      "Route calculator created successfully. Calculator name is ",
      response.CalculatorName,
    );
  } catch (caught) {
    if (caught instanceof ConflictException) {
      console.error(
        `An conflict occurred: ${caught.message} \n Exiting program.`,
      );
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateRouteCalculatora Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}
```

- Para obter detalhes da API, consulte a CreateRouteCalculatorreferência da API AWS SDK for Kotlin.

Use **`CreateTracker`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateTracker`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
     *
     * @param trackerName the name of the tracker to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
     */
    public CompletableFuture<String> createTracker(String trackerName) {
        CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
```

```
            .description("Created using the Java V2 SDK")
            .trackerName(trackerName)
            .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
 AccuracyBased
            .build();

        return getClient().createTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("Conflict occurred while
 creating tracker: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating tracker: " +
 exception.getMessage(), exception);
                }
            })
            .thenApply(CreateTrackerResponse::trackerArn); // Return only the
 tracker ARN
    }
```

- Para obter detalhes da API, consulte CreateTrackera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import { CreateTrackerCommand, LocationClient } from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
```

```
const region = "eu-west-1";

export const main = async () => {
  const createTrackerParams = {
    TrackerName: `${data.inputs.trackerName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateTrackerCommand(createTrackerParams);
    const response = await locationClient.send(command);
    //state.trackerName - response.TrackerName;
    console.log("Tracker created. Tracker name is : ", response.TrackerName);
  } catch (error) {
    console.error("Error creating map: ", error);
    throw error;
  }
};
```

- Para obter detalhes da API, consulte CreateTrackera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the
 Amazon Resource Name (ARN) of the created tracker
 */
```

```
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}
```

- Para obter detalhes da API, consulte a CreateTrackerreferência da API AWS SDK for Kotlin.

Use **`DeleteGeofenceCollection`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteGeofenceCollection`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a geofence collection asynchronously.
     *
```

```
     * @param collectionName the name of the geofence collection to be deleted
     * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
     */
    public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
        DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .build();

        return getClient().deleteGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested geofence
collection was not found.", cause);
                    }
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte DeleteGeofenceCollectiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  DeleteGeofenceCollectionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteGeofenceCollParams = {
    CollectionName: `${data.inputs.collectionName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new DeleteGeofenceCollectionCommand(
      deleteGeofenceCollParams,
    );
    const response = await locationClient.send(command);
    console.log("Collection deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `${data.inputs.collectionName} Geofence collection not found.`,
      );
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteGeofenceCollectiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence
 collection has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteGeofenceCollectionreferência da API AWS SDK for Kotlin.

Use **`DeleteKey`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteKey`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes the specified key from the key-value store.
    *
    * @param keyName the name of the key to be deleted
    * @return a {@link CompletableFuture} that completes when the key has been
deleted
    * @throws CompletionException if the key was not found or if an error
occurred during the deletion process
    */
   public CompletableFuture<Void> deleteKey(String keyName) {
       DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
           .keyName(keyName)
           .build();

       return getClient().deleteKey(keyRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The key was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
               }
               logger.info("The key {} was deleted.", keyName);
           })
           .thenApply(response -> null);
   }
```

- Para obter detalhes da API, consulte DeleteKeya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteKeyreferência da API AWS SDK for Kotlin.

Use **`DeleteMap`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteMap`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a map with the specified name.
    *
    * @param mapName the name of the map to be deleted
    * @return a {@link CompletableFuture} that completes when the map deletion
is successful, or throws a {@link CompletionException} if an error occurs
    */
   public CompletableFuture<Void> deleteMap(String mapName) {
       DeleteMapRequest mapRequest = DeleteMapRequest.builder()
           .mapName(mapName)
           .build();

       return getClient().deleteMap(mapRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The map was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
               }
               logger.info("The map {} was deleted.", mapName);
           })
           .thenApply(response -> null);
   }
```

- Para obter detalhes da API, consulte DeleteMapa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  DeleteMapCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteMapParams = {
    MapName: `${data.inputs.mapName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteMapCommand(deleteMapParams);
    const response = await locationClient.send(command);
    console.log("Map deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(`${data.inputs.mapName} map not found.`);
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteMapa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteMapreferência da API AWS SDK for Kotlin.

Use **`DeleteRouteCalculator`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteRouteCalculator`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a route calculator from the system.
    *
    * @param calcName the name of the route calculator to delete
    * @return a {@link CompletableFuture} that completes when the route
calculator has been deleted
    * @throws CompletionException if an error occurs while deleting the route
calculator
    *                            - If the route calculator was not found, a
{@link ResourceNotFoundException} will be thrown
    *                            - If any other error occurs, a generic {@link
CompletionException} will be thrown
    */
   public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
       DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
           .calculatorName(calcName)
           .build();

       return getClient().deleteRouteCalculator(calculatorRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The route calculator was
not found.", cause);
                   }
                   throw new CompletionException("Failed to delete the route
calculator: " + exception.getMessage(), exception);
               }
               logger.info("The route calculator {} was deleted.", calcName);
```

```
            })
            .thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte DeleteRouteCalculatora Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  DeleteRouteCalculatorCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteRouteCalculatorParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteRouteCalculatorCommand(
      deleteRouteCalculatorParams,
    );
    const response = await locationClient.send(command);
    console.log("Route calculator deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
```

```
      console.error(
        `${data.inputs.calculatorName} route calculator not found.`,
      );
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteRouteCalculatora Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteRouteCalculatorreferência da API AWS SDK for Kotlin.

Use **`DeleteTracker`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteTracker`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has
been deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();

        return getClient().deleteTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
```

```
                        throw new CompletionException("The tracker was not
 found.", cause);
                    }
                    throw new CompletionException("Failed to delete the tracker:
 " + exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }
```

- Para obter detalhes da API, consulte DeleteTrackera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  DeleteTrackerCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteTrackerParams = {
    TrackerName: `${data.inputs.trackerName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
```

```
    const command = new DeleteTrackerCommand(deleteTrackerParams);
    const response = await locationClient.send(command);
    console.log("Tracker deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(`${data.inputs.trackerName} tracker not found.`);
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteTrackera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTrackerreferência da API AWS SDK for Kotlin.

Use **`GetDevicePosition`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetDevicePosition`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the position of a device using the provided LocationClient.
    *
    * @param trackerName The name of the tracker associated with the device.
    * @param deviceId    The ID of the device to retrieve the position for.
    * @throws RuntimeException If there is an error fetching the device
position.
    */
   public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
       GetDevicePositionRequest request = GetDevicePositionRequest.builder()
           .trackerName(trackerName)
           .deviceId(deviceId)
           .build();

       return getClient().getDevicePosition(request)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
 found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device
 position: " + exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte GetDevicePositiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  GetDevicePositionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const locationClient = new LocationClient({ region: region });
  const deviceId = `${data.inputs.deviceId}`;
  const trackerName = `${data.inputs.trackerName}`;
```

```
  const devicePositionParams = {
    DeviceId: deviceId,
    TrackerName: trackerName,
  };
  try {
    const command = new GetDevicePositionCommand(devicePositionParams);
    const response = await locationClient.send(command);
    //state.position = response.position;
    console.log("Successfully fetched device position: ", response);
  } catch (error) {
    console.log("Error ", error);
    /*  if (caught instanceof ResourceNotFoundException) {
      console.error(
        `"The resource was not found: ${caught.message} \n Exiting program.`,
      );
    } else {
      `An unexpected error error occurred: ${caught.message} \n Exiting
 program.`;
    }
    return;*/
  }
};
```

- Para obter detalhes da API, consulte GetDevicePositiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Retrieves the position of a device using the provided LocationClient.
 *
```

```
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}
```

- Para obter detalhes da API, consulte a GetDevicePositionreferência da API AWS SDK for Kotlin.

Use **`PutGeofence`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PutGeofence`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Adds a new geofence to the specified collection.
```

```
     *
     * @param collectionName the name of the geofence collection to add the
geofence to
     * @param geoId          the unique identifier for the geofence
     */
    public CompletableFuture<PutGeofenceResponse> putGeofence(String
collectionName, String geoId) {
        // Define the geofence geometry (polygon).
        GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
            .polygon(List.of(
                List.of(
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();

        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte PutGeofencea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  PutGeofenceCommand,
  LocationClient,
  ValidationException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });
export const main = async () => {
  const geoFenceGeoParams = {
    CollectionName: `${data.inputs.collectionName}`,
    GeofenceId: `${data.inputs.geoId}`,
    Geometry: {
      Polygon: [
        [
          [-122.3381, 47.6101],
          [-122.3281, 47.6101],
          [-122.3281, 47.6201],
          [-122.3381, 47.6201],
          [-122.3381, 47.6101],
        ],
      ],
    },
  };
  try {
    const command = new PutGeofenceCommand(geoFenceGeoParams);
    const response = await locationClient.send(command);
    console.log("GeoFence created. GeoFence ID is: ", response.GeofenceId);
  } catch (error) {
    console.error(
```

```
      `A validation error occurred while creating geofence: ${error} \n Exiting
 program.`,
    );
    return;
  }
};
```

- Para obter detalhes da API, consulte PutGeofencea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence
 to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }
```

```
    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}
```

- Para obter detalhes da API, consulte a PutGeofencereferência da API AWS SDK for Kotlin.

Exemplos de código para Location Service Places usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Location Service Places com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Location Service Places: mais informações sobre o Location Service Places.
- Referência de API do Location Service Places: detalhes sobre todas as ações disponíveis do Location Service Places.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Location Service Places usando AWS SDKs

- Ações para Location Service Places usando AWS SDKs
  - Use ReverseGeocodecom um AWS SDK
  - Use SearchNearbycom um AWS SDK
  - Use SearchTextcom um AWS SDK

Exemplos básicos de Location Service Places usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Location Service Places com AWS SDKs.

Exemplos

- Ações para Location Service Places usando AWS SDKs
  - Use ReverseGeocodecom um AWS SDK
  - Use SearchNearbycom um AWS SDK
  - Use SearchTextcom um AWS SDK

Ações para Location Service Places usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Location Service Places com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Location Service Places.

Exemplos

- Use ReverseGeocodecom um AWS SDK
- Use SearchNearbycom um AWS SDK
- Use SearchTextcom um AWS SDK

Use **`ReverseGeocode`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ReverseGeocode`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Performs reverse geocoding using the AWS Geo Places API.
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       logger.info("Use latitude 37.7749 and longitude -122.4194");

       // AWS expects [longitude, latitude].
       List<Double> queryPosition = List.of(longitude, latitude);
       ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
           .queryPosition(queryPosition)
           .build();
       CompletableFuture<ReverseGeocodeResponse> futureResponse =
           getGeoPlacesClient().reverseGeocode(request);

       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                   throw new CompletionException("A validation error occurred: "
+ cause.getMessage(), cause);
               }
               throw new CompletionException("Error performing reverse
geocoding", exception);
           }
```

```
            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }
```

- Para obter detalhes da API, consulte ReverseGeocodea Referência AWS SDK for Java 2.x da API.

Use **`SearchNearby`**com um AWS SDK

O código de exemplo a seguir mostra como usar `SearchNearby`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Performs a nearby places search based on the provided geographic
 coordinates (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes
 successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
```

```
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error
occurred: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place
search", exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }
```

- Para obter detalhes da API, consulte SearchNearbya Referência AWS SDK for Java 2.x da API.

Use **`SearchText`**com um AWS SDK

O código de exemplo a seguir mostra como usar `SearchText`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Searches for a place using the provided search query and prints the
detailed information of the first result.
    *
    * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
    */
   public CompletableFuture<Void> searchText(String searchQuery) {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       List<Double> queryPosition = List.of(longitude, latitude);

       SearchTextRequest request = SearchTextRequest.builder()
               .queryText(searchQuery)
               .biasPosition(queryPosition)
               .build();

       return getGeoPlacesClient().searchText(request)
               .thenCompose(response -> {
                   if (response.resultItems().isEmpty()) {
                       logger.info("No places found.");
                       return CompletableFuture.completedFuture(null);
                   }

                   // Get the first place ID
                   String placeId = response.resultItems().get(0).placeId();
                   logger.info("Found Place with id: " + placeId);

                   // Fetch detailed info using getPlace
                   GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                           .placeId(placeId)
```

```
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType ->
 {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error
 occurred: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place
 search", exception);
                });
    }
```

- Para obter detalhes da API, consulte SearchTexta Referência AWS SDK for Java 2.x da API.

Exemplos de código para Amazon Managed Grafana usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Managed Grafana com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Managed Grafana — Mais informações sobre o Amazon Managed Grafana.
- Referência da API Amazon Managed Grafana — Detalhes sobre todas as ações disponíveis do Amazon Managed Grafana.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para o Amazon Managed Grafana usando AWS SDKs
  - Como criar um espaço de trabalho de monitoramento gerenciado

Cenários para o Amazon Managed Grafana usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Managed Grafana com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon Managed Grafana ou combinadas com outras. Serviços da AWS Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Como criar um espaço de trabalho de monitoramento gerenciado

Como criar um espaço de trabalho de monitoramento gerenciado

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para seu espaço de trabalho.
- Criar um espaço de trabalho do Grafana.
- Configurar autenticação
- Definir configurações opcionais.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
```

```
    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"
```

```
echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
```

```
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
```

```
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-
grafana-gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id
 "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -
f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
```

```
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML
 settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
```

```
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 |
 grep -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
```

```
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Exemplos de código para Managed Service for Apache Flink usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Managed Service para Apache Flink com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do Desenvolvedor do Managed Service for Apache Flink: mais informações sobre Managed Service for Apache Flink.
- Referência da API Managed Service for Apache Flink: detalhes sobre todas as ações do Managed Service for Apache Flink.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de serviço gerenciado para Apache Flink usando AWS SDKs
  - Ações do Managed Service para Apache Flink usando AWS SDKs
    - Use AddApplicationInputcom um AWS SDK
    - Use AddApplicationOutputcom um AWS SDK
    - Use CreateApplicationcom um AWS SDK
    - Use DeleteApplicationcom um AWS SDK
    - Use DescribeApplicationcom um AWS SDK
    - Use DescribeApplicationSnapshotcom um AWS SDK
    - Use DiscoverInputSchemacom um AWS SDK
    - Use StartApplicationcom um AWS SDK
    - Use StopApplicationcom um AWS SDK
    - Use UpdateApplicationcom um AWS SDK
- Gerador de dados para o Managed Service for Apache Flink
  - Gere um stream do Kinesis com um referenciador usando um AWS SDK
  - Gere um stream do Kinesis com anomalias da pressão arterial usando um AWS SDK
  - Gere um stream do Kinesis com dados em colunas usando um AWS SDK
  - Gere um stream do Kinesis com anomalias da frequência cardíaca usando um AWS SDK

- Gere um stream do Kinesis com pontos de acesso usando um AWS SDK
- Gere um stream do Kinesis com entradas de registro usando um AWS SDK
- Gere um stream do Kinesis com dados escalonados usando um AWS SDK
- Gere um stream do Kinesis com dados de ações usando um AWS SDK
- Gere um stream do Kinesis com dois tipos de dados usando um AWS SDK
- Gere um stream do Kinesis com dados de log da web usando um AWS SDK

Exemplos básicos de serviço gerenciado para Apache Flink usando AWS SDKs

Os exemplos de código a seguir mostram como usar os elementos básicos do Amazon Managed Service for Apache Flink com AWS SDKs.

Exemplos

- Ações do Managed Service para Apache Flink usando AWS SDKs
  - Use AddApplicationInputcom um AWS SDK
  - Use AddApplicationOutputcom um AWS SDK
  - Use CreateApplicationcom um AWS SDK
  - Use DeleteApplicationcom um AWS SDK
  - Use DescribeApplicationcom um AWS SDK
  - Use DescribeApplicationSnapshotcom um AWS SDK
  - Use DiscoverInputSchemacom um AWS SDK
  - Use StartApplicationcom um AWS SDK
  - Use StopApplicationcom um AWS SDK
  - Use UpdateApplicationcom um AWS SDK

Ações do Managed Service para Apache Flink usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Managed Service for Apache Flink com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Managed Service for Apache Flink.

Exemplos

- Use AddApplicationInputcom um AWS SDK
- Use AddApplicationOutputcom um AWS SDK
- Use CreateApplicationcom um AWS SDK
- Use DeleteApplicationcom um AWS SDK
- Use DescribeApplicationcom um AWS SDK
- Use DescribeApplicationSnapshotcom um AWS SDK
- Use DiscoverInputSchemacom um AWS SDK
- Use StartApplicationcom um AWS SDK
- Use StopApplicationcom um AWS SDK
- Use UpdateApplicationcom um AWS SDK

Use **`AddApplicationInput`**com um AWS SDK

O código de exemplo a seguir mostra como usar `AddApplicationInput`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
```

```
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def add_input(self, input_prefix, stream_arn, input_schema):
        """
        Adds an input stream to the application. The input stream data is mapped
        to an in-application stream that can be processed by your code running in
        Kinesis Data Analytics.

        :param input_prefix: The prefix prepended to in-application input stream
names.
        :param stream_arn: The ARN of the input stream.
        :param input_schema: A schema that maps the data in the input stream to
the
                             runtime environment. This can be automatically
generated
                             by using `discover_input_schema` or you can create
it
                             yourself.
        :return: Metadata about the newly added input.
        """
        try:
            response = self.analytics_client.add_application_input(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Input={
                    "NamePrefix": input_prefix,
                    "KinesisStreamsInput": {"ResourceARN": stream_arn},
                    "InputSchema": input_schema,
                },
            )
            self.version_id = response["ApplicationVersionId"]
            logger.info("Add input stream %s to application %s.", stream_arn,
self.name)
        except ClientError:
            logger.exception(
                "Couldn't add input stream %s to application %s.", stream_arn,
self.name
            )
            raise
        else:
```

```
            return response
```

- Para obter detalhes da API, consulte a AddApplicationInputReferência da API AWS SDK for Python (Boto3).

Use **AddApplicationOutput**com um AWS SDK

O código de exemplo a seguir mostra como usar `AddApplicationOutput`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def add_output(self, in_app_stream_name, output_arn):
        """
        Adds an output stream to the application. Kinesis Data Analytics maps
 data
        from the specified in-application stream to the output stream.
```

```
        :param in_app_stream_name: The name of the in-application stream to map
                                   to the output stream.
        :param output_arn: The ARN of the output stream.
        :return: A list of metadata about the output resources currently assigned
                 to the application.
        """
        try:
            response = self.analytics_client.add_application_output(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Output={
                    "Name": in_app_stream_name,
                    "KinesisStreamsOutput": {"ResourceARN": output_arn},
                    "DestinationSchema": {"RecordFormatType": "JSON"},
                },
            )
            outputs = response["OutputDescriptions"]
            self.version_id = response["ApplicationVersionId"]
            logging.info(
                "Added output %s to %s, which now has %s outputs.",
                output_arn,
                self.name,
                len(outputs),
            )
        except ClientError:
            logger.exception("Couldn't add output %s to %s.", output_arn,
self.name)
            raise
        else:
            return outputs
```

- Para obter detalhes da API, consulte a AddApplicationOutputReferência da API AWS SDK for Python (Boto3).

Use **`CreateApplication`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateApplication`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def create(self, app_name, role_arn, env="SQL-1_0"):
        """
        Creates a Kinesis Data Analytics application.

        :param app_name: The name of the application.
        :param role_arn: The ARN of a role that can be assumed by Kinesis Data
                         Analytics and grants needed permissions.
        :param env: The runtime environment of the application, such as SQL. Code
                    uploaded to the application runs in this environment.
        :return: Metadata about the newly created application.
        """
        try:
            response = self.analytics_client.create_application(
                ApplicationName=app_name,
                RuntimeEnvironment=env,
                ServiceExecutionRole=role_arn,
            )
            details = response["ApplicationDetail"]
```

```
            self._update_details(details)
            logger.info("Application %s created.", app_name)
        except ClientError:
            logger.exception("Couldn't create application %s.", app_name)
            raise
        else:
            return details
```

- Para obter detalhes da API, consulte a CreateApplicationReferência da API AWS SDK for Python (Boto3).

Use **`DeleteApplication`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteApplication`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None
```

```
    def delete(self):
        """
        Deletes an application.
        """
        try:
            self.analytics_client.delete_application(
                ApplicationName=self.name, CreateTimestamp=self.create_timestamp
            )
            logger.info("Deleted application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete application %s.", self.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteApplicationReferência da API AWS SDK for Python (Boto3).

Use **`DescribeApplication`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeApplication`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
```

```
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def describe(self, name):
        """
        Gets metadata about an application.

        :param name: The name of the application to look up.
        :return: Metadata about the application.
        """
        try:
            response =
self.analytics_client.describe_application(ApplicationName=name)
            details = response["ApplicationDetail"]
            self._update_details(details)
            logger.info("Got metadata for application %s.", name)
        except ClientError:
            logger.exception("Couldn't get metadata for application %s.", name)
            raise
        else:
            return details
```

- Para obter detalhes da API, consulte a DescribeApplicationReferência da API AWS SDK for Python (Boto3).

Use **`DescribeApplicationSnapshot`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeApplicationSnapshot`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def describe_snapshot(self, application_name, snapshot_name):
        """
        Gets metadata about a previously saved application snapshot.

        :param application_name: The name of the application.
        :param snapshot_name: The name of the snapshot.
        :return: Metadata about the snapshot.
        """
        try:
            response = self.analytics_client.describe_application_snapshot(
                ApplicationName=application_name, SnapshotName=snapshot_name
            )
            snapshot = response["SnapshotDetails"]
            logger.info(
                "Got metadata for snapshot %s of application %s.",
                snapshot_name,
                application_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get metadata for snapshot %s of application %s.",
                snapshot_name,
                application_name,
            )
            raise
        else:
            return snapshot
```

- Para obter detalhes da API, consulte a DescribeApplicationSnapshotReferência da API AWS SDK for Python (Boto3).

Use **`DiscoverInputSchema`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DiscoverInputSchema`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def discover_input_schema(self, stream_arn, role_arn):
        """
        Discovers a schema that maps data in a stream to a format that is usable
 by
        an application's runtime environment. The stream must be active and have
        enough data moving through it for the service to sample. The returned
 schema
        can be used when you add the stream as an input to the application or you
 can
        write your own schema.
```

```
        :param stream_arn: The ARN of the stream to map.
        :param role_arn: A role that lets Kinesis Data Analytics read from the
stream.
        :return: The discovered schema of the data in the input stream.
        """
        try:
            response = self.analytics_client.discover_input_schema(
                ResourceARN=stream_arn,
                ServiceExecutionRole=role_arn,
                InputStartingPositionConfiguration={"InputStartingPosition":
"NOW"},
            )
            schema = response["InputSchema"]
            logger.info("Discovered input schema for stream %s.", stream_arn)
        except ClientError:
            logger.exception(
                "Couldn't discover input schema for stream %s.", stream_arn
            )
            raise
        else:
            return schema
```

- Para obter detalhes da API, consulte a DiscoverInputSchemaReferência da API AWS SDK for Python (Boto3).

Use **`StartApplication`**com um AWS SDK

O código de exemplo a seguir mostra como usar `StartApplication`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def start(self, input_id):
        """
        Starts an application. After the application is running, it reads from
 the
        specified input stream and runs the application code on the incoming
 data.

        :param input_id: The ID of the input to read.
        """
        try:
            self.analytics_client.start_application(
                ApplicationName=self.name,
                RunConfiguration={
                    "SqlRunConfigurations": [
                        {
                            "InputId": input_id,
                            "InputStartingPositionConfiguration": {
                                "InputStartingPosition": "NOW"
                            },
                        }
                    ]
                },
            )
            logger.info("Started application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't start application %s.", self.name)
            raise
```

- Para obter detalhes da API, consulte a StartApplicationReferência da API AWS SDK for Python (Boto3).

Use **`StopApplication`**com um AWS SDK

O código de exemplo a seguir mostra como usar `StopApplication`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def stop(self):
        """
        Stops an application. This stops the application from processing data but
        does not delete any resources.
        """
        try:
            self.analytics_client.stop_application(ApplicationName=self.name)
            logger.info("Stopping application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't stop application %s.", self.name)
            raise
```

- Para obter detalhes da API, consulte a StopApplicationReferência da API AWS SDK for Python (Boto3).

Use **`UpdateApplication`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UpdateApplication`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo atualiza o código que é executado em um aplicativo existente.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def update_code(self, code):
        """
        Updates the code that runs in the application. The code must run in the
        runtime environment of the application, such as SQL. Application code
```

```
        typically reads data from in-application streams and transforms it in
some way.

        :param code: The code to upload. This completely replaces any existing
code
                     in the application.
        :return: Metadata about the application.
        """
        try:
            response = self.analytics_client.update_application(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                ApplicationConfigurationUpdate={
                    "ApplicationCodeConfigurationUpdate": {
                        "CodeContentTypeUpdate": "PLAINTEXT",
                        "CodeContentUpdate": {"TextContentUpdate": code},
                    }
                },
            )
            details = response["ApplicationDetail"]
            self.version_id = details["ApplicationVersionId"]
            logger.info("Update code for application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't update code for application %s.",
self.name)
            raise
        else:
            return details
```

- Para obter detalhes da API, consulte a UpdateApplicationReferência da API AWS SDK for Python (Boto3).

Gerador de dados para o Managed Service for Apache Flink

Os exemplos de código a seguir mostram como usar o Managed Service para Apache Flink com AWS SDKs.

Exemplos

- Gere um stream do Kinesis com um referenciador usando um AWS SDK

- Gere um stream do Kinesis com anomalias da pressão arterial usando um AWS SDK
- Gere um stream do Kinesis com dados em colunas usando um AWS SDK
- Gere um stream do Kinesis com anomalias da frequência cardíaca usando um AWS SDK
- Gere um stream do Kinesis com pontos de acesso usando um AWS SDK
- Gere um stream do Kinesis com entradas de registro usando um AWS SDK
- Gere um stream do Kinesis com dados escalonados usando um AWS SDK
- Gere um stream do Kinesis com dados de ações usando um AWS SDK
- Gere um stream do Kinesis com dois tipos de dados usando um AWS SDK
- Gere um stream do Kinesis com dados de log da web usando um AWS SDK

Gere um stream do Kinesis com um referenciador usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com um referenciador.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"REFERRER": "http://www.amazon.com"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
```

```
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gere um stream do Kinesis com anomalias da pressão arterial usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com anomalias de pressão arterial.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


class PressureType(Enum):
    low = "LOW"
    normal = "NORMAL"
    high = "HIGH"
```

```
def get_blood_pressure(pressure_type):
    pressure = {"BloodPressureLevel": pressure_type.value}
    if pressure_type == PressureType.low:
        pressure["Systolic"] = random.randint(50, 80)
        pressure["Diastolic"] = random.randint(30, 50)
    elif pressure_type == PressureType.normal:
        pressure["Systolic"] = random.randint(90, 120)
        pressure["Diastolic"] = random.randint(60, 80)
    elif pressure_type == PressureType.high:
        pressure["Systolic"] = random.randint(130, 200)
        pressure["Diastolic"] = random.randint(90, 150)
    else:
        raise TypeError
    return pressure


def generate(stream_name, kinesis_client):
    while True:
        rnd = random.random()
        pressure_type = (
            PressureType.low
            if rnd < 0.005
            else PressureType.high
            if rnd > 0.995
            else PressureType.normal
        )
        blood_pressure = get_blood_pressure(pressure_type)
        print(blood_pressure)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(blood_pressure),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gere um stream do Kinesis com dados em colunas usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dados em colunas.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"Col_A": "a", "Col_B": "b", "Col_C": "c", "Col_E_Unstructured":
 "x,y,z"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gere um stream do Kinesis com anomalias da frequência cardíaca usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com anomalias da frequência cardíaca.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


class RateType(Enum):
    normal = "NORMAL"
    high = "HIGH"


def get_heart_rate(rate_type):
    if rate_type == RateType.normal:
        rate = random.randint(60, 100)
    elif rate_type == RateType.high:
        rate = random.randint(150, 200)
    else:
        raise TypeError
    return {"heartRate": rate, "rateType": rate_type.value}


def generate(stream_name, kinesis_client, output=True):
    while True:
```

```
        rnd = random.random()
        rate_type = RateType.high if rnd < 0.01 else RateType.normal
        heart_rate = get_heart_rate(rate_type)
        if output:
            print(heart_rate)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(heart_rate),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gere um stream do Kinesis com pontos de acesso usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com hotspots.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
from pprint import pprint
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_hotspot(field, spot_size):
```

```
    hotspot = {
        "left": field["left"] + random.random() * (field["width"] - spot_size),
        "width": spot_size,
        "top": field["top"] + random.random() * (field["height"] - spot_size),
        "height": spot_size,
    }
    return hotspot


def get_record(field, hotspot, hotspot_weight):
    rectangle = hotspot if random.random() < hotspot_weight else field
    point = {
        "x": rectangle["left"] + random.random() * rectangle["width"],
        "y": rectangle["top"] + random.random() * rectangle["height"],
        "is_hot": "Y" if rectangle is hotspot else "N",
    }
    return {"Data": json.dumps(point), "PartitionKey": "partition_key"}


def generate(
    stream_name, field, hotspot_size, hotspot_weight, batch_size, kinesis_client
):
    """
    Generates points used as input to a hotspot detection algorithm.
    With probability hotspot_weight (20%), a point is drawn from the hotspot;
    otherwise, it is drawn from the base field. The location of the hotspot
    changes for every 1000 points generated.
    """
    points_generated = 0
    hotspot = None
    while True:
        if points_generated % 1000 == 0:
            hotspot = get_hotspot(field, hotspot_size)
        records = [
            get_record(field, hotspot, hotspot_weight) for _ in range(batch_size)
        ]
        points_generated += len(records)
        pprint(records)
        kinesis_client.put_records(StreamName=stream_name, Records=records)

        time.sleep(0.1)


if __name__ == "__main__":
```

```
    generate(
        stream_name=STREAM_NAME,
        field={"left": 0, "width": 10, "top": 0, "height": 10},
        hotspot_size=1,
        hotspot_weight=0.2,
        batch_size=10,
        kinesis_client=boto3.client("kinesis"),
    )
```

Gere um stream do Kinesis com entradas de registro usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com entradas de log.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "
        '"GET /index.php HTTP/1.1" 200 125 "-" '
        '"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0"'
    }


def generate(stream_name, kinesis_client):
    while True:
```

```
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gere um stream do Kinesis com dados escalonados usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dados escalonados.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import datetime
import json
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    event_time = datetime.datetime.utcnow() - datetime.timedelta(seconds=10)
    return {
        "EVENT_TIME": event_time.isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
```

```
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        # Send six records, ten seconds apart, with the same event time and
 ticker
        for _ in range(6):
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey",
            )
            time.sleep(10)


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gere um stream do Kinesis com dados de ações usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dados de cotação da bolsa.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import datetime
import json
import random
```

```
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gere um stream do Kinesis com dois tipos de dados usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dois tipos de dados.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import random
import boto3

STREAM_NAME = "OrdersAndTradesStream"
PARTITION_KEY = "partition_key"


def get_order(order_id, ticker):
    return {
        "RecordType": "Order",
        "Oid": order_id,
        "Oticker": ticker,
        "Oprice": random.randint(500, 10000),
        "Otype": "Sell",
    }


def get_trade(order_id, trade_id, ticker):
    return {
        "RecordType": "Trade",
        "Tid": trade_id,
        "Toid": order_id,
        "Tticker": ticker,
        "Tprice": random.randint(0, 3000),
    }


def generate(stream_name, kinesis_client):
    order_id = 1
    while True:
        ticker = random.choice(["AAAA", "BBBB", "CCCC"])
        order = get_order(order_id, ticker)
        print(order)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(order),
 PartitionKey=PARTITION_KEY
        )
        for trade_id in range(1, random.randint(0, 6)):
            trade = get_trade(order_id, trade_id, ticker)
            print(trade)
            kinesis_client.put_record(
```

```
                StreamName=stream_name,
                Data=json.dumps(trade),
                PartitionKey=PARTITION_KEY,
            )
        order_id += 1


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gere um stream do Kinesis com dados de log da web usando um AWS SDK

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dados de log da web.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "log": "192.168.254.30 - John [24/May/2004:22:01:02 -0700] "
        '"GET /icons/apache_pb.gif HTTP/1.1" 304 0'
    }


def generate(stream_name, kinesis_client):
    while True:
```

```
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Exemplos de código para AWS Marketplace API de catálogo usando AWS SDKs

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com um kit AWS de desenvolvimento de software (SDK).

Mais atributos

- AWS Marketplace Guia do vendedor da API de catálogo — Mais informações sobre a API de AWS Marketplace catálogo.
- AWS Marketplace Referência da API do Catálogo — Detalhes sobre todas as ações disponíveis da API do AWS Marketplace Catálogo.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Produtos AMI para AWS Marketplace API do Catalog
  - Adicione uma dimensão a um produto AMI existente e atualize os termos de preços da oferta usando um AWS SDK
  - Adicione uma região em que um produto AMI é implantado usando um AWS SDK
  - Crie um produto AMI público ou limitado e uma oferta pública com preços anuais por hora usando um AWS SDK

- Crie um produto AMI público ou limitado e uma oferta pública com preços mensais por hora usando um AWS SDK
- Crie um produto de AMI público ou limitado e uma oferta pública com preços por hora usando um AWS SDK
- Crie um rascunho de produto AMI com um rascunho de oferta pública usando um AWS SDK
- Restrinja uma região em que um produto AMI é implantado usando um AWS SDK
- Restrinja a visibilidade do produto usando um AWS SDK
- Especifique se os ativos da AMI são implantados em novas regiões usando um AWS SDK

- Ofertas de parceiros de canal para AWS Marketplace API do Catalog
  - Crie um rascunho de CPPO para qualquer tipo de produto usando um AWS SDK
  - Crie uma oferta privada de substituição de autorização de revenda a partir de um contrato existente com preços contratuais usando um AWS SDK
  - Liste todos os CPPOs criados por um parceiro de canal usando um AWS SDK
  - Liste todas as autorizações de revenda compartilhadas disponíveis para um parceiro de canal usando um AWS SDK
  - Publique um CPPO e anexe um EULA do comprador usando um AWS SDK
  - Publique um CPPO usando uma autorização de revenda única e atualize a marcação de preço usando um AWS SDK
  - Publique um rascunho de CPPO e atualize a marcação de preço usando um AWS SDK
  - Atualize a data de expiração de um CPPO usando um AWS SDK
- Produtos de contêiner para AWS Marketplace API do Catalog
  - Crie um rascunho de produto de contêiner com um rascunho de oferta pública usando um AWS SDK
  - Crie um produto de contêiner limitado com oferta pública, contrate preços usando um AWS SDK
- Entidades para AWS Marketplace API do Catalog
  - Descreva todas as entidades em uma única chamada usando um AWS SDK
  - Liste e descreva todas as ofertas associadas a um produto usando um AWS SDK
- Ofertas para AWS Marketplace API do Catalog
  - Crie uma dimensão personalizada para um produto SaaS e crie uma oferta privada usando um AWS SDK
  - Crie um rascunho de oferta privada para um produto AMI ou SaaS usando um AWS SDK

- Crie uma oferta privada com contrato e Pay-As-You-Go preços para um produto SaaS usando um AWS SDK
- Crie uma oferta privada com preços contratuais e um cronograma de pagamento flexível para um produto SaaS usando um AWS SDK
- Crie uma oferta privada com preços contratuais para um produto Container usando um AWS SDK
- Crie uma oferta privada com preços contratuais para um produto AMI usando um AWS SDK
- Crie uma oferta privada com preços anuais por hora e um cronograma de pagamento flexível para um produto AMI usando um AWS SDK
- Crie uma oferta privada com preços anuais por hora para um produto AMI usando um AWS SDK
- Crie uma oferta privada com preços por hora para um produto AMI usando um AWS SDK
- Crie uma oferta privada com preços de assinatura para um produto SaaS usando um AWS SDK
- Crie uma oferta privada com preços contratuais diferenciados para um produto SaaS usando um AWS SDK
- Crie uma oferta pública de teste gratuito com preços de assinatura para um produto SaaS usando um AWS SDK
- Crie uma oferta privada substituta a partir de um contrato existente com preços contratuais usando um AWS SDK
- Descreva uma oferta pública usando um AWS SDK
- Expirar uma oferta privada usando um AWS SDK
- Liste todas as ofertas privadas usando um AWS SDK
- Liste as ofertas públicas e privadas lançadas para um ID de produto específico usando um AWS SDK
- Atualize uma oferta para aplicar um contrato com Pay-As-You-Go preços usando um AWS SDK
- Atualize uma oferta para aplicar preços anuais por hora usando um AWS SDK
- Atualize uma oferta para aplicar a segmentação a regiões geográficas específicas usando um AWS SDK
- Atualize o nome e a descrição de uma oferta pública usando um AWS SDK
- Atualize o EULA de uma oferta usando um AWS SDK
- Atualize a data de expiração de uma oferta privada para uma data futura usando um AWS SDK
- Atualize a duração do teste gratuito de uma oferta pública de teste gratuito para um produto SaaS usando um AWS SDK

- Atualize a política de reembolso de uma oferta usando um AWS SDK
- Produtos para AWS Marketplace API do Catalog
  - Descreva um produto AMI, SaaS ou contêiner usando um AWS SDK
  - Liste todos os produtos AMI, SaaS ou Container e ofertas públicas associadas usando um AWS SDK
- Autorização de revenda para AWS Marketplace API do Catalog
  - Crie um rascunho de autorização de revenda para qualquer tipo de produto usando um AWS SDK
  - Descreva uma autorização de revenda usando um AWS SDK
  - Publique uma autorização de revenda única com uma oferta privada usando um AWS SDK
  - Publique uma autorização de revenda multiuso com uma data de validade usando um AWS SDK
  - Publique uma autorização de revenda multiuso com uma data de validade e um EULA a ser enviado ao comprador usando um AWS SDK
  - Publique a autorização de revenda multiuso com uma data de validade e adicione a documentação do contrato de revendedor usando um AWS SDK
  - Publique a autorização de revenda multiuso com expiração e adicione uma conta de comprador específica para a revenda usando um AWS SDK
  - Publique a autorização de revenda multiuso sem data de validade usando um AWS SDK
  - Publique a autorização de revenda multiuso sem uma data de validade e adicione um EULA personalizado usando um AWS SDK
  - Publique a autorização de revenda multiuso sem uma data de expiração e adicione a documentação do contrato de revendedor usando um AWS SDK
  - Publique a autorização de revenda multiuso sem data de validade e adicione uma conta de comprador específica para a revenda usando um AWS SDK
  - Publique uma autorização de revenda única e adicione um cronograma de pagamento flexível usando um AWS SDK
  - Publique uma autorização de revenda única para qualquer tipo de produto e adicione um EULA usando um AWS SDK
  - Publique uma autorização de revenda única e adicione uma conta de comprador específica para a revenda usando um AWS SDK
  - Publique uma autorização de revenda única para qualquer tipo de produto e adicione a documentação do contrato de revendedor usando um AWS SDK

- Publique uma autorização de revenda única e adicione se é uma renovação usando um AWS SDK
- Restrinja a autorização de revenda usando um AWS SDK
- Atualize o nome e a descrição da autorização de revenda única ou multiuso usando um AWS SDK
- Produtos SaaS para AWS Marketplace API do Catalog
  - Crie um rascunho de produto SaaS com um rascunho de oferta pública usando um AWS SDK
  - Crie um produto SaaS público ou limitado e uma oferta pública com preços contratuais usando um AWS SDK
  - Crie um produto SaaS público ou limitado e uma oferta pública com contrato e Pay-As-You-Go preços usando um AWS SDK
  - Crie um produto SaaS público ou limitado e uma oferta pública com preços de assinatura usando um AWS SDK
  - Publique um produto SaaS e uma oferta pública associada usando um AWS SDK
  - Publique um produto SaaS e uma oferta pública associada a partir de um rascunho existente usando um AWS SDK
  - Atualize as dimensões em um produto AMI ou SaaS usando um AWS SDK
- Utilitários para AWS Marketplace API do Catalog
  - Utilitários para iniciar um conjunto de alterações usando um AWS SDK

Produtos AMI para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Adicione uma dimensão a um produto AMI existente e atualize os termos de preços da oferta usando um AWS SDK
- Adicione uma região em que um produto AMI é implantado usando um AWS SDK
- Crie um produto AMI público ou limitado e uma oferta pública com preços anuais por hora usando um AWS SDK
- Crie um produto AMI público ou limitado e uma oferta pública com preços mensais por hora usando um AWS SDK

- Crie um produto de AMI público ou limitado e uma oferta pública com preços por hora usando um AWS SDK
- Crie um rascunho de produto AMI com um rascunho de oferta pública usando um AWS SDK
- Restrinja uma região em que um produto AMI é implantado usando um AWS SDK
- Restrinja a visibilidade do produto usando um AWS SDK
- Especifique se os ativos da AMI são implantados em novas regiões usando um AWS SDK

Adicione uma dimensão a um produto AMI existente e atualize os termos de preços da oferta usando um AWS SDK

Os exemplos de código a seguir mostram como adicionar uma dimensão a um produto de AMI existente e atualizar os termos de preços da oferta.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
```

```
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
```

```
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
```

```
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to add a dimension to an existing
 AMI product and update the offer pricing terms.
CAPI-23
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Add dimension for AMI product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Adicione uma região em que um produto AMI é implantado usando um AWS SDK

Os exemplos de código a seguir mostram como adicionar uma região em que um produto de AMI é implantado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to add a region where my
AMI product is deployed
```

```
CAPI-25A
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Add a region where my AMI product is deployed",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um produto AMI público ou limitado e uma oferta pública com preços anuais por hora usando um AWS SDK

Os exemplos de código a seguir mostram como criar um produto de AMI público ou limitado e uma oferta pública com preços anuais calculados por hora. Este exemplo cria um EULA padrão ou personalizado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
```

```
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
```

```
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
```

```
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [],
                "AdditionalResources": [],
                "SupportDescription": "For support, contact support@example.com"
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
```

```
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
```

```
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
```

```
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public or limited AMI
product and public offer with hourly-annual pricing and standard or custom EULA
CAPI-06
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited AMI product and public offer with hourly-annual pricing
 and standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um produto AMI público ou limitado e uma oferta pública com preços mensais por hora usando um AWS SDK

Os exemplos de código a seguir mostram como criar um produto de AMI público ou limitado e uma oferta pública com preços mensais calculados por hora. Este exemplo cria um EULA padrão ou personalizado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
```

```
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
```

```
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
```

```
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "RecurringPaymentTerm",
                        "CurrencyCode": "USD",
                        "BillingPeriod": "Monthly",
                        "Price": "15.0"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
```

```
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
```

```
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
```

```
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
```

```
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "RecurringPaymentTerm",
                        "CurrencyCode": "USD",
                        "BillingPeriod": "Monthly",
                        "Price": "15.0"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public or limited AMI
product and public offer with hourly-monthly pricing and standard or custom EULA
CAPI-08
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "create limited AMI product and public offer with hourly-monthly pricing
 and standard EULA",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um produto de AMI público ou limitado e uma oferta pública com preços por hora usando um AWS SDK

Os exemplos de código a seguir mostram como criar um produto de AMI público ou limitado e uma oferta pública com preços por hora. Este exemplo cria um EULA padrão ou personalizado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
```

```
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
```

```
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
```

```
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
```

```
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
```

```
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
```

```
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
```

```
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
```

```
                "Description": "Test public offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to Create a public or limited AMI
 product
and public offer with hourly pricing and standard or custom EULA
CAPI-07
```

```
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited AMI product and public offer with hourly pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um rascunho de produto AMI com um rascunho de oferta pública usando um AWS SDK

Os exemplos de código a seguir mostram como criar um rascunho de produto de AMI com um rascunho de oferta pública.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an AMI draft product
with a draft public offer.
CAPI-02
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "AMI draft product with draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Restrinja uma região em que um produto AMI é implantado usando um AWS SDK

Os exemplos de código a seguir mostram como restringir uma região onde um produto de AMI é implantado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to restrict a region where my
AMI product is deployed
CAPI-25B
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Add a region where my AMI product is deployed",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Restrinja a visibilidade do produto usando um AWS SDK

Os exemplos de código a seguir mostram como restringir a visibilidade do produto.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to change a product visibility to
 restricted
CAPI-17
"""

import os

import utils.start_changeset as sc
```

```
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Restrict existing AMI")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Especifique se os ativos da AMI são implantados em novas regiões usando um AWS SDK

Os exemplos de código a seguir mostram como especificar se os ativos da AMI são implantados em novas regiões criadas AWS para oferecer suporte a futuras regiões.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
```

```
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to modify a product to support
 all future regions
CAPI-26
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update future region support")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Ofertas de parceiros de canal para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Crie um rascunho de CPPO para qualquer tipo de produto usando um AWS SDK
- Crie uma oferta privada de substituição de autorização de revenda a partir de um contrato existente com preços contratuais usando um AWS SDK
- Liste todos os CPPOs criados por um parceiro de canal usando um AWS SDK
- Liste todas as autorizações de revenda compartilhadas disponíveis para um parceiro de canal usando um AWS SDK
- Publique um CPPO e anexe um EULA do comprador usando um AWS SDK
- Publique um CPPO usando uma autorização de revenda única e atualize a marcação de preço usando um AWS SDK
- Publique um rascunho de CPPO e atualize a marcação de preço usando um AWS SDK
- Atualize a data de expiração de um CPPO usando um AWS SDK

Crie um rascunho de CPPO para qualquer tipo de produto usando um AWS SDK

Os exemplos de código a seguir mostram como criar um rascunho de CPPO para qualquer tipo de produto a fim de analisá-lo internamente antes de publicar para os compradores.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer",
                "Description": "Test product"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer name"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” CPPO
for any product type (AMI/SaaS/Container) that can be reviewed internally
before publishing to buyers
CAPI-60
"""
import os

import utils.start_changeset as sc  # noqa: E402
import utils.stringify_details as sd  # noqa: E402

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Create a draft CPPO offer for a product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada de substituição de autorização de revenda a partir de um contrato existente com preços contratuais usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada de substituição de autorização de revenda com base em um contrato existente com preços contratuais.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
 "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test replacement offer for SaaSProduct using AWS
Marketplace API Reference Codes",
                "Description": "Test private resale replacement offer with
contract pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
 "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
 "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
Marketplace API Reference Codes",
                "Description": "Test private resale replacement offer with
contract pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
```

```
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a resale authorization
 replacement private offer
from an existing agreement with contract pricing
CAPI-96
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create resale authorization replacement private offer with contract
 pricing",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Liste todos os CPPOs criados por um parceiro de canal usando um AWS SDK

Os exemplos de código a seguir mostram como listar todos os CPPOs criados por um parceiro de canal.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
```

```
public class ListAllCppoOffers {

 /*
  * List all CPPOs created by a channel partner
  */
 public static void main(String[] args) {

  List<String> cppoOfferIds = getAllCppoOfferIds();

  ReferenceCodesUtils.formatOutput(cppoOfferIds);
 }

 public static List<String> getAllCppoOfferIds() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all offer entity ids
  List<String> entityIdList = new ArrayList<String>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
   entityIdList.add(entitySummary.entityId());
  }

  while (listEntitiesResponse.nextToken() != null) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
```

```
      .nextToken(listEntitiesResponse.nextToken())
      .build();
    listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

    for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
     entityIdList.add(entitySummary.entityId());
    }
   }

   // filter for CPPO offers: ResaleAuthorizationId exists in Details

   List<String> cppoOfferIds = new ArrayList<String>();

   for (String entityId : entityIdList) {
    DescribeEntityRequest describeEntityRequest =
      DescribeEntityRequest.builder()
      .catalog(AWS_MP_CATALOG)
      .entityId(entityId)
      .build();
    DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);

    Document resaleAuthorizationDocument =
 describeEntityResponse.detailsDocument().asMap().get(ATTRIBUTE_RESALE_AUTHORIZATION_ID);
    String resaleAuthorizationId = resaleAuthorizationDocument != null ?
 resaleAuthorizationDocument.asString() : "";

    if (!resaleAuthorizationId.isEmpty()) {
        cppoOfferIds.add(resaleAuthorizationId);
    }
   }
   return cppoOfferIds;
 }

}
```

- Para obter detalhes da API, consulte ListEntitiesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to list all Channel Partner
 Offers
in an account

Program executed with no arguments:
ie. python3 list_all_cppo_offers.py

CAPI-93
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-catalog")


def get_offer_entities():
    """
    Returns a list of all offers in the account
    """

    next_token = ""  # nosec: B105
    response_list = []
```

```
    try:
        response = mp_client.list_entities(Catalog="AWSMarketplace",
 EntityType="Offer")
    except ClientError as e:
        logging.exception(f"Couldn't list entities. {e}")
        raise

    response_list.append(response)

    # Results are paginated depending on number of entities returned
    while "NextToken" in response:
        next_token = response["NextToken"]

        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType="Offer",
                NextToken=next_token,
            )
        except ClientError as e:
            logging.exception(f"Couldn't list entities. {e}")
            raise

        if "NextToken" in response:
            response_list.append(response)

    return response_list


def build_offer_list(response_list):
    """
    Cleans up list_entities response list with just list of offer IDs
    """
    offer_list = []

    for response in response_list:
        for entity in response["EntitySummaryList"]:
            offer_list.append(entity["EntityId"])

    return offer_list


def check_offer_resaleauth(offer_id):
    """
```

```
    Checks to see if an offer is based on a resale authorization
    """
    offer_response = describe_entity(offer_id)
    offer_details = json.loads(offer_response["Details"])
    if offer_details is None:
        offer_details = offer_response["DetailsDocument"]
    if "ResaleAuthorizationId" in offer_details and
 offer_details["ResaleAuthorizationId"] is not None:
        return offer_id
    else:
        return None


def describe_entity(entity_id):
    """
    General purpose describe entity call
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )
    except ClientError as e:
        logging.exception(f"Couldn't describe entity. {e}")
        raise

    return response


def get_resaleauth_offers():
    """
    Returns a list of all offers in the account that are
    based on a resale authorization
    """
    resale_offer_list = []

    response_list = get_offer_entities()
    offer_list = build_offer_list(response_list)
    for offer in offer_list:
        print ("offer id " + offer)
        offer_info = check_offer_resaleauth(offer)
        if offer_info is not None:
            resale_offer_list.append(offer_info)
```

```
    return resale_offer_list


if __name__ == "__main__":
    print(get_resaleauth_offers())
```

- Para obter detalhes da API, consulte a ListEntitiesReferência da API AWS SDK for Python (Boto3).

Liste todas as autorizações de revenda compartilhadas disponíveis para um parceiro de canal usando um AWS SDK

Os exemplos de código a seguir mostram como listar todas as autorizações de revenda compartilhadas disponíveis para um parceiro de canal.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllSharedResaleAuthorizations {

 /*
  * list all resale authorizations shared to an account
  */
 public static void main(String[] args) {

  List<ListEntitiesResponse> responseList = getListEntityResponseList();
  ReferenceCodesUtils.formatOutput(responseList);
 }

 public static List<ListEntitiesResponse> getListEntityResponseList() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  List<ListEntitiesResponse> responseList = new
 ArrayList<ListEntitiesResponse>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
    .maxResults(10)
    .ownershipType(OWNERSHIP_TYPE_SHARED)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  responseList.add(listEntitiesResponse);

  while (listEntitiesResponse.nextToken() != null) {
   listEntitiesRequest = ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
     .maxResults(10)
     .ownershipType(OWNERSHIP_TYPE_SHARED)
```

```
      .nextToken(listEntitiesResponse.nextToken())
      .build();

    listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

    responseList.add(listEntitiesResponse);
   }
   return responseList;
 }

}
```

- Para obter detalhes da API, consulte ListEntitiesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to list all resale authorizations
shared to an account

Program executed with no arguments:
ie. python3 list_all_resale_authorizations.py

CAPI-94
"""

import logging
```

```
import boto3
import utils.helpers as hlp  # noqa: E402
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-catalog")


def get_shared_entities():
    next_token = ""  # nosec: B105
    response_list = []

    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType="ResaleAuthorization",
            OwnershipType="SHARED",
        )
    except ClientError as e:
        logging.exception(f"Couldn't list entities. {e}")
        raise

    response_list.append(response)

    # Results can be paginated depending on number of entities returned
    while "NextToken" in response:
        next_token = response["NextToken"]

        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType="ResaleAuthorization",
                OwnershipType="SHARED",
                NextToken=next_token,
            )
        except ClientError as e:
            logging.exception(f"Couldn't list entities. {e}")
            raise

        if "NextToken" in response:
            response_list.append(response)

    return response_list
```

```
if __name__ == "__main__":
    response_list = get_shared_entities()
    hlp.pretty_print_datetime(response_list)
```

- Para obter detalhes da API, consulte a ListEntitiesReferência da API AWS SDK for Python (Boto3).

Publique um CPPO e anexe um EULA do comprador usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma CPPO e acrescentar um EULA do comprador.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
                {
                 "Type": "CustomEula",
                 "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                }
               ]
           }
       ]
         }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
```

```
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
                {
                 "Type": "CustomEula",
                 "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                }
               ]
           }
       ]
         }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
```

```
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to publish CPPO
for any product type (AMI/SaaS/Container) and append buyer EULA
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Publish CPPO for any product type and append buyer EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique um CPPO usando uma autorização de revenda única e atualize a marcação de preço usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma CPPO usando autorização única de revenda em produtos de AMI, de SaaS ou de contêiner e atualizar a margem de preço.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPO",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test Offer name",
                "Description":"Test Offer description"
            }
        },
        {
```

```
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["111111111111"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
                }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to Create CPPO using one-time
 resale
authorization on AMI, SaaS or Container products and update price markup
CAPI-63
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create CPPO using one-time resale authorization and update price markup"
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique um rascunho de CPPO e atualize a marcação de preço usando um AWS SDK

Os exemplos de código a seguir mostram como publicar um rascunho de CPPO e atualizar a margem de preço.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPO",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test Offer name",
                "Description":"Test Offer description"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["111111111111"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to publish “draft” CPPO
for any product type (AMI/SaaS/Container) and update price markup
CAPI-72
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)
```

```
    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Publish draft CPPO for any product type adn update price markup"
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize a data de expiração de um CPPO usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar a data de validade de uma CPPO para dar aos compradores mais tempo para avaliar e aceitar a oferta.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
```

```
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update the expiry
date of a CPPO offer
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Update the expiry date of a CPPO offer"
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Produtos de contêiner para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Crie um rascunho de produto de contêiner com um rascunho de oferta pública usando um AWS SDK
- Crie um produto de contêiner limitado com oferta pública, contrate preços usando um AWS SDK

Crie um rascunho de produto de contêiner com um rascunho de oferta pública usando um AWS SDK

Os exemplos de código a seguir mostram como criar um rascunho de produto de contêiner com um rascunho de oferta pública.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog":"AWSMarketplace",
    "changeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
```

```
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
```

```
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an container draft
 product
with a draft public offer.
CAPI-03
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set
```

```
    response = sc.usage_demo(
        stringified_change_set,
        "Create a draft container product with a draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um produto de contêiner limitado com oferta pública, contrate preços usando um AWS SDK

Os exemplos de código a seguir mostram como criar um produto de contêiner limitado com uma oferta pública, preços contratuais e EULA padrão.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
```

```
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
                    "Name": "Cores",
                    "Types": [
```

```
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
```

```
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
                "Description": "Some interesting container offer description"
            }
```

```
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
                    "Name": "Cores",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
```

```
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
                "Description": "Some interesting container offer description"
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create limited container
product with public offer, contract pricing and standard EULA
CAPI-15
```

```
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited container product with public offer contract pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Entidades para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Descreva todas as entidades em uma única chamada usando um AWS SDK

- Liste e descreva todas as ofertas associadas a um produto usando um AWS SDK

Descreva todas as entidades em uma única chamada usando um AWS SDK

Os exemplos de código a seguir mostram como descrever todas as entidades em uma única chamada.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesRequest;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityDetail;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeErrorDetail;

import java.util.Arrays;
import java.util.Map;

public class BatchDescribeEntities {
```

```
    /*
     * BatchDescribe my entities in a single call and
     *  check if it contains all the information I need to know about the
entities.
     */
    public static void main(String[] args) {

        MarketplaceCatalogClient marketplaceCatalogClient =
                MarketplaceCatalogClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        BatchDescribeEntitiesRequest batchDescribeEntitiesRequest =
                BatchDescribeEntitiesRequest.builder()
                        .entityRequestList(Arrays.asList(
                                EntityRequest.builder()

.catalog(AWS_MP_CATALOG).entityId(OFFER_ID)
                                        .build(),
                                EntityRequest.builder()

.catalog(AWS_MP_CATALOG).entityId(PRODUCT_ID)
                                        .build()))
                        .build();

        BatchDescribeEntitiesResponse batchDescribeEntitiesResponse =
marketplaceCatalogClient.batchDescribeEntities(batchDescribeEntitiesRequest);

        // Reading the successful entities response
        Map<String, EntityDetail> entityDetailsMap =
batchDescribeEntitiesResponse.entityDetails();
        for (Map.Entry<String, EntityDetail> entry : entityDetailsMap.entrySet())
{
            System.out.println("EntityId: " + entry.getKey());
            ReferenceCodesUtils.formatOutput(entry.getValue());
        }

        // Logging the failed entities error details
        Map<String, BatchDescribeErrorDetail> entityErrorsMap =
batchDescribeEntitiesResponse.errors();
        for (Map.Entry<String, BatchDescribeErrorDetail> entry :
entityErrorsMap.entrySet()) {
```

```
            System.out.println(String.format("EntityId: %s, ErrorCode: %s,
 ErrorMessage: %s", entry.getKey(),
                    entry.getValue().errorCode(),
 entry.getValue().errorMessage()));
        }
    }
}
```

- Para obter detalhes da API, consulte BatchDescribeEntiticsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to describe for multiple entities
 information in the AWS Marketplace Catalog
CAPI-98
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

PRODUCT_ID = "prod-1111111111111"
OFFER_ID = "offer-1111111111111"
MARKETPLACE_CATALOG = "AWSMarketplace"
```

```
def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_entities_information(mp_client):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    entity_request_list_param = [
        {'EntityId': PRODUCT_ID, 'Catalog': MARKETPLACE_CATALOG},
        {'EntityId': OFFER_ID, 'Catalog': MARKETPLACE_CATALOG}
    ]
    try:
        response = mp_client.batch_describe_entities(
            EntityRequestList=entity_request_list_param
        )

        return response

    except ClientError as e:
        logger.exception("Unexpected error: %s", e)
        raise


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

    print("-" * 88)
    print("Looking for entities in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    response = get_entities_information(mp_client)
    print("Successful entities response -")
    pretty_print(response["EntityDetails"])
    print("Failed entities response -")
    pretty_print(response["Errors"])
```

```
if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a BatchDescribeEntitiesReferência da API AWS SDK for Python (Boto3).

Liste e descreva todas as ofertas associadas a um produto usando um AWS SDK

Os exemplos de código a seguir mostram como listar e descrever todas as ofertas associadas a um produto.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
```

```
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPrivateOffers {

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();
 /*
  * retrieve all private offer information related to a single product
  */
 public static void main(String[] args) {

  List<EntitySummary> entitySummaryList = getEntitySummaryList();

  // for each offer id, output the offer detail using DescribeEntity API

  for (EntitySummary entitySummary : entitySummaryList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entitySummary.entityId())
     .build();
   DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   ReferenceCodesUtils.formatOutput(describeEntityResponse);
  }
 }
 public static List<EntitySummary> getEntitySummaryList() {
  // define list entities filters

  EntityTypeFilters entityTypeFilters =
```

```
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(PRODUCT_ID)
        .build())
      .build())
    .build();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER).maxResults(50)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER).maxResults(50)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }
  return entitySummaryList;
 }
```

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DescribeEntity
  - ListEntities

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to retrieve all offer information
related to a single product
CAPI-97
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError
from utils import helpers

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-catalog")


def get_entity_information(entity_id):
    """
```

```
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def list_entity_details(entity_type, entity_id):
    """
    Returns details about a given entity and entity type
    """

    entity_summary_list = []

    # filter will return details for given entity_id with BuyerAccounts targeting
    filter_list_param = {
        'OfferFilters':{
            'ProductId':{
                'ValueList':[entity_id]
            },
            'Targeting': {
                'ValueList': ["BuyerAccounts"]
            }
        }
    }

    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType=entity_type,
            EntityTypeFilters = filter_list_param,
```

```
            MaxResults=10
        )

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)

    # add results to entity_summary_list
    entity_summary_list.extend(response["EntitySummaryList"])

    # if there are more than 10 offers, paginate through the results
    while "NextToken" in response and response["NextToken"] is not None:
        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType=entity_type,
                EntityTypeFilters = filter_list_param,
                NextToken=response["NextToken"],
                MaxResults=10
            )

        except ClientError as e:
            if e.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Entity ID %s not found.", entity_id)
            else:
                logger.error("Unexpected error: %s", e)

        # add results to entity_summary_list
        entity_summary_list.extend(response["EntitySummaryList"])

        return entity_summary_list

if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "--entity-id",
        "-eid",
        help="Provide Entity ID corresponding to a product to filter offers on",
        required=True,
    )
```

```
    args = parser.parse_args()

    # Gets a offers associated with the entity_id
    response = list_entity_details(
        "Offer",
        entity_id=args.entity_id
    )

    if response: # if response is not empty

        # list_entity_details returns a list of offers
        for offer in response:

            print("-"*128)
            print(f"Terms for Offer ID: {offer['EntityId']}")
            print("-"*128)

            #retrieve offer information for each offer
            entity_information = get_entity_information(offer["EntityId"])

            helpers.pretty_print_datetime(entity_information)

    else:
        print(f"No information found for Entity ID: {args.entity_id}")
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DescribeEntity
  - ListEntities

Ofertas para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Crie uma dimensão personalizada para um produto SaaS e crie uma oferta privada usando um AWS SDK
- Crie um rascunho de oferta privada para um produto AMI ou SaaS usando um AWS SDK

- Crie uma oferta privada com contrato e Pay-As-You-Go preços para um produto SaaS usando um AWS SDK
- Crie uma oferta privada com preços contratuais e um cronograma de pagamento flexível para um produto SaaS usando um AWS SDK
- Crie uma oferta privada com preços contratuais para um produto Container usando um AWS SDK
- Crie uma oferta privada com preços contratuais para um produto AMI usando um AWS SDK
- Crie uma oferta privada com preços anuais por hora e um cronograma de pagamento flexível para um produto AMI usando um AWS SDK
- Crie uma oferta privada com preços anuais por hora para um produto AMI usando um AWS SDK
- Crie uma oferta privada com preços por hora para um produto AMI usando um AWS SDK
- Crie uma oferta privada com preços de assinatura para um produto SaaS usando um AWS SDK
- Crie uma oferta privada com preços contratuais diferenciados para um produto SaaS usando um AWS SDK
- Crie uma oferta pública de teste gratuito com preços de assinatura para um produto SaaS usando um AWS SDK
- Crie uma oferta privada substituta a partir de um contrato existente com preços contratuais usando um AWS SDK
- Descreva uma oferta pública usando um AWS SDK
- Expirar uma oferta privada usando um AWS SDK
- Liste todas as ofertas privadas usando um AWS SDK
- Liste as ofertas públicas e privadas lançadas para um ID de produto específico usando um AWS SDK
- Atualize uma oferta para aplicar um contrato com Pay-As-You-Go preços usando um AWS SDK
- Atualize uma oferta para aplicar preços anuais por hora usando um AWS SDK
- Atualize uma oferta para aplicar a segmentação a regiões geográficas específicas usando um AWS SDK
- Atualize o nome e a descrição de uma oferta pública usando um AWS SDK
- Atualize o EULA de uma oferta usando um AWS SDK
- Atualize a data de expiração de uma oferta privada para uma data futura usando um AWS SDK
- Atualize a duração do teste gratuito de uma oferta pública de teste gratuito para um produto SaaS usando um AWS SDK
- Atualize a política de reembolso de uma oferta usando um AWS SDK

Crie uma dimensão personalizada para um produto SaaS e crie uma oferta privada usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma dimensão personalizada para um produto de SaaS e criar uma oferta privada.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
 defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
```

```
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
```

```
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
 defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a SaaS product custom
 dimension and private offer
CAPI-91
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "Create a SaaS product custom dimension and private offer"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um rascunho de oferta privada para um produto AMI ou SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como criar um rascunho de oferta privada para um produto de AMI ou de SaaS a fim de analisá-lo internamente antes de publicá-lo para os compradores.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” Private Offer
for any AMI or SAAS product type that can be reviewed internally
before publishing to buyers
CAPI-30
"""

import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Private offer for AMI product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com contrato e Pay-As-You-Go preços para um produto SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com contrato e Pay-As-You-Go preços para um produto SaaS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
```

```
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
```

```
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 contract with PAYG pricing for my SaaS product
CAPI-34
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create private offer with contract with PAYG pricing for my SaaS
 product",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com preços contratuais e um cronograma de pagamento flexível para um produto SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com preços contratuais e uma programação de pagamento flexível para um produto de SaaS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            },
                            {
                                "DimensionKey": "PremiumService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Some kind of refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer
with contract pricing and flexible payment schedule for my SaaS product
CAPI-39
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create private offer with contract pricing and flexible payment schedule
 for my SaaS product",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com preços contratuais para um produto Container usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com preços contratuais para um produto de contêiner.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
 Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
 contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
```

```
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
```

```
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer (target
 buyers)
for my Container product with contract pricing
CAPI-36
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "publish a private offer for my Container product with contract pricing",
    )
```

```
    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com preços contratuais para um produto AMI usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com preços contratuais para um produto de AMI.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
```

```
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
```

```
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
```

```
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
```

```
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 contract pricing for my AMI product
CAPI-35
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(
        change_set, "create private offer with contract pricing for my AMI
 product"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com preços anuais por hora e um cronograma de pagamento flexível para um produto AMI usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com preços anuais calculados por hora e uma programação de pagamento flexível para um produto de AMI.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
```

```
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
```

```
                        ]
                    },
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P365D",
                        "Grants": [
                            {
                                "DimensionKey": "t2.micro",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
```

```
                                {
                                    "ChargeDate": "2024-01-01",
                                    "ChargeAmount": "200.00"
                                },
                                {
                                    "ChargeDate": "2024-02-01",
                                    "ChargeAmount": "170.00"
                                }
                            ]
                        }
                    ]
                }
            },
            {
                "ChangeType": "ReleaseOffer",
                "Entity": {
                    "Type": "Offer@1.0",
                    "Identifier": "$CreateOfferChange.Entity.Identifier"
                },
                "DetailsDocument": {}
            }
        ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
```

```
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P365D",
                        "Grants": [
                            {
                                "DimensionKey": "t2.micro",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
```

```
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly annual pricing and flexible payment schedule for my AMI product
CAPI-XX
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create private offer with hourly annual pricing and flexible payment
 schedule for my AMI product",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com preços anuais por hora para um produto AMI usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com preços anuais calculados por hora para um produto de AMI.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
```

```
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
```

```
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-12-31"
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Usage",
            "Terms": [
                {
                    "Type": "UsageBasedPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "0.17"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
```

```
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly annual pricing for my AMI product
CAPI-31
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with hourly annual pricing for my AMI
 product"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com preços por hora para um produto AMI usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com preços por hora para um produto de AMI.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
 API Reference Code",
```

```
                "Description": "Test private offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-01-01"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
```

```
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-01-01"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly pricing for my AMI product
CAPI-32
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(
        change_set, "create private offer with hourly pricing for my AMI product"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com preços de assinatura para um produto SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com preços de assinatura para um produto de SaaS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
```

```
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
```

```
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 subscription pricing for my SaaS product
CAPI-33
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "Create private offer with subscription pricing for my SaaS
 product"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com preços contratuais diferenciados para um produto SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada com preços contratuais por níveis para um produto de SaaS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
```

```
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
```

```
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
```

```
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 tiered contract pricing for my SaaS product
CAPI-XX
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(
        change_set,
        "Create private offer with tiered contract pricing for my SaaS product",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta pública de teste gratuito com preços de assinatura para um produto SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta pública de teste gratuito com preços de assinatura de um produto de SaaS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
 Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
 pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
 Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
 pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public free trial
 offer with subscription pricing for SaaS product
CAPI-13
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create public free trial offer with subscription pricing for SaaS
 product",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie uma oferta privada substituta a partir de um contrato existente com preços contratuais usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma oferta privada de substituição com base em um contrato existente com preços contratuais.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
```

```
                "Description": "Test private replacement offer with contract
pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111"
```

```
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract
 pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a replacement private
 offer
from an existing agreement with contract pricing
CAPI-95
"""


import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create replacement private offer with contract pricing..",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Descreva uma oferta pública usando um AWS SDK

Os exemplos de código a seguir mostram como descrever uma oferta pública.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();
```

```
  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Para obter detalhes da API, consulte DescribeEntitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for offer information in
 the AWS Marketplace Catalog
CAPI-29
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

OFFER_ID = "offer-1111111111111"
```

```
def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_offer_information(mp_client, entity_id):
    """
    Returns information about a given offer
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of offer information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Offer with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for an offer in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_offer_information(mp_client, OFFER_ID))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeEntityReferência da API AWS SDK for Python (Boto3).

Expirar uma oferta privada usando um AWS SDK

Os exemplos de código a seguir mostram como definir a data de validade de uma oferta privada como uma data anterior para que os compradores não vejam mais a oferta.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-01-01"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” Private Offer
for any AMI or SAAS product type that can be reviewed internally
before publishing to buyers
CAPI-30
"""

import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Private offer for AMI product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Liste todas as ofertas privadas usando um AWS SDK

Os exemplos de código a seguir mostram como listar todas as ofertas privadas.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
```

```
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilterD
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferBuyerAccountsFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilterDateRange
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListAllPrivateOffers {

 /*
  * List all my private offers and sort or filter them by Offer Publish Date,
 Offer Expiry Date and Buyer IDs
  *
  * OfferTargetingFilter = BuyerAccounts (private offer);
  * OfferBuyerAccountsFilter: Buyer IDs filter
  * OfferAvailabilityEndDateFilter : Offer Expiry Date filter
  * OfferReleaseDateFilter : Offer Publish Date filter
  */

 private static MarketplaceCatalogClient marketplaceCatalogClient =
```

```
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

public static void main(String[] args) {

 String offerReleaseDateAfterValue = "2023-01-01T23:59:59Z";
 String offerAvailableEndDateAfterValue = "2040-12-24T23:59:59Z";

 List<EntitySummary> entitySummaryList =
getEntitySummaryList(offerReleaseDateAfterValue,
offerAvailableEndDateAfterValue);

 // for each offer id, output the offer detail using DescribeEntity API


 for (EntitySummary entitySummary : entitySummaryList) {
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(entitySummary.entityId())
    .build();
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}

public static List<EntitySummary> getEntitySummaryList (String
offerReleaseDateAfterValue, String offerAvailableEndDateAfterValue) {

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .buyerAccounts(OfferBuyerAccountsFilter.builder()
       .wildCardValue(BUYER_ACCOUNT_ID)
       .build())
     .availabilityEndDate(OfferAvailabilityEndDateFilter.builder()
       .dateRange(OfferAvailabilityEndDateFilterDateRange.builder()
         .afterValue(offerAvailableEndDateAfterValue).build())
```

```
        .build())
      .releaseDate(OfferReleaseDateFilter.builder()
        .dateRange(OfferReleaseDateFilterDateRange.builder()
          .afterValue(offerReleaseDateAfterValue)
          .build())
        .build())
      .build())
    .build();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER).maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }

  return entitySummaryList;
 }

}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) for listing offers in the AWS
 Marketplace Catalog
CAPI-40
"""
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# Constants
MAX_RESULTS = 10
CATALOG = "AWSMarketplace"
ENTITY_TYPE = "Offer"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)
```

```
def list_private_offers(mp_client, return_all_private_offers):
    """
    This method retrieves list of all Private Offers for this account.
    """
    entity_summary_list = []
    filter_list_param = {
        'OfferFilters': {
            'Targeting': {
                'ValueList': ["BuyerAccounts"]
            }
        }
    }
    try:
        response = mp_client.list_entities(
            Catalog=CATALOG,
            EntityType=ENTITY_TYPE,
            EntityTypeFilters=filter_list_param,
            MaxResults=MAX_RESULTS
        )
    except ClientError as e:
        logger.error("Could not complete list_entities request: %s", e)
        raise

    entity_summary_list.extend(response["EntitySummaryList"])
    logger.info("Number of results in first iteration: %d " %
 len(entity_summary_list))

    # Get subsequent pages of results if previous response contained a NextToken
    while "NextToken" in response and return_all_private_offers:
        try:
            logger.info("Getting Next Token results: %s " %
 response["NextToken"])
            response = mp_client.list_entities(
                Catalog=CATALOG,
                EntityType=ENTITY_TYPE,
                EntityTypeFilters=filter_list_param,
                MaxResults=MAX_RESULTS,
                NextToken=response["NextToken"]
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request: %s", e)
            raise

        entity_summary_list.extend(response["EntitySummaryList"])
```

```
        logger.info(
            "Number of results in the current iteration: %d "
            % len(response["EntitySummaryList"])
        )

    return entity_summary_list


def get_offer_details(mp_client, offer):
    """
    Describe the details of the Offer.
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace", EntityId=offer["EntityId"]
        )

        return response
    except ClientError:
        logger.exception("Error: Couldn't get details of the Offer.")
        raise


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

    print("-" * 88)
    print("Demo  - List Private offers.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    # Get list of all Offers.
    private_offers = list_private_offers(mp_client, False)
    count = len(private_offers)

    logger.info("Number of Offers: %d " % count)
    offer_counter = 0
    # Display details of each Offer.
    for offer in private_offers:
        print("-" * 88)
        offer_counter += 1
        print("Displaying Offer details for Offer# %d" % offer_counter)
        entity = get_offer_details(mp_client, offer)
```

```
        pretty_print(entity)

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Liste as ofertas públicas e privadas lançadas para um ID de produto específico usando um AWS SDK

O exemplo de código a seguir mostra como listar ofertas públicas e privadas lançadas para um ID de produto específico.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
```

```
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPublicOrPrivateReleasedOffers {

 /*
  * List released Public/Private offers for a specific product id.
  * Example below is to list released public offers.
  * To change to released private offers, change OFFER_TARGETING_NONE (None) to
 OFFER_TARGETING_BUYERACCOUNTS(BuyerAccounts)
  */
 public static void main(String[] args) {

  List<EntitySummary> entitySummaryList = getEntitySummaryLIst();
  ReferenceCodesUtils.formatOutput(entitySummaryList);
 }

 public static List<EntitySummary> getEntitySummaryLIst() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // define list entities filters

  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
```

```
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(PRODUCT_ID)
        .build())
      .build())
    .build();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }
  return entitySummaryList;
 }

}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Atualize uma oferta para aplicar um contrato com Pay-As-You-Go preços usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar uma oferta para aplicar um contrato com Pay-As-You-Go preços.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
```

```
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
```

```
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
```

```
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply
 contract with PAYG pricing
CAPI-21
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer to apply contract with PAYG pricing")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize uma oferta para aplicar preços anuais por hora usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar uma oferta para aplicar preços anuais calculados por hora.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
```

```
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply
 hourly annual pricing
CAPI-20
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer with hourly annual pricing")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize uma oferta para aplicar a segmentação a regiões geográficas específicas usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar uma oferta para aplicar segmentação a regiões geográficas específicas.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply
 targeting to certain geographic regions.
CAPI-19
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer targeting")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize o nome e a descrição de uma oferta pública usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar o nome e a descrição de uma oferta pública.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update name and description of
 my offer
CAPI-18
```

```
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update name and description of my offer")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize o EULA de uma oferta usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar o EULA de uma oferta.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update EULA of my offer
CAPI-18
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update EULA of my offer")
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize a data de expiração de uma oferta privada para uma data futura usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar a data de validade de uma oferta privada para uma data futura a fim de dar aos compradores mais tempo para avaliar e aceitar a oferta.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
        }
```

```
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to set expiry date of a private
 offer to a date in the future so that my buyers get more time to evaluate and
 accept the offer.
CAPI-37
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer expiration date")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize a duração do teste gratuito de uma oferta pública de teste gratuito para um produto SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar a duração do teste gratuito de uma oferta pública de teste gratuito de um produto de SaaS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to change free trial duration for
 my SaaS product by modifying my free trial public offer
CAPI-14
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Change free trial duration for SaaS product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize a política de reembolso de uma oferta usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar a política de reembolso de uma oferta.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update refund policy of my
 offer
```

```
CAPI-18
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update refund policy of my public offer")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Produtos para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Descreva um produto AMI, SaaS ou contêiner usando um AWS SDK
- Liste todos os produtos AMI, SaaS ou Container e ofertas públicas associadas usando um AWS SDK

Descreva um produto AMI, SaaS ou contêiner usando um AWS SDK

Os exemplos de código a seguir mostram como descrever um produto de AMI, de SaaS ou de contêiner e verificar se ele contém todas as informações que você deseja saber sobre o produto.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
```

```
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Para obter detalhes da API, consulte DescribeEntitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for product information in
 the AWS Marketplace Catalog
CAPI-28
"""

import json
```

```
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

PRODUCT_ID = "prod-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_product_information(mp_client, entity_id):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Product with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for a product in the AWS Marketplace Catalog.")
    print("-" * 88)
```

```
    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_product_information(mp_client, PRODUCT_ID))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeEntityReferência da API AWS SDK for Python (Boto3).

Liste todos os produtos AMI, SaaS ou Container e ofertas públicas associadas usando um AWS SDK

Os exemplos de código a seguir mostram como listar todos os produtos de AMI, de SaaS ou de contêiner e ofertas públicas associadas.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListEntities {

 /*
  * List all my AMI or SaaS or Container products and associated public offers
  */
 public static void main(String[] args) {

  Map<String, List<EntitySummary>> allProductsWithOffers =
 getAllProductsWithOffers();

  ReferenceCodesUtils.formatOutput(allProductsWithOffers);
 }

 public static Map<String, List<EntitySummary>> getAllProductsWithOffers() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  Map<String, List<EntitySummary>> allProductsWithOffers = new HashMap<String,
 List<EntitySummary>> ();

  // get all product entities
  List<EntitySummary> productEntityList = new ArrayList<EntitySummary>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
```

```
    .catalog(AWS_MP_CATALOG)
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
    .nextToken(null)
    .build();


 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 productEntityList.addAll(listEntitiesResponse.entitySummaryList());


 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
  productEntityList.addAll(listEntitiesResponse.entitySummaryList());
 }

 // loop through each product entity and get the public released offers
associated using product id filter

 for ( EntitySummary productEntitySummary : productEntityList) {
  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(productEntitySummary.entityId())
        .build())
      .build())
    .build();
```

```
    listEntitiesRequest =
      ListEntitiesRequest.builder()
      .catalog(AWS_MP_CATALOG)
      .entityType(ENTITY_TYPE_OFFER)
      .maxResults(10)
      .entityTypeFilters(entityTypeFilters)
      .nextToken(null)
      .build();

    listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);

    // save all entitySummary of the results into entitySummaryList

    List<EntitySummary> offerEntitySummaryList = new ArrayList<EntitySummary>();

    offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

    while ( listEntitiesResponse.nextToken() != null &&
  listEntitiesResponse.nextToken().length() > 0) {
     listEntitiesRequest =
       ListEntitiesRequest.builder()
       .catalog(AWS_MP_CATALOG)
       .entityType(ENTITY_TYPE_OFFER)
       .maxResults(10)
       .entityTypeFilters(entityTypeFilters)
       .nextToken(listEntitiesResponse.nextToken())
       .build();
     listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);
     offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
    }

    // save final results into map; key = product id; value = offer entity summary
  list

    allProductsWithOffers.put(productEntitySummary.entityId(),
  offerEntitySummaryList);
   }
   return allProductsWithOffers;
  }

 }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DescribeEntity
  - ListEntities

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to display information about AMI
 products and their associated offers in the AWS Marketplace Catalog
CAPI-27
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

try:
    mp_client = boto3.client("marketplace-catalog")
except ClientError as e:
    logger.error("Could not create boto3 client.")
    raise
```

```
def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def describe_entity(entity_id):
    """
    Returns entity details
    Args: entity_id str: The entity ID of the product or offer
    Returns: dict: The entity details
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace", EntityId=entity_id
        )
    except ClientError as e:
        logger.error("Could not complete describe_entity request.")
        raise

    # De-stringify the details
    response["Details"] = json.loads(response["Details"])

    return response


def get_entities(entity_type, visibility=None):
    """
    Returns list of entities for provided entity_type
    Args: entity_type str: Type of entity list to return, in our case AmiProduct
 or Offer
    Returns: list: Abbreviated list of entity information
    """
    EntitySummaryList = []

    # Get the first page of results
    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType=entity_type,
            MaxResults=MAX_PAGE_RESULTS,
        )
    except ClientError as e:
```

```
        logger.error("Could not complete list_entities request.")
        raise

    EntitySummaryList.extend(response["EntitySummaryList"])

    # Get subsequent pages of results if previous response contained a NextToken
    while "NextToken" in response:
        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType=entity_type,
                MaxResults=MAX_PAGE_RESULTS,
                NextToken=response["NextToken"],
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request.")
            raise

        EntitySummaryList.extend(response["EntitySummaryList"])

    # if visibility is provided, filter the list to only include entities with
 that visibility
    if visibility is not None:
        EntitySummaryList = [
            entity for entity in EntitySummaryList if entity["Visibility"] ==
 visibility
        ]

    return EntitySummaryList


def get_enhanced_product_list(entity_type):
    """
    Returns an enhanced list of products with product details and offer details
    Args: entity_type str: Type of entity list to return, in our case AmiProduct
    Returns: list: Enhanced list of dictionary objects containing product and
 offer details
    """

    product_list = get_entities(entity_type)

    # Loop through product list and append product details to each product
    for product in product_list:
        # appends product details to product dictionary
```

```
        product["ProductDetails"] = describe_entity(product["EntityId"])
["Details"]
        # creating an empty list for offer details
        product["OfferDetailsList"] = []

    return product_list


def attach_offer_details(product_list):
    """
    Loops through offer information and appends offer details to product list
    Args: product_list list: List of product dictionaries
    Returns: list: Enhanced list of dictionary objects containing product and
 offer details
    """
    offer_list = get_entities("Offer", "Public")

    # Loop through offer list and append offer details to each product
    for offer in offer_list:
        offer["OfferDetails"] = describe_entity(offer["EntityId"])["Details"]

        # Extracts product-id from offer
        product_id = offer["OfferDetails"]["ProductId"]

        # Determines if product-id referenced in offer matches product-id in
 product list
        product_dict = next(
            filter(lambda product: product["EntityId"] == product_id,
 product_list),
            None,
        )

        # If product-id matches, appends offer details to product dictionary
        if product_dict is not None:
            # logger.info(f"Offer product Id {offer['OfferDetails']['ProductId']}
 found in product dictionary. Updating product dictionary with offer details")
            product_dict["OfferDetailsList"].append(offer["OfferDetails"])

        else:
            # logger.info("Offer product Id {offer['OfferDetails']['ProductId']}
 not found. Skipping offer details update")
            pass

    return product_list
```

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

    print("-" * 88)
    print("Retrieving products and public offer information....")
    print("-" * 88)

    # Builds a list of products and their details
    product_list = get_enhanced_product_list("AmiProduct")

    # Queries offer information and attaches it to the product list
    product_offer_list = attach_offer_details(product_list)

    pretty_print(product_offer_list)
    return product_offer_list


if __name__ == "__main__":
    usage_demo()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DescribeEntity
  - ListEntities

Autorização de revenda para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Crie um rascunho de autorização de revenda para qualquer tipo de produto usando um AWS SDK
- Descreva uma autorização de revenda usando um AWS SDK
- Publique uma autorização de revenda única com uma oferta privada usando um AWS SDK
- Publique uma autorização de revenda multiuso com uma data de validade usando um AWS SDK

- Publique uma autorização de revenda multiuso com uma data de validade e um EULA a ser enviado ao comprador usando um AWS SDK
- Publique a autorização de revenda multiuso com uma data de validade e adicione a documentação do contrato de revendedor usando um AWS SDK
- Publique a autorização de revenda multiuso com expiração e adicione uma conta de comprador específica para a revenda usando um AWS SDK
- Publique a autorização de revenda multiuso sem data de validade usando um AWS SDK
- Publique a autorização de revenda multiuso sem uma data de validade e adicione um EULA personalizado usando um AWS SDK
- Publique a autorização de revenda multiuso sem uma data de expiração e adicione a documentação do contrato de revendedor usando um AWS SDK
- Publique a autorização de revenda multiuso sem data de validade e adicione uma conta de comprador específica para a revenda usando um AWS SDK
- Publique uma autorização de revenda única e adicione um cronograma de pagamento flexível usando um AWS SDK
- Publique uma autorização de revenda única para qualquer tipo de produto e adicione um EULA usando um AWS SDK
- Publique uma autorização de revenda única e adicione uma conta de comprador específica para a revenda usando um AWS SDK
- Publique uma autorização de revenda única para qualquer tipo de produto e adicione a documentação do contrato de revendedor usando um AWS SDK
- Publique uma autorização de revenda única e adicione se é uma renovação usando um AWS SDK
- Restrinja a autorização de revenda usando um AWS SDK
- Atualize o nome e a descrição da autorização de revenda única ou multiuso usando um AWS SDK

Crie um rascunho de autorização de revenda para qualquer tipo de produto usando um AWS SDK

Os exemplos de código a seguir mostram como criar um rascunho de autorização de revenda para qualquer tipo de produto a fim de analisá-lo internamente antes de publicar para um parceiro de canal.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product so my CP
 can use that to create Channel Partner Private Offer (CPPO)
CAPI-41
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "draft resale auth")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Descreva uma autorização de revenda usando um AWS SDK

Os exemplos de código a seguir mostram como descrever uma autorização de revenda.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
```

```
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
```

```
}
```

- Para obter detalhes da API, consulte DescribeEntitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for product information in
 the AWS Marketplace Catalog
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

resaleAuthorizationId = "resaleauthz-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)
```

```
def get_product_information(mp_client, entity_id):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Product with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for a product in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_product_information(mp_client, resaleAuthorizationId))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeEntityReferência da API AWS SDK for Python (Boto3).

Publique uma autorização de revenda única com uma oferta privada usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda única com uma oferta privada para que um parceiro de canal possa usá-la a fim de criar uma oferta privada de parceiro de canal (CPPO).

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
```

```
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product so my CP
can use that to create Channel Partner Private Offer (CPPO)
CAPI-42
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set, "onetime resale auth with private offer"
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique uma autorização de revenda multiuso com uma data de validade usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda multiuso com uma data de validade para um produto de AMI com preços anuais calculados por hora a fim de que um parceiro de canal possa usá-la para criar uma CPPO.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
```

```
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Contract",
            "Terms": [
                {
                    "Type": "ResaleConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.small",
                                    "Price": "150"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI product
 so my CP can use that to create Channel Partner Private Offer (CPPO)
CAPI-48
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multi-use resale auth with expiry date")
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique uma autorização de revenda multiuso com uma data de validade e um EULA a ser enviado ao comprador usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
```

```
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
```

```
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI/
Container product and add custom EULA to be sent to the buyer
CAPI-56
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multiuse resale auth with expiry date and custom
 EULA")
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique a autorização de revenda multiuso com uma data de validade e adicione a documentação do contrato de revendedor usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
```

```
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
```

```
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
```

```
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleUsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI/
Container product
and add reseller contract documentation between the ISV and channel partner
CAPI-57
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "multi use resale auth with contract doc",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique a autorização de revenda multiuso com expiração e adicione uma conta de comprador específica para a revenda usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Publish multi-use resale authorization with expiry date for any product type
 (AMI/SaaS/Container) and add specific buyer account for the resale
CAPI-82
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Publish multi-use resale authorization with expiry date and add specific
 buyer account",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique a autorização de revenda multiuso sem data de validade usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda multiuso sem data de validade para um produto de AMI com preços anuais calculados por hora para que um CP possa usá-la para criar uma CPPO.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
```

```
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
```

```
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI
 product so my CP can use that to create Channel Partner Private Offer (CPPO)
CAPI-52
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "multi use resale auth with no expiry date")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique a autorização de revenda multiuso sem uma data de validade e adicione um EULA personalizado usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
```

```
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI/
Container product and add custom EULA to be sent to the buyer
CAPI-58
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "multi use resale auth with no expiry date and custom EULA"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique a autorização de revenda multiuso sem uma data de expiração e adicione a documentação do contrato de revendedor usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI/
Container product and add reseller contract documentation between the ISV and
 channel partner
CAPI-59
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "multi use resale auth with no expiry date and contract doc"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique a autorização de revenda multiuso sem data de validade e adicione uma conta de comprador específica para a revenda usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish multi-use resale authorization without expiry date for any product type
 (AMI/SaaS/Container) and add specific buyer account for the resale
CAPI-83
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "publish multi-use resale authorization without expiry date and add
 specific buyer account",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique uma autorização de revenda única e adicione um cronograma de pagamento flexível usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar uma programação de pagamento flexível.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
                            "DimensionKey": "Users",
                            "MaxQuantity": 10
                          }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ResalePaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2023-09-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2023-12-01",
                                "ChargeAmount": "250.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-06-30",
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
                            "DimensionKey": "Users",
                            "MaxQuantity": 10
                          }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ResalePaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2023-09-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2023-12-01",
                                "ChargeAmount": "250.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-06-30",
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
```

```
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container)
 and add Flexible payment schedule
CAPI-78
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with flexible payment
 schedule")
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique uma autorização de revenda única para qualquer tipo de produto e adicione um EULA usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
```

```
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product and add
 custom EULA to be sent to the buyer
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with custom EULA")
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique uma autorização de revenda única e adicione uma conta de comprador específica para a revenda usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
```

```
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": "1"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
```

```
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
```

```
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container)
 and add specific buyer account for the resale
CAPI-81
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale authorization for specific buyer
 account")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique uma autorização de revenda única para qualquer tipo de produto e adicione a documentação do contrato de revendedor usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
```

```
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product and add
 reseller contract documentation between the ISV and channel partner
CAPI-47
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with reseller contract doc")


if __name__ == "__main__":
```

```
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique uma autorização de revenda única e adicione se é uma renovação usando um AWS SDK

Os exemplos de código a seguir mostram como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar se é uma renovação.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
```

```
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
```

```
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container)
and add whether it is renewal or not
CAPI-90
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(stringified_change_set, "onetime resale auth
 renewal")

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Restrinja a autorização de revenda usando um AWS SDK

Os exemplos de código a seguir mostram como restringir a autorização de revenda.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Restrict a  authorization for any product type (AMI/SaaS/Container)
CAPI-84
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Restrict resale authorization")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Atualize o nome e a descrição da autorização de revenda única ou multiuso usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar o nome e a descrição da autorização de revenda única ou multiuso antes da publicação de qualquer tipo de produto.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
```

```
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Update name/description of one-time or multi-use resale authorization before
 publishing for any product type (AMI/SaaS/Container)
CAPI-77
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "update name and description of one-time or multi-use resale
 authorization before publishing",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Produtos SaaS para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Crie um rascunho de produto SaaS com um rascunho de oferta pública usando um AWS SDK
- Crie um produto SaaS público ou limitado e uma oferta pública com preços contratuais usando um AWS SDK
- Crie um produto SaaS público ou limitado e uma oferta pública com contrato e Pay-As-You-Go preços usando um AWS SDK
- Crie um produto SaaS público ou limitado e uma oferta pública com preços de assinatura usando um AWS SDK
- Publique um produto SaaS e uma oferta pública associada usando um AWS SDK
- Publique um produto SaaS e uma oferta pública associada a partir de um rascunho existente usando um AWS SDK
- Atualize as dimensões em um produto AMI ou SaaS usando um AWS SDK

Crie um rascunho de produto SaaS com um rascunho de oferta pública usando um AWS SDK

Os exemplos de código a seguir mostram como criar um rascunho de produto de SaaS com um rascunho de oferta pública.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {
              "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an SaaS draft product
with a draft public offer.
CAPI-04
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create a draft saas product with a draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um produto SaaS público ou limitado e uma oferta pública com preços contratuais usando um AWS SDK

Os exemplos de código a seguir mostram como criar um produto de SaaS público ou limitado e uma oferta pública com preços contratuais. Este exemplo cria um EULA padrão ou personalizado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
```

```
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
```

```
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
```

```
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
```

```
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
```

```
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
```

```
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
```

```
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
```

```
                                {
                                    "Selector": {
                                        "Type": "Duration",
                                        "Value": "P1M"
                                    },
                                    "RateCard": [
                                        {
                                            "DimensionKey": "BasicService",
                                            "Price": "20"
                                        },
                                        {
                                            "DimensionKey": "PremiumService",
                                            "Price": "25"
                                        }
                                    ],
                                    "Constraints": {
                                        "MultipleDimensionSelection": "Allowed",
                                        "QuantityConfiguration": "Allowed"
                                    }
                                },
                                {
                                    "Selector": {
                                        "Type": "Duration",
                                        "Value": "P12M"
                                    },
                                    "RateCard": [
                                        {
                                            "DimensionKey": "BasicService",
                                            "Price": "150"
                                        },
                                        {
                                            "DimensionKey": "PremiumService",
                                            "Price": "300"
                                        }
                                    ],
                                    "Constraints": {
                                        "MultipleDimensionSelection": "Allowed",
                                        "QuantityConfiguration": "Allowed"
                                    }
                                }
                            ]
                        }
                    ]
                }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with contract pricing and
 standard EULA
CAPI-11
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
```

```
        change_set,
        "Create a limited saas product with a public offer with contract
 pricing",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um produto SaaS público ou limitado e uma oferta pública com contrato e Pay-As-You-Go preços usando um AWS SDK

Os exemplos de código a seguir mostram como criar um produto SaaS público ou limitado e uma oferta pública com um contrato com Pay-As-You-Go preços. Este exemplo cria um EULA padrão ou personalizado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
```

```
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
```

```
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
```

```
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
```

```
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
```

```
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
```

```
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
```

```
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
```

```
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with contract with PAYG pricing
 and standard EULA
CAPI-10
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create limited SaaS product with public offer with contract with payg
 pricing",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Crie um produto SaaS público ou limitado e uma oferta pública com preços de assinatura usando um AWS SDK

Os exemplos de código a seguir mostram como criar um produto de SaaS público ou limitado e uma oferta pública com preços de assinatura. Este exemplo cria um EULA padrão ou personalizado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
```

```
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
```

```
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
```

```
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
```

```
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
```

```
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with subscription(usage) pricing
 and standard EULA
CAPI-09
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "public saas public offer with subscription
 pricing")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique um produto SaaS e uma oferta pública associada usando um AWS SDK

Os exemplos de código a seguir mostram como publicar um produto de SaaS e a oferta pública associada. Por padrão, o produto estará em um estado limitado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
```

```
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
```

```
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
```

```
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
```

```
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
```

```
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish my SaaS product and associated public offer (product will be in limited
 state by default)
CAPI-05A
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
```

```
    if change_set is None:
        fname = "changeset1.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "publish saas product and associated public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Publique um produto SaaS e uma oferta pública associada a partir de um rascunho existente usando um AWS SDK

O exemplo de código a seguir mostra como publicar um produto de SaaS e uma oferta pública associada com base em um rascunho existente. Por padrão, o produto estará em um estado limitado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Public"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Atualize as dimensões em um produto AMI ou SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar dimensões em um produto de AMI ou de SaaS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update (e.g name) dimensions
 on my AMI or SaaS product
CAPI-24
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
```

```
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Update name dimensions on my AMI or SaaS product",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Utilitários para AWS Marketplace API do Catalog

Os exemplos de código a seguir mostram como usar a API de AWS Marketplace catálogo com AWS SDKs.

Exemplos

- Utilitários para iniciar um conjunto de alterações usando um AWS SDK

Utilitários para iniciar um conjunto de alterações usando um AWS SDK

Os exemplos de código a seguir mostram como definir utilitários para iniciar um conjunto de alterações.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Utilitário para carregar um conjunto de alterações de um arquivo JSON e começar a processá-lo.

```
package com.example.awsmarketplace.catalogapi;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.protocols.json.internal.unmarshall.document.DocumentUnmarshaller;
import software.amazon.awssdk.protocols.jsoncore.JsonNodeParser;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.Change;
import software.amazon.awssdk.services.marketplacecatalog.model.Entity;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetResponse;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;
```

```
import com.example.awsmarketplace.catalogapi.Entity.ChangeSet;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSetEntity;
import com.example.awsmarketplace.catalogapi.Entity.Root;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
import com.example.awsmarketplace.utils.StringSerializer;

/**
 * Before running this Java V2 code example, convert all Details attribute to
 DetailsDocument if any
 */

public class RunChangesets {

 private static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 public static void main(String[] args) {

  // input json can be specified here or passed from input parameter
  String inputChangeSetFile = "changeSets/offers/
CreateReplacementOfferFromAGWithContractPricingDetailDocument.json";

  if (args.length > 0)
   inputChangeSetFile = args[0];

  // parse the input changeset file to string for process
  String changeSetsInput = readChangeSetToString(inputChangeSetFile);

  // process the changeset request
  try {
   StartChangeSetResponse result = getChangeSetRequestResult(changeSetsInput);
   ReferenceCodesUtils.formatOutput(result);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }

 public static StartChangeSetResponse getChangeSetRequestResult(String
 changeSetsInput) throws IOException {

  //set up AWS credentials
  MarketplaceCatalogClient marketplaceCatalogClient =
```

```
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  //changeset list to save all the changesets in the changesets file
  List<Change> changeSetLists = new ArrayList<Change>();

  // read all changesets into object
  Root root = GSON.fromJson(changeSetsInput, Root.class);

  // process each changeset and add each changeset request to changesets list
  for (ChangeSet cs : root.changeSet) {

   ChangeSetEntity entity = cs.Entity;
   String entityType = entity.Type;
   String entityIdentifier = StringUtils.defaultIfBlank(entity.Identifier, null);
   Document detailsDocument = getDocumentFromObject(cs.DetailsDocument);

   Entity awsEntity =
     Entity.builder()
     .type(entityType)
     .identifier(entityIdentifier)
     .build();

   Change inputChangeRequest =
     Change.builder()
     .changeType(cs.ChangeType)
     .changeName(cs.ChangeName)
     .entity(awsEntity)
     .detailsDocument(detailsDocument)
     .build();

   changeSetLists.add(inputChangeRequest);
  }

  // process all changeset requests
  StartChangeSetRequest startChangeSetRequest =
    StartChangeSetRequest.builder()
    .catalog(root.catalog)
    .changeSet(changeSetLists)
    .build();
```

```
  StartChangeSetResponse result =
 marketplaceCatalogClient.startChangeSet(startChangeSetRequest);

  return result;
 }

 public static Document getDocumentFromObject(Object detailsObject) {

  String detailsString = "{}";
  try {
   detailsString = IOUtils.toString(new
 ByteArrayInputStream(GSON.toJson(detailsObject).getBytes()), "UTF-8");
  } catch (IOException e) {
   e.printStackTrace();
  }

  JsonNodeParser jsonNodeParser = JsonNodeParser.create();
  Document doc = jsonNodeParser.parse(detailsString).visit(new
 DocumentUnmarshaller());
  return doc;
 }


 public static String readChangeSetToString (String inputChangeSetFile) {

  InputStream changesetInputStream =
 RunChangesets.class.getClassLoader().getResourceAsStream(inputChangeSetFile);

  String changeSetsInput = null;

  try {
   changeSetsInput = IOUtils.toString(changesetInputStream, "UTF-8");
  } catch (IOException e) {
   e.printStackTrace();
  }

  return changeSetsInput;

 }
}
```

- Para obter detalhes da API, consulte StartChangeSeta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Utilitário para iniciar um conjunto de alterações.

```
"""
Purpose:

Generic function to start a changeset
"""

import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def generate_changeset(mp_client, change_set, change_set_name):
    """
    Start changeset
    """
    try:
        response = mp_client.start_change_set(
            Catalog="AWSMarketplace",
            ChangeSet=change_set,
            ChangeSetName=change_set_name,
        )
        logger.info("Changeset created!")
```

```
        logger.info("ChangeSet ID: %s", response["ChangeSetId"])
        logger.info("ChangeSet ARN: %s", response["ChangeSetArn"])

        return response

    except ClientError as e:
        logger.exception("Unexpected error: %s", e)
        raise


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

    print("-" * 88)
    print("Executing changeset: " + change_set_name)
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    response = generate_changeset(mp_client, change_set, change_set_name)

    return response

    print("-" * 88)
```

Utilitário para carregar um conjunto de alterações de um arquivo JSON.

```
"""
Purpose:

This module will stringify the details sections of a changeset file.
"""

import json


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


# open json file from path
```

```
def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


def stringify_details_sections(json_object):
    """
    Loops through every change type in the changeset to look for non-empty
    details section and stringifies them
    """
    for change_type in json_object["ChangeSet"]:
        # Only stringify details section if it is not empty
        if "Details" in change_type and change_type["Details"] != "{}":
            string_details = json.dumps(change_type["Details"])
            change_type["Details"] = string_details
        else:
            pass

    return json_object["ChangeSet"]


def stringify_changeset(file_path):
    changeset_file = open_json_file(file_path)
    changeset_stringified = stringify_details_sections(changeset_file)

    return changeset_stringified
```

- Para obter detalhes da API, consulte a StartChangeSetReferência da API AWS SDK for Python (Boto3).

Exemplos de código para AWS Marketplace API de contrato usando AWS SDKs

Os exemplos de código a seguir mostram como usar a API AWS Marketplace Agreement com um kit AWS de desenvolvimento de software (SDK).

Mais atributos

- AWS Marketplace Guia do vendedor da API do Contrato — Mais informações sobre a API AWS Marketplace do Contrato.

- AWS Marketplace Referência da API do Contrato — Detalhes sobre todas as ações disponíveis da API do AWS Marketplace Contrato.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Acordos para AWS Marketplace API do contrato
  - Aceite uma solicitação de cancelamento de contrato usando um AWS SDK
  - Aceite uma solicitação de pagamento de contrato usando um AWS SDK
  - Altere o prazo de renovação de um contrato de SaaS usando um AWS SDK
  - Altere uma AMI ConfigurableUpfrontPricingTerm para o modelo de preços de uso usando um AWS SDK
  - Crie um novo contrato de teste gratuito da AMI usando um AWS SDK
  - Crie um novo contrato de SaaS com pagamento adiantado usando um AWS SDK
  - Obtenha todas as IDs do contrato usando um AWS SDK
  - Obtenha todos os contratos usando um AWS SDK
  - Obtenha o ID do cliente de um contrato usando um AWS SDK
  - Obtenha detalhes de uma solicitação de cancelamento de contrato usando um AWS SDK
  - Obtenha detalhes de uma solicitação de pagamento de contrato usando um AWS SDK
  - Obtenha detalhes financeiros de um contrato usando um AWS SDK
  - Obtenha detalhes do teste gratuito de um contrato usando um AWS SDK
  - Obtenha informações sobre um contrato usando um AWS SDK
  - Obtenha detalhes do produto e da oferta de um contrato usando um AWS SDK
  - Obtenha o EULA de um contrato usando um AWS SDK
  - Obtenha os termos de renovação automática de um contrato usando um AWS SDK
  - Obtenha as dimensões adquiridas em um contrato usando um AWS SDK
  - Obtenha as instâncias de cada dimensão comprada em um contrato usando um AWS SDK
  - Obtenha o cronograma de pagamento de um contrato usando um AWS SDK
  - Obtenha o preço por dimensão em um contrato usando um AWS SDK

- Obtenha o tipo de preço de um contrato usando um AWS SDK
- Obtenha o tipo de produto de um contrato usando um AWS SDK
- Obtenha o status de um contrato usando um AWS SDK
- Obtenha os termos de suporte de um contrato usando um AWS SDK
- Obtenha os termos de um contrato usando um AWS SDK
- Listar solicitações de cancelamento de contrato usando um AWS SDK
- Liste as solicitações de pagamento do contrato usando um AWS SDK
- Rejeitar uma solicitação de cancelamento de contrato usando um AWS SDK
- Rejeitar uma solicitação de pagamento de contrato usando um AWS SDK
- Substitua um contrato de SaaS por uma oferta baseada em contrato usando um AWS SDK
- Substitua uma avaliação gratuita de SaaS por Contrato com preços de consumo usando um AWS SDK
- Substitua um termo de preço baseado no uso de SaaS e cancele o contrato anterior usando um AWS SDK
- Substitua um termo de preço baseado no uso da AMI, mas mantenha o existente ConfigurableUpfrontPricingTerm usando um AWS SDK
- Pesquise contratos por ID da conta usando um AWS SDK
- Pesquise contratos por ID do contrato usando um AWS SDK
- Pesquise contratos por data de término usando um AWS SDK
- Pesquise contratos por ID de oferta usando um AWS SDK
- Pesquise contratos por ID do produto usando um AWS SDK
- Pesquise contratos por status usando um AWS SDK
- Pesquise contratos com um filtro personalizado usando um AWS SDK
- Pesquise contratos com dois filtros personalizados usando um AWS SDK
- Atualize os pedidos de compra após a aceitação do contrato usando um AWS SDK

Acordos para AWS Marketplace API do contrato

Os exemplos de código a seguir mostram como usar a API AWS Marketplace Agreement com AWS SDKs.

Exemplos

- Aceite uma solicitação de cancelamento de contrato usando um AWS SDK
- Aceite uma solicitação de pagamento de contrato usando um AWS SDK
- Altere o prazo de renovação de um contrato de SaaS usando um AWS SDK
- Altere uma AMI ConfigurableUpfrontPricingTerm para o modelo de preços de uso usando um AWS SDK
- Crie um novo contrato de teste gratuito da AMI usando um AWS SDK
- Crie um novo contrato de SaaS com pagamento adiantado usando um AWS SDK
- Obtenha todas as IDs do contrato usando um AWS SDK
- Obtenha todos os contratos usando um AWS SDK
- Obtenha o ID do cliente de um contrato usando um AWS SDK
- Obtenha detalhes de uma solicitação de cancelamento de contrato usando um AWS SDK
- Obtenha detalhes de uma solicitação de pagamento de contrato usando um AWS SDK
- Obtenha detalhes financeiros de um contrato usando um AWS SDK
- Obtenha detalhes do teste gratuito de um contrato usando um AWS SDK
- Obtenha informações sobre um contrato usando um AWS SDK
- Obtenha detalhes do produto e da oferta de um contrato usando um AWS SDK
- Obtenha o EULA de um contrato usando um AWS SDK
- Obtenha os termos de renovação automática de um contrato usando um AWS SDK
- Obtenha as dimensões adquiridas em um contrato usando um AWS SDK
- Obtenha as instâncias de cada dimensão comprada em um contrato usando um AWS SDK
- Obtenha o cronograma de pagamento de um contrato usando um AWS SDK
- Obtenha o preço por dimensão em um contrato usando um AWS SDK
- Obtenha o tipo de preço de um contrato usando um AWS SDK
- Obtenha o tipo de produto de um contrato usando um AWS SDK
- Obtenha o status de um contrato usando um AWS SDK
- Obtenha os termos de suporte de um contrato usando um AWS SDK
- Obtenha os termos de um contrato usando um AWS SDK
- Listar solicitações de cancelamento de contrato usando um AWS SDK
- Liste as solicitações de pagamento do contrato usando um AWS SDK
- Rejeitar uma solicitação de cancelamento de contrato usando um AWS SDK
- Rejeitar uma solicitação de pagamento de contrato usando um AWS SDK

- Substitua um contrato de SaaS por uma oferta baseada em contrato usando um AWS SDK
- Substitua uma avaliação gratuita de SaaS por Contrato com preços de consumo usando um AWS SDK
- Substitua um termo de preço baseado no uso de SaaS e cancele o contrato anterior usando um AWS SDK
- Substitua um termo de preço baseado no uso da AMI, mas mantenha o existente ConfigurableUpfrontPricingTerm usando um AWS SDK
- Pesquise contratos por ID da conta usando um AWS SDK
- Pesquise contratos por ID do contrato usando um AWS SDK
- Pesquise contratos por data de término usando um AWS SDK
- Pesquise contratos por ID de oferta usando um AWS SDK
- Pesquise contratos por ID do produto usando um AWS SDK
- Pesquise contratos por status usando um AWS SDK
- Pesquise contratos com um filtro personalizado usando um AWS SDK
- Pesquise contratos com dois filtros personalizados usando um AWS SDK
- Atualize os pedidos de compra após a aceitação do contrato usando um AWS SDK

Aceite uma solicitação de cancelamento de contrato usando um AWS SDK

Os exemplos de código a seguir mostram como aceitar uma solicitação de cancelamento de contrato iniciada pelo vendedor.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementCancellationRe
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementCancellationRe

public class AcceptAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";

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

    private static void acceptAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        AcceptAgreementCancellationRequestRequest request =
                AcceptAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)

 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .build();

        AcceptAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.acceptAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Description: " + response.description());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
```

```
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte AcceptAgreementCancellationRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    AcceptAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";

async function acceptAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new AcceptAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
```

```
    console.log("Status: " + response.status);
    console.log("Description: " + response.description);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

acceptAgreementCancellationRequest();
```

- Para obter detalhes da API, consulte AcceptAgreementCancellationRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class AcceptAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>"

    @staticmethod
    def accept_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")
```

```
        response = client.accept_agreement_cancellation_request(
            agreementId=AcceptAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=AcceptAgreementCancellationRequest.AGREEMENT_CANCELLATION
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Description: " + str(response.get("description", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    AcceptAgreementCancellationRequest.accept_agreement_cancellation_request()
```

- Para obter detalhes da API, consulte a AcceptAgreementCancellationRequestReferência da API AWS SDK for Python (Boto3).

Aceite uma solicitação de pagamento de contrato usando um AWS SDK

Os exemplos de código a seguir mostram como aceitar uma solicitação de pagamento do contrato iniciada pelo vendedor.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementPaymentRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementPaymentRequest

public class AcceptAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
    private static final String PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER
 REFERENCE HERE>";

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

    private static void acceptAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        AcceptAgreementPaymentRequestRequest request =
                AcceptAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                        .build();

        AcceptAgreementPaymentRequestResponse response =

 marketplaceAgreementClient.acceptAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Name: " + response.name());
        System.out.println("Charge Amount: " + response.chargeAmount());
```

```
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte AcceptAgreementPaymentRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    AcceptAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
const PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER REFERENCE HERE>";

async function acceptAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new AcceptAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
            purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
        })
    );
```

```
    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Name: " + response.name);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

acceptAgreementPaymentRequest();
```

- Para obter detalhes da API, consulte AcceptAgreementPaymentRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class AcceptAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"
    PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER REFERENCE HERE>"
```

```
    @staticmethod
    def accept_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.accept_agreement_payment_request(
            agreementId=AcceptAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=AcceptAgreementPaymentRequest.PAYMENT_REQUEST_ID,

 purchaseOrderReference=AcceptAgreementPaymentRequest.PURCHASE_ORDER_REFERENCE,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    AcceptAgreementPaymentRequest.accept_agreement_payment_request()
```

- Para obter detalhes da API, consulte a AcceptAgreementPaymentRequestReferência da API AWS SDK for Python (Boto3).

Altere o prazo de renovação de um contrato de SaaS usando um AWS SDK

Os exemplos de código a seguir mostram como alterar um contrato de SaaS para adicionar ou atualizar seu prazo de renovação.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RenewalTermConfiguration;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model and
 then turn on
 * the auto-renewal setting using the AWS Marketplace Agreement Service APIs.
```

```
 *
 * <p>Scenario: A buyer subscribes to a SaaS product using a public offer that
 supports
 * auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
 * auto-renewal via the RenewalTerm configuration.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code P1M}
 for 1 month).</li>
 *   <li>{@code DIMENSION_1_KEY} — the dimension key defined in the offer.</li>
 * </ul>
 */
public class AmendSaaSContractRenewalTerm {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // The dimension key defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the RenewalTerm in your offer.
    private static final String RENEWAL_TERM_ID = "<your-renewal-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";
```

```
    // Term ID for the SupportTerm in your offer.
    private static final String SUPPORT_TERM_ID = "<your-support-term-id>";

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

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
disabled.
     * 2. Wait for entitlements to become active.
     * 3. Amend the agreement to enable auto-renewal.
     */
    private static void amendSaaSContractAgreementRenewalTerm() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build())
                                .build()))
                .build();

        // Initial agreement: auto-renewal disabled.
        RequestedTerm renewalTerm = RequestedTerm.builder()
                .id(RENEWAL_TERM_ID)

.configuration(RequestedTermConfiguration.fromRenewalTermConfiguration(
                        RenewalTermConfiguration.builder()
                                .enableAutoRenew(false)
```

```
                                .build()))
                .build();

        RequestedTerm legalTerm =
RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm supportTerm =
RequestedTerm.builder().id(SUPPORT_TERM_ID).build();

        // --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
renewalTerm, legalTerm, supportTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before amending.
        System.out.println("Waiting for entitlements to become active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("Entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);
```

```
        // --- Amend: enable auto-renewal ---
        RequestedTerm renewalTermAmended = RequestedTerm.builder()
                .id(RENEWAL_TERM_ID)

 .configuration(RequestedTermConfiguration.fromRenewalTermConfiguration(
                        RenewalTermConfiguration.builder()
                                .enableAutoRenew(true)
                                .build()))
                .build();

        // Use Intent.AMEND and sourceAgreementIdentifier to target the existing
 agreement.
        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.AMEND)
                        .requestedTerms(configurableUpfrontPricingTerm,
 renewalTermAmended, legalTerm, supportTerm)

 .sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Amend agreement request created. AgreementRequestId:
 " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        System.out.println("Amendment accepted. Auto-renewal enabled. New
 AgreementId: " + aarResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model and
 then turn on
 * the auto-renewal setting using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product using a public offer that
 supports
 * auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
 * auto-renewal via the RenewalTerm configuration.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P1M" for 1 month).
 *   - DIMENSION_1_KEY — the dimension key defined in the offer.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";
```

```
// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
const SELECTOR_VALUE = "<your-selector-value>";

// The dimension key defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-key>";

// Quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the RenewalTerm in your offer.
const RENEWAL_TERM_ID = "<your-renewal-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the SupportTerm in your offer.
const SUPPORT_TERM_ID = "<your-support-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
 disabled.
 * 2. Wait for entitlements to become active.
 * 3. Amend the agreement to enable auto-renewal.
 */
async function amendSaaSContractAgreementRenewalTerm() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                ],
            },
        },
    };
```

```
   // Initial agreement: auto-renewal disabled.
   const renewalTerm = {
       id: RENEWAL_TERM_ID,
       configuration: {
           renewalTermConfiguration: {
               enableAutoRenew: false,
           },
       },
   };

   const legalTerm = { id: LEGAL_TERM_ID };
   const supportTerm = { id: SUPPORT_TERM_ID };

   // --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
   const createAgreementRequestResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "NEW",
           requestedTerms: [configurableUpfrontPricingTerm, renewalTerm,
legalTerm, supportTerm],
           agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
       })
   );
   console.log("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

   const acceptAgreementRequestResponse = await client.send(
       new AcceptAgreementRequestCommand({
           agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
       })
   );
   console.log("Agreement request accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId);

   // Wait for entitlements to become active before amending.
   console.log("Waiting for entitlements to become active...");
   const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
   console.log("Entitlements are now active.");
   formatOutput(entitlementsResponse);
```

```
    // --- Amend: enable auto-renewal ---
    const renewalTermAmended = {
        id: RENEWAL_TERM_ID,
        configuration: {
            renewalTermConfiguration: {
                enableAutoRenew: true,
            },
        },
    };

    // Use Intent.AMEND and sourceAgreementIdentifier to target the existing
 agreement.
    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "AMEND",
            requestedTerms: [configurableUpfrontPricingTerm, renewalTermAmended,
 legalTerm, supportTerm],
            sourceAgreementIdentifier:
 acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Amend agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    console.log("Amendment accepted. Auto-renewal enabled. New AgreementId: " +
 aarResponse.agreementId);
}

amendSaaSContractAgreementRenewalTerm();
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create a SaaS agreement with CONTRACT pricing model and then
 turn on
the auto-renewal setting using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product using a public offer that supports
auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
auto-renewal via the RenewalTerm configuration.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P1M for 1 month).
  - DIMENSION_1_KEY — the dimension key defined in the offer.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class AmendSaaSContractRenewalTerm:

    # The agreementProposalId from the offer.
```

```
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
    SELECTOR_VALUE = "<your-selector-value>"

    # The dimension key defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the RenewalTerm in your offer.
    RENEWAL_TERM_ID = "<your-renewal-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the SupportTerm in your offer.
    SUPPORT_TERM_ID = "<your-support-term-id>"

    @staticmethod
    def amend_saas_contract_agreement_renewal_term():
        """
        Full end-to-end flow:
        1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
 disabled.
        2. Wait for entitlements to become active.
        3. Amend the agreement to enable auto-renewal.
        """
        client = boto3.client("marketplace-agreement")
        cls = AmendSaaSContractRenewalTerm

        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
```

```
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        # Initial agreement: auto-renewal disabled.
        renewal_term = {
            "id": cls.RENEWAL_TERM_ID,
            "configuration": {
                "renewalTermConfiguration": {
                    "enableAutoRenew": False,
                }
            },
        }

        legal_term = {"id": cls.LEGAL_TERM_ID}
        support_term = {"id": cls.SUPPORT_TERM_ID}

        # --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, renewal_term,
legal_term, support_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request accepted. AgreementId: " + agreement_id)

        # Wait for entitlements to become active before amending.
        print("Waiting for entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("Entitlements are now active.")
```

```
        format_output(entitlements_response)

        # --- Amend: enable auto-renewal ---
        renewal_term_amended = {
            "id": cls.RENEWAL_TERM_ID,
            "configuration": {
                "renewalTermConfiguration": {
                    "enableAutoRenew": True,
                }
            },
        }

        # Use Intent.AMEND and sourceAgreementIdentifier to target the existing
 agreement.
        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="AMEND",
            requestedTerms=[configurable_upfront_pricing_term,
 renewal_term_amended, legal_term, support_term],
            sourceAgreementIdentifier=agreement_id,
        )
        print("Amend agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        print("Amendment accepted. Auto-renewal enabled. New AgreementId: " +
 aar_response["agreementId"])


if __name__ == "__main__":
    AmendSaaSContractRenewalTerm.amend_saas_contract_agreement_renewal_term()
```

- Para obter detalhes da API, consulte a CreateAgreementRequestReferência da API AWS SDK for Python (Boto3).

Altere uma AMI ConfigurableUpfrontPricingTerm para o modelo de preços de uso usando um AWS SDK

Os exemplos de código a seguir mostram como alterar um contrato de AMI ConfigurableUpfrontPricingTerm atualizando a quantidade da dimensão para o modelo de preços de uso.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
```

```
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create an AMI agreement with
 ConfigurableUpfrontPricingTerm and then amend the dimension quantity
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: An AMI product with USAGE pricing requires two agreements:
 * <ol>
 *   <li>An agreement with <b>usageBasedPricingTerm (UBPT)</b> — accepted first
 to establish the base agreement.</li>
 *   <li>An agreement with <b>configurableUpfrontPricingTerm (CUPT)</b> —
 accepted after the UBPT agreement entitlements are active.</li>
 * </ol>
 * Once both agreement entitlements are available, this sample shows how to
 <b>amend</b> the agreement
 * with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} —  duration for the agreement
 *       (e.g., {@code P365D} for one year).</li>
 *   <li>{@code DIMENSION_1_KEY} — the dimension key defined in the offer (e.g.,
 instance type).</li>
 *   <li>{@code DIMENSION_1_VALUE} — initial quantity; {@code
 NEW_DIMENSION_1_VALUE} — amended quantity.</li>
 * </ul>
 */
public class AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";
```

```
    // Duration for the agreement (e.g., "P365D" for 365 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Initial quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the UsageBasedPricingTerm in your offer.
    private static final String USAGE_TERM_ID = "<your-usage-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in your offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // New quantity to use when amending the dimension of CUPT.
    private static final int NEW_DIMENSION_1_VALUE = 5;

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

    /**
     * Full end-to-end flow:
     * 1. Create and accept an agreement request with usageBasedPricingTerm.
     * 2. Wait for entitlements to become active, then create and accept an
 agreement request with configurableUpfrontPricingTerm (CUPT).
     * 3. Wait for CUPT entitlements to become active, then amend the dimension
 quantity.
     */
    private static void amendAmiCUPTAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageTerm = RequestedTerm.builder()
                .id(USAGE_TERM_ID)
                .build();
```

```
        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
                .build();
        RequestedTerm validityTerm = RequestedTerm.builder()
                .id(VALIDITY_TERM_ID)
                .build();

        // --- Agreement with UBPT ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageTerm, legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with UBPT accepted. AgreementId: "
+ acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before creating the agreement
with CUPT.
        System.out.println("Waiting for UBPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("UBPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);
```

```
        // --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                      ConfigurableUpfrontPricingTermConfiguration.builder()
                              .selectorValue(SELECTOR_VALUE)
                              .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                      .build())
                              .build()))
                .build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Agreement request with CUPT created.
AgreementRequestId: " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String cuptAgreementId = aarResponse.agreementId();
        System.out.println("Agreement request with CUPT accepted. AgreementId: "
+ cuptAgreementId);

        // Wait for entitlements to become active before amending.
        System.out.println("Waiting for CUPT agreement entitlements to become
active...");
```

```
        GetAgreementEntitlementsResponse cuptEntitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, cuptAgreementId);
        System.out.println("CUPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(cuptEntitlementsResponse);

        // --- Amend Agreement with CUPT ---
        // Increase the dimension quantity using Intent.AMEND and
sourceAgreementIdentifier.
        RequestedTerm newConfig = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(NEW_DIMENSION_1_VALUE) // Increase quantity for this dimension
key
                                                        .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carAmendRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.AMEND)
                        .requestedTerms(newConfig, legalTerm, validityTerm)
                        .sourceAgreementIdentifier(cuptAgreementId)
                        .build();
        CreateAgreementRequestResponse carAmendResponse =

marketplaceAgreementClient.createAgreementRequest(carAmendRequest);
        System.out.println("Amendment of CUPT agreement request created.
AgreementRequestId: " + carAmendResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarAmendRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(carAmendResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarAmendResponse =
```

```
 marketplaceAgreementClient.acceptAgreementRequest(aarAmendRequest);
        System.out.println("Amendment accepted. New AgreementId: " +
 aarAmendResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create an AMI agreement with
 ConfigurableUpfrontPricingTerm and then amend the dimension quantity
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: An AMI product with USAGE pricing requires two agreements:
 *   1. An agreement with usageBasedPricingTerm (UBPT) — accepted first to
 establish the base agreement.
 *   2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after
 the UBPT agreement entitlements are active.
 *
```

```
 * Once both agreement entitlements are available, this sample shows how to amend
 the agreement
 * with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P365D" for one year).
 *   - DIMENSION_1_KEY — the dimension key defined in the offer (e.g., instance
 type).
 *   - DIMENSION_1_VALUE — initial quantity; NEW_DIMENSION_1_VALUE — amended
 quantity.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P365D" for 365 days).
const SELECTOR_VALUE = "<your-selector-value>";

// The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
const DIMENSION_1_KEY = "<your-dimension-key>";

// Initial quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the UsageBasedPricingTerm in your offer.
const USAGE_TERM_ID = "<your-usage-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in your offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// New quantity to use when amending the dimension of CUPT.
const NEW_DIMENSION_1_VALUE = 5;
```

```
/**
 * Full end-to-end flow:
 * 1. Create and accept an agreement request with usageBasedPricingTerm.
 * 2. Wait for entitlements to become active, then create and accept an agreement
 request with configurableUpfrontPricingTerm (CUPT).
 * 3. Wait for CUPT entitlements to become active, then amend the dimension
 quantity.
 */
async function amendAmiCUPTAgreement() {
    const client = new MarketplaceAgreementClient();

    const usageTerm = { id: USAGE_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Agreement with UBPT ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageTerm, legalTerm, validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with UBPT accepted. AgreementId: " +
 acceptAgreementRequestResponse.agreementId);

    // Wait for entitlements to become active before creating the agreement with
 CUPT.
    console.log("Waiting for UBPT agreement entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
 acceptAgreementRequestResponse.agreementId);
    console.log("UBPT agreement entitlements are now active.");
    formatOutput(entitlementsResponse);
```

```
   // --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
   const configurableUpfrontPricingTerm = {
       id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
       configuration: {
           configurableUpfrontPricingTermConfiguration: {
               selectorValue: SELECTOR_VALUE,
               dimensions: [
                   { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
               ],
           },
       },
   };

   const carResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "NEW",
           requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
           agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
       })
   );
   console.log("Agreement request with CUPT created. AgreementRequestId: " +
carResponse.agreementRequestId);

   const aarResponse = await client.send(
       new AcceptAgreementRequestCommand({
           agreementRequestId: carResponse.agreementRequestId,
       })
   );
   const cuptAgreementId = aarResponse.agreementId;
   console.log("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

   // Wait for entitlements to become active before amending.
   console.log("Waiting for CUPT agreement entitlements to become active...");
   const cuptEntitlementsResponse = await pollUntilEntitlementsAvailable(client,
cuptAgreementId);
   console.log("CUPT agreement entitlements are now active.");
   formatOutput(cuptEntitlementsResponse);

   // --- Amend Agreement with CUPT ---
```

```
    // Increase the dimension quantity using Intent.AMEND and
 sourceAgreementIdentifier.
    const newConfig = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 NEW_DIMENSION_1_VALUE }, // Increase quantity for this dimension key
                ],
            },
        },
    };

    const carAmendResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "AMEND",
            requestedTerms: [newConfig, legalTerm, validityTerm],
            sourceAgreementIdentifier: cuptAgreementId,
        })
    );
    console.log("Amendment of CUPT agreement request created. AgreementRequestId:
 " + carAmendResponse.agreementRequestId);

    const aarAmendResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carAmendResponse.agreementRequestId,
        })
    );
    console.log("Amendment accepted. New AgreementId: " +
 aarAmendResponse.agreementId);
}

amendAmiCUPTAgreement();
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create an AMI agreement with ConfigurableUpfrontPricingTerm
 and then amend
the dimension quantity using the AWS Marketplace Agreement Service APIs.

Scenario: An AMI product with USAGE pricing requires two agreements:
  1. An agreement with usageBasedPricingTerm (UBPT) — accepted first to establish
 the base agreement.
  2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
 UBPT agreement
     entitlements are active.
Once both agreement entitlements are available, this sample shows how to amend
 the agreement
with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P365D for one year).
  - DIMENSION_1_KEY — the dimension key defined in the offer (e.g., instance
 type).
  - DIMENSION_1_VALUE — initial quantity; NEW_DIMENSION_1_VALUE — amended
 quantity.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
```

```
    generate_client_token,
    poll_until_entitlements_available,
)


class AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P365D" for 365 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Initial quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the UsageBasedPricingTerm in your offer.
    USAGE_TERM_ID = "<your-usage-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in your offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # New quantity to use when amending the dimension of CUPT.
    NEW_DIMENSION_1_VALUE = 5

    @staticmethod
    def amend_ami_cupt_agreement():
        """
        Full end-to-end flow:
        1. Create and accept an agreement request with usageBasedPricingTerm.
        2. Wait for entitlements to become active, then create and accept an
 agreement request
           with configurableUpfrontPricingTerm (CUPT).
```

```
        3. Wait for CUPT entitlements to become active, then amend the dimension
quantity.
        """
        client = boto3.client("marketplace-agreement")
        cls = AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel

        usage_term = {"id": cls.USAGE_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Agreement with UBPT ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_term, legal_term, validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " +
agreement_id)

        # Wait for entitlements to become active before creating the agreement
with CUPT.
        print("Waiting for UBPT agreement entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("UBPT agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
```

```
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        print("Agreement request with CUPT created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        cupt_agreement_id = aar_response["agreementId"]
        print("Agreement request with CUPT accepted. AgreementId: " +
cupt_agreement_id)

        # Wait for entitlements to become active before amending.
        print("Waiting for CUPT agreement entitlements to become active...")
        cupt_entitlements_response = poll_until_entitlements_available(client,
cupt_agreement_id)
        print("CUPT agreement entitlements are now active.")
        format_output(cupt_entitlements_response)

        # --- Amend Agreement with CUPT ---
        # Increase the dimension quantity using Intent.AMEND and
sourceAgreementIdentifier.
        new_config = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
```

```
                            "dimensionValue": cls.NEW_DIMENSION_1_VALUE,  #
 Increase quantity for this dimension key
                        },
                    ],
                }
            },
        }

        car_amend_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="AMEND",
            requestedTerms=[new_config, legal_term, validity_term],
            sourceAgreementIdentifier=cupt_agreement_id,
        )
        print(
            "Amendment of CUPT agreement request created. AgreementRequestId: "
            + car_amend_response["agreementRequestId"]
        )

        aar_amend_response = client.accept_agreement_request(
            agreementRequestId=car_amend_response["agreementRequestId"]
        )
        print("Amendment accepted. New AgreementId: " +
 aar_amend_response["agreementId"])


if __name__ == "__main__":

 AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel.amend_ami_cupt_agreement()
```

- Para obter detalhes da API, consulte a CreateAgreementRequestReferência da API AWS SDK for Python (Boto3).

Crie um novo contrato de teste gratuito da AMI usando um AWS SDK

Os exemplos de código a seguir mostram como criar um novo contrato de teste gratuito da AMI.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create an AMI Free Trial agreement
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to an AMI product that offers a free trial
 period.
 * The free trial includes a FreeTrialPricingTerm alongside a
 UsageBasedPricingTerm.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
```

```
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 * </ul>
 */
public class NewAmiFreeTrial {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the FreeTrialPricingTerm in your offer.
    private static final String FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-
pricing-term-id>";

    // Term ID for the UsageBasedPricingTerm in your offer (applies after the
 trial ends).
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the SupportTerm in your offer.
    private static final String SUPPORT_TERM_ID = "<your-support-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

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

    /**
     * Creates an AMI Free Trial agreement.
     * The FreeTrialPricingTerm grants access at no cost for the trial period.
     * The UsageBasedPricingTerm defines the charges that apply once the trial
 ends.
     */
    private static void createAndAcceptAmiFreeTrialAgreementRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();
```

```
        RequestedTerm freeTrialPricingTerm = RequestedTerm.builder()
                .id(FREE_TRIAL_PRICING_TERM_ID)
                .build();
        RequestedTerm usageBasedPricingTerm = RequestedTerm.builder()
                .id(USAGE_BASED_PRICING_TERM_ID)
                .build();
        RequestedTerm supportTerm = RequestedTerm.builder()
                .id(SUPPORT_TERM_ID)
                .build();
        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
                .build();

        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(freeTrialPricingTerm,
 usageBasedPricingTerm, supportTerm, legalTerm)

 .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with freeTrialPricingTerm accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create an AMI Free Trial agreement
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to an AMI product that offers a free trial
 period.
 * The free trial includes a FreeTrialPricingTerm alongside a
 UsageBasedPricingTerm.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the FreeTrialPricingTerm in your offer.
```

```
const FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>";

// Term ID for the UsageBasedPricingTerm in your offer (applies after the trial
 ends).
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the SupportTerm in your offer.
const SUPPORT_TERM_ID = "<your-support-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

/**
 * Creates an AMI Free Trial agreement.
 * The FreeTrialPricingTerm grants access at no cost for the trial period.
 * The UsageBasedPricingTerm defines the charges that apply once the trial ends.
 */
async function createAndAcceptAmiFreeTrialAgreementRequest() {
    const client = new MarketplaceAgreementClient();

    const freeTrialPricingTerm = { id: FREE_TRIAL_PRICING_TERM_ID };
    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const supportTerm = { id: SUPPORT_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };

    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [freeTrialPricingTerm, usageBasedPricingTerm,
 supportTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
```

```
    console.log("Agreement request with freeTrialPricingTerm accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId);
}

createAndAcceptAmiFreeTrialAgreementRequest();
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create an AMI Free Trial agreement
using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to an AMI product that offers a free trial period.
The free trial includes a FreeTrialPricingTerm alongside a UsageBasedPricingTerm.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
"""

import boto3

from utils.agreement_api_utils import generate_client_token


class NewAmiFreeTrial:
```

```
    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the FreeTrialPricingTerm in your offer.
    FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>"

    # Term ID for the UsageBasedPricingTerm in your offer (applies after the
trial ends).
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the SupportTerm in your offer.
    SUPPORT_TERM_ID = "<your-support-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    @staticmethod
    def create_and_accept_ami_free_trial_agreement_request():
        """
        Create an AMI Free Trial agreement.

        The FreeTrialPricingTerm grants access at no cost for the trial period.
        The UsageBasedPricingTerm defines the charges that apply once the trial
ends.
        """
        client = boto3.client("marketplace-agreement")

        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[
                {"id": NewAmiFreeTrial.FREE_TRIAL_PRICING_TERM_ID},
                {"id": NewAmiFreeTrial.USAGE_BASED_PRICING_TERM_ID},
                {"id": NewAmiFreeTrial.SUPPORT_TERM_ID},
                {"id": NewAmiFreeTrial.LEGAL_TERM_ID},
            ],

agreementProposalIdentifier=NewAmiFreeTrial.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)
```

```
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print(
            "Agreement request with freeTrialPricingTerm accepted. AgreementId: "
            + accept_response["agreementId"]
        )


if __name__ == "__main__":
    NewAmiFreeTrial.create_and_accept_ami_free_trial_agreement_request()
```

- Para obter detalhes da API, consulte a CreateAgreementRequestReferência da API AWS SDK for Python (Boto3).

Crie um novo contrato de SaaS com pagamento adiantado usando um AWS SDK

Os exemplos de código a seguir mostram como criar um novo contrato de SaaS com pagamento adiantado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import
 software.amazon.awssdk.services.marketplaceagreement.model.TaxConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
 * selecting an agreement duration and specifying quantities for multiple
 dimensions.
 * Tax estimation is enabled at the time of agreement creation.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code P12M}
 for 12 months).</li>
 *   <li>{@code DIMENSION_1_KEY}, {@code DIMENSION_2_KEY} — dimension keys
 defined in the offer.</li>
 *   <li>{@code DIMENSION_1_VALUE}, {@code DIMENSION_2_VALUE} — quantities for
 each dimension.</li>
 * </ul>
 */
```

```
public class NewSaaSContractWithUpfrontPayment {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P12M" for 12 months).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // First dimension key and quantity defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-1-key>";
    private static final int DIMENSION_1_VALUE = 10;

    // Second dimension key and quantity defined in your offer.
    private static final String DIMENSION_2_KEY = "<your-dimension-2-key>";
    private static final int DIMENSION_2_VALUE = 20;

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Tax estimation setting: "ENABLED" to include estimated taxes in the
 agreement.
    private static final String TAX_ESTIMATION = "ENABLED";

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

    /**
     * Creates a SaaS agreement with CONTRACT pricing model with configurable
 upfront pricing
     * for multiple dimensions and tax estimation.
     */
    private static void createSaaSContractAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();
```

```
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build(),
                                                Dimension.builder()

.dimensionKey(DIMENSION_2_KEY)

.dimensionValue(DIMENSION_2_VALUE)
                                                        .build())
                                .build()))
                .build();

        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
                .build();

        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
legalTerm)

.taxConfiguration(TaxConfiguration.builder().taxEstimation(TAX_ESTIMATION).build())

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()
```

```
 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("SaaS agreement request with CONTRACT pricing model
 accepted. AgreementId: " + acceptAgreementRequestResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
```

```
 * selecting an agreement duration and specifying quantities for multiple
 dimensions.
 * Tax estimation is enabled at the time of agreement creation.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P12M" for 12 months).
 *   - DIMENSION_1_KEY, DIMENSION_2_KEY — dimension keys defined in the offer.
 *   - DIMENSION_1_VALUE, DIMENSION_2_VALUE — quantities for each dimension.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P12M" for 12 months).
const SELECTOR_VALUE = "<your-selector-value>";

// First dimension key and quantity defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-1-key>";
const DIMENSION_1_VALUE = 10;

// Second dimension key and quantity defined in your offer.
const DIMENSION_2_KEY = "<your-dimension-2-key>";
const DIMENSION_2_VALUE = 20;

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Tax estimation setting: "ENABLED" to include estimated taxes in the agreement.
const TAX_ESTIMATION = "ENABLED";

/**
 * Creates a SaaS agreement with CONTRACT pricing model with configurable upfront
 pricing
 * for multiple dimensions and tax estimation.
 */
async function createSaaSContractAgreement() {
```

```
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                    { dimensionKey: DIMENSION_2_KEY, dimensionValue:
 DIMENSION_2_VALUE },
                ],
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };

    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm],
            taxConfiguration: { taxEstimation: TAX_ESTIMATION },
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("SaaS agreement request with CONTRACT pricing model accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId);
}

createSaaSContractAgreement();
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
selecting an agreement duration and specifying quantities for multiple
 dimensions.
Tax estimation is enabled at the time of agreement creation.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P12M for 12 months).
  - DIMENSION_1_KEY, DIMENSION_2_KEY — dimension keys defined in the offer.
  - DIMENSION_1_VALUE, DIMENSION_2_VALUE — quantities for each dimension.
"""

import boto3

from utils.agreement_api_utils import generate_client_token


class NewSaaSContractWithUpfrontPayment:
```

```
    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P12M" for 12 months).
    SELECTOR_VALUE = "<your-selector-value>"

    # First dimension key and quantity defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-1-key>"
    DIMENSION_1_VALUE = 10

    # Second dimension key and quantity defined in your offer.
    DIMENSION_2_KEY = "<your-dimension-2-key>"
    DIMENSION_2_VALUE = 20

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Tax estimation setting: "ENABLED" to include estimated taxes in the
 agreement.
    TAX_ESTIMATION = "ENABLED"

    @staticmethod
    def create_saas_contract_agreement():
        """
        Create a SaaS agreement with CONTRACT pricing model with configurable
 upfront pricing
        for multiple dimensions and tax estimation.
        """
        client = boto3.client("marketplace-agreement")

        configurable_upfront_pricing_term = {
            "id":
 NewSaaSContractWithUpfrontPayment.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue":
 NewSaaSContractWithUpfrontPayment.SELECTOR_VALUE,
                    "dimensions": [
                        {
```

```
                            "dimensionKey":
 NewSaaSContractWithUpfrontPayment.DIMENSION_1_KEY,
                            "dimensionValue":
 NewSaaSContractWithUpfrontPayment.DIMENSION_1_VALUE,
                        },
                        {
                            "dimensionKey":
 NewSaaSContractWithUpfrontPayment.DIMENSION_2_KEY,
                            "dimensionValue":
 NewSaaSContractWithUpfrontPayment.DIMENSION_2_VALUE,
                        },
                    ],
                }
            },
        }

        legal_term = {"id": NewSaaSContractWithUpfrontPayment.LEGAL_TERM_ID}

        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term],
            taxConfiguration={"taxEstimation":
 NewSaaSContractWithUpfrontPayment.TAX_ESTIMATION},

 agreementProposalIdentifier=NewSaaSContractWithUpfrontPayment.AGREEMENT_PROPOSAL_IDENTIF
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print(
            "SaaS agreement request with CONTRACT pricing model accepted.
 AgreementId: "
            + accept_response["agreementId"]
        )


if __name__ == "__main__":
    NewSaaSContractWithUpfrontPayment.create_saas_contract_agreement()
```

- Para obter detalhes da API, consulte a CreateAgreementRequestReferência da API AWS SDK for Python (Boto3).

Obtenha todas as IDs do contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter todos os IDs de contratos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
public class GetAllAgreementsIds {

 /*
  * Get all purchase agreements ids for your PartyType (Proposer, or Acceptor)
  * Depend on the number of agreements in your account, this code may take some
 time to finish.
  */
 public static void main(String[] args) {

  List<String> agreementIds = getAllAgreementIds();

  ReferenceCodesUtils.formatOutput(agreementIds);

 }

 public static List<String> getAllAgreementIds() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
 agreements where you are the proposer.
  // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
 are the acceptor.
  Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  List<Filter> searchFilters = new ArrayList<Filter>();

  searchFilters.addAll(Arrays.asList(partyType, agreementType));

  // Save all results in a list array
  List<AgreementViewSummary> agreementSummaryList = new
 ArrayList<AgreementViewSummary>();

  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(searchFilters)
```

```
    .build();

  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .nextToken(searchAgreementsResponse.nextToken())
     .filters(searchFilters)
     .build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }

  List<String> agreementIds = new ArrayList<String>();
  for (AgreementViewSummary summary : agreementSummaryList) {
   agreementIds.add(summary.agreementId());
  }
  return agreementIds;
 }

}
```

- Para obter detalhes da API, consulte SearchAgreementsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreement ids
AG-09
"""

import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10


def get_agreements():
    AgreementSummaryList = []
    agreement_id_list = []

    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you
 are the proposer.
```

```
            # Change to "Acceptor" to return agreements where you are the
 acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise

    AgreementSummaryList.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreements["agreementViewSummaries"])

    for agreement in AgreementSummaryList:
        agreement_id_list.append(agreement["agreementId"])

    return agreement_id_list


if __name__ == "__main__":
    agreement_id_list = get_agreements()

    print(agreement_id_list)
```

- Para obter detalhes da API, consulte a SearchAgreementsReferência da API AWS SDK for Python (Boto3).

Obtenha todos os contratos usando um AWS SDK

Os exemplos de código a seguir mostram como obter todos os contratos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreements {
```

```
/*
 * Get all purchase agreements for your PartyType (Proposer, or Acceptor)
 * Depend on the number of agreements in your account, this code may take some
time to finish.
 */
public static void main(String[] args) {

 List<AgreementViewSummary> agreementSummaryList = getAllAgreements();

 ReferenceCodesUtils.formatOutput(agreementSummaryList);
}

public static List<AgreementViewSummary> getAllAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

       // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return
agreements where you are the acceptor.
       // Change to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements where
you are the proposer.
 Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_ACCEPTOR).build();

 Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 List<Filter> searchFilters = new ArrayList<Filter>();

 searchFilters.addAll(Arrays.asList(partyType, agreementType));

 // Save all results in a list array

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(searchFilters)
   .build();
```

```
  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .nextToken(searchAgreementsResponse.nextToken())
     .filters(searchFilters).build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }
  return agreementSummaryList;
 }

}
```

- Para obter detalhes da API, consulte SearchAgreementsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
```

```
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreements
AG-01
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

party_type_list = ["Acceptor"]
agreement_type_list = ["PurchaseAgreement"]

filter_list = [
    {"name": "PartyType", "values": party_type_list},
    {"name": "AgreementType", "values": agreement_type_list},
]

agreement_results_list = []


def get_agreements(filter_list=filter_list):
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            filters=filter_list,
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e
```

```
    agreement_results_list.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=filter_list,
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        agreement_results_list.extend(agreements["agreementViewSummaries"])

    return agreement_results_list


if __name__ == "__main__":
    agreements_list = get_agreements(filter_list)
    helper.pretty_print_datetime(agreements_list)
```

- Para obter detalhes da API, consulte a SearchAgreementsReferência da API AWS SDK for Python (Boto3).

Obtenha o ID do cliente de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter um ID de cliente de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementCustomerInfo {

 /*
  * Obtain metadata about the customer who created the agreement, such as the
 customer's AWS Account ID
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Customer's AWS Account ID is " +
 describeAgreementResponse.acceptor().accountId());

 }

 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();
```

```
  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Para obter detalhes da API, consulte DescribeAgreementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get customer AWS account id
from a given agreement
AG-08
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)
```

```
def get_agreement_information(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
            raise e
        else:
            logger.error("Unexpected error: %s", e)
            raise e

    return response


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement_information(agreement_id=args.agreement_id)

    print(f"Customer account: {response['acceptor']['accountId']}")
```

- Para obter detalhes da API, consulte a DescribeAgreementReferência da API AWS SDK for Python (Boto3).

Obtenha detalhes de uma solicitação de cancelamento de contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter detalhes de uma solicitação de cancelamento de contrato específica.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementCancellationReque
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementCancellationReque

public class GetAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";

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

    private static void getAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        GetAgreementCancellationRequestRequest request =
                GetAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
```

```
 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .build();

        GetAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.getAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte GetAgreementCancellationRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    GetAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
```

```
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";

async function getAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new GetAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

getAgreementCancellationRequest();
```

- Para obter detalhes da API, consulte GetAgreementCancellationRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
```

```
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class GetAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>"

    @staticmethod
    def get_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.get_agreement_cancellation_request(
            agreementId=GetAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=GetAgreementCancellationRequest.AGREEMENT_CANCELLATION_RE
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    GetAgreementCancellationRequest.get_agreement_cancellation_request()
```

- Para obter detalhes da API, consulte a GetAgreementCancellationRequestReferência da API AWS SDK for Python (Boto3).

Obtenha detalhes de uma solicitação de pagamento de contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter detalhes de uma solicitação de pagamento de um contrato específico.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementPaymentRequestReq
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementPaymentRequestRes

public class GetAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";

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

    private static void getAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();
```

```
        GetAgreementPaymentRequestRequest request =
                GetAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .build();

        GetAgreementPaymentRequestResponse response =
                marketplaceAgreementClient.getAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Name: " + response.name());
        System.out.println("Charge ID: " + response.chargeId());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte GetAgreementPaymentRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    GetAgreementPaymentRequestCommand,
```

```
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";

async function getAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new GetAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Name: " + response.name);
    console.log("Charge ID: " + response.chargeId);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

getAgreementPaymentRequest();
```

- Para obter detalhes da API, consulte GetAgreementPaymentRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class GetAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"

    @staticmethod
    def get_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.get_agreement_payment_request(
            agreementId=GetAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=GetAgreementPaymentRequest.PAYMENT_REQUEST_ID,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge ID: " + str(response.get("chargeId", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
```

```
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    GetAgreementPaymentRequest.get_agreement_payment_request()
```

- Para obter detalhes da API, consulte a GetAgreementPaymentRequestReferência da API AWS SDK for Python (Boto3).

Obtenha detalhes financeiros de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter os detalhes financeiros de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementFinancialDetails {

 /*
```

```
  * Obtain financial details, such as Total Contract Value of the agreement from
 a given agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String totalContractValue = getTotalContractValue(agreementId);

  System.out.println("Total Contract Value is " + totalContractValue);

 }

 public static String getTotalContractValue(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  String totalContractValue = "N/A";

  if ( describeAgreementResponse.estimatedCharges() != null ) {
   totalContractValue =
 describeAgreementResponse.estimatedCharges().agreementValue()
     + " "
     + describeAgreementResponse.estimatedCharges().currencyCode();
  }
  return totalContractValue;
 }
}
```

- Para obter detalhes da API, consulte DescribeAgreementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain financial details, such as Total Contract Value of the agreementfrom a
 given agreement
AG-14

Example Usage: python3 get_agreement_financial_details.py --agreement-id
 <agreement-id>
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_information(agreement_id):
    try:
        agreement = mp_client.describe_agreement(agreementId=agreement_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
            logger.error("Agreement with ID %s not found.", agreement_id)
        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_information(args.agreement_id)

    if agreement is not None:
        print(f"Agreement Id: {args.agreement_id}")
        print(
            f"Agreement Value: {agreement['estimatedCharges']['currencyCode']}
 {agreement['estimatedCharges']['agreementValue']}"
        )

    else:
        print(f"Agreement with ID {args.agreement_id} is not found")
```

- Para obter detalhes da API, consulte a DescribeAgreementReferência da API AWS SDK for Python (Boto3).

Obtenha detalhes do teste gratuito de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter os detalhes do teste gratuito com base em um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.FreeTrialPricingTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsFreeTrialDetails {

 /*
  * Obtain the details from an agreement of a free trial I have provided to the
 customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;
```

```
  List<FreeTrialPricingTerm> freeTrialPricingTerms =
 getFreeTrialPricingTerms(agreementId);

  ReferenceCodesUtils.formatOutput(freeTrialPricingTerms);
 }

 public static List<FreeTrialPricingTerm> getFreeTrialPricingTerms(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
     .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<FreeTrialPricingTerm> freeTrialPricingTerms = new
 ArrayList<FreeTrialPricingTerm>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.freeTrialPricingTerm() != null) {
    freeTrialPricingTerms.add(acceptedTerm.freeTrialPricingTerm());
   }
  }
  return freeTrialPricingTerms;
 }
}
```

- Para obter detalhes da API, consulte DescribeAgreementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the details from an agreement of a free trial I have provided to the
 customer
AG-20

Example Usage: python3 get_agreement_free_trial_details.py --agreement-id
 <agreement-id>
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_terms(agreement_id):
    try:
```

```
        agreement = mp_client.get_agreement_terms(agreementId=agreement_id)
        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)

        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_terms(agreement_id=args.agreement_id)

    if agreement is not None:
        freetrial_found = False

        for term in agreement["acceptedTerms"]:
            if "freeTrialPricingTerm" in term.keys():
                helper.pretty_print_datetime(term)
                freetrial_found = True

        if not freetrial_found:
            print(f"No free trial term found for agreement: {args.agreement_id}")
```

- Para obter detalhes da API, consulte a DescribeAgreementReferência da API AWS SDK for Python (Boto3).

Obtenha informações sobre um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter informações sobre um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class DescribeAgreement {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse = getResponse(agreementId);

  ReferenceCodesUtils.formatOutput(describeAgreementResponse);

 }

 public static DescribeAgreementResponse getResponse(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
```

```
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Para obter detalhes da API, consulte DescribeAgreementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get agreement information
AG-07
"""

import argparse
import logging

import boto3
```

```
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)


def get_agreement_information(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
            raise e
        else:
            logger.error("Unexpected error: %s", e)
            raise e

    return response


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement_information(agreement_id=args.agreement_id)

    helper.pretty_print_datetime(response)
```

- Para obter detalhes da API, consulte a DescribeAgreementReferência da API AWS SDK for Python (Boto3).

Obtenha detalhes do produto e da oferta de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter os detalhes de produtos e ofertas com base em um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class GetProductAndOfferDetailFromAgreement {

 public static void main(String[] args) {

  // call Agreement API to get offer and product information for the agreement

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DescribeEntityResponse> entityResponseList = getEntities(agreementId);

  for (DescribeEntityResponse response : entityResponseList) {
   ReferenceCodesUtils.formatOutput(response);
  }
 }

 public static List<DescribeEntityResponse> getEntities(String agreementId) {
  List<DescribeEntityResponse> entityResponseList = new
 ArrayList<DescribeEntityResponse> ();

  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  // get offer id for the given agreement

  String offerId = describeAgreementResponse.proposalSummary().offerId();

  // get all the product ids for this agreement

  List<String> productIds = new ArrayList<String>();
```

```
  for (Resource resource :
 describeAgreementResponse.proposalSummary().resources()) {
   productIds.add(resource.id());
  }

  // call Catalog API to get the details of the offer and products

  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId).build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);

  entityResponseList.add(describeEntityResponse);

  for (String productId : productIds) {
   describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(productId).build();
   describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   System.out.println("Print details for product " + productId);
   entityResponseList.add(describeEntityResponse);
  }
  return entityResponseList;
 }
}
```

- Para obter detalhes da API, consulte DescribeAgreementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get product and offer details
 in a given agreement
AG-10
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mpa_client = boto3.client("marketplace-agreement")
mpc_client = boto3.client("marketplace-catalog")

logger = logging.getLogger(__name__)


def get_agreement_information(agreement_id):
    """
    Returns information about a given agreement
    Args: agreement_id str: Entity to return
```

```
    Returns: dict: Dictionary of agreement information
    """

    try:
        agreement = mpa_client.describe_agreement(agreementId=agreement_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_entity_information(entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mpc_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_agreement_components(agreement_id):
    agreement_component_list = []

    agreement = get_agreement_information(agreement_id)

    if agreement is not None:
        productIds = []
```

```
        for resource in agreement["proposalSummary"]["resources"]:
            productIds.append(resource["id"])

        for product_id in productIds:
            product_document = get_entity_information(product_id)

            product_document_dict = {}
            product_document_dict["product_id"] = product_id
            product_document_dict["document"] = product_document
            agreement_component_list.append(product_document_dict)

        offerId = agreement["proposalSummary"]["offerId"]

        offer_document = get_entity_information(offerId)

        offer_document_dict = {}
        offer_document_dict["offer_id"] = offerId
        offer_document_dict["document"] = offer_document
        agreement_component_list.append(offer_document_dict)

        return agreement_component_list

    else:
        print("Agreement with ID " + args.agreement_id + " is not found")


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

    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement_id",
        "-aid",
        help="Provide agreement ID to search for product and offer detail",
        required=True,
    )
    args = parser.parse_args()

    product_offer_detail =
 get_agreement_components(agreement_id=args.agreement_id)

    helper.pretty_print_datetime(product_offer_detail)
```

- Para obter detalhes da API, consulte a DescribeAgreementReferência da API AWS SDK for Python (Boto3).

Obtenha o EULA de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter o EULA de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.DocumentItem;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsEula {

 /*
  * Obtain the EULA I have entered into with my customer via the agreement
```

```
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DocumentItem> legalEulaArray = getLegalEula(agreementId);

  ReferenceCodesUtils.formatOutput(legalEulaArray);
 }

 public static List<DocumentItem> getLegalEula(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<DocumentItem> legalEulaArray = new ArrayList<>();

  getAgreementTermsResponse.acceptedTerms().stream()
      .filter(acceptedTerm -> acceptedTerm.legalTerm() != null &&
 acceptedTerm.legalTerm().hasDocuments())
      .flatMap(acceptedTerm -> acceptedTerm.legalTerm().documents().stream())
      .filter(docItem -> docItem.type() != null)
      .forEach(legalEulaArray::add);
  return legalEulaArray;
 }

}
```

- Para obter detalhes da API, consulte GetAgreementTermsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the EULA I have entered into with my customer via the agreement
AG-18
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
```

```
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "legalTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "documents"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        legalEulaArray = []
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term and ATTRIBUTE_NAME in term[TERM_NAME]:
                docs = term[TERM_NAME][ATTRIBUTE_NAME]
                for doc in docs:
                    if "type" in doc:
                        legalEulaArray.append(doc)
        return legalEulaArray

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)
```

```
    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTermsReferência da API AWS SDK for Python (Boto3).

Obtenha os termos de renovação automática de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter os termos de renovação automática de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementAutoRenewal {

 /*
  * Obtain the auto-renewal status of the agreement
  */

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String autoRenewal = getAutoRenewal(agreementId);

  System.out.println("Auto-Renewal status is " + autoRenewal);
 }

 public static String getAutoRenewal(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  String autoRenewal = "No Auto Renewal";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.renewalTerm() != null &&
 acceptedTerm.renewalTerm().configuration() != null
```

```
     && acceptedTerm.renewalTerm().configuration().enableAutoRenew() != null) {
    autoRenewal =
 String.valueOf(acceptedTerm.renewalTerm().configuration().enableAutoRenew().booleanValue
    break;
   }
  }
  return autoRenewal;
 }

}
```

- Para obter detalhes da API, consulte GetAgreementTermsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the auto-renewal status of the agreement
AG-15
"""

import json
import logging
import os
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))
```

```
import utils.helpers as helper


import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-11111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "renewalTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "enableAutoRenew"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        auto_renewal = "No Auto Renewal"
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    auto_renewal = term[TERM_NAME][CONFIG_ELEM][ATTRIBUTE_NAME]
                    break
```

```
        return auto_renewal

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    agreement = get_agreement_information(mp_client, AGREEMENT_ID)

    if agreement is not None:
        print(f"Auto Renewal is {agreement}")
    else:
        print("Agreement with ID " + AGREEMENT_ID + " is not found")


# open json file from path
def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTermsReferência da API AWS SDK for Python (Boto3).

Obtenha as dimensões adquiridas em um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter as dimensões compradas em um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionPurchased {

 /*
  * Obtain the dimensions the buyer has purchased from me via the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> dimensionKeys = getDimensionKeys(agreementId);
```

```
  ReferenceCodesUtils.formatOutput(dimensionKeys);
 }

 public static List<String> getDimensionKeys(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<String> dimensionKeys = new ArrayList<String>();
  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !
= null) {
     List<Dimension> dimensions =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     for (Dimension dimension : dimensions) {
      dimensionKeys.add(dimension.dimensionKey());
     }
    }

   }
  }
  return dimensionKeys;
 }
}
```

- Para obter detalhes da API, consulte GetAgreementTermsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the dimensions the buyer has purchased from me via the agreement
AG-28
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
```

```
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        dimensionKeys = []

        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    confParam = term[TERM_NAME][CONFIG_ELEM]
                    if ATTRIBUTE_NAME in confParam:
                        if "dimensions" in confParam:
                            for dimension in confParam["dimensions"]:
                                if "dimensionKey" in dimension:
                                    dimensionKey = dimension["dimensionKey"]
                                    print(f"Dimension Key: {dimensionKey}")
                                    dimensionKeys.append(dimensionKey)
        return dimensionKeys

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
```

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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTermsReferência da API AWS SDK for Python (Boto3).

Obtenha as instâncias de cada dimensão comprada em um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter as instâncias de cada dimensão compradas em um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionInstances {

 /*
  * get instances of each dimension that buyer has purchased in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;
```

```
  Map<String, List<Dimension>> dimensionMap = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionMap);
 }

 public static Map<String, List<Dimension>> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  Map<String, List<Dimension>> dimensionMap = new HashMap<String,
 List<Dimension>>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Dimension> dimensionsList = new ArrayList<Dimension>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    String selectorValue = "";
    if (acceptedTerm.configurableUpfrontPricingTerm().configuration() != null) {
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !
= null) {
      selectorValue =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue();
     }
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().hasDimensions())
 {
      dimensionsList =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     }
    }
    if (selectorValue.length() > 0) {
     dimensionMap.put(selectorValue, dimensionsList);
    }
   }
```

```
  }
  return dimensionMap;
 }
}
```

- Para obter detalhes da API, consulte GetAgreementTermsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain instances of each dimension that buyer has purchased in the agreement
AG-30
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"

logger = logging.getLogger(__name__)


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        terms = mp_client.get_agreement_terms(agreementId=entity_id)
        dimensionKeyValueMap = {}
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    confParam = term[TERM_NAME][CONFIG_ELEM]
                    if ATTRIBUTE_NAME in confParam:
                        selectValue = confParam["selectorValue"]
                        dimensionKeyValueMap["selectorValue"] = selectValue
                        if "dimensions" in confParam:
                            dimensionKeyValueMap["dimensions"] =
 confParam["dimensions"]
                            """
                            for dimension in confParam['dimensions']:
                                if 'dimensionKey' in dimension:

                                    dimensionValue = dimension['dimensionValue']
                                    dimensionKey = dimension['dimensionKey']
                                    print(f"Selector: {selectValue}, Dimension
 Key: {dimensionKey}, Dimension Value: {dimensionValue}")
                                    dimensionKeyValueMap[dimensionKey] =
 dimensionValue
                            """
        return dimensionKeyValueMap
```

```
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTermsReferência da API AWS SDK for Python (Boto3).

Obtenha o cronograma de pagamento de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter a programação de pagamento de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentScheduleTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.ScheduleItem;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPaymentSchedule {

 /*
  * Obtain the payment schedule I have agreed to with the agreement, including
 the invoice date and invoice amount
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Map<String, Object>> paymentScheduleArray =
 getPaymentSchedules(agreementId);

  ReferenceCodesUtils.formatOutput(paymentScheduleArray);
 }

 public static List<Map<String, Object>> getPaymentSchedules(String agreementId)
 {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  List<Map<String, Object>> paymentScheduleArray = new ArrayList<>();

  String currencyCode = "";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.paymentScheduleTerm() != null) {
    PaymentScheduleTerm paymentScheduleTerm = acceptedTerm.paymentScheduleTerm();
    if (paymentScheduleTerm.currencyCode() != null) {
     currencyCode = paymentScheduleTerm.currencyCode();
    }
    if (paymentScheduleTerm.hasSchedule()) {
     for (ScheduleItem schedule : paymentScheduleTerm.schedule()) {
      if (schedule.chargeDate() != null) {
       String chargeDate = schedule.chargeDate().toString();
       String chargeAmount = schedule.chargeAmount();
       Map<String, Object> scheduleMap = new HashMap<>();
       scheduleMap.put(ATTRIBUTE_CURRENCY_CODE, currencyCode);
       scheduleMap.put(ATTRIBUTE_CHARGE_DATE, chargeDate);
       scheduleMap.put(ATTRIBUTE_CHARGE_AMOUNT, chargeAmount);
       paymentScheduleArray.add(scheduleMap);
      }
     }
    }
   }
  }
  return paymentScheduleArray;
 }
}
```

- Para obter detalhes da API, consulte GetAgreementTermsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the payment schedule I have agreed to with the agreement, including the
 invoice date and invoice amount
AG-17
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"
```

```
# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "paymentScheduleTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        paymentScheduleArray = []
        currencyCode = ""
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                paymentSchedule = term[TERM_NAME]
                if "currencyCode" in paymentSchedule:
                    currencyCode = paymentSchedule["currencyCode"]
                if "schedule" in paymentSchedule:
                    for sch in paymentSchedule["schedule"]:
                        if "chargeDate" in sch:
                            chargeDate = sch["chargeDate"]
                            chargeAmount = sch["chargeAmount"]
                            # print(f"chargeDate: {chargeDate}, chargeAmount:
 {chargeAmount}")
                            schedule = {
                                "currencyCode": currencyCode,
                                "chargeDate": chargeDate,
                                "chargeAmount": chargeAmount,
                            }
                            paymentScheduleArray.append(schedule)

        return paymentScheduleArray
```

```
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTermsReferência da API AWS SDK for Python (Boto3).

Obtenha o preço por dimensão em um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter os preços por dimensão em um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPricingEachDimension {

 /*
  * Obtain pricing per each dimension in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Object> dimensions = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensions);
```

```
 }

 public static List<Object> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<Object> dimensions = new ArrayList<Object>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Object> rateInfo = new ArrayList<Object>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if (acceptedTerm.configurableUpfrontPricingTerm().type() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().type());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().currencyCode() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().currencyCode());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().hasRateCards()) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().rateCards());
    }
    dimensions.add(rateInfo);
   }
  }
  return dimensions;
 }
}
```

- Para obter detalhes da API, consulte GetAgreementTermsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain pricing per each dimension in the agreement
AG-29
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
```

```
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"

TERMS_TO_SEARCH = [
    "configurableUpfrontPricingTerm",
    "usageBasedPricingTerm",
    "fixedUpfrontPricingTerm",
]


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        dimentions = []
        for term in terms[ROOT_ELEM]:
            for t in TERMS_TO_SEARCH:
                rateInfo = []
                if t in term:
                    if "type" in term[t]:
                        rateInfo.append(term[t]["type"])
                    if "currencyCode" in term[t]:
                        rateInfo.append(term[t]["currencyCode"])
                    if "rateCards" in term[t]:
                        rateInfo.append(term[t]["rateCards"])
                    dimentions.append(rateInfo)
        return dimentions

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
```

```
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTermsReferência da API AWS SDK for Python (Boto3).

Obtenha o tipo de preço de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter o tipo de preço de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import com.fasterxml.jackson.annotation.JsonAutoDetect.Visibility;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.Set;
```

```
import org.apache.commons.lang3.tuple.Triple;

import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

/*
 * Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
 */
public class GetAgreementPricingType {

 private static final String FILTER_NAME = "OfferId";

 private static final String FILTER_VALUE = OFFER_ID;

 // Product types
 private static final String SAAS_PRODUCT = "SaaSProduct";
 private static final String AMI_PRODUCT = "AmiProduct";
 private static final String ML_PRODUCT = "MachineLearningProduct";
 private static final String CONTAINER_PRODUCT = "ContainerProduct";
 private static final String DATA_PRODUCT = "DataProduct";
 private static final String PROSERVICE_PRODUCT = "ProfessionalServicesProduct";
 private static final String AIQ_PRODUCT = "AiqProduct";

 // Pricing types
 private static final String CCP = "CCP";
 private static final String ANNUAL = "Annual";
 private static final String CONTRACT = "Contract";
 private static final String SFT = "SaaS Free Trial";
 private static final String HMA = "Hourly and Monthly Agreements";
 private static final String HOURLY = "Hourly";
 private static final String MONTHLY = "Monthly";
 private static final String AFPS = "Annual FPS";
```

```
private static final String CFPS = "Contract FPS";
private static final String CCPFPS = "CCP with FPS";
private static final String BYOL = "BYOL";
private static final String FREE = "Free";
private static final String FTH = "Free Trials and Hourly";

// Agreement term pricing types
private static final Set<String> LEGAL = Set.of("LegalTerm");
private static final Set<String> CONFIGURABLE_UPFRONT =
Set.of("ConfigurableUpfrontPricingTerm");
private static final Set<String> USAGE_BASED = Set.of("UsageBasedPricingTerm");
private static final Set<String> CONFIGURABLE_UPFRONT_AND_USAGE_BASED =
Set.of("ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm");
private static final Set<String> FREE_TRIAL = Set.of("FreeTrialPricingTerm");
private static final Set<String> RECURRING_PAYMENT =
Set.of("RecurringPaymentTerm");
private static final Set<String> USAGE_BASED_AND_RECURRING_PAYMENT =
Set.of("UsageBasedPricingTerm", "RecurringPaymentTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE =
Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm");
private static final Set<String>
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED =
Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm",
"UsageBasedPricingTerm");
private static final Set<String> BYOL_PRICING = Set.of("ByolPricingTerm");
private static final Set<String> FREE_TRIAL_AND_USAGE_BASED =
Set.of("FreeTrialPricingTerm", "UsageBasedPricingTerm");

private static final List<Set<String>> ALL_AGREEMENT_TERM_TYPES_COMBINATION
= Arrays.asList(LEGAL, CONFIGURABLE_UPFRONT, USAGE_BASED,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
  FREE_TRIAL, RECURRING_PAYMENT, USAGE_BASED_AND_RECURRING_PAYMENT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, BYOL_PRICING,
FREE_TRIAL_AND_USAGE_BASED);

private static  MarketplaceAgreementClient marketplaceAgreementClient =
  MarketplaceAgreementClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

private static MarketplaceCatalogClient marketplaceCatalogClient =
  MarketplaceCatalogClient.builder()
```

```
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

   /*
    * Get agreement Pricing Type given product type, agreement term types and
offer types if needed
    */
public static String getPricingType(String productType, Set<String>
agreementTermType, Set<String> offerType) {
 Map<Triple<String, Set<String>, Set<String>>, String> pricingTypes = new
HashMap<>();

 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
new HashSet<>()), CCP);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
new HashSet<>()), CCP);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(ML_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FREE_TRIAL, new HashSet<>()), SFT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED_AND_RECURRING_PAYMENT, new
HashSet<>()), HMA);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, USAGE_BASED, new HashSet<>()),
HOURLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(ML_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, RECURRING_PAYMENT, new
HashSet<>()), MONTHLY);
```

```
 pricingTypes.put(Triple.of(AMI_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(ML_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), AFPS);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(ML_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(DATA_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, LEGAL, new HashSet<>()), FREE);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
```

```
 pricingTypes.put(Triple.of(ML_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);

 Triple<String, Set<String>, Set<String>> key = Triple.of(productType,
agreementTermType, offerType);

 if (pricingTypes.containsKey(key)) {
  return pricingTypes.get(key);
 } else {
  return "Unknown";
 }
}

/*
 * Given product type and agreement term types, some combinations need to check
offer term types as well.
 */
public static String needToCheckOfferTermsType(String productType, Set<String>
agreementTermTypes) {
 Map<KeyPair, String> offerTermTypes = new HashMap<>();
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");

 KeyPair key = new KeyPair(productType, agreementTermTypes);
 if (offerTermTypes.containsKey(key)) {
  return offerTermTypes.get(key);
 } else {
  return null;
 }
}

public static List<AgreementViewSummary> getAgreementsById() {

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
agreements where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
are the acceptor.
```

```
 Filter partyType =
Filter.builder().name(PARTY_TYPE_FILTER_NAME).values(PARTY_TYPE_FILTER_VALUE_PROPOSER).b

 Filter agreementType =
Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME).values(AGREEMENT_TYPE_FILTER_VALUE_PUR

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(partyType, agreementType, customizeFilter).build();

 SearchAgreementsResponse searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());

 while (searchResultResponse.nextToken() != null &&
searchResultResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
SearchAgreementsRequest.builder().catalog(AWS_MP_CATALOG)
    .filters(partyType,
agreementType).nextToken(searchResultResponse.nextToken()).build();
  searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
 }
 return agreementSummaryList;

}

static class KeyPair {
 private final String first;
 private final Set<String> second;

 public KeyPair(String productType, Set<String> second) {
  this.first = productType;
  this.second = second;
 }

 @Override
 public int hashCode() {
```

```
    return Objects.hash(first, second);
  }

  @Override
  public boolean equals(Object obj) {
    if (this == obj)
      return true;
    if (obj == null || getClass() != obj.getClass())
      return false;
    KeyPair other = (KeyPair) obj;
    return Objects.equals(first, other.first) && Objects.equals(second,
other.second);
  }
}

/*
 * Get all the term types for the offer
 */
public static Set<String> getOfferTermTypes(String offerId) {

  Set<String> offerTermTypes = new HashSet<String>();

  DescribeEntityRequest request =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse result =
marketplaceCatalogClient.describeEntity(request);

  String details = result.details();

  try {
    ObjectMapper objectMapper = new ObjectMapper();
    JsonNode rootNode = objectMapper.readTree(details);
    JsonNode termsNode = rootNode.get(ATTRIBUTE_TERMS);

    for (JsonNode termNode : termsNode) {
      if (termNode.get(ATTRIBUTE_TYPE_ENTITY) != null ) {
        offerTermTypes.add(termNode.get(ATTRIBUTE_TYPE_ENTITY).asText());
      }
    }
  } catch (Exception e) {
```

```
  e.printStackTrace();
 }

 return offerTermTypes;

}

/*
 * Get all the agreement term types
 */
public static Set<String> getAgreementTermTypes(GetAgreementTermsResponse
agreementTerm) {
 Set<String> agreementTermTypes = new HashSet<String>();
 try {
  for (AcceptedTerm term : agreementTerm.acceptedTerms()) {
   ObjectMapper objectMapper = new ObjectMapper();
   JsonNode termNode = objectMapper.readTree(getJson(term));
   Iterator<Map.Entry<String, JsonNode>> fieldsIterator = termNode.fields();
   while (fieldsIterator.hasNext()) {
    Map.Entry<String, JsonNode> entry = fieldsIterator.next();
    JsonNode value = entry.getValue();
    if (value.isObject() && value.has(ATTRIBUTE_TYPE_AGREEMENT)) {
     agreementTermTypes.add(value.get(ATTRIBUTE_TYPE_AGREEMENT).asText());
    }
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }
 return agreementTermTypes;

}

/*
 * make sure all elements in array2 exist in array1
 */
public static boolean allElementsExist(Set<String> array1, Set<String> array2) {
 for (String element : array2) {
  boolean found = false;
  for (String str : array1) {
   if (element.equals(str)) {
    found = true;
    break;
   }
```

```
    }
    if (!found) {
     return false;
    }
   }
   return true;
  }

  /*
   * Find the combinations of the agreement term types for the agreement
   */
  public static Set<String> getMatchedTermTypesCombination(Set<String>
  agreementTermTypes) {
   Set<String> matchedCombination = new HashSet<String>();
   for (Set<String> element : ALL_AGREEMENT_TERM_TYPES_COMBINATION) {
    if (allElementsExist(agreementTermTypes, element)) {
     matchedCombination = element;
    }
   }
   return matchedCombination;
  }

  public static void main(String[] args) {

   List<AgreementViewSummary> agreements = getAgreementsById();

   for (AgreementViewSummary summary : agreements) {
    String pricingType = "";
    String agreementId = summary.agreementId();
    System.out.println(agreementId);
    String offerId = summary.proposalSummary().offerId();

    //get all pricing term types for the offer in the agreement
    Set<String> offerTermTypes = getOfferTermTypes(offerId);
    String productType = summary.proposalSummary().resources().get(0).type();

    //get all pricing term types for the agreement
    GetAgreementTermsRequest getAgreementTermsRequest =
      GetAgreementTermsRequest.builder().agreementId(agreementId)
      .build();
    GetAgreementTermsResponse getAgreementTermsResponse =
  marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
    Set<String> agreementTermTypes =
  getAgreementTermTypes(getAgreementTermsResponse);
```

```
    //get matched pricing term type combination set
    Set<String> agreementMatchedTermType =
  getMatchedTermTypesCombination(agreementTermTypes);

    //check to see if this agreement pricing term combination needs additional
  check on offer pricing terms
    String needToCheckOfferType = needToCheckOfferTermsType(productType,
  agreementMatchedTermType);

    // get the pricing type for the agreement based on the product type, agreement
  term types and offer term types if needed
    if (needToCheckOfferType != null) {
     Set<String> offerMatchedTermType =
  getMatchedTermTypesCombination(offerTermTypes);
     pricingType = getPricingType(productType, agreementMatchedTermType,
  offerMatchedTermType);
    } else if (agreementMatchedTermType == LEGAL) {
     pricingType = FREE;
    } else {
     pricingType = getPricingType(productType, agreementMatchedTermType, new
  HashSet());
    }
    System.out.println("Pricing type is " + pricingType);
   }
  }

  private static String getJson(Object result) {
   String json = "";

   try {
    ObjectMapper om = new ObjectMapper();
    om.setVisibility(PropertyAccessor.FIELD, Visibility.ANY);
    om.registerModule(new JavaTimeModule());
    ObjectWriter ow = om.writer().withDefaultPrettyPrinter();

    json = ow.writeValueAsString(result);
   } catch (JsonProcessingException e) {
    e.printStackTrace();
   }
   return json;
  }

 }
```

- Para obter detalhes da API, consulte DescribeAgreementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
AG-16
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

# To search by offer id: OfferId; by product id: ResourceIdentifier; by product
 type: ResourceType
idType = "OfferId"

# replace id value as needed
idValue = "offer-1111111111111"

MAX_PAGE_RESULTS = 10

# catalog; switch to AWSMarketplace for release
AWSMPCATALOG = "AWSMarketplace"
```

```
# product types

SaaSProduct = "SaaSProduct"
AmiProduct = "AmiProduct"
MLProduct = "MachineLearningProduct"
ContainerProduct = "ContainerProduct"
DataProduct = "DataProduct"
ProServiceProduct = "ProfessionalServicesProduct"
AiqProduct = "AiqProduct"

# Define pricing types
CCP = "CCP"
Annual = "Annual"
Contract = "Contract"
SFT = "SaaS Free Trial"
HMA = "Hourly and Monthly Agreements"
Hourly = "Hourly"
Monthly = "Monthly"
AFPS = "Annual FPS"
CFPS = "Contract FPS"
CCPFPS = "CCP with FPS"
BYOL = "BYOL"
Free = "Free"
FTH = "Free Trials and Hourly"

# Define Agreement Term Types
legal = ["LegalTerm"]
config = ["ConfigurableUpfrontPricingTerm"]
usage = ["UsageBasedPricingTerm"]
config_usage = ["ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm"]
freeTrial = ["FreeTrialPricingTerm"]
recur = ["RecurringPaymentTerm"]
usage_recur = ("UsageBasedPricingTerm", "RecurringPaymentTerm")
fixed_payment = ["FixedUpfrontPricingTerm", "PaymentScheduleTerm"]
fixed_payment_usage = [
    "FixedUpfrontPricingTerm",
    "PaymentScheduleTerm",
    "UsageBasedPricingTerm",
]
byol = ["ByolPricingTerm"]
freeTrial_usage = ("FreeTrialPricingTerm", "UsageBasedPricingTerm")
all_agreement_types_combination = (
    legal,
```

```
    config,
    usage,
    config_usage,
    freeTrial,
    recur,
    usage_recur,
    fixed_payment,
    fixed_payment_usage,
    byol,
    freeTrial_usage,
)


# get pricing type method given product type, agreement temr type and offer type
 if needed
def get_pricing_type(product_type, agreement_term_type, offer_type):
    pricing_types = {
        (SaaSProduct, frozenset(config_usage), frozenset("")): CCP,
        (DataProduct, frozenset(config_usage), frozenset("")): CCP,
        (ContainerProduct, frozenset(config), frozenset(config_usage)): Annual,
        (AmiProduct, frozenset(config), frozenset(config_usage)): Annual,
        (MLProduct, frozenset(config), frozenset(config_usage)): Annual,
        (ContainerProduct, frozenset(config), frozenset(config)): Contract,
        (AmiProduct, frozenset(config), frozenset(config)): Contract,
        (SaaSProduct, frozenset(config), frozenset("")): Contract,
        (DataProduct, frozenset(config), frozenset("")): Contract,
        (AiqProduct, frozenset(config), frozenset("")): Contract,
        (ProServiceProduct, frozenset(config), frozenset("")): Contract,
        (SaaSProduct, frozenset(freeTrial), frozenset("")): SFT,
        (AmiProduct, frozenset(usage_recur), frozenset("")): HMA,
        (SaaSProduct, frozenset(usage), frozenset("")): Hourly,
        (AmiProduct, frozenset(usage), frozenset("")): Hourly,
        (ContainerProduct, frozenset(usage), frozenset("")): Hourly,
        (MLProduct, frozenset(usage), frozenset("")): Hourly,
        (ContainerProduct, frozenset(recur), frozenset("")): Monthly,
        (AmiProduct, frozenset(recur), frozenset("")): Monthly,
        (
            ContainerProduct,
            frozenset(fixed_payment),
            frozenset(fixed_payment_usage),
        ): AFPS,
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment_usage)):
 AFPS,
        (MLProduct, frozenset(fixed_payment), frozenset("")): AFPS,
```

```
        (ContainerProduct, frozenset(fixed_payment), frozenset(fixed_payment)):
 CFPS,
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment)): CFPS,
        (SaaSProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (DataProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (AiqProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (ProServiceProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (SaaSProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (DataProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (AiqProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (ProServiceProduct, frozenset(fixed_payment_usage), frozenset("")):
 CCPFPS,
        (AmiProduct, frozenset(byol), frozenset("")): BYOL,
        (SaaSProduct, frozenset(byol), frozenset("")): BYOL,
        (ProServiceProduct, frozenset(byol), frozenset("")): BYOL,
        (AiqProduct, frozenset(byol), frozenset("")): BYOL,
        (MLProduct, frozenset(byol), frozenset("")): BYOL,
        (ContainerProduct, frozenset(byol), frozenset("")): BYOL,
        (DataProduct, frozenset(byol), frozenset("")): BYOL,
        (ContainerProduct, frozenset(legal), frozenset("")): Free,
        (AmiProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
        (ContainerProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
        (MLProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
    }

    key = (product_type, agreement_term_type, offer_type)
    if key in pricing_types:
        return pricing_types[key]
    else:
        return "Unknown"


# Example usage for testing purpose
"""
product_type = SaaSProduct
agreement_term_type = frozenset(config_usage)
offer_type = frozenset('')
pricing_type = get_pricing_type(product_type, agreement_term_type, offer_type)
print("pricing type = " + pricing_type)  # Output: CCP
"""


# check if offer term types are needed; if Y, needed
def get_offer_term_type(product_type, agreement_term_type):
```

```
    offer_term_types = {
        (ContainerProduct, frozenset(config)): "Y",
        (AmiProduct, frozenset(config)): "Y",
        (ContainerProduct, frozenset(fixed_payment)): "Y",
        (AmiProduct, frozenset(fixed_payment)): "Y",
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment)): "Y",
    }

    key = (product_type, agreement_term_type)
    if key in offer_term_types:
        return offer_term_types[key]
    else:
        return


logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the
 proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog=AWSMPCATALOG,
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": idType, "values": [idValue]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
```

```
                    catalog=AWSMPCATALOG,
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": idType, "values": [idValue]},
                        {"name": "AgreementType", "values":
 ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    # find all agreements matching the specified idType and idValue
    agreements = get_agreements(mp_client)

    for item in agreements:
        pricingType = ""
        agreement_id = item["agreementId"]

        # get term types inside offer
        offer_term_types = get_offer_term_types(item)

        # even though multiple product types are allowed for one agreement, only
 need the first one
        productType = item["proposalSummary"]["resources"][0]["type"]

        # get agreement terms types
        agreementTerm = mp_client.get_agreement_terms(agreementId=agreement_id)
```

```
        agreementTermTypes = get_agreement_term_types(agreementTerm)

        # match with agreement term type group
        matchedTermType = getMatchedTermTypesCombination(agreementTermTypes)

        # check if offer term type is needed.
        offer_term_type_needed = get_offer_term_type(
            productType, frozenset(matchedTermType)
        )

        # get pricing type given product type, agreement term types and offer
 type if needed;
        # one excpetion is Container with Legal term. LegalTerm needs to be the
 only term present
        if offer_term_type_needed is not None:
            matchedOfferTermTypes =
 getMatchedTermTypesCombination(offer_term_types)
            print(f"matchedOfferTermType = {matchedOfferTermTypes}")
            pricingType = get_pricing_type(
                productType,
                frozenset(matchedTermType),
                frozenset(matchedOfferTermTypes),
            )
        elif set(matchedTermType) == set(legal):
            pricingType = Free
        else:
            pricingType = get_pricing_type(
                productType, frozenset(matchedTermType), frozenset("")
            )

        print(
            f"agreementId={agreement_id};productType={productType};
 agreementTermTypes={agreementTermTypes}; matchedTermType={matchedTermType};
 offerTermTypeNeeded={offer_term_type_needed};
 offer_term_types={offer_term_types}"
        )
        print(f"pricing type={pricingType}")


def getMatchedTermTypesCombination(agreementTermTypes):
    matchedCombination = ()
    for element in all_agreement_types_combination:
        if check_elements(agreementTermTypes, element):
```

```
            matchedCombination = element
    return matchedCombination


def get_offer_term_types(item):
    offer_id = item["proposalSummary"]["offerId"]
    mp_catalogAPI_client = boto3.client("marketplace-catalog")
    offer_document = get_entity_information(mp_catalogAPI_client, offer_id)
    offerDetail = offer_document["Details"]
    offerDetail_json_object = json.loads(offerDetail)
    offer_term_types = [term["Type"] for term in
 offerDetail_json_object["Terms"]]
    return offer_term_types


# make sure all elements in array2 exist in array1
def check_elements(array1, array2):
    for element in array2:
        if element not in array1:
            return False
    return True


def get_entity_information(mp_client, entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)
```

```
def get_agreement_term_types(agreementTerm):
    types = []
    for term in agreementTerm["acceptedTerms"]:
        for value in term.values():
            if isinstance(value, dict) and "type" in value:
                types.append(value["type"])
    return types


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeAgreementReferência da API AWS SDK for Python (Boto3).

Obtenha o tipo de produto de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter o tipo de produto de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementProductType {

 /*
  * Obtain the Product Type of the product the agreement was created on
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> productIds = getProducts(agreementId);

  ReferenceCodesUtils.formatOutput(productIds);
 }

 public static List<String> getProducts(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  List<String> productIds = new ArrayList<String>();
  for (Resource resource :
 describeAgreementResponse.proposalSummary().resources()) {
   productIds.add(resource.id() + ":" + resource.type());
```

```
  }
  return productIds;
 }
}
```

- Para obter detalhes da API, consulte DescribeAgreementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the Product Type of the product the agreement was created on
AG-11
"""

import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"


def get_agreement_information(mp_client, entity_id):
    """
```

```
    Returns information about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        agreement = mp_client.describe_agreement(agreementId=entity_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


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

    print("-" * 88)
    print("Looking for offer and product details in a given agreement by
 agreement id.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    agreement = get_agreement_information(mp_client, AGREEMENT_ID)

    if agreement is not None:
        for resource in agreement["proposalSummary"]["resources"]:
            print(f"Product ID: {resource['id']}  |  Product Type:
 {resource['type']}")
    else:
        print("Agreement with ID " + AGREEMENT_ID + " is not found")


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeAgreementReferência da API AWS SDK for Python (Boto3).

Obtenha o status de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter o status de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementStatus {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Agreement status is " +
 describeAgreementResponse.status());

 }
```

```
 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Para obter detalhes da API, consulte DescribeAgreementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreement status
```

```
AG-13

Example Usage: python3 get_agreement_status.py --agreement-id <agreement-id>
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)


def get_agreement(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
        return response
    except ClientError as e:
        logger.error(f"Could not complete search_agreements request. {e}")

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement(agreement_id=args.agreement_id)

    if response is not None:
        print(f"Agreement status: {response['status']}")
    else:
        print(f"No agreement found for {args.agreement_id}")
```

- Para obter detalhes da API, consulte a DescribeAgreementReferência da API AWS SDK for Python (Boto3).

Obtenha os termos de suporte de um contrato usando um AWS SDK

Os exemplos de código a seguir mostram como obter os termos de suporte de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.SupportTerm;

import java.util.ArrayList;
import java.util.List;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsSupportTerm {

 /*
```

```
  * Obtain the support and refund policy I have provided to the customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<SupportTerm> supportTerms = getSupportTerms(agreementId);

  ReferenceCodesUtils.formatOutput(supportTerms);
 }

 public static List<SupportTerm> getSupportTerms(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<SupportTerm> supportTerms = new ArrayList<>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.supportTerm() != null) {
    supportTerms.add(acceptedTerm.supportTerm());
   }
  }
  return supportTerms;
 }

}
```

- Para obter detalhes da API, consulte GetAgreementTermsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the support and refund policy I have provided to the customer for an
 agreement
AG-19

Example Usage: python3 get_agreement_support_terms.py --agreement-id <agreement-
id>
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_terms(agreement_id):
    try:
```

```
        agreement = mp_client.get_agreement_terms(agreementId=agreement_id)
        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)

        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_terms(agreement_id=args.agreement_id)

    if agreement is not None:
        support_found = False

        for term in agreement["acceptedTerms"]:
            if "supportTerm" in term.keys():
                helper.pretty_print_datetime(term)
                support_found = True

        if not support_found:
            print(f"No support term found for agreement: {args.agreement_id}")
```

- Para obter detalhes da API, consulte a GetAgreementTermsReferência da API AWS SDK for Python (Boto3).

Obtenha os termos de um contrato usando um AWS SDK

O exemplo de código a seguir mostra como obter os termos de um contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementTerms {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  GetAgreementTermsResponse getAgreementTermsResponse =
 getAgreementTermsResponse(agreementId);

  ReferenceCodesUtils.formatOutput(getAgreementTermsResponse);

 }

 public static GetAgreementTermsResponse getAgreementTermsResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  return getAgreementTermsResponse;
 }

}
```

- Para obter detalhes da API, consulte GetAgreementTermsa Referência AWS SDK for Java 2.x da API.

Listar solicitações de cancelamento de contrato usando um AWS SDK

Os exemplos de código a seguir mostram como listar as solicitações de cancelamento de contratos dos quais participo como aceitantes.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementCancellationRequestS
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementCancellationRequ
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementCancellationRequ

public class ListAgreementCancellationRequests {

    private static final String PARTY_TYPE = "Proposer";

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

    private static void listAgreementCancellationRequests() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        String nextToken = null;

        do {
            ListAgreementCancellationRequestsRequest request =
                    ListAgreementCancellationRequestsRequest.builder()
                            .partyType(PARTY_TYPE)
                            .nextToken(nextToken)
                            .build();

            ListAgreementCancellationRequestsResponse response =

 marketplaceAgreementClient.listAgreementCancellationRequests(request);

            for (AgreementCancellationRequestSummary summary : response.items())
 {
                System.out.println("Cancellation Request ID: " +
 summary.agreementCancellationRequestId());
                System.out.println("Agreement ID: " + summary.agreementId());
                System.out.println("Status: " + summary.statusAsString());
```

```
                System.out.println("Reason Code: " +
 summary.reasonCodeAsString());
                System.out.println("Agreement Type: " + summary.agreementType());
                System.out.println("Catalog: " + summary.catalog());
                System.out.println("Created At: " + summary.createdAt());
                System.out.println("Updated At: " + summary.updatedAt());
                System.out.println("---");
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    }
}
```

- Para obter detalhes da API, consulte ListAgreementCancellationRequestsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    ListAgreementCancellationRequestsCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const PARTY_TYPE = "Proposer";

async function listAgreementCancellationRequests() {
    const client = new MarketplaceAgreementClient();

    let nextToken = null;
```

```
    do {
        const response = await client.send(
            new ListAgreementCancellationRequestsCommand({
                partyType: PARTY_TYPE,
                nextToken: nextToken,
            })
        );

        for (const summary of response.items) {
            console.log("Cancellation Request ID: " +
 summary.agreementCancellationRequestId);
            console.log("Agreement ID: " + summary.agreementId);
            console.log("Status: " + summary.status);
            console.log("Reason Code: " + summary.reasonCode);
            console.log("Agreement Type: " + summary.agreementType);
            console.log("Catalog: " + summary.catalog);
            console.log("Created At: " + summary.createdAt);
            console.log("Updated At: " + summary.updatedAt);
            console.log("---");
        }

        nextToken = response.nextToken;
    } while (nextToken != null);
}

listAgreementCancellationRequests();
```

- Para obter detalhes da API, consulte ListAgreementCancellationRequestsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class ListAgreementCancellationRequests:

    PARTY_TYPE = "Proposer"

    @staticmethod
    def list_agreement_cancellation_requests():
        client = boto3.client("marketplace-agreement")

        next_token = None

        while True:
            kwargs = {"partyType": ListAgreementCancellationRequests.PARTY_TYPE}
            if next_token:
                kwargs["nextToken"] = next_token

            response = client.list_agreement_cancellation_requests(**kwargs)

            for summary in response.get("items", []):
                print("Cancellation Request ID: " +
 summary["agreementCancellationRequestId"])
                print("Agreement ID: " + summary["agreementId"])
                print("Status: " + str(summary.get("status", "")))
                print("Reason Code: " + str(summary.get("reasonCode", "")))
                print("Agreement Type: " + str(summary.get("agreementType", "")))
                print("Catalog: " + str(summary.get("catalog", "")))
                print("Created At: " + str(summary.get("createdAt", "")))
                print("Updated At: " + str(summary.get("updatedAt", "")))
                print("---")

            next_token = response.get("nextToken")
            if not next_token:
                break


if __name__ == "__main__":
```

```
    ListAgreementCancellationRequests.list_agreement_cancellation_requests()
```

- Para obter detalhes da API, consulte a ListAgreementCancellationRequestsReferência da API AWS SDK for Python (Boto3).

Liste as solicitações de pagamento do contrato usando um AWS SDK

Os exemplos de código a seguir mostram como listar as solicitações de pagamento de contratos dos quais participo como aceitantes.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementPaymentRequestsR
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementPaymentRequestsR
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentRequestSummary;

public class ListAgreementPaymentRequests {

    private static final String PARTY_TYPE = "Proposer";

    public static void main(String[] args) {
        listAgreementPaymentRequests();
```

```
    }

    private static void listAgreementPaymentRequests() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        String nextToken = null;

        do {
            ListAgreementPaymentRequestsRequest request =
                    ListAgreementPaymentRequestsRequest.builder()
                            .partyType(PARTY_TYPE)
                            .nextToken(nextToken)
                            .build();

            ListAgreementPaymentRequestsResponse response =

 marketplaceAgreementClient.listAgreementPaymentRequests(request);

            for (PaymentRequestSummary summary : response.items()) {
                System.out.println("Payment Request ID: " +
 summary.paymentRequestId());
                System.out.println("Agreement ID: " + summary.agreementId());
                System.out.println("Status: " + summary.statusAsString());
                System.out.println("Name: " + summary.name());
                System.out.println("Charge ID: " + summary.chargeId());
                System.out.println("Charge Amount: " + summary.chargeAmount());
                System.out.println("Currency Code: " + summary.currencyCode());
                System.out.println("Created At: " + summary.createdAt());
                System.out.println("Updated At: " + summary.updatedAt());
                System.out.println("---");
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    }
}
```

- Para obter detalhes da API, consulte ListAgreementPaymentRequestsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    ListAgreementPaymentRequestsCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const PARTY_TYPE = "Proposer";

async function listAgreementPaymentRequests() {
    const client = new MarketplaceAgreementClient();

    let nextToken = null;

    do {
        const response = await client.send(
            new ListAgreementPaymentRequestsCommand({
                partyType: PARTY_TYPE,
                nextToken: nextToken,
            })
        );

        for (const summary of response.items) {
            console.log("Payment Request ID: " + summary.paymentRequestId);
            console.log("Agreement ID: " + summary.agreementId);
            console.log("Status: " + summary.status);
            console.log("Name: " + summary.name);
            console.log("Charge ID: " + summary.chargeId);
            console.log("Charge Amount: " + summary.chargeAmount);
```

```
            console.log("Currency Code: " + summary.currencyCode);
            console.log("Created At: " + summary.createdAt);
            console.log("Updated At: " + summary.updatedAt);
            console.log("---");
        }

        nextToken = response.nextToken;
    } while (nextToken != null);
}

listAgreementPaymentRequests();
```

- Para obter detalhes da API, consulte ListAgreementPaymentRequestsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class ListAgreementPaymentRequests:

    PARTY_TYPE = "Proposer"

    @staticmethod
    def list_agreement_payment_requests():
```

```
        client = boto3.client("marketplace-agreement")

        next_token = None

        while True:
            kwargs = {"partyType": ListAgreementPaymentRequests.PARTY_TYPE}
            if next_token:
                kwargs["nextToken"] = next_token

            response = client.list_agreement_payment_requests(**kwargs)

            for summary in response.get("items", []):
                print("Payment Request ID: " + summary["paymentRequestId"])
                print("Agreement ID: " + summary["agreementId"])
                print("Status: " + str(summary.get("status", "")))
                print("Name: " + str(summary.get("name", "")))
                print("Charge ID: " + str(summary.get("chargeId", "")))
                print("Charge Amount: " + str(summary.get("chargeAmount", "")))
                print("Currency Code: " + str(summary.get("currencyCode", "")))
                print("Created At: " + str(summary.get("createdAt", "")))
                print("Updated At: " + str(summary.get("updatedAt", "")))
                print("---")

            next_token = response.get("nextToken")
            if not next_token:
                break


if __name__ == "__main__":
    ListAgreementPaymentRequests.list_agreement_payment_requests()
```

- Para obter detalhes da API, consulte a ListAgreementPaymentRequestsReferência da API AWS SDK for Python (Boto3).

Rejeitar uma solicitação de cancelamento de contrato usando um AWS SDK

Os exemplos de código a seguir mostram como rejeitar uma solicitação de cancelamento de contrato iniciada pelo vendedor.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementCancellationRe
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementCancellationRe

public class RejectAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";
    private static final String REJECTION_REASON = "<REJECTION REASON HERE>";

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

    private static void rejectAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RejectAgreementCancellationRequestRequest request =
                RejectAgreementCancellationRequestRequest.builder()
```

```
                        .agreementId(AGREEMENT_ID)

 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .rejectionReason(REJECTION_REASON)
                        .build();

        RejectAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.rejectAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte RejectAgreementCancellationRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    RejectAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
```

```
const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";
const REJECTION_REASON = "<REJECTION REASON HERE>";

async function rejectAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new RejectAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
            rejectionReason: REJECTION_REASON,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

rejectAgreementCancellationRequest();
```

- Para obter detalhes da API, consulte RejectAgreementCancellationRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class RejectAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>"
    REJECTION_REASON = "<REJECTION REASON HERE>"

    @staticmethod
    def reject_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.reject_agreement_cancellation_request(
            agreementId=RejectAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=RejectAgreementCancellationRequest.AGREEMENT_CANCELLATION
            rejectionReason=RejectAgreementCancellationRequest.REJECTION_REASON,
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    RejectAgreementCancellationRequest.reject_agreement_cancellation_request()
```

- Para obter detalhes da API, consulte a RejectAgreementCancellationRequestReferência da API AWS SDK for Python (Boto3).

Rejeitar uma solicitação de pagamento de contrato usando um AWS SDK

Os exemplos de código a seguir mostram como rejeitar uma solicitação de pagamento do contrato iniciada pelo vendedor.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementPaymentRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementPaymentRequest

public class RejectAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
    private static final String REJECTION_REASON = "<REJECTION REASON HERE>";

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

    private static void rejectAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
```

```
                        .build();

        RejectAgreementPaymentRequestRequest request =
                RejectAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .rejectionReason(REJECTION_REASON)
                        .build();

        RejectAgreementPaymentRequestResponse response =

 marketplaceAgreementClient.rejectAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Name: " + response.name());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte RejectAgreementPaymentRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
```

```
    MarketplaceAgreementClient,
    RejectAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
const REJECTION_REASON = "<REJECTION REASON HERE>";

async function rejectAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new RejectAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
            rejectionReason: REJECTION_REASON,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Name: " + response.name);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

rejectAgreementPaymentRequest();
```

- Para obter detalhes da API, consulte RejectAgreementPaymentRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class RejectAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"
    REJECTION_REASON = "<REJECTION REASON HERE>"

    @staticmethod
    def reject_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.reject_agreement_payment_request(
            agreementId=RejectAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=RejectAgreementPaymentRequest.PAYMENT_REQUEST_ID,
            rejectionReason=RejectAgreementPaymentRequest.REJECTION_REASON,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
```

```
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    RejectAgreementPaymentRequest.reject_agreement_payment_request()
```

- Para obter detalhes da API, consulte a RejectAgreementPaymentRequestReferência da API AWS SDK for Python (Boto3).

Substitua um contrato de SaaS por uma oferta baseada em contrato usando um AWS SDK

Os exemplos de código a seguir mostram como substituir um contrato de SaaS por uma nova oferta baseada em contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
 * Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model
 and receives a private
 * Agreement-Based Offer from the seller. The buyer replaces the existing
 agreement
 * with the new ABO offer, which may include updated pricing terms and payment
 schedule.
 *
 * <p>Note: Unlike other Replace samples, this example starts from an already-
active
 * {@code EXISTING_AGREEMENT_ID} rather than creating a new agreement first. Use
 this
 * pattern when you already have an agreement ID and want to replace it directly.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code EXISTING_AGREEMENT_ID} — the agreement ID of the active agreement
 to replace (starts with {@code agmt-}).</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new ABO offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the new offer's term
 list.</li>
 * </ul>
 */
public class ReplaceSaaSContractWithAgreementBasedOffer {

    // The agreement ID of the active SaaS agreement with CONTRACT pricing model
 to replace (starts with "agmt-").
    private static final String EXISTING_AGREEMENT_ID = "<your-existing-
agreement-id>";

    // The agreementProposalId from the new Agreement-Based Offer.
```

```
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the LegalTerm in the new offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the FixedUpfrontPricingTerm in the new offer.
    private static final String FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-
upfront-pricing-term-id>";

    // Term ID for the PaymentScheduleTerm in the new offer.
    private static final String PAYMENT_SCHEDULE_TERM_ID = "<your-payment-
schedule-term-id>";

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

    /**
     * Replaces an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.
     * Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
     */
    private static void replaceExistingAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm legalTerm =
 RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm validityTerm =
 RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm fixedUpfrontPricingTerm =
 RequestedTerm.builder().id(FIXED_UPFRONT_PRICING_TERM_ID).build();
        RequestedTerm paymentScheduleTerm =
 RequestedTerm.builder().id(PAYMENT_SCHEDULE_TERM_ID).build();

        // Replace the agreement with the new offer
```

```
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(fixedUpfrontPricingTerm,
 paymentScheduleTerm, legalTerm, validityTerm)

 .agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)
                        .sourceAgreementIdentifier(EXISTING_AGREEMENT_ID)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Replace agreement request created.
 AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement replaced with ABO. New AgreementId: " +
 acceptAgreementRequestResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
 * Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
 * Agreement-Based Offer from the seller. The buyer replaces the existing
 agreement
 * with the new ABO offer, which may include updated pricing terms and payment
 schedule.
 *
 * Note: Unlike other Replace samples, this example starts from an already-active
 * EXISTING_AGREEMENT_ID rather than creating a new agreement first. Use this
 * pattern when you already have an agreement ID and want to replace it directly.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 *   - EXISTING_AGREEMENT_ID — the agreement ID of the active agreement to
 replace (starts with "agmt-").
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 ABO offer.
 *   - Term IDs (starting with "term-") — found in the new offer's term list.
```

```
 */

// The agreement ID of the active SaaS agreement with CONTRACT pricing model to
 replace (starts with "agmt-").
const EXISTING_AGREEMENT_ID = "<your-existing-agreement-id>";

// The agreementProposalId from the new Agreement-Based Offer.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the LegalTerm in the new offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in the new offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the FixedUpfrontPricingTerm in the new offer.
const FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>";

// Term ID for the PaymentScheduleTerm in the new offer.
const PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>";

/**
 * Replaces an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.
 * Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
 */
async function replaceExistingAgreement() {
    const client = new MarketplaceAgreementClient();

    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const fixedUpfrontPricingTerm = { id: FIXED_UPFRONT_PRICING_TERM_ID };
    const paymentScheduleTerm = { id: PAYMENT_SCHEDULE_TERM_ID };

    // Replace the agreement with the new offer
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [fixedUpfrontPricingTerm, paymentScheduleTerm,
 legalTerm, validityTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
```

```
            sourceAgreementIdentifier: EXISTING_AGREEMENT_ID,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement replaced with ABO. New AgreementId: " +
 acceptAgreementRequestResponse.agreementId);
}

replaceExistingAgreement();
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
Agreement-Based Offer from the seller. The buyer replaces the existing agreement
```

```
with the new ABO offer, which may include updated pricing terms and payment
 schedule.

Note: Unlike other Replace samples, this example starts from an already-active
EXISTING_AGREEMENT_ID rather than creating a new agreement first. Use this
pattern when you already have an agreement ID and want to replace it directly.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offers:
  - EXISTING_AGREEMENT_ID — the agreement ID of the active agreement to replace
 (starts with agmt-).
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new ABO
 offer.
  - Term IDs (starting with term-) — found in the new offer's term list.
"""

import boto3

from utils.agreement_api_utils import generate_client_token


class ReplaceSaaSContractWithAgreementBasedOffer:

    # The agreement ID of the active SaaS agreement with CONTRACT pricing model
 to replace (starts with "agmt-").
    EXISTING_AGREEMENT_ID = "<your-existing-agreement-id>"

    # The agreementProposalId from the new Agreement-Based Offer.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>"

    # Term ID for the LegalTerm in the new offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the FixedUpfrontPricingTerm in the new offer.
    FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>"

    # Term ID for the PaymentScheduleTerm in the new offer.
    PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>"
```

```
    @staticmethod
    def replace_existing_agreement():
        """
        Replace an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.

        Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSContractWithAgreementBasedOffer

        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        fixed_upfront_pricing_term = {"id": cls.FIXED_UPFRONT_PRICING_TERM_ID}
        payment_schedule_term = {"id": cls.PAYMENT_SCHEDULE_TERM_ID}

        # Replace the agreement with the new offer
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[fixed_upfront_pricing_term, payment_schedule_term,
 legal_term, validity_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=cls.EXISTING_AGREEMENT_ID,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Replace agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print("Agreement replaced with ABO. New AgreementId: " +
 accept_response["agreementId"])


if __name__ == "__main__":
    ReplaceSaaSContractWithAgreementBasedOffer.replace_existing_agreement()
```

- Para obter detalhes da API, consulte a CreateAgreementRequestReferência da API AWS SDK for Python (Boto3).

Substitua uma avaliação gratuita de SaaS por Contrato com preços de consumo usando um AWS SDK

Os exemplos de código a seguir mostram como substituir um teste gratuito de SaaS por um Contrato com Preços de Consumo (CCP).

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS free trial agreement and then replace it
 with a
 * paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.
```

```
 *
 * <p>Scenario: A buyer first starts a free trial on a SaaS product. Once the
 trial is active,
 * they decide to convert by replacing it with a paid offer that includes a
 * FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.
 *
 * <p>Flow:
 * <ol>
 *   <li>Create a SaaS free trial agreement with freeTrialPricingTerm.</li>
 *   <li>Wait for freeTrialPricingTerm agreement entitlements to become active.</
li>
 *   <li>Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 free trial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the paid CCP offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term
 list.</li>
 * </ul>
 */
public class ReplaceSaaSFreeTrialWithCCP {

    // The agreementProposalId from the free trial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-
trial-agreement-proposal-identifier>";

    // Term ID for the FreeTrialPricingTerm in the free trial offer.
    private static final String FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-
pricing-term-id>";

    // Term ID for the LegalTerm in the free trial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // The agreementProposalId from the paid CCP offer to convert to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-
agreement-proposal-identifier>";
```

```
    // Term ID for the FixedUpfrontPricingTerm in the CCP offer.
    private static final String FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-
upfront-pricing-term-id>";

    // Term ID for the PaymentScheduleTerm in the CCP offer.
    private static final String PAYMENT_SCHEDULE_TERM_ID = "<your-payment-
schedule-term-id>";

    // Term ID for the UsageBasedPricingTerm in the CCP offer.
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the ValidityTerm in the CCP offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the LegalTerm in the CCP offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

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

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS free trial agreement with freeTrialPricingTerm.
     * 2. Wait for freeTrialPricingTerm agreement entitlements to become active.
     * 3. Replace the free trial agreement with the paid CCP offer.
     */
    private static void createSaaSFreeTrialAndReplaceWithCCP() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm legalTerm =
 RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm freeTrialPricingTerm =
 RequestedTerm.builder().id(FREE_TRIAL_PRICING_TERM_ID).build();

        // --- Step 1: Agreement with freeTrialPricingTerm ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
```

```
                        .intent(Intent.NEW)
                        .requestedTerms(freeTrialPricingTerm, legalTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
       CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
       System.out.println("Agreement request with freeTrialPricingTerm created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

       AcceptAgreementRequestRequest acceptAgreementRequestRequest =
               AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
       AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
       System.out.println("Agreement request with freeTrialPricingTerm accepted.
AgreementId: " + acceptAgreementRequestResponse.agreementId());

       // Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
       System.out.println("Waiting for freeTrialPricingTerm agreement
entitlements to become active...");
       GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
               marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
       System.out.println("freeTrialPricingTerm agreement entitlements are now
active.");
       AgreementApiUtils.formatOutput(entitlementsResponse);

       // --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
       // Use Intent.REPLACE and sourceAgreementIdentifier to replace the free
trial agreement.
       RequestedTerm usageBasedPricingTerm =
RequestedTerm.builder().id(USAGE_BASED_PRICING_TERM_ID).build();
       RequestedTerm fixedUpfrontPricingTerm =
RequestedTerm.builder().id(FIXED_UPFRONT_PRICING_TERM_ID).build();
       RequestedTerm paymentScheduleTerm =
RequestedTerm.builder().id(PAYMENT_SCHEDULE_TERM_ID).build();
```

```
        RequestedTerm validityTerm =
 RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm newLegalTerm =
 RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(usageBasedPricingTerm,
 fixedUpfrontPricingTerm, paymentScheduleTerm, validityTerm, newLegalTerm)

 .agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)

 .sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Replace agreement request created.
 AgreementRequestId: " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        System.out.println("Agreement with freeTrialPricingTerm replaced with
 paid CCP offer. New AgreementId: " + aarResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS free trial agreement and then replace it
 with a
 * paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.
 *
 * Scenario: A buyer first starts a free trial on a SaaS product. Once the trial
 is active,
 * they decide to convert by replacing it with a paid offer that includes a
 * FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.
 *
 * Flow:
 *   1. Create a SaaS free trial agreement with freeTrialPricingTerm.
 *   2. Wait for freeTrialPricingTerm agreement entitlements to become active.
 *   3. Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the free
 trial offer.
```

```
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the paid
 CCP offer.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 */

// The agreementProposalId from the free trial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-agreement-proposal-
identifier>";

// Term ID for the FreeTrialPricingTerm in the free trial offer.
const FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>";

// Term ID for the LegalTerm in the free trial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// The agreementProposalId from the paid CCP offer to convert to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-agreement-proposal-
identifier>";

// Term ID for the FixedUpfrontPricingTerm in the CCP offer.
const FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>";

// Term ID for the PaymentScheduleTerm in the CCP offer.
const PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>";

// Term ID for the UsageBasedPricingTerm in the CCP offer.
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the CCP offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the LegalTerm in the CCP offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS free trial agreement with freeTrialPricingTerm.
 * 2. Wait for freeTrialPricingTerm agreement entitlements to become active.
 * 3. Replace the free trial agreement with the paid CCP offer.
 */
async function createSaaSFreeTrialAndReplaceWithCCP() {
    const client = new MarketplaceAgreementClient();

    const legalTerm = { id: LEGAL_TERM_ID };
```

```
   const freeTrialPricingTerm = { id: FREE_TRIAL_PRICING_TERM_ID };

   // --- Step 1: Agreement with freeTrialPricingTerm ---
   const createAgreementRequestResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "NEW",
           requestedTerms: [freeTrialPricingTerm, legalTerm],
           agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
       })
   );
   console.log("Agreement request with freeTrialPricingTerm created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId);

   const acceptAgreementRequestResponse = await client.send(
       new AcceptAgreementRequestCommand({
           agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
       })
   );
   console.log("Agreement request with freeTrialPricingTerm accepted.
AgreementId: " + acceptAgreementRequestResponse.agreementId);

   // Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
   console.log("Waiting for freeTrialPricingTerm agreement entitlements to
become active...");
   const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
   console.log("freeTrialPricingTerm agreement entitlements are now active.");
   formatOutput(entitlementsResponse);

   // --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
   // Use Intent.REPLACE and sourceAgreementIdentifier to replace the free trial
agreement.
   const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
   const fixedUpfrontPricingTerm = { id: FIXED_UPFRONT_PRICING_TERM_ID };
   const paymentScheduleTerm = { id: PAYMENT_SCHEDULE_TERM_ID };
   const validityTerm = { id: VALIDITY_TERM_ID };
   const newLegalTerm = { id: NEW_LEGAL_TERM_ID };

   const carResponse = await client.send(
       new CreateAgreementRequestCommand({
```

```
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [usageBasedPricingTerm, fixedUpfrontPricingTerm,
 paymentScheduleTerm, validityTerm, newLegalTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier:
 acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    console.log("Agreement with freeTrialPricingTerm replaced with paid CCP
 offer. New AgreementId: " + aarResponse.agreementId);
}

createSaaSFreeTrialAndReplaceWithCCP();
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create a SaaS free trial agreement and then replace it with a
```

```
paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.

Scenario: A buyer first starts a free trial on a SaaS product. Once the trial is
 active,
they decide to convert by replacing it with a paid offer that includes a
FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.

Flow:
  1. Create a SaaS free trial agreement with freeTrialPricingTerm.
  2. Wait for freeTrialPricingTerm agreement entitlements to become active.
  3. Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the free trial
 offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the paid CCP
 offer.
  - Term IDs (starting with term-) — found in each offer's term list.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class ReplaceSaaSFreeTrialWithCCP:

    # The agreementProposalId from the free trial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-agreement-proposal-
identifier>"

    # Term ID for the FreeTrialPricingTerm in the free trial offer.
    FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>"

    # Term ID for the LegalTerm in the free trial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"
```

```
    # The agreementProposalId from the paid CCP offer to convert to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-agreement-proposal-
identifier>"

    # Term ID for the FixedUpfrontPricingTerm in the CCP offer.
    FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>"

    # Term ID for the PaymentScheduleTerm in the CCP offer.
    PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>"

    # Term ID for the UsageBasedPricingTerm in the CCP offer.
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the CCP offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the LegalTerm in the CCP offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    @staticmethod
    def create_saas_free_trial_and_replace_with_ccp():
        """
        Full end-to-end flow:
        1. Create a SaaS free trial agreement with freeTrialPricingTerm.
        2. Wait for freeTrialPricingTerm agreement entitlements to become active.
        3. Replace the free trial agreement with the paid CCP offer.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSFreeTrialWithCCP

        legal_term = {"id": cls.LEGAL_TERM_ID}
        free_trial_pricing_term = {"id": cls.FREE_TRIAL_PRICING_TERM_ID}

        # --- Step 1: Agreement with freeTrialPricingTerm ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[free_trial_pricing_term, legal_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with freeTrialPricingTerm created.
 AgreementRequestId: " + agreement_request_id)
```

```
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with freeTrialPricingTerm accepted. AgreementId:
" + agreement_id)

        # Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
        print("Waiting for freeTrialPricingTerm agreement entitlements to become
active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("freeTrialPricingTerm agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
        # Use Intent.REPLACE and sourceAgreementIdentifier to replace the free
trial agreement.
        usage_based_pricing_term = {"id": cls.USAGE_BASED_PRICING_TERM_ID}
        fixed_upfront_pricing_term = {"id": cls.FIXED_UPFRONT_PRICING_TERM_ID}
        payment_schedule_term = {"id": cls.PAYMENT_SCHEDULE_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[
                usage_based_pricing_term, fixed_upfront_pricing_term,
                payment_schedule_term, validity_term, new_legal_term,
            ],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        print(
```

```
            "Agreement with freeTrialPricingTerm replaced with paid CCP offer.
 New AgreementId: "
            + aar_response["agreementId"]
        )


if __name__ == "__main__":
    ReplaceSaaSFreeTrialWithCCP.create_saas_free_trial_and_replace_with_ccp()
```

- Para obter detalhes da API, consulte a CreateAgreementRequestReferência da API AWS SDK for Python (Boto3).

Substitua um termo de preço baseado no uso de SaaS e cancele o contrato anterior usando um AWS SDK

Os exemplos de código a seguir mostram como substituir um termo de preço baseado no uso de SaaS e cancelar o contrato anterior.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.CancelAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT)
 and then replace it
 * with a new offer using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
 * The buyer then converts to a different offer by replacing the existing
 agreement.
 *
 * <p>Flow:
 * <ol>
 *   <li>Create and accept the initial agreement request with UBPT.</li>
 *   <li>Wait for entitlements to become active.</li>
 *   <li>Replace the agreement with a new offer using Intent.REPLACE.</li>
 *   <li>Cancel the new agreement using CancelAgreement.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 initial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new offer to replace to.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term
 list.</li>
 * </ul>
 */
public class ReplaceSaaSUsageBasedPricingTermAndCancel {

    // The agreementProposalId from the initial offer.
```

```
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the initial offer.
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the ValidityTerm in the initial offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the LegalTerm in the initial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // The agreementProposalId from the new offer to replace to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the new offer.
    private static final String NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-
usage-based-pricing-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String NEW_VALIDITY_TERM_ID = "<your-new-validity-term-
id>";

    // Term ID for the LegalTerm in the new offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

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

    /**
     * Full end-to-end flow:
     * 1. Create and accept the initial agreement request with
 UsageBasedPricingTerm.
     * 2. Wait for entitlements to become active.
     * 3. Replace the agreement with a new offer.
     * 4. Cancel the new agreement.
     */
    private static void replaceSaaSUbptAndCancel() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
```

```
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageBasedPricingTerm =
RequestedTerm.builder().id(USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm legalTerm =
RequestedTerm.builder().id(LEGAL_TERM_ID).build();

        // --- Step 1: Create and accept the initial UBPT agreement request ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageBasedPricingTerm, validityTerm,
legalTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with UBPT accepted. AgreementId: "
+ acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before replacing.
        System.out.println("Waiting for entitlements to become active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("Entitlements are now active.");
```

```
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Replace the UBPT agreement with a new offer ---
        // Use Intent.REPLACE and sourceAgreementIdentifier to replace the
existing agreement.
        RequestedTerm newUsageBasedPricingTerm =
RequestedTerm.builder().id(NEW_USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm newValidityTerm =
RequestedTerm.builder().id(NEW_VALIDITY_TERM_ID).build();
        RequestedTerm newLegalTerm =
RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(newUsageBasedPricingTerm,
newValidityTerm, newLegalTerm)

.agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)

.sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Replace agreement request created.
AgreementRequestId: " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String agreementIdWithTheNewOffer = aarResponse.agreementId();
        System.out.println("UBPT agreement replaced. New AgreementId: " +
agreementIdWithTheNewOffer);

        // --- Step 3: Cancel the new agreement ---
        CancelAgreementRequest cancelAgreementRequest =
                CancelAgreementRequest.builder()
                        .agreementId(agreementIdWithTheNewOffer)
                        .build();
        marketplaceAgreementClient.cancelAgreement(cancelAgreementRequest);
```

```
        System.out.println("The new agreement has been cancelled. AgreementId: "
 + agreementIdWithTheNewOffer);
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
    CancelAgreementCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT)
 and then replace it
 * with a new offer using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
 * The buyer then converts to a different offer by replacing the existing
 agreement.
 *
 * Flow:
 *   1. Create and accept the initial agreement request with UBPT.
 *   2. Wait for entitlements to become active.
```

```
 *   3. Replace the agreement with a new offer using Intent.REPLACE.
 *   4. Cancel the new agreement using CancelAgreement.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 */

// The agreementProposalId from the initial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the UsageBasedPricingTerm in the initial offer.
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the initial offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the LegalTerm in the initial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// The agreementProposalId from the new offer to replace to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the UsageBasedPricingTerm in the new offer.
const NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the new offer.
const NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>";

// Term ID for the LegalTerm in the new offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create and accept the initial agreement request with UsageBasedPricingTerm.
 * 2. Wait for entitlements to become active.
 * 3. Replace the agreement with a new offer.
 * 4. Cancel the new agreement.
```

```
 */
async function replaceSaaSUbptAndCancel() {
    const client = new MarketplaceAgreementClient();

    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };

    // --- Step 1: Create and accept the initial UBPT agreement request ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageBasedPricingTerm, validityTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with UBPT accepted. AgreementId: " +
 acceptAgreementRequestResponse.agreementId);

    // Wait for entitlements to become active before replacing.
    console.log("Waiting for entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
 acceptAgreementRequestResponse.agreementId);
    console.log("Entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Step 2: Replace the UBPT agreement with a new offer ---
    // Use Intent.REPLACE and sourceAgreementIdentifier to replace the existing
 agreement.
    const newUsageBasedPricingTerm = { id: NEW_USAGE_BASED_PRICING_TERM_ID };
    const newValidityTerm = { id: NEW_VALIDITY_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };

    const carResponse = await client.send(
```

```
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [newUsageBasedPricingTerm, newValidityTerm,
 newLegalTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier:
 acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const agreementIdWithTheNewOffer = aarResponse.agreementId;
    console.log("UBPT agreement replaced. New AgreementId: " +
 agreementIdWithTheNewOffer);

    // --- Step 3: Cancel the new agreement ---
    await client.send(
        new CancelAgreementCommand({
            agreementId: agreementIdWithTheNewOffer,
        })
    );
    console.log("The new agreement has been cancelled. AgreementId: " +
 agreementIdWithTheNewOffer);
}

replaceSaaSUbptAndCancel();
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT) and
 then replace it
with a new offer using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
The buyer then converts to a different offer by replacing the existing agreement.

Flow:
  1. Create and accept the initial agreement request with UBPT.
  2. Wait for entitlements to become active.
  3. Replace the agreement with a new offer using Intent.REPLACE.
  4. Cancel the new agreement using CancelAgreement.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
  - Term IDs (starting with term-) — found in each offer's term list.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)
```

```
class ReplaceSaaSUsageBasedPricingTermAndCancel:

    # The agreementProposalId from the initial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the UsageBasedPricingTerm in the initial offer.
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the initial offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the LegalTerm in the initial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # The agreementProposalId from the new offer to replace to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>"

    # Term ID for the UsageBasedPricingTerm in the new offer.
    NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>"

    # Term ID for the LegalTerm in the new offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    @staticmethod
    def replace_saas_ubpt_and_cancel():
        """
        Full end-to-end flow:
        1. Create and accept the initial agreement request with
 UsageBasedPricingTerm.
        2. Wait for entitlements to become active.
        3. Replace the agreement with a new offer.
        4. Cancel the new agreement.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSUsageBasedPricingTermAndCancel

        usage_based_pricing_term = {"id": cls.USAGE_BASED_PRICING_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
```

```
        legal_term = {"id": cls.LEGAL_TERM_ID}

        # --- Step 1: Create and accept the initial UBPT agreement request ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_based_pricing_term, validity_term, legal_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " +
agreement_id)

        # Wait for entitlements to become active before replacing.
        print("Waiting for entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("Entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Replace the UBPT agreement with a new offer ---
        # Use Intent.REPLACE and sourceAgreementIdentifier to replace the
existing agreement.
        new_usage_based_pricing_term = {"id":
cls.NEW_USAGE_BASED_PRICING_TERM_ID}
        new_validity_term = {"id": cls.NEW_VALIDITY_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[new_usage_based_pricing_term, new_validity_term,
new_legal_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=agreement_id,
        )
```

```
        print("Replace agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        agreement_id_with_new_offer = aar_response["agreementId"]
        print("UBPT agreement replaced. New AgreementId: " +
 agreement_id_with_new_offer)

        # --- Step 3: Cancel the new agreement ---
        client.cancel_agreement(agreementId=agreement_id_with_new_offer)
        print("The new agreement has been cancelled. AgreementId: " +
 agreement_id_with_new_offer)


if __name__ == "__main__":
    ReplaceSaaSUsageBasedPricingTermAndCancel.replace_saas_ubpt_and_cancel()
```

- Para obter detalhes da API, consulte a CreateAgreementRequestReferência da API AWS SDK for Python (Boto3).

Substitua um termo de preço baseado no uso da AMI, mas mantenha o existente ConfigurableUpfrontPricingTerm usando um AWS SDK

Os exemplos de código a seguir mostram como substituir um termo de preço baseado no uso da AMI e, ao mesmo tempo, manter o existente. ConfigurableUpfrontPricingTerm

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
 * the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active,
 using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: An AMI product with USAGE pricing requires two agreements:
 * <ol>
 *   <li>An agreement with <b>usageBasedPricingTerm</b> — accepted first to
 establish the base agreement.</li>
 *   <li>An agreement with <b>configurableUpfrontPricingTerm (CUPT)</b> —
 accepted after the usageBasedPricingTerm agreement entitlements are active.</li>
 * </ol>
 * <p> This sample shows how to replace only the agreement with
 usageBasedPricingTerm with a
 * new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
 *
 * <p>Flow:
 * <ol>
```

```
 *   <li>Create and accept the initial agreement request with
 usageBasedPricingTerm.</li>
 *   <li>Create and accept the agreement request with
 configurableUpfrontPricingTerm (CUPT) (after usageBasedPricingTerm agreement
 entitlements are active).</li>
 *   <li>Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 initial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new offer to replace to.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code
 P365D}).</li>
 *   <li>{@code DIMENSION_1_KEY} — dimension key defined in the CUPT term.</li>
 * </ul>
 */
public class ReplaceAmiUsageBasedPricingTermButNotCupt {

    // The agreementProposalId from the initial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P365D" for 365 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // Dimension key defined in the CUPT term.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the UsageBasedPricingTerm in the initial offer.
```

```
    private static final String USAGE_TERM_ID = "<your-usage-term-id>";

    // Term ID for the LegalTerm in the initial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in the initial offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // The agreementProposalId from the new offer to replace to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the new offer.
    private static final String NEW_USAGE_TERM_ID = "<your-new-usage-term-id>";

    // Term ID for the LegalTerm in the new offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String NEW_VALIDITY_TERM_ID = "<your-new-validity-term-
id>";

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

    /**
     * Full end-to-end flow:
     * 1. Create and accept an agreement request with usageBasedPricingTerm, then
 wait for entitlements.
     * 2. Create and accept an agreement request with CUPT, then wait for
 entitlements.
     * 3. Replace the agreement with usageBasedPricingTerm with a new offer
 (agreement with CUPT is unaffected).
     */
    private static void replaceAmiUsageWhenCuptExists() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageTerm =
 RequestedTerm.builder().id(USAGE_TERM_ID).build();
```

```
        RequestedTerm legalTerm =
RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();

        // --- Step 1: Agreement with UBPT ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageTerm, legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        final String usageAgreementId =
acceptAgreementRequestResponse.agreementId();
        System.out.println("Agreement request with UBPT accepted. AgreementId: "
+ usageAgreementId);

        // Wait for UBPT agreement entitlements to become active before creating
the agreement with CUPT.
        System.out.println("Waiting for UBPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, usageAgreementId);
        System.out.println("UBPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
```

```
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Agreement request with CUPT created.
AgreementRequestId: " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String cuptAgreementId = aarResponse.agreementId();
        System.out.println("Agreement request with CUPT accepted. AgreementId: "
+ cuptAgreementId);

        // Wait for CUPT agreement entitlements to become active before replacing
usage.
        System.out.println("Waiting for CUPT agreement entitlements to become
active...");
```

```
        GetAgreementEntitlementsResponse cuptEntitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, cuptAgreementId);
        System.out.println("CUPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(cuptEntitlementsResponse);

        // --- Step 3: Replace Agreement with usageBasedPricingTerm with a new
offer ---
        RequestedTerm newUsageTerm =
RequestedTerm.builder().id(NEW_USAGE_TERM_ID).build();
        RequestedTerm newLegalTerm =
RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();
        RequestedTerm newValidityTerm =
RequestedTerm.builder().id(NEW_VALIDITY_TERM_ID).build();

        // Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
usageBasedPricingTerm agreement only.
        // The agreement with configurableUpfrontPricingTerm (CUPT) is NOT
affected by this replacement.
        CreateAgreementRequestRequest carReplaceRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(newUsageTerm, newLegalTerm,
newValidityTerm)

.agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)
                        .sourceAgreementIdentifier(usageAgreementId)
                        .build();
        CreateAgreementRequestResponse carReplaceResponse =

marketplaceAgreementClient.createAgreementRequest(carReplaceRequest);
        System.out.println("Replace agreement request created.
AgreementRequestId: " + carReplaceResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarReplaceRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(carReplaceResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarReplaceResponse =

marketplaceAgreementClient.acceptAgreementRequest(aarReplaceRequest);
```

```
        System.out.println("Agreement with UBPT replaced. New AgreementId: " +
 aarReplaceResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
 * the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active,
 using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: An AMI product with USAGE pricing requires two agreements:
 *   1. An agreement with usageBasedPricingTerm — accepted first to establish the
 base agreement.
 *   2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after
 the usageBasedPricingTerm agreement entitlements are active.
 *
```

```
 * This sample shows how to replace only the agreement with usageBasedPricingTerm
 with a
 * new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
 *
 * Flow:
 *   1. Create and accept the initial agreement request with
 usageBasedPricingTerm.
 *   2. Create and accept the agreement request with
 configurableUpfrontPricingTerm (CUPT) (after usageBasedPricingTerm agreement
 entitlements are active).
 *   3. Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P365D").
 *   - DIMENSION_1_KEY — dimension key defined in the CUPT term.
 */

// The agreementProposalId from the initial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P365D" for 365 days).
const SELECTOR_VALUE = "<your-selector-value>";

// Dimension key defined in the CUPT term.
const DIMENSION_1_KEY = "<your-dimension-key>";

// Quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the UsageBasedPricingTerm in the initial offer.
const USAGE_TERM_ID = "<your-usage-term-id>";
```

```
// Term ID for the LegalTerm in the initial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in the initial offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// The agreementProposalId from the new offer to replace to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the UsageBasedPricingTerm in the new offer.
const NEW_USAGE_TERM_ID = "<your-new-usage-term-id>";

// Term ID for the LegalTerm in the new offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

// Term ID for the ValidityTerm in the new offer.
const NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create and accept an agreement request with usageBasedPricingTerm, then
 wait for entitlements.
 * 2. Create and accept an agreement request with CUPT, then wait for
 entitlements.
 * 3. Replace the agreement with usageBasedPricingTerm with a new offer
 (agreement with CUPT is unaffected).
 */
async function replaceAmiUsageWhenCuptExists() {
    const client = new MarketplaceAgreementClient();

    const usageTerm = { id: USAGE_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Step 1: Agreement with UBPT ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageTerm, legalTerm, validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
```

```
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
        })
    );
    const usageAgreementId = acceptAgreementRequestResponse.agreementId;
    console.log("Agreement request with UBPT accepted. AgreementId: " +
usageAgreementId);

    // Wait for UBPT agreement entitlements to become active before creating the
agreement with CUPT.
    console.log("Waiting for UBPT agreement entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
usageAgreementId);
    console.log("UBPT agreement entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
```

```
    );
    console.log("Agreement request with CUPT created. AgreementRequestId: " +
carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const cuptAgreementId = aarResponse.agreementId;
    console.log("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

    // Wait for CUPT agreement entitlements to become active before replacing
usage.
    console.log("Waiting for CUPT agreement entitlements to become active...");
    const cuptEntitlementsResponse = await pollUntilEntitlementsAvailable(client,
cuptAgreementId);
    console.log("CUPT agreement entitlements are now active.");
    formatOutput(cuptEntitlementsResponse);

    // --- Step 3: Replace Agreement with usageBasedPricingTerm with a new offer
---
    const newUsageTerm = { id: NEW_USAGE_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };
    const newValidityTerm = { id: NEW_VALIDITY_TERM_ID };

    // Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
usageBasedPricingTerm agreement only.
    // The agreement with configurableUpfrontPricingTerm (CUPT) is NOT affected
by this replacement.
    const carReplaceResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [newUsageTerm, newLegalTerm, newValidityTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: usageAgreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
carReplaceResponse.agreementRequestId);

    const aarReplaceResponse = await client.send(
```

```
        new AcceptAgreementRequestCommand({
            agreementRequestId: carReplaceResponse.agreementRequestId,
        })
    );
    console.log("Agreement with UBPT replaced. New AgreementId: " +
 aarReplaceResponse.agreementId);
}

replaceAmiUsageWhenCuptExists();
```

- Para obter detalhes da API, consulte CreateAgreementRequesta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active, using
 the
AWS Marketplace Agreement Service APIs.

Scenario: An AMI product with USAGE pricing requires two agreements:
  1. An agreement with usageBasedPricingTerm — accepted first to establish the
 base agreement.
  2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
     usageBasedPricingTerm agreement entitlements are active.
This sample shows how to replace only the agreement with usageBasedPricingTerm
 with a
new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
```

```
Flow:
  1. Create and accept the initial agreement request with usageBasedPricingTerm.
  2. Create and accept the agreement request with configurableUpfrontPricingTerm
 (CUPT)
     (after usageBasedPricingTerm agreement entitlements are active).
  3. Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
  - Term IDs (starting with term-) — found in each offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P365D).
  - DIMENSION_1_KEY — dimension key defined in the CUPT term.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class ReplaceAmiUsageBasedPricingTermButNotCupt:

    # The agreementProposalId from the initial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P365D" for 365 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # Dimension key defined in the CUPT term.
    DIMENSION_1_KEY = "<your-dimension-key>"
```

```
    # Quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the UsageBasedPricingTerm in the initial offer.
    USAGE_TERM_ID = "<your-usage-term-id>"

    # Term ID for the LegalTerm in the initial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in the initial offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # The agreementProposalId from the new offer to replace to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>"

    # Term ID for the UsageBasedPricingTerm in the new offer.
    NEW_USAGE_TERM_ID = "<your-new-usage-term-id>"

    # Term ID for the LegalTerm in the new offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>"

    @staticmethod
    def replace_ami_usage_when_cupt_exists():
        """
        Full end-to-end flow:
        1. Create and accept an agreement request with usageBasedPricingTerm,
 then wait for entitlements.
        2. Create and accept an agreement request with CUPT, then wait for
 entitlements.
        3. Replace the agreement with usageBasedPricingTerm with a new offer
           (agreement with CUPT is unaffected).
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceAmiUsageBasedPricingTermButNotCupt

        usage_term = {"id": cls.USAGE_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
```

```
        # --- Step 1: Agreement with UBPT ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_term, legal_term, validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        usage_agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " +
usage_agreement_id)

        # Wait for UBPT agreement entitlements to become active before creating
the agreement with CUPT.
        print("Waiting for UBPT agreement entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
usage_agreement_id)
        print("UBPT agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
```

```
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        print("Agreement request with CUPT created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        cupt_agreement_id = aar_response["agreementId"]
        print("Agreement request with CUPT accepted. AgreementId: " +
cupt_agreement_id)

        # Wait for CUPT agreement entitlements to become active before replacing
usage.
        print("Waiting for CUPT agreement entitlements to become active...")
        cupt_entitlements_response = poll_until_entitlements_available(client,
cupt_agreement_id)
        print("CUPT agreement entitlements are now active.")
        format_output(cupt_entitlements_response)

        # --- Step 3: Replace Agreement with usageBasedPricingTerm with a new
offer ---
        new_usage_term = {"id": cls.NEW_USAGE_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}
        new_validity_term = {"id": cls.NEW_VALIDITY_TERM_ID}

        # Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
usageBasedPricingTerm agreement only.
        # The agreement with configurableUpfrontPricingTerm (CUPT) is NOT
affected by this replacement.
        car_replace_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[new_usage_term, new_legal_term, new_validity_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=usage_agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
car_replace_response["agreementRequestId"])

        aar_replace_response = client.accept_agreement_request(
```

```
            agreementRequestId=car_replace_response["agreementRequestId"]
        )
        print("Agreement with UBPT replaced. New AgreementId: " +
 aar_replace_response["agreementId"])


if __name__ == "__main__":

 ReplaceAmiUsageBasedPricingTermButNotCupt.replace_ami_usage_when_cupt_exists()
```

- Para obter detalhes da API, consulte a CreateAgreementRequestReferência da API AWS SDK for Python (Boto3).

Pesquise contratos por ID da conta usando um AWS SDK

O exemplo de código a seguir mostra como pesquisar contratos por ID da conta.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get agreement by customer AWS
 account ID
AG-02
"""

import argparse
import logging
```

```
import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")
logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10


def get_agreements(account_id):
    AgreementSummaryList = []

    try:
        agreement = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you
 are the proposer.
            # Change to "Acceptor" to return agreements where you are the
 acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": "AcceptorId", "values": [account_id]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    while "nextToken" in agreement and agreement["nextToken"] is not None:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreement["nextToken"],
```

```
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {"name": "AcceptorId", "values": [account_id]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--account_id",
        "-aid",
        help="Provide accepting account ID to search for agreements",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreements(account_id=args.account_id)

    helper.pretty_print_datetime(response)
```

- Para obter detalhes da API, consulte a SearchAgreementsReferência da API AWS SDK for Python (Boto3).

Pesquise contratos por ID do contrato usando um AWS SDK

O exemplo de código a seguir mostra como pesquisar contratos por ID do contrato.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreements give id
 information
AG-02-A
"""


import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# To search by offer id: OfferId; by product id: ResourceIdentifier; by product
 type: ResourceType
idType = "ResourceType"

# replace id value as needed
idValue = "SaaSProduct"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)
```

```
def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the
 proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": idType, "values": [idValue]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": idType, "values": [idValue]},
                        {"name": "AgreementType", "values":
 ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise e

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])
```

```
    return AgreementSummaryList


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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a SearchAgreementsReferência da API AWS SDK for Python (Boto3).

Pesquise contratos por data de término usando um AWS SDK

Os exemplos de código a seguir mostram como pesquisar contratos por data de término.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class SearchAgreementsByEndDate {

 static String beforeOrAfterEndtimeFilterName =
 BeforeOrAfterEndTimeFilterName.BeforeEndTime.name();

 static String cutoffDate = "2050-11-18T00:00:00Z";

 static String partyTypeFilterValue = PARTY_TYPE_FILTER_VALUE_PROPOSER;

 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
 agreements where you are the proposer.
  // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
 are the acceptor.
```

```
 Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 Filter customizeFilter =
Filter.builder().name(beforeOrAfterEndtimeFilterName).values(cutoffDate).build();

 List<Filter> filters = new ArrayList<Filter>();

 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter));

 // search agreement with filters

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementResponse=
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());

 while (searchAgreementResponse.nextToken() != null &&
searchAgreementResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementResponse.nextToken())
    .build();
  searchAgreementResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}
```

```
}
```

- Para obter detalhes da API, consulte SearchAgreementsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreement
 information before or after end date
AG-03
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

# change to 'AfterEndTime' if after endtime is desired
```

```
beforeOrAfterEndtimeFilterName = "BeforeEndTime"

# Make sure to use the same date format as below
cutoffDate = "2322-11-18T00:00:00Z"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements():
    AgreementSummaryList = []

    try:
        agreement = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you
 are the proposer.
            # Change to "Acceptor" to return agreements where you are the
 acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": beforeOrAfterEndtimeFilterName, "values": [cutoffDate]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise

    AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    while "nextToken" in agreement:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreement["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {
                        "name": beforeOrAfterEndtimeFilterName,
                        "values": [cutoffDate],
```

```
                    },
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


if __name__ == "__main__":
    agreements = get_agreements()
    helper.pretty_print_datetime(agreements)
```

- Para obter detalhes da API, consulte a SearchAgreementsReferência da API AWS SDK for Python (Boto3).

Pesquise contratos por ID de oferta usando um AWS SDK

O exemplo de código a seguir mostra como pesquisar contratos por ID da oferta.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to search for agreements by offer
 id
AG-0
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# offer id to search by
offerId = "11111111111111111111111"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the
 proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": "OfferId", "values": [offerId]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise
```

```
        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": "OfferId", "values": [offerId]},
                        {"name": "AgreementType", "values":
 ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


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

    print("-" * 88)
    print("Looking for an agreement by offer id.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a SearchAgreementsReferência da API AWS SDK for Python (Boto3).

Pesquise contratos por ID do produto usando um AWS SDK

O exemplo de código a seguir mostra como pesquisar contratos por ID do produto.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreement by
 product id
AG-02
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# product id to search by
resourceId = "prod-1111111111111"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)
```

```
def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the
 proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": "ResourceIdentifier", "values": [resourceId]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": "ResourceIdentifier", "values": [resourceId]},
                        {"name": "AgreementType", "values":
 ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList
```

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

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a SearchAgreementsReferência da API AWS SDK for Python (Boto3).

Pesquise contratos por status usando um AWS SDK

O exemplo de código a seguir mostra como pesquisar contratos por status.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to filter agreements by status
```

```
AG-04

Example Usage: python3 search_agreements_by_status.py
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

# Set PartyType to "Proposer" to return agreements where you are the proposer.
# Change to "Acceptor" to return agreements where you are the acceptor.
party_type_list = ["Proposer"]
agreement_type_list = ["PurchaseAgreement"]

# Accepted values: "ACTIVE", "TERMINATED", "CANCELED", "EXPIRED", "REPLACED",
 "RENEWED"
status_list = ["ACTIVE"]

filter_list = [
    {"name": "PartyType", "values": party_type_list},
    {"name": "AgreementType", "values": agreement_type_list},
    {"name": "Status", "values": status_list},
]

agreement_results_list = []


def get_agreements(filter_list=filter_list):
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
```

```
            maxResults=MAX_PAGE_RESULTS,
            filters=filter_list,
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    agreement_results_list.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=filter_list,
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        agreement_results_list.extend(agreements["agreementViewSummaries"])

    helper.pretty_print_datetime(agreement_results_list)
    return agreement_results_list


if __name__ == "__main__":
    agreements_list = get_agreements(filter_list)
```

- Para obter detalhes da API, consulte a SearchAgreementsReferência da API AWS SDK for Python (Boto3).

Pesquise contratos com um filtro personalizado usando um AWS SDK

O exemplo de código a seguir mostra como pesquisar contratos com um filtro personalizado.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * All filter combinations we support for Proposer and Acceptor:
 * https://docs.aws.amazon.com/marketplace/latest/APIReference/API_marketplace-
agreements_SearchAgreements.html
 */

public class SearchAgreementsByOneFilter {

 private static final String FILTER_NAME = "ResourceType";
```

```
private static final String FILTER_VALUE = "SaaSProduct";

/*
 * search agreements by one customize filter
 */
public static void main(String[] args) {

 List<AgreementViewSummary> agreementSummaryList = getAgreements();

 ReferenceCodesUtils.formatOutput(agreementSummaryList);
}

public static List<AgreementViewSummary> getAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
agreements where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
are the acceptor.
 Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 List<Filter> filters = new ArrayList<Filter>();

 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter));

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();
```

```
 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Para obter detalhes da API, consulte SearchAgreementsa Referência AWS SDK for Java 2.x da API.

Pesquise contratos com dois filtros personalizados usando um AWS SDK

O exemplo de código a seguir mostra como pesquisar contratos com dois filtros personalizados.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * All filter combinations we support for Proposer and Acceptor:
 * https://docs.aws.amazon.com/marketplace/latest/APIReference/API_marketplace-
agreements_SearchAgreements.html
 */

public class SearchAgreementsByTwoFilters {
```

```
public static final String FILTER_1_NAME = "ResourceType";

public static final String FILTER_1_VALUE = "SaaSProduct";

public static final String FILTER_2_NAME = "Status";

public static final String FILTER_2_VALUE = "ACTIVE";

/*
 * search agreements by two customize filter
 */
public static void main(String[] args) {

 List<AgreementViewSummary> agreementSummaryList = getAgreements();

 ReferenceCodesUtils.formatOutput(agreementSummaryList);

}

public static List<AgreementViewSummary> getAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
agreements where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
are the acceptor.
 Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 Filter customizeFilter1 =
Filter.builder().name(FILTER_1_NAME).values(FILTER_1_VALUE).build();

 Filter customizeFilter2 =
Filter.builder().name(FILTER_2_NAME).values(FILTER_2_VALUE).build();


 List<Filter> filters = new ArrayList<Filter>();
```

```
 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter1, customizeFilter2));

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Para obter detalhes da API, consulte SearchAgreementsa Referência AWS SDK for Java 2.x da API.

Atualize os pedidos de compra após a aceitação do contrato usando um AWS SDK

Os exemplos de código a seguir mostram como atualizar os pedidos de compra após a aceitação do contrato.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Charge;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementChargesRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementChargesResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.PurchaseOrder;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import
 software.amazon.awssdk.services.marketplaceagreement.model.UpdatePurchaseOrdersRequest;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT pricing model
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing
 model and provides a purchase order
 * reference in AcceptAgreementRequest. After acceptance, the buyer lists the
 resulting
 * charges via ListAgreementCharges and associates the purchase order reference
 with a
 * specific charge via UpdatePurchaseOrders.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement.</li>
 *   <li>{@code DIMENSION_1_KEY} — dimension key defined in the offer.</li>
 *   <li>{@code PURCHASE_ORDER_REFERENCE} — your internal purchase order number
 (e.g., {@code po-123456}).</li>
 * </ul>
 */
public class UpdatePurchaseOrdersAfterAgreementAcceptance {

    // Your internal purchase order reference number (e.g., "po-123456").
    private static final String PURCHASE_ORDER_REFERENCE = "po-123456";

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
```

```
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P366D" for 366 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // Dimension key and quantity defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in your offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

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

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS agreement with CONTRACT pricing model with a purchase
 order reference.
     * 2. List charges to retrieve charge IDs and revisions.
     * 3. Associate the purchase order reference with a specific charge via
 UpdatePurchaseOrders.
     * 4. List charges again to confirm the update.
     */
    private static void listAgreementChargesAndUpdatePurchaseOrders() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

 .configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
```

```
.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build())
                                .build()))
                .build();

        RequestedTerm legalTerm =
RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();

        // --- Create Agreement ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        // --- Accept Agreement Request with Purchase Order ---
        // The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
        String chargeId =
createAgreementRequestResponse.chargeSummary().expectedCharges().get(0).id();
        PurchaseOrder purchaseOrderAtAcceptance = PurchaseOrder.builder()
                .chargeId(chargeId)
                .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                .build();
        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .purchaseOrders(purchaseOrderAtAcceptance)
                        .build();
```

```
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        final String agreementId = acceptAgreementRequestResponse.agreementId();
        System.out.println("Agreement request accepted with purchase order
reference '" + PURCHASE_ORDER_REFERENCE
                + "'. AgreementId: " + agreementId);

        // --- List Agreement Charges ---
        ListAgreementChargesRequest listAgreementChargesRequest =
                ListAgreementChargesRequest.builder()
                        .agreementId(agreementId)
                        .build();
        ListAgreementChargesResponse listAgreementChargesResponse =

marketplaceAgreementClient.listAgreementCharges(listAgreementChargesRequest);

        System.out.println("All charges for agreement " + agreementId + ":");
        AgreementApiUtils.formatOutput(listAgreementChargesResponse);

        // --- Update Purchase Order ---
        Charge firstCharge = listAgreementChargesResponse.items().get(0);
        PurchaseOrder purchaseOrder = PurchaseOrder.builder()
                .agreementId(agreementId)
                .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                .chargeRevision(firstCharge.revision())
                .chargeId(firstCharge.id())
                .build();
        UpdatePurchaseOrdersRequest updatePurchaseOrdersRequest =
                UpdatePurchaseOrdersRequest.builder()
                        .purchaseOrders(purchaseOrder)
                        .build();

marketplaceAgreementClient.updatePurchaseOrders(updatePurchaseOrdersRequest);
        System.out.println("Purchase order reference '" +
PURCHASE_ORDER_REFERENCE
                + "' updated for ChargeId: " + firstCharge.id());

        // --- Verify Update ---
        ListAgreementChargesRequest lacRequest =
                ListAgreementChargesRequest.builder()
                        .agreementId(agreementId)
                        .build();
        ListAgreementChargesResponse lacResponse =
```

```
                marketplaceAgreementClient.listAgreementCharges(lacRequest);
        System.out.println("Verified updated charge:");
        AgreementApiUtils.formatOutput(lacResponse.items().get(0));
    }
}
```

- Para obter detalhes da API, consulte UpdatePurchaseOrdersa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
    ListAgreementChargesCommand,
    UpdatePurchaseOrdersCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput } = require("./utils/
AgreementApiUtils");

/**
 * Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT pricing model
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model
 and provides a purchase order
 * reference in AcceptAgreementRequest. After acceptance, the buyer lists the
 resulting
```

```
 * charges via ListAgreementCharges and associates the purchase order reference
 with a
 * specific charge via UpdatePurchaseOrders.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement.
 *   - DIMENSION_1_KEY — dimension key defined in the offer.
 *   - PURCHASE_ORDER_REFERENCE — your internal purchase order number (e.g.,
 "po-123456").
 */

// Your internal purchase order reference number (e.g., "po-123456").
const PURCHASE_ORDER_REFERENCE = "po-123456";

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P366D" for 366 days).
const SELECTOR_VALUE = "<your-selector-value>";

// Dimension key and quantity defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-key>";
const DIMENSION_1_VALUE = 1;

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in your offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS agreement with CONTRACT pricing model with a purchase order
 reference.
 * 2. List charges to retrieve charge IDs and revisions.
```

```
 * 3. Associate the purchase order reference with a specific charge via
 UpdatePurchaseOrders.
 * 4. List charges again to confirm the update.
 */
async function listAgreementChargesAndUpdatePurchaseOrders() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Create Agreement ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
 validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    // --- Accept Agreement Request with Purchase Order ---
    // The chargeId is available from the CAR response's
 chargeSummary.expectedCharges.
    const chargeId =
 createAgreementRequestResponse.chargeSummary.expectedCharges[0].id;
    const purchaseOrderAtAcceptance = {
        chargeId: chargeId,
        purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
```

```
    };
    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
            purchaseOrders: [purchaseOrderAtAcceptance],
        })
    );
    const agreementId = acceptAgreementRequestResponse.agreementId;
    console.log("Agreement request accepted with purchase order reference '" +
PURCHASE_ORDER_REFERENCE
        + "'. AgreementId: " + agreementId);

    // --- List Agreement Charges ---
    const listAgreementChargesResponse = await client.send(
        new ListAgreementChargesCommand({
            agreementId: agreementId,
        })
    );

    console.log("All charges for agreement " + agreementId + ":");
    formatOutput(listAgreementChargesResponse);

    // --- Update Purchase Order ---
    const firstCharge = listAgreementChargesResponse.items[0];
    const purchaseOrder = {
        agreementId: agreementId,
        purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
        chargeRevision: firstCharge.revision,
        chargeId: firstCharge.id,
    };
    await client.send(
        new UpdatePurchaseOrdersCommand({
            purchaseOrders: [purchaseOrder],
        })
    );
    console.log("Purchase order reference '" + PURCHASE_ORDER_REFERENCE
        + "' updated for ChargeId: " + firstCharge.id);

    // --- Verify Update ---
    const lacResponse = await client.send(
        new ListAgreementChargesCommand({
            agreementId: agreementId,
        })
```

```
    );
    console.log("Verified updated charge:");
    formatOutput(lacResponse.items[0]);
}

listAgreementChargesAndUpdatePurchaseOrders();
```

- Para obter detalhes da API, consulte UpdatePurchaseOrdersa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT
pricing model using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model
 and provides
a purchase order reference in AcceptAgreementRequest. After acceptance, the buyer
 lists the
resulting charges via ListAgreementCharges and associates the purchase order
 reference with a
specific charge via UpdatePurchaseOrders.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement.
```

```
  - DIMENSION_1_KEY — dimension key defined in the offer.
  - PURCHASE_ORDER_REFERENCE — your internal purchase order number (e.g.,
 po-123456).
"""

import boto3

from utils.agreement_api_utils import format_output, generate_client_token


class UpdatePurchaseOrdersAfterAgreementAcceptance:

    # Your internal purchase order reference number (e.g., "po-123456").
    PURCHASE_ORDER_REFERENCE = "po-123456"

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P366D" for 366 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # Dimension key and quantity defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-key>"
    DIMENSION_1_VALUE = 1

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in your offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    @staticmethod
    def list_agreement_charges_and_update_purchase_orders():
        """
        Full end-to-end flow:
        1. Create a SaaS agreement with CONTRACT pricing model with a purchase
 order reference.
        2. List charges to retrieve charge IDs and revisions.
        3. Associate the purchase order reference with a specific charge via
 UpdatePurchaseOrders.
```

```
        4. List charges again to confirm the update.
        """
        client = boto3.client("marketplace-agreement")
        cls = UpdatePurchaseOrdersAfterAgreementAcceptance

        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Create Agreement ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)

        # --- Accept Agreement Request with Purchase Order ---
        # The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
        charge_id = create_response["chargeSummary"]["expectedCharges"][0]["id"]
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id,
            purchaseOrders=[
                {
                    "chargeId": charge_id,
```

```
                    "purchaseOrderReference": cls.PURCHASE_ORDER_REFERENCE,
                }
            ],
        )
        agreement_id = accept_response["agreementId"]
        print(
            "Agreement request accepted with purchase order reference '"
            + cls.PURCHASE_ORDER_REFERENCE + "'. AgreementId: " + agreement_id
        )

        # --- List Agreement Charges ---
        list_charges_response =
 client.list_agreement_charges(agreementId=agreement_id)
        print("All charges for agreement " + agreement_id + ":")
        format_output(list_charges_response)

        # --- Update Purchase Order ---
        first_charge = list_charges_response["items"][0]
        client.update_purchase_orders(
            purchaseOrders=[
                {
                    "agreementId": agreement_id,
                    "purchaseOrderReference": cls.PURCHASE_ORDER_REFERENCE,
                    "chargeRevision": first_charge["revision"],
                    "chargeId": first_charge["id"],
                }
            ],
        )
        print(
            "Purchase order reference '" + cls.PURCHASE_ORDER_REFERENCE
            + "' updated for ChargeId: " + first_charge["id"]
        )

        # --- Verify Update ---
        lac_response = client.list_agreement_charges(agreementId=agreement_id)
        print("Verified updated charge:")
        format_output(lac_response["items"][0])


if __name__ == "__main__":

 UpdatePurchaseOrdersAfterAgreementAcceptance.list_agreement_charges_and_update_purchase_
```

- Para obter detalhes da API, consulte a UpdatePurchaseOrdersReferência da API AWS SDK for Python (Boto3).

Exemplos de código para MediaConnect usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Elemental MediaConnect com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- MediaConnect Guia do desenvolvedor — Mais informações sobre MediaConnect.
- MediaConnect Referência da API — Detalhes sobre todas as MediaConnect ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para MediaConnect usar AWS SDKs
  - Introdução à transmissão de transporte de vídeo

Cenários para MediaConnect usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns MediaConnect com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas MediaConnect ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução à transmissão de transporte de vídeo

Introdução à transmissão de transporte de vídeo

O exemplo de código a seguir mostra como:

- Verifique o acesso ao Elemental MediaConnect
- Criar um fluxo.
- Adicionar uma saída.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Elemental MediaConnect Getting Started Tutorial Script
# This script creates a MediaConnect flow, adds an output, grants an entitlement,
# and then cleans up the resources.

set -euo pipefail

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="mediaconnect-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Elemental MediaConnect tutorial script at $(date)"
```

```
echo "All commands and outputs will be logged to $LOG_FILE"

# Tags for all resources
TAGS_ARRAY=("Key=project,Value=doc-smith" "Key=tutorial,Value=aws-elemental-
mediaconnect-gs")

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI is available
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        handle_error "AWS CLI is not installed or not in PATH"
    fi

    # Security: Verify AWS CLI version is recent
    local aws_version
    aws_version=$(aws --version 2>&1 | head -1)
    echo "AWS CLI version: $aws_version"

    if ! aws sts get-caller-identity &> /dev/null; then
        handle_error "AWS credentials are not configured or invalid"
    fi

    # Security: Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null)
    if [ -z "$account_id" ]; then
        handle_error "Failed to retrieve AWS account ID"
    fi
    echo "AWS Account ID: $account_id"
}

# Function to safely extract JSON values using jq (preferred) or fallback
extract_json_value() {
    local json_output="$1"
    local key="$2"
```

```
    if [ -z "$json_output" ]; then
        return 1
    fi

    # Security: Use jq if available for safer JSON parsing
    if command -v jq &> /dev/null; then
        echo "$json_output" | jq -r ".${key} // empty" 2>/dev/null || return 1
    else
        # Fallback with additional validation
        if ! echo "$json_output" | grep -q "\"$key\""; then
            return 1
        fi
        echo "$json_output" | grep -o "\"$key\": \"[^\"]*" | head -1 | cut -d'"'
 -f4
    fi
}

# Function to tag MediaConnect resource
tag_mediaconnect_resource() {
    local resource_arn="$1"
    echo "Tagging resource: $resource_arn"

    for tag in "${TAGS_ARRAY[@]}"; do
        if ! aws mediaconnect tag-resource --resource-arn "$resource_arn" --tags
 "$tag" 2>&1; then
            echo "WARNING: Failed to apply tag $tag to resource"
        fi
    done
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "${FLOW_ARN:-}" ]; then
        # Security: Validate ARN format before using it
        if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-
zA-Z0-9:-]+$ ]]; then
            echo "WARNING: Invalid Flow ARN format, skipping cleanup: $FLOW_ARN"
            return 1
        fi

        # Check flow status before attempting to stop
        echo "Checking flow status..."
```

```
        local flow_status_output
        if flow_status_output=$(aws mediaconnect describe-flow --flow-arn
 "$FLOW_ARN" --query "Flow.Status" --output text 2>&1); then
            echo "Current flow status: $flow_status_output"

            if [ "$flow_status_output" == "ACTIVE" ] || [ "$flow_status_output"
 == "UPDATING" ]; then
                echo "Stopping flow: $FLOW_ARN"
                if aws mediaconnect stop-flow --flow-arn "$FLOW_ARN" 2>&1; then
                    # Wait for flow to stop before deleting
                    echo "Waiting for flow to stop..."
                    sleep 10
                else
                    echo "WARNING: Failed to stop flow. Attempting to delete
 anyway..."
                fi
            else
                echo "Flow is not in ACTIVE or UPDATING state, skipping stop
 operation."
            fi

            # Delete the flow
            echo "Deleting flow: $FLOW_ARN"
            if aws mediaconnect delete-flow --flow-arn "$FLOW_ARN" 2>&1; then
                echo "Flow deleted successfully"
            else
                echo "WARNING: Failed to delete flow. You may need to manually
 delete it from the AWS console."
            fi
        else
            echo "WARNING: Could not check flow status"
        fi
    fi
}

# Set trap to cleanup on script exit
trap cleanup_resources EXIT

# Validate AWS CLI setup
validate_aws_cli

# Get the current AWS region
aws_region=""
if aws_region=$(aws configure get region 2>/dev/null); then
```

```
    if [ -z "$aws_region" ]; then
        handle_error "Failed to get AWS region. Please make sure AWS CLI is
 configured."
    fi
else
    handle_error "Failed to retrieve AWS region configuration"
fi

# Security: Validate region format
if [[ ! "$aws_region" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $aws_region"
fi

AWS_REGION="$aws_region"
echo "Using AWS Region: $AWS_REGION"

# Get available availability zones in the current region
echo "Getting available availability zones in region $AWS_REGION..."
az_output=""
if az_output=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --query
 "AvailabilityZones[0].ZoneName" --output text 2>&1); then
    AVAILABILITY_ZONE="$az_output"
    if [ -z "$AVAILABILITY_ZONE" ]; then
        handle_error "Failed to retrieve availability zones"
    fi

    # Security: Validate AZ format
    if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9][a-z]$ ]]; then
        handle_error "Invalid availability zone format: $AVAILABILITY_ZONE"
    fi

    echo "Using availability zone: $AVAILABILITY_ZONE"
else
    handle_error "Failed to get availability zones"
fi

# Generate a unique suffix for resource names
SUFFIX=$(date +%s | cut -c 6-10)
FLOW_NAME="AwardsNYCShow-${SUFFIX}"
SOURCE_NAME="AwardsNYCSource-${SUFFIX}"
OUTPUT_NAME="AwardsNYCOutput-${SUFFIX}"
ENTITLEMENT_NAME="PhillyTeam-${SUFFIX}"

echo "Using the following resource names:"
```

```
echo "Flow name: $FLOW_NAME"
echo "Source name: $SOURCE_NAME"
echo "Output name: $OUTPUT_NAME"
echo "Entitlement name: $ENTITLEMENT_NAME"

# Step 1: Verify access to MediaConnect
echo "Step 1: Verifying access to AWS Elemental MediaConnect..."
list_flows_output=""
if list_flows_output=$(aws mediaconnect list-flows 2>&1); then
    echo "$list_flows_output"
else
    handle_error "Failed to list flows. Please check your AWS credentials and
 permissions."
fi

# Step 2: Create a flow
echo "Step 2: Creating a flow..."
create_flow_output=""
if create_flow_output=$(aws mediaconnect create-flow \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "$FLOW_NAME" \
    --source "Name=$SOURCE_NAME,Protocol=zixi-
push,WhitelistCidr=10.24.34.0/23,StreamId=ZixiAwardsNYCFeed" 2>&1); then
    echo "$create_flow_output"
else
    handle_error "Failed to create flow"
fi

# Extract the flow ARN from the output
FLOW_ARN=$(echo "$create_flow_output" | jq -r '.Flow.FlowArn // empty' 2>/dev/
null)
if [ -z "$FLOW_ARN" ]; then
    FLOW_ARN=$(echo "$create_flow_output" | grep -o '"FlowArn": "[^"]*' | head -1
 | cut -d'"' -f4)
fi
if [ -z "$FLOW_ARN" ]; then
    handle_error "Failed to extract flow ARN from output"
fi
echo "Flow ARN: $FLOW_ARN"

# Validate flow ARN format
if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-
Z0-9:-]+$ ]]; then
    handle_error "Invalid Flow ARN format: $FLOW_ARN"
```

```
fi

# Tag the flow
tag_mediaconnect_resource "$FLOW_ARN"

# Step 3: Add an output
echo "Step 3: Adding an output to the flow..."
add_output_output=""
if add_output_output=$(aws mediaconnect add-flow-outputs \
    --flow-arn "$FLOW_ARN" \
    --outputs "Name=$OUTPUT_NAME,Protocol=zixi-
push,Destination=198.51.100.11,Port=1024,StreamId=ZixiAwardsOutput" 2>&1); then
    echo "$add_output_output"
else
    handle_error "Failed to add output to flow"
fi

# Extract the output ARN
output_arn=""
output_arn=$(echo "$add_output_output" | jq -r ".Output.OutputArn // empty" 2>/
dev/null)
if [ -z "$output_arn" ]; then output_arn=$(echo "$add_output_output" | grep -o
 '"OutputArn": "[^"]*' | head -1 | cut -d'"' -f4); fi
if [ -z "$output_arn" ]; then
    echo "WARNING: Failed to extract output ARN from output"
else
    OUTPUT_ARN="$output_arn"
    echo "Output ARN: $OUTPUT_ARN"
    tag_mediaconnect_resource "$OUTPUT_ARN"
fi

# Step 4: Grant an entitlement
echo "Step 4: Granting an entitlement..."
grant_entitlement_output=""
if grant_entitlement_output=$(aws mediaconnect grant-flow-entitlements \
    --flow-arn "$FLOW_ARN" \
    --entitlements "Name=$ENTITLEMENT_NAME,Subscribers=222233334444" 2>&1); then
    echo "$grant_entitlement_output"
else
    handle_error "Failed to grant entitlement"
fi

# Extract the entitlement ARN
entitlement_arn=""
```

```
entitlement_arn=$(echo "$grant_entitlement_output" | jq -r
 '.Entitlement.EntitlementArn // empty' 2>/dev/null)
if [ -z "$entitlement_arn" ]; then
    entitlement_arn=$(echo "$grant_entitlement_output" | grep -o
 '"EntitlementArn": "[^"]*' | head -1 | cut -d'"' -f4)
fi
if [ -z "$entitlement_arn" ]; then
    echo "WARNING: Failed to extract entitlement ARN from output"
else
    ENTITLEMENT_ARN="$entitlement_arn"
    echo "Entitlement ARN: $ENTITLEMENT_ARN"
    tag_mediaconnect_resource "$ENTITLEMENT_ARN"
fi

# Step 5: List entitlements to share with affiliates
echo "Step 5: Listing entitlements for the flow..."
describe_flow_output=""
if describe_flow_output=$(aws mediaconnect describe-flow --flow-arn "$FLOW_ARN"
 --query "Flow.Entitlements" 2>&1); then
    echo "Entitlements for the flow:"
    echo "$describe_flow_output"
else
    handle_error "Failed to describe flow"
fi

# Display information to share with affiliates
echo ""
echo "Information to share with your Philadelphia affiliate:"
echo "Entitlement ARN: ${ENTITLEMENT_ARN:-N/A}"
echo "AWS Region: $AWS_REGION"

# Display resource summary
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "The following resources were created:"
echo "1. Flow: $FLOW_NAME (ARN: $FLOW_ARN)"
echo "2. Output: $OUTPUT_NAME (ARN: ${OUTPUT_ARN:-N/A})"
echo "3. Entitlement: $ENTITLEMENT_NAME (ARN: ${ENTITLEMENT_ARN:-N/A})"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Automatically cleaning up all created resources..."

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddFlowOutputs
  - CreateFlow
  - DeleteFlow
  - DescribeAvailabilityZones
  - DescribeFlow
  - GrantFlowEntitlements
  - ListFlows
  - StopFlow

Exemplos de código para MediaConvert usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Elemental MediaConvert com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- MediaConvert Guia do usuário — Mais informações sobre MediaConvert.
- MediaConvert Referência da API — Detalhes sobre todas as MediaConvert ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de MediaConvert uso AWS SDKs
  - Olá MediaConvert

- Ações para MediaConvert usar AWS SDKs
  - Use CreateJobcom um AWS SDK ou CLI
  - Use GetJobcom um AWS SDK ou CLI
  - Use ListJobscom um AWS SDK ou CLI

Exemplos básicos de MediaConvert uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Elemental MediaConvert com AWS SDKs.

Exemplos

- Olá MediaConvert
- Ações para MediaConvert usar AWS SDKs
  - Use CreateJobcom um AWS SDK ou CLI
  - Use GetJobcom um AWS SDK ou CLI
  - Use ListJobscom um AWS SDK ou CLI

Olá MediaConvert

O exemplo de código a seguir mostra como começar a usar o AWS Elemental MediaConvert.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.MediaConvert;
using Amazon.MediaConvert.Model;

namespace MediaConvertActions;

public static class HelloMediaConvert
```

```
{
    static async Task Main(string[] args)
    {
        // Create the client using the default profile.
        var mediaConvertClient = new AmazonMediaConvertClient();

        Console.WriteLine($"Hello AWS Elemental MediaConvert! Your MediaConvert
 Jobs are:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get some MediaConvert jobs.
        var response = await mediaConvertClient.ListJobsAsync(
            new ListJobsRequest()
            {
                MaxResults = 10
            }
            );

        foreach (var job in response.Jobs)
        {
            Console.WriteLine($"\tJob: {job.Id} status {job.Status}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeEndpointsa Referência AWS SDK para .NET da API.

Ações para MediaConvert usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar MediaConvert ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Elemental MediaConvert.

Exemplos

- Use CreateJobcom um AWS SDK ou CLI

- Use GetJobcom um AWS SDK ou CLI
- Use ListJobscom um AWS SDK ou CLI

Use **`CreateJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateJob`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o wrapper, o cliente e os locais do arquivo.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-
role-in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Creating job for input file {fileInput}.");
        var jobId = await wrapper.CreateJob(mediaConvertRole!, fileInput!,
 fileOutput!);
        Console.WriteLine($"Created job with Job ID: {jobId}");
```

```
        Console.WriteLine(new string('-', 80));
```

Criar o trabalho usando o método wrapper e retorne o ID do trabalho.

```
    /// <summary>
    /// Create a job to convert a media file.
    /// </summary>
    /// <param name="mediaConvertRole">The Amazon Resource Name (ARN) of the
 media convert role, as specified here:
    /// https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-
in-mediaconvert-configured.html</param>
    /// <param name="fileInput">The Amazon Simple Storage Service (Amazon S3)
 location of the input media file.</param>
    /// <param name="fileOutput">The Amazon S3 location for the output media
 file.</param>
    /// <returns>The ID of the new job.</returns>
    public async Task<string> CreateJob(string mediaConvertRole, string
 fileInput,
        string fileOutput)
    {
        CreateJobRequest createJobRequest = new CreateJobRequest
        {
            Role = mediaConvertRole
        };

        createJobRequest.UserMetadata.Add("Customer", "Amazon");

        JobSettings jobSettings = new JobSettings
        {
            AdAvailOffset = 0,
            TimecodeConfig = new TimecodeConfig
            {
                Source = TimecodeSource.EMBEDDED
            }
        };
        createJobRequest.Settings = jobSettings;

        #region OutputGroup

        OutputGroup ofg = new OutputGroup
        {
```

```
            Name = "File Group",
            OutputGroupSettings = new OutputGroupSettings
            {
                Type = OutputGroupType.FILE_GROUP_SETTINGS,
                FileGroupSettings = new FileGroupSettings
                {
                    Destination = fileOutput
                }
            }
        };

        Output output = new Output
        {
            NameModifier = "_1"
        };

        #region VideoDescription

        VideoDescription vdes = new VideoDescription
        {
            ScalingBehavior = ScalingBehavior.DEFAULT,
            TimecodeInsertion = VideoTimecodeInsertion.DISABLED,
            AntiAlias = AntiAlias.ENABLED,
            Sharpness = 50,
            AfdSignaling = AfdSignaling.NONE,
            DropFrameTimecode = DropFrameTimecode.ENABLED,
            RespondToAfd = RespondToAfd.NONE,
            ColorMetadata = ColorMetadata.INSERT,
            CodecSettings = new VideoCodecSettings
            {
                Codec = VideoCodec.H_264
            }
        };
        output.VideoDescription = vdes;

        H264Settings h264 = new H264Settings
        {
            InterlaceMode = H264InterlaceMode.PROGRESSIVE,
            NumberReferenceFrames = 3,
            Syntax = H264Syntax.DEFAULT,
            Softness = 0,
            GopClosedCadence = 1,
            GopSize = 90,
            Slices = 1,
```

```
            GopBReference = H264GopBReference.DISABLED,
            SlowPal = H264SlowPal.DISABLED,
            SpatialAdaptiveQuantization =
H264SpatialAdaptiveQuantization.ENABLED,
            TemporalAdaptiveQuantization =
H264TemporalAdaptiveQuantization.ENABLED,
            FlickerAdaptiveQuantization =
H264FlickerAdaptiveQuantization.DISABLED,
            EntropyEncoding = H264EntropyEncoding.CABAC,
            Bitrate = 5000000,
            FramerateControl = H264FramerateControl.SPECIFIED,
            RateControlMode = H264RateControlMode.CBR,
            CodecProfile = H264CodecProfile.MAIN,
            Telecine = H264Telecine.NONE,
            MinIInterval = 0,
            AdaptiveQuantization = H264AdaptiveQuantization.HIGH,
            CodecLevel = H264CodecLevel.AUTO,
            FieldEncoding = H264FieldEncoding.PAFF,
            SceneChangeDetect = H264SceneChangeDetect.ENABLED,
            QualityTuningLevel = H264QualityTuningLevel.SINGLE_PASS,
            FramerateConversionAlgorithm =
                H264FramerateConversionAlgorithm.DUPLICATE_DROP,
            UnregisteredSeiTimecode = H264UnregisteredSeiTimecode.DISABLED,
            GopSizeUnits = H264GopSizeUnits.FRAMES,
            ParControl = H264ParControl.SPECIFIED,
            NumberBFramesBetweenReferenceFrames = 2,
            RepeatPps = H264RepeatPps.DISABLED,
            FramerateNumerator = 30,
            FramerateDenominator = 1,
            ParNumerator = 1,
            ParDenominator = 1
        };
        output.VideoDescription.CodecSettings.H264Settings = h264;

        #endregion VideoDescription

        #region AudioDescription

        AudioDescription ades = new AudioDescription
        {
            LanguageCodeControl = AudioLanguageCodeControl.FOLLOW_INPUT,
            // This name matches one specified in the following Inputs.
            AudioSourceName = "Audio Selector 1",
            CodecSettings = new AudioCodecSettings
```

```
            {
                Codec = AudioCodec.AAC
            }
        };

        AacSettings aac = new AacSettings
        {
            AudioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.NORMAL,
            RateControlMode = AacRateControlMode.CBR,
            CodecProfile = AacCodecProfile.LC,
            CodingMode = AacCodingMode.CODING_MODE_2_0,
            RawFormat = AacRawFormat.NONE,
            SampleRate = 48000,
            Specification = AacSpecification.MPEG4,
            Bitrate = 64000
        };
        ades.CodecSettings.AacSettings = aac;
        output.AudioDescriptions.Add(ades);

        #endregion AudioDescription

        #region Mp4 Container

        output.ContainerSettings = new ContainerSettings
        {
            Container = ContainerType.MP4
        };
        Mp4Settings mp4 = new Mp4Settings
        {
            CslgAtom = Mp4CslgAtom.INCLUDE,
            FreeSpaceBox = Mp4FreeSpaceBox.EXCLUDE,
            MoovPlacement = Mp4MoovPlacement.PROGRESSIVE_DOWNLOAD
        };
        output.ContainerSettings.Mp4Settings = mp4;

        #endregion Mp4 Container

        ofg.Outputs.Add(output);
        createJobRequest.Settings.OutputGroups.Add(ofg);

        #endregion OutputGroup

        #region Input
```

```
        Input input = new Input
        {
            FilterEnable = InputFilterEnable.AUTO,
            PsiControl = InputPsiControl.USE_PSI,
            FilterStrength = 0,
            DeblockFilter = InputDeblockFilter.DISABLED,
            DenoiseFilter = InputDenoiseFilter.DISABLED,
            TimecodeSource = InputTimecodeSource.EMBEDDED,
            FileInput = fileInput
        };

        AudioSelector audsel = new AudioSelector
        {
            Offset = 0,
            DefaultSelection = AudioDefaultSelection.NOT_DEFAULT,
            ProgramSelection = 1,
            SelectorType = AudioSelectorType.TRACK
        };
        audsel.Tracks.Add(1);
        input.AudioSelectors.Add("Audio Selector 1", audsel);

        input.VideoSelector = new VideoSelector
        {
            ColorSpace = ColorSpace.FOLLOW
        };

        createJobRequest.Settings.Inputs.Add(input);

        #endregion Input

        CreateJobResponse createJobResponse =
            await _amazonMediaConvert.CreateJobAsync(createJobRequest);

        var jobId = createJobResponse.Job.Id;

        return jobId;
    }
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an AWS Elemental MediaConvert job.
/*!
  \param mediaConvertRole: An Amazon Resource Name (ARN) for the AWS Identity and
                           Access Management (IAM) role for the job.
  \param fileInput: A URI to an input file that is stored in Amazon Simple
 Storage Service
                    (Amazon S3) or on an HTTP(S) server.
  \param fileOutput: A URI for an Amazon S3 output location and the output file
 name base.
  \param jobSettingsFile: An optional JSON settings file.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::MediaConvert::createJob(const Aws::String &mediaConvertRole,
                                     const Aws::String &fileInput,
                                     const Aws::String &fileOutput,
                                     const Aws::String &jobSettingsFile,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::MediaConvert::Model::CreateJobRequest createJobRequest;

    createJobRequest.SetRole(mediaConvertRole);
    Aws::Http::HeaderValueCollection hvc;
    hvc.emplace("Customer", "Amazon");
    createJobRequest.SetUserMetadata(hvc);

    if (!jobSettingsFile.empty()) // Use a JSON file for the job settings.
    {
        std::ifstream jobSettingsStream(jobSettingsFile, std::ios::ate);
        if (!jobSettingsStream) {
            std::cerr << "Unable to open the job template file." << std::endl;
```

```
            return false;
        }
        std::vector<char> buffer(jobSettingsStream.tellg());
        jobSettingsStream.seekg(0);
        jobSettingsStream.read(buffer.data(), buffer.size());
        std::string jobSettingsJSON(buffer.data(), buffer.size());
        size_t pos = jobSettingsJSON.find(INPUT_FILE_PLACEHOLDER);
        if (pos != std::string::npos) {
            jobSettingsJSON.replace(pos, strlen(INPUT_FILE_PLACEHOLDER),
fileInput);
        }

        pos = jobSettingsJSON.find(OUTPUT_FILE_PLACEHOLDER);
        if (pos != std::string::npos) {
            jobSettingsJSON.replace(pos, strlen(OUTPUT_FILE_PLACEHOLDER),
fileOutput);
        }
        Aws::Utils::Json::JsonValue jsonValue(jobSettingsJSON);
        Aws::MediaConvert::Model::JobSettings jobSettings(jsonValue);

        createJobRequest.SetSettings(jobSettings);
    }
    else { // Configure the job settings programmatically.
        Aws::MediaConvert::Model::JobSettings jobSettings;
        jobSettings.SetAdAvailOffset(0);
        Aws::MediaConvert::Model::TimecodeConfig timecodeConfig;

timecodeConfig.SetSource(Aws::MediaConvert::Model::TimecodeSource::EMBEDDED);
        jobSettings.SetTimecodeConfig(timecodeConfig);

        // Configure the output group.
        Aws::MediaConvert::Model::OutputGroup outputGroup;
        outputGroup.SetName("File Group");
        Aws::MediaConvert::Model::OutputGroupSettings outputGroupSettings;
        outputGroupSettings.SetType(
                Aws::MediaConvert::Model::OutputGroupType::FILE_GROUP_SETTINGS);
        Aws::MediaConvert::Model::FileGroupSettings fileGroupSettings;
        fileGroupSettings.SetDestination(fileOutput);
        outputGroupSettings.SetFileGroupSettings(fileGroupSettings);
        outputGroup.SetOutputGroupSettings(outputGroupSettings);

        Aws::MediaConvert::Model::Output output;
        output.SetNameModifier("_1");
```

```
        Aws::MediaConvert::Model::VideoDescription videoDescription;
        videoDescription.SetScalingBehavior(
                Aws::MediaConvert::Model::ScalingBehavior::DEFAULT);
        videoDescription.SetTimecodeInsertion(
                Aws::MediaConvert::Model::VideoTimecodeInsertion::DISABLED);

videoDescription.SetAntiAlias(Aws::MediaConvert::Model::AntiAlias::ENABLED);
        videoDescription.SetSharpness(50);

videoDescription.SetAfdSignaling(Aws::MediaConvert::Model::AfdSignaling::NONE);
        videoDescription.SetDropFrameTimecode(
                Aws::MediaConvert::Model::DropFrameTimecode::ENABLED);

videoDescription.SetRespondToAfd(Aws::MediaConvert::Model::RespondToAfd::NONE);
        videoDescription.SetColorMetadata(
                Aws::MediaConvert::Model::ColorMetadata::INSERT);

        Aws::MediaConvert::Model::VideoCodecSettings videoCodecSettings;
        videoCodecSettings.SetCodec(Aws::MediaConvert::Model::VideoCodec::H_264);
        Aws::MediaConvert::Model::H264Settings h264Settings;
        h264Settings.SetNumberReferenceFrames(3);
        h264Settings.SetSyntax(Aws::MediaConvert::Model::H264Syntax::DEFAULT);
        h264Settings.SetSoftness(0);
        h264Settings.SetGopClosedCadence(1);
        h264Settings.SetGopSize(90);
        h264Settings.SetSlices(1);
        h264Settings.SetGopBReference(
                Aws::MediaConvert::Model::H264GopBReference::DISABLED);
        h264Settings.SetSlowPal(Aws::MediaConvert::Model::H264SlowPal::DISABLED);
        h264Settings.SetSpatialAdaptiveQuantization(

Aws::MediaConvert::Model::H264SpatialAdaptiveQuantization::ENABLED);
        h264Settings.SetTemporalAdaptiveQuantization(

Aws::MediaConvert::Model::H264TemporalAdaptiveQuantization::ENABLED);
        h264Settings.SetFlickerAdaptiveQuantization(

Aws::MediaConvert::Model::H264FlickerAdaptiveQuantization::DISABLED);
        h264Settings.SetEntropyEncoding(
                Aws::MediaConvert::Model::H264EntropyEncoding::CABAC);
        h264Settings.SetBitrate(5000000);
        h264Settings.SetFramerateControl(
                Aws::MediaConvert::Model::H264FramerateControl::SPECIFIED);
        h264Settings.SetRateControlMode(
```

```
                Aws::MediaConvert::Model::H264RateControlMode::CBR);

h264Settings.SetCodecProfile(Aws::MediaConvert::Model::H264CodecProfile::MAIN);
        h264Settings.SetTelecine(Aws::MediaConvert::Model::H264Telecine::NONE);
        h264Settings.SetMinIInterval(0);
        h264Settings.SetAdaptiveQuantization(
                Aws::MediaConvert::Model::H264AdaptiveQuantization::HIGH);

h264Settings.SetCodecLevel(Aws::MediaConvert::Model::H264CodecLevel::AUTO);
        h264Settings.SetFieldEncoding(
                Aws::MediaConvert::Model::H264FieldEncoding::PAFF);
        h264Settings.SetSceneChangeDetect(
                Aws::MediaConvert::Model::H264SceneChangeDetect::ENABLED);
        h264Settings.SetQualityTuningLevel(
                Aws::MediaConvert::Model::H264QualityTuningLevel::SINGLE_PASS);
        h264Settings.SetFramerateConversionAlgorithm(

Aws::MediaConvert::Model::H264FramerateConversionAlgorithm::DUPLICATE_DROP);
        h264Settings.SetUnregisteredSeiTimecode(
                Aws::MediaConvert::Model::H264UnregisteredSeiTimecode::DISABLED);
        h264Settings.SetGopSizeUnits(
                Aws::MediaConvert::Model::H264GopSizeUnits::FRAMES);

h264Settings.SetParControl(Aws::MediaConvert::Model::H264ParControl::SPECIFIED);
        h264Settings.SetNumberBFramesBetweenReferenceFrames(2);

h264Settings.SetRepeatPps(Aws::MediaConvert::Model::H264RepeatPps::DISABLED);
        h264Settings.SetFramerateNumerator(30);
        h264Settings.SetFramerateDenominator(1);
        h264Settings.SetParNumerator(1);
        h264Settings.SetParDenominator(1);
        videoCodecSettings.SetH264Settings(h264Settings);
        videoDescription.SetCodecSettings(videoCodecSettings);
        output.SetVideoDescription(videoDescription);

        Aws::MediaConvert::Model::AudioDescription audioDescription;
        audioDescription.SetLanguageCodeControl(

Aws::MediaConvert::Model::AudioLanguageCodeControl::FOLLOW_INPUT);
        audioDescription.SetAudioSourceName(AUDIO_SOURCE_NAME);
        Aws::MediaConvert::Model::AudioCodecSettings audioCodecSettings;
        audioCodecSettings.SetCodec(Aws::MediaConvert::Model::AudioCodec::AAC);
        Aws::MediaConvert::Model::AacSettings aacSettings;
        aacSettings.SetAudioDescriptionBroadcasterMix(
```

```
Aws::MediaConvert::Model::AacAudioDescriptionBroadcasterMix::NORMAL);
       aacSettings.SetRateControlMode(
               Aws::MediaConvert::Model::AacRateControlMode::CBR);

aacSettings.SetCodecProfile(Aws::MediaConvert::Model::AacCodecProfile::LC);
       aacSettings.SetCodingMode(
               Aws::MediaConvert::Model::AacCodingMode::CODING_MODE_2_0);
       aacSettings.SetRawFormat(Aws::MediaConvert::Model::AacRawFormat::NONE);
       aacSettings.SetSampleRate(48000);

aacSettings.SetSpecification(Aws::MediaConvert::Model::AacSpecification::MPEG4);
       aacSettings.SetBitrate(64000);
       audioCodecSettings.SetAacSettings(aacSettings);
       audioDescription.SetCodecSettings(audioCodecSettings);
       Aws::Vector<Aws::MediaConvert::Model::AudioDescription>
audioDescriptions;
       audioDescriptions.emplace_back(audioDescription);
       output.SetAudioDescriptions(audioDescriptions);

       Aws::MediaConvert::Model::ContainerSettings mp4container;
       mp4container.SetContainer(Aws::MediaConvert::Model::ContainerType::MP4);
       Aws::MediaConvert::Model::Mp4Settings mp4Settings;
       mp4Settings.SetCslgAtom(Aws::MediaConvert::Model::Mp4CslgAtom::INCLUDE);

mp4Settings.SetFreeSpaceBox(Aws::MediaConvert::Model::Mp4FreeSpaceBox::EXCLUDE);
       mp4Settings.SetMoovPlacement(

Aws::MediaConvert::Model::Mp4MoovPlacement::PROGRESSIVE_DOWNLOAD);
       mp4container.SetMp4Settings(mp4Settings);
       output.SetContainerSettings(mp4container);

       outputGroup.AddOutputs(output);
       jobSettings.AddOutputGroups(outputGroup);

       // Configure inputs.
       Aws::MediaConvert::Model::Input input;
       input.SetFilterEnable(Aws::MediaConvert::Model::InputFilterEnable::AUTO);
       input.SetPsiControl(Aws::MediaConvert::Model::InputPsiControl::USE_PSI);
       input.SetFilterStrength(0);

input.SetDeblockFilter(Aws::MediaConvert::Model::InputDeblockFilter::DISABLED);

input.SetDenoiseFilter(Aws::MediaConvert::Model::InputDenoiseFilter::DISABLED);
```

```
        input.SetTimecodeSource(
                Aws::MediaConvert::Model::InputTimecodeSource::EMBEDDED);
        input.SetFileInput(fileInput);

        Aws::MediaConvert::Model::AudioSelector audioSelector;
        audioSelector.SetOffset(0);
        audioSelector.SetDefaultSelection(
                Aws::MediaConvert::Model::AudioDefaultSelection::NOT_DEFAULT);
        audioSelector.SetProgramSelection(1);
        audioSelector.SetSelectorType(
                Aws::MediaConvert::Model::AudioSelectorType::TRACK);
        audioSelector.AddTracks(1);
        input.AddAudioSelectors(AUDIO_SOURCE_NAME, audioSelector);

        Aws::MediaConvert::Model::VideoSelector videoSelector;

 videoSelector.SetColorSpace(Aws::MediaConvert::Model::ColorSpace::FOLLOW);
        input.SetVideoSelector(videoSelector);

        jobSettings.AddInputs(input);

        createJobRequest.SetSettings(jobSettings);
    }

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);
    Aws::MediaConvert::Model::CreateJobOutcome outcome = client.CreateJob(
            createJobRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Job successfully created with ID - "
                  << outcome.GetResult().GetJob().GetId() << std::endl;
    }
    else {
        std::cerr << "Error CreateJob - " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como criar um trabalho

O exemplo de `create-job` a seguir cria um trabalho de transcodificação com as configurações especificadas em um arquivo `job.json` presente no sistema que você usou para enviar o comando. Essa especificação de trabalho JSON pode especificar cada configuração individualmente, fazer referência a um modelo de trabalho ou a predefinições de saída.

```
aws mediaconvert create-job \
    --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \
    --region region-name-1 \
    --cli-input-json file://~/job.json
```

Você pode usar o MediaConvert console AWS Elemental para gerar a especificação de trabalho JSON escolhendo suas configurações de trabalho e, em seguida, escolhendo Mostrar trabalho JSON na parte inferior da seção Trabalho.

Use `describe-endpoints` para obter o endpoint específico da sua conta ou envie o comando sem o endpoint. O serviço retorna um erro e o endpoint.

Se a solicitação for bem-sucedida, o serviço retornará a especificação do trabalho JSON enviado com a solicitação.

Para obter mais informações, consulte Trabalhando com MediaConvert tarefas AWS elementares no Guia do MediaConvert usuário do AWS Elemental.

- Para obter detalhes da API, consulte CreateJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import
 software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import
 software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import
 software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
```

```
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
 software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
 software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
 software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
```

```
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import
 software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */
public class CreateJob {
        public static void main(String[] args) {
                final String usage = """

                                    Usage:
```

```
                                    <mcRoleARN> <fileInput>\s

                                Where:
                                    mcRoleARN - The MediaConvert Role ARN.\s
                                    fileInput -  The URL of an Amazon S3 bucket
where the input file is located.\s
                                """;

               if (args.length != 2) {
                       System.out.println(usage);
                       System.exit(1);
               }

               String mcRoleARN = args[0];
               String fileInput = args[1];
               Region region = Region.US_WEST_2;
               MediaConvertClient mc = MediaConvertClient.builder()
                               .region(region)
                               .build();

               String id = createMediaJob(mc, mcRoleARN, fileInput);
               System.out.println("MediaConvert job created. Job Id = " + id);
               mc.close();
       }

       public static String createMediaJob(MediaConvertClient mc, String
mcRoleARN, String fileInput) {

               String s3path = fileInput.substring(0, fileInput.lastIndexOf('/')
+ 1) + "javasdk/out/";
               String fileOutput = s3path + "index";
               String thumbsOutput = s3path + "thumbs/";
               String mp4Output = s3path + "mp4/";

               try {
                       System.out.println("MediaConvert role arn: " +
mcRoleARN);
                       System.out.println("MediaConvert input file: " +
fileInput);
                       System.out.println("MediaConvert output path: " +
s3path);

                       // output group Preset HLS low profile
```

```
                        Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
 750000, 7, 1920, 1080, 640);
                        // output group Preset HLS media profile
                        Output hlsMedium = createOutput("hls_medium", "_medium",
 "_$dt$", 1200000, 7, 1920, 1080, 1280);
                        // output group Preset HLS high profole
                        Output hlsHigh = createOutput("hls_high", "_high", "_$dt
$", 3500000, 8, 1920, 1080, 1920);

                        OutputGroup appleHLS = OutputGroup.builder().name("Apple
 HLS").customName("Example")

 .outputGroupSettings(OutputGroupSettings.builder()

 .type(OutputGroupType.HLS_GROUP_SETTINGS)

 .hlsGroupSettings(HlsGroupSettings.builder()

 .directoryStructure(

       HlsDirectoryStructure.SINGLE_DIRECTORY)

 .manifestDurationFormat(

       HlsManifestDurationFormat.INTEGER)

 .streamInfResolution(

       HlsStreamInfResolution.INCLUDE)

 .clientCache(HlsClientCache.ENABLED)

 .captionLanguageSetting(

       HlsCaptionLanguageSetting.OMIT)

 .manifestCompression(

       HlsManifestCompression.NONE)

 .codecSpecification(

       HlsCodecSpecification.RFC_4281)
```

```
.outputSelection(

      HlsOutputSelection.MANIFESTS_AND_SEGMENTS)

.programDateTime(HlsProgramDateTime.EXCLUDE)

.programDateTimePeriod(600)

.timedMetadataId3Frame(

      HlsTimedMetadataId3Frame.PRIV)

.timedMetadataId3Period(10)

.destination(fileOutput)

.segmentControl(HlsSegmentControl.SEGMENTED_FILES)

.minFinalSegmentLength((double) 0)

.segmentLength(4).minSegmentLength(0).build())
                                                        .build())
                                          .outputs(hlsLow, hlsMedium,
hlsHigh).build();

                       OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(mp4Output).build())
                                                        .build())

.outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.MP4).build())
```

```
.videoDescription(VideoDescription.builder().width(1280)
.height(720)
.scalingBehavior(ScalingBehavior.DEFAULT)
.sharpness(50).antiAlias(AntiAlias.ENABLED)
.timecodeInsertion(
      VideoTimecodeInsertion.DISABLED)
.colorMetadata(ColorMetadata.INSERT)
.respondToAfd(RespondToAfd.NONE)
.afdSignaling(AfdSignaling.NONE)
.dropFrameTimecode(DropFrameTimecode.ENABLED)
.codecSettings(VideoCodecSettings.builder()
      .codec(VideoCodec.H_264)
      .h264Settings(H264Settings
                      .builder()
                      .rateControlMode(
                                      H264RateControlMode.QVBR)
                      .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                      .qualityTuningLevel(
                                      H264QualityTuningLevel.SINGLE_PASS)
                      .qvbrSettings(
                                      H264QvbrSettings.builder()
                                                      .qvbrQualityLevel(
```

```
                                                                    8)
                                                        .build())
                        .codecLevel(H264CodecLevel.AUTO)
                        .codecProfile(H264CodecProfile.MAIN)
                        .maxBitrate(2400000)
                        .framerateControl(

H264FramerateControl.INITIALIZE_FROM_SOURCE)
                        .gopSize(2.0)
                        .gopSizeUnits(H264GopSizeUnits.SECONDS)
                        .numberBFramesBetweenReferenceFrames(
                                        2)
                        .gopClosedCadence(
                                        1)
                        .gopBReference(H264GopBReference.DISABLED)
                        .slowPal(H264SlowPal.DISABLED)
                        .syntax(H264Syntax.DEFAULT)
                        .numberReferenceFrames(
                                        3)
                        .dynamicSubGop(H264DynamicSubGop.STATIC)
                        .fieldEncoding(H264FieldEncoding.PAFF)
                        .sceneChangeDetect(
```

```
                                        H264SceneChangeDetect.ENABLED)
                        .minIInterval(0)
                        .telecine(H264Telecine.NONE)
                        .framerateConversionAlgorithm(

H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                        .entropyEncoding(
                                        H264EntropyEncoding.CABAC)
                        .slices(1)
                        .unregisteredSeiTimecode(
                                        H264UnregisteredSeiTimecode.DISABLED)
                        .repeatPps(H264RepeatPps.DISABLED)
                        .adaptiveQuantization(
                                        H264AdaptiveQuantization.HIGH)
                        .spatialAdaptiveQuantization(
                                        H264SpatialAdaptiveQuantization.ENABLED)
                        .temporalAdaptiveQuantization(
                                        H264TemporalAdaptiveQuantization.ENABLED)
                        .flickerAdaptiveQuantization(
                                        H264FlickerAdaptiveQuantization.DISABLED)
                        .softness(0)
                        .interlaceMode(H264InterlaceMode.PROGRESSIVE)
```

```
                        .build())

      .build())
                                                                                .build())

.audioDescriptions(AudioDescription.builder()

.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

.languageCodeControl(

      AudioLanguageCodeControl.FOLLOW_INPUT)

.codecSettings(AudioCodecSettings.builder()

      .codec(AudioCodec.AAC)

      .aacSettings(AacSettings

                        .builder()

                        .codecProfile(AacCodecProfile.LC)

                        .rateControlMode(

                                        AacRateControlMode.CBR)

                        .codingMode(AacCodingMode.CODING_MODE_2_0)

                        .sampleRate(44100)

                        .bitrate(160000)

                        .rawFormat(AacRawFormat.NONE)

                        .specification(AacSpecification.MPEG4)

                        .audioDescriptionBroadcasterMix(

                                        AacAudioDescriptionBroadcasterMix.NORMAL)

                        .build())
```

```
      .build())
                                                                                .build())
                                                                .build())
                                                .build();
                        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(thumbsOutput).build())
                                                                .build())

.outputs(Output.builder().extension("jpg")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder()

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

      VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

      .codec(VideoCodec.FRAME_CAPTURE)

      .frameCaptureSettings(

                        FrameCaptureSettings
```

```
                                        .builder()

                                        .framerateNumerator(

                                                        1)

                                        .framerateDenominator(

                                                        1)

                                        .maxCaptures(10000000)

                                        .quality(80)

                                        .build())

      .build())
                                                                                .build())
                                                        .build())
                                         .build();

                       Map<String, AudioSelector> audioSelectors = new
HashMap<>();
                       audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
                                                        .offset(0).build());

                       JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                         .audioSelectors(audioSelectors)
                                         .videoSelector(

VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

.rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0)

.deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)
```

```
.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                                        .outputGroups(appleHLS, thumbs,
fileMp4).build();

                        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
                                        .settings(jobSettings)
                                        .build();

                        CreateJobResponse createJobResponse =
mc.createJob(createJobRequest);
                        return createJobResponse.job().id();

                } catch (MediaConvertException e) {
                        System.out.println(e.toString());
                        System.exit(0);
                }
                return "";
        }

        private final static Output createOutput(String customName,
                        String nameModifier,
                        String segmentModifier,
                        int qvbrMaxBitrate,
                        int qvbrQualityLevel,
                        int originWidth,
                        int originHeight,
                        int targetWidth) {

                int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
                Output output = null;
                try {
                        output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
```

```
                                        .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,

      489, 490, 491, 492)
                                                                                .build())
                                                                  .build())
                                        .videoDescription(

VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

      VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)
```

```
.dropFrameTimecode(DropFrameTimecode.ENABLED)
.codecSettings(VideoCodecSettings.builder()
      .codec(VideoCodec.H_264)
      .h264Settings(H264Settings
                      .builder()
                      .rateControlMode(
                                      H264RateControlMode.QVBR)
                      .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                      .qualityTuningLevel(
                                      H264QualityTuningLevel.SINGLE_PASS)
                      .qvbrSettings(H264QvbrSettings
                                      .builder()
                                      .qvbrQualityLevel(
                                                      qvbrQualityLevel)
                                      .build())
                      .codecLevel(H264CodecLevel.AUTO)
                      .codecProfile((targetHeight > 720
                                      && targetWidth > 1280)
                                                      ? H264CodecProfile.HIGH
                                                      : H264CodecProfile.MAIN)
                      .maxBitrate(qvbrMaxBitrate)
                      .framerateControl(
```

```
H264FramerateControl.INITIALIZE_FROM_SOURCE)
                        .gopSize(2.0)
                        .gopSizeUnits(H264GopSizeUnits.SECONDS)
                        .numberBFramesBetweenReferenceFrames(
                                        2)
                        .gopClosedCadence(
                                        1)
                        .gopBReference(H264GopBReference.DISABLED)
                        .slowPal(H264SlowPal.DISABLED)
                        .syntax(H264Syntax.DEFAULT)
                        .numberReferenceFrames(
                                        3)
                        .dynamicSubGop(H264DynamicSubGop.STATIC)
                        .fieldEncoding(H264FieldEncoding.PAFF)
                        .sceneChangeDetect(
                                        H264SceneChangeDetect.ENABLED)
                        .minIInterval(0)
                        .telecine(H264Telecine.NONE)
                        .framerateConversionAlgorithm(

H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                        .entropyEncoding(
```

```
                                        H264EntropyEncoding.CABAC)
                        .slices(1)
                        .unregisteredSeiTimecode(
                                        H264UnregisteredSeiTimecode.DISABLED)
                        .repeatPps(H264RepeatPps.DISABLED)
                        .adaptiveQuantization(
                                        H264AdaptiveQuantization.HIGH)
                        .spatialAdaptiveQuantization(
                                        H264SpatialAdaptiveQuantization.ENABLED)
                        .temporalAdaptiveQuantization(
                                        H264TemporalAdaptiveQuantization.ENABLED)
                        .flickerAdaptiveQuantization(
                                        H264FlickerAdaptiveQuantization.DISABLED)
                        .softness(0)
                        .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                        .build())
        .build())
                                                                        .build())
.audioDescriptions(AudioDescription.builder()
.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
.languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)
.codecSettings(AudioCodecSettings.builder()
```

```
 .codec(AudioCodec.AAC).aacSettings(AacSettings

       .builder()

       .codecProfile(AacCodecProfile.LC)

       .rateControlMode(

                       AacRateControlMode.CBR)

       .codingMode(AacCodingMode.CODING_MODE_2_0)

       .sampleRate(44100)

       .bitrate(96000)

       .rawFormat(AacRawFormat.NONE)

       .specification(AacSpecification.MPEG4)

       .audioDescriptionBroadcasterMix(

                       AacAudioDescriptionBroadcasterMix.NORMAL)

       .build())
                                                                    .build())
                                                        .build())
                                     .build();
               } catch (MediaConvertException e) {
                       e.printStackTrace();
                       System.exit(0);
               }
               return output;
        }
}
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInput1: String,
): String? {
    // Step 1: Describe endpoints to get the MediaConvert endpoint URL
    val describeResponse = mcClient.describeEndpoints(
        DescribeEndpointsRequest {
            maxResults = 1
        },
    )

    val endpointUrl = describeResponse.endpoints?.firstOrNull()?.url
        ?: error("No MediaConvert endpoint found")

    // Step 2: Create MediaConvert client with resolved endpoint
    val mediaConvert = MediaConvertClient.fromEnvironment {
        region = "us-west-2"
        endpointProvider = MediaConvertEndpointProvider {
            Endpoint(endpointUrl)
        }
    }

    // Output destination folder in S3 - put in 'output/' folder beside input
    val outputDestination = fileInput1.substringBeforeLast('/') + "/output/"

    // Step 3: Create the job request with minimal valid video codec settings
    val jobRequest = CreateJobRequest {
        role = mcRoleARN
        settings = JobSettings {
            inputs = listOf(
                Input {
```

```
                    fileInput = fileInput1
                },
            )
            outputGroups = listOf(
                OutputGroup {
                    outputGroupSettings = OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings = FileGroupSettings {
                            destination = outputDestination
                        }
                    }
                    outputs = listOf(
                        Output {
                            containerSettings = ContainerSettings {
                                container = ContainerType.Mp4
                            }
                            videoDescription = VideoDescription {
                                width = 1280
                                height = 720
                                codecSettings = VideoCodecSettings {
                                    codec = VideoCodec.H264
                                    h264Settings = H264Settings {
                                        rateControlMode =
H264RateControlMode.Qvbr
                                        qvbrSettings = H264QvbrSettings {
                                            qvbrQualityLevel = 7
                                        }
                                        maxBitrate = 5_000_000
                                        codecLevel = H264CodecLevel.Auto
                                        codecProfile = H264CodecProfile.Main
                                        framerateControl =
H264FramerateControl.InitializeFromSource
                                    }
                                }
                            }
                        },
                    )
                },
            )
        }
    }

    // Step 4: Call MediaConvert to create the job
    val response = mediaConvert.createJob(jobRequest)
```

```
    // Return the job ID or null if not found
    return response.job?.id
}
```

- Para obter detalhes da API, consulte a CreateJobreferência da API AWS SDK for Kotlin.

Use **`GetJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetJob`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o wrapper, o cliente e os locais do arquivo.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-
role-in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

Obter um trabalho pelo ID.

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Getting job information for Job ID {jobId}");
        var job = await wrapper.GetJobById(jobId);
        Console.WriteLine($"Job {job.Id} created on {job.CreatedAt:d} has status
 {job.Status}.");
        Console.WriteLine(new string('-', 80));
```

```
    /// <summary>
    /// Get the job information for a job by its ID.
    /// </summary>
    /// <param name="jobId">The ID of the job.</param>
    /// <returns>The Job object.</returns>
    public async Task<Job> GetJobById(string jobId)
    {
        var jobResponse = await _amazonMediaConvert.GetJobAsync(
                new GetJobRequest
                {
                    Id = jobId
                });

        return jobResponse.Job;
    }
```

- Para obter detalhes da API, consulte GetJoba Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve the information for a specific completed transcoding job.
/*!
  \param jobID: A job ID.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::getJob(const Aws::String &jobID,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    Aws::MediaConvert::Model::GetJobRequest request;
    request.SetId(jobID);
    const Aws::MediaConvert::Model::GetJobOutcome outcome = client.GetJob(
            request);
    if (outcome.IsSuccess()) {
        std::cout <<
 outcome.GetResult().GetJob().Jsonize().View().WriteReadable()
                  << std::endl;
    }
    else {
        std::cerr << "DescribeEndpoints error - " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }


    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetJoba Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como obter detalhes de um trabalho específico

O exemplo a seguir solicita as informações do trabalho com o ID `1234567890987-1ab2c3` que, neste exemplo, resultou em erro.

```
aws mediaconvert get-job \
    --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \
    --region region-name-1 \
```

```
    --id 1234567890987-1ab2c3
```

Use `describe-endpoints` para obter o endpoint específico da sua conta ou envie o comando sem o endpoint. O serviço retorna um erro e o endpoint.

Se a solicitação for bem-sucedida, o serviço retornará um arquivo JSON com as informações do trabalho, incluindo as configurações do trabalho, quaisquer erros retornados e outros dados do trabalho, da seguinte forma:

```
{
    "Job": {
        "Status": "ERROR",
        "Queue": "arn:aws:mediaconvert:region-name-1:012345678998:queues/Queue1",
        "Settings": {
            ...<truncated for brevity>...
        },
        "ErrorMessage": "Unable to open input file [s3://my-input-bucket/file-
name.mp4]: [Failed probe/open: [Failed to read data: AssumeRole failed]]",
        "ErrorCode": 1434,
        "Role": "arn:aws:iam::012345678998:role/MediaConvertServiceRole",
        "Arn": "arn:aws:mediaconvert:us-
west-1:012345678998:jobs/1234567890987-1ab2c3",
        "UserMetadata": {},
        "Timing": {
            "FinishTime": 1517442131,
            "SubmitTime": 1517442103,
            "StartTime": 1517442104
        },
        "Id": "1234567890987-1ab2c3",
        "CreatedAt": 1517442103
    }
}
```

Para obter mais informações, consulte Trabalhando com MediaConvert tarefas AWS elementares no Guia do MediaConvert usuário do AWS Elemental.

- Para obter detalhes da API, consulte GetJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

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

    public static void main(String[] args) {

        final String usage = "\n" +
                "  <jobId> \n\n" +
                "Where:\n" +
                "  jobId - The job id value.\n\n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
            GetJobRequest jobRequest = GetJobRequest.builder()
                    .id(jobId)
                    .build();

            GetJobResponse response = mc.getJob(jobRequest);
            System.out.println("The ARN of the job is " + response.job().arn());

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- Para obter detalhes da API, consulte GetJoba Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSpecificJob(mcClient: MediaConvertClient, jobId: String) {
```

```
    // 1. Discover the correct endpoint
    val res = mcClient.describeEndpoints(DescribeEndpointsRequest { maxResults =
 1 })
    var endpointUrl = res.endpoints?.firstOrNull()?.url
        ?: error(" No MediaConvert endpoint found")

    // 2. Create a new client using the endpoint
    val clientWithEndpoint = MediaConvertClient {
        region = "us-west-2"
        endpointUrl = endpointUrl
    }

    // 3. Get the job details
    val jobResponse = clientWithEndpoint.getJob(GetJobRequest { id = jobId })
    val job = jobResponse.job

    println("Job status: ${job?.status}")
    println("Job ARN: ${job?.arn}")
    println("Output group count: ${job?.settings?.outputGroups?.size}")
}
```

- Para obter detalhes da API, consulte a GetJobreferência da API AWS SDK for Kotlin.

Use **`ListJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListJobs`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o wrapper, o cliente e os locais do arquivo.

```
        // MediaConvert role Amazon Resource Name (ARN).
```

```
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-
role-in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

Liste os trabalhos usando um status específico.

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Listing all complete jobs.");
        var completeJobs = await wrapper.ListAllJobsByStatus(JobStatus.COMPLETE);
        completeJobs.ForEach(j =>
        {
            Console.WriteLine($"Job {j.Id} created on {j.CreatedAt:d} has status
 {j.Status}.");
        });
```

Liste os trabalhos usando um paginador.

```
    /// <summary>
    /// List all of the jobs with a particular status using a paginator.
    /// </summary>
    /// <param name="status">The status to use when listing jobs.</param>
    /// <returns>The list of jobs matching the status.</returns>
    public async Task<List<Job>> ListAllJobsByStatus(JobStatus? status = null)
    {
        var returnedJobs = new List<Job>();

        var paginatedJobs = _amazonMediaConvert.Paginators.ListJobs(
                new ListJobsRequest
                {
                    Status = status
                });
```

```
        // Get the entire list using the paginator.
        await foreach (var job in paginatedJobs.Jobs)
        {
            returnedJobs.Add(job);
        }

        return returnedJobs;
    }
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve a list of created jobs.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::listJobs(
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    bool result = true;
    Aws::String nextToken; // Used to handle paginated results.
    do {
        Aws::MediaConvert::Model::ListJobsRequest request;
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        const Aws::MediaConvert::Model::ListJobsOutcome outcome =
 client.ListJobs(
```

```
                request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::MediaConvert::Model::Job> &jobs =
                    outcome.GetResult().GetJobs();
            std::cout << jobs.size() << " jobs retrieved." << std::endl;
            for (const Aws::MediaConvert::Model::Job &job: jobs) {
                std::cout << "  " << job.Jsonize().View().WriteReadable() <<
 std::endl;
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "DescribeEndpoints error - " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
            break;

        }
    } while (!nextToken.empty());


    return result;
}
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como obter detalhes de todos os trabalhos de uma região

O exemplo a seguir solicita as informações de todos os trabalhos na região especificada.

```
aws mediaconvert list-jobs \
    --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \
    --region region-name-1
```

Use `describe-endpoints` para obter o endpoint específico da sua conta ou envie o comando sem o endpoint. O serviço retorna um erro e o endpoint.

Para obter mais informações, consulte Trabalhando com MediaConvert tarefas AWS elementares no Guia do MediaConvert usuário do AWS Elemental.

- Para obter detalhes da API, consulte ListJobsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListJobs {
    public static void main(String[] args) {
```

```
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            // Create the ListJobsRequest
            ListJobsRequest jobsRequest = ListJobsRequest.builder()
                    .maxResults(10)
                    .status("COMPLETE")
                    .build();

            // Call the listJobs operation
            ListJobsResponse jobsResponse = mc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listCompleteJobs(mcClient: MediaConvertClient) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
    val endpointURL = res.endpoints!![0].url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobsRequest =
        ListJobsRequest {
            maxResults = 10
            status = JobStatus.fromValue("COMPLETE")
        }

    val jobsResponse = mediaConvert.listJobs(jobsRequest)
    val jobs = jobsResponse.jobs
    if (jobs != null) {
        for (job in jobs) {
            println("The JOB ARN is ${job.arn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListJobsreferência da API AWS SDK for Kotlin.

Exemplos de código para MediaLive usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Elemental MediaLive com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- MediaLive Guia do usuário — Mais informações sobre MediaLive.
- MediaLive Referência da API — Detalhes sobre todas as MediaLive ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de MediaLive uso AWS SDKs
  - Ações para MediaLive usar AWS SDKs
    - Use ListInputscom um AWS SDK

Exemplos básicos de MediaLive uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Elemental MediaLive com AWS SDKs.

Exemplos

- Ações para MediaLive usar AWS SDKs
  - Use ListInputscom um AWS SDK

Ações para MediaLive usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar MediaLive ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Elemental MediaLive.

Exemplos

- Use ListInputscom um AWS SDK

Use **`ListInputs`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListInputs`.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste seus nomes MediaLive de entrada e ARNs na região.

```
async fn show_inputs(client: &Client) -> Result<(), Error> {
    let input_list = client.list_inputs().send().await?;

    for i in input_list.inputs() {
        let input_arn = i.arn().unwrap_or_default();
        let input_name = i.name().unwrap_or_default();

        println!("Input Name : {}", input_name);
        println!("Input ARN : {}", input_arn);
        println!();
    }

    Ok(())
```

```
}
```

- Para obter detalhes da API, consulte a ListInputsreferência da API AWS SDK for Rust.

Exemplos de código para MediaPackage usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Elemental MediaPackage com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- MediaPackage Guia do usuário — Mais informações sobre MediaPackage.
- MediaPackage Referência da API — Detalhes sobre todas as MediaPackage ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de MediaPackage uso AWS SDKs
  - Ações para MediaPackage usar AWS SDKs
    - Use ListChannelscom um AWS SDK ou CLI
    - Use ListOriginEndpointscom um AWS SDK ou CLI

Exemplos básicos de MediaPackage uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Elemental MediaPackage com AWS SDKs.

Exemplos

- Ações para MediaPackage usar AWS SDKs

- Use ListChannelscom um AWS SDK ou CLI
- Use ListOriginEndpointscom um AWS SDK ou CLI

Ações para MediaPackage usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar MediaPackage ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Elemental MediaPackage.

Exemplos

- Use ListChannelscom um AWS SDK ou CLI
- Use ListOriginEndpointscom um AWS SDK ou CLI

Use **`ListChannels`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListChannels`.

CLI

AWS CLI

Para listar todos os canais

O `list-channels` comando a seguir lista todos os canais que estão configurados na AWS conta atual.

```
aws mediapackage list-channels
```

Saída:

```
{
    "Channels": [
        {
            "Arn": "arn:aws:mediapackage:us-
west-2:111222333:channels/584797f1740548c389a273585dd22a63",
            "HlsIngest": {
                "IngestEndpoints": [
```

```
                    {
                        "Id": "584797f1740548c389a273585dd22a63",
                        "Password": "webdavgeneratedpassword1",
                        "Url": "https://9be9c4405c474882.mediapackage.us-
west-2.amazonaws.com/in/
v2/584797f1740548c389a273585dd22a63/584797f1740548c389a273585dd22a63/channel",
                        "Username": "webdavgeneratedusername1"
                    },
                    {
                        "Id": "7d187c8616fd455f88aaa5a9fcf74442",
                        "Password": "webdavgeneratedpassword2",
                        "Url": "https://7bf454c57220328d.mediapackage.us-
west-2.amazonaws.com/in/
v2/584797f1740548c389a273585dd22a63/7d187c8616fd455f88aaa5a9fcf74442/channel",
                        "Username": "webdavgeneratedusername2"
                    }
                ]
            },
            "Id": "test",
            "Tags": {}
        }
    ]
}
```

Para obter mais informações, consulte Visualizando detalhes do canal no Guia do MediaPackage usuário do AWS Elemental.

- Para obter detalhes da API, consulte ListChannelsem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar canais ARNs e as descrições.

```
async fn show_channels(client: &Client) -> Result<(), Error> {
```

```
    let list_channels = client.list_channels().send().await?;

    println!("Channels:");

    for c in list_channels.channels() {
        let description = c.description().unwrap_or_default();
        let arn = c.arn().unwrap_or_default();

        println!("  Description : {}", description);
        println!("  ARN :         {}", arn);
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListChannelsreferência da API AWS SDK for Rust.

Use **`ListOriginEndpoints`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListOriginEndpoints`.

CLI

AWS CLI

Para listar todos os endpoints de origem de um canal

O comando `list-origin-endpoints` a seguir lista todos endpoints de origem que estão configurados no canal `test`.

```
aws mediapackage list-origin-endpoints \
    --channel-id test
```

Saída:

```
{
    "OriginEndpoints": [
        {
            "Arn": "arn:aws:mediapackage:us-
west-2:111222333:origin_endpoints/247cff871f2845d3805129be22f2c0a2",
```

```
            "ChannelId": "test",
            "DashPackage": {
                "ManifestLayout": "FULL",
                "ManifestWindowSeconds": 60,
                "MinBufferTimeSeconds": 30,
                "MinUpdatePeriodSeconds": 15,
                "PeriodTriggers": [],
                "Profile": "NONE",
                "SegmentDurationSeconds": 2,
                "SegmentTemplateFormat": "NUMBER_WITH_TIMELINE",
                "StreamSelection": {
                    "MaxVideoBitsPerSecond": 2147483647,
                    "MinVideoBitsPerSecond": 0,
                    "StreamOrder": "ORIGINAL"
                },
                "SuggestedPresentationDelaySeconds": 25
            },
            "Id": "tester2",
            "ManifestName": "index",
            "StartoverWindowSeconds": 0,
            "Tags": {},
            "TimeDelaySeconds": 0,
            "Url": "https://8343f7014c0ea438.mediapackage.us-
west-2.amazonaws.com/out/v1/247cff871f2845d3805129be22f2c0a2/index.mpd",
            "Whitelist": []
        },
        {
            "Arn": "arn:aws:mediapackage:us-
west-2:111222333:origin_endpoints/869e237f851549e9bcf10e3bc2830839",
            "ChannelId": "test",
            "HlsPackage": {
                "AdMarkers": "NONE",
                "IncludeIframeOnlyStream": false,
                "PlaylistType": "EVENT",
                "PlaylistWindowSeconds": 60,
                "ProgramDateTimeIntervalSeconds": 0,
                "SegmentDurationSeconds": 6,
                "StreamSelection": {
                    "MaxVideoBitsPerSecond": 2147483647,
                    "MinVideoBitsPerSecond": 0,
                    "StreamOrder": "ORIGINAL"
                },
                "UseAudioRenditionGroup": false
            },
```

```
            "Id": "tester",
            "ManifestName": "index",
            "StartoverWindowSeconds": 0,
            "Tags": {},
            "TimeDelaySeconds": 0,
            "Url": "https://8343f7014c0ea438.mediapackage.us-
west-2.amazonaws.com/out/v1/869e237f851549e9bcf10e3bc2830839/index.m3u8",
            "Whitelist": []
        }
    ]
}
```

Para obter mais informações, consulte Visualização de todos os endpoints associados a um canal no Guia do MediaPackage usuário do AWS Elemental.

- Para obter detalhes da API, consulte ListOriginEndpointsem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar descrições de endpoint e URLs.

```
async fn show_endpoints(client: &Client) -> Result<(), Error> {
    let or_endpoints = client.list_origin_endpoints().send().await?;

    println!("Endpoints:");

    for e in or_endpoints.origin_endpoints() {
        let endpoint_url = e.url().unwrap_or_default();
        let endpoint_description = e.description().unwrap_or_default();
        println!("  Description: {}", endpoint_description);
        println!("  URL :        {}", endpoint_url);
        println!();
```

```
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListOriginEndpointsreferência da API AWS SDK for Rust.

Exemplos de código para o Migration Hub usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Migration Hub com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário do Migration Hub: mais informações sobre o Migration Hub.
- Referência da API Migration Hub: detalhes sobre todas as ações disponíveis do Migration Hub.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Migration Hub usando AWS SDKs
  - Ações para o Migration Hub usando AWS SDKs
    - Use DeleteProgressUpdateStreamcom um AWS SDK
    - Use DescribeApplicationStatecom um AWS SDK
    - Use DescribeMigrationTaskcom um AWS SDK
    - Use ImportMigrationTaskcom um AWS SDK
    - Use ListApplicationscom um AWS SDK

- Use ListCreatedArtifactscom um AWS SDK
- Use ListMigrationTaskscom um AWS SDK

Exemplos básicos para o Migration Hub usando AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Migration Hub com AWS SDKs.

Exemplos

- Ações para o Migration Hub usando AWS SDKs
  - Use DeleteProgressUpdateStreamcom um AWS SDK
  - Use DescribeApplicationStatecom um AWS SDK
  - Use DescribeMigrationTaskcom um AWS SDK
  - Use ImportMigrationTaskcom um AWS SDK
  - Use ListApplicationscom um AWS SDK
  - Use ListCreatedArtifactscom um AWS SDK
  - Use ListMigrationTaskscom um AWS SDK

Ações para o Migration Hub usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Migration Hub com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Migration Hub.

Exemplos

- Use DeleteProgressUpdateStreamcom um AWS SDK
- Use DescribeApplicationStatecom um AWS SDK
- Use DescribeMigrationTaskcom um AWS SDK
- Use ImportMigrationTaskcom um AWS SDK
- Use ListApplicationscom um AWS SDK
- Use ListCreatedArtifactscom um AWS SDK

- Use ListMigrationTaskscom um AWS SDK

Use **DeleteProgressUpdateStream**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteProgressUpdateStream`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <progressStream>\s

                Where:
                    progressStream - the name of a progress stream to delete.\s
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    public static void deleteStream(MigrationHubClient migrationClient, String
 streamName) {
        try {
            DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
 DeleteProgressUpdateStreamRequest
                    .builder()
                    .progressUpdateStreamName(streamName)
                    .build();


 migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
            System.out.println(streamName + " is deleted");

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteProgressUpdateStreama Referência AWS SDK for Java 2.x da API.

Use **`DescribeApplicationState`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeApplicationState`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeAppState <appId>\s

                Where:
                    appId -  the application id value.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String appId = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeApplicationState(migrationClient, appId);
        migrationClient.close();
    }

    public static void describeApplicationState(MigrationHubClient
 migrationClient, String appId) {
        try {
            DescribeApplicationStateRequest applicationStateRequest =
 DescribeApplicationStateRequest.builder()
                    .applicationId(appId)
                    .build();

            DescribeApplicationStateResponse applicationStateResponse =
 migrationClient
                    .describeApplicationState(applicationStateRequest);
            System.out.println("The application status is " +
 applicationStateResponse.applicationStatusAsString());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeApplicationStatea Referência AWS SDK for Java 2.x da API.

Use **`DescribeMigrationTask`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeMigrationTask`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

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

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

                Usage:
                    DescribeMigrationTask <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length < 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeMigTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void describeMigTask(MigrationHubClient migrationClient, String
 migrationTask,
            String progressStream) {
        try {
            DescribeMigrationTaskRequest migrationTaskRequestRequest =
 DescribeMigrationTaskRequest.builder()
                    .progressUpdateStream(progressStream)
                    .migrationTaskName(migrationTask)
                    .build();

            DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
                    .describeMigrationTask(migrationTaskRequestRequest);
            System.out.println("The name is " +
 migrationTaskResponse.migrationTask().migrationTaskName());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeMigrationTaska Referência AWS SDK for Java 2.x da API.

Use **`ImportMigrationTask`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ImportMigrationTask`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        importMigrTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void importMigrTask(MigrationHubClient migrationClient, String
 migrationTask, String progressStream) {
        try {
            CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
 CreateProgressUpdateStreamRequest.builder()
                    .progressUpdateStreamName(progressStream)
                    .dryRun(false)
                    .build();


 migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
            ImportMigrationTaskRequest migrationTaskRequest =
 ImportMigrationTaskRequest.builder()
                    .migrationTaskName(migrationTask)
                    .progressUpdateStream(progressStream)
                    .dryRun(false)
                    .build();

            migrationClient.importMigrationTask(migrationTaskRequest);

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ImportMigrationTaska Referência AWS SDK for Java 2.x da API.

Use **`ListApplications`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListApplications`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listApps(migrationClient);
        migrationClient.close();
    }
```

```
    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
 ListApplicationStatesRequest.builder()
                    .maxResults(10)
                    .build();

            ListApplicationStatesResponse response =
 migrationClient.listApplicationStates(applicationStatesRequest);
            List<ApplicationState> apps = response.applicationStateList();
            for (ApplicationState appState : apps) {
                System.out.println("App Id is " + appState.applicationId());
                System.out.println("The status is " +
 appState.applicationStatus().toString());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListApplicationsa Referência AWS SDK for Java 2.x da API.

Use **`ListCreatedArtifacts`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListCreatedArtifacts`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
 ListCreatedArtifactsRequest.builder()
                    .maxResults(10)
                    .migrationTaskName("SampleApp5")
                    .progressUpdateStream("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
 migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
```

```
                System.out.println("APp Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListCreatedArtifactsa Referência AWS SDK for Java 2.x da API.

Use **`ListMigrationTasks`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListMigrationTasks`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listMigrTasks(migrationClient);
        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
 ListMigrationTasksRequest.builder()
                    .maxResults(10)
                    .build();

            ListMigrationTasksResponse response =
 migrationClient.listMigrationTasks(listMigrationTasksRequest);
            List<MigrationTaskSummary> migrationList =
 response.migrationTaskSummaryList();
            for (MigrationTaskSummary migration : migrationList) {
                System.out.println("Migration task name is " +
 migration.migrationTaskName());
                System.out.println("The Progress update stream is " +
 migration.progressUpdateStream());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListMigrationTasksa Referência AWS SDK for Java 2.x da API.

Exemplos de código para o Amazon MSK usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Managed Streaming para Apache Kafka AWS com um kit de desenvolvimento de software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon MSK: mais informações sobre o Amazon MSK.
- Referência de API do Amazon MSK: detalhes sobre todas as ações disponíveis do Amazon MSK.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para o Amazon MSK usando AWS SDKs
  - Conceitos básicos de streaming gerenciado
- Exemplos de tecnologia sem servidor para o Amazon MSK
  - Invocar uma função do Lambda em um gatinho do Amazon MSK

Cenários para o Amazon MSK usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon MSK com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon MSK ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos de streaming gerenciado

Conceitos básicos de streaming gerenciado

O exemplo de código a seguir mostra como:

- Criar um cluster do Amazon MSK.
- Criar permissões do IAM para acessar o MSK.
- Criar uma máquina cliente
- Obter os agentes de bootstrap.
- Configurar a máquina cliente.
- Crie um tópico e produce/consume dados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
```

```
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
```

```
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)
```

```
                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
```

```
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
 to terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
 "$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are
 removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
```

```
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
```

```
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)
```

```
            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
```

```
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
```

```
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
```

```
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
```

```
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"
```

```
# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi
```

```
echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"
```

```
# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)
```

```
if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
```

```
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
```

```
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
```

```
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
```

```
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
```

```
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
```

```
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
```

```
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
```

```
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
```

```
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress

- CreateCluster
- CreateInstanceProfile
- CreateKeyPair
- CreatePolicy
- CreateRole
- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress

- RevokeSecurityGroupIngress

- RunInstances
- TerminateInstances
- Wait

Exemplos de tecnologia sem servidor para o Amazon MSK

Os exemplos de código a seguir mostram como usar o Amazon MSK com AWS SDKs.

Exemplos

- Invocar uma função do Lambda em um gatinho do Amazon MSK

Invocar uma função do Lambda em um gatinho do Amazon MSK

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento disparado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando .NET.

```
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KafkaEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
```

```
namespace MSKLambda;

public class Function
{


    /// <param name="input">The event for the Lambda function handler to
 process.</param>
    /// <param name="context">The ILambdaContext that provides methods for
 logging and describing the Lambda environment.</param>
    /// <returns></returns>
    public void FunctionHandler(KafkaEvent evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Records)
        {
            Console.WriteLine("Key:" + record.Key);
            foreach (var eventRecord in record.Value)
            {
                var valueBytes = eventRecord.Value.ToArray();
                var valueText = Encoding.UTF8.GetString(valueBytes);

                Console.WriteLine("Message:" + valueText);
            }
        }
    }


}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando Go.

```
package main

import (
 "encoding/base64"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
 for key, records := range event.Records {
  fmt.Println("Key:", key)

  for _, record := range records {
   fmt.Println("Record:", record)

   decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
   message := string(decodedValue)
   fmt.Println("Message:", message)
  }
 }
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do Amazon MSK com o uso do JavaScript Lambda.

```
exports.handler = async (event) => {
    // Iterate through keys
    for (let key in event.records) {
      console.log('Key: ', key)
      // Iterate through records
      event.records[key].map((record) => {
        console.log('Record: ', record)
        // Decode base64
        const msg = Buffer.from(record.value, 'base64').toString()
        console.log('Message:', msg)
      })
    }
}
```

Consumindo um evento do Amazon MSK com o uso do TypeScript Lambda.

```
import { MSKEvent, Context } from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "msk-handler-sample",
});

export const handler = async (
  event: MSKEvent,
  context: Context
): Promise<void> => {
  for (const [topic, topicRecords] of Object.entries(event.records)) {
    logger.info(`Processing key: ${topic}`);

    // Process each record in the partition
    for (const record of topicRecords) {
      try {
        // Decode the message value from base64
        const decodedMessage = Buffer.from(record.value, 'base64').toString();

        logger.info({
          message: decodedMessage
```

```
        });
      }
      catch (error) {
        logger.error('Error processing event', { error });
        throw error;
      }
    };
  }
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

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

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Python.

```
import base64

def lambda_handler(event, context):
    # Iterate through keys
    for key in event['records']:
        print('Key:', key)
        # Iterate through records
        for record in event['records'][key]:
            print('Record:', record)
            # Decode base64
            msg = base64.b64decode(record['value']).decode('utf-8')
            print('Message:', msg)
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Ruby.

```
require 'base64'
```

```
def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando o Rust.

```
use aws_lambda_events::event::kafka::KafkaEvent;
use lambda_runtime::{run, service_fn, tracing, Error, LambdaEvent};
use base64::prelude::*;
use serde_json::{Value};
use tracing::{info};

/// Pre-Requisites:
/// 1. Install Cargo Lambda - see https://www.cargo-lambda.info/guide/getting-
started.html
/// 2. Add packages tracing, tracing-subscriber, serde_json, base64
///
/// This is the main body for the function.
/// Write your code inside it.
```

```
/// There are some code example in the following URLs:
/// - https://github.com/awslabs/aws-lambda-rust-runtime/tree/main/examples
/// - https://github.com/aws-samples/serverless-rust-demo/

async fn function_handler(event: LambdaEvent<KafkaEvent>) -> Result<Value, Error>
 {

    let payload = event.payload.records;

    for (_name, records) in payload.iter() {

        for record in records {

         let record_text = record.value.as_ref().ok_or("Value is None")?;
         info!("Record: {}", &record_text);

         // perform Base64 decoding
         let record_bytes = BASE64_STANDARD.decode(record_text)?;
         let message = std::str::from_utf8(&record_bytes)?;

         info!("Message: {}", message);
        }

    }

    Ok(().into())
}

#[tokio::main]
async fn main() -> Result<(), Error> {

    // required to enable CloudWatch error logging by the runtime
    tracing::init_default_subscriber();
    info!("Setup CW subscriber!");

    run(service_fn(function_handler)).await
}
```

Exemplos de código para Neptune usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Neptune com AWS um kit de desenvolvimento de software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Neptune: mais informações sobre o Neptune.
- Referência de API do Neptune: detalhes sobre todas as ações disponíveis do Neptune.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Netuno usando AWS SDKs
  - Olá, Amazon Neptune
  - Aprenda o básico de Netuno com um AWS SDK
  - Ações para Neptune usando AWS SDKs
    - Use createdBCluster com um AWS SDK
    - Use createdBInstance com um AWS SDK
    - Use CreateDBSubnetGroupcom um AWS SDK
    - Use CreateGraphcom um AWS SDK
    - Use DeleteDBCluster com um AWS SDK
    - Use DeletedBInstance com um AWS SDK
    - Use DeleteDBSubnetGroupcom um AWS SDK
    - Use DescribedBClusters com um AWS SDK
    - Use DescribedBInstances com um AWS SDK
    - Use ExecuteGremlinProfileQuerycom um AWS SDK

- Use ExecuteGremlinQuerycom um AWS SDK
- Use ExecuteOpenCypherExplainQuerycom um AWS SDK
- Use ExecuteQuerycom um AWS SDK
- Use o StartDBCluster com um AWS SDK
- Use o StopDBCluster com um AWS SDK

- Cenários para Netuno usando AWS SDKs
  - Introdução aos bancos de dados de grafos
  - Use a API do Amazon Neptune para desenvolver uma função do Lambda que consulta dados de gráficos

Exemplos básicos de Netuno usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Neptune com AWS SDKs.

Exemplos

- Olá, Amazon Neptune
- Aprenda o básico de Netuno com um AWS SDK
- Ações para Neptune usando AWS SDKs
  - Use createdBCluster com um AWS SDK
  - Use createdBInstance com um AWS SDK
  - Use CreateDBSubnetGroupcom um AWS SDK
  - Use CreateGraphcom um AWS SDK
  - Use DeleteDBCluster com um AWS SDK
  - Use DeletedBInstance com um AWS SDK
  - Use DeleteDBSubnetGroupcom um AWS SDK
  - Use DescribedBClusters com um AWS SDK
  - Use DescribedBInstances com um AWS SDK
  - Use ExecuteGremlinProfileQuerycom um AWS SDK
  - Use ExecuteGremlinQuerycom um AWS SDK
  - Use ExecuteOpenCypherExplainQuerycom um AWS SDK
  - Use ExecuteQuerycom um AWS SDK

- Use o StartDBCluster com um AWS SDK
- Use o StopDBCluster com um AWS SDK

Olá, Amazon Neptune

O exemplo de código a seguir mostra como começar a usar o Neptune.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloNeptune {
    public static void main(String[] args) {
        NeptuneAsyncClient neptuneClient = NeptuneAsyncClient.create();
        describeDbCluster(neptuneClient).join(); // This ensures the async code
 runs to completion
    }

    /**
     * Describes the Amazon Neptune DB clusters.
     *
     * @param neptuneClient the Neptune asynchronous client used to make the
 request
     * @return a {@link CompletableFuture} that completes when the operation is
 finished
     */
```

```
   public static CompletableFuture<Void> describeDbCluster(NeptuneAsyncClient
neptuneClient) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .maxRecords(20)
               .build();

       SdkPublisher<DescribeDbClustersResponse> paginator =
neptuneClient.describeDBClustersPaginator(request);
       CompletableFuture<Void> future = new CompletableFuture<>();

       paginator.subscribe(new Subscriber<DescribeDbClustersResponse>() {
           private Subscription subscription;

           @Override
           public void onSubscribe(Subscription s) {
               this.subscription = s;
               s.request(Long.MAX_VALUE); // request all items
           }

           @Override
           public void onNext(DescribeDbClustersResponse response) {
               response.dbClusters().forEach(cluster -> {
                   System.out.println("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                   System.out.println("Status: " + cluster.status());
               });
           }

           @Override
           public void onError(Throwable t) {
               future.completeExceptionally(t);
           }

           @Override
           public void onComplete() {
               future.complete(null);
           }
       });

       return future.whenComplete((result, throwable) -> {
           neptuneClient.close();
           if (throwable != null) {
               System.err.println("Error describing DB clusters: " +
throwable.getMessage());
```

```
            }
        });
    }
```

- Para obter detalhes da API, consulte DescribeDBClustersPaginatora Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
from botocore.exceptions import ClientError


def describe_db_clusters(neptune_client):
    """
    Describes the Amazon Neptune DB clusters using a paginator to handle multiple
 pages.
    Raises ClientError with 'ResourceNotFoundException' if no clusters are found.
    """
    paginator = neptune_client.get_paginator("describe_db_clusters")
    clusters_found = False

    for page in paginator.paginate():
        for cluster in page.get("DBClusters", []):
            clusters_found = True
            print(f"Cluster Identifier: {cluster['DBClusterIdentifier']}")
            print(f"Status: {cluster['Status']}")

    if not clusters_found:
        raise ClientError(
            {
                "Error": {
```

```
                    "Code": "ResourceNotFoundException",
                    "Message": "No Neptune DB clusters found."
                }
            },
            operation_name="DescribeDBClusters"
        )

def main():
    """
    Main entry point: creates the Neptune client and calls the describe
 operation.
    """
    neptune_client = boto3.client("neptune")
    try:
        describe_db_clusters(neptune_client)
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "ResourceNotFoundException":
            print(f"Resource not found: {e.response['Error']['Message']}")
        else:
            print(f"Unexpected ClientError: {e.response['Error']['Message']}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a DescribeDBClustersPaginatorReferência da API AWS SDK for Python (Boto3).

Aprenda o básico de Netuno com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar um grupo de sub-redes do Amazon Neptune.
- Criar um cluster do Neptune.
- Criar uma instância do Neptune.
- Conferir o status da instância do Neptune.
- Mostrar detalhes do cluster do Neptune.

- Interromper o cluster do Neptune.
- Iniciar o cluster do Neptune.
- Excluir os ativos do Neptune.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos do Neptune.

```
public class NeptuneScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneScenario.class);
    static Scanner scanner = new Scanner(System.in);
    static NeptuneActions neptuneActions = new NeptuneActions();

    public static void main(String[] args) {
        final String usage =
                """
                Usage:
                    <subnetGroupName> <clusterName> <dbInstanceId>

                Where:
                    subnetGroupName - The name of an existing Neptune DB subnet
 group that includes subnets in at least two Availability Zones.
                    clusterName     - The unique identifier for the Neptune DB
 cluster.
                    dbInstanceId    - The identifier for a specific Neptune DB
 instance within the cluster.
                """;
        String subnetGroupName = "neptuneSubnetGroup65";
        String clusterName = "neptuneCluster65";
        String dbInstanceId = "neptuneDB65";
```

```
        logger.info("""
                   Amazon Neptune is a fully managed graph
                   database service by AWS, designed specifically
                   for handling complex relationships and connected
                   datasets at scale. It supports two popular graph models:
                   property graphs (via openCypher and Gremlin) and RDF
                   graphs (via SPARQL). This makes Neptune ideal for
                   use cases such as knowledge graphs, fraud detection,
                   social networking, recommendation engines, and
                   network management, where relationships between
                   entities are central to the data.

                   Being fully managed, Neptune handles database
                   provisioning, patching, backups, and replication,
                   while also offering high availability and durability
                   within AWS's infrastructure.

                   For developers, programming with Neptune allows
                   for building intelligent, relationship-aware
                   applications that go beyond traditional tabular
                   databases. Developers can use the AWS SDK for Java
                   to automate infrastructure operations (via NeptuneClient).

                    Let's get started...
                    """);
        waitForInputToContinue(scanner);
        runScenario(subnetGroupName, dbInstanceId, clusterName);
    }

    public static void runScenario(String subnetGroupName, String dbInstanceId,
 String clusterName) {
        logger.info(DASHES);
        logger.info("1. Create a Neptune DB Subnet Group");
        logger.info("The Neptune DB subnet group is used when launching a Neptune
 cluster");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.createSubnetGroupAsync(subnetGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
```

```
                logger.error("The request failed due to service quota exceeded:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create a Neptune Cluster");
        logger.info("A Neptune Cluster allows you to store and query highly
connected datasets with low latency.");
        waitForInputToContinue(scanner);
        String dbClusterId;
        try {
            dbClusterId =
neptuneActions.createDBClusterAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a Neptune DB Instance");
        logger.info("In this step, we add a new database instance to the Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
        neptuneActions.createDBInstanceAsync(dbInstanceId, dbClusterId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
```

```
                logger.error("The request failed due to service quota exceeded:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Check the status of the Neptune DB Instance");
        logger.info("""
                    In this step, we will wait until the DB instance
                    becomes available. This may take around 10 minutes.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5.Show Neptune Cluster details");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.describeDBClustersAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("6. Stop the Amazon Neptune cluster");
        logger.info("""
                    Once stopped, this step polls the status
                    until the cluster is in a stopped state.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.stopDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "stopped");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Start the Amazon Neptune cluster");
        logger.info("""
                    Once started, this step polls the clusters
                    status until it's in an available state.
                    We will also poll the instance status.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.startDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "available");
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
```

```
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the Neptune Assets");
        logger.info("Would you like to delete the Neptune Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the Neptune assets.");
            try {
                neptuneActions.deleteNeptuneResourcesAsync(dbInstanceId,
 clusterName, subnetGroupName);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The request failed due to the resource not
 found: {}", cause.getMessage());
                } else {
                    logger.error("An unexpected error occurred.", cause);
                }
                return;
            }
        } else {
            logger.info("You selected not to delete Neptune assets.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(
                """
                Thank you for checking out the Amazon Neptune Service Use demo.
 We hope you
                learned something new, or got some inspiration for your own apps
 today.
                For more AWS code examples, have a look at:
                https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                """);
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
```

```
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper para métodos de SDK do Neptune.

```
public class NeptuneActions {
    private CompletableFuture<Void> instanceCheckFuture;
    private static NeptuneAsyncClient neptuneAsyncClient;
    private final Region region = Region.US_EAST_1;
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneActions.class);
    private final NeptuneClient neptuneClient =
 NeptuneClient.builder().region(region).build();

    /**
     * Retrieves an instance of the NeptuneAsyncClient.
     * <p>
     * This method initializes and returns a singleton instance of the
 NeptuneAsyncClient. The client
     * is configured with the following settings:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
```

```
     * The client is built using the NettyNioAsyncHttpClient.
     *
     * @return the singleton instance of the NeptuneAsyncClient
     */
    private static NeptuneAsyncClient getAsyncClient() {
        if (neptuneAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            neptuneAsyncClient = NeptuneAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return neptuneAsyncClient;
    }

    /**
     * Asynchronously deletes a set of Amazon Neptune resources in a defined
order.
     * <p>
     * The method performs the following operations in sequence:
     * <ol>
     *     <li>Deletes the Neptune DB instance identified by {@code
dbInstanceId}.</li>
     *     <li>Waits until the DB instance is fully deleted.</li>
     *     <li>Deletes the Neptune DB cluster identified by {@code
dbClusterId}.</li>
     *     <li>Deletes the Neptune DB subnet group identified by {@code
subnetGroupName}.</li>
     * </ol>
     * <p>
```

```
    * If any step fails, the subsequent operations are not performed, and the
exception
    * is logged. This method blocks the calling thread until all operations
complete.
    *
    * @param dbInstanceId      the ID of the Neptune DB instance to delete
    * @param dbClusterId       the ID of the Neptune DB cluster to delete
    * @param subnetGroupName   the name of the Neptune DB subnet group to delete
    */
   public void deleteNeptuneResourcesAsync(String dbInstanceId, String
dbClusterId, String subnetGroupName) {
       deleteDBInstanceAsync(dbInstanceId)
               .thenCompose(v -> waitUntilInstanceDeletedAsync(dbInstanceId))
               .thenCompose(v -> deleteDBClusterAsync(dbClusterId))
               .thenCompose(v -> deleteDBSubnetGroupAsync(subnetGroupName))
               .whenComplete((v, ex) -> {
                   if (ex != null) {
                       logger.info("Failed to delete Neptune resources: " +
ex.getMessage());
                   } else {
                       logger.info("Neptune resources deleted successfully.");
                   }
               })
               .join(); // Waits for the entire async chain to complete
   }

   /**
    * Deletes a subnet group.
    *
    * @param subnetGroupName the identifier of the subnet group to delete
    * @return a {@link CompletableFuture} that completes when the cluster has
been deleted
    */
   public CompletableFuture<Void> deleteDBSubnetGroupAsync(String
subnetGroupName) {
       DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(subnetGroupName)
               .build();

       return getAsyncClient().deleteDBSubnetGroup(request)
               .thenAccept(response -> logger.info("## Deleting Subnet Group: "
+ subnetGroupName));
   }
```

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has
been deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB
Cluster: " + clusterId));
    }

    public CompletableFuture<Void> waitUntilInstanceDeletedAsync(String
instanceId) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkInstanceDeletedRecursive(instanceId, startTime, future);
        return future;
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance
has been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB
Instance: " + instanceId));
    }
```

```
   private void checkInstanceDeletedRecursive(String instanceId, long startTime,
CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof NeptuneException &&
                               ((NeptuneException)
cause).awsErrorDetails().errorCode().equals("DBInstanceNotFound")) {
                           long elapsed = (System.currentTimeMillis() -
startTime) / 1000;
                           logger.info("\r Instance %s deleted after %ds%n",
instanceId, elapsed);
                           future.complete(null);
                           return;
                       }
                       future.completeExceptionally(new
CompletionException("Error polling DB instance", cause));
                       return;
                   }

                   String status =
response.dbInstances().get(0).dbInstanceStatus();
                   long elapsed = (System.currentTimeMillis() - startTime) /
1000;
                   System.out.printf("\r  Waiting: Instance %s status: %-10s
(%ds elapsed)", instanceId, status, elapsed);
                   System.out.flush();

                   CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                           .execute(() ->
checkInstanceDeletedRecursive(instanceId, startTime, future));
               });
   }


   public void waitForClusterStatus(String clusterId, String desiredStatus) {
       System.out.printf("Waiting for cluster '%s' to reach status '%s'...\n",
clusterId, desiredStatus);
```

```
        CompletableFuture<Void> future = new CompletableFuture<>();
        checkClusterStatusRecursive(clusterId, desiredStatus,
System.currentTimeMillis(), future);
        future.join();
    }

    private void checkClusterStatusRecursive(String clusterId, String
desiredStatus, long startTime, CompletableFuture<Void> future) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        getAsyncClient().describeDBClusters(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
cluster status", cause)
                        );
                        return;
                    }

                    List<DBCluster> clusters = response.dbClusters();
                    if (clusters.isEmpty()) {
                        future.completeExceptionally(new
RuntimeException("Cluster not found: " + clusterId));
                        return;
                    }

                    String currentStatus = clusters.get(0).status();
                    long elapsedSeconds = (System.currentTimeMillis() -
startTime) / 1000;
                    System.out.printf("\r Elapsed: %-20s  Cluster status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune cluster reached
desired status '%s' after %s.\n", desiredStatus, formatElapsedTime((int)
elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
```

```
                                .execute(() ->
checkClusterStatusRecursive(clusterId, desiredStatus, startTime, future));
                    }
                });
    }


    /**
     * Starts an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
        StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().startDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to start DB cluster: "
+ cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }

    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
```

```
   public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
       StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().stopDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }

                       throw new RuntimeException("Failed to stop DB cluster: "
+ cause.getMessage(), cause);
                   } else {
                       logger.info("DB Cluster stopped: " + clusterIdentifier);
                   }
               });
   }



   /**
    * Asynchronously describes the specified Amazon RDS DB cluster.
    *
    * @param clusterId the identifier of the DB cluster to describe
    * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
    * if an error occurs
    */
   public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .dbClusterIdentifier(clusterId)
               .build();

       return getAsyncClient().describeDBClusters(request)
               .thenAccept(response -> {
                   for (DBCluster cluster : response.dbClusters()) {
                       logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
```

```
                        logger.info("Status: " + cluster.status());
                        logger.info("Engine: " + cluster.engine());
                        logger.info("Engine Version: " +
cluster.engineVersion());
                        logger.info("Endpoint: " + cluster.endpoint());
                        logger.info("Reader Endpoint: " +
cluster.readerEndpoint());
                        logger.info("Availability Zones: " +
cluster.availabilityZones());
                        logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                        logger.info("VPC Security Groups:");
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB
cluster: " + cause.getMessage(), cause);
                });
    }


    public CompletableFuture<Void> checkInstanceStatus(String instanceId, String
desiredStatus) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
```

```
        checkStatusRecursive(instanceId, desiredStatus.toLowerCase(), startTime,
future);
        return future;
    }

    /**
     * Checks the status of a Neptune instance recursively until the desired
status is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate
the elapsed time
     * @param future         a {@link CompletableFuture} that will be completed
when the desired status is reached
     */
    private void checkStatusRecursive(String instanceId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
instance status", cause)
                        );
                        return;
                    }

                    List<DBInstance> instances = response.dbInstances();
                    if (instances.isEmpty()) {
                        future.completeExceptionally(new
RuntimeException("Instance not found: " + instanceId));
                        return;
                    }

                    String currentStatus = instances.get(0).dbInstanceStatus();
                    long elapsedSeconds = (System.currentTimeMillis() -
startTime) / 1000;
```

```
                    System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune instance reached
desired status '%s' after %s.\n", desiredStatus, formatElapsedTime((int)
elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }


    private String formatElapsedTime(int seconds) {
        int minutes = seconds / 60;
        int remainingSeconds = seconds % 60;

        if (minutes > 0) {
            return minutes + (minutes == 1 ? " min" : " mins") + ", " +
                    remainingSeconds + (remainingSeconds == 1 ? " sec" : "
secs");
        } else {
            return remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        }
    }


    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
     *
     * @param dbInstanceId the identifier for the new DB instance
     * @param dbClusterId  the identifier for the DB cluster that the new
instance will be a part of
     * @return a {@link CompletableFuture} that completes with the identifier of
the newly created DB instance
     * @throws CompletionException if the operation fails, with a cause of
either:
     *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
```

```
     *                            - a general exception with the failure message
     */
    public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
 String dbClusterId) {
        CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceId)
                .dbInstanceClass("db.r5.large")
                .engine("neptune")
                .dbClusterIdentifier(dbClusterId)
                .build();

        return getAsyncClient().createDBInstance(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was
 denied because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune
 DB instance: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String instanceId =
 response.dbInstance().dbInstanceIdentifier();
                    logger.info("Created Neptune DB Instance: " + instanceId);
                    return instanceId;
                });
    }


    /**
     * Creates a new Amazon Neptune DB cluster asynchronously.
     *
     * @param dbName the name of the DB cluster to be created
     * @return a CompletableFuture that, when completed, provides the ID of the
 created DB cluster
     * @throws CompletionException if the operation fails for any reason,
 including if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createDBClusterAsync(String dbName) {
        CreateDbClusterRequest request = CreateDbClusterRequest.builder()
                .dbClusterIdentifier(dbName)
```

```
                .engine("neptune")
                .deletionProtection(false)
                .backupRetentionPeriod(1)
                .build();

        return getAsyncClient().createDBCluster(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune
DB cluster: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String clusterId =
response.dbCluster().dbClusterIdentifier();
                    logger.info("DB Cluster created: " + clusterId);
                    return clusterId;
                });
    }


    /**
     * Creates a new DB subnet group asynchronously.
     *
     * @param groupName the name of the subnet group to create
     * @return a CompletableFuture that, when completed, returns the Amazon
Resource Name (ARN) of the created subnet group
     * @throws CompletionException if the operation fails, with a cause that may
be a ServiceQuotaExceededException if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

        // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster
and resources will be created
        String vpcId = getDefaultVpcId();
        logger.info("VPC is : " + vpcId);

        List<String> subnetList = getSubnetIds(vpcId);
        for (String subnetId : subnetList) {
```

```
            System.out.println("Subnet group:" +subnetId);
        }

        CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(groupName)
                .dbSubnetGroupDescription("Subnet group for Neptune cluster")
                .subnetIds(subnetList)
                .build();

        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }

    private List<String> getSubnetIds(String vpcId) {
        try (Ec2Client ec2 = Ec2Client.builder().region(region).build()) {
            DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                    .filters(builder -> builder.name("vpc-id").values(vpcId))
                    .build();

            DescribeSubnetsResponse response = ec2.describeSubnets(request);
            return response.subnets().stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());
        }
    }

    public static String getDefaultVpcId() {
        Ec2Client ec2 = Ec2Client.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        Filter myFilter = Filter.builder()
                .name("isDefault")
                .values("true")
                .build();

        List<Filter> filterList = new ArrayList<>();
        filterList.add(myFilter);

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .filters(filterList)
                .build();


        DescribeVpcsResponse response = ec2.describeVpcs(request);
        if (!response.vpcs().isEmpty()) {
            Vpc defaultVpc = response.vpcs().get(0);
            return defaultVpc.vpcId();
        } else {
            throw new RuntimeException("No default VPC found in this region.");
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDBCluster
  - CreateDBInstance
  - CreateDBSubnetGroup
  - CreateGraph
  - DeleteDBCluster
  - DeleteDBInstance
  - DeleteDBSubnetGroup
  - DescribeDBClusters
  - DescribeDBInstances
  - ExecuteGremlinProfileQuery

- ExecuteGremlinQuery
- ExecuteOpenCypherExplainQuery
- ExecuteQuery
- StartDBCluster
- StopDBCluster

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
import time
from botocore.exceptions import ClientError

# Constants used in this scenario
POLL_INTERVAL_SECONDS = 10
TIMEOUT_SECONDS = 1200  # 20 minutes

def delete_db_cluster(neptune_client, cluster_id: str):
    """
    Deletes a Neptune DB cluster and throws exceptions to the caller.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        cluster_id (str): The ID of the Neptune DB cluster to be deleted.

    Raises:
        ClientError: If the delete operation fails.
    """
    request = {
        'DBClusterIdentifier': cluster_id,
        'SkipFinalSnapshot': True
    }
```

```
    try:
        print(f"Deleting DB Cluster: {cluster_id}")
        neptune_client.delete_db_cluster(**request)

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBClusterNotFoundFault":
            print(f"Cluster '{cluster_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB cluster. {code}: {message}")
        raise

def format_elapsed_time(seconds: int) -> str:
    mins, secs = divmod(seconds, 60)
    hours, mins = divmod(mins, 60)
    return f"{hours:02}:{mins:02}:{secs:02}"


def delete_db_instance(neptune_client, instance_id: str):
    """
    Deletes a Neptune DB instance and waits for its deletion to complete.
    Raises exception to be handled by calling code.
    """
    print(f"Initiating deletion of DB Instance: {instance_id}")
    try:
        neptune_client.delete_db_instance(
            DBInstanceIdentifier=instance_id,
            SkipFinalSnapshot=True
        )

        print(f"Waiting for DB Instance '{instance_id}' to be deleted...")
        waiter = neptune_client.get_waiter('db_instance_deleted')
        waiter.wait(
            DBInstanceIdentifier=instance_id,
            WaiterConfig={
                'Delay': 30,
                'MaxAttempts': 40
            }
        )
```

```
        print(f"DB Instance '{instance_id}' successfully deleted.")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBInstanceNotFoundFault":
            print(f"Instance '{instance_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB instance. {code}: {message}")
        raise

def delete_db_subnet_group(neptune_client, subnet_group_name):
    """
    Deletes a Neptune DB subnet group synchronously using Boto3.

    Args:
        neptune_client (boto3.client): The Neptune client.
        subnet_group_name (str): The name of the DB subnet group to delete.

    Raises:
        ClientError: If the delete operation fails.
    """
    delete_group_request = {
        'DBSubnetGroupName': subnet_group_name
    }

    try:
        neptune_client.delete_db_subnet_group(**delete_group_request)
        print(f"# Deleting Subnet Group: {subnet_group_name}")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found or already
 deleted.")
        elif code == "AccessDeniedException":
```

```
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete subnet group. {code}: {message}")
        raise

def wait_for_cluster_status(
        neptune_client,
        cluster_id: str,
        desired_status: str,
        timeout_seconds: int = TIMEOUT_SECONDS,
        poll_interval_seconds: int = POLL_INTERVAL_SECONDS
):
    """
    Waits for a Neptune DB cluster to reach a desired status.

    Args:
        neptune_client (boto3.client): The Amazon Neptune client.
        cluster_id (str): The identifier of the Neptune DB cluster.
        desired_status (str): The target status (e.g., "available", "stopped").
        timeout_seconds (int): Max time to wait in seconds (default: 1200).
        poll_interval_seconds (int): Polling interval in seconds (default: 10).

    Raises:
        RuntimeError: If the desired status is not reached before timeout.
    """
    print(f"Waiting for cluster '{cluster_id}' to reach status
 '{desired_status}'...")
    start_time = time.time()

    while True:
        # Prepare request object
        describe_cluster_request = {
            'DBClusterIdentifier': cluster_id
        }

        # Call the Neptune API
        response =
 neptune_client.describe_db_clusters(**describe_cluster_request)
        clusters = response.get('DBClusters', [])
        current_status = clusters[0].get('Status') if clusters else None
        elapsed_seconds = int(time.time() - start_time)

        status_str = current_status if current_status else "Unknown"
```

```
        print(
            f"\r Elapsed: {format_elapsed_time(elapsed_seconds):<20}  Cluster
 status: {status_str:<20}",
            end="", flush=True
        )

        if current_status and current_status.lower() == desired_status.lower():
            print(
                f"\nNeptune cluster reached desired status '{desired_status}'
 after {format_elapsed_time(elapsed_seconds)}."
            )
            return

        if elapsed_seconds > timeout_seconds:
            raise RuntimeError(f"Timeout waiting for Neptune cluster to reach
 status: {desired_status}")

        time.sleep(poll_interval_seconds)


def start_db_cluster(neptune_client, cluster_identifier: str):
    """
    Starts an Amazon Neptune DB cluster and waits until it reaches 'available'.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: Propagates AWS API issues like resource not found.
        RuntimeError: If cluster doesn't reach 'available' within timeout.
    """
    try:
        # Initial wait in case the cluster was just stopped
        time.sleep(30)
        neptune_client.start_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
```

```
            print(f"Couldn't start DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling.
 It may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
 {message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
 flush=True)

        if status and status.lower() == 'available':
            print(f"\n# Cluster '{cluster_identifier}' is available.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster
 '{cluster_identifier}' to become available.")

        time.sleep(POLL_INTERVAL_SECONDS)


def stop_db_cluster(neptune_client, cluster_identifier: str):
    """
    Stops an Amazon Neptune DB cluster and waits until it's fully stopped.
```

```
    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: For AWS API errors (e.g., resource not found).
        RuntimeError: If the cluster doesn't stop within the timeout.
    """
    try:
        neptune_client.stop_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't stop DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling.
It may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
```

```
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
 flush=True)

        if status and status.lower() == 'stopped':
            print(f"\nCluster '{cluster_identifier}' is now stopped.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster
 '{cluster_identifier}' to stop.")

        time.sleep(POLL_INTERVAL_SECONDS)



def describe_db_clusters(neptune_client, cluster_id: str):
    """
    Describes details of a Neptune DB cluster, paginating if needed.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_id (str): The ID of the cluster to describe.

    Raises:
        ClientError: If there's an AWS API error (e.g., cluster not found).
    """
    paginator = neptune_client.get_paginator('describe_db_clusters')

    try:
        pages = paginator.paginate(DBClusterIdentifier=cluster_id)

        found = False
        for page in pages:
            for cluster in page.get('DBClusters', []):
                found = True
                print(f"Cluster Identifier:
 {cluster.get('DBClusterIdentifier')}")
                print(f"Status: {cluster.get('Status')}")
                print(f"Engine: {cluster.get('Engine')}")
                print(f"Engine Version: {cluster.get('EngineVersion')}")
                print(f"Endpoint: {cluster.get('Endpoint')}")
                print(f"Reader Endpoint: {cluster.get('ReaderEndpoint')}")
```

```
                print(f"Availability Zones: {cluster.get('AvailabilityZones')}")
                print(f"Subnet Group: {cluster.get('DBSubnetGroup')}")
                print("VPC Security Groups:")
                for vpc_group in cluster.get('VpcSecurityGroups', []):
                    print(f"  - {vpc_group.get('VpcSecurityGroupId')}")
                print(f"Storage Encrypted: {cluster.get('StorageEncrypted')}")
                print(f"IAM Auth Enabled:
 {cluster.get('IAMDatabaseAuthenticationEnabled')}")
                print(f"Backup Retention Period:
 {cluster.get('BackupRetentionPeriod')} days")
                print(f"Preferred Backup Window:
 {cluster.get('PreferredBackupWindow')}")
                print(f"Preferred Maintenance Window:
 {cluster.get('PreferredMaintenanceWindow')}")
                print("------")

        if not found:
            # Treat empty response as cluster not found
            raise ClientError(
                {"Error": {"Code": "DBClusterNotFound", "Message": f"No cluster
 found with ID '{cluster_id}'"}},
                "DescribeDBClusters"
            )

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        elif code == "DBClusterNotFound":
            print(f"Cluster '{cluster_id}' not found. Please verify the cluster
 ID.")
        else:
            print(f"Couldn't describe DB cluster. Here's why: {code}: {message}")
        raise

def check_instance_status(neptune_client, instance_id: str, desired_status: str):
    """
    Polls the status of a Neptune DB instance until it reaches desired_status.
    Uses pagination via describe_db_instances — even for a single instance.

    Raises:
```

```
      ClientError: If describe_db_instances fails (e.g., instance not found).
      RuntimeError: If timeout expires before reaching desired status.
    """
    paginator = neptune_client.get_paginator('describe_db_instances')
    start_time = time.time()

    while True:
        try:
            pages = paginator.paginate(DBInstanceIdentifier=instance_id)
            instances = []
            for page in pages:
                instances.extend(page.get('DBInstances', []))

        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBInstanceNotFound":
                print(f"Instance '{instance_id}' not found. Please verify the
instance ID.")
            else:
                print(f"Failed to describe DB instance. {code}: {message}")
            raise

        current_status = instances[0].get('DBInstanceStatus') if instances else
None
        elapsed = int(time.time() - start_time)

        print(f"\rElapsed: {format_elapsed_time(elapsed)}  Status:
{current_status}", end="", flush=True)

        if current_status and current_status.lower() == desired_status.lower():
            print(f"\nInstance '{instance_id}' reached '{desired_status}' in
{format_elapsed_time(elapsed)}.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for '{instance_id}' to reach
'{desired_status}'")

        time.sleep(POLL_INTERVAL_SECONDS)
```

```
def create_db_instance(neptune_client, db_instance_id: str, db_cluster_id: str) -
> str:
    try:
        request = {
            'DBInstanceIdentifier': db_instance_id,
            'DBInstanceClass': 'db.r5.large',
            'Engine': 'neptune',
            'DBClusterIdentifier': db_cluster_id
        }

        print(f"Creating Neptune DB Instance: {db_instance_id}")
        response = neptune_client.create_db_instance(**request)

        instance = response.get('DBInstance')
        if not instance or 'DBInstanceIdentifier' not in instance:
            raise RuntimeError("Instance creation succeeded but no ID returned.")

        print(f"Waiting for DB Instance '{db_instance_id}' to become
 available...")
        waiter = neptune_client.get_waiter('db_instance_available')
        waiter.wait(
            DBInstanceIdentifier=db_instance_id,
            WaiterConfig={'Delay': 30, 'MaxAttempts': 40}
        )

        print(f"DB Instance '{db_instance_id}' is now available.")
        return instance['DBInstanceIdentifier']

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't create DB instance. Here's why: {code}: {message}")
        raise

    except Exception as e:
        print(f"Unexpected error creating DB instance '{db_instance_id}': {e}")
        raise RuntimeError(f"Unexpected error creating DB instance
 '{db_instance_id}': {e}") from e
```

```
def create_db_cluster(neptune_client, db_name: str) -> str:
    """
    Creates a Neptune DB cluster and returns its identifier.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        db_name (str): The desired cluster identifier.

    Returns:
        str: The DB cluster identifier.

    Raises:
        RuntimeError: For any failure or AWS error, with a user-friendly message.
    """
    request = {
        'DBClusterIdentifier': db_name,
        'Engine': 'neptune',
        'DeletionProtection': False,
        'BackupRetentionPeriod': 1
    }

    try:
        response = neptune_client.create_db_cluster(**request)
        cluster = response.get('DBCluster') or {}

        cluster_id = cluster.get('DBClusterIdentifier')
        if not cluster_id:
            raise RuntimeError("Cluster created but no ID returned.")

        print(f"DB Cluster created: {cluster_id}")
        return cluster_id

    except ClientError as e:
        code = e.response["Error"]["Code"]
        message = e.response["Error"]["Message"]

        if code in ("ServiceQuotaExceededException",
 "DBClusterQuotaExceededFault"):
            raise RuntimeError("You have exceeded the quota for Neptune DB
 clusters.") from e
        else:
            raise RuntimeError(f"AWS error [{code}]: {message}") from e
```

```
    except Exception as e:
        raise RuntimeError(f"Unexpected error creating DB cluster '{db_name}':
 {e}") from e

def get_subnet_ids(vpc_id: str) -> list[str]:
    ec2_client = boto3.client('ec2')

    describe_subnets_request = {
        'Filters': [{'Name': 'vpc-id', 'Values': [vpc_id]}]
    }

    response = ec2_client.describe_subnets(**describe_subnets_request)
    subnets = response.get('Subnets', [])
    subnet_ids = [subnet['SubnetId'] for subnet in subnets if 'SubnetId' in
 subnet]
    return subnet_ids


def get_default_vpc_id() -> str:
    ec2_client = boto3.client('ec2')
    describe_vpcs_request = {
        'Filters': [{'Name': 'isDefault', 'Values': ['true']}]
    }

    response = ec2_client.describe_vpcs(**describe_vpcs_request)
    vpcs = response.get('Vpcs', [])
    if not vpcs:
        raise RuntimeError("No default VPC found in this region.")

    default_vpc_id = vpcs[0]['VpcId']
    print(f"Default VPC ID: {default_vpc_id}")
    return default_vpc_id


def create_subnet_group(neptune_client, group_name: str):
    """
    Creates a Neptune DB subnet group and returns its name and ARN.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        group_name (str): The desired name of the subnet group.

    Returns:
        tuple(str, str): (subnet_group_name, subnet_group_arn)
```

```
    Raises:
        RuntimeError: For quota errors or other AWS-related failures.
    """
    vpc_id = get_default_vpc_id()
    subnet_ids = get_subnet_ids(vpc_id)

    request = {
        'DBSubnetGroupName': group_name,
        'DBSubnetGroupDescription': 'My Neptune subnet group',
        'SubnetIds': subnet_ids,
        'Tags': [{'Key': 'Environment', 'Value': 'Dev'}]
    }

    try:
        response = neptune_client.create_db_subnet_group(**request)
        sg = response.get("DBSubnetGroup", {})
        name = sg.get("DBSubnetGroupName")
        arn = sg.get("DBSubnetGroupArn")

        if not name or not arn:
            raise RuntimeError("Response missing subnet group name or ARN.")

        print(f"Subnet group created: {name}")
        print(f"ARN: {arn}")
        return name, arn

    except ClientError as e:
        code = e.response["Error"]["Code"]
        msg = e.response["Error"]["Message"]

        if code == "ServiceQuotaExceededException":
            print("Subnet group quota exceeded.")
            raise RuntimeError("Subnet group quota exceeded.") from e
        else:
            print(f"AWS error [{code}]: {msg}")
            raise RuntimeError(f"AWS error [{code}]: {msg}") from e

    except Exception as e:
        print(f"Unexpected error creating subnet group '{group_name}': {e}")
        raise RuntimeError(f"Unexpected error creating subnet group
 '{group_name}': {e}") from e

def wait_for_input_to_continue():
```

```
    input("\nPress <ENTER> to continue...")
    print("Continuing with the program...\n")


def run_scenario(neptune_client, subnet_group_name: str, db_instance_id: str,
 cluster_name: str):
    print("-" * 88)
    print("1. Create a Neptune DB Subnet Group")
    wait_for_input_to_continue()

    try:
        name, arn = create_subnet_group(neptune_client, subnet_group_name)
        print(f"Subnet group successfully created: {name}")

        print("-" * 88)
        print("2. Create a Neptune Cluster")
        wait_for_input_to_continue()
        db_cluster_id = create_db_cluster(neptune_client, cluster_name)

        print("-" * 88)
        print("3. Create a Neptune DB Instance")
        wait_for_input_to_continue()
        create_db_instance(neptune_client, db_instance_id, cluster_name)

        print("-" * 88)
        print("4. Check the status of the Neptune DB Instance")
        print("""
        Even though you're targeting a single DB instance,
        describe_db_instances supports pagination and can return multiple pages.

        Handling paginated responses ensures your method continues to work
 reliably
        even if AWS returns large or paged results.
        """)
        wait_for_input_to_continue()
        check_instance_status(neptune_client, db_instance_id, "available")

        print("-" * 88)
        print("5. Show Neptune Cluster details")
        wait_for_input_to_continue()
        describe_db_clusters(neptune_client, db_cluster_id)

        print("-" * 88)
        print("6. Stop the Amazon Neptune cluster")
```

```
        print("""
            Boto3 doesn't currently offer a
            built-in waiter for stop_db_cluster,
            This example implements a custom polling
            strategy until the cluster is in a stopped state.
        """)
        wait_for_input_to_continue()
        stop_db_cluster(neptune_client, db_cluster_id)
        check_instance_status(neptune_client, db_instance_id, "stopped")

        print("-" * 88)
        print("7. Start the Amazon Neptune cluster")
        print("""
            Boto3 doesn't currently offer a
            built-in waiter for start_db_cluster,
            This example implements a custom polling
            strategy until the cluster is in an available state.
        """)
        wait_for_input_to_continue()
        start_db_cluster(neptune_client, db_cluster_id)
        wait_for_cluster_status(neptune_client, db_cluster_id, "available")
        check_instance_status(neptune_client, db_instance_id, "available")

        print("All Neptune resources are now available.")
        print("-" * 88)

        print("-" * 88)
        print("8. Delete the Neptune Assets")
        print("Would you like to delete the Neptune Assets? (y/n)")
        del_ans = input().strip().lower()

        if del_ans == "y":
            print("You selected to delete the Neptune assets.")

            delete_db_instance(neptune_client, db_instance_id)
            delete_db_cluster(neptune_client, db_cluster_id)
            delete_db_subnet_group(neptune_client, subnet_group_name)

            print("Neptune resources deleted successfully")

    except ClientError as ce:
        code = ce.response["Error"]["Code"]
```

```
        if code in ("DBInstanceNotFound", "DBInstanceNotFoundFault",
 "ResourceNotFound"):
            print(f"Instance '{db_instance_id}' not found.")
        elif code in ("DBClusterNotFound", "DBClusterNotFoundFault",
 "ResourceNotFoundFault"):
            print(f"Cluster '{cluster_name}' not found.")
        elif code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"AWS error [{code}]: {ce.response['Error']['Message']}")
            raise  # re-raise unexpected errors

    except RuntimeError as re:
        print(f"Runtime error or timeout: {re}")


def main():
    neptune_client = boto3.client('neptune')

    # Customize the following names to match your Neptune setup
    # (You must change these to unique values for your environment)
    subnet_group_name = "neptuneSubnetGroup111"
    cluster_name = "neptuneCluster111"
    db_instance_id = "neptuneDB111"

    print("""
    Amazon Neptune is a fully managed graph database service by AWS...
    Let's get started!
    """)
    wait_for_input_to_continue()
    run_scenario(neptune_client, subnet_group_name, db_instance_id, cluster_name)

    print("""
    Thank you for checking out the Amazon Neptune Service Use demo.
    For more AWS code examples, visit:
    https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-library.html
    """)

if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDBCluster
  - CreateDBInstance
  - CreateDBSubnetGroup
  - CreateGraph
  - DeleteDBCluster
  - DeleteDBInstance
  - DeleteDBSubnetGroup
  - DescribeDBClusters
  - DescribeDBInstances
  - ExecuteGremlinProfileQuery
  - ExecuteGremlinQuery
  - ExecuteOpenCypherExplainQuery
  - ExecuteQuery
  - StartDBCluster
  - StopDBCluster

Ações para Neptune usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Neptune com SDKs. AWS Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Neptune e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para Netuno usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Neptune.

Exemplos

- Use createdBCluster com um AWS SDK

- Use createdBInstance com um AWS SDK
- Use CreateDBSubnetGroupcom um AWS SDK
- Use CreateGraphcom um AWS SDK
- Use DeleteDBCluster com um AWS SDK
- Use DeletedBInstance com um AWS SDK
- Use DeleteDBSubnetGroupcom um AWS SDK
- Use DescribedBClusters com um AWS SDK
- Use DescribedBInstances com um AWS SDK
- Use ExecuteGremlinProfileQuerycom um AWS SDK
- Use ExecuteGremlinQuerycom um AWS SDK
- Use ExecuteOpenCypherExplainQuerycom um AWS SDK
- Use ExecuteQuerycom um AWS SDK
- Use o StartDBCluster com um AWS SDK
- Use o StopDBCluster com um AWS SDK

Use **`createdBCluster`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDBCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new Amazon Neptune DB cluster asynchronously.
     *
     * @param dbName the name of the DB cluster to be created
     * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
     * @throws CompletionException if the operation fails for any reason,
including if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createDBClusterAsync(String dbName) {
        CreateDbClusterRequest request = CreateDbClusterRequest.builder()
                .dbClusterIdentifier(dbName)
                .engine("neptune")
                .deletionProtection(false)
                .backupRetentionPeriod(1)
                .build();

        return getAsyncClient().createDBCluster(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune
DB cluster: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String clusterId =
response.dbCluster().dbClusterIdentifier();
                    logger.info("DB Cluster created: " + clusterId);
                    return clusterId;
                });
    }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_db_cluster(neptune_client, db_name: str) -> str:
    """
    Creates a Neptune DB cluster and returns its identifier.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        db_name (str): The desired cluster identifier.

    Returns:
        str: The DB cluster identifier.

    Raises:
        RuntimeError: For any failure or AWS error, with a user-friendly message.
    """
    request = {
        'DBClusterIdentifier': db_name,
        'Engine': 'neptune',
        'DeletionProtection': False,
        'BackupRetentionPeriod': 1
    }

    try:
        response = neptune_client.create_db_cluster(**request)
        cluster = response.get('DBCluster') or {}

        cluster_id = cluster.get('DBClusterIdentifier')
        if not cluster_id:
            raise RuntimeError("Cluster created but no ID returned.")

        print(f"DB Cluster created: {cluster_id}")
        return cluster_id
```

```
    except ClientError as e:
        code = e.response["Error"]["Code"]
        message = e.response["Error"]["Message"]

        if code in ("ServiceQuotaExceededException",
"DBClusterQuotaExceededFault"):
            raise RuntimeError("You have exceeded the quota for Neptune DB
clusters.") from e
        else:
            raise RuntimeError(f"AWS error [{code}]: {message}") from e

    except Exception as e:
        raise RuntimeError(f"Unexpected error creating DB cluster '{db_name}':
{e}") from e
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Python (Boto3).

Use **`createdBInstance`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDBInstance`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
```

```
    *
    * @param dbInstanceId the identifier for the new DB instance
    * @param dbClusterId  the identifier for the DB cluster that the new
instance will be a part of
    * @return a {@link CompletableFuture} that completes with the identifier of
the newly created DB instance
    * @throws CompletionException if the operation fails, with a cause of
either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune
DB instance: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String instanceId =
response.dbInstance().dbInstanceIdentifier();
                   logger.info("Created Neptune DB Instance: " + instanceId);
                   return instanceId;
               });
   }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_db_instance(neptune_client, db_instance_id: str, db_cluster_id: str) -
> str:
    try:
        request = {
            'DBInstanceIdentifier': db_instance_id,
            'DBInstanceClass': 'db.r5.large',
            'Engine': 'neptune',
            'DBClusterIdentifier': db_cluster_id
        }

        print(f"Creating Neptune DB Instance: {db_instance_id}")
        response = neptune_client.create_db_instance(**request)

        instance = response.get('DBInstance')
        if not instance or 'DBInstanceIdentifier' not in instance:
            raise RuntimeError("Instance creation succeeded but no ID returned.")

        print(f"Waiting for DB Instance '{db_instance_id}' to become
 available...")
        waiter = neptune_client.get_waiter('db_instance_available')
        waiter.wait(
            DBInstanceIdentifier=db_instance_id,
            WaiterConfig={'Delay': 30, 'MaxAttempts': 40}
        )

        print(f"DB Instance '{db_instance_id}' is now available.")
        return instance['DBInstanceIdentifier']
```

```
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't create DB instance. Here's why: {code}: {message}")
        raise

    except Exception as e:
        print(f"Unexpected error creating DB instance '{db_instance_id}': {e}")
        raise RuntimeError(f"Unexpected error creating DB instance
'{db_instance_id}': {e}") from e
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Python (Boto3).

Use **`CreateDBSubnetGroup`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDBSubnetGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new DB subnet group asynchronously.
    *
    * @param groupName the name of the subnet group to create
    * @return a CompletableFuture that, when completed, returns the Amazon
Resource Name (ARN) of the created subnet group
    * @throws CompletionException if the operation fails, with a cause that may
be a ServiceQuotaExceededException if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

       // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster
and resources will be created
       String vpcId = getDefaultVpcId();
       logger.info("VPC is : " + vpcId);

       List<String> subnetList = getSubnetIds(vpcId);
       for (String subnetId : subnetList) {
           System.out.println("Subnet group:" +subnetId);
       }

       CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(groupName)
               .dbSubnetGroupDescription("Subnet group for Neptune cluster")
               .subnetIds(subnetList)
               .build();

       return getAsyncClient().createDBSubnetGroup(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String name = response.dbSubnetGroup().dbSubnetGroupName();
                   String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                   logger.info("Subnet group created: " + name);
                   return arn;
```

```
                });
    }
```

- Para obter detalhes da API, consulte CreateDBSubnetGroupa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_subnet_group(neptune_client, group_name: str):
    """
    Creates a Neptune DB subnet group and returns its name and ARN.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        group_name (str): The desired name of the subnet group.

    Returns:
        tuple(str, str): (subnet_group_name, subnet_group_arn)

    Raises:
        RuntimeError: For quota errors or other AWS-related failures.
    """
    vpc_id = get_default_vpc_id()
    subnet_ids = get_subnet_ids(vpc_id)

    request = {
        'DBSubnetGroupName': group_name,
        'DBSubnetGroupDescription': 'My Neptune subnet group',
        'SubnetIds': subnet_ids,
        'Tags': [{'Key': 'Environment', 'Value': 'Dev'}]
    }
```

```
    try:
        response = neptune_client.create_db_subnet_group(**request)
        sg = response.get("DBSubnetGroup", {})
        name = sg.get("DBSubnetGroupName")
        arn = sg.get("DBSubnetGroupArn")

        if not name or not arn:
            raise RuntimeError("Response missing subnet group name or ARN.")

        print(f"Subnet group created: {name}")
        print(f"ARN: {arn}")
        return name, arn

    except ClientError as e:
        code = e.response["Error"]["Code"]
        msg = e.response["Error"]["Message"]

        if code == "ServiceQuotaExceededException":
            print("Subnet group quota exceeded.")
            raise RuntimeError("Subnet group quota exceeded.") from e
        else:
            print(f"AWS error [{code}]: {msg}")
            raise RuntimeError(f"AWS error [{code}]: {msg}") from e

    except Exception as e:
        print(f"Unexpected error creating subnet group '{group_name}': {e}")
        raise RuntimeError(f"Unexpected error creating subnet group
'{group_name}': {e}") from e
```

- Para obter detalhes da API, consulte a CreateDBSubnetGroupReferência da API AWS SDK for Python (Boto3).

Use **`CreateGraph`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateGraph`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Executes the process of creating a new Neptune graph.
    *
    * @param client        the Neptune graph client used to interact with the
Neptune service
    * @param graphName     the name of the graph to be created
    * @throws NeptuneGraphException if an error occurs while creating the graph
    */
   public static void executeCreateGraph(NeptuneGraphClient client, String
graphName) {
       try {
           // Create the graph request
           CreateGraphRequest request = CreateGraphRequest.builder()
                   .graphName(graphName)
                   .provisionedMemory(16)
                   .build();

           // Create the graph
           CreateGraphResponse response = client.createGraph(request);

           // Extract the graph name and ARN
           String createdGraphName = response.name();
           String graphArn = response.arn();
           String graphEndpoint = response.endpoint();

           System.out.println("Graph created successfully!");
           System.out.println("Graph Name: " + createdGraphName);
           System.out.println("Graph ARN: " + graphArn);
           System.out.println("Graph Endpoint: " +graphEndpoint );

       } catch (NeptuneGraphException e) {
```

```
            System.err.println("Failed to create graph: " +
 e.awsErrorDetails().errorMessage());
        } finally {
            client.close();
        }
    }
```

- Para obter detalhes da API, consulte CreateGraph a Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune
 cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a
 **peered VPC**

"""

GRAPH_NAME = "sample-analytics-graph"

def main():
```

```
    config = Config(retries={"total_max_attempts": 1, "mode": "standard"},
 read_timeout=None)
    client = boto3.client("neptune-graph", config=config)
    execute_create_graph(client, GRAPH_NAME)

def execute_create_graph(client, graph_name):
    try:
        print("Creating Neptune graph...")
        response = client.create_graph(
            graphName=graph_name,
            provisionedMemory = 16
        )

        created_graph_name = response.get("name")
        graph_arn = response.get("arn")
        graph_endpoint = response.get("endpoint")

        print("Graph created successfully!")
        print(f"Graph Name: {created_graph_name}")
        print(f"Graph ARN: {graph_arn}")
        print(f"Graph Endpoint: {graph_endpoint}")

    except ClientError as e:
        print(f"Failed to create graph: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"Failed to create graph: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a CreateGraphReferência da API AWS SDK for Python (Boto3).

Use **`DeleteDBCluster`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteDBCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a DB instance asynchronously.
    *
    * @param clusterId the identifier of the cluster to delete
    * @return a {@link CompletableFuture} that completes when the cluster has
been deleted
    */
   public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
       DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
               .dbClusterIdentifier(clusterId)
               .skipFinalSnapshot(true)
               .build();

       return getAsyncClient().deleteDBCluster(request)
               .thenAccept(response -> System.out.println("## Deleting DB
Cluster: " + clusterId));
   }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_db_cluster(neptune_client, cluster_id: str):
    """
    Deletes a Neptune DB cluster and throws exceptions to the caller.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        cluster_id (str): The ID of the Neptune DB cluster to be deleted.

    Raises:
        ClientError: If the delete operation fails.
    """
    request = {
        'DBClusterIdentifier': cluster_id,
        'SkipFinalSnapshot': True
    }

    try:
        print(f"Deleting DB Cluster: {cluster_id}")
        neptune_client.delete_db_cluster(**request)

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBClusterNotFoundFault":
            print(f"Cluster '{cluster_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB cluster. {code}: {message}")
        raise
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Python (Boto3).

Use **`DeletedBInstance com`** um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteDBInstance`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance
has been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB
Instance: " + instanceId));
    }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_db_instance(neptune_client, instance_id: str):
    """
    Deletes a Neptune DB instance and waits for its deletion to complete.
    Raises exception to be handled by calling code.
    """
    print(f"Initiating deletion of DB Instance: {instance_id}")
    try:
        neptune_client.delete_db_instance(
            DBInstanceIdentifier=instance_id,
            SkipFinalSnapshot=True
        )

        print(f"Waiting for DB Instance '{instance_id}' to be deleted...")
        waiter = neptune_client.get_waiter('db_instance_deleted')
        waiter.wait(
            DBInstanceIdentifier=instance_id,
            WaiterConfig={
                'Delay': 30,
                'MaxAttempts': 40
            }
        )

        print(f"DB Instance '{instance_id}' successfully deleted.")

    except ClientError as err:
        code = err.response["Error"]["Code"]
```

```
        message = err.response["Error"]["Message"]

        if code == "DBInstanceNotFoundFault":
            print(f"Instance '{instance_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't delete DB instance. {code}: {message}")
        raise
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Python (Boto3).

Use **`DeleteDBSubnetGroup`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteDBSubnetGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has
been deleted
     */
```

```
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String
 subnetGroupName) {
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: "
 + subnetGroupName));
    }
```

- Para obter detalhes da API, consulte DeleteDBSubnetGroupa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_db_subnet_group(neptune_client, subnet_group_name):
    """
    Deletes a Neptune DB subnet group synchronously using Boto3.

    Args:
        neptune_client (boto3.client): The Neptune client.
        subnet_group_name (str): The name of the DB subnet group to delete.

    Raises:
        ClientError: If the delete operation fails.
    """
    delete_group_request = {
        'DBSubnetGroupName': subnet_group_name
    }

    try:
```

```
        neptune_client.delete_db_subnet_group(**delete_group_request)
        print(f"# Deleting Subnet Group: {subnet_group_name}")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found or already
deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't delete subnet group. {code}: {message}")
        raise
```

- Para obter detalhes da API, consulte a DeleteDBSubnetGroupReferência da API AWS SDK for Python (Boto3).

Use **`DescribedBClusters`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeDBClusters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously describes the specified Amazon RDS DB cluster.
     *
     * @param clusterId the identifier of the DB cluster to describe
     * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
     * if an error occurs
     */
    public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        return getAsyncClient().describeDBClusters(request)
                .thenAccept(response -> {
                    for (DBCluster cluster : response.dbClusters()) {
                        logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                        logger.info("Status: " + cluster.status());
                        logger.info("Engine: " + cluster.engine());
                        logger.info("Engine Version: " +
cluster.engineVersion());
                        logger.info("Endpoint: " + cluster.endpoint());
                        logger.info("Reader Endpoint: " +
cluster.readerEndpoint());
                        logger.info("Availability Zones: " +
cluster.availabilityZones());
                        logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                        logger.info("VPC Security Groups:");
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
```

```
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB
 cluster: " + cause.getMessage(), cause);
                });
    }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def describe_db_clusters(neptune_client, cluster_id: str):
    """
    Describes details of a Neptune DB cluster, paginating if needed.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_id (str): The ID of the cluster to describe.

    Raises:
        ClientError: If there's an AWS API error (e.g., cluster not found).
    """
    paginator = neptune_client.get_paginator('describe_db_clusters')

    try:
```

```
        pages = paginator.paginate(DBClusterIdentifier=cluster_id)

        found = False
        for page in pages:
            for cluster in page.get('DBClusters', []):
                found = True
                print(f"Cluster Identifier:
{cluster.get('DBClusterIdentifier')}")
                print(f"Status: {cluster.get('Status')}")
                print(f"Engine: {cluster.get('Engine')}")
                print(f"Engine Version: {cluster.get('EngineVersion')}")
                print(f"Endpoint: {cluster.get('Endpoint')}")
                print(f"Reader Endpoint: {cluster.get('ReaderEndpoint')}")
                print(f"Availability Zones: {cluster.get('AvailabilityZones')}")
                print(f"Subnet Group: {cluster.get('DBSubnetGroup')}")
                print("VPC Security Groups:")
                for vpc_group in cluster.get('VpcSecurityGroups', []):
                    print(f"  - {vpc_group.get('VpcSecurityGroupId')}")
                print(f"Storage Encrypted: {cluster.get('StorageEncrypted')}")
                print(f"IAM Auth Enabled:
{cluster.get('IAMDatabaseAuthenticationEnabled')}")
                print(f"Backup Retention Period:
{cluster.get('BackupRetentionPeriod')} days")
                print(f"Preferred Backup Window:
{cluster.get('PreferredBackupWindow')}")
                print(f"Preferred Maintenance Window:
{cluster.get('PreferredMaintenanceWindow')}")
                print("------")

        if not found:
            # Treat empty response as cluster not found
            raise ClientError(
                {"Error": {"Code": "DBClusterNotFound", "Message": f"No cluster
found with ID '{cluster_id}'"}},
                "DescribeDBClusters"
            )

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
```

```
        elif code == "DBClusterNotFound":
            print(f"Cluster '{cluster_id}' not found. Please verify the cluster
ID.")
        else:
            print(f"Couldn't describe DB cluster. Here's why: {code}: {message}")
        raise
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Python (Boto3).

Use **`DescribedBInstances`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeDBInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Checks the status of a Neptune instance recursively until the desired
status is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate
the elapsed time
     * @param future         a {@link CompletableFuture} that will be completed
when the desired status is reached
     */
```

```
   private void checkStatusRecursive(String instanceId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       future.completeExceptionally(
                               new CompletionException("Error checking Neptune
instance status", cause)
                       );
                       return;
                   }

                   List<DBInstance> instances = response.dbInstances();
                   if (instances.isEmpty()) {
                       future.completeExceptionally(new
RuntimeException("Instance not found: " + instanceId));
                       return;
                   }

                   String currentStatus = instances.get(0).dbInstanceStatus();
                   long elapsedSeconds = (System.currentTimeMillis() -
startTime) / 1000;
                   System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                   System.out.flush();

                   if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                       System.out.printf("\r Neptune instance reached
desired status '%s' after %s.\n", desiredStatus, formatElapsedTime((int)
elapsedSeconds));
                       future.complete(null);
                   } else {
                       CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                               .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                   }
               });
   }
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def check_instance_status(neptune_client, instance_id: str, desired_status: str):
    """
    Polls the status of a Neptune DB instance until it reaches desired_status.
    Uses pagination via describe_db_instances — even for a single instance.

    Raises:
      ClientError: If describe_db_instances fails (e.g., instance not found).
      RuntimeError: If timeout expires before reaching desired status.
    """
    paginator = neptune_client.get_paginator('describe_db_instances')
    start_time = time.time()

    while True:
        try:
            pages = paginator.paginate(DBInstanceIdentifier=instance_id)
            instances = []
            for page in pages:
                instances.extend(page.get('DBInstances', []))

        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBInstanceNotFound":
                print(f"Instance '{instance_id}' not found. Please verify the
 instance ID.")
```

```
            else:
                print(f"Failed to describe DB instance. {code}: {message}")
            raise

        current_status = instances[0].get('DBInstanceStatus') if instances else
None
        elapsed = int(time.time() - start_time)

        print(f"\rElapsed: {format_elapsed_time(elapsed)}  Status:
{current_status}", end="", flush=True)

        if current_status and current_status.lower() == desired_status.lower():
            print(f"\nInstance '{instance_id}' reached '{desired_status}' in
{format_elapsed_time(elapsed)}.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for '{instance_id}' to reach
'{desired_status}'")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

Use **`ExecuteGremlinProfileQuery`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ExecuteGremlinProfileQuery`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Executes a Gremlin query against an Amazon Neptune database using the
provided {@link NeptunedataClient}.
    *
    * @param client the {@link NeptunedataClient} instance to use for executing
the Gremlin query
    */
   public static void executeGremlinQuery(NeptunedataClient client) {
       try {
           System.out.println("Querying Neptune...");
           ExecuteGremlinQueryRequest request =
ExecuteGremlinQueryRequest.builder()
                   .gremlinQuery("g.V().has('code', 'ANC')")
                   .build();

           ExecuteGremlinQueryResponse response =
client.executeGremlinQuery(request);

           System.out.println("Full Response:");
           System.out.println(response);

           // Retrieve and print the result
           if (response.result() != null) {
               System.out.println("Query Result:");
               System.out.println(response.result().toString());
           } else {
               System.out.println("No result returned from the query.");
           }
       } catch (NeptunedataException e) {
           System.err.println("Error calling Neptune: " +
e.awsErrorDetails().errorMessage());
       } catch (Exception e) {
```

```
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Para obter detalhes da API, consulte ExecuteGremlinProfileQuerya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Replace this with your actual Neptune endpoint
NEPTUNE_ENDPOINT = "https://[Specify Endpoint]:8182"

def main():
    """
    Entry point of the program. Initializes the Neptune client and executes the
 Gremlin query.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    execute_gremlin_query(neptune_client)


def execute_gremlin_query(neptune_client):
```

```
    """
    Executes a Gremlin query against an Amazon Neptune database.
    """
    try:
        print("Querying Neptune...")

        response = neptune_client.execute_gremlin_explain_query(
            gremlinQuery="g.V().has('code', 'ANC')"
        )

        print("Full Response:")
        print(response['output'].read().decode('UTF-8'))

    except ClientError as e:
        print(f"Error calling Neptune: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ExecuteGremlinProfileQueryReferência da API AWS SDK for Python (Boto3).

Use **`ExecuteGremlinQuery`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ExecuteGremlinQuery`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Executes a Gremlin PROFILE query using the provided NeptunedataClient.
    *
    * @param client The NeptunedataClient instance to be used for executing the
Gremlin PROFILE query.
    */
   private static void executeGremlinProfileQuery(NeptunedataClient client) {
       System.out.println("Executing Gremlin PROFILE query...");

       ExecuteGremlinProfileQueryRequest request =
ExecuteGremlinProfileQueryRequest.builder()
               .gremlinQuery("g.V().has('code', 'ANC')")
               .build();

       ExecuteGremlinProfileQueryResponse response =
client.executeGremlinProfileQuery(request);
       if (response.output() != null) {
           System.out.println("Query Profile Output:");
           System.out.println(response.output());
       } else {
           System.out.println("No output returned from the profile query.");
       }
   }
```

- Para obter detalhes da API, consulte ExecuteGremlinQuerya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune
 cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a
 **peered VPC**

"""

# Replace with your actual Neptune endpoint
NEPTUNE_ENDPOINT = "https://[Specify-Your-Endpoint]:8182"

def main():
    """
    Entry point of the program. Initializes the Neptune client and runs both
 EXPLAIN and PROFILE queries.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
```

```
    )

    try:
        run_profile_query(neptune_client)
    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

def run_profile_query(neptune_client):
    """
    Runs a PROFILE query on the Neptune graph database.
    """
    print("Running Gremlin PROFILE query...")

    try:
        response = neptune_client.execute_gremlin_profile_query(
            gremlinQuery="g.V().has('code', 'ANC')"
        )
        print("Profile Query Result:")
        output = response.get("output")
        if output:
            print(output.read().decode('utf-8'))
        else:
            print("No explain output returned.")
    except Exception as e:
        print(f"Failed to execute PROFILE query: {str(e)}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ExecuteGremlinQueryReferência da API AWS SDK for Python (Boto3).

Use **`ExecuteOpenCypherExplainQuery`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ExecuteOpenCypherExplainQuery`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Executes an OpenCypher EXPLAIN query using the provided Neptune data
client.
    *
    * @param client The Neptune data client to use for the query execution.
    */
   public static void executeGremlinQuery(NeptunedataClient client) {
       try {
           System.out.println("Executing OpenCypher EXPLAIN query...");
           ExecuteOpenCypherExplainQueryRequest request =
ExecuteOpenCypherExplainQueryRequest.builder()
                   .openCypherQuery("MATCH (n {code: 'ANC'}) RETURN n")
                   .explainMode("debug")
                   .build();

           ExecuteOpenCypherExplainQueryResponse response =
client.executeOpenCypherExplainQuery(request);

           if (response.results() != null) {
               System.out.println("Explain Results:");
               System.out.println(response.results().asUtf8String());
           } else {
               System.out.println("No explain results returned.");
           }

       } catch (NeptunedataException e) {
```

```
            System.err.println("Neptune error: " +
 e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Para obter detalhes da API, consulte ExecuteOpenCypherExplainQuerya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Replace with your actual Neptune endpoint URL
NEPTUNE_ENDPOINT = "https://<your-neptune-endpoint>:8182"

def main():
    """
    Entry point: Create Neptune client and execute different OpenCypher queries.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    execute_open_cypher_query_without_params(neptune_client)
```

```
    execute_open_cypher_query_with_params(neptune_client)
    execute_open_cypher_explain_query(neptune_client)

def execute_open_cypher_query_without_params(client):
    """
    Executes a simple OpenCypher query without parameters.
    """
    try:
        print("\nRunning OpenCypher query without parameters...")
        resp = client.execute_open_cypher_query(
            openCypherQuery="MATCH (n {code: 'ANC'}) RETURN n"
        )
        print("Results:")
        print(resp['results'])

    except Exception as e:
        print(f"Error in simple OpenCypher query: {str(e)}")


def execute_open_cypher_query_with_params(client):
    """
    Executes an OpenCypher query using parameters.
    """
    try:
        print("\nRunning OpenCypher query with parameters...")
        parameters = {'code': 'ANC'}
        resp = client.execute_open_cypher_query(
            openCypherQuery="MATCH (n {code: $code}) RETURN n",
            parameters=json.dumps(parameters)
        )
        print("Results:")
        print(resp['results'])

    except Exception as e:
        print(f"Error in parameterized OpenCypher query: {str(e)}")

def execute_open_cypher_explain_query(client):
    """
    Runs an OpenCypher EXPLAIN query in debug mode.
    """
    try:
        print("\nRunning OpenCypher EXPLAIN query (debug mode)...")
        resp = client.execute_open_cypher_explain_query(
            openCypherQuery="MATCH (n {code: 'ANC'}) RETURN n",
```

```
            explainMode="details"
        )
        results = resp.get('results')
        if results is None:
            print("No explain results returned.")
        else:
            try:
                print("Explain Results:")
                print(results.read().decode('UTF-8'))
            except Exception as e:
                print(f"Error in OpenCypher EXPLAIN query: {str(e)}")

    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ExecuteOpenCypherExplainQueryReferência da API AWS SDK for Python (Boto3).

Use **`ExecuteQuery`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ExecuteQuery`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Executes a Gremlin profile query on the Neptune Analytics graph.
    *
    * @param client       the {@link NeptuneGraphClient} instance to use for the
query
    * @param graphId      the identifier of the graph to execute the query on
    *
    * @throws NeptuneGraphException if an error occurs while executing the query
on the Neptune Graph
    * @throws Exception if an unexpected error occurs
    */
   public static void executeGremlinProfileQuery(NeptuneGraphClient client,
String graphId) {

       try {
           System.out.println("Running openCypher query on Neptune
Analytics...");

           ExecuteQueryRequest request = ExecuteQueryRequest.builder()
                   .graphIdentifier(graphId)
                   .queryString("MATCH (n {code: 'ANC'}) RETURN n")
                   .language("OPEN_CYPHER")
                   .build();

           ResponseInputStream<ExecuteQueryResponse> response =
client.executeQuery(request);
           try (BufferedReader reader = new BufferedReader(new
InputStreamReader(response, StandardCharsets.UTF_8))) {
               String result = reader.lines().collect(Collectors.joining("\n"));
               System.out.println("Query Result:");
               System.out.println(result);
           } catch (Exception e) {
```

```
                System.err.println("Error reading response: " + e.getMessage());
            }

        } catch (NeptuneGraphException e) {
            System.err.println("NeptuneGraph error: " +
 e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Para obter detalhes da API, consulte ExecuteQuerya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune
 cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a
 **peered VPC**
```

```
"""

GRAPH_ID = "<your-graph-id>"

def main():
    config = Config(retries={"total_max_attempts": 1, "mode": "standard"},
 read_timeout=None)
    client = boto3.client("neptune-graph", config=config)

    try:
        print("\n--- Running OpenCypher query without parameters ---")
        run_open_cypher_query(client, GRAPH_ID)

        print("\n--- Running OpenCypher query with parameters ---")
        run_open_cypher_query_with_params(client, GRAPH_ID)

        print("\n--- Running OpenCypher explain query ---")
        run_open_cypher_explain_query(client, GRAPH_ID)

    except Exception as e:
        print(f"Unexpected error in main: {e}")

def run_open_cypher_query(client, graph_id):
    """
    Run an OpenCypher query without parameters.
    """
    try:
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: 'ANC'}) RETURN n",
            language='OPEN_CYPHER'
        )
        print(resp['payload'].read().decode('UTF-8'))

    except client.exceptions.InternalServerException as e:
        print(f"InternalServerException: {e.response['Error']['Message']}")
    except ClientError as e:
        print(f"ClientError: {e.response['Error']['Message']}")
    except Exception as e:  # <--- ADD THIS BLOCK
        print(f"Unexpected error: {e}")

def run_open_cypher_query_with_params(client, graph_id):
    """
    Run an OpenCypher query with parameters.
```

```
    """
    try:
        parameters = {'code': 'ANC'}
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: $code}) RETURN n",
            language='OPEN_CYPHER',
            parameters=parameters
        )
        print(resp['payload'].read().decode('UTF-8'))

    except client.exceptions.InternalServerException as e:
        print(f"InternalServerException: {e.response['Error']['Message']}")
    except ClientError as e:
        print(f"ClientError: {e.response['Error']['Message']}")
    except Exception as e:  # <--- ADD THIS BLOCK
        print(f"Unexpected error: {e}")

def run_open_cypher_explain_query(client, graph_id):
    """
    Run an OpenCypher explain query (explainMode = "debug").
    """
    try:
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: 'ANC'}) RETURN n",
            language='OPEN_CYPHER',
            explainMode='DETAILS'
        )
        print(resp['payload'].read().decode('UTF-8'))

    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"Unexpected Boto3 error: {str(e)}")
    except Exception as e:  # <-- Add this generic catch
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ExecuteQueryReferência da API AWS SDK for Python (Boto3).

Use o **`StartDBCluster`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartDBCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Starts an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
       StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().startDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
```

```
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to start DB cluster: "
 + cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }
```

- Consulte detalhes da API em StartDBCluster na Referência de API do AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def start_db_cluster(neptune_client, cluster_identifier: str):
    """
    Starts an Amazon Neptune DB cluster and waits until it reaches 'available'.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: Propagates AWS API issues like resource not found.
        RuntimeError: If cluster doesn't reach 'available' within timeout.
    """
    try:
        # Initial wait in case the cluster was just stopped
        time.sleep(30)
        neptune_client.start_db_cluster(DBClusterIdentifier=cluster_identifier)
```

```
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't start DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling.
It may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'available':
            print(f"\n# Cluster '{cluster_identifier}' is available.")
            return

        if elapsed > TIMEOUT_SECONDS:
```

```
            raise RuntimeError(f"Timeout waiting for cluster
'{cluster_identifier}' to become available.")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Consulte detalhes da API em StartDBCluster na Referência de API do AWS SDK para Python (Boto3).

Use o **`StopDBCluster`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StopDBCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
        StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();
```

```
        return getAsyncClient().stopDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
 error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: "
 + cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }
```

- Consulte os detalhes da API em StopDBCluster na Referência de API do AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def stop_db_cluster(neptune_client, cluster_identifier: str):
    """
    Stops an Amazon Neptune DB cluster and waits until it's fully stopped.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.
```

```
    Raises:
        ClientError: For AWS API errors (e.g., resource not found).
        RuntimeError: If the cluster doesn't stop within the timeout.
    """
    try:
        neptune_client.stop_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't stop DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling.
It may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
flush=True)
```

```
        if status and status.lower() == 'stopped':
            print(f"\nCluster '{cluster_identifier}' is now stopped.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster
'{cluster_identifier}' to stop.")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Consulte detalhes da API em StopDBCluster na Referência de API do AWS SDK para Python (Boto3).

Cenários para Netuno usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Neptune AWS com SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Neptune ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução aos bancos de dados de grafos
- Use a API do Amazon Neptune para desenvolver uma função do Lambda que consulta dados de gráficos

Introdução aos bancos de dados de grafos

O exemplo de código a seguir mostra como:

- Criar uma VPC para um banco de dados do Neptune.
- Criar sub-redes em várias zonas de disponibilidade.
- Configurar a segurança do banco de dados do Neptune.

- Criar um grupo de sub-redes do Neptune.
- Criar uma instância e um cluster de banco de dados do Neptune.
- Adicionar dados a um banco de dados de grafos.
- Consultar um banco de dados de grafos.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Neptune Getting Started Script
# This script creates an Amazon Neptune database cluster and demonstrates basic
 operations

# Set up logging
LOG_FILE="neptune-setup.log"
echo "Starting Neptune setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
```

```
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..." | tee -a "$LOG_FILE"

    # Only attempt to delete resources that were successfully created
    if [[ -n "$DB_INSTANCE_ID" ]]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"
        log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"
    fi

    if [[ -n "$DB_CLUSTER_ID" ]]; then
        echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --skip-final-snapshot"
    fi

    if [[ -n "$DB_SUBNET_GROUP" ]]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"
    fi

    if [[ -n "$SECURITY_GROUP_ID" ]]; then
        echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
    fi

    if [[ -n "$SUBNET_IDS" ]]; then
        for SUBNET_ID in $SUBNET_IDS; do
            echo "Deleting subnet $SUBNET_ID..." | tee -a "$LOG_FILE"
            log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET_ID"
        done
    fi

    if [[ -n "$VPC_ID" ]]; then
        echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
```

```
        log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"
    fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
VPC_NAME="neptune-vpc-$RANDOM_ID"
DB_SUBNET_GROUP="neptune-subnet-group-$RANDOM_ID"
DB_CLUSTER_ID="neptune-cluster-$RANDOM_ID"
DB_INSTANCE_ID="neptune-instance-$RANDOM_ID"
SG_NAME="neptune-sg-$RANDOM_ID"

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "VPC Name: $VPC_NAME" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "DB Cluster ID: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "DB Instance ID: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "Security Group Name: $SG_NAME" | tee -a "$LOG_FILE"

# Step 1: Create VPC
echo "Creating VPC..." | tee -a "$LOG_FILE"
VPC_OUTPUT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=$VPC_NAME},{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-neptune-gs}]" --output json)
check_error "$VPC_OUTPUT" $? "Failed to create VPC"

VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID" | tee -a "$LOG_FILE"

# Enable DNS support for the VPC
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support"
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames"

# Step 2: Create Internet Gateway and attach to VPC
echo "Creating Internet Gateway..." | tee -a "$LOG_FILE"
IGW_OUTPUT=$(aws ec2 create-internet-gateway --tag-specifications
 'ResourceType=internet-gateway,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$IGW_OUTPUT" $? "Failed to create Internet Gateway"

IGW_ID=$(echo "$IGW_OUTPUT" | grep -o '"InternetGatewayId": "[^"]*' | cut -d'"' -
f4)
echo "Internet Gateway created with ID: $IGW_ID" | tee -a "$LOG_FILE"
```

```
log_cmd "aws ec2 attach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"

# Step 3: Create subnets in different AZs
echo "Creating subnets..." | tee -a "$LOG_FILE"

# Get available AZs
AZ_OUTPUT=$(aws ec2 describe-availability-zones --output json)
check_error "$AZ_OUTPUT" $? "Failed to get availability zones"

# Extract first 3 AZ names
AZ1=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -1)
AZ2=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -2
 | tail -1)
AZ3=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -3
 | tail -1)

# Create 3 subnets in different AZs
SUBNET1_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.1.0/24 --availability-zone $AZ1 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET1_OUTPUT" $? "Failed to create subnet 1"
SUBNET1_ID=$(echo "$SUBNET1_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET2_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.2.0/24 --availability-zone $AZ2 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET2_OUTPUT" $? "Failed to create subnet 2"
SUBNET2_ID=$(echo "$SUBNET2_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET3_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.3.0/24 --availability-zone $AZ3 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET3_OUTPUT" $? "Failed to create subnet 3"
SUBNET3_ID=$(echo "$SUBNET3_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET_IDS="$SUBNET1_ID $SUBNET2_ID $SUBNET3_ID"
```

```
echo "Created subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a
 "$LOG_FILE"

# Step 4: Create route table and add route to Internet Gateway
echo "Creating route table..." | tee -a "$LOG_FILE"
ROUTE_TABLE_OUTPUT=$(aws ec2 create-route-table --vpc-id $VPC_ID --tag-
specifications 'ResourceType=route-table,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$ROUTE_TABLE_OUTPUT" $? "Failed to create route table"

ROUTE_TABLE_ID=$(echo "$ROUTE_TABLE_OUTPUT" | grep -o '"RouteTableId": "[^"]*' |
 cut -d'"' -f4)
echo "Route table created with ID: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"

# Add route to Internet Gateway
log_cmd "aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-
cidr-block 0.0.0.0/0 --gateway-id $IGW_ID"

# Associate route table with subnets
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET1_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET2_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET3_ID"

# Step 5: Create security group
echo "Creating security group..." | tee -a "$LOG_FILE"
SG_OUTPUT=$(aws ec2 create-security-group --group-name $SG_NAME --description
 "Security group for Neptune" --vpc-id $VPC_ID --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SG_OUTPUT" $? "Failed to create security group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
echo "Security group created with ID: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"

# Add inbound rule for Neptune port (8182)
# Note: In production, you should restrict this to specific IP ranges
echo "Adding security group rule for Neptune port 8182..." | tee -a "$LOG_FILE"
log_cmd "aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID
 --protocol tcp --port 8182 --cidr 10.0.0.0/16"
```

```
# Step 6: Create DB subnet group
echo "Creating DB subnet group..." | tee -a "$LOG_FILE"
DB_SUBNET_GROUP_OUTPUT=$(aws neptune create-db-subnet-group --db-subnet-group-
name $DB_SUBNET_GROUP --db-subnet-group-description "Subnet group for Neptune" --
subnet-ids $SUBNET1_ID $SUBNET2_ID $SUBNET3_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_SUBNET_GROUP_OUTPUT" $? "Failed to create DB subnet group"
echo "DB subnet group created: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"

# Step 7: Create Neptune DB cluster
echo "Creating Neptune DB cluster..." | tee -a "$LOG_FILE"
DB_CLUSTER_OUTPUT=$(aws neptune create-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --engine neptune --vpc-security-group-ids $SECURITY_GROUP_ID
 --db-subnet-group-name $DB_SUBNET_GROUP --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_CLUSTER_OUTPUT" $? "Failed to create Neptune DB cluster"
echo "Neptune DB cluster created: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"

# Step 8: Create Neptune DB instance
echo "Creating Neptune DB instance..." | tee -a "$LOG_FILE"
DB_INSTANCE_OUTPUT=$(aws neptune create-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --db-instance-class db.r5.large --engine neptune --db-
cluster-identifier $DB_CLUSTER_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_INSTANCE_OUTPUT" $? "Failed to create Neptune DB instance"
echo "Neptune DB instance created: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"

# Step 9: Wait for the DB instance to become available
echo "Waiting for Neptune DB instance to become available..." | tee -a
 "$LOG_FILE"
log_cmd "aws neptune wait db-instance-available --db-instance-identifier
 $DB_INSTANCE_ID"

# Step 10: Get the Neptune endpoint
echo "Getting Neptune endpoint..." | tee -a "$LOG_FILE"
ENDPOINT_OUTPUT=$(aws neptune describe-db-clusters --db-cluster-identifier
 $DB_CLUSTER_ID --output json)
check_error "$ENDPOINT_OUTPUT" $? "Failed to get Neptune endpoint"

NEPTUNE_ENDPOINT=$(echo "$ENDPOINT_OUTPUT" | grep -o '"Endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Neptune endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"

# Step 11: Display information about how to connect to Neptune
```

```
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "NEPTUNE SETUP COMPLETE" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Neptune Endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"
echo "Port: 8182" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To query your Neptune database using Gremlin, you can use curl:" | tee -a
 "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.V().limit(1)\"}' https://
$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To add data to your graph:" | tee -a "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.addV(\\\"person\\\").property(\\\"name
\\\", \\\"Howard\\\")\"}' https://$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Note: You may need to configure your client machine to access the Neptune
 instance within the VPC." | tee -a "$LOG_FILE"
echo "For production use, consider using an EC2 instance in the same VPC or
 setting up a bastion host." | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 12: List all created resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "RESOURCES CREATED" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "VPC: $VPC_ID" | tee -a "$LOG_FILE"
echo "Internet Gateway: $IGW_ID" | tee -a "$LOG_FILE"
echo "Subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"
echo "Route Table: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"
echo "Security Group: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "Neptune DB Cluster: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "Neptune DB Instance: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 13: Ask if user wants to clean up resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
```

```
echo "Do you want to clean up all created resources? (y/n): " | tee -a
 "$LOG_FILE"
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..." | tee -a "$LOG_FILE"

    # Delete DB instance
    echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"

    # Wait for DB instance to be deleted
    echo "Waiting for DB instance to be deleted..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"

    # Delete DB cluster
    echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-cluster --db-cluster-identifier $DB_CLUSTER_ID
 --skip-final-snapshot"

    # Wait for DB cluster to be deleted (no specific wait command for this, so
 we'll sleep)
    echo "Waiting for DB cluster to be deleted..." | tee -a "$LOG_FILE"
    sleep 60

    # Delete DB subnet group
    echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"

    # Delete security group
    echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"

    # Detach and delete internet gateway
    echo "Detaching and deleting internet gateway $IGW_ID..." | tee -a
 "$LOG_FILE"
    log_cmd "aws ec2 detach-internet-gateway --internet-gateway-id $IGW_ID --vpc-
id $VPC_ID"
    log_cmd "aws ec2 delete-internet-gateway --internet-gateway-id $IGW_ID"

    # Delete subnets
```

```
    echo "Deleting subnets..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET1_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET2_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET3_ID"

    # Delete route table
    echo "Deleting route table $ROUTE_TABLE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-route-table --route-table-id $ROUTE_TABLE_ID"

    # Delete VPC
    echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"

    echo "Cleanup complete!" | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can delete them manually later."
 | tee -a "$LOG_FILE"
    echo "See the list of resources above for reference." | tee -a "$LOG_FILE"
fi

echo "Script completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateInternetGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - DeleteDbCluster

- DeleteDbInstance
- DeleteDbSubnetGroup
- DeleteInternetGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeDbClusters
- DetachInternetGateway
- ModifyVpcAttribute
- Wait

Use a API do Amazon Neptune para desenvolver uma função do Lambda que consulta dados de gráficos

O exemplo de código a seguir mostra como usar a API do Neptune para consultar dados de gráficos.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Neptune para criar uma função do Lambda que consulta dados de gráficos na VPC.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Lambda
- Neptune

Exemplos de código para organizações que usam AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Organizations com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Organizations: mais informações sobre o Organizations.
- Referência da API Organizations: detalhes sobre todas as ações disponíveis do Organizations.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para organizações que usam AWS SDKs
  - Ações para organizações usando AWS SDKs
    - Use AttachPolicycom um AWS SDK ou CLI
    - Use CreateAccountcom um AWS SDK ou CLI
    - Use CreateOrganizationcom um AWS SDK ou CLI
    - Use CreateOrganizationalUnitcom um AWS SDK ou CLI
    - Use CreatePolicycom um AWS SDK ou CLI
    - Use DeleteOrganizationcom um AWS SDK ou CLI
    - Use DeleteOrganizationalUnitcom um AWS SDK ou CLI
    - Use DeletePolicycom um AWS SDK ou CLI
    - Use DescribePolicycom um AWS SDK ou CLI
    - Use DetachPolicycom um AWS SDK ou CLI

    - Use ListAccountscom um AWS SDK ou CLI
    - Use ListOrganizationalUnitsForParentcom um AWS SDK ou CLI
    - Use ListPoliciescom um AWS SDK ou CLI
- Cenários para organizações que usam AWS SDKs
  - Permite o AWS Compute Optimizer Recurso de automação para aplicar ações recomendadas
  - Política para permitir a automação em toda a sua organização
  - Política para permitir a automação na sua conta
  - Política para conceder acesso total ao Compute Optimizer Automation para uma conta gerencial de uma organização
  - Política para conceder acesso total à Automação do Compute Optimizer para uso autônomo AWS contas
  - Política para conceder acesso somente leitura ao Compute Optimizer Automation para uma conta gerencial de uma organização
  - Política para conceder acesso somente de leitura à Automação do Compute Optimizer para uso autônomo AWS contas
  - Política para conceder permissões de perfil vinculado ao serviço para o Compute Optimization Automation

Exemplos básicos para organizações que usam AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Organizations com AWS SDKs.

Exemplos

- Ações para organizações usando AWS SDKs
  - Use AttachPolicycom um AWS SDK ou CLI
  - Use CreateAccountcom um AWS SDK ou CLI
  - Use CreateOrganizationcom um AWS SDK ou CLI
  - Use CreateOrganizationalUnitcom um AWS SDK ou CLI
  - Use CreatePolicycom um AWS SDK ou CLI
  - Use DeleteOrganizationcom um AWS SDK ou CLI
  - Use DeleteOrganizationalUnitcom um AWS SDK ou CLI

- Use DeletePolicycom um AWS SDK ou CLI
- Use DescribePolicycom um AWS SDK ou CLI
- Use DetachPolicycom um AWS SDK ou CLI
- Use ListAccountscom um AWS SDK ou CLI
- Use ListOrganizationalUnitsForParentcom um AWS SDK ou CLI
- Use ListPoliciescom um AWS SDK ou CLI

Ações para organizações usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais da Organizations com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Organizations e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para organizações que usam AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Organizations.

Exemplos

- Use AttachPolicycom um AWS SDK ou CLI
- Use CreateAccountcom um AWS SDK ou CLI
- Use CreateOrganizationcom um AWS SDK ou CLI
- Use CreateOrganizationalUnitcom um AWS SDK ou CLI
- Use CreatePolicycom um AWS SDK ou CLI
- Use DeleteOrganizationcom um AWS SDK ou CLI
- Use DeleteOrganizationalUnitcom um AWS SDK ou CLI
- Use DeletePolicycom um AWS SDK ou CLI
- Use DescribePolicycom um AWS SDK ou CLI
- Use DetachPolicycom um AWS SDK ou CLI
- Use ListAccountscom um AWS SDK ou CLI
- Use ListOrganizationalUnitsForParentcom um AWS SDK ou CLI
- Use ListPoliciescom um AWS SDK ou CLI

Use **`AttachPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AttachPolicy`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to attach an AWS Organizations policy to an organization,
/// an organizational unit, or an account.
/// </summary>
public class AttachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then calls the
    /// AttachPolicyAsync method to attach the policy to the root
    /// organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var policyId = "p-00000000";
        var targetId = "r-0000";

        var request = new AttachPolicyRequest
        {
            PolicyId = policyId,
            TargetId = targetId,
        };

        var response = await client.AttachPolicyAsync(request);
```

```
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully attached Policy ID {policyId} to
 Target ID: {targetId}.");
            }
            else
            {
                Console.WriteLine("Was not successful in attaching the policy.");
            }
        }
    }
```

- Para obter detalhes da API, consulte AttachPolicya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como anexar uma política a uma raiz, unidade operacional ou conta

Exemplo 1

O seguinte exemplo mostra com anexar uma política de controle de serviços (SCP) a uma unidade operacional (OU):

```
aws organizations attach-policy
                --policy-id p-examplepolicyid111
                --target-id ou-examplerootid111-exampleouid111
```

Exemplo 2

O seguinte exemplo mostra como anexar uma política de controle de serviços a uma conta:

```
aws organizations attach-policy
                --policy-id p-examplepolicyid111
                --target-id 333333333333
```

- Para obter detalhes da API, consulte AttachPolicyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def attach_policy(policy_id, target_id, orgs_client):
    """
    Attaches a policy to a target. The target is an organization root, account,
 or
    organizational unit.

    :param policy_id: The ID of the policy to attach.
    :param target_id: The ID of the resources to attach the policy to.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.attach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Attached policy %s to target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't attach policy %s to target %s.", policy_id, target_id
        )
        raise
```

- Para obter detalhes da API, consulte a AttachPolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_org->attachpolicy(
      iv_policyid = iv_policy_id
      iv_targetid = iv_target_id ).
    MESSAGE 'Policy attached to target.' TYPE 'I'.
  CATCH /aws1/cx_orgaccessdeniedex.
    MESSAGE 'You do not have permission to attach the policy.' TYPE 'E'.
  CATCH /aws1/cx_orgpolicynotfoundex.
    MESSAGE 'The specified policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgtargetnotfoundex.
    MESSAGE 'The specified target does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgduplicateplyatta00.
    MESSAGE 'The policy is already attached to the target.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a AttachPolicyreferência da API AWS SDK for SAP ABAP.

Use **`CreateAccount`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateAccount`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Creates a new AWS Organizations account.
/// </summary>
public class CreateAccount
{
    /// <summary>
    /// Initializes an Organizations client object and uses it to create
    /// the new account with the name specified in accountName.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var accountName = "ExampleAccount";
        var email = "someone@example.com";

        var request = new CreateAccountRequest
        {
            AccountName = accountName,
            Email = email,
        };

        var response = await client.CreateAccountAsync(request);
        var status = response.CreateAccountStatus;

        Console.WriteLine($"The staus of {status.AccountName} is
{status.State}.");
    }
```

```
    }
```

- Para obter detalhes da API, consulte CreateAccounta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar uma conta de membro que automaticamente faça parte da organização

O exemplo a seguir mostra como criar uma conta de membro em uma organização. A conta de membro é configurada com o nome Production Account e o endereço de e-mail susan@example.com. Organizations cria automaticamente uma função do IAM usando o nome padrão de OrganizationAccountAccessRole porque o parâmetro roleName não está especificado. Além disso, a configuração que permite que usuários ou funções do IAM com permissões suficientes acessem os dados de faturamento da conta é definida com o valor padrão de ALLOW porque o IamUserAccessToBilling parâmetro não foi especificado. Organizations envia automaticamente a Susan um e-mail de “Bem-vindo a AWS”:

```
aws organizations create-account --email susan@example.com --account-
name "Production Account"
```

A saída inclui um objeto de solicitação que mostra que o status agora é `IN_PROGRESS`:

```
{
        "CreateAccountStatus": {
                "State": "IN_PROGRESS",
                "Id": "car-examplecreateaccountrequestid111"
        }
}
```

Você pode consultar o status atual da solicitação posteriormente. Para fazer isso, forneça o valor de resposta do ID ao comando describe-create-account-status como o valor do parâmetro create-account-request-id.

Para obter mais informações, consulte Criando uma AWS conta em sua organização no Guia do Usuário do AWS Organizations.

- Para obter detalhes da API, consulte CreateAccountem Referência de AWS CLI Comandos.

Use **`CreateOrganization`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateOrganization`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Creates an organization in AWS Organizations.
/// </summary>
public class CreateOrganization
{
    /// <summary>
    /// Creates an Organizations client object and then uses it to create
    /// a new organization with the default user as the administrator, and
    /// then displays information about the new organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.CreateOrganizationAsync(new
CreateOrganizationRequest
        {
            FeatureSet = "ALL",
        });

        Organization newOrg = response.Organization;
```

```
            Console.WriteLine($"Organization: {newOrg.Id} Main Accoount:
 {newOrg.MasterAccountId}");
        }
    }
```

- Para obter detalhes da API, consulte CreateOrganizationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: como criar uma organização

Bill quer criar uma organização usando as credenciais da conta 111111111111. O exemplo a seguir mostra que a conta se torna a conta principal na nova organização. Como ele não especificou um conjunto de recursos, a nova organização usa como padrão todos os recursos habilitados e as políticas de controle de serviços são habilitadas na raiz.

```
aws organizations create-organization
```

A saída inclui um objeto de organização com detalhes sobre a nova organização:

```
{
        "Organization": {
                "AvailablePolicyTypes": [
                        {
                                "Status": "ENABLED",
                                "Type": "SERVICE_CONTROL_POLICY"
                        }
                ],
                "MasterAccountId": "111111111111",
                "MasterAccountArn": "arn:aws:organizations::111111111111:account/
o-exampleorgid/111111111111",
                "MasterAccountEmail": "bill@example.com",
                "FeatureSet": "ALL",
                "Id": "o-exampleorgid",
                "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid"
```

```
        }
}
```

Exemplo 2: como criar uma organização apenas com os recursos de faturamento consolidados

O seguinte exemplo cria uma organização compatível apenas com os recursos de faturamento consolidados:

```
aws organizations create-organization --feature-set CONSOLIDATED_BILLING
```

A saída inclui um objeto de organização com detalhes sobre a nova organização:

```
{
        "Organization": {
                "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid",
                "AvailablePolicyTypes": [],
                "Id": "o-exampleorgid",
                "MasterAccountArn": "arn:aws:organizations::111111111111:account/
o-exampleorgid/111111111111",
                "MasterAccountEmail": "bill@example.com",
                "MasterAccountId": "111111111111",
                "FeatureSet": "CONSOLIDATED_BILLING"
        }
}
```

Para obter informações, consulte Criar uma organização no Guia do usuário do AWS Organizations.

- Para obter detalhes da API, consulte CreateOrganizationem Referência de AWS CLI Comandos.

Use **`CreateOrganizationalUnit`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateOrganizationalUnit`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Creates a new organizational unit in AWS Organizations.
/// </summary>
public class CreateOrganizationalUnit
{
    /// <summary>
    /// Initializes an Organizations client object and then uses it to call
    /// the CreateOrganizationalUnit method. If the call succeeds, it
    /// displays information about the new organizational unit.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitName = "ProductDevelopmentUnit";

        var request = new CreateOrganizationalUnitRequest
        {
            Name = orgUnitName,
            ParentId = "r-0000",
        };

        var response = await client.CreateOrganizationalUnitAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
```

```
                Console.WriteLine($"Successfully created organizational unit:
 {orgUnitName}.");
                Console.WriteLine($"Organizational unit {orgUnitName} Details");
                Console.WriteLine($"ARN: {response.OrganizationalUnit.Arn} Id:
 {response.OrganizationalUnit.Id}");
            }
            else
            {
                Console.WriteLine("Could not create new organizational unit.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateOrganizationalUnita Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar uma unidade organizacional em uma unidade organizacional raiz ou pai

O seguinte exemplo mostra como criar uma UO chamada AccountingOU:

```
aws organizations create-organizational-unit --parent-id r-examplerootid111 --
name AccountingOU
```

A saída inclui um objeto organizationalUnit que contém detalhes sobre a nova UO:

```
{
        "OrganizationalUnit": {
                "Id": "ou-examplerootid111-exampleouid111",
                "Arn": "arn:aws:organizations::111111111111:ou/o-exampleorgid/ou-
examplerootid111-exampleouid111",
                "Name": "AccountingOU",
                "Path": "o-exampleorgid/r-examplerootid111/ou-examplerootid111-
exampleouid111/"
        }
}
```

- Para obter detalhes da API, consulte CreateOrganizationalUnitem Referência de AWS CLI Comandos.

Use **`CreatePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreatePolicy`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Creates a new AWS Organizations Policy.
/// </summary>
public class CreatePolicy
{
    /// <summary>
    /// Initializes the AWS Organizations client object, uses it to
    /// create a new Organizations Policy, and then displays information
    /// about the newly created Policy.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var policyContent = "{" +
            "   \"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:*\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
```

```
            "}";

            try
            {
                var response = await client.CreatePolicyAsync(new
CreatePolicyRequest
                {
                    Content = policyContent,
                    Description = "Enables admins of attached accounts to
delegate all Amazon S3 permissions",
                    Name = "AllowAllS3Actions",
                    Type = "SERVICE_CONTROL_POLICY",
                });

                Policy policy = response.Policy;
                Console.WriteLine($"{policy.PolicySummary.Name} has the following
content: {policy.Content}");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte CreatePolicya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: como criar uma política com um arquivo de origem de texto na política JSON

O exemplo a seguir mostra como criar uma política de controle de serviço (SCP) chamada `AllowAllS3Actions`. O conteúdo da política provém de um arquivo chamado `policy.json` presente no computador local.

```
aws organizations create-policy --content file://policy.json --
name AllowAllS3Actions, --type SERVICE_CONTROL_POLICY --description "Allows
 delegation of all S3 actions"
```

A saída inclui um objeto de política com detalhes sobre a nova política:

```
{
        "Policy": {
                "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\":\"Allow\",\"Action\":[\"s3:*\"],\"Resource\":[\"*\"]}]}",
                "PolicySummary": {
                        "Arn": "arn:aws:organizations::o-exampleorgid:policy/
service_control_policy/p-examplepolicyid111",
                        "Description": "Allows delegation of all S3 actions",
                        "Name": "AllowAllS3Actions",
                        "Type":"SERVICE_CONTROL_POLICY"
                }
        }
}
```

Exemplo 2: como criar uma política tendo uma política JSON como parâmetro

O exemplo a seguir mostra como criar a mesma SCP, mas, desta vez, incorporando o conteúdo da política como uma string JSON no parâmetro. A string deve ser recuada com barras invertidas antes das aspas duplas para garantir que ela seja tratada como literal no parâmetro (que está entre aspas duplas):

```
aws organizations create-policy --content "{\"Version\":\"2012-10-17\",
\"Statement\":[{\"Effect\":\"Allow\",\"Action\":[\"s3:*\"],\"Resource
\":[\"*\"]}]}" --name AllowAllS3Actions --type SERVICE_CONTROL_POLICY --
description "Allows delegation of all S3 actions"
```

Para obter mais informações sobre como criar e usar políticas em sua organização, consulte Gerenciamento de políticas organizacionais no Guia do usuário do AWS Organizations.

- Para obter detalhes da API, consulte CreatePolicyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_policy(name, description, content, policy_type, orgs_client):
    """
    Creates a policy.

    :param name: The name of the policy.
    :param description: The description of the policy.
    :param content: The policy content as a dict. This is converted to JSON
 before
                        it is sent to AWS. The specific format depends on the policy
 type.
    :param policy_type: The type of the policy.
    :param orgs_client: The Boto3 Organizations client.
    :return: The newly created policy.
    """
    try:
        response = orgs_client.create_policy(
            Name=name,
            Description=description,
            Content=json.dumps(content),
            Type=policy_type,
        )
        policy = response["Policy"]
        logger.info("Created policy %s.", name)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- Para obter detalhes da API, consulte a CreatePolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_org->createpolicy(       " oo_result is returned for
testing purposes. "
         iv_name        = iv_policy_name
         iv_description = iv_policy_description
         iv_content     = iv_policy_content
         iv_type        = iv_policy_type ).
       MESSAGE 'Policy created.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to create a policy.' TYPE 'E'.
     CATCH /aws1/cx_orgduplicatepolicyex.
       MESSAGE 'A policy with this name already exists.' TYPE 'E'.
     CATCH /aws1/cx_orgmalformedplydocex.
       MESSAGE 'The policy content is malformed.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreatePolicyreferência da API AWS SDK for SAP ABAP.

Use **`DeleteOrganization`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteOrganization`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to delete an existing organization using the AWS
/// Organizations Service.
/// </summary>
public class DeleteOrganization
{
    /// <summary>
    /// Initializes the Organizations client and then calls
    /// DeleteOrganizationAsync to delete the organization.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.DeleteOrganizationAsync(new
DeleteOrganizationRequest());

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Successfully deleted organization.");
        }
        else
        {
            Console.WriteLine("Could not delete organization.");
        }
    }
```

```
    }
```

- Para obter detalhes da API, consulte DeleteOrganizationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir uma organização

O exemplo a seguir mostra como excluir uma organização. Você deve ser administrador da conta principal na organização para poder realizar essa operação. O exemplo pressupõe que você já removeu todas as contas de membro, UOs e políticas da organização:

```
aws organizations delete-organization
```

- Para obter detalhes da API, consulte DeleteOrganizationem Referência de AWS CLI Comandos.

Use **`DeleteOrganizationalUnit`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteOrganizationalUnit`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
```

```
    using Amazon.Organizations.Model;

    /// <summary>
    /// Shows how to delete an existing AWS Organizations organizational unit.
    /// </summary>
    public class DeleteOrganizationalUnit
    {
        /// <summary>
        /// Initializes the Organizations client object and calls
        /// DeleteOrganizationalUnitAsync to delete the organizational unit
        /// with the selected ID.
        /// </summary>
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var orgUnitId = "ou-0000-00000000";

            var request = new DeleteOrganizationalUnitRequest
            {
                OrganizationalUnitId = orgUnitId,
            };

            var response = await client.DeleteOrganizationalUnitAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted the organizational unit
with ID: {orgUnitId}.");
            }
            else
            {
                Console.WriteLine($"Could not delete the organizational unit with
ID: {orgUnitId}.");
            }
        }
    }
```

- Para obter detalhes da API, consulte DeleteOrganizationalUnita Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir uma UO

O exemplo a seguir mostra como excluir uma UO. O exemplo pressupõe que você já removeu todas as contas e outras UOs da UO:

```
aws organizations delete-organizational-unit --organizational-unit-id ou-
examplerootid111-exampleouid111
```

- Para obter detalhes da API, consulte DeleteOrganizationalUnitem Referência de AWS CLI Comandos.

Use **`DeletePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeletePolicy`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Deletes an existing AWS Organizations policy.
/// </summary>
public class DeletePolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
```

```
        /// delete the policy with the specified policyId.
        /// </summary>
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var policyId = "p-00000000";

            var request = new DeletePolicyRequest
            {
                PolicyId = policyId,
            };

            var response = await client.DeletePolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted Policy: {policyId}.");
            }
            else
            {
                Console.WriteLine($"Could not delete Policy: {policyId}.");
            }
        }
    }
```

- Para obter detalhes da API, consulte DeletePolicya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir uma política

O exemplo a seguir mostra como excluir uma política de uma organização. O exemplo pressupõe que você já separou a política de todas as entidades:

```
aws organizations delete-policy --policy-id p-examplepolicyid111
```

- Para obter detalhes da API, consulte DeletePolicyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_policy(policy_id, orgs_client):
    """
    Deletes a policy.

    :param policy_id: The ID of the policy to delete.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.delete_policy(PolicyId=policy_id)
        logger.info("Deleted policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_id)
        raise
```

- Para obter detalhes da API, consulte a DeletePolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_org->deletepolicy(
      iv_policyid = iv_policy_id ).
    MESSAGE 'Policy deleted.' TYPE 'I'.
  CATCH /aws1/cx_orgaccessdeniedex.
    MESSAGE 'You do not have permission to delete the policy.' TYPE 'E'.
  CATCH /aws1/cx_orgpolicynotfoundex.
    MESSAGE 'The specified policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgpolicyinuseex.
    MESSAGE 'The policy is still attached to one or more targets.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeletePolicyreferência da API AWS SDK for SAP ABAP.

Use **`DescribePolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribePolicy`.

CLI

AWS CLI

Como obter informações sobre uma política

O seguinte exemplo mostra como solicitar informações sobre uma política:

```
aws organizations describe-policy --policy-id p-examplepolicyid111
```

A saída inclui um objeto de política que contém detalhes sobre a política:

```
{
        "Policy": {
                "Content": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement
\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": \"*\",\n
 \"Resource\": \"*\"\n    }\n  ]\n}",
                "PolicySummary": {
                        "Arn": "arn:aws:organizations::111111111111:policy/o-
exampleorgid/service_control_policy/p-examplepolicyid111",
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Id": "p-examplepolicyid111",
                        "AwsManaged": false,
                        "Name": "AllowAllS3Actions",
                        "Description": "Enables admins to delegate S3
 permissions"
                }
        }
}
```

- Para obter detalhes da API, consulte DescribePolicyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def describe_policy(policy_id, orgs_client):
    """
    Describes a policy.

    :param policy_id: The ID of the policy to describe.
    :param orgs_client: The Boto3 Organizations client.
    :return: The description of the policy.
    """
    try:
        response = orgs_client.describe_policy(PolicyId=policy_id)
        policy = response["Policy"]
```

```
        logger.info("Got policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't get policy %s.", policy_id)
        raise
    else:
        return policy
```

- Para obter detalhes da API, consulte a DescribePolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_org->describepolicy(     " oo_result is returned for
testing purposes. "
          iv_policyid = iv_policy_id ).
        DATA(lo_policy) = oo_result->get_policy( ).
        MESSAGE 'Retrieved policy details.' TYPE 'I'.
      CATCH /aws1/cx_orgaccessdeniedex.
        MESSAGE 'You do not have permission to describe the policy.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicynotfoundex.
        MESSAGE 'The specified policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribePolicyreferência da API AWS SDK for SAP ABAP.

Use **`DetachPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetachPolicy`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to detach a policy from an AWS Organizations organization,
/// organizational unit, or account.
/// </summary>
public class DetachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and uses it to call
    /// DetachPolicyAsync to detach the policy.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";
        var targetId = "r-0000";

        var request = new DetachPolicyRequest
        {
            PolicyId = policyId,
            TargetId = targetId,
        };
```

```
            var response = await client.DetachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully detached policy with Policy Id:
{policyId}.");
            }
            else
            {
                Console.WriteLine("Could not detach the policy.");
            }
        }
    }
```

- Para obter detalhes da API, consulte DetachPolicya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como separar uma política de uma raiz, UO ou conta

O seguinte exemplo mostra como separar uma política de uma UO:

```
aws organizations  detach-policy  --target-id ou-examplerootid111-exampleouid111
 --policy-id p-examplepolicyid111
```

- Para obter detalhes da API, consulte DetachPolicyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def detach_policy(policy_id, target_id, orgs_client):
    """
    Detaches a policy from a target.

    :param policy_id: The ID of the policy to detach.
    :param target_id: The ID of the resource where the policy is currently
 attached.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.detach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Detached policy %s from target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from target %s.", policy_id, target_id
        )
        raise
```

- Para obter detalhes da API, consulte a DetachPolicyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_org->detachpolicy(
          iv_policyid = iv_policy_id
          iv_targetid = iv_target_id ).
        MESSAGE 'Policy detached from target.' TYPE 'I'.
      CATCH /aws1/cx_orgaccessdeniedex.
        MESSAGE 'You do not have permission to detach the policy.' TYPE 'E'.
```

```
      CATCH /aws1/cx_orgpolicynotfoundex.
        MESSAGE 'The specified policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_orgtargetnotfoundex.
        MESSAGE 'The specified target does not exist.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicynotattex.
        MESSAGE 'The policy is not attached to the target.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetachPolicyreferência da API AWS SDK for SAP ABAP.

Use **`ListAccounts`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListAccounts`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Uses the AWS Organizations service to list the accounts associated
/// with the default account.
/// </summary>
public class ListAccounts
{
    /// <summary>
    /// Creates the Organizations client and then calls its
    /// ListAccountsAsync method.
    /// </summary>
```

```
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var request = new ListAccountsRequest
            {
                MaxResults = 5,
            };

            var response = new ListAccountsResponse();
            try
            {
                do
                {
                    response = await client.ListAccountsAsync(request);
                    response.Accounts.ForEach(a => DisplayAccounts(a));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about an Organizations account.
        /// </summary>
        /// <param name="account">An Organizations account for which to display
        /// information on the console.</param>
        private static void DisplayAccounts(Account account)
        {
            string accountInfo = $"{account.Id}
{account.Name}\t{account.Status}";

            Console.WriteLine(accountInfo);
        }
    }
```

- Para obter detalhes da API, consulte ListAccountsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar uma lista de todas as contas de uma organização

O seguinte exemplo mostra como solicitar uma lista das contas de uma organização:

```
aws organizations list-accounts
```

A saída inclui uma lista de objetos de resumo da conta.

```
{
        "Accounts": [
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/111111111111",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481830215.45,
                        "Id": "111111111111",
                        "Name": "Master Account",
                        "Email": "bill@example.com",
                        "Paths": [
                                "o-exampleorgid/r-examplerootid111/111111111111/"
                        ],
                        "State": "ACTIVE",
                        "Status": "ACTIVE"
                },
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/222222222222",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835741.044,
                        "Id": "222222222222",
                        "Name": "Production Account",
                        "Email": "alice@example.com",
                        "Paths": [
```

```
                                "o-exampleorgid/r-examplerootid111/111111111111/"
                        ],
                        "State": "ACTIVE",
                        "Status": "ACTIVE"
                },
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/333333333333",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835795.536,
                        "Id": "333333333333",
                        "Name": "Development Account",
                        "Email": "juan@example.com",
                        "Paths": [
                                "o-exampleorgid/r-examplerootid111/111111111111/"
                        ],
                        "State": "ACTIVE",
                        "Status": "ACTIVE"
                },
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/444444444444",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835812.143,
                        "Id": "444444444444",
                        "Name": "Test Account",
                        "Email": "anika@example.com",
                        "Paths": [
                                "o-exampleorgid/r-examplerootid111/111111111111/"
                        ],
                        "State": "ACTIVE",
                        "Status": "ACTIVE"
                }
        ]
}
```

- Para obter detalhes da API, consulte ListAccountsem Referência de AWS CLI Comandos.

Use **`ListOrganizationalUnitsForParent`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListOrganizationalUnitsForParent`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Lists the AWS Organizations organizational units that belong to an
/// organization.
/// </summary>
public class ListOrganizationalUnitsForParent
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// call the ListOrganizationalUnitsForParentAsync method to retrieve
    /// the list of organizational units.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var parentId = "r-0000";

        var request = new ListOrganizationalUnitsForParentRequest
        {
            ParentId = parentId,
            MaxResults = 5,
        };

        var response = new ListOrganizationalUnitsForParentResponse();
        try
        {
```

```
                do
                {
                    response = await
client.ListOrganizationalUnitsForParentAsync(request);
                    response.OrganizationalUnits.ForEach(u =>
DisplayOrganizationalUnit(u));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about an Organizations organizational unit.
        /// </summary>
        /// <param name="unit">The OrganizationalUnit for which to display
        /// information.</param>
        public static void DisplayOrganizationalUnit(OrganizationalUnit unit)
        {
            string accountInfo = $"{unit.Id} {unit.Name}\t{unit.Arn}";

            Console.WriteLine(accountInfo);
        }
    }
```

- Para obter detalhes da API, consulte ListOrganizationalUnitsForParent a Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar uma lista das OUs em uma OU principal ou raiz

O exemplo a seguir mostra como obter uma lista de OUs em uma raiz especificada:

```
aws organizations list-organizational-units-for-parent --parent-id r-
examplerootid111
```

A saída mostra que a raiz especificada contém duas OUs e mostra os detalhes de cada uma:

```
{
        "OrganizationalUnits": [
                {
                        "Name": "AccountingDepartment",
                        "Arn": "arn:aws:organizations::o-exampleorgid:ou/r-
examplerootid111/ou-examplerootid111-exampleouid111",
                        "Id": "ou-examplerootid111-exampleouid111",
                        "Path": "o-exampleorgid/r-examplerootid111/ou-
examplerootid111-exampleouid111/"
                },
                {
                        "Name": "ProductionDepartment",
                        "Arn": "arn:aws:organizations::o-exampleorgid:ou/r-
examplerootid111/ou-examplerootid111-exampleouid222",
                        "Id": "ou-examplerootid111-exampleouid222",
                        "Path": "o-exampleorgid/r-examplerootid111/ou-
examplerootid111-exampleouid222/"
                }
        ]
}
```

- Para obter detalhes da API, consulte ListOrganizationalUnitsForParentem Referência de AWS CLI Comandos.

Use **`ListPolicies`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListPolicies`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

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

/// <summary>
/// Shows how to list the AWS Organizations policies associated with an
/// organization.
/// </summary>
public class ListPolicies
{
    /// <summary>
    /// Initializes an Organizations client object, and then calls its
    /// ListPoliciesAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        // The value for the Filter parameter is required and must must be
        // one of the following:
        //     AISERVICES_OPT_OUT_POLICY
        //     BACKUP_POLICY
        //     SERVICE_CONTROL_POLICY
        //     TAG_POLICY
        var request = new ListPoliciesRequest
        {
            Filter = "SERVICE_CONTROL_POLICY",
            MaxResults = 5,
        };
```

```
            var response = new ListPoliciesResponse();
            try
            {
                do
                {
                    response = await client.ListPoliciesAsync(request);
                    response.Policies.ForEach(p => DisplayPolicies(p));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about the Organizations policies associated
        /// with an organization.
        /// </summary>
        /// <param name="policy">An Organizations policy summary to display
        /// information on the console.</param>
        private static void DisplayPolicies(PolicySummary policy)
        {
            string policyInfo = $"{policy.Id}
{policy.Name}\t{policy.Description}";

            Console.WriteLine(policyInfo);
        }
    }
```

- Para obter detalhes da API, consulte ListPoliciesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar uma lista de todas as políticas de um determinado tipo de uma organização

O seguinte exemplo mostra como obter uma lista de SCPs, conforme especificado pelo parâmetro do filtro:

```
aws organizations list-policies --filter SERVICE_CONTROL_POLICY
```

A saída inclui uma lista de políticas com informações resumidas:

```
{
        "Policies": [
                {
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Name": "AllowAllS3Actions",
                        "AwsManaged": false,
                        "Id": "p-examplepolicyid111",
                        "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid111",
                        "Description": "Enables account admins to delegate
 permissions for any S3 actions to users and roles in their accounts."
                },
                {
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Name": "AllowAllEC2Actions",
                        "AwsManaged": false,
                        "Id": "p-examplepolicyid222",
                        "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid222",
                        "Description": "Enables account admins to delegate
 permissions for any EC2 actions to users and roles in their accounts."
                },
                {
                        "AwsManaged": true,
                        "Description": "Allows access to every operation",
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Id": "p-FullAWSAccess",
                        "Arn": "arn:aws:organizations::aws:policy/
service_control_policy/p-FullAWSAccess",
                        "Name": "FullAWSAccess"
```

```
                }
        ]
}
```

- Para obter detalhes da API, consulte ListPoliciesem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_policies(policy_filter, orgs_client):
    """
    Lists the policies for the account, limited to the specified filter.

    :param policy_filter: The kind of policies to return.
    :param orgs_client: The Boto3 Organizations client.
    :return: The list of policies found.
    """
    try:
        response = orgs_client.list_policies(Filter=policy_filter)
        policies = response["Policies"]
        logger.info("Found %s %s policies.", len(policies), policy_filter)
    except ClientError:
        logger.exception("Couldn't get %s policies.", policy_filter)
        raise
    else:
        return policies
```

- Para obter detalhes da API, consulte a ListPoliciesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_org->listpolicies(       " oo_result is returned for
testing purposes. "
         iv_filter = iv_filter ).
       DATA(lt_policies) = oo_result->get_policies( ).
       MESSAGE 'Retrieved list of policies.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to list policies.' TYPE 'E'.
     CATCH /aws1/cx_orgawsorgsnotinuseex.
       MESSAGE 'Your account is not a member of an organization.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListPoliciesreferência da API AWS SDK for SAP ABAP.

Cenários para organizações que usam AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns em Organizations with AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções dentro do Organizations ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Permite o AWS Compute Optimizer Recurso de automação para aplicar ações recomendadas

- Política para permitir a automação em toda a sua organização
- Política para permitir a automação na sua conta
- Política para conceder acesso total ao Compute Optimizer Automation para uma conta gerencial de uma organização
- Política para conceder acesso total à Automação do Compute Optimizer para uso autônomo AWS contas
- Política para conceder acesso somente leitura ao Compute Optimizer Automation para uma conta gerencial de uma organização
- Política para conceder acesso somente de leitura à Automação do Compute Optimizer para uso autônomo AWS contas
- Política para conceder permissões de perfil vinculado ao serviço para o Compute Optimization Automation

Permite o AWS Compute Optimizer Recurso de automação para aplicar ações recomendadas

O exemplo de código a seguir mostra como: Essa política baseada em permissões permite que o recurso de AWS Compute Optimizer automação aplique as ações recomendadas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "aco-automation.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Política para permitir a automação em toda a sua organização

O exemplo de código a seguir mostra como esta política baseada em permissões permite a automação em toda a sua organização

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:AssociateAccounts",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:DisassociateAccounts",
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "aco-automation:ListAccounts",
            "Resource": "*"
        }
    ]
}
```

Política para permitir a automação na sua conta

O exemplo de código a seguir mostra como esta política baseada em permissões permite a automação na sua conta

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        }
    ]
}
```

Política para conceder acesso total ao Compute Optimizer Automation para uma conta gerencial de uma organização

O exemplo de código a seguir mostra como esta política baseada em permissões concede acesso total à automação do Compute Optimizer Automation para uma conta gerencial de uma organização

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
               "ec2:DescribeVolumes",
               "organizations:ListAccounts",
               "organizations:DescribeOrganization",
               "organizations:DescribeAccount",
               "organizations:EnableAWSServiceAccess",
               "organizations:ListDelegatedAdministrators",
               "organizations:RegisterDelegatedAdministrator",
               "organizations:DeregisterDelegatedAdministrator"
            ],
            "Resource": "*"
        }
    ]
}
```

Política para conceder acesso total à Automação do Compute Optimizer para uso autônomo AWS contas

O exemplo de código a seguir mostra como essa política baseada em permissões concede acesso total à Automação do Compute Optimizer para contas autônomas. AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
            "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

Política para conceder acesso somente leitura ao Compute Optimizer Automation para uma conta gerencial de uma organização

O exemplo de código a seguir mostra como esta política baseada em permissões concede acesso somente leitura ao Compute Optimizer Automation para uma conta gerencial de uma organização

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:GetEnrollmentConfiguration",
               "aco-automation:GetAutomationEvent",
               "aco-automation:GetAutomationRule",
               "aco-automation:ListAccounts",
               "aco-automation:ListAutomationEvents",
               "aco-automation:ListAutomationEventSteps",
               "aco-automation:ListAutomationEventSummaries",
               "aco-automation:ListAutomationRules",
               "aco-automation:ListAutomationRulePreview",
               "aco-automation:ListAutomationRulePreviewSummaries",
```

```
                "aco-automation:ListRecommendedActions",
                "aco-automation:ListRecommendedActionSummaries",
                "aco-automation:ListTagsForResource",
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

Política para conceder acesso somente de leitura à Automação do Compute Optimizer para uso autônomo AWS contas

O exemplo de código a seguir mostra como: Essa política baseada em permissões concede acesso somente de leitura à Automação do Compute Optimizer para contas autônomas. AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aco-automation:GetEnrollmentConfiguration",
                "aco-automation:GetAutomationEvent",
                "aco-automation:GetAutomationRule",
                "aco-automation:ListAutomationEvents",
                "aco-automation:ListAutomationEventSteps",
                "aco-automation:ListAutomationEventSummaries",
                "aco-automation:ListAutomationRules",
                "aco-automation:ListAutomationRulePreview",
                "aco-automation:ListAutomationRulePreviewSummaries",
                "aco-automation:ListRecommendedActions",
                "aco-automation:ListRecommendedActionSummaries",
                "aco-automation:ListTagsForResource",
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Política para conceder permissões de perfil vinculado ao serviço para o Compute Optimization Automation

O exemplo de código a seguir mostra como esta política baseada em permissões concede permissões de perfil vinculado ao serviço para o Compute Optimization Automation

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        }
    ]
}
```

Exemplos de código para o Partner Central usando AWS SDKs

Os exemplos de código a seguir mostram como usar o AWS Partner Central com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Conceitos básicos da Central de Parceiros: mais informações sobre a Central de Parceiros.
- Referência de API da Central de Parceiros: detalhes sobre todas as ações disponíveis da Central de Parceiros.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de uso do Partner Central AWS SDKs
  - Ações para o Partner Central usando AWS SDKs
    - Use AssignOpportunitycom um AWS SDK
    - Use AssociateOpportunitycom um AWS SDK
    - Use CreateOpportunitycom um AWS SDK
    - Use DisassociateOpportunitycom um AWS SDK
    - Use GetAwsOpportunitySummarycom um AWS SDK
    - Use GetEngagementInvitationcom um AWS SDK
    - Use GetOpportunitycom um AWS SDK
    - Use ListEngagementInvitationscom um AWS SDK
    - Use ListOpportunitiescom um AWS SDK
    - Use ListSolutionscom um AWS SDK
    - Use RejectEngagementInvitationcom um AWS SDK
    - Use StartEngagementByAcceptingInvitationTaskcom um AWS SDK
    - Use StartEngagementFromOpportunityTaskcom um AWS SDK
    - Use UpdateOpportunitycom um AWS SDK
- Cenários para o uso do Partner Central AWS SDKs
  - Atualizar a entidade associada de uma oportunidade

Exemplos básicos de uso do Partner Central AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do AWS Partner Central com AWS SDKs.

Exemplos

- Ações para o Partner Central usando AWS SDKs
  - Use AssignOpportunitycom um AWS SDK
  - Use AssociateOpportunitycom um AWS SDK
  - Use CreateOpportunitycom um AWS SDK
  - Use DisassociateOpportunitycom um AWS SDK
  - Use GetAwsOpportunitySummarycom um AWS SDK
  - Use GetEngagementInvitationcom um AWS SDK
  - Use GetOpportunitycom um AWS SDK
  - Use ListEngagementInvitationscom um AWS SDK
  - Use ListOpportunitiescom um AWS SDK
  - Use ListSolutionscom um AWS SDK
  - Use RejectEngagementInvitationcom um AWS SDK
  - Use StartEngagementByAcceptingInvitationTaskcom um AWS SDK
  - Use StartEngagementFromOpportunityTaskcom um AWS SDK
  - Use UpdateOpportunitycom um AWS SDK

Ações para o Partner Central usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Partner Central com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API da Central de Parceiros e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Partner Central AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API da Central de Parceiros da AWS.

Exemplos

- Use AssignOpportunitycom um AWS SDK
- Use AssociateOpportunitycom um AWS SDK
- Use CreateOpportunitycom um AWS SDK
- Use DisassociateOpportunitycom um AWS SDK
- Use GetAwsOpportunitySummarycom um AWS SDK
- Use GetEngagementInvitationcom um AWS SDK
- Use GetOpportunitycom um AWS SDK
- Use ListEngagementInvitationscom um AWS SDK
- Use ListOpportunitiescom um AWS SDK
- Use ListSolutionscom um AWS SDK
- Use RejectEngagementInvitationcom um AWS SDK
- Use StartEngagementByAcceptingInvitationTaskcom um AWS SDK
- Use StartEngagementFromOpportunityTaskcom um AWS SDK
- Use UpdateOpportunitycom um AWS SDK

Use **`AssignOpportunity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `AssignOpportunity`.

Java

SDK para Java 2.x

Reatribua uma oportunidade existente a outro usuário.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssigneeContact;

/*
Purpose
PC-API-07 Assigning a new owner
*/

public class AssignOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String assigneeFirstName = "John";

     String assigneeLastName = "Doe";

     String assigneeEmail = "test@test.com";

     String businessTitle = "PartnerAccountManager";

     AssignOpportunityResponse response = getResponse(opportunityId,
 assigneeFirstName, assigneeLastName, assigneeEmail, businessTitle);

     ReferenceCodesUtils.formatOutput(response);
    }

 static AssignOpportunityResponse getResponse(String opportunityId, String
 assigneeFirstName, String assigneeLastName, String assigneeEmail, String
 businessTitle) {
```

```
  AssignOpportunityRequest assignOpportunityRequest =
 AssignOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .assignee(AssigneeContact.builder()
            .firstName(assigneeFirstName)
            .lastName(assigneeLastName)
            .email(assigneeEmail)
            .businessTitle(businessTitle)
            .build())
          .build();

        AssignOpportunityResponse response =
 client.assignOpportunity(assignOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte AssignOpportunitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Reatribua uma oportunidade existente a outro usuário.

```
#!/usr/bin/env python

"""
Purpose
PC-API-07 Assigning a new owner
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE
```

```
serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def assign_opportunity(identifier):
    assign_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "Assignee": {
            "BusinessTitle": "OpportunityOwner",
            "Email": "test@test.com",
            "FirstName": "John",
            "LastName": "Doe"
        }
    }
    try:
        # Perform an API call
        response =
 partner_central_client.assign_opportunity(**assign_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O4236468"

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

    print("-" * 88)
    print("Assigning a new owner to an opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(assign_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a AssignOpportunityReferência da API AWS SDK for Python (Boto3).

Use **`AssociateOpportunity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `AssociateOpportunity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Atualizar a entidade associada de uma oportunidade

Java

SDK para Java 2.x

Crie uma associação formal entre uma oportunidade e várias entidades relacionadas.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse

/*
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/
```

```
public class AssociateOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     AssociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

static AssociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte AssociateOpportunitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Crie uma associação formal entre uma oportunidade e várias entidades relacionadas.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def associate_opportunity(entity_type, entity_identifier, opportunityIdentifier):
    associate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
```

```
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0059717"
    opportunityIdentifier = "O5465588"

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

    print("-" * 88)
    print("Associate Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(associate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a AssociateOpportunityReferência da API AWS SDK for Python (Boto3).

Use **`CreateOpportunity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateOpportunity`.

.NET

SDK para .NET

Crie uma oportunidade.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
```

```
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new CreateOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Origin = "Partner Referral",
                    Customer = new Customer
                    {
                        Account = new Account
                        {
                            Address = new Address
                            {
                                CountryCode = "US",
                                PostalCode = "99502",
                                StateOrRegion = "Alaska"
                            },
                            CompanyName = "TestCompanyName",
                            Duns = "123456789",
                            WebsiteUrl = "www.test.io",
                            Industry = "Automotive"
                        },
                        Contacts = new List<Contact>
                        {
```

```
                            new Contact
                            {
                                Email = "test@test.io",
                                FirstName = "John  ",
                                LastName = "Doe",
                                Phone = "+14444444444",
                                BusinessTitle = "test title"
                            }
                        }
                    },
                    LifeCycle = new LifeCycle
                    {
                        ReviewStatus = "Submitted",
                        TargetCloseDate = "2024-12-30"
                    },
                    Marketing = new Marketing
                    {
                        Source = "None"
                    },
                    OpportunityType = "Net New Business",
                    PrimaryNeedsFromAws = new List<string> { "Co-Sell -
Architectural Validation" },
                    Project = new Project
                    {
                        Title = "Moin Test UUID",
                        CustomerBusinessProblem = "Sandbox is not working as
expected",
                        CustomerUseCase = "AI Machine Learning and Analytics",
                        DeliveryModels = new List<string> { "SaaS or PaaS" },
                        ExpectedCustomerSpend = new List<ExpectedCustomerSpend>
                        {
                            new ExpectedCustomerSpend
                            {
                                Amount = "2000.0",
                                CurrencyCode = "USD",
                                Frequency = "Monthly",
                                TargetCompany = "Ibexlabs"
                            }
                        },
                        SalesActivities = new List<string> { "Initialized
discussions with customer" }
                    }
                };
```

```
                try
                {
                    var response = await client.CreateOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                }
                catch (ValidationException ex)
                {
                    Console.WriteLine("Validation error: " + ex.Message);
                }
                catch (AmazonPartnerCentralSellingException e)
                {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- Para obter detalhes da API, consulte CreateOpportunitya Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

Crie uma oportunidade.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;
```

```
import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.MonetaryValue;
import
 software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import
 software.amazon.awssdk.services.partnercentralselling.model.SoftwareRevenue;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

public class CreateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
```

```
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

public static void main(String[] args) {

 String inputFile = "CreateOpportunity2.json";

 if (args.length > 0)
  inputFile = args[0];

 CreateOpportunityResponse response = createOpportunity(inputFile);

 client.close();
}

static CreateOpportunityResponse createOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);

 List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
 if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
  for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
   NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
     .time(Instant.parse(nextStepsHistoryJson.time))
     .value(nextStepsHistoryJson.value)
                 .build();
   nextStepsHistories.add(nextStepsHistory);
  }
 }

 LifeCycle lifeCycle = null;
 if ( root.lifeCycle != null ) {
  lifeCycle = LifeCycle.builder()
   .closedLostReason(root.lifeCycle.closedLostReason)
   .nextSteps(root.lifeCycle.nextSteps)
   .nextStepsHistory(nextStepsHistories)
   .reviewComments(root.lifeCycle.reviewComments)
   .reviewStatus(root.lifeCycle.reviewStatus)
   .reviewStatusReason(root.lifeCycle.reviewStatusReason)
   .stage(root.lifeCycle.stage)
```

```
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
  }

  Marketing marketing = null;
  if ( root.marketing != null ) {
   marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();

  }

  Address address = null;
  if ( root.customer != null && root.customer.account != null &&
 root.customer.account.address != null ) {
   address = Address.builder()
    .city(root.customer.account.address.city)
                .postalCode(root.customer.account.address.postalCode)
                .stateOrRegion(root.customer.account.address.stateOrRegion)
                .countryCode(root.customer.account.address.countryCode)
                .streetAddress(root.customer.account.address.streetAddress)
                .build();
  }

  Account account = null;
  if ( root.customer != null && root.customer.account!= null) {
   account = Account.builder()
             .address(address)
             .awsAccountId(root.customer.account.awsAccountId)
                .duns(root.customer.account.duns)
                .industry(root.customer.account.industry)
                .otherIndustry(root.customer.account.otherIndustry)
                .companyName(root.customer.account.companyName)
                .websiteUrl(root.customer.account.websiteUrl)
                .build();
  }

  List<Contact> contacts = new ArrayList<Contact>();
  if ( root.customer != null && root.customer.contacts != null) {
   for (org.example.entity.Contact jsonContact : root.customer.contacts) {
```

```
    Contact contact = Contact.builder()
                  .email(jsonContact.email)
                  .firstName(jsonContact.firstName)
                  .lastName(jsonContact.lastName)
                  .phone(jsonContact.phone)
                  .businessTitle(jsonContact.businessTitle)
                  .build();
    contacts.add(contact);
   }
  }

  Customer customer = Customer.builder()
    .account(account)
    .contacts(contacts)
    .build();

  Contact oportunityTeamContact = null;
  if (root.opportunityTeam != null && root.opportunityTeam.get(0) != null ) {
   oportunityTeamContact = Contact.builder()
                .firstName(root.opportunityTeam.get(0).firstName)
                .lastName(root.opportunityTeam.get(0).lastName)
                .email(root.opportunityTeam.get(0).email)
                .phone(root.opportunityTeam.get(0).phone)
                .businessTitle(root.opportunityTeam.get(0).businessTitle)
                .build();
  }

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
 ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
 root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

        Project project = null;
```

```
        if ( root.project != null) {
         project = Project.builder()
                .title(root.project.title)
                .customerBusinessProblem(root.project.customerBusinessProblem)
                .customerUseCase(root.project.customerUseCase)
                .deliveryModels(root.project.deliveryModels)
                .expectedCustomerSpend(expectedCustomerSpends)
                .salesActivities(root.project.salesActivities)
                .competitorName(root.project.competitorName)
                .otherSolutionDescription(root.project.otherSolutionDescription)
                .build();
        }

        SoftwareRevenue softwareRevenue = null;
        if ( root.softwareRevenue != null) {
         MonetaryValue monetaryValue = null;
         if ( root.softwareRevenue.value != null) {
          monetaryValue = MonetaryValue.builder()
            .amount(root.softwareRevenue.value.amount)
            .currencyCode(root.softwareRevenue.value.currencyCode)
            .build();
         }
         softwareRevenue = SoftwareRevenue.builder()
           .deliveryModel(root.softwareRevenue.deliveryModel)
           .effectiveDate(root.softwareRevenue.effectiveDate)
           .expirationDate(root.softwareRevenue.expirationDate)
           .value(monetaryValue)
           .build();
        }

  // Building the Actual CreateOpportunity Request
  CreateOpportunityRequest createOpportunityRequest =
 CreateOpportunityRequest.builder()
    .catalog(CATALOG_TO_USE)
    .clientToken(root.clientToken)
    .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws)
    .opportunityType(root.opportunityType)
    .lifeCycle(lifeCycle)
    .marketing(marketing)
    .nationalSecurity(root.nationalSecurity)
    .origin(root.origin)
    .customer(customer)
    .project(project)
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
```

```
    .opportunityTeam(oportunityTeamContact)
    .softwareRevenue(softwareRevenue)
    .build();

  CreateOpportunityResponse response =
 client.createOpportunity(createOpportunityRequest);
  System.out.println("Successfully created: " + response);

  return response;
    }

}
```

- Para obter detalhes da API, consulte CreateOpportunitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Crie uma oportunidade.

```
#!/usr/bin/env python
import boto3
import logging
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
import utils.stringify_details as sd
from botocore.client import ClientError
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

def create_opportunity(partner_central_client):
    create_opportunity_request = helper.remove_nulls(sd.stringify_json("src/
create_opportunity/createOpportunity.json"))
    try:
        # Perform an API call
```

```
        response =
 partner_central_client.create_opportunity(**create_opportunity_request)

        helper.pretty_print_datetime(response)

        # Retrieve the opportunity details
        get_response = partner_central_client.get_opportunity(
            Identifier=response["Id"],
            Catalog=CATALOG_TO_USE
        )
        helper.pretty_print_datetime(get_response)
        return response
    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

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

    print("-" * 88)
    print("Create Opportunity.")
    print("-" * 88)

    partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
    )

    create_opportunity(partner_central_client)

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a CreateOpportunityReferência da API AWS SDK for Python (Boto3).

Use **`DisassociateOpportunity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DisassociateOpportunity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Atualizar a entidade associada de uma oportunidade

Java

SDK para Java 2.x

Remova uma associação existente entre uma oportunidade e entidades relacionadas.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityReque
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRespo

/*
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class DisassociateOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
```

```
    String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

    String entityType = "Solutions";

    String entityIdentifier = "S-0000000";

    DisassociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

    ReferenceCodesUtils.formatOutput(response);
   }

 static DisassociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

  DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

       DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

       return response;
 }
}
```

- Para obter detalhes da API, consulte DisassociateOpportunitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Remova uma associação existente entre uma oportunidade e entidades relacionadas.

```
#!/usr/bin/env python
```

```
"""
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def disassociate_opportunity(entity_type, entity_identifier,
 opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0049999"
    opportunityIdentifier = "O4397574"
```

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

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(disassociate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DisassociateOpportunityReferência da API AWS SDK for Python (Boto3).

Use **`GetAwsOpportunitySummary`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetAwsOpportunitySummary`.

Java

SDK para Java 2.x

Recupera um resumo de uma AWS oportunidade.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRequ
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryResp
```

```
/*
 * Purpose
 * PC-API-25 Retrieves a summary of an AWS Opportunity.
 */

public class GetAwsOpportunitySummary {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetAwsOpportunitySummaryResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetAwsOpportunitySummaryResponse getResponse(String opportunityId)
 {

  GetAwsOpportunitySummaryRequest getOpportunityRequest =
 GetAwsOpportunitySummaryRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .relatedOpportunityIdentifier(opportunityId)
          .build();

  GetAwsOpportunitySummaryResponse response =
 client.getAwsOpportunitySummary(getOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte GetAwsOpportunitySummarya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Recupera um resumo de uma AWS oportunidade.

```
#!/usr/bin/env python

"""
Purpose
PC-API-25 Retrieves a summary of an AWS Opportunity.
 LifeCycle.ReviewStatus=Approved
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "RelatedOpportunityIdentifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_aws_opportunity_summary(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
```

```
    identifier = "05465588"

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

    print("-" * 88)
    print("Get AWS Opportunity summary.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAwsOpportunitySummaryReferência da API AWS SDK for Python (Boto3).

Use **`GetEngagementInvitation`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetEngagementInvitation`.

Java

SDK para Java 2.x

Recupera os detalhes de um convite de engajamento compartilhado AWS por um parceiro.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRespo
```

```
/*
 * Purpose
 * PC-API-22 Get engagement invitation opportunity
 */

public class GetEngagementInvitation {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static GetEngagementInvitationResponse getResponse(String opportunityId) {

  GetEngagementInvitationRequest getOpportunityRequest =
 GetEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte GetEngagementInvitationa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Recupera os detalhes de um convite de engajamento compartilhado AWS por um parceiro.

```
#!/usr/bin/env python

"""
Purpose
PC-API-22  GetOpportunityEngagementInvitation - Retrieves details of a specific
 engagement invitation.
This operation allows partners to view the invitation and its associated
 information,
such as the customer, project, and lifecycle details.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity_engagement_invitation(identifier):
    get_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_engagement_invitation
        return response

    except Exception as err:
        # Catch all client exceptions
```

```
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral-selling:us-east-1:aws:catalog/Sandbox/
engagement-invitation/engi-0000000IS0Qga"

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

    print("-" * 88)
    print("Given the ARN identifier, retrieve details of Opportunity Engagement
 Invitation.")
    print("-" * 88)


 helper.pretty_print_datetime(get_opportunity_engagement_invitation(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetEngagementInvitationReferência da API AWS SDK for Python (Boto3).

Use **`GetOpportunity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetOpportunity`.

.NET

SDK para .NET

Tenha uma oportunidade.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;
```

```
namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static readonly string identifier = "O1111111";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new GetOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Identifier = identifier
                };

                try {
                    var response = await client.GetOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte GetOpportunitya Referência AWS SDK para .NET da API.

Go

SDK para Go V2

Tenha uma oportunidade.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.GetOpportunity(context.TODO(),
 &partnercentralselling.GetOpportunityInput{
  Identifier: aws.String("O1111111"),
  Catalog:    aws.String("AWS"),
```

```
 })

 if err != nil {
  log.Fatal(err)
 }
 log.Println("printing opportuniy...\n")

 jsonOutput, err := json.MarshalIndent(output, "", "    ")

 fmt.Println(string(jsonOutput))
}
```

- Para obter detalhes da API, consulte GetOpportunitya Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Tenha uma oportunidade.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;

/*
 * Purpose
 * PC-API-08 Get updated Opportunity
 */
```

```
public class GetOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetOpportunityResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

        GetOpportunityResponse response =
 client.getOpportunity(getOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte GetOpportunitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Tenha uma oportunidade.

```
#!/usr/bin/env python

"""
Purpose
PC-API -08 Get updated Opportunity given opportunity id
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O5465588"

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

    print("-" * 88)
    print("Get updated Opportunity.")
```

```
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetOpportunityReferência da API AWS SDK for Python (Boto3).

Use **`ListEngagementInvitations`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListEngagementInvitations`.

Java

SDK para Java 2.x

Recupera uma lista de convites de engajamento enviados ao parceiro.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsReq
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsRes
import
 software.amazon.awssdk.services.partnercentralselling.model.ParticipantType;
import
 software.amazon.awssdk.services.partnercentralselling.model.EngagementInvitationSummary;
```

```
public class ListEngagementInvitations {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     List<EngagementInvitationSummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    static List<EngagementInvitationSummary> getResponse() {

  List<EngagementInvitationSummary> opportunitySummaries = new
 ArrayList<EngagementInvitationSummary>();

  ListEngagementInvitationsRequest listOpportunityRequest =
 ListEngagementInvitationsRequest.builder()
                .catalog(CATALOG_TO_USE)
                .participantType(ParticipantType.RECEIVER)
          .maxResults(5)
          .build();

  ListEngagementInvitationsResponse response =
 client.listEngagementInvitations(listOpportunityRequest);

     opportunitySummaries.addAll(response.engagementInvitationSummaries());

     client.close();

        return opportunitySummaries;
 }
}
```

- Para obter detalhes da API, consulte ListEngagementInvitationsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Recupera uma lista de convites de engajamento enviados ao parceiro.

```
#!/usr/bin/env python

"""
Purpose
PC-API-21 ListEngagementInvitations - Retrieves a list of engagement invitations
 based on specified criteria.
This operation allows partners to view all invitations to engagement.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def list_engagement_invitations():
    list_engagement_invitations_request ={
        "Catalog": CATALOG_TO_USE,
        "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_engagement_invitations(**list_engagement_invitations_request
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))
```

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

    print("-" * 88)
    print("Retrieve list of Engagement Invitations.")
    print("-" * 88)

    helper.pretty_print_datetime(list_engagement_invitations())

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a ListEngagementInvitationsReferência da API AWS SDK for Python (Boto3).

Use **`ListOpportunities`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListOpportunities`.

.NET

SDK para .NET

Liste oportunidades.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "Sandbox";
        static async Task Main(string[] args)
        {
```

```
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                //var config = new AmazonPartnerCentralSellingConfig()
                //{
                //    ServiceURL = "https://partnercentral-selling.us-
east-1.api.aws",
                //};
                //var client = new
 AmazonPartnerCentralSellingClient(awsCredentials, config);
                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);
                var request = new ListOpportunitiesRequest
                {
                    Catalog = catalogToUse,
                    MaxResults = 2
                };

                try {
                    var response = await client.ListOpportunitiesAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    foreach (var opportunity in response.OpportunitySummaries)
                    {
                        Console.WriteLine("Opportunity id: " + opportunity.Id);
                    }
                    string formattedJson =
 JsonConvert.SerializeObject(response.OpportunitySummaries, Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
```

```
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- Para obter detalhes da API, consulte ListOpportunitiesa Referência AWS SDK para .NET da API.

Go

SDK para Go V2

Liste oportunidades.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.ListOpportunities(context.TODO(),
 &partnercentralselling.ListOpportunitiesInput{
```

```
  MaxResults: aws.Int32(2),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }

 jsonOutput, err := json.MarshalIndent(output, "", "    ")
 fmt.Println(string(jsonOutput))
}
```

- Para obter detalhes da API, consulte ListOpportunitiesa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

Liste oportunidades.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.OpportunitySummary;

/*
```

```
 * Purpose
 * PC-API-18 Getting list of Opportunities
 */

public class ListOpportunititesPaging {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<OpportunitySummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    private static List<OpportunitySummary> getResponse() {
     List<OpportunitySummary> opportunitySummaries = new
 ArrayList<OpportunitySummary>();

  ListOpportunitiesRequest listOpportunityRequest =
 ListOpportunitiesRequest.builder()
                .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListOpportunitiesResponse response =
 client.listOpportunities(listOpportunityRequest);

     opportunitySummaries.addAll(response.opportunitySummaries());

     while (response.nextToken() != null && response.nextToken().length() > 0) {
      listOpportunityRequest = ListOpportunitiesRequest.builder()
                    .catalog(CATALOG_TO_USE)
              .maxResults(5)
              .nextToken(response.nextToken())
              .build();
      response = client.listOpportunities(listOpportunityRequest);
      opportunitySummaries.addAll(response.opportunitySummaries());
     }

     client.close();
```

```
        return opportunitySummaries;
 }
}
```

- Para obter detalhes da API, consulte ListOpportunitiesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Liste oportunidades.

```
#!/usr/bin/env python

"""
Purpose
PC-API -18 Getting list of Opportunities
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_opportunities():

    opportunity_list = []

    list_opportunities_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
```

```
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_opportunities(**list_opportunities_request)
        opportunity_list.extend(response["OpportunitySummaries"])

        while "NextToken" in response and response["NextToken"] is not None:
            list_opportunities_request["NextToken"] = response["NextToken"]
            response =
 partner_central_client.list_opportunities(**list_opportunities_request)
            opportunity_list.extend(response["OpportunitySummaries"])

        return opportunity_list

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

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

    print("-" * 88)
    print("Getting list of Opportunities.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_opportunities())

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a ListOpportunitiesReferência da API AWS SDK for Python (Boto3).

Use **`ListSolutions`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListSolutions`.

Java

SDK para Java 2.x

Recupera uma lista de soluções de parceiros que o parceiro registrou na Central de Parceiros.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsResponse;
import software.amazon.awssdk.services.partnercentralselling.model.SolutionBase;

/*
 * Purpose
 * PC-API-10 Getting list of solutions
 */

public class ListSolutions {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<SolutionBase> solutionSummaries = getResponse();
        ReferenceCodesUtils.formatOutput(solutionSummaries);
    }
```

```
    static List<SolutionBase> getResponse() {
  List<SolutionBase> solutionSummaries = new ArrayList<SolutionBase>();

  ListSolutionsRequest listSolutionsRequest = ListSolutionsRequest.builder()
    .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListSolutionsResponse response = client.listSolutions(listSolutionsRequest);

     solutionSummaries.addAll(response.solutionSummaries());

        return solutionSummaries;
 }
}
```

- Para obter detalhes da API, consulte ListSolutionsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Recupera uma lista de soluções de parceiros que o parceiro registrou na Central de Parceiros.

```
#!/usr/bin/env python

"""
Purpose
PC-API-10 Getting list of solutions
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
```

```
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_solutions():
    list_solutions_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_solutions(**list_solutions_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

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

    print("-" * 88)
    print("Getting list of solutions.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_solutions())

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a ListSolutionsReferência da API AWS SDK for Python (Boto3).

Use **`RejectEngagementInvitation`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `RejectEngagementInvitation`.

Java

SDK para Java 2.x

Rejeita e EngagementInvitation isso é AWS compartilhado.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRe
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRe

/*
 * Purpose
 * PC-API-05 AWS Originated(AO) rejection
 */

public class RejectEngagementInvitation {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

  RejectEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
```

```
    }

 static RejectEngagementInvitationResponse getResponse(String invitationId) {

        RejectEngagementInvitationRequest rejectOpportunityRequest =
 RejectEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .rejectionReason("Unable to support")
          .build();

  RejectEngagementInvitationResponse response =
 client.rejectEngagementInvitation(rejectOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte RejectEngagementInvitationa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Rejeita e EngagementInvitation isso é AWS compartilhado.

```
#!/usr/bin/env python

"""
Purpose
PC-API-05 AWS Originated AO rejection - RejectOpportunityEngagementInvitation -
 Rejects a engagement invitation.
This action indicates that the partner does not wish to participate in the
 engagement and
provides a reason for the rejection.
Upon rejection, a OpportunityEngagementInvitationRejected event is triggered.
Subsequently, the invitation will no longer be available for the partner to act
 on.
"""
import json
```

```
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def reject_opportunity_engagement_invitation(identifier, reject_reason):
    reject_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "RejectionReason": reject_reason
    }
    try:
        # Perform an API call
        response =
 partner_central_client.reject_engagement_invitation(**reject_opportunity_engagement_invi
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isviga"
    reject_reason = "Customer problem unclear"

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

    print("-" * 88)
    print("Given the ARN identifier and reject reason, reject the Opportunity
 Engagement Invitation.")
    print("-" * 88)


 helper.pretty_print_datetime(reject_opportunity_engagement_invitation(identifier,
 reject_reason))
```

```
if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a RejectEngagementInvitationReferência da API AWS SDK for Python (Boto3).

Use **`StartEngagementByAcceptingInvitationTask`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartEngagementByAcceptingInvitationTask`.

Java

SDK para Java 2.x

Inicia o noivado aceitando um EngagementInvitation.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingIn
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingIn
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRespo
import
 software.amazon.awssdk.services.partnercentralselling.model.InvitationStatus;

/*
```

```
Purpose
PC-API-04: Start Engagement By Accepting InvitationTask for AWS Originated(AO)
 opportunity
*/

public class StartEngagementByAcceptingInvitationTask {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String clientToken = "test-a30d161";

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementByAcceptingInvitationTaskResponse response =
 getResponse(opportunityId);

     if ( response == null) {
      System.out.println("Opportunity is not AWS Originated.");
     } else {
      ReferenceCodesUtils.formatOutput(response);
     }
    }

    private static GetEngagementInvitationResponse getInvitation(String
 invitationId) {

     GetEngagementInvitationRequest getRequest =
 GetEngagementInvitationRequest.builder()
          .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getRequest);

        return response;
 }
```

```
 static StartEngagementByAcceptingInvitationTaskResponse getResponse(String
 invitationId) {

  if ( getInvitation(invitationId).status().equals(InvitationStatus.PENDING)) {
   StartEngagementByAcceptingInvitationTaskRequest acceptOpportunityRequest =
     StartEngagementByAcceptingInvitationTaskRequest.builder()
     .catalog(Constants.CATALOG_TO_USE)
           .identifier(invitationId)
           .clientToken(clientToken)
           .build();

   StartEngagementByAcceptingInvitationTaskResponse response =
 client.startEngagementByAcceptingInvitationTask(acceptOpportunityRequest);
         return response;
  }
  return null;
 }
}
```

- Para obter detalhes da API, consulte StartEngagementByAcceptingInvitationTaska Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Inicia o noivado aceitando um EngagementInvitation.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError
```

```
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def start_engagement_by_accepting_invitation_task(identifier):

    response = get_opportunity(identifier)

    if ( response['Status'] == 'PENDING') :
        accept_opportunity_engagement_invitation_request ={
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-123456"
        }
        try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_by_accepting_invitation_task(**accept_opportunit
            return response

        except ClientError as err:
            # Catch all client exceptions
            print(err.response)
```

```
            return None
    else:
        return None

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isusga"
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)


 helper.pretty_print_datetime(start_engagement_by_accepting_invitation_task(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a StartEngagementByAcceptingInvitationTaskReferência da API AWS SDK for Python (Boto3).

Use **`StartEngagementFromOpportunityTask`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartEngagementFromOpportunityTask`.

Java

SDK para Java 2.x

Inicia o processo de engajamento com base em uma oportunidade existente aceitando o convite de engajamento e criando uma oportunidade correspondente no sistema do parceiro.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.AwsSubmission;
import
 software.amazon.awssdk.services.partnercentralselling.model.SalesInvolvementType;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportuni
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportuni
import software.amazon.awssdk.services.partnercentralselling.model.Visibility;

/*
 * Purpose
 * PC-API-01 Partner Originated (PO) opp submission(Start Engagement From
 Opportunity Task for AO Originated Opportunity)
 */

public class StartEngagementFromOpportunityTask {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementFromOpportunityTaskResponse response =
 getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static StartEngagementFromOpportunityTaskResponse getResponse(String
 opportunityId) {
```

```
  StartEngagementFromOpportunityTaskRequest submitOpportunityRequest =
 StartEngagementFromOpportunityTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .clientToken("test-annjqwesdsd99")

 .awsSubmission(AwsSubmission.builder().involvementType(SalesInvolvementType.CO_SELL).vis
          .build();

  StartEngagementFromOpportunityTaskResponse response =
 client.startEngagementFromOpportunityTask(submitOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte StartEngagementFromOpportunityTaska Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Inicia o processo de engajamento com base em uma oportunidade existente aceitando o convite de engajamento e criando uma oportunidade correspondente no sistema do parceiro.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE
```

```
serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def start_engagement_from_opportunity_task(identifier):

    start_engagement_from_opportunity_task_request ={
            "AwsSubmission": {
                "InvolvementType": "Co-Sell",
                "Visibility": "Full"
            },
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-annjqwesdsd99"
    }
    try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_from_opportunity_task(**start_engagement_from_op
            return response

    except ClientError as err:
            # Catch all client exceptions
            print(err.response)
            return None

def usage_demo():
    identifier = "O5465588"

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

    print("-" * 88)
    print("Start Engagement from Opportunity Task.")
    print("-" * 88)


 helper.pretty_print_datetime(start_engagement_from_opportunity_task(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a StartEngagementFromOpportunityTaskReferência da API AWS SDK for Python (Boto3).

Use **`UpdateOpportunity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `UpdateOpportunity`.

Java

SDK para Java 2.x

Atualize uma oportunidade.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
```

```
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import
 software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.ReviewStatus;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

/*
 * Purpose
 * PC-API-02/06 Update opportunity when LifeCycle.ReviewStatus is not Submitted
 or In-Review
 */

public class UpdateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String OPPORTUNITY_ORIGIN = ORIGIN_PARTNER_ORIGINATED;

 public static void main(String[] args) {

  String inputFile = "updateOpportunity.json";

  if (args.length > 0)
   inputFile = args[0];
```

```
 UpdateOpportunityResponse response = updateOpportunity(inputFile);

 client.close();
}

public static GetOpportunityResponse getResponse(String opportunityId) {

       GetOpportunityRequest getOpportunityRequest =
GetOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .identifier(opportunityId)
         .build();

       GetOpportunityResponse response =
client.getOpportunity(getOpportunityRequest);
       System.out.println(opportunityId + ":" + response);
       return response;
}

public static UpdateOpportunityResponse updateOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);
 GetOpportunityResponse response = getResponse(root.identifier);

 if (response != null
   && response.lifeCycle() != null
   && response.lifeCycle().reviewStatus() != null
   && response.lifeCycle().reviewStatus() != ReviewStatus.SUBMITTED
   && response.lifeCycle().reviewStatus() != ReviewStatus.IN_REVIEW) {

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
      .time(Instant.parse(nextStepsHistoryJson.time))
      .value(nextStepsHistoryJson.value)
                  .build();
    nextStepsHistories.add(nextStepsHistory);
   }
  }
```

```
  LifeCycle lifeCycle = null;
  if ( root.lifeCycle != null ) {
   lifeCycle = LifeCycle.builder()
    .closedLostReason(root.lifeCycle.closedLostReason)
    .nextSteps(root.lifeCycle.nextSteps)
    .nextStepsHistory(nextStepsHistories)
    .reviewComments(root.lifeCycle.reviewComments)
    .reviewStatus(root.lifeCycle.reviewStatus)
    .reviewStatusReason(root.lifeCycle.reviewStatusReason)
    .stage(root.lifeCycle.stage)
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
  }

  Marketing marketing = null;
  if ( root.marketing != null ) {
   marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();

  }

  Address address = null;
  if (root.customer != null && root.customer.account != null &&
root.customer.account.address != null) {
   address =
Address.builder().postalCode(root.customer.account.address.postalCode)
     .stateOrRegion(root.customer.account.address.stateOrRegion)
     .countryCode(root.customer.account.address.countryCode).build();
  }

  Account account = null;
  if (root.customer != null && root.customer.account != null) {
   account = Account.builder().address(address).duns(root.customer.account.duns)

.industry(root.customer.account.industry).companyName(root.customer.account.companyName)
     .websiteUrl(root.customer.account.websiteUrl).build();
  }

  List<Contact> contacts = new ArrayList<Contact>();
```

```
  if ( root.customer != null && root.customer.contacts != null) {
   for (org.example.entity.Contact jsonContact : root.customer.contacts) {
    Contact contact = Contact.builder()
                  .email(jsonContact.email)
                  .firstName(jsonContact.firstName)
                  .lastName(jsonContact.lastName)
                  .phone(jsonContact.phone)
                  .businessTitle(jsonContact.businessTitle)
                  .build();
    contacts.add(contact);
   }
  }

  Customer customer =
Customer.builder().account(account).contacts(contacts).build();

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

  Project project = null;
  if (root.project != null) {
   project = Project.builder().title(root.project.title)
     .customerBusinessProblem(root.project.customerBusinessProblem)

.customerUseCase(root.project.customerUseCase).deliveryModels(root.project.deliveryModel
     .expectedCustomerSpend(expectedCustomerSpends)

.salesActivities(root.project.salesActivities).competitorName(root.project.competitorNam
     .otherSolutionDescription(root.project.otherSolutionDescription).build();
  }
```

```
    // Building the Actual CreateOpportunity Request
    UpdateOpportunityRequest updateOpportunityRequest =
  UpdateOpportunityRequest.builder().catalog(root.catalog)

  .identifier(root.identifier).lastModifiedDate(Instant.parse(root.lastModifiedDate))

  .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws).opportunityType(root.opportuni
      .lifeCycle(lifeCycle)
      .customer(customer)
      .project(project)
      .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
      .marketing(marketing)
      .nationalSecurity(root.nationalSecurity)
      .opportunityType(root.opportunityType)
      .build();

    UpdateOpportunityResponse updateResponse =
  client.updateOpportunity(updateOpportunityRequest);
    System.out.println("Successfully updated opportunity: " + updateResponse);

    return updateResponse;
   } else {
    System.out.println("Opportunity cannot be updated.");
    return null;
   }
     }
}
```

- Para obter detalhes da API, consulte UpdateOpportunitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

Atualize uma oportunidade.

```
#!/usr/bin/env python

"""
Purpose
```

```
PC-API-2  Updating Partner Originated Opportunity
"""
import logging
import boto3
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
from botocore.client import ClientError
import utils.stringify_details as sd
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def update_opportunity():
    update_opportunity_request_orig = sd.stringify_json("src/update_opportunity/
update_opportunity_technical_validation.json")
    update_opportunity_request =
 helper.remove_nulls(update_opportunity_request_orig)

    try:
        # Perform an API call
        response =
 partner_central_client.update_opportunity(**update_opportunity_request)
```

```
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def update_opportunity_if_eligible(identifier):
    response = get_opportunity(identifier)
    if response is not None:
        return update_opportunity()
    else:
        print("Failed to retrieve opportunity details")

def usage_demo():
    identifier = "05465588"

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

    print("-" * 88)
    print("Updating opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(update_opportunity_if_eligible(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a UpdateOpportunityReferência da API AWS SDK for Python (Boto3).

Cenários para o uso do Partner Central AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Partner Central com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções na Central de Parceiros ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Atualizar a entidade associada de uma oportunidade

Atualizar a entidade associada de uma oportunidade

Os exemplos de código a seguir mostram como:

- Desassociar uma entidade antiga.
- Associar uma entidade nova.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Cenários.

Atualizar a entidade associada de uma oportunidade

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityReque
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRespo

/*
Purpose
PC-API -17 Replacing a solution
*/

public class ReplaceSolution {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";
     String originalEntityIdentifier = "S-0000000";
     String newEntityIdentifier = "S-0011111";

     disassociateOppornitityResponse(opportunityId, entityType,
 originalEntityIdentifier );
     AssociateOpportunityResponse associateOpportunityResponse =
 associateOpportunityResponse(opportunityId, entityType, newEntityIdentifier );

     ReferenceCodesUtils.formatOutput(associateOpportunityResponse);
    }

 private static AssociateOpportunityResponse associateOpportunityResponse(String
 opportunityId, String entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();
```

```
        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }

 private static DisassociateOpportunityResponse
 disassociateOppornitityResponse(String opportunityId, String entityType, String
 entityIdentifier) {
  PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

        DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AssociateOpportunity
  - DisassociateOpportunity

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualizar a entidade associada de uma oportunidade

```
#!/usr/bin/env python

"""
Purpose
PC-API -17 Replacing a solution
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def replace_solution(original_entity_identifier, new_entity_identifier,
 opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : original_entity_identifier
    }

    associate_opportunity_request ={
```

```
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : new_entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    original_entity_identifier = "S-0049999"
    new_entity_identifier = "S-0050014"
    opportunityIdentifier = "O4397574"

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

    print("-" * 88)
    print("Replacing a solution.")
    print("-" * 88)

    helper.pretty_print_datetime(replace_solution(original_entity_identifier,
 new_entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Python (Boto3).
  - AssociateOpportunity
  - DisassociateOpportunity

Exemplos de código para AWS Payment Cryptography usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Payment Cryptography com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS Payment Cryptography Guia do usuário — Mais informações sobre AWS Payment Cryptography.
- AWS Payment Cryptography Referência da API — Detalhes sobre todas as AWS Payment Cryptography ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para AWS Payment Cryptography usar AWS SDKs
  - Introdução à criptografia de segurança de pagamento

Cenários para AWS Payment Cryptography usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS Payment Cryptography com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS Payment Cryptography ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução à criptografia de segurança de pagamento

Introdução à criptografia de segurança de pagamento

O exemplo de código a seguir mostra como:

- Criar uma chave.
- Verificar o valor CVV2.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Payment Cryptography Getting Started Script
# This script demonstrates how to use AWS Payment Cryptography to create a key,
# generate and verify CVV2 values, and clean up resources.

set -euo pipefail

# Initialize log file with secure permissions
LOG_FILE="payment-cryptography-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "AWS Payment Cryptography Tutorial - $(date)" > "$LOG_FILE"

# Function to log messages
log() {
    local message="$1"
    echo "$(date +"%Y-%m-%d %H:%M:%S") - $message" | tee -a "$LOG_FILE"
}
```

```
# Function to handle errors
handle_error() {
    local error_message="$1"
    log "ERROR: $error_message"
    log "Script failed. Please check the log file: $LOG_FILE"

    echo ""
    echo "==========================================="
    echo "ERROR ENCOUNTERED"
    echo "==========================================="
    echo "The script encountered an error: $error_message"
    echo "Resources created will be listed below."
    echo ""

    if [ -n "${KEY_ARN:-}" ]; then
        echo "Key ARN: $KEY_ARN"
    fi

    exit 1
}

# Function to check command output for errors
check_error() {
    local output="$1"
    local command="$2"

    if echo "$output" | grep -iq "error\|exception\|fail"; then
        handle_error "Command failed: $command"
    fi
}

# Validate AWS CLI is available and credentials are configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not properly configured"
fi

log "Starting AWS Payment Cryptography tutorial"

# Step 1: Create a key
log "Step 1: Creating a card verification key (CVK)"
```

```
if ! KEY_OUTPUT=$(aws payment-cryptography create-key \
  --exportable \
  --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,Key
 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-payment-cryptography-
gs 2>&1); then
    handle_error "Failed to create key"
fi

echo "$KEY_OUTPUT"
check_error "$KEY_OUTPUT" "create-key"

# Extract the Key ARN from the output
KEY_ARN=$(echo "$KEY_OUTPUT" | grep -o '"KeyArn": "[^"]*' | cut -d'"' -f4)

if [ -z "$KEY_ARN" ]; then
    handle_error "Failed to extract Key ARN from output"
fi

log "Successfully created key with ARN: $KEY_ARN"

# Step 2: Generate a CVV2 value
log "Step 2: Generating a CVV2 value"
if ! CVV2_OUTPUT=$(aws payment-cryptography-data generate-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --generation-attributes CardVerificationValue2={CardExpiryDate=0123} 2>&1);
 then
    handle_error "Failed to generate CVV2 value"
fi

echo "$CVV2_OUTPUT"
check_error "$CVV2_OUTPUT" "generate-card-validation-data"

# Extract the CVV2 value from the output - updated to use ValidationData instead
 of CardDataValue
CVV2_VALUE=$(echo "$CVV2_OUTPUT" | grep -o '"ValidationData": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CVV2_VALUE" ]; then
    handle_error "Failed to extract CVV2 value from output"
fi
```

```
log "Successfully generated CVV2 value"

# Step 3: Verify the CVV2 value
log "Step 3: Verifying the CVV2 value"
if ! VERIFY_OUTPUT=$(aws payment-cryptography-data verify-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data "$CVV2_VALUE" 2>&1); then
    handle_error "Failed to verify CVV2 value"
fi

echo "$VERIFY_OUTPUT"
check_error "$VERIFY_OUTPUT" "verify-card-validation-data"

log "Successfully verified CVV2 value"

# Step 4: Perform a negative test
log "Step 4: Performing a negative test with incorrect CVV2"
if aws payment-cryptography-data verify-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data 999 2>&1; then
    handle_error "Negative test did not fail as expected"
fi

log "Negative test completed successfully (verification failed as expected)"

# Display created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "Key ARN: $KEY_ARN"
echo ""

# Auto-confirm cleanup
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Proceeding with cleanup of all created resources..."

log "Step 5: Cleaning up resources"
```

```
# Delete the key
log "Deleting key: $KEY_ARN"
if ! DELETE_OUTPUT=$(aws payment-cryptography delete-key \
  --key-identifier "$KEY_ARN" 2>&1); then
    handle_error "Failed to delete key"
fi

echo "$DELETE_OUTPUT"
check_error "$DELETE_OUTPUT" "delete-key"

log "Key scheduled for deletion. Default waiting period is 7 days."
log "To cancel deletion before the waiting period ends, use:"
log "aws payment-cryptography restore-key --key-identifier $KEY_ARN"

echo ""
echo "==========================================="
echo "CLEANUP COMPLETE"
echo "==========================================="
echo "The key has been scheduled for deletion after the default waiting period (7
 days)."
echo "To cancel deletion before the waiting period ends, use:"
echo "aws payment-cryptography restore-key --key-identifier $KEY_ARN"

log "Tutorial completed successfully"
echo ""
echo "Tutorial completed successfully. See $LOG_FILE for details."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateKey
  - DeleteKey
  - RestoreKey

Exemplos de código para o Amazon Personalize usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Personalize com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Amazon Personalize: mais informações sobre o Amazon Personalize.
- Referência da API Amazon Personalize: detalhes sobre todas as ações disponíveis do Amazon Personalize.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon Personalize usando AWS SDKs
  - Ações para o Amazon Personalize usando AWS SDKs
    - Use CreateBatchInferenceJobcom um AWS SDK
    - Use CreateBatchSegmentJobcom um AWS SDK
    - Use CreateCampaigncom um AWS SDK
    - Use CreateDatasetcom um AWS SDK
    - Use CreateDatasetExportJobcom um AWS SDK
    - Use CreateDatasetGroupcom um AWS SDK
    - Use CreateDatasetImportJobcom um AWS SDK
    - Use CreateEventTrackercom um AWS SDK
    - Use CreateFiltercom um AWS SDK
    - Use CreateRecommendercom um AWS SDK
    - Use CreateSchemacom um AWS SDK
    - Use CreateSolutioncom um AWS SDK
    - Use CreateSolutionVersioncom um AWS SDK
    - Use DeleteCampaigncom um AWS SDK
    - Use DeleteEventTrackercom um AWS SDK
    - Use DeleteSolutioncom um AWS SDK

- Use DescribeCampaigncom um AWS SDK
- Use DescribeRecipecom um AWS SDK
- Use DescribeSolutioncom um AWS SDK
- Use ListCampaignscom um AWS SDK
- Use ListDatasetGroupscom um AWS SDK
- Use ListRecipescom um AWS SDK
- Use ListSolutionscom um AWS SDK
- Use UpdateCampaigncom um AWS SDK

Exemplos básicos para o Amazon Personalize usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Personalize com os AWS SDKs.

Exemplos

- Ações para o Amazon Personalize usando AWS SDKs
  - Use CreateBatchInferenceJobcom um AWS SDK
  - Use CreateBatchSegmentJobcom um AWS SDK
  - Use CreateCampaigncom um AWS SDK
  - Use CreateDatasetcom um AWS SDK
  - Use CreateDatasetExportJobcom um AWS SDK
  - Use CreateDatasetGroupcom um AWS SDK
  - Use CreateDatasetImportJobcom um AWS SDK
  - Use CreateEventTrackercom um AWS SDK
  - Use CreateFiltercom um AWS SDK
  - Use CreateRecommendercom um AWS SDK
  - Use CreateSchemacom um AWS SDK
  - Use CreateSolutioncom um AWS SDK
  - Use CreateSolutionVersioncom um AWS SDK
  - Use DeleteCampaigncom um AWS SDK
  - Use DeleteEventTrackercom um AWS SDK
  - Use DeleteSolutioncom um AWS SDK

- Use DescribeCampaigncom um AWS SDK
- Use DescribeRecipecom um AWS SDK
- Use DescribeSolutioncom um AWS SDK
- Use ListCampaignscom um AWS SDK
- Use ListDatasetGroupscom um AWS SDK
- Use ListRecipescom um AWS SDK
- Use ListSolutionscom um AWS SDK
- Use UpdateCampaigncom um AWS SDK

Ações para o Amazon Personalize usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Personalize com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para ver uma lista completa, consulte a Referência da API do Amazon Personalize.

Exemplos

- Use CreateBatchInferenceJobcom um AWS SDK
- Use CreateBatchSegmentJobcom um AWS SDK
- Use CreateCampaigncom um AWS SDK
- Use CreateDatasetcom um AWS SDK
- Use CreateDatasetExportJobcom um AWS SDK
- Use CreateDatasetGroupcom um AWS SDK
- Use CreateDatasetImportJobcom um AWS SDK
- Use CreateEventTrackercom um AWS SDK
- Use CreateFiltercom um AWS SDK
- Use CreateRecommendercom um AWS SDK
- Use CreateSchemacom um AWS SDK
- Use CreateSolutioncom um AWS SDK
- Use CreateSolutionVersioncom um AWS SDK
- Use DeleteCampaigncom um AWS SDK

- Use DeleteEventTrackercom um AWS SDK
- Use DeleteSolutioncom um AWS SDK
- Use DescribeCampaigncom um AWS SDK
- Use DescribeRecipecom um AWS SDK
- Use DescribeSolutioncom um AWS SDK
- Use ListCampaignscom um AWS SDK
- Use ListDatasetGroupscom um AWS SDK
- Use ListRecipescom um AWS SDK
- Use ListSolutionscom um AWS SDK
- Use UpdateCampaigncom um AWS SDK

Use **`CreateBatchInferenceJob`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateBatchInferenceJob`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                        String solutionVersionArn,
                        String jobName,
                        String s3InputDataSourcePath,
                        String s3DataDestinationPath,
                        String roleArn,
                        String explorationWeight,
                        String explorationItemAgeCutOff) {

                long waitInMilliseconds = 60 * 1000;
                String status;
                String batchInferenceJobArn;
```

```
                try {

                        // Set up data input and output parameters.
                        S3DataConfig inputSource = S3DataConfig.builder()
                                        .path(s3InputDataSourcePath)
                                        .build();

                        S3DataConfig outputDestination = S3DataConfig.builder()
                                        .path(s3DataDestinationPath)
                                        .build();

                        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
                                        .s3DataSource(inputSource)
                                        .build();

                        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
                                        .s3DataDestination(outputDestination)
                                        .build();

                        // Optional code to build the User-Personalization
specific item exploration
                        // config.
                        HashMap<String, String> explorationConfig = new
HashMap<>();

                        explorationConfig.put("explorationWeight",
explorationWeight);
                        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

                        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
                                        .itemExplorationConfig(explorationConfig)
                                        .build();

                        // End optional User-Personalization recipe specific
code.

                        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
                                        .builder()
```

```
                                        .solutionVersionArn(solutionVersionArn)
                                        .jobInput(jobInput)
                                        .jobOutput(jobOutputLocation)
                                        .jobName(jobName)
                                        .roleArn(roleArn)
                                        .batchInferenceJobConfig(jobConfig) //
Optional
                                        .build();

                        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                                        .batchInferenceJobArn();

                        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
                                        .builder()

.batchInferenceJobArn(batchInferenceJobArn)
                                        .build();

                        long maxTime = Instant.now().getEpochSecond() + 3 * 60 *
60;
                        while (Instant.now().getEpochSecond() < maxTime) {

                                BatchInferenceJob batchInferenceJob =
personalizeClient

.describeBatchInferenceJob(describeBatchInferenceJobRequest)
                                                .batchInferenceJob();

                                status = batchInferenceJob.status();
                                System.out.println("Batch inference job status: "
+ status);

                                if (status.equals("ACTIVE") ||
status.equals("CREATE FAILED")) {
                                        break;
                                }
                                try {
                                        Thread.sleep(waitInMilliseconds);
                                } catch (InterruptedException e) {
                                        System.out.println(e.getMessage());
                                }
                        }
```

```
                        return batchInferenceJobArn;

                } catch (PersonalizeException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return "";
        }
```

- Para obter detalhes da API, consulte CreateBatchInferenceJoba Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchInferenceJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch inference job's parameters.

export const createBatchInferenceJobParam = {
  jobName: "JOB_NAME",
  jobInput: {
    s3DataSource: {
      path: "INPUT_PATH",
    },
  },
  jobOutput: {
    s3DataDestination: {
      path: "OUTPUT_PATH",
```

```
    },
  },
  roleArn: "ROLE_ARN",
  solutionVersionArn: "SOLUTION_VERSION_ARN",
  numResults: 20,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateBatchInferenceJobCommand(createBatchInferenceJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateBatchInferenceJoba Referência AWS SDK para JavaScript da API.

Use **`CreateBatchSegmentJob`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateBatchSegmentJob`.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchSegmentJobCommand } from "@aws-sdk/client-personalize";
```

```
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch segment job's parameters.

export const createBatchSegmentJobParam = {
  jobName: "NAME",
  jobInput: {
    s3DataSource: {
      path: "INPUT_PATH",
    },
  },
  jobOutput: {
    s3DataDestination: {
      path: "OUTPUT_PATH",
    },
  },
  roleArn: "ROLE_ARN",
  solutionVersionArn: "SOLUTION_VERSION_ARN",
  numResults: 20,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateBatchSegmentJobCommand(createBatchSegmentJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateBatchSegmentJoba Referência AWS SDK para JavaScript da API.

Use **`CreateCampaign`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateCampaign`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createPersonalCompaign(PersonalizeClient
personalizeClient, String solutionVersionArn,
           String name) {

       try {
           CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
                   .minProvisionedTPS(1)
                   .solutionVersionArn(solutionVersionArn)
                   .name(name)
                   .build();

           CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
           System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());
           return campaignResponse.campaignArn();
       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return null;
   }
```

- Para obter detalhes da API, consulte CreateCampaigna Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.

import { CreateCampaignCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the campaign's parameters.
export const createCampaignParam = {
  solutionVersionArn: "SOLUTION_VERSION_ARN" /* required */,
  name: "NAME" /* required */,
  minProvisionedTPS: 1 /* optional integer */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateCampaignCommand(createCampaignParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateCampaigna Referência AWS SDK para JavaScript da API.

Use **`CreateDataset`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDataset`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static String createDataset(PersonalizeClient personalizeClient,
        String datasetName,
        String datasetGroupArn,
        String datasetType,
        String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
                .name(datasetName)
                .datasetGroupArn(datasetGroupArn)
                .datasetType(datasetType)
                .schemaArn(schemaArn)
                .build();

        String datasetArn = personalizeClient.createDataset(request)
                .datasetArn();
        System.out.println("Dataset " + datasetName + " created.");
        return datasetArn;

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

- Para obter detalhes da API, consulte CreateDataset a Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  datasetType: "DATASET_TYPE" /* required */,
  name: "NAME" /* required */,
  schemaArn: "SCHEMA_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetCommand(createDatasetParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateDataset a Referência AWS SDK para JavaScript da API.

Use **`CreateDatasetExportJob`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDatasetExportJob`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createDatasetExportJob(PersonalizeClient
personalizeClient,
           String jobName,
           String datasetArn,
           IngestionMode ingestionMode,
           String roleArn,
           String s3BucketPath,
           String kmsKeyArn) {

       long waitInMilliseconds = 30 * 1000; // 30 seconds
       String status = null;

       try {

           S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
           DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
                   .build();

           CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
                   .jobName(jobName)
                   .datasetArn(datasetArn)
                   .ingestionMode(ingestionMode)
                   .jobOutput(jobOutput)
                   .roleArn(roleArn)
                   .build();
```

```
            String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

            DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
                    .datasetExportJobArn(datasetExportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetExportJob datasetExportJob = personalizeClient

.describeDatasetExportJob(describeDatasetExportJobRequest)
                        .datasetExportJob();

                status = datasetExportJob.status();
                System.out.println("Export job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    return status;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateDatasetExportJoba Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetExportJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the export job parameters.
export const datasetExportJobParam = {
  datasetArn: "DATASET_ARN" /* required */,
  jobOutput: {
    s3DataDestination: {
      path: "S3_DESTINATION_PATH" /* required */,
      //kmsKeyArn: 'ARN'  /* include if your bucket uses AWS KMS for encryption
    },
  },
  jobName: "NAME" /* required */,
  roleArn: "ROLE_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetExportJobCommand(datasetExportJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateDatasetExportJoba Referência AWS SDK para JavaScript da API.

Use **`CreateDatasetGroup`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDatasetGroup`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

       try {
           CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                   .name(datasetGroupName)
                   .build();
           return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
       } catch (PersonalizeException e) {
           System.out.println(e.awsErrorDetails().errorMessage());
       }
       return "";
   }
```

Criar um grupo de conjunto de dados de domínio.

```
   public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
           String datasetGroupName,
```

```
            String domain) {

        try {
            CreateDatasetGroupRequest createDatasetGroupRequest =
 CreateDatasetGroupRequest.builder()
                    .name(datasetGroupName)
                    .domain(domain)
                    .build();
            return
 personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateDatasetGroupa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.

import { CreateDatasetGroupCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset group parameters.
export const createDatasetGroupParam = {
  name: "NAME" /* required */,
```

```
};

export const run = async (createDatasetGroupParam) => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetGroupCommand(createDatasetGroupParam),
    );
    console.log("Success", response);
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run(createDatasetGroupParam);
```

Criar um grupo de conjunto de dados de domínio.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetGroupCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: "NAME" /* required */,
  domain:
    "DOMAIN" /* required for a domain dsg, specify ECOMMERCE or VIDEO_ON_DEMAND
 */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetGroupCommand(domainDatasetGroupParams),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

- Para obter detalhes da API, consulte CreateDatasetGroupa Referência AWS SDK para JavaScript da API.

Use **`CreateDatasetImportJob`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateDatasetImportJob`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
           String jobName,
           String datasetArn,
           String s3BucketPath,
           String roleArn) {

       long waitInMilliseconds = 60 * 1000;
       String status;
       String datasetImportJobArn;

       try {
           DataSource importDataSource = DataSource.builder()
                   .dataLocation(s3BucketPath)
                   .build();

           CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
                   .datasetArn(datasetArn)
                   .dataSource(importDataSource)
```

```
                    .jobName(jobName)
                    .roleArn(roleArn)
                    .build();

            datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                    .datasetImportJobArn();
            DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
                    .datasetImportJobArn(datasetImportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetImportJob datasetImportJob = personalizeClient

.describeDatasetImportJob(describeDatasetImportJobRequest)
                        .datasetImportJob();

                status = datasetImportJob.status();
                System.out.println("Dataset import job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return datasetImportJobArn;

        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateDatasetImportJoba Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetImportJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: "DATASET_ARN" /* required */,
  dataSource: {
    /* required */
    dataLocation: "S3_PATH",
  },
  jobName: "NAME" /* required */,
  roleArn: "ROLE_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetImportJobCommand(datasetImportJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

- Para obter detalhes da API, consulte CreateDatasetImportJoba Referência AWS SDK para JavaScript da API.

Use **`CreateEventTracker`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateEventTracker`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
           String datasetGroupArn) {

       String eventTrackerId = "";
       String eventTrackerArn;
       long maxTime = 3 * 60 * 60; // 3 hours
       long waitInMilliseconds = 20 * 1000; // 20 seconds
       String status;

       try {

           CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
                   .name(eventTrackerName)
                   .datasetGroupArn(datasetGroupArn)
                   .build();

           CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
```

```
                    .createEventTracker(createEventTrackerRequest);

            eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
            eventTrackerId = createEventTrackerResponse.trackingId();
            System.out.println("Event tracker ARN: " + eventTrackerArn);
            System.out.println("Event tracker ID: " + eventTrackerId);

            maxTime = Instant.now().getEpochSecond() + maxTime;

            DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
                    .eventTrackerArn(eventTrackerArn)
                    .build();

            while (Instant.now().getEpochSecond() < maxTime) {

                status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
                System.out.println("EventTracker status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return eventTrackerId;
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return eventTrackerId;
    }
```

- Para obter detalhes da API, consulte CreateEventTrackera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateEventTrackerCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the event tracker's parameters.
export const createEventTrackerParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  name: "NAME" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateEventTrackerCommand(createEventTrackerParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateEventTrackera Referência AWS SDK para JavaScript da API.

Use **`CreateFilter`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateFilter`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static String createFilter(PersonalizeClient personalizeClient,
        String filterName,
        String datasetGroupArn,
        String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .datasetGroupArn(datasetGroupArn)
                .filterExpression(filterExpression)
                .build();

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obter detalhes da API, consulte CreateFiltera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateFilterCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the filter's parameters.
export const createFilterParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  name: "NAME" /* required */,
  filterExpression: "FILTER_EXPRESSION" /*required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateFilterCommand(createFilterParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateFiltera Referência AWS SDK para JavaScript da API.

Use **`CreateRecommender`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateRecommender`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);

            DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
```

```
            while (Instant.now().getEpochSecond() < maxTime) {

                recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
                System.out.println("Recommender status: " + recommenderStatus);

                if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return recommenderArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateRecommendera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
```

```
import { CreateRecommenderCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the recommender's parameters.
export const createRecommenderParam = {
  name: "NAME" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateRecommenderCommand(createRecommenderParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateRecommendera Referência AWS SDK para JavaScript da API.

Use **`CreateSchema`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateSchema`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

       String schema = null;
       try {
           schema = new String(Files.readAllBytes(Paths.get(filePath)));
       } catch (IOException e) {
           System.out.println(e.getMessage());
       }

       try {
           CreateSchemaRequest createSchemaRequest =
CreateSchemaRequest.builder()
                   .name(schemaName)
                   .schema(schema)
                   .build();

           String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

           System.out.println("Schema arn: " + schemaArn);

           return schemaArn;

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

Crie um esquema com um domínio.

```
   public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
           String filePath) {

       String schema = null;
       try {
           schema = new String(Files.readAllBytes(Paths.get(filePath)));
       } catch (IOException e) {
           System.out.println(e.getMessage());
       }

       try {
           CreateSchemaRequest createSchemaRequest =
CreateSchemaRequest.builder()
                   .name(schemaName)
                   .domain(domain)
                   .schema(schema)
                   .build();

           String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

           System.out.println("Schema arn: " + schemaArn);

           return schemaArn;

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte CreateSchemaa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from "node:fs";

const schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = "TEST"; // For unit tests.
}
// Set the schema parameters.
export const createSchemaParam = {
  name: "NAME" /* required */,
  schema: mySchema /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSchemaCommand(createSchemaParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
```

```
  }
};
run();
```

Crie um esquema com um domínio.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from "node:fs";

const schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = "TEST"; // for unit tests.
}

// Set the domain schema parameters.
export const createDomainSchemaParam = {
  name: "NAME" /* required */,
  schema: mySchema /* required */,
  domain:
    "DOMAIN" /* required for a domain dataset group, specify ECOMMERCE or
 VIDEO_ON_DEMAND */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSchemaCommand(createDomainSchemaParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
```

```
  }
};
run();
```

- Para obter detalhes da API, consulte CreateSchemaa Referência AWS SDK para JavaScript da API.

Use **`CreateSolution`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateSolution`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
           String datasetGroupArn,
           String solutionName,
           String recipeArn) {

       try {
           CreateSolutionRequest solutionRequest =
CreateSolutionRequest.builder()
                   .name(solutionName)
                   .datasetGroupArn(datasetGroupArn)
                   .recipeArn(recipeArn)
                   .build();

           CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
           return solutionResponse.solutionArn();

       } catch (PersonalizeException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateSolutiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution parameters.
export const createSolutionParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  name: "NAME" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionCommand(createSolutionParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
```

```
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateSolutiona Referência AWS SDK para JavaScript da API.

Use **`CreateSolutionVersion`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateSolutionVersion`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
       long maxTime = 0;
       long waitInMilliseconds = 30 * 1000; // 30 seconds
       String solutionStatus = "";
       String solutionVersionStatus = "";
       String solutionVersionArn = "";

       try {
           DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

           // Wait until solution is active.
           while (Instant.now().getEpochSecond() < maxTime) {
```

```
                solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
                System.out.println("Solution status: " + solutionStatus);

                if (solutionStatus.equals("ACTIVE") ||
solutionStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }

            if (solutionStatus.equals("ACTIVE")) {

                CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                        .solutionArn(solutionArn)
                        .build();

                CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                        .createSolutionVersion(createSolutionVersionRequest);
                solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

                System.out.println("Solution version ARN: " +
solutionVersionArn);

                DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                        .solutionVersionArn(solutionVersionArn)
                        .build();

                while (Instant.now().getEpochSecond() < maxTime) {

                    solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                            .solutionVersion().status();
                    System.out.println("Solution version status: " +
solutionVersionStatus);
```

```
                    if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                        break;
                    }
                    try {
                        Thread.sleep(waitInMilliseconds);
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                return solutionVersionArn;
            }
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateSolutionVersiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionVersionCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution version parameters.
```

```
export const solutionVersionParam = {
  solutionArn: "SOLUTION_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionVersionCommand(solutionVersionParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateSolutionVersiona Referência AWS SDK para JavaScript da API.

Use **`DeleteCampaign`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteCampaign`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteSpecificCampaign(PersonalizeClient
personalizeClient, String campaignArn) {
       try {
           DeleteCampaignRequest campaignRequest =
DeleteCampaignRequest.builder()
                   .campaignArn(campaignArn)
```

```
                    .build();

            personalizeClient.deleteCampaign(campaignRequest);
            System.out.println("Delete request sent successfully.");
        } catch (PersonalizeException e) {
            System.err.println("Error deleting campaign: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteCampaigna Referência AWS SDK for Java 2.x da API.

Use **`DeleteEventTracker`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteEventTracker`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteEventTracker(PersonalizeClient personalizeClient,
 String eventTrackerArn) {
        try {
            DeleteEventTrackerRequest deleteEventTrackerRequest =
 DeleteEventTrackerRequest.builder()
                    .eventTrackerArn(eventTrackerArn)
                    .build();

            int status =
 personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().status
```

```
            System.out.println("Status code:" + status);

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteEventTrackera Referência AWS SDK for Java 2.x da API.

Use **`DeleteSolution`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteSolution`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

        try {
            DeleteSolutionRequest solutionRequest =
DeleteSolutionRequest.builder()
                    .solutionArn(solutionArn)
                    .build();

            personalizeClient.deleteSolution(solutionRequest);
            System.out.println("Done");

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
```

- Para obter detalhes da API, consulte DeleteSolutiona Referência AWS SDK for Java 2.x da API.

Use **`DescribeCampaign`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeCampaign`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeSpecificCampaign(PersonalizeClient
personalizeClient, String campaignArn) {

       try {
           DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                   .campaignArn(campaignArn)
                   .build();

           DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
           Campaign myCampaign = campaignResponse.campaign();
           System.out.println("The Campaign name is " + myCampaign.name());
           System.out.println("The Campaign status is " + myCampaign.status());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeCampaigna Referência AWS SDK for Java 2.x da API.

Use **`DescribeRecipe`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeRecipe`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeSpecificRecipe(PersonalizeClient
personalizeClient, String recipeArn) {

       try {
           DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
                   .recipeArn(recipeArn)
                   .build();

           DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
           System.out.println("The recipe name is " +
recipeResponse.recipe().name());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeRecipea Referência AWS SDK for Java 2.x da API.

Use **`DescribeSolution`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeSolution`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeSpecificSolution(PersonalizeClient
personalizeClient, String solutionArn) {

       try {
           DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
           System.out.println("The Solution name is " +
response.solution().name());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeSolutiona Referência AWS SDK for Java 2.x da API.

Use **`ListCampaigns`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListCampaigns`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listAllCampaigns(PersonalizeClient personalizeClient,
String solutionArn) {

       try {
           ListCampaignsRequest campaignsRequest =
ListCampaignsRequest.builder()
                   .maxResults(10)
                   .solutionArn(solutionArn)
                   .build();

           ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
           List<CampaignSummary> campaigns = response.campaigns();
           for (CampaignSummary campaign : campaigns) {
               System.out.println("Campaign name is : " + campaign.name());
               System.out.println("Campaign ARN is : " +
campaign.campaignArn());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ListCampaignsa Referência AWS SDK for Java 2.x da API.

Use **`ListDatasetGroups`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListDatasetGroups`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listDSGroups(PersonalizeClient personalizeClient) {

       try {
           ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
                   .maxResults(15)
                   .build();

           ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
           List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
           for (DatasetGroupSummary group : groups) {
               System.out.println("The DataSet name is : " + group.name());
               System.out.println("The DataSet ARN is : " +
group.datasetGroupArn());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ListDatasetGroupsa Referência AWS SDK for Java 2.x da API.

Use **`ListRecipes`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListRecipes`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listAllRecipes(PersonalizeClient personalizeClient) {

        try {
            ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
                    .maxResults(15)
                    .build();

            ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
            List<RecipeSummary> recipes = response.recipes();
            for (RecipeSummary recipe : recipes) {
                System.out.println("The recipe ARN is: " + recipe.recipeArn());
                System.out.println("The recipe name is: " + recipe.name());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListRecipesa Referência AWS SDK for Java 2.x da API.

Use **`ListSolutions`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListSolutions`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listAllSolutions(PersonalizeClient personalizeClient,
String datasetGroupArn) {

       try {
           ListSolutionsRequest solutionsRequest =
ListSolutionsRequest.builder()
                   .maxResults(10)
                   .datasetGroupArn(datasetGroupArn)
                   .build();

           ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
           List<SolutionSummary> solutions = response.solutions();
           for (SolutionSummary solution : solutions) {
               System.out.println("The solution ARN is: " +
solution.solutionArn());
               System.out.println("The solution name is: " + solution.name());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ListSolutionsa Referência AWS SDK for Java 2.x da API.

Use **`UpdateCampaign`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UpdateCampaign`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String updateCampaign(PersonalizeClient personalizeClient,
            String campaignArn,
            String solutionVersionArn,
            Integer minProvisionedTPS) {

        try {
            // build the updateCampaignRequest
            UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .solutionVersionArn(solutionVersionArn)
                    .minProvisionedTPS(minProvisionedTPS)
                    .build();

            // update the campaign
            personalizeClient.updateCampaign(updateCampaignRequest);

            DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
            Campaign updatedCampaign = campaignResponse.campaign();

            System.out.println("The Campaign status is " +
updatedCampaign.status());
            return updatedCampaign.status();
```

```
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte UpdateCampaigna Referência AWS SDK for Java 2.x da API.

Exemplos de código para Amazon Personalize Events usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Personalize Events com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Amazon Personalize Events: mais informações sobre o Amazon Personalize Events.
- Referência da API Amazon Personalize Events: detalhes sobre todas as ações disponíveis do Amazon Personalize Events.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon Personalize Events usando AWS SDKs
  - Ações para Amazon Personalize Events usando AWS SDKs

- Use PutEventscom um AWS SDK
- Use PutItemscom um AWS SDK
- Use PutUserscom um AWS SDK

Exemplos básicos para o Amazon Personalize Events usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Personalize Events com os AWS SDKs.

Exemplos

- Ações para Amazon Personalize Events usando AWS SDKs
  - Use PutEventscom um AWS SDK
  - Use PutItemscom um AWS SDK
  - Use PutUserscom um AWS SDK

Ações para Amazon Personalize Events usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Personalize Events com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de API do Amazon Personalize Events.

Exemplos

- Use PutEventscom um AWS SDK
- Use PutItemscom um AWS SDK
- Use PutUserscom um AWS SDK

Use **`PutEvents`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PutEvents`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        public static int putItems(PersonalizeEventsClient
personalizeEventsClient,
                        String datasetArn,
                        String item1Id,
                        String item1PropertyName,
                        String item1PropertyValue,
                        String item2Id,
                        String item2PropertyName,
                        String item2PropertyValue) {

                int responseCode = 0;
                ArrayList<Item> items = new ArrayList<>();

                try {
                        Item item1 = Item.builder()
                                        .itemId(item1Id)
                                        .properties(String.format("{\"%1$s\":
\"%2$s\"}",
                                                        item1PropertyName,
item1PropertyValue))
                                        .build();

                        items.add(item1);

                        Item item2 = Item.builder()
                                        .itemId(item2Id)
                                        .properties(String.format("{\"%1$s\":
\"%2$s\"}",
                                                        item2PropertyName,
item2PropertyValue))
                                        .build();
```

```
                        items.add(item2);

                        PutItemsRequest putItemsRequest =
 PutItemsRequest.builder()
                                        .datasetArn(datasetArn)
                                        .items(items)
                                        .build();

                        responseCode =
 personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
        }
```

- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { PutEventsCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
 "REGION"});

// Convert your UNIX timestamp to a Date.
const sentAtDate = new Date(1613443801 * 1000); // 1613443801 is a testing value.
 Replace it with your sentAt timestamp in UNIX format.
```

```
// Set put events parameters.
const putEventsParam = {
  eventList: [
    /* required */
    {
      eventType: "EVENT_TYPE" /* required */,
      sentAt: sentAtDate /* required, must be a Date with js */,
      eventId: "EVENT_ID" /* optional */,
      itemId: "ITEM_ID" /* optional */,
    },
  ],
  sessionId: "SESSION_ID" /* required */,
  trackingId: "TRACKING_ID" /* required */,
  userId: "USER_ID" /* required */,
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutEventsCommand(putEventsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte PutEventsa Referência AWS SDK para JavaScript da API.

Use **`PutItems`**com um AWS SDK

O código de exemplo a seguir mostra como usar `PutItems`.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { PutItemsCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
 "REGION"});

// Set the put items parameters. For string properties and values, use the \
 character to escape quotes.
const putItemsParam = {
  datasetArn: "DATASET_ARN" /* required */,
  items: [
    /* required */
    {
      itemId: "ITEM_ID" /*  required */,
      properties:
        '{"PROPERTY1_NAME": "PROPERTY1_VALUE", "PROPERTY2_NAME":
 "PROPERTY2_VALUE", "PROPERTY3_NAME": "PROPERTY3_VALUE"}' /* optional */,
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutItemsCommand(putItemsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

- Para obter detalhes da API, consulte PutItemsa Referência AWS SDK para JavaScript da API.

Use **PutUsers**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PutUsers`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        public static int putUsers(PersonalizeEventsClient
personalizeEventsClient,
                        String datasetArn,
                        String user1Id,
                        String user1PropertyName,
                        String user1PropertyValue,
                        String user2Id,
                        String user2PropertyName,
                        String user2PropertyValue) {

                int responseCode = 0;
                ArrayList<User> users = new ArrayList<>();

                try {
                        User user1 = User.builder()
                                        .userId(user1Id)
                                        .properties(String.format("{\"%1$s\":
\"%2$s\"}",
                                                        user1PropertyName,
user1PropertyValue))
                                        .build();
```

```
                        users.add(user1);

                        User user2 = User.builder()
                                        .userId(user2Id)
                                        .properties(String.format("{\"%1$s\":
\"%2$s\"}",
                                                        user2PropertyName,
user2PropertyValue))
                                        .build();

                        users.add(user2);

                        PutUsersRequest putUsersRequest =
PutUsersRequest.builder()
                                        .datasetArn(datasetArn)
                                        .users(users)
                                        .build();

                        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
        }
```

- Para obter detalhes da API, consulte PutUsersa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { PutUsersCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
 "REGION"});

// Set the put users parameters. For string properties and values, use the \
 character to escape quotes.
const putUsersParam = {
  datasetArn: "DATASET_ARN",
  users: [
    {
      userId: "USER_ID",
      properties: '{"PROPERTY1_NAME": "PROPERTY1_VALUE"}',
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutUsersCommand(putUsersParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte PutUsersa Referência AWS SDK para JavaScript da API.

Exemplos de código para o Amazon Personalize Runtime usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Personalize Runtime com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Amazon Personalize Runtime: mais informações sobre o Amazon Personalize Runtime.
- Referência da API Amazon Personalize: detalhes sobre todas as ações disponíveis do Amazon Personalize Runtime.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Amazon Personalize Runtime usando AWS SDKs
  - Ações para o Amazon Personalize Runtime usando AWS SDKs
    - Use GetPersonalizedRankingcom um AWS SDK
    - Use GetRecommendationscom um AWS SDK

Exemplos básicos do Amazon Personalize Runtime usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Personalize Runtime com os AWS SDKs.

Exemplos

- Ações para o Amazon Personalize Runtime usando AWS SDKs
  - Use GetPersonalizedRankingcom um AWS SDK
  - Use GetRecommendationscom um AWS SDK

Ações para o Amazon Personalize Runtime usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Personalize Runtime com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para ver uma lista completa, consulte a Referência da API do Amazon Personalize Runtime.

Exemplos

- Use GetPersonalizedRankingcom um AWS SDK
- Use GetRecommendationscom um AWS SDK

Use **`GetPersonalizedRanking`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetPersonalizedRanking`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
           String campaignArn,
           String userId,
           ArrayList<String> items) {

       try {
           GetPersonalizedRankingRequest rankingRecommendationsRequest =
GetPersonalizedRankingRequest.builder()
                   .campaignArn(campaignArn)
                   .userId(userId)
                   .inputList(items)
                   .build();
```

```
            GetPersonalizedRankingResponse recommendationsResponse =
 personalizeRuntimeClient
                    .getPersonalizedRanking(rankingRecommendationsRequest);
            List<PredictedItem> rankedItems =
 recommendationsResponse.personalizedRanking();
            int rank = 1;
            for (PredictedItem item : rankedItems) {
                System.out.println("Item ranked at position " + rank + "
 details");
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());

 System.out.println("---------------------------------------------");
                rank++;
            }
            return rankedItems;
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte GetPersonalizedRankinga Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { GetPersonalizedRankingCommand } from "@aws-sdk/client-personalize-
runtime";
```

```
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the ranking request parameters.
export const getPersonalizedRankingParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  inputList: ["ITEM_ID_1", "ITEM_ID_2", "ITEM_ID_3", "ITEM_ID_4"],
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetPersonalizedRankingCommand(getPersonalizedRankingParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte GetPersonalizedRanking a Referência AWS SDK para JavaScript da API.

Use **`GetRecommendations`** com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetRecommendations`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obter uma lista de itens recomendados.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .campaignArn(campaignArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();
           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
           }

       } catch (AwsServiceException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Obtenha uma lista de itens recomendados de um recomendador criado em um grupo de conjunto de dados de domínio.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
           String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .recommenderArn(recommenderArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();

           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
           }
       } catch (AwsServiceException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Use um filtro ao solicitar recomendações.

```
   public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
           String campaignArn,
           String userId,
           String filterArn,
           String parameter1Name,
           String parameter1Value1,
           String parameter1Value2,
           String parameter2Name,
           String parameter2Value) {

       try {
```

```
            Map<String, String> filterValues = new HashMap<>();

            filterValues.put(parameter1Name, String.format("\"%1$s\",\"%2$s\"",
                    parameter1Value1, parameter1Value2));
            filterValues.put(parameter2Name, String.format("\"%1$s\"",
                    parameter2Value));

            GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                    .campaignArn(campaignArn)
                    .numResults(20)
                    .userId(userId)
                    .filterArn(filterArn)
                    .filterValues(filterValues)
                    .build();

            GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                    .getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();

            for (PredictedItem item : items) {
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
            }
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte GetRecommendationsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";

import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Obtenha recomendações com um filtro (grupo de conjunto de dados personalizados).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  recommenderArn: "RECOMMENDER_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Obtenha recomendações filtradas de um recomendador criado em um grupo de conjunto de dados de domínio.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here:
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
```

```
  filterArn: "FILTER_ARN" /* required to filter recommendations */,
  filterValues: {
    PROPERTY:
      '"VALUE"' /* Only required if your filter has a placeholder parameter */,
  },
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte GetRecommendationsa Referência AWS SDK para JavaScript da API.

Exemplos de código para o Amazon Pinpoint usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Pinpoint com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Pinpoint: mais informações sobre o Amazon Pinpoint.
- Referência da API Amazon Pinpoint: detalhes sobre todas as ações disponíveis do Amazon Pinpoint.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon Pinpoint AWS SDKs
  - Ações para o Amazon Pinpoint usando AWS SDKs
    - Use CreateAppcom um AWS SDK ou CLI
    - Use CreateCampaigncom um AWS SDK
    - Use CreateExportJobcom um AWS SDK
    - Use CreateImportJobcom um AWS SDK
    - Use CreateSegmentcom um AWS SDK
    - Use DeleteAppcom um AWS SDK ou CLI
    - Use DeleteEndpointcom um AWS SDK
    - Use GetEndpointcom um AWS SDK ou CLI
    - Use GetSegmentscom um AWS SDK
    - Use GetSmsChannelcom um AWS SDK ou CLI
    - Use GetUserEndpointscom um AWS SDK
    - Use SendMessagescom um AWS SDK ou CLI
    - Use UpdateEndpointcom um AWS SDK
- Cenários para o uso do Amazon Pinpoint AWS SDKs
  - Conceitos básicos das notificações push

Exemplos básicos para o uso do Amazon Pinpoint AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Pinpoint com SDKs da AWS .

Exemplos

- Ações para o Amazon Pinpoint usando AWS SDKs
  - Use CreateAppcom um AWS SDK ou CLI

- Use CreateCampaigncom um AWS SDK
- Use CreateExportJobcom um AWS SDK
- Use CreateImportJobcom um AWS SDK
- Use CreateSegmentcom um AWS SDK
- Use DeleteAppcom um AWS SDK ou CLI
- Use DeleteEndpointcom um AWS SDK
- Use GetEndpointcom um AWS SDK ou CLI
- Use GetSegmentscom um AWS SDK
- Use GetSmsChannelcom um AWS SDK ou CLI
- Use GetUserEndpointscom um AWS SDK
- Use SendMessagescom um AWS SDK ou CLI
- Use UpdateEndpointcom um AWS SDK

Ações para o Amazon Pinpoint usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Pinpoint com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Pinpoint e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Amazon Pinpoint AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Pinpoint.

Exemplos

- Use CreateAppcom um AWS SDK ou CLI
- Use CreateCampaigncom um AWS SDK
- Use CreateExportJobcom um AWS SDK
- Use CreateImportJobcom um AWS SDK
- Use CreateSegmentcom um AWS SDK
- Use DeleteAppcom um AWS SDK ou CLI
- Use DeleteEndpointcom um AWS SDK

- Use GetEndpointcom um AWS SDK ou CLI
- Use GetSegmentscom um AWS SDK
- Use GetSmsChannelcom um AWS SDK ou CLI
- Use GetUserEndpointscom um AWS SDK
- Use SendMessagescom um AWS SDK ou CLI
- Use UpdateEndpointcom um AWS SDK

Use **`CreateApp`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateApp`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos das notificações push

CLI

AWS CLI

Exemplo 1: como criar uma aplicação

O exemplo de `create-app` a seguir cria uma aplicação (projeto).

```
aws pinpoint create-app \
    --create-application-request Name=ExampleCorp
```

Saída:

```
{
    "ApplicationResponse": {
        "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",
        "Id": "810c7aab86d42fb2b56c8c966example",
        "Name": "ExampleCorp",
        "tags": {}
    }
}
```

Exemplo 2: como criar uma aplicação marcada

O exemplo de `create-app` a seguir cria uma aplicação (projeto) e associa uma tag (chave e valor) a ela.

```
aws pinpoint create-app \
    --create-application-request Name=ExampleCorp,tags={"Stack"="Test"}
```

Saída:

```
{
    "ApplicationResponse": {
        "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",
        "Id": "810c7aab86d42fb2b56c8c966example",
        "Name": "ExampleCorp",
        "tags": {
            "Stack": "Test"
        }
    }
}
```

- Para obter detalhes da API, consulte CreateAppem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = """

                 Usage:  <appName>

                 Where:
                  appName - The name of the application to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String appName = args[0];
        System.out.println("Creating an application with name: " + appName);

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String appID = createApplication(pinpoint, appName);
        System.out.println("App ID is: " + appID);
        pinpoint.close();
    }

    public static String createApplication(PinpointClient pinpoint, String
 appName) {
        try {
            CreateApplicationRequest appRequest =
 CreateApplicationRequest.builder()
                    .name(appName)
                    .build();
```

```
            CreateAppRequest request = CreateAppRequest.builder()
                    .createApplicationRequest(appRequest)
                    .build();

            CreateAppResponse result = pinpoint.createApp(request);
            return result.applicationResponse().id();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateApp a Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
```

```
    }
}
```

- Para obter detalhes da API, consulte a CreateAppreferência da API AWS SDK for Kotlin.

Use **`CreateCampaign`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateCampaign`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma campanha.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class CreateCampaign {
    public static void main(String[] args) {

        final String usage = """

                Usage:   <appId> <segmentId>

                Where:
                  appId - The ID of the application to create the campaign in.
                  segmentId - The ID of the segment to create the campaign from.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String segmentId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPinCampaign(pinpoint, appId, segmentId);
        pinpoint.close();
    }

    public static void createPinCampaign(PinpointClient pinpoint, String appId,
 String segmentId) {
        CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
        System.out.println("Campaign " + result.name() + " created.");
        System.out.println(result.description());
    }

    public static CampaignResponse createCampaign(PinpointClient client, String
 appID, String segmentID) {

        try {
            Schedule schedule = Schedule.builder()
                    .startTime("IMMEDIATE")
                    .build();

            Message defaultMessage = Message.builder()
```

```
                    .action(Action.OPEN_APP)
                    .body("My message body.")
                    .title("My message title.")
                    .build();

            MessageConfiguration messageConfiguration =
 MessageConfiguration.builder()
                    .defaultMessage(defaultMessage)
                    .build();

            WriteCampaignRequest request = WriteCampaignRequest.builder()
                    .description("My description")
                    .schedule(schedule)
                    .name("MyCampaign")
                    .segmentId(segmentID)
                    .messageConfiguration(messageConfiguration)
                    .build();

            CreateCampaignResponse result =
 client.createCampaign(CreateCampaignRequest.builder()
                    .applicationId(appID)
                    .writeCampaignRequest(request).build());

            System.out.println("Campaign ID: " + result.campaignResponse().id());
            return result.campaignResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
}
```

- Para obter detalhes da API, consulte CreateCampaigna Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val scheduleOb =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessageOb =
        Message {
            action = Action.OpenApp
            body = "My message body"
            title = "My message title"
        }

    val messageConfigurationOb =
        MessageConfiguration {
            defaultMessage = defaultMessageOb
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = scheduleOb
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfigurationOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
```

```
            pinpoint.createCampaign(
                CreateCampaignRequest {
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- Para obter detalhes da API, consulte a CreateCampaignreferência da API AWS SDK for Kotlin.

Use **`CreateExportJob`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateExportJob`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exportar um endpoint.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
```

```
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-
export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = """

                This program performs the following steps:

                1. Exports the endpoints to an Amazon S3 bucket.
                2. Downloads the exported endpoints files from Amazon S3.
                3. Parses the endpoints files to obtain the endpoint IDs and
 prints them.
                Usage: ExportEndpoints <applicationId> <s3BucketName>
 <iamExportRoleArn> <path>

                Where:
```

```
                 applicationId - The ID of the Amazon Pinpoint application that
has the endpoint.
                 s3BucketName - The name of the Amazon S3 bucket to export the
JSON file to.\s
                 iamExportRoleArn - The ARN of an IAM role that grants Amazon
Pinpoint write permissions to the S3 bucket.  path - The path where the files
downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).
              """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String applicationId = args[0];
       String s3BucketName = args[1];
       String iamExportRoleArn = args[2];
       String path = args[3];
       System.out.println("Deleting an application with ID: " + applicationId);

       Region region = Region.US_EAST_1;
       PinpointClient pinpoint = PinpointClient.builder()
               .region(region)
               .build();

       S3Client s3Client = S3Client.builder()
               .region(region)
               .build();

       exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
       pinpoint.close();
       s3Client.close();
   }

   public static void exportAllEndpoints(PinpointClient pinpoint,
           S3Client s3Client,
           String applicationId,
           String s3BucketName,
           String path,
           String iamExportRoleArn) {

       try {
```

```
            List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
 s3BucketName, iamExportRoleArn,
                    applicationId);
            List<String> endpointFileKeys = objectKeys.stream().filter(o ->
 o.endsWith(".gz"))
                    .collect(Collectors.toList());
            downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> exportEndpointsToS3(PinpointClient pinpoint,
 S3Client s3Client, String s3BucketName,
            String iamExportRoleArn, String applicationId) {

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
        String endpointsKeyPrefix = "exports/" + applicationId + "_" +
 dateFormat.format(new Date());
        String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
 "/";
        List<String> objectKeys = new ArrayList<>();
        String key;

        try {
            // Defines the export job that Amazon Pinpoint runs.
            ExportJobRequest jobRequest = ExportJobRequest.builder()
                    .roleArn(iamExportRoleArn)
                    .s3UrlPrefix(s3UrlPrefix)
                    .build();

            CreateExportJobRequest exportJobRequest =
 CreateExportJobRequest.builder()
                    .applicationId(applicationId)
                    .exportJobRequest(jobRequest)
                    .build();

            System.out.format("Exporting endpoints from Amazon Pinpoint
 application %s to Amazon S3 " +
                    "bucket %s . . .\n", applicationId, s3BucketName);
```

```
            CreateExportJobResponse exportResult =
 pinpoint.createExportJob(exportJobRequest);
            String jobId = exportResult.exportJobResponse().id();
            System.out.println(jobId);
            printExportJobStatus(pinpoint, applicationId, jobId);

            ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
                    .bucket(s3BucketName)
                    .prefix(endpointsKeyPrefix)
                    .build();

            // Create a list of object keys.
            ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
            List<S3Object> objects = v2Response.contents();
            for (S3Object object : objects) {
                key = object.key();
                objectKeys.add(key);
            }

            return objectKeys;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void printExportJobStatus(PinpointClient pinpointClient,
            String applicationId,
            String jobId) {

        GetExportJobResponse getExportJobResult;
        String status;

        try {
            // Checks the job status until the job completes or fails.
            GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
                    .jobId(jobId)
                    .applicationId(applicationId)
                    .build();

            do {
```

```
                getExportJobResult =
pinpointClient.getExportJob(exportJobRequest);
                status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
                System.out.format("Export job %s . . .\n", status);
                TimeUnit.SECONDS.sleep(3);

            } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

            if (status.equals("COMPLETED")) {
                System.out.println("Finished exporting endpoints.");
            } else {
                System.err.println("Failed to export endpoints.");
                System.exit(1);
            }

        } catch (PinpointException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Download files from an Amazon S3 bucket and write them to the path
location.
    public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

        String newPath;
        try {
            for (String key : objectKeys) {
                GetObjectRequest objectRequest = GetObjectRequest.builder()
                        .bucket(s3BucketName)
                        .key(key)
                        .build();

                ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
                byte[] data = objectBytes.asByteArray();

                // Write the data to a local file.
                String fileSuffix = new
SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
                newPath = path + fileSuffix + ".gz";
                File myFile = new File(newPath);
```

```
                OutputStream os = new FileOutputStream(myFile);
                os.write(data);
            }
            System.out.println("Download finished.");

        } catch (S3Exception | NullPointerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateExportJoba Referência AWS SDK for Java 2.x da API.

Use **`CreateImportJob`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateImportJob`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importar um segmento.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <bucket> <key> <roleArn>\s

                Where:
                  appId - The application ID to create a segment for.
                  bucket - The name of the Amazon S3 bucket that contains the
 segment definitons.
                  key - The key of the S3 object.
                  roleArn - ARN of the role that allows Amazon
 Pinpoint to access S3. You need to set trust management for this
 to work. See https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_policies_elements_principal.html
                  """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
 key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
 import job status.");
```

```
        System.out.println("See application " + response.jobStatus() + " for
 import job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
            String appId,
            String bucket,
            String key,
            String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                    .defineSegment(true)
                    .registerEndpoints(true)
                    .roleArn(roleArn)
                    .format(Format.JSON)
                    .s3Url("s3://" + bucket + "/" + key)
                    .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                    .importJobRequest(importRequest)
                    .applicationId(appId)
                    .build();

            CreateImportJobResponse jobResponse =
 client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Para obter detalhes da API, consulte CreateImportJoba Referência AWS SDK for Java 2.x da API.

Use **`CreateSegment`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateSegment`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateSegment {
        public static void main(String[] args) {
                final String usage = """
```

```
                                Usage:   <appId>

                                Where:
                                  appId - The application ID to create a segment
for.

                                """;

                if (args.length != 1) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String appId = args[0];
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                SegmentResponse result = createSegment(pinpoint, appId);
                System.out.println("Segment " + result.name() + " created.");
                System.out.println(result.segmentType());
                pinpoint.close();
        }

        public static SegmentResponse createSegment(PinpointClient client, String
appId) {
                try {
                        Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();
                        segmentAttributes.put("Team",
AttributeDimension.builder()
                                        .attributeType(AttributeType.INCLUSIVE)
                                        .values("Lakers")
                                        .build());

                        RecencyDimension recencyDimension =
RecencyDimension.builder()
                                        .duration("DAY_30")
                                        .recencyType("ACTIVE")
                                        .build();

                        SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
```

```
                                        .recency(recencyDimension)
                                        .build();

                        SegmentDemographics segmentDemographics =
SegmentDemographics
                                        .builder()
                                        .build();

                        SegmentLocation segmentLocation = SegmentLocation
                                        .builder()
                                        .build();

                        SegmentDimensions dimensions = SegmentDimensions
                                        .builder()
                                        .attributes(segmentAttributes)
                                        .behavior(segmentBehaviors)
                                        .demographic(segmentDemographics)
                                        .location(segmentLocation)
                                        .build();

                        WriteSegmentRequest writeSegmentRequest =
WriteSegmentRequest.builder()
                                        .name("MySegment")
                                        .dimensions(dimensions)
                                        .build();

                        CreateSegmentRequest createSegmentRequest =
CreateSegmentRequest.builder()
                                        .applicationId(appId)
                                        .writeSegmentRequest(writeSegmentRequest)
                                        .build();

                        CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
                        System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
                        System.out.println("Done");
                        return createSegmentResult.segmentResponse();

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
                return null;
```

```
        }
}
```

- Para obter detalhes da API, consulte CreateSegmenta Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {
    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts =
        AttributeDimension {
            attributeType = AttributeType.Inclusive
            values = myList
        }

    segmentAttributes["Team"] = atts
    val recencyDimension =
        RecencyDimension {
            duration = Duration.fromValue("DAY_30")
            recencyType = RecencyType.fromValue("ACTIVE")
        }

    val segmentBehaviors =
        SegmentBehaviors {
            recency = recencyDimension
        }

    val segmentLocation = SegmentLocation {}
```

```
    val dimensionsOb =
        SegmentDimensions {
            attributes = segmentAttributes
            behavior = segmentBehaviors
            demographic = SegmentDemographics {}
            location = segmentLocation
        }

    val writeSegmentRequestOb =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensionsOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequestOb
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- Para obter detalhes da API, consulte a CreateSegmentreferência da API AWS SDK for Kotlin.

Use **`DeleteApp`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteApp`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos das notificações push

CLI

AWS CLI

Como excluir uma aplicação

O exemplo de `delete-app` a seguir exclui uma aplicação (projeto).

```
aws pinpoint delete-app \
    --application-id 810c7aab86d42fb2b56c8c966example
```

Saída:

```
{
    "ApplicationResponse": {
        "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",
        "Id": "810c7aab86d42fb2b56c8c966example",
        "Name": "ExampleCorp",
        "tags": {}
    }
}
```

- Para obter detalhes da API, consulte DeleteAppem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir um aplicativo.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
```

```
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                 appId - The ID of the application to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        System.out.println("Deleting an application with ID: " + appId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinApp(pinpoint, appId);
        System.out.println("Done");
        pinpoint.close();
    }

    public static void deletePinApp(PinpointClient pinpoint, String appId) {
        try {
            DeleteAppRequest appRequest = DeleteAppRequest.builder()
                    .applicationId(appId)
                    .build();
```

```
            DeleteAppResponse result = pinpoint.deleteApp(appRequest);
            String appName = result.applicationResponse().name();
            System.out.println("Application " + appName + " has been deleted.");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteApp a Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAppreferência da API AWS SDK for Kotlin.

Use **`DeleteEndpoint`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteEndpoint`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir um endpoint

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appName> <endpointId >

                Where:
                  appId - The id of the application to delete.
                  endpointId - The id of the endpoint to delete.
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinEncpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    public static void deletePinEncpoint(PinpointClient pinpoint, String appId,
 String endpointId) {
        try {
            DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
            String id = result.endpointResponse().id();
            System.out.println("The deleted endpoint id  " + id);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte DeleteEndpointa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- Para obter detalhes da API, consulte a DeleteEndpointreferência da API AWS SDK for Kotlin.

Use **`GetEndpoint`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetEndpoint`.

CLI

AWS CLI

Para recuperar informações sobre as configurações e os atributos de um endpoint específico de uma aplicação

O exemplo de `get-endpoint` a seguir recupera informações sobre as configurações e os atributos de um endpoint específico de uma aplicação.

```
aws pinpoint get-endpoint \
    --application-id 611e3e3cdd47474c9c1399a505665b91 \
    --endpoint-id testendpoint \
    --region us-east-1
```

Saída:

```
{
    "EndpointResponse": {
        "Address": "+11234567890",
        "ApplicationId": "611e3e3cdd47474c9c1399a505665b91",
        "Attributes": {},
        "ChannelType": "SMS",
        "CohortId": "63",
        "CreationDate": "2019-01-28T23:55:11.534Z",
        "EffectiveDate": "2021-08-06T00:04:51.763Z",
        "EndpointStatus": "ACTIVE",
        "Id": "testendpoint",
        "Location": {
            "Country": "USA"
        },
        "Metrics": {
            "SmsDelivered": 1.0
        },
        "OptOut": "ALL",
        "RequestId": "a204b1f2-7e26-48a7-9c80-b49a2143489d",
        "User": {
            "UserAttributes": {
                "Age": [
                    "24"
                ]
            },
        "UserId": "testuser"
        }
    }
}
```

- Para obter detalhes da API, consulte GetEndpointem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <endpoint>

                Where:
                  appId - The ID of the application to delete.
                  endpoint - The ID of the endpoint.\s
                  """;

        if (args.length != 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }

    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String
 appId, String endpoint) {
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpoint)
                    .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();

            // Uses the Google Gson library to pretty print the endpoint JSON.
            Gson gson = new GsonBuilder()
                    .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                    .setPrettyPrinting()
                    .create();

            String endpointJson = gson.toJson(endResponse);
            System.out.println(endpointJson);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte GetEndpointa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.getEndpoint(
                GetEndpointRequest {
                    applicationId = appId
                    endpointId = endpoint
                },
            )
        val endResponse = result.endpointResponse

        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson =
            GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create()

        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

- Para obter detalhes da API, consulte a GetEndpointreferência da API AWS SDK for Kotlin.

Use **`GetSegments`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetSegments`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar segmentos.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId>

                Where:
                  appId - The ID of the application that contains a segment.

                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSegs(pinpoint, appId);
        pinpoint.close();
    }

    public static void listSegs(PinpointClient pinpoint, String appId) {
        try {
            GetSegmentsRequest request = GetSegmentsRequest.builder()
                    .applicationId(appId)
                    .build();

            GetSegmentsResponse response = pinpoint.getSegments(request);
            List<SegmentResponse> segments = response.segmentsResponse().item();
            for (SegmentResponse segment : segments) {
                System.out
                        .println("Segement " + segment.id() + " " +
 segment.name() + " " + segment.lastModifiedDate());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetSegmentsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listSegs(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetSegmentsreferência da API AWS SDK for Kotlin.

Use **`GetSmsChannel`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetSmsChannel`.

CLI

AWS CLI

Para recuperar informações sobre o status e as configurações do canal de SMS de uma aplicação

O exemplo de `get-sms-channel` a seguir recupera o status e as configurações do canal de SMS de uma aplicação.

```
aws pinpoint get-sms-channel \
    --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \
    --region us-east-1
```

Saída:

```
{
    "SMSChannelResponse": {
        "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
        "CreationDate": "2019-10-08T18:39:18.511Z",
        "Enabled": true,
        "Id": "sms",
        "IsArchived": false,
        "LastModifiedDate": "2019-10-08T18:39:18.511Z",
        "Platform": "SMS",
        "PromotionalMessagesPerSecond": 20,
        "TransactionalMessagesPerSecond": 20,
        "Version": 1
    }
}
```

- Para obter detalhes da API, consulte GetSmsChannelem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = """

                Usage: CreateChannel <appId>

                Where:
                  appId - The name of the application whose channel is updated.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        SMSChannelResponse getResponse = getSMSChannel(pinpoint, appId);
        toggleSmsChannel(pinpoint, appId, getResponse);
        pinpoint.close();
    }

    private static SMSChannelResponse getSMSChannel(PinpointClient client, String
 appId) {
        try {
            GetSmsChannelRequest request = GetSmsChannelRequest.builder()
                    .applicationId(appId)
                    .build();
```

```
            SMSChannelResponse response =
 client.getSmsChannel(request).smsChannelResponse();
            System.out.println("Channel state is " + response.enabled());
            return response;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void toggleSmsChannel(PinpointClient client, String appId,
 SMSChannelResponse getResponse) {
        boolean enabled = !getResponse.enabled();
        try {
            SMSChannelRequest request = SMSChannelRequest.builder()
                    .enabled(enabled)
                    .build();

            UpdateSmsChannelRequest updateRequest =
 UpdateSmsChannelRequest.builder()
                    .smsChannelRequest(request)
                    .applicationId(appId)
                    .build();

            UpdateSmsChannelResponse result =
 client.updateSmsChannel(updateRequest);
            System.out.println("Channel state: " +
 result.smsChannelResponse().enabled());

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetSmsChannela Referência AWS SDK for Java 2.x da API.

Use **GetUserEndpoints**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetUserEndpoints`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <applicationId> <userId>

                Where:
                   applicationId - The ID of the Amazon Pinpoint application that
 has the endpoint.
                   userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String applicationId = args[0];
        String userId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllEndpoints(pinpoint, applicationId, userId);
        pinpoint.close();
    }

    public static void listAllEndpoints(PinpointClient pinpoint,
            String applicationId,
            String userId) {

        try {
            GetUserEndpointsRequest endpointsRequest =
 GetUserEndpointsRequest.builder()
                    .userId(userId)
                    .applicationId(applicationId)
                    .build();

            GetUserEndpointsResponse response =
 pinpoint.getUserEndpoints(endpointsRequest);
            List<EndpointResponse> endpoints =
 response.endpointsResponse().item();

            // Display the results.
            for (EndpointResponse endpoint : endpoints) {
                System.out.println("The channel type is: " +
 endpoint.channelType());
                System.out.println("The address is  " + endpoint.address());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetUserEndpointsa Referência AWS SDK for Java 2.x da API.

Use **`SendMessages`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SendMessages`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos das notificações push

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;
using Microsoft.Extensions.Configuration;

namespace SendEmailMessage;

public class SendEmailMainClass
{
    public static async Task Main(string[] args)
    {
        var configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load test settings from .json file.
```

```
        .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();

        // The AWS Region that you want to use to send the email. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        string region = "us-east-1";

        // The "From" address. This address has to be verified in Amazon
Pinpoint
        // in the region you're using to send email.
        string senderAddress = configuration["SenderAddress"]!;

        // The address on the "To" line. If your Amazon Pinpoint account is in
        // the sandbox, this address also has to be verified.
        string toAddress = configuration["ToAddress"]!;

        // The Amazon Pinpoint project/application ID to use when you send this
message.
        // Make sure that the SMS channel is enabled for the project or
application
        // that you choose.
        string appId = configuration["AppId"]!;

        try
        {
            await SendEmailMessage(region, appId, toAddress, senderAddress);
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
ex.Message);
        }
    }

    public static async Task<MessageResponse> SendEmailMessage(
        string region, string appId, string toAddress, string senderAddress)
    {
        var client = new
AmazonPinpointClient(RegionEndpoint.GetBySystemName(region));

        // The subject line of the email.
        string subject = "Amazon Pinpoint Email test";
```

```
        // The body of the email for recipients whose email clients don't
        // support HTML content.
        string textBody = @"Amazon Pinpoint Email Test (.NET)"
                          + "\n---------------------------------"
                          + "\nThis email was sent using the Amazon Pinpoint API
 using the AWS SDK for .NET.";

        // The body of the email for recipients whose email clients support
        // HTML content.
        string htmlBody = @"<html>"
                          + "\n<head></head>"
                          + "\n<body>"
                          + "\n  <h1>Amazon Pinpoint Email Test (AWS SDK
 for .NET)</h1>"
                          + "\n  <p>This email was sent using the "
                          + "\n    <a href='https://aws.amazon.com/
pinpoint/'>Amazon Pinpoint</a> API "
                          + "\n    using the <a href='https://aws.amazon.com/sdk-
for-net/'>AWS SDK for .NET</a>"
                          + "\n  </p>"
                          + "\n</body>"
                          + "\n</html>";

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        string charset = "UTF-8";

        var sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses = new Dictionary<string, AddressConfiguration>
                {
                    {
                        toAddress,
                        new AddressConfiguration
                        {
                            ChannelType = ChannelType.EMAIL
                        }
                    }
                },
                MessageConfiguration = new DirectMessageConfiguration
```

```
                {
                    EmailMessage = new EmailMessage
                    {
                        FromAddress = senderAddress,
                        SimpleEmail = new SimpleEmail
                        {
                            HtmlPart = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = htmlBody
                            },
                            TextPart = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = textBody
                            },
                            Subject = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = subject
                            }
                        }
                    }
                }
            }
        };
        Console.WriteLine("Sending message...");
        SendMessagesResponse response = await
 client.SendMessagesAsync(sendRequest);
        Console.WriteLine("Message sent!");
        return response.MessageResponse;
    }
}
```

Envie uma mensagem SMS.

```
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;
using Microsoft.Extensions.Configuration;
```

```
namespace SendSmsMessage;

public class SendSmsMessageMainClass
{
    public static async Task Main(string[] args)
    {
        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // The AWS Region that you want to use to send the message. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        string region = "us-east-1";

        // The phone number or short code to send the message from. The phone
 number
        // or short code that you specify has to be associated with your Amazon
 Pinpoint
        // account. For best results, specify long codes in E.164 format.
        string originationNumber = configuration["OriginationNumber"]!;

        // The recipient's phone number.  For best results, you should specify
 the
        // phone number in E.164 format.
        string destinationNumber = configuration["DestinationNumber"]!;

        // The Pinpoint project/ application ID to use when you send this
 message.
        // Make sure that the SMS channel is enabled for the project or
 application
        // that you choose.
        string appId = configuration["AppId"]!;

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        MessageType messageType = MessageType.TRANSACTIONAL;

        // The registered keyword associated with the originating short code.
```

```
        string? registeredKeyword = configuration["RegisteredKeyword"];

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        string? senderId = configuration["SenderId"];

        try
        {
            var response = await SendSmsMessage(region, appId, destinationNumber,
                originationNumber, registeredKeyword, senderId, messageType);
            Console.WriteLine($"Message sent to
 {response.MessageResponse.Result.Count} recipient(s).");
            foreach (var messageResultValue in
                     response.MessageResponse.Result.Select(r => r.Value))
            {
                Console.WriteLine($"{messageResultValue.MessageId} Status:
 {messageResultValue.DeliveryStatus}");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
 ex.Message);
        }
    }

    public static async Task<SendMessagesResponse> SendSmsMessage(
        string region, string appId, string destinationNumber, string
 originationNumber,
        string? keyword, string? senderId, MessageType messageType)
    {

        // The content of the SMS message.
        string message = "This message was sent through Amazon Pinpoint using" +
                         " the AWS SDK for .NET. Reply STOP to opt out.";


        var client = new
 AmazonPinpointClient(RegionEndpoint.GetBySystemName(region));

        SendMessagesRequest sendRequest = new SendMessagesRequest
        {
```

```
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses =
                    new Dictionary<string, AddressConfiguration>
                    {
                        {
                            destinationNumber,
                            new AddressConfiguration { ChannelType =
 ChannelType.SMS }
                        }
                    },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    SMSMessage = new SMSMessage
                    {
                        Body = message,
                        MessageType = MessageType.TRANSACTIONAL,
                        OriginationNumber = originationNumber,
                        SenderId = senderId,
                        Keyword = keyword
                    }
                }
            }
        };
        SendMessagesResponse response = await
 client.SendMessagesAsync(sendRequest);
        return response;
    }
}
```

- Para obter detalhes da API, consulte SendMessagesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como enviar uma mensagem SMS usando o endpoint de uma aplicação

O exemplo de `send-messages` a seguir envia uma mensagem direta a uma aplicação com um endpoint.

```
aws pinpoint send-messages \
    --application-id 611e3e3cdd47474c9c1399a505665b91 \
    --message-request file://myfile.json \
    --region us-west-2
```

Conteúdo de `myfile.json`:

```
{
    "MessageConfiguration": {
        "SMSMessage": {
            "Body": "hello, how are you?"
        }
    },
    "Endpoints": {
        "testendpoint": {}
    }
}
```

Saída:

```
{
    "MessageResponse": {
        "ApplicationId": "611e3e3cdd47474c9c1399a505665b91",
        "EndpointResult": {
            "testendpoint": {
                "Address": "+12345678900",
                "DeliveryStatus": "SUCCESSFUL",
                "MessageId": "itnuqhai5alf1n6ahv3udc05n7hhddr6gb3lq6g0",
                "StatusCode": 200,
                "StatusMessage": "MessageId:
 itnuqhai5alf1n6ahv3udc05n7hhddr6gb3lq6g0"
            }
        },
        "RequestId": "c7e23264-04b2-4a46-b800-d24923f74753"
    }
}
```

Para obter mais informações, consulte Canal de SMS do Amazon Pinpoint no Guia do usuário do Amazon Pinpoint.

- Para obter detalhes da API, consulte SendMessagesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class SendEmailMessage {

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
 content.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS
 SDK for Java 2.x

        """;

        public static void main(String[] args) {
                final String usage = """

                                Usage:    <subject> <appId> <senderAddress>
 <toAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified
 in Amazon Pinpoint in the region you're using to send email\s
               toAddress - The to address. This address has to be verified in
 Amazon Pinpoint in the region you're using to send email\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendEmail(pinpoint, subject, senderAddress, toAddress);
```

```
        System.out.println("Email was sent");
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
 subject, String senderAddress, String toAddress) {
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Envie uma mensagem de e-mail com cópia para outros destinatários.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import  software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS
 SDK for Java 2.x

        """;
    public static void main(String[] args) {
        final String usage = """

            Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified
 in Amazon Pinpoint in the region you're using to send email\s
```

```
               toAddress - The to address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
               ccAddress - The CC address.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        String ccAddress = args[3];

        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ArrayList<String> ccList = new ArrayList<>();
        ccList.add(ccAddress);
        sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses)
{
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
```

```
                .toAddresses(toAddress)
                .ccAddresses(ccAddresses)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            // Handle exception
            e.printStackTrace();
        }
    }
}
```

Envie uma mensagem SMS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendMessage {

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        public static String messageType = "TRANSACTIONAL";

        // The registered keyword associated with the originating short code.
        public static String registeredKeyword = "myKeyword";

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        public static String senderId = "MySenderID";

        public static void main(String[] args) {
                final String usage = """

                                Usage:   <message> <appId> <originationNumber>
 <destinationNumber>\s

                                Where:
                                  message - The body of the message to send.
                                  appId - The Amazon Pinpoint project/application
 ID to use when you send this message.
                                  originationNumber - The phone number or
 short code that you specify has to be associated with your Amazon Pinpoint
 account. For best results, specify long codes in E.164 format (for example,
 +1-555-555-5654).
                                  destinationNumber - The recipient's phone
 number.  For best results, you should specify the phone number in E.164 format
 (for example, +1-555-555-5654).\s
                                  """;

                if (args.length != 4) {
                        System.out.println(usage);
```

```
                        System.exit(1);
                }

                String message = args[0];
                String appId = args[1];
                String originationNumber = args[2];
                String destinationNumber = args[3];
                System.out.println("Sending a message");
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
                pinpoint.close();
        }

        public static void sendSMSMessage(PinpointClient pinpoint, String
message, String appId,
                        String originationNumber,
                        String destinationNumber) {
                try {
                        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
                        AddressConfiguration addConfig =
AddressConfiguration.builder()
                                        .channelType(ChannelType.SMS)
                                        .build();

                        addressMap.put(destinationNumber, addConfig);
                        SMSMessage smsMessage = SMSMessage.builder()
                                        .body(message)
                                        .messageType(messageType)
                                        .originationNumber(originationNumber)
                                        .senderId(senderId)
                                        .keyword(registeredKeyword)
                                        .build();

                        // Create a DirectMessageConfiguration object.
                        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
                                        .smsMessage(smsMessage)
                                        .build();
```

```
                        MessageRequest msgReq = MessageRequest.builder()
                                        .addresses(addressMap)
                                        .messageConfiguration(direct)
                                        .build();

                        // create a SendMessagesRequest object
                        SendMessagesRequest request =
 SendMessagesRequest.builder()
                                        .applicationId(appId)
                                        .messageRequest(msgReq)
                                        .build();

                        SendMessagesResponse response =
 pinpoint.sendMessages(request);
                        MessageResponse msg1 = response.messageResponse();
                        Map map1 = msg1.result();

                        // Write out the result of sendMessage.
                        map1.forEach((k, v) -> System.out.println((k + ":" +
 v)));

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

Envie mensagens SMS em lote.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

```
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
    public static String senderId = "MySenderID";

    public static void main(String[] args) {
        final String usage = """

                Usage:   <message> <appId> <originationNumber>
 <destinationNumber> <destinationNumber1>\s

                Where:
                  message - The body of the message to send.
                  appId - The Amazon Pinpoint project/application ID to use when
 you send this message.
                  originationNumber - The phone number or short code that
 you specify has to be associated with your Amazon Pinpoint account. For best
 results, specify long codes in E.164 format (for example, +1-555-555-5654).
```

```
                 destinationNumber - The recipient's phone number.  For best
results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
                 destinationNumber1 - The second recipient's phone number.  For
best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
               """;

       if (args.length != 5) {
           System.out.println(usage);
           System.exit(1);
       }

       String message = args[0];
       String appId = args[1];
       String originationNumber = args[2];
       String destinationNumber = args[3];
       String destinationNumber1 = args[4];
       System.out.println("Sending a message");
       PinpointClient pinpoint = PinpointClient.builder()
               .region(Region.US_EAST_1)
               .build();

       sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
       pinpoint.close();
   }

   public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                                     String originationNumber,
                                     String destinationNumber, String
destinationNumber1) {
       try {
           Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
           AddressConfiguration addConfig = AddressConfiguration.builder()
                   .channelType(ChannelType.SMS)
                   .build();

           // Add an entry to the Map object for each number to whom you want to
send a
           // message.
           addressMap.put(destinationNumber, addConfig);
```

```
            addressMap.put(destinationNumber1, addConfig);
            SMSMessage smsMessage = SMSMessage.builder()
                    .body(message)
                    .messageType(messageType)
                    .originationNumber(originationNumber)
                    .senderId(senderId)
                    .keyword(registeredKeyword)
                    .build();

            // Create a DirectMessageConfiguration object.
            DirectMessageConfiguration direct =
 DirectMessageConfiguration.builder()
                    .smsMessage(smsMessage)
                    .build();

            MessageRequest msgReq = MessageRequest.builder()
                    .addresses(addressMap)
                    .messageConfiguration(direct)
                    .build();

            // Create a SendMessagesRequest object.
            SendMessagesRequest request = SendMessagesRequest.builder()
                    .applicationId(appId)
                    .messageRequest(msgReq)
                    .build();

            SendMessagesResponse response = pinpoint.sendMessages(request);
            MessageResponse msg1 = response.messageResponse();
            Map map1 = msg1.result();

            // Write out the result of sendMessage.
            map1.forEach((k, v) -> System.out.println((k + ":" + v)));

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SendMessagesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { PinpointClient } from "@aws-sdk/client-pinpoint";
// Set the AWS Region.
const REGION = "us-east-1";
export const pinClient = new PinpointClient({ region: REGION });
```

Envie uma mensagem de e-mail.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

// The FromAddress must be verified in SES.
const fromAddress = "FROM_ADDRESS";
const toAddress = "TO_ADDRESS";
const projectId = "PINPOINT_PROJECT_ID";

// The subject line of the email.
const subject = "Amazon Pinpoint Test (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
const body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
----------------------------------------------------
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
 Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
const body_html = `<html>
<head></head>
```

```
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint Email API</a>
 using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding for the subject line and message body of the email.
const charset = "UTF-8";

const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: fromAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

const run = async () => {
```

```
  try {
    const { MessageResponse } = await pinClient.send(
      new SendMessagesCommand(params),
    );

    if (!MessageResponse) {
      throw new Error("No message response.");
    }

    if (!MessageResponse.Result) {
      throw new Error("No message result.");
    }

    const recipientResult = MessageResponse.Result[toAddress];

    if (recipientResult.StatusCode !== 200) {
      throw new Error(recipientResult.StatusMessage);
    }
    console.log(recipientResult.MessageId);
  } catch (err) {
    console.log(err.message);
  }
};

run();
```

Envie uma mensagem SMS.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

/* The phone number or short code to send the message from. The phone number
 or short code that you specify has to be associated with your Amazon Pinpoint
account. For best results, specify long codes in E.164 format. */
const originationNumber = "SENDER_NUMBER"; //e.g., +1XXXXXXXXXX

// The recipient's phone number.  For best results, you should specify the phone
 number in E.164 format.
const destinationNumber = "RECEIVER_NUMBER"; //e.g., +1XXXXXXXXXX
```

```
// The content of the SMS message.
const message =
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out.";

/*The Amazon Pinpoint project/application ID to use when you send this message.
Make sure that the SMS channel is enabled for the project or application
that you choose.*/
const projectId = "PINPOINT_PROJECT_ID"; //e.g., XXXXXXXX66e4e9986478cXXXXXXXXX

/* The type of SMS message that you want to send. If you plan to send
time-sensitive content, specify TRANSACTIONAL. If you plan to send
marketing-related content, specify PROMOTIONAL.*/
const messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
const registeredKeyword = "myKeyword";

/* The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html.*/

const senderId = "MySenderID";

// Specify the parameters to pass to the API.
const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
```

```
  },
};

const run = async () => {
  try {
    const data = await pinClient.send(new SendMessagesCommand(params));
    console.log(
      `Message sent!
 ${data.MessageResponse.Result[destinationNumber].StatusMessage}`,
    );
  } catch (err) {
    console.log(err);
  }
};
run();
```

- Para obter detalhes da API, consulte SendMessagesa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
"use strict";

const AWS = require("aws-sdk");

// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/
const aws_region = "us-west-2";

// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
const senderAddress = "sender@example.com";
```

```
// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
var toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
const appId = "ce796be37f32f178af652b26eexample";

// The subject line of the email.
var subject = "Amazon Pinpoint (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----------------------------------------------------
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
 Node.js.
For more information, see https:\/\/aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint API</a> using
 the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding the you want to use for the subject line and
// message body of the email.
var charset = "UTF-8";

// Specify that you're using a shared credentials file.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });
```

```
//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: appId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: senderAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

//Try to send the email.
pinpoint.sendMessages(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
  } else {
    console.log(
      "Email sent! Message ID: ",
      data["MessageResponse"]["Result"][toAddress]["MessageId"]
    );
```

```
  }
});
```

Envie uma mensagem SMS.

```
"use strict";

var AWS = require("aws-sdk");

// The AWS Region that you want to use to send the message. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/.
var aws_region = "us-east-1";

// The phone number or short code to send the message from. The phone number
// or short code that you specify has to be associated with your Amazon Pinpoint
// account. For best results, specify long codes in E.164 format.
var originationNumber = "+12065550199";

// The recipient's phone number.  For best results, you should specify the
// phone number in E.164 format.
var destinationNumber = "+14255550142";

// The content of the SMS message.
var message =
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out.";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
var applicationId = "ce796be37f32f178af652b26eexample";

// The type of SMS message that you want to send. If you plan to send
// time-sensitive content, specify TRANSACTIONAL. If you plan to send
// marketing-related content, specify PROMOTIONAL.
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
```

```
var registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
var senderId = "MySenderID";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: applicationId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
  },
};

//Try to send the message.
pinpoint.sendMessages(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
```

```
    console.log(err.message);
    // Otherwise, show the unique ID for the message.
  } else {
    console.log(
      "Message sent! " +
        data["MessageResponse"]["Result"][destinationNumber]["StatusMessage"]
    );
  }
});
```

- Para obter detalhes da API, consulte SendMessagesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val body: String =
    """
    Amazon Pinpoint test (AWS SDK for Kotlin)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK
 for Kotlin.
```

```
    """.trimIndent()

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <subject> <appId> <senderAddress> <toAddress>

    Where:
        subject - The email subject to use.
        senderAddress - The from address. This address has to be verified in
 Amazon Pinpoint in the region you're using to send email
        toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
    """

    if (args.size != 3) {
        println(usage)
        exitProcess(0)
    }

    val subject = args[0]
    val senderAddress = args[1]
    val toAddress = args[2]
    sendEmail(subject, senderAddress, toAddress)
}

suspend fun sendEmail(
    subjectVal: String?,
    senderAddress: String,
    toAddressVal: String,
) {
    var content =
        Content {
            data = body
        }

    val messageBody =
        Body {
            text = content
        }

    val subContent =
        Content {
            data = subjectVal
```

```
        }

    val message =
        Message {
            body = messageBody
            subject = subContent
        }

    val destinationOb =
        Destination {
            toAddresses = listOf(toAddressVal)
        }

    val emailContent =
        EmailContent {
            simple = message
        }

    val sendEmailRequest =
        SendEmailRequest {
            fromEmailAddress = senderAddress
            destination = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient.fromEnvironment { region = "us-east-1" }.use
 { pinpointemail ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- Para obter detalhes da API, consulte a SendMessagesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_email_message(
    pinpoint_client,
    app_id,
    sender,
    to_addresses,
    char_set,
    subject,
    html_message,
    text_message,
):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project ID to use when you send this
 message.
    :param sender: The "From" address. This address must be verified in
                   Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
 account
                         is in the sandbox, these addresses must be verified.
    :param char_set: The character encoding to use for the subject line and
 message
```

```
                     body of the email.
    :param subject: The subject line of the email.
    :param html_message: The body of the email for recipients whose email clients
 can
                         display HTML content.
    :param text_message: The body of the email for recipients whose email clients
                         don't support HTML content.
    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                "Addresses": {
                    to_address: {"ChannelType": "EMAIL"} for to_address in
 to_addresses
                },
                "MessageConfiguration": {
                    "EmailMessage": {
                        "FromAddress": sender,
                        "SimpleEmail": {
                            "Subject": {"Charset": char_set, "Data": subject},
                            "HtmlPart": {"Charset": char_set, "Data":
 html_message},
                            "TextPart": {"Charset": char_set, "Data":
 text_message},
                        },
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
    else:
        return {
            to_address: message["MessageId"]
            for to_address, message in response["MessageResponse"]
["Result"].items()
        }


def main():
    app_id = "ce796be37f32f178af652b26eexample"
```

```
    sender = "sender@example.com"
    to_address = "recipient@example.com"
    char_set = "UTF-8"
    subject = "Amazon Pinpoint Test (SDK for Python (Boto3))"
    text_message = """Amazon Pinpoint Test (SDK for Python)
    -------------------------------------
    This email was sent with Amazon Pinpoint using the AWS SDK for Python
 (Boto3).
    For more information, see https://aws.amazon.com/sdk-for-python/
                """
    html_message = """<html>
    <head></head>
    <body>
      <h1>Amazon Pinpoint Test (SDK for Python (Boto3)</h1>
      <p>This email was sent with
        <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> using the
        <a href='https://aws.amazon.com/sdk-for-python/'>
          AWS SDK for Python (Boto3)</a>.</p>
    </body>
    </html>
                """

    print("Sending email.")
    message_ids = send_email_message(
        boto3.client("pinpoint"),
        app_id,
        sender,
        [to_address],
        char_set,
        subject,
        html_message,
        text_message,
    )
    print(f"Message sent! Message IDs: {message_ids}")


if __name__ == "__main__":
    main()
```

Envie uma mensagem SMS.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_sms_message(
    pinpoint_client,
    app_id,
    origination_number,
    destination_number,
    message,
    message_type,
):
    """
    Sends an SMS message with Amazon Pinpoint.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project/application ID to use when you
 send
                   this message. The SMS channel must be enabled for the project
 or
                   application.
    :param destination_number: The recipient's phone number in E.164 format.
    :param origination_number: The phone number to send the message from. This
 phone
                               number must be associated with your Amazon
 Pinpoint
                               account and be in E.164 format.
    :param message: The content of the SMS message.
    :param message_type: The type of SMS message that you want to send. If you
 send
                         time-sensitive content, specify TRANSACTIONAL. If you
 send
                         marketing-related content, specify PROMOTIONAL.
    :return: The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                "Addresses": {destination_number: {"ChannelType": "SMS"}},
                "MessageConfiguration": {
```

```
                    "SMSMessage": {
                        "Body": message,
                        "MessageType": message_type,
                        "OriginationNumber": origination_number,
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response["MessageResponse"]["Result"][destination_number]
["MessageId"]


def main():
    app_id = "ce796be37f32f178af652b26eexample"
    origination_number = "+12065550199"
    destination_number = "+14255550142"
    message = (
        "This is a sample message sent from Amazon Pinpoint by using the AWS SDK
 for "
        "Python (Boto 3)."
    )
    message_type = "TRANSACTIONAL"

    print("Sending SMS message.")
    message_id = send_sms_message(
        boto3.client("pinpoint"),
        app_id,
        origination_number,
        destination_number,
        message,
        message_type,
    )
    print(f"Message sent! Message ID: {message_id}.")


if __name__ == "__main__":
    main()
```

Enviar uma mensagem com um modelo de e-mail existente.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_templated_email_message(
    pinpoint_client, project_id, sender, to_addresses, template_name,
 template_version
):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: The Amazon Pinpoint project ID to use when you send this
 message.
    :param sender: The "From" address. This address must be verified in
                   Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
                         account is in the sandbox, these addresses must be
 verified.
    :param template_name: The name of the email template to use when sending the
 message.
    :param template_version: The version number of the message template.

    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
            MessageRequest={
                "Addresses": {
                    to_address: {"ChannelType": "EMAIL"} for to_address in
 to_addresses
                },
                "MessageConfiguration": {"EmailMessage": {"FromAddress":
 sender}},
                "TemplateConfiguration": {
                    "EmailTemplate": {
                        "Name": template_name,
                        "Version": template_version,
```

```
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
    else:
        return {
            to_address: message["MessageId"]
            for to_address, message in response["MessageResponse"]
["Result"].items()
        }


def main():
    project_id = "296b04b342374fceb661bf494example"
    sender = "sender@example.com"
    to_addresses = ["recipient@example.com"]
    template_name = "My_Email_Template"
    template_version = "1"

    print("Sending email.")
    message_ids = send_templated_email_message(
        boto3.client("pinpoint"),
        project_id,
        sender,
        to_addresses,
        template_name,
        template_version,
    )
    print(f"Message sent! Message IDs: {message_ids}")


if __name__ == "__main__":
    main()
```

Envie uma mensagem de texto com um modelo de SMS existente.

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)


def send_templated_sms_message(
    pinpoint_client,
    project_id,
    destination_number,
    message_type,
    origination_number,
    template_name,
    template_version,
):
    """
    Sends an SMS message to a specific phone number using a pre-defined template.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: An Amazon Pinpoint project (application) ID.
    :param destination_number: The phone number to send the message to.
    :param message_type: The type of SMS message (promotional or transactional).
    :param origination_number: The phone number that the message is sent from.
    :param template_name: The name of the SMS template to use when sending the
 message.
    :param template_version: The version number of the message template.

    :return The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
            MessageRequest={
                "Addresses": {destination_number: {"ChannelType": "SMS"}},
                "MessageConfiguration": {
                    "SMSMessage": {
                        "MessageType": message_type,
                        "OriginationNumber": origination_number,
                    }
                },
                "TemplateConfiguration": {
                    "SMSTemplate": {"Name": template_name, "Version":
 template_version}
                },
            },
        )
```

```
    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response["MessageResponse"]["Result"][destination_number]
["MessageId"]


def main():
    region = "us-east-1"
    origination_number = "+18555550001"
    destination_number = "+14255550142"
    project_id = "7353f53e6885409fa32d07cedexample"
    message_type = "TRANSACTIONAL"
    template_name = "My_SMS_Template"
    template_version = "1"
    message_id = send_templated_sms_message(
        boto3.client("pinpoint", region_name=region),
        project_id,
        destination_number,
        message_type,
        origination_number,
        template_name,
        template_version,
    )
    print(f"Message sent! Message ID: {message_id}.")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a SendMessagesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
" Build the addresses map from the list of to_addresses
DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
LOOP AT it_to_addresses INTO DATA(lo_address).
  INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
    key = lo_address->get_value( )
    value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'EMAIL' )
  ) INTO TABLE lt_addresses.
ENDLOOP.

" Send the email message
DATA(lo_result) = lo_ppt->sendmessages(
  iv_applicationid = iv_app_id
  io_messagerequest = NEW /aws1/cl_pptmessagerequest(
    it_addresses = lt_addresses
    io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
      io_emailmessage = NEW /aws1/cl_pptemailmessage(
        iv_fromaddress = iv_sender
        io_simpleemail = NEW /aws1/cl_pptsimpleemail(
          io_subject = NEW /aws1/cl_pptsimpleemailpart(
            iv_charset = iv_char_set
            iv_data = iv_subject
          )
          io_htmlpart = NEW /aws1/cl_pptsimpleemailpart(
            iv_charset = iv_char_set
            iv_data = iv_html_message
          )
          io_textpart = NEW /aws1/cl_pptsimpleemailpart(
            iv_charset = iv_char_set
            iv_data = iv_text_message
          )
        )
```

```
        )
      )
    )
  ).

  " Extract message IDs from response
  DATA(lo_message_response) = lo_result->get_messageresponse( ).
  ot_message_ids = lo_message_response->get_result( ).

  MESSAGE 'Email message sent successfully.' TYPE 'I'.
```

Envie uma mensagem SMS.

```
  " Build the addresses map for the destination number
  DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
  INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
    key = iv_destination_number
    value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'SMS' )
  ) INTO TABLE lt_addresses.

  " Send the SMS message
  DATA(lo_result) = lo_ppt->sendmessages(
    iv_applicationid = iv_app_id
    io_messagerequest = NEW /aws1/cl_pptmessagerequest(
      it_addresses = lt_addresses
      io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
        io_smsmessage = NEW /aws1/cl_pptsmsmessage(
          iv_body = iv_message
          iv_messagetype = iv_message_type
          iv_originationnumber = iv_origination_number
        )
      )
    )
  ).

  " Extract message ID from response
  DATA(lo_message_response) = lo_result->get_messageresponse( ).
  DATA(lt_results) = lo_message_response->get_result( ).
  LOOP AT lt_results INTO DATA(ls_result).
    IF ls_result-key = iv_destination_number.
      ov_message_id = ls_result-value->get_messageid( ).
      EXIT.
```

```
    ENDIF.
  ENDLOOP.

  MESSAGE 'SMS message sent successfully.' TYPE 'I'.
```

Enviar uma mensagem com um modelo de e-mail existente.

```
  " Build the addresses map from the list of to_addresses
  DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
  LOOP AT it_to_addresses INTO DATA(lo_address).
    INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
      key = lo_address->get_value( )
      value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'EMAIL' )
    ) INTO TABLE lt_addresses.
  ENDLOOP.

  " Send the email message using a template
  DATA(lo_result) = lo_ppt->sendmessages(
    iv_applicationid = iv_app_id
    io_messagerequest = NEW /aws1/cl_pptmessagerequest(
      it_addresses = lt_addresses
      io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
        io_emailmessage = NEW /aws1/cl_pptemailmessage(
          iv_fromaddress = iv_sender
        )
      )
      io_templateconfiguration = NEW /aws1/cl_ppttemplateconf(
        io_emailtemplate = NEW /aws1/cl_ppttemplate(
          iv_name = iv_template_name
          iv_version = iv_template_version
        )
      )
    )
  ).

  " Extract message IDs from response
  DATA(lo_message_response) = lo_result->get_messageresponse( ).
  ot_message_ids = lo_message_response->get_result( ).

  MESSAGE 'Templated email message sent successfully.' TYPE 'I'.
```

Envie uma mensagem de texto com um modelo de SMS existente.

```
" Build the addresses map for the destination number
DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
  key = iv_destination_number
  value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'SMS' )
) INTO TABLE lt_addresses.

" Send the SMS message using a template
DATA(lo_result) = lo_ppt->sendmessages(
  iv_applicationid = iv_app_id
  io_messagerequest = NEW /aws1/cl_pptmessagerequest(
    it_addresses = lt_addresses
    io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
      io_smsmessage = NEW /aws1/cl_pptsmsmessage(
        iv_messagetype = iv_message_type
        iv_originationnumber = iv_origination_number
      )
    )
    io_templateconfiguration = NEW /aws1/cl_ppttemplateconf(
      io_smstemplate = NEW /aws1/cl_ppttemplate(
        iv_name = iv_template_name
        iv_version = iv_template_version
      )
    )
  )
).

" Extract message ID from response
DATA(lo_message_response) = lo_result->get_messageresponse( ).
DATA(lt_results) = lo_message_response->get_result( ).
LOOP AT lt_results INTO DATA(ls_result).
  IF ls_result-key = iv_destination_number.
    ov_message_id = ls_result-value->get_messageid( ).
    EXIT.
  ENDIF.
ENDLOOP.

MESSAGE 'Templated SMS message sent successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a SendMessagesreferência da API AWS SDK for SAP ABAP.

Use **`UpdateEndpoint`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UpdateEndpoint`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                  appId - The ID of the application to create an endpoint for.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        EndpointResponse response = createEndpoint(pinpoint, appId);
        System.out.println("Got Endpoint: " + response.id());
        pinpoint.close();
    }

    public static EndpointResponse createEndpoint(PinpointClient client, String
 appId) {
        String endpointId = UUID.randomUUID().toString();
        System.out.println("Endpoint ID: " + endpointId);

        try {
            EndpointRequest endpointRequest = createEndpointRequestData();
            UpdateEndpointRequest updateEndpointRequest =
 UpdateEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .endpointRequest(endpointRequest)
                    .build();
```

```
            UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
            System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

            GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
            System.out.println(getEndpointResponse.endpointResponse().address());

System.out.println(getEndpointResponse.endpointResponse().channelType());

System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());

System.out.println(getEndpointResponse.endpointResponse().requestId());
            System.out.println(getEndpointResponse.endpointResponse().user());

            return getEndpointResponse.endpointResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static EndpointRequest createEndpointRequestData() {
        try {
            List<String> favoriteTeams = new ArrayList<>();
            favoriteTeams.add("Lakers");
            favoriteTeams.add("Warriors");
            HashMap<String, List<String>> customAttributes = new HashMap<>();
            customAttributes.put("team", favoriteTeams);

            EndpointDemographic demographic = EndpointDemographic.builder()
                    .appVersion("1.0")
                    .make("apple")
```

```
                    .model("iPhone")
                    .modelVersion("7")
                    .platform("ios")
                    .platformVersion("10.1.1")
                    .timezone("America/Los_Angeles")
                    .build();

            EndpointLocation location = EndpointLocation.builder()
                    .city("Los Angeles")
                    .country("US")
                    .latitude(34.0)
                    .longitude(-118.2)
                    .postalCode("90068")
                    .region("CA")
                    .build();

            Map<String, Double> metrics = new HashMap<>();
            metrics.put("health", 100.00);
            metrics.put("luck", 75.00);

            EndpointUser user = EndpointUser.builder()
                    .userId(UUID.randomUUID().toString())
                    .build();

            DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); //
Quoted "Z" to indicate UTC, no timezone
                                                                           //
offset
            String nowAsISO = df.format(new Date());

            return EndpointRequest.builder()
                    .address(UUID.randomUUID().toString())
                    .attributes(customAttributes)
                    .channelType("APNS")
                    .demographic(demographic)
                    .effectiveDate(nowAsISO)
                    .location(location)
                    .metrics(metrics)
                    .optOut("NONE")
                    .requestId(UUID.randomUUID().toString())
                    .user(user)
                    .build();

        } catch (PinpointException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Para obter detalhes da API, consulte UpdateEndpointa Referência AWS SDK for Java 2.x da API.

Cenários para o uso do Amazon Pinpoint AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Pinpoint com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Amazon Pinpoint ou combinadas com outras. Serviços da AWS Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos das notificações push

Conceitos básicos das notificações push

O exemplo de código a seguir mostra como:

- Criar uma aplicação.
- Habilitar canais de notificações por push.
- Enviar notificações por push
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail

# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
```

```
cleanup() {
    local exit_code=$?
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
        echo "ERROR: Command failed: $cmd" >&2
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
```

```
}

# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi

    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
    echo ""

    return 0
}

# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
```

```
        echo "jq is not available, using grep for JSON parsing"
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."

    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
        echo "  - mobiletargeting:CreateApp" >&2
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
```

```
    local app_name=$1

    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters"
 >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi

check_json_tools
validate_permissions

# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
```

```
fi

echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi

echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "============================================"
echo "ENABLING FCM (GCM) CHANNEL"
echo "============================================"
echo "Note: This is using a placeholder API key for demonstration purposes only."
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
```

```
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-
demo-only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server
 Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real
 certificates)
echo ""
echo "============================================"
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================"
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --
apns-channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
    --application-id "$APP_ID" \
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
```

```
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-
channel command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "============================================"
echo "CREATING MESSAGE FILES"
echo "============================================"

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
        "key1": "value1",
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
```

```
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF

echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
echo "To send real messages, you would need to replace the sample device tokens
 with actual ones."

# Step 5: Demonstrate how to send messages (this will fail with placeholder
 tokens)
echo ""
echo "============================================"
echo "DEMONSTRATING MESSAGE SENDING"
echo "============================================"
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
```

```
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
echo "Retrieving application details..."

echo "Executing: aws pinpoint get-app --application-id $APP_ID"
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
```

```
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
echo "2. Attempting to enable FCM and APNS channels (with placeholder
 credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
echo "5. Retrieving application details"
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
```

```
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps
  - GetCallerIdentity
  - SendMessages
  - UpdateApnsChannel
  - UpdateGcmChannel

Exemplos de código para Amazon Pinpoint SMS e Voice API usando AWS SDKs

Os exemplos de código a seguir mostram como usar a API de voz e SMS do Amazon Pinpoint com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor da API SMS and Voice do Amazon Pinpoint: mais informações sobre a API SMS and Voice do Amazon Pinpoint.
- Referência da API SMS and Voice do Amazon Pinpoint: detalhes sobre todas as ações disponíveis da API SMS and Voice do Amazon Pinpoint.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Amazon Pinpoint SMS e Voice API usando AWS SDKs
  - Ações para a API Amazon Pinpoint SMS e Voice usando AWS SDKs
    - Use CreateConfigurationSetcom um AWS SDK
    - Use CreateConfigurationSetEventDestinationcom um AWS SDK
    - Use DeleteConfigurationSetcom um AWS SDK
    - Use DeleteConfigurationSetEventDestinationcom um AWS SDK
    - Use GetConfigurationSetEventDestinationscom um AWS SDK
    - Use ListConfigurationSetscom um AWS SDK
    - Use SendVoiceMessagecom um AWS SDK
    - Use UpdateConfigurationSetEventDestinationcom um AWS SDK

Exemplos básicos de Amazon Pinpoint SMS e Voice API usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos da API SMS e voz do Amazon Pinpoint com SDKs da AWS .

Exemplos

- Ações para a API Amazon Pinpoint SMS e Voice usando AWS SDKs
  - Use CreateConfigurationSetcom um AWS SDK
  - Use CreateConfigurationSetEventDestinationcom um AWS SDK

- Use DeleteConfigurationSetcom um AWS SDK
- Use DeleteConfigurationSetEventDestinationcom um AWS SDK
- Use GetConfigurationSetEventDestinationscom um AWS SDK
- Use ListConfigurationSetscom um AWS SDK
- Use SendVoiceMessagecom um AWS SDK
- Use UpdateConfigurationSetEventDestinationcom um AWS SDK

Ações para a API Amazon Pinpoint SMS e Voice usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais da API de voz e SMS do Amazon Pinpoint com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para uma lista completa, consulte a Referência da API de SMS e voz do Amazon Pinpoint.

Exemplos

- Use CreateConfigurationSetcom um AWS SDK
- Use CreateConfigurationSetEventDestinationcom um AWS SDK
- Use DeleteConfigurationSetcom um AWS SDK
- Use DeleteConfigurationSetEventDestinationcom um AWS SDK
- Use GetConfigurationSetEventDestinationscom um AWS SDK
- Use ListConfigurationSetscom um AWS SDK
- Use SendVoiceMessagecom um AWS SDK
- Use UpdateConfigurationSetEventDestinationcom um AWS SDK

Use **`CreateConfigurationSet`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateConfigurationSet`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create a new configuration set
        lo_pps->createconfigurationset(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
        ).

        MESSAGE 'Configuration set created successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsalreadyexistsex INTO DATA(lo_already_exists_ex).
        MESSAGE lo_already_exists_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_already_exists_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppslimitexceededex INTO DATA(lo_limit_exceeded_ex).
        MESSAGE lo_limit_exceeded_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_limit_exceeded_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateConfigurationSetreferência da API AWS SDK for SAP ABAP.

Use **`CreateConfigurationSetEventDestination`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateConfigurationSetEventDestination`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create event destination for the configuration set
        lo_pps->createconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-
event-dest'
          io_eventdestination = io_event_destination
        ).

        MESSAGE 'Event destination created successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsalreadyexistsex INTO DATA(lo_already_exists_ex).
        MESSAGE lo_already_exists_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_already_exists_ex.
      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppslimitexceededex INTO DATA(lo_limit_exceeded_ex).
        MESSAGE lo_limit_exceeded_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_limit_exceeded_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
```

```
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateConfigurationSetEventDestinationreferência da API AWS SDK for SAP ABAP.

Use **`DeleteConfigurationSet`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteConfigurationSet`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Delete the configuration set
        lo_pps->deleteconfigurationset(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
        ).

        MESSAGE 'Configuration set deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
```

```
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteConfigurationSetreferência da API AWS SDK for SAP ABAP.

Use **`DeleteConfigurationSetEventDestination`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteConfigurationSetEventDestination`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Delete the event destination
        lo_pps->deleteconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-
event-dest'
        ).

        MESSAGE 'Event destination deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
```

```
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteConfigurationSetEventDestinationreferência da API AWS SDK for SAP ABAP.

Use **`GetConfigurationSetEventDestinations`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetConfigurationSetEventDestinations`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Get event destinations for the configuration set
        oo_result = lo_pps->getconfseteventdestinations(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
        ).

        " Process the event destinations
        LOOP AT oo_result->get_eventdestinations( ) INTO DATA(lo_event_dest).
          DATA(lv_dest_name) = lo_event_dest->get_name( ).
          DATA(lv_enabled) = lo_event_dest->get_enabled( ).

          MESSAGE |Event destination: { lv_dest_name }, Enabled: { lv_enabled }|
 TYPE 'I'.

          " Check for CloudWatch Logs destination
          DATA(lo_cloudwatch_dest) = lo_event_dest->get_cloudwatchlogsdst( ).
          IF lo_cloudwatch_dest IS NOT INITIAL.
```

```
            DATA(lv_log_group_arn) = lo_cloudwatch_dest->get_loggrouparn( ).
            MESSAGE |  CloudWatch Logs destination: { lv_log_group_arn }| TYPE
 'I'.
          ENDIF.

          " Check for Kinesis Firehose destination
          DATA(lo_firehose_dest) = lo_event_dest->get_kinesisfirehosedst( ).
          IF lo_firehose_dest IS NOT INITIAL.
            DATA(lv_delivery_stream) = lo_firehose_dest-
>get_deliverystreamarn( ).
            MESSAGE |  Kinesis Firehose destination: { lv_delivery_stream }| TYPE
 'I'.
          ENDIF.

          " Check for SNS destination
          DATA(lo_sns_dest) = lo_event_dest->get_snsdestination( ).
          IF lo_sns_dest IS NOT INITIAL.
            DATA(lv_topic_arn) = lo_sns_dest->get_topicarn( ).
            MESSAGE |  SNS destination: { lv_topic_arn }| TYPE 'I'.
          ENDIF.
        ENDLOOP.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetConfigurationSetEventDestinationsreferência da API AWS SDK for SAP ABAP.

Use **`ListConfigurationSets`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListConfigurationSets`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       " List all configuration sets
       oo_result = lo_pps->listconfigurationsets(
         iv_nexttoken = iv_next_token    " Optional: Token for pagination
         iv_pagesize = iv_page_size      " Optional: Number of results per page,
e.g., '10'
       ).

       " Process the configuration sets
       LOOP AT oo_result->get_configurationsets( ) INTO DATA(lo_config_set).
         DATA(lv_config_set_name) = lo_config_set->get_value( ).
         MESSAGE |Configuration set: { lv_config_set_name }| TYPE 'I'.
       ENDLOOP.

       " Check if there are more results
       DATA(lv_next_token) = oo_result->get_nexttoken( ).
       IF lv_next_token IS NOT INITIAL.
         MESSAGE |More results available. Next token: { lv_next_token }| TYPE
'I'.
       ENDIF.

     CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
       MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_bad_request_ex.
     CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
       MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_internal_error_ex.
     CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
       MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_too_many_requests_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListConfigurationSetsreferência da API AWS SDK for SAP ABAP.

Use **SendVoiceMessage**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `SendVoiceMessage`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
 list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";

    // The content of the message. This example uses SSML to customize and
 control
    // certain aspects of the message, such as by adding pauses and changing
    // phonation. The message can't contain any line breaks.
    static final String ssmlMessage = "<speak>This is a test message sent from "
            + "<emphasis>Amazon Pinpoint</emphasis> "
            + "using the <break strength='weak'/>AWS "
            + "SDK for Java. "
            + "<amazon:effect phonation='soft'>Thank "
            + "you for listening.</amazon:effect></speak>";

    public static void main(String[] args) {

        final String usage = """
                Usage:   <originationNumber> <destinationNumber>\s

                Where:
                  originationNumber - The phone number or short code that
 you specify has to be associated with your Amazon Pinpoint account. For best
 results, specify long codes in E.164 format (for example, +1-555-555-5654).
                  destinationNumber - The recipient's phone number.  For best
 results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
        String originationNumber = args[0];
        String destinationNumber = args[1];
```

```
        System.out.println("Sending a voice message");

        // Set the content type to application/json.
        List<String> listVal = new ArrayList<>();
        listVal.add("application/json");
        Map<String, List<String>> values = new HashMap<>();
        values.put("Content-Type", listVal);

        ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
                .headers(values)
                .build();

        PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
                .overrideConfiguration(config2)
                .region(Region.US_EAST_1)
                .build();

        sendVoiceMsg(client, originationNumber, destinationNumber);
        client.close();
    }

    public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
                                    String destinationNumber) {
        try {
            SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                    .languageCode(languageCode)
                    .text(ssmlMessage)
                    .voiceId(voiceName)
                    .build();

            VoiceMessageContent content = VoiceMessageContent.builder()
                    .ssmlMessage(ssmlMessageType)
                    .build();

            SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
                    .destinationPhoneNumber(destinationNumber)
                    .originationPhoneNumber(originationNumber)
                    .content(content)
                    .build();

            client.sendVoiceMessage(voiceMessageRequest);
```

```
            System.out.println("The message was sent successfully.");

        } catch (PinpointSmsVoiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SendVoiceMessagea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"use strict";

var AWS = require("aws-sdk");

// The AWS Region that you want to use to send the voice message. For a list of
// AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see
// https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/
var aws_region = "us-east-1";

// The phone number that the message is sent from. The phone number that you
// specify has to be associated with your Amazon Pinpoint account. For best
 results, you
// should specify the phone number in E.164 format.
var originationNumber = "+12065550110";

// The recipient's phone number. For best results, you should specify the phone
// number in E.164 format.
```

```
var destinationNumber = "+12065550142";

// The language to use when sending the message. For a list of supported
// languages, see https://docs.aws.amazon.com/polly/latest/dg/
SupportedLanguage.html
var languageCode = "en-US";

// The Amazon Polly voice that you want to use to send the message. For a list
// of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
var voiceId = "Matthew";

// The content of the message. This example uses SSML to customize and control
// certain aspects of the message, such as the volume or the speech rate.
// The message can't contain any line breaks.
var ssmlMessage =
  "<speak>" +
  "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> " +
  "using the <break strength='weak'/>AWS SDK for JavaScript in Node.js. " +
  "<amazon:effect phonation='soft'>Thank you for listening." +
  "</amazon:effect>" +
  "</speak>";

// The phone number that you want to appear on the recipient's device. The phone
// number that you specify has to be associated with your Amazon Pinpoint
 account.
var callerId = "+12065550199";

// The configuration set that you want to use to send the message.
var configurationSet = "ConfigSet";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpointsmsvoice = new AWS.PinpointSMSVoice();

var params = {
  CallerId: callerId,
  ConfigurationSetName: configurationSet,
```

```
  Content: {
    SSMLMessage: {
      LanguageCode: languageCode,
      Text: ssmlMessage,
      VoiceId: voiceId,
    },
  },
  DestinationPhoneNumber: destinationNumber,
  OriginationPhoneNumber: originationNumber,
};

//Try to send the message.
pinpointsmsvoice.sendVoiceMessage(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
    // Otherwise, show the unique ID for the message.
  } else {
    console.log("Message sent! Message ID: " + data["MessageId"]);
  }
});
```

- Para obter detalhes da API, consulte SendVoiceMessagea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)


def send_voice_message(
    sms_voice_client,
    origination_number,
    caller_id,
    destination_number,
    language_code,
    voice_id,
    ssml_message,
):
    """
    Sends a voice message using speech synthesis provided by Amazon Polly.

    :param sms_voice_client: A Boto3 PinpointSMSVoice client.
    :param origination_number: The phone number that the message is sent from.
                               The phone number must be associated with your
 Amazon
                               Pinpoint account and be in E.164 format.
    :param caller_id: The phone number that you want to appear on the recipient's
                      device. The phone number must be associated with your
 Amazon
                      Pinpoint account and be in E.164 format.
    :param destination_number: The recipient's phone number. Specify the phone
                               number in E.164 format.
    :param language_code: The language to use when sending the message.
    :param voice_id: The Amazon Polly voice that you want to use to send the
 message.
    :param ssml_message: The content of the message. This example uses SSML to
 control
                         certain aspects of the message, such as the volume and
 the
                         speech rate. The message must not contain line breaks.
    :return: The ID of the message.
    """
    try:
        response = sms_voice_client.send_voice_message(
            DestinationPhoneNumber=destination_number,
            OriginationPhoneNumber=origination_number,
            CallerId=caller_id,
            Content={
                "SSMLMessage": {
```

```
                    "LanguageCode": language_code,
                    "VoiceId": voice_id,
                    "Text": ssml_message,
                }
            },
        )
    except ClientError:
        logger.exception(
            "Couldn't send message from %s to %s.",
            origination_number,
            destination_number,
        )
        raise
    else:
        return response["MessageId"]


def main():
    origination_number = "+12065550110"
    caller_id = "+12065550199"
    destination_number = "+12065550142"
    language_code = "en-US"
    voice_id = "Matthew"
    ssml_message = (
        "<speak>"
        "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "
        "using the <break strength='weak'/>AWS SDK for Python (Boto3). "
        "<amazon:effect phonation='soft'>Thank you for listening."
        "</amazon:effect>"
        "</speak>"
    )
    print(f"Sending voice message from {origination_number} to
 {destination_number}.")
    message_id = send_voice_message(
        boto3.client("pinpoint-sms-voice"),
        origination_number,
        caller_id,
        destination_number,
        language_code,
        voice_id,
        ssml_message,
    )
    print(f"Message sent!\nMessage ID: {message_id}")
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a SendVoiceMessageReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create SSML message type object with voice parameters
        DATA(lo_ssml_message) = NEW /aws1/cl_ppsssmlmessagetype(
          iv_languagecode = iv_language_code    " e.g., 'en-US'
          iv_voiceid = iv_voice_id              " e.g., 'Matthew'
          iv_text = iv_ssml_message             " SSML formatted message text
        ).

        " Create voice message content with the SSML message
        DATA(lo_content) = NEW /aws1/cl_ppsvoicemessagecont(
          io_ssmlmessage = lo_ssml_message
        ).

        " Send the voice message
        DATA(lo_result) = lo_pps->sendvoicemessage(
          iv_originationphonenumber = iv_origination_number  " e.g.,
'+12065550110'
          iv_callerid = iv_caller_id                         " e.g.,
'+12065550199'
          iv_destinationphonenumber = iv_destination_number  " e.g.,
'+12065550142'
          io_content = lo_content
        ).
```

```
        " Retrieve the message ID from the response
        ov_message_id = lo_result->get_messageid( ).

        MESSAGE 'Voice message sent successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendVoiceMessagereferência da API AWS SDK for SAP ABAP.

Use **`UpdateConfigurationSetEventDestination`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UpdateConfigurationSetEventDestination`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Update the event destination
        lo_pps->updateconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
```

```
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-
event-dest'
          io_eventdestination = io_event_destination
        ).

        MESSAGE 'Event destination updated successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateConfigurationSetEventDestinationreferência da API AWS SDK for SAP ABAP.

Exemplos de código para o Amazon Polly usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Polly com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Polly: mais informações sobre o Amazon Polly.
- Referência da API Amazon Polly: detalhes sobre todas as ações disponíveis do Amazon Polly.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon Polly AWS SDKs
  - Ações para o Amazon Polly usando AWS SDKs
    - Use DeleteLexiconcom um AWS SDK ou CLI
    - Use DescribeVoicescom um AWS SDK
    - Use GetLexiconcom um AWS SDK ou CLI
    - Use GetSpeechSynthesisTaskcom um AWS SDK ou CLI
    - Use ListLexiconscom um AWS SDK ou CLI
    - Use ListSpeechSynthesisTaskscom um AWS SDK ou CLI
    - Use PutLexiconcom um AWS SDK ou CLI
    - Use StartSpeechSynthesisTaskcom um AWS SDK ou CLI
    - Use SynthesizeSpeechcom um AWS SDK
- Cenários para o uso do Amazon Polly AWS SDKs
  - Converta texto em fala e volte para texto usando um AWS SDK
  - Crie um aplicativo de sincronização labial com o Amazon Polly usando um AWS SDK
  - Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
  - Introdução à síntese de conversão de texto em fala

Exemplos básicos para o uso do Amazon Polly AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Polly com AWS SDKs.

Exemplos

- Ações para o Amazon Polly usando AWS SDKs
  - Use DeleteLexiconcom um AWS SDK ou CLI
  - Use DescribeVoicescom um AWS SDK

- Use GetLexiconcom um AWS SDK ou CLI
- Use GetSpeechSynthesisTaskcom um AWS SDK ou CLI
- Use ListLexiconscom um AWS SDK ou CLI
- Use ListSpeechSynthesisTaskscom um AWS SDK ou CLI
- Use PutLexiconcom um AWS SDK ou CLI
- Use StartSpeechSynthesisTaskcom um AWS SDK ou CLI
- Use SynthesizeSpeechcom um AWS SDK

Ações para o Amazon Polly usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Polly com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Polly e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Amazon Polly AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Polly.

Exemplos

- Use DeleteLexiconcom um AWS SDK ou CLI
- Use DescribeVoicescom um AWS SDK
- Use GetLexiconcom um AWS SDK ou CLI
- Use GetSpeechSynthesisTaskcom um AWS SDK ou CLI
- Use ListLexiconscom um AWS SDK ou CLI
- Use ListSpeechSynthesisTaskscom um AWS SDK ou CLI
- Use PutLexiconcom um AWS SDK ou CLI
- Use StartSpeechSynthesisTaskcom um AWS SDK ou CLI
- Use SynthesizeSpeechcom um AWS SDK

Use **`DeleteLexicon`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteLexicon`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução à síntese de conversão de texto em fala

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Deletes an existing Amazon Polly lexicon using the AWS SDK for .NET.
/// </summary>
public class DeleteLexicon
{
    public static async Task Main()
    {
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();

        var success = await DeletePollyLexiconAsync(client, lexiconName);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {lexiconName}.");
        }
        else
        {
            Console.WriteLine($"Could not delete {lexiconName}.");
        }
```

```
        }

        /// <summary>
        /// Deletes the named Amazon Polly lexicon.
        /// </summary>
        /// <param name="client">The initialized Amazon Polly client object.</
param>
        /// <param name="lexiconName">The name of the Amazon Polly lexicon to
        /// delete.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeletePollyLexiconAsync(
            AmazonPollyClient client,
            string lexiconName)
        {
            var deleteLexiconRequest = new DeleteLexiconRequest()
            {
                Name = lexiconName,
            };

            var response = await client.DeleteLexiconAsync(deleteLexiconRequest);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte DeleteLexicona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir um léxico

O exemplo de `delete-lexicon` a seguir exclui o léxico especificado.

```
aws polly delete-lexicon \
    --name w3c
```

Este comando não produz saída.

Para obter mais informações, consulte Como usar a DeleteLexicon operação no Amazon Polly Developer Guide.

- Para obter detalhes da API, consulte DeleteLexiconem Referência de AWS CLI Comandos.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ply->deletelexicon( iv_name ).
    MESSAGE 'Lexicon deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_plylexiconnotfoundex.
    MESSAGE 'Lexicon not found.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteLexiconreferência da API AWS SDK for SAP ABAP.

Use **`DescribeVoices`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeVoices`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução à síntese de conversão de texto em fala

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DescribeVoices
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();

        var allVoicesRequest = new DescribeVoicesRequest();
        var enUsVoicesRequest = new DescribeVoicesRequest()
        {
            LanguageCode = "en-US",
        };

        try
        {
            string nextToken;
            do
            {
                var allVoicesResponse = await
client.DescribeVoicesAsync(allVoicesRequest);
                nextToken = allVoicesResponse.NextToken;
                allVoicesRequest.NextToken = nextToken;

                Console.WriteLine("\nAll voices: ");
                allVoicesResponse.Voices.ForEach(voice =>
                {
                    DisplayVoiceInfo(voice);
                });
```

```
                }
                while (nextToken is not null);

                do
                {
                    var enUsVoicesResponse = await
client.DescribeVoicesAsync(enUsVoicesRequest);
                    nextToken = enUsVoicesResponse.NextToken;
                    enUsVoicesRequest.NextToken = nextToken;

                    Console.WriteLine("\nen-US voices: ");
                    enUsVoicesResponse.Voices.ForEach(voice =>
                    {
                        DisplayVoiceInfo(voice);
                    });
                }
                while (nextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception caught: " + ex.Message);
            }
        }

        public static void DisplayVoiceInfo(Voice voice)
        {
            Console.WriteLine($" Name: {voice.Name}\tGender:
{voice.Gender}\tLanguageName: {voice.LanguageName}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeVoicesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                    .languageCode("en-US")
                    .build();
```

```
            DescribeVoicesResponse enUsVoicesResult =
 polly.describeVoices(voicesRequest);
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
 myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeVoicesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
```

```
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def describe_voices(self):
        """
        Gets metadata about available voices.

        :return: The list of voice metadata.
        """
        try:
            response = self.polly_client.describe_voices()
            self.voice_metadata = response["Voices"]
            logger.info("Got metadata about %s voices.",
len(self.voice_metadata))
        except ClientError:
            logger.exception("Couldn't get voice metadata.")
            raise
        else:
            return self.voice_metadata
```

- Para obter detalhes da API, consulte a DescribeVoicesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
```

```
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/
config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts "  #{v.gender}"
    puts
  end
rescue StandardError => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- Para obter detalhes da API, consulte DescribeVoicesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn list_voices(client: &Client) -> Result<(), Error> {
    let resp = client.describe_voices().send().await?;

    println!("Voices:");

    let voices = resp.voices();
    for voice in voices {
        println!("  Name:     {}", voice.name().unwrap_or("No name!"));
```

```
        println!(
            "  Language: {}",
            voice.language_name().unwrap_or("No language!")
        );

        println!();
    }

    println!("Found {} voices", voices.len());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeVoicesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Only pass optional parameters if they have values
        IF iv_engine IS NOT INITIAL AND iv_language IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_engine = iv_engine
            iv_languagecode = iv_language ).
        ELSEIF iv_engine IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_engine = iv_engine ).
        ELSEIF iv_language IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_languagecode = iv_language ).
        ELSE.
          oo_result = lo_ply->describevoices( ).
        ENDIF.
        MESSAGE 'Retrieved voice metadata.' TYPE 'I'.
```

```
    CATCH /aws1/cx_plyinvalidnexttokenex.
      MESSAGE 'The NextToken is invalid.' TYPE 'E'.
    CATCH /aws1/cx_plyservicefailureex.
      MESSAGE 'Service failure occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeVoicesreferência da API AWS SDK for SAP ABAP.

Use **`GetLexicon`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetLexicon`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução à síntese de conversão de texto em fala

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Retrieves information about a specific Amazon Polly lexicon.
/// </summary>
public class GetLexicon
{
    public static async Task Main(string[] args)
    {
```

```
            string lexiconName = "SampleLexicon";

            var client = new AmazonPollyClient();

            await GetPollyLexiconAsync(client, lexiconName);
        }

        public static async Task GetPollyLexiconAsync(AmazonPollyClient client,
 string lexiconName)
        {
            var getLexiconRequest = new GetLexiconRequest()
            {
                Name = lexiconName,
            };

            try
            {
                var response = await client.GetLexiconAsync(getLexiconRequest);
                Console.WriteLine($"Lexicon:\n Name: {response.Lexicon.Name}");
                Console.WriteLine($"Content: {response.Lexicon.Content}");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error: " + ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte GetLexicona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar o conteúdo de um léxico

O exemplo de `get-lexicon` a seguir recupera o conteúdo do léxico de pronúncia especificado.

```
aws polly get-lexicon \
```

```
    --name w3c
```

Saída:

```
{
    "Lexicon": {
        "Content": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<lexicon version=
\"1.0\" \n      xmlns=    \"http://www.w3.org/2005/01/pronunciation-lexicon
\"\n      xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" \n
 xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon \n
 http://www.w3.org/TR/2007/CR-pronunciation-    lexicon-20071212/pls.xsd\"\n
   alphabet=\"ipa\" \n      xml:lang=\"en-US\">\n  <lexeme>\n    <grapheme>W3C</
grapheme>\n        <alias>World Wide Web Consortium</alias>\n  </lexeme>\n</
lexicon>\n",
        "Name": "w3c"
    },
    "LexiconAttributes": {
        "Alphabet": "ipa",
        "LanguageCode": "en-US",
        "LastModified": 1603908910.99,
        "LexiconArn": "arn:aws:polly:us-west-2:880185128111:lexicon/w3c",
        "LexemesCount": 1,
        "Size": 492
    }
}
```

Para obter mais informações, consulte Como usar a GetLexicon operação no Amazon Polly Developer Guide.

- Para obter detalhes da API, consulte GetLexiconem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def get_lexicon(self, name):
        """
        Gets metadata and contents of an existing lexicon.

        :param name: The name of the lexicon to retrieve.
        :return: The retrieved lexicon.
        """
        try:
            response = self.polly_client.get_lexicon(Name=name)
            logger.info("Got lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't get lexicon %s.", name)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a GetLexiconReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ply->getlexicon( iv_name ).
        DATA(lo_lexicon) = oo_result->get_lexicon( ).
        IF lo_lexicon IS BOUND.
          DATA(lv_lex_name) = lo_lexicon->get_name( ).
          MESSAGE |Retrieved lexicon: { lv_lex_name }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetLexiconreferência da API AWS SDK for SAP ABAP.

Use **`GetSpeechSynthesisTask`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetSpeechSynthesisTask`.

CLI

AWS CLI

Como obter informações sobre uma tarefa de síntese de fala

O exemplo de `get-speech-synthesis-task` a seguir recupera informações sobre a tarefa de síntese de fala especificada.

```
aws polly get-speech-synthesis-task \
```

```
    --task-id 70b61c0f-57ce-4715-a247-cae8729dcce9
```

Saída:

```
{
    "SynthesisTask": {
        "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
        "TaskStatus": "completed",
        "OutputUri": "https://s3.us-west-2.amazonaws.com/amzn-s3-demo-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
        "CreationTime": 1603911042.689,
        "RequestCharacters": 1311,
        "OutputFormat": "mp3",
        "TextType": "text",
        "VoiceId": "Joanna"
    }
}
```

Para obter mais informações, consulte Criar arquivos de áudio longos no Guia do desenvolvedor do Amazon Polly.

- Para obter detalhes da API, consulte GetSpeechSynthesisTaskem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
```

```
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def get_speech_synthesis_task(self, task_id):
        """
        Gets metadata about an asynchronous speech synthesis task, such as its
status.

        :param task_id: The ID of the task to retrieve.
        :return: Metadata about the task.
        """
        try:
            response =
self.polly_client.get_speech_synthesis_task(TaskId=task_id)
            task = response["SynthesisTask"]
            logger.info("Got synthesis task. Status is %s.", task["TaskStatus"])
        except ClientError:
            logger.exception("Couldn't get synthesis task %s.", task_id)
            raise
        else:
            return task
```

- Para obter detalhes da API, consulte a GetSpeechSynthesisTaskReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ply->getspeechsynthesistask( iv_task_id ).
        DATA(lo_task) = oo_result->get_synthesistask( ).
        IF lo_task IS BOUND.
          DATA(lv_status) = lo_task->get_taskstatus( ).
          MESSAGE |Task status: { lv_status }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_plyinvalidtaskidex.
        MESSAGE 'Invalid task ID.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plysynthesistsknotf00.
        MESSAGE 'Synthesis task not found.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetSpeechSynthesisTaskreferência da API AWS SDK for SAP ABAP.

Use **`ListLexicons`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListLexicons`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução à síntese de conversão de texto em fala

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
```

```
    using Amazon.Polly;
    using Amazon.Polly.Model;

    /// <summary>
    /// Lists the Amazon Polly lexicons that have been defined. By default,
    /// lists the lexicons that are defined in the same AWS Region as the default
    /// user. To view Amazon Polly lexicons that are defined in a different AWS
    /// Region, supply it as a parameter to the Amazon Polly constructor.
    /// </summary>
    public class ListLexicons
    {
        public static async Task Main()
        {
            var client = new AmazonPollyClient();
            var request = new ListLexiconsRequest();

            try
            {
                Console.WriteLine("All voices: ");

                do
                {
                    var response = await client.ListLexiconsAsync(request);
                    request.NextToken = response.NextToken;

                    response.Lexicons.ForEach(lexicon =>
                    {
                        var attributes = lexicon.Attributes;
                        Console.WriteLine($"Name: {lexicon.Name}");
                        Console.WriteLine($"\tAlphabet: {attributes.Alphabet}");
                        Console.WriteLine($"\tLanguageCode:
{attributes.LanguageCode}");
                        Console.WriteLine($"\tLastModified:
{attributes.LastModified}");
                        Console.WriteLine($"\tLexemesCount:
{attributes.LexemesCount}");
                        Console.WriteLine($"\tLexiconArn:
{attributes.LexiconArn}");
                        Console.WriteLine($"\tSize: {attributes.Size}");
                    });
                }
                while (request.NextToken is not null);
            }
            catch (Exception ex)
```

```
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- Para obter detalhes da API, consulte ListLexiconsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os léxicos

O exemplo de `list-lexicons` a seguir lista os léxicos de pronúncia.

```
aws polly list-lexicons
```

Saída:

```
{
    "Lexicons": [
        {
            "Name": "w3c",
            "Attributes": {
                "Alphabet": "ipa",
                "LanguageCode": "en-US",
                "LastModified": 1603908910.99,
                "LexiconArn": "arn:aws:polly:us-east-2:123456789012:lexicon/w3c",
                "LexemesCount": 1,
                "Size": 492
            }
        }
    ]
}
```

Para obter mais informações, consulte Como usar a ListLexicons operação no Amazon Polly Developer Guide.

- Para obter detalhes da API, consulte ListLexiconsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listLexicons(polly);
        polly.close();
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest =
 ListLexiconsRequest.builder()
                    .build();
```

```
            ListLexiconsResponse listLexiconsResult =
 client.listLexicons(listLexiconsRequest);
            List<LexiconDescription> lexiconDescription =
 listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
 lexDescription.name());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListLexiconsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
```

```
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def list_lexicons(self):
        """
        Lists lexicons in the current account.

        :return: The list of lexicons.
        """
        try:
            response = self.polly_client.list_lexicons()
            lexicons = response["Lexicons"]
            logger.info("Got %s lexicons.", len(lexicons))
        except ClientError:
            logger.exception(
                "Couldn't get  %s.",
            )
            raise
        else:
            return lexicons
```

- Para obter detalhes da API, consulte a ListLexiconsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
```

```
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/
config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:#{l.attributes.alphabet}"
    puts "  Language:#{l.attributes.language}"
    puts
  end
rescue StandardError => e
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts e.message
end
```

- Para obter detalhes da API, consulte ListLexiconsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_lexicons(client: &Client) -> Result<(), Error> {
    let resp = client.list_lexicons().send().await?;

    println!("Lexicons:");

    let lexicons = resp.lexicons();
```

```
    for lexicon in lexicons {
        println!("  Name:     {}", lexicon.name().unwrap_or_default());
        println!(
            "  Language: {:?}\n",
            lexicon
                .attributes()
                .as_ref()
                .map(|attrib| attrib
                    .language_code
                    .as_ref()
                    .expect("languages must have language codes"))
                .expect("languages must have attributes")
        );
    }

    println!();
    println!("Found {} lexicons.", lexicons.len());
    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListLexiconsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ply->listlexicons( ).
        DATA(lt_lexicons) = oo_result->get_lexicons( ).
        DATA(lv_count) = lines( lt_lexicons ).
        MESSAGE |Found { lv_count } lexicons| TYPE 'I'.
      CATCH /aws1/cx_plyinvalidnexttokenex.
        MESSAGE 'Invalid NextToken.' TYPE 'E'.
```

```
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListLexiconsreferência da API AWS SDK for SAP ABAP.

Use **`ListSpeechSynthesisTasks`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListSpeechSynthesisTasks`.

CLI

AWS CLI

Para listar suas tarefas de síntese de fala

O exemplo `list-speech-synthesis-tasks` a seguir lista suas tarefas de síntese de fala.

```
aws polly list-speech-synthesis-tasks
```

Saída:

```
{
    "SynthesisTasks": [
        {
            "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
            "TaskStatus": "completed",
            "OutputUri": "https://s3.us-west-2.amazonaws.com/amzn-s3-demo-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
            "CreationTime": 1603911042.689,
            "RequestCharacters": 1311,
            "OutputFormat": "mp3",
            "TextType": "text",
            "VoiceId": "Joanna"
        }
    ]
}
```

Para obter mais informações, consulte Criar arquivos de áudio longos no Guia do desenvolvedor do Amazon Polly.

- Para obter detalhes da API, consulte ListSpeechSynthesisTasksem Referência de AWS CLI Comandos.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Only pass optional parameters if they have values
    IF iv_max_results IS NOT INITIAL AND iv_status IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_maxresults = iv_max_results
        iv_status = iv_status ).
    ELSEIF iv_max_results IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_maxresults = iv_max_results ).
    ELSEIF iv_status IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_status = iv_status ).
    ELSE.
      oo_result = lo_ply->listspeechsynthesistasks( ).
    ENDIF.
    DATA(lt_tasks) = oo_result->get_synthesistasks( ).
    DATA(lv_count) = lines( lt_tasks ).
    MESSAGE |Found { lv_count } synthesis tasks| TYPE 'I'.
  CATCH /aws1/cx_plyinvalidnexttokenex.
    MESSAGE 'Invalid NextToken.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListSpeechSynthesisTasksreferência da API AWS SDK for SAP ABAP.

Use **PutLexicon**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutLexicon`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução à síntese de conversão de texto em fala

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Polly;
    using Amazon.Polly.Model;

    /// <summary>
    /// Creates a new Amazon Polly lexicon using the AWS SDK for .NET.
    /// </summary>
    public class PutLexicon
    {
        public static async Task Main()
        {
            string lexiconContent = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"
 +
                "<lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/
pronunciation-lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" "
 +
                "xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-
lexicon http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" " +
                "alphabet=\"ipa\" xml:lang=\"en-US\">" +
                "<lexeme><grapheme>test1</grapheme><alias>test2</alias></lexeme>"
 +
```

```
                "</lexicon>";
            string lexiconName = "SampleLexicon";

            var client = new AmazonPollyClient();
            var putLexiconRequest = new PutLexiconRequest()
            {
                Name = lexiconName,
                Content = lexiconContent,
            };

            try
            {
                var response = await client.PutLexiconAsync(putLexiconRequest);
                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    Console.WriteLine($"Successfully created Lexicon:
{lexiconName}.");
                }
                else
                {
                    Console.WriteLine($"Could not create Lexicon:
{lexiconName}.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception caught: " + ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte PutLexicona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como armazenar um léxico

O exemplo de `put-lexicon` a seguir armazena o léxico de pronúncia especificado. O `example.pls` arquivo especifica um léxico W3C PLS-compliant .

```
aws polly put-lexicon \
    --name w3c \
    --content file://example.pls
```

Conteúdo de `example.pls`

```
{
    <?xml version="1.0" encoding="UTF-8"?>
    <lexicon version="1.0"
        xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
            http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
        alphabet="ipa"
        xml:lang="en-US">
        <lexeme>
            <grapheme>W3C</grapheme>
            <alias>World Wide Web Consortium</alias>
        </lexeme>
    </lexicon>
}
```

Este comando não produz saída.

Para obter mais informações, consulte Como usar a PutLexicon operação no Amazon Polly Developer Guide.

- Para obter detalhes da API, consulte PutLexiconem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
```

```
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def create_lexicon(self, name, content):
        """
        Creates a lexicon with the specified content. A lexicon contains custom
        pronunciations.

        :param name: The name of the lexicon.
        :param content: The content of the lexicon.
        """
        try:
            self.polly_client.put_lexicon(Name=name, Content=content)
            logger.info("Created lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't create lexicon %s.")
            raise
```

- Para obter detalhes da API, consulte a PutLexiconReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_lexicon(client: &Client, name: &str, from: &str, to: &str) ->
 Result<(), Error> {
    let content = format!("<?xml version=\"1.0\" encoding=\"UTF-8\"?>
    <lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/pronunciation-
lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
    xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon http://
www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\"
    alphabet=\"ipa\" xml:lang=\"en-US\">
    <lexeme><grapheme>{}</grapheme><alias>{}</alias></lexeme>
    </lexicon>", from, to);

    client
        .put_lexicon()
        .name(name)
        .content(content)
        .send()
        .await?;

    println!("Added lexicon");

    Ok(())
}
```

- Para obter detalhes da API, consulte a PutLexiconreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ply->putlexicon(
          iv_name = iv_name
          iv_content = iv_content ).
        MESSAGE 'Lexicon created successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_plyinvalidlexiconex.
        MESSAGE 'Invalid lexicon.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconsizeexcdex.
        MESSAGE 'Lexicon size exceeded.' TYPE 'E'.
      CATCH /aws1/cx_plymaxlexemelengthe00.
        MESSAGE 'Maximum lexeme length exceeded.' TYPE 'E'.
      CATCH /aws1/cx_plymaxlexiconsnoexc00.
        MESSAGE 'Maximum number of lexicons exceeded.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plyunsuppedplsalpha00.
        MESSAGE 'Unsupported PLS alphabet.' TYPE 'E'.
      CATCH /aws1/cx_plyunsuppedplslangu00.
        MESSAGE 'Unsupported PLS language.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutLexiconreferência da API AWS SDK for SAP ABAP.

Use **`StartSpeechSynthesisTask`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartSpeechSynthesisTask`.

CLI

AWS CLI

Como sintetizar texto

O exemplo de `start-speech-synthesis-task` a seguir sintetiza o texto em `text_file.txt` e armazena o arquivo MP3 resultante no bucket especificado.

```
aws polly start-speech-synthesis-task \
    --output-format mp3 \
    --output-s3-bucket-name amzn-s3-demo-bucket \
    --text  file://text_file.txt \
    --voice-id Joanna
```

Saída:

```
{
```

```
    "SynthesisTask": {
        "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
        "TaskStatus": "scheduled",
        "OutputUri": "https://s3.us-east-2.amazonaws.com/amzn-s3-demo-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
        "CreationTime": 1603911042.689,
        "RequestCharacters": 1311,
        "OutputFormat": "mp3",
        "TextType": "text",
        "VoiceId": "Joanna"
    }
}
```

Para obter mais informações, consulte Criar arquivos de áudio longos no Guia do desenvolvedor do Amazon Polly.

- Para obter detalhes da API, consulte StartSpeechSynthesisTaskem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None
```

```
    def do_synthesis_task(
        self,
        text,
        engine,
        voice,
        audio_format,
        s3_bucket,
        lang_code=None,
        include_visemes=False,
        wait_callback=None,
    ):
        """
        Start an asynchronous task to synthesize speech or speech marks, wait for
        the task to complete, retrieve the output from Amazon S3, and return the
        data.

        An asynchronous task is required when the text is too long for near-real
time
        synthesis.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech.
When
                             speech marks are synthesized, the output format is
JSON.
        :param s3_bucket: The name of an existing Amazon S3 bucket that you have
                          write access to. Synthesis output is written to this
bucket.
        :param lang_code: The language code of the voice to use. This has an
effect
                          only when a bilingual voice is selected.
        :param include_visemes: When True, a second request is made to Amazon
Polly
                                to synthesize a list of visemes, using the
specified
                                text and voice. A viseme represents the visual
position
                                of the face and mouth when saying part of a word.
        :param wait_callback: A callback function that is called periodically
during
```

```
                                task processing, to give the caller an opportunity
to
                                take action, such as to display status.
        :return: The audio stream that contains the synthesized speech and a list
                 of visemes that are associated with the speech audio.
        """
        try:
            kwargs = {
                "Engine": engine,
                "OutputFormat": audio_format,
                "OutputS3BucketName": s3_bucket,
                "Text": text,
                "VoiceId": voice,
            }
            if lang_code is not None:
                kwargs["LanguageCode"] = lang_code
            response = self.polly_client.start_speech_synthesis_task(**kwargs)
            speech_task = response["SynthesisTask"]
            logger.info("Started speech synthesis task %s.",
speech_task["TaskId"])

            viseme_task = None
            if include_visemes:
                kwargs["OutputFormat"] = "json"
                kwargs["SpeechMarkTypes"] = ["viseme"]
                response =
self.polly_client.start_speech_synthesis_task(**kwargs)
                viseme_task = response["SynthesisTask"]
                logger.info("Started viseme synthesis task %s.",
viseme_task["TaskId"])
        except ClientError:
            logger.exception("Couldn't start synthesis task.")
            raise
        else:
            bucket = self.s3_resource.Bucket(s3_bucket)
            audio_stream = self._wait_for_task(
                10, speech_task["TaskId"], "speech", wait_callback, bucket
            )

            visemes = None
            if include_visemes:
                viseme_data = self._wait_for_task(
                    10, viseme_task["TaskId"], "viseme", wait_callback, bucket
                )
```

```
                visemes = [
                    json.loads(v) for v in viseme_data.read().decode().split() if
v
                ]

            return audio_stream, visemes
```

- Para obter detalhes da API, consulte a StartSpeechSynthesisTaskReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Only pass optional parameters if they have values
        IF iv_lang_code IS NOT INITIAL AND iv_s3_key_prefix IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_outputs3keyprefix = iv_s3_key_prefix
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSEIF iv_lang_code IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
```

```
        ELSEIF iv_s3_key_prefix IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_outputs3keyprefix = iv_s3_key_prefix
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ELSE.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ENDIF.
        MESSAGE 'Speech synthesis task started.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalids3bucketex.
        MESSAGE 'Invalid S3 bucket.' TYPE 'E'.
      CATCH /aws1/cx_plyinvalidssmlex.
        MESSAGE 'Invalid SSML.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plytextlengthexcdex.
        MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartSpeechSynthesisTaskreferência da API AWS SDK for SAP ABAP.

Use **`SynthesizeSpeech`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `SynthesizeSpeech`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução à síntese de conversão de texto em fala

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeech
{
    public static async Task Main()
    {
        string outputFileName = "speech.mp3";
        string text = "Twas brillig, and the slithy toves did gyre and gimbol
in the wabe";

        var client = new AmazonPollyClient();
        var response = await PollySynthesizeSpeech(client, text);

        WriteSpeechToStream(response.AudioStream, outputFileName);
    }

    /// <summary>
    /// Calls the Amazon Polly SynthesizeSpeechAsync method to convert text
    /// to speech.
    /// </summary>
    /// <param name="client">The Amazon Polly client object used to connect
    /// to the Amazon Polly service.</param>
    /// <param name="text">The text to convert to speech.</param>
    /// <returns>A SynthesizeSpeechResponse object that includes an
AudioStream
    /// object with the converted text.</returns>
    private static async Task<SynthesizeSpeechResponse>
PollySynthesizeSpeech(IAmazonPolly client, string text)
```

```
        {
            var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
            {
                OutputFormat = OutputFormat.Mp3,
                VoiceId = VoiceId.Joanna,
                Text = text,
            };

            var synthesizeSpeechResponse =
                await client.SynthesizeSpeechAsync(synthesizeSpeechRequest);

            return synthesizeSpeechResponse;
        }

        /// <summary>
        /// Writes the AudioStream returned from the call to
        /// SynthesizeSpeechAsync to a file in MP3 format.
        /// </summary>
        /// <param name="audioStream">The AudioStream returned from the
        /// call to the SynthesizeSpeechAsync method.</param>
        /// <param name="outputFileName">The full path to the file in which to
        /// save the audio stream.</param>
        private static void WriteSpeechToStream(Stream audioStream, string
outputFileName)
        {
            var outputStream = new FileStream(
                outputFileName,
                FileMode.Create,
                FileAccess.Write);
            byte[] buffer = new byte[2 * 1024];
            int readBytes;

            while ((readBytes = audioStream.Read(buffer, 0, 2 * 1024)) > 0)
            {
                outputStream.Write(buffer, 0, readBytes);
            }

            // Flushes the buffer to avoid losing the last second or so of
            // the synthesized text.
            outputStream.Flush();
            Console.WriteLine($"Saved {outputFileName} to disk.");
        }
    }
```

Sintetize fala a partir de texto usando marcas de fala com o Amazon Polly usando um SDK. AWS

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeechMarks
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();
        string outputFileName = "speechMarks.json";

        var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
        {
            OutputFormat = OutputFormat.Json,
            SpeechMarkTypes = new List<string>
            {
                SpeechMarkType.Viseme,
                SpeechMarkType.Word,
            },
            VoiceId = VoiceId.Joanna,
            Text = "This is a sample text to be synthesized.",
        };

        try
        {
            using (var outputStream = new FileStream(outputFileName,
FileMode.Create, FileAccess.Write))
            {
                var synthesizeSpeechResponse = await
client.SynthesizeSpeechAsync(synthesizeSpeechRequest);
                var buffer = new byte[2 * 1024];
                int readBytes;

                var inputStream = synthesizeSpeechResponse.AudioStream;
```

```
                    while ((readBytes = inputStream.Read(buffer, 0, 2 * 1024)) >
 0)
                    {
                        outputStream.Write(buffer, 0, readBytes);
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- Para obter detalhes da API, consulte SynthesizeSpeecha Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
```

```
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
 built this working demo " +
            " of Amazon Polly in Java Version 2. Have fun building voice enabled
 apps with Amazon Polly (that's me!), and always "
            +
            " look at the AWS website for tips and tricks on using Amazon Polly
 and other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
            DescribeVoicesRequest describeVoiceRequest =
 DescribeVoicesRequest.builder()
                    .engine("standard")
                    .build();

            DescribeVoicesResponse describeVoicesResult =
 polly.describeVoices(describeVoiceRequest);
            Voice voice = describeVoicesResult.voices().stream()
                    .filter(v -> v.name().equals("Joanna"))
                    .findFirst()
```

```
                    .orElseThrow(() -> new RuntimeException("Voice not found"));
            InputStream stream = synthesize(polly, SAMPLE, voice,
 OutputFormat.MP3);
            AdvancedPlayer player = new AdvancedPlayer(stream,

 javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
            player.setPlayBackListener(new PlaybackListener() {
                public void playbackStarted(PlaybackEvent evt) {
                    System.out.println("Playback started");
                    System.out.println(SAMPLE);
                }

                public void playbackFinished(PlaybackEvent evt) {
                    System.out.println("Playback finished");
                }
            });

            // play it!
            player.play();

        } catch (PollyException | JavaLayerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static InputStream synthesize(PollyClient polly, String text, Voice
 voice, OutputFormat format)
            throws IOException {
        SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
                .text(text)
                .voiceId(voice.id())
                .outputFormat(format)
                .build();

        ResponseInputStream<SynthesizeSpeechResponse> synthRes =
 polly.synthesizeSpeech(synthReq);
        return synthRes;
    }
}
```

- Para obter detalhes da API, consulte SynthesizeSpeecha Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def synthesize(
        self, text, engine, voice, audio_format, lang_code=None,
 include_visemes=False
    ):
        """
        Synthesizes speech or speech marks from text, using the specified voice.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech.
 When
                             speech marks are synthesized, the output format is
 JSON.
```

```
        :param lang_code: The language code of the voice to use. This has an
effect
                          only when a bilingual voice is selected.
        :param include_visemes: When True, a second request is made to Amazon
Polly
                                to synthesize a list of visemes, using the
specified
                                text and voice. A viseme represents the visual
position
                                of the face and mouth when saying part of a word.
        :return: The audio stream that contains the synthesized speech and a list
                 of visemes that are associated with the speech audio.
        """
        try:
            kwargs = {
                "Engine": engine,
                "OutputFormat": audio_format,
                "Text": text,
                "VoiceId": voice,
            }
            if lang_code is not None:
                kwargs["LanguageCode"] = lang_code
            response = self.polly_client.synthesize_speech(**kwargs)
            audio_stream = response["AudioStream"]
            logger.info("Got audio stream spoken by %s.", voice)
            visemes = None
            if include_visemes:
                kwargs["OutputFormat"] = "json"
                kwargs["SpeechMarkTypes"] = ["viseme"]
                response = self.polly_client.synthesize_speech(**kwargs)
                visemes = [
                    json.loads(v)
                    for v in response["AudioStream"].read().decode().split()
                    if v
                ]
                logger.info("Got %s visemes.", len(visemes))
        except ClientError:
            logger.exception("Couldn't get audio stream.")
            raise
        else:
            return audio_stream, visemes
```

- Para obter detalhes da API, consulte a SynthesizeSpeechReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?
    puts 'You must supply a filename'
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: #{filename}"
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/
config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
                                  output_format: 'mp3',
```

```
                                    text: contents,
                                    voice_id: 'Joanna'
                                  })

  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = "#{first_part}.mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: #{mp3_file}"
rescue StandardError => e
  puts 'Got error:'
  puts 'Error message:'
  puts e.message
end
```

- Para obter detalhes da API, consulte SynthesizeSpeecha Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn synthesize(client: &Client, filename: &str) -> Result<(), Error> {
    let content = fs::read_to_string(filename);

    let resp = client
```

```
        .synthesize_speech()
        .output_format(OutputFormat::Mp3)
        .text(content.unwrap())
        .voice_id(VoiceId::Joanna)
        .send()
        .await?;

    // Get MP3 data from response and save it
    let mut blob = resp
        .audio_stream
        .collect()
        .await
        .expect("failed to read data");

    let parts: Vec<&str> = filename.split('.').collect();
    let out_file = format!("{}{}", String::from(parts[0]), ".mp3");

    let mut file = tokio::fs::File::create(out_file)
        .await
        .expect("failed to create file");

    file.write_all_buf(&mut blob)
        .await
        .expect("failed to write to file");

    Ok(())
}
```

- Para obter detalhes da API, consulte a SynthesizeSpeechreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Only pass optional language code if it has a value
        IF iv_lang_code IS NOT INITIAL.
          oo_result = lo_ply->synthesizespeech(
            iv_engine = iv_engine
            iv_outputformat = iv_output_fmt
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSE.
          oo_result = lo_ply->synthesizespeech(
            iv_engine = iv_engine
            iv_outputformat = iv_output_fmt
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ENDIF.
        MESSAGE 'Speech synthesized successfully.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalidssmlex.
        MESSAGE 'Invalid SSML.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plytextlengthexcdex.
        MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SynthesizeSpeechreferência da API AWS SDK for SAP ABAP.

Cenários para o uso do Amazon Polly AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Polly com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon Polly ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Converta texto em fala e volte para texto usando um AWS SDK
- Crie um aplicativo de sincronização labial com o Amazon Polly usando um AWS SDK
- Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
- Introdução à síntese de conversão de texto em fala

Converta texto em fala e volte para texto usando um AWS SDK

O exemplo de código a seguir mostra como:

- Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio.
- Fazer upload do arquivo de áudio para um bucket do Amazon S3.
- Usar o Amazon Transcribe para converter o arquivo de áudio em texto.
- Exibir o texto.

Rust

SDK para Rust

Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio, fazer upload do arquivo de áudio em um bucket do Amazon S3, usar o Amazon Transcribe para converter esse arquivo de áudio em texto e exibir o texto.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Crie um aplicativo de sincronização labial com o Amazon Polly usando um AWS SDK

O exemplo de código a seguir mostra como criar um aplicativo de sincronização de lábios com o Amazon Polly.

Python

SDK para Python (Boto3)

Mostra como usar o Amazon Polly e o Tkinter para criar um aplicativo de sincronização labial que exibe um rosto animado falando junto com a fala sintetizada pelo Amazon Polly. Lip-sync é realizado solicitando uma lista de fonemas do Amazon Polly que correspondam à fala sintetizada.

- Obter metadados de voz do Amazon Polly e exibi-os em um aplicativo Tkinter.
- Obter áudio de fala sintetizado e respectivas marcas de fala de fonema do Amazon Polly.
- Reproduzir o áudio com movimentos sincronizados da boca em um rosto animado.
- Enviar tarefas de síntese assíncrona para textos longos e recuperar a saída de um bucket do Amazon Simple Storage Service (Amazon S3).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Polly

Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio

Os exemplos de código a seguir mostram como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

.NET

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.

- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract

- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado nas funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
```

```
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
```

```
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
```

```
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Introdução à síntese de conversão de texto em fala

O exemplo de código a seguir mostra como:

- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Polly Getting Started Script
# This script demonstrates how to use Amazon Polly with the AWS CLI

set -euo pipefail

# Set up logging
LOG_FILE="polly-tutorial.log"
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
WORK_DIR=$(mktemp -d)
trap 'cleanup_temp' EXIT

cleanup_temp() {
    rm -rf "$WORK_DIR"
}

echo "Starting Amazon Polly tutorial at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    # Use bash array to safely handle arguments
    bash -c "$1" 2>&1 | tee -a "$LOG_FILE" || return $?
}

# Function to check for errors
check_error() {
    if echo "$1" | grep -iq "error"; then
        echo "ERROR detected in output. Exiting script." | tee -a "$LOG_FILE"
        echo "$1" | tee -a "$LOG_FILE"
        return 1
```

```
    fi
    return 0
}

# Function to handle errors and cleanup
handle_error() {
    local line_number=$1
    echo "Error occurred at line $line_number. Attempting cleanup..." | tee -a
 "$LOG_FILE"
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "" | tee -a "$LOG_FILE"
    echo "============================================================" | tee -a
 "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "============================================================" | tee -a
 "$LOG_FILE"

    # Delete lexicon if it exists
    if [[ -n "${LEXICON_NAME:-}" ]]; then
        echo "Deleting lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
        if aws polly delete-lexicon --name "$LEXICON_NAME" 2>&1 | tee -a
 "$LOG_FILE"; then
            echo "Lexicon deleted successfully." | tee -a "$LOG_FILE"
        else
            echo "Warning: Failed to delete lexicon." | tee -a "$LOG_FILE"
        fi
    fi

    # Remove audio files
    for file in output.mp3 ssml-output.mp3 lexicon-output.mp3 example.pls; do
        if [[ -f "$file" ]]; then
            rm -f "$file"
            echo "Removed $file" | tee -a "$LOG_FILE"
        fi
    done

    echo "Cleanup complete." | tee -a "$LOG_FILE"
}
```

```
# Trap errors with line number
trap 'handle_error ${LINENO}' ERR

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    echo "AWS CLI is not installed. Please install it first." | tee -a
 "$LOG_FILE"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "AWS credentials are not configured. Please configure them first." | tee
 -a "$LOG_FILE"
    exit 1
fi

# Step 1: Verify Amazon Polly is available
echo "Step 1: Verifying Amazon Polly availability" | tee -a "$LOG_FILE"
if aws polly describe-voices --query 'Voices[0].Name' --output text &> /dev/null;
 then
    echo "Amazon Polly is available. Proceeding with tutorial." | tee -a
 "$LOG_FILE"
else
    echo "Amazon Polly is not available in your AWS CLI installation or region."
 | tee -a "$LOG_FILE"
    echo "Please update your AWS CLI to the latest version or check your region."
 | tee -a "$LOG_FILE"
    exit 1
fi

# Step 2: List available voices
echo "" | tee -a "$LOG_FILE"
echo "Step 2: Listing available voices" | tee -a "$LOG_FILE"
log_cmd "aws polly describe-voices --language-code en-US --output text --query
 'Voices[0:3].[Id, LanguageCode, Gender]'" || true

# Step 3: Basic text-to-speech conversion
echo "" | tee -a "$LOG_FILE"
echo "Step 3: Converting text to speech" | tee -a "$LOG_FILE"
OUTPUT_FILE="${WORK_DIR}/output.mp3"
POLLY_TEXT="Hello, welcome to Amazon Polly. This is a sample text to speech
 conversion."
```

```
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --text
 '$POLLY_TEXT' '$OUTPUT_FILE'" || true

if [[ -f "$OUTPUT_FILE" ]]; then
    echo "Successfully created output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$OUTPUT_FILE" output.mp3
else
    echo "Failed to create output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Step 4: Using SSML for enhanced speech
echo "" | tee -a "$LOG_FILE"
echo "Step 4: Using SSML for enhanced speech" | tee -a "$LOG_FILE"
SSML_OUTPUT="${WORK_DIR}/ssml-output.mp3"
SSML_TEXT='<speak>Hello! <break time="1s"/> This is a sample of <emphasis>SSML
 enhanced speech</emphasis>.</speak>'
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Matthew --
text-type ssml --text '$SSML_TEXT' '$SSML_OUTPUT'" || true

if [[ -f "$SSML_OUTPUT" ]]; then
    echo "Successfully created ssml-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$SSML_OUTPUT" ssml-output.mp3
else
    echo "Failed to create ssml-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Step 5: Working with lexicons
echo "" | tee -a "$LOG_FILE"
echo "Step 5: Working with lexicons" | tee -a "$LOG_FILE"

# Generate a random identifier for the lexicon (max 20 chars, alphanumeric only)
LEXICON_NAME="example$(openssl rand -hex 6 | cut -c 1-10)"
echo "Using lexicon name: $LEXICON_NAME" | tee -a "$LOG_FILE"

# Create a lexicon file
echo "Creating lexicon file..." | tee -a "$LOG_FILE"
LEXICON_FILE="${WORK_DIR}/example.pls"
cat > "$LEXICON_FILE" << 'EOF'
```

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
      xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
      alphabet="ipa"
      xml:lang="en-US">
  <lexeme>
    <grapheme>AWS</grapheme>
    <alias>Amazon Web Services</alias>
  </lexeme>
</lexicon>
EOF

# Upload the lexicon
echo "Uploading lexicon..." | tee -a "$LOG_FILE"
LEXICON_ARN=$(aws polly put-lexicon --name "$LEXICON_NAME" --content
 file://"$LEXICON_FILE" --query 'LexiconArn' --output text 2>&1) || true
if [[ -n "$LEXICON_ARN" && "$LEXICON_ARN" != "" && ! "$LEXICON_ARN" =~ error ]];
 then
    echo "Lexicon uploaded with ARN: $LEXICON_ARN" | tee -a "$LOG_FILE"
    aws polly tag-resource --resource-arn "$LEXICON_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-polly-gs 2>&1 | tee -a
 "$LOG_FILE" || true
else
    echo "Lexicon uploaded." | tee -a "$LOG_FILE"
fi

# List available lexicons
echo "Listing available lexicons..." | tee -a "$LOG_FILE"
log_cmd "aws polly list-lexicons --output text --query 'Lexicons[*].[Name]'" ||
 true

# Get details about the lexicon
echo "Getting details about the lexicon..." | tee -a "$LOG_FILE"
log_cmd "aws polly get-lexicon --name '$LEXICON_NAME' --output text --query
 'Lexicon.Name'" || true

# Use the lexicon when synthesizing speech
echo "Using the lexicon for speech synthesis..." | tee -a "$LOG_FILE"
LEXICON_OUTPUT="${WORK_DIR}/lexicon-output.mp3"
LEXICON_TEXT="I work with AWS every day."
```

```
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --
lexicon-names '$LEXICON_NAME' --text '$LEXICON_TEXT' '$LEXICON_OUTPUT'" || true

if [[ -f "$LEXICON_OUTPUT" ]]; then
    echo "Successfully created lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$LEXICON_OUTPUT" lexicon-output.mp3
else
    echo "Failed to create lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Summary of created resources
echo "" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "TUTORIAL SUMMARY" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "Created resources:" | tee -a "$LOG_FILE"
echo "1. Lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
echo "2. Audio files:" | tee -a "$LOG_FILE"
echo "   - output.mp3" | tee -a "$LOG_FILE"
echo "   - ssml-output.mp3" | tee -a "$LOG_FILE"
echo "   - lexicon-output.mp3" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Cleanup with auto-confirmation
echo "" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"
cleanup

echo "" | tee -a "$LOG_FILE"
echo "Tutorial completed successfully!" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- DeleteLexicon
- DescribeVoices
- GetLexicon
- Help
- ListLexicons
- PutLexicon
- SynthesizeSpeech

Exemplos de código para AWS Price List usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Price List com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- AWS Price List Guia do usuário — Mais informações sobre AWS Price List.
- AWS Price List Referência da API — Detalhes sobre todas as AWS Price List ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS Price List usar AWS SDKs
  - Ações para AWS Price List usar AWS SDKs
    - Usar DescribeServices com uma CLI
    - Usar GetAttributeValues com uma CLI
    - Usar GetProducts com uma CLI

Exemplos básicos para AWS Price List usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Price List com AWS SDKs.

Exemplos

- Ações para AWS Price List usar AWS SDKs
  - Usar DescribeServices com uma CLI
  - Usar GetAttributeValues com uma CLI
  - Usar GetProducts com uma CLI

Ações para AWS Price List usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS Price List ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Price List.

Exemplos

- Usar DescribeServices com uma CLI
- Usar GetAttributeValues com uma CLI
- Usar GetProducts com uma CLI

Usar **`DescribeServices`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeServices`.

CLI

AWS CLI

Para recuperar metadados do serviço

Este exemplo recupera os metadados do código de serviço do Amazon EC2.

Comando:

```
aws pricing describe-services --service-code AmazonEC2 --format-version aws_v1 --
max-items 1
```

Saída:

```
{
  "Services": [
      {
          "ServiceCode": "AmazonEC2",
          "AttributeNames": [
              "volumeType",
              "maxIopsvolume",
              "instance",
              "instanceCapacity10xlarge",
              "locationType",
              "instanceFamily",
              "operatingSystem",
              "clockSpeed",
              "LeaseContractLength",
              "ecu",
              "networkPerformance",
              "instanceCapacity8xlarge",
              "group",
              "maxThroughputvolume",
              "gpuMemory",
              "ebsOptimized",
              "elasticGpuType",
              "maxVolumeSize",
              "gpu",
              "processorFeatures",
              "intelAvxAvailable",
              "instanceCapacity4xlarge",
              "servicecode",
              "groupDescription",
              "processorArchitecture",
              "physicalCores",
              "productFamily",
              "enhancedNetworkingSupported",
              "intelTurboAvailable",
              "memory",
              "dedicatedEbsThroughput",
              "vcpu",
              "OfferingClass",
```

```
                "instanceCapacityLarge",
                "capacitystatus",
                "termType",
                "storage",
                "intelAvx2Available",
                "storageMedia",
                "physicalProcessor",
                "provisioned",
                "servicename",
                "PurchaseOption",
                "instanceCapacity18xlarge",
                "instanceType",
                "tenancy",
                "usagetype",
                "normalizationSizeFactor",
                "instanceCapacity2xlarge",
                "instanceCapacity16xlarge",
                "maxIopsBurstPerformance",
                "instanceCapacity12xlarge",
                "instanceCapacity32xlarge",
                "instanceCapacityXlarge",
                "licenseModel",
                "currentGeneration",
                "preInstalledSw",
                "location",
                "instanceCapacity24xlarge",
                "instanceCapacity9xlarge",
                "instanceCapacityMedium",
                "operation"
            ]
        }
    ],
    "FormatVersion": "aws_v1"
}
```

- Para obter detalhes da API, consulte DescribeServicesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna os metadados de todos os códigos de serviço disponíveis na região us-east-1.

```
Get-PLSService -Region us-east-1
```

Saída:

```
AttributeNames                                                  ServiceCode
--------------                                                  -----------
{productFamily, servicecode, groupDescription, termType...}     AWSBudgets
{productFamily, servicecode, termType, usagetype...}            AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}            AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}            AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}            AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}            AWSConfig
...
```

Exemplo 2: retorna os metadados para o serviço do Amazon EC2 na região us-east-1.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Saída:

```
AttributeNames
 ServiceCode
--------------
 -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Para obter detalhes da API, consulte DescribeServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna os metadados de todos os códigos de serviço disponíveis na região us-east-1.

```
Get-PLSService -Region us-east-1
```

Saída:

```
AttributeNames                                                  ServiceCode
--------------                                                  -----------
{productFamily, servicecode, groupDescription, termType...}     AWSBudgets
{productFamily, servicecode, termType, usagetype...}            AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}            AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}            AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}            AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}            AWSConfig
...
```

Exemplo 2: retorna os metadados para o serviço do Amazon EC2 na região us-east-1.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Saída:

```
AttributeNames
 ServiceCode
--------------
 -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Para obter detalhes da API, consulte DescribeServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetAttributeValues`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetAttributeValues`.

CLI

AWS CLI

Para recuperar uma lista de valores de atributos

O exemplo `get-attribute-values` a seguir recupera uma lista de valores disponíveis para o atributo especificado.

```
aws pricing get-attribute-values \
    --service-code AmazonEC2 \
    --attribute-name volumeType \
    --max-items 2
```

Saída:

```
{
    "NextToken":
 "eyJOZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ==",
    "AttributeValues": [
        {
            "Value": "Cold HDD"
        },
        {
            "Value": "General Purpose"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetAttributeValuesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna os valores do atributo “volumeType” para o Amazon EC2 na região us-east-1.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region
 us-east-1
```

Saída:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
```

```
Throughput Optimized HDD
```

- Para obter detalhes da API, consulte GetAttributeValuesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna os valores do atributo “volumeType” para o Amazon EC2 na região us-east-1.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region
 us-east-1
```

Saída:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Para obter detalhes da API, consulte GetAttributeValuesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetProducts`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetProducts`.

CLI

AWS CLI

Para recuperar uma lista de produtos

Este exemplo recupera uma lista de produtos que correspondem aos critérios fornecidos.

Comando:

```
aws pricing get-products --filters file://filters.json --format-version aws_v1 --
max-results 1 --service-code AmazonEC2
```

filters.json:

```
        [
  {
    "Type": "TERM_MATCH",
    "Field": "ServiceCode",
    "Value": "AmazonEC2"
  },
  {
    "Type": "TERM_MATCH",
    "Field": "volumeType",
    "Value": "Provisioned IOPS"
  }
]
```

Saída:


```
{
  "FormatVersion": "aws_v1",
  "NextToken": "WGDY7ko8fQXdlaUZVdasFQ==:RVSagyIFn770XQOzdUIcO9BY6ucBG9itXAZGZF/
zioUzOsUKh6PCcPWaOyPZRiMePb986TeoKYB9l55fw/
CyoMq5ymnGmT1Vj39TljbbAlhcqnVfTmPIilx8Uy5bdDaBYy/e/2Ofw9Edzsykbs8LTBuNbiDQ
+BBds5yeI9AQkUepruKk3aEahFPxJ55kx/zk",
  "PriceList": [
      "{\"product\":{\"productFamily\":\"Storage\",\"attributes\":{\"storageMedia
\":\"SSD-backed\",\"maxThroughputvolume\":\"320 MB/sec\",\"volumeType
\":\"Provisioned IOPS\",\"maxIopsvolume\":\"20000\",\"servicecode\":
\"AmazonEC2\",\"usagetype\":\"APS1-EBS:VolumeUsage.piops\",\"locationType
\":\"AWS Region\",\"location\":\"Asia Pacific (Singapore)\",\"servicename
\":\"Amazon Elastic Compute Cloud\",\"maxVolumeSize\":\"16 TiB\",\"operation
\":\"\"},\"sku\":\"3MKHN58N7RDDVGKJ\"},\"serviceCode\":\"AmazonEC2\",\"terms
\":{\"OnDemand\":{\"3MKHN58N7RDDVGKJ.JRTCKXETXF\":{\"priceDimensions\":
{\"3MKHN58N7RDDVGKJ.JRTCKXETXF.6YS6EN2CT7\":{\"unit\":\"GB-Mo\",\"endRange\":
\"Inf\",\"description\":\"$0.138 per GB-month of Provisioned IOPS SSD (io1)
 provisioned storage - Asia Pacific (Singapore)\",\"appliesTo\":[],\"rateCode\":
\"3MKHN58N7RDDVGKJ.JRTCKXETXF.6YS6EN2CT7\",\"beginRange\":\"0\",\"pricePerUnit
\":{\"USD\":\"0.1380000000\"}}},\"sku\":\"3MKHN58N7RDDVGKJ\",\"effectiveDate\":
\"2018-08-01T00:00:00Z\",\"offerTermCode\":\"JRTCKXETXF\",\"termAttributes\":
{}}}},\"version\":\"20180808005701\",\"publicationDate\":\"2018-08-08T00:57:01Z
\"}"
  ]
}
```

- Para obter detalhes da API, consulte GetProductsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna detalhes de todos os produtos para o Amazon EC2.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Saída:

```
{"product":{"productFamily":"Compute Instance","attributes":
{"enhancedNetworkingSupported":"Yes","memory":"30.5
 GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS
 Region","storage":"EBS only","instanceFamily":"Memory
 optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4
 (Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up
 to 10 Gigabit","servicename":"Amazon Elastic Compute
 Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-
BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX,
 Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License
 required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West
 (Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...
```

Exemplo 2: Retorna dados do Amazon EC2 na região us-east-1 filtrados por tipos de volume de “Uso geral” que são. SSD-backed

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
 @{Type="TERM_MATCH";Field="volumeType";Value="General
 Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region
 us-east-1
```

Saída:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- Para obter detalhes da API, consulte GetProductsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes de todos os produtos para o Amazon EC2.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Saída:

```
{"product":{"productFamily":"Compute Instance","attributes":
{"enhancedNetworkingSupported":"Yes","memory":"30.5
 GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS
 Region","storage":"EBS only","instanceFamily":"Memory
 optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4
 (Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up
 to 10 Gigabit","servicename":"Amazon Elastic Compute
 Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-
BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX,
 Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License
 required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West
 (Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...
```

Exemplo 2: Retorna dados do Amazon EC2 na região us-east-1 filtrados por tipos de volume de “Uso geral” que são. SSD-backed

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
 @{Type="TERM_MATCH";Field="volumeType";Value="General
 Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region
 us-east-1
```

Saída:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- Para obter detalhes da API, consulte GetProductsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para o Amazon RDS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Relational Database Service com AWS um kit de desenvolvimento de software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon RDS: mais informações sobre o Amazon RDS.
- Referência da API Amazon RDS — Detalhes sobre todas as ações disponíveis do Amazon RDS.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon RDS AWS SDKs
  - Olá, Amazon RDS
  - Aprenda as noções básicas do Amazon RDS com um AWS SDK
  - Ações para o Amazon RDS usando AWS SDKs
    - Use createdBInstance com um AWS SDK ou CLI
    - Use CreateDBParameterGroupcom um AWS SDK ou CLI
    - Use createDBSnapshot com um AWS SDK ou CLI
    - Use DeletedBInstance com um AWS SDK ou CLI
    - Use DeleteDBParameterGroupcom um AWS SDK ou CLI
    - Use DescribeAccountAttributescom um AWS SDK ou CLI
    - Use DescribeDBEngineVersionscom um AWS SDK ou CLI

    - Use DescribedBInstances com um AWS SDK ou CLI
    - Use DescribeDBParameterGroupscom um AWS SDK ou CLI
    - Use DescribedBParameters com um AWS SDK ou CLI
    - Use DescribedBSnapshots com um AWS SDK ou CLI
    - Use DescribeOrderableDBInstanceOptionscom um AWS SDK ou CLI
    - Use GenerateRDSAuthTokencom um AWS SDK
    - Use ModifyDBInstance com um AWS SDK ou CLI
    - Use ModifyDBParameterGroupcom um AWS SDK ou CLI
    - Use RebootDBInstance com um AWS SDK ou CLI
- Cenários para o Amazon RDS usando AWS SDKs
  - Crie um rastreador de itens de trabalho do Aurora Sem Servidor
  - Criação de uma instância de banco de dados relacional
- Exemplos de tecnologia sem servidor para o Amazon RDS
  - Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

Exemplos básicos para o uso do Amazon RDS AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Relational Database Service com SDKs da AWS .

Exemplos

- Olá, Amazon RDS
- Aprenda as noções básicas do Amazon RDS com um AWS SDK
- Ações para o Amazon RDS usando AWS SDKs
  - Use createdBInstance com um AWS SDK ou CLI
  - Use CreateDBParameterGroupcom um AWS SDK ou CLI
  - Use createDBSnapshot com um AWS SDK ou CLI
  - Use DeletedBInstance com um AWS SDK ou CLI
  - Use DeleteDBParameterGroupcom um AWS SDK ou CLI
  - Use DescribeAccountAttributescom um AWS SDK ou CLI
  - Use DescribeDBEngineVersionscom um AWS SDK ou CLI
  - Use DescribedBInstances com um AWS SDK ou CLI

- Use DescribeDBParameterGroupscom um AWS SDK ou CLI
- Use DescribedBParameters com um AWS SDK ou CLI
- Use DescribedBSnapshots com um AWS SDK ou CLI
- Use DescribeOrderableDBInstanceOptionscom um AWS SDK ou CLI
- Use GenerateRDSAuthTokencom um AWS SDK
- Use ModifyDBInstance com um AWS SDK ou CLI
- Use ModifyDBParameterGroupcom um AWS SDK ou CLI
- Use RebootDBInstance com um AWS SDK ou CLI

Olá, Amazon RDS

O exemplo de código a seguir mostra como começar a usar o Amazon RDS.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.RDS;
using Amazon.RDS.Model;

namespace RDSActions;

public static class HelloRds
{
    static async Task Main(string[] args)
    {
        var rdsClient = new AmazonRDSClient();

        Console.WriteLine($"Hello Amazon RDS! Following are some of your DB
 instances:");
```

```
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first twenty DB instances.
        var response = await rdsClient.DescribeDBInstancesAsync(
            new DescribeDBInstancesRequest()
            {
                MaxRecords = 20 // Must be between 20 and 100.
            });

        foreach (var instance in response.DBInstances)
        {
            Console.WriteLine($"\tDB name: {instance.DBName}");
            Console.WriteLine($"\tArn: {instance.DBInstanceArn}");
            Console.WriteLine($"\tIdentifier: {instance.DBInstanceIdentifier}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)
```

```
# Set this project's name.
project("hello_rds")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rds.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_rds.cpp.

```
#include <aws/core/Aws.h>
```

```
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBInstancesRequest.h>
#include <iostream>

/*
 *  A "Hello Rds" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client and
 *  describes the Amazon RDS instances.
 *
 *  main function
 *
 *  Usage: 'hello_rds'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);
        Aws::String marker;
        std::vector<Aws::String> instanceDBIDs;

        do {
            Aws::RDS::Model::DescribeDBInstancesRequest request;

            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
                    rdsClient.DescribeDBInstances(request);

            if (outcome.IsSuccess()) {
                for (auto &instance: outcome.GetResult().GetDBInstances()) {
                    instanceDBIDs.push_back(instance.GetDBInstanceIdentifier());
                }
```

```
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with RDS::DescribeDBInstances. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << instanceDBIDs.size() << " RDS instances found." <<
 std::endl;
        for (auto &instanceDBID: instanceDBIDs) {
            std::cout << "   Instance: " << instanceDBID << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para C++ .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Relational Database
 Service (Amazon RDS)
// client and list up to 20 DB instances in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 rdsClient := rds.NewFromConfig(sdkConfig)
 const maxInstances = 20
 fmt.Printf("Let's list up to %v DB instances.\n", maxInstances)
 output, err := rdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{MaxRecords: aws.Int32(maxInstances)})
 if err != nil {
  fmt.Printf("Couldn't list DB instances: %v\n", err)
  return
 }
 if len(output.DBInstances) == 0 {
  fmt.Println("No DB instances found.")
 } else {
  for _, instance := range output.DBInstances {
   fmt.Printf("DB instance %v has database %v.\n",
 *instance.DBInstanceIdentifier,
    *instance.DBName)
  }
 }
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }
```

```
    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " +
 instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Purpose

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational
 Database Service
(Amazon RDS) to list the databases in your account.
"""
```

```
import boto3
from botocore.exceptions import ClientError

# Create an RDS client
rds_client = boto3.client("rds")

# Create a paginator for the describe_db_instances operation
paginator = rds_client.get_paginator("describe_db_instances")

try:
    # Use the paginator to get a list of DB instances
    response_iterator = paginator.paginate(
        PaginationConfig={
            "MaxItems": 123,
            "PageSize": 50,  # Adjust PageSize as needed
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response
    instances_found = False
    for page in response_iterator:
        if "DBInstances" in page and page["DBInstances"]:
            instances_found = True
            print("Your RDS instances are:")
            for db in page["DBInstances"]:
                print(db["DBInstanceIdentifier"])

    if not instances_found:
        print("No RDS instances found!")

except ClientError as e:
    print(f"Couldn't list RDS instances. Here's why: {e.response['Error']
['Message']}")
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds'
require 'logger'

# RDSManager is a class responsible for managing RDS operations
# such as listing all RDS DB instances in the current AWS account.
class RDSManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all RDS DB instances in the current AWS account.
  def list_db_instances
    @logger.info('Listing RDS DB instances')

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
 (#{instance.db_instance_status})")
      end
    end
```

```
  end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para Ruby .

Aprenda as noções básicas do Amazon RDS com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um grupo de parâmetros de banco de dados e defina os valores dos parâmetros.
- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.
- Exclua a instância e o grupo de parâmetros.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
/// <summary>
/// Scenario for RDS DB instance example.
```

```
/// </summary>
public class RDSInstanceScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Returns a list of the available DB engine families using the
 DescribeDBEngineVersionsAsync method.
    2.  Selects an engine family and creates a custom DB parameter group using
 the CreateDBParameterGroupAsync method.
    3.  Gets the parameter groups using the DescribeDBParameterGroupsAsync
 method.
    4.  Gets parameters in the group using the DescribeDBParameters method.
    5.  Parses and displays parameters in the group.
    6.  Modifies both the auto_increment_offset and auto_increment_increment
 parameters
        using the ModifyDBParameterGroupAsync method.
    7.  Gets and displays the updated parameters using the DescribeDBParameters
 method with a source of "user".
    8.  Gets a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Displays and selects from a list of micro instance classes available for
 the selected engine and version.
    10. Creates an RDS DB instance that contains a MySql database and uses the
 parameter group
        using the CreateDBInstanceAsync method.
    11. Waits for DB instance to be ready using the DescribeDBInstancesAsync
 method.
    12. Prints out the connection endpoint string for the new DB instance.
    13. Creates a snapshot of the DB instance using the CreateDBSnapshotAsync
 method.
    14. Waits for DB snapshot to be ready using the DescribeDBSnapshots method.
    15. Deletes the DB instance using the DeleteDBInstanceAsync method.
    16. Waits for DB instance to be deleted using the DescribeDbInstances method.
    17. Deletes the parameter group using the DeleteDBParameterGroupAsync.
    */

    private static readonly string sepBar = new('-', 80);
    private static RDSWrapper rdsWrapper = null!;
    private static ILogger logger = null!;
    private static readonly string engine = "mysql";
    static async Task Main(string[] args)
```

```
    {
        // Set up dependency injection for the Amazon RDS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
                    .AddTransient<RDSWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger<RDSInstanceScenario>();

        rdsWrapper = host.Services.GetRequiredService<RDSWrapper>();

        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) DB
instance scenario example.");
        Console.WriteLine(sepBar);

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamily();

            var parameterGroup = await
CreateDbParameterGroup(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroup(parameterGroup.DBParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await ModifyParameters(parameterGroup.DBParameterGroupName,
parameters);
```

```
            await
DescribeUserSourceParameters(parameterGroup.DBParameterGroupName);

            var engineVersionChoice = await
ChooseDbEngineVersion(parameterGroupFamily);

            var instanceChoice = await ChooseDbInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            var newInstance = await CreateRdsNewInstance(parameterGroup, engine,
engineVersionChoice.EngineVersion,
                instanceChoice.DBInstanceClass, newInstanceIdentifier);
            if (newInstance != null)
            {
                DisplayConnectionString(newInstance);

                await CreateSnapshot(newInstance);

                await DeleteRdsInstance(newInstance);
            }

            await DeleteParameterGroup(parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the RDS DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamily()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await rdsWrapper.DescribeDBEngineVersions(engine);
```

```
        Console.WriteLine("1. The following is a list of available DB parameter
group families:");
        int i = 1;
        var parameterGroupFamilies = engines.GroupBy(e =>
e.DBParameterGroupFamily).ToList();
        foreach (var parameterGroupFamily in parameterGroupFamilies)
        {
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("Select an available DB parameter group family by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBParameterGroup> CreateDbParameterGroup(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

        var parameterGroup = await rdsWrapper.CreateDBParameterGroup(
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            dbParameterGroupFamily, "New example parameter group");
```

```
        var groupInfo =
            await rdsWrapper.DescribeDBParameterGroups(parameterGroup
                .DBParameterGroupName);

        Console.WriteLine(
            $"3. New DB parameter group: \n\t{groupInfo[0].Description}, \n\tARN
{groupInfo[0].DBParameterGroupArn}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameterNames">Optional specific names of parameters to
describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>> DescribeParametersInGroup(string
parameterGroupName, List<string>? parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
```

```
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParameters(string parameterGroupName,
List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        foreach (var p in parameters)
        {
            if (p.IsModifiable && p.DataType == "integer")
            {
                int newValue = 0;
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    Int32.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        await rdsWrapper.ModifyDBParameterGroup(parameterGroupName, parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
```

```
    public static async Task DescribeUserSourceParameters(string
parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe user source parameters in the group.");

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName, "user");


        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }


    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDbEngineVersion(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await rdsWrapper.DescribeDBEngineVersions(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. Engine: {version.Engine} Version
{version.EngineVersion}.");
```

```
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDbInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await rdsWrapper.DescribeOrderableDBInstanceOptions(engine,
 engineVersion);

        Console.WriteLine($"8. Available micro DB instance classes for engine
 {engine} and version {engineVersion}:");
        int i = 1;

        // Filter to micro instances for this example.
        allowedInstances = allowedInstances
            .Where(i => i.DBInstanceClass.Contains("micro")).ToList();

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
```

```
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("9. Select an available DB instance class by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new RDS DB instance.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB
 instance.</param>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateRdsNewInstance(DBParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string instanceClass, string
 instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"10. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await rdsWrapper.DescribeDBInstances();
        isInstanceReady = instances.FirstOrDefault(i =>
```

```
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
        }
        else
        {
            Console.WriteLine("Please enter an admin user name:");
            var username = Console.ReadLine();

            Console.WriteLine("Please enter an admin password:");
            var password = Console.ReadLine();

            newInstance = await rdsWrapper.CreateDBInstance(
                "ExampleInstance",
                instanceIdentifier,
                parameterGroup.DBParameterGroupName,
                engineName,
                engineVersion,
                instanceClass,
                20,
                username,
                password
            );

            // 11. Wait for the DB instance to be ready.

            Console.WriteLine("11. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                instances = await
rdsWrapper.DescribeDBInstances(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
                newInstance = instances.First();
                Thread.Sleep(30000);
            }
        }

        Console.WriteLine(sepBar);
```

```
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to use to get a connection
 string.</param>
    public static void DisplayConnectionString(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("12. New DB instance connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {instance.Endpoint.Address} -P
 {instance.Endpoint.Port} "
            + $"-u {instance.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an RDS DB instance.
    /// </summary>
    /// <param name="instance">DB instance to use when creating a snapshot.</
param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBSnapshot> CreateSnapshot(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"13. Creating snapshot from DB instance
 {instance.DBInstanceIdentifier}.");
        var snapshot = await
 rdsWrapper.CreateDBSnapshot(instance.DBInstanceIdentifier, "ExampleSnapshot-" +
 DateTime.Now.Ticks);

        // Wait for the snapshot to be available
        bool isSnapshotReady = false;

        Console.WriteLine($"14. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
```

```
            var snapshots = await
rdsWrapper.DescribeDBSnapshots(instance.DBInstanceIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
            Thread.Sleep(30000);
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Delete an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteRdsInstance(DBInstance newInstance)
    {
        Console.WriteLine(sepBar);
        // Delete the DB instance.
        Console.WriteLine($"15. Delete the DB instance
{newInstance.DBInstanceIdentifier}.");
        await rdsWrapper.DeleteDBInstance(newInstance.DBInstanceIdentifier);

        // Wait for the DB instance to delete.
        Console.WriteLine($"16. Waiting for the DB instance to delete...");
        bool isInstanceDeleted = false;

        while (!isInstanceDeleted)
        {
            var instance = await rdsWrapper.DescribeDBInstances();
            isInstanceDeleted = instance.All(i => i.DBInstanceIdentifier !=
newInstance.DBInstanceIdentifier);
            Thread.Sleep(30000);
        }

        Console.WriteLine("DB instance deleted.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
```

```
    /// Delete a DB parameter group.
    /// </summary>
    /// <param name="parameterGroup">The parameter group to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteParameterGroup(DBParameterGroup
 parameterGroup)
    {
        Console.WriteLine(sepBar);
        // Delete the parameter group.
        Console.WriteLine($"17. Delete the DB parameter group
 {parameterGroup.DBParameterGroupName}.");
        await
 rdsWrapper.DeleteDBParameterGroup(parameterGroup.DBParameterGroupName);

        Console.WriteLine(sepBar);
    }
```

Os métodos de wrapper usados pelo cenário para as ações de instância de banco de dados.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 DB instance operations.
/// </summary>
public partial class RDSWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public RDSWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }


    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
 engine,
```

```
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }



    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }



    /// <summary>
```

```
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }



    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB)
to allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
```

```
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }



    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

Os métodos de wrapper usados pelo cenário para o grupos de parâmetros de banco de dados.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 parameter groups.
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string
 name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }



    /// <summary>
    /// Create a new DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
```

```
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }



    /// <summary>
    /// Update a DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }



    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
 group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
```

```
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }



    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter
 group.</param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
 dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

Os métodos de wrapper usados pelo cenário para as ações de snapshot de banco de dados.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 snapshots.
/// </summary>
```

```
public partial class RDSWrapper
{

    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
 string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }



    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
 dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
```

```
        return results;
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon RDS instance and demonstrates several
 operations
```

```
//! on that instance.
/*!
 \sa gettingStartedWithDBInstances()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::gettingStartedWithDBInstances(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon RDS)"
              << std::endl;
    std::cout << "get started with DB instances demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB parameter group named '" <<
              PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB parameter group already exists.
        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' does not exist." << std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available engine versions for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available database engine versions for " <<
DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB parameter group.
        Aws::RDS::Model::CreateDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example parameter group.");

        Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
                client.CreateDBParameterGroup(request);

        if (outcome.IsSuccess()) {
```

```
            std::cout << "The DB parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your parameter group."
              << std::endl;

    Aws::String marker;
    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB parameter group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX, NO_SOURCE,
                         autoIncrementParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value in the range ") +
                        autoIncParameter.GetAllowedValues() + ": ",
```

```
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the group.
        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
userParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
```

```
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB instance." << std::endl;

    Aws::RDS::Model::DBInstance dbInstance;
    // 7.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters):
");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of available engine versions.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                 client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group are:" <<
std::endl;

        int index = 1;
```

```
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
            std::cout << "  " << index << ": " <<
engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                                1];

        Aws::String dbInstanceClass;
        // 9.  Get a list of micro instance classes.
        if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
                                        engineVersion.GetEngineVersion(),
                                        dbInstanceClass,
                                        client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB instance is configured to use your custom parameter
group '"
                  << PARAMETER_GROUP_NAME << "',\n"
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ",\n"
                  << "selected DB instance class '" << dbInstanceClass << "',"
                  << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
                  << " storage.\nThis typically takes several minutes." <<
std::endl;

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
```

```
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
```

```
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    printAsterisksLine();

    // 12. Display the connection string that can be used to connect a 'mysql'
 shell to the database.
    displayConnection(dbInstance);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB instance (y/n)? ")) {
        Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 13. Create a snapshot of the DB instance.
            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for snapshot to become available." << std::endl;

        Aws::RDS::Model::DBSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                        << counter
                        << " seconds." << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 14. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                        << outcome.GetError().GetMessage()
                        << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            if ((counter % 20) == 0) {
```

```
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB instance and parameter group (y/n)? "))
 {
        result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
    }

    return result;
}


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
```

```
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
```

```
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());
```

```
    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

```
//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
```

```
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database
 engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::cleanUpResources(const Aws::String &parameterGroupName,
                                   const Aws::String &dbInstanceIdentifier,
                                   const Aws::RDS::RDSClient &client) {
    bool result = true;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 15. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        std::cout
                << "Waiting for DB instance to delete before deleting the
parameter group."
                << std::endl;
        std::cout << "This may take a while." << std::endl;

        int counter = 0;
        Aws::RDS::Model::DBInstance dbInstance;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 800) {
                std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                          << " seconds." << std::endl;
                return false;
            }

            dbInstance = Aws::RDS::Model::DBInstance();
            // 16. Wait for the DB instance to be deleted.
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }

            if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) ==
0) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (dbInstance.DBInstanceIdentifierHasBeenSet());
```

```
    }

    if (!parameterGroupName.empty()) {
        // 17. Delete the parameter group.
        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions

- ModifyDBParameterGroup

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "fmt"
 "log"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/rds/actions"
 "github.com/google/uuid"
)

// GetStartedInstances is an interactive example that shows you how to use the
 AWS SDK for Go
// with Amazon Relation Database Service (Amazon RDS) to do the following:
//
//  1. Create a custom DB parameter group and set parameter values.
//  2. Create a DB instance that is configured to use the parameter group. The DB
 instance
//     also contains a database.
//  3. Take a snapshot of the DB instance.
//  4. Delete the DB instance and parameter group.
```

```
type GetStartedInstances struct {
 sdkConfig  aws.Config
 instances  actions.DbInstances
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedInstances constructs a GetStartedInstances instance from a
 configuration.
// It uses the specified config to get an Amazon RDS
// client and create wrappers for the actions used in the scenario.
func NewGetStartedInstances(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedInstances {
 rdsClient := rds.NewFromConfig(sdkConfig)
 return GetStartedInstances{
  sdkConfig:  sdkConfig,
  instances:  actions.DbInstances{RdsClient: rdsClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedInstances) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 instanceName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Relational Database Service (Amazon RDS) DB
 Instance demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 instance := scenario.CreateInstance(ctx, instanceName, dbEngine, dbName,
 parameterGroup)
```

```
 scenario.DisplayConnection(instance)
 scenario.CreateSnapshot(ctx, instance)
 scenario.Cleanup(ctx, instance, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedInstances) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBParameterGroup {

 log.Printf("Checking for an existing DB parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB parameter group.")
  _, err = scenario.instances.CreateParameterGroup(
```

```
    ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
   if err != nil {
    panic(err)
   }
   parameterGroup, err = scenario.instances.GetParameterGroup(ctx,
  parameterGroupName)
   if err != nil {
    panic(err)
   }
  }
  log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
  log.Printf("\tName: %v\n", *parameterGroup.DBParameterGroupName)
  log.Printf("\tARN: %v\n", *parameterGroup.DBParameterGroupArn)
  log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
  log.Printf("\tDescription: %v\n", *parameterGroup.Description)
  log.Println(strings.Repeat("-", 88))
  return parameterGroup
}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedInstances) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
```

```
  }
 }
 err = scenario.instances.UpdateParameters(ctx, parameterGroupName, updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("To get a list of parameters that you set previously, specify a
 source of 'user'.")
 userParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
 "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateInstance shows how to create a DB instance that contains a database of a
// specified type. The database is also configured to use a custom DB parameter
 group.
func (scenario GetStartedInstances) CreateInstance(ctx context.Context,
 instanceName string, dbEngine string,
 dbName string, parameterGroup *types.DBParameterGroup) *types.DBInstance {

 log.Println("Checking for an existing DB instance.")
 instance, err := scenario.instances.GetInstance(ctx, instanceName)
 if err != nil {
  panic(err)
 }
 if instance == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator username for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.AskPassword(
   "Enter a password for the administrator (at least 8 characters): ", 7)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine,
   *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
```

```
  engineChoices = append(engineChoices, *engine.EngineVersion)
 }
 engineIndex := scenario.questioner.AskChoice(
  "The available engines for your parameter group are:\n", engineChoices)
 engineSelection := engineVersions[engineIndex]
 instOpts, err := scenario.instances.GetOrderableInstances(ctx,
*engineSelection.Engine,
  *engineSelection.EngineVersion)
 if err != nil {
  panic(err)
 }
 optSet := map[string]struct{}{}
 for _, opt := range instOpts {
  if strings.Contains(*opt.DBInstanceClass, "micro") {
   optSet[*opt.DBInstanceClass] = struct{}{}
  }
 }
 var optChoices []string
 for opt := range optSet {
  optChoices = append(optChoices, opt)
 }
 sort.Strings(optChoices)
 optIndex := scenario.questioner.AskChoice(
  "The available micro DB instance classes for your database engine are:\n",
optChoices)
 storageType := "standard"
 allocatedStorage := int32(5)
 log.Printf("Creating a DB instance named %v and database %v.\n"+
  "The DB instance is configured to use your custom parameter group %v,\n"+
  "selected engine %v,\n"+
  "selected DB instance class %v,"+
  "and %v GiB of %v storage.\n"+
  "This typically takes several minutes.",
  instanceName, dbName, *parameterGroup.DBParameterGroupName,
*engineSelection.EngineVersion,
  optChoices[optIndex], allocatedStorage, storageType)
 instance, err = scenario.instances.CreateInstance(
  ctx, instanceName, dbName, *engineSelection.Engine,
*engineSelection.EngineVersion,
  *parameterGroup.DBParameterGroupName, optChoices[optIndex], storageType,
  allocatedStorage, adminUsername, adminPassword)
 if err != nil {
  panic(err)
 }
```

```
  for *instance.DBInstanceStatus != "available" {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(ctx, instanceName)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Instance created and available.")
 }
 log.Println("Instance data:")
 log.Printf("\tDBInstanceIdentifier: %v\n", *instance.DBInstanceIdentifier)
 log.Printf("\tARN: %v\n", *instance.DBInstanceArn)
 log.Printf("\tStatus: %v\n", *instance.DBInstanceStatus)
 log.Printf("\tEngine: %v\n", *instance.Engine)
 log.Printf("\tEngine version: %v\n", *instance.EngineVersion)
 log.Println(strings.Repeat("-", 88))
 return instance
}

// DisplayConnection displays connection information about a DB instance and tips
// on how to connect to it.
func (scenario GetStartedInstances) DisplayConnection(instance *types.DBInstance)
 {
 log.Println(
  "You can now connect to your database by using your favorite MySQL client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your DB instance. Pass the endpoint,\n" +
   "port, and administrator username to 'mysql'. Then, enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *instance.Endpoint.Address, instance.Endpoint.Port, *instance.MasterUsername)
 log.Println("For more information, see the User Guide for RDS:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB instance snapshot and wait until it's
 available.
func (scenario GetStartedInstances) CreateSnapshot(ctx context.Context, instance
 *types.DBInstance) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB instance (y/n)? ", "y") {
```

```
  snapshotId := fmt.Sprintf("%v-%v", *instance.DBInstanceIdentifier,
 scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.instances.CreateSnapshot(ctx,
 *instance.DBInstanceIdentifier, snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.instances.GetSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBSnapshotIdentifier: %v\n", *snapshot.DBSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBInstanceIdentifier: %v\n", *snapshot.DBInstanceIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance and DB parameter group.
// Before the DB parameter group can be deleted, all associated DB instances must
 first be deleted.
func (scenario GetStartedInstances) Cleanup(
 ctx context.Context, instance *types.DBInstance, parameterGroup
 *types.DBParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance and parameter group (y/n)? ",
 "y") {
  log.Printf("Deleting database instance %v.\n", *instance.DBInstanceIdentifier)
  err := scenario.instances.DeleteInstance(ctx, *instance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
```

```
    "Waiting for the DB instance to delete. This typically takes several
 minutes.")
  for instance != nil {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(ctx,
 *instance.DBInstanceIdentifier)
   if err != nil {
    panic(err)
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBParameterGroupName)
  err = scenario.instances.DeleteParameterGroup(ctx,
 *parameterGroup.DBParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Amazon RDS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}


// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}



// CreateParameterGroup creates a DB parameter group that is based on the
 specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
```

```
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context,
 parameterGroupName string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
```

```
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
```

```
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName
 string) (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string,
 dbInstanceClass string,
 storageType string, allocatedStorage int32, adminName string, adminPassword
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
```

```
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
```

```
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine
 string, parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
```

```
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions,
 output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute várias operações.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
```

```
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySql database and uses
```

```
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
 <dbName> <dbSnapshotIdentifier> <secretName>

                Where:
                    dbGroupName - The database group name.\s
                    dbParameterGroupFamily - The database parameter group name
 (for example, mysql8.0).
                    dbInstanceIdentifier - The database instance identifier\s
                    dbName - The database name.\s
                    dbSnapshotIdentifier - The snapshot identifier.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials"
                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceIdentifier = args[2];
        String dbName = args[3];
        String dbSnapshotIdentifier = args[4];
        String secretName = args[5];

        Gson gson = new Gson();
```

```
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String masterUsername = user.getUsername();
        String masterUserPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon RDS example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbParameterGroups(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbParameters(rdsClient, dbGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBParas(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated value");
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get a list of micro instance classes available for
the selected engine");
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "9. Create an RDS database instance that contains a MySql
database and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
                masterUserPassword);
        System.out.println("The ARN of the new database is " + dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while
database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
```

```
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Delete the DB instance.
    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();
```

```
            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the snapshot instance is available.
    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
            String dbSnapshotIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
                List<DBSnapshot> snapshotList = response.dbSnapshots();
                for (DBSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
```

```
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Create a database instance and return the ARN of the database.
    public static String createDatabaseInstance(RdsClient rdsClient,
            String dbGroupName,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported
version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();
```

```
            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }

        return "";
    }

    // Get a list of micro instances.
    public static void getMicroInstances(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("mysql")
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient

.describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption :
orderableDBInstances) {
                System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
                System.out.println("The engine description is " +
dbInstanceOption.engine());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
```

```
   public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
       try {
           DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .engine("mysql")
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
           List<DBEngineVersion> dbEngines = response.dbEngineVersions();
           for (DBEngineVersion dbEngine : dbEngines) {
               System.out.println("The engine version is " +
dbEngine.engineVersion());
               System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   // Modify auto_increment_offset and auto_increment_increment parameters.
   public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
       try {
           Parameter parameter1 = Parameter.builder()
                   .parameterName("auto_increment_offset")
                   .applyMethod("immediate")
                   .parameterValue("5")
                   .build();

           List<Parameter> paraList = new ArrayList<>();
           paraList.add(parameter1);
           ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .parameters(paraList)
                   .build();

           ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
```

```
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String
dbGroupName, int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
```

```
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
```

```
                    .build();

            CreateDbParameterGroupResponse response =
 rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
 response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
 DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this code example, set up your development environment, including
 your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

```
This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking
 the createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment
 parameters by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the
 describeDbEngineVersions method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance
 that contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint
 value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
 <dbName> <dbSnapshotIdentifier><secretName>

        Where:
            dbGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceIdentifier - The database instance identifier.
            dbName -  The database name.
```

```
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val dbSnapshotIdentifier = args[4]
    val secretName = args[5]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeDBEngines()

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)
```

```
    println("8. Get a list of micro instance classes available for the selected
 engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database
 and uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
 username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
```

```
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

```
// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
```

```
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
```

```
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is
 ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
```

```
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is
 ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
```

```
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset
 or auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    System.out.println("*** The parameter value is
 ${para.parameterValue}")
                    System.out.println("*** The parameter data type is
 ${para.dataType}")
                    System.out.println("*** The parameter description is
 ${para.description}")
                    System.out.println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
```

```
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
```

```
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the
 database engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-west-2" }.use
 { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance

- DeleteDBParameterGroup
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class RdsInstanceScenario:
    """Runs a scenario that shows how to get started using Amazon RDS DB
 instances."""

    def __init__(self, instance_wrapper):
        """
        :param instance_wrapper: An object that wraps Amazon RDS DB instance
 actions.
        """
        self.instance_wrapper = instance_wrapper

    def create_parameter_group(self, parameter_group_name, db_engine):
        """
        Shows how to get available engine versions for a specified database
 engine and
        create a DB parameter group that is compatible with a selected engine
 family.
```

```
        :param parameter_group_name: The name given to the newly created
parameter group.
        :param db_engine: The database engine to use as a basis.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB instance parameter group named
{parameter_group_name}."
        )
        parameter_group = self.instance_wrapper.get_parameter_group(
            parameter_group_name
        )
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions =
self.instance_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a parameter group.")
            self.instance_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.instance_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group {parameter_group['DBParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def update_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
```

```
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.instance_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_instance(self, instance_name, db_name, db_engine,
parameter_group):
        """
        Shows how to create a DB instance that contains a database of a specified
        type and is configured to use a custom DB parameter group.

        :param instance_name: The name given to the newly created DB instance.
        :param db_name: The name given to the created database.
        :param db_engine: The engine of the created database.
        :param parameter_group: The parameter group that is associated with the
DB instance.
        :return: The newly created DB instance.
        """
        print("Checking for an existing DB instance.")
        db_inst = self.instance_wrapper.get_db_instance(instance_name)
```

```
        if db_inst is None:
            print("Let's create a DB instance.")
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.instance_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [ver["EngineVersion"] for ver in engine_versions]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            engine_selection = engine_versions[engine_index]
            print(
                "The available micro DB instance classes for your database engine
are:"
            )
            inst_opts = self.instance_wrapper.get_orderable_instances(
                engine_selection["Engine"], engine_selection["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"]
                    for opt in inst_opts
                    if "micro" in opt["DBInstanceClass"]
                }
            )
            inst_index = q.choose(
                "Which micro DB instance class do you want to use? ",
inst_choices
            )
            group_name = parameter_group["DBParameterGroupName"]
            storage_type = "standard"
            allocated_storage = 5
            print(
                f"Creating a DB instance named {instance_name} and database
{db_name}.\n"
```

```
                f"The DB instance is configured to use your custom parameter
 group {group_name},\n"
                f"selected engine {engine_selection['EngineVersion']},\n"
                f"selected DB instance class {inst_choices[inst_index]},"
                f"and {allocated_storage} GiB of {storage_type} storage.\n"
                f"This typically takes several minutes."
            )
            db_inst = self.instance_wrapper.create_db_instance(
                db_name,
                instance_name,
                group_name,
                engine_selection["Engine"],
                engine_selection["EngineVersion"],
                inst_choices[inst_index],
                storage_type,
                allocated_storage,
                admin_username,
                admin_password,
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(instance_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(db_inst):
        """
        Displays connection information about a DB instance and tips on how to
        connect to it.

        :param db_inst: The DB instance to display.
        """
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your DB instance. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
```

```
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {db_inst['Endpoint']['Address']} -P
 {db_inst['Endpoint']['Port']} "
            f"-u {db_inst['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Amazon RDS:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL"
        )
        print("-" * 88)

    def create_snapshot(self, instance_name):
        """
        Shows how to create a DB instance snapshot and wait until it's available.

        :param instance_name: The name of a DB instance to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB instance (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{instance_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.instance_wrapper.create_snapshot(snapshot_id,
 instance_name)
            while snapshot.get("Status") != "available":
                wait(10)
                snapshot = self.instance_wrapper.get_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)

    def cleanup(self, db_inst, parameter_group_name):
        """
        Shows how to clean up a DB instance and parameter group.
        Before the parameter group can be deleted, all associated DB instances
 must first
        be deleted.
```

```
        :param db_inst: The DB instance to delete.
        :param parameter_group_name: The DB parameter group to delete.
        """
        if q.ask(
            "\nDo you want to delete the DB instance and parameter group (y/n)?
",
            q.is_yesno,
        ):
            print(f"Deleting DB instance {db_inst['DBInstanceIdentifier']}.")

self.instance_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(
                "Waiting for the DB instance to delete. This typically takes
several minutes."
            )
            while db_inst is not None:
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(
                    db_inst["DBInstanceIdentifier"]
                )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.instance_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, instance_name,
db_name):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS)\n"
            "get started with DB instances demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(parameter_group_name,
db_engine)
        self.update_parameters(parameter_group_name)
        db_inst = self.create_instance(
            instance_name, db_name, db_engine, parameter_group
        )
        self.display_connection(db_inst)
        self.create_snapshot(instance_name)
        self.cleanup(db_inst, parameter_group_name)
```

```
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = RdsInstanceScenario(InstanceWrapper.from_client())
        scenario.run_scenario(
            "mysql",
            "doc-example-parameter-group",
            "doc-example-instance",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Amazon RDS.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
```

```
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter
group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
```

```
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
```

```
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id,
DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
```

```
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
```

```
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB
instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
```

```
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
```

```
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "rds-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "rds-scenario",
```

```
            dependencies: [
                .product(name: "AWSRDS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo de código Swift, `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Relational Database Service (RDS).
//

import ArgumentParser
import Foundation
import AWSRDS

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"
    @Option(help: "The username to use for the database administrator.")
    var dbUsername = "admin"
    @Option(help: "The password to use for the database administrator.")
    var dbPassword: String

    static var configuration = CommandConfiguration(
        commandName: "rds-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon RDS
 Instances
        using the AWS SDK for Swift.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
```

```
        let example = try await Example(region: awsRegion, username: dbUsername,
 password: dbPassword)

        await example.run()
    }
}

class Example {
    let rdsClient: RDSClient

    // Storage for AWS RDS properties

    let dbUsername: String
    let dbPassword: String
    var dbInstanceIdentifier: String
    var dbSnapshotIdentifier: String
    var dbParameterGroupName: String
    var dbParameterGroup: RDSClientTypes.DBParameterGroup?
    var selectedEngineVersion: String?

    init(region: String, username: String, password: String) async throws{
        let rdsConfig = try await RDSClient.RDSClientConfiguration(region:
 region)
        rdsClient = RDSClient(config: rdsConfig)

        dbUsername = username
        dbPassword = password
        dbParameterGroupName = ""
        dbInstanceIdentifier = ""
        dbSnapshotIdentifier = ""
    }

    /// The example's main body.
    func run() async {
        var parameterGroupFamilies: Set<String> = []

        //=====================================================================
        // 1. Get available database engine families for MySQL.
        //=====================================================================

        let engineVersions = await getDBEngineVersions(engineName: "mysql")

        for version in engineVersions {
            if version.dbParameterGroupFamily != nil {
```

```
                parameterGroupFamilies.insert(version.dbParameterGroupFamily!)
            }
        }

        if engineVersions.count > 0 {
            selectedEngineVersion = engineVersions.last!.engineVersion
        } else {
            print("*** Unable to find a valid database engine version. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Found \(parameterGroupFamilies.count) parameter group families:")
        for family in parameterGroupFamilies {
            print("    \(family)")
        }

        //=====================================================================
        // 2. Select an engine family and create a custom DB parameter group.
        //    We select a family by sorting the set of family names, then
        //    choosing the last one.
        //=====================================================================

        let sortedFamilies = parameterGroupFamilies.sorted()

        guard let selectedFamily = sortedFamilies.last else {
            print("*** Unable to find a database engine family. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Selected database engine family \(selectedFamily)")

        dbParameterGroupName = tempName(prefix: "rds-example")
        print("Creating a database parameter group named \(dbParameterGroupName)
using \(selectedFamily)")
        dbParameterGroup = await createDBParameterGroup(groupName:
dbParameterGroupName,
                                                        familyName:
selectedFamily)

        //=====================================================================
```

```
        // 3. Get the parameter group's details.
        //=====================================================================

        print("Getting the database parameter group list...")
        let dbParameterGroupList = await describeDBParameterGroups(groupName:
dbParameterGroupName)
        guard let dbParameterGroupList else {
            await cleanUp()
            return
        }

        print("Found \(dbParameterGroupList.count) parameter groups...")
        for group in dbParameterGroupList {
            print("    \(group.dbParameterGroupName ?? "<unknown>")")
        }
        print()

        //=====================================================================
        // 4. Get a list of the parameter group's parameters. This list is
        //    likely to be long, so use pagination. Find the
        //    auto_increment_offset and auto_increment_increment parameters.
        //=====================================================================

        let parameters = await describeDBParameters(groupName:
dbParameterGroupName)

        //=====================================================================
        // 5. Parse and display each parameter's name, description, and
        //    allowed values.
        //=====================================================================

        for parameter in parameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }

            if name == "auto_increment_offset" || name ==
"auto_increment_increment" {
                print("Parameter \(name):")
                print("          Value: \(parameter.parameterValue ??
"<undefined>")")
                print("      Data type: \(parameter.dataType ?? "<unknown>")")
```

```
                print("    Description: \(parameter.description ?? "")")
                print(" Allowed values: \(parameter.allowedValues ??
"<unspecified">)")
                print(String(repeating: "=", count: 78))
            }
        }

        //=====================================================================
        // 6. Modify both the auto_increment_offset and
        //    auto_increment_increment parameters in one call in the custom
        //    parameter group. Set their parameterValue fields to a new
        //    permitted value.
        //=====================================================================

        print("Setting auto_increment_offset and auto_increment_increment both to
5...")
        await modifyDBParameters(groupName: dbParameterGroupName)

        //=====================================================================
        // 7. Get and display the updated parameters, specifying a source of
        //    "user" to get only the modified parameters.
        //=====================================================================

        let updatedParameters = await describeDBParameters(groupName:
dbParameterGroupName, source: "user")

        for parameter in updatedParameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }

            print("Parameter \(name):")
            print("          Value: \(parameter.parameterValue ??
"<undefined>")")
            print("      Data type: \(parameter.dataType ?? "<unknown>")")
            print("    Description: \(parameter.description ?? "")")
            print(" Allowed values: \(parameter.allowedValues ??
"<unspecified">)")
            print(String(repeating: "=", count: 78))
        }

        //=====================================================================
```

```
        // 8. Get a list of allowed engine versions using
        //    DescribeRDSEngineVersions.
        //=====================================================================

        await listAllowedEngines(family: selectedFamily)

        //=====================================================================
        // 9. Get a list of micro instance classes available for the selected
        //    engine and engine version.
        //=====================================================================

        let dbInstanceClass = await chooseMicroInstance(engine: "mysql",
engineVersion: selectedEngineVersion)
        guard let dbInstanceClass else {
            print("Did not get a valid instance class. Canceling operations.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 10. Create an RDS database that contains a MySQL database and uses
        //     the parameter group we created.
        //=====================================================================

        print("Creating the database instance...")

        guard let instanceClass = dbInstanceClass.dbInstanceClass else {
            print("Instance class name is unknown. Canceling operations.")
            await cleanUp()
            return
        }

        dbInstanceIdentifier = tempName(prefix: "sample-identifier")
        let dbInstanceArn = await createDBInstance(
            name: "SampleDatabase\(Int.random(in: 1000000..<1000000000))",
            instanceIdentifier: dbInstanceIdentifier,
            parameterGroupName: dbParameterGroupName,
            engine: "mysql",
            engineVersion: selectedEngineVersion!,
            instanceClass: instanceClass,
            username: dbUsername,
            password: dbPassword
        )
```

```
        if dbInstanceArn == nil {
            await cleanUp()
            return
        }

        //=====================================================================
        // 11. Wait for the database instance to be ready by calling
        //     DescribeDBInstances repeatedly until it reports
        //     dbInstanceStatus as "available". This can take upwards of 10
        //     minutes, let the user know that.
        //=====================================================================

        guard let endpoint = await waitUntilDBInstanceReady(instanceIdentifier:
dbInstanceIdentifier) else {
            print("\nDid not get a valid endpoint from AWS RDS.")
            await cleanUp()
            return
        }

        guard let endpointAddress = endpoint.address else {
            print("\nNo endpoint address returned.")
            await cleanUp()
            return
        }
        guard let endpointPort = endpoint.port else {
            print("\nNo endpoint port returned.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 12. Display connection information for the database instance.
        //=====================================================================

        print("\nTo connect to the new database instance using 'mysql' from the
shell:")
        print("    mysql -h \(endpointAddress) -P \(endpointPort) -u
\(self.dbUsername)")

        //=====================================================================
        // 13. Create a snapshot of the database instance.
        //=====================================================================

        dbSnapshotIdentifier = tempName(prefix: "sample-snapshot")
```

```
        await createDBSnapshot(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        //=====================================================================
        // 14. Wait for the snapshot to be ready.
        //=====================================================================

        await waitUntilDBSnapshotReady(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        // That's it! Clean up and exit!

        print("Example complete! Cleaning up...")
        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items.
    func cleanUp() async {
        print("Deleting the database instance \(dbInstanceIdentifier)...")
        await deleteDBInstance(instanceIdentifier: dbInstanceIdentifier)
        await waitUntilDBInstanceDeleted(instanceIdentifier:
dbInstanceIdentifier)

        print("Deleting the database parameter group \(dbParameterGroupName)...")
        await deleteDBParameterGroup(groupName: dbParameterGroupName)
    }

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
[RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
```

```
        } catch {
            return []
        }
    }

    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
        } catch {
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
    func describeDBParameterGroups(groupName: String) async ->
[RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
```

```
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
[RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
\(error.localizedDescription)")
            return []
        }

        return parameterList
```

```
    }

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
    }

    /// Output a list of the database engine versions supported by the
    /// specified family.
    ///
    /// - Parameter family: The family for which to list allowed database
    ///   engines.
    func listAllowedEngines(family: String?) async {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
```

```
                input: DescribeDBEngineVersionsInput(
                    dbParameterGroupFamily: family,
                    engine: "mysql"
                )
            )

            guard let engineVersions = output.dbEngineVersions else {
                print("No engine versions returned.")
                return
            }

            print("Found \(engineVersions.count) database engine versions:")
            for version in engineVersions {
                print("    \(version.engineVersion ?? "<unknown>"):
\(version.dbEngineDescription ?? "")")
            }
        } catch {
            print("*** Error getting database engine version list:
\(error.localizedDescription)")
            return
        }
    }

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
    ///   - engineVersion: The database version for which to list instances.
    ///
    /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
    ///   the selected instance type.
    func chooseMicroInstance(engine: String = "mysql", engineVersion: String? =
nil) async -> RDSClientTypes.OrderableDBInstanceOption? {
        do {
            let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
                input: DescribeOrderableDBInstanceOptionsInput(
                    engine: engine,
                    engineVersion: engineVersion
                )
            )

            var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []
```

```
            for try await page in pages {
                guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                    continue
                }

                for dbInstanceOption in orderableDBInstanceOptions {
                    guard let className = dbInstanceOption.dbInstanceClass else {
                        continue
                    }
                    if className.contains("micro") {
                        optionsList.append(dbInstanceOption)
                    }
                }
            }

            print("Found \(optionsList.count) database instances of 'micro' class
types:")
            for dbInstanceOption in optionsList {
                print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
            }

            return optionsList[0]
        } catch {
            print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Create a new database instance.
    ///
    /// - Parameters:
    ///   - name: The name of the database to create.
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
```

```
   ///   - instanceClass: The memory and compute capacity of the database
   ///     instance, such as `db.m5.large``.
   ///   - username: The admin user's username to establish for the new
   ///     instance.
   ///   - password: The password to use for the specified user's access.
   ///
   /// - Returns: A string indicating the ARN of the newly created database
   ///   instance, or nil if the instance couldn't be created.
   func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                         engine: String, engineVersion: String, instanceClass:
String,
                         username: String, password: String) async -> String? {
       do {
           let output = try await rdsClient.createDBInstance(
               input: CreateDBInstanceInput(
                   allocatedStorage: 100,
                   dbInstanceClass: instanceClass,
                   dbInstanceIdentifier: instanceIdentifier,
                   dbName: name,
                   dbParameterGroupName: parameterGroupName,
                   engine: engine,
                   engineVersion: engineVersion,
                   masterUserPassword: password,
                   masterUsername: username,
                   storageType: "gp2"
               )
           )

           guard let dbInstance = output.dbInstance else {
               print("*** Unable to get the database instance.")
               return nil
           }

           return dbInstance.dbInstanceArn
       } catch {
           print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
           return nil
       }
   }

   /// Wait until the specified database is available to use.
   ///
```

```
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }
```

```
   /// Create a snapshot of the specified name.
   ///
   /// - Parameters:
   ///   - instanceIdentifier: The identifier of the database instance to
   ///     snapshot.
   ///   - snapshotIdentifier: A unique identifier to give the newly-created
   ///     snapshot.
   func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
async {
       do {
           let output = try await rdsClient.createDBSnapshot(
               input: CreateDBSnapshotInput(
                   dbInstanceIdentifier: instanceIdentifier,
                   dbSnapshotIdentifier: snapshotIdentifier
               )
           )

           guard let snapshot = output.dbSnapshot else {
               print("No snapshot returned.")
               return
           }

           print("The snapshot has been created with ID
\(snapshot.dbiResourceId ?? "<unknown>")")
       } catch {
           print("*** Unable to create the database snapshot named
\(snapshotIdentifier): \(error.localizedDescription)")
       }
   }

   /// Wait until the specified database snapshot is available to use.
   ///
   /// - Parameters:
   ///   - instanceIdentifier: The identifier of the database for which the
   ///     snapshot was taken.
   ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
   func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
String) async {
       var snapshotReady = false

       putString("Waiting for the snapshot to be ready...")

       do {
```

```
            while !snapshotReady {
                let output = try await rdsClient.describeDBSnapshots(
                    input: DescribeDBSnapshotsInput(
                        dbInstanceIdentifier: instanceIdentifier,
                        dbSnapshotIdentifier: snapshotIdentifier
                    )
                )

                guard let snapshotList = output.dbSnapshots else {
                    return
                }

                for snapshot in snapshotList {
                    guard let snapshotReadyStr = snapshot.status else {
                        return
                    }

                    if snapshotReadyStr.contains("available") {
                        snapshotReady = true
                        print()
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
        }
    }

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
```

```
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
        } catch {
            print("*** Error deleting the database instance
\(instanceIdentifier): \(error.localizedDescription)")
        }
    }

    /// Wait until the specified database instance has been deleted.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to wait for.
    func waitUntilDBInstanceDeleted(instanceIdentifier: String) async {
        putString("Waiting for the database instance to be deleted. This may take
a few minutes...")
        do {
            var isDatabaseDeleted = false
            var foundInstance = false

            while !isDatabaseDeleted {
                let output = try await rdsClient.describeDBInstances(input:
DescribeDBInstancesInput())
                guard let instanceList = output.dbInstances else {
                    return
                }

                foundInstance = false

                for instance in instanceList {
                    guard let foundInstanceIdentifier =
instance.dbInstanceIdentifier else {
                        continue
                    }

                    if instanceIdentifier == foundInstanceIdentifier {
                        foundInstance = true
                        break
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
```

```
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
                if !foundInstance {
                    isDatabaseDeleted = true
                    print()
                }
            }
        } catch {
            print("\n*** Error waiting for the database instance to be deleted:
\(error.localizedDescription)")
        }
    }

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName):
\(error.localizedDescription)")
        }
    }

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
```

```
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Print a string to standard output without a trailing newline, and
    /// without buffering.
    ///
    /// - Parameter str: The string to output.
    func putString(_ str: String = "") {
        if str.length >= 1 {
            let data = str.data(using: .utf8)
            guard let data else {
                return
            }
            FileHandle.standardOutput.write(data)
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance

- DeleteDBParameterGroup
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Ações para o Amazon RDS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon RDS com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon RDS e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon RDS usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para ter uma lista completa, consulte a Referência de API do Amazon Relational Database Service.

Exemplos

- Use createdBInstance com um AWS SDK ou CLI
- Use CreateDBParameterGroupcom um AWS SDK ou CLI
- Use createDBSnapshot com um AWS SDK ou CLI
- Use DeletedBInstance com um AWS SDK ou CLI
- Use DeleteDBParameterGroupcom um AWS SDK ou CLI
- Use DescribeAccountAttributescom um AWS SDK ou CLI
- Use DescribeDBEngineVersionscom um AWS SDK ou CLI
- Use DescribedBInstances com um AWS SDK ou CLI
- Use DescribeDBParameterGroupscom um AWS SDK ou CLI
- Use DescribedBParameters com um AWS SDK ou CLI

- Use DescribedBSnapshots com um AWS SDK ou CLI
- Use DescribeOrderableDBInstanceOptionscom um AWS SDK ou CLI
- Use GenerateRDSAuthTokencom um AWS SDK
- Use ModifyDBInstance com um AWS SDK ou CLI
- Use ModifyDBParameterGroupcom um AWS SDK ou CLI
- Use RebootDBInstance com um AWS SDK ou CLI

Use **`createdBInstance`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateDBInstance`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
```

```
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB)
to allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Como criar uma instância de banco de dados

O exemplo de `create-db-instance` a seguir usa as opções necessárias para iniciar uma nova instância de banco de dados.

```
aws rds create-db-instance \
    --db-instance-identifier test-mysql-instance \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username admin \
    --master-user-password secret99 \
    --allocated-storage 20
```

Saída:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-mysql-instance",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 20,
        "PreferredBackupWindow": "12:55-13:25",
        "BackupRetentionPeriod": 1,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-12345abc",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.mysql5.7",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2ff2ff2f",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
```

```
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2c"
                },
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2d"
                },
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                },
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2b"
                },
                "SubnetStatus": "Active"
            }
        ]
    },
    "PreferredMaintenanceWindow": "sun:08:07-sun:08:37",
    "PendingModifiedValues": {
        "MasterUserPassword": "****"
    },
    "MultiAZ": false,
    "EngineVersion": "5.7.22",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",
    "OptionGroupMemberships": [
        {
            "OptionGroupName": "default:mysql-5-7",
            "Status": "in-sync"
        }
    ],
```

```
        "PubliclyAccessible": true,
        "StorageType": "gp2",
        "DbInstancePort": 0,
        "StorageEncrypted": false,
        "DbiResourceId": "db-5555EXAMPLE44444444EXAMPLE",
        "CACertificateIdentifier": "rds-ca-2019",
        "DomainMemberships": [],
        "CopyTagsToSnapshot": false,
        "MonitoringInterval": 0,
        "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:test-mysql-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": []
    }
}
```

Para obter mais informações, consulte Criar uma instância de banco de dados do Amazon RDS no Guia do usuário do Amazon RDS.

- Consulte detalhes da API em CreateDBInstance na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
```

```
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string,
 dbInstanceClass string,
 storageType string, allocatedStorage int32, adminName string, adminPassword
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
```

```
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 *
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbName> <secretName>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    dbName - The database name.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials."
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
 user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }
```

```
    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported
version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            // Loop until the cluster is ready.
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true;
                    else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the database instance is available.
```

```
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
```

```
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB
instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Create a new database instance.
    ///
    /// - Parameters:
    ///   - name: The name of the database to create.
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
    ///   - instanceClass: The memory and compute capacity of the database
    ///     instance, such as `db.m5.large``.
    ///   - username: The admin user's username to establish for the new
    ///     instance.
    ///   - password: The password to use for the specified user's access.
    ///
```

```
    /// - Returns: A string indicating the ARN of the newly created database
    ///   instance, or nil if the instance couldn't be created.
    func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                          engine: String, engineVersion: String, instanceClass:
String,
                          username: String, password: String) async -> String? {
        do {
            let output = try await rdsClient.createDBInstance(
                input: CreateDBInstanceInput(
                    allocatedStorage: 100,
                    dbInstanceClass: instanceClass,
                    dbInstanceIdentifier: instanceIdentifier,
                    dbName: name,
                    dbParameterGroupName: parameterGroupName,
                    engine: engine,
                    engineVersion: engineVersion,
                    masterUserPassword: password,
                    masterUsername: username,
                    storageType: "gp2"
                )
            )

            guard let dbInstance = output.dbInstance else {
                print("*** Unable to get the database instance.")
                return nil
            }

            return dbInstance.dbInstanceArn
        } catch {
            print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência de API do AWS SDK para Swift.

Use **`CreateDBParameterGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateDBParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create a new DB parameter group. Use the action
DescribeDBParameterGroupsAsync
   /// to determine when the DB parameter group is ready to use.
   /// </summary>
   /// <param name="name">Name of the DB parameter group.</param>
   /// <param name="family">Family of the DB parameter group.</param>
   /// <param name="description">Description of the DB parameter group.</param>
   /// <returns>The new DB parameter group.</returns>
   public async Task<DBParameterGroup> CreateDBParameterGroup(
       string name, string family, string description)
   {
       var response = await _amazonRDS.CreateDBParameterGroupAsync(
           new CreateDBParameterGroupRequest()
           {
               DBParameterGroupName = name,
               DBParameterGroupFamily = family,
               Description = description
           });
       return response.DBParameterGroup;
   }
```

- Para obter detalhes da API, consulte CreateDBParameterGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example parameter group.");

    Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
            client.CreateDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully created."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
```

- Para obter detalhes da API, consulte CreateDBParameterGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como criar um grupo de parâmetros de banco de dados

O exemplo de `create-db-parameter-group` a seguir cria um grupo de parâmetros de banco de dados.

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family MySQL5.6 \
    --description "My new parameter group"
```

Saída:

```
{
    "DBParameterGroup": {
        "DBParameterGroupName": "mydbparametergroup",
        "DBParameterGroupFamily": "mysql5.6",
        "Description": "My new parameter group",
        "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:mydbparametergroup"
    }
}
```

Para obter mais informações, consulte Criar um grupo de parâmetros de banco de dados no Guia do usuário do Amazon RDS.

- Para obter detalhes da API, consulte CreateDBParameterGroupem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateParameterGroup creates a DB parameter group that is based on the
 specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}
```

- Para obter detalhes da API, consulte CreateDBParameterGroupa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
       try {
           CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CreateDBParameterGroupa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter
 group
        family.

        :param parameter_group_name: The name of the newly created parameter
 group.
        :param parameter_group_family: The family that is used as the basis of
 the new
                                       parameter group.
        :param description: A description given to the parameter group.
```

```
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateDBParameterGroupReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_dbparametergroupname   = 'mydbparametergroup'
    " iv_dbparametergroupfamily = 'mysql8.0'
    " iv_description            = 'My custom DB parameter group for MySQL 8.0'
    TRY.
        oo_result = lo_rds->createdbparametergroup(
          iv_dbparametergroupname   = iv_dbparametergroupname
          iv_dbparametergroupfamily = iv_dbparametergroupfamily
```

```
          iv_description            = iv_description ).
        MESSAGE 'DB parameter group created.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbparmgralrexfault.
        MESSAGE 'DB parameter group already exists.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
        MESSAGE 'DB parameter group quota exceeded.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDBParameterGroupreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
 RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
```

```
        } catch {
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte CreateDBParameterGroupa referência da API AWS SDK for Swift.

Use **`createDBSnapshot`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateDBSnapshot`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
```

```
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
```

```
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Como criar um snapshot de banco de dados

O exemplo de `create-db-snapshot` a seguir cria um snapshot de banco de dados.

```
aws rds create-db-snapshot \
    --db-instance-identifier database-mysql \
    --db-snapshot-identifier mydbsnapshot
```

Saída:

```
{
    "DBSnapshot": {
        "DBSnapshotIdentifier": "mydbsnapshot",
        "DBInstanceIdentifier": "database-mysql",
        "Engine": "mysql",
        "AllocatedStorage": 100,
        "Status": "creating",
        "Port": 3306,
        "AvailabilityZone": "us-east-1b",
        "VpcId": "vpc-6594f31c",
        "InstanceCreateTime": "2019-04-30T15:45:53.663Z",
        "MasterUsername": "admin",
        "EngineVersion": "5.6.40",
        "LicenseModel": "general-public-license",
        "SnapshotType": "manual",
```

```
        "Iops": 1000,
        "OptionGroupName": "default:mysql-5-6",
        "PercentProgress": 0,
        "StorageType": "io1",
        "Encrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
AKIAIOSFODNN7EXAMPLE",
        "DBSnapshotArn": "arn:aws:rds:us-
east-1:123456789012:snapshot:mydbsnapshot",
        "IAMDatabaseAuthenticationEnabled": false,
        "ProcessorFeatures": [],
        "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"
    }
}
```

Para obter mais informações, consulte Criar um snapshot de banco de dados no Guia do usuário do Amazon RDS.

- Consulte detalhes da API em CreateDBSnapshot na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
```

```
 RdsClient *rds.Client
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Create an Amazon RDS snapshot.
   public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
       try {
           CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
 rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
 response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK for Java 2.x .

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
```

```
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id,
DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
                                db_snapshot_identifier: id
                              })
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para Ruby .

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Create a snapshot of the specified name.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database instance to
    ///     snapshot.
    ///   - snapshotIdentifier: A unique identifier to give the newly-created
    ///     snapshot.
    func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
 async {
        do {
            let output = try await rdsClient.createDBSnapshot(
                input: CreateDBSnapshotInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    dbSnapshotIdentifier: snapshotIdentifier
                )
            )

            guard let snapshot = output.dbSnapshot else {
                print("No snapshot returned.")
                return
            }

            print("The snapshot has been created with ID
 \(snapshot.dbiResourceId ?? "<unknown>")")
        } catch {
            print("*** Unable to create the database snapshot named
 \(snapshotIdentifier): \(error.localizedDescription)")
        }
    }
```

- Consulte detalhes da API em CreateDBSnapshot na Referência de API do AWS SDK para Swift.

Use **`DeletedBInstance com`** um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteDBInstance`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete a particular DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <returns>DB instance object.</returns>
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
{
    var response = await _amazonRDS.DeleteDBInstanceAsync(
        new DeleteDBInstanceRequest()
        {
            DBInstanceIdentifier = dbInstanceIdentifier,
            SkipFinalSnapshot = true,
            DeleteAutomatedBackups = true
        });

    return response.DBInstance;
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBInstanceRequest request;
        request.SetDBInstanceIdentifier(dbInstanceIdentifier);
        request.SetSkipFinalSnapshot(true);
        request.SetDeleteAutomatedBackups(true);

        Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                client.DeleteDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB instance deletion has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Como excluir uma instância de banco de dados

O exemplo de `delete-db-instance` a seguir exclui a instância de banco de dados especificada depois de criar um snapshot de banco de dados final chamado `test-instance-final-snap`.

```
aws rds delete-db-instance \
    --db-instance-identifier test-instance \
    --final-db-snapshot-identifier test-instance-final-snap
```

Saída:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-instance",
        "DBInstanceStatus": "deleting",
        ...some output truncated...
    }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
```

```
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    echo $e->getMessage();
    echo "\n";
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.
```

```
        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
```

```
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
        } catch {
            print("*** Error deleting the database instance
\(instanceIdentifier): \(error.localizedDescription)")
        }
    }
```

- Consulte detalhes da API em DeleteDBInstance na Referência de API do AWS SDK para Swift.

Use **`DeleteDBParameterGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteDBParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Delete a DB parameter group. The group cannot be a default DB parameter
group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteDBParameterGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
```

```
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Para obter detalhes da API, consulte DeleteDBParameterGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir um grupo de parâmetros de banco de dados

O exemplo de `command` a seguir exclui um grupo de parâmetros de banco de dados.

```
aws rds delete-db-parameter-group \
    --db-parameter-group-name mydbparametergroup
```

Este comando não produz saída.

Para obter mais informações, consulte Como trabalhar com grupos de parâmetros de banco de dados no Guia do usuário do Amazon RDS.

- Para obter detalhes da API, consulte DeleteDBParameterGroupem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obter detalhes da API, consulte DeleteDBParameterGroupa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Delete the parameter group after database has been deleted.
   // An exception is thrown if you attempt to delete the para group while
database
   // exists.
   public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(dbARN) == 0) {
                       System.out.println(dbARN + " still exists");
                       didFind = true;
                   }
```

```
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
 DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteDBParameterGroupa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteDBParameterGroupReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    TRY.
        lo_rds->deletedbparametergroup(
          iv_dbparametergroupname = iv_dbparametergroupname ).
        MESSAGE 'DB parameter group deleted.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDBParameterGroupreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
```

```
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

- Para obter detalhes da API, consulte DeleteDBParameterGroupa referência da API AWS SDK for Swift.

Use **`DescribeAccountAttributes`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAccountAttributes`.

CLI

AWS CLI

Como descrever os atributos da conta

O `describe-account-attributes` exemplo a seguir recupera os atributos da AWS conta atual.

```
aws rds describe-account-attributes
```

Saída:

```
{
    "AccountQuotas": [
        {
            "Max": 40,
            "Used": 4,
            "AccountQuotaName": "DBInstances"
        },
        {
```

```
            "Max": 40,
            "Used": 0,
            "AccountQuotaName": "ReservedDBInstances"
        },
        {
            "Max": 100000,
            "Used": 40,
            "AccountQuotaName": "AllocatedStorage"
        },
        {
            "Max": 25,
            "Used": 0,
            "AccountQuotaName": "DBSecurityGroups"
        },
        {
            "Max": 20,
            "Used": 0,
            "AccountQuotaName": "AuthorizationsPerDBSecurityGroup"
        },
        {
            "Max": 50,
            "Used": 1,
            "AccountQuotaName": "DBParameterGroups"
        },
        {
            "Max": 100,
            "Used": 3,
            "AccountQuotaName": "ManualSnapshots"
        },
        {
            "Max": 20,
            "Used": 0,
            "AccountQuotaName": "EventSubscriptions"
        },
        {
            "Max": 50,
            "Used": 1,
            "AccountQuotaName": "DBSubnetGroups"
        },
        {
            "Max": 20,
            "Used": 1,
            "AccountQuotaName": "OptionGroups"
        },
```

```
        {
            "Max": 20,
            "Used": 6,
            "AccountQuotaName": "SubnetsPerDBSubnetGroup"
        },
        {
            "Max": 5,
            "Used": 0,
            "AccountQuotaName": "ReadReplicasPerMaster"
        },
        {
            "Max": 40,
            "Used": 1,
            "AccountQuotaName": "DBClusters"
        },
        {
            "Max": 50,
            "Used": 0,
            "AccountQuotaName": "DBClusterParameterGroups"
        },
        {
            "Max": 5,
            "Used": 0,
            "AccountQuotaName": "DBClusterRoles"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeAccountAttributesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import
 software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
 rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte DescribeAccountAttributesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAccountAttributes() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAccountAttributesreferência da API AWS SDK for Kotlin.

Use **`DescribeDBEngineVersions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeDBEngineVersions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.
```

```
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como descrever as versões do mecanismo de banco de dados do mecanismo de banco de dados do MySQL

O exemplo de `describe-db-engine-versions` a seguir exibe os detalhes de cada uma das versões do mecanismo de banco de dados para o mecanismo de banco de dados especificado.

```
aws rds describe-db-engine-versions \
    --engine mysql
```

Saída:

```
{
    "DBEngineVersions": [
        {
            "Engine": "mysql",
            "EngineVersion": "5.5.46",
            "DBParameterGroupFamily": "mysql5.5",
            "DBEngineDescription": "MySQL Community Edition",
            "DBEngineVersionDescription": "MySQL 5.5.46",
            "ValidUpgradeTarget": [
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.53",
                    "Description": "MySQL 5.5.53",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.54",
                    "Description": "MySQL 5.5.54",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.57",
                    "Description": "MySQL 5.5.57",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                ...some output truncated...
            ]
        }
```

Para obter mais informações sobre o Amazon RDS, consulte O que é o Amazon Relational Database Service (Amazon RDS)? no Guia do usuário do Amazon RDS.

- Para obter detalhes da API, consulte DescribeDBEngineVersionsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine
 string, parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
```

```
    DBParameterGroupFamily: aws.String(parameterGroupFamily),
   })
  if err != nil {
   log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
   return nil, err
  } else {
   return output.DBEngineVersions, nil
  }
 }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
```

```
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
```

```
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersionsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " iv_engine                 = 'mysql'
   " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter
group family)
   TRY.
       oo_result = lo_rds->describedbengineversions(
         iv_engine                 = iv_engine
         iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
       DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
       MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersionsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
```

```
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
[RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
        } catch {
            return []
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersionsa referência da API AWS SDK for Swift.

Use **`DescribedBInstances`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeDBInstances`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para C++ .

CLI

AWS CLI

Como descrever uma instância de banco de dados

O exemplo de `describe-db-instances` a seguir recupera detalhes sobre a instância de banco de dados especificada.

```
aws rds describe-db-instances \
    --db-instance-identifier mydbinstancecf
```

Saída:

```
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "mydbinstancecf",
            "DBInstanceClass": "db.t3.small",
            "Engine": "mysql",
            "DBInstanceStatus": "available",
            "MasterUsername": "masterawsuser",
            "Endpoint": {
                "Address": "mydbinstancecf.abcexample.us-
east-1.rds.amazonaws.com",
                "Port": 3306,
                "HostedZoneId": "Z2R2ITUGPM61AM"
            },
            ...some output truncated...
        }
    ]
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
```

```
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
```

```
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " +
 instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeInstances() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest
 {})
        response.dbInstances?.forEach { instance ->
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
```

```
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
```

```
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
                          "name": i.id,
                          "status": i.db_instance_status
                        })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para Ruby .

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Wait until the specified database is available to use.
    ///
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
 RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
 may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
```

```
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para Swift.

Use **`DescribeDBParameterGroups`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeDBParameterGroups`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string
 name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }
```

- Para obter detalhes da API, consulte DescribeDBParameterGroupsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
```

```
        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte DescribeDBParameterGroupsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como descrever um grupo de parâmetros de banco de dados

O exemplo de `describe-db-parameter-groups` a seguir recupera detalhes sobre os grupos de parâmetros de banco de dados.

```
aws rds describe-db-parameter-groups
```

Saída:

```
{
    "DBParameterGroups": [
        {
            "DBParameterGroupName": "default.aurora-mysql5.7",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "Default parameter group for aurora-mysql5.7",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-mysql5.7"
```

```
        },
        {
            "DBParameterGroupName": "default.aurora-postgresql9.6",
            "DBParameterGroupFamily": "aurora-postgresql9.6",
            "Description": "Default parameter group for aurora-postgresql9.6",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-postgresql9.6"
        },
        {
            "DBParameterGroupName": "default.aurora5.6",
            "DBParameterGroupFamily": "aurora5.6",
            "Description": "Default parameter group for aurora5.6",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora5.6"
        },
        {
            "DBParameterGroupName": "default.mariadb10.1",
            "DBParameterGroupFamily": "mariadb10.1",
            "Description": "Default parameter group for mariadb10.1",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.mariadb10.1"
        },
        ...some output truncated...
    ]
}
```

Para obter mais informações, consulte Como trabalhar com grupos de parâmetros de banco de dados no Guia do usuário do Amazon RDS.

- Para obter detalhes da API, consulte DescribeDBParameterGroupsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}
```

- Para obter detalhes da API, consulte DescribeDBParameterGroupsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
       try {
           DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .maxRecords(20)
                   .build();

           DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
           List<DBParameterGroup> groups = response.dbParameterGroups();
           for (DBParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbParameterGroupName());
               System.out.println("The group description is " +
group.description());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeDBParameterGroupsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
```

```
                logger.info("Parameter group %s does not exist.",
 parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- Para obter detalhes da API, consulte a DescribeDBParameterGroupsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
```

```
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Para obter detalhes da API, consulte DescribeDBParameterGroupsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    TRY.
        oo_result = lo_rds->describedbparametergroups(
          iv_dbparametergroupname = iv_dbparametergroupname ).
        MESSAGE 'DB parameter group retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBParameterGroupsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
    func describeDBParameterGroups(groupName: String) async ->
 [RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBParameterGroupsa referência da API AWS SDK for Swift.

Use **`DescribedBParameters`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeDBParameters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter
group.</param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
```

```
        return results;
    }
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
```

```
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Como descrever os parâmetros de um grupo de parâmetros de banco de dados

O exemplo de `describe-db-parameters` a seguir recupera detalhes sobre o grupo de parâmetros de banco de dados especificado.

```
aws rds describe-db-parameters \
    --db-parameter-group-name mydbpg
```

Saída:

```
{
    "Parameters": [
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have
 only an xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "auto_generate_certs",
            "Description": "Controls whether the server autogenerates SSL key and
 certificate files in the data directory, if they do not already exist.",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot"
        },
        ...some output truncated...
```

```
    ]
}
```

Para obter mais informações, consulte Como trabalhar com grupos de parâmetros de banco de dados no Guia do usuário do Amazon RDS.

- Consulte detalhes da API em DescribeDBParameters na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context,
 parameterGroupName string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
```

```
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Retrieve parameters in the group.
   public static void describeDbParameters(RdsClient rdsClient, String
dbGroupName, int flag) {
       try {
           DescribeDbParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
```

```
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
 filtered
                            to contain only parameters that start with this
 prefix.
        :param source: When specified, only parameters from this source are
 retrieved.
                       For example, a source of 'user' retrieves only parameters
 that
                       were set by a user.
```

```
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'
```

```
# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para Ruby .

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    " iv_source               = 'user' (optional - filters by parameter source)
    TRY.
        oo_result = lo_rds->describedbparameters(
          iv_dbparametergroupname = iv_dbparametergroupname
          iv_source               = iv_source ).
        DATA(lv_param_count) = lines( oo_result->get_parameters( ) ).
        MESSAGE |Retrieved { lv_param_count } parameters.| TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
```

```
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
    ENDTRY.
```

- Para ver detalhes da API, consulte DescribeDBParameters na Referência de API do SDK da AWS para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
 [RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )
```

```
            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
\(error.localizedDescription)")
            return []
        }

        return parameterList
    }
```

- Consulte detalhes da API em DescribeDBParameter na Referência de API do AWS SDK para Swift.

Use **`DescribedBSnapshots`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeDBSnapshots`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
        return results;
    }
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Exemplo 1: como descrever um snapshot de banco de dados de uma instância de banco de dados

O exemplo de `describe-db-snapshots` a seguir recupera os detalhes de um snapshot de banco de dados para uma instância de banco de dados.

```
aws rds describe-db-snapshots \
    --db-snapshot-identifier mydbsnapshot
```

Saída:

```
{
    "DBSnapshots": [
```

```
        {
            "DBSnapshotIdentifier": "mydbsnapshot",
            "DBInstanceIdentifier": "mysqldb",
            "SnapshotCreateTime": "2018-02-08T22:28:08.598Z",
            "Engine": "mysql",
            "AllocatedStorage": 20,
            "Status": "available",
            "Port": 3306,
            "AvailabilityZone": "us-east-1f",
            "VpcId": "vpc-6594f31c",
            "InstanceCreateTime": "2018-02-08T22:24:55.973Z",
            "MasterUsername": "mysqladmin",
            "EngineVersion": "5.6.37",
            "LicenseModel": "general-public-license",
            "SnapshotType": "manual",
            "OptionGroupName": "default:mysql-5-6",
            "PercentProgress": 100,
            "StorageType": "gp2",
            "Encrypted": false,
            "DBSnapshotArn": "arn:aws:rds:us-
east-1:123456789012:snapshot:mydbsnapshot",
            "IAMDatabaseAuthenticationEnabled": false,
            "ProcessorFeatures": [],
            "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"
        }
    ]
}
```

Para obter mais informações, consulte Criar um snapshot de banco de dados no Guia do usuário do Amazon RDS.

Exemplo 2: como encontrar o número de snapshots manuais criados

O `describe-db-snapshots` exemplo a seguir usa o `length` operador na `--query` opção para retornar o número de instantâneos manuais que foram tirados em uma AWS região específica.

```
aws rds describe-db-snapshots \
    --snapshot-type manual \
    --query "length(*[].{DBSnapshots:SnapshotType})" \
    --region eu-central-1
```

Saída:

```
35
```

Para obter mais informações, consulte Criar um snapshot de banco de dados no Guia do usuário do Amazon RDS.

- Consulte detalhes da API em DescribeDBSnapshots na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName
 string) (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
```

```
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência da API AWS SDK para Go .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Consulte detalhes da API em DescribeDBSnapshots na referência da API do AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'
```

```
# List all Amazon Relational Database Service (Amazon RDS) DB instance
# snapshots.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return instance_snapshots [Array, nil] All instance snapshots, or nil if
 error.
def list_instance_snapshots(rds_resource)
  instance_snapshots = []
  rds_resource.db_snapshots.each do |s|
    instance_snapshots.append({
                                "id": s.snapshot_id,
                                "status": s.status
                              })
  end
  instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
end
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência da API AWS SDK para Ruby .

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Wait until the specified database snapshot is available to use.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database for which the
    ///     snapshot was taken.
```

```
   ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
   func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
String) async {
       var snapshotReady = false

       putString("Waiting for the snapshot to be ready...")

       do {
           while !snapshotReady {
               let output = try await rdsClient.describeDBSnapshots(
                   input: DescribeDBSnapshotsInput(
                       dbInstanceIdentifier: instanceIdentifier,
                       dbSnapshotIdentifier: snapshotIdentifier
                   )
               )

               guard let snapshotList = output.dbSnapshots else {
                   return
               }

               for snapshot in snapshotList {
                   guard let snapshotReadyStr = snapshot.status else {
                       return
                   }

                   if snapshotReadyStr.contains("available") {
                       snapshotReady = true
                       print()
                   } else {
                       putString(".")
                       do {
                           try await Task.sleep(for: .seconds(15))
                       } catch {
                           print("\n*** Error pausing the task!")
                       }
                   }
               }
           }
       } catch {
           print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
       }
   }
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência de API do AWS SDK para Swift.

Use **`DescribeOrderableDBInstanceOptions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeOrderableDBInstanceOptions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
```

```
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
 paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
```

```
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
```

```
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database
 engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como descrever as opções de instância solicitável do banco de dados

O exemplo de `describe-orderable-db-instance-options` a seguir recupera detalhes sobre as opções que podem ser solicitadas em uma instância de banco de dados que executa o mecanismo de banco de dados do MySQL.

```
aws rds describe-orderable-db-instance-options \
    --engine mysql
```

Saída:

```
{
    "OrderableDBInstanceOptions": [
        {
            "MinStorageSize": 5,
            "ReadReplicaCapable": true,
```

```
            "MaxStorageSize": 6144,
            "AvailabilityZones": [
                {
                    "Name": "us-east-1a"
                },
                {
                    "Name": "us-east-1b"
                },
                {
                    "Name": "us-east-1c"
                },
                {
                    "Name": "us-east-1d"
                }
            ],
            "SupportsIops": false,
            "AvailableProcessorFeatures": [],
            "MultiAZCapable": true,
            "DBInstanceClass": "db.m1.large",
            "Vpc": true,
            "StorageType": "gp2",
            "LicenseModel": "general-public-license",
            "EngineVersion": "5.5.46",
            "SupportsStorageEncryption": false,
            "SupportsEnhancedMonitoring": true,
            "Engine": "mysql",
            "SupportsIAMDatabaseAuthentication": false,
            "SupportsPerformanceInsights": false
        }
    ]
    ...some output truncated...
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
```

```
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions,
 output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsa Referência AWS SDK para Go da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
```

```
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptionsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_engine        = 'mysql'
    " iv_engineversion = '8.0.35'
    TRY.
        oo_result = lo_rds->descrorderabledbinstoptions(
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result-
>get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.|
 TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptionsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
```

```
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
    ///   - engineVersion: The database version for which to list instances.
    ///
    /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
    ///   the selected instance type.
    func chooseMicroInstance(engine: String = "mysql", engineVersion: String? =
nil) async -> RDSClientTypes.OrderableDBInstanceOption? {
        do {
            let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
                input: DescribeOrderableDBInstanceOptionsInput(
                    engine: engine,
                    engineVersion: engineVersion
                )
            )

            var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []

            for try await page in pages {
                guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                    continue
                }

                for dbInstanceOption in orderableDBInstanceOptions {
                    guard let className = dbInstanceOption.dbInstanceClass else {
                        continue
                    }
                    if className.contains("micro") {
                        optionsList.append(dbInstanceOption)
                    }
                }
            }

            print("Found \(optionsList.count) database instances of 'micro' class
types:")
            for dbInstanceOption in optionsList {
                print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
            }
```

```
            return optionsList[0]
        } catch {
            print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptionsa referência da API AWS SDK for Swift.

Use **`GenerateRDSAuthToken`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GenerateRDSAuthToken`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a RdsUtilitiesclasse para gerar um token de autenticação.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <masterUsername>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUsername - The master user name.\s
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
 masterUsername);
        System.out.println("The token response is " + token);
    }

    public static String getAuthToken(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
 GenerateAuthenticationTokenRequest.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .username(masterUsername)
                    .port(3306)
                    .hostname(dbInstanceIdentifier)
                    .build();

            return utilities.generateAuthenticationToken(tokenRequest);

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte GenerateRDSAuthTokena Referência AWS SDK for Java 2.x da API.

Use **`ModifyDBInstance`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ModifyDBInstance`.

CLI

AWS CLI

Exemplo 1: Como modificar uma instância de banco de dados

O exemplo de `modify-db-instance` a seguir associa um grupo de opção e um grupo de parâmetros a uma instância de banco de dados do Microsoft SQL Server compatível. O parâmetro `--apply-immediately` faz com que os grupos de opção e parâmetro sejam associados imediatamente em vez de ter que esperar a próxima janela de manutenção.

```
aws rds modify-db-instance \
    --db-instance-identifier database-2 \
    --option-group-name test-se-2017 \
    --db-parameter-group-name test-sqlserver-se-2017 \
    --apply-immediately
```

Saída:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "database-2",
        "DBInstanceClass": "db.r4.large",
        "Engine": "sqlserver-se",
        "DBInstanceStatus": "available",

        ...output omitted...

        "DBParameterGroups": [
            {
                "DBParameterGroupName": "test-sqlserver-se-2017",
                "ParameterApplyStatus": "applying"
            }
        ],
        "AvailabilityZone": "us-west-2d",

        ...output omitted...

        "MultiAZ": true,
```

```
        "EngineVersion": "14.00.3281.6.v1",
        "AutoMinorVersionUpgrade": false,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "license-included",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "test-se-2017",
                "Status": "pending-apply"
            }
        ],
        "CharacterSetName": "SQL_Latin1_General_CP1_CI_AS",
        "SecondaryAvailabilityZone": "us-west-2c",
        "PubliclyAccessible": true,
        "StorageType": "gp2",

        ...output omitted...

        "DeletionProtection": false,
        "AssociatedRoles": [],
        "MaxAllocatedStorage": 1000
    }
}
```

Para obter mais informações, consulte Modificar uma instância de banco de dados do Amazon RDS no Guia do usuário do Amazon RDS.

Exemplo 2: Como associar um grupo de segurança da VPC a uma instância de banco de dados

O seguinte exemplo de `modify-db-instance` associa um grupo de segurança da VPC específico e remove grupos de segurança de banco de dados da instância de banco de dados:

```
aws rds modify-db-instance \
    --db-instance-identifier dbName \
    --vpc-security-group-ids sg-ID
```

Saída:

```
{
"DBInstance": {
    "DBInstanceIdentifier": "dbName",
```

```
    "DBInstanceClass": "db.t3.micro",
    "Engine": "mysql",
    "DBInstanceStatus": "available",
    "MasterUsername": "admin",
    "Endpoint": {
        "Address": "dbName.abcdefghijk.us-west-2.rds.amazonaws.com",
        "Port": 3306,
        "HostedZoneId": "ABCDEFGHIJK1234"
    },
    "AllocatedStorage": 20,
    "InstanceCreateTime": "2024-02-15T00:37:58.793000+00:00",
    "PreferredBackupWindow": "11:57-12:27",
    "BackupRetentionPeriod": 7,
    "DBSecurityGroups": [],
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-ID",
            "Status": "active"
        }
    ],
    ... output omitted ...
    "MultiAZ": false,
    "EngineVersion": "8.0.35",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",

    ... output omitted ...
    }
}
```

Para ter mais informações, consulte Controlar acesso com grupos de segurança no Guia do usuário do Amazon RDS.

- Consulte detalhes da API em ModifyDBInstance na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUserPassword - The updated password that corresponds to
 the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master
 password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
 ModifyDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .publiclyAccessible(true)
                    .masterUserPassword(masterUserPassword)
                    .build();

            ModifyDbInstanceResponse instanceResponse =
 rdsClient.modifyDBInstance(modifyDbInstanceRequest);
            System.out.print("The ARN of the modified database is: " +
 instanceResponse.dbInstance().dbInstanceArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em ModifyDBInstance na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- Consulte detalhes da API em ModifyDBInstance na Referência da API AWS SDK para Kotlin.

Use **`ModifyDBParameterGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ModifyDBParameterGroup`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update a DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }
```

- Para obter detalhes da API, consulte ModifyDBParameterGroupa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::ModifyDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetParameters(updateParameters);

    Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
            client.ModifyDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully modified."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Para obter detalhes da API, consulte ModifyDBParameterGroupa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como modificar um grupo de parâmetros de banco de dados

O exemplo de `modify-db-parameter-group` a seguir altera o valor do parâmetro `clr enabled` em um grupo de parâmetros de banco de dados. O parâmetro `--apply-immediately` faz com que o grupo de parâmetros de banco de dados seja modificado imediatamente em vez de ter que esperar a próxima janela de manutenção.

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name test-sqlserver-se-2017 \
    --parameters "ParameterName='clr
 enabled',ParameterValue=1,ApplyMethod=immediate"
```

Saída:

```
{
    "DBParameterGroupName": "test-sqlserver-se-2017"
}
```

Para obter mais informações, consulte Modificar parâmetros em um grupo de parâmetros de banco de dados no Guia do usuário do Amazon RDS.

- Para obter detalhes da API, consulte ModifyDBParameterGroupem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obter detalhes da API, consulte ModifyDBParameterGroupa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ModifyDBParameterGroupa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
 Parameters=update_parameters
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a ModifyDBParameterGroupReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" iv_dbparametergroupname = 'mydbparametergroup'
" it_parameters - table containing parameter objects with:
"   - parametername = 'max_connections'
"   - parametervalue = '100'
"   - applymethod = 'immediate' or 'pending-reboot'
TRY.
    oo_result = lo_rds->modifydbparametergroup(
      iv_dbparametergroupname = iv_dbparametergroupname
      it_parameters           = it_parameters ).
    MESSAGE 'DB parameter group modified.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    MESSAGE 'DB parameter group not found.' TYPE 'I'.
  CATCH /aws1/cx_rdsinvdbprmgrstatef00.
    MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ModifyDBParameterGroupreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )
```

```
            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
 \(error.localizedDescription)")
        }
    }
```

- Para obter detalhes da API, consulte ModifyDBParameterGroupa referência da API AWS SDK for Swift.

Use **`RebootDBInstance`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RebootDBInstance`.

CLI

AWS CLI

Como reinicializar uma instância de banco de dados

O exemplo de `reboot-db-instance` a seguir inicia uma reinicialização da instância de banco de dados especificada.

```
aws rds reboot-db-instance \
    --db-instance-identifier test-mysql-instance
```

Saída:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-mysql-instance",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "rebooting",
        "MasterUsername": "admin",
        "Endpoint": {
            "Address": "test-mysql-instance.############.us-
west-2.rds.amazonaws.com",
            "Port": 3306,
            "HostedZoneId": "Z1PVIF0EXAMPLE"
```

```
        },

    ... output omitted...

    }
}
```

Para obter mais informações, consulte Reinicializar uma instância de banco de dados no Guia do usuário do Amazon RDS.

- Consulte detalhes da API em RebootDBInstance na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
 RebootDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            RebootDbInstanceResponse instanceResponse =
 rdsClient.rebootDBInstance(rebootDbInstanceRequest);
            System.out.print("The database " +
 instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em RebootDBInstance na Referência da API AWS SDK for Java 2.x .

Cenários para o Amazon RDS usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon RDS com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon RDS ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie um rastreador de itens de trabalho do Aurora Sem Servidor
- Criação de uma instância de banco de dados relacional

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Os exemplos de código a seguir mostram como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

.NET

SDK para .NET

Mostra como usar o AWS SDK para .NET para criar um aplicativo web que rastreia itens de trabalho em um banco de dados Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful .NET.

- Integre um aplicativo web React com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

C++

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar uma API REST C++ que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Java

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar um aplicativo web que rastreia itens de trabalho em um banco de dados Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um Node.js back-end Express.

- Integre um aplicativo React.js web com Serviços da AWS o.
- Liste, adicione e atualize itens em uma tabela do Aurora.
- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

PHP

SDK para PHP

Mostra como usar o AWS SDK para PHP para criar uma aplicação web que rastreia itens de trabalho em um banco de dados do Amazon RDS e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end PHP RESTful.

- Integre um aplicativo React.js web com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS

- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um AWS Secrets Manager segredo que contenha as credenciais do banco de dados e use-o para autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Criação de uma instância de banco de dados relacional

O exemplo de código a seguir mostra como:

- Configurar componentes de rede.
- Criar um grupo de sub-redes de banco de dados
- Criar uma instância de banco de dados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/
latest/UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
```

```
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier
 "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ];
 then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ];
 then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
```

```
if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found
 ${#SUBNET_IDS[@]} subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP
 ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
```

```
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
```

```
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id
 "$SECRET_NAME" --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -
d'"' -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -
d'"' -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi
```

```
echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."
```

```
    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue

- Wait

Exemplos de tecnologia sem servidor para o Amazon RDS

Os exemplos de código a seguir mostram como usar o Amazon RDS com AWS SDKs.

Exemplos

- Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

Os exemplos de código a seguir mostram como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando .NET.

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace aws_rds;
```

```
public class InputModel
{
    public string key1 { get; set; }
    public string key2 { get; set; }
}

public class Function
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
 FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
 $"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

 $"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +

 $"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                 $"Pwd={authToken};";



        try
        {
            await using var connection = new MySqlConnection(connectionString);
```

```
            await connection.OpenAsync();

            const string sql = "SELECT @param1 + @param2 AS Sum";

            await using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
 "0"));
            command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
 "0"));

            await using var reader = await command.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                int result = reader.GetInt32("Sum");

                //Sample Response: {"statusCode":200,"body":"{\"message\":\"The
 sum is: 45\"}","isBase64Encoded":false}
                return new APIGatewayProxyResponse
                {
                    StatusCode = 200,
                    Body = JsonSerializer.Serialize(new { message = $"The sum is:
 {result}" })
                };
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = 500,
            Body = JsonSerializer.Serialize(new { error = "Internal server
 error" })
        };
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Go.

```
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"

 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
 var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
 var region string = os.Getenv("AWS_REGION")
```

```
 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error: " + err.Error())
 }

 authenticationToken, err := auth.BuildAuthToken(
  context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
 if err != nil {
  panic("failed to create authentication token: " + err.Error())
 }

 dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
  dbUser, authenticationToken, dbEndpoint, dbName,
 )

 db, err := sql.Open("mysql", dsn)
 if err != nil {
  panic(err)
 }

 defer db.Close()

 var sum int
 err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
 if err != nil {
  panic(err)
 }
 s := fmt.Sprint(sum)
 message := fmt.Sprintf("The selected sum is: %s", s)

 messageBytes, err := json.Marshal(message)
 if err != nil {
  return nil, err
 }

 messageString := string(messageBytes)
 return map[string]interface{}{
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}
```

```
func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements
 RequestHandler<APIGatewayProxyRequestEvent, APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new
 APIGatewayProxyResponseEvent();
```

```
        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"),
 token);
                 PreparedStatement statement =
 connection.prepareStatement("SELECT ? + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();
```

```
        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response =
 rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectando-se a um banco de dados do Amazon RDS em uma função Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';
```

```
async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}

export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
```

```
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

Conectando-se a um banco de dados do Amazon RDS em uma função Lambda usando. TypeScript

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that
 the DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}

async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
```

```
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number;
 body: string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
    };
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];
```

```
        // Create RDS AuthTokenGenerator object
        $generator = new
AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }
```

```
    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Python.

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
```

```
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
```

```
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Rust.

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
```

```
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
        port = port,
        db_user = db_username
    );

    let signable_request =
```

```
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
```

```
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

Exemplos de código para o Amazon RDS Data Service usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Relational Database Service (Amazon Relational Database Service) com AWS um kit de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon RDS Data Service: mais informações sobre o Amazon RDS Data Service.
- Referência da API Amazon RDS Data Service: detalhes sobre todas as ações disponíveis do Amazon RDS Data Service.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.

- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon RDS Data Service usando AWS SDKs
  - Ações para o Amazon RDS Data Service usando AWS SDKs
    - Use ExecuteStatementcom um AWS SDK ou CLI
- Cenários para o Amazon RDS Data Service usando AWS SDKs
  - Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Exemplos básicos para o Amazon RDS Data Service usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Relational Database Service Data Service com AWS SDKs.

Exemplos

- Ações para o Amazon RDS Data Service usando AWS SDKs
  - Use ExecuteStatementcom um AWS SDK ou CLI

Ações para o Amazon RDS Data Service usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon RDS Data Service com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon RDS Data Service e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon RDS Data Service usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Relational Database Service Data Service.

Exemplos

- Use ExecuteStatementcom um AWS SDK ou CLI

Use **`ExecuteStatement`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ExecuteStatement`.

CLI

AWS CLI

Exemplo 1: como executar uma instrução SQL que faz parte de uma transação

O exemplo de `execute-statement` a seguir executa uma instrução SQL que é parte de uma transação.

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:us-west-2:123456789012:cluster:mydbcluster" \
    --database "mydb" \
    --secret-arn "arn:aws:secretsmanager:us-west-2:123456789012:secret:mysecret"
 \
    --sql "update mytable set quantity=5 where id=201" \
    --transaction-id "ABC1234567890xyz"
```

Saída:

```
{
    "numberOfRecordsUpdated": 1
}
```

Exemplo 2: como executar uma instrução SQL com parâmetros

O exemplo de `execute-statement` a seguir executa uma instrução SQL com parâmetros.

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster" \
    --database "mydb" \
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret"
 \
    --sql "insert into mytable values (:id, :val)" \
    --parameters "[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"value1\"}}]"
```

Saída:

```
{
    "numberOfRecordsUpdated": 1
}
```

Para obter mais informações, consulte Using the Data API for Aurora Serverless no Guia do usuário do Amazon RDS.

- Para obter detalhes da API, consulte ExecuteStatementem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn query_cluster(
    client: &Client,
    cluster_arn: &str,
    query: &str,
    secret_arn: &str,
) -> Result<(), Error> {
    let st = client
        .execute_statement()
        .resource_arn(cluster_arn)
        .database("postgres") // Do not confuse this with db instance name
        .sql(query)
        .secret_arn(secret_arn);

    let result = st.send().await?;

    println!("{:?}", result);
    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a ExecuteStatementreferência da API AWS SDK for Rust.

Cenários para o Amazon RDS Data Service usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon RDS Data Service com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon RDS Data Service ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Os exemplos de código a seguir mostram como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

.NET

SDK para .NET

Mostra como usar o AWS SDK para .NET para criar um aplicativo web que rastreia itens de trabalho em um banco de dados Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful .NET.

- Integre um aplicativo web React com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

C++

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar uma API REST C++ que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Java

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar um aplicativo web que rastreia itens de trabalho em um banco de dados Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um Node.js back-end Express.

- Integre um aplicativo React.js web com Serviços da AWS o.
- Liste, adicione e atualize itens em uma tabela do Aurora.
- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

PHP

SDK para PHP

Mostra como usar o AWS SDK para PHP para criar uma aplicação web que rastreia itens de trabalho em um banco de dados do Amazon RDS e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end PHP RESTful.

- Integre um aplicativo React.js web com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um AWS Secrets Manager segredo que contenha as credenciais do banco de dados e use-o para autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos de código para o Amazon Redshift usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Redshift com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia de gerenciamento do Amazon Redshift: mais informações sobre o Amazon Redshift.
- Referência da API Amazon Redshift: detalhes sobre todas as ações disponíveis do Amazon Redshift.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon Redshift usando AWS SDKs
  - Olá, Amazon Redshift
  - Aprenda as noções básicas do Amazon Redshift com um AWS SDK
  - Ações para o Amazon Redshift usando AWS SDKs
    - Use CreateClustercom um AWS SDK ou CLI
    - Use DeleteClustercom um AWS SDK ou CLI
    - Use DescribeClusterscom um AWS SDK ou CLI
    - Use DescribeStatementcom um AWS SDK
    - Use ExecuteStatementcom um AWS SDK
    - Use GetStatementResultcom um AWS SDK
    - Use ListDatabasescom um AWS SDK
    - Use ModifyClustercom um AWS SDK ou CLI
- Cenários para o Amazon Redshift usando AWS SDKs
  - Criar um rastreador de itens do Amazon Redshift
  - Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI
  - Conceitos básicos de clusters de data warehouse provisionados

Exemplos básicos para o Amazon Redshift usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Redshift com SDKs da AWS .

Exemplos

- Olá, Amazon Redshift
- Aprenda as noções básicas do Amazon Redshift com um AWS SDK
- Ações para o Amazon Redshift usando AWS SDKs
  - Use CreateClustercom um AWS SDK ou CLI
  - Use DeleteClustercom um AWS SDK ou CLI
  - Use DescribeClusterscom um AWS SDK ou CLI
  - Use DescribeStatementcom um AWS SDK
  - Use ExecuteStatementcom um AWS SDK
  - Use GetStatementResultcom um AWS SDK
  - Use ListDatabasescom um AWS SDK
  - Use ModifyClustercom um AWS SDK ou CLI

Olá, Amazon Redshift

Os exemplos de código a seguir mostram como começar a usar o Amazon Redshift.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Main method to run the Hello Amazon Redshift example.
/// </summary>
/// <param name="args">Command line arguments (not used).</param>
```

```
    public static async Task Main(string[] args)
    {
        var redshiftClient = new AmazonRedshiftClient();

        Console.WriteLine("Hello, Amazon Redshift! Let's list available
clusters:");

        var clusters = new List<Cluster>();

        try
        {
            // Use pagination to retrieve all clusters.
            var clustersPaginator =
redshiftClient.Paginators.DescribeClusters(new DescribeClustersRequest());

            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");

            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }
        }
        catch (AmazonRedshiftException ex)
        {
            Console.WriteLine($"Couldn't list clusters. Here's why:
{ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeClustersa Referência AWS SDK para .NET da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
)

// main uses the AWS SDK for Go V2 to create a Redshift client
// and list up to 10 clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 redshiftClient := redshift.NewFromConfig(sdkConfig)
 count := 20
 fmt.Printf("Let's list up to %v clusters for your account.\n", count)
 result, err := redshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  MaxRecords: aws.Int32(int32(count)),
 })
```

```
 if err != nil {
  fmt.Printf("Couldn't list clusters for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Clusters) == 0 {
  fmt.Println("You don't have any clusters!")
  return
 }
 for _, cluster := range result.Clusters {
  fmt.Printf("\t%v : %v\n", *cluster.ClusterIdentifier, *cluster.ClusterStatus)
 }
}
```

- Para obter detalhes da API, consulte DescribeClustersa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import
 software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class HelloRedshift {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        listClustersPaginator(redshiftClient);
    }

    public static void listClustersPaginator(RedshiftClient redshiftClient) {
        DescribeClustersIterable clustersIterable =
 redshiftClient.describeClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.clusters().stream())
            .forEach(cluster -> System.out
                .println(" Cluster identifier: " + cluster.clusterIdentifier() +
 " status = " + cluster.clusterStatus()));
    }
}
```

- Para obter detalhes da API, consulte DescribeClustersa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_redshift(redshift_client):
    """
```

```
    Use the AWS SDK for Python (Boto3) to create an Amazon Redshift client and
 list
    the clusters in your account. This list might be empty if you haven't created
    any clusters.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param redshift_client: A Boto3 Redshift Client object.
    """
    print("Hello, Redshift! Let's list your clusters:")
    paginator = redshift_client.get_paginator("describe_clusters")
    clusters = []
    for page in paginator.paginate():
        clusters.extend(page["Clusters"])

    print(f"{len(clusters)} cluster(s) were found.")

    for cluster in clusters:
        print(f"  {cluster['ClusterIdentifier']}")


if __name__ == "__main__":
    hello_redshift(boto3.client("redshift"))
```

- Para obter detalhes da API, consulte a DescribeClustersReferência da API AWS SDK for Python (Boto3).

Aprenda as noções básicas do Amazon Redshift com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar um cluster do Redshift.
- Listar bancos de dados no cluster.
- Criar uma tabela chamada Filmes.
- Preencher a tabela Filmes.
- Consultar a tabela Filmes por ano.
- Modificar o cluster do Redshift.
- Excluir o cluster do Amazon Redshift.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe de wrapper do Redshift para gerenciar as operações.

```
/// <summary>
/// Wrapper class for Amazon Redshift operations.
/// </summary>
public class RedshiftWrapper
{
    private readonly IAmazonRedshift _redshiftClient;
    private readonly IAmazonRedshiftDataAPIService _redshiftDataClient;

    /// <summary>
    /// Constructor for RedshiftWrapper.
    /// </summary>
    /// <param name="redshiftClient">Amazon Redshift client.</param>
    /// <param name="redshiftDataClient">Amazon Redshift Data API client.</param>
    public RedshiftWrapper(IAmazonRedshift redshiftClient,
 IAmazonRedshiftDataAPIService redshiftDataClient)
    {
        _redshiftClient = redshiftClient;
        _redshiftDataClient = redshiftDataClient;
    }

    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
    /// <param name="masterUsername">The master username.</param>
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier,
 string databaseName,
```

```
        string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
    {
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create cluster. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe Amazon Redshift clusters.
    /// </summary>
    /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
    /// <returns>A list of clusters.</returns>
    public async Task<List<Cluster>> DescribeClustersAsync(string?
clusterIdentifier = null)
    {
        try
        {
```

```
            var clusters = new List<Cluster>();
            var request = new DescribeClustersRequest();
            if (!string.IsNullOrEmpty(clusterIdentifier))
            {
                request.ClusterIdentifier = clusterIdentifier;
            }

            var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance
window.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
 preferredMaintenanceWindow)
    {
        try
        {
            var request = new ModifyClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                PreferredMaintenanceWindow = preferredMaintenanceWindow
            };

            var response = await _redshiftClient.ModifyClusterAsync(request);
            Console.WriteLine($"The modified cluster was successfully modified
 and has {response.Cluster.PreferredMaintenanceWindow} as the maintenance
 window");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't modify cluster. Here's why:
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete an Amazon Redshift cluster without a final snapshot.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
 clusterIdentifier)
    {
        try
        {
            var request = new DeleteClusterRequest
            {
```

```
                ClusterIdentifier = clusterIdentifier,
                SkipFinalClusterSnapshot = true
            };

            var response = await _redshiftClient.DeleteClusterAsync(request);
            Console.WriteLine($"The {clusterIdentifier} was deleted");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete cluster. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();

            foreach (var database in response.Databases)
```

```
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create a table in the Redshift database.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> CreateTableAsync(string clusterIdentifier, string
database, string dbUser)
    {
        try
        {
            var sqlStatement = @"
                CREATE TABLE Movies (
                    id INTEGER PRIMARY KEY,
                    title VARCHAR(250) NOT NULL,
                    year INTEGER NOT NULL
                )";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
```

```
                Sql = sqlStatement
            };

            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine("Table created: Movies");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create table. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Insert a record into the Movies table using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="id">The movie ID.</param>
    /// <param name="title">The movie title.</param>
    /// <param name="year">The movie year.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> InsertMovieAsync(string clusterIdentifier, string
database, string dbUser,
        int id, string title, int year)
    {
        try
        {
            var sqlStatement = "INSERT INTO Movies (id, title, year) VALUES
(:id, :title, :year)";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
```

```
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "id", Value = id.ToString() },
                    new SqlParameter { Name = "title", Value = title },
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };

            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine($"Inserted: {title} ({year})");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't insert movie. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Query movies by year using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="year">The year to query.</param>
    /// <returns>A list of movie titles.</returns>
    public async Task<List<string>> QueryMoviesByYearAsync(string
clusterIdentifier, string database,
        string dbUser, int year)
    {
        try
        {
```

```
            var sqlStatement = "SELECT title FROM Movies WHERE year = :year";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };

            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            Console.WriteLine($"The identifier of the statement is
{response.Id}");

            await WaitForStatementToCompleteAsync(response.Id);

            var results = await GetStatementResultAsync(response.Id);
            var movieTitles = new List<string>();

            foreach (var row in results)
            {
                if (row.Count > 0)
                {
                    var title = row[0].StringValue;
                    Console.WriteLine($"The Movie title field is {title}");
                    movieTitles.Add(title);
                }
            }

            return movieTitles;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine($"Couldn't query movies. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
    public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
    {
        try
        {
            var request = new DescribeStatementRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get the results of a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A list of result rows.</returns>
```

```
    public async Task<List<List<Field>>> GetStatementResultAsync(string
statementId)
    {
        try
        {
            var request = new GetStatementResultRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.GetStatementResultAsync(request);
            return response.Records;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait for a statement to complete execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    private async Task WaitForStatementToCompleteAsync(string statementId)
    {
        var status = StatusString.SUBMITTED;
        DescribeStatementResponse? response = null;

        while (status == StatusString.SUBMITTED || status == StatusString.PICKED
|| status == StatusString.STARTED)
        {
            await Task.Delay(1000); // Wait 1 second
            response = await DescribeStatementAsync(statementId);
            status = response.Status;
            Console.WriteLine($"...{status}");
```

```
        }

        if (status == StatusString.FINISHED)
        {
            Console.WriteLine("The statement is finished!");
        }
        else
        {
            var errorMessage = response?.Error ?? "Unknown error";
            Console.WriteLine($"The statement failed with status: {status}");
            Console.WriteLine($"Error message: {errorMessage}");
        }
    }

    /// <summary>
    /// Wait for a cluster to become available.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public async Task WaitForClusterAvailableAsync(string clusterIdentifier)
    {
        Console.WriteLine($"Wait until {clusterIdentifier} is available. This may
 take a few minutes.");

        var startTime = DateTime.Now;
        var clusters = await DescribeClustersAsync(clusterIdentifier);

        while (clusters[0].ClusterStatus != "available")
        {
            var elapsed = DateTime.Now - startTime;
            Console.WriteLine($"Elapsed Time: {elapsed:mm\\:ss} - Waiting for
 cluster...");

            await Task.Delay(5000); // Wait 5 seconds
            clusters = await DescribeClustersAsync(clusterIdentifier);
        }

        var totalElapsed = DateTime.Now - startTime;
        Console.WriteLine($"Cluster is available! Total Elapsed Time:
 {totalElapsed:mm\\:ss}");
    }
}
```

Execute um cenário interativo que demonstra os fundamentos do Redshift.

```
/// <summary>
/// Amazon Redshift Getting Started Scenario.
/// </summary>
public class RedshiftBasics
{
    public static bool IsInteractive = true;
    public static RedshiftWrapper? Wrapper = null;
    public static ILogger logger = null!;
    private static readonly string _moviesFilePath = "../../../../../../
resources/sample_files/movies.json";

    /// <summary>
    /// Main method for the Amazon Redshift Getting Started scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRedshift>()
                    .AddAWSService<IAmazonRedshiftDataAPIService>()
                    .AddTransient<RedshiftWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<RedshiftBasics>();

        Wrapper = host.Services.GetRequiredService<RedshiftWrapper>();

        await RunScenarioAsync();
    }

    /// <summary>
    /// Run the complete Amazon Redshift scenario.
    /// </summary>
    public static async Task RunScenarioAsync()
    {
        // Set all variables to default values
        string userName = "awsuser";
        string userPassword = "AwsUser1000";
        string clusterIdentifier = "redshift-cluster-movies";
```

```
        var databaseName = "dev";
        int recordCount = 50;
        int year = 2013;
        try
        {
            Console.WriteLine(

 "==============================================================================");
            Console.WriteLine("Welcome to the Amazon Redshift SDK Getting Started
 scenario.");
            Console.WriteLine(
                "This .NET program demonstrates how to interact with Amazon
 Redshift by using the AWS SDK for .NET.");
            Console.WriteLine("Let's get started...");
            Console.WriteLine(

 "==============================================================================");

            // Step 1: Get user credentials (if interactive)
            if (IsInteractive)
            {
                Console.WriteLine("Please enter a user name for the cluster
 (default is awsuser):");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    userName = userInput;


 Console.WriteLine("==================================================================
                Console.WriteLine("Please enter a user password for the cluster
 (default is AwsUser1000):");
                var passwordInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(passwordInput))
                    userPassword = passwordInput;


 Console.WriteLine("==================================================================

                // Step 2: Get cluster identifier
                Console.WriteLine("Enter a cluster id value (default is redshift-
cluster-movies):");
                var clusterInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(clusterInput))
                    clusterIdentifier = clusterInput;
```

```
            }
            else
            {
                Console.WriteLine($"Using default values: userName={userName},
clusterIdentifier={clusterIdentifier}");
            }

            // Step 3: Create Redshift cluster
            await Wrapper!.CreateClusterAsync(clusterIdentifier, databaseName,
userName, userPassword);

Console.WriteLine("=========================================================================

            // Step 4: Wait for cluster to become available

Console.WriteLine("=========================================================================
            await Wrapper.WaitForClusterAvailableAsync(clusterIdentifier);

Console.WriteLine("=========================================================================

            // Step 5: List databases

Console.WriteLine("=========================================================================
            Console.WriteLine($" When you created {clusterIdentifier}, the dev
database is created by default and used in this scenario.");
            Console.WriteLine(" To create a custom database, you need to have a
CREATEDB privilege.");
            Console.WriteLine(" For more information, see the documentation here:
https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

Console.WriteLine("=========================================================================


Console.WriteLine("=========================================================================
            Console.WriteLine($"List databases in {clusterIdentifier}");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
```

```
            }
            await Wrapper.ListDatabasesAsync(clusterIdentifier, userName,
databaseName);

Console.WriteLine("=========================================================================

            // Step 6: Create Movies table

Console.WriteLine("=========================================================================
            Console.WriteLine("Now you will create a table named Movies.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.CreateTableAsync(clusterIdentifier, databaseName,
userName);

Console.WriteLine("=========================================================================

            // Step 7: Populate the Movies table

Console.WriteLine("=========================================================================
            Console.WriteLine("Populate the Movies table using the Movies.json
file.");

            if (IsInteractive)
            {
                Console.WriteLine("Specify the number of records you would like
to add to the Movies Table.");
                Console.WriteLine("Please enter a value between 50 and 200.");
                Console.Write("Enter a value: ");

                var recordCountInput = Console.ReadLine();
                if (int.TryParse(recordCountInput, out var inputCount) &&
inputCount is >= 50 and <= 200)
                {
                    recordCount = inputCount;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{recordCount}.");
                }
```

```
            }
            else
            {
                Console.WriteLine($"Using default record count: {recordCount}");
            }

            await PopulateMoviesTableAsync(clusterIdentifier, databaseName,
userName, recordCount);
            Console.WriteLine($"{recordCount} records were added to the Movies
table.");

Console.WriteLine("=========================================================================

            // Step 8 & 9: Query movies by year

Console.WriteLine("=========================================================================
            Console.WriteLine("Query the Movies table by year. Enter a value
between 2012-2014.");

            if (IsInteractive)
            {
                Console.Write("Enter a year: ");
                var yearInput = Console.ReadLine();
                if (int.TryParse(yearInput, out var inputYear) && inputYear is >=
2012 and <= 2014)
                {
                    year = inputYear;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{year}.");
                }
            }
            else
            {
                Console.WriteLine($"Using default year: {year}");
            }

            await Wrapper.QueryMoviesByYearAsync(clusterIdentifier, databaseName,
userName, year);

Console.WriteLine("=========================================================================
```

```
            // Step 10: Modify the cluster

 Console.WriteLine("==========================================================================
            Console.WriteLine("Now you will modify the Redshift cluster.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.ModifyClusterAsync(clusterIdentifier, "wed:07:30-
wed:08:00");

 Console.WriteLine("==========================================================================

            // Step 11 & 12: Delete cluster confirmation

 Console.WriteLine("==========================================================================
            if (IsInteractive)
            {
                Console.WriteLine("Would you like to delete the Amazon Redshift
 cluster? (y/n)");
                var deleteResponse = Console.ReadLine();
                if (deleteResponse?.ToLower() == "y")
                {
                    await
 Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
                }
            }
            else
            {
                Console.WriteLine("Deleting the Amazon Redshift cluster...");
                await
 Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            }

 Console.WriteLine("==========================================================================



 Console.WriteLine("==========================================================================
            Console.WriteLine("This concludes the Amazon Redshift SDK Getting
 Started scenario.");

 Console.WriteLine("==========================================================================
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during the scenario:
{ex.Message}");
            Console.WriteLine("Deleting the Amazon Redshift cluster...");
            await Wrapper!.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            throw;
        }
    }

    /// <summary>
    /// Populate the Movies table with data from the JSON file.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="recordCount">Number of records to insert.</param>
    private static async Task PopulateMoviesTableAsync(string clusterIdentifier,
string database, string dbUser, int recordCount)
    {
        if (!File.Exists(_moviesFilePath))
        {
            throw new FileNotFoundException($"Required movies data file not found
at: {_moviesFilePath}");
        }

        var jsonContent = await File.ReadAllTextAsync(_moviesFilePath);
        var options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };
        var movies = JsonSerializer.Deserialize<List<Movie>>(jsonContent,
options);

        if (movies == null || movies.Count == 0)
        {
            throw new InvalidOperationException("Failed to parse movies JSON file
or file is empty.");
        }

        var insertCount = Math.Min(recordCount, movies.Count);

        for (int i = 0; i < insertCount; i++)
        {
```

```
            var movie = movies[i];
            await Wrapper!.InsertMovieAsync(clusterIdentifier, database, dbUser,
 i, movie.Title, movie.Year);
        }
    }

    /// <summary>
    /// Movie data model.
    /// </summary>
    private class Movie
    {
        public string Title { get; set; } = string.Empty;
        public int Year { get; set; }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package scenarios

import (
 "context"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 redshift_types "github.com/aws/aws-sdk-go-v2/service/redshift/types"
 redshiftdata_types "github.com/aws/aws-sdk-go-v2/service/redshiftdata/types"
 "github.com/aws/aws-sdk-go-v2/service/secretsmanager"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/redshift/actions"

 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshiftdata"
)

// IScenarioHelper abstracts input and wait functions from a scenario so that
 they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// RedshiftBasicsScenario separates the steps of this scenario into individual
 functions so that
```

```
// they are simpler to read and understand.
type RedshiftBasicsScenario struct {
 sdkConfig         aws.Config
 helper            IScenarioHelper
 questioner        demotools.IQuestioner
 pauser            demotools.IPausable
 filesystem        demotools.IFileSystem
 redshiftActor     *actions.RedshiftActions
 redshiftDataActor *actions.RedshiftDataActions
 secretsmanager    *SecretsManager
}

// SecretsManager is used to retrieve username and password information from a
 secure service.
type SecretsManager struct {
 SecretsManagerClient *secretsmanager.Client
}

// RedshiftBasics constructs a new Redshift Basics runner.
func RedshiftBasics(sdkConfig aws.Config, questioner demotools.IQuestioner,
 pauser demotools.IPausable, filesystem demotools.IFileSystem, helper
 IScenarioHelper) RedshiftBasicsScenario {
 scenario := RedshiftBasicsScenario{
  sdkConfig:         sdkConfig,
  helper:            helper,
  questioner:        questioner,
  pauser:            pauser,
  filesystem:        filesystem,
  secretsmanager:    &SecretsManager{SecretsManagerClient:
 secretsmanager.NewFromConfig(sdkConfig)},
  redshiftActor:     &actions.RedshiftActions{RedshiftClient:
 redshift.NewFromConfig(sdkConfig)},
  redshiftDataActor: &actions.RedshiftDataActions{RedshiftDataClient:
 redshiftdata.NewFromConfig(sdkConfig)},
 }
 return scenario
}


// Movie makes it easier to use Movie objects given in json format.
type Movie struct {
 ID    int    `json:"id"`
 Title string `json:"title"`
 Year  int    `json:"year"`
```

```
}


// User makes it easier to get the User data back from SecretsManager and use it
 later.
type User struct {
 Username string `json:"userName"`
 Password string `json:"userPassword"`
}

// Run runs the RedshiftBasics interactive example that shows you how to use
 Amazon
// Redshift and how to interact with its common endpoints.
//
// 0. Retrieve username and password information to access Redshift.
// 1. Create a cluster.
// 2. Wait for the cluster to become available.
// 3. List the available databases in the region.
// 4. Create a table named "Movies" in the "dev" database.
// 5. Populate the movies table from the "movies.json" file.
// 6. Query the movies table by year.
// 7. Modify the cluster's maintenance window.
// 8. Optionally clean up all resources created during this demo.
//
// This example creates an Amazon Redshift service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func (runner *RedshiftBasicsScenario) Run(ctx context.Context) {

 user := User{}
 secretId := "s3express/basics/secrets"
 clusterId := "demo-cluster-1"
 maintenanceWindow := "wed:07:30-wed:08:00"
 databaseName := "dev"
 tableName := "Movies"
 fileName := "Movies.json"
 nodeType := "ra3.xlplus"
 clusterType := "single-node"

 defer func() {
```

```
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := runner.questioner.(*demotools.MockQuestioner)
   if isMock || runner.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   runner.cleanUpResources(ctx, clusterId, databaseName, tableName,
 user.Username, runner.questioner)
  }
 }()

 // Retrieve the userName and userPassword from SecretsManager
 output, err := runner.secretsmanager.SecretsManagerClient.GetSecretValue(ctx,
 &secretsmanager.GetSecretValueInput{
  SecretId: aws.String(secretId),
 })
 if err != nil {
  log.Printf("There was a problem getting the secret value: %s", err)
  log.Printf("Please make sure to create a secret named 's3express/basics/
secrets' with keys of 'userName' and 'userPassword'.")
  panic(err)
 }

 err = json.Unmarshal([]byte(*output.SecretString), &user)
 if err != nil {
  log.Printf("There was a problem parsing the secret value from JSON: %s", err)
  panic(err)
 }

 // Create the Redshift cluster
 _, err = runner.redshiftActor.CreateCluster(ctx, clusterId, user.Username,
 user.Password, nodeType, clusterType, true)
 if err != nil {
  var clusterAlreadyExistsFault *redshift_types.ClusterAlreadyExistsFault
  if errors.As(err, &clusterAlreadyExistsFault) {
   log.Println("Cluster already exists. Continuing.")
  } else {
   log.Println("Error creating cluster.")
   panic(err)
  }
 }

 // Wait for the cluster to become available
```

```
waiter :=
redshift.NewClusterAvailableWaiter(runner.redshiftActor.RedshiftClient)
err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
 ClusterIdentifier: aws.String(clusterId),
}, 5*time.Minute)
if err != nil {
 log.Println("An error occurred waiting for the cluster.")
 panic(err)
}

// Get some info about the cluster
describeOutput, err := runner.redshiftActor.DescribeClusters(ctx, clusterId)
if err != nil {
 log.Println("Something went wrong trying to get information about the
cluster.")
 panic(err)
}
log.Println("Here's some information about the cluster.")
log.Printf("The cluster's status is %s",
*describeOutput.Clusters[0].ClusterStatus)
log.Printf("The cluster was created at %s",
*describeOutput.Clusters[0].ClusterCreateTime)

// List databases
log.Println("List databases in", clusterId)
runner.questioner.Ask("Press Enter to continue...")
err = runner.redshiftDataActor.ListDatabases(ctx, clusterId, databaseName,
user.Username)
if err != nil {
 log.Printf("Failed to list databases: %v\n", err)
 panic(err)
}

// Create the "Movies" table
log.Println("Now you will create a table named " + tableName + ".")
runner.questioner.Ask("Press Enter to continue...")
err = nil
result, err := runner.redshiftDataActor.CreateTable(ctx, clusterId,
databaseName, tableName, user.Username, runner.pauser, []string{"title
VARCHAR(256)", "year INT"})
if err != nil {
 log.Printf("Failed to create table: %v\n", err)
 panic(err)
}
```

```
 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }
 query := actions.RedshiftQuery{
  Context: ctx,
  Input:   describeInput,
  Result:  result,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute query: %v\n", err)
  panic(err)
 }
 log.Printf("Successfully executed query\n")

 // Populate the "Movies" table
 runner.PopulateMoviesTable(ctx, clusterId, databaseName, tableName,
 user.Username, fileName)

 // Query the "Movies" table by year
 log.Println("Query the Movies table by year.")
 year := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 2012, 2014),
  demotools.InIntRange{Lower: 2012, Upper: 2014})
 runner.QueryMoviesByYear(ctx, clusterId, databaseName, tableName, user.Username,
 year)

 // Modify the cluster's maintenance window
 runner.redshiftActor.ModifyCluster(ctx, clusterId, maintenanceWindow)

 // Delete the Redshift cluster if confirmed
 runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username,
 runner.questioner)

 log.Println("Thanks for watching!")
}

// cleanUpResources asks the user if they would like to delete each resource
 created during the scenario, from most
// impactful to least impactful. If any choice to delete is made, further
 deletion attempts are skipped.
```

```
func (runner *RedshiftBasicsScenario) cleanUpResources(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string,
 questioner demotools.IQuestioner) {
 deleted := false
 var err error = nil
 if questioner.AskBool("Do you want to delete the entire cluster? This will clean
 up all resources. (y/n)", "y") {
  deleted, err = runner.redshiftActor.DeleteCluster(ctx, clusterId)
  if err != nil {
   log.Printf("Error deleting cluster: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete the dev table? This
 will clean up all inserted records but keep your cluster intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteTable(ctx, clusterId,
 databaseName, tableName, userName)
  if err != nil {
   log.Printf("Error deleting movies table: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete all rows in the Movies
 table? This will clean up all inserted records but keep your cluster and table
 intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteDataRows(ctx, clusterId,
 databaseName, tableName, userName, runner.pauser)
  if err != nil {
   log.Printf("Error deleting data rows: %v", err)
  }
 }
 if !deleted {
  log.Print("Please manually delete any unwanted resources.")
 }
}


// loadMoviesFromJSON takes the <fileName> file and populates a slice of Movie
 objects.
func (runner *RedshiftBasicsScenario) loadMoviesFromJSON(fileName string,
 filesystem demotools.IFileSystem) ([]Movie, error) {
 file, err := filesystem.OpenFile("../../resources/sample_files/" + fileName)
 if err != nil {
  return nil, err
 }
 defer filesystem.CloseFile(file)
```

```
 var movies []Movie
 err = json.NewDecoder(file).Decode(&movies)
 if err != nil {
  return nil, err
 }

 return movies, nil
}



// PopulateMoviesTable reads data from the <fileName> file and inserts records
 into the "Movies" table.
func (runner *RedshiftBasicsScenario) PopulateMoviesTable(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string,
 fileName string) {
 log.Println("Populate the " + tableName + " table using the " + fileName + "
 file.")
 numRecords := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 10, 100),
  demotools.InIntRange{Lower: 10, Upper: 100})

 movies, err := runner.loadMoviesFromJSON(fileName, runner.filesystem)
 if err != nil {
  log.Printf("Failed to load movies from JSON: %v\n", err)
  panic(err)
 }

 var sqlStatements []string

 for i, movie := range movies {
  if i >= numRecords {
   break
  }

  sqlStatement := fmt.Sprintf(`INSERT INTO %s (title, year) VALUES ('%s', %d);`,
   tableName,
   strings.Replace(movie.Title, "'", "''", -1), // Double any single quotes to
 escape them
   movie.Year)

  sqlStatements = append(sqlStatements, sqlStatement)
 }
```

```
 input := &redshiftdata.BatchExecuteStatementInput{
  ClusterIdentifier: aws.String(clusterId),
  Database:          aws.String(databaseName),
  DbUser:            aws.String(userName),
  Sqls:              sqlStatements,
 }

 result, err := runner.redshiftDataActor.ExecuteBatchStatement(ctx, *input)
 if err != nil {
  log.Printf("Failed to execute batch statement: %v\n", err)
  panic(err)
 }

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }

 query := actions.RedshiftQuery{
  Context: ctx,
  Result:  result,
  Input:   describeInput,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute batch insert query: %v\n", err)
  return
 }
 log.Printf("Successfully executed batch statement\n")

 log.Printf("%d records were added to the Movies table.\n", numRecords)
}



// QueryMoviesByYear retrieves only movies from the "Movies" table which match
 the given year.
func (runner *RedshiftBasicsScenario) QueryMoviesByYear(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string, year
 int) {

 sqlStatement := fmt.Sprintf(`SELECT title FROM %s WHERE year = %d;`, tableName,
 year)
```

```
input := &redshiftdata.ExecuteStatementInput{
 ClusterIdentifier: aws.String(clusterId),
 Database:          aws.String(databaseName),
 DbUser:            aws.String(userName),
 Sql:               aws.String(sqlStatement),
}

result, err := runner.redshiftDataActor.ExecuteStatement(ctx, *input)
if err != nil {
 log.Printf("Failed to query movies: %v\n", err)
 panic(err)
}

log.Println("The identifier of the statement is ", *result.Id)

describeInput := redshiftdata.DescribeStatementInput{
 Id: result.Id,
}

query := actions.RedshiftQuery{
 Context: ctx,
 Input:   describeInput,
 Result:  result,
}
err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
if err != nil {
 log.Printf("Failed to execute query: %v\n", err)
 panic(err)
}
log.Printf("Successfully executed query\n")

getResultOutput, err := runner.redshiftDataActor.GetStatementResult(ctx,
*result.Id)
if err != nil {
 log.Printf("Failed to query movies: %v\n", err)
 panic(err)
}
for _, row := range getResultOutput.Records {
 for _, col := range row {
  title, ok := col.(*redshiftdata_types.FieldMemberStringValue)
  if !ok {
   log.Println("Failed to parse the field")
  } else {
   log.Printf("The Movie title field is %s\n", title.Value)
```

```
    }
   }
  }
 }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstra os recursos do Amazon Redshift.

```
import com.example.redshift.User;
import com.google.gson.Gson;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.redshift.model.ClusterAlreadyExistsException;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
```

```
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import
 software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 *
 *  This example requires an AWS Secrets Manager secret that contains the
 *  database credentials. If you do not create a
 *  secret that specifies user name and password, this example will not work. For
 details, see:
 *
 *  https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
 * 6. Inserts a specified number of records into the "Movies" table by reading
 the Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
```

```
 * 11. If confirmed, deletes the specified Redshift cluster.
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftScenario.class);

    static RedshiftActions redshiftActions = new RedshiftActions();
    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <jsonFilePath> <secretName>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate
 that file in ../../../resources/sample_files/movies.json)
                secretName - The name of the secret that belongs to Secret
 Manager that stores the user name and password used in this scenario.
            """;

        if (args.length != 2) {
            logger.info(usage);
            return;
        }

        String jsonFilePath = args[0];
        String secretName = args[1];
        Scanner scanner = new Scanner(System.in);
        logger.info(DASHES);
        logger.info("Welcome to the Amazon Redshift SDK Basics scenario.");
        logger.info("""
            This Java program demonstrates how to interact with Amazon Redshift
 by using the AWS SDK for Java (v2).\s
            Amazon Redshift is a fully managed, petabyte-scale data warehouse
 service hosted in the cloud.

            The program's primary functionalities include cluster creation,
 verification of cluster readiness,\s
            list databases, table creation, data population within the table, and
 execution of SQL statements.
```

```
            Furthermore, it demonstrates the process of querying data from the
 Movie table.\s

            Upon completion of the program, all AWS resources are cleaned up.
            """);

        logger.info("Lets get started...");
        logger.info("""
            First, we will retrieve the user name and password from Secrets
 Manager.

            Using Amazon Secrets Manager to store Redshift credentials provides
 several security benefits.
            It allows you to securely store and manage sensitive information,
 such as passwords, API keys, and
            database credentials, without embedding them directly in your
 application code.

            More information can be found here:

            https://docs.aws.amazon.com/secretsmanager/latest/userguide/
integrating_how-services-use-secrets_RS.html
            """);
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(user, scanner, jsonFilePath);
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }
    }

    private static void runScenario(User user, Scanner scanner,  String
 jsonFilePath) throws Throwable {
        String databaseName = "dev";
        System.out.println(DASHES);
        logger.info("Create a Redshift Cluster");
```

```
        logger.info("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
        logger.info("Enter a cluster id value or accept the default by hitting
Enter (default is redshift-cluster-movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
        try {
            CompletableFuture<CreateClusterResponse> future =
redshiftActions.createClusterAsync(clusterId, user.getUserName(),
user.getUserPassword());
            CreateClusterResponse response = future.join();
            logger.info("Cluster successfully created. Cluster Identifier {} ",
response.cluster().clusterIdentifier());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof ClusterAlreadyExistsException) {
                logger.info("The Cluster {} already exists. Moving on...",
clusterId);
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Wait until {} is available.", clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.waitForClusterReadyAsync(clusterId);
            future.join();
            logger.info("Cluster is ready!");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftException redshiftEx) {
                logger.info("Redshift error occurred: Error message: {}, Error
code {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        String databaseInfo = """
            When you created $clusteridD, the dev database is created by default
 and used in this scenario.\s

            To create a custom database, you need to have a CREATEDB privilege.\s
            For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
           """.replace("$clusteridD", clusterId);

        logger.info(databaseInfo);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("List databases in {} ",clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.listAllDatabasesAsync(clusterId, user.getUserName(), "dev");
            future.join();
            logger.info("Databases listed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.error("Redshift Data error occurred: {} Error code: {}",
 redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.error("An unexpected error occurred: {}",
 rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will create a table named Movies.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<ExecuteStatementResponse> future =
redshiftActions.createTableAsync(clusterId, databaseName, user.getUserName());
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Populate the Movies table using the Movies.json file.");
        logger.info("Specify the number of records you would like to add to the
Movies Table.");
        logger.info("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            logger.info("Enter a value: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a value between 50 and
200.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        try {
            redshiftActions.popTableAsync(clusterId, databaseName,
user.getUserName(), jsonFilePath, numRecords).join();  // Wait for the operation
to complete
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
```

```
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            logger.info("Enter a year: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a valid year between
2012 and 2014.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id;
        try {
            CompletableFuture<String> future =
redshiftActions.queryMoviesByYearAsync(databaseName, user.getUserName(),
movieYear, clusterId);
            id = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        logger.info("The identifier of the statement is " + id);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.checkStatementAsync(id);
```

```
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = redshiftActions.getResultsAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will modify the Redshift cluster.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ModifyClusterResponse> future =
redshiftActions.modifyClusterAsync(clusterId);;
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Would you like to delete the Amazon Redshift cluster? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete {} ", clusterId);
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<DeleteClusterResponse> future =
 redshiftActions.deleteRedshiftClusterAsync(clusterId);;
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof RedshiftDataException redshiftEx) {
                    logger.info("Redshift Data error occurred: {} Error code:
 {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: {}",
 rt.getMessage());
                }
                throw cause;
            }
        } else {
            logger.info("The {}  was not deleted", clusterId);
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the Amazon Redshift SDK Basics scenario.");
        logger.info(DASHES);
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_EAST_1;
        return SecretsManagerClient.builder()
            .region(region)
```

```
            .build();
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    // Get the Amazon Redshift credentials from AWS Secrets Manager.
    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }
}
```

Uma classe de wrapper para métodos do SDK do Amazon Redshift.

```
public class RedshiftActions {

    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftActions.class);
    private static RedshiftDataAsyncClient redshiftDataAsyncClient;

    private static RedshiftAsyncClient redshiftAsyncClient;
```

```
    private static RedshiftAsyncClient getAsyncClient() {
        if (redshiftAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftAsyncClient = RedshiftAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftAsyncClient;
    }

    private static RedshiftDataAsyncClient getAsyncDataClient() {
        if (redshiftDataAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftDataAsyncClient = RedshiftDataAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
```

```
                .build();
        }
        return redshiftDataAsyncClient;
    }

    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is
ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
```

```
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void>
checkClusterStatusAsync(DescribeClustersRequest clustersRequest, long startTime)
{
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
```

```
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists all databases asynchronously for the specified cluster, database
user, and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database
listing is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response ->
{
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
```

```
        });
    }

    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the
SQL statement execution
     * @throws RuntimeException if there is an error creating the table
     */
    public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies (" +
                "id INT PRIMARY KEY, " +
                "title VARCHAR(100), " +
                "year INT)")
            .build();

        return getAsyncDataClient().executeStatement(createTableRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }

    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
```

```
    * @param fileName    the name of the JSON file
    * @param number      the number of records to process
    * @return a CompletableFuture that completes with the number of records
added to the Movies table
    */
   public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
       return CompletableFuture.supplyAsync(() -> {
               try {
                   JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                   JsonNode rootNode = new ObjectMapper().readTree(parser);
                   Iterator<JsonNode> iter = rootNode.iterator();
                   return iter;
               } catch (IOException e) {
                   throw new RuntimeException("Failed to read or parse JSON
file: " + e.getMessage(), e);
               }
           }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   logger.info("Error {} ", exception.getMessage());
               } else {
                   logger.info("{} records were added to the Movies table." ,
result);
               }
           });
   }

   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
                       break;
                   JsonNode currentNode = iter.next();
                   int year = currentNode.get("year").asInt();
                   String title = currentNode.get("title").asText();

                   // Use SqlParameter to avoid SQL injection.
                   List<SqlParameter> parameterList = new ArrayList<>();
```

```
                    String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                    SqlParameter idParam = SqlParameter.builder()
                        .name("id")
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();

                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();


getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }

    /**
```

```
    * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
    *
    * @param sqlId the ID of the SQL statement to check
    * @return a {@link CompletableFuture} that completes when the SQL
statement's status is either "FINISHED" or "FAILED"
    */
   public CompletableFuture<Void> checkStatementAsync(String sqlId) {
       DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
           .id(sqlId)
           .build();

       return getAsyncDataClient().describeStatement(statementRequest)
           .thenCompose(response -> {
               String status = response.statusAsString();
               logger.info("... Status: {} ", status);

               if ("FAILED".equals(status)) {
                   throw new RuntimeException("The Query Failed. Ending
program");
               } else if ("FINISHED".equals(status)) {
                   return CompletableFuture.completedFuture(null);
               } else {
                   // Sleep for 1 second and recheck status
                   return CompletableFuture.runAsync(() -> {
                       try {
                           TimeUnit.SECONDS.sleep(1);
                       } catch (InterruptedException e) {
                           throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                       }
                   }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
               }
           }).whenComplete((result, exception) -> {
               if (exception != null) {
                   // Handle exceptions
                   logger.info("Error: {} ", exception.getMessage());
               } else {
                   logger.info("The statement is finished!");
               }
           });
   }
```

```
   /**
    * Asynchronously retrieves the results of a statement execution.
    *
    * @param statementId the ID of the statement for which to retrieve the
results
    * @return a {@link CompletableFuture} that completes when the statement
result has been processed
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: "
+ exception.getMessage(), exception);
               }

               // Extract and print the field values using streams if the
response is valid.
               response.records().stream()
                   .flatMap(List::stream)
                   .map(Field::stringValue)
                   .filter(value -> value != null)
                   .forEach(value -> System.out.println("The Movie title field
is " + value));

               return response;
           }).thenAccept(response -> {
               // Optionally add more logic here if needed after handling the
response
           });
   }


   /**
    * Asynchronously queries movies by a given year from a Redshift database.
    *
```

```
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String
clusterId) {

        String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

```
    /**
     * Modifies an Amazon Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the cluster to be modified
     * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
     */
    public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
        ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified
and has "
                        + clusterResponse.cluster().preferredMaintenanceWindow()
+ " as the maintenance window");
                }
            });
    }

    /**
     * Deletes a Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
     */
    public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
```

```
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {}",
 exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftScenario:
    """Runs an interactive scenario that shows how to get started with
 Redshift."""

    def __init__(self, redshift_wrapper, redshift_data_wrapper):
        self.redshift_wrapper = redshift_wrapper
        self.redshift_data_wrapper = redshift_data_wrapper

    def redhift_scenario(self, json_file_path):
        database_name = "dev"

        print(DASHES)
        print("Welcome to the Amazon Redshift SDK Getting Started example.")
        print(
            """
      This Python program demonstrates how to interact with Amazon Redshift
      using the AWS SDK for Python (Boto3).

      Amazon Redshift is a fully managed, petabyte-scale data warehouse
      service hosted in the cloud.

      The program's primary functionalities include cluster creation,
      verification of cluster readiness, listing databases, table creation,
      populating data within the table, and executing SQL statements.

      It also demonstrates querying data from the Movies table.

      Upon completion, all AWS resources are cleaned up.
    """
        )
        if not os.path.isfile(json_file_path):
            logging.error(f"The file {json_file_path} does not exist.")
```

```
            return

        print("Let's get started...")
        user_name = q.ask("Please enter your user name (default is awsuser):")
        user_name = user_name if user_name else "awsuser"

        print(DASHES)
        user_password = q.ask(
            "Please enter your user password (default is AwsUser1000):"
        )
        user_password = user_password if user_password else "AwsUser1000"

        print(DASHES)
        print(
            """A Redshift cluster refers to the collection of computing resources
 and storage that work
            together to process and analyze large volumes of data."""
        )
        cluster_id = q.ask(
            "Enter a cluster identifier value (default is redshift-cluster-
movies): "
        )
        cluster_id = cluster_id if cluster_id else "redshift-cluster-movies"

        self.redshift_wrapper.create_cluster(
            cluster_id, "ra3.4xlarge", user_name, user_password, True, 2
        )

        print(DASHES)
        print(f"Wait until {cluster_id} is available. This may take a few
 minutes...")
        q.ask("Press Enter to continue...")

        self.wait_cluster_available(cluster_id)

        print(DASHES)

        print(
            f"""
       When you created {cluster_id}, the dev database is created by default and
 used in this scenario.

       To create a custom database, you need to have a CREATEDB privilege.
       For more information, see the documentation here:
```

```
        https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
      """
        )
        q.ask("Press Enter to continue...")
        print(DASHES)

        print(DASHES)
        print(f"List databases in {cluster_id}")
        q.ask("Press Enter to continue...")
        databases = self.redshift_data_wrapper.list_databases(
            cluster_id, database_name, user_name
        )
        print(f"The cluster contains {len(databases)} database(s).")
        for database in databases:
            print(f"    Database: {database}")
        print(DASHES)

        print(DASHES)
        print("Now you will create a table named Movies.")
        q.ask("Press Enter to continue...")

        self.create_table(cluster_id, database_name, user_name)

        print(DASHES)

        print("Populate the Movies table using the Movies.json file.")
        print(
            "Specify the number of records you would like to add to the Movies
Table."
        )
        print("Please enter a value between 50 and 200.")

        while True:
            try:
                num_records = int(q.ask("Enter a value: ", q.is_int))
                if 50 <= num_records <= 200:
                    break
                else:
                    print("Invalid input. Please enter a value between 50 and
200.")
            except ValueError:
                print("Invalid input. Please enter a value between 50 and 200.")

        self.populate_table(
```

```
            cluster_id, database_name, user_name, json_file_path, num_records
        )

        print(DASHES)
        print("Query the Movies table by year. Enter a value between 2012-2014.")

        while True:
            movie_year = int(q.ask("Enter a year: ", q.is_int))
            if 2012 <= movie_year <= 2014:
                break
            else:
                print("Invalid input. Please enter a valid year between 2012 and
2014.")

        # Function to query database
        sql_id = self.query_movies_by_year(
            database_name, user_name, movie_year, cluster_id
        )

        print(f"The identifier of the statement is {sql_id}")

        print("Checking statement status...")
        self.wait_statement_finished(sql_id)
        result = self.redshift_data_wrapper.get_statement_result(sql_id)

        self.display_movies(result)

        print(DASHES)

        print(DASHES)
        print("Now you will modify the Redshift cluster.")
        q.ask("Press Enter to continue...")

        preferred_maintenance_window = "wed:07:30-wed:08:00"
        self.redshift_wrapper.modify_cluster(cluster_id,
preferred_maintenance_window)

        print(DASHES)

        print(DASHES)
        delete = q.ask("Do you want to delete the cluster? (y/n) ", q.is_yesno)

        if delete:
            print(f"You selected to delete {cluster_id}")
```

```
            q.ask("Press Enter to continue...")
            self.redshift_wrapper.delete_cluster(cluster_id)
        else:
            print(f"Cluster {cluster_id}cluster_id was not deleted")

        print(DASHES)
        print("This concludes the Amazon Redshift SDK Getting Started scenario.")
        print(DASHES)

    def create_table(self, cluster_id, database, username):
        self.redshift_data_wrapper.execute_statement(
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql="CREATE TABLE Movies (statement_id INT PRIMARY KEY, title
VARCHAR(100), year INT)",
        )

        print("Table created: Movies")


    def populate_table(self, cluster_id, database, username, file_name, number):
        with open(file_name) as f:
            data = json.load(f)

        i = 0
        for record in data:
            if i == number:
                break

            statement_id = i
            title = record["title"]
            year = record["year"]
            i = i + 1
            parameters = [
                {"name": "statement_id", "value": str(statement_id)},
                {"name": "title", "value": title},
                {"name": "year", "value": str(year)},
            ]

            self.redshift_data_wrapper.execute_statement(
                cluster_identifier=cluster_id,
                database_name=database,
                user_name=username,
```

```
                sql="INSERT INTO Movies VALUES(:statement_id, :title, :year)",
                parameter_list=parameters,
            )

        print(f"{i} records inserted into Movies table")

    def wait_cluster_available(self, cluster_id):
        """
        Waits for a cluster to be available.

        :param cluster_id: The cluster identifier.

        Note: The cluster_available waiter can also be used.
        It is not used in this case to allow an elapsed time message.
        """
        cluster_ready = False
        start_time = time.time()

        while not cluster_ready:
            time.sleep(30)
            cluster = self.redshift_wrapper.describe_clusters(cluster_id)
            status = cluster[0]["ClusterStatus"]
            if status == "available":
                cluster_ready = True
            elif status != "creating":
                raise Exception(
                    f"Cluster {cluster_id} creation failed with status {status}."
                )

            elapsed_seconds = int(round(time.time() - start_time))
            minutes = int(elapsed_seconds // 60)
            seconds = int(elapsed_seconds % 60)

            print(f"Elapsed Time: {minutes}:{seconds:02d} - status {status}...")

            if minutes > 30:
                raise Exception(
                    f"Cluster {cluster_id} is not available after 30 minutes."
                )

    def query_movies_by_year(self, database, username, year, cluster_id):
        sql = "SELECT * FROM Movies WHERE year = :year"

        params = [{"name": "year", "value": str(year)}]
```

```
        response = self.redshift_data_wrapper.execute_statement(
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql=sql,
            parameter_list=params,
        )

        return response["Id"]

    @staticmethod
    def display_movies(response):
        metadata = response["ColumnMetadata"]
        records = response["Records"]

        title_column_index = None
        for i in range(len(metadata)):
            if metadata[i]["name"] == "title":
                title_column_index = i
                break

        if title_column_index is None:
            print("No title column found.")
            return

        print(f"Found {len(records)} movie(s).")
        for record in records:
            print(f"   {record[title_column_index]['stringValue']}")

    def wait_statement_finished(self, sql_id):
        while True:
            time.sleep(1)
            response = self.redshift_data_wrapper.describe_statement(sql_id)
            status = response["Status"]
            print(f"Statement status is {status}.")

            if status == "FAILED":
                print(f"The query failed because {response['Error']}. Ending
program")
                raise Exception("The Query Failed. Ending program")
            elif status == "FINISHED":
                break
```

Função principal mostrando a implementação do cenário.

```
def main():
    redshift_client = boto3.client("redshift")
    redshift_data_client = boto3.client("redshift-data")
    redshift_wrapper = RedshiftWrapper(redshift_client)
    redshift_data_wrapper = RedshiftDataWrapper(redshift_data_client)
    redshift_scenario = RedshiftScenario(redshift_wrapper, redshift_data_wrapper)
    redshift_scenario.redhift_scenario(
        f"{os.path.dirname(__file__)}/../../../resources/sample_files/
movies.json"
    )
```

As funções wrapper usadas no cenário.

```
    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """

        try:
```

```
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def execute_statement(
        self, cluster_identifier, database_name, user_name, sql,
parameter_list=None
    ):
        """
        Executes a SQL statement.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
        :param user_name: The user's name.
        :param sql: The SQL statement.
        :param parameter_list: The optional SQL statement parameters.
        :return: The SQL statement result.
        """

        try:
            kwargs = {
                "ClusterIdentifier": cluster_identifier,
                "Database": database_name,
                "DbUser": user_name,
                "Sql": sql,
            }
            if parameter_list:
                kwargs["Parameters"] = parameter_list
            response = self.client.execute_statement(**kwargs)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't execute statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
```

```
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_statement_result(self, statement_id):
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
```

```
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_databases(self, cluster_identifier, database_name, database_user):
        """
        Lists databases in a cluster.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
        :param database_user: The database user.
        :return: The list of databases.
        """
        try:
            paginator = self.client.get_paginator("list_databases")
            databases = []
            for page in paginator.paginate(
                ClusterIdentifier=cluster_identifier,
                Database=database_name,
                DbUser=database_user,
            ):
                databases.extend(page["Databases"])

            return databases
        except ClientError as err:
            logging.error(
                "Couldn't list databases. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier,
SkipFinalClusterSnapshot=True
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir em AWS SDK for Python (Boto3) API Reference.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Ações para o Amazon Redshift usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Redshift com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Redshift e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Redshift usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para ver uma lista completa, consulte Amazon Redshift API Reference.

Exemplos

- Use CreateClustercom um AWS SDK ou CLI
- Use DeleteClustercom um AWS SDK ou CLI
- Use DescribeClusterscom um AWS SDK ou CLI
- Use DescribeStatementcom um AWS SDK
- Use ExecuteStatementcom um AWS SDK
- Use GetStatementResultcom um AWS SDK
- Use ListDatabasescom um AWS SDK
- Use ModifyClustercom um AWS SDK ou CLI

Use **`CreateCluster`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de clusters de data warehouse provisionados

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
    /// <param name="masterUsername">The master username.</param>
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier,
string databaseName,
        string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
    {
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create cluster. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateClustera Referência AWS SDK para .NET da API.

CLI

AWS CLI

O ParametersThis exemplo Criar um cluster com mínimo cria um cluster com o conjunto mínimo de parâmetros. Por padrão, o formato da saída é JSON. Comando:

```
aws redshift create-cluster --node-type dw.hs1.xlarge --number-of-nodes 2 --
master-username adminuser --master-user-password TopSecret1 --cluster-identifier
 mycluster
```

Resultado:

```
{
   "Cluster": {
      "NodeType": "dw.hs1.xlarge",
      "ClusterVersion": "1.0",
      "PubliclyAccessible": "true",
      "MasterUsername": "adminuser",
      "ClusterParameterGroups": [
         {
            "ParameterApplyStatus": "in-sync",
            "ParameterGroupName": "default.redshift-1.0"
         } ],
      "ClusterSecurityGroups": [
         {
            "Status": "active",
            "ClusterSecurityGroupName": "default"
         } ],
      "AllowVersionUpgrade": true,
      "VpcSecurityGroups": \[],
      "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
      "AutomatedSnapshotRetentionPeriod": 1,
      "ClusterStatus": "creating",
      "ClusterIdentifier": "mycluster",
      "DBName": "dev",
      "NumberOfNodes": 2,
```

```
        "PendingModifiedValues": {
           "MasterUserPassword": "\****"
        }
    },
    "ResponseMetadata": {
        "RequestId": "7cf4bcfc-64dd-11e2-bea9-49e0ce183f07"
    }
}
```

- Para obter detalhes da API, consulte CreateClusterem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}
```

```
// CreateCluster sends a request to create a cluster with the given clusterId
 using the provided credentials.
func (actor RedshiftActions) CreateCluster(ctx context.Context, clusterId string,
 userName string, userPassword string, nodeType string, clusterType string,
 publiclyAccessible bool) (*redshift.CreateClusterOutput, error) {
 // Create a new Redshift cluster
 input := &redshift.CreateClusterInput{
  ClusterIdentifier:  aws.String(clusterId),
  MasterUserPassword: aws.String(userPassword),
  MasterUsername:     aws.String(userName),
  NodeType:           aws.String(nodeType),
  ClusterType:        aws.String(clusterType),
  PubliclyAccessible: aws.Bool(publiclyAccessible),
 }
 var opErr *types.ClusterAlreadyExistsFault
 output, err := actor.RedshiftClient.CreateCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster already exists")
  return nil, nil
 } else if err != nil {
  log.Printf("Failed to create Redshift cluster: %v\n", err)
  return nil, err
 }

 log.Printf("Created cluster %s\n", *output.Cluster.ClusterIdentifier)
 return output, nil
}
```

- Para obter detalhes da API, consulte CreateClustera Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cluster.

```
   /**
    * Creates a new Amazon Redshift cluster asynchronously.
    * @param clusterId     the unique identifier for the cluster
    * @param username      the username for the administrative user
    * @param userPassword  the password for the administrative user
    * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
    * @throws RuntimeException if the cluster creation fails
    */
   public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
       CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .masterUsername(username)
           .masterUserPassword(userPassword)
           .nodeType("ra3.4xlarge")
           .publiclyAccessible(true)
           .numberOfNodes(2)
           .build();

       return getAsyncClient().createCluster(clusterRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Created cluster ");
               } else {
                   throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
               }
           });
   }
```

- Para obter detalhes da API, consulte CreateClustera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Crie o cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME", // Required
  NodeType: "NODE_TYPE", //Required
  MasterUsername: "MASTER_USER_NAME", // Required - must be lowercase
  MasterUserPassword: "MASTER_USER_PASSWORD", // Required - must contain at least
 one uppercase letter, and one number
  ClusterType: "CLUSTER_TYPE", // Required
  IAMRoleARN: "IAM_ROLE_ARN", // Optional - the ARN of an IAM role with
 permissions your cluster needs to access other AWS services on your behalf, such
 as Amazon S3.
  ClusterSubnetGroupName: "CLUSTER_SUBNET_GROUPNAME", //Optional - the name of a
 cluster subnet group to be associated with this cluster. Defaults to 'default'
 if not specified.
  DBName: "DATABASE_NAME", // Optional - defaults to 'dev' if not specified
  Port: "PORT_NUMBER", // Optional - defaults to '5439' if not specified
};
```

```
const run = async () => {
  try {
    const data = await redshiftClient.send(new CreateClusterCommand(params));
    console.log(
      `Cluster ${data.Cluster.ClusterIdentifier} successfully created`,
    );
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateClustera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cluster.

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
```

```
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient.fromEnvironment { region = "us-east-1" }.use { redshiftClient
 ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- Para obter detalhes da API, consulte a CreateClusterreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
```

```
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """

        try:
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftWrapper objeto.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Para obter detalhes da API, consulte a CreateClusterReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cluster.

```
TRY.
    " Example values: iv_cluster_identifier = 'my-redshift-cluster'
    " Example values: iv_node_type = 'ra3.4xlarge'
    " Example values: iv_master_username = 'awsuser'
    " Example values: iv_master_password = 'AwsUser1000'
    " Example values: iv_publicly_accessible = abap_true
    " Example values: iv_number_of_nodes = 2
    oo_result = lo_rsh->createcluster(
      iv_clusteridentifier = iv_cluster_identifier
      iv_nodetype = iv_node_type
      iv_masterusername = iv_master_username
      iv_masteruserpassword = iv_master_password
      iv_publiclyaccessible = iv_publicly_accessible
      iv_numberofnodes = iv_number_of_nodes
    ).
    MESSAGE 'Redshift cluster created successfully.' TYPE 'I'.
  CATCH /aws1/cx_rshclustalrdyexfault.
    MESSAGE 'Cluster already exists.' TYPE 'I'.
  CATCH /aws1/cx_rshclstquotaexcdfault.
    MESSAGE 'Cluster quota exceeded.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateClusterreferência da API AWS SDK for SAP ABAP.

Use **`DeleteCluster`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de clusters de data warehouse provisionados

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an Amazon Redshift cluster without a final snapshot.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
clusterIdentifier)
    {
        try
        {
            var request = new DeleteClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                SkipFinalClusterSnapshot = true
            };

            var response = await _redshiftClient.DeleteClusterAsync(request);
            Console.WriteLine($"The {clusterIdentifier} was deleted");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"Couldn't delete cluster. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteClustera Referência AWS SDK para .NET da API.

CLI

AWS CLI

O SnapshotThis exemplo Excluir um cluster sem cluster final exclui um cluster, forçando a exclusão de dados para que nenhum instantâneo final do cluster seja criado.Comando:

```
aws redshift delete-cluster --cluster-identifier mycluster --skip-final-cluster-
snapshot
```

Excluir um cluster, Permitindo um SnapshotThis exemplo de cluster final exclui um cluster, mas especifica um instantâneo final do cluster.Comando:

```
aws redshift delete-cluster --cluster-identifier mycluster --final-cluster-
snapshot-identifier myfinalsnapshot
```

- Para obter detalhes da API, consulte DeleteClusterem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// DeleteCluster deletes the given cluster.
func (actor RedshiftActions) DeleteCluster(ctx context.Context, clusterId string)
 (bool, error) {
 input := redshift.DeleteClusterInput{
  ClusterIdentifier:        aws.String(clusterId),
  SkipFinalClusterSnapshot: aws.Bool(true),
 }
 _, err := actor.RedshiftClient.DeleteCluster(ctx, &input)
 var opErr *types.ClusterNotFoundFault
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster was not found. Where could it be?")
  return false, err
 } else if err != nil {
  log.Printf("Failed to delete Redshift cluster: %v\n", err)
  return false, err
 }
 waiter := redshift.NewClusterDeletedWaiter(actor.RedshiftClient)
 err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 }, 5*time.Minute)
 if err != nil {
  log.Printf("Wait time exceeded for deleting cluster, continuing: %v\n", err)
```

```
 }
 log.Printf("The cluster %s was deleted\n", clusterId)
 return true, nil
}
```

- Para obter detalhes da API, consulte DeleteCluster a Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir o cluster.

```
   /**
    * Deletes a Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the Redshift cluster to be deleted
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
    */
   public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
       DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .skipFinalClusterSnapshot(true)
           .build();

       return getAsyncClient().deleteCluster(deleteClusterRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   // Handle exceptions
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {}", exception.getMessage());
```

```
                    } else {
                        logger.info("Unexpected error: {}",
exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
response.cluster().clusterStatus());
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteClustera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Crie o cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";
```

```
const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  SkipFinalClusterSnapshot: false,
  FinalClusterSnapshotIdentifier: "CLUSTER_SNAPSHOT_ID",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DeleteClusterCommand(params));
    console.log("Success, cluster deleted. ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte DeleteCluster a Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir o cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }
```

```
    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient
 ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- Para obter detalhes da API, consulte a DeleteClusterreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier,
 SkipFinalClusterSnapshot=True
```

```
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftWrapper objeto.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Para obter detalhes da API, consulte a DeleteClusterReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir o cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-cluster'
        lo_rsh->deletecluster(
          iv_clusteridentifier = iv_cluster_identifier
          iv_skipfinalclustersnapshot = abap_true
        ).
        MESSAGE 'Redshift cluster deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rshclustnotfoundfault.
```

```
        MESSAGE 'Cluster not found.' TYPE 'I'.
      CATCH /aws1/cx_rshinvcluststatefault.
        MESSAGE 'Invalid cluster state for deletion.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteClusterreferência da API AWS SDK for SAP ABAP.

Use **`DescribeClusters`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeClusters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de clusters de data warehouse provisionados

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe Amazon Redshift clusters.
    /// </summary>
    /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
    /// <returns>A list of clusters.</returns>
    public async Task<List<Cluster>> DescribeClustersAsync(string?
clusterIdentifier = null)
    {
        try
        {
```

```
            var clusters = new List<Cluster>();
            var request = new DescribeClustersRequest();
            if (!string.IsNullOrEmpty(clusterIdentifier))
            {
                request.ClusterIdentifier = clusterIdentifier;
            }

            var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeClustersa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O ClustersThis exemplo Obter uma descrição de tudo retorna uma descrição de todos os clusters da conta. Por padrão, o formato da saída é JSON. Comando:

```
aws redshift describe-clusters
```

Resultado:

```
{
   "Clusters": [
   {
      "NodeType": "dw.hs1.xlarge",
      "Endpoint": {
         "Port": 5439,
         "Address": "mycluster.coqoarplqhsn.us-east-1.redshift.amazonaws.com"
      },
      "ClusterVersion": "1.0",
      "PubliclyAccessible": "true",
      "MasterUsername": "adminuser",
      "ClusterParameterGroups": [
         {
            "ParameterApplyStatus": "in-sync",
            "ParameterGroupName": "default.redshift-1.0"
         } ],
      "ClusterSecurityGroups": [
         {
            "Status": "active",
            "ClusterSecurityGroupName": "default"
         } ],
      "AllowVersionUpgrade": true,
      "VpcSecurityGroups": \[],
      "AvailabilityZone": "us-east-1a",
      "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
      "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
      "AutomatedSnapshotRetentionPeriod": 1,
      "ClusterStatus": "available",
      "ClusterIdentifier": "mycluster",
      "DBName": "dev",
      "NumberOfNodes": 2,
      "PendingModifiedValues": {}
```

```
    } ],
    "ResponseMetadata": {
       "RequestId": "65b71cac-64df-11e2-8f5b-e90bd6c77476"
    }
}
```

Você pode obter as mesmas informações em formato de texto com a opção `--output text`. Comando:

opção `--output text`. Comando:

opção. Comando:

```
aws redshift describe-clusters --output text
```

Resultado:

```
dw.hs1.xlarge       1.0     true    adminuser       True    us-east-1a
 2013-01-22T21:59:29.559Z        sat:03:30-sat:04:00     1       available
 mycluster       dev     2
ENDPOINT    5439    mycluster.coqoarplqhsn.us-east-1.redshift.amazonaws.com
in-sync     default.redshift-1.0
active      default
PENDINGMODIFIEDVALUES
RESPONSEMETADATA    934281a8-64df-11e2-b07c-f7fbdd006c67
```

- Para obter detalhes da API, consulte DescribeClustersem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// DescribeClusters returns information about the given cluster.
func (actor RedshiftActions) DescribeClusters(ctx context.Context, clusterId
 string) (*redshift.DescribeClustersOutput, error) {
 input, err := actor.RedshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 })
 var opErr *types.AccessToClusterDeniedFault
 if errors.As(err, &opErr) {
  println("Access to cluster denied.")
  panic(err)
 } else if err != nil {
  println("Failed to describe Redshift clusters.")
  return nil, err
 }
 return input, nil
}
```

- Para obter detalhes da API, consulte DescribeClustersa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Descrever o cluster.

```
   /**
    * Waits asynchronously for the specified cluster to become available.
    * @param clusterId the identifier of the cluster to wait for
    * @return a {@link CompletableFuture} that completes when the cluster is
ready
    */
   public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
       DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
           .clusterIdentifier(clusterId)
           .build();

       logger.info("Waiting for cluster to become available. This may take a few
minutes.");
       long startTime = System.currentTimeMillis();

       // Recursive method to poll the cluster status.
       return checkClusterStatusAsync(clustersRequest, startTime);
   }

   private CompletableFuture<Void>
checkClusterStatusAsync(DescribeClustersRequest clustersRequest, long startTime)
{
       return getAsyncClient().describeClusters(clustersRequest)
           .thenCompose(clusterResponse -> {
               List<Cluster> clusterList = clusterResponse.clusters();
               boolean clusterReady = false;
               for (Cluster cluster : clusterList) {
                   if ("available".equals(cluster.clusterStatus())) {
                       clusterReady = true;
                       break;
```

```
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }
```

- Para obter detalhes da API, consulte DescribeClustersa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Descreva os clusters.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeClustersCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DescribeClustersCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte DescribeClustersa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Descrever o cluster.

```
suspend fun describeRedshiftClusters() {
    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient
 ->
        val clusterResponse =
 redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeClustersreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
```

```
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftWrapper objeto.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Para obter detalhes da API, consulte a DescribeClustersReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Descrever o cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-
cluster' (optional)
        oo_result = lo_rsh->describeclusters(
          iv_clusteridentifier = iv_cluster_identifier
        ).
        lt_clusters = oo_result->get_clusters( ).
        lv_cluster_count = lines( lt_clusters ).
        MESSAGE |Retrieved { lv_cluster_count } cluster(s).| TYPE 'I'.
      CATCH /aws1/cx_rshclustnotfoundfault.
```

```
        MESSAGE 'Cluster not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeClustersreferência da API AWS SDK for SAP ABAP.

Use **`DescribeStatement`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeStatement`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Describe a statement execution.
   /// </summary>
   /// <param name="statementId">The statement ID.</param>
   /// <returns>The statement description.</returns>
   public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
   {
       try
       {
           var request = new DescribeStatementRequest
           {
               Id = statementId
           };
```

```
            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeStatementa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL
statement's status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
```

```
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending
program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }
```

- Para obter detalhes da API, consulte DescribeStatementa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftDataWrapper objeto.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- Para obter detalhes da API, consulte a DescribeStatementReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx'
        oo_result = lo_rsd->describestatement(
          iv_id = iv_statement_id
        ).
        lv_status = oo_result->get_status( ).
        MESSAGE |Statement status: { lv_status }| TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Statement not found.' TYPE 'I'.
      CATCH /aws1/cx_rsdinternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeStatementreferência da API AWS SDK for SAP ABAP.

Use **`ExecuteStatement`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ExecuteStatement`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Executa uma instrução SQL para criar uma tabela de banco de dados.

```
   /**
    * Creates an asynchronous task to execute a SQL statement for creating a new
table.
    *
    * @param clusterId    the identifier of the Amazon Redshift cluster
    * @param databaseName the name of the database to create the table in
    * @param userName     the username to use for the database connection
    * @return a {@link CompletableFuture} that completes with the result of the
SQL statement execution
    * @throws RuntimeException if there is an error creating the table
    */
   public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
       ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(userName)
           .database(databaseName)
           .sql("CREATE TABLE Movies (" +
               "id INT PRIMARY KEY, " +
               "title VARCHAR(100), " +
               "year INT)")
           .build();

       return getAsyncDataClient().executeStatement(createTableRequest)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }
```

Executa uma instrução SQL para inserir dados em uma tabela de banco de dados.

```
    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records
added to the Movies table
     */
    public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
        return CompletableFuture.supplyAsync(() -> {
                try {
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON
file: " + e.getMessage(), e);
                }
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
```

```
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
    }

    private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
        return CompletableFuture.supplyAsync(() -> {
            int t = 0;
            try {
                while (iter.hasNext()) {
                    if (t == number)
                        break;
                    JsonNode currentNode = iter.next();
                    int year = currentNode.get("year").asInt();
                    String title = currentNode.get("title").asText();

                    // Use SqlParameter to avoid SQL injection.
                    List<SqlParameter> parameterList = new ArrayList<>();
                    String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                    SqlParameter idParam = SqlParameter.builder()
                        .name("id")
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();

                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
```

```
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();


getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }
```

Executa uma instrução SQL para consultar uma tabela de banco de dados.

```
    /**
     * Asynchronously queries movies by a given year from a Redshift database.
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String
clusterId) {

        String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
```

```
        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

- Para obter detalhes da API, consulte ExecuteStatementa Referência AWS SDK for Java 2.x da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
```

```
        " Example values: iv_database_name = 'dev'
        " Example values: iv_user_name = 'awsuser'
        " Example values: iv_sql = 'SELECT * FROM movies WHERE year = :year'
        " Example values: it_parameter_list - SQL parameters for parameterized
queries

        " Only pass parameters if the list is not empty
        IF it_parameter_list IS NOT INITIAL.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
            it_parameters = it_parameter_list
          ).
        ELSE.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
          ).
        ENDIF.

        lv_statement_id = oo_result->get_id( ).
        MESSAGE |Statement executed. ID: { lv_statement_id }| TYPE 'I'.
      CATCH /aws1/cx_rsdexecutestatementex.
        MESSAGE 'Statement execution error.' TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ExecuteStatementreferência da API AWS SDK for SAP ABAP.

Use **`GetStatementResult`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetStatementResult`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Get the results of a statement execution.
   /// </summary>
   /// <param name="statementId">The statement ID.</param>
   /// <returns>A list of result rows.</returns>
   public async Task<List<List<Field>>> GetStatementResultAsync(string
statementId)
   {
       try
       {
           var request = new GetStatementResultRequest
           {
               Id = statementId
           };

           var response = await
_redshiftDataClient.GetStatementResultAsync(request);
           return response.Records;
       }
       catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
       {
           Console.WriteLine($"Statement not found: {ex.Message}");
           throw;
       }
       catch (Exception ex)
       {
           Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
           throw;
       }
```

```
    }
```

- Para obter detalhes da API, consulte GetStatementResulta Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Verifique o resultado da instrução

```
   /**
    * Asynchronously retrieves the results of a statement execution.
    *
    * @param statementId the ID of the statement for which to retrieve the
results
    * @return a {@link CompletableFuture} that completes when the statement
result has been processed
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: "
+ exception.getMessage(), exception);
               }
```

```
                // Extract and print the field values using streams if the
response is valid.
                response.records().stream()
                    .flatMap(List::stream)
                    .map(Field::stringValue)
                    .filter(value -> value != null)
                    .forEach(value -> System.out.println("The Movie title field
is " + value));

                return response;
            }).thenAccept(response -> {
                // Optionally add more logic here if needed after handling the
response
            });
    }
```

- Para obter detalhes da API, consulte GetStatementResulta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def get_statement_result(self, statement_id):
```

```
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftDataWrapper objeto.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- Para obter detalhes da API, consulte a GetStatementResultReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Verifique o resultado da instrução

```
    TRY.
        " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx'
        " Handle pagination for large result sets

        DO.
          lo_result_page = lo_rsd->getstatementresult(
            iv_id = iv_statement_id
            iv_nexttoken = lv_next_token
          ).

          " Collect records from this page
          lt_page_records = lo_result_page->get_records( ).
          APPEND LINES OF lt_page_records TO lt_all_records.

          " Check if there are more pages
          lv_next_token = lo_result_page->get_nexttoken( ).
          IF lv_next_token IS INITIAL.
            EXIT. " No more pages
          ENDIF.
        ENDDO.

        " For the last call, set oo_result for return value
        oo_result = lo_result_page.
        lv_record_count = lines( lt_all_records ).
        MESSAGE |Retrieved { lv_record_count } record(s).| TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Statement not found or results not available.' TYPE 'I'.
      CATCH /aws1/cx_rsdinternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetStatementResultreferência da API AWS SDK for SAP ABAP.

Use **`ListDatabases`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListDatabases`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();
```

```
            foreach (var database in response.Databases)
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListDatabasesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all databases asynchronously for the specified cluster, database
user, and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
```

```
    * @return a {@link CompletableFuture} that completes when the database
listing is complete, or throws a {@link RuntimeException} if there was an error
    */
   public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
       ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(dbUser)
           .database(database)
           .build();

       // Asynchronous paginator for listing databases.
       ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
       CompletableFuture<Void> future = databasesPaginator.subscribe(response ->
{
           response.databases().forEach(db -> {
               logger.info("The database name is {} ", db);
           });
       });

       // Return the future for asynchronous handling.
       return future.exceptionally(exception -> {
           throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
       });
   }
```

- Para obter detalhes da API, consulte ListDatabasesa Referência AWS SDK for Java 2.x da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
        " Example values: iv_database_name = 'dev'
        " Example values: iv_database_user = 'awsuser'
        oo_result = lo_rsd->listdatabases(
          iv_clusteridentifier = iv_cluster_identifier
          iv_database = iv_database_name
          iv_dbuser = iv_database_user
        ).
        lt_databases = oo_result->get_databases( ).
        lv_db_count = lines( lt_databases ).
        MESSAGE |Retrieved { lv_db_count } database(s).| TYPE 'I'.
      CATCH /aws1/cx_rsddatabaseconnex.
        MESSAGE 'Database connection error.' TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Cluster not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListDatabasesreferência da API AWS SDK for SAP ABAP.

Use **`ModifyCluster`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ModifyCluster`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance
window.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
preferredMaintenanceWindow)
    {
        try
        {
            var request = new ModifyClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                PreferredMaintenanceWindow = preferredMaintenanceWindow
            };

            var response = await _redshiftClient.ModifyClusterAsync(request);
            Console.WriteLine($"The modified cluster was successfully modified
and has {response.Cluster.PreferredMaintenanceWindow} as the maintenance
window");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't modify cluster. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte ModifyClustera Referência AWS SDK para .NET da API.

CLI

AWS CLI

Associar um grupo de segurança a um ClusterThis exemplo mostra como associar um grupo de segurança de cluster ao cluster especificado.Comando:

```
aws redshift modify-cluster --cluster-identifier mycluster --cluster-security-
groups mysecuritygroup
```

Modificar a janela de manutenção para a ClusterThis mostra como alterar a janela de manutenção semanal preferencial de um cluster para a janela mínima de quatro horas, começando aos domingos às 23h15 e terminando às segundas-feiras às 15h15: AM.Command

```
aws redshift modify-cluster --cluster-identifier mycluster --preferred-
maintenance-window Sun:23:15-Mon:03:15
```

Alterar a senha mestra do ClusterThis exemplo mostra como alterar a senha mestra de um cluster.Comando:

```
aws redshift modify-cluster --cluster-identifier mycluster --master-user-password
 A1b2c3d4
```

- Para obter detalhes da API, consulte ModifyClusterem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
```

```
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// ModifyCluster sets the preferred maintenance window for the given cluster.
func (actor RedshiftActions) ModifyCluster(ctx context.Context, clusterId string,
 maintenanceWindow string) *redshift.ModifyClusterOutput {
 // Modify the cluster's maintenance window
 input := &redshift.ModifyClusterInput{
  ClusterIdentifier:          aws.String(clusterId),
  PreferredMaintenanceWindow: aws.String(maintenanceWindow),
 }

 var opErr *types.InvalidClusterStateFault
 output, err := actor.RedshiftClient.ModifyCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster is in an invalid state.")
  panic(err)
 } else if err != nil {
  log.Printf("Failed to modify Redshift cluster: %v\n", err)
  panic(err)
 }

 log.Printf("The cluster was successfully modified and now has %s as the
 maintenance window\n", *output.Cluster.PreferredMaintenanceWindow)
 return output
}
```

- Para obter detalhes da API, consulte ModifyClustera Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Modificar um cluster.

```
   /**
    * Modifies an Amazon Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the cluster to be modified
    * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
    */
   public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
       ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .preferredMaintenanceWindow("wed:07:30-wed:08:00")
           .build();

       return getAsyncClient().modifyCluster(modifyClusterRequest)
           .whenComplete((clusterResponse, exception) -> {
               if (exception != null) {
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {} ", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {} ",
exception.getMessage());
                   }
               } else {
                   logger.info("The modified cluster was successfully modified
and has "
```

```
                        + clusterResponse.cluster().preferredMaintenanceWindow()
 + " as the maintenance window");
                }
            });
    }
```

- Para obter detalhes da API, consulte ModifyClustera Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Modificar um cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { ModifyClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

// Set the parameters
const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  MasterUserPassword: "NEW_MASTER_USER_PASSWORD",
};
```

```
const run = async () => {
  try {
    const data = await redshiftClient.send(new ModifyClusterCommand(params));
    console.log("Success was modified.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte ModifyClustera Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Modificar um cluster.

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
            "The modified cluster was successfully modified and has
 ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance
 window",
```

```
        )
    }
}
```

- Para obter detalhes da API, consulte a ModifyClusterreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
```

```
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftWrapper objeto.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Para obter detalhes da API, consulte a ModifyClusterReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Modificar um cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-cluster'
        " Example values: iv_pref_maintenance_wn = 'wed:07:30-wed:08:00'
        lo_rsh->modifycluster(
          iv_clusteridentifier = iv_cluster_identifier
          iv_preferredmaintenancewin00 = iv_pref_maintenance_wn
        ).
        MESSAGE 'Redshift cluster modified successfully.' TYPE 'I'.
      CATCH /aws1/cx_rshclustnotfoundfault.
        MESSAGE 'Cluster not found.' TYPE 'I'.
      CATCH /aws1/cx_rshinvcluststatefault.
```

```
        MESSAGE 'Invalid cluster state for modification.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ModifyClusterreferência da API AWS SDK for SAP ABAP.

Cenários para o Amazon Redshift usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Redshift com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon Redshift ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar um rastreador de itens do Amazon Redshift
- Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI
- Conceitos básicos de clusters de data warehouse provisionados

Criar um rastreador de itens do Amazon Redshift

Os exemplos de código a seguir mostram como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho usando um banco de dados do Amazon Redshift.

Java

SDK para Java 2.x

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- banco de dados de origem
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- banco de dados de origem
- Amazon SES

Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI

O exemplo de código a seguir mostra como:

- Use o gerenciador de segredos CreateSecret
- Use o gerenciador de segredos DeleteSecret
- Use o gerenciador de segredos GetSecretValue
- Use redshift CreateNamespace
- Use redshift CreateWorkgroup
- Use redshift DeleteNamespace
- Use iam CreateRole

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq
 dependency)
# This script creates a Redshift Serverless environment, loads sample data, and
 runs queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
```

```
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}
```

```
# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}
```

```
# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status:
 $status. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be
 available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
```

```
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "============================================"
  echo "CLEANUP CONFIRMATION"
  echo "============================================"
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --
workgroup-name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
```

```
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --
namespace-name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
```

```
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin
 credentials for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"
```

```
# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless
 get-namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless
 get-workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
```

```
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5
```

```
# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
```

```
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
```

```
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
```

```
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID"
 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi
```

```
# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time
 total sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price
 desc) as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
```

```
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID"
 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
```

```
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole
  - GetSecretValue
  - GetWorkgroup
  - PutRolePolicy
  - UpdateNamespace

Conceitos básicos de clusters de data warehouse provisionados

O exemplo de código a seguir mostra como:

- Criar um cluster do Redshift.
- Criar um perfil do IAM para acessar o S3.
- Criar tabelas e carregar dados.

- Executar consultas de exemplo.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-
final-cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/
dev/null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""

    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
```

```
        status=$(aws redshift-data describe-statement --id "$statement_id" --
query 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id
 "$statement_id" --query 'Error' --output text 2>/dev/null || echo "Unknown
 error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus"
 \
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
```

```
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and
 try again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi
```

```
# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
```

```
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed
 to attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"
```

```
echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer
 not null, eventid integer not null, dateid smallint not null sortkey, qtysold
 smallint not null, pricepaid decimal(8,2), commission decimal(8,2), saletime
 timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint
 not null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi

echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
```

```
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';"
 \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt'
 DELIMITER '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
```

```
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid
 AND caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

Exemplos de código para o Amazon Rekognition usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Rekognition AWS com um kit de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Rekognition: mais informações sobre o Amazon Rekognition.
- Referência da API Amazon Rekognition: detalhes sobre todas as ações disponíveis do Amazon Rekognition.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Amazon Rekognition usando AWS SDKs
  - Olá, Amazon Rekognition
  - Ações para o Amazon Rekognition usando AWS SDKs
    - Use CompareFacescom um AWS SDK ou CLI
    - Use CreateCollectioncom um AWS SDK ou CLI
    - Use DeleteCollectioncom um AWS SDK ou CLI
    - Use DeleteFacescom um AWS SDK ou CLI
    - Use DescribeCollectioncom um AWS SDK ou CLI
    - Use DetectFacescom um AWS SDK ou CLI
    - Use DetectLabelscom um AWS SDK ou CLI
    - Use DetectModerationLabelscom um AWS SDK ou CLI
    - Use DetectTextcom um AWS SDK ou CLI
    - Use GetCelebrityInfocom um AWS SDK ou CLI
    - Use IndexFacescom um AWS SDK ou CLI
    - Use ListCollectionscom um AWS SDK ou CLI
    - Use ListFacescom um AWS SDK ou CLI
    - Use RecognizeCelebritiescom um AWS SDK ou CLI
    - Use SearchFacescom um AWS SDK ou CLI
    - Use SearchFacesByImagecom um AWS SDK ou CLI
- Cenários para o Amazon Rekognition usando AWS SDKs
  - Crie uma coleção do Amazon Rekognition e encontre rostos nela usando um AWS SDK
  - Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
  - Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK
  - Detecte e exiba elementos em imagens com o Amazon Rekognition usando um AWS SDK
  - Detecte rostos em uma imagem usando um AWS SDK
  - Detecte informações em vídeos usando o Amazon Rekognition e o AWS SDK
  - Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK
  - Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
  - Salve EXIF e outras informações da imagem usando um AWS SDK

Exemplos básicos do Amazon Rekognition usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Rekognition com AWS SDKs.

Exemplos

- Olá, Amazon Rekognition
- Ações para o Amazon Rekognition usando AWS SDKs
  - Use CompareFacescom um AWS SDK ou CLI
  - Use CreateCollectioncom um AWS SDK ou CLI
  - Use DeleteCollectioncom um AWS SDK ou CLI
  - Use DeleteFacescom um AWS SDK ou CLI
  - Use DescribeCollectioncom um AWS SDK ou CLI
  - Use DetectFacescom um AWS SDK ou CLI
  - Use DetectLabelscom um AWS SDK ou CLI
  - Use DetectModerationLabelscom um AWS SDK ou CLI
  - Use DetectTextcom um AWS SDK ou CLI
  - Use GetCelebrityInfocom um AWS SDK ou CLI
  - Use IndexFacescom um AWS SDK ou CLI
  - Use ListCollectionscom um AWS SDK ou CLI
  - Use ListFacescom um AWS SDK ou CLI
  - Use RecognizeCelebritiescom um AWS SDK ou CLI
  - Use SearchFacescom um AWS SDK ou CLI
  - Use SearchFacesByImagecom um AWS SDK ou CLI

Olá, Amazon Rekognition

O exemplo de código a seguir mostra como começar a usar o Amazon Rekognition.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_rekognition.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rekognition/RekognitionClient.h>
#include <aws/rekognition/model/ListCollectionsRequest.h>
#include <iostream>

/*
 *  A "Hello Rekognition" starter application which initializes an Amazon
 Rekognition client and
 *  lists the Amazon Rekognition collections in the current account and region.
 *
 *  main function
 *
 *  Usage: 'hello_rekognition'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
        Aws::Rekognition::RekognitionClient rekognitionClient(clientConfig);
        Aws::Rekognition::Model::ListCollectionsRequest request;
        Aws::Rekognition::Model::ListCollectionsOutcome outcome =
                rekognitionClient.ListCollections(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String>& collectionsIds =
 outcome.GetResult().GetCollectionIds();
            if (!collectionsIds.empty()) {
                std::cout << "collectionsIds: " << std::endl;
                for (auto &collectionId : collectionsIds) {
                    std::cout << "- " << collectionId << std::endl;
                }
            } else {
                std::cout << "No collections found" << std::endl;
            }
        } else {
            std::cerr << "Error with ListCollections: " << outcome.GetError()
                      << std::endl;
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListCollectionsa Referência AWS SDK para C++ da API.

Ações para o Amazon Rekognition usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Rekognition com SDKs. AWS Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Rekognition e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Rekognition usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Rekognition.

Exemplos

- Use CompareFacescom um AWS SDK ou CLI
- Use CreateCollectioncom um AWS SDK ou CLI
- Use DeleteCollectioncom um AWS SDK ou CLI
- Use DeleteFacescom um AWS SDK ou CLI
- Use DescribeCollectioncom um AWS SDK ou CLI
- Use DetectFacescom um AWS SDK ou CLI
- Use DetectLabelscom um AWS SDK ou CLI
- Use DetectModerationLabelscom um AWS SDK ou CLI
- Use DetectTextcom um AWS SDK ou CLI
- Use GetCelebrityInfocom um AWS SDK ou CLI
- Use IndexFacescom um AWS SDK ou CLI
- Use ListCollectionscom um AWS SDK ou CLI
- Use ListFacescom um AWS SDK ou CLI
- Use RecognizeCelebritiescom um AWS SDK ou CLI
- Use SearchFacescom um AWS SDK ou CLI
- Use SearchFacesByImagecom um AWS SDK ou CLI

Use **`CompareFaces`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CompareFaces`.

Para obter mais informações, consulte Comparação de faces em imagens.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to compare faces in two images.
    /// </summary>
    public class CompareFaces
    {
        public static async Task Main()
        {
            float similarityThreshold = 70F;
            string sourceImage = "source.jpg";
            string targetImage = "target.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            Amazon.Rekognition.Model.Image imageSource = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageSource.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine($"Failed to load source image: {sourceImage}");
                return;
            }

            Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read);
```

```
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Failed to load target image: {targetImage}");
                Console.WriteLine(ex.Message);
                return;
            }

            var compareFacesRequest = new CompareFacesRequest
            {
                SourceImage = imageSource,
                TargetImage = imageTarget,
                SimilarityThreshold = similarityThreshold,
            };

            // Call operation
            var compareFacesResponse = await
rekognitionClient.CompareFacesAsync(compareFacesRequest);

            // Display results
            compareFacesResponse.FaceMatches.ForEach(match =>
            {
                ComparedFace face = match.Face;
                BoundingBox position = face.BoundingBox;
                Console.WriteLine($"Face at {position.Left} {position.Top}
matches with {match.Similarity}% confidence.");
            });

            Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
face(s) that did not match.");
        }
    }
```

- Para obter detalhes da API, consulte CompareFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como comparar faces em duas imagens

O comando `compare-faces` a seguir compara faces em duas imagens armazenadas em um bucket do Amazon S3.

```
aws rekognition compare-faces \
    --source-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \
    --target-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

Saída:

```
{
    "UnmatchedFaces": [],
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.12368916720151901,
                    "Top": 0.16007372736930847,
                    "Left": 0.5901257991790771,
                    "Height": 0.25140416622161865
                },
                "Confidence": 100.0,
                "Pose": {
                    "Yaw": -3.7351467609405518,
                    "Roll": -0.10309021919965744,
                    "Pitch": 0.8637830018997192
                },
                "Quality": {
                    "Sharpness": 95.51618957519531,
                    "Brightness": 65.29893493652344
                },
                "Landmarks": [
                    {
                        "Y": 0.26721030473709106,
                        "X": 0.6204193830490112,
                        "Type": "eyeLeft"
```

```
                    },
                    {
                        "Y": 0.26831310987472534,
                        "X": 0.6776827573776245,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.3514654338359833,
                        "X": 0.6241428852081299,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.35258132219314575,
                        "X": 0.6713621020317078,
                        "Type": "mouthRight"
                    },
                    {
                        "Y": 0.3140771687030792,
                        "X": 0.6428444981575012,
                        "Type": "nose"
                    }
                ]
            },
            "Similarity": 100.0
        }
    ],
    "SourceImageFace": {
        "BoundingBox": {
            "Width": 0.12368916720151901,
            "Top": 0.16007372736930847,
            "Left": 0.5901257991790771,
            "Height": 0.25140416622161865
        },
        "Confidence": 100.0
    }
}
```

Para obter mais informações, consulte Compare rostos em imagens no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte CompareFacesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import software.amazon.awssdk.core.SdkBytes;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

   /**
    * Compares two faces from images stored in an Amazon S3 bucket using AWS
Rekognition.
    *
    * <p>This method takes two image keys from an S3 bucket and compares the
faces within them.
    * It prints out the confidence level of matched faces and reports the number
of unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the
S3 bucket.
    * @param targetKey   The object key (file path) for the target image in the
S3 bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
        try {
            Float similarityThreshold = 70F;
            S3Object s3ObjectSource = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceKey)
                    .build();

            Image sourceImage = Image.builder()
```

```
                    .s3Object(s3ObjectSource)
                    .build();

            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(targetKey)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                    .sourceImage(sourceImage)
                    .targetImage(targetImage)
                    .similarityThreshold(similarityThreshold)
                    .build();

            // Compare the two images.
            CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails =
compareFacesResult.faceMatches();

            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");
            }

            List<ComparedFace> unmatchedFaces =
compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte CompareFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }
```

```
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches
 with ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- Para obter detalhes da API, consulte a CompareFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
 greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
 have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
 response["FaceMatches"]
            ]
```

```
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches
```

- Para obter detalhes da API, consulte a CompareFacesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the source image
        DATA(lo_source_s3obj) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_source_s3_bucket
          iv_name = iv_source_s3_key ).

        " Create source image object
        DATA(lo_source_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_source_s3obj ).
```

```
      " Create S3 object reference for the target image
      DATA(lo_target_s3obj) = NEW /aws1/cl_reks3object(
        iv_bucket = iv_target_s3_bucket
        iv_name = iv_target_s3_key ).

      " Create target image object
      DATA(lo_target_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_target_s3obj ).

      " Compare faces
      oo_result = lo_rek->comparefaces(
        io_sourceimage = lo_source_image
        io_targetimage = lo_target_image
        iv_similaritythreshold = iv_similarity ).

      DATA(lt_face_matches) = oo_result->get_facematches( ).
      DATA(lt_unmatched_faces) = oo_result->get_unmatchedfaces( ).

      " Get counts of matched and unmatched faces
      DATA(lv_matched_count) = lines( lt_face_matches ).
      DATA(lv_unmatched_count) = lines( lt_unmatched_faces ).

      " Output detailed comparison results
      DATA(lv_message) = |Face comparison completed: | &&
                         |{ lv_matched_count } matched face(s), | &&
                         |{ lv_unmatched_count } unmatched face(s).|.
      MESSAGE lv_message TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CompareFacesreferência da API AWS SDK for SAP ABAP.

Use **`CreateCollection`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateCollection`.

Para obter mais informações, consulte Criar uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to create a collection to which you can add
/// faces using the IndexFaces operation.
/// </summary>
public class CreateCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        var createCollectionRequest = new CreateCollectionRequest
        {
            CollectionId = collectionId,
        };

        CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
        Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
        Console.WriteLine($"Status code :
{createCollectionResponse.StatusCode}");
    }
}
```

- Para obter detalhes da API, consulte CreateCollectiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar uma coleção

O comando `create-collection` a seguir cria uma coleção com o nome especificado.

```
aws rekognition create-collection \
    --collection-id "MyCollection"
```

Saída:

```
{
    "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0",
    "StatusCode": 200
}
```

Para obter mais informações, consulte Criar uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte CreateCollectionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = """

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
```

```
     * Creates a new Amazon Rekognition collection.
     *
     * @param rekClient    the Amazon Rekognition client used to interact with
 the Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateCollectiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- Para obter detalhes da API, consulte a CreateCollectionreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client
```

```
    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection
```

- Para obter detalhes da API, consulte a CreateCollectionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_rek->createcollection(
          iv_collectionid = iv_collection_id ).
        MESSAGE 'Collection created successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rekresrcalrdyexistsex.
        MESSAGE 'Collection already exists.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateCollectionreferência da API AWS SDK for SAP ABAP.

Use **`DeleteCollection`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteCollection`.

Para obter mais informações, consulte Excluir uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete an existing collection.
/// </summary>
public class DeleteCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
```

```
            Console.WriteLine("Deleting collection: " + collectionId);

            var deleteCollectionRequest = new DeleteCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var deleteCollectionResponse = await
 rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
            Console.WriteLine($"{collectionId}:
 {deleteCollectionResponse.StatusCode}");
        }
    }
```

- Para obter detalhes da API, consulte DeleteCollectiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir uma coleção

O comando `delete-collection` a seguir exclui a coleção especificada.

```
aws rekognition delete-collection \
    --collection-id MyCollection
```

Saída:

```
{
    "StatusCode": 200
}
```

Para obter mais informações, consulte Excluir uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DeleteCollectionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId>\s

            Where:
                collectionId - The id of the collection to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
```

```
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteCollectiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Para obter detalhes da API, consulte a DeleteCollectionreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
```

```
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
```

```
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise
```

- Para obter detalhes da API, consulte a DeleteCollectionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_rek->deletecollection(
          iv_collectionid = iv_collection_id ).
        MESSAGE 'Collection deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCollectionreferência da API AWS SDK for SAP ABAP.

Use **`DeleteFaces`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteFaces`.

Para obter mais informações, consulte Excluir faces de uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete one or more faces from
    /// a Rekognition collection.
    /// </summary>
    public class DeleteFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };

            var rekognitionClient = new AmazonRekognitionClient();

            var deleteFacesRequest = new DeleteFacesRequest()
            {
                CollectionId = collectionId,
                FaceIds = faces,
            };

            DeleteFacesResponse deleteFacesResponse = await
 rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
            deleteFacesResponse.DeletedFaces.ForEach(face =>
            {
                Console.WriteLine($"FaceID: {face}");
            });
```

```
        }
    }
```

- Para obter detalhes da API, consulte DeleteFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir faces de uma coleção

O comando `delete-faces` a seguir exclui uma face especificada de uma coleção.

```
aws rekognition delete-faces \
    --collection-id MyCollection
    --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'
```

Saída:

```
{
    "DeletedFaces": [
        "0040279c-0178-436e-b70a-e61b074e96b0"
    ]
}
```

Para obter mais informações, consulte Excluir faces de uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DeleteFacesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <faceId>\s

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
```

```
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should
 be deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Para obter detalhes da API, consulte a DeleteFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
```

```
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_faces(self, face_ids):
        """
        Deletes faces from the collection.

        :param face_ids: The list of IDs of faces to delete.
        :return: The list of IDs of faces that were deleted.
        """
        try:
```

```
            response = self.rekognition_client.delete_faces(
                CollectionId=self.collection_id, FaceIds=face_ids
            )
            deleted_ids = response["DeletedFaces"]
            logger.info(
                "Deleted %s faces from %s.", len(deleted_ids), self.collection_id
            )
        except ClientError:
            logger.exception("Couldn't delete faces from %s.",
self.collection_id)
            raise
        else:
            return deleted_ids
```

- Para obter detalhes da API, consulte a DeleteFacesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_rek->deletefaces(
          iv_collectionid = iv_collection_id
          it_faceids = it_face_ids ).

        DATA(lt_deleted_faces) = oo_result->get_deletedfaces( ).
        DATA(lv_deleted_count) = lines( lt_deleted_faces ).
        DATA(lv_msg6) = |{ lv_deleted_count } face(s) deleted successfully.|.
        MESSAGE lv_msg6 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
```

```
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteFacesreferência da API AWS SDK for SAP ABAP.

Use **`DescribeCollection`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeCollection`.

Para obter mais informações, consulte Descrever uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to describe the contents of a
/// collection.
/// </summary>
public class DescribeCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine($"Describing collection: {collectionId}");
```

```
            var describeCollectionRequest = new DescribeCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var describeCollectionResponse = await
 rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
            Console.WriteLine($"Collection ARN:
 {describeCollectionResponse.CollectionARN}");
            Console.WriteLine($"Face count:
 {describeCollectionResponse.FaceCount}");
            Console.WriteLine($"Face model version:
 {describeCollectionResponse.FaceModelVersion}");
            Console.WriteLine($"Created:
 {describeCollectionResponse.CreationTimestamp}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeCollectiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como descrever uma coleção

O exemplo de `describe-collection` a seguir exibe os detalhes da coleção especificada.

```
aws rekognition describe-collection \
    --collection-id MyCollection
```

Saída:

```
{
    "FaceCount": 200,
    "CreationTimestamp": 1569444828.274,
    "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0"
```

```
}
```

Para obter mais informações, consulte Descrever uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DescribeCollectionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
```

```
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeCollectiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeCollectionreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()
```

- Para obter detalhes da API, consulte a DescribeCollectionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_rek->describecollection(
         iv_collectionid = iv_collection_id ).
       DATA(lv_face_count) = oo_result->get_facecount( ).
       DATA(lv_msg) = |Collection described: { lv_face_count } face(s)
indexed.|.
       MESSAGE lv_msg TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeCollectionreferência da API AWS SDK for SAP ABAP.

Use **`DetectFaces`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectFaces`.

Para obter mais informações, consulte Detectar faces em uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectFaces
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },

            // Attributes can be "ALL" or "DEFAULT".
```

```
                // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and
 Quality.
                // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
                Attributes = new List<string>() { "ALL" },
            };

            try
            {
                DetectFacesResponse detectFacesResponse = await
 rekognitionClient.DetectFacesAsync(detectFacesRequest);
                bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
                foreach (FaceDetail face in detectFacesResponse.FaceDetails)
                {
                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"Confidence: {face.Confidence}");
                    Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
                    Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
 roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                    Console.WriteLine($"Brightness:
 {face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                    if (hasAll)
                    {
                        Console.WriteLine($"Estimated age is between
 {face.AgeRange.Low} and {face.AgeRange.High} years old.");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Exiba as informações da caixa delimitadora de todas as faces em uma imagem.

```
    using System;
```

```
    using System.Collections.Generic;
    using System.Drawing;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to display the details of the
    /// bounding boxes around the faces detected in an image.
    /// </summary>
    public class ImageOrientationBoundingBox
    {
        public static async Task Main()
        {
            string photo = @"D:\Development\AWS-Examples\Rekognition
\target.jpg"; // "photo.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
 FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            int height;
            int width;

            // Used to extract original photo width/height
            using (var imageBitmap = new Bitmap(photo))
            {
                height = imageBitmap.Height;
                width = imageBitmap.Width;
```

```
            }

            Console.WriteLine("Image Information:");
            Console.WriteLine(photo);
            Console.WriteLine("Image Height: " + height);
            Console.WriteLine("Image Width: " + width);

            try
            {
                var detectFacesRequest = new DetectFacesRequest()
                {
                    Image = image,
                    Attributes = new List<string>() { "ALL" },
                };

                DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
                detectFacesResponse.FaceDetails.ForEach(face =>
                {
                    Console.WriteLine("Face:");
                    ShowBoundingBoxPositions(
                        height,
                        width,
                        face.BoundingBox,
                        detectFacesResponse.OrientationCorrection);

                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
                    Console.WriteLine($"The detected face is estimated to be
between {face.AgeRange.Low} and {face.AgeRange.High} years old.\n");
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Display the bounding box information for an image.
        /// </summary>
        /// <param name="imageHeight">The height of the image.</param>
        /// <param name="imageWidth">The width of the image.</param>
```

```
        /// <param name="box">The bounding box for a face found within the
 image.</param>
        /// <param name="rotation">The rotation of the face's bounding box.</
param>
        public static void ShowBoundingBoxPositions(int imageHeight, int
 imageWidth, BoundingBox box, string rotation)
        {
            float left;
            float top;

            if (rotation == null)
            {
                Console.WriteLine("No estimated orientation. Check Exif data.");
                return;
            }

            // Calculate face position based on image orientation.
            switch (rotation)
            {
                case "ROTATE_0":
                    left = imageWidth * box.Left;
                    top = imageHeight * box.Top;
                    break;
                case "ROTATE_90":
                    left = imageHeight * (1 - (box.Top + box.Height));
                    top = imageWidth * box.Left;
                    break;
                case "ROTATE_180":
                    left = imageWidth - (imageWidth * (box.Left + box.Width));
                    top = imageHeight * (1 - (box.Top + box.Height));
                    break;
                case "ROTATE_270":
                    left = imageHeight * box.Top;
                    top = imageWidth * (1 - box.Left - box.Width);
                    break;
                default:
                    Console.WriteLine("No estimated orientation information.
 Check Exif data.");
                    return;
            }

            // Display face location information.
            Console.WriteLine($"Left: {left}");
            Console.WriteLine($"Top: {top}");
```

```
            Console.WriteLine($"Face Width: {imageWidth * box.Width}");
            Console.WriteLine($"Face Height: {imageHeight * box.Height}");
        }
    }
```

- Para obter detalhes da API, consulte DetectFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como detectar faces em uma imagem

O comando `detect-faces` a seguir detecta faces na imagem especificada armazenada em um bucket do Amazon S3.

```
aws rekognition detect-faces \
    --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
    --attributes "ALL"
```

Saída:

```
{
    "FaceDetails": [
        {
            "Confidence": 100.0,
            "Eyeglasses": {
                "Confidence": 98.91107940673828,
                "Value": false
            },
            "Sunglasses": {
                "Confidence": 99.7966537475586,
                "Value": false
            },
            "Gender": {
                "Confidence": 99.56611633300781,
                "Value": "Male"
            },
            "Landmarks": [
```

```
                {
                    "Y": 0.26721030473709106,
                    "X": 0.6204193830490112,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.26831310987472534,
                    "X": 0.6776827573776245,
                    "Type": "eyeRight"
                },
                {
                    "Y": 0.3514654338359833,
                    "X": 0.6241428852081299,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.35258132219314575,
                    "X": 0.6713621020317078,
                    "Type": "mouthRight"
                },
                {
                    "Y": 0.3140771687030792,
                    "X": 0.6428444981575012,
                    "Type": "nose"
                },
                {
                    "Y": 0.24662546813488007,
                    "X": 0.6001564860343933,
                    "Type": "leftEyeBrowLeft"
                },
                {
                    "Y": 0.24326619505882263,
                    "X": 0.6303644776344299,
                    "Type": "leftEyeBrowRight"
                },
                {
                    "Y": 0.23818562924861908,
                    "X": 0.6146903038024902,
                    "Type": "leftEyeBrowUp"
                },
                {
                    "Y": 0.24373626708984375,
                    "X": 0.6640064716339111,
                    "Type": "rightEyeBrowLeft"
```

```
                },
                {
                    "Y": 0.24877218902111053,
                    "X": 0.7025929093360901,
                    "Type": "rightEyeBrowRight"
                },
                {
                    "Y": 0.23938551545143127,
                    "X": 0.6823262572288513,
                    "Type": "rightEyeBrowUp"
                },
                {
                    "Y": 0.265746533870697,
                    "X": 0.6112898588180542,
                    "Type": "leftEyeLeft"
                },
                {
                    "Y": 0.2676128149032593,
                    "X": 0.6317071914672852,
                    "Type": "leftEyeRight"
                },
                {
                    "Y": 0.262735515832901,
                    "X": 0.6201658248901367,
                    "Type": "leftEyeUp"
                },
                {
                    "Y": 0.27025148272514343,
                    "X": 0.6206279993057251,
                    "Type": "leftEyeDown"
                },
                {
                    "Y": 0.268223375082016,
                    "X": 0.6658390760421753,
                    "Type": "rightEyeLeft"
                },
                {
                    "Y": 0.2672517001628876,
                    "X": 0.687832236289978,
                    "Type": "rightEyeRight"
                },
                {
                    "Y": 0.26383838057518005,
                    "X": 0.6769183874130249,
```

```
                "Type": "rightEyeUp"
            },
            {
                "Y": 0.27138751745224,
                "X": 0.676596462726593,
                "Type": "rightEyeDown"
            },
            {
                "Y": 0.32283174991607666,
                "X": 0.6350004076957703,
                "Type": "noseLeft"
            },
            {
                "Y": 0.3219289481639862,
                "X": 0.6567046642303467,
                "Type": "noseRight"
            },
            {
                "Y": 0.3420318365097046,
                "X": 0.6450609564781189,
                "Type": "mouthUp"
            },
            {
                "Y": 0.3664324879646301,
                "X": 0.6455618143081665,
                "Type": "mouthDown"
            },
            {
                "Y": 0.26721030473709106,
                "X": 0.6204193830490112,
                "Type": "leftPupil"
            },
            {
                "Y": 0.26831310987472534,
                "X": 0.6776827573776245,
                "Type": "rightPupil"
            },
            {
                "Y": 0.26343393325805664,
                "X": 0.5946047306060791,
                "Type": "upperJawlineLeft"
            },
            {
                "Y": 0.3543180525302887,
```

```
                "X": 0.6044883728027344,
                "Type": "midJawlineLeft"
            },
            {
                "Y": 0.4084877669811249,
                "X": 0.6477024555206299,
                "Type": "chinBottom"
            },
            {
                "Y": 0.3562754988670349,
                "X": 0.707981526851654,
                "Type": "midJawlineRight"
            },
            {
                "Y": 0.26580461859703064,
                "X": 0.7234612107276917,
                "Type": "upperJawlineRight"
            }
        ],
        "Pose": {
            "Yaw": -3.7351467609405518,
            "Roll": -0.10309021919965744,
            "Pitch": 0.8637830018997192
        },
        "Emotions": [
            {
                "Confidence": 8.74203109741211,
                "Type": "SURPRISED"
            },
            {
                "Confidence": 2.501944065093994,
                "Type": "ANGRY"
            },
            {
                "Confidence": 0.7378743290901184,
                "Type": "DISGUSTED"
            },
            {
                "Confidence": 3.5296201705932617,
                "Type": "HAPPY"
            },
            {
                "Confidence": 1.7162904739379883,
                "Type": "SAD"
```

```
                },
                {
                    "Confidence": 9.518536567687988,
                    "Type": "CONFUSED"
                },
                {
                    "Confidence": 0.45474427938461304,
                    "Type": "FEAR"
                },
                {
                    "Confidence": 72.79895782470703,
                    "Type": "CALM"
                }
            ],
            "AgeRange": {
                "High": 48,
                "Low": 32
            },
            "EyesOpen": {
                "Confidence": 98.93987274169922,
                "Value": true
            },
            "BoundingBox": {
                "Width": 0.12368916720151901,
                "Top": 0.16007372736930847,
                "Left": 0.5901257991790771,
                "Height": 0.25140416622161865
            },
            "Smile": {
                "Confidence": 93.4493179321289,
                "Value": false
            },
            "MouthOpen": {
                "Confidence": 90.53053283691406,
                "Value": false
            },
            "Quality": {
                "Sharpness": 95.51618957519531,
                "Brightness": 65.29893493652344
            },
            "Mustache": {
                "Confidence": 89.85221099853516,
                "Value": false
            },
```

```
            "Beard": {
                "Confidence": 86.1991195678711,
                "Value": true
            }
        }
    ]
}
```

Para obter mais informações, consulte Detectar faces em uma imagem no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DetectFacesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = """
```

```
            Usage:   <bucketName> <sourceImage>

            Where:
                bucketName = The name of the Amazon S3 bucket where the source
image is stored.
                sourceImage - The name of the source image file in the Amazon S3
bucket. (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with
the Rekognition service.
     * @param bucketName   The name of the Amazon S3 bucket where the source
image is stored.
     * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
     */
    public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();
```

```
            Image targetImage = Image.builder()
                .s3Object(s3ObjectTarget)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(targetImage)
                .build();

            DetectFacesResponse facesResponse =
 rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between
 "
                        + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                        + " years old.");

                System.out.println("There is a smile : " +
 face.smile().value().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between
 ${ageRange?.low} and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
```

```
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces
```

- Para obter detalhes da API, consulte a DetectFacesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect faces in the image with all attributes
        DATA(lt_attributes) = VALUE /aws1/cl_rekattributes_w=>tt_attributes( ).
        DATA(lo_attr_wrapper) = NEW /aws1/cl_rekattributes_w( iv_value = 'ALL' ).
        INSERT lo_attr_wrapper INTO TABLE lt_attributes.

        oo_result = lo_rek->detectfaces(
          io_image = lo_image
          it_attributes = lt_attributes ).

        DATA(lt_face_details) = oo_result->get_facedetails( ).
```

```
        DATA(lv_detected_count) = lines( lt_face_details ).
        DATA(lv_msg8) = |{ lv_detected_count } face(s) detected in image.|.
        MESSAGE lv_msg8 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetectFacesreferência da API AWS SDK for SAP ABAP.

Use **`DetectLabels`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectLabels`.

Para obter mais informações, consulte Detectar rótulos em uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectLabels
{
    public static async Task Main()
```

```
        {
            string photo = "del_river_02092020_01.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectlabelsRequest = new DetectLabelsRequest
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
                MaxLabels = 10,
                MinConfidence = 75F,
            };

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Detecte rótulos em um arquivo de imagem armazenado em seu computador.

```
    using System;
```

```
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to detect labels within an image
    /// stored locally.
    /// </summary>
    public class DetectLabelsLocalFile
    {
        public static async Task Main()
        {
            string photo = "input.jpg";

            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            var rekognitionClient = new AmazonRekognitionClient();

            var detectlabelsRequest = new DetectLabelsRequest
            {
                Image = image,
                MaxLabels = 10,
                MinConfidence = 77F,
            };

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
```

```
                Console.WriteLine($"Detected labels for {photo}");
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"{label.Name}: {label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectLabelsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Detect instances of real-world entities within an image by using Amazon
 Rekognition
/*!
  \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
 containing an image.
  \param imageKey: The Amazon S3 key of an image object.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
                                       const Aws::String &imageKey,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3Object;
    s3Object.SetBucket(imageBucket);
    s3Object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3Object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
 rekognitionClient.DetectLabels(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
 outcome.GetResult().GetLabels();
        if (labels.empty()) {
            std::cout << "No labels detected" << std::endl;
        } else {
            for (const Aws::Rekognition::Model::Label &label: labels) {
                std::cout << label.GetName() << ": " << label.GetConfidence() <<
 std::endl;
            }
        }
    } else {
        std::cerr << "Error while detecting labels: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DetectLabelsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como detectar rótulos em uma imagem

O exemplo de `detect-labels` a seguir detecta cenas e objetos em uma imagem armazenada em um bucket do Amazon S3.

```
aws rekognition detect-labels \
    --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}'
```

Saída:

```
{
    "Labels": [
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Automobile"
        },
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
            "Parents": [
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Vehicle"
        },
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
            "Parents": [],
```

```
            "Name": "Transportation"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.10616336017847061,
                        "Top": 0.5039216876029968,
                        "Left": 0.0037978808395564556,
                        "Height": 0.18528179824352264
                    },
                    "Confidence": 99.15271759033203
                },
                {
                    "BoundingBox": {
                        "Width": 0.2429988533258438,
                        "Top": 0.5251884460449219,
                        "Left": 0.7309805154800415,
                        "Height": 0.21577216684818268
                    },
                    "Confidence": 99.1286392211914
                },
                {
                    "BoundingBox": {
                        "Width": 0.14233611524105072,
                        "Top": 0.5333095788955688,
                        "Left": 0.6494812965393066,
                        "Height": 0.15528248250484467
                    },
                    "Confidence": 98.48368072509766
                },
                {
                    "BoundingBox": {
                        "Width": 0.11086395382881165,
                        "Top": 0.5354844927787781,
                        "Left": 0.10355594009160995,
                        "Height": 0.10271988064050674
                    },
                    "Confidence": 96.45606231689453
                },
                {
                    "BoundingBox": {
                        "Width": 0.06254628300666809,
                        "Top": 0.5573825240135193,
```

```
                "Left": 0.46083059906959534,
                "Height": 0.053911514580249786
            },
            "Confidence": 93.65448760986328
        },
        {
            "BoundingBox": {
                "Width": 0.10105438530445099,
                "Top": 0.534368634223938,
                "Left": 0.5743985772132874,
                "Height": 0.12226245552301407
            },
            "Confidence": 93.06217193603516
        },
        {
            "BoundingBox": {
                "Width": 0.056389667093753815,
                "Top": 0.5235804319381714,
                "Left": 0.9427769780158997,
                "Height": 0.17163699865341187
            },
            "Confidence": 92.6864013671875
        },
        {
            "BoundingBox": {
                "Width": 0.06003860384225845,
                "Top": 0.5441341400146484,
                "Left": 0.22409997880458832,
                "Height": 0.06737709045410156
            },
            "Confidence": 90.4227066040039
        },
        {
            "BoundingBox": {
                "Width": 0.02848697081208229,
                "Top": 0.5107086896896362,
                "Left": 0,
                "Height": 0.19150497019290924
            },
            "Confidence": 86.65286254882812
        },
        {
            "BoundingBox": {
                "Width": 0.04067881405353546,
```

```
                        "Top": 0.5566273927688599,
                        "Left": 0.316415935754776,
                        "Height": 0.03428703173995018
                    },
                    "Confidence": 85.36471557617188
                },
                {
                    "BoundingBox": {
                        "Width": 0.043411049991846085,
                        "Top": 0.5394920110702515,
                        "Left": 0.18293385207653046,
                        "Height": 0.0893595889210701
                    },
                    "Confidence": 82.21705627441406
                },
                {
                    "BoundingBox": {
                        "Width": 0.031183116137981415,
                        "Top": 0.5579366683959961,
                        "Left": 0.2853088080883026,
                        "Height": 0.03989990055561066
                    },
                    "Confidence": 81.0157470703125
                },
                {
                    "BoundingBox": {
                        "Width": 0.031113790348172188,
                        "Top": 0.5504819750785828,
                        "Left": 0.2580395042896271,
                        "Height": 0.056484755128622055
                    },
                    "Confidence": 56.13441467285156
                },
                {
                    "BoundingBox": {
                        "Width": 0.08586374670267105,
                        "Top": 0.5438792705535889,
                        "Left": 0.5128012895584106,
                        "Height": 0.08550430089235306
                    },
                    "Confidence": 52.37760925292969
                }
            ],
            "Confidence": 99.15271759033203,
```

```
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Car"
        },
        {
            "Instances": [],
            "Confidence": 98.9914321899414,
            "Parents": [],
            "Name": "Human"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.19360728561878204,
                        "Top": 0.35072067379951477,
                        "Left": 0.43734854459762573,
                        "Height": 0.2742200493812561
                    },
                    "Confidence": 98.9914321899414
                },
                {
                    "BoundingBox": {
                        "Width": 0.03801717236638069,
                        "Top": 0.5010883808135986,
                        "Left": 0.9155802130699158,
                        "Height": 0.06597328186035156
                    },
                    "Confidence": 85.02790832519531
                }
            ],
            "Confidence": 98.9914321899414,
            "Parents": [],
            "Name": "Person"
        },
        {
            "Instances": [],
            "Confidence": 93.24951934814453,
```

```
            "Parents": [],
            "Name": "Machine"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.03561960905790329,
                        "Top": 0.6468243598937988,
                        "Left": 0.7850857377052307,
                        "Height": 0.08878646790981293
                    },
                    "Confidence": 93.24951934814453
                },
                {
                    "BoundingBox": {
                        "Width": 0.02217046171426773,
                        "Top": 0.6149078607559204,
                        "Left": 0.04757237061858177,
                        "Height": 0.07136218994855881
                    },
                    "Confidence": 91.5025863647461
                },
                {
                    "BoundingBox": {
                        "Width": 0.016197510063648224,
                        "Top": 0.6274210214614868,
                        "Left": 0.6472989320755005,
                        "Height": 0.04955997318029404
                    },
                    "Confidence": 85.14686584472656
                },
                {
                    "BoundingBox": {
                        "Width": 0.020207518711686134,
                        "Top": 0.6348286867141724,
                        "Left": 0.7295016646385193,
                        "Height": 0.07059963047504425
                    },
                    "Confidence": 83.34547424316406
                },
                {
                    "BoundingBox": {
                        "Width": 0.020280985161662102,
```

```
                "Top": 0.6171894669532776,
                "Left": 0.08744934946298599,
                "Height": 0.05297485366463661
            },
            "Confidence": 79.9981460571289
        },
        {
            "BoundingBox": {
                "Width": 0.018318990245461464,
                "Top": 0.623889148235321,
                "Left": 0.6836880445480347,
                "Height": 0.06730121374130249
            },
            "Confidence": 78.87144470214844
        },
        {
            "BoundingBox": {
                "Width": 0.021310249343514442,
                "Top": 0.6167286038398743,
                "Left": 0.004064912907779217,
                "Height": 0.08317798376083374
            },
            "Confidence": 75.89361572265625
        },
        {
            "BoundingBox": {
                "Width": 0.03604431077837944,
                "Top": 0.7030032277107239,
                "Left": 0.9254803657531738,
                "Height": 0.04569442570209503
            },
            "Confidence": 64.402587890625
        },
        {
            "BoundingBox": {
                "Width": 0.009834849275648594,
                "Top": 0.5821820497512817,
                "Left": 0.28094568848609924,
                "Height": 0.01964157074689865
            },
            "Confidence": 62.79907989501953
        },
        {
            "BoundingBox": {
```

```
                        "Width": 0.01475677452981472,
                        "Top": 0.6137543320655823,
                        "Left": 0.5950819253921509,
                        "Height": 0.039063986390829086
                    },
                    "Confidence": 59.40483474731445
                }
            ],
            "Confidence": 93.24951934814453,
            "Parents": [
                {
                    "Name": "Machine"
                }
            ],
            "Name": "Wheel"
        },
        {
            "Instances": [],
            "Confidence": 92.61514282226562,
            "Parents": [],
            "Name": "Road"
        },
        {
            "Instances": [],
            "Confidence": 92.37877655029297,
            "Parents": [
                {
                    "Name": "Person"
                }
            ],
            "Name": "Sport"
        },
        {
            "Instances": [],
            "Confidence": 92.37877655029297,
            "Parents": [
                {
                    "Name": "Person"
                }
            ],
            "Name": "Sports"
        },
        {
            "Instances": [
```

```
                {
                    "BoundingBox": {
                        "Width": 0.12326609343290329,
                        "Top": 0.6332163214683533,
                        "Left": 0.44815489649772644,
                        "Height": 0.058117982000112534
                    },
                    "Confidence": 92.37877655029297
                }
            ],
            "Confidence": 92.37877655029297,
            "Parents": [
                {
                    "Name": "Person"
                },
                {
                    "Name": "Sport"
                }
            ],
            "Name": "Skateboard"
        },
        {
            "Instances": [],
            "Confidence": 90.62931060791016,
            "Parents": [
                {
                    "Name": "Person"
                }
            ],
            "Name": "Pedestrian"
        },
        {
            "Instances": [],
            "Confidence": 88.81334686279297,
            "Parents": [],
            "Name": "Asphalt"
        },
        {
            "Instances": [],
            "Confidence": 88.81334686279297,
            "Parents": [],
            "Name": "Tarmac"
        },
        {
```

```
            "Instances": [],
            "Confidence": 88.23201751708984,
            "Parents": [],
            "Name": "Path"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [],
            "Name": "Urban"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "Town"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [],
            "Name": "Building"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "City"
        },
        {
```

```
            "Instances": [],
            "Confidence": 78.37934875488281,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Parking Lot"
        },
        {
            "Instances": [],
            "Confidence": 78.37934875488281,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Parking"
        },
        {
            "Instances": [],
            "Confidence": 74.37590026855469,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                },
                {
                    "Name": "City"
                }
```

```
            ],
            "Name": "Downtown"
        },
        {
            "Instances": [],
            "Confidence": 69.84622955322266,
            "Parents": [
                {
                    "Name": "Road"
                }
            ],
            "Name": "Intersection"
        },
        {
            "Instances": [],
            "Confidence": 57.68518829345703,
            "Parents": [
                {
                    "Name": "Sports Car"
                },
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Coupe"
        },
        {
            "Instances": [],
            "Confidence": 57.68518829345703,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
```

```
                }
            ],
            "Name": "Sports Car"
        },
        {
            "Instances": [],
            "Confidence": 56.59492111206055,
            "Parents": [
                {
                    "Name": "Path"
                }
            ],
            "Name": "Sidewalk"
        },
        {
            "Instances": [],
            "Confidence": 56.59492111206055,
            "Parents": [
                {
                    "Name": "Path"
                }
            ],
            "Name": "Pavement"
        },
        {
            "Instances": [],
            "Confidence": 55.58770751953125,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "Neighborhood"
        }
    ],
    "LabelModelVersion": "2.0"
}
```

Para obter mais informações, consulte Detectar rótulos em uma imagem no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DetectLabelsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
 stored
                sourceImage - The name of the image file (for example, pic1.png).
\s
            """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the
Amazon Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the
detection request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectLabelsRequest detectLabelsRequest =
DetectLabelsRequest.builder()
                    .image(souImage)
                    .maxLabels(10)
                    .build();
```

```
            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label : labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectLabelsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }
```

```
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectLabelsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client
```

```
    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels
```

- Para obter detalhes da API, consulte a DetectLabelsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
```

```
      DATA(lo_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_s3object ).

      " Detect labels in the image
      oo_result = lo_rek->detectlabels(
        io_image = lo_image
        iv_maxlabels = iv_max_labels ).

      DATA(lt_labels) = oo_result->get_labels( ).
      DATA(lv_label_count) = lines( lt_labels ).
      DATA(lv_msg9) = |{ lv_label_count } label(s) detected in image.|.
      MESSAGE lv_msg9 TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DetectLabelsreferência da API AWS SDK for SAP ABAP.

Use **`DetectModerationLabels`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectModerationLabels`.

Para obter mais informações, consulte Detectar imagens impróprias.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```

```
    /// <summary>
    /// Uses the Amazon Rekognition Service to detect unsafe content in a
    /// JPEG or PNG format image.
    /// </summary>
    public class DetectModerationLabels
    {
        public static async Task Main(string[] args)
        {
            string photo = "input.jpg";
            string bucket = "amzn-s3-demo-bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectModerationLabelsRequest = new
DetectModerationLabelsRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
                MinConfidence = 60F,
            };

            try
            {
                var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
                {
                    Console.WriteLine($"Label: {label.Name}");
                    Console.WriteLine($"Confidence: {label.Confidence}");
                    Console.WriteLine($"Parent: {label.ParentName}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
```

```
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectModerationLabelsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como detectar conteúdo não seguro em uma imagem

O comando `detect-moderation-labels` a seguir detecta conteúdo não seguro na imagem especificada armazenada em um bucket do Amazon S3.

```
aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

Saída:

```
{
    "ModerationModelVersion": "3.0",
    "ModerationLabels": [
        {
            "Confidence": 97.29618072509766,
            "ParentName": "Violence",
            "Name": "Weapon Violence"
        },
        {
            "Confidence": 97.29618072509766,
            "ParentName": "",
            "Name": "Violence"
        }
    ]
}
```

Para obter mais informações, consulte Detectando imagens inapropriadas no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte DetectModerationLabelsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = """
            Usage:  <bucketName>  <sourceImage>

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
                sourceImage - The name of the image (for example, pic1.png).\s
```

```
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the
detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image
detection process
     */
    public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
```

```
                    .image(targetImage)
                    .minConfidence(60F)
                    .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                    .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
 moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                        + "\n Confidence: " + label.confidence().toString() + "%"
                        + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectModerationLabelsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }
```

```
    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectModerationLabelsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
```

```
                        an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

- Para obter detalhes da API, consulte a DetectModerationLabelsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Detect moderation labels
    oo_result = lo_rek->detectmoderationlabels(
      io_image = lo_image ).

    DATA(lt_moderation_labels) = oo_result->get_moderationlabels( ).
    DATA(lv_mod_count) = lines( lt_moderation_labels ).
    DATA(lv_msg10) = |{ lv_mod_count } moderation label(s) detected.|.
    MESSAGE lv_msg10 TYPE 'I'.
  CATCH /aws1/cx_rekinvalids3objectex.
    MESSAGE 'Invalid S3 object.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectModerationLabelsreferência da API AWS SDK for SAP ABAP.

Use **`DetectText`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectText`.

Para obter mais informações, consulte Detectar texto em uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect text in an image. The
/// example was created using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
/// </summary>
public class DetectText
{
    public static async Task Main()
    {
        string photo = "Dad_photographer.jpg"; // "input.jpg";
        string bucket = "amzn-s3-demo-bucket"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
        };
```

```
            try
            {
                DetectTextResponse detectTextResponse = await
 rekognitionClient.DetectTextAsync(detectTextRequest);
                Console.WriteLine($"Detected lines and words for {photo}");
                detectTextResponse.TextDetections.ForEach(text =>
                {
                    Console.WriteLine($"Detected: {text.DetectedText}");
                    Console.WriteLine($"Confidence: {text.Confidence}");
                    Console.WriteLine($"Id : {text.Id}");
                    Console.WriteLine($"Parent Id: {text.ParentId}");
                    Console.WriteLine($"Type: {text.Type}");
                });
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectTexta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como detectar texto em uma imagem

O comando `detect-text` a seguir detecta texto na imagem especificada.

```
aws rekognition detect-text \
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

Saída:

```
{
    "TextDetections": [
        {
```

```
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.24624845385551453,
                    "Top": 0.28288066387176514,
                    "Left": 0.391388863325119,
                    "Height": 0.022687450051307678
                },
                "Polygon": [
                    {
                        "Y": 0.28288066387176514,
                        "X": 0.391388863325119
                    },
                    {
                        "Y": 0.2826388478279114,
                        "X": 0.6376373171806335
                    },
                    {
                        "Y": 0.30532628297805786,
                        "X": 0.637677013874054
                    },
                    {
                        "Y": 0.305568128824234,
                        "X": 0.39142853021621704
                    }
                ]
            },
            "Confidence": 94.35709381103516,
            "DetectedText": "ESTD 1882",
            "Type": "LINE",
            "Id": 0
        },
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33933889865875244,
                    "Top": 0.32603850960731506,
                    "Left": 0.34534579515457153,
                    "Height": 0.07126858830451965
                },
                "Polygon": [
                    {
                        "Y": 0.32603850960731506,
                        "X": 0.34534579515457153
                    },
```

```
                {
                    "Y": 0.32633158564567566,
                    "X": 0.684684693813324
                },
                {
                    "Y": 0.3976001739501953,
                    "X": 0.684575080871582
                },
                {
                    "Y": 0.3973070979118347,
                    "X": 0.345236212015152
                }
            ]
        },
        "Confidence": 99.95779418945312,
        "DetectedText": "BRAINS",
        "Type": "LINE",
        "Id": 1
    },
    {
        "Confidence": 97.22098541259766,
        "Geometry": {
            "BoundingBox": {
                "Width": 0.061079490929841995,
                "Top": 0.2843210697174072,
                "Left": 0.391391396522522,
                "Height": 0.021029088646173477
            },
            "Polygon": [
                {
                    "Y": 0.2843210697174072,
                    "X": 0.391391396522522
                },
                {
                    "Y": 0.2828207015991211,
                    "X": 0.4524524509906769
                },
                {
                    "Y": 0.3038259446620941,
                    "X": 0.4534534513950348
                },
                {
                    "Y": 0.30532634258270264,
                    "X": 0.3923923969268799
```

```
                    }
                ]
            },
            "DetectedText": "ESTD",
            "ParentId": 0,
            "Type": "WORD",
            "Id": 2
        },
        {
            "Confidence": 91.49320983886719,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.07007007300853729,
                    "Top": 0.2828207015991211,
                    "Left": 0.5675675868988037,
                    "Height": 0.02250562608242035
                },
                "Polygon": [
                    {
                        "Y": 0.2828207015991211,
                        "X": 0.5675675868988037
                    },
                    {
                        "Y": 0.2828207015991211,
                        "X": 0.6376376152038574
                    },
                    {
                        "Y": 0.30532634258270264,
                        "X": 0.6376376152038574
                    },
                    {
                        "Y": 0.30532634258270264,
                        "X": 0.5675675868988037
                    }
                ]
            },
            "DetectedText": "1882",
            "ParentId": 0,
            "Type": "WORD",
            "Id": 3
        },
        {
            "Confidence": 99.95779418945312,
            "Geometry": {
```

```
                "BoundingBox": {
                    "Width": 0.33933934569358826,
                    "Top": 0.32633158564567566,
                    "Left": 0.3453453481197357,
                    "Height": 0.07127484679222107
                },
                "Polygon": [
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.3453453481197357
                    },
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.684684693813324
                    },
                    {
                        "Y": 0.39759939908981323,
                        "X": 0.6836836934089661
                    },
                    {
                        "Y": 0.39684921503067017,
                        "X": 0.3453453481197357
                    }
                ]
            },
            "DetectedText": "BRAINS",
            "ParentId": 1,
            "Type": "WORD",
            "Id": 4
        }
    ]
}
```

- Para obter detalhes da API, consulte DetectTextem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored
\n" +
            "   sourceImage - The path to the image that contains text (for
 example, pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
Rekognition API
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
```

```
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " +
 text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectTexta Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
```

```
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectTextreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client
```

```
    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

- Para obter detalhes da API, consulte a DetectTextReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
```

```
          io_s3object = lo_s3object ).

        " Detect text in the image
        oo_result = lo_rek->detecttext(
          io_image = lo_image ).

        DATA(lt_text_detections) = oo_result->get_textdetections( ).
        DATA(lv_text_count) = lines( lt_text_detections ).
        DATA(lv_msg11) = |{ lv_text_count } text detection(s) found.|.
        MESSAGE lv_msg11 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetectTextreferência da API AWS SDK for SAP ABAP.

Use **`GetCelebrityInfo`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetCelebrityInfo`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
```

```
    /// celebrity identified by the supplied celebrity Id.
    /// </summary>
    public class CelebrityInfo
    {
        public static async Task Main()
        {
            string celebId = "nnnnnnnn";

            var rekognitionClient = new AmazonRekognitionClient();

            var celebrityInfoRequest = new GetCelebrityInfoRequest
            {
                Id = celebId,
            };

            Console.WriteLine($"Getting information for celebrity: {celebId}");

            var celebrityInfoResponse = await
rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

            // Display celebrity information.
            Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
            Console.WriteLine("Further information (if available):");
            celebrityInfoResponse.Urls.ForEach(url =>
            {
                Console.WriteLine(url);
            });
        }
    }
```

- Para obter detalhes da API, consulte GetCelebrityInfoa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como obter informações sobre uma celebridade

O comando `get-celebrity-info` a seguir exibe informações sobre a celebridade especificada. O parâmetro `id` é proveniente de uma chamada anterior a `recognize-celebrities`.

```
aws rekognition get-celebrity-info --id nnnnnnn
```

Saída:

```
{
    "Name": "Celeb A",
    "Urls": [
        "www.imdb.com/name/aaaaaaaaa"
    ]
}
```

Para obter mais informações, consulte Obter informações sobre uma celebridade no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte GetCelebrityInfoem Referência de AWS CLI Comandos.

Use **`IndexFaces`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `IndexFaces`.

Para obter mais informações, consulte Adicionar faces a uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
```

```
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to detect faces in an image
    /// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
    /// bucket and then adds the information to a collection.
    /// </summary>
    public class AddFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection2";
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Image
            {
                S3Object = new S3Object
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var indexFacesRequest = new IndexFacesRequest
            {
                Image = image,
                CollectionId = collectionId,
                ExternalImageId = photo,
                DetectionAttributes = new List<string>() { "ALL" },
            };

            IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

            Console.WriteLine($"{photo} added");
            foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            {
                Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
            }
        }
```

```
    }
```

- Para obter detalhes da API, consulte IndexFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como adicionar faces a uma coleção

O comando `index-faces` a seguir adiciona as faces encontradas em uma imagem à coleção especificada.

```
aws rekognition index-faces \
    --image '{"S3Object":{"Bucket":"MyVideoS3Bucket","Name":"MyPicture.jpg"}}' \
    --collection-id MyCollection \
    --max-faces 1 \
    --quality-filter "AUTO" \
    --detection-attributes "ALL" \
    --external-image-id "MyPicture.jpg"
```

Saída:

```
{
    "FaceRecords": [
        {
            "FaceDetail": {
                "Confidence": 99.993408203125,
                "Eyeglasses": {
                    "Confidence": 99.11750030517578,
                    "Value": false
                },
                "Sunglasses": {
                    "Confidence": 99.98249053955078,
                    "Value": false
                },
                "Gender": {
                    "Confidence": 99.92769622802734,
                    "Value": "Male"
                },
```

```
                "Landmarks": [
                    {
                        "Y": 0.26750367879867554,
                        "X": 0.6202793717384338,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.26642778515815735,
                        "X": 0.6787431836128235,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.31361380219459534,
                        "X": 0.6421601176261902,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.3495299220085144,
                        "X": 0.6216195225715637,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.35194727778434753,
                        "X": 0.669899046421051,
                        "Type": "mouthRight"
                    },
                    {
                        "Y": 0.26844894886016846,
                        "X": 0.6210268139839172,
                        "Type": "leftPupil"
                    },
                    {
                        "Y": 0.26707562804222107,
                        "X": 0.6817160844802856,
                        "Type": "rightPupil"
                    },
                    {
                        "Y": 0.24834522604942322,
                        "X": 0.6018546223640442,
                        "Type": "leftEyeBrowLeft"
                    },
                    {
                        "Y": 0.24397172033786774,
                        "X": 0.6172008514404297,
```

```
                "Type": "leftEyeBrowUp"
            },
            {
                "Y": 0.24677404761314392,
                "X": 0.6339119076728821,
                "Type": "leftEyeBrowRight"
            },
            {
                "Y": 0.24582654237747192,
                "X": 0.6619398593902588,
                "Type": "rightEyeBrowLeft"
            },
            {
                "Y": 0.23973053693771362,
                "X": 0.6804757118225098,
                "Type": "rightEyeBrowUp"
            },
            {
                "Y": 0.24441994726657867,
                "X": 0.6978968977928162,
                "Type": "rightEyeBrowRight"
            },
            {
                "Y": 0.2695908546447754,
                "X": 0.6085202693939209,
                "Type": "leftEyeLeft"
            },
            {
                "Y": 0.26716896891593933,
                "X": 0.6315826177597046,
                "Type": "leftEyeRight"
            },
            {
                "Y": 0.26289820671081543,
                "X": 0.6202316880226135,
                "Type": "leftEyeUp"
            },
            {
                "Y": 0.27123287320137024,
                "X": 0.6205548048019409,
                "Type": "leftEyeDown"
            },
            {
                "Y": 0.2668408751487732,
```

```
                "X": 0.6663622260093689,
                "Type": "rightEyeLeft"
            },
            {
                "Y": 0.26741549372673035,
                "X": 0.6910083889961243,
                "Type": "rightEyeRight"
            },
            {
                "Y": 0.2614026665687561,
                "X": 0.6785826086997986,
                "Type": "rightEyeUp"
            },
            {
                "Y": 0.27075251936912537,
                "X": 0.6789616942405701,
                "Type": "rightEyeDown"
            },
            {
                "Y": 0.3211299479007721,
                "X": 0.6324167847633362,
                "Type": "noseLeft"
            },
            {
                "Y": 0.32276326417922974,
                "X": 0.6558475494384766,
                "Type": "noseRight"
            },
            {
                "Y": 0.34385165572166443,
                "X": 0.6444970965385437,
                "Type": "mouthUp"
            },
            {
                "Y": 0.3671635091304779,
                "X": 0.6459195017814636,
                "Type": "mouthDown"
            }
        ],
        "Pose": {
            "Yaw": -9.54541015625,
            "Roll": -0.5709401965141296,
            "Pitch": 0.6045494675636292
        },
```

```
                "Emotions": [
                    {
                        "Confidence": 39.90074157714844,
                        "Type": "HAPPY"
                    },
                    {
                        "Confidence": 23.38753890991211,
                        "Type": "CALM"
                    },
                    {
                        "Confidence": 5.840933322906494,
                        "Type": "CONFUSED"
                    }
                ],
                "AgeRange": {
                    "High": 63,
                    "Low": 45
                },
                "EyesOpen": {
                    "Confidence": 99.80887603759766,
                    "Value": true
                },
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618015021085739,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770642817020416
                },
                "Smile": {
                    "Confidence": 99.69740295410156,
                    "Value": false
                },
                "MouthOpen": {
                    "Confidence": 99.97393798828125,
                    "Value": false
                },
                "Quality": {
                    "Sharpness": 95.54405975341797,
                    "Brightness": 63.867706298828125
                },
                "Mustache": {
                    "Confidence": 97.05007934570312,
                    "Value": false
                },
```

```
                "Beard": {
                    "Confidence": 87.34505462646484,
                    "Value": false
                }
            },
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618015021085739,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770642817020416
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.993408203125,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            }
        }
    ],
    "UnindexedFaces": [],
    "FaceModelVersion": "3.0",
    "OrientationCorrection": "ROTATE_0"
}
```

Para obter mais informações, consulte Adicionar faces a uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte IndexFacesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <sourceImage> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
```

```
    * @param bucketName    the name of the Amazon S3 bucket containing the image
    * @param sourceImage   the name of the image file to add to the collection
    * @throws RekognitionException if there is an error while interacting with
the Amazon Rekognition service
    */
   public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                   .collectionId(collectionId)
                   .image(targetImage)
                   .maxFaces(1)
                   .qualityFilter(QualityFilter.AUTO)
                   .detectionAttributes(Attribute.DEFAULT)
                   .build();

           IndexFacesResponse facesResponse =
rekClient.indexFaces(facesRequest);
           System.out.println("Results for the image");
           System.out.println("\n Faces indexed:");
           List<FaceRecord> faceRecords = facesResponse.faceRecords();
           for (FaceRecord faceRecord : faceRecords) {
               System.out.println("  Face ID: " + faceRecord.face().faceId());
               System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
           }

           List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
           System.out.println("Faces not indexed:");
           for (UnindexedFace unindexedFace : unindexedFaces) {
               System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
               System.out.println("  Reasons:");
               for (Reason reason : unindexedFace.reasons()) {
                   System.out.println("Reason:  " + reason);
```

```
                }
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte IndexFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }
```

```
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a IndexFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
```

```
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces
```

- Para obter detalhes da API, consulte a IndexFacesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Index faces in the image
        oo_result = lo_rek->indexfaces(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_externalimageid = iv_external_id
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_records) = oo_result->get_facerecords( ).
        DATA(lv_indexed_count) = lines( lt_face_records ).
        DATA(lv_msg2) = |{ lv_indexed_count } face(s) indexed successfully.|.
        MESSAGE lv_msg2 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a IndexFacesreferência da API AWS SDK for SAP ABAP.

Use **`ListCollections`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListCollections`.

Para obter mais informações, consulte Listar coleções.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to list the collection IDs in the
/// current account.
/// </summary>
public class ListCollections
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        var listCollectionsRequest = new ListCollectionsRequest
        {
            MaxResults = limit,
        };

        var listCollectionsResponse = new ListCollectionsResponse();

        do
        {
            if (listCollectionsResponse is not null)
            {
                listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
            }
```

```
                listCollectionsResponse = await
 rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

                listCollectionsResponse.CollectionIds.ForEach(id =>
                {
                    Console.WriteLine(id);
                });
            }
            while (listCollectionsResponse.NextToken is not null);
        }
    }
```

- Para obter detalhes da API, consulte ListCollectionsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar as coleções disponíveis

O `list-collections` comando a seguir lista as coleções disponíveis na AWS conta.

```
aws rekognition list-collections
```

Saída:

```
{
    "FaceModelVersions": [
        "2.0",
        "3.0",
        "3.0",
        "3.0",
        "4.0",
        "1.0",
        "3.0",
        "4.0",
        "4.0",
        "4.0"
```

```
    ],
    "CollectionIds": [
        "MyCollection1",
        "MyCollection2",
        "MyCollection3",
        "MyCollection4",
        "MyCollection5",
        "MyCollection6",
        "MyCollection7",
        "MyCollection8",
        "MyCollection9",
        "MyCollection10"
    ]
}
```

Para obter mais informações, consulte Listar coleções no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte ListCollectionsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
                    .maxResults(10)
                    .build();

            ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListCollectionsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- Para obter detalhes da API, consulte a ListCollectionsreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
```

```
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections
```

- Para obter detalhes da API, consulte a ListCollectionsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_rek->listcollections(
      iv_maxresults = iv_max_results ).

    DATA(lt_collection_ids) = oo_result->get_collectionids( ).
    DATA(lv_coll_count) = lines( lt_collection_ids ).
    DATA(lv_msg7) = |{ lv_coll_count } collection(s) found.|.
    MESSAGE lv_msg7 TYPE 'I'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListCollectionsreferência da API AWS SDK for SAP ABAP.

Use **`ListFaces`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListFaces`.

Para obter mais informações, consulte Listar faces em uma coleção.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to retrieve the list of faces
    /// stored in a collection.
    /// </summary>
    public class ListFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection2";

            var rekognitionClient = new AmazonRekognitionClient();

            var listFacesResponse = new ListFacesResponse();
            Console.WriteLine($"Faces in collection {collectionId}");

            var listFacesRequest = new ListFacesRequest
            {
                CollectionId = collectionId,
                MaxResults = 1,
            };

            do
            {
                listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
                listFacesResponse.Faces.ForEach(face =>
                {
                    Console.WriteLine(face.FaceId);
                });

                listFacesRequest.NextToken = listFacesResponse.NextToken;
            }
            while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
        }
    }
```

- Para obter detalhes da API, consulte ListFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar faces de uma coleção

O comando `list-faces` a seguir lista as faces na coleção especificada.

```
aws rekognition list-faces \
    --collection-id MyCollection
```

Saída:

```
{
    "FaceModelVersion": "3.0",
    "Faces": [
        {
            "BoundingBox": {
                "Width": 0.5216310024261475,
                "Top": 0.3256250023841858,
                "Left": 0.13394300639629364,
                "Height": 0.3918749988079071
            },
            "FaceId": "0040279c-0178-436e-b70a-e61b074e96b0",
            "ExternalImageId": "image1.jpg",
            "Confidence": 100.0,
            "ImageId": "f976e487-3719-5e2d-be8b-ea2724c26991"
        },
        {
            "BoundingBox": {
                "Width": 0.5074880123138428,
                "Top": 0.3774999976158142,
                "Left": 0.18302799761295319,
                "Height": 0.3812499940395355
            },
            "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
            "ExternalImageId": "image2.jpg",
            "Confidence": 99.99930572509766,
            "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
        },
        {
```

```
            "BoundingBox": {
                "Width": 0.5574039816856384,
                "Top": 0.37187498807907104,
                "Left": 0.14559100568294525,
                "Height": 0.4181250035762787
            },
            "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
            "ExternalImageId": "image3.jpg",
            "Confidence": 99.99960327148438,
            "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
        },
        {
            "BoundingBox": {
                "Width": 0.18562500178813934,
                "Top": 0.1618019938468933,
                "Left": 0.5575000047683716,
                "Height": 0.24770599603652954
            },
            "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
            "ExternalImageId": "image4.jpg",
            "Confidence": 99.99340057373047,
            "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
        },
        {
            "BoundingBox": {
                "Width": 0.5307819843292236,
                "Top": 0.2862499952316284,
                "Left": 0.1564060002565384,
                "Height": 0.3987500071525574
            },
            "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
            "ExternalImageId": "image5.jpg",
            "Confidence": 99.99970245361328,
            "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
        },
        {
            "BoundingBox": {
                "Width": 0.5773710012435913,
                "Top": 0.34437501430511475,
                "Left": 0.12396000325679779,
                "Height": 0.4337500035762787
            },
            "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
            "ExternalImageId": "image6.jpg",
```

```
            "Confidence": 100.0,
            "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
        },
        {
            "BoundingBox": {
                "Width": 0.5349419713020325,
                "Top": 0.29124999046325684,
                "Left": 0.16389399766921997,
                "Height": 0.40187498927116394
            },
            "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
            "ExternalImageId": "image7.jpg",
            "Confidence": 99.99979400634766,
            "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
        },
        {
            "BoundingBox": {
                "Width": 0.41499999165534973,
                "Top": 0.09187500178813934,
                "Left": 0.28083300590515137,
                "Height": 0.3112500011920929
            },
            "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
            "ExternalImageId": "image8.jpg",
            "Confidence": 99.99769592285156,
            "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
        },
        {
            "BoundingBox": {
                "Width": 0.48166701197624207,
                "Top": 0.20999999344348907,
                "Left": 0.21250000596046448,
                "Height": 0.36125001311302185
            },
            "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
            "ExternalImageId": "image9.jpg",
            "Confidence": 99.99949645996094,
            "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
        },
        {
            "BoundingBox": {
                "Width": 0.18562500178813934,
                "Top": 0.1618019938468933,
                "Left": 0.5575000047683716,
```

```
                "Height": 0.24770599603652954
            },
            "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
            "ExternalImageId": "image10.jpg",
            "Confidence": 99.99340057373047,
            "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
        }
    ]
}
```

Para obter mais informações, consulte Listar faces em uma coleção no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte ListFacesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId>

                Where:
                   collectionId - The name of the collection.\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                    .collectionId(collectionId)
                    .maxResults(10)
                    .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
 face.confidence());
                System.out.println("The face Id value is " + face.faceId());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListFacesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListFacesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces
```

- Para obter detalhes da API, consulte a ListFacesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_rek->listfaces(
      iv_collectionid = iv_collection_id
      iv_maxresults = iv_max_results ).

    DATA(lt_faces) = oo_result->get_faces( ).
    DATA(lv_face_count2) = lines( lt_faces ).
    DATA(lv_msg3) = |{ lv_face_count2 } face(s) found in collection.|.
    MESSAGE lv_msg3 TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListFacesreferência da API AWS SDK for SAP ABAP.

Use **`RecognizeCelebrities`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RecognizeCelebrities`.

Para obter mais informações, consulte Reconhecer celebridades em uma imagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// </summary>
public class CelebritiesInImage
{
    public static async Task Main(string[] args)
    {
        string photo = "moviestars.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

        var img = new Amazon.Rekognition.Model.Image();
        byte[] data = null;
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
        }
        catch (Exception)
        {
            Console.WriteLine($"Failed to load file {photo}");
            return;
```

```
            }

            img.Bytes = new MemoryStream(data);
            recognizeCelebritiesRequest.Image = img;

            Console.WriteLine($"Looking for celebrities in image {photo}\n");

            var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);


Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
            recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
            {
                Console.WriteLine($"Celebrity recognized: {celeb.Name}");
                Console.WriteLine($"Celebrity ID: {celeb.Id}");
                BoundingBox boundingBox = celeb.Face.BoundingBox;
                Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
                Console.WriteLine("Further information (if available):");
                celeb.Urls.ForEach(url =>
                {
                    Console.WriteLine(url);
                });
            });


Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count}
face(s) were unrecognized.");
        }
    }
```

- Para obter detalhes da API, consulte RecognizeCelebritiesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como reconhecer celebridades em uma imagem

O seguinte comando `recognize-celebrities` reconhece celebridades na imagem especificada armazenada em um bucket do Amazon S3:

```
aws rekognition recognize-celebrities \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

Saída:

```
{
    "UnrecognizedFaces": [
        {
            "BoundingBox": {
                "Width": 0.14416666328907013,
                "Top": 0.07777778059244156,
                "Left": 0.625,
                "Height": 0.2746031880378723
            },
            "Confidence": 99.9990234375,
            "Pose": {
                "Yaw": 10.80408763885498,
                "Roll": -12.761146545410156,
                "Pitch": 10.96889877319336
            },
            "Quality": {
                "Sharpness": 94.1185531616211,
                "Brightness": 79.18367004394531
            },
            "Landmarks": [
                {
                    "Y": 0.18220913410186768,
                    "X": 0.6702951788902283,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.16337193548679352,
                    "X": 0.7188183665275574,
                    "Type": "eyeRight"
                },
                {
                    "Y": 0.20739148557186127,
                    "X": 0.7055801749229431,
                    "Type": "nose"
                },
```

```
                {
                    "Y": 0.2889308035373688,
                    "X": 0.687512218952179,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.2706988751888275,
                    "X": 0.7250053286552429,
                    "Type": "mouthRight"
                }
            ]
        }
    ],
    "CelebrityFaces": [
        {
            "MatchConfidence": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.14000000059604645,
                    "Top": 0.1190476194024086,
                    "Left": 0.82833331823349,
                    "Height": 0.2666666805744171
                },
                "Confidence": 99.99359130859375,
                "Pose": {
                    "Yaw": -10.509642601013184,
                    "Roll": -14.51749324798584,
                    "Pitch": 13.799399375915527
                },
                "Quality": {
                    "Sharpness": 78.74752044677734,
                    "Brightness": 42.201324462890625
                },
                "Landmarks": [
                    {
                        "Y": 0.2290833294391632,
                        "X": 0.8709492087364197,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.20639978349208832,
                        "X": 0.9153988361358643,
                        "Type": "eyeRight"
                    },
```

```
                    {
                        "Y": 0.25417643785476685,
                        "X": 0.8907724022865295,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.32729196548461914,
                        "X": 0.8876466155052185,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.3115464746952057,
                        "X": 0.9238573312759399,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb A",
            "Urls": [
                "www.imdb.com/name/aaaaaaaaa"
            ],
            "Id": "1111111"
        },
        {
            "MatchConfidence": 97.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.13333334028720856,
                    "Top": 0.2492063492536544 8,
                    "Left": 0.4449999928474426,
                    "Height": 0.2539682686328888
                },
                "Confidence": 99.99979400634766,
                "Pose": {
                    "Yaw": 6.557040691375732,
                    "Roll": -7.316643714904785,
                    "Pitch": 9.272967338562012
                },
                "Quality": {
                    "Sharpness": 83.23492431640625,
                    "Brightness": 78.83267974853516
                },
                "Landmarks": [
                    {
```

```
                        "Y": 0.3625510632991791,
                        "X": 0.48898839950561523,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.35366007685661316,
                        "X": 0.5313721299171448,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.3894785940647125,
                        "X": 0.5173314809799194,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.44889405369758606,
                        "X": 0.5020005702972412,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.4408611059188843,
                        "X": 0.5351271629333496,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb B",
            "Urls": [
                "www.imdb.com/name/bbbbbbbbb"
            ],
            "Id": "2222222"
        },
        {
            "MatchConfidence": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.12416666746139526,
                    "Top": 0.2968254089355469,
                    "Left": 0.2150000035762787,
                    "Height": 0.23650793731212616
                },
                "Confidence": 99.99958801269531,
                "Pose": {
                    "Yaw": 7.801797866821289,
```

```
                    "Roll": -8.326810836791992,
                    "Pitch": 7.844768047332764
                },
                "Quality": {
                    "Sharpness": 86.93206024169922,
                    "Brightness": 79.81291198730469
                },
                "Landmarks": [
                    {
                        "Y": 0.4027804136276245,
                        "X": 0.2575301229953766,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.3934555947780609,
                        "X": 0.2956969439983368,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.4309830069541931,
                        "X": 0.2837020754814148,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.48186683654785156,
                        "X": 0.26812544465065,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.47338807582855225,
                        "X": 0.29905644059181213,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb C",
            "Urls": [
                "www.imdb.com/name/ccccccccc"
            ],
            "Id": "3333333"
        },
        {
            "MatchConfidence": 97.0,
            "Face": {
```

```
                "BoundingBox": {
                    "Width": 0.11916666477918625,
                    "Top": 0.3698412775993347,
                    "Left": 0.008333333767950535,
                    "Height": 0.22698412835597992
                },
                "Confidence": 99.99999237060547,
                "Pose": {
                    "Yaw": 16.38478660583496,
                    "Roll": -1.0260354280471802,
                    "Pitch": 5.975185394287109
                },
                "Quality": {
                    "Sharpness": 83.23492431640625,
                    "Brightness": 61.408443450927734
                },
                "Landmarks": [
                    {
                        "Y": 0.4632347822189331,
                        "X": 0.049406956881284714,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.46388113498687744,
                        "X": 0.08722897619009018,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.5020678639411926,
                        "X": 0.0758260041475296,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.544157862663269,
                        "X": 0.054029736667871475,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.5463630557060242,
                        "X": 0.08464983850717545,
                        "Type": "mouthRight"
                    }
                ]
            },
```

```
            "Name": "Celeb D",
            "Urls": [
                "www.imdb.com/name/ddddddddd"
            ],
            "Id": "4444444"
        }
    ]
}
```

Para obter mais informações, consulte Reconhecer celebridades em uma imagem no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte RecognizeCelebritiesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = """
                Usage:   <bucketName> <sourceImage>

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
       }

        String bucketName = args[0];;
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source
 image is stored
     * @param sourceImage  the name of the source image file stored in the Amazon
 S3 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient,
 String bucketName, String sourceImage) {
```

```
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            RecognizeCelebritiesRequest request =
 RecognizeCelebritiesRequest.builder()
                    .image(souImage)
                    .build();

            RecognizeCelebritiesResponse result =
 rekClient.recognizeCelebrities(request);
            List<Celebrity> celebs = result.celebrityFaces();
            System.out.println(celebs.size() + " celebrity(s) were recognized.
\n");
            for (Celebrity celebrity : celebs) {
                System.out.println("Celebrity recognized: " + celebrity.name());
                System.out.println("Celebrity ID: " + celebrity.id());

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte RecognizeCelebritiesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- Para obter detalhes da API, consulte a RecognizeCelebritiesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
 self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
```

```
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces
```

- Para obter detalhes da API, consulte a RecognizeCelebritiesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).
```

```
      " Create image object
      DATA(lo_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_s3object ).

      " Recognize celebrities
      oo_result = lo_rek->recognizecelebrities(
        io_image = lo_image ).

      DATA(lt_celebrity_faces) = oo_result->get_celebrityfaces( ).
      DATA(lv_celeb_count) = lines( lt_celebrity_faces ).
      DATA(lv_msg12) = |{ lv_celeb_count } celebrity/celebrities recognized.|.
      MESSAGE lv_msg12 TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a RecognizeCelebritiesreferência da API AWS SDK for SAP ABAP.

Use **`SearchFaces`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SearchFaces`.

Para obter mais informações, consulte Pesquisar uma face (face ID).

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
```

```
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to find faces in an image that
/// match the face Id provided in the method request.
/// </summary>
public class SearchFacesMatchingId
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

        var rekognitionClient = new AmazonRekognitionClient();

        // Search collection for faces matching the face id.
        var searchFacesRequest = new SearchFacesRequest
        {
            CollectionId = collectionId,
            FaceId = faceId,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

        Console.WriteLine("Face matching faceId " + faceId);

        Console.WriteLine("Matche(s): ");
        searchFacesResponse.FaceMatches.ForEach(face =>
        {
            Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
{face.Similarity}");
        });
    }
}
```

- Para obter detalhes da API, consulte SearchFacesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como pesquisar faces em uma coleção que corresponda a um ID facial.

O comando `search-faces` a seguir pesquisa faces em uma coleção que correspondem ao ID facial especificado.

```
aws rekognition search-faces \
    --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \
    --collection-id MyCollection
```

Saída:

```
{
    "SearchedFaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
    "FaceModelVersion": "3.0",
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.48166701197624207,
                    "Top": 0.20999999344348907,
                    "Left": 0.21250000596046448,
                    "Height": 0.36125001311302185
                },
                "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
                "ExternalImageId": "image1.jpg",
                "Confidence": 99.99949645996094,
                "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
            },
            "Similarity": 99.30997467041016
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
```

```
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            },
            "Similarity": 99.24862670898438
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
                "ExternalImageId": "image3.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
            },
            "Similarity": 99.24862670898438
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5349419713020325,
                    "Top": 0.29124999046325684,
                    "Left": 0.16389399766921997,
                    "Height": 0.40187498927116394
                },
                "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
                "ExternalImageId": "image9.jpg",
                "Confidence": 99.99979400634766,
                "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
            },
            "Similarity": 96.73158264160156
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5307819843292236,
                    "Top": 0.2862499952316284,
                    "Left": 0.1564060002565384,
                    "Height": 0.3987500071525574
                },
```

```
                "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
                "ExternalImageId": "image10.jpg",
                "Confidence": 99.99970245361328,
                "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
            },
            "Similarity": 96.48291015625
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5074880123138428,
                    "Top": 0.3774999976158142,
                    "Left": 0.18302799761295319,
                    "Height": 0.3812499940395355
                },
                "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
                "ExternalImageId": "image6.jpg",
                "Confidence": 99.99930572509766,
                "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
            },
            "Similarity": 96.43287658691406
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5574039816856384,
                    "Top": 0.37187498807907104,
                    "Left": 0.14559100568294525,
                    "Height": 0.4181250035762787
                },
                "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
                "ExternalImageId": "image5.jpg",
                "Confidence": 99.99960327148438,
                "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
            },
            "Similarity": 95.25305938720703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5773710012435913,
                    "Top": 0.34437501430511475,
                    "Left": 0.12396000325679779,
                    "Height": 0.4337500035762787
```

```
                },
                "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
                "ExternalImageId": "image8.jpg",
                "Confidence": 100.0,
                "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
            },
            "Similarity": 95.22837829589844
        }
    ]
}
```

Para obter mais informações, consulte Procurando uma face com um ID facial no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte SearchFacesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
 collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new
 File(sourceImage));
```

```
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
 SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
 rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SearchFacesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces
```

- Para obter detalhes da API, consulte a SearchFacesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_rek->searchfaces(
         iv_collectionid = iv_collection_id
         iv_faceid = iv_face_id
         iv_facematchthreshold = iv_threshold
         iv_maxfaces = iv_max_faces ).

       DATA(lt_face_matches) = oo_result->get_facematches( ).
       DATA(lv_match_count2) = lines( lt_face_matches ).
       DATA(lv_msg5) = |Face search completed: { lv_match_count2 } match(es)
found.|.
       MESSAGE lv_msg5 TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection or face not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a SearchFacesreferência da API AWS SDK for SAP ABAP.

Use **`SearchFacesByImage`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SearchFacesByImage`.

Para obter mais informações, consulte Pesquisar uma face (imagem).

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection.
/// </summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string bucket = "amzn-s3-demo-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        var image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var searchFacesByImageRequest = new SearchFacesByImageRequest()
```

```
            {
                CollectionId = collectionId,
                Image = image,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesByImageResponse searchFacesByImageResponse = await
 rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

            Console.WriteLine("Faces matching largest face in image from " +
 photo);
            searchFacesByImageResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
 {face.Similarity}");
            });
        }
    }
```

- Para obter detalhes da API, consulte SearchFacesByImagea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como pesquisar faces em uma coleção que corresponda à maior face em uma imagem.

O seguinte comando `search-faces-by-image` pesquisa faces em uma coleção que corresponda à maior face na imagem especificada:

```
aws rekognition search-faces-by-image \
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePerson.jpg"}}' \
    --collection-id MyFaceImageCollection

{
    "SearchedFaceBoundingBox": {
        "Width": 0.18562500178813934,
        "Top": 0.1618015021085739,
```

```
        "Left": 0.5575000047683716,
        "Height": 0.24770642817020416
    },
    "SearchedFaceConfidence": 99.993408203125,
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            },
            "Similarity": 99.97913360595703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
                "ExternalImageId": "image3.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
            },
            "Similarity": 99.97913360595703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.41499999165534973,
                    "Top": 0.09187500178813934,
                    "Left": 0.28083300590515137,
                    "Height": 0.3112500011920929
                },
                "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
```

```
                "ExternalImageId": "image2.jpg",
                "Confidence": 99.99769592285156,
                "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
            },
            "Similarity": 99.18069458007812
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.48166701197624207,
                    "Top": 0.20999999344348907,
                    "Left": 0.21250000596046448,
                    "Height": 0.36125001311302185
                },
                "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
                "ExternalImageId": "image1.jpg",
                "Confidence": 99.99949645996094,
                "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
            },
            "Similarity": 98.66607666015625
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5349419713020325,
                    "Top": 0.29124999046325684,
                    "Left": 0.16389399766921997,
                    "Height": 0.40187498927116394
                },
                "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
                "ExternalImageId": "image9.jpg",
                "Confidence": 99.99979400634766,
                "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
            },
            "Similarity": 98.24278259277344
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5307819843292236,
                    "Top": 0.2862499952316284,
                    "Left": 0.1564060002565384,
                    "Height": 0.3987500071525574
                },
```

```
                "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
                "ExternalImageId": "image10.jpg",
                "Confidence": 99.99970245361328,
                "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
            },
            "Similarity": 98.10665893554688
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5074880123138428,
                    "Top": 0.3774999976158142,
                    "Left": 0.18302799761295319,
                    "Height": 0.3812499940395355
                },
                "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
                "ExternalImageId": "image6.jpg",
                "Confidence": 99.99930572509766,
                "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
            },
            "Similarity": 98.10526275634766
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5574039816856384,
                    "Top": 0.37187498807907104,
                    "Left": 0.14559100568294525,
                    "Height": 0.4181250035762787
                },
                "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
                "ExternalImageId": "image5.jpg",
                "Confidence": 99.99960327148438,
                "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
            },
            "Similarity": 97.94659423828125
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5773710012435913,
                    "Top": 0.34437501430511475,
                    "Left": 0.12396000325679779,
                    "Height": 0.4337500035762787
```

```
                },
                "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
                "ExternalImageId": "image8.jpg",
                "Confidence": 100.0,
                "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
            },
            "Similarity": 97.93476867675781
        }
    ],

 "FaceModelVersion": "3.0"
}
```

Para obter mais informações, consulte Procurando um rosto com uma imagem no Guia do desenvolvedor do Amazon Rekognition.

- Para obter detalhes da API, consulte SearchFacesByImageem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();
```

```
            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SearchFacesByImagea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.
```

```
        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
```

```
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces
```

- Para obter detalhes da API, consulte a SearchFacesByImageReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       " Create S3 object reference for the image
       DATA(lo_s3object) = NEW /aws1/cl_reks3object(
         iv_bucket = iv_s3_bucket
         iv_name = iv_s3_key ).

       " Create image object
       DATA(lo_image) = NEW /aws1/cl_rekimage(
         io_s3object = lo_s3object ).

       " Search for matching faces
       oo_result = lo_rek->searchfacesbyimage(
         iv_collectionid = iv_collection_id
         io_image = lo_image
         iv_facematchthreshold = iv_threshold
         iv_maxfaces = iv_max_faces ).

       DATA(lt_face_matches) = oo_result->get_facematches( ).
       DATA(lv_match_count) = lines( lt_face_matches ).
       DATA(lv_msg4) = |Face search completed: { lv_match_count } match(es)
found.|.
       MESSAGE lv_msg4 TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalids3objectex.
       MESSAGE 'Invalid S3 object.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a SearchFacesByImagereferência da API AWS SDK for SAP ABAP.

Cenários para o Amazon Rekognition usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Rekognition com SDKs. AWS Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon Rekognition ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie uma coleção do Amazon Rekognition e encontre rostos nela usando um AWS SDK
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK
- Detecte e exiba elementos em imagens com o Amazon Rekognition usando um AWS SDK
- Detecte rostos em uma imagem usando um AWS SDK
- Detecte informações em vídeos usando o Amazon Rekognition e o AWS SDK
- Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
- Salve EXIF e outras informações da imagem usando um AWS SDK

Crie uma coleção do Amazon Rekognition e encontre rostos nela usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar uma coleção do Amazon Rekognition.
- Adicionar imagens à coleção e detectar faces nela.
- Pesquisar na coleção faces que correspondam a uma imagem de referência.
- Excluir uma coleção.

Para obter mais informações, consulte Pesquisar faces em uma coleção.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar classes que envolvam as funções do Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client
```

```
    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
 and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
 the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
```

```
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
 self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
 self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections



class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
        rendering = {
            "collection_id": self.collection_id,
            "collection_arn": self.collection_arn,
            "face_count": self.face_count,
            "created": self.created,
        }
        return rendering
```

```
    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise


    def index_faces(self, image, max_faces):
```

```
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
```

```
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
```

```
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
```

```
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces


class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
```

```
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
```

```
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering
```

Usar as classes wrapper para criar uma coleção de faces a partir de um conjunto de imagens e, em seguida, pesquisar faces na coleção.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition face collection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    rekognition_client = boto3.client("rekognition")
    images = [
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128316.jpg",
            rekognition_client,
            image_name="sitting",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128317.jpg",
            rekognition_client,
            image_name="hopping",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128318.jpg",
            rekognition_client,
            image_name="biking",
        ),
    ]

    collection_mgr = RekognitionCollectionManager(rekognition_client)
    collection = collection_mgr.create_collection("doc-example-collection-demo")
```

```
    print(f"Created collection {collection.collection_id}:")
    pprint(collection.describe_collection())

    print("Indexing faces from three images:")
    for image in images:
        collection.index_faces(image, 10)
    print("Listing faces in collection:")
    faces = collection.list_faces(10)
    for face in faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the first face in the
"
        f"list (Face ID: {faces[0].face_id}."
    )
    found_faces = collection.search_faces(faces[0].face_id, 80, 10)
    print(f"Found {len(found_faces)} matching faces.")
    for face in found_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the largest face in "
        f"{images[0].image_name}."
    )
    image_face, match_faces = collection.search_faces_by_image(images[0], 80, 10)
    print(f"The largest face in {images[0].image_name} is:")
    pprint(image_face.to_dict())
    print(f"Found {len(match_faces)} matching faces.")
    for face in match_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    collection.delete_collection()
    print("Thanks for watching!")
    print("-" * 88)
```

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detecte e exiba elementos em imagens com o Amazon Rekognition usando um AWS SDK

O exemplo de código a seguir mostra como:

- Detectar elementos em imagens usando o Amazon Rekognition.
- Exibir imagens e desenhar caixas delimitadoras ao redor dos elementos detectados.

Para obter mais informações, consulte Exibir caixas delimitadoras.

Python

SDK para Python (Boto3)

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar classes para agrupar as funções do Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace,
    RekognitionCelebrity,
    RekognitionLabel,
    RekognitionModerationLabel,
    RekognitionText,
    show_bounding_boxes,
    show_polygons,
)

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


    @classmethod
    def from_bucket(cls, s3_object, rekognition_client):
        """
        Creates a RekognitionImage object from an Amazon S3 object.

        :param s3_object: An Amazon S3 object that identifies the image. The
image
                          is not retrieved until needed for a later call.
```

```
        :param rekognition_client: A Boto3 Rekognition client.
        :return: The RekognitionImage object, initialized with Amazon S3 object
data.
        """
        image = {"S3Object": {"Bucket": s3_object.bucket_name, "Name":
s3_object.key}}
        return cls(image, s3_object.key, rekognition_client)


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
```

```
        else:
            return labels


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces



    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
greater
```

```
                            than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
```

```
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
 response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

Criar funções auxiliares para desenhar caixas delimitadoras e polígonos.

```
import io
import logging
from PIL import Image, ImageDraw
```

```
logger = logging.getLogger(__name__)


def show_bounding_boxes(image_bytes, box_sets, colors):
    """
    Draws bounding boxes on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box["Left"]
            top = image.height * box["Top"]
            right = (image.width * box["Width"]) + left
            bottom = (image.height * box["Height"]) + top
            draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()



def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param polygons: The list of polygons to draw on the image.
    :param color: The color to use to draw the polygons.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for polygon in polygons:
        draw.polygon(
            [
                (image.width * point["X"], image.height * point["Y"])
                for point in polygon
            ],
            outline=color,
        )
    image.show()
```

Criar classes para analisar objetos retornados pelo Amazon Rekognition.

```
class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp
```

```
    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering



class RekognitionCelebrity:
```

```
    """Encapsulates an Amazon Rekognition celebrity."""

    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.

        :param celebrity: Celebrity data, in the format returned by Amazon
 Rekognition
                          functions.
        :param timestamp: The time when the celebrity was detected, if the
 celebrity
                          was detected in a video.
        """
        self.info_urls = celebrity.get("Urls")
        self.name = celebrity.get("Name")
        self.id = celebrity.get("Id")
        self.face = RekognitionFace(celebrity.get("Face"))
        self.confidence = celebrity.get("MatchConfidence")
        self.bounding_box = celebrity.get("BoundingBox")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
        if self.name is not None:
            rendering["name"] = self.name
        if self.info_urls:
            rendering["info URLs"] = self.info_urls
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""

    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.
```

```
        :param person: Person data, in the format returned by Amazon Rekognition
                       functions.
        :param timestamp: The time when the person was detected, if the person
                          was detected in a video.
        """
        self.index = person.get("Index")
        self.bounding_box = person.get("BoundingBox")
        face = person.get("Face")
        self.face = RekognitionFace(face) if face is not None else None
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the person data to a dict.

        :return: A dict that contains the person data.
        """
        rendering = self.face.to_dict() if self.face is not None else {}
        if self.index is not None:
            rendering["index"] = self.index
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the label was detected, if the label
                          was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.instances = label.get("Instances")
```

```
        self.parents = label.get("Parents")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the label data to a dict.

        :return: A dict that contains the label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionModerationLabel:
    """Encapsulates an Amazon Rekognition moderation label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the moderation label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the moderation label was detected, if the
                          label was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.parent_name = label.get("ParentName")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the moderation label data to a dict.

        :return: A dict that contains the moderation label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
```

```
        if self.parent_name is not None:
            rendering["parent_name"] = self.parent_name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionText:
    """Encapsulates an Amazon Rekognition text element."""

    def __init__(self, text_data):
        """
        Initializes the text object.

        :param text_data: Text data, in the format returned by Amazon Rekognition
                          functions.
        """
        self.text = text_data.get("DetectedText")
        self.kind = text_data.get("Type")
        self.id = text_data.get("Id")
        self.parent_id = text_data.get("ParentId")
        self.confidence = text_data.get("Confidence")
        self.geometry = text_data.get("Geometry")

    def to_dict(self):
        """
        Renders some of the text data to a dict.

        :return: A dict that contains the text data.
        """
        rendering = {}
        if self.text is not None:
            rendering["text"] = self.text
        if self.kind is not None:
            rendering["kind"] = self.kind
        if self.geometry is not None:
            rendering["polygon"] = self.geometry.get("Polygon")
        return rendering
```

Use as classes wrapper para detectar elementos em imagens e exibir suas caixas delimitadoras. As imagens usadas neste exemplo podem ser encontradas GitHub junto com instruções e mais códigos.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition image detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    rekognition_client = boto3.client("rekognition")
    street_scene_file_name = ".media/pexels-kaique-rocha-109919.jpg"
    celebrity_file_name = ".media/pexels-pixabay-53370.jpg"
    one_girl_url = "https://dhei5unw3vrsx.cloudfront.net/images/
source3_resized.jpg"
    three_girls_url = "https://dhei5unw3vrsx.cloudfront.net/images/
target3_resized.jpg"
    swimwear_object = boto3.resource("s3").Object(
        "console-sample-images-pdx", "yoga_swimwear.jpg"
    )
    book_file_name = ".media/pexels-christina-morillo-1181671.jpg"

    street_scene_image = RekognitionImage.from_file(
        street_scene_file_name, rekognition_client
    )
    print(f"Detecting faces in {street_scene_image.image_name}...")
    faces = street_scene_image.detect_faces()
    print(f"Found {len(faces)} faces, here are the first three.")
    for face in faces[:3]:
        pprint(face.to_dict())
    show_bounding_boxes(
        street_scene_image.image["Bytes"],
        [[face.bounding_box for face in faces]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    print(f"Detecting labels in {street_scene_image.image_name}...")
    labels = street_scene_image.detect_labels(100)
    print(f"Found {len(labels)} labels.")
    for label in labels:
        pprint(label.to_dict())
    names = []
```

```
    box_sets = []
    colors = ["aqua", "red", "white", "blue", "yellow", "green"]
    for label in labels:
        if label.instances:
            names.append(label.name)
            box_sets.append([inst["BoundingBox"] for inst in label.instances])
    print(f"Showing bounding boxes for {names} in {colors[:len(names)]}.")
    show_bounding_boxes(
        street_scene_image.image["Bytes"], box_sets, colors[: len(names)]
    )
    input("Press Enter to continue.")

    celebrity_image = RekognitionImage.from_file(
        celebrity_file_name, rekognition_client
    )
    print(f"Detecting celebrities in {celebrity_image.image_name}...")
    celebs, others = celebrity_image.recognize_celebrities()
    print(f"Found {len(celebs)} celebrities.")
    for celeb in celebs:
        pprint(celeb.to_dict())
    show_bounding_boxes(
        celebrity_image.image["Bytes"],
        [[celeb.face.bounding_box for celeb in celebs]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    girl_image_response = requests.get(one_girl_url)
    girl_image = RekognitionImage(
        {"Bytes": girl_image_response.content}, "one-girl", rekognition_client
    )
    group_image_response = requests.get(three_girls_url)
    group_image = RekognitionImage(
        {"Bytes": group_image_response.content}, "three-girls",
rekognition_client
    )
    print("Comparing reference face to group of faces...")
    matches, unmatches = girl_image.compare_faces(group_image, 80)
    print(f"Found {len(matches)} face matching the reference face.")
    show_bounding_boxes(
        group_image.image["Bytes"],
        [[match.bounding_box for match in matches]],
        ["aqua"],
    )
```

```
    input("Press Enter to continue.")

    swimwear_image = RekognitionImage.from_bucket(swimwear_object,
rekognition_client)
    print(f"Detecting suggestive content in {swimwear_object.key}...")
    labels = swimwear_image.detect_moderation_labels()
    print(f"Found {len(labels)} moderation labels.")
    for label in labels:
        pprint(label.to_dict())
    input("Press Enter to continue.")

    book_image = RekognitionImage.from_file(book_file_name, rekognition_client)
    print(f"Detecting text in {book_image.image_name}...")
    texts = book_image.detect_text()
    print(f"Found {len(texts)} text instances. Here are the first seven:")
    for text in texts[:7]:
        pprint(text.to_dict())
    show_polygons(
        book_image.image["Bytes"], [text.geometry["Polygon"] for text in texts],
"aqua"
    )

    print("Thanks for watching!")
    print("-" * 88)
```

Detecte rostos em uma imagem usando um AWS SDK

O exemplo de código a seguir mostra como:

- Salvar uma imagem em um bucket do Amazon S3.
- Usar o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.).
- Exibir esses detalhes.

Rust

SDK para Rust

Salve a imagem em um bucket do Amazon S3 com um prefixo uploads, use o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.), e exiba esses detalhes.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3

Detecte informações em vídeos usando o Amazon Rekognition e o AWS SDK

Os exemplos de código a seguir mostram como:

- Iniciar trabalhos do Amazon Rekognition para detectar elementos como pessoas, objetos e texto em vídeos.
- Verificar o status do trabalho até que os trabalhos terminem.
- Visualizar a lista de elementos detectados por cada trabalho.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha resultados de celebridades a partir de um vídeo localizado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-
sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
```

```
                    bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                    video - The name of video (for example, people.mp4).\s
                    topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                    roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startCelebrityDetection(rekClient, channel, bucket, video);
        getCelebrityDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startCelebrityDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();
```

```
            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
                    .startCelebrityRecognition(recognitionRequest);
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCelebrityDetectionResults(RekognitionClient rekClient)
{
        try {
            String paginationToken = null;
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (recognitionResponse != null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
```

```
                while (!finished) {
                    recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs =
recognitionResponse.celebrities();
                for (CelebrityRecognition celeb : celebs) {
                    long seconds = celeb.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details = celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

Detecte rótulos em um vídeo por meio de uma operação de detecção de rótulos.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
```

```
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
```

```
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
```

```
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
```

```
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels =
labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
```

```
                            System.out.println("        Confidence: " +
 instance.confidence().toString());
                            System.out.println("        Bounding box: " +
 instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
 ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Detecte faces em um vídeo armazenado em um bucket do Amazon S3.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
```

```
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
```

```
                    roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();
```

```
            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
```

```
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
```

```
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
```

```
                VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels =
labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
```

```
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Detecte conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
 software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }
```

```
    public static void startModerationDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {

        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();
```

```
                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }
```

```
            } while (modDetectionResponse != null &&
 modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte segmentos de sinal técnico e segmentos de detecção de tomada em um vídeo armazenado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startSegmentDetection(rekClient, channel, bucket, video);
        getSegmentResults(rekClient);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startSegmentDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
                    .shotFilter(cueDetectionFilter)
                    .technicalCueFilter(technicalCueDetectionFilter)
                    .build();
```

```
            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                    .video(vidOb)
                    .filters(filters)
                    .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getSegmentResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();
```

```
                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }

                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
                        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
                    }

                    long seconds = detectedSegment.durationMillis();
```

```
                    System.out.println("\tDuration : " + seconds + "
 milliseconds");
                    System.out.println("\tStart time code: " +
 detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
 detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
 detectedSegment.durationSMPTE());
                    System.out.println();
                }

            } while (segDetectionResponse != null &&
 segDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte texto em um vídeo armazenado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();
```

```
        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
```

```
            int yy = 0;

            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
```

```
                    System.out.println("Confidence: " +
 detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " +
 detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
 detectedText.textDetection().parentId());
                    System.out.println("Type: " +
 detectedText.textDetection().type());
                    System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                    System.out.println();
                }

            } while (textDetectionResponse != null &&
 textDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte pessoas em um vídeo armazenado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
```

```
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
```

```
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
```

```
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
 personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
 detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
 personTrackingResult.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Detecte faces em um vídeo armazenado em um bucket do Amazon S3.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
```

```
    var status: String
    var yy = 0
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Detecte conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket do Amazon S3.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }
```

```
        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration

- StartLabelDetection
- StartPersonTracking
- StartSegmentDetection
- StartTextDetection

Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Rekognition AWS SDK para JavaScript com o para criar um aplicativo que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Aprenda como:

- Criar um usuário não autenticado usando o Amazon Cognito.
- Analisar imagens em busca de objetos usando o Amazon Rekognition.
- Verificar um endereço de e-mail para o Amazon SES.
- Enviar uma notificação por e-mail usando o Amazon SES.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um aplicativo web que permite fazer o seguinte:

- Carregar fotos em um bucket do Amazon Simple Storage Service (Amazon S3).
- Usar o Amazon Rekognition para analisar e rotular as fotos.
- Usar o Amazon Simple Email Service (Amazon SES) para enviar relatórios de análise da imagem por e-mail.

Este exemplo contém dois componentes principais: uma página da web escrita com React e um serviço REST escrito em Python JavaScript que é construído com. Flask-RESTful

Você pode usar a página da Web do React para:

- Exibir uma lista de imagens que estão armazenadas no bucket do S3.
- Carregar imagens do computador para o bucket do S3.
- Exibir imagens e rótulos que identificam os itens detectados na imagem.
- Obter um relatório de todas as imagens no bucket do S3 e enviar um relatório por e-mail.

A página da Web chama o serviço REST. O serviço envia solicitações à AWS para realizar as seguintes ações:

- Obter e filtrar a lista de imagens no bucket do S3.
- Carregar fotos no bucket do S3.
- Usar o Amazon Rekognition para analisar fotos individuais e obter uma lista dos rótulos que identifiquem os itens detectados nas fotos.
- Analisar todas as fotos no bucket do S3 e usar o Amazon SES para enviar um relatório por e-mail.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition

- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Salve EXIF e outras informações da imagem usando um AWS SDK

O exemplo de código a seguir mostra como:

- Obter informações de EXIF de um arquivo JPG, JPEG ou PNG.
- Fazer upload do arquivo de imagem para um bucket do Amazon S3.
- Usar o Amazon Rekognition para identificar os três principais atributos (rótulos) no arquivo.
- Adicionar as informações de EXIF e rótulo a uma tabela do Amazon DynamoDB na região.

Rust

SDK para Rust

Obtenha informações de EXIF de um arquivo JPG, JPEG ou PNG, faça upload do arquivo de imagem para um bucket do Amazon S3, use o Amazon Rekognition para identificar os três principais atributos (rótulos no Amazon Rekognition) no arquivo e adicione as informações de EXIF e de rótulo a uma tabela do Amazon DynamoDB na região.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Exemplos de código para Resource Groups usando AWS SDKs

Os exemplos de código a seguir mostram como usar Grupos de recursos da AWS com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário de Resource Groups: mais informações sobre Resource Groups.
- Referência de API de Resource Groups: detalhes sobre todas as ações disponíveis de Resource Groups.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para Resource Groups usando AWS SDKs
  - Ações para Resource Groups usando AWS SDKs
    - Usar CreateGroup com uma CLI
    - Usar DeleteGroup com uma CLI
    - Usar GetGroup com uma CLI
    - Usar GetGroupQuery com uma CLI
    - Usar GetTags com uma CLI
    - Usar ListGroupResources com uma CLI
    - Usar ListGroups com uma CLI
    - Usar SearchResources com uma CLI
    - Use a tag com uma CLI
    - Use Untag com uma CLI
    - Usar UpdateGroup com uma CLI
    - Usar UpdateGroupQuery com uma CLI

Exemplos básicos para Resource Groups usando AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do Grupos de recursos da AWS com AWS SDKs.

Exemplos

- Ações para Resource Groups usando AWS SDKs
  - Usar CreateGroup com uma CLI
  - Usar DeleteGroup com uma CLI
  - Usar GetGroup com uma CLI
  - Usar GetGroupQuery com uma CLI
  - Usar GetTags com uma CLI
  - Usar ListGroupResources com uma CLI
  - Usar ListGroups com uma CLI
  - Usar SearchResources com uma CLI
  - Use a tag com uma CLI
  - Use Untag com uma CLI

- Usar UpdateGroup com uma CLI
- Usar UpdateGroupQuery com uma CLI

Ações para Resource Groups usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Resource Groups com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do Grupos de recursos da AWS.

Exemplos

- Usar CreateGroup com uma CLI
- Usar DeleteGroup com uma CLI
- Usar GetGroup com uma CLI
- Usar GetGroupQuery com uma CLI
- Usar GetTags com uma CLI
- Usar ListGroupResources com uma CLI
- Usar ListGroups com uma CLI
- Usar SearchResources com uma CLI
- Use a tag com uma CLI
- Use Untag com uma CLI
- Usar UpdateGroup com uma CLI
- Usar UpdateGroupQuery com uma CLI

Usar **`CreateGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateGroup`.

CLI

AWS CLI

Exemplo 1: criar um grupo de recursos baseado em tags

O exemplo `create-group` a seguir cria um grupo de recursos baseado em tag de instâncias do Amazon EC2 na Região atual. Ele é baseado em uma consulta de recursos marcados com a chave `Name` e o valor `WebServers`. O nome do grupo é `tbq-WebServer`. A consulta está em um arquivo JSON separado que é passado para o comando.

```
aws resource-groups create-group \
    --name tbq-WebServer \
    --resource-query file://query.json
```

Conteúdo de `query.json`:

```
{
    "Type": "TAG_FILTERS_1_0",
    "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],\"TagFilters\":
[{\"Key\":\"Name\", \"Values\":[\"WebServers\"]}]}"
}
```

Saída:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-
WebServer",
        "Name": "tbq-WebServer"
    },
    "ResourceQuery": {
        "Type": "TAG_FILTERS_1_0",
        "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],\"TagFilters
\":[{\"Key\":\"Name\", \"Values\":[\"WebServers\"]}]}"
    }
}
```

Exemplo 2: Para criar um grupo de recursos CloudFormation baseado em pilhas

O `create-group` exemplo a seguir cria um grupo de recursos AWS CloudFormation baseado em pilhas chamado. `sampleCFNstackgroup` A consulta inclui todos os recursos na CloudFormation pilha especificada que são compatíveis com AWS Resource Groups.

```
aws resource-groups create-group \
    --name cbq-CFNstackgroup \
    --resource-query file://query.json
```

Conteúdo de `query.json`:

```
{
    "Type": "CLOUDFORMATION_STACK_1_0",
    "Query": "{\"ResourceTypeFilters\":[\"AWS::AllSupported\"],\"StackIdentifier
\":\"arn:aws:cloudformation:us-west-2:123456789012:stack/MyCFNStack/1415z9z0-
z39z-11z8-97z5-500z212zz6fz\"}"
}
```

Saída:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/cbq-
CFNstackgroup",
        "Name": "cbq-CFNstackgroup"
    },
    "ResourceQuery": {
        "Type": "CLOUDFORMATION_STACK_1_0",
        "Query": "{\"ResourceTypeFilters\":[\"AWS::AllSupported\"],
\"StackIdentifier\":\"arn:aws:cloudformation:us-east-2:123456789012:stack/
MyCFNStack/1415z9z0-z39z-11z8-97z5-500z212zz6fz\"}"}'
    }
}
```

Para obter mais informações, consulte Criar grupos no Guia do usuário do AWS Resource Groups.

- Para obter detalhes da API, consulte CreateGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria um novo grupo de AWS recursos de Resource Groups baseado em tags chamado TestPowerShellGroup. O grupo inclui instâncias do Amazon EC2 na região atual que são marcadas com a chave de tag “Name” e o valor da tag “test2". O comando retorna a consulta, o tipo de grupo e os resultados da operação.

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
$ResourceQuery.Type = "TAG_FILTERS_1_0"
```

```
$ResourceQuery.Query = '{"ResourceTypeFilters":
["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":["test2"]}]}'
$ResourceQuery

New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
 "Test resource group."
```

Saída:

```
Query
               Type
-----
               ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"Name","Values":["test2"]}]} TAG_FILTERS_1_0

LoggedAt         : 11/20/2018 2:40:59 PM
Group            : Amazon.ResourceGroups.Model.Group
ResourceQuery    : Amazon.ResourceGroups.Model.ResourceQuery
Tags             : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 338
HttpStatusCode   : OK
```

- Para obter detalhes da API, consulte CreateGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um novo grupo de AWS recursos de Resource Groups baseado em tags chamado TestPowerShellGroup. O grupo inclui instâncias do Amazon EC2 na região atual que são marcadas com a chave de tag “Name” e o valor da tag “test2". O comando retorna a consulta, o tipo de grupo e os resultados da operação.

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
$ResourceQuery.Type = "TAG_FILTERS_1_0"
$ResourceQuery.Query = '{"ResourceTypeFilters":
["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":["test2"]}]}'
$ResourceQuery

New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
 "Test resource group."
```

Saída:

```
Query
               Type
-----
               ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"Name","Values":["test2"]}]} TAG_FILTERS_1_0

LoggedAt         : 11/20/2018 2:40:59 PM
Group            : Amazon.ResourceGroups.Model.Group
ResourceQuery    : Amazon.ResourceGroups.Model.ResourceQuery
Tags             : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 338
HttpStatusCode   : OK
```

- Para obter detalhes da API, consulte CreateGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteGroup`.

CLI

AWS CLI

Atualizar a descrição de um grupo de recursos

O exemplo `delete-group` a seguir atualiza o grupo de recursos especificado.

```
aws resource-groups delete-group \
    --group-name tbq-WebServer
```

Saída:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:1234567890:group/tbq-
WebServer",
        "Name": "tbq-WebServer"
```

```
    }
}
```

Para obter mais informações, consulte Excluir Grupos no Guia do usuário do AWS Resource Groups.

- Para obter detalhes da API, consulte DeleteGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove o grupo de recursos nomeado

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Description GroupArn
  Name
----------- --------
  ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-
elbv2 non-tag-cfn-elbv2
```

- Para obter detalhes da API, consulte DeleteGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o grupo de recursos nomeado

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Description GroupArn
  Name
----------- --------
  ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-
elbv2 non-tag-cfn-elbv2
```

- Para obter detalhes da API, consulte DeleteGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetGroup`.

CLI

AWS CLI

Para obter informações sobre um grupo de recursos

O exemplo `get-group` a seguir exibe os detalhes do grupo de destino especificado. Para anexar a consulta ao grupo, use `get-group-query`.

```
aws resource-groups get-group \
    --group-name tbq-WebServer
```

Saída:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-
WebServer",
        "Name": "tbq-WebServer",
        "Description": "A tag-based query resource group of WebServers."
```

```
    }
}
```

- Para obter detalhes da API, consulte GetGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo recupera o grupo de recursos de acordo com o nome do grupo

```
Get-RGGroup -GroupName auto-no
```

Saída:

```
Description GroupArn                                                  Name
----------- --------                                                  ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Para obter detalhes da API, consulte GetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera o grupo de recursos de acordo com o nome do grupo

```
Get-RGGroup -GroupName auto-no
```

Saída:

```
Description GroupArn                                                  Name
----------- --------                                                  ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Para obter detalhes da API, consulte GetGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetGroupQuery`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetGroupQuery`.

CLI

AWS CLI

Para anexar a consulta a um grupo de recursos

O exemplo `get-group-query` a seguir exibe uma consulta anexada ao grupo de recursos especificado.

```
aws resource-groups get-group-query \
    --group-name tbq-WebServer
```

Saída:

```
{
    "GroupQuery": {
        "GroupName": "tbq-WebServer",
        "ResourceQuery": {
            "Type": "TAG_FILTERS_1_0",
            "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],
\"TagFilters\":[{\"Key\":\"Name\", \"Values\":[\"WebServers\"]}]}"
        }
    }
}
```

- Para obter detalhes da API, consulte GetGroupQueryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo busca a consulta de recursos para determinado grupo de recursos

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Saída:

```
Query
            Type
```

```
-----
            ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"auto","Values":["no"]}]} TAG_FILTERS_1_0
```

- Para obter detalhes da API, consulte GetGroupQueryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo busca a consulta de recursos para determinado grupo de recursos

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Saída:

```
Query
            Type
-----
            ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"auto","Values":["no"]}]} TAG_FILTERS_1_0
```

- Para obter detalhes da API, consulte GetGroupQueryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetTags`.

CLI

AWS CLI

Para recuperar as tags anexadas a um grupo de recursos

O exemplo `get-tags` a seguir exibe os pares de chave e valor de tag anexados ao grupo de recursos especificado (o grupo em si, não seus membros).

```
aws resource-groups get-tags \
    --arn arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer
```

Saída:

```
{
    "Arn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer",
    "Tags": {
        "QueryType": "tags",
        "QueryResources": "ec2-instances"
    }
}
```

- Para obter detalhes da API, consulte GetTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista tags para o ARN do grupo de recursos fornecido

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Saída:

```
Key       Value
---       -----
Instances workboxes
```

- Para obter detalhes da API, consulte GetTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista tags para o ARN do grupo de recursos fornecido

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Saída:

```
Key       Value
---       -----
```

```
Instances workboxes
```

- Para obter detalhes da API, consulte GetTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListGroupResources`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListGroupResources`.

CLI

AWS CLI

Para listar todos os recursos em um grupo de recursos

Exemplo 1: o exemplo `list-resource-groups` a seguir lista todos os recursos que fazem parte do grupo de recursos especificado.

```
aws resource-groups list-group-resources \
    --group-name tbq-WebServer
```

Saída:

```
{
    "ResourceIdentifiers": [
        {
            "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/
i-09f77fa38c12345ab",
            "ResourceType": "AWS::EC2::Instance"
        }
    ]
}
```

Exemplo 2: o exemplo a seguir lista todos os recursos do grupo que também têm um 'resource-type' igual a 'AWS::EC2::Instance'. :

aws resource-groups list-group-resources --group-name tbq- --filters name=Tipo de recurso, Values=: :EC2: :Instance WebServer AWS

- Para obter detalhes da API, consulte ListGroupResourcesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista os recursos do grupo com base no filtro por tipo de recurso

```
Get-RGGroupResourceList -Filter @{Name="resource-
type";Values="AWS::EC2::Instance"} -GroupName auto-yes | Select-Object -
ExpandProperty ResourceIdentifiers
```

Saída:

```
ResourceArn                                                     ResourceType
-----------                                                     ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012
 AWS::EC2::Instance
```

- Para obter detalhes da API, consulte ListGroupResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os recursos do grupo com base no filtro por tipo de recurso

```
Get-RGGroupResourceList -Filter @{Name="resource-
type";Values="AWS::EC2::Instance"} -GroupName auto-yes | Select-Object -
ExpandProperty ResourceIdentifiers
```

Saída:

```
ResourceArn                                                     ResourceType
-----------                                                     ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc
 AWS::EC2::Instance
```

```
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012
 AWS::EC2::Instance
```

- Para obter detalhes da API, consulte ListGroupResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListGroups`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListGroups`.

CLI

AWS CLI

Para listar os grupos de recursos disponíveis

O exemplo `list-groups` a seguir exibe uma lista de todos os grupos de recursos.

```
aws resource-groups list-groups
```

Saída:

```
{
    "GroupIdentifiers": [
        {
            "GroupName": "tbq-WebServer",
            "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/
tbq-WebServer3"
        },
        {
            "GroupName": "cbq-CFNStackQuery",
            "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/
cbq-CFNStackQuery"
        }
    ],
    "Groups": [
        {
            "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/
tbq-WebServer",
            "Name": "tbq-WebServer"
```

```
        },
        {
            "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/
cbq-CFNStackQuery",
            "Name": "cbq-CFNStackQuery"
        }
    ]
}
```

- Para obter detalhes da API, consulte ListGroupsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista o grupo de recursos já criado.

```
Get-RGGroupList
```

Saída:

```
GroupArn
 GroupName
--------
 ---------
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no             auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes            auto-
yes
arn:aws:resource-groups:eu-west-1:123456789012:group/build600
 build600
```

- Para obter detalhes da API, consulte ListGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista o grupo de recursos já criado.

```
Get-RGGroupList
```

Saída:

```
GroupArn
 GroupName
--------
 ---------
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no              auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes             auto-
yes
arn:aws:resource-groups:eu-west-1:123456789012:group/build600
 build600
```

- Para obter detalhes da API, consulte ListGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`SearchResources`** com uma CLI

Os exemplos de código a seguir mostram como usar o `SearchResources`.

CLI

AWS CLI

Para encontrar recursos que correspondam a uma consulta

O `search-resources` exemplo a seguir recupera uma lista de todos os AWS recursos que correspondem à consulta especificada.

```
aws resource-groups search-resources \
    --resource-query file://query.json
```

Conteúdo de `query.json`:

```
{
    "Type": "TAG_FILTERS_1_0",
    "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],\"TagFilters\":
[{\"Key\":\"Patch Group\", \"Values\":[\"Dev\"]}]}"
}
```

Saída:

```
{
```

```
    "ResourceIdentifiers": [
        {
            "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/
i-01a23bc45d67890ef",
            "ResourceType": "AWS::EC2::Instance"
        }
    ]
}
```

- Para obter detalhes da API, consulte SearchResourcesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo cria um tipo de recurso ResourceQuery for Instance com filtros de tag e encontra recursos.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
    Key = 'auto'
    Values = @('no')
  })
 }

Find-RGResource -ResourceQuery $query | Select-Object -ExpandProperty
 ResourceIdentifiers
```

Saída:

```
ResourceArn                                                    ResourceType
-----------                                                    ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b
 AWS::EC2::Instance
```

- Para obter detalhes da API, consulte SearchResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um tipo de recurso ResourceQuery for Instance com filtros de tag e encontra recursos.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
    Key = 'auto'
    Values = @('no')
  })
 }

Find-RGResource -ResourceQuery $query | Select-Object -ExpandProperty
 ResourceIdentifiers
```

Saída:

```
ResourceArn                                                        ResourceType
-----------                                                        ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b
 AWS::EC2::Instance
```

- Para obter detalhes da API, consulte SearchResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use a **`tag`** com uma CLI

Os exemplos de código a seguir mostram como usar o `Tag`.

CLI

AWS CLI

Para anexar uma tag a um grupo de recursos

O exemplo `tag` a seguir anexa os pares de chave e valor de tag especificados ao grupo de recursos especificado (o grupo em si, não seus membros).

```
aws resource-groups tag \
```

```
    --tags QueryType=tags,QueryResources=ec2-instances \
    --arn arn:aws:resource-groups:us-west-2:128716708097:group/tbq-WebServer
```

Saída:

```
{
    "Arn": "arn:aws:resource-groups:us-west-2:128716708097:group/tbq-WebServer",
    "Tags": {
        "QueryType": "tags",
        "QueryResources": "ec2-instances"
    }
}
```

Para obter mais informações, consulte Como gerenciar tags no Guia do usuário do AWS Resource Groups.

- Para ver detalhes da API, consulte Tag na Referência de comandos da AWS CLI .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona a chave de tag “Instances” com o valor “workboxes” ao ARN do grupo de recursos fornecido

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Saída:

```
Arn                                                          Tags
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Consulte detalhes da API em Tag na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a chave de tag “Instances” com o valor “workboxes” ao ARN do grupo de recursos fornecido

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Saída:

```
Arn                                                          Tags
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Consulte detalhes da API em Tag na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Use **`Untag`** com uma CLI

Os exemplos de código a seguir mostram como usar o `Untag`.

CLI

AWS CLI

Para remover tags de um grupo de recursos

O exemplo `untags` a seguir remove qualquer tag com a chave especificada do grupo de recursos em si, não seus membros.

```
aws resource-groups untag \
    --arn arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer \
    --keys QueryType
```

Saída:

```
{
    "Arn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer",
    "Keys": [
        "QueryType"
    ]
}
```

Para obter mais informações, consulte Como gerenciar tags no Guia do usuário do AWS Resource Groups.

- Para ver detalhes da API, consulte Untag na Referência de comandos da AWS CLI .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a tag mencionada do grupo de recursos

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGResourceTag (Untag)" on target
 "arn:aws:resource-groups:eu-west-1:933303704102:group/workboxes".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                                                          Keys
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- Consulte detalhes da API em Untag na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag mencionada do grupo de recursos

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGResourceTag (Untag)" on target
 "arn:aws:resource-groups:eu-west-1:933303704102:group/workboxes".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                                                          Keys
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- Consulte detalhes da API em Untag na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Usar **`UpdateGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateGroup`.

CLI

AWS CLI

Atualizar a descrição de um grupo de recursos

O exemplo `update-group` a seguir atualiza a descrição do grupo de recursos especificado.

```
aws resource-groups update-group \
    --group-name tbq-WebServer \
    --description "Resource group for all web server resources."
```

Saída:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-
WebServer",
        "Name": "tbq-WebServer"
        "Description": "Resource group for all web server resources."
    }
}
```

Para obter mais informações, consulte Atualizar grupos no Guia do usuário do AWS Resource Groups.

- Para obter detalhes da API, consulte UpdateGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza a descrição do grupo

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Saída:

```
Description             GroupArn
     Name
-----------             --------
     ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/
auto-yes auto-yes
```

- Para obter detalhes da API, consulte UpdateGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza a descrição do grupo

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Saída:

```
Description             GroupArn
     Name
-----------             --------
     ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/
auto-yes auto-yes
```

- Para obter detalhes da API, consulte UpdateGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateGroupQuery`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateGroupQuery`.

CLI

AWS CLI

Exemplo 1: atualizar a consulta para um grupo de recursos baseado em tags

O exemplo `update-group-query` a seguir atualiza a consulta anexada ao grupo de recursos baseado em tags especificado.

```
aws resource-groups update-group-query \
    --group-name tbq-WebServer \
    --resource-query '{"Type":"TAG_FILTERS_1_0", "Query":"{\"ResourceTypeFilters
\":[\"AWS::EC2::Instance\"],\"TagFilters\":[{\"Key\":\"Name\", \"Values\":
[\"WebServers\"]}]}"}'
```

Saída:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-east-2:123456789012:group/tbq-
WebServer",
        "Name": "tbq-WebServer"
    },
    "ResourceQuery": {
        "Type": "TAG_FILTERS_1_0",
        "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],\"TagFilters
\":[{\"Key\":\"Name\", \"Values\":[\"WebServers\"]}]}"
    }
}
```

Para obter mais informações, consulte Atualizar grupos no Guia do usuário do AWS Resource Groups.

Exemplo 2: Para atualizar a consulta de um grupo de recursos CloudFormation baseado em pilhas

O `update-group-query` exemplo a seguir atualiza a consulta anexada ao grupo de recursos AWS CloudFormation baseado em pilha especificado.

```
aws resource-groups update-group-query \
    --group-name cbq-CFNstackgroup \
```

```
    --resource-query '{"Type": "CLOUDFORMATION_STACK_1_0", "Query":
 "{\"ResourceTypeFilters\":[\"AWS::AllSupported\"],\"StackIdentifier\":
\"arn:aws:cloudformation:us-west-2:123456789012:stack/MyCFNStack/1415z9z0-
z39z-11z8-97z5-500z212zz6fz\"}"}'
```

Saída:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/cbq-
CFNstackgroup",
        "Name": "cbq-CFNstackgroup"
    },
    "ResourceQuery": {
        "Type": "CLOUDFORMATION_STACK_1_0",
        "Query": "{\"ResourceTypeFilters\":[\"AWS::AllSupported\"],
\"StackIdentifier\":\"arn:aws:cloudformation:us-west-2:123456789012:stack/
MyCFNStack/1415z9z0-z39z-11z8-97z5-500z212zz6fz\"}"
    }
}
```

Para obter mais informações, consulte Atualizar grupos no Guia do usuário do AWS Resource Groups.

- Para obter detalhes da API, consulte UpdateGroupQueryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um objeto de consulta e atualiza a consulta para o grupo.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
  Key='Environment'
  Values='Build600.11'
  })
} | ConvertTo-Json -Compress -Depth 4
```

```
Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

Saída:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Para obter detalhes da API, consulte UpdateGroupQueryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um objeto de consulta e atualiza a consulta para o grupo.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
  Key='Environment'
  Values='Build600.11'
  })
} | ConvertTo-Json -Compress -Depth 4

Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

Saída:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Para obter detalhes da API, consulte UpdateGroupQueryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para a API Resource Groups Tagging usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Resource Groups Tagging API com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário da API de marcação com tags de Resource Groups: mais informações sobre a API de marcação com tags de Resource Groups.
- Referência de API da API de marcação com tags de Resource Groups: detalhes sobre todas as ações disponíveis da API de marcação com tags de Resource Groups.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos da API Resource Groups Tagging usando AWS SDKs
  - Ações para a API Resource Groups Tagging usando AWS SDKs
    - Usar GetResources com uma CLI
    - Usar GetTagKeys com uma CLI
    - Usar GetTagValues com uma CLI
    - Usar TagResources com uma CLI
    - Usar UntagResources com uma CLI

Exemplos básicos da API Resource Groups Tagging usando AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Resource Groups Tagging API com AWS SDKs.

Exemplos

- Ações para a API Resource Groups Tagging usando AWS SDKs
  - Usar GetResources com uma CLI
  - Usar GetTagKeys com uma CLI
  - Usar GetTagValues com uma CLI
  - Usar TagResources com uma CLI
  - Usar UntagResources com uma CLI

Ações para a API Resource Groups Tagging usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais da API Resource Groups Tagging com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Resource Groups Tagging API.

Exemplos

- Usar GetResources com uma CLI
- Usar GetTagKeys com uma CLI
- Usar GetTagValues com uma CLI
- Usar TagResources com uma CLI
- Usar UntagResources com uma CLI

Usar **`GetResources`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetResources`.

CLI

AWS CLI

Obter uma lista de recursos marcados

O exemplo `get-resources` a seguir exibe uma lista de recursos na conta que estão marcados com o nome e o valor da chave especificados.

```
aws resourcegroupstaggingapi get-resources \
    --tag-filters Key=Environment,Values=Production \
    --tags-per-page 100
```

Saída:

```
{
    "ResourceTagMappingList": [
        {
            "ResourceARN": " arn:aws:inspector:us-west-2:123456789012:target/0-
nvgVhaxX/template/0-7sbz2Kz0",
            "Tags": [
                {
                    "Key": "Environment",
                    "Value": "Production"
                }
            ]
        }
    ]
}
```

Para obter mais informações, consulte a Referência GetResourcesda API Resource Groups Tagging.

- Para obter detalhes da API, consulte GetResourcesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna todos os recursos marcados em uma região e as chaves de tag associadas ao recurso. Se nenhum parâmetro -Region for fornecido ao cmdlet, ele tentará inferir a região pelo shell ou pelos metadados da instância do EC2.

```
Get-RGTResource
```

Saída:

```
ResourceARN                                                    Tags
-----------                                                    ----
```

```
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}

arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Exemplo 2: retorna todos os recursos marcados do tipo especificado em uma região. A string para cada nome de serviço e tipo de recurso é a mesma incorporada ao nome do recurso da Amazon (ARN) de um recurso.

```
Get-RGTResource -ResourceType "s3"
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Exemplo 3: retorna todos os recursos marcados do tipo especificado em uma região. Observe que quando os tipos de recursos são canalizados para o cmdlet, uma chamada para o serviço é feita para cada tipo de recurso fornecido.

```
"dynamodb","s3" | Get-RGTResource
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}

arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Exemplo 4: retorna todos os recursos marcados que correspondem ao filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Saída:

```
ResourceARN                                                     Tags
```

```
-----------                                                    ----
arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Exemplo 5: retorna todos os recursos marcados que correspondem ao filtro e ao tipo de recurso especificados.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Saída:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

Exemplo 6: retorna todos os recursos marcados que correspondem ao filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Saída:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

- Para obter detalhes da API, consulte GetResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna todos os recursos marcados em uma região e as chaves de tag associadas ao recurso. Se nenhum parâmetro -Region for fornecido ao cmdlet, ele tentará inferir a região pelo shell ou pelos metadados da instância do EC2.

```
Get-RGTResource
```

Saída:

```
ResourceARN                                                    Tags
-----------                                                    ----
```

```
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}

arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Exemplo 2: retorna todos os recursos marcados do tipo especificado em uma região. A string para cada nome de serviço e tipo de recurso é a mesma incorporada ao nome do recurso da Amazon (ARN) de um recurso.

```
Get-RGTResource -ResourceType "s3"
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Exemplo 3: retorna todos os recursos marcados do tipo especificado em uma região. Observe que quando os tipos de recursos são canalizados para o cmdlet, uma chamada para o serviço é feita para cada tipo de recurso fornecido.

```
"dynamodb","s3" | Get-RGTResource
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}

arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Exemplo 4: retorna todos os recursos marcados que correspondem ao filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Saída:

```
ResourceARN                                                     Tags
```

```
-----------                                                    ----
arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Exemplo 5: retorna todos os recursos marcados que correspondem ao filtro e ao tipo de recurso especificados.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Saída:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

Exemplo 6: retorna todos os recursos marcados que correspondem ao filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Saída:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

- Para obter detalhes da API, consulte GetResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetTagKeys`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetTagKeys`.

CLI

AWS CLI

Obter uma lista de todas as chaves de tag

O exemplo `get-tag-keys` a seguir recupera a lista de todos os nomes de chave de tag usados pelos recursos na conta.

```
aws resourcegroupstaggingapi get-tag-keys
```

Saída:

```
{
    "TagKeys": [
        "Environment",
        "CostCenter",
        "Department"
    ]
}
```

Para obter mais informações, consulte a Referência GetTagKeysda API Resource Groups Tagging.

- Para obter detalhes da API, consulte GetTagKeysem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna todas as chaves de tag na região especificada. Se o parâmetro -Region não for especificado, o cmdlet tentará inferir a região por meio da região padrão do shell ou dos metadados da instância do EC2. Observe que as chaves da tag não são retornadas em nenhuma ordem específica.

```
Get-RGTTagKey -region us-west-2
```

Saída:

```
version
stage
```

- Para obter detalhes da API, consulte GetTagKeysem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna todas as chaves de tag na região especificada. Se o parâmetro -Region não for especificado, o cmdlet tentará inferir a região por meio da região padrão do shell ou

dos metadados da instância do EC2. Observe que as chaves da tag não são retornadas em nenhuma ordem específica.

```
Get-RGTTagKey -region us-west-2
```

Saída:

```
version
stage
```

- Para obter detalhes da API, consulte GetTagKeysem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetTagValues`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetTagValues`.

CLI

AWS CLI

Obter uma lista de todos os valores de tag

O exemplo `get-tag-values` a seguir exibe todos os valores usados para a chave especificada para todos os recursos no

```
aws resourcegroupstaggingapi get-tag-values \
    --key=Environment
```

Saída:

```
{
    "TagValues": [
        "Alpha",
        "Gamma",
        "Production"
    ]
}
```

Para obter mais informações, consulte a Referência GetTagValuesda API Resource Groups Tagging.

- Para obter detalhes da API, consulte GetTagValuesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna o valor da tag especificada em uma região. Se o parâmetro -Region não for especificado, o cmdlet tentará inferir a região por meio da região padrão do shell ou dos metadados da instância do EC2.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Saída:

```
beta
```

- Para obter detalhes da API, consulte GetTagValuesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna o valor da tag especificada em uma região. Se o parâmetro -Region não for especificado, o cmdlet tentará inferir a região por meio da região padrão do shell ou dos metadados da instância do EC2.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Saída:

```
beta
```

- Para obter detalhes da API, consulte GetTagValuesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`TagResources`** com uma CLI

Os exemplos de código a seguir mostram como usar o `TagResources`.

CLI

AWS CLI

Adicionar uma tag a um recurso

O exemplo `tag-resources` a seguir adiciona tags com o nome de chave e o valor à visualização especificada.

```
aws resourcegroupstaggingapi tag-resources \
    --resource-arn-list arn:aws:s3:::MyProductionBucket \
    --tags Environment=Production,CostCenter=1234
```

Saída:

```
{
    "FailedResourcesMap": {}
}
```

Para obter mais informações, consulte a Referência TagResourcesda API Resource Groups Tagging.

- Para obter detalhes da API, consulte TagResourcesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona as chaves de tag “stage” e “version” com os valores “beta” e “preprod_test” a um bucket do Amazon S3 e a uma tabela do Amazon DynamoDB. Uma única chamada será feita para o serviço a fim de aplicar as tags.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

Exemplo 2: esse exemplo adiciona as tags e os valores especificados a um bucket do Amazon S3 e a uma tabela do Amazon DynamoDB. Duas chamadas são feitas para o serviço, uma para cada ARN de recurso canalizado para o cmdlet.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

- Para obter detalhes da API, consulte TagResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona as chaves de tag “stage” e “version” com os valores “beta” e “preprod_test” a um bucket do Amazon S3 e a uma tabela do Amazon DynamoDB. Uma única chamada será feita para o serviço a fim de aplicar as tags.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

Exemplo 2: esse exemplo adiciona as tags e os valores especificados a um bucket do Amazon S3 e a uma tabela do Amazon DynamoDB. Duas chamadas são feitas para o serviço, uma para cada ARN de recurso canalizado para o cmdlet.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

- Para obter detalhes da API, consulte TagResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UntagResources`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UntagResources`.

CLI

AWS CLI

Para remover uma tag de um recurso

O exemplo `untag-resources` a seguir remove as chaves de tag especificadas e seus valores associados do recurso especificado.

```
aws resourcegroupstaggingapi untag-resources \
    --resource-arn-list arn:aws:s3:::amzn-s3-demo-bucket \
    --tag-keys Environment CostCenter
```

Saída:

```
{
    "FailedResourcesMap": {}
}
```

Para obter mais informações, consulte a Referência UntagResourcesda API Resource Groups Tagging.

- Para obter detalhes da API, consulte UntagResourcesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: remove as chaves de tag “stage” e “version” e os valores associados de um bucket do Amazon S3 e de uma tabela do Amazon DynamoDB. Uma única chamada será feita para o serviço a fim de remover as tags. Antes que as tags sejam removidas, o cmdlet solicitará a confirmação. Para ignorar a confirmação, adicione o parâmetro -Force.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Exemplo 2: remove as chaves de tag “stage” e “version” e os valores associados de um bucket do Amazon S3 e de uma tabela do Amazon DynamoDB. Duas chamadas são feitas

para o serviço, uma para cada ARN de recurso canalizado para o cmdlet. Antes de cada chamada, o cmdlet solicitará a confirmação. Para ignorar a confirmação, adicione o parâmetro -Force.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Para obter detalhes da API, consulte UntagResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: remove as chaves de tag “stage” e “version” e os valores associados de um bucket do Amazon S3 e de uma tabela do Amazon DynamoDB. Uma única chamada será feita para o serviço a fim de remover as tags. Antes que as tags sejam removidas, o cmdlet solicitará a confirmação. Para ignorar a confirmação, adicione o parâmetro -Force.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Exemplo 2: remove as chaves de tag “stage” e “version” e os valores associados de um bucket do Amazon S3 e de uma tabela do Amazon DynamoDB. Duas chamadas são feitas para o serviço, uma para cada ARN de recurso canalizado para o cmdlet. Antes de cada chamada, o cmdlet solicitará a confirmação. Para ignorar a confirmação, adicione o parâmetro -Force.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Para obter detalhes da API, consulte UntagResourcesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para o Route 53 usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Route 53 com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do Route 53: mais informações sobre o Route 53.
- Referência da API Route 53: detalhes sobre todas as ações disponíveis do Route 53.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Route 53 usando AWS SDKs
  - Ações para o Route 53 usando AWS SDKs
    - Usar ChangeResourceRecordSets com uma CLI
    - Usar CreateHostedZone com uma CLI
    - Usar DeleteHostedZone com uma CLI
    - Usar GetHostedZone com uma CLI
    - Use ListHostedZonescom um AWS SDK ou CLI
    - Usar ListHostedZonesByName com uma CLI
    - Usar ListQueryLoggingConfigs com uma CLI

Exemplos básicos do Route 53 usando AWS SDKs

Os exemplos de código a seguir mostram como usar os elementos básicos do Amazon Route 53 com os SDKs da AWS .

Exemplos

- Ações para o Route 53 usando AWS SDKs
  - Usar ChangeResourceRecordSets com uma CLI
  - Usar CreateHostedZone com uma CLI
  - Usar DeleteHostedZone com uma CLI
  - Usar GetHostedZone com uma CLI
  - Use ListHostedZonescom um AWS SDK ou CLI
  - Usar ListHostedZonesByName com uma CLI
  - Usar ListQueryLoggingConfigs com uma CLI

Ações para o Route 53 usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Route 53 com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para uma lista completa, consulte a Amazon Route 53 API Reference.

Exemplos

- Usar ChangeResourceRecordSets com uma CLI
- Usar CreateHostedZone com uma CLI
- Usar DeleteHostedZone com uma CLI
- Usar GetHostedZone com uma CLI
- Use ListHostedZonescom um AWS SDK ou CLI
- Usar ListHostedZonesByName com uma CLI
- Usar ListQueryLoggingConfigs com uma CLI

Usar **`ChangeResourceRecordSets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ChangeResourceRecordSets`.

CLI

AWS CLI

Informações sobre o conjunto de registros de recursos a ser criado, atualizado ou excluído.

O `change-resource-record-sets` comando a seguir cria um conjunto de registros de recursos usando a JSON-formatted configuração `hosted-zone-id` `Z1R8UBAEXAMPLE` e no arquivo`C:\awscli\route53\change-resource-record-sets.json`:

```
aws route53 change-resource-record-sets --hosted-zone-id Z1R8UBAEXAMPLE --change-batch file://C:\awscli\route53\change-resource-record-sets.json
```

Para obter mais informações, consulte POST ChangeResourceRecordSets na Amazon Route 53 API Reference.

A configuração no arquivo JSON depende do tipo de conjunto de registros de recurso que você deseja criar:

BasicWeightedAliasWeighted AliasLatencyLatency AliasFailoverFailover Pseudônimo

Sintaxe básica:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ]
      }
    },
    {...}
  ]
}
```

Sintaxe ponderada:

```
{
  "Comment": "optional comment about the changes in this change batch request",
```

```
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Weight": value between 0 and 255,
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ],
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Sintaxe de alias:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
 Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
 bucket, Elastic Load Balancing load balancer, or another resource record set in
 this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
```

```
    },
    {...}
  ]
}
```

Sintaxe de alias ponderado:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Weight": value between 0 and 255,
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
 Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
 bucket, Elastic Load Balancing load balancer, or another resource record set in
 this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Sintaxe de latência:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
```

```
        "SetIdentifier": "unique description for this resource record set",
        "Region": "Amazon EC2 region name",
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ],
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Sintaxe de alias de latência:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Region": "Amazon EC2 region name",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
 Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
 bucket, Elastic Load Balancing load balancer, or another resource record set in
 this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Sintaxe de failover:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Failover": "PRIMARY" | "SECONDARY",
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ],
        "HealthCheckId": "ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Sintaxe de alias de failover:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Failover": "PRIMARY" | "SECONDARY",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
 Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
 hosted zone",
```

```
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
 bucket, Elastic Load Balancing load balancer, or another resource record set in
 this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

- Para obter detalhes da API, consulte ChangeResourceRecordSetsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo cria um registro A para www.example.com e altera o registro A para test.example.com de 192.0.2.3 a 192.0.2.1. Observe que os valores dos TXT-type registros de alterações devem estar entre aspas duplas. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
```

```
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
 ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
 and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
 ChangeBatch_Change=$change1,$change2,$change3
}

Edit-R53ResourceRecordSet @params
```

Exemplo 2: este exemplo mostra como criar conjuntos de registros de recurso do alias. 'Z222222222' é o ID da zona hospedada do Amazon Route 53 na qual você está criando o conjunto de registros de recurso do alias. 'exemplo.com' é o ápex da zona para o qual você deseja criar um alias e 'www.exemplo.com' é um subdomínio para o qual você também deseja criar um alias. 'Z1111111111111' é um exemplo de ID de zona hospedada para o balanceador de carga e 'example-load-balancer-1111111111.us-east-1.elb.amazonaws.com' é um exemplo de nome de domínio do balanceador de carga com o qual o Amazon Route 53 responde às consultas para example.com e www.example.com. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
```

```
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets,
 one for the zone apex, example.com, and one for www.example.com, that both point
 to example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemplo 3: Este exemplo cria dois registros A para www.example.com. One-fourth Na maioria das vezes (1/ (1+3)), o Amazon Route 53 responde às consultas de www.example.com com os dois valores do primeiro conjunto de registros de recursos (192.0.2.9 e 192.0.2.10). Three-fourths da vez (3/ (1+3)) O Amazon Route 53 responde às consultas de www.example.com com os dois valores para o segundo conjunto de registros de recursos (192.0.2.11 e 192.0.2.12). Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
```

```
$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets,
 each of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemplo 4: Este exemplo mostra como criar conjuntos de registros de recursos de alias ponderados, supondo que example.com seja o domínio para o qual você deseja criar conjuntos de registros de recursos de alias ponderados. SetIdentifier diferencia os dois conjuntos de registros de recursos de alias ponderados um do outro. Esse elemento é necessário porque os elementos Nome e Tipo têm os mesmos valores para os dois conjuntos de registros de recurso. Z1111111111111 e Z3333333333333 são exemplos de IDs de zona hospedada para o balanceador de carga do ELB especificado pelo valor de DNSName. example-load-balancer-2222222222.us-east-1.elb.amazonaws.com e example-load-balancer-4444444444.us-east-1.amazonaws.com são exemplos de domínios do Elastic Load Balancing dos quais o Amazon Route 53 responde às consultas para example.com. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
```

```
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
 record sets. Amazon Route 53 responds to queries for example.com with the first
 ELB domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemplo 5: este exemplo cria dois conjuntos de registros de recurso do alias de latência, um para um balanceador de carga do ELB na região Oeste dos EUA (Oregon) (us-west-2) e outro para um balanceador de carga na região Ásia-Pacífico (Singapura) (ap-southeast-1). Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
```

```
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource
 record sets, one for the US West (Oregon) region and one for the Asia Pacific
 (Singapore) region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- Para obter detalhes da API, consulte ChangeResourceRecordSetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um registro A para www.example.com e altera o registro A para test.example.com de 192.0.2.3 a 192.0.2.1. Observe que os valores dos TXT-type registros de alterações devem estar entre aspas duplas. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
```

```
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords = @()
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords = @()
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
$change3.ResourceRecordSet.ResourceRecords = @()
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
 ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
 and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
 ChangeBatch_Change=$change1,$change2,$change3
}

Edit-R53ResourceRecordSet @params
```

Exemplo 2: este exemplo mostra como criar conjuntos de registros de recurso do alias. 'Z222222222' é o ID da zona hospedada do Amazon Route 53 na qual você está criando o conjunto de registros de recurso do alias. 'exemplo.com' é o ápex da zona para o qual você deseja criar um alias e 'www.exemplo.com' é um subdomínio para o qual você também deseja criar um alias. 'Z1111111111111' é um exemplo de ID de zona hospedada para o balanceador de carga e 'example-load-balancer-1111111111.us-east-1.elb.amazonaws.com' é um exemplo de nome de domínio do balanceador de carga com o qual o Amazon Route 53 responde às consultas para example.com e www.example.com. Consulte a documentação do Amazon

Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets,
 one for the zone apex, example.com, and one for www.example.com, that both point
 to example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemplo 3: Este exemplo cria dois registros A para www.example.com. One-fourth Na maioria das vezes (1/ (1+3)), o Amazon Route 53 responde às consultas de www.example.com com os dois valores do primeiro conjunto de registros de recursos (192.0.2.9 e 192.0.2.10). Three-fourths da vez (3/ (1+3)) O Amazon Route 53 responde às consultas de www.example.com com os dois valores para o segundo conjunto de registros de recursos (192.0.2.11 e

192.0.2.12). Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords = @()
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords = @()
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets,
 each of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemplo 4: Este exemplo mostra como criar conjuntos de registros de recursos de alias ponderados, supondo que example.com seja o domínio para o qual você deseja criar conjuntos de registros de recursos de alias ponderados. SetIdentifier diferencia os dois conjuntos de registros de recursos de alias ponderados um do outro. Esse elemento é necessário porque os elementos Nome e Tipo têm os mesmos valores para os dois conjuntos

de registros de recurso. Z1111111111111 e Z3333333333333 são exemplos de IDs de zona hospedada para o balanceador de carga do ELB especificado pelo valor de DNSName. example-load-balancer-2222222222.us-east-1.elb.amazonaws.com e example-load-balancer-4444444444.us-east-1.amazonaws.com são exemplos de domínios do Elastic Load Balancing dos quais o Amazon Route 53 responde às consultas para example.com. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
 record sets. Amazon Route 53 responds to queries for example.com with the first
 ELB domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
```

```
}

Edit-R53ResourceRecordSet @params
```

Exemplo 5: este exemplo cria dois conjuntos de registros de recurso do alias de latência, um para um balanceador de carga do ELB na região Oeste dos EUA (Oregon) (us-west-2) e outro para um balanceador de carga na região Ásia-Pacífico (Singapura) (ap-southeast-1). Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar para determinar quando as alterações foram concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource
 record sets, one for the US West (Oregon) region and one for the Asia Pacific
 (Singapore) region."
```

```
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- Para obter detalhes da API, consulte ChangeResourceRecordSetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateHostedZone`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateHostedZone`.

CLI

AWS CLI

Para criar uma hosted zone

O comando `create-hosted-zone` a seguir adiciona uma zona hospedada denominada `example.com` usando a referência de chamador `2014-04-01-18:47`. O comentário opcional inclui um espaço, portanto, ele deve ficar entre aspas:

```
aws route53 create-hosted-zone --name example.com --caller-
reference 2014-04-01-18:47 --hosted-zone-config Comment="command-line version"
```

Para obter mais informações, consulte Trabalhar com zonas hospedadas no Guia do desenvolvedor do Amazon Route 53.

- Para obter detalhes da API, consulte CreateHostedZoneem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: cria uma nova zona hospedada denominada 'example.com', associada a um conjunto de delegações reutilizável. Observe que você deve fornecer um valor para o CallerReference parâmetro para que as solicitações precisem ser repetidas, se necessário, sem o risco de executar a operação duas vezes. Como a zona hospedada está sendo criada em uma VPC, ela é automaticamente privada e você não deve definir o parâmetro -HostedZoneConfig _PrivateZone .

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
    VPC_VPCRegion="us-east-1"
}

New-R53HostedZone @params
```

- Para obter detalhes da API, consulte CreateHostedZoneem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: cria uma nova zona hospedada denominada 'example.com', associada a um conjunto de delegações reutilizável. Observe que você deve fornecer um valor para o CallerReference parâmetro para que as solicitações precisem ser repetidas, se necessário, sem o risco de executar a operação duas vezes. Como a zona hospedada está sendo criada em uma VPC, ela é automaticamente privada e você não deve definir o parâmetro -HostedZoneConfig _PrivateZone .

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
    VPC_VPCRegion="us-east-1"
}

New-R53HostedZone @params
```

- Para obter detalhes da API, consulte CreateHostedZoneem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteHostedZone`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteHostedZone`.

CLI

AWS CLI

Para excluir uma zona hospedada do

O comando `delete-hosted-zone` a seguir exclui a zona hospedada com um `id` de`Z36KTIQEXAMPLE`:

```
aws route53 delete-hosted-zone --id Z36KTIQEXAMPLE
```

- Para obter detalhes da API, consulte DeleteHostedZoneem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui a zona hospedada com o ID especificado. Será solicitada uma confirmação antes que o comando continue, a menos que você adicione o parâmetro de opção -Force.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Para obter detalhes da API, consulte DeleteHostedZoneem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui a zona hospedada com o ID especificado. Será solicitada uma confirmação antes que o comando continue, a menos que você adicione o parâmetro de opção -Force.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Para obter detalhes da API, consulte DeleteHostedZoneem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetHostedZone`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetHostedZone`.

CLI

AWS CLI

Para obter informações sobre uma zona hospedada

O comando `get-hosted-zone` a seguir obtém informações sobre a zona hospedada com um `id` de `Z1R8UBAEXAMPLE`:

```
aws route53 get-hosted-zone --id Z1R8UBAEXAMPLE
```

- Para obter detalhes da API, consulte GetHostedZoneem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna detalhes da zona hospedada com o ID Z1D633PJN98FT9.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Para obter detalhes da API, consulte GetHostedZoneem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes da zona hospedada com o ID Z1D633PJN98FT9.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Para obter detalhes da API, consulte GetHostedZoneem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListHostedZones`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListHostedZones`.

CLI

AWS CLI

Para listar as zonas hospedadas associadas à AWS conta atual

O `list-hosted-zones` comando a seguir lista informações resumidas sobre as primeiras 100 zonas hospedadas associadas à AWS conta atual. :

```
aws route53 list-hosted-zones
```

Se você tiver mais de 100 zonas hospedadas ou se quiser listá-las em grupos de menos de 100 zonas, inclua o parâmetro `--max-items`. Por exemplo, para listar as zonas hospedadas, use o seguinte comando:

```
aws route53 list-hosted-zones --max-items 1
```

Para visualizar informações sobre a próxima zona hospedada, pegue o valor de `NextToken` da resposta ao comando anterior e inclua-o no parâmetro `--starting-token`, por exemplo:

```
aws route53 list-hosted-zones --max-items 1 --starting-token Z3M3LMPEXAMPLE
```

- Para obter detalhes da API, consulte ListHostedZonesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: apresenta todas as zonas hospedadas públicas e privadas.

```
Get-R53HostedZoneList
```

Exemplo 2: apresenta todas as zonas hospedadas associadas ao conjunto de delegações reutilizáveis que tem o ID NZ8X2CISAMPLE

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Para obter detalhes da API, consulte ListHostedZonesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: apresenta todas as zonas hospedadas públicas e privadas.

```
Get-R53HostedZoneList
```

Exemplo 2: apresenta todas as zonas hospedadas associadas ao conjunto de delegações reutilizáveis que tem o ID NZ8X2CISAMPLE

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Para obter detalhes da API, consulte ListHostedZonesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_host_info(client: &aws_sdk_route53::Client) -> Result<(),
 aws_sdk_route53::Error> {
    let hosted_zone_count = client.get_hosted_zone_count().send().await?;

    println!(
        "Number of hosted zones in region : {}",
        hosted_zone_count.hosted_zone_count(),
    );

    let hosted_zones = client.list_hosted_zones().send().await?;

    println!("Zones:");
```

```
    for hz in hosted_zones.hosted_zones() {
        let zone_name = hz.name();
        let zone_id = hz.id();

        println!("  ID :   {}", zone_id);
        println!("  Name : {}", zone_name);
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListHostedZonesreferência da API AWS SDK for Rust.

Usar **`ListHostedZonesByName`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListHostedZonesByName`.

CLI

AWS CLI

O comando a seguir lista até 100 zonas hospedadas ordenadas por nome de domínio:

```
aws route53 list-hosted-zones-by-name
```

Saída:

```
{
  "HostedZones": [
      {
          "ResourceRecordSetCount": 2,
          "CallerReference": "test20150527-2",
          "Config": {
              "Comment": "test2",
              "PrivateZone": false
          },
          "Id": "/hostedzone/Z119WBBTVP5WFX",
          "Name": "2.example.com."
      },
```

```
        {
            "ResourceRecordSetCount": 2,
            "CallerReference": "test20150527-1",
            "Config": {
                "Comment": "test",
                "PrivateZone": false
            },
            "Id": "/hostedzone/Z3P5QSUBK4POTI",
            "Name": "www.example.com."
        }
    ],
    "IsTruncated": false,
    "MaxItems": "100"
}
```

O comando a seguir lista as zonas hospedadas ordenadas por nome, começando com`www.example.com`:

```
aws route53 list-hosted-zones-by-name --dns-name www.example.com
```

Saída:

```
{
    "HostedZones": [
        {
            "ResourceRecordSetCount": 2,
            "CallerReference": "mwunderl20150527-1",
            "Config": {
                "Comment": "test",
                "PrivateZone": false
            },
            "Id": "/hostedzone/Z3P5QSUBK4POTI",
            "Name": "www.example.com."
        }
    ],
    "DNSName": "www.example.com",
    "IsTruncated": false,
    "MaxItems": "100"
}
```

- Para obter detalhes da API, consulte ListHostedZonesByNameem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna todas as zonas hospedadas públicas e privadas em ordem ASCII por nome de domínio.

```
Get-R53HostedZonesByName
```

Exemplo 2: retorna as zonas hospedadas públicas e privadas em ordem ASCII por nome de domínio, a partir do nome DNS especificado.

```
Get-R53HostedZonesByName -DnsName example2.com
```

- Para obter detalhes da API, consulte ListHostedZonesByNameem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna todas as zonas hospedadas públicas e privadas em ordem ASCII por nome de domínio.

```
Get-R53HostedZonesByName
```

Exemplo 2: retorna as zonas hospedadas públicas e privadas em ordem ASCII por nome de domínio, a partir do nome DNS especificado.

```
Get-R53HostedZonesByName -DnsName example2.com
```

- Para obter detalhes da API, consulte ListHostedZonesByNameem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListQueryLoggingConfigs`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListQueryLoggingConfigs`.

CLI

AWS CLI

Configurações de logs de consulta do

O `list-query-logging-configs` exemplo a seguir lista informações sobre as primeiras 100 configurações de registro de consultas em sua AWS conta, para a zona `Z1OX3WQEXAMPLE` hospedada.

```
aws route53 list-query-logging-configs \
    --hosted-zone-id Z1OX3WQEXAMPLE
```

Saída:

```
{
    "QueryLoggingConfigs": [
        {
            "Id": "964ff34e-ae03-4f06-80a2-9683cexample",
            "HostedZoneId": "Z1OX3WQEXAMPLE",
            "CloudWatchLogsLogGroupArn": "arn:aws:logs:us-
east-1:111122223333:log-group:/aws/route53/example.com:*"
        }
    ]
}
```

Para obter mais informações, consulte Log de consultas ao DNS no Guia do desenvolvedor do Amazon Route 53.

- Para obter detalhes da API, consulte ListQueryLoggingConfigsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo retorna todas as configurações de registro em log de consultas ao DNS associadas à Conta da AWS atual.

```
Get-R53QueryLoggingConfigList
```

Saída:

```
Id                                   HostedZoneId   CloudWatchLogsLogGroupArn
--                                   ------------   -------------------------
59b0fa33-4fea-4471-a88c-926476aaa40d Z385PDS6EAAAZR arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example1.com:*
```

```
ee528e95-4e03-4fdc-9d28-9e24ddaaa063 Z94SJHBV1AAAAZ arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b Z3MEQ8T7AAA1BF arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example3.com:*
```

- Para obter detalhes da API, consulte ListQueryLoggingConfigsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna todas as configurações de registro em log de consultas ao DNS associadas à Conta da AWS atual.

```
Get-R53QueryLoggingConfigList
```

Saída:

```
Id                                   HostedZoneId   CloudWatchLogsLogGroupArn
--                                   ------------   -------------------------
59b0fa33-4fea-4471-a88c-926476aaa40d Z385PDS6EAAAZR arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example1.com:*
ee528e95-4e03-4fdc-9d28-9e24ddaaa063 Z94SJHBV1AAAAZ arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b Z3MEQ8T7AAA1BF arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example3.com:*
```

- Para obter detalhes da API, consulte ListQueryLoggingConfigsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Exemplos de código para registro de domínio do Route 53 usando AWS SDKs

Os exemplos de código a seguir mostram como usar Amazon Route 53 domain registration com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do desenvolvedor do registro de domínios do Route 53: mais informações sobre o registro de domínios do Route 53.
- Referência da API registro de domínios do Route 53: detalhes sobre todas as ações disponíveis do registro de domínios do Route 53.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para registro de domínio do Route 53 usando AWS SDKs
  - Registro de domínios do Olá, Route 53
  - Aprenda as noções básicas do registro de domínio do Route 53 com um AWS SDK
  - Ações para registro de domínio do Route 53 usando AWS SDKs
    - Use CheckDomainAvailabilitycom um AWS SDK ou CLI
    - Use CheckDomainTransferabilitycom um AWS SDK ou CLI
    - Use GetDomainDetailcom um AWS SDK ou CLI
    - Use GetDomainSuggestionscom um AWS SDK ou CLI
    - Use GetOperationDetailcom um AWS SDK ou CLI
    - Use ListDomainscom um AWS SDK ou CLI
    - Use ListOperationscom um AWS SDK ou CLI
    - Use ListPricescom um AWS SDK
    - Use RegisterDomaincom um AWS SDK ou CLI
    - Use ViewBillingcom um AWS SDK ou CLI

Exemplos básicos para registro de domínio do Route 53 usando AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do Amazon Route 53 domain registration com AWS SDKs.

Exemplos

- Registro de domínios do Olá, Route 53
- Aprenda as noções básicas do registro de domínio do Route 53 com um AWS SDK
- Ações para registro de domínio do Route 53 usando AWS SDKs
  - Use CheckDomainAvailabilitycom um AWS SDK ou CLI
  - Use CheckDomainTransferabilitycom um AWS SDK ou CLI
  - Use GetDomainDetailcom um AWS SDK ou CLI
  - Use GetDomainSuggestionscom um AWS SDK ou CLI
  - Use GetOperationDetailcom um AWS SDK ou CLI
  - Use ListDomainscom um AWS SDK ou CLI
  - Use ListOperationscom um AWS SDK ou CLI
  - Use ListPricescom um AWS SDK
  - Use RegisterDomaincom um AWS SDK ou CLI
  - Use ViewBillingcom um AWS SDK ou CLI

Registro de domínios do Olá, Route 53

O exemplo de código a seguir mostra como começar a usar o registro de domínios do Route 53.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static class HelloRoute53Domains
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon Route 53 domain registration service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRoute53Domains>()
            ).Build();

        // Now the client is available for injection.
        var route53Client =
 host.Services.GetRequiredService<IAmazonRoute53Domains>();

        // You can use await and any of the async methods to get a response.
        var response = await route53Client.ListPricesAsync(new ListPricesRequest
 { Tld = "com" });
        Console.WriteLine($"Hello Amazon Route 53 Domains! Following are prices
 for .com domain operations:");
        var comPrices = response.Prices.FirstOrDefault();
        if (comPrices != null)
        {
            Console.WriteLine($"\tRegistration:
 {comPrices.RegistrationPrice?.Price} {comPrices.RegistrationPrice?.Currency}");
            Console.WriteLine($"\tRenewal: {comPrices.RenewalPrice?.Price}
 {comPrices.RenewalPrice?.Currency}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListPricesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code examples performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the “com” type
 * and displays the prices for Registration and Renewal.
 *
 */
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
                "    <hostedZoneId> \n\n" +
                "Where:\n" +
```

```
                "    hostedZoneId - The id value of an existing hosted zone. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient =
Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Invokes ListPrices for at least one domain type.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient,
String domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .maxItems(10)
                    .tld(domainType)
                    .build();

            ListPricesResponse response =
route53DomainsClient.listPrices(pricesRequest);
            List<DomainPrice> prices = response.prices();
            for (DomainPrice pr : prices) {
                System.out.println("Name: " + pr.name());
                System.out.println(
                        "Registration: " + pr.registrationPrice().price() + " " +
pr.registrationPrice().currency());
                System.out.println("Renewal: " + pr.renewalPrice().price() + " "
+ pr.renewalPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + "
" + pr.transferPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + "
" + pr.transferPrice().currency());
                System.out.println("Change Ownership: " +
pr.changeOwnershipPrice().price() + " "
```

```
                        + pr.changeOwnershipPrice().currency());
                System.out.println(
                        "Restoration: " + pr.restorationPrice().price() + " " +
 pr.restorationPrice().currency());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListPricesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <domainType>

       Where:
           domainType - The domain type (for example, com).
```

```
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val domainType = args[0]
    println("Invokes ListPrices using a Paginated method.")
    listPricesPaginated(domainType)
}

suspend fun listPricesPaginated(domainType: String) {
    val pricesRequest =
        ListPricesRequest {
            maxItems = 10
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListPricesreferência da API AWS SDK for Kotlin.

Aprenda as noções básicas do registro de domínio do Route 53 com um AWS SDK

Os exemplos de código a seguir mostram como:

- Liste os domínios atuais e as operações do ano passado.
- Ver o faturamento do ano passado e os preços dos tipos de domínio.
- Receber sugestões de domínio.
- Verificar a disponibilidade e a transferibilidade de um domínio.
- Opcionalmente, solicitar o registro de um domínio.
- Obter os detalhes de uma operação.
- Opcionalmente, obtenha os detalhes de um domínio.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
public static class Route53DomainScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. List current domains.
        2. List operations in the past year.
        3. View billing for the account in the past year.
        4. View prices for domain types.
        5. Get domain suggestions.
        6. Check domain availability.
        7. Check domain transferability.
        8. Optionally, request a domain registration.
        9. Get an operation detail.
       10. Optionally, get a domain detail.
   */
```

```
    private static Route53Wrapper _route53Wrapper = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
                    .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonRoute53Domains>()
                .AddTransient<Route53Wrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        var logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger(typeof(Route53DomainScenario));

        _route53Wrapper = host.Services.GetRequiredService<Route53Wrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon Route 53 domains example
scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            await ListDomains();
            await ListOperations();
            await ListBillingRecords();
```

```
            await ListPrices();
            await ListDomainSuggestions();
            await CheckDomainAvailability();
            await CheckDomainTransferability();
            var operationId = await RequestDomainRegistration();
            await GetOperationalDetail(operationId);
            await GetDomainDetails();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon Route 53 domains example scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List account registered domains.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomains()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. List account domains.");
        var domains = await _route53Wrapper.ListDomains();
        for (int i = 0; i < domains.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {domains[i].DomainName}");
        }

        if (!domains.Any())
        {
            Console.WriteLine("\tNo domains found in this account.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List domain operations in the past year.
    /// </summary>
```

```
    /// <returns>Async task.</returns>
    private static async Task ListOperations()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List account domain operations in the past
year.");
        var operations = await _route53Wrapper.ListOperations(
            DateTime.Today.AddYears(-1));
        for (int i = 0; i < operations.Count; i++)
        {
            Console.WriteLine($"\tOperation Id: {operations[i].OperationId}");
            Console.WriteLine($"\tStatus: {operations[i].Status}");
            Console.WriteLine($"\tDate: {operations[i].SubmittedDate}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List billing in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListBillingRecords()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. View billing for the account in the past year.");
        var billingRecords = await _route53Wrapper.ViewBilling(
            DateTime.Today.AddYears(-1),
            DateTime.Today);
        for (int i = 0; i < billingRecords.Count; i++)
        {
            Console.WriteLine($"\tBill Date:
{billingRecords[i].BillDate.ToShortDateString()}");
            Console.WriteLine($"\tOperation: {billingRecords[i].Operation}");
            Console.WriteLine($"\tPrice: {billingRecords[i].Price}");
        }
        if (!billingRecords.Any())
        {
            Console.WriteLine("\tNo billing records found in this account for the
past year.");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
```

```
    /// List prices for a few domain types.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListPrices()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. View prices for domain types.");
        var domainTypes = new List<string> { "net", "com", "org", "co" };

        var prices = await _route53Wrapper.ListPrices(domainTypes);
        foreach (var pr in prices)
        {
            Console.WriteLine($"\tName: {pr.Name}");
            Console.WriteLine($"\tRegistration: {pr.RegistrationPrice?.Price}
{pr.RegistrationPrice?.Currency}");
            Console.WriteLine($"\tRenewal: {pr.RenewalPrice?.Price}
{pr.RenewalPrice?.Currency}");
            Console.WriteLine($"\tTransfer: {pr.TransferPrice?.Price}
{pr.TransferPrice?.Currency}");
            Console.WriteLine($"\tChange Ownership:
{pr.ChangeOwnershipPrice?.Price} {pr.ChangeOwnershipPrice?.Currency}");
            Console.WriteLine($"\tRestoration: {pr.RestorationPrice?.Price}
{pr.RestorationPrice?.Currency}");
            Console.WriteLine();
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List domain suggestions for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomainSuggestions()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Get domain suggestions.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to get available domain
suggestions.");
            domainName = Console.ReadLine();
        }
```

```
        var suggestions = await _route53Wrapper.GetDomainSuggestions(domainName,
true, 5);
        foreach (var suggestion in suggestions)
        {
            Console.WriteLine($"\tSuggestion Name: {suggestion.DomainName}");
            Console.WriteLine($"\tAvailability: {suggestion.Availability}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check availability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckDomainAvailability()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Check domain availability.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to check domain
availability.");
            domainName = Console.ReadLine();
        }

        var availability = await
_route53Wrapper.CheckDomainAvailability(domainName);
        Console.WriteLine($"\tAvailability: {availability}");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check transferability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckDomainTransferability()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Check domain transferability.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
```

```
            Console.WriteLine($"Enter a domain name to check domain
 transferability.");
            domainName = Console.ReadLine();
        }

        var transferability = await
 _route53Wrapper.CheckDomainTransferability(domainName);
        Console.WriteLine($"\tTransferability: {transferability}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check transferability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> RequestDomainRegistration()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Optionally, request a domain registration.");

        Console.WriteLine($"\tNote: This example uses domain request settings in
 settings.json.");
        Console.WriteLine($"\tTo change the domain registration settings, set the
 values in that file.");
        Console.WriteLine($"\tRemember, registering an actual domain will incur
 an account billing cost.");
        Console.WriteLine($"\tWould you like to begin a domain registration? (y/
n)");
        var ynResponse = Console.ReadLine();
        if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
        {
            string domainName = _configuration["DomainName"];
            ContactDetail contact = new ContactDetail();
            contact.CountryCode =
 CountryCode.FindValue(_configuration["Contact:CountryCode"]);
            contact.ContactType =
 ContactType.FindValue(_configuration["Contact:ContactType"]);

            _configuration.GetSection("Contact").Bind(contact);

            var operationId = await _route53Wrapper.RegisterDomain(
                domainName,
```

```
                Convert.ToBoolean(_configuration["AutoRenew"]),
                Convert.ToInt32(_configuration["DurationInYears"]),
                contact);
            if (operationId != null)
            {
                Console.WriteLine(
                    $"\tRegistration requested. Operation Id: {operationId}");
            }

            return operationId;
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }

    /// <summary>
    /// Get details for an operation.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetOperationalDetail(string? operationId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Get an operation detail.");

        var operationDetails =
            await _route53Wrapper.GetOperationDetail(operationId);

        Console.WriteLine(operationDetails);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Optionally, get details for a registered domain.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> GetDomainDetails()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Get details on a domain.");

        Console.WriteLine($"\tNote: you must have a registered domain to get
details.");
```

```
        Console.WriteLine($"\tWould you like to get domain details? (y/n)");
        var ynResponse = Console.ReadLine();
        if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
        {
            string? domainName = null;
            while (domainName == null)
            {
                Console.WriteLine($"\tEnter a domain name to get details.");
                domainName = Console.ReadLine();
            }

            var domainDetails = await
 _route53Wrapper.GetDomainDetail(domainName);
            Console.WriteLine(domainDetails);
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }
}
```

Os métodos de wrapper usados pelo cenário para as ações do registro de domínios do Route 53.

```
public class Route53Wrapper
{
    private readonly IAmazonRoute53Domains _amazonRoute53Domains;
    private readonly ILogger<Route53Wrapper> _logger;
    public Route53Wrapper(IAmazonRoute53Domains amazonRoute53Domains,
 ILogger<Route53Wrapper> logger)
    {
        _amazonRoute53Domains = amazonRoute53Domains;
        _logger = logger;
    }


    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
```

```
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginators.ListPrices(new
 ListPricesRequest());
        // Get the entire list using the paginator.
        await foreach (var prices in paginatePrices.Prices)
        {
            results.Add(prices);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }


    /// <summary>
    /// Check the availability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for availability.</param>
    /// <returns>An availability result string.</returns>
    public async Task<string> CheckDomainAvailability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
            new CheckDomainAvailabilityRequest
            {
                DomainName = domain
            }
        );
        return result.Availability.Value;
    }


    /// <summary>
    /// Check the transferability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for transferability.</param>
    /// <returns>A transferability result string.</returns>
    public async Task<string> CheckDomainTransferability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
            new CheckDomainTransferabilityRequest
            {
                DomainName = domain
            }
```

```
        );
        return result.Transferability.Transferable.Value;
    }


    /// <summary>
    /// Get a list of suggestions for a given domain.
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
    /// <param name="onlyAvailable">If true, only returns available domains.</
param>
    /// <param name="suggestionCount">The number of suggestions to return.
 Defaults to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
 bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
    }


    /// <summary>
    /// Get details for a domain action operation.
    /// </summary>
    /// <param name="operationId">The operational Id.</param>
    /// <returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
    {
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
```

```
                        OperationId = operationId
                    }
                );

            var details = $"\tOperation {operationId}:\n" +
                          $"\tFor domain {operationDetails.DomainName} on
 {operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                          $"\tMessage is {operationDetails.Message}.\n" +
                          $"\tStatus is {operationDetails.Status}.\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
    }


    /// <summary>
    /// Initiate a domain registration request.
    /// </summary>
    /// <param name="contact">Contact details.</param>
    /// <param name="domainName">The domain name to register.</param>
    /// <param name="autoRenew">True if the domain should automatically renew.</
param>
    /// <param name="duration">The duration in years for the domain
 registration.</param>
    /// <returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew,
 int duration, ContactDetail contact)
    {
        // This example uses the same contact information for admin, registrant,
 and tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
```

```
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
        catch (InvalidInputException)
        {
            _logger.LogInformation($"Unable to request registration for domain
{domainName}");
            return null;
        }
    }


    /// <summary>
    /// View billing records for the account between a start and end date.
    /// </summary>
    /// <param name="startDate">The start date for billing results.</param>
    /// <param name="endDate">The end date for billing results.</param>
    /// <returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate,
DateTime endDate)
    {
        var results = new List<BillingRecord>();
        var paginateBilling = _amazonRoute53Domains.Paginators.ViewBilling(
            new ViewBillingRequest()
            {
                Start = startDate,
                End = endDate
            });

        // Get the entire list using the paginator.
        await foreach (var billingRecords in paginateBilling.BillingRecords)
        {
            results.Add(billingRecords);
        }
        return results;
    }


    /// <summary>
```

```
    /// List the domains for the account.
    /// </summary>
    /// <returns>A collection of domain summary records.</returns>
    public async Task<List<DomainSummary>> ListDomains()
    {
        var results = new List<DomainSummary>();
        var paginateDomains = _amazonRoute53Domains.Paginators.ListDomains(
            new ListDomainsRequest());

        // Get the entire list using the paginator.
        await foreach (var domain in paginateDomains.Domains)
        {
            results.Add(domain);
        }
        return results;
    }


    /// <summary>
    /// List operations for the account that are submitted after a specified
 date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime
 submittedSince)
    {
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginators.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });

        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
        {
            results.Add(operations);
        }
        return results;
    }


    /// <summary>
    /// Get details for a domain.
```

```
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
    {
        try
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
                    DomainName = domainName
                });
            var details = $"\tDomain {domainName}:\n" +
                          $"\tCreated on
 {result.CreationDate.ToShortDateString()}.\n" +
                          $"\tAdmin contact is {result.AdminContact.Email}.\n" +
                          $"\tAuto-renew is {result.AutoRenew}.\n";

            return details;
        }
        catch (InvalidInputException)
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/
pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");

   public static void main(String[] args) {
       final String usage = """

               Usage:
                   <domainType> <phoneNumber> <email> <domainSuggestion>
<firstName> <lastName> <city>

               Where:
                   domainType - The domain type (for example, com).\s
                   phoneNumber - The phone number to use (for example,
+91.9966564xxx)      email - The email address to use.      domainSuggestion -
The domain suggestion (for example, findmy.accountants).\s
                   firstName - The first name to use to register a domain.\s
                   lastName -  The last name to use to register a domain.\s
                   city - the city to use to register a domain.\s
                   """;

       if (args.length != 7) {
           System.out.println(usage);
           System.exit(1);
       }

       String domainType = args[0];
       String phoneNumber = args[1];
       String email = args[2];
       String domainSuggestion = args[3];
       String firstName = args[4];
       String lastName = args[5];
       String city = args[6];
       Region region = Region.US_EAST_1;
       Route53DomainsClient route53DomainsClient =
Route53DomainsClient.builder()
               .region(region)
               .build();

       System.out.println(DASHES);
       System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
```

```
        System.out.println("1. List current domains.");
        listDomains(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List operations in the past year.");
        listOperations(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. View billing for the account in the past year.");
        listBillingRecords(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. View prices for domain types.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get domain suggestions.");
        listDomainSuggestions(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Check domain availability.");
        checkDomainAvailability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Check domain transferability.");
        checkDomainTransferability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Request a domain registration.");
        String opId = requestDomainRegistration(route53DomainsClient,
domainSuggestion, phoneNumber, email, firstName,
                lastName, city);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get operation details.");
        getOperationalDetail(route53DomainsClient, opId);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get domain details.");
        System.out.println("Note: You must have a registered domain to get
details.");
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxx not found in xxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest =
GetDomainDetailRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                    .operationId(operationId)
                    .build();
```

```
            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " +
response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
            String domainSuggestion,
            String phoneNumber,
            String email,
            String firstName,
            String lastName,
            String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                    .contactType(ContactType.COMPANY)
                    .state("LA")
                    .countryCode(CountryCode.IN)
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();
```

```
            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainTransferabilityResponse response = route53DomainsClient
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                    .checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion + " is " +
response.availability().toString());
```

```
        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                    .domainName(domainSuggestion)
                    .suggestionCount(5)
                    .onlyAvailable(true)
                    .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " +
suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient,
String domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .tld(domainType)
                    .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                    .flatMap(r -> r.prices().stream())
```

```
                    .forEach(content -> System.out.println(" Name: " +
content.name() +
                            " Registration: " +
content.registrationPrice().price() + " "
                            + content.registrationPrice().currency() +
                            " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listBillingRecords(Route53DomainsClient
route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
            Instant myEndTime = localDateTime.toInstant(zoneOffset);

            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                    .start(myStartTime)
                    .end(myEndTime)
                    .build();

            ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
            listRes.stream()
                    .flatMap(r -> r.billingRecords().stream())
                    .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                            " Operation: " + content.operationAsString() +
                            " Price: " + content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   public static void listOperations(Route53DomainsClient route53DomainsClient)
{
       try {
           Date currentDate = new Date();
           LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
           ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
           localDateTime = localDateTime.minusYears(1);
           Instant myTime = localDateTime.toInstant(zoneOffset);

           ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                   .submittedSince(myTime)
                   .build();

           ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
           listRes.stream()
                   .flatMap(r -> r.operations().stream())
                   .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                           " Status: " + content.statusAsString() +
                           " Date: " + content.submittedDate()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void listDomains(Route53DomainsClient route53DomainsClient) {
       try {
           ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
           listRes.stream()
                   .flatMap(r -> r.domains().stream())
                   .forEach(content -> System.out.println("The domain name is "
+ content.domainName()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin code example performs the following operations:
```

```
1. List current domains.
2. List operations in the past year.
3. View billing for the account in the past year.
4. View prices for domain types.
5. Get domain suggestions.
6. Check domain availability.
7. Check domain transferability.
8. Request a domain registration.
9. Get operation details.
10. Optionally, get domain details.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
 <lastName> <city>
        Where:
           domainType - The domain type (for example, com).
           phoneNumber - The phone number to use (for example, +1.2065550100)

           email - The email address to use.
           domainSuggestion - The domain suggestion (for example,
 findmy.example).
           firstName - The first name to use to register a domain.
           lastName -  The last name to use to register a domain.
           city - The city to use to register a domain.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val domainType = args[0]
    val phoneNumber = args[1]
    val email = args[2]
    val domainSuggestion = args[3]
    val firstName = args[4]
    val lastName = args[5]
    val city = args[6]
```

```
    println(DASHES)
    println("Welcome to the Amazon Route 53 domains example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List current domains.")
    listDomains()
    println(DASHES)

    println(DASHES)
    println("2. List operations in the past year.")
    listOperations()
    println(DASHES)

    println(DASHES)
    println("3. View billing for the account in the past year.")
    listBillingRecords()
    println(DASHES)

    println(DASHES)
    println("4. View prices for domain types.")
    listAllPrices(domainType)
    println(DASHES)

    println(DASHES)
    println("5. Get domain suggestions.")
    listDomainSuggestions(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("6. Check domain availability.")
    checkDomainAvailability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("7. Check domain transferability.")
    checkDomainTransferability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("8. Request a domain registration.")
    val opId = requestDomainRegistration(domainSuggestion, phoneNumber, email,
firstName, lastName, city)
    println(DASHES)
```

```
    println(DASHES)
    println("9. Get operation details.")
    getOperationalDetail(opId)
    println(DASHES)

    println(DASHES)
    println("10. Get domain details.")
    println("Note: You must have a registered domain to get details.")
    println("Otherwise an exception is thrown that states ")
    println("Domain xxxxxxx not found in xxxxxxx account.")
    getDomainDetails(domainSuggestion)
    println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is
 ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}

suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}

suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
```

```
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}

suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}

suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}

suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
```

```
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}

suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
```

```
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}

suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}

suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
```

```
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Ações para registro de domínio do Route 53 usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais de registro de domínio do Route 53 com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do Amazon Route 53 domain registration.

Exemplos

- Use CheckDomainAvailabilitycom um AWS SDK ou CLI
- Use CheckDomainTransferabilitycom um AWS SDK ou CLI
- Use GetDomainDetailcom um AWS SDK ou CLI
- Use GetDomainSuggestionscom um AWS SDK ou CLI
- Use GetOperationDetailcom um AWS SDK ou CLI
- Use ListDomainscom um AWS SDK ou CLI
- Use ListOperationscom um AWS SDK ou CLI
- Use ListPricescom um AWS SDK

- Use RegisterDomaincom um AWS SDK ou CLI
- Use ViewBillingcom um AWS SDK ou CLI

Use **`CheckDomainAvailability`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CheckDomainAvailability`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Check the availability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for availability.</param>
/// <returns>An availability result string.</returns>
public async Task<string> CheckDomainAvailability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
        new CheckDomainAvailabilityRequest
        {
            DomainName = domain
        }
    );
    return result.Availability.Value;
}
```

- Para obter detalhes da API, consulte CheckDomainAvailabilitya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para determinar se você pode registrar um nome de domínio com o Route 53

O comando `check-domain-availability` a seguir informa se o nome de domínio `example.com` está ou não disponível para ser registrado usando o Route 53.

Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains check-domain-availability \
    --region us-east-1 \
    --domain-name example.com
```

Saída:

```
{
    "Availability": "UNAVAILABLE"
}
```

O Route 53 é compatível com um grande número de domínios do nível superior (TLDs), como `.com` e `.jp`, mas não com todos os TLDs disponíveis. Se você verificar a disponibilidade de um domínio e o Route 53 não for compatível com o TLD, `check-domain-availability` retornará a mensagem a seguir.

```
An error occurred (UnsupportedTLD) when calling the CheckDomainAvailability
 operation: <top-level domain> tld is not supported.
```

Para obter uma lista dos TLDs genéricos que você pode usar para registrar domínios com o Route 53, consulte Domains That You Can Register with Amazon Route 53 no Guia do desenvolvedor do Amazon Route 53. Para obter mais informações sobre registro de domínios com o Route 53, consulte Registrar um novo domínio no Guia do desenvolvedor do Amazon Route 53.

- Para obter detalhes da API, consulte CheckDomainAvailabilityem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           CheckDomainAvailabilityResponse response = route53DomainsClient
                   .checkDomainAvailability(availabilityRequest);
           System.out.println(domainSuggestion + " is " +
response.availability().toString());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CheckDomainAvailabilitya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- Para obter detalhes da API, consulte a CheckDomainAvailabilityreferência da API AWS SDK for Kotlin.

Use **`CheckDomainTransferability`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CheckDomainTransferability`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Check the transferability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for transferability.</param>
/// <returns>A transferability result string.</returns>
public async Task<string> CheckDomainTransferability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
        new CheckDomainTransferabilityRequest
        {
            DomainName = domain
        }
    );
    return result.Transferability.Transferable.Value;
}
```

- Para obter detalhes da API, consulte CheckDomainTransferabilitya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para determinar se um domínio pode ser transferido para o Route 53

O comando `check-domain-transferability` a seguir retorna informações sobre a possibilidade de transferir um nome de domínio `example.com` para o Route 53.

Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains check-domain-transferability \
    --region us-east-1 \
    --domain-name example.com
```

Saída:

```
{
    "Transferability": {
        "Transferable": "UNTRANSFERABLE"
    }
}
```

Para obter mais informações, consulte Transferring Registration for a Domain to Amazon Route 53 no Guia do desenvolvedor do Amazon Route 53.

- Para obter detalhes da API, consulte CheckDomainTransferabilityem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           CheckDomainTransferabilityResponse response = route53DomainsClient
```

```
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
 response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte CheckDomainTransferabilitya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- Para obter detalhes da API, consulte a CheckDomainTransferabilityreferência da API AWS SDK for Kotlin.

Use **`GetDomainDetail`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetDomainDetail`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get details for a domain.
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
    {
        try
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
                    DomainName = domainName
                });
            var details = $"\tDomain {domainName}:\n" +
                          $"\tCreated on
{result.CreationDate.ToShortDateString()}.\n" +
                          $"\tAdmin contact is {result.AdminContact.Email}.\n" +
                          $"\tAuto-renew is {result.AutoRenew}.\n";

            return details;
        }
        catch (InvalidInputException)
```

```
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
```

- Para obter detalhes da API, consulte GetDomainDetail a Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para obter informações detalhadas sobre um domínio especificado

O comando `get-domain-detail` a seguir exibe informações sobre o domínio especificado.

Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains get-domain-detail \
    --region us-east-1 \
    --domain-name example.com
```

Saída:

```
{
    "DomainName": "example.com",
    "Nameservers": [
        {
            "Name": "ns-2048.awsdns-64.com",
            "GlueIps": []
        },
        {
            "Name": "ns-2049.awsdns-65.net",
            "GlueIps": []
        },
        {
            "Name": "ns-2050.awsdns-66.org",
            "GlueIps": []
        },
        {
```

```
            "Name": "ns-2051.awsdns-67.co.uk",
            "GlueIps": []
        }
    ],
    "AutoRenew": true,
    "AdminContact": {
        "FirstName": "Saanvi",
        "LastName": "Sarkar",
        "ContactType": "COMPANY",
        "OrganizationName": "Example",
        "AddressLine1": "123 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "ssarkar@example.com",
        "ExtraParams": []
    },
    "RegistrantContact": {
        "FirstName": "Alejandro",
        "LastName": "Rosalez",
        "ContactType": "COMPANY",
        "OrganizationName": "Example",
        "AddressLine1": "123 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "arosalez@example.com",
        "ExtraParams": []
    },
    "TechContact": {
        "FirstName": "Wang",
        "LastName": "Xiulan",
        "ContactType": "COMPANY",
        "OrganizationName": "Example",
        "AddressLine1": "123 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
```

```
        "Email": "wxiulan@example.com",
        "ExtraParams": []
    },
    "AdminPrivacy": true,
    "RegistrantPrivacy": true,
    "TechPrivacy": true,
    "RegistrarName": "Amazon Registrar, Inc.",
    "WhoIsServer": "whois.registrar.amazon",
    "RegistrarUrl": "http://registrar.amazon.com",
    "AbuseContactEmail": "abuse@registrar.amazon.com",
    "AbuseContactPhone": "+1.2062661000",
    "CreationDate": 1444934889.601,
    "ExpirationDate": 1602787689.0,
    "StatusList": [
        "clientTransferProhibited"
    ]
}
```

- Para obter detalhes da API, consulte GetDomainDetailem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void getDomainDetails(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           GetDomainDetailRequest detailRequest =
GetDomainDetailRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
```

```
            System.out.println("The contact first name is " +
 response.registrantContact().firstName());
            System.out.println("The contact last name is " +
 response.registrantContact().lastName());
            System.out.println("The contact org name is " +
 response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte GetDomainDetaila Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is
 ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
```

```
    }
}
```

- Para obter detalhes da API, consulte a GetDomainDetailreferência da API AWS SDK for Kotlin.

Use **`GetDomainSuggestions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetDomainSuggestions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of suggestions for a given domain.
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
    /// <param name="onlyAvailable">If true, only returns available domains.</
param>
    /// <param name="suggestionCount">The number of suggestions to return.
 Defaults to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
 bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
```

```
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
    }
```

- Para obter detalhes da API, consulte GetDomainSuggestionsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para obter uma lista de nomes de domínio sugeridos

O comando `get-domain-suggestions` a seguir exibe uma lista de nomes de domínio sugeridos com base no nome de domínio `example.com`. A resposta só inclui os nomes de domínio que estão disponíveis. Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains get-domain-suggestions \
    --region us-east-1 \
    --domain-name example.com \
    --suggestion-count 10 \
    --only-available
```

Saída:

```
{
    "SuggestionsList": [
        {
            "DomainName": "egzaampal.com",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "examplelaw.com",
```

```
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "examplehouse.net",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "homeexample.net",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "examplelist.com",
            "Availability": "AVAILABLE"
       },
        {
            "DomainName": "examplenews.net",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "officeexample.com",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "exampleworld.com",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "exampleart.com",
            "Availability": "AVAILABLE"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetDomainSuggestionsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                   .domainName(domainSuggestion)
                   .suggestionCount(5)
                   .onlyAvailable(true)
                   .build();

           GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
           List<DomainSuggestion> suggestions = response.suggestionsList();
           for (DomainSuggestion suggestion : suggestions) {
               System.out.println("Suggestion Name: " +
suggestion.domainName());
               System.out.println("Availability: " + suggestion.availability());
               System.out.println(" ");
           }

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte GetDomainSuggestionsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetDomainSuggestionsreferência da API AWS SDK for Kotlin.

Use **`GetOperationDetail`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetOperationDetail`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get details for a domain action operation.
    /// </summary>
    /// <param name="operationId">The operational Id.</param>
    /// <returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
    {
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
                        OperationId = operationId
                    }
                );

            var details = $"\tOperation {operationId}:\n" +
                          $"\tFor domain {operationDetails.DomainName} on
{operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                          $"\tMessage is {operationDetails.Message}.\n" +
                          $"\tStatus is {operationDetails.Status}.\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
```

```
    }
```

- Para obter detalhes da API, consulte GetOperationDetaila Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para obter o status atual de uma operação

Algumas operações de registro de domínio são executadas de modo assíncrono e retornam uma resposta antes de serem concluídas. Essas operações retornam um ID de operação que você pode usar para obter o status atual. O comando `get-operation-detail` retorna o status da operação especificada.

Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains get-operation-detail \
    --region us-east-1 \
    --operation-id edbd8d63-7fe7-4343-9bc5-54033example
```

Saída:

```
{
    "OperationId": "edbd8d63-7fe7-4343-9bc5-54033example",
    "Status": "SUCCESSFUL",
    "DomainName": "example.com",
    "Type": "DOMAIN_LOCK",
    "SubmittedDate": 1573749367.864
}
```

- Para obter detalhes da API, consulte GetOperationDetailem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
       try {
           GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                   .operationId(operationId)
                   .build();

           GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
           System.out.println("Operation detail message is " +
response.message());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte GetOperationDetaila Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- Para obter detalhes da API, consulte a GetOperationDetailreferência da API AWS SDK for Kotlin.

Use **`ListDomains`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListDomains`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// List the domains for the account.
/// </summary>
/// <returns>A collection of domain summary records.</returns>
public async Task<List<DomainSummary>> ListDomains()
{
    var results = new List<DomainSummary>();
    var paginateDomains = _amazonRoute53Domains.Paginators.ListDomains(
        new ListDomainsRequest());

    // Get the entire list using the paginator.
    await foreach (var domain in paginateDomains.Domains)
    {
        results.Add(domain);
    }
    return results;
}
```

- Para obter detalhes da API, consulte ListDomainsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os domínios que estão registrados com a conta atual AWS

O `list-domains` comando a seguir lista informações resumidas sobre os domínios registrados na AWS conta atual.

Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains list-domains
    --region us-east-1
```

Saída:

```
{
    "Domains": [
        {
            "DomainName": "example.com",
            "AutoRenew": true,
            "TransferLock": true,
            "Expiry": 1602712345.0
        },
        {
            "DomainName": "example.net",
            "AutoRenew": true,
            "TransferLock": true,
            "Expiry": 1602723456.0
        },
        {
            "DomainName": "example.org",
            "AutoRenew": true,
            "TransferLock": true,
            "Expiry": 1602734567.0
        }
    ]
}
```

- Para obter detalhes da API, consulte ListDomainsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
 route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is "
 + content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListDomainsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListDomainsreferência da API AWS SDK for Kotlin.

Use **`ListOperations`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListOperations`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List operations for the account that are submitted after a specified
date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime
submittedSince)
    {
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginators.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });

        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
```

```
        {
            results.Add(operations);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte ListOperationsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar o status das operações que retornam um ID de operação

Algumas operações de registro de domínio são executadas de modo assíncrono e retornam uma resposta antes de serem concluídas. Essas operações retornam um ID de operação que você pode usar para obter o status atual. O comando `list-operations` a seguir lista informações resumidas, incluindo o status, sobre as operações atuais de registro de domínio.

Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains list-operations
    --region us-east-1
```

Saída:

```
{
    "Operations": [
        {
            "OperationId": "aab9822f-1da0-4bf3-8a15-fd4e0example",
            "Status": "SUCCESSFUL",
            "Type": "DOMAIN_LOCK",
            "SubmittedDate": 1455321739.986
        },
        {
            "OperationId": "c24379ed-76be-42f8-bdad-9379bexample",
            "Status": "SUCCESSFUL",
            "Type": "UPDATE_NAMESERVER",
            "SubmittedDate": 1468960475.109
```

```
        },
        {
            "OperationId": "f47e1297-ef9e-4c2b-ae1e-a5fcbexample",
            "Status": "SUCCESSFUL",
            "Type": "RENEW_DOMAIN",
            "SubmittedDate": 1473561835.943
        },
        {
            "OperationId": "75584f23-b15f-459e-aed7-dc6f5example",
            "Status": "SUCCESSFUL",
            "Type": "UPDATE_DOMAIN_CONTACT",
            "SubmittedDate": 1547501003.41
        }
    ]
}
```

A saída inclui todas as operações que retornam um ID de operação e que você executou em todos os domínios que você já registrou usando a AWS conta atual. Se desejar obter apenas as operações enviadas após uma determinada data, você pode incluir o parâmetro `submitted-since` e especificar uma data no formato Unix e no Horário Universal Coordenado (UTC). O comando a seguir obtém o status de todas as operações enviadas após as 12h UTC de 1º de janeiro de 2020.

```
aws route53domains list-operations \
    --submitted-since 1577836800
```

- Para obter detalhes da API, consulte ListOperationsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listOperations(Route53DomainsClient route53DomainsClient)
 {
```

```
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListOperationsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListOperationsreferência da API AWS SDK for Kotlin.

Use **`ListPrices`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListPrices`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginators.ListPrices(new
ListPricesRequest());
        // Get the entire list using the paginator.
        await foreach (var prices in paginatePrices.Prices)
        {
            results.Add(prices);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }
```

- Para obter detalhes da API, consulte ListPricesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listPrices(Route53DomainsClient route53DomainsClient,
String domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .tld(domainType)
                    .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                    .flatMap(r -> r.prices().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.name() +
                            " Registration: " +
content.registrationPrice().price() + " "
                            + content.registrationPrice().currency() +
                            " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListPricesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
```

```
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListPricesreferência da API AWS SDK for Kotlin.

Use **`RegisterDomain`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RegisterDomain`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Initiate a domain registration request.
    /// </summary>
    /// <param name="contact">Contact details.</param>
    /// <param name="domainName">The domain name to register.</param>
    /// <param name="autoRenew">True if the domain should automatically renew.</
param>
    /// <param name="duration">The duration in years for the domain
 registration.</param>
    /// <returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew,
 int duration, ContactDetail contact)
    {
        // This example uses the same contact information for admin, registrant,
 and tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
        catch (InvalidInputException)
        {
            _logger.LogInformation($"Unable to request registration for domain
 {domainName}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte RegisterDomaina Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para registrar um domínio

O `register-domain` comando a seguir registra um domínio, recuperando todos os valores dos parâmetros de um JSON-formatted arquivo.

Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains register-domain \
    --region us-east-1 \
    --cli-input-json file://register-domain.json
```

Conteúdo de `register-domain.json`:

```
{
    "DomainName": "example.com",
    "DurationInYears": 1,
    "AutoRenew": true,
    "AdminContact": {
        "FirstName": "Martha",
        "LastName": "Rivera",
        "ContactType": "PERSON",
        "OrganizationName": "Example",
        "AddressLine1": "1 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "mrivera@example.com"
    },
    "RegistrantContact": {
        "FirstName": "Li",
        "LastName": "Juan",
        "ContactType": "PERSON",
```

```
        "OrganizationName": "Example",
        "AddressLine1": "1 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "ljuan@example.com"
    },
    "TechContact": {
        "FirstName": "Mateo",
        "LastName": "Jackson",
        "ContactType": "PERSON",
        "OrganizationName": "Example",
        "AddressLine1": "1 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "mjackson@example.com"
    },
    "PrivacyProtectAdminContact": true,
    "PrivacyProtectRegistrantContact": true,
    "PrivacyProtectTechContact": true
}
```

Saída:

```
{
    "OperationId": "b114c44a-9330-47d1-a6e8-a0b11example"
}
```

Para confirmar que a operação foi bem-sucedida, você pode executar `get-operation-detail`. Para obter mais informações, consulte get-operation-detail.

Para obter mais informações, consulte Registrar um novo domínio no Guia do desenvolvedor do Amazon Route 53.

Para obter informações sobre quais domínios de primeiro nível (TLDs) exigem valores `ExtraParams` e quais são os valores válidos, consulte ExtraParama Amazon Route 53 API Reference.

- Para obter detalhes da API, consulte RegisterDomainem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
           String domainSuggestion,
           String phoneNumber,
           String email,
           String firstName,
           String lastName,
           String city) {

       try {
           ContactDetail contactDetail = ContactDetail.builder()
                   .contactType(ContactType.COMPANY)
                   .state("LA")
                   .countryCode(CountryCode.IN)
                   .email(email)
                   .firstName(firstName)
                   .lastName(lastName)
                   .city(city)
                   .phoneNumber(phoneNumber)
                   .organizationName("My Org")
                   .addressLine1("My Address")
                   .zipCode("123 123")
                   .build();

           RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                   .adminContact(contactDetail)
                   .registrantContact(contactDetail)
                   .techContact(contactDetail)
```

```
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();

            RegisterDomainResponse response =
 route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
 response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte RegisterDomaina Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
```

```
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- Para obter detalhes da API, consulte a RegisterDomainreferência da API AWS SDK for Kotlin.

Use **`ViewBilling`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ViewBilling`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// View billing records for the account between a start and end date.
    /// </summary>
    /// <param name="startDate">The start date for billing results.</param>
    /// <param name="endDate">The end date for billing results.</param>
    /// <returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate,
DateTime endDate)
    {
        var results = new List<BillingRecord>();
        var paginateBilling = _amazonRoute53Domains.Paginators.ViewBilling(
            new ViewBillingRequest()
            {
                Start = startDate,
                End = endDate
            });

        // Get the entire list using the paginator.
        await foreach (var billingRecords in paginateBilling.BillingRecords)
        {
            results.Add(billingRecords);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte ViewBillinga Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para obter informações de cobrança das cobranças de registro de domínio da conta corrente AWS

O comando `view-billing` a seguir retorna todos os registros de faturamento referentes a domínios para a conta atual no período de 1º de janeiro de 2018 (1514764800 no horário Unix) e meia-noite de 31 de dezembro de 2019 (1577836800 no horário Unix).

Esse comando só pode ser executado na região `us-east-1`. Se sua região padrão estiver definida como `us-east-1`, você poderá omitir o parâmetro `region`.

```
aws route53domains view-billing \
    --region us-east-1 \
    --start-time 1514764800 \
    --end-time 1577836800
```

Saída:

```
{
    "BillingRecords": [
        {
            "DomainName": "example.com",
            "Operation": "RENEW_DOMAIN",
            "InvoiceId": "149962827",
            "BillDate": 1536618063.181,
            "Price": 12.0
        },
        {
            "DomainName": "example.com",
            "Operation": "RENEW_DOMAIN",
            "InvoiceId": "290913289",
            "BillDate": 1568162630.884,
            "Price": 12.0
        }
    ]
}
```

Para obter mais informações, consulte ViewBillinga Referência de API do Amazon Route 53.

- Para obter detalhes da API, consulte ViewBillingem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listBillingRecords(Route53DomainsClient
route53DomainsClient) {
       try {
           Date currentDate = new Date();
           LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
           ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
           LocalDateTime localDateTime2 = localDateTime.minusYears(1);
           Instant myStartTime = localDateTime2.toInstant(zoneOffset);
           Instant myEndTime = localDateTime.toInstant(zoneOffset);

           ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                   .start(myStartTime)
                   .end(myEndTime)
                   .build();

           ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
           listRes.stream()
                   .flatMap(r -> r.billingRecords().stream())
                   .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                           " Operation: " + content.operationAsString() +
                           " Price: " + content.price()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ViewBillinga Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
```

```
}
```

- Para obter detalhes da API, consulte a ViewBillingreferência da API AWS SDK for Kotlin.

Exemplos de código para o Amazon S3 usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Simple Storage Service com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Amazon S3: mais informações sobre o Amazon S3.
- Referência da API Amazon S3: detalhes sobre todas as ações disponíveis do Amazon S3.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon S3 usando AWS SDKs
  - Olá, Amazon S3
  - Aprenda as noções básicas do Amazon S3 com um AWS SDK
  - Ações para o Amazon S3 usando AWS SDKs
    - Use AbortMultipartUploadcom um AWS SDK ou CLI
    - Use CompleteMultipartUploadcom um AWS SDK ou CLI

- Use CopyObjectcom um AWS SDK ou CLI
- Use CreateBucketcom um AWS SDK ou CLI
- Use CreateMultiRegionAccessPointcom um AWS SDK
- Use CreateMultipartUploadcom um AWS SDK ou CLI
- Use CreatePresignedPostcom um AWS SDK
- Use DeleteBucketcom um AWS SDK ou CLI
- Usar DeleteBucketAnalyticsConfiguration com uma CLI
- Use DeleteBucketCorscom um AWS SDK ou CLI
- Usar DeleteBucketEncryption com uma CLI
- Usar DeleteBucketInventoryConfiguration com uma CLI
- Use DeleteBucketLifecyclecom um AWS SDK ou CLI
- Usar DeleteBucketMetricsConfiguration com uma CLI
- Use DeleteBucketPolicycom um AWS SDK ou CLI
- Usar DeleteBucketReplication com uma CLI
- Usar DeleteBucketTagging com uma CLI
- Use DeleteBucketWebsitecom um AWS SDK ou CLI
- Use DeleteObjectcom um AWS SDK ou CLI
- Usar DeleteObjectTagging com uma CLI
- Use DeleteObjectscom um AWS SDK ou CLI
- Usar DeletePublicAccessBlock com uma CLI
- Usar GetBucketAccelerateConfiguration com uma CLI
- Use GetBucketAclcom um AWS SDK ou CLI
- Usar GetBucketAnalyticsConfiguration com uma CLI
- Use GetBucketCorscom um AWS SDK ou CLI
- Use GetBucketEncryptioncom um AWS SDK ou CLI
- Usar GetBucketInventoryConfiguration com uma CLI
- Use GetBucketLifecycleConfigurationcom um AWS SDK ou CLI
- Use GetBucketLocationcom um AWS SDK ou CLI
- Usar GetBucketLogging com uma CLI

- Usar GetBucketMetricsConfiguration com uma CLI

- Usar GetBucketNotification com uma CLI
- Use GetBucketPolicycom um AWS SDK ou CLI
- Usar GetBucketPolicyStatus com uma CLI
- Use GetBucketReplicationcom um AWS SDK ou CLI
- Usar GetBucketRequestPayment com uma CLI
- Usar GetBucketTagging com uma CLI
- Usar GetBucketVersioning com uma CLI
- Use GetBucketWebsitecom um AWS SDK ou CLI
- Use GetObjectcom um AWS SDK ou CLI
- Use GetObjectAclcom um AWS SDK ou CLI
- Use GetObjectAttributescom um AWS SDK ou CLI
- Use GetObjectLegalHoldcom um AWS SDK ou CLI
- Use GetObjectLockConfigurationcom um AWS SDK ou CLI
- Use GetObjectRetentioncom um AWS SDK ou CLI
- Usar GetObjectTagging com uma CLI
- Usar GetPublicAccessBlock com uma CLI
- Use HeadBucketcom um AWS SDK ou CLI
- Use HeadObjectcom um AWS SDK ou CLI
- Usar ListBucketAnalyticsConfigurations com uma CLI
- Usar ListBucketInventoryConfigurations com uma CLI
- Use ListBucketscom um AWS SDK ou CLI
- Use ListMultipartUploadscom um AWS SDK ou CLI
- Use ListObjectVersionscom um AWS SDK ou CLI
- Usar ListObjects com uma CLI
- Use ListObjectsV2 com um AWS SDK ou CLI
- Use PutBucketAccelerateConfigurationcom um AWS SDK ou CLI
- Use PutBucketAclcom um AWS SDK ou CLI
- Use PutBucketCorscom um AWS SDK ou CLI
- Use PutBucketEncryptioncom um AWS SDK ou CLI

- Use PutBucketLifecycleConfigurationcom um AWS SDK ou CLI

- Use PutBucketLoggingcom um AWS SDK ou CLI
- Usar PutBucketNotification com uma CLI
- Use PutBucketNotificationConfigurationcom um AWS SDK ou CLI
- Use PutBucketPolicycom um AWS SDK ou CLI
- Use PutBucketReplicationcom um AWS SDK ou CLI
- Usar PutBucketRequestPayment com uma CLI
- Usar PutBucketTagging com uma CLI
- Use PutBucketVersioningcom um AWS SDK ou CLI
- Use PutBucketWebsitecom um AWS SDK ou CLI
- Use PutObjectcom um AWS SDK ou CLI
- Use PutObjectAclcom um AWS SDK ou CLI
- Use PutObjectLegalHoldcom um AWS SDK ou CLI
- Use PutObjectLockConfigurationcom um AWS SDK ou CLI
- Use PutObjectRetentioncom um AWS SDK ou CLI
- Use RestoreObjectcom um AWS SDK ou CLI
- Use SelectObjectContentcom um AWS SDK ou CLI
- Use UploadPartcom um AWS SDK ou CLI
- Use UploadPartCopycom um AWS SDK ou CLI

- Cenários para o Amazon S3 usando AWS SDKs
  - Verificar se um bucket existe
  - Verifique se existe um objeto
  - Converta texto em fala e volte para texto usando um AWS SDK
  - Crie uma URL pré-assinada para o Amazon S3 usando um AWS SDK
  - Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
  - Uma página da web que lista objetos do Amazon S3 usando um AWS SDK
  - Criar uma aplicação de exploração do Amazon Textract
  - Exclua todos os objetos em um determinado bucket do Amazon S3 usando um AWS SDK
  - Exclua uploads incompletos de várias partes para o Amazon S3 usando um AWS SDK

  - Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK

- Detecte entidades no texto extraído de uma imagem usando um AWS SDK
- Detecte rostos em uma imagem usando um AWS SDK
- Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
- Baixar “diretórios” do S3 de um bucket do Amazon Simple Storage Service (Amazon S3)
- Fazer download de todos os objetos de um bucket do Amazon Simple Storage Service (Amazon S3) em um diretório local
- Baixe um stream de tamanho desconhecido de um objeto do Amazon S3 usando um AWS SDK
- Obtenha um objeto Amazon S3 de um ponto de Multi-Region acesso usando um AWS SDK
- Obtenha um objeto de um bucket do Amazon S3 usando um AWS SDK, especificando um cabeçalho If-Modified-Since
- Comece a usar a criptografia para objetos do Amazon S3 usando um AWS SDK
- Comece a usar tags para objetos do Amazon S3 usando um AWS SDK
- Introdução aos clusters de processamento de big data
- Conceitos básicos do gerenciamento de configurações
- Introdução à extração de texto de documentos
- Conceitos básicos de arquivos de atributos de machine learning
- Introdução ao armazenamento de objetos
- Introdução à análise de consulta
- Trabalhe com os recursos de bloqueio de objetos do Amazon S3 usando um AWS SDK
- Faça solicitações condicionais do Amazon S3 usando um AWS SDK
- Gerencie listas de controle de acesso (ACLs) para buckets do Amazon S3 usando um AWS SDK
- Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um AWS SDK
- Gerencie objetos versionados do Amazon S3 em lotes com uma função Lambda usando um AWS SDK
- Analise os URIs do Amazon S3 usando um AWS SDK
- Execute uma cópia em várias partes de um objeto do Amazon S3 usando um AWS SDK
- Receba e processe notificações de eventos do Amazon S3 usando um AWS SDK
- Salve EXIF e outras informações da imagem usando um AWS SDK
- Envie notificações de eventos do S3 para a Amazon EventBridge usando um AWS SDK
- Rastreie o upload ou o download de um objeto do Amazon S3 usando um AWS SDK

- Como transformar dados para sua aplicação com o S3 Object Lambda
- Exemplos de abordagens para testes unitários e de integração com um AWS SDK
- Fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service (Amazon S3)
- Carregue ou baixe arquivos grandes de e para o Amazon S3 usando um AWS SDK
- Faça upload de um stream de tamanho desconhecido para um objeto do Amazon S3 usando um AWS SDK
- Use somas de verificação para trabalhar com um objeto do Amazon S3 usando um AWS SDK
- Trabalhe com os recursos de integridade de objetos do Amazon S3 usando um AWS SDK
- Trabalhe com objetos versionados do Amazon S3 usando um AWS SDK
- Exemplos de tecnologia sem servidor para o Amazon S3
  - Invocar uma função do Lambda em um acionador do Amazon S3

Exemplos básicos para o Amazon S3 usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Simple Storage Service com SDKs da AWS .

Exemplos

- Olá, Amazon S3
- Aprenda as noções básicas do Amazon S3 com um AWS SDK
- Ações para o Amazon S3 usando AWS SDKs
  - Use AbortMultipartUploadcom um AWS SDK ou CLI
  - Use CompleteMultipartUploadcom um AWS SDK ou CLI
  - Use CopyObjectcom um AWS SDK ou CLI
  - Use CreateBucketcom um AWS SDK ou CLI
  - Use CreateMultiRegionAccessPointcom um AWS SDK
  - Use CreateMultipartUploadcom um AWS SDK ou CLI
  - Use CreatePresignedPostcom um AWS SDK
  - Use DeleteBucketcom um AWS SDK ou CLI
  - Usar DeleteBucketAnalyticsConfiguration com uma CLI
  - Use DeleteBucketCorscom um AWS SDK ou CLI

- Usar DeleteBucketEncryption com uma CLI
- Usar DeleteBucketInventoryConfiguration com uma CLI
- Use DeleteBucketLifecyclecom um AWS SDK ou CLI
- Usar DeleteBucketMetricsConfiguration com uma CLI
- Use DeleteBucketPolicycom um AWS SDK ou CLI
- Usar DeleteBucketReplication com uma CLI
- Usar DeleteBucketTagging com uma CLI
- Use DeleteBucketWebsitecom um AWS SDK ou CLI
- Use DeleteObjectcom um AWS SDK ou CLI
- Usar DeleteObjectTagging com uma CLI
- Use DeleteObjectscom um AWS SDK ou CLI
- Usar DeletePublicAccessBlock com uma CLI
- Usar GetBucketAccelerateConfiguration com uma CLI
- Use GetBucketAclcom um AWS SDK ou CLI
- Usar GetBucketAnalyticsConfiguration com uma CLI
- Use GetBucketCorscom um AWS SDK ou CLI
- Use GetBucketEncryptioncom um AWS SDK ou CLI
- Usar GetBucketInventoryConfiguration com uma CLI
- Use GetBucketLifecycleConfigurationcom um AWS SDK ou CLI
- Use GetBucketLocationcom um AWS SDK ou CLI
- Usar GetBucketLogging com uma CLI
- Usar GetBucketMetricsConfiguration com uma CLI
- Usar GetBucketNotification com uma CLI
- Use GetBucketPolicycom um AWS SDK ou CLI
- Usar GetBucketPolicyStatus com uma CLI
- Use GetBucketReplicationcom um AWS SDK ou CLI
- Usar GetBucketRequestPayment com uma CLI
- Usar GetBucketTagging com uma CLI
- Usar GetBucketVersioning com uma CLI

- Use GetBucketWebsitecom um AWS SDK ou CLI

- Use GetObjectcom um AWS SDK ou CLI
- Use GetObjectAclcom um AWS SDK ou CLI
- Use GetObjectAttributescom um AWS SDK ou CLI
- Use GetObjectLegalHoldcom um AWS SDK ou CLI
- Use GetObjectLockConfigurationcom um AWS SDK ou CLI
- Use GetObjectRetentioncom um AWS SDK ou CLI
- Usar GetObjectTagging com uma CLI
- Usar GetPublicAccessBlock com uma CLI
- Use HeadBucketcom um AWS SDK ou CLI
- Use HeadObjectcom um AWS SDK ou CLI
- Usar ListBucketAnalyticsConfigurations com uma CLI
- Usar ListBucketInventoryConfigurations com uma CLI
- Use ListBucketscom um AWS SDK ou CLI
- Use ListMultipartUploadscom um AWS SDK ou CLI
- Use ListObjectVersionscom um AWS SDK ou CLI
- Usar ListObjects com uma CLI
- Use ListObjectsV2 com um AWS SDK ou CLI
- Use PutBucketAccelerateConfigurationcom um AWS SDK ou CLI
- Use PutBucketAclcom um AWS SDK ou CLI
- Use PutBucketCorscom um AWS SDK ou CLI
- Use PutBucketEncryptioncom um AWS SDK ou CLI
- Use PutBucketLifecycleConfigurationcom um AWS SDK ou CLI
- Use PutBucketLoggingcom um AWS SDK ou CLI
- Usar PutBucketNotification com uma CLI
- Use PutBucketNotificationConfigurationcom um AWS SDK ou CLI
- Use PutBucketPolicycom um AWS SDK ou CLI
- Use PutBucketReplicationcom um AWS SDK ou CLI
- Usar PutBucketRequestPayment com uma CLI
- Usar PutBucketTagging com uma CLI

- Use PutBucketVersioningcom um AWS SDK ou CLI

- Use PutBucketWebsitecom um AWS SDK ou CLI
- Use PutObjectcom um AWS SDK ou CLI
- Use PutObjectAclcom um AWS SDK ou CLI
- Use PutObjectLegalHoldcom um AWS SDK ou CLI
- Use PutObjectLockConfigurationcom um AWS SDK ou CLI
- Use PutObjectRetentioncom um AWS SDK ou CLI
- Use RestoreObjectcom um AWS SDK ou CLI
- Use SelectObjectContentcom um AWS SDK ou CLI
- Use UploadPartcom um AWS SDK ou CLI
- Use UploadPartCopycom um AWS SDK ou CLI

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Hello Amazon Simple Storage Service
// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
```

```
    {
        var s3Client = new AmazonS3Client();

        try
        {
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
            Console.WriteLine(new string('-', 80));

            // Use the built-in paginator to list buckets
            var request = new ListBucketsRequest();
            var paginator = s3Client.Paginators.ListBuckets(request);

            var buckets = new List<S3Bucket>();

            await foreach (var response in paginator.Responses)
            {
                buckets.AddRange(response.Buckets);
            }

            if (buckets.Any())
            {
                Console.WriteLine($"Found {buckets.Count} S3 buckets:");
                Console.WriteLine();

                foreach (var bucket in buckets)
                {
                    Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
                    Console.WriteLine($"  Creation Date:
{bucket.CreationDate:yyyy-MM-dd HH:mm:ss UTC}");
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No S3 buckets found in your account.");
            }

            Console.WriteLine("Hello S3 completed successfully.");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"S3 service error occurred: {ex.Message}");
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"Couldn't list S3 buckets. Here's why:
 {ex.Message}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_s3.cpp.

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
 *  Usage: 'hello_s3'
 *
```

```
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
 S3 service permits anonymous requests, thus the s3Client will return "success"
 and 0 buckets even if you are unauthenticated, which can be confusing to a new
 user.
        auto provider =
 Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed with error: " << outcome.GetError() <<
 std::endl;
            result = 1;
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                      << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "errors"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
```

```
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  var ae smithy.APIError
  if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
   fmt.Println("You don't have permission to list buckets for this account.")
  } else {
   fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  }
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {
  // When no region or credentials are provided, the SDK will use the
  // region and credentials from the local AWS config.
  const client = new S3Client({});

  try {
    /**
     * @type { import("@aws-sdk/client-s3").Bucket[] }
     */
    const buckets = [];

    for await (const page of paginateListBuckets({ client }, {})) {
      buckets.push(...page.Buckets);
    }
    console.log("Buckets: ");
    console.log(buckets.map((bucket) => bucket.Name).join("\n"));
    return buckets;
  } catch (caught) {
    // ListBuckets does not throw any modeled errors. Any error caught
```

```
    // here will be something generic like `AccessDenied`.
    if (caught instanceof S3ServiceException) {
      console.error(`${caught.name}: ${caught.message}`);
    } else {
      // Something besides S3 failed.
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")

    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
            "PageSize": 50,  # Adjust PageSize as needed.
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response.
    buckets_found = False
    for page in response_iterator:
        if "Buckets" in page and page["Buckets"]:
            buckets_found = True
```

```
            for bucket in page["Buckets"]:
                print(f"\t{bucket['Name']}")

    if not buckets_found:
        print("No buckets found!")


if __name__ == "__main__":
    hello_s3()
```

- Para obter detalhes da API, consulte a ListBucketsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets
```

```
    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the
 object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
```

```
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
 present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the
 bucket.
async fn list_bucket_and_upload_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
        .bucket(bucket)
        .into_paginator()
        .send();

    println!("key\tetag\tlast_modified\tstorage_class");
    while let Some(Ok(object)) = objects.next().await {
        for item in object.contents() {
            println!(
                "{}\t{}\t{}\t{}",
                item.key().unwrap_or_default(),
                item.e_tag().unwrap_or_default(),
                item.last_modified()
                    .map(|lm| format!("{lm}"))
                    .unwrap_or_default(),
                item.storage_class()
                    .map(|sc| format!("{sc}"))
                    .unwrap_or_default()
            );
        }
    }

    // Prepare a ByteStream around the file, and upload the object using that
 ByteStream.
    let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
        .await
        .map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to create bytestream for {filepath:?} ({err:?})"
            ))
```

```
        })?;
    let resp = client
        .put_object()
        .bucket(bucket)
        .key(key)
        .body(body)
        .send()
        .await?;

    println!(
        "Upload success. Version: {:?}",
        resp.version_id()
            .expect("S3 Object upload missing version ID")
    );

    // Retrieve the just-uploaded object.
    let resp = client.get_object().bucket(bucket).key(key).send().await?;
    println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
    println!("version: {}", resp.version_id().unwrap_or("(missing)"));

    Ok(())
}
```

S3ExampleError utilitários.

```
/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
```

```
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

- Para obter detalhes da API, consulte a ListBucketsreferência da API AWS SDK for Rust.

Aprenda as noções básicas do Amazon S3 com um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um bucket e faça upload de um arquivo para ele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon S3.

```
public class S3_Basics
{
    public static bool IsInteractive = true;
    public static string BucketName = null!;
    public static string TempFilePath = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static ILogger<S3_Basics> _logger = null!;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddLogging(builder => builder.AddConsole()))
            .Build();

        _logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<S3_Basics>();

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();

        var sepBar = new string('-', 45);

        Console.WriteLine(sepBar);
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
        Console.WriteLine("procedures. This application will:");
        Console.WriteLine("\n\t1. Create a bucket");
        Console.WriteLine("\n\t2. Upload an object to the new bucket");
```

```
        Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
        Console.WriteLine("\n\t4. List the items in the new bucket");
        Console.WriteLine("\n\t5. Delete all the items in the bucket");
        Console.WriteLine("\n\t6. Delete the bucket");
        Console.WriteLine(sepBar);

        await RunScenario(_s3Wrapper, _logger);

        Console.WriteLine(sepBar);
        Console.WriteLine("The Amazon S3 scenario has successfully completed.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Run the S3 Basics scenario with injected dependencies.
    /// </summary>
    /// <param name="s3Wrapper">The S3 wrapper instance.</param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
scenarioLogger)
    {
        string bucketName = BucketName;
        string filePath = TempFilePath;
        string keyName = string.Empty;

        var sepBar = new string('-', 45);

        try
        {
            // Create a bucket.
            Console.WriteLine($"\n{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                Console.Write("Please enter a name for the new bucket: ");
                bucketName = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Using bucket name: {bucketName}");
```

```
            }

            var success = await s3Wrapper.CreateBucketAsync(bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                // Get the local path and filename for the file to upload.
                while (string.IsNullOrEmpty(filePath))
                {
                    Console.Write("Please enter the path and filename of the file
 to upload: ");
                    filePath = Console.ReadLine();

                    // Confirm that the file exists on the local computer.
                    if (!File.Exists(filePath))
                    {
                        Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
                        filePath = string.Empty;
                    }
                }
            }
            else
            {
                // Use the public variable if set, otherwise create a temp file
                if (!string.IsNullOrEmpty(TempFilePath))
                {
                    filePath = TempFilePath;
                    Console.WriteLine($"Using provided test file: {filePath}");
                }
                else
```

```
                {
                    // Create a temporary test file for non-interactive mode
                    filePath = Path.GetTempFileName();
                    var testContent = "This is a test file for S3 basics
scenario.\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss
UTC");
                    await File.WriteAllTextAsync(filePath, testContent);
                    Console.WriteLine($"Created temporary test file:
{filePath}");
                }
            }

            // Get the file name from the full path.
            keyName = Path.GetFileName(filePath);

            success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
filePath);

            if (success)
            {
                Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not upload {keyName}.\n");
            }

            // Set up download path
            string downloadPath = string.Empty;

            if (IsInteractive)
            {
                // Now get a new location where we can save the file.
                while (string.IsNullOrEmpty(downloadPath))
                {
                    // First get the path to which the file will be downloaded.
                    Console.Write("Please enter the path where the file will be
downloaded: ");
                    downloadPath = Console.ReadLine();

                    // Confirm that the file doesn't already exist on the local
computer.
                    if (File.Exists($"{downloadPath}\\{keyName}"))
```

```
                    {
                        Console.WriteLine($"Sorry, the file already exists in
that location.\n");
                        downloadPath = string.Empty;
                    }
                }
            }
            else
            {
                downloadPath = Path.GetTempPath();
                var downloadFile = Path.Combine(downloadPath, keyName);
                if (File.Exists(downloadFile))
                {
                    File.Delete(downloadFile);
                }

                Console.WriteLine($"Using download path: {downloadPath}");
            }

            // Download an object from a bucket.
            success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

            if (success)
            {
                Console.WriteLine($"Successfully downloaded {keyName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not download {keyName}.\n");
            }

            // Copy the object to a different folder in the bucket.
            string folderName = string.Empty;

            if (IsInteractive)
            {
                while (string.IsNullOrEmpty(folderName))
                {
                    Console.Write("Please enter the name of the folder to copy
your object to: ");
                    folderName = Console.ReadLine();
                }
            }
```

```
            else
            {
                folderName = "test-folder";
                Console.WriteLine($"Using folder name: {folderName}");
            }

            await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
folderName);

            // List the objects in the bucket.
            await s3Wrapper.ListBucketContentsAsync(bucketName);

            // Delete the contents of the bucket.
            if (IsInteractive)
            {
                Console.WriteLine("Press <Enter> when you are ready to delete the
bucket contents.");
                _ = Console.ReadLine();
            }

            var deleteContentsSuccess = await
s3Wrapper.DeleteBucketContentsAsync(bucketName);
            if (deleteContentsSuccess)
            {
                Console.WriteLine($"Successfully deleted contents of
{bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
            }

            if (IsInteractive)
            {
                // Deleting the bucket too quickly after separately deleting its
contents can
                // cause an error that the bucket isn't empty. To delete contents
and bucket in one
                // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
                Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
                _ = Console.ReadLine();
            }
```

```
            else
            {
                // Add a small delay for non-interactive mode to ensure objects
are fully deleted.
                Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
                await Task.Delay(2000);
            }

            // Delete the bucket.
            var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
            if (deleteSuccess)
            {
                Console.WriteLine($"Successfully deleted {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
            }

            // Clean up temporary files in non-interactive mode
            if (!IsInteractive)
            {
                try
                {
                    if (File.Exists(filePath))
                    {
                        File.Delete(filePath);
                        Console.WriteLine("Cleaned up temporary test file.");
                    }

                    var downloadFile = Path.Combine(downloadPath, keyName);
                    if (File.Exists(downloadFile))
                    {
                        File.Delete(downloadFile);
                        Console.WriteLine("Cleaned up downloaded test file.");
                    }
                }
                catch (Exception ex)
                {
                    scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
                }
            }
```

```
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
 execution.");

            // Clean up on error - delete bucket if it exists
            try
            {
                if (!string.IsNullOrEmpty(bucketName))
                {
                    await s3Wrapper.DeleteBucketContentsAsync(bucketName);
                    await s3Wrapper.DeleteBucketAsync(bucketName);
                }
            }
            catch (Exception cleanupEx)
            {
                scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
            }

            // Clean up temporary files in non-interactive mode
            if (!IsInteractive)
            {
                try
                {
                    if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
                    {
                        File.Delete(filePath);
                    }
                }
                catch (Exception fileCleanupEx)
                {
                    scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
 temporary files during error handling.");
                }
            }

            throw;
        }
    }
}
```

Uma classe de wrapper para os métodos de SDK do Amazon S3.

```
using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Initializes a new instance of the <see cref="S3Wrapper"/> class.
    /// </summary>
    /// <param name="amazonS3">An initialized Amazon S3 client object.</param>
    public S3Wrapper(IAmazonS3 amazonS3)
    {
        _amazonS3 = amazonS3;
    }


    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
    {
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
            };

            var response = await _amazonS3.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }



    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }
```

```
    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await
 _amazonS3.GetObjectAsync(request);

        try
        {
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }



    /// <summary>
```

```
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }



    /// <summary>
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
```

```
    {
        try
        {
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
                MaxKeys = 5,
            };

            if (printList)
            {
                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");
            }

            var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });
            var s3Objects = new List<S3Object>();
            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                if (response.S3Objects != null)
                {
                    s3Objects.AddRange(response.S3Objects);
                }
            }

            if (printList)
            {
                Console.WriteLine($"Number of Objects: {s3Objects.Count}");
                foreach (var entry in s3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }

            return s3Objects;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' getting list of objects.");
            return null;
        }
    }



    /// <summary>
    /// Delete all of the objects stored in an existing Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket from which the
    /// contents will be deleted.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// deleting all of the objects in the bucket.</returns>
    public async Task<bool> DeleteBucketContentsAsync(string bucketName)
    {
        // Iterate over the contents of the bucket and delete all objects.
        try
        {
            // Delete all objects in the bucket.
            var deleteList = await ListBucketContentsAsync(bucketName, false);
            if (deleteList != null && deleteList.Any())
            {
                await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
                });
            }

            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
        }
    }



    /// <summary>
```

```
    /// Shows how to delete an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
    /// <returns>A boolean value that represents the success or failure of
    /// the delete operation.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest { BucketName = bucketName, };

            await _amazonS3.DeleteBucketAsync(request);
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function s3_getting_started() {
  {
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
      cd bucket-lifecycle-operations || exit

      source ./bucket_operations.sh
      cd ..
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the Amazon S3 getting started demo."
  echo_repeat "*" 88
    echo "A unique bucket will be created by appending a Universally Unique
 Identifier to a bucket name prefix."
    echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
    get_input
    bucket_name_prefix=$get_input_result
  local bucket_name
  bucket_name=$(generate_random_name "$bucket_name_prefix")

  local region_code
  region_code=$(aws configure get region)
```

```
 if create_bucket -b "$bucket_name" -r "$region_code"; then
   echo "Created demo bucket named $bucket_name"
 else
   errecho "The bucket failed to create. This demo will exit."
   return 1
 fi

 local file_name
 while [ -z "$file_name" ]; do
   echo -n "Enter a file you want to upload to your bucket: "
   get_input
   file_name=$get_input_result

   if [ ! -f "$file_name" ]; then
     echo "Could not find file $file_name. Are you sure it exists?"
     file_name=""
   fi
 done

 local key
 key="$(basename "$file_name")"

 local result=0
 if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
   echo "Uploaded file $file_name into bucket $bucket_name with key $key."
 else
   result=1
 fi

 local destination_file
 destination_file="$file_name.download"
 if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
   if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
     echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
   else
     result=1
   fi
 fi
```

```
  if yes_no_input "Would you like to copy $key a new object key in your bucket?
 (y/n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
      echo "Copied $key in the bucket $bucket_name to the  $to_key."
    else
      result=1
    fi
  fi

  local bucket_items
  bucket_items=$(list_items_in_bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    result=1
  fi

  echo "Your bucket contains the following items."
  echo -e "Name\t\tSize"
  echo "$bucket_items"

  if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
 (y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
      echo "The following items were deleted from the bucket $bucket_name"
      echo "$bucket_items"
    else
      result=1
    fi

    if delete_bucket "$bucket_name"; then
      echo "Deleted the bucket $bucket_name"
    else
      result=1
    fi
  fi

  return $result
}
```

As funções do Amazon S3 usadas nesse cenário.

```
##############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
```

```
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
```

```
}

##############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

##############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

##############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
```

```
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

```
###############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}

###############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
```

```
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param client: An S3 client.
          \return bool: Function succeeded.
        */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
```

```
          \param bucketName: The S3 bucket's name.
          \param key: The key for the object in the S3 bucket.
          \param client: An S3 client.
          \return bool: Function succeeded.
         */
        static bool
        deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
 &key,
                               Aws::S3::S3Client &client);
    }
}


//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
  \param bucketNamePrefix: A prefix for a bucket name.
  \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
  \param saveFilePath: Path for saving a downloaded S3 object.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
        const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = bucketNamePrefix +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(
```

```
Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                            clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                      "', in the region, '" << clientConfig.region << "'." <<
std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
                Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                              uploadFilePath,
                                              std::ios_base::in |
                                              std::ios_base::binary);

        if (!input_data->is_open()) {
            std::cerr << "Error: unable to open file, '" << uploadFilePath <<
"'."
                      << std::endl;
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        }

        request.SetBody(input_data);
```

```
        Aws::S3::Model::PutObjectOutcome outcome =
                client.PutObject(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: putObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
            AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        } else {
            std::cout << "Added the object with the key, '" << key
                      << "', to the bucket, '"
                      << bucketName << "'." << std::endl;
        }
    }

    // 3. Download the object to a local file.
    {
        Aws::S3::Model::GetObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        Aws::S3::Model::GetObjectOutcome outcome =
                client.GetObject(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: getObject: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        } else {
            std::cout << "Downloaded the object with the key, '" << key
                      << "', in the bucket, '"
                      << bucketName << "'." << std::endl;

            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                    GetBody();
            Aws::OFStream outStream(saveFilePath,
                                    std::ios_base::out | std::ios_base::binary);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath <<
"'."
                          << std::endl;
```

```
            } else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file '"
                          << saveFilePath << "'." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
                .WithKey(copiedToKey)
                .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
                client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: copyObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Copied the object with the key, '" << key
                      << "', to the key, '" << copiedToKey
                      << ", in the bucket, '" << bucketName << "'." << std::endl;
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsV2Request request;
        request.WithBucket(bucketName);

        Aws::String continuationToken;
        Aws::Vector<Aws::S3::Model::Object> allObjects;

        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
                    request);

            if (!outcome.IsSuccess()) {
```

```
                std::cerr << "Error: ListObjects: " <<
                          outcome.GetError().GetMessage() << std::endl;
                break;
            } else {
                Aws::Vector<Aws::S3::Model::Object> objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
 objects.end());
                continuationToken = outcome.GetResult().GetContinuationToken();
            }
        } while (!continuationToken.empty());

        std::cout << allObjects.size() << " objects in the bucket, '" <<
 bucketName
                  << "':" << std::endl;

        for (Aws::S3::Model::Object &object: allObjects) {
            std::cout << "     '" << object.GetKey() << "'" << std::endl;
        }
    }

    // 6. Delete all objects in the bucket.
    // All objects in the bucket must be deleted before deleting the bucket.
    AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
    AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

    // 7. Delete the bucket.
    return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
```

```
        std::cout << "Deleted the object with the key, '" << key
                  << "', from the bucket, '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
 &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina um struct que envolva as ações de bucket e objeto usadas pelo cenário.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
```

```
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
 error) {
 var err error
 var output *s3.ListBucketsOutput
 var buckets []types.Bucket
 bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
 &s3.ListBucketsInput{})
 for bucketPaginator.HasMorePages() {
  output, err = bucketPaginator.NextPage(ctx)
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
   } else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
   }
   break
  } else {
   buckets = append(buckets, output.Buckets...)
  }
 }
 return buckets, err
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
 (bool, error) {
 _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
```

```
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
 string) error {
 _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", name)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", name)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
  }
 }
 return err
}
```

```
// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
     "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
     "or the multipart upload API (5TB max).", bucketName)
   } else {
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
     fileName, bucketName, objectKey, err)
   }
  } else {
   err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
   }
  }
 }
 return err
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
 string, objectKey string, largeObject []byte) error {
```

```
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
```

```
    log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
  objectKey, bucketName)
    err = noKey
   } else {
    log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
  objectKey, err)
   }
   return err
  }
  defer result.Body.Close()
  file, err := os.Create(fileName)
  if err != nil {
   log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
   return err
  }
  defer file.Close()
  body, err := io.ReadAll(result.Body)
  if err != nil {
   log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
  err)
  }
  _, err = file.Write(body)
  return err
}



// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
```

```
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}



// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
 objectKey string, folderName string) error {
 objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(bucketName),
  CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
  Key:        aws.String(objectDest),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, bucketName)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectDest)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
  }
 }
 return err
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
 destinationBucket string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
```

```
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, sourceBucket)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }
 return err
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
 ([]types.Object, error) {
 var err error
 var output *s3.ListObjectsV2Output
 input := &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 }
 var objects []types.Object
 objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
 for objectPaginator.HasMorePages() {
  output, err = objectPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
   break
  } else {
```

```
    objects = append(objects, output.Contents...)
   }
  }
  return objects, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
 objectKeys []string) error {
 var objectIds []types.ObjectIdentifier
 for _, key := range objectKeys {
  objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
 }
 output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
  Bucket: aws.String(bucketName),
  Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
 })
 if err != nil || len(output.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucketName)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
  } else if len(output.Errors) > 0 {
   for _, outErr := range output.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *output.Errors[0].Message)
  }
 } else {
  for _, delObjs := range output.Deleted {
   err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
 time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to be deleted.\n",
 *delObjs.Key)
   } else {
    log.Printf("Deleted %s.\n", *delObjs.Key)
   }
```

```
  }
 }
 return err
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
 error {
 _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucketName)
   err = noBucket
  } else {
   log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
  }
 } else {
  err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
 bucketName)
  } else {
   log.Printf("Deleted %s.\n", bucketName)
  }
 }
 return err
}
```

Execute um cenário interativo que mostre como trabalhar com buckets e objetos do S3.

```
import (
 "context"
 "fmt"
 "log"
```

```
 "os"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use
 Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
 objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 getting started demo.")
```

```
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

count := 10
log.Printf("Let's list up to %v buckets for your account:", count)
buckets, err := bucketBasics.ListBuckets(ctx)
if err != nil {
 panic(err)
}
if len(buckets) == 0 {
 log.Println("You don't have any buckets!")
} else {
 if count > len(buckets) {
  count = len(buckets)
 }
 for _, bucket := range buckets[:count] {
  log.Printf("\t%v\n", *bucket.Name)
 }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
 demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
```

```
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
 panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
 panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
 panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
 objKeys = append(objKeys, *object.Key)
 log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
 "contents? (y/n)", "y") {
 log.Println("Deleting objects.")
 err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
 if err != nil {
```

```
    panic(err)
   }
   log.Println("Deleting bucket.")
   err = bucketBasics.DeleteBucket(ctx, bucketName)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleting downloaded file %v.\n", downloadFileName)
   err = os.Remove(downloadFileName)
   if err != nil {
    panic(err)
   }
  } else {
   log.Println("Okay. Don't forget to delete objects from your bucket to avoid
  charges.")
  }
  log.Println(strings.Repeat("-", 88))

  log.Println("Thanks for watching!")
  log.Println(strings.Repeat("-", 88))
 }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Uma situação de exemplo.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */
```

```
public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = """
            Usage:
               <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The name of the  S3 bucket.
                key - The unique identifier for the object stored in the S3
 bucket.
                objectPath - The full file path of the object within the S3
 bucket (e.g., "documents/reports/annual_report.pdf").
                savePath - The local file path where the object will be
 downloaded and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
                toBucket - The name of the S3 bucket to which the object will be
 copied.
            """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
 scenario.");
        logger.info("""
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to
 store and retrieve
```

```
            any amount of data, from anywhere on the web, at any time.

            The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
            programmatically interact with the Amazon S3 (Simple Storage Service)
service. This allows
            developers to automate the management and manipulation of S3 buckets
and objects as
            part of their application deployment pipelines. With S3, teams can
focus on building
            and deploying their applications without having to worry about the
underlying storage
            infrastructure required to host and manage large amounts of data.

            This scenario walks you through how to perform key operations for
this service.
            Let's get started...
            """);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            // Run the methods that belong to this scenario.
            runScenario(bucketName, key, objectPath, savePath, toBucket);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create an Amazon S3 bucket.");
        try {
            CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
            future.join();
```

```
            waitForInputToContinue(scanner);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Upload a local file to the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
            future.join();
            logger.info("File uploaded successfully to {}/{}", bucketName, key);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("3. Download the object to another local file.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
```

```
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future =
s3Actions.multipartUpload(bucketName, multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{}'
and key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. List all objects located in the Amazon S3 bucket.");
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
s3Actions.performMultiCopy(toBucket, bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
```

```
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
 {}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Delete the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 s3Actions.deleteBucketAsync(bucketName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
 {}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon S3 scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
```

```
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper que contém as operações.

```
public class S3Actions {

    private static final Logger logger =
 LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
```

```
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
     * @throws RuntimeException if there is a failure while creating the bucket
     */
    public CompletableFuture<Void> createBucketAsync(String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
        return response.thenCompose(resp -> {
            S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
```

```
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                    logger.info(bucketName + " is ready");
                });
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }


    /**
     * Uploads a local file to an AWS S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (object name) to use for the uploaded file
     * @param objectPath the local file path of the file to be uploaded
     * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
     */
    public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }


    /**
```

```
    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
and writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes
have been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
       return response.thenAccept(objectBytes -> {
           try {
               byte[] data = objectBytes.asByteArray();
               Path filePath = Paths.get(path);
               Files.write(filePath, data);
               logger.info("Successfully obtained bytes from an S3 object");
           } catch (IOException ex) {
               throw new RuntimeException("Failed to write data to file", ex);
           }
       }).whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to get object bytes from S3",
ex);
           }
       });
   }


   /**
    * Asynchronously lists all objects in the specified S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to list objects for
    * @return a {@link CompletableFuture} that completes when all objects have
been listed
    */
```

```
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }


    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as
a {@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3
exception occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
```

```
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

    /**
     * Performs a multipart upload to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (name) of the file to be uploaded
     * @return a {@link CompletableFuture} that completes when the multipart
upload is successful
     */
    public CompletableFuture<Void> multipartUpload(String bucketName, String key)
{
        int mB = 1024 * 1024;

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        return
getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenCompose(createResponse -> {
                String uploadId = createResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 =
UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(1)
                    .contentLength((long) (5 * mB)) // Specify the content length
```

```
                    .build();

                CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(1)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Upload part 2.
                UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(2)
                    .contentLength((long) (3 * mB))
                    .build();

                CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(2)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Combine the results of both parts.
                return CompletableFuture.allOf(part1Future, part2Future)
                    .thenCompose(v -> {
                        CompletedPart part1 = part1Future.join();
                        CompletedPart part2 = part2Future.join();

                        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                            .parts(part1, part2)
                            .build();

                        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                            .bucket(bucketName)
```

```
                            .key(key)
                            .uploadId(uploadId)
                            .multipartUpload(completedMultipartUpload)
                            .build();

                        // Complete the multipart upload
                        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
                    });
            })
            .thenAccept(response -> System.out.println("Multipart upload
completed successfully"))
            .exceptionally(ex -> {
                System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }


    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the key (file name) of the object to be deleted
     * @return a {@link CompletableFuture} that completes when the object has
been deleted
     */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during
delete", ex);
            }
```

```
        });

        return response.thenApply(r -> null);
    }


    /**
     * Deletes an S3 bucket asynchronously.
     *
     * @param bucket the name of the bucket to be deleted
     * @return a {@link CompletableFuture} that completes when the bucket
deletion is successful, or throws a {@link RuntimeException}
     * if an error occurs during the deletion process
     */
    public CompletableFuture<Void> deleteBucketAsync(String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }

    public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);
```

```
                UploadPartCopyRequest uploadPartCopyRequest =
 UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
 exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
 ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        ByteBuffer buffer = ByteBuffer.allocate(size);
        for (int i = 0; i < size; i++) {
            buffer.put((byte) (Math.random() * 256));
        }
        buffer.flip();
        return buffer;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Primeiro, importe todos os módulos necessários.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

As importações anteriores fazem referência a alguns utilitários auxiliares. Esses utilitários são locais para o GitHub repositório vinculado no início desta seção. Para sua referência, consulte as implementações a seguir desses utilitários.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {{ message: string }} options
   */
  password(options) {
    return password({ ...options, mask: true });
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && `${prompt} `;
    const ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };
```

```
  /**
   * @param {{ message: string }} options
   */
  confirm(options) {
    return confirm(options);
  }

  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  checkbox(options) {
    return checkbox(options);
  }
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

Os objetos são armazenados em “baldes”. Vamos definir uma função para criar um bucket.

```
export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};
```

Os buckets contêm “objetos”. Essa função faz upload do conteúdo de um diretório para seu bucket como objetos.

```
export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
    const filePath = `${folderPath}/${key}`;
```

```
    const fileContent = readFileSync(filePath);
    return {
      Key: key,
      Body: fileContent,
    };
  });

  for (const file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
};
```

Depois de fazer upload dos objetos, confira se eles foram carregados corretamente. Você pode usar ListObjects para isso. Você usará a propriedade “Key”, mas também há outras propriedades úteis na resposta.

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(`${contentsList}\n`);
};
```

Às vezes é necessário copiar um objeto de um bucket em outro. Use o CopyObject comando para isso.

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {
```

```
    return;
  }
  const copy = async () => {
    try {
      const sourceBucket = await prompter.input({
        message: "Enter source bucket name:",
      });
      const sourceKey = await prompter.input({
        message: "Enter source key:",
      });
      const destinationKey = await prompter.input({
        message: "Enter destination key:",
      });

      const command = new CopyObjectCommand({
        Bucket: destinationBucket,
        CopySource: `${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });
      await s3Client.send(command);
      await copyFileFromBucket({ destinationBucket });
    } catch (err) {
      console.error("Copy error.");
      console.error(err);
      const retryAnswer = await prompter.confirm({ message: "Try again?" });
      if (retryAnswer) {
        await copy();
      }
    }
  };
  await copy();
};
```

Não há um método de SDK para obter vários objetos de um bucket. Em vez disso, você criará uma lista de objetos para baixar e iterar sobre eles.

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
```

```
  });

  for (const content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

É hora de limpar seus recursos. Um bucket deve estar vazio para poder ser excluído. Essas duas funções esvaziam e excluem o bucket.

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

A função “principal” reúne tudo. Se você executar esse arquivo diretamente, a função principal será chamada.

```
const main = async () => {
```

```
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to
 provide your own.",
    );
    await uploadFilesToBucket({
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
 example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
```

```
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
```

```
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }
```

```
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
```

```
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
```

```
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
```

```
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

```
        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)
```

```
    bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            }
        )
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
        return

    file_name = None
    while file_name is None:
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
            file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(
            f"Uploaded file {file_name} into bucket {bucket.name} with key
 {obj.key}."
        )
    except S3UploadFailedError as err:
        print(f"Couldn't upload file {file_name} to {bucket.name}.")
        print(f"\t{err}")

    answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
    if answer.lower() == "y":
        data = io.BytesIO()
        try:
            obj.download_fileobj(data)
            data.seek(0)
            print(f"Got your object. Here are the first 20 bytes:\n")
            print(f"\t{data.read(20)}")
        except ClientError as err:
```

```
            print(f"Couldn't download {obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
 "
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
```

```
            )

    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
```

```
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
  def upload_file(bucket)
    File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
    s3_object = bucket.object(File.basename('demo.txt'))
    s3_object.upload_file('demo.txt')
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
```

```
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      puts('Enter a name for the downloaded file: ')
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your
 bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
```

```
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
 ")
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
```

```
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
 == __FILE__
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para a caixa binária que executa o cenário.

```
#![allow(clippy::result_large_err)]

//!  Purpose
//!  Shows how to use the AWS SDK for Rust to get started using
//!  Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects
 into and out of it,
//!  and delete all resources at the end of the demo.
//!
//!  This example follows the steps in "Getting started with Amazon S3" in the
 Amazon S3
//!  user guide.
//!  - https://docs.aws.amazon.com/AmazonS3/latest/userguide/
GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
 key, target_key).await
    {
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,
```

```
    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), S3ExampleError> {
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
    let run_example: Result<(), S3ExampleError> = (async {
        s3_code_examples::upload_object(&client, &bucket_name, &file_name,
 &key).await?;
        let _object = s3_code_examples::download_object(&client, &bucket_name,
 &key).await;
        s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
 &target_key)
            .await?;
        s3_code_examples::list_objects(&client, &bucket_name).await?;
        s3_code_examples::clear_bucket(&client, &bucket_name).await?;
        Ok(())
    })
    .await;
    if let Err(err) = run_example {
        eprintln!("Failed to complete getting-started example: {err:?}");
    }
    s3_code_examples::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}
```

Ações comuns usadas na situação.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
 S3ExampleError> {
    let constraint =
 aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
```

```
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
 task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
 se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
 aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {
```

```
    client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await
        .map_err(S3ExampleError::from)
}

/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
 with etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
 aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}

pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
 Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
```

```
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();

    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;
```

```
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(S3ExampleError::new(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject

- ListObjectsV2
- PutObject

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

    " Create an Amazon Simple Storage Service (Amazon S3) bucket. "
    TRY.
        " determine our region from our session
        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
        DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
        " When in the us-east-1 region, you must not specify a constraint
        " In all other regions, specify the region as the constraint
        IF lv_region = 'us-east-1'.
          CLEAR lo_constraint.
        ELSE.
          lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
        ENDIF.

        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
            io_createbucketconfiguration  = lo_constraint ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.
```

```
    "Upload an object to an S3 bucket."
    TRY.
        "Get contents of file from application server."
        DATA lv_file_content TYPE xstring.
        OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
        READ DATASET iv_key INTO lv_file_content.
        CLOSE DATASET iv_key.

        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key
            iv_body = lv_file_content ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.

    " Get an object from a bucket. "
    TRY.
        DATA(lo_result) = lo_s3->getobject(
                   iv_bucket = iv_bucket_name
                   iv_key = iv_key ).
        DATA(lv_object_data) = lo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " Copy an object to a subfolder in a bucket. "
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_bucket_name
          iv_key = |{ iv_copy_to_folder }/{ iv_key }|
          iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
        MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " List objects in the bucket. "
```

```
    TRY.
        DATA(lo_list) = lo_s3->listobjects(
           iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
    DATA text TYPE string VALUE 'Object List - '.
    DATA lv_object_key TYPE /aws1/s3_objectkey.
    LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
      lv_object_key = lo_object->get_key( ).
      CONCATENATE lv_object_key ', ' INTO text.
    ENDLOOP.
    MESSAGE text TYPE'I'.

    " Delete the objects in a bucket. "
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key ).
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.


    " Delete the bucket. "
    TRY.
        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
 drive the AWS calls
    /// used for the example.
    ///
```

```
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///            execute AWS operations.
    public init() async throws {
        do {
            configuration = try await S3Client.S3ClientConfiguration()
         //   configuration.region = "us-east-2" // Uncomment this to set the
 region programmatically.
            client = S3Client(config: configuration)
        }
        catch {
            print("ERROR: ", dump(error, name: "Initializing S3 client"))
            throw error
        }
    }


    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        var input = CreateBucketInput(
            bucket: name
        )

        // For regions other than "us-east-1", you must set the
 locationConstraint in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
 S3ClientTypes.CreateBucketConfiguration(locationConstraint:
 S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
```

```
            print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        do {
            _ = try await client.deleteBucket(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a bucket"))
            throw error
        }
    }

    /// Upload a file from local storage to the bucket.
    /// - Parameters:
    ///   - bucket: Name of the bucket to upload the file to.
    ///   - key: Name of the file to create.
    ///   - file: Path name of the file to upload.
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        do {
            let fileData = try Data(contentsOf: fileUrl)
            let dataStream = ByteStream.data(fileData)

            let input = PutObjectInput(
                body: dataStream,
                bucket: bucket,
                key: key
            )

            _ = try await client.putObject(input: input)
```

```
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }

    /// Create a file in the specified bucket with the given name. The new
    /// file's contents are uploaded from a `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket to create a file in.
    ///   - key: Name of the file to create.
    ///   - data: A `Data` object to write into the new file.
    public func createFile(bucket: String, key: String, withData data: Data)
 async throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }

    /// Download the named file to the given directory on the local device.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket that contains the file to be copied.
    ///   - key: The name of the file to copy from the bucket.
    ///   - to: The path of the directory on the local device where you want to
    ///     download the file.
    public func downloadFile(bucket: String, key: String, to: String) async
 throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)
```

```
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file."))
            throw error
        }
    }

    /// Read the specified file from the given S3 bucket into a Swift
    /// `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to read.
    ///   - key: Name of the file within the bucket to read.
    ///
    /// - Returns: A `Data` object containing the complete file data.
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }
```

```
            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
 read data.")
            }

            return data
        }
        catch {
            print("ERROR: ", dump(error, name: "Reading a file."))
            throw error
        }
    }


    /// Copy a file from one bucket to another.
    ///
    /// - Parameters:
    ///   - sourceBucket: Name of the bucket containing the source file.
    ///   - name: Name of the source file.
    ///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
            _ = try await client.copyObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object."))
            throw error
        }
    }

    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
```

```
    ///
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a file."))
            throw error
        }
    }

    /// Returns an array of strings, each naming one file in the
    /// specified bucket.
    ///
    /// - Parameter bucket: Name of the bucket to get a file listing for.
    /// - Returns: An array of `String` objects, each giving the name of
    ///            one file contained in the bucket.
    public func listBucketFiles(bucket: String) async throws -> [String] {
        do {
            let input = ListObjectsV2Input(
                bucket: bucket
            )

            // Use "Paginated" to get all the objects.
            // This lets the SDK handle the 'continuationToken' in
 "ListObjectsV2Output".
            let output = client.listObjectsV2Paginated(input: input)
            var names: [String] = []

            for try await page in output {
                guard let objList = page.contents else {
                    print("ERROR: listObjectsV2Paginated returned nil contents.")
                    continue
                }

                for obj in objList {
                    if let objName = obj.key {
                        names.append(objName)
                    }
```

```
                }
            }


            return names
        }
        catch {
            print("ERROR: ", dump(error, name: "Listing objects."))
            throw error
        }
    }
}
```

```
import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
 bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
```

```
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
    )

    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = try await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

        // 4. Copy the file to another bucket.
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
        let fileList = try await serviceHandler.listBucketFiles(bucket:
bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
```

```
            print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
 \(bucketName):")
            for name in fileList {
                print("  \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }

        // 6. Delete the objects from the bucket.

        print("Deleting the file \(objName) from the bucket \(bucketName)...")
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações para o Amazon S3 usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon S3 com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon S3 e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon S3 usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de API do Amazon Simple Storage Service.

Exemplos

- Use AbortMultipartUploadcom um AWS SDK ou CLI
- Use CompleteMultipartUploadcom um AWS SDK ou CLI
- Use CopyObjectcom um AWS SDK ou CLI
- Use CreateBucketcom um AWS SDK ou CLI
- Use CreateMultiRegionAccessPointcom um AWS SDK
- Use CreateMultipartUploadcom um AWS SDK ou CLI
- Use CreatePresignedPostcom um AWS SDK
- Use DeleteBucketcom um AWS SDK ou CLI
- Usar DeleteBucketAnalyticsConfiguration com uma CLI
- Use DeleteBucketCorscom um AWS SDK ou CLI

- Usar DeleteBucketEncryption com uma CLI
- Usar DeleteBucketInventoryConfiguration com uma CLI
- Use DeleteBucketLifecyclecom um AWS SDK ou CLI
- Usar DeleteBucketMetricsConfiguration com uma CLI
- Use DeleteBucketPolicycom um AWS SDK ou CLI
- Usar DeleteBucketReplication com uma CLI
- Usar DeleteBucketTagging com uma CLI
- Use DeleteBucketWebsitecom um AWS SDK ou CLI
- Use DeleteObjectcom um AWS SDK ou CLI
- Usar DeleteObjectTagging com uma CLI
- Use DeleteObjectscom um AWS SDK ou CLI
- Usar DeletePublicAccessBlock com uma CLI
- Usar GetBucketAccelerateConfiguration com uma CLI
- Use GetBucketAclcom um AWS SDK ou CLI
- Usar GetBucketAnalyticsConfiguration com uma CLI
- Use GetBucketCorscom um AWS SDK ou CLI
- Use GetBucketEncryptioncom um AWS SDK ou CLI
- Usar GetBucketInventoryConfiguration com uma CLI
- Use GetBucketLifecycleConfigurationcom um AWS SDK ou CLI
- Use GetBucketLocationcom um AWS SDK ou CLI
- Usar GetBucketLogging com uma CLI
- Usar GetBucketMetricsConfiguration com uma CLI
- Usar GetBucketNotification com uma CLI
- Use GetBucketPolicycom um AWS SDK ou CLI
- Usar GetBucketPolicyStatus com uma CLI
- Use GetBucketReplicationcom um AWS SDK ou CLI
- Usar GetBucketRequestPayment com uma CLI
- Usar GetBucketTagging com uma CLI
- Usar GetBucketVersioning com uma CLI
- Use GetBucketWebsitecom um AWS SDK ou CLI

- Use GetObjectcom um AWS SDK ou CLI
- Use GetObjectAclcom um AWS SDK ou CLI
- Use GetObjectAttributescom um AWS SDK ou CLI
- Use GetObjectLegalHoldcom um AWS SDK ou CLI
- Use GetObjectLockConfigurationcom um AWS SDK ou CLI
- Use GetObjectRetentioncom um AWS SDK ou CLI
- Usar GetObjectTagging com uma CLI
- Usar GetPublicAccessBlock com uma CLI
- Use HeadBucketcom um AWS SDK ou CLI
- Use HeadObjectcom um AWS SDK ou CLI
- Usar ListBucketAnalyticsConfigurations com uma CLI
- Usar ListBucketInventoryConfigurations com uma CLI
- Use ListBucketscom um AWS SDK ou CLI
- Use ListMultipartUploadscom um AWS SDK ou CLI
- Use ListObjectVersionscom um AWS SDK ou CLI
- Usar ListObjects com uma CLI
- Use ListObjectsV2 com um AWS SDK ou CLI
- Use PutBucketAccelerateConfigurationcom um AWS SDK ou CLI
- Use PutBucketAclcom um AWS SDK ou CLI
- Use PutBucketCorscom um AWS SDK ou CLI
- Use PutBucketEncryptioncom um AWS SDK ou CLI
- Use PutBucketLifecycleConfigurationcom um AWS SDK ou CLI
- Use PutBucketLoggingcom um AWS SDK ou CLI
- Usar PutBucketNotification com uma CLI
- Use PutBucketNotificationConfigurationcom um AWS SDK ou CLI
- Use PutBucketPolicycom um AWS SDK ou CLI
- Use PutBucketReplicationcom um AWS SDK ou CLI
- Usar PutBucketRequestPayment com uma CLI
- Usar PutBucketTagging com uma CLI
- Use PutBucketVersioningcom um AWS SDK ou CLI

- Use PutBucketWebsitecom um AWS SDK ou CLI
- Use PutObjectcom um AWS SDK ou CLI
- Use PutObjectAclcom um AWS SDK ou CLI
- Use PutObjectLegalHoldcom um AWS SDK ou CLI
- Use PutObjectLockConfigurationcom um AWS SDK ou CLI
- Use PutObjectRetentioncom um AWS SDK ou CLI
- Use RestoreObjectcom um AWS SDK ou CLI
- Use SelectObjectContentcom um AWS SDK ou CLI
- Use UploadPartcom um AWS SDK ou CLI
- Use UploadPartCopycom um AWS SDK ou CLI

Use **`AbortMultipartUpload`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AbortMultipartUpload`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Excluir uploads multiparte incompletos
- Trabalhar com a integridade de objetos do Amazon S3

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
```

```
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AbortMultipartUploada Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para interromper um carregamento fracionado especificado

O comando `abort-multipart-upload` a seguir interrompe um carregamento fracionado da chave `multipart/01` no bucket `amzn-s3-demo-bucket`.

```
aws s3api abort-multipart-upload \
    --bucket amzn-s3-demo-bucket \
    --key multipart/01 \
```

```
    --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

O ID de upload exigido por esse comando é a saída de `create-multipart-upload` e também pode ser recuperado com `list-multipart-uploads`.

- Para obter detalhes da API, consulte AbortMultipartUploadem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;
```

```
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
 LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
        String key = UUID.randomUUID().toString();

        createBucket(bucketName);
        try {
            initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
            abortIncompleteMultipartUploadsFromList(bucketName);
```

```
            abortMultipartUploadUsingUploadId(bucketName, key);
            abortMultipartUploadsUsingLifecycleConfig(bucketName);
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources(bucketName, key);
        }
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     */
    public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
            }
        }
    }

    /**
```

```
    * Aborts incomplete multipart uploads older than the specified point in
time.
    *
    * @param bucketName  the name of the S3 bucket
    * @param pointInTime the cutoff time; uploads initiated before this are
aborted
    */
   static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
           .bucket(bucketName)
           .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
           if (upload.initiated().isBefore(pointInTime)) {
               abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                   .bucket(bucketName)
                   .key(upload.key())
                   .expectedBucketOwner(accountId)
                   .uploadId(upload.uploadId())
                   .build();

               AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
               if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                   logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
               }
           }
       }
   }

   /**
    * Aborts a multipart upload using the upload ID.
```

```
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key)
 {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
 uploadId, bucketName);
        }
    }

    /**
     * Configures a lifecycle rule to abort incomplete multipart uploads after 7
 days.
     *
     * @param bucketName the name of the S3 bucket
     */
    static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
        Collection<LifecycleRule> lifeCycleRules =
 List.of(LifecycleRule.builder()
            .abortIncompleteMultipartUpload(b -> b.
                daysAfterInitiation(7))
            .status("Enabled")
            .filter(SdkBuilder::build) // Filter element is required.
            .build());

        PutBucketLifecycleConfigurationResponse response =
 s3Client.putBucketLifecycleConfiguration(b -> b
            .bucket(bucketName)
            .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
 bucket.");
        } else {
```

```
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }

    /************************
     Multipart upload methods
     ************************/

    static void initiateAndInterruptMultiPartUpload(String bucketName, String
key, String threadName) {
        Runnable upload = () -> {
            try {
                doMultipartUpload(bucketName, key);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
        return secondUploadInstant;
    }

    static void doMultipartUpload(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
```

```
        List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
        step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
    }

    static String step1CreateMultipartUpload(String bucketName, String key) {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip();
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);
                logger.info("Part {} upload", partNumber);
```

```
                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }

    static void step3CompleteMultipartUpload(String bucketName, String key,
String uploadId, List<CompletedPart> completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }

    static String startUploadReturningUploadId(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        doMultipartUploadWithUploadId(bucketName, key, uploadId);
        return uploadId;
    }

    static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
                step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
```

```
            System.exit(1);
        }
    }

    /************************
     Resource handling methods
     ************************/

    static void createBucket(String bucketName) {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources(String bucketName, String key) {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL =
PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
```

```
    }
}
```

- Para obter detalhes da API, consulte AbortMultipartUploada Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando interrompe os carregamentos fracionados criados há mais de cinco dias.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Exemplo 2: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
 "Thursday, January 02, 2014"
```

Exemplo 3: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014, às 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
 "2014/01/02 10:45:37"
```

- Para obter detalhes da API, consulte AbortMultipartUploadem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando interrompe os carregamentos fracionados criados há mais de cinco dias.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Exemplo 2: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
 "Thursday, January 02, 2014"
```

Exemplo 3: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014, às 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
 "2014/01/02 10:45:37"
```

- Para obter detalhes da API, consulte AbortMultipartUploadem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CompleteMultipartUpload`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CompleteMultipartUpload`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Executar uma cópia multipart
- Usar somas de verificação
- Trabalhar com a integridade de objetos do Amazon S3

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
```

```
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

  const Aws::String &key,

  const Aws::String &uploadID,

  const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}
```

- Para obter detalhes da API, consulte CompleteMultipartUploada Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando conclui um upload fracionado da chave `multipart/01` no bucket `amzn-s3-demo-bucket`:

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

O ID de upload exigido por esse comando é a saída de `create-multipart-upload` e também pode ser recuperado com `list-multipart-uploads`.

A opção de upload fracionado no comando acima usa uma estrutura JSON que descreve quais partes do upload fracionado devem ser reagrupadas no arquivo completo. Neste exemplo, o prefixo `file://` é usado para carregar a estrutura JSON de um arquivo na pasta local chamada `mpustruct`.

mpustruct:

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

O valor de ETag de cada parte carregada é gerado sempre que você fizer o upload de uma parte usando o comando `upload-part`. Ele também pode ser recuperado ao chamar `list-parts` ou calculado fazendo a soma de verificação MD5 de cada parte.

Saída:

```
{
    "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
    "Bucket": "amzn-s3-demo-bucket",
    "Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",
    "Key": "multipart/01"
```

```
}
```

- Para obter detalhes da API, consulte CompleteMultipartUploadem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
```

```
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

- Para obter detalhes da API, consulte a CompleteMultipartUploadreferência da API AWS SDK for Rust.

Use **`CopyObject`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CopyObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos de criptografia
- Introdução ao armazenamento de objetos
- Fazer solicitações condicionais

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
```

```
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK para .NET da API.

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie um objeto usando uma solicitação condicional.

```
    /// <summary>
    /// Copies an object from one Amazon S3 bucket to another with a conditional
 request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
    /// <param name="conditionType">The type of condition to apply, e.g.
 'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
 'CopySourceIfUnmodifiedSince'.</param>
```

```
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string
 destKey, string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
 string? etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
            }

            await _amazonS3.CopyObjectAsync(copyObjectRequest);
```

```
            _logger.LogInformation($"Conditional copy successful for key
 {destKey} in bucket {destBucket}.");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional copy failed: Precondition failed");
            }
            else if (e.ErrorCode == "304")
            {
                _logger.LogError("Conditional copy failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte CopyObjectem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
 &fromBucket, const Aws::String &toBucket,
                            const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;

    } else {
        std::cout << "Successfully copied " << objectKey << " from " <<
 fromBucket <<
                  " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK para C++ da API.

CLI

AWS CLI

O comando a seguir copia um objeto de `bucket-1` para `bucket-2`:

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --
bucket bucket-2
```

Saída:

```
{
    "CopyObjectResult": {
        "LastModified": "2015-11-10T01:07:25.000Z",
        "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
    },
    "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifMlkA"
}
```

- Para obter detalhes da API, consulte CopyObjectem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
 destinationBucket string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, sourceBucket)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }
```

```
 return err
}
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie um objeto usando um S3Client.

```
   /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param fromBucket the name of the source S3 bucket
    * @param objectKey  the key (name) of the object to be copied
    * @param toBucket   the name of the destination S3 bucket
    * @return a {@link CompletableFuture} that completes with the copy result as
a {@link String}
    * @throws RuntimeException if the URL could not be encoded or an S3
exception occurred during the copy
    */
   public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
       CopyObjectRequest copyReq = CopyObjectRequest.builder()
           .sourceBucket(fromBucket)
           .sourceKey(objectKey)
           .destinationBucket(toBucket)
           .destinationKey(objectKey)
           .build();

       CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
```

```
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
 copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }
```

Use an S3TransferManagerpara copiar um objeto de um bucket para outro. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
 bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);
```

```
        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie o objeto.

```
import {
  S3Client,
  CopyObjectCommand,
  ObjectNotInActiveTierError,
  waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{
 *   sourceBucket: string,
 *   sourceKey: string,
 *   destinationBucket: string,
 *   destinationKey: string }} config
 */
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
```

```
  destinationKey,
}) => {
  const client = new S3Client({});

  try {
    await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucket}/${sourceKey}`,
        Bucket: destinationBucket,
        Key: destinationKey,
      }),
    );
    await waitUntilObjectExists(
      { client },
      { Bucket: destinationBucket, Key: destinationKey },
    );
    console.log(
      `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/
${destinationKey}`,
    );
  } catch (caught) {
    if (caught instanceof ObjectNotInActiveTierError) {
      console.error(
        `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the
 active tier.`,
      );
    } else {
      throw caught;
    }
  }
};
```

Copie o objeto caso seu ETag não corresponda ao fornecido.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
// Optionally edit the default key name of the copied object in
 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
 destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;
  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
 "${sourceBucketName}":  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
```

```
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copie o objeto caso seu ETag não corresponda ao fornecido.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
 destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfNoneMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
```

```
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
 "${sourceBucketName}":  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};
```

```
if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copie o objeto usando a condição de que ele tenha sido criado ou modificado em determinado período.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
 destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const name = data.name;
  const client = new S3Client({});
```

```
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
  const copiedKey = name + sourceKeyName;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfModifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
```

```
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copie o objeto usando a condição de que ele não tenha sido criado ou modificado em determinado período.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
```

```
 * @param {{ sourceBucketName: string, sourceKeyName: string,
 destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);
  const client = new S3Client({});
  const name = data.name;
  const copiedKey = name + sourceKeyName;
  const copySource = `${sourceBucketName}/${sourceKeyName}`;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfUnmodifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
```

```
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- Para obter detalhes da API, consulte a CopyObjectreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cópia simples de um objeto.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando copia o objeto “sample.txt” do bucket “test-files” para o mesmo bucket, mas com uma nova chave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

Exemplo 2: este comando copia o objeto “sample.txt” do bucket “test-files” para o bucket “backup-files” com uma nova chave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Exemplo 3: este comando baixa o objeto “sample.txt” do bucket “test-files” em um arquivo local com o nome “local-sample.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Exemplo 4: um único objeto é baixado no arquivo especificado. O arquivo baixado encontra-se em c:\downloads\data\archive.zip.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder
 c:\downloads
```

Exemplo 5: todos os objetos que correspondem ao prefixo de chave especificado são baixados na pasta local. A hierarquia relativa de chaves será preservada como subpastas no local geral do download.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

- Para obter detalhes da API, consulte CopyObjectem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando copia o objeto “sample.txt” do bucket “test-files” para o mesmo bucket, mas com uma nova chave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

Exemplo 2: este comando copia o objeto “sample.txt” do bucket “test-files” para o bucket “backup-files” com uma nova chave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Exemplo 3: este comando baixa o objeto “sample.txt” do bucket “test-files” em um arquivo local com o nome “local-sample.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Exemplo 4: um único objeto é baixado no arquivo especificado. O arquivo baixado encontra-se em c:\downloads\data\archive.zip.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder
 c:\downloads
```

Exemplo 5: todos os objetos que correspondem ao prefixo de chave especificado são baixados na pasta local. A hierarquia relativa de chaves será preservada como subpastas no local geral do download.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

Exemplo 6: baixa um único objeto em um arquivo local usando o download paralelo de várias partes para melhorar o desempenho em arquivos grandes.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt -UseMultipartDownload
```

Exemplo 7: baixa um objeto grande usando o modo RANGE com um tamanho de peça personalizado de 64 MB. O modo RANGE usa solicitações de intervalo de bytes HTTP e funciona para todos os objetos, independentemente de como eles foram carregados.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile
 local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -
PartSize 64MB
```

Exemplo 8: baixa todos os objetos que correspondem ao prefixo de chave em uma pasta local usando o download de várias partes com downloads simultâneos de arquivos.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:
\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Exemplo 9: baixa um único objeto usando o download de várias partes com 20 conexões simultâneas para obter a máxima taxa de transferência em redes de alta largura de banda.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile
 local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Exemplo 10: baixa todos os objetos que correspondem ao prefixo de chave para uma pasta local usando o download de várias partes, continuando com as falhas de arquivos individuais em vez de abortá-las.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:
\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Para obter detalhes da API, consulte CopyObjectem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
```

```
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Copie um objeto usando uma solicitação condicional.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
 request.

        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
 'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
```

```
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
 {dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Para obter detalhes da API, consulte a CopyObjectReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie um objeto.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
```

```
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copie um objeto e adicione criptografia do lado do servidor ao objeto de destino.

```
require 'aws-sdk-s3'
```

```
# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                          copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
 target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
 target_encryption)
  return unless target_object
```

```
  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
 with etag {}",
```

```
        response
            .copy_object_result
            .unwrap_or_else(||
 aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}
```

- Para obter detalhes da API, consulte a CopyObjectreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_dest_bucket
          iv_key = iv_dest_object
          iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).
        MESSAGE 'Object copied to another bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CopyObjectreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
            _ = try await client.copyObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CopyObjecta referência da API AWS SDK for Swift.

Use **`CreateBucket`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateBucket`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos de arquivos de atributos de machine learning
- Introdução ao armazenamento de objetos
- Gerenciar mensagens grandes usando o S3
- Trabalhar com objetos versionados

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
```

```
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK para .NET da API.

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket com o bloqueio de objetos habilitado.

```
    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);
```

```
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
```

```
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte CreateBucketem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                        clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
                  " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: como criar um bucket

O seguinte exemplo de `create-bucket` cria um bucket chamado `amzn-s3-demo-bucket`:

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket \
    --region us-east-1
```

Saída:

```
{
    "Location": "/amzn-s3-demo-bucket"
}
```

Para obter mais informações, consulte Como criar um bucket no Guia do usuário do Amazon S3.

Exemplo 2: como criar um bucket com aplicação de políticas de proprietário

O exemplo de `create-bucket` a seguir cria um bucket chamado `amzn-s3-demo-bucket` que usa a configuração aplicada de proprietário de bucket para a propriedade de objetos do S3.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket \
    --region us-east-1 \
    --object-ownership BucketOwnerEnforced
```

Saída:

```
{
```

```
    "Location": "/amzn-s3-demo-bucket"
}
```

Para obter mais informações, consulte Controlar a propriedade de objetos e desabilitar ACLs, no Guia do usuário do Amazon S3.

Exemplo 3: como criar um bucket fora da região ``us-east-1``

O exemplo de `create-bucket` a seguir cria um bucket chamado `amzn-s3-demo-bucket` na região `eu-west-1`. Regiões fora da `us-east-1` exigem que o `LocationConstraint` adequado seja especificado para poder criar o bucket na região desejada.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket \
    --region eu-west-1 \
    --create-bucket-configuration LocationConstraint=eu-west-1
```

Saída:

```
{
    "Location": "http://amzn-s3-demo-bucket.s3.amazonaws.com/"
}
```

Para obter mais informações, consulte Como criar um bucket no Guia do usuário do Amazon S3.

- Para obter detalhes da API, consulte CreateBucketem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar um bucket com a configuração padrão.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
 string) error {
 _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", name)
   err = owned
```

```
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", name)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
  }
 }
 return err
}
```

Crie um bucket com bloqueio de objetos e espere que ele exista.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}
```

- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket.

```
   /**
    * Creates an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to create
    * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
    * @throws RuntimeException if there is a failure while creating the bucket
    */
   public CompletableFuture<Void> createBucketAsync(String bucketName) {
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .build();

       CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
       return response.thenCompose(resp -> {
           S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
           HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
               .bucket(bucketName)
               .build();

           CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
               s3Waiter.waitUntilBucketExists(bucketRequestWait);
           return waiterResponseFuture.thenAccept(waiterResponse -> {
               waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                   logger.info(bucketName + " is ready");
               });
           });
```

```
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }
```

Crie um bucket com o bloqueio de objetos habilitado.

```
    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
 bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }
```

- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o bucket.

```
import {
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  CreateBucketCommand,
  S3Client,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
 constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket
 names must be globally unique.`,
      );
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`,
      );
```

```
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- Para obter detalhes da API, consulte a CreateBucketreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket com as configurações padrão.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in
 the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
 is
                                not specified, the Region configured in your
 shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
 region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
 region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error
```

Crie um bucket versionado com uma configuração de ciclo de vida.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
```

```
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

- Para obter detalhes da API, consulte a CreateBucketReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else
```

```
      @bucket.client.get_bucket_location(bucket:
 @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
 S3ExampleError> {
```

```
    let constraint =
 aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
 task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
 se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}
```

- Para obter detalhes da API, consulte a CreateBucketreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        " determine our region from our session
        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
        DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
        " When in the us-east-1 region, you must not specify a constraint
        " In all other regions, specify the region as the constraint
        IF lv_region = 'us-east-1'.
          CLEAR lo_constraint.
        ELSE.
          lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
        ENDIF.

        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
            io_createbucketconfiguration  = lo_constraint ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateBucketreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func createBucket(name: String) async throws {
        var input = CreateBucketInput(
            bucket: name
```

```
        )

        // For regions other than "us-east-1", you must set the
 locationConstraint in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
 S3ClientTypes.CreateBucketConfiguration(locationConstraint:
 S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
            print("The bucket '\(name)' already exists and is owned by you. You
 may wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateBucketa referência da API AWS SDK for Swift.

Use **`CreateMultiRegionAccessPoint`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateMultiRegionAccessPoint`.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o cliente de controle do S3 para enviar a solicitação à região us-west-2.

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West
(Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

Crie o ponto de Multi-Region acesso.

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
```

```
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

Aguarde até que o ponto de Multi-Region acesso fique disponível.

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
```

```
            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Kotlin.
- Para obter detalhes da API, consulte a CreateMultiRegionAccessPointreferência da API AWS SDK for Kotlin.

Use **`CreateMultipartUpload`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateMultipartUpload`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Executar uma cópia multipart
- Usar somas de verificação
- Trabalhar com a integridade de objetos do Amazon S3

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
 &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}
```

- Para obter detalhes da API, consulte CreateMultipartUploada Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando cria um upload fracionado no bucket `amzn-s3-demo-bucket` com a chave `multipart/01`:

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key
 'multipart/01'
```

Saída:

```
{
    "Bucket": "amzn-s3-demo-bucket",
    "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
    "Key": "multipart/01"
}
```

O arquivo concluído terá o nome `01` e estará em uma pasta chamada `multipart` no bucket `amzn-s3-demo-bucket`. Salve o ID de upload, a chave e o nome do bucket para usar com o comando `upload-part`.

- Para obter detalhes da API, consulte CreateMultipartUploadem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
```

```
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
```

```
        );
    }
```

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

- Para obter detalhes da API, consulte a CreateMultipartUploadreferência da API AWS SDK for Rust.

Use **`CreatePresignedPost`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreatePresignedPost`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um URL POST pré-assinado.

```
    /// <summary>
    /// Create a presigned POST URL with conditions.
```

```
    /// </summary>
    /// <param name="s3Client">The Amazon S3 client.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <param name="objectKey">The object key (path) where the uploaded file
 will be stored.</param>
    /// <param name="expires">When the presigned URL expires.</param>
    /// <param name="fields">Dictionary of fields to add to the form.</param>
    /// <param name="conditions">List of conditions to apply.</param>
    /// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
    public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
        IAmazonS3 s3Client,
        string bucketName,
        string objectKey,
        DateTime expires,
        Dictionary<string, string>? fields = null,
        List<S3PostCondition>? conditions = null)
    {
        var request = new CreatePresignedPostRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Expires = expires
        };

        // Add custom fields if provided
        if (fields != null)
        {
            foreach (var field in fields)
            {
                request.Fields.Add(field.Key, field.Value);
            }
        }

        // Add conditions if provided
        if (conditions != null)
        {
            foreach (var condition in conditions)
            {
                request.Conditions.Add(condition);
            }
        }

        return await s3Client.CreatePresignedPostAsync(request);
```

```
    }
```

- Para obter detalhes da API, consulte CreatePresignedPosta Referência AWS SDK para .NET da API.

Use **`DeleteBucket`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucket`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos de arquivos de atributos de machine learning
- Introdução ao armazenamento de objetos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Shows how to delete an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
    /// <returns>A boolean value that represents the success or failure of
    /// the delete operation.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
```

```
        try
        {
            var request = new DeleteBucketRequest { BucketName = bucketName, };

            await _amazonS3.DeleteBucketAsync(request);
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteBucketa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_bucket
#
```

```
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte DeleteBucketem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {

    Aws::S3::S3Client client(clientConfig);
```

```
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteBucketa Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando exclui o bucket `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

- Para obter detalhes da API, consulte DeleteBucketem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
 error {
 _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucketName)
   err = noBucket
  } else {
   log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
```

```
  }
 } else {
  err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
 bucketName)
  } else {
   log.Printf("Deleted %s.\n", bucketName)
  }
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteBucketa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an S3 bucket asynchronously.
     *
     * @param bucket the name of the bucket to be deleted
     * @return a {@link CompletableFuture} that completes when the bucket
 deletion is successful, or throws a {@link RuntimeException}
     * if an error occurs during the deletion process
     */
    public CompletableFuture<Void> deleteBucketAsync(String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();
```

```
        CompletableFuture<DeleteBucketResponse> response =
 getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during
 bucket deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }
```

- Para obter detalhes da API, consulte DeleteBucketa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua o bucket.

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
```

```
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting the bucket. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteBucketa Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um bucket vazio.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- Para obter detalhes da API, consulte DeleteBucketa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove todos os objetos e versões de objetos do bucket “test-files” e, em seguida, exclui o bucket. O comando solicitará a confirmação antes de continuar. Adicione a opção -Force para ignorar a confirmação. Observe que os buckets que não estão vazios não podem ser excluídos.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Para obter detalhes da API, consulte DeleteBucketem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todos os objetos e versões de objetos do bucket “test-files” e, em seguida, exclui o bucket. O comando solicitará a confirmação antes de continuar. Adicione a opção -Force para ignorar a confirmação. Observe que os buckets que não estão vazios não podem ser excluídos.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Para obter detalhes da API, consulte DeleteBucketem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteBucketReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Para obter detalhes da API, consulte DeleteBucketa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DeleteBucketreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.

        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteBucketreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
```

```
        do {
            _ = try await client.deleteBucket(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a bucket"))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteBucketa referência da API AWS SDK for Swift.

Usar **`DeleteBucketAnalyticsConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketAnalyticsConfiguration`.

CLI

AWS CLI

Para excluir uma configuração de análise de um bucket

O exemplo `delete-bucket-analytics-configuration` a seguir recupera a configuração referente ao bucket e ID especificados.

```
aws s3api delete-bucket-analytics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteBucketAnalyticsConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando remove o filtro de análise com o nome “testfilter” no bucket do S3 em questão.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -
AnalyticsId 'testfilter'
```

- Para obter detalhes da API, consulte DeleteBucketAnalyticsConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando remove o filtro de análise com o nome “testfilter” no bucket do S3 em questão.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -
AnalyticsId 'testfilter'
```

- Para obter detalhes da API, consulte DeleteBucketAnalyticsConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteBucketCors`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketCors`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Deletes a CORS configuration from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to delete the CORS configuration from the bucket.</param>
        private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
        {
```

```
            DeleteCORSConfigurationRequest request = new
 DeleteCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            await client.DeleteCORSConfigurationAsync(request);
        }
```

- Para obter detalhes da API, consulte DeleteBucketCorsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O comando a seguir exclui uma configuração Cross-Origin de compartilhamento de recursos de um bucket chamado`amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeleteBucketCorsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
```

```
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteBucketCorsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
```

```
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obter detalhes da API, consulte DeleteBucketCorsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_s3->deletebucketcors(
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
```

```
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteBucketCorsreferência da API AWS SDK for SAP ABAP.

Usar **`DeleteBucketEncryption`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketEncryption`.

CLI

AWS CLI

Para excluir a configuração de criptografia do lado do servidor de um bucket

O exemplo `delete-bucket-encryption` a seguir exclui a configuração de criptografia do lado do servidor do bucket especificado.

```
aws s3api delete-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteBucketEncryptionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: isso desabilita a criptografia habilitada para o bucket do S3 fornecido.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteBucketEncryptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: isso desabilita a criptografia habilitada para o bucket do S3 fornecido.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteBucketEncryptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteBucketInventoryConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketInventoryConfiguration`.

CLI

AWS CLI

Para excluir a configuração de inventário de um bucket

O exemplo `delete-bucket-inventory-configuration` a seguir exclui a configuração do inventário com ID 1 do bucket especificado.

```
aws s3api delete-bucket-inventory-configuration \
```

```
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteBucketInventoryConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse comando remove o inventário chamado InventoryName 'test' correspondente ao determinado bucket do S3.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -
InventoryId 'testInventoryName'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteBucketInventoryConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse comando remove o inventário chamado InventoryName 'test' correspondente ao determinado bucket do S3.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -
InventoryId 'testInventoryName'
```

Saída:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteBucketInventoryConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteBucketLifecycle`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketLifecycle`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// This method removes the Lifecycle configuration from the named
        /// S3 bucket.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the RemoveLifecycleConfigAsync method.</param>
        /// <param name="bucketName">A string representing the name of the
        /// S3 bucket from which the configuration will be removed.</param>
        public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
 string bucketName)
        {
            var request = new DeleteLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
```

- Para obter detalhes da API, consulte DeleteBucketLifecyclea Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando exclui a configuração de ciclo de vida do bucket `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeleteBucketLifecycleem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteBucketLifecycleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_s3->deletebucketlifecycle(
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteBucketLifecyclereferência da API AWS SDK for SAP ABAP.

Usar **`DeleteBucketMetricsConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketMetricsConfiguration`.

CLI

AWS CLI

Para excluir uma configuração de métricas referente a um bucket

O exemplo `delete-bucket-metrics-configuration` a seguir remove a configuração referente ao bucket e ID especificados.

```
aws s3api delete-bucket-metrics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 123
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteBucketMetricsConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando remove o filtro de métricas com o nome “testmetrics” no bucket do S3 em questão.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- Para obter detalhes da API, consulte DeleteBucketMetricsConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando remove o filtro de métricas com o nome “testmetrics” no bucket do S3 em questão.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- Para obter detalhes da API, consulte DeleteBucketMetricsConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteBucketPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketPolicy`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
 client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte DeleteBucketPolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando exclui uma política de bucket do bucket `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeleteBucketPolicyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from
 (for example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.
     *
     * @param s3 the {@link S3Client} instance to use for the operation
     * @param bucketName the name of the S3 bucket for which the policy should be
 deleted
     *
     * @throws S3Exception if there is an error deleting the bucket policy
     */
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();
```

```
        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteBucketPolicya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a política de bucket.

```
import {
  DeleteBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
```

```
    await client.send(
      new DeleteBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Bucket policy deleted from "${bucketName}".`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteBucketPolicya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- Para obter detalhes da API, consulte a DeleteBucketPolicyreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando remove a política de bucket associada ao bucket do S3 em questão.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte DeleteBucketPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando remove a política de bucket associada ao bucket do S3 em questão.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte DeleteBucketPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteBucketPolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
 why: #{e.message}"
    false
  end

end
```

- Para obter detalhes da API, consulte DeleteBucketPolicya Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_s3->deletebucketpolicy(
      iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket policy deleted.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteBucketPolicyreferência da API AWS SDK for SAP ABAP.

Usar **`DeleteBucketReplication`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketReplication`.

CLI

AWS CLI

O seguinte comando exclui uma configuração de replicação de um bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeleteBucketReplicationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui a configuração de replicação associada ao bucket denominado “amzn-s3-demo-bucket”. Observe que essa operação requer permissão para a DeleteReplicationConfiguration ação s3:. Será solicitada uma confirmação antes que a operação continue. Para ignorar a confirmação, use a opção -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeleteBucketReplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui a configuração de replicação associada ao bucket denominado “amzn-s3-demo-bucket”. Observe que essa operação requer permissão para a DeleteReplicationConfiguration ação s3:. Será solicitada uma confirmação antes que a operação continue. Para ignorar a confirmação, use a opção -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeleteBucketReplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteBucketTagging`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketTagging`.

CLI

AWS CLI

O seguinte comando exclui uma configuração de marcação de um bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeleteBucketTaggingem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove todas as tags associadas ao bucket do S3 em questão.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteBucketTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todas as tags associadas ao bucket do S3 em questão.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteBucketTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteBucketWebsite`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteBucketWebsite`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
            client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteBucketWebsitea Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando exclui uma configuração de site do bucket `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeleteBucketWebsiteem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the website
 configuration from.
            """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket:
 %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     * @param bucketName The name of the S3 bucket for which the website
 configuration should be deleted.
     * @throws S3Exception If an error occurs while deleting the website
 configuration.
     */
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
 {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteBucketWebsitea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a configuração de site do bucket.

```
import {
  DeleteBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the website configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    // The response code will be successful for both removed configurations and
    // configurations that did not exist in the first place.
    console.log(
      `The bucket "${bucketName}" is not longer configured as a website, or it
 never was.`,
    );
```

```
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
 The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteBucketWebsitea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando desabilita a propriedade de hospedagem de site estático do bucket do S3 em questão.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "amzn-s3-demo-bucket".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteBucketWebsiteem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando desabilita a propriedade de hospedagem de site estático do bucket do S3 em questão.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteBucketWebsiteem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteObject`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Trabalhar com a integridade de objetos do Amazon S3
- Trabalhar com objetos versionados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um objeto de um bucket do S3 sem versionamento.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete an object from a non-versioned Amazon
    /// Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteObject
    {
        /// <summary>
        /// The Main method initializes the necessary variables and then calls
        /// the DeleteObjectNonVersionedBucketAsync method to delete the object
        /// named by the keyName parameter.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";
            const string keyName = "testfile.txt";

            // If the Amazon S3 bucket is located in an AWS Region other than the
            // Region of the default account, define the AWS Region for the
            // Amazon S3 bucket in your call to the AmazonS3Client constructor.
            // For example RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();
            await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
        }

        /// <summary>
```

```
        /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
        /// desired object from the named bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to delete
        /// an object from an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// object will be deleted.</param>
        /// <param name="keyName">The name of the object to delete.</param>
        public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
 client, string bucketName, string keyName)
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                Console.WriteLine($"Deleting object: {keyName}");
                await client.DeleteObjectAsync(deleteObjectRequest);
                Console.WriteLine($"Object: {keyName} deleted from
 {bucketName}.");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
 Message:'{ex.Message}' when deleting an object.");
            }
        }
    }
```

Exclua um objeto de um bucket do S3 com versionamento.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example creates an object in an Amazon Simple Storage Service
```

```
    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
keyName);

                // Delete the object by specifying an object key and a version
ID.
                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
```

```
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be
 used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// will be created.</param>
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
        public static async Task<string> PutAnObject(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!",
            };

            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjecta Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - The key (file name) in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_item_in_bucket() {
  local bucket_name=$1
  local key=$2
  local response

  response=$(aws s3api delete-object \
    --bucket "$bucket_name" \
    --key "$key")

  # shellcheck disable=SC2181
```

```
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte DeleteObjectem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
            .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteObjecta Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando exclui um objeto `test.txt` do bucket `amzn-s3-demo-bucket`:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

Se o versionamento de bucket estiver habilitado, a saída conterá o ID de versão do marcador excluído:

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tNDlWlGq",
  "DeleteMarker": true
}
```

Para obter mais informações sobre como excluir objetos, consulte Excluir objetos no Guia do desenvolvedor do Amazon S3.

- Para obter detalhes da API, consulte DeleteObjectem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
```

```
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
```

```
    case "AccessDenied":
     log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
     err = nil
    case "InvalidArgument":
     if bypassGovernance {
      log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
      err = nil
     }
    }
   }
  } else {
   err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
 time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
   } else {
    deleted = true
   }
  }
  return deleted, err
 }
```

- Para obter detalhes da API, consulte DeleteObjecta Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an object from an S3 bucket asynchronously.
```

```
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the key (file name) of the object to be deleted
     * @return a {@link CompletableFuture} that completes when the object has
been deleted
     */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during
delete", ex);
            }
        });

        return response.thenApply(r -> null);
    }
```

- Para obter detalhes da API, consulte DeleteObjecta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um objeto.

```
import {
  DeleteObjectCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
 exist.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
```

```
        `Error from S3 while deleting object from ${bucketName}.  ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteObjecta Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public function deleteObject(string $bucketName, string $fileName, array
$args = [])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
       try {
           $this->client->deleteObject($parameters);
           if ($this->verbose) {
               echo "Deleted the object named: $fileName from $bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object deletion before continuing.";
           }
           throw $exception;
```

```
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjecta Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um objeto.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def delete(self):
        """
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
```

```
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
```

Reverta um objeto para uma versão anterior excluindo versões posteriores do objeto.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
```

```
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )
```

Restaure um objeto excluído removendo o marcador de exclusão ativo do objeto.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )
```

```
    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

Crie um manipulador do Lambda que remova um marcador de exclusão de um objeto do S3. Esse manipulador pode ser usado para limpar com eficiência marcadores de exclusão estranhos em um bucket versionado.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")
```

```
def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of
 the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
 obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
 delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
```

```
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
 marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
 obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
 retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
```

```
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

- Para obter detalhes da API, consulte a DeleteObjectReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// Delete an object from a bucket.
pub async fn remove_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    key: &str,
) -> Result<(), S3ExampleError> {
```

```
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    // There are no modeled errors to handle when deleting an object.

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteObjectreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_object_key ).
        MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteObjectreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a file."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjecta referência da API AWS SDK for Swift.

Usar **`DeleteObjectTagging`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteObjectTagging`.

CLI

AWS CLI

Para excluir os conjuntos de tags de um objeto

O exemplo `delete-object-tagging` a seguir exclui a tag com a chave especificada do objeto `doc1.rtf`.

```
aws s3api delete-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteObjectTaggingem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove todas as tags associadas ao objeto com a chave “testfile.txt” no bucket do S3 em questão.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -
Select '^Key'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- Para obter detalhes da API, consulte DeleteObjectTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todas as tags associadas ao objeto com a chave “testfile.txt” no bucket do S3 em questão.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -
Select '^Key'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- Para obter detalhes da API, consulte DeleteObjectTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteObjects`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteObjects`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Excluir todos os objetos em um bucket
- Introdução ao armazenamento de objetos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete all of the objects stored in an existing Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket from which the
    /// contents will be deleted.</param>
    /// <returns>A boolean value that represents the success or failure of
```

```
    /// deleting all of the objects in the bucket.</returns>
    public async Task<bool> DeleteBucketContentsAsync(string bucketName)
    {
        // Iterate over the contents of the bucket and delete all objects.
        try
        {
            // Delete all objects in the bucket.
            var deleteList = await ListBucketContentsAsync(bucketName, false);
            if (deleteList != null && deleteList.Any())
            {
                await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
                });
            }

            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK para .NET da API.

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua vários objetos de um bucket do S3 sem versionamento.

```
    using System;
```

```
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete multiple objects from an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        /// <summary>
        /// The Main method initializes the Amazon S3 client and the name of
        /// the bucket and then passes those values to MultiObjectDeleteAsync.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";

            // If the Amazon S3 bucket from which you wish to delete objects is
not
            // located in the same AWS Region as the default user, define the
            // AWS Region for the Amazon S3 bucket as a parameter to the client
            // constructor.
            IAmazonS3 s3Client = new AmazonS3Client();

            await MultiObjectDeleteAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method uses the passed Amazon S3 client to first create and then
        /// delete three files from the named bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// Amazon S3 methods.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
        /// will be created and then deleted.</param>
        public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
        {
            // Create three sample objects which we will then delete.
            var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);
```

```
            // Now perform the multi-object delete, passing the key names and
            // version IDs. Since we are working with a non-versioned bucket,
            // the object keys collection includes null version IDs.
            DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keysAndVersions,
            };

            // You can add a specific object key to the delete request using the
            // AddKey method of the multiObjectDeleteRequest.
            try
            {
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
        }

        /// <summary>
        /// Prints the list of errors raised by the call to DeleteObjectsAsync.
        /// </summary>
        /// <param name="ex">A collection of exceptions returned by the call to
        /// DeleteObjectsAsync.</param>
        public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
        {
            DeleteObjectsResponse errorResponse = ex.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

            Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
```

```
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// This method creates simple text file objects that can be used in
        /// the delete method.
        /// </summary>
        /// <param name="client">The Amazon S3 client used to call
PutObjectAsync.</param>
        /// <param name="number">The number of objects to create.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created.</param>
        /// <returns>A list of keys (object keys) and versions that the calling
        /// method will use to delete the newly created files.</returns>
        public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
        {
            List<KeyVersion> keys = new List<KeyVersion>();
            for (int i = 0; i < number; i++)
            {
                string key = "ExampleObject-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                PutObjectResponse response = await
client.PutObjectAsync(request);

                // For non-versioned bucket operations, we only need the
                // object key.
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                };
                keys.Add(keyVersion);
            }

            return keys;
        }
```

```
    }
```

Exclua vários objetos de um bucket do S3 com versionamento.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete objects in a version-enabled Amazon
    /// Simple StorageService (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";

            // If the AWS Region for your Amazon S3 bucket is different from
            // the AWS Region of the default user, define the AWS Region for
            // the Amazon S3 bucket and pass it to the client constructor
            // like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 s3Client;

            s3Client = new AmazonS3Client();
            await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
 bucketName);
        }

        /// <summary>
        /// This method removes multiple versions and objects from a
        /// version-enabled Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
```

```
        /// objects.</param>
        public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
        {
            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync(client, bucketName);

            // Delete objects (without specifying object version in the request).
            var deletedObjects = await DeleteObjectsAsync(client, bucketName);

            // Additional exercise - remove the delete markers Amazon S3 returned
from
            // the preceding response. This results in the objects reappearing
            // in the bucket (you can verify the appearance/disappearance of
            // objects in the console).
            await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
        }

        /// <summary>
        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects
deleted.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker
and
            // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }
```

```
        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }

        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
            Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// Delete multiple objects from a version-enabled bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
```

```
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        private static async Task VersionedDeleteAsync(IAmazonS3 client, string
 bucketName, List<KeyVersion> keys)
        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keys, // This includes the object keys and specific
 version IDs.
            };

            try
            {
                Console.WriteLine("Executing VersionedDelete...");
                DeleteObjectsResponse response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
 {response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
        private static async Task<List<DeletedObject>>
 NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
 keys)
        {
            // Create a request that includes only the object key names.
```

```
            DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)
            {
                multiObjectDeleteRequest.AddKey(key.Key);
            }

            // Execute DeleteObjectsAsync.
            // The DeleteObjectsAsync method adds a delete marker for each
            // object deleted. You can verify that the objects were removed
            // using the Amazon S3 console.
            DeleteObjectsResponse response;
            try
            {
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
                throw; // Some deletions failed. Investigate before continuing.
            }

            // This response contains the DeletedObjects list which we use to
delete the delete markers.
            return response.DeletedObjects;
        }

        /// <summary>
        /// Deletes the markers left after deleting the temporary objects.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="deletedObjects">A list of the objects that were
deleted.</param>
```

```
        private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
        {
            var keyVersionList = new List<KeyVersion>();

            foreach (var deletedObject in deletedObjects)
            {
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = deletedObject.Key,
                    VersionId = deletedObject.DeleteMarkerVersionId,
                };
                keyVersionList.Add(keyVersion);
            }

            // Create another request to delete the delete markers.
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keyVersionList,
            };

            // Now, delete the delete marker to bring your objects back to the
bucket.
            try
            {
                Console.WriteLine("Removing the delete markers .....");
                var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Create temporary Amazon S3 objects to show how object deletion wors
in an
        /// Amazon S3 bucket with versioning enabled.
        /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync to create temporary objects for the example.</param>
        /// <param name="bucketName">A string representing the name of the S3
        /// bucket where we will create the temporary objects.</param>
        /// <param name="number">The number of temporary objects to create.</
param>
        /// <returns>A list of the KeyVersion objects.</returns>
        private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
 client, string bucketName, int number)
        {
            var keys = new List<KeyVersion>();

            for (var i = 0; i < number; i++)
            {
                string key = "ObjectToDelete-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                var response = await client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    VersionId = response.VersionId,
                };

                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
```

```
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte DeleteObjectsem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

 deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }
```

```
    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
            client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }

        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando exclui um objeto do bucket `amzn-s3-demo-bucket`:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` é um documento JSON no diretório atual que especifica o objeto a ser excluído:

```
{
```

```
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Saída:

```
{
    "Deleted": [
        {
            "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUPO1koHwzU",
            "Key": "test1.txt",
            "DeleteMarker": true
        }
    ]
}
```

- Para obter detalhes da API, consulte DeleteObjectsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
```

```
    }
    err = fmt.Errorf("%s", *delOut.Errors[0].Message)
   }
  } else {
   for _, delObjs := range delOut.Deleted {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
     ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
  time.Minute)
    if err != nil {
     log.Printf("Failed attempt to wait for object %s to be deleted.\n",
  *delObjs.Key)
    } else {
     log.Printf("Deleted %s.\n", *delObjs.Key)
    }
   }
  }
  return err
 }
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName>

            Where:
               bucketName - the Amazon S3 bucket name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
```

```
    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                .key(keyName)
                .build();

            putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
            .objects(keys)
            .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(del)
                .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua vários objetos.

```
import {
  DeleteObjectsCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
```

```
        { client },
        { Bucket: bucketName, Key: key },
      );
    }
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
 objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- Para obter detalhes da API, consulte a DeleteObjectsreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um conjunto de objetos de uma lista de chaves.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove o objeto “sample.txt” do bucket “test-files”. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a solicitação, use a opção -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 2: este comando remove a versão especificada do objeto “sample.txt” do bucket “test-files”, supondo que o bucket tenha sido configurado para habilitar versões de objetos.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemplo 3: este comando remove objetos “sample1.txt”, “sample2.txt” e “sample3.txt” do bucket “test-files” como uma operação em lote única. A resposta do serviço listará todas as chaves processadas, independentemente do status de êxito ou erro da exclusão. Para obter somente erros para chaves que não puderam ser processadas pelo serviço, adicione o ReportErrorsOnly parâmetro - (esse parâmetro também pode ser especificado com o alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Exemplo 4: Este exemplo usa uma expressão embutida com o KeyCollection parâmetro - para obter as chaves dos objetos a serem excluídos. Get-S3Object retorna uma coleção de Amazon.S3.Model.S3Object instâncias, cada uma com um membro-chave do tipo string identificando o objeto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Exemplo 5: Esse exemplo obtém todos os objetos que têm um prefixo de chave prefix/subprefix "" no bucket e os exclui. Observe que os objetos de entrada são processados um de cada vez. Para coleções grandes, considere passar a coleção para o parâmetro - InputObject (alias -S3ObjectCollection) do cmdlet para permitir que a exclusão ocorra como um lote com uma única chamada para o serviço.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

Exemplo 6: Este exemplo canaliza uma coleção de Amazon.S3.Model.S3ObjectVersion instâncias que representam marcadores de exclusão para o cmdlet para exclusão. Observe que os objetos de entrada são processados um de cada vez. Para coleções grandes, considere passar a coleção para o parâmetro - InputObject (alias -S3ObjectCollection) do

cmdlet para permitir que a exclusão ocorra como um lote com uma única chamada para o serviço.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemplo 7: Esse script mostra como realizar uma exclusão em lote de um conjunto de objetos (nesse caso, marcadores de exclusão) construindo uma matriz de objetos a serem usados com o parâmetro -KeyAndVersionCollection .

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- Para obter detalhes da API, consulte DeleteObjectsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove o objeto “sample.txt” do bucket “test-files”. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a solicitação, use a opção -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 2: este comando remove a versão especificada do objeto “sample.txt” do bucket “test-files”, supondo que o bucket tenha sido configurado para habilitar versões de objetos.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemplo 3: este comando remove objetos “sample1.txt”, “sample2.txt” e “sample3.txt” do bucket “test-files” como uma operação em lote única. A resposta do serviço listará todas as chaves processadas, independentemente do status de êxito ou erro da exclusão. Para obter somente erros para chaves que não puderam ser processadas pelo serviço, adicione o

ReportErrorsOnly parâmetro - (esse parâmetro também pode ser especificado com o alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Exemplo 4: Este exemplo usa uma expressão embutida com o KeyCollection parâmetro - para obter as chaves dos objetos a serem excluídos. Get-S3Object retorna uma coleção de Amazon.S3.Model.S3Object instâncias, cada uma com um membro-chave do tipo string identificando o objeto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Exemplo 5: Esse exemplo obtém todos os objetos que têm um prefixo de chave prefix/subprefix "" no bucket e os exclui. Observe que os objetos de entrada são processados um de cada vez. Para coleções grandes, considere passar a coleção para o parâmetro - InputObject (alias -S3ObjectCollection) do cmdlet para permitir que a exclusão ocorra como um lote com uma única chamada para o serviço.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

Exemplo 6: Este exemplo canaliza uma coleção de Amazon.S3.Model.S3ObjectVersion instâncias que representam marcadores de exclusão para o cmdlet para exclusão. Observe que os objetos de entrada são processados um de cada vez. Para coleções grandes, considere passar a coleção para o parâmetro - InputObject (alias -S3ObjectCollection) do cmdlet para permitir que a exclusão ocorra como um lote com uma única chamada para o serviço.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemplo 7: Esse script mostra como realizar uma exclusão em lote de um conjunto de objetos (nesse caso, marcadores de exclusão) construindo uma matriz de objetos a serem usados com o parâmetro -KeyAndVersionCollection .

```
$keyVersions = @()
```

```
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- Para obter detalhes da API, consulte DeleteObjectsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um conjunto de objetos de uma lista de chaves.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.
```

```
        :param bucket: The bucket that contains the objects. This is a Boto3
 Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                 and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
        except ClientError:
            logger.exception("Couldn't delete any objects from bucket %s.",
 bucket.name)
            raise
        else:
            return response
```

Exclua todos os objetos em um bucket.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
```

```
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise
```

Exclua permanentemente um objeto versionado excluindo todas as suas versões.

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- Para obter detalhes da API, consulte a DeleteObjectsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use `?` early return errors while building
 object keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for
 delete_object: {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }

    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(
            aws_sdk_s3::types::Delete::builder()
                .set_objects(Some(delete_object_ids))
                .build()
                .map_err(|err| {
                    S3ExampleError::new(format!("Failed to build delete_object
 input {err:?}"))
                })?,
        )
```

```
        .send()
        .await?;
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteObjectsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->deleteobjects(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
          io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ) ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteObjectsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func deleteObjects(bucket: String, keys: [String]) async throws {
        let input = DeleteObjectsInput(
            bucket: bucket,
            delete: S3ClientTypes.Delete(
                objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
                quiet: true
            )
        )

        do {
            _ = try await client.deleteObjects(input: input)
        } catch {
            print("ERROR: deleteObjects:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjectsa referência da API AWS SDK for Swift.

Usar **`DeletePublicAccessBlock`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeletePublicAccessBlock`.

CLI

AWS CLI

Para excluir a configuração do Bloqueio de Acesso Público para um bucket

O exemplo `delete-public-access-block` a seguir remove a configuração do Bloqueio de Acesso Público no bucket especificado.

```
aws s3api delete-public-access-block \
    --bucket amzn-s3-demo-bucket
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeletePublicAccessBlockem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando desativa a configuração do Bloqueio de Acesso Público do bucket em questão.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

Saída:

```
amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeletePublicAccessBlockem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando desativa a configuração do Bloqueio de Acesso Público do bucket em questão.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

Saída:

```
amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte DeletePublicAccessBlockem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetBucketAccelerateConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketAccelerateConfiguration`.

CLI

AWS CLI

Para recuperar a configuração de aceleração de um bucket

O exemplo `get-bucket-accelerate-configuration` a seguir recupera a configuração de aceleração do bucket especificado.

```
aws s3api get-bucket-accelerate-configuration \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "Status": "Enabled"
}
```

- Para obter detalhes da API, consulte GetBucketAccelerateConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: se as configurações de aceleração de transferência estiverem habilitadas para o bucket especificado, este comando retornará o valor Habilitado.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
```

```
-----
Enabled
```

- Para obter detalhes da API, consulte GetBucketAccelerateConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: se as configurações de aceleração de transferência estiverem habilitadas para o bucket especificado, este comando retornará o valor Habilitado.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
-----
Enabled
```

- Para obter detalhes da API, consulte GetBucketAccelerateConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetBucketAcl`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBucketAcl`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Verificar se um bucket existe
- Gerenciar listas de controle de acesso (ACLs)

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Get the access control list (ACL) for the new bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to get the
        /// access control list (ACL) of the bucket.</param>
        /// <param name="newBucketName">The name of the newly created bucket.</
param>
        /// <returns>An S3AccessControlList.</returns>
        public static async Task<S3AccessControlList>
 GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
        {
            // Retrieve bucket ACL to show that the ACL was properly applied to
            // the new bucket.
            GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = newBucketName,
            });

            return getACLResponse.AccessControlList;
        }
```

- Para obter detalhes da API, consulte GetBucketAcla Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
```

```
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
            s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) <<
std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }
```

```
            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
*/

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
*/
```

```
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}
```

- Para obter detalhes da API, consulte GetBucketAcla Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando recupera a lista de controle de acesso do bucket `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "Owner": {
        "DisplayName": "my-username",
```

```
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetBucketAclem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <objectKey>

            Where:
              bucketName - The Amazon S3 bucket to get the access control list
 (ACL) for.
              objectKey - The object to get the ACL for.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3
 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be
 retrieved.
     * @param bucketName The name of the bucket containing the object.
```

```
     * @return The ID of the grantee who has permission on the object, or an
 empty string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
 bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
 grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- Para obter detalhes da API, consulte GetBucketAcla Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha as permissões de ACL.

```
import {
  GetBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketAclCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`ACL for bucket "${bucketName}":`);
    console.log(JSON.stringify(response, null, 2));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
```

```
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetBucketAcla Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V5

Exemplo 1: o comando obtém os detalhes do proprietário do objeto do S3.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Saída:

```
DisplayName Id
----------- --
testusername
 9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Para obter detalhes da API, consulte GetBucketAclem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
 acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- Para obter detalhes da API, consulte a GetBucketAclReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getbucketacl(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetBucketAclreferência da API AWS SDK for SAP ABAP.

Usar **`GetBucketAnalyticsConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketAnalyticsConfiguration`.

CLI

AWS CLI

Para recuperar a configuração de análise de um bucket com um ID específico

O exemplo `get-bucket-analytics-configuration` a seguir recupera a configuração de análise referente ao bucket e ID especificados.

```
aws s3api get-bucket-analytics-configuration \
```

```
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Saída:

```
{
    "AnalyticsConfiguration": {
        "StorageClassAnalysis": {},
        "Id": "1"
    }
}
```

- Para obter detalhes da API, consulte GetBucketAnalyticsConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna os detalhes do filtro de análise com o nome “testfilter” no bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Para obter detalhes da API, consulte GetBucketAnalyticsConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes do filtro de análise com o nome “testfilter” no bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Para obter detalhes da API, consulte GetBucketAnalyticsConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetBucketCors`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBucketCors`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Retrieve the CORS configuration applied to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to retrieve the CORS configuration.</param>
        /// <returns>The created CORS configuration object.</returns>
        private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
        {
            GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            var response = await client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }
```

- Para obter detalhes da API, consulte GetBucketCorsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O comando a seguir recupera a configuração do compartilhamento de Cross-Origin recursos para um bucket chamado`amzn-s3-demo-bucket`:

```
aws s3api get-bucket-cors --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "CORSRules": [
        {
            "AllowedHeaders": [
                "*"
            ],
            "ExposeHeaders": [
                "x-amz-server-side-encryption"
            ],
            "AllowedMethods": [
                "PUT",
                "POST",
                "DELETE"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedOrigins": [
                "http://www.example.com"
            ]
        },
        {
            "AllowedHeaders": [
                "Authorization"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedMethods": [
                "GET"
            ],
            "AllowedOrigins": [
                "*"
            ]
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte GetBucketCorsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política de CORS para o bucket.

```
import {
  GetBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-".repeat(10)}`,
        `\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\nAllowedMethods: ${cr.AllowedMethods}`,
```

```
        `\nAllowedOrigins: ${cr.AllowedOrigins}`,
        `\nExposeHeaders: ${cr.ExposeHeaders}`,
        `\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. The
 bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetBucketCorsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.",
 self.bucket.name))
            raise
        else:
            return cors
```

- Para obter detalhes da API, consulte a GetBucketCorsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
 configuration for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- Para obter detalhes da API, consulte GetBucketCorsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getbucketcors(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetBucketCorsreferência da API AWS SDK for SAP ABAP.

Use **`GetBucketEncryption`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBucketEncryption`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get and print the encryption settings of a bucket.
```

```
    /// </summary>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task GetEncryptionSettings(string bucketName)
    {
        // Check and print the bucket encryption settings.
        Console.WriteLine($"Getting encryption settings for bucket
{bucketName}.");

        try
        {
            var settings =
                await _s3Client.GetBucketEncryptionAsync(
                    new GetBucketEncryptionRequest() { BucketName =
bucketName });

            foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
            {
                Console.WriteLine(
                    $"\tAlgorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
                Console.WriteLine(
                    $"\tKey:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementServi
            }
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                ? $"Bucket {bucketName} was not found."
                : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
        }
    }
```

- Para obter detalhes da API, consulte GetBucketEncryptiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar a configuração de criptografia do lado do servidor para um bucket

O exemplo `get-bucket-encryption` a seguir recupera a configuração de criptografia do lado do servidor do bucket `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }
        ]
    }
}
```

- Para obter detalhes da API, consulte GetBucketEncryptionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna todas as regras de criptografia do lado do servidor associadas ao bucket em questão.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketEncryptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todas as regras de criptografia do lado do servidor associadas ao bucket em questão.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketEncryptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetBucketInventoryConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketInventoryConfiguration`.

CLI

AWS CLI

Para recuperar a configuração de inventário de um bucket

O exemplo `get-bucket-inventory-configuration` a seguir recupera a configuração de inventário do bucket especificado com ID 1.

```
aws s3api get-bucket-inventory-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Saída:

```
{
    "InventoryConfiguration": {
        "IsEnabled": true,
        "Destination": {
            "S3BucketDestination": {
                "Format": "ORC",
                "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                "AccountId": "123456789012"
            }
        },
        "IncludedObjectVersions": "Current",
```

```
        "Id": "1",
        "Schedule": {
            "Frequency": "Weekly"
        }
    }
}
```

- Para obter detalhes da API, consulte GetBucketInventoryConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna os detalhes do inventário denominado “testinventory” para o bucket do S3 em questão.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- Para obter detalhes da API, consulte GetBucketInventoryConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes do inventário denominado “testinventory” para o bucket do S3 em questão.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- Para obter detalhes da API, consulte GetBucketInventoryConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetBucketLifecycleConfiguration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBucketLifecycleConfiguration`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Returns a configuration object for the supplied bucket name.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the GetLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">The name of the S3 bucket for which a
        /// configuration will be created.</param>
        /// <returns>Returns a new LifecycleConfiguration object.</returns>
        public static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
        {
            var request = new GetLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }
```

- Para obter detalhes da API, consulte GetBucketLifecycleConfigurationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando recupera a configuração do ciclo de vida do bucket `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 0,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetBucketLifecycleConfigurationem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
```

```
            return config.rules
```

- Para obter detalhes da API, consulte a GetBucketLifecycleConfigurationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getbucketlifecycleconf(         " oo_result is
returned for testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetBucketLifecycleConfigurationreferência da API AWS SDK for SAP ABAP.

Use **`GetBucketLocation`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBucketLocation`.

CLI

AWS CLI

O seguinte comando recupera a restrição de localização do bucket `amzn-s3-demo-bucket`, se houver uma restrição:

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "LocationConstraint": "us-west-2"
}
```

- Para obter detalhes da API, consulte GetBucketLocationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: se houver uma restrição, este comando retornará a restrição de localização do bucket “amzn-s3-demo-bucket”.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
-----
ap-south-1
```

- Para obter detalhes da API, consulte GetBucketLocationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: se houver uma restrição, este comando retornará a restrição de localização do bucket “amzn-s3-demo-bucket”.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
```

```
-----
ap-south-1
```

- Para obter detalhes da API, consulte GetBucketLocationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
```

```
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetBucketLocationreferência da API AWS SDK for Rust.

Usar **`GetBucketLogging`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketLogging`.

CLI

AWS CLI

Para recuperar o status de registro em log de um bucket

O exemplo `get-bucket-logging` a seguir recupera o status de registro em log do bucket especificado.

```
aws s3api get-bucket-logging \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
```

```
    "LoggingEnabled": {
        "TargetPrefix": "",
        "TargetBucket": "amzn-s3-demo-bucket-logs"
          }
}
```

- Para obter detalhes da API, consulte GetBucketLoggingem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna o status de registro em log do bucket especificado.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- Para obter detalhes da API, consulte GetBucketLoggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status de registro em log do bucket especificado.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- Para obter detalhes da API, consulte GetBucketLoggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetBucketMetricsConfiguration`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketMetricsConfiguration`.

CLI

AWS CLI

Para recuperar a configuração de métricas de um bucket com um ID específico

O exemplo `get-bucket-metrics-configuration` a seguir exibe a configuração de métricas referente ao bucket e ID especificados.

```
aws s3api get-bucket-metrics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 123
```

Saída:

```
{
    "MetricsConfiguration": {
        "Filter": {
            "Prefix": "logs"
        },
        "Id": "123"
    }
}
```

- Para obter detalhes da API, consulte GetBucketMetricsConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna detalhes sobre o filtro de métricas denominado “testfilter” para o bucket do S3 em questão.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- Para obter detalhes da API, consulte GetBucketMetricsConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna detalhes sobre o filtro de métricas denominado “testfilter” para o bucket do S3 em questão.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- Para obter detalhes da API, consulte GetBucketMetricsConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetBucketNotification`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketNotification`.

CLI

AWS CLI

O seguinte comando recupera a configuração de notificação do bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "TopicConfiguration": {
        "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
        "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
        "Event": "s3:ObjectCreated:*",
        "Events": [
            "s3:ObjectCreated:*"
        ]
    }
}
```

- Para obter detalhes da API, consulte GetBucketNotificationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera a configuração de notificação do bucket em questão.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

Saída:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Para obter detalhes da API, consulte GetBucketNotificationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a configuração de notificação do bucket em questão.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

Saída:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Para obter detalhes da API, consulte GetBucketNotificationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetBucketPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBucketPolicy`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
            s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
 <<
                  policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetBucketPolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando recupera política de bucket do bucket `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "Policy": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":\"\",\"Effect
\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":
\"arn:aws:s3:::amzn-s3-demo-bucket/*\"},{\"Sid\":\"\",\"Effect\":\"Deny\",
\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::amzn-
s3-demo-bucket/secret/*\"}]}"
}
```

Obter e colocar uma política de bucket em vigor O exemplo a seguir mostra como fazer download de uma política do Amazon S3, fazer alterações no arquivo e usar `put-bucket-policy` para aplicar a política de bucket modificada. Para fazer o download de uma política de bucket em um arquivo, você pode executar:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output
 text > policy.json
```

Em seguida, modifique o arquivo `policy.json` conforme necessário. Por fim, aplique essa política modificada ao bucket do S3 executando:

o arquivo `policy.json` conforme necessário. Por fim, aplique essa política modificada ao bucket do S3 executando:

conforme necessário. Por fim, aplique essa política modificada ao bucket do S3 executando:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json
```

- Para obter detalhes da API, consulte GetBucketPolicyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     * @param bucketName the name of the S3 bucket for which to retrieve the
policy
     * @return the policy text for the specified bucket, or an empty string if an
error occurs
     */
    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
```

```
    }
}
```

- Para obter detalhes da API, consulte GetBucketPolicya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política de bucket.

```
import {
  GetBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Policy } = await client.send(
      new GetBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
  } catch (caught) {
    if (
```

```
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetBucketPolicya Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }
```

```
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- Para obter detalhes da API, consulte a GetBucketPolicyreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando gera a política de bucket associada ao bucket do S3 em questão.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando gera a política de bucket associada ao bucket do S3 em questão.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketPolicyem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```

- Para obter detalhes da API, consulte a GetBucketPolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- Para obter detalhes da API, consulte GetBucketPolicya Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getbucketpolicy(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       DATA(lv_policy) = oo_result->get_policy( ).
       MESSAGE 'Retrieved bucket policy.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetBucketPolicyreferência da API AWS SDK for SAP ABAP.

Usar **`GetBucketPolicyStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketPolicyStatus`.

CLI

AWS CLI

Para recuperar o status da política de um bucket que indica se o bucket é público

O exemplo `get-bucket-policy-status` a seguir recupera o status da política do bucket `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-policy-status \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "PolicyStatus": {
        "IsPublic": false
    }
}
```

- Para obter detalhes da API, consulte GetBucketPolicyStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna o status da política do bucket do S3 em questão, indicando se o bucket é público.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketPolicyStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status da política do bucket do S3 em questão, indicando se o bucket é público.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketPolicyStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetBucketReplication`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBucketReplication`.

CLI

AWS CLI

O seguinte comando recupera a configuração de replicação de um bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "ReplicationConfiguration": {
        "Rules": [
            {
                "Status": "Enabled",
                "Prefix": "",
                "Destination": {
                    "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket-backup",
                    "StorageClass": "STANDARD"
                },
                "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
            }
        ],
        "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
    }
}
```

- Para obter detalhes da API, consulte GetBucketReplicationem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the replication details for the specified S3 bucket.
     *
     * @param s3Client           the S3 client used to interact with the S3
service
```

```
    * @param sourceBucketName   the name of the S3 bucket to retrieve the
replication details for
    *
    * @throws S3Exception if there is an error retrieving the replication
details
    */
   public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
       GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
           .bucket(sourceBucketName)
           .build();

       try {
           ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
           ReplicationRule rule = replicationConfig.rules().get(0);
           System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
           System.out.println("Retrieved priority: " + rule.priority());
           System.out.println("Retrieved source-bucket replication rule status:
" + rule.status());

       } catch (S3Exception e) {
           System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
       }
   }
```

- Para obter detalhes da API, consulte GetBucketReplicationa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna as informações da configuração de replicação definida no bucket denominado “amzn-s3-demo-bucket”.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte GetBucketReplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna as informações da configuração de replicação definida no bucket denominado “amzn-s3-demo-bucket”.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte GetBucketReplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetBucketRequestPayment`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketRequestPayment`.

CLI

AWS CLI

Para recuperar a configuração de pagamento de solicitação de um bucket

O exemplo `get-bucket-request-payment` a seguir recupera a configuração de pagamento de solicitação do bucket especificado.

```
aws s3api get-bucket-request-payment \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "Payer": "BucketOwner"
}
```

- Para obter detalhes da API, consulte GetBucketRequestPaymentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna as informações da configuração de pagamento de solicitação do bucket denominado “amzn-s3-demo-bucket”. Por padrão, o proprietário do bucket paga pelos downloads feitos no bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte GetBucketRequestPaymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna as informações da configuração de pagamento de solicitação do bucket denominado “amzn-s3-demo-bucket”. Por padrão, o proprietário do bucket paga pelos downloads feitos no bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte GetBucketRequestPaymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetBucketTagging`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketTagging`.

CLI

AWS CLI

O seguinte comando recupera a configuração de marcação de um bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Saída:

```
{
```

```
    "TagSet": [
        {
            "Value": "marketing",
            "Key": "organization"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetBucketTaggingem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna todas as tags associadas ao bucket em questão.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todas as tags associadas ao bucket em questão.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetBucketVersioning`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketVersioning`.

CLI

AWS CLI

O seguinte comando recupera a configuração de versionamento de um bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "Status": "Enabled"
}
```

- Para obter detalhes da API, consulte GetBucketVersioningem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna o status de versionamento referente ao bucket em questão.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketVersioningem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status de versionamento referente ao bucket em questão.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketVersioningem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetBucketWebsite`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetBucketWebsite`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
                // Get the website configuration.
                GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
                Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- Para obter detalhes da API, consulte GetBucketWebsitea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
```

```
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
            s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
 outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
                  << websiteResult.GetErrorDocument().GetKey()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetBucketWebsitea Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando recupera a configuração estática do site do bucket `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- Para obter detalhes da API, consulte GetBucketWebsiteem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a configuração do site.

```
import {
  GetBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
```

```
} from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:
\n${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}. The
 bucket isn't configured as a website.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte GetBucketWebsitea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna os detalhes das configurações do site estático do bucket do S3 em questão.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketWebsiteem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes das configurações do site estático do bucket do S3 em questão.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetBucketWebsiteem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetObject`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Obter um objeto de um bucket que foi modificado
- Obter um objeto de um ponto de Multi-Region acesso
- Conceitos básicos de criptografia
- Introdução ao armazenamento de objetos
- Fazer solicitações condicionais
- Monitorar uploads e downloads

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await
 _amazonS3.GetObjectAsync(request);

        try
        {
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK para .NET da API.

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um objeto usando uma solicitação condicional.

```
    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch',
 'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
 sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
 string? etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
```

```
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read
 successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional read failed: Precondition failed");
            }
            else if (e.ErrorCode == "NotModified")
            {
                _logger.LogError("Conditional read failed: Object not modified");
            }
```

```
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
```

```
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte GetObjectem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);
```

```
    Aws::S3::Model::GetObjectOutcome outcome =
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte exemplo usa o comando `get-object` para fazer download de um objeto do Amazon S3:

```
aws s3api get-object --bucket text-content --key dir/
my_images.tar.bz2 my_images.tar.bz2
```

Observe que o parâmetro outfile é especificado sem um nome de opção, como “--outfile”. O nome do arquivo de saída deve ser o último parâmetro no comando.

O exemplo abaixo demonstra o uso de `--range` para fazer download de um intervalo de bytes específico de um objeto. Observe que os intervalos de bytes precisam ser prefixados com “bytes=”:

```
aws s3api get-object --bucket text-content --key dir/my_data --
range bytes=8888-9999 my_data_range
```

Para obter mais informações sobre como recuperar objetos, consulte Obter um objeto no Guia do desenvolvedor do Amazon S3.

- Para obter detalhes da API, consulte GetObjectem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}
```

```
// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
 objectKey, bucketName)
   err = noKey
  } else {
   log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  }
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Leia dados como uma matriz de bytes usando um S3Client.

```
   /**
    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
and writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes
have been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
       return response.thenAccept(objectBytes -> {
           try {
               byte[] data = objectBytes.asByteArray();
               Path filePath = Paths.get(path);
               Files.write(filePath, data);
               logger.info("Successfully obtained bytes from an S3 object");
           } catch (IOException ex) {
               throw new RuntimeException("Failed to write data to file", ex);
           }
       }).whenComplete((resp, ex) -> {
```

```
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
 ex);
            }
        });
    }
```

Use an S3TransferManagerpara baixar um objeto em um bucket do S3 para um arquivo local. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);
```

```
        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Leia etiquetas que pertencem a um objeto usando um S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Lists the tags associated with an Amazon S3 object.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket that contains the object
     * @param keyName the key (name) of the S3 object
     */
    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            GetObjectTaggingResponse tags =
 s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Obtenha um URL para um objeto usando um S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
            .region(region)
            .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be
 retrieved
     * @throws S3Exception if there is an error retrieving the URL for the
 specified object
     */
    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Obtenha um objeto usando o objeto de cliente S3Presigner usando um S3Client.

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();
```

```
        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    /**
     * Generates a pre-signed URL for an Amazon S3 object.
     *
     * @param presigner The {@link S3Presigner} instance to use for generating
the pre-signed URL.
     * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
     * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
     *
     * @throws S3Exception If there is an error interacting with the Amazon S3
service.
     * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
     */
    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
```

```
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                     OutputStream httpOutputStream =
 connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.printStackTrace();
        }
    }
}
```

Obtenha um objeto usando um ResponseTransformer objeto e o S3Client.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves
 them to a local file.
```

```
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3
 service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
     * @param path The local file path where the object's bytes will be saved.
     * @throws IOException If an I/O error occurs while writing the bytes to the
 local file.
     * @throws S3Exception If an error occurs while retrieving the object from
 the S3 bucket.
     */
    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Baixe o objeto.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
```

```
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Baixe o objeto com a condição de que sua ETag corresponda à fornecida.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
```

```
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
```

```
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Baixe o objeto com a condição de que sua ETag não corresponda à fornecida.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
```

```
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
```

```
  }
}
```

Baixe o objeto usando a condição de que ele tenha sido criado ou modificado em determinado período.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfModifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
```

```
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Baixe o objeto usando a condição de que ele não tenha sido criado ou modificado em determinado período.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfUnmodifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
```

```
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetObjectreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um objeto.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse comando recupera o item "sample.txt" do bucket “amzn-s3-demo-bucket” e o salva em um arquivo chamado "local-sample.txt" no local atual. O arquivo “local-sample.txt” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```

Exemplo 2: Esse comando recupera o diretório virtual “DIR” do bucket “amzn-s3-demo-bucket” e o salva em uma pasta chamada "" no local atual. Local-DIR A pasta "Local-DIR" não precisa existir antes que esse comando seja chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Exemplo 3: baixa todos os objetos com chaves terminadas em “.json” de buckets com “config” no respectivo nome em arquivos na pasta especificada. As chaves do objeto são usadas para definir o nome dos arquivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Para obter detalhes da API, consulte GetObjectem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse comando recupera o item "sample.txt" do bucket “amzn-s3-demo-bucket” e o salva em um arquivo chamado "local-sample.txt" no local atual. O arquivo “local-sample.txt” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```

Exemplo 2: Esse comando recupera o diretório virtual “DIR” do bucket “amzn-s3-demo-bucket” e o salva em uma pasta chamada "" no local atual. Local-DIR A pasta "Local-DIR" não precisa existir antes que esse comando seja chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Exemplo 3: baixa todos os objetos com chaves terminadas em “.json” de buckets com “config” no respectivo nome em arquivos na pasta especificada. As chaves do objeto são usadas para definir o nome dos arquivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

Exemplo 4: baixa um único objeto usando o download paralelo de várias partes para melhorar o desempenho em arquivos grandes.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt -UseMultipartDownload
```

Exemplo 5: baixa um objeto grande usando o modo RANGE com um tamanho de peça personalizado de 64 MB. O modo RANGE usa solicitações de intervalo de bytes HTTP e funciona para todos os objetos, independentemente de como eles foram carregados.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-
large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Exemplo 6: baixa todos os objetos com o prefixo de chave “data/” para uma pasta local usando o download de várias partes com downloads simultâneos de arquivos.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-
data -UseMultipartDownload -DownloadFilesConcurrently
```

Exemplo 7: baixa um único objeto usando o download de várias partes com 20 conexões simultâneas para obter a máxima taxa de transferência em redes de alta largura de banda.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-
large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Exemplo 8: baixa todos os objetos com o prefixo de chave “data/” para uma pasta local usando o download de várias partes, continuando com as falhas de arquivos individuais em vez de abortá-las.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-
data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Para obter detalhes da API, consulte GetObjectem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
            raise
        else:
            return body
```

Obtenha um objeto usando uma solicitação condicional.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
 'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
```

```
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of
 the object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Para obter detalhes da API, consulte a GetObjectReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um objeto.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
 downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Obtenha um objeto e relate seu estado de criptografia do lado do servidor.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}"
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
 {err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }
```

```
    Ok(byte_count)
}
```

- Para obter detalhes da API, consulte a GetObjectreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getobject(           " oo_result is returned for
testing purposes. "
                 iv_bucket = iv_bucket_name
                 iv_key = iv_object_key ).
       DATA(lv_object_data) = oo_result->get_body( ).
       MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetObjectreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func downloadFile(bucket: String, key: String, to: String) async
 throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
 read data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file."))
            throw error
        }
    }
```

```
import AWSS3
```

```
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
            }

            return data
        }
        catch {
            print("ERROR: ", dump(error, name: "Reading a file."))
            throw error
        }
  }
```

- Para obter detalhes da API, consulte GetObjecta referência da API AWS SDK for Swift.

Use **`GetObjectAcl`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetObjectAcl`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar listas de controle de acesso (ACLs)

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
            s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                      << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
```

```
                          << getGranteeTypeString(grantee.GetType()) <<
 std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
```

```
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                   "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
        default:
            return "Permission unknown";
    }
}
```

- Para obter detalhes da API, consulte GetObjectAcla Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando recupera a lista de controle de acesso de um objeto em um bucket chamado `amzn-s3-demo-bucket`:

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

Saída:

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetObjectAclem Referência de AWS CLI Comandos.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetObjectAclreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V5

Exemplo 1: o comando obtém os detalhes do proprietário do objeto do S3.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
 *).Owner
```

Saída:

```
DisplayName Id
----------- --
testusername
 9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Para obter detalhes da API, consulte GetObjectAclem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
```

```
                self.object.key,
                acl.owner["DisplayName"],
            )
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- Para obter detalhes da API, consulte a GetObjectAclReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getobjectacl(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name
         iv_key = iv_object_key ).
       MESSAGE 'Retrieved object ACL.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetObjectAclreferência da API AWS SDK for SAP ABAP.

Use **`GetObjectAttributes`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetObjectAttributes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Trabalhar com a integridade de objetos do Amazon S3

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of
 the object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
```

```
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
```

```
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
```

```
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object
 parts." <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte GetObjectAttributesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para recuperar metadados de um objeto sem exibir o próprio objeto

O exemplo de `get-object-attributes` a seguir recupera metadados do objeto `doc1.rtf`.

```
aws s3api get-object-attributes \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf \
    --object-attributes "StorageClass" "ETag" "ObjectSize"
```

Saída:

```
{
    "LastModified": "2022-03-15T19:37:31+00:00",
    "VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
    "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
    "StorageClass": "STANDARD",
    "ObjectSize": 405
}
```

Para obter mais informações, consulte a GetObjectAttributesReferência de API do Amazon S3.

- Para obter detalhes da API, consulte GetObjectAttributesem Referência de AWS CLI Comandos.

Use **`GetObjectLegalHold`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetObjectLegalHold`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Bloquear objetos do Amazon S3

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
```

```
            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
 {bucketName}: " +
                                $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- Para obter detalhes da API, consulte GetObjectLegalHolda Referência AWS SDK para .NET da API.

CLI

AWS CLI

Recupera o status de “Retenção jurídica” de um objeto

O exemplo de `get-object-legal-hold` a seguir recupera o status de “Retenção jurídica” do objeto especificado.

```
aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

Saída:

```
{
    "LegalHold": {
        "Status": "ON"
    }
}
```

- Para obter detalhes da API, consulte GetObjectLegalHoldem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
```

```
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}
```

- Para obter detalhes da API, consulte GetObjectLegalHolda Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
 objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
 GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
 getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
 bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
 ex.getMessage() + "'");
        }

        return null;
    }
```

- Para obter detalhes da API, consulte GetObjectLegalHolda Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
```

```
  GetObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
 bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
 ${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
 ${bucketName} from ${bucketName}.  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

```
// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte GetObjectLegalHolda Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a retenção legal de objetos.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
 bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
            bucket,
            e,
        )
```

- Para obter detalhes da API, consulte a GetObjectLegalHoldReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getobjectlegalhold(         " oo_result is returned
for testing purposes. "
         iv_bucket = iv_bucket_name
         iv_key = iv_object_key ).
       MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetObjectLegalHoldreferência da API AWS SDK for SAP ABAP.

Use **`GetObjectLockConfiguration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetObjectLockConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Bloquear objetos do Amazon S3

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Get the object lock configuration details for an S3 bucket.
   /// </summary>
   /// <param name="bucketName">The bucket to get details.</param>
   /// <returns>The bucket's object lock configuration details.</returns>
   public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
   {
       try
       {
           var request = new GetObjectLockConfigurationRequest()
           {
               BucketName = bucketName
           };

           var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
           Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                             $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                             $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

           return response.ObjectLockConfiguration;
       }
       catch (AmazonS3Exception ex)
       {
           Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
           return new ObjectLockConfiguration();
       }
```

```
    }
```

- Para obter detalhes da API, consulte GetObjectLockConfigurationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar uma configuração de bloqueio de objetos de um bucket

O exemplo de `get-object-lock-configuration` a seguir recupera a configuração de bloqueio de objetos do bucket especificado.

```
aws s3api get-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock
```

Saída:

```
{
    "ObjectLockConfiguration": {
        "ObjectLockEnabled": "Enabled",
        "Rule": {
            "DefaultRetention": {
                "Mode": "COMPLIANCE",
                "Days": 50
            }
        }
    }
}
```

- Para obter detalhes da API, consulte GetObjectLockConfigurationem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }
```

```
 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}
```

- Para obter detalhes da API, consulte GetObjectLockConfigurationa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();
```

```
        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
```

- Para obter detalhes da API, consulte GetObjectLockConfigurationa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { ObjectLockConfiguration } = await client.send(
      new GetObjectLockConfigurationCommand({
        Bucket: bucketName,
```

```
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
 bucket>",
      }),
    );
    console.log(
      `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting object lock configuration for ${bucketName}.
 The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object lock configuration for ${bucketName}.
  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
```

```
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte GetObjectLockConfigurationa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: se a configuração do Bloqueio de Objetos estiver habilitada para o bucket do S3 em questão, este comando retornará o valor “Habilitado”.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Saída:

```
Value
-----
Enabled
```

- Para obter detalhes da API, consulte GetObjectLockConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: se a configuração do Bloqueio de Objetos estiver habilitada para o bucket do S3 em questão, este comando retornará o valor “Habilitado”.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Saída:

```
Value
-----
Enabled
```

- Para obter detalhes da API, consulte GetObjectLockConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de bloqueio de objetos.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:
    """
    Check if object lock is enabled for a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket to check.

    Returns:
        True if object lock is enabled, False otherwise.
    """
    try:
        response = s3_client.get_object_lock_configuration(Bucket=bucket)
        return (
            "ObjectLockConfiguration" in response
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
 "Enabled"
```

```
        )
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
            return False
        else:
            raise
```

- Para obter detalhes da API, consulte a GetObjectLockConfigurationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getobjectlockconfiguration(         " oo_result is
returned for testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetObjectLockConfigurationreferência da API AWS SDK for SAP ABAP.

Use **`GetObjectRetention`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetObjectRetention`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Bloquear objetos do Amazon S3

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"\tUnable to fetch object lock retention:
 '{ex.Message}'");
            return new ObjectLockRetention();
        }
    }
```

- Para obter detalhes da API, consulte GetObjectRetentiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para recuperar a configuração de retenção de objetos de um objeto

O exemplo de `get-object-retention` a seguir recupera a configuração de retenção de objetos do objeto especificado.

```
aws s3api get-object-retention \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

Saída:

```
{
    "Retention": {
        "Mode": "GOVERNANCE",
        "RetainUntilDate": "2025-01-01T00:00:00.000Z"
    }
}
```

- Para obter detalhes da API, consulte GetObjectRetentionem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
```

```
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}
```

- Para obter detalhes da API, consulte GetObjectRetentiona Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Get the retention period for an S3 object.
```

```
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
 GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
 getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
 in "+ bucketName +": " + response.retention().mode() +" until "+
 response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }
```

- Para obter detalhes da API, consulte GetObjectRetentiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const { Retention } = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    console.log(
      `${key} in ${bucketName} will be retained until
 ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
        `The object "${key}" in the bucket "${bucketName}" does not have an
 ObjectLock configuration.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object retention settings for
 "${bucketName}".  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
```

```
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte GetObjectRetentiona Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando retorna o modo e a data até a qual o objeto ficará retido.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Para obter detalhes da API, consulte GetObjectRetentionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando retorna o modo e a data até a qual o objeto ficará retido.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Para obter detalhes da API, consulte GetObjectRetentionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetObjectTagging`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetObjectTagging`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de etiquetas

CLI

AWS CLI

Para recuperar as tags anexadas a um objeto

O exemplo `get-object-tagging` a seguir recupera os valores da chave especificada do objeto especificado.

```
aws s3api get-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf
```

Saída:

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        }
    ]
```

```
}
```

O exemplo `get-object-tagging` a seguir tenta recuperar os conjuntos de tags do objeto `doc2.rtf`, o qual não tem tags.

```
aws s3api get-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc2.rtf
```

Saída:

```
{
    "TagSet": []
}
```

O exemplo `get-object-tagging` a seguir recupera os conjuntos de tags do objeto `doc3.rtf`, o qual não tem tags.

```
aws s3api get-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc3.rtf
```

Saída:

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        },
        {
            "Value": "finance",
            "Key": "department"
        },
        {
            "Value": "payroll",
            "Key": "team"
        }
    ]
}
```

- Para obter detalhes da API, consulte GetObjectTaggingem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o exemplo retorna as tags associadas ao objeto presente no bucket do S3 em questão.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Key  Value
---  -----
test value
```

- Para obter detalhes da API, consulte GetObjectTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o exemplo retorna as tags associadas ao objeto presente no bucket do S3 em questão.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Key  Value
---  -----
test value
```

- Para obter detalhes da API, consulte GetObjectTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetPublicAccessBlock`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetPublicAccessBlock`.

CLI

AWS CLI

Para definir ou modificar a configuração do Bloqueio de Acesso Público de um bucket

O exemplo `get-public-access-block` a seguir exibe a configuração do Bloqueio de Acesso Público do bucket especificado.

```
aws s3api get-public-access-block \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "PublicAccessBlockConfiguration": {
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "BlockPublicAcls": true,
        "RestrictPublicBuckets": true
    }
}
```

- Para obter detalhes da API, consulte GetPublicAccessBlockem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando retorna a configuração do Bloqueio de Acesso Público do bucket do S3 em questão.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetPublicAccessBlockem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando retorna a configuração do Bloqueio de Acesso Público do bucket do S3 em questão.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte GetPublicAccessBlockem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`HeadBucket`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `HeadBucket`.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#       $1 - The name of the bucket to check.
#
# Returns:
#       0 - If the bucket already exists.
#       1 - If the bucket doesn't exist.
###############################################################################
function bucket_exists() {
  local bucket_name
  bucket_name=$1

  # Check whether the bucket already exists.
  # We suppress all output - we're interested only in the return code.

  if aws s3api head-bucket \
    --bucket "$bucket_name" \
    >/dev/null 2>&1; then
```

```
    return 0 # 0 in Bash script means true.
  else
    return 1 # 1 in Bash script means false.
  fi
}
```

- Para obter detalhes da API, consulte HeadBucketem Referência de AWS CLI Comandos.

CLI

AWS CLI

O seguinte comando verifica o acesso ao bucket `amzn-s3-demo-bucket`:

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

Se o bucket existir e você tiver acesso a ele, nenhuma saída será retornada. Caso contrário, uma mensagem de erro será exibida. Por exemplo:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- Para obter detalhes da API, consulte HeadBucketem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
```

```
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
 (bool, error) {
 _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
```

```
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}
```

- Para obter detalhes da API, consulte HeadBucketa Referência AWS SDK para Go da API.

PowerShell

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna a saída com o código de status HTTP 200 OK para o bucket existente quando o usuário tem permissão para acessá-lo. BucketArn o parâmetro só é compatível com buckets de diretório do S3.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Saída:

```
AccessPointAlias   : False
BucketArn          :
BucketLocationName :
BucketLocationType :
BucketRegion       : us-east-2
ResponseMetadata   : Amazon.Runtime.ResponseMetadata
ContentLength      : 0
HttpStatusCode     : OK
```

Exemplo 2: Esse comando gera um erro com o código de status HTTP NotFound para um bucket inexistente.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Saída:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status
 Code NotFound. No further error information was returned by the service.
```

Exemplo 3: esse comando gera um erro com o código de status HTTP Proibido para um bucket existente em que o usuário não tem permissão para acessá-lo.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Saída:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status
 Code Forbidden. No further error information was returned by the service.
```

- Para obter detalhes da API, consulte HeadBucketem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
            exists = False
        return exists
```

- Para obter detalhes da API, consulte a HeadBucketReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->headbucket(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a HeadBucketreferência da API AWS SDK for SAP ABAP.

Use **`HeadObject`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `HeadObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Verifique se existe um objeto
- Introdução ao armazenamento de objetos

CLI

AWS CLI

O seguinte comando recupera metadados de um objeto no bucket `amzn-s3-demo-bucket`:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Saída:

```
{
    "AcceptRanges": "bytes",
    "ContentType": "text/html",
    "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
    "ContentLength": 77,
    "VersionId": "null",
    "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
    "Metadata": {}
}
```

- Para obter detalhes da API, consulte HeadObjectem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Determine o tipo de conteúdo de um objeto.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the content type of an object stored in an Amazon S3 bucket.
     *
     * @param s3 an instance of the {@link S3Client} class, which is used to
 interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the key (file name) of the object in the S3 bucket
     */
    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Obtenha o status de restauração de um objeto.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3         an instance of the {@link S3Client} class used to
 interact with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is
 stored
     * @param keyName    the name of the Amazon S3 object to be checked
```

```
     * @throws S3Exception if an error occurs while interacting with the Amazon
 S3 service
     */
    public static void checkStatus(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte HeadObjecta Referência AWS SDK for Java 2.x da API.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
 #{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte HeadObjecta Referência AWS SDK para Ruby da API.

Usar **`ListBucketAnalyticsConfigurations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListBucketAnalyticsConfigurations`.

CLI

AWS CLI

Para recuperar uma lista de configurações de análise de um bucket

O exemplo `list-bucket-analytics-configurations` a seguir recupera uma lista de configurações de análise do bucket especificado.

```
aws s3api list-bucket-analytics-configurations \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "AnalyticsConfigurationList": [
        {
            "StorageClassAnalysis": {},
            "Id": "1"
        }
    ],
    "IsTruncated": false
}
```

- Para obter detalhes da API, consulte ListBucketAnalyticsConfigurationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna as cem primeiras análises do bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte ListBucketAnalyticsConfigurationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna as cem primeiras análises do bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte ListBucketAnalyticsConfigurationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListBucketInventoryConfigurations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListBucketInventoryConfigurations`.

CLI

AWS CLI

Para recuperar uma lista de configurações de inventário de um bucket

O exemplo `list-bucket-inventory-configurations` a seguir recupera as configurações de inventário do bucket especificado.

```
aws s3api list-bucket-inventory-configurations \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "InventoryConfigurationList": [
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "ORC",
                    "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "1",
            "Schedule": {
                "Frequency": "Weekly"
            }
        },
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "CSV",
                    "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "AccountId": "123456789012"
                }
```

```
            },
            "IncludedObjectVersions": "Current",
            "Id": "2",
            "Schedule": {
                "Frequency": "Daily"
            }
        }
    ],
    "IsTruncated": false
}
```

- Para obter detalhes da API, consulte ListBucketInventoryConfigurationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna as cem primeiras configurações de inventário do bucket do S3 em questão.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte ListBucketInventoryConfigurationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna as cem primeiras configurações de inventário do bucket do S3 em questão.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes da API, consulte ListBucketInventoryConfigurationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListBuckets`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListBuckets`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
 client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
```

```
        /// <param name="bucketList">A List of S3 bucket objects.</param>
        public static void DisplayBucketList(List<S3Bucket> bucketList)
        {
            bucketList
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
 created on: {b.CreationDate}"));
        }

        public static async Task Main()
        {
            // The client uses the AWS Region of the default user.
            // If the Region where the buckets were created is different,
            // pass the Region to the client constructor. For example:
            // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
            _s3Client = new AmazonS3Client();
            var response = await GetBuckets(_s3Client);
            DisplayBucketList(response.Buckets);
        }
    }
}
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
```

```
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
 buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando usa o comando `list-buckets` para exibir os nomes de todos os buckets do Amazon S3 (em todas as regiões):

```
aws s3api list-buckets --query "Buckets[].Name"
```

A opção de consulta filtra a saída de `list-buckets` até sobrar apenas os nomes dos buckets.

Para obter mais informações sobre buckets, consulte Working with Amazon S3 Buckets no Guia do desenvolvedor do Amazon S3.

- Para obter detalhes da API, consulte ListBucketsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
```

```
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
 error) {
 var err error
 var output *s3.ListBucketsOutput
 var buckets []types.Bucket
 bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
 &s3.ListBucketsInput{})
 for bucketPaginator.HasMorePages() {
  output, err = bucketPaginator.NextPage(ctx)
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
   } else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
   }
   break
  } else {
   buckets = append(buckets, output.Buckets...)
  }
 }
 return buckets, err
}
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
 Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os buckets.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the Amazon S3 buckets in your account.
 */
export const main = async () => {
  const client = new S3Client({});
  /** @type {?import('@aws-sdk/client-s3').Owner} */
  let Owner = null;

  /** @type {import('@aws-sdk/client-s3').Bucket[]} */
  const Buckets = [];

  try {
    const paginator = paginateListBuckets({ client }, {});

    for await (const page of paginator) {
      if (!Owner) {
        Owner = page.Owner;
      }

      Buckets.push(...page.Buckets);
    }

    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
```

```
      }:`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (caught) {
    if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing buckets.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna todos os buckets do S3.

```
Get-S3Bucket
```

Exemplo 2: este comando retorna um bucket denominado “test-files”.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte ListBucketsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todos os buckets do S3.

```
Get-S3Bucket
```

Exemplo 2: este comando retorna um bucket denominado “test-files”.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes da API, consulte ListBucketsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
 Boto3
                            that contains collections and factory methods to
 create
                            other high-level S3 sub-resources.
        :return: The list of buckets.
```

```
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets
```

- Para obter detalhes da API, consulte a ListBucketsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
```

```
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte ListBucketsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;
```

```
    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListBucketsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->listbuckets(         " oo_result is returned for
testing purposes. "
        ).
        DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).
        MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list buckets.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListBucketsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    /// Return an array containing information about every available bucket.
    ///
    /// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
    ///   each bucket.
```

```
    public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
        return try await client.listBuckets(input: ListBucketsInput())
    }
```

- Para obter detalhes da API, consulte ListBucketsa referência da API AWS SDK for Swift.

Use **`ListMultipartUploads`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListMultipartUploads`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Excluir uploads multiparte incompletos

CLI

AWS CLI

O seguinte comando lista todos os uploads fracionados ativos do bucket `amzn-s3-demo-bucket`:

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

Saída:

```
{
    "Uploads": [
        {
            "Initiator": {
                "DisplayName": "username",
                "ID": "arn:aws:iam::0123456789012:user/username"
            },
            "Initiated": "2015-06-02T18:01:30.000Z",
            "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
            "StorageClass": "STANDARD",
            "Key": "multipart/01",
            "Owner": {
                "DisplayName": "aws-account-name",
```

```
                "ID":
 "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
            }
        }
    ],
    "CommonPrefixes": []
}
```

Os uploads fracionados em andamento incorrem em custos de armazenamento no Amazon S3. Conclua ou cancele um upload fracionado ativo para remover suas partes da conta.

- Para obter detalhes da API, consulte ListMultipartUploadsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon
 S3 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
 uploads for
     */
    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
```

```
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key()
 + "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListMultipartUploadsa Referência AWS SDK for Java 2.x da API.

Use **`ListObjectVersions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListObjectVersions`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Introdução ao armazenamento de objetos
- Trabalhar com objetos versionados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;
```

```
    /// <summary>
    /// This example lists the versions of the objects in a version enabled
    /// Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectVersions
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";

            // If the AWS Region where your bucket is defined is different from
            // the AWS Region where the Amazon S3 bucket is defined, pass the
 constant
            // for the AWS Region to the client constructor like this:
            //      var client = new AmazonS3Client(RegionEndpoint.USWest2);
            IAmazonS3 client = new AmazonS3Client();
            await GetObjectListWithAllVersionsAsync(client, bucketName);
        }

        /// <summary>
        /// This method lists all versions of the objects within an Amazon S3
        /// version enabled bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// ListVersionsAsync.</param>
        /// <param name="bucketName">The name of the version enabled Amazon S3
 bucket
        /// for which you want to list the versions of the contained objects.</
param>
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
 client, string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
```

```
                    MaxKeys = 2,
                };

                do
                {
                    ListVersionsResponse response = await
client.ListVersionsAsync(request);

                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine($"key: {entry.Key} size:
{entry.Size}");
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                }
                while (request != null);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}'");
            }
        }
    }
```

- Para obter detalhes da API, consulte ListObjectVersionsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando recupera as informações da versão de um objeto do bucket `amzn-s3-demo-bucket`:

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

Saída:

```
{
    "DeleteMarkers": [
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": true,
            "VersionId": "B2VsEK5saUNNHKcOAJj7hIE86RozToyq",
            "Key": "index.html",
            "LastModified": "2015-11-10T00:57:03.000Z"
        },
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
            "Key": "index.html",
            "LastModified": "2015-11-09T23:32:20.000Z"
        }
    ],
    "Versions": [
        {
            "LastModified": "2015-11-10T00:20:11.000Z",
            "VersionId": "Rb_l2T8UHDkFEwCgJjhlgPOZC0qJ.vpD",
            "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
```

```
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T23:26:41.000Z",
            "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAFf",
            "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T22:50:50.000Z",
            "VersionId": "null",
            "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 533823
        }
    ]
}
```

- Para obter detalhes da API, consulte ListObjectVersionsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
```

```
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

- Para obter detalhes da API, consulte ListObjectVersionsa Referência AWS SDK para Go da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
```

```
}
```

- Para obter detalhes da API, consulte a ListObjectVersionsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->listobjectversions(         " oo_result is returned
for testing purposes. "
         iv_bucket = iv_bucket_name
         iv_prefix = iv_prefix ).
       MESSAGE 'Retrieved object versions.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListObjectVersionsreferência da API AWS SDK for SAP ABAP.

Usar **`ListObjects`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListObjects`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar uma página da web que oferece uma lista de objetos do Amazon S3

CLI

AWS CLI

O seguinte exemplo usa o comando `list-objects` para exibir os nomes de todos os objetos presentes no bucket especificado:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size:
 Size}'
```

O exemplo usa o argumento `--query` para filtrar a saída de `list-objects` até encontrar o valor e o tamanho da chave de cada objeto

Para obter mais informações sobre objetos, consulte Working with Amazon S3 Buckets no Guia do desenvolvedor do Amazon S3.

- Para obter detalhes da API, consulte ListObjectsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando recupera as informações sobre todos os itens no bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Exemplo 2: este comando recupera as informações sobre o item “sample.txt” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 3: este comando recupera as informações sobre todos os itens com prefixo “sample” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Para obter detalhes da API, consulte ListObjectsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando recupera as informações sobre todos os itens no bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Exemplo 2: este comando recupera as informações sobre o item “sample.txt” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 3: este comando recupera as informações sobre todos os itens com prefixo “sample” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Para obter detalhes da API, consulte ListObjectsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListObjectsV2`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListObjectsV2`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Excluir todos os objetos em um bucket
- Introdução ao armazenamento de objetos

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
    {
        try
        {
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
                MaxKeys = 5,
            };

            if (printList)
            {
                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");
            }

            var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });
            var s3Objects = new List<S3Object>();
            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                if (response.S3Objects != null)
                {
                    s3Objects.AddRange(response.S3Objects);
                }
            }

            if (printList)
            {
                Console.WriteLine($"Number of Objects: {s3Objects.Count}");
                foreach (var entry in s3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
```

```
            }

            return s3Objects;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' getting list of objects.");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK para .NET da API.

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste objetos com um paginador.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// The following example lists objects in an Amazon Simple Storage
    /// Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectsPaginator
    {
        private const string BucketName = "amzn-s3-demo-bucket";

        public static async Task Main()
        {
            IAmazonS3 s3Client = new AmazonS3Client();
```

```
            Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");
            await ListingObjectsAsync(s3Client, BucketName);
        }

        /// <summary>
        /// This method uses a paginator to retrieve the list of objects in an
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the S3 bucket whose objects
        /// you want to list.</param>
        public static async Task ListingObjectsAsync(IAmazonS3 client, string
 bucketName)
        {
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
 ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
```

```
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);
```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                    outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como obter uma lista dos objetos em um bucket

O exemplo de `list-objects-v2` a seguir lista os objetos no bucket especificado.

```
aws s3api list-objects-v2 \
    --bucket amzn-s3-demo-bucket
```

Saída:

```
{
```

```
    "Contents": [
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"621503c373607d548b37cff8778d992c\"",
            "StorageClass": "STANDARD",
            "Key": "doc1.rtf",
            "Size": 391
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
            "StorageClass": "STANDARD",
            "Key": "doc2.rtf",
            "Size": 373
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
            "StorageClass": "STANDARD",
            "Key": "doc3.rtf",
            "Size": 399
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"d1852dd683f404306569471af106988e\"",
            "StorageClass": "STANDARD",
            "Key": "doc4.rtf",
            "Size": 6225
        }
    ]
}
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
 ([]types.Object, error) {
 var err error
 var output *s3.ListObjectsV2Output
 input := &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 }
 var objects []types.Object
 objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
 for objectPaginator.HasMorePages() {
  output, err = objectPaginator.NextPage(ctx)
  if err != nil {
```

```
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
     log.Printf("Bucket %s does not exist.\n", bucketName)
     err = noBucket
    }
    break
   } else {
    objects = append(objects, output.Contents...)
   }
  }
  return objects, err
}
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have
been listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();
```

```
        ListObjectsV2Publisher paginator =
 getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
 bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
```

Liste objetos usando paginação.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
```

```
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the S3 bucket to list the objects from
     */
    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() +
 " size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todos os objetos no bucket. Se houver mais de um objeto, IsTruncated NextContinuationToken ele será usado para iterar a lista completa.

```
import {
  S3Client,
  S3ServiceException,
  // This command supersedes the ListObjectsCommand and is the recommended way to
 list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );

    for await (const page of paginator) {
      objects.push(page.Contents.map((o) => o.Key));
    }
    objects.forEach((objectList, pageNum) => {
      console.log(
        `Page ${pageNum + 1}\n------\n${objectList.map((o) => `•
 ${o}`).join("\n")}\n`,
```

```
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}". The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}".
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }
```

```
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- Para obter detalhes da API, consulte ListObjectsV2 no AWS SDK para referência da API Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste objetos em um bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
```

```
            exit("Please fix error with listing objects before continuing.");
        }
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
 are listed.
        :return: The list of objects.
        """
        try:
```

```
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
 bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
 bucket.name)
            raise
        else:
            return objects
```

- Para obter detalhes da API, consulte Referência da API ListObjectsV2 no AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end
```

```
  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
 #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
 Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte ListObjectsV2 na referência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->listobjectsv2(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte ListObjectsV2 na referência da API ABAP do AWS SDK para SAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func listBucketFiles(bucket: String) async throws -> [String] {
        do {
            let input = ListObjectsV2Input(
                bucket: bucket
```

```
            )

            // Use "Paginated" to get all the objects.
            // This lets the SDK handle the 'continuationToken' in
"ListObjectsV2Output".
            let output = client.listObjectsV2Paginated(input: input)
            var names: [String] = []

            for try await page in output {
                guard let objList = page.contents else {
                    print("ERROR: listObjectsV2Paginated returned nil contents.")
                    continue
                }

                for obj in objList {
                    if let objName = obj.key {
                        names.append(objName)
                    }
                }
            }


            return names
        }
        catch {
            print("ERROR: ", dump(error, name: "Listing objects."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte ListObjectsV2 na referência da API AWS SDK for Swift.

Use **`PutBucketAccelerateConfiguration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketAccelerateConfiguration`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "amzn-s3-demo-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
```

```
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
        /// method will be enabling acceleration.</param>
        private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
        {
            try
            {
                var putRequest = new PutBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                    AccelerateConfiguration = new AccelerateConfiguration
                    {
                        Status = BucketAccelerateStatus.Enabled,
                    },
                };
                await client.PutBucketAccelerateConfigurationAsync(putRequest);

                var getRequest = new GetBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                };
                var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

                Console.WriteLine($"Acceleration state = '{response.Status}' ");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
            }
        }
    }
```

- Para obter detalhes da API, consulte PutBucketAccelerateConfiguration a Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para definir a configuração de aceleração de um bucket

O exemplo `put-bucket-accelerate-configuration` a seguir habilita a configuração de aceleração do bucket especificado.

```
aws s3api put-bucket-accelerate-configuration \
    --bucket amzn-s3-demo-bucket \
    --accelerate-configuration Status=Enabled
```

Este comando não produz saída.

- Para obter detalhes da API, consulte PutBucketAccelerateConfigurationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando habilita a aceleração de transferência do bucket do S3 em questão.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- Para obter detalhes da API, consulte PutBucketAccelerateConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando habilita a aceleração de transferência do bucket do S3 em questão.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- Para obter detalhes da API, consulte PutBucketAccelerateConfigurationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutBucketAcl`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketAcl`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar listas de controle de acesso (ACLs)

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Creates an Amazon S3 bucket with an ACL to control access to the
        /// bucket and the objects stored in it.
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// an Amazon S3 bucket, with an ACL applied to the bucket.
        /// </param>
        /// <param name="region">The AWS Region where the bucket will be
created.</param>
        /// <param name="newBucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value indicating success or failure.</returns>
        public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
        {
            try
            {
                // Create a new Amazon S3 bucket with Canned ACL.
                var putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = region,
                    CannedACL = S3CannedACL.LogDeliveryWrite,
```

```
                };

                PutBucketResponse putBucketResponse = await
 client.PutBucketAsync(putBucketRequest);

                return putBucketResponse.HttpStatusCode ==
 System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Amazon S3 error: {ex.Message}");
            }

            return false;
        }
```

- Para obter detalhes da API, consulte PutBucketAcla Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
 &ownerID,
                              const Aws::String &granteePermission,
                              const Aws::String &granteeType, const Aws::String
 &granteeID,
                              const Aws::String &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);
```

```
    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
            s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
```

```
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
*/

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- Para obter detalhes da API, consulte PutBucketAcla Referência AWS SDK para C++ da API.

CLI

AWS CLI

Este exemplo `full control` concede `read` permissão a dois AWS usuários (user1@example.com e user2@example.com) a todos:

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --grant-full-control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

Consulte http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html para obter detalhes sobre ACLs personalizadas (os comandos da ACL s3api, como`put-bucket-acl`, usam a mesma notação abreviada de argumentos).

- Para obter detalhes da API, consulte PutBucketAclem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;
```

```
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <id>\s

            Where:
              bucketName - The Amazon S3 bucket to grant permissions on.\s
              id - The ID of the owner of this bucket (you can get this value
 from the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }
```

```
    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full
 control of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)
                    .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();

            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                .owner(builder -> builder.id(id))
                .grants(grantList2)
                .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutBucketAcla Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloque a ACL do bucket.

```
import {
  PutBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant read access to a user using their canonical AWS account ID.
 *
 * Most Amazon S3 use cases don't require the use of access control lists (ACLs).
 * We recommend that you disable ACLs, except in unusual circumstances where
 * you need to control access for each object individually. Consider a policy
 instead.
 * For more information see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/bucket-policies.html.
 * @param {{ bucketName: string, granteeCanonicalUserId: string,
 ownerCanonicalUserId }}
 */
export const main = async ({
  bucketName,
  granteeCanonicalUserId,
  ownerCanonicalUserId,
}) => {
  const client = new S3Client({});
  const command = new PutBucketAclCommand({
    Bucket: bucketName,
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
```

```
            // The canonical ID of the user. This ID is an obfuscated form of
 your AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/finding-canonical-user-id.html.
            ID: granteeCanonicalUserId,
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "READ",
        },
      ],
      Owner: {
        ID: ownerCanonicalUserId,
      },
    },
  });

  try {
    await client.send(command);
    console.log(`Granted READ access to ${bucketName}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutBucketAcla Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
```

```
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- Para obter detalhes da API, consulte a PutBucketAclreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a PutBucketAclReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example: Grant log delivery access to a bucket
    " iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'
    lo_s3->putbucketacl(
      iv_bucket = iv_bucket_name
      iv_grantwrite = iv_grantwrite ).
    MESSAGE 'Bucket ACL updated.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketAclreferência da API AWS SDK for SAP ABAP.

Use **`PutBucketCors`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketCors`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Add CORS configuration to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to apply the CORS configuration to an Amazon S3 bucket.</param>
        /// <param name="configuration">The CORS configuration to apply.</param>
        private static async Task PutCORSConfigurationAsync(AmazonS3Client
 client, CORSConfiguration configuration)
        {
            PutCORSConfigurationRequest request = new
 PutCORSConfigurationRequest()
            {
                BucketName = BucketName,
                Configuration = configuration,
            };

            _ = await client.PutCORSConfigurationAsync(request);
        }
```

- Para obter detalhes da API, consulte PutBucketCorsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte exemplo habilita solicitações PUT, POST e DELETE de www.example.com e solicitações GET de qualquer domínio:

```
aws s3api put-bucket-cors --bucket amzn-s3-demo-bucket --cors-
configuration file://cors.json

cors.json:
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
```

```
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- Para obter detalhes da API, consulte PutBucketCorsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
import java.util.List;

import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
```

```
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an
 Amazon S3 bucket.
     *
     * @param s3            the {@link S3Client} instance used to interact with
 the Amazon S3 service
     * @param bucketName    the name of the Amazon S3 bucket for which the CORS
 configuration should be deleted
     * @param accountId     the expected AWS account ID of the bucket owner
```

```
    *
    * @throws S3Exception if an error occurs while deleting the CORS
configuration for the bucket
    */
   public static void deleteBucketCorsInformation(S3Client s3, String
bucketName, String accountId) {
       try {
           DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
               .bucket(bucketName)
               .expectedBucketOwner(accountId)
               .build();

           s3.deleteBucketCors(bucketCorsRequest);

       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   /**
    * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
specified S3 bucket.
    *
    * @param s3 the S3Client instance to use for the operation
    * @param bucketName the name of the S3 bucket to retrieve the CORS
configuration for
    * @param accountId the expected bucket owner's account ID
    *
    * @throws S3Exception if there is an error retrieving the CORS configuration
    */
   public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
       try {
           GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
               .bucket(bucketName)
               .expectedBucketOwner(accountId)
               .build();

           GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
           List<CORSRule> corsRules = corsResponse.corsRules();
```

```
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3
bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket to set the CORS rules for.
     * @param accountId The AWS account ID of the bucket owner.
     */
    public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                .allowedMethods(allowMethods)
                .allowedOrigins(allowOrigins)
                .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                .corsRules(corsRules)
                .build();

            PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
```

```
                .bucket(bucketName)
                .corsConfiguration(configuration)
                .expectedBucketOwner(accountId)
                .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutBucketCorsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione uma regra de CORS.

```
import {
  PutBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS
 configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
```

```
 const client = new S3Client({});

 try {
   await client.send(
     new PutBucketCorsCommand({
       Bucket: bucketName,
       CORSConfiguration: {
         CORSRules: [
           {
             // Allow all headers to be sent to this bucket.
             AllowedHeaders: ["*"],
             // Allow only GET and PUT methods to be sent to this bucket.
             AllowedMethods: ["GET", "PUT"],
             // Allow only requests from the specified origin.
             AllowedOrigins: ["https://www.example.com"],
             // Allow the entity tag (ETag) header to be returned in the
response. The ETag header
             // The entity tag represents a specific version of the object. The
ETag reflects
             // changes only to the contents of an object, not its metadata.
             ExposeHeaders: ["ETag"],
             // How long the requesting browser should cache the preflight
response. After
             // this time, the preflight request will have to be made again.
             MaxAgeSeconds: 3600,
           },
         ],
       },
     }),
   );
   console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
 } catch (caught) {
   if (
     caught instanceof S3ServiceException &&
     caught.name === "NoSuchBucket"
   ) {
     console.error(
       `Error from S3 while setting CORS rules for ${bucketName}. The bucket
doesn't exist.`,
     );
   } else if (caught instanceof S3ServiceException) {
     console.error(
       `Error from S3 while setting CORS rules for ${bucketName}.
${caught.name}: ${caught.message}`,
```

```
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutBucketCorsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
 are
        allowed from other domains.
```

```
        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
 self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a PutBucketCorsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end
```

```
  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obter detalhes da API, consulte PutBucketCorsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example: Allow PUT, POST, DELETE methods from http://www.example.com
        lo_s3->putbucketcors(
          iv_bucket = iv_bucket_name
          io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(
            it_corsrules = it_cors_rules ) ).
        MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketCorsreferência da API AWS SDK for SAP ABAP.

Use **`PutBucketEncryption`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketEncryption`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução ao armazenamento de objetos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set the bucket server side encryption to use AWSKMS with a customer-
managed key id.
    /// </summary>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <param name="kmsKeyId">The Id of the KMS Key.</param>
```

```
    /// <returns>True if successful.</returns>
    public static async Task<bool> SetBucketServerSideEncryption(string
bucketName, string kmsKeyId)
    {
        var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
        {
            ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
            {
                new ServerSideEncryptionRule
                {
                    ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
                    {
                        ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
                        ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
                    }
                }
            }
        };
        try
        {
            var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
            {
                BucketName = bucketName,
                ServerSideEncryptionConfiguration =
serverSideEncryptionByDefault,
            });

            return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "AccessDenied"
                ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
                : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
        }
        return false;
    }
```

- Para obter detalhes da API, consulte PutBucketEncryptiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para configurar a criptografia do lado do servidor para um bucket

O exemplo `put-bucket-encryption` a seguir define a criptografia AES256 como padrão para o bucket especificado.

```
aws s3api put-bucket-encryption \
    --bucket amzn-s3-demo-bucket \
    --server-side-encryption-configuration '{"Rules":
 [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

Este comando não produz saída.

- Para obter detalhes da API, consulte PutBucketEncryptionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse comando habilita a criptografia padrão AES256 do lado do servidor com Amazon S3 Managed Keys SSE-S3 () no bucket especificado.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Para obter detalhes da API, consulte PutBucketEncryptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse comando habilita a criptografia padrão AES256 do lado do servidor com Amazon S3 Managed Keys SSE-S3 () no bucket especificado.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Para obter detalhes da API, consulte PutBucketEncryptionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutBucketLifecycleConfiguration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketLifecycleConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Excluir uploads multiparte incompletos
- Gerenciar mensagens grandes usando o S3
- Trabalhar com objetos versionados

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Adds lifecycle configuration information to the S3 bucket named in
        /// the bucketName parameter.
        /// </summary>
        /// <param name="client">The S3 client used to call the
        /// PutLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">A string representing the S3 bucket to
        /// which configuration information will be added.</param>
        /// <param name="configuration">A LifecycleConfiguration object that
```

```
        /// will be applied to the S3 bucket.</param>
        public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
 string bucketName, LifecycleConfiguration configuration)
        {
            var request = new PutLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
                Configuration = configuration,
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }
```

- Para obter detalhes da API, consulte PutBucketLifecycleConfigurationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

O seguinte comando aplica a configuração de ciclo de vida ao bucket `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --
lifecycle-configuration  file://lifecycle.json
```

O arquivo `lifecycle.json` é um documento JSON na pasta atual que especifica duas regras:

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
```

```
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 2,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

A primeira regra move arquivos com o prefixo `rotated` para o Glacier na data especificada. A segunda regra move versões antigas de objetos para o Glacier quando elas não estão mais atualizadas. Para obter mais informações sobre formatos de data/hora aceitáveis, consulte Especificar valores de parâmetro Guia do usuário da AWS CLI.

- Para obter detalhes da API, consulte PutBucketLifecycleConfigurationem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
```

```
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <accountId>\s

            Where:
              bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
 to upload an object into.
              accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
```

```
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3           The Amazon S3 client to use for the operation.
     * @param bucketName   The name of the Amazon S3 bucket.
     * @param accountId    The expected owner of the Amazon S3 bucket.
     *
     * @throws S3Exception if there is an error setting the lifecycle
configuration.
     */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix
to the
            // S3 Glacier Flexible Retrieval storage class immediately.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();
```

```
            // Create a second rule.
            Transition transition2 = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            List<Transition> transitionList = new ArrayList<>();
            transitionList.add(transition2);

            LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            LifecycleRule rule2 = LifecycleRule.builder()
                .id("Archive and then delete rule")
                .filter(ruleFilter2)
                .transitions(transitionList)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the LifecycleRule objects to an ArrayList.
            ArrayList<LifecycleRule> ruleList = new ArrayList<>();
            ruleList.add(rule1);
            ruleList.add(rule2);

            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(ruleList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    /**
     * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a
new lifecycle rule to it.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket
     * @param accountId the expected owner of the Amazon S3 bucket
     */
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag predicate.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
```

```
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the new rule to the list.
            newList.add(rule1);
            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} to use for the operation
     * @param bucketName the name of the S3 bucket
     * @param accountId the expected account owner of the S3 bucket
     *
     * @throws S3Exception if an error occurs while deleting the lifecycle
configuration
     */
    public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
                .builder()
                .bucket(bucketName)
```

```
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutBucketLifecycleConfigurationa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_lifecycle_configuration(self, lifecycle_rules):
        """
```

```
        Apply a lifecycle configuration to the bucket. The lifecycle
configuration can
        be used to archive or delete the objects in the bucket according to
specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
```

- Para obter detalhes da API, consulte a PutBucketLifecycleConfigurationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example: Expire objects with prefix 'logs/' after 30 days
        lo_s3->putbucketlifecycleconf(
```

```
          iv_bucket = iv_bucket_name
          io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
            it_rules = it_lifecycle_rule ) ).
        MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketLifecycleConfigurationreferência da API AWS SDK for SAP ABAP.

Use **`PutBucketLogging`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketLogging`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
```

```
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
```

```
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
```

```
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
```

```
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
 local settings.
                .Build();
        }
    }
```

- Para obter detalhes da API, consulte PutBucketLoggingga Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: definir o registro em log da política de bucket

O exemplo de `put-bucket-logging` a seguir define a política de registro em log para amzn-s3-demo-bucket. Primeiro, conceda permissão à entidade principal do serviço de registro em log na política de bucket usando o comando `put-bucket-policy`.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://policy.json
```

Conteúdo de `policy.json`:

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {"Service": "logging.s3.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/Logs/*",
            "Condition": {
                "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
                "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
            }
        }
    ]
}
```

Para aplicar a política de registro em log, use `put-bucket-logging`.

```
aws s3api put-bucket-logging \
    --bucket amzn-s3-demo-bucket \
    --bucket-logging-status file://logging.json
```

Conteúdo de `logging.json`:

```
{
     "LoggingEnabled": {
         "TargetBucket": "amzn-s3-demo-bucket",
         "TargetPrefix": "Logs/"
     }
 }
```

O comando `put-bucket-policy` é necessário para conceder as permissões `s3:PutObject` à entidade principal do serviço de registro em log.

Consulte mais informações em Registrar em log as solicitações com registro em log de acesso ao servidor no Guia do usuário do Amazon S3.

Exemplo 2: definir uma política de bucket para registrar em log o acesso a um único usuário

O exemplo de `put-bucket-logging` a seguir define a política de registro em log para amzn-s3-demo-bucket. O AWS usuário bob@example.com terá controle total sobre os arquivos de log e ninguém mais terá acesso. Primeiro, conceda permissão ao S3 com `put-bucket-acl`.

```
aws s3api put-bucket-acl \
    --bucket amzn-s3-demo-bucket \
    --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
    --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

Depois, aplique a política de registro em log usando `put-bucket-logging`.

```
aws s3api put-bucket-logging \
    --bucket amzn-s3-demo-bucket \
    --bucket-logging-status file://logging.json
```

Conteúdo de `logging.json`:

```
{
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-bucket",
        "TargetPrefix": "amzn-s3-demo-bucket-logs/",
        "TargetGrants": [
            {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "bob@example.com"
                },
                "Permission": "FULL_CONTROL"
            }
        ]
    }
}
```

O comando `put-bucket-acl` é necessário para conceder as permissões necessárias ao sistema de entrega de log do S3 (permissões de gravação e read-acp).

Consulte mais informações em Habilitar o log de acesso ao servidor do Amazon S3 no Guia do desenvolvedor do Amazon S3.

- Para obter detalhes da API, consulte PutBucketLoggingem Referência de AWS CLI Comandos.

Usar **`PutBucketNotification`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutBucketNotification`.

CLI

AWS CLI

O exemplo aplica uma configuração de notificação ao bucket `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-
configuration file://notification.json
```

O arquivo `notification.json` é um documento JSON na pasta atual que especifica um tópico do SNS e o tipo de evento a ser monitorado:

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

O tópico do SNS deve ter uma política do IAM anexada que permita que o Amazon S3 publique nele:

```
{
 "Version":"2012-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
     "Service": "s3.amazonaws.com"
   },
   "Action": [
    "SNS:Publish"
   ],
   "Resource": "arn:aws:sns:us-west-2:123456789012:amzn-s3-demo-bucket",
   "Condition": {
      "ArnLike": {
      "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
    }
   }
  }
```

```
 ]
}
```

- Para obter detalhes da API, consulte PutBucketNotificationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo configura a configuração do tópico do SNS para o evento do S3 ObjectRemovedDelete e ativa a notificação para o determinado bucket do s3

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
 $topic
```

Exemplo 2: Este exemplo habilita notificações ObjectCreatedAll para o determinado bucket, enviando-o para a função Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

Exemplo 3: este exemplo cria duas configurações diferentes do Lambda com base em diferentes sufixos de chave e configura ambas em um único comando.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- Para obter detalhes da API, consulte PutBucketNotificationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo configura a configuração do tópico do SNS para o evento do S3 ObjectRemovedDelete e ativa a notificação para o determinado bucket do s3

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
 $topic
```

Exemplo 2: Este exemplo habilita notificações ObjectCreatedAll para o determinado bucket, enviando-o para a função Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

Exemplo 3: este exemplo cria duas configurações diferentes do Lambda com base em diferentes sufixos de chave e configura ambas em um único comando.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
```

```
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- Para obter detalhes da API, consulte PutBucketNotificationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutBucketNotificationConfiguration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketNotificationConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Processar notificações de eventos do S3

- Envie notificações de eventos para EventBridge

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to enable notifications for an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class EnableNotifications
    {
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket1";
            const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
            const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

            IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
            await EnableNotificationAsync(client, bucketName, snsTopic,
 sqsQueue);
        }

        /// <summary>
        /// This method makes the call to the PutBucketNotificationAsync method.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to call
        /// the PutBucketNotificationAsync method.</param>
```

```
        /// <param name="bucketName">The name of the bucket for which
        /// notifications will be turned on.</param>
        /// <param name="snsTopic">The ARN for the Amazon Simple Notification
        /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
        /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
        /// (Amazon SQS) queue to which notifications will be pushed.</param>
        public static async Task EnableNotificationAsync(
            IAmazonS3 client,
            string bucketName,
            string snsTopic,
            string sqsQueue)
        {
            try
            {
                // The bucket for which we are setting up notifications.
                var request = new PutBucketNotificationRequest()
                {
                    BucketName = bucketName,
                };

                // Defines the topic to use when sending a notification.
                var topicConfig = new TopicConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic,
                };
                request.TopicConfigurations = new List<TopicConfiguration>
                {
                    topicConfig,
                };
                request.QueueConfigurations = new List<QueueConfiguration>
                {
                    new QueueConfiguration()
                    {
                        Events = new List<EventType>
{ EventType.ObjectCreatedPut },
                        Queue = sqsQueue,
                    },
                };

                // Now apply the notification settings to the bucket.
                PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
            }
```

```
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- Para obter detalhes da API, consulte PutBucketNotificationConfigurationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para habilitar as notificações especificadas em um bucket

O exemplo de `put-bucket-notification-configuration` a seguir aplica uma configuração de notificação a um bucket chamado `amzn-s3-demo-bucket`. O arquivo `notification.json` é um documento JSON na pasta atual que especifica um tópico do SNS e o tipo de evento a ser monitorado.

```
aws s3api put-bucket-notification-configuration \
    --bucket amzn-s3-demo-bucket \
    --notification-configuration file://notification.json
```

Conteúdo de `notification.json`:

```
{
    "TopicConfigurations": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
            "Events": [
                "s3:ObjectCreated:*"
            ]
        }
    ]
}
```

O tópico do SNS deve ter uma política do IAM anexada que permita que o Amazon S3 publique nele.

```
{
    "Version":"2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
                }
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte PutBucketNotificationConfigurationem Referência de AWS CLI Comandos.

Use **`PutBucketPolicy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketPolicy`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
            Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
            s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}


//! Build a policy JSON string.
/*!
```

```
  \param userArn: Aws user Amazon Resource Name (ARN).
      For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
  \param bucketName: Name of a bucket.
  \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
            "{\n"
            "   \"Version\":\"2012-10-17\",\n"
            "   \"Statement\":[\n"
            "       {\n"
            "           \"Sid\": \"1\",\n"
            "           \"Effect\": \"Allow\",\n"
            "           \"Principal\": {\n"
            "               \"AWS\": \""
            + userArn +
            "\"\n""           },\n"
            "           \"Action\": [ \"s3:getObject\" ],\n"
            "           \"Resource\": [ \"arn:aws:s3:::"
            + bucketName +
            "/*\" ]\n"
            "       }\n"
            "   ]\n"
            "}";
}
```

- Para obter detalhes da API, consulte PutBucketPolicya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Este exemplo permite que todos os usuários recuperem qualquer objeto no amzn-s3-demo-bucket, exceto aqueles no. MySecretFolder Ele também `put` concede `delete` permissão ao usuário raiz da AWS conta`1234-5678-9012`:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file:// policy.json

policy.json:
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      },
      {
         "Effect": "Deny",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/MySecretFolder/*"
      },
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
         },
         "Action": [
            "s3:DeleteObject",
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }

  ]
}
```

- Para obter detalhes da API, consulte PutBucketPolicyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <polFile>

            Where:
```

```
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3         the {@link S3Client} object used to interact with the
Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
     */
    public static void setPolicy(S3Client s3, String bucketName, String
policyText) {
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)
                .build();
```

```
            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    /**
     * Retrieves the bucket policy from a specified file.
     *
     * @param policyFile the path to the file containing the bucket policy
     * @return the content of the bucket policy file as a string
     */
    public static String getBucketPolicyFromFile(String policyFile) {
        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
 StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- Para obter detalhes da API, consulte PutBucketPolicya Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione a política.

```
import {
  PutBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
  const client = new S3Client({});
  const command = new PutBucketPolicyCommand({
    // This is a resource-based policy. For more information on resource-based
 policies,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/
access_policies.html#policies_resource-based.
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
```

```
            AWS: iamRoleArn,
          },
          Action: "s3:GetObject",
          Resource: `arn:aws:s3:::${bucketName}/*`,
        },
      ],
    }),
    // Apply the preceding policy to this bucket.
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log(
      `GetObject access to the bucket "${bucketName}" was granted to the provided
 IAM role.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "MalformedPolicy"
    ) {
      console.error(
        `Error from S3 while setting the bucket policy for the bucket
 "${bucketName}". The policy was malformed.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting the bucket policy for the bucket
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutBucketPolicya Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
 self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
 self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a PutBucketPolicyReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obter detalhes da API, consulte PutBucketPolicya Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example policy JSON string
        " iv_policy = '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::123456789012:user/
user"},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::bucketname/*"]}]}'
        lo_s3->putbucketpolicy(
          iv_bucket = iv_bucket_name
          iv_policy = iv_policy ).
        MESSAGE 'Bucket policy set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketPolicyreferência da API AWS SDK for SAP ABAP.

Use **`PutBucketReplication`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketReplication`.

CLI

AWS CLI

Como configurar a replicação para um bucket do S3

O exemplo `put-bucket-replication` a seguir aplica uma configuração de replicação ao bucket do S3 especificado.

```
aws s3api put-bucket-replication \
```

```
    --bucket amzn-s3-demo-bucket1 \
    --replication-configuration file://replication.json
```

Conteúdo de `replication.json`:

```
{
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role",
    "Rules": [
        {
            "Status": "Enabled",
            "Priority": 1,
            "DeleteMarkerReplication": { "Status": "Disabled" },
            "Filter" : { "Prefix": ""},
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket2"
            }
        }
    ]
}
```

O bucket de destino deve ter o versionamento habilitado. A função especificada deve ter permissão para gravar no bucket de destino e ter uma relação de confiança que permita que o Amazon S3 assuma a função.

Exemplo de política de permissão de função:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetReplicationConfiguration",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ReplicateTags"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket2/*"
        }
    ]
}
```

Exemplo de política de relação de confiança:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Este comando não produz saída.

Para obter mais informações, consulte This is the topic title no Guia do usuário do Amazon Simple Storage Service.

- Para obter detalhes da API, consulte PutBucketReplicationem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Sets the replication configuration for an Amazon S3 bucket.
    *
    * @param s3Client             the S3Client instance to use for the operation
    * @param sourceBucketName     the name of the source bucket
    * @param destBucketName       the name of the destination bucket
    * @param destinationBucketARN the Amazon Resource Name (ARN) of the
destination bucket
    * @param roleARN              the ARN of the IAM role to use for the
replication configuration
    */
   public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
       try {
           Destination destination = Destination.builder()
               .bucket(destinationBucketARN)
               .storageClass(StorageClass.STANDARD)
               .build();

           // Define a prefix filter for replication.
           ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
               .prefix("documents/")
               .build();

           // Define delete marker replication setting.
           DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
               .status(DeleteMarkerReplicationStatus.DISABLED)
               .build();

           // Create the replication rule.
           ReplicationRule replicationRule = ReplicationRule.builder()
```

```
                .priority(1)
                .filter(ruleFilter)
                .status(ReplicationRuleStatus.ENABLED)
                .deleteMarkerReplication(deleteMarkerReplication)
                .destination(destination)
                .build();

            List<ReplicationRule> replicationRuleList = new ArrayList<>();
            replicationRuleList.add(replicationRule);

            // Define the replication configuration with IAM role.
            ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
                .role(roleARN)
                .rules(replicationRuleList)
                .build();

            // Apply the replication configuration to the source bucket.
            PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
                .bucket(sourceBucketName)
                .replicationConfiguration(configuration)
                .build();

            s3Client.putBucketReplication(replicationRequest);
            System.out.println("Replication configuration set successfully.");

        } catch (IllegalArgumentException e) {
            System.err.println("Configuration error: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
            System.err.println("Status Code: " + e.statusCode());
            System.err.println("Error Code: " + e.awsErrorDetails().errorCode());


        } catch (SdkException e) {
            System.err.println("SDK Exception: " + e.getMessage());
        }
    }
```

- Para obter detalhes da API, consulte PutBucketReplicationa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo define uma configuração de replicação com uma única regra que permite a replicação para o bucket 'amzn-s3-demo-bucket' de qualquer novo objeto criado com o prefixo de nome de chave "" no bucket 'amzn-s3-demo-bucket'. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Exemplo 2: Este exemplo define uma configuração de replicação com várias regras que permitem a replicação para o bucket 'amzn-s3-demo-bucket' de qualquer novo objeto criado com o prefixo de nome de chave "" ou "“. TaxDocs OtherDocs Os prefixos de chave não devem se sobrepor.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
```

```
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Exemplo 3: Este exemplo atualiza a configuração de replicação no bucket especificado para desativar a regra que controla a replicação de objetos com o prefixo de nome de chave "TaxDocs" no bucket 'amzn-s3-demo-bucket'.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Para obter detalhes da API, consulte PutBucketReplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo define uma configuração de replicação com uma única regra que permite a replicação para o bucket 'amzn-s3-demo-bucket' de qualquer novo objeto criado com o prefixo de nome de chave "" no bucket 'amzn-s3-demo-bucket'. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Exemplo 2: Este exemplo define uma configuração de replicação com várias regras que permitem a replicação para o bucket 'amzn-s3-demo-bucket' de qualquer novo objeto criado com o prefixo de nome de chave "" ou "“. TaxDocs OtherDocs Os prefixos de chave não devem se sobrepor.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}
```

```
Write-S3BucketReplication @params
```

Exemplo 3: Este exemplo atualiza a configuração de replicação no bucket especificado para desativar a regra que controla a replicação de objetos com o prefixo de nome de chave "TaxDocs" no bucket 'amzn-s3-demo-bucket'.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Para obter detalhes da API, consulte PutBucketReplicationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutBucketRequestPayment`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutBucketRequestPayment`.

CLI

AWS CLI

Exemplo 1: para habilitar a configuração “requester pays” para um bucket

O exemplo `put-bucket-request-payment` a seguir habilita `requester pays` do bucket especificado.

```
aws s3api put-bucket-request-payment \
    --bucket amzn-s3-demo-bucket \
    --request-payment-configuration '{"Payer":"Requester"}'
```

Este comando não produz saída.

Exemplo 2: para desabilitar a configuração “requester pays” para um bucket

O exemplo `put-bucket-request-payment` a seguir desabilita `requester pays` do bucket especificado.

```
aws s3api put-bucket-request-payment \
    --bucket amzn-s3-demo-bucket \
    --request-payment-configuration '{"Payer":"BucketOwner"}'
```

Este comando não produz saída.

- Para obter detalhes da API, consulte PutBucketRequestPaymentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: atualiza a configuração de pagamento de solicitação do bucket denominado “amzn-s3-demo-bucket” para que a pessoa que está solicitando downloads pelo bucket seja cobrada pelo download. Por padrão, o proprietário do bucket paga pelos downloads. Para definir o pagamento da solicitação de volta ao padrão, use 'BucketOwner' para o parâmetro RequestPaymentConfiguration _Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Para obter detalhes da API, consulte PutBucketRequestPaymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: atualiza a configuração de pagamento de solicitação do bucket denominado “amzn-s3-demo-bucket” para que a pessoa que está solicitando downloads pelo bucket seja cobrada pelo download. Por padrão, o proprietário do bucket paga pelos downloads. Para definir o pagamento da solicitação de volta ao padrão, use 'BucketOwner' para o parâmetro RequestPaymentConfiguration _Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Para obter detalhes da API, consulte PutBucketRequestPaymentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutBucketTagging`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutBucketTagging`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução ao armazenamento de objetos

CLI

AWS CLI

O seguinte comando aplica a configuração de marcação a um bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://
tagging.json
```

O arquivo `tagging.json` é um documento JSON na pasta atual que especifica duas tags:

```
{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
   ]
}
```

Ou aplique uma configuração de marcação a `amzn-s3-demo-bucket` diretamente da linha de comando:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging
 'TagSet=[{Key=organization,Value=marketing}]'
```

- Para obter detalhes da API, consulte PutBucketTaggingem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando aplica duas tags a um bucket denominado **cloudtrail-test-2018** (uma tag com uma chave de Stage e um valor de Test e uma tag com uma chave de Environment e um valor de Alpha). Para verificar se as tags foram adicionadas ao bucket, execute **Get-S3BucketTagging -BucketName bucket_name**. Os resultados devem mostrar as tags que você aplicou ao bucket no primeiro comando. Observe que **Write-S3BucketTagging** substitui todo o conjunto de tags existente em um bucket. Para adicionar ou excluir tags individuais, execute os cmdlets da API de grupos de recursos e marcação: **Add-RGTResourceTag** e **Remove-RGTResourceTag**. Como alternativa, use o Editor de tags no console AWS de gerenciamento para gerenciar as tags de bucket do S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemplo 2: este comando envia um bucket denominado **cloudtrail-test-2018** para o cmdlet **Write-S3BucketTagging**. Ele aplica tags Stage:Production e Department:Finance ao bucket. Observe que **Write-S3BucketTagging** substitui todo o conjunto de tags existente em um bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
 -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
 Value="Finance" } )
```

- Para obter detalhes da API, consulte PutBucketTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando aplica duas tags a um bucket denominado **cloudtrail-test-2018** (uma tag com uma chave de Stage e um valor de Test e uma tag com uma chave

de Environment e um valor de Alpha). Para verificar se as tags foram adicionadas ao bucket, execute **`Get-S3BucketTagging -BucketName bucket_name`**. Os resultados devem mostrar as tags que você aplicou ao bucket no primeiro comando. Observe que **`Write-S3BucketTagging`** substitui todo o conjunto de tags existente em um bucket. Para adicionar ou excluir tags individuais, execute os cmdlets da API de grupos de recursos e marcação: **`Add-RGTResourceTag`** e **`Remove-RGTResourceTag`**. Como alternativa, use o Editor de tags no console AWS de gerenciamento para gerenciar as tags de bucket do S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemplo 2: este comando envia um bucket denominado **`cloudtrail-test-2018`** para o cmdlet **`Write-S3BucketTagging`**. Ele aplica tags Stage:Production e Department:Finance ao bucket. Observe que **`Write-S3BucketTagging`** substitui todo o conjunto de tags existente em um bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
 -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
 Value="Finance" } )
```

- Para obter detalhes da API, consulte PutBucketTaggingem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutBucketVersioning`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketVersioning`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução ao armazenamento de objetos

CLI

AWS CLI

O comando a seguir habilita o versionamento em um bucket denominado `amzn-s3-demo-bucket`.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled
```

O comando a seguir habilita o versionamento e usa um código mfa.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled --mfa "SERIAL 123456"
```

- Para obter detalhes da API, consulte PutBucketVersioningem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Enables bucket versioning for the specified S3 bucket.
    *
    * @param s3Client the S3 client to use for the operation
    * @param bucketName the name of the S3 bucket to enable versioning for
    */
   public static void enableBucketVersioning(S3Client s3Client, String
bucketName){
       VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
           .status(BucketVersioningStatus.ENABLED)
           .build();

       PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
           .bucket(bucketName)
           .versioningConfiguration(versioningConfiguration)
           .build();
```

```
        s3Client.putBucketVersioning(versioningRequest);
        System.out.println("Bucket versioning has been enabled for "+bucketName);
    }
```

- Para obter detalhes da API, consulte PutBucketVersioninga Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando habilita o versionamento do bucket do S3 em questão.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Para obter detalhes da API, consulte PutBucketVersioningem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita o versionamento do bucket do S3 em questão.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Para obter detalhes da API, consulte PutBucketVersioningem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example: Enable versioning on a bucket
        " iv_status = 'Enabled'
        lo_s3->putbucketversioning(
          iv_bucket = iv_bucket_name
          io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(
            iv_status = iv_status ) ).
        MESSAGE 'Bucket versioning enabled.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketVersioningreferência da API AWS SDK for SAP ABAP.

Use **`PutBucketWebsite`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutBucketWebsite`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
                // Put the website configuration.
                PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument,
                    },
                };
```

```
                PutBucketWebsiteResponse response = await
 client.PutBucketWebsiteAsync(putRequest);
```

- Para obter detalhes da API, consulte PutBucketWebsitea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
            client.PutBucketWebsite(request);
```

```
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte PutBucketWebsitea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Aplica uma configuração de site estático ao bucket `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-
configuration file://website.json
```

O arquivo `website.json` é um documento JSON na pasta atual que especifica páginas de índice e erro para o site:

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- Para obter detalhes da API, consulte PutBucketWebsiteem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName> [indexdoc]\s

            Where:
               bucketName   - The Amazon S3 bucket to set the website
 configuration on.\s
               indexdoc - The index document, ex. 'index.html'
                          If not specified, 'index.html' will be set.
            """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
     * @param indexDoc The name of the index document to use for the website
configuration.
     */
    public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                .build();

            PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .websiteConfiguration(websiteConfig)
                .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte PutBucketWebsitea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração do site.

```
import {
  PutBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/
WebsiteAccessPermissionsReqd.html.
 *
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
```

```
      },
      IndexDocument: {
        // A suffix that is appended to a request when the request is
        // for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
    await client.send(command);
    console.log(
      `The bucket "${bucketName}" has been configured as a static website.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static
 website. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static
 website. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutBucketWebsitea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando habilita a hospedagem de sites para o bucket em questão com o documento de índice como “index.html” e o documento de erro como “error.html”.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Para obter detalhes da API, consulte PutBucketWebsiteem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita a hospedagem de sites para o bucket em questão com o documento de índice como “index.html” e o documento de erro como “error.html”.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Para obter detalhes da API, consulte PutBucketWebsiteem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website
```

```
  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
 website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte PutBucketWebsitea Referência AWS SDK para Ruby da API.

Use **`PutObject`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao armazenamento de objetos
- Fazer solicitações condicionais
- Monitorar uploads e downloads
- Trabalhar com a integridade de objetos do Amazon S3

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
```

```
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK para .NET da API.

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de um objeto com criptografia do lado do servidor.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
```

```
    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket with server-side encryption enabled.
    /// </summary>
    public class ServerSideEncryption
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "samplefile.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await WritingAnObjectAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// Upload a sample object include a setting for encryption.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// to upload a file and apply server-side encryption.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// encrypted object will reside.</param>
        /// <param name="keyName">The name for the object that you want to
        /// create in the supplied bucket.</param>
        public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
                };
```

```
                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

                Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
            }
        }
    }
```

Coloque um objeto usando uma solicitação condicional.

```
    /// <summary>
    /// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
    /// </summary>
    /// <param name="objectKey">The key of the object to upload.</param>
    /// <param name="bucket">The source bucket of the object.</param>
    /// <param name="content">The content to upload as a string.</param>
    /// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
    public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
    {
        try
        {
```

```
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition
failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte PutObjectem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
 needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          fileName.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
```

```
        std::cout << "Added object '" << fileName << "' to bucket '"
                  << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: fazer upload de um objeto no Amazon S3

O exemplo do comando `put-object` a seguir faz upload de um objeto no Amazon S3:

```
aws s3api put-object \
    --bucket amzn-s3-demo-bucket \
    --key my-dir/MySampleImage.png \
    --body MySampleImage.png
```

Para obter mais informações sobre o upload de objetos, consulte Uploading Objects < http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html > no Amazon S3 Developer Guide.

Exemplo 2: fazer upload de um arquivo de vídeo no Amazon S3

O exemplo do comando `put-object` a seguir faz upload de um arquivo de vídeo.

```
aws s3api put-object \
    --bucket amzn-s3-demo-bucket \
    --key my-dir/big-video-file.mp4 \
    --body /media/videos/f-sharp-3-data-services.mp4
```

Para obter mais informações sobre o upload de objetos, consulte Uploading Objects < http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html > no Amazon S3 Developer Guide.

- Para obter detalhes da API, consulte PutObjectem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloque um objeto em um bucket usando a API de baixo nível.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadFile reads from a file and puts the data into an object in a bucket.
```

```
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
     "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
     "or the multipart upload API (5TB max).", bucketName)
   } else {
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
     fileName, bucketName, objectKey, err)
   }
  } else {
   err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
   }
  }
 }
 return err
}
```

Faça upload de um objeto em um bucket usando um gerenciador de transferência.

```
import (
 "bytes"
 "context"
 "errors"
```

```
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
```

```
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de um arquivo em um bucket usando um S3Client.

```
   /**
    * Uploads a local file to an AWS S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
    * @param key        the key (object name) to use for the uploaded file
    * @param objectPath the local file path of the file to be uploaded
    * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
    */
   public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();
```

```
        CompletableFuture<PutObjectResponse> response =
 getAsyncClient().putObject(objectRequest,
 AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }
```

Use an S3TransferManagerpara fazer upload de um arquivo em um bucket. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Faça upload de um objeto em um bucket e defina etiquetas usando um S3Client.

```
    /**
     * Puts tags on an Amazon S3 object.
     *
     * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
     * @param bucketName The name of the Amazon S3 bucket.
     * @param objectKey The key of the Amazon S3 object.
     * @param objectPath The file path of the object to be uploaded.
     */
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                .key("Tag 1")
                .value("This is tag 1")
                .build();

            Tag tag2 = Tag.builder()
                .key("Tag 2")
                .value("This is tag 2")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);

            Tagging allTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(allTags)
                .build();

            s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   /**
    * Updates the tags associated with an object in an Amazon S3 bucket.
    *
    * @param s3 an instance of the S3Client class, which is used to interact
with the Amazon S3 service
    * @param bucketName the name of the S3 bucket containing the object
    * @param objectKey the key (or name) of the object in the S3 bucket
    * @throws S3Exception if there is an error updating the object's tags
    */
   public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
       try {
           GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
           List<Tag> obTags = getTaggingRes.tagSet();
           for (Tag sinTag : obTags) {
               System.out.println("The tag key is: " + sinTag.key());
               System.out.println("The tag value is: " + sinTag.value());
           }

           // Replace the object's tags with two new tags.
           Tag tag3 = Tag.builder()
               .key("Tag 3")
               .value("This is tag 3")
               .build();

           Tag tag4 = Tag.builder()
               .key("Tag 4")
               .value("This is tag 4")
               .build();

           List<Tag> tags = new ArrayList<>();
           tags.add(tag3);
           tags.add(tag4);

           Tagging updatedTags = Tagging.builder()
               .tagSet(tags)
```

```
                .build();

            PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(updatedTags)
                .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the contents of a file as a byte array.
     *
     * @param filePath the path of the file to be read
     * @return a byte array containing the contents of the file, or null if an
error occurs
     */
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
```

```
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

Faça upload de um objeto em um bucket e defina metadados usando um S3Client.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
              <bucketName> <objectKey> <objectPath>\s
```

```
            Where:
              bucketName - The Amazon S3 bucket to upload an object into.
              objectKey - The object to upload (for example, book.pdf).
              objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
            """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    /**
     * Uploads an object to an Amazon S3 bucket with metadata.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
```

```
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Faça upload de um objeto em um bucket e defina um valor de retenção de objetos usando um S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
```

```
        final String usage = """

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
     *
     * @param s3     the S3Client object used to interact with the Amazon S3
service
     * @param key    the key (name) of the object in the S3 bucket
     * @param bucket the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
period
     */
    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);
```

```
            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                .mode("COMPLIANCE")
                .retainUntilDate(instant)
                .build();

            PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
                .bucket(bucket)
                .key(key)
                .bypassGovernanceRetention(true)
                .retention(lockRetention)
                .build();

            // To set Retention on an object, the Amazon S3 bucket must support
 object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
 placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Carregue o objeto.

```
import { readFile } from "node:fs/promises";

import {
  PutObjectCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "EntityTooLarge"
    ) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console
 (160GB max) \
or the multipart upload API (5TB max).`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
```

```
};
```

Carregue o objeto com a condição de que sua ETag corresponda à fornecida.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
```

```
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- Para obter detalhes da API, consulte a PutObjectreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Carregue um objeto em um bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK para PHP da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Exemplo 2: este comando carrega o arquivo único “sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”. Quando o parâmetro -Key não é fornecido, usa-se o nome do arquivo como chave do objeto do S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Exemplo 3: Esse comando carrega o arquivo único "local-sample.txt" para o Amazon S3, criando um objeto com a chave prefix/to "/sample.txt" no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Exemplo 4: Esse comando carrega todos os arquivos no subdiretório “Scripts” para o bucket “test-files” e aplica o prefixo de chave comum "" a cada objeto. SampleScripts Cada arquivo enviado terá uma chave de "SampleScripts/filename", onde o 'nome do arquivo' varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

Exemplo 5: Esse comando carrega todos os arquivos*.ps1 no diretório local “Scripts” para o bucket “test-files” e aplica o prefixo de chave comum "" a cada objeto. SampleScripts Cada arquivo enviado terá uma chave de "SampleScripts/filename.ps1", onde o 'nome do arquivo' varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

Exemplo 6: este comando cria um objeto do S3 contendo a string de conteúdo especificada com a chave “sample.txt”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

Exemplo 7: este comando carrega o arquivo especificado (o nome do arquivo é usado como chave) e aplica as tags especificadas ao novo objeto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Exemplo 8: este comando carrega recursivamente a pasta especificada e aplica as tags especificadas a todos os novos objetos.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
 -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Para obter detalhes da API, consulte PutObjectem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Exemplo 2: este comando carrega o arquivo único “sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”. Quando o parâmetro -Key não é fornecido, usa-se o nome do arquivo como chave do objeto do S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Exemplo 3: Esse comando carrega o arquivo único "local-sample.txt" para o Amazon S3, criando um objeto com a chave prefix/to "/sample.txt" no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Exemplo 4: Esse comando carrega todos os arquivos no subdiretório “Scripts” para o bucket “test-files” e aplica o prefixo de chave comum "" a cada objeto. SampleScripts Cada arquivo enviado terá uma chave de "SampleScripts/filename", onde o 'nome do arquivo' varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

Exemplo 5: Esse comando carrega todos os arquivos*.ps1 no diretório local “Scripts” para o bucket “test-files” e aplica o prefixo de chave comum "" a cada objeto. SampleScripts Cada arquivo enviado terá uma chave de "SampleScripts/filename.ps1", onde o 'nome do arquivo' varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

Exemplo 6: este comando cria um objeto do S3 contendo a string de conteúdo especificada com a chave “sample.txt”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

Exemplo 7: este comando carrega o arquivo especificado (o nome do arquivo é usado como chave) e aplica as tags especificadas ao novo objeto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Exemplo 8: este comando carrega recursivamente a pasta especificada e aplica as tags especificadas a todos os novos objetos.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
 -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Para obter detalhes da API, consulte PutObjectem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
```

```
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
When this
                     argument is a string, it is interpreted as a file name,
which is
                     opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
data)
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
```

```
                put_data.close()
```

Faça upload de um objeto usando uma solicitação condicional.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def put_object_conditional(self, object_key: str, source_bucket: str, data:
 bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
 overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
 {source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
```

```
            if error_code == "PreconditionFailed":
                print("\tConditional write failed: Precondition failed")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Para obter detalhes da API, consulte a PutObjectReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de um arquivo usando um carregador gerenciado (Object.upload_file).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Faça upload de um arquivo usando Object.put.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
```

```
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Faça upload de um arquivo usando Object.put e adicione criptografia do lado do servidor.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
```

```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
 aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
```

```
        .send()
        .await
        .map_err(S3ExampleError::from)
}
```

- Para obter detalhes da API, consulte a PutObjectreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Get contents of file from application server."
    DATA lv_body TYPE xstring.
    OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
    READ DATASET iv_file_name INTO lv_body.
    CLOSE DATASET iv_file_name.

    "Upload/put an object to an S3 bucket."
    TRY.
        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_file_name
            iv_body = lv_body ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3
import Smithy

    public func uploadFile(bucket: String, key: String, file: String) async
 throws {
        let fileUrl = URL(fileURLWithPath: file)
        do {
            let fileData = try Data(contentsOf: fileUrl)
            let dataStream = ByteStream.data(fileData)

            let input = PutObjectInput(
                body: dataStream,
                bucket: bucket,
                key: key
            )

            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

```
import AWSS3
import Smithy

    public func createFile(bucket: String, key: String, withData data: Data)
 async throws {
        let dataStream = ByteStream.data(data)
```

```
        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte PutObjecta referência da API AWS SDK for Swift.

Use **`PutObjectAcl`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutObjectAcl`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Gerenciar listas de controle de acesso (ACLs)

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
 &objectKey, const Aws::String &ownerID,
                              const Aws::String &granteePermission, const
 Aws::String &granteeType,
```

```
                                const Aws::String &granteeID, const Aws::String
&granteeEmailAddress,
                                const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
   Aws::S3::S3Client s3Client(clientConfig);

   Aws::S3::Model::Owner owner;
   owner.SetID(ownerID);

   Aws::S3::Model::Grantee grantee;
   grantee.SetType(setGranteeType(granteeType));

   if (!granteeEmailAddress.empty()) {
       grantee.SetEmailAddress(granteeEmailAddress);
   }

   if (!granteeID.empty()) {
       grantee.SetID(granteeID);
   }

   if (!granteeURI.empty()) {
       grantee.SetURI(granteeURI);
   }

   Aws::S3::Model::Grant grant;
   grant.SetGrantee(grantee);
   grant.SetPermission(setGranteePermission(granteePermission));

   Aws::Vector<Aws::S3::Model::Grant> grants;
   grants.push_back(grant);

   Aws::S3::Model::AccessControlPolicy acp;
   acp.SetOwner(owner);
   acp.SetGrants(grants);

   Aws::S3::Model::PutObjectAclRequest request;
   request.SetAccessControlPolicy(acp);
   request.SetBucket(bucketName);
   request.SetKey(objectKey);

   Aws::S3::Model::PutObjectAclOutcome outcome =
           s3Client.PutObjectAcl(request);

   if (!outcome.IsSuccess()) {
```

```
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
                  << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
```

```
    return Aws::S3::Model::Type::NOT_SET;
}
```

- Para obter detalhes da API, consulte PutObjectAcla Referência AWS SDK para C++ da API.

CLI

AWS CLI

O comando a seguir concede `read` permissão `full control` a dois AWS usuários (user1@example.com e user2@example.com) a todos:

```
aws s3api put-object-acl --bucket amzn-s3-demo-bucket --key file.txt --grant-full-control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

Consulte http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html para obter detalhes sobre ACLs personalizadas (os comandos da ACL s3api, como`put-object-acl`, usam a mesma notação abreviada de argumentos).

- Para obter detalhes da API, consulte PutObjectAclem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
```

```
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- Para obter detalhes da API, consulte a PutObjectAclReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example: Grant read access to an AWS user
    " iv_grantread = 'emailAddress=user@example.com'
    lo_s3->putobjectacl(
      iv_bucket = iv_bucket_name
      iv_key = iv_object_key
      iv_grantread = iv_grantread ).
    MESSAGE 'Object ACL updated.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectAclreferência da API AWS SDK for SAP ABAP.

Use **`PutObjectLegalHold`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutObjectLegalHold`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Bloquear objetos do Amazon S3

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
```

```
    }
```

- Para obter detalhes da API, consulte PutObjectLegalHolda Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como aplicar uma retenção legal a um objeto

O exemplo de `put-object-legal-hold` a seguir define uma retenção legal no objeto `doc1.rtf`.

```
aws s3api put-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf \
    --legal-hold Status=ON
```

Este comando não produz saída.

- Para obter detalhes da API, consulte PutObjectLegalHoldem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
```

```
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- Para obter detalhes da API, consulte PutObjectLegalHolda Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Set or modify a legal hold on an object in an S3 bucket.
   public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
       ObjectLockLegalHold legalHold ;
       if (legalHoldOn) {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.ON)
               .build();
       } else {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.OFF)
               .build();
       }

       PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
           .bucket(bucketName)
           .key(objectKey)
           .legalHold(legalHold)
           .build();

       getClient().putObjectLegalHold(legalHoldRequest) ;
       System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
   }
```

- Para obter detalhes da API, consulte PutObjectLegalHolda Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  PutObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" |
 "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'.",
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
```

```
    },
  });

  try {
    await client.send(command);
    console.log(
      `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in
 "${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
 "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    objectKey: {
      type: "string",
```

```
      required: true,
    },
    legalHoldStatus: {
      type: "string",
      default: "ON",
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte PutObjectLegalHolda Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a retenção legal de objetos.

```
def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
```

```
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()
    logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
    try:
        before_status = "OFF"
        after_status = "ON"
        s3_client.put_object_legal_hold(
            Bucket=bucket, Key=key, LegalHold={"Status": after_status}
        )
        logger.debug(
            "Legal hold set successfully on file [%s] in bucket [%s]", key,
bucket
        )
        _print_legal_hold_update(bucket, key, before_status, after_status)
    except Exception as e:
        logger.error(
            "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
bucket, e
        )
```

- Para obter detalhes da API, consulte a PutObjectLegalHoldReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example: Set legal hold status to ON
```

```
        " iv_status = 'ON'
        lo_s3->putobjectlegalhold(
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key
          io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(
            iv_status = iv_status ) ).
        MESSAGE 'Object legal hold status set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectLegalHoldreferência da API AWS SDK for SAP ABAP.

Use **`PutObjectLockConfiguration`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutObjectLockConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Bloquear objetos do Amazon S3

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de Bloqueio de Objetos de um bucket.

```
    /// <summary>
    /// Enable object lock on an existing bucket.
```

```
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

Defina o período de retenção padrão de um bucket.

```
    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };
```

```
            var response = await
 _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
 {bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte PutObjectLockConfigurationa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como definir a configuração de bloqueio de objetos em um bucket

O exemplo de `put-object-lock-configuration` a seguir define um bloqueio de objetos de 50 dias no bucket especificado.

```
aws s3api put-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
 { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

Este comando não produz saída.

- Para obter detalhes da API, consulte PutObjectLockConfigurationem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de Bloqueio de Objetos de um bucket.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
```

```
    MFADelete: types.MFADeleteDisabled,
    Status:    types.BucketVersioningStatusEnabled,
   },
  }
  _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   return err
  }

  input := &s3.PutObjectLockConfigurationInput{
   Bucket: aws.String(bucket),
   ObjectLockConfiguration: &types.ObjectLockConfiguration{
    ObjectLockEnabled: types.ObjectLockEnabledEnabled,
   },
  }
  _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  }

  return err
}
```

Defina o período de retenção padrão de um bucket.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
```

```
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }
```

```
 return err
}
```

- Para obter detalhes da API, consulte PutObjectLockConfigurationa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de Bloqueio de Objetos de um bucket.

```
    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
```

```
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }
```

Defina o período de retenção padrão de um bucket.

```
    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
```

```
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
 PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


 getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
 +".");
    }
```

- Para obter detalhes da API, consulte PutObjectLockConfigurationa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de Bloqueio de Objetos de um bucket.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
```

```
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
 bucket "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
 bucket "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
```

```
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Defina o período de retenção padrão de um bucket.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
export const main = async ({ bucketName, retentionDays }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: bucketName,
        // The Object Lock configuration that you want to apply to the specified
 bucket.
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
```

```
          Rule: {
            // The default Object Lock retention mode and period that you want to
apply
            // to new objects placed in the specified bucket. Bucket settings
require
            // both a mode and a period. The period can be either Days or Years
but
            // you must select one.
            DefaultRetention: {
              // In governance mode, users can't overwrite or delete an object
version
              // or alter its lock settings unless they have special permissions.
With
              // governance mode, you protect objects against being deleted by
most users,
              // but you can still grant some users permission to alter the
retention settings
              // or delete the objects if necessary.
              Mode: "GOVERNANCE",
              Days: Number.parseInt(retentionDays),
            },
          },
        },
      }),
    );
    console.log(
      `Set default retention mode to "GOVERNANCE" with a retention period of
${retentionDays} day(s).`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting the default object retention for a bucket.
The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting the default object retention for a bucket.
${caught.name}: ${caught.message}`,
      );
    } else {
```

```
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    retentionDays: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte PutObjectLockConfigurationa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de bloqueio de objetos.

```
        s3_client.put_object_lock_configuration(
            Bucket=bucket,
            ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":
{}},
        )
```

- Para obter detalhes da API, consulte a PutObjectLockConfigurationReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example: Enable object lock with default retention
        " iv_enabled = 'Enabled'
        lo_s3->putobjectlockconfiguration(
          iv_bucket = iv_bucket_name
          io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(
            iv_objectlockenabled = iv_enabled ) ).
        MESSAGE 'Object lock configuration set.' TYPE 'I'.
```

```
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectLockConfigurationreferência da API AWS SDK for SAP ABAP.

Use **`PutObjectRetention`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutObjectRetention`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Bloquear objetos do Amazon S3

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
```

```
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
 until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
 '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte PutObjectRetentiona Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como definir uma configuração de retenção de objetos para um objeto

O exemplo de `put-object-retention` a seguir define uma configuração de retenção de objetos para o objeto especificado até 2025-01-01.

```
aws s3api put-object-retention \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf \
    --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":
 "2025-01-01T00:00:00" }'
```

Este comando não produz saída.

- Para obter detalhes da API, consulte PutObjectRetentionem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectRetention sets the object retention configuration for an S3 object.
```

```
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- Para obter detalhes da API, consulte PutObjectRetentiona Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Set or modify a retention period on an object in an S3 bucket.
```

```
   public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
       // Calculate the instant one day from now.
       Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

       // Convert the Instant to a ZonedDateTime object with a specific time
zone.
       ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

       // Define a formatter for human-readable output.
       DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

       // Format the ZonedDateTime object to a human-readable date string.
       String humanReadableDate = formatter.format(zonedDateTime);

       // Print the formatted date string.
       System.out.println("Formatted Date: " + humanReadableDate);
       ObjectLockRetention retention = ObjectLockRetention.builder()
           .mode(ObjectLockRetentionMode.GOVERNANCE)
           .retainUntilDate(futureInstant)
           .build();

       PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
           .bucket(bucketName)
           .key(objectKey)
           .retention(retention)
           .build();

       getClient().putObjectRetention(retentionRequest);
       System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
   }
```

- Para obter detalhes da API, consulte PutObjectRetentiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  PutObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most
 users,
      // but you can still grant some users permission to alter the retention
 settings
      // or delete the objects if necessary.
      Mode: "GOVERNANCE",
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
  });

  try {
    await client.send(command);
    console.log("Object Retention settings updated.");
```

```
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period
 on an object. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period
 on an object. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
```

```
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte PutObjectRetentiona Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando habilita o modo de retenção de governança até a data “31st Dec 2019 00:00:00” para o objeto “testfile.txt” no bucket do S3 em questão.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Para obter detalhes da API, consulte PutObjectRetentionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita o modo de retenção de governança até a data “31st Dec 2019 00:00:00” para o objeto “testfile.txt” no bucket do S3 em questão.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Para obter detalhes da API, consulte PutObjectRetentionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina uma retenção de objetos.

```
            s3_client.put_object_retention(
                Bucket=bucket,
                Key=key,
                VersionId=version_id,
                Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
far_future_date},
                BypassGovernanceRetention=True,
            )
```

- Para obter detalhes da API, consulte a PutObjectRetentionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example: Set retention mode to GOVERNANCE for 30 days
        " iv_mode = 'GOVERNANCE'
        " iv_retain_date should be a timestamp in the future
        lo_s3->putobjectretention(
```

```
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key
          io_retention = NEW /aws1/cl_s3_objectlockret(
            iv_mode = iv_mode
            iv_retainuntildate = iv_retain_date )
          iv_bypassgovernanceretention = abap_true ).
        MESSAGE 'Object retention set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectRetentionreferência da API AWS SDK for SAP ABAP.

Use **`RestoreObject`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `RestoreObject`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
```

```
    public class RestoreArchivedObject
    {
        public static void Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string objectKey = "archived-object.txt";

            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            IAmazonS3 client = new AmazonS3Client(bucketRegion);
            RestoreObjectAsync(client, bucketName, objectKey).Wait();
        }

        /// <summary>
        /// This method restores an archived object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// RestoreObjectAsync.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object was located before it was archived.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object to restore.</param>
        public static async Task RestoreObjectAsync(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            try
            {
                var restoreRequest = new RestoreObjectRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Days = 2,
                };
                RestoreObjectResponse response = await
 client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Error: {amazonS3Exception.Message}");
```

```
            }
        }

        /// <summary>
        /// This method retrieves the status of the object's restoration.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetObjectMetadataAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
        /// S3 bucket which contains the archived object.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object you want to restore.</param>
        public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
        {
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
            };

            GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

            var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
            Console.WriteLine($"Restoration status: {restStatus}");
        }
    }
```

- Para obter detalhes da API, consulte RestoreObjecta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar uma solicitação de restauração para um objeto

O exemplo de `restore-object` a seguir restaura o objeto Amazon S3 Glacier especificado do bucket `my-glacier-bucket` por 10 dias.

```
aws s3api restore-object \
    --bucket my-glacier-bucket \
    --key doc1.rtf \
    --restore-request Days=10
```

Este comando não produz saída.

- Para obter detalhes da API, consulte RestoreObjectem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 *  Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
```

```
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
 Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
 obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *
     * @param s3                   an instance of the {@link S3Client} to be used
 for interacting with Amazon S3
     * @param bucketName           the name of the S3 bucket where the object is
 stored
     * @param keyName              the key (object name) of the S3 object to be
 restored
```

```
     * @param expectedBucketOwner  the AWS account ID of the expected bucket
owner
     */
    public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                .expectedBucketOwner(expectedBucketOwner)
                .bucket(bucketName)
                .key(keyName)
                .restoreRequest(restoreRequest)
                .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte RestoreObjecta Referência AWS SDK for Java 2.x da API.

Use **`SelectObjectContent`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SelectObjectContent`.

CLI

AWS CLI

Como filtrar o conteúdo de um objeto do Amazon S3 com base em uma instrução SQL

O exemplo de `select-object-content` a seguir filtra o objeto `my-data-file.csv` com a instrução SQL especificada e envia a saída para um arquivo.

```
aws s3api select-object-content \
    --bucket amzn-s3-demo-bucket \
    --key my-data-file.csv \
    --expression "select * from s3object limit 100" \
    --expression-type 'SQL' \
    --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
    --output-serialization '{"CSV": {}}' "output.csv"
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SelectObjectContentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo a seguir mostra uma consulta usando um objeto JSON. O exemplo completo também mostra o uso de um objeto CSV.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
```

```
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
 software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your
 bucket name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
```

```
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where
c.area < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
                .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
```

```
        logger.info("Visitor onRecords for fileType {} called {} times",
 fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
 eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
 eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
 buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
 information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
 SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }
```

```
    // The EventStreamInfo class is used to store information gathered while
processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- Para obter detalhes da API, consulte SelectObjectContenta Referência AWS SDK for Java 2.x da API.

Use **`UploadPart`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UploadPart`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Usar somas de verificação
- Trabalhar com a integridade de objetos do Amazon S3

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
 checksum algorithm,
                            ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
 &bucket,
```

```
                                                                const Aws::String &key,
                                                                const Aws::String
 &uploadID,
                                                                int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                                const Aws::String
 &calculatedHash,
                                                                const
 std::shared_ptr<Aws::IOStream> &body,
                                                                const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}
```

- Para obter detalhes da API, consulte UploadPart a Referência AWS SDK para C++ da API.

CLI

AWS CLI

O seguinte comando faz o upload da primeira parte de um upload fracionado iniciado com o comando `create-multipart-upload`:

```
aws s3api upload-part --bucket amzn-s3-demo-bucket --key
 'multipart/01' --part-number 1 --body part01 --upload-id
  "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

A opção `body` usa o nome ou o caminho de um arquivo local para upload (não use o prefixo file://). O tamanho mínimo de uma parte é de 5 MB. O ID de upload é retornado por `create-multipart-upload` e também pode ser recuperado com `list-multipart-uploads`. O bucket e a chave são especificados ao criar o upload fracionado.

Saída:

```
{
    "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

Salve o valor do ETag de cada parte para etapas posteriores. Eles são necessários para concluir o upload fracionado.

- Para obter detalhes da API, consulte UploadPartem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
```

```
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

- Para obter detalhes da API, consulte a UploadPartreferência da API AWS SDK for Rust.

Use **`UploadPartCopy`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UploadPartCopy`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Executar uma cópia multipart

CLI

AWS CLI

Como fazer upload de parte de um objeto copiando dados de um objeto existente como a fonte de dados

O exemplo `upload-part-copy` a seguir faz upload de uma parte copiando dados de um objeto existente como uma fonte de dados.

```
aws s3api upload-part-copy \
    --bucket amzn-s3-demo-bucket \
    --key "Map_Data_June.mp4" \
    --copy-source "amzn-s3-demo-bucket/copy_of_Map_Data_June.mp4" \
    --part-number 1 \
    --upload-
id "bq0tdE1CDpWQYRPLHuNG50xAT6pA5D.m_RiBy0ggOH6b13pVRY7QjvLlf75iFdJqp_2wztk5hvpUM2SesXgrz
```

Saída:

```
{
    "CopyPartResult": {
        "LastModified": "2019-12-13T23:16:03.000Z",
        "ETag": "\"711470fc377698c393d94aed6305e245\""
    }
}
```

- Para obter detalhes da API, consulte UploadPartCopyem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public CompletableFuture<String> performMultiCopy(String toBucket, String
 bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();
```

```
        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }
```

- Para obter detalhes da API, consulte UploadPartCopya Referência AWS SDK for Java 2.x da API.

Cenários para o Amazon S3 usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon S3 com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções

dentro do Amazon S3 ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Verificar se um bucket existe
- Verifique se existe um objeto
- Converta texto em fala e volte para texto usando um AWS SDK
- Crie uma URL pré-assinada para o Amazon S3 usando um AWS SDK
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Uma página da web que lista objetos do Amazon S3 usando um AWS SDK
- Criar uma aplicação de exploração do Amazon Textract
- Exclua todos os objetos em um determinado bucket do Amazon S3 usando um AWS SDK
- Exclua uploads incompletos de várias partes para o Amazon S3 usando um AWS SDK
- Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK
- Detecte entidades no texto extraído de uma imagem usando um AWS SDK
- Detecte rostos em uma imagem usando um AWS SDK
- Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
- Baixar “diretórios” do S3 de um bucket do Amazon Simple Storage Service (Amazon S3)
- Fazer download de todos os objetos de um bucket do Amazon Simple Storage Service (Amazon S3) em um diretório local
- Baixe um stream de tamanho desconhecido de um objeto do Amazon S3 usando um AWS SDK
- Obtenha um objeto Amazon S3 de um ponto de Multi-Region acesso usando um AWS SDK
- Obtenha um objeto de um bucket do Amazon S3 usando um AWS SDK, especificando um cabeçalho If-Modified-Since
- Comece a usar a criptografia para objetos do Amazon S3 usando um AWS SDK

- Comece a usar tags para objetos do Amazon S3 usando um AWS SDK
- Introdução aos clusters de processamento de big data
- Conceitos básicos do gerenciamento de configurações
- Introdução à extração de texto de documentos
- Conceitos básicos de arquivos de atributos de machine learning
- Introdução ao armazenamento de objetos
- Introdução à análise de consulta
- Trabalhe com os recursos de bloqueio de objetos do Amazon S3 usando um AWS SDK
- Faça solicitações condicionais do Amazon S3 usando um AWS SDK
- Gerencie listas de controle de acesso (ACLs) para buckets do Amazon S3 usando um AWS SDK
- Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um AWS SDK
- Gerencie objetos versionados do Amazon S3 em lotes com uma função Lambda usando um AWS SDK
- Analise os URIs do Amazon S3 usando um AWS SDK
- Execute uma cópia em várias partes de um objeto do Amazon S3 usando um AWS SDK
- Receba e processe notificações de eventos do Amazon S3 usando um AWS SDK
- Salve EXIF e outras informações da imagem usando um AWS SDK
- Envie notificações de eventos do S3 para a Amazon EventBridge usando um AWS SDK
- Rastreie o upload ou o download de um objeto do Amazon S3 usando um AWS SDK
- Como transformar dados para sua aplicação com o S3 Object Lambda
- Exemplos de abordagens para testes unitários e de integração com um AWS SDK
- Fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service (Amazon S3)
- Carregue ou baixe arquivos grandes de e para o Amazon S3 usando um AWS SDK
- Faça upload de um stream de tamanho desconhecido para um objeto do Amazon S3 usando um AWS SDK
- Use somas de verificação para trabalhar com um objeto do Amazon S3 usando um AWS SDK
- Trabalhe com os recursos de integridade de objetos do Amazon S3 usando um AWS SDK
- Trabalhe com objetos versionados do Amazon S3 usando um AWS SDK

Verificar se um bucket existe

O exemplo de código a seguir mostra como verificar se um bucket existe.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Você pode usar o `doesBucketExists` método a seguir como substituto do método SDK for Java V1 AmazonS3Client BucketExist #does V2 (String).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

public class DoesBucketExist {
    private static final Logger logger =
 LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the
 bucket name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName,
 s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
```

```
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
 global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't
 be used to create a new bucket.
     * <p>
     * Internally this method uses the <a
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
     *
     * @param bucketName   The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks
 for the bucket in the AWS Region
     *                     configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the
 value false if there is no bucket in
     *         Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket
 exists but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
 "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte GetBucketAcla Referência AWS SDK for Java 2.x da API.

Verifique se existe um objeto

O exemplo de código a seguir mostra como verificar se um objeto existe.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Você pode usar o `doesObjectExist` método a seguir como substituto do SDK for Java V1 AmazonS3Client ObjectExist #does (String, String).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;
import software.amazon.awssdk.utils.Validate;

public class DoesObjectExist {
    private static final Logger logger =
 LoggerFactory.getLogger(DoesObjectExist.class);

    public static void main(String[] args) {
        DoesObjectExist doesObjectExist = new DoesObjectExist();

        final S3Client s3SyncClient = S3Client.builder().build();
```

```
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with your
 bucket name.
        final String key = "my-key"; // Replace with your object key.

        boolean exists = doesObjectExist.doesObjectExist(bucketName, key,
 s3SyncClient);
        logger.info("Object exists: {}", exists);
    }

    /**
     * Checks if the specified object exists in the specified bucket.
     * <p>
     * Internally this method uses the
     * <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#headObject(java.util.function.Consumer)">S3Client.headObject</a>
     * operation to determine whether the object exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's
     * <a href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/
amazonaws/services/s3/AmazonS3Client.html#doesObjectExist-java.lang.String-
java.lang.String-">AmazonS3Client#doesObjectExist(String, String)</a>.
     * <p>
     * <b>Note:</b> This method returns {@code false} only when S3 responds with
 404 (NoSuchKey). If the caller
     * does not have {@code s3:ListBucket} permission on the bucket, S3 may
 return 403 instead of 404 for
     * non-existent objects, which will be thrown as an exception.
     *
     * @param bucketName   The name of the bucket containing the object.
     * @param key          The key of the object to check.
     * @param s3SyncClient An {@code S3Client} instance.
     * @return {@code true} if the object exists; {@code false} if it does not
 exist.
     */
    public boolean doesObjectExist(String bucketName, String key, S3Client
 s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            Validate.notEmpty(key, "The object key must not be null or an empty
 string.", "");
            s3SyncClient.headObject(HeadObjectRequest.builder()
                    .bucket(bucketName)
```

```
                    .key(key)
                    .build());
            return true;
        } catch (NoSuchKeyException e) {
            return false;
        }
    }
}
```

- Para obter detalhes da API, consulte HeadObjecta Referência AWS SDK for Java 2.x da API.

Converta texto em fala e volte para texto usando um AWS SDK

O exemplo de código a seguir mostra como:

- Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio.
- Fazer upload do arquivo de áudio para um bucket do Amazon S3.
- Usar o Amazon Transcribe para converter o arquivo de áudio em texto.
- Exibir o texto.

Rust

SDK para Rust

Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio, fazer upload do arquivo de áudio em um bucket do Amazon S3, usar o Amazon Transcribe para converter esse arquivo de áudio em texto e exibir o texto.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Crie uma URL pré-assinada para o Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como criar um URL pré-assinado para o Amazon S3 e carregar um objeto.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere um URL pré-assinado que possa executar uma ação do Amazon S3 por tempo limitado.

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

public class GenPresignedUrl
{
    public static void Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string objectKey = "sample.txt";

        // Specify how long the presigned URL lasts, in hours
        const double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
set to Version 4 as shown below.
```

```
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

            string urlString = GeneratePresignedURL(s3Client, bucketName,
 objectKey, timeoutDuration);
            Console.WriteLine($"The generated URL is: {urlString}.");
        }

        /// <summary>
        /// Generate a presigned URL that can be used to access the file named
        /// in the objectKey parameter for the amount of time specified in the
        /// duration parameter.
        /// </summary>
        /// <param name="client">An initialized S3 client object used to call
        /// the GetPresignedUrl method.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// object for which to create the presigned URL.</param>
        /// <param name="objectKey">The name of the object to access with the
        /// presigned URL.</param>
        /// <param name="duration">The length of time for which the presigned
        /// URL will be valid.</param>
        /// <returns>A string representing the generated presigned URL.</returns>
        public static string GeneratePresignedURL(IAmazonS3 client, string
 bucketName, string objectKey, double duration)
        {
            string urlString = string.Empty;
            try
            {
                var request = new GetPreSignedUrlRequest()
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration),
                };
                urlString = client.GetPreSignedURL(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:'{ex.Message}'");
            }
```

```
            return urlString;
        }
    }
```

Gere um URL pré-assinado e faça um upload usando esse URL.

```
    using System;
    using System.IO;
    using System.Net.Http;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket using a presigned URL. The code first
    /// creates a presigned URL and then uses it to upload an object to an
    /// Amazon S3 bucket using that URL.
    /// </summary>
    public class UploadUsingPresignedURL
    {
        private static HttpClient httpClient = new HttpClient();

        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "samplefile.txt";
            string filePath = $"source\\{keyName}";

            // Specify how long the signed URL will be valid in hours.
            double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
KMS, you must specify Signature Version 4.
```

```
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

            var url = GeneratePreSignedURL(client, bucketName, keyName,
 timeoutDuration);
            var success = await UploadObject(filePath, url);

            if (success)
            {
                Console.WriteLine("Upload succeeded.");
            }
            else
            {
                Console.WriteLine("Upload failed.");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket using the presigned URL
 passed in
        /// the url parameter.
        /// </summary>
        /// <param name="filePath">The path (including file name) to the local
        /// file you want to upload.</param>
        /// <param name="url">The presigned URL that will be used to upload the
        /// file to the Amazon S3 bucket.</param>
        /// <returns>A Boolean value indicating the success or failure of the
        /// operation, based on the HttpWebResponse.</returns>
        public static async Task<bool> UploadObject(string filePath, string url)
        {
            using var streamContent = new StreamContent(
                new FileStream(filePath, FileMode.Open, FileAccess.Read));

            var response = await httpClient.PutAsync(url, streamContent);
            return response.IsSuccessStatusCode;
        }

        /// <summary>
```

```
        /// Generates a presigned URL which will be used to upload an object to
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetPreSignedURL.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
 the
        /// presigned URL will point.</param>
        /// <param name="objectKey">The name of the file that will be uploaded.</
param>
        /// <param name="duration">How long (in hours) the presigned URL will
        /// be valid.</param>
        /// <returns>The generated URL.</returns>
        public static string GeneratePreSignedURL(
            IAmazonS3 client,
            string bucketName,
            string objectKey,
            double duration)
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Verb = HttpVerb.PUT,
                Expires = DateTime.UtcNow.AddHours(duration),
            };

            string url = client.GetPreSignedURL(request);
            return url;
        }
    }
```

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie e use URLs POST pré-assinados para uploads diretos do navegador.

```
/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
    public static string? _bucketName;
    public static string? _objectKey;

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host">The IHost instance.</param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
        _httpClientFactory =
 host.Services.GetRequiredService<IHttpClientFactory>();
        _uiMethods = new UiMethods();
    }

    /// <summary>
    /// Perform the actions defined for the Amazon S3 Presigned POST scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
```

```
        _isInteractive = !args.Contains("--non-interactive");

        // Set up dependency injection for Amazon S3
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddHttpClient()
            )
            .Build();

        SetUpServices(host);

        try
        {
            // Display overview
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(_isInteractive);

            // Step 1: Create bucket
            await CreateBucketAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 2: Create presigned URL
            _uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
            var response = await CreatePresignedPostAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 3: Display URL and fields
            _uiMethods.DisplayTitle("Step 3: Presigned POST URL details");
            DisplayPresignedPostFields(response);
            _uiMethods.PressEnter(_isInteractive);

            // Step 4: Upload file
            _uiMethods.DisplayTitle("Step 4: Upload test file using presigned
POST URL");
            await UploadFileAsync(response);
            _uiMethods.PressEnter(_isInteractive);

            // Step 5: Verify file exists
            await VerifyFileExistsAsync();
            _uiMethods.PressEnter(_isInteractive);
```

```
            // Step 6: Cleanup
            _uiMethods.DisplayTitle("Step 6: Clean up resources");
            await CleanupAsync();

            _uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
 successfully!");
            _uiMethods.PressEnter(_isInteractive);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error in scenario");
            Console.WriteLine($"Error: {ex.Message}");

            // Attempt cleanup if there was an error
            if (!string.IsNullOrEmpty(_bucketName))
            {
                _uiMethods.DisplayTitle("Cleaning up resources after error");
                await _s3Wrapper.DeleteBucketAsync(_bucketName);
                Console.WriteLine($"Cleaned up bucket: {_bucketName}");
            }
        }
    }

    /// <summary>
    /// Create an S3 bucket for the scenario.
    /// </summary>
    private static async Task CreateBucketAsync()
    {
        _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

        // Generate a default bucket name for the scenario
        var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

        // Prompt user for bucket name or use default in non-interactive mode
        _bucketName = _uiMethods.GetUserInput(
            $"Enter S3 bucket name (or press Enter for '{defaultBucketName}'): ",
            defaultBucketName,
            _isInteractive);

        // Basic validation to ensure bucket name is not empty
        if (string.IsNullOrWhiteSpace(_bucketName))
        {
            _bucketName = defaultBucketName;
```

```
        }

        Console.WriteLine($"Creating bucket: {_bucketName}");

        await _s3Wrapper.CreateBucketAsync(_bucketName);

        Console.WriteLine($"Successfully created bucket: {_bucketName}");
    }


    /// <summary>
    /// Create a presigned POST URL.
    /// </summary>
    private static async Task<CreatePresignedPostResponse>
 CreatePresignedPostAsync()
    {
        _objectKey = "example-upload.txt";
        var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for
 the demo

        Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
        Console.WriteLine($"Expiration: {expiration} UTC");

        var s3Client = _s3Wrapper.GetS3Client();

        var response = await _s3Wrapper.CreatePresignedPostAsync(
            s3Client, _bucketName!, _objectKey, expiration);

        Console.WriteLine("Successfully created presigned POST URL");
        return response;
    }

    /// <summary>
    /// Upload a file using the presigned POST URL.
    /// </summary>
    private static async Task UploadFileAsync(CreatePresignedPostResponse
 response)
    {

        // Create a temporary test file to upload
        string testFilePath = Path.GetTempFileName();
        string testContent = "This is a test file for the S3 presigned POST
 scenario.";
```

```
        await File.WriteAllTextAsync(testFilePath, testContent);
        Console.WriteLine($"Created test file at: {testFilePath}");

        // Upload the file using the presigned POST URL
        Console.WriteLine("\nUploading file using the presigned POST URL...");
        var uploadResult = await UploadFileWithPresignedPostAsync(response,
testFilePath);

        // Display the upload result
        if (uploadResult.Success)
        {
            Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
        }
        else
        {
            Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
            Console.WriteLine($"Error: {uploadResult.Response}");
            throw new Exception("File upload failed");
        }

        // Clean up the temporary file
        File.Delete(testFilePath);
        Console.WriteLine("Temporary file deleted");
    }

    /// <summary>
    /// Helper method to upload a file using a presigned POST URL.
    /// </summary>
    private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
        CreatePresignedPostResponse response,
        string filePath)
    {
        try
        {
            _logger.LogInformation("Uploading file {filePath} using presigned
POST URL", filePath);

            using var httpClient = _httpClientFactory.CreateClient();
            using var formContent = new MultipartFormDataContent();
```

```
            // Add all the fields from the presigned POST response
            foreach (var field in response.Fields)
            {
                formContent.Add(new StringContent(field.Value), field.Key);
            }

            // Add the file content
            var fileStream = File.OpenRead(filePath);
            var fileName = Path.GetFileName(filePath);
            var fileContent = new StreamContent(fileStream);
            fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
            formContent.Add(fileContent, "file", fileName);

            // Send the POST request
            var httpResponse = await httpClient.PostAsync(response.Url,
 formContent);
            var responseContent = await httpResponse.Content.ReadAsStringAsync();

            // Log and return the result
            _logger.LogInformation("Upload completed with status code
 {statusCode}", httpResponse.StatusCode);

            return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
 responseContent);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error uploading file");
            return (false, HttpStatusCode.InternalServerError, ex.Message);
        }
    }

    /// <summary>
    /// Verify that the uploaded file exists in the S3 bucket.
    /// </summary>
    private static async Task VerifyFileExistsAsync()
    {
        _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

        Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

        try
```

```
        {
            var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
_objectKey!);

            Console.WriteLine($"File verification successful! File exists in the
bucket.");
            Console.WriteLine($"File size: {metadata.ContentLength} bytes");
            Console.WriteLine($"File type: {metadata.Headers.ContentType}");
            Console.WriteLine($"Last modified: {metadata.LastModified}");
        }
        catch (AmazonS3Exception ex) when (ex.StatusCode ==
System.Net.HttpStatusCode.NotFound)
        {
            Console.WriteLine($"Error: File was not found in the bucket.");
            throw;
        }
    }

    private static void DisplayPresignedPostFields(CreatePresignedPostResponse
response)
    {
        Console.WriteLine($"Presigned POST URL: {response.Url}");
        Console.WriteLine("Form fields to include:");

        foreach (var field in response.Fields)
        {
            Console.WriteLine($"  {field.Key}: {field.Value}");
        }
    }

    /// <summary>
    /// Clean up resources created by the scenario.
    /// </summary>
    private static async Task CleanupAsync()
    {
        if (!string.IsNullOrEmpty(_bucketName))
        {
            Console.WriteLine($"Deleting bucket {_bucketName} and its
contents...");
            bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

            if (result)
            {
                Console.WriteLine("Bucket deleted successfully");
```

```
            }
            else
            {
                Console.WriteLine("Failed to delete bucket - it may have been
 already deleted");
            }
        }
    }
}
```

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere um URL pré-assinado para baixar um objeto.

```
//! Routine which demonstrates creating a pre-signed URL to download an object
 from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param expirationSeconds: Expiration in seconds for pre-signed URL.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
 &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_GET,
```

```
                                            expirationSeconds);
}
```

Baixe usando a libcurl.

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                 Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }
```

```
    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
 std::endl;
        return false;
    }

    return true;
}
```

Gere um URL pré-assinado para carregar um objeto.

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to
 an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
 &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
 Aws::S3::S3ClientConfiguration &clientConfig) {
```

```
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_PUT,
                                        expirationSeconds);
}
```

Carregue usando a libcurl.

```
static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to put an object in a bucket.
  \param data: Body of the putObject request.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                 const Aws::String &data) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);
```

```
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);
```

```
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                  << std::endl;
        return false;
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que envolvam ações de pré-assinatura do S3.

```
import (
 "context"
 "log"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
 actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
 S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
 client.
type Presigner struct {
 PresignClient *s3.PresignClient
}



// GetObject makes a presigned request that can be used to get an object from a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
 ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignGetObject(ctx,
 &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// PutObject makes a presigned request that can be used to put an object in a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
```

```
 ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignPutObject(ctx,
 &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// DeleteObject makes a presigned request that can be used to delete an object
 from a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
 objectKey string) (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignDeleteObject(ctx,
 &s3.DeleteObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
 %v\n", objectKey, err)
 }
 return request, err
}



func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName
 string, objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error)
 {
 request, err := presigner.PresignClient.PresignPostObject(ctx,
 &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(options *s3.PresignPostOptions) {
```

```
  options.Expires = time.Duration(lifetimeSecs) * time.Second
 })
 if err != nil {
  log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v
\n", bucketName, objectKey, err)
 }
 return request, nil
}
```

Execute um exemplo interativo que gera e usa URLs pré-assinados para fazer upload, fazer download e excluir um objeto do S3.

```
import (
 "bytes"
 "context"
 "io"
 "log"
 "mime/multipart"
 "net/http"
 "os"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)



// RunPresigningScenario is an interactive example that shows you how to get
 presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
 Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
 can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
```

```
// 2. Use the net/http package to use the presigned request to upload a local
 file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
 object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
 mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner, httpRequester IHttpRequester) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 presigning demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}
 presignClient := s3.NewPresignClient(s3Client)
 presigner := actions.Presigner{PresignClient: presignClient}

 bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
  "you own or one you want to create:", demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
```

```
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
 demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
 panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
 presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
 panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
 uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))
```

```
log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
 presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
 uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName,
uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
 panic(err)
}
```

```
 defer uploadFile.Close()
 multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
 presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
 if err != nil {
  panic(err)
 }
 log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
 multiPartResponse.StatusCode)

 log.Println("Let's presign a request to delete the object.")
 questioner.Ask("Press Enter when you're ready.")
 presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
 if err != nil {
  panic(err)
 }
 log.Printf("Got a presigned %v request to URL:\n\t%v\n",
 presignedDelRequest.Method,
  presignedDelRequest.URL)
 log.Println("Using net/http to send the request...")
 delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
 if err != nil {
  panic(err)
 }
 log.Printf("%v object %v with presigned URL returned %v.\n",
 presignedDelRequest.Method,
  uploadKey, delResponse.StatusCode)
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina um wrapper de solicitação HTTP usado pelo exemplo para fazer solicitações HTTP.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
 during
// unit testing.
type IHttpRequester interface {
 Get(url string) (resp *http.Response, err error)
 Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
```

```
 Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
 error)
 Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
 scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
 return http.Get(url)
}
func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
 *http.Response, err error) {
 postRequest, err := http.NewRequest("POST", url, body)
 if err != nil {
  return nil, err
 }
 postRequest.Header.Set("Content-Type", contentType)
 return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
 (resp *http.Response, err error) {
 putRequest, err := http.NewRequest("PUT", url, body)
 if err != nil {
  return nil, err
 }
 putRequest.ContentLength = contentLength
 return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
 {
 delRequest, err := http.NewRequest("DELETE", url, nil)
 if err != nil {
  return nil, err
 }
 return http.DefaultClient.Do(delRequest)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Veja a seguir três exemplos de como criar URLs pré-assinados e usar os URLs com bibliotecas de cliente HTTP:

- Uma solicitação HTTP GET que usa o URL com três bibliotecas de cliente HTTP
- Uma solicitação HTTP PUT com metadados nos cabeçalhos que usa o URL com três bibliotecas de cliente HTTP
- Uma solicitação HTTP PUT com parâmetros de consulta que usa o URL com uma biblioteca cliente HTTP

Gere um URL pré-assinado para um objeto e faça download dele (solicitação GET).

Importações.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
```

```
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Gere o URL.

```
    /* Create a pre-signed URL to download an object in a subsequent GET request.
 */
    public String createPresignedGetUrl(String bucketName, String keyName) {
        try (S3Presigner presigner = S3Presigner.create()) {

            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            GetObjectPresignRequest presignRequest =
 GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL will
 expire in 10 minutes.
                    .getObjectRequest(objectRequest)
                    .build();

            PresignedGetObjectRequest presignedRequest =
 presigner.presignGetObject(presignRequest);
            logger.info("Presigned URL: [{}]",
 presignedRequest.url().toString());
            logger.info("HTTP method: [{}]",
 presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Faça o download do objeto usando uma das três abordagens a seguir.

Use a classe `HttpURLConnection` do JDK (desde a v1.1) para fazer o download.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
    public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setRequestMethod("GET");
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                IoUtils.copy(content, byteArrayOutputStream);
            }
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Use a classe `HttpClient` do JDK (desde a v11) para fazer o download.

```
    /* Use the JDK HttpClient (since v11) class to do the download. */
    public byte[] useHttpClientToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpResponse<InputStream> response = httpClient.send(requestBuilder
                            .uri(presignedUrl.toURI())
                            .GET()
```

```
                            .build(),
                    HttpResponse.BodyHandlers.ofInputStream());

            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Use a classe AWS SDK for `SdkHttpClient` Java para fazer o download.

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToGet(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
```

```
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Gere um URL pré-assinado com metadados em cabeçalhos para um upload e faça upload de um arquivo (solicitação PUT).

Importações.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
```

```
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Gere o URL.

```
   /* Create a presigned URL to use in a subsequent PUT request */
   public String createPresignedUrl(String bucketName, String keyName,
 Map<String, String> metadata) {
       try (S3Presigner presigner = S3Presigner.create()) {

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .metadata(metadata)
                   .build();

           PutObjectPresignRequest presignRequest =
 PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL
 expires in 10 minutes.
                   .putObjectRequest(objectRequest)
                   .build();


           PresignedPutObjectRequest presignedRequest =
 presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
 presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

Faça upload de um objeto de arquivo usando uma das três abordagens a seguir.

Use a classe `HttpURLConnection` do JDK (desde a v1.1) para fazer o upload.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File
 fileToPut, Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
 presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Use a classe `HttpClient` do JDK (desde a v11) para fazer o upload.

```
   /* Use the JDK HttpClient (since v11) class to do the upload. */
   public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
       metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

       HttpClient httpClient = HttpClient.newHttpClient();
       try {
           final HttpResponse<Void> response = httpClient.send(requestBuilder
                           .uri(new URL(presignedUrlString).toURI())

.PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                           .build(),
                   HttpResponse.BodyHandlers.discarding());

           logger.info("HTTP response code is " + response.statusCode());

       } catch (URISyntaxException | InterruptedException | IOException e) {
           logger.error(e.getMessage(), e);
       }
   }
```

Use a `SdkHttpClient` classe AWS for Java V2 para fazer o upload.

```
   /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
   public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       try {
           URL presignedUrl = new URL(presignedUrlString);

           SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                   .method(SdkHttpMethod.PUT)
                   .uri(presignedUrl.toURI());
           // Add headers
```

```
            metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
 k, v));
            // Finish building the request.
            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
 FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Gere um URL pré-assinado com parâmetros de consulta para um upload e faça upload de um arquivo (solicitação PUT).

Importações.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
```

```
import
 software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Gere o URL.

```
   /**
    *  Creates a presigned URL to use in a subsequent HTTP PUT request. The code
 adds query parameters
    *  to the request instead of using headers. By using query parameters, you
 do not need to add the
    *  the parameters as headers when the PUT request is eventually sent.
    *
    * @param bucketName Bucket name where the object will be uploaded.
    * @param keyName Key name of the object that will be uploaded.
    * @param queryParams Query string parameters to be added to the presigned
 URL.
    * @return
    */
   public String createPresignedUrl(String bucketName, String keyName,
 Map<String, String> queryParams) {
       try (S3Presigner presigner = S3Presigner.create()) {
           // Create an override configuration to store the query parameters.
           AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder
 = AwsRequestOverrideConfiguration.builder();


 queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
```

```
.overrideConfiguration(overrideConfigurationBuilder.build()) // Add the override
configuration.
                    .build();

            PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL
expires in 10 minutes.
                    .putObjectRequest(objectRequest)
                    .build();


            PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
            String myURL = presignedRequest.url().toString();
            logger.info("Presigned URL to upload a file to: [{}]", myURL);
            logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Use a `SdkHttpClient` classe AWS for Java V2 para fazer o upload.

```
    /**
     * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT
request. Since the
     * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
     *
     * @param presignedUrlString The URL for the PUT request.
     * @param fileToPut File to uplaod
     */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut)
{
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);
```

```
            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());

            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um URL pré-assinado para carregar um objeto em um bucket.

```
import https from "node:https";

import { XMLParser } from "fast-xml-parser";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
```

```
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Make a PUT request to the provided URL.
 *
 * @param {string} url
 * @param {string} data
 */
const put = (url, data) => {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
```

```
        res.on("end", () => {
          const parser = new XMLParser();
          if (res.statusCode >= 200 && res.statusCode <= 299) {
            resolve(parser.parse(responseBody, true));
          } else {
            reject(parser.parse(responseBody, true));
          }
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
};

/**
 * Create two presigned urls for uploading an object to an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");
```

```
    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
 configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Crie um URL pré-assinado para baixar um objeto de um bucket.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
```

```
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      region,
      key,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
 configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma solicitação pré-assinada de `GetObject` e use o URL para fazer download de um objeto.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
 request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

Crie uma solicitação `GetObject` atribuída previamente com opções avançadas.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
 used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Crie uma solicitação pré-assinada de `PutObject` e use-a para fazer upload de um objeto.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
```

```
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
```

```
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
 S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
 the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();



namespace S3;
```

```
use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
```

```
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "\nPlease fix error with bucket deletion before continuing.
\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }
```

```
   public function putObject(string $bucketName, string $key, array $args = [])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
       try {
           $this->client->putObject($parameters);
           if ($this->verbose) {
               echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object uploading before continuing.";
           }
           throw $exception;
       }
   }



   public function getObject(string $bucketName, string $key, array $args = []):
Result
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
       try {
           $object = $this->client->getObject($parameters);
           if ($this->verbose) {
               echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object downloading before
continuing.";
           }
           throw $exception;
       }
```

```
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
```

```
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args
= [])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
```

```
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
 error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array
 $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
```

```
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
```

```
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }

}
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere um URL pré-assinado que possa realizar uma ação do S3 por tempo limitado. Use o pacote Requests para fazer uma solicitação com o URL.

```
import argparse
```

```
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
 ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
        )
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
```

```
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to
 perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
        if response.status_code == 200:
            with open(args.key.split("/")[-1], 'wb') as object_file:
                object_file.write(response.content)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "rb") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be
 the "
                f"name of a file that exists on your computer."
            )

    if response is not None:
        print(f"Status: {response.status_code}\nReason: {response.reason}")

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

Gere uma solicitação POST pré-assinada para carregar um arquivo.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
 AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                 required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
 '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
 why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
```

```
    http.send_request('PUT', presigned_url.request_uri, object_content,
 'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie solicitações de pré-assinatura para objetos GET do S3.

```
/// Generate a URL for a presigned GET request.
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;
```

```
    println!("Object URI: {}", presigned_request.uri());
    let valid_until = chrono::offset::Local::now() + expires_in;
    println!("Valid until: {valid_until}");

    Ok(())
}
```

Crie solicitações de pré-assinatura para objetos PUT do S3.

```
async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<String, S3ExampleError> {
    let expires_in: std::time::Duration =
 std::time::Duration::from_secs(expires_in);
    let expires_in: aws_sdk_s3::presigning::PresigningConfig =
        PresigningConfig::expires_in(expires_in).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to convert expiration to PresigningConfig: {err:?}"
            ))
        })?;
    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(expires_in)
        .await?;

    Ok(presigned_request.uri().into())
}
```

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie solicitações pré-assinadas para objetos GET do S3.

```
" iv_bucket_name is the bucket name
" iv_key is the object name like "myfile.txt"

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
    DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call.
" to the service.  It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
  iv_bucket = iv_bucket_name
  iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file.  The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).
```

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Uma página da web que lista objetos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como listar objetos do Amazon S3 em uma página da web.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O código a seguir é o componente relevante do React que faz chamadas para o AWS SDK. Uma versão executável do aplicativo contendo esse componente pode ser encontrada no link anterior GitHub .

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
```

```
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
 a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
 traffic from
      // this example site. Here's an example configuration that allows all
 origins. Don't
      // do this in production.
      //[
      //  {
      //    "AllowedHeaders": ["*"],
      //    "AllowedMethods": ["GET"],
      //    "AllowedOrigins": ["*"],
      //    "ExposeHeaders": [],
      //  },
      //]
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
      }),
    });
    const command = new ListObjectsCommand({ Bucket: "bucket-name" });
    client.send(command).then(({ Contents }) => setObjects(Contents || []));
  }, []);

  return (
    <div className="App">
      {objects.map((o) => (
        <div key={o.ETag}>{o.Key}</div>
      ))}
    </div>
  );
}
```

```
export default App;
```

- Para obter detalhes da API, consulte ListObjectsa Referência AWS SDK para JavaScript da API.

Criar uma aplicação de exploração do Amazon Textract

Os exemplos de código a seguir mostram como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Exclua todos os objetos em um determinado bucket do Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como excluir todos os objetos de um bucket do Amazon S3.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua todos os objetos de um bucket específico do Amazon S3.

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 *
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  try {
    console.log(`Deleting all objects in bucket: ${bucketName}`);

    const paginator = paginateListObjectsV2(
      { client },
      {
        Bucket: bucketName,
      },
    );

    const objectKeys = [];
    for await (const { Contents } of paginator) {
      objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
    }

    const deleteCommand = new DeleteObjectsCommand({
      Bucket: bucketName,
      Delete: { Objects: objectKeys },
```

```
    });

    await client.send(deleteCommand);

    console.log(`All objects deleted from bucket: ${bucketName}`);
  } catch (caught) {
    if (caught instanceof Error) {
      console.error(
        `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    bucketName: {
      type: "string",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteObjects
  - ListObjectsV2

Exclua uploads incompletos de várias partes para o Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como excluir ou interromper uploads multiparte incompletos do Amazon S3.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para interromper uploads multiparte que estão em andamento ou incompletos por qualquer motivo, você pode obter uma lista de uploads para excluí-los, conforme mostrado no exemplo a seguir.

```
   /**
    * Aborts all incomplete multipart uploads from the specified S3 bucket.
    *
    * @param bucketName the name of the S3 bucket
    */
   public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
           .bucket(bucketName)
           .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(upload.key())
               .expectedBucketOwner(accountId)
               .uploadId(upload.uploadId())
               .build();

           AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
           if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
```

```
                logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
            }
        }
    }
```

Para excluir uploads multiparte incompletos que foram iniciados antes ou depois de uma data, você pode excluir seletivamente os uploads multiparte com base em um momento específico, conforme mostrado no exemplo a seguir.

```
    /**
     * Aborts incomplete multipart uploads older than the specified point in
time.
     *
     * @param bucketName  the name of the S3 bucket
     * @param pointInTime the cutoff time; uploads initiated before this are
aborted
     */
    static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();
```

```
                AbortMultipartUploadResponse abortMultipartUploadResponse =
 s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
 (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
 aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
```

Se você tiver acesso ao ID de upload depois de iniciar um upload multiparte, poderá excluir o upload em andamento usando esse ID.

```
    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key)
 {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
 uploadId, bucketName);
        }
    }
```

Para excluir uploads multiparte incompletos após determinado número de dias de maneira consistente, defina uma configuração de ciclo de vida para o bucket. O exemplo a seguir mostra como criar uma regra para excluir uploads multiparte existentes há mais de 7 dias.

```
    /**
```

```
    * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
    *
    * @param bucketName the name of the S3 bucket
    */
   static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
       Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
           .abortIncompleteMultipartUpload(b -> b.
               daysAfterInitiation(7))
           .status("Enabled")
           .filter(SdkBuilder::build) // Filter element is required.
           .build());

       PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
           .bucket(bucketName)
           .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

       if (response.sdkHttpResponse().isSuccessful()) {
           logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
       } else {
           logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
       }
   }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AbortMultipartUpload
  - ListMultipartUploads
  - PutBucketLifecycleConfiguration

Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detecte entidades no texto extraído de uma imagem usando um AWS SDK

O exemplo de código a seguir mostra como usar o Amazon Comprehend para detectar entidades em texto extraído pelo Amazon Textract de uma imagem armazenada no Amazon S3.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) em um notebook Jupyter para detectar entidades no texto extraído de uma imagem. Este exemplo usa o Amazon Textract para extrair texto de uma imagem armazenada no Amazon Simple Storage Service (Amazon S3) e no Amazon Comprehend para detectar entidades no texto extraído.

Este exemplo é um caderno Jupyter e deve ser executado em um ambiente que possa hospedar blocos de anotações. Para obter instruções sobre como executar o exemplo usando o Amazon SageMaker AI, consulte as instruções em TextractAndComprehendNotebook.ipynb.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend

- Amazon S3
- Amazon Textract

Detecte rostos em uma imagem usando um AWS SDK

O exemplo de código a seguir mostra como:

- Salvar uma imagem em um bucket do Amazon S3.
- Usar o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.).
- Exibir esses detalhes.

Rust

SDK para Rust

Salve a imagem em um bucket do Amazon S3 com um prefixo uploads, use o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.), e exiba esses detalhes.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3

Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um

bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Rekognition AWS SDK para JavaScript com o para criar um aplicativo que usa o Amazon Rekognition para identificar objetos por categoria em imagens

localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Aprenda como:

- Criar um usuário não autenticado usando o Amazon Cognito.
- Analisar imagens em busca de objetos usando o Amazon Rekognition.
- Verificar um endereço de e-mail para o Amazon SES.
- Enviar uma notificação por e-mail usando o Amazon SES.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um aplicativo web que permite fazer o seguinte:

- Carregar fotos em um bucket do Amazon Simple Storage Service (Amazon S3).
- Usar o Amazon Rekognition para analisar e rotular as fotos.
- Usar o Amazon Simple Email Service (Amazon SES) para enviar relatórios de análise da imagem por e-mail.

Este exemplo contém dois componentes principais: uma página da web escrita com React e um serviço REST escrito em Python JavaScript que é construído com. Flask-RESTful

Você pode usar a página da Web do React para:

- Exibir uma lista de imagens que estão armazenadas no bucket do S3.
- Carregar imagens do computador para o bucket do S3.
- Exibir imagens e rótulos que identificam os itens detectados na imagem.
- Obter um relatório de todas as imagens no bucket do S3 e enviar um relatório por e-mail.

A página da Web chama o serviço REST. O serviço envia solicitações à AWS para realizar as seguintes ações:

- Obter e filtrar a lista de imagens no bucket do S3.
- Carregar fotos no bucket do S3.
- Usar o Amazon Rekognition para analisar fotos individuais e obter uma lista dos rótulos que identifiquem os itens detectados nas fotos.
- Analisar todas as fotos no bucket do S3 e usar o Amazon SES para enviar um relatório por e-mail.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Baixar “diretórios” do S3 de um bucket do Amazon Simple Storage Service (Amazon S3)

O exemplo de código a seguir mostra como baixar e filtrar o conteúdo de “diretórios” de bucket do Amazon S3.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo mostra como usar o S3TransferManagerin the AWS SDK for Java 2.x para baixar 'diretórios' de um bucket do Amazon S3. Também demonstra como usar DownloadFiltersna solicitação.

```
    /**
     * For standard buckets, S3 provides the illusion of a directory structure
through the use of keys. When you upload
     * an object to an S3 bucket, you specify a key, which is essentially the
"path" to the object. The key can contain
     * forward slashes ("/") to make it appear as if the object is stored in a
directory structure, but this is just a
     * logical representation, not an actual directory.
     * <p><pre>
     * In this example, our S3 bucket contains the following objects:
     *
     * folder1/file1.txt
```

```
     * folder1/file2.txt
     * folder1/file3.txt
     * folder2/file1.txt
     * folder2/file2.txt
     * folder2/file3.txt
     * folder3/file1.txt
     * folder3/file2.txt
     * folder3/file3.txt
     *
     * When method `downloadS3Directories` is invoked with
     * `destinationPathURI` set to `/test`, the downloaded
     * directory looks like:
     *
     * |- test
     *    |- folder1
     *       |- file1.txt
     *       |- file2.txt
     *       |- file3.txt
     *    |- folder3
     *       |- file1.txt
     *       |- file2.txt
     *       |- file3.txt
     * </pre>
     *
     * @param transferManager    An S3TransferManager instance.
     * @param destinationPathURI local directory to hold the downloaded S3
 'directories' and files.
     * @param bucketName         The S3 bucket that contains the 'directories' to
 download.
     * @return The number of objects (files, in this case) that were downloaded.
     */
    public Integer downloadS3Directories(S3TransferManager transferManager,
                                         URI destinationPathURI, String
 bucketName) {

        // Define the filters for which 'directories' we want to download.
        DownloadFilter folder1Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder1/");
        DownloadFilter folder3Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder3/");
        DownloadFilter folderFilter = s3Object ->
 folder1Filter.or(folder3Filter).test(s3Object);
```

```
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .filter(folderFilter)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
 Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
 Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2`
 was not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object failed to transfer  [{}]",
 fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
    }
```

Fazer download de todos os objetos de um bucket do Amazon Simple Storage Service (Amazon S3) em um diretório local

O exemplo de código a seguir mostra como fazer download de todos os objetos de um bucket do Amazon Simple Storage Service (Amazon S3) em um diretório local.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use an S3TransferManagerpara baixar todos os objetos do S3 no mesmo bucket do S3. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();
```

```
        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- Para obter detalhes da API, consulte DownloadDirectorya Referência AWS SDK for Java 2.x da API.

Baixe um stream de tamanho desconhecido de um objeto do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como baixar um fluxo de tamanho desconhecido de um objeto do Amazon S3.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams


    /// Download a file from the specified bucket.
    ///
    /// - Parameters:
    ///   - bucket: The Amazon S3 bucket name to get the file from.
    ///   - key: The name (or path) of the file to download from the bucket.
    ///   - destPath: The pathname on the local filesystem at which to store
```

```
   ///     the downloaded file.
   func downloadFile(bucket: String, key: String, destPath: String?) async
throws {
       let fileURL: URL

       // If no destination path was provided, use the key as the name to use
       // for the file in the downloads folder.

       if destPath == nil {
           do {
               try fileURL = FileManager.default.url(
                   for: .downloadsDirectory,
                   in: .userDomainMask,
                   appropriateFor: URL(string: key),
                   create: true
               ).appendingPathComponent(key)
           } catch {
               throw TransferError.directoryError
           }
       } else {
           fileURL = URL(fileURLWithPath: destPath!)
       }

       let config = try await S3Client.S3ClientConfiguration(region: region)
       let s3Client = S3Client(config: config)

       // Create a `FileHandle` referencing the local destination. Then
       // create a `ByteStream` from that.

       FileManager.default.createFile(atPath: fileURL.path, contents: nil,
attributes: nil)
       let fileHandle = try FileHandle(forWritingTo: fileURL)

       // Download the file using `GetObject`.

       let getInput = GetObjectInput(
           bucket: bucket,
           key: key
       )

       do {
           let getOutput = try await s3Client.getObject(input: getInput)

           guard let body = getOutput.body else {
```

```
                throw TransferError.downloadError("Error: No data returned for
download")
            }

            // If the body is returned as a `Data` object, write that to the
            // file. If it's a stream, read the stream chunk by chunk,
            // appending each chunk to the destination file.

            switch body {
            case .data:
                guard let data = try await body.readData() else {
                    throw TransferError.downloadError("Download error")
                }

                // Write the `Data` to the file.

                do {
                    try data.write(to: fileURL)
                } catch {
                    throw TransferError.writeError
                }
                break

            case .stream(let stream as ReadableStream):
                while (true) {
                    let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)
                    guard let chunk = chunk else {
                        break
                    }

                    // Write the chunk to the destination file.

                    do {
                        try fileHandle.write(contentsOf: chunk)
                    } catch {
                        throw TransferError.writeError
                    }
                }

                break
            default:
                throw TransferError.downloadError("Received data is unknown
object type")
```

```
            }
        } catch {
            throw TransferError.downloadError("Error downloading the file:
\(error)")
        }

        print("File downloaded to \(fileURL.path).")
    }
```

Obtenha um objeto Amazon S3 de um ponto de Multi-Region acesso usando um AWS SDK

O exemplo de código a seguir mostra como obter um objeto de um ponto de Multi-Region acesso.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o cliente do S3 para usar o algoritmo de assinatura Asymmetric Sigv4 (Sigv4a).

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a)
signing algorithm.
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

Use o ARN do ponto de Multi-Region acesso em vez de um nome de bucket para recuperar o objeto.

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Kotlin.
- Para obter detalhes da API, consulte a GetObjectreferência da API AWS SDK for Kotlin.

Obtenha um objeto de um bucket do Amazon S3 usando um AWS SDK, especificando um cabeçalho If-Modified-Since

O exemplo de código a seguir mostra como ler dados de um objeto em um bucket do S3, mas somente se esse bucket não tiver sido modificado desde a última recuperação.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};
use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{uuid}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
```

```
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };

    // Request the object's headers.
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await;

    // If the `HeadObject` request succeeded, create a tuple containing the
    // values of the headers `last-modified` and `etag`. If the request
    // failed, return the error in a tuple instead.
    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => (Err(err), String::new()),
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and first GetObject had differing eTags"
    );

    println!("First value of last_modified: {last_modified:?}");
    println!("First tag: {}\n", e_tag_1);
```

```
    // Send a second `HeadObject` request. This time, the `if_modified_since`
    // option is specified, giving the `last_modified` value returned by the
    // first call to `HeadObject`.
    //
    // Since the object hasn't been changed, and there are no other objects in
    // the bucket, there should be no matching objects.

    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

    // If the `HeadObject` request succeeded, the result is a typle containing
    // the `last_modified` and `e_tag_1` properties. This is _not_ the expected
    // result.
    //
    // The _expected_ result of the second call to `HeadObject` is an
    // `SdkError::ServiceError` containing the HTTP error response. If that's
    // the case and the HTTP status is 304 (not modified), the output is a
    // tuple containing the values of the HTTP `last-modified` and `etag`
    // headers.
    //
    // If any other HTTP error occurred, the error is returned as an
    // `SdkError::ServiceError`.

    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => match err {
            SdkError::ServiceError(err) => {
                // Get the raw HTTP response. If its status is 304, the
                // object has not changed. This is the expected code path.
                let http = err.raw();
                match http.status().as_u16() {
                    // If the HTTP status is 304: Not Modified, return a
                    // tuple containing the values of the HTTP
                    // `last-modified` and `etag` headers.
                    304 => (
                        Ok(DateTime::from_str(
```

```
                            http.headers().get("last-modified").unwrap(),
                            DateTimeFormat::HttpDate,
                        )
                        .unwrap()),
                        http.headers().get("etag").map(|t| t.into()).unwrap(),
                    ),
                    // Any other HTTP status code is returned as an
                    // `SdkError::ServiceError`.
                    _ => (Err(SdkError::ServiceError(err)), String::new()),
                }
            }
            // Any other kind of error is returned in a tuple containing the
            // error and an empty string.
            _ => (Err(err), String::new()),
        },
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and second HeadObject had different eTags"
    );

    println!("Second value of last modified: {last_modified:?}");
    println!("Second tag: {}", e_tag_2);

    // Clean up by deleting the object and the bucket.
    client
        .delete_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await?;

    client
        .delete_bucket()
        .bucket(bucket_name.as_str())
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetObjectreferência da API AWS SDK for Rust.

Comece a usar a criptografia para objetos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como começar a usar a criptografia de objetos do Amazon S3.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        try
```

```
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

                // Download the object and verify that its contents match what
you uploaded.
                await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

                // Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

                // Copy both the source and target objects using server-side
encryption with
                // an encryption key.
                await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
        /// object will be uploaded.</param>
        /// <param name="keyName">The name of the object to upload to the Amazon
S3
```

```
        /// bucket.</param>
        /// <param name="base64Key">The encryption key.</param>
        /// <returns>The PutObjectRequest object for use by
 DownloadObjectAsync.</returns>
        public static async Task<PutObjectRequest> UploadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
            PutObjectResponse putObjectResponse = await
 client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }

        /// <summary>
        /// Downloads an encrypted object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
        /// <param name="base64Key">The encryption key used to encrypt the
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key,
```

```
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same
as we set");
                }
                else
                {
                    Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
                }
            }
        }

        /// <summary>
```

```
        /// Retrieves the metadata associated with an Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to call GetObjectMetadataAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
        /// object for which we want to retrieve metadata.</param>
        /// <param name="keyName">The name of the object for which we wish to
        /// retrieve the metadata.</param>
        /// <param name="base64Key">The encryption key associated with the
        /// object.</param>
        public static async Task GetObjectMetadataAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }

        /// <summary>
        /// Copies an encrypted object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// CopyObjectAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket containing the object
        /// to copy.</param>
```

```
        /// <param name="keyName">The name of the object to copy.</param>
        /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
        /// will be copied.</param>
        /// <param name="aesEncryption">The encryption type to use.</param>
        /// <param name="base64Key">The encryption key to use.</param>
        public static async Task CopyObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string copyTargetKeyName,
            Aes aesEncryption,
            string base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,

                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CopyObject

- GetObject
- GetObjectMetadata

Comece a usar tags para objetos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como começar a usar etiquetas de objetos do Amazon S3.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "newobject.txt";
        string filePath = @"*** file path ***";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        var client = new AmazonS3Client(bucketRegion);
        await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
```

```
        }

        /// <summary>
        /// This method uploads an object with tags. It then shows the tag
        /// values, changes the tags, and shows the new tags.
        /// </summary>
        /// <param name="client">The Initialized Amazon S3 client object used
        /// to call the methods to create and change an objects tags.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object will be stored.</param>
        /// <param name="keyName">A string representing the key name of the
        /// object to be tagged.</param>
        /// <param name="filePath">The directory location and file name of the
        /// object to be uploaded to the Amazon S3 bucket.</param>
        public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
        {
            try
            {
                // Create an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Keyx1", Value = "Value1" },
                        new Tag { Key = "Keyx2", Value = "Value2" },
                    },
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);

                // Now retrieve the new object's tags.
                GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
```

```
                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

                // Display the tag values.
                objectTags.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

                Tagging newTagSet = new Tagging()
                {
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Key3", Value = "Value3" },
                        new Tag { Key = "Key4", Value = "Value4" },
                    },
                };

                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet,
                };

                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // Retrieve the tags again and show the values.
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

                objectTags2.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
            }
            catch (AmazonS3Exception ex)
            {
```

```
                Console.WriteLine(
                        $"Error: '{ex.Message}'");
            }
        }
    }
```

- Para obter detalhes da API, consulte GetObjectTaggingga Referência AWS SDK para .NET da API.

Introdução aos clusters de processamento de big data

O exemplo de código a seguir mostra como:

- Criar um par de chaves do EC2.
- Configurar o armazenamento e preparar uma aplicação.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
```

```
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
 true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
 then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
```

```
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"
```

```
# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
 find the top 10 establishments
```

```
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
 's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red
 violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py
```

```
# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
 data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
 text 2>/dev/null || true)
```

```
if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi
```

```
if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout
 period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
```

```
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
 "$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
 files"

# Download results
```

```
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
 | head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
 | head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup
```

```
echo "Script completed successfully."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb
  - Rb
  - Rm
  - ssh
  - TerminateClusters
  - Wait

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
```

```
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
```

```
    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
```

```
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
```

```
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
```

```
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
```

```
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Introdução à extração de texto de documentos

O exemplo de código a seguir mostra como:

- Criar um bucket do S3.
- Fazer upload de um documento no S3.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon Textract Getting Started Tutorial"
echo "====================================================="
echo "This script will guide you through using Amazon Textract to analyze
 document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
```

```
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*/\1=***REDACTED***/g'
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to
 delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ];
 then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi
```

```
AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
 default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()
```

```
# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
 2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
 {"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
 50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi
```

```
# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "./$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
 2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ./$DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
```

```
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "====================================================="
echo "Analysis Summary"
echo "====================================================="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
 '{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
 echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
 echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
 echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
 echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
 echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
 | wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
 wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
```

```
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "====================================================="
echo "RESOURCES CREATED"
echo "====================================================="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "====================================================="
echo "CLEANUP CONFIRMATION"
echo "====================================================="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

echo ""
echo "====================================================="
```

```
echo "Tutorial complete!"
echo "==================================================="
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AnalyzeDocument
  - Cp
  - Help
  - Mb
  - Rb
  - Rm

Conceitos básicos de arquivos de atributos de machine learning

O exemplo de código a seguir mostra como:

- Configurar permissões do IAM.
- Crie uma função SageMaker de execução
- Criar grupo de atributos.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"
```

```
        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
```

```
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2
```

```
    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
```

```
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
```

```
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
```

```
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
```

```
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
```

```
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
```

```
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)
```

```
if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"
```

```
# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
```

```
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Introdução ao armazenamento de objetos

O exemplo de código a seguir mostra como:

- Criar seu primeiro bucket do S3.
- Fazer upload de um objeto.

- Habilitar o versionamento.
- Configurar a criptografia padrão.
- Adicionar tags a um bucket.
- Listar objetos e versões.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail

# ===============================================================================
# Prerequisites check
# ===============================================================================

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
 AWS_DEFAULT_REGION."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
    exit 1
```

```
fi

# ==============================================================================
# Setup: logging, temp directory, resource tracking
# ==============================================================================

UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || true)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="s3api-${UNIQUE_ID}"
fi

TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
LOG_FILE="${TEMP_DIR}/s3-gettingstarted.log"
CREATED_RESOURCES=()

exec > >(tee -a "$LOG_FILE") 2>&1

echo "============================================"
echo "S3 Getting Started"
echo "============================================"
echo "Bucket name: ${BUCKET_NAME}"
echo "Temp directory: ${TEMP_DIR}"
echo "Log file: ${LOG_FILE}"
echo ""

# ==============================================================================
# Helper functions
# ==============================================================================

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}
```

```
delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0

    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
 then
        return 0
    fi

    echo "$versions" | jq -r '.[] | "\(.Key)\t\(.VersionId)"' 2>/dev/null | while
 IFS=$'\t' read -r key version_id; do
        if [ -n "$key" ] && [ "$key" != "null" ]; then
            aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
 "$version_id" >/dev/null 2>&1 || true
        fi
    done

    return 0
}

# ==============================================================================
# Error handling and cleanup functions
# ==============================================================================

cleanup() {
    echo ""
    echo "============================================="
    echo "CLEANUP"
    echo "============================================="

    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[].
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[].
{Key:Key,VersionId:VersionId}" || true
```

```
        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/
null; then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "============================================"
    echo "ERROR on line ${line_number}"
    echo "============================================"
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@]}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}
```

```
trap 'handle_error "$LINENO"' ERR

# ==============================================================================
# Step 1: Create a bucket
# ==============================================================================

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1;
 then
            echo "ERROR: Failed to create bucket $BUCKET_NAME"
            exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
 2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
        fi
    fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag bucket"
    fi
```

```
fi
echo ""

# ==============================================================================
# Step 2: Upload a sample text file
# ==============================================================================

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# ==============================================================================
# Step 3: Download the object
# ==============================================================================

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then
    echo "ERROR: Failed to download sample.txt"
    exit 1
fi
echo "Downloaded to: ${DOWNLOAD_FILE}"
echo "Contents:"
cat "$DOWNLOAD_FILE"
echo ""
```

```
# ==============================================================================
# Step 4: Copy the object to a folder prefix
# ==============================================================================

echo "Step 4: Copying object to a folder prefix..."

if ! aws s3api copy-object \
    --bucket "$BUCKET_NAME" \
    --copy-source "${BUCKET_NAME}/sample.txt" \
    --key "backup/sample.txt" \
    --server-side-encryption AES256 \
    --metadata-directive COPY >/dev/null 2>&1; then
    echo "ERROR: Failed to copy object to backup/sample.txt"
    exit 1
fi
echo "Object copied to backup/sample.txt."
echo ""

# ==============================================================================
# Step 5: Enable versioning and upload a second version
# ==============================================================================

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \
    --versioning-configuration Status=Enabled >/dev/null 2>&1; then
    echo "ERROR: Failed to enable versioning"
    exit 1
fi
echo "Versioning enabled."

echo "Uploading a second version of sample.txt..."
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is version 2 of the sample file.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
```

```
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# ==============================================================================
# Step 6: Configure SSE-S3 encryption
# ==============================================================================

echo "Step 6: Configuring SSE-S3 default encryption..."

if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi
echo "SSE-S3 encryption configured."
echo ""

# ==============================================================================
# Step 7: Block all public access
# ==============================================================================

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
```

```
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# ==============================================================================
# Step 8: Configure bucket logging
# ==============================================================================

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
 true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
 2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \
        --bucket "$LOG_TARGET_BUCKET" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag log bucket"
    fi
```

```
    aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

    if ! aws s3api put-bucket-logging \
        --bucket "$BUCKET_NAME" \
        --bucket-logging-status '{
            "LoggingEnabled": {
                "TargetBucket": "'$LOG_TARGET_BUCKET'",
                "TargetPrefix": "logs/"
            }
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to configure bucket logging"
    else
        echo "Bucket logging configured."
    fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

# ==============================================================================
# Step 9: Tag the bucket
# ==============================================================================

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            },
            {
                "Key": "Environment",
                "Value": "Tutorial"
            },
            {
                "Key": "Project",
```

```
                "Value": "S3-GettingStarted"
            },
            {
                "Key": "ManagedBy",
                "Value": "Bash-Tutorial"
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to tag bucket"
    exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# ==============================================================================
# Step 10: List objects and versions
# ==============================================================================

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# ==============================================================================
# Step 11: Cleanup
# ==============================================================================

echo ""
echo "============================================"
echo "TUTORIAL COMPLETE"
```

```
echo "============================================="
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "============================================="
echo "CLEANUP"
echo "============================================="
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - HeadObject
  - ListObjectVersions
  - ListObjectsV2
  - PutBucketEncryption
  - PutBucketTagging
  - PutBucketVersioning
  - PutObject
  - PutPublicAccessBlock

Introdução à análise de consulta

O exemplo de código a seguir mostra como:

- Criar um bucket do S3 para resultados de consulta.
- Criar um banco de dados.
- Criar uma tabela.
- Executar uma consulta.
- Criar e usar consultas nomeadas.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
```

```
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
 [ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
```

```
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
```

```
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~
 ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
 2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'

    # Security: Enable S3 bucket encryption with KMS validation
    echo "Enabling default encryption on S3 bucket..."
    if ! aws s3api put-bucket-encryption \
        --bucket "$S3_BUCKET" \
        --server-side-encryption-configuration '{
            "Rules": [{
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
```

```
                }
            }]
        }' 2>&1; then
        echo "Warning: Could not enable encryption on bucket"
    fi

    # Security: Block public access
    echo "Blocking public access to S3 bucket..."
    if ! aws s3api put-public-access-block \
        --bucket "$S3_BUCKET" \
        --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1; then
        echo "Warning: Could not block public access on bucket"
    fi

    # Security: Enable versioning for data protection
    echo "Enabling versioning on S3 bucket..."
    if ! aws s3api put-bucket-versioning \
        --bucket "$S3_BUCKET" \
        --versioning-configuration Status=Enabled 2>&1; then
        echo "Warning: Could not enable versioning on bucket"
    fi

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
```

```
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
 "$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.
$TABLE_NAME (
  \`Date\` DATE,
  Time STRING,
```

```
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  \"input.regex\" = \"^(?!#)([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s
+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+[^
\\\\(]+[\\\\(]([^\\\\;]+).*\\\\%20([^\\\\/]+)[\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
```

```
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi
```

```
QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
 "$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
```

```
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
```

```
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID"
 \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
```

```
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."
```

```
    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
 "$NAMED_QUERY_ID" \
        --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
 | cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Table deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Table deletion in progress, status: $QUERY_STATUS"
                sleep 2
```

```
                ((ELAPSED+=2))
            done
        fi
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \
        --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Empty and delete S3 bucket (only if not shared)
```

```
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi

        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
        if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
        else
            echo "S3 bucket deleted successfully."
        fi
    else
        echo "Skipping S3 bucket deletion (shared resource)"
    fi

    # Security: Remove downloaded query results
    if [ -f "./query-results.csv" ]; then
        if command -v shred &>/dev/null; then
            shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - Cp
  - CreateNamedQuery
  - DeleteNamedQuery
  - GetNamedQuery

- GetQueryExecution
- GetQueryResults
- ListDatabases
- ListNamedQueries
- ListTableMetadata
- Mb
- Rb
- Rm
- StartQueryExecution

Trabalhe com os recursos de bloqueio de objetos do Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como trabalhar com os recursos do Bloqueio de Objetos do S3.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do Amazon S3.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
```

```
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
 lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
 or overwrite objects.
        5. Clean up objects and buckets.
   */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    private static string _resourcePrefix = null!;
    private static string noLockBucketName = null!;
    private static string lockEnabledBucketName = null!;
    private static string retentionAfterCreationBucketName = null!;
    private static List<string> bucketNames = new List<string>();
    private static List<string> fileNames = new List<string>();

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3ActionsWrapper>()
            )
            .Build();
```

```
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ConfigurationSetup();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            await Setup(true);

            await DemoActionChoices();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
```

```
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-
creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }

    // <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
 several S3\n" +
            "buckets and files to demonstrate working with S3 locking features.
\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without
 object lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
 false);
```

```
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
true);
        await
_s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nA bucket can be configured to use object locking
with a default retention period.");
        await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
```

```
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
                    var exampleFileName = fileNames[i];
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
```

```
                                    $"\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for
1 day.
                                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                        bucketName, exampleFileName,
                                        ObjectLockRetentionMode.Governance,
                                        DateTime.UtcNow.AddDays(1));
                                }
                                break;
                            }
                        }
                    }
                }
            }
            Console.WriteLine(new string('-', 80));
            return true;
        }

        // <summary>
        /// List all of the current buckets and objects.
        /// </summary>
        /// <param name="interactive">True to run as interactive.</param>
        /// <returns>The list of buckets and objects.</returns>
        public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
        {
            var allObjects = new List<S3ObjectVersion>();
            foreach (var bucketName in bucketNames)
            {
                var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
                foreach (var objectKey in objectsInBucket.Versions)
                {
                    allObjects.Add(objectKey);
                }
            }
```

```
        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the scenario."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
        {
            Console.WriteLine(new string('-', 80));
            choice = GetChoiceResponse(
                "\nExplore the S3 locking features by selecting one of the
following choices:"
                , choices);
            Console.WriteLine(new string('-', 80));
            switch (choice)
            {
                case 0:
```

```
                    {
                        await ListBucketsAndObjects(true);
                        break;
                    }
                case 1:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 2:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 3:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
overwrite:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        // Create the file if it does not already exist.
                        if (!File.Exists(allFiles[fileChoice].Key))
                        {
                            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
                            await sw.WriteLineAsync(
                                "This is a sample file for uploading to a
bucket.");
                        }
```

```
                        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
                        break;
                    }
                case 4:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
                        break;
                    }
                case 5:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        break;
                    }
            }
        }
        return true;
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
```

```
        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
                // Check for a legal hold.
                var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
                var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
                await
_s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
```

```
        return choiceNumber - 1;
    }
}
```

Uma classe de wrapper para funções do S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
 bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
```

```
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
```

```
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
```

```
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
```

```
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };
```

```
            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                                $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
```

```
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                              $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.<returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };
```

```
        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\tSuccessfully uploaded {objectName} to
 {bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\tCould not upload {objectName} to
 {bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
 bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
 objectKey, bool hasRetention, string? versionId = null)
```

```
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do Amazon S3.

```
import (
 "context"
 "fmt"
 "log"
```

```
 "strings"

 "s3_object_lock/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
 questioner demotools.IQuestioner
 resources  Resources
 s3Actions  *actions.S3Actions
 sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) ObjectLockScenario {
 scenario := ObjectLockScenario{
  questioner: questioner,
  resources:  Resources{},
  s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
  sdkConfig:  sdkConfig,
 }
 scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
 scenario.resources.init(scenario.s3Actions, questioner)
 return scenario
}

type nameLocked struct {
 name   string
 locked bool
}

var createInfo = []nameLocked{
 {"standard-bucket", false},
 {"lock-bucket", true},
 {"retention-bucket", false},
}
```

```
// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
 log.Println("Let's create some S3 buckets to use for this workflow.")
 success := false
 for !success {
  prefix := scenario.questioner.Ask(
   "This example creates three buckets. Enter a prefix to name your buckets
 (remember bucket names must be globally unique):")

  for _, info := range createInfo {
   log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
   bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
 fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
   if err != nil {
    switch err.(type) {
    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
     log.Printf("Couldn't create bucket %s.\n", bucketName)
    default:
     panic(err)
    }
    break
   }
   scenario.resources.demoBuckets[info.name] = &DemoBucket{
    name:       bucketName,
    objectKeys: []string{},
   }
   log.Printf("Created bucket %s.\n", bucketName)
  }

  if len(scenario.resources.demoBuckets) < len(createInfo) {
   scenario.resources.deleteBuckets(ctx)
  } else {
   success = true
  }
 }

 log.Println("S3 buckets created.")
 log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking.")
 scenario.questioner.Ask("Press Enter to continue.")
```

```
 var err error
 bucket := scenario.resources.demoBuckets["retention-bucket"]
 err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Object locking enabled on bucket %s.", bucket.name)
 }

 log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
 bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
 context.Context) {
 log.Println("\nA bucket can be configured to use object locking with a default
 retention period.")

 bucket := scenario.resources.demoBuckets["retention-bucket"]
 retentionPeriod := scenario.questioner.AskInt("Enter the default retention
 period in days: ")
 err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
 bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
 types.ObjectLockRetentionModeGovernance)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't configure a default retention period on bucket %s.\n",
 bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Default retention policy set on bucket %s with %d day retention
 period.", bucket.name, retentionPeriod)
  bucket.retentionEnabled = true
 }
```

```
 log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
 log.Println("Uploading test objects to S3 buckets.")

 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  for i := 0; i < 2; i++ {
   key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
 fmt.Sprintf("example-%d", i),
    fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
   if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
     log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
   }
  }
 }

 scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
 log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
 objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
 context.Context) {
 log.Println("Now let's set object lock configurations on individual objects.")

 buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
 scenario.resources.demoBuckets["retention-bucket"]}
 for _, bucket := range buckets {
  for index, objKey := range bucket.objectKeys {
   switch index {
   case 0:
```

```
    if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
 to %s in %s (y/n)? ", objKey, bucket.name), "y") {
     err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
 types.ObjectLockLegalHoldStatusOn)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set legal hold on %s.\n", objKey)
      default:
       panic(err)
      }
     } else {
      log.Printf("Legal hold set on %s.\n", objKey)
     }
    }
   case 1:
    q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
 %s in %s?\n"+
     "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
 able to delete this object\n"+
     "or its bucket until the retention period has expired. (y/n) ", objKey,
 bucket.name)
    if scenario.questioner.AskBool(q, "y") {
     err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
 types.ObjectLockRetentionModeGovernance, 1)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
 bucket.name)
      default:
       panic(err)
      }
     } else {
      log.Printf("Retention period set to 1 for %s.", objKey)
      bucket.retentionEnabled = true
     }
    }
   }
  }
 }
 log.Println(strings.Repeat("-", 88))
}
```

```
const (
 ListAll = iota
 DeleteObject
 DeleteRetentionObject
 OverwriteObject
 ViewRetention
 ViewLegalHold
 Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
 object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
 log.Println("Now you can interact with the objects to explore the object lock
 configurations.")
 interactiveChoices := []string{
  "List all objects and buckets.",
  "Attempt to delete an object.",
  "Attempt to delete an object with retention period bypass.",
  "Attempt to overwrite a file.",
  "View the retention settings for an object.",
  "View the legal hold settings for an object.",
  "Finish the workflow."}

 choice := ListAll
 for choice != Finish {
  objList := scenario.GetAllObjects(ctx)
  objChoices := scenario.makeObjectChoiceList(objList)
  choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
 interactiveChoices)
  switch choice {
  case ListAll:
   log.Println("The current objects in the example buckets are:")
   for _, objChoice := range objChoices {
    log.Println("\t", objChoice)
   }
  case DeleteObject, DeleteRetentionObject:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
 delete:\n", objChoices)
   obj := objList[objChoice]
   deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
 obj.versionId, choice == DeleteRetentionObject)
   if err != nil {
    switch err.(type) {
```

```
    case *types.NoSuchKey:
     log.Println("Nothing to delete.")
    default:
     panic(err)
    }
   } else if deleted {
    log.Printf("Object %s deleted.\n", obj.key)
   }
  case OverwriteObject:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
   obj := objList[objChoice]
   _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
   if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
     log.Println("Couldn't upload to nonexistent bucket.")
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded new content to object %s.\n", obj.key)
   }
  case ViewRetention:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
   obj := objList[objChoice]
   retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Can't get retention configuration for %s.\n", obj.key)
    default:
     panic(err)
    }
   } else if retention != nil {
    log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
   } else {
    log.Printf("Object %s does not have object retention configured.\n", obj.key)
   }
  case ViewLegalHold:
```

```
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
 view:\n", objChoices)
    obj := objList[objChoice]
    legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
 obj.key, obj.versionId)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
     default:
      panic(err)
     }
    } else if legalHold != nil {
     log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
    } else {
     log.Printf("Object %s does not have legal hold configured.", obj.key)
    }
   case Finish:
    log.Println("Let's clean up.")
   }
   log.Println(strings.Repeat("-", 88))
 }
}

type BucketKeyVersionId struct {
 bucket    string
 key       string
 versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
 them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
 []BucketKeyVersionId {
 var objectList []BucketKeyVersionId
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't get object versions for %s.\n", bucket.name)
   default:
    panic(err)
```

```
    }
   } else {
    for _, version := range versions {
     objectList = append(objectList,
      BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
 *version.VersionId})
    }
   }
  }
  return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
 are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
 []BucketKeyVersionId) []string {
 choices := make([]string, len(bucketObjects))
 for i := 0; i < len(bucketObjects); i++ {
  choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
   bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
 }
 return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := scenario.questioner.(*demotools.MockQuestioner)
   if isMock || scenario.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   scenario.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
 log.Println(strings.Repeat("-", 88))

 scenario.CreateBuckets(ctx)
```

```
 scenario.EnableLockOnBucket(ctx)
 scenario.SetDefaultRetentionPolicy(ctx)
 scenario.UploadTestObjects(ctx)
 scenario.SetObjectLockConfigurations(ctx)
 scenario.InteractWithObjects(ctx)

 scenario.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina um struct que envolva as ações do S3 usadas neste exemplo.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
```

```
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
```

```
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
```

```
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
  "ObjectLockConfigurationNotFoundError" {
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  } else {
   lockConfig = output.ObjectLockConfiguration
  }

  return lockConfig, err
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
```

```
 }

 return retention, err
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
```

```
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }
```

```
 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }
```

```
 return err
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
```

```
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
```

```
    case "AccessDenied":
     log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
     err = nil
    case "InvalidArgument":
     if bypassGovernance {
      log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
      err = nil
     }
    }
   }
  } else {
   err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
 time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
   } else {
    deleted = true
   }
  }
  return deleted, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
```

```
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *delOut.Errors[0].Message)
  }
 } else {
  for _, delObjs := range delOut.Deleted {
   err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
 time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to be deleted.\n",
 *delObjs.Key)
   } else {
    log.Printf("Deleted %s.\n", *delObjs.Key)
   }
  }
 }
 return err
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "s3_object_lock/actions"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
 name             string
 retentionEnabled bool
 objectKeys       []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
 and handles
// cleanup when the scenario finishes.
type Resources struct {
 demoBuckets map[string]*DemoBucket

 s3Actions  *actions.S3Actions
 questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
 demotools.IQuestioner) {
 resources.s3Actions = s3Actions
 resources.questioner = questioner
 resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 " +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
```

```
  "during this demo (y/n)?", "y")
 if !wantDelete {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
  return
 }

 log.Println("Removing objects from S3 buckets and deleting buckets...")
 resources.deleteBuckets(ctx)
 //resources.deleteRetentionObjects(resources.retentionBucket,
 resources.retentionObjects)

 log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
 ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  resources.deleteObjects(ctx, bucket)
  _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucket.name),
  })
  if err != nil {
   panic(err)
  }
 }
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
 bucket.name)
  } else {
   log.Printf("Deleted %s.\n", bucket.name)
  }
 }
 resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
```

```
func (resources *Resources) deleteObjects(ctx context.Context, bucket
 *DemoBucket) {
 lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
 bucket.name)
 if err != nil {
  panic(err)
 }
 versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("No objects to get from %s.\n", bucket.name)
  default:
   panic(err)
  }
 }
 delObjects := make([]types.ObjectIdentifier, len(versions))
 for i, version := range versions {
  if lockConfig != nil && lockConfig.ObjectLockEnabled ==
 types.ObjectLockEnabledEnabled {
   status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
 *version.Key, *version.VersionId)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Couldn't determine legal hold status for %s in %s.\n",
 *version.Key, bucket.name)
    default:
     panic(err)
    }
   } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
    err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
 *version.VersionId, types.ObjectLockLegalHoldStatusOff)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
 bucket.name)
     default:
      panic(err)
     }
    }
   }
  }
```

```
  delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
 version.VersionId}
 }
 err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
 bucket.retentionEnabled)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Println("Nothing to delete.")
  default:
   panic(err)
  }
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do Amazon S3.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        final String usage = """
            Usage:
                <bucketName> \s

            Where:
                bucketName - The Amazon S3 bucket name.
           """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
 Object Locking Feature Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
```

```
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
```

```
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
```

```
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to
view:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }

    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
```

```
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                    }
                }
                // Check for a retention period.
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal
lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking
features.
                """);
```

```
        System.out.println("S3 buckets can be created either with or without
object lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
```

```
                    fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on
individual files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
```

```
                                Reminder: Only a user with the
 s3:BypassGovernanceRetention permission will be able to delete this file or its
 bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

 s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
    }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }
```

```
        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid
 number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

Uma classe de wrapper para funções do S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
```

```
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
 software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
 {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time
 zone.
        ZonedDateTime zonedDateTime =
 futureInstant.atZone(ZoneId.systemDefault());
```

```
        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
```

```
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                }))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": "+ s3InfoObject.getKeyName());
```

```
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();
```

```
        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
```

```
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath)
{
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
```

```
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
```

```
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }

    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
```

```
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Ponto de entrada do cenário (index.js). Isso orquestra todas as etapas. Visite GitHub para ver os detalhes de implementação do Cenário ScenarioInput ScenarioOutput,, ScenarioAction e.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
```

```
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
```

```
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Object Locking - Demo",
      [loadState, replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Object Locking - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
```

```
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
 features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Envie mensagens de boas-vindas para o console (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature
 Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
 several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Implante buckets, objetos e configurações de arquivos (setup.steps.js).

```
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
  GetBucketVersioningCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

/**
 * @param {Scenarios} scenarios
 */
```

```
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
         ${state.bucketPrefix}-no-lock with object lock False.
         ${state.bucketPrefix}-lock-enabled with object lock True.
         ${state.bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${state.bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

    try {
      await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
      await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: lockEnabledBucketName,
          ObjectLockEnabledForBucket: true,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: lockEnabledBucketName },
      );
      await client.send(
        new CreateBucketCommand({ Bucket: retentionBucketName }),
      );
```

```
      await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

      state.noLockBucketName = noLockBucketName;
      state.lockEnabledBucketName = lockEnabledBucketName;
      state.retentionBucketName = retentionBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
         file0.txt in ${state.bucketPrefix}-no-lock.
         file1.txt in ${state.bucketPrefix}-no-lock.
         file0.txt in ${state.bucketPrefix}-lock-enabled.
         file1.txt in ${state.bucketPrefix}-lock-enabled.
         file0.txt in ${state.bucketPrefix}-retention-after-creation.
         file1.txt in ${state.bucketPrefix}-retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );
```

```
/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          Body: "Content",
```

```
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
 ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    (state) => `A bucket can be configured to use object locking with a default
 retention period.
A default retention period will be configured for ${state.bucketPrefix}-
retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
```

```
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    const getBucketVersioning = new GetBucketVersioningCommand({
      Bucket: state.retentionBucketName,
    });

    await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
      const { Status } = await client.send(getBucketVersioning);
      if (Status !== "Enabled") {
        throw new Error("Bucket versioning is not enabled.");
      }
    });

    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });
```

```
/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
```

```
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.lockEnabledBucketName}?
```

```
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
```

```
      `Would you like to add a legal hold to file0.txt in
 ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
```

```
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
  getBucketPrefix,
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
```

```
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

Visualize e exclua arquivos nos buckets (repl.steps.js).

```
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};
```

```
/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
```

```
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.noLockBucketName,
            state.lockEnabledBucketName,
            state.retentionBucketName,
          ]);
          state.replOutput = files
```

```
            .map(
              (file) =>
                `${file.bucket}: ${file.key} (version: ${file.version})`,
            )
            .join("\n");
          break;
        }
        case choices.DELETE_FILE: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.DELETE_FILE_WITH_RETENTION: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
```

```
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.OVERWRITE_FILE: {
          /** @type {number} */
          const fileToOverwrite = await fileInput.handle(state);
          const selectedFile = files[fileToOverwrite];
          try {
            await client.send(
              new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
              }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.VIEW_RETENTION_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const retention = await client.send(
              new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            const bucketConfig = await client.send(
              new GetObjectLockConfigurationCommand({
                Bucket: selectedFile.bucket,
              }),
            );
```

```
            state.replOutput = `Object retention for ${selectedFile.key}
 in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
 ${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
 ${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
          } catch (err) {
            state.replOutput = `Unable to fetch object lock retention:
 '${err.message}'`;
          }
          break;
        }
        case choices.VIEW_LEGAL_HOLD_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const legalHold = await client.send(
              new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Object legal hold for ${selectedFile.key} in
 ${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
          } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
          }
          break;
        }
        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
          "REPL output",
          (state) => state.replOutput,
          { preformatted: true },
```

```
        ),
      },
    },
  );

export { replInput, replAction, choices };
```

Destrua todos os recursos criados (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;
```

```
const buckets = [
  noLockBucketName,
  lockEnabledBucketName,
  retentionBucketName,
];

for (const bucket of buckets) {
  /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
  let objectsResponse;

  try {
    objectsResponse = await client.send(
      new ListObjectVersionsCommand({
        Bucket: bucket,
      }),
    );
  } catch (e) {
    if (e instanceof Error && e.name === "NoSuchBucket") {
      console.log("Object's bucket has already been deleted.");
      continue;
    }
    throw e;
  }

  for (const version of objectsResponse.Versions || []) {
    const { Key, VersionId } = version;

    try {
      const legalHold = await client.send(
        new GetObjectLegalHoldCommand({
          Bucket: bucket,
          Key,
          VersionId,
        }),
      );

      if (legalHold.LegalHold?.Status === "ON") {
        await client.send(
          new PutObjectLegalHoldCommand({
            Bucket: bucket,
            Key,
            VersionId,
            LegalHold: {
```

```
                  Status: "OFF",
                },
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch legal hold for ${Key} in ${bucket}:
'${err.message}'`,
          );
        }

        try {
          const retention = await client.send(
            new GetObjectRetentionCommand({
              Bucket: bucket,
              Key,
              VersionId,
            }),
          );

          if (retention.Retention?.Mode === "GOVERNANCE") {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
                BypassGovernanceRetention: true,
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch object lock retention for ${Key} in ${bucket}:
'${err.message}'`,
          );
        }

        await client.send(
          new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
```

```
        );
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Faça solicitações condicionais do Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como adicionar pré-condições a solicitações do Amazon S3.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos de solicitação condicional do Amazon S3.

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This example demonstrates the use of conditional requests for S3 operations.
    You can use conditional requests to add preconditions to S3 read requests to
 return or copy
    an object based on its Entity tag (ETag), or last modified date.
    You can use a conditional write requests to prevent overwrites by ensuring
    there is no existing object with the same key.
   */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    public static string _resourcePrefix = null!;
    public static string _sourceBucketName = null!;
    public static string _destinationBucketName = null!;
    public static string _sampleObjectKey = null!;
    public static string _sampleObjectEtag = null!;
    public static bool _interactive = true;


    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3ActionsWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Conditional Requests Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            ConfigurationSetup();
            _sampleObjectEtag = await Setup(_sourceBucketName,
_destinationBucketName, _sampleObjectKey);

            await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,
_sampleObjectKey, _sampleObjectEtag, 0);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await CleanupScenario(_sourceBucketName, _destinationBucketName);
            Console.WriteLine(new string('-', 80));
```

```
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        _sourceBucketName = _resourcePrefix + "-source";
        _destinationBucketName = _resourcePrefix + "-dest";
        _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
    }

    /// <summary>
    /// Sets up the scenario by creating a source and destination bucket, and
uploading a test file to the source bucket.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The name of the test file to add to the source
bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string
destBucket, string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
several S3\n" +
            "buckets and files to demonstrate working with S3 conditional
requests.\n" +
            "This example demonstrates the use of conditional requests for S3
operations.\r\n" +
```

```
            "You can use conditional requests to add preconditions to S3 read
 requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r
\n" +
            "You can use a conditional write requests to prevent overwrites by
 ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.
\r\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
 example.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();

        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
 sourceBucket,
            "Test file content.");

        return eTag;
    }

    /// <summary>
    /// Cleans up the scenario by deleting the source and destination buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task CleanupScenario(string sourceBucket, string
 destBucket)
    {
        await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
        await _s3ActionsWrapper.CleanupBucketByName(destBucket);
    }

    /// <summary>
    /// Displays a list of the objects in the test buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
```

```
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task DisplayBuckets(string sourceBucket, string
 destBucket)
    {
        await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
        await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
    }

    /// <summary>
    /// Displays the menu of conditional request options for the user.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The key of the test object in the source
 bucket.</param>
    /// <param name="etag">The ETag of the test object in the source bucket.</
param>
    public static async Task DisplayDemoChoices(string sourceBucket, string
 destBucket, string objectKey, string etag, int defaultChoice)
    {
        var actions = new[]
        {
            "Print a list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit."
        };

        var conditions = new[]
        {
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
 fail.",
            "If-Modified-Since: using yesterday's date. This condition should
 succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
 fail."
        };

        var conditionTypes = new[]
        {
            S3ConditionType.IfMatch,
            S3ConditionType.IfNoneMatch,
```

```
            S3ConditionType.IfModifiedSince,
            S3ConditionType.IfUnmodifiedSince,
        };

        var yesterdayDate = DateTime.UtcNow.AddDays(-1);

        int choice;
        while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
 features by selecting one of the following choices:", actions, defaultChoice)) !
= 4)
        {
            switch (choice)
            {
                case 0:
                    Console.WriteLine("Listing the objects and buckets.");
                    await DisplayBuckets(sourceBucket, destBucket);
                    break;
                case 1:
                    int conditionTypeIndex = GetChoiceResponse("Perform a
 conditional read:", conditions, 1);
                    if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
                    {
                        await _s3ActionsWrapper.GetObjectConditional(objectKey,
 sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
                    }
                    else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)
                    {
                        await _s3ActionsWrapper.GetObjectConditional(objectKey,
 sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
                    }
                    break;
                case 2:
                    int copyConditionTypeIndex = GetChoiceResponse("Perform a
 conditional copy:", conditions, 1);
                    string destKey = GetStringResponse("Enter an object key:",
 "sampleObjectKey");
                    if (copyConditionTypeIndex == 0 || copyConditionTypeIndex ==
 1)
                    {
                        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
 destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
 etag);
                    }
```

```
                    else if (copyConditionTypeIndex == 2 ||
copyConditionTypeIndex == 3)
                    {
                        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
                    }
                    break;
                case 3:
                    Console.WriteLine("Perform a conditional write using
IfNoneMatch condition on the object key.");
                    Console.WriteLine("If the key is a duplicate, the write will
fail.");
                    string newObjectKey = GetStringResponse("Enter an object
key:", "newObjectKey");
                    await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
                    break;
            }

            if (!_interactive)
            {
                break;
            }
        }

        Console.WriteLine("Proceeding to cleanup.");
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
            await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);
```

```
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices, int
 defaultChoice)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
```

```
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        if (!_interactive)
            return defaultChoice;

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }

    /// <summary>
    /// Get a string response from the user.
    /// </summary>
    /// <param name="question">The question to print.</param>
    /// <param name="defaultAnswer">A default answer to use when not
interactive.</param>
    /// <returns>The string response.</returns>
    public static string GetStringResponse(string? question, string
defaultAnswer)
    {
        string? answer = "";
        if (_interactive)
        {
            do
            {
                Console.WriteLine(question);
                answer = Console.ReadLine();
            } while (string.IsNullOrWhiteSpace(answer));
        }
        else
        {
            answer = defaultAnswer;
        }

        return answer;
    }
```

```
}
```

Uma classe de wrapper para funções do S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    /// <param name="logger">The class logger.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
    {
        _amazonS3 = amazonS3;
        _logger = logger;
    }

    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch',
 'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
```

```
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
 sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
 string? etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read
 successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
            return true;
        }
```

```
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional read failed: Precondition failed");
            }
            else if (e.ErrorCode == "NotModified")
            {
                _logger.LogError("Conditional read failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }

    /// <summary>
    /// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
    /// </summary>
    /// <param name="objectKey">The key of the object to upload.</param>
    /// <param name="bucket">The source bucket of the object.</param>
    /// <param name="content">The content to upload as a string.</param>
    /// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
    public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
    {
        try
        {
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
```

```
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition
 failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }

    /// <summary>
    /// Copies an object from one Amazon S3 bucket to another with a conditional
 request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
    /// <param name="conditionType">The type of condition to apply, e.g.
 'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
 'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string
 destKey, string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
 string? etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
```

```
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            await _amazonS3.CopyObjectAsync(copyObjectRequest);
            _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional copy failed: Precondition failed");
            }
            else if (e.ErrorCode == "304")
            {
                _logger.LogError("Conditional copy failed: Object not modified");
            }
            else
            {
```

```
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithName(string bucketName)
    {
        Console.WriteLine($"\tCreating bucket {bucketName}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true
            };

            await _amazonS3.PutBucketAsync(request);
            var bucketReady = false;
            var retries = 5;
            while (!bucketReady && retries > 0)
            {
                Thread.Sleep(5000);
                bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
                retries--;
            }

            return bucketReady;
        }
        catch (BucketAlreadyExistsException ex)
        {
            Console.WriteLine($"Bucket already exists: '{ex.Message}'");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Cleans up objects and deletes the bucket by name.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public async Task CleanupBucketByName(string bucketName)
    {
        try
        {
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
            }
            await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
            Console.WriteLine($"Cleaned up bucket: {bucketName}.");
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
            }
            else
            {
                Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
                throw;
            }
        }
    }

    /// <summary>
    /// List the contents of the bucket with their ETag.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
    {
        var results = new List<S3Object>();
        try
        {
            Console.WriteLine($"\t Items in bucket {bucketName}");
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            if (listObjectsResponse.S3Objects.Count == 0)
            {
                Console.WriteLine("\t\tNo objects found.");
            }
            else
            {
                foreach (var obj in listObjectsResponse.S3Objects)
                {
                    Console.WriteLine($"\t\t object: {obj.Key} ETag {obj.ETag}");
                }
            }
            results = listObjectsResponse.S3Objects;

        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                _logger.LogError($"Bucket {bucketName} does not exist.");
            }
            else
            {
                _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
                throw;
            }
        }

        return results;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
```

```
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket by deleting the objects and then the bucket
itself.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CleanUpBucketByName(string bucketName)
    {
        try
        {
            var allFiles = await ListBucketContentsByName(bucketName);

            foreach (var fileInfo in allFiles)
            {
                await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
            }

            var request = new DeleteBucketRequest() { BucketName = bucketName, };
```

```
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CopyObject
  - GetObject
  - PutObject

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Ponto de entrada do fluxo de trabalho (index.js). Isso orquestra todas as etapas. Visite GitHub para ver os detalhes de implementação do Cenário ScenarioInput ScenarioOutput,, ScenarioAction e.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
```

```
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Conditional Requests - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Conditional Requests - Demo",
      [loadState, welcome(scenarios), replAction(scenarios, client)],
```

```
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Conditional Requests - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
 features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Envie mensagens de boas-vindas para o console (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
```

```
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3
 operations." +
      " You can use conditional requests to add preconditions to S3 read requests
 to return " +
      "or copy an object based on its Entity tag (ETag), or last modified
 date.You can use " +
      "a conditional write requests to prevent overwrites by ensuring there is no
 existing " +
      "object with the same key.\n" +
      "This example will enable you to perform conditional reads and writes that
 will succeed " +
      "or fail based on your selected options.\n" +
      "Sample buckets and a sample object will be created as part of the example.
\n" +
      "Some steps require a key name prefix to be defined by the user. Before you
 begin, you can " +
      "optionally edit this prefix in ./object_name.json. If you do so, please
 reload the scenario before you begin.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Implante buckets e objetos (setup.steps.js).

```
import {
  ChecksumAlgorithm,
  CreateBucketCommand,
```

```
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );
/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
         ${state.bucketPrefix}-source-bucket.
         ${state.bucketPrefix}-destination-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });
```

```
/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const sourceBucketName = `${state.bucketPrefix}-source-bucket`;
    const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;

    try {
      await client.send(
        new CreateBucketCommand({
          Bucket: sourceBucketName,
        }),
      );
      await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: destinationBucketName,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: destinationBucketName },
      );

      state.sourceBucketName = sourceBucketName;
      state.destinationBucketName = destinationBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
```

```
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
         file01.txt in ${state.bucketPrefix}-source-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.sourceBucketName,
          Key: "file01.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
 ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });
```

```
export {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
};
```

Obtenha, copie e coloque objetos usando solicitações condicionais do S3 (repl.steps.js).

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
  CopyObjectCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";
import data from "./object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
```

```
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
    "Explore the S3 conditional request features by selecting one of the
 following choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
        {
          name: "Perform a conditional copy. These examples use the key name
 prefix defined in ./object_name.json.",
          value: choices.CONDITIONAL_COPY,
        },
        {
          name: "Perform a conditional write. This example use the sample file ./
text02.txt.",
          value: choices.CONDITIONAL_WRITE,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
```

```
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key } = version;
      files.push({ bucket, key: Key });
    }
  }
  return files;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
  const objectsResponse = await client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    }),
  );
  return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
```

```
    ]);

    const fileInput = new scenarios.ScenarioInput(
      "selectedFile",
      "Select a file to use:",
      {
        type: "select",
        choices: files.map((file, index) => ({
          name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
            file.version
          })`,
          value: index,
        })),
      },
    );
    const condReadOptions = new scenarios.ScenarioInput(
      "selectOption",
      "Which conditional read action would you like to take?",
      {
        type: "select",
        choices: [
          "If-Match: using the object's ETag. This condition should succeed.",
          "If-None-Match: using the object's ETag. This condition should
fail.",
          "If-Modified-Since: using yesterday's date. This condition should
succeed.",
          "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ],
      },
    );
    const condCopyOptions = new scenarios.ScenarioInput(
      "selectOption",
      "Which conditional copy action would you like to take?",
      {
        type: "select",
        choices: [
          "If-Match: using the object's ETag. This condition should succeed.",
          "If-None-Match: using the object's ETag. This condition should
fail.",
          "If-Modified-Since: using yesterday's date. This condition should
succeed.",
          "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
```

```
          ],
        },
      );
      const condWriteOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional write action would you like to take?",
        {
          type: "select",
          choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate,
the write will fail.",
          ],
        },
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.sourceBucketName,
            state.destinationBucketName,
          ]);
          state.replOutput = files
            .map(
              (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
            )
            .join("\n");
          break;
        }
        case choices.CONDITIONAL_READ:
          {
            const selectedCondRead = await condReadOptions.handle(state);
            if (
              selectedCondRead ===
              "If-Match: using the object's ETag. This condition should succeed."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              const ETag = await getEtag(client, bucket, key);

              try {
                await client.send(
                  new GetObjectCommand({
```

```
                    Bucket: bucket,
                    Key: key,
                    IfMatch: ETag,
                  }),
                );
                state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because ETag provided matches the object's ETag.`;
              } catch (err) {
                state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-None-Match: using the object's ETag. This condition should
fail."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              const ETag = await getEtag(client, bucket, key);

              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfNoneMatch: ETag,
                  }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
              } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-Modified-Since: using yesterday's date. This condition should
succeed."
            ) {
              const date = new Date();
```

```
              date.setDate(date.getDate() - 1);

              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfModifiedSince: date,
                  }),
                );
                state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because it has been created or modified in the last 24 hours.`;
              } catch (err) {
                state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-Unmodified-Since: using yesterday's date. This condition should
fail."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";

              const date = new Date();
              date.setDate(date.getDate() - 1);
              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfUnmodifiedSince: date,
                  }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was
read.`;
              } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
              }
```

```
              break;
            }
          }
          break;
        case choices.CONDITIONAL_COPY: {
          const selectedCondCopy = await condCopyOptions.handle(state);
          if (
            selectedCondCopy ===
            "If-Match: using the object's ETag. This condition should succeed."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);

            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;
            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfMatch: ETag,
                }),
              );
              state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's
ETag.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-None-Match: using the object's ETag. This condition should fail."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);
```

```
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfNoneMatch: ETag,
                }),
              );
              state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
            } catch (err) {
              state.replOutput = `Unable to copy object as ${key} as as
${copiedKey} to bucket ${state.destinationBucketName}: ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-Modified-Since: using yesterday's date. This condition should
succeed."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            const date = new Date();
            date.setDate(date.getDate() - 1);

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
```

```
                  Key: copiedKey,
                  CopySourceIfModifiedSince: date,
                }),
              );
              state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the
last 24 hours.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-Unmodified-Since: using yesterday's date. This condition should
fail."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            const date = new Date();
            date.setDate(date.getDate() - 1);

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfUnmodifiedSince: date,
                }),
              );
              state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in
the last 24 hours.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
```

```
            }
          }
          break;
        }
        case choices.CONDITIONAL_WRITE:
          {
            const selectedCondWrite = await condWriteOptions.handle(state);
            if (
              selectedCondWrite ===
              "IfNoneMatch condition on the object key: If the key is a
duplicate, the write will fail."
            ) {
              // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
              const key = "text02.txt";
              const __filename = fileURLToPath(import.meta.url);
              const __dirname = dirname(__filename);
              const filePath = path.join(__dirname, "text02.txt");
              try {
                await client.send(
                  new PutObjectCommand({
                    Bucket: `${state.destinationBucketName}`,
                    Key: `${key}`,
                    Body: await readFile(filePath),
                    IfNoneMatch: "*",
                  }),
                );
                state.replOutput = `${key} uploaded to bucket
${state.destinationBucketName} because the key is not a duplicate.`;
              } catch (err) {
                state.replOutput = `Unable to upload object to bucket
${state.destinationBucketName}:${err.message}`;
              }
              break;
            }
          }
          break;

        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
```

```
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
          preformatted: true,
        }),
      },
    },
  );

export { replInput, choices };
```

Destrua todos os recursos criados (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
```

```
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          } catch (err) {
            console.log(`An error occurred: ${err.message} `);
          }
        }
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Objects and buckets have already been deleted.");
          continue;
        }
        throw e;
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CopyObject

- GetObject
- PutObject

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre as solicitações condicionais do Amazon S3.

```
"""
Purpose

Shows how to use AWS SDK for Python (Boto3) to get started using conditional
 requests for
Amazon Simple Storage Service (Amazon S3).

"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
 setup.
sys.path.append("../../../..")
import demo_tools.question as q  # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
```

```
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)


class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
 actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key:
 str):
        """
        Sets up the scenario by creating a source and destination bucket.
        Prompts the user to provide a bucket name prefix.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The name of a test file to add to the source bucket.
        """

        # Create the buckets.
        try:
            self.s3_client.create_bucket(Bucket=source_bucket)
            self.s3_client.create_bucket(Bucket=dest_bucket)
            print(
                f"Created source bucket: {source_bucket} and destination bucket:
 {dest_bucket}"
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            logger.error(f"Error creating buckets: {error_code}")
            raise

        # Upload test file into the source bucket.
        try:
            print(f"Uploading file {object_key} to bucket {source_bucket}")
```

```
            response = self.s3_client.put_object(
                Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
            )
            object_etag = response["ETag"]
            return object_etag

        except Exception as e:
            logger.error(
                f"Failed to upload file {object_key} to bucket {source_bucket}:
{e}"
            )


    def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
        """
        Cleans up the scenario by deleting the source and destination buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.cleanup_bucket(source_bucket)
        self.cleanup_bucket(dest_bucket)

    def cleanup_bucket(self, bucket_name: str):
        """
        Cleans up the bucket by deleting all objects and then the bucket itself.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            for obj in objs:
                key = obj["Key"]
                self.s3_client.delete_object(Bucket=bucket_name, Key=key)
            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Cleaned up bucket: {bucket_name}.")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
            else:
```

```
                logger.error(f"Error deleting bucket: {error_code}")
                raise


    def display_buckets(self, source_bucket: str, dest_bucket: str):
        """
        Display a list of the objects in the test buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.list_bucket_contents(source_bucket)
        self.list_bucket_contents(dest_bucket)

    def list_bucket_contents(self, bucket_name):
        """
        Display a list of the objects in the bucket.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            print(f"\t Items in bucket {bucket_name}")
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            if not objs:
                print("\t\tNo objects found.")
            for obj in objs:
                key = obj["Key"]
                print(f"\t\t object: {key} ETag {obj['ETag']}")
            return objs
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist.")
            else:
                logger.error(f"Error listing bucket and objects: {error_code}")
                raise


    def display_menu(
        self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
    ):
        """
```

```
        Displays the menu of conditional request options for the user.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The key of the test object in the source bucket.
        :param etag: The etag of the test object in the source bucket.
        """

        actions = [
            "Print list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit.",
        ]

        conditions = [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ]

        condition_types = [
            "IfMatch",
            "IfNoneMatch",
            "IfModifiedSince",
            "IfUnmodifiedSince",
        ]
        copy_condition_types = [
            "CopySourceIfMatch",
            "CopySourceIfNoneMatch",
            "CopySourceIfModifiedSince",
            "CopySourceIfUnmodifiedSince",
        ]

        yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

        choice = 0
        while choice != 4:
            print("-" * 88)
```

```
            print("Choose an action to explore some example conditional
requests.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                print("Listing the objects and buckets.")
                self.display_buckets(source_bucket, dest_bucket)
            elif choice == 1:
                print("Perform a conditional read.")
                condition_type = q.choose("Enter the condition type : ",
conditions)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.get_object_conditional(
                        object_key, source_bucket,
condition_types[condition_type], etag
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.get_object_conditional(
                        object_key,
                        source_bucket,
                        condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 2:
                print("Perform a conditional copy.")
                condition_type = q.choose("Enter the condition type : ",
conditions)
                dest_key = q.ask("Enter an object key: ", q.non_empty)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.copy_object_conditional(
                        object_key,
                        dest_key,
                        source_bucket,
                        dest_bucket,
                        copy_condition_types[condition_type],
                        etag,
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.copy_object_conditional(
                        object_key,
                        dest_key,
                        copy_condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 3:
```

```
                print(
                    "Perform a conditional write using IfNoneMatch condition on
the object key."
                )
                print("If the key is a duplicate, the write will fail.")
                object_key = q.ask("Enter an object key: ", q.non_empty)
                self.conditional_requests.put_object_conditional(
                    object_key, source_bucket, b"Conditional write example data."
                )
            elif choice == 4:
                print("Proceeding to cleanup.")


    def run_scenario(self):
        """
        Runs the interactive scenario.
        """
        print("-" * 88)
        print("Welcome to the Amazon S3 conditional requests example.")
        print("-" * 88)

        print(
            f"""\
        This example demonstrates the use of conditional requests for S3
operations.
        You can use conditional requests to add preconditions to S3 read requests
to return or copy
        an object based on its Entity tag (ETag), or last modified date.
        You can use a conditional write requests to prevent overwrites by
ensuring
        there is no existing object with the same key.

        This example will allow you to perform conditional reads
        and writes that will succeed or fail based on your selected options.

        Sample buckets and a sample object will be created as part of the
example.
        """
        )

        bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
        source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
        dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
        object_key = "test-upload-file.txt"
```

```
        try:
            etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
 object_key)
            self.display_menu(source_bucket_name, dest_bucket_name, object_key,
 etag)
        finally:
            self.cleanup_scenario(source_bucket_name, dest_bucket_name)

        print("-" * 88)
        print("Thanks for watching.")
        print("-" * 88)


if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()
```

Uma classe de wrapper que define as operações de solicitação condicional.

```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)


class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

```
    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional write failed: Precondition failed")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise


    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
request.

        :param source_key: The key of the source object to copy.
```

```
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CopyObject
  - GetObject
  - PutObject

Gerencie listas de controle de acesso (ACLs) para buckets do Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como gerenciar listas de controle de acesso (ACLs) para buckets do Amazon S3.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
/// </summary>
public class ManageACLs
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket1";
        string newBucketName = "amzn-s3-demo-bucket2";
        string keyName = "sample-object.txt";
        string emailAddress = "someone@example.com";

        // If the AWS Region where your bucket is located is different from
        // the Region defined for the default user, pass the Amazon S3
bucket's
        // name to the client constructor. It should look like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
        IAmazonS3 client = new AmazonS3Client();
```

```
            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
            IAmazonS3 client,
            string bucketName,
            string newBucketName,
            string keyName,
            string emailAddress)
        {
            try
            {
                // Create a new Amazon S3 bucket and specify canned ACL.
                var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

                // Get the ACL on a bucket.
                await GetBucketACLAsync(client, bucketName);

                // Add (replace) the ACL on an object in a bucket.
```

```
                await AddACLToExistingObjectAsync(client, bucketName, keyName,
 emailAddress);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Exception: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL attached.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new Amazon S3 bucket.</param>
        /// <returns>Returns a boolean value indicating success or failure.</
returns>
        public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
 client, string newBucketName)
        {
            var request = new PutBucketRequest()
            {
                BucketName = newBucketName,
                BucketRegion = S3Region.EUWest1,

                // Add a canned ACL.
                CannedACL = S3CannedACL.LogDeliveryWrite,
            };

            var response = await client.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Retrieves the ACL associated with the Amazon S3 bucket name in the
        /// bucketName parameter.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket for which we want to
 get the
        /// ACL list.</param>
```

```
        /// <returns>Returns an S3AccessControlList returned from the call to
        /// GetACLAsync.</returns>
        public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
 client, string bucketName)
        {
            GetACLResponse response = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
            });

            return response.AccessControlList;
        }



        /// <summary>
        /// Adds a new ACL to an existing object in the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
 S3
        /// bucket containing the object to which we want to apply a new ACL.</
param>
        /// <param name="keyName">A string representing the name of the object
        /// to which we want to apply the new ACL.</param>
        /// <param name="emailAddress">The email address of the person to whom
        /// we will be applying to whom access will be granted.</param>
        public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
 string bucketName, string keyName, string emailAddress)
        {
            // Retrieve the ACL for an object.
            GetACLResponse aclResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
            });

            S3AccessControlList acl = aclResponse.AccessControlList;

            // Retrieve the owner.
            Owner owner = acl.Owner;
```

```
            // Clear existing grants.
            acl.Grants.Clear();

            // Add a grant to reset the owner's full permission
            // (the previous clear statement removed all permissions).
            var fullControlGrant = new S3Grant
            {
                Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
            };
            acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

            // Specify email to identify grantee for granting permissions.
            var grantUsingEmail = new S3Grant
            {
                Grantee = new S3Grantee { EmailAddress = emailAddress },
                Permission = S3Permission.WRITE_ACP,
            };

            // Specify log delivery group as grantee.
            var grantLogDeliveryGroup = new S3Grant
            {
                Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
                Permission = S3Permission.WRITE,
            };

            // Create a new ACL.
            var newAcl = new S3AccessControlList
            {
                Grants = new List<S3Grant> { grantUsingEmail,
 grantLogDeliveryGroup },
                Owner = owner,
            };

            // Set the new ACL. We're throwing away the response here.
            _ = await client.PutACLAsync(new PutACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
                AccessControlList = newAcl,
            });
        }

    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - GetBucketAcl
  - GetObjectAcl
  - PutBucketAcl
  - PutObjectAcl

Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um AWS SDK

O exemplo de código a seguir mostra como usar a biblioteca do cliente em versão ampliada do Amazon SQS para trabalhar com mensagens grandes do SQS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
```

```
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
```

```
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
```

```
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(),
e);
            throw e;
        }
    }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK for Java 2.x.
- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .

- CreateBucket
- PutBucketLifecycleConfiguration
- ReceiveMessage
- SendMessage

Gerencie objetos versionados do Amazon S3 em lotes com uma função Lambda usando um AWS SDK

O exemplo de código a seguir mostra como gerenciar objetos do S3 versionados em lotes com uma função do Lambda.

Python

SDK para Python (Boto3)

Mostra como manipular objetos versionados do Amazon Simple Storage Service (Amazon S3) em lotes criando trabalhos que chamam funções para realizar o processamento. AWS Lambda Este exemplo cria um bucket habilitado para versão, carrega as estrofes do poema Você é velho, padre William, de Lewis Carroll, e usa trabalhos em lote do Amazon S3 para distorcer o poema de várias maneiras.

Aprenda como:

- Criar funções do Lambda que operam em objetos versionados.
- Criar um manifesto de objetos para atualizar.
- Cria trabalhos em lote que invocam funções do Lambda para atualizar objetos.
- Excluir funções do Lambda.
- Esvaziar e excluir um bucket versionado.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon S3

Analise os URIs do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como usar analisar URIs do Amazon S3 para extrair componentes importantes como o nome do bucket e a chave do objeto.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Analise um URI do Amazon S3 usando a classe S3Uri.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

    /**
     *
     * @param s3Client    - An S3Client through which you acquire an S3Uri
 instance.
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate
 S3Uri
     *                    capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.
```

```
        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri()
method.
        URI uri = URI.create(s3ObjectUrl);
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.
```

```
        String partNumber =
 s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
 s3Uri.firstMatchingRawQueryParameters("partNumber");
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
 must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
 the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
```

Execute uma cópia em várias partes de um objeto do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como executar uma cópia multipart de um objeto do Amazon S3.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }
```

```
        /// <summary>
        /// This method uses the passed client object to perform a multipart
        /// copy operation.
        /// </summary>
        /// <param name="client">An Amazon S3 client object that will be used
        /// to perform the copy.</param>
        public static async Task MPUCopyObjectAsync(AmazonS3Client client)
        {
            // Create a list to store the copy part responses.
            var copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            var initiateRequest = new InitiateMultipartUploadRequest
            {
                BucketName = TargetBucket,
                Key = TargetObjectKey,
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            string uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                var metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = SourceBucket,
                    Key = SourceObjectKey,
                };

                GetObjectMetadataResponse metadataResponse =
                    await client.GetObjectMetadataAsync(metadataRequest);
                var objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
```

```
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    var copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = TargetBucket,
                        DestinationKey = TargetObjectKey,
                        SourceBucket = SourceBucket,
                        SourceKey = SourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i,
                    };

                    copyResponses.Add(await client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                var completeRequest = new CompleteMultipartUploadRequest
                {
                    BucketName = TargetBucket,
                    Key = TargetObjectKey,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{e.Message}' when writing an object");
            }
            catch (Exception e)
            {
                Console.WriteLine($"Unknown encountered on server.
Message:'{e.Message}' when writing an object");
            }
        }
```

```
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - GetObjectMetadata
  - UploadPartCopy

Receba e processe notificações de eventos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como trabalhar com notificações de eventos do S3 de uma forma orientada a objetos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo mostra como processar o evento de notificação do S3 usando o Amazon SQS.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload
and logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
notifications.
```

```
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
 API</a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
     * <groupId>software.amazon.awssdk</groupId>
     * <artifactId>s3-event-notifications</artifactId>
     * <version><LATEST></version>
     * </dependency>
     * <p>
     * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
     * <p>
     * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
     * in AWS SNS or AWS Lambda as well.
     * <p>
     * Note: The S3EventNotification class does not work with messages routed
 through AWS EventBridge.
     */
    static void processS3Events(String bucketName, String queueUrl, String
 queueArn) {
        try {
            // Configure the bucket to send Object Created and Object Tagging
 notifications to an existing SQS queue.
            s3Client.putBucketNotificationConfiguration(b -> b
                    .notificationConfiguration(ncb -> ncb
                            .queueConfigurations(qcb -> qcb
                                    .events(Event.S3_OBJECT_CREATED,
 Event.S3_OBJECT_TAGGING)
                                    .queueArn(queueArn)))
                            .bucket(bucketName)
            ).join();

            triggerS3EventNotifications(bucketName);
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {
```

```
                // Display the number of messages that are available in the
queue.
                sqsClient.getQueueAttributes(b -> b
                                .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                        ).thenAccept(attributeResponse ->
                                logger.info("Approximate number of messages in
the queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification
object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}",
eventName, key);
                        });
                    }
```

```
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {

sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Salve EXIF e outras informações da imagem usando um AWS SDK

O exemplo de código a seguir mostra como:

- Obter informações de EXIF de um arquivo JPG, JPEG ou PNG.
- Fazer upload do arquivo de imagem para um bucket do Amazon S3.
- Usar o Amazon Rekognition para identificar os três principais atributos (rótulos) no arquivo.
- Adicionar as informações de EXIF e rótulo a uma tabela do Amazon DynamoDB na região.

Rust

SDK para Rust

Obtenha informações de EXIF de um arquivo JPG, JPEG ou PNG, faça upload do arquivo de imagem para um bucket do Amazon S3, use o Amazon Rekognition para identificar os três principais atributos (rótulos no Amazon Rekognition) no arquivo e adicione as informações de EXIF e de rótulo a uma tabela do Amazon DynamoDB na região.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Envie notificações de eventos do S3 para a Amazon EventBridge usando um AWS SDK

O exemplo de código a seguir mostra como habilitar um bucket para enviar notificações de eventos do S3 EventBridge e rotear notificações para um tópico do Amazon SNS e uma fila do Amazon SQS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
     * to route the S3 object created events to a topic and a queue.
     *
     * @param bucketName Name of existing bucket
     * @param topicArn ARN of existing topic to receive S3 event notifications
```

```
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
                           logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                           throw new RuntimeException(t.getCause().getMessage(),
t);
                       }
                       logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
```

```
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Rastreie o upload ou o download de um objeto do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como usar o Amazon S3 para carregar ou baixar arquivos grandes.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Acompanhe o andamento do carregamento de um arquivo.

```
   public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                            String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
               .putObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .source(Paths.get(filePathURI))
               .build();

       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

       fileUpload.completionFuture().join();
       /*
           The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
           You can also implement the interface to provide your own logic.

           Configure log4J2 with settings such as the following.
               <Configuration status="WARN">
                   <Appenders>
                       <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                           <PatternLayout pattern="%m%n"/>
                       </Console>
                   </Appenders>

                   <Loggers>
                       <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
```

```
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file upload.
                Transfer initiated...
                |                    | 0.0%
                |====                | 21.1%
                |============        | 60.5%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

Acompanhe o andamento do download de um arquivo.

```
    public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())  // Add
listener.
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
```

```
                        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetObject
  - PutObject

Como transformar dados para sua aplicação com o S3 Object Lambda

O exemplo de código a seguir mostra como transformar dados para sua aplicação com o S3 Object Lambda.

.NET

SDK para .NET

Mostra como adicionar código personalizado a solicitações GET padrão do S3 para modificar o objeto solicitado e recuperado do S3 e possibilitar que o objeto atenda às necessidades do cliente ou aplicação solicitante.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Lambda
- Amazon S3

Exemplos de abordagens para testes unitários e de integração com um AWS SDK

O exemplo de código a seguir mostra exemplos de técnicas de melhores práticas ao escrever testes unitários e de integração usando um AWS SDK.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cargo.toml para exemplos de testes.

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"
```

```
[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-
credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
path = "src/main.rs"
```

Exemplo de teste de unidade usando automock e um wrapper de serviço.

```
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

#[cfg_attr(test, automock)]
```

```
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
```

```
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
```

```
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .set_next_continuation_token(Some("next".to_string()))
                    .build())
            });
        mock.expect_list_objects()
            .with(
                eq("test-bucket"),
                eq("test-prefix"),
                eq(Some("next".to_string())),
            )
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(3).build(),
                        s3::types::Object::builder().size(9).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
    }
}
```

Exemplo de teste de integração usando StaticReplayClient.

```
use aws_sdk_s3 as s3;

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
```

```
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}

#[cfg(test)]
mod test {
    use super::*;
```

```
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
 StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
```

```
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
                    .unwrap(),
            );
        let page_2 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
```

```
        replay_client.assert_requests_match(&[]);
    }
}
```

Fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service (Amazon S3)

O exemplo de código a seguir mostra como fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service (Amazon S3).

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use an S3TransferManagerpara fazer upload de um diretório local. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;

    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
 transferManager.uploadDirectory(UploadDirectoryRequest.builder()
```

```
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- Para obter detalhes da API, consulte UploadDirectorya Referência AWS SDK for Java 2.x da API.

Carregue ou baixe arquivos grandes de e para o Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como fazer upload ou download de arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte Carregar um objeto usando carregamento fracionado.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Chame funções que transferem arquivos de e para um bucket do S3 usando o Amazon TransferUtility S3.

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;
```

```
// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
 $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
 bucketName, fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
 {bucketName}.");
```

```
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
 bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
 {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
 keyName, localPath);
if (success)
{
    Console.WriteLine("$Successfully downloaded the file, {keyName} from
 {bucketName}.");
```

```
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
 bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
 {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
```

```
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
 Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
 an\n\t  S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
 directory.");
    Console.WriteLine($"\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
 textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}
```

```
// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
 s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

Carregue um único arquivo.

```
        /// <summary>
        /// Uploads a single file from the local computer to an S3 bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the file
        /// will be stored.</param>
        /// <param name="fileName">The name of the file to upload.</param>
        /// <param name="localPath">The local path where the file is stored.</
param>
```

```
        /// <returns>A boolean value indicating the success of the action.</
returns>
        public static async Task<bool> UploadSingleFileAsync(
            TransferUtility transferUtil,
            string bucketName,
            string fileName,
            string localPath)
        {
            if (File.Exists($"{localPath}\\{fileName}"))
            {
                try
                {
                    await transferUtil.UploadAsync(new
 TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        Key = fileName,
                        FilePath = $"{localPath}\\{fileName}",
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Could not upload {fileName} from
 {localPath} because:");
                    Console.WriteLine(s3Ex.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"{fileName} does not exist in {localPath}");
                return false;
            }
        }
```

Carregue um diretório local inteiro.

```
        /// <summary>
        /// Uploads all the files in a local directory to a directory in an S3
```

```
        /// bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the files
        /// will be stored.</param>
        /// <param name="keyPrefix">The key prefix is the S3 directory where
        /// the files will be stored.</param>
        /// <param name="localPath">The local directory that contains the files
        /// to be uploaded.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> UploadFullDirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string keyPrefix,
            string localPath)
        {
            if (Directory.Exists(localPath))
            {
                try
                {
                    await transferUtil.UploadDirectoryAsync(new
 TransferUtilityUploadDirectoryRequest
                    {
                        BucketName = bucketName,
                        KeyPrefix = keyPrefix,
                        Directory = localPath,
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Can't upload the contents of {localPath}
 because:");
                    Console.WriteLine(s3Ex?.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"The directory {localPath} does not exist.");
                return false;
```

```
            }
        }
```

Baixe um único arquivo.

```
        /// <summary>
        /// Download a single file from an S3 bucket to the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// file to download.</param>
        /// <param name="keyName">The name of the file to download.</param>
        /// <param name="localPath">The path on the local computer where the
        /// downloaded file will be saved.</param>
        /// <returns>A Boolean value indicating the results of the action.</
returns>
        public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
            string bucketName,
            string keyName,
            string localPath)
        {
            await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = $"{localPath}\\{keyName}",
            });

            return (File.Exists($"{localPath}\\{keyName}"));
        }
```

Baixe o conteúdo de um bucket do S3.

```
        /// <summary>
        /// Downloads the contents of a directory in an S3 bucket to a
```

```
        /// directory on the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The bucket containing the files to
 download.</param>
        /// <param name="s3Path">The S3 directory where the files are located.</
param>
        /// <param name="localPath">The local path to which the files will be
        /// saved.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> DownloadS3DirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string s3Path,
            string localPath)
        {
            int fileCount = 0;

            // If the directory doesn't exist, it will be created.
            if (Directory.Exists(s3Path))
            {
                var files = Directory.GetFiles(localPath);
                fileCount = files.Length;
            }

            await transferUtil.DownloadDirectoryAsync(new
 TransferUtilityDownloadDirectoryRequest
            {
                BucketName = bucketName,
                LocalDirectory = localPath,
                S3Directory = s3Path,
            });

            if (Directory.Exists(localPath))
            {
                var files = Directory.GetFiles(localPath);
                if (files.Length > fileCount)
                {
                    return true;
                }

                // No change in the number of files. Assume
```

```
                // the download failed.
                return false;
            }

            // The local directory doesn't exist. No files
            // were downloaded.
            return false;
        }
```

Acompanhe o progresso de um upload usando TransferUtility o.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Transfer;

    /// <summary>
    /// This example shows how to track the progress of a multipart upload
    /// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
    /// upload to an Amazon S3 bucket.
    /// </summary>
    public class TrackMPUUsingHighLevelAPI
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "sample_pic.png";
            string path = "filepath/directory/";
            string filePath = $"{path}{keyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            await TrackMPUAsync(client, bucketName, filePath, keyName);
        }

        /// <summary>
        /// Starts an Amazon S3 multipart upload and assigns an event handler to
```

```
        /// track the progress of the upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// perform the multipart upload.</param>
        /// <param name="bucketName">The name of the bucket to which to upload
        /// the file.</param>
        /// <param name="filePath">The path, including the file name of the
        /// file to be uploaded to the Amazon S3 bucket.</param>
        /// <param name="keyName">The file name to be used in the
        /// destination Amazon S3 bucket.</param>
        public static async Task TrackMPUAsync(
            IAmazonS3 client,
            string bucketName,
            string filePath,
            string keyName)
        {
            try
            {
                var fileTransferUtility = new TransferUtility(client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName,
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>(
                        UploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:: {ex.Message}");
            }
        }

        /// <summary>
```

```
        /// Event handler to check the progress of the multipart upload.
        /// </summary>
        /// <param name="sender">The object that raised the event.</param>
        /// <param name="e">The object that contains multipart upload
        /// information.</param>
        public static void UploadRequest_UploadPartProgressEvent(object sender,
 UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
        }
    }
```

Faça upload de um objeto com criptografia.

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Security.Cryptography;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
    /// perform a multipart upload to an Amazon S3 bucket.
    /// </summary>
    public class SSECLowLevelMPUcopyObject
    {
        public static async Task Main()
        {
            string existingBucketName = "amzn-s3-demo-bucket";
            string sourceKeyName = "sample_file.txt";
            string targetKeyName = "sample_file_copy.txt";
            string filePath = $"sample\\{targetKeyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USEast1.
            IAmazonS3 client = new AmazonS3Client();
```

```
            // Create the encryption key.
            var base64Key = CreateEncryptionKey();

            await CreateSampleObjUsingClientEncryptionKeyAsync(
                client,
                existingBucketName,
                sourceKeyName,
                filePath,
                base64Key);
        }

        /// <summary>
        /// Creates the encryption key to use with the multipart upload.
        /// </summary>
        /// <returns>A string containing the base64-encoded key for encrypting
        /// the multipart upload.</returns>
        public static string CreateEncryptionKey()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
            return base64Key;
        }

        /// <summary>
        /// Creates and uploads an object using a multipart upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object used to
        /// initialize and perform the multipart upload.</param>
        /// <param name="existingBucketName">The name of the bucket to which
        /// the object will be uploaded.</param>
        /// <param name="sourceKeyName">The source object name.</param>
        /// <param name="filePath">The location of the source object.</param>
        /// <param name="base64Key">The encryption key to use with the upload.</
param>
        public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
            IAmazonS3 client,
            string existingBucketName,
            string sourceKeyName,
            string filePath,
            string base64Key)
        {
```

```
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
               await client.InitiateMultipartUploadAsync(initiateRequest);

            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key,
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }
```

```
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine($"Exception occurred: {exception.Message}");

                // If there was an error, abort the multipart upload.
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };

                await client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que usam gerenciadores de upload e download para dividir os dados em partes e transferi-los simultaneamente.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
 string, objectKey string, largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
```

```
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}



// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
```

```
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

Execute um cenário interativo que mostre como usar os gerenciadores de upload e download em contexto.

```
import (
 "context"
 "crypto/rand"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use
 Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
```

```
// This package can be found in the ..\..\demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 large object demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
 bucket:",
  demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
 if err != nil {
  panic(err)
 }
 if !bucketExists {
  err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
  if err != nil {
   panic(err)
  } else {
   log.Println("Bucket created.")
  }
 }
 log.Println(strings.Repeat("-", 88))

 mibs := 30
 log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
 bucket. ", mibs)
 questioner.Ask("Press Enter when you're ready.")
 largeBytes := make([]byte, 1024*1024*mibs)
 _, _ = rand.Read(largeBytes)
```

```
 largeKey := "doc-example-large"
 log.Println("Uploading...")
 err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
 if err != nil {
  panic(err)
 }
 log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
 log.Println(strings.Repeat("-", 88))

 log.Printf("Let's download the %v MiB object.", mibs)
 questioner.Ask("Press Enter when you're ready.")
 log.Println("Downloading...")
 largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName,
 largeKey)
 if err != nil {
  panic(err)
 }
 log.Printf("Downloaded %v bytes.", len(largeDownload))
 log.Println(strings.Repeat("-", 88))

 if questioner.AskBool("Do you want to delete your bucket and all of its "+
  "contents? (y/n)", "y") {
  log.Println("Deleting object.")
  err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
  if err != nil {
   panic(err)
  }
  log.Println("Deleting bucket.")
  err = bucketBasics.DeleteBucket(ctx, bucketName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Chame funções que transferem arquivos de e para um bucket do S3 usando o. S3TransferManager

```
   public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
           URI destinationPathURI, String bucketName) {
       DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
               .destination(Paths.get(destinationPathURI))
               .bucket(bucketName)
               .build());
       CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

       completedDirectoryDownload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryDownload.failedTransfers().size();
   }
```

Carregue um diretório local inteiro.

```
   public Integer uploadDirectory(S3TransferManager transferManager,
           URI sourceDirectory, String bucketName) {
       DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
               .source(Paths.get(sourceDirectory))
               .bucket(bucketName)
               .build());

       CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
```

```
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
 fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

Carregue um único arquivo.

```
    public String uploadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Os exemplos de código usam as importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
```

```
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Use o S3 Transfer Manager na parte superior do cliente AWS CRT-based S3 para realizar de forma transparente um upload de várias partes quando o tamanho do conteúdo exceder um limite. O limite padrão é 8 MB.

```
    /**
     * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the file path of the file to be uploaded
     */
    public void multipartUploadWithTransferManager(String bucketName, String key,
 String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b
                .bucket(bucketName)
                .key(key))
            .source(Paths.get(filePath))
            .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

Use a API S3Client para realizar um carregamento fracionado.

```
    /**
```

```
     * Performs a multipart upload to Amazon S3 using the provided S3 client.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the path to the file to be uploaded
     */
    public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
```

```
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

Use a S3AsyncClient API com suporte a várias partes ativado para realizar um upload de várias partes.

```
    /**
     * Uploads a file to an S3 bucket using the S3AsyncClient and enabling
multipart support.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the local file path of the file to be uploaded
     */
    public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
            .multipartEnabled(true)
            .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
                .bucket(bucketName)
```

```
                .key(key),
            Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using
 S3AsyncClient.");
    }
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de um arquivo grande.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
  ProgressBar,
  logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
```

```
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
        Bucket: bucketName,
        Key: key,
        Body: buffer,
      },
    });

    upload.on("httpUploadProgress", ({ loaded, total }) => {
      progressBar.update({ current: loaded, total });
    });

    await upload.done();
  } catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
      logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Baixe um arquivo grande.

```
import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
```

```
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
```

```
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que transfiram arquivos usando várias configurações do gerenciador de transferências disponíveis. Use uma classe de retorno de chamada para gravar o andamento do retorno de chamada durante a transferência de arquivos.

```
import sys
```

```
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
```

```
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
```

```
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
```

```
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

Demonstre as funções do gerenciador de transferências e relate resultados.

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
```

```
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")


class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
 files? "
            )
```

```
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
```

```
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
                self.demo_bucket,
                object_key,
                self.file_size_mb,
                **upload_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Downloading", object_key, download_file_path, **download_args
            )
            start_time = time.perf_counter()
            thread_info = download_func(
                self.demo_bucket,
                object_key,
                download_file_path,
                self.file_size_mb,
                **download_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
```

```
        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
            self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported."
            )

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.
```

```
        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1

        local_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, local_suffix)
        )

        s3_object_key = file_name_template.format(file_tag, object_suffix)

        downloaded_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag,
download_suffix)
        )

        filled_cmd = self._create_file_cmd.format(
            local_file_path, self.file_size_mb * self._size_multiplier
        )

        print(
            f"Creating file of size {self.file_size_mb} MB "
            f"in {self.demo_folder} by running:"
        )
        print(f"{'':4}{filled_cmd}")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print("-" * self._terminal_width)
        print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
        if kwargs:
            print("With extra args:")
            for arg, value in kwargs.items():
                print(f'{"":4}{arg:<20}: {value}')
```

```
    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
            print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")


def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
```

```
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )
```

```
    # Download without specifying an encryption key to show that the
    # encryption key must be included to download an encrypted object.
    if demo_manager.ask_user(
        "Do you want to try to download the encrypted "
        "object without sending the required key?"
    ):
        try:
            _, object_key, download_file_path = demo_manager.last_name_set()
            file_transfer.download_with_default_configuration(
                demo_manager.demo_bucket,
                object_key,
                download_file_path,
                demo_manager.file_size_mb,
            )
        except ClientError as err:
            print(
                "Got expected error when trying to download an encrypted "
                "object without specifying encryption info:"
            )
            print(f"{'':4}{err}")

    # Remove all created and downloaded files, remove all objects from
    # S3 storage.
    if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files " "and S3
 objects?"
    ):
        demo_manager.cleanup()


if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
 get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}
```

```
async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
```

```
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }

    if file_size == 0 {
        return Err(S3ExampleError::new("Bad file size."));
    }
    if chunk_count > MAX_CHUNKS {
        return Err(S3ExampleError::new(
            "Too many chunks! Try increasing your chunk size.",
        ));
    }

    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
```

```
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }

    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;

    let data: GetObjectOutput =
        s3_code_examples::download_object(&client, &bucket_name, &key).await?;
    let data_length: u64 = data
        .content_length()
        .unwrap_or_default()
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_code_examples::clear_bucket(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_code_examples::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");
```

```
    Ok(())
}
```

Faça upload de um stream de tamanho desconhecido para um objeto do Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como fazer upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o cliente AWS CRT-based S3.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsync {
```

```
   private static final Logger logger =
LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

   public static void main(String[] args) {
       String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
       String key = UUID.randomUUID().toString();

       AsyncExampleUtils.createBucket(bucketName);
       try {
           PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
           S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
           PutObjectResponse putObjectResponse =
example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
           logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
           logger.info("Object {} uploaded to bucket {}.", key, bucketName);
       } catch (SdkException e) {
           logger.error(e.getMessage(), e);
       } finally {
           AsyncExampleUtils.deleteObject(bucketName, key);
           AsyncExampleUtils.deleteBucket(bucketName);
       }
   }

   /**
    * @param s33CrtAsyncClient - To upload content from a stream of unknown
size, use can the AWS CRT-based S3 client.
    * @param bucketName - The name of the bucket.
    * @param key - The name of the object.
    * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
Returns metadata pertaining to the put object operation.
    */
   public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient
s33CrtAsyncClient, String bucketName, String key) {

       // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
       String randomString = AsyncExampleUtils.randomString();
       logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
       InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());
```

```
        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
 null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Use o [cliente do S3 assíncrono padrão com o upload de várias partes ativado](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {
    private static final Logger logger =
 LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);
```

```
    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsyncMp example = new
PutObjectFromStreamAsyncMp();
            S3AsyncClient s3AsyncClientMp =
S3AsyncClient.builder().multipartEnabled(true).build();
            PutObjectResponse putObjectResponse =
example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3AsyncClientMp - To upload content from a stream of unknown size,
use can the S3 asynchronous client with multipart enabled.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
Returns metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
```

```
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
 null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key),
 body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Use o Gerenciador de transferências do Amazon S3.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
 LoggerFactory.getLogger(UploadStream.class);

    public static void main(String[] args) {
```

```
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param transferManager - To upload content from a stream of unknown size,
you can use the S3TransferManager based on the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
length.
```

```
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
 null, executor);

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        CompletedUpload completedUpload = upload.completionFuture().join();
        executor.shutdown();
        return completedUpload;
    }
}
```

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams


    /// Upload a file to the specified bucket.
    ///
    /// - Parameters:
    ///   - bucket: The Amazon S3 bucket name to store the file into.
    ///   - key: The name (or path) of the file to upload to in the `bucket`.
    ///   - sourcePath: The pathname on the local filesystem of the file to
    ///     upload.
```

```
    func uploadFile(sourcePath: String, bucket: String, key: String?) async
throws {
        let fileURL: URL = URL(fileURLWithPath: sourcePath)
        let fileName: String

        // If no key was provided, use the last component of the filename.

        if key == nil {
            fileName = fileURL.lastPathComponent
        } else {
            fileName = key!
        }

        let s3Client = try await S3Client()

        // Create a FileHandle for the source file.

        let fileHandle = FileHandle(forReadingAtPath: sourcePath)
        guard let fileHandle = fileHandle else {
            throw TransferError.readError
        }

        // Create a byte stream to retrieve the file's contents. This uses the
        // Smithy FileStream and ByteStream types.

        let stream = FileStream(fileHandle: fileHandle)
        let body = ByteStream.stream(stream)

        // Create a `PutObjectInput` with the ByteStream as the body of the
        // request's data. The AWS SDK for Swift will handle sending the
        // entire file in chunks, regardless of its size.

        let putInput = PutObjectInput(
            body: body,
            bucket: bucket,
            key: fileName
        )

        do {
            _ = try await s3Client.putObject(input: putInput)
        } catch {
            throw TransferError.uploadError("Error uploading the file: \(error)")
        }
```

```
        print("File uploaded to \(fileURL.path).")
    }
```

Use somas de verificação para trabalhar com um objeto do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como usar somas de verificação para trabalhar com um objeto do Amazon S3.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Os exemplos de código usam um subconjunto das importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
```

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Especifique um algoritmo de soma de verificação para o método `putObject` ao criar `PutObjectRequest`.

```
    public void putObjectWithChecksum(String bucketName, String key) {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
            RequestBody.fromString("This is a test"));
    }
```

Verifique a soma de verificação do `getObject` método ao criar o. GetObjectRequest

```
    public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
 {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
            .response();
    }
```

Pre-calculate uma soma de verificação para o `putObject` método quando você cria o. `PutObjectRequest`

```
   public void putObjectWithPrecalculatedChecksum(String bucketName, String key,
String filePath) {
       String checksum = calculateChecksum(filePath, "SHA-256");

       s3Client.putObject((b -> b
               .bucket(bucketName)
               .key(key)
               .checksumSHA256(checksum)),
           RequestBody.fromFile(Paths.get(filePath)));
   }
```

Use o S3 Transfer Manager na parte superior do cliente AWS CRT-based S3 para realizar de forma transparente um upload de várias partes quando o tamanho do conteúdo exceder um limite. O limite padrão é 8 MB.

Você pode especificar um algoritmo de soma de verificação para o SDK usar. Por padrão, o SDK usa o algoritmo CRC32.

```
   public void multipartUploadWithChecksumTm(String bucketName, String key,
String filePath) {
       S3TransferManager transferManager = S3TransferManager.create();
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
           .putObjectRequest(b -> b
               .bucket(bucketName)
               .key(key)
               .checksumAlgorithm(ChecksumAlgorithm.SHA1))
           .source(Paths.get(filePath))
           .build();
       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
       fileUpload.completionFuture().join();
       transferManager.close();
   }
```

Use a API ou (S3AsyncClient API) do S3Client para realizar um upload de várias partes. Se você especificar uma soma de verificação adicional, deverá indicar o algoritmo a ser usado no

início do carregamento. Você também deve especificar o algoritmo para a solicitação de cada parte e fornecer a soma de verificação calculada para cada parte após o carregamento.

```
   public void multipartUploadWithChecksumS3Client(String bucketName, String
key, String filePath) {
       ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

       // Initiate the multipart upload.
       CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
           .bucket(bucketName)
           .key(key)
           .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
       String uploadId = createMultipartUploadResponse.uploadId();

       // Upload the parts of the file.
       int partNumber = 1;
       List<CompletedPart> completedParts = new ArrayList<>();
       ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

       try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
           long fileSize = file.length();
           long position = 0;
           while (position < fileSize) {
               file.seek(position);
               long read = file.getChannel().read(bb);

               bb.flip(); // Swap position and limit before reading from the
buffer.
               UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                   .bucket(bucketName)
                   .key(key)
                   .uploadId(uploadId)
                   .checksumAlgorithm(algorithm) // Checksum specified on each
part.
                   .partNumber(partNumber)
                   .build();

               UploadPartResponse partResponse = s3Client.uploadPart(
                   uploadPartRequest,
                   RequestBody.fromByteBuffer(bb));

               CompletedPart part = CompletedPart.builder()
```

```
                    .partNumber(partNumber)
                    .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

Trabalhe com os recursos de integridade de objetos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como trabalhar com os recursos de integridade de objetos do S3.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstra os recursos de integridade do Amazon S3.

```
//! Routine which runs the S3 object integrity workflow.
/*!
   \param clientConfig: Aws client configuration.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
        const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
 std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
 std::endl;
        return false;
    }
```

```
   Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
   std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

   printAsterisksLine();
   std::cout << "Choose from one of the following checksum algorithms."
             << std::endl;

   for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
       std::cout << "  " << hashMethod << " - " <<
stringForHashMethod(hashMethod)
                 << std::endl;
   }

   HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                                         SHA256);


   gUseCalculatedChecksum = !askYesNoQuestion(
           "Let the SDK calculate the checksum for you? (y/n) ");

   printAsterisksLine();

   std::cout << "The workflow will now upload a file using PutObject."
             << std::endl;
   std::cout << "Object integrity will be verified using the "
             << stringForHashMethod(chosenHashMethod) << " algorithm."
             << std::endl;
   if (gUseCalculatedChecksum) {
       std::cout
               << "A checksum computed by this workflow will be used for object
integrity verification,"
               << std::endl;
       std::cout << "except for the TransferManager upload." << std::endl;
   } else {
       std::cout
               << "A checksum computed by the SDK will be used for object
integrity verification."
               << std::endl;
   }

   pressEnterToContinue();
   printAsterisksLine();
```

```
    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for
PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                           inputData, chosenHashMethod == DEFAULT,
                           *client)) {
        std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
```

```
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                            putObjectHashMethod, retrievedHash,
                            nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                         "PutObject upload", putObjectHashMethod);


    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32;  // The default hash method for the
TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "
                  << stringForHashMethod(transferManagerHashMethod)
                  << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
chosenHashMethod == DEFAULT,
                                 client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
```

```
   std::vector<Aws::String> retrievedTransferManagerPartHashes;
   Aws::String retrievedTransferManagerFinalHash;

   // Retrieve all the hashes for the TransferManager upload.
   if (!retrieveObjectHash(bucketName, key,
                           transferManagerHashMethod,
                           retrievedTransferManagerFinalHash,
                           &retrievedTransferManagerPartHashes, *client)) {
       std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   AwsDoc::S3::Hasher locallyCalculatedFinalHash;
   std::vector<Aws::String> locallyCalculatedPartHashes;

   // Calculate the hashes locally to demonstrate how TransferManager hashes are
computed.
   if (!calculatePartHashesForFile(transferManagerHashMethod,
MULTI_PART_TEST_FILE,
                                   UPLOAD_BUFFER_SIZE,
                                   locallyCalculatedFinalHash,
                                   locallyCalculatedPartHashes)) {
       std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   verifyHashingResults(retrievedTransferManagerFinalHash,
                        locallyCalculatedFinalHash, "TransferManager upload",
                        transferManagerHashMethod,
                        retrievedTransferManagerPartHashes,
                        locallyCalculatedPartHashes);

   printAsterisksLine();

   key = "mp_";
   key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

   multiPartUploadExplanations(key, chosenHashMethod);
```

```
    pressEnterToContinue();

    std::shared_ptr<Aws::IOStream> largeFileInputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          MULTI_PART_TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!largeFileInputData->good()) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
    if (multipartUploadHashMethod == DEFAULT) {
        multipartUploadHashMethod = MD5;  // The default hash method for
multipart uploads is MD5.

        std::cout << "The default checksum algorithm for multipart upload is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    AwsDoc::S3::Hasher hashData;
    std::vector<Aws::String> partHashes;

    if (!doMultipartUpload(bucketName, key,
                           multipartUploadHashMethod,
                           largeFileInputData, chosenHashMethod == DEFAULT,
                           hashData,
                           partHashes,
                           *client)) {
        std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::cout << "Finished multipart upload of with hash method " <<
              stringForHashMethod(multipartUploadHashMethod) << std::endl;

    std::cout << "Now we will retrieve the checksums from the server." <<
std::endl;
```

```
    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                            multipartUploadHashMethod,
                            retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
 multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                         multipartUploadHashMethod,
                         retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
 workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
 std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object
 integrity hashing methods.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashData: The hash value that will be associated with the uploaded
 object.
   \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
   \param body: The data content of the object being uploaded.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
```

```
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
 &key,
                                   const Aws::String &hashData,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &body,
                                   bool useDefaultHashMethod,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {

 request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {
        std::cout << "Object successfully uploaded." << std::endl;
```

```
    } else {
        std::cerr << "Error uploading object." <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}


// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of
 the object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
```

```
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
```

```
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
 result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
 NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object
 parts." <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}
```

```
//! Verifies the hashing results between the retrieved and local hashes.
/*!
 \param retrievedHash The hash value retrieved from the remote source.
 \param localHash The hash value calculated locally.
 \param uploadtype The type of upload (e.g., "multipart", "single-part").
 \param hashMethod The hashing method used (e.g., MD5, SHA-256).
 \param retrievedPartHashes (Optional) The list of hashes for the individual
 parts retrieved from the remote source.
 \param localPartHashes (Optional) The list of hashes for the individual parts
 calculated locally.
 */
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                      const Hasher &localHash,
                                      const Aws::String &uploadtype,
                                      HASH_METHOD hashMethod,
                                      const std::vector<Aws::String>
 &retrievedPartHashes,
                                      const std::vector<Aws::String>
 &localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
 << std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
 retrieved."
                  << std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << "  Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
 std::endl;
```

```
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do
 not match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
 "'"
                      << std::endl;
            std::cerr << "Remote number of hashes - '"
                      << retrievedPartHashes.size()
                      << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not
 match for part " << i + 1
                          << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                          << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                          << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
 hashes all match!" << std::endl;
    }

}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &,
                                         const
 Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
```

```
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" <<
 std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                          const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
 std::endl;
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
 Transfer Manager.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    bool useDefaultHashMethod,
                                    const std::shared_ptr<Aws::S3::S3Client>
 &client) {
    std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = client;
    transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
    if (!useDefaultHashMethod) {
        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
            transfer_config.checksumAlgorithm =
 getChecksumAlgorithmForHashMethod(
```

```
                    hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback =
 transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
 Aws::Transfer::TransferManager::Create(
            transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
 transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

              bucket, key,

              "text/plain",

              Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
            uploadHandle->GetStatus() ==
 Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
>GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being
 uploaded to an S3 bucket.
/*!
   \param hashMethod: The hashing algorithm to use when calculating the hash
 values.
   \param fileName: The path to the file for which the part hashes will be
 calculated.
   \param bufferSize: The size of the buffer to use when reading the file.
   \param[out] hashDataResult: The Hasher object that will store the concatenated
 hash value.
```

```
   \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                            const Aws::String &fileName,
                                            size_t bufferSize,
                                            AwsDoc::S3::Hasher &hashDataResult,
                                            std::vector<Aws::String> &partHashes)
 {
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;


    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
 static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
 uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
 preallocatedStreamBuf(buffer.data(),

 bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
                Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                               &preallocatedStreamBuf);
        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
 hashBuffer.GetUnderlyingData(),
```

```
                                hashBuffer.GetUnderlyingData() +
 hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
 &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}
```

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
 checksum algorithm,
                            ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
 &bucket,
                                                          const Aws::String &key,
                                                          const Aws::String
 &uploadID,
                                                          int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                          const Aws::String
 &calculatedHash,
                                                          const
 std::shared_ptr<Aws::IOStream> &body,
                                                          const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
```

```
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

  const Aws::String &key,

  const Aws::String &uploadID,

  const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

//! Routine which performs a multi-part upload.
```

```
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default
 hash method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the
 concatenated hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream>
 &ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {
    // Get object size.
    ioStream->seekg(0, ioStream->end);
    size_t objectSize = ioStream->tellg();
    ioStream->seekg(0, ioStream->beg);

    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
        }
    }
    Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
 client);
    if (uploadID.empty()) {
        return false;
    }

    std::vector<unsigned char> totalHashBuffer;
    bool uploadSucceeded = true;
```

```
   std::streamsize uploadedBytes = 0;
   int partNumber = 1;
   Aws::Vector<Aws::S3::Model::CompletedPart> parts;
   while (uploadedBytes < objectSize) {
       std::cout << "Uploading part " << partNumber << "." << std::endl;

       std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
       std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),

objectSize - uploadedBytes));
       ioStream->read((char *) buffer.data(), bytesToRead);
       Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
       std::shared_ptr<Aws::IOStream> body =
               Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                              &preallocatedStreamBuf);

       Hasher hasher;
       if (!hasher.calculateObjectHash(*body, hashMethod)) {
           std::cerr << "Error calculating hash." << std::endl;
           uploadSucceeded = false;
           break;
       }

       Aws::String base64HashString = hasher.getBase64HashString();
       partHashes.push_back(base64HashString);

       Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

       totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                              hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

       Aws::String calculatedHash;
       if (gUseCalculatedChecksum) {
           calculatedHash = base64HashString;
       }
       Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,
```

```
checksumAlgorithm, base64HashString, body,
                                                                    client);
       if (uploadPartOutcome.IsSuccess()) {
           const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
           Aws::S3::Model::CompletedPart completedPart;
           completedPart.SetETag(uploadPartResult.GetETag());
           completedPart.SetPartNumber(partNumber);
           switch (hashMethod) {
               case AwsDoc::S3::MD5:
                   break; // Do nothing.
               case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                   break;
               case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                   break;
               case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                   break;
               case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                   break;
               default:
                   std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                   break;
           }

           parts.push_back(completedPart);
       } else {
           std::cerr << "Error uploading part. " <<
                     uploadPartOutcome.GetError().GetMessage() << std::endl;
           uploadSucceeded = false;
           break;
       }

       uploadedBytes += bytesToRead;
       partNumber++;
```

```
    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
 completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
 {
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
        } else {
            std::cerr << "Error completing multipart upload." <<
                      completeMultipartUploadOutcome.GetError().GetMessage()
                      << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            return "Default";
```

```
        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
 {
    Aws::S3::Model::ChecksumAlgorithm result =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
 valid." << std::endl;
            break;  // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
```

```
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;

    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                         const Aws::S3::S3ClientConfiguration
 &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                               clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
  \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
            << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
 integrity workflow."
            << std::endl;
    printAsterisksLine();
```

```
    std::cout
            << "This workflow demonstrates how Amazon S3 uses checksum values to
 verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
            << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
            << "By default it calculates a checksum that is uploaded with an
 object.\n"
            << "The default checksum algorithm for PutObject and MultiPart upload
 is an MD5 hash.\n"
            << "The default checksum algorithm for TransferManager uploads is a
 CRC32 checksum."
            << std::endl;
    std::cout
            << "You can override the default behavior, requiring one of the
 following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
 the SDK calculate the value."
              << std::endl;
    std::cout
            << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
            << std::endl;

    std::cout
            << "This workflow will locally compute checksums for files uploaded
 to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
            << "This is done to provide demonstration code for how the checksums
 are calculated."
            << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
 object uploads."
              << std::endl;
}

//! Console interaction which explains the PutObject results.
/*!
*/
void AwsDoc::S3::explainPutObjectResults() {
```

```
    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have
 failed."
              << std::endl;
    std::cout
            << "The checksums calculated by the server have been retrieved using
 the GetObjectAttributes."
            << std::endl;
    std::cout
            << "The locally calculated checksums have been verified against the
 retrieved checksums."
            << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
  \param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
        const Aws::String &objectKey) {
    std::cout
            << "Now the workflow will demonstrate object integrity for
 TransferManager multi-part uploads."
            << std::endl;
    std::cout
            << "The AWS C++ SDK has a TransferManager class which simplifies
 multipart uploads."
            << std::endl;
    std::cout
            << "The following code lets the TransferManager handle much of the
 checksum configuration."
            << std::endl;

    std::cout << "An object with the key '" << objectKey
              << " will be uploaded by the TransferManager using a "
              << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
 calculate the hash value."
                  << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
```

```
}

//! Console interaction explaining multi-part uploads.
/*!
  \param objectKey: The key for the object being uploaded.
  \param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                             HASH_METHOD chosenHashMethod) {
    std::cout
            << "Now we will provide an in-depth demonstration of multi-part
 uploading by calling the multi-part upload APIs directly."
            << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when
 uploading large files."
              << std::endl;
    std::cout
            << "In the following code, the checksums are also calculated locally
 and then compared."
            << std::endl;
    std::cout
            << "For multi-part uploads, a checksum is uploaded with each part.
 The final checksum is a concatenation of"
            << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
            << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,\""
            << " in the section \"Using part-level checksums for multipart
 uploads\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
              stringForHashMethod(chosenHashMethod) << " uploading to with object
 key\n"
              << "'" << objectKey << "'," << std::endl;

}

//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
 file.
```

```
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                                | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
 MULTI_PART_TEST_FILE <<
                  std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in

                                  | std::ios::binary);
    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
                  std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();

    return true;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - AbortMultipartUpload

- CompleteMultipartUpload
- CreateMultipartUpload
- DeleteObject
- GetObjectAttributes
- PutObject
- UploadPart

Trabalhe com objetos versionados do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um bucket do S3 versionado.
- Obter todas as versões de um objeto.
- Reverter um objeto para uma versão anterior.
- Excluir e restaurar um objeto versionado.
- Excluir, permanentemente, todas as versões de um objeto.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que envolvam ações do S3.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
```

```
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
```

```
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket



def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
```

```
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )



def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
```

```
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)



def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
```

```
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

Carregue uma estrofe de poema para um objeto versionado e realize uma série de ações nela.

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
    print(
        "The latest version of the stanza is now:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
```

```
        sep="\n",
    )

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions =
 bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[
            f"    {version.version_id}, last modified {version.last_modified}"
            for version in obj_stanza_1_versions
        ],
        sep="\n",
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print(
        "The latest version of the stanza:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
    try:
        obj_stanza_1.get()
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            print("The stanza is now deleted (as expected).")
        else:
            raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )
```

```
    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateBucket
  - DeleteObject
  - ListObjectVersions
  - PutBucketLifecycleConfiguration

Exemplos de tecnologia sem servidor para o Amazon S3

Os exemplos de código a seguir mostram como usar o Amazon S3 com AWS SDKs.

Exemplos

- Invocar uma função do Lambda em um acionador do Amazon S3

Invocar uma função do Lambda em um acionador do Amazon S3

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto em um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
```

```
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request -
 {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);
```

```
            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do S3 com o uso do JavaScript Lambda.

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g,
 ' '));
```

```
    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Consumindo um evento do S3 com o uso do TypeScript Lambda.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
```

```
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do S3 com o Lambda usando PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
```

```
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
 $bucket . '. Make sure they exist and your bucket is in the same region as this
 function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
```

```
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
 your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do S3 com o Lambda usando Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'
```

```
def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
```

```
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
 name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;
```

```
    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

Exemplos de código para o Amazon S3 Control usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon S3 Control com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- Guia do usuário do Amazon S3 Control: mais informações sobre o Amazon S3 Control.
- Referência de API do Amazon S3 Control: detalhes sobre todas as ações disponíveis do Amazon S3 Control.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon S3 Control usando AWS SDKs
  - Olá, Amazon S3 Control
  - Aprenda as noções básicas do Amazon S3 Control com um AWS SDK
  - Ações para o Amazon S3 Control usando AWS SDKs

- Use CreateJobcom um AWS SDK ou CLI
- Use DeleteJobTaggingcom um AWS SDK
- Use DescribeJobcom um AWS SDK ou CLI
- Use GetJobTaggingcom um AWS SDK
- Use PutJobTaggingcom um AWS SDK
- Use UpdateJobPrioritycom um AWS SDK ou CLI
- Use UpdateJobStatuscom um AWS SDK ou CLI

Exemplos básicos para o Amazon S3 Control usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon S3 Control com AWS SDKs.

Exemplos

- Olá, Amazon S3 Control
- Aprenda as noções básicas do Amazon S3 Control com um AWS SDK
- Ações para o Amazon S3 Control usando AWS SDKs
  - Use CreateJobcom um AWS SDK ou CLI
  - Use DeleteJobTaggingcom um AWS SDK
  - Use DescribeJobcom um AWS SDK ou CLI
  - Use GetJobTaggingcom um AWS SDK
  - Use PutJobTaggingcom um AWS SDK
  - Use UpdateJobPrioritycom um AWS SDK ou CLI
  - Use UpdateJobStatuscom um AWS SDK ou CLI

Olá, Amazon S3 Control

Os exemplos de código a seguir mostram como começar a usar o Amazon S3 Control.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 not configured
 * authentication for SDKs and tools,see https://docs.aws.amazon.com/sdkref/
latest/guide/access.html in the AWS SDKs and Tools Reference Guide.
 * <p/>
 * You must have a runtime environment configured with the Java SDK.
 * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
 in the Developer Guide if this is not set up.
 */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;
```

```
    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
```

```
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation
is finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)
            .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
```

```
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
```

- Para obter detalhes da API, consulte ListJobsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def list_jobs(self, account_id: str) -> None:
        """
        List all batch jobs for the account.

        Args:
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.list_jobs(
                AccountId=account_id,
                JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
            )
            jobs = response.get('Jobs', [])
            for job in jobs:
                print(f"The job id is {job['JobId']}")
                print(f"The job priority is {job['Priority']}")
        except ClientError as e:
            print(f"Error listing jobs: {e}")
            raise
```

- Para obter detalhes da API, consulte a ListJobsReferência da API AWS SDK for Python (Boto3).

Aprenda as noções básicas do Amazon S3 Control com um AWS SDK

Os exemplos de código a seguir mostram como aprender as principais operações do Amazon S3 Control.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Conheça as principais operações.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;

public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing
of large-scale
            data stored in Amazon S3. It automatically scales resources to handle
varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove
tags on thousands or millions of
            objects in a single operation, streamlining the management and
organization of their data.

            This can be particularly useful for tasks such as cost allocation,
lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
essential.
            S3 Batch's scalability and serverless nature make it an ideal
solution for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this
scenario.
        System.out.println("Use CloudFormation to stand up the resource required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
```

```
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
 with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
 manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
 priority");
        System.out.println("""
             In this step, we modify the job priority value. The higher the
 number, the higher the priority.
             So, a job with a priority of `30` would have a higher priority than
 a job with
```

```
            a priority of `20`. This is a common way to represent the priority
of a task
            or job, with higher numbers indicating a higher priority.

            Ensure that the job status allows for priority updates. Jobs in
certain
            states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
            updated. Only jobs in the Active or Suspended state typically allow
priority
            updates.
            """);

       try {
           actions.updateJobPriorityAsync(jobId, accountId)
               .exceptionally(ex -> {
                   System.err.println("Update job priority failed: " +
ex.getMessage());
                   return null;
               })
               .join();
       } catch (CompletionException ex) {
           System.err.println("Failed to update job priority: " +
ex.getMessage());
       }
       waitForInputToContinue(scanner);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("3. Cancel the S3 Batch job");
       System.out.print("Do you want to cancel the Batch job? (y/n): ");
       String cancelAns = scanner.nextLine();
       if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
           try {
               actions.cancelJobAsync(jobId, accountId)
                   .exceptionally(ex -> {
                       System.err.println("Cancel job failed: " +
ex.getMessage());
                       return null;
                   })
                   .join();
           } catch (CompletionException ex) {
               System.err.println("Failed to cancel job: " + ex.getMessage());
           }
```

```
        } else {
            System.out.println("Job " +jobId +" was not canceled.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Describe the job that was just created");
        waitForInputToContinue(scanner);
        try {
            actions.describeJobAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Describe job failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to describe job: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the tags associated with the job");
        waitForInputToContinue(scanner);
        try {
            actions.getJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Get job tags failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to get job tags: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update Batch Job Tags");
        waitForInputToContinue(scanner);
        try {
            actions.putJobTaggingAsync(jobId, accountId)
                .exceptionally(ex -> {
```

```
                    System.err.println("Put job tagging failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to put job tagging: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the Amazon S3 Batch job tagging.");
        System.out.print("Do you want to delete Batch job tagging? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.deleteBatchJobTagsAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
            }
        } else {
            System.out.println("Tagging was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
```

```
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
 completed.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println();
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println();
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }


}
```

Uma classe de ação que envolve operações.

```
public class S3BatchActions {

    private static S3ControlAsyncClient asyncClient;

    private static S3AsyncClient s3AsyncClient ;
    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
```

```
    * has not been created yet, it will be initialized with the following
configuration:
    * <ul>
    *   <li>Maximum concurrency: 100</li>
    *   <li>Connection timeout: 60 seconds</li>
    *   <li>Read timeout: 60 seconds</li>
    *   <li>Write timeout: 60 seconds</li>
    *   <li>API call timeout: 2 minutes</li>
    *   <li>API call attempt timeout: 90 seconds</li>
    *   <li>Retry policy: 3 retries</li>
    *   <li>Region: US_EAST_1</li>
    *   <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
    * </ul>
    *
    * @return the asynchronous S3 Control client instance
    */
   private static S3ControlAsyncClient getAsyncClient() {
       if (asyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryPolicy(RetryPolicy.builder()
                   .numRetries(3)
                   .build())
               .build();

           asyncClient = S3ControlAsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return asyncClient;
   }
```

```
    private static S3AsyncClient getS3AsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
{
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();
```

```
        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error
has occurred
     */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
                return null; // Return null to handle the exception
            });
```

```
        return future;
    }

    /**
     * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
     *
     * @param jobId     the ID of the job for which to retrieve the tags
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
     * @throws RuntimeException if an error occurs while retrieving the job tags
     */
    public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
        GetJobTaggingRequest request = GetJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Asynchronously deletes the tags associated with a specific batch job.
     *
     * @param jobId     The ID of the batch job whose tags should be deleted.
     * @param accountId The ID of the account associated with the batch job.
```

```
    * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
    */
   public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
       DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .build();

       return asyncClient.deleteJobTagging(jobTaggingRequest)
           .thenAccept(response -> {
               System.out.println("You have successfully deleted " + jobId + "
tagging.");
           })
           .exceptionally(ex -> {
               System.err.println("Failed to delete job tags: " +
ex.getMessage());
               throw new RuntimeException(ex);
           });
   }

   /**
    * Asynchronously describes the specified job.
    *
    * @param jobId     the ID of the job to describe
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job
description is available
    * @throws RuntimeException if an error occurs while describing the job
    */
   public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
       DescribeJobRequest jobRequest = DescribeJobRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return getAsyncClient().describeJob(jobRequest)
           .thenAccept(response -> {
               System.out.println("Job ID: " + response.job().jobId());
               System.out.println("Description: " +
response.job().description());
```

```
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " +
manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " +
report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId         the AWS account ID associated with the job
     * @param iamRoleArn        the ARN of the IAM role to be used for the job
     * @param manifestLocation  the location of the job manifest file in S3
     * @param reportBucketName  the name of the S3 bucket to store the job report
     * @param uuid              a unique identifier for the job
```

```
     * @return a CompletableFuture that represents the asynchronous creation of
 the S3 job.
     *         The CompletableFuture will return the job ID if the job is created
 successfully,
     *         or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
 iamRoleArn,
                                                       String manifestLocation,
 String reportBucketName, String uuid) {

        String[] bucketName = new String[]{""};
        String[] parts = reportBucketName.split(":::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
 S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();
```

```
                JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
```

```
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }

    /**
     * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3
bucket.
     *
     * @param bucketName the name of the Amazon S3 bucket where the object is
stored
     * @param key the key (file name) of the object in the Amazon S3 bucket
     * @return the ETag of the object
     */
    public String getETag(String bucketName, String key) {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        return headObjectResponse.eTag();
    }

    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId     the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
```

```
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " +
ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    // Setup the S3 bucket required for this scenario.
    /**
     * Creates an Amazon S3 bucket with the specified name.
     *
     * @param bucketName the name of the S3 bucket to create
     * @throws S3Exception if there is an error creating the bucket
     */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();


            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
```

```
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
        Path filePath = Paths.get("src/main/resources/batch/",
fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        CompletableFuture<PutObjectResponse> future =
getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into
bucket " + bucketName);
```

```
            }
        }).join();
    }


    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Deletes an object from an Amazon S3 bucket asynchronously.
     *
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param objectName The name of the object to be deleted.
     * @return A {@link CompletableFuture} that completes when the object has
 been deleted,
     *         or throws a {@link RuntimeException} if an error occurs during the
 deletion.
     */
    public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
 objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
```

```
            .key(objectName)
            .build());

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        return getS3AsyncClient().deleteObjects(dor)
            .thenAccept(result -> {
                System.out.println("The object was deleted!");
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
            });
    }

    /**
     * Deletes a folder and all its contents asynchronously from an Amazon S3
bucket.
     *
     * @param bucketName the name of the S3 bucket containing the folder to be
deleted
     * @return a {@link CompletableFuture} that completes when the folder and its
contents have been deleted
     * @throws RuntimeException if any error occurs during the deletion process
     */
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(folderName)
            .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
```

```
                        .bucket(bucketName)
                        .key(obj.key())
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
                .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
                .thenCompose(v -> {
                    // Delete the folder.
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(folderName)
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted folder: " + folderName);
                            return deleteResponse;
                        });
                });
        }).join();
    }

    /**
     * Deletes an Amazon S3 bucket.
     *
     * @param bucketName the name of the bucket to delete
     * @return a {@link CompletableFuture} that completes when the bucket has
been deleted, or exceptionally if there is an error
     * @throws RuntimeException if there is an error deleting the bucket
     */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
```

```
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " +
bucketName, ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or
uploading the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " +
bucketName);
    }

    /**
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     * @param fileNames an array of file names to be deleted from the bucket
     * @param actions the S3BatchActions instance to be used for the file
deletion
     * @throws IOException if an I/O error occurs during the file deletion
     */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
                    actions.deleteBucketObjects(bucketName, fileName)
                   .thenRun(() -> System.out.println("Object deletion completed"))
                   .exceptionally(ex -> {
```

```
                        System.err.println("Error occurred: " + ex.getMessage());
                        return null;
                    });
        }
        System.out.println("All files have been deleted from the bucket " +
 bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\
\batch";
        Path path = Paths.get(directoryPath);

        try {
            // Create the directory if it doesn't exist.
            if (Files.notExists(path)) {
                Files.createDirectories(path);
                System.out.println("Created directory: " + path.toString());
            } else {
                System.out.println("Directory already exists: " +
 path.toString());
            }

            for (String fileName : fileNames) {
                // Check if the file is a .txt file.
                if (fileName.endsWith(".txt")) {
                    // Define the path for the new file.
                    Path filePath = path.resolve(fileName);
                    System.out.println("Attempting to create file: " +
 filePath.toString());

                    // Create and write content to the new file.
                    Files.write(filePath, "This is a test".getBytes());

                    // Verify the file was created.
                    if (Files.exists(filePath)) {
                        System.out.println("Successfully created file: " +
 filePath.toString());
                    } else {
                        System.out.println("Failed to create file: " +
 filePath.toString());
                    }
                }
```

```
            }

        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public String getAccountId() {
        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetCallerIdentityResponse callerIdentityResponse =
 stsClient.getCallerIdentity();
        return callerIdentityResponse.account();
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority
  - UpdateJobStatus

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Conheça o cenário básico do S3 Batch.

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
 None:
        """
        Initializes the S3BatchWrapper with AWS service clients.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                          access to AWS S3 services.
        :param s3control_client: A Boto3 Amazon S3 Control client. This client
 provides
                                 low-level access to AWS S3 Control services.
        :param sts_client: A Boto3 AWS STS client. This client provides low-level
                           access to AWS STS services.
        """
        self.s3_client = s3_client
        self.s3control_client = s3control_client
        self.sts_client = sts_client
        # Get region from the client for bucket creation logic
        self.region_name = self.s3_client.meta.region_name

    def get_account_id(self) -> str:
        """
        Get AWS account ID.

        Returns:
            str: AWS account ID
        """
        return self.sts_client.get_caller_identity()["Account"]
```

```
    def create_bucket(self, bucket_name: str) -> None:
        """
        Create an S3 bucket.

        Args:
            bucket_name (str): Name of the bucket to create

        Raises:
            ClientError: If bucket creation fails
        """
        try:
            if self.region_name and self.region_name != 'us-east-1':
                self.s3_client.create_bucket(
                    Bucket=bucket_name,
                    CreateBucketConfiguration={
                        'LocationConstraint': self.region_name
                    }
                )
            else:
                self.s3_client.create_bucket(Bucket=bucket_name)
            print(f"Created bucket: {bucket_name}")
        except ClientError as e:
            print(f"Error creating bucket: {e}")
            raise

    def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) ->
str:
        """
        Upload files to S3 bucket including manifest file.

        Args:
            bucket_name (str): Target bucket name
            file_names (list): List of file names to upload

        Returns:
            str: ETag of the manifest file

        Raises:
            ClientError: If file upload fails
        """
        try:
            for file_name in file_names:
                if file_name != "job-manifest.csv":
```

```
                    content = f"Content for {file_name}"
                    self.s3_client.put_object(
                        Bucket=bucket_name,
                        Key=file_name,
                        Body=content.encode('utf-8')
                    )
                    print(f"Uploaded {file_name} to {bucket_name}")

            manifest_content = ""
            for file_name in file_names:
                if file_name != "job-manifest.csv":
                    manifest_content += f"{bucket_name},{file_name}\n"

            manifest_response = self.s3_client.put_object(
                Bucket=bucket_name,
                Key="job-manifest.csv",
                Body=manifest_content.encode('utf-8')
            )
            print(f"Uploaded manifest file to {bucket_name}")
            print(f"Manifest content:\n{manifest_content}")
            return manifest_response['ETag'].strip('"')

        except ClientError as e:
            print(f"Error uploading files: {e}")
            raise

    def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                            report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports

        Returns:
            str: Job ID

        Raises:
            ClientError: If job creation fails
        """
```

```
        try:
            bucket_name = manifest_location.split(':::')[1].split('/')[0]
            manifest_key = 'job-manifest.csv'
            manifest_obj = self.s3_client.head_object(
                Bucket=bucket_name,
                Key=manifest_key
            )
            etag = manifest_obj['ETag'].strip('"')

            response = self.s3control_client.create_job(
                AccountId=account_id,
                Operation={
                    'S3PutObjectTagging': {
                        'TagSet': [
                            {
                                'Key': 'BatchTag',
                                'Value': 'BatchValue'
                            },
                        ]
                    }
                },
                Report={
                    'Bucket': report_bucket_name,
                    'Format': 'Report_CSV_20180820',
                    'Enabled': True,
                    'Prefix': 'batch-op-reports',
                    'ReportScope': 'AllTasks'
                },
                Manifest={
                    'Spec': {
                        'Format': 'S3BatchOperations_CSV_20180820',
                        'Fields': ['Bucket', 'Key']
                    },
                    'Location': {
                        'ObjectArn': manifest_location,
                        'ETag': etag
                    }
                },
                Priority=10,
                RoleArn=role_arn,
                Description='Batch job for tagging objects',
                ConfirmationRequired=True
            )
            job_id = response['JobId']
```

```
            print(f"The Job id is {job_id}")
            return job_id
        except ClientError as e:
            print(f"Error creating batch job: {e}")
            if 'Message' in str(e):
                print(f"Detailed error message: {e.response['Message']}")
            raise

    def check_job_failure_reasons(self, job_id: str, account_id: str) ->
 List[Dict[str, Any]]:
        """
        Check for any failure reasons of a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID

        Returns:
            list: List of failure reasons

        Raises:
            ClientError: If checking job failure reasons fails
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            if 'FailureReasons' in response['Job']:
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return response['Job'].get('FailureReasons', [])
        except ClientError as e:
            print(f"Error checking job failure reasons: {e}")
            raise

    def wait_for_job_ready(self, job_id: str, account_id: str, desired_status:
 str = 'Ready') -> bool:
        """
        Wait for a job to reach the desired status.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
```

```
            desired_status (str): Target status to wait for

        Returns:
            bool: True if desired status is reached, False otherwise

        Raises:
            ClientError: If checking job status fails
        """
        print(f"Waiting for job to become {desired_status}...")
        max_attempts = 60
        attempt = 0
        while attempt < max_attempts:
            try:
                response = self.s3control_client.describe_job(
                    AccountId=account_id,
                    JobId=job_id
                )
                current_status = response['Job']['Status']
                print(f"Current job status: {current_status}")
                if current_status == desired_status:
                    return True
                if current_status == 'Suspended':
                    print("Job is in Suspended state, can proceed with
activation")
                    return True
                if current_status in ['Active', 'Failed', 'Cancelled',
'Complete']:
                    print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
                    if 'FailureReasons' in response['Job']:
                        print("Failure reasons:")
                        for reason in response['Job']['FailureReasons']:
                            print(f"- {reason}")
                    return False

                time.sleep(20)
                attempt += 1
            except ClientError as e:
                print(f"Error checking job status: {e}")
                raise
        print(f"Timeout waiting for job to become {desired_status}")
        return False

    def update_job_priority(self, job_id: str, account_id: str) -> None:
```

```
        """
        Update the priority of a batch job and start it.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended']:
                self.s3control_client.update_job_priority(
                    AccountId=account_id,
                    JobId=job_id,
                    Priority=60
                )
                print("The job priority was updated")

                try:
                    self.s3control_client.update_job_status(
                        AccountId=account_id,
                        JobId=job_id,
                        RequestedJobStatus='Ready'
                    )
                    print("Job activated successfully")
                except ClientError as activation_error:
                    print(f"Note: Could not activate job automatically:
{activation_error}")
                    print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
```

```
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return

    def cancel_job(self, job_id: str, account_id: str) -> None:
        """
        Cancel an S3 batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended', 'Active']:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Cancelled'
                )
                print(f"Job {job_id} was successfully canceled.")
            elif current_status in ['Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - cannot be
cancelled")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
            else:
                print(f"Job is in '{current_status}' state - cancel not allowed")
        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise
```

```
    def describe_job_details(self, job_id: str, account_id: str) -> None:
        """
        Describe detailed information about a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            job = response['Job']
            print(f"Job ID: {job['JobId']}")
            print(f"Description: {job.get('Description', 'N/A')}")
            print(f"Status: {job['Status']}")
            print(f"Role ARN: {job['RoleArn']}")
            print(f"Priority: {job['Priority']}")
            if 'ProgressSummary' in job:
                progress = job['ProgressSummary']
                print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                      f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                      f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise

    def get_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Get tags associated with a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.get_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            tags = response.get('Tags', [])
```

```
            if tags:
                print(f"Tags for job {job_id}:")
                for tag in tags:
                    print(f"  {tag['Key']}: {tag['Value']}")
            else:
                print(f"No tags found for job ID: {job_id}")
        except ClientError as e:
            print(f"Error getting job tags: {e}")
            raise

    def put_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Add tags to a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.put_job_tagging(
                AccountId=account_id,
                JobId=job_id,
                Tags=[
                    {'Key': 'Environment', 'Value': 'Development'},
                    {'Key': 'Team', 'Value': 'DataProcessing'}
                ]
            )
            print(f"Additional tags were added to job {job_id}")
        except ClientError as e:
            print(f"Error adding job tags: {e}")
            raise

    def list_jobs(self, account_id: str) -> None:
        """
        List all batch jobs for the account.

        Args:
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.list_jobs(
                AccountId=account_id,
                JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
 'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
```

```
            )
            jobs = response.get('Jobs', [])
            for job in jobs:
                print(f"The job id is {job['JobId']}")
                print(f"The job priority is {job['Priority']}")
        except ClientError as e:
            print(f"Error listing jobs: {e}")
            raise

    def delete_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Delete all tags from a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.delete_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            print(f"You have successfully deleted {job_id} tagging.")
        except ClientError as e:
            print(f"Error deleting job tags: {e}")
            raise

    def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
        """
        Clean up all resources created during the scenario.

        Args:
            bucket_name (str): Name of the bucket to clean up
            file_names (list): List of files to delete

        Raises:
            ClientError: If cleanup fails
        """
        try:
            for file_name in file_names:
                self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
                print(f"Deleted {file_name}")

            response = self.s3_client.list_objects_v2(
```

```
                Bucket=bucket_name,
                Prefix='batch-op-reports/'
            )
            if 'Contents' in response:
                for obj in response['Contents']:
                    self.s3_client.delete_object(
                        Bucket=bucket_name,
                        Key=obj['Key']
                    )
                    print(f"Deleted {obj['Key']}")

            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Deleted bucket {bucket_name}")
        except ClientError as e:
            print(f"Error in cleanup: {e}")
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority
  - UpdateJobStatus

Ações para o Amazon S3 Control usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais de controle do Amazon S3 com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon S3 Control.

Exemplos

- Use CreateJobcom um AWS SDK ou CLI
- Use DeleteJobTaggingcom um AWS SDK
- Use DescribeJobcom um AWS SDK ou CLI
- Use GetJobTaggingcom um AWS SDK
- Use PutJobTaggingcom um AWS SDK
- Use UpdateJobPrioritycom um AWS SDK ou CLI
- Use UpdateJobStatuscom um AWS SDK ou CLI

Use **`CreateJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para criar um novo trabalho de Operações em Lote do Amazon S3

O exemplo `create-job` a seguir cria um trabalho de Operações em Lote do Amazon S3 para marcar objetos como `confidential` ` in the bucket ``employee-records`.

```
aws s3control create-job \
    --account-id 123456789012 \
    --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",
 "Value":"true"}] }}' \
    --report '{"Bucket":"arn:aws:s3:::employee-records-logs","Prefix":"batch-op-
create-job",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/
inv-report/7a6a9be4-072c-407e-85a2-
ec3e982f773e.csv","ETag":"69f52a4e9f797e987155d9c8f5880897"}}' \
    --priority 42 \
    --role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```

Saída:

```
{
    "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- Para obter detalhes da API, consulte CreateJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um trabalho assíncrono do S3.

```
   /**
    * Creates an asynchronous S3 job using the AWS Java SDK.
    *
    * @param accountId         the AWS account ID associated with the job
    * @param iamRoleArn        the ARN of the IAM role to be used for the job
    * @param manifestLocation  the location of the job manifest file in S3
    * @param reportBucketName  the name of the S3 bucket to store the job report
    * @param uuid              a unique identifier for the job
    * @return a CompletableFuture that represents the asynchronous creation of
the S3 job.
    *         The CompletableFuture will return the job ID if the job is created
successfully,
    *         or throw an exception if there is an error.
    */
   public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                     String manifestLocation,
String reportBucketName, String uuid) {

       String[] bucketName = new String[]{""};
       String[] parts = reportBucketName.split(":::");
       if (parts.length > 1) {
```

```
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
 S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
 JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
```

```
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }
```

Crie um trabalho de retenção de conformidade.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows
 you to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API
 call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();
```

```
        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Crie uma suspensão legal do trabalho.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
```

```
     * A compliance retention job in Amazon S3 Control is a feature that allows
 you to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API
 call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();
```

```
        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Crie um trabalho de retenção de governança.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateGovernanceRetentionJob {
```

```
   public static void main(String[]args) throws ParseException {
       final String usage = """

           Usage:
               <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
<manifestObjectVersionId>

           Where:
               manifestObjectArn - The Amazon Resource Name (ARN) of the S3
object that contains the manifest file for the governance objects.\s
               bucketName - The ARN of the S3 bucket where the job report will
be stored.
               roleArn - The ARN of the IAM role that will be used to perform
the governance retention operation.
               accountId - Your AWS account Id.
               manifestObjectVersionId =  A unique value that is used as the
`eTag` property of the `JobManifestLocation` object.
           """;

       if (args.length != 4) {
           System.out.println(usage);
           return;
       }

       String manifestObjectArn = args[0];
       String jobReportBucketArn = args[1];
       String roleArn = args[2];
       String accountId = args[3];
       String manifestObjectVersionId = args[4];

       S3ControlClient s3ControlClient = S3ControlClient.create();
       createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
   }

   public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
roleArn, String accountId, String manifestObjectVersionId) throws ParseException
{
       final JobManifestLocation manifestLocation =
JobManifestLocation.builder()
           .objectArn(manifestObjectArn)
           .eTag(manifestObjectVersionId)
```

```
            .build();

        final JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
            .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
            .build();

        final JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        final String jobReportPrefix = "reports/governance-objects";
        final JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        final Date jan30th = format.parse("30/01/2025");

        final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.GOVERNANCE)
                .retainUntilDate(jan30th.toInstant())
                .build())
            .build();

        final JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(s3SetObjectRetentionOperation)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
```

```
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
}
```

- Para obter detalhes da API, consulte CreateJoba Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                            report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports

        Returns:
            str: Job ID

        Raises:
            ClientError: If job creation fails
```

```
        """
        try:
            bucket_name = manifest_location.split(':::')[1].split('/')[0]
            manifest_key = 'job-manifest.csv'
            manifest_obj = self.s3_client.head_object(
                Bucket=bucket_name,
                Key=manifest_key
            )
            etag = manifest_obj['ETag'].strip('"')

            response = self.s3control_client.create_job(
                AccountId=account_id,
                Operation={
                    'S3PutObjectTagging': {
                        'TagSet': [
                            {
                                'Key': 'BatchTag',
                                'Value': 'BatchValue'
                            },
                        ]
                    }
                },
                Report={
                    'Bucket': report_bucket_name,
                    'Format': 'Report_CSV_20180820',
                    'Enabled': True,
                    'Prefix': 'batch-op-reports',
                    'ReportScope': 'AllTasks'
                },
                Manifest={
                    'Spec': {
                        'Format': 'S3BatchOperations_CSV_20180820',
                        'Fields': ['Bucket', 'Key']
                    },
                    'Location': {
                        'ObjectArn': manifest_location,
                        'ETag': etag
                    }
                },
                Priority=10,
                RoleArn=role_arn,
                Description='Batch job for tagging objects',
                ConfirmationRequired=True
            )
```

```
            job_id = response['JobId']
            print(f"The Job id is {job_id}")
            return job_id
        except ClientError as e:
            print(f"Error creating batch job: {e}")
            if 'Message' in str(e):
                print(f"Detailed error message: {e.response['Message']}")
            raise
```

- Para obter detalhes da API, consulte a CreateJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_manifest_arn  = 'arn:aws:s3:::my-bucket/job-manifest.csv'
        " iv_manifest_etag = 'abc123def456'
        " iv_report_bucket = 'arn:aws:s3:::my-report-bucket'
        DATA(lo_result) = lo_s3c->createjob(
          iv_accountid            = iv_account_id
          iv_rolearn              = iv_role_arn
          iv_confirmationrequired = abap_true
          iv_priority             = 10
          iv_description          = 'Batch job for tagging objects'
          io_operation            = NEW /aws1/cl_s3cjoboperation(
            io_s3putobjecttagging = NEW /aws1/cl_s3cs3setobjecttagop(
              it_tagset = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
                ( NEW /aws1/cl_s3cs3tag(
                    iv_key   = 'BatchTag'
                    iv_value = 'BatchValue' ) )
              )
            )
```

```
          )
          io_manifest             = NEW /aws1/cl_s3cjobmanifest(
            io_spec     = NEW /aws1/cl_s3cjobmanifestspec(
              iv_format = 'S3BatchOperations_CSV_20180820'
              it_fields = VALUE /aws1/
cl_s3cjobmanifestfield00=>tt_jobmanifestfieldlist(
                ( NEW /aws1/cl_s3cjobmanifestfield00( 'Bucket' ) )
                ( NEW /aws1/cl_s3cjobmanifestfield00( 'Key' ) )
              )
            )
            io_location = NEW /aws1/cl_s3cjobmanifestloc(
              iv_objectarn = iv_manifest_arn
              iv_etag      = iv_manifest_etag
            )
          )
          io_report               = NEW /aws1/cl_s3cjobreport(
            iv_bucket      = iv_report_bucket
            iv_format      = 'Report_CSV_20180820'
            iv_enabled     = abap_true
            iv_prefix      = 'batch-op-reports'
            iv_reportscope = 'AllTasks'
          )
        ).
        ov_job_id = lo_result->get_jobid( ).
        MESSAGE |S3 Batch Operations job created: { ov_job_id }| TYPE 'I'.
      CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
        MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_bad.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateJobreferência da API AWS SDK for SAP ABAP.

Use **`DeleteJobTagging`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteJobTagging`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously deletes the tags associated with a specific batch job.
    *
    * @param jobId     The ID of the batch job whose tags should be deleted.
    * @param accountId The ID of the account associated with the batch job.
    * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
    */
   public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
       DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .build();

       return asyncClient.deleteJobTagging(jobTaggingRequest)
           .thenAccept(response -> {
               System.out.println("You have successfully deleted " + jobId + "
tagging.");
           })
           .exceptionally(ex -> {
```

```
                System.err.println("Failed to delete job tags: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- Para obter detalhes da API, consulte DeleteJobTaggingna Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def delete_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Delete all tags from a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.delete_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            print(f"You have successfully deleted {job_id} tagging.")
        except ClientError as e:
            print(f"Error deleting job tags: {e}")
            raise
```

- Para obter detalhes da API, consulte a DeleteJobTaggingReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_s3c->deletejobtagging(
      iv_accountid = iv_account_id
      iv_jobid     = iv_job_id
    ).
    MESSAGE |Tags deleted from job { iv_job_id }| TYPE 'I'.
  CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_nf.
  CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_cli.
  CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_srv.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteJobTaggingreferência da API AWS SDK for SAP ABAP.

Use **`DescribeJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para descrever um novo trabalho de Operações em Lote do Amazon S3

O exemplo `describe-job` a seguir fornece os parâmetros de configuração e o status do trabalho de operações em lote especificado.

```
aws s3control describe-job \
    --account-id 123456789012 \
    --job-id 93735294-df46-44d5-8638-6356f335324e
```

Saída:

```
{
    "Job": {
        "TerminationDate": "2019-10-03T21:49:53.944Z",
        "JobId": "93735294-df46-44d5-8638-6356f335324e",
        "FailureReasons": [],
        "Manifest": {
            "Spec": {
                "Fields": [
                    "Bucket",
                    "Key"
                ],
                "Format": "S3BatchOperations_CSV_20180820"
            },
            "Location": {
                "ETag": "69f52a4e9f797e987155d9c8f5880897",
                "ObjectArn": "arn:aws:s3:::employee-records-logs/inv-
report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv"
            }
        },
        "Operation": {
            "S3PutObjectTagging": {
                "TagSet": [
                    {
                        "Value": "true",
                        "Key": "confidential"
```

```
                    }
                ]
            }
        },
        "RoleArn": "arn:aws:iam::123456789012:role/S3BatchJobRole",
        "ProgressSummary": {
            "TotalNumberOfTasks": 8,
            "NumberOfTasksFailed": 0,
            "NumberOfTasksSucceeded": 8
        },
        "Priority": 42,
        "Report": {
            "ReportScope": "AllTasks",
            "Format": "Report_CSV_20180820",
            "Enabled": true,
            "Prefix": "batch-op-create-job",
            "Bucket": "arn:aws:s3:::employee-records-logs"
        },
        "JobArn": "arn:aws:s3:us-west-2:123456789012:job/93735294-
df46-44d5-8638-6356f335324e",
        "CreationTime": "2019-10-03T21:48:48.048Z",
        "Status": "Complete"
    }
}
```

- Para obter detalhes da API, consulte DescribeJobem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
```

```
    * @return a {@link CompletableFuture} that completes when the job
description is available
    * @throws RuntimeException if an error occurs while describing the job
    */
   public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
       DescribeJobRequest jobRequest = DescribeJobRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return getAsyncClient().describeJob(jobRequest)
           .thenAccept(response -> {
               System.out.println("Job ID: " + response.job().jobId());
               System.out.println("Description: " +
response.job().description());
               System.out.println("Status: " + response.job().statusAsString());
               System.out.println("Role ARN: " + response.job().roleArn());
               System.out.println("Priority: " + response.job().priority());
               System.out.println("Progress Summary: " +
response.job().progressSummary());

               // Print out details about the job manifest.
               JobManifest manifest = response.job().manifest();
               System.out.println("Manifest Location: " +
manifest.location().objectArn());
               System.out.println("Manifest ETag: " +
manifest.location().eTag());

               // Print out details about the job operation.
               JobOperation operation = response.job().operation();
               if (operation.s3PutObjectTagging() != null) {
                   System.out.println("Operation: S3 Put Object Tagging");
                   System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
               }

               // Print out details about the job report.
               JobReport report = response.job().report();
               System.out.println("Report Bucket: " + report.bucket());
               System.out.println("Report Prefix: " + report.prefix());
               System.out.println("Report Format: " + report.format());
               System.out.println("Report Enabled: " + report.enabled());
```

```
                System.out.println("Report Scope: " +
report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- Para obter detalhes da API, consulte DescribeJoba Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def describe_job_details(self, job_id: str, account_id: str) -> None:
        """
        Describe detailed information about a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            job = response['Job']
            print(f"Job ID: {job['JobId']}")
            print(f"Description: {job.get('Description', 'N/A')}")
            print(f"Status: {job['Status']}")
            print(f"Role ARN: {job['RoleArn']}")
```

```
            print(f"Priority: {job['Priority']}")
            if 'ProgressSummary' in job:
                progress = job['ProgressSummary']
                print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                      f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                      f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise
```

- Para obter detalhes da API, consulte a DescribeJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3c->describejob(         " oo_result is returned for
testing purposes.
         iv_accountid = iv_account_id
         iv_jobid     = iv_job_id
       ).
       DATA(lo_job) = oo_result->get_job( ).
       DATA(lv_status) = lo_job->get_status( ).
       DATA(lv_priority) = lo_job->get_priority( ).
       DATA(lo_progress) = lo_job->get_progresssummary( ).
       IF lo_progress IS NOT INITIAL.
         MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }, | &&
                 |total={ lo_progress->get_totalnumberoftasks( ) }, | &&
                 |succeeded={ lo_progress->get_numberoftaskssucceeded( ) }, | &&
                 |failed={ lo_progress->get_numberoftasksfailed( ) }| TYPE 'I'.
```

```
        ELSE.
          MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeJobreferência da API AWS SDK for SAP ABAP.

Use **`GetJobTagging`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetJobTagging`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
     *
     * @param jobId     the ID of the job for which to retrieve the tags
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
     * @throws RuntimeException if an error occurs while retrieving the job tags
     */
    public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
        GetJobTaggingRequest request = GetJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Para obter detalhes da API, consulte GetJobTaggingga Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise
```

- Para obter detalhes da API, consulte a GetJobTaggingReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3c->getjobtagging(       " oo_result is returned for
testing purposes.
         iv_accountid = iv_account_id
         iv_jobid     = iv_job_id
       ).
       DATA(lt_tags) = oo_result->get_tags( ).
       MESSAGE |Retrieved { lines( lt_tags ) } tag(s) for job { iv_job_id }|
TYPE 'I'.
     CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
       MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_nf.
     CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
       MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_cli.
     CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
       MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_srv.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetJobTaggingreferência da API AWS SDK for SAP ABAP.

Use **`PutJobTagging`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `PutJobTagging`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously adds tags to a job in the system.
    *
    * @param jobId     the ID of the job to add tags to
    * @param accountId the account ID associated with the job
    * @return a CompletableFuture that completes when the tagging operation is
finished
    */
   public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
       S3Tag departmentTag = S3Tag.builder()
           .key("department")
           .value("Marketing")
           .build();

       S3Tag fiscalYearTag = S3Tag.builder()
           .key("FiscalYear")
           .value("2020")
           .build();

       PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .tags(departmentTag, fiscalYearTag)
           .build();
```

```
        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " +
ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Para obter detalhes da API, consulte PutJobTaggingga Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def put_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Add tags to a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.put_job_tagging(
                AccountId=account_id,
                JobId=job_id,
                Tags=[
                    {'Key': 'Environment', 'Value': 'Development'},
                    {'Key': 'Team', 'Value': 'DataProcessing'}
```

```
                ]
            )
            print(f"Additional tags were added to job {job_id}")
        except ClientError as e:
            print(f"Error adding job tags: {e}")
            raise
```

- Para obter detalhes da API, consulte a PutJobTaggingReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_s3c->putjobtagging(
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
          it_tags      = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
            ( NEW /aws1/cl_s3cs3tag(
                iv_key   = 'Environment'
                iv_value = 'Development' ) )
            ( NEW /aws1/cl_s3cs3tag(
                iv_key   = 'Team'
                iv_value = 'DataProcessing' ) )
          )
        ).
        MESSAGE |Tags added to job { iv_job_id }| TYPE 'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3ctoomanytagsex INTO DATA(lo_ex_tags).
        MESSAGE lo_ex_tags->get_text( ) TYPE 'I'.
```

```
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_tags.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutJobTaggingreferência da API AWS SDK for SAP ABAP.

Use **`UpdateJobPriority`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateJobPriority`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para atualizar a prioridade de um trabalho de Operações em Lote do Amazon S3

O exemplo `update-job-priority` a seguir atualiza o trabalho especificado para uma nova prioridade.

```
aws s3control update-job-priority \
    --account-id 123456789012 \
    --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
    --priority 52
```

Saída:

```
{
    "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",
    "Priority": 52
}
```

- Para obter detalhes da API, consulte UpdateJobPriorityem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates the priority of a job asynchronously.
    *
    * @param jobId     the ID of the job to update
    * @param accountId the ID of the account associated with the job
    * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error
has occurred
    */
   public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
       UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .priority(60)
           .build();

       CompletableFuture<Void> future = new CompletableFuture<>();
       getAsyncClient().updateJobPriority(priorityRequest)
           .thenAccept(response -> {
               System.out.println("The job priority was updated");
```

```
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }
```

- Para obter detalhes da API, consulte UpdateJobPrioritya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def update_job_priority(self, job_id: str, account_id: str) -> None:
        """
        Update the priority of a batch job and start it.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
```

```
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended']:
                self.s3control_client.update_job_priority(
                    AccountId=account_id,
                    JobId=job_id,
                    Priority=60
                )
                print("The job priority was updated")

                try:
                    self.s3control_client.update_job_status(
                        AccountId=account_id,
                        JobId=job_id,
                        RequestedJobStatus='Ready'
                    )
                    print("Job activated successfully")
                except ClientError as activation_error:
                    print(f"Note: Could not activate job automatically:
{activation_error}")
                    print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return
```

- Para obter detalhes da API, consulte a UpdateJobPriorityReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3c->updatejobpriority(   " oo_result is returned for
 testing purposes.
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
          iv_priority  = 60
        ).
        MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
        MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_bad.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateJobPriorityreferência da API AWS SDK for SAP ABAP.

Use **`UpdateJobStatus`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateJobStatus`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para atualizar o status de um trabalho de Operações em Lote do Amazon S3

O exemplo `update-job-status` a seguir cancela o trabalho especificado que está aguardando aprovação.

```
aws s3control update-job-status \
    --account-id 123456789012 \
    --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
    --requested-job-status Cancelled
```

Saída:

```
{
    "Status": "Cancelled",
    "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

O exemplo `update-job-status` a seguir confirma e executa o trabalho especificado que está aguardando aprovação.

```
aws s3control update-job-status \
    --account-id 123456789012 \
    --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
    --requested-job-status Ready
```

```
Output::

{
    "Status": "Ready",
    "JobId": "5782949f-3301-4fb3-
be34-8d5bab54dbca"
}
```

O exemplo `update-job-status` a seguir cancela o trabalho especificado que em execução.

```
 aws s3control update-job-status \
    --account-id 123456789012 \
    --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
    --requested-job-status Cancelled

Output::
{
         "Status": "Cancelling",
         "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

- Para obter detalhes da API, consulte UpdateJobStatusem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
 been updated to "CANCELLED".
```

```
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
{
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Para obter detalhes da API, consulte UpdateJobStatusa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def cancel_job(self, job_id: str, account_id: str) -> None:
        """
        Cancel an S3 batch job.
```

```
        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended', 'Active']:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Cancelled'
                )
                print(f"Job {job_id} was successfully canceled.")
            elif current_status in ['Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - cannot be
cancelled")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
            else:
                print(f"Job is in '{current_status}' state - cancel not allowed")
        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise
```

- Para obter detalhes da API, consulte a UpdateJobStatusReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_requested_status = 'Cancelled'
        oo_result = lo_s3c->updatejobstatus(     " oo_result is returned for
 testing purposes.
          iv_accountid          = iv_account_id
          iv_jobid              = iv_job_id
          iv_requestedjobstatus = iv_requested_status
        ).
        MESSAGE |Job { oo_result->get_jobid( ) } status updated to { oo_result-
>get_status( ) }| TYPE 'I'.
      CATCH /aws1/cx_s3cjobstatusexception INTO DATA(lo_ex_js).
        MESSAGE lo_ex_js->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_js.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateJobStatusreferência da API AWS SDK for SAP ABAP.

Exemplos de código para buckets do S3 Directory usando AWS SDKs

Os exemplos de código a seguir mostram como usar Amazon S3 Directory Buckets com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário de buckets de diretório do S3: mais informações sobre os buckets de diretório do S3.
- Referência de API de buckets de diretório do S3: detalhes sobre todas as ações disponíveis de buckets de diretório do S3.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de buckets do S3 Directory usando AWS SDKs
  - Olá, Amazon S3 directory buckets
  - Aprenda as noções básicas dos buckets do S3 Directory com um AWS SDK
  - Ações para buckets do S3 Directory usando AWS SDKs
    - Use AbortMultipartUploadcom um AWS SDK
    - Use CompleteMultipartUploadcom um AWS SDK
    - Use CopyObjectcom um AWS SDK
    - Use CreateBucketcom um AWS SDK

- Use CreateMultipartUploadcom um AWS SDK
- Use CreateSessioncom um AWS SDK
- Use DeleteBucketcom um AWS SDK
- Use DeleteBucketEncryptioncom um AWS SDK
- Use DeleteBucketPolicycom um AWS SDK
- Use DeleteObjectcom um AWS SDK
- Use DeleteObjectscom um AWS SDK
- Use GetBucketEncryptioncom um AWS SDK
- Use GetBucketPolicycom um AWS SDK
- Use GetObjectcom um AWS SDK
- Use GetObjectAttributescom um AWS SDK
- Use HeadBucketcom um AWS SDK
- Use HeadObjectcom um AWS SDK
- Use ListDirectoryBucketscom um AWS SDK
- Use ListMultipartUploadscom um AWS SDK
- Use ListObjectsV2 com um AWS SDK
- Use ListPartscom um AWS SDK
- Use PutBucketEncryptioncom um AWS SDK
- Use PutBucketPolicycom um AWS SDK
- Use PutObjectcom um AWS SDK
- Use UploadPartcom um AWS SDK
- Use UploadPartCopycom um AWS SDK

- Cenários para buckets do S3 Directory usando AWS SDKs
  - Crie uma URL pré-assinada para buckets de diretório do Amazon S3 para obter um objeto usando um AWS SDK

Exemplos básicos de buckets do S3 Directory usando AWS SDKs

Os exemplos de código a seguir mostram como usar os elementos básicos de buckets de diretório do Amazon S3 com AWS SDKs.

Exemplos

- Olá, Amazon S3 directory buckets
- Aprenda as noções básicas dos buckets do S3 Directory com um AWS SDK
- Ações para buckets do S3 Directory usando AWS SDKs
  - Use AbortMultipartUploadcom um AWS SDK
  - Use CompleteMultipartUploadcom um AWS SDK
  - Use CopyObjectcom um AWS SDK
  - Use CreateBucketcom um AWS SDK
  - Use CreateMultipartUploadcom um AWS SDK
  - Use CreateSessioncom um AWS SDK
  - Use DeleteBucketcom um AWS SDK
  - Use DeleteBucketEncryptioncom um AWS SDK
  - Use DeleteBucketPolicycom um AWS SDK
  - Use DeleteObjectcom um AWS SDK
  - Use DeleteObjectscom um AWS SDK
  - Use GetBucketEncryptioncom um AWS SDK
  - Use GetBucketPolicycom um AWS SDK
  - Use GetObjectcom um AWS SDK
  - Use GetObjectAttributescom um AWS SDK
  - Use HeadBucketcom um AWS SDK
  - Use HeadObjectcom um AWS SDK
  - Use ListDirectoryBucketscom um AWS SDK
  - Use ListMultipartUploadscom um AWS SDK
  - Use ListObjectsV2 com um AWS SDK
  - Use ListPartscom um AWS SDK
  - Use PutBucketEncryptioncom um AWS SDK
  - Use PutBucketPolicycom um AWS SDK
  - Use PutObjectcom um AWS SDK
  - Use UploadPartcom um AWS SDK
  - Use UploadPartCopycom um AWS SDK

Olá, Amazon S3 directory buckets

O exemplo de código a seguir mostra como começar a usar buckets de diretório do Amazon S3.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.s3.directorybucket;


import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
```

```
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
```

```
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error
code: {}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected
error: {}", e.getMessage(), e);
            } finally {
                s3Client.close();
            }
        }
    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
```

```
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Lists all S3 directory buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
```

```
            logger.error("Failed to list buckets: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String
 bucketName) {
        try {
            DeleteBucketRequest deleteBucketRequest =
 DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            s3Client.deleteBucket(deleteBucketRequest);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: " + bucketName + " - Error
 code: " + e.awsErrorDetails().errorCode(),
                    e);
            throw e;
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateBucket
  - ListDirectoryBuckets

Aprenda as noções básicas dos buckets do S3 Directory com um AWS SDK

Os exemplos de código a seguir mostram como:

- Configure uma VPC e um endpoint da VPC.

- Configure as políticas, os perfis e o usuário para trabalhar com os buckets de diretório do S3 e a classe de armazenamento S3 Express One Zone.
- Crie dois clientes do S3.
- Crie dois buckets.
- Crie um objeto e copie-o.
- Demonstre a diferença de desempenho.
- Preencha os buckets para mostrar a diferença lexicográfica.
- Confirme se o usuário deseja limpar os recursos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon S3.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;

    private static String stackName = "cfn-stack-s3-express-basics--" +
 UUID.randomUUID();
```

```
    private static String regularUser = "";
    private static String vpcId = "";
    private static String expressUser = "";

    private static String vpcEndpointId = "";

    private static final S3DirectoriesActions s3DirectoriesActions = new
S3DirectoriesActions();

    public static void main(String[] args) {
        try {
            s3ExpressScenario();
        } catch (RuntimeException e) {
            logger.info(e.getMessage());
        }
    }

    // Runs the scenario.
    private static void s3ExpressScenario() {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK
for Java V2.");
        logger.info("""
            Let's get started! First, please note that S3 Express One Zone works
best when working within the AWS infrastructure,
            specifically when working in the same Availability Zone (AZ). To see
the best results in this example and when you implement
            directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
            bucket.
            """);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        // Create an optional VPC and create 2 IAM users.
        UserNames userNames = createVpcUsers();
        String expressUserName = userNames.getExpressUserName();
        String regularUserName = userNames.getRegularUserName();

        //  Set up two S3 clients, one regular and one express,
        //  and two buckets, one regular and one directory.
        setupClientsAndBuckets(expressUserName, regularUserName);
```

```
        // Create an S3 session for the express S3 client and add objects to the
buckets.
        logger.info("Now let's add some objects to our buckets and demonstrate
how to work with S3 Sessions.");
        waitForInputToContinue(scanner);
        String bucketObject = createSessionAddObjects();

        // Demonstrate performance differences between regular and directory
buckets.
        demonstratePerformance(bucketObject);

        // Populate the buckets to show the lexicographical difference between
        // regular and express buckets.
        showLexicographicalDifferences(bucketObject);

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3
Express One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /*
      Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            }
            logger.info("Deleted directory bucket " + mdirectoryBucketName);
            mdirectoryBucketName = null;
            if (mregularBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
            }
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
```

```
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {
        logger.info(DASHES);
        logger.info("""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
            when object names are listed. Now let's explore how directory buckets
store
            objects in a different manner to regular buckets. The key is in the
name
            "Directory". Where regular buckets store their key/value pairs in a
            flat manner, directory buckets use actual directories/folders.
            This allows for more rapid indexing, traversing, and therefore
            retrieval times!

            The more segmented your bucket is, with lots of
            directories, sub-directories, and objects, the more efficient it
becomes.
            This structural difference also causes `ListObject` operations to
behave
            differently, which can cause unexpected results. Let's add a few
more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);

        //  Populate a few more files in each bucket so that we can use
        //  ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
```

```
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof NoSuchBucketException) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        try {
            // List objects in both S3 buckets.
            List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,
mregularBucketName).join();

            logger.info("Directory bucket content");
            for (String obj : dirBucketObjects) {
                logger.info(obj);
            }

            logger.info("Regular bucket content");
            for (String obj : regBucketObjects) {
                logger.info(obj);
```

```
            }
        } catch (CompletionException e) {
            logger.error("Async operation failed: {} ",
e.getCause().getMessage());
            return;
        }

        logger.info("""
            Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
            because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
            creates directories and uses the object "key" as a path to the
object.
            """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from
a directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("""
```

```
            Now, let's do a performance test. We'll download the same object from
each
            bucket repeatedly and compare the total time needed.

            Note: the performance difference will be much more pronounced if this
            example is run in an EC2 instance in the same Availability Zone as
            the bucket.
            """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

        // Ask if the user wants to download a different number.
        logger.info("Would you like to download the file a different number of
times? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            int maxDownloads = 1_000_000;

            // Ask for a valid number of downloads.
            while (true) {
                logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
                if (scanner.hasNextInt()) {
                    downloads = scanner.nextInt();
                    if (downloads >= 1 && downloads <= maxDownloads) {
                        break;
                    } else {
                        logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                    }
                } else {
                    logger.info("Invalid input. Please enter a valid integer.");
                    scanner.next();
                }
            }

            logger.info("You have chosen to download {}  items.", downloads);
        } else {
            logger.info("No changes made. Using default downloads: {}",
downloads);
        }
```

```
        // Simulating the download process for the directory bucket.
        logger.info("Downloading from the directory bucket.");
        long directoryTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                // Get the object from the directory bucket.
                s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
        long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

        // Download from the regular bucket.
        logger.info("Downloading from the regular bucket.");
        long normalTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
```

```
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }

        long normalTimeDifference = System.nanoTime() - normalTimeStart;
        logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
        long difference = normalTimeDifference - directoryTimeDifference;
        logger.info("That's a difference of " + difference + " nanoseconds, or");
        logger.info(difference / 1_000_000_000.0 + " seconds.");

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAddObjects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
```

```
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with
permission to
            work with directory buckets.

            It's important to remember the user permissions when interacting
with
            directory buckets. Instead of validating permissions on every call
as
            regular buckets do, directory buckets utilize the user credentials
and session
            token to validate. This allows for much faster connection speeds on
every call.
            For single calls, this is low, but for many concurrent calls
            this adds up to a lot of time saved.
            """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
     *     <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
     * </ol>
     *
     * @return a {@link UserNames} object containing the names of the created IAM
users
     */
    public static UserNames createVpcUsers() {
```

```
        /*
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
without.
        */
        logger.info(DASHES);
        logger.info("""
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("""
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                """);
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("""
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            """);
```

```
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");

        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the
stack outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express
permissions
     * @param regularUserName the username for the user with regular S3
permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
```

```
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
```

```
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        logger.info("""
            We can now use the ExpressUser client to make calls to S3 Express
operations.
            """);
        waitForInputToContinue(locscanner);
        logger.info(DASHES);
        logger.info("""
            4. Create two buckets.
            Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
            behave differently from regular S3 buckets which we will explore
here. We'll also create a regular bucket, put
            an object into the regular bucket, and copy it to the directory
bucket.
            """);

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory
bucket.
            We'll choose one that is supported.
            """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
```

```
            regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
 bucket name.
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(),
 ce);
            } else {
                logger.error("An unexpected error occurred: {}",
 cause.getMessage(), ce);
            }
            return;
        }
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
 bucket."
             """);

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
 directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
 cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
 cause.getMessage(), ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
 regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**
     * Creates an asynchronous S3 client with the specified access key and secret
access key.
     *
     * @param accessKeyId     the AWS access key ID
     * @param secretAccessKey the AWS secret access key
     * @return a {@link CompletableFuture} that asynchronously creates the S3
client
     * @throws IllegalArgumentException if the access key ID or secret access key
is null
     */
    public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
== null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret
Access Key must not be null or empty");
            }

            AwsBasicCredentials awsCredentials =
AwsBasicCredentials.create(accessKeyId, secretAccessKey);
            return S3AsyncClient.builder()

.credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
                .region(Region.US_WEST_2)
                .build();
        });
    }
```

```
    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper para os métodos de SDK do Amazon S3.

```
public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
```

```
                .build();

            iamAsyncClient = IamAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iamAsyncClient;
    }

    private static Ec2AsyncClient getEc2AsyncClient() {
        if (ec2AsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .httpClient(httpClient)
                .region(Region.US_WEST_2)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes the specified S3 bucket and all the objects within it
asynchronously.
     *
     * @param s3AsyncClient the S3 asynchronous client to use for the operations
     * @param bucketName the name of the S3 bucket to be deleted
     * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
```

```
    *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
    * @throws CompletionException if there was an error deleting the bucket or
its objects
    */
   public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
       ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .build();

       return s3AsyncClient.listObjectsV2(listRequest)
           .thenCompose(listResponse -> {
               if (!listResponse.contents().isEmpty()) {
                   List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                       .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                       .collect(Collectors.toList());

                   DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                       .bucket(bucketName)

.delete(Delete.builder().objects(objectIdentifiers).build())
                       .build();

                   return s3AsyncClient.deleteObjects(deleteRequest)
                       .thenAccept(deleteResponse -> {
                           if (!deleteResponse.errors().isEmpty()) {
                               deleteResponse.errors().forEach(error ->
                                   logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                           }
                       });
               }
               return CompletableFuture.completedFuture(null);
           })
           .thenCompose(ignored -> {
               DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                   .bucket(bucketName)
                   .build();
               return s3AsyncClient.deleteBucket(deleteBucketRequest);
```

```
            })
            .thenCompose(ignored -> {
                S3AsyncWaiter waiter = s3AsyncClient.waiter();
                HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
                return waiter.waitUntilBucketNotExists(headBucketRequest);
            })
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
                    }
                    throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
                }
                logger.info("Bucket deleted successfully: " + bucketName);
            });
    }

    /**
     *  Lists the objects in an S3 bucket asynchronously.
     *
     * @param s3Client the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the objects to list
     * @return a {@link CompletableFuture} that contains the list of object keys
in the specified bucket
     */
    public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3Client.listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .toList())
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
                }
```

```
            });
    }

    /**
     * Retrieves an object from an Amazon S3 bucket asynchronously.
     *
     * @param s3Client   the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName    the unique identifier (key) of the object to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
     */
    public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        // Get the object asynchronously and transform it into a byte array
        return s3Client.getObject(objectRequest,
AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client           the S3 async client to use for the copy
operation
     * @param sourceBucket       the name of the source bucket
     * @param sourceKey          the key of the object to be copied in the source
bucket
     * @param destinationBucket  the name of the destination bucket
     * @param destinationKey     the key of the copied object in the destination
bucket
```

```
    * @return a {@link CompletableFuture} that completes when the copy operation
is finished
    */
   public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
{
       CopyObjectRequest copyRequest = CopyObjectRequest.builder()
           .sourceBucket(sourceBucket)
           .sourceKey(sourceKey)
           .destinationBucket(destinationBucket)
           .destinationKey(destinationKey)
           .build();

       return s3Client.copyObject(copyRequest)
           .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
           .whenComplete((ignored, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof S3Exception) {
                       throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                   }
                   throw new CompletionException("Failed to copy object",
exception);
               }
           });
   }

   /**
    * Asynchronously creates a session for the specified S3 bucket.
    *
    * @param s3Client   the S3 asynchronous client to use for creating the
session
    * @param bucketName the name of the S3 bucket for which to create the
session
    * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
    */
   public CompletableFuture<CreateSessionResponse>
createSessionAsync(S3AsyncClient s3Client, String bucketName) {
       CreateSessionRequest request = CreateSessionRequest.builder()
```

```
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });

    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The asynchronous S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The Availability Zone where the bucket will be created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
```

```
            .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
     * @param bucketName  the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
     *         when the bucket is successfully created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createBucket(bucketRequest)
            .thenCompose(response -> {
                S3AsyncWaiter s3Waiter = s3Client.waiter();
                HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
```

```
                    .build();
                return s3Waiter.waitUntilBucketExists(bucketRequestWait);
            })
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                    }
                }
                logger.info(bucketName + " is ready");
            });
    }

    /**
     * Uploads an object to an Amazon S3 bucket asynchronously.
     *
     * @param s3Client     the S3 async client to use for the upload
     * @param bucketName   the destination S3 bucket name
     * @param bucketObject the name of the object to be uploaded
     * @param text         the content to be uploaded as the object
     */
    public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(bucketObject)
            .build();

        return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof NoSuchBucketException) {
                        throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
```

```
                    }
                }
            });
    }

    /**
     * Creates an AWS IAM access key asynchronously for the specified user name.
     *
     * @param userName the name of the IAM user for whom to create the access key
     * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
     */
    public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(userName)
            .build();

        return getIAMAsyncClient().createAccessKey(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Access Key Created.");
                } else {
                    if (exception == null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof IamException) {
                            throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                        } else {
                            throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                        }
                    }
                }
            });
    }

    /**
     * Asynchronously selects an Availability Zone ID from the available EC2
zones.
     *
     * @return A {@link CompletableFuture} that resolves to the selected
Availability Zone ID.
```

```
    * @throws CompletionException if an error occurs during the request or
processing.
    */
   public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
       DescribeAvailabilityZonesRequest zonesRequest =
DescribeAvailabilityZonesRequest.builder()
           .build();

       return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
           .thenCompose(response -> {
               List<AvailabilityZone> zonesList = response.availabilityZones();
               if (zonesList.isEmpty()) {
                   logger.info("No availability zones found.");
                   return CompletableFuture.completedFuture(null); // Return
null if no zones are found
               }

               List<String> zoneIds = zonesList.stream()
                   .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
                   .toList();

               return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
                   .thenApply(selectedZone -> {
                       // Return only the selected Zone ID (e.g., "usw2-az1").
                       return selectedZone.zoneId();
                   });
           })
           .whenComplete((result, exception) -> {
               if (exception == null) {
                   if (result != null) {
                       logger.info("Selected Availability Zone ID: " + result);
                   } else {
                       logger.info("No availability zone selected.");
                   }
               } else {
                   Throwable cause = exception.getCause();
                   if (cause instanceof Ec2Exception) {
                       throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
```

```
                }
            });
    }

    /**
     * Prompts the user to select an Availability Zone from the given list.
     *
     * @param zonesList the list of Availability Zones
     * @param zoneIds the list of zone IDs
     * @return the selected Availability Zone
     */
    private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
        Scanner scanner = new Scanner(System.in);
        int index = -1;

        while (index < 0 || index >= zoneIds.size()) {
            logger.info("Select an availability zone:");
            IntStream.range(0, zoneIds.size()).forEach(i ->
                logger.info(i + ": " + zoneIds.get(i))
            );

            logger.info("Enter the number corresponding to your choice: ");
            if (scanner.hasNextInt()) {
                index = scanner.nextInt();
            } else {
                scanner.next();
            }
        }

        AvailabilityZone selectedZone = zonesList.get(index);
        logger.info("You selected: " + selectedZone.zoneId());
        return selectedZone;
    }

    /**
     * Asynchronously sets up a new VPC, including creating the VPC, finding the
associated route table, and
     * creating a VPC endpoint for the S3 service.
     *
     * @return a {@link CompletableFuture} that, when completed, contains a
AbstractMap with the
     *         VPC ID and VPC endpoint ID.
```

```
     */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();

                DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
```

```
                    });
            })
            .thenCompose(vpcAndRouteTable -> {
                String vpcId = vpcAndRouteTable.getKey();
                String routeTableId = vpcAndRouteTable.getValue();
                Region region =
 getEc2AsyncClient().serviceClientConfiguration().region();
                String serviceName = String.format("com.amazonaws.%s.s3express",
 region.id());

                CreateVpcEndpointRequest endpointRequest =
 CreateVpcEndpointRequest.builder()
                    .vpcId(vpcId)
                    .routeTableIds(routeTableId)
                    .serviceName(serviceName)
                    .build();

                return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
                    .thenApply(vpcEndpointResponse -> {
                        String vpcEndpointId =
 vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                        logger.info("VPC Endpoint created: {}", vpcEndpointId);
                        return new AbstractMap.SimpleEntry<>(vpcId,
 vpcEndpointId);
                    });
            })
            .exceptionally(exception -> {
                Throwable cause = exception.getCause() != null ?
 exception.getCause() : exception;
                if (cause instanceof Ec2Exception) {
                    logger.error("EC2 error during VPC setup: {}",
 cause.getMessage(), cause);
                    throw new CompletionException("EC2 error during VPC setup: "
 + cause.getMessage(), cause);
                }

                logger.error("VPC setup failed: {}", cause.getMessage(), cause);
                throw new CompletionException("VPC setup failed: " +
 cause.getMessage(), cause);
            });
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário demonstrando os conceitos básicos sobre os buckets de diretório do Amazon S3 e a classe S3 Express One Zone.

```
        echo "\n";
        echo "--------------------------------------\n";
        echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
        echo "--------------------------------------\n";

        // Change these both of these values to use a different region/
availability zone.
        $region = "us-west-2";
        $az = "usw2-az1";

        $this->s3Service = new S3Service(new S3Client(['region' => $region]));
        $this->iamService = new IAMService(new IamClient(['region' => $region]));
```

```
        $uuid = uniqid();

        echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
 in this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute
 resources in the same AZ as your Directory
bucket.\n
INTRO;
        pressEnter();
        // 1. Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
        // the need to pass through an internet gateway or NAT device.
        echo "\n";
        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program
 is running in an EC2 instance in the same AZ as your Directory buckets will be.
\n";
        $ec2Choice = testable_readline("Are you running this in an EC2 instance
 located in the same AZ as your intended Directory buckets? Enter Y/y to setup a
 VPC Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it,
 and create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
 $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
 ".s3express";
```

```
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
 $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
        echo "Now, we'll set up some policies, roles, and a user. This user will
 only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();
        $file = file_get_contents(__DIR__ . "/../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
        $result = $this->cloudFormationClient->createStack([
            'StackName' => $stackName,
            'TemplateBody' => $file,
            'Capabilities' => ['CAPABILITY_IAM'],
        ]);
        $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
 ['StackName' => $stackName]);
        try {
            $waiter->promise()->wait();
        }catch(CloudFormationException $caught){
            echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
        $this->resources['stackName'] = $stackName;
        $stackInfo = $this->cloudFormationClient->describeStacks([
            'StackName' => $result['StackId'],
        ]);

        $expressUserName = "";
        $regularUserName = "";
        foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
            if ($output['OutputKey'] == "RegularUser") {
                $regularUserName = $output['OutputValue'];
            }
```

```
            if ($output['OutputKey'] == "ExpressUser") {
                $expressUserName = $output['OutputValue'];
            }
        }
        $regularKey = $this->iamService->createAccessKey($regularUserName);
        $regularCredentials = new Credentials($regularKey['AccessKeyId'],
 $regularKey['SecretAccessKey']);
        $expressKey = $this->iamService->createAccessKey($expressUserName);
        $expressCredentials = new Credentials($expressKey['AccessKeyId'],
 $expressKey['SecretAccessKey']);

        // 3. Create an additional client using the credentials with S3 Express
 permissions.
        echo "\n";
        echo "3. Create an additional client using the credentials with S3
 Express permissions.\n";
        echo "This client is created with the credentials associated with the
 user account with the S3 Express policy attached, so it can perform S3 Express
 operations.\n";
        pressEnter();
        $s3RegularClient = new S3Client([
            'Region' => $region,
            'Credentials' => $regularCredentials,
        ]);
        $s3RegularService = new S3Service($s3RegularClient);
        $s3ExpressClient = new S3Client([
            'Region' => $region,
            'Credentials' => $expressCredentials,
        ]);
        $s3ExpressService = new S3Service($s3ExpressClient);
        echo "All the roles and policies were created an attached to the user.
 Then, a new S3 Client and Service were created using that user's credentials.
\n";
        echo "We can now use this client to make calls to S3 Express operations.
 Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
 Express.\n";
        pressEnter();

        // 4. Create two buckets.
        echo "\n";
        echo "3. Create two buckets.\n";
        echo "Now we will create a Directory bucket, which is the linchpin of the
 S3 Express One Zone service.\n";
```

```
        echo "Directory buckets behave in different ways from regular S3 buckets,
 which we will explore here.\n";
        echo "We'll also create a normal bucket, put an object into the normal
 bucket, and copy it over to the Directory bucket.\n";
        pressEnter();

        // Create a directory bucket. These are different from normal S3 buckets
 in subtle ways.
        $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-
s3";
        echo "Now, let's create the actual Directory bucket, as well as a regular
 bucket.\n";
        pressEnter();
        $s3ExpressService->createBucket($directoryBucketName, [
            'CreateBucketConfiguration' => [
                'Bucket' => [
                    'Type' => "Directory", // This is what causes S3 to create a
 Directory bucket as opposed to a normal bucket.
                    'DataRedundancy' => "SingleAvailabilityZone",
                ],
                'Location' => [
                    'Name' => $az,
                    'Type' => "AvailabilityZone",
                ],
            ],
        ]);
        $this->resources['directoryBucketName'] = $directoryBucketName;

        // Create a normal bucket.
        $normalBucketName = "normal-bucket-$uuid";
        $s3RegularService->createBucket($normalBucketName);
        $this->resources['normalBucketName'] = $normalBucketName;
        echo "Great! Both buckets were created.\n";
        pressEnter();

        // 5. Create an object and copy it over.
        echo "\n";
        echo "5. Create an object and copy it over.\n";
        echo "We'll create a basic object consisting of some text and upload it
 to the normal bucket.\n";
        echo "Next, we'll copy the object into the Directory bucket using the
 regular client.\n";
        echo "This works fine, because Copy operations are not restricted for
 Directory buckets.\n";
```

```
        pressEnter();

        $objectKey = "basic-text-object";
        $s3RegularService->putObject($normalBucketName, $objectKey, $args =
['Body' => "Look Ma, I'm a bucket!"]);
        $this->resources['objectKey'] = $objectKey;

        // Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
        $s3ExpressService->createSession($directoryBucketName);
        $s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

        echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
        echo "Instead of validating permissions on every call as normal buckets
do, Directory buckets utilize the user credentials and session token to
validate.\n";
        echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot
of time saved.\n";
        pressEnter();

        // 6. Demonstrate performance difference.
        echo "\n";
        echo "6. Demonstrate performance difference.\n";
        $downloads = 1000;
        echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
        $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
        if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
            $downloads = $downloadChoice;
        }

        // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
        $directoryStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3ExpressService->getObject($directoryBucketName, $objectKey);
```

```
        }
        $directoryEndTime = hrtime(true);
        $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

        $normalStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3RegularService->getObject($normalBucketName, $objectKey);
        }
        $normalEndTime = hrtime(true);
        $normalTimeDiff = $normalEndTime - $normalStartTime;

        echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
 normal bucket took $normalTimeDiff.\n";
        echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
 " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
 seconds.\n";
        pressEnter();

        // 7. Populate the buckets to show the lexicographical difference.
        echo "\n";
        echo "7. Populate the buckets to show the lexicographical difference.\n";
        echo "Now let's explore how Directory buckets store objects in a
 different manner to regular buckets.\n";
        echo "The key is in the name \"Directory!\"\n";
        echo "Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual directories/folders.\n";
        echo "This allows for more rapid indexing, traversing, and therefore
 retrieval times!\n";
        echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
        echo "This structural difference also causes ListObjects to behave
 differently, which can cause unexpected results.\n";
        echo "Let's add a few more objects with layered directories as see how
 the output of ListObjects changes.\n";
        pressEnter();

        // Populate a few more files in each bucket so that we can use
 ListObjects and show the difference.
        $otherObject = "other/$objectKey";
        $altObject = "alt/$objectKey";
        $otherAltObject = "other/alt/$objectKey";
        $s3ExpressService->putObject($directoryBucketName, $otherObject);
        $s3RegularService->putObject($normalBucketName, $otherObject);
        $this->resources['otherObject'] = $otherObject;
```

```
        $s3ExpressService->putObject($directoryBucketName, $altObject);
        $s3RegularService->putObject($normalBucketName, $altObject);
        $this->resources['altObject'] = $altObject;
        $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
        $s3RegularService->putObject($normalBucketName, $otherAltObject);
        $this->resources['otherAltObject'] = $otherAltObject;

        $listDirectoryBucket = $s3ExpressService-
>listObjects($directoryBucketName);
        $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

        // Directory bucket content
        echo "Directory bucket content\n";
        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical
 order, while the directory bucket does not. This is because the normal bucket
 considers the whole \"key\" to be the object identifies, while the directory
 bucket actually creates directories and uses the object \"key\" as a path to the
 object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
 Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
 created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }



namespace S3;

use Aws\CommandInterface;
```

```
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }
```

```
    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "\nPlease fix error with bucket deletion before continuing.
\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }
```

```
    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object downloading before
continuing.";
            }
            throw $exception;
        }
        return $object;
    }
```

```
   public function copyObject($bucketName, $key, $copySource, array $args = [])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
       try {
           $this->client->copyObject($parameters);
           if ($this->verbose) {
               echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object copying before continuing.";
           }
           throw $exception;
       }
   }



   public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
       try {
           $objects = $this->client->listObjectsV2($parameters);
           if ($this->verbose) {
               echo "Retrieved the list of objects from: $bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with list objects before continuing.";
           }
           throw $exception;
       }
       return $objects;
```

```
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args
 = [])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
 ['Objects' => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
```

```
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
 error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array
 $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
```

```
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
```

```
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário demonstrando os conceitos básicos sobre os buckets de diretório do Amazon S3 e a classe S3 Express One Zone.

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
 Express."""

    def __init__(
        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
        Runs the scenario.
        """
        print("")
        print_dashes()
```

```
        print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
 3)!")
        print_dashes()
        print(
            """
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
 in this example and when you implement
Directory buckets into your infrastructure, it is best to put your compute
 resources in the same AZ as your Directory
bucket.
    """
        )
        press_enter_to_continue()

        # Create an optional VPC and create 2 IAM users.
        express_user_name, regular_user_name = self.create_vpc_and_users()

        # Set up two S3 clients, one regular and one express, and two buckets,
 one regular and one express.
        self.setup_clients_and_buckets(express_user_name, regular_user_name)

        # Create an S3 session for the express S3 client and add objects to the
 buckets.
        bucket_object = self.create_session_and_add_objects()

        # Demonstrate performance differences between regular and express
 buckets.
        self.demonstrate_performance(bucket_object)

        # Populate the buckets to show the lexicographical difference between
 regular and express buckets.
        self.show_lexicographical_differences(bucket_object)

        print("")
        print("That's it for our tour of the basic operations for S3 Express One
 Zone.")

        if q.ask(
            "Would you like to delete all the resources created during this demo
 (y/n)? ",
            q.is_yesno,
        ):
```

```
            self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
 without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
 an EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
 same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in
 production!")
        print(
            """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
 permissions to do S3 Express One Zone actions.
            """
        )
        press_enter_to_continue()
        stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
        template_as_string = S3ExpressScenario.get_template_as_string()
        self.stack = self.deploy_cloudformation_stack(stack_name,
 template_as_string)
        regular_user_name = None
        express_user_name = None
        outputs = self.stack.outputs
```

```
        for output in outputs:
            if output.get("OutputKey") == "RegularUser":
                regular_user_name = output.get("OutputValue")
            elif output.get("OutputKey") == "ExpressUser":
                express_user_name = output.get("OutputValue")
        if not regular_user_name or not express_user_name:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
            'regular_user_name'={regular_user_name},
 'express_user_name'={express_user_name}
            """
            logger.error(error_string)
            raise ValueError(error_string)
        return express_user_name, regular_user_name

    def setup_clients_and_buckets(
        self, express_user_name: str, regular_user_name: str
    ) -> None:
        """
        Set up two S3 clients, one regular and one express, and two buckets, one
 regular and one express.
        :param express_user_name: The name of the user with S3 Express
 permissions.
        :param regular_user_name: The name of the user with regular S3
 permissions.
        """
        regular_credentials = self.create_access_key(regular_user_name)
        express_credentials = self.create_access_key(express_user_name)
        # 3. Create an additional client using the credentials with S3 Express
 permissions.
        print(
            """
3. Create an additional client using the credentials with S3 Express permissions.
 This client is created with the
credentials associated with the user account with the S3 Express policy attached,
 so it can perform S3 Express operations.
"""
        )
        press_enter_to_continue()
        s3_regular_client = self.create_s3__client_with_access_key_credentials(
            regular_credentials
        )
        self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
        s3_express_client = self.create_s3__client_with_access_key_credentials(
```

```
            express_credentials
        )
        self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
        print(
            """
All the roles and policies were created and attached to the user. Then a new S3
 Client were created using
that user's credentials. We can now use this client to make calls to S3 Express
 operations. Keeping permissions in mind
(and adhering to least-privilege) is crucial to S3 Express.
 """
        )
        press_enter_to_continue()
        # 4. Create two buckets.
        print(
            """
3. Create two buckets.
Now we will create a Directory bucket which is the linchpin of the S3 Express One
 Zone service. Directory buckets
behave in different ways from regular S3 buckets which we will explore here.
 We'll also create a normal bucket, put
an object into the normal bucket, and copy it over to the Directory bucket.
"""
        )

        # Create a directory bucket. These are different from normal S3 buckets
 in subtle ways.
        bucket_prefix = q.ask(
            "Enter a bucket name prefix that will be used for both buckets: ",
            q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
        )

        # Some availability zones are not supported for Directory buckets. We'll
 choose one that is supported.
        print(
            "Now, let's choose an availability zone for the Directory bucket.
 We'll choose one that is supported."
        )
        while True:
            availability_zone = self.select_availability_zone_id(self.region)
            # Construct the parts of a directory bucket name that is made unique
 with a UUID string.
            directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"
```

```
            max_uuid_length = 63 - len(bucket_prefix) -
len(directory_bucket_suffix) - 1
            bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
            directory_bucket_name = (
                f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
            )
            regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
            configuration = {
                "Bucket": {
                    "Type": "Directory",
                    "DataRedundancy": "SingleAvailabilityZone",
                },
                "Location": {
                    "Name": availability_zone["ZoneId"],
                    "Type": "AvailabilityZone",
                },
            }
            press_enter_to_continue()
            print(
                "Now, let's create the actual Directory bucket, as well as a
regular bucket."
            )
            press_enter_to_continue()
            try:
                self.s3_express_wrapper.create_bucket(
                    directory_bucket_name, configuration
                )
                break
            except ClientError as client_error:
                if client_error.response["Error"]["Code"] == "InvalidBucketName":
                    print(
                        f"Bucket '{directory_bucket_name}' is invalid. This may
be because of selected availability zone."
                    )
                    if q.ask(
                        "Would you like to select a different availability zone?
",
                        q.is_yesno,
                    ):
                        continue
                    else:
                        raise
                else:
                    raise
```

```
        print(f"Created directory bucket, '{directory_bucket_name}'")
        self.directory_bucket_name = directory_bucket_name

        self.s3_regular_wrapper.create_bucket(regular_bucket_name)
        print(f"Created regular bucket, '{regular_bucket_name}'")
        self.regular_bucket_name = regular_bucket_name
        print("Great! Both buckets were created.")
        press_enter_to_continue()

    def create_session_and_add_objects(self) -> None:
        """
        Create a session for the express S3 client and add objects to the
 buckets.
        """
        print(
            """
5. Create an object and copy it over.
We'll create a basic object consisting of some text and upload it to the normal
 bucket. Next we'll copy the object
into the Directory bucket using the regular client. This works fine because copy
 operations are not restricted for
Directory buckets.
        """
        )
        press_enter_to_continue()
        bucket_object = "basic-text-object"
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
        )
        self.s3_express_wrapper.create_session(self.directory_bucket_name)
        self.s3_express_wrapper.copy_object(
            self.regular_bucket_name,
            bucket_object,
            self.directory_bucket_name,
            bucket_object,
        )
        print(
            """
It worked! It's important to remember the user permissions when interacting with
 Directory buckets. Instead of validating
permissions on every call as normal buckets do, Directory buckets utilize the
 user credentials and session token to validate.
This allows for much faster connection speeds on every call. For single calls,
 this is low, but for many concurrent calls
```

```
this adds up to a lot of time saved.
"""
        )
        press_enter_to_continue()
        return bucket_object

    def demonstrate_performance(self, bucket_object: str) -> None:
        """
        Demonstrate performance differences between regular and Directory
 buckets.
        :param bucket_object: The name of the object to download from each
 bucket.
        """
        print("")
        print("6. Demonstrate performance difference.")
        print(
            """
Now, let's do a performance test. We'll download the same object from each bucket
 'downloads' times
and compare the total time needed. Note: the performance difference will be much
 more pronounced if this
example is run in an EC2 instance in the same Availability Zone as the bucket.
"""
        )
        downloads = 1000
        print(
            f"The number of downloads of the same object for this example is set
 at {downloads}."
        )
        if q.ask("Would you like to download a different number? (y/n) ",
 q.is_yesno):
            max_downloads = 1000000
            downloads = q.ask(
                f"Enter a number between 1 and {max_downloads} for the number of
 downloads: ",
                q.is_int,
                q.in_range(1, max_downloads),
            )
        # Download the object 'downloads' times from each bucket and time it to
 demonstrate the speed difference.
        print("Downloading from the Directory bucket.")
        directory_time_start = time.time_ns()

        for index in range(downloads):
```

```
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")

            self.s3_express_wrapper.get_object(
                self.directory_bucket_name, bucket_object
            )

        directory_time_difference = time.time_ns() - directory_time_start
        print("Downloading from the normal bucket.")
        normal_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")
            self.s3_regular_wrapper.get_object(self.regular_bucket_name,
 bucket_object)

        normal_time_difference = time.time_ns() - normal_time_start
        print(
            f"The directory bucket took {directory_time_difference} nanoseconds,
 while the normal bucket took {normal_time_difference}."
        )
        difference = normal_time_difference - directory_time_difference
        print(f"That's a difference of {difference} nanoseconds, or")
        print(f"{(difference) / 1000000000} seconds.")
        if difference < 0:
            print(
                "The directory buckets were slower. This can happen if you are
 not running on the cloud within a vpc."
            )
        press_enter_to_continue()

    def show_lexicographical_differences(self, bucket_object: str) -> None:
        """
        Show the lexicographical difference between Directory buckets and regular
 buckets.
        This is done by creating a few objects in each bucket and listing them to
 show the difference.
        :param bucket_object: The object to use for the listing operations.
        """
        print(
            """
 7. Populate the buckets to show the lexicographical difference.
```

```
Now let's explore how Directory buckets store objects in a different manner to
 regular buckets. The key is in the name
"Directory". Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual
directories/folders. This allows for more rapid indexing, traversing, and
 therefore retrieval times! The more segmented
your bucket is, with lots of directories, sub-directories, and objects, the more
 efficient it becomes. This structural
difference also causes ListObjects to behave differently, which can cause
 unexpected results. Let's add a few more
objects with layered directories to see how the output of ListObjects changes.
        """
        )
        press_enter_to_continue()
        # Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        other_object = f"other/{bucket_object}"
        alt_object = f"alt/{bucket_object}"
        other_alt_object = f"other/alt/{bucket_object}"
        self.s3_regular_wrapper.put_object(self.regular_bucket_name,
 other_object, "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
 other_object, "")
        self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
 "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
 alt_object, "")
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, other_alt_object, ""
        )
        self.s3_express_wrapper.put_object(
            self.directory_bucket_name, other_alt_object, ""
        )
        directory_bucket_objects = self.s3_express_wrapper.list_objects(
            self.directory_bucket_name
        )

        regular_bucket_objects = self.s3_regular_wrapper.list_objects(
            self.regular_bucket_name
        )

        print("Directory bucket content")
        for bucket_object in directory_bucket_objects:
            print(f"   {bucket_object['Key']}")
```

```
        print("Normal bucket content")
        for bucket_object in regular_bucket_objects:
            print(f"   {bucket_object['Key']}")
        print(
            """
Notice how the normal bucket lists objects in lexicographical order, while the
 directory bucket does not. This is
because the normal bucket considers the whole "key" to be the object identifier,
 while the directory bucket actually
creates directories and uses the object "key" as a path to the object.
            """
        )
        press_enter_to_continue()

    def cleanup(self) -> None:
        """
        Delete resources created by this scenario.
        """
        if self.directory_bucket_name is not None:
            self.s3_express_wrapper.delete_bucket_and_objects(
                self.directory_bucket_name
            )
            print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
            self.directory_bucket_name = None

        if self.regular_bucket_name is not None:

 self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
            print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
            self.regular_bucket_name = None

        if self.stack is not None:
            self.destroy_cloudformation_stack(self.stack)
            self.stack = None

        self.tear_done_vpc()

    def create_access_key(self, user_name: str) -> dict[str, any]:
        """
        Creates an access key for the user.
        :param user_name: The name of the user.
        :return: The access key for the user.
        """
        try:
```

```
            access_key = self.iam_client.create_access_key(UserName=user_name)
            return access_key["AccessKey"]
        except ClientError as client_error:
            logging.error(
                "Couldn't create the access key. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def create_s3__client_with_access_key_credentials(
        self, access_key: dict[str, any]
    ) -> client:
        """
        Creates an S3 client with access key credentials.
        :param access_key: The access key for the user.
        :return: The S3 Express One Zone client.
        """
        try:
            s3_express_client = boto3.client(
                "s3",
                aws_access_key_id=access_key["AccessKeyId"],
                aws_secret_access_key=access_key["SecretAccessKey"],
                region_name=self.region,
            )
            return s3_express_client
        except ClientError as client_error:
            logging.error(
                "Couldn't create the S3 Express One Zone client. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def select_availability_zone_id(self, region: str) -> dict[str, any]:
        """
        Selects an availability zone.
        :param region: The region to select the availability zone from.
        :return: The availability zone dictionary.
        """
        try:
            response = self.ec2_client.describe_availability_zones(
                Filters=[{"Name": "region-name", "Values": [region]}]
            )
            availability_zones = response["AvailabilityZones"]
            zone_names = [zone["ZoneName"] for zone in availability_zones]
```

```
            index = q.choose("Select an availability zone: ", zone_names)
            return availability_zones[index]
        except ClientError as client_error:
            logging.error(
                "Couldn't describe availability zones. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `cfn_template.yaml` AWS CloudFormation script
and are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.
```

```
        :param stack: The CloudFormation stack that manages the example
resources.
        """
        try:
            print(
                f"CloudFormation stack '{stack.name}' is being deleted. This may
take a few minutes."
            )
            stack.delete()
            waiter = self.cloud_formation_resource.meta.client.get_waiter(
                "stack_delete_complete"
            )
            waiter.wait(StackName=stack.name)
            print(f"CloudFormation stack '{stack.name}' has been deleted.")
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the CloudFormation stack. Here's why: %s",
                client_error.response["Error"]["Message"],
            )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
"s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:
            logging.error(
```

```
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
            )
            route_table_id = response["RouteTables"][0]["RouteTableId"]
            service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

            response = self.ec2_client.create_vpc_endpoint(
                VpcId=self.vpc_id,
                RouteTableIds=[route_table_id],
                ServiceName=service_name,
            )
            self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
            print(f"Created vpc endpoint {self.vpc_endpoint_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc endpoint. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def tear_done_vpc(self) -> None:
        if self.vpc_endpoint_id is not None:
            try:
                self.ec2_client.delete_vpc_endpoints(
                    VpcEndpointIds=[self.vpc_endpoint_id]
                )
                print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
                self.vpc_endpoint_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc endpoint %s. Here's why: %s",
                    self.vpc_endpoint_id,
                    client_error.response["Error"]["Message"],
                )
        if self.vpc_id is not None:
            try:
                self.ec2_client.delete_vpc(VpcId=self.vpc_id)
```

```
                print(f"Deleted vpc {self.vpc_id}")
                self.vpc_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc %s. Here's why: %s",
                    self.vpc_id,
                    client_error.response["Error"]["Message"],
                )
```

Uma classe de wrapper para funções de SDK do Amazon S3 Express.

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
 interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                              access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def create_bucket(
        self, bucket_name: str, bucket_configuration: dict[str, any] = None
    ) -> None:
```

```
        """
        Creates a bucket.
        :param bucket_name: The name of the bucket.
        :param bucket_configuration: The optional configuration for the bucket.
        """
        try:
            params = {"Bucket": bucket_name}
            if bucket_configuration:
                params["CreateBucketConfiguration"] = bucket_configuration

            self.s3_client.create_bucket(**params)
        except ClientError as client_error:
            # Do not log InvalidBucketName error because it is logged elsewhere.
            if client_error.response["Error"]["Code"] != "InvalidBucketName":
                logging.error(
                    "Couldn't create the bucket %s. Here's why: %s",
                    bucket_name,
                    client_error.response["Error"]["Message"],
                )
            raise

    def delete_bucket_and_objects(self, bucket_name: str) -> None:
        """
        Deletes a bucket and its objects.
         :param bucket_name: The name of the bucket.
        """
        try:
            # Delete the objects in the bucket first. This is required for a
bucket to be deleted.
            paginator = self.s3_client.get_paginator("list_objects_v2")
            page_iterator = paginator.paginate(Bucket=bucket_name)
            for page in page_iterator:
                if "Contents" in page:
                    delete_keys = {
                        "Objects": [{"Key": obj["Key"]} for obj in
page["Contents"]]
                    }
                    response = self.s3_client.delete_objects(
                        Bucket=bucket_name, Delete=delete_keys
                    )
                    if "Errors" in response:
                        for error in response["Errors"]:
                            logging.error(
                                "Couldn't delete object %s. Here's why: %s",
```

```
                                error["Key"],
                                error["Message"],
                            )

            self.s3_client.delete_bucket(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )

    def put_object(self, bucket_name: str, object_key: str, content: str) ->
None:
        """
        Puts an object into a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        :param content: The content of the object.
        """
        try:
            self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't put the object %s into bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

    def list_objects(self, bucket: str) -> list[str]:
        """
        Lists objects in a bucket.
        :param bucket: The name of the bucket.
        :return: The list of objects in the bucket.
        """
        try:
            response = self.s3_client.list_objects_v2(Bucket=bucket)
            return response.get("Contents", [])
        except ClientError as client_error:
            logging.error(
                "Couldn't list objects in bucket %s. Here's why: %s",
```

```
                bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def copy_object(
        self,
        source_bucket: str,
        source_key: str,
        destination_bucket: str,
        destination_key: str,
    ) -> None:
        """
        Copies an object from one bucket to another.
        :param source_bucket: The source bucket.
        :param source_key: The source key.
        :param destination_bucket: The destination bucket.
        :param destination_key: The destination key.
        :return: None
        """
        try:
            self.s3_client.copy_object(
                CopySource={"Bucket": source_bucket, "Key": source_key},
                Bucket=destination_bucket,
                Key=destination_key,
            )
        except ClientError as client_error:
            logging.error(
                "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",
                source_key,
                source_bucket,
                destination_bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
```

```
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why:
%s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise


    def get_object(self, bucket_name: str, object_key: str) -> None:
        """
        Gets an object from a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        """
        try:
            self.s3_client.get_object(Bucket=bucket_name, Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't get the object %s from bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Ações para buckets do S3 Directory usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do S3 Directory Buckets com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API de buckets de diretório do S3 e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para buckets do S3 Directory usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API de buckets de diretório do Amazon S3.

Exemplos

- Use AbortMultipartUploadcom um AWS SDK
- Use CompleteMultipartUploadcom um AWS SDK
- Use CopyObjectcom um AWS SDK
- Use CreateBucketcom um AWS SDK
- Use CreateMultipartUploadcom um AWS SDK
- Use CreateSessioncom um AWS SDK
- Use DeleteBucketcom um AWS SDK
- Use DeleteBucketEncryptioncom um AWS SDK
- Use DeleteBucketPolicycom um AWS SDK
- Use DeleteObjectcom um AWS SDK
- Use DeleteObjectscom um AWS SDK
- Use GetBucketEncryptioncom um AWS SDK
- Use GetBucketPolicycom um AWS SDK
- Use GetObjectcom um AWS SDK
- Use GetObjectAttributescom um AWS SDK
- Use HeadBucketcom um AWS SDK
- Use HeadObjectcom um AWS SDK
- Use ListDirectoryBucketscom um AWS SDK
- Use ListMultipartUploadscom um AWS SDK

- Use ListObjectsV2 com um AWS SDK
- Use ListPartscom um AWS SDK
- Use PutBucketEncryptioncom um AWS SDK
- Use PutBucketPolicycom um AWS SDK
- Use PutObjectcom um AWS SDK
- Use UploadPartcom um AWS SDK
- Use UploadPartCopycom um AWS SDK

Use **`AbortMultipartUpload`**com um AWS SDK

O código de exemplo a seguir mostra como usar `AbortMultipartUpload`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Anular um upload fracionado em um bucket de diretórios.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
    /**
     * Aborts a specific multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID of the multipart upload to abort
     * @return True if the multipart upload is successfully aborted, false
otherwise
     */
    public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName,
            String objectKey, String uploadId) {
        logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
bucketName);
        try {
            // Abort the multipart upload
            AbortMultipartUploadRequest abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .build();

            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            logger.info("Aborted multipart upload: {} for object: {}", uploadId,
objectKey);
            return true;
        } catch (S3Exception e) {
            logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Para obter detalhes da API, consulte AbortMultipartUploada Referência AWS SDK for Java 2.x da API.

Use **`CompleteMultipartUpload`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CompleteMultipartUpload`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Concluir um upload fracionado em um bucket de diretórios.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * This method completes the multipart upload request by collating all the
     * upload parts.
```

```
     *
     * @param s3Client    The S3 client used to interact with S3
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to be uploaded
     * @param uploadId    The upload ID used to track the multipart upload
     * @param uploadParts The list of completed parts
     * @return True if the multipart upload is successfully completed, false
     *         otherwise
     */
    public static boolean completeDirectoryBucketMultipartUpload(S3Client
s3Client, String bucketName, String objectKey,
            String uploadId, List<CompletedPart> uploadParts) {
        try {
            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                    .parts(uploadParts)
                    .build();
            CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse response =
s3Client.completeMultipartUpload(completeMultipartUploadRequest);
            logger.info("Multipart upload completed. ETag: {}", response.eTag());
            return true;
        } catch (S3Exception e) {
            logger.error("Failed to complete multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CompleteMultipartUploada Referência AWS SDK for Java 2.x da API.

Use **`CopyObject`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CopyObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie um objeto de um bucket de diretório para um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;
```

```
    /**
     * Copies an object from one S3 general purpose bucket to one S3 directory
     * bucket.
     *
     * @param s3Client     The S3 client used to interact with S3
     * @param sourceBucket The name of the source bucket
     * @param objectKey    The key (name) of the object to be copied
     * @param targetBucket The name of the target bucket
     */
    public static void copyDirectoryBucketObject(S3Client s3Client, String
sourceBucket, String objectKey,
            String targetBucket) {
        logger.info("Copying object: {} from bucket: {} to bucket: {}",
objectKey, sourceBucket, targetBucket);

        try {
            // Create a CopyObjectRequest
            CopyObjectRequest copyReq = CopyObjectRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(objectKey)
                    .destinationBucket(targetBucket)
                    .destinationKey(objectKey)
                    .build();

            // Copy the object
            CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
            logger.info("Successfully copied {} from bucket {} into bucket {}.
CopyObjectResponse: {}",
                    objectKey, sourceBucket, targetBucket,
copyRes.copyObjectResult().toString());

        } catch (S3Exception e) {
            logger.error("Failed to copy object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CopyObjecta Referência AWS SDK for Java 2.x da API.

Use **`CreateBucket`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateBucket`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket de diretório do S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
```

```
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CreateBucketa Referência AWS SDK for Java 2.x da API.

Use **`CreateMultipartUpload`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateMultipartUpload`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um upload fracionado em um bucket de diretório.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * This method creates a multipart upload request that generates a unique
 upload
     * ID used to track
     * all the upload parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @return The upload ID used to track the multipart upload
     */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
```

```
            CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;

        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CreateMultipartUploada Referência AWS SDK for Java 2.x da API.

Use **`CreateSession`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateSession`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class S3ExpressWrapper:
```

```
    """Encapsulates Amazon S3 Express One Zone actions using the client
 interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                              access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why:
 %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateSessionReferência da API AWS SDK for Python (Boto3).

Use **`DeleteBucket`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteBucket`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um bucket de diretório do S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to delete
     */
```

```
    public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
        logger.info("Deleting bucket: {}", bucketName);

        try {
            // Create a DeleteBucketRequest
            DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket
            s3Client.deleteBucket(deleteBucketRequest);
            logger.info("Successfully deleted bucket: {}", bucketName);

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteBucketa Referência AWS SDK for Java 2.x da API.

Use **`DeleteBucketEncryption`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteBucketEncryption`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a configuração de criptografia de um bucket de diretório.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Deletes the encryption configuration from an S3 bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        DeleteBucketEncryptionRequest deleteRequest =
 DeleteBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3Client.deleteBucketEncryption(deleteRequest);
            logger.info("Bucket encryption deleted for bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket encryption: {} - Error code:
 {}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteBucketEncryptiona Referência AWS SDK for Java 2.x da API.

Use **`DeleteBucketPolicy`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua uma política de um bucket de diretório.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


    /**
     * Deletes the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Deleting policy for bucket: {}", bucketName);

        try {
            // Create a DeleteBucketPolicyRequest
            DeleteBucketPolicyRequest deletePolicyReq =
 DeleteBucketPolicyRequest.builder()
```

```
                    .bucket(bucketName)
                    .build();

            // Delete the bucket policy
            s3Client.deleteBucketPolicy(deletePolicyReq);
            logger.info("Successfully deleted bucket policy");

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteBucketPolicya Referência AWS SDK for Java 2.x da API.

Use **`DeleteObject`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um objeto em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;




    /**
     * Deletes an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be deleted
     */
    public static void deleteDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Deleting object: {} from bucket: {}", objectKey,
 bucketName);

        try {
            // Create a DeleteObjectRequest
            DeleteObjectRequest deleteObjectRequest =
 DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Delete the object
            s3Client.deleteObject(deleteObjectRequest);
            logger.info("Object {} has been deleted", objectKey);

        } catch (S3Exception e) {
```

```
            logger.error("Failed to delete object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjecta Referência AWS SDK for Java 2.x da API.

Use **`DeleteObjects`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteObjects`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua vários objetos em um bucket de diretório.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Deletes multiple objects from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKeys The list of keys (names) of the objects to be deleted
     */
    public static void deleteDirectoryBucketObjects(S3Client s3Client, String
 bucketName, List<String> objectKeys) {
        logger.info("Deleting objects from bucket: {}", bucketName);

        try {
            // Create a list of ObjectIdentifier.
            List<ObjectIdentifier> identifiers = objectKeys.stream()
                    .map(key -> ObjectIdentifier.builder().key(key).build())
                    .toList();

            Delete delete = Delete.builder()
                    .objects(identifiers)
                    .build();

            DeleteObjectsRequest deleteObjectsRequest =
 DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(delete)
                    .build();

            DeleteObjectsResponse deleteObjectsResponse =
 s3Client.deleteObjects(deleteObjectsRequest);
            deleteObjectsResponse.deleted().forEach(deleted ->
 logger.info("Deleted object: {}", deleted.key()));

        } catch (S3Exception e) {
```

```
            logger.error("Failed to delete objects: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjectsa Referência AWS SDK for Java 2.x da API.

Use **`GetBucketEncryption`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetBucketEncryption`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a configuração de criptografia de um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
    /**
     * Retrieves the encryption configuration for an S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The type of server-side encryption applied to the bucket (e.g.,
     *         AES256, aws:kms)
     */
    public static String getDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
        try {
            // Create a GetBucketEncryptionRequest
            GetBucketEncryptionRequest getRequest =
GetBucketEncryptionRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket encryption configuration
            GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
            ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

            String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
            logger.info("Bucket encryption algorithm: {}", encryptionType);
            logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
            logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

            return encryptionType;
        } catch (S3Exception e) {
            logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte GetBucketEncryptiona Referência AWS SDK for Java 2.x da API.

Use **`GetBucketPolicy`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política de um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

    /**
     * Retrieves the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The bucket policy text
     */
    public static String getDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Getting policy for bucket: {}", bucketName);
```

```
        try {
            // Create a GetBucketPolicyRequest
            GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket policy
            GetBucketPolicyResponse response =
s3Client.getBucketPolicy(policyReq);

            // Print and return the policy text
            String policyText = response.policy();
            logger.info("Bucket policy: {}", policyText);
            return policyText;

        } catch (S3Exception e) {
            logger.error("Failed to get bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte GetBucketPolicya Referência AWS SDK for Java 2.x da API.

Use **`GetObject`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Criar um URL pré-assinado para obter um objeto

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um objeto de um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be retrieved
     * @return The retrieved object as a ResponseInputStream
     */
```

```
   public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
       logger.info("Retrieving object: {} from bucket: {}", objectKey,
bucketName);

       try {
           // Create a GetObjectRequest
           GetObjectRequest objectRequest = GetObjectRequest.builder()
                   .key(objectKey)
                   .bucket(bucketName)
                   .build();

           // Retrieve the object as bytes
           ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
           byte[] data = objectBytes.asByteArray();

           // Print object contents to console
           String objectContent = new String(data, StandardCharsets.UTF_8);
           logger.info("Object contents: \n{}", objectContent);

           return true;

       } catch (S3Exception e) {
           logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode(), e);
           return false;
       }
   }
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK for Java 2.x da API.

Use **`GetObjectAttributes`**com um AWS SDK

O código de exemplo a seguir mostra como usar `GetObjectAttributes`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha atributos de um objeto de um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves attributes for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve attributes for
     * @return True if the object attributes are successfully retrieved, false
     *         otherwise
     */
```

```
   public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
String bucketName, String objectKey) {
       logger.info("Retrieving attributes for object: {} from bucket: {}",
objectKey, bucketName);

       try {
           // Create a GetObjectAttributesRequest
           GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                           ObjectAttributes.OBJECT_SIZE)
                   .build();

           // Retrieve the object attributes
           GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
           logger.info("Attributes for object {}:", objectKey);
           logger.info("ETag: {}", response.eTag());
           logger.info("Storage Class: {}", response.storageClass());
           logger.info("Object Size: {}", response.objectSize());
           return true;

       } catch (S3Exception e) {
           logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
                   e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
           return false;
       }
   }
```

- Para obter detalhes da API, consulte GetObjectAttributesa Referência AWS SDK for Java 2.x da API.

Use **`HeadBucket`**com um AWS SDK

O código de exemplo a seguir mostra como usar `HeadBucket`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Verifica se o bucket de diretório do S3 especificado existe e está acessível.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Checks if the specified S3 directory bucket exists and is accessible.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to check
     * @return True if the bucket exists and is accessible, false otherwise
     */
    public static boolean headDirectoryBucket(S3Client s3Client, String
 bucketName) {
        logger.info("Checking if bucket exists: {}", bucketName);

        try {
            // Create a HeadBucketRequest
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            // If the bucket doesn't exist, the following statement throws
 NoSuchBucketException,
            // which is a subclass of S3Exception.
```

```
            s3Client.headBucket(headBucketRequest);
            logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to access bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte HeadBucketa Referência AWS SDK for Java 2.x da API.

Use **`HeadObject`**com um AWS SDK

O código de exemplo a seguir mostra como usar `HeadObject`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha metadados de um objeto em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves metadata for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve metadata for
     * @return True if the object exists, false otherwise
     */
    public static boolean headDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Retrieving metadata for object: {} from bucket: {}",
 objectKey, bucketName);

        try {
            // Create a HeadObjectRequest
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Retrieve the object metadata
            HeadObjectResponse response = s3Client.headObject(headObjectRequest);
            logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with
 ETag: \"{}\"", objectKey, bucketName,
                    response.eTag());
            logger.info("Content-Type: {}", response.contentType());
            logger.info("Content-Length: {}", response.contentLength());
            logger.info("Last Modified: {}", response.lastModified());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object metadata: {} - Error code:
 {}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
```

```
            return false;
        }
    }
```

- Para obter detalhes da API, consulte HeadObjecta Referência AWS SDK for Java 2.x da API.

Use **`ListDirectoryBuckets`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListDirectoryBuckets`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todos os buckets de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
    /**
    * Lists all S3 directory buckets and no general purpose buckets.
    *
    * @param s3Client The S3 client used to interact with S3
    * @return A list of bucket names
    */
   public static List<String> listDirectoryBuckets(S3Client s3Client) {
       logger.info("Listing all directory buckets");

       try {
           // Create a ListBucketsRequest
           ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

           // Retrieve the list of buckets
           ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

           // Extract bucket names
           List<String> bucketNames = response.buckets().stream()
                   .map(Bucket::name)
                   .collect(Collectors.toList());

           return bucketNames;
       } catch (S3Exception e) {
           logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode());
           throw e;
       }
   }
```

- Para obter detalhes da API, consulte ListDirectoryBucketsa Referência AWS SDK for Java 2.x da API.

Use **`ListMultipartUploads`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListMultipartUploads`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os uploads fracionados em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * Lists multipart uploads for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
```

```
    * @return A list of MultipartUpload objects representing the multipart
uploads
    */
   public static List<MultipartUpload>
listDirectoryBucketMultipartUploads(S3Client s3Client, String bucketName) {
       logger.info("Listing in-progress multipart uploads for bucket: {}",
bucketName);

       try {
           // Create a ListMultipartUploadsRequest
           ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                   .bucket(bucketName)
                   .build();

           // List the multipart uploads
           ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
           List<MultipartUpload> uploads = response.uploads();
           for (MultipartUpload upload : uploads) {
               logger.info("In-progress multipart upload: Upload ID: {}, Key:
{}, Initiated: {}", upload.uploadId(),
                       upload.key(), upload.initiated());
           }
           return uploads;

       } catch (S3Exception e) {
           logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode());
           return List.of(); // Return an empty list if an exception is thrown
       }
   }
```

- Para obter detalhes da API, consulte ListMultipartUploadsa Referência AWS SDK for Java 2.x da API.

Use **`ListObjectsV2`** com um AWS SDK

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os objetos em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Lists objects in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of object keys in the bucket
     */
```

```
   public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
String bucketName) {
       logger.info("Listing objects in bucket: {}", bucketName);

       try {
           // Create a ListObjectsV2Request
           ListObjectsV2Request listObjectsV2Request =
ListObjectsV2Request.builder()
                   .bucket(bucketName)
                   .build();

           // Retrieve the list of objects
           ListObjectsV2Response response =
s3Client.listObjectsV2(listObjectsV2Request);

           // Extract and return the object keys
           return response.contents().stream()
                   .map(S3Object::key)
                   .collect(Collectors.toList());

       } catch (S3Exception e) {
           logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode());
           throw e;
       }
   }
```

- Para obter detalhes da API, consulte ListObjectsV2 na Referência AWS SDK for Java 2.x da API.

Use **`ListParts`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListParts`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as partes de um upload fracionado em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Lists the parts of a multipart upload for the specified S3 directory
 bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
```

```
     * @param objectKey  The key (name) of the object being uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @return A list of Part representing the parts of the multipart upload
     */
    public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
s3Client, String bucketName,
            String objectKey, String uploadId) {
        logger.info("Listing parts for object: {} in bucket: {}", objectKey,
bucketName);

        try {
            // Create a ListPartsRequest
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                    .bucket(bucketName)
                    .uploadId(uploadId)
                    .key(objectKey)
                    .build();

            // List the parts of the multipart upload
            ListPartsResponse response = s3Client.listParts(listPartsRequest);
            List<Part> parts = response.parts();
            for (Part part : parts) {
                logger.info("Uploaded part: Part number = \"{}\", etag = {}",
part.partNumber(), part.eTag());
            }
            return parts;

        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- Para obter detalhes da API, consulte ListPartsa Referência AWS SDK for Java 2.x da API.

Use **`PutBucketEncryption`**com um AWS SDK

O código de exemplo a seguir mostra como usar `PutBucketEncryption`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a criptografia em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import
 software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

    /**
     * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param kmsKeyId   The ID of the customer-managed KMS key
     */
    public static void putDirectoryBucketEncryption(S3Client s3Client, String
 bucketName, String kmsKeyId) {
        // Define the default encryption configuration to use SSE-KMS. For
 directory
```

```
        // buckets, AWS managed KMS keys aren't supported. Only customer-managed
keys
        // are supported.
        ServerSideEncryptionByDefault encryptionByDefault =
ServerSideEncryptionByDefault.builder()
                .sseAlgorithm(ServerSideEncryption.AWS_KMS)
                .kmsMasterKeyID(kmsKeyId)
                .build();

        // Create a server-side encryption rule to apply the default encryption
        // configuration. For directory buckets, the bucketKeyEnabled field is
enforced
        // to be true.
        ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .applyServerSideEncryptionByDefault(encryptionByDefault)
                .build();

        // Create the server-side encryption configuration for the bucket
        ServerSideEncryptionConfiguration encryptionConfiguration =
ServerSideEncryptionConfiguration.builder()
                .rules(rule)
                .build();

        // Create the PutBucketEncryption request
        PutBucketEncryptionRequest putRequest =
PutBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .serverSideEncryptionConfiguration(encryptionConfiguration)
                .build();

        // Set the bucket encryption
        try {
            s3Client.putBucketEncryption(putRequest);
            logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte PutBucketEncryptiona Referência AWS SDK for Java 2.x da API.

Use **`PutBucketPolicy`**com um AWS SDK

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Aplique uma política a um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

    /**
     * Sets the following bucket policy for the specified S3 directory bucket.
     *<pre>
     * {
     *     "Version":"2012-10-17",
     *     "Statement": [
     *         {
     *             "Sid": "AdminPolicy",
```

```
     *             "Effect": "Allow",
     *             "Principal": {
     *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
     *             },
     *             "Action": "s3express:*",
     *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
     *         }
     *     ]
     * }
     * </pre>
     * This policy grants all S3 directory bucket actions to identities in the
 same account as the bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param policyText The policy text to be applied
     */
    public static void putDirectoryBucketPolicy(S3Client s3Client, String
 bucketName, String policyText) {
        logger.info("Setting policy on bucket: {}", bucketName);
        logger.info("Policy: {}", policyText);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3Client.putBucketPolicy(policyReq);
            logger.info("Bucket policy set successfully!");

        } catch (S3Exception e) {
            logger.error("Failed to set bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte PutBucketPolicya Referência AWS SDK for Java 2.x da API.

Use **`PutObject`**com um AWS SDK

O código de exemplo a seguir mostra como usar `PutObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloque um objeto em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
```

```
     * Puts an object into the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be placed in the bucket
     * @param filePath   The path of the file to be uploaded
     */
    public static void putDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey, Path filePath) {
        logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

        try {
            // Create a PutObjectRequest
            PutObjectRequest putObj = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Upload the object
            s3Client.putObject(putObj, filePath);
            logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

        } catch (UncheckedIOException e) {
            throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
                    .awsErrorDetails(AwsErrorDetails.builder()
                            .errorCode("ClientSideException:FailedToReadFile")
                            .errorMessage(e.getMessage())
                            .build())
                    .build();
        } catch (S3Exception e) {
            logger.error("Failed to put object: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte PutObjecta Referência AWS SDK for Java 2.x da API.

Use **`UploadPart`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UploadPart`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de uma parte de um upload fracionado em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * This method creates part requests and uploads individual parts to S3.
```

```
    * While it uses the UploadPart API to upload a single part, it does so
    * sequentially to handle multiple parts of a file, returning all the
completed
    * parts.
    *
    * @param s3Client   The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKey  The key (name) of the object to be uploaded
    * @param uploadId   The upload ID used to track the multipart upload
    * @param filePath   The path to the file to be uploaded
    * @return A list of uploaded parts
    * @throws IOException if an I/O error occurs
    */
   public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
           String objectKey, String uploadId, Path filePath) throws IOException
{
       logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

       int partNumber = 1;
       List<CompletedPart> uploadedParts = new ArrayList<>();
       ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

       // Read the local file, break down into chunks and process
       try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(),
"r")) {
           long fileSize = file.length();
           int position = 0;

           // Sequentially upload parts of the file
           while (position < fileSize) {
               file.seek(position);
               int read = file.getChannel().read(bb);

               bb.flip(); // Swap position and limit before reading from the
buffer
               UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                       .bucket(bucketName)
                       .key(objectKey)
                       .uploadId(uploadId)
                       .partNumber(partNumber)
                       .build();
```

```
                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                // Build the uploaded part
                CompletedPart uploadedPart = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();

                // Add the uploaded part to the list
                uploadedParts.add(uploadedPart);

                // Log to indicate the part upload is done
                logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
        return uploadedParts;
    }
```

- Para obter detalhes da API, consulte UploadPart a Referência AWS SDK for Java 2.x da API.

Use **`UploadPartCopy`** com um AWS SDK

O código de exemplo a seguir mostra como usar `UploadPartCopy`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie partes de cópia com base no tamanho do objeto de origem e copie partes individuais para um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
 com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Creates copy parts based on source object size and copies over individual
     * parts.
     *
     * @param s3Client          The S3 client used to interact with S3
     * @param sourceBucket      The name of the source bucket
     * @param sourceKey         The key (name) of the source object
     * @param destinationBucket The name of the destination bucket
     * @param destinationKey    The key (name) of the destination object
     * @param uploadId          The upload ID used to track the multipart upload
     * @return A list of completed parts
     */
    public static List<CompletedPart>
 multipartUploadCopyForDirectoryBucket(S3Client s3Client, String sourceBucket,
            String sourceKey, String destinationBucket, String destinationKey,
 String uploadId) {
        // Get the object size to track the end of the copy operation
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(sourceBucket)
                .key(sourceKey)
                .build();
        HeadObjectResponse headObjectResponse =
 s3Client.headObject(headObjectRequest);
        long objectSize = headObjectResponse.contentLength();

        logger.info("Source Object size: {}", objectSize);

        // Copy the object using 20 MB parts
        long partSize = 20 * 1024 * 1024; // 20 MB
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
 1);
            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
 partNum, bytePosition, lastByte);

            try {
```

```
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                        .sourceBucket(sourceBucket)
                        .sourceKey(sourceKey)
                        .destinationBucket(destinationBucket)
                        .destinationKey(destinationKey)
                        .uploadId(uploadId)
                        .copySourceRange("bytes=" + bytePosition + "-" +
lastByte)
                        .partNumber(partNum)
                        .build();
                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNum)
                        .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                        .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code:
{}", partNum,
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
                throw e;
            }
        }

        return uploadedParts;
    }
```

- Para obter detalhes da API, consulte UploadPartCopya Referência AWS SDK for Java 2.x da API.

Cenários para buckets do S3 Directory usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns em buckets do diretório S3 com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando

várias funções em buckets de diretório do S3 ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Crie uma URL pré-assinada para buckets de diretório do Amazon S3 para obter um objeto usando um AWS SDK

Crie uma URL pré-assinada para buckets de diretório do Amazon S3 para obter um objeto usando um AWS SDK

O exemplo de código a seguir mostra como criar um URL pré-assinado para buckets de diretório do S3 e obter um objeto.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere um URL GET pré-assinado para acessar um objeto em um bucket de diretório do S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
```

```
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Generates a presigned URL for accessing an object in the specified S3
     * directory bucket.
     *
     * @param s3Presigner The S3 presigner client used to generate the presigned
 URL
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to access
     * @return A presigned URL for accessing the specified object
     */
    public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
 s3Presigner, String bucketName,
            String objectKey) {
        logger.info("Generating presigned URL for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Create a GetObjectPresignRequest
            GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
```

```
                    .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
                    .getObjectRequest(getObjectRequest)
                    .build();

            // Generate the presigned URL
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte GetObjecta Referência AWS SDK for Java 2.x da API.

Exemplos de código para uso de SageMaker IA AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon SageMaker AI com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- SageMaker Guia do desenvolvedor de IA — Mais informações sobre SageMaker IA.

- SageMaker Referência da API de IA — Detalhes sobre todas as ações de SageMaker IA disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de uso de SageMaker IA AWS SDKs
  - Olá SageMaker AI
  - Ações para uso de SageMaker IA AWS SDKs
    - Use CreateEndpointcom um AWS SDK
    - Use CreateModelcom um AWS SDK
    - Use CreatePipelinecom um AWS SDK
    - Use CreateTrainingJobcom um AWS SDK
    - Use CreateTransformJobcom um AWS SDK
    - Use DeleteEndpointcom um AWS SDK
    - Use DeleteModelcom um AWS SDK
    - Use DeletePipelinecom um AWS SDK
    - Use DescribePipelineExecutioncom um AWS SDK
    - Use DescribeTrainingJobcom um AWS SDK
    - Use ListAlgorithmscom um AWS SDK
    - Use ListModelscom um AWS SDK
    - Use ListNotebookInstancescom um AWS SDK
    - Use ListTrainingJobscom um AWS SDK
    - Use StartPipelineExecutioncom um AWS SDK
    - Use UpdatePipelinecom um AWS SDK
- Cenários para uso de SageMaker IA AWS SDKs
  - Comece a usar trabalhos geoespaciais de SageMaker IA em um pipeline usando um AWS SDK
  - Comece a usar modelos e endpoints de SageMaker IA usando um AWS SDK

  - Conceitos básicos de arquivos de atributos de machine learning

Exemplos básicos de uso de SageMaker IA AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos da Amazon SageMaker AI com AWS SDKs.

Exemplos

- Olá SageMaker AI
- Ações para uso de SageMaker IA AWS SDKs
  - Use CreateEndpointcom um AWS SDK
  - Use CreateModelcom um AWS SDK
  - Use CreatePipelinecom um AWS SDK
  - Use CreateTrainingJobcom um AWS SDK
  - Use CreateTransformJobcom um AWS SDK
  - Use DeleteEndpointcom um AWS SDK
  - Use DeleteModelcom um AWS SDK
  - Use DeletePipelinecom um AWS SDK
  - Use DescribePipelineExecutioncom um AWS SDK
  - Use DescribeTrainingJobcom um AWS SDK
  - Use ListAlgorithmscom um AWS SDK
  - Use ListModelscom um AWS SDK
  - Use ListNotebookInstancescom um AWS SDK
  - Use ListTrainingJobscom um AWS SDK
  - Use StartPipelineExecutioncom um AWS SDK
  - Use UpdatePipelinecom um AWS SDK

Olá SageMaker AI

Os exemplos de código a seguir mostram como começar a usar a SageMaker IA.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.SageMaker;
using Amazon.SageMaker.Model;

namespace SageMakerActions;

public static class HelloSageMaker
{
    static async Task Main(string[] args)
    {
        var sageMakerClient = new AmazonSageMakerClient();

        Console.WriteLine($"Hello Amazon SageMaker! Let's list some of your
 notebook instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five notebook instances.
        var response = await sageMakerClient.ListNotebookInstancesAsync(
            new ListNotebookInstancesRequest()
            {
                MaxResults = 5
            });

        if (!response.NotebookInstances.Any())
        {
            Console.WriteLine($"No notebook instances found.");
            Console.WriteLine("See https://docs.aws.amazon.com/sagemaker/latest/
dg/howitworks-create-ws.html to create one.");
        }

        foreach (var notebookInstance in response.NotebookInstances)
```

```
        {
            Console.WriteLine($"\tInstance:
 {notebookInstance.NotebookInstanceName}");
            Console.WriteLine($"\tArn: {notebookInstance.NotebookInstanceArn}");
            Console.WriteLine($"\tCreation Date:
 {notebookInstance.CreationTime.ToShortDateString()}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte ListNotebookInstancesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();
```

```
        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
 sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
 notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
 item.notebookInstanceName());
            }

        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListNotebookInstancesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  SageMakerClient,
  ListNotebookInstancesCommand,
} from "@aws-sdk/client-sagemaker";
```

```
const client = new SageMakerClient({
  region: "us-west-2",
});

export const helloSagemaker = async () => {
  const command = new ListNotebookInstancesCommand({ MaxResults: 5 });

  const response = await client.send(command);
  console.log(
    "Hello Amazon SageMaker! Let's list some of your notebook instances:",
  );

  const instances = response.NotebookInstances || [];

  if (instances.length === 0) {
    console.log(
      "• No notebook instances found. Try creating one in the AWS Management
 Console or with the CreateNotebookInstanceCommand.",
    );
  } else {
    console.log(
      instances
        .map(
          (i) =>
            `• Instance: ${i.NotebookInstanceName}\n  Arn:${
              i.NotebookInstanceArn
            } \n  Creation Date: ${i.CreationTime.toISOString()}`,
        )
        .join("\n"),
    );
  }

  return response;
};
```

- Para obter detalhes da API, consulte ListNotebookInstancesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listBooks() {
    SageMakerClient.fromEnvironment { region = "us-west-2" }.use
 { sageMakerClient ->
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListNotebookInstancesreferência da API AWS SDK for Kotlin.

Ações para uso de SageMaker IA AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais de SageMaker IA com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API de SageMaker IA e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para uso de SageMaker IA AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon SageMaker AI API Reference.

Exemplos

- Use CreateEndpointcom um AWS SDK

- Use CreateModelcom um AWS SDK
- Use CreatePipelinecom um AWS SDK
- Use CreateTrainingJobcom um AWS SDK
- Use CreateTransformJobcom um AWS SDK
- Use DeleteEndpointcom um AWS SDK
- Use DeleteModelcom um AWS SDK
- Use DeletePipelinecom um AWS SDK
- Use DescribePipelineExecutioncom um AWS SDK
- Use DescribeTrainingJobcom um AWS SDK
- Use ListAlgorithmscom um AWS SDK
- Use ListModelscom um AWS SDK
- Use ListNotebookInstancescom um AWS SDK
- Use ListTrainingJobscom um AWS SDK
- Use StartPipelineExecutioncom um AWS SDK
- Use UpdatePipelinecom um AWS SDK

Use **`CreateEndpoint`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateEndpoint`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de usar modelos e endpoints

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
    DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
    DATA oo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.

    "Create a production variant as an ABAP object."
    "Identifies a model that you want to host and the resources chosen to deploy
 for hosting it."
    lo_production_variants = NEW #( iv_variantname = iv_variant_name
                                    iv_modelname = iv_model_name
                                    iv_initialinstancecount =
 iv_initial_instance_count
                                    iv_instancetype = iv_instance_type ).

    INSERT lo_production_variants INTO TABLE lt_production_variants.

    "Create an endpoint configuration."
    TRY.
        oo_ep_config_result = lo_sgm->createendpointconfig(
          iv_endpointconfigname = iv_endpoint_config_name
          it_productionvariants = lt_production_variants ).
        MESSAGE 'Endpoint configuration created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
 'E'.
    ENDTRY.

    "Create an endpoint."
    TRY.
        oo_result = lo_sgm->createendpoint(     " oo_result is returned for
 testing purposes. "
            iv_endpointconfigname = iv_endpoint_config_name
            iv_endpointname = iv_endpoint_name ).
        MESSAGE 'Endpoint created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateEndpointreferência da API AWS SDK for SAP ABAP.

Use **`CreateModel`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateModel`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de usar modelos e endpoints

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.

   "Create an ABAP object for the container image based on input variables."
   lo_primarycontainer = NEW #( iv_image = iv_container_image
                                iv_modeldataurl = iv_model_data_url ).

   "Create an Amazon SageMaker model."
   TRY.
       oo_result = lo_sgm->createmodel(        " oo_result is returned for
testing purposes. "
         iv_executionrolearn = iv_execution_role_arn
         iv_modelname = iv_model_name
         io_primarycontainer = lo_primarycontainer ).
       MESSAGE 'Model created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateModelreferência da API AWS SDK for SAP ABAP.

Use **`CreatePipeline`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreatePipeline`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de trabalhos geoespaciais e pipelines

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create a pipeline from a JSON definition, or update it if the pipeline
already exists.
   /// </summary>
   /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
   public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
string name, string description, string displayName)
   {
       try
       {
           var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
               new UpdatePipelineRequest()
               {
                   PipelineDefinition = pipelineJson,
                   PipelineDescription = description,
                   PipelineDisplayName = displayName,
                   PipelineName = name,
```

```
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }
```

- Para obter detalhes da API, consulte CreatePipelinea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();
```

```
        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest =
CreatePipelineRequest.builder()
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte CreatePipelinea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Uma função que cria um pipeline de SageMaker IA usando uma definição JSON fornecida localmente.

```
/**
 * Create the Amazon SageMaker pipeline using a JSON pipeline definition. The
 definition
 * can also be provided as an Amazon S3 object using
 PipelineDefinitionS3Location.
 * @param {{roleArn: string, name: string, sagemakerClient: import('@aws-sdk/
client-sagemaker').SageMakerClient}} props
 */
export async function createSagemakerPipeline({
  // Assumes an AWS IAM role has been created for this pipeline.
  roleArn,
  name,
  // Assumes an AWS Lambda function has been created for this pipeline.
  functionArn,
  sagemakerClient,
}) {
  const pipelineDefinition = readFileSync(
    // dirnameFromMetaUrl is a local utility function. You can find its
 implementation
    // on GitHub.
    `${dirnameFromMetaUrl(
      import.meta.url,
    )}../../../../../scenarios/features/sagemaker_pipelines/resources/
GeoSpatialPipeline.json`,
  )
    .toString()
    .replace(/\*FUNCTION_ARN\*/g, functionArn);

  let arn = null;
```

```
  const createPipeline = () =>
    sagemakerClient.send(
      new CreatePipelineCommand({
        PipelineName: name,
        PipelineDefinition: pipelineDefinition,
        RoleArn: roleArn,
      }),
    );

  try {
    const { PipelineArn } = await createPipeline();
    arn = PipelineArn;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ValidationException" &&
      caught.message.includes(
        "Pipeline names must be unique within an AWS account and region",
      )
    ) {
      const { PipelineArn } = await sagemakerClient.send(
        new DescribePipelineCommand({ PipelineName: name }),
      );
      arn = PipelineArn;
    } else {
      throw caught;
    }
  }

  return {
    arn,
    cleanUp: async () => {
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
    },
  };
}
```

- Para obter detalhes da API, consulte CreatePipelinea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}
```

- Para obter detalhes da API, consulte a CreatePipelinereferência da API AWS SDK for Kotlin.

Use **`CreateTrainingJob`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateTrainingJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de usar modelos e endpoints

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
    DATA lt_hyperparameters TYPE /aws1/
cl_sgmhyperparameters_w=>tt_hyperparameters.
    DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
    DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
    DATA lo_resource_config  TYPE REF TO /aws1/cl_sgmresourceconfig.
    DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
    DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.

    "Create ABAP internal table for hyperparameters based on input variables."
```

```
   "These hyperparameters are based on the Amazon SageMaker built-in algorithm,
XGBoost."
   lo_hyperparameters_w = NEW #( iv_value = iv_hp_max_depth ).
   INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eta ).
   INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eval_metric ).
   INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_scale_pos_weight ).
   INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_subsample ).
   INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_objective ).
   INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_num_round ).
   INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   "Create ABAP objects for training data sources."
   lo_trn_s3datasource = NEW #( iv_s3datatype = iv_trn_data_s3datatype
                                iv_s3datadistributiontype =
iv_trn_data_s3datadistribution
                                iv_s3uri = iv_trn_data_s3uri ).

   lo_trn_datasource = NEW #( io_s3datasource = lo_trn_s3datasource ).

   lo_trn_channel = NEW #( iv_channelname = 'train'
                           io_datasource = lo_trn_datasource
                           iv_compressiontype = iv_trn_data_compressiontype
                           iv_contenttype = iv_trn_data_contenttype ).

   INSERT lo_trn_channel INTO TABLE lt_input_data_config.
```

```
   "Create ABAP objects for validation data sources."
   lo_val_s3datasource = NEW #( iv_s3datatype = iv_val_data_s3datatype
                                iv_s3datadistributiontype =
iv_val_data_s3datadistribution
                                iv_s3uri = iv_val_data_s3uri ).

   lo_val_datasource = NEW #( io_s3datasource = lo_val_s3datasource ).

   lo_val_channel = NEW #( iv_channelname = 'validation'
                           io_datasource = lo_val_datasource
                           iv_compressiontype = iv_val_data_compressiontype
                           iv_contenttype = iv_val_data_contenttype ).

   INSERT lo_val_channel INTO TABLE lt_input_data_config.

   "Create an ABAP object for algorithm specification."
   lo_algorithm_specification = NEW #( iv_trainingimage = iv_training_image
                                       iv_traininginputmode =
iv_training_input_mode ).

   "Create an ABAP object for resource configuration."
   lo_resource_config = NEW #( iv_instancecount = iv_instance_count
                               iv_instancetype = iv_instance_type
                               iv_volumesizeingb = iv_volume_sizeingb ).

   "Create an ABAP object for output data configuration."
   lo_output_data_config = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create an ABAP object for stopping condition."
   lo_stopping_condition = NEW #( iv_maxruntimeinseconds =
iv_max_runtime_in_seconds ).

   "Create a training job."
   TRY.
       oo_result = lo_sgm->createtrainingjob(    " oo_result is returned for
testing purposes. "
         iv_trainingjobname           = iv_training_job_name
         iv_rolearn                   = iv_role_arn
         it_hyperparameters           = lt_hyperparameters
         it_inputdataconfig           = lt_input_data_config
         io_algorithmspecification    = lo_algorithm_specification
         io_outputdataconfig          = lo_output_data_config
         io_resourceconfig            = lo_resource_config
```

```
          io_stoppingcondition         = lo_stopping_condition ).
        MESSAGE 'Training job created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourceinuse.
        MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcenotfound.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTrainingJobreferência da API AWS SDK for SAP ABAP.

Use **`CreateTransformJob`**com um AWS SDK

O código de exemplo a seguir mostra como usar `CreateTransformJob`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lo_transforminput TYPE REF TO /aws1/cl_sgmtransforminput.
    DATA lo_transformoutput TYPE REF TO /aws1/cl_sgmtransformoutput.
    DATA lo_transformresources TYPE REF TO /aws1/cl_sgmtransformresources.
    DATA lo_datasource  TYPE REF TO /aws1/cl_sgmtransformdatasrc.
    DATA lo_s3datasource  TYPE REF TO /aws1/cl_sgmtransforms3datasrc.

    "Create an ABAP object for an Amazon Simple Storage Service (Amazon S3) data
source."
    lo_s3datasource = NEW #( iv_s3uri = iv_tf_data_s3uri
                             iv_s3datatype = iv_tf_data_s3datatype ).
```

```
   "Create an ABAP object for data source."
   lo_datasource = NEW #( io_s3datasource = lo_s3datasource ).

   "Create an ABAP object for transform data source."
   lo_transforminput = NEW #( io_datasource = lo_datasource
                              iv_contenttype = iv_tf_data_contenttype
                              iv_compressiontype = iv_tf_data_compressiontype ).

   "Create an ABAP object for resource configuration."
   lo_transformresources = NEW #( iv_instancecount = iv_instance_count
                                  iv_instancetype = iv_instance_type ).

   "Create an ABAP object for output data configuration."
   lo_transformoutput = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create a transform job."
   TRY.
       oo_result = lo_sgm->createtransformjob(     " oo_result is returned for
testing purposes. "
           iv_modelname = iv_tf_model_name
           iv_transformjobname = iv_tf_job_name
           io_transforminput = lo_transforminput
           io_transformoutput = lo_transformoutput
           io_transformresources = lo_transformresources ).
       MESSAGE 'Transform job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTransformJobreferência da API AWS SDK for SAP ABAP.

Use **`DeleteEndpoint`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteEndpoint`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de usar modelos e endpoints

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Delete an endpoint."
    TRY.
        lo_sgm->deleteendpoint(
            iv_endpointname = iv_endpoint_name ).
        MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpoint_exception).
        DATA(lv_endpoint_error) = |"{ lo_endpoint_exception->av_err_code }" -
 { lo_endpoint_exception->av_err_msg }|.
        MESSAGE lv_endpoint_error TYPE 'E'.
    ENDTRY.

    "Delete an endpoint configuration."
    TRY.
        lo_sgm->deleteendpointconfig(
          iv_endpointconfigname = iv_endpoint_config_name ).
        MESSAGE 'Endpoint deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteEndpointreferência da API AWS SDK for SAP ABAP.

Use **`DeleteModel`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DeleteModel`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de usar modelos e endpoints

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sgm->deletemodel(
                  iv_modelname = iv_model_name ).
        MESSAGE 'Model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteModelreferência da API AWS SDK for SAP ABAP.

Use **`DeletePipeline`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeletePipeline`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de trabalhos geoespaciais e pipelines

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete a SageMaker pipeline by name.
/// </summary>
/// <param name="pipelineName">The name of the pipeline to delete.</param>
/// <returns>The ARN of the pipeline.</returns>
public async Task<string> DeletePipelineByName(string pipelineName)
{
    var deleteResponse = await _amazonSageMaker.DeletePipelineAsync(
        new DeletePipelineRequest()
        {
            PipelineName = pipelineName
        });

    return deleteResponse.PipelineArn;
}
```

- Para obter detalhes da API, consulte DeletePipelinea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
 pipelineName) {
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }
```

- Para obter detalhes da API, consulte DeletePipelinea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A sintaxe para excluir um pipeline de SageMaker IA. Esse código faz parte de uma função maior. Consulte “Criar um pipeline” ou o GitHub repositório para obter mais contexto.

```
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
```

- Para obter detalhes da API, consulte DeletePipelinea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeletePipelinereferência da API AWS SDK for Kotlin.

Use **`DescribePipelineExecution`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribePipelineExecution`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de trabalhos geoespaciais e pipelines

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Check the status of a run.
   /// </summary>
   /// <param name="pipelineExecutionArn">The ARN.</param>
   /// <returns>The status of the pipeline.</returns>
   public async Task<PipelineExecutionStatus>
CheckPipelineExecutionStatus(string pipelineExecutionArn)
   {
       var describeResponse = await
_amazonSageMaker.DescribePipelineExecutionAsync(
           new DescribePipelineExecutionRequest()
           {
               PipelineExecutionArn = pipelineExecutionArn
           });

       return describeResponse.PipelineExecutionStatus;
   }
```

- Para obter detalhes da API, consulte DescribePipelineExecutiona Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
            throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                    .pipelineExecutionArn(executionArn)
                    .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " +
status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }
```

- Para obter detalhes da API, consulte DescribePipelineExecutiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Espere que a execução de um pipeline de SageMaker IA seja bem-sucedida, falhe ou pare.

```
/**
```

```
 * Poll the executing pipeline until the status is 'SUCCEEDED', 'STOPPED', or
 'FAILED'.
 * @param {{ arn: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient, wait: (ms: number) => Promise<void>}} props
 */
export async function waitForPipelineComplete({ arn, sagemakerClient, wait }) {
  const command = new DescribePipelineExecutionCommand({
    PipelineExecutionArn: arn,
  });

  let complete = false;
  const intervalInSeconds = 15;
  const COMPLETION_STATUSES = [
    PipelineExecutionStatus.FAILED,
    PipelineExecutionStatus.STOPPED,
    PipelineExecutionStatus.SUCCEEDED,
  ];

  do {
    const { PipelineExecutionStatus: status, FailureReason } =
      await sagemakerClient.send(command);

    complete = COMPLETION_STATUSES.includes(status);

    if (!complete) {
      console.log(
        `Pipeline is ${status}. Waiting ${intervalInSeconds} seconds before
 checking again.`,
      );
      await wait(intervalInSeconds);
    } else if (status === PipelineExecutionStatus.FAILED) {
      throw new Error(`Pipeline failed because: ${FailureReason}`);
    } else if (status === PipelineExecutionStatus.STOPPED) {
      throw new Error("Pipeline was forcefully stopped.");
    } else {
      console.log(`Pipeline execution ${status}.`);
    }
  } while (!complete);
}
```

- Para obter detalhes da API, consulte DescribePipelineExecutiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}
```

- Para obter detalhes da API, consulte a DescribePipelineExecutionreferência da API AWS SDK for Kotlin.

Use **`DescribeTrainingJob`**com um AWS SDK

O código de exemplo a seguir mostra como usar `DescribeTrainingJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de usar modelos e endpoints

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->describetrainingjob(      " oo_result is returned for
 testing purposes. "
          iv_trainingjobname = iv_training_job_name ).
        MESSAGE 'Retrieved description of training job.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeTrainingJobreferência da API AWS SDK for SAP ABAP.

Use **`ListAlgorithms`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListAlgorithms`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->listalgorithms(         " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of algorithms.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListAlgorithmsreferência da API AWS SDK for SAP ABAP.

Use **`ListModels`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListModels`.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->listmodels(           " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of models.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListModelsreferência da API AWS SDK for SAP ABAP.

Use **`ListNotebookInstances`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListNotebookInstances`.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_instances(client: &Client) -> Result<(), Error> {
    let notebooks = client.list_notebook_instances().send().await?;

    println!("Notebooks:");

    for n in notebooks.notebook_instances() {
        let n_instance_type = n.instance_type().unwrap();
        let n_status = n.notebook_instance_status().unwrap();
        let n_name = n.notebook_instance_name();

        println!("  Name :          {}", n_name.unwrap_or("Unknown"));
        println!("  Status :        {}", n_status.as_ref());
        println!("  Instance Type : {}", n_instance_type.as_ref());
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListNotebookInstancesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->listnotebookinstances(        " oo_result is returned
 for testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of notebook instances.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListNotebookInstancesreferência da API AWS SDK for SAP ABAP.

Use **`ListTrainingJobs`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListTrainingJobs`.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_jobs(client: &Client) -> Result<(), Error> {
```

```
    let job_details = client.list_training_jobs().send().await?;

    println!("Jobs:");

    for j in job_details.training_job_summaries() {
        let name = j.training_job_name().unwrap_or("Unknown");
        let creation_time = j.creation_time().expect("creation
 time").to_chrono_utc()?;
        let training_end_time = j
            .training_end_time()
            .expect("Training end time")
            .to_chrono_utc()?;

        let status = j.training_job_status().expect("training status");
        let duration = training_end_time - creation_time;

        println!("  Name:               {}", name);
        println!(
            "  Creation date/time: {}",
            creation_time.format("%Y-%m-%d@%H:%M:%S")
        );
        println!("  Duration (seconds): {}", duration.num_seconds());
        println!("  Status:             {:?}", status);

        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListTrainingJobsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->listtrainingjobs(       " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains
          iv_maxresults = iv_max_results ).
        MESSAGE 'Retrieved list of training jobs.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTrainingJobsreferência da API AWS SDK for SAP ABAP.

Use **`StartPipelineExecution`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartPipelineExecution`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de trabalhos geoespaciais e pipelines

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Run a pipeline with input and output file locations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue to use for pipeline
callbacks.</param>
    /// <param name="inputLocationUrl">The input location in Amazon Simple
Storage Service (Amazon S3).</param>
    /// <param name="outputLocationUrl">The output location in Amazon S3.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="executionRoleArn">The ARN of the role.</param>
    /// <returns>The ARN of the pipeline run.</returns>
    public async Task<string> ExecutePipeline(
        string queueUrl,
        string inputLocationUrl,
        string outputLocationUrl,
        string pipelineName,
        string executionRoleArn)
    {
        var inputConfig = new VectorEnrichmentJobInputConfig()
        {
            DataSourceConfig = new()
            {
                S3Data = new VectorEnrichmentJobS3Data()
                {
                    S3Uri = inputLocationUrl
                }
            },
            DocumentType = VectorEnrichmentJobDocumentType.CSV
        };

        var exportConfig = new ExportVectorEnrichmentJobOutputConfig()
        {
```

```
            S3Data = new VectorEnrichmentJobS3Data()
            {
                S3Uri = outputLocationUrl
            }
        };

        var jobConfig = new VectorEnrichmentJobConfig()
        {
            ReverseGeocodingConfig = new ReverseGeocodingConfig()
            {
                XAttributeName = "Longitude",
                YAttributeName = "Latitude"
            }
        };

#pragma warning disable SageMaker1002 // Property value does not match required
 pattern is allowed here to match the pipeline definition.
        var startExecutionResponse = await
 _amazonSageMaker.StartPipelineExecutionAsync(
            new StartPipelineExecutionRequest()
            {
                PipelineName = pipelineName,
                PipelineExecutionDisplayName = pipelineName + "-example-
execution",
                PipelineParameters = new List<Parameter>()
                {
                    new Parameter() { Name = "parameter_execution_role", Value =
 executionRoleArn },
                    new Parameter() { Name = "parameter_queue_url", Value =
 queueUrl },
                    new Parameter() { Name = "parameter_vej_input_config", Value
 = JsonSerializer.Serialize(inputConfig) },
                    new Parameter() { Name = "parameter_vej_export_config", Value
 = JsonSerializer.Serialize(exportConfig) },
                    new Parameter() { Name = "parameter_step_1_vej_config", Value
 = JsonSerializer.Serialize(jobConfig) }
                }
            });
#pragma warning restore SageMaker1002
        return startExecutionResponse.PipelineExecutionArn;
    }
```

- Para obter detalhes da API, consulte StartPipelineExecutiona Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
```

```
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";

        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();

        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();
```

```
        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig
\":{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();

        System.out.println("parameter_step_1_vej_config:" +
 gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);

        StartPipelineExecutionRequest pipelineExecutionRequest =
 StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-
execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }
```

- Para obter detalhes da API, consulte StartPipelineExecutiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicie a execução de um pipeline de SageMaker IA.

```
/**
 * Start the execution of the Amazon SageMaker pipeline. Parameters that are
 * passed in are used in the AWS Lambda function.
 * @param {{
 *   name: string,
 *   sagemakerClient: import('@aws-sdk/client-sagemaker').SageMakerClient,
 *   roleArn: string,
 *   queueUrl: string,
 *   s3InputBucketName: string,
 * }} props
 */
export async function startPipelineExecution({
  sagemakerClient,
  name,
  bucketName,
  roleArn,
  queueUrl,
}) {
  /**
   * The Vector Enrichment Job requests CSV data. This configuration points to a
 CSV
   * file in an Amazon S3 bucket.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobInputConfig}
   */
  const inputConfig = {
    DataSourceConfig: {
      S3Data: {
        S3Uri: `s3://${bucketName}/input/sample_data.csv`,
      },
    },
    DocumentType: VectorEnrichmentJobDocumentType.CSV,
  };

  /**
   * The Vector Enrichment Job adds additional data to the source CSV. This
 configuration points
   * to an Amazon S3 prefix where the output will be stored.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").ExportVectorEnrichmentJobOutputConfig}
   */
  const outputConfig = {
```

```
    S3Data: {
      S3Uri: `s3://${bucketName}/output/`,
    },
  };

  /**
   * This job will be a Reverse Geocoding Vector Enrichment Job. Reverse
 Geocoding requires
   * latitude and longitude values.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobConfig}
   */
  const jobConfig = {
    ReverseGeocodingConfig: {
      XAttributeName: "Longitude",
      YAttributeName: "Latitude",
    },
  };

  const { PipelineExecutionArn } = await sagemakerClient.send(
    new StartPipelineExecutionCommand({
      PipelineName: name,
      PipelineExecutionDisplayName: `${name}-example-execution`,
      PipelineParameters: [
        { Name: "parameter_execution_role", Value: roleArn },
        { Name: "parameter_queue_url", Value: queueUrl },
        {
          Name: "parameter_vej_input_config",
          Value: JSON.stringify(inputConfig),
        },
        {
          Name: "parameter_vej_export_config",
          Value: JSON.stringify(outputConfig),
        },
        {
          Name: "parameter_step_1_vej_config",
          Value: JSON.stringify(jobConfig),
        },
      ],
    }),
  );

  return {
    arn: PipelineExecutionArn,
```

```
  };
}
```

- Para obter detalhes da API, consulte StartPipelineExecutiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn:
 String?, pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }
```

```
    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName
\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
```

```
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}
```

- Para obter detalhes da API, consulte a StartPipelineExecutionreferência da API AWS SDK for Kotlin.

Use **`UpdatePipeline`**com um AWS SDK

O código de exemplo a seguir mostra como usar `UpdatePipeline`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de trabalhos geoespaciais e pipelines

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }
```

- Para obter detalhes da API, consulte UpdatePipelinea Referência AWS SDK para .NET da API.

Cenários para uso de SageMaker IA AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns em SageMaker IA com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções dentro da SageMaker IA ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Comece a usar trabalhos geoespaciais de SageMaker IA em um pipeline usando um AWS SDK
- Comece a usar modelos e endpoints de SageMaker IA usando um AWS SDK
- Conceitos básicos de arquivos de atributos de machine learning

Comece a usar trabalhos geoespaciais de SageMaker IA em um pipeline usando um AWS SDK

Os exemplos de código a seguir mostram como:

- Configurar recursos para um pipeline.
- Configurar um pipeline que executa um trabalho geoespacial.
- Iniciar a execução de um pipeline.
- Monitorar o status da execução.
- Ver a saída do pipeline.
- Limpar recursos.

Para obter mais informações, consulte Criar e executar SageMaker pipelines usando AWS SDKs em. Community.aws

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as operações de SageMaker IA.

```
using System.Text.Json;
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
using Amazon.SageMakerGeospatial;
using Amazon.SageMakerGeospatial.Model;

namespace SageMakerActions;

/// <summary>
/// Wrapper class for Amazon SageMaker actions and logic.
/// </summary>
public class SageMakerWrapper
{
    private readonly IAmazonSageMaker _amazonSageMaker;
    public SageMakerWrapper(IAmazonSageMaker amazonSageMaker)
    {
        _amazonSageMaker = amazonSageMaker;
    }

    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
```

```
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }

    /// <summary>
    /// Run a pipeline with input and output file locations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue to use for pipeline
callbacks.</param>
    /// <param name="inputLocationUrl">The input location in Amazon Simple
Storage Service (Amazon S3).</param>
    /// <param name="outputLocationUrl">The output location in Amazon S3.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="executionRoleArn">The ARN of the role.</param>
    /// <returns>The ARN of the pipeline run.</returns>
    public async Task<string> ExecutePipeline(
        string queueUrl,
        string inputLocationUrl,
        string outputLocationUrl,
        string pipelineName,
        string executionRoleArn)
    {
```

```
        var inputConfig = new VectorEnrichmentJobInputConfig()
        {
            DataSourceConfig = new()
            {
                S3Data = new VectorEnrichmentJobS3Data()
                {
                    S3Uri = inputLocationUrl
                }
            },
            DocumentType = VectorEnrichmentJobDocumentType.CSV
        };

        var exportConfig = new ExportVectorEnrichmentJobOutputConfig()
        {
            S3Data = new VectorEnrichmentJobS3Data()
            {
                S3Uri = outputLocationUrl
            }
        };

        var jobConfig = new VectorEnrichmentJobConfig()
        {
            ReverseGeocodingConfig = new ReverseGeocodingConfig()
            {
                XAttributeName = "Longitude",
                YAttributeName = "Latitude"
            }
        };

#pragma warning disable SageMaker1002 // Property value does not match required
 pattern is allowed here to match the pipeline definition.
        var startExecutionResponse = await
 _amazonSageMaker.StartPipelineExecutionAsync(
            new StartPipelineExecutionRequest()
            {
                PipelineName = pipelineName,
                PipelineExecutionDisplayName = pipelineName + "-example-
execution",
                PipelineParameters = new List<Parameter>()
                {
                    new Parameter() { Name = "parameter_execution_role", Value =
 executionRoleArn },
                    new Parameter() { Name = "parameter_queue_url", Value =
 queueUrl },
```

```
                    new Parameter() { Name = "parameter_vej_input_config", Value
 = JsonSerializer.Serialize(inputConfig) },
                    new Parameter() { Name = "parameter_vej_export_config", Value
 = JsonSerializer.Serialize(exportConfig) },
                    new Parameter() { Name = "parameter_step_1_vej_config", Value
 = JsonSerializer.Serialize(jobConfig) }
                }
            });
#pragma warning restore SageMaker1002
        return startExecutionResponse.PipelineExecutionArn;
    }

    /// <summary>
    /// Check the status of a run.
    /// </summary>
    /// <param name="pipelineExecutionArn">The ARN.</param>
    /// <returns>The status of the pipeline.</returns>
    public async Task<PipelineExecutionStatus>
 CheckPipelineExecutionStatus(string pipelineExecutionArn)
    {
        var describeResponse = await
 _amazonSageMaker.DescribePipelineExecutionAsync(
            new DescribePipelineExecutionRequest()
            {
                PipelineExecutionArn = pipelineExecutionArn
            });

        return describeResponse.PipelineExecutionStatus;
    }

    /// <summary>
    /// Delete a SageMaker pipeline by name.
    /// </summary>
    /// <param name="pipelineName">The name of the pipeline to delete.</param>
    /// <returns>The ARN of the pipeline.</returns>
    public async Task<string> DeletePipelineByName(string pipelineName)
    {
        var deleteResponse = await _amazonSageMaker.DeletePipelineAsync(
            new DeletePipelineRequest()
            {
                PipelineName = pipelineName
            });

        return deleteResponse.PipelineArn;
```

```
    }
}
```

Crie uma função que gerencie retornos de chamada do pipeline de SageMaker IA.

```
using System.Text.Json;
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
using Amazon.SageMakerGeospatial;
using Amazon.SageMakerGeospatial.Model;

// Assembly attribute to enable the AWS Lambda function's JSON input to be
 converted into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SageMakerLambda;

/// <summary>
/// The AWS Lambda function handler for the Amazon SageMaker pipeline.
/// </summary>
public class SageMakerLambdaFunction
{
    /// <summary>
    /// Default constructor. This constructor is used by AWS Lambda to construct
 the instance. When invoked in a Lambda environment
    /// the AWS credentials will come from the AWS Identity and Access Management
 (IAM) role associated with the function. The AWS Region will be set to the
    /// Region that the Lambda function is running in.
    /// </summary>
    public SageMakerLambdaFunction()
    {
    }

    /// <summary>
    /// The AWS Lambda function handler that processes events from the SageMaker
 pipeline and starts a job or export.
    /// </summary>
```

```
    /// <param name="request">The custom SageMaker pipeline request object.</
param>
    /// <param name="context">The Lambda context.</param>
    /// <returns>The dictionary of output parameters.</returns>
    public async Task<Dictionary<string, string>> FunctionHandler(PipelineRequest
 request, ILambdaContext context)
    {
        var geoSpatialClient = new AmazonSageMakerGeospatialClient();
        var sageMakerClient = new AmazonSageMakerClient();
        var responseDictionary = new Dictionary<string, string>();
        context.Logger.LogInformation("Function handler started with request: " +
 JsonSerializer.Serialize(request));
        if (request.Records != null && request.Records.Any())
        {
            context.Logger.LogInformation("Records found, this is a queue event.
 Processing the queue records.");
            foreach (var message in request.Records)
            {
                await ProcessMessageAsync(message, context, geoSpatialClient,
 sageMakerClient);
            }
        }
        else if (!string.IsNullOrEmpty(request.vej_export_config))
        {
            context.Logger.LogInformation("Export configuration found, this is an
 export. Start the Vector Enrichment Job (VEJ) export.");

            var outputConfig =

 JsonSerializer.Deserialize<ExportVectorEnrichmentJobOutputConfig>(
                    request.vej_export_config);

            var exportResponse = await
 geoSpatialClient.ExportVectorEnrichmentJobAsync(
                new ExportVectorEnrichmentJobRequest()
                {
                    Arn = request.vej_arn,
                    ExecutionRoleArn = request.Role,
                    OutputConfig = outputConfig
                });
            context.Logger.LogInformation($"Export response:
 {JsonSerializer.Serialize(exportResponse)}");
            responseDictionary = new Dictionary<string, string>
            {
```

```
                { "export_eoj_status", exportResponse.ExportStatus.ToString() },
                { "vej_arn", exportResponse.Arn }
            };
        }
        else if (!string.IsNullOrEmpty(request.vej_name))
        {
            context.Logger.LogInformation("Vector Enrichment Job name found,
starting the job.");
            var inputConfig =
                JsonSerializer.Deserialize<VectorEnrichmentJobInputConfig>(
                    request.vej_input_config);

            var jobConfig =
                JsonSerializer.Deserialize<VectorEnrichmentJobConfig>(
                    request.vej_config);

            var jobResponse = await
geoSpatialClient.StartVectorEnrichmentJobAsync(
                new StartVectorEnrichmentJobRequest()
                {
                    ExecutionRoleArn = request.Role,
                    InputConfig = inputConfig,
                    Name = request.vej_name,
                    JobConfig = jobConfig

                });
            context.Logger.LogInformation("Job response: " +
JsonSerializer.Serialize(jobResponse));
            responseDictionary = new Dictionary<string, string>
            {
                { "vej_arn", jobResponse.Arn },
                { "statusCode", jobResponse.HttpStatusCode.ToString() }
            };
        }
        return responseDictionary;
    }

    /// <summary>
    /// Process a queue message and check the status of a SageMaker job.
    /// </summary>
    /// <param name="message">The queue message.</param>
    /// <param name="context">The Lambda context.</param>
    /// <param name="geoClient">The SageMaker GeoSpatial client.</param>
    /// <param name="sageMakerClient">The SageMaker client.</param>
```

```
    /// <returns>Async task.</returns>
    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
ILambdaContext context,
        AmazonSageMakerGeospatialClient geoClient, AmazonSageMakerClient
sageMakerClient)
    {
        context.Logger.LogInformation($"Processed message {message.Body}");

        // Get information about the SageMaker job.
        var payload = JsonSerializer.Deserialize<QueuePayload>(message.Body);
        context.Logger.LogInformation($"Payload token {payload!.token}");
        var token = payload.token;

        if (payload.arguments.ContainsKey("vej_arn"))
        {
            // Use the job ARN and the token to get the job status.
            var job_arn = payload.arguments["vej_arn"];
            context.Logger.LogInformation($"Token: {token}, arn {job_arn}");

            var jobInfo = geoClient.GetVectorEnrichmentJobAsync(
                new GetVectorEnrichmentJobRequest()
                {
                    Arn = job_arn
                });
            context.Logger.LogInformation("Job info: " +
JsonSerializer.Serialize(jobInfo));
            if (jobInfo.Result.Status == VectorEnrichmentJobStatus.COMPLETED)
            {
                context.Logger.LogInformation($"Status completed, resuming
pipeline...");
                await sageMakerClient.SendPipelineExecutionStepSuccessAsync(
                    new SendPipelineExecutionStepSuccessRequest()
                    {
                        CallbackToken = token,
                        OutputParameters = new List<OutputParameter>()
                        {
                            new OutputParameter()
                                { Name = "export_status", Value =
jobInfo.Result.Status }
                        }
                    });
            }
            else if (jobInfo.Result.Status == VectorEnrichmentJobStatus.FAILED)
            {
```

```
                context.Logger.LogInformation($"Status failed, stopping
 pipeline...");
                await sageMakerClient.SendPipelineExecutionStepFailureAsync(
                    new SendPipelineExecutionStepFailureRequest()
                    {
                        CallbackToken = token,
                        FailureReason = jobInfo.Result.ErrorDetails.ErrorMessage
                    });
            }
            else if (jobInfo.Result.Status ==
 VectorEnrichmentJobStatus.IN_PROGRESS)
            {
                // Put this message back in the queue to reprocess later.
                context.Logger.LogInformation(
                    $"Status still in progress, check back later.");
                throw new("Job still running.");
            }
        }
    }
}
```

Execute um cenário interativo em um prompt de comando.

```
public static class PipelineWorkflow
{
    public static IAmazonIdentityManagementService _iamClient = null!;
    public static SageMakerWrapper _sageMakerWrapper = null!;
    public static IAmazonSQS _sqsClient = null!;
    public static IAmazonS3 _s3Client = null!;
    public static IAmazonLambda _lambdaClient = null!;
    public static IConfiguration _configuration = null!;

    public static string lambdaFunctionName = "SageMakerExampleFunction";
    public static string sageMakerRoleName = "SageMakerExampleRole";
    public static string lambdaRoleName = "SageMakerExampleLambdaRole";

    private static string[] lambdaRolePolicies = null!;
    private static string[] sageMakerRolePolicies = null!;

    static async Task Main(string[] args)
    {
        var options = new AWSOptions() { Region = RegionEndpoint.USWest2 };
```

```
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>(options)
                    .AddAWSService<IAmazonEC2>(options)
                    .AddAWSService<IAmazonSageMaker>(options)
                    .AddAWSService<IAmazonSageMakerGeospatial>(options)
                    .AddAWSService<IAmazonSQS>(options)
                    .AddAWSService<IAmazonS3>(options)
                    .AddAWSService<IAmazonLambda>(options)
                    .AddTransient<SageMakerWrapper>()
        )
        .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ServicesSetup(host);
        string queueUrl = "";
        string queueName = _configuration["queueName"];
        string bucketName = _configuration["bucketName"];
        var pipelineName = _configuration["pipelineName"];

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(
                "Welcome to the Amazon SageMaker pipeline example scenario.");
            Console.WriteLine(
                "\nThis example scenario will guide you through setting up and
running an" +
                "\nAmazon SageMaker pipeline. The pipeline uses an AWS Lambda
function and an" +
```

```
                "\nAmazon SQS Queue. It runs a vector enrichment reverse geocode
 job to" +
                "\nreverse geocode addresses in an input file and store the
 results in an export file.");
            Console.WriteLine(new string('-', 80));

            Console.WriteLine(new string('-', 80));
            Console.WriteLine(
                "First, we will set up the roles, functions, and queue needed by
 the SageMaker pipeline.");
            Console.WriteLine(new string('-', 80));

            var lambdaRoleArn = await CreateLambdaRole();
            var sageMakerRoleArn = await CreateSageMakerRole();
            var functionArn = await SetupLambda(lambdaRoleArn, true);
            queueUrl = await SetupQueue(queueName);
            await SetupBucket(bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Now we can create and run our pipeline.");
            Console.WriteLine(new string('-', 80));

            await SetupPipeline(sageMakerRoleArn, functionArn, pipelineName);
            var executionArn = await ExecutePipeline(queueUrl, sageMakerRoleArn,
 pipelineName, bucketName);
            await WaitForPipelineExecution(executionArn);

            await GetOutputResults(bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("The pipeline has completed. To view the pipeline
 and runs " +
                              "in SageMaker Studio, follow these instructions:" +
                              "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/
pipelines-studio.html");
            Console.WriteLine(new string('-', 80));

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await CleanupResources(true, queueUrl, pipelineName, bucketName);

            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine("SageMaker pipeline scenario is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources(true, queueUrl, pipelineName, bucketName);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sageMakerWrapper = host.Services.GetRequiredService<SageMakerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _sqsClient = host.Services.GetRequiredService<IAmazonSQS>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _lambdaClient = host.Services.GetRequiredService<IAmazonLambda>();
    }

    /// <summary>
    /// Set up AWS Lambda, either by updating an existing function or creating a
new function.
    /// </summary>
    /// <param name="roleArn">The role Amazon Resource Name (ARN) to use for the
Lambda function.</param>
    /// <param name="askUser">True to ask the user before updating.</param>
    /// <returns>The ARN of the function.</returns>
    public static async Task<string> SetupLambda(string roleArn, bool askUser)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Setting up the Lambda function for the pipeline.");
        var handlerName =
"SageMakerLambda::SageMakerLambda.SageMakerLambdaFunction::FunctionHandler";
        var functionArn = "";
        try
        {
```

```
            var functionInfo = await _lambdaClient.GetFunctionAsync(new
GetFunctionRequest()
            {
                FunctionName = lambdaFunctionName
            });

            var updateFunction = true;
            if (askUser)
            {
                updateFunction = GetYesNoResponse(
                    $"\tThe Lambda function {lambdaFunctionName} already exists,
do you want to update it?");
            }

            if (updateFunction)
            {
                // Update the Lambda function.
                using var zipMemoryStream = new MemoryStream(await
File.ReadAllBytesAsync("SageMakerLambda.zip"));
                await _lambdaClient.UpdateFunctionCodeAsync(
                    new UpdateFunctionCodeRequest()
                    {
                        FunctionName = lambdaFunctionName,
                        ZipFile = zipMemoryStream,
                    });
            }

            functionArn = functionInfo.Configuration.FunctionArn;
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"\tThe Lambda function {lambdaFunctionName} was
not found, creating the new function.");

            // Create the function if it does not already exist.
            using var zipMemoryStream = new MemoryStream(await
File.ReadAllBytesAsync("SageMakerLambda.zip"));
            var createResult = await _lambdaClient.CreateFunctionAsync(
                new CreateFunctionRequest()
                {
                    FunctionName = lambdaFunctionName,
                    Runtime = Runtime.Dotnet6,
                    Description = "SageMaker example function.",
                    Code = new FunctionCode()
```

```
                    {
                        ZipFile = zipMemoryStream
                    },
                    Handler = handlerName,
                    Role = roleArn,
                    Timeout = 30
                });

            functionArn = createResult.FunctionArn;
        }

        Console.WriteLine($"\tLambda ready with ARN {functionArn}.");
        Console.WriteLine(new string('-', 80));
        return functionArn;
    }

    /// <summary>
    /// Create a role to be used by AWS Lambda. Does not create the role if it
 already exists.
    /// </summary>
    /// <returns>The role ARN.</returns>
    public static async Task<string> CreateLambdaRole()
    {
        Console.WriteLine(new string('-', 80));

        lambdaRolePolicies = new string[]{
            "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess",
            "arn:aws:iam::aws:policy/AmazonSQSFullAccess",
            "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess",
            "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy",
            "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
        };

        var roleArn = await GetRoleArnIfExists(lambdaRoleName);
        if (!string.IsNullOrEmpty(roleArn))
        {
            return roleArn;
        }

        Console.WriteLine("\tCreating a role to for AWS Lambda to use.");
```

```
        var assumeRolePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": [" +
                                             "\"sagemaker.amazonaws.com\"," +
                                             "\"sagemaker-geospatial.amazonaws.com
\"," +
                                             "\"lambda.amazonaws.com\"," +
                                             "\"s3.amazonaws.com\"" +
                                         "]" +
                                     "}," +
                                     "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = lambdaRoleName
            });
        foreach (var policy in lambdaRolePolicies)
        {
            await _iamClient.AttachRolePolicyAsync(
                new AttachRolePolicyRequest()
                {
                    PolicyArn = policy,
                    RoleName = lambdaRoleName
                });
        }

        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        Console.WriteLine($"\tRole ready with ARN {roleResult.Role.Arn}.");
        Console.WriteLine(new string('-', 80));

        return roleResult.Role.Arn;
    }


    /// <summary>
```

```
    /// Create a role to be used by SageMaker.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateSageMakerRole()
    {
        Console.WriteLine(new string('-', 80));

        sageMakerRolePolicies = new string[]{
            "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess",
            "arn:aws:iam::aws:policy/AmazonSageMakerGeospatialFullAccess",
        };

        var roleArn = await GetRoleArnIfExists(sageMakerRoleName);
        if (!string.IsNullOrEmpty(roleArn))
        {
            return roleArn;
        }

        Console.WriteLine("\tCreating a role to use with SageMaker.");

        var assumeRolePolicy = "{" +
                                        "\"Version\": \"2012-10-17\"," +
                                        "\"Statement\": [{" +
                                            "\"Effect\": \"Allow\"," +
                                            "\"Principal\": {" +
                                                $"\"Service\": [" +
                                                    "\"sagemaker.amazonaws.com
\"," +
                                                    "\"sagemaker-
geospatial.amazonaws.com\"," +
                                                    "\"lambda.amazonaws.com\"," +
                                                    "\"s3.amazonaws.com\"" +
                                                "]" +
                                            "}," +
                                            "\"Action\": \"sts:AssumeRole\"" +
                                        "}]" +
                                    "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = sageMakerRoleName
```

```
            });

        foreach (var policy in sageMakerRolePolicies)
        {
            await _iamClient.AttachRolePolicyAsync(
                new AttachRolePolicyRequest()
                {
                    PolicyArn = policy,
                    RoleName = sageMakerRoleName
                });
        }

        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        Console.WriteLine($"\tRole ready with ARN {roleResult.Role.Arn}.");
        Console.WriteLine(new string('-', 80));
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Set up the SQS queue to use with the pipeline.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <returns>The URL for the queue.</returns>
    public static async Task<string> SetupQueue(string queueName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up queue {queueName}.");

        try
        {
            var queueInfo = await _sqsClient.GetQueueUrlAsync(new
GetQueueUrlRequest()
            { QueueName = queueName });
            return queueInfo.QueueUrl;
        }
        catch (QueueDoesNotExistException)
        {
            var attrs = new Dictionary<string, string>
            {
                {
                    QueueAttributeName.DelaySeconds,
                    "5"
                },
```

```
                {
                    QueueAttributeName.ReceiveMessageWaitTimeSeconds,
                    "5"
                },
                {
                    QueueAttributeName.VisibilityTimeout,
                    "300"
                },
            };

            var request = new CreateQueueRequest
            {
                Attributes = attrs,
                QueueName = queueName,
            };

            var response = await _sqsClient.CreateQueueAsync(request);
            Thread.Sleep(10000);
            await ConnectLambda(response.QueueUrl);
            Console.WriteLine($"\tQueue ready with Url {response.QueueUrl}.");
            Console.WriteLine(new string('-', 80));
            return response.QueueUrl;
        }
    }

    /// <summary>
    /// Connect the queue to the Lambda function as an event source.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue.</param>
    /// <returns>Async task.</returns>
    public static async Task ConnectLambda(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Connecting the Lambda function and queue for the
pipeline.");

        var queueAttributes = await _sqsClient.GetQueueAttributesAsync(
            new GetQueueAttributesRequest() { QueueUrl = queueUrl, AttributeNames
= new List<string>() { "All" } });
        var queueArn = queueAttributes.QueueARN;

        var eventSource = await _lambdaClient.ListEventSourceMappingsAsync(
             new ListEventSourceMappingsRequest()
             {
```

```
                 FunctionName = lambdaFunctionName
             });

        if (!eventSource.EventSourceMappings.Any())
        {
            // Only add the event source mapping if it does not already exist.
            await _lambdaClient.CreateEventSourceMappingAsync(
                new CreateEventSourceMappingRequest()
                {
                    EventSourceArn = queueArn,
                    FunctionName = lambdaFunctionName,
                    Enabled = true
                });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the bucket to use for pipeline input and output.
    /// </summary>
    /// <param name="bucketName">The name for the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task SetupBucket(string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up bucket {bucketName}.");

        var bucketExists = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            bucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = bucketName,
                BucketRegion = S3Region.USWest2
            });

            Thread.Sleep(5000);

            await _s3Client.PutObjectAsync(new PutObjectRequest()
            {
```

```
                BucketName = bucketName,
                Key = "samplefiles/latlongtest.csv",
                FilePath = "latlongtest.csv"
            });
        }

        Console.WriteLine($"\tBucket {bucketName} ready.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Display some results from the output directory.
    /// </summary>
    /// <param name="bucketName">The name for the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task<string> GetOutputResults(string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Getting output results {bucketName}.");
        string outputKey = "";
        Thread.Sleep(15000);
        var outputFiles = await _s3Client.ListObjectsAsync(
              new ListObjectsRequest()
              {
                  BucketName = bucketName,
                  Prefix = "outputfiles/"
              });

        if (outputFiles.S3Objects.Any())
        {
            var sampleOutput = outputFiles.S3Objects.OrderBy(s =>
s.LastModified).Last();
            Console.WriteLine($"\tOutput file: {sampleOutput.Key}");
            var outputSampleResponse = await _s3Client.GetObjectAsync(
                new GetObjectRequest()
                {
                    BucketName = bucketName,
                    Key = sampleOutput.Key
                });
            outputKey = sampleOutput.Key;
            StreamReader reader = new
StreamReader(outputSampleResponse.ResponseStream);
            await reader.ReadLineAsync();
            Console.WriteLine("\tOutput file contents: \n");
```

```
            for (int i = 0; i < 10; i++)
            {
                if (!reader.EndOfStream)
                {
                    Console.WriteLine("\t" + await reader.ReadLineAsync());
                }
            }
        }

        Console.WriteLine(new string('-', 80));
        return outputKey;
    }

    /// <summary>
    /// Create a pipeline from the example pipeline JSON
    /// that includes the Lambda, callback, processing, and export jobs.
    /// </summary>
    /// <param name="roleArn">The ARN of the role for the pipeline.</param>
    /// <param name="functionArn">The ARN of the Lambda function for the
pipeline.</param>
    /// <param name="pipelineName">The name for the pipeline.</param>
    /// <returns>The ARN of the pipeline.</returns>
    public static async Task<string> SetupPipeline(string roleArn, string
functionArn, string pipelineName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up the pipeline.");

        var pipelineJson = await
File.ReadAllTextAsync("GeoSpatialPipeline.json");

        // Add the correct function ARN instead of the placeholder.
        pipelineJson = pipelineJson.Replace("*FUNCTION_ARN*", functionArn);

        var pipelineArn = await _sageMakerWrapper.SetupPipeline(pipelineJson,
roleArn, pipelineName,
            "sdk example pipeline", pipelineName);

        Console.WriteLine($"\tPipeline set up with ARN {pipelineArn}.");
        Console.WriteLine(new string('-', 80));

        return pipelineArn;
    }
```

```
    /// <summary>
    /// Start a pipeline run with job configurations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue used in the pipeline.</
param>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>The pipeline run ARN.</returns>
    public static async Task<string> ExecutePipeline(
        string queueUrl,
        string roleArn,
        string pipelineName,
        string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Starting pipeline execution.");

        var input = $"s3://{bucketName}/samplefiles/latlongtest.csv";
        var output = $"s3://{bucketName}/outputfiles/";

        var executionARN =
            await _sageMakerWrapper.ExecutePipeline(queueUrl, input, output,
                pipelineName, roleArn);

        Console.WriteLine($"\tRun started with ARN {executionARN}.");
        Console.WriteLine(new string('-', 80));

        return executionARN;
    }

    /// <summary>
    /// Wait for a pipeline run to complete.
    /// </summary>
    /// <param name="executionArn">The pipeline run ARN.</param>
    /// <returns>Async task.</returns>
    public static async Task WaitForPipelineExecution(string executionArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Waiting for pipeline to finish.");

        PipelineExecutionStatus status;
        do
        {
```

```
            status = await
 _sageMakerWrapper.CheckPipelineExecutionStatus(executionArn);
            Thread.Sleep(30000);
            Console.WriteLine($"\tStatus is {status}.");
        } while (status == PipelineExecutionStatus.Executing);

        Console.WriteLine($"\tPipeline finished with status {status}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="askUser">True to ask the user for cleanup.</param>
    /// <param name="queueUrl">The URL of the queue to clean up.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> CleanupResources(
        bool askUser,
        string queueUrl,
        string pipelineName,
        string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (!askUser || GetYesNoResponse($"\tDelete pipeline {pipelineName}? (y/
n)"))
        {
            Console.WriteLine($"\tDeleting pipeline.");
            // Delete the pipeline.
            await _sageMakerWrapper.DeletePipelineByName(pipelineName);
        }

        if (!string.IsNullOrEmpty(queueUrl) && (!askUser ||
 GetYesNoResponse($"\tDelete queue {queueUrl}? (y/n)")))
        {
            Console.WriteLine($"\tDeleting queue.");
            // Delete the queue.
            await _sqsClient.DeleteQueueAsync(new DeleteQueueRequest(queueUrl));
        }
```

```
        if (!askUser || GetYesNoResponse($"\tDelete Amazon S3 bucket
 {bucketName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
 ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            if (deleteList.KeyCount > 0)
            {
                await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.S3Objects
                        .Select(o => new KeyVersion { Key = o.Key }).ToList()
                });
            }

            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete lambda {lambdaFunctionName}?
 (y/n)"))
        {
            Console.WriteLine($"\tDeleting lambda function.");

            await _lambdaClient.DeleteFunctionAsync(new DeleteFunctionRequest()
            {
                FunctionName = lambdaFunctionName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete role {lambdaRoleName}? (y/
n)"))
        {
            Console.WriteLine($"\tDetaching policies and deleting role.");

            foreach (var policy in lambdaRolePolicies)
```

```
            {
                await _iamClient!.DetachRolePolicyAsync(new
 DetachRolePolicyRequest()
                {
                    RoleName = lambdaRoleName,
                    PolicyArn = policy
                });
            }

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = lambdaRoleName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete role {sageMakerRoleName}? (y/
n)"))
        {
            Console.WriteLine($"\tDetaching policies and deleting role.");

            foreach (var policy in sageMakerRolePolicies)
            {
                await _iamClient!.DetachRolePolicyAsync(new
 DetachRolePolicyRequest()
                {
                    RoleName = sageMakerRoleName,
                    PolicyArn = policy
                });
            }

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = sageMakerRoleName
            });
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a role's ARN if it already exists.
    /// </summary>
```

```
    /// <param name="roleName">The name of the AWS Identity and Access Management
 (IAM) Role to look for.</param>
    /// <returns>The role ARN if it exists, otherwise an empty string.</returns>
    private static async Task<string> GetRoleArnIfExists(string roleName)
    {
        Console.WriteLine($"Checking for role named {roleName}.");

        try
        {
            var existingRole = await _iamClient.GetRoleAsync(new GetRoleRequest()
            {
                RoleName = lambdaRoleName
            });
            return existingRole.Role.Arn;
        }
        catch (NoSuchEntityException)
        {
            return string.Empty;
        }
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreatePipeline

- DeletePipeline
- DescribePipelineExecution
- StartPipelineExecution
- UpdatePipeline

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
 <functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
 <pipelineName>\n\n"
                +
                "Where:\n" +
                "    sageMakerRoleName - The name of the Amazon SageMaker role.\n
\n" +
                "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
                "    functionFileLocation - The file location where the JAR file
 that represents the AWS Lambda function is located.\n\n"
                +
                "    functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).\n\n" +
                "    queueName - The name of the Amazon Simple Queue Service
 (Amazon SQS) queue.\n\n" +
                "    bucketName - The name of the Amazon Simple Storage Service
 (Amazon S3) bucket.\n\n" +
```

```
                "    lnglatData - The file location of the latlongtest.csv file
required for this use case.\n\n" +
                "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"
                +
                "    pipelineName - The name of the pipeline to create (for
example, sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String sageMakerRoleName = args[0];
        String lambdaRoleName = args[1];
        String functionFileLocation = args[2];
        String functionName = args[3];
        String queueName = args[4];
        String bucketName = args[5];
        String lnglatData = args[6];
        String spatialPipelinePath = args[7];
        String pipelineName = args[8];
        String handlerName =
"org.example.SageMakerLambdaFunction::handleRequest";

        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        LambdaClient lambdaClient = LambdaClient.builder()
                .region(region)
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
```

```
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
        System.out.println(
                "\nThis example workflow will guide you through setting up and
running an" +
                        "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
                        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
                        "\nreverse geocode addresses in an input file and store
the results in an export file.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
        String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
        String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

        String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
                handlerName);
        String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
        System.out.println("The queue URL is " + queueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setting up bucket " + bucketName);
        if (!checkBucket(s3Client, bucketName)) {
            setupBucket(s3Client, bucketName);
            System.out.println("Put " + lnglatData + " into " + bucketName);
            putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Now we can create and run our pipeline.");
        setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
```

```
        String pipelineExecutionARN = executePipeline(sageMakerClient,
 bucketName, queueUrl, sageMakerRoleArn,
                pipelineName);
        System.out.println("The pipeline execution ARN value is " +
 pipelineExecutionARN);
        waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
        System.out.println("Getting output results " + bucketName);
        getOutputResults(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The pipeline has completed. To view the pipeline and
 runs " +
                "in SageMaker Studio, follow these instructions:" +
                "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
 Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30
 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }
```

```
   private static void readObject(S3Client s3Client, String bucketName, String
key) {
       System.out.println("Output file contents: \n");
       GetObjectRequest objectRequest = GetObjectRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();

       ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
       byte[] byteArray = objectBytes.asByteArray();
       String text = new String(byteArray, StandardCharsets.UTF_8);
       System.out.println("Text output: " + text);
   }

   // Display some results from the output directory.
   public static void getOutputResults(S3Client s3Client, String bucketName) {
       System.out.println("Getting output results {bucketName}.");
       ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
               .bucket(bucketName)
               .prefix("outputfiles/")
               .build();

       ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
       List<S3Object> s3Objects = response.contents();
       for (S3Object object : s3Objects) {
           readObject(s3Client, bucketName, object.key());
       }
   }

   // Check the status of a pipeline execution.
   public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
           throws InterruptedException {
       String status;
       int index = 0;
       do {
           DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                   .pipelineExecutionArn(executionArn)
                   .build();

           DescribePipelineExecutionResponse response = sageMakerClient
```

```
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " +
status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }

    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }

    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest =
CreatePipelineRequest.builder()
```

```
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
    }

    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
```

```
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";

        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();

        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();
```

```
        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig
\":{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();

        System.out.println("parameter_step_1_vej_config:" +
 gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);

        StartPipelineExecutionRequest pipelineExecutionRequest =
 StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-
execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
 DeleteEventSourceMappingRequest.builder()
                .uuid(eventSourceMapping)
                .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
```

```
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        try {
            for (String policy : lambdaRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific AWS Lambda function.
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("*** " + functionName + " was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Delete the specific S3 bucket.
    public static void deleteBucket(S3Client s3Client, String bucketName) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("*** " + bucketName + " was deleted.");
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
```

```
                deleteBucketObjects(s3, bucketName, myValue.key());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName, String
 objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
                .key(objectName)
                .build());
        try {
            DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(Delete.builder()
                            .objects(toDelete).build())
                    .build();

            s3.deleteObjects(dor);
            System.out.println("*** " + bucketName + " objects were deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific Amazon SQS queue.
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key("samplefiles/" + objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
" + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void setupBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
```

```
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Set up the SQS queue to use with the pipeline.
    public static String setupQueue(SqsClient sqsClient, LambdaClient
lambdaClient, String queueName,
            String lambdaName) {
        System.out.println("Setting up queue named " + queueName);
        try {
            Map<QueueAttributeName, String> queueAtt = new HashMap<>();
            queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
            queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS,
"5");
            queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(queueAtt)
                    .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            TimeUnit.SECONDS.sleep(15);

            connectLambda(sqsClient, lambdaClient,
getQueueUrlResponse.queueUrl(), lambdaName);
            System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
```

```
    }

    // Connect the queue to the Lambda function as an event source.
    public static void connectLambda(SqsClient sqsClient, LambdaClient
lambdaClient, String queueUrl,
            String lambdaName) {
        System.out.println("Connecting the Lambda function and queue for the
pipeline.");
        String queueArn = "";

        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);
        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
            System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
            queueArn = queueAtt.getValue();
        }

        CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
                .eventSourceArn(queueArn)
                .functionName(lambdaName)
                .build();

        CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
        eventSourceMapping = response1.uuid();
        System.out.println("The mapping between the event source and Lambda
function was successful");
    }

    // Create an AWS Lambda function.
    public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
```

```
            String handler) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                    .functionName(functionName)
                    .description("SageMaker example function.")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA11)
                    .timeout(200)
                    .memorySize(1024)
                    .role(role)
                    .build();

            // Create a Lambda function using a waiter.
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The function ARN is " +
functionResponse.functionArn());
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSageMakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        System.out.println("Creating a role to use with SageMaker.");
```

```
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : sageMakerRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    private static String createLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : lambdaRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }
```

```
            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());

        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
            String handler) {
        System.out.println("Create an AWS Lambda function used in this
workflow.");
        String functionArn;
        try {
            // Does this function already exist.
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
            functionArn = response.configuration().functionArn();

        } catch (LambdaException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            functionArn = createLambdaFunction(lambdaClient, functionName,
filePath, role, handler);
        }
        return functionArn;
    }

    // Check to see if the specific S3 bucket exists. If the S3 bucket exists,
this
    // method returns true.
    public static boolean checkBucket(S3Client s3, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
```

```
                    .bucket(bucketName)
                    .build();

            s3.headBucket(headBucketRequest);
            System.out.println(bucketName + " exists");
            return true;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Checks to see if the Amazon SQS queue exists. If not, this method creates
a
    // new queue
    // and returns the ARN value.
    public static String checkQueue(SqsClient sqsClient, LambdaClient
lambdaClient, String queueName,
            String lambdaName) {
        System.out.println("Creating a queue for this use case.");
        String queueUrl;
        try {
            GetQueueUrlRequest request = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
            queueUrl = response.queueUrl();
            System.out.println(queueUrl);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            queueUrl = setupQueue(sqsClient, lambdaClient, queueName,
lambdaName);
        }
        return queueUrl;
    }

    // Checks to see if the Lambda role exists. If not, this method creates it.
    public static String checkLambdaRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS Lambda to use.");
        String roleArn;
        try {
```

```
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createLambdaRole(iam, roleName);
        }
        return roleArn;
    }

    // Checks to see if the SageMaker role exists. If not, this method creates
 it.
    public static String checkSageMakerRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS SageMaker to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createSageMakerRole(iam, roleName);
        }
        return roleArn;
    }

    private static String[] getSageMakerRolePolicies() {
        String[] sageMakerRolePolicies = new String[3];
        sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
        sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
 "AmazonSageMakerGeospatialFullAccess";
        sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        return sageMakerRolePolicies;
```

```
    }

    private static String[] getLambdaRolePolicies() {
        String[] lambdaRolePolicies = new String[5];
        lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
        lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess";
        lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
                + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
        lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole";
        return lambdaRolePolicies;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O trecho do arquivo a seguir contém funções que usam o cliente de SageMaker IA para gerenciar um pipeline.

```
import { readFileSync } from "node:fs";

import {
  CreateRoleCommand,
  DeleteRoleCommand,
  CreatePolicyCommand,
  DeletePolicyCommand,
  AttachRolePolicyCommand,
  DetachRolePolicyCommand,
  GetRoleCommand,
  ListPoliciesCommand,
} from "@aws-sdk/client-iam";

import {
  PublishLayerVersionCommand,
  DeleteLayerVersionCommand,
  CreateFunctionCommand,
  Runtime,
  DeleteFunctionCommand,
  CreateEventSourceMappingCommand,
  DeleteEventSourceMappingCommand,
  GetFunctionCommand,
} from "@aws-sdk/client-lambda";

import {
  PutObjectCommand,
  CreateBucketCommand,
  DeleteBucketCommand,
  DeleteObjectCommand,
  GetObjectCommand,
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

import {
  CreatePipelineCommand,
  DeletePipelineCommand,
  DescribePipelineCommand,
  DescribePipelineExecutionCommand,
  PipelineExecutionStatus,
  StartPipelineExecutionCommand,
} from "@aws-sdk/client-sagemaker";
```

```
import { VectorEnrichmentJobDocumentType } from "@aws-sdk/client-sagemaker-
geospatial";

import {
  CreateQueueCommand,
  DeleteQueueCommand,
  GetQueueAttributesCommand,
  GetQueueUrlCommand,
} from "@aws-sdk/client-sqs";

import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * Create the AWS IAM role that will be assumed by AWS Lambda.
 * @param {{ name: string, iamClient: import('@aws-sdk/client-iam').IAMClient }}
 props
 */
export async function createLambdaExecutionRole({ name, iamClient }) {
  const createRole = () =>
    iamClient.send(
      new CreateRoleCommand({
        RoleName: name,
        AssumeRolePolicyDocument: JSON.stringify({
          Version: "2012-10-17",
          Statement: [
            {
              Effect: "Allow",
              Action: ["sts:AssumeRole"],
              Principal: { Service: ["lambda.amazonaws.com"] },
            },
          ],
        }),
      }),
    );

  let role = null;

  try {
    const { Role } = await createRole();
    role = Role;
  } catch (caught) {
    if (
      caught instanceof Error &&
```

```
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Role } = await iamClient.send(
        new GetRoleCommand({ RoleName: name }),
      );
      role = Role;
    } else {
      throw caught;
    }
  }

  return {
    arn: role.Arn,
    cleanUp: async () => {
      await iamClient.send(new DeleteRoleCommand({ RoleName: name }));
    },
  };
}

/**
 * Create an AWS IAM policy that will be attached to the AWS IAM role assumed by
 the AWS Lambda function.
 * The policy grants permission to work with Amazon SQS, Amazon CloudWatch, and
 Amazon SageMaker.
 * @param {{name: string, iamClient: import('@aws-sdk/client-iam').IAMClient,
 pipelineExecutionRoleArn: string}} props
 */
export async function createLambdaExecutionPolicy({
  name,
  iamClient,
  pipelineExecutionRoleArn,
}) {
  const policyConfig = {
    Version: "2012-10-17",
    Statement: [
      {
        Effect: "Allow",
        Action: [
          "sqs:ReceiveMessage",
          "sqs:DeleteMessage",
          "sqs:GetQueueAttributes",
          "logs:CreateLogGroup",
          "logs:CreateLogStream",
          "logs:PutLogEvents",
```

```
          "sagemaker-geospatial:StartVectorEnrichmentJob",
          "sagemaker-geospatial:GetVectorEnrichmentJob",
          "sagemaker:SendPipelineExecutionStepFailure",
          "sagemaker:SendPipelineExecutionStepSuccess",
          "sagemaker-geospatial:ExportVectorEnrichmentJob",
        ],
        Resource: "*",
      },
      {
        Effect: "Allow",
        // The AWS Lambda function needs permission to pass the pipeline
 execution role to
        // the StartVectorEnrichmentCommand. This restriction prevents an AWS
 Lambda function
        // from elevating privileges. For more information, see:
        // https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_use_passrole.html
        Action: ["iam:PassRole"],
        Resource: `${pipelineExecutionRoleArn}`,
        Condition: {
          StringEquals: {
            "iam:PassedToService": [
              "sagemaker.amazonaws.com",
              "sagemaker-geospatial.amazonaws.com",
            ],
          },
        },
      },
    ],
  };

  const createPolicy = () =>
    iamClient.send(
      new CreatePolicyCommand({
        PolicyDocument: JSON.stringify(policyConfig),
        PolicyName: name,
      }),
    );

  let policy = null;

  try {
    const { Policy } = await createPolicy();
    policy = Policy;
```

```
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Policies } = await iamClient.send(new ListPoliciesCommand({}));
      if (Policies) {
        policy = Policies.find((p) => p.PolicyName === name);
      } else {
        throw new Error("No policies found.");
      }
    } else {
      throw caught;
    }
  }

  return {
    arn: policy?.Arn,
    policyConfig,
    cleanUp: async () => {
      await iamClient.send(new DeletePolicyCommand({ PolicyArn: policy?.Arn }));
    },
  };
}

/**
 * Attach an AWS IAM policy to an AWS IAM role.
 * @param {{roleName: string, policyArn: string, iamClient: import('@aws-sdk/
client-iam').IAMClient}} props
 */
export async function attachPolicy({ roleName, policyArn, iamClient }) {
  const attachPolicyCommand = new AttachRolePolicyCommand({
    RoleName: roleName,
    PolicyArn: policyArn,
  });

  await iamClient.send(attachPolicyCommand);
  return {
    cleanUp: async () => {
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: roleName,
          PolicyArn: policyArn,
        }),
```

```
      );
    },
  };
}

/**
 * Create an AWS Lambda layer that contains the Amazon SageMaker and Amazon
 SageMaker Geospatial clients
 * in the runtime. The default runtime supports v3.188.0 of the JavaScript SDK.
 The Amazon SageMaker
 * Geospatial client wasn't introduced until v3.221.0.
 * @param {{ name: string, lambdaClient: import('@aws-sdk/client-
lambda').LambdaClient }} props
 */
export async function createLambdaLayer({ name, lambdaClient }) {
  const layerPath = `${dirnameFromMetaUrl(import.meta.url)}lambda/nodejs.zip`;
  const { LayerVersionArn, Version } = await lambdaClient.send(
    new PublishLayerVersionCommand({
      LayerName: name,
      Content: {
        ZipFile: Uint8Array.from(readFileSync(layerPath)),
      },
    }),
  );

  return {
    versionArn: LayerVersionArn,
    version: Version,
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteLayerVersionCommand({
          LayerName: name,
          VersionNumber: Version,
        }),
      );
    },
  };
}

/**
 * Deploy the AWS Lambda function that will be used to respond to Amazon
 SageMaker pipeline
 * execution steps.
```

```
 * @param {{roleArn: string, name: string, lambdaClient: import('@aws-sdk/client-
lambda').LambdaClient, layerVersionArn: string}} props
 */
export async function createLambdaFunction({
  name,
  roleArn,
  lambdaClient,
  layerVersionArn,
}) {
  const lambdaPath = `${dirnameFromMetaUrl(
    import.meta.url,
  )}lambda/dist/index.mjs.zip`;

  // If a function of the same name already exists, return that
  // function's ARN instead. By default this is
  // "sagemaker-wkflw-lambda-function", so collisions are
  // unlikely.
  const createFunction = async () => {
    try {
      return await lambdaClient.send(
        new CreateFunctionCommand({
          Code: {
            ZipFile: Uint8Array.from(readFileSync(lambdaPath)),
          },
          Runtime: Runtime.nodejs18x,
          Handler: "index.handler",
          Layers: [layerVersionArn],
          FunctionName: name,
          Role: roleArn,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "ResourceConflictException"
      ) {
        const { Configuration } = await lambdaClient.send(
          new GetFunctionCommand({ FunctionName: name }),
        );
        return Configuration;
      }
      throw caught;
    }
  };
```

```
  // Function creation fails if the Role is not ready. This retries
  // function creation until it succeeds or it times out.
  const { FunctionArn } = await retry(
    { intervalInMs: 1000, maxRetries: 60 },
    createFunction,
  );

  return {
    arn: FunctionArn,
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteFunctionCommand({ FunctionName: name }),
      );
    },
  };
}

/**
 * This uploads some sample coordinate data to an Amazon S3 bucket.
 * The Amazon SageMaker Geospatial vector enrichment job will take the simple
 Lat/Long
 * coordinates in this file and augment them with more detailed location data.
 * @param {{bucketName: string, s3Client: import('@aws-sdk/client-s3').S3Client}}
 props
 */
export async function uploadCSVDataToS3({ bucketName, s3Client }) {
  const s3Path = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../../scenarios/features/sagemaker_pipelines/resources/
latlongtest.csv`;

  await s3Client.send(
    new PutObjectCommand({
      Bucket: bucketName,
      Key: "input/sample_data.csv",
      Body: readFileSync(s3Path),
    }),
  );
}

/**
 * Create the AWS IAM role that will be assumed by the Amazon SageMaker pipeline.
```

```
 * @param {{name: string, iamClient: import('@aws-sdk/client-iam').IAMClient,
 wait: (ms: number) => Promise<void>}} props
 */
export async function createSagemakerRole({ name, iamClient, wait }) {
  let role = null;

  const createRole = () =>
    iamClient.send(
      new CreateRoleCommand({
        RoleName: name,
        AssumeRolePolicyDocument: JSON.stringify({
          Version: "2012-10-17",
          Statement: [
            {
              Effect: "Allow",
              Action: ["sts:AssumeRole"],
              Principal: {
                Service: [
                  "sagemaker.amazonaws.com",
                  "sagemaker-geospatial.amazonaws.com",
                ],
              },
            },
          ],
        }),
      }),
    );

  try {
    const { Role } = await createRole();
    role = Role;
    // Wait for the role to be ready.
    await wait(10);
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Role } = await iamClient.send(
        new GetRoleCommand({ RoleName: name }),
      );
      role = Role;
    } else {
      throw caught;
```

```
    }
  }

  return {
    arn: role.Arn,
    cleanUp: async () => {
      await iamClient.send(new DeleteRoleCommand({ RoleName: name }));
    },
  };
}

/**
 * Create the Amazon SageMaker execution policy. This policy grants permission to
 * invoke the AWS Lambda function, read/write to the Amazon S3 bucket, and send
 messages to
 * the Amazon SQS queue.
 * @param {{ name: string, sqsQueueArn: string, lambdaArn: string, iamClient:
 import('@aws-sdk/client-iam').IAMClient, s3BucketName: string}} props
 */
export async function createSagemakerExecutionPolicy({
  sqsQueueArn,
  lambdaArn,
  iamClient,
  name,
  s3BucketName,
}) {
  const policyConfig = {
    Version: "2012-10-17",
    Statement: [
      {
        Effect: "Allow",
        Action: ["lambda:InvokeFunction"],
        Resource: lambdaArn,
      },
      {
        Effect: "Allow",
        Action: ["s3:*"],
        Resource: [
          `arn:aws:s3:::${s3BucketName}`,
          `arn:aws:s3:::${s3BucketName}/*`,
        ],
      },
      {
        Effect: "Allow",
```

```
        Action: ["sqs:SendMessage"],
        Resource: sqsQueueArn,
      },
    ],
  };

  const createPolicy = () =>
    iamClient.send(
      new CreatePolicyCommand({
        PolicyDocument: JSON.stringify(policyConfig),
        PolicyName: name,
      }),
    );

  let policy = null;

  try {
    const { Policy } = await createPolicy();
    policy = Policy;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Policies } = await iamClient.send(new ListPoliciesCommand({}));
      if (Policies) {
        policy = Policies.find((p) => p.PolicyName === name);
      } else {
        throw new Error("No policies found.");
      }
    } else {
      throw caught;
    }
  }

  return {
    arn: policy?.Arn,
    policyConfig,
    cleanUp: async () => {
      await iamClient.send(new DeletePolicyCommand({ PolicyArn: policy?.Arn }));
    },
  };
}
```

```
/**
 * Create the Amazon SageMaker pipeline using a JSON pipeline definition. The
 definition
 * can also be provided as an Amazon S3 object using
 PipelineDefinitionS3Location.
 * @param {{roleArn: string, name: string, sagemakerClient: import('@aws-sdk/
client-sagemaker').SageMakerClient}} props
 */
export async function createSagemakerPipeline({
  // Assumes an AWS IAM role has been created for this pipeline.
  roleArn,
  name,
  // Assumes an AWS Lambda function has been created for this pipeline.
  functionArn,
  sagemakerClient,
}) {
  const pipelineDefinition = readFileSync(
    // dirnameFromMetaUrl is a local utility function. You can find its
 implementation
    // on GitHub.
    `${dirnameFromMetaUrl(
      import.meta.url,
    )}../../../../../scenarios/features/sagemaker_pipelines/resources/
GeoSpatialPipeline.json`,
  )
    .toString()
    .replace(/\*FUNCTION_ARN\*/g, functionArn);

  let arn = null;

  const createPipeline = () =>
    sagemakerClient.send(
      new CreatePipelineCommand({
        PipelineName: name,
        PipelineDefinition: pipelineDefinition,
        RoleArn: roleArn,
      }),
    );

  try {
    const { PipelineArn } = await createPipeline();
    arn = PipelineArn;
  } catch (caught) {
    if (
```

```
      caught instanceof Error &&
      caught.name === "ValidationException" &&
      caught.message.includes(
        "Pipeline names must be unique within an AWS account and region",
      )
    ) {
      const { PipelineArn } = await sagemakerClient.send(
        new DescribePipelineCommand({ PipelineName: name }),
      );
      arn = PipelineArn;
    } else {
      throw caught;
    }
  }

  return {
    arn,
    cleanUp: async () => {
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
    },
  };
}

/**
 * Create an Amazon SQS queue. The Amazon SageMaker pipeline will send messages
 * to this queue that are then processed by the AWS Lambda function.
 * @param {{name: string, sqsClient: import('@aws-sdk/client-sqs').SQSClient}}
 props
 */
export async function createSQSQueue({ name, sqsClient }) {
  const createSqsQueue = () =>
    sqsClient.send(
      new CreateQueueCommand({
        QueueName: name,
        Attributes: {
          DelaySeconds: "5",
          ReceiveMessageWaitTimeSeconds: "5",
          VisibilityTimeout: "300",
        },
      }),
    );
```

```
  let queueUrl = null;
  try {
    const { QueueUrl } = await createSqsQueue();
    queueUrl = QueueUrl;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "QueueNameExists") {
      const { QueueUrl } = await sqsClient.send(
        new GetQueueUrlCommand({ QueueName: name }),
      );
      queueUrl = QueueUrl;
    } else {
      throw caught;
    }
  }

  const { Attributes } = await retry(
    { intervalInMs: 1000, maxRetries: 60 },
    () =>
      sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: queueUrl,
          AttributeNames: ["QueueArn"],
        }),
      ),
  );

  return {
    queueUrl,
    queueArn: Attributes.QueueArn,
    cleanUp: async () => {
      await sqsClient.send(new DeleteQueueCommand({ QueueUrl: queueUrl }));
    },
  };
}

/**
 * Configure the AWS Lambda function to long poll for messages from the Amazon
 SQS
 * queue.
 * @param {{
 *   paginateListEventSourceMappings: () => Generator<import('@aws-sdk/client-
lambda').ListEventSourceMappingsCommandOutput>,
 *   lambdaName: string,
 *   queueArn: string,
```

```
 *   lambdaClient: import('@aws-sdk/client-lambda').LambdaClient}} props
 */
export async function configureLambdaSQSEventSource({
  lambdaName,
  queueArn,
  lambdaClient,
  paginateListEventSourceMappings,
}) {
  let uuid = null;
  const createEvenSourceMapping = () =>
    lambdaClient.send(
      new CreateEventSourceMappingCommand({
        EventSourceArn: queueArn,
        FunctionName: lambdaName,
      }),
    );

  try {
    const { UUID } = await createEvenSourceMapping();
    uuid = UUID;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceConflictException"
    ) {
      const paginator = paginateListEventSourceMappings(
        { client: lambdaClient },
        {},
      );
      /**
       * @type {import('@aws-sdk/client-
lambda').EventSourceMappingConfiguration[]}
       */
      const eventSourceMappings = [];
      for await (const page of paginator) {
        eventSourceMappings.concat(page.EventSourceMappings || []);
      }

      const { Configuration } = await lambdaClient.send(
        new GetFunctionCommand({ FunctionName: lambdaName }),
      );

      uuid = eventSourceMappings.find(
        (mapping) =>
```

```
          mapping.EventSourceArn === queueArn &&
          mapping.FunctionArn === Configuration.FunctionArn,
      ).UUID;
    } else {
      throw caught;
    }
  }

  return {
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteEventSourceMappingCommand({
          UUID: uuid,
        }),
      );
    },
  };
}

/**
 * Create an Amazon S3 bucket that will store the simple coordinate file as input
 * and the output of the Amazon SageMaker Geospatial vector enrichment job.
 * @param {{
 *   s3Client: import('@aws-sdk/client-s3').S3Client,
 *   name: string,
 *   paginateListObjectsV2: () => Generator<import('@aws-sdk/client-
s3').ListObjectsCommandOutput>
 * }} props
 */
export async function createS3Bucket({
  name,
  s3Client,
  paginateListObjectsV2,
}) {
  await s3Client.send(new CreateBucketCommand({ Bucket: name }));

  return {
    cleanUp: async () => {
      const paginator = paginateListObjectsV2(
        { client: s3Client },
        { Bucket: name },
      );
      for await (const page of paginator) {
        const objects = page.Contents;
```

```
        if (objects) {
          for (const object of objects) {
            await s3Client.send(
              new DeleteObjectCommand({ Bucket: name, Key: object.Key }),
            );
          }
        }
      }
      await s3Client.send(new DeleteBucketCommand({ Bucket: name }));
    },
  };
}

/**
 * Start the execution of the Amazon SageMaker pipeline. Parameters that are
 * passed in are used in the AWS Lambda function.
 * @param {{
 *   name: string,
 *   sagemakerClient: import('@aws-sdk/client-sagemaker').SageMakerClient,
 *   roleArn: string,
 *   queueUrl: string,
 *   s3InputBucketName: string,
 * }} props
 */
export async function startPipelineExecution({
  sagemakerClient,
  name,
  bucketName,
  roleArn,
  queueUrl,
}) {
  /**
   * The Vector Enrichment Job requests CSV data. This configuration points to a
 CSV
   * file in an Amazon S3 bucket.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobInputConfig}
   */
  const inputConfig = {
    DataSourceConfig: {
      S3Data: {
        S3Uri: `s3://${bucketName}/input/sample_data.csv`,
      },
    },
```

```
    DocumentType: VectorEnrichmentJobDocumentType.CSV,
  };

  /**
   * The Vector Enrichment Job adds additional data to the source CSV. This
 configuration points
   * to an Amazon S3 prefix where the output will be stored.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").ExportVectorEnrichmentJobOutputConfig}
   */
  const outputConfig = {
    S3Data: {
      S3Uri: `s3://${bucketName}/output/`,
    },
  };

  /**
   * This job will be a Reverse Geocoding Vector Enrichment Job. Reverse
 Geocoding requires
   * latitude and longitude values.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobConfig}
   */
  const jobConfig = {
    ReverseGeocodingConfig: {
      XAttributeName: "Longitude",
      YAttributeName: "Latitude",
    },
  };

  const { PipelineExecutionArn } = await sagemakerClient.send(
    new StartPipelineExecutionCommand({
      PipelineName: name,
      PipelineExecutionDisplayName: `${name}-example-execution`,
      PipelineParameters: [
        { Name: "parameter_execution_role", Value: roleArn },
        { Name: "parameter_queue_url", Value: queueUrl },
        {
          Name: "parameter_vej_input_config",
          Value: JSON.stringify(inputConfig),
        },
        {
          Name: "parameter_vej_export_config",
          Value: JSON.stringify(outputConfig),
```

```
        },
        {
          Name: "parameter_step_1_vej_config",
          Value: JSON.stringify(jobConfig),
        },
      ],
    }),
  );

  return {
    arn: PipelineExecutionArn,
  };
}

/**
 * Poll the executing pipeline until the status is 'SUCCEEDED', 'STOPPED', or
 'FAILED'.
 * @param {{ arn: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient, wait: (ms: number) => Promise<void>}} props
 */
export async function waitForPipelineComplete({ arn, sagemakerClient, wait }) {
  const command = new DescribePipelineExecutionCommand({
    PipelineExecutionArn: arn,
  });

  let complete = false;
  const intervalInSeconds = 15;
  const COMPLETION_STATUSES = [
    PipelineExecutionStatus.FAILED,
    PipelineExecutionStatus.STOPPED,
    PipelineExecutionStatus.SUCCEEDED,
  ];

  do {
    const { PipelineExecutionStatus: status, FailureReason } =
      await sagemakerClient.send(command);

    complete = COMPLETION_STATUSES.includes(status);

    if (!complete) {
      console.log(
        `Pipeline is ${status}. Waiting ${intervalInSeconds} seconds before
 checking again.`,
      );
```

```
      await wait(intervalInSeconds);
    } else if (status === PipelineExecutionStatus.FAILED) {
      throw new Error(`Pipeline failed because: ${FailureReason}`);
    } else if (status === PipelineExecutionStatus.STOPPED) {
      throw new Error("Pipeline was forcefully stopped.");
    } else {
      console.log(`Pipeline execution ${status}.`);
    }
  } while (!complete);
}

/**
 * Return the string value of an Amazon S3 object.
 * @param {{ bucket: string, key: string, s3Client: import('@aws-sdk/client-
s3').S3Client}} param0
 */
export async function getObject({ bucket, s3Client }) {
  const prefix = "output/";
  const { Contents } = await s3Client.send(
    new ListObjectsV2Command({ MaxKeys: 1, Bucket: bucket, Prefix: prefix }),
  );

  if (!Contents.length) {
    throw new Error("No objects found in bucket.");
  }

  // Find the CSV file.
  const outputObject = Contents.find((obj) => obj.Key.endsWith(".csv"));

  if (!outputObject) {
    throw new Error(`No CSV file found in bucket with the prefix "${prefix}".`);
  }

  const { Body } = await s3Client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: outputObject.Key,
    }),
  );

  return Body.transformToString();
}
```

Essa função é um trecho de um arquivo que usa as funções anteriores da biblioteca para configurar um pipeline de SageMaker IA, executá-lo e excluir todos os recursos criados.

```
import { retry, wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  attachPolicy,
  configureLambdaSQSEventSource,
  createLambdaExecutionPolicy,
  createLambdaExecutionRole,
  createLambdaFunction,
  createLambdaLayer,
  createS3Bucket,
  createSQSQueue,
  createSagemakerExecutionPolicy,
  createSagemakerPipeline,
  createSagemakerRole,
  getObject,
  startPipelineExecution,
  uploadCSVDataToS3,
  waitForPipelineComplete,
} from "./lib.js";
import { MESSAGES } from "./messages.js";

export class SageMakerPipelinesWkflw {
  names = {
    LAMBDA_EXECUTION_ROLE: "sagemaker-wkflw-lambda-execution-role",
    LAMBDA_EXECUTION_ROLE_POLICY:
      "sagemaker-wkflw-lambda-execution-role-policy",
    LAMBDA_FUNCTION: "sagemaker-wkflw-lambda-function",
    LAMBDA_LAYER: "sagemaker-wkflw-lambda-layer",
    SAGE_MAKER_EXECUTION_ROLE: "sagemaker-wkflw-pipeline-execution-role",
    SAGE_MAKER_EXECUTION_ROLE_POLICY:
      "sagemaker-wkflw-pipeline-execution-role-policy",
    SAGE_MAKER_PIPELINE: "sagemaker-wkflw-pipeline",
    SQS_QUEUE: "sagemaker-wkflw-sqs-queue",
    S3_BUCKET: `sagemaker-wkflw-s3-bucket-${Date.now()}`,
  };

  cleanUpFunctions = [];

  /**
   * @param {import("@aws-doc-sdk-examples/lib/prompter.js").Prompter} prompter
   * @param {import("@aws-doc-sdk-examples/lib/logger.js").Logger} logger
```

```
   * @param {{ IAM: import("@aws-sdk/client-iam").IAMClient, Lambda:
 import("@aws-sdk/client-lambda").LambdaClient, SageMaker: import("@aws-sdk/
client-sagemaker").SageMakerClient, S3: import("@aws-sdk/client-s3").S3Client,
 SQS: import("@aws-sdk/client-sqs").SQSClient }} clients
   */
  constructor(prompter, logger, clients) {
    this.prompter = prompter;
    this.logger = logger;
    this.clients = clients;
  }

  async run() {
    try {
      await this.startWorkflow();
    } catch (err) {
      console.error(err);
      throw err;
    } finally {
      this.logger.logSeparator();
      const doCleanUp = await this.prompter.confirm({
        message: "Clean up resources?",
      });
      if (doCleanUp) {
        await this.cleanUp();
      }
    }
  }

  async cleanUp() {
    // Run all of the clean up functions. If any fail, we log the error and
 continue.
    // This ensures all clean up functions are run.
    for (let i = this.cleanUpFunctions.length - 1; i >= 0; i--) {
      await retry(
        { intervalInMs: 1000, maxRetries: 60, swallowError: true },
        this.cleanUpFunctions[i],
      );
    }
  }

  async startWorkflow() {
    this.logger.logSeparator(MESSAGES.greetingHeader);
    await this.logger.log(MESSAGES.greeting);
```

```
    this.logger.logSeparator();
    await this.logger.log(
      MESSAGES.creatingRole.replace(
        "${ROLE_NAME}",
        this.names.LAMBDA_EXECUTION_ROLE,
      ),
    );

    // Create an IAM role that will be assumed by the AWS Lambda function. This
function
    // is triggered by Amazon SQS messages and calls SageMaker and SageMaker
GeoSpatial actions.
    const { arn: lambdaExecutionRoleArn, cleanUp: lambdaExecutionRoleCleanUp } =
      await createLambdaExecutionRole({
        name: this.names.LAMBDA_EXECUTION_ROLE,
        iamClient: this.clients.IAM,
      });
    // Add a clean up step to a stack for every resource created.
    this.cleanUpFunctions.push(lambdaExecutionRoleCleanUp);

    await this.logger.log(
      MESSAGES.roleCreated.replace(
        "${ROLE_NAME}",
        this.names.LAMBDA_EXECUTION_ROLE,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingRole.replace(
        "${ROLE_NAME}",
        this.names.SAGE_MAKER_EXECUTION_ROLE,
      ),
    );

    // Create an IAM role that will be assumed by the SageMaker pipeline. The
pipeline
    // sends messages to an Amazon SQS queue and puts/retrieves Amazon S3
objects.
    const {
      arn: pipelineExecutionRoleArn,
      cleanUp: pipelineExecutionRoleCleanUp,
    } = await createSagemakerRole({
```

```
      iamClient: this.clients.IAM,
      name: this.names.SAGE_MAKER_EXECUTION_ROLE,
      wait,
    });
    this.cleanUpFunctions.push(pipelineExecutionRoleCleanUp);

    await this.logger.log(
      MESSAGES.roleCreated.replace(
        "${ROLE_NAME}",
        this.names.SAGE_MAKER_EXECUTION_ROLE,
      ),
    );

    this.logger.logSeparator();

    // Create an IAM policy that allows the AWS Lambda function to invoke
 SageMaker APIs.
    const {
      arn: lambdaExecutionPolicyArn,
      policy: lambdaPolicy,
      cleanUp: lambdaExecutionPolicyCleanUp,
    } = await createLambdaExecutionPolicy({
      name: this.names.LAMBDA_EXECUTION_ROLE_POLICY,
      s3BucketName: this.names.S3_BUCKET,
      iamClient: this.clients.IAM,
      pipelineExecutionRoleArn,
    });
    this.cleanUpFunctions.push(lambdaExecutionPolicyCleanUp);

    console.log(JSON.stringify(lambdaPolicy, null, 2), "\n");

    await this.logger.log(
      MESSAGES.attachPolicy
        .replace("${POLICY_NAME}", this.names.LAMBDA_EXECUTION_ROLE_POLICY)
        .replace("${ROLE_NAME}", this.names.LAMBDA_EXECUTION_ROLE),
    );

    await this.prompter.checkContinue();

    // Attach the Lambda execution policy to the execution role.
    const { cleanUp: lambdaExecutionRolePolicyCleanUp } = await attachPolicy({
      roleName: this.names.LAMBDA_EXECUTION_ROLE,
      policyArn: lambdaExecutionPolicyArn,
      iamClient: this.clients.IAM,
```

```
    });
    this.cleanUpFunctions.push(lambdaExecutionRolePolicyCleanUp);

    await this.logger.log(MESSAGES.policyAttached);

    this.logger.logSeparator();

    // Create Lambda layer for SageMaker packages.
    const { versionArn: layerVersionArn, cleanUp: lambdaLayerCleanUp } =
      await createLambdaLayer({
        name: this.names.LAMBDA_LAYER,
        lambdaClient: this.clients.Lambda,
      });
    this.cleanUpFunctions.push(lambdaLayerCleanUp);

    await this.logger.log(
      MESSAGES.creatingFunction.replace(
        "${FUNCTION_NAME}",
        this.names.LAMBDA_FUNCTION,
      ),
    );

    // Create the Lambda function with the execution role.
    const { arn: lambdaArn, cleanUp: lambdaCleanUp } =
      await createLambdaFunction({
        roleArn: lambdaExecutionRoleArn,
        lambdaClient: this.clients.Lambda,
        name: this.names.LAMBDA_FUNCTION,
        layerVersionArn,
      });
    this.cleanUpFunctions.push(lambdaCleanUp);

    await this.logger.log(
      MESSAGES.functionCreated.replace(
        "${FUNCTION_NAME}",
        this.names.LAMBDA_FUNCTION,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingSQSQueue.replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );
```

```
    // Create an SQS queue for the SageMaker pipeline.
    const {
      queueUrl,
      queueArn,
      cleanUp: queueCleanUp,
    } = await createSQSQueue({
      name: this.names.SQS_QUEUE,
      sqsClient: this.clients.SQS,
    });
    this.cleanUpFunctions.push(queueCleanUp);

    await this.logger.log(
      MESSAGES.sqsQueueCreated.replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.configuringLambdaSQSEventSource
        .replace("${LAMBDA_NAME}", this.names.LAMBDA_FUNCTION)
        .replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    // Configure the SQS queue as an event source for the Lambda.
    const { cleanUp: lambdaSQSEventSourceCleanUp } =
      await configureLambdaSQSEventSource({
        lambdaArn,
        lambdaName: this.names.LAMBDA_FUNCTION,
        queueArn,
        sqsClient: this.clients.SQS,
        lambdaClient: this.clients.Lambda,
      });
    this.cleanUpFunctions.push(lambdaSQSEventSourceCleanUp);

    await this.logger.log(
      MESSAGES.lambdaSQSEventSourceConfigured
        .replace("${LAMBDA_NAME}", this.names.LAMBDA_FUNCTION)
        .replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    this.logger.logSeparator();
```

```
    // Create an IAM policy that allows the SageMaker pipeline to invoke AWS
Lambda
    // and send messages to the Amazon SQS queue.
    const {
      arn: pipelineExecutionPolicyArn,
      policy: sagemakerPolicy,
      cleanUp: pipelineExecutionPolicyCleanUp,
    } = await createSagemakerExecutionPolicy({
      sqsQueueArn: queueArn,
      lambdaArn,
      iamClient: this.clients.IAM,
      name: this.names.SAGE_MAKER_EXECUTION_ROLE_POLICY,
      s3BucketName: this.names.S3_BUCKET,
    });
    this.cleanUpFunctions.push(pipelineExecutionPolicyCleanUp);

    console.log(JSON.stringify(sagemakerPolicy, null, 2));

    await this.logger.log(
      MESSAGES.attachPolicy
        .replace("${POLICY_NAME}", this.names.SAGE_MAKER_EXECUTION_ROLE_POLICY)
        .replace("${ROLE_NAME}", this.names.SAGE_MAKER_EXECUTION_ROLE),
    );

    await this.prompter.checkContinue();

    // Attach the SageMaker execution policy to the execution role.
    const { cleanUp: pipelineExecutionRolePolicyCleanUp } = await attachPolicy({
      roleName: this.names.SAGE_MAKER_EXECUTION_ROLE,
      policyArn: pipelineExecutionPolicyArn,
      iamClient: this.clients.IAM,
    });
    this.cleanUpFunctions.push(pipelineExecutionRolePolicyCleanUp);
    // Wait for the role to be ready. If the role is used immediately,
    // the pipeline will fail.
    await wait(5);

    await this.logger.log(MESSAGES.policyAttached);

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingPipeline.replace(
        "${PIPELINE_NAME}",
```

```
        this.names.SAGE_MAKER_PIPELINE,
      ),
    );

    // Create the SageMaker pipeline.
    const { cleanUp: pipelineCleanUp } = await createSagemakerPipeline({
      roleArn: pipelineExecutionRoleArn,
      functionArn: lambdaArn,
      sagemakerClient: this.clients.SageMaker,
      name: this.names.SAGE_MAKER_PIPELINE,
    });
    this.cleanUpFunctions.push(pipelineCleanUp);

    await this.logger.log(
      MESSAGES.pipelineCreated.replace(
        "${PIPELINE_NAME}",
        this.names.SAGE_MAKER_PIPELINE,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingS3Bucket.replace("${BUCKET_NAME}", this.names.S3_BUCKET),
    );

    // Create an S3 bucket for storing inputs and outputs.
    const { cleanUp: s3BucketCleanUp } = await createS3Bucket({
      name: this.names.S3_BUCKET,
      s3Client: this.clients.S3,
    });
    this.cleanUpFunctions.push(s3BucketCleanUp);

    await this.logger.log(
      MESSAGES.s3BucketCreated.replace("${BUCKET_NAME}", this.names.S3_BUCKET),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.uploadingInputData.replace(
        "${BUCKET_NAME}",
        this.names.S3_BUCKET,
      ),
```

```
    );

    // Upload CSV Lat/Long data to S3.
    await uploadCSVDataToS3({
      bucketName: this.names.S3_BUCKET,
      s3Client: this.clients.S3,
    });

    await this.logger.log(MESSAGES.inputDataUploaded);

    this.logger.logSeparator();

    await this.prompter.checkContinue(MESSAGES.executePipeline);

    // Execute the SageMaker pipeline.
    const { arn: pipelineExecutionArn } = await startPipelineExecution({
      name: this.names.SAGE_MAKER_PIPELINE,
      sagemakerClient: this.clients.SageMaker,
      roleArn: pipelineExecutionRoleArn,
      bucketName: this.names.S3_BUCKET,
      queueUrl,
    });

    // Wait for the pipeline execution to finish.
    await waitForPipelineComplete({
      arn: pipelineExecutionArn,
      sagemakerClient: this.clients.SageMaker,
      wait,
    });

    this.logger.logSeparator();

    await this.logger.log(MESSAGES.outputDelay);

    // The getOutput function will throw an error if the output is not
    // found. The retry function will retry a failed function call once
    // ever 10 seconds for 2 minutes.
    const output = await retry({ intervalInMs: 10000, maxRetries: 12 }, () =>
      getObject({
        bucket: this.names.S3_BUCKET,
        s3Client: this.clients.S3,
      }),
    );
```

```
    this.logger.logSeparator();
    await this.logger.log(MESSAGES.outputDataRetrieved);
    console.log(output.split("\n").slice(0, 6).join("\n"));
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
          <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
 <queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
 <pipelineName>

    Where:
        sageMakerRoleName - The name of the Amazon SageMaker role.
        lambdaRoleName - The name of the AWS Lambda role.
```

```
        functionName - The name of the AWS Lambda function (for
example,SageMakerExampleFunction).
        functionKey - The name of the Amazon S3 key name that represents the
Lambda function (for example, SageMakerLambda.zip).
        queueName - The name of the Amazon Simple Queue Service (Amazon SQS)
queue.
        bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
bucket.
        bucketFunction - The name of the Amazon S3 bucket that contains the
Lambda ZIP file.
        lnglatData - The file location of the latlongtest.csv file required for
this use case.
        spatialPipelinePath - The file location of the GeoSpatialPipeline.json
file required for this use case.
        pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).
    """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]
    val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

    println(DASHES)
    println("Welcome to the Amazon SageMaker pipeline example scenario.")
    println(
        """
         This example workflow will guide you through setting up and running an
         Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and
an
         Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
```

```
        reverse geocode addresses in an input file and store the results in an
export file.
       """.trimIndent(),
   )
   println(DASHES)

   println(DASHES)
   println("First, we will set up the roles, functions, and queue needed by the
SageMaker pipeline.")
   val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
   val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
   val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
   val queueUrl = checkQueue(queueName, functionName)
   println(DASHES)

   println(DASHES)
   println("Setting up bucket $bucketName")
   if (!checkBucket(bucketName)) {
       setupBucket(bucketName)
       println("Put $lnglatData into $bucketName")
       val objectKey = "samplefiles/latlongtest.csv"
       putS3Object(bucketName, objectKey, lnglatData)
   }
   println(DASHES)

   println(DASHES)
   println("Now we can create and run our pipeline.")
   setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn,
pipelineName)
   val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
   println("The pipeline execution ARN value is $pipelineExecutionARN")
   waitForPipelineExecution(pipelineExecutionARN)
   println("Wait 30 secs to get output results $bucketName")
   TimeUnit.SECONDS.sleep(30)
   getOutputResults(bucketName)
   println(DASHES)

   println(DASHES)
   println(
       """
            The pipeline has completed. To view the pipeline and runs in
SageMaker Studio, follow these instructions:
```

```
            https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("Do you want to delete the AWS resources used in this Workflow? (y/
n)")
    val `in` = Scanner(System.`in`)
    val delResources = `in`.nextLine()
    if (delResources.compareTo("y") == 0) {
        println("Lets clean up the AWS resources. Wait 30 seconds")
        TimeUnit.SECONDS.sleep(30)
        deleteEventSourceMapping(functionName)
        deleteSQSQueue(queueName)
        listBucketObjects(bucketName)
        deleteBucket(bucketName)
        delLambdaFunction(functionName)
        deleteLambdaRole(lambdaRoleName)
        deleteSagemakerRole(sageMakerRoleName)
        deletePipeline(pipelineName)
    } else {
        println("The AWS Resources were not deleted!")
    }
    println(DASHES)

    println(DASHES)
    println("SageMaker pipeline scenario is complete.")
    println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

```
suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun delLambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
```

```
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}

suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
        bucket = bucketNameVal
    }
```

```
    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
                    eventSourceMapping = event.uuid.toString()
                }
            }
        }
    }
```

```
    val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
        uuid = eventSourceMapping
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
        println("The event mapping is deleted!")
    }
}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3Objects: List<Object>? = response.contents
        if (s3Objects != null) {
            for (`object` in s3Objects) {
                if (bucketName != null) {
                    readObject(bucketName, (`object`.key))
                }
            }
        }
    }
}
```

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn:
 String?, pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }
```

```
    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName
\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
```

```
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }
```

```
        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath:
 String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            println("$bucketName exists")
            return true
        }
    } catch (e: S3Exception) {
        println("Bucket does not exist")
    }
```

```
    return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }
    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
 lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
 successful")
    }
}

// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
    val queueAtt: MutableMap<String, String> = HashMap()
    queueAtt.put("DelaySeconds", "5")
```

```
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
 { queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a
 new queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
 created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
}
```

```
suspend fun createNewFunction(myFunctionName: String, s3BucketName: String,
 myS3Key: String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
 function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val response = lambdaClient.getFunction(functionRequest)
            functionArn = response.configuration?.functionArn.toString()
            println("$functionArn exists")
        }
    } catch (e: LambdaException) {
```

```
        println(e.message + " A new function will be created")
        functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
 myHandler, myRole)
    }
    return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be
 created for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
```

```
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in sageMakerRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        System.out.println("Role ready with ARN ${roleResult.role?.arn}")
        return roleResult.role?.arn.toString()
    }
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be
 created for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
```

```
        roleArn = createLambdaRole(roleNameVal)
    }

    return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
```

```
        println("Role ready with ARN " + roleResult.role?.arn)
        return roleResult.role?.arn.toString()
    }
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<String>(5)
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy"
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
    return lambdaRolePolicies
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Comece a usar modelos e endpoints de SageMaker IA usando um AWS SDK

O exemplo de código a seguir mostra como:

- Comece um trabalho de treinamento e crie um modelo de SageMaker IA.
- Criar uma configuração de endpoint.
- Criar um endpoint e, em seguida, limpar os recursos.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
    DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
    DATA lo_resource_config  TYPE REF TO /aws1/cl_sgmresourceconfig.
    DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
    DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.
    DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.
    DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
    DATA lo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.
    DATA lo_training_result TYPE REF TO /aws1/cl_sgmdescrtrnjobrsp.
    DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
    DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
    DATA lt_hyperparameters TYPE /aws1/
cl_sgmhyperparameters_w=>tt_hyperparameters.
    DATA lv_model_data_url TYPE /aws1/sgmurl.

    lv_model_data_url = iv_s3_output_path && iv_training_job_name && '/output/
model.tar.gz'.

    "Create ABAP internal table for hyperparameters based on input variables."
```

```
   "These hyperparameters are based on Amazon SageMaker built-in algorithm -
XGBoost"
   lo_hyperparameters_w = NEW #( iv_value = iv_hp_max_depth ).
   INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eta ).
   INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eval_metric ).
   INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_scale_pos_weight ).
   INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_subsample ).
   INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_objective ).
   INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_num_round ).
   INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   "Create ABAP internal table for data based on input variables."
   "Training data."
   lo_trn_s3datasource = NEW #( iv_s3datatype = iv_trn_data_s3datatype
                                iv_s3datadistributiontype =
iv_trn_data_s3datadistribution
                                iv_s3uri = iv_trn_data_s3uri ).

   lo_trn_datasource = NEW #( io_s3datasource = lo_trn_s3datasource ).

   lo_trn_channel = NEW #( iv_channelname = 'train'
                           io_datasource = lo_trn_datasource
                           iv_compressiontype = iv_trn_data_compressiontype
                           iv_contenttype = iv_trn_data_contenttype ).
   INSERT lo_trn_channel INTO TABLE lt_input_data_config.
```

```
   "Validation data."
   lo_val_s3datasource = NEW #( iv_s3datatype = iv_val_data_s3datatype
                                iv_s3datadistributiontype =
iv_val_data_s3datadistribution
                                iv_s3uri = iv_val_data_s3uri ).

   lo_val_datasource = NEW #( io_s3datasource = lo_val_s3datasource ).

   lo_val_channel = NEW #( iv_channelname = 'validation'
                           io_datasource = lo_val_datasource
                           iv_compressiontype = iv_val_data_compressiontype
                           iv_contenttype = iv_val_data_contenttype ).
   INSERT lo_val_channel INTO TABLE lt_input_data_config.

   "Create an ABAP object for algorithm specification based on input variables."
   lo_algorithm_specification = NEW #( iv_trainingimage = iv_training_image
                                       iv_traininginputmode =
iv_training_input_mode ).

   "Create an ABAP object for resource configuration."
   lo_resource_config = NEW #( iv_instancecount = iv_instance_count
                               iv_instancetype = iv_instance_type
                               iv_volumesizeingb = iv_volume_sizeingb ).

   "Create an ABAP object for output data configuration."
   lo_output_data_config = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create an ABAP object for stopping condition."
   lo_stopping_condition = NEW #( iv_maxruntimeinseconds =
iv_max_runtime_in_seconds ).

   TRY.
       lo_sgm->createtrainingjob(
         iv_trainingjobname           = iv_training_job_name
         iv_rolearn                   = iv_role_arn
         it_hyperparameters           = lt_hyperparameters
         it_inputdataconfig           = lt_input_data_config
         io_algorithmspecification    = lo_algorithm_specification
         io_outputdataconfig          = lo_output_data_config
         io_resourceconfig            = lo_resource_config
         io_stoppingcondition         = lo_stopping_condition ).
       MESSAGE 'Training job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
```

```
        MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcenotfound.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
    ENDTRY.

    "Wait for training job to be completed."
    lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
    WHILE lo_training_result->get_trainingjobstatus( ) <> 'Completed'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
    ENDWHILE.

    "Create ABAP object for the container image based on input variables."
    lo_primarycontainer = NEW #( iv_image = iv_training_image
                                 iv_modeldataurl = lv_model_data_url ).

    "Create an Amazon SageMaker model."
    TRY.
        lo_sgm->createmodel(
          iv_executionrolearn = iv_role_arn
          iv_modelname = iv_model_name
          io_primarycontainer = lo_primarycontainer ).
        MESSAGE 'Model created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
    ENDTRY.

    "Create an endpoint production variant."
    lo_production_variants = NEW #( iv_variantname = iv_ep_variant_name
                                    iv_modelname = iv_model_name
                                    iv_initialinstancecount =
iv_ep_initial_instance_count
                                    iv_instancetype = iv_ep_instance_type ).
    INSERT lo_production_variants INTO TABLE lt_production_variants.
```

```
   TRY.
       "Create an endpoint configuration."
       lo_ep_config_result = lo_sgm->createendpointconfig(
         iv_endpointconfigname = iv_ep_cfg_name
         it_productionvariants = lt_production_variants ).
       MESSAGE 'Endpoint configuration created.' TYPE 'I'.

       "Create an endpoint."
       oo_ep_output = lo_sgm->createendpoint(        " oo_ep_output is returned
for testing purposes. "
           iv_endpointconfigname = iv_ep_cfg_name
           iv_endpointname = iv_ep_name ).
       MESSAGE 'Endpoint created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
   ENDTRY.

   "Wait for endpoint creation to be completed."
   DATA(lo_endpoint_result) = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
   WHILE lo_endpoint_result->get_endpointstatus( ) <> 'InService'.
     IF sy-index = 30.
       EXIT.               "Maximum 900 seconds."
     ENDIF.
     WAIT UP TO 30 SECONDS.
     lo_endpoint_result = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
   ENDWHILE.

   TRY.
       "Delete an endpoint."
       lo_sgm->deleteendpoint(
           iv_endpointname = iv_ep_name ).
       MESSAGE 'Endpoint deleted' TYPE 'I'.

       "Delete an endpoint configuration."
       lo_sgm->deleteendpointconfig(
         iv_endpointconfigname = iv_ep_cfg_name ).
       MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.

       "Delete model."
       lo_sgm->deletemodel(
                 iv_modelname = iv_model_name ).
```

```
        MESSAGE 'Model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - CreateEndpoint
  - CreateEndpointConfig
  - CreateModel
  - CreateTrainingJob
  - DeleteEndpoint
  - DeleteEndpointConfig
  - DeleteModel
  - DescribeEndpoint
  - DescribeTrainingJob

Conceitos básicos de arquivos de atributos de machine learning

O exemplo de código a seguir mostra como:

- Configurar permissões do IAM.
- Crie uma função SageMaker de execução
- Criar grupo de atributos.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
```

```
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
```

```
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
```

```
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}
```

```
# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
```

```
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1
```

```
# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
```

```
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)
```

```
if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
```

```
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi
```

```
echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
```

```
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole

- ListFeatureGroups
- PutPublicAccessBlock
- Rm

Exemplos de código para o Secrets Manager usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Secrets Manager com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Secrets Manager: mais informações sobre o Secrets Manager.
- Referência da API Secrets Manager: detalhes sobre todas as ações disponíveis do Secrets Manager.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Secrets Manager AWS SDKs
  - Ações para o Secrets Manager usando AWS SDKs
    - Use BatchGetSecretValuecom um AWS SDK ou CLI
    - Use GetSecretValuecom um AWS SDK ou CLI
- Cenários para o uso do Secrets Manager AWS SDKs
  - Criar uma API REST de biblioteca de empréstimos
  - Criação de uma instância de banco de dados relacional

- Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI
- Introdução aos bancos de dados de documentos
- Como mover segredos codificados para o armazenamento seguro de segredos

Exemplos básicos para o uso do Secrets Manager AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Secrets Manager com AWS SDKs.

Exemplos

- Ações para o Secrets Manager usando AWS SDKs
  - Use BatchGetSecretValuecom um AWS SDK ou CLI
  - Use GetSecretValuecom um AWS SDK ou CLI

Ações para o Secrets Manager usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Secrets Manager com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Secrets Manager e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o uso do Secrets Manager AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Secrets Manager.

Exemplos

- Use BatchGetSecretValuecom um AWS SDK ou CLI
- Use GetSecretValuecom um AWS SDK ou CLI

Use **`BatchGetSecretValue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `BatchGetSecretValue`.

CLI

AWS CLI

Exemplo 1: recuperar o valor do segredo de um grupo de segredos listados por nome

O exemplo `batch-get-secret-value` a seguir obtém segredos do valor do segredo para três segredos.

```
aws secretsmanager batch-get-secret-value \
    --secret-id-list MySecret1 MySecret2 MySecret3
```

Saída:

```
{
    "SecretValues": [
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret1-a1b2c3",
            "Name": "MySecret1",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "SecretString": "{\"username\":\"diego_ramirez\",
\"password\":\"EXAMPLE-PASSWORD\",\"engine\":\"mysql\",\"host\":
\"secretsmanagertutorial.cluster.us-west-2.rds.amazonaws.com\",\"port\":3306,
\"dbClusterIdentifier\":\"secretsmanagertutorial\"}",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1523477145.729"
        },
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret2-a1b2c3",
            "Name": "MySecret2",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "SecretString": "{\"username\":\"akua_mansa\",\"password\":\"EXAMPLE-
PASSWORD\"",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1673477781.275"
        },
        {
```

```
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret3-a1b2c3",
            "Name": "MySecret3",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "SecretString": "{\"username\":\"jie_liu\",\"password\":\"EXAMPLE-
PASSWORD\"",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1373477721.124"
        }
    ],
    "Errors": []
}
```

Para obter mais informações, consulte Recuperar um grupo de segredos em um lote no Guia do usuário do AWS Secrets Manager.

Exemplo 2: recuperar o valor do segredo de um grupo de segredos selecionados pelo filtro

O exemplo `batch-get-secret-value` a seguir obtém os segredos do valor do segredo em sua conta que têm `MySecret` no nome. A filtragem por nome diferencia maiúsculas de minúsculas.

```
aws secretsmanager batch-get-secret-value \
    --filters Key="name",Values="MySecret"
```

Saída:

```
{
    "SecretValues": [
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret1-a1b2c3",
            "Name": "MySecret1",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "SecretString": "{\"username\":\"diego_ramirez\",
\"password\":\"EXAMPLE-PASSWORD\",\"engine\":\"mysql\",\"host\":
\"secretsmanagertutorial.cluster.us-west-2.rds.amazonaws.com\",\"port\":3306,
\"dbClusterIdentifier\":\"secretsmanagertutorial\"}",
            "VersionStages": [
                "AWSCURRENT"
```

```
            ],
            "CreatedDate": "1523477145.729"
        },
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret2-a1b2c3",
            "Name": "MySecret2",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "SecretString": "{\"username\":\"akua_mansa\",\"password\":\"EXAMPLE-
PASSWORD\"",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1673477781.275"
        },
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret3-a1b2c3",
            "Name": "MySecret3",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "SecretString": "{\"username\":\"jie_liu\",\"password\":\"EXAMPLE-
PASSWORD\"",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1373477721.124"
        }
    ],
    "Errors": []
}
```

Para obter mais informações, consulte Recuperar um grupo de segredos em um lote no Guia do usuário do AWS Secrets Manager.

- Para obter detalhes da API, consulte BatchGetSecretValueem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BatchGetSecretsWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def batch_get_secrets(self, filter_name):
        """
        Retrieve multiple secrets from AWS Secrets Manager using the
 batch_get_secret_value API.
        This function assumes the stack mentioned in the source code README has
 been successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param filter_name: The full or partial name of secrets to be fetched.
        :type filter_name: str
        """
        try:
            secrets = []
            response = self.client.batch_get_secret_value(
                Filters=[{"Key": "name", "Values": [f"{filter_name}"]}]
            )
            for secret in response["SecretValues"]:
                secrets.append(json.loads(secret["SecretString"]))
            if secrets:
                logger.info("Secrets retrieved successfully.")
            else:
                logger.info("Zero secrets returned without error.")
            return secrets
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"One or more requested secrets were not found with filter:
 {filter_name}"
```

```
            logger.info(msg)
            return msg
        except Exception as e:
            logger.error(f"An unknown error occurred:\n{str(e)}.")
            raise
```

- Para obter detalhes da API, consulte a BatchGetSecretValueReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_filter_name = 'mySecret'
        DATA(lo_result) = lo_smr->batchgetsecretvalue(
          it_filters = VALUE /aws1/cl_smrfilter=>tt_filterslisttype(
            (
              NEW /aws1/cl_smrfilter(
                iv_key = 'name'
                it_values = VALUE /aws1/
cl_smrfiltvalsstrlist_w=>tt_filtervaluesstringlist(
                  ( NEW /aws1/cl_smrfiltvalsstrlist_w( iv_value =
 iv_filter_name ) )
                )
              )
            )
          )
        ).
        ot_secret_values = lo_result->get_secretvalues( ).
        MESSAGE 'Secrets retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_smrresourcenotfoundex.
```

```
      MESSAGE 'One or more requested secrets were not found.' TYPE 'E'.
    CATCH /aws1/cx_smrdecryptionfailure.
      MESSAGE 'Failed to decrypt one or more secrets.' TYPE 'E'.
    CATCH /aws1/cx_smrinvalidparameterex.
      MESSAGE 'Invalid parameter provided.' TYPE 'E'.
    CATCH /aws1/cx_smrinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a BatchGetSecretValuereferência da API AWS SDK for SAP ABAP.

Use **`GetSecretValue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetSecretValue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar uma instância de banco de dados relacional
- Conceitos básicos de data warehouses de tecnologia sem servidor
- Introdução aos bancos de dados de documentos
- Como mover segredos codificados para o armazenamento seguro de segredos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.SecretsManager;
```

```
using Amazon.SecretsManager.Model;

/// <summary>
/// This example uses the Amazon Web Service Secrets Manager to retrieve
/// the secret value for the provided secret name.
/// </summary>
public class GetSecretValue
{
    /// <summary>
    /// The main method initializes the necessary values and then calls
    /// the GetSecretAsync and DecodeString methods to get the decoded
    /// secret value for the secret named in secretName.
    /// </summary>
    public static async Task Main()
    {
        string secretName = "<<{{MySecretName}}>>";
        string secret;

        IAmazonSecretsManager client = new AmazonSecretsManagerClient();

        var response = await GetSecretAsync(client, secretName);

        if (response is not null)
        {
            secret = DecodeString(response);

            if (!string.IsNullOrEmpty(secret))
            {
                Console.WriteLine($"The decoded secret value is: {secret}.");
            }
            else
            {
                Console.WriteLine("No secret value was returned.");
            }
        }
    }

    /// <summary>
    /// Retrieves the secret value given the name of the secret to
    /// retrieve.
    /// </summary>
    /// <param name="client">The client object used to retrieve the secret
    /// value for the given secret name.</param>
```

```
        /// <param name="secretName">The name of the secret value to retrieve.</
param>
        /// <returns>The GetSecretValueReponse object returned by
        /// GetSecretValueAsync.</returns>
        public static async Task<GetSecretValueResponse> GetSecretAsync(
            IAmazonSecretsManager client,
            string secretName)
        {
            GetSecretValueRequest request = new GetSecretValueRequest()
            {
                SecretId = secretName,
                VersionStage = "AWSCURRENT", // VersionStage defaults to
 AWSCURRENT if unspecified.
            };

            GetSecretValueResponse response = null;

            // For the sake of simplicity, this example handles only the most
            // general SecretsManager exception.
            try
            {
                response = await client.GetSecretValueAsync(request);
            }
            catch (AmazonSecretsManagerException e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }

            return response;
        }

        /// <summary>
        /// Decodes the secret returned by the call to GetSecretValueAsync and
        /// returns it to the calling program.
        /// </summary>
        /// <param name="response">A GetSecretValueResponse object containing
        /// the requested secret value returned by GetSecretValueAsync.</param>
        /// <returns>A string representing the decoded secret value.</returns>
        public static string DecodeString(GetSecretValueResponse response)
        {
            // Decrypts secret using the associated AWS Key Management Service
            // Customer Master Key (CMK.) Depending on whether the secret is a
            // string or binary value, one of these fields will be populated.
            if (response.SecretString is not null)
```

```
            {
                var secret = response.SecretString;
                return secret;
            }
            else if (response.SecretBinary is not null)
            {
                var memoryStream = response.SecretBinary;
                StreamReader reader = new StreamReader(memoryStream);
                string decodedBinarySecret =
 System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(reader.ReadToEnd()));
                return decodedBinarySecret;
            }
            else
            {
                return string.Empty;
            }
        }
    }
```

- Para obter detalhes da API, consulte GetSecretValuea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve an AWS Secrets Manager encrypted secret.
/*!
  \param secretID: The ID for the secret.
  \return bool: Function succeeded.
 */
bool AwsDoc::SecretsManager::getSecretValue(const Aws::String &secretID,
```

```
                                                const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SecretsManager::SecretsManagerClient
 secretsManagerClient(clientConfiguration);

    Aws::SecretsManager::Model::GetSecretValueRequest request;
    request.SetSecretId(secretID);

    Aws::SecretsManager::Model::GetSecretValueOutcome getSecretValueOutcome =
 secretsManagerClient.GetSecretValue(
            request);
    if (getSecretValueOutcome.IsSuccess()) {
        std::cout << "Secret is: "
                  << getSecretValueOutcome.GetResult().GetSecretString() <<
 std::endl;
    }
    else {
        std::cerr << "Failed with Error: " << getSecretValueOutcome.GetError()
                  << std::endl;
    }

    return getSecretValueOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetSecretValuea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: recuperar o valor secreto criptografado de um segredo

O exemplo de `get-secret-value` a seguir obtém o valor atual do segredo.

```
aws secretsmanager get-secret-value \
    --secret-id MyTestSecret
```

Saída:

```
{
```

```
    "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
    "Name": "MyTestSecret",
    "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "SecretString": "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1523477145.713
}
```

Para obter mais informações, consulte Recuperar segredos no Guia do usuário do Secrets Manager.

Exemplo 2: como recuperar o valor secreto anterior

O seguinte exemplo de `get-secret-value` recupera o valor secreto anterior:

```
aws secretsmanager get-secret-value \
    --secret-id MyTestSecret
    --version-stage AWSPREVIOUS
```

Saída:

```
{
    "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
    "Name": "MyTestSecret",
    "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "SecretString": "{\"user\":\"diegor\",\"password\":\"PREVIOUS-EXAMPLE-
PASSWORD\"}",
    "VersionStages": [
        "AWSPREVIOUS"
    ],
    "CreatedDate": 1523477145.713
}
```

Para obter mais informações, consulte Recuperar segredos no Guia do usuário do Secrets Manager.

- Para obter detalhes da API, consulte GetSecretValueem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import
 software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-
secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <secretName>\s
```

```
                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
                .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }

    public static void getValue(SecretsManagerClient secretsClient, String
 secretName) {
        try {
            GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                    .secretId(secretName)
                    .build();

            GetSecretValueResponse valueResponse =
 secretsClient.getSecretValue(valueRequest);
            String secret = valueResponse.secretString();
            System.out.println(secret);

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetSecretValuea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetSecretValueCommand,
  SecretsManagerClient,
} from "@aws-sdk/client-secrets-manager";

export const getSecretValue = async (secretName = "SECRET_NAME") => {
  const client = new SecretsManagerClient();
  const response = await client.send(
    new GetSecretValueCommand({
      SecretId: secretName,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '584eb612-f8b0-48c9-855e-6d246461b604',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   ARN: 'arn:aws:secretsmanager:us-east-1:xxxxxxxxxxxx:secret:binary-
secret-3873048-xxxxxx',
  //   CreatedDate: 2023-08-08T19:29:51.294Z,
  //   Name: 'binary-secret-3873048',
  //   SecretBinary: Uint8Array(11) [
  //      98, 105, 110, 97, 114,
  //     121,  32, 100, 97, 116,
  //      97
  //   ],
  //   VersionId: '712083f4-0d26-415e-8044-16735142cd6a',
```

```
  //   VersionStages: [ 'AWSCURRENT' ]
  // }

  if (response.SecretString) {
    return response.SecretString;
  }

  if (response.SecretBinary) {
    return response.SecretBinary;
  }
};
```

- Para obter detalhes da API, consulte GetSecretValuea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use
 { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

- Para obter detalhes da API, consulte a GetSecretValuereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GetSecretWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def get_secret(self, secret_name):
        """
        Retrieve individual secrets from AWS Secrets Manager using the
 get_secret_value API.
        This function assumes the stack mentioned in the source code README has
 been successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param secret_name: The name of the secret fetched.
        :type secret_name: str
        """
        try:
            get_secret_value_response = self.client.get_secret_value(
                SecretId=secret_name
            )
            logging.info("Secret retrieved successfully.")
            return get_secret_value_response["SecretString"]
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"The requested secret {secret_name} was not found."
            logger.info(msg)
            return msg
        except Exception as e:
```

```
            logger.error(f"An unknown error occurred: {str(e)}.")
            raise
```

- Para obter detalhes da API, consulte a GetSecretValueReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_secret(client: &Client, name: &str) -> Result<(), Error> {
    let resp = client.get_secret_value().secret_id(name).send().await?;

    println!("Value: {}", resp.secret_string().unwrap_or("No value!"));

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetSecretValuereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_secret_name = 'MySecretName'
        DATA(lo_result) = lo_smr->getsecretvalue( iv_secretid = iv_secret_name ).
        ov_secret_value = lo_result->get_secretstring( ).
        MESSAGE 'Secret value retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_smrresourcenotfoundex.
        MESSAGE 'The requested secret was not found.' TYPE 'E'.
      CATCH /aws1/cx_smrdecryptionfailure.
        MESSAGE 'Failed to decrypt the secret.' TYPE 'E'.
      CATCH /aws1/cx_smrinvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'E'.
      CATCH /aws1/cx_smrinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetSecretValuereferência da API AWS SDK for SAP ABAP.

Cenários para o uso do Secrets Manager AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Secrets Manager com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Secrets Manager ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma API REST de biblioteca de empréstimos
- Criação de uma instância de banco de dados relacional
- Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI
- Introdução aos bancos de dados de documentos
- Como mover segredos codificados para o armazenamento seguro de segredos

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice para criar uma API REST apoiada por um banco de dados Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Use AWS Secrets Manager para gerenciar as credenciais do banco de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST sem servidor no Amazon API Gateway e. AWS Lambda
- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Criação de uma instância de banco de dados relacional

O exemplo de código a seguir mostra como:

- Configurar componentes de rede.

- Criar um grupo de sub-redes de banco de dados
- Criar uma instância de banco de dados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/
latest/UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier
 "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ];
 then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ];
 then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
```

```
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found
 ${#SUBNET_IDS[@]} subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"
```

```
echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP
 ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
```

```
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id
 "$SECRET_NAME" --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -
d'"' -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -
d'"' -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
```

```
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
```

```
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup

- DescribeDbInstances
- DescribeSubnets
- DescribeVpcs
- GetSecretValue
- Wait

Conceitos básicos de data warehouses com tecnologia sem servidor usando a CLI

O exemplo de código a seguir mostra como:

- Use o gerenciador de segredos CreateSecret
- Use o gerenciador de segredos DeleteSecret
- Use o gerenciador de segredos GetSecretValue
- Use redshift CreateNamespace
- Use redshift CreateWorkgroup
- Use redshift DeleteNamespace
- Use iam CreateRole

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq
 dependency)
# This script creates a Redshift Serverless environment, loads sample data, and
 runs queries
# Uses AWS Secrets Manager for secure password management without requiring jq
```

```
# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)
```

```
    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
```

```
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi
```

```
    echo "Attempt $i/$max_attempts: $resource_type $resource_name status:
 $status. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be
 available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "============================================"
  echo "CLEANUP CONFIRMATION"
  echo "============================================"
  echo "The following resources were created:"
```

```
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --
workgroup-name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --
namespace-name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
```

```
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin
 credentials for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
```

```
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF
```

```
# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless
 get-namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
```

```
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless
 get-workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
```

```
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
```

```
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
```

```
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 sales)"
```

```
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
```

```
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID"
 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time
 total sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price
 desc) as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10
```

```
# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID"
 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
```

```
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret

- DeleteWorkgroup
- GetNamespace
- GetRole
- GetSecretValue
- GetWorkgroup
- PutRolePolicy
- UpdateNamespace

Introdução aos bancos de dados de documentos

O exemplo de código a seguir mostra como:

- Criar um grupo de sub-redes de banco de dados
- Criar um cluster do DocumentDB.
- Criar uma instância do DocumentDB.
- Configurar a segurança e a conectividade.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

################################################################################
```

```
# Configuration
##############################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

##############################################################################
# Logging
##############################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""

##############################################################################
# Region pre-check
##############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

##############################################################################
# Error handler
##############################################################################
```

```
handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

################################################################################
# Wait function
################################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi
```

```
        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
```

```
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound
\|not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG
 ingress rule."
    fi

    # Delete instance (must be deleted before cluster)
```

```
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed
 to delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete
 cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}

##############################################################################
# Step 1: Generate password and store in Secrets Manager
##############################################################################
echo "==========================================="
echo "Step 1: Create master password in Secrets Manager"
```

```
echo "============================================"
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

################################################################################
# Step 2: Find default VPC and subnets
################################################################################
echo "============================================"
echo "Step 2: Find default VPC and subnets"
echo "============================================"
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
```

```
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
```

```
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

################################################################################
# Step 3: Create subnet group
################################################################################
echo "============================================"
echo "Step 3: Create DocumentDB subnet group"
echo "============================================"
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

################################################################################
# Step 4: Create DocumentDB cluster
################################################################################
echo "============================================"
echo "Step 4: Create DocumentDB cluster"
echo "============================================"
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
```

```
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "==========================================="
echo "Step 5: Create DocumentDB instance"
echo "==========================================="
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)
```

```
if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

##############################################################################
# Step 6: Get cluster endpoint and security group
##############################################################################
echo "============================================"
echo "Step 6: Get cluster endpoint and security group"
echo "============================================"
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
##############################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""
```

```
# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null
 || true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi
```

```
echo ""

##############################################################################
# Step 8: Download CA certificate
##############################################################################
echo "============================================="
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "============================================="
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

##############################################################################
# Step 9: Display connection information
##############################################################################
echo "============================================="
echo "CONNECTION INFORMATION"
echo "============================================="
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
```

```
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH}
 \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString
 --output text)"
echo ""

##############################################################################
# Step 10: Cleanup
##############################################################################
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster

- CreateDbInstance
- CreateDbSubnetGroup
- CreateDefaultVpc
- CreateSecret
- DeleteDbCluster
- DeleteDbInstance
- DeleteDbSubnetGroup
- DeleteSecret
- DescribeDbClusters
- DescribeDbInstances
- DescribeSubnets
- DescribeVpcs
- GetSecretValue
- RevokeSecurityGroupIngress
- Wait

Como mover segredos codificados para o armazenamento seguro de segredos

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Criar um segredo no Secrets Manager.
- Atualizar o código de uma aplicação.
- Atualizar o segredo.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
```

```
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-
delete-without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name
 "$RUNTIME_ROLE_NAME" --query 'PolicyNames[]' --output text 2>/dev/null || true);
 do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
```

```
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs
```

```
# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_u
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
```

```
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/
null || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
```

```
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').tok

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi
```

```
echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy
  - GetCallerIdentity

- GetSecretValue
- PutResourcePolicy
- UpdateSecret

Exemplos de código para o Amazon SES usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Simple Email Service com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon SES: mais informações sobre o Amazon SES.
- Referência da API Amazon SES: detalhes sobre todas as ações disponíveis do Amazon SES.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon SES AWS SDKs
  - Ações para o Amazon SES usando AWS SDKs
    - Use CreateReceiptFiltercom um AWS SDK
    - Use CreateReceiptRulecom um AWS SDK
    - Use CreateReceiptRuleSetcom um AWS SDK
    - Use CreateTemplatecom um AWS SDK
    - Use DeleteIdentitycom um AWS SDK ou CLI
    - Use DeleteReceiptFiltercom um AWS SDK

- Use DeleteReceiptRulecom um AWS SDK
- Use DeleteReceiptRuleSetcom um AWS SDK
- Use DeleteTemplatecom um AWS SDK
- Use DescribeReceiptRuleSetcom um AWS SDK
- Use GetIdentityVerificationAttributescom um AWS SDK ou CLI
- Use GetSendQuotacom um AWS SDK ou CLI
- Usar GetSendStatistics com uma CLI
- Use GetTemplatecom um AWS SDK
- Use ListIdentitiescom um AWS SDK ou CLI
- Use ListReceiptFilterscom um AWS SDK
- Use ListTemplatescom um AWS SDK
- Use SendBulkTemplatedEmailcom um AWS SDK
- Use SendEmailcom um AWS SDK ou CLI
- Use SendRawEmailcom um AWS SDK ou CLI
- Use SendTemplatedEmailcom um AWS SDK
- Use UpdateTemplatecom um AWS SDK
- Use VerifyDomainIdentitycom um AWS SDK ou CLI
- Use VerifyEmailIdentitycom um AWS SDK ou CLI

- Cenários para o Amazon SES usando AWS SDKs
  - Criar uma aplicação de transmissão do Amazon Transcribe
  - Copie as identidades de e-mail e domínio do Amazon SES de uma AWS Região para outra usando um AWS SDK
  - Criar uma aplicação Web para monitorar dados do DynamoDB
  - Criar um rastreador de itens do Amazon Redshift
  - Crie um rastreador de itens de trabalho do Aurora Sem Servidor
  - Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK
  - Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK
  - Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
  - Gerar credenciais para estabelecer conexão com um endpoint SMTP do Amazon SES

- Configuração de serviços de envio de e-mails

- Usar Step Functions para invocar funções do Lambda
- Verifique uma identidade de e-mail e envie mensagens com o Amazon SES usando um AWS SDK

Exemplos básicos para o uso do Amazon SES AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Simple Email Service com SDKs da AWS .

Exemplos

- Ações para o Amazon SES usando AWS SDKs
  - Use CreateReceiptFiltercom um AWS SDK
  - Use CreateReceiptRulecom um AWS SDK
  - Use CreateReceiptRuleSetcom um AWS SDK
  - Use CreateTemplatecom um AWS SDK
  - Use DeleteIdentitycom um AWS SDK ou CLI
  - Use DeleteReceiptFiltercom um AWS SDK
  - Use DeleteReceiptRulecom um AWS SDK
  - Use DeleteReceiptRuleSetcom um AWS SDK
  - Use DeleteTemplatecom um AWS SDK
  - Use DescribeReceiptRuleSetcom um AWS SDK
  - Use GetIdentityVerificationAttributescom um AWS SDK ou CLI
  - Use GetSendQuotacom um AWS SDK ou CLI
  - Usar GetSendStatistics com uma CLI
  - Use GetTemplatecom um AWS SDK
  - Use ListIdentitiescom um AWS SDK ou CLI
  - Use ListReceiptFilterscom um AWS SDK
  - Use ListTemplatescom um AWS SDK
  - Use SendBulkTemplatedEmailcom um AWS SDK
  - Use SendEmailcom um AWS SDK ou CLI
  - Use SendRawEmailcom um AWS SDK ou CLI
  - Use SendTemplatedEmailcom um AWS SDK

- Use UpdateTemplatecom um AWS SDK
- Use VerifyDomainIdentitycom um AWS SDK ou CLI
- Use VerifyEmailIdentitycom um AWS SDK ou CLI

Ações para o Amazon SES usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon SES com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon SES e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon SES usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Simple Email Service.

Exemplos

- Use CreateReceiptFiltercom um AWS SDK
- Use CreateReceiptRulecom um AWS SDK
- Use CreateReceiptRuleSetcom um AWS SDK
- Use CreateTemplatecom um AWS SDK
- Use DeleteIdentitycom um AWS SDK ou CLI
- Use DeleteReceiptFiltercom um AWS SDK
- Use DeleteReceiptRulecom um AWS SDK
- Use DeleteReceiptRuleSetcom um AWS SDK
- Use DeleteTemplatecom um AWS SDK
- Use DescribeReceiptRuleSetcom um AWS SDK
- Use GetIdentityVerificationAttributescom um AWS SDK ou CLI
- Use GetSendQuotacom um AWS SDK ou CLI
- Usar GetSendStatistics com uma CLI
- Use GetTemplatecom um AWS SDK
- Use ListIdentitiescom um AWS SDK ou CLI
- Use ListReceiptFilterscom um AWS SDK

- Use ListTemplatescom um AWS SDK
- Use SendBulkTemplatedEmailcom um AWS SDK
- Use SendEmailcom um AWS SDK ou CLI
- Use SendRawEmailcom um AWS SDK ou CLI
- Use SendTemplatedEmailcom um AWS SDK
- Use UpdateTemplatecom um AWS SDK
- Use VerifyDomainIdentitycom um AWS SDK ou CLI
- Use VerifyEmailIdentitycom um AWS SDK ou CLI

Use **`CreateReceiptFilter`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateReceiptFilter`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt filter..
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param cidr: IP address or IP address range in Classless Inter-Domain Routing
 (CIDR) notation.
  \param policy: Block or allow enum of type ReceiptFilterPolicy.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::String &cidr,
                                      Aws::SES::Model::ReceiptFilterPolicy
 policy,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
```

```
    Aws::SES::Model::CreateReceiptFilterRequest createReceiptFilterRequest;
    Aws::SES::Model::ReceiptFilter receiptFilter;
    Aws::SES::Model::ReceiptIpFilter receiptIpFilter;
    receiptIpFilter.SetCidr(cidr);
    receiptIpFilter.SetPolicy(policy);
    receiptFilter.SetName(receiptFilterName);
    receiptFilter.SetIpFilter(receiptIpFilter);
    createReceiptFilterRequest.SetFilter(receiptFilter);
    Aws::SES::Model::CreateReceiptFilterOutcome createReceiptFilterOutcome =
 sesClient.CreateReceiptFilter(
            createReceiptFilterRequest);
    if (createReceiptFilterOutcome.IsSuccess()) {
        std::cout << "Successfully created receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt filter: " <<
                  createReceiptFilterOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return createReceiptFilterOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateReceiptFiltera Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  CreateReceiptFilterCommand,
  ReceiptFilterPolicy,
} from "@aws-sdk/client-ses";
```

```
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const createCreateReceiptFilterCommand = ({ policy, ipOrRange, name }) => {
  return new CreateReceiptFilterCommand({
    Filter: {
      IpFilter: {
        Cidr: ipOrRange, // string, either a single IP address (10.0.0.1) or an
 IP address range in CIDR notation (10.0.0.1/24)).
        Policy: policy, // enum ReceiptFilterPolicy, email traffic from the
 filtered addressesOptions.
      },
      /*
        The name of the IP address filter. Only ASCII letters, numbers,
 underscores, or dashes.
        Must be less than 64 characters and start and end with a letter or
 number.
       */
      Name: name,
    },
  });
};

const FILTER_NAME = getUniqueName("ReceiptFilter");

const run = async () => {
  const createReceiptFilterCommand = createCreateReceiptFilterCommand({
    policy: ReceiptFilterPolicy.Allow,
    ipOrRange: "10.0.0.1",
    name: FILTER_NAME,
  });

  try {
    return await sesClient.send(createReceiptFilterCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte CreateReceiptFiltera Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_filter(self, filter_name, ip_address_or_range, allow):
        """
        Creates a filter that allows or blocks incoming mail from an IP address
 or
        range.

        :param filter_name: The name to give the filter.
        :param ip_address_or_range: The IP address or range to block or allow.
        :param allow: When True, incoming mail is allowed from the specified IP
                      address or range; otherwise, it is blocked.
        """
        try:
            policy = "Allow" if allow else "Block"
            self.ses_client.create_receipt_filter(
```

```
                Filter={
                    "Name": filter_name,
                    "IpFilter": {"Cidr": ip_address_or_range, "Policy": policy},
                }
            )
            logger.info(
                "Created receipt filter %s to %s IP of %s.",
                filter_name,
                policy,
                ip_address_or_range,
            )
        except ClientError:
            logger.exception("Couldn't create receipt filter %s.", filter_name)
            raise
```

- Para obter detalhes da API, consulte a CreateReceiptFilterReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" iv_allow = abap_true means 'Allow', abap_false means 'Block'
DATA(lv_policy) = COND /aws1/sesreceiptfilterpolicy(
  WHEN iv_allow = abap_true THEN 'Allow'
  ELSE 'Block'
).

DATA(lo_ip_filter) = NEW /aws1/cl_sesreceiptipfilter(
  iv_policy = lv_policy
  iv_cidr = iv_ip_address_or_range
).
```

```
    DATA(lo_filter) = NEW /aws1/cl_sesreceiptfilter(
      iv_name = iv_filter_name
      io_ipfilter = lo_ip_filter
    ).

    TRY.
        lo_ses->createreceiptfilter( io_filter = lo_filter ).
        MESSAGE 'Receipt filter created successfully' TYPE 'I'.
      CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
        DATA(lv_error) = |Filter already exists: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateReceiptFilterreferência da API AWS SDK for SAP ABAP.

Use **`CreateReceiptRule`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateReceiptRule`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
  \param receiptRuleName: The name for the receipt rule.
  \param s3BucketName: The name of the S3 bucket for incoming mail.
  \param s3ObjectKeyPrefix: The prefix for the objects in the S3 bucket.
```

```
  \param ruleSetName: The name of the rule set where the receipt rule is added.
  \param recipients: Aws::Vector of recipients.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &s3BucketName,
                                    const Aws::String &s3ObjectKeyPrefix,
                                    const Aws::String &ruleSetName,
                                    const Aws::Vector<Aws::String> &recipients,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleRequest createReceiptRuleRequest;

    Aws::SES::Model::S3Action s3Action;
    s3Action.SetBucketName(s3BucketName);
    s3Action.SetObjectKeyPrefix(s3ObjectKeyPrefix);

    Aws::SES::Model::ReceiptAction receiptAction;
    receiptAction.SetS3Action(s3Action);

    Aws::SES::Model::ReceiptRule receiptRule;
    receiptRule.SetName(receiptRuleName);
    receiptRule.WithRecipients(recipients);

    Aws::Vector<Aws::SES::Model::ReceiptAction> receiptActionList;
    receiptActionList.emplace_back(receiptAction);
    receiptRule.SetActions(receiptActionList);

    createReceiptRuleRequest.SetRuleSetName(ruleSetName);
    createReceiptRuleRequest.SetRule(receiptRule);

    auto outcome = sesClient.CreateReceiptRule(createReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateReceiptRulea Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateReceiptRuleCommand, TlsPolicy } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");
const RULE_NAME = getUniqueName("RuleName");
const S3_BUCKET_NAME = getUniqueName("S3BucketName");

const createS3ReceiptRuleCommand = ({
  bucketName,
  emailAddresses,
  name,
  ruleSet,
}) => {
  return new CreateReceiptRuleCommand({
    Rule: {
      Actions: [
        {
          S3Action: {
            BucketName: bucketName,
            ObjectKeyPrefix: "email",
          },
```

```
        },
      ],
      Recipients: emailAddresses,
      Enabled: true,
      Name: name,
      ScanEnabled: false,
      TlsPolicy: TlsPolicy.Optional,
    },
    RuleSetName: ruleSet, // Required
  });
};

const run = async () => {
  const s3ReceiptRuleCommand = createS3ReceiptRuleCommand({
    bucketName: S3_BUCKET_NAME,
    emailAddresses: ["email@example.com"],
    name: RULE_NAME,
    ruleSet: RULE_SET_NAME,
  });

  try {
    return await sesClient.send(s3ReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to create S3 receipt rule.", err);
    throw err;
  }
};
```

- Para obter detalhes da API, consulte CreateReceiptRulea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket do Amazon S3 no qual o Amazon SES possa colocar cópias de e-mails recebidos e crie uma regra que copia para o bucket os e-mails recebidos de uma lista específica de destinatários.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_bucket_for_copy(self, bucket_name):
        """
        Creates a bucket that can receive copies of emails from Amazon SES. This
        includes adding a policy to the bucket that grants Amazon SES permission
        to put objects in the bucket.

        :param bucket_name: The name of the bucket to create.
        :return: The newly created bucket.
        """
        allow_ses_put_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowSESPut",
                    "Effect": "Allow",
                    "Principal": {"Service": "ses.amazonaws.com"},
                    "Action": "s3:PutObject",
                    "Resource": f"arn:aws:s3:::{bucket_name}/*",
                }
            ],
        }
        bucket = None
        try:
            bucket = self.s3_resource.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
```

```
                    "LocationConstraint":
self.s3_resource.meta.client.meta.region_name
                },
            )
            bucket.wait_until_exists()
            bucket.Policy().put(Policy=json.dumps(allow_ses_put_policy))
            logger.info("Created bucket %s to receive copies of emails.",
bucket_name)
        except ClientError:
            logger.exception("Couldn't create bucket to receive copies of
emails.")
            if bucket is not None:
                bucket.delete()
            raise
        else:
            return bucket


    def create_s3_copy_rule(
        self, rule_set_name, rule_name, recipients, bucket_name, prefix
    ):
        """
        Creates a rule so that all emails received by the specified recipients
are
        copied to an Amazon S3 bucket.

        :param rule_set_name: The name of a previously created rule set to
contain
                              this rule.
        :param rule_name: The name to give the rule.
        :param recipients: When an email is received by one of these recipients,
it
                           is copied to the Amazon S3 bucket.
        :param bucket_name: The name of the bucket to receive email copies. This
                            bucket must allow Amazon SES to put objects into it.
        :param prefix: An object key prefix to give the emails copied to the
bucket.
        """
        try:
            self.ses_client.create_receipt_rule(
                RuleSetName=rule_set_name,
                Rule={
                    "Name": rule_name,
                    "Enabled": True,
```

```
                    "Recipients": recipients,
                    "Actions": [
                        {
                            "S3Action": {
                                "BucketName": bucket_name,
                                "ObjectKeyPrefix": prefix,
                            }
                        }
                    ],
                },
            )
            logger.info(
                "Created rule %s to copy mail received by %s to bucket %s.",
                rule_name,
                recipients,
                bucket_name,
            )
        except ClientError:
            logger.exception("Couldn't create rule %s.", rule_name)
            raise
```

- Para obter detalhes da API, consulte a CreateReceiptRuleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Create S3 action for copying emails to S3
DATA(lo_s3_action) = NEW /aws1/cl_sess3action(
  iv_bucketname = iv_bucket_name
  iv_objectkeyprefix = iv_prefix
).
```

```
" Create receipt action with S3 action
DATA(lo_action) = NEW /aws1/cl_sesreceiptaction(
  io_s3action = lo_s3_action
).

" Create list of actions
DATA lt_actions TYPE /aws1/cl_sesreceiptaction=>tt_receiptactionslist.
APPEND lo_action TO lt_actions.

" Create receipt rule
DATA(lo_rule) = NEW /aws1/cl_sesreceiptrule(
  iv_name = iv_rule_name
  iv_enabled = abap_true
  it_recipients = it_recipients
  it_actions = lt_actions
).

TRY.
    lo_ses->createreceiptrule(
      iv_rulesetname = iv_rule_set_name
      io_rule = lo_rule
    ).
    MESSAGE 'Receipt rule created successfully' TYPE 'I'.
  CATCH /aws1/cx_sesinvalids3confex INTO DATA(lo_ex1).
    DATA(lv_error) = |Invalid S3 configuration: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateReceiptRulereferência da API AWS SDK for SAP ABAP.

Use **`CreateReceiptRuleSet`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateReceiptRuleSet`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param ruleSetName: The name of the rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRuleSet(const Aws::String &ruleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleSetRequest createReceiptRuleSetRequest;

    createReceiptRuleSetRequest.SetRuleSetName(ruleSetName);

    Aws::SES::Model::CreateReceiptRuleSetOutcome outcome =
 sesClient.CreateReceiptRuleSet(
            createReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule set." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateReceiptRuleSeta Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createCreateReceiptRuleSetCommand = (ruleSetName) => {
  return new CreateReceiptRuleSetCommand({ RuleSetName: ruleSetName });
};

const run = async () => {
  const createReceiptRuleSetCommand =
    createCreateReceiptRuleSetCommand(RULE_SET_NAME);

  try {
    return await sesClient.send(createReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to create receipt rule set", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte CreateReceiptRuleSeta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_rule_set(self, rule_set_name):
        """
        Creates an empty rule set. Rule sets contain individual rules and can be
        used to organize rules.

        :param rule_set_name: The name to give the rule set.
        """
        try:
            self.ses_client.create_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Created receipt rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't create receipt rule set %s.",
 rule_set_name)
            raise
```

- Para obter detalhes da API, consulte a CreateReceiptRuleSetReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ses->createreceiptruleset( iv_rulesetname = iv_rule_set_name ).
    MESSAGE 'Receipt rule set created successfully' TYPE 'I'.
  CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
    DATA(lv_error) = |Rule set already exists: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateReceiptRuleSetreferência da API AWS SDK for SAP ABAP.

Use **`CreateTemplate`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an email template.
    /// </summary>
    /// <param name="name">Name of the template.</param>
    /// <param name="subject">Email subject.</param>
    /// <param name="text">Email body text.</param>
    /// <param name="html">Email HTML body text.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string name, string subject,
string text,
        string html)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.CreateTemplateAsync(
                new CreateTemplateRequest
                {
                    Template = new Template
                    {
                        TemplateName = name,
                        SubjectPart = subject,
                        TextPart = text,
                        HtmlPart = html
                    }
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine("CreateEmailTemplateAsync failed with exception: "
 + ex.Message);
        }

        return success;
    }
```

- Para obter detalhes da API, consulte CreateTemplatea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateTemplateRequest createTemplateRequest;
    Aws::SES::Model::Template aTemplate;
```

```
    aTemplate.SetTemplateName(templateName);
    aTemplate.SetHtmlPart(htmlPart);
    aTemplate.SetSubjectPart(subjectPart);
    aTemplate.SetTextPart(textPart);

    createTemplateRequest.SetTemplate(aTemplate);

    Aws::SES::Model::CreateTemplateOutcome outcome = sesClient.CreateTemplate(
            createTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created template." << templateName << "."
                  << std::endl;
    }
    else {
        std::cerr << "Error creating template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTemplatea Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateTemplateCommand } from "@aws-sdk/client-ses";
```

```
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const TEMPLATE_NAME = getUniqueName("TestTemplateName");

const createCreateTemplateCommand = () => {
  return new CreateTemplateCommand({
    /**
     * The template feature in Amazon SES is based on the Handlebars template
 system.
     */
    Template: {
      /**
       * The name of an existing template in Amazon SES.
       */
      TemplateName: TEMPLATE_NAME,
      HtmlPart: `
        <h1>Hello, {{contact.firstName}}!</h1>
        <p>
        Did you know Amazon has a mascot named Peccy?
        </p>
      `,
      SubjectPart: "Amazon Tip",
    },
  });
};

const run = async () => {
  const createTemplateCommand = createCreateTemplateCommand();

  try {
    return await sesClient.send(createTemplateCommand);
  } catch (err) {
    console.log("Failed to create template.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte CreateTemplatea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
```

```
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise
```

- Para obter detalhes da API, consulte a CreateTemplateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA(lo_template) = NEW /aws1/cl_sestemplate(
      iv_templatename = iv_name
      iv_subjectpart = iv_subject
      iv_textpart = iv_text
      iv_htmlpart = iv_html
    ).

    TRY.
        lo_ses->createtemplate( io_template = lo_template ).
        MESSAGE 'Template created successfully' TYPE 'I'.
```

```
      CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
        DATA(lv_error) = |Template already exists: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_sesinvalidtemplateex INTO DATA(lo_ex2).
        lv_error = |Invalid template: { lo_ex2->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex2.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTemplatereferência da API AWS SDK for SAP ABAP.

Use **`DeleteIdentity`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteIdentity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configuração de serviços de envio de e-mails
- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Delete an email identity.
    /// </summary>
    /// <param name="identityEmail">The identity email to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteIdentityAsync(string identityEmail)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteIdentityAsync(
                new DeleteIdentityRequest
                {
                    Identity = identityEmail
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteIdentityAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

- Para obter detalhes da API, consulte DeleteIdentitya Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete the specified identity (an email address or a domain).
```

```
/*!
  \param identity: The identity to delete.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteIdentity(const Aws::String &identity,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteIdentityRequest deleteIdentityRequest;

    deleteIdentityRequest.SetIdentity(identity);

    Aws::SES::Model::DeleteIdentityOutcome outcome = sesClient.DeleteIdentity(
            deleteIdentityRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted identity." << std::endl;
    }
    else {
        std::cerr << "Error deleting identity. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteIdentitya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir uma identidade

O exemplo a seguir usa o comando `delete-identity` para excluir uma identidade da lista de identidades verificadas com o Amazon SES:

```
aws ses delete-identity --identity user@example.com
```

Para saber mais sobre identidades verificadas, consulte Verificar endereços de e-mail e domínios no Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte DeleteIdentityem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const IDENTITY_EMAIL = "fake@example.com";

const createDeleteIdentityCommand = (identityName) => {
  return new DeleteIdentityCommand({
    Identity: identityName,
  });
};

const run = async () => {
  const deleteIdentityCommand = createDeleteIdentityCommand(IDENTITY_EMAIL);

  try {
    return await sesClient.send(deleteIdentityCommand);
  } catch (err) {
    console.log("Failed to delete identity.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteIdentitya Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise
```

- Para obter detalhes da API, consulte a DeleteIdentityReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ses->deleteidentity( iv_identity = iv_identity ).
        MESSAGE 'Identity deleted successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteIdentityreferência da API AWS SDK for SAP ABAP.

Use **`DeleteReceiptFilter`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteReceiptFilter`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt filter.
/*!
```

```
  \param receiptFilterName: The name for the receipt filter.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptFilterRequest deleteReceiptFilterRequest;

    deleteReceiptFilterRequest.SetFilterName(receiptFilterName);

    Aws::SES::Model::DeleteReceiptFilterOutcome outcome =
 sesClient.DeleteReceiptFilter(
            deleteReceiptFilterRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error deleting receipt filter. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteReceiptFiltera Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteReceiptFilterCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RECEIPT_FILTER_NAME = getUniqueName("ReceiptFilterName");

const createDeleteReceiptFilterCommand = (filterName) => {
  return new DeleteReceiptFilterCommand({ FilterName: filterName });
};

const run = async () => {
  const deleteReceiptFilterCommand =
    createDeleteReceiptFilterCommand(RECEIPT_FILTER_NAME);

  try {
    return await sesClient.send(deleteReceiptFilterCommand);
  } catch (err) {
    console.log("Error deleting receipt filter.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteReceiptFiltera Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
```

```
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_filter(self, filter_name):
        """
        Deletes a receipt filter.

        :param filter_name: The name of the filter to delete.
        """
        try:
            self.ses_client.delete_receipt_filter(FilterName=filter_name)
            logger.info("Deleted receipt filter %s.", filter_name)
        except ClientError:
            logger.exception("Couldn't delete receipt filter %s.", filter_name)
            raise
```

- Para obter detalhes da API, consulte a DeleteReceiptFilterReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ses->deletereceiptfilter( iv_filtername = iv_filter_name ).
        MESSAGE 'Receipt filter deleted successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteReceiptFilterreferência da API AWS SDK for SAP ABAP.

Use **`DeleteReceiptRule`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteReceiptRule`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
  \param receiptRuleName: The name for the receipt rule.
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &receiptRuleSetName,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleRequest deleteReceiptRuleRequest;

    deleteReceiptRuleRequest.SetRuleName(receiptRuleName);
    deleteReceiptRuleRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleOutcome outcome =
 sesClient.DeleteReceiptRule(
```

```
            deleteReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule." << std::endl;
    }
    else {
        std::cout << "Error deleting receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteReceiptRulea Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteReceiptRuleCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_NAME = getUniqueName("RuleName");
const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleCommand = () => {
  return new DeleteReceiptRuleCommand({
    RuleName: RULE_NAME,
    RuleSetName: RULE_SET_NAME,
  });
```

```
};

const run = async () => {
  const deleteReceiptRuleCommand = createDeleteReceiptRuleCommand();
  try {
    return await sesClient.send(deleteReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteReceiptRulea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule(self, rule_set_name, rule_name):
        """
        Deletes a rule.
```

```
        :param rule_set_name: The rule set that contains the rule to delete.
        :param rule_name: The rule to delete.
        """
        try:
            self.ses_client.delete_receipt_rule(
                RuleSetName=rule_set_name, RuleName=rule_name
            )
            logger.info("Removed rule %s from rule set %s.", rule_name,
rule_set_name)
        except ClientError:
            logger.exception(
                "Couldn't remove rule %s from rule set %s.", rule_name,
rule_set_name
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteReceiptRuleReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ses->deletereceiptrule(
          iv_rulesetname = iv_rule_set_name
          iv_rulename = iv_rule_name
        ).
        MESSAGE 'Receipt rule deleted successfully' TYPE 'I'.
      CATCH /aws1/cx_sesrulesetdoesnotexex INTO DATA(lo_ex1).
        DATA(lv_error) = |Rule set does not exist: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteReceiptRulereferência da API AWS SDK for SAP ABAP.

Use **`DeleteReceiptRuleSet`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteReceiptRuleSet`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRuleSet(const Aws::String &receiptRuleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleSetRequest deleteReceiptRuleSetRequest;

    deleteReceiptRuleSetRequest.SetRuleSetName(receiptRuleSetName);
```

```
    Aws::SES::Model::DeleteReceiptRuleSetOutcome outcome =
 sesClient.DeleteReceiptRuleSet(
            deleteReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule set." << std::endl;
    }

    else {
        std::cerr << "Error deleting receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteReceiptRuleSeta Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleSetCommand = () => {
  return new DeleteReceiptRuleSetCommand({ RuleSetName: RULE_SET_NAME });
};
```

```
const run = async () => {
  const deleteReceiptRuleSetCommand = createDeleteReceiptRuleSetCommand();

  try {
    return await sesClient.send(deleteReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule set.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteReceiptRuleSeta Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule_set(self, rule_set_name):
        """
        Deletes a rule set. When a rule set is deleted, all of the rules it
 contains
```

```
        are also deleted.

        :param rule_set_name: The name of the rule set to delete.
        """
        try:
            self.ses_client.delete_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Deleted rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't delete rule set %s.", rule_set_name)
            raise
```

- Para obter detalhes da API, consulte a DeleteReceiptRuleSetReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ses->deletereceiptruleset( iv_rulesetname = iv_rule_set_name ).
    MESSAGE 'Receipt rule set deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_sescannotdeleteex INTO DATA(lo_ex1).
    DATA(lv_error) = |Cannot delete rule set: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteReceiptRuleSetreferência da API AWS SDK for SAP ABAP.

Use **`DeleteTemplate`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete an email template.
/// </summary>
/// <param name="templateName">Name of the template.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteEmailTemplateAsync(string templateName)
{
    var success = false;
    try
    {
        var response = await _amazonSimpleEmailService.DeleteTemplateAsync(
            new DeleteTemplateRequest
            {
                TemplateName = templateName
            });
        success = response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (Exception ex)
```

```
        {
            Console.WriteLine("DeleteEmailTemplateAsync failed with exception: "
 + ex.Message);
        }

        return success;
    }
```

- Para obter detalhes da API, consulte DeleteTemplatea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteTemplate(const Aws::String &templateName,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteTemplateRequest deleteTemplateRequest;

    deleteTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::DeleteTemplateOutcome outcome = sesClient.DeleteTemplate(
            deleteTemplateRequest);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted template." << std::endl;
    }
    else {
        std::cerr << "Error deleting template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTemplatea Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createDeleteTemplateCommand = (templateName) =>
  new DeleteTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const deleteTemplateCommand = createDeleteTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(deleteTemplateCommand);
```

```
  } catch (err) {
    console.log("Failed to delete template.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteTemplatea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
```

```
        logger.info("Extracted template tags: %s", self.template_tags)


    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTemplateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ses->deletetemplate( iv_templatename = iv_template_name ).
        MESSAGE 'Template deleted successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTemplatereferência da API AWS SDK for SAP ABAP.

Use **`DescribeReceiptRuleSet`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeReceiptRuleSet`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def describe_receipt_rule_set(self, rule_set_name):
        """
        Gets data about a rule set.

        :param rule_set_name: The name of the rule set to retrieve.
        :return: Data about the rule set.
        """
        try:
            response = self.ses_client.describe_receipt_rule_set(
```

```
                RuleSetName=rule_set_name
            )
            logger.info("Got data for rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't get data for rule set %s.", rule_set_name)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DescribeReceiptRuleSetReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ses->describereceiptruleset(
      iv_rulesetname = iv_rule_set_name
    ).
    MESSAGE 'Receipt rule set described successfully' TYPE 'I'.
  CATCH /aws1/cx_sesrulesetdoesnotexex INTO DATA(lo_ex1).
    DATA(lv_error) = |Rule set does not exist: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeReceiptRuleSetreferência da API AWS SDK for SAP ABAP.

Use **`GetIdentityVerificationAttributes`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetIdentityVerificationAttributes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configuração de serviços de envio de e-mails
- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get identity verification status for an email.
    /// </summary>
    /// <returns>The verification status of the email.</returns>
    public async Task<VerificationStatus> GetIdentityStatusAsync(string email)
    {
        var result = VerificationStatus.TemporaryFailure;
        try
        {
            var response =
                await
_amazonSimpleEmailService.GetIdentityVerificationAttributesAsync(
                    new GetIdentityVerificationAttributesRequest
                    {
                        Identities = new List<string> { email }
                    });
```

```
            if (response.VerificationAttributes.ContainsKey(email))
                result =
response.VerificationAttributes[email].VerificationStatus;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetIdentityStatusAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- Para obter detalhes da API, consulte GetIdentityVerificationAttributesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para obter o status de verificação do Amazon SES para uma lista de identidades

O exemplo a seguir usa o comando `get-identity-verification-attributes` para recuperar o status de verificação do Amazon SES para uma lista de identidades:

```
aws ses get-identity-verification-attributes --
identities "user1@example.com" "user2@example.com"
```

Saída:

```
{
   "VerificationAttributes": {
       "user1@example.com": {
           "VerificationStatus": "Success"
       },
       "user2@example.com": {
           "VerificationStatus": "Pending"
       }
   }
```

```
}
```

Se você chamar esse comando com uma identidade que nunca foi enviada para verificação, essa identidade não aparecerá na saída.

Para saber mais sobre identidades verificadas, consulte Verificar endereços de e-mail e domínios no Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte GetIdentityVerificationAttributesem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
```

```
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status
```

- Para obter detalhes da API, consulte a GetIdentityVerificationAttributesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                              identity_type: 'EmailAddress'
                            })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
```

```
                                                          identities: [email]
                                                        })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Para obter detalhes da API, consulte GetIdentityVerificationAttributesa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   DATA lt_identities TYPE /aws1/cl_sesidentitylist_w=>tt_identitylist.
   APPEND NEW /aws1/cl_sesidentitylist_w( iv_value = iv_identity ) TO
lt_identities.

   TRY.
       DATA(lo_result) = lo_ses->getidentityverificationattrs(
         it_identities = lt_identities
       ).

       DATA(lt_attrs) = lo_result->get_verificationattributes( ).
       IF lt_attrs IS NOT INITIAL.
         LOOP AT lt_attrs ASSIGNING FIELD-SYMBOL(<ls_attr>).
           ov_status = <ls_attr>-value->get_verificationstatus( ).
           EXIT.
         ENDLOOP.
       ELSE.
         ov_status = 'NotFound'.
       ENDIF.
```

```
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
      DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetIdentityVerificationAttributesreferência da API AWS SDK for SAP ABAP.

Use **`GetSendQuota`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetSendQuota`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Configuração de serviços de envio de e-mails

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get information on the current account's send quota.
/// </summary>
/// <returns>The send quota response data.</returns>
public async Task<GetSendQuotaResponse> GetSendQuotaAsync()
{
    var result = new GetSendQuotaResponse();
    try
    {
        var response = await _amazonSimpleEmailService.GetSendQuotaAsync(
```

```
                new GetSendQuotaRequest());
            result = response;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetSendQuotaAsync failed with exception: " +
 ex.Message);
        }

        return result;
    }
```

- Para obter detalhes da API, consulte GetSendQuotaa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para obter limites do envio do Amazon SES

O exemplo a seguir usa o comando `get-send-quota` para retornar seus limites de envio do Amazon SES:

```
aws ses get-send-quota
```

Saída:

```
{
   "Max24HourSend": 200.0,
   "SentLast24Hours": 1.0,
   "MaxSendRate": 1.0
}
```

Max24HourSend é sua cota de envio, que é o número máximo de e-mails que você pode enviar em um período de 24 horas. A cota de envio reflete um período de tempo acumulado. Sempre que você tenta enviar um e-mail, o Amazon SES verifica quantos e-mails foram enviados nas 24 horas anteriores. Desde que o número total de e-mails que você enviou seja menor que a sua cota, sua solicitação de envio será aceito e seus e-mails serão enviados.

SentLast24Hours é o número de e-mails que você enviou nas últimas 24 horas.

MaxSendRate é o número máximo de e-mails que você pode enviar por segundo.

Os limites de envio se baseiam em destinatários, e não em mensagens. Por exemplo, um e-mail com dez destinatários conta como dez em sua cota de envio.

Para obter mais informações, consulte Gerenciamento de limites do envio do Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte GetSendQuotaem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse comando retorna os limites de envio atuais do usuário.

```
Get-SESSendQuota
```

- Para obter detalhes da API, consulte GetSendQuotaem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse comando retorna os limites de envio atuais do usuário.

```
Get-SESSendQuota
```

- Para obter detalhes da API, consulte GetSendQuotaem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetSendStatistics`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetSendStatistics`.

CLI

AWS CLI

Para obter estatísticas de envio do Amazon SES

O exemplo a seguir usa o comando `get-send-statistics` para retornar suas estatísticas de envio do Amazon SES

```
aws ses get-send-statistics
```

Saída:

```
{
   "SendDataPoints": [
       {
           "Complaints": 0,
           "Timestamp": "2013-06-12T19:32:00Z",
           "DeliveryAttempts": 2,
           "Bounces": 0,
           "Rejects": 0
       },
       {
           "Complaints": 0,
           "Timestamp": "2013-06-12T00:47:00Z",
           "DeliveryAttempts": 1,
           "Bounces": 0,
           "Rejects": 0
       }
   ]
}
```

O resultado é uma lista de pontos de dados que representa as duas últimas semanas de atividades de envio. Cada ponto de dados na lista contém estatísticas para um intervalo de 15 minutos.

Neste exemplo, há apenas dois pontos de dados porque os únicos e-mails que o usuário enviou nas últimas duas semanas caíram em dois intervalos de 15 minutos.

Para obter mais informações, consulte Monitoramento das estatísticas de uso do Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte GetSendStatisticsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse comando retorna as estatísticas de envio do usuário. O resultado é uma lista de pontos de dados que representa as duas últimas semanas de atividades de envio. Cada ponto de dados na lista contém estatísticas para um intervalo de 15 minutos.

```
Get-SESSendStatistic
```

- Para obter detalhes da API, consulte GetSendStatisticsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse comando retorna as estatísticas de envio do usuário. O resultado é uma lista de pontos de dados que representa as duas últimas semanas de atividades de envio. Cada ponto de dados na lista contém estatísticas para um intervalo de 15 minutos.

```
Get-SESSendStatistic
```

- Para obter detalhes da API, consulte GetSendStatisticsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetTemplate`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Verificar uma identidade de e-mail e enviar mensagens

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get a template's attributes.
/*!
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::getTemplate(const Aws::String &templateName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::GetTemplateRequest getTemplateRequest;

    getTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::GetTemplateOutcome outcome = sesClient.GetTemplate(
            getTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully got template." << std::endl;
    }

    else {
        std::cerr << "Error getting template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetTemplatea Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { GetTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createGetTemplateCommand = (templateName) =>
  new GetTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const getTemplateCommand = createGetTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(getTemplateCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte GetTemplatea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
```

```
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template
```

- Para obter detalhes da API, consulte a GetTemplateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       DATA(lo_result) = lo_ses->gettemplate( iv_templatename =
iv_template_name ).
       oo_template = lo_result->get_template( ).
       MESSAGE 'Template retrieved successfully' TYPE 'I'.
     CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
       DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
       MESSAGE lv_error TYPE 'I'.
       RAISE EXCEPTION lo_ex1.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
       lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
       MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetTemplatereferência da API AWS SDK for SAP ABAP.

Use **`ListIdentities`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListIdentities`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Copiar identidades de domínio e e-mail entre regiões
- Configuração de serviços de envio de e-mails
- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the identities of a specified type for the current account.
    /// </summary>
    /// <param name="identityType">IdentityType to list.</param>
    /// <returns>The list of identities.</returns>
    public async Task<List<string>> ListIdentitiesAsync(IdentityType
identityType)
    {
        var result = new List<string>();
        try
```

```
        {
            var response = await _amazonSimpleEmailService.ListIdentitiesAsync(
                new ListIdentitiesRequest
                {
                    IdentityType = identityType
                });
            result = response.Identities;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListIdentitiesAsync failed with exception: " +
 ex.Message);
        }

        return result;
    }
```

- Para obter detalhes da API, consulte ListIdentitiesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the identities associated with this account.
/*!
  \param identityType: The identity type enum. "NOT_SET" is a valid option.
  \param identities; A vector to receive the retrieved identities.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::listIdentities(Aws::SES::Model::IdentityType identityType,
                                 Aws::Vector<Aws::String> &identities,
```

```
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
   Aws::SES::SESClient sesClient(clientConfiguration);

   Aws::SES::Model::ListIdentitiesRequest listIdentitiesRequest;

   if (identityType != Aws::SES::Model::IdentityType::NOT_SET) {
       listIdentitiesRequest.SetIdentityType(identityType);
   }

   Aws::String nextToken; // Used for paginated results.
   do {
       if (!nextToken.empty()) {
           listIdentitiesRequest.SetNextToken(nextToken);
       }
       Aws::SES::Model::ListIdentitiesOutcome outcome =
sesClient.ListIdentities(
               listIdentitiesRequest);

       if (outcome.IsSuccess()) {
           const auto &retrievedIdentities =
outcome.GetResult().GetIdentities();
           if (!retrievedIdentities.empty()) {
               identities.insert(identities.cend(),
retrievedIdentities.cbegin(),
                                 retrievedIdentities.cend());
           }
           nextToken = outcome.GetResult().GetNextToken();
       }
       else {
           std::cout << "Error listing identities. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   } while (!nextToken.empty());

   return true;
}
```

- Para obter detalhes da API, consulte ListIdentitiesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar todas as identidades (endereços de e-mail e domínios) de uma conta específica AWS

O exemplo a seguir usa o comando `list-identities` para listar todas as identidades que foram enviadas para verificação com o Amazon SES:

```
aws ses list-identities
```

Saída:

```
{
    "Identities": [
      "user@example.com",
      "example.com"
    ]
}
```

A lista retornada contém todas as identidades, independentemente do status da verificação (verificada, verificação pendente, falha etc.).

Neste exemplo, endereços de e-mail e domínios são retornados porque não especificamos o parâmetro identity-type.

Para obter mais informações sobre verificação, consulte Verificar endereços de e-mail e domínios no Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte ListIdentitiesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        listSESIdentities(client);
    }

    public static void listSESIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListIdentitiesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ListIdentitiesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListIdentitiesCommand = () =>
  new ListIdentitiesCommand({ IdentityType: "EmailAddress", MaxItems: 10 });

const run = async () => {
  const listIdentitiesCommand = createListIdentitiesCommand();

  try {
    return await sesClient.send(listIdentitiesCommand);
  } catch (err) {
    console.log("Failed to list identities.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte ListIdentitiesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Esse comando retorna uma lista contendo todas as identidades (endereços de e-mail e domínios) de uma AWS conta específica, independentemente do status da verificação.

```
Get-SESIdentity
```

- Para obter detalhes da API, consulte ListIdentitiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna uma lista contendo todas as identidades (endereços de e-mail e domínios) de uma AWS conta específica, independentemente do status da verificação.

```
Get-SESIdentity
```

- Para obter detalhes da API, consulte ListIdentitiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
 EmailAddress.
```

```
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

- Para obter detalhes da API, consulte a ListIdentitiesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
```

```
                                identity_type: 'EmailAddress'
                              })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Para obter detalhes da API, consulte ListIdentitiesa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_ses->listidentities(
          iv_identitytype = iv_identity_type
          iv_maxitems = iv_max_items
        ).
        ot_identities = lo_result->get_identities( ).
        MESSAGE 'Identities retrieved successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListIdentitiesreferência da API AWS SDK for SAP ABAP.

Use **`ListReceiptFilters`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListReceiptFilters`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the receipt filters associated with this account.
/*!
  \param filters; A vector of "ReceiptFilter" to receive the retrieved filters.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SES::listReceiptFilters(Aws::Vector<Aws::SES::Model::ReceiptFilter>
 &filters,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::ListReceiptFiltersRequest listReceiptFiltersRequest;

    Aws::SES::Model::ListReceiptFiltersOutcome outcome =
 sesClient.ListReceiptFilters(
            listReceiptFiltersRequest);
    if (outcome.IsSuccess()) {
        auto &retrievedFilters = outcome.GetResult().GetFilters();
        if (!retrievedFilters.empty()) {
            filters.insert(filters.cend(), retrievedFilters.cbegin(),
                           retrievedFilters.cend());
        }
    }
    else {
```

```
        std::cerr << "Error retrieving IP address filters: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ListReceiptFiltersa Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ListReceiptFiltersCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListReceiptFiltersCommand = () => new ListReceiptFiltersCommand({});

const run = async () => {
  const listReceiptFiltersCommand = createListReceiptFiltersCommand();

  return await sesClient.send(listReceiptFiltersCommand);
};
```

- Para obter detalhes da API, consulte ListReceiptFiltersa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def list_receipt_filters(self):
        """
        Gets the list of receipt filters for the current account.

        :return: The list of receipt filters.
        """
        try:
            response = self.ses_client.list_receipt_filters()
            filters = response["Filters"]
            logger.info("Got %s receipt filters.", len(filters))
        except ClientError:
            logger.exception("Couldn't get receipt filters.")
            raise
        else:
            return filters
```

- Para obter detalhes da API, consulte a ListReceiptFiltersReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_ses->listreceiptfilters( ).
    ot_filters = lo_result->get_filters( ).
    MESSAGE 'Receipt filters retrieved successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListReceiptFiltersreferência da API AWS SDK for SAP ABAP.

Use **`ListTemplates`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListTemplates`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List email templates for the current account.
    /// </summary>
    /// <returns>A list of template metadata.</returns>
    public async Task<List<TemplateMetadata>> ListEmailTemplatesAsync()
    {
        var result = new List<TemplateMetadata>();
        try
        {
            var response = await _amazonSimpleEmailService.ListTemplatesAsync(
                new ListTemplatesRequest());
            result = response.TemplatesMetadata;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListEmailTemplatesAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- Para obter detalhes da API, consulte ListTemplatesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        listAllTemplates(sesv2Client);
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
 ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();

            ListEmailTemplatesResponse response =
 sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                    .forEach(template -> System.out.println("Template name: " +
 template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListTemplatesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ListTemplatesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListTemplatesCommand = (maxItems) =>
  new ListTemplatesCommand({ MaxItems: maxItems });

const run = async () => {
  const listTemplatesCommand = createListTemplatesCommand(10);

  try {
    return await sesClient.send(listTemplatesCommand);
  } catch (err) {
    console.log("Failed to list templates.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte ListTemplatesa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
```

```
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates
```

- Para obter detalhes da API, consulte a ListTemplatesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_ses->listtemplates( iv_maxitems = iv_max_items ).
        ot_templates = lo_result->get_templatesmetadata( ).
        MESSAGE 'Templates retrieved successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTemplatesreferência da API AWS SDK for SAP ABAP.

Use **`SendBulkTemplatedEmail`**com um AWS SDK

O código de exemplo a seguir mostra como usar `SendBulkTemplatedEmail`.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SendBulkTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL_1 = postfix(getUniqueName("Bilbo"), "@example.com");
const VERIFIED_EMAIL_2 = postfix(getUniqueName("Frodo"), "@example.com");

const USERS = [
  { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL_1 },
  { firstName: "Frodo", emailAddress: VERIFIED_EMAIL_2 },
];

/**
 *
 * @param { { emailAddress: string, firstName: string }[] } users
 * @param { string } templateName the name of an existing template in SES
 * @returns { SendBulkTemplatedEmailCommand }
 */
const createBulkReminderEmailCommand = (users, templateName) => {
  return new SendBulkTemplatedEmailCommand({
    /**
```

```
     * Each 'Destination' uses a corresponding set of replacement data. We can
 map each user
     * to a 'Destination' and provide user specific replacement data to create
 personalized emails.
     *
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{name}},</h1><p>Don't forget about the party gifts!</
p>
     * Destination 1: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!
</p>
     * Destination 2: <h1>Hello Frodo,</h1><p>Don't forget about the party gifts!
</p>
     */
    Destinations: users.map((user) => ({
      Destination: { ToAddresses: [user.emailAddress] },
      ReplacementTemplateData: JSON.stringify({ name: user.firstName }),
    })),
    DefaultTemplateData: JSON.stringify({ name: "Shireling" }),
    Source: VERIFIED_EMAIL_1,
    Template: templateName,
  });
};

const run = async () => {
  const sendBulkTemplateEmailCommand = createBulkReminderEmailCommand(
    USERS,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendBulkTemplateEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte SendBulkTemplatedEmail a Referência AWS SDK para JavaScript da API.

Use **SendEmail**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SendEmail`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Configuração de serviços de envio de e-mails
- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
///  Send an email by using Amazon SES.
/// </summary>
/// <param name="toAddresses">List of recipients.</param>
/// <param name="ccAddresses">List of cc recipients.</param>
/// <param name="bccAddresses">List of bcc recipients.</param>
/// <param name="bodyHtml">Body of the email in HTML.</param>
/// <param name="bodyText">Body of the email in plain text.</param>
/// <param name="subject">Subject line of the email.</param>
/// <param name="senderAddress">From address.</param>
/// <returns>The messageId of the email.</returns>
public async Task<string> SendEmailAsync(List<string> toAddresses,
    List<string> ccAddresses, List<string> bccAddresses,
    string bodyHtml, string bodyText, string subject, string senderAddress)
{
    var messageId = "";
    try
    {
        var response = await _amazonSimpleEmailService.SendEmailAsync(
            new SendEmailRequest
```

```
                {
                    Destination = new Destination
                    {
                        BccAddresses = bccAddresses,
                        CcAddresses = ccAddresses,
                        ToAddresses = toAddresses
                    },
                    Message = new Message
                    {
                        Body = new Body
                        {
                            Html = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyHtml
                            },
                            Text = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyText
                            }
                        },
                        Subject = new Content
                        {
                            Charset = "UTF-8",
                            Data = subject
                        }
                    },
                    Source = senderAddress
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendEmailAsync failed with exception: " +
ex.Message);
        }

        return messageId;
    }
```

- Para obter detalhes da API, consulte SendEmaila Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Send an email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param subject: Email subject.
  \param htmlBody: Email body as HTML. At least one body data is required.
  \param textBody: Email body as plain text. At least one body data is required.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendEmail(const Aws::Vector<Aws::String> &recipients,
                            const Aws::String &subject,
                            const Aws::String &htmlBody,
                            const Aws::String &textBody,
                            const Aws::String &senderEmailAddress,
                            const Aws::Vector<Aws::String> &ccAddresses,
                            const Aws::String &replyToAddress,
                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::Body message_body;
```

```
    if (!htmlBody.empty()) {
        message_body.SetHtml(

 Aws::SES::Model::Content().WithCharset("UTF-8").WithData(htmlBody));
    }

    if (!textBody.empty()) {
        message_body.SetText(

 Aws::SES::Model::Content().WithCharset("UTF-8").WithData(textBody));
    }

    Aws::SES::Model::Message message;
    message.SetBody(message_body);
    message.SetSubject(
            Aws::SES::Model::Content().WithCharset("UTF-8").WithData(subject));

    Aws::SES::Model::SendEmailRequest sendEmailRequest;
    sendEmailRequest.SetDestination(destination);
    sendEmailRequest.SetMessage(message);
    if (!senderEmailAddress.empty()) {
        sendEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendEmail(sendEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending message. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte SendEmaila Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para enviar um e-mail formatado usando o Amazon SES

O exemplo a seguir usa o comando `send-email` para enviar um e-mail formatado:

```
aws ses send-email --from sender@example.com --destination file://
destination.json --message file://message.json
```

Saída:

```
{
   "MessageId": "EXAMPLEf3a5efcd1-51adec81-d2a4-4e3f-9fe2-5d85c1b23783-000000"
}
```

O destino e a mensagem são estruturas de dados JSON salvas em arquivos .json no diretório atual. Esses arquivos são os seguintes:

`destination.json`:

```
{
  "ToAddresses":  ["recipient1@example.com", "recipient2@example.com"],
  "CcAddresses":  ["recipient3@example.com"],
  "BccAddresses": []
}
```

`message.json`:

```
{
   "Subject": {
       "Data": "Test email sent using the AWS CLI",
       "Charset": "UTF-8"
   },
   "Body": {
       "Text": {
           "Data": "This is the message body in text format.",
           "Charset": "UTF-8"
       },
       "Html": {
```

```
            "Data": "This message body contains HTML formatting. It can, for
 example, contain links like this one: <a class=\"ulink\" href=\"http://
docs.aws.amazon.com/ses/latest/DeveloperGuide\" target=\"_blank\">Amazon SES
 Developer Guide</a>.",
            "Charset": "UTF-8"
        }
    }
}
```

Substitua os endereços de e-mail do remetente e do destinatário por aqueles que você deseja usar. O endereço de e-mail do remetente deverá ser verificado com o Amazon SES. Até que você tenha acesso de produção ao Amazon SES, você também deverá verificar o endereço de e-mail de cada destinatário, a menos que o destinatário seja o simulador de caixa de correio do Amazon SES. Para obter mais informações sobre verificação, consulte Verificar endereços de e-mail e domínios no Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

O ID da mensagem na saída indica que a chamada para send-email foi bem-sucedida.

Se você não receber o e-mail, verifique a caixa de lixo eletrônico.

Para obter mais informações sobre como enviar e-mail formatado, consulte Envio de e-mail formatado usando a API do Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte SendEmailem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
```

```
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.
\s
                    subject - The  subject line.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
                .region(region)
                .build();
```

```
        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</
h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        try {
            send(client, sender, recipient, subject, bodyHTML);
            client.close();
            System.out.println("Done");

        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void send(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyHTML) throws MessagingException {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
```

```
                .message(msg)
                .source(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES "
 + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject> <fileLocation>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.
\s
                    subject - The  subject line.\s
                    fileLocation - The location of a Microsoft Excel file to use
 as an attachment (C:/AWS/customers.xls).\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list
 "
                + "of customers to contact.";

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</
h1>"
                + "<p>Please see the attached file for a " + "list of customers
 to contact.</p>" + "</body>"
                + "</html>";

        Region region = Region.US_WEST_2;
```

```
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
            client.close();
            System.out.println("Done");

        } catch (IOException | MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void sendemailAttachment(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyText,
            String bodyHTML,
            String fileLocation) throws AddressException, MessagingException,
IOException {

        java.io.File theFile = new java.io.File(fileLocation);
        byte[] fileContent = Files.readAllBytes(theFile.toPath());

        Session session = Session.getDefaultInstance(new Properties());

        // Create a new MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Add subject, from and to lines.
        message.setSubject(subject, "UTF-8");
        message.setFrom(new InternetAddress(sender));
        message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

        // Create a multipart/alternative child container.
        MimeMultipart msgBody = new MimeMultipart("alternative");

        // Create a wrapper for the HTML and text parts.
        MimeBodyPart wrap = new MimeBodyPart();
```

```
        // Define the text part.
        MimeBodyPart textPart = new MimeBodyPart();
        textPart.setContent(bodyText, "text/plain; charset=UTF-8");

        // Define the HTML part.
        MimeBodyPart htmlPart = new MimeBodyPart();
        htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

        // Add the text and HTML parts to the child container.
        msgBody.addBodyPart(textPart);
        msgBody.addBodyPart(htmlPart);

        // Add the child container to the wrapper object.
        wrap.setContent(msgBody);

        // Create a multipart/mixed parent container.
        MimeMultipart msg = new MimeMultipart("mixed");

        // Add the parent container to the message.
        message.setContent(msg);
        msg.addBodyPart(wrap);

        // Define the attachment.
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new ByteArrayDataSource(fileContent,
                "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
        att.setDataHandler(new DataHandler(fds));

        String reportName = "WorkReport.xls";
        att.setFileName(reportName);

        // Add the attachment to the message.
        msg.addBodyPart(att);

        try {
            System.out.println("Attempting to send an email through Amazon SES "
 + "using the AWS SDK for Java...");

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);

            ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());
```

```
            byte[] arr = new byte[buf.remaining()];
            buf.get(arr);

            SdkBytes data = SdkBytes.fromByteArray(arr);
            RawMessage rawMessage = RawMessage.builder()
                    .data(data)
                    .build();

            SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
                    .rawMessage(rawMessage)
                    .build();

            client.sendRawEmail(rawEmailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Email sent using SesClient with attachment");
    }
}
```

- Para obter detalhes da API, consulte SendEmaila Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SendEmailCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createSendEmailCommand = (toAddress, fromAddress) => {
  return new SendEmailCommand({
    Destination: {
      /* required */
```

```
      CcAddresses: [
        /* more items */
      ],
      ToAddresses: [
        toAddress,
        /* more To-email addresses */
      ],
    },
    Message: {
      /* required */
      Body: {
        /* required */
        Html: {
          Charset: "UTF-8",
          Data: "HTML_FORMAT_BODY",
        },
        Text: {
          Charset: "UTF-8",
          Data: "TEXT_FORMAT_BODY",
        },
      },
      Subject: {
        Charset: "UTF-8",
        Data: "EMAIL_SUBJECT",
      },
    },
    Source: fromAddress,
    ReplyToAddresses: [
      /* more items */
    ],
  });
};

const run = async () => {
  const sendEmailCommand = createSendEmailCommand(
    "recipient@example.com",
    "sender@example.com",
  );

  try {
    return await sesClient.send(sendEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
```

```
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte SendEmaila Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html,
 reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
```

```
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the
recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source,
destination.tos
            )
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id
```

- Para obter detalhes da API, consulte a SendEmailReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
```

```
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- Para obter detalhes da API, consulte SendEmaila Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Create message object
DATA(lo_subject) = NEW /aws1/cl_sescontent( iv_data = iv_subject ).
DATA(lo_text_body) = NEW /aws1/cl_sescontent( iv_data = iv_text ).
DATA(lo_html_body) = NEW /aws1/cl_sescontent( iv_data = iv_html ).
DATA(lo_body) = NEW /aws1/cl_sesbody(
  io_text = lo_text_body
  io_html = lo_html_body
).
DATA(lo_message) = NEW /aws1/cl_sesmessage(
  io_subject = lo_subject
  io_body = lo_body
).

TRY.
    " Send email
    DATA(lo_result) = lo_ses->sendemail(
      iv_source = iv_source
      io_destination = io_destination
      io_message = lo_message
      it_replytoaddresses = it_reply_tos
    ).
    ov_msg_id = lo_result->get_messageid( ).
    MESSAGE 'Email sent successfully' TYPE 'I'.
  CATCH /aws1/cx_sesacctsendingpause00 INTO DATA(lo_ex1).
    DATA(lv_error) = |Account sending paused: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_sesmessagerejected INTO DATA(lo_ex2).
    lv_error = |Message rejected: { lo_ex2->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex2.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
```

```
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendEmailreferência da API AWS SDK for SAP ABAP.

Use **SendRawEmail**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SendRawEmail`.

CLI

AWS CLI

Para enviar e-mail bruto usando o Amazon SES

O exemplo a seguir usa o comando `send-raw-email` para enviar um e-mail com um anexo TXT:

```
aws ses send-raw-email --raw-message file://message.json
```

Saída:

```
{
   "MessageId": "EXAMPLEf3f73d99b-c63fb06f-d263-41f8-a0fb-d0dc67d56c07-000000"
}
```

A mensagem bruta é uma estrutura de dados JSON salva em um arquivo chamado `message.json` no diretório atual. Ele contém o seguinte:

```
{
   "Data": "From: sender@example.com\nTo: recipient@example.com\nSubject:
 Test email sent using the AWS CLI (contains an attachment)\nMIME-Version:
 1.0\nContent-type: Multipart/Mixed; boundary=\"NextPart\"\n\n--NextPart
\nContent-Type: text/plain\n\nThis is the message body.\n\n--NextPart\nContent-
Type: text/plain;\nContent-Disposition: attachment; filename=\"attachment.txt\"\n
\nThis is the text in the attachment.\n\n--NextPart--"
}
```

Como você pode ver, “Dados” é uma longa sequência de caracteres que contém todo o conteúdo bruto do e-mail no formato MIME, incluindo um anexo chamado attachment.txt.

Substitua sender@example.com e recipient@example.com pelos endereços que você deseja usar. O endereço de e-mail do remetente deverá ser verificado com o Amazon SES. Até que você tenha acesso de produção ao Amazon SES, você também deverá verificar o endereço de e-mail do destinatário, a menos que o destinatário seja o simulador de caixa de correio do Amazon SES. Para obter mais informações sobre verificação, consulte Verificar endereços de e-mail e domínios no Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

O ID da mensagem na saída indica que a chamada para send-raw-email foi bem-sucedida.

Se você não receber o e-mail, verifique a caixa de lixo eletrônico.

Para obter mais informações sobre como enviar e-mail bruto, consulte Enviar e-mail bruto usando a API do Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte SendRawEmailem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use nodemailer para enviar um e-mail com anexo.

```
import sesClientModule from "@aws-sdk/client-ses";
/**
 * nodemailer wraps the SES SDK and calls SendRawEmail. Use this for more
 advanced
 * functionality like adding attachments to your email.
 *
 * https://nodemailer.com/transports/ses
```

```
 */
import nodemailer from "nodemailer";

/**
 * @param {string} from An Amazon SES verified email address.
 * @param {*} to An Amazon SES verified email address.
 */
export const sendEmailWithAttachments = (
  from = "from@example.com",
  to = "to@example.com",
) => {
  const ses = new sesClientModule.SESClient({});
  const transporter = nodemailer.createTransport({
    SES: { ses, aws: sesClientModule },
  });

  return new Promise((resolve, reject) => {
    transporter.sendMail(
      {
        from,
        to,
        subject: "Hello World",
        text: "Greetings from Amazon SES!",
        attachments: [{ content: "Hello World!", filename: "hello.txt" }],
      },
      (err, info) => {
        if (err) {
          reject(err);
        } else {
          resolve(info);
        }
      },
    );
  });
};
```

- Para obter detalhes da API, consulte SendRawEmaila Referência AWS SDK para JavaScript da API.

Use **`SendTemplatedEmail`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `SendTemplatedEmail`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Send an email using a template.
    /// </summary>
    /// <param name="sender">Address of the sender.</param>
    /// <param name="recipients">Addresses of the recipients.</param>
    /// <param name="templateName">Name of the email template.</param>
    /// <param name="templateDataObject">Data for the email template.</param>
    /// <returns>The messageId of the email.</returns>
    public async Task<string> SendTemplateEmailAsync(string sender, List<string>
recipients,
        string templateName, object templateDataObject)
    {
        var messageId = "";
        try
        {
            // Template data should be serialized JSON from either a class or a
dynamic object.
            var templateData = JsonSerializer.Serialize(templateDataObject);

            var response = await
_amazonSimpleEmailService.SendTemplatedEmailAsync(
                new SendTemplatedEmailRequest
                {
                    Source = sender,
                    Destination = new Destination
```

```
                    {
                        ToAddresses = recipients
                    },
                    Template = templateName,
                    TemplateData = templateData
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendTemplateEmailAsync failed with exception: " +
 ex.Message);
        }

        return messageId;
    }
```

- Para obter detalhes da API, consulte SendTemplatedEmail a Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Send a templated email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param templateName: The name of the template to use.
  \param templateData: Map of key-value pairs for replacing text in template.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendTemplatedEmail(const Aws::Vector<Aws::String> &recipients,
                                     const Aws::String &templateName,
                                     const Aws::Map<Aws::String, Aws::String>
 &templateData,
                                     const Aws::String &senderEmailAddress,
                                     const Aws::Vector<Aws::String> &ccAddresses,
                                     const Aws::String &replyToAddress,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::SendTemplatedEmailRequest sendTemplatedEmailRequest;
    sendTemplatedEmailRequest.SetDestination(destination);
    sendTemplatedEmailRequest.SetTemplate(templateName);

    std::ostringstream templateDataStream;
    templateDataStream << "{";
    size_t dataCount = 0;
    for (auto &pair: templateData) {
        templateDataStream << "\"" << pair.first << "\":\"" << pair.second <<
 "\"";
        dataCount++;
        if (dataCount < templateData.size()) {
            templateDataStream << ",";
        }
    }
    templateDataStream << "}";

    sendTemplatedEmailRequest.SetTemplateData(templateDataStream.str());

    if (!senderEmailAddress.empty()) {
        sendTemplatedEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
```

```
        sendTemplatedEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendTemplatedEmail(sendTemplatedEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent templated message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending templated message. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte SendTemplatedEmaila Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;
```

```
/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <template> <sender> <recipient>\s

                Where:
                    template - The name of the email template.
                    sender - An email address that represents the sender.\s
                    recipient - An email address that represents the recipient.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String templateName = args[0];
        String sender = args[1];
        String recipient = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        send(sesv2Client, sender, recipient, templateName);
    }
```

```
    public static void send(SesV2Client client, String sender, String recipient,
 String templateName) {
        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        /*
         * Specify both name and favorite animal (favoriteanimal) in your code
 when
         * defining the Template object.
         * If you don't specify all the variables in the template, Amazon SES
 doesn't
         * send the email.
         */
        Template myTemplate = Template.builder()
                .templateName(templateName)
                .templateData("{\n" +
                        "  \"name\": \"Jason\"\n," +
                        "  \"favoriteanimal\": \"Cat\"\n" +
                        "}")
                .build();

        EmailContent emailContent = EmailContent.builder()
                .template(myTemplate)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email based on a template
 using the AWS SDK for Java (v2)...");
            client.sendEmail(emailRequest);
            System.out.println("email based on a template was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SendTemplatedEmaila Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SendTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL = postfix(getUniqueName("Bilbo"), "@example.com");

const USER = { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL };

/**
 *
 * @param { { emailAddress: string, firstName: string } } user
 * @param { string } templateName - The name of an existing template in Amazon
 SES.
 * @returns { SendTemplatedEmailCommand }
 */
```

```
const createReminderEmailCommand = (user, templateName) => {
  return new SendTemplatedEmailCommand({
    /**
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{contact.firstName}},</h1><p>Don't forget about the
 party gifts!</p>
     * Destination: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!</
p>
     */
    Destination: { ToAddresses: [user.emailAddress] },
    TemplateData: JSON.stringify({ contact: { firstName: user.firstName } }),
    Source: VERIFIED_EMAIL,
    Template: templateName,
  });
};

const run = async () => {
  const sendReminderEmailCommand = createReminderEmailCommand(
    USER,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendReminderEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte SendTemplatedEmaila Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
 passed
        in this function contains key-value pairs that define the values to
 insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement
 values
                              that are inserted in the template before it is
 sent.
        :return: The ID of the message, assigned by Amazon SES.
```

```
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
            raise
        else:
            return message_id
```

- Para obter detalhes da API, consulte a SendTemplatedEmailReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Send templated email
        DATA(lo_result) = lo_ses->sendtemplatedemail(
          iv_source = iv_source
          io_destination = io_destination
          iv_template = iv_template_name
          iv_templatedata = iv_template_data
          it_replytoaddresses = it_reply_tos
        ).
        ov_msg_id = lo_result->get_messageid( ).
        MESSAGE 'Templated email sent successfully' TYPE 'I'.
      CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
        DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendTemplatedEmailreferência da API AWS SDK for SAP ABAP.

Use **`UpdateTemplate`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `UpdateTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Verificar uma identidade de e-mail e enviar mensagens

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::updateTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Template templateValues;

    templateValues.SetTemplateName(templateName);
    templateValues.SetSubjectPart(subjectPart);
    templateValues.SetHtmlPart(htmlPart);
    templateValues.SetTextPart(textPart);

    Aws::SES::Model::UpdateTemplateRequest updateTemplateRequest;
    updateTemplateRequest.SetTemplate(templateValues);

    Aws::SES::Model::UpdateTemplateOutcome outcome =
 sesClient.UpdateTemplate(updateTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated template." << std::endl;
```

```
    } else {
        std::cerr << "Error updating template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateTemplatea Referência AWS SDK para C++ da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { UpdateTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");
const HTML_PART = "<h1>Hello, World!</h1>";

const createUpdateTemplateCommand = () => {
  return new UpdateTemplateCommand({
    Template: {
      TemplateName: TEMPLATE_NAME,
      HtmlPart: HTML_PART,
      SubjectPart: "Example",
      TextPart: "Updated template text.",
    },
  });
};
```

```
const run = async () => {
  const updateTemplateCommand = createUpdateTemplateCommand();

  try {
    return await sesClient.send(updateTemplateCommand);
  } catch (err) {
    console.log("Failed to update template.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte UpdateTemplatea Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
```

```
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

- Para obter detalhes da API, consulte a UpdateTemplateReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA(lo_template) = NEW /aws1/cl_sestemplate(
  iv_templatename = iv_name
  iv_subjectpart = iv_subject
  iv_textpart = iv_text
  iv_htmlpart = iv_html
).

TRY.
    lo_ses->updatetemplate( io_template = lo_template ).
    MESSAGE 'Template updated successfully' TYPE 'I'.
  CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
    DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_sesinvalidtemplateex INTO DATA(lo_ex2).
    lv_error = |Invalid template: { lo_ex2->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex2.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateTemplatereferência da API AWS SDK for SAP ABAP.

Use **`VerifyDomainIdentity`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `VerifyDomainIdentity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Copiar identidades de domínio e e-mail entre regiões
- Configuração de serviços de envio de e-mails
- Verificar uma identidade de e-mail e enviar mensagens

CLI

AWS CLI

Para verificar um domínio com o Amazon SES

O exemplo a seguir usa o comando `verify-domain-identity` para verificar um domínio:

```
aws ses verify-domain-identity --domain example.com
```

Saída:

```
{
   "VerificationToken": "eoEmxw+YaYhb3h3iVJHuXMJXqeu1q1/wwmvjuEXAMPLE"
}
```

Para concluir a verificação do domínio, você deverá adicionar um registro TXT com o token de verificação retornado às configurações de DNS do seu domínio. Para obter mais informações, consulte Verificar domínios no Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte VerifyDomainIdentityem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { VerifyDomainIdentityCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * You must have access to the domain's DNS settings to complete the
 * domain verification process.
 */
const DOMAIN_NAME = postfix(getUniqueName("Domain"), ".example.com");

const createVerifyDomainIdentityCommand = () => {
  return new VerifyDomainIdentityCommand({ Domain: DOMAIN_NAME });
};

const run = async () => {
  const VerifyDomainIdentityCommand = createVerifyDomainIdentityCommand();

  try {
    return await sesClient.send(VerifyDomainIdentityCommand);
  } catch (err) {
    console.log("Failed to verify domain.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte VerifyDomainIdentitya Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
        """
        Starts verification of a domain identity. To complete verification, you
 must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token
```

- Para obter detalhes da API, consulte a VerifyDomainIdentityReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       DATA(lo_result) = lo_ses->verifydomainidentity( iv_domain =
iv_domain_name ).
       ov_token = lo_result->get_verificationtoken( ).
       MESSAGE 'Domain verification initiated' TYPE 'I'.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
       DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
       MESSAGE lv_error TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a VerifyDomainIdentityreferência da API AWS SDK for SAP ABAP.

Use **`VerifyEmailIdentity`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `VerifyEmailIdentity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Copiar identidades de domínio e e-mail entre regiões
- Configuração de serviços de envio de e-mails
- Verificar uma identidade de e-mail e enviar mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Starts verification of an email identity. This request sends an email
   /// from Amazon SES to the specified email address. To complete
   /// verification, follow the instructions in the email.
   /// </summary>
   /// <param name="recipientEmailAddress">Email address to verify.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> VerifyEmailIdentityAsync(string
recipientEmailAddress)
   {
       var success = false;
       try
       {
           var response = await
_amazonSimpleEmailService.VerifyEmailIdentityAsync(
               new VerifyEmailIdentityRequest
               {
                   EmailAddress = recipientEmailAddress
               });

           success = response.HttpStatusCode == HttpStatusCode.OK;
       }
       catch (Exception ex)
       {
           Console.WriteLine("VerifyEmailIdentityAsync failed with exception: "
+ ex.Message);
       }

       return success;
   }
```

- Para obter detalhes da API, consulte VerifyEmailIdentitya Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Add an email address to the list of identities associated with this account
 and
//! initiate verification.
/*!
  \param emailAddress; The email address to add.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::verifyEmailIdentity(const Aws::String &emailAddress,
                        const Aws::Client::ClientConfiguration
 &clientConfiguration)
{
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::VerifyEmailIdentityRequest verifyEmailIdentityRequest;

    verifyEmailIdentityRequest.SetEmailAddress(emailAddress);

    Aws::SES::Model::VerifyEmailIdentityOutcome outcome =
 sesClient.VerifyEmailIdentity(verifyEmailIdentityRequest);

    if (outcome.IsSuccess())
    {
        std::cout << "Email verification initiated." << std::endl;
    }
```

```
    else
    {
        std::cerr << "Error initiating email verification. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte VerifyEmailIdentitya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para verificar um endereço de e-mail com o Amazon SES

O exemplo a seguir usa o comando `verify-email-identity` para verificar um endereço de e-mail:

```
aws ses verify-email-identity --email-address user@example.com
```

Antes de enviar e-mails usando o Amazon SES, você deve verificar que o endereço ou domínio do qual você está enviando o email para provar que você é o proprietário. Se você ainda não tem acesso de produção, também precisará verificar todos os endereços de e-mail aos quais envia e-mails, exceto aqueles fornecidos pelo simulador de caixa de correio do Amazon SES.

Depois que verify-email-identity for chamado, o endereço de e-mail receberá um e-mail de verificação. O usuário deve clicar no link do e-mail para concluir o processo de verificação.

Para saber mais, consulte Verificar endereços de e-mail no Amazon SES no Guia do desenvolvedor do Amazon Simple Email Service.

- Para obter detalhes da API, consulte VerifyEmailIdentityem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Import required AWS SDK clients and commands for Node.js
import { VerifyEmailIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const EMAIL_ADDRESS = "name@example.com";

const createVerifyEmailIdentityCommand = (emailAddress) => {
  return new VerifyEmailIdentityCommand({ EmailAddress: emailAddress });
};

const run = async () => {
  const verifyEmailIdentityCommand =
    createVerifyEmailIdentityCommand(EMAIL_ADDRESS);
  try {
    return await sesClient.send(verifyEmailIdentityCommand);
  } catch (err) {
    console.log("Failed to verify email identity.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte VerifyEmailIdentitya Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise
```

- Para obter detalhes da API, consulte a VerifyEmailIdentityReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
                              email_address: recipient
                            })

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- Para obter detalhes da API, consulte VerifyEmailIdentitya Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ses->verifyemailidentity( iv_emailaddress = iv_email_address ).
    MESSAGE 'Email verification initiated' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a VerifyEmailIdentityreferência da API AWS SDK for SAP ABAP.

Cenários para o Amazon SES usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon SES com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon SES ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação de transmissão do Amazon Transcribe
- Copie as identidades de e-mail e domínio do Amazon SES de uma AWS Região para outra usando um AWS SDK

- Criar uma aplicação Web para monitorar dados do DynamoDB
- Criar um rastreador de itens do Amazon Redshift
- Crie um rastreador de itens de trabalho do Aurora Sem Servidor
- Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK
- Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
- Gerar credenciais para estabelecer conexão com um endpoint SMTP do Amazon SES
- Configuração de serviços de envio de e-mails
- Usar Step Functions para invocar funções do Lambda
- Verifique uma identidade de e-mail e envie mensagens com o Amazon SES usando um AWS SDK

Criar uma aplicação de transmissão do Amazon Transcribe

O exemplo de código a seguir mostra como construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Transcribe para construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Copie as identidades de e-mail e domínio do Amazon SES de uma AWS Região para outra usando um AWS SDK

O exemplo de código a seguir mostra como copiar as identidades de e-mail e domínio do Amazon SES de uma AWS região para outra. Quando as identidades de domínio são gerenciadas pelo Route 53, os registros de verificação são copiados para o domínio da região de destino.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_identities(ses_client):
    """
    Gets the identities for the current Region. The Region is specified in the
    Boto3 Amazon SES client object.

    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of email identities and the list of domain identities.
    """
    email_identities = []
    domain_identities = []
    try:
        identity_paginator = ses_client.get_paginator("list_identities")
        identity_iterator = identity_paginator.paginate(
            PaginationConfig={"PageSize": 20}
        )
```

```
        for identity_page in identity_iterator:
            for identity in identity_page["Identities"]:
                if "@" in identity:
                    email_identities.append(identity)
                else:
                    domain_identities.append(identity)
        logger.info(
            "Found %s email and %s domain identities.",
            len(email_identities),
            len(domain_identities),
        )
    except ClientError:
        logger.exception("Couldn't get identities.")
        raise
    else:
        return email_identities, domain_identities


def verify_emails(email_list, ses_client):
    """
    Starts verification of a list of email addresses. Verification causes an
 email
    to be sent to each address. To complete verification, the recipient must
 follow
    the instructions in the email.

    :param email_list: The list of email addresses to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of emails that were successfully submitted for
 verification.
    """
    verified_emails = []
    for email in email_list:
        try:
            ses_client.verify_email_identity(EmailAddress=email)
            verified_emails.append(email)
            logger.info("Started verification of %s.", email)
        except ClientError:
            logger.warning("Couldn't start verification of %s.", email)
    return verified_emails


def verify_domains(domain_list, ses_client):
    """
```

```
    Starts verification for a list of domain identities. This returns a token for
    each domain, which must be registered as a TXT record with the DNS provider
 for
    the domain.

    :param domain_list: The list of domains to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The generated domain tokens to use to completed verification.
    """
    domain_tokens = {}
    for domain in domain_list:
        try:
            response = ses_client.verify_domain_identity(Domain=domain)
            token = response["VerificationToken"]
            domain_tokens[domain] = token
            logger.info("Got verification token %s for domain %s.", token,
 domain)
        except ClientError:
            logger.warning("Couldn't get verification token for domain %s.",
 domain)
    return domain_tokens


def get_hosted_zones(route53_client):
    """
    Gets the Amazon Route 53 hosted zones for the current account.

    :param route53_client: A Boto3 Route 53 client.
    :return: The list of hosted zones.
    """
    zones = []
    try:
        zone_paginator = route53_client.get_paginator("list_hosted_zones")
        zone_iterator = zone_paginator.paginate(PaginationConfig={"PageSize":
 20})
        zones = [
            zone for zone_page in zone_iterator for zone in
 zone_page["HostedZones"]
        ]
        logger.info("Found %s hosted zones.", len(zones))
    except ClientError:
        logger.warning("Couldn't get hosted zones.")
    return zones
```

```
def find_domain_zone_matches(domains, zones):
    """
    Finds matches between Amazon SES verified domains and Route 53 hosted zones.
    Subdomain matches are taken when found, otherwise root domain matches are
 taken.

    :param domains: The list of domains to match.
    :param zones: The list of hosted zones to match.
    :return: The set of matched domain-zone pairs. When a match is not found, the
             domain is included in the set with a zone value of None.
    """
    domain_zones = {}
    for domain in domains:
        domain_zones[domain] = None
        # Start at the most specific sub-domain and walk up to the root domain
 until a
        # zone match is found.
        domain_split = domain.split(".")
        for index in range(0, len(domain_split) - 1):
            sub_domain = ".".join(domain_split[index:])
            for zone in zones:
                # Normalize the zone name from Route 53 by removing the trailing
 '.'.
                zone_name = zone["Name"][:-1]
                if sub_domain == zone_name:
                    domain_zones[domain] = zone
                    break
            if domain_zones[domain] is not None:
                break
    return domain_zones


def add_route53_verification_record(domain, token, zone, route53_client):
    """
    Adds a domain verification TXT record to the specified Route 53 hosted zone.
    When a TXT record already exists in the hosted zone for the specified domain,
    the existing values are preserved and the new token is added to the list.

    :param domain: The domain to add.
    :param token: The verification token for the domain.
    :param zone: The hosted zone where the domain verification record is added.
    :param route53_client: A Boto3 Route 53 client.
    """
```

```
    domain_token_record_set_name = f"_amazonses.{domain}"
    record_set_paginator =
route53_client.get_paginator("list_resource_record_sets")
    record_set_iterator = record_set_paginator.paginate(
        HostedZoneId=zone["Id"], PaginationConfig={"PageSize": 20}
    )
    records = []
    for record_set_page in record_set_iterator:
        try:
            txt_record_set = next(
                record_set
                for record_set in record_set_page["ResourceRecordSets"]
                if record_set["Name"][:-1] == domain_token_record_set_name
                and record_set["Type"] == "TXT"
            )
            records = txt_record_set["ResourceRecords"]
            logger.info(
                "Existing TXT record found in set %s for zone %s.",
                domain_token_record_set_name,
                zone["Name"],
            )
            break
        except StopIteration:
            pass
    records.append({"Value": json.dumps(token)})
    changes = [
        {
            "Action": "UPSERT",
            "ResourceRecordSet": {
                "Name": domain_token_record_set_name,
                "Type": "TXT",
                "TTL": 1800,
                "ResourceRecords": records,
            },
        }
    ]
    try:
        route53_client.change_resource_record_sets(
            HostedZoneId=zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Created or updated the TXT record in set %s for zone %s.",
            domain_token_record_set_name,
            zone["Name"],
```

```
        )
    except ClientError as err:
        logger.warning(
            "Got error %s. Couldn't create or update the TXT record for zone
 %s.",
            err.response["Error"]["Code"],
            zone["Name"],
        )


def generate_dkim_tokens(domain, ses_client):
    """
    Generates DKIM tokens for a domain. These must be added as CNAME records to
 the
    DNS provider for the domain.

    :param domain: The domain to generate tokens for.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of generated DKIM tokens.
    """
    dkim_tokens = []
    try:
        dkim_tokens = ses_client.verify_domain_dkim(Domain=domain)["DkimTokens"]
        logger.info("Generated %s DKIM tokens for domain %s.", len(dkim_tokens),
 domain)
    except ClientError:
        logger.warning("Couldn't generate DKIM tokens for domain %s.", domain)
    return dkim_tokens


def add_dkim_domain_tokens(hosted_zone, domain, tokens, route53_client):
    """
    Adds DKIM domain token CNAME records to a Route 53 hosted zone.

    :param hosted_zone: The hosted zone where the records are added.
    :param domain: The domain to add.
    :param tokens: The DKIM tokens for the domain to add.
    :param route53_client: A Boto3 Route 53 client.
    """
    try:
        changes = [
            {
                "Action": "UPSERT",
                "ResourceRecordSet": {
```

```
                    "Name": f"{token}._domainkey.{domain}",
                    "Type": "CNAME",
                    "TTL": 1800,
                    "ResourceRecords": [{"Value":
 f"{token}.dkim.amazonses.com"}],
                },
            }
            for token in tokens
        ]
        route53_client.change_resource_record_sets(
            HostedZoneId=hosted_zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Added %s DKIM CNAME records to %s in zone %s.",
            len(tokens),
            domain,
            hosted_zone["Name"],
        )
    except ClientError:
        logger.warning(
            "Couldn't add DKIM CNAME records for %s to zone %s.",
            domain,
            hosted_zone["Name"],
        )


def configure_sns_topics(identity, topics, ses_client):
    """
    Configures Amazon Simple Notification Service (Amazon SNS) notifications for
    an identity. The Amazon SNS topics must already exist.

    :param identity: The identity to configure.
    :param topics: The list of topics to configure. The choices are Bounce,
 Delivery,
                   or Complaint.
    :param ses_client: A Boto3 Amazon SES client.
    """
    for topic in topics:
        topic_arn = input(
            f"Enter the Amazon Resource Name (ARN) of the {topic} topic or press
 "
            f"Enter to skip: "
        )
        if topic_arn != "":
```

```
            try:
                ses_client.set_identity_notification_topic(
                    Identity=identity, NotificationType=topic, SnsTopic=topic_arn
                )
                logger.info("Configured %s for %s notifications.", identity,
 topic)
            except ClientError:
                logger.warning(
                    "Couldn't configure %s for %s notifications.", identity,
 topic
                )


def replicate(source_client, destination_client, route53_client):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print(
        f"Replicating Amazon SES identities and other configuration from "
        f"{source_client.meta.region_name} to
 {destination_client.meta.region_name}."
    )
    print("-" * 88)

    print(f"Retrieving identities from {source_client.meta.region_name}.")
    source_emails, source_domains = get_identities(source_client)
    print("Email addresses found:")
    print(*source_emails)
    print("Domains found:")
    print(*source_domains)

    print("Starting verification for email identities.")
    dest_emails = verify_emails(source_emails, destination_client)
    print("Getting domain tokens for domain identities.")
    dest_domain_tokens = verify_domains(source_domains, destination_client)

    # Get Route 53 hosted zones and match them with Amazon SES domains.
    answer = input(
        "Is the DNS configuration for your domains managed by Amazon Route 53 (y/
n)? "
    )
    use_route53 = answer.lower() == "y"
    hosted_zones = get_hosted_zones(route53_client) if use_route53 else []
    if use_route53:
```

```
        print("Adding or updating Route 53 TXT records for your domains.")
        domain_zones = find_domain_zone_matches(dest_domain_tokens.keys(),
 hosted_zones)
        for domain in domain_zones:
            add_route53_verification_record(
                domain, dest_domain_tokens[domain], domain_zones[domain],
 route53_client
            )
    else:
        print(
            "Use these verification tokens to create TXT records through your DNS
 "
            "provider:"
        )
        pprint(dest_domain_tokens)

    answer = input("Do you want to configure DKIM signing for your identities (y/
n)? ")
    if answer.lower() == "y":
        # Build a set of unique domains from email and domain identities.
        domains = {email.split("@")[1] for email in dest_emails}
        domains.update(dest_domain_tokens)
        domain_zones = find_domain_zone_matches(domains, hosted_zones)
        for domain, zone in domain_zones.items():
            answer = input(
                f"Do you want to configure DKIM signing for {domain} (y/n)? "
            )
            if answer.lower() == "y":
                dkim_tokens = generate_dkim_tokens(domain, destination_client)
                if use_route53 and zone is not None:
                    add_dkim_domain_tokens(zone, domain, dkim_tokens,
 route53_client)
                else:
                    print(
                        "Add the following DKIM tokens as CNAME records through
 your "
                        "DNS provider:"
                    )
                    print(*dkim_tokens, sep="\n")

    answer = input(
        "Do you want to configure Amazon SNS notifications for your identities
 (y/n)? "
    )
```

```
    if answer.lower() == "y":
        for identity in dest_emails + list(dest_domain_tokens.keys()):
            answer = input(
                f"Do you want to configure Amazon SNS topics for {identity} (y/
n)? "
            )
            if answer.lower() == "y":
                configure_sns_topics(
                    identity, ["Bounce", "Delivery", "Complaint"],
 destination_client
                )

    print(f"Replication complete for {destination_client.meta.region_name}.")
    print("-" * 88)


def main():
    boto3_session = boto3.Session()
    ses_regions = boto3_session.get_available_regions("ses")
    parser = argparse.ArgumentParser(
        description="Copies email address and domain identities from one AWS
 Region to "
        "another. Optionally adds records for domain verification and DKIM "
        "signing to domains that are managed by Amazon Route 53, "
        "and sets up Amazon SNS notifications for events of interest."
    )
    parser.add_argument(
        "source_region", choices=ses_regions, help="The region to copy from."
    )
    parser.add_argument(
        "destination_region", choices=ses_regions, help="The region to copy to."
    )
    args = parser.parse_args()
    source_client = boto3.client("ses", region_name=args.source_region)
    destination_client = boto3.client("ses", region_name=args.destination_region)
    route53_client = boto3.client("route53")
    replicate(source_client, destination_client, route53_client)


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - ListIdentities
  - SetIdentityNotificationTopic
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Java

SDK para Java 2.x

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho no Amazon DynamoDB e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em uma tabela do DynamoDB.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo no Repositório de exemplos de AWS código em GitHub.

Serviços usados neste exemplo

- DynamoDB

- Amazon SES

Criar um rastreador de itens do Amazon Redshift

Os exemplos de código a seguir mostram como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho usando um banco de dados do Amazon Redshift.

Java

SDK para Java 2.x

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- banco de dados de origem
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- banco de dados de origem
- Amazon SES

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

Os exemplos de código a seguir mostram como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

.NET

SDK para .NET

Mostra como usar o AWS SDK para .NET para criar um aplicativo web que rastreia itens de trabalho em um banco de dados Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful .NET.

- Integre um aplicativo web React com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

C++

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar uma API REST C++ que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Java

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar um aplicativo web que rastreia itens de trabalho em um banco de dados Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um Node.js back-end Express.

- Integre um aplicativo React.js web com Serviços da AWS o.
- Liste, adicione e atualize itens em uma tabela do Aurora.

- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

PHP

SDK para PHP

Mostra como usar o AWS SDK para PHP para criar uma aplicação web que rastreia itens de trabalho em um banco de dados do Amazon RDS e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end PHP RESTful.

- Integre um aplicativo React.js da web com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um AWS Secrets Manager segredo que contenha as credenciais do banco de dados e use-o para autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS

- Serviços de dados do Amazon RDS
- Amazon SES

Detecte PPE em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detecte objetos em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Rekognition AWS SDK para JavaScript com o para criar um aplicativo que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Aprenda como:

- Criar um usuário não autenticado usando o Amazon Cognito.
- Analisar imagens em busca de objetos usando o Amazon Rekognition.
- Verificar um endereço de e-mail para o Amazon SES.
- Enviar uma notificação por e-mail usando o Amazon SES.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um aplicativo web que permite fazer o seguinte:

- Carregar fotos em um bucket do Amazon Simple Storage Service (Amazon S3).
- Usar o Amazon Rekognition para analisar e rotular as fotos.
- Usar o Amazon Simple Email Service (Amazon SES) para enviar relatórios de análise da imagem por e-mail.

Este exemplo contém dois componentes principais: uma página da web escrita com React e um serviço REST escrito em Python JavaScript que é construído com. Flask-RESTful

Você pode usar a página da Web do React para:

- Exibir uma lista de imagens que estão armazenadas no bucket do S3.
- Carregar imagens do computador para o bucket do S3.
- Exibir imagens e rótulos que identificam os itens detectados na imagem.
- Obter um relatório de todas as imagens no bucket do S3 e enviar um relatório por e-mail.

A página da Web chama o serviço REST. O serviço envia solicitações à AWS para realizar as seguintes ações:

- Obter e filtrar a lista de imagens no bucket do S3.
- Carregar fotos no bucket do S3.
- Usar o Amazon Rekognition para analisar fotos individuais e obter uma lista dos rótulos que identifiquem os itens detectados nas fotos.
- Analisar todas as fotos no bucket do S3 e usar o Amazon SES para enviar um relatório por e-mail.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS

- Amazon SQS

Gerar credenciais para estabelecer conexão com um endpoint SMTP do Amazon SES

O exemplo de código a seguir mostra como gerar credenciais para estabelecer conexão com um endpoint SMTP do Amazon SES.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
#!/usr/bin/env python3

import hmac
import hashlib
import base64
import argparse

SMTP_REGIONS = [
    "us-east-2",  # US East (Ohio)
    "us-east-1",  # US East (N. Virginia)
    "us-west-2",  # US West (Oregon)
    "ap-south-1",  # Asia Pacific (Mumbai)
    "ap-northeast-2",  # Asia Pacific (Seoul)
    "ap-southeast-1",  # Asia Pacific (Singapore)
    "ap-southeast-2",  # Asia Pacific (Sydney)
    "ap-northeast-1",  # Asia Pacific (Tokyo)
    "ca-central-1",  # Canada (Central)
    "eu-central-1",  # Europe (Frankfurt)
    "eu-west-1",  # Europe (Ireland)
    "eu-west-2",  # Europe (London)
    "eu-south-1",  # Europe (Milan)
    "eu-north-1",  # Europe (Stockholm)
    "sa-east-1",  # South America (Sao Paulo)
    "us-gov-west-1",  # AWS GovCloud (US)
```

```
    "us-gov-east-1",  # AWS GovCloud (US)
]

# These values are required to calculate the signature. Do not change them.
DATE = "11111111"
SERVICE = "ses"
MESSAGE = "SendRawEmail"
TERMINAL = "aws4_request"
VERSION = 0x04


def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def calculate_key(secret_access_key, region):
    if region not in SMTP_REGIONS:
        raise ValueError(f"The {region} Region doesn't have an SMTP endpoint.")

    signature = sign(("AWS4" + secret_access_key).encode("utf-8"), DATE)
    signature = sign(signature, region)
    signature = sign(signature, SERVICE)
    signature = sign(signature, TERMINAL)
    signature = sign(signature, MESSAGE)
    signature_and_version = bytes([VERSION]) + signature
    smtp_password = base64.b64encode(signature_and_version)
    return smtp_password.decode("utf-8")


def main():
    parser = argparse.ArgumentParser(
        description="Convert a Secret Access Key to an SMTP password."
    )
    parser.add_argument("secret", help="The Secret Access Key to convert.")
    parser.add_argument(
        "region",
        help="The AWS Region where the SMTP password will be used.",
        choices=SMTP_REGIONS,
    )
    args = parser.parse_args()
    print(calculate_key(args.secret, args.region))


if __name__ == "__main__":
```

```
    main()
```

Configuração de serviços de envio de e-mails

O exemplo de código a seguir mostra como:

- Verificar um endereço de e-mail.
- Verificar um domínio (opcional).
- Verificar limites de envio.
- Enviar um e-mail de teste.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SES Setup Script (v2)
# This script helps you set up Amazon SES for sending emails

# Initialize log file
LOG_FILE="ses-setup.log"
echo "Starting Amazon SES setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}
```

```
# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
    local ignore_error="${4:-false}"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        if [[ "$ignore_error" != "true" ]]; then
            cleanup_resources
            exit 1
        fi
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"

    # No physical resources to clean up for SES setup
    # Email identities can be deleted if needed
    if [[ -n "$EMAIL_ADDRESS" ]]; then
        echo "Deleting email identity: $EMAIL_ADDRESS" | tee -a "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$EMAIL_ADDRESS\""
    fi

    if [[ -n "$RECIPIENT_EMAIL" && "$RECIPIENT_EMAIL" != "$EMAIL_ADDRESS" ]];
 then
        echo "Deleting recipient email identity: $RECIPIENT_EMAIL" | tee -a
 "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$RECIPIENT_EMAIL\""
    fi

    if [[ -n "$DOMAIN_NAME" ]]; then
        echo "Deleting domain identity: $DOMAIN_NAME" | tee -a "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$DOMAIN_NAME\""
    fi
}

# Track created resources
CREATED_RESOURCES=()

# Welcome message
```

```
echo "==============================================="
echo "Amazon SES Setup Script"
echo "==============================================="
echo "This script will help you set up Amazon SES for sending emails."
echo "You'll need to verify at least one email address that you own."
echo ""
echo "NOTE: New SES accounts are placed in the sandbox environment."
echo "In the sandbox, both sender AND recipient email addresses must be
 verified."
echo ""

# Get email address to verify
echo "Please enter an email address that you own and can access:"
EMAIL_ADDRESS="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1)@example.com"
echo "$EMAIL_ADDRESS"

# Verify email identity
echo "Verifying email address: $EMAIL_ADDRESS" | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses verify-email-identity --email-address \"$EMAIL_ADDRESS
\"")
check_error "$OUTPUT" $? "Failed to verify email address"

CREATED_RESOURCES+=("Email identity: $EMAIL_ADDRESS")
echo "A verification email has been sent to $EMAIL_ADDRESS."
echo "Please check your inbox and click the verification link before continuing."
echo ""
echo "Press Enter after you've verified your email address..."
sleep 1

# Check verification status
echo "Checking verification status..." | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses list-identities --identity-type EmailAddress")
check_error "$OUTPUT" $? "Failed to list identities"

OUTPUT=$(log_cmd "aws ses get-identity-verification-attributes --identities
 \"$EMAIL_ADDRESS\"")
check_error "$OUTPUT" $? "Failed to get verification attributes"

# Check if the email is verified
VERIFICATION_STATUS=$(echo "$OUTPUT" | grep -o '"VerificationStatus": "[^"]*' |
 cut -d'"' -f4)
if [[ "$VERIFICATION_STATUS" != "Success" ]]; then
```

```
    echo "Email address $EMAIL_ADDRESS is not verified yet. Please check your
 inbox and verify before continuing."
    echo "Exiting script..."
    exit 1
fi

# Ask if user wants to verify a domain
echo ""
echo "Do you want to verify a domain for sending emails? (y/n):"
VERIFY_DOMAIN="y"
echo "$VERIFY_DOMAIN"

if [[ "$VERIFY_DOMAIN" =~ ^[Yy] ]]; then
    echo "Please enter the domain name you want to verify:"
    DOMAIN_NAME="test-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1).example.com"
    echo "$DOMAIN_NAME"

    # Verify domain identity
    echo "Verifying domain: $DOMAIN_NAME" | tee -a "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses verify-domain-identity --domain \"$DOMAIN_NAME\"")
    check_error "$OUTPUT" $? "Failed to verify domain identity"

    # Extract verification token
    VERIFICATION_TOKEN=$(echo "$OUTPUT" | grep -o '"VerificationToken": "[^"]*' |
 cut -d'"' -f4)

    CREATED_RESOURCES+=("Domain identity: $DOMAIN_NAME")

    echo ""
    echo "=============================================="
    echo "Domain Verification Instructions"
    echo "=============================================="
    echo "To verify your domain ownership, you need to add a TXT record"
    echo "to your domain's DNS settings with the following values:"
    echo ""
    echo "Record Type: TXT"
    echo "Record Name: _amazonses.$DOMAIN_NAME"
    echo "Record Value: $VERIFICATION_TOKEN"
    echo ""
    echo "After adding this DNS record, verification may take up to 72 hours."
    echo ""

    # Set up DKIM for the domain
```

```
    echo "Setting up DKIM for domain: $DOMAIN_NAME" | tee -a "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses verify-domain-dkim --domain \"$DOMAIN_NAME\"")
    check_error "$OUTPUT" $? "Failed to set up DKIM"

    # Extract DKIM tokens
    DKIM_TOKENS=$(echo "$OUTPUT" | grep -o '"DkimTokens": \[[^]]*\]' | sed
 's/"DkimTokens": \[\|\]//g' | sed 's/,//g' | sed 's/"//g')

    echo "============================================="
    echo "DKIM Configuration Instructions"
    echo "============================================="
    echo "To configure DKIM for your domain, add the following CNAME records"
    echo "to your domain's DNS settings:"
    echo ""

    for token in $DKIM_TOKENS; do
        echo "Record Type: CNAME"
        echo "Record Name: ${token}._domainkey.$DOMAIN_NAME"
        echo "Record Value: ${token}.dkim.amazonses.com"
        echo ""
    done

    echo "After adding these DNS records, DKIM verification may take up to 72
 hours."
    echo ""
fi

# Check sending limits
echo "Checking your SES sending limits..." | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses get-send-quota")
check_error "$OUTPUT" $? "Failed to get sending quota"

# Extract quota information
MAX_SEND_RATE=$(echo "$OUTPUT" | grep -o '"MaxSendRate": [0-9.]*' | cut -d' ' -
f2)
MAX_24_HOUR_SEND=$(echo "$OUTPUT" | grep -o '"Max24HourSend": [0-9.]*' | cut -d'
 ' -f2)
SENT_LAST_24_HOURS=$(echo "$OUTPUT" | grep -o '"SentLast24Hours": [0-9.]*' | cut
 -d' ' -f2)

echo ""
echo "============================================="
echo "Your SES Sending Limits"
echo "============================================="
```

```
echo "Maximum send rate: $MAX_SEND_RATE emails/second"
echo "Maximum 24-hour send: $MAX_24_HOUR_SEND emails"
echo "Sent in the last 24 hours: $SENT_LAST_24_HOURS emails"
echo ""

# Ask if user wants to send a test email
echo "Do you want to send a test email? (y/n):"
SEND_TEST="y"
echo "$SEND_TEST"

if [[ "$SEND_TEST" =~ ^[Yy] ]]; then
    echo ""
    echo "==============================================="
    echo "SANDBOX ENVIRONMENT NOTICE"
    echo "==============================================="
    echo "Your account is likely in the SES sandbox environment."
    echo "In the sandbox, you can only send emails to verified email addresses."
    echo ""
    echo "Do you want to:"
    echo "1. Send a test email to the same verified address ($EMAIL_ADDRESS)"
    echo "2. Verify another email address to use as recipient"
    echo ""
    echo "Enter your choice (1 or 2):"
    RECIPIENT_CHOICE="1"
    echo "$RECIPIENT_CHOICE"

    if [[ "$RECIPIENT_CHOICE" == "1" ]]; then
        RECIPIENT_EMAIL="$EMAIL_ADDRESS"
    else
        echo "Please enter the recipient email address you want to verify:"
        RECIPIENT_EMAIL="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 |
 head -n 1)@example.com"
        echo "$RECIPIENT_EMAIL"

        # Verify recipient email identity if different from sender
        if [[ "$RECIPIENT_EMAIL" != "$EMAIL_ADDRESS" ]]; then
            echo "Verifying recipient email address: $RECIPIENT_EMAIL" | tee -a
 "$LOG_FILE"
            OUTPUT=$(log_cmd "aws ses verify-email-identity --email-address
 \"$RECIPIENT_EMAIL\"")
            check_error "$OUTPUT" $? "Failed to verify recipient email address"

            CREATED_RESOURCES+=("Email identity: $RECIPIENT_EMAIL")
            echo "A verification email has been sent to $RECIPIENT_EMAIL."
```

```
            echo "Please check the inbox and click the verification link before
 continuing."
            echo ""
            echo "Press Enter after you've verified the recipient email
 address..."
            sleep 1

            # Check recipient verification status
            OUTPUT=$(log_cmd "aws ses get-identity-verification-attributes --
identities \"$RECIPIENT_EMAIL\"")
            check_error "$OUTPUT" $? "Failed to get recipient verification
 attributes"

            # Check if the recipient email is verified
            RECIPIENT_VERIFICATION_STATUS=$(echo "$OUTPUT" | grep -o
 '"VerificationStatus": "[^"]*' | cut -d'"' -f4)
            if [[ "$RECIPIENT_VERIFICATION_STATUS" != "Success" ]]; then
                echo "Recipient email address $RECIPIENT_EMAIL is not verified
 yet."
                echo "You can try sending the email anyway, but it may fail."
            fi
        fi
    fi

    echo "Sending test email from $EMAIL_ADDRESS to $RECIPIENT_EMAIL..." | tee -a
 "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses send-email \
        --from \"$EMAIL_ADDRESS\" \
        --destination \"ToAddresses=$RECIPIENT_EMAIL\" \
        --message \"Subject={Data=SES Test
 Email,Charset=UTF-8},Body={Text={Data=This is a test email sent from Amazon SES
 using the AWS CLI,Charset=UTF-8}}\"")

    # Don't exit on send email error, just report it
    check_error "$OUTPUT" $? "Failed to send test email" "true"

    # Check if the email was sent successfully
    if [[ "$OUTPUT" =~ "MessageId" ]]; then
        # Extract message ID
        MESSAGE_ID=$(echo "$OUTPUT" | grep -o '"MessageId": "[^"]*' | cut -d'"' -
f4)
        echo "Test email sent successfully! Message ID: $MESSAGE_ID"
    else
```

```
        echo "Failed to send test email. This is likely because your account is
 in the sandbox environment."
        echo "In the sandbox, both sender AND recipient email addresses must be
 verified."
    fi
    echo ""
fi

# Summary of created resources
echo ""
echo "==============================================="
echo "Setup Complete - Resources Created"
echo "==============================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo ""

# Ask if user wants to clean up resources
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
echo "Do you want to clean up all created resources? (y/n):"
CLEANUP_CHOICE="y"
echo "$CLEANUP_CHOICE"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
fi

echo ""
echo "==============================================="
echo "Amazon SES Setup Complete"
echo "==============================================="
echo "For production use, you may need to request to be moved out of the SES
 sandbox."
echo "Visit the SES console and navigate to 'Account dashboard' to request
 production access."
echo ""
echo "Log file: $LOG_FILE"
echo "==============================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - DeleteIdentity
  - GetIdentityVerificationAttributes
  - GetSendQuota
  - ListIdentities
  - SendEmail
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Usar Step Functions para invocar funções do Lambda

O exemplo de código a seguir mostra como criar uma máquina de AWS Step Functions estado que invoca AWS Lambda funções em sequência.

Java

SDK para Java 2.x

Mostra como criar um fluxo de trabalho AWS sem servidor usando AWS Step Functions e. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Verifique uma identidade de e-mail e envie mensagens com o Amazon SES usando um AWS SDK

O exemplo de código a seguir mostra como:

- Adicionar e verificar um endereço de e-mail com o Amazon SES.
- Enviar uma mensagem de e-mail padrão.
- Criar um modelo e envie uma mensagem de e-mail com modelo.
- Enviar uma mensagem usando um servidor SMTP do Amazon SES.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Verifique um endereço de e-mail com o Amazon SES e envie mensagens.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Simple Email Service (Amazon SES) email demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    ses_client = boto3.client("ses")
    ses_identity = SesIdentity(ses_client)
    ses_mail_sender = SesMailSender(ses_client)
    ses_template = SesTemplate(ses_client)
    email = input("Enter an email address to send mail with Amazon SES: ")
    status = ses_identity.get_identity_status(email)
    verified = status == "Success"
    if not verified:
        answer = input(
            f"The address '{email}' is not verified with Amazon SES. Unless your
 "
```

```
            f"Amazon SES account is out of sandbox, you can send mail only from "
            f"and to verified accounts. Do you want to verify this account for
use "
            f"with Amazon SES? If yes, the address will receive a verification "
            f"email (y/n): "
        )
        if answer.lower() == "y":
            ses_identity.verify_email_identity(email)
            print(f"Follow the steps in the email to {email} to complete
verification.")
            print("Waiting for verification...")
            try:
                ses_identity.wait_until_identity_exists(email)
                print(f"Identity verified for {email}.")
                verified = True
            except WaiterError:
                print(
                    f"Verification timeout exceeded. You must complete the "
                    f"steps in the email sent to {email} to verify the address."
                )

    if verified:
        test_message_text = "Hello from the Amazon SES mail demo!"
        test_message_html = "<p>Hello!</p><p>From the <b>Amazon SES</b> mail
demo!</p>"

        print(f"Sending mail from {email} to {email}.")
        ses_mail_sender.send_email(
            email,
            SesDestination([email]),
            "Amazon SES demo",
            test_message_text,
            test_message_html,
        )
        input("Mail sent. Check your inbox and press Enter to continue.")

        template = {
            "name": "doc-example-template",
            "subject": "Example of an email template.",
            "text": "This is what {{name}} will {{action}} if {{name}} can't
display "
            "HTML.",
            "html": "<p><i>This</i> is what {{name}} will {{action}} if {{name}}
"
```

```
            "<b>can</b> display HTML.</p>",
        }
        print("Creating a template and sending a templated email.")
        ses_template.create_template(**template)
        template_data = {"name": email.split("@")[0], "action": "read"}
        if ses_template.verify_tags(template_data):
            ses_mail_sender.send_templated_email(
                email, SesDestination([email]), ses_template.name(),
 template_data
            )
            input("Mail sent. Check your inbox and press Enter to continue.")

        print("Sending mail through the Amazon SES SMTP server.")
        boto3_session = boto3.Session()
        region = boto3_session.region_name
        credentials = boto3_session.get_credentials()
        port = 587
        smtp_server = f"email-smtp.{region}.amazonaws.com"
        password = calculate_key(credentials.secret_key, region)
        message = """
Subject: Hi there

This message is sent from the Amazon SES SMTP mail demo."""
        context = ssl.create_default_context()
        with smtplib.SMTP(smtp_server, port) as server:
            server.starttls(context=context)
            server.login(credentials.access_key, password)
            server.sendmail(email, email, message)
        print("Mail sent. Check your inbox!")

    if ses_template.template is not None:
        print("Deleting demo template.")
        ses_template.delete_template()
    if verified:
        answer = input(f"Do you want to remove {email} from Amazon SES (y/n)? ")
        if answer.lower() == "y":
            ses_identity.delete_identity(email)
    print("Thanks for watching!")
    print("-" * 88)
```

Crie funções para encapsular ações de identidade do Amazon SES.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
        """
        Starts verification of a domain identity. To complete verification, you
 must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
```

```
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise


    def wait_until_identity_exists(self, identity):
        """
        Waits until an identity exists. The waiter polls Amazon SES until the
        identity has been successfully verified or until it exceeds its maximum
time.

        :param identity: The identity to wait for.
        """
        try:
            waiter = self.ses_client.get_waiter("identity_exists")
            logger.info("Waiting until %s exists.", identity)
            waiter.wait(Identities=[identity])
        except WaiterError:
            logger.error("Waiting for identity %s failed or timed out.",
identity)
            raise


    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
```

```
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status


    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

Crie funções para encapsular ações com modelo do Amazon SES.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
```

```
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise


    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
```

```
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
```

```
            logger.exception("Couldn't update template %s.", name)
            raise
```

Crie funções para encapsular ações de e-mail do Amazon SES.

```
class SesDestination:
    """Contains data about an email destination."""

    def __init__(self, tos, ccs=None, bccs=None):
        """
        :param tos: The list of recipients on the 'To:' line.
        :param ccs: The list of recipients on the 'CC:' line.
        :param bccs: The list of recipients on the 'BCC:' line.
        """
        self.tos = tos
        self.ccs = ccs
        self.bccs = bccs

    def to_service_format(self):
        """
        :return: The destination data in the format expected by Amazon SES.
        """
        svc_format = {"ToAddresses": self.tos}
        if self.ccs is not None:
            svc_format["CcAddresses"] = self.ccs
        if self.bccs is not None:
            svc_format["BccAddresses"] = self.bccs
        return svc_format



class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
```

```
   def send_email(self, source, destination, subject, text, html,
reply_tos=None):
       """
       Sends an email.

       Note: If your account is in the Amazon SES  sandbox, the source and
       destination email accounts must both be verified.

       :param source: The source email account.
       :param destination: The destination email account.
       :param subject: The subject of the email.
       :param text: The plain text version of the body of the email.
       :param html: The HTML version of the body of the email.
       :param reply_tos: Email accounts that will receive a reply if the
recipient
                         replies to the message.
       :return: The ID of the message, assigned by Amazon SES.
       """
       send_args = {
           "Source": source,
           "Destination": destination.to_service_format(),
           "Message": {
               "Subject": {"Data": subject},
               "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
           },
       }
       if reply_tos is not None:
           send_args["ReplyToAddresses"] = reply_tos
       try:
           response = self.ses_client.send_email(**send_args)
           message_id = response["MessageId"]
           logger.info(
               "Sent mail %s from %s to %s.", message_id, source,
destination.tos
           )
       except ClientError:
           logger.exception(
               "Couldn't send mail from %s to %s.", source, destination.tos
           )
           raise
       else:
           return message_id
```

```
    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
passed
        in this function contains key-value pairs that define the values to
insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement
values
                              that are inserted in the template before it is
sent.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
```

```
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
            raise
        else:
            return message_id
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTemplate
  - DeleteIdentity
  - DeleteTemplate
  - GetIdentityVerificationAttributes
  - GetTemplate
  - ListIdentities
  - ListTemplates
  - SendEmail
  - SendTemplatedEmail
  - UpdateTemplate
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Exemplos de código para a API v2 do Amazon SES usando AWS SDKs

Os exemplos de código a seguir mostram como usar a API v2 do Amazon Simple Email Service com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon SES API v2: mais informações sobre o Amazon SES API v2.
- Referência do Amazon SES API v2: detalhes sobre todas as ações disponíveis do Amazon SES API v2.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos da API v2 do Amazon SES usando AWS SDKs
  - Olá, Amazon SES API v2
  - Ações para a API v2 do Amazon SES usando AWS SDKs
    - Use CreateContactcom um AWS SDK
    - Use CreateContactListcom um AWS SDK
    - Use CreateEmailIdentitycom um AWS SDK
    - Use CreateEmailTemplatecom um AWS SDK
    - Use DeleteContactListcom um AWS SDK
    - Use DeleteEmailIdentitycom um AWS SDK
    - Use DeleteEmailTemplatecom um AWS SDK
    - Use GetEmailIdentitycom um AWS SDK
    - Use ListContactListscom um AWS SDK
    - Use ListContactscom um AWS SDK
    - Use SendBulkEmailcom um AWS SDK
    - Use SendEmailcom um AWS SDK
- Cenários para a API v2 do Amazon SES usando AWS SDKs
  - Envie e-mails com anexos usando a API v2 do Amazon SES
  - Um cenário completo de boletim informativo da API v2 do Amazon SES usando um AWS SDK

Exemplos básicos da API v2 do Amazon SES usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos da API v2 do Amazon Simple Email Service com SDKs da AWS .

Exemplos

- Olá, Amazon SES API v2
- Ações para a API v2 do Amazon SES usando AWS SDKs
  - Use CreateContactcom um AWS SDK
  - Use CreateContactListcom um AWS SDK
  - Use CreateEmailIdentitycom um AWS SDK
  - Use CreateEmailTemplatecom um AWS SDK
  - Use DeleteContactListcom um AWS SDK
  - Use DeleteEmailIdentitycom um AWS SDK
  - Use DeleteEmailTemplatecom um AWS SDK
  - Use GetEmailIdentitycom um AWS SDK
  - Use ListContactListscom um AWS SDK
  - Use ListContactscom um AWS SDK
  - Use SendBulkEmailcom um AWS SDK
  - Use SendEmailcom um AWS SDK

Olá, Amazon SES API v2

O exemplo de código a seguir mostra como começar a usar a API v2 do Amazon SES.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def hello_sesv2(sesv2_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon SESv2 client and
    list the email identities in your account. This example uses the default
    settings specified in your shared credentials and config files.

    :param sesv2_client: A Boto3 SESv2 client object.
    """
    print("Hello, Amazon SESv2. Let's list up to 5 email identities:\n")

    try:
        response = sesv2_client.list_email_identities(PageSize=5)
        identities = response["EmailIdentities"]

        if not identities:
            print(
                "No email identities found. "
                "Use CreateEmailIdentity to add one."
            )
        else:
            for identity in identities:
                print(
                    f"  Identity: {identity['IdentityName']}"
                    f"  Type: {identity['IdentityType']}"
                    f"  Status: {identity['VerificationStatus']}"
                    f"  Sending: {'Enabled' if identity['SendingEnabled'] else
 'Disabled'}"
                )
            print(f"\nShowing {len(identities)} email identity(ies).")

    except ClientError as err:
        logger.error(
            "Couldn't list email identities. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a ListEmailIdentitiesReferência da API AWS SDK for Python (Boto3).

Ações para a API v2 do Amazon SES usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais da API v2 do Amazon SES com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API e a API v2 do Amazon SES e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para a API v2 do Amazon SES usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API v2 do Amazon Simple Email Service.

Exemplos

- Use CreateContactcom um AWS SDK
- Use CreateContactListcom um AWS SDK
- Use CreateEmailIdentitycom um AWS SDK
- Use CreateEmailTemplatecom um AWS SDK
- Use DeleteContactListcom um AWS SDK
- Use DeleteEmailIdentitycom um AWS SDK
- Use DeleteEmailTemplatecom um AWS SDK
- Use GetEmailIdentitycom um AWS SDK
- Use ListContactListscom um AWS SDK
- Use ListContactscom um AWS SDK
- Use SendBulkEmailcom um AWS SDK
- Use SendEmailcom um AWS SDK

Use **`CreateContact`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateContact`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Cenário de boletim informativo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a contact and adds it to the specified contact list.
    /// </summary>
    /// <param name="emailAddress">The email address of the contact.</param>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
```

```
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }
```

- Para obter detalhes da API, consulte CreateContacta Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
      try {
        // Create a new contact with the provided email address in the
        CreateContactRequest contactRequest = CreateContactRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .emailAddress(emailAddress)
            .build();

        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);

        System.out.println("Contact created: " + emailAddress);
```

```
        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }
```

- Para obter detalhes da API, consulte CreateContacta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
```

```
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
Skipping...")
                else:
                    raise e
```

- Para obter detalhes da API, consulte a CreateContactReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn add_contact(client: &Client, list: &str, email: &str) -> Result<(),
 Error> {
    client
        .create_contact()
        .contact_list_name(list)
        .email_address(email)
        .send()
        .await?;

    println!("Created contact");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateContactreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_se2->createcontact(
          iv_contactlistname = iv_contact_list_name
```

```
          iv_emailaddress = iv_email_address ).
        MESSAGE 'Contact created successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2alreadyexistsex.
        MESSAGE 'Contact already exists.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex.
        MESSAGE 'Bad request.' TYPE 'E'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE 'Contact list not found.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateContactreferência da API AWS SDK for SAP ABAP.

Use **`CreateContactList`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateContactList`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Cenário de boletim informativo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Creates a contact list with the specified name.
/// </summary>
/// <param name="contactListName">The name of the contact list.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateContactListAsync(string contactListName)
{
```

```
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact
list: {ex.Message}");
        }
        return false;
    }
```

- Para obter detalhes da API, consulte CreateContactLista Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
 CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }
```

- Para obter detalhes da API, consulte CreateContactLista Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
```

```
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e
```

- Para obter detalhes da API, consulte a CreateContactListReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_list(client: &Client, contact_list: &str) -> Result<(), Error> {
    client
        .create_contact_list()
        .contact_list_name(contact_list)
        .send()
        .await?;

    println!("Created contact list.");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateContactListreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_se2->createcontactlist(
      iv_contactlistname = iv_contact_list_name ).
    MESSAGE 'Contact list created successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2alreadyexistsex.
    MESSAGE 'Contact list already exists.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE 'Bad request - contact list limit may be reached.' TYPE 'I'.
    " Re-raise the exception so the caller can handle it
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_se2limitexceededex INTO DATA(lo_limit_exceeded).
    MESSAGE 'Limit exceeded - contact list limit reached.' TYPE 'I'.
    " Re-raise the exception so the caller can handle it
    RAISE EXCEPTION lo_limit_exceeded.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateContactListreferência da API AWS SDK for SAP ABAP.

Use **`CreateEmailIdentity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateEmailIdentity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Cenário de anexos de e-mail
- Cenário de boletim informativo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Creates an email identity (email address or domain) and starts the
verification process.
   /// </summary>
   /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
   /// <returns>The response from the CreateEmailIdentity operation.</returns>
   public async Task<CreateEmailIdentityResponse>
CreateEmailIdentityAsync(string emailIdentity)
   {
       var request = new CreateEmailIdentityRequest
       {
           EmailIdentity = emailIdentity
       };

       try
       {
           var response = await _sesClient.CreateEmailIdentityAsync(request);
           return response;
       }
       catch (AlreadyExistsException ex)
       {
           Console.WriteLine($"Email identity {emailIdentity} already exists.");
           Console.WriteLine(ex.Message);
           throw;
       }
       catch (ConcurrentModificationException ex)
       {
           Console.WriteLine($"The email identity {emailIdentity} is being
modified by another operation or thread.");
           Console.WriteLine(ex.Message);
           throw;
```

```
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been
exceeded.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
identity: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateEmailIdentitya Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   try {
     CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
         .emailIdentity(verifiedEmail)
         .build();
     sesClient.createEmailIdentity(createEmailIdentityRequest);
     System.out.println("Email identity created: " + verifiedEmail);
   } catch (AlreadyExistsException e) {
     System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
   } catch (NotFoundException e) {
     System.err.println("The provided email address is not verified: " +
verifiedEmail);
     throw e;
   } catch (LimitExceededException e) {
     System.err
         .println("You have reached the limit for email identities. Please
remove some identities and try again.");
     throw e;
   } catch (SesV2Exception e) {
     System.err.println("Error creating email identity: " + e.getMessage());
     throw e;
   }
```

- Para obter detalhes da API, consulte CreateEmailIdentitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def create_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Starts the process of verifying an email identity (email address or
 domain).

        :param email_address: The email address or domain to verify.
        :return: A dictionary with the identity type and verification status.
        :raises ClientError: If the limit is exceeded (LimitExceededException).
        """
        try:
```

```
            response = self.sesv2_client.create_email_identity(
                EmailIdentity=email_address
            )
            logger.info(
                "Started verification for email identity %s.", email_address
            )
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email identity %s. You have exceeded "
                    "the maximum number of email identities. "
                    "Use an existing verified identity.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't create email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a CreateEmailIdentityReferência da API AWS SDK for Python (Boto3).

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
```

```
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created
 successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e
```

- Para obter detalhes da API, consulte a CreateEmailIdentityReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created
successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }
```

- Para obter detalhes da API, consulte a CreateEmailIdentityreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_se2->createemailidentity(
      iv_emailidentity = iv_email_identity ).
    MESSAGE 'Email identity created successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2alreadyexistsex.
    MESSAGE 'Email identity already exists.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
  CATCH /aws1/cx_se2limitexceededex INTO DATA(lo_limit_exceeded).
    MESSAGE lo_limit_exceeded TYPE 'I' DISPLAY LIKE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateEmailIdentityreferência da API AWS SDK for SAP ABAP.

Use **`CreateEmailTemplate`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateEmailTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Cenário de anexos de e-mail
- Cenário de boletim informativo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
exists.");
            Console.WriteLine(ex.Message);
```

```
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been
 exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
 template: {ex.Message}");
        }

        return false;
    }
```

- Para obter detalhes da API, consulte CreateEmailTemplatea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
```

```
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
              .html(newsletterHtml)
              .text(newsletterText)
              .build())
          .build();

      sesClient.createEmailTemplate(templateRequest);

      System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
      // If the template already exists, skip this step and proceed with the next
      // operation
      System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
      // If the limit for email templates is exceeded, fail the workflow and
 inform
      // the user
      System.err.println("You have reached the limit for email templates. Please
 remove some templates and try again.");
      throw e;
    } catch (Exception e) {
      System.err.println("Error occurred while creating email template: " +
 e.getMessage());
      throw e;
    }
```

- Para obter detalhes da API, consulte CreateEmailTemplatea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def create_email_template(
        self,
        template_name: str,
        subject: str,
        html_body: str,
        text_body: str,
    ) -> None:
        """
        Creates an email template for use with templated and bulk email sends.
```

```
        :param template_name: The name for the new template.
        :param subject: The subject line of the template. May include
{{placeholders}}.
        :param html_body: The HTML body of the template.
        :param text_body: The plain text body of the template.
        :raises ClientError: If the template limit is exceeded
(LimitExceededException).
        """
        try:
            self.sesv2_client.create_email_template(
                TemplateName=template_name,
                TemplateContent={
                    "Subject": subject,
                    "Html": html_body,
                    "Text": text_body,
                },
            )
            logger.info("Created email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email template %s. You have exceeded "
                    "the maximum number of email templates. "
                    "Delete unused templates first.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't create email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a CreateEmailTemplateReferência da API AWS SDK for Python (Boto3).

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
```

```
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e
```

- Para obter detalhes da API, consulte a CreateEmailTemplateReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-
newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();
```

```
        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created
successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }
```

- Para obter detalhes da API, consulte a CreateEmailTemplatereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_template_content) = NEW /aws1/cl_se2emailtmplcontent(
          iv_subject = iv_subject
          iv_html = iv_html
          iv_text = iv_text ).
```

```
        lo_se2->createemailtemplate(
          iv_templatename = iv_template_name
          io_templatecontent = lo_template_content ).
        MESSAGE 'Email template created successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2alreadyexistsex.
        MESSAGE 'Email template already exists.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex.
        MESSAGE 'Bad request.' TYPE 'E'.
      CATCH /aws1/cx_se2limitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateEmailTemplatereferência da API AWS SDK for SAP ABAP.

Use **`DeleteContactList`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteContactList`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Cenário de boletim informativo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being
modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact
list: {ex.Message}");
        }

        return false;
    }
```

- Para obter detalhes da API, consulte DeleteContactLista Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // Delete the contact list
      DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      sesClient.deleteContactList(deleteContactListRequest);

      System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
    } catch (NotFoundException e) {
      // If the contact list does not exist, log the error and proceed
      System.out.println("Contact list not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }
```

- Para obter detalhes da API, consulte DeleteContactLista Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
```

```
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)
```

- Para obter detalhes da API, consulte a DeleteContactListReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted
successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }
```

- Para obter detalhes da API, consulte a DeleteContactListreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_se2->deletecontactlist(
      iv_contactlistname = iv_contact_list_name ).
    MESSAGE 'Contact list deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2notfoundexception.
    MESSAGE 'Contact list not found.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE 'Bad request.' TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteContactListreferência da API AWS SDK for SAP ABAP.

Use **`DeleteEmailIdentity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteEmailIdentity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Cenário de anexos de e-mail
- Cenário de boletim informativo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
```

```
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
identity: {ex.Message}");
        }

        return false;
    }
```

- Para obter detalhes da API, consulte DeleteEmailIdentitya Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
      try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
```

```
        System.err.println("Error occurred while deleting the email identity: " +
 e.getMessage());
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }
```

- Para obter detalhes da API, consulte DeleteEmailIdentitya Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
```

```
        return cls(sesv2_client)


    def delete_email_identity(self, email_address: str) -> None:
        """
        Deletes an email identity.

        :param email_address: The email address or domain to delete.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Deleted email identity %s.", email_address)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found or already deleted.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't delete email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteEmailIdentityReferência da API AWS SDK for Python (Boto3).

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:

 self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
 successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not
 exist.")
                else:
                    print(e)
```

- Para obter detalhes da API, consulte a DeleteEmailIdentityReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }
```

- Para obter detalhes da API, consulte a DeleteEmailIdentityreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_se2->deleteemailidentity(
          iv_emailidentity = iv_email_identity ).
        MESSAGE 'Email identity deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE 'Email identity not found.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteEmailIdentityreferência da API AWS SDK for SAP ABAP.

Use **`DeleteEmailTemplate`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteEmailTemplate`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Cenário de anexos de e-mail
- Cenário de boletim informativo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</
param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
template: {ex.Message}");
        }

        return false;
    }
```

- Para obter detalhes da API, consulte DeleteEmailTemplatea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
      e.printStackTrace();
    }
```

- Para obter detalhes da API, consulte DeleteEmailTemplatea Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def delete_email_template(self, template_name: str) -> None:
        """
        Deletes an email template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If the template is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_template(
                TemplateName=template_name
```

```
            )
            logger.info("Deleted email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email template %s not found or already deleted.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't delete email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteEmailTemplateReferência da API AWS SDK for Python (Boto3).

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
```

```
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

- Para obter detalhes da API, consulte a DeleteEmailTemplateReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        match self
            .client
```

```
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted
successfully.")?,
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- Para obter detalhes da API, consulte a DeleteEmailTemplatereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_se2->deleteemailtemplate(
          iv_templatename = iv_template_name ).
        MESSAGE 'Email template deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE 'Email template not found.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteEmailTemplatereferência da API AWS SDK for SAP ABAP.

Use **`GetEmailIdentity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetEmailIdentity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Cenário de anexos de e-mail

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)
```

```
    def get_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Gets information about an email identity, including its verification
status.

        :param email_address: The email address or domain to look up.
        :return: A dictionary with identity information including verification
status.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            response = self.sesv2_client.get_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Got email identity for %s.", email_address)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found.", email_address
                )
            else:
                logger.error(
                    "Couldn't get email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a GetEmailIdentityReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Determina se um endereço de e-mail foi verificado.

```
async fn is_verified(client: &Client, email: &str) -> Result<(), Error> {
    let resp = client
        .get_email_identity()
        .email_identity(email)
        .send()
        .await?;

    if resp.verified_for_sending_status() {
        println!("The address is verified");
    } else {
        println!("The address is not verified");
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetEmailIdentityreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_se2->getemailidentity(
          iv_emailidentity = iv_email_identity ).
        MESSAGE |Identity type: { oo_result->get_identitytype( ) }, | &&
                |verified for sending: { oo_result-
>get_verifiedforsendingstatus( ) }| TYPE 'I'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE |Email identity { iv_email_identity } not found.| TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetEmailIdentityreferência da API AWS SDK for SAP ABAP.

Use **`ListContactLists`**com um AWS SDK

O código de exemplo a seguir mostra como usar `ListContactLists`.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_lists(client: &Client) -> Result<(), Error> {
    let resp = client.list_contact_lists().send().await?;

    println!("Contact lists:");

    for list in resp.contact_lists() {
        println!("  {}", list.contact_list_name().unwrap_or_default());
    }

    Ok(())
```

```
}
```

- Para obter detalhes da API, consulte a ListContactListsreferência da API AWS SDK for Rust.

Use **`ListContacts`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListContacts`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Cenário de boletim informativo

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.ListContactsAsync(request);
```

```
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
{ex.Message}");
        }

        return new List<Contact>();
    }
```

- Para obter detalhes da API, consulte ListContactsa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    ListContactsRequest contactListRequest = ListContactsRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .build();

    List<String> contactEmails;
    try {
```

```
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }
```

- Para obter detalhes da API, consulte ListContactsa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()
```

```
class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e
```

- Para obter detalhes da API, consulte a ListContactsReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_contacts(client: &Client, list: &str) -> Result<(), Error> {
    let resp = client
        .list_contacts()
```

```
        .contact_list_name(list)
        .send()
        .await?;

    println!("Contacts:");

    for contact in resp.contacts() {
        println!("  {}", contact.email_address().unwrap_or_default());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListContactsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_se2->listcontacts(
          iv_contactlistname = iv_contact_list_name ).
        DATA(lv_count) = lines( oo_result->get_contacts( ) ).
        MESSAGE |Retrieved { lv_count } contacts from list.| TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2notfoundexception INTO DATA(lo_not_found).
        MESSAGE 'Contact list not found.' TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListContactsreferência da API AWS SDK for SAP ABAP.

Use **SendBulkEmail**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `SendBulkEmail`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Cenário de anexos de e-mail

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
```

```
        return cls(sesv2_client)


    def send_bulk_email(
        self,
        from_address: str,
        template_name: str,
        default_template_data: str,
        bulk_entries: List[Dict[str, Any]],
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Sends a templated email to multiple recipients in a single API call.

        All recipients receive the same attachment(s) defined in the default
        content, while template data can be personalized per recipient.

        :param from_address: The verified sender email address.
        :param template_name: The name of an existing email template.
        :param default_template_data: Default JSON template data string.
        :param bulk_entries: A list of BulkEmailEntry dicts, each containing
            'Destination' and optionally 'ReplacementEmailContent'.
        :param attachments: An optional list of attachment dicts for all
            recipients.
        :return: A list of BulkEmailEntryResult dicts with status and MessageId.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            template_content: Dict[str, Any] = {
                "TemplateName": template_name,
                "TemplateData": default_template_data,
            }

            if attachments:
                template_content["Attachments"] = attachments

            response = self.sesv2_client.send_bulk_email(
                FromEmailAddress=from_address,
                DefaultContent={"Template": template_content},
                BulkEmailEntries=bulk_entries,
            )
            results = response.get("BulkEmailEntryResults", [])
            logger.info(
                "Sent bulk email from %s to %d recipients.",
```

```
                from_address,
                len(bulk_entries),
            )
            return results
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Bulk message was rejected. Check that the template "
                    "exists, attachment file types are supported, and "
                    "total message size is within limits. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send bulk email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a SendBulkEmailReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Build the default template content used for all bulk recipients
    DATA(lo_template) = NEW /aws1/cl_se2template(
      iv_templatename = iv_template_name
      iv_templatedata = iv_template_data ).
```

```
        DATA(lo_default_content) = NEW /aws1/cl_se2bulkemailcontent(
          io_template = lo_template ).

        DATA(lo_result) = lo_se2->sendbulkemail(
          iv_fromemailaddress = iv_from_address
          io_defaultcontent   = lo_default_content
          it_bulkemailentries = it_bulk_entries ).

        ot_results = lo_result->get_bulkemailentryresults( ).
        MESSAGE |Bulk email sent to { lines( it_bulk_entries ) } recipient(s).|
TYPE 'I'.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_rejected).
        MESSAGE lo_rejected TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_rejected.
      CATCH /aws1/cx_se2mailfrmdomnotver00 INTO DATA(lo_not_verified).
        MESSAGE lo_not_verified TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_not_verified.
      CATCH /aws1/cx_se2notfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendBulkEmailreferência da API AWS SDK for SAP ABAP.

Use **`SendEmail`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `SendEmail`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Cenário de anexos de e-mail

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Sends an email with the specified content and options.
   /// </summary>
   /// <param name="fromEmailAddress">The email address to send the email
from.</param>
   /// <param name="toEmailAddresses">The email addresses to send the email
to.</param>
   /// <param name="subject">The subject of the email.</param>
   /// <param name="htmlContent">The HTML content of the email.</param>
   /// <param name="textContent">The text content of the email.</param>
   /// <param name="templateName">The name of the email template to use
(optional).</param>
   /// <param name="templateData">The data to replace placeholders in the email
template (optional).</param>
   /// <param name="contactListName">The name of the contact list for
unsubscribe functionality (optional).</param>
   /// <returns>The MessageId response from the SendEmail operation.</returns>
   public async Task<string> SendEmailAsync(string fromEmailAddress,
List<string> toEmailAddresses, string? subject,
       string? htmlContent, string? textContent, string? templateName = null,
string? templateData = null, string? contactListName = null)
   {
       var request = new SendEmailRequest
       {
           FromEmailAddress = fromEmailAddress
       };

       if (toEmailAddresses.Any())
       {
           request.Destination = new Destination { ToAddresses =
toEmailAddresses };
       }
```

```
        if (!string.IsNullOrEmpty(templateName))
        {
            request.Content = new EmailContent()
            {
                Template = new Template
                {
                    TemplateName = templateName,
                    TemplateData = templateData
                }
            };
        }
        else
        {
            request.Content = new EmailContent
            {
                Simple = new Message
                {
                    Subject = new Content { Data = subject },
                    Body = new Body
                    {
                        Html = new Content { Data = htmlContent },
                        Text = new Content { Data = textContent }
                    }
                }
            };
        }

        if (!string.IsNullOrEmpty(contactListName))
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
```

```
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
{ex.Message}");
        }

        return string.Empty;
    }
```

- Para obter detalhes da API, consulte SendEmaila Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = """

                                Usage:
                                    <sender> <recipient> <subject>\s

                                Where:
                                    sender - An email address that represents the
 sender.\s
```

```
                                recipient - An email address that represents the
 recipient.\s
                                subject - The subject line.\s
                            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</
h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        send(sesv2Client, sender, recipient, subject, bodyHTML);
    }

    public static void send(SesV2Client client,
                            String sender,
                            String recipient,
                            String subject,
                            String bodyHTML) {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();
```

```
        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        EmailContent emailContent = EmailContent.builder()
                .simple(msg)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES "
                    + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Envia uma mensagem usando um modelo.

```
      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
```

```
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
     }
```

Envia uma mensagem com informações do cabeçalho.

```
public class SendwithHeader {

    public static void main(String[] args) {
        final String usage = """

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
```

```
                .build();

        String bodyHTML = """
                <html>
                    <head></head>
                    <body>
                        <h1>Hello!</h1>
                        <p>See the list of customers.</p>
                    </body>
                </html>
                """;

        sendWithHeader(sesv2Client, sender, recipient, subject, bodyHTML);
        sesv2Client.close();
    }

    /**
     * Sends an email using the AWS SES V2 client.
     *
     * @param sesv2Client the SES V2 client to use for sending the email
     * @param sender the email address of the sender
     * @param recipient the email address of the recipient
     * @param subject the subject of the email
     * @param bodyHTML the HTML content of the email body
     */
    public static void sendWithHeader(SesV2Client sesv2Client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyHTML) {
        EmailContent emailContent = EmailContent.builder()
                .simple(Message.builder()
                        .body(b -> b.html(c ->
 c.charset(UTF_8.name()).data(bodyHTML))
                                .text(c ->
 c.charset(UTF_8.name()).data(bodyHTML)))
                        .subject(c -> c.charset(UTF_8.name()).data(subject))
                        .headers(List.of(
                                MessageHeader.builder()
                                        .name("List-Unsubscribe")
                                        .value("<https://nutrition.co/?
address=x&topic=x>, <mailto:unsubscribe@nutrition.co?subject=TopicUnsubscribe>")
                                        .build(),
                                MessageHeader.builder()
```

```
                                        .name("List-Unsubscribe-Post")
                                        .value("List-Unsubscribe=One-Click")
                                        .build()))
                        .build())
                .build();

        SendEmailRequest request = SendEmailRequest.builder()
                .fromEmailAddress(sender)
                .destination(d -> d.toAddresses(recipient))
                .content(emailContent)
                .build();

        try {
            SendEmailResponse response = sesv2Client.sendEmail(request);
            System.out.println("Email sent! Message ID: " +
 response.messageId());
        } catch (SesV2Exception e) {
            System.err.println("Failed to send email: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte SendEmaila Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem com anexos opcionais.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""
```

```
    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def send_email(
        self,
        from_address: str,
        to_addresses: List[str],
        subject: str,
        html_body: str,
        text_body: str,
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> str:
        """
        Sends a simple email message with optional attachments.

        SES handles MIME construction automatically when using attachments
        with the Simple content type, so developers don't need to build
        raw MIME messages.

        :param from_address: The verified sender email address.
        :param to_addresses: A list of recipient email addresses.
        :param subject: The subject line of the email.
        :param html_body: The HTML body content.
        :param text_body: The plain text body content.
        :param attachments: An optional list of attachment dictionaries. Each
            attachment should contain 'RawContent' (bytes), 'FileName' (str),
            and optionally 'ContentType', 'ContentDisposition', 'ContentId',
            'ContentDescription', and 'ContentTransferEncoding'.
```

```
        :return: The MessageId of the sent email.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            simple_message: Dict[str, Any] = {
                "Subject": {"Data": subject},
                "Body": {
                    "Html": {"Data": html_body},
                    "Text": {"Data": text_body},
                },
            }

            if attachments:
                simple_message["Attachments"] = attachments

            response = self.sesv2_client.send_email(
                FromEmailAddress=from_address,
                Destination={"ToAddresses": to_addresses},
                Content={"Simple": simple_message},
            )
            message_id = response["MessageId"]
            logger.info(
                "Sent email from %s to %s. MessageId: %s",
                from_address,
                to_addresses,
                message_id,
            )
            return message_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Message was rejected. Check that attachments use "
                    "supported file types and total message size is "
                    "under 40 MB. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a SendEmailReferência da API AWS SDK for Python (Boto3).

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem a todos os membros da lista de contatos.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep
```

```
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
 Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
```

Envia uma mensagem a todos os membros da lista de contatos usando um modelo.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """
```

```
    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )
```

- Para obter detalhes da API, consulte a SendEmailReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sesv2'
require_relative 'config' # Recipient and sender email addresses.

# Set up the SESv2 client.
client = Aws::SESV2::Client.new(region: AWS_REGION)

def send_email(client, sender_email, recipient_email)
  response = client.send_email(
    {
```

```
      from_email_address: sender_email,
      destination: {
        to_addresses: [recipient_email]
      },
      content: {
        simple: {
          subject: {
            data: 'Test email subject'
          },
          body: {
            text: {
              data: 'Test email body'
            }
          }
        }
      }
    }
  )
  puts "Email sent from #{SENDER_EMAIL} to #{RECIPIENT_EMAIL} with message ID:
 #{response.message_id}"
end

send_email(client, SENDER_EMAIL, RECIPIENT_EMAIL)
```

- Para obter detalhes da API, consulte SendEmaila Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem a todos os membros da lista de contatos.

```
async fn send_message(
    client: &Client,
    list: &str,
```

```
    from: &str,
    subject: &str,
    message: &str,
) -> Result<(), Error> {
    // Get list of email addresses from contact list.
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    let contacts = resp.contacts();

    let cs: Vec<String> = contacts
        .iter()
        .map(|i| i.email_address().unwrap_or_default().to_string())
        .collect();

    let mut dest: Destination = Destination::builder().build();
    dest.to_addresses = Some(cs);
    let subject_content = Content::builder()
        .data(subject)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body_content = Content::builder()
        .data(message)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body = Body::builder().text(body_content).build();

    let msg = Message::builder()
        .subject(subject_content)
        .body(body)
        .build();

    let email_content = EmailContent::builder().simple(msg).build();

    client
        .send_email()
        .from_email_address(from)
        .destination(dest)
        .content(email_content)
```

```
        .send()
        .await?;

    println!("Email sent to list");

    Ok(())
}
```

Envia uma mensagem a todos os membros da lista de contatos usando um modelo.

```
            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

 .destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
```

```
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}:
 {}", email, e)),
            }
```

- Para obter detalhes da API, consulte a SendEmailreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem.

```
    TRY.
        " Create destination with recipient address
        DATA lt_to_addresses TYPE /aws1/
cl_se2emailaddresslist_w=>tt_emailaddresslist.
        APPEND NEW /aws1/cl_se2emailaddresslist_w( iv_value =
 iv_to_email_address ) TO lt_to_addresses.
        DATA(lo_destination) = NEW /aws1/cl_se2destination(
          it_toaddresses = lt_to_addresses ).

        " Create message content
        DATA(lo_subject) = NEW /aws1/cl_se2content( iv_data = iv_subject ).
        DATA(lo_text_body) = NEW /aws1/cl_se2content( iv_data = iv_text_body ).
        DATA(lo_html_body) = NEW /aws1/cl_se2content( iv_data = iv_html_body ).
        DATA(lo_body) = NEW /aws1/cl_se2body(
          io_text = lo_text_body
          io_html = lo_html_body ).
        DATA(lo_message) = NEW /aws1/cl_se2message(
```

```
          io_subject = lo_subject
          io_body = lo_body ).

        DATA(lo_content) = NEW /aws1/cl_se2emailcontent(
          io_simple = lo_message ).

        " Send the email
        lo_se2->sendemail(
          iv_fromemailaddress = iv_from_email_address
          io_destination = lo_destination
          io_content = lo_content ).
        MESSAGE 'Email sent successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2accountsuspendedex INTO DATA(lo_account_suspended).
        MESSAGE 'Account suspended.' TYPE 'I'.
        RAISE EXCEPTION lo_account_suspended.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_message_rejected).
        MESSAGE 'Message rejected - check email verification.' TYPE 'I'.
        RAISE EXCEPTION lo_message_rejected.
    ENDTRY.
```

Envia uma mensagem usando um modelo.

```
    TRY.
        " Create destination with recipient address
        DATA lt_to_addresses TYPE /aws1/
cl_se2emailaddresslist_w=>tt_emailaddresslist.
        APPEND NEW /aws1/cl_se2emailaddresslist_w( iv_value =
 iv_to_email_address ) TO lt_to_addresses.
        DATA(lo_destination) = NEW /aws1/cl_se2destination(
          it_toaddresses = lt_to_addresses ).

        " Create template reference
        DATA(lo_template) = NEW /aws1/cl_se2template(
          iv_templatename = iv_template_name
          iv_templatedata = iv_template_data ).

        DATA(lo_content) = NEW /aws1/cl_se2emailcontent(
          io_template = lo_template ).
```

```
        " Create list management options
        DATA(lo_list_mgmt) = NEW /aws1/cl_se2listmanagementopts(
          iv_contactlistname = iv_contact_list_name ).

        " Send the email using template
        lo_se2->sendemail(
          iv_fromemailaddress = iv_from_email_address
          io_destination = lo_destination
          io_content = lo_content
          io_listmanagementoptions = lo_list_mgmt ).
        MESSAGE 'Email sent using template successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2accountsuspendedex INTO DATA(lo_account_suspended).
        MESSAGE 'Account suspended.' TYPE 'I'.
        RAISE EXCEPTION lo_account_suspended.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_message_rejected).
        MESSAGE 'Message rejected - check email verification.' TYPE 'I'.
        RAISE EXCEPTION lo_message_rejected.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendEmailreferência da API AWS SDK for SAP ABAP.

Cenários para a API v2 do Amazon SES usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns na API v2 do Amazon SES com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro da API v2 do Amazon SES ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Envie e-mails com anexos usando a API v2 do Amazon SES
- Um cenário completo de boletim informativo da API v2 do Amazon SES usando um AWS SDK

Envie e-mails com anexos usando a API v2 do Amazon SES

O exemplo de código a seguir mostra como enviar e-mails com anexos usando a API v2 do Amazon SES.

- Crie um modelo de e-mail para envios em massa.
- Envie um e-mail simples com um anexo de arquivo.
- Envie um e-mail simples com uma imagem embutida.
- Envie e-mails modelados em massa com anexos.
- Limpe recursos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando anexos de e-mail.

```
class SESv2EmailAttachmentsScenario:
    """
    Demonstrates sending emails with attachments using Amazon SESv2.

    This scenario demonstrates:
    1. Setting up an email identity and template.
    2. Sending a simple email with a file attachment.
    3. Sending a simple email with an inline image.
    4. Sending bulk templated emails with attachments.
    5. Cleaning up created resources.
    """

    TEMPLATE_NAME = "AttachmentDemoTemplate"

    def __init__(self, sesv2_wrapper: SESv2Wrapper) -> None:
        """
        :param sesv2_wrapper: An instance of the SESv2Wrapper class.
```

```
        """
        self.sesv2_wrapper = sesv2_wrapper
        self.sender_email = ""
        self.recipient_emails: list = []
        self.identity_was_created = False

    def run_scenario(self) -> None:
        """Runs the SESv2 email attachments scenario."""
        print("-" * 88)
        print("Welcome to the Amazon SESv2 Email Attachments Scenario.")
        print("-" * 88)
        print(
            "This scenario demonstrates how to send emails with attachments\n"
            "using SESv2 attachment support. SES handles MIME\n"
            "construction automatically, so you don't need to build raw\n"
            "MIME messages.\n"
        )

        try:
            self._setup()
            self._step1_send_email_with_attachment()
            self._step2_send_email_with_inline_image()
            self._step3_send_bulk_email_with_attachments()
        except Exception as e:
            logger.error("Scenario failed: %s", e)
            print(f"\nThe scenario encountered an error: {e}")
        finally:
            self._cleanup()

    # ---------- Setup ----------

    def _setup(self) -> None:
        """
        Prompts for configuration, verifies the sender identity, prepares a
        sample attachment, and creates an email template.
        """
        print("\n--- Setup ---\n")

        # Prompt for sender and recipient addresses.
        print(
            "Both sender and recipient addresses must be verified if your\n"
            "account is in the SES sandbox.\n"
        )
        self.sender_email = q.ask(
```

```
            "Enter a verified sender email address: "
        )
        recipient_input = q.ask(
            "Enter one or more recipient email addresses (comma-separated): "
        )
        self.recipient_emails = [
            addr.strip() for addr in recipient_input.split(",") if addr.strip()
        ]

        # Verify the sender identity.
        print(f"\nChecking identity for {self.sender_email}...")
        try:
            identity_info = self.sesv2_wrapper.get_email_identity(
                self.sender_email
            )
            verified = identity_info.get("VerifiedForSendingStatus", False)
            if verified:
                print(f"  {self.sender_email} is verified and ready to send.")
            else:
                print(
                    f"  {self.sender_email} exists but is not yet verified."
                )
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                print(
                    f"  Identity {self.sender_email} not found. "
                    "Creating it now..."
                )
                result = self.sesv2_wrapper.create_email_identity(
                    self.sender_email
                )
                self.identity_was_created = True
                print(
                    f"  Identity created. Verification status: "
                    f"{result.get('VerifiedForSendingStatus', False)}"
                )
                print(
                    "  Check your inbox and click the verification link "
                    "before continuing."
                )
                q.ask("Press Enter when you have verified the address...")
            else:
                raise
```

```
        # Create the email template for the bulk-send step.
        print("\nCreating email template for the bulk email step...")
        try:
            self.sesv2_wrapper.create_email_template(
                template_name=self.TEMPLATE_NAME,
                subject="Bulk Email with Attachment for {{name}}",
                html_body=(
                    "<h1>Hello {{name}}</h1>"
                    "<p>Please find the attached document.</p>"
                ),
                text_body=(
                    "Hello {{name}}, Please find the attached document."
                ),
            )
            print(f"  Template '{self.TEMPLATE_NAME}' created.\n")
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                print(
                    f"  Template '{self.TEMPLATE_NAME}' already exists. "
                    "Using it.\n"
                )
            else:
                raise

    # ---------- Step 1: Simple email with file attachment ----------

    def _step1_send_email_with_attachment(self) -> None:
        """Sends a simple email with a text file attachment."""
        print("\n--- Step 1: Send a Simple Email with a File Attachment ---\n")
        print(
            "Creating a sample text file attachment and sending it with\n"
            "the Simple email content type. SES constructs the MIME message\n"
            "automatically.\n"
        )

        # Prepare a sample text file as bytes.
        sample_content = b"This is a sample report attachment."

        attachment = {
            "RawContent": sample_content,
            "FileName": "sample-report.txt",
            "ContentType": "text/plain",
            "ContentDisposition": "ATTACHMENT",
            "ContentDescription": "Sample report text file",
```

```
            "ContentTransferEncoding": "BASE64",
        }

        print(
            "Note: When using an AWS SDK, the SDK handles base64 encoding\n"
            "automatically. Direct API callers must encode content themselves.\n"
        )

        message_id = self.sesv2_wrapper.send_email(
            from_address=self.sender_email,
            to_addresses=self.recipient_emails,
            subject="SESv2 Attachment Demo - Simple Email with Attachment",
            html_body=(
                "<h1>Attachment Demo</h1>"
                "<p>Please see the attached <b>report document</b>.</p>"
            ),
            text_body="Please see the attached report document.",
            attachments=[attachment],
        )

        print(f"  Email sent. MessageId: {message_id}")
        print(
            "  SES automatically constructed the MIME message with the "
            "attachment.\n"
        )

    # ---------- Step 2: Simple email with inline image ----------

    def _step2_send_email_with_inline_image(self) -> None:
        """Sends a simple email with an inline image that renders in HTML."""
        print("\n--- Step 2: Send a Simple Email with an Inline Image ---\n")
        print(
            "This step demonstrates INLINE disposition. The image renders\n"
            "directly in the HTML body using a 'cid:' reference instead of\n"
            "appearing as a downloadable attachment.\n"
        )

        # Create a minimal 1x1 red PNG (valid PNG file).
        sample_image = (
            b"\x89PNG\r\n\x1a\n"  # PNG signature
            b"\x00\x00\x00\rIHDR\x00\x00\x00\x01\x00\x00\x00\x01"
            b"\x08\x02\x00\x00\x00\x90wS\xde"  # 1x1 RGB
            b"\x00\x00\x00\x0cIDATx\x9cc\xf8\x0f\x00\x00\x01\x01"
            b"\x00\x05\x18\xd8N"  # compressed data
```

```
            b"\x00\x00\x00\x00IEND\xaeB`\x82"  # IEND
        )

        attachment = {
            "RawContent": sample_image,
            "FileName": "logo.png",
            "ContentType": "image/png",
            "ContentDisposition": "INLINE",
            "ContentId": "logo123",
            "ContentDescription": "Company logo",
            "ContentTransferEncoding": "BASE64",
        }

        html_body = (
            "<html><body>"
            "<h1>Inline Image Demo</h1>"
            "<p>Here is our logo:</p>"
            '<img src="cid:logo123" alt="Company Logo">'
            "</body></html>"
        )

        message_id = self.sesv2_wrapper.send_email(
            from_address=self.sender_email,
            to_addresses=self.recipient_emails,
            subject="SESv2 Attachment Demo - Inline Image",
            html_body=html_body,
            text_body=(
                "Inline Image Demo - Please view this email in an "
                "HTML-capable client to see the embedded image."
            ),
            attachments=[attachment],
        )

        print(f"  Email sent. MessageId: {message_id}")
        print(
            "  The ContentId 'logo123' is referenced in the HTML body via\n"
            "  'cid:logo123', which lets the image render inline.\n"
        )

    # ---------- Step 3: Bulk templated email with attachments ----------

    def _step3_send_bulk_email_with_attachments(self) -> None:
        """Sends bulk templated emails with attachments to multiple
recipients."""
```

```
        print("\n--- Step 3: Send Bulk Templated Emails with Attachments ---\n")
        print(
            "Using SendBulkEmail to send a templated email with an attachment\n"
            "to multiple recipients in a single API call. Each recipient gets\n"
            "personalized content via template data.\n"
        )

        sample_content = b"This is a sample report attachment."

        attachment = {
            "RawContent": sample_content,
            "FileName": "sample-report.txt",
            "ContentType": "text/plain",
            "ContentDisposition": "ATTACHMENT",
            "ContentDescription": "Sample report for bulk recipients",
            "ContentTransferEncoding": "BASE64",
        }

        # Build one entry per recipient with personalized names.
        names = ["Alice", "Bob", "Charlie", "Diana", "Eve"]
        bulk_entries = []
        for i, email in enumerate(self.recipient_emails):
            name = names[i] if i < len(names) else f"Recipient{i + 1}"
            bulk_entries.append(
                {
                    "Destination": {"ToAddresses": [email]},
                    "ReplacementEmailContent": {
                        "ReplacementTemplate": {
                            "ReplacementTemplateData": json.dumps(
                                {"name": name}
                            )
                        }
                    },
                }
            )

        results = self.sesv2_wrapper.send_bulk_email(
            from_address=self.sender_email,
            template_name=self.TEMPLATE_NAME,
            default_template_data='{"name": "Valued Customer"}',
            bulk_entries=bulk_entries,
            attachments=[attachment],
        )
```

```
        print("  Bulk email results:")
        for idx, result in enumerate(results):
            status = result.get("Status", "Unknown")
            msg_id = result.get("MessageId", "N/A")
            error = result.get("Error", "")
            recipient = (
                self.recipient_emails[idx]
                if idx < len(self.recipient_emails)
                else "Unknown"
            )
            print(f"    {recipient}: Status={status}, MessageId={msg_id}")
            if error:
                print(f"      Error: {error}")

        print(
            "\n  All recipients receive the same attachment(s) defined in\n"
            "  DefaultContent. Template data is personalized per recipient.\n"
        )

    # ---------- Cleanup ----------

    def _cleanup(self) -> None:
        """Deletes the email template and optionally the email identity."""
        print("\n--- Cleanup ---\n")

        # Delete the email template.
        try:
            self.sesv2_wrapper.delete_email_template(self.TEMPLATE_NAME)
            print(f"  Template '{self.TEMPLATE_NAME}' deleted.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                print(
                    f"  Template '{self.TEMPLATE_NAME}' was already deleted."
                )
            else:
                logger.error("Failed to delete template: %s", err)

        # Optionally delete the email identity.
        if self.identity_was_created and self.sender_email:
            delete_identity = q.ask(
                f"Delete the email identity '{self.sender_email}'? (y/n) ",
                q.is_yesno,
            )
            if delete_identity:
```

```
                try:
                    self.sesv2_wrapper.delete_email_identity(
                        self.sender_email
                    )
                    print(
                        f"  Email identity '{self.sender_email}' deleted."
                    )
                except ClientError as err:
                    if err.response["Error"]["Code"] == "NotFoundException":
                        print(
                            f"  Identity '{self.sender_email}' was "
                            "already deleted."
                        )
                    else:
                        logger.error(
                            "Failed to delete identity: %s", err
                        )
            else:
                print(
                    f"  Skipping identity deletion for {self.sender_email}."
                )
        else:
            print(
                "  Sender identity was pre-existing. Skipping deletion."
            )

        print("\nAll resources have been cleaned up.")
        print("-" * 88)
```

Crie uma classe de wrapper SESv2 para gerenciar as operações.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
```

```
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def get_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Gets information about an email identity, including its verification
status.

        :param email_address: The email address or domain to look up.
        :return: A dictionary with identity information including verification
status.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            response = self.sesv2_client.get_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Got email identity for %s.", email_address)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found.", email_address
                )
            else:
                logger.error(
                    "Couldn't get email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
    def create_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Starts the process of verifying an email identity (email address or
domain).

        :param email_address: The email address or domain to verify.
        :return: A dictionary with the identity type and verification status.
        :raises ClientError: If the limit is exceeded (LimitExceededException).
        """
        try:
            response = self.sesv2_client.create_email_identity(
                EmailIdentity=email_address
            )
            logger.info(
                "Started verification for email identity %s.", email_address
            )
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email identity %s. You have exceeded "
                    "the maximum number of email identities. "
                    "Use an existing verified identity.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't create email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def create_email_template(
        self,
        template_name: str,
        subject: str,
        html_body: str,
        text_body: str,
    ) -> None:
        """
        Creates an email template for use with templated and bulk email sends.
```

```
        :param template_name: The name for the new template.
        :param subject: The subject line of the template. May include
{{placeholders}}.
        :param html_body: The HTML body of the template.
        :param text_body: The plain text body of the template.
        :raises ClientError: If the template limit is exceeded
(LimitExceededException).
        """
        try:
            self.sesv2_client.create_email_template(
                TemplateName=template_name,
                TemplateContent={
                    "Subject": subject,
                    "Html": html_body,
                    "Text": text_body,
                },
            )
            logger.info("Created email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email template %s. You have exceeded "
                    "the maximum number of email templates. "
                    "Delete unused templates first.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't create email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def send_email(
        self,
        from_address: str,
        to_addresses: List[str],
        subject: str,
        html_body: str,
        text_body: str,
```

```
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> str:
        """
        Sends a simple email message with optional attachments.

        SES handles MIME construction automatically when using attachments
        with the Simple content type, so developers don't need to build
        raw MIME messages.

        :param from_address: The verified sender email address.
        :param to_addresses: A list of recipient email addresses.
        :param subject: The subject line of the email.
        :param html_body: The HTML body content.
        :param text_body: The plain text body content.
        :param attachments: An optional list of attachment dictionaries. Each
            attachment should contain 'RawContent' (bytes), 'FileName' (str),
            and optionally 'ContentType', 'ContentDisposition', 'ContentId',
            'ContentDescription', and 'ContentTransferEncoding'.
        :return: The MessageId of the sent email.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            simple_message: Dict[str, Any] = {
                "Subject": {"Data": subject},
                "Body": {
                    "Html": {"Data": html_body},
                    "Text": {"Data": text_body},
                },
            }

            if attachments:
                simple_message["Attachments"] = attachments

            response = self.sesv2_client.send_email(
                FromEmailAddress=from_address,
                Destination={"ToAddresses": to_addresses},
                Content={"Simple": simple_message},
            )
            message_id = response["MessageId"]
            logger.info(
                "Sent email from %s to %s. MessageId: %s",
                from_address,
                to_addresses,
                message_id,
```

```
            )
            return message_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Message was rejected. Check that attachments use "
                    "supported file types and total message size is "
                    "under 40 MB. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def send_bulk_email(
        self,
        from_address: str,
        template_name: str,
        default_template_data: str,
        bulk_entries: List[Dict[str, Any]],
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Sends a templated email to multiple recipients in a single API call.

        All recipients receive the same attachment(s) defined in the default
        content, while template data can be personalized per recipient.

        :param from_address: The verified sender email address.
        :param template_name: The name of an existing email template.
        :param default_template_data: Default JSON template data string.
        :param bulk_entries: A list of BulkEmailEntry dicts, each containing
            'Destination' and optionally 'ReplacementEmailContent'.
        :param attachments: An optional list of attachment dicts for all
            recipients.
        :return: A list of BulkEmailEntryResult dicts with status and MessageId.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
```

```
            template_content: Dict[str, Any] = {
                "TemplateName": template_name,
                "TemplateData": default_template_data,
            }

            if attachments:
                template_content["Attachments"] = attachments

            response = self.sesv2_client.send_bulk_email(
                FromEmailAddress=from_address,
                DefaultContent={"Template": template_content},
                BulkEmailEntries=bulk_entries,
            )
            results = response.get("BulkEmailEntryResults", [])
            logger.info(
                "Sent bulk email from %s to %d recipients.",
                from_address,
                len(bulk_entries),
            )
            return results
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Bulk message was rejected. Check that the template "
                    "exists, attachment file types are supported, and "
                    "total message size is within limits. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send bulk email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def delete_email_template(self, template_name: str) -> None:
        """
        Deletes an email template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If the template is not found (NotFoundException).
        """
```

```
        try:
            self.sesv2_client.delete_email_template(
                TemplateName=template_name
            )
            logger.info("Deleted email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email template %s not found or already deleted.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't delete email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def delete_email_identity(self, email_address: str) -> None:
        """
        Deletes an email identity.

        :param email_address: The email address or domain to delete.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Deleted email identity %s.", email_address)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found or already deleted.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't delete email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - GetEmailIdentity
  - SendBulkEmail
  - SendEmail

Um cenário completo de boletim informativo da API v2 do Amazon SES usando um AWS SDK

Os exemplos de código a seguir mostram como executar o cenário de boletim informativo da API v2 do Amazon SES.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
```

```
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace Sesv2Scenario;

public static class NewsletterWorkflow
{
    /*
      This scenario demonstrates how to use the Amazon Simple Email Service (SES)
 v2 to send a coupon newsletter to a list of subscribers.
      The scenario performs the following tasks:

      1. Prepare the application:
         - Create a verified email identity for sending and replying to emails.
         - Create a contact list to store the subscribers' email addresses.
         - Create an email template for the coupon newsletter.

      2. Gather subscriber email addresses:
         - Prompt the user for a base email address.
         - Create 3 variants of the email address using subaddress extensions
 (e.g., user+ses-weekly-newsletter-1@example.com).
         - Add each variant as a contact to the contact list.
         - Send a welcome email to each new contact.

      3. Send the coupon newsletter:
         - Retrieve the list of contacts from the contact list.
         - Send the coupon newsletter using the email template to each contact.

      4. Monitor and review:
         - Provide instructions for the user to review the sending activity and
 metrics in the AWS console.

      5. Clean up resources:
         - Delete the contact list (which also deletes all contacts within it).
         - Delete the email template.
         - Optionally delete the verified email identity.

    */
```

```
    public static SESv2Wrapper _sesv2Wrapper;
    public static string? _baseEmailAddress = null;
    public static string? _verifiedEmail = null;
    private static string _contactListName = "weekly-coupons-newsletter";
    private static string _templateName = "weekly-coupons";
    private static string _subject = "Weekly Coupons Newsletter";
    private static string _htmlContentFile = "coupon-newsletter.html";
    private static string _textContentFile = "coupon-newsletter.txt";
    private static string _htmlWelcomeFile = "welcome.html";
    private static string _textWelcomeFile = "welcome.txt";
    private static string _couponsDataFile = "sample_coupons.json";

    // Relative location of the resources folder.
    private static string _resourcesFilePathLocation = "../../../../resources/";

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSimpleEmailServiceV2>()
                    .AddTransient<SESv2Wrapper>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon SES v2 Coupon Newsletter
Scenario.");
            Console.WriteLine("This scenario demonstrates how to use the Amazon
Simple Email Service (SES) v2 " +
                              "\r\nto send a coupon newsletter to a list of
subscribers.");
```

```
            // Prepare the application.
            var emailIdentity = await PrepareApplication();

            // Gather subscriber email addresses.
            await GatherSubscriberEmailAddresses(emailIdentity);

            // Send the coupon newsletter.
            await SendCouponNewsletter(emailIdentity);

            // Monitor and review.
            MonitorAndReview(true);

            // Clean up resources.
            await Cleanup(emailIdentity, true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon SES v2 Coupon Newsletter scenario is
complete.");
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sesv2Wrapper = host.Services.GetRequiredService<SESv2Wrapper>();
    }

    /// <summary>
    /// Set up the resources for the scenario.
    /// </summary>
    /// <returns>The email address of the verified identity.</returns>
    public static async Task<string?> PrepareApplication()
    {
        var htmlContent = await File.ReadAllTextAsync(_resourcesFilePathLocation
+ _htmlContentFile);
```

```
        var textContent = await File.ReadAllTextAsync(_resourcesFilePathLocation
+ _textContentFile);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("1. In this step, we will prepare the application:" +
                          "\r\n  - Create a verified email identity for sending
and replying to emails." +
                          "\r\n  - Create a contact list to store the
subscribers' email addresses." +
                          "\r\n  - Create an email template for the coupon
newsletter.\r\n");

        // Prompt the user for a verified email address.
        while (!IsEmail(_verifiedEmail))
        {
            Console.Write("Enter a verified email address or an email to verify:
");
            _verifiedEmail = Console.ReadLine();
        }

        try
        {
            // Create an email identity and start the verification process.
            await _sesv2Wrapper.CreateEmailIdentityAsync(_verifiedEmail);
            Console.WriteLine($"Identity {_verifiedEmail} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Identity {_verifiedEmail} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email identity: {ex.Message}");
        }

        // Create a contact list.
        try
        {
            await _sesv2Wrapper.CreateContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} created.");
        }
        catch (AlreadyExistsException)
        {
```

```
            Console.WriteLine($"Contact list {_contactListName} already
exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating contact list: {ex.Message}");
        }

        // Create an email template.
        try
        {
            await _sesv2Wrapper.CreateEmailTemplateAsync(_templateName, _subject,
htmlContent, textContent);
            Console.WriteLine($"Email template {_templateName} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Email template {_templateName} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email template: {ex.Message}");
        }

        return _verifiedEmail;
    }

    /// <summary>
    /// Generate subscriber addresses and send welcome emails.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> GatherSubscriberEmailAddresses(string
fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("2. In Step 2, we will gather subscriber email
addresses:" +
                          "\r\n  - Prompt the user for a base email address." +
                          "\r\n  - Create 3 variants of the email address using
subaddress extensions (e.g., user+ses-weekly-newsletter-1@example.com)." +
                          "\r\n  - Add each variant as a contact to the contact
list." +
```

```
                            "\r\n  - Send a welcome email to each new contact.\r
\n");

        // Prompt the user for a base email address.
        while (!IsEmail(_baseEmailAddress))
        {
            Console.Write("Enter a base email address (e.g., user@example.com):
 ");
            _baseEmailAddress = Console.ReadLine();
        }

        // Create 3 variants of the email address using +ses-weekly-newsletter-1,
 +ses-weekly-newsletter-2, etc.
        var baseEmailAddressParts = _baseEmailAddress!.Split("@");
        for (int i = 1; i <= 3; i++)
        {
            string emailAddress = $"{baseEmailAddressParts[0]}+ses-weekly-
newsletter-{i}@{baseEmailAddressParts[1]}";

            try
            {
                // Create a contact with the email address in the contact list.
                await _sesv2Wrapper.CreateContactAsync(emailAddress,
 _contactListName);
                Console.WriteLine($"Contact {emailAddress} added to the
 {_contactListName} contact list.");
            }
            catch (AlreadyExistsException)
            {
                Console.WriteLine($"Contact {emailAddress} already exists in the
 {_contactListName} contact list.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error creating contact {emailAddress}:
 {ex.Message}");
                return false;
            }

            // Send a welcome email to the new contact.
            try
            {
                string subject = "Welcome to the Weekly Coupons Newsletter";
```

```
                string htmlContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _htmlWelcomeFile);
                string textContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _textWelcomeFile);

                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress, new
List<string> { emailAddress }, subject, htmlContent, textContent);
                Console.WriteLine($"Welcome email sent to {emailAddress}.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending welcome email to
{emailAddress}: {ex.Message}");
                return false;
            }

            // Wait 2 seconds before sending the next email (if the account is in
the SES Sandbox).
            await Task.Delay(2000);
        }

        return true;
    }

    /// <summary>
    ///  Send the coupon newsletter to the subscribers in the contact list.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> SendCouponNewsletter(string fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("3. In this step, we will send the coupon newsletter:"
+
                          "\r\n  - Retrieve the list of contacts from the contact
list." +
                          "\r\n  - Send the coupon newsletter using the email
template to each contact.\r\n");


        // Retrieve the list of contacts from the contact list.
        var contacts = await _sesv2Wrapper.ListContactsAsync(_contactListName);
        if (!contacts.Any())
```

```
        {
            Console.WriteLine($"No contacts found in the {_contactListName}
contact list.");
            return false;
        }

        // Load the coupon data from the sample_coupons.json file.
        string couponsData = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _couponsDataFile);

        // Send the coupon newsletter to each contact using the email template.
        try
        {
            foreach (var contact in contacts)
            {
                // To use the Contact List for list management, send to only one
address at a time.
                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress,
                    new List<string> { contact.EmailAddress },
                    null, null, null, _templateName, couponsData,
_contactListName);
            }

            Console.WriteLine($"Coupon newsletter sent to contact list
{_contactListName}.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error sending coupon newsletter to contact list
{_contactListName}: {ex.Message}");
            return false;
        }

        return true;
    }

    /// <summary>
    /// Provide instructions for monitoring sending activity and metrics.
    /// </summary>
    /// <param name="interactive">True to run in interactive mode.</param>
    /// <returns>True if successful.</returns>
    public static bool MonitorAndReview(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("4. In step 4, we will monitor and review:" +
                          "\r\n  - Provide instructions for the user to review
the sending activity and metrics in the AWS console.\r\n");

        Console.WriteLine("Review your sending activity using the SES Homepage in
the AWS console.");
        Console.WriteLine("Press Enter to open the SES Homepage in your default
browser...");
        if (interactive)
        {
            Console.ReadLine();
            try
            {
                // Open the SES Homepage in the default browser.
                Process.Start(new ProcessStartInfo
                {
                    FileName = "https://console.aws.amazon.com/ses/home",
                    UseShellExecute = true
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error opening the SES Homepage:
{ex.Message}");
                return false;
            }
        }

        Console.WriteLine("Review the sending activity and email metrics, then
press Enter to continue...");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Clean up the resources used in the scenario.
    /// </summary>
    /// <param name="verifiedEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <param name="interactive">True if interactive.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Cleanup(string verifiedEmailAddress, bool
interactive)
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("5. Finally, we clean up resources:" +
                          "\r\n  - Delete the contact list (which also deletes
 all contacts within it)." +
                          "\r\n  - Delete the email template." +
                          "\r\n  - Optionally delete the verified email identity.
\r\n");

        Console.WriteLine("Cleaning up resources...");

        // Delete the contact list (this also deletes all contacts in the list).
        try
        {
            await _sesv2Wrapper.DeleteContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} deleted.");
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Contact list {_contactListName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting contact list {_contactListName}:
 {ex.Message}");
            return false;
        }

        // Delete the email template.
        try
        {
            await _sesv2Wrapper.DeleteEmailTemplateAsync(_templateName);
            Console.WriteLine($"Email template {_templateName} deleted.");
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Email template {_templateName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting email template {_templateName}:
 {ex.Message}");
            return false;
        }
```

```
        // Ask the user if they want to delete the email identity.
        var deleteIdentity = !interactive ||
            GetYesNoResponse(
                $"Do you want to delete the email identity
 {verifiedEmailAddress}? (y/n) ");
        if (deleteIdentity)
        {
            try
            {
                await
 _sesv2Wrapper.DeleteEmailIdentityAsync(verifiedEmailAddress);
                Console.WriteLine($"Email identity {verifiedEmailAddress}
 deleted.");
            }
            catch (NotFoundException)
            {
                Console.WriteLine(
                    $"Email identity {verifiedEmailAddress} not found.");
            }
            catch (Exception ex)
            {
                Console.WriteLine(
                    $"Error deleting email identity {verifiedEmailAddress}:
 {ex.Message}");
                return false;
            }
        }
        else
        {
            Console.WriteLine(
                $"Skipping deletion of email identity {verifiedEmailAddress}.");
        }

        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
```

```
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Simple check to verify a string is an email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email.</returns>
    private static bool IsEmail(string? email)
    {
        if (string.IsNullOrEmpty(email))
            return false;
        return Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+\.[^@\s]+$",
 RegexOptions.IgnoreCase);
    }
}
```

Wrapper para operações de serviço.

```
using System.Net;
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;

namespace Sesv2Scenario;

/// <summary>
/// Wrapper class for Amazon Simple Email Service (SES) v2 operations.
/// </summary>
public class SESv2Wrapper
{

    private readonly IAmazonSimpleEmailServiceV2 _sesClient;

    /// <summary>
    /// Constructor for the SESv2Wrapper.
    /// </summary>
    /// <param name="sesClient">The injected SES v2 client.</param>
```

```
    public SESv2Wrapper(IAmazonSimpleEmailServiceV2 sesClient)
    {
        _sesClient = sesClient;
    }

    /// <summary>
    /// Creates a contact and adds it to the specified contact list.
    /// </summary>
    /// <param name="emailAddress">The email address of the contact.</param>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
 contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
 exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"An error occurred while creating the contact
list: {ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates an email identity (email address or domain) and starts the
verification process.
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
    /// <returns>The response from the CreateEmailIdentity operation.</returns>
    public async Task<CreateEmailIdentityResponse>
CreateEmailIdentityAsync(string emailIdentity)
    {
        var request = new CreateEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.CreateEmailIdentityAsync(request);
            return response;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email identity {emailIdentity} already exists.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
modified by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been
exceeded.");
```

```
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
identity: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
```

```
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
 exists.");
            Console.WriteLine(ex.Message);
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been
 exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
 template: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
```

```
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact
 list: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
```

```
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
 identity: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
```

```
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
 template: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };
```

```
        try
        {
            var response = await _sesClient.ListContactsAsync(request);
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
{ex.Message}");
        }

        return new List<Contact>();
    }

    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email
from.</param>
    /// <param name="toEmailAddresses">The email addresses to send the email
to.</param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
(optional).</param>
    /// <param name="templateData">The data to replace placeholders in the email
template (optional).</param>
    /// <param name="contactListName">The name of the contact list for
unsubscribe functionality (optional).</param>
    /// <returns>The MessageId response from the SendEmail operation.</returns>
```

```
   public async Task<string> SendEmailAsync(string fromEmailAddress,
List<string> toEmailAddresses, string? subject,
       string? htmlContent, string? textContent, string? templateName = null,
string? templateData = null, string? contactListName = null)
   {
       var request = new SendEmailRequest
       {
           FromEmailAddress = fromEmailAddress
       };

       if (toEmailAddresses.Any())
       {
           request.Destination = new Destination { ToAddresses =
toEmailAddresses };
       }

       if (!string.IsNullOrEmpty(templateName))
       {
           request.Content = new EmailContent()
           {
               Template = new Template
               {
                   TemplateName = templateName,
                   TemplateData = templateData
               }
           };
       }
       else
       {
           request.Content = new EmailContent
           {
               Simple = new Message
               {
                   Subject = new Content { Data = subject },
                   Body = new Body
                   {
                       Html = new Content { Data = htmlContent },
                       Text = new Content { Data = textContent }
                   }
               }
           };
       }

       if (!string.IsNullOrEmpty(contactListName))
```

```
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
{ex.Message}");
```

```
        }

        return string.Empty;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
```

```
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }

      try {
        // Create a new contact with the provided email address in the
        CreateContactRequest contactRequest = CreateContactRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .emailAddress(emailAddress)
            .build();

        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);

        System.out.println("Contact created: " + emailAddress);

        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
```

```
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }

      ListContactsRequest contactListRequest = ListContactsRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      List<String> contactEmails;
      try {
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }


      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
```

```
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
      }

    try {
      CreateEmailIdentityRequest createEmailIdentityRequest =
 CreateEmailIdentityRequest.builder()
          .emailIdentity(verifiedEmail)
          .build();
      sesClient.createEmailIdentity(createEmailIdentityRequest);
      System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
      System.out.println("Email identity already exists, skipping creation: " +
 verifiedEmail);
    } catch (NotFoundException e) {
      System.err.println("The provided email address is not verified: " +
 verifiedEmail);
      throw e;
    } catch (LimitExceededException e) {
      System.err
          .println("You have reached the limit for email identities. Please
 remove some identities and try again.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating email identity: " + e.getMessage());
      throw e;
    }

    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");
```

```
     CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
         .templateName(TEMPLATE_NAME)
         .templateContent(EmailTemplateContent.builder()
             .subject("Weekly Coupons Newsletter")
             .html(newsletterHtml)
             .text(newsletterText)
             .build())
         .build();

     sesClient.createEmailTemplate(templateRequest);

     System.out.println("Email template created: " + TEMPLATE_NAME);
   } catch (AlreadyExistsException e) {
     // If the template already exists, skip this step and proceed with the next
     // operation
     System.out.println("Email template already exists, skipping creation...");
   } catch (LimitExceededException e) {
     // If the limit for email templates is exceeded, fail the workflow and
inform
     // the user
     System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
     throw e;
   } catch (Exception e) {
     System.err.println("Error occurred while creating email template: " +
e.getMessage());
     throw e;
   }

   try {
     // Delete the contact list
     DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
         .contactListName(CONTACT_LIST_NAME)
         .build();

     sesClient.deleteContactList(deleteContactListRequest);

     System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
   } catch (NotFoundException e) {
     // If the contact list does not exist, log the error and proceed
     System.out.println("Contact list not found. Skipping deletion...");
```

```
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }

      try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }

    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
```

```
        e.printStackTrace();
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
```

```
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e

            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
```

```
                                "Data": "Welcome to the Weekly Coupons
Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
Skipping...")
                else:
                    raise e

        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
```

```
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )

        try:

self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created
successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e

        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
```

```
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e

        try:

self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)

            try:

self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not
exist.")
                else:
                    print(e)

        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateContact

- CreateContactList
- CreateEmailIdentity
- CreateEmailTemplate
- DeleteContactList
- DeleteEmailIdentity
- DeleteEmailTemplate
- ListContacts
- SendEmail.simple
- SendEmail.template

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        match self
            .client
            .create_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list created
successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateContactListError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Contact list already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating contact list: {}", e)),
            },
```

```
        }

            match self
                .client
                .create_contact()
                .contact_list_name(CONTACT_LIST_NAME)
                .email_address(email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Contact created for {}", email)?,
                Err(e) => match e.into_service_error() {
                    CreateContactError::AlreadyExistsException(_) => writeln!(
                        self.stdout,
                        "Contact already exists for {}, skipping creation.",
                        email
                    )?,
                    e => return Err(anyhow!("Error creating contact for {}: {}",
 email, e)),
                },
            }

        let contacts: Vec<Contact> = match self
            .client
            .list_contacts()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(list_contacts_output) => {
                list_contacts_output.contacts.unwrap().into_iter().collect()
            }
            Err(e) => {
                return Err(anyhow!(
                    "Error retrieving contact list {}: {}",
                    CONTACT_LIST_NAME,
                    e
                ))
            }
        };

            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
```

```
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

.destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}:
{}", email, e)),
            }

        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
```

```
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created
 successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }

        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-
newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();

        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created
 successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
```

```
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }

        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted
successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }

            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }

        match self
            .client
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted
successfully.")?,
```

```
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Exemplos de código para o Amazon SNS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Simple Notification Service com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon SNS: mais informações sobre o Amazon SNS.
- Referência da API Amazon SNS: detalhes sobre todas as ações disponíveis do Amazon SNS.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon SNS AWS SDKs
  - Olá, Amazon SNS
  - Ações para o Amazon SNS usando AWS SDKs
    - Use CheckIfPhoneNumberIsOptedOutcom um AWS SDK ou CLI
    - Use ConfirmSubscriptioncom um AWS SDK ou CLI
    - Use CreateTopiccom um AWS SDK ou CLI
    - Use DeleteTopiccom um AWS SDK ou CLI
    - Use getSmsAttributes com um AWS SDK ou CLI
    - Use GetTopicAttributescom um AWS SDK ou CLI
    - Use ListPhoneNumbersOptedOutcom um AWS SDK ou CLI
    - Use ListSubscriptionscom um AWS SDK ou CLI
    - Use ListTopicscom um AWS SDK ou CLI
    - Use o Publish com um AWS SDK ou CLI
    - Use setSMsAttributes com um AWS SDK ou CLI
    - Use SetSubscriptionAttributescom um AWS SDK ou CLI
    - Use SetSubscriptionAttributesRedrivePolicycom um AWS SDK
    - Use SetTopicAttributescom um AWS SDK ou CLI
    - Use Inscrever-se com um AWS SDK ou CLI
    - Use TagResourcecom um AWS SDK ou CLI
    - Use Cancelar assinatura com um AWS SDK ou CLI
- Cenários para o Amazon SNS usando AWS SDKs
  - Criar uma aplicação para enviar dados para uma tabela do DynamoDB
  - Criar uma aplicação de publicação e assinatura que traduz mensagens
  - Criar um tópico e publicar mensagens
  - Crie um endpoint de plataforma para notificações push do Amazon SNS usando um AWS SDK

- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Criar uma aplicação de exploração do Amazon Textract
- Crie e publique em um tópico FIFO do Amazon SNS usando um AWS SDK
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Publique mensagens SMS em um tópico do Amazon SNS usando um AWS SDK
- Publique uma mensagem grande no Amazon SNS com o Amazon S3 usando um AWS SDK
- Publique uma mensagem de texto SMS do Amazon SNS usando um AWS SDK
- Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um AWS SDK
- Usar o API Gateway para invocar uma função do Lambda
- Usar eventos programados para chamar uma função do Lambda

- Exemplos de tecnologia sem servidor para o Amazon SNS
  - Invocar uma função do Lambda em um acionador do Amazon SNS

Exemplos básicos para o uso do Amazon SNS AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Simple Notification Service com SDKs da AWS .

Exemplos

- Olá, Amazon SNS
- Ações para o Amazon SNS usando AWS SDKs
  - Use CheckIfPhoneNumberIsOptedOutcom um AWS SDK ou CLI
  - Use ConfirmSubscriptioncom um AWS SDK ou CLI
  - Use CreateTopiccom um AWS SDK ou CLI
  - Use DeleteTopiccom um AWS SDK ou CLI
  - Use getSmsAttributes com um AWS SDK ou CLI
  - Use GetTopicAttributescom um AWS SDK ou CLI
  - Use ListPhoneNumbersOptedOutcom um AWS SDK ou CLI
  - Use ListSubscriptionscom um AWS SDK ou CLI

- Use ListTopicscom um AWS SDK ou CLI
- Use o Publish com um AWS SDK ou CLI
- Use setSMsAttributes com um AWS SDK ou CLI
- Use SetSubscriptionAttributescom um AWS SDK ou CLI
- Use SetSubscriptionAttributesRedrivePolicycom um AWS SDK
- Use SetTopicAttributescom um AWS SDK ou CLI
- Use Inscrever-se com um AWS SDK ou CLI
- Use TagResourcecom um AWS SDK ou CLI
- Use Cancelar assinatura com um AWS SDK ou CLI

Olá, Amazon SNS

Os exemplos de código a seguir mostram como começar a usar o Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
```

```
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)
```

```
# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_sns.cpp.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
```

```
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
```

```
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main
```

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
```

```
 }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicialize um cliente SNS e liste tópicos em sua conta.

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});

  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
    }
  }
```

```
  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your
 account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
```

```
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for Kotlin.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O Package.swift arquivo.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
```

```
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

O programa principal do Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)
```

```
        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa referência da API AWS SDK for Swift.

Ações para o Amazon SNS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon SNS com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon SNS e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon SNS usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Simple Notification Service.

Exemplos

- Use CheckIfPhoneNumberIsOptedOutcom um AWS SDK ou CLI
- Use ConfirmSubscriptioncom um AWS SDK ou CLI
- Use CreateTopiccom um AWS SDK ou CLI
- Use DeleteTopiccom um AWS SDK ou CLI
- Use getSmsAttributes com um AWS SDK ou CLI
- Use GetTopicAttributescom um AWS SDK ou CLI
- Use ListPhoneNumbersOptedOutcom um AWS SDK ou CLI
- Use ListSubscriptionscom um AWS SDK ou CLI
- Use ListTopicscom um AWS SDK ou CLI
- Use o Publish com um AWS SDK ou CLI
- Use setSMsAttributes com um AWS SDK ou CLI
- Use SetSubscriptionAttributescom um AWS SDK ou CLI
- Use SetSubscriptionAttributesRedrivePolicycom um AWS SDK

- Use SetTopicAttributescom um AWS SDK ou CLI
- Use Inscrever-se com um AWS SDK ou CLI
- Use TagResourcecom um AWS SDK ou CLI
- Use Cancelar assinatura com um AWS SDK ou CLI

Use **`CheckIfPhoneNumberIsOptedOut`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CheckIfPhoneNumberIsOptedOut`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use the Amazon Simple Notification Service
/// (Amazon SNS) to check whether a phone number has been opted out.
/// </summary>
public class IsPhoneNumOptedOut
{
    public static async Task Main()
    {
        string phoneNumber = "+15551112222";

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        await CheckIfOptedOutAsync(client, phoneNumber);
    }
```

```
        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para verificar o cancelamento de mensagens SMS para um número de telefone

O `check-if-phone-number-is-opted-out` exemplo a seguir verifica se o número de telefone especificado optou por não receber mensagens SMS da AWS conta atual.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Saída:

```
{
    "isOptedOut": false
}
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOutem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
```

```
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";
```

```
export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOuta Referência AWS SDK para PHP da API.

Use **`ConfirmSubscription`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ConfirmSubscription`.

CLI

AWS CLI

Para confirmar uma assinatura

O comando `confirm-subscription` a seguir conclui o processo de confirmação iniciado quando você se inscreveu em um tópico do SNS chamado `my-topic`. O parâmetro --token vem da mensagem de confirmação enviada ao endpoint de notificação especificado na chamada de assinatura.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

Saída:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Para obter detalhes da API, consulte ConfirmSubscriptionem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
```

```
                    .topicArn(topicArn)
                    .build();

            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ConfirmSubscriptiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
      // If this is true, the subscriber cannot unsubscribe while
 unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ConfirmSubscriptiona Referência AWS SDK para JavaScript da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
```

```
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte ConfirmSubscriptiona Referência AWS SDK para PHP da API.

Use **`CreateTopic`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateTopic`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar um tópico e publicar mensagens
- Criar e publicar em um tópico FIFO
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico com um nome específico.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

Crie um tópico com um nome e atributos específicos de FIFO e desduplicação.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }
```

```
        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
```

```
                    << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                    outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar um tópico do SNS

O exemplo `create-topic` a seguir cria um tópico do SNS chamado `my-topic`.

```
aws sns create-topic \
    --name my-topic
```

Saída:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

Para obter mais informações, consulte Usando a interface de linha de AWS comando com o Amazon SQS e o Amazon SNS no Guia do usuário AWS da interface de linha de comando.

- Para obter detalhes da API, consulte CreateTopicem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
```

```
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
```

```
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
```

```
}
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
```

```
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
 deduplication for FIFO topics.
```

```
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} -
{e}")
            raise
```

- Para obter detalhes da API, consulte a CreateTopicReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# This class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = 'YourTopicName' # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
```

```
  unless sns_topic_creator.create_topic(topic_name)
    puts 'The topic was not created. Stopping program.'
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte CreateTopica Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
is returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTopicreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(name: name)
```

```
        )

        guard let arn = output.topicArn else {
            print("No topic ARN returned by Amazon SNS.")
            return
        }
```

- Para obter detalhes da API, consulte CreateTopica referência da API AWS SDK for Swift.

Use **`DeleteTopic`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteTopic`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar um tópico e publicar mensagens
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um tópico por meio do respectivo ARN.

```
    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
```

```
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
```

```
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir um tópico do SNS

O exemplo `delete-topic` a seguir exclui o tópico do SNS especificado.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Este comando não produz saída.

- Para obter detalhes da API, consulte DeleteTopicem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();
```

```
            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
```

```
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }
```

```
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTopicreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
```

```
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte DeleteTopica Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
```

```
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
```

```
            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise
```

- Para obter detalhes da API, consulte a DeleteTopicReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
    MESSAGE 'SNS topic deleted.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTopicreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: arn)
        )
```

- Para obter detalhes da API, consulte DeleteTopica referência da API AWS SDK for Swift.

Use **`getSmsAttributes`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetSMSAttributes`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve the default settings for sending SMS messages from your AWS account
 by using
//! Amazon Simple Notification Service (Amazon SNS).
```

```
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Consulte detalhes da API em GetSMSAttributes na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Para listar os atributos padrão da mensagem SMS

O exemplo `get-sms-attributes` a seguir lista os atributos padrão para o envio de mensagens SMS.

```
aws sns get-sms-attributes
```

Saída:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Consulte detalhes da API em GetSMSAttributes na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
```

```
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- Consulte detalhes da API em GetSMSAttributes na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
    // the DefaultSMSType is undefined. For this example, it was set to
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em GetSMSAttributes na Referência da API AWS SDK para JavaScript .

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Consulte detalhes da API em GetSMSAttributes na Referência da API AWS SDK para PHP .

Use **`GetTopicAttributes`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetTopicAttributes`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
west-2:000000000000:ExampleSNSTopic";
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
```

```
        /// Given the ARN of the Amazon SNS topic, this method retrieves the
topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
        public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para recuperar os atributos de um tópico

O exemplo `get-topic-attributes` a seguir exibe os atributos do tópico especificado.

```
aws sns get-topic-attributes \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Saída:

```
{
    "Attributes": {
        "SubscriptionsConfirmed": "1",
        "DisplayName": "my-topic",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy
\":{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,
\"numMaxDelayRetries\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0,
\"backoffFunction\":\"linear\"},\"disableSubscriptionOverrides\":false}}",
        "Owner": "123456789012",
        "Policy": "{\"Version\":\"2008-10-17\",\"Id\":\"__default_policy_ID
\",\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":
\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",\"SNS:DeleteTopic\",\"SNS:GetTopicAttributes
\",\"SNS:Publish\",\"SNS:RemovePermission\",\"SNS:AddPermission\",
\"SNS:SetTopicAttributes\"],\"Resource\":\"arn:aws:sns:us-west-2:123456789012:my-
topic\",\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":
\"0123456789012\"}}}]}",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
        "SubscriptionsPending": "0"
    }
}
```

- Para obter detalhes da API, consulte GetTopicAttributesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Getting attributes for a topic with name: " +
 topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String
 topicArn) {
        try {
            GetTopicAttributesRequest request =
 GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
 snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
 result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { GetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDela
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
```

```
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Para obter detalhes da API, consulte a GetTopicAttributesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte GetTopicAttributesa Referência AWS SDK para PHP da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
       DATA(lt_attributes) = oo_result->get_attributes( ).
       MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetTopicAttributesreferência da API AWS SDK for SAP ABAP.

Use **`ListPhoneNumbersOptedOut`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Para listar as opções de cancelamento de mensagens SMS

O exemplo `list-phone-numbers-opted-out` a seguir lista os números de telefone que optaram por não receber mensagens SMS.

```
aws sns list-phone-numbers-opted-out
```

Saída:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Para obter detalhes da API, consulte ListPhoneNumbersOptedOutem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListPhoneNumbersOptedOuta Referência AWS SDK for Java 2.x da API.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte ListPhoneNumbersOptedOuta Referência AWS SDK para PHP da API.

Use **`ListSubscriptions`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListSubscriptions`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example will retrieve a list of the existing Amazon Simple
    /// Notification Service (Amazon SNS) subscriptions.
    /// </summary>
    public class ListSubscriptions
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            Console.WriteLine("Enter a topic ARN to list subscriptions for a
 specific topic, " +
                              "or press Enter to list subscriptions for all
 topics.");
            var topicArn = Console.ReadLine();
            Console.WriteLine();

            var subscriptions = await GetSubscriptionsListAsync(client,
 topicArn);

            DisplaySubscriptionList(subscriptions);
        }

        /// <summary>
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults
 to null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string
 topicArn = null)
```

```
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
                }
            }
            else
            {
                var paginateAllSubscriptions =
client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }

            return results;
        }

        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
```

```
                Console.WriteLine($"Subscription ARN:
 {subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;
```

```
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
                      << outcome.GetError().GetMessage()
                      <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar suas assinaturas do SNS

O `list-subscriptions` exemplo a seguir exibe uma lista das assinaturas do SNS em sua conta. AWS

```
aws sns list-subscriptions
```

Saída:

```
{
    "Subscriptions": [
        {
            "Owner": "123456789012",
            "Endpoint": "my-email@example.com",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
            "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
        }
    ]
}
```

- Para obter detalhes da API, consulte ListSubscriptionsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result =
 snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
```

```
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listSNSSubscriptions() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte a ListSubscriptionsreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are
 returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
```

```
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- Para obter detalhes da API, consulte a ListSubscriptionsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# This class demonstrates how to list subscriptions to an Amazon Simple
 Notification Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
```

```
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = 'SNS_TOPIC_ARN' # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte ListSubscriptionsa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sns->listsubscriptions( ).                " oo_result is
returned for testing purposes. "
        DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
```

```
      MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
    CATCH /aws1/cx_rt_generic.
      MESSAGE 'Unable to list subscribers.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListSubscriptionsreferência da API AWS SDK for SAP ABAP.

Use **`ListTopics`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTopics`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos da proteção de dispositivos da Internet das Coisas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// Lists the Amazon Simple Notification Service (Amazon SNS)
/// topics for the current account.
/// </summary>
public class ListSNSTopics
{
    public static async Task Main()
```

```
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await GetTopicListAsync(client);
        }

        /// <summary>
        /// Retrieves the list of Amazon SNS topics in groups of up to 100
        /// topics.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the list of topics.</param>
        public static async Task
GetTopicListAsync(IAmazonSimpleNotificationService client)
        {
            // If there are more than 100 Amazon SNS topics, the call to
            // ListTopicsAsync will return a value to pass to the
            // method to retrieve the next 100 (or less) topics.
            string nextToken = string.Empty;

            do
            {
                var response = await client.ListTopicsAsync(nextToken);
                DisplayTopicsList(response.Topics);
                nextToken = response.NextToken;
            }
            while (!string.IsNullOrEmpty(nextToken));
        }

        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
```

```
        else {
            std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage() <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Listar os tópicos do SNS

O `list-topics` exemplo a seguir lista todos os tópicos do SNS em sua AWS conta.

```
aws sns list-topics
```

Saída:

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Para obter detalhes da API, consulte ListTopicsem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { ListTopicsCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listSNSTopics() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
```

```
 * Returns a list of the requester's topics from your AWS SNS account in the
 region specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para PHP da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
```

```
        """
        self.sns_resource = sns_resource


    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- Para obter detalhes da API, consulte a ListTopicsReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'aws-sdk-sns' # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
```

```
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte ListTopicsa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
```

```
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sns->listtopics( ).            " oo_result is returned for
testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTopicsreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }
```

- Para obter detalhes da API, consulte ListTopicsa referência da API AWS SDK for Swift.

Use o **`Publish`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Publish`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar um tópico e publicar mensagens
- Criar e publicar em um tópico FIFO

- Publicar uma mensagem de texto SMS
- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Publique uma mensagem em um tópico.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
```

```
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

Publique uma mensagem em um tópico com opções de grupo, duplicação e atributo.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");
```

```
            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
```

```
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
 messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
 false);
        }
    }
```

Aplique as seleções do usuário à ação de publicação.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };
```

```
        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
```

```
    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Publicar uma mensagem com um atributo.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
```

```
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Exemplo 1: Para publicar uma mensagem em um tópico:

O exemplo `publish` a seguir publica a mensagem específica no tópico do SNS especificado. A mensagem é proveniente de um arquivo de texto, o que permite incluir quebras de linha.

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
```

```
    --message file://message.txt
```

Conteúdo de `message.txt`:

```
Hello World
Second Line
```

Saída:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

Exemplo 2: Para publicar uma mensagem SMS em um número de telefone

O exemplo `publish` a seguir publica a mensagem `Hello world!` no número de telefone `+1-555-555-0100`.

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

Saída:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- Consulte detalhes da API em Publish na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
```

```
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                    if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
```

```
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Publique uma mensagem em um tópico com opções de grupo, duplicação e atributo.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
```

```
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Publish na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Consulte detalhes da API em Publish na Referência da API AWS SDK para PHP .

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo mostra a publicação de uma mensagem com uma única MessageAttribute declaração em linha.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

Exemplo 2: Este exemplo mostra a publicação de uma mensagem com várias MessageAttributes declaradas com antecedência.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- Consulte detalhes da API em Publish na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo mostra a publicação de uma mensagem com uma única MessageAttribute declaração em linha.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

Exemplo 2: Este exemplo mostra a publicação de uma mensagem com várias MessageAttributes declaradas com antecedência.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- Consulte detalhes da API em Publish na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Publique uma mensagem com atributos para que uma assinatura possa filtrar com base em atributos.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
 Values
                            must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
```

```
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

Publique uma mensagem que assume diferentes formas com base no protocolo do assinante.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
```

```
        :param default_message: The default version of the message. This version
 is
                                sent to subscribers that have protocols that are
 not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
 subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
 MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
```

```
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }
```

```
            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
 {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} -
 {e}")
            raise
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
```

```
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'SNS_TOPIC_ARN' # Should be replaced with a real topic ARN
  message = 'MESSAGE'         # Should be replaced with the actual message
 content

  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info('Sending message.')
  unless message_sender.send_message(topic_arn, message)
    @logger.error('Message sending failed. Stopping program.')
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Consulte detalhes da API em Publish na Referência da API AWS SDK para Ruby .

Rust

SDK for Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
         iv_topicarn = iv_topic_arn
         iv_message = iv_message ).
       MESSAGE 'Message published to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
   ENDTRY.
```

Publique uma mensagem com atributos para um tópico.

```
   TRY.
       oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
         iv_topicarn = iv_topic_arn
         iv_message = iv_message
         it_messageattributes = it_msg_attrs ).
       MESSAGE 'Message with attributes published to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
   ENDTRY.
```

Publique uma mensagem em vários formatos em um tópico.

```
   " Build JSON message structure for multi-format message
   DATA(lv_json_message) = |\{ "default": "{ iv_default_message }", "sms":
"{ iv_sms_message }", "email": "{ iv_email_message }" \}|.
```

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
 testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = lv_json_message
          iv_subject = iv_subject
          iv_messagestructure = 'json' ).
        MESSAGE 'Multi-format message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Publicar na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.publish(
            input: PublishInput(
                message: message,
                topicArn: arn
            )
        )

        guard let messageId = output.messageId else {
            print("No message ID received from Amazon SNS.")
            return
```

```
        }

        print("Published message with ID \(messageId)")
```

- Consulte detalhes da API em Publish na Referência de API do AWS SDK para Swift.

Use **`setSMsAttributes`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetSMSAttributes`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Como usar o Amazon SNS para definir o atributo DefaultSMSType.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
```

```
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para receber detalhes da API, consulte SetSMSAttributes na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Para definir atributos de mensagens SMS

O exemplo `set-sms-attributes` a seguir define o ID do remetente padrão para mensagens SMS como `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Este comando não produz saída.

- Para encontrar detalhes da API, consulte SetSMSAttributes na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para receber detalhes da API, consulte SetSMSAttributes na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
```

```
      attributes: {
        // Promotional - (Default) Noncritical messages, such as marketing
 messages.
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para receber detalhes da API, consulte SetSMSAttributes na Referência da API AWS SDK para JavaScript .

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
```

```
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para receber detalhes da API, consulte SetSMSAttributes na Referência da API AWS SDK para PHP .

Use **`SetSubscriptionAttributes`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetSubscriptionAttributes`.

CLI

AWS CLI

Para definir atributos de assinatura

O exemplo `set-subscription-attributes` a seguir define o atributo `RawMessageDelivery` para uma assinatura do SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

Este comando não produz saída.

O exemplo `set-subscription-attributes` a seguir define um atributo `FilterPolicy` para uma assinatura do SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Este comando não produz saída.

O exemplo `set-subscription-attributes` a seguir remove o atributo `FilterPolicy` de uma assinatura do SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SetSubscriptionAttributesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
```

```
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SetSubscriptionAttributesa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- Para obter detalhes da API, consulte a SetSubscriptionAttributesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sns->setsubscriptionattributes(
            iv_subscriptionarn = iv_subscription_arn
            iv_attributename  = 'FilterPolicy'
            iv_attributevalue = iv_filter_policy ).
        MESSAGE 'Added filter policy to subscription.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SetSubscriptionAttributesreferência da API AWS SDK for SAP ABAP.

Use **`SetSubscriptionAttributesRedrivePolicy`**com um AWS SDK

O código de exemplo a seguir mostra como usar `SetSubscriptionAttributesRedrivePolicy`.

Java

SDK para Java 1.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Specify the ARN of the Amazon SNS subscription.
```

```
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

Use **`SetTopicAttributes`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetTopicAttributes`.

CLI

AWS CLI

Para definir um atributo para um tópico

O exemplo `set-topic-attributes` a seguir define o atributo `DisplayName` para o tópico especificado.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Este comando não produz saída.

- Para obter detalhes da API, consulte SetTopicAttributesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
```

```
}
```

- Para obter detalhes da API, consulte a SetTopicAttributesreferência da API AWS SDK for Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
```

```
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para PHP da API.

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
```

```
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: '2008-10-17',
      Id: '__default_policy_ID',
      Statement: [{
        Sid: '__default_statement_ID',
        Effect: 'Allow',
        Principal: { "AWS": '*' },
        Action: ['SNS:Publish'],
        Resource: topic_arn,
        Condition: {
          ArnEquals: {
            "AWS:SourceArn": resource_arn
          }
        }
      }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'MY_TOPIC_ARN' # Should be replaced with a real topic ARN
  resource_arn = 'MY_RESOURCE_ARN' # Should be replaced with a real resource ARN
```

```
  policy_name = 'POLICY_NAME' # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte SetTopicAttributesa Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sns->settopicattributes(
            iv_topicarn = iv_topic_arn
            iv_attributename  = iv_attribute_name
            iv_attributevalue = iv_attribute_value ).
        MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SetTopicAttributesreferência da API AWS SDK for SAP ABAP.

Use **`Inscrever-se`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Subscribe`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar um tópico e publicar mensagens
- Criar e publicar em um tópico FIFO
- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
/// <summary>
/// Creates a new subscription to a topic.
/// </summary>
/// <param name="client">The initialized Amazon SNS client object, used
/// to create an Amazon SNS subscription.</param>
/// <param name="topicArn">The ARN of the topic to subscribe to.</param>
/// <returns>A SubscribeResponse object which includes the subscription
/// ARN for the new subscription.</returns>
public static async Task<SubscribeResponse> TopicSubscribeAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    SubscribeRequest request = new SubscribeRequest()
    {
        TopicArn = topicArn,
        ReturnSubscriptionArn = true,
        Protocol = "email",
        Endpoint = "recipient@example.com",
    };

    var response = await client.SubscribeAsync(request);
```

```
            return response;
        }
```

Inscreva uma fila em um tópico com filtros opcionais.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
```

```
                    << std::endl;
    }

    return outcome.IsSuccess();
}
```

Inscrever uma aplicação móvel em um tópico.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                    << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                    << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

Inscrever uma função do Lambda em um tópico.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

Assinar uma fila do SQS em um tópico.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);

            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

Assinar com um filtro em um tópico.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
```

```
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
```

```
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";
```

```
        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Para inscrever-se em um tópico

O comando `subscribe` a seguir inscreve um endereço de e-mail no tópico especificado.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Saída:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Consulte detalhes da API em Subscribe na Referência de comandos da AWS CLI .

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscreva uma fila em um tópico com filtros opcionais.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
```

```
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para Go .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
 email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
```

```
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Inscrever um endpoint HTTP em um tópico.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
```

```
                    url - The HTTPS endpoint that you want to receive
 notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn()
 + "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }
```

```
    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
```

```
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

Inscrever uma aplicação móvel em um tópico.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Inscrever uma função do Lambda em um tópico.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
```

```
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Assinar uma fila do SQS em um tópico.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
```

```
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Assinar com um filtro em um tópico.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
```

```
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
```

```
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Consulte detalhes da API em Subscribe na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Inscrever um endpoint HTTP em um tópico.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
 subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.
```

```
        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
number
                         (in E.164 format) for SMS messages, or an email address
for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
topic.arn
            )
            raise
        else:
            return subscription
```

Inscreva uma fila em um tópico com filtros opcionais.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.
```

```
        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise
```

- Consulte detalhes da API em Subscribe na Referência da API do AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
require 'aws-sdk-sns'
require 'logger'

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
```

```
    @logger.info('Subscription created successfully.')
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = 'email'
  endpoint = 'EMAIL_ADDRESS' # Should be replaced with a real email address
  topic_arn = 'TOPIC_ARN'    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info('Creating the subscription.')
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error('Subscription creation failed. Stopping program.')
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para Ruby .

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Consulte detalhes da API em Subscribe na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
   TRY.
       oo_result = lo_sns->subscribe(                      "oo_result is
returned for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- Consulte detalhes da API em Inscrever-se na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: email,
```

```
                protocol: "email",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

Inscrição de um número de telefone em um tópico para receber notificações por SMS.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: phone,
                protocol: "sms",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

- Consulte detalhes da API em Subscribe na Referência de API do AWS SDK para Swift.

Use **`TagResource`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TagResource`.

CLI

AWS CLI

Para adicionar uma tag a um tópico

O exemplo `tag-resource` a seguir adiciona uma tag de metadados ao tópico do Amazon SNS especificado.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Este comando não produz saída.

- Para obter detalhes da API, consulte TagResourceem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
```

```
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte TagResourcea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
        Tag {
```

```
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request =
        TagResourceRequest {
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- Para obter detalhes da API, consulte a TagResourcereferência da API AWS SDK for Kotlin.

Use **`Cancelar assinatura`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `Unsubscribe`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar um tópico e publicar mensagens
- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cancele a assinatura de um tópico por meio de um ARN de assinatura.

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK para C++ .

CLI

AWS CLI

Para cancelar a assinatura de um tópico

O exemplo `unsubscribe` a seguir exclui a assinatura especificada de um tópico.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Este comando não produz saída.

- Consulte detalhes da API em Unsubscribe na Referência de comandos da AWS CLI .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
```

```
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK for Java 2.x .

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK para JavaScript .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Consulte detalhes da API em Cancelar assinatura na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
```

```
            logger.error(f"Error unsubscribing: {error_code} - {e}")
            raise
```

- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
       MESSAGE 'Subscription deleted.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Subscription does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snsinvalidparameterex.
       MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
   ENDTRY.
```

- Consulte detalhes da API em Cancelar assinatura na Referência da API do AWS SDK para SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(
                subscriptionArn: arn
            )
        )

        print("Unsubscribed.")
```

- Consulte detalhes da API em Unsubscribe na Referência de API do AWS SDK para Swift.

Cenários para o Amazon SNS usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon SNS com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon SNS ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação para enviar dados para uma tabela do DynamoDB

- Criar uma aplicação de publicação e assinatura que traduz mensagens
- Criar um tópico e publicar mensagens
- Crie um endpoint de plataforma para notificações push do Amazon SNS usando um AWS SDK
- Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos
- Criar uma aplicação de exploração do Amazon Textract
- Crie e publique em um tópico FIFO do Amazon SNS usando um AWS SDK
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos da proteção de dispositivos da Internet das Coisas
- Publique mensagens SMS em um tópico do Amazon SNS usando um AWS SDK
- Publique uma mensagem grande no Amazon SNS com o Amazon S3 usando um AWS SDK
- Publique uma mensagem de texto SMS do Amazon SNS usando um AWS SDK
- Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um AWS SDK
- Usar o API Gateway para invocar uma função do Lambda
- Usar eventos programados para chamar uma função do Lambda

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

Os exemplos de código a seguir mostram como criar uma aplicação que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

Java

SDK para Java 2.x

Mostra como criar uma aplicação Web dinâmica que envia dados usando a API Java do Amazon DynamoDB e envia uma mensagem de texto usando a API Java do Amazon Simple Notification Service.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB

- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Este exemplo mostra como criar uma aplicação que permite que os usuários enviem dados para uma tabela do Amazon DynamoDB e enviem uma mensagem de texto ao administrador usando o Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Criar uma aplicação de publicação e assinatura que traduz mensagens

Os exemplos de código a seguir mostram como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo que usa a API Java Async, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Kotlin

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon SNS para criar uma aplicação com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e as instruções sobre como criar um aplicativo web, veja o exemplo completo em GitHub.

Para ver o código-fonte completo e instruções sobre como criar um aplicativo Android nativo, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Criar um tópico e publicar mensagens

O exemplo de código a seguir mostra como:

- Criar um tópico do Amazon SNS.
- Inscrever um endpoint de e-mail no tópico.
- Verificar sua assinatura.
- Publicar uma mensagem no tópico.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SNS Getting Started Script
# This script demonstrates how to create an SNS topic, subscribe to it, publish a
 message,
# and clean up resources.

set -euo pipefail

# Set up logging with secure file permissions
LOG_FILE="sns-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon SNS Getting Started Tutorial..."
echo "$(date)"
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
```

```
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    if [ -n "${SUBSCRIPTION_ARN:-}" ] && [ "$SUBSCRIPTION_ARN" != "pending
 confirmation" ] && [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ]; then
        echo "Deleting subscription: $SUBSCRIPTION_ARN"
        if ! aws sns unsubscribe --subscription-arn "$SUBSCRIPTION_ARN" --region
 "$AWS_REGION" 2>/dev/null; then
            echo "Warning: Failed to delete subscription" >&2
        fi
    fi

    if [ -n "${TOPIC_ARN:-}" ]; then
        echo "Deleting topic: $TOPIC_ARN"
        if ! aws sns delete-topic --topic-arn "$TOPIC_ARN" --region "$AWS_REGION"
 2>/dev/null; then
            echo "Warning: Failed to delete topic" >&2
        fi
    fi

    return $exit_code
}

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Set trap to cleanup on exit
trap cleanup_resources EXIT

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi
```

```
if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random topic name suffix using secure method
RANDOM_SUFFIX=$(openssl rand -hex 4)
TOPIC_NAME="my-topic-${RANDOM_SUFFIX}"

# Validate topic name length (max 256 characters)
if [ ${#TOPIC_NAME} -gt 256 ]; then
    handle_error "Topic name exceeds maximum length of 256 characters"
fi

# Step 1: Create an SNS topic
echo "Creating SNS topic: $TOPIC_NAME"
TOPIC_RESULT=$(aws sns create-topic \
    --name "$TOPIC_NAME" \
    --region "$AWS_REGION" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
    --output json) || handle_error "Failed to create SNS topic"

# Extract the topic ARN using jq for reliable parsing
TOPIC_ARN=$(echo "$TOPIC_RESULT" | jq -r '.TopicArn // empty') || handle_error
 "Failed to parse topic result"

if [ -z "$TOPIC_ARN" ]; then
    handle_error "Failed to extract topic ARN from result: $TOPIC_RESULT"
fi

# Validate ARN format
if [[ ! "$TOPIC_ARN" =~ ^arn:aws:sns:[a-z0-9-]+:[0-9]{12}:[a-zA-Z0-9_-]+$ ]];
 then
    handle_error "Invalid SNS topic ARN format: $TOPIC_ARN"
fi

echo "Successfully created topic with ARN: $TOPIC_ARN"

# Step 2: Subscribe to the topic using Email-JSON protocol to reduce costs
echo ""
echo "==============================================="
echo "EMAIL SUBSCRIPTION"
echo "==============================================="
```

```
EMAIL_ADDRESS="test-${RANDOM_SUFFIX}@example.com"

# Validate email format (basic validation)
if [[ ! "$EMAIL_ADDRESS" =~ ^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]];
 then
    handle_error "Invalid email format: $EMAIL_ADDRESS"
fi

echo "Subscribing email: $EMAIL_ADDRESS to topic using Email-JSON protocol"
SUBSCRIPTION_RESULT=$(aws sns subscribe \
    --topic-arn "$TOPIC_ARN" \
    --protocol email-json \
    --notification-endpoint "$EMAIL_ADDRESS" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to create subscription"

# Extract the subscription ARN using jq
SUBSCRIPTION_ARN=$(echo "$SUBSCRIPTION_RESULT" | jq -r '.SubscriptionArn //
 empty') || handle_error "Failed to parse subscription result"

echo "Subscription created: $SUBSCRIPTION_ARN"
echo "A confirmation email has been sent to $EMAIL_ADDRESS"
echo ""

# Tag the subscription
if [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ] && [ "$SUBSCRIPTION_ARN" !=
 "pending confirmation" ]; then
    aws sns tag-resource \
        --resource-arn "$SUBSCRIPTION_ARN" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
        --region "$AWS_REGION" 2>/dev/null || echo "Warning: Failed to tag
 subscription"
fi

# Step 3: List subscriptions to verify
echo "Listing subscriptions for topic: $TOPIC_ARN"
SUBSCRIPTIONS=$(aws sns list-subscriptions-by-topic --topic-arn "$TOPIC_ARN"
 --region "$AWS_REGION" --output json) || handle_error "Failed to list
 subscriptions"

echo "Current subscriptions:"
echo "$SUBSCRIPTIONS" | jq '.'
```

```
# Get the confirmed subscription ARN with optimized jq query and improved error
 handling
CONFIRMED_SUBSCRIPTION=$(echo "$SUBSCRIPTIONS" | jq -r '.Subscriptions[]? |
 select(.SubscriptionArn != "PendingConfirmation") | .SubscriptionArn' 2>/dev/
null | head -n 1)

if [ -n "$CONFIRMED_SUBSCRIPTION" ]; then
    SUBSCRIPTION_ARN="$CONFIRMED_SUBSCRIPTION"
else
    echo "Warning: No confirmed subscription found. You may not have confirmed
 the subscription yet."
    echo "The script will continue, but you may not receive the test message."
fi

# Step 4: Publish a message to the topic
echo ""
echo "Publishing a test message to the topic"
MESSAGE="Hello from Amazon SNS! This is a test message sent at $(date)."

# Validate message length (max 256 KB for SNS)
if [ ${#MESSAGE} -gt 262144 ]; then
    handle_error "Message exceeds maximum size of 256 KB"
fi

PUBLISH_RESULT=$(aws sns publish \
    --topic-arn "$TOPIC_ARN" \
    --message "$MESSAGE" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to publish message"

MESSAGE_ID=$(echo "$PUBLISH_RESULT" | jq -r '.MessageId // empty') ||
 handle_error "Failed to parse publish result"

if [ -z "$MESSAGE_ID" ]; then
    handle_error "No message ID returned from publish operation"
fi

# Validate message ID format (UUID v4)
if [[ ! "$MESSAGE_ID" =~ ^[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-
f0-9]{12}$ ]]; then
    handle_error "Unexpected message ID format: $MESSAGE_ID"
fi

echo "Message published successfully with ID: $MESSAGE_ID"
```

```
echo "Check your email for the message."

# Pause to allow the user to check their email
echo ""
echo "Pausing for 3 seconds to allow message delivery..."
sleep 3

# Step 5: Clean up resources
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Resources created:"
echo "- SNS Topic: $TOPIC_ARN"
echo "- Subscription: ${SUBSCRIPTION_ARN:-N/A}"
echo ""
echo "Cleaning up resources to avoid unnecessary charges..."

echo ""
echo "Tutorial completed successfully!"
echo "$(date)"
echo "================================================"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTopic
  - DeleteTopic
  - ListSubscriptionsByTopic
  - Publicar
  - Assinar
  - Cancelar assinatura

Crie um endpoint de plataforma para notificações push do Amazon SNS usando um AWS SDK

Os exemplos de código a seguir mostram como criar um endpoint de plataforma para notificações por push do Amazon SNS.

CLI

AWS CLI

Para criar um endpoint de aplicação de plataforma

O exemplo `create-platform-endpoint` a seguir cria um endpoint para a aplicação de plataforma especificada usando o token especificado.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Saída:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
               token - The device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               platformApplicationArn - The ARN value of platform application.
 You can get this value from the AWS Management Console.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação de exploração do Amazon Textract

Os exemplos de código a seguir mostram como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.

- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Crie e publique em um tópico FIFO do Amazon SNS usando um AWS SDK

Os exemplos de código a seguir mostram como criar e publicar em um tópico FIFO do Amazon SNS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O teste verifica o recebimento da mensagem em cada fila. O exemplo completo também mostra a adição de políticas de acesso e exclui os recursos no final.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
            "Where:\n" +
            "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
            "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
            +
            "   retailQueueARN - The name of a SQS FIFO queue that will created
 for the retail consumer. \n\n" +
            "   analyticsQueueARN - The name of a SQS standard queue that will be
 created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
 ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);
```

```
        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
 "Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
```

```
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();
```

```
            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTopic
  - Publicar
  - Assinar

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO do Amazon SNS, inscreva filas padrão e FIFO do Amazon SQS no tópico e publique uma mensagem no tópico.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
```

```
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])
```

```
    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource
```

```
    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version":"2012-10-17",
```

```
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
```

```
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
 "wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTopic
  - Publicar
  - Assinar

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO, inscreva uma fila FIFO do Amazon SQS no tópico e publique uma mensagem em um tópico do Amazon SNS.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
```

```
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String'

 iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
```

```
            iv_message = 'The price of your mobile plan has been increased from
$19 to $23'
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes ).
       ov_message_id = lo_result->get_messageid( ).                    "
ov_message_id is returned for testing purposes. "
       MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para SAP ABAP.
  - CreateTopic
  - Publicar
  - Assinar

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.

- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
```

```
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
```

```
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
```

```
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
```

```
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
```

```
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
```

```
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"
```

```
    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
```

```
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Conceitos básicos da proteção de dispositivos da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM obrigatórios.
- Habiltiar verificações de auditoria do IoT Device Defender.
- Execute uma On-Demand auditoria
- Criar uma ação de mitigação.
- Aplicar ações de mitigação às descobertas.
- Configurar notificações do SNS (opcional).
- Habilitar o registro em log de IoT

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "==================================================="
echo "Starting script execution at $(date)"
```

```
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /
dev/null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null) || true
```

```
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn'
 --output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
```

```
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)
```

```
        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
```

```
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}
```

```
# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "====================================================="
echo "Step 1: Creating required IAM roles"
echo "====================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")
```

```
# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
```

```
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "====================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "====================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1)
 || true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"
```

```
# Step 3: Run an on-demand audit
echo ""
echo "====================================================="
echo "Step 3: Running an on-demand audit"
echo "====================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi
```

```
    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null ||
 echo "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi
```

```
# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "====================================================="
echo "Step 4: Creating a mitigation action"
echo "====================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting
 it first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction"
 2>&1 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi
```

```
MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1
 || true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "==================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "==================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)
```

```
    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1)
 || true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
```

```
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --
output json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
```

```
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS
 notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "====================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "====================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi
```

```
    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2
 methods, continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved
 but could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10
```

```
# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
```

```
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --
output json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null
 || echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}
```

```
cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies
  - ListTopics
  - PutRolePolicy
  - SetLoggingOptions

- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Publique mensagens SMS em um tópico do Amazon SNS usando um AWS SDK

O código de exemplo a seguir mostra como:

- Criar um tópico do Amazon SNS.
- Inscrever números de telefone no tópico.
- Publicar mensagens SMS no tópico para que todos os números de telefone inscritos recebam a mensagem de uma só vez.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar um tópico e retorne seu ARN.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

Inscreva um endpoint em um tópico.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Defina atributos na mensagem, como o ID do remetente, o preço máximo e seu tipo. Os atributos de mensagem são opcionais.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Publique uma mensagem em um tópico. A mensagem é enviada para todos os assinantes.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
```

```
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Publique uma mensagem grande no Amazon SNS com o Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como publicar uma mensagem grande no Amazon SNS usando o Amazon S3 para armazenar a carga útil da mensagem.

Java

SDK para Java 1.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para publicar uma mensagem grande, use a Amazon SNS Extended Client Library for Java. A mensagem que você envia faz referência a um objeto do Amazon S3 que contém o conteúdo real da mensagem.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;

                // Message threshold controls the maximum message size that will
 be allowed to
                // be published
                // through SNS using the extended client. Payload of messages
 exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
 maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
 AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
 AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
 AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
```

```
                final String topicArn = snsClient.createTopic(
                                new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest
subscriptionAttributesRequest = new SetSubscriptionAttributesRequest();

subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");

snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3
when the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration
snsExtendedClientConfiguration = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

.withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it
exceeds the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
```

```
                final ExtendedClientConfiguration sqsExtendedClientConfiguration
 = new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
 AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
 sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
 result.getMessages().get(0).getBody());
        }
}
```

Publique uma mensagem de texto SMS do Amazon SNS usando um AWS SDK

Os exemplos de código a seguir mostram como publicar mensagens SMS usando o Amazon SNS.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
```

```
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
```

```
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para .NET .

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */
```

```
//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para C++ .

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte Publish na Referência da API do AWS SDK para PHP .

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " iv_phone_number = '+12065550101' - Phone number in E.164 format
   TRY.
       oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
         iv_phonenumber = iv_phone_number
         iv_message = iv_message ).
       MESSAGE 'Message published to phone number.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Phone number does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter os detalhes da API, consulte Publicar na Referência da API do AWS SDK para SAP ABAP.

Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
/// <summary>
/// Console application to run a feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
```

```
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
```

```
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
```

```
                            $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                            $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
```

```
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
```

```
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
```

```
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
```

```
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
```

```
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }
```

```
        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
```

```
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
```

```
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Crie uma classe que envolva operações do Amazon SQS.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;
```

```
    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
```

```
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
```

```
                                        "\"Effect\": \"Allow\"," +
                                        "\"Principal\": {" +
                                            $"\"Service\": " +
                                                "\"sns.amazonaws.com\"" +
                                               "}," +
                                        "\"Action\": \"sqs:SendMessage\"," +
                                        $"\"Resource\": \"{queueArn}\"," +
                                         "\"Condition\": {" +
                                              "\"ArnEquals\": {" +
                                                  $"\"aws:SourceArn\":
 \"{topicArn}\"" +
                                               "}" +
                                            "}" +
                                     "}]" +
                                  "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
```

```
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
}
```

Crie uma classe que envolva operações do Amazon SNS.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
```

```
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
```

```
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
```

```
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
```

```
   printAsterisksLine();
   std::cout << "In this workflow, you will create an SNS topic and subscribe "
             << NUMBER_OF_QUEUES <<
             " SQS queues to the topic." << std::endl;
   std::cout
           << "You can select from several options for configuring the topic and
the subscriptions for the "
           << NUMBER_OF_QUEUES << " queues." << std::endl;
   std::cout << "You can then post to the topic and see the results in the
queues."
             << std::endl;

   Aws::SNS::SNSClient snsClient(clientConfiguration);

   printAsterisksLine();

   std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
             << std::endl;
   std::cout
           << "FIFO topics deliver messages in order and support deduplication
and message filtering."
           << std::endl;
   bool isFifoTopic = askYesNoQuestion(
           "Would you like to work with FIFO topics? (y/n) ");

   bool contentBasedDeduplication = false;
   Aws::String topicName;
   if (isFifoTopic) {
       printAsterisksLine();
       std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                 << std::endl;
       std::cout
               << "Deduplication IDs are either set in the message or
automatically generated "
               << "from content using a hash function." << std::endl;
       std::cout
               << "If a message is successfully published to an SNS FIFO topic,
any message "
               << "published and determined to have the same deduplication ID, "
               << std::endl;
       std::cout
               << "within the five-minute deduplication interval, is accepted
but not delivered."
```

```
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
```

```
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                        "true");
```

```
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
```

```
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
```

```
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
```

```
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;
```

```
                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

```
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
```

```
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
```

```
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
```

```
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);
```

```
                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

```
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
```

```
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publicar
- ReceiveMessage
- SetQueueAttributes
- Assinar
- Cancelar assinatura

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
 "strings"
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
```

```
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string,
 bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
```

```
    "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
   contentBasedDeduplication = runner.questioner.AskBool(
    "\nDo you want to use content-based deduplication instead of entering a
  deduplication ID? (y/n) ", "y")
  }
  log.Println(strings.Repeat("-", 88))

  topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
  if isFifoTopic {
   topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
   log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
    "the topic name.", FIFO_SUFFIX)
  }

  topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
  contentBasedDeduplication)
  if err != nil {
   panic(err)
  }
  log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
  \n"+
   "'%v' has been created.", topicName, topicArn)

  return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
  isFifoTopic bool) (string, string) {
  queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
  queue. ", ordinal))
  if isFifoTopic {
   queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
   if ordinal == "first" {
    log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
     "be appended to the queue name.\n", FIFO_SUFFIX)
   }
  }
  queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
  if err != nil {
   panic(err)
  }
  log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
   "'%v' has been created.", queueName, queueUrl)
```

```
 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string,
 topicArn string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
    "If you add a filter to this subscription, then only the filtered messages\n"
 +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
  if wantFiltering {
   log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")

   var toneSelections []string
   askAboutTones := true
```

```
    for askAboutTones {
     toneIndex := runner.questioner.AskChoice(
      "Enter the number of the tone you want to filter by:\n", ToneChoices)
     toneSelections = append(toneSelections, ToneChoices[toneIndex])
     askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
  the filter? (y/n) ", "y")
    }
    log.Printf("Your subscription will be filtered to only pass the following
  tones: %v\n", toneSelections)
    filterPolicy = map[string][]string{TONE_KEY: toneSelections}
   }
  }

  subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
  filterPolicy)
  if err != nil {
   panic(err)
  }
  log.Printf("The queue %v is now subscribed to the topic %v with the subscription
  ARN %v.\n",
   queueName, topicName, subscriptionArn)

  return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn
  string, isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
  var message string
  var groupId string
  var dedupId string
  var toneSelection string
  publishMore := true
  for publishMore {
   groupId = ""
   dedupId = ""
   toneSelection = ""
   message = runner.questioner.Ask("Enter a message to publish: ")
   if isFifoTopic {
    log.Println("Because you are using a FIFO topic, you must set a message group
  ID.\n" +
     "All messages within the same group will be received in the order they were
  published.")
    groupId = runner.questioner.Ask("Enter a message group ID: ")
    if !contentBasedDeduplication {
```

```
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId,
 TONE_KEY, toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
```

```
    log.Printf("One message was received by queue %v:\n", queueUrl)

   } else {
    log.Printf("%v messages were received by queue %v:\n", len(messages),
  queueUrl)
   }
   for msgIndex, message := range messages {
    messageBody := MessageBody{}
    err := json.Unmarshal([]byte(*message.Body), &messageBody)
    if err != nil {
     panic(err)
    }
    log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
   }

   if len(messages) > 0 {
    log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
     panic(err)
    }
   }
  }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
```

```
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
 topicName, topicArn, ordinal, isFifoTopic)
```

```
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina um struct que envolva as ações do Amazon SNS usadas neste exemplo.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
```

```
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

```
// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
```

```
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Defina um struct que envolva as ações do Amazon SQS usadas neste exemplo.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
```

```
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
```

```
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
```

```
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
```

```
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Limpe recursos.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management
 Console\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
```

```
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
```

```
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
```

```
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and
 the subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,
\n"
                +
```

```
                "        within the five-minute deduplication interval, is
 accepted but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
 must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
```

```
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
```

```
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

        } else {
```

```
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
```

```
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
```

```
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
```

```
                                        String msgAttValue,
                                        String duplication,
                                        String groupId,
                                        String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
```

```
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
```

```
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);
```

```
            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
```

```
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Publicar
- ReceiveMessage
- SetQueueAttributes
- Assinar
- Cancelar assinatura

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o ponto de entrada para esse cenário.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

O código anterior fornece as dependências necessárias e inicia o cenário. A próxima seção contém a maior parte do exemplo.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
    this.sqsClient = sqsClient;
    this.prompter = prompter;
    this.logger = logger;
  }
```

```
  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
      await this.logger.log(MESSAGES.deduplicationNotice);
      await this.logger.log(MESSAGES.deduplicationDescription);
      this.autoDedup = await this.prompter.confirm({
        message: MESSAGES.deduplicationPrompt,
      });
    }
  }

  async createTopic() {
    await this.logger.log(MESSAGES.creatingTopics);
    this.topicName = await this.prompter.input({
      message: MESSAGES.topicNamePrompt,
    });
    if (this.isFifo) {
      this.topicName += ".fifo";
      this.logger.logSeparator(MESSAGES.headerFifoNaming);
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
```

```
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });
```

```
    await this.logger.log(
      MESSAGES.queueCreatedNotice
        .replace("${QUEUE_NAME}", queueName)
        .replace("${QUEUE_URL}", response.QueueUrl)
        .replace("${QUEUE_ARN}", Attributes.QueueArn),
    );
  }
}

async attachQueueIamPolicies() {
  for (const [index, queue] of this.queues.entries()) {
    const policy = JSON.stringify(
      {
        Statement: [
          {
            Effect: "Allow",
            Principal: {
              Service: "sns.amazonaws.com",
            },
            Action: "sqs:SendMessage",
            Resource: queue.queueArn,
            Condition: {
              ArnEquals: {
                "aws:SourceArn": this.topicArn,
              },
            },
          },
        ],
      },
      null,
      2,
    );

    if (index !== 0) {
      this.logger.logSeparator();
    }

    await this.logger.log(MESSAGES.attachPolicyNotice);
    console.log(policy);
    const addPolicy = await this.prompter.confirm({
      message: MESSAGES.addPolicyConfirmation.replace(
        "${QUEUE_NAME}",
        queue.queueName,
```

```
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
```

```
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });
```

```
      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });
```

```
    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });
```

```
    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
```

```
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns
```

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an
 SQS queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
 and message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
```

```
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
 generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
```

```
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)
```

```
    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's
 subscription to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the
 following \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
```

```
            println("You can filter messages by one or more of the following
\"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
```

```
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
 to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }
```

```
        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
```

```
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
```

```
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }
```

```
            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")
```

```
            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
```

```
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
```

```
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
```

```
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
        self.queue_count = 2
        self.queue_urls = []
        self.subscription_arns = []
        self.tones = ["cheerful", "funny", "serious", "sincere"]

    def run_scenario(self) -> None:
        """Run the Topics and Queues feature scenario."""
        print(self.DASHES)
        print("Welcome to messaging with topics and queues.")
        print(self.DASHES)
        print(f"""
    In this scenario, you will create an SNS topic and subscribe
{self.queue_count} SQS queues to the topic.
    You can select from several options for configuring the topic and the
subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
        """)

        try:
            # Setup Phase
            print(self.DASHES)
            self._setup_topic()
            print(self.DASHES)

            self._setup_queues()
            print(self.DASHES)

            # Demonstration Phase
            self._publish_messages()
            print(self.DASHES)

            # Examination Phase
            self._poll_queues_for_messages()
            print(self.DASHES)
```

```
            # Cleanup Phase
            self._cleanup_resources()
            print(self.DASHES)

        except Exception as e:
            logger.error(f"Scenario failed: {e}")
            print(f"There was a problem with the scenario: {e}")
            print("\nInitiating cleanup...")
            try:
                self._cleanup_resources()
            except Exception as cleanup_error:
                logger.error(f"Error during cleanup: {cleanup_error}")

        print("Messaging with topics and queues scenario is complete.")
        print(self.DASHES)

    def _setup_topic(self) -> None:
        """Set up the SNS topic to be used with the queues."""
        print("SNS topics can be configured as FIFO (First-In-First-Out).")
        print("FIFO topics deliver messages in order and support deduplication
 and message filtering.")
        print()

        self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/
n): ", q.is_yesno)

        if self.use_fifo_topic:
            print(self.DASHES)
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
 q.non_empty)
            print("Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name.")
            print()

            print(self.DASHES)
            print("""
    Because you have chosen a FIFO topic, deduplication is supported.
    Deduplication IDs are either set in the message or automatically generated
    from content using a hash function.

    If a message is successfully published to an SNS FIFO topic, any message
    published and determined to have the same deduplication ID,
    within the five-minute deduplication interval, is accepted but not delivered.
```

```
    For more information about deduplication,
    see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
            """)

            self.use_content_based_deduplication = q.ask(
                "Use content-based deduplication instead of entering a
deduplication ID? (y/n): ",
                q.is_yesno
            )
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)

        print(self.DASHES)

        # Create the topic
        self.topic_arn = self.sns_wrapper.create_topic(
            self.topic_name,
            self.use_fifo_topic,
            self.use_content_based_deduplication
        )

        print(f"Your new topic with the name {self.topic_name}")
        print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
        print(f"  has been created.")
        print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
        print(f"Now you will create {self.queue_count} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ",
q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)
```

```
            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()

            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")
                print(self.DASHES)

            # Get queue ARN
            queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

            if i == 0:
                print("An AWS Identity and Access Management (IAM) policy must be
 attached to an SQS queue,")
                print("enabling it to receive messages from an SNS topic.")

            # Set queue policy to allow SNS to send messages
            self.sqs_wrapper.set_queue_policy_for_topic(queue_arn,
 self.topic_arn, queue_url)

            # Set up message filtering if using FIFO
            subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
 queue_name)
            self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
 queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
 filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
```

```
                print()
                print("For this example, you can filter messages by a TONE
attribute.")

            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription
to the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
{self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

    def _create_filter_policy(self) -> str:
        """Create a message filter policy based on user selections."""
        print(self.DASHES)
        print("You can filter messages by one or more of the following TONE
attributes.")

        filter_selections = []
        selection_number = 0

        while True:
            print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
            for i, tone in enumerate(self.tones, 1):
                print(f"  {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

            if selection == 0:
                break
            elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
                filter_selections.append(self.tones[selection - 1])
                print(f"Added '{self.tones[selection - 1]}' to filter list.")
```

```
        if filter_selections:
            filters = {"tone": filter_selections}
            return json.dumps(filters)
        return None

    def _publish_messages(self) -> None:
        """Publish messages to the topic with various options."""
        print("Now we can publish messages.")

        keep_sending = True
        while keep_sending:
            print()
            message = q.ask("Enter a message to publish: ", q.non_empty)

            message_group_id = None
            deduplication_id = None
            tone_attribute = None

            if self.use_fifo_topic:
                print("Because you are using a FIFO topic, you must set a message
group ID.")
                print("All messages within the same group will be received in the
order they were published.")
                print()
                message_group_id = q.ask("Enter a message group ID for this
message: ", q.non_empty)

                if not self.use_content_based_deduplication:
                    print("Because you are not using content-based
deduplication,")
                    print("you must enter a deduplication ID.")
                    deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

                # Ask about tone attribute
                add_attribute = q.ask("Add an attribute to this message? (y/n):
", q.is_yesno)
                if add_attribute:
                    print("Enter a number for an attribute:")
                    for i, tone in enumerate(self.tones, 1):
                        print(f"  {i}. {tone}")
```

```
                    selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
 len(self.tones)))
                    if 1 <= selection <= len(self.tones):
                        tone_attribute = self.tones[selection - 1]

            # Publish the message
            message_id = self.sns_wrapper.publish_message(
                self.topic_arn,
                message,
                tone_attribute,
                deduplication_id,
                message_group_id
            )

            print(f"Message published with ID: {message_id}")

            keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

    def _poll_queues_for_messages(self) -> None:
        """Poll all queues for messages and display results."""
        for i, queue_url in enumerate(self.queue_urls):
            print(f"Polling queue #{i+1} at {queue_url} for messages...")

            q.ask("Press Enter to continue...")

            messages = self._poll_queue_for_messages(queue_url)

            if messages:
                print(f"{len(messages)} message(s) were received by queue #{i
+1}")
                for j, message in enumerate(messages, 1):
                    print(f"  Message {j}:")
                    # Parse the SNS message body to get the actual message
                    try:
                        sns_message = json.loads(message['Body'])
                        actual_message = sns_message.get('Message',
 message['Body'])
                        print(f"    {actual_message}")
                    except (json.JSONDecodeError, KeyError):
                        print(f"    {message['Body']}")

                # Delete the messages
                self.sqs_wrapper.delete_messages(queue_url, messages)
                print(f"Messages deleted from queue #{i+1}")
```

```
            else:
                print(f"No messages received by queue #{i+1}")

            print(self.DASHES)

    def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
        """Poll a single queue for messages."""
        all_messages = []
        max_polls = 3  # Limit polling to avoid infinite loops

        for poll_count in range(max_polls):
            messages = self.sqs_wrapper.receive_messages(queue_url, 10)

            if messages:
                all_messages.extend(messages)
                print(f"  Received {len(messages)} messages in poll {poll_count +
1}")
                # Small delay between polls
                time.sleep(1)
            else:
                print(f"  No messages in poll {poll_count + 1}")
                break

        return all_messages

    def _cleanup_resources(self) -> None:
        """Clean up all resources created during the scenario."""
        print("Cleaning up resources...")

        # Delete queues
        for i, queue_url in enumerate(self.queue_urls):
            if queue_url:
                delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}?
(y/n): ", q.is_yesno)
                if delete_queue:
                    try:
                        self.sqs_wrapper.delete_queue(queue_url)
                        print(f"Deleted queue #{i+1}")
                    except Exception as e:
                        print(f"Error deleting queue #{i+1}: {e}")

        # Unsubscribe from topic
        for i, subscription_arn in enumerate(self.subscription_arns):
            if subscription_arn:
```

```
                try:
                    self.sns_wrapper.unsubscribe(subscription_arn)
                    print(f"Unsubscribed subscription #{i+1}")
                except Exception as e:
                    print(f"Error unsubscribing #{i+1}: {e}")

        # Delete topic
        if self.topic_arn:
            delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
 q.is_yesno)
            if delete_topic:
                try:
                    self.sns_wrapper.delete_topic(self.topic_arn)
                    print(f"Deleted topic {self.topic_name}")
                except Exception as e:
                    print(f"Error deleting topic: {e}")

        print("Resource cleanup complete.")
```

Crie classes que envolvam as operações do Amazon SNS e do Amazon SQS para uso no cenário.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
```

```
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} -
{e}")
            raise
```

```
    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
```

```
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
```

```
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} -
{e}")
            raise


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
                raise


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True
```

```
        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise


    def list_topics(self) -> list:
        """
        List all SNS topics in the account using pagination.

        :return: List of topic ARNs.
        :raises ClientError: If listing topics fails.
        """
        try:
            topics = []
            paginator = self.sns_client.get_paginator('list_topics')

            for page in paginator.paginate():
                topics.extend([topic['TopicArn'] for topic in page.get('Topics',
 [])])

            logger.info(f"Found {len(topics)} topics")
            return topics

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AuthorizationError':
                logger.error("Authorization error listing topics - check IAM
 permissions")
            else:
                logger.error(f"Error listing topics: {error_code} - {e}")
            raise


class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
```

```
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url
```

```
        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} -
{e}")
            raise


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
queue_url: str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
```

```
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
```

```
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
```

```
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
```

```
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise


    def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
        """
        List all SQS queues in the account using pagination.

        :param queue_name_prefix: Optional prefix to filter queue names.
        :return: List of queue URLs.
        :raises ClientError: If listing queues fails.
        """
        try:
            queue_urls = []
            paginator = self.sqs_client.get_paginator('list_queues')

            page_params = {}
            if queue_name_prefix:
                page_params['QueueNamePrefix'] = queue_name_prefix

            for page in paginator.paginate(**page_params):
                queue_urls.extend(page.get('QueueUrls', []))

            logger.info(f"Found {len(queue_urls)} queues")
            return queue_urls

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AccessDenied':
                logger.error("Access denied listing queues - check IAM
permissions")
            else:
                logger.error(f"Error listing queues: {error_code} - {e}")
            raise

    def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
        """
        Send a message to an SQS queue.

        :param queue_url: The URL of the queue.
        :param message_body: The message content.
```

```
        :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
        :return: The message ID.
        :raises ClientError: If sending the message fails.
        """
        try:
            send_params = {
                'QueueUrl': queue_url,
                'MessageBody': message_body,
                **kwargs
            }

            response = self.sqs_client.send_message(**send_params)

            message_id = response['MessageId']
            logger.info(f"Sent message to {queue_url} with ID: {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error sending message: {error_code} - {e}")
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
```

```
            str = readLine()

            if str != nil && str?.count == 0 {
                str = nil
            }
        }

        return str!
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt: prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Display a menu of options then request a selection.
    ///
    /// - Parameters:
    ///   - prompt: A prompt string to display before the menu.
    ///   - options: An array of strings giving the menu options.
    ///
    /// - Returns: The index number of the selected option or 0 if no item was
    ///   selected.
    func menuRequest(prompt: String, options: [String]) -> Int {
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }
```

```
        repeat {
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]

    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
```

```
            print("  [\(index)] \(value)")
        }

        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true

        while let ans = askForFilter(message: "Choose a value to apply to the
'tone' attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false

            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
```

```
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]

            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that
name was recently deleted.")
                continue
            }
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws ->
String? {
```

```
        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }

        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
```

```
                                }
                            ]
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }

    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async
throws
                                ->
[SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.
```

```
        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl:
queueUrl)
        )

        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown
ID>") with error: \(message.code ?? "<unknown>") (\(message.message ?? "...")")
            }
        }
```

```
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues
 example.
              In this workflow, you'll create an SNS topic, then create two SQS
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well
 as
              the queue subscriptions. You can then post messages to the topic
 and
              receive the results on the queues.
              \(rowOfStars)\n
              """
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region:
 region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region:
 region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //    to use content-based deduplication instead of requiring a
        //    deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/
N)? ")
        var isContentBasedDeduplication = false

        if isFIFO {
            print("""
                  \(rowOfStars)
                  Because you've chosen to create a FIFO topic, deduplication is
```

```
                  supported.

                  Deduplication IDs are either set in the message or are
 automatically
                  generated from the content using a hash function.

                  If a message is successfully published to an SNS FIFO topic,
 any
                  message published and found to have the same deduplication ID
                  (within a five-minute deduplication interval), is accepted but
                  not delivered.

                  For more information about deduplication, see:
                  https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html.
                  """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
 deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
 create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
        // "ContentBasedDeduplication" attribute.
```

```
        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
            )
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print("""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print("""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
        }
```

```
        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")

        // 5. Attach an AWS IAM policy to the queue using
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
 messages will
                be received in the queue. For information about message
 filtering, see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's
 subscription to the
                topic \(topicName) (Y/N)?
```

```
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil
```

```
        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.
```

```
            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.
\n
                        """
                    )
                }
                publishInput.messageGroupId = stringRequest(prompt: "Enter a
 message group ID for this message: ")

                if !isContentBasedDeduplication {
                    if first {
                        print("""
                              Because you're not using content-based
 deduplication, you
                              must enter a deduplication ID. If other messages
 with the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
 "Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue]
 = [:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
 this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType:
 "String", stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }
```

```
            publishInput.messageAttributes = messageAttributes

            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }

            print("Message published with ID \(messageID).")
            first = false

            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.
```

```
        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Usar o API Gateway para invocar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

Java

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Lambda Java. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3).

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base

em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.
- Use o pacote Requests para enviar solicitações à API REST.
- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar eventos programados para chamar uma função do Lambda

Os exemplos de código a seguir mostram como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

Java

SDK para Java 2.x

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge programado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda é invocada. Neste exemplo, você cria uma função Lambda usando a API de tempo de execução do JavaScript Lambda. Este exemplo invoca AWS serviços diferentes para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento programado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função Lambda o alvo.
- Concede permissão para permitir a EventBridge invocação da função Lambda.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações programadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos de tecnologia sem servidor para o Amazon SNS

Os exemplos de código a seguir mostram como usar o Amazon SNS com AWS SDKs.

Exemplos

- Invocar uma função do Lambda em um acionador do Amazon SNS

Invocar uma função do Lambda em um acionador do Amazon SNS

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");
```

```
            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
```

```
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;
```

```
    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do SNS com o JavaScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Consumindo um evento do SNS com o TypeScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
```

```
    throw err;
  }
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
```

```
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here
```

```
    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SNS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}
```

```
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Exemplos de código para o Amazon SQS usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Simple Queue Service com um kit de desenvolvimento AWS de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon SQS: mais informações sobre o Amazon SQS.
- Referência da API Amazon SQS: detalhes sobre todas as ações disponíveis do Amazon SQS.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon SQS AWS SDKs
  - Olá, Amazon SQS
  - Ações para o Amazon SQS usando AWS SDKs

- Usar AddPermission com uma CLI
- Use ChangeMessageVisibilitycom um AWS SDK ou CLI
- Usar ChangeMessageVisibilityBatch com uma CLI
- Use CreateQueuecom um AWS SDK ou CLI
- Use DeleteMessagecom um AWS SDK ou CLI
- Use DeleteMessageBatchcom um AWS SDK ou CLI
- Use DeleteQueuecom um AWS SDK ou CLI
- Use GetQueueAttributescom um AWS SDK ou CLI
- Use GetQueueUrlcom um AWS SDK ou CLI
- Usar ListDeadLetterSourceQueues com uma CLI
- Use ListQueuescom um AWS SDK ou CLI
- Usar PurgeQueue com uma CLI
- Use ReceiveMessagecom um AWS SDK ou CLI
- Usar RemovePermission com uma CLI
- Use SendMessagecom um AWS SDK ou CLI
- Use SendMessageBatchcom um AWS SDK ou CLI
- Use SetQueueAttributescom um AWS SDK ou CLI

- Cenários para o Amazon SQS usando AWS SDKs
  - Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS
  - Criar uma aplicação de mensageiro com o Step Functions
  - Criar uma aplicação de exploração do Amazon Textract
  - Crie e publique em um tópico FIFO do Amazon SNS usando um AWS SDK
  - Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
  - Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um AWS SDK
  - Receba e processe notificações de eventos do Amazon S3 usando um AWS SDK
  - Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um AWS SDK
  - Envie e receba lotes de mensagens com o Amazon SQS usando um AWS SDK
  - Use o comando AWS Estrutura de processamento de mensagens para.NET para publicar e receber mensagens do Amazon SQS

  - Usar a biblioteca de mensagens Java do Amazon SQS para trabalhar com a interface do Serviço de Mensagens Java (JMS) para o Amazon SQS

- Trabalhe com tags de fila e o Amazon SQS usando um AWS SDK
- Exemplos de tecnologia sem servidor para o Amazon SQS
  - Invocar uma função do Lambda em um trigger do Amazon SQS
  - Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

Exemplos básicos para o uso do Amazon SQS AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Simple Queue Service com SDKs da AWS .

Exemplos

- Olá, Amazon SQS
- Ações para o Amazon SQS usando AWS SDKs
  - Usar AddPermission com uma CLI
  - Use ChangeMessageVisibilitycom um AWS SDK ou CLI
  - Usar ChangeMessageVisibilityBatch com uma CLI
  - Use CreateQueuecom um AWS SDK ou CLI
  - Use DeleteMessagecom um AWS SDK ou CLI
  - Use DeleteMessageBatchcom um AWS SDK ou CLI
  - Use DeleteQueuecom um AWS SDK ou CLI
  - Use GetQueueAttributescom um AWS SDK ou CLI
  - Use GetQueueUrlcom um AWS SDK ou CLI
  - Usar ListDeadLetterSourceQueues com uma CLI
  - Use ListQueuescom um AWS SDK ou CLI
  - Usar PurgeQueue com uma CLI
  - Use ReceiveMessagecom um AWS SDK ou CLI
  - Usar RemovePermission com uma CLI
  - Use SendMessagecom um AWS SDK ou CLI
  - Use SendMessageBatchcom um AWS SDK ou CLI
  - Use SetQueueAttributescom um AWS SDK ou CLI

Olá, Amazon SQS

Os exemplos de código a seguir mostram como começar a usar o Amazon SQS.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.SQS;
using Amazon.SQS.Model;

namespace SQSActions;

public static class HelloSQS
{
    static async Task Main(string[] args)
    {
        var sqsClient = new AmazonSQSClient();

        Console.WriteLine($"Hello Amazon SQS! Following are some of your
 queues:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five queues.
        var response = await sqsClient.ListQueuesAsync(
            new ListQueuesRequest()
            {
                MaxResults = 5
            });

        foreach (var queue in response.QueueUrls)
        {
            Console.WriteLine($"\tQueue Url: {queue}");
            Console.WriteLine();
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sqs)

# Set this project's name.
project("hello_sqs")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()
```

```
# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if(WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif()

add_executable(${PROJECT_NAME}
        hello_sqs.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_sqs.cpp.

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>

/*
 *  A "Hello SQS" starter application that initializes an Amazon Simple Queue
 Service
 *  (Amazon SQS) client and lists the SQS queues in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sqs'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
```

```
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SQS::SQSClient sqsClient(clientConfig);

        Aws::Vector<Aws::String> allQueueUrls;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SQS::Model::ListQueuesRequest request;

            Aws::SQS::Model::ListQueuesOutcome outcome =
 sqsClient.ListQueues(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::String> &pageOfQueueUrls =
 outcome.GetResult().GetQueueUrls();
                if (!pageOfQueueUrls.empty()) {
                    allQueueUrls.insert(allQueueUrls.cend(),
 pageOfQueueUrls.cbegin(),
                                        pageOfQueueUrls.cend());
                }
            }
            else {
                std::cerr << "Error with SQS::ListQueues. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());


        std::cout << "Hello Amazon SQS! You have " << allQueueUrls.size() << "
 queue"
                  << (allQueueUrls.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allQueueUrls.empty()) {
            std::cout << "Here are your queue URLs." << std::endl;
```

```
            for (const Aws::String &queueUrl: allQueueUrls) {
                std::cout << "  * " << queueUrl << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para C++ da API.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
```

```
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 sqsClient := sqs.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the queues for your account.")
 var queueUrls []string
 paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get queues. Here's why: %v\n", err)
   break
  } else {
   queueUrls = append(queueUrls, output.QueueUrls...)
  }
 }
 if len(queueUrls) == 0 {
  fmt.Println("You don't have any queues!")
 } else {
  for _, queueUrl := range queueUrls {
   fmt.Printf("\t%v\n", queueUrl)
  }
 }
}
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                    .flatMap(r -> r.queueUrls().stream())
                    .forEach(content -> System.out.println(" Queue URL: " +
 content.toLowerCase()));

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicializar um cliente Amazon SQS e listar as filas.

```
import { SQSClient, paginateListQueues } from "@aws-sdk/client-sqs";

export const helloSqs = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SQSClient({});

  // You can also use `ListQueuesCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListQueuesCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedQueues = paginateListQueues({ client }, {});
  const queues = [];

  for await (const page of paginatedQueues) {
    if (page.QueueUrls?.length) {
      queues.push(...page.QueueUrls);
    }
  }

  const suffix = queues.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SQS! You have ${queues.length} queue${suffix} in your
 account.`,
  );
  console.log(queues.map((t) => `  * ${t}`).join("\n"));
};
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListQueuesreferência da API AWS SDK for Kotlin.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
import PackageDescription

let package = Package(
    name: "sqs-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sqs-basics",
            dependencies: [
                .product(name: "AWSSQS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
```

```
            path: "Sources")

    ]
)
```

O código-fonte do Swift, `entry.swift`.

```
import ArgumentParser
import AWSClientRuntime
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sqs-basics",
        abstract: """
        This example shows how to list all of your available Amazon SQS queues.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        var queues: [String] = []
        let outputPages = sqsClient.listQueuesPaginated(
            input: ListQueuesInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let urls = output.queueUrls else {
                print("No queues found.")
                return
            }
```

```
            // Iterate over the queue URLs listed on this page, adding them
            // to the `queues` array.

            for queueUrl in urls {
                queues.append(queueUrl)
            }
        }

        print("You have \(queues.count) queues:")
        for queue in queues {
            print("   \(queue)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte ListQueuesa referência da API AWS SDK for Swift.

Ações para o Amazon SQS usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon SQS com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon SQS e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon SQS usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência da API do Amazon Simple Queue Service.

Exemplos

- Usar AddPermission com uma CLI
- Use ChangeMessageVisibilitycom um AWS SDK ou CLI
- Usar ChangeMessageVisibilityBatch com uma CLI
- Use CreateQueuecom um AWS SDK ou CLI
- Use DeleteMessagecom um AWS SDK ou CLI
- Use DeleteMessageBatchcom um AWS SDK ou CLI
- Use DeleteQueuecom um AWS SDK ou CLI
- Use GetQueueAttributescom um AWS SDK ou CLI
- Use GetQueueUrlcom um AWS SDK ou CLI
- Usar ListDeadLetterSourceQueues com uma CLI
- Use ListQueuescom um AWS SDK ou CLI
- Usar PurgeQueue com uma CLI
- Use ReceiveMessagecom um AWS SDK ou CLI
- Usar RemovePermission com uma CLI
- Use SendMessagecom um AWS SDK ou CLI
- Use SendMessageBatchcom um AWS SDK ou CLI
- Use SetQueueAttributescom um AWS SDK ou CLI

Usar **`AddPermission`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddPermission`.

CLI

AWS CLI

Para adicionar uma permissão a uma fila

Este exemplo permite que a AWS conta especificada envie mensagens para a fila especificada.

Comando:

```
aws sqs add-permission --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --label SendMessagesFromMyQueue --aws-
account-ids 12345EXAMPLE --actions SendMessage
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte AddPermissionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo permite que Conta da AWS o especificado envie mensagens da fila especificada.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE
 -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte AddPermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo permite que Conta da AWS o especificado envie mensagens da fila especificada.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE
 -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte AddPermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **ChangeMessageVisibility**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ChangeMessageVisibility`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Changes the visibility timeout of a message in an Amazon Simple Queue Service
//! (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageReceiptHandle: A message receipt handle.
  \param visibilityTimeoutSeconds: Visibility timeout in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::changeMessageVisibility(
        const Aws::String &queue_url,
        const Aws::String &messageReceiptHandle,
        int visibilityTimeoutSeconds,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ChangeMessageVisibilityRequest request;
    request.SetQueueUrl(queue_url);
    request.SetReceiptHandle(messageReceiptHandle);
    request.SetVisibilityTimeout(visibilityTimeoutSeconds);

    auto outcome = sqsClient.ChangeMessageVisibility(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully changed visibility of message " <<
```

```
                    messageReceiptHandle << " from queue " << queue_url <<
 std::endl;
    }
    else {
        std::cout << "Error changing visibility of message from queue "
                  << queue_url << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ChangeMessageVisibilitya Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como alterar a visibilidade do tempo limite de uma mensagem

Este exemplo altera a visibilidade do tempo limite da mensagem especificada para 10 horas (10 horas * 60 minutos * 60 segundos).

Comando:

```
aws sqs change-message-visibility --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --receipt-handle AQEBTpyI...t6HyQg== --
visibility-timeout 36000
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte ChangeMessageVisibilityem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem do Amazon SQS e altere sua visibilidade de tempo limite.

```
import {
  ReceiveMessageCommand,
  ChangeMessageVisibilityCommand,
  SQSClient,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 1,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 1,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  const response = await client.send(
    new ChangeMessageVisibilityCommand({
      QueueUrl: queueUrl,
      ReceiptHandle: Messages[0].ReceiptHandle,
      VisibilityTimeout: 20,
    }),
  );
  console.log(response);
```

```
  return response;
};
```

- Para obter detalhes da API, consulte ChangeMessageVisibilitya Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem do Amazon SQS e altere sua visibilidade de tempo limite.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region to us-west-2
AWS.config.update({ region: "us-west-2" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else {
    // Make sure we have a message
    if (data.Messages != null) {
      var visibilityParams = {
        QueueUrl: queueURL,
        ReceiptHandle: data.Messages[0].ReceiptHandle,
```

```
        VisibilityTimeout: 20, // 20 second timeout
      };
      sqs.changeMessageVisibility(visibilityParams, function (err, data) {
        if (err) {
          console.log("Delete Error", err);
        } else {
          console.log("Timeout Changed", data);
        }
      });
    } else {
      console.log("No messages to change");
    }
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ChangeMessageVisibilitya Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera o tempo limite de visibilidade da mensagem com o identificador de recebimento especificado na fila especificada para 10 horas (1 hora x 60 minutos x 60 segundos = 36.000 segundos).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-
east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -
VisibilityTimeout 36000
```

- Para obter detalhes da API, consulte ChangeMessageVisibilityem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o tempo limite de visibilidade da mensagem com o identificador de recebimento especificado na fila especificada para 10 horas (1 hora x 60 minutos x 60 segundos = 36.000 segundos).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-
east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -
VisibilityTimeout 36000
```

- Para obter detalhes da API, consulte ChangeMessageVisibilityem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

begin
  queue_name = 'my-queue'
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Receive up to 10 messages
  receive_message_result_before = sqs.receive_message({
                                                        queue_url: queue_url,
                                                        max_number_of_messages:
 10
                                                      })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
                                    queue_url: queue_url,
                                    receipt_handle: message.receipt_handle,
```

```
                                    visibility_timeout: 30 # This message will
 not be visible for 30 seconds after first receipt.
                                 })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
                                                       queue_url: queue_url,
                                                       max_number_of_messages: 10
                                                     })

  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
 exist."
end
```

- Para obter detalhes da API, consulte ChangeMessageVisibilitya Referência AWS SDK para Ruby da API.

Usar **`ChangeMessageVisibilityBatch`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ChangeMessageVisibilityBatch`.

CLI

AWS CLI

Para alterar as visibilidades de tempo limite de várias mensagens como um lote

Este exemplo altera as visibilidades do tempo limite das duas mensagens especificadas para 10 horas (10 horas * 60 minutos * 60 segundos).

Comando:

```
aws sqs change-message-visibility-batch --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://change-message-
visibility-batch.json
```

Arquivo de entrada (change-message-visibility-batch.json):

```
[
  {
    "Id": "FirstMessage",
        "ReceiptHandle": "AQEBhz2q...Jf3kaw==",
        "VisibilityTimeout": 36000
  },
  {
    "Id": "SecondMessage",
        "ReceiptHandle": "AQEBkTUH...HifSnw==",
        "VisibilityTimeout": 36000
  }
]
```

Saída:

```
{
  "Successful": [
    {
      "Id": "SecondMessage"
    },
    {
      "Id": "FirstMessage"
    }
  ]
}
```

- Para obter detalhes da API, consulte ChangeMessageVisibilityBatchem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera o tempo limite de visibilidade para duas mensagens com os identificadores de recebimento especificados na fila especificada. O tempo limite de visibilidade da primeira mensagem é alterado para 10 horas (10 horas x 60 minutos x 60 segundos = 36.000 segundos). O tempo limite de visibilidade da segunda mensagem é alterado para 5 horas (5 horas x 60 minutos x 60 segundos = 18.000 segundos).

```
$changeVisibilityRequest1 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
```

```
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
 $changeVisibilityRequest2
```

Saída:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- Para obter detalhes da API, consulte ChangeMessageVisibilityBatchem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o tempo limite de visibilidade para duas mensagens com os identificadores de recebimento especificados na fila especificada. O tempo limite de visibilidade da primeira mensagem é alterado para 10 horas (10 horas x 60 minutos x 60 segundos = 36.000 segundos). O tempo limite de visibilidade da segunda mensagem é alterado para 5 horas (5 horas x 60 minutos x 60 segundos = 18.000 segundos).

```
$changeVisibilityRequest1 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
```

```
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
 $changeVisibilityRequest2
```

Saída:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- Para obter detalhes da API, consulte ChangeMessageVisibilityBatchem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateQueue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateQueue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Publicar mensagens em filas
- Enviar e receber lotes de mensagens
- Utilizar a biblioteca de mensagens Java do Amazon SQS para trabalhar com a interface do JMS

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma fila com um nome específico.

```
    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
```

```
    }
```

Crie uma fila do Amazon SQS e envie uma mensagem para ela.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SQS;
    using Amazon.SQS.Model;

    public class CreateSendExample
    {
        // Specify your AWS Region (an example Region is shown).
        private static readonly string QueueName = "Example_Queue";
        private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
        private static IAmazonSQS client;

        public static async Task Main()
        {
            client = new AmazonSQSClient(ServiceRegion);
            var createQueueResponse = await CreateQueue(client, QueueName);

            string queueUrl = createQueueResponse.QueueUrl;

            Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
            {
                { "Title",   new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
                { "Author",  new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
                { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } },
            };

            string messageBody = "Information about current NY Times fiction
bestseller for week of 12/11/2016.";

            var sendMsgResponse = await SendMessage(client, queueUrl,
messageBody, messageAttributes);
```

```
        }

        /// <summary>
        /// Creates a new Amazon SQS queue using the queue name passed to it
        /// in queueName.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueName">A string representing the name of the queue
        /// to create.</param>
        /// <returns>A CreateQueueResponse that contains information about the
        /// newly created queue.</returns>
        public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS
 client, string queueName)
        {
            var request = new CreateQueueRequest
            {
                QueueName = queueName,
                Attributes = new Dictionary<string, string>
                {
                    { "DelaySeconds", "60" },
                    { "MessageRetentionPeriod", "86400" },
                },
            };

            var response = await client.CreateQueueAsync(request);
            Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

            return response;
        }

        /// <summary>
        /// Sends a message to an SQS queue.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueUrl">The URL of the queue to which to send the
        /// message.</param>
        /// <param name="messageBody">A string representing the body of the
        /// message to be sent to the queue.</param>
        /// <param name="messageAttributes">Attributes for the message to be
        /// sent to the queue.</param>
        /// <returns>A SendMessageResponse object that contains information
        /// about the message that was sent.</returns>
```

```
        public static async Task<SendMessageResponse> SendMessage(
            IAmazonSQS client,
            string queueUrl,
            string messageBody,
            Dictionary<string, MessageAttributeValue> messageAttributes)
        {
            var sendMessageRequest = new SendMessageRequest
            {
                DelaySeconds = 10,
                MessageAttributes = messageAttributes,
                MessageBody = messageBody,
                QueueUrl = queueUrl,
            };

            var response = await client.SendMessageAsync(sendMessageRequest);
            Console.WriteLine($"Sent a message with id : {response.MessageId}");

            return response;
        }
    }
```

- Para obter detalhes da API, consulte CreateQueuea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Create an Amazon Simple Queue Service (Amazon SQS) queue.
```

```
/*!
  \param queueName: An Amazon SQS queue name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::createQueue(const Aws::String &queueName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::CreateQueueRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::CreateQueueOutcome outcome =
 sqsClient.CreateQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created queue " << queueName << " with a queue
 URL "
                  << outcome.GetResult().GetQueueUrl() << "." << std::endl;
    }
    else {
        std::cerr << "Error creating queue " << queueName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateQueuea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar uma fila

Este exemplo cria uma fila com o nome especificado, define o período de retenção da mensagem para 3 dias (3 dias * 24 horas * 60 minutos * 60 segundos) e define a fila de mensagens não entregues da fila especificada com uma contagem máxima de recebimento de 1.000 mensagens.

Comando:

```
aws sqs create-queue --queue-name MyQueue --attributes file://create-queue.json
```

Arquivo de entrada (create-queue.json):

```
{
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":\"1000\"}",
  "MessageRetentionPeriod": "259200"
}
```

Saída:

```
{
  "QueueUrl": "https://queue.amazonaws.com/80398EXAMPLE/MyQueue"
}
```

- Para obter detalhes da API, consulte CreateQueueem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
```

```
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}
```

- Para obter detalhes da API, consulte CreateQueuea Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();
```

```
        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void listQueues(SqsClient sqsClient) {

        System.out.println("\nList Queues");
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
```

```
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
        // List queues with filters
        String namePrefix = "queue";
        ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
                .queueNamePrefix(namePrefix)
                .build();

        ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
        System.out.println("Queue URLs with prefix: " + namePrefix);
        for (String url : listQueuesFilteredResponse.queueUrls()) {
            System.out.println(url);
        }

        System.out.println("\nSend message");
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody("Hello world!")
                    .delaySeconds(10)
                    .build());

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

        System.out.println("\nSend multiple messages");
        try {
```

```
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello
from msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                        .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

        System.out.println("\nReceive messages");
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

        System.out.println("\nChange Message Visibility");
        try {
```

```
            for (Message message : messages) {
                ChangeMessageVisibilityRequest req =
 ChangeMessageVisibilityRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .visibilityTimeout(100)
                        .build();
                sqsClient.changeMessageVisibility(req);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
 List<Message> messages) {
        System.out.println("\nDelete Messages");

        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
 DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateQueuea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma fila padrão do Amazon SQS.

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (sqsQueueName = SQS_QUEUE_NAME) => {
  const command = new CreateQueueCommand({
    QueueName: sqsQueueName,
    Attributes: {
      DelaySeconds: "60",
      MessageRetentionPeriod: "86400",
    },
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Crie uma fila do Amazon SQS com sondagem longa.

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "queue_name";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const response = await client.send(
    new CreateQueueCommand({
```

```
      QueueName: queueName,
      Attributes: {
        // When the wait time for the ReceiveMessage API action is greater than
 0,
        // long polling is in effect. The maximum long polling wait time is 20
        // seconds. Long polling helps reduce the cost of using Amazon SQS by,
        // eliminating the number of empty responses and false empty responses.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        ReceiveMessageWaitTimeSeconds: "20",
      },
    }),
  );
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateQueuea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma fila padrão do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
```

```
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    DelaySeconds: "60",
    MessageRetentionPeriod: "86400",
  },
};

sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

Crie uma fila do Amazon SQS que aguarda a chegada de uma mensagem.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
};

sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateQueuea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val getQueueUrlRequest =
            GetQueueUrlRequest {
                queueName = queueNameVal
            }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

- Para obter detalhes da API, consulte a CreateQueuereferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma fila com o nome especificado.

```
New-SQSQueue -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte CreateQueueem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma fila com o nome especificado.

```
New-SQSQueue -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte CreateQueueem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_queue(name, attributes=None):
```

```
    """
    Creates an Amazon SQS queue.

    :param name: The name of the queue. This is part of the URL assigned to the
 queue.
    :param attributes: The attributes of the queue, such as maximum message size
 or
                       whether it's a FIFO queue.
    :return: A Queue object that contains metadata about the queue and that can
 be used
             to perform queue operations like sending and receiving messages.
    """
    if not attributes:
        attributes = {}

    try:
        queue = sqs.create_queue(QueueName=name, Attributes=attributes)
        logger.info("Created queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't create queue named '%s'.", name)
        raise error
    else:
        return queue
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.
```

```
        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} -
{e}")
            raise
```

- Para obter detalhes da API, consulte a CreateQueueReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# This code example demonstrates how to create a queue in Amazon Simple Queue
 Service (Amazon SQS).

require 'aws-sdk-sqs'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts 'Queue created.'
```

```
  else
    puts 'Queue not created.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateQueuea Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma fila padrão do Amazon SQS.

```
    TRY.
        oo_result = lo_sqs->createqueue( iv_queuename = iv_queue_name ).        "
oo_result is returned for testing purposes. "
        MESSAGE 'SQS queue created.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
queue with the same name.' TYPE 'E'.
      CATCH /aws1/cx_sqsqueuenameexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.
```

Crie uma fila do Amazon SQS que aguarda a chegada de uma mensagem.

```
    TRY.
        DATA lt_attributes TYPE /aws1/cl_sqsqueueattrmap_w=>tt_queueattributemap.
```

```
        DATA ls_attribute TYPE /aws1/
cl_sqsqueueattrmap_w=>ts_queueattributemap_maprow.
        ls_attribute-key = 'ReceiveMessageWaitTimeSeconds'.               " Time
 in seconds for long polling, such as how long the call waits for a message to
 arrive in the queue before returning. "
        ls_attribute-value = NEW /aws1/cl_sqsqueueattrmap_w( iv_value =
 iv_wait_time ).
        INSERT ls_attribute INTO TABLE lt_attributes.
        oo_result = lo_sqs->createqueue(                  " oo_result is returned
 for testing purposes. "
                iv_queuename = iv_queue_name
                it_attributes = lt_attributes ).
        MESSAGE 'SQS queue created.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
 queue with the same name.' TYPE 'E'.
      CATCH /aws1/cx_sqsqueuenameexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateQueuereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.createQueue(
            input: CreateQueueInput(
```

```
                queueName: queueName
            )
        )

        guard let queueUrl = output.queueUrl else {
            print("No queue URL returned.")
            return
        }
```

- Para obter detalhes da API, consulte CreateQueuea referência da API AWS SDK for Swift.

Use **`DeleteMessage`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteMessage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Enviar e receber lotes de mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem de uma fila do Amazon SQS e, em seguida, exclua a mensagem.

```
        public static async Task Main()
        {
            // If the AWS Region you want to use is different from
            // the AWS Region defined for the default user, supply
            // the specify your AWS Region to the client constructor.
            var client = new AmazonSQSClient();
            string queueName = "Example_Queue";
```

```
            var queueUrl = await GetQueueUrl(client, queueName);
            Console.WriteLine($"The SQS queue's URL is {queueUrl}");

            var response = await ReceiveAndDeleteMessage(client, queueUrl);

            Console.WriteLine($"Message: {response.Messages[0]}");
        }

        /// <summary>
        /// Retrieve the queue URL for the queue named in the queueName
        /// property using the client object.
        /// </summary>
        /// <param name="client">The Amazon SQS client used to retrieve the
        /// queue URL.</param>
        /// <param name="queueName">A string representing  name of the queue
        /// for which to retrieve the URL.</param>
        /// <returns>The URL of the queue.</returns>
        public static async Task<string> GetQueueUrl(IAmazonSQS client, string
 queueName)
        {
            var request = new GetQueueUrlRequest
            {
                QueueName = queueName,
            };

            GetQueueUrlResponse response = await
 client.GetQueueUrlAsync(request);
            return response.QueueUrl;
        }

        /// <summary>
        /// Retrieves the message from the quque at the URL passed in the
        /// queueURL parameters using the client.
        /// </summary>
        /// <param name="client">The SQS client used to retrieve a message.</
param>
        /// <param name="queueUrl">The URL of the queue from which to retrieve
        /// a message.</param>
        /// <returns>The response from the call to ReceiveMessageAsync.</returns>
        public static async Task<ReceiveMessageResponse>
 ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
        {
            // Receive a single message from the queue.
            var receiveMessageRequest = new ReceiveMessageRequest
```

```
            {
                AttributeNames = { "SentTimestamp" },
                MaxNumberOfMessages = 1,
                MessageAttributeNames = { "All" },
                QueueUrl = queueUrl,
                VisibilityTimeout = 0,
                WaitTimeSeconds = 0,
            };

            var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

            // Delete the received message from the queue.
            var deleteMessageRequest = new DeleteMessageRequest
            {
                QueueUrl = queueUrl,
                ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
            };

            await client.DeleteMessageAsync(deleteMessageRequest);

            return receiveMessageResponse;
        }
    }
```

- Para obter detalhes da API, consulte DeleteMessagea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Delete a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageReceiptHandle: A message receipt handle.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteMessage(const Aws::String &queueUrl,
                                const Aws::String &messageReceiptHandle,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetReceiptHandle(messageReceiptHandle);

    const Aws::SQS::Model::DeleteMessageOutcome outcome =
 sqsClient.DeleteMessage(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted message from queue " << queueUrl
                  << std::endl;
    }
    else {
        std::cerr << "Error deleting message from queue " << queueUrl << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteMessagea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir uma mensagem

Este exemplo exclui a mensagem especificada.

Comando:

```
aws sqs delete-message --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --receipt-handle AQEBRXTo...q2doVA==
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte DeleteMessageem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- Para obter detalhes da API, consulte DeleteMessagea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receber e excluir mensagens do Amazon SQS.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
```

```
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- Para obter detalhes da API, consulte DeleteMessagea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receber e excluir mensagens do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else if (data.Messages) {
    var deleteParams = {
      QueueUrl: queueURL,
      ReceiptHandle: data.Messages[0].ReceiptHandle,
    };
    sqs.deleteMessage(deleteParams, function (err, data) {
      if (err) {
        console.log("Delete Error", err);
      } else {
        console.log("Message Deleted", data);
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteMessagea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Para obter detalhes da API, consulte a DeleteMessagereferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a mensagem com os identificadores de recebimento especificados na fila especificada.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -ReceiptHandle AQEBd329...v6gl8Q==
```

- Para obter detalhes da API, consulte DeleteMessageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a mensagem com os identificadores de recebimento especificados na fila especificada.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -ReceiptHandle AQEBd329...v6gl8Q==
```

- Para obter detalhes da API, consulte DeleteMessageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_message(message):
    """
    Delete a message from a queue. Clients must delete messages after they
    are received and processed to remove them from the queue.

    :param message: The message to delete. The message's queue URL is contained
 in
```

```
                    the message's metadata.
    :return: None
    """
    try:
        message.delete()
        logger.info("Deleted message: %s", message.message_id)
    except ClientError as error:
        logger.exception("Couldn't delete message: %s", message.message_id)
        raise error
```

- Para obter detalhes da API, consulte a DeleteMessageReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_sqs->deletemessage(
          iv_queueurl = iv_queue_url
          iv_receipthandle = iv_receipt_handle ).
       MESSAGE 'Message deleted from SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsinvalididformat.
       MESSAGE 'The specified receipt handle is not valid.' TYPE 'E'.
     CATCH /aws1/cx_sqsreceipthandleisinv.
       MESSAGE 'The specified receipt handle is not valid for the current
version.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteMessagereferência da API AWS SDK for SAP ABAP.

Use **`DeleteMessageBatch`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteMessageBatch`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Processar notificações de eventos do S3
- Publicar mensagens em filas
- Enviar e receber lotes de mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
```

```
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }
```

- Para obter detalhes da API, consulte DeleteMessageBatcha Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
```

```
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

- Para obter detalhes da API, consulte DeleteMessageBatcha Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como excluir várias mensagens como um lote

Este exemplo exclui as mensagens especificadas.

Comando:

```
aws sqs delete-message-batch --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://delete-message-batch.json
```

Arquivo de entrada (delete-message-batch.json):

```
[
  {
```

```
        "Id": "FirstMessage",
        "ReceiptHandle": "AQEB1mgl...Z4GuLw=="
  },
  {
    "Id": "SecondMessage",
        "ReceiptHandle": "AQEBLsYM...VQubAA=="
  }
]
```

Saída:

```
{
  "Successful": [
    {
      "Id": "FirstMessage"
    },
    {
      "Id": "SecondMessage"
    }
  ]
}
```

- Para obter detalhes da API, consulte DeleteMessageBatchem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteMessageBatcha Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
```

```
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- Para obter detalhes da API, consulte DeleteMessageBatcha Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui duas mensagens com os identificadores de recebimento especificados na fila especificada.

```
$deleteMessageRequest1 = New-Object
 Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object
 Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBqOVY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $deleteMessageRequest1,
 $deleteMessageRequest2
```

Saída:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- Para obter detalhes da API, consulte DeleteMessageBatchem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui duas mensagens com os identificadores de recebimento especificados na fila especificada.

```
$deleteMessageRequest1 = New-Object
 Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object
 Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBqOVY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $deleteMessageRequest1,
 $deleteMessageRequest2
```

Saída:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- Para obter detalhes da API, consulte DeleteMessageBatchem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and
 failed
             message deletions.
    """
    try:
        entries = [
            {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
            for ind, msg in enumerate(messages)
        ]
        response = queue.delete_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info("Deleted %s",
 messages[int(msg_meta["Id"])].receipt_handle)
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Could not delete %s",
 messages[int(msg_meta["Id"])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
 bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
```

```
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a DeleteMessageBatchReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sqs->deletemessagebatch(       " oo_result is returned for
testing purposes. "
          iv_queueurl = iv_queue_url
          it_entries = it_entries ).
       MESSAGE 'Messages deleted from SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsbtcentidsnotdist00.
       MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
     CATCH /aws1/cx_sqsemptybatchrequest.
       MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
     CATCH /aws1/cx_sqsinvbatchentryid.
       MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE
'E'.
     CATCH /aws1/cx_sqstoomanyentriesin00.
       MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteMessageBatchreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        // Create the list of message entries.

        var entries: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []
        var messageNumber = 1
```

```
        for handle in handles {
            let entry = SQSClientTypes.DeleteMessageBatchRequestEntry(
                id: "\(messageNumber)",
                receiptHandle: handle
            )
            entries.append(entry)
            messageNumber += 1
        }

        // Delete the messages.

        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(
                entries: entries,
                queueUrl: queue
            )
        )

        // Get the lists of failed and successful deletions from the output.

        guard let failedEntries = output.failed else {
            print("Failed deletion list is missing!")
            return
        }
        guard let successfulEntries = output.successful else {
            print("Successful deletion list is missing!")
            return
        }

        // Display a list of the failed deletions along with their
        // corresponding explanation messages.

        if failedEntries.count != 0 {
            print("Failed deletions:")

            for entry in failedEntries {
                print("Message #\(entry.id ?? "<unknown>") failed:
\(entry.message ?? "<unknown>")")
            }
        } else {
            print("No failed deletions.")
        }
```

```
        // Output a list of the message numbers that were successfully deleted.

        if successfulEntries.count != 0 {
            var successes = ""

            for entry in successfulEntries {
                if successes.count == 0 {
                    successes = entry.id ?? "<unknown>"
                } else {
                    successes = "\(successes), \(entry.id ?? "<unknown>")"
                }
            }
            print("Succeeded: ", successes)
        } else {
            print("No successful deletions.")
        }
```

- Para obter detalhes da API, consulte DeleteMessageBatcha referência da API AWS SDK for Swift.

Use **`DeleteQueue`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteQueue`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Publicar mensagens em filas
- Enviar e receber lotes de mensagens
- Utilizar a biblioteca de mensagens Java do Amazon SQS para trabalhar com a interface do JMS

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma fila usando seu URL.

```
/// <summary>
/// Delete a queue by its URL.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteQueueByUrl(string queueUrl)
{
    var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
        new DeleteQueueRequest()
        {
            QueueUrl = queueUrl
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DeleteQueuea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Delete an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueURL: An Amazon SQS queue URL.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteQueue(const Aws::String &queueURL,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::SQS::Model::DeleteQueueRequest request;
    request.SetQueueUrl(queueURL);

    const Aws::SQS::Model::DeleteQueueOutcome outcome =
 sqsClient.DeleteQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted queue with url " << queueURL <<
                  std::endl;
    }
    else {
        std::cerr << "Error deleting queue " << queueURL << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteQueuea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir uma fila

Este exemplo exclui a fila especificada.

Comando:

```
aws sqs delete-queue --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyNewerQueue
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte DeleteQueueem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
```

```
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteQueuea Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName>
```

```
                Where:
                   queueName - The name of the Amazon SQS queue to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        SqsClient sqs = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        deleteSQSQueue(sqs, queueName);
        sqs.close();
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteQueuea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma fila do Amazon SQS.

```
import { DeleteQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "test-queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new DeleteQueueCommand({ QueueUrl: queueUrl });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteQueuea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueUrl: "SQS_QUEUE_URL",
};

sqs.deleteQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteQueuea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
```

```
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Para obter detalhes da API, consulte a DeleteQueuereferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a fila especificada.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Para obter detalhes da API, consulte DeleteQueueem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a fila especificada.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Para obter detalhes da API, consulte DeleteQueueem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def remove_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.
```

```
        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise
```

- Para obter detalhes da API, consulte a DeleteQueueReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

- Para obter detalhes da API, consulte DeleteQueuea Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sqs->deletequeue( iv_queueurl = iv_queue_url ).
        MESSAGE 'SQS queue deleted' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteQueuereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        do {
            _ = try await sqsClient.deleteQueue(
                input: DeleteQueueInput(
                    queueUrl: queueUrl
                )
            )
        } catch _ as AWSSQS.QueueDoesNotExist {
            print("Error: The specified queue doesn't exist.")
            return
        }
```

- Para obter detalhes da API, consulte DeleteQueuea referência da API AWS SDK for Swift.

Use **`GetQueueAttributes`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetQueueAttributes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Processar notificações de eventos do S3

- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }
```

- Para obter detalhes da API, consulte GetQueueAttributesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }

            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
            }
```

- Para obter detalhes da API, consulte GetQueueAttributesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como obter os atributos de uma fila

Este exemplo obtém todos os atributos da fila especificada.

Comando:

```
aws sqs get-queue-attributes --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names All
```

Saída:

```
{
  "Attributes": {
    "ApproximateNumberOfMessagesNotVisible": "0",
    "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":1000}",
    "MessageRetentionPeriod": "345600",
    "ApproximateNumberOfMessagesDelayed": "0",
    "MaximumMessageSize": "262144",
    "CreatedTimestamp": "1442426968",
    "ApproximateNumberOfMessages": "0",
    "ReceiveMessageWaitTimeSeconds": "0",
    "DelaySeconds": "0",
    "VisibilityTimeout": "30",
    "LastModifiedTimestamp": "1442426968",
    "QueueArn": "arn:aws:sqs:us-east-1:80398EXAMPLE:MyNewQueue"
  }
}
```

Este exemplo obtém somente os atributos especificados de tempo limite de visibilidade e de tamanho máximo da mensagem da fila.

Comando:

```
aws sqs get-queue-attributes --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyNewQueue --attribute-
names MaximumMessageSize VisibilityTimeout
```

Saída:

```
{
  "Attributes": {
    "VisibilityTimeout": "30",
    "MaximumMessageSize": "262144"
  }
}
```

- Para obter detalhes da API, consulte GetQueueAttributesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}
```

- Para obter detalhes da API, consulte GetQueueAttributesa Referência AWS SDK para Go da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { GetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const getQueueAttributes = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new GetQueueAttributesCommand({
    QueueUrl: queueUrl,
    AttributeNames: ["DelaySeconds"],
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '747a1192-c334-5682-a508-4cd5e8dc4e79',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: { DelaySeconds: '1' }
  // }
  return response;
};
```

- Para obter detalhes da API, consulte GetQueueAttributesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todos os atributos da fila especificada.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14

 495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                          398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                          arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                          [ApproximateNumberOfMessagesNotVisible,
 0], [ApproximateNumberOfMessagesDelayed, 0]...}
```

Exemplo 2: esse exemplo lista separadamente somente os atributos especificados da fila especificada.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -
QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14

 495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                          398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                          arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- Para obter detalhes da API, consulte GetQueueAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os atributos da fila especificada.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout                      : 30
DelaySeconds                           : 0
MaximumMessageSize                     : 262144
MessageRetentionPeriod                 : 345600
ApproximateNumberOfMessages            : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed     : 0
CreatedTimestamp                       : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                  : 12/29/2015 2:23:17 PM
QueueARN                               : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                                 : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14

 495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                          398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                          arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                             : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                          [ApproximateNumberOfMessagesNotVisible,
 0], [ApproximateNumberOfMessagesDelayed, 0]...}
```

Exemplo 2: esse exemplo lista separadamente somente os atributos especificados da fila especificada.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -
QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout                      : 30
DelaySeconds                           : 0
```

```
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14

 495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                          398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                          arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- Para obter detalhes da API, consulte GetQueueAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""
```

```
    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a GetQueueAttributesReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [
                    .approximatenumberofmessages,
                    .maximummessagesize
                ],
                queueUrl: url
            )
        )

        guard let attributes = output.attributes else {
            print("No queue attributes returned.")
            return
        }

        for (attr, value) in attributes {
            switch(attr) {
            case "ApproximateNumberOfMessages":
                print("Approximate message count: \(value)")
            case "MaximumMessageSize":
                print("Maximum message size: \(value)kB")
```

```
            default:
                continue
            }
        }
```

- Para obter detalhes da API, consulte GetQueueAttributesa referência da API AWS SDK for Swift.

Use **`GetQueueUrl`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetQueueUrl`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SQS;
    using Amazon.SQS.Model;

    public class GetQueueUrl
    {
        /// <summary>
        /// Initializes the Amazon SQS client object and then calls the
        /// GetQueueUrlAsync method to retrieve the URL of an Amazon SQS
        /// queue.
        /// </summary>
        public static async Task Main()
        {
            // If the Amazon SQS message queue is not in the same AWS Region as
your
            // default user, you need to provide the AWS Region as a parameter to
the
```

```
            // client constructor.
            var client = new AmazonSQSClient();

            string queueName = "New-Example-Queue";

            try
            {
                var response = await client.GetQueueUrlAsync(queueName);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    Console.WriteLine($"The URL for {queueName} is:
{response.QueueUrl}");
                }
            }
            catch (QueueDoesNotExistException ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine($"The queue {queueName} was not found.");
            }
        }
    }
```

- Para obter detalhes da API, consulte GetQueueUrla Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
//! Get the URL for an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueName: An Amazon SQS queue name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::getQueueUrl(const Aws::String &queueName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::GetQueueUrlRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::GetQueueUrlOutcome outcome =
 sqsClient.GetQueueUrl(request);
    if (outcome.IsSuccess()) {
        std::cout << "Queue " << queueName << " has url " <<
                  outcome.GetResult().GetQueueUrl() << std::endl;
    }
    else {
        std::cerr << "Error getting url for queue " << queueName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetQueueUrla Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como obter um URL de fila

Este exemplo obtém o URL da fila especificada.

Comando:

```
aws sqs get-queue-url --queue-name MyQueue
```

Saída:

```
{
  "QueueUrl": "https://queue.amazonaws.com/80398EXAMPLE/MyQueue"
}
```

- Para obter detalhes da API, consulte GetQueueUrlem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
```

- Para obter detalhes da API, consulte GetQueueUrla Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha o URL para uma fila do Amazon SQS.

```
import { GetQueueUrlCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const command = new GetQueueUrlCommand({ QueueName: queueName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetQueueUrla Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha o URL para uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
};

sqs.getQueueUrl(params, function (err, data) {
```

```
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetQueueUrla Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista o URL da fila com o nome especificado.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte GetQueueUrlem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista o URL da fila com o nome especificado.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte GetQueueUrlem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_queue(name):
    """
    Gets an SQS queue by name.

    :param name: The name that was used to create the queue.
    :return: A Queue object.
    """
    try:
        queue = sqs.get_queue_by_name(QueueName=name)
        logger.info("Got queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't get queue named %s.", name)
        raise error
    else:
        return queue
```

- Para obter detalhes da API, consulte a GetQueueUrlReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sqs->getqueueurl( iv_queuename = iv_queue_name ).        "
 oo_result is returned for testing purposes. "
        MESSAGE 'Queue URL retrieved.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedoesnotexist.
        MESSAGE 'The requested queue does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetQueueUrlreferência da API AWS SDK for SAP ABAP.

Usar **`ListDeadLetterSourceQueues`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListDeadLetterSourceQueues`.

CLI

AWS CLI

Para listar filas de origem de letras mortas

Este exemplo lista as filas associadas à fila de origem de mensagens não entregues especificada.

Comando:

```
aws sqs list-dead-letter-source-queues --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Saída:

```
{
  "queueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue"
  ]
}
```

- Para obter detalhes da API, consulte ListDeadLetterSourceQueuesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista os URLs de todas as filas que dependem da fila especificada como a fila de mensagens não entregues.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Para obter detalhes da API, consulte ListDeadLetterSourceQueuesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os URLs de todas as filas que dependem da fila especificada como a fila de mensagens não entregues.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Para obter detalhes da API, consulte ListDeadLetterSourceQueuesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListQueues`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListQueues`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! List the Amazon Simple Queue Service (Amazon SQS) queues within an AWS
 account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SQS::listQueues(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ListQueuesRequest listQueuesRequest;

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::String> allQueueUrls;

    do {
        if (!nextToken.empty()) {
            listQueuesRequest.SetNextToken(nextToken);
        }
        const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
                listQueuesRequest);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &queueUrls =
 outcome.GetResult().GetQueueUrls();
            allQueueUrls.insert(allQueueUrls.end(),
                                queueUrls.begin(),
                                queueUrls.end());
```

```
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing queues: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." <<
 std::endl;
    for (const auto &iter: allQueueUrls) {
        std::cout << " " << iter << std::endl;
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para listar filas

Este exemplo lista todas as filas.

Comando:

```
aws sqs list-queues
```

Saída:

```
{
  "QueueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue",
```

```
    "https://queue.amazonaws.com/80398EXAMPLE/TestQueue1",
        "https://queue.amazonaws.com/80398EXAMPLE/TestQueue2"
  ]
}
```

Este exemplo lista somente as filas que começam com “My”.

Comando:

```
aws sqs list-queues --queue-name-prefix My
```

Saída:

```
{
  "QueueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue"
  ]
}
```

- Para obter detalhes da API, consulte ListQueuesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 sqsClient := sqs.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the queues for your account.")
 var queueUrls []string
 paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get queues. Here's why: %v\n", err)
   break
  } else {
   queueUrls = append(queueUrls, output.QueueUrls...)
  }
 }
 if len(queueUrls) == 0 {
  fmt.Println("You don't have any queues!")
 } else {
  for _, queueUrl := range queueUrls {
   fmt.Printf("\t%v\n", queueUrl)
  }
 }
}
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar filas do Amazon SQS.

```
import { paginateListQueues, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});

export const main = async () => {
  const paginatedListQueues = paginateListQueues({ client }, {});

  /** @type {string[]} */
  const urls = [];
  for await (const page of paginatedListQueues) {
    const nextUrls = page.QueueUrls?.filter((qurl) => !!qurl) || [];
    urls.push(...nextUrls);
    for (const url of urls) {
      console.log(url);
    }
  }

  return urls;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar filas do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {};

sqs.listQueues(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrls);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
```

```
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- Para obter detalhes da API, consulte a ListQueuesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as filas.

```
Get-SQSQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Exemplo 2: esse exemplo lista qualquer fila que comece com o nome especificado.

```
Get-SQSQueue -QueueNamePrefix My
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Para obter detalhes da API, consulte ListQueuesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as filas.

```
Get-SQSQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Exemplo 2: esse exemplo lista qualquer fila que comece com o nome especificado.

```
Get-SQSQueue -QueueNamePrefix My
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Para obter detalhes da API, consulte ListQueuesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_queues(prefix=None):
    """
    Gets a list of SQS queues. When a prefix is specified, only queues with names
    that start with the prefix are returned.

    :param prefix: The prefix used to restrict the list of returned queues.
```

```
    :return: A list of Queue objects.
    """
    if prefix:
        queue_iter = sqs.queues.filter(QueueNamePrefix=prefix)
    else:
        queue_iter = sqs.queues.all()
    queues = list(queue_iter)
    if queues:
        logger.info("Got queues: %s", ", ".join([q.url for q in queues]))
    else:
        logger.warning("No queues found.")
    return queues
```

- Para obter detalhes da API, consulte a ListQueuesReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-west-2'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
```

```
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: ['All']
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end
rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'

  sqs_client = Aws::SQS::Client.new(region: region)

  puts 'Listing available queue URLs...'
  list_queue_urls(sqs_client)

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"
```

```
  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end
```

- Para obter detalhes da API, consulte ListQueuesa Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recuperar a primeira fila do Amazon SQS listada na região.

```
async fn find_first_queue(client: &Client) -> Result<String, Error> {
    let queues = client.list_queues().send().await?;
    let queue_urls = queues.queue_urls();
    Ok(queue_urls
        .first()
        .expect("No queues in this account and Region. Create a queue to
 proceed.")
        .to_string())
}
```

- Para obter detalhes da API, consulte a ListQueuesreferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sqs->listqueues( ).        " oo_result is returned for
testing purposes. "
       MESSAGE 'Retrieved list of queues.' TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListQueuesreferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        var queues: [String] = []
        let outputPages = sqsClient.listQueuesPaginated(
            input: ListQueuesInput()
        )
```

```
        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let urls = output.queueUrls else {
                print("No queues found.")
                return
            }

            // Iterate over the queue URLs listed on this page, adding them
            // to the `queues` array.

            for queueUrl in urls {
                queues.append(queueUrl)
            }
        }
```

- Para obter detalhes da API, consulte ListQueuesa referência da API AWS SDK for Swift.

Usar **`PurgeQueue`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PurgeQueue`.

CLI

AWS CLI

Para limpar uma fila

Este exemplo exclui todas as mensagens na fila especificada.

Comando:

```
aws sqs purge-queue --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyNewQueue
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte PurgeQueueem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui todas as mensagens da fila especificada.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte PurgeQueueem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui todas as mensagens da fila especificada.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte PurgeQueueem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ReceiveMessage`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ReceiveMessage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Gerenciar mensagens grandes usando o S3
- Processar notificações de eventos do S3
- Publicar mensagens em filas
- Enviar e receber lotes de mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba mensagens de uma fila usando seu URL.

```
    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }
```

Receba uma mensagem de uma fila do Amazon SQS e, em seguida, exclua a mensagem.

```
        public static async Task Main()
        {
            // If the AWS Region you want to use is different from
            // the AWS Region defined for the default user, supply
            // the specify your AWS Region to the client constructor.
```

```
            var client = new AmazonSQSClient();
            string queueName = "Example_Queue";

            var queueUrl = await GetQueueUrl(client, queueName);
            Console.WriteLine($"The SQS queue's URL is {queueUrl}");

            var response = await ReceiveAndDeleteMessage(client, queueUrl);

            Console.WriteLine($"Message: {response.Messages[0]}");
        }

        /// <summary>
        /// Retrieve the queue URL for the queue named in the queueName
        /// property using the client object.
        /// </summary>
        /// <param name="client">The Amazon SQS client used to retrieve the
        /// queue URL.</param>
        /// <param name="queueName">A string representing  name of the queue
        /// for which to retrieve the URL.</param>
        /// <returns>The URL of the queue.</returns>
        public static async Task<string> GetQueueUrl(IAmazonSQS client, string
 queueName)
        {
            var request = new GetQueueUrlRequest
            {
                QueueName = queueName,
            };

            GetQueueUrlResponse response = await
 client.GetQueueUrlAsync(request);
            return response.QueueUrl;
        }

        /// <summary>
        /// Retrieves the message from the quque at the URL passed in the
        /// queueURL parameters using the client.
        /// </summary>
        /// <param name="client">The SQS client used to retrieve a message.</
param>
        /// <param name="queueUrl">The URL of the queue from which to retrieve
        /// a message.</param>
        /// <returns>The response from the call to ReceiveMessageAsync.</returns>
        public static async Task<ReceiveMessageResponse>
 ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
```

```
        {
            // Receive a single message from the queue.
            var receiveMessageRequest = new ReceiveMessageRequest
            {
                AttributeNames = { "SentTimestamp" },
                MaxNumberOfMessages = 1,
                MessageAttributeNames = { "All" },
                QueueUrl = queueUrl,
                VisibilityTimeout = 0,
                WaitTimeSeconds = 0,
            };

            var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

            // Delete the received message from the queue.
            var deleteMessageRequest = new DeleteMessageRequest
            {
                QueueUrl = queueUrl,
                ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
            };

            await client.DeleteMessageAsync(deleteMessageRequest);

            return receiveMessageResponse;
        }
    }
```

- Para obter detalhes da API, consulte ReceiveMessagea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Receive a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::receiveMessage(const Aws::String &queueUrl,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ReceiveMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMaxNumberOfMessages(1);

    const Aws::SQS::Model::ReceiveMessageOutcome outcome =
 sqsClient.ReceiveMessage(
            request);
    if (outcome.IsSuccess()) {

        const Aws::Vector<Aws::SQS::Model::Message> &messages =
                outcome.GetResult().GetMessages();
        if (!messages.empty()) {
            const Aws::SQS::Model::Message &message = messages[0];
            std::cout << "Received message:" << std::endl;
            std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
            std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
 std::endl;
            std::cout << "  Body: " << message.GetBody() << std::endl <<
 std::endl;
        }
        else {
            std::cout << "No messages received from queue " << queueUrl <<
                      std::endl;

        }
    }
    else {
        std::cerr << "Error receiving message from queue " << queueUrl << ": "
```

```
                    << outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ReceiveMessagea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como receber uma mensagem

Este exemplo recebe até 10 mensagens disponíveis e retorna todos os atributos disponíveis.

Comando:

```
aws sqs receive-message --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names All --message-
attribute-names All --max-number-of-messages 10
```

Saída:

```
{
  "Messages": [
    {
      "Body": "My first message.",
      "ReceiptHandle": "AQEBzbVv...fqNzFw==",
      "MD5OfBody": "1000f835...a35411fa",
      "MD5OfMessageAttributes": "9424c491...26bc3ae7",
      "MessageId": "d6790f8d-d575-4f01-bc51-40122EXAMPLE",
      "Attributes": {
        "ApproximateFirstReceiveTimestamp": "1442428276921",
        "SenderId": "AIDAIAZKMSNQ7TEXAMPLE",
        "ApproximateReceiveCount": "5",
        "SentTimestamp": "1442428276921"
      },
      "MessageAttributes": {
        "PostalCode": {
          "DataType": "String",
```

```
          "StringValue": "ABC123"
        },
        "City": {
          "DataType": "String",
          "StringValue": "Any City"
        }
      }
    }
  ]
}
```

Este exemplo recebe a próxima mensagem disponível, retornando somente os SentTimestamp atributos SenderId e, bem como o atributo da PostalCode mensagem.

Comando:

```
aws sqs receive-message --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-
names SenderId SentTimestamp --message-attribute-names PostalCode
```

Saída:

```
{
  "Messages": [
    {
      "Body": "My first message.",
      "ReceiptHandle": "AQEB6nR4...HzlvZQ==",
      "MD5OfBody": "1000f835...a35411fa",
      "MD5OfMessageAttributes": "b8e89563...e088e74f",
      "MessageId": "d6790f8d-d575-4f01-bc51-40122EXAMPLE",
      "Attributes": {
        "SenderId": "AIDAIAZKMSNQ7TEXAMPLE",
        "SentTimestamp": "1442428276921"
      },
      "MessageAttributes": {
        "PostalCode": {
          "DataType": "String",
          "StringValue": "ABC123"
        }
      }
    }
  ]
```

```
}
```

- Para obter detalhes da API, consulte ReceiveMessageem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
```

```
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}
```

- Para obter detalhes da API, consulte ReceiveMessagea Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        try {
            ReceiveMessageRequest receiveMessageRequest =
 ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

- Para obter detalhes da API, consulte ReceiveMessagea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem de uma fila do Amazon SQS.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);
```

```
  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
import { ReceiveMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new ReceiveMessageCommand({
    AttributeNames: ["SentTimestamp"],
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ["All"],
    QueueUrl: queueUrl,
    // The duration (in seconds) for which the call waits for a message
    // to arrive in the queue before returning. If a message is available,
    // the call returns sooner than WaitTimeSeconds. If no messages are
    // available and the wait time expires, the call returns successfully
```

```
    // with an empty list of messages.
    // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/
API_ReceiveMessage.html#API_ReceiveMessage_RequestSyntax
    WaitTimeSeconds: 20,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte ReceiveMessagea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  WaitTimeSeconds: 20,
};
```

```
sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ReceiveMessagea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- Para obter detalhes da API, consulte a ReceiveMessagereferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista as informações de até as próximas 10 mensagens a serem recebidas na fila especificada. As informações conterão valores para os atributos de mensagem especificados, se existirem.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
Attributes             : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
 1451495923744]}
Body                   : Information about John Doe's grade.
MD5OfBody              : ea572796e3c231f974fe75d89EXAMPLE
MD5OfMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==
```

- Para obter detalhes da API, consulte ReceiveMessageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as informações de até as próximas 10 mensagens a serem recebidas na fila especificada. As informações conterão valores para os atributos de mensagem especificados, se existirem.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
Attributes             : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
 1451495923744]}
Body                   : Information about John Doe's grade.
MD5OfBody              : ea572796e3c231f974fe75d89EXAMPLE
MD5OfMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==
```

- Para obter detalhes da API, consulte ReceiveMessageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual
 number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning.
 When
                      this number is greater than zero, long polling is used.
 This
                      can result in reduced costs and fewer false empty
 responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
```

```
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return messages
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
 List[Dict[str, Any]]:
        """
```

```
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a ReceiveMessageReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     10
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)
  if max_number_of_messages > 10
    puts 'Maximum number of messages to receive must be 10 or less. ' \
      'Stopping program.'
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )

  if response.messages.count.zero?
    puts 'No messages to receive, or all messages have already ' \
      'been previously received.'
    return
  end

  response.messages.each do |message|
    puts '-' * 20
    puts "Message body: #{message.body}"
    puts "Message ID:   #{message.message_id}"
  end
rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end
```

```
# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte ReceiveMessagea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn receive(client: &Client, queue_url: &String) -> Result<(), Error> {
    let rcv_message_output =
 client.receive_message().queue_url(queue_url).send().await?;
```

```
    println!("Messages from queue with url: {}", queue_url);

    for message in rcv_message_output.messages.unwrap_or_default() {
        println!("Got the message: {:#?}", message);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ReceiveMessagereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem de uma fila do Amazon SQS.

```
    TRY.
        oo_result = lo_sqs->receivemessage( iv_queueurl = iv_queue_url ).    "
oo_result is returned for testing purposes. "
        DATA(lt_messages) = oo_result->get_messages( ).
        MESSAGE 'Message received from SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsoverlimit.
        MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
    ENDTRY.
```

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
    TRY.
        oo_result = lo_sqs->receivemessage(           " oo_result is returned for
testing purposes. "
```

```
                iv_queueurl = iv_queue_url
                iv_waittimeseconds = iv_wait_time ).    " Time in seconds for
long polling, such as how long the call waits for a message to arrive in the
queue before returning. " ).
       DATA(lt_messages) = oo_result->get_messages( ).
       MESSAGE 'Message received from SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsoverlimit.
       MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ReceiveMessagereferência da API AWS SDK for SAP ABAP.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: maxMessages,
                queueUrl: url
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return
        }
```

```
        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print(message.body ?? "<body missing>")
            print("---")
        }
```

- Para obter detalhes da API, consulte ReceiveMessagea referência da API AWS SDK for Swift.

Usar **`RemovePermission`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemovePermission`.

CLI

AWS CLI

Para remover uma permissão

Este exemplo remove a permissão com o rótulo especificado da fila especificada.

Comando:

```
aws sqs remove-permission --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --label SendMessagesFromMyQueue
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte RemovePermissionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove as configurações permissão com o rótulo especificado da fila especificada.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte RemovePermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove as configurações permissão com o rótulo especificado da fila especificada.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte RemovePermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`SendMessage`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SendMessage`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Gerenciar mensagens grandes usando o S3
- Enviar e receber lotes de mensagens

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma fila do Amazon SQS e envie uma mensagem para ela.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SQS;
    using Amazon.SQS.Model;

    public class CreateSendExample
    {
        // Specify your AWS Region (an example Region is shown).
        private static readonly string QueueName = "Example_Queue";
        private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
        private static IAmazonSQS client;

        public static async Task Main()
        {
            client = new AmazonSQSClient(ServiceRegion);
            var createQueueResponse = await CreateQueue(client, QueueName);

            string queueUrl = createQueueResponse.QueueUrl;

            Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
            {
                { "Title",   new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
                { "Author",  new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
                { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } },
```

```
            };

            string messageBody = "Information about current NY Times fiction
 bestseller for week of 12/11/2016.";

            var sendMsgResponse = await SendMessage(client, queueUrl,
 messageBody, messageAttributes);
        }

        /// <summary>
        /// Creates a new Amazon SQS queue using the queue name passed to it
        /// in queueName.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueName">A string representing the name of the queue
        /// to create.</param>
        /// <returns>A CreateQueueResponse that contains information about the
        /// newly created queue.</returns>
        public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS
 client, string queueName)
        {
            var request = new CreateQueueRequest
            {
                QueueName = queueName,
                Attributes = new Dictionary<string, string>
                {
                    { "DelaySeconds", "60" },
                    { "MessageRetentionPeriod", "86400" },
                },
            };

            var response = await client.CreateQueueAsync(request);
            Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

            return response;
        }

        /// <summary>
        /// Sends a message to an SQS queue.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueUrl">The URL of the queue to which to send the
```

```
        /// message.</param>
        /// <param name="messageBody">A string representing the body of the
        /// message to be sent to the queue.</param>
        /// <param name="messageAttributes">Attributes for the message to be
        /// sent to the queue.</param>
        /// <returns>A SendMessageResponse object that contains information
        /// about the message that was sent.</returns>
        public static async Task<SendMessageResponse> SendMessage(
            IAmazonSQS client,
            string queueUrl,
            string messageBody,
            Dictionary<string, MessageAttributeValue> messageAttributes)
        {
            var sendMessageRequest = new SendMessageRequest
            {
                DelaySeconds = 10,
                MessageAttributes = messageAttributes,
                MessageBody = messageBody,
                QueueUrl = queueUrl,
            };

            var response = await client.SendMessageAsync(sendMessageRequest);
            Console.WriteLine($"Sent a message with id : {response.MessageId}");

            return response;
        }
    }
```

- Para obter detalhes da API, consulte SendMessagea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Send a message to an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageBody: A message body.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::sendMessage(const Aws::String &queueUrl,
                              const Aws::String &messageBody,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SendMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMessageBody(messageBody);

    const Aws::SQS::Model::SendMessageOutcome outcome =
 sqsClient.SendMessage(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message to " << queueUrl <<
                  std::endl;
    }
    else {
        std::cerr << "Error sending message to " << queueUrl << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte SendMessagea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para enviar uma mensagem

Este exemplo envia uma mensagem com o corpo da mensagem, o período de atraso e os atributos da mensagem especificados para a fila especificada.

Comando:

```
aws sqs send-message --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --message-body "Information about the largest city in Any Region." --delay-seconds 10 --message-attributes file://send-message.json
```

Arquivo de entrada (send-message.json):

```
{
  "City": {
    "DataType": "String",
    "StringValue": "Any City"
  },
  "Greeting": {
    "DataType": "Binary",
    "BinaryValue": "Hello, World!"
  },
  "Population": {
    "DataType": "Number",
    "StringValue": "1250800"
  }
}
```

Saída:

```
{
  "MD5OfMessageBody": "51b0a325...39163aa0",
  "MD5OfMessageAttributes": "00484c68...59e48f06",
  "MessageId": "da68f62c-0c07-4bee-bf5f-7e856EXAMPLE"
}
```

- Para obter detalhes da API, consulte SendMessageem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Seguem dois exemplos da operação `SendMessage`:

- Enviar uma mensagem com um corpo e um atraso
- Enviar uma mensagem com um corpo e atributos de mensagem

Enviar uma mensagem com um corpo e um atraso.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName> <message>

                Where:
                   queueName - The name of the queue.
                   message - The message to send.
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }

    public static void sendMessage(SqsClient sqsClient, String queueName, String
 message) {
        try {
            CreateQueueRequest request = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
            sqsClient.createQueue(request);

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(message)
                    .delaySeconds(5)
                    .build();

            sqsClient.sendMessage(sendMsgRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Enviar uma mensagem com um corpo e atributos de mensagem.

```
    /**
     * <p>This method demonstrates how to add message attributes to a message.
     * Each attribute must specify a name, value, and data type. You use a Java
 Map to supply the attributes. The map's
     * key is the attribute name, and you specify the map's entry value using a
 builder that includes the attribute
     * value and data type.</p>
     *
     * <p>The data type must start with one of "String", "Number" or "Binary".
 You can optionally
     * define a custom extension by using a "." and your extension.</p>
     *
     * <p>The SQS Developer Guide provides more information on @see <a
     * href="https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-message-metadata.html#sqs-message-attributes">message
     * attributes</a>.</p>
     *
     * @param thumbailPath Filesystem path of the image.
     * @param queueUrl     URL of the SQS queue.
     */
    static void sendMessageWithAttributes(Path thumbailPath, String queueUrl) {
        Map<String, MessageAttributeValue> messageAttributeMap;
        try {
            messageAttributeMap = Map.of(
                    "Name", MessageAttributeValue.builder()
                            .stringValue("Jane Doe")
                            .dataType("String").build(),
                    "Age", MessageAttributeValue.builder()
                            .stringValue("42")
                            .dataType("Number.int").build(),
                    "Image", MessageAttributeValue.builder()

 .binaryValue(SdkBytes.fromByteArray(Files.readAllBytes(thumbailPath)))
                            .dataType("Binary.jpg").build()
            );
        } catch (IOException e) {
            LOGGER.error("An I/O exception occurred reading thumbnail image: {}",
 e.getMessage(), e);
            throw new RuntimeException(e);
```

```
        }

        SendMessageRequest request = SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello SQS")
                .messageAttributes(messageAttributeMap)
                .build();
        try {
            SendMessageResponse sendMessageResponse =
SQS_CLIENT.sendMessage(request);
            LOGGER.info("Message ID: {}", sendMessageResponse.messageId());
        } catch (SqsException e) {
            LOGGER.error("Exception occurred sending message: {}",
e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte SendMessagea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Enviar uma mensagem para uma fila do Amazon SQS.

```
import { SendMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

export const main = async (sqsQueueUrl = SQS_QUEUE_URL) => {
  const command = new SendMessageCommand({
    QueueUrl: sqsQueueUrl,
```

```
    DelaySeconds: 10,
    MessageAttributes: {
      Title: {
        DataType: "String",
        StringValue: "The Whistler",
      },
      Author: {
        DataType: "String",
        StringValue: "John Grisham",
      },
      WeeksOn: {
        DataType: "Number",
        StringValue: "6",
      },
    },
    MessageBody:
      "Information about current NY Times fiction bestseller for week of
 12/11/2016.",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte SendMessagea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Enviar uma mensagem para uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  // Remove DelaySeconds parameter and value for FIFO queues
  DelaySeconds: 10,
  MessageAttributes: {
    Title: {
      DataType: "String",
      StringValue: "The Whistler",
    },
    Author: {
      DataType: "String",
      StringValue: "John Grisham",
    },
    WeeksOn: {
      DataType: "Number",
      StringValue: "6",
    },
  },
  MessageBody:
    "Information about current NY Times fiction bestseller for week of
 12/11/2016.",
  // MessageDeduplicationId: "TheWhistler",  // Required for FIFO queues
  // MessageGroupId: "Group1",  // Required for FIFO queues
  QueueUrl: "SQS_QUEUE_URL",
};

sqs.sendMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.MessageId);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

- Para obter detalhes da API, consulte SendMessagea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
            messageBody = message
            delaySeconds = 10
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }
```

```
    val msg2 =
        SendMessageBatchRequestEntry {
            id = "id2"
            messageBody = "Hello from msg 2"
        }

    val sendMessageBatchRequest =
        SendMessageBatchRequest {
            queueUrl = queueUrlVal
            entries = listOf(msg1, msg2)
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- Para obter detalhes da API, consulte a SendMessagereferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo envia uma mensagem com os atributos e o corpo da mensagem especificados para a fila especificada com a entrega da mensagem atrasada por 10 segundos.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)
```

```
Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
MD5OfMessageAttributes              MD5OfMessageBody                    MessageId

----------------------              ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE    51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- Para obter detalhes da API, consulte SendMessageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo envia uma mensagem com os atributos e o corpo da mensagem especificados para a fila especificada com a entrega da mensagem atrasada por 10 segundos.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
MD5OfMessageAttributes              MD5OfMessageBody                    MessageId
```

```
----------------------              ----------------                   ---------

1d3e51347bc042efbdf6dda31EXAMPLE    51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- Para obter detalhes da API, consulte SendMessageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def send_message(queue, message_body, message_attributes=None):
    """
    Send a message to an Amazon SQS queue.

    :param queue: The queue that receives the message.
    :param message_body: The body text of the message.
    :param message_attributes: Custom attributes of the message. These are key-
value
                               pairs that can be whatever you want.
    :return: The response from SQS that contains the assigned message ID.
    """
    if not message_attributes:
        message_attributes = {}

    try:
        response = queue.send_message(
            MessageBody=message_body, MessageAttributes=message_attributes
        )
    except ClientError as error:
        logger.exception("Send message failed: %s", message_body)
        raise error
    else:
        return response
```

- Para obter detalhes da API, consulte a SendMessageReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end
```

```
# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  message_body = 'This is my message.'

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending a message to the queue named '#{queue_name}'..."

  if message_sent?(sqs_client, queue_url, message_body)
    puts 'Message sent.'
  else
    puts 'Message not sent.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte SendMessagea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn send(client: &Client, queue_url: &String, message: &SQSMessage) ->
 Result<(), Error> {
    println!("Sending message to queue with URL: {}", queue_url);

    let rsp = client
        .send_message()
        .queue_url(queue_url)
        .message_body(&message.body)
        // If the queue is FIFO, you need to set .message_deduplication_id
        // and message_group_id or configure the queue for
 ContentBasedDeduplication.
        .send()
        .await?;

    println!("Send message to the queue: {:#?}", rsp);

    Ok(())
}
```

- Para obter detalhes da API, consulte a SendMessagereferência da API AWS SDK for Rust.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sqs->sendmessage(              " oo_result is returned for
 testing purposes. "
           iv_queueurl = iv_queue_url
           iv_messagebody = iv_message ).
        MESSAGE 'Message sent to SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsinvalidmsgconts.
        MESSAGE 'Message contains non-valid characters.' TYPE 'E'.
      CATCH /aws1/cx_sqsunsupportedop.
```

```
        MESSAGE 'Operation not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendMessagereferência da API AWS SDK for SAP ABAP.

Use **SendMessageBatch**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SendMessageBatch`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Enviar e receber lotes de mensagens

CLI

AWS CLI

Como enviar várias mensagens como um lote

Este exemplo envia duas mensagens com os corpos da mensagem, os períodos de atraso e os atributos de mensagem especificados para a fila especificada.

Comando:

```
aws sqs send-message-batch --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://send-message-
batch.json
```

Arquivo de entrada (send-message-batch.json):

```
[
  {
    "Id": "FuelReport-0001-2015-09-16T140731Z",
        "MessageBody": "Fuel report for account 0001 on 2015-09-16 at 02:07:31
 PM.",
        "DelaySeconds": 10,
        "MessageAttributes": {
          "SellerName": {
            "DataType": "String",
```

```
                "StringValue": "Example Store"
      },
          "City": {
        "DataType": "String",
        "StringValue": "Any City"
      },
          "Region": {
            "DataType": "String",
                "StringValue": "WA"
      },
          "PostalCode": {
            "DataType": "String",
                "StringValue": "99065"
          },
          "PricePerGallon": {
            "DataType": "Number",
                "StringValue": "1.99"
      }
        }
  },
  {
    "Id": "FuelReport-0002-2015-09-16T140930Z",
        "MessageBody": "Fuel report for account 0002 on 2015-09-16 at 02:09:30
PM.",
        "DelaySeconds": 10,
        "MessageAttributes": {
          "SellerName": {
            "DataType": "String",
                "StringValue": "Example Fuels"
      },
          "City": {
        "DataType": "String",
        "StringValue": "North Town"
      },
          "Region": {
            "DataType": "String",
                "StringValue": "WA"
      },
          "PostalCode": {
            "DataType": "String",
                "StringValue": "99123"
          },
          "PricePerGallon": {
            "DataType": "Number",
```

```
                "StringValue": "1.87"
      }
        }
  }
]
```

Saída:

```
{
  "Successful": [
    {
      "MD5OfMessageBody": "203c4a38...7943237e",
      "MD5OfMessageAttributes": "10809b55...baf283ef",
      "Id": "FuelReport-0001-2015-09-16T140731Z",
      "MessageId": "d175070c-d6b8-4101-861d-adeb3EXAMPLE"
    },
    {
      "MD5OfMessageBody": "2cf0159a...c1980595",
      "MD5OfMessageAttributes": "55623928...ae354a25",
      "Id": "FuelReport-0002-2015-09-16T140930Z",
      "MessageId": "f9b7d55d-0570-413e-b9c5-a9264EXAMPLE"
    }
  ]
}
```

- Para obter detalhes da API, consulte SendMessageBatchem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
            SendMessageBatchRequest sendMessageBatchRequest =
 SendMessageBatchRequest.builder()
```

```
                    .queueUrl(queueUrl)

 .entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello
 from msg 1").build(),

 SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
 2").delaySeconds(10)
                                     .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Para obter detalhes da API, consulte SendMessageBatcha Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo envia duas mensagens com os atributos especificados e corpos de mensagem para a fila especificada. A entrega é adiada por 15 segundos para a primeira mensagem e 10 segundos para a segunda mensagem.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"

$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
```

```
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

Saída:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- Para obter detalhes da API, consulte SendMessageBatchem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo envia duas mensagens com os atributos especificados e corpos de mensagem para a fila especificada. A entrega é adiada por 15 segundos para a primeira mensagem e 10 segundos para a segunda mensagem.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"
```

```
$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

Saída:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- Para obter detalhes da API, consulte SendMessageBatchem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not
 sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and
 failed
             messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
            for ind, msg in enumerate(messages)
        ]
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
```

```
        return response
```

- Para obter detalhes da API, consulte a SendMessageBatchReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
```

```
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
```

```
  end
end
```

- Para obter detalhes da API, consulte SendMessageBatcha Referência AWS SDK para Ruby da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sqs->sendmessagebatch(         " oo_result is returned for
testing purposes. "
          iv_queueurl = iv_queue_url
          it_entries = it_messages ).
       MESSAGE 'Messages sent to SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsbtcentidsnotdist00.
       MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
     CATCH /aws1/cx_sqsbatchreqtoolong.
       MESSAGE 'The length of all the messages put together is more than the
limit.' TYPE 'E'.
     CATCH /aws1/cx_sqsemptybatchrequest.
       MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
     CATCH /aws1/cx_sqsinvbatchentryid.
       MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE
'E'.
     CATCH /aws1/cx_sqstoomanyentriesin00.
       MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
     CATCH /aws1/cx_sqsunsupportedop.
       MESSAGE 'Operation not supported.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a SendMessageBatchreferência da API AWS SDK for SAP ABAP.

Use **`SetQueueAttributes`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SetQueueAttributes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Publicar mensagens em filas

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Definir o atributo de política de uma fila para um tópico.

```
    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
```

```
                                                "}," +
                                        "\"Action\": \"sqs:SendMessage\"," +
                                        $"\"Resource\": \"{queueArn}\"," +
                                         "\"Condition\": {" +
                                              "\"ArnEquals\": {" +
                                                   $"\"aws:SourceArn\":
 \"{topicArn}\"" +
                                                "}" +
                                           "}" +
                                      "}]" +
                                  "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte SetQueueAttributesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Set the value for an attribute in an Amazon Simple Queue Service (Amazon SQS)
 queue.
```

```
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param attributeName: An attribute name enum.
  \param attribute: The attribute value as a string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setQueueAttributes(const Aws::String &queueURL,
                                     Aws::SQS::Model::QueueAttributeName
 attributeName,
                                     const Aws::String &attribute,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
            attributeName,
            attribute);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
 sqsClient.SetQueueAttributes(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set the attribute  " <<

 Aws::SQS::Model::QueueAttributeNameMapper::GetNameForQueueAttributeName(
                         attributeName)
                  << " with value " << attribute << " in queue " <<
                  queueURL << "." << std::endl;
    }
    else {
        std::cout << "Error setting attribute for  queue " <<
                  queueURL << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

Configurar uma dead-letter queue.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Connect an Amazon Simple Queue Service (Amazon SQS) queue to an associated
//! dead-letter queue.
/*!
  \param srcQueueUrl: An Amazon SQS queue URL.
  \param deadLetterQueueARN: The Amazon Resource Name (ARN) of an Amazon SQS
 dead-letter queue.
  \param maxReceiveCount: The max receive count of a message before it is sent to
 the dead-letter queue.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setDeadLetterQueue(const Aws::String &srcQueueUrl,
                                     const Aws::String &deadLetterQueueARN,
                                     int maxReceiveCount,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String redrivePolicy = MakeRedrivePolicy(deadLetterQueueARN,
 maxReceiveCount);

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(srcQueueUrl);
    request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
            redrivePolicy);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set dead letter queue for queue  " <<
                  srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
    }
    else {
        std::cerr << "Error setting dead letter queue for queue " <<
                  srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Make a redrive policy for a dead-letter queue.
/*!
  \param queueArn: An Amazon SQS ARN for the dead-letter queue.
  \param maxReceiveCount: The max receive count of a message before it is sent to
 the dead-letter queue.
  \return Aws::String: Policy as JSON string.
 */
Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queueArn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(maxReceiveCount);

    Aws::Utils::Json::JsonValue policy_map;
    policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
    policy_map.WithObject("maxReceiveCount", max_msg_entry);

    return policy_map.View().WriteReadable();
}
```

Configurar uma fila do Amazon SQS para usar sondagem longa.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Set the wait time for an Amazon Simple Queue Service (Amazon SQS) queue poll.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param pollTimeSeconds: The receive message wait time in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setQueueLongPollingAttribute(const Aws::String &queueURL,
                                               const Aws::String
 &pollTimeSeconds,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
```

```
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
            pollTimeSeconds);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
 sqsClient.SetQueueAttributes(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated long polling time for queue " <<
                  queueURL << " to " << pollTimeSeconds << std::endl;
    }
    else {
        std::cout << "Error updating long polling time for queue " <<
                  queueURL << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte SetQueueAttributesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como definir atributos de fila

Este exemplo define a fila especificada com um atraso de entrega de 10 segundos, um tamanho máximo de mensagem de 128 KB (128 KB * 1.024 bytes), um período de retenção de mensagens de 3 dias (3 dias * 24 horas * 60 minutos * 60 segundos), um tempo de espera de recebimento de mensagens de 20 segundos e um tempo limite de visibilidade padrão de 60 segundos. Este exemplo também associa a fila de mensagens não entregues especificada a uma contagem máxima de recebimento de 1.000 mensagens.

Comando:

```
aws sqs set-queue-attributes --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyNewQueue --attributes file://set-queue-
attributes.json
```

Arquivo de entrada (set-queue-attributes.json):

```
{
  "DelaySeconds": "10",
  "MaximumMessageSize": "131072",
  "MessageRetentionPeriod": "259200",
  "ReceiveMessageWaitTimeSeconds": "20",
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":\"1000\"}",
  "VisibilityTimeout": "60"
}
```

Saída:

```
None.
```

- Para obter detalhes da API, consulte SetQueueAttributesem Referência de AWS CLI Comandos.

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
```

```
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string
```

- Para obter detalhes da API, consulte SetQueueAttributesa Referência AWS SDK para Go da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure um Amazon SQS para usar a criptografia do lado do servidor (SSE) usando uma chave do KMS personalizada.

```
   public static void addEncryption(String queueName, String kmsMasterKeyAlias)
{
       SqsClient sqsClient = SqsClient.create();

       GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
               .queueName(queueName)
               .build();

       GetQueueUrlResponse getQueueUrlResponse;
       try {
           getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
       } catch (QueueDoesNotExistException e) {
           LOGGER.error(e.getMessage(), e);
           throw new RuntimeException(e);
       }
       String queueUrl = getQueueUrlResponse.queueUrl();


       Map<QueueAttributeName, String> attributes = Map.of(
               QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
               QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" //
Set the data key reuse period to 140 seconds.
       );                                                                  //
This is how long SQS can reuse the data key before requesting a new one from
KMS.

       SetQueueAttributesRequest attRequest =
SetQueueAttributesRequest.builder()
               .queueUrl(queueUrl)
               .attributes(attributes)
               .build();
       try {
           sqsClient.setQueueAttributes(attRequest);
           LOGGER.info("The attributes have been applied to {}", queueName);
       } catch (InvalidAttributeNameException | InvalidAttributeValueException
e) {
           LOGGER.error(e.getMessage(), e);
           throw new RuntimeException(e);
       } finally {
           sqsClient.close();
```

```
        }
    }
```

- Para obter detalhes da API, consulte SetQueueAttributesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    QueueUrl: queueUrl,
    Attributes: {
      DelaySeconds: "1",
    },
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Configurar uma fila do Amazon SQS para usar sondagem longa.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
```

```
const SQS_QUEUE_URL = "queue_url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    Attributes: {
      ReceiveMessageWaitTimeSeconds: "20",
    },
    QueueUrl: queueUrl,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Configurar uma dead-letter queue.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";
const DEAD_LETTER_QUEUE_ARN = "dead_letter_queue_arn";

export const main = async (
  queueUrl = SQS_QUEUE_URL,
  deadLetterQueueArn = DEAD_LETTER_QUEUE_ARN,
) => {
  const command = new SetQueueAttributesCommand({
    Attributes: {
      RedrivePolicy: JSON.stringify({
        // Amazon SQS supports dead-letter queues (DLQ), which other
        // queues (source queues) can target for messages that can't
        // be processed (consumed) successfully.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-dead-letter-queues.html
        deadLetterTargetArn: deadLetterQueueArn,
        maxReceiveCount: "10",
      }),
    },
    QueueUrl: queueUrl,
  });
```

```
  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte SetQueueAttributesa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra como definir uma política para assinar um tópico do SNS com uma fila. Quando uma mensagem é publicada no tópico, ela é enviada à fila assinada.

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName
 "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
      {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
          }
```

```
      }
    }
  ]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

Exemplo 2: esse exemplo define os atributos especificados da fila especificada.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte SetQueueAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como definir uma política para assinar um tópico do SNS com uma fila. Quando uma mensagem é publicada no tópico, ela é enviada à fila assinada.

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName
 "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
      {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
```

```
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
          }
      }
    }
  ]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

Exemplo 2: esse exemplo define os atributos especificados da fila especificada.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes da API, consulte SetQueueAttributesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Definir o atributo de política de uma fila para um tópico.

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.
```

```
        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
queue_url: str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
```

```
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
 {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a SetQueueAttributesReferência da API AWS SDK for Python (Boto3).

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        do {
            _ = try await sqsClient.setQueueAttributes(
```

```
                input: SetQueueAttributesInput(
                    attributes: [
                        "MaximumMessageSize": "\(maxSize)"
                    ],
                    queueUrl: url
                )
            )
        } catch _ as AWSSQS.InvalidAttributeValue {
            print("Invalid maximum message size: \(maxSize) kB.")
        }
```

- Para obter detalhes da API, consulte SetQueueAttributesa referência da API AWS SDK for Swift.

Cenários para o Amazon SQS usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon SQS com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon SQS ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS
- Criar uma aplicação de mensageiro com o Step Functions
- Criar uma aplicação de exploração do Amazon Textract
- Crie e publique em um tópico FIFO do Amazon SNS usando um AWS SDK
- Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK
- Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um AWS SDK
- Receba e processe notificações de eventos do Amazon S3 usando um AWS SDK
- Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um AWS SDK
- Envie e receba lotes de mensagens com o Amazon SQS usando um AWS SDK

- Use o comando AWS Estrutura de processamento de mensagens para.NET para publicar e receber mensagens do Amazon SQS
- Usar a biblioteca de mensagens Java do Amazon SQS para trabalhar com a interface do Serviço de Mensagens Java (JMS) para o Amazon SQS
- Trabalhe com tags de fila e o Amazon SQS usando um AWS SDK

Criar um aplicativo web que envie e recupere mensagens usando o Amazon SQS

Os exemplos de código a seguir mostram como criar uma aplicação de mensagens usando o Amazon SQS.

Java

SDK para Java 2.x

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Kotlin

SDK para Kotlin

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend

- Amazon SQS

Criar uma aplicação de mensageiro com o Step Functions

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.
- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

Criar uma aplicação de exploração do Amazon Textract

Os exemplos de código a seguir mostram como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.

- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Crie e publique em um tópico FIFO do Amazon SNS usando um AWS SDK

Os exemplos de código a seguir mostram como criar e publicar em um tópico FIFO do Amazon SNS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O teste verifica o recebimento da mensagem em cada fila. O exemplo completo também mostra a adição de políticas de acesso e exclui os recursos no final.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {
```

```
        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "   fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "   retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "   analyticsQueueARN - The name of a SQS standard queue that will be
created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);
```

```
        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();
```

```
            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTopic
  - Publicar
  - Assinar

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO do Amazon SNS, inscreva filas padrão e FIFO do Amazon SQS no tópico e publique uma mensagem no tópico.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()
```

```
    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
```

```
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
```

```
        numbers, underscores, and hyphens, and must be between 1 and 256
characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version":"2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
```

```
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
```

```
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTopic
  - Publicar
  - Assinar

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO, inscreva uma fila FIFO do Amazon SQS no tópico e publique uma mensagem em um tópico do Amazon SNS.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
```

```
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String'

 iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
```

```
            it_messageattributes = lt_msg_attributes ).
        ov_message_id = lo_result->get_messageid( ).                    "
ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para SAP ABAP.
  - CreateTopic
  - Publicar
  - Assinar

Detecte pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Gerencie grandes mensagens do Amazon SQS usando o Amazon S3 com um AWS SDK

O exemplo de código a seguir mostra como usar a biblioteca do cliente em versão ampliada do Amazon SQS para trabalhar com mensagens grandes do SQS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
```

```
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
 ExtendedClientConfiguration()
```

```
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
```

```
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(),
e);
            throw e;
        }
    }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK for Java 2.x.
- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

Receba e processe notificações de eventos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como trabalhar com notificações de eventos do S3 de uma forma orientada a objetos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo mostra como processar o evento de notificação do S3 usando o Amazon SQS.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload
 and logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
 API</a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
     * <groupId>software.amazon.awssdk</groupId>
     * <artifactId>s3-event-notifications</artifactId>
     * <version><LATEST></version>
     * </dependency>
     * <p>
     * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
     * <p>
     * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
     * in AWS SNS or AWS Lambda as well.
     * <p>
```

```
    * Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String
queueArn) {
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the
queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                       ).thenAccept(attributeResponse ->
                               logger.info("Approximate number of messages in
the queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                       .join();

               // Receive the messages.
               ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                       .queueUrl(queueUrl)
               ).get();
               logger.info("Count of received messages: {}",
response.messages().size());
               didReceiveMessages = !response.messages().isEmpty();
```

```
                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification
object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}",
eventName, key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {

sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Publique mensagens do Amazon SNS nas filas do Amazon SQS usando um AWS SDK

Os exemplos de código a seguir mostram como:

- Crie um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
/// <summary>
/// Console application to run a feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
```

```
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
"sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
```

```
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
```

```
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
 subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
 configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");
```

```
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
 Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
```

```
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
```

```
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
 queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
```

```
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
```

```
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
```

```
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                        "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
```

```
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Crie uma classe que envolva operações do Amazon SQS.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;

    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
 useFifoQueue)
    {
        int maxMessage = 256 * 1024;
```

```
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
```

```
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
                                               $"\"aws:SourceArn\":
\"{topicArn}\"" +
                                            "}" +
                                        "}" +
                                "}]" +
                            "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
```

```
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
 List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
```

```
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
 _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Crie uma classe que envolva operações do Amazon SNS.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
```

```
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
```

```
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
 subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
 filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
 { { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
```

```
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
_amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
```

```
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
 queues."
              << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
```

```
            << "FIFO topics deliver messages in order and support deduplication
 and message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or
 automatically generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic,
 any message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted
 but not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
```

```
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

```
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
```

```
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
```

```
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }
```

```
        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
filters."
```

```
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
 filter "
                                << "will be added to this subscription." <<
 std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);
```

```
            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
```

```
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
```

```
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
```

```
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
```

```
                                        bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
```

```
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                           const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
```

```
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
 "strings"
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
```

```
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string,
 bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
```

```
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
 isFifoTopic bool) (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string,
 topicArn string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
```

```
}
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
 "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
 if ordinal == "first" {
  log.Println("Subscriptions to a FIFO topic can have filters.\n" +
   "If you add a filter to this subscription, then only the filtered messages\n"
+
   "will be received in the queue.\n" +
   "For information about message filtering, see\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
   "For this example, you can filter messages by a \"tone\" attribute.")
 }

 wantFiltering := runner.questioner.AskBool(
  fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
   "from the %v topic? (y/n) ", queueName, topicName), "y")
 if wantFiltering {
  log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

  var toneSelections []string
  askAboutTones := true
  for askAboutTones {
   toneIndex := runner.questioner.AskChoice(
    "Enter the number of the tone you want to filter by:\n", ToneChoices)
   toneSelections = append(toneSelections, ToneChoices[toneIndex])
   askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
  }
  log.Printf("Your subscription will be filtered to only pass the following
tones: %v\n", toneSelections)
  filterPolicy = map[string][]string{TONE_KEY: toneSelections}
 }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)
if err != nil {
 panic(err)
}
```

```
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn
 string, isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId,
 TONE_KEY, toneSelection)
  if err != nil {
   panic(err)
```

```
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
```

```
    if err != nil {
     panic(err)
    }
   }
  }
 }

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)
```

```
 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
 topicName, topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina um struct que envolva as ações do Amazon SNS usadas neste exemplo.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
```

```
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
```

```
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
```

```
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Defina um struct que envolva as ações do Amazon SQS usadas neste exemplo.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
```

```
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
```

```
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
```

```
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
```

```
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Limpe recursos.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
```

```
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management
 Console\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar

- Cancelar assinatura

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
```

```
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
```

```
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
```

```
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and
 the subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,
\n"
                +
                "        within the five-minute deduplication interval, is
 accepted but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
```

```
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");
```

```
        // Define the policy to use. Make sure that you change the REGION if you
 are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
```

```
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
```

```
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
```

```
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
```

```
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
```

```
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   // Subscribe to the SQS queue.
   public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
       try {
           SubscribeRequest request;
           if (filterList.isEmpty()) {
               // No filter subscription is added.
               request = SubscribeRequest.builder()
                   .protocol("sqs")
                   .endpoint(queueArn)
                   .returnSubscriptionArn(true)
                   .topicArn(topicArn)
                   .build();

               SubscribeResponse result = snsClient.subscribe(request);
               System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                   "with the subscription ARN " + result.subscriptionArn());
               return result.subscriptionArn();
           } else {
               request = SubscribeRequest.builder()
                   .protocol("sqs")
                   .endpoint(queueArn)
                   .returnSubscriptionArn(true)
                   .topicArn(topicArn)
                   .build();

               SubscribeResponse result = snsClient.subscribe(request);
               System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                   "with the subscription ARN " + result.subscriptionArn());

               String attributeName = "FilterPolicy";
               Gson gson = new Gson();
               String jsonString = "{\"tone\": []}";
               JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
               JsonArray toneArray = jsonObject.getAsJsonArray("tone");
               for (String value : filterList) {
                   toneArray.add(new JsonPrimitive(value));
               }

               String updatedJsonString = gson.toJson(jsonObject);
```

```
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
```

```
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
```

```
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }
```

```
            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o ponto de entrada para esse cenário.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

O código anterior fornece as dependências necessárias e inicia o cenário. A próxima seção contém a maior parte do exemplo.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];
```

```
export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
    this.sqsClient = sqsClient;
    this.prompter = prompter;
    this.logger = logger;
  }

  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
```

```
      await this.logger.log(MESSAGES.deduplicationNotice);
      await this.logger.log(MESSAGES.deduplicationDescription);
      this.autoDedup = await this.prompter.confirm({
        message: MESSAGES.deduplicationPrompt,
      });
    }
  }

  async createTopic() {
    await this.logger.log(MESSAGES.creatingTopics);
    this.topicName = await this.prompter.input({
      message: MESSAGES.topicNamePrompt,
    });
    if (this.isFifo) {
      this.topicName += ".fifo";
      this.logger.logSeparator(MESSAGES.headerFifoNaming);
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
```

```
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
```

```
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
```

```
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
```

```
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }
```

```
    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );
```

```
      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }
```

```
    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
```

```
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
```

```
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an
SQS queue to the topic.
                You can select from several options for configuring the topic and
the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
and message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
delivered.
```

```
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
```

```
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
```

```
        println("Would you like to filter messages for $sqsQueueName's
subscription to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the
following \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following
\"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
```

```
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)
```

```
    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
```

```
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
```

```
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
```

```
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
```

```
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
```

```
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
```

```
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
```

```
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic

- GetQueueAttributes
- Publicar
- ReceiveMessage
- SetQueueAttributes
- Assinar
- Cancelar assinatura

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
```

```
        self.queue_count = 2
        self.queue_urls = []
        self.subscription_arns = []
        self.tones = ["cheerful", "funny", "serious", "sincere"]

    def run_scenario(self) -> None:
        """Run the Topics and Queues feature scenario."""
        print(self.DASHES)
        print("Welcome to messaging with topics and queues.")
        print(self.DASHES)
        print(f"""
    In this scenario, you will create an SNS topic and subscribe
 {self.queue_count} SQS queues to the topic.
    You can select from several options for configuring the topic and the
 subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
        """)

        try:
            # Setup Phase
            print(self.DASHES)
            self._setup_topic()
            print(self.DASHES)

            self._setup_queues()
            print(self.DASHES)

            # Demonstration Phase
            self._publish_messages()
            print(self.DASHES)

            # Examination Phase
            self._poll_queues_for_messages()
            print(self.DASHES)

            # Cleanup Phase
            self._cleanup_resources()
            print(self.DASHES)

        except Exception as e:
            logger.error(f"Scenario failed: {e}")
            print(f"There was a problem with the scenario: {e}")
            print("\nInitiating cleanup...")
            try:
```

```
                self._cleanup_resources()
            except Exception as cleanup_error:
                logger.error(f"Error during cleanup: {cleanup_error}")

        print("Messaging with topics and queues scenario is complete.")
        print(self.DASHES)

    def _setup_topic(self) -> None:
        """Set up the SNS topic to be used with the queues."""
        print("SNS topics can be configured as FIFO (First-In-First-Out).")
        print("FIFO topics deliver messages in order and support deduplication
 and message filtering.")
        print()

        self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/
n): ", q.is_yesno)

        if self.use_fifo_topic:
            print(self.DASHES)
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
 q.non_empty)
            print("Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name.")
            print()

            print(self.DASHES)
            print("""
    Because you have chosen a FIFO topic, deduplication is supported.
    Deduplication IDs are either set in the message or automatically generated
    from content using a hash function.

    If a message is successfully published to an SNS FIFO topic, any message
    published and determined to have the same deduplication ID,
    within the five-minute deduplication interval, is accepted but not delivered.

    For more information about deduplication,
    see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
            """)

            self.use_content_based_deduplication = q.ask(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n): ",
                q.is_yesno
            )
```

```
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)

        print(self.DASHES)

        # Create the topic
        self.topic_arn = self.sns_wrapper.create_topic(
            self.topic_name,
            self.use_fifo_topic,
            self.use_content_based_deduplication
        )

        print(f"Your new topic with the name {self.topic_name}")
        print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
        print(f"  has been created.")
        print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
        print(f"Now you will create {self.queue_count} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ",
q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)

            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()

            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")
```

```
                print(self.DASHES)

            # Get queue ARN
            queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

            if i == 0:
                print("An AWS Identity and Access Management (IAM) policy must be
 attached to an SQS queue,")
                print("enabling it to receive messages from an SNS topic.")

            # Set queue policy to allow SNS to send messages
            self.sqs_wrapper.set_queue_policy_for_topic(queue_arn,
 self.topic_arn, queue_url)

            # Set up message filtering if using FIFO
            subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
 queue_name)
            self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
 queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
 filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
                print()
                print("For this example, you can filter messages by a TONE
 attribute.")

            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription
 to the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()
```

```
        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
{self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

    def _create_filter_policy(self) -> str:
        """Create a message filter policy based on user selections."""
        print(self.DASHES)
        print("You can filter messages by one or more of the following TONE
attributes.")

        filter_selections = []
        selection_number = 0

        while True:
            print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
            for i, tone in enumerate(self.tones, 1):
                print(f"  {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

            if selection == 0:
                break
            elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
                filter_selections.append(self.tones[selection - 1])
                print(f"Added '{self.tones[selection - 1]}' to filter list.")

        if filter_selections:
            filters = {"tone": filter_selections}
            return json.dumps(filters)
        return None

    def _publish_messages(self) -> None:
        """Publish messages to the topic with various options."""
        print("Now we can publish messages.")
```

```
        keep_sending = True
        while keep_sending:
            print()
            message = q.ask("Enter a message to publish: ", q.non_empty)

            message_group_id = None
            deduplication_id = None
            tone_attribute = None

            if self.use_fifo_topic:
                print("Because you are using a FIFO topic, you must set a message
group ID.")
                print("All messages within the same group will be received in the
order they were published.")
                print()
                message_group_id = q.ask("Enter a message group ID for this
message: ", q.non_empty)

                if not self.use_content_based_deduplication:
                    print("Because you are not using content-based
deduplication,")
                    print("you must enter a deduplication ID.")
                    deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

                # Ask about tone attribute
                add_attribute = q.ask("Add an attribute to this message? (y/n):
", q.is_yesno)
                if add_attribute:
                    print("Enter a number for an attribute:")
                    for i, tone in enumerate(self.tones, 1):
                        print(f"  {i}. {tone}")

                    selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
len(self.tones)))
                    if 1 <= selection <= len(self.tones):
                        tone_attribute = self.tones[selection - 1]

            # Publish the message
            message_id = self.sns_wrapper.publish_message(
                self.topic_arn,
                message,
                tone_attribute,
```

```
                deduplication_id,
                message_group_id
            )

            print(f"Message published with ID: {message_id}")

            keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

    def _poll_queues_for_messages(self) -> None:
        """Poll all queues for messages and display results."""
        for i, queue_url in enumerate(self.queue_urls):
            print(f"Polling queue #{i+1} at {queue_url} for messages...")

            q.ask("Press Enter to continue...")

            messages = self._poll_queue_for_messages(queue_url)

            if messages:
                print(f"{len(messages)} message(s) were received by queue #{i
+1}")
                for j, message in enumerate(messages, 1):
                    print(f"  Message {j}:")
                    # Parse the SNS message body to get the actual message
                    try:
                        sns_message = json.loads(message['Body'])
                        actual_message = sns_message.get('Message',
 message['Body'])
                        print(f"    {actual_message}")
                    except (json.JSONDecodeError, KeyError):
                        print(f"    {message['Body']}")

                # Delete the messages
                self.sqs_wrapper.delete_messages(queue_url, messages)
                print(f"Messages deleted from queue #{i+1}")
            else:
                print(f"No messages received by queue #{i+1}")

            print(self.DASHES)

    def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
        """Poll a single queue for messages."""
        all_messages = []
        max_polls = 3  # Limit polling to avoid infinite loops
```

```
        for poll_count in range(max_polls):
            messages = self.sqs_wrapper.receive_messages(queue_url, 10)

            if messages:
                all_messages.extend(messages)
                print(f"  Received {len(messages)} messages in poll {poll_count +
1}")
                # Small delay between polls
                time.sleep(1)
            else:
                print(f"  No messages in poll {poll_count + 1}")
                break

        return all_messages

    def _cleanup_resources(self) -> None:
        """Clean up all resources created during the scenario."""
        print("Cleaning up resources...")

        # Delete queues
        for i, queue_url in enumerate(self.queue_urls):
            if queue_url:
                delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}?
(y/n): ", q.is_yesno)
                if delete_queue:
                    try:
                        self.sqs_wrapper.delete_queue(queue_url)
                        print(f"Deleted queue #{i+1}")
                    except Exception as e:
                        print(f"Error deleting queue #{i+1}: {e}")

        # Unsubscribe from topic
        for i, subscription_arn in enumerate(self.subscription_arns):
            if subscription_arn:
                try:
                    self.sns_wrapper.unsubscribe(subscription_arn)
                    print(f"Unsubscribed subscription #{i+1}")
                except Exception as e:
                    print(f"Error unsubscribing #{i+1}: {e}")

        # Delete topic
        if self.topic_arn:
            delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
q.is_yesno)
```

```
            if delete_topic:
                try:
                    self.sns_wrapper.delete_topic(self.topic_arn)
                    print(f"Deleted topic {self.topic_name}")
                except Exception as e:
                    print(f"Error deleting topic: {e}")

        print("Resource cleanup complete.")
```

Crie classes que envolvam as operações do Amazon SNS e do Amazon SQS para uso no cenário.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
```

```
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} -
{e}")
            raise


    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.
```

```
        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
```

```
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} -
{e}")
            raise


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.
```

```
        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
                raise


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise
```

```
    def list_topics(self) -> list:
        """
        List all SNS topics in the account using pagination.

        :return: List of topic ARNs.
        :raises ClientError: If listing topics fails.
        """
        try:
            topics = []
            paginator = self.sns_client.get_paginator('list_topics')

            for page in paginator.paginate():
                topics.extend([topic['TopicArn'] for topic in page.get('Topics',
 [])])

            logger.info(f"Found {len(topics)} topics")
            return topics

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AuthorizationError':
                logger.error("Authorization error listing topics - check IAM
 permissions")
            else:
                logger.error(f"Error listing topics: {error_code} - {e}")
            raise


class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
```

```
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} -
{e}")
            raise


    def get_queue_arn(self, queue_url: str) -> str:
        """
```

```
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
queue_url: str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
```

```
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
```

```
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
```

```
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise


    def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
        """
        List all SQS queues in the account using pagination.
```

```
        :param queue_name_prefix: Optional prefix to filter queue names.
        :return: List of queue URLs.
        :raises ClientError: If listing queues fails.
        """
        try:
            queue_urls = []
            paginator = self.sqs_client.get_paginator('list_queues')

            page_params = {}
            if queue_name_prefix:
                page_params['QueueNamePrefix'] = queue_name_prefix

            for page in paginator.paginate(**page_params):
                queue_urls.extend(page.get('QueueUrls', []))

            logger.info(f"Found {len(queue_urls)} queues")
            return queue_urls

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AccessDenied':
                logger.error("Access denied listing queues - check IAM
permissions")
            else:
                logger.error(f"Error listing queues: {error_code} - {e}")
            raise

    def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
        """
        Send a message to an SQS queue.

        :param queue_url: The URL of the queue.
        :param message_body: The message content.
        :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
        :return: The message ID.
        :raises ClientError: If sending the message fails.
        """
        try:
            send_params = {
                'QueueUrl': queue_url,
                'MessageBody': message_body,
                **kwargs
```

```
            }

            response = self.sqs_client.send_message(**send_params)

            message_id = response['MessageId']
            logger.info(f"Sent message to {queue_url} with ID: {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error sending message: {error_code} - {e}")
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
```

```
            str = readLine()

            if str != nil && str?.count == 0 {
                str = nil
            }
        }

        return str!
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt: prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Display a menu of options then request a selection.
    ///
    /// - Parameters:
    ///   - prompt: A prompt string to display before the menu.
    ///   - options: An array of strings giving the menu options.
    ///
    /// - Returns: The index number of the selected option or 0 if no item was
    ///   selected.
    func menuRequest(prompt: String, options: [String]) -> Int {
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }
```

```
        repeat {
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]

    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
```

```
            print("  [\(index)] \(value)")
        }

        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true

        while let ans = askForFilter(message: "Choose a value to apply to the
'tone' attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false

            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
```

```
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]

            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that
name was recently deleted.")
                continue
            }
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws ->
String? {
```

```
        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }

        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
```

```
                                }
                            ]
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }

    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async
throws
                                ->
[SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.
```

```
        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl:
queueUrl)
        )

        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown
ID>") with error: \(message.code ?? "<unknown>") (\(message.message ?? "...")")
            }
        }
```

```
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues
 example.
              In this workflow, you'll create an SNS topic, then create two SQS
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well
 as
              the queue subscriptions. You can then post messages to the topic
 and
              receive the results on the queues.
              \(rowOfStars)\n
              """
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region:
 region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region:
 region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //    to use content-based deduplication instead of requiring a
        //    deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/
N)? ")
        var isContentBasedDeduplication = false

        if isFIFO {
            print("""
                  \(rowOfStars)
                  Because you've chosen to create a FIFO topic, deduplication is
```

```
                  supported.

                  Deduplication IDs are either set in the message or are
 automatically
                  generated from the content using a hash function.

                  If a message is successfully published to an SNS FIFO topic,
 any
                  message published and found to have the same deduplication ID
                  (within a five-minute deduplication interval), is accepted but
                  not delivered.

                  For more information about deduplication, see:
                  https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html.
                  """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
 deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
 create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
        // "ContentBasedDeduplication" attribute.
```

```
        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
            )
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print("""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print("""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
        }
```

```
        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")

        // 5. Attach an AWS IAM policy to the queue using
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
 messages will
                be received in the queue. For information about message
 filtering, see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's
 subscription to the
                topic \(topicName) (Y/N)?
```

```
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil
```

```
        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.
```

```
            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.
\n
                        """
                    )
                }
                publishInput.messageGroupId = stringRequest(prompt: "Enter a
 message group ID for this message: ")

                if !isContentBasedDeduplication {
                    if first {
                        print("""
                              Because you're not using content-based
 deduplication, you
                              must enter a deduplication ID. If other messages
 with the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
 "Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue]
 = [:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
 this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType:
 "String", stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }
```

```
            publishInput.messageAttributes = messageAttributes

            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }

            print("Message published with ID \(messageID).")
            first = false

            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.
```

```
        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Envie e receba lotes de mensagens com o Amazon SQS usando um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar uma fila do Amazon SQS.
- Enviar lotes de mensagens para a fila.
- Receber lotes de mensagens de uma fila.
- Excluir lotes de mensagens de uma fila.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Conforme mostrado nos exemplos a seguir, você pode gerenciar operações de mensagens em lote com o Amazon SQS usando duas abordagens diferentes com o AWS SDK for Java 2.x:

SendRecvBatch.javausa operações em lote explícitas. Você cria manualmente lotes de mensagens e chama `sendMessageBatch()` e `deleteMessageBatch()` diretamente. Você também gerencia respostas em lote, incluindo mensagens com falha. Essa abordagem oferece controle total sobre o dimensionamento do lote e o gerenciamento de erros. No entanto, ela requer mais código para gerenciar a lógica de agrupamento em lote.

SimpleProducerConsumer.javausa a `SqsAsyncBatchManager` biblioteca de alto nível para agrupamento automático de solicitações. Você faz chamadas individuais para `sendMessage()` e `deleteMessage()` com as mesmas assinaturas de método do cliente padrão. O SDK armazena automaticamente essas chamadas em buffer e as envia como operações em lote. Essa abordagem requer alterações mínimas no código e, ao mesmo tempo, fornece benefícios de desempenho em lote.

Use lotes explícitos quando precisar de um controle refinado sobre a composição do lote e o gerenciamento de erros. Use o agrupamento automático em lote quando quiser otimizar o desempenho com o mínimo de alterações no código.

SendRecvBatch.java - Usa operações em lote explícitas com mensagens.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.BatchResultErrorEntry;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.MessageAttributeValue;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.SqsException;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

/**
 * This code demonstrates basic message operations in Amazon Simple Queue Service
 (Amazon SQS).
 */

public class SendRecvBatch {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SendRecvBatch.class);
    private static final SqsClient sqsClient = SqsClient.create();


    public static void main(String[] args) {
        usageDemo();
    }
    /**
     * Send a batch of messages in a single request to an SQS queue.
     * This request may return overall success even when some messages were not
 sent.
     * The caller must inspect the Successful and Failed lists in the response
 and
     * resend any failed messages.
     *
     * @param queueUrl  The URL of the queue to receive the messages.
```

```
    * @param messages  The messages to send to the queue. Each message contains
a body and attributes.
    * @return The response from SQS that contains the list of successful and
failed messages.
    */
   public static SendMessageBatchResponse sendMessages(
           String queueUrl, List<MessageEntry> messages) {

       try {
           List<SendMessageBatchRequestEntry> entries = new ArrayList<>();

           for (int i = 0; i < messages.size(); i++) {
               MessageEntry message = messages.get(i);
               entries.add(SendMessageBatchRequestEntry.builder()
                       .id(String.valueOf(i))
                       .messageBody(message.getBody())
                       .messageAttributes(message.getAttributes())
                       .build());
           }

           SendMessageBatchRequest sendBatchRequest =
SendMessageBatchRequest.builder()
                   .queueUrl(queueUrl)
                   .entries(entries)
                   .build();

           SendMessageBatchResponse response =
sqsClient.sendMessageBatch(sendBatchRequest);

           if (!response.successful().isEmpty()) {
               for (SendMessageBatchResultEntry resultEntry :
response.successful()) {
                   LOGGER.info("Message sent: {}: {}", resultEntry.messageId(),

messages.get(Integer.parseInt(resultEntry.id())).getBody());
               }
           }

           if (!response.failed().isEmpty()) {
               for (BatchResultErrorEntry errorEntry : response.failed()) {
                   LOGGER.warn("Failed to send: {}: {}", errorEntry.id(),

messages.get(Integer.parseInt(errorEntry.id())).getBody());
               }
```

```
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Send messages failed to queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Receive a batch of messages in a single request from an SQS queue.
     *
     * @param queueUrl   The URL of the queue from which to receive messages.
     * @param maxNumber  The maximum number of messages to receive (capped at 10
by SQS).
     *                   The actual number of messages received might be less.
     * @param waitTime   The maximum time to wait (in seconds) before returning.
When
     *                   this number is greater than zero, long polling is used.
This
     *                   can result in reduced costs and fewer false empty
responses.
     * @return The list of Message objects received. These each contain the body
     *         of the message and metadata and custom attributes.
     */
    public static List<Message> receiveMessages(String queueUrl, int maxNumber,
int waitTime) {
        try {
            ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(maxNumber)
                    .waitTimeSeconds(waitTime)
                    .messageAttributeNames("All")
                    .build();

            List<Message> messages =
sqsClient.receiveMessage(receiveRequest).messages();

            for (Message message : messages) {
                LOGGER.info("Received message: {}: {}", message.messageId(),
message.body());
            }
```

```
            return messages;

        } catch (SqsException e) {
            LOGGER.error("Couldn't receive messages from queue: {}", queueUrl,
e);
            throw e;
        }
    }

    /**
     * Delete a batch of messages from a queue in a single request.
     *
     * @param queueUrl  The URL of the queue from which to delete the messages.
     * @param messages  The list of messages to delete.
     * @return The response from SQS that contains the list of successful and
failed
     *         message deletions.
     */
    public static DeleteMessageBatchResponse deleteMessages(String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();

            for (int i = 0; i < messages.size(); i++) {
                entries.add(DeleteMessageBatchRequestEntry.builder()
                        .id(String.valueOf(i))
                        .receiptHandle(messages.get(i).receiptHandle())
                        .build());
            }

            DeleteMessageBatchRequest deleteRequest =
DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            DeleteMessageBatchResponse response =
sqsClient.deleteMessageBatch(deleteRequest);

            if (!response.successful().isEmpty()) {
                for (DeleteMessageBatchResultEntry resultEntry :
response.successful()) {
```

```
                    LOGGER.info("Deleted {}",
messages.get(Integer.parseInt(resultEntry.id())).receiptHandle());
                }
            }

            if (!response.failed().isEmpty()) {
                for (BatchResultErrorEntry errorEntry : response.failed()) {
                    LOGGER.warn("Could not delete {}",
messages.get(Integer.parseInt(errorEntry.id())).receiptHandle());
                }
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Couldn't delete messages from queue {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Helper class to represent a message with body and attributes.
     */
    public static class MessageEntry {
        private final String body;
        private final Map<String, MessageAttributeValue> attributes;

        public MessageEntry(String body, Map<String, MessageAttributeValue>
attributes) {
            this.body = body;
            this.attributes = attributes != null ? attributes : new HashMap<>();
        }

        public String getBody() {
            return body;
        }

        public Map<String, MessageAttributeValue> getAttributes() {
            return attributes;
        }
    }

    /**
     * Shows how to:
```

```
     * * Read the lines from a file and send the lines in
     *   batches of 10 as messages to a queue.
     * * Receive the messages in batches until the queue is empty.
     * * Reassemble the lines of the file and verify they match the original
 file.
     */
    public static void usageDemo() {
        LOGGER.info("-".repeat(88));
        LOGGER.info("Welcome to the Amazon Simple Queue Service (Amazon SQS)
 demo!");
        LOGGER.info("-".repeat(88));

        String queueUrl = null;
        try {
            // Create a queue for the demo.
            String queueName = "sqs-usage-demo-message-wrapper-" +
 System.currentTimeMillis();
            CreateQueueRequest createRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
            queueUrl = sqsClient.createQueue(createRequest).queueUrl();
            LOGGER.info("Created queue: {}", queueUrl);

            try (InputStream inputStream =
 SendRecvBatch.class.getResourceAsStream("/log4j2.xml");
                 BufferedReader reader = new BufferedReader(new
 InputStreamReader(inputStream))) {

                List<String> lines = reader.lines().toList();

                // Send file lines in batches.
                int batchSize = 10;
                LOGGER.info("Sending file lines in batches of {} as messages.",
 batchSize);

                for (int i = 0; i < lines.size(); i += batchSize) {
                    List<MessageEntry> messageBatch = new ArrayList<>();

                    for (int j = i; j < Math.min(i + batchSize, lines.size()); j+
+) {
                        String line = lines.get(j);
                        if (line == null || line.trim().isEmpty()) {
                            continue; // Skip empty lines.
                        }
```

```
                        Map<String, MessageAttributeValue> attributes = new
HashMap<>();
                        attributes.put("line", MessageAttributeValue.builder()
                                .dataType("String")
                                .stringValue(String.valueOf(j))
                                .build());

                        messageBatch.add(new MessageEntry(lines.get(j),
attributes));
                    }

                    sendMessages(queueUrl, messageBatch);
                    System.out.print(".");
                    System.out.flush();
                }

                LOGGER.info("\nDone. Sent {} messages.", lines.size());

                // Receive and process messages.
                LOGGER.info("Receiving, handling, and deleting messages in
batches of {}.", batchSize);
                String[] receivedLines = new String[lines.size()];
                boolean moreMessages = true;

                while (moreMessages) {
                    List<Message> receivedMessages = receiveMessages(queueUrl,
batchSize, 5);

                    for (Message message : receivedMessages) {
                        int lineNumber =
Integer.parseInt(message.messageAttributes().get("line").stringValue());
                        receivedLines[lineNumber] = message.body();
                    }

                    if (!receivedMessages.isEmpty()) {
                        deleteMessages(queueUrl, receivedMessages);
                    } else {
                        moreMessages = false;
                    }
                }

                LOGGER.info("\nDone.");
```

```
                // Verify that all lines were received correctly.
                boolean allLinesMatch = true;
                for (int i = 0; i < lines.size(); i++) {
                    String originalLine = lines.get(i);
                    String receivedLine = receivedLines[i] == null ? "" :
receivedLines[i];

                    if (!originalLine.equals(receivedLine)) {
                        allLinesMatch = false;
                        break;
                    }
                }

                if (allLinesMatch) {
                    LOGGER.info("Successfully reassembled all file lines!");
                } else {
                    LOGGER.info("Uh oh, some lines were missed!");
                }
            }
        } catch (SqsException e) {
            LOGGER.error("SQS operation failed", e);
        } catch (RuntimeException | IOException e) {
            LOGGER.error("Unexpected runtime error during demo", e);
        } finally {
            // Clean up by deleting the queue if it was created.
            if (queueUrl != null) {
                try {
                    DeleteQueueRequest deleteQueueRequest =
DeleteQueueRequest.builder()
                            .queueUrl(queueUrl)
                            .build();
                    sqsClient.deleteQueue(deleteQueueRequest);
                    LOGGER.info("Deleted queue: {}", queueUrl);
                } catch (SqsException e) {
                    LOGGER.error("Failed to delete queue: {}", queueUrl, e);
                }
            }
        }

        LOGGER.info("Thanks for watching!");
        LOGGER.info("-".repeat(88));
    }
}
```

SimpleProducerConsumer.java - Usa o agrupamento automático de mensagens.

```
package com.example.sqs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsAsyncClient;
import software.amazon.awssdk.services.sqs.batchmanager.SqsAsyncBatchManager;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageResponse;
import software.amazon.awssdk.core.exception.SdkException;

import java.math.BigInteger;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Demonstrates the AWS SDK for Java 2.x Automatic Request Batching API for
 Amazon SQS.
 *
 * This example showcases the high-level SqsAsyncBatchManager library that
 provides
 * efficient batching and buffering for SQS operations. The batch manager offers
 * methods that directly mirror SqsAsyncClient methods—sendMessage,
 changeMessageVisibility,
 * deleteMessage, and receiveMessage—making it a drop-in replacement with minimal
 code changes.
 *
 * Key features of the SqsAsyncBatchManager:
```

```
 * - Automatic batching: The SDK automatically buffers individual requests and
 sends them
 *   as batches when maxBatchSize (default: 10) or sendRequestFrequency (default:
 200ms)
 *   thresholds are reached
 * - Familiar API: Method signatures match SqsAsyncClient exactly, requiring no
 learning curve
 * - Background optimization: The batch manager maintains internal buffers and
 handles
 *   batching logic transparently
 * - Asynchronous operations: All methods return CompletableFuture for non-
blocking execution
 *
 * Performance benefits demonstrated:
 * - Reduced API calls: Multiple individual requests are consolidated into single
 batch operations
 * - Lower costs: Fewer API calls result in reduced SQS charges
 * - Higher throughput: Batch operations process more messages per second
 * - Efficient resource utilization: Fewer network round trips and better
 connection reuse
 *
 * This example compares:
 * 1. Single-message operations using SqsAsyncClient directly
 * 2. Batch operations using SqsAsyncBatchManager with identical method calls
 *
 * Usage patterns:
 * - Set batch size to 1 to use SqsAsyncClient for baseline performance
 measurement
 * - Set batch size > 1 to use SqsAsyncBatchManager for optimized batch
 processing
 * - Monitor real-time throughput metrics to observe performance improvements
 *
 * Prerequisites:
 * - AWS SDK for Java 2.x version 2.28.0 or later
 * - An existing SQS queue
 * - Valid AWS credentials configured
 *
 * The program displays real-time metrics showing the dramatic performance
 difference
 * between individual operations and automatic batching.
 */
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
```

```
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Logger log =
LoggerFactory.getLogger(SimpleProducerConsumer.class);

    /**
     * Runs the SQS batching demonstration with user-configured parameters.
     *
     * Prompts for queue name, thread counts, batch size, message size, and
runtime.
     * Creates producer and consumer threads to demonstrate batching performance.
     *
     * @param args command line arguments (not used)
     * @throws InterruptedException if thread operations are interrupted
     */
    public static void main(String[] args) throws InterruptedException {

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
        final int producerCount = input.nextInt();

        System.out.print("Enter the number of consumers: ");
        final int consumerCount = input.nextInt();

        System.out.print("Enter the number of messages per batch: ");
        final int batchSize = input.nextInt();

        System.out.print("Enter the message size in bytes: ");
        final int messageSizeByte = input.nextInt();

        System.out.print("Enter the run time in minutes: ");
        final int runTimeMinutes = input.nextInt();

        // Create SQS async client and batch manager for all operations.
        // The SqsAsyncBatchManager is created from the SqsAsyncClient using the
        // batchManager() factory method, which provides default batching
configuration.
        // This high-level library automatically handles request buffering and
batching
        // while maintaining the same method signatures as SqsAsyncClient.
        final SqsAsyncClient sqsAsyncClient = SqsAsyncClient.create();
```

```
        final SqsAsyncBatchManager batchManager = sqsAsyncClient.batchManager();

        final String queueUrl =
sqsAsyncClient.getQueueUrl(GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build()).join().queueUrl();

        // The flag used to stop producer, consumer, and monitor threads.
        final AtomicBoolean stop = new AtomicBoolean(false);

        // Start the producers.
        final AtomicInteger producedCount = new AtomicInteger();
        final Thread[] producers = new Thread[producerCount];
        for (int i = 0; i < producerCount; i++) {
            if (batchSize == 1) {
                producers[i] = new Producer(sqsAsyncClient, queueUrl,
messageSizeByte,
                        producedCount, stop);
            } else {
                producers[i] = new BatchProducer(batchManager, queueUrl,
batchSize,
                        messageSizeByte, producedCount, stop);
            }
            producers[i].start();
        }

        // Start the consumers.
        final AtomicInteger consumedCount = new AtomicInteger();
        final Thread[] consumers = new Thread[consumerCount];
        for (int i = 0; i < consumerCount; i++) {
            if (batchSize == 1) {
                consumers[i] = new Consumer(sqsAsyncClient, queueUrl,
consumedCount, stop);
            } else {
                consumers[i] = new BatchConsumer(batchManager, queueUrl,
batchSize,
                        consumedCount, stop);
            }
            consumers[i].start();
        }

        // Start the monitor thread.
        final Thread monitor = new Monitor(producedCount, consumedCount, stop);
        monitor.start();
```

```
        // Wait for the specified amount of time then stop.
        Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runTimeMinutes,
                MAX_RUNTIME_MINUTES)));
        stop.set(true);

        // Join all threads.
        for (int i = 0; i < producerCount; i++) {
            producers[i].join();
        }

        for (int i = 0; i < consumerCount; i++) {
            consumers[i].join();
        }

        monitor.interrupt();
        monitor.join();

        // Close resources
        batchManager.close();
        sqsAsyncClient.close();
    }

    /**
     * Creates a random string of approximately the specified size in bytes.
     *
     * @param sizeByte the target size in bytes for the generated string
     * @return a random string encoded in base-32
     */
    private static String makeRandomString(int sizeByte) {
        final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
        new Random().nextBytes(bs);
        bs[0] = (byte) ((bs[0] | 64) & 127);
        return new BigInteger(bs).toString(32);
    }

    /**
     * Sends messages individually using SqsAsyncClient for baseline performance
measurement.
     *
     * This producer demonstrates traditional single-message operations without
batching.
     * Each sendMessage() call results in a separate API request to SQS,
providing
```

```
    * a performance baseline for comparison with the batch operations.
    *
    * The sendMessage() method signature is identical to
SqsAsyncBatchManager.sendMessage(),
    * showing how the high-level batching library maintains API compatibility
while
    * adding automatic optimization behind the scenes.
    */
   private static class Producer extends Thread {
       final SqsAsyncClient sqsAsyncClient;
       final String queueUrl;
       final AtomicInteger producedCount;
       final AtomicBoolean stop;
       final String theMessage;

       /**
        * Creates a producer thread for single-message operations.
        *
        * @param sqsAsyncClient the SQS client for sending messages
        * @param queueUrl the URL of the target queue
        * @param messageSizeByte the size of messages to generate
        * @param producedCount shared counter for tracking sent messages
        * @param stop shared flag to signal thread termination
        */
       Producer(SqsAsyncClient sqsAsyncClient, String queueUrl, int
messageSizeByte,
                AtomicInteger producedCount, AtomicBoolean stop) {
           this.sqsAsyncClient = sqsAsyncClient;
           this.queueUrl = queueUrl;
           this.producedCount = producedCount;
           this.stop = stop;
           this.theMessage = makeRandomString(messageSizeByte);
       }

       /**
        * Continuously sends messages until the stop flag is set.
        *
        * Uses SqsAsyncClient.sendMessage() directly, resulting in one API call
per message.
        * This approach provides baseline performance metrics for comparison
with batching.
        * Each call blocks until the individual message is sent, demonstrating
traditional
        * one-request-per-operation behavior.
```

```
         */
        public void run() {
            try {
                while (!stop.get()) {
                    sqsAsyncClient.sendMessage(SendMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .messageBody(theMessage)
                            .build()).join();
                    producedCount.incrementAndGet();
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Producer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Sends messages using SqsAsyncBatchManager for automatic request batching
and optimization.
     *
     * This producer demonstrates the AWS SDK for Java 2.x high-level batching
library.
     * The SqsAsyncBatchManager automatically buffers individual sendMessage()
calls and
     * sends them as batches when thresholds are reached:
     * - maxBatchSize: Maximum 10 messages per batch (default)
     * - sendRequestFrequency: 200ms timeout before sending partial batches
(default)
     *
     * Key advantages of the batching approach:
     * - Identical API: batchManager.sendMessage() has the same signature as
sqsAsyncClient.sendMessage()
     * - Automatic optimization: No code changes needed to benefit from batching
     * - Transparent buffering: The SDK handles batching logic internally
     * - Reduced API calls: Multiple messages sent in single batch requests
     * - Lower costs: Fewer API calls result in reduced SQS charges
     * - Higher throughput: Batch operations process significantly more messages
per second
     */
    private static class BatchProducer extends Thread {
```

```
        final SqsAsyncBatchManager batchManager;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for batch operations.
         *
         * @param batchManager the batch manager for efficient message sending
         * @param queueUrl the URL of the target queue
         * @param batchSize the number of messages to send per batch
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        BatchProducer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      int messageSizeByte, AtomicInteger producedCount,
                      AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends batches of messages using the high-level batching
library.
         *
         * Notice how batchManager.sendMessage() uses the exact same method
signature
         * and request builder pattern as SqsAsyncClient.sendMessage(). This
demonstrates
         * the drop-in replacement capability of the SqsAsyncBatchManager.
         *
         * The SDK automatically:
         * - Buffers individual sendMessage() calls internally
         * - Groups them into batch requests when thresholds are met
         * - Sends SendMessageBatchRequest operations to SQS
         * - Returns individual CompletableFuture responses for each message
```

```
        *
        * This transparent batching provides significant performance
improvements
        * without requiring changes to application logic or error handling
patterns.
        */
       public void run() {
           try {
               while (!stop.get()) {
                   // Send multiple messages using the high-level batch manager.
                   // Each batchManager.sendMessage() call uses identical syntax
to
                   // sqsAsyncClient.sendMessage(), demonstrating API
compatibility.
                   // The SDK automatically buffers these calls and sends them
as
                   // batch operations when maxBatchSize (10) or
sendRequestFrequency (200ms)
                   // thresholds are reached, significantly improving
throughput.
                   for (int i = 0; i < batchSize; i++) {
                       CompletableFuture<SendMessageResponse> future =
batchManager.sendMessage(
                               SendMessageRequest.builder()
                                       .queueUrl(queueUrl)
                                       .messageBody(theMessage)
                                       .build());

                       // Handle the response asynchronously
                       future.whenComplete((response, throwable) -> {
                           if (throwable == null) {
                               producedCount.incrementAndGet();
                           } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                   !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                               log.error("BatchProducer: Failed to send
message", throwable);
                           }
                           // Ignore CancellationException and executor shutdown
errors - expected during shutdown
                       });
                   }
```

```
                    // Small delay to allow batching to occur
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchProducer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                log.error("BatchProducer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Receives and deletes messages individually using SqsAsyncClient for
baseline measurement.
     *
     * This consumer demonstrates traditional single-message operations without
batching.
     * Each receiveMessage() and deleteMessage() call results in separate API
requests,
     * providing a performance baseline for comparison with batch operations.
     *
     * The method signatures are identical to SqsAsyncBatchManager methods:
     * - receiveMessage() matches batchManager.receiveMessage()
     * - deleteMessage() matches batchManager.deleteMessage()
     *
     * This API consistency allows easy migration to the high-level batching
library.
     */
    private static class Consumer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger consumedCount;
        final AtomicBoolean stop;

        /**
         * Creates a consumer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for receiving messages
         * @param queueUrl the URL of the source queue
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
```

```
        Consumer(SqsAsyncClient sqsAsyncClient, String queueUrl, AtomicInteger
 consumedCount,
                 AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using traditional single-
request operations.
         *
         * Uses SqsAsyncClient methods directly:
         * - receiveMessage(): One API call per receive operation
         * - deleteMessage(): One API call per delete operation
         *
         * This approach demonstrates the baseline performance without batching
 optimization.
         * Compare these method calls with the identical signatures used in
 BatchConsumer
         * to see how the high-level batching library maintains API
 compatibility.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    try {
                        final ReceiveMessageResponse result =
 sqsAsyncClient.receiveMessage(
                                ReceiveMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .build()).join();

                        if (!result.messages().isEmpty()) {
                            final Message m = result.messages().get(0);
                            // Note: deleteMessage() signature identical to
 batchManager.deleteMessage()

 sqsAsyncClient.deleteMessage(DeleteMessageRequest.builder()
                                    .queueUrl(queueUrl)
                                    .receiptHandle(m.receiptHandle())
                                    .build()).join();
                            consumedCount.incrementAndGet();
```

```
                        }
                    } catch (SdkException |
java.util.concurrent.CompletionException e) {
                        log.error(e.getMessage());
                    }
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Consumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Receives and deletes messages using SqsAsyncBatchManager for automatic
optimization.
     *
     * This consumer demonstrates the AWS SDK for Java 2.x high-level batching
library
     * for message consumption. The SqsAsyncBatchManager provides two key
optimizations:
     *
     * 1. Receive optimization: Maintains an internal buffer of messages fetched
in the
     *    background, so receiveMessage() calls return immediately from the
buffer
     * 2. Delete batching: Automatically buffers deleteMessage() calls and sends
them
     *    as DeleteMessageBatchRequest operations when thresholds are reached
     *
     * Key features:
     * - Identical API: receiveMessage() and deleteMessage() have the same
signatures
     *   as SqsAsyncClient methods, making this a true drop-in replacement
     * - Background fetching: The batch manager continuously fetches messages to
keep
     *   the internal buffer populated, reducing receive latency
     * - Automatic delete batching: Individual deleteMessage() calls are buffered
and
     *   sent as batch operations (up to 10 per batch, 200ms frequency)
     * - Transparent optimization: No application logic changes needed to benefit
```

```
     *
     * Performance benefits:
     * - Reduced API calls through automatic batching of delete operations
     * - Lower latency for receives due to background message buffering
     * - Higher overall throughput with fewer network round trips
     */
    private static class BatchConsumer extends Thread {
        final SqsAsyncBatchManager batchManager;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger consumedCount;
        final AtomicBoolean stop;

        /**
         * Creates a consumer thread for batch operations.
         *
         * @param batchManager the batch manager for efficient message processing
         * @param queueUrl the URL of the source queue
         * @param batchSize the maximum number of messages to receive per batch
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
        BatchConsumer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      AtomicInteger consumedCount, AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using the high-level
batching library.
         *
         * Demonstrates the key advantage of SqsAsyncBatchManager: identical
method signatures
         * with automatic optimization. Notice how:
         *
         * - batchManager.receiveMessage() uses the same syntax as
sqsAsyncClient.receiveMessage()
         * - batchManager.deleteMessage() uses the same syntax as
sqsAsyncClient.deleteMessage()
```

```
        *
        * Behind the scenes, the batch manager:
        * 1. Maintains an internal message buffer populated by background
fetching
        * 2. Returns messages immediately from the buffer (reduced latency)
        * 3. Automatically batches deleteMessage() calls into
DeleteMessageBatchRequest operations
        * 4. Sends batch deletes when maxBatchSize (10) or sendRequestFrequency
(200ms) is reached
        *
        * This provides significant performance improvements with zero code
changes
        * compared to traditional SqsAsyncClient usage patterns.
        */
       public void run() {
           try {
               while (!stop.get()) {
                   // Receive messages using the high-level batch manager.
                   // This call uses identical syntax to
sqsAsyncClient.receiveMessage()
                   // but benefits from internal message buffering for improved
performance.
                   final ReceiveMessageResponse result =
batchManager.receiveMessage(
                           ReceiveMessageRequest.builder()
                                   .queueUrl(queueUrl)
                                   .maxNumberOfMessages(Math.min(batchSize, 10))
                                   .build()).join();

                   if (!result.messages().isEmpty()) {
                       final List<Message> messages = result.messages();

                       // Delete messages using the batch manager.
                       // Each deleteMessage() call uses identical syntax to
SqsAsyncClient
                       // but the SDK automatically buffers these calls and
sends them
                       // as DeleteMessageBatchRequest operations for optimal
performance.
                       for (Message message : messages) {
                           CompletableFuture<DeleteMessageResponse> future =
batchManager.deleteMessage(
                                   DeleteMessageRequest.builder()
                                           .queueUrl(queueUrl)
```

```
.receiptHandle(message.receiptHandle())
                                            .build());

                            future.whenComplete((response, throwable) -> {
                                if (throwable == null) {
                                    consumedCount.incrementAndGet();
                                } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                        !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                    log.error("BatchConsumer: Failed to delete
message", throwable);
                                }
                                // Ignore CancellationException and executor
shutdown errors - expected during shutdown
                            });
                        }
                    }

                    // Small delay to prevent tight polling
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchConsumer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("BatchConsumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Displays real-time throughput statistics every second.
     *
     * This thread logs the current count of produced and consumed messages
     * to help you monitor the performance comparison.
     */
    private static class Monitor extends Thread {
        private final AtomicInteger producedCount;
```

```
        private final AtomicInteger consumedCount;
        private final AtomicBoolean stop;

        /**
         * Creates a monitoring thread that displays throughput statistics.
         *
         * @param producedCount shared counter for messages sent
         * @param consumedCount shared counter for messages processed
         * @param stop shared flag to signal thread termination
         */
        Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
                AtomicBoolean stop) {
            this.producedCount = producedCount;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Logs throughput statistics every second until stopped.
         *
         * Displays the current count of produced and consumed messages
         * to help monitor the performance comparison between batching
 strategies.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    Thread.sleep(1000);
                    log.info("produced messages = " + producedCount.get()
                            + ", consumed messages = " + consumedCount.get());
                }
            } catch (InterruptedException e) {
                // Allow the thread to exit.
            }
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue

- DeleteMessage
- DeleteMessageBatch
- DeleteQueue
- ReceiveMessage
- SendMessage
- SendMessageBatch

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar funções para encapsular funções de mensagem do Amazon SQS.

```
import logging
import sys

import boto3
from botocore.exceptions import ClientError

import queue_wrapper

logger = logging.getLogger(__name__)
sqs = boto3.resource("sqs")

def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not
 sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
```

```
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and
 failed
             messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
            for ind, msg in enumerate(messages)
        ]
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response



def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
```

```
    :param max_number: The maximum number of messages to receive. The actual
 number
                        of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning.
 When
                       this number is greater than zero, long polling is used.
 This
                       can result in reduced costs and fewer false empty
 responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return messages



def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and
 failed
             message deletions.
    """
    try:
        entries = [
            {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
            for ind, msg in enumerate(messages)
        ]
        response = queue.delete_messages(Entries=entries)
```

```
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info("Deleted %s",
 messages[int(msg_meta["Id"])].receipt_handle)
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Could not delete %s",
 messages[int(msg_meta["Id"])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

Use as funções de wrapper para enviar e receber mensagens em lotes.

```
def usage_demo():
    """
    Shows how to:
    * Read the lines from this Python file and send the lines in
      batches of 10 as messages to a queue.
    * Receive the messages in batches until the queue is empty.
    * Reassemble the lines of the file and verify they match the original file.
    """

    def pack_message(msg_path, msg_body, msg_line):
        return {
            "body": msg_body,
            "attributes": {
                "path": {"StringValue": msg_path, "DataType": "String"},
                "line": {"StringValue": str(msg_line), "DataType": "String"},
            },
        }

    def unpack_message(msg):
        return (
            msg.message_attributes["path"]["StringValue"],
            msg.body,
            int(msg.message_attributes["line"]["StringValue"]),
```

```
        )

    print("-" * 88)
    print("Welcome to the Amazon Simple Queue Service (Amazon SQS) demo!")
    print("-" * 88)

    queue = queue_wrapper.create_queue("sqs-usage-demo-message-wrapper")

    with open(__file__) as file:
        lines = file.readlines()

    line = 0
    batch_size = 10
    received_lines = [None] * len(lines)
    print(f"Sending file lines in batches of {batch_size} as messages.")
    while line < len(lines):
        messages = [
            pack_message(__file__, lines[index], index)
            for index in range(line, min(line + batch_size, len(lines)))
        ]
        line = line + batch_size
        send_messages(queue, messages)
        print(".", end="")
        sys.stdout.flush()
    print(f"Done. Sent {len(lines) - 1} messages.")

    print(f"Receiving, handling, and deleting messages in batches of
{batch_size}.")
    more_messages = True
    while more_messages:
        received_messages = receive_messages(queue, batch_size, 2)
        print(".", end="")
        sys.stdout.flush()
        for message in received_messages:
            path, body, line = unpack_message(message)
            received_lines[line] = body
        if received_messages:
            delete_messages(queue, received_messages)
        else:
            more_messages = False
    print("Done.")

    if all([lines[index] == received_lines[index] for index in
range(len(lines))]):
```

```
        print(f"Successfully reassembled all file lines!")
    else:
        print(f"Uh oh, some lines were missed!")

    queue.delete()

    print("Thanks for watching!")
    print("-" * 88)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage
  - SendMessage
  - SendMessageBatch

Use o comando AWS Estrutura de processamento de mensagens para.NET para publicar e receber mensagens do Amazon SQS

O exemplo de código a seguir mostra como criar aplicativos que publicam e recebem mensagens do Amazon SQS usando o AWS Message Processing Framework para.NET.

.NET

SDK para .NET

Fornece um tutorial para o AWS Message Processing Framework para.NET. O tutorial cria uma aplicação web que permite ao usuário publicar uma mensagem do Amazon SQS e uma aplicação de linha de comando que recebe a mensagem.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o tutorial completo no Guia do AWS SDK para .NET desenvolvedor e o exemplo em GitHub.

Serviços usados neste exemplo

- Amazon SQS

Usar a biblioteca de mensagens Java do Amazon SQS para trabalhar com a interface do Serviço de Mensagens Java (JMS) para o Amazon SQS

O exemplo de código a seguir mostra como usar a Biblioteca de Mensagens Java do Amazon SQS para trabalhar com a interface do JMS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Os exemplos a seguir funcionam com filas padrão do Amazon SQS e incluem:

- Envio de uma mensagem de texto.
- Recebimento de mensagens de forma síncrona.
- Recebimento de mensagens de forma assíncrona.
- Recebimento de mensagens usando o modo CLIENT_ACKNOWLEDGE.
- Recebimento de mensagens usando o modo UNORDERED_ACKNOWLEDGE.
- Uso do Spring para injetar dependências.
- Uma classe de utilitário que fornece métodos comuns usados pelos outros exemplos.

Consulte mais informações sobre como usar o JMS com o Amazon SQS no Guia do desenvolvedor do Amazon SQS.

Envio de uma mensagem de texto.

```
    /**
     * This method establishes a connection to a standard Amazon SQS queue using
the Amazon SQS
     * Java Messaging Library and sends text messages to it. It uses JMS (Java
Message Service) API
```

```
    * with automatic acknowledgment mode to ensure reliable message delivery,
and automatically
    * manages all messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or sending
messages to the queue
    */
   public static void doSendTextMessage() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try (SQSConnection connection = connectionFactory.createConnection()) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session that uses the JMS auto-acknowledge mode.
           Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));

           createAndSendMessages(session, producer);
       } // The connection closes automatically. This also closes the session.
       LOGGER.info("Connection closed");
   }

   /**
    * This method reads text input from the keyboard and sends each line as a
separate message
    * to a standard Amazon SQS queue using the Amazon SQS Java Messaging
Library. It continues
    * to accept input until the user enters an empty line, using JMS (Java
Message Service) API to
    * handle the message delivery.
    *
    * @param session The JMS session used to create messages
```

```
    * @param producer The JMS message producer used to send messages to the
queue
    */
   private static void createAndSendMessages(Session session, MessageProducer
producer) {
       BufferedReader inputReader = new BufferedReader(
               new InputStreamReader(System.in, Charset.defaultCharset()));

       try {
           String input;
           while (true) {
               LOGGER.info("Enter message to send (leave empty to exit): ");
               input = inputReader.readLine();
               if (input == null || input.isEmpty()) break;

               TextMessage message = session.createTextMessage(input);
               producer.send(message);
               LOGGER.info("Send message {}", message.getJMSMessageID());
           }
       } catch (EOFException e) {
           // Just return on EOF
       } catch (IOException e) {
           LOGGER.error("Failed reading input: {}", e.getMessage(), e);
       } catch (JMSException e) {
           LOGGER.error("Failed sending message: {}", e.getMessage(), e);
       }
   }
```

Recebimento de mensagens de forma síncrona.

```
   /**
    * This method receives messages from a standard Amazon SQS queue using the
Amazon SQS Java
    * Messaging Library. It creates a connection to the queue using JMS (Java
Message Service),
    * waits for messages to arrive, and processes them one at a time. The method
handles all
    * necessary setup and cleanup of messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
    */
```

```
    public static void doReceiveMessageSync() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            connection.start();

            receiveMessages(consumer);
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
    }

    /**
     * This method continuously checks for new messages from a standard Amazon
SQS queue using
     * the Amazon SQS Java Messaging Library. It waits up to 20 seconds for each
message, processes
     * it using JMS (Java Message Service), and confirms receipt. The method
stops checking for
     * messages after 20 seconds of no activity.
     *
     * @param consumer The JMS message consumer that receives messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 1 minute for a message
```

```
                Message message =
consumer.receive(Duration.ofSeconds(20).toMillis());
                if (message == null) {
                    LOGGER.info("Shutting down after 20 seconds of silence.");
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Acknowledged message {}",
message.getJMSMessageID());
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS: {}", e.getMessage(), e);
        }
    }
```

Recebimento de mensagens de forma assíncrona.

```
    /**
     * This method sets up automatic message handling for a standard Amazon SQS
queue using the
     * Amazon SQS Java Messaging Library. It creates a listener that processes
messages as soon
     * as they arrive using JMS (Java Message Service), runs for 5 seconds, then
cleans up all
     * messaging resources.
     *
     * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
     */
    public static void doReceiveMessageAsync() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
```

```
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

            try {
                // Create a consumer for the queue.
                MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
                // Provide an implementation of the MessageListener interface,
which has a single 'onMessage' method.
                // We use a lambda expression for the implementation.
                consumer.setMessageListener(message -> {
                    try {
                        SqsJmsExampleUtils.handleMessage(message);
                        message.acknowledge();
                    } catch (JMSException e) {
                        LOGGER.error("Error processing message: {}",
e.getMessage());
                    }
                });
                // Start receiving incoming messages.
                connection.start();
                LOGGER.info("Waiting for messages...");
            } catch (JMSException e) {
                throw new RuntimeException(e);
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info( "Connection closed" );
    }
```

Recebimento de mensagens usando o modo CLIENT_ACKNOWLEDGE.

```
    /**
```

```
    * This method demonstrates how message acknowledgment affects message
processing in a standard
    * Amazon SQS queue using the Amazon SQS Java Messaging Library. It sends
messages to the queue,
    * then shows how JMS (Java Message Service) client acknowledgment mode
handles both explicit
    * and implicit message confirmations, including how acknowledging one
message can automatically
    * acknowledge previous messages.
    *
    * @throws JMSException If there is a problem with the messaging operations
    */
   public static void doReceiveMessagesSyncClientAcknowledge() throws
JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

           // Create a session with client acknowledge mode.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

           // Create a producer and consumer.
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
           MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

           // Open the connection.
           connection.start();

           // Send two text messages.
           sendMessage(producer, session, "Message 1");
           sendMessage(producer, session, "Message 2");
```

```
            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue,
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout",
e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }

            /*  We will attempt to receive another message, but none will be
available. This is because in
                CLIENT_ACKNOWLEDGE mode, when we acknowledged the second message,
all previous messages were
                automatically acknowledged as well. Therefore, although we never
directly acknowledged the first
                message, it was implicitly acknowledged when we confirmed the
second one. */
            receiveMessage(consumer, true);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed.");

    }


    /**
     * Sends a text message using the specified JMS MessageProducer and Session.
     *
     * @param producer    The JMS MessageProducer used to send the message
     * @param session     The JMS Session used to create the text message
     * @param messageText The text content to be sent in the message
     * @throws JMSException If there is an error creating or sending the message
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
```

```
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }

    /**
     * Receives and processes a message from a JMS queue using the specified
consumer.
     * The method waits for a message until the configured timeout period is
reached.
     * If a message is received, it is logged and optionally acknowledged based
on the
     * acknowledge parameter.
     *
     * @param consumer    The JMS MessageConsumer used to receive messages from
the queue
     * @param acknowledge Boolean flag indicating whether to acknowledge the
message.
     *                    If true, the message will be acknowledged after
processing
     * @throws JMSException If there is an error receiving, processing, or
acknowledging the message
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object
and print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

Recebimento de mensagens usando o modo UNORDERED_ACKNOWLEDGE.

```
   /**
    * Demonstrates message acknowledgment behavior in UNORDERED_ACKNOWLEDGE mode
with Amazon SQS JMS.
    * In this mode, each message must be explicitly acknowledged regardless of
receive order.
    * Unacknowledged messages return to the queue after the visibility timeout
expires,
    * unlike CLIENT_ACKNOWLEDGE mode where acknowledging one message
acknowledges all previous messages.
    *
    * @throws JMSException         If a JMS-related error occurs during message
operations
    */
   public static void doReceiveMessagesUnorderedAcknowledge() throws
JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try( SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

           // Create a session with unordered acknowledge mode.
           Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

           // Create the producer and consumer.
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
           MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

           // Open a connection.
           connection.start();

           // Send two text messages.
```

```
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue.
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout",
e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }

            /*  We will attempt to receive another message, and we'll get the
first message again. This occurs
                because in UNORDERED_ACKNOWLEDGE mode, each message requires its
own separate acknowledgment.
                Since we only acknowledged the second message, the first message
remains in the queue for
                redelivery. */
            receiveMessage(consumer, true);

            LOGGER.info("Connection closed.");
        } // The connection closes automatically. This also closes the session.
    }

    /**
     * Sends a text message to an Amazon SQS queue using JMS.
     *
     * @param producer    The JMS MessageProducer for the queue
     * @param session     The JMS Session for message creation
     * @param messageText The message content
     * @throws JMSException If message creation or sending fails
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
```

```
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }
    /**
     * Synchronously receives a message from an Amazon SQS queue using the JMS
API
     * with an acknowledgment parameter.
     *
     * @param consumer    The JMS MessageConsumer for the queue
     * @param acknowledge If true, acknowledges the message after receipt
     * @throws JMSException If message reception or acknowledgment fails
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object
and print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

Uso do Spring para injetar dependências.

```
package com.example.sqs.jms.spring;

import com.amazon.sqs.javamessaging.SQSConnection;
import com.example.sqs.jms.SqsJmsExampleUtils;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.NoSuchBeanDefinitionException;
import org.springframework.context.support.FileSystemXmlApplicationContext;
```

```
import java.io.File;
import java.net.URL;
import java.util.concurrent.TimeUnit;

/**
 * Demonstrates how to send and receive messages using the Amazon SQS Java
 Messaging Library
 * with Spring Framework integration. This example connects to a standard Amazon
 SQS message
 * queue using Spring's dependency injection to configure the connection and
 messaging components.
 * The application uses the JMS (Java Message Service) API to handle message
 operations.
 */
public class SpringExample {
    private static final Integer POLLING_SECONDS = 15;
    private static final String SPRING_XML_CONFIG_FILE =
 "SpringExampleConfiguration.xml.txt";
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SpringExample.class);

    /**
     * Demonstrates sending and receiving messages through a standard Amazon SQS
 message queue
     * using Spring Framework configuration. This method loads connection
 settings from an XML file,
     * establishes a messaging session using the Amazon SQS Java Messaging
 Library, and processes
     * messages using JMS (Java Message Service) operations. If the queue doesn't
 exist, it will
     * be created automatically.
     *
     * @param args Command line arguments (not used)
     */
    public static void main(String[] args) {

        URL resource =
 SpringExample.class.getClassLoader().getResource(SPRING_XML_CONFIG_FILE);
        File springFile = new File(resource.getFile());
        if (!springFile.exists() || !springFile.canRead()) {
            LOGGER.error("File " + SPRING_XML_CONFIG_FILE + " doesn't exist or
 isn't readable.");
            System.exit(1);
        }
```

```
        try (FileSystemXmlApplicationContext context =
                    new FileSystemXmlApplicationContext("file://" +
springFile.getAbsolutePath())) {

            Connection connection;
            try {
                connection = context.getBean(Connection.class);
            } catch (NoSuchBeanDefinitionException e) {
                LOGGER.error("Can't find the JMS connection to use: " +
e.getMessage(), e);
                System.exit(2);
                return;
            }

            String queueName;
            try {
                queueName = context.getBean("queueName", String.class);
            } catch (NoSuchBeanDefinitionException e) {
                LOGGER.error("Can't find the name of the queue to use: " +
e.getMessage(), e);
                System.exit(3);
                return;
            }
            try {
                if (connection instanceof SQSConnection) {
                    SqsJmsExampleUtils.ensureQueueExists((SQSConnection)
connection, queueName, SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
                }
                // Create the JMS session.
                Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

                SqsJmsExampleUtils.sendTextMessage(session, queueName);
                MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));

                receiveMessages(consumer);
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        }   // Spring context autocloses. Managed Spring beans that implement
AutoClosable, such as the
```

```
        // 'connection' bean, are also closed.
        LOGGER.info("Context closed");
    }

    /**
     * Continuously checks for and processes messages from a standard Amazon SQS
 message queue
     * using the Amazon SQS Java Messaging Library underlying the JMS API. This
 method waits for incoming messages,
     * processes them when they arrive, and acknowledges their receipt using JMS
 (Java Message
     * Service) operations. The method will stop checking for messages after 15
 seconds of
     * inactivity.
     *
     * @param consumer The JMS message consumer used to receive messages from the
 queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 15 seconds for a message.
                Message message =
 consumer.receive(TimeUnit.SECONDS.toMillis(POLLING_SECONDS));
                if (message == null) {
                    LOGGER.info("Shutting down after {} seconds of silence.",
 POLLING_SECONDS);
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Message acknowledged.");
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS.", e);
        }
    }
}
```

Definições de bean do Spring.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/
schema/beans/spring-beans-3.0.xsd
        ">
    <!-- Define the AWS Region -->
    <bean id="region" class="software.amazon.awssdk.regions.Region" factory-
method="of">
        <constructor-arg value="us-east-1"/>
    </bean>

    <bean id="credentialsProviderBean"
 class="software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider"
          factory-method="create"/>

    <bean id="clientBuilder"
 class="software.amazon.awssdk.services.sqs.SqsClient" factory-method="builder"/>

    <bean id="regionSetClientBuilder" factory-bean="clientBuilder" factory-
method="region">
        <constructor-arg ref="region"/>
    </bean>

    <!-- Configure the Builder with Credentials Provider -->
    <bean id="sqsClient" factory-bean="regionSetClientBuilder" factory-
method="credentialsProvider">
        <constructor-arg ref="credentialsProviderBean"/>
    </bean>

    <bean id="providerConfiguration"
 class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="connectionFactory"
 class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="providerConfiguration"/>
        <constructor-arg ref="clientBuilder"/>
    </bean>
```

```
    <bean id="connection"
          factory-bean="connectionFactory"
          factory-method="createConnection"
          init-method="start"
          destroy-method="close"/>

    <bean id="queueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
    </bean>
</beans>
```

Uma classe de utilitário que fornece métodos comuns usados pelos outros exemplos.

```
package com.example.sqs.jms;

import com.amazon.sqs.javamessaging.AmazonSQSMessagingClientWrapper;
import com.amazon.sqs.javamessaging.ProviderConfiguration;
import com.amazon.sqs.javamessaging.SQSConnection;
import com.amazon.sqs.javamessaging.SQSConnectionFactory;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;

import java.time.Duration;
import java.util.Base64;
import java.util.Map;

/**
 * This utility class provides helper methods for working with Amazon Simple
 Queue Service (Amazon SQS)
 * through the Java Message Service (JMS) interface. It contains common
 operations for managing message
 * queues and handling message delivery.
 */
public class SqsJmsExampleUtils {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SqsJmsExampleUtils.class);
```

```
   public static final Long QUEUE_VISIBILITY_TIMEOUT = 5L;

   /**
    * This method verifies that a message queue exists and creates it if
necessary. The method checks for
    * an existing queue first to optimize performance.
    *
    * @param connection The active connection to the messaging service
    * @param queueName The name of the queue to verify or create
    * @param visibilityTimeout The duration in seconds that messages will be
hidden after being received
    * @throws JMSException If there is an error accessing or creating the queue
    */
   public static void ensureQueueExists(SQSConnection connection, String
queueName, Long visibilityTimeout) throws JMSException {
       AmazonSQSMessagingClientWrapper client =
connection.getWrappedAmazonSQSClient();

      /* In most cases, you can do this with just a 'createQueue' call, but
'getQueueUrl'
      (called by 'queueExists') is a faster operation for the common case where
the queue
      already exists. Also, many users and roles have permission to call
'getQueueUrl'
      but don't have permission to call 'createQueue'.
      */
       if( !client.queueExists(queueName) ) {
           CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .attributes(Map.of(QueueAttributeName.VISIBILITY_TIMEOUT,
String.valueOf(visibilityTimeout)))
                   .build();
           client.createQueue( createQueueRequest );
       }
   }

   /**
    * This method sends a simple text message to a specified message queue. It
handles all necessary
    * setup for the message delivery process.
    *
    * @param session The active messaging session used to create and send the
message
    * @param queueName The name of the queue where the message will be sent
```

```
     */
    public static void sendTextMessage(Session session, String queueName) {
        // Rest of implementation...

        try {
            MessageProducer producer =
session.createProducer( session.createQueue( queueName) );
            Message message = session.createTextMessage("Hello world!");
            producer.send(message);
        } catch (JMSException e) {
            LOGGER.error( "Error receiving from SQS", e );
        }
    }

    /**
     * This method processes incoming messages and logs their content based on
the message type.
     * It supports text messages, binary data, and Java objects.
     *
     * @param message The message to be processed and logged
     * @throws JMSException If there is an error reading the message content
     */
    public static void handleMessage(Message message) throws JMSException {
        // Rest of implementation...
        LOGGER.info( "Got message {}", message.getJMSMessageID() );
        LOGGER.info( "Content: ");
        if(message instanceof TextMessage txtMessage) {
            LOGGER.info( "\t{}", txtMessage.getText() );
        } else if(message instanceof BytesMessage byteMessage){
            // Assume the length fits in an int - SQS only supports sizes up to
256k so that
            // should be true
            byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
            byteMessage.readBytes(bytes);
            LOGGER.info( "\t{}", Base64.getEncoder().encodeToString( bytes ) );
        } else if( message instanceof ObjectMessage) {
            ObjectMessage objMessage = (ObjectMessage) message;
            LOGGER.info( "\t{}", objMessage.getObject() );
        }
    }

    /**
     * This method sets up automatic message processing for a specified queue. It
creates a listener
```

```
    * that will receive and handle incoming messages without blocking the main
program.
    *
    * @param session The active messaging session
    * @param queueName The name of the queue to monitor
    * @param connection The active connection to the messaging service
    */
   public static void receiveMessagesAsync(Session session, String queueName,
Connection connection) {
       // Rest of implementation...
       try {
           // Create a consumer for the queue.
           MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));
           // Provide an implementation of the MessageListener interface, which
has a single 'onMessage' method.
           // We use a lambda expression for the implementation.
           consumer.setMessageListener(message -> {
               try {
                   SqsJmsExampleUtils.handleMessage(message);
                   message.acknowledge();
               } catch (JMSException e) {
                   LOGGER.error("Error processing message: {}", e.getMessage());
               }
           });
           // Start receiving incoming messages.
           connection.start();
       } catch (JMSException e) {
           throw new RuntimeException(e);
       }
       try {
           Thread.sleep(2000);
       } catch (InterruptedException e) {
           throw new RuntimeException(e);
       }
   }


   /**
    * This method performs cleanup operations after message processing is
complete. It receives
    * any messages in the specified queue, removes the message queue and closes
all
    * active connections to prevent resource leaks.
```

```
     *
     * @param queueName The name of the queue to be removed
     * @param visibilityTimeout The duration in seconds that messages are hidden
after being received
     * @throws JMSException If there is an error during the cleanup process
     */
    public static void cleanUpExample(String queueName, Long visibilityTimeout)
throws JMSException {
        LOGGER.info("Performing cleanup.");

        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        try (SQSConnection connection = connectionFactory.createConnection() ) {
            ensureQueueExists(connection, queueName, visibilityTimeout);
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

            receiveMessagesAsync(session, queueName, connection);

            SqsClient sqsClient =
connection.getWrappedAmazonSQSClient().getAmazonSQSClient();
            try {
                String queueUrl = sqsClient.getQueueUrl(b ->
b.queueName(queueName)).queueUrl();
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue deleted: {}", queueUrl);
            } catch (SdkException e) {
                LOGGER.error("Error during SQS operations: ", e);
            }
        }
        LOGGER.info("Clean up: Connection closed");
    }

    /**
     * This method creates a background task that sends multiple messages to a
specified queue
     * after waiting for a set time period. The task operates independently to
ensure efficient
     * message processing without interrupting other operations.
     *
     * @param queueName The name of the queue where messages will be sent
```

```
     * @param secondsToWait The number of seconds to wait before sending messages
     * @param numMessages The number of messages to send
     * @param visibilityTimeout The duration in seconds that messages remain
 hidden after being received
     * @return A task that can be executed to send the messages
     */
    public static Runnable sendAMessageAsync(String queueName, Long
 secondsToWait, Integer numMessages, Long visibilityTimeout) {
        return () -> {
            try {
                Thread.sleep(Duration.ofSeconds(secondsToWait).toMillis());
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException(e);
            }
            try {
                SQSConnectionFactory connectionFactory = new
 SQSConnectionFactory(
                        new ProviderConfiguration(),
                        SqsClient.create()
                );
                try (SQSConnection connection =
 connectionFactory.createConnection()) {
                    ensureQueueExists(connection, queueName, visibilityTimeout);
                    Session session = connection.createSession(false,
 Session.CLIENT_ACKNOWLEDGE);
                    for (int i = 1; i <= numMessages; i++) {
                        MessageProducer producer =
 session.createProducer(session.createQueue(queueName));
                        producer.send(session.createTextMessage("Hello World " +
 i + "!"));
                    }
                }
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        };
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue
  - DeleteQueue

Trabalhe com tags de fila e o Amazon SQS usando um AWS SDK

O exemplo de código a seguir mostra como realizar uma operação de marcação com o Amazon SQS.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo a seguir cria tags para uma fila, lista tags e remove uma tag.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ListQueueTagsResponse;
import software.amazon.awssdk.services.sqs.model.QueueDoesNotExistException;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.Map;
import java.util.UUID;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials. For more
 * information, see the <a href="https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/get-started.html">AWS
 * SDK for Java Developer Guide</a>.
 */
```

```
public class TagExamples {
    static final SqsClient sqsClient = SqsClient.create();
    static final String queueName = "TagExamples-queue-" +
 UUID.randomUUID().toString().replace("-", "").substring(0, 20);
    private static final Logger LOGGER =
 LoggerFactory.getLogger(TagExamples.class);

    public static void main(String[] args) {
        final String queueUrl;
        try {
            queueUrl = sqsClient.createQueue(b ->
 b.queueName(queueName)).queueUrl();
            LOGGER.info("Queue created. The URL is: {}", queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because queue was not created.");
            throw new RuntimeException(e);
        }
        try {
            addTags(queueUrl);
            listTags(queueUrl);
            removeTags(queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because of an error in a method.");
        } finally {
            try {
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue successfully deleted. Program ending.");
                sqsClient.close();
            } catch (RuntimeException e) {
                LOGGER.error("Program ending.");
            } finally {
                sqsClient.close();
            }
        }
    }

    /** This method demonstrates how to use a Java Map to a tag a aueue.
     * @param queueUrl The URL of the queue to tag.
     */
    public static void addTags(String queueUrl) {
        // Build a map of the tags.
        final Map<String, String> tagsToAdd = Map.of(
                "Team", "Development",
                "Priority", "Beta",
```

```
                "Accounting ID", "456def");

        try {
            // Add tags to the queue using a Consumer<TagQueueRequest.Builder>
parameter.
            sqsClient.tagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tags(tagsToAdd)
            );
        } catch (QueueDoesNotExistException e) {
            LOGGER.error("Queue does not exist: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }

    /** This method demonstrates how to view the tags for a queue.
     * @param queueUrl The URL of the queue whose tags you want to list.
     */
    public static void listTags(String queueUrl) {
        ListQueueTagsResponse response;
        try {
            // Call the listQueueTags method with a
Consumer<ListQueueTagsRequest.Builder> parameter that creates a
ListQueueTagsRequest.
            response = sqsClient.listQueueTags(b -> b
                    .queueUrl(queueUrl));
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }

        // Log the tags.
        response.tags()
                .forEach((k, v) ->
                        LOGGER.info("Key: {} -> Value: {}", k, v));
    }

    /**
     * This method demonstrates how to remove tags from a queue.
     * @param queueUrl The URL of the queue whose tags you want to remove.
     */
    public static void removeTags(String queueUrl) {
        try {
```

```
            // Call the untagQueue method with a
 Consumer<UntagQueueRequest.Builder> parameter.
            sqsClient.untagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tagKeys("Accounting ID") // Remove a single tag.
            );
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - ListQueueTags
  - TagQueue
  - UntagQueue

Exemplos de tecnologia sem servidor para o Amazon SQS

Os exemplos de código a seguir mostram como usar o Amazon SQS com AWS SDKs.

Exemplos

- Invocar uma função do Lambda em um trigger do Amazon SQS
- Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

Invocar uma função do Lambda em um trigger do Amazon SQS

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }

        context.Logger.LogInformation("done");
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
```

```
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }

    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
 for _, record := range event.Records {
  err := processMessage(record)
  if err != nil {
   return err
  }
```

```
 }
 fmt.Println("done")
 return nil
}

func processMessage(record events.SQSMessage) error {
 fmt.Printf("Processed message %s\n", record.Body)
 // TODO: Do interesting work based on the new message
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
```

```
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }

    }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento SQS com o JavaScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message) {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
```

```
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Consumindo um evento SQS com o TypeScript Lambda usando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";

export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
 record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

Os exemplos de código a seguir mostram como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
namespace sqsSample;

public class Function
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
 ILambdaContext context)
    {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
 SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
```

```
        }
        return new SQSBatchResponse(batchItemFailures);
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
        context.Logger.LogInformation($"Processed message {message.Body}");
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

Go

SDK para Go V2

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent)
 (map[string]interface{}, error) {
```

```
 batchItemFailures := []map[string]interface{}{}

 for _, message := range sqsEvent.Records {
  if len(message.Body) > 0 {
   // Your message processing condition here
   fmt.Printf("Successfully processed message: %s\n", message.Body)
  } else {
   // Message processing failed
   fmt.Printf("Failed to process message %s\n", message.MessageId)
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": message.MessageId})
  }
 }

 sqsBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return sqsBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
             }
         }
         return new SQSBatchResponse(batchItemFailures);
     }
}
```

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatando falhas de itens em lote do SQS com o uso do JavaScript Lambda.

```
// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
```

```
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
    console.log(`Processed message: ${record.body}`);
}
```

Relatando falhas de itens em lote do SQS com o uso do TypeScript Lambda.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord }
 from 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
 Promise<SQSBatchResponse> => {
    const batchItemFailures: SQSBatchItemFailure[] = [];

    for (const record of event.Records) {
        try {
            await processMessageAsync(record);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }

    return {batchItemFailures: batchItemFailures};
};

async function processMessageAsync(record: SQSRecord): Promise<void> {
    if (record.body && record.body.includes("error")) {
        throw new Error('There is an error in the SQS Message.');
    }
    console.log(`Processed message ${record.body}`);
}
```

PHP

SDK para PHP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();
```

```
        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

def lambda_handler(event, context):
    if event:
        batch_item_failures = []
        sqs_batch_response = {}

        for record in event["Records"]:
```

```
            try:
                print(f"Processed message: {record['body']}")
            except Exception as e:
                batch_item_failures.append({"itemIdentifier":
 record['messageId']})

        sqs_batch_response["batchItemFailures"] = batch_item_failures
        return sqs_batch_response
```

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) ->
 Result<SqsBatchResponse, Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
        }
    }

    Ok(SqsBatchResponse {
        batch_item_failures,
    })
}
```

```
#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(function_handler)).await
}
```

Exemplos de código para Step Functions usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Step Functions com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Step Functions: mais informações sobre o Step Functions.
- Referência da API Step Functions: detalhes sobre todas as ações disponíveis do Step Functions.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de Step Functions usando AWS SDKs
  - Olá, Step Functions
  - Aprenda as noções básicas de Step Functions com um AWS SDK
  - Ações para Step Functions usando AWS SDKs
    - Use CreateActivitycom um AWS SDK

    - Use CreateStateMachinecom um AWS SDK
    - Use DeleteActivitycom um AWS SDK
    - Use DeleteStateMachinecom um AWS SDK
    - Use DescribeExecutioncom um AWS SDK
    - Use DescribeStateMachinecom um AWS SDK
    - Use GetActivityTaskcom um AWS SDK
    - Use ListActivitiescom um AWS SDK
    - Use ListExecutionscom um AWS SDK
    - Use ListStateMachinescom um AWS SDK
    - Use SendTaskSuccesscom um AWS SDK
    - Use StartExecutioncom um AWS SDK
- Cenários para Step Functions usando AWS SDKs
  - Criar uma aplicação de mensageiro com o Step Functions
  - Conceitos básicos da orquestração de fluxo de trabalaho
  - Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions
  - Usar Step Functions para invocar funções do Lambda

Exemplos básicos de Step Functions usando AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Step Functions com AWS SDKs.

Exemplos

- Olá, Step Functions
- Aprenda as noções básicas de Step Functions com um AWS SDK
- Ações para Step Functions usando AWS SDKs
  - Use CreateActivitycom um AWS SDK
  - Use CreateStateMachinecom um AWS SDK
  - Use DeleteActivitycom um AWS SDK
  - Use DeleteStateMachinecom um AWS SDK
  - Use DescribeExecutioncom um AWS SDK
  - Use DescribeStateMachinecom um AWS SDK

- Use GetActivityTaskcom um AWS SDK
- Use ListActivitiescom um AWS SDK
- Use ListExecutionscom um AWS SDK
- Use ListStateMachinescom um AWS SDK
- Use SendTaskSuccesscom um AWS SDK
- Use StartExecutioncom um AWS SDK

Olá, Step Functions

Os exemplos de código a seguir mostram como começar a usar o Step Functions.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace StepFunctionsActions;

using Amazon.StepFunctions;
using Amazon.StepFunctions.Model;

public class HelloStepFunctions
{
    static async Task Main()
    {
        var stepFunctionsClient = new AmazonStepFunctionsClient();

        Console.Clear();
        Console.WriteLine("Welcome to AWS Step Functions");
        Console.WriteLine("Let's list up to 10 of your state machines:");
        var stateMachineListRequest = new ListStateMachinesRequest { MaxResults =
 10 };
```

```
        // Get information for up to 10 Step Functions state machines.
        var response = await
 stepFunctionsClient.ListStateMachinesAsync(stateMachineListRequest);

        if (response.StateMachines.Count > 0)
        {
            response.StateMachines.ForEach(stateMachine =>
            {
                Console.WriteLine($"State Machine
 Name: {stateMachine.Name}\tAmazon Resource Name (ARN):
 {stateMachine.StateMachineArn}");
            });
        }
        else
        {
            Console.WriteLine("\tNo state machines were found.");
        }
    }
}
```

- Para obter detalhes da API, consulte ListStateMachinesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Versão Java do Olá.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListStateMachinesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListStateMachinesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_stepfunctions(stepfunctions_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Step Functions client and
 list
    the state machines in your account. This list might be empty if you haven't
 created
    any state machines.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param stepfunctions_client: A Boto3 Step Functions Client object.
    """
    print("Hello, Step Functions! Let's list up to 10 of your state machines:")
    state_machines = stepfunctions_client.list_state_machines(maxResults=10)
    for sm in state_machines["stateMachines"]:
        print(f"\t{sm['name']}: {sm['stateMachineArn']}")


if __name__ == "__main__":
    hello_stepfunctions(boto3.client("stepfunctions"))
```

- Para obter detalhes da API, consulte a ListStateMachinesReferência da API AWS SDK for Python (Boto3).

Aprenda as noções básicas de Step Functions com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar uma atividade.
- Criar uma máquina de estado a partir de uma definição da Amazon States Language que contenha a atividade criada anteriormente como uma etapa.
- Executar a máquina de estado e responder à atividade com entrada do usuário.
- Obtenha o status e a saída finais após a conclusão da execução e, em seguida, limpe os recursos.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
global using System.Text.Json;
global using Amazon.StepFunctions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using StepFunctionsActions;
global using LogLevel = Microsoft.Extensions.Logging.LogLevel;



using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using Amazon.StepFunctions.Model;

namespace StepFunctionsBasics;
```

```
public class StepFunctionsBasics
{
    private static ILogger _logger = null!;
    private static IConfigurationRoot _configuration = null!;
    private static IAmazonIdentityManagementService _iamService = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Step Functions.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonStepFunctions>()
                    .AddAWSService<IAmazonIdentityManagementService>()
                    .AddTransient<StepFunctionsWrapper>()
            )
            .Build();

        _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<StepFunctionsBasics>();

        // Load configuration settings.
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var activityName = _configuration["ActivityName"];
        var stateMachineName = _configuration["StateMachineName"];

        var roleName = _configuration["RoleName"];
        var repoBaseDir = _configuration["RepoBaseDir"];
        var jsonFilePath = _configuration["JsonFilePath"];
        var jsonFileName = _configuration["JsonFileName"];

        var uiMethods = new UiMethods();
```

```
        var stepFunctionsWrapper =
 host.Services.GetRequiredService<StepFunctionsWrapper>();

        _iamService =
 host.Services.GetRequiredService<IAmazonIdentityManagementService>();

        // Load definition for the state machine from a JSON file.
        var stateDefinitionJson = File.ReadAllText($"{repoBaseDir}{jsonFilePath}
{jsonFileName}");

        Console.Clear();
        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("Create activity");
        Console.WriteLine("Let's start by creating an activity.");
        string activityArn;
        string stateMachineArn;

        // Check to see if the activity already exists.
        var activityList = await stepFunctionsWrapper.ListActivitiesAsync();
        var existingActivity = activityList.FirstOrDefault(activity =>
 activity.Name == activityName);
        if (existingActivity is not null)
        {
            activityArn = existingActivity.ActivityArn;
            Console.WriteLine($"Activity, {activityName}, already exists.");
        }
        else
        {
            activityArn = await
 stepFunctionsWrapper.CreateActivity(activityName);
        }

        // Swap the placeholder in the JSON file with the Amazon Resource Name
 (ARN)
        // of the recently created activity.
        var stateDefinition =
 stateDefinitionJson.Replace("{{DOC_EXAMPLE_ACTIVITY_ARN}}", activityArn);

        uiMethods.DisplayTitle("Create state machine");
        Console.WriteLine("Now we'll create a state machine.");

        // Find or create an IAM role that can be assumed by Step Functions.
```

```
        var role = await GetOrCreateStateMachineRole(roleName);

        // See if the state machine already exists.
        var stateMachineList = await
stepFunctionsWrapper.ListStateMachinesAsync();
        var existingStateMachine =
            stateMachineList.FirstOrDefault(stateMachine => stateMachine.Name ==
stateMachineName);
        if (existingStateMachine is not null)
        {
            Console.WriteLine($"State machine, {stateMachineName}, already
exists.");
            stateMachineArn = existingStateMachine.StateMachineArn;
        }
        else
        {
            // Create the state machine.
            stateMachineArn =
                await stepFunctionsWrapper.CreateStateMachine(stateMachineName,
stateDefinition, role.Arn);
            uiMethods.PressEnter();
        }

        Console.WriteLine("The state machine has been created.");
        var describeStateMachineResponse = await
stepFunctionsWrapper.DescribeStateMachineAsync(stateMachineArn);


Console.WriteLine($"{describeStateMachineResponse.Name}\t{describeStateMachineResponse.S
        Console.WriteLine($"Current status:
{describeStateMachineResponse.Status}");
        Console.WriteLine($"Amazon Resource Name (ARN) of the role assumed by the
state machine: {describeStateMachineResponse.RoleArn}");

        var userName = string.Empty;
        Console.Write("Before we start the state machine, tell me what should
ChatSFN call you? ");
        userName = Console.ReadLine();

        // Keep asking until the user enters a string value.
        while (string.IsNullOrEmpty(userName))
        {
            Console.Write("Enter your name: ");
            userName = Console.ReadLine();
```

```
        }

        var executionJson = @"{""name"": """ + userName + @"""}";

        // Start the state machine execution.
        Console.WriteLine("Now we'll start execution of the state machine.");
        var executionArn = await
stepFunctionsWrapper.StartExecutionAsync(executionJson, stateMachineArn);
        Console.WriteLine("State machine started.");

        Console.WriteLine($"Thank you, {userName}. Now let's get started...");
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("ChatSFN");

        var isDone = false;
        var response = new GetActivityTaskResponse();
        var taskToken = string.Empty;
        var userChoice = string.Empty;

        while (!isDone)
        {
            response = await
stepFunctionsWrapper.GetActivityTaskAsync(activityArn, "MvpWorker");
            taskToken = response.TaskToken;

            // Parse the returned JSON string.
            var taskJsonResponse = JsonDocument.Parse(response.Input);
            var taskJsonObject = taskJsonResponse.RootElement;
            var message = taskJsonObject.GetProperty("message").GetString();
            var actions =
taskJsonObject.GetProperty("actions").EnumerateArray().Select(x =>
x.ToString()).ToList();
            Console.WriteLine($"\n{message}\n");

            // Prompt the user for another choice.
            Console.WriteLine("ChatSFN: What would you like me to do?");
            actions.ForEach(action => Console.WriteLine($"\t{action}"));
            Console.Write($"\n{userName}, tell me your choice: ");
            userChoice = Console.ReadLine();
            if (userChoice?.ToLower() == "done")
            {
                isDone = true;
            }
```

```
            Console.WriteLine($"You have selected: {userChoice}");
            var jsonResponse = @"{""action"": """ + userChoice + @"""}";

            await stepFunctionsWrapper.SendTaskSuccessAsync(taskToken,
jsonResponse);
        }

        await stepFunctionsWrapper.StopExecution(executionArn);
        Console.WriteLine("Now we will wait for the execution to stop.");
        DescribeExecutionResponse executionResponse;
        do
        {
            executionResponse = await
stepFunctionsWrapper.DescribeExecutionAsync(executionArn);
        } while (executionResponse.Status == ExecutionStatus.RUNNING);

        Console.WriteLine("State machine stopped.");
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("State machine executions");
        Console.WriteLine("Now let's take a look at the execution values for the
state machine.");

        // List the executions.
        var executions = await
stepFunctionsWrapper.ListExecutionsAsync(stateMachineArn);

        uiMethods.DisplayTitle("Step function execution values");
        executions.ForEach(execution =>
        {
            Console.WriteLine($"{execution.Name}\t{execution.StartDate} to
{execution.StopDate}");
        });

        uiMethods.PressEnter();

        // Now delete the state machine and the activity.
        uiMethods.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the state machine...");

        await stepFunctionsWrapper.DeleteStateMachine(stateMachineArn);
        Console.WriteLine("State machine deleted.");
```

```
        Console.WriteLine("Deleting the activity...");
        await stepFunctionsWrapper.DeleteActivity(activityArn);
        Console.WriteLine("Activity deleted.");

        Console.WriteLine("The Amazon Step Functions scenario is now complete.");
    }

    static async Task<Role> GetOrCreateStateMachineRole(string roleName)
    {
        // Define the policy document for the role.
        var stateMachineRolePolicy = @"{
         ""Version"": ""2012-10-17"",
        ""Statement"": [{
            ""Sid"": """",
            ""Effect"": ""Allow"",
            ""Principal"": {
                ""Service"": ""states.amazonaws.com""},
            ""Action"": ""sts:AssumeRole""}]}";

        var role = new Role();
        var roleExists = false;

        try
        {
            var getRoleResponse = await _iamService.GetRoleAsync(new
GetRoleRequest { RoleName = roleName });
            roleExists = true;
            role = getRoleResponse.Role;
        }
        catch (NoSuchEntityException)
        {
            // The role doesn't exist. Create it.
            Console.WriteLine($"Role, {roleName} doesn't exist. Creating it...");
        }

        if (!roleExists)
        {
            var request = new CreateRoleRequest
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = stateMachineRolePolicy,
            };

            var createRoleResponse = await _iamService.CreateRoleAsync(request);
```

```
            role = createRoleResponse.Role;
        }

        return role;
    }
}


namespace StepFunctionsBasics;

/// <summary>
/// Some useful methods to make screen display easier.
/// </summary>
public class UiMethods
{
    private readonly string _sepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Welcome to the AWS Step Functions Demo");

        Console.WriteLine("This example application will do the following:");
        Console.WriteLine("\t 1. Create an activity.");
        Console.WriteLine("\t 2. Create a state machine.");
        Console.WriteLine("\t 3. Start an execution.");
        Console.WriteLine("\t 4. Run the worker, then stop it.");
        Console.WriteLine("\t 5. List executions.");
        Console.WriteLine("\t 6. Clean up the resources created for the
 example.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue.");
        _ = Console.ReadLine();
    }
```

```
    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter"></param>
    /// <returns></returns>
    private string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(_sepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(_sepBar);
    }
}
```

Defina uma classe que envolva ações de máquina de estado e atividade.

```
namespace StepFunctionsActions;

using Amazon.StepFunctions;
using Amazon.StepFunctions.Model;

/// <summary>
/// Wrapper that performs AWS Step Functions actions.
/// </summary>
public class StepFunctionsWrapper
{
    private readonly IAmazonStepFunctions _amazonStepFunctions;

    /// <summary>
```

```
    /// The constructor for the StepFunctionsWrapper. Initializes the
    /// client object passed to it.
    /// </summary>
    /// <param name="amazonStepFunctions">An initialized Step Functions client
object.</param>
    public StepFunctionsWrapper(IAmazonStepFunctions amazonStepFunctions)
    {
        _amazonStepFunctions = amazonStepFunctions;
    }

    /// <summary>
    /// Create a Step Functions activity using the supplied name.
    /// </summary>
    /// <param name="activityName">The name for the new Step Functions
activity.</param>
    /// <returns>The Amazon Resource Name (ARN) for the new activity.</returns>
    public async Task<string> CreateActivity(string activityName)
    {
        var response = await _amazonStepFunctions.CreateActivityAsync(new
CreateActivityRequest { Name = activityName });
        return response.ActivityArn;
    }


    /// <summary>
    /// Create a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineName">Name for the new Step Functions state
    /// machine.</param>
    /// <param name="definition">A JSON string that defines the Step Functions
    /// state machine.</param>
    /// <param name="roleArn">The Amazon Resource Name (ARN) of the role.</param>
    /// <returns></returns>
    public async Task<string> CreateStateMachine(string stateMachineName, string
definition, string roleArn)
    {
        var request = new CreateStateMachineRequest
        {
            Name = stateMachineName,
            Definition = definition,
            RoleArn = roleArn
        };

        var response =
```

```
            await _amazonStepFunctions.CreateStateMachineAsync(request);
        return response.StateMachineArn;
    }


    /// <summary>
    /// Delete a Step Machine activity.
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the activity.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteActivity(string activityArn)
    {
        var response = await _amazonStepFunctions.DeleteActivityAsync(new
 DeleteActivityRequest { ActivityArn = activityArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// state machine.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteStateMachine(string stateMachineArn)
    {
        var response = await _amazonStepFunctions.DeleteStateMachineAsync(new
 DeleteStateMachineRequest
        { StateMachineArn = stateMachineArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Retrieve information about the specified Step Functions execution.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of the
    /// Step Functions execution.</param>
    /// <returns>The API response returned by the API.</returns>
    public async Task<DescribeExecutionResponse> DescribeExecutionAsync(string
 executionArn)
    {
```

```
        var response = await _amazonStepFunctions.DescribeExecutionAsync(new
 DescribeExecutionRequest { ExecutionArn = executionArn });
        return response;
    }


    /// <summary>
    /// Retrieve information about the specified Step Functions state machine.
    /// </summary>
    /// <param name="StateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine to retrieve.</param>
    /// <returns>Information about the specified Step Functions state machine.</
returns>
    public async Task<DescribeStateMachineResponse>
 DescribeStateMachineAsync(string StateMachineArn)
    {
        var response = await _amazonStepFunctions.DescribeStateMachineAsync(new
 DescribeStateMachineRequest { StateMachineArn = StateMachineArn });
        return response;
    }


    /// <summary>
    /// Retrieve a task with the specified Step Functions activity
    /// with the specified Amazon Resource Name (ARN).
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the Step Functions activity.</param>
    /// <param name="workerName">The name of the Step Functions worker.</param>
    /// <returns>The response from the Step Functions activity.</returns>
    public async Task<GetActivityTaskResponse> GetActivityTaskAsync(string
 activityArn, string workerName)
    {
        var response = await _amazonStepFunctions.GetActivityTaskAsync(new
 GetActivityTaskRequest
        { ActivityArn = activityArn, WorkerName = workerName });
        return response;
    }


    /// <summary>
    /// List the Step Functions activities for the current account.
    /// </summary>
    /// <returns>A list of ActivityListItems.</returns>
```

```
    public async Task<List<ActivityListItem>> ListActivitiesAsync()
    {
        var request = new ListActivitiesRequest();
        var activities = new List<ActivityListItem>();

        do
        {
            var response = await
_amazonStepFunctions.ListActivitiesAsync(request);

            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }

            activities.AddRange(response.Activities);
        }
        while (request.NextToken is not null);

        return activities;
    }


    /// <summary>
    /// Retrieve information about executions of a Step Functions
    /// state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>A list of ExecutionListItem objects.</returns>
    public async Task<List<ExecutionListItem>> ListExecutionsAsync(string
stateMachineArn)
    {
        var executions = new List<ExecutionListItem>();
        ListExecutionsResponse response;
        var request = new ListExecutionsRequest { StateMachineArn =
stateMachineArn };

        do
        {
            response = await _amazonStepFunctions.ListExecutionsAsync(request);
            executions.AddRange(response.Executions);
            if (response.NextToken is not null)
            {
```

```
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);

        return executions;
    }


    /// <summary>
    /// Retrieve a list of Step Functions state machines.
    /// </summary>
    /// <returns>A list of StateMachineListItem objects.</returns>
    public async Task<List<StateMachineListItem>> ListStateMachinesAsync()
    {
        var stateMachines = new List<StateMachineListItem>();
        var listStateMachinesPaginator =
            _amazonStepFunctions.Paginators.ListStateMachines(new
ListStateMachinesRequest());

        await foreach (var response in listStateMachinesPaginator.Responses)
        {
            stateMachines.AddRange(response.StateMachines);
        }

        return stateMachines;
    }


    /// <summary>
    /// Indicate that the Step Functions task, indicated by the
    /// task token, has completed successfully.
    /// </summary>
    /// <param name="taskToken">Identifies the task.</param>
    /// <param name="taskResponse">The response received from executing the
task.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SendTaskSuccessAsync(string taskToken, string
taskResponse)
    {
        var response = await _amazonStepFunctions.SendTaskSuccessAsync(new
SendTaskSuccessRequest
        { TaskToken = taskToken, Output = taskResponse });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }


    /// <summary>
    /// Start execution of an AWS Step Functions state machine.
    /// </summary>
    /// <param name="executionName">The name to use for the execution.</param>
    /// <param name="executionJson">The JSON string to pass for execution.</
param>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>The Amazon Resource Name (ARN) of the AWS Step Functions
    /// execution.</returns>
    public async Task<string> StartExecutionAsync(string executionJson, string
 stateMachineArn)
    {
        var executionRequest = new StartExecutionRequest
        {
            Input = executionJson,
            StateMachineArn = stateMachineArn
        };

        var response = await
 _amazonStepFunctions.StartExecutionAsync(executionRequest);
        return response.ExecutionArn;
    }


    /// <summary>
    /// Stop execution of a Step Functions workflow.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of
    /// the Step Functions execution to stop.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StopExecution(string executionArn)
    {
        var response =
            await _amazonStepFunctions.StopExecutionAsync(new
 StopExecutionRequest { ExecutionArn = executionArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 * <p>
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/
sample_files
 * <p>
```

```
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 * <p>
 * Also, set up your development environment, including your credentials.
 * <p>
 * For information, see this documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
 * This Java code example performs the following tasks:
 * <p>
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <roleARN> <activityName> <stateMachineName>

            Where:
                roleName - The name of the IAM role to create for this state
 machine.
                activityName - The name of an activity to create.
                stateMachineName - The name of the state machine to create.
                jsonFile - The location of the chat_sfn_state_machine.json file.
 You can located it in resources/sample_files.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String jsonFile = args[3];
        String polJSON = """
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "states.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
            """;


        Scanner sc = new Scanner(System.in);
        boolean action = false;

        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(regionGl)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Step Functions example
 scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an activity.");
        String activityArn = createActivity(sfnClient, activityName);
        System.out.println("The ARN of the activity is " + activityArn);
        System.out.println(DASHES);
```

```
        // Read the file using FileInputStream
        FileInputStream inputStream = new FileInputStream(jsonFile);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(inputStream, JsonNode.class);
        String jsonString = mapper.writeValueAsString(jsonNode);

        // Modify the Resource node.
        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode root = objectMapper.readTree(jsonString);
        ((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

        // Convert the modified Java object back to a JSON string.
        String stateDefinition = objectMapper.writeValueAsString(root);
        System.out.println(stateDefinition);

        System.out.println(DASHES);
        System.out.println("2. Create a state machine.");
        String roleARN = createIAMRole(iam, roleName, polJSON);
        String stateMachineArn = createMachine(sfnClient, roleARN,
stateMachineName, stateDefinition);
        System.out.println("The ARN of the state machine is " + stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe the state machine.");
        describeStateMachine(sfnClient, stateMachineArn);
        System.out.println("What should ChatSFN call you?");
        String userName = sc.nextLine();
        System.out.println("Hello " + userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        // The JSON to pass to the StartExecution call.
        String executionJson = "{ \"name\" : \"" + userName + "\" }";
        System.out.println(executionJson);
        System.out.println("4. Start execution of the state machine and interact
with it.");
        String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
        System.out.println("The ARN of the state machine execution is " +
runArn);
        List<String> myList;
        while (!action) {
```

```
            myList = getActivityTask(sfnClient, activityArn);
            System.out.println("ChatSFN: " + myList.get(1));
            System.out.println(userName + " please specify a value.");
            String myAction = sc.nextLine();
            if (myAction.compareTo("done") == 0)
                action = true;

            System.out.println("You have selected " + myAction);
            String taskJson = "{ \"action\" : \"" + myAction + "\" }";
            System.out.println(taskJson);
            sendTaskSuccess(sfnClient, myList.get(0), taskJson);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the execution.");
        describeExe(sfnClient, runArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Delete the activity.");
        deleteActivity(sfnClient, activityArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the state machines.");
        deleteMachine(sfnClient, stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Step Functions example scenario is
complete.");
        System.out.println(DASHES);
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(polJSON)
                .description("Created using the AWS SDK for Java")
                .build();
```

```
            CreateRoleResponse response = iam.createRole(request);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has
succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   public static void sendTaskSuccess(SfnClient sfnClient, String token, String
 json) {
       try {
           SendTaskSuccessRequest successRequest =
 SendTaskSuccessRequest.builder()
               .taskToken(token)
               .output(json)
               .build();

           sfnClient.sendTaskSuccess(successRequest);

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static List<String> getActivityTask(SfnClient sfnClient, String
 actArn) {
       List<String> myList = new ArrayList<>();
       GetActivityTaskRequest getActivityTaskRequest =
 GetActivityTaskRequest.builder()
           .activityArn(actArn)
           .build();

       GetActivityTaskResponse response =
 sfnClient.getActivityTask(getActivityTaskRequest);
       myList.add(response.taskToken());
       myList.add(response.input());
       return myList;
   }

   public static void deleteActivity(SfnClient sfnClient, String actArn) {
       try {
           DeleteActivityRequest activityRequest =
 DeleteActivityRequest.builder()
               .activityArn(actArn)
               .build();

           sfnClient.deleteActivity(activityRequest);
           System.out.println("You have deleted " + actArn);

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }

    public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
response.name());
            System.out.println("The status of the State machine is " +
response.status());
            System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }

    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn)
{
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
```

```
                System.out.println("The state machine is not deleted yet. The
status is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }

    public static String startWorkflow(SfnClient sfnClient, String
stateMachineArn, String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest =
StartExecutionRequest.builder()
                .input(jsonEx)
                .stateMachineArn(stateMachineArn)
                .name(uuidValue)
                .build();

            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createMachine(SfnClient sfnClient, String roleARN,
String stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
                .definition(json)
                .name(stateMachineName)
                .roleArn(roleARN)
                .type(StateMachineType.STANDARD)
                .build();
```

```
            CreateStateMachineResponse response =
 sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createActivity(SfnClient sfnClient, String activityName)
 {
        try {
            CreateActivityRequest activityRequest =
 CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
 sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine

- GetActivityTask
- ListActivities
- ListStateMachines
- SendTaskSuccess
- StartExecution
- StopExecution

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
```

```
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.

 You can obtain the JSON file to create a state machine in the following GitHub
 location:

 https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files

 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin code example performs the following tasks:

 1. List activities using a paginator.
 2. List state machines using a paginator.
 3. Creates an activity.
 4. Creates a state machine.
 5. Describes the state machine.
 6. Starts execution of the state machine and interacts with it.
 7. Describes the execution.
 8. Deletes the activity.
 9. Deletes the state machine.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleARN> <activityName> <stateMachineName>

    Where:
        roleName - The name of the IAM role to create for this state machine.
        activityName - The name of an activity to create.
        stateMachineName - The name of the state machine to create.
```

```
        jsonFile - The location of the chat_sfn_state_machine.json file. You can
located it in resources/sample_files.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
    val stateMachineName = args[2]
    val jsonFile = args[3]
    val sc = Scanner(System.`in`)
    var action = false

    val polJSON = """{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }"""

    println(DASHES)
    println("Welcome to the AWS Step Functions example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List activities using a Paginator.")
    listActivitesPagnator()
    println(DASHES)

    println(DASHES)
    println("2. List state machines using a paginator.")
    listStatemachinesPagnator()
    println(DASHES)
```

```
    println(DASHES)
    println("3. Create a new activity.")
    val activityArn = createActivity(activityName)
    println("The ARN of the Activity is $activityArn")
    println(DASHES)

    // Get JSON to use for the state machine and place the activityArn value into
it.
    val stream = GetStream()
    val jsonString = stream.getStream(jsonFile)

    // Modify the Resource node.
    val objectMapper = ObjectMapper()
    val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
activityArn)

    // Convert the modified Java object back to a JSON string.
    val stateDefinition = objectMapper.writeValueAsString(root)
    println(stateDefinition)

    println(DASHES)
    println("4. Create a state machine.")
    val roleARN = createIAMRole(roleName, polJSON)
    val stateMachineArn = createMachine(roleARN, stateMachineName,
stateDefinition)
    println("The ARN of the state machine is $stateMachineArn")
    println(DASHES)

    println(DASHES)
    println("5. Describe the state machine.")
    describeStateMachine(stateMachineArn)
    println("What should ChatSFN call you?")
    val userName = sc.nextLine()
    println("Hello $userName")
    println(DASHES)

    println(DASHES)
    // The JSON to pass to the StartExecution call.
    val executionJson = "{ \"name\" : \"$userName\" }"
    println(executionJson)
    println("6. Start execution of the state machine and interact with it.")
    val runArn = startWorkflow(stateMachineArn, executionJson)
    println("The ARN of the state machine execution is $runArn")
```

```
    var myList: List<String>
    while (!action) {
        myList = getActivityTask(activityArn)
        println("ChatSFN: " + myList[1])
        println("$userName please specify a value.")
        val myAction = sc.nextLine()
        if (myAction.compareTo("done") == 0) {
            action = true
        }
        println("You have selected $myAction")
        val taskJson = "{ \"action\" : \"$myAction\" }"
        println(taskJson)
        sendTaskSuccess(myList[0], taskJson)
    }
    println(DASHES)

    println(DASHES)
    println("7. Describe the execution.")
    describeExe(runArn)
    println(DASHES)

    println(DASHES)
    println("8. Delete the activity.")
    deleteActivity(activityArn)
    println(DASHES)

    println(DASHES)
    println("9. Delete the state machines.")
    deleteMachine(stateMachineArn)
    println(DASHES)

    println(DASHES)
    println("The AWS Step Functions example scenario is complete.")
    println(DASHES)
}

suspend fun listStatemachinesPagnator() {
    val machineRequest =
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient
```

```
            .listStateMachinesPaginated(machineRequest)
            .transform { it.stateMachines?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The state machine ARN is ${obj.stateMachineArn}")
            }
    }
}

suspend fun listActivitesPagnator() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listActivitiesPaginated(activitiesRequest)
            .transform { it.activities?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The activity ARN is ${obj.activityArn}")
            }
    }
}

suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}

suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
```

```
        println("You have deleted $actArn")
    }
}

suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}

suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is
 ${response.stateMachineArn}")
```

```
        println("The role ARN value is ${response.roleArn}")
    }
}

suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}

suspend fun createIAMRole(
    roleNameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = roleNameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
```

```
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class StateMachineScenario:
    """Runs an interactive scenario that shows how to get started using Step
 Functions."""

    def __init__(self, activity, state_machine, iam_client):
        """
        :param activity: An object that wraps activity actions.
        :param state_machine: An object that wraps state machine actions.
        :param iam_client: A Boto3 AWS Identity and Access Management (IAM)
 client.
        """
        self.activity = activity
        self.state_machine = state_machine
        self.iam_client = iam_client
        self.state_machine_role = None

    def prerequisites(self, state_machine_role_name):
        """
        Finds or creates an IAM role that can be assumed by Step Functions.
        A role of this kind is required to create a state machine.
        The state machine used in this example does not call any additional
 services,
        so it needs no additional permissions.

        :param state_machine_role_name: The name of the role.
        :return: Data about the role.
        """
        trust_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "",
                    "Effect": "Allow",
                    "Principal": {"Service": "states.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        try:
            role = self.iam_client.get_role(RoleName=state_machine_role_name)
            print(f"Prerequisite IAM role {state_machine_role_name} already
 exists.")
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "NoSuchEntity":
                role = None
            else:
                logger.error(
                    "Couldn't get prerequisite IAM role %s. Here's why: %s: %s",
                    state_machine_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        if role is None:
            try:
                role = self.iam_client.create_role(
                    RoleName=state_machine_role_name,
                    AssumeRolePolicyDocument=json.dumps(trust_policy),
                )
            except ClientError as err:
                logger.error(
                    "Couldn't create prerequisite IAM role %s. Here's why: %s:
%s",
                    state_machine_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        self.state_machine_role = role["Role"]

    def find_or_create_activity(self, activity_name):
        """
        Finds or creates a Step Functions activity.

        :param activity_name: The name of the activity.
        :return: The Amazon Resource Name (ARN) of the activity.
        """
        print("First, let's set up an activity and state machine.")
        activity_arn = self.activity.find(activity_name)
        if activity_arn is None:
            activity_arn = self.activity.create(activity_name)
            print(
                f"Activity {activity_name} created. Its Amazon Resource Name
(ARN) is "
                f"{activity_arn}."
            )
        else:
```

```
            print(f"Activity {activity_name} already exists.")
        return activity_arn

    def find_or_create_state_machine(
        self, state_machine_name, activity_arn, state_machine_file
    ):
        """
        Finds or creates a Step Functions state machine.

        :param state_machine_name: The name of the state machine.
        :param activity_arn: The ARN of an activity that is used as a step in the
state
                                machine. This ARN is injected into the state machine
                                definition that's used to create the state machine.
        :param state_machine_file: The path to a file containing the state
machine
                                      definition.
        :return: The ARN of the state machine.
        """
        state_machine_arn = self.state_machine.find(state_machine_name)
        if state_machine_arn is None:
            with open(state_machine_file) as state_machine_file:
                state_machine_def = state_machine_file.read().replace(
                    "{{DOC_EXAMPLE_ACTIVITY_ARN}}", activity_arn
                )
                state_machine_arn = self.state_machine.create(
                    state_machine_name,
                    state_machine_def,
                    self.state_machine_role["Arn"],
                )
            print(f"State machine {state_machine_name} created.")
        else:
            print(f"State machine {state_machine_name} already exists.")
        print("-" * 88)
        print(f"Here's some information about state machine
{state_machine_name}:")
        state_machine_info = self.state_machine.describe(state_machine_arn)
        for field in ["name", "status", "stateMachineArn", "roleArn"]:
            print(f"\t{field}: {state_machine_info[field]}")
        return state_machine_arn

    def run_state_machine(self, state_machine_arn, activity_arn):
        """
        Run the state machine. The state machine used in this example is a simple
```

```
        chat simulation. It contains an activity step in a loop that is used for
user
        interaction. When the state machine gets to the activity step, it waits
for
        an external application to get task data and submit a response. This
function
        acts as the activity application by getting task input and responding
with
        user input.

        :param state_machine_arn: The ARN of the state machine.
        :param activity_arn: The ARN of the activity used as a step in the state
machine.
        :return: The ARN of the run.
        """
        print(
            f"Let's run the state machine. It's a simplistic, non-AI chat
simulator "
            f"we'll call ChatSFN."
        )
        user_name = q.ask("What should ChatSFN call you? ", q.non_empty)
        run_input = {"name": user_name}
        print("Starting state machine...")
        run_arn = self.state_machine.start(state_machine_arn,
json.dumps(run_input))
        action = None
        while action != "done":
            activity_task = self.activity.get_task(activity_arn)
            task_input = json.loads(activity_task["input"])
            print(f"ChatSFN: {task_input['message']}")
            action = task_input["actions"][
                q.choose("What now? ", task_input["actions"])
            ]
            task_response = {"action": action}
            self.activity.send_task_success(
                activity_task["taskToken"], json.dumps(task_response)
            )
        return run_arn

    def finish_state_machine_run(self, run_arn):
        """
        Wait for the state machine run to finish, then print final status and
output.
```

```
        :param run_arn: The ARN of the run to retrieve.
        """
        print(f"Let's get the final output from the state machine:")
        status = "RUNNING"
        while status == "RUNNING":
            run_output = self.state_machine.describe_run(run_arn)
            status = run_output["status"]
            if status == "RUNNING":
                print(
                    "The state machine is still running, let's wait for it to
finish."
                )
                wait(1)
            elif status == "SUCCEEDED":
                print(f"ChatSFN: {json.loads(run_output['output'])['message']}")
            else:
                print(f"Run status: {status}.")

    def cleanup(
        self,
        state_machine_name,
        state_machine_arn,
        activity_name,
        activity_arn,
        state_machine_role_name,
    ):
        """
        Clean up resources created by this example.

        :param state_machine_name: The name of the state machine.
        :param state_machine_arn: The ARN of the state machine.
        :param activity_name: The name of the activity.
        :param activity_arn: The ARN of the activity.
        :param state_machine_role_name: The name of the role used by the state
machine.
        """
        if q.ask(
            "Do you want to delete the state machine, activity, and role created
for this "
            "example? (y/n) ",
            q.is_yesno,
        ):
            self.state_machine.delete(state_machine_arn)
            print(f"Deleted state machine {state_machine_name}.")
```

```
            self.activity.delete(activity_arn)
            print(f"Deleted activity {activity_name}.")
            self.iam_client.delete_role(RoleName=state_machine_role_name)
            print(f"Deleted role {state_machine_role_name}.")

    def run_scenario(self, activity_name, state_machine_name):
        print("-" * 88)
        print("Welcome to the AWS Step Functions state machines demo.")
        print("-" * 88)

        activity_arn = self.find_or_create_activity(activity_name)
        state_machine_arn = self.find_or_create_state_machine(
            state_machine_name,
            activity_arn,
            "../../../resources/sample_files/chat_sfn_state_machine.json",
        )
        print("-" * 88)
        run_arn = self.run_state_machine(state_machine_arn, activity_arn)
        print("-" * 88)
        self.finish_state_machine_run(run_arn)
        print("-" * 88)
        self.cleanup(
            state_machine_name,
            state_machine_arn,
            activity_name,
            activity_arn,
            self.state_machine_role["RoleName"],
        )

        print("-" * 88)
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        stepfunctions_client = boto3.client("stepfunctions")
        iam_client = boto3.client("iam")
        scenario = StateMachineScenario(
            Activity(stepfunctions_client),
            StateMachine(stepfunctions_client),
            iam_client,
        )
```

```
        scenario.prerequisites("doc-example-state-machine-chat")
        scenario.run_scenario("doc-example-activity", "doc-example-state-
machine")
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que envolva ações de máquina de estado.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name, definition, role_arn):
        """
        Creates a state machine with the specific definition. The state machine
 assumes
        the provided role before it starts a run.

        :param name: The name to give the state machine.
        :param definition: The Amazon States Language definition of the steps in
 the
                           the state machine.
        :param role_arn: The Amazon Resource Name (ARN) of the role that is
 assumed by
                         Step Functions when the state machine is run.
        :return: The ARN of the newly created state machine.
        """
        try:
            response = self.stepfunctions_client.create_state_machine(
                name=name, definition=definition, roleArn=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't create state machine %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["stateMachineArn"]


    def find(self, name):
        """
        Find a state machine by name. This requires listing the state machines
until
        one is found with a matching name.

        :param name: The name of the state machine to search for.
        :return: The ARN of the state machine if found; otherwise, None.
        """
        try:
            paginator =
self.stepfunctions_client.get_paginator("list_state_machines")
            for page in paginator.paginate():
                for state_machine in page.get("stateMachines", []):
                    if state_machine["name"] == name:
                        return state_machine["stateMachineArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list state machines. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self, state_machine_arn):
        """
        Get data about a state machine.

        :param state_machine_arn: The ARN of the state machine to look up.
        :return: The retrieved state machine data.
        """
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't describe state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def start(self, state_machine_arn, run_input):
        """
        Start a run of a state machine with a specified input. A run is also
known
        as an "execution" in Step Functions.

        :param state_machine_arn: The ARN of the state machine to run.
        :param run_input: The input to the state machine, in JSON format.
        :return: The ARN of the run. This can be used to get information about
the run,
                 including its current status and final output.
        """
        try:
            response = self.stepfunctions_client.start_execution(
                stateMachineArn=state_machine_arn, input=run_input
            )
        except ClientError as err:
            logger.error(
                "Couldn't start state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["executionArn"]


    def describe_run(self, run_arn):
        """
        Get data about a state machine run, such as its current status or final
output.
```

```
        :param run_arn: The ARN of the run to look up.
        :return: The retrieved run data.
        """
        try:
            response = self.stepfunctions_client.describe_execution(
                executionArn=run_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe run %s. Here's why: %s: %s",
                run_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete(self, state_machine_arn):
        """
        Delete a state machine and all of its run data.

        :param state_machine_arn: The ARN of the state machine to delete.
        """
        try:
            response = self.stepfunctions_client.delete_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

Defina uma classe que envolva ações de atividade.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name):
        """
        Create an activity.

        :param name: The name of the activity to create.
        :return: The Amazon Resource Name (ARN) of the newly created activity.
        """
        try:
            response = self.stepfunctions_client.create_activity(name=name)
        except ClientError as err:
            logger.error(
                "Couldn't create activity %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["activityArn"]


    def find(self, name):
        """
        Find an activity by name. This requires listing activities until one is
 found
        with a matching name.

        :param name: The name of the activity to search for.
        :return: If found, the ARN of the activity; otherwise, None.
        """
        try:
```

```
            paginator =
self.stepfunctions_client.get_paginator("list_activities")
            for page in paginator.paginate():
                for activity in page.get("activities", []):
                    if activity["name"] == name:
                        return activity["activityArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list activities. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_task(self, activity_arn):
        """
        Gets task data for an activity. When a state machine is waiting for the
        specified activity, a response is returned with data from the state
machine.
        When a state machine is not waiting, this call blocks for 60 seconds.

        :param activity_arn: The ARN of the activity to get task data for.
        :return: The task data for the activity.
        """
        try:
            response = self.stepfunctions_client.get_activity_task(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't get a task for activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def send_task_success(self, task_token, task_response):
        """
```

```
        Sends a success response to a waiting activity step. A state machine with
an
        activity step waits for the activity to get task data and then respond
with
        either success or failure before it resumes processing.

        :param task_token: The token associated with the task. This is included
in the
                           response to the get_activity_task action and must be
sent
                           without modification.
        :param task_response: The response data from the activity. This data is
                              received and processed by the state machine.
        """
        try:
            self.stepfunctions_client.send_task_success(
                taskToken=task_token, output=task_response
            )
        except ClientError as err:
            logger.error(
                "Couldn't send task success. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self, activity_arn):
        """
        Delete an activity.

        :param activity_arn: The ARN of the activity to delete.
        """
        try:
            response = self.stepfunctions_client.delete_activity(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Ações para Step Functions usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Step Functions com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Step Functions e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para Step Functions usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Step Functions.

Exemplos

- Use CreateActivitycom um AWS SDK
- Use CreateStateMachinecom um AWS SDK
- Use DeleteActivitycom um AWS SDK
- Use DeleteStateMachinecom um AWS SDK
- Use DescribeExecutioncom um AWS SDK
- Use DescribeStateMachinecom um AWS SDK
- Use GetActivityTaskcom um AWS SDK
- Use ListActivitiescom um AWS SDK
- Use ListExecutionscom um AWS SDK
- Use ListStateMachinescom um AWS SDK
- Use SendTaskSuccesscom um AWS SDK
- Use StartExecutioncom um AWS SDK

Use **`CreateActivity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateActivity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create a Step Functions activity using the supplied name.
/// </summary>
```

```
    /// <param name="activityName">The name for the new Step Functions
activity.</param>
    /// <returns>The Amazon Resource Name (ARN) for the new activity.</returns>
    public async Task<string> CreateActivity(string activityName)
    {
        var response = await _amazonStepFunctions.CreateActivityAsync(new
CreateActivityRequest { Name = activityName });
        return response.ActivityArn;
    }
```

- Para obter detalhes da API, consulte CreateActivitya Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String createActivity(SfnClient sfnClient, String activityName)
{
        try {
            CreateActivityRequest activityRequest =
CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }
```

- Para obter detalhes da API, consulte CreateActivitya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Para obter detalhes da API, consulte a CreateActivityreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name):
        """
        Create an activity.

        :param name: The name of the activity to create.
        :return: The Amazon Resource Name (ARN) of the newly created activity.
        """
        try:
            response = self.stepfunctions_client.create_activity(name=name)
        except ClientError as err:
            logger.error(
                "Couldn't create activity %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["activityArn"]
```

- Para obter detalhes da API, consulte a CreateActivityReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_sfn->createactivity(
      iv_name = iv_name
    ).
    ov_activity_arn = lo_result->get_activityarn( ).
    MESSAGE 'Activity created successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnactivityalrdyex.
    MESSAGE 'Activity already exists.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidname.
    MESSAGE 'Invalid activity name.' TYPE 'E'.
  CATCH /aws1/cx_sfnactivitylimitexcd.
    MESSAGE 'Activity limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateActivityreferência da API AWS SDK for SAP ABAP.

Use **`CreateStateMachine`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `CreateStateMachine`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos da orquestração de fluxo de trabalaho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineName">Name for the new Step Functions state
    /// machine.</param>
    /// <param name="definition">A JSON string that defines the Step Functions
    /// state machine.</param>
    /// <param name="roleArn">The Amazon Resource Name (ARN) of the role.</param>
    /// <returns></returns>
    public async Task<string> CreateStateMachine(string stateMachineName, string
 definition, string roleArn)
    {
        var request = new CreateStateMachineRequest
        {
            Name = stateMachineName,
            Definition = definition,
            RoleArn = roleArn
        };

        var response =
            await _amazonStepFunctions.CreateStateMachineAsync(request);
        return response.StateMachineArn;
    }
```

- Para obter detalhes da API, consulte CreateStateMachinea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createMachine(SfnClient sfnClient, String roleARN,
String stateMachineName, String json) {
       try {
           CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
               .definition(json)
               .name(stateMachineName)
               .roleArn(roleARN)
               .type(StateMachineType.STANDARD)
               .build();

           CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
           return response.stateMachineArn();

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte CreateStateMachinea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- Para obter detalhes da API, consulte a CreateStateMachinereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name, definition, role_arn):
        """
        Creates a state machine with the specific definition. The state machine
 assumes
        the provided role before it starts a run.

        :param name: The name to give the state machine.
        :param definition: The Amazon States Language definition of the steps in
 the
                           the state machine.
        :param role_arn: The Amazon Resource Name (ARN) of the role that is
 assumed by
                         Step Functions when the state machine is run.
        :return: The ARN of the newly created state machine.
        """
        try:
            response = self.stepfunctions_client.create_state_machine(
                name=name, definition=definition, roleArn=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't create state machine %s. Here's why: %s: %s",
```

```
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["stateMachineArn"]
```

- Para obter detalhes da API, consulte a CreateStateMachineReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_sfn->createstatemachine(
          iv_name = iv_name
          iv_definition = iv_definition
          iv_rolearn = iv_role_arn
        ).
        ov_state_machine_arn = lo_result->get_statemachinearn( ).
        MESSAGE 'State machine created successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfnstatemachinealrex.
        MESSAGE 'State machine already exists.' TYPE 'E'.
      CATCH /aws1/cx_sfninvaliddefinition.
        MESSAGE 'Invalid state machine definition.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidname.
        MESSAGE 'Invalid state machine name.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid role ARN.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateStateMachinereferência da API AWS SDK for SAP ABAP.

Use **`DeleteActivity`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteActivity`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a Step Machine activity.
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the activity.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteActivity(string activityArn)
    {
        var response = await _amazonStepFunctions.DeleteActivityAsync(new
DeleteActivityRequest { ActivityArn = activityArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteActivitya Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest =
DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteActivitya Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- Para obter detalhes da API, consulte a DeleteActivityreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def delete(self, activity_arn):
        """
        Delete an activity.

        :param activity_arn: The ARN of the activity to delete.
        """
```

```
        try:
            response = self.stepfunctions_client.delete_activity(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteActivityReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sfn->deleteactivity(
          iv_activityarn = iv_activity_arn
        ).
        MESSAGE 'Activity deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid activity ARN.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteActivityreferência da API AWS SDK for SAP ABAP.

Use **`DeleteStateMachine`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteStateMachine`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos da orquestração de fluxo de trabalaho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// state machine.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteStateMachine(string stateMachineArn)
    {
        var response = await _amazonStepFunctions.DeleteStateMachineAsync(new
DeleteStateMachineRequest
        { StateMachineArn = stateMachineArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteStateMachinea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteMachine(SfnClient sfnClient, String stateMachineArn)
{
       try {
           DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
               .stateMachineArn(stateMachineArn)
               .build();

           sfnClient.deleteStateMachine(deleteStateMachineRequest);
           DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
               .stateMachineArn(stateMachineArn)
               .build();

           while (true) {
               DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
               System.out.println("The state machine is not deleted yet. The
status is " + response.status());
               Thread.sleep(3000);
           }

       } catch (SfnException | InterruptedException e) {
           System.err.println(e.getMessage());
       }
       System.out.println(stateMachineArn + " was successfully deleted.");
   }
```

- Para obter detalhes da API, consulte DeleteStateMachinea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteStateMachinereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def delete(self, state_machine_arn):
        """
        Delete a state machine and all of its run data.

        :param state_machine_arn: The ARN of the state machine to delete.
        """
        try:
            response = self.stepfunctions_client.delete_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteStateMachineReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_sfn->deletestatemachine(
      iv_statemachinearn = iv_state_machine_arn
    ).
    MESSAGE 'State machine deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfnvalidationex.
    MESSAGE 'Validation error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteStateMachinereferência da API AWS SDK for SAP ABAP.

Use **`DescribeExecution`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeExecution`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos da orquestração de fluxo de trabalaho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve information about the specified Step Functions execution.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of the
    /// Step Functions execution.</param>
    /// <returns>The API response returned by the API.</returns>
    public async Task<DescribeExecutionResponse> DescribeExecutionAsync(string
executionArn)
    {
        var response = await _amazonStepFunctions.DescribeExecutionAsync(new
DescribeExecutionRequest { ExecutionArn = executionArn });
        return response;
    }
```

- Para obter detalhes da API, consulte DescribeExecutiona Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void describeExe(SfnClient sfnClient, String executionArn) {
```

```
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has
succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeExecutiona Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}
```

- Para obter detalhes da API, consulte a DescribeExecutionreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def describe_run(self, run_arn):
        """
        Get data about a state machine run, such as its current status or final
output.

        :param run_arn: The ARN of the run to look up.
        :return: The retrieved run data.
        """
        try:
            response = self.stepfunctions_client.describe_execution(
                executionArn=run_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe run %s. Here's why: %s: %s",
                run_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DescribeExecutionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_sfn->describeexecution(
      iv_executionarn = iv_execution_arn
    ).
    MESSAGE 'Execution described successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnexecdoesnotexist.
    MESSAGE 'Execution does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid execution ARN.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeExecutionreferência da API AWS SDK for SAP ABAP.

Use **`DescribeStateMachine`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeStateMachine`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve information about the specified Step Functions state machine.
    /// </summary>
    /// <param name="StateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine to retrieve.</param>
    /// <returns>Information about the specified Step Functions state machine.</
returns>
    public async Task<DescribeStateMachineResponse>
 DescribeStateMachineAsync(string StateMachineArn)
    {
        var response = await _amazonStepFunctions.DescribeStateMachineAsync(new
 DescribeStateMachineRequest { StateMachineArn = StateMachineArn });
        return response;
    }
```

- Para obter detalhes da API, consulte DescribeStateMachinea Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void describeStateMachine(SfnClient sfnClient, String
 stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
 DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            DescribeStateMachineResponse response =
 sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
 response.name());
            System.out.println("The status of the State machine is " +
 response.status());
            System.out.println("The ARN value of the State machine is " +
 response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }
```

- Para obter detalhes da API, consulte DescribeStateMachinea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
```

```
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is
 ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeStateMachinereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def describe(self, state_machine_arn):
        """
        Get data about a state machine.

        :param state_machine_arn: The ARN of the state machine to look up.
        :return: The retrieved state machine data.
```

```
        """
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DescribeStateMachineReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sfn->describestatemachine(
          iv_statemachinearn = iv_state_machine_arn
        ).
        MESSAGE 'State machine described successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfnstatemachinedoes00.
        MESSAGE 'State machine does not exist.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeStateMachinereferência da API AWS SDK for SAP ABAP.

Use **`GetActivityTask`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `GetActivityTask`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve a task with the specified Step Functions activity
    /// with the specified Amazon Resource Name (ARN).
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the Step Functions activity.</param>
    /// <param name="workerName">The name of the Step Functions worker.</param>
    /// <returns>The response from the Step Functions activity.</returns>
    public async Task<GetActivityTaskResponse> GetActivityTaskAsync(string
activityArn, string workerName)
    {
        var response = await _amazonStepFunctions.GetActivityTaskAsync(new
GetActivityTaskRequest
        { ActivityArn = activityArn, WorkerName = workerName });
        return response;
    }
```

- Para obter detalhes da API, consulte GetActivityTaska Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static List<String> getActivityTask(SfnClient sfnClient, String
actArn) {
       List<String> myList = new ArrayList<>();
       GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
           .activityArn(actArn)
           .build();

       GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
       myList.add(response.taskToken());
       myList.add(response.input());
       return myList;
   }
```

- Para obter detalhes da API, consulte GetActivityTaska Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

- Para obter detalhes da API, consulte a GetActivityTaskreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
```

```
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def get_task(self, activity_arn):
        """
        Gets task data for an activity. When a state machine is waiting for the
        specified activity, a response is returned with data from the state
machine.
        When a state machine is not waiting, this call blocks for 60 seconds.

        :param activity_arn: The ARN of the activity to get task data for.
        :return: The task data for the activity.
        """
        try:
            response = self.stepfunctions_client.get_activity_task(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't get a task for activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a GetActivityTaskReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_sfn->getactivitytask(
      iv_activityarn = iv_activity_arn
    ).
    MESSAGE 'Activity task retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnactivitydoesnotex.
    MESSAGE 'Activity does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid activity ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfnactivityworkerlm00.
    MESSAGE 'Activity worker limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetActivityTaskreferência da API AWS SDK for SAP ABAP.

Use **`ListActivities`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListActivities`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the Step Functions activities for the current account.
    /// </summary>
    /// <returns>A list of ActivityListItems.</returns>
    public async Task<List<ActivityListItem>> ListActivitiesAsync()
    {
        var request = new ListActivitiesRequest();
        var activities = new List<ActivityListItem>();

        do
        {
            var response = await
_amazonStepFunctions.ListActivitiesAsync(request);

            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }

            activities.AddRange(response.Activities);
        }
        while (request.NextToken is not null);

        return activities;
    }
```

- Para obter detalhes da API, consulte ListActivitiesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
 ListActivitiesRequest.builder()
```

```
                    .maxResults(10)
                    .build();

            ListActivitiesResponse response =
 sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListActivitiesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listActivities(activitiesRequest)
        response.activities?.forEach { item ->
```

```
            println("The activity ARN is ${item.activityArn}")
            println("The activity name is ${item.name}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListActivitiesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def find(self, name):
        """
        Find an activity by name. This requires listing activities until one is
 found
        with a matching name.

        :param name: The name of the activity to search for.
        :return: If found, the ARN of the activity; otherwise, None.
        """
        try:
            paginator =
 self.stepfunctions_client.get_paginator("list_activities")
            for page in paginator.paginate():
```

```
                for activity in page.get("activities", []):
                    if activity["name"] == name:
                        return activity["activityArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list activities. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListActivitiesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_sfn->listactivities( ).
    DATA(lt_activities) = lo_result->get_activities( ).
    LOOP AT lt_activities INTO DATA(lo_activity).
      IF lo_activity->get_name( ) = iv_name.
        ov_activity_arn = lo_activity->get_activityarn( ).
        EXIT.
      ENDIF.
    ENDLOOP.
    MESSAGE 'Activities listed successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfninvalidtoken.
    MESSAGE 'Invalid pagination token.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListActivitiesreferência da API AWS SDK for SAP ABAP.

Use **`ListExecutions`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListExecutions`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Retrieve information about executions of a Step Functions
   /// state machine.
   /// </summary>
   /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
   /// Step Functions state machine.</param>
   /// <returns>A list of ExecutionListItem objects.</returns>
   public async Task<List<ExecutionListItem>> ListExecutionsAsync(string
stateMachineArn)
   {
       var executions = new List<ExecutionListItem>();
       ListExecutionsResponse response;
       var request = new ListExecutionsRequest { StateMachineArn =
stateMachineArn };

       do
       {
           response = await _amazonStepFunctions.ListExecutionsAsync(request);
           executions.AddRange(response.Executions);
           if (response.NextToken is not null)
           {
               request.NextToken = response.NextToken;
           }
       } while (response.NextToken is not null);
```

```
        return executions;
    }
```

- Para obter detalhes da API, consulte ListExecutionsa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void getExeHistory(SfnClient sfnClient, String exeARN) {
        try {
            GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
                    .executionArn(exeARN)
                    .maxResults(10)
                    .build();

            GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
            List<HistoryEvent> events = historyResponse.events();
            for (HistoryEvent event : events) {
                System.out.println("The event type is " +
event.type().toString());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListExecutionsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getExeHistory(exeARN: String?) {
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListExecutionsreferência da API AWS SDK for Kotlin.

Use **`ListStateMachines`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListStateMachines`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve a list of Step Functions state machines.
    /// </summary>
    /// <returns>A list of StateMachineListItem objects.</returns>
    public async Task<List<StateMachineListItem>> ListStateMachinesAsync()
    {
        var stateMachines = new List<StateMachineListItem>();
        var listStateMachinesPaginator =
            _amazonStepFunctions.Paginators.ListStateMachines(new
ListStateMachinesRequest());

        await foreach (var response in listStateMachinesPaginator.Responses)
        {
            stateMachines.AddRange(response.StateMachines);
        }

        return stateMachines;
    }
```

- Para obter detalhes da API, consulte ListStateMachinesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
```

```
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListStateMachinesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
```

```
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListStateMachinesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Encontre uma máquina de estado pelo nome pesquisando a lista de máquinas de estado da conta.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
```

```
    def find(self, name):
        """
        Find a state machine by name. This requires listing the state machines
until
        one is found with a matching name.

        :param name: The name of the state machine to search for.
        :return: The ARN of the state machine if found; otherwise, None.
        """
        try:
            paginator =
self.stepfunctions_client.get_paginator("list_state_machines")
            for page in paginator.paginate():
                for state_machine in page.get("stateMachines", []):
                    if state_machine["name"] == name:
                        return state_machine["stateMachineArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list state machines. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListStateMachinesReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_sfn->liststatemachines( ).
        DATA(lt_state_machines) = lo_result->get_statemachines( ).
        LOOP AT lt_state_machines INTO DATA(lo_state_machine).
          IF lo_state_machine->get_name( ) = iv_name.
            ov_state_machine_arn = lo_state_machine->get_statemachinearn( ).
            EXIT.
          ENDIF.
        ENDLOOP.
        MESSAGE 'State machines listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfninvalidtoken.
        MESSAGE 'Invalid pagination token.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListStateMachinesreferência da API AWS SDK for SAP ABAP.

Use **SendTaskSuccess**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `SendTaskSuccess`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Indicate that the Step Functions task, indicated by the
    /// task token, has completed successfully.
```

```
    /// </summary>
    /// <param name="taskToken">Identifies the task.</param>
    /// <param name="taskResponse">The response received from executing the
task.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SendTaskSuccessAsync(string taskToken, string
taskResponse)
    {
        var response = await _amazonStepFunctions.SendTaskSuccessAsync(new
SendTaskSuccessRequest
        { TaskToken = taskToken, Output = taskResponse });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte SendTaskSuccessa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
        try {
            SendTaskSuccessRequest successRequest =
SendTaskSuccessRequest.builder()
                .taskToken(token)
                .output(json)
                .build();

            sfnClient.sendTaskSuccess(successRequest);
```

```
        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte SendTaskSuccessa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- Para obter detalhes da API, consulte a SendTaskSuccessreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def send_task_success(self, task_token, task_response):
        """
        Sends a success response to a waiting activity step. A state machine with
 an
        activity step waits for the activity to get task data and then respond
 with
        either success or failure before it resumes processing.

        :param task_token: The token associated with the task. This is included
 in the
                           response to the get_activity_task action and must be
 sent
                           without modification.
        :param task_response: The response data from the activity. This data is
                              received and processed by the state machine.
        """
        try:
            self.stepfunctions_client.send_task_success(
                taskToken=task_token, output=task_response
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't send task success. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a SendTaskSuccessReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sfn->sendtasksuccess(
          iv_tasktoken = iv_task_token
          iv_output = iv_task_response
        ).
        MESSAGE 'Task success sent successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfninvalidtoken.
        MESSAGE 'Invalid task token.' TYPE 'E'.
      CATCH /aws1/cx_sfntaskdoesnotexist.
        MESSAGE 'Task does not exist.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidoutput.
        MESSAGE 'Invalid task output.' TYPE 'E'.
      CATCH /aws1/cx_sfntasktimedout.
        MESSAGE 'Task timed out.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendTaskSuccessreferência da API AWS SDK for SAP ABAP.

Use **`StartExecution`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartExecution`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Conceitos básicos da orquestração de fluxo de trabalaho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start execution of an AWS Step Functions state machine.
    /// </summary>
    /// <param name="executionName">The name to use for the execution.</param>
    /// <param name="executionJson">The JSON string to pass for execution.</
param>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>The Amazon Resource Name (ARN) of the AWS Step Functions
    /// execution.</returns>
    public async Task<string> StartExecutionAsync(string executionJson, string
 stateMachineArn)
    {
        var executionRequest = new StartExecutionRequest
        {
            Input = executionJson,
            StateMachineArn = stateMachineArn
        };

        var response = await
 _amazonStepFunctions.StartExecutionAsync(executionRequest);
```

```
        return response.ExecutionArn;
    }
```

- Para obter detalhes da API, consulte StartExecutiona Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String startWorkflow(SfnClient sfnClient, String
stateMachineArn, String jsonEx) {
       UUID uuid = UUID.randomUUID();
       String uuidValue = uuid.toString();
       try {
           StartExecutionRequest executionRequest =
StartExecutionRequest.builder()
               .input(jsonEx)
               .stateMachineArn(stateMachineArn)
               .name(uuidValue)
               .build();

           StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
           return response.executionArn();

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte StartExecutiona Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SFNClient, StartExecutionCommand } from "@aws-sdk/client-sfn";

/**
 * @param {{ sfnClient: SFNClient, stateMachineArn: string }} config
 */
export async function startExecution({ sfnClient, stateMachineArn }) {
  const response = await sfnClient.send(
    new StartExecutionCommand({
      stateMachineArn,
    }),
  );
  console.log(response);
  // Example response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '202a9309-c16a-454b-adeb-c4d19afe3bf2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   executionArn: 'arn:aws:states:us-
east-1:000000000000:execution:MyStateMachine:aaaaaaaa-f787-49fb-
a20c-1b61c64eafe6',
  //   startDate: 2024-01-04T15:54:08.362Z
  // }
```

```
  return response;
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  startExecution({ sfnClient: new SFNClient({}), stateMachineArn: "ARN" });
}
```

- Para obter detalhes da API, consulte StartExecutiona Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- Para obter detalhes da API, consulte a StartExecutionreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def start(self, state_machine_arn, run_input):
        """
        Start a run of a state machine with a specified input. A run is also
 known
        as an "execution" in Step Functions.

        :param state_machine_arn: The ARN of the state machine to run.
        :param run_input: The input to the state machine, in JSON format.
        :return: The ARN of the run. This can be used to get information about
 the run,
                 including its current status and final output.
        """
        try:
            response = self.stepfunctions_client.start_execution(
                stateMachineArn=state_machine_arn, input=run_input
            )
        except ClientError as err:
            logger.error(
                "Couldn't start state machine %s. Here's why: %s: %s",
                state_machine_arn,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["executionArn"]
```

- Para obter detalhes da API, consulte a StartExecutionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_sfn->startexecution(
      iv_statemachinearn = iv_state_machine_arn
      iv_input = iv_input
    ).
    ov_execution_arn = lo_result->get_executionarn( ).
    MESSAGE 'Execution started successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnstatemachinedoes00.
    MESSAGE 'State machine does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidexecinput.
    MESSAGE 'Invalid execution input.' TYPE 'E'.
  CATCH /aws1/cx_sfnexeclimitexceeded.
    MESSAGE 'Execution limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a StartExecutionreferência da API AWS SDK for SAP ABAP.

Cenários para Step Functions usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns em Step Functions com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Step Functions ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação de mensageiro com o Step Functions
- Conceitos básicos da orquestração de fluxo de trabalaho
- Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions
- Usar Step Functions para invocar funções do Lambda

Criar uma aplicação de mensageiro com o Step Functions

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.

- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

Conceitos básicos da orquestração de fluxo de trabalaho

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Step Functions.
- Criar sua primeira máquina de estado.
- Iniciar a execução da máquina de estado.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS
 Step Functions Getting Started tutorial

set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
done

# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT

echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"
```

```
# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be
 less reliable."
    echo "Consider installing jq for better error handling: brew install jq
 (macOS) or apt-get install jq (Ubuntu)"
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
export AWS_REGION="$CURRENT_REGION"

# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
```

```
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}

# Function to extract JSON field efficiently
extract_json_field() {
```

```
    local json="$1"
    local field="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] ||
 [ "$wait_time" -gt 300 ]; then
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"

    if [[ "$USE_JQ" == "true" ]]; then
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
```

```
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."

    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
 $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for detach operations to complete
    wait 2>/dev/null || true
```

```
    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/
null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed
 to delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
        "input.json"
        "sentiment-input.json"
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
            fi
        fi
    done
```

```
}

# Security: Set trap to cleanup on script exit
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
```

```
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
```

```
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)

check_api_error "$ROLE_RESULT" "Create IAM role"
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"
```

```
# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")

if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi

# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs
 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)

check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"
```

```
# Wait for execution to complete (the workflow has a 10-second wait state)
echo "Waiting for execution to complete (12 seconds)..."
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
```

```
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
```

```
    }
  }
}
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)

check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"
```

```
# Wait for execution to complete (the workflow has a 5-second wait state)
echo "Waiting for execution to complete (8 seconds)..."
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")

    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
```

```
    fi
    echo "Policy ARN: $POLICY_ARN"

    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
```

```
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
```

```
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment
 analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"

    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
```

```
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)

    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3

    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in
 $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
```

```
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
fi
echo "==========================================="

# Cleanup
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy
  - StartExecution
  - UpdateStateMachine

Construir e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

Python

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Step Functions

Usar Step Functions para invocar funções do Lambda

O exemplo de código a seguir mostra como criar uma máquina de AWS Step Functions estado que invoca AWS Lambda funções em sequência.

Java

SDK para Java 2.x

Mostra como criar um fluxo de trabalho AWS sem servidor usando AWS Step Functions e. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Exemplos de código para AWS STS usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Security Token Service com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS STS Guia do usuário — Mais informações sobre AWS STS.
- AWS STS Referência da API — Detalhes sobre todas as AWS STS ações disponíveis.

- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS STS usar AWS SDKs
  - Ações para AWS STS usar AWS SDKs
    - Use AssumeRolecom um AWS SDK ou CLI
    - Usar AssumeRoleWithWebIdentity com uma CLI
    - Usar DecodeAuthorizationMessage com uma CLI
    - Usar GetFederationToken com uma CLI
    - Use GetSessionTokencom um AWS SDK ou CLI
- Cenários para AWS STS usar AWS SDKs
  - Assuma uma função do IAM que exija um token de MFA com AWS STS usando um AWS SDK
  - Configurar a conectividade do serviço de contêiner
  - Construa um URL com AWS STS para usuários federados usando um AWS SDK
  - Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
  - Criar uma API REST com integração de proxy de função
  - Como criar um serviço de contêiner para instâncias de máquinas virtuais
  - Como criar um espaço de trabalho de monitoramento gerenciado
  - Como criar um painel de monitoramento com o nome da função como uma variável
  - Obtenha um token de sessão que exija um token de MFA com AWS STS usando um AWS SDK
  - Conceitos básicos do gerenciamento de configurações
  - Conceitos básicos de registros de contêineres
  - Conceitos básicos de mensagens da Internet das Coisas
  - Conceitos básicos de arquivos de atributos de machine learning
  - Conceitos básicos de clusters gerenciados de kubernetes
  - Conceitos básicos de streaming gerenciado
  - Conceitos básicos das notificações push
  - Conceitos básicos dos mecanismos de pesquisa e analytics

- Conceitos básicos de firewalls do aplicativo Web
- Como mover segredos codificados para o armazenamento seguro de segredos
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Como configurar o gerenciamento de sistemas

Exemplos básicos para AWS STS usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Security Token Service com AWS SDKs.

Exemplos

- Ações para AWS STS usar AWS SDKs
  - Use AssumeRolecom um AWS SDK ou CLI
  - Usar AssumeRoleWithWebIdentity com uma CLI
  - Usar DecodeAuthorizationMessage com uma CLI
  - Usar GetFederationToken com uma CLI
  - Use GetSessionTokencom um AWS SDK ou CLI

Ações para AWS STS usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS STS ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a AWS STS API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para AWS STS usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Security Token Service.

Exemplos

- Use AssumeRolecom um AWS SDK ou CLI
- Usar AssumeRoleWithWebIdentity com uma CLI

- Usar DecodeAuthorizationMessage com uma CLI
- Usar GetFederationToken com uma CLI
- Use GetSessionTokencom um AWS SDK ou CLI

Use **`AssumeRole`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AssumeRole`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Assumir um perfil do IAM que exija um token de MFA
- Criar um URL para usuários federados usando

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
```

```
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        // A region property may be used if the profile or credentials loaded do
 not specify a region,
        // or to use a specific region.
        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of
 the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
```

```
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials, REGION);

            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- Para obter detalhes da API, consulte AssumeRolea Referência AWS SDK para .NET da API.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
```

```
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary
 credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte AssumeRoleem Referência de AWS CLI Comandos.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
```

```
                                Aws::Auth::AWSCredentials &credentials,
                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AssumeRolea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como assumir um perfil

O comando `assume-role`, apresentado a seguir, recupera um conjunto de credenciais de curto prazo para o perfil do IAM `s3-access-example`.

```
aws sts assume-role \
    --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
    --role-session-name s3-access-example
```

Saída:

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

A saída do comando contém uma chave de acesso, uma chave secreta e um token de sessão que você pode usar para se autenticar na AWS.

Para uso da AWS CLI, você pode configurar um perfil nomeado associado a uma função. Quando você usa o perfil, a AWS CLI chama assume-role e gerencia as credenciais para você. Para obter mais informações, consulte Usar uma função do IAM na AWS CLI no Guia do usuário da AWS CLI.

- Para obter detalhes da API, consulte AssumeRoleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version":"2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
```

```
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to
 assume (for example, arn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn,
 String roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
```

```
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " +
 exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte AssumeRolea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

Assuma um perfil do IAM.

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte AssumeRolea Referência AWS SDK para JavaScript da API.

SDK para JavaScript (v2)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);
```

```
  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- Para obter detalhes da API, consulte AssumeRolea Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna um conjunto de credenciais temporárias (chave de acesso, chave secreta e token de sessão) que podem ser usadas por uma hora para acessar AWS recursos aos quais o usuário solicitante normalmente não teria acesso. As credenciais retornadas têm as permissões permitidas pela política de acesso do perfil assumido e pela política fornecida (não é possível usar a política fornecida para conceder permissões além das definidas pela política de acesso do perfil que está sendo assumido).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

Exemplo 2: retorna um conjunto de credenciais temporárias, válidas por uma hora, que têm as mesmas permissões definidas na política de acesso do perfil que está sendo assumido.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

Exemplo 3: retorna um conjunto de credenciais temporárias que fornecem o número de série e o token gerado de uma MFA associada às credenciais do usuário usadas para executar o cmdlet.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Exemplo 4: retorna um conjunto de credenciais temporárias que assumiram um perfil definido em uma conta de cliente. Para cada função que o terceiro possa assumir, a conta do cliente deve criar uma função usando um identificador que deve ser passado no ExternalId parâmetro - sempre que a função for assumida.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- Para obter detalhes da API, consulte AssumeRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna um conjunto de credenciais temporárias (chave de acesso, chave secreta e token de sessão) que podem ser usadas por uma hora para acessar AWS recursos aos quais o usuário solicitante normalmente não teria acesso. As credenciais retornadas têm as permissões permitidas pela política de acesso do perfil assumido e pela política fornecida (não é possível usar a política fornecida para conceder permissões além das definidas pela política de acesso do perfil que está sendo assumido).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

Exemplo 2: retorna um conjunto de credenciais temporárias, válidas por uma hora, que têm as mesmas permissões definidas na política de acesso do perfil que está sendo assumido.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

Exemplo 3: retorna um conjunto de credenciais temporárias que fornecem o número de série e o token gerado de uma MFA associada às credenciais do usuário usadas para executar o cmdlet.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Exemplo 4: retorna um conjunto de credenciais temporárias que assumiram um perfil definido em uma conta de cliente. Para cada função que o terceiro possa assumir, a conta do cliente deve criar uma função usando um identificador que deve ser passado no ExternalId parâmetro - sempre que a função for assumida.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- Para obter detalhes da API, consulte AssumeRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Assuma um perfil do IAM que exija um token de MFA e use credenciais temporárias para listar os buckets do Amazon S3 para a conta.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
```

```
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- Para obter detalhes da API, consulte a AssumeRoleReferência da API AWS SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
```

```
  # @param key_secret [String] The secret part of the access key used by the STS
 client.
  def create_sts_client(key_id, key_secret)
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
 credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
 assume the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end
```

- Para obter detalhes da API, consulte AssumeRolea Referência AWS SDK para Ruby da API.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
```

```
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}",
 e.user_id().unwrap_or_default());
            println!("Account:               {}",
 e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- Para obter detalhes da API, consulte a AssumeRolereferência da API AWS SDK for Rust.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSTS

    public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
        async throws -> STSClientTypes.Credentials
    {
```

```
        let input = AssumeRoleInput(
            roleArn: role.arn,
            roleSessionName: sessionName
        )
        do {
            let output = try await stsClient.assumeRole(input: input)

            guard let credentials = output.credentials else {
                throw ServiceHandlerError.authError
            }

            return credentials
        } catch {
            print("Error assuming role: ", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte AssumeRolea referência da API AWS SDK for Swift.

Usar **`AssumeRoleWithWebIdentity`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AssumeRoleWithWebIdentity`.

CLI

AWS CLI

Obter credenciais de curto prazo para um perfil autenticado com identidade Web (OAuth 2."0)

O comando `assume-role-with-web-identity`, apresentado a seguir, recupera um conjunto de credenciais de curto prazo para o perfil do IAM `app1`. A solicitação é autenticada com o token de identidade Web fornecido pelo provedor de identidade Web especificado. Duas políticas adicionais são aplicadas à sessão para restringir ainda mais o que o usuário pode fazer. As credenciais retornadas expiram uma hora após serem geradas.

```
aws sts assume-role-with-web-identity \
    --duration-seconds 3600 \
    --role-session-name "app1" \
    --provider-id "www.amazon.com" \
```

```
    --policy-arns "arn:aws:iam::123456789012:policy/
q=webidentitydemopolicy1","arn:aws:iam::123456789012:policy/
webidentitydemopolicy2" \
    --role-arn arn:aws:iam::123456789012:role/FederatedWebIdentityRole \
    --web-identity-token "Atza
%7CIQEBLjAsAhRFiXuWpUXuRvQ9PZL3GMFcYevydwIUFAHZwXZXXXXXXXXJnrulxKDHwy87oGKPznh0D6bEQZTSCz
CrKqjG7nPBjNIL016GGvuS5gSvPRUxWES3VYfm1wl7WTI7jn-Pcb6M-
buCgHhFOzTQxod27L9CqnOLio7N3gZAGpsp6n1-
AJBOCJckcyXe2c6uD0srOJeZlKUm2eTDVMf8IehDVI0r1QOnTV6KzzAI3OY87Vd_cVMQ"
```

Saída:

```
{
    "SubjectFromWebIdentityToken": "amzn1.account.AF6RHO7KZU5XRVQJGXK6HB56KR2A",
    "Audience": "client.5498841531868486423.1548@apps.example.com",
    "AssumedRoleUser": {
        "Arn": "arn:aws:sts::123456789012:assumed-role/FederatedWebIdentityRole/
app1",
        "AssumedRoleId": "AROACLKWSDQRAOEXAMPLE:app1"
    },
    "Credentials": {
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    },
    "Provider": "www.amazon.com"
}
```

Para obter mais informações, consulte Solicitação de credenciais de segurança temporárias no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte AssumeRoleWithWebIdentityem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna um conjunto temporário de credenciais, válido por uma hora, para um usuário que foi autenticado com o provedor de identidade Login with Amazon. As credenciais assumem a política de acesso associada ao perfil identificado pelo ARN do perfil. Opcionalmente, você pode transmitir uma política JSON ao parâmetro -Policy que refina ainda mais as permissões de acesso (não é possível conceder mais permissões do que as disponíveis nas permissões associadas ao perfil). O valor fornecido ao - WebIdentityToken é o identificador de usuário exclusivo que foi retornado pelo provedor de identidade.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Para obter detalhes da API, consulte AssumeRoleWithWebIdentityem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna um conjunto temporário de credenciais, válido por uma hora, para um usuário que foi autenticado com o provedor de identidade Login with Amazon. As credenciais assumem a política de acesso associada ao perfil identificado pelo ARN do perfil. Opcionalmente, você pode transmitir uma política JSON ao parâmetro -Policy que refina ainda mais as permissões de acesso (não é possível conceder mais permissões do que as disponíveis nas permissões associadas ao perfil). O valor fornecido ao - WebIdentityToken é o identificador de usuário exclusivo que foi retornado pelo provedor de identidade.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Para obter detalhes da API, consulte AssumeRoleWithWebIdentityem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DecodeAuthorizationMessage`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DecodeAuthorizationMessage`.

CLI

AWS CLI

Para decodificar uma mensagem de autorização codificada retornada em resposta a uma solicitação

O exemplo `decode-authorization-message` a seguir decodifica informações adicionais sobre o status da autorização de uma solicitação de uma mensagem codificada retornada em resposta a uma solicitação da Amazon Web Services.

```
aws sts decode-authorization-message \
    --encoded-message EXAMPLEWodyRNrtlQARDip-
eTA6i6DrlUhHhPQrLWB_lAbl5pAKxl9mPDLexYcGBreyIKQC1BGBIpBKr3dFDkwqeO7e2NMk5j_hmzAiChJN-8oy3
Ojau7BMjOTWwOtHPhV_Zaz87yENdipr745EjQwRd5LaoL3vN8_5ZfA9UiBMKDgVh1gjqZJFUiQoubv78V1RbHNYnK
pOu3FZjwYStfvTb3GHs3-6rLribGO9jZOktkfE6vqxlFzLyeDr4P2ihC1wty9tArCvvGzIAUNmARQJ2VVWPxioqgo
JWP5pwe_mAyqh0NLw-r1S56YC_90onj9A80sNrHlI-
tIiNd7tgNTYzDuPQYD2FMDBnp82V9eVmYGtPp5NIeSpuf3fOHanFuBZgENxZQZ2dlH3xJGMTtYayzZrRXjiq_SfX9
FaoPIb8LmmKVBLpIB0iFhU9sEHPqKHVPi6jdxXqKaZaFGvYVmVOiuQdNQKuyk0p067POFrZECLjjOtNPBOZCcuEKE
```

Saída:

```
{
    "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":true,
\"matchedStatements\":{\"items\":[{\"statementId\":\"VisualEditor0\",\"effect
\":\"DENY\",\"principals\":{\"items\":[{\"value\":\"AROA123456789EXAMPLE
\"}]},\"principalGroups\":{\"items\":[]},\"actions\":{\"items\":[{\"value
\":\"ec2:RunInstances\"}]},\"resources\":{\"items\":[{\"value\":\"*
\"}]},\"conditions\":{\"items\":[]}}]},\"failures\":{\"items\":[]},
\"context\":{\"principal\":{\"id\":\"AROA123456789EXAMPLE:Ana\",\"arn
\":\"arn:aws:sts::111122223333:assumed-role/Developer/Ana\"},\"action\":
\"RunInstances\",\"resource\":\"arn:aws:ec2:us-east-1:111122223333:instance/*
\",\"conditions\":{\"items\":[{\"key\":\"ec2:MetadataHttpPutResponseHopLimit\",
\"values\":{\"items\":[{\"value\":\"2\"}]}},{\"key\":\"ec2:InstanceMarketType
\",\"values\":{\"items\":[{\"value\":\"on-demand\"}]}},{\"key\":\"aws:Resource
\",\"values\":{\"items\":[{\"value\":\"instance/*\"}]}},{\"key\":\"aws:Account
\",\"values\":{\"items\":[{\"value\":\"111122223333\"}]}},{\"key\":
\"ec2:AvailabilityZone\",\"values\":{\"items\":[{\"value\":\"us-east-1f\"}]}},
{\"key\":\"ec2:ebsOptimized\",\"values\":{\"items\":[{\"value\":\"false\"}]}},
{\"key\":\"ec2:IsLaunchTemplateResource\",\"values\":{\"items\":[{\"value\":
\"false\"}]}},{\"key\":\"ec2:InstanceType\",\"values\":{\"items\":[{\"value
\":\"t2.micro\"}]}},{\"key\":\"ec2:RootDeviceType\",\"values\":{\"items\":
```

```
[{\"value\":\"ebs\"}]}},{\"key\":\"aws:Region\",\"values\":{\"items\":[{\"value
\":\"us-east-1\"}]}},{\"key\":\"ec2:MetadataHttpEndpoint\",\"values\":{\"items
\":[{\"value\":\"enabled\"}]}},{\"key\":\"aws:Service\",\"values\":{\"items
\":[{\"value\":\"ec2\"}]}},{\"key\":\"ec2:InstanceID\",\"values\":{\"items\":
[{\"value\":\"*\"}]}},{\"key\":\"ec2:MetadataHttpTokens\",\"values\":{\"items
\":[{\"value\":\"required\"}]}},{\"key\":\"aws:Type\",\"values\":{\"items
\":[{\"value\":\"instance\"}]}},{\"key\":\"ec2:Tenancy\",\"values\":{\"items
\":[{\"value\":\"default\"}]}},{\"key\":\"ec2:Region\",\"values\":{\"items
\":[{\"value\":\"us-east-1\"}]}},{\"key\":\"aws:ARN\",\"values\":{\"items\":
[{\"value\":\"arn:aws:ec2:us-east-1:111122223333:instance/*\"}]}}]}}}"
}
```

Para obter mais informações, consulte Lógica da avaliação de política no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte DecodeAuthorizationMessageem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: decodifica as informações adicionais contidas no conteúdo da mensagem codificada fornecida que foi retornada em resposta a uma solicitação. As informações adicionais são codificadas porque os detalhes do status da autorização podem constituir informações privilegiadas que o usuário responsável por solicitar a ação não deve ver.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Para obter detalhes da API, consulte DecodeAuthorizationMessageem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: decodifica as informações adicionais contidas no conteúdo da mensagem codificada fornecida que foi retornada em resposta a uma solicitação. As informações adicionais são codificadas porque os detalhes do status da autorização podem constituir informações privilegiadas que o usuário responsável por solicitar a ação não deve ver.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Para obter detalhes da API, consulte DecodeAuthorizationMessageem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetFederationToken`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetFederationToken`.

CLI

AWS CLI

Para retornar um conjunto de credenciais de segurança temporárias usando as credenciais da chave de acesso do usuário do IAM

O exemplo `get-federation-token` a seguir retorna um conjunto de credenciais de segurança temporárias (que consistem em um ID de chave de acesso, uma chave de acesso secreta e um token de segurança) para um usuário. Você deve chamar a operação `GetFederationToken` usando as credenciais de segurança de longo prazo de um usuário do IAM.

```
aws sts get-federation-token \
    --name Bob \
    --policy file://myfile.json \
    --policy-arns arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
    --duration-seconds 900
```

Conteúdo de `myfile.json`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe*",
            "Resource": "*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe*",
            "Resource": "*"
        }
    ]
}
```

Saída:

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": "EXAMPLEpZ2luX2VjEGoaCXVzLXdlc3QtMiJIMEYCIQC/
W9pL5ArQyDD5JwFL3/h5+WGopQ24GEXweNctwhi9sgIhAMkg
+MZE35iWM8s4r5Lr25f9rSTVPFH98G42QQunWMTfKq0DCOP/////////
wEQAxoMNDUyOTI1MTcwNTA3Igxuy3AOpuuoLsk3MJwqgQPg8QOd9HuoClUxq26wnc/nm
+eZLjHDyGf2KUAHK2DuaS/nrGSEXAMPLE",
        "Expiration": "2023-12-20T02:06:07+00:00"
    },
    "FederatedUser": {
        "FederatedUserId": "111122223333:Bob",
        "Arn": "arn:aws:sts::111122223333:federated-user/Bob"
    },
    "PackedPolicySize": 36
}
```

Para obter mais informações, consulte Solicitação de credenciais de segurança temporárias no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetFederationTokenem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: solicita um token federado válido por uma hora usando "Bob" como nome do usuário federado. Esse nome pode ser usado para referenciar o nome do usuário federado em uma política baseada em recursos (como uma política de bucket do Amazon S3). A política do IAM fornecida, no formato JSON, é usada para definir o escopo das permissões que estão disponíveis para o usuário do IAM. A política fornecida não pode conceder mais permissões do que as concedidas ao usuário solicitante, com as permissões finais do usuário federado sendo o conjunto mais restritivo com base na interseção da política aprovada com a política de usuário do IAM.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- Para obter detalhes da API, consulte GetFederationTokenem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: solicita um token federado válido por uma hora usando "Bob" como nome do usuário federado. Esse nome pode ser usado para referenciar o nome do usuário federado em uma política baseada em recursos (como uma política de bucket do Amazon S3). A política do IAM fornecida, no formato JSON, é usada para definir o escopo das permissões que estão disponíveis para o usuário do IAM. A política fornecida não pode conceder mais permissões do que as concedidas ao usuário solicitante, com as permissões finais do usuário federado sendo o conjunto mais restritivo com base na interseção da política aprovada com a política de usuário do IAM.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- Para obter detalhes da API, consulte GetFederationTokenem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetSessionToken`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetSessionToken`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Obtenha um token de sessão que requeira um token de MFA

CLI

AWS CLI

Como obter um conjunto de credenciais de curto prazo para uma identidade do IAM

O comando `get-session-token`, apresentado a seguir, recupera um conjunto de credenciais de curto prazo para a identidade do IAM que executa a chamada. As credenciais resultantes podem ser usadas para solicitações em que a autenticação multifator (MFA) é requerida pela política. As credenciais expiram 15 minutos após serem geradas.

```
aws sts get-session-token \
    --duration-seconds 900 \
    --serial-number "YourMFADeviceSerialNumber" \
    --token-code 123456
```

Saída:

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    }
}
```

Para obter mais informações, consulte Solicitação de credenciais de segurança temporárias no Guia do usuário do AWS IAM.

- Para obter detalhes da API, consulte GetSessionTokenem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por um determinado período. As credenciais usadas para solicitar credenciais temporárias são inferidas dos padrões atuais do shell. Para especificar outras credenciais, use os parâmetros - ProfileName ou - AccessKey /-SecretKey .

```
Get-STSSessionToken
```

Saída:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Exemplo 2: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por uma hora. As credenciais usadas para fazer a solicitação são obtidas do perfil especificado.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Saída:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Exemplo 3: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por uma hora usando o número de identificação do dispositivo de MFA associado à conta cujas credenciais estão especificadas no perfil 'myprofilename' e o valor fornecido pelo dispositivo.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Saída:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- Para obter detalhes da API, consulte GetSessionTokenem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por um determinado período. As credenciais usadas para solicitar credenciais temporárias são inferidas dos padrões atuais do shell. Para especificar outras credenciais, use os parâmetros - ProfileName ou - AccessKey /-SecretKey .

```
Get-STSSessionToken
```

Saída:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Exemplo 2: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por uma hora. As credenciais usadas para fazer a solicitação são obtidas do perfil especificado.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Saída:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Exemplo 3: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por uma hora usando o número de identificação do dispositivo de MFA associado à conta cujas credenciais estão especificadas no perfil 'myprofilename' e o valor fornecido pelo dispositivo.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Saída:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- Para obter detalhes da API, consulte GetSessionTokenem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um token de sessão passando um token de MFA e use-o para listar os buckets do Amazon S3 para a conta.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- Para obter detalhes da API, consulte a GetSessionTokenReferência da API AWS SDK for Python (Boto3).

Cenários para AWS STS usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS STS com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS STS ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Assuma uma função do IAM que exija um token de MFA com AWS STS usando um AWS SDK
- Configurar a conectividade do serviço de contêiner
- Construa um URL com AWS STS para usuários federados usando um AWS SDK
- Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor
- Criar uma API REST com integração de proxy de função
- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Como criar um espaço de trabalho de monitoramento gerenciado
- Como criar um painel de monitoramento com o nome da função como uma variável
- Obtenha um token de sessão que exija um token de MFA com AWS STS usando um AWS SDK
- Conceitos básicos do gerenciamento de configurações
- Conceitos básicos de registros de contêineres
- Conceitos básicos de mensagens da Internet das Coisas
- Conceitos básicos de arquivos de atributos de machine learning
- Conceitos básicos de clusters gerenciados de kubernetes
- Conceitos básicos de streaming gerenciado
- Conceitos básicos das notificações push
- Conceitos básicos dos mecanismos de pesquisa e analytics
- Conceitos básicos de firewalls do aplicativo Web
- Como mover segredos codificados para o armazenamento seguro de segredos
- Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas
- Como configurar o gerenciamento de sistemas

Assuma uma função do IAM que exija um token de MFA com AWS STS usando um AWS SDK

O exemplo de código a seguir mostra como assumir um perfil que exige um token de MFA.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Criar um perfil do IAM que conceda permissão para listar os buckets do Amazon S3.
- Criar um usuário do IAM que tenha permissão para assumir o perfil somente quando as credenciais de MFA forem fornecidas.
- Registrar um dispositivo MFA para o usuário.
- Assumir o perfil e usar credenciais temporárias para listar os buckets do S3.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário do IAM, registre um dispositivo de MFA e crie um perfil que conceda permissão para listar os buckets do Amazon S3. O usuário só tem direitos para assumir a função.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
```

```
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the
role,
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                         that has permissions to create users, roles, and
policies
                         in the account.
    :return: The newly created user, user key, virtual MFA device, and role.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
```

```
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": user.arn},
                        "Action": "sts:AssumeRole",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
    print(f"Created role {role.name} that requires MFA.")

    policy = iam_resource.create_policy(
        PolicyName=unique_name("policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                    }
                ],
            }
```

```
        ),
    )
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "sts:AssumeRole",
                        "Resource": role.arn,
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user assume "
        f"the role."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device, role
```

Mostre que não é permitido assumir uma função sem um token de MFA.

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials
 results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
```

```
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
 RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Got AccessDenied.")
        else:
            raise
```

Assuma o perfil que concede permissão para listar os buckets do S3 passando o token de MFA necessário e mostre que os buckets podem ser listados.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
```

```
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Destrua os recursos criados para a demonstração.

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
```

```
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Execute esse cenário usando a funções definidas anteriormente.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
        )
        try_to_assume_role_without_mfa(role.arn, "demo-sts-session", sts_client)
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_from_assumed_role_with_mfa(
            role.arn,
            "demo-sts-session",
            virtual_mfa_device.serial_number,
            mfa_totp,
            sts_client,
        )
    finally:
        teardown(user, virtual_mfa_device, role)
        print("Thanks for watching!")
```

- Para obter detalhes da API, consulte a AssumeRoleReferência da API AWS SDK for Python (Boto3).

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
```

```
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"
```

```
    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
```

```
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
```

```
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
```

```
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
```

```
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
```

```
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
```

```
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF
```

```
    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
```

```
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}
```

```
# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
```

```
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
```

```
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources
```

```
    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace

- DeleteRole
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeClusters
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Construa um URL com AWS STS para usuários federados usando um AWS SDK

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM que conceda acesso somente leitura aos recursos do Amazon S3 da conta atual.
- Obtenha um token de segurança do endpoint da AWS federação.
- Criar um URL que possa ser usado para acessar o console com credenciais federadas.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um perfil que conceda acesso somente leitura aos recursos do S3 da conta atual.

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 instance
                         that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": iam_resource.CurrentUser().arn},
                        "Action": "sts:AssumeRole",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn="arn:aws:iam::aws:policy/
AmazonS3ReadOnlyAccess")
    print(f"Created role {role.name}.")

    print("Give AWS time to propagate these new resources and connections.",
 end="")
```

```
    progress_bar(10)

    return role
```

Obtenha um token de segurança do endpoint da AWS federação e crie uma URL que possa ser usada para acessar o console com credenciais federadas.

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS
 Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS)
 that
       can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
       AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS
 federation
       endpoint, includes the sign-in token for authentication, and redirects to
       the AWS Management Console with permissions defined by the role that was
       specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are
 granted.
                            The current user must have permission to assume the
 role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
```

```
        "sessionId": temp_credentials["AccessKeyId"],
        "sessionKey": temp_credentials["SecretAccessKey"],
        "sessionToken": temp_credentials["SessionToken"],
    }
    aws_federated_signin_endpoint = "https://signin.aws.amazon.com/federation"

    # Make a request to the AWS federation endpoint to get a sign-in token.
    # The requests.get function URL-encodes the parameters and builds the query
 string
    # before making the request.
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            "Action": "getSigninToken",
            "SessionDuration": str(datetime.timedelta(hours=12).seconds),
            "Session": json.dumps(session_data),
        },
    )
    signin_token = json.loads(response.text)
    print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

    # Make a federated URL that can be used to sign into the AWS Management
 Console.
    query_string = urllib.parse.urlencode(
        {
            "Action": "login",
            "Issuer": issuer,
            "Destination": "https://console.aws.amazon.com/",
            "SigninToken": signin_token["SigninToken"],
        }
    )
    federated_url = f"{aws_federated_signin_endpoint}?{query_string}"
    return federated_url
```

Destrua os recursos criados para a demonstração.

```
def teardown(role):
    """
    Removes all resources created during setup.
```

```
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

Execute esse cenário usando a funções definidas anteriormente.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    role = setup(iam_resource)
    sts_client = boto3.client("sts")
    try:
        federated_url = construct_federated_url(
            role.arn, "AssumeRoleDemoSession", "example.org", sts_client
        )
        print(
            "Constructed a federated URL that can be used to connect to the "
            "AWS Management Console with role-defined permissions:"
        )
        print("-" * 88)
        print(federated_url)
        print("-" * 88)
        _ = input(
            "Copy and paste the above URL into a browser to open the AWS "
            "Management Console with limited permissions. When done, press "
            "Enter to clean up and complete this demo."
        )
    finally:
        teardown(role)
        print("Thanks for watching!")
```

- Para obter detalhes da API, consulte a AssumeRoleReferência da API AWS SDK for Python (Boto3).

Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um cluster.
- Criar uma definição de tarefa.
- Criar o serviço.
- Fazer a limpeza.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"
```

```
# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "==========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "==========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
```

```
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
```

```
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
```

```
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."
```

```
        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
'$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"
```

```
            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "============================================"
        echo "CLEANUP COMPLETED"
        echo "============================================"
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
```

```
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"
```

```
CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
```

```
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
```

```
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
```

```
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"
```

```
echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
```

```
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole

- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeNetworkInterfaces
- DescribeSecurityGroups
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Criar uma API REST com integração de proxy de função

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para execução do Lambda.
- Criar e implantar uma função do Lambda.
- Criar uma API REST
- Configurar a integração de proxy do Lambda.
- Implantar e testar a API
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
```

```
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
```

```
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
```

```
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"
```

```
# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2
```

```
aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()
```

```
# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi
```

```
        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
 "$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
 $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
 2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
```

```
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
```

```
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
 file for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"
```

```
    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
```

```
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
```

```
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
```

```
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."
```

```
    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
```

```
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
```

```
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"
```

```
    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
```

```
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
```

```
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
```

```
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetParameters
  - GetRole
  - ListContainerInstances
  - ListServices
  - ListTasks
  - RegisterTaskDefinition
  - RunInstances

- StopTask
- TerminateInstances
- UpdateService
- Wait

Como criar um espaço de trabalho de monitoramento gerenciado

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para seu espaço de trabalho.
- Criar um espaço de trabalho do Grafana.
- Configurar autenticação
- Definir configurações opcionais.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
```

```
# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}
```

```
# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"
```

```
# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"
```

```
# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-
grafana-gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id
 "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -
f4)
```

```
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML
 settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
```

```
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 |
 grep -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
```

```
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Como criar um painel de monitoramento com o nome da função como uma variável

O exemplo de código a seguir mostra como:

- Crie um CloudWatch painel
- Adicionar widgets de métricas do Lambda com uma variável de nome de função.
- Verificar o painel.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
```

```
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
```

```
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

```
    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
```

```
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
```

```
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
```

```
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
```

```
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
```

```
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
```

```
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AttachRolePolicy
- CreateFunction
- CreateRole
- DeleteDashboards
- DeleteFunction
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetDashboard
- Invoke
- ListDashboards
- ListFunctions
- PutDashboard

Obtenha um token de sessão que exija um token de MFA com AWS STS usando um AWS SDK

O exemplo de código a seguir mostra como obter um token de sessão que exige um token de MFA.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Criar um perfil do IAM que conceda permissão para listar os buckets do Amazon S3.
- Criar um usuário do IAM que tenha permissão para assumir o perfil somente quando as credenciais de MFA forem fornecidas.
- Registrar um dispositivo MFA para o usuário.
- Forneça credenciais de MFA para obter um token de sessão e use credenciais temporárias para listar os buckets do S3.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário do IAM, registre um dispositivo de MFA e crie um perfil que conceda permissão para deixar o usuário listar os buckets do S3 somente quando credenciais de MFA forem usadas.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3
 buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, MFA devices, and
                         policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
```

```
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
```

```
    print(
        f"Created an inline policy for {user.name} that lets the user list
 buckets, "
        f"but only when MFA credentials are present."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device
```

Obtenha credenciais de sessão temporárias passando um token de MFA e use-as para listar os buckets do S3 para a conta.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
```

```
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Destrua os recursos criados para a demonstração.

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Execute esse cenário usando a funções definidas anteriormente.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
```

```
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
        )
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response["Error"]["Code"] == "AccessDenied":
                print("Got expected AccessDenied error.")
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_with_session_token_with_mfa(
            virtual_mfa_device.serial_number, mfa_totp, sts_client
        )
    finally:
        teardown(user, virtual_mfa_device)
        print("Thanks for watching!")
```

- Para obter detalhes da API, consulte a GetSessionTokenReferência da API AWS SDK for Python (Boto3).

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.

- Verificar a configuração do Config

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
```

```
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
```

```
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
```

```
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
```

```
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
```

```
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
```

```
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Conceitos básicos de registros de contêineres

O exemplo de código a seguir mostra como:

- Criar uma imagem do Docker.
- Criar um repositório do Amazon ECR.
- Excluir recursos.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "====================================================="

# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
```

```
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-
cliv2.html for installation instructions."
    exit 1
fi

# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure'
 to set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running
 this script."
    echo "Visit https://docs.docker.com/get-docker/ for installation
 instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try
 again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"

    echo "====================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
```

```
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "==================================================="

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "==================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force ||
 echo "Failed to delete repository, it may not exist or may have already been
 deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local
 image, it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove
 tagged image, it may not exist or may have already been deleted."
    fi

    echo "Cleanup completed."
    echo "==================================================="
}
```

```
# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build
 Docker image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list
 Docker images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"

# Get AWS account ID
echo "Getting AWS account ID..."
```

```
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials
 are configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"

# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --
tags key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-
registry-gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username
 AWS --password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" ||
 handle_error "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"

# Tag the image
echo "Tagging Docker image..."
```

```
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image
 to ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged
 image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "====================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "====================================================="

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"
```

```
    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]];
 then
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-
ids imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "==================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- BatchDeleteImage
- CreateRepository
- DeleteRepository
- GetCallerIdentity
- GetLoginPassword

Conceitos básicos de mensagens da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar recursos de IoT
- Configurar o dispositivo
- Executar os aplicativos de exemplo
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
```

```
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
}

# Function to check for errors
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 $LOG_FILE
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID
 --new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
```

```
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
}

# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "==================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a
 $LOG_FILE
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
    echo "AWS region not configured. Please run 'aws configure' to set your
 region." | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE
```

```
# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE

# Create IoT policy
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
```

```
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-
document file://iot-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

# Create IoT thing
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn'
 --output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?

# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active
 --certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile
 $CERTS_DIR/public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
```

```
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
check_error $?

# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." |
 tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
```

```
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-
python-v2.git"
    check_error $?
else
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
fi

# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE

# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
```

```
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE

    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
```

```
    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"

    # Delete policies
    echo "Deleting policies..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE" | tee -a $LOG_FILE
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachPolicy
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreatePolicy
  - CreateThing

- DeleteCertificate
- DeletePolicy
- DeleteThing
- DescribeEndpoint
- DetachPolicy
- DetachThingPrincipal
- GetCallerIdentity
- UpdateCertificate

Conceitos básicos de arquivos de atributos de machine learning

O exemplo de código a seguir mostra como:

- Configurar permissões do IAM.
- Crie uma função SageMaker de execução
- Criar grupo de atributos.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
```

```
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"
```

```
    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
```

```
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
```

```
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
```

```
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."
```

```
# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
```

```
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3:////'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
```

```
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
```

```
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi
```

```
echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
```

```
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
```

```
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Conceitos básicos de clusters gerenciados de kubernetes

O exemplo de código a seguir mostra como:

- Criar uma VPC para um cluster do EKS.
- Criar perfis do IAM para um cluster do EKS.
- Criação do seu cluster de EKS
- Configurar o kubectl para se comunicar com o cluster.
- Crie um grupo de nós gerenciados
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
```

```
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
```

```
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
```

```
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."
```

```
# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
```

```
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
```

```
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"
```

```
# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
```

```
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
```

```
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
```

```
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "==========================================="

# Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNodegroups
  - UpdateKubeconfig
  - Wait

Conceitos básicos de streaming gerenciado

O exemplo de código a seguir mostra como:

- Criar um cluster do Amazon MSK.
- Criar permissões do IAM para acessar o MSK.
- Criar uma máquina cliente
- Obter os agentes de bootstrap.
- Configurar a máquina cliente.
- Crie um tópico e produce/consume dados

- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
```

```
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
```

```
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
```

```
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
 to terminate instance"
        echo "Waiting for instance to terminate..."
```

```
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
"Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
"$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are
removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
```

```
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
 profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
 "$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
 delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
 "$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
```

```
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}
```

```
# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"
```

```
if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)
```

```
if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
```

```
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
```

```
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10
```

```
echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
```

```
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
```

```
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
```

```
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi
```

```
echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
```

```
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
```

```
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
```

```
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "================================================"

# Create a script to set up environment variables
```

```
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
```

```
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "================================================"
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
```

```
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "==============================================="
echo ""

# Ask if user wants to clean up resources
```

```
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair
  - DeletePolicy
  - DeleteRole
  - DeleteSecurityGroup
  - DescribeAvailabilityZones

- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Conceitos básicos das notificações push

O exemplo de código a seguir mostra como:

- Criar uma aplicação.
- Habilitar canais de notificações por push.
- Enviar notificações por push
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail

# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
```

```
cleanup() {
    local exit_code=$?
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
        echo "ERROR: Command failed: $cmd" >&2
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
```

```
}

# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi

    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
    echo ""

    return 0
}

# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
```

```
        echo "jq is not available, using grep for JSON parsing"
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."

    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
        echo "  - mobiletargeting:CreateApp" >&2
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
```

```
    local app_name=$1

    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters"
 >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi

check_json_tools
validate_permissions

# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
```

```
fi

echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi

echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "============================================"
echo "ENABLING FCM (GCM) CHANNEL"
echo "============================================"
echo "Note: This is using a placeholder API key for demonstration purposes only."
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
```

```
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-
demo-only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server
 Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real
 certificates)
echo ""
echo "============================================"
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================"
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --
apns-channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
    --application-id "$APP_ID" \
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
```

```
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-
channel command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "============================================"
echo "CREATING MESSAGE FILES"
echo "============================================"

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
        "key1": "value1",
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
```

```
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF

echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
echo "To send real messages, you would need to replace the sample device tokens
 with actual ones."

# Step 5: Demonstrate how to send messages (this will fail with placeholder
 tokens)
echo ""
echo "============================================"
echo "DEMONSTRATING MESSAGE SENDING"
echo "============================================"
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
```

```
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
echo "Retrieving application details..."

echo "Executing: aws pinpoint get-app --application-id $APP_ID"
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
```

```
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
echo "2. Attempting to enable FCM and APNS channels (with placeholder
 credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
echo "5. Retrieving application details"
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
```

```
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps
  - GetCallerIdentity
  - SendMessages
  - UpdateApnsChannel
  - UpdateGcmChannel

Conceitos básicos dos mecanismos de pesquisa e analytics

O exemplo de código a seguir mostra como:

- Criar um domínio OpenSearch de serviço
- Carregar dados em seu domínio.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md

# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Track if domain was successfully created
DOMAIN_CREATED=false
```

```
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [[ "$DOMAIN_CREATED" == "true" ]]; then
        echo "Checking if domain $DOMAIN_NAME exists before attempting to
 delete..."

        # Check if domain exists before trying to delete
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/
null 2>&1; then
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
}
```

```
# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
```

```
# This policy restricts access to specific actions and should be further
 restricted to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for
 completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack
 creation failed"

DOMAIN_CREATED=true
echo "Domain created successfully via CloudFormation."

# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120

# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
```

```
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo
 "No"; fi)"

# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel",
 "Malahide, Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas",
 "Stellate, Nick", "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese",
 "Spielbauer, Bruce", "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett,
 Charlie"], "title": "U.S. Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955,
 "actor": ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal",
 "Backus, Jim", "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen,
```

```
 Corey", "Birch, Paul", "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh,
 Nelson", "Williams, Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira",
 "McMahon, David", "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"

    # Test 1: Basic authentication with root endpoint
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)

    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }
```

```
    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

        if is_success_code "$HEALTH_HTTP_CODE"; then
            echo "# Cluster health authentication test successful! (HTTP
 $HEALTH_HTTP_CODE)"
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
            echo "Cluster health authentication also failed with HTTP code:
 $HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
 then
                echo "Error: Request is being treated as anonymous
 (authentication not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential
 backoff..."
```

```
            # Retry authentication test with exponential backoff
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT
+1))/$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

                echo "Retry result (HTTP $HTTP_CODE):"
                echo "$AUTH_TEST_RESULT"

                if is_success_code "$HTTP_CODE"; then
                    echo "# Authentication test successful after retry! (HTTP
 $HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:"
 | cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
 (HTTP $HEALTH_HTTP_CODE)"
```

```
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
 minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding
 with data operations."

        # Upload single document (matches tutorial exactly)
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
 '"result"'; then
            echo "# Single document uploaded successfully! (HTTP
 $UPLOAD_HTTP_CODE)"
        else
            echo "# Warning: Single document upload may have failed (HTTP
 $UPLOAD_HTTP_CODE)"
        fi
```

```
        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
 '"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
 $BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
        echo "Waiting for documents to be indexed..."
        sleep 5

        # Step 3: Search Documents (matches tutorial exactly)
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
```

```
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
 '"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."

        # Debug domain configuration
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-
name "$DOMAIN_NAME" --query 'DomainStatus.{AdvancedSecurityOptions:
 AdvancedSecurityOptions, AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
```

```
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user
 works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi

# Display OpenSearch Dashboards URL (matches tutorial)
echo ""
echo "============================================"
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================"
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
```

```
echo "============================================"
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
echo "Cleaning up temporary files..."
```

```
# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/
_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - GetCallerIdentity

Conceitos básicos de firewalls do aplicativo Web

O exemplo de código a seguir mostra como:

- Criar uma ACL da web.
- Adicionar uma regra de correspondência de string.
- Adicionar regras gerenciadas.
- Configurar o registro em log da
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
```

```
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
```

```
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken'
 2>/dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the
 AWS Console."
```

```
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi
```

```
# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "====================================================="
echo "STEP 2: Adding String Match Rule"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
```

```
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
```

```
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
```

```
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
```

```
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
```

```
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "====================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "====================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "====================================================="
```

```
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "====================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo
 "$first_dist" | grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution
 successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
```

```
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

Como mover segredos codificados para o armazenamento seguro de segredos

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.

- Criar um segredo no Secrets Manager.
- Atualizar o código de uma aplicação.
- Atualizar o segredo.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
```

```
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-
delete-without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
```

```
        for policy in $(aws iam list-role-policies --role-name
 "$RUNTIME_ROLE_NAME" --query 'PolicyNames[]' --output text 2>/dev/null || true);
 do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""
```

```
# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
```

```
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_u
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
```

```
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/
null || true
```

```
# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').tok

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
```

```
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole

- CreateSecret
- DeleteRole
- DeleteSecret
- DetachRolePolicy
- GetCallerIdentity
- GetSecretValue
- PutResourcePolicy
- UpdateSecret

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial
```

```
#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
```

```
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}
```

```
# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
```

```
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
```

```
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
```

```
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"
```

```
    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
```

```
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
```

```
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done
```

```
if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
```

```
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
```

```
   echo "Deleting experiment template $TEMPLATE_ID..."
   DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
"$TEMPLATE_ID")
   check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

   # Delete CloudWatch alarm
   echo "Deleting CloudWatch alarm $ALARM_NAME..."
   DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
"$ALARM_NAME")
   check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

   # Terminate EC2 instance
   echo "Terminating EC2 instance $INSTANCE_ID..."
   TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
   check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
   echo "Waiting for instance to terminate..."
   aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

   # Clean up IAM resources
   echo "Removing role from instance profile..."
   REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
     --instance-profile-name "$INSTANCE_PROFILE_NAME" \
     --role-name "$EC2_ROLE_NAME")
   check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

   echo "Deleting instance profile..."
   DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
     --instance-profile-name "$INSTANCE_PROFILE_NAME")
   check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

   echo "Deleting FIS role policy..."
   DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
     --role-name "$FIS_ROLE_NAME" \
     --policy-name "$FIS_POLICY_NAME")
   check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

   echo "Detaching policy from EC2 role..."
   DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
     --role-name "$EC2_ROLE_NAME" \
     --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
   check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

   echo "Deleting FIS role..."
   DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
```

```
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy

- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Como configurar o gerenciamento de sistemas

O exemplo de código a seguir mostra como:

- Criar permissões do IAM para o Systems Manager.
- Criar um perfil do IAM para o Systems Manager
- Configurar o Systems Manager.
- Verificar a configuração.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
```

```
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
```

```
                    log_cmd "aws iam detach-role-policy --role-name $resource_id
 --policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --
manager-arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/
null || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and
 region."
echo "It will create IAM policies and roles, then enable Systems Manager
 features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi
```

```
echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
         "ssm:GetPatchBaseline",
         "ssm:DescribeMaintenanceWindows",
         "ssm:DescribeMaintenanceWindowTasks"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmDocument",
       "Effect": "Allow",
       "Action": [
         "ssm:GetDocument",
         "ssm:DescribeDocument"
       ],
       "Resource": [
```

```
            "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
            "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
          ]
        },
        {
          "Sid": "SsmEnableExplorer",
          "Effect": "Allow",
          "Action": "ssm:StartAutomationExecution",
          "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
        },
        {
          "Sid": "SsmExplorerRds",
          "Effect": "Allow",
          "Action": [
            "ssm:GetOpsSummary",
            "ssm:CreateResourceDataSync",
            "ssm:UpdateResourceDataSync"
          ],
          "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
        },
        {
          "Sid": "OrgsReadOnly",
          "Effect": "Allow",
          "Action": [
            "organizations:DescribeAccount",
            "organizations:DescribeOrganization",
            "organizations:ListDelegatedAdministrators",
            "organizations:ListRoots",
            "organizations:ListParents",
            "organizations:ListOrganizationalUnitsForParent",
            "organizations:DescribeOrganizationalUnit",
            "organizations:ListAWSServiceAccessForOrganization"
          ],
          "Resource": "*"
        },
        {
          "Sid": "OrgsAdministration",
          "Effect": "Allow",
          "Action": [
            "organizations:EnableAWSServiceAccess",
            "organizations:RegisterDelegatedAdministrator",
            "organizations:DeregisterDelegatedAdministrator"
          ],
          "Resource": "*",
```

```
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
```

```
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
```

```
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
```

```
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
      }
    ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
```

```
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
```

```
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"
```

```
# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --
policy-arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"
```

```
# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few
 minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-
arn $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration
 manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later." | tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetConfigurationManager

Exemplos de código para Suporte usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Support com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Suporte Guia do usuário — Mais informações sobre Suporte.
- Suporte Referência da API — Detalhes sobre todas as Suporte ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para Suporte usar AWS SDKs
  - Hello Suporte
  - Aprenda as noções básicas do Suporte com um AWS SDK
  - Ações para Suporte usar AWS SDKs
    - Use AddAttachmentsToSetcom um AWS SDK ou CLI
    - Use AddCommunicationToCasecom um AWS SDK ou CLI
    - Use CreateCasecom um AWS SDK ou CLI
    - Use DescribeAttachmentcom um AWS SDK ou CLI
    - Use DescribeCasescom um AWS SDK ou CLI
    - Use DescribeCommunicationscom um AWS SDK ou CLI
    - Use DescribeServicescom um AWS SDK ou CLI
    - Use DescribeSeverityLevelscom um AWS SDK ou CLI
    - Usar DescribeTrustedAdvisorCheckRefreshStatuses com uma CLI
    - Usar DescribeTrustedAdvisorCheckResult com uma CLI

    - Usar DescribeTrustedAdvisorCheckSummaries com uma CLI
    - Usar DescribeTrustedAdvisorChecks com uma CLI
    - Usar RefreshTrustedAdvisorCheck com uma CLI
    - Use ResolveCasecom um AWS SDK ou CLI
- Cenários para Suporte usar AWS SDKs
  - Introdução ao suporte técnico

Exemplos básicos para Suporte usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Support com AWS SDKs.

Exemplos

- Hello Suporte
- Aprenda as noções básicas do Suporte com um AWS SDK
- Ações para Suporte usar AWS SDKs
  - Use AddAttachmentsToSetcom um AWS SDK ou CLI
  - Use AddCommunicationToCasecom um AWS SDK ou CLI
  - Use CreateCasecom um AWS SDK ou CLI
  - Use DescribeAttachmentcom um AWS SDK ou CLI
  - Use DescribeCasescom um AWS SDK ou CLI
  - Use DescribeCommunicationscom um AWS SDK ou CLI
  - Use DescribeServicescom um AWS SDK ou CLI
  - Use DescribeSeverityLevelscom um AWS SDK ou CLI
  - Usar DescribeTrustedAdvisorCheckRefreshStatuses com uma CLI
  - Usar DescribeTrustedAdvisorCheckResult com uma CLI
  - Usar DescribeTrustedAdvisorCheckSummaries com uma CLI
  - Usar DescribeTrustedAdvisorChecks com uma CLI
  - Usar RefreshTrustedAdvisorCheck com uma CLI
  - Use ResolveCasecom um AWS SDK ou CLI

Hello Suporte

O exemplo de código a seguir mostra como começar a usar o Suporte.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.AWSSupport;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

public static class HelloSupport
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        // You must have one of the following AWS Support plans: Business,
 Enterprise On-Ramp, or Enterprise. Otherwise, an exception will be thrown.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonAWSSupport>()
            ).Build();

        // Now the client is available for injection.
        var supportClient =
 host.Services.GetRequiredService<IAmazonAWSSupport>();

        // You can use await and any of the async methods to get a response.
        var response = await supportClient.DescribeServicesAsync();
        Console.WriteLine($"\tHello AWS Support! There are
 {response.Services.Count} services available.");
```

```
    }
}
```

- Para obter detalhes da API, consulte DescribeServicesa Referência AWS SDK para .NET da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
```

```
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
                .region(region)
                .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
 DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
```

```
                index++;
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeServicesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Invoque `main()` para executar o exemplo.

```
import {
  DescribeServicesCommand,
  SupportClient,
} from "@aws-sdk/client-support";

// Change the value of 'region' to your preferred AWS Region.
const client = new SupportClient({ region: "us-east-1" });

const getServiceCount = async () => {
  try {
    const { services } = await client.send(new DescribeServicesCommand({}));
    return services.length;
  } catch (err) {
    if (err.name === "SubscriptionRequiredException") {
      throw new Error(
        "You must be subscribed to the AWS Support plan to use this feature.",
```

```
      );
    }
    throw err;
  }
};

export const main = async () => {
  try {
    const count = await getServiceCount();
    console.log(`Hello, AWS Support! There are ${count} services available.`);
  } catch (err) {
    console.error("Failed to get service count: ", err.message);
  }
};
```

- Para obter detalhes da API, consulte DescribeServicesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you must have the AWS Business Support Plan to use the AWS Support
 Java API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/
```

```
This Kotlin example performs the following task:

1. Gets and displays available services.
 */

suspend fun main() {
    displaySomeServices()
}

// Return a List that contains a Service name and Category name.
suspend fun displaySomeServices() {
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is: " + service.name)

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                index++
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeServicesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def hello_support(support_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Support client and count
    the available services in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param support_client: A Boto3 Support Client object.
    """
    try:
        print("Hello, AWS Support! Let's count the available Support services:")
        response = support_client.describe_services()
        print(f"There are {len(response['services'])} services available.")
    except ClientError as err:
        if err.response["Error"]["Code"] == "SubscriptionRequiredException":
            logger.info(
                "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
                "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                "examples."
            )
        else:
            logger.error(
                "Couldn't count services. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


if __name__ == "__main__":
    hello_support(boto3.client("support"))
```

- Para obter detalhes da API, consulte a DescribeServicesReferência da API AWS SDK for Python (Boto3).

Aprenda as noções básicas do Suporte com um AWS SDK

Os exemplos de código a seguir mostram como:

- Obtenha e exiba os serviços disponíveis e os níveis de gravidade dos casos.
- Criar um caso de suporte usando um serviço, uma categoria e um nível de gravidade selecionados.
- Obter e exibir uma lista de casos em aberto para o dia atual.
- Adicionar um conjunto de anexos e uma comunicação ao novo caso.
- Descrever o novo anexo e a comunicação para o caso.
- Resolver o caso.
- Obtenha e exiba uma lista de casos resolvidos para o dia atual.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
/// <summary>
/// Hello AWS Support example.
/// </summary>
public static class SupportCaseScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    To use the AWS Support API, you must have one of the following AWS Support
 plans: Business, Enterprise On-Ramp, or Enterprise.

    This .NET example performs the following tasks:
    1.  Get and display services. Select a service from the list.
    2.  Select a category from the selected service.
    3.  Get and display severity levels and select a severity level from the
 list.
    4.  Create a support case using the selected service, category, and severity
 level.
    5.  Get and display a list of open support cases for the current day.
    6.  Create an attachment set with a sample text file to add to the case.
    7.  Add a communication with the attachment to the support case.
    8.  List the communications of the support case.
    9.  Describe the attachment set.
    10. Resolve the support case.
    11. Get a list of resolved cases for the current day.
   */

    private static SupportWrapper _supportWrapper = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
```

```
                services.AddAWSService<IAmazonAWSSupport>(new AWSOptions()
{ Profile = "default" })
                    .AddTransient<SupportWrapper>()
            )
            .Build();

        var logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger(typeof(SupportCaseScenario));

        _supportWrapper = host.Services.GetRequiredService<SupportWrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS Support case example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            var apiSupported = await _supportWrapper.VerifySubscription();
            if (!apiSupported)
            {
                logger.LogError("You must have a Business, Enterprise On-Ramp, or
Enterprise Support " +
                                "plan to use the AWS Support API. \n\tPlease
upgrade your subscription to run these examples.");
                return;
            }

            var service = await DisplayAndSelectServices();

            var category = DisplayAndSelectCategories(service);

            var severityLevel = await DisplayAndSelectSeverity();

            var caseId = await CreateSupportCase(service, category,
severityLevel);

            await DescribeTodayOpenCases();

            var attachmentSetId = await CreateAttachmentSet();

            await AddCommunicationToCase(attachmentSetId, caseId);
```

```
            var attachmentId = await ListCommunicationsForCase(caseId);

            await DescribeCaseAttachment(attachmentId);

            await ResolveCase(caseId);

            await DescribeTodayResolvedCases();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("AWS Support case example scenario complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// List some available services from AWS Support, and select a service for
 the example.
    /// </summary>
    /// <returns>The selected service.</returns>
    private static async Task<Service> DisplayAndSelectServices()
    {
        Console.WriteLine(new string('-', 80));
        var services = await _supportWrapper.DescribeServices();
        Console.WriteLine($"AWS Support client returned {services.Count}
 services.");

        Console.WriteLine($"1. Displaying first 10 services:");
        for (int i = 0; i < 10 && i < services.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {services[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > services.Count)
        {
            Console.WriteLine(
                "Select an example support service by entering a number from the
 preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
```

```
        }
        Console.WriteLine(new string('-', 80));

        return services[choiceNumber - 1];
    }

    /// <summary>
    /// List the available categories for a service and select a category for the
example.
    /// </summary>
    /// <param name="service">Service to use for displaying categories.</param>
    /// <returns>The selected category.</returns>
    private static Category DisplayAndSelectCategories(Service service)
    {
        Console.WriteLine(new string('-', 80));

        Console.WriteLine($"2. Available support categories for Service
\"{service.Name}\":");
        for (int i = 0; i < service.Categories.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {service.Categories[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > service.Categories.Count)
        {
            Console.WriteLine(
                "Select an example support category by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        Console.WriteLine(new string('-', 80));

        return service.Categories[choiceNumber - 1];
    }

    /// <summary>
    /// List available severity levels from AWS Support, and select a level for
the example.
    /// </summary>
    /// <returns>The selected severity level.</returns>
    private static async Task<SeverityLevel> DisplayAndSelectSeverity()
```

```
    {
        Console.WriteLine(new string('-', 80));
        var severityLevels = await _supportWrapper.DescribeSeverityLevels();

        Console.WriteLine($"3. Get and display available severity levels:");
        for (int i = 0; i < 10 && i < severityLevels.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {severityLevels[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > severityLevels.Count)
        {
            Console.WriteLine(
                "Select an example severity level by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        Console.WriteLine(new string('-', 80));

        return severityLevels[choiceNumber - 1];
    }

    /// <summary>
    /// Create an example support case.
    /// </summary>
    /// <param name="service">Service to use for the new case.</param>
    /// <param name="category">Category to use for the new case.</param>
    /// <param name="severity">Severity to use for the new case.</param>
    /// <returns>The caseId of the new support case.</returns>
    private static async Task<string> CreateSupportCase(Service service,
        Category category, SeverityLevel severity)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Create an example support case" +
                          $" with the following settings:" +
                          $" \n\tService: {service.Name}, Category:
{category.Name} " +
                          $"and Severity Level: {severity.Name}.");
        var caseId = await _supportWrapper.CreateCase(service.Code,
category.Code, severity.Code,
            "Example case for testing, ignore.", "This is my example support
case.");
```

```
        Console.WriteLine($"\tNew case created with ID {caseId}");

        Console.WriteLine(new string('-', 80));

        return caseId;
    }

    /// <summary>
    /// List open cases for the current day.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeTodayOpenCases()
    {
        Console.WriteLine($"5. List the open support cases for the current
day.");
        // Describe the cases. If it is empty, try again and allow time for the
new case to appear.
        List<CaseDetails> currentOpenCases = null!;
        while (currentOpenCases == null || currentOpenCases.Count == 0)
        {
            Thread.Sleep(1000);
            currentOpenCases = await _supportWrapper.DescribeCases(
                new List<string>(),
                null,
                false,
                false,
                DateTime.UtcNow.Date,
                DateTime.UtcNow);
        }

        foreach (var openCase in currentOpenCases)
        {
            Console.WriteLine($"\tCase: {openCase.CaseId} created
{openCase.TimeCreated}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an attachment set for a support case.
    /// </summary>
    /// <returns>The attachment set id.</returns>
```

```
    private static async Task<string> CreateAttachmentSet()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Create an attachment set for a support case.");
        var fileName = "example_attachment.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for attachment to a support case.");
        }

        await using var ms = new MemoryStream(await
 File.ReadAllBytesAsync(fileName));

        var attachmentSetId = await _supportWrapper.AddAttachmentToSet(
            ms,
            fileName);

        Console.WriteLine($"\tNew attachment set created with id: \n
\t{attachmentSetId.Substring(0, 65)}...");

        Console.WriteLine(new string('-', 80));

        return attachmentSetId;
    }

    /// <summary>
    /// Add an attachment set and communication to a case.
    /// </summary>
    /// <param name="attachmentSetId">Id of the attachment set.</param>
    /// <param name="caseId">Id of the case to receive the attachment set.</
param>
    /// <returns>Async task.</returns>
    private static async Task AddCommunicationToCase(string attachmentSetId,
 string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Add attachment set and communication to
 {caseId}.");

        await _supportWrapper.AddCommunicationToCase(
```

```
            caseId,
            "This is an example communication added to a support case.",
            attachmentSetId);

        Console.WriteLine($"\tNew attachment set and communication added to
{caseId}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the communications for a case.
    /// </summary>
    /// <param name="caseId">Id of the case to describe.</param>
    /// <returns>An attachment id.</returns>
    private static async Task<string> ListCommunicationsForCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. List communications for case {caseId}.");

        var communications = await
_supportWrapper.DescribeCommunications(caseId);
        var attachmentId = "";
        foreach (var communication in communications)
        {
            Console.WriteLine(
                $"\tCommunication created on: {communication.TimeCreated} has
{communication.AttachmentSet.Count} attachments.");
            if (communication.AttachmentSet.Any())
            {
                attachmentId = communication.AttachmentSet.First().AttachmentId;
            }
        }

        Console.WriteLine(new string('-', 80));
        return attachmentId;
    }

    /// <summary>
    /// Describe an attachment by id.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment to describe.</param>
    /// <returns>Async task.</returns>
    private static async Task DescribeCaseAttachment(string attachmentId)
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Describe the attachment set.");

        var attachment = await _supportWrapper.DescribeAttachment(attachmentId);
        var data = Encoding.ASCII.GetString(attachment.Data.ToArray());
        Console.WriteLine($"\tAttachment includes {attachment.FileName} with
data: \n\t{data}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Resolve the support case.
    /// </summary>
    /// <param name="caseId">Id of the case to resolve.</param>
    /// <returns>Async task.</returns>
    private static async Task ResolveCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Resolve case {caseId}.");

        var status = await _supportWrapper.ResolveCase(caseId);
        Console.WriteLine($"\tCase {caseId} has final status {status}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List resolved cases for the current day.
    /// </summary>
    /// <returns>Async Task.</returns>
    private static async Task DescribeTodayResolvedCases()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. List the resolved support cases for the current
day.");
        var currentCases = await _supportWrapper.DescribeCases(
            new List<string>(),
            null,
            false,
            true,
            DateTime.UtcNow.Date,
            DateTime.UtcNow);
```

```
        foreach (var currentCase in currentCases)
        {
            if (currentCase.Status == "resolved")
            {
                Console.WriteLine(
                    $"\tCase: {currentCase.CaseId}: status
 {currentCase.Status}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }
}
```

Métodos de embalagem usados pelo cenário para Suporte ações.

```
/// <summary>
/// Wrapper methods to use AWS Support for working with support cases.
/// </summary>
public class SupportWrapper
{
    private readonly IAmazonAWSSupport _amazonSupport;
    public SupportWrapper(IAmazonAWSSupport amazonSupport)
    {
        _amazonSupport = amazonSupport;
    }


    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
 (“ko”) are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
            {
```

```
                Language = language
            });
        return response.Services;
    }



    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
    /// <returns>The list of support severity levels.</returns>
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language
= "en")
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
            {
                Language = language
            });
        return response.SeverityLevels;
    }



    /// <summary>
    /// Create a new support case.
    /// </summary>
    /// <param name="serviceCode">Service code for the new case.</param>
    /// <param name="categoryCode">Category for the new case.</param>
    /// <param name="severityCode">Severity code for the new case.</param>
    /// <param name="subject">Subject of the new case.</param>
    /// <param name="body">Body text of the new case.</param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set for the
new case.</param>
    /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
    /// <returns>The caseId of the new support case.</returns>
```

```
    public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
        string body, string language = "en", string? attachmentSetId = null,
string issueType = "customer-service")
    {
        var response = await _amazonSupport.CreateCaseAsync(
            new CreateCaseRequest()
            {
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
                CommunicationBody = body
            });
        return response.CaseId;
    }



    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does
not exist.
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates
a new attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string
fileName, string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
```

```
                    }
                }
            });
        return response.AttachmentSetId;
    }



    /// <summary>
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
 describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }



    /// <summary>
    /// Add communication to a case, including optional attachment set ID and CC
 email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
```

```
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }



    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
 DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
 _amazonSupport.Paginators.DescribeCommunications(
            new DescribeCommunicationsRequest()
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in
 paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }



    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
```

```
    /// <param name="displayId">Optional display ID.</param>
    /// <param name="includeCommunication">True to include communication.
 Defaults to true.</param>
    /// <param name="includeResolvedCases">True to include resolved cases.
 Defaults to false.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
 (“ko”) are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds,
 string? displayId = null, bool includeCommunication = true,
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
 beforeTime = null,
        string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginators.DescribeCases(
            new DescribeCasesRequest()
            {
                CaseIdList = caseIds,
                DisplayId = displayId,
                IncludeCommunications = includeCommunication,
                IncludeResolvedCases = includeResolvedCases,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s"),
                Language = language
            });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }



    /// <summary>
    /// Resolve a support case by caseId.
    /// </summary>
```

```
    /// <param name="caseId">Id for the support case.</param>
    /// <returns>The final status of the case after resolving.</returns>
    public async Task<string> ResolveCase(string caseId)
    {
        var response = await _amazonSupport.ResolveCaseAsync(
            new ResolveCaseRequest()
            {
                CaseId = caseId
            });
        return response.FinalCaseStatus;
    }


    /// <summary>
    /// Verify the support level for AWS Support API access.
    /// </summary>
    /// <returns>True if the subscription level supports API access.</returns>
    public async Task<bool> VerifySubscription()
    {
        try
        {
            var response = await _amazonSupport.DescribeServicesAsync(
                new DescribeServicesRequest()
                {
                    Language = "en"
                });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Amazon.AWSSupport.AmazonAWSSupportException ex)
        {
            if (ex.ErrorCode == "SubscriptionRequiredException")
            {
                return false;
            }
            else throw;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AddAttachmentsToSet

- AddCommunicationToCase
- CreateCase
- DescribeAttachment
- DescribeCases
- DescribeCommunications
- DescribeServices
- DescribeSeverityLevels
- ResolveCase

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute várias Suporte operações.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import
 software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import
 software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
```

```
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import
 software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import
 software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import
 software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import
 software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
```

```
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <fileAttachment>Where:
                    fileAttachment - The file can be a simple saved .txt file to
 use as an email attachment.\s
                """;

      //  if (args.length != 1) {
      //      System.out.println(usage);
      //      System.exit(1);
      //  }

        String fileAttachment = "C:\\AWS\\test.txt" ; //args[0];
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("***** Welcome to the AWS Support case example
 scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Get and display available services.");
        List<String> sevCatList = displayServices(supportClient);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("2. Get and display Support severity levels.");
        String sevLevel = displaySevLevels(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a support case using the selected service,
category, and severity level.");
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
        if (caseId.compareTo("") == 0) {
            System.out.println("A support case was not successfully created!");
            System.exit(1);
        } else
            System.out.println("Support case " + caseId + " was successfully
created!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to
add to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the
support case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current
day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(30)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .includeResolvedCases(true)
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                if (sinCase.status().compareTo("resolved") == 0)
                    System.out.println("The case status is " + sinCase.status());
            }
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();

            ResolveCaseResponse response =
supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeAttachment(SupportClient supportClient, String
attachId) {
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                    .attachmentId(attachId)
                    .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static String listCommunications(SupportClient supportClient, String
caseId) {
       try {
           String attachId = null;
           DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                   .caseId(caseId)
                   .maxResults(10)
                   .build();

           DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
           List<Communication> communications = response.communications();
           for (Communication comm : communications) {
               System.out.println("the body is: " + comm.body());

               // Get the attachment id value.
               List<AttachmentDetails> attachments = comm.attachmentSet();
               for (AttachmentDetails detail : attachments) {
                   attachId = detail.attachmentId();
               }
           }
           return attachId;

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }

   public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
       try {
           AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                   .caseId(caseId)
                   .attachmentSetId(attachmentSetId)
                   .communicationBody("Please refer to attachment for details.")
                   .build();

           AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
           if (response.result())
```

```
                System.out.println("You have successfully added a communication
to an AWS Support case");
            else
                System.out.println("There was an error adding the communication
to an AWS Support case");

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
        try {
            File myFile = new File(fileAttachment);
            InputStream sourceStream = new FileInputStream(myFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            Attachment attachment = Attachment.builder()
                    .fileName(myFile.getName())
                    .data(sourceBytes)
                    .build();

            AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                    .attachments(attachment)
                    .build();

            AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
            return response.attachmentSetId();

        } catch (SupportException | FileNotFoundException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
```

```
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String createSupportCase(SupportClient supportClient,
List<String> sevCatList, String sevLevel) {
        try {
            String serviceCode = sevCatList.get(0);
            String caseCat = sevCatList.get(1);
            CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                    .categoryCode(caseCat.toLowerCase())
                    .serviceCode(serviceCode.toLowerCase())
                    .severityCode(sevLevel.toLowerCase())
                    .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                    .subject("Test case, please ignore")
                    .language("en")
                    .issueType("technical")
                    .build();

            CreateCaseResponse response = supportClient.createCase(caseRequest);
            return response.caseId();
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static String displaySevLevels(SupportClient supportClient) {
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
```

```
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AddAttachmentsToSet
  - AddCommunicationToCase

- CreateCase
- DescribeAttachment
- DescribeCases
- DescribeCommunications
- DescribeServices
- DescribeSeverityLevels
- ResolveCase

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo no terminal.

```
import {
  AddAttachmentsToSetCommand,
  AddCommunicationToCaseCommand,
  CreateCaseCommand,
  DescribeAttachmentCommand,
  DescribeCasesCommand,
  DescribeCommunicationsCommand,
  DescribeServicesCommand,
  DescribeSeverityLevelsCommand,
  ResolveCaseCommand,
  SupportClient,
} from "@aws-sdk/client-support";
import * as inquirer from "@inquirer/prompts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

```
const client = new SupportClient({ region: "us-east-1" });

// Verify that the account has a Support plan.
export const verifyAccount = async () => {
  const command = new DescribeServicesCommand({});

  try {
    await client.send(command);
  } catch (err) {
    if (err.name === "SubscriptionRequiredException") {
      throw new Error(
        "You must be subscribed to the AWS Support plan to use this feature.",
      );
    }
    throw err;
  }
};

/**
 * Select a service from the list returned from DescribeServices.
 */
export const getService = async () => {
  const { services } = await client.send(new DescribeServicesCommand({}));
  const selectedService = await inquirer.select({
    message:
      "Select a service. Your support case will be created for this service. The
 list of services is truncated for readability.",
    choices: services.slice(0, 10).map((s) => ({ name: s.name, value: s })),
  });
  return selectedService;
};

/**
 * @param {{ categories: import('@aws-sdk/client-support').Category[]}} service
 */
export const getCategory = async (service) => {
  const selectedCategory = await inquirer.select({
    message: "Select a category.",
    choices: service.categories.map((c) => ({ name: c.name, value: c })),
  });
  return selectedCategory;
};
```

```
// Get the available severity levels for the account.
export const getSeverityLevel = async () => {
  const command = new DescribeSeverityLevelsCommand({});
  const { severityLevels } = await client.send(command);
  const selectedSeverityLevel = await inquirer.select({
    message: "Select a severity level.",
    choices: severityLevels.map((s) => ({ name: s.name, value: s })),
  });
  return selectedSeverityLevel;
};

/**
 * Create a new support case
 * @param {{
 *  selectedService: import('@aws-sdk/client-support').Service
 *  selectedCategory: import('@aws-sdk/client-support').Category
 *  selectedSeverityLevel: import('@aws-sdk/client-support').SeverityLevel
 * }} selections
 * @returns
 */
export const createCase = async ({
  selectedService,
  selectedCategory,
  selectedSeverityLevel,
}) => {
  const command = new CreateCaseCommand({
    subject: "IGNORE: Test case",
    communicationBody: "This is a test. Please ignore.",
    serviceCode: selectedService.code,
    categoryCode: selectedCategory.code,
    severityCode: selectedSeverityLevel.code,
  });
  const { caseId } = await client.send(command);
  return caseId;
};

// Get a list of open support cases created today.
export const getTodaysOpenCases = async () => {
  const d = new Date();
  const startOfToday = new Date(d.getFullYear(), d.getMonth(), d.getDate());
  const command = new DescribeCasesCommand({
    includeCommunications: false,
    afterTime: startOfToday.toISOString(),
  });
```

```
  const { cases } = await client.send(command);

  if (cases.length === 0) {
    throw new Error(
      "Unexpected number of cases. Expected more than 0 open cases.",
    );
  }
  return cases;
};

// Create an attachment set.
export const createAttachmentSet = async () => {
  const command = new AddAttachmentsToSetCommand({
    attachments: [
      {
        fileName: "example.txt",
        data: new TextEncoder().encode("some example text"),
      },
    ],
  });
  const { attachmentSetId } = await client.send(command);
  return attachmentSetId;
};

export const linkAttachmentSetToCase = async (attachmentSetId, caseId) => {
  const command = new AddCommunicationToCaseCommand({
    attachmentSetId,
    caseId,
    communicationBody: "Adding attachment set to case.",
  });
  await client.send(command);
};

// Get all communications for a support case.
export const getCommunications = async (caseId) => {
  const command = new DescribeCommunicationsCommand({
    caseId,
  });
  const { communications } = await client.send(command);
  return communications;
};

/**
```

```
 * @param {import('@aws-sdk/client-support').Communication[]} communications
 */
export const getFirstAttachment = (communications) => {
  const firstCommWithAttachment = communications.find(
    (c) => c.attachmentSet.length > 0,
  );
  return firstCommWithAttachment?.attachmentSet[0].attachmentId;
};

// Get an attachment.
export const getAttachment = async (attachmentId) => {
  const command = new DescribeAttachmentCommand({
    attachmentId,
  });
  const { attachment } = await client.send(command);
  return attachment;
};

// Resolve the case matching the given case ID.
export const resolveCase = async (caseId) => {
  const shouldResolve = await inquirer.confirm({
    message: `Do you want to resolve ${caseId}?`,
  });

  if (shouldResolve) {
    const command = new ResolveCaseCommand({
      caseId: caseId,
    });

    await client.send(command);
    return true;
  }
  return false;
};

/**
 * Find a specific case in the list of provided cases by case ID.
 * If the case is not found, and the results are paginated, continue
 * paging through the results.
 * @param {{
 *   caseId: string,
 *   cases: import('@aws-sdk/client-support').CaseDetails[]
 *   nextToken: string
 * }} options
```

```
 * @returns
 */
export const findCase = async ({ caseId, cases, nextToken }) => {
  const foundCase = cases.find((c) => c.caseId === caseId);

  if (foundCase) {
    return foundCase;
  }

  if (nextToken) {
    const response = await client.send(
      new DescribeCasesCommand({
        nextToken,
        includeResolvedCases: true,
      }),
    );
    return findCase({
      caseId,
      cases: response.cases,
      nextToken: response.nextToken,
    });
  }

  throw new Error(`${caseId} not found.`);
};

// Get all cases created today.
export const getTodaysResolvedCases = async (caseIdToWaitFor) => {
  const d = new Date("2023-01-18");
  const startOfToday = new Date(d.getFullYear(), d.getMonth(), d.getDate());
  const command = new DescribeCasesCommand({
    includeCommunications: false,
    afterTime: startOfToday.toISOString(),
    includeResolvedCases: true,
  });
  const { cases, nextToken } = await client.send(command);
  await findCase({ cases, caseId: caseIdToWaitFor, nextToken });
  return cases.filter((c) => c.status === "resolved");
};

const main = async () => {
  let caseId;
  try {
    console.log(wrapText("Welcome to the AWS Support basic usage scenario."));
```

```
   // Verify that the account is subscribed to support.
   await verifyAccount();

   // Provided a truncated list of services and prompt the user to select one.
   const selectedService = await getService();

   // Provided the categories for the selected service and prompt the user to
select one.
   const selectedCategory = await getCategory(selectedService);

   // Provide the severity available severity levels for the account and prompt
the user to select one.
   const selectedSeverityLevel = await getSeverityLevel();

   // Create a support case.
   console.log("\nCreating a support case.");
   caseId = await createCase({
     selectedService,
     selectedCategory,
     selectedSeverityLevel,
   });
   console.log(`Support case created: ${caseId}`);

   // Display a list of open support cases created today.
   const todaysOpenCases = await retry(
     { intervalInMs: 1000, maxRetries: 15 },
     getTodaysOpenCases,
   );
   console.log(
     `\nOpen support cases created today: ${todaysOpenCases.length}`,
   );
   console.log(todaysOpenCases.map((c) => `${c.caseId}`).join("\n"));

   // Create an attachment set.
   console.log("\nCreating an attachment set.");
   const attachmentSetId = await createAttachmentSet();
   console.log(`Attachment set created: ${attachmentSetId}`);

   // Add the attachment set to the support case.
   console.log(`\nAdding attachment set to ${caseId}`);
   await linkAttachmentSetToCase(attachmentSetId, caseId);
   console.log(`Attachment set added to ${caseId}`);
```

```
    // List the communications for a support case.
    console.log(`\nListing communications for ${caseId}`);
    const communications = await getCommunications(caseId);
    console.log(
      communications
        .map(
          (c) =>
            `Communication created on ${c.timeCreated}. Has
 ${c.attachmentSet.length} attachments.`,
        )
        .join("\n"),
    );

    // Describe the first attachment.
    console.log(`\nDescribing attachment ${attachmentSetId}`);
    const attachmentId = getFirstAttachment(communications);
    const attachment = await getAttachment(attachmentId);
    console.log(
      `Attachment is the file '${
        attachment.fileName
      }' with data: \n${new TextDecoder().decode(attachment.data)}`,
    );

    // Confirm that the support case should be resolved.
    const isResolved = await resolveCase(caseId);
    if (isResolved) {
      // List the resolved cases and include the one previously created.
      // Resolved cases can take a while to appear.
      console.log(
        "\nWaiting for case status to be marked as resolved. This can take some
 time.",
      );
      const resolvedCases = await retry(
        { intervalInMs: 20000, maxRetries: 15 },
        () => getTodaysResolvedCases(caseId),
      );
      console.log("Resolved cases:");
      console.log(resolvedCases.map((c) => c.caseId).join("\n"));
    }
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
In addition, you must have the AWS Business Support Plan to use the AWS Support
 Java API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/
```

```
This Kotlin example performs the following tasks:
1. Gets and displays available services.
2. Gets and displays severity levels.
3. Creates a support case by using the selected service, category, and severity
 level.
4. Gets a list of open cases for the current day.
5. Creates an attachment set with a generated file.
6. Adds a communication with the attachment to the support case.
7. Lists the communications of the support case.
8. Describes the attachment set included with the communication.
9. Resolves the support case.
10. Gets a list of resolved cases for the current day.
*/

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <fileAttachment>
    Where:
         fileAttachment - The file can be a simple saved .txt file to use as an
 email attachment.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val fileAttachment = args[0]
    println("***** Welcome to the AWS Support case example scenario.")
    println("***** Step 1. Get and display available services.")
    val sevCatList = displayServices()

    println("***** Step 2. Get and display Support severity levels.")
    val sevLevel = displaySevLevels()

    println("***** Step 3. Create a support case using the selected service,
 category, and severity level.")
    val caseIdVal = createSupportCase(sevCatList, sevLevel)
    if (caseIdVal != null) {
        println("Support case $caseIdVal was successfully created!")
    } else {
        println("A support case was not successfully created!")
        exitProcess(1)
```

```
    }

    println("***** Step 4. Get open support cases.")
    getOpenCase()

    println("***** Step 5. Create an attachment set with a generated file to add
 to the case.")
    val attachmentSetId = addAttachment(fileAttachment)
    println("The Attachment Set id value is $attachmentSetId")

    println("***** Step 6. Add communication with the attachment to the support
 case.")
    addAttachSupportCase(caseIdVal, attachmentSetId)

    println("***** Step 7. List the communications of the support case.")
    val attachId = listCommunications(caseIdVal)
    println("The Attachment id value is $attachId")

    println("***** Step 8. Describe the attachment set included with the
 communication.")
    describeAttachment(attachId)

    println("***** Step 9. Resolve the support case.")
    resolveSupportCase(caseIdVal)

    println("***** Step 10. Get a list of resolved cases for the current day.")
    getResolvedCase()
    println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 30
            afterTime = yesterday.toString()
            beforeTime = now.toString()
            includeResolvedCases = true
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
```

```
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response =
 supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
```

```
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS
 Support case")
        } else {
            println("There was an error adding the communication to an AWS
 Support case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }
```

```
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
```

```
            issueType = "technical"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response =
 supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1
```

```
        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class SupportCasesScenario:
    """Runs an interactive scenario that shows how to get started using AWS
 Support."""

    def __init__(self, support_wrapper):
        """
        :param support_wrapper: An object that wraps AWS Support actions.
        """
        self.support_wrapper = support_wrapper

    def display_and_select_service(self):
        """
        Lists support services and prompts the user to select one.

        :return: The support service selected by the user.
        """
        print("-" * 88)
        services_list = self.support_wrapper.describe_services("en")
        print(f"AWS Support client returned {len(services_list)} services.")
        print("Displaying first 10 services:")

        service_choices = [svc["name"] for svc in services_list[:10]]
        selected_index = q.choose(
            "Select an example support service by entering a number from the
 preceding list:",
            service_choices,
        )
        selected_service = services_list[selected_index]
        print("-" * 88)
        return selected_service
```

```
    def display_and_select_category(self, service):
        """
        Lists categories for a support service and prompts the user to select
one.

        :param service: The service of the categories.
        :return: The selected category.
        """
        print("-" * 88)
        print(
            f"Available support categories for Service {service['name']}
{len(service['categories'])}:"
        )
        categories_choices = [category["name"] for category in
service["categories"]]
        selected_index = q.choose(
            "Select an example support category by entering a number from the
preceding list:",
            categories_choices,
        )
        selected_category = service["categories"][selected_index]
        print("-" * 88)
        return selected_category

    def display_and_select_severity(self):
        """
        Lists available severity levels and prompts the user to select one.

        :return: The selected severity level.
        """
        print("-" * 88)
        severity_levels_list =
self.support_wrapper.describe_severity_levels("en")
        print(f"Available severity levels:")
        severity_choices = [level["name"] for level in severity_levels_list]
        selected_index = q.choose(
            "Select an example severity level by entering a number from the
preceding list:",
            severity_choices,
        )
        selected_severity = severity_levels_list[selected_index]
        print("-" * 88)
        return selected_severity
```

```
    def create_example_case(self, service, category, severity_level):
        """
        Creates an example support case with the user's selections.

        :param service: The service for the new case.
        :param category: The category for the new case.
        :param severity_level: The severity level for the new case.
        :return: The caseId of the new support case.
        """
        print("-" * 88)
        print(f"Creating new case for service {service['name']}.")
        case_id = self.support_wrapper.create_case(service, category,
severity_level)
        print(f"\tNew case created with ID {case_id}.")
        print("-" * 88)
        return case_id

    def list_open_cases(self):
        """
        List the open cases for the current day.
        """
        print("-" * 88)
        print("Let's list the open cases for the current day.")
        start_time = str(datetime.utcnow().date())
        end_time = str(datetime.utcnow().date() + timedelta(days=1))
        open_cases = self.support_wrapper.describe_cases(start_time, end_time,
False)
        for case in open_cases:
            print(f"\tCase: {case['caseId']}: status {case['status']}.")
        print("-" * 88)

    def create_attachment_set(self):
        """
        Create an attachment set with a sample file.

        :return: The attachment set ID of the new attachment set.
        """
        print("-" * 88)
        print("Creating attachment set with a sample file.")
        attachment_set_id = self.support_wrapper.add_attachment_to_set()
        print(f"\tNew attachment set created with ID {attachment_set_id}.")
        print("-" * 88)
        return attachment_set_id
```

```
    def add_communication(self, case_id, attachment_set_id):
        """
        Add a communication with an attachment set to the case.

        :param case_id: The ID of the case for the communication.
        :param attachment_set_id: The ID of the attachment set to
        add to the communication.
        """
        print("-" * 88)
        print(f"Adding a communication and attachment set to the case.")
        self.support_wrapper.add_communication_to_case(attachment_set_id,
case_id)
        print(
            f"Added a communication and attachment set {attachment_set_id} to the
case {case_id}."
        )
        print("-" * 88)

    def list_communications(self, case_id):
        """
        List the communications associated with a case.

        :param case_id: The ID of the case.
        :return: The attachment ID of an attachment.
        """
        print("-" * 88)
        print("Let's list the communications for our case.")
        attachment_id = ""
        communications =
self.support_wrapper.describe_all_case_communications(case_id)
        for communication in communications:
            print(
                f"\tCommunication created on {communication['timeCreated']} "
                f"has {len(communication['attachmentSet'])} attachments."
            )
            if len(communication["attachmentSet"]) > 0:
                attachment_id = communication["attachmentSet"][0]["attachmentId"]
        print("-" * 88)
        return attachment_id

    def describe_case_attachment(self, attachment_id):
        """
        Describe an attachment associated with a case.
```

```
        :param attachment_id: The ID of the attachment.
        """
        print("-" * 88)
        print("Let's list the communications for our case.")
        attached_file = self.support_wrapper.describe_attachment(attachment_id)
        print(f"\tAttachment includes file {attached_file}.")
        print("-" * 88)

    def resolve_case(self, case_id):
        """
        Shows how to resolve an AWS Support case by its ID.

        :param case_id: The ID of the case to resolve.
        """
        print("-" * 88)
        print(f"Resolving case with ID {case_id}.")
        case_status = self.support_wrapper.resolve_case(case_id)
        print(f"\tFinal case status is {case_status}.")
        print("-" * 88)

    def list_resolved_cases(self):
        """
        List the resolved cases for the current day.
        """
        print("-" * 88)
        print("Let's list the resolved cases for the current day.")
        start_time = str(datetime.utcnow().date())
        end_time = str(datetime.utcnow().date() + timedelta(days=1))
        resolved_cases = self.support_wrapper.describe_cases(start_time,
end_time, True)
        for case in resolved_cases:
            print(f"\tCase: {case['caseId']}: status {case['status']}.")
        print("-" * 88)

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        print("-" * 88)
        print("Welcome to the AWS Support get started with support cases demo.")
        print("-" * 88)

        selected_service = self.display_and_select_service()
        selected_category = self.display_and_select_category(selected_service)
```

```
        selected_severity = self.display_and_select_severity()
        new_case_id = self.create_example_case(
            selected_service, selected_category, selected_severity
        )
        wait(10)
        self.list_open_cases()
        new_attachment_set_id = self.create_attachment_set()
        self.add_communication(new_case_id, new_attachment_set_id)
        new_attachment_id = self.list_communications(new_case_id)
        self.describe_case_attachment(new_attachment_id)
        self.resolve_case(new_case_id)
        wait(10)
        self.list_resolved_cases()

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = SupportCasesScenario(SupportWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que envolva ações de suporte ao cliente.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
```

```
        return cls(support_client)


    def describe_services(self, language):
        """
        Get the descriptions of AWS services available for support for a
language.

        :param language: The language for support services.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of AWS service descriptions.
        """
        try:
            response = self.support_client.describe_services(language=language)
            services = response["services"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get Support services for language %s. Here's why:
%s: %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return services


    def describe_severity_levels(self, language):
        """
        Get the descriptions of available severity levels for support cases for a
language.

        :param language: The language for support severity levels.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
```

```
        :return: The list of severity levels.
        """
        try:
            response =
self.support_client.describe_severity_levels(language=language)
            severity_levels = response["severityLevels"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get severity levels for language %s. Here's why:
%s: %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return severity_levels


    def create_case(self, service, category, severity):
        """
        Create a new support case.

        :param service: The service to use for the new case.
        :param category: The category to use for the new case.
        :param severity: The severity to use for the new case.
        :return: The caseId of the new case.
        """
        try:
            response = self.support_client.create_case(
                subject="Example case for testing, ignore.",
                serviceCode=service["code"],
                severityCode=severity["code"],
                categoryCode=category["code"],
                communicationBody="Example support case body.",
```

```
                language="en",
                issueType="customer-service",
            )
            case_id = response["caseId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't create case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return case_id


    def add_attachment_to_set(self):
        """
        Add an attachment to a set, or create a new attachment set if one does
not exist.

        :return: The attachment set ID.
        """
        try:
            response = self.support_client.add_attachments_to_set(
                attachments=[
                    {
                        "fileName": "attachment_file.txt",
                        "data": b"This is a sample file for attachment to a
support case.",
                    }
                ]
            )
            new_set_id = response["attachmentSetId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
```

```
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add attachment. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return new_set_id


    def add_communication_to_case(self, attachment_set_id, case_id):
        """
        Add a communication and an attachment set to a case.

        :param attachment_set_id: The ID of an existing attachment set.
        :param case_id: The ID of the case.
        """
        try:
            self.support_client.add_communication_to_case(
                caseId=case_id,
                communicationBody="This is an example communication added to a
support case.",
                attachmentSetId=attachment_set_id,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add communication. Here's why: %s: %s",
```

```
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise


    def describe_all_case_communications(self, case_id):
        """
        Describe all the communications for a case using a paginator.

        :param case_id: The ID of the case.
        :return: The communications for the case.
        """
        try:
            communications = []
            paginator =
self.support_client.get_paginator("describe_communications")
            for page in paginator.paginate(caseId=case_id):
                communications += page["communications"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe communications. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return communications


    def describe_attachment(self, attachment_id):
        """
        Get information about an attachment by its attachmentID.

        :param attachment_id: The ID of the attachment.
```

```
        :return: The name of the attached file.
        """
        try:
            response = self.support_client.describe_attachment(
                attachmentId=attachment_id
            )
            attached_file = response["attachment"]["fileName"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get attachment description. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return attached_file


    def resolve_case(self, case_id):
        """
        Resolve a support case by its caseId.

        :param case_id: The ID of the case to resolve.
        :return: The final status of the case.
        """
        try:
            response = self.support_client.resolve_case(caseId=case_id)
            final_status = response["finalCaseStatus"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
```

```
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't resolve case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return final_status


    def describe_cases(self, after_time, before_time, resolved):
        """
        Describe support cases over a period of time, optionally filtering
        by status.

        :param after_time: The start time to include for cases.
        :param before_time: The end time to include for cases.
        :param resolved: True to include resolved cases in the results,
            otherwise results are open cases.
        :return: The final status of the case.
        """
        try:
            cases = []
            paginator = self.support_client.get_paginator("describe_cases")
            for page in paginator.paginate(
                afterTime=after_time,
                beforeTime=before_time,
                includeResolvedCases=resolved,
                language="en",
            ):
                cases += page["cases"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
```

```
                logger.error(
                    "Couldn't describe cases. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            if resolved:
                cases = filter(lambda case: case["status"] == "resolved", cases)
            return cases
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Ações para Suporte usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar Suporte ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a Suporte API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para Suporte usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Support.

Exemplos

- Use AddAttachmentsToSetcom um AWS SDK ou CLI
- Use AddCommunicationToCasecom um AWS SDK ou CLI
- Use CreateCasecom um AWS SDK ou CLI
- Use DescribeAttachmentcom um AWS SDK ou CLI
- Use DescribeCasescom um AWS SDK ou CLI
- Use DescribeCommunicationscom um AWS SDK ou CLI
- Use DescribeServicescom um AWS SDK ou CLI
- Use DescribeSeverityLevelscom um AWS SDK ou CLI
- Usar DescribeTrustedAdvisorCheckRefreshStatuses com uma CLI
- Usar DescribeTrustedAdvisorCheckResult com uma CLI
- Usar DescribeTrustedAdvisorCheckSummaries com uma CLI
- Usar DescribeTrustedAdvisorChecks com uma CLI
- Usar RefreshTrustedAdvisorCheck com uma CLI
- Use ResolveCasecom um AWS SDK ou CLI

Use **`AddAttachmentsToSet`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AddAttachmentsToSet`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does
not exist.
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates
a new attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string
fileName, string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
                    }
                }
            });
        return response.AttachmentSetId;
    }
```

- Para obter detalhes da API, consulte AddAttachmentsToSeta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para adicionar um anexo a um conjunto

O `add-attachments-to-set` exemplo a seguir adiciona uma imagem a um conjunto que você pode especificar para um caso de suporte em sua AWS conta.

```
aws support add-attachments-to-set \
    --attachment-set-id "as-2f5a6faa2a4a1e600-mu-nk5xQlBr70-
G1cUos5LZkd38KOAHZa9BMDVzNEXAMPLE" \
    --attachments fileName=troubleshoot-screenshot.png,data=base64-encoded-string
```

Saída:

```
{
    "attachmentSetId": "as-2f5a6faa2a4a1e600-mu-nk5xQlBr70-
G1cUos5LZkd38KOAHZa9BMDVzNEXAMPLE",
    "expiryTime": "2020-05-14T17:04:40.790+0000"
}
```

Para obter mais informações, consulte Case management no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte AddAttachmentsToSetem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
       try {
           File myFile = new File(fileAttachment);
           InputStream sourceStream = new FileInputStream(myFile);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           Attachment attachment = Attachment.builder()
                   .fileName(myFile.getName())
                   .data(sourceBytes)
                   .build();

           AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                   .attachments(attachment)
                   .build();

           AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
           return response.attachmentSetId();

       } catch (SupportException | FileNotFoundException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte AddAttachmentsToSeta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { AddAttachmentsToSetCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a new attachment set or add attachments to an existing set.
    // Provide an 'attachmentSetId' value to add attachments to an existing set.
    // Use AddCommunicationToCase or CreateCase to associate an attachment set
 with a support case.
    const response = await client.send(
      new AddAttachmentsToSetCommand({
        // You can add up to three attachments per set. The size limit is 5 MB
 per attachment.
        attachments: [
          {
            fileName: "example.txt",
            data: new TextEncoder().encode("some example text"),
          },
        ],
      }),
    );
    // Use this ID in AddCommunicationToCase or CreateCase.
    console.log(response.attachmentSetId);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte AddAttachmentsToSeta Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- Para obter detalhes da API, consulte a AddAttachmentsToSetreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def add_attachment_to_set(self):
        """
        Add an attachment to a set, or create a new attachment set if one does
 not exist.

        :return: The attachment set ID.
        """
        try:
            response = self.support_client.add_attachments_to_set(
                attachments=[
                    {
                        "fileName": "attachment_file.txt",
                        "data": b"This is a sample file for attachment to a
 support case.",
```

```
                    }
                ]
            )
            new_set_id = response["attachmentSetId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add attachment. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return new_set_id
```

- Para obter detalhes da API, consulte a AddAttachmentsToSetReferência da API AWS SDK for Python (Boto3).

Use **`AddCommunicationToCase`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AddCommunicationToCase`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao suporte técnico

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add communication to a case, including optional attachment set ID and CC
 email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }
```

- Para obter detalhes da API, consulte AddCommunicationToCasea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para adicionar uma comunicação a um caso

O `add-communication-to-case` exemplo a seguir adiciona comunicações a um caso de suporte em sua AWS conta.

```
aws support add-communication-to-case \
    --case-id "case-12345678910-2013-c4c1d2bf33c5cf47" \
    --communication-body "I'm attaching a set of images to this case." \
    --cc-email-addresses "myemail@example.com" \
    --attachment-set-id "as-2f5a6faa2a4a1e600-mu-nk5xQlBr70-
G1cUos5LZkd38KOAHZa9BMDVzNEXAMPLE"
```

Saída:

```
{
    "result": true
}
```

Para obter mais informações, consulte Case management no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte AddCommunicationToCaseem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void addAttachSupportCase(SupportClient supportClient, String
 caseId, String attachmentSetId) {
        try {
```

```
            AddCommunicationToCaseRequest caseRequest =
 AddCommunicationToCaseRequest.builder()
                    .caseId(caseId)
                    .attachmentSetId(attachmentSetId)
                    .communicationBody("Please refer to attachment for details.")
                    .build();

            AddCommunicationToCaseResponse response =
 supportClient.addCommunicationToCase(caseRequest);
            if (response.result())
                System.out.println("You have successfully added a communication
 to an AWS Support case");
            else
                System.out.println("There was an error adding the communication
 to an AWS Support case");

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte AddCommunicationToCasea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { AddCommunicationToCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
```

```
  let attachmentSetId;

  try {
    // Add a communication to a case.
    const response = await client.send(
      new AddCommunicationToCaseCommand({
        communicationBody: "Adding an attachment.",
        // Set value to an existing support case id.
        caseId: "CASE_ID",
        // Optional. Set value to an existing attachment set id to add
 attachments to the case.
        attachmentSetId,
      }),
    );
    console.log(response);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte AddCommunicationToCasea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
```

```
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS
 Support case")
        } else {
            println("There was an error adding the communication to an AWS
 Support case")
        }
    }
}
```

- Para obter detalhes da API, consulte a AddCommunicationToCasereferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: adiciona o corpo de uma comunicação por e-mail ao caso especificado.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Exemplo 2: adiciona o corpo de uma comunicação por e-mail ao caso especificado mais um ou mais endereços de e-mail contidos na linha CC do e-mail.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- Para obter detalhes da API, consulte AddCommunicationToCaseem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: adiciona o corpo de uma comunicação por e-mail ao caso especificado.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Exemplo 2: adiciona o corpo de uma comunicação por e-mail ao caso especificado mais um ou mais endereços de e-mail contidos na linha CC do e-mail.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- Para obter detalhes da API, consulte AddCommunicationToCaseem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
```

```
        support_client = boto3.client("support")
        return cls(support_client)


    def add_communication_to_case(self, attachment_set_id, case_id):
        """
        Add a communication and an attachment set to a case.

        :param attachment_set_id: The ID of an existing attachment set.
        :param case_id: The ID of the case.
        """
        try:
            self.support_client.add_communication_to_case(
                caseId=case_id,
                communicationBody="This is an example communication added to a
support case.",
                attachmentSetId=attachment_set_id,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add communication. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a AddCommunicationToCaseReferência da API AWS SDK for Python (Boto3).

Use **`CreateCase`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateCase`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao suporte técnico

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create a new support case.
   /// </summary>
   /// <param name="serviceCode">Service code for the new case.</param>
   /// <param name="categoryCode">Category for the new case.</param>
   /// <param name="severityCode">Severity code for the new case.</param>
   /// <param name="subject">Subject of the new case.</param>
   /// <param name="body">Body text of the new case.</param>
   /// <param name="language">Optional language support for your case.
   /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
   /// <param name="attachmentSetId">Optional Id for an attachment set for the
new case.</param>
   /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
   /// <returns>The caseId of the new support case.</returns>
   public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
       string body, string language = "en", string? attachmentSetId = null,
string issueType = "customer-service")
   {
       var response = await _amazonSupport.CreateCaseAsync(
           new CreateCaseRequest()
           {
```

```
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
                CommunicationBody = body
            });
        return response.CaseId;
    }
```

- Para obter detalhes da API, consulte CreateCasea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar um caso

O `create-case` exemplo a seguir cria um caso de suporte para sua AWS conta.

```
aws support create-case \
    --category-code "using-aws" \
    --cc-email-addresses "myemail@example.com" \
    --communication-body "I want to learn more about an AWS service." \
    --issue-type "technical" \
    --language "en" \
    --service-code "general-info" \
    --severity-code "low" \
    --subject "Question about my account"
```

Saída:

```
{
    "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47"
}
```

Para obter mais informações, consulte Case management no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte CreateCaseem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createSupportCase(SupportClient supportClient,
List<String> sevCatList, String sevLevel) {
       try {
           String serviceCode = sevCatList.get(0);
           String caseCat = sevCatList.get(1);
           CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                   .categoryCode(caseCat.toLowerCase())
                   .serviceCode(serviceCode.toLowerCase())
                   .severityCode(sevLevel.toLowerCase())
                   .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                   .subject("Test case, please ignore")
                   .language("en")
                   .issueType("technical")
                   .build();

           CreateCaseResponse response = supportClient.createCase(caseRequest);
           return response.caseId();

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte CreateCasea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a new case and log the case id.
    // Important: This creates a real support case in your account.
    const response = await client.send(
      new CreateCaseCommand({
        // The subject line of the case.
        subject: "IGNORE: Test case",
        // Use DescribeServices to find available service codes for each service.
        serviceCode: "service-quicksight-end-user",
        // Use DescribeSecurityLevels to find available severity codes for your
 support plan.
        severityCode: "low",
        // Use DescribeServices to find available category codes for each
 service.
        categoryCode: "end-user-support",
        // The main description of the support case.
        communicationBody: "This is a test. Please ignore.",
      }),
    );
    console.log(response.caseId);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte CreateCasea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- Para obter detalhes da API, consulte a CreateCasereferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Cria um novo caso no AWS Support Center. Os valores dos CategoryCode parâmetros - ServiceCode e - podem ser obtidos usando o Get-ASAService cmdlet. O valor do SeverityCode parâmetro - pode ser obtido usando o Get-ASASeverityLevel cmdlet. O valor do IssueType parâmetro - pode ser “atendimento ao cliente” ou “técnico”. Se for bem-sucedido, o número do caso de AWS Support será exibido. Por padrão, o caso será tratado em inglês. Para usar o japonês, adicione o parâmetro -Language “ja”. Os CommunicationBody parâmetros -ServiceCode, -CategoryCode, -Assunto e - são obrigatórios.

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode
 "low" -Subject "subject text" -CommunicationBody "description of the case" -
CcEmailAddress @("email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- Para obter detalhes da API, consulte CreateCaseem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Cria um novo caso no AWS Support Center. Os valores dos CategoryCode parâmetros - ServiceCode e - podem ser obtidos usando o Get-ASAService cmdlet. O valor do SeverityCode parâmetro - pode ser obtido usando o Get-ASASeverityLevel cmdlet. O valor do IssueType parâmetro - pode ser “atendimento ao cliente” ou “técnico”. Se for bem-sucedido, o número do caso de AWS Support será exibido. Por padrão, o caso será tratado em inglês. Para usar o japonês, adicione o parâmetro -Language “ja”. Os CommunicationBody parâmetros -ServiceCode, -CategoryCode, -Assunto e - são obrigatórios.

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode
 "low" -Subject "subject text" -CommunicationBody "description of the case" -
CcEmailAddress @("email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- Para obter detalhes da API, consulte CreateCaseem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def create_case(self, service, category, severity):
        """
        Create a new support case.

        :param service: The service to use for the new case.
        :param category: The category to use for the new case.
        :param severity: The severity to use for the new case.
        :return: The caseId of the new case.
        """
        try:
            response = self.support_client.create_case(
                subject="Example case for testing, ignore.",
                serviceCode=service["code"],
                severityCode=severity["code"],
```

```
                categoryCode=category["code"],
                communicationBody="Example support case body.",
                language="en",
                issueType="customer-service",
            )
            case_id = response["caseId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't create case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return case_id
```

- Para obter detalhes da API, consulte a CreateCaseReferência da API AWS SDK for Python (Boto3).

Use **`DescribeAttachment`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeAttachment`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
 describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }
```

- Para obter detalhes da API, consulte DescribeAttachmenta Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como descrever um anexo

O exemplo de `describe-attachment` a seguir retorna informações sobre o anexo com o ID especificado.

```
aws support describe-attachment \
    --attachment-id "attachment-KBnjRNrePd9D6Jx0-Mm00xZuDEaL2JAj_0-
gJv9qqDooTipsz3V1Nb19rCfkZneeQeDPgp8X1iVJyHH7UuhZDdNeqGoduZsPrAhyMakqlc60-
iJjL5HqyYGiT1FG8EXAMPLE"
```

Saída:

```
{
    "attachment": {
        "fileName": "troubleshoot-screenshot.png",
        "data": "base64-blob"
    }
}
```

Para obter mais informações, consulte Case management no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeAttachmentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeAttachment(SupportClient supportClient, String
 attachId) {
        try {
            DescribeAttachmentRequest attachmentRequest =
 DescribeAttachmentRequest.builder()
                    .attachmentId(attachId)
                    .build();

            DescribeAttachmentResponse response =
 supportClient.describeAttachment(attachmentRequest);
```

```
            System.out.println("The name of the file is " +
 response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeAttachmenta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeAttachmentCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get the metadata and content of an attachment.
    const response = await client.send(
      new DescribeAttachmentCommand({
        // Set value to an existing attachment id.
        // Use DescribeCommunications or DescribeCases to find an attachment id.
        attachmentId: "ATTACHMENT_ID",
      }),
    );
    console.log(response.attachment?.fileName);
    return response;
  } catch (err) {
    console.error(err);
```

```
  }
};
```

- Para obter detalhes da API, consulte DescribeAttachmenta Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeAttachmentreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_attachment(self, attachment_id):
        """
        Get information about an attachment by its attachmentID.

        :param attachment_id: The ID of the attachment.
        :return: The name of the attached file.
        """
        try:
            response = self.support_client.describe_attachment(
                attachmentId=attachment_id
            )
            attached_file = response["attachment"]["fileName"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
```

```
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get attachment description. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return attached_file
```

- Para obter detalhes da API, consulte a DescribeAttachmentReferência da API AWS SDK for Python (Boto3).

Use **`DescribeCases`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeCases`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao suporte técnico

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
    /// <param name="displayId">Optional display ID.</param>
    /// <param name="includeCommunication">True to include communication.
 Defaults to true.</param>
    /// <param name="includeResolvedCases">True to include resolved cases.
 Defaults to false.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
 (“ko”) are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds,
 string? displayId = null, bool includeCommunication = true,
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
 beforeTime = null,
        string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginators.DescribeCases(
            new DescribeCasesRequest()
            {
                CaseIdList = caseIds,
                DisplayId = displayId,
                IncludeCommunications = includeCommunication,
                IncludeResolvedCases = includeResolvedCases,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s"),
                Language = language
            });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeCasesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como descrever um caso

O `describe-cases` exemplo a seguir retorna informações sobre o caso de suporte especificado em sua AWS conta.

```
aws support describe-cases \
    --display-id "1234567890" \
    --after-time "2020-03-23T21:31:47.774Z" \
    --include-resolved-cases \
    --language "en" \
    --no-include-communications \
    --max-item 1
```

Saída:

```
{
    "cases": [
        {
            "status": "resolved",
            "ccEmailAddresses": [],
            "timeCreated": "2020-03-23T21:31:47.774Z",
            "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47",
            "severityCode": "low",
            "language": "en",
            "categoryCode": "using-aws",
            "serviceCode": "general-info",
            "submittedBy": "myemail@example.com",
            "displayId": "1234567890",
            "subject": "Question about my account"
        }
    ]
}
```

Para obter mais informações, consulte Case management no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeCasesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
```

```
    }
```

- Para obter detalhes da API, consulte DescribeCasesa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeCasesCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get all of the unresolved cases in your account.
    // Filter or expand results by providing parameters to the
 DescribeCasesCommand. Refer
    // to the TypeScript definition and the API doc for more information on
 possible parameters.
    // https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
support/interfaces/describecasescommandinput.html
    const response = await client.send(new DescribeCasesCommand({}));
    const caseIds = response.cases.map((supportCase) => supportCase.caseId);
    console.log(caseIds);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte DescribeCasesa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeCasesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna os detalhes de todos os casos de suporte.

```
Get-ASACase
```

Exemplo 2: retorna os detalhes de todos os casos de suporte desde a data e a hora especificadas.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Exemplo 3: retorna os detalhes dos primeiros 10 casos de suporte, incluindo aqueles que foram resolvidos.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Exemplo 4: retorna os detalhes do único caso de suporte especificado.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemplo 5: retorna os detalhes dos casos de suporte especificados.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

- Para obter detalhes da API, consulte DescribeCasesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna os detalhes de todos os casos de suporte.

```
Get-ASACase
```

Exemplo 2: retorna os detalhes de todos os casos de suporte desde a data e a hora especificadas.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Exemplo 3: retorna os detalhes dos primeiros 10 casos de suporte, incluindo aqueles que foram resolvidos.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Exemplo 4: retorna os detalhes do único caso de suporte especificado.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemplo 5: retorna os detalhes dos casos de suporte especificados.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

- Para obter detalhes da API, consulte DescribeCasesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_cases(self, after_time, before_time, resolved):
```

```
        """
        Describe support cases over a period of time, optionally filtering
        by status.

        :param after_time: The start time to include for cases.
        :param before_time: The end time to include for cases.
        :param resolved: True to include resolved cases in the results,
            otherwise results are open cases.
        :return: The final status of the case.
        """
        try:
            cases = []
            paginator = self.support_client.get_paginator("describe_cases")
            for page in paginator.paginate(
                afterTime=after_time,
                beforeTime=before_time,
                includeResolvedCases=resolved,
                language="en",
            ):
                cases += page["cases"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe cases. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            if resolved:
                cases = filter(lambda case: case["status"] == "resolved", cases)
            return cases
```

- Para obter detalhes da API, consulte a DescribeCasesReferência da API AWS SDK for Python (Boto3).

Use **`DescribeCommunications`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeCommunications`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao suporte técnico

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
 DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
 _amazonSupport.Paginators.DescribeCommunications(
            new DescribeCommunicationsRequest()
```

```
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in
 paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeCommunicationsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como descrever a comunicação mais recente de um caso

O `describe-communications` exemplo a seguir retorna a comunicação mais recente para o caso de suporte especificado em sua AWS conta.

```
aws support describe-communications \
    --case-id "case-12345678910-2013-c4c1d2bf33c5cf47" \
    --after-time "2020-03-23T21:31:47.774Z" \
    --max-item 1
```

Saída:

```
{
    "communications": [
        {
            "body": "I want to learn more about an AWS service.",
            "attachmentSet": [],
            "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47",
            "timeCreated": "2020-05-12T23:12:35.000Z",
```

```
            "submittedBy": "Amazon Web Services"
        }
    ],
    "NextToken":
 "eyJuZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQEXAMPLE=="
}
```

Para obter mais informações, consulte Case management no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeCommunicationsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String listCommunications(SupportClient supportClient, String
caseId) {
       try {
           String attachId = null;
           DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                   .caseId(caseId)
                   .maxResults(10)
                   .build();

           DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
           List<Communication> communications = response.communications();
           for (Communication comm : communications) {
               System.out.println("the body is: " + comm.body());

               // Get the attachment id value.
               List<AttachmentDetails> attachments = comm.attachmentSet();
               for (AttachmentDetails detail : attachments) {
```

```
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte DescribeCommunicationsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeCommunicationsCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get all communications for the support case.
    // Filter results by providing parameters to the
 DescribeCommunicationsCommand. Refer
    // to the TypeScript definition and the API doc for more information on
 possible parameters.
    // https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
support/interfaces/describecommunicationscommandinput.html
    const response = await client.send(
      new DescribeCommunicationsCommand({
```

```
        // Set value to an existing case id.
        caseId: "CASE_ID",
      }),
    );
    const text = response.communications.map((item) => item.body).join("\n");
    console.log(text);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte DescribeCommunicationsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response =
 supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
```

```
        }
    }
    return ""
}
```

- Para obter detalhes da API, consulte a DescribeCommunicationsreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna todas as comunicações do caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemplo 2: retorna todas as comunicações desde a meia-noite UTC de 1.º de janeiro de 2012 para o caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

- Para obter detalhes da API, consulte DescribeCommunicationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna todas as comunicações do caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemplo 2: retorna todas as comunicações desde a meia-noite UTC de 1.º de janeiro de 2012 para o caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

- Para obter detalhes da API, consulte DescribeCommunicationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_all_case_communications(self, case_id):
        """
        Describe all the communications for a case using a paginator.

        :param case_id: The ID of the case.
        :return: The communications for the case.
        """
        try:
            communications = []
            paginator =
 self.support_client.get_paginator("describe_communications")
            for page in paginator.paginate(caseId=case_id):
                communications += page["communications"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe communications. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return communications
```

- Para obter detalhes da API, consulte a DescribeCommunicationsReferência da API AWS SDK for Python (Boto3).

Use **`DescribeServices`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeServices`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao suporte técnico

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
            {
                Language = language
            });
        return response.Services;
    }
```

- Para obter detalhes da API, consulte DescribeServicesa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar AWS serviços e categorias de serviços

O exemplo de `describe-services` a seguir lista as categorias de serviço disponíveis para a solicitação de informações gerais.

```
aws support describe-services \
    --service-code-list "general-info"
```

Saída:

```
{
    "services": [
        {
            "code": "general-info",
            "name": "General Info and Getting Started",
            "categories": [
                {
                    "code": "charges",
                    "name": "How Will I Be Charged?"
                },
                {
                    "code": "gdpr-queries",
                    "name": "Data Privacy Query"
                },
                {
                    "code": "reserved-instances",
                    "name": "Reserved Instances"
                },
                {
                    "code": "resource",
                    "name": "Where is my Resource?"
                },
                {
                    "code": "using-aws",
                    "name": "Using AWS & Services"
                },
                {
                    "code": "free-tier",
                    "name": "Free Tier"
                },
                {
                    "code": "security-and-compliance",
                    "name": "Security & Compliance"
                },
                {
                    "code": "account-structure",
                    "name": "Account Structure"
                }
```

```
            ]
        }
    ]
}
```

Para obter mais informações, consulte Case management no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeServicesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;
```

```
                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte DescribeServicesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Return a List that contains a Service name and Category name.
```

```
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Para obter detalhes da API, consulte a DescribeServicesreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna todos os códigos de serviço, nomes e categorias disponíveis.

```
Get-ASAService
```

Exemplo 2: retorna o nome e as categorias do serviço com o código especificado.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Exemplo 3: retorna o nome e as categorias dos códigos de serviço especificados.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Exemplo 4: retorna o nome e as categorias (em japonês) dos códigos de serviço especificados. Atualmente, os códigos de idioma inglês (“en”) e japonês (“ja”) são compatíveis.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- Para obter detalhes da API, consulte DescribeServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna todos os códigos de serviço, nomes e categorias disponíveis.

```
Get-ASAService
```

Exemplo 2: retorna o nome e as categorias do serviço com o código especificado.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Exemplo 3: retorna o nome e as categorias dos códigos de serviço especificados.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Exemplo 4: retorna o nome e as categorias (em japonês) dos códigos de serviço especificados. Atualmente, os códigos de idioma inglês (“en”) e japonês (“ja”) são compatíveis.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- Para obter detalhes da API, consulte DescribeServicesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_services(self, language):
```

```
        """
        Get the descriptions of AWS services available for support for a
language.

        :param language: The language for support services.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of AWS service descriptions.
        """
        try:
            response = self.support_client.describe_services(language=language)
            services = response["services"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get Support services for language %s. Here's why:
%s: %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return services
```

- Para obter detalhes da API, consulte a DescribeServicesReferência da API AWS SDK for Python (Boto3).

Use **`DescribeSeverityLevels`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeSeverityLevels`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao suporte técnico

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
    /// <returns>The list of support severity levels.</returns>
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language
= "en")
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
            {
                Language = language
            });
        return response.SeverityLevels;
    }
```

- Para obter detalhes da API, consulte DescribeSeverityLevelsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os níveis de gravidade disponíveis

O exemplo de `describe-severity-levels` a seguir lista os níveis de gravidade disponíveis para um caso de suporte.

```
aws support describe-severity-levels
```

Saída:

```
{
    "severityLevels": [
        {
            "code": "low",
            "name": "Low"
        },
        {
            "code": "normal",
            "name": "Normal"
        },
        {
            "code": "high",
            "name": "High"
        },
        {
            "code": "urgent",
            "name": "Urgent"
        },
        {
            "code": "critical",
            "name": "Critical"
        }
    ]
}
```

Para obter mais informações, consulte Choosing a severity no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeSeverityLevelsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String displaySevLevels(SupportClient supportClient) {
       try {
           DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                   .language("en")
                   .build();

           DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
           List<SeverityLevel> severityLevels = response.severityLevels();
           String levelName = null;
           for (SeverityLevel sevLevel : severityLevels) {
               System.out.println("The severity level name is: " +
sevLevel.name());
               if (sevLevel.name().compareTo("High") == 0)
                   levelName = sevLevel.name();
           }
           return levelName;

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte DescribeSeverityLevelsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeSeverityLevelsCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get the list of severity levels.
    // The available values depend on the support plan for the account.
    const response = await client.send(new DescribeSeverityLevelsCommand({}));
    console.log(response.severityLevels);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte DescribeSeverityLevelsa Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response =
 supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}
```

- Para obter detalhes da API, consulte a DescribeSeverityLevelsreferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Retorna a lista de níveis de severidade que podem ser atribuídos a um caso de AWS Support.

```
Get-ASASeverityLevel
```

Exemplo 2: Retorna a lista de níveis de severidade que podem ser atribuídos a um caso de AWS Support. Os nomes dos níveis são retornados em japonês.

```
Get-ASASeverityLevel -Language "ja"
```

- Para obter detalhes da API, consulte DescribeSeverityLevelsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Retorna a lista de níveis de severidade que podem ser atribuídos a um caso de AWS Support.

```
Get-ASASeverityLevel
```

Exemplo 2: Retorna a lista de níveis de severidade que podem ser atribuídos a um caso de AWS Support. Os nomes dos níveis são retornados em japonês.

```
Get-ASASeverityLevel -Language "ja"
```

- Para obter detalhes da API, consulte DescribeSeverityLevelsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
```

```
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_severity_levels(self, language):
        """
        Get the descriptions of available severity levels for support cases for a
language.

        :param language: The language for support severity levels.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of severity levels.
        """
        try:
            response =
self.support_client.describe_severity_levels(language=language)
            severity_levels = response["severityLevels"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get severity levels for language %s. Here's why:
%s: %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return severity_levels
```

- Para obter detalhes da API, consulte a DescribeSeverityLevelsReferência da API AWS SDK for Python (Boto3).

Usar **`DescribeTrustedAdvisorCheckRefreshStatuses`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTrustedAdvisorCheckRefreshStatuses`.

CLI

AWS CLI

Para listar os status de atualização das verificações do AWS Trusted Advisor

O exemplo `describe-trusted-advisor-check-refresh-statuses` a seguir lista os status de atualização de duas verificações do Trusted Advisor: permissões do bucket do Amazon S3 e uso do IAM.

```
aws support describe-trusted-advisor-check-refresh-statuses \
    --check-id "Pfx0RwqBli" "zXCkfM1nI3"
```

Saída:

```
{
    "statuses": [
        {
            "checkId": "Pfx0RwqBli",
            "status": "none",
            "millisUntilNextRefreshable": 0
        },
        {
            "checkId": "zXCkfM1nI3",
            "status": "none",
            "millisUntilNextRefreshable": 0
        }
    ]
}
```

Para obter mais informações, consulte AWS Trusted Advisor no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckRefreshStatusesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna o status atual das solicitações de atualização para as verificações especificadas. Request-ASATrustedAdvisorCheckRefresh pode ser usado para solicitar que as informações de status das verificações sejam atualizadas.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckRefreshStatusesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna o status atual das solicitações de atualização para as verificações especificadas. Request-ASATrustedAdvisorCheckRefresh pode ser usado para solicitar que as informações de status das verificações sejam atualizadas.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckRefreshStatusesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTrustedAdvisorCheckResult`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTrustedAdvisorCheckResult`.

CLI

AWS CLI

Para listar os resultados de uma verificação do AWS Trusted Advisor

O exemplo `describe-trusted-advisor-check-result` a seguir lista os resultados da verificação de uso do IAM.

```
aws support describe-trusted-advisor-check-result \
    --check-id "zXCkfM1nI3"
```

Saída:

```
{
    "result": {
        "checkId": "zXCkfM1nI3",
        "timestamp": "2020-05-13T21:38:05Z",
        "status": "ok",
        "resourcesSummary": {
            "resourcesProcessed": 1,
            "resourcesFlagged": 0,
            "resourcesIgnored": 0,
            "resourcesSuppressed": 0
        },
        "categorySpecificSummary": {
            "costOptimizing": {
                "estimatedMonthlySavings": 0.0,
                "estimatedPercentMonthlySavings": 0.0
            }
        },
        "flaggedResources": [
            {
                "status": "ok",
                "resourceId": "47DEQpj8HBSa-_TImW-5JCeuQeRkm5NMpJWZEXAMPLE",
                "isSuppressed": false
            }
        ]
    }
}
```

Para obter mais informações, consulte AWS Trusted Advisor no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckResultem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna os resultados de uma verificação do Trusted Advisor. A lista de verificações disponíveis do Trusted Advisor pode ser obtida usando Get-ASATrustedAdvisorChecks. A saída é o status geral da verificação, a data e hora em que

a verificação foi executada pela última vez e o ID de verificação exclusivo da verificação específica. Para que os resultados sejam exibidos em japonês, adicione o parâmetro -Language “ja”.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckResultem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna os resultados de uma verificação do Trusted Advisor. A lista de verificações disponíveis do Trusted Advisor pode ser obtida usando Get-ASATrustedAdvisorChecks. A saída é o status geral da verificação, a data e hora em que a verificação foi executada pela última vez e o ID de verificação exclusivo da verificação específica. Para que os resultados sejam exibidos em japonês, adicione o parâmetro -Language “ja”.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckResultem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTrustedAdvisorCheckSummaries`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTrustedAdvisorCheckSummaries`.

CLI

AWS CLI

Para listar os resumos das verificações do AWS Trusted Advisor

O exemplo `describe-trusted-advisor-check-summaries` a seguir lista os resultados de duas verificações do Trusted Advisor: permissões do bucket do Amazon S3 e uso do IAM.

```
aws support describe-trusted-advisor-check-summaries \
    --check-ids "Pfx0RwqBli" "zXCkfM1nI3"
```

Saída:

```
{
    "summaries": [
        {
            "checkId": "Pfx0RwqBli",
            "timestamp": "2020-05-13T21:38:12Z",
            "status": "ok",
            "hasFlaggedResources": true,
            "resourcesSummary": {
                "resourcesProcessed": 44,
                "resourcesFlagged": 0,
                "resourcesIgnored": 0,
                "resourcesSuppressed": 0
            },
            "categorySpecificSummary": {
                "costOptimizing": {
                    "estimatedMonthlySavings": 0.0,
                    "estimatedPercentMonthlySavings": 0.0
                }
            }
        },
        {
            "checkId": "zXCkfM1nI3",
            "timestamp": "2020-05-13T21:38:05Z",
            "status": "ok",
            "hasFlaggedResources": true,
            "resourcesSummary": {
                "resourcesProcessed": 1,
                "resourcesFlagged": 0,
                "resourcesIgnored": 0,
                "resourcesSuppressed": 0
            },
            "categorySpecificSummary": {
                "costOptimizing": {
                    "estimatedMonthlySavings": 0.0,
                    "estimatedPercentMonthlySavings": 0.0
                }
            }
        }
    ]
}
```

Para obter mais informações, consulte AWS Trusted Advisor no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckSummariesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna o resumo mais recente da verificação especificada do Trusted Advisor.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Exemplo 2: retorna os resumos mais recentes das verificações especificadas do Trusted Advisor.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckSummariesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna o resumo mais recente da verificação especificada do Trusted Advisor.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Exemplo 2: retorna os resumos mais recentes das verificações especificadas do Trusted Advisor.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Para obter detalhes da API, consulte DescribeTrustedAdvisorCheckSummariesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTrustedAdvisorChecks`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTrustedAdvisorChecks`.

CLI

AWS CLI

Para listar as verificações disponíveis do AWS Trusted Advisor

O `describe-trusted-advisor-checks` exemplo a seguir lista os cheques do Trusted Advisor disponíveis em sua AWS conta. Essas informações incluem o nome, ID, descrição, categoria e metadados da verificação. Observe que a saída é reduzida para facilitar a leitura.

```
aws support describe-trusted-advisor-checks \
    --language "en"
```

Saída:

```
{
    "checks": [
        {
            "id": "zXCkfM1nI3",
            "name": "IAM Use",
            "description": "Checks for your use of AWS Identity and Access
 Management (IAM). You can use IAM to create users, groups, and roles in
 AWS, and you can use permissions to control access to AWS resources. \n<br>
\n<br>\n<b>Alert Criteria</b><br>\nYellow: No IAM users have been created
 for this account.\n<br>\n<br>\n<b>Recommended Action</b><br>\nCreate one or
 more IAM users and groups in your account. You can then create additional
 users whose permissions are limited to perform specific tasks in your AWS
 environment. For more information, see <a href=\"https://docs.aws.amazon.com/
IAM/latest/UserGuide/IAMGettingStarted.html\" target=\"_blank\">Getting
 Started</a>. \n<br><br>\n<b>Additional Resources</b><br>\n<a href=\"https://
docs.aws.amazon.com/IAM/latest/UserGuide/IAM_Introduction.html\" target=\"_blank
\">What Is IAM?</a>",
            "category": "security",
            "metadata": []
        }
    ]
}
```

Para obter mais informações, consulte AWS Trusted Advisor no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte DescribeTrustedAdvisorChecksem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna a coleção de verificações do Trusted Advisor. Você deve especificar o parâmetro Language, que pode aceitar “en” para saída em inglês ou “ja” para saída em japonês.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Para obter detalhes da API, consulte DescribeTrustedAdvisorChecksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de verificações do Trusted Advisor. Você deve especificar o parâmetro Language, que pode aceitar “en” para saída em inglês ou “ja” para saída em japonês.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Para obter detalhes da API, consulte DescribeTrustedAdvisorChecksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RefreshTrustedAdvisorCheck`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RefreshTrustedAdvisorCheck`.

CLI

AWS CLI

Para atualizar uma verificação do AWS Trusted Advisor

O `refresh-trusted-advisor-check` exemplo a seguir atualiza o cheque do Amazon S3 Bucket Permissions Trusted Advisor em AWS sua conta.

```
aws support refresh-trusted-advisor-check \
    --check-id "Pfx0RwqBli"
```

Saída:

```
{
    "status": {
        "checkId": "Pfx0RwqBli",
        "status": "enqueued",
        "millisUntilNextRefreshable": 3599992
    }
}
```

Para obter mais informações, consulte AWS Trusted Advisor no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte RefreshTrustedAdvisorCheckem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: solicita uma atualização para a verificação especificada do Trusted Advisor.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Para obter detalhes da API, consulte RefreshTrustedAdvisorCheckem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: solicita uma atualização para a verificação especificada do Trusted Advisor.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Para obter detalhes da API, consulte RefreshTrustedAdvisorCheckem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ResolveCase`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ResolveCase`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Conheça os conceitos básicos
- Introdução ao suporte técnico

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Resolve a support case by caseId.
/// </summary>
/// <param name="caseId">Id for the support case.</param>
/// <returns>The final status of the case after resolving.</returns>
public async Task<string> ResolveCase(string caseId)
{
    var response = await _amazonSupport.ResolveCaseAsync(
        new ResolveCaseRequest()
        {
            CaseId = caseId
        });
    return response.FinalCaseStatus;
}
```

- Para obter detalhes da API, consulte ResolveCasea Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como solucionar um caso de suporte

O `resolve-case` exemplo a seguir resolve um caso de suporte em sua AWS conta.

```
aws support resolve-case \
    --case-id "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Saída:

```
{
    "finalCaseStatus": "resolved",
    "initialCaseStatus": "work-in-progress"
}
```

Para obter mais informações, consulte Case management no Guia do usuário do AWS Support.

- Para obter detalhes da API, consulte ResolveCaseem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void resolveSupportCase(SupportClient supportClient, String
 caseId) {
       try {
           ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                   .caseId(caseId)
                   .build();

           ResolveCaseResponse response =
 supportClient.resolveCase(caseRequest);
           System.out.println("The status of case " + caseId + " is " +
 response.finalCaseStatus());

       } catch (SupportException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ResolveCasea Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ResolveCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

const main = async () => {
  try {
    const response = await client.send(
      new ResolveCaseCommand({
        caseId: "CASE_ID",
      }),
    );

    console.log(response.finalCaseStatus);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte ResolveCasea Referência AWS SDK para JavaScript da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- Para obter detalhes da API, consulte a ResolveCasereferência da API AWS SDK for Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna o estado inicial do caso especificado e o estado atual após a conclusão da chamada para resolvê-lo.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Para obter detalhes da API, consulte ResolveCaseem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna o estado inicial do caso especificado e o estado atual após a conclusão da chamada para resolvê-lo.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Para obter detalhes da API, consulte ResolveCaseem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def resolve_case(self, case_id):
        """
        Resolve a support case by its caseId.

        :param case_id: The ID of the case to resolve.
        :return: The final status of the case.
        """
        try:
```

```
            response = self.support_client.resolve_case(caseId=case_id)
            final_status = response["finalCaseStatus"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't resolve case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return final_status
```

- Para obter detalhes da API, consulte a ResolveCaseReferência da API AWS SDK for Python (Boto3).

Cenários para Suporte usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns Suporte com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas Suporte ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Introdução ao suporte técnico

Introdução ao suporte técnico

O exemplo de código a seguir mostra como:

- Verificar os serviços disponíveis e os níveis de gravidade.
- Criar um caso de suporte.
- Adicionar comunicações a um caso.

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Support CLI Tutorial Script
# This script demonstrates how to use AWS Support API through AWS CLI

set -euo pipefail

# Security: Validate script location and permissions
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
readonly SCRIPT_DIR

# Security: Use secure temporary directory
readonly TEMP_DIR="$(mktemp -d)" || { echo "Failed to create temp directory";
 exit 1; }
trap "rm -rf '$TEMP_DIR'" EXIT

# Set up logging with secure permissions
LOG_FILE="${TEMP_DIR}/aws-support-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Validate AWS CLI is available
if ! command -v aws &> /dev/null; then
```

```
    echo "ERROR: AWS CLI is not installed or not in PATH" >&2
    exit 1
fi

# Security: Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not properly configured" >&2
    exit 1
fi

{
    echo "Starting AWS Support Tutorial at $(date)"
    echo "Script: $0"
    echo "User: $(whoami)"
    echo "---"
} >> "$LOG_FILE"

# Function to log commands and their outputs securely
log_cmd() {
    local cmd="$1"
    echo "$(date): Running command" >> "$LOG_FILE"
    # Don't echo the actual command to prevent credential leakage
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
    local is_fatal="${4:-true}"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command returned status: $cmd_status" >> "$LOG_FILE"

        # Check for subscription error
        if [[ "$cmd_output" =~ "SubscriptionRequiredException" ]]; then
            {
                echo ""
                echo "====================================================="
                echo "IMPORTANT: This account does not have the required AWS
 Support plan."
```

```
                echo "You need a Business, Enterprise On-Ramp, or Enterprise
 Support plan"
                echo "to use the AWS Support API."
                echo ""
                echo "This script will now demonstrate the commands that would be
 run"
                echo "if you had the appropriate support plan, but will not
 execute them."
                echo "====================================================="
            } | tee -a "$LOG_FILE"

            DEMO_MODE=true
            return 0
        fi

        if [[ "$is_fatal" == "true" ]]; then
            cleanup_resources
            exit 1
        fi
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    echo "No persistent resources were created that need cleanup." | tee -a
 "$LOG_FILE"
}

# Function to run a command in demo mode
demo_cmd() {
    local cmd="$1"
    local description="$2"

    {
        echo ""
        echo "DEMO: $description"
        echo "Command that would be executed:"
        echo "  [Command hidden for security]"
        echo ""
    } | tee -a "$LOG_FILE"
}

# Function to safely extract JSON values
```

```
extract_json_value() {
    local json_output="$1"
    local key="$2"

    echo "$json_output" | grep -o "\"$key\": \"[^\"]*\"" | head -1 | cut -d'"' -
f4 || echo ""
}

# Array to track created resources
declare -a CREATED_RESOURCES

# Initialize demo mode flag
DEMO_MODE=false

# Security: Validate input parameters
if [[ $# -gt 0 ]]; then
    echo "ERROR: This script does not accept parameters" >&2
    exit 1
fi

{
    echo "==================================================="
    echo "AWS Support CLI Tutorial"
    echo "==================================================="
    echo "This script demonstrates how to use AWS Support API"
    echo "Note: You must have a Business, Enterprise On-Ramp,"
    echo "or Enterprise Support plan to use the AWS Support API."
    echo "==================================================="
    echo ""
} | tee -a "$LOG_FILE"

# Step 1: Check available services
echo "Step 1: Checking available AWS Support services..." | tee -a "$LOG_FILE"
SERVICES_OUTPUT=$(log_cmd "aws support describe-services --language en" 2>&1) ||
 SERVICES_OUTPUT=""
check_error "$SERVICES_OUTPUT" $? "Failed to retrieve AWS Support services"

# If we're in demo mode, set default values
if [[ "$DEMO_MODE" == "true" ]]; then
    SERVICE_CODE="general-info"
    echo "Using demo service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
else
    # Extract a service code for demonstration using safer method
```

```
    SERVICE_CODE=$(extract_json_value "$SERVICES_OUTPUT" "code") ||
 SERVICE_CODE=""
    if [[ -z "$SERVICE_CODE" ]]; then
        SERVICE_CODE="general-info"
        echo "Using default service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
    else
        echo "Found service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
    fi
fi

# Step 2: Check available severity levels
echo "Step 2: Checking available severity levels..." | tee -a "$LOG_FILE"
if [[ "$DEMO_MODE" == "true" ]]; then
    demo_cmd "aws support describe-severity-levels --language en" "Check
 available severity levels"
    SEVERITY_CODE="low"
    echo "Using demo severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
else
    SEVERITY_OUTPUT=$(log_cmd "aws support describe-severity-levels --language
 en" 2>&1) || SEVERITY_OUTPUT=""
    check_error "$SEVERITY_OUTPUT" $? "Failed to retrieve severity levels"

    SEVERITY_CODE=$(extract_json_value "$SEVERITY_OUTPUT" "code") ||
 SEVERITY_CODE=""
    if [[ -z "$SEVERITY_CODE" ]]; then
        SEVERITY_CODE="low"
        echo "Using default severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
    else
        echo "Found severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
    fi
fi

# Step 3: Create a test support case
{
    echo ""
    echo "==================================================="
    echo "SUPPORT CASE CREATION"
    echo "==================================================="
} | tee -a "$LOG_FILE"

if [[ "$DEMO_MODE" == "true" ]]; then
    {
        echo "DEMO MODE: The following steps would create and manage a support
 case"
```

```
        echo "if you had a Business, Enterprise On-Ramp, or Enterprise Support
 plan."
        echo ""
    } | tee -a "$LOG_FILE"

    USER_EMAIL="example@example.com"

    demo_cmd "aws support create-case" "Create a support case"

    CASE_ID="case-12345678910-2013-c4c1d2bf33c5cf47"
    echo "Demo case ID: $CASE_ID" | tee -a "$LOG_FILE"

    demo_cmd "aws support describe-cases" "List support cases"
    demo_cmd "aws support add-communication-to-case" "Add communication to case"
    demo_cmd "aws support describe-communications" "View case communications"
    demo_cmd "aws support resolve-case" "Resolve the support case"

else
    echo "Creating a test support case..." | tee -a "$LOG_FILE"

    USER_EMAIL="example@example.com"
    CC_EMAIL_PARAM="--cc-email-addresses $USER_EMAIL"

    # Create the case
    CASE_OUTPUT=$(log_cmd "aws support create-case --subject 'AWS CLI Tutorial
 Test Case' --service-code '$SERVICE_CODE' --category-code 'using-aws' --
communication-body 'This is a test case created as part of an AWS CLI tutorial.'
 --severity-code '$SEVERITY_CODE' --language 'en' $CC_EMAIL_PARAM" 2>&1) ||
 CASE_OUTPUT=""

    check_error "$CASE_OUTPUT" $? "Failed to create support case"

    # Extract the case ID safely
    CASE_ID=$(extract_json_value "$CASE_OUTPUT" "caseId") || CASE_ID=""

    if [[ -n "$CASE_ID" ]]; then
        echo "Successfully created support case with ID: $CASE_ID" | tee -a
 "$LOG_FILE"
        CREATED_RESOURCES+=("Support Case: $CASE_ID")

        # Step 4: List the case we just created
        echo "" | tee -a "$LOG_FILE"
        echo "Step 4: Listing the support case we just created..." | tee -a
 "$LOG_FILE"
```

```
        CASES_OUTPUT=$(log_cmd "aws support describe-cases --case-id-
list '$CASE_ID' --include-resolved-cases false --language 'en'" 2>&1) ||
 CASES_OUTPUT=""

        check_error "$CASES_OUTPUT" $? "Failed to retrieve case details"

        # Step 5: Add a communication to the case
        echo "" | tee -a "$LOG_FILE"
        echo "Step 5: Adding a communication to the support case..." | tee -a
 "$LOG_FILE"
        COMM_OUTPUT=$(log_cmd "aws support add-communication-to-case --case-id
 '$CASE_ID' --communication-body 'This is an additional communication for the
 test case.' $CC_EMAIL_PARAM" 2>&1) || COMM_OUTPUT=""

        check_error "$COMM_OUTPUT" $? "Failed to add communication to case"

        # Step 6: View communications for the case
        echo "" | tee -a "$LOG_FILE"
        echo "Step 6: Viewing communications for the support case..." | tee -a
 "$LOG_FILE"
        COMMS_OUTPUT=$(log_cmd "aws support describe-communications --case-id
 '$CASE_ID' --language 'en'" 2>&1) || COMMS_OUTPUT=""

        check_error "$COMMS_OUTPUT" $? "Failed to retrieve case communications"

        # Step 7: Resolve the case
        {
            echo ""
            echo "==================================================="
            echo "CASE RESOLUTION"
            echo "==================================================="
            echo "Resolving the support case..."
        } | tee -a "$LOG_FILE"

        RESOLVE_OUTPUT=$(log_cmd "aws support resolve-case --case-id '$CASE_ID'"
 2>&1) || RESOLVE_OUTPUT=""

        check_error "$RESOLVE_OUTPUT" $? "Failed to resolve case"
        echo "Successfully resolved support case: $CASE_ID" | tee -a "$LOG_FILE"
    else
        echo "Could not extract case ID from the response." | tee -a "$LOG_FILE"
    fi
fi
```

```
# Display summary of created resources
{
    echo ""
    echo "====================================================="
    echo "TUTORIAL SUMMARY"
    echo "====================================================="
} | tee -a "$LOG_FILE"

if [[ "$DEMO_MODE" == "true" ]]; then
    {
        echo "This was a demonstration in DEMO MODE."
        echo "No actual AWS Support cases were created."
        echo "To use the AWS Support API, you need a Business, Enterprise On-
Ramp,"
        echo "or Enterprise Support plan."
    } | tee -a "$LOG_FILE"
else
    {
        echo "Resources created during this tutorial:"
        if [[ ${#CREATED_RESOURCES[@]} -eq 0 ]]; then
            echo "No resources were created."
        else
            for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}";
 do
                echo "- $resource"
            done
        fi
    } | tee -a "$LOG_FILE"
fi

{
    echo ""
    echo "Tutorial completed successfully!"
    echo "Log file: $LOG_FILE"
    echo "====================================================="
} | tee -a "$LOG_FILE"

# Display log file path to user
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddCommunicationToCase
  - CreateCase

- DescribeCases
- DescribeCommunications
- DescribeServices
- DescribeSeverityLevels
- ResolveCase

Exemplos de código para Systems Manager usando AWS SDKs

Os exemplos de código a seguir mostram como usar AWS Systems Manager com um kit de desenvolvimento de AWS software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do usuário do Systems Manager: mais informações sobre o Systems Manager.
- Referência da API Systems Manager: detalhes sobre todas as ações disponíveis do Systems Manager.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Systems Manager usando AWS SDKs
  - Hello Systems Manager
  - Aprenda os conceitos básicos do Systems Manager com um AWS SDK
  - Ações para o Systems Manager usando AWS SDKs

- Usar AddTagsToResource com uma CLI
- Usar CancelCommand com uma CLI
- Usar CreateActivation com uma CLI
- Usar CreateAssociation com uma CLI
- Usar CreateAssociationBatch com uma CLI
- Use CreateDocumentcom um AWS SDK ou CLI
- Use CreateMaintenanceWindowcom um AWS SDK ou CLI
- Use CreateOpsItemcom um AWS SDK ou CLI
- Usar CreatePatchBaseline com uma CLI
- Usar DeleteActivation com uma CLI
- Usar DeleteAssociation com uma CLI
- Use DeleteDocumentcom um AWS SDK ou CLI
- Use DeleteMaintenanceWindowcom um AWS SDK ou CLI
- Use DeleteOpsItemcom um AWS SDK
- Usar DeleteParameter com uma CLI
- Usar DeletePatchBaseline com uma CLI
- Usar DeregisterManagedInstance com uma CLI
- Usar DeregisterPatchBaselineForPatchGroup com uma CLI
- Usar DeregisterTargetFromMaintenanceWindow com uma CLI
- Usar DeregisterTaskFromMaintenanceWindow com uma CLI
- Usar DescribeActivations com uma CLI
- Usar DescribeAssociation com uma CLI
- Usar DescribeAssociationExecutionTargets com uma CLI
- Usar DescribeAssociationExecutions com uma CLI
- Usar DescribeAutomationExecutions com uma CLI
- Usar DescribeAutomationStepExecutions com uma CLI
- Usar DescribeAvailablePatches com uma CLI
- Use DescribeDocumentcom um AWS SDK ou CLI
- Usar DescribeDocumentPermission com uma CLI

- Usar DescribeEffectiveInstanceAssociations com uma CLI

- Usar DescribeEffectivePatchesForPatchBaseline com uma CLI
- Usar DescribeInstanceAssociationsStatus com uma CLI
- Usar DescribeInstanceInformation com uma CLI
- Usar DescribeInstancePatchStates com uma CLI
- Usar DescribeInstancePatchStatesForPatchGroup com uma CLI
- Usar DescribeInstancePatches com uma CLI
- Usar DescribeMaintenanceWindowExecutionTaskInvocations com uma CLI
- Usar DescribeMaintenanceWindowExecutionTasks com uma CLI
- Usar DescribeMaintenanceWindowExecutions com uma CLI
- Usar DescribeMaintenanceWindowTargets com uma CLI
- Usar DescribeMaintenanceWindowTasks com uma CLI
- Usar DescribeMaintenanceWindows com uma CLI
- Use DescribeOpsItemscom um AWS SDK ou CLI
- Use DescribeParameterscom um AWS SDK ou CLI
- Usar DescribePatchBaselines com uma CLI
- Usar DescribePatchGroupState com uma CLI
- Usar DescribePatchGroups com uma CLI
- Usar GetAutomationExecution com uma CLI
- Usar GetCommandInvocation com uma CLI
- Usar GetConnectionStatus com uma CLI
- Usar GetDefaultPatchBaseline com uma CLI
- Usar GetDeployablePatchSnapshotForInstance com uma CLI
- Usar GetDocument com uma CLI
- Usar GetInventory com uma CLI
- Usar GetInventorySchema com uma CLI
- Usar GetMaintenanceWindow com uma CLI
- Usar GetMaintenanceWindowExecution com uma CLI
- Usar GetMaintenanceWindowExecutionTask com uma CLI
- Use GetParametercom um AWS SDK ou CLI

- Usar GetParameterHistory com uma CLI

- Usar GetParameters com uma CLI
- Usar GetPatchBaseline com uma CLI
- Usar GetPatchBaselineForPatchGroup com uma CLI
- Usar ListAssociationVersions com uma CLI
- Usar ListAssociations com uma CLI
- Use ListCommandInvocationscom um AWS SDK ou CLI
- Usar ListCommands com uma CLI
- Usar ListComplianceItems com uma CLI
- Usar ListComplianceSummaries com uma CLI
- Usar ListDocumentVersions com uma CLI
- Usar ListDocuments com uma CLI
- Usar ListInventoryEntries com uma CLI
- Usar ListResourceComplianceSummaries com uma CLI
- Usar ListTagsForResource com uma CLI
- Usar ModifyDocumentPermission com uma CLI
- Usar PutComplianceItems com uma CLI
- Usar PutInventory com uma CLI
- Use PutParametercom um AWS SDK ou CLI
- Usar RegisterDefaultPatchBaseline com uma CLI
- Usar RegisterPatchBaselineForPatchGroup com uma CLI
- Usar RegisterTargetWithMaintenanceWindow com uma CLI
- Usar RegisterTaskWithMaintenanceWindow com uma CLI
- Usar RemoveTagsFromResource com uma CLI
- Use SendCommandcom um AWS SDK ou CLI
- Usar StartAutomationExecution com uma CLI
- Usar StartSession com uma CLI
- Usar StopAutomationExecution com uma CLI
- Usar UpdateAssociation com uma CLI
- Usar UpdateAssociationStatus com uma CLI

- Usar UpdateDocument com uma CLI

    - Usar UpdateDocumentDefaultVersion com uma CLI
    - Use UpdateMaintenanceWindowcom um AWS SDK ou CLI
    - Usar UpdateManagedInstanceRole com uma CLI
    - Use UpdateOpsItemcom um AWS SDK ou CLI
    - Usar UpdatePatchBaseline com uma CLI
- Cenários para o Systems Manager usando AWS SDKs
  - Como criar um serviço de contêiner para instâncias de máquinas virtuais
  - Introdução às máquinas virtuais
  - Como configurar o gerenciamento de sistemas

Exemplos básicos para o Systems Manager usando AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS Systems Manager com AWS SDKs.

Exemplos

- Hello Systems Manager
- Aprenda os conceitos básicos do Systems Manager com um AWS SDK
- Ações para o Systems Manager usando AWS SDKs
  - Usar AddTagsToResource com uma CLI
  - Usar CancelCommand com uma CLI
  - Usar CreateActivation com uma CLI
  - Usar CreateAssociation com uma CLI
  - Usar CreateAssociationBatch com uma CLI
  - Use CreateDocumentcom um AWS SDK ou CLI
  - Use CreateMaintenanceWindowcom um AWS SDK ou CLI
  - Use CreateOpsItemcom um AWS SDK ou CLI
  - Usar CreatePatchBaseline com uma CLI
  - Usar DeleteActivation com uma CLI
  - Usar DeleteAssociation com uma CLI
  - Use DeleteDocumentcom um AWS SDK ou CLI
  - Use DeleteMaintenanceWindowcom um AWS SDK ou CLI

- Use DeleteOpsItemcom um AWS SDK
- Usar DeleteParameter com uma CLI
- Usar DeletePatchBaseline com uma CLI
- Usar DeregisterManagedInstance com uma CLI
- Usar DeregisterPatchBaselineForPatchGroup com uma CLI
- Usar DeregisterTargetFromMaintenanceWindow com uma CLI
- Usar DeregisterTaskFromMaintenanceWindow com uma CLI
- Usar DescribeActivations com uma CLI
- Usar DescribeAssociation com uma CLI
- Usar DescribeAssociationExecutionTargets com uma CLI
- Usar DescribeAssociationExecutions com uma CLI
- Usar DescribeAutomationExecutions com uma CLI
- Usar DescribeAutomationStepExecutions com uma CLI
- Usar DescribeAvailablePatches com uma CLI
- Use DescribeDocumentcom um AWS SDK ou CLI
- Usar DescribeDocumentPermission com uma CLI
- Usar DescribeEffectiveInstanceAssociations com uma CLI
- Usar DescribeEffectivePatchesForPatchBaseline com uma CLI
- Usar DescribeInstanceAssociationsStatus com uma CLI
- Usar DescribeInstanceInformation com uma CLI
- Usar DescribeInstancePatchStates com uma CLI
- Usar DescribeInstancePatchStatesForPatchGroup com uma CLI
- Usar DescribeInstancePatches com uma CLI
- Usar DescribeMaintenanceWindowExecutionTaskInvocations com uma CLI
- Usar DescribeMaintenanceWindowExecutionTasks com uma CLI
- Usar DescribeMaintenanceWindowExecutions com uma CLI
- Usar DescribeMaintenanceWindowTargets com uma CLI
- Usar DescribeMaintenanceWindowTasks com uma CLI
- Usar DescribeMaintenanceWindows com uma CLI

- Use DescribeOpsItemscom um AWS SDK ou CLI

- Use DescribeParameterscom um AWS SDK ou CLI
- Usar DescribePatchBaselines com uma CLI
- Usar DescribePatchGroupState com uma CLI
- Usar DescribePatchGroups com uma CLI
- Usar GetAutomationExecution com uma CLI
- Usar GetCommandInvocation com uma CLI
- Usar GetConnectionStatus com uma CLI
- Usar GetDefaultPatchBaseline com uma CLI
- Usar GetDeployablePatchSnapshotForInstance com uma CLI
- Usar GetDocument com uma CLI
- Usar GetInventory com uma CLI
- Usar GetInventorySchema com uma CLI
- Usar GetMaintenanceWindow com uma CLI
- Usar GetMaintenanceWindowExecution com uma CLI
- Usar GetMaintenanceWindowExecutionTask com uma CLI
- Use GetParametercom um AWS SDK ou CLI
- Usar GetParameterHistory com uma CLI
- Usar GetParameters com uma CLI
- Usar GetPatchBaseline com uma CLI
- Usar GetPatchBaselineForPatchGroup com uma CLI
- Usar ListAssociationVersions com uma CLI
- Usar ListAssociations com uma CLI
- Use ListCommandInvocationscom um AWS SDK ou CLI
- Usar ListCommands com uma CLI
- Usar ListComplianceItems com uma CLI
- Usar ListComplianceSummaries com uma CLI
- Usar ListDocumentVersions com uma CLI
- Usar ListDocuments com uma CLI
- Usar ListInventoryEntries com uma CLI

- Usar ListResourceComplianceSummaries com uma CLI

- Usar ListTagsForResource com uma CLI
- Usar ModifyDocumentPermission com uma CLI
- Usar PutComplianceItems com uma CLI
- Usar PutInventory com uma CLI
- Use PutParametercom um AWS SDK ou CLI
- Usar RegisterDefaultPatchBaseline com uma CLI
- Usar RegisterPatchBaselineForPatchGroup com uma CLI
- Usar RegisterTargetWithMaintenanceWindow com uma CLI
- Usar RegisterTaskWithMaintenanceWindow com uma CLI
- Usar RemoveTagsFromResource com uma CLI
- Use SendCommandcom um AWS SDK ou CLI
- Usar StartAutomationExecution com uma CLI
- Usar StartSession com uma CLI
- Usar StopAutomationExecution com uma CLI
- Usar UpdateAssociation com uma CLI
- Usar UpdateAssociationStatus com uma CLI
- Usar UpdateDocument com uma CLI
- Usar UpdateDocumentDefaultVersion com uma CLI
- Use UpdateMaintenanceWindowcom um AWS SDK ou CLI
- Usar UpdateManagedInstanceRole com uma CLI
- Use UpdateOpsItemcom um AWS SDK ou CLI
- Usar UpdatePatchBaseline com uma CLI

Hello Systems Manager

Os exemplos de código a seguir mostram como começar a usar o Systems Manager.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <awsAccount>

                Where:
                    awsAccount - Your AWS Account number.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        listDocuments(ssmClient, awsAccount);
    }
```

```
    /*
    This code automatically fetches the next set of results using the `nextToken`
 and
    stops once the desired maxResults (20 in this case) have been reached.
    */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                    )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
 System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- Para obter detalhes da API, consulte ListDocumentsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { paginateListDocuments, SSMClient } from "@aws-sdk/client-ssm";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new SSMClient();
  const listDocumentsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListDocuments({ client }, { MaxResults: 5 });
    for await (const page of paginator) {
      listDocumentsPaginated.push(...page.DocumentIdentifiers);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }

  for (const { Name, DocumentFormat, CreatedDate } of listDocumentsPaginated) {
    console.log(`${Name} - ${DocumentFormat} - ${CreatedDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte ListDocumentsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
from botocore.exceptions import ClientError


def hello_systems_manager(ssm_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Systems Manager
    client and list the first 5 documents in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ssm_client: A Boto3 AWS Systems Manager Client object. This object
 wraps
                               the low-level AWS Systems Manager service API.
    """
    print("Hello, AWS Systems Manager! Let's list some of your documents:\n")

    paginator = ssm_client.get_paginator("list_documents")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 5})
    for page in page_iterator:
        for document in page["DocumentIdentifiers"]:
            print(f"  {document['Name']}")


if __name__ == "__main__":
    try:
        hello_systems_manager(boto3.client("ssm"))
    except ClientError as err:
```

```
        print("Hello systems manager had an error.")
        print(err.response["Error"]["Code"])
        print(err.response["Error"]["Message"])
```

- Para obter detalhes da API, consulte a ListDocumentsReferência da API AWS SDK for Python (Boto3).

Aprenda os conceitos básicos do Systems Manager com um AWS SDK

Os exemplos de código a seguir mostram como:

- Criar uma janela de manutenção.
- Modificar a programação da janela de manutenção.
- Criar um documento.
- Enviar um comando para uma instância do EC2 especificada.
- Crie um OpsItem.
- Atualize e resolva OpsItem o.
- Exclua a janela de manutenção OpsItem e o documento.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Map;
import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
```

```
    private static final String ROLES_STACK = "SsmStack3`1";

    public static void main(String[] args) {
        String usage = """
            Usage:
              <title> <source> <category> <severity>

            Where:
                title - The title of the parameter (default is Disk Space Alert).
                source - The source of the parameter (default is EC2).
                category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
                severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
        """;

        Scanner scanner = new Scanner(System.in);
        SSMActions actions = new SSMActions();
        String documentName;
        String windowName;

        System.out.println("Use AWS CloudFormation to create the EC2 instance
that is required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String instanceId = stackOutputs.get("InstanceId");
        System.out.println("The Instance ID: " + instanceId +" was created.");
        String title = "Disk Space Alert" ;
        String source = "EC2" ;
        String category = "Availability" ;
        String severity = "2" ;

        System.out.println(DASHES);
        System.out.println("""
                Welcome to the AWS Systems Manager SDK Basics scenario.
                This Java program demonstrates how to interact with AWS Systems
Manager using the AWS SDK for Java (v2).
                AWS Systems Manager is the operations hub for your AWS
applications and resources and a secure end-to-end management solution.
                The program's primary functionalities include creating a
maintenance window, creating a document, sending a command to a document,
```

```
                listing documents, listing commands, creating an OpsItem,
modifying an OpsItem, and deleting AWS SSM resources.
                Upon completion of the program, all AWS resources are cleaned up.
                Let's get started...

                """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("1. Create an SSM maintenance window.");
        System.out.println("Please enter the maintenance window name (default is
ssm-maintenance-window):");
        String win = scanner.nextLine();
        windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
        String winId = null;
        try {
            winId = actions.createMaintenanceWindow(windowName);
            waitForInputToContinue(scanner);
            System.out.println("The maintenance window ID is: " + winId);
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM maintenance window already exists.
Retrieving existing window ID...");
            String existingWinId = actions.createMaintenanceWindow(windowName);
            System.out.println("Existing window ID: " + existingWinId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("2. Modify the maintenance window by changing the
schedule");
        waitForInputToContinue(scanner);
        try {
            actions.updateSSMMaintenanceWindow(winId, windowName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM maintenance window was successfully
updated");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
```

```
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("3. Create an SSM document that defines the actions
that Systems Manager performs on your managed nodes.");
        System.out.println("Please enter the document name (default is
ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        try {
            actions.createSSMDoc(documentName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM document was successfully created");
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM document already exists. Moving on");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("4. Now we are going to run a command on an EC2
instance");
        waitForInputToContinue(scanner);
        String commandId="";
        try {
            commandId = actions.sendSSMCommand(documentName, instanceId);
            waitForInputToContinue(scanner);
            System.out.println("The command was successfully sent. Command ID: "
+ commandId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
        } catch (InterruptedException e) {
            System.err.println("Thread was interrupted: " + e.getMessage());
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
```

```
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("5. Lets get the time when the specific command was
sent to the specific managed node");
        waitForInputToContinue(scanner);
        try {
            actions.displayCommands(commandId);
            System.out.println("The command invocations were successfully
displayed.");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
             6. Now we will create an SSM OpsItem.
             A SSM OpsItem is a feature provided by Amazon's Systems Manager
(SSM) service.
             It is a type of operational data item that allows you to manage and
track various operational issues,
             events, or tasks within your AWS environment.

             You can create OpsItems to track and manage operational issues as
they arise.
             For example, you could create an OpsItem whenever your application
detects a critical error
             or an anomaly in your infrastructure.
            """);

        waitForInputToContinue(scanner);
        String opsItemId;
        try {
            opsItemId = actions.createSSMOpsItem(title, source, category,
severity);
            System.out.println(opsItemId + " was created");
        } catch (SsmException e) {
```

```
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Now we will update the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        String description = "An update to "+opsItemId ;
        try {
            actions.updateOpsItem(opsItemId, title, description);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("8. Now we will get the status of the SSM OpsItem
 "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.describeOpsItems(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("9. Now we will resolve the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.resolveOpsItem(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
```

```
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("10. Would you like to delete the AWS Systems Manager
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            waitForInputToContinue(scanner);
            try {
                actions.deleteMaintenanceWindow(winId);
                actions.deleteDoc(documentName);
            } catch (SsmException e) {
                System.err.println("SSM error: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("Unexpected error: " + e.getMessage());
                return;
            }
        } else {
            System.out.println("The AWS Systems Manager resources will not be
deleted");
        }
        System.out.println(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        System.out.println("This concludes the AWS Systems Manager SDK Basics
scenario.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
```

```
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper para métodos do SDK do Systems Manager.

```
public class SSMActions {

    private static SsmAsyncClient ssmAsyncClient;

    private static SsmAsyncClient getAsyncClient() {
        if (ssmAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryPolicy(RetryPolicy.builder()
                            .numRetries(3)
                            .build())
                    .build();

            ssmAsyncClient = SsmAsyncClient.builder()
                    .region(Region.US_WEST_2)
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ssmAsyncClient;
    }

    /**
```

```
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
```

```
    * @param winId The ID of the Maintenance Window to delete.
    * <p>
    * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteMaintenanceWindow(String winId) {
       DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
               .windowId(winId)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteMaintenanceWindow(windowRequest)
                   .thenAccept(response -> {
                       System.out.println("The maintenance window was
successfully deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });

       try {
           future.join();
       } catch (CompletionException ex) {
           throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
       }
   }

   /**
    * Resolves an AWS SSM OpsItem asynchronously.
    *
```

```
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
```

```
    * This method initiates an asynchronous request to describe SSM OpsItems.
    * If the request is successful, it prints the title and status of each
OpsItem.
    * If an exception occurs, it handles the error appropriately.
    */
   public void describeOpsItems(String key) {
       OpsItemFilter filter = OpsItemFilter.builder()
               .key(OpsItemFilterKey.OPS_ITEM_ID)
               .values(key)
               .operator(OpsItemFilterOperator.EQUAL)
               .build();

       DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
               .maxResults(10)
               .opsItemFilters(filter)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().describeOpsItems(itemsRequest)
                   .thenAccept(itemsResponse -> {
                       List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                       for (OpsItemSummary item : items) {
                           System.out.println("The item title is " +
item.title() + " and the status is " + item.status().toString());
                       }
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });

       try {
           future.join();
```

```
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Updates the AWS SSM OpsItem asynchronously.
     *
     * @param opsItemId The ID of the OpsItem to update.
     * @param title The new title of the OpsItem.
     * @param description The new description of the OpsItem.
     * <p>
     * This method initiates an asynchronous request to update an SSM OpsItem.
     * If the request is successful, it completes without returning a value.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateOpsItem(String opsItemId, String title, String description)
{
        Map<String, OpsItemDataValue> operationalData = new HashMap<>();
        operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
        operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());

        CompletableFuture<Void> future =
getOpsItem(opsItemId).thenCompose(opsItem -> {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                    .opsItemId(opsItemId)
                    .title(title)
                    .operationalData(operationalData)
                    .status(opsItem.statusAsString())
                    .description(description)
                    .build();

            return getAsyncClient().updateOpsItem(request).thenAccept(response ->
{
                System.out.println(opsItemId + " updated successfully.");
            }).exceptionally(ex -> {
                throw new CompletionException(ex);
            });
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
```

```
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }


    private static CompletableFuture<OpsItem> getOpsItem(String opsItemId) {
        GetOpsItemRequest request =
GetOpsItemRequest.builder().opsItemId(opsItemId).build();
        return
getAsyncClient().getOpsItem(request).thenApply(GetOpsItemResponse::opsItem);
    }

    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
```

```
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }

    /**
     * Displays the date and time when the specific command was invoked.
     *
     * @param commandId The ID of the command to describe.
     * <p>
     * This method initiates an asynchronous request to list command invocations
and prints the date and time of each command invocation.
     * If an exception occurs, it handles the error appropriately.
     */
    public void displayCommands(String commandId) {
        ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
                .commandId(commandId)
                .build();

        CompletableFuture<ListCommandInvocationsResponse> future =
getAsyncClient().listCommandInvocations(commandInvocationsRequest);
        future.thenAccept(response -> {
            List<CommandInvocation> commandList = response.commandInvocations();
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
            for (CommandInvocation invocation : commandList) {
                System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
```

```
            }
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }

    /**
     * Sends a SSM command to a managed node asynchronously.
     *
     * @param documentName The name of the document to use.
     * @param instanceId The ID of the instance to send the command to.
     * @return The command ID.
     * <p>
     * This method initiates asynchronous requests to send a SSM command to a
managed node.
     * It waits until the document is active, sends the command, and checks the
command execution status.
     */
    public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
        // Before we use Document to send a command - make sure it is active.
        CompletableFuture<Void> documentActiveFuture =
CompletableFuture.runAsync(() -> {
            boolean isDocumentActive = false;
            DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                    .name(documentName)
                    .build();

            while (!isDocumentActive) {
                CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
                String documentStatus =
response.join().document().statusAsString();
                if (documentStatus.equals("Active")) {
                    System.out.println("The SSM document is active and ready to
use.");
                    isDocumentActive = true;
                } else {
```

```
                    System.out.println("The SSM document is not active. Status: "
+ documentStatus);
                    try {
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        documentActiveFuture.join();

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse>
invocationFuture = getAsyncClient().getCommandInvocation(invocationRequest);
```

```
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution
successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }

    /**
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
```

```
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

```
    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }

    /**
     * Creates an SSM maintenance window asynchronously.
     *
```

```
    * @param winName The name of the maintenance window.
    * @return The ID of the created or existing maintenance window.
    * <p>
    * This method initiates an asynchronous request to create an SSM maintenance
window.
    * If the request is successful, it prints the maintenance window ID.
    * If an exception occurs, it handles the error appropriately.
    */
   public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
       CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
               .name(winName)
               .description("This is my maintenance window")
               .allowUnassociatedTargets(true)
               .duration(2)
               .cutoff(1)
               .schedule("cron(0 10 ? * MON-FRI *)")
               .build();

       CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
       final String[] windowId = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String maintenanceWindowId = response.windowId();
               System.out.println("The maintenance window id is " +
maintenanceWindowId);
               windowId[0] = maintenanceWindowId;
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof DocumentAlreadyExistsException) {
                   throw new CompletionException(cause);
               } else if (cause instanceof SsmException) {
                   throw new CompletionException(cause);
               } else {
                   throw new RuntimeException(cause);
               }
           }
       }).join();

       if (windowId[0] == null) {
           MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
```

```
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
 DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture
 = getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
 describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
 CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
 windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow

- ListCommandInvocations
- SendCommand
- UpdateOpsItem

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { fileURLToPath } from "node:url";
import {
  CreateDocumentCommand,
  CreateMaintenanceWindowCommand,
  CreateOpsItemCommand,
  DeleteDocumentCommand,
  DeleteMaintenanceWindowCommand,
  DeleteOpsItemCommand,
  DescribeOpsItemsCommand,
  DocumentAlreadyExists,
  OpsItemStatus,
  waitUntilCommandExecuted,
  CancelCommandCommand,
  paginateListCommandInvocations,
  SendCommandCommand,
  UpdateMaintenanceWindowCommand,
  UpdateOpsItemCommand,
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";
```

```
/**
 * @typedef {{
 *   ssmClient: import('@aws-sdk/client-ssm').SSMClient,
 *   documentName?: string
 *   maintenanceWindow?: string
 *   winId?: int
 *   ec2InstanceId?: string
 *   requestedDateTime?: Date
 *   opsItemId?: string
 *   askToDeleteResources?: boolean
 * }} State
 */

const defaultMaintenanceWindow = "ssm-maintenance-window";
const defaultDocumentName = "ssmdocument";
// The timeout duration is highly dependent on the specific setup and environment
 necessary. This example handles only the most common error cases, and uses a
 much shorter duration than most productions systems would use.
const COMMAND_TIMEOUT_DURATION_SECONDS = 30; // 30 seconds

const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `Welcome to the AWS Systems Manager SDK Getting Started scenario.
    This program demonstrates how to interact with Systems Manager using the AWS
 SDK for JavaScript V3.
    Systems Manager is the operations hub for your AWS applications and resources
 and a secure end-to-end management solution.
    The program's primary functions include creating a maintenance window,
 creating a document, sending a command to a document,
    listing documents, listing commands, creating an OpsItem, modifying an
 OpsItem, and deleting Systems Manager resources.
    Upon completion of the program, all AWS resources are cleaned up.
    Let's get started...`,
  { header: true },
);

const createMaintenanceWindow = new ScenarioOutput(
  "createMaintenanceWindow",
  "Step 1: Create a Systems Manager maintenance window.",
);
```

```
const getMaintenanceWindow = new ScenarioInput(
  "maintenanceWindow",
  "Please enter the maintenance window name:",
  { type: "input", default: defaultMaintenanceWindow },
);

export const sdkCreateMaintenanceWindow = new ScenarioAction(
  "sdkCreateMaintenanceWindow",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          Schedule: "cron(0 10 ? * MON-FRI *)", //The schedule of the maintenance
 window in the form of a cron or rate expression.
          Duration: 2, //The duration of the maintenance window in hours.
          Cutoff: 1, //The number of hours before the end of the maintenance
 window that Amazon Web Services Systems Manager stops scheduling new tasks for
 execution.
          AllowUnassociatedTargets: true, //Allow the maintenance window to run
 on managed nodes, even if you haven't registered those nodes as targets.
        }),
      );
      state.winId = response.WindowId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const modifyMaintenanceWindow = new ScenarioOutput(
  "modifyMaintenanceWindow",
  "Modify the maintenance window by changing the schedule.",
);

const sdkModifyMaintenanceWindow = new ScenarioAction(
  "sdkModifyMaintenanceWindow",
  async (/** @type {State} */ state) => {
```

```
    try {
      await state.ssmClient.send(
        new UpdateMaintenanceWindowCommand({
          WindowId: state.winId,
          Schedule: "cron(0 0 ? * MON *)",
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while modifying the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const createSystemsManagerActions = new ScenarioOutput(
  "createSystemsManagerActions",
  "Create a document that defines the actions that Systems Manager performs on
 your EC2 instance.",
);

const getDocumentName = new ScenarioInput(
  "documentName",
  "Please enter the document: ",
  { type: "input", default: defaultDocumentName },
);

const sdkCreateSSMDoc = new ScenarioAction(
  "sdkCreateSSMDoc",
  async (/** @type {State} */ state) => {
    const contentData = `{
                "schemaVersion": "2.2",
                "description": "Run a simple shell command",
                "mainSteps": [
                    {
                        "action": "aws:runShellScript",
                        "name": "runEchoCommand",
                        "inputs": {
                          "runCommand": [
                            "echo 'Hello, world!'"
                          ]
```

```
                        }
                    }
                ]
            }`;
    try {
      await state.ssmClient.send(
        new CreateDocumentCommand({
          Content: contentData,
          Name: state.documentName,
          DocumentType: "Command",
        }),
      );
    } catch (caught) {
      console.log(`Exception type: (${typeof caught})`);
      if (caught instanceof DocumentAlreadyExists) {
        console.log("Document already exists. Continuing...\n");
      } else {
        console.error(caught.message);
        console.log(
          `An error occurred while creating the document. Please fix the error
 and try again. Error message: ${caught.message}`,
        );
        throw caught;
      }
    }
  },
);

const ec2HelloWorld = new ScenarioOutput(
  "ec2HelloWorld",
  `Now you have the option of running a command on an EC2 instance that echoes
 'Hello, world!'. In order to run this command, you must provide the instance ID
 of a Linux EC2 instance. If you do not already have a running Linux EC2 instance
 in your account, you can create one using the AWS console. For information
 about creating an EC2 instance, see https://docs.aws.amazon.com/AWSEC2/latest/
UserGuide/ec2-launch-instance-wizard.html.`,
);

const enterIdOrSkipEC2HelloWorld = new ScenarioInput(
  "enterIdOrSkipEC2HelloWorld",
  "Enter your EC2 InstanceId or press enter to skip this step: ",
  { type: "input", default: "" },
);
```

```
const sdkEC2HelloWorld = new ScenarioAction(
  "sdkEC2HelloWorld",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new SendCommandCommand({
          DocumentName: state.documentName,
          InstanceIds: [state.ec2InstanceId],
          TimeoutSeconds: COMMAND_TIMEOUT_DURATION_SECONDS,
        }),
      );
      state.CommandId = response.Command.CommandId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while sending the command. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const sdkGetCommandTime = new ScenarioAction(
  "sdkGetCommandTime",
  async (/** @type {State} */ state) => {
    const listInvocationsPaginated = [];
    console.log(
      "Let's get the time when the specific command was sent to the specific
 managed node.",
    );

    console.log(
      `First, we'll wait for the command to finish executing. This may take up to
 ${COMMAND_TIMEOUT_DURATION_SECONDS} seconds.`,
    );
    const commandExecutedResult = waitUntilCommandExecuted(
      { client: state.ssmClient },
      {
        CommandId: state.CommandId,
```

```
        InstanceId: state.ec2InstanceId,
      },
    );
    // This is necessary because the TimeoutSeconds of SendCommandCommand is only
for the delivery, not execution.
    try {
      await new Promise((_, reject) =>
        setTimeout(
          reject,
          COMMAND_TIMEOUT_DURATION_SECONDS * 1000,
          new Error("Command Timed Out"),
        ),
      );
    } catch (caught) {
      if (caught.message === "Command Timed Out") {
        commandExecutedResult.state = "TIMED_OUT";
      } else {
        throw caught;
      }
    }

    if (commandExecutedResult.state !== "SUCCESS") {
      console.log(
        `The command with id: ${state.CommandId} did not execute in the allotted
time. Canceling command.`,
      );
      state.ssmClient.send(
        new CancelCommandCommand({
          CommandId: state.CommandId,
        }),
      );
      state.enterIdOrSkipEC2HelloWorld === "";
      return;
    }

    for await (const page of paginateListCommandInvocations(
      { client: state.ssmClient },
      { CommandId: state.CommandId },
    )) {
      listInvocationsPaginated.push(...page.CommandInvocations);
    }
    /**
     * @type {import('@aws-sdk/client-ssm').CommandInvocation}
     */
```

```
    const commandInvocation = listInvocationsPaginated.shift(); // Because the
 call was made with CommandId, there's only one result, so shift it off.
    state.requestedDateTime = commandInvocation.RequestedDateTime;

    console.log(
      `The command invocation happened at: ${state.requestedDateTime}.`,
    );
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const createSSMOpsItem = new ScenarioOutput(
  "createSSMOpsItem",
  `Now we will create a Systems Manager OpsItem. An OpsItem is a feature provided
 by the Systems Manager service. It is a type of operational data item that
 allows you to manage and track various operational issues, events, or tasks
 within your AWS environment.
You can create OpsItems to track and manage operational issues as they arise.
 For example, you could create an OpsItem whenever your application detects a
 critical error or an anomaly in your infrastructure.`,
);

const sdkCreateSSMOpsItem = new ScenarioAction(
  "sdkCreateSSMOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateOpsItemCommand({
          Description: "Created by the System Manager Javascript API",
          Title: "Disk Space Alert",
          Source: "EC2",
          Category: "Performance",
          Severity: "2",
        }),
      );
      state.opsItemId = response.OpsItemId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
```

```
      );
      throw caught;
    }
  },
);

const updateOpsItem = new ScenarioOutput(
  "updateOpsItem",
  (/** @type {State} */ state) =>
    `Now we will update the OpsItem: ${state.opsItemId}`,
);

const sdkUpdateOpsItem = new ScenarioAction(
  "sdkUpdateOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Description: `An update to ${state.opsItemId}`,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const getOpsItemStatus = new ScenarioOutput(
  "getOpsItemStatus",
  (/** @type {State} */ state) =>
    `Now we will get the status of the OpsItem: ${state.opsItemId}`,
);

const sdkOpsItemStatus = new ScenarioAction(
  "sdkGetOpsItemStatus",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
```

```
        new DescribeOpsItemsCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      state.opsItemStatus = response.OpsItemStatus;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while describing the ops item. Please fix the error
 and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const resolveOpsItem = new ScenarioOutput(
  "resolveOpsItem",
  (/** @type {State} */ state) =>
    `Now we will resolve the OpsItem: ${state.opsItemId}`,
);

const sdkResolveOpsItem = new ScenarioAction(
  "sdkResolveOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Status: OpsItemStatus.RESOLVED,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
```

```
  "askToDeleteResources",
  "Would you like to delete the Systems Manager resources created during this
 example run?",
  { type: "confirm" },
);

const confirmDeleteChoice = new ScenarioOutput(
  "confirmDeleteChoice",
  (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You chose to delete the resources.";
    }
    return "The Systems Manager resources will not be deleted. Please delete them
 manually to avoid charges.";
  },
);

export const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    try {
      await state.ssmClient.send(
        new DeleteOpsItemCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      console.log(`The ops item: ${state.opsItemId} was successfully deleted.`);
    } catch (caught) {
      console.log(
        `There was a problem deleting the ops item: ${state.opsItemId}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          WindowId: state.winId,
        }),
      );
      console.log(
        `The maintenance window: ${state.maintenanceWindow} was successfully
 deleted.`,
```

```
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the maintenance window: ${state.opsItemId}.
 Please delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteDocumentCommand({
          Name: state.documentName,
        }),
      );
      console.log(
        `The document: ${state.documentName} was successfully deleted.`,
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the document: ${state.documentName}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the Systems Manager Basics scenario for the AWS Javascript SDK
 v3. Thank you!",
);

const myScenario = new Scenario(
  "SSM Basics",
  [
    greet,
    pressEnter,
    createMaintenanceWindow,
    getMaintenanceWindow,
    sdkCreateMaintenanceWindow,
    modifyMaintenanceWindow,
    pressEnter,
    sdkModifyMaintenanceWindow,
```

```
    createSystemsManagerActions,
    getDocumentName,
    sdkCreateSSMDoc,
    ec2HelloWorld,
    enterIdOrSkipEC2HelloWorld,
    sdkEC2HelloWorld,
    sdkGetCommandTime,
    pressEnter,
    createSSMOpsItem,
    pressEnter,
    sdkCreateSSMOpsItem,
    updateOpsItem,
    pressEnter,
    sdkUpdateOpsItem,
    getOpsItemStatus,
    pressEnter,
    sdkOpsItemStatus,
    resolveOpsItem,
    pressEnter,
    sdkResolveOpsItem,
    askToDeleteResources,
    confirmDeleteChoice,
    sdkDeleteResources,
    goodbye,
  ],
  { ssmClient: new SSMClient({}) },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
```

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class SystemsManagerScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Systems Manager."""

    def __init__(self, document_wrapper, maintenance_window_wrapper,
 ops_item_wrapper):
        """
        :param document_wrapper: An object that wraps Systems Manager document
 functions.
        :param maintenance_window_wrapper: An object that wraps Systems Manager
 maintenance window functions.
        :param ops_item_wrapper: An object that wraps Systems Manager OpsItem
 functions.
        """
```

```
        self.document_wrapper = document_wrapper
        self.maintenance_window_wrapper = maintenance_window_wrapper
        self.ops_item_wrapper = ops_item_wrapper

    def run(self):
        """Demonstrates how to use the AWS SDK for Python (Boto3) to get started
 with Systems Manager."""
        try:
            print("-" * 88)
            print(
                """
Welcome to the AWS Systems Manager SDK Getting Started scenario.
This program demonstrates how to interact with Systems Manager using the AWS SDK
 for Python (Boto3).
Systems Manager is the operations hub for your AWS applications and resources and
 a secure end-to-end management
solution. The program's primary functions include creating a maintenance window,
 creating a document, sending a
command to a document, listing documents, listing commands, creating an OpsItem,
 modifying an OpsItem, and deleting
Systems Manager resources. Upon completion of the program, all AWS resources are
 cleaned up.
Let's get started..."""
            )
            q.ask("Please hit Enter")

            print("-" * 88)
            print("Create a Systems Manager maintenance window.")
            maintenance_window_name = q.ask(
                "Please enter the maintenance window name (default is ssm-
maintenance-window):",
            )
            if not maintenance_window_name:
                maintenance_window_name = "ssm-maintenance-window"

            self.maintenance_window_wrapper.create(
                name=maintenance_window_name,
                schedule="cron(0 10 ? * MON-FRI *)",
                duration=2,
                cutoff=1,
                allow_unassociated_targets=True,
            )

            print("-" * 88)
```

```
            print("Modify the maintenance window by changing the schedule")
            q.ask("Please hit Enter")

            self.maintenance_window_wrapper.update(
                name=maintenance_window_name,
                schedule="cron(0 0 ? * MON *)",
                duration=24,
                cutoff=1,
                allow_unassociated_targets=True,
                enabled=True,
            )

            print("-" * 88)
            print(
                "Create a document that defines the actions that Systems Manager
 performs on your EC2 instance."
            )
            document_name = q.ask(
                "Please enter the document name (default is ssmdocument):"
            )

            if not document_name:
                document_name = "ssmdocument"

            self.document_wrapper.create(
                name=document_name,
                content="""
{
    "schemaVersion": "2.2",
    "description": "Run a simple shell command",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runEchoCommand",
            "inputs": {
              "runCommand": [
                "echo 'Hello, world!'"
              ]
            }
        }
    ]
}
            """,
            )
```

```
            self.document_wrapper.wait_until_active()

            print(
                """
Now you have the option of running a command on an EC2 instance that echoes
 'Hello, world!'.
In order to run this command, you must provide the instance ID of a Linux EC2
 instance. If you do
not already have a running Linux EC2 instance in your account, you can create one
 using the AWS console.
For information about creating an EC2 instance, see
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-instance-
wizard.html.
            """
            )

            if q.ask(
                "Would you like to run a command on an EC2 instance? (y/n)",
                q.is_yesno,
            ):
                instance_id = q.ask(
                    "Please enter the instance ID of the EC2 instance:",
 q.non_empty
                )
                command_id = self.document_wrapper.send_command(
                    instance_ids=[instance_id]
                )

                self.document_wrapper.wait_command_executed(
                    command_id=command_id, instance_id=instance_id
                )

                print("-" * 88)
                print(
                    "Lets get the time when the specific command was sent to the
 specific managed node"
                )
                q.ask("Please hit Enter")


 self.document_wrapper.list_command_invocations(instance_id=instance_id)

            print("-" * 88)
```

```
            print("-" * 88)
            print(
                """
Now we will create a  Systems Manager OpsItem.
An OpsItem is a feature provided by the Systems Manager service.
It is a type of operational data item that allows you to manage and track various
 operational issues,
events, or tasks within your AWS environment.

You can create OpsItems to track and manage operational issues as they arise.
For example, you could create an OpsItem whenever your application detects a
 critical error
or an anomaly in your infrastructure.
            """
            )
            q.ask("Please hit Enter")

            self.ops_item_wrapper.create(
                title="Disk Space Alert",
                description="Created by the Systems Manager Python (Boto3) API",
                source="EC2",
                category="Performance",
                severity="2",
            )

            print("-" * 88)
            print("-" * 88)
            print(f"Now we will update  the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(
                title="Disk Space Alert",
                description=f"An update to {self.ops_item_wrapper.id}",
            )

            print(
                f"Now we will get the status of the OpsItem
 {self.ops_item_wrapper.id}"
            )
            q.ask("Please hit Enter")

            # It may take a second for the ops item to be available
            counter = 0
            while not self.ops_item_wrapper.describe() and counter < 5:
```

```
                counter += 1
                time.sleep(1)

            print(f"Now we will resolve the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(status="Resolved")

            print("-" * 88)
            print("-" * 88)
            if q.ask(
                "Would you like to delete the Systems Manager resources? (y/n)",
                q.is_yesno,
            ):
                print("You selected to delete the resources.")
                self.cleanup()
            else:
                print("The Systems Manager resources will not be deleted")

            print("-" * 88)
            print("This concludes the Systems Manager SDK Getting Started
 scenario.")
            print("-" * 88)

        except Exception:
            self.cleanup()
            raise

    def cleanup(self):
        self.maintenance_window_wrapper.delete()
        self.ops_item_wrapper.delete()
        self.document_wrapper.delete()


if __name__ == "__main__":
    try:
        scenario = SystemsManagerScenario(
            DocumentWrapper.from_client(),
            MaintenanceWindowWrapper.from_client(),
            OpsItemWrapper.from_client(),
        )
        scenario.run()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que englobe as ações de documentos e comandos.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
TimeoutSeconds=3600
            )
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def describe(self):
        """
        Describes the document.

        :return: Document status.
        """
        try:
            response = self.ssm_client.describe_document(Name=self.name)
            return response["Document"]["Status"]
        except ClientError as err:
            logger.error(
                "Couldn't get %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def wait_until_active(self, max_attempts=20, delay=5):
        """
        Waits until the document is active.

        :param max_attempts: The maximum number of attempts for checking the
status.
        :param delay: The delay in seconds between each check.
        """
        attempt = 0
        status = ""
        while attempt <= max_attempts:
            status = self.describe()
            if status == "Active":
                break
            attempt += 1
            time.sleep(delay)

        if status != "Active":
            logger.error("Document is not active.")
        else:
            logger.info("Document is active.")

    def wait_command_executed(self, command_id, instance_id):
```

```
        """
        Waits until the command is executed on the instance.

        :param command_id: The ID of the command.
        :param instance_id: The ID of the instance.
        """

        waiter = self.ssm_client.get_waiter("command_executed")
        waiter.wait(CommandId=command_id, InstanceId=instance_id)

    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

Defina uma classe que englobe as ações de itens de operação.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem

        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
```

```
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
 OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self):
        """
```

```
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator":
"Equal"}
                ]
            ):
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
                print(
                    f"The item title is {item['Title']} and the status is
{item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
            args["Description"] = description
        if status is not None:
```

```
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Defina uma classe que englobe as ações de janelas de manutenção.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff,
 allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
```

```
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
```

```
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(
        self, name, enabled, schedule, duration, cutoff,
allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on
managed nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
        except ParamValidationError as error:
```

```
            logger.error(
                "Parameter validation error when trying to update maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Ações para o Systems Manager usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Systems Manager com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Systems Manager e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Systems Manager usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS Systems Manager.

Exemplos

- Usar AddTagsToResource com uma CLI
- Usar CancelCommand com uma CLI
- Usar CreateActivation com uma CLI
- Usar CreateAssociation com uma CLI
- Usar CreateAssociationBatch com uma CLI
- Use CreateDocumentcom um AWS SDK ou CLI
- Use CreateMaintenanceWindowcom um AWS SDK ou CLI
- Use CreateOpsItemcom um AWS SDK ou CLI
- Usar CreatePatchBaseline com uma CLI
- Usar DeleteActivation com uma CLI
- Usar DeleteAssociation com uma CLI
- Use DeleteDocumentcom um AWS SDK ou CLI
- Use DeleteMaintenanceWindowcom um AWS SDK ou CLI
- Use DeleteOpsItemcom um AWS SDK
- Usar DeleteParameter com uma CLI
- Usar DeletePatchBaseline com uma CLI
- Usar DeregisterManagedInstance com uma CLI
- Usar DeregisterPatchBaselineForPatchGroup com uma CLI
- Usar DeregisterTargetFromMaintenanceWindow com uma CLI
- Usar DeregisterTaskFromMaintenanceWindow com uma CLI
- Usar DescribeActivations com uma CLI
- Usar DescribeAssociation com uma CLI
- Usar DescribeAssociationExecutionTargets com uma CLI
- Usar DescribeAssociationExecutions com uma CLI
- Usar DescribeAutomationExecutions com uma CLI
- Usar DescribeAutomationStepExecutions com uma CLI
- Usar DescribeAvailablePatches com uma CLI

- Use DescribeDocumentcom um AWS SDK ou CLI
- Usar DescribeDocumentPermission com uma CLI
- Usar DescribeEffectiveInstanceAssociations com uma CLI
- Usar DescribeEffectivePatchesForPatchBaseline com uma CLI
- Usar DescribeInstanceAssociationsStatus com uma CLI
- Usar DescribeInstanceInformation com uma CLI
- Usar DescribeInstancePatchStates com uma CLI
- Usar DescribeInstancePatchStatesForPatchGroup com uma CLI
- Usar DescribeInstancePatches com uma CLI
- Usar DescribeMaintenanceWindowExecutionTaskInvocations com uma CLI
- Usar DescribeMaintenanceWindowExecutionTasks com uma CLI
- Usar DescribeMaintenanceWindowExecutions com uma CLI
- Usar DescribeMaintenanceWindowTargets com uma CLI
- Usar DescribeMaintenanceWindowTasks com uma CLI
- Usar DescribeMaintenanceWindows com uma CLI
- Use DescribeOpsItemscom um AWS SDK ou CLI
- Use DescribeParameterscom um AWS SDK ou CLI
- Usar DescribePatchBaselines com uma CLI
- Usar DescribePatchGroupState com uma CLI
- Usar DescribePatchGroups com uma CLI
- Usar GetAutomationExecution com uma CLI
- Usar GetCommandInvocation com uma CLI
- Usar GetConnectionStatus com uma CLI
- Usar GetDefaultPatchBaseline com uma CLI
- Usar GetDeployablePatchSnapshotForInstance com uma CLI
- Usar GetDocument com uma CLI
- Usar GetInventory com uma CLI
- Usar GetInventorySchema com uma CLI
- Usar GetMaintenanceWindow com uma CLI
- Usar GetMaintenanceWindowExecution com uma CLI

- Usar GetMaintenanceWindowExecutionTask com uma CLI
- Use GetParametercom um AWS SDK ou CLI
- Usar GetParameterHistory com uma CLI
- Usar GetParameters com uma CLI
- Usar GetPatchBaseline com uma CLI
- Usar GetPatchBaselineForPatchGroup com uma CLI
- Usar ListAssociationVersions com uma CLI
- Usar ListAssociations com uma CLI
- Use ListCommandInvocationscom um AWS SDK ou CLI
- Usar ListCommands com uma CLI
- Usar ListComplianceItems com uma CLI
- Usar ListComplianceSummaries com uma CLI
- Usar ListDocumentVersions com uma CLI
- Usar ListDocuments com uma CLI
- Usar ListInventoryEntries com uma CLI
- Usar ListResourceComplianceSummaries com uma CLI
- Usar ListTagsForResource com uma CLI
- Usar ModifyDocumentPermission com uma CLI
- Usar PutComplianceItems com uma CLI
- Usar PutInventory com uma CLI
- Use PutParametercom um AWS SDK ou CLI
- Usar RegisterDefaultPatchBaseline com uma CLI
- Usar RegisterPatchBaselineForPatchGroup com uma CLI
- Usar RegisterTargetWithMaintenanceWindow com uma CLI
- Usar RegisterTaskWithMaintenanceWindow com uma CLI
- Usar RemoveTagsFromResource com uma CLI
- Use SendCommandcom um AWS SDK ou CLI
- Usar StartAutomationExecution com uma CLI
- Usar StartSession com uma CLI
- Usar StopAutomationExecution com uma CLI

- Usar UpdateAssociation com uma CLI
- Usar UpdateAssociationStatus com uma CLI
- Usar UpdateDocument com uma CLI
- Usar UpdateDocumentDefaultVersion com uma CLI
- Use UpdateMaintenanceWindowcom um AWS SDK ou CLI
- Usar UpdateManagedInstanceRole com uma CLI
- Use UpdateOpsItemcom um AWS SDK ou CLI
- Usar UpdatePatchBaseline com uma CLI

Usar **`AddTagsToResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `AddTagsToResource`.

CLI

AWS CLI

Exemplo 1: adicionar tags a uma janela de manutenção

O exemplo de `add-tags-to-resource` a seguir adiciona uma tag à janela de manutenção especificada.

```
aws ssm add-tags-to-resource \
    --resource-type "MaintenanceWindow" \
    --resource-id "mw-03eb9db428EXAMPLE" \
    --tags "Key=Stack,Value=Production"
```

Este comando não produz saída.

Exemplo 2: adicionar tags a um parâmetro

O exemplo de `add-tags-to-resource` a seguir adiciona duas tags ao parâmetro especificado.

```
aws ssm add-tags-to-resource \
    --resource-type "Parameter" \
    --resource-id "My-Parameter" \
    --tags '[{"Key":"Region","Value":"East"},{"Key":"Environment",
 "Value":"Production"}]'
```

Este comando não produz saída.

Exemplo 3: adicionar tags a um documento do SSM

O exemplo de `add-tags-to-resource` a seguir adiciona uma tag ao documento especificado.

```
aws ssm add-tags-to-resource \
    --resource-type "Document" \
    --resource-id "My-Document" \
    --tags "Key=Quarter,Value=Q322"
```

Este comando não produz saída.

Para obter mais informações, consulte Marcar recursos do Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte AddTagsToResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza uma janela de manutenção com novas tags. Não haverá saída se o comando for bem-sucedido. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada tag. Não haverá saída se o comando for bem-sucedido.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- Para obter detalhes da API, consulte AddTagsToResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza uma janela de manutenção com novas tags. Não haverá saída se o comando for bem-sucedido. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada tag. Não haverá saída se o comando for bem-sucedido.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- Para obter detalhes da API, consulte AddTagsToResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CancelCommand`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CancelCommand`.

CLI

AWS CLI

Exemplo 1: cancelar um comando para todas as instâncias

O exemplo de `cancel-command` a seguir tenta cancelar o comando especificado que já está em execução para todas as instâncias.

```
aws ssm cancel-command \
    --command-id "662add3d-5831-4a10-b64a-f2ff3EXAMPLE"
```

Este comando não produz saída.

Exemplo 2: cancelar um comando para instâncias específicas

O exemplo de `cancel-command` a seguir tenta cancelar um comando somente para a instância especificada.

```
aws ssm cancel-command \
    --command-id "662add3d-5831-4a10-b64a-f2ff3EXAMPLE"
    --instance-ids "i-02573cafcfEXAMPLE"
```

Este comando não produz saída.

Para obter mais informações, consulte Marcar parâmetros do Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte CancelCommandem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo tenta cancelar um comando. Não haverá saída se a operação for bem-sucedida.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Para obter detalhes da API, consulte CancelCommandem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo tenta cancelar um comando. Não haverá saída se a operação for bem-sucedida.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Para obter detalhes da API, consulte CancelCommandem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateActivation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateActivation`.

CLI

AWS CLI

Para criar uma ativação de instância gerenciada

O exemplo de `create-activation` a seguir cria uma ativação de instância gerenciada.

```
aws ssm create-activation \
    --default-instance-name "HybridWebServers" \
    --iam-role "HybridWebServersRole" \
    --registration-limit 5
```

Saída:

```
{
    "ActivationId": "5743558d-563b-4457-8682-d16c3EXAMPLE",
    "ActivationCode": "dRmgnYaFv567vEXAMPLE"
}
```

Para obter mais informações, consulte Etapa 4: Criar uma Managed-Instance ativação para um ambiente híbrido no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte CreateActivationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma instância gerenciada.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole
 "SSMAutomationRole" -RegistrationLimit 10
```

Saída:

```
ActivationCode       ActivationId
```

```
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Para obter detalhes da API, consulte CreateActivationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma instância gerenciada.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole
 "SSMAutomationRole" -RegistrationLimit 10
```

Saída:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Para obter detalhes da API, consulte CreateActivationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateAssociation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateAssociation`.

CLI

AWS CLI

Exemplo 1: associar um documento usando IDs de instância

Esse exemplo associa um documento de configuração a uma instância usando IDs de instância.

```
aws ssm create-association \
    --instance-id "i-0cb2b964d3e14fd9f" \
    --name "AWS-UpdateSSMAgent"
```

Saída:

```
{
```

```
    "AssociationDescription": {
        "Status": {
            "Date": 1487875500.33,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

Para obter mais informações, consulte CreateAssociationa Referência da API do AWS Systems Manager.

Exemplo 2: associar um documento usando destinos

Esse exemplo associa um documento de configuração a uma instância usando destinos.

```
aws ssm create-association \
    --name "AWS-UpdateSSMAgent" \
    --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f"
```

Saída:

```
{
    "AssociationDescription": {
        "Status": {
```

```
            "Date": 1487875500.33,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

Para obter mais informações, consulte CreateAssociationa Referência da API do AWS Systems Manager.

Exemplo 3: criar uma associação para ser executada somente uma vez

Esse exemplo cria uma nova associação que só é executada uma vez na data e na hora especificadas. As associações criadas com uma data no passado ou no presente (no momento em que são processadas, a data está no passado) são executadas imediatamente.

```
aws ssm create-association \
    --name "AWS-UpdateSSMAgent" \
    --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" \
    --schedule-expression "at(2020-05-14T15:55:00)"  \
    --apply-only-at-cron-interval
```

Saída:

```
{
```

```
    "AssociationDescription": {
        "Status": {
            "Date": 1487875500.33,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

Para obter mais informações, consulte CreateAssociationa Referência ou Referência da API do AWS Systems Manager: expressões Cron e de taxa para o Systems Manager no Guia do Usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte CreateAssociationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo associa um documento de configuração a uma instância usando IDs de instância.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Associated
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
Status.AdditionalInfo :
```

Exemplo 2: esse exemplo associa um documento de configuração a uma instância usando destinos.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Exemplo 3: esse exemplo associa um documento de configuração a uma instância usando destinos e parâmetros.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName
 "CWConfiguration" -Target $target -Parameter $params
```

Saída:

```
Name                  : Configure-CloudWatch
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Exemplo 4: esse exemplo cria uma associação com todas as instâncias na região, com **`AWS-GatherSoftwareInventory`**. Ele também fornece arquivos personalizados e locais de registro nos parâmetros a serem coletados

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Saída:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Para obter detalhes da API, consulte CreateAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa um documento de configuração a uma instância usando IDs de instância.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Associated
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
Status.AdditionalInfo :
```

Exemplo 2: esse exemplo associa um documento de configuração a uma instância usando destinos.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Exemplo 3: esse exemplo associa um documento de configuração a uma instância usando destinos e parâmetros.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
```

```
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName
 "CWConfiguration" -Target $target -Parameter $params
```

Saída:

```
Name                  : Configure-CloudWatch
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Exemplo 4: esse exemplo cria uma associação com todas as instâncias na região, com **`AWS-GatherSoftwareInventory`**. Ele também fornece arquivos personalizados e locais de registro nos parâmetros a serem coletados

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Saída:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Para obter detalhes da API, consulte CreateAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateAssociationBatch`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateAssociationBatch`.

CLI

AWS CLI

Para criar várias associações

Este exemplo associa um documento de configuração a várias instâncias. A saída retorna uma lista de operações bem e malsucedidas, se aplicável.

Comando:

```
aws ssm create-association-batch --entries "Name=AWS-UpdateSSMAgent,InstanceId=i-1234567890abcdef0" "Name=AWS-UpdateSSMAgent,InstanceId=i-9876543210abcdef0"
```

Saída:

```
{
  "Successful": [
      {
          "Name": "AWS-UpdateSSMAgent",
          "InstanceId": "i-1234567890abcdef0",
          "AssociationVersion": "1",
          "Date": 1550504725.007,
          "LastUpdateAssociationDate": 1550504725.007,
          "Status": {
              "Date": 1550504725.007,
              "Name": "Associated",
              "Message": "Associated with AWS-UpdateSSMAgent"
          },
          "Overview": {
              "Status": "Pending",
              "DetailedStatus": "Creating"
          },
          "DocumentVersion": "$DEFAULT",
          "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
          "Targets": [
              {
```

```
                    "Key": "InstanceIds",
                    "Values": [
                        "i-1234567890abcdef0"
                    ]
                }
            ]
        },
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-9876543210abcdef0",
            "AssociationVersion": "1",
            "Date": 1550504725.057,
            "LastUpdateAssociationDate": 1550504725.057,
            "Status": {
                "Date": 1550504725.057,
                "Name": "Associated",
                "Message": "Associated with AWS-UpdateSSMAgent"
            },
            "Overview": {
                "Status": "Pending",
                "DetailedStatus": "Creating"
            },
            "DocumentVersion": "$DEFAULT",
            "AssociationId": "9c9f7f20-5154-4fed-a83e-0123456789ab",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-9876543210abcdef0"
                    ]
                }
            ]
        }
    ],
    "Failed": []
}
```

- Para obter detalhes da API, consulte CreateAssociationBatchem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo associa um documento de configuração a várias instâncias. A saída retorna uma lista de operações bem e malsucedidas, se aplicável.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Saída:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Exemplo 2: esse exemplo mostrará os detalhes completos de uma operação bem-sucedida.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Para obter detalhes da API, consulte CreateAssociationBatchem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa um documento de configuração a várias instâncias. A saída retorna uma lista de operações bem e malsucedidas, se aplicável.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Saída:

```
Failed  Successful
------  ----------
```

```
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Exemplo 2: esse exemplo mostrará os detalhes completos de uma operação bem-sucedida.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Para obter detalhes da API, consulte CreateAssociationBatchem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`CreateDocument`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateDocument`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para criar um documento

O exemplo de `create-document` a seguir cria um documento do Systems Manager.

```
aws ssm create-document \
    --content file://exampleDocument.yml \
    --name "Example" \
    --document-type "Automation" \
    --document-format YAML
```

Saída:

```
{
    "DocumentDescription": {
```

```
        "Hash":
 "fc2410281f40779e694a8b95975d0f9f316da8a153daa94e3d9921102EXAMPLE",
        "HashType": "Sha256",
        "Name": "Example",
        "Owner": "29884EXAMPLE",
        "CreatedDate": 1583256349.452,
        "Status": "Creating",
        "DocumentVersion": "1",
        "Description": "Document Example",
        "Parameters": [
            {
                "Name": "AutomationAssumeRole",
                "Type": "String",
                "Description": "(Required) The ARN of the role that allows
 Automation to perform the actions on your behalf. If no role is specified,
 Systems Manager Automation uses your IAM permissions to execute this document.",
                "DefaultValue": ""
            },
            {
                "Name": "InstanceId",
                "Type": "String",
                "Description": "(Required) The ID of the Amazon EC2 instance.",
                "DefaultValue": ""
            }
        ],
        "PlatformTypes": [
            "Windows",
            "Linux"
        ],
        "DocumentType": "Automation",
        "SchemaVersion": "0.3",
        "LatestVersion": "1",
        "DefaultVersion": "1",
        "DocumentFormat": "YAML",
        "Tags": []
    }
}
```

Para obter mais informações, consulte Criar documentos do Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte CreateDocumentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates an AWS SSM document asynchronously.
 *
 * @param docName The name of the document to create.
 * <p>
 * This method initiates an asynchronous request to create an SSM document.
 * If the request is successful, it prints the document status.
 * If an exception occurs, it handles the error appropriately.
 */
public void createSSMDoc(String docName) throws SsmException {
    String jsonData = """
    {
    "schemaVersion": "2.2",
    "description": "Run a simple shell command",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runEchoCommand",
            "inputs": {
              "runCommand": [
                "echo 'Hello, world!'"
              ]
            }
          }
        ]
    }
    """;

    CreateDocumentRequest request = CreateDocumentRequest.builder()
            .content(jsonData)
            .name(docName)
            .documentType(DocumentType.COMMAND)
```

```
                .build();

        CompletableFuture<CreateDocumentResponse> future =
 getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
 response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

- Para obter detalhes da API, consulte CreateDocumenta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
```

```
export const main = async ({ content, name, documentType }) => {
  const client = new SSMClient({});
  try {
    const { documentDescription } = await client.send(
      new CreateDocumentCommand({
        Content: content, // The content for the new SSM document. The content
 must not exceed 64KB.
        Name: name,
        DocumentType: documentType, // Document format type can be JSON, YAML, or
 TEXT. The default format is JSON.
      }),
    );
    console.log("Document created successfully.");
    return { DocumentDescription: documentDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DocumentAlreadyExists") {
      console.warn(`${caught.message}. Did you provide a new document name?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateDocumenta Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um documento na sua conta. O documento deve estar em formato JSON. Para obter mais informações sobre como escrever um documento de configuração, consulte Documento de configuração na Referência da API do SSM.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Saída:

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
```

```
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Creating
```

- Para obter detalhes da API, consulte CreateDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um documento na sua conta. O documento deve estar em formato JSON. Para obter mais informações sobre como escrever um documento de configuração, consulte Documento de configuração na Referência da API do SSM.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Saída:

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
```

```
Sha1            :
Status          : Creating
```

- Para obter detalhes da API, consulte CreateDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
```

```
                Name=name, Content=content, DocumentType="Command"
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateDocumentReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ssm->createdocument(
            iv_name = iv_name
            iv_content = iv_content
            iv_documenttype = 'Command' ).
        MESSAGE 'Document created.' TYPE 'I'.
      CATCH /aws1/cx_ssmdocalreadyexists.
        MESSAGE 'Document already exists.' TYPE 'I'.
      CATCH /aws1/cx_ssminvaliddoccontent.
        MESSAGE 'Invalid document content.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDocumentreferência da API AWS SDK for SAP ABAP.

Use **`CreateMaintenanceWindow`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateMaintenanceWindow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Exemplo 1: criar uma janela de manutenção

O exemplo de `create-maintenance-window` a seguir cria uma nova janela de manutenção que, a cada cinco minutos, e por até duas horas (conforme necessário), impede que novas tarefas sejam iniciadas dentro de uma hora do final da execução da janela de manutenção, permite destinos não associados (instâncias que você não registrou na janela de manutenção) e indica, por meio do uso de tags personalizadas, que seu criador pretende usá-la em um tutorial.

```
aws ssm create-maintenance-window \
    --name "My-Tutorial-Maintenance-Window" \
    --schedule "rate(5 minutes)" \
    --duration 2 --cutoff 1 \
    --allow-unassociated-targets \
    --tags "Key=Purpose,Value=Tutorial"
```

Saída:

```
{
    "WindowId": "mw-0c50858d01EXAMPLE"
}
```

Exemplo 2: criar uma janela de manutenção que é executada somente uma vez

O exemplo de `create-maintenance-window` a seguir cria uma nova janela de manutenção que só é executada uma vez na data e na hora especificadas.

```
aws ssm create-maintenance-window \
    --name My-One-Time-Maintenance-Window \
    --schedule "at(2020-05-14T15:55:00)" \
    --duration 5 \
    --cutoff 2 \
    --allow-unassociated-targets \
    --tags "Key=Environment,Value=Production"
```

Saída:

```
{
    "WindowId": "mw-01234567890abcdef"
}
```

Para obter mais informações, consulte Janelas de manutenção no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte CreateMaintenanceWindowem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
```

```
    * This method initiates an asynchronous request to create an SSM maintenance
window.
    * If the request is successful, it prints the maintenance window ID.
    * If an exception occurs, it handles the error appropriately.
    */
   public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
       CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
               .name(winName)
               .description("This is my maintenance window")
               .allowUnassociatedTargets(true)
               .duration(2)
               .cutoff(1)
               .schedule("cron(0 10 ? * MON-FRI *)")
               .build();

       CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
       final String[] windowId = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String maintenanceWindowId = response.windowId();
               System.out.println("The maintenance window id is " +
maintenanceWindowId);
               windowId[0] = maintenanceWindowId;
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof DocumentAlreadyExistsException) {
                   throw new CompletionException(cause);
               } else if (cause instanceof SsmException) {
                   throw new CompletionException(cause);
               } else {
                   throw new RuntimeException(cause);
               }
           }
       }).join();

       if (windowId[0] == null) {
           MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                   .key("name")
                   .values(winName)
                   .build();
```

```
            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture
= getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
```

- Para obter detalhes da API, consulte CreateMaintenanceWindowa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM maintenance window.
 * @param {{ name: string, allowUnassociatedTargets: boolean, duration: number,
 cutoff: number, schedule: string, description?: string }}
 */
export const main = async ({
  name,
  allowUnassociatedTargets, // Allow the maintenance window to run on managed
 nodes, even if you haven't registered those nodes as targets.
  duration, // The duration of the maintenance window in hours.
  cutoff, // The number of hours before the end of the maintenance window that
 Amazon Web Services Systems Manager stops scheduling new tasks for execution.
  schedule, // The schedule of the maintenance window in the form of a cron or
 rate expression.
  description = undefined,
}) => {
  const client = new SSMClient({});

  try {
    const { windowId } = await client.send(
      new CreateMaintenanceWindowCommand({
        Name: name,
        Description: description,
        AllowUnassociatedTargets: allowUnassociatedTargets, // Allow the
 maintenance window to run on managed nodes, even if you haven't registered those
 nodes as targets.
        Duration: duration, // The duration of the maintenance window in hours.
```

```
        Cutoff: cutoff, // The number of hours before the end of the maintenance
 window that Amazon Web Services Systems Manager stops scheduling new tasks for
 execution.
        Schedule: schedule, // The schedule of the maintenance window in the form
 of a cron or rate expression.
      }),
    );
    console.log(`Maintenance window created with Id: ${windowId}`);
    return { WindowId: windowId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateMaintenanceWindowa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma nova janela de manutenção com o nome especificado que é executada às 16h toda terça-feiras por 4 horas, com um limite de 1 hora, e que permite destinos não associados.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Saída:

```
mw-03eb53e1ea7383998
```

- Para obter detalhes da API, consulte CreateMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma nova janela de manutenção com o nome especificado que é executada às 16h toda terça-feiras por 4 horas, com um limite de 1 hora, e que permite destinos não associados.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Saída:

```
mw-03eb53e1ea7383998
```

- Para obter detalhes da API, consulte CreateMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
```

```
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff,
allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateMaintenanceWindowReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ssm->createmaintenancewindow(
        iv_name = iv_name
        iv_schedule = iv_schedule
        iv_duration = iv_duration
        iv_cutoff = iv_cutoff
        iv_allowunassociatedtargets = iv_allow_unassociated_targets ).
    MESSAGE 'Maintenance window created.' TYPE 'I'.
  CATCH /aws1/cx_ssmresrclimitexcdex.
    MESSAGE 'Resource limit exceeded.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateMaintenanceWindowreferência da API AWS SDK for SAP ABAP.

Use **`CreateOpsItem`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateOpsItem`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para criar um OpsItems

O `create-ops-item` exemplo a seguir usa a aws/resources chave/ OperationalData para criar um OpsItem com um recurso relacionado ao Amazon DynamoDB.

```
aws ssm create-ops-item \
    --title "EC2 instance disk full" \
    --description "Log clean up may have failed which caused the disk to be full"
 \
    --priority 2 \
    --source ec2 \
    --operational-data '{"/aws/resources":{"Value":"[{\"arn
\": \"arn:aws:dynamodb:us-west-2:12345678:table/OpsItems
\"}]","Type":"SearchableString"}}' \
    --notifications Arn="arn:aws:sns:us-west-2:12345678:TestUser"
```

Saída:

```
{
    "OpsItemId": "oi-1a2b3c4d5e6f"
}
```

Para obter mais informações, consulte Creating OpsItems in the AWS Systems Manager User Guide.

- Para obter detalhes da API, consulte CreateOpsItemem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
 String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
 getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateOpsItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM OpsItem.
 * @param {{ title: string, source: string, category?: string, severity?:
 string }}
 */
export const main = async ({
  title,
  source,
  category = undefined,
  severity = undefined,
}) => {
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new CreateOpsItemCommand({
        Title: title,
        Source: source, // The origin of the OpsItem, such as Amazon EC2 or
 Systems Manager.
        Category: category,
        Severity: severity,
      }),
    );
    console.log(`Ops item created with id: ${opsItemId}`);
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
```

```
    }
  }
};
```

- Para obter detalhes da API, consulte CreateOpsItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem
```

```
        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
 OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateOpsItemReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ssm->createopsitem(
        iv_title = iv_title
        iv_source = iv_source
        iv_category = iv_category
        iv_severity = iv_severity
        iv_description = iv_description ).
    MESSAGE 'OpsItem created.' TYPE 'I'.
  CATCH /aws1/cx_ssmopsitemlimitexcdex.
    MESSAGE 'You have exceeded your open OpsItem limit.' TYPE 'I'.
  CATCH /aws1/cx_ssmopsitemalrdyexex.
    MESSAGE 'OpsItem already exists.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateOpsItemreferência da API AWS SDK for SAP ABAP.

Usar **`CreatePatchBaseline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreatePatchBaseline`.

CLI

AWS CLI

Exemplo 1: criar uma lista de referência de patches com aprovação automática

O exemplo de `create-patch-baseline` a seguir cria uma lista de referência de patches para o Windows Server que aprova patches para instâncias de produção sete dias após serem lançados pela Microsoft.

```
aws ssm create-patch-baseline \
    --name "Windows-Production-Baseline-AutoApproval" \
    --operating-system "WINDOWS" \
    --approval-
rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,1
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},App
 \
    --description "Baseline containing all updates approved for Windows Server
 production systems"
```

Saída:

```
{
    "BaselineId": "pb-045f10b4f3EXAMPLE"
}
```

Exemplo 2: criar uma lista de referência de patches com uma data limite para aprovação

O exemplo de `create-patch-baseline` a seguir cria uma lista de referência de patches para o Windows Server que aprova todos os patches para um ambiente de produção lançados até 7 de julho de 2020.

```
aws ssm create-patch-baseline \
    --name "Windows-Production-Baseline-AutoApproval" \
    --operating-system "WINDOWS" \
    --approval-
rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,1
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},App
 \
    --description "Baseline containing all updates approved for Windows Server
 production systems"
```

Saída:

```
{
    "BaselineId": "pb-045f10b4f3EXAMPLE"
}
```

Exemplo 3: criar uma lista de referência de patches com regras de aprovação armazenadas em um arquivo JSON

O exemplo de `create-patch-baseline` a seguir cria uma lista de referência de patches para o Amazon Linux 2017.09 que aprova patches para um ambiente de produção sete dias após seu lançamento, especifica regras de aprovação para a lista de referência de patches e especifica um repositório personalizado para patches.

```
aws ssm create-patch-baseline \
    --cli-input-json file://my-amazon-linux-approval-rules-and-repo.json
```

Conteúdo de `my-amazon-linux-approval-rules-and-repo.json`:

```
{
    "Name": "Amazon-Linux-2017.09-Production-Baseline",
    "Description": "My approval rules patch baseline for Amazon Linux 2017.09
 instances",
    "OperatingSystem": "AMAZON_LINUX",
    "Tags": [
        {
            "Key": "Environment",
            "Value": "Production"
        }
    ],
    "ApprovalRules": {
        "PatchRules": [
            {
                "ApproveAfterDays": 7,
                "EnableNonSecurity": true,
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "SEVERITY",
                            "Values": [
                                "Important",
                                "Critical"
                            ]
                        },
                        {
                            "Key": "CLASSIFICATION",
                            "Values": [
                                "Security",
                                "Bugfix"
                            ]
                        },
```

```
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "AmazonLinux2017.09"
                            ]
                        }
                    ]
                }
            }
        ]
    },
    "Sources": [
        {
            "Name": "My-AL2017.09",
            "Products": [
                "AmazonLinux2017.09"
            ],
            "Configuration": "[amzn-main] \nname=amzn-main-Base
\nmirrorlist=http://repo./$awsregion./$awsdomain//$releasever/main/
mirror.list //nmirrorlist_expire=300//nmetadata_expire=300 \npriority=10
 \nfailovermethod=priority \nfastestmirror_enabled=0 \ngpgcheck=1
 \ngpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-ga \nenabled=1 \nretries=3
 \ntimeout=5\nreport_instanceid=yes"
        }
    ]
}
```

Exemplo 4: criar uma lista de referência de patches que especifica patches aprovados e rejeitados

O exemplo de `create-patch-baseline` a seguir especifica explicitamente os patches a serem aprovados e rejeitados como exceção às regras de aprovação padrão.

```
aws ssm create-patch-baseline \
    --name "Amazon-Linux-2017.09-Alpha-Baseline" \
    --description "My custom approve/reject patch baseline for Amazon Linux
 2017.09 instances" \
    --operating-system "AMAZON_LINUX" \
    --approved-patches "CVE-2018-1234567,example-pkg-EE-2018*.amzn1.noarch" \
    --approved-patches-compliance-level "HIGH" \
    --approved-patches-enable-non-security \
    --tags "Key=Environment,Value=Alpha"
```

Para obter mais informações, consulte Criar uma lista de referência de patches personalizada no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte CreatePatchBaselineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria uma lista de referência de patches que aprova patches, sete dias após serem lançados pela Microsoft, para instâncias gerenciadas que executam o Windows Server 2019 em um ambiente de produção.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters
```

```
New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description
 "Baseline containing all updates approved for production systems" -
ApprovalRules_PatchRule $rule
```

Saída:

```
pb-0z4z6221c4296b23z
```

- Para obter detalhes da API, consulte CreatePatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma lista de referência de patches que aprova patches, sete dias após serem lançados pela Microsoft, para instâncias gerenciadas que executam o Windows Server 2019 em um ambiente de produção.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
```

```
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description
 "Baseline containing all updates approved for production systems" -
ApprovalRules_PatchRule $rule
```

Saída:

```
pb-0z4z6221c4296b23z
```

- Para obter detalhes da API, consulte CreatePatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteActivation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteActivation`.

CLI

AWS CLI

Para excluir uma ativação de instância gerenciada

O exemplo de `delete-activation` a seguir exclui uma ativação de instância gerenciada.

```
aws ssm delete-activation \
    --activation-id "aa673477-d926-42c1-8757-1358cEXAMPLE"
```

Este comando não produz saída.

Para obter mais informações, consulte Configurando o AWS Systems Manager para ambientes híbridos no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeleteActivationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui uma ativação. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Para obter detalhes da API, consulte DeleteActivationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui uma ativação. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Para obter detalhes da API, consulte DeleteActivationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteAssociation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteAssociation`.

CLI

AWS CLI

Exemplo 1: excluir uma associação usando o ID da associação

O exemplo de `delete-association` a seguir exclui a associação para o ID de associação especificado. Não haverá saída se o comando for bem-sucedido.

```
aws ssm delete-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Este comando não produz saída.

Para obter mais informações, consulte Editar e criar uma nova versão de uma associação no Guia do usuário do AWS Systems Manager.

Exemplo 2: excluir uma associação

O exemplo de `delete-association` a seguir exclui a associação entre uma instância e um documento. Não haverá saída se o comando for bem-sucedido.

```
aws ssm delete-association \
    --instance-id "i-1234567890abcdef0" \
    --name "AWS-UpdateSSMAgent"
```

Este comando não produz saída.

Para obter mais informações, consulte Trabalhar com associações no Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeleteAssociationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui a associação entre uma instância e um documento. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-
UpdateSSMAgent"
```

- Para obter detalhes da API, consulte DeleteAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a associação entre uma instância e um documento. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-
UpdateSSMAgent"
```

- Para obter detalhes da API, consulte DeleteAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DeleteDocument`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteDocument`.

CLI

AWS CLI

Para excluir um documento

O exemplo de `delete-document` a seguir exclui um documento do Systems Manager.

```
aws ssm delete-document \
    --name "Example"
```

Este comando não produz saída.

Para obter mais informações, consulte Criar documentos do Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeleteDocumentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();
```

```
        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Para obter detalhes da API, consulte DeleteDocumenta Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM document.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(new DeleteDocumentCommand({ Name: documentName }));
    console.log(`Document '${documentName}' deleted.`);
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteDocumenta Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui um documento. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Para obter detalhes da API, consulte DeleteDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui um documento. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Para obter detalhes da API, consulte DeleteDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
```

```
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteDocumentReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ssm->deletedocument( iv_name = iv_name ).
    MESSAGE 'Document deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssminvaliddocument.
    MESSAGE 'Invalid document.' TYPE 'I'.
  CATCH /aws1/cx_ssmassocdinstances.
    MESSAGE 'Document has associated instances.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDocumentreferência da API AWS SDK for SAP ABAP.

Use **`DeleteMaintenanceWindow`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteMaintenanceWindow`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para excluir uma janela de manutenção

Este exemplo de `delete-maintenance-window` remove a janela de manutenção especificada.

```
aws ssm delete-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9"
```

Saída:

```
{
    "WindowId":"mw-1a2b3c4d5e6f7g8h9"
}
```

Para obter mais informações, consulte Excluir uma janela de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeleteMaintenanceWindowem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes an AWS SSM Maintenance Window asynchronously.
    *
    * @param winId The ID of the Maintenance Window to delete.
    * <p>
    * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteMaintenanceWindow(String winId) {
       DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
               .windowId(winId)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteMaintenanceWindow(windowRequest)
                   .thenAccept(response -> {
                       System.out.println("The maintenance window was
successfully deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
```

```
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }
```

- Para obter detalhes da API, consulte DeleteMaintenanceWindowa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM maintenance window.
 * @param {{ windowId: string }}
 */
export const main = async ({ windowId }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new DeleteMaintenanceWindowCommand({ WindowId: windowId }),
    );
    console.log(`Maintenance window '${windowId}' deleted.`);
```

```
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteMaintenanceWindowa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove uma janela de manutenção.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Saída:

```
mw-06d59c1a07c022145
```

- Para obter detalhes da API, consulte DeleteMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove uma janela de manutenção.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Saída:

```
mw-06d59c1a07c022145
```

- Para obter detalhes da API, consulte DeleteMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
```

```
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteMaintenanceWindowReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ssm->deletemaintenancewindow( iv_windowid = iv_window_id ).
    MESSAGE 'Maintenance window deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssminternalservererr.
    MESSAGE 'Internal server error occurred.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteMaintenanceWindowreferência da API AWS SDK for SAP ABAP.

Use **`DeleteOpsItem`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DeleteOpsItem`.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
```

```
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteOpsItemReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ssm->deleteopsitem( iv_opsitemid = iv_ops_item_id ).
    MESSAGE 'OpsItem deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssmopsiteminvparamex.
    MESSAGE 'Invalid OpsItem parameter.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteOpsItemreferência da API AWS SDK for SAP ABAP.

Usar **`DeleteParameter`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteParameter`.

CLI

AWS CLI

Para excluir um parâmetro

O exemplo de `delete-parameter` a seguir exclui o parâmetro único especificado.

```
aws ssm delete-parameter \
    --name "MyParameter"
```

Este comando não produz saída.

Para obter mais informações, consulte Trabalhar com o Parameter Store no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeleteParameterem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMParameter -Name "helloWorld"
```

- Para obter detalhes da API, consulte DeleteParameterem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMParameter -Name "helloWorld"
```

- Para obter detalhes da API, consulte DeleteParameterem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeletePatchBaseline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeletePatchBaseline`.

CLI

AWS CLI

Para excluir uma lista de referência de patches

O exemplo de `delete-patch-baseline` a seguir exclui a lista de referência de patches especificada.

```
aws ssm delete-patch-baseline \
    --baseline-id "pb-045f10b4f382baeda"
```

Saída:

```
{
    "BaselineId": "pb-045f10b4f382baeda"
}
```

Para obter mais informações, consulte Atualizar ou excluir uma lista de referência de patches (Console) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeletePatchBaselineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exclui uma lista de referência de patches.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Saída:

```
pb-045f10b4f382baeda
```

- Para obter detalhes da API, consulte DeletePatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui uma lista de referência de patches.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Saída:

```
pb-045f10b4f382baeda
```

- Para obter detalhes da API, consulte DeletePatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeregisterManagedInstance`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterManagedInstance`.

CLI

AWS CLI

Para cancelar o registro de uma instância gerenciada

O exemplo de `deregister-managed-instance` a seguir cancela o registro da instância gerenciada especificada.

```
aws ssm deregister-managed-instance \
    --instance-id 'mi-08ab247cdfEXAMPLE'
```

Este comando não produz saída.

Para obter mais informações, consulte Cancelar o registro de nós gerenciados em ambientes híbridos e multinuvem no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeregisterManagedInstanceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela o registro de uma instância gerenciada. Não haverá saída se o comando for bem-sucedido.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Para obter detalhes da API, consulte DeregisterManagedInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro de uma instância gerenciada. Não haverá saída se o comando for bem-sucedido.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Para obter detalhes da API, consulte DeregisterManagedInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeregisterPatchBaselineForPatchGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterPatchBaselineForPatchGroup`.

CLI

AWS CLI

Para cancelar o registro de um grupo de patches de uma lista de referência de patches

O exemplo de `deregister-patch-baseline-for-patch-group` a seguir cancela o registro do grupo de patches especificado da lista de referência de patches especificada.

```
aws ssm deregister-patch-baseline-for-patch-group \
    --patch-group "Production" \
    --baseline-id "pb-0ca44a362fEXAMPLE"
```

Saída:

```
{
  "PatchGroup":"Production",
  "BaselineId":"pb-0ca44a362fEXAMPLE"
}
```

Para obter mais informações, consulte Adicionar um grupo de patches a uma lista de referência de patches no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeregisterPatchBaselineForPatchGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cancela o registro de um grupo de patches de uma lista de referência de patches.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Para obter detalhes da API, consulte DeregisterPatchBaselineForPatchGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro de um grupo de patches de uma lista de referência de patches.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Para obter detalhes da API, consulte DeregisterPatchBaselineForPatchGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeregisterTargetFromMaintenanceWindow`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterTargetFromMaintenanceWindow`.

CLI

AWS CLI

Para remover um destino de uma janela de manutenção

O exemplo de `deregister-target-from-maintenance-window` a seguir remove o destino especificado da janela de manutenção especificada.

```
aws ssm deregister-target-from-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --window-target-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

Saída:

```
{
    "WindowId":"mw-ab12cd34ef56gh78",
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Para obter mais informações, consulte Update a Maintenance Window (AWS CLI) no AWS Systems Manager User Guide.

- Para obter detalhes da API, consulte DeregisterTargetFromMaintenanceWindowem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove um destino de uma janela de manutenção.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId
 "6ab5c208-9fc4-4697-84b7-b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Para obter detalhes da API, consulte DeregisterTargetFromMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove um destino de uma janela de manutenção.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId
 "6ab5c208-9fc4-4697-84b7-b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Para obter detalhes da API, consulte DeregisterTargetFromMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeregisterTaskFromMaintenanceWindow`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeregisterTaskFromMaintenanceWindow`.

CLI

AWS CLI

Para remover uma tarefa de uma janela de manutenção

O exemplo de `deregister-task-from-maintenance-window` a seguir remove a tarefa especificada da janela de manutenção especificada.

```
aws ssm deregister-task-from-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --window-task-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c"
```

Saída:

```
{
    "WindowTaskId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c",
    "WindowId":"mw-ab12cd34ef56gh78"
}
```

Para obter mais informações, consulte Systems Manager Maintenance Windows Tutorials (AWS CLI) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DeregisterTaskFromMaintenanceWindowem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove uma tarefa de uma janela de manutenção.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Saída:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Para obter detalhes da API, consulte DeregisterTaskFromMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove uma tarefa de uma janela de manutenção.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Saída:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Para obter detalhes da API, consulte DeregisterTaskFromMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeActivations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeActivations`.

CLI

AWS CLI

Para descrever as ativações

O `describe-activations` exemplo a seguir lista detalhes sobre as ativações em sua AWS conta.

```
aws ssm describe-activations
```

Saída:

```
{
    "ActivationList": [
        {
            "ActivationId": "5743558d-563b-4457-8682-d16c3EXAMPLE",
            "Description": "Example1",
            "IamRole": "HybridWebServersRole,
```

```
            "RegistrationLimit": 5,
            "RegistrationsCount": 5,
            "ExpirationDate": 1584316800.0,
            "Expired": false,
            "CreatedDate": 1581954699.792
        },
        {
            "ActivationId": "3ee0322b-f62d-40eb-b672-13ebfEXAMPLE",
            "Description": "Example2",
            "IamRole": "HybridDatabaseServersRole",
            "RegistrationLimit": 5,
            "RegistrationsCount": 5,
            "ExpirationDate": 1580515200.0,
            "Expired": true,
            "CreatedDate": 1578064132.002
        },
    ]
}
```

Para obter mais informações, consulte Etapa 4: Criar uma Managed-Instance ativação para um ambiente híbrido no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeActivationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo fornece detalhes sobre as ativações em sua conta.

```
Get-SSMActivation
```

Saída:

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
```

```
RegistrationLimit   : 10
RegistrationsCount  : 0
```

- Para obter detalhes da API, consulte DescribeActivationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo fornece detalhes sobre as ativações em sua conta.

```
Get-SSMActivation
```

Saída:

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
RegistrationLimit   : 10
RegistrationsCount  : 0
```

- Para obter detalhes da API, consulte DescribeActivationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAssociation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAssociation`.

CLI

AWS CLI

Exemplo 1: obter detalhes de uma associação

O exemplo de `describe-association` a seguir descreve a associação para o ID de associação especificado.

```
aws ssm describe-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Saída:

```
{
    "AssociationDescription": {
        "Name": "AWS-GatherSoftwareInventory",
        "AssociationVersion": "1",
        "Date": 1534864780.995,
        "LastUpdateAssociationDate": 1543235759.81,
        "Overview": {
            "Status": "Success",
            "AssociationStatusAggregatedCount": {
                "Success": 2
            }
        },
        "DocumentVersion": "$DEFAULT",
        "Parameters": {
            "applications": [
                "Enabled"
            ],
            "awsComponents": [
                "Enabled"
            ],
            "customInventory": [
                "Enabled"
            ],
            "files": [
                ""
            ],
            "instanceDetailedInformation": [
                "Enabled"
            ],
            "networkConfig": [
                "Enabled"
            ],
            "services": [
                "Enabled"
            ],
            "windowsRegistry": [
                ""
            ],
            "windowsRoles": [
                "Enabled"
            ],
            "windowsUpdates": [
```

```
                "Enabled"
            ]
        },
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "InstanceIds",
                "Values": [
                    "*"
                ]
            }
        ],
        "ScheduleExpression": "rate(24 hours)",
        "LastExecutionDate": 1550501886.0,
        "LastSuccessfulExecutionDate": 1550501886.0,
        "AssociationName": "Inventory-Association"
    }
}
```

Para obter mais informações, consulte Editar e criar uma nova versão de uma associação no Guia do usuário do AWS Systems Manager.

Exemplo 2: obter detalhes de uma associação para uma instância e um documento específicos

O exemplo de `describe-association` a seguir descreve a associação entre uma instância e um documento.

```
aws ssm describe-association \
    --instance-id "i-1234567890abcdef0" \
    --name "AWS-UpdateSSMAgent"
```

Saída:

```
{
    "AssociationDescription": {
        "Status": {
            "Date": 1487876122.564,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
```

```
        "InstanceId": "i-1234567890abcdef0",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Associated",
            "AssociationStatusAggregatedCount": {
                "Pending": 1
            }
        },
        "AssociationId": "d8617c07-2079-4c18-9847-1234567890ab",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487876122.564,
        "Date": 1487876122.564,
        "Targets": [
            {
                "Values": [
                    "i-1234567890abcdef0"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

Para obter mais informações, consulte Editar e criar uma nova versão de uma associação no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeAssociationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve a associação entre uma instância e um documento.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
```

```
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Para obter detalhes da API, consulte DescribeAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a associação entre uma instância e um documento.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Para obter detalhes da API, consulte DescribeAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAssociationExecutionTargets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAssociationExecutionTargets`.

CLI

AWS CLI

Para obter detalhes da execução de uma associação

O exemplo de `describe-association-execution-targets` a seguir descreve a execução da associação especificada.

```
aws ssm describe-association-execution-targets \
```

```
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --execution-id "7abb6378-a4a5-4f10-8312-0123456789ab"
```

Saída:

```
{
    "AssociationExecutionTargets": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "ResourceId": "i-1234567890abcdef0",
            "ResourceType": "ManagedInstance",
            "Status": "Success",
            "DetailedStatus": "Success",
            "LastExecutionDate": 1550505538.497,
            "OutputSource": {
                "OutputSourceId": "97fff367-fc5a-4299-aed8-0123456789ab",
                "OutputSourceType": "RunCommand"
            }
        }
    ]
}
```

Para obter mais informações, consulte Visualizar históricos de associações no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeAssociationExecutionTargetsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe o ID do recurso e seu status de execução que fazem parte dos destinos de execução da associação

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Saída:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Exemplo 2: esse comando verifica a execução específica de uma automação específica desde ontem, onde documento de comandos está associado. Além disso, ele verifica se a execução da associação falhou e, em caso afirmativo, exibie os detalhes da invocação do comando para a execução junto com o ID da instância

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId
 $execution.OutputSource.OutputSourceId -Detail:$true | Select-Object -
ExpandProperty CommandPlugins
    }
}
```

Saída:

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0
 on i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
```

```
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
StatusDetails          : Failed
```

- Para obter detalhes da API, consulte DescribeAssociationExecutionTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe o ID do recurso e seu status de execução que fazem parte dos destinos de execução da associação

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Saída:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Exemplo 2: esse comando verifica a execução específica de uma automação específica desde ontem, onde documento de comandos está associado. Além disso, ele verifica se a execução da associação falhou e, em caso afirmativo, exibie os detalhes da invocação do comando para a execução junto com o ID da instância

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
```

```
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId
 $execution.OutputSource.OutputSourceId -Detail:$true | Select-Object -
ExpandProperty CommandPlugins
    }
}
```

Saída:

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0
 on i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
StatusDetails          : Failed
```

- Para obter detalhes da API, consulte DescribeAssociationExecutionTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAssociationExecutions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAssociationExecutions`.

CLI

AWS CLI

Exemplo 1: obter detalhes de todas as execuções de uma associação

O exemplo de `describe-association-executions` a seguir descreve todas as execuções da associação especificada.

```
aws ssm describe-association-executions \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Saída:

```
{
    "AssociationExecutions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "474925ef-1249-45a2-b93d-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505827.119,
            "ResourceCountByStatus": "{Success=1}"
        },
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505536.843,
            "ResourceCountByStatus": "{Success=1}"
        },
        ...
    ]
}
```

Para obter mais informações, consulte Visualizar históricos de associações no Guia do usuário do AWS Systems Manager.

Exemplo 2: obter detalhes de todas as execuções de uma associação após uma data e uma hora específicas

O exemplo de `describe-association-executions` a seguir descreve todas as execuções de uma associação após a data e a hora especificadas.

```
aws ssm describe-association-executions \
```

```
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --filters "Key=CreatedTime,Value=2019-02-18T16:00:00Z,Type=GREATER_THAN"
```

Saída:

```
{
    "AssociationExecutions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "474925ef-1249-45a2-b93d-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505827.119,
            "ResourceCountByStatus": "{Success=1}"
        },
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505536.843,
            "ResourceCountByStatus": "{Success=1}"
        },
        ...
    ]
}
```

Para obter mais informações, consulte Visualizar históricos de associações no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeAssociationExecutionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo retorna as execuções para o ID de associação fornecido

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Saída:

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- Para obter detalhes da API, consulte DescribeAssociationExecutionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna as execuções para o ID de associação fornecido

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Saída:

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- Para obter detalhes da API, consulte DescribeAssociationExecutionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAutomationExecutions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAutomationExecutions`.

CLI

AWS CLI

Para descrever uma execução do Automation

O exemplo de `describe-automation-executions` a seguir exibe detalhes sobre uma execução do Automation.

```
aws ssm describe-automation-executions \
    --filters Key=ExecutionId,Values=73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Saída:

```
{
    "AutomationExecutionMetadataList": [
        {
            "AutomationExecutionId": "73c8eef8-f4ee-4a05-820c-e354fEXAMPLE",
            "DocumentName": "AWS-StartEC2Instance",
            "DocumentVersion": "1",
            "AutomationExecutionStatus": "Success",
            "ExecutionStartTime": 1583737233.748,
            "ExecutionEndTime": 1583737234.719,
            "ExecutedBy": "arn:aws:sts::29884EXAMPLE:assumed-role/
mw_service_role/OrchestrationService",
            "LogFile": "",
            "Outputs": {},
            "Mode": "Auto",
            "Targets": [],
            "ResolvedTargets": {
                "ParameterValues": [],
                "Truncated": false
            },
            "AutomationType": "Local"
        }
    ]
}
```

Para obter mais informações, consulte Executar um fluxo de trabalho simples do Automation no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeAutomationExecutionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve todas as execuções do Automation ativas e encerradas associadas à sua conta.

```
Get-SSMAutomationExecutionList
```

Saída:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Exemplo 2: Este exemplo exibe ExecutionID, documento e start/end data e hora de execução para execuções que não sejam 'Success' AutomationExecutionStatus

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Saída:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
---------------------                ------------
 ------------------------- ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
```

```
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi
 Failed                    4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- Para obter detalhes da API, consulte DescribeAutomationExecutionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve todas as execuções do Automation ativas e encerradas associadas à sua conta.

```
Get-SSMAutomationExecutionList
```

Saída:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Exemplo 2: Este exemplo exibe ExecutionID, documento e start/end data e hora de execução para execuções que não sejam 'Success' AutomationExecutionStatus

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Saída:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
---------------------                ------------
 ------------------------- ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
```

```
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi
 Failed                    4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- Para obter detalhes da API, consulte DescribeAutomationExecutionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAutomationStepExecutions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAutomationStepExecutions`.

CLI

AWS CLI

Exemplo 1: descrever todas as etapas de uma execução de automação

O exemplo de `describe-automation-step-executions` a seguir exibe detalhes sobre as etapas de uma execução do Automation.

```
aws ssm describe-automation-step-executions \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Saída:

```
{
    "StepExecutions": [
        {
            "StepName": "startInstances",
            "Action": "aws:changeInstanceState",
            "ExecutionStartTime": 1583737234.134,
            "ExecutionEndTime": 1583737234.672,
            "StepStatus": "Success",
            "Inputs": {
                "DesiredState": "\"running\"",
                "InstanceIds": "[\"i-0cb99161f6EXAMPLE\"]"
            },
            "Outputs": {
                "InstanceStates": [
                    "running"
                ]
```

```
            },
            "StepExecutionId": "95e70479-cf20-4d80-8018-7e4e2EXAMPLE",
            "OverriddenParameters": {}
        }
    ]
}
```

Exemplo 2: descrever uma etapa específica de uma execução do Automation

O exemplo de `describe-automation-step-executions` a seguir exibe detalhes sobre uma etapa específica de uma execução do Automation.

```
aws ssm describe-automation-step-executions \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE \
    --filters Key=StepExecutionId,Values=95e70479-cf20-4d80-8018-7e4e2EXAMPLE
```

Para obter mais informações, consulte Executar um fluxo de trabalho do Automation passo a passo (Linha de comando) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeAutomationStepExecutionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe informações sobre todas as execuções de etapas ativas e encerradas em um fluxo de trabalho do Automation.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Saída:

```
StepName                  Action                  StepStatus
--------                  ------                  ----------
LaunchInstance            aws:runInstances        Success
OSCompatibilityCheck      aws:runCommand          Success
RunPreUpdateScript        aws:runCommand          Success
UpdateEC2Config           aws:runCommand          Cancelled
UpdateSSMAgent            aws:runCommand          Pending
```

```
UpdateAWSPVDriver          aws:runCommand          Pending
UpdateAWSEnaNetworkDriver aws:runCommand          Pending
UpdateAWSNVMe              aws:runCommand          Pending
InstallWindowsUpdates      aws:runCommand          Pending
RunPostUpdateScript        aws:runCommand          Pending
RunSysprepGeneralize       aws:runCommand          Pending
StopInstance               aws:changeInstanceState Pending
CreateImage                aws:createImage         Pending
TerminateInstance          aws:changeInstanceState Pending
```

- Para obter detalhes da API, consulte DescribeAutomationStepExecutionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe informações sobre todas as execuções de etapas ativas e encerradas em um fluxo de trabalho do Automation.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Saída:

```
StepName                   Action                  StepStatus
--------                   ------                  ----------
LaunchInstance             aws:runInstances        Success
OSCompatibilityCheck       aws:runCommand          Success
RunPreUpdateScript         aws:runCommand          Success
UpdateEC2Config            aws:runCommand          Cancelled
UpdateSSMAgent             aws:runCommand          Pending
UpdateAWSPVDriver          aws:runCommand          Pending
UpdateAWSEnaNetworkDriver aws:runCommand          Pending
UpdateAWSNVMe              aws:runCommand          Pending
InstallWindowsUpdates      aws:runCommand          Pending
RunPostUpdateScript        aws:runCommand          Pending
RunSysprepGeneralize       aws:runCommand          Pending
StopInstance               aws:changeInstanceState Pending
CreateImage                aws:createImage         Pending
TerminateInstance          aws:changeInstanceState Pending
```

- Para obter detalhes da API, consulte DescribeAutomationStepExecutionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeAvailablePatches`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeAvailablePatches`.

CLI

AWS CLI

Para obter os patches disponíveis

O exemplo de `describe-available-patches` a seguir recupera detalhes sobre todos os patches disponíveis para o Windows Server 2019 que apresentam gravidade MSRC crítica.

```
aws ssm describe-available-patches \
    --
filters "Key=PRODUCT,Values=WindowsServer2019" "Key=MSRC_SEVERITY,Values=Critical"
```

Saída:

```
{
    "Patches": [
        {
            "Id": "fe6bd8c2-3752-4c8b-ab3e-1a7ed08767ba",
            "ReleaseDate": 1544047205.0,
            "Title": "2018-11 Update for Windows Server 2019 for x64-based
 Systems (KB4470788)",
            "Description": "Install this update to resolve issues in Windows.
 For a complete listing of the issues that are included in this update, see the
 associated Microsoft Knowledge Base article for more information. After you
 install this item, you may have to restart your computer.",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4470788",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2019",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4470788",
            "MsrcNumber": "",
            "Language": "All"
        },
        {
            "Id": "c96115e1-5587-4115-b851-22baa46a3f11",
            "ReleaseDate": 1549994410.0,
```

```
            "Title": "2019-02 Security Update for Adobe Flash Player for Windows
 Server 2019 for x64-based Systems (KB4487038)",
            "Description": "A security issue has been identified in a Microsoft
 software product that could affect your system. You can help protect your system
 by installing this update from Microsoft. For a complete listing of the issues
 that are included in this update, see the associated Microsoft Knowledge Base
 article. After you install this update, you may have to restart your system.",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4487038",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2019",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4487038",
            "MsrcNumber": "",
            "Language": "All"
        },
        ...
    ]
}
```

Para obter detalhes de um patch específico

O exemplo de `describe-available-patches` a seguir recupera detalhes do patch especificado.

```
aws ssm describe-available-patches \
    --filters "Key=PATCH_ID,Values=KB4480979"
```

Saída:

```
{
    "Patches": [
        {
            "Id": "680861e3-fb75-432e-818e-d72e5f2be719",
            "ReleaseDate": 1546970408.0,
            "Title": "2019-01 Security Update for Adobe Flash Player for Windows
 Server 2016 for x64-based Systems (KB4480979)",
            "Description": "A security issue has been identified in a Microsoft
 software product that could affect your system. You can help protect your system
 by installing this update from Microsoft. For a complete listing of the issues
 that are included in this update, see the associated Microsoft Knowledge Base
 article. After you install this update, you may have to restart your system.",
```

```
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4480979",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2016",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4480979",
            "MsrcNumber": "",
            "Language": "All"
        }
    ]
}
```

Para obter mais informações, consulte Como as operações do Patch Manager funcionam no Guia do Usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeAvailablePatchesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém todos os patches disponíveis para o Windows Server 2012 que apresentam gravidade MSRC crítica A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Saída:

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
Description    : A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control
                 over it. You can help protect your system by installing this
 update from Microsoft. After you install this update, you may have to
                 restart your system.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
```

```
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada filtro.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Exemplo 3: Este exemplo busca todas as atualizações lançadas nos últimos 20 dias e aplicáveis aos produtos correspondentes WindowsServer a 2019

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20)
 | Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate,
 Product, Title
```

Saída:

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash
 Player for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows
 Server 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- Para obter detalhes da API, consulte DescribeAvailablePatchesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém todos os patches disponíveis para o Windows Server 2012 que apresentam gravidade MSRC crítica A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Saída:

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
Description    : A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control
                 over it. You can help protect your system by installing this
 update from Microsoft. After you install this update, you may have to
                 restart your system.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada filtro.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
```

```
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Exemplo 3: Este exemplo busca todas as atualizações lançadas nos últimos 20 dias e aplicáveis aos produtos correspondentes WindowsServer a 2019

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20)
 | Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate,
 Product, Title
```

Saída:

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash
 Player for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows
 Server 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- Para obter detalhes da API, consulte DescribeAvailablePatchesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeDocument`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeDocument`.

CLI

AWS CLI

Para exibir detalhes de um documento

O `describe-document` exemplo a seguir exibe detalhes sobre um documento do Systems Manager em sua AWS conta.

```
aws ssm describe-document \
```

```
    --name "Example"
```

Saída:

```
{
    "Document": {
        "Hash":
 "fc2410281f40779e694a8b95975d0f9f316da8a153daa94e3d9921102EXAMPLE",
        "HashType": "Sha256",
        "Name": "Example",
        "Owner": "29884EXAMPLE",
        "CreatedDate": 1583257938.266,
        "Status": "Active",
        "DocumentVersion": "1",
        "Description": "Document Example",
        "Parameters": [
            {
                "Name": "AutomationAssumeRole",
                "Type": "String",
                "Description": "(Required) The ARN of the role that allows
 Automation to perform the actions on your behalf. If no role is specified,
 Systems Manager Automation uses your IAM permissions to execute this document.",
                "DefaultValue": ""
            },
            {
                "Name": "InstanceId",
                "Type": "String",
                "Description": "(Required) The ID of the Amazon EC2 instance.",
                "DefaultValue": ""
            }
        ],
        "PlatformTypes": [
            "Windows",
            "Linux"
        ],
        "DocumentType": "Automation",
        "SchemaVersion": "0.3",
        "LatestVersion": "1",
        "DefaultVersion": "1",
        "DocumentFormat": "YAML",
        "Tags": []
    }
}
```

Para obter mais informações, consulte Criar documentos do Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeDocumentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo retorna informações sobre um documento.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Saída:

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- Para obter detalhes da API, consulte DescribeDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna informações sobre um documento.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Saída:

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- Para obter detalhes da API, consulte DescribeDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_ssm->describedocument( iv_name = iv_name ).
        DATA(lo_document) = lo_result->get_document( ).
        IF lo_document IS BOUND.
          rv_status = lo_document->get_status( ).
          MESSAGE |Document status: { rv_status }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_ssminvaliddocument.
        MESSAGE 'Invalid document.' TYPE 'I'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDocumentreferência da API AWS SDK for SAP ABAP.

Usar **`DescribeDocumentPermission`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeDocumentPermission`.

CLI

AWS CLI

Para descrever permissões do documento

O exemplo de `describe-document-permission` a seguir exibe detalhes de permissão sobre um documento do Systems Manager que é compartilhado publicamente.

```
aws ssm describe-document-permission \
    --name "Example" \
    --permission-type "Share"
```

Saída:

```
{
    "AccountIds": [
        "all"
    ],
    "AccountSharingInfoList": [
        {
            "AccountId": "all",
            "SharedDocumentVersion": "$DEFAULT"
        }
    ]
}
```

Para obter mais informações, consulte Compartilhar um documento do Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeDocumentPermissionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as versões de um documento.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Saída:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- Para obter detalhes da API, consulte DescribeDocumentPermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as versões de um documento.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Saída:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- Para obter detalhes da API, consulte DescribeDocumentPermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeEffectiveInstanceAssociations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeEffectiveInstanceAssociations`.

CLI

AWS CLI

Para obter detalhes das associações efetivas de uma instância

O exemplo de `describe-effective-instance-associations` a seguir recupera detalhes sobre as associações efetivas de uma instância.

Comando:

```
aws ssm describe-effective-instance-associations --instance-
id "i-1234567890abcdef0"
```

Saída:

```
{
    "Associations": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "InstanceId": "i-1234567890abcdef0",
            "Content": "{\n    \"schemaVersion\": \"1.2\",\n    \"description\":
 \"Update the Amazon SSM Agent to the latest version or specified version.\",\n
  \"parameters\": {\n        \"version\": {\n            \"default\": \"\",\n
        \"description\": \"(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agent will be updated to the latest version.
\",\n            \"type\": \"String\"\n        },\n        \"allowDowngrade
\": {\n            \"default\": \"false\",\n            \"description\":
 \"(Optional) Allow the Amazon SSM Agent service to be downgraded to an earlier
 version. If set to false, the service can be upgraded to newer versions only
 (default). If set to true, specify the earlier version.\",\n            \"type
\": \"String\",\n            \"allowedValues\": [\n                \"true\",\n
               \"false\"\n            ]\n        }\n    },\n    \"runtimeConfig
\": {\n        \"aws:updateSsmAgent\": {\n            \"properties\": [\n
         {\n                \"agentName\": \"amazon-ssm-agent\",\n
  \"source\": \"https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-
manifest.json\",\n                \"allowDowngrade\": \"{{ allowDowngrade }}\",\n
                \"targetVersion\": \"{{ version }}\"\n                }\n
    ]\n        }\n    }\n}\n",
            "AssociationVersion": "1"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeEffectiveInstanceAssociationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo descreve as associações efetivas de uma instância.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5
```

Saída:

```
AssociationId                       Content
-------------                       -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Exemplo 2: esse exemplo exibe o conteúdo das associações efetivas de uma instância.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Saída:

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or
 specified version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "allowDowngrade": {
            "default": "false",
            "description": "(Optional) Allow the Amazon SSM Agent service to be
 downgraded to an earlier version. If set
 to false, the service can be upgraded to newer versions only (default). If set
 to true, specify the earlier version.",
            "type": "String",
            "allowedValues": [
```

```
                "true",
                "false"
            ]
        }
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeEffectiveInstanceAssociationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as associações efetivas de uma instância.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5
```

Saída:

```
AssociationId                       Content
-------------                       -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Exemplo 2: esse exemplo exibe o conteúdo das associações efetivas de uma instância.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Saída:

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or
 specified version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "allowDowngrade": {
            "default": "false",
            "description": "(Optional) Allow the Amazon SSM Agent service to be
 downgraded to an earlier version. If set
 to false, the service can be upgraded to newer versions only (default). If set
 to true, specify the earlier version.",
            "type": "String",
            "allowedValues": [
                "true",
                "false"
            ]
        }
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeEffectiveInstanceAssociationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeEffectivePatchesForPatchBaseline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeEffectivePatchesForPatchBaseline`.

CLI

AWS CLI

Exemplo 1: obter todos os patches definidos por uma lista de referência de patches

O `describe-effective-patches-for-patch-baseline` exemplo a seguir retorna os patches definidos por uma linha de base de patch personalizada na AWS conta atual. Observe que, para uma lista de referência personalizada, somente a ID é necessária para `--baseline-id`.

```
aws ssm describe-effective-patches-for-patch-baseline \
    --baseline-id "pb-08b654cf9b9681f04"
```

Saída:

```
{
    "EffectivePatches": [
        {
            "Patch": {
                "Id": "fe6bd8c2-3752-4c8b-ab3e-1a7ed08767ba",
                "ReleaseDate": 1544047205.0,
                "Title": "2018-11 Update for Windows Server 2019 for x64-based
 Systems (KB4470788)",
                "Description": "Install this update to resolve issues in Windows.
 For a complete listing of the issues that are included in this update, see the
 associated Microsoft Knowledge Base article for more information. After you
 install this item, you may have to restart your computer.",
                "ContentUrl": "https://support.microsoft.com/en-us/kb/4470788",
                "Vendor": "Microsoft",
                "ProductFamily": "Windows",
                "Product": "WindowsServer2019",
                "Classification": "SecurityUpdates",
                "MsrcSeverity": "Critical",
                "KbNumber": "KB4470788",
                "MsrcNumber": "",
                "Language": "All"
```

```
            },
            "PatchStatus": {
                "DeploymentStatus": "APPROVED",
                "ComplianceLevel": "CRITICAL",
                "ApprovalDate": 1544047205.0
            }
        },
        {
            "Patch": {
                "Id": "915a6b1a-f556-4d83-8f50-b2e75a9a7e58",
                "ReleaseDate": 1549994400.0,
                "Title": "2019-02 Cumulative Update for .NET Framework 3.5 and
 4.7.2 for Windows Server 2019 for x64 (KB4483452)",
                "Description": "A security issue has been identified in a
 Microsoft software product that could affect your system. You can help protect
 your system by installing this update from Microsoft. For a complete listing
 of the issues that are included in this update, see the associated Microsoft
 Knowledge Base article. After you install this update, you may have to restart
 your system.",
                "ContentUrl": "https://support.microsoft.com/en-us/kb/4483452",
                "Vendor": "Microsoft",
                "ProductFamily": "Windows",
                "Product": "WindowsServer2019",
                "Classification": "SecurityUpdates",
                "MsrcSeverity": "Important",
                "KbNumber": "KB4483452",
                "MsrcNumber": "",
                "Language": "All"
            },
            "PatchStatus": {
                "DeploymentStatus": "APPROVED",
                "ComplianceLevel": "CRITICAL",
                "ApprovalDate": 1549994400.0
            }
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

Exemplo 2: Para obter todos os patches definidos por uma linha de base de patches AWS gerenciados

O `describe-effective-patches-for-patch-baseline` exemplo a seguir retorna os patches definidos por uma linha de base de patches AWS gerenciados. Observe que, para uma linha de base AWS gerenciada, o ARN completo da linha de base é necessário para `--baseline-id`

```
aws ssm describe-effective-patches-for-patch-baseline \
    --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-020d361a05defe4ed"
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Como os patches de segurança são selecionados no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeEffectivePatchesForPatchBaselineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra todas as listas de referência de patches, com uma lista de resultados máxima de 1.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Saída:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Exemplo 2: esse exemplo mostra o status do patch para todas as listas de referência de patches, com uma lista de resultados máxima de 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Saída:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- Para obter detalhes da API, consulte DescribeEffectivePatchesForPatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra todas as listas de referência de patches, com uma lista de resultados máxima de 1.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Saída:

```
Patch                                     PatchStatus
-----                                     -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Exemplo 2: esse exemplo mostra o status do patch para todas as listas de referência de patches, com uma lista de resultados máxima de 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Saída:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- Para obter detalhes da API, consulte DescribeEffectivePatchesForPatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeInstanceAssociationsStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstanceAssociationsStatus`.

CLI

AWS CLI

Para descrever o status das associações de uma instância

Este exemplo mostra detalhes das associações de uma instância.

Comando:

```
aws ssm describe-instance-associations-status --instance-id "i-1234567890abcdef0"
```

Saída:

```
{
  "InstanceAssociationStatusInfos": [
      {
          "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
          "Name": "AWS-GatherSoftwareInventory",
          "DocumentVersion": "1",
          "AssociationVersion": "1",
          "InstanceId": "i-1234567890abcdef0",
          "ExecutionDate": 1550501886.0,
          "Status": "Success",
          "ExecutionSummary": "1 out of 1 plugin processed, 1 success, 0 failed,
 0 timedout, 0 skipped. ",
          "AssociationName": "Inventory-Association"
      },
      {
          "AssociationId": "5c5a31f6-6dae-46f9-944c-0123456789ab",
          "Name": "AWS-UpdateSSMAgent",
          "DocumentVersion": "1",
          "AssociationVersion": "1",
          "InstanceId": "i-1234567890abcdef0",
          "ExecutionDate": 1550505828.548,
          "Status": "Success",
          "DetailedStatus": "Success",
```

```
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

- Para obter detalhes da API, consulte DescribeInstanceAssociationsStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra detalhes das associações de uma instância.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Saída:

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
Name             : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Exemplo 2: esse exemplo verifica o status da associação da instância para o ID da instância fornecido e exibe o status de execução dessas associações

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-
Object {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Saída:

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion    : 2
```

```
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status                : Success
```

- Para obter detalhes da API, consulte DescribeInstanceAssociationsStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra detalhes das associações de uma instância.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Saída:

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
Name             : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Exemplo 2: esse exemplo verifica o status da associação da instância para o ID da instância fornecido e exibe o status de execução dessas associações

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-
Object {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Saída:

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
```

```
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status                : Success
```

- Para obter detalhes da API, consulte DescribeInstanceAssociationsStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeInstanceInformation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstanceInformation`.

CLI

AWS CLI

Exemplo 1: descrever as informações da instância gerenciada

O exemplo de `describe-instance-information` a seguir recupera detalhes de cada uma das suas instâncias gerenciadas.

```
aws ssm describe-instance-information
```

Exemplo 2: descrever informações sobre uma instância gerenciada específica

O exemplo de `describe-instance-information` a seguir mostra os detalhes da instância gerenciada `i-028ea792daEXAMPLE`.

```
aws ssm describe-instance-information \
    --filters "Key=InstanceIds,Values=i-028ea792daEXAMPLE"
```

Exemplo 3: descrever informações sobre instâncias gerenciadas com uma chave de tag específica

O exemplo de `describe-instance-information` a seguir mostra detalhes de instâncias gerenciadas que têm a chave de tag `DEV`.

```
aws ssm describe-instance-information \
    --filters "Key=tag-key,Values=DEV"
```

Saída:

```
{
    "InstanceInformationList": [
        {
            "InstanceId": "i-028ea792daEXAMPLE",
            "PingStatus": "Online",
            "LastPingDateTime": 1582221233.421,
            "AgentVersion": "2.3.842.0",
            "IsLatestVersion": true,
            "PlatformType": "Linux",
            "PlatformName": "SLES",
            "PlatformVersion": "15.1",
            "ResourceType": "EC2Instance",
            "IPAddress": "192.0.2.0",
            "ComputerName": "ip-198.51.100.0.us-east-2.compute.internal",
            "AssociationStatus": "Success",
            "LastAssociationExecutionDate": 1582220806.0,
            "LastSuccessfulAssociationExecutionDate": 1582220806.0,
            "AssociationOverview": {
                "DetailedStatus": "Success",
                "InstanceAssociationStatusAggregatedCount": {
                    "Success": 2
                }
            }
        }
    ]
}
```

Para obter mais informações, consulte Instâncias gerenciadas no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeInstanceInformationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra detalhes de cada uma de suas instâncias.

```
Get-SSMInstanceInformation
```

Saída:

```
ActivationId                            :
AgentVersion                            : 2.0.672.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : ip-172-31-44-222.us-
west-2.compute.internal
IamRole                                 :
InstanceId                              : i-0cb2b964d3e14fd9f
IPAddress                               : 172.31.44.222
IsLatestVersion                         : True
LastAssociationExecutionDate            : 2/24/2017 3:18:09 AM
LastPingDateTime                        : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                    :
PingStatus                              : ConnectionLost
PlatformName                            : Amazon Linux AMI
PlatformType                            : Linux
PlatformVersion                         : 2016.09
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Exemplo 2: Este exemplo mostra como usar o parâmetro -Filter para filtrar os resultados somente para as instâncias do AWS Systems Manager na região **`us-east-1`** com um **`AgentVersion`** de**`2.2.800.0`**. Você pode encontrar uma lista de valores-chave válidos de -Filter no tópico de referência da InstanceInformation API (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager -Type-InstanceInformation-ActivationId).

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Saída:

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
```

```
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEb0792d98ce
IPAddress                              : 10.0.0.01
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:02:50 AM
LastPingDateTime                       : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance

ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEac7501d023
IPAddress                              : 10.0.0.02
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:00:20 AM
LastPingDateTime                       : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Exemplo 3: Este exemplo mostra como usar o InstanceInformationFilterList parâmetro - para filtrar os resultados somente para as instâncias do AWS Systems Manager na região **`us-east-1`** com **`PlatformTypes`** **`Windows`** ou**`Linux`**. Você pode encontrar

uma lista de InstanceInformationFilterList valores-chave válidos no tópico de referência da InstanceInformationFilter API (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList
 $Filters
```

Saída:

```
ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEb0792d98ce
IPAddress                             : 10.0.0.27
IsLatestVersion                       : False
LastAssociationExecutionDate          : 8/16/2018 12:02:50 AM
LastPingDateTime                      : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                  :
PingStatus                            : Online
PlatformName                          : Ubuntu Server 18.04 LTS
PlatformType                          : Linux
PlatformVersion                       : 18.04
RegistrationDate                      : 1/1/0001 12:00:00 AM
ResourceType                          : EC2Instance

ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEac7501d023
IPAddress                             : 10.0.0.100
```

```
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:00:20 AM
LastPingDateTime                        : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Exemplo 4: Este exemplo lista as instâncias e exportações InstanceId gerenciadas por ssm LastPingDateTime e PlatformName em um arquivo csv. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus,
 LastPingDateTime, PlatformName | Export-Csv Instance-details.csv -
NoTypeInformation
```

- Para obter detalhes da API, consulte DescribeInstanceInformationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra detalhes de cada uma de suas instâncias.

```
Get-SSMInstanceInformation
```

Saída:

```
ActivationId                            :
AgentVersion                            : 2.0.672.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : ip-172-31-44-222.us-
west-2.compute.internal
IamRole                                 :
InstanceId                              : i-0cb2b964d3e14fd9f
IPAddress                               : 172.31.44.222
IsLatestVersion                         : True
LastAssociationExecutionDate            : 2/24/2017 3:18:09 AM
```

```
LastPingDateTime                       : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                   :
PingStatus                             : ConnectionLost
PlatformName                           : Amazon Linux AMI
PlatformType                           : Linux
PlatformVersion                        : 2016.09
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Exemplo 2: Este exemplo mostra como usar o parâmetro -Filter para filtrar os resultados somente para as instâncias do AWS Systems Manager na região **`us-east-1`** com um **`AgentVersion`** de**`2.2.800.0`**. Você pode encontrar uma lista de valores-chave válidos de -Filter no tópico de referência da InstanceInformation API (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager -Type-InstanceInformation-ActivationId).

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Saída:

```
ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEb0792d98ce
IPAddress                              : 10.0.0.01
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:02:50 AM
LastPingDateTime                       : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
```

```
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance

ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEac7501d023
IPAddress                              : 10.0.0.02
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:00:20 AM
LastPingDateTime                       : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Exemplo 3: Este exemplo mostra como usar o InstanceInformationFilterList parâmetro - para filtrar os resultados somente para as instâncias do AWS Systems Manager na região **`us-east-1`** com **`PlatformTypes`** **`Windows`** ou**`Linux`**. Você pode encontrar uma lista de InstanceInformationFilterList valores-chave válidos no tópico de referência da InstanceInformationFilter API (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList
 $Filters
```

Saída:

```
ActivationId                           :
```

```
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEb0792d98ce
IPAddress                               : 10.0.0.27
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:02:50 AM
LastPingDateTime                        : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Ubuntu Server 18.04 LTS
PlatformType                            : Linux
PlatformVersion                         : 18.04
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance

ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEac7501d023
IPAddress                               : 10.0.0.100
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:00:20 AM
LastPingDateTime                        : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Exemplo 4: Este exemplo lista as instâncias e exportações InstanceId gerenciadas por ssm LastPingDateTime e PlatformName em um arquivo csv. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus,
 LastPingDateTime, PlatformName | Export-Csv Instance-details.csv -
NoTypeInformation
```

- Para obter detalhes da API, consulte DescribeInstanceInformationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeInstancePatchStates`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstancePatchStates`.

CLI

AWS CLI

Para obter os estados resumidos de patches para instâncias

Este exemplo de `describe-instance-patch-states` obtém os estados resumidos de patches para uma instância.

```
aws ssm describe-instance-patch-states \
    --instance-ids "i-1234567890abcdef0"
```

Saída:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PatchGroup": "my-patch-group",
            "BaselineId": "pb-0713accee01234567",
            "SnapshotId": "521c3536-930c-4aa9-950e-01234567abcd",
            "CriticalNonCompliantCount": 2,
            "SecurityNonCompliantCount": 2,
            "OtherNonCompliantCount": 1,
            "InstalledCount": 123,
            "InstalledOtherCount": 334,
            "InstalledPendingRebootCount": 0,
            "InstalledRejectedCount": 0,
            "MissingCount": 1,
            "FailedCount": 2,
```

```
            "UnreportedNotApplicableCount": 11,
            "NotApplicableCount": 2063,
            "OperationStartTime": "2021-05-03T11:00:56-07:00",
            "OperationEndTime": "2021-05-03T11:01:09-07:00",
            "Operation": "Scan",
            "LastNoRebootInstallOperationTime": "2020-06-14T12:17:41-07:00",
            "RebootOption": "RebootIfNeeded"
        }
    ]
}
```

Para obter mais informações, consulte Sobre a conformidade de patches no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeInstancePatchStatesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém os estados resumidos de patches para uma instância.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Exemplo 2: esse exemplo obtém os estados resumidos de patches para duas instâncias.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Para obter detalhes da API, consulte DescribeInstancePatchStatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os estados resumidos de patches para uma instância.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Exemplo 2: esse exemplo obtém os estados resumidos de patches para duas instâncias.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Para obter detalhes da API, consulte DescribeInstancePatchStatesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeInstancePatchStatesForPatchGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstancePatchStatesForPatchGroup`.

CLI

AWS CLI

Exemplo 1: obter os estados da instância de um grupo de patches

O exemplo de `describe-instance-patch-states-for-patch-group` a seguir recupera detalhes sobre os estados resumidos de patches por instância para o grupo de patches especificado.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --patch-group "Production"
```

Saída:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 32,
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 0,
            "InstalledRejectedCount": 0,
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 2671,
            "NotApplicableCount": 400,
            "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-04T11:04:21.555000-07:00",
            "Operation": "Scan",
```

```
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        },
        {
            "InstanceId": "i-0471e04240EXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-09ca3fb51fEXAMPLE",
            "SnapshotId": "05d8ffb0-1bbe-4812-ba2d-d9b7bEXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 32,
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 0,
            "InstalledRejectedCount": 0,
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 2671,
            "NotApplicableCount": 400,
            "OperationStartTime": "2021-08-04T22:06:20.340000-07:00",
            "OperationEndTime": "2021-08-04T22:07:11.220000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        }
    ]
}
```

Exemplo 2: obter os estados da instância de um grupo de patches com mais de cinco patches ausentes

O exemplo de `describe-instance-patch-states-for-patch-group` a seguir recupera detalhes sobre os estados resumidos de patches para o grupo de patches especificado por instâncias com mais de cinco patches ausentes.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --filters Key=MissingCount,Type=GreaterThan,Values=5 \
    --patch-group "Production"
```

Saída:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 46,
            "InstalledOtherCount": 4,
            "InstalledPendingRebootCount": 1,
            "InstalledRejectedCount": 1,
            "MissingCount": 7,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 232,
            "NotApplicableCount": 654,
            "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-04T11:04:21.555000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 1
        }
    ]
}
```

Exemplo 3: obter os estados da instância de um grupo de patches com menos de dez instâncias que exigem uma reinicialização

O exemplo de `describe-instance-patch-states-for-patch-group` a seguir recupera detalhes sobre os estados resumidos de patches para o grupo de patches especificado por instâncias com menos de dez instâncias que exigem uma reinicialização.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --filters Key=InstalledPendingRebootCount,Type=LessThan,Values=10 \
    --patch-group "Production"
```

Saída:

```
{
    "InstancePatchStates": [
```

```
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "PatchGroup": "Production",
            "OwnerInformation": "",
            "InstalledCount": 32,
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 4,
            "InstalledRejectedCount": 0,
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 846,
            "NotApplicableCount": 212,
            "OperationStartTime": "2021-08-046T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-06T11:04:21.555000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        }
    ]
}
```

Para obter mais informações, consulte Noções básicas sobre valores de estado de conformidade de patches no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeInstancePatchStatesForPatchGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém os estados de resumo de patches por instância de um grupo de patches.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Para obter detalhes da API, consulte DescribeInstancePatchStatesForPatchGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os estados de resumo de patches por instância de um grupo de patches.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Para obter detalhes da API, consulte DescribeInstancePatchStatesForPatchGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeInstancePatches`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeInstancePatches`.

CLI

AWS CLI

Exemplo 1: obter os detalhes do estado do patch para uma instância

O exemplo de `describe-instance-patches` a seguir recupera detalhes sobre os patches para a instância especificada.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0"
```

Saída:

```
{
    "Patches": [
        {
            "Title": "2019-01 Security Update for Adobe Flash Player for Windows
 Server 2016 for x64-based Systems (KB4480979)",
            "KBId": "KB4480979",
            "Classification": "SecurityUpdates",
            "Severity": "Critical",
            "State": "Installed",
            "InstalledTime": "2019-01-09T00:00:00+00:00"
        },
        {
```

```
            "Title": "",
            "KBId": "KB4481031",
            "Classification": "",
            "Severity": "",
            "State": "InstalledOther",
            "InstalledTime": "2019-02-08T00:00:00+00:00"
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

Exemplo 2: obter uma lista de patches no estado Ausente para uma instância

O exemplo de `describe-instance-patches` a seguir recupera informações sobre patches que estão no estado Ausente para a instância especificada.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0" \
    --filters Key=State,Values=Missing
```

Saída:

```
{
    "Patches": [
        {
            "Title": "Windows Malicious Software Removal Tool x64 - February 2019
 (KB890830)",
            "KBId": "KB890830",
            "Classification": "UpdateRollups",
            "Severity": "Unspecified",
            "State": "Missing",
            "InstalledTime": "1970-01-01T00:00:00+00:00"
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

Para obter mais informações, consulte Sobre estados de conformidade de patches no Guia do usuário do AWS Systems Manager.

Exemplo 3: Para obter uma lista de patches instalados desde um especificado InstalledTime para uma instância

O exemplo de `describe-instance-patches` a seguir recupera informações sobre patches instalados desde um horário especificado para a instância especificada combinando o uso de `--filters` e `--query`.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0" \
    --filters Key=State,Values=Installed \
    --query "Patches[?InstalledTime >= `2023-01-01T16:00:00`]"
```

Saída:

```
{
    "Patches": [
        {
            "Title": "2023-03 Cumulative Update for Windows Server 2019 (1809)
 for x64-based Systems (KB5023702)",
            "KBId": "KB5023702",
            "Classification": "SecurityUpdates",
            "Severity": "Critical",
            "State": "Installed",
            "InstalledTime": "2023-03-16T11:00:00+00:00"
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

- Para obter detalhes da API, consulte DescribeInstancePatchesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém os detalhes de conformidade do patch para uma instância.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Para obter detalhes da API, consulte DescribeInstancePatchesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os detalhes de conformidade do patch para uma instância.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Para obter detalhes da API, consulte DescribeInstancePatchesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeMaintenanceWindowExecutionTaskInvocations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeMaintenanceWindowExecutionTaskInvocations`.

CLI

AWS CLI

Para fazer com que as invocações da tarefa específica sejam realizadas para a execução de tarefa de uma janela de manutenção

O exemplo de `describe-maintenance-window-execution-task-invocations` a seguir lista as invocações para a tarefa especificada executada como parte da execução da janela de manutenção especificada.

```
aws ssm describe-maintenance-window-execution-task-invocations \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2a638355" \
    --task-id "ac0c6ae1-daa3-4a89-832e-d384503b6586"
```

Saída:

```
{
    "WindowExecutionTaskInvocationIdentities": [
        {
            "Status": "SUCCESS",
            "Parameters": "{\"documentName\":\"AWS-RunShellScript\",
\"instanceIds\":[\"i-0000293ffd8c57862\"],\"parameters\":{\"commands\":[\"df\"]},
\"maxConcurrency\":\"1\",\"maxErrors\":\"1\"}",
            "InvocationId": "e274b6e1-fe56-4e32-bd2a-8073c6381d8b",
```

```
            "StartTime": 1487692834.723,
            "EndTime": 1487692834.871,
            "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2a638355",
            "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d384503b6586"
        }
    ]
}
```

Para obter mais informações, consulte Exibir informações sobre tarefas e execuções de tarefas (AWS CLI) no Guia do usuário AWS do Systems Manager.

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionTaskInvocationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista as invocações de uma tarefa executada como parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionTaskInvocationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as invocações de uma tarefa executada como parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionTaskInvocationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeMaintenanceWindowExecutionTasks`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeMaintenanceWindowExecutionTasks`.

CLI

AWS CLI

Para listar todas as tarefas associadas à execução de uma janela de manutenção

O exemplo de `ssm describe-maintenance-window-execution-tasks` a seguir lista as tarefas associadas à execução da janela de manutenção especificada.

```
aws ssm describe-maintenance-window-execution-tasks \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE"
```

Saída:

```
{
    "WindowExecutionTaskIdentities": [
        {
            "Status": "SUCCESS",
            "TaskArn": "AWS-RunShellScript",
            "StartTime": 1487692834.684,
            "TaskType": "RUN_COMMAND",
            "EndTime": 1487692835.005,
            "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
            "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
        }
    ]
}
```

Para obter mais informações, consulte Exibir informações sobre tarefas e execuções de tarefas (AWS CLI) no Guia do usuário AWS do Systems Manager.

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionTasksem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as tarefas associadas à execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as tarefas associadas à execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeMaintenanceWindowExecutions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeMaintenanceWindowExecutions`.

CLI

AWS CLI

Exemplo 1: listar todas as execuções para uma janela de manutenção

O exemplo de `describe-maintenance-window-executions` a seguir lista todas as execuções da janela de manutenção especificada.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE"
```

Saída:

```
{
    "WindowExecutions": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "6027b513-64fe-4cf0-be7d-1191aEXAMPLE",
            "Status": "IN_PROGRESS",
            "StartTime": "2021-08-04T11:00:00.000000-07:00"

        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "ff75b750-4834-4377-8f61-b3cadEXAMPLE",
            "Status": "SUCCESS",
            "StartTime": "2021-08-03T11:00:00.000000-07:00",
            "EndTime": "2021-08-03T11:37:21.450000-07:00"
        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "9fac7dd9-ff21-42a5-96ad-bbc4bEXAMPLE",
            "Status": "FAILED",
            "StatusDetails": "One or more tasks in the orchestration failed.",
            "StartTime": "2021-08-02T11:00:00.000000-07:00",
            "EndTime": "2021-08-02T11:22:36.190000-07:00"
        }
    ]
}
```

Exemplo 2: listar todas as execuções para uma janela de manutenção antes de uma data especificada

O exemplo de `describe-maintenance-window-executions` a seguir lista todas as execuções da janela de manutenção especificada antes da data especificada.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=ExecutedBefore,Values=2021-08-03T00:00:00Z"
```

Saída:

```
{
    "WindowExecutions": [
        {
        "WindowId": "mw-ab12cd34eEXAMPLE",
        "WindowExecutionId": "9fac7dd9-ff21-42a5-96ad-bbc4bEXAMPLE",
        "Status": "FAILED",
        "StatusDetails": "One or more tasks in the orchestration failed.",
        "StartTime": "2021-08-02T11:00:00.000000-07:00",
        "EndTime": "2021-08-02T11:22:36.190000-07:00"
    }
    ]
}
```

Exemplo 3: listar todas as execuções para uma janela de manutenção após uma data especificada

O exemplo de `describe-maintenance-window-executions` a seguir lista todas as execuções da janela de manutenção especificada após a data especificada.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=ExecutedAfter,Values=2021-08-04T00:00:00Z"
```

Saída:

```
{
    "WindowExecutions": [
        {
        "WindowId": "mw-ab12cd34eEXAMPLE",
        "WindowExecutionId": "6027b513-64fe-4cf0-be7d-1191aEXAMPLE",
        "Status": "IN_PROGRESS",
```

```
        "StartTime": "2021-08-04T11:00:00.000000-07:00"
        }
    ]
}
```

Para obter mais informações, consulte Exibir informações sobre tarefas e execuções de tarefas (AWS CLI) no Guia do usuário AWS do Systems Manager.

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as execuções para uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Saída:

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Exemplo 2: esse exemplo lista todas as execuções para uma janela de manutenção antes de uma data especificada.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Exemplo 3: esse exemplo lista todas as execuções para uma janela de manutenção após uma data especificada.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
```

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as execuções para uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Saída:

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Exemplo 2: esse exemplo lista todas as execuções para uma janela de manutenção antes de uma data especificada.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Exemplo 3: esse exemplo lista todas as execuções para uma janela de manutenção após uma data especificada.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowExecutionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeMaintenanceWindowTargets`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeMaintenanceWindowTargets`.

CLI

AWS CLI

Exemplo 1: listar todos os destinos para uma janela de manutenção

O exemplo de `describe-maintenance-window-targets` a seguir lista todos os destinos para uma janela de manutenção.

```
aws ssm describe-maintenance-window-targets \
    --window-id "mw-06cf17cbefEXAMPLE"
```

Saída:

```
{
    "Targets": [
        {
            "ResourceType": "INSTANCE",
            "OwnerInformation": "Single instance",
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "Targets": [
                {
                    "Values": [
                        "i-0000293ffdEXAMPLE"
                    ],
                    "Key": "InstanceIds"
                }
            ],
            "WindowTargetId": "350d44e6-28cc-44e2-951f-4b2c9EXAMPLE"
        },
        {
            "ResourceType": "INSTANCE",
            "OwnerInformation": "Two instances in a list",
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "Targets": [
                {
                    "Values": [
                        "i-0000293ffdEXAMPLE",
                        "i-0cb2b964d3EXAMPLE"
                    ],
                    "Key": "InstanceIds"
                }
            ],
```

```
            "WindowTargetId": "e078a987-2866-47be-bedd-d9cf4EXAMPLE"
        }
    ]
}
```

Exemplo 2: listar todos os destinos para uma janela de manutenção que correspondem ao valor das informações de um proprietário específico

Esse exemplo de `describe-maintenance-window-targets` lista todos os destinos de uma janela de manutenção com um valor específico.

```
aws ssm describe-maintenance-window-targets \
    --window-id "mw-0ecb1226ddEXAMPLE" \
    --filters "Key=OwnerInformation,Values=CostCenter1"
```

Saída:

```
{
    "Targets": [
        {
            "WindowId": "mw-0ecb1226ddEXAMPLE",
            "WindowTargetId": "da89dcc3-7f9c-481d-ba2b-edcb7d0057f9",
            "ResourceType": "INSTANCE",
            "Targets": [
                {
                    "Key": "tag:Environment",
                    "Values": [
                        "Prod"
                    ]
                }
            ],
            "OwnerInformation": "CostCenter1",
            "Name": "ProdTarget1"
        }
    ]
}
```

Para obter mais informações, consulte Exibir informações sobre janelas de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeMaintenanceWindowTargetsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todos os destinos para uma janela de manutenção.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os destinos para uma janela de manutenção.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
```

```
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowTargetsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeMaintenanceWindowTasks`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeMaintenanceWindowTasks`.

CLI

AWS CLI

Exemplo 1: listar todas as tarefas para uma janela de manutenção

O exemplo de `describe-maintenance-window-tasks` a seguir lista todas as tarefas para a janela de manutenção especificada.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-06cf17cbefEXAMPLE"
```

Saída:

```
{
    "Tasks": [
        {
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "WindowTaskId": "018b31c3-2d77-4b9e-bd48-c91edEXAMPLE",
            "TaskArn": "AWS-RestartEC2Instance",
            "TaskParameters": {},
            "Type": "AUTOMATION",
            "Description": "Restarting EC2 Instance for maintenance",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "My-Automation-Example-Task",
            "Priority": 0,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
```

```
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ]
        },
        {
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "WindowTaskId": "1943dee0-0a17-4978-9bf4-3cc2fEXAMPLE",
            "TaskArn": "AWS-DisableS3BucketPublicReadWrite",
            "TaskParameters": {},
            "Type": "AUTOMATION",
            "Description": "Automation task to disable read/write access on
 public S3 buckets",
            "MaxConcurrency": "10",
            "MaxErrors": "5",
            "Name": "My-Disable-S3-Public-Read-Write-Access-Automation-Task",
            "Priority": 0,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ]
        }
    ]
}
```

Exemplo 2: Para listar todas as tarefas de uma janela de manutenção que invoca o documento de comando AWS- RunPowerShellScript

O exemplo de `describe-maintenance-window-tasks` a seguir lista todas as tarefas para a janela de manutenção especificada que invoca o documento de comandos do `AWS-RunPowerShellScript`.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=TaskArn,Values=AWS-RunPowerShellScript"
```

Saída:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 1,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyTask"
        }
    ]
}
```

Exemplo 3: listar todas as tarefas de uma janela de manutenção que têm a prioridade igual a 3

O exemplo de `describe-maintenance-window-tasks` a seguir lista todas as tarefas para a janela de manutenção especificada que tem `Priority` igual a `3`.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=Priority,Values=3"
```

Saída:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
```

```
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 3,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyRunCommandTask"
        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "ee45feff-ad65-4a6c-b478-5cab8EXAMPLE",
            "TaskArn": "AWS-RestartEC2Instance",
            "Type": "AUTOMATION",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 3,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "10",
            "MaxErrors": "5",
            "Name": "My-Automation-Task",
            "Description": "A description for my Automation task"
        }
    ]
}
```

Exemplo 4: listar todas as tarefas de uma janela de manutenção que têm a prioridade igual a 1 e usam o Run Command

Esse exemplo de `describe-maintenance-window-tasks` lista todas as tarefas para a janela de manutenção especificada que tem `Priority` igual a 1 e usam `Run Command`.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"
```

Saída:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 1,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyRunCommandTask"
        }
    ]
}
```

Para obter mais informações, consulte Exibir informações sobre janelas de manutenção (AWS CLI) no Guia do Usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeMaintenanceWindowTasksem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as tarefas para uma janela de manutenção.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
LoggingInfo    :
MaxConcurrency : 1
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as tarefas para uma janela de manutenção.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
LoggingInfo    :
MaxConcurrency : 1
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
```

```
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowTasksem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeMaintenanceWindows`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeMaintenanceWindows`.

CLI

AWS CLI

Exemplo 1: listar todas as janelas de manutenção

O `describe-maintenance-windows` exemplo a seguir lista todas as janelas de manutenção em sua AWS conta na região atual.

```
aws ssm describe-maintenance-windows
```

Saída:

```
{
    "WindowIdentities": [
        {
            "WindowId": "mw-0ecb1226ddEXAMPLE",
            "Name": "MyMaintenanceWindow-1",
            "Enabled": true,
            "Duration": 2,
            "Cutoff": 1,
            "Schedule": "rate(180 minutes)",
            "NextExecutionTime": "2020-02-12T23:19:20.596Z"
        },
        {
            "WindowId": "mw-03eb9db428EXAMPLE",
            "Name": "MyMaintenanceWindow-2",
            "Enabled": true,
```

```
            "Duration": 3,
            "Cutoff": 1,
            "Schedule": "rate(7 days)",
            "NextExecutionTime": "2020-02-17T23:22:00.956Z"
        },
    ]
}
```

Exemplo 2: listar todas as janelas de manutenção habilitadas

O exemplo de `describe-maintenance-windows` a seguir lista todas as janelas de manutenção habilitadas.

```
aws ssm describe-maintenance-windows \
    --filters "Key=Enabled,Values=true"
```

Exemplo 3: listar janelas de manutenção que correspondem a um nome específico

Esse exemplo de `describe-maintenance-windows` lista todas as janelas de manutenção com o nome especificado.

```
aws ssm describe-maintenance-windows \
    --filters "Key=Name,Values=MyMaintenanceWindow"
```

Para obter mais informações, consulte Exibir informações sobre janelas de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeMaintenanceWindowsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as janelas de manutenção em sua conta.

```
Get-SSMMaintenanceWindowList
```

Saída:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as janelas de manutenção em sua conta.

```
Get-SSMMaintenanceWindowList
```

Saída:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- Para obter detalhes da API, consulte DescribeMaintenanceWindowsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`DescribeOpsItems`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeOpsItems`.

CLI

AWS CLI

Para listar um conjunto de OpsItems

O `describe-ops-items` exemplo a seguir exibe uma lista de todos os abertos OpsItems em sua AWS conta.

```
aws ssm describe-ops-items \
```

```
    --ops-item-filters "Key=Status,Values=Open,Operator=Equal"
```

Saída:

```
{
    "OpsItemSummaries": [
        {
            "CreatedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-
Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "CreatedTime": "2020-03-14T17:02:46.375000-07:00",
            "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-
CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "LastModifiedTime": "2020-03-14T17:02:46.375000-07:00",
            "Source": "SSM",
            "Status": "Open",
            "OpsItemId": "oi-7cfc5EXAMPLE",
            "Title": "SSM Maintenance Window execution failed",
            "OperationalData": {
                "/aws/dedup": {
                    "Value": "{\"dedupString\":\"SSMOpsItems-SSM-maintenance-
window-execution-failed\"}",
                    "Type": "SearchableString"
                },
                "/aws/resources": {
                    "Value": "[{\"arn\":\"arn:aws:ssm:us-
east-2:111222333444:maintenancewindow/mw-034093d322EXAMPLE\"}]",
                    "Type": "SearchableString"
                }
            },
            "Category": "Availability",
            "Severity": "3"
        },
        {
            "CreatedBy": "arn:aws:sts::1112223233444:assumed-role/OpsItem-CWE-
Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "CreatedTime": "2020-02-26T11:43:15.426000-08:00",
            "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-
CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "LastModifiedTime": "2020-02-26T11:43:15.426000-08:00",
            "Source": "EC2",
            "Status": "Open",
            "OpsItemId": "oi-6f966EXAMPLE",
            "Title": "EC2 instance stopped",
```

```
            "OperationalData": {
                "/aws/automations": {
                    "Value": "[ { \"automationType\": \"AWS:SSM:Automation\",
 \"automationId\": \"AWS-RestartEC2Instance\" } ]",
                    "Type": "SearchableString"
                },
                "/aws/dedup": {
                    "Value": "{\"dedupString\":\"SSMOpsItems-EC2-instance-stopped
\"}",
                    "Type": "SearchableString"
                },
                "/aws/resources": {
                    "Value": "[{\"arn\":\"arn:aws:ec2:us-
east-2:111222333444:instance/i-0beccfbc02EXAMPLE\"}]",
                    "Type": "SearchableString"
                }
            },
            "Category": "Availability",
            "Severity": "3"
        }
    ]
}
```

Para obter mais informações, consulte Trabalhando com OpsItems no Guia do Usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeOpsItemsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
```

```
    * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
    *
    * This method initiates an asynchronous request to describe SSM OpsItems.
    * If the request is successful, it prints the title and status of each
OpsItem.
    * If an exception occurs, it handles the error appropriately.
    */
   public void describeOpsItems(String key) {
       OpsItemFilter filter = OpsItemFilter.builder()
               .key(OpsItemFilterKey.OPS_ITEM_ID)
               .values(key)
               .operator(OpsItemFilterOperator.EQUAL)
               .build();

       DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
               .maxResults(10)
               .opsItemFilters(filter)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().describeOpsItems(itemsRequest)
                   .thenAccept(itemsResponse -> {
                       List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                       for (OpsItemSummary item : items) {
                           System.out.println("The item title is " +
item.title() + " and the status is " + item.status().toString());
                       }
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });
```

```
        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }
```

- Para obter detalhes da API, consulte DescribeOpsItemsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  OpsItemFilterOperator,
  OpsItemFilterKey,
  paginateDescribeOpsItems,
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Describe SSM OpsItems.
 * @param {{ opsItemId: string }}
 */
export const main = async ({ opsItemId }) => {
  const client = new SSMClient({});
  try {
    const describeOpsItemsPaginated = [];
    for await (const page of paginateDescribeOpsItems(
      { client },
      {
```

```
        OpsItemFilters: {
          Key: OpsItemFilterKey.OPSITEM_ID,
          Operator: OpsItemFilterOperator.EQUAL,
          Values: opsItemId,
        },
      },
    )) {
      describeOpsItemsPaginated.push(...page.OpsItemSummaries);
    }
    console.log("Here are the ops items:");
    console.log(describeOpsItemsPaginated);
    return { OpsItemSummaries: describeOpsItemsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte DescribeOpsItemsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
```

```
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def describe(self):
        """
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator":
"Equal"}
                ]
            ):
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
                print(
                    f"The item title is {item['Title']} and the status is
{item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeOpsItemsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create filter for OpsItem ID
        DATA(lt_filters) = VALUE /aws1/cl_ssmopsitemfilter=>tt_opsitemfilters(
          ( NEW /aws1/cl_ssmopsitemfilter(
              iv_key = 'OpsItemId'
              it_values = VALUE /aws1/
cl_ssmopsitemfiltvals_w=>tt_opsitemfiltervalues(
                ( NEW /aws1/cl_ssmopsitemfiltvals_w( iv_value =
 iv_ops_item_id ) )
              )
              iv_operator = 'Equal'
            ) )
        ).

        " Use paginator to get all results
        DATA(lo_paginator) = lo_ssm->get_paginator( ).
        DATA(lo_iterator) = lo_paginator->describeopsitems(
          it_opsitemfilters = lt_filters ).

        rv_found = abap_false.

        WHILE lo_iterator->has_next( ).
          DATA(lo_result) = CAST /aws1/cl_ssmdescropsitemsrsp( lo_iterator-
>get_next( ) ).
          LOOP AT lo_result->get_opsitemsummaries( ) INTO DATA(lo_item).
            DATA(lv_title) = lo_item->get_title( ).
            DATA(lv_status) = lo_item->get_status( ).
```

```
            MESSAGE |The OpsItem title is { lv_title } and the status is
 { lv_status }| TYPE 'I'.
            rv_found = abap_true.
          ENDLOOP.
        ENDWHILE.
      CATCH /aws1/cx_ssminternalservererr.
        MESSAGE 'Internal server error occurred.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeOpsItemsreferência da API AWS SDK for SAP ABAP.

Use **`DescribeParameters`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeParameters`.

CLI

AWS CLI

Exemplo 1: listar todos os parâmetros

O `describe-parameters` exemplo a seguir lista todos os parâmetros na AWS conta atual e na região.

```
aws ssm describe-parameters
```

Saída:

```
{
    "Parameters": [
        {
            "Name": "MySecureStringParameter",
            "Type": "SecureString",
            "KeyId": "alias/aws/ssm",
            "LastModifiedDate": 1582155479.205,
            "LastModifiedUser": "arn:aws:sts::111222333444:assumed-role/Admin/
Richard-Roe-Managed",
            "Description": "This is a SecureString parameter",
            "Version": 2,
            "Tier": "Advanced",
```

```
            "Policies": [
                {
                    "PolicyText": "{\"Type\":\"Expiration\",\"Version\":\"1.0\",
\"Attributes\":{\"Timestamp\":\"2020-07-07T22:30:00Z\"}}",
                    "PolicyType": "Expiration",
                    "PolicyStatus": "Pending"
                },
                {
                    "PolicyText": "{\"Type\":\"ExpirationNotification\",\"Version
\":\"1.0\",\"Attributes\":{\"Before\":\"12\",\"Unit\":\"Hours\"}}",
                    "PolicyType": "ExpirationNotification",
                    "PolicyStatus": "Pending"
                }
            ]
        },
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "LastModifiedDate": 1582154764.222,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is a StringList parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582154711.976,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Alejandro-
Rosalez",
            "Description": "This is a String parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "latestAmi",
            "Type": "String",
            "LastModifiedDate": 1580862415.521,
            "LastModifiedUser": "arn:aws:sts::111222333444:assumed-role/lambda-
ssm-role/Automation-UpdateSSM-Param",
            "Version": 3,
            "Tier": "Standard",
```

```
            "Policies": []
        }
    ]
}
```

Exemplo 2: como listar todos os parâmetros que correspondem a metadados específicos

Esse exemplo de `describe-parameters` lista todos os parâmetros que correspondem a um filtro.

aws ssm describe-parameters --filters “Chave = Tipo, Valores =” StringList

Saída:

```
{
    "Parameters": [
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "LastModifiedDate": 1582154764.222,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is a StringList parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
```

Para obter mais informações, consulte Pesquisando parâmetros do Systems Manager, no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribeParametersem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetParameter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
```

```
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                    .name(paraName)
                    .build();

            GetParameterResponse parameterResponse =
 ssmClient.getParameter(parameterRequest);
            System.out.println("The parameter value is " +
 parameterResponse.parameter().value());

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeParametersa Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todos os parâmetros.

```
Get-SSMParameterList
```

Saída:

```
Description      :
```

```
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- Para obter detalhes da API, consulte DescribeParametersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os parâmetros.

```
Get-SSMParameterList
```

Saída:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- Para obter detalhes da API, consulte DescribeParametersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_parameters(client: &Client) -> Result<(), Error> {
    let resp = client.describe_parameters().send().await?;
```

```
    for param in resp.parameters() {
        println!("  {}", param.name().unwrap_or_default());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeParametersreferência da API AWS SDK for Rust.

Usar **`DescribePatchBaselines`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribePatchBaselines`.

CLI

AWS CLI

Exemplo 1: Para listar todas as linhas de base do patch

O exemplo de `describe-patch-baselines` a seguir recupera os detalhes de todas as listas de referências de patches de sua conta na região atual.

```
aws ssm describe-patch-baselines
```

Saída:

```
{
    "BaselineIdentities": [
        {
            "BaselineName": "AWS-SuseDefaultPatchBaseline",
            "DefaultBaseline": true,
            "BaselineDescription": "Default Patch Baseline for Suse Provided by
 AWS.",
            "BaselineId": "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-0123fdb36e334a3b2",
            "OperatingSystem": "SUSE"
        },
        {
            "BaselineName": "AWS-DefaultPatchBaseline",
```

```
            "DefaultBaseline": false,
            "BaselineDescription": "Default Patch Baseline Provided by AWS.",
            "BaselineId": "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-020d361a05defe4ed",
            "OperatingSystem": "WINDOWS"
        },
        ...
        {
            "BaselineName": "MyWindowsPatchBaseline",
            "DefaultBaseline": true,
            "BaselineDescription": "My patch baseline for EC2 instances for
 Windows Server",
            "BaselineId": "pb-0ad00e0dd7EXAMPLE",
            "OperatingSystem": "WINDOWS"
        }
    ]
}
```

Exemplo 2: Para listar todas as linhas de base do patch fornecidas pelo AWS

O `describe-patch-baselines` exemplo a seguir lista todas as linhas de base de patch fornecidas pelo. AWS

```
aws ssm describe-patch-baselines \
    --filters "Key=OWNER,Values=[AWS]"
```

Exemplo 3: listar todas as listas de referência de patches pertencentes a você

O exemplo de `describe-patch-baselines` a seguir lista todas as listas de referências de patches criadas em sua conta na região atual.

```
aws ssm describe-patch-baselines \
    --filters "Key=OWNER,Values=[Self]"
```

Para obter mais informações, consulte Sobre listas de referência de patches predefinidas e personalizadas no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribePatchBaselinesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra todas as listas de referência de patches.

```
Get-SSMPatchBaseline
```

Saída:

```
BaselineDescription                                                BaselineId
                                                        BaselineName
-------------------                                                ----------
                                                        ------------
Default Patch Baseline Provided by AWS.                            arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems
 pb-045f10b4f382baeda
 Production-B...
Baseline containing all updates approved for production systems
 pb-0a2f1059b670ebd31
 Production-B...
```

Exemplo 2: Este exemplo lista todas as linhas de base de patch fornecidas pelo. AWS A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Saída:

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemplo 3: esse exemplo mostra todas as listas de referência de patches pertencentes a você. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Saída:

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar cada tag.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Saída:

```
BaselineDescription                     BaselineId
                            BaselineName             DefaultBaselin


                                                     e
-------------------                     ----------
                            ------------             --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline
 True
```

- Para obter detalhes da API, consulte DescribePatchBaselinesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra todas as listas de referência de patches.

```
Get-SSMPatchBaseline
```

Saída:

```
BaselineDescription                                                BaselineId
                                                        BaselineName
-------------------                                                ----------
                                                        ------------
Default Patch Baseline Provided by AWS.                            arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems
 pb-045f10b4f382baeda
 Production-B...
```

```
Baseline containing all updates approved for production systems
 pb-0a2f1059b670ebd31
 Production-B...
```

Exemplo 2: Este exemplo lista todas as linhas de base de patch fornecidas pelo. AWS A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Saída:

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemplo 3: esse exemplo mostra todas as listas de referência de patches pertencentes a você. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Saída:

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar cada tag.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Saída:

```
BaselineDescription                     BaselineId
                            BaselineName             DefaultBaselin


                                                     e
-------------------                     ----------
                            ------------             --------------
```

```
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline
 True
```

- Para obter detalhes da API, consulte DescribePatchBaselinesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribePatchGroupState`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribePatchGroupState`.

CLI

AWS CLI

Para obter o estado de um grupo de patches

O exemplo de `describe-patch-group-state` a seguir recupera o resumo de conformidade de patches de alto nível para um grupo de patches.

```
aws ssm describe-patch-group-state \
    --patch-group "Production"
```

Saída:

```
{
    "Instances": 21,
    "InstancesWithCriticalNonCompliantPatches": 1,
    "InstancesWithFailedPatches": 2,
    "InstancesWithInstalledOtherPatches": 3,
    "InstancesWithInstalledPatches": 21,
    "InstancesWithInstalledPendingRebootPatches": 2,
    "InstancesWithInstalledRejectedPatches": 1,
    "InstancesWithMissingPatches": 3,
    "InstancesWithNotApplicablePatches": 4,
    "InstancesWithOtherNonCompliantPatches": 1,
    "InstancesWithSecurityNonCompliantPatches": 1,
    "InstancesWithUnreportedNotApplicablePatches": 2
}
```

Para obter mais informações, consulte Sobre grupos de patches < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-patchgroups.html

>__ e Entendendo os valores do estado de conformidade de patches no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribePatchGroupStateem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém o resumo da conformidade de patches de alto nível para um grupo de patches.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Saída:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- Para obter detalhes da API, consulte DescribePatchGroupStateem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém o resumo da conformidade de patches de alto nível para um grupo de patches.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Saída:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
```

```
InstancesWithNotApplicablePatches  : 0
```

- Para obter detalhes da API, consulte DescribePatchGroupStateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribePatchGroups`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribePatchGroups`.

CLI

AWS CLI

Para exibir registros de grupos de patches

O exemplo de `describe-patch-groups` a seguir lista os registros de grupos de patches.

```
aws ssm describe-patch-groups
```

Saída:

```
{
    "Mappings": [
        {
            "PatchGroup": "Production",
            "BaselineIdentity": {
                "BaselineId": "pb-0123456789abcdef0",
                "BaselineName": "ProdPatching",
                "OperatingSystem": "WINDOWS",
                "BaselineDescription": "Patches for Production",
                "DefaultBaseline": false
            }
        },
        {
            "PatchGroup": "Development",
            "BaselineIdentity": {
                "BaselineId": "pb-0713accee01234567",
                "BaselineName": "DevPatching",
                "OperatingSystem": "WINDOWS",
                "BaselineDescription": "Patches for Development",
                "DefaultBaseline": true
            }
```

```
        },
        ...
    ]
}
```

Para obter mais informações, consulte Criar um grupo de patches < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html >__ e Adicionar um grupo de patches a uma linha de base de patches no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte DescribePatchGroupsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista os registros do grupo de patches.

```
Get-SSMPatchGroup
```

Saída:

```
BaselineIdentity                                          PatchGroup
----------------                                          ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Para obter detalhes da API, consulte DescribePatchGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os registros do grupo de patches.

```
Get-SSMPatchGroup
```

Saída:

```
BaselineIdentity                                          PatchGroup
----------------                                          ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Para obter detalhes da API, consulte DescribePatchGroupsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetAutomationExecution`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetAutomationExecution`.

CLI

AWS CLI

Para exibir detalhes sobre uma execução do Automation

O exemplo de `get-automation-execution` a seguir exibe informações detalhadas sobre uma execução do Automation.

```
aws ssm get-automation-execution \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Saída:

```
{
    "AutomationExecution": {
        "AutomationExecutionId": "73c8eef8-f4ee-4a05-820c-e354fEXAMPLE",
        "DocumentName": "AWS-StartEC2Instance",
        "DocumentVersion": "1",
        "ExecutionStartTime": 1583737233.748,
        "ExecutionEndTime": 1583737234.719,
        "AutomationExecutionStatus": "Success",
        "StepExecutions": [
            {
                "StepName": "startInstances",
                "Action": "aws:changeInstanceState",
                "ExecutionStartTime": 1583737234.134,
                "ExecutionEndTime": 1583737234.672,
                "StepStatus": "Success",
                "Inputs": {
                    "DesiredState": "\"running\"",
                    "InstanceIds": "[\"i-0cb99161f6EXAMPLE\"]"
                },
                "Outputs": {
                    "InstanceStates": [
```

```
                        "running"
                    ]
                },
                "StepExecutionId": "95e70479-cf20-4d80-8018-7e4e2EXAMPLE",
                "OverriddenParameters": {}
            }
        ],
        "StepExecutionsTruncated": false,
        "Parameters": {
            "AutomationAssumeRole": [
                ""
            ],
            "InstanceId": [
                "i-0cb99161f6EXAMPLE"
            ]
        },
        "Outputs": {},
        "Mode": "Auto",
        "ExecutedBy": "arn:aws:sts::29884EXAMPLE:assumed-role/mw_service_role/
OrchestrationService",
        "Targets": [],
        "ResolvedTargets": {
            "ParameterValues": [],
            "Truncated": false
        }
    }
}
```

Para obter mais informações, consulte Passo a passo: corrigir uma AMI do Linux (AWS CLI) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetAutomationExecutionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe os detalhes de uma execução do Automation.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Saída:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU

 gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-q0CK0ezS3xfh_y0ilaUGOAZG-
xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                            6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions            : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Exemplo 2: esse exemplo lista os detalhes da etapa para o ID de execução do Automation fornecido

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Saída:

```
StepName                  Action                  StepStatus ValidNextSteps
--------                  ------                  ---------- --------------
LaunchInstance            aws:runInstances        Success
 {OSCompatibilityCheck}
OSCompatibilityCheck      aws:runCommand          Success    {RunPreUpdateScript}
RunPreUpdateScript        aws:runCommand          Success    {UpdateEC2Config}
UpdateEC2Config           aws:runCommand          Cancelled  {}
UpdateSSMAgent            aws:runCommand          Pending    {}
UpdateAWSPVDriver         aws:runCommand          Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand          Pending    {}
UpdateAWSNVMe             aws:runCommand          Pending    {}
InstallWindowsUpdates     aws:runCommand          Pending    {}
RunPostUpdateScript       aws:runCommand          Pending    {}
RunSysprepGeneralize      aws:runCommand          Pending    {}
StopInstance              aws:changeInstanceState Pending    {}
CreateImage               aws:createImage         Pending    {}
TerminateInstance         aws:changeInstanceState Pending    {}
```

- Para obter detalhes da API, consulte GetAutomationExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe os detalhes de uma execução do Automation.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Saída:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
```

```
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU

 gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-q0CK0ezS3xfh_y0ilaUGOAZG-
xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                            6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions            : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Exemplo 2: esse exemplo lista os detalhes da etapa para o ID de execução do Automation fornecido

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Saída:

```
StepName                  Action                  StepStatus ValidNextSteps
--------                  ------                  ---------- --------------
LaunchInstance            aws:runInstances        Success
 {OSCompatibilityCheck}
OSCompatibilityCheck      aws:runCommand          Success    {RunPreUpdateScript}
RunPreUpdateScript        aws:runCommand          Success    {UpdateEC2Config}
UpdateEC2Config           aws:runCommand          Cancelled  {}
UpdateSSMAgent            aws:runCommand          Pending    {}
UpdateAWSPVDriver         aws:runCommand          Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand          Pending    {}
UpdateAWSNVMe             aws:runCommand          Pending    {}
```

```
InstallWindowsUpdates     aws:runCommand          Pending    {}
RunPostUpdateScript       aws:runCommand          Pending    {}
RunSysprepGeneralize      aws:runCommand          Pending    {}
StopInstance              aws:changeInstanceState Pending    {}
CreateImage               aws:createImage         Pending    {}
TerminateInstance         aws:changeInstanceState Pending    {}
```

- Para obter detalhes da API, consulte GetAutomationExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetCommandInvocation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetCommandInvocation`.

CLI

AWS CLI

Para exibir os detalhes de uma invocação de comando

O exemplo de `get-command-invocation` a seguir lista todas as invocações do comando especificado na instância especificada.

```
aws ssm get-command-invocation \
    --command-id "ef7fdfd8-9b57-4151-a15c-db9a12345678" \
    --instance-id "i-1234567890abcdef0"
```

Saída:

```
{
    "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
    "InstanceId": "i-1234567890abcdef0",
    "Comment": "b48291dd-ba76-43e0-b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-
d6ce8EXAMPLE",
    "DocumentName": "AWS-UpdateSSMAgent",
    "DocumentVersion": "",
    "PluginName": "aws:updateSsmAgent",
    "ResponseCode": 0,
    "ExecutionStartDateTime": "2020-02-19T18:18:03.419Z",
    "ExecutionElapsedTime": "PT0.091S",
    "ExecutionEndDateTime": "2020-02-19T18:18:03.419Z",
    "Status": "Success",
```

```
    "StatusDetails": "Success",
    "StandardOutputContent": "Updating amazon-ssm-agent from 2.3.842.0 to latest
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-
east-2/ssm-agent-manifest.json\namazon-ssm-agent 2.3.842.0 has already been
 installed, update skipped\n",
    "StandardOutputUrl": "",
    "StandardErrorContent": "",
    "StandardErrorUrl": "",
    "CloudWatchOutputConfig": {
        "CloudWatchLogGroupName": "",
        "CloudWatchOutputEnabled": false
    }
}
```

Para obter mais informações, consulte Entender os status dos comandos no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetCommandInvocationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe os detalhes de um comando executado em uma instância.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Saída:

```
CommandId              : b8eac879-0541-439d-94ec-47a80d554f44
Comment                : IP config
DocumentName           : AWS-RunShellScript
ExecutionElapsedTime   : PT0.004S
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
StandardErrorUrl       :
```

```
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Para obter detalhes da API, consulte GetCommandInvocationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe os detalhes de um comando executado em uma instância.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Saída:

```
CommandId              : b8eac879-0541-439d-94ec-47a80d554f44
Comment                : IP config
DocumentName           : AWS-RunShellScript
ExecutionElapsedTime   : PT0.004S
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
StandardErrorUrl       :
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Para obter detalhes da API, consulte GetCommandInvocationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetConnectionStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetConnectionStatus`.

CLI

AWS CLI

Para exibir o status da conexão de uma instância gerenciada

Este exemplo de `get-connection-status` retorna o status da conexão da instância gerenciada especificada.

```
aws ssm get-connection-status \
    --target i-1234567890abcdef0
```

Saída:

```
{
    "Target": "i-1234567890abcdef0",
    "Status": "connected"
}
```

- Para obter detalhes da API, consulte GetConnectionStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo recupera o status de conexão do Gerenciador de Sessões de uma instância para determinar se ela está conectada e pronta para receber conexões do Gerenciador de Sessões.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Saída:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- Para obter detalhes da API, consulte GetConnectionStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera o status de conexão do Gerenciador de Sessões de uma instância para determinar se ela está conectada e pronta para receber conexões do Gerenciador de Sessões.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Saída:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- Para obter detalhes da API, consulte GetConnectionStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetDefaultPatchBaseline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetDefaultPatchBaseline`.

CLI

AWS CLI

Exemplo 1: para exibir a lista de referência de patches padrão do Windows

O exemplo de `get-default-patch-baseline` a seguir recupera detalhes da lista de referência de patches padrão para o Windows Server.

```
aws ssm get-default-patch-baseline
```

Saída:

```
{
  "BaselineId": "pb-0713accee01612345",
  "OperatingSystem": "WINDOWS"
}
```

Exemplo 2: para exibir a lista de referência de patches padrão do Amazon Linux

O exemplo de `get-default-patch-baseline` a seguir recupera detalhes da lista de referência de patches padrão para o Amazon Linux.

```
aws ssm get-default-patch-baseline \
    --operating-system AMAZON_LINUX
```

Saída:

```
{
    "BaselineId": "pb-047c6eb9c8fc12345",
    "OperatingSystem": "AMAZON_LINUX"
}
```

Para obter mais informações, consulte Sobre linhas de base de patches predefinidas e personalizadas < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-baselines.html >__ e Definir uma linha de base de patch existente como padrão no Guia do usuário do Systems Manager AWS .

- Para obter detalhes da API, consulte GetDefaultPatchBaselineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe a lista de referência de patches padrão.

```
Get-SSMDefaultPatchBaseline
```

Saída:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Para obter detalhes da API, consulte GetDefaultPatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe a lista de referência de patches padrão.

```
Get-SSMDefaultPatchBaseline
```

Saída:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Para obter detalhes da API, consulte GetDefaultPatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetDeployablePatchSnapshotForInstance`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetDeployablePatchSnapshotForInstance`.

CLI

AWS CLI

Para recuperar o instantâneo atual da lista de referência de patches usado por uma instância

O exemplo de `get-deployable-patch-snapshot-for-instance` a seguir recupera detalhes do instantâneo atual da lista de referência de patches especificada usada por uma instância. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que ele use as credenciais da instância, execute `aws configure` e especifique somente a região da sua instância. Deixe os campos `Access Key` e `Secret Key` vazios.

Dica: use `uuidgen` para gerar um `snapshot-id`.

```
aws ssm get-deployable-patch-snapshot-for-instance \
    --instance-id "i-1234567890abcdef0" \
    --snapshot-id "521c3536-930c-4aa9-950e-01234567abcd"
```

Saída:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "SnapshotId": "521c3536-930c-4aa9-950e-01234567abcd",
    "Product": "AmazonLinux2018.03",
    "SnapshotDownloadUrl": "https://patch-baseline-snapshot-us-
east-1.s3.amazonaws.com/
```

```
ed85194ef27214f5984f28b4d664d14f7313568fea7d4b6ac6c10ad1f729d7e7-773304212436/
AMAZON_LINUX-521c3536-930c-4aa9-950e-01234567abcd?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20190215T164031Z&X-Amz-
SignedHeaders=host&X-Amz-Expires=86400&X-Amz-Credential=AKIAJ5C56P35AEBRX2QQ
%2F20190215%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Signature=efaaaf6e3878e77f48a6697e015efdbda9c426b09c5822055075c062f6ad2149"
}
```

Para obter mais informações, consulte Nome do parâmetro: ID do instantâneo no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetDeployablePatchSnapshotForInstanceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe o instantâneo atual da lista de referência de patches usada por uma instância. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que use as credenciais da instância, o exemplo passa um objeto **`Amazon.Runtime.InstanceProfileAWSCredentials`** para o parâmetro Credentials.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Saída:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Exemplo 2: Este exemplo mostra como obter o completo SnapshotDownloadUrl. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que ele use as credenciais da instância, o exemplo configura a PowerShell sessão para usar um **`Amazon.Runtime.InstanceProfileAWSCredentials`** objeto.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
```

```
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Saída:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Para obter detalhes da API, consulte GetDeployablePatchSnapshotForInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe o instantâneo atual da lista de referência de patches usada por uma instância. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que use as credenciais da instância, o exemplo passa um objeto **`Amazon.Runtime.InstanceProfileAWSCredentials`** para o parâmetro Credentials.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Saída:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Exemplo 2: Este exemplo mostra como obter o completo SnapshotDownloadUrl. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que ele use as credenciais da instância, o exemplo configura a PowerShell sessão para usar um **`Amazon.Runtime.InstanceProfileAWSCredentials`** objeto.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Saída:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Para obter detalhes da API, consulte GetDeployablePatchSnapshotForInstanceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetDocument`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetDocument`.

CLI

AWS CLI

Para obter conteúdo do documento

O exemplo de `get-document` a seguir exibe o conteúdo de um documento do Systems Manager.

```
aws ssm get-document \
    --name "AWS-RunShellScript"
```

Saída:

```
{
    "Name": "AWS-RunShellScript",
    "DocumentVersion": "1",
    "Status": "Active",
    "Content": "{\n    \"schemaVersion\":\"1.2\",\n    \"description\":\"Run
 a shell script or specify the commands to run.\",\n    \"parameters\":{\n
        \"commands\":{\n            \"type\":\"StringList\",\n
 \"description\":\"(Required) Specify a shell script or a command to run.\",
\n            \"minItems\":1,\n            \"displayType\":\"textarea\"\n
      },\n        \"workingDirectory\":{\n            \"type\":\"String\",\n
            \"default\":\"\",\n            \"description\":\"(Optional) The
 path to the working directory on your instance.\",\n            \"maxChars
\":4096\n        },\n        \"executionTimeout\":{\n            \"type\":
\"String\",\n            \"default\":\"3600\",\n            \"description
\":\"(Optional) The time in seconds for a command to complete before it is
 considered to have failed. Default is 3600 (1 hour). Maximum is 172800 (48
 hours).\",\n            \"allowedPattern\":\"([1-9][0-9]{0,4})|(1[0-6][0-9]
```

```
{4})|(17[0-1][0-9]{3})|(172[0-7][0-9]{2})|(172800)\"\n        }\n    },\n
 \"runtimeConfig\":{\n        \"aws:runShellScript\":{\n            \"properties
\":[\n                {\n                    \"id\":\"0.aws:runShellScript
\",\n                    \"runCommand\":\"{{ commands }}\",\n
      \"workingDirectory\":\"{{ workingDirectory }}\",\n
 \"timeoutSeconds\":\"{{ executionTimeout }}\"\n                }\n
  ]\n        }\n    }\n}\n",
    "DocumentType": "Command",
    "DocumentFormat": "JSON"
}
```

Para obter mais informações, consulte Documentos do AWS Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetDocumentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo retorna o conteúdo de um documento.

```
Get-SSMDocument -Name "RunShellScript"
```

Saída:

```
Content
-------
{...
```

Exemplo 2: esse exemplo exibe o conteúdo completo de um documento.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
```

```
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      }
   ]
}
```

- Para obter detalhes da API, consulte GetDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o conteúdo de um documento.

```
Get-SSMDocument -Name "RunShellScript"
```

Saída:

```
Content
-------
{...
```

Exemplo 2: esse exemplo exibe o conteúdo completo de um documento.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
   "parameters":{
      "commands":{
```

```
            "type":"StringList",
            "description":"(Required) Specify a shell script or a command to run.",
            "minItems":1,
            "displayType":"textarea"
         }
      },
      "mainSteps":[
         {
            "action":"aws:runShellScript",
            "name":"runShellScript",
            "inputs":{
               "commands":"{{ commands }}"
            }
         },
         {
            "action":"aws:runPowerShellScript",
            "name":"runPowerShellScript",
            "inputs":{
               "commands":"{{ commands }}"
            }
         }
      ]
   }
```

- Para obter detalhes da API, consulte GetDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetInventory`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetInventory`.

CLI

AWS CLI

Para visualizar o inventário

Este exemplo obtém os metadados personalizados do seu inventário.

Comando:

```
aws ssm get-inventory
```

Saída:

```
{
  "Entities": [
      {
          "Data": {
              "AWS:InstanceInformation": {
                  "Content": [
                      {
                          "ComputerName": "ip-172-31-44-222.us-
west-2.compute.internal",
                          "InstanceId": "i-0cb2b964d3e14fd9f",
                          "IpAddress": "172.31.44.222",
                          "AgentType": "amazon-ssm-agent",
                          "ResourceType": "EC2Instance",
                          "AgentVersion": "2.0.672.0",
                          "PlatformVersion": "2016.09",
                          "PlatformName": "Amazon Linux AMI",
                          "PlatformType": "Linux"
                      }
                  ],
                  "TypeName": "AWS:InstanceInformation",
                  "SchemaVersion": "1.0",
                  "CaptureTime": "2017-02-20T18:03:58Z"
              }
          },
          "Id": "i-0cb2b964d3e14fd9f"
      }
  ]
}
```

- Para obter detalhes da API, consulte GetInventoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém os metadados personalizados do seu inventário.

```
Get-SSMInventory
```

Saída:

```
Data
     Id
----
     --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Para obter detalhes da API, consulte GetInventoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os metadados personalizados do seu inventário.

```
Get-SSMInventory
```

Saída:

```
Data
     Id
----
     --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Para obter detalhes da API, consulte GetInventoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetInventorySchema`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetInventorySchema`.

CLI

AWS CLI

Para visualizar seu esquema de inventário

Este exemplo retorna uma lista de nomes de tipos de inventário para a conta.

Comando:

```
aws ssm get-inventory-schema
```

Saída:

```
{
  "Schemas": [
      {
          "TypeName": "AWS:AWSComponent",
          "Version": "1.0",
          "Attributes": [
              {
                  "Name": "Name",
                  "DataType": "STRING"
              },
              {
                  "Name": "ApplicationType",
                  "DataType": "STRING"
              },
              {
                  "Name": "Publisher",
                  "DataType": "STRING"
              },
              {
                  "Name": "Version",
                  "DataType": "STRING"
              },
              {
                  "Name": "InstalledTime",
                  "DataType": "STRING"
              },
              {
                  "Name": "Architecture",
                  "DataType": "STRING"
              },
              {
                  "Name": "URL",
                  "DataType": "STRING"
              }
          ]
      },
      ...
  ],
  "NextToken": "--token string truncated--"
```

```
}
```

Para visualizar o esquema de inventário de um tipo de inventário específico

Este exemplo retorna o esquema de inventário para o tipo de inventário AWS:AWS Componente.

Comando:

```
aws ssm get-inventory-schema --type-name "AWS:AWSComponent"
```

- Para obter detalhes da API, consulte GetInventorySchemaem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo retorna uma lista de nomes de tipos de inventário para a conta.

```
Get-SSMInventorySchema
```

- Para obter detalhes da API, consulte GetInventorySchemaem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna uma lista de nomes de tipos de inventário para a conta.

```
Get-SSMInventorySchema
```

- Para obter detalhes da API, consulte GetInventorySchemaem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetMaintenanceWindow`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetMaintenanceWindow`.

CLI

AWS CLI

Para obter informações sobre uma janela de manutenção

O exemplo de `get-maintenance-window` a seguir recupera detalhes sobre a janela de manutenção especificada.

```
aws ssm get-maintenance-window \
    --window-id "mw-03eb9db428EXAMPLE"
```

Saída:

```
{
    "AllowUnassociatedTargets": true,
    "CreatedDate": 1515006912.957,
    "Cutoff": 1,
    "Duration": 6,
    "Enabled": true,
    "ModifiedDate": 2020-01-01T10:04:04.099Z,
    "Name": "My-Maintenance-Window",
    "Schedule": "rate(3 days)",
    "WindowId": "mw-03eb9db428EXAMPLE",
    "NextExecutionTime": "2020-02-25T00:08:15.099Z"
}
```

Para obter mais informações, consulte Exibir informações sobre janelas de manutenção (AWS CLI) no Guia do Usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetMaintenanceWindowem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém detalhes sobre uma janela de manutenção.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Saída:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para obter detalhes da API, consulte GetMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém detalhes sobre uma janela de manutenção.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Saída:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para obter detalhes da API, consulte GetMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetMaintenanceWindowExecution`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetMaintenanceWindowExecution`.

CLI

AWS CLI

Para obter informações sobre a execução de uma tarefa da janela de manutenção

O exemplo de `get-maintenance-window-execution` a seguir lista informações sobre uma tarefa que é executada como parte da execução da janela de manutenção especificada.

```
aws ssm get-maintenance-window-execution \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE"
```

Saída:

```
{
    "Status": "SUCCESS",
    "TaskIds": [
        "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
    ],
    "StartTime": 1487692834.595,
    "EndTime": 1487692835.051,
    "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
}
```

Para obter mais informações, consulte Exibir informações sobre tarefas e execuções de tarefas (AWS CLI) no Guia do usuário AWS do Systems Manager.

- Para obter detalhes da API, consulte GetMaintenanceWindowExecutionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista informações sobre uma tarefa que foi executada como parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte GetMaintenanceWindowExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista informações sobre uma tarefa que foi executada como parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte GetMaintenanceWindowExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetMaintenanceWindowExecutionTask`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetMaintenanceWindowExecutionTask`.

CLI

AWS CLI

Para obter informações sobre a execução de uma tarefa da janela de manutenção

O exemplo de `get-maintenance-window-execution-task` a seguir lista informações sobre uma tarefa que faz parte da execução da janela de manutenção especificada.

```
aws ssm get-maintenance-window-execution-task \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE" \
    --task-id "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
```

Saída:

```
{
    "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
    "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE",
    "TaskArn": "AWS-RunPatchBaseline",
    "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
    "Type": "RUN_COMMAND",
    "TaskParameters": [
        {
            "BaselineOverride": {
                "Values": [
                    ""
                ]
            },
            "InstallOverrideList": {
                "Values": [
                    ""
                ]
            },
            "Operation": {
                "Values": [
                    "Scan"
                ]
            },
            "RebootOption": {
                "Values": [
                    "RebootIfNeeded"
                ]
            },
            "SnapshotId": {
                "Values": [
                    "{{ aws:ORCHESTRATION_ID }}"
                ]
            },
            "aws:InstanceId": {
                "Values": [
                    "i-02573cafcfEXAMPLE",
```

```
                    "i-0471e04240EXAMPLE",
                    "i-07782c72faEXAMPLE"
                ]
            }
        }
    ],
    "Priority": 1,
    "MaxConcurrency": "1",
    "MaxErrors": "3",
    "Status": "SUCCESS",
    "StartTime": "2021-08-04T11:45:35.088000-07:00",
    "EndTime": "2021-08-04T11:53:09.079000-07:00"
}
```

Para obter mais informações, consulte Exibir informações sobre tarefas e execuções de tarefas (AWS CLI) no Guia do usuário AWS do Systems Manager.

- Para obter detalhes da API, consulte GetMaintenanceWindowExecutionTaskem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista informações sobre uma tarefa que fazia parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-
d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
```

```
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsM
                    meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte GetMaintenanceWindowExecutionTaskem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista informações sobre uma tarefa que fazia parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-
d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsM
                    meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes da API, consulte GetMaintenanceWindowExecutionTaskem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`GetParameter`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetParameter`.

CLI

AWS CLI

Exemplo 1: exibir o valor de um parâmetro

O exemplo de `get-parameter` a seguir lista o valor do parâmetro único especificado.

```
aws ssm get-parameter \
    --name "MyStringParameter"
```

Saída:

```
{
    "Parameter": {
        "Name": "MyStringParameter",
        "Type": "String",
        "Value": "Veni",
        "Version": 1,
        "LastModifiedDate": 1530018761.888,
        "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/MyStringParameter"
        "DataType": "text"
    }
}
```

Para obter mais informações, consulte Trabalhar com o Parameter Store no Guia do usuário do AWS Systems Manager.

Exemplo 2: Para descriptografar o valor de um parâmetro SecureString

O exemplo de `get-parameter` a seguir descriptografa o valor do parâmetro `SecureString` especificado.

```
aws ssm get-parameter \
    --name "MySecureStringParameter" \
    --with-decryption
```

Saída:

```
{
    "Parameter": {
```

```
        "Name": "MySecureStringParameter",
        "Type": "SecureString",
        "Value": "16679b88-310b-4895-a943-e0764EXAMPLE",
        "Version": 2,
        "LastModifiedDate": 1582155479.205,
        "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MySecureStringParameter"
        "DataType": "text"
    }
}
```

Para obter mais informações, consulte Trabalhar com o Parameter Store no Guia do usuário do AWS Systems Manager.

Exemplo 3: exibir o valor de um parâmetro usando rótulos

O exemplo de `get-parameter` a seguir lista o valores do parâmetros especificado com um rótulo especificado.

```
aws ssm get-parameter \
    --name "MyParameter:label"
```

Saída:

```
{
    "Parameter": {
        "Name": "MyParameter",
        "Type": "String",
        "Value": "parameter version 2",
        "Version": 2,
        "Selector": ":label",
        "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
        "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
        "DataType": "text"
    }
}
```

Para obter mais informações, consulte Trabalhar com rótulos de parâmetros no Guia do usuário do AWS Systems Manager.

Exemplo 4: exibir o valor de um parâmetro usando versões

O exemplo de `get-parameter` a seguir lista o valor da versão do parâmetro único especificado.

```
aws ssm get-parameter \
    --name "MyParameter:2"
```

Saída:

```
{
    "Parameter": {
        "Name": "MyParameter",
        "Type": "String",
        "Value": "parameter version 2",
        "Version": 2,
        "Selector": ":2",
        "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
        "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
        "DataType": "text"
    }
}
```

Para obter mais informações, consulte Trabalhar com rótulos de parâmetros no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetParameterem Referência de AWS CLI Comandos.

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error>
 {
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
```

```
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
        .await;
        // Fail on the first error
        let params = maybe_params
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
    }
```

- Para obter detalhes da API, consulte a GetParameterreferência da API AWS SDK for Rust.

Usar **`GetParameterHistory`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetParameterHistory`.

CLI

AWS CLI

Para obter o histórico de valores de um parâmetro

O exemplo de `get-parameter-history` a seguir lista o histórico de alterações do parâmetro especificado, incluindo seu valor.

```
aws ssm get-parameter-history \
    --name "MyStringParameter"
```

Saída:

```
{
    "Parameters": [
        {
            "Name": "MyStringParameter",
            "Type": "String",
```

```
            "LastModifiedDate": 1582154711.976,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the first version of my String parameter",
            "Value": "Veni",
            "Version": 1,
            "Labels": [],
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582156093.471,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the second version of my String parameter",
            "Value": "Vidi",
            "Version": 2,
            "Labels": [],
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582156117.545,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the third version of my String parameter",
            "Value": "Vici",
            "Version": 3,
            "Labels": [],
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
```

Para obter mais informações, consulte Trabalhar com versões de parâmetros no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetParameterHistoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista o histórico de valores de um parâmetro.

```
Get-SSMParameterHistory -Name "Welcome"
```

Saída:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- Para obter detalhes da API, consulte GetParameterHistoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista o histórico de valores de um parâmetro.

```
Get-SSMParameterHistory -Name "Welcome"
```

Saída:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- Para obter detalhes da API, consulte GetParameterHistoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetParameters`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetParameters`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Como criar um serviço de contêiner para instâncias de máquinas virtuais

CLI

AWS CLI

Exemplo 1: listar os valores de um parâmetro

O exemplo de `get-parameters` a seguir lista os valores dos três parâmetros especificados.

```
aws ssm get-parameters \
    --names "MyStringParameter" "MyStringListParameter" "MyInvalidParameterName"
```

Saída:

```
{
    "Parameters": [
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "Value": "alpha,beta,gamma",
            "Version": 1,
            "LastModifiedDate": 1582154764.222,
            "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MyStringListParameter"
            "DataType": "text"
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "Value": "Vici",
            "Version": 3,
            "LastModifiedDate": 1582156117.545,
            "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MyStringParameter"
```

```
            "DataType": "text"
        }
    ],
    "InvalidParameters": [
        "MyInvalidParameterName"
    ]
}
```

Para obter mais informações, consulte Trabalhar com o Parameter Store no Guia do usuário do AWS Systems Manager.

Exemplo 2: listar nomes e valores de vários parâmetros usando a opção "--query"

O exemplo de `get-parameters` a seguir lista os nomes e valores dos parâmetros especificados.

```
aws ssm get-parameters \
    --names MyStringParameter MyStringListParameter \
    --query "Parameters[*].{Name:Name,Value:Value}"
```

Saída:

```
[
    {
        "Name": "MyStringListParameter",
        "Value": "alpha,beta,gamma"
    },
    {
        "Name": "MyStringParameter",
        "Value": "Vidi"
    }
]
```

Para obter mais informações, consulte Trabalhar com o Parameter Store no Guia do usuário do AWS Systems Manager.

Exemplo 3: exibir o valor de um parâmetro usando rótulos

O exemplo de `get-parameter` a seguir lista o valores do parâmetros especificado com um rótulo especificado.

```
aws ssm get-parameter \
```

```
    --name "MyParameter:label"
```

Saída:

```
{
    "Parameters": [
        {
            "Name": "MyLabelParameter",
            "Type": "String",
            "Value": "parameter by label",
            "Version": 1,
            "Selector": ":label",
            "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
            "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
            "DataType": "text"
        },
        {
            "Name": "MyVersionParameter",
            "Type": "String",
            "Value": "parameter by version",
            "Version": 2,
            "Selector": ":2",
            "LastModifiedDate": "2021-03-24T16:20:28.236000-07:00",
            "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/unlabel-param",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```

Para obter mais informações, consulte Trabalhar com rótulos de parâmetros no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetParametersem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista os valores de um parâmetro.

```
Get-SSMParameterValue -Name "Welcome"
```

Saída:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Exemplo 2: esse exemplo retorna os detalhes do valor.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Saída:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- Para obter detalhes da API, consulte GetParametersem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os valores de um parâmetro.

```
Get-SSMParameterValue -Name "Welcome"
```

Saída:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Exemplo 2: esse exemplo retorna os detalhes do valor.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Saída:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- Para obter detalhes da API, consulte GetParametersem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetPatchBaseline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetPatchBaseline`.

CLI

AWS CLI

Para exibir uma lista de referência de patches

O exemplo de `get-patch-baseline` a seguir recupera os detalhes da lista de referência de patches especificada.

```
aws ssm get-patch-baseline \
    --baseline-id "pb-0123456789abcdef0"
```

Saída:

```
{
    "BaselineId": "pb-0123456789abcdef0",
    "Name": "WindowsPatching",
    "OperatingSystem": "WINDOWS",
    "GlobalFilters": {
        "PatchFilters": []
    },
    "ApprovalRules": {
        "PatchRules": [
            {
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "WindowsServer2016"
                            ]
                        }
                    ]
                },
                "ComplianceLevel": "CRITICAL",
```

```
                "ApproveAfterDays": 0,
                "EnableNonSecurity": false
            }
        ]
    },
    "ApprovedPatches": [],
    "ApprovedPatchesComplianceLevel": "UNSPECIFIED",
    "ApprovedPatchesEnableNonSecurity": false,
    "RejectedPatches": [],
    "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
    "PatchGroups": [
        "QA",
        "DEV"
    ],
    "CreatedDate": 1550244180.465,
    "ModifiedDate": 1550244180.465,
    "Description": "Patches for Windows Servers",
    "Sources": []
}
```

Para obter mais informações, consulte Sobre listas de referência de patches no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetPatchBaselineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe os detalhes de uma lista de referência de patches.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Saída:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
```

```
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- Para obter detalhes da API, consulte GetPatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe os detalhes de uma lista de referência de patches.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Saída:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- Para obter detalhes da API, consulte GetPatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`GetPatchBaselineForPatchGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetPatchBaselineForPatchGroup`.

CLI

AWS CLI

Para exibir uma lista de referência de patches de um grupo de patches

O exemplo de `get-patch-baseline-for-patch-group` a seguir recupera detalhes sobre a lista de referência de patches para a instância especificada.

```
aws ssm get-patch-baseline-for-patch-group \
    --patch-group "DEV"
```

Saída:

```
{
    "PatchGroup": "DEV",
    "BaselineId": "pb-0123456789abcdef0",
    "OperatingSystem": "WINDOWS"
}
```

Para obter mais informações, consulte Criar um grupo de patches < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html >__ e Adicionar um grupo de patches a uma linha de base de patches no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte GetPatchBaselineForPatchGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo exibe a lista de referência de patches para um grupo de patches.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Para obter detalhes da API, consulte GetPatchBaselineForPatchGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe a lista de referência de patches para um grupo de patches.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Para obter detalhes da API, consulte GetPatchBaselineForPatchGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListAssociationVersions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListAssociationVersions`.

CLI

AWS CLI

Para listar todas as versões de uma associação de um ID de associação específico

O exemplo de `list-association-versions` a seguir lista todas as versões das associações especificadas.

```
aws ssm list-association-versions \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Saída:

```
{
"AssociationVersions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "CreatedDate": 1550505536.726,
            "Name": "AWS-UpdateSSMAgent",
            "Parameters": {
                "allowDowngrade": [
                    "false"
                ],
                "version": [
                    ""
                ]
            },
```

```
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-1234567890abcdef0"
                    ]
                }
            ],
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

Para obter mais informações, consulte Trabalhar com associações no Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte ListAssociationVersionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo recupera todas as versões da associação fornecida.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Saída:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
```

```
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- Para obter detalhes da API, consulte ListAssociationVersionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera todas as versões da associação fornecida.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Saída:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
```

```
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- Para obter detalhes da API, consulte ListAssociationVersionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListAssociations`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListAssociations`.

CLI

AWS CLI

Exemplo 1: listar suas associações para uma instância específica

O exemplo de associações de lista a seguir lista todas as associações com o, AssociationName updatessMagent.

```
aws ssm list-associations /
    --association-filter-list "key=AssociationName,value=UpdateSSMAgent"
```

Saída:

```
{
    "Associations": [
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-1234567890abcdef0",
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "Targets": [
                {
```

```
                    "Key": "InstanceIds",
                    "Values": [
                        "i-016648b75dd622dab"
                    ]
                }
            ],
            "Overview": {
                "Status": "Pending",
                "DetailedStatus": "Associated",
                "AssociationStatusAggregatedCount": {
                    "Pending": 1
                }
            },
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

Para obter mais informações, consulte Trabalhar com associações no Systems Manager no Guia do usuário do Systems Manager.

Exemplo 2: listar suas associações para um documento específico

O exemplo de associações de lista a seguir lista todas as associações do documento especificado.

```
aws ssm list-associations /
    --association-filter-list "key=Name,value=AWS-UpdateSSMAgent"
```

Saída:

```
{
    "Associations": [
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-1234567890abcdef0",
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
```

```
                        "i-1234567890abcdef0"
                    ]
                }
            ],
            "LastExecutionDate": 1550505828.548,
            "Overview": {
                "Status": "Success",
                "DetailedStatus": "Success",
                "AssociationStatusAggregatedCount": {
                    "Success": 1
                }
            },
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        },
    {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-9876543210abcdef0",
            "AssociationId": "fbc07ef7-b985-4684-b82b-0123456789ab",
            "AssociationVersion": "1",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-9876543210abcdef0"
                    ]
                }
            ],
            "LastExecutionDate": 1550507531.0,
            "Overview": {
                "Status": "Success",
                "AssociationStatusAggregatedCount": {
                    "Success": 1
                }
            }
        }
    ]
}
```

Para obter mais informações, consulte Trabalhar com associações no Systems Manager no Guia do usuário do Systems Manager.

- Para obter detalhes da API, consulte ListAssociationsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as associações para uma instância. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Exemplo 2: esse exemplo lista todas as associações para um documento de configuração. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar cada filtro.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
```

```
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- Para obter detalhes da API, consulte ListAssociationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as associações para uma instância. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Exemplo 2: esse exemplo lista todas as associações para um documento de configuração. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
```

```
Get-SSMAssociationList -AssociationFilterList $filter2
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar cada filtro.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- Para obter detalhes da API, consulte ListAssociationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`ListCommandInvocations`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListCommandInvocations`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para listar as invocações de um comando específico

O exemplo de `list-command-invocations` a seguir lista todas as invocações de um comando.

```
aws ssm list-command-invocations \
    --command-id "ef7fdfd8-9b57-4151-a15c-db9a12345678" \
    --details
```

Saída:

```
{
    "CommandInvocations": [
        {
            "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
            "InstanceId": "i-02573cafcfEXAMPLE",
            "InstanceName": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "RequestedDateTime": 1582136283.089,
            "Status": "Success",
            "StatusDetails": "Success",
            "StandardOutputUrl": "",
            "StandardErrorUrl": "",
            "CommandPlugins": [
                {
                    "Name": "aws:updateSsmAgent",
                    "Status": "Success",
                    "StatusDetails": "Success",
                    "ResponseCode": 0,
                    "ResponseStartDateTime": 1582136283.419,
                    "ResponseFinishDateTime": 1582136283.51,
                    "Output": "Updating amazon-ssm-agent from 2.3.842.0 to latest
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-
```

```
east-2/ssm-agent-manifest.json\namazon-ssm-agent 2.3.842.0 has already been
 installed, update skipped\n",
                    "StandardOutputUrl": "",
                    "StandardErrorUrl": "",
                    "OutputS3Region": "us-east-2",
                    "OutputS3BucketName": "",
                    "OutputS3KeyPrefix": ""
                }
            ],
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        },
        {
            "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
            "InstanceId": "i-0471e04240EXAMPLE",
            "InstanceName": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "RequestedDateTime": 1582136283.02,
            "Status": "Success",
            "StatusDetails": "Success",
            "StandardOutputUrl": "",
            "StandardErrorUrl": "",
            "CommandPlugins": [
                {
                    "Name": "aws:updateSsmAgent",
                    "Status": "Success",
                    "StatusDetails": "Success",
                    "ResponseCode": 0,
                    "ResponseStartDateTime": 1582136283.812,
                    "ResponseFinishDateTime": 1582136295.031,
                    "Output": "Updating amazon-ssm-agent from 2.3.672.0 to
 latest\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-
ssm-us-east-2/ssm-agent-manifest.json\nSuccessfully downloaded https://s3.us-
```

```
east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent-updater/2.3.842.0/
amazon-ssm-agent-updater-snap-amd64.tar.gz\nSuccessfully downloaded https://
s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent/2.3.672.0/
amazon-ssm-agent-snap-amd64.tar.gz\nSuccessfully downloaded https://s3.us-
east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent/2.3.842.0/amazon-ssm-
agent-snap-amd64.tar.gz\nInitiating amazon-ssm-agent update to 2.3.842.0\namazon-
ssm-agent updated successfully to 2.3.842.0",
                    "StandardOutputUrl": "",
                    "StandardErrorUrl": "",
                    "OutputS3Region": "us-east-2",
                    "OutputS3BucketName": "",
                    "OutputS3KeyPrefix": "8bee3135-398c-4d31-99b6-e42d2EXAMPLE/
i-0471e04240EXAMPLE/awsupdateSsmAgent"
                }
            ],
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
    ]
}
```

Para obter mais informações, consulte Entender os status dos comandos no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte ListCommandInvocationsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { paginateListCommandInvocations, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * List SSM command invocations on an instance.
 * @param {{ instanceId: string }}
 */
export const main = async ({ instanceId }) => {
  const client = new SSMClient({});
  try {
    const listCommandInvocationsPaginated = [];
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListCommandInvocations(
      { client },
      {
        InstanceId: instanceId,
      },
    );
    for await (const page of paginator) {
      listCommandInvocationsPaginated.push(...page.CommandInvocations);
    }
    console.log("Here is the list of command invocations:");
    console.log(listCommandInvocationsPaginated);
    return { CommandInvocations: listCommandInvocationsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid instance ID?`);
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte ListCommandInvocationsa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as invocações de um comando.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -
Detail $true
```

Saída:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :
```

Exemplo 2: Este exemplo lista a invocação do ID de comando CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Saída:

```
Name                   : aws:runPowerShellScript
Output                 : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
```

```
                          kumo available

OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Para obter detalhes da API, consulte ListCommandInvocationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as invocações de um comando.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -
Detail $true
```

Saída:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :
```

Exemplo 2: Este exemplo lista a invocação do ID de comando CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Saída:

```
Name                   : aws:runPowerShellScript
Output                 : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                          kumo available

OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Para obter detalhes da API, consulte ListCommandInvocationsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
```

```
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Para obter detalhes da API, consulte a ListCommandInvocationsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Use paginator to get all results
        DATA(lo_paginator) = lo_ssm->get_paginator( ).
        DATA(lo_iterator) = lo_paginator->listcommandinvocations(
          iv_instanceid = iv_instance_id ).

        DATA lv_count TYPE i VALUE 0.

        WHILE lo_iterator->has_next( ).
          DATA(lo_result) = CAST /aws1/cl_ssmlistcmdinvcsresult( lo_iterator-
>get_next( ) ).
          LOOP AT lo_result->get_commandinvocations( ) INTO DATA(lo_invocation).
            lv_count = lv_count + 1.
            DATA(lv_requested_datetime) = lo_invocation-
>get_requesteddatetime( ).
            MESSAGE |Command invocation requested at: { lv_requested_datetime }|
 TYPE 'I'.
          ENDLOOP.
        ENDWHILE.

        MESSAGE |{ lv_count } command invocation(s) found for instance
 { iv_instance_id }.| TYPE 'I'.
      CATCH /aws1/cx_ssminvalidinstanceid.
        MESSAGE 'Invalid instance ID.' TYPE 'I'.
```

```
      CATCH /aws1/cx_ssminvalidcommandid.
        MESSAGE 'Invalid command ID.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListCommandInvocationsreferência da API AWS SDK for SAP ABAP.

Usar **`ListCommands`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListCommands`.

CLI

AWS CLI

Exemplo 1: obter o status de um comando específico

O exemplo de `list-commands` a seguir recupera e exibe o status do comando especificado.

```
aws ssm list-commands \
    --command-id "0831e1a8-a1ac-4257-a1fd-c831bEXAMPLE"
```

Exemplo 2: para obter o status dos comandos solicitados após uma data específica

O exemplo de `list-commands` a seguir recupera os detalhes dos comandos solicitados após a data especificada.

```
aws ssm list-commands \
    --filter "key=InvokedAfter,value=2020-02-01T00:00:00Z"
```

Exemplo 3: Para listar todos os comandos solicitados em uma AWS conta

O `list-commands` exemplo a seguir lista todos os comandos solicitados pelos usuários na AWS conta atual e na região.

```
aws ssm list-commands
```

Saída:

```
{
```

```
    "Commands": [
        {
            "CommandId": "8bee3135-398c-4d31-99b6-e42d2EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "ExpiresAfter": "2020-02-19T11:28:02.500000-08:00",
            "Parameters": {},
            "InstanceIds": [
                "i-028ea792daEXAMPLE",
                "i-02feef8c46EXAMPLE",
                "i-038613f3f0EXAMPLE",
                "i-03a530a2d4EXAMPLE",
                "i-083b678d37EXAMPLE",
                "i-0dee81debaEXAMPLE"
            ],
            "Targets": [],
            "RequestedDateTime": "2020-02-19T10:18:02.500000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "",
            "OutputS3KeyPrefix": "",
            "MaxConcurrency": "50",
            "MaxErrors": "100%",
            "TargetCount": 6,
            "CompletedCount": 6,
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
        {
            "CommandId": "e9ade581-c03d-476b-9b07-26667EXAMPLE",
            "DocumentName": "AWS-FindWindowsUpdates",
            "DocumentVersion": "1",
```

```
            "Comment": "",
            "ExpiresAfter": "2020-01-24T12:37:31.874000-08:00",
            "Parameters": {
                "KbArticleIds": [
                    ""
                ],
                "UpdateLevel": [
                    "All"
                ]
            },
            "InstanceIds": [],
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-00ec29b21eEXAMPLE",
                        "i-09911ddd90EXAMPLE"
                    ]
                }
            ],
            "RequestedDateTime": "2020-01-24T11:27:31.874000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "my-us-east-2-bucket",
            "OutputS3KeyPrefix": "my-rc-output",
            "MaxConcurrency": "50",
            "MaxErrors": "0",
            "TargetCount": 2,
            "CompletedCount": 2,
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
            "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "NotificationConfig": {
                "NotificationArn": "arn:aws:sns:us-east-2:111222333444:my-us-
east-2-notification-arn",
                "NotificationEvents": [
                    "All"
                ],
                "NotificationType": "Invocation"
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
```

```
            }
        }
        {
            "CommandId": "d539b6c3-70e8-4853-80e5-0ce4fEXAMPLE",
            "DocumentName": "AWS-RunPatchBaseline",
            "DocumentVersion": "1",
            "Comment": "",
            "ExpiresAfter": "2020-01-24T12:21:04.350000-08:00",
            "Parameters": {
                "InstallOverrideList": [
                    ""
                ],
                "Operation": [
                    "Install"
                ],
                "RebootOption": [
                    "RebootIfNeeded"
                ],
                "SnapshotId": [
                    ""
                ]
            },
            "InstanceIds": [],
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-00ec29b21eEXAMPLE",
                        "i-09911ddd90EXAMPLE"
                    ]
                }
            ],
            "RequestedDateTime": "2020-01-24T11:11:04.350000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "my-us-east-2-bucket",
            "OutputS3KeyPrefix": "my-rc-output",
            "MaxConcurrency": "50",
            "MaxErrors": "0",
            "TargetCount": 2,
            "CompletedCount": 2,
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
```

```
            "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "NotificationConfig": {
                "NotificationArn": "arn:aws:sns:us-east-2:111222333444:my-us-
east-2-notification-arn",
                "NotificationEvents": [
                    "All"
                ],
                "NotificationType": "Invocation"
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
    ]
}
```

Para obter mais informações, consulte Executar comandos usando o Systems Manager Run Command no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte ListCommandsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todos os comandos solicitados.

```
Get-SSMCommand
```

Saída:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
CompletedCount     : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
```

```
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Exemplo 2: esse exemplo obtém o status de um comando específico

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Exemplo 3: Este exemplo recupera todos os comandos SSM invocados após 01/04/2019 T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} |
 Select-Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -
Property RequestedDateTime -Descending
```

Saída:

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled
 4/16/2019 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success
 4/6/2019 9:19:42 AM
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed
 4/2/2019 4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed
 4/1/2019 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success
 4/1/2019 1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed
 4/1/2019 11:13:09 AM
```

- Para obter detalhes da API, consulte ListCommandsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os comandos solicitados.

```
Get-SSMCommand
```

Saída:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
CompletedCount     : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Exemplo 2: esse exemplo obtém o status de um comando específico

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Exemplo 3: Este exemplo recupera todos os comandos SSM invocados após 01/04/2019 T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} |
 Select-Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -
Property RequestedDateTime -Descending
```

Saída:

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled
 4/16/2019 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success
 4/6/2019 9:19:42 AM
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed
 4/2/2019 4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed
 4/1/2019 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success
 4/1/2019 1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed
 4/1/2019 11:13:09 AM
```

- Para obter detalhes da API, consulte ListCommandsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListComplianceItems`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListComplianceItems`.

CLI

AWS CLI

Para listar itens de conformidade para uma instância específica

Este exemplo lista todos os itens de conformidade para a instância especificada.

Comando:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance"
```

Saída:

```
{
  "ComplianceItems": [
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Id": "8dfe3659-4309-493a-8755-0123456789ab",
          "Title": "",
          "Status": "COMPLIANT",
          "Severity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550408470.0
          },
          "Details": {
              "DocumentName": "AWS-GatherSoftwareInventory",
              "DocumentVersion": "1"
          }
      },
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Id": "e4c2ed6d-516f-41aa-aa2a-0123456789ab",
          "Title": "",
          "Status": "COMPLIANT",
          "Severity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550508475.0
          },
          "Details": {
              "DocumentName": "AWS-UpdateSSMAgent",
              "DocumentVersion": "1"
          }
      },
              ...
  ],
  "NextToken": "--token string truncated--"
}
```

Para listar itens de conformidade para uma instância e um ID de associação específicos

Este exemplo lista todos os itens de conformidade para a instância e o ID de associação especificados.

Comando:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance" --
filters "Key=ComplianceType,Values=Association,Type=EQUAL" "Key=Id,Values=e4c2ed6d-516f-4
aa2a-0123456789ab,Type=EQUAL"
```

Para listar itens de conformidade para uma instância específica após uma data e uma hora específicas

Este exemplo lista todos os itens de conformidade para uma instância após a data e a hora especificadas.

Comando:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance" --
filters "Key=ExecutionTime,Values=2019-02-18T16:00:00Z,Type=GREATER_THAN"
```

- Para obter detalhes da API, consulte ListComplianceItemsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo mostra a lista de itens de conformidade para o ID e tipo de recurso fornecidos, filtrados pelo tipo de conformidade "Associação"

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Saída:

```
ComplianceType   : Association
Details          : {[DocumentName, AWS-GatherSoftwareInventory],
 [DocumentVersion, 1]}
```

```
ExecutionSummary :
 Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id                : 123a45a1-c234-1234-1245-67891236db4e
ResourceId        : i-1a2caf345f67d0dc2
ResourceType      : ManagedInstance
Severity          : UNSPECIFIED
Status            : COMPLIANT
Title             :
```

- Para obter detalhes da API, consulte ListComplianceItemsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra a lista de itens de conformidade para o ID e tipo de recurso fornecidos, filtrados pelo tipo de conformidade "Associação"

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Saída:

```
ComplianceType    : Association
Details           : {[DocumentName, AWS-GatherSoftwareInventory],
 [DocumentVersion, 1]}
ExecutionSummary :
 Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id                : 123a45a1-c234-1234-1245-67891236db4e
ResourceId        : i-1a2caf345f67d0dc2
ResourceType      : ManagedInstance
Severity          : UNSPECIFIED
Status            : COMPLIANT
Title             :
```

- Para obter detalhes da API, consulte ListComplianceItemsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListComplianceSummaries`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListComplianceSummaries`.

CLI

AWS CLI

Para listar resumos de conformidade para todos os tipos de conformidade

Este exemplo lista resumos de conformidade para todos os tipos de conformidade em sua conta.

Comando:

```
aws ssm list-compliance-summaries
```

Saída:

```
{
  "ComplianceSummaryItems": [
      {
          "ComplianceType": "Association",
          "CompliantSummary": {
              "CompliantCount": 2,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 2
              }
          },
          "NonCompliantSummary": {
              "NonCompliantCount": 0,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 0
              }
          }
      },
      {
```

```
            "ComplianceType": "Patch",
            "CompliantSummary": {
                "CompliantCount": 1,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 1
                }
            },
            "NonCompliantSummary": {
                "NonCompliantCount": 1,
                "SeveritySummary": {
                    "CriticalCount": 1,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 0
                }
            }
        },
                ...
    ],
    "NextToken": "eyJOZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}
```

Para listar resumos de conformidade para um tipo de conformidade específico

Este exemplo lista o resumo de conformidade para o tipo de conformidade do patch.

Comando:

```
aws ssm list-compliance-summaries --
filters "Key=ComplianceType,Values=Patch,Type=EQUAL"
```

- Para obter detalhes da API, consulte ListComplianceSummariesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo devolve uma contagem resumida de recursos em ou fora de conformidade para todos os tipos de conformidade.

```
Get-SSMComplianceSummaryList
```

Saída:

```
ComplianceType CompliantSummary
 NonCompliantSummary
-------------- ----------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- Para obter detalhes da API, consulte ListComplianceSummariesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo devolve uma contagem resumida de recursos em ou fora de conformidade para todos os tipos de conformidade.

```
Get-SSMComplianceSummaryList
```

Saída:

```
ComplianceType CompliantSummary
 NonCompliantSummary
-------------- ----------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

```
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- Para obter detalhes da API, consulte ListComplianceSummariesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListDocumentVersions`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListDocumentVersions`.

CLI

AWS CLI

Para listar versões de documentos

O exemplo de `list-document-versions` a seguir lista todas as versões de um documento do Systems Manager.

```
aws ssm list-document-versions \
    --name "Example"
```

Saída:

```
{
    "DocumentVersions": [
        {
            "Name": "Example",
            "DocumentVersion": "1",
            "CreatedDate": 1583257938.266,
            "IsDefaultVersion": true,
            "DocumentFormat": "YAML",
            "Status": "Active"
        }
    ]
}
```

Para obter mais informações, consulte Enviar comandos que usam o parâmetro de versão do documento no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte ListDocumentVersionsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as versões de um documento.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

Saída:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
Status            : Active
```

- Para obter detalhes da API, consulte ListDocumentVersionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as versões de um documento.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

Saída:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
Status            : Active
```

- Para obter detalhes da API, consulte ListDocumentVersionsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListDocuments`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListDocuments`.

CLI

AWS CLI

Exemplo 1: listar documentos

O exemplo de `list-documents` a seguir lista documentos pertencentes à conta solicitante marcados com a tag personalizada.

```
aws ssm list-documents \
    --filters Key=Owner,Values=Self Key=tag:DocUse,Values=Testing
```

Saída:

```
{
    "DocumentIdentifiers": [
        {
            "Name": "Example",
            "Owner": "29884EXAMPLE",
            "PlatformTypes": [
                "Windows",
                "Linux"
            ],
            "DocumentVersion": "1",
            "DocumentType": "Automation",
            "SchemaVersion": "0.3",
            "DocumentFormat": "YAML",
            "Tags": [
                {
                    "Key": "DocUse",
                    "Value": "Testing"
                }
            ]
        }
    ]
```

```
}
```

Para obter mais informações, consulte Documentos do AWS Systems Manager no Guia do usuário do AWS Systems Manager.

Exemplo 2: listar documentos compartilhados

O `list-documents` exemplo a seguir lista documentos compartilhados, incluindo documentos compartilhados privados que não são de propriedade da AWS.

```
aws ssm list-documents \
    --filters Key=Name,Values=sharedDocNamePrefix  Key=Owner,Values=Private
```

Saída:

```
{
    "DocumentIdentifiers": [
        {
            "Name": "Example",
            "Owner": "12345EXAMPLE",
            "PlatformTypes": [
                "Windows",
                "Linux"
            ],
            "DocumentVersion": "1",
            "DocumentType": "Command",
            "SchemaVersion": "0.3",
            "DocumentFormat": "YAML",
            "Tags": []
        }
    ]
}
```

Para obter mais informações, consulte Documentos do AWS Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte ListDocumentsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: lista todos os documentos de configuração em sua conta.

```
Get-SSMDocumentList
```

Saída:

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Exemplo 2: esse exemplo recupera todos os documentos de automação com o nome correspondente a "Plataform"

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"}
 | Where-Object Name -Match "Platform"
```

Saída:

```
DocumentFormat  : JSON
```

```
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- Para obter detalhes da API, consulte ListDocumentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: lista todos os documentos de configuração em sua conta.

```
Get-SSMDocumentList
```

Saída:

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Exemplo 2: esse exemplo recupera todos os documentos de automação com o nome correspondente a "Plataform"

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"}
 | Where-Object Name -Match "Platform"
```

Saída:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- Para obter detalhes da API, consulte ListDocumentsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListInventoryEntries`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListInventoryEntries`.

CLI

AWS CLI

Exemplo 1: visualizar entradas específicas do tipo de inventário para uma instância

O `list-inventory-entries` exemplo a seguir lista as entradas de inventário do tipo de inventário:Application AWS em uma instância específica.

```
aws ssm list-inventory-entries \
    --instance-id "i-1234567890abcdef0" \
    --type-name "AWS:Application"
```

Saída:

```
{
  "TypeName": "AWS:Application",
  "InstanceId": "i-1234567890abcdef0",
  "SchemaVersion": "1.1",
  "CaptureTime": "2019-02-15T12:17:55Z",
  "Entries": [
    {
      "Architecture": "i386",
      "Name": "Amazon SSM Agent",
      "PackageId": "{88a60be2-89a1-4df8-812a-80863c2a2b68}",
      "Publisher": "Amazon Web Services",
      "Version": "2.3.274.0"
    },
    {
      "Architecture": "x86_64",
      "InstalledTime": "2018-05-03T13:42:34Z",
      "Name": "AmazonCloudWatchAgent",
      "Publisher": "",
      "Version": "1.200442.0"
    }
  ]
}
```

Exemplo 2: visualizar entradas de inventário personalizadas atribuídas a uma instância

O exemplo de `list-inventory-entries` a seguir lista uma entrada de inventário personalizada atribuída a uma instância.

```
aws ssm list-inventory-entries \
    --instance-id "i-1234567890abcdef0" \
    --type-name "Custom:RackInfo"
```

Saída:

```
{
  "TypeName": "Custom:RackInfo",
  "InstanceId": "i-1234567890abcdef0",
  "SchemaVersion": "1.0",
  "CaptureTime": "2021-05-22T10:01:01Z",
  "Entries": [
    {
      "RackLocation": "Bay B/Row C/Rack D/Shelf E"
```

```
    }
  ]
}
```

- Para obter detalhes da API, consulte ListInventoryEntriesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista todas as entradas de inventário personalizadas para uma instância.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Saída:

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Exemplo 2: esse exemplo lista os detalhes.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Saída:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Para obter detalhes da API, consulte ListInventoryEntriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as entradas de inventário personalizadas para uma instância.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Saída:

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Exemplo 2: esse exemplo lista os detalhes.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Saída:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Para obter detalhes da API, consulte ListInventoryEntriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListResourceComplianceSummaries`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListResourceComplianceSummaries`.

CLI

AWS CLI

Para listar as contagens resumidas de conformidade em nível de recurso

Este exemplo lista as contagens resumidas de conformidade em nível de recurso.

Comando:

```
aws ssm list-resource-compliance-summaries
```

Saída:

```
{
  "ResourceComplianceSummaryItems": [
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Status": "COMPLIANT",
          "OverallSeverity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550509273.0
          },
          "CompliantSummary": {
              "CompliantCount": 2,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 2
              }
          },
          "NonCompliantSummary": {
              "NonCompliantCount": 0,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 0
              }
          }
      },
      {
```

```
            "ComplianceType": "Patch",
            "ResourceType": "ManagedInstance",
            "ResourceId": "i-9876543210abcdef0",
            "Status": "COMPLIANT",
            "OverallSeverity": "UNSPECIFIED",
            "ExecutionSummary": {
                "ExecutionTime": 1550248550.0,
                "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
                "ExecutionType": "Command"
            },
            "CompliantSummary": {
                "CompliantCount": 397,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 397
                }
            },
            "NonCompliantSummary": {
                "NonCompliantCount": 0,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 0
                }
            }
        }
    ],
    "NextToken": "--token string truncated--"
}
```

Para listar resumos de conformidade em nível de recurso para um tipo de conformidade específico

Este exemplo lista resumos de conformidade em nível de recurso para o tipo de conformidade do patch.

Comando:

```
aws ssm list-resource-compliance-summaries --
filters "Key=ComplianceType,Values=Patch,Type=EQUAL"
```

- Para obter detalhes da API, consulte ListResourceComplianceSummariesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo obtém uma contagem resumida em nível de recurso. O resumo inclui informações sobre status de conformidade e não conformidade e contagens detalhadas de gravidade de itens de conformidade para produtos que correspondem a "Windows10". Como o MaxResult padrão é 100 se o parâmetro não for especificado e esse valor não for válido, o MaxResult parâmetro será adicionado e o valor será definido como 50.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Para obter detalhes da API, consulte ListResourceComplianceSummariesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma contagem resumida em nível de recurso. O resumo inclui informações sobre status de conformidade e não conformidade e contagens detalhadas de gravidade de itens de conformidade para produtos que correspondem a "Windows10". Como o MaxResult padrão é 100 se o parâmetro não for especificado e esse valor não for válido, o MaxResult parâmetro será adicionado e o valor será definido como 50.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
```

```
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Para obter detalhes da API, consulte ListResourceComplianceSummariesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ListTagsForResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ListTagsForResource`.

CLI

AWS CLI

Para listar as tags aplicadas a uma lista de referência de patches

O exemplo de `list-tags-for-resource` a seguir lista as tags para uma lista de referência de patches.

```
aws ssm list-tags-for-resource \
    --resource-type "PatchBaseline" \
    --resource-id "pb-0123456789abcdef0"
```

Saída:

```
{
    "TagList": [
        {
            "Key": "Environment",
            "Value": "Production"
        },
        {
            "Key": "Region",
            "Value": "EMEA"
        }
    ]
}
```

Para obter mais informações, consulte AWS Recursos de marcação na Referência AWS geral.

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo lista as tags para uma janela de manutenção.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Saída:

```
Key   Value
---   -----
Stack Production
```

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as tags para uma janela de manutenção.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Saída:

```
Key   Value
---   -----
Stack Production
```

- Para obter detalhes da API, consulte ListTagsForResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyDocumentPermission`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyDocumentPermission`.

CLI

AWS CLI

Para modificar as permissões do documento

O exemplo de `modify-document-permission` a seguir compartilha publicamente um documento do Systems Manager.

```
aws ssm modify-document-permission \
    --name "Example" \
    --permission-type "Share" \
    --account-ids-to-add "All"
```

Este comando não produz saída.

Para obter mais informações, consulte Compartilhar um documento do Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte ModifyDocumentPermissionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona permissões de "compartilhamento" a todas as contas para um documento. Não haverá saída se o comando for bem-sucedido.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Exemplo 2: esse exemplo adiciona permissões de "compartilhamento" a uma conta específica para um documento. Não haverá saída se o comando for bem-sucedido.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Para obter detalhes da API, consulte ModifyDocumentPermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona permissões de "compartilhamento" a todas as contas para um documento. Não haverá saída se o comando for bem-sucedido.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Exemplo 2: esse exemplo adiciona permissões de "compartilhamento" a uma conta específica para um documento. Não haverá saída se o comando for bem-sucedido.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Para obter detalhes da API, consulte ModifyDocumentPermissionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutComplianceItems`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutComplianceItems`.

CLI

AWS CLI

Para registrar um tipo de conformidade e detalhes de conformidade em uma instância designada

Este exemplo registra o tipo de conformidade `Custom:AVCheck` na instância gerenciada especificada. Não haverá saída se o comando for bem-sucedido.

Comando:

```
aws ssm put-compliance-items --resource-id "i-1234567890abcdef0" --
resource-type "ManagedInstance" --compliance-type "Custom:AVCheck"
 --execution-summary "ExecutionTime=2019-02-18T16:00:00Z" --
items "Id=Version2.0,Title=ScanHost,Severity=CRITICAL,Status=COMPLIANT"
```

- Para obter detalhes da API, consulte PutComplianceItemsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo grava um item de conformidade personalizado para a instância gerenciada especificada

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- Para obter detalhes da API, consulte PutComplianceItemsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo grava um item de conformidade personalizado para a instância gerenciada especificada

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- Para obter detalhes da API, consulte PutComplianceItemsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`PutInventory`** com uma CLI

Os exemplos de código a seguir mostram como usar o `PutInventory`.

CLI

AWS CLI

Para atribuir metadados de inventário personalizados a uma instância

Este exemplo atribui informações de localização de rack a uma instância. Não haverá saída se o comando for bem-sucedido.

Comando (Linux):

```
aws ssm put-inventory --instance-id "i-016648b75dd622dab" --items
 '[{"TypeName": "Custom:RackInfo","SchemaVersion": "1.0","CaptureTime":
 "2019-01-22T10:01:01Z","Content":[{"RackLocation": "Bay B/Row C/Rack D/Shelf
 E"}]}]'
```

Comando (Windows):

```
aws ssm put-inventory --instance-id "i-016648b75dd622dab" --
items "TypeName=Custom:RackInfo,SchemaVersion=1.0,CaptureTime=2019-01-22T10:01:01Z,Conten
 B/Row C/Rack D/Shelf F'}]"
```

- Para obter detalhes da API, consulte PutInventoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atribui informações de localização de rack a uma instância. Não haverá saída se o comando for bem-sucedido.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
$items.Add($data)

$customInventoryItem = New-Object
 Amazon.SimpleSystemsManagement.Model.InventoryItem
```

```
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- Para obter detalhes da API, consulte PutInventoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atribui informações de localização de rack a uma instância. Não haverá saída se o comando for bem-sucedido.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
$items.Add($data)

$customInventoryItem = New-Object
 Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- Para obter detalhes da API, consulte PutInventoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`PutParameter`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `PutParameter`.

CLI

AWS CLI

Exemplo 1: como alterar o valor de um parâmetro

O exemplo de `put-parameter` a seguir altera o valor do parâmetro especificado.

```
aws ssm put-parameter \
    --name "MyStringParameter" \
    --type "String" \
    --value "Vici" \
    --overwrite
```

Saída:

```
{
    "Version": 2,
    "Tier": "Standard"
}
```

Para obter mais informações, consulte Criar um parâmetro do Systems Manager (AWS CLI), Gerenciar camadas de parâmetros e Trabalhar com políticas de parâmetros no Guia do usuário do AWS Systems Manager.

Exemplo 2: como criar um parâmetro avançado

O exemplo de `put-parameter` a seguir cria um parâmetro avançado.

```
aws ssm put-parameter \
    --name "MyAdvancedParameter" \
    --description "This is an advanced parameter" \
    --value "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
 veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
 consequat [truncated]" \
    --type "String" \
    --tier Advanced
```

Saída:

```
{
    "Version": 1,
    "Tier": "Advanced"
}
```

Para obter mais informações, consulte Criar um parâmetro do Systems Manager (AWS CLI), Gerenciar camadas de parâmetros e Trabalhar com políticas de parâmetros no Guia do usuário do AWS Systems Manager.

Exemplo 3: como converter um parâmetro padrão em um parâmetro avançado

O exemplo `put-parameter` a seguir converte um parâmetro padrão existente em um parâmetro avançado.

```
aws ssm put-parameter \
    --name "MyConvertedParameter" \
    --value "abc123" \
    --type "String" \
    --tier Advanced \
    --overwrite
```

Saída:

```
{
    "Version": 2,
    "Tier": "Advanced"
}
```

Para obter mais informações, consulte Criar um parâmetro do Systems Manager (AWS CLI), Gerenciar camadas de parâmetros e Trabalhar com políticas de parâmetros no Guia do usuário do AWS Systems Manager.

Exemplo 4: como criar um parâmetro com uma política anexada

O exemplo de `put-parameter` a seguir cria um parâmetro avançado com uma política de parâmetros anexada.

```
aws ssm put-parameter \
    --name "/Finance/Payroll/q2accesskey" \
    --value "P@sSwW)rd" \
```

```
    --type "SecureString" \
    --tier Advanced \
    --policies "[{\"Type\":\"Expiration\",\"Version\":\"1.0\",\"Attributes\":
{\"Timestamp\":\"2020-06-30T00:00:00.000Z\"}},{\"Type\":\"ExpirationNotification
\",\"Version\":\"1.0\",\"Attributes\":{\"Before\":\"5\",\"Unit\":\"Days\"}},
{\"Type\":\"NoChangeNotification\",\"Version\":\"1.0\",\"Attributes\":{\"After\":
\"60\",\"Unit\":\"Days\"}}]"
```

Saída:

```
{
    "Version": 1,
    "Tier": "Advanced"
}
```

Para obter mais informações, consulte Criar um parâmetro do Systems Manager (AWS CLI), Gerenciar camadas de parâmetros e Trabalhar com políticas de parâmetros no Guia do usuário do AWS Systems Manager.

Exemplo 5: como adicionar uma política a um parâmetro existente

O exemplo de `put-parameter` a seguir anexa uma política a um parâmetro avançado existente.

```
aws ssm put-parameter \
    --name "/Finance/Payroll/q2accesskey" \
    --value "N3wP@sSwW)rd" \
    --type "SecureString" \
    --tier Advanced \
    --policies "[{\"Type\":\"Expiration\",\"Version\":\"1.0\",\"Attributes\":
{\"Timestamp\":\"2020-06-30T00:00:00.000Z\"}},{\"Type\":\"ExpirationNotification
\",\"Version\":\"1.0\",\"Attributes\":{\"Before\":\"5\",\"Unit\":\"Days\"}},
{\"Type\":\"NoChangeNotification\",\"Version\":\"1.0\",\"Attributes\":{\"After\":
\"60\",\"Unit\":\"Days\"}}]"
    --overwrite
```

Saída:

```
{
    "Version": 2,
    "Tier": "Advanced"
```

```
}
```

Para obter mais informações, consulte Criar um parâmetro do Systems Manager (AWS CLI), Gerenciar camadas de parâmetros e Trabalhar com políticas de parâmetros no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte PutParameterem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                    paraValue - The value of the parameter.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String paraName = args[0];
        String paraValue = args[1];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        putParaValue(ssmClient, paraName, paraValue);
        ssmClient.close();
    }

    public static void putParaValue(SsmClient ssmClient, String paraName, String
 value) {
        try {
            PutParameterRequest parameterRequest = PutParameterRequest.builder()
                    .name(paraName)
                    .type(ParameterType.STRING)
                    .value(value)
                    .build();

            ssmClient.putParameter(parameterRequest);
            System.out.println("The parameter was successfully added.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutParameter a Referência AWS SDK for Java 2.x da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo cria um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Exemplo 2: esse exemplo altera um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- Para obter detalhes da API, consulte PutParameterem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Exemplo 2: esse exemplo altera um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- Para obter detalhes da API, consulte PutParameterem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_parameter(
    client: &Client,
    name: &str,
    value: &str,
    description: &str,
```

```
) -> Result<(), Error> {
    let resp = client
        .put_parameter()
        .overwrite(true)
        .r#type(ParameterType::String)
        .name(name)
        .value(value)
        .description(description)
        .send()
        .await?;

    println!("Success! Parameter now has version: {}", resp.version());

    Ok(())
}
```

- Para obter detalhes da API, consulte a PutParameterreferência da API AWS SDK for Rust.

Usar **`RegisterDefaultPatchBaseline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterDefaultPatchBaseline`.

CLI

AWS CLI

Para definir a lista de referência de patches padrão

O exemplo de `register-default-patch-baseline` a seguir registra a lista de referência de patches personalizada especificada como a lista de referência de patches padrão para o tipo de sistema operacional ao qual ela oferece suporte.

```
aws ssm register-default-patch-baseline \
    --baseline-id "pb-abc123cf9bEXAMPLE"
```

Saída:

```
{
    "BaselineId":"pb-abc123cf9bEXAMPLE"
}
```

O `register-default-patch-baseline` exemplo a seguir registra a linha de base do patch padrão fornecida pelo AWS CentOS como a linha de base do patch padrão.

```
aws ssm register-default-patch-baseline \
    --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-0574b43a65ea646ed"
```

Saída:

```
{
    "BaselineId":"pb-abc123cf9bEXAMPLE"
}
```

Para obter mais informações, consulte Sobre listas de referência de patches predefinidas e personalizadas no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte RegisterDefaultPatchBaselineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo registra uma lista de referência de patches como a lista de referência de patches padrão.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Saída:

```
pb-03da896ca3b68b639
```

- Para obter detalhes da API, consulte RegisterDefaultPatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo registra uma lista de referência de patches como a lista de referência de patches padrão.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Saída:

```
pb-03da896ca3b68b639
```

- Para obter detalhes da API, consulte RegisterDefaultPatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RegisterPatchBaselineForPatchGroup`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterPatchBaselineForPatchGroup`.

CLI

AWS CLI

Para registrar uma lista de referência de patches de um grupo de patches

O exemplo de `register-patch-baseline-for-patch-group` a seguir registra uma lista de referência de patches para um grupo de patches.

```
aws ssm register-patch-baseline-for-patch-group \
    --baseline-id "pb-045f10b4f382baeda" \
    --patch-group "Production"
```

Saída:

```
{
    "BaselineId": "pb-045f10b4f382baeda",
    "PatchGroup": "Production"
}
```

Para obter mais informações, consulte Criar um grupo de patches < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html >__ e Adicionar um grupo de patches a uma linha de base de patches no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte RegisterPatchBaselineForPatchGroupem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra uma lista de referência de patches para um grupo de patches.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- Para obter detalhes da API, consulte RegisterPatchBaselineForPatchGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma lista de referência de patches para um grupo de patches.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- Para obter detalhes da API, consulte RegisterPatchBaselineForPatchGroupem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RegisterTargetWithMaintenanceWindow`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterTargetWithMaintenanceWindow`.

CLI

AWS CLI

Exemplo 1: registrar um único destino com uma janela de manutenção

O exemplo de `register-target-with-maintenance-window` a seguir registra uma instância com uma janela de manutenção.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --target "Key=InstanceIds,Values=i-0000293ffd8c57862" \
    --owner-information "Single instance" \
    --resource-type "INSTANCE"
```

Saída:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Exemplo 2: registrar vários destinos em uma janela de manutenção usando IDs de instância

O exemplo de `register-target-with-maintenance-window` a seguir registra duas instâncias com uma janela de manutenção especificando seus IDs de instância.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --target "Key=InstanceIds,Values=i-0000293ffd8c57862,i-0cb2b964d3e14fd9f" \
    --owner-information "Two instances in a list" \
    --resource-type "INSTANCE"
```

Saída:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Exemplo 3: registrar destinos com uma janela de manutenção usando tags de recursos

O exemplo de `register-target-with-maintenance-window` a seguir registra instâncias com uma janela de manutenção especificando tags de recursos que foram aplicadas às instâncias.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-06cf17cbefcb4bf4f" \
    --targets "Key=tag:Environment,Values=Prod" "Key=Role,Values=Web" \
    --owner-information "Production Web Servers" \
    --resource-type "INSTANCE"
```

Saída:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Exemplo 4: registrar destinos usando um grupo de chaves de tag

O exemplo de `register-target-with-maintenance-window` a seguir registra instâncias que têm uma ou mais chaves de tags atribuídas a elas, independentemente de seus valores de chave.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
    --resource-type "INSTANCE" \
    --target "Key=tag-key,Values=Name,Instance-Type,CostCenter"
```

Saída:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Exemplo 5: registrar destinos usando um nome de grupo de recursos

O exemplo de `register-target-with-maintenance-window` a seguir registra um grupo de recursos especificado, independentemente do tipo de recurso que ele contém.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
```

```
    --resource-type "RESOURCE_GROUP" \
    --target "Key=resource-groups:Name,Values=MyResourceGroup"
```

Saída:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Para obter mais informações, consulte Registrar uma instância de destino com a janela de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte RegisterTargetWithMaintenanceWindowem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra uma instância com uma janela de manutenção.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Saída:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Exemplo 2: esse exemplo registra várias instâncias com uma janela de manutenção.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Saída:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Exemplo 3: esse exemplo registra uma instância com uma janela de manutenção usando tags do EC2.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Saída:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Para obter detalhes da API, consulte RegisterTargetWithMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma instância com uma janela de manutenção.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Saída:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Exemplo 2: esse exemplo registra várias instâncias com uma janela de manutenção.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Saída:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Exemplo 3: esse exemplo registra uma instância com uma janela de manutenção usando tags do EC2.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Saída:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Para obter detalhes da API, consulte RegisterTargetWithMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RegisterTaskWithMaintenanceWindow`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterTaskWithMaintenanceWindow`.

CLI

AWS CLI

Exemplo 1: registrar uma tarefa do Automation com uma janela de manutenção

O exemplo de `register-task-with-maintenance-window` a seguir registra uma tarefa do Automation com uma janela de manutenção voltada para uma instância.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-082dcd7649EXAMPLE" \
    --targets Key=InstanceIds,Values=i-1234520122EXAMPLE \
    --task-arn AWS-RestartEC2Instance \
    --service-role-arn arn:aws:iam::111222333444:role/SSM --task-type AUTOMATION
 \
    --task-invocation-parameters "{\"Automation\":{\"DocumentVersion\":\"\$LATEST
\",\"Parameters\":{\"InstanceId\":[\"{{RESOURCE_ID}}\"]}}}" \
    --priority 0 \
    --max-concurrency 1 \
    --max-errors 1 \
    --name "AutomationExample" \
    --description "Restarting EC2 Instance for maintenance"
```

Saída:

```
{
```

```
    "WindowTaskId":"11144444-5555-6666-7777-88888888"
}
```

Para obter mais informações, consulte Registrar uma tarefa com a janela de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

Exemplo 2: registrar uma tarefa do Lambda com uma janela de manutenção

O exemplo de `register-task-with-maintenance-window` a seguir registra uma tarefa do Lambda com uma janela de manutenção voltada para uma instância.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-082dcd7649dee04e4" \
    --targets Key=InstanceIds,Values=i-12344d305eEXAMPLE \
    --task-arn arn:aws:lambda:us-east-1:111222333444:function:SSMTestLAMBDA \
    --service-role-arn arn:aws:iam::111222333444:role/SSM \
    --task-type LAMBDA \
    --task-invocation-parameters '{"Lambda":{"Payload":"{\"InstanceId\":
\"{{RESOURCE_ID}}\",\"targetType\":\"{{TARGET_TYPE}}\"}","Qualifier":"$LATEST"}}'
 \
    --priority 0 \
    --max-concurrency 10 \
    --max-errors 5 \
    --name "Lambda_Example" \
    --description "My Lambda Example"
```

Saída:

```
{
    "WindowTaskId":"22244444-5555-6666-7777-88888888"
}
```

Para obter mais informações, consulte Registrar uma tarefa com a janela de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

Exemplo 3: registrar uma tarefa do Run Command com uma janela de manutenção

O exemplo de `register-task-with-maintenance-window` a seguir registra uma tarefa do Run Command com uma janela de manutenção voltada para uma instância.

```
aws ssm register-task-with-maintenance-window \
```

```
    --window-id "mw-082dcd7649dee04e4" \
    --targets "Key=InstanceIds,Values=i-12344d305eEXAMPLE" \
    --service-role-arn "arn:aws:iam::111222333444:role/SSM" \
    --task-type "RUN_COMMAND" \
    --name "SSMInstallPowerShellModule" \
    --task-arn "AWS-InstallPowerShellModule" \
    --task-invocation-parameters "{\"RunCommand\":{\"Comment\":\"\",
\"OutputS3BucketName\":\"runcommandlogs\",\"Parameters\":{\"commands\":[\"Get-
Module -ListAvailable\"],\"executionTimeout\":[\"3600\"],\"source\":[\"https:\/
\/gallery.technet.microsoft.com\/EZOut-33ae0fb7\/file\/110351\/1\/EZOut.zip\"],
\"workingDirectory\":[\"\\\\\"]},\"TimeoutSeconds\":600}}" \
    --max-concurrency 1 \
    --max-errors 1 \
    --priority 10
```

Saída:

```
{
    "WindowTaskId":"33344444-5555-6666-7777-88888888"
}
```

Para obter mais informações, consulte Registrar uma tarefa com a janela de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

Exemplo 4: registrar uma tarefa do Step Functions com uma janela de manutenção

O exemplo de `register-task-with-maintenance-window` a seguir registra uma tarefa do Step Functions com uma janela de manutenção voltada para uma instância.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-1234d787d6EXAMPLE" \
    --targets Key=WindowTargetIds,Values=12347414-69c3-49f8-95b8-ed2dcEXAMPLE \
    --task-arn arn:aws:states:us-
east-1:111222333444:stateMachine:SSMTestStateMachine \
    --service-role-arn arn:aws:iam::111222333444:role/MaintenanceWindows \
    --task-type STEP_FUNCTIONS \
    --task-invocation-parameters '{"StepFunctions":{"Input":"{\"InstanceId\":
\"{{RESOURCE_ID}}\"}"}}' \
    --priority 0 \
    --max-concurrency 10 \
    --max-errors 5 \
    --name "Step_Functions_Example" \
```

```
    --description "My Step Functions Example"
```

Saída:

```
{
    "WindowTaskId":"44444444-5555-6666-7777-88888888"
}
```

Para obter mais informações, consulte Registrar uma tarefa com a janela de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

Exemplo 5: registrar uma tarefa usando um ID de destino de janela de manutenção

O exemplo de `register-task-with-maintenance-window` a seguir registrar uma tarefa usando um ID de destino de janela de manutenção. O ID de destino da janela de manutenção estava presente na saída do comando `aws ssm register-target-with-maintenance-window`. Também é possível recuperá-lo da saída do comando `aws ssm describe-maintenance-window-targets`.

```
aws ssm register-task-with-maintenance-window \
    --targets "Key=WindowTargetIds,Values=350d44e6-28cc-44e2-951f-4b2c9EXAMPLE" \
    --task-arn "AWS-RunShellScript" \
    --service-role-arn "arn:aws:iam::111222333444:role/MaintenanceWindowsRole" \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --task-type "RUN_COMMAND" \
    --task-parameters  "{\"commands\":{\"Values\":[\"df\"]}}" \
    --max-concurrency 1 \
    --max-errors 1 \
    --priority 10
```

Saída:

```
{
    "WindowTaskId":"33344444-5555-6666-7777-88888888"
}
```

Para obter mais informações, consulte Registrar uma tarefa com a janela de manutenção (AWS CLI) no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte RegisterTaskWithMaintenanceWindowem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra uma tarefa com uma janela de manutenção usando um ID de instância. A saída é o ID da tarefa.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Saída:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemplo 2: esse exemplo registra uma tarefa com uma janela de manutenção usando um ID de destino. A saída é o ID da tarefa.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -
TaskType "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Saída:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemplo 3: esse exemplo cria um objeto de parâmetro para o documento de comandos de execução **`AWS-RunPowerShellScript`** e cria uma tarefa com uma janela de manutenção determinada usando o ID de destino. A saída devolvida é o ID da tarefa.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target =
 @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Saída:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Exemplo 4: Este exemplo registra uma tarefa do AWS Systems Manager Automation usando um documento chamado**`Create-Snapshots`**.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
```

```
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- Para obter detalhes da API, consulte RegisterTaskWithMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma tarefa com uma janela de manutenção usando um ID de instância. A saída é o ID da tarefa.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Saída:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemplo 2: esse exemplo registra uma tarefa com uma janela de manutenção usando um ID de destino. A saída é o ID da tarefa.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
```

```
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -
TaskType "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Saída:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemplo 3: esse exemplo cria um objeto de parâmetro para o documento de comandos de execução **`AWS-RunPowerShellScript`** e cria uma tarefa com uma janela de manutenção determinada usando o ID de destino. A saída devolvida é o ID da tarefa.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target =
 @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Saída:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Exemplo 4: Este exemplo registra uma tarefa do AWS Systems Manager Automation usando um documento chamado**`Create-Snapshots`**.

```
$automationParameters = @{}
```

```
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- Para obter detalhes da API, consulte RegisterTaskWithMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RemoveTagsFromResource`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RemoveTagsFromResource`.

CLI

AWS CLI

Remover uma tag de uma lista de referência de patches

O exemplo de `remove-tags-from-resource` a seguir remove duas tags de uma lista de referência de patches.

```
aws ssm remove-tags-from-resource \
    --resource-type "PatchBaseline" \
    --resource-id "pb-0123456789abcdef0" \
    --tag-keys "Region"
```

Este comando não produz saída.

Para obter mais informações, consulte AWS Recursos de marcação na Referência AWS geral.

- Para obter detalhes da API, consulte RemoveTagsFromResourceem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove uma tag de uma janela de manutenção. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- Para obter detalhes da API, consulte RemoveTagsFromResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove uma tag de uma janela de manutenção. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- Para obter detalhes da API, consulte RemoveTagsFromResourceem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`SendCommand`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SendCommand`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Exemplo 1: executar um comando em uma ou mais instâncias remotas

O exemplo de `send-command` a seguir executa um comando `echo` em uma instância de destino.

```
aws ssm send-command \
    --document-name "AWS-RunShellScript" \
    --parameters 'commands=["echo HelloWorld"]' \
    --targets "Key=instanceids,Values=i-1234567890abcdef0" \
    --comment "echo HelloWorld"
```

Saída:

```
{
    "Command": {
        "CommandId": "92853adf-ba41-4cd6-9a88-142d1EXAMPLE",
        "DocumentName": "AWS-RunShellScript",
        "DocumentVersion": "",
        "Comment": "echo HelloWorld",
        "ExpiresAfter": 1550181014.717,
        "Parameters": {
            "commands": [
                "echo HelloWorld"
            ]
        },
        "InstanceIds": [
            "i-0f00f008a2dcbefe2"
        ],
        "Targets": [],
        "RequestedDateTime": 1550173814.717,
        "Status": "Pending",
        "StatusDetails": "Pending",
        "OutputS3BucketName": "",
        "OutputS3KeyPrefix": "",
        "MaxConcurrency": "50",
        "MaxErrors": "0",
        "TargetCount": 1,
        "CompletedCount": 0,
        "ErrorCount": 0,
        "DeliveryTimedOutCount": 0,
        "ServiceRole": "",
        "NotificationConfig": {
            "NotificationArn": "",
            "NotificationEvents": [],
            "NotificationType": ""
        },
        "CloudWatchOutputConfig": {
            "CloudWatchLogGroupName": "",
```

```
            "CloudWatchOutputEnabled": false
        }
    }
}
```

Para obter mais informações, consulte Executar comandos usando o Systems Manager Run Command no Guia do usuário do AWS Systems Manager.

Exemplo 2: obter informações de IP sobre uma instância

O exemplo de `send-command` a seguir retorna as informações de IP`sobre uma instância.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig"
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Executar comandos usando o Systems Manager Run Command no Guia do usuário do AWS Systems Manager.

Exemplo 3: executar um comando em instâncias com tags específicas

O exemplo de `send-command` a seguir executa um comando em instâncias que têm a chave de tag "ENV" e o valor "Dev".

```
aws ssm send-command \
    --targets "Key=tag:ENV,Values=Dev" \
    --document-name "AWS-RunShellScript" \
    --parameters "commands=ifconfig"
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Executar comandos usando o Systems Manager Run Command no Guia do usuário do AWS Systems Manager.

Exemplo 4: executar um comando que envia notificações do SNS

O exemplo de `send-command` a seguir executa um comando que envia notificações do SNS para todos os eventos de notificação e o tipo de notificação `Command`.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig" \
    --service-role-arn "arn:aws:iam::123456789012:role/SNS_Role" \
    --notification-config "NotificationArn=arn:aws:sns:us-
east-1:123456789012:SNSTopicName,NotificationEvents=All,NotificationType=Command"
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Executar comandos usando o Systems Manager Run Command no Guia do usuário do AWS Systems Manager.

Exemplo 5: Para executar um comando que retorna para S3 e CloudWatch

O `send-command` exemplo a seguir executa um comando que envia detalhes do comando para um bucket do S3 e para um grupo de registros de CloudWatch registros.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig" \
    --output-s3-bucket-name "s3-bucket-name" \
    --output-s3-key-prefix "runcommand" \
    --cloud-watch-output-
config "CloudWatchOutputEnabled=true,CloudWatchLogGroupName=CWLGroupName"
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Executar comandos usando o Systems Manager Run Command no Guia do usuário do AWS Systems Manager.

Exemplo 6: executar comandos em várias instâncias com tags diferentes

O exemplo de `send-command` a seguir executa um comando em instâncias com duas chaves e valores de tag diferentes.

```
aws ssm send-command \
    --document-name "AWS-RunPowerShellScript" \
```

```
    --parameters commands=["echo helloWorld"] \
    --targets Key=tag:Env,Values=Dev Key=tag:Role,Values=WebServers
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Executar comandos usando o Systems Manager Run Command no Guia do usuário do AWS Systems Manager.

Exemplo 7: usar várias instâncias com a mesma chave de tag

O exemplo de `send-command` a seguir executa um comando em instâncias que têm a mesma chave de tag, mas com valores diferentes.

```
aws ssm send-command \
    --document-name "AWS-RunPowerShellScript" \
    --parameters commands=["echo helloWorld"] \
    --targets Key=tag:Env,Values=Dev,Test
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Executar comandos usando o Systems Manager Run Command no Guia do usuário do AWS Systems Manager.

Exemplo 8: executar um comando que usa um documento compartilhado

O exemplo de `send-command` a seguir executa um comando compartilhado em uma instância de destino.

```
aws ssm send-command \
    --document-name "arn:aws:ssm:us-east-1:123456789012:document/ExampleDocument"
 \
    --targets "Key=instanceids,Values=i-1234567890abcdef0"
```

Consulte uma amostra de saída no exemplo 1.

Para obter mais informações, consulte Usar documentos do SSM compartilhados no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte SendCommandem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture =
CompletableFuture.runAsync(() -> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus =
response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
```

```
                    System.out.println("The SSM document is not active. Status: "
+ documentStatus);
                    try {
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        documentActiveFuture.join();

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse>
invocationFuture = getAsyncClient().getCommandInvocation(invocationRequest);
```

```
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution
successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }
```

- Para obter detalhes da API, consulte SendCommanda Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SendCommandCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Send an SSM command to a managed node.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new SendCommandCommand({
        DocumentName: documentName,
      }),
    );
    console.log("Command sent successfully.");
    return { Success: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid document name?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte SendCommanda Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo executa um comando echo em uma instância de destino.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Saída:

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Exemplo 2: esse exemplo mostra como executar um comando que aceita parâmetros aninhados.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- Para obter detalhes da API, consulte SendCommandem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo executa um comando echo em uma instância de destino.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Saída:

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Exemplo 2: esse exemplo mostra como executar um comando que aceita parâmetros aninhados.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- Para obter detalhes da API, consulte SendCommandem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
 TimeoutSeconds=3600
            )
```

```
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a SendCommandReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_ssm->sendcommand(
        it_instanceids = it_instance_ids
        iv_documentname = iv_document_name
        iv_timeoutseconds = 3600 ).
    DATA(lo_command) = lo_result->get_command( ).
    IF lo_command IS BOUND.
      rv_command_id = lo_command->get_commandid( ).
      MESSAGE 'Command sent successfully.' TYPE 'I'.
    ENDIF.
  CATCH /aws1/cx_ssminvaliddocument.
    MESSAGE 'Invalid document.' TYPE 'I'.
  CATCH /aws1/cx_ssminvalidinstanceid.
    MESSAGE 'Invalid instance ID.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a SendCommandreferência da API AWS SDK for SAP ABAP.

Usar **`StartAutomationExecution`** com uma CLI

Os exemplos de código a seguir mostram como usar o `StartAutomationExecution`.

CLI

AWS CLI

Exemplo 1: executar um documento do Automation

O exemplo de `start-automation-execution` a seguir executa um documento do Automation.

```
aws ssm start-automation-execution \
    --document-name "AWS-UpdateLinuxAmi" \
    --parameters "AutomationAssumeRole=arn:aws:iam::123456789012:role/
SSMAutomationRole,SourceAmiId=ami-EXAMPLE,IamInstanceProfileName=EC2InstanceRole"
```

Saída:

```
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
}
```

Para obter mais informações, consulte Executar um fluxo de trabalho do Automation manualmente no Guia do usuário do AWS Systems Manager.

Exemplo 2: executar um documento do Automation compartilhado

O exemplo de `start-automation-execution` a seguir executa um documento do Automation compartilhado.

```
aws ssm start-automation-execution \
    --document-name "arn:aws:ssm:us-east-1:123456789012:document/ExampleDocument"
```

Saída:

```
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
}
```

Para obter mais informações, consulte Usar documentos do SSM compartilhados no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte StartAutomationExecutionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo executa um documento especificando um perfil do Automation, um ID de origem da AMI e um perfil de instância do Amazon EC2.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-
f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Saída:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Para obter detalhes da API, consulte StartAutomationExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo executa um documento especificando um perfil do Automation, um ID de origem da AMI e um perfil de instância do Amazon EC2.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-
f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Saída:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Para obter detalhes da API, consulte StartAutomationExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`StartSession`** com uma CLI

Os exemplos de código a seguir mostram como usar o `StartSession`.

CLI

AWS CLI

Exemplo 1: Para iniciar uma sessão do Session Manager

Este exemplo de `start-session` estabelece uma conexão com uma instância para uma sessão do Session Manager. Observe que esse comando interativo exige que o plug-in do Session Manager seja instalado na máquina-cliente que faz a chamada.

```
aws ssm start-session \
    --target "i-1234567890abcdef0"
```

Saída:

```
Starting session with SessionId: Jane-Roe-07a16060613c408b5
```

Exemplo 2: Para iniciar uma sessão do Session Manager usando SSH

Este exemplo de `start-session` estabelece uma conexão com uma instância para uma sessão do Session Manager usando SSH. Observe que esse comando interativo exige que o plug-in do Session Manager seja instalado na máquina-cliente que faz a chamada e que o comando use o usuário padrão na instância, como `ec2-user` nas instâncias do EC2 para Linux.

```
ssh -i /path/my-key-pair.pem ec2-user@i-02573cafcfEXAMPLE
```

Saída:

```
Starting session with SessionId: ec2-user-07a16060613c408b5
```

Para obter mais informações, consulte Iniciar uma sessão e instalar o plug-in do gerenciador de sessões para a AWS CLI no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte StartSessionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Este exemplo inicia uma conexão com um destino para uma sessão do Session Manager, habilitando o encaminhamento de portas.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' }
```

Saída:

```
SessionId    StreamUrl
----------    ----------
random-id0    wss://ssmmessages.amazonaws.com/v1/data-channel/random-id
```

- Para obter detalhes da API, consulte StartSessionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo inicia uma conexão com um destino para uma sessão do Session Manager, habilitando o encaminhamento de portas. Observação: você deverá adicionar o parâmetro Region, se ainda não estiver preparado para usar o AWSCredentials.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' } -Region 'us-west-1'
```

Saída:

```
Starting session with SessionId: testUser-xi4glew849asyeryde34u4dfsdfy
Port 8080 opened for sessionId testUser-xi4glew849asyeryde34u4dfsdfy.
Waiting for connections...
```

Exemplo 2: esse exemplo cria uma sessão interativa com uma instância especificada para uma sessão do Session Manager.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1'
```

Saída:

```
Starting session with SessionId : testUser-xi4glew849asyeryde34u4dfsdfy
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.



Install the latest PowerShell for new features and improvements!


PS C:\Windows\system32> whoami
ec2amaz-fnsdrwv\ec2-test-user
PS C:\Windows\system32>
```

Exemplo 3: Este exemplo cria uma sessão sem se conectar a ela e retorna as TokenValue propriedades SessionId StreamUrl,, e necessárias para se conectar à sessão.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1' -
DisablePluginInvocation
```

Saída:

```
SessionId        : testUser-xi4glew849asyeryde34u4dfsdfy
StreamUrl        : {StreamUrl value redacted}
TokenValue       : {Token value redacted}
ContentLength    : 1207
HttpStatusCode   : OK
```

- Para obter detalhes da API, consulte StartSessionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`StopAutomationExecution`** com uma CLI

Os exemplos de código a seguir mostram como usar o `StopAutomationExecution`.

CLI

AWS CLI

Para interromper a execução de uma automação

O exemplo de `stop-automation-execution` a seguir interrompe um documento do Automation.

```
aws ssm stop-automation-execution
    --automation-execution-id "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
```

Este comando não produz saída.

Para obter mais informações, consulte Executar um fluxo de trabalho do Automation manualmente no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte StopAutomationExecutionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo interrompe uma execução do Automation. Não haverá saída se o comando for bem-sucedido.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- Para obter detalhes da API, consulte StopAutomationExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo interrompe uma execução do Automation. Não haverá saída se o comando for bem-sucedido.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- Para obter detalhes da API, consulte StopAutomationExecutionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateAssociation`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateAssociation`.

CLI

AWS CLI

Exemplo 1: atualizar uma associação de documentos

O exemplo de `update-association` a seguir atualiza uma associação com uma nova versão de documento.

```
aws ssm update-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --document-version "\$LATEST"
```

Saída:

```
{
    "AssociationDescription": {
        "Name": "AWS-UpdateSSMAgent",
        "AssociationVersion": "2",
        "Date": 1550508093.293,
        "LastUpdateAssociationDate": 1550508106.596,
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "DocumentVersion": "$LATEST",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "tag:Name",
                "Values": [
                    "Linux"
                ]
            }
        ],
```

```
        "LastExecutionDate": 1550508094.879,
        "LastSuccessfulExecutionDate": 1550508094.879
    }
}
```

Para obter mais informações, consulte Editar e criar uma nova versão de uma associação no Guia do usuário do AWS Systems Manager.

Exemplo 2: atualizar a expressão de programação de uma associação

O exemplo de `update-association` a seguir atualiza a expressão de programação para a associação especificada.

```
aws ssm update-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --schedule-expression "cron(0 0 0/4 1/1 * ? *)"
```

Saída:

```
{
    "AssociationDescription": {
        "Name": "AWS-HelloWorld",
        "AssociationVersion": "2",
        "Date": "2021-02-08T13:54:19.203000-08:00",
        "LastUpdateAssociationDate": "2021-06-29T11:51:07.933000-07:00",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "DocumentVersion": "$DEFAULT",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "aws:NoOpAutomationTag",
                "Values": [
                    "AWS-NoOpAutomationTarget-Value"
                ]
            }
        ],
        "ScheduleExpression": "cron(0 0 0/4 1/1 * ? *)",
        "LastExecutionDate": "2021-06-26T19:00:48.110000-07:00",
        "ApplyOnlyAtCronInterval": false
```

```
    }
}
```

Para obter mais informações, consulte Editar e criar uma nova versão de uma associação no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte UpdateAssociationem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza uma associação com uma nova versão de documento.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Para obter detalhes da API, consulte UpdateAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza uma associação com uma nova versão de documento.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
```

```
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Para obter detalhes da API, consulte UpdateAssociationem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateAssociationStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateAssociationStatus`.

CLI

AWS CLI

Para atualizar o status da associação

O exemplo de `update-association-status` a seguir atualiza o status da associação entre uma instância e um documento.

```
aws ssm update-association-status \
    --name "AWS-UpdateSSMAgent" \
    --instance-id "i-1234567890abcdef0" \
    --association-
status "Date=1424421071.939,Name=Pending,Message=temp_status_change,AdditionalInfo=Additi
Config-Needed"
```

Saída:

```
{
    "AssociationDescription": {
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-1234567890abcdef0",
        "AssociationVersion": "1",
        "Date": 1550507529.604,
        "LastUpdateAssociationDate": 1550507806.974,
        "Status": {
            "Date": 1424421071.0,
            "Name": "Pending",
```

```
            "Message": "temp_status_change",
            "AdditionalInfo": "Additional-Config-Needed"
        },
        "Overview": {
            "Status": "Success",
            "AssociationStatusAggregatedCount": {
                "Success": 1
            }
        },
        "DocumentVersion": "$DEFAULT",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "InstanceIds",
                "Values": [
                    "i-1234567890abcdef0"
                ]
            }
        ],
        "LastExecutionDate": 1550507808.0,
        "LastSuccessfulExecutionDate": 1550507808.0
    }
}
```

Para obter mais informações, consulte Trabalhar com associações no Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte UpdateAssociationStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza o status da associação entre uma instância e um documento de configuração.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Saída:

```
Name                   : AWS-UpdateSSMAgent
InstanceId             : i-0000293ffd8c57862
Date                   : 2/23/2017 6:55:22 PM
Status.Name            : Pending
Status.Date            : 2/20/2015 8:31:11 AM
Status.Message         : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Para obter detalhes da API, consulte UpdateAssociationStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza o status da associação entre uma instância e um documento de configuração.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Saída:

```
Name                   : AWS-UpdateSSMAgent
InstanceId             : i-0000293ffd8c57862
Date                   : 2/23/2017 6:55:22 PM
Status.Name            : Pending
Status.Date            : 2/20/2015 8:31:11 AM
Status.Message         : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Para obter detalhes da API, consulte UpdateAssociationStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateDocument`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateDocument`.

CLI

AWS CLI

Para criar uma nova versão de um documento

O exemplo de `update-document` a seguir cria uma nova versão de um documento quando executado em um computador Windows. O documento especificado por `--document` deve estar em formato JSON. Observe que `file://` deve ser referenciado seguido pelo caminho do arquivo de conteúdo. Em função disso do $ no início do parâmetro `--document-version`, o valor deve ser colocado entre aspas duplas no Windows. No Linux, no macOS ou em um PowerShell prompt, você deve colocar o valor entre aspas simples.

Versão do Windows:

```
aws ssm update-document \
    --name "RunShellScript" \
    --content "file://RunShellScript.json" \
    --document-version "$LATEST"
```

Linux/Mac versão:

```
aws ssm update-document \
    --name "RunShellScript" \
    --content "file://RunShellScript.json" \
    --document-version '$LATEST'
```

Saída:

```
{
  "DocumentDescription": {
      "Status": "Updating",
      "Hash": "f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b",
      "Name": "RunShellScript",
      "Parameters": [
          {
              "Type": "StringList",
              "Name": "commands",
              "Description": "(Required) Specify a shell script or a command to
 run."
          }
```

```
        ],
        "DocumentType": "Command",
        "PlatformTypes": [
            "Linux"
        ],
        "DocumentVersion": "2",
        "HashType": "Sha256",
        "CreatedDate": 1487899655.152,
        "Owner": "809632081692",
        "SchemaVersion": "2.0",
        "DefaultVersion": "1",
        "LatestVersion": "2",
        "Description": "Run an updated script"
    }
}
```

- Para obter detalhes da API, consulte UpdateDocumentem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: isso cria uma nova versão de um documento com o conteúdo atualizado do arquivo json que você especificar. O documento deve estar em formato JSON. Você pode obter a versão do documento com o cmdlet Get-SSMDocumentVersionList "".

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-
Content -Raw "c:\temp\RunShellScript.json")
```

Saída:

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
```

```
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- Para obter detalhes da API, consulte UpdateDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: isso cria uma nova versão de um documento com o conteúdo atualizado do arquivo json que você especificar. O documento deve estar em formato JSON. Você pode obter a versão do documento com o cmdlet Get-SSMDocumentVersionList "".

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-
Content -Raw "c:\temp\RunShellScript.json")
```

Saída:

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- Para obter detalhes da API, consulte UpdateDocumentem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`UpdateDocumentDefaultVersion`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateDocumentDefaultVersion`.

CLI

AWS CLI

Para atualizar a versão padrão de um documento

O exemplo de `update-document-default-version` a seguir atualiza a versão padrão de um documento do Systems Manager.

```
aws ssm update-document-default-version \
    --name "Example" \
    --document-version "2"
```

Saída:

```
{
    "Description": {
        "Name": "Example",
        "DefaultVersion": "2"
    }
}
```

Para obter mais informações, consulte Escrever conteúdo de documentos do SSM no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte UpdateDocumentDefaultVersionem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza a versão padrão de um documento. Você pode obter as versões disponíveis do documento com o cmdlet Get-SSMDocumentVersionList "”.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Saída:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- Para obter detalhes da API, consulte UpdateDocumentDefaultVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza a versão padrão de um documento. Você pode obter as versões disponíveis do documento com o cmdlet Get-SSMDocumentVersionList "".

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Saída:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- Para obter detalhes da API, consulte UpdateDocumentDefaultVersionem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`UpdateMaintenanceWindow`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateMaintenanceWindow`.

CLI

AWS CLI

Exemplo 1: atualizar uma janela de manutenção

O exemplo de `update-maintenance-window` a seguir atualiza o nome de uma janela de manutenção.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
```

```
    --name "My-Renamed-MW"
```

Saída:

```
{
    "Cutoff": 1,
    "Name": "My-Renamed-MW",
    "Schedule": "cron(0 16 ? * TUE *)",
    "Enabled": true,
    "AllowUnassociatedTargets": true,
    "WindowId": "mw-1a2b3c4d5e6f7g8h9",
    "Duration": 4
}
```

Exemplo 2: desabilitar uma janela de manutenção

O exemplo de `update-maintenance-window` a seguir desabilita uma janela de manutenção.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
    --no-enabled
```

Exemplo 3: habilitar uma janela de manutenção

O exemplo de `update-maintenance-window` a seguir habilita uma janela de manutenção.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
    --enabled
```

Para obter mais informações, consulte Update a Maintenance Window (AWS CLI) no AWS Systems Manager User Guide.

- Para obter detalhes da API, consulte UpdateMaintenanceWindowem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates an SSM maintenance window asynchronously.
    *
    * @param id The ID of the maintenance window to update.
    * @param name The new name for the maintenance window.
    * <p>
    * This method initiates an asynchronous request to update an SSM maintenance
window.
    * If the request is successful, it prints a success message.
    * If an exception occurs, it handles the error appropriately.
    */
   public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
       UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
               .windowId(id)
               .allowUnassociatedTargets(true)
               .duration(24)
               .enabled(true)
               .name(name)
               .schedule("cron(0 0 ? * MON *)")
               .build();

       CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
       future.whenComplete((response, ex) -> {
           if (response != null) {
               System.out.println("The SSM maintenance window was successfully
updated");
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
```

```
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }
```

- Para obter detalhes da API, consulte UpdateMaintenanceWindowa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { UpdateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Update an SSM maintenance window.
 * @param {{ windowId: string, allowUnassociatedTargets?: boolean, duration?:
 number, enabled?: boolean, name?: string, schedule?: string }}
 */
export const main = async ({
  windowId,
  allowUnassociatedTargets = undefined, //Allow the maintenance window to run on
 managed nodes, even if you haven't registered those nodes as targets.
  duration = undefined, //The duration of the maintenance window in hours.
  enabled = undefined,
  name = undefined,
  schedule = undefined, //The schedule of the maintenance window in the form of a
 cron or rate expression.
}) => {
```

```
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new UpdateMaintenanceWindowCommand({
        WindowId: windowId,
        AllowUnassociatedTargets: allowUnassociatedTargets,
        Duration: duration,
        Enabled: enabled,
        Name: name,
        Schedule: schedule,
      }),
    );
    console.log("Maintenance window updated.");
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Are these values correct?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte UpdateMaintenanceWindowa Referência AWS SDK para JavaScript da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza o nome de uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-
MW"
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
```

```
Name                      : My-Renamed-MW
Schedule                  : cron(0 */30 * * * ? *)
WindowId                  : mw-03eb9db42890fb82d
```

Exemplo 2: esse exemplo habilita uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Exemplo 3: esse exemplo desabilita uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para obter detalhes da API, consulte UpdateMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza o nome de uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-
MW"
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Exemplo 2: esse exemplo habilita uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Exemplo 3: esse exemplo desabilita uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para obter detalhes da API, consulte UpdateMaintenanceWindowem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def update(
        self, name, enabled, schedule, duration, cutoff,
 allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on
 managed nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
 managed nodes, even
```

```
                                                if you haven't registered those nodes
as targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to update maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateMaintenanceWindowReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ssm->updatemaintenancewindow(
        iv_windowid = iv_window_id
        iv_name = iv_name
        iv_enabled = iv_enabled
        iv_schedule = iv_schedule
        iv_duration = iv_duration
        iv_cutoff = iv_cutoff
        iv_allowunassociatedtargets = iv_allow_unassociated_targets ).
    MESSAGE 'Maintenance window updated.' TYPE 'I'.
  CATCH /aws1/cx_ssmdoesnotexistex.
    MESSAGE 'Maintenance window does not exist.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateMaintenanceWindowreferência da API AWS SDK for SAP ABAP.

Usar **`UpdateManagedInstanceRole`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdateManagedInstanceRole`.

CLI

AWS CLI

Para atualizar o perfil do IAM de uma instância gerenciada

O exemplo de `update-managed-instance-role` a seguir atualiza o perfil da instância do IAM de uma instância gerenciada.

```
aws ssm update-managed-instance-role \
    --instance-id "mi-08ab247cdfEXAMPLE" \
    --iam-role "ExampleRole"
```

Este comando não produz saída.

Para obter mais informações, consulte Etapa 4: criar um perfil de instância do IAM para o Systems Manager no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte UpdateManagedInstanceRoleem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo atualiza o perfil de uma instância gerenciada. Não haverá saída se o comando for bem-sucedido.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- Para obter detalhes da API, consulte UpdateManagedInstanceRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza o perfil de uma instância gerenciada. Não haverá saída se o comando for bem-sucedido.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- Para obter detalhes da API, consulte UpdateManagedInstanceRoleem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Use **`UpdateOpsItem`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateOpsItem`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conheça os conceitos básicos

CLI

AWS CLI

Para atualizar um OpsItem

O `update-ops-item` exemplo a seguir atualiza a descrição, a prioridade e a categoria de um OpsItem. Além disso, o comando especifica um tópico do SNS para o qual as notificações são enviadas quando isso OpsItem é editado ou alterado.

```
aws ssm update-ops-item \
    --ops-item-id "oi-287b5EXAMPLE" \
    --description "Primary OpsItem for failover event 2020-01-01-fh398yf" \
    --priority 2 \
    --category "Security" \
    --notifications "Arn=arn:aws:sns:us-east-2:111222333444:my-us-east-2-topic"
```

Saída:

```
This command produces no output.
```

Para obter mais informações, consulte Trabalhando com OpsItems no Guia do Usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte UpdateOpsItemem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Para obter detalhes da API, consulte UpdateOpsItema Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { UpdateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Update an SSM OpsItem.
 * @param {{ opsItemId: string, status?: OpsItemStatus }}
 */
export const main = async ({
  opsItemId,
  status = undefined, // The OpsItem status. Status can be Open, In Progress, or
 Resolved
}) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new UpdateOpsItemCommand({
        OpsItemId: opsItemId,
        Status: status,
      }),
    );
    console.log("Ops item updated.");
    return { Success: true };
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "OpsItemLimitExceededException"
    ) {
      console.warn(
```

```
        `Couldn't create ops item because you have exceeded your open OpsItem
 limit. ${caught.message}.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte UpdateOpsItema Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)
```

```
    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
            args["Description"] = description
        if status is not None:
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateOpsItemReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ssm->updateopsitem(
            iv_opsitemid = iv_ops_item_id
            iv_title = iv_title
            iv_description = iv_description
            iv_status = iv_status ).
        MESSAGE 'OpsItem updated.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsitemnotfoundex.
        MESSAGE 'OpsItem not found.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsiteminvparamex.
        MESSAGE 'Invalid OpsItem parameter.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateOpsItemreferência da API AWS SDK for SAP ABAP.

Usar **`UpdatePatchBaseline`** com uma CLI

Os exemplos de código a seguir mostram como usar o `UpdatePatchBaseline`.

CLI

AWS CLI

Exemplo 1: atualizar uma lista de referência de patches

O exemplo de `update-patch-baseline` a seguir adiciona os dois patches especificados como rejeitados e um patch como aprovado à lista de referência de patches especificada.

```
aws ssm update-patch-baseline \
        --baseline-id "pb-0123456789abcdef0" \
        --rejected-patches "KB2032276" "MS10-048" \
        --approved-patches "KB2124261"
```

Saída:

```
{
    "BaselineId": "pb-0123456789abcdef0",
    "Name": "WindowsPatching",
    "OperatingSystem": "WINDOWS",
    "GlobalFilters": {
```

```
        "PatchFilters": []
    },
    "ApprovalRules": {
        "PatchRules": [
            {
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "WindowsServer2016"
                            ]
                        }
                    ]
                },
                "ComplianceLevel": "CRITICAL",
                "ApproveAfterDays": 0,
                "EnableNonSecurity": false
            }
        ]
    },
    "ApprovedPatches": [
        "KB2124261"
    ],
    "ApprovedPatchesComplianceLevel": "UNSPECIFIED",
    "ApprovedPatchesEnableNonSecurity": false,
    "RejectedPatches": [
        "KB2032276",
        "MS10-048"
    ],
    "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
    "CreatedDate": 1550244180.465,
    "ModifiedDate": 1550244180.465,
    "Description": "Patches for Windows Servers",
    "Sources": []
}
```

Exemplo 2: renomear uma lista de referência de patches

O exemplo de `update-patch-baseline` a seguir renomeia lista de referência de patches especificada.

```
aws ssm update-patch-baseline \
```

```
    --baseline-id "pb-0713accee01234567" \
    --name "Windows-Server-2012-R2-Important-and-Critical-Security-Updates"
```

Para obter mais informações, consulte Atualizar ou excluir uma linha de base de patch < https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-baseline-update-or-delete.html >`__ no Guia do usuário do AWS Systems Manager.

- Para obter detalhes da API, consulte UpdatePatchBaselineem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona dois patches como rejeitados e um patch como aprovado a uma lista de referência de patches existente.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Saída:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- Para obter detalhes da API, consulte UpdatePatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona dois patches como rejeitados e um patch como aprovado a uma lista de referência de patches existente.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Saída:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- Para obter detalhes da API, consulte UpdatePatchBaselineem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para o Systems Manager usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Systems Manager com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções no Systems Manager ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Como criar um serviço de contêiner para instâncias de máquinas virtuais
- Introdução às máquinas virtuais
- Como configurar o gerenciamento de sistemas

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
```

```
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
```

```
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
 "$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
 $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
```

```
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
```

```
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
 file for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
```

```
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}
```

```
# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
```

```
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

```
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
```

```
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")
```

```
    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
```

```
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
```

```
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
```

```
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table
```

```
    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "============================================"
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "============================================"
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
```

```
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
```

```
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetParameters
  - GetRole
  - ListContainerInstances
  - ListServices
  - ListTasks

- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Introdução às máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um par de chaves.
- Criar um grupo de segurança.
- Interromper e iniciar uma instância.
- Testar a persistência de um IP elástico.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EC2 Basics Tutorial Script - Revised
# This script demonstrates the basics of working with EC2 instances using AWS CLI
# Updated to use Amazon Linux 2023 and enhanced security settings

# Set up logging
LOG_FILE="ec2_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
# Function to log messages
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

# Function to handle errors
handle_error() {
  log "ERROR: $1"
  log "Cleaning up resources..."
  cleanup
  exit 1
}

# Function to clean up resources
cleanup() {
  log "Resources created:"

  if [ -n "$ASSOCIATION_ID" ]; then
    log "- Elastic IP Association: $ASSOCIATION_ID"
  fi

  if [ -n "$ALLOCATION_ID" ]; then
    log "- Elastic IP Allocation: $ALLOCATION_ID (IP: $ELASTIC_IP)"
  fi

  if [ -n "$INSTANCE_ID" ]; then
    log "- EC2 Instance: $INSTANCE_ID"
  fi

  if [ -n "$SECURITY_GROUP_ID" ]; then
    log "- Security Group: $SECURITY_GROUP_ID"
  fi

  if [ -n "$KEY_NAME" ]; then
    log "- Key Pair: $KEY_NAME (File: $KEY_FILE)"
  fi

  REPLY=y

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    log "Starting cleanup..."

    # Track cleanup failures
    CLEANUP_FAILURES=0
```

```
    # Disassociate Elastic IP if it exists
    if [ -n "$ASSOCIATION_ID" ]; then
      log "Disassociating Elastic IP..."
      if ! aws ec2 disassociate-address --association-id "$ASSOCIATION_ID"; then
        log "Failed to disassociate Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Release Elastic IP if it exists
    if [ -n "$ALLOCATION_ID" ]; then
      log "Releasing Elastic IP..."
      if ! aws ec2 release-address --allocation-id "$ALLOCATION_ID"; then
        log "Failed to release Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Terminate instance if it exists
    if [ -n "$INSTANCE_ID" ]; then
      log "Terminating instance $INSTANCE_ID..."
      if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" > /dev/null;
then
        log "Failed to terminate instance"
        ((CLEANUP_FAILURES++))
      else
        log "Waiting for instance to terminate..."
        if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"; then
          log "Failed while waiting for instance to terminate"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Delete security group if it exists
    if [ -n "$SECURITY_GROUP_ID" ]; then
      log "Deleting security group..."
      if ! aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"; then
        log "Failed to delete security group"
        ((CLEANUP_FAILURES++))
      fi
    fi
```

```
    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ]; then
      log "Deleting key pair..."
      if ! aws ec2 delete-key-pair --key-name "$KEY_NAME"; then
        log "Failed to delete key pair"
        ((CLEANUP_FAILURES++))
      fi

      # Remove key file
      if [ -f "$KEY_FILE" ]; then
        log "Removing key file..."
        if ! rm -f "$KEY_FILE"; then
          log "Failed to remove key file"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Report cleanup status
    if [ $CLEANUP_FAILURES -eq 0 ]; then
      log "Cleanup completed successfully."
    else
      log "WARNING: Cleanup completed with $CLEANUP_FAILURES failures. Some
 resources may not have been deleted properly."
    fi
  else
    log "Resources were not deleted."
  fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
KEY_NAME="ec2-tutorial-key-$RANDOM_ID"
SG_NAME="ec2-tutorial-sg-$RANDOM_ID"

# Create a directory for the key file
KEY_DIR=$(mktemp -d)
KEY_FILE="$KEY_DIR/$KEY_NAME.pem"

log "Starting EC2 basics tutorial script"
log "Random identifier: $RANDOM_ID"
log "Key name: $KEY_NAME"
log "Security group name: $SG_NAME"
```

```
# Step 1: Create a key pair
log "Creating key pair..."
KEY_RESULT=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial'
 --output text)

if [ $? -ne 0 ] || [ -z "$KEY_RESULT" ]; then
  handle_error "Failed to create key pair"
fi

echo "$KEY_RESULT" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
log "Created key pair and saved to $KEY_FILE"

# Step 2: Create a security group
log "Creating security group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --group-name "$SG_NAME" \
  --description "Security group for EC2 tutorial" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query "GroupId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$SECURITY_GROUP_ID" ]; then
  handle_error "Failed to create security group"
fi

log "Created security group: $SECURITY_GROUP_ID"

# Get current public IP address for SSH access
MY_IP=$(curl -s http://checkip.amazonaws.com)
if [ $? -ne 0 ] || [ -z "$MY_IP" ]; then
  handle_error "Failed to get current IP address"
fi

log "Adding SSH ingress rule for IP $MY_IP..."
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr "$MY_IP/32" > /dev/null

if [ $? -ne 0 ]; then
  handle_error "Failed to add security group ingress rule"
```

```
fi

log "Added SSH ingress rule for IP $MY_IP"

# Step 3: Find an Amazon Linux 2023 AMI (updated from AL2)
log "Finding latest Amazon Linux 2023 AMI..."
AMI_ID=$(aws ssm get-parameters-by-path \
  --path "/aws/service/ami-amazon-linux-latest" \
  --query "Parameters[?contains(Name, 'al2023-ami-kernel-default-x86_64')].Value"
 \
  --output text | head -1)

if [ $? -ne 0 ] || [ -z "$AMI_ID" ]; then
  handle_error "Failed to find Amazon Linux 2023 AMI"
fi

log "Selected AMI: $AMI_ID"

# Get the architecture of the AMI
log "Getting AMI architecture..."
AMI_ARCH=$(aws ec2 describe-images \
  --image-ids "$AMI_ID" \
  --query "Images[0].Architecture" \
  --output text)

if [ $? -ne 0 ] || [ -z "$AMI_ARCH" ]; then
  handle_error "Failed to get AMI architecture"
fi

log "AMI architecture: $AMI_ARCH"

# Find a compatible instance type
log "Finding compatible instance type..."
# Directly use t2.micro for simplicity
INSTANCE_TYPE="t2.micro"
log "Using instance type: $INSTANCE_TYPE"

# Step 4: Launch an EC2 instance with enhanced security
log "Launching EC2 instance with IMDSv2 and encryption enabled..."
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
```

```
  --metadata-options "HttpTokens=required,HttpEndpoint=enabled" \
  --block-device-mappings "DeviceName=/dev/xvda,Ebs={Encrypted=true}" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]'
 'ResourceType=volume,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' \
  --count 1 \
  --query 'Instances[0].InstanceId' \
  --output text)

if [ $? -ne 0 ] || [ -z "$INSTANCE_ID" ]; then
  handle_error "Failed to launch EC2 instance"
fi

log "Launched instance $INSTANCE_ID. Waiting for it to start..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get instance details
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].
{ID:InstanceId,Type:InstanceType,State:State.Name,PublicIP:PublicIpAddress}' \
  --output json)

if [ $? -ne 0 ]; then
  handle_error "Failed to get instance details"
fi

log "Instance details: $INSTANCE_DETAILS"

# Get the public IP address
PUBLIC_IP=$(echo "$INSTANCE_DETAILS" | grep -oP '"PublicIP": "\K[^"]+')
if [ -z "$PUBLIC_IP" ]; then
  handle_error "Failed to get instance public IP"
fi

log "Instance public IP: $PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$PUBLIC_IP"
```

```
# Pause to allow user to connect if desired
sleep 2

# Step 6: Stop and Start the Instance
log "Stopping instance $INSTANCE_ID..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get the new public IP address
NEW_PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$NEW_PUBLIC_IP" ]; then
  handle_error "Failed to get new public IP"
fi

log "Instance restarted with new public IP: $NEW_PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$NEW_PUBLIC_IP"

# Step 7: Allocate and Associate an Elastic IP Address
log "Allocating Elastic IP address..."
ALLOCATION_RESULT=$(aws ec2 allocate-address \
```

```
  --domain vpc \
  --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query '[PublicIp,AllocationId]' \
  --output text)

if [ $? -ne 0 ] || [ -z "$ALLOCATION_RESULT" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

ELASTIC_IP=$(echo "$ALLOCATION_RESULT" | awk '{print $1}')
ALLOCATION_ID=$(echo "$ALLOCATION_RESULT" | awk '{print $2}')

log "Allocated Elastic IP: $ELASTIC_IP with ID: $ALLOCATION_ID"

log "Associating Elastic IP with instance..."
ASSOCIATION_ID=$(aws ec2 associate-address \
  --instance-id "$INSTANCE_ID" \
  --allocation-id "$ALLOCATION_ID" \
  --query "AssociationId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$ASSOCIATION_ID" ]; then
  handle_error "Failed to associate Elastic IP"
fi

log "Associated Elastic IP with instance. Association ID: $ASSOCIATION_ID"
log "To connect to your instance using the Elastic IP, run: ssh -i $KEY_FILE ec2-
user@$ELASTIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 8: Test the Elastic IP by Stopping and Starting the Instance
log "Stopping instance $INSTANCE_ID to test Elastic IP persistence..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
```

```
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Verify the Elastic IP is still associated
CURRENT_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$CURRENT_IP" ]; then
  handle_error "Failed to get current public IP"
fi

log "Current public IP address: $CURRENT_IP"
log "Elastic IP address: $ELASTIC_IP"

if [ "$CURRENT_IP" = "$ELASTIC_IP" ]; then
  log "Success! The Elastic IP is still associated with your instance."
else
  log "Something went wrong. The Elastic IP is not associated with your
 instance."
fi

log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$ELASTIC_IP"

# Step 9: Clean up resources
log "Tutorial completed successfully!"
cleanup

exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AllocateAddress
- AssociateAddress
- AuthorizeSecurityGroupIngress
- CreateKeyPair
- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstances
- DisassociateAddress
- GetParametersByPath
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- Wait

Como configurar o gerenciamento de sistemas

O exemplo de código a seguir mostra como:

- Criar permissões do IAM para o Systems Manager.
- Criar um perfil do IAM para o Systems Manager
- Configurar o Systems Manager.
- Verificar a configuração.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
```

```
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"
```

```
        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id
 --policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --
manager-arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/
null || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and
 region."
echo "It will create IAM policies and roles, then enable Systems Manager
 features."
echo ""
```

```
# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
         "ssm:GetPatchBaseline",
         "ssm:DescribeMaintenanceWindows",
         "ssm:DescribeMaintenanceWindowTasks"
```

```
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
```

```
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
```

```
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
```

```
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
```

```
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
      }
    ]
}
```

```
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
```

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
```

```
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --
policy-arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF
```

```
echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few
 minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-
arn $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration
 manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "==========================================="
```

```
echo "CREATED RESOURCES"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later." | tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy

- GetCallerIdentity
- GetConfigurationManager

Exemplos de código para o Amazon Textract usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Textract com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Textract: mais informações sobre o Amazon Textract.
- Referência da API Amazon Textract: detalhes sobre todas as ações disponíveis do Amazon Textract.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon Textract AWS SDKs
  - Ações para o Amazon Textract usando AWS SDKs
    - Use AnalyzeDocumentcom um AWS SDK ou CLI
    - Use DetectDocumentTextcom um AWS SDK ou CLI
    - Use GetDocumentAnalysiscom um AWS SDK ou CLI
    - Use StartDocumentAnalysiscom um AWS SDK ou CLI
    - Use StartDocumentTextDetectioncom um AWS SDK ou CLI
- Cenários para o Amazon Textract usando AWS SDKs
  - Criar uma aplicação de exploração do Amazon Textract

- Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
- Detecte entidades no texto extraído de uma imagem usando um AWS SDK
- Comece a usar a análise de documentos do Amazon Textract usando um AWS SDK
- Introdução à extração de texto de documentos

Exemplos básicos para o uso do Amazon Textract AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Textract com AWS SDKs.

Exemplos

- Ações para o Amazon Textract usando AWS SDKs
  - Use AnalyzeDocumentcom um AWS SDK ou CLI
  - Use DetectDocumentTextcom um AWS SDK ou CLI
  - Use GetDocumentAnalysiscom um AWS SDK ou CLI
  - Use StartDocumentAnalysiscom um AWS SDK ou CLI
  - Use StartDocumentTextDetectioncom um AWS SDK ou CLI

Ações para o Amazon Textract usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Textract com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Textract e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Textract usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Textract.

Exemplos

- Use AnalyzeDocumentcom um AWS SDK ou CLI
- Use DetectDocumentTextcom um AWS SDK ou CLI
- Use GetDocumentAnalysiscom um AWS SDK ou CLI

- Use StartDocumentAnalysiscom um AWS SDK ou CLI
- Use StartDocumentTextDetectioncom um AWS SDK ou CLI

Use **`AnalyzeDocument`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AnalyzeDocument`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Introdução à extração de texto de documentos

CLI

AWS CLI

Como analisar texto em um documento

O exemplo de `analyze-document` a seguir mostra como analisar texto em um documento.

Linux/macOS:

```
aws textract analyze-document \
    --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
    --feature-types '["TABLES","FORMS"]'
```

Windows:

```
aws textract analyze-document \
    --document "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document\"}}" \
    --feature-types "[\"TABLES\",\"FORMS\"]" \
    --region region-name
```

Saída:

```
{
    "Blocks": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
```

```
                    "Top": 0.0,
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
                ]
            },
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "87586964-d50d-43e2-ace5-8a890657b9a0",
                        "a1e72126-21d9-44f4-a8d6-5c385f9002ba",
                        "e889d012-8a6b-4d2e-b7cd-7a8b327d876a"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "c2227f12-b25d-4e1f-baea-1ee180d926b2"
        }
    ],
    "DocumentMetadata": {
        "Pages": 1
    }
}
```

Para obter mais informações, consulte Analyzing Document Text with Amazon Textract Guia do desenvolvedor do Amazon Textract

- Para obter detalhes da API, consulte AnalyzeDocumentem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
                    sourceDoc - The path where the document is located (must be
 an image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        analyzeDoc(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void analyzeDoc(TextractClient textractClient, String
 sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            List<FeatureType> featureTypes = new ArrayList<FeatureType>();
            featureTypes.add(FeatureType.FORMS);
            featureTypes.add(FeatureType.TABLES);

            AnalyzeDocumentRequest analyzeDocumentRequest =
 AnalyzeDocumentRequest.builder()
```

```
                    .featureTypes(featureTypes)
                    .document(myDoc)
                    .build();

            AnalyzeDocumentResponse analyzeDocument =
 textractClient.analyzeDocument(analyzeDocumentRequest);
            List<Block> docInfo = analyzeDocument.blocks();
            Iterator<Block> blockIterator = docInfo.iterator();

            while (blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is " +
 block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte AnalyzeDocumenta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
```

```
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def analyze_file(
        self, feature_types, *, document_file_name=None, document_bytes=None
    ):
        """
        Detects text and additional elements, such as forms or tables, in a local
image
        file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to
detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, "rb") as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={"Bytes": document_bytes}, FeatureTypes=feature_types
            )
            logger.info("Detected %s blocks.", len(response["Blocks"]))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a AnalyzeDocumentReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Detects text and additional elements, such as forms or tables,"
   "in a local image file or from in-memory byte data."
   "The image must be in PNG or JPG format."


   "Create ABAP objects for feature type."
   "Add TABLES to return information about the tables."
   "Add FORMS to return detected form data."
   "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).

   "Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).

   "Analyze document stored in Amazon S3."
   TRY.
       oo_result = lo_tex->analyzedocument(      "oo_result is returned for
testing purposes."
         io_document        = lo_document
```

```
          it_featuretypes    = lt_featuretypes ).
        LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
          IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
            MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
          ENDIF.
        ENDLOOP.
        MESSAGE 'Analyze document completed.' TYPE 'I'.
      CATCH /aws1/cx_texaccessdeniedex.
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
      CATCH /aws1/cx_texbaddocumentex.
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
      CATCH /aws1/cx_texdocumenttoolargeex.
        MESSAGE 'The document is too large.' TYPE 'E'.
      CATCH /aws1/cx_texhlquotaexceededex.
        MESSAGE 'Human loop quota exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.

      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AnalyzeDocumentreferência da API AWS SDK for SAP ABAP.

Use **`DetectDocumentText`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetectDocumentText`.

CLI

AWS CLI

Como detectar texto em um documento

O exemplo de `detect-document-text` a seguir mostra como detectar texto em um documento.

Linux/macOS:

```
aws textract detect-document-text \
    --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}'
```

Windows:

```
aws textract detect-document-text \
    --document "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document\"}}" \
    --region region-name
```

Saída:

```
{
    "Blocks": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Top": 0.0,
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
```

```
                ]
            },
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "896a9f10-9e70-4412-81ce-49ead73ed881",
                        "0da18623-dc4c-463d-a3d1-9ac050e9e720",
                        "167338d7-d38c-4760-91f1-79a8ec457bb2"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "21f0535e-60d5-4bc7-adf2-c05dd851fa25"
        },
        {
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "62490c26-37ea-49fa-8034-7a9ff9369c9c",
                        "1e4f3f21-05bd-4da9-ba10-15d01e66604c"
                    ]
                }
            ],
            "Confidence": 89.11581420898438,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33642634749412537,
                    "Top": 0.17169663310050964,
                    "Left": 0.13885067403316498,
                    "Height": 0.49159330129623413
                },
                "Polygon": [
                    {
                        "Y": 0.17169663310050964,
                        "X": 0.13885067403316498
                    },
                    {
                        "Y": 0.17169663310050964,
                        "X": 0.47527703642845154
                    },
                    {
                        "Y": 0.6632899641990662,
```

```
                        "X": 0.47527703642845154
                    },
                    {
                        "Y": 0.6632899641990662,
                        "X": 0.13885067403316498
                    }
                ]
            },
            "Text": "He llo,",
            "BlockType": "LINE",
            "Id": "896a9f10-9e70-4412-81ce-49ead73ed881"
        },
        {
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "19b28058-9516-4352-b929-64d7cef29daf"
                    ]
                }
            ],
            "Confidence": 85.5694351196289,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33182239532470703,
                    "Top": 0.23131252825260162,
                    "Left": 0.5091826915740967,
                    "Height": 0.3766750991344452
                },
                "Polygon": [
                    {
                        "Y": 0.23131252825260162,
                        "X": 0.5091826915740967
                    },
                    {
                        "Y": 0.23131252825260162,
                        "X": 0.8410050868988037
                    },
                    {
                        "Y": 0.607987642288208,
                        "X": 0.8410050868988037
                    },
                    {
                        "Y": 0.607987642288208,
```

```
                        "X": 0.5091826915740967
                    }
                ]
            },
            "Text": "worlc",
            "BlockType": "LINE",
            "Id": "0da18623-dc4c-463d-a3d1-9ac050e9e720"
        }
    ],
    "DocumentMetadata": {
        "Pages": 1
    }
}
```

Para obter mais informações, consulte Detecting Document Text with Amazon Textract Guia do desenvolvedor do Amazon Textract

- Para obter detalhes da API, consulte DetectDocumentTextem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Detecte texto de um documento de entrada.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
                    sourceDoc - The path where the document is located (must be
 an image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocText(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void detectDocText(TextractClient textractClient, String
 sourceDoc) {
        try {
```

```
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes.
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the Detect operation.
            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            List<Block> docInfo = textResponse.blocks();
            for (Block block : docInfo) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte texto de um documento localizado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
```

```
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains
 the document.\s

                    docName - The document name (must be an image, i.e.,
 book.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }
```

```
    public static void detectDocTextS3(TextractClient textractClient, String
 bucketName, String docName) {
        try {
            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            // Create a Document object and reference the s3Object instance.
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            for (Block block : textResponse.blocks()) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectDocumentTexta Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, "rb") as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.detect_document_text(
                Document={"Bytes": document_bytes}
            )
            logger.info("Detected %s blocks.", len(response["Blocks"]))
```

```
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DetectDocumentTextReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Detects text in the input document."
   "Amazon Textract can detect lines of text and the words that make up a line
of text."
   "The input document must be in one of the following image formats: JPEG, PNG,
PDF, or TIFF."

   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).
   "Analyze document stored in Amazon S3."
   TRY.
       oo_result = lo_tex->detectdocumenttext( io_document = lo_document ).
   "oo_result is returned for testing purposes."
       LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
         IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
           MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
```

```
          ENDIF.
        ENDLOOP.
        DATA(lo_metadata) = oo_result->get_documentmetadata( ).
        MESSAGE 'The number of pages in the document is ' && lo_metadata-
>ask_pages( ) TYPE 'I'.
        MESSAGE 'Detect document text completed.' TYPE 'I'.
      CATCH /aws1/cx_texaccessdeniedex.
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
      CATCH /aws1/cx_texbaddocumentex.
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
      CATCH /aws1/cx_texdocumenttoolargeex.
        MESSAGE 'The document is too large.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetectDocumentTextreferência da API AWS SDK for SAP ABAP.

Use **`GetDocumentAnalysis`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetDocumentAnalysis`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de análise de documentos

CLI

AWS CLI

Como obter os resultados de uma análise assíncrona de texto em um documento com várias páginas

O exemplo de `get-document-analysis` a seguir mostra como obter os resultados de uma análise assíncrona de texto em um documento com várias páginas.

```
aws textract get-document-analysis \
    --job-id df7cf32ebbd2a5de113535fcf4d921926a701b09b4e7d089f3aebadb41e0712b \
    --max-results 1000
```

Saída:

```
{
    "Blocks": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Top": 0.0,
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
```

```
                ]
            },
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "75966e64-81c2-4540-9649-d66ec341cd8f",
                        "bb099c24-8282-464c-a179-8a9fa0a057f0",
                        "5ebf522d-f9e4-4dc7-bfae-a288dc094595"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "247c28ee-b63d-4aeb-9af0-5f7ea8ba109e",
            "Page": 1
        }
    ],
    "NextToken": "cY1W3eTFvoB0cH7YrKVudI4Gb0H8J0xAYLo8xI/JunCIPWCthaKQ+07n/
ElyutsSy0+1VOImoTRmP1zw4P0RFtaeV9Bzhnfedpx1YqwB4xaGDA==",
    "DocumentMetadata": {
        "Pages": 1
    },
    "JobStatus": "SUCCEEDED"
}
```

Para obter mais informações, consulte Detecção e análise de texto em Multi-Page documentos no Amazon Textract Developers Guide

- Para obter detalhes da API, consulte GetDocumentAnalysisem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class TextractWrapper:
```

```
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                 detected in the image.
        """
        try:
            response = self.textract_client.get_document_analysis(JobId=job_id)
            job_status = response["JobStatus"]
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a GetDocumentAnalysisReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Gets the results for an Amazon Textract"
   "asynchronous operation that analyzes text in a document."
   TRY.
       oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
"oo_result is returned for testing purposes."
       WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
         IF sy-index = 10.
           EXIT.               "Maximum 300 seconds.
         ENDIF.
         WAIT UP TO 30 SECONDS.
         oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
       ENDWHILE.

       DATA(lt_blocks) = oo_result->get_blocks( ).
       LOOP AT lt_blocks INTO DATA(lo_block).
         IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
           MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
         ENDIF.
       ENDLOOP.
       MESSAGE 'Document analysis retrieved.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidjobidex.
       MESSAGE 'Job ID is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
     CATCH /aws1/cx_texinvalids3objectex.
```

```
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetDocumentAnalysisreferência da API AWS SDK for SAP ABAP.

Use **`StartDocumentAnalysis`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartDocumentAnalysis`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de análise de documentos

CLI

AWS CLI

Como começar a analisar texto em um documento com várias páginas

O exemplo de `start-document-analysis` a seguir mostra como iniciar a análise assíncrona de texto em um documento com várias páginas.

Linux/macOS:

```
aws textract start-document-analysis \
    --document-location '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
    --feature-types '["TABLES","FORMS"]' \
    --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"
```

Windows:

```
aws textract start-document-analysis \
    --document-location "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document
\"}}" \
    --feature-types "[\"TABLES\", \"FORMS\"]" \
```

```
    --region region-name \
    --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"
```

Saída:

```
{
    "JobId": "df7cf32ebbd2a5de113535fcf4d921926a701b09b4e7d089f3aebadb41e0712b"
}
```

Para obter mais informações, consulte Detecção e análise de texto em Multi-Page documentos no Amazon Textract Developers Guide

- Para obter detalhes da API, consulte StartDocumentAnalysisem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import
 software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import
 software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import
 software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains
 the document.\s
                    docName - The document name (must be an image, for example,
 book.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
        String status = getJobResults(textractClient, jobId);
        System.out.println("The job status is " + status);
        textractClient.close();
    }

    public static String startDocAnalysisS3(TextractClient textractClient, String
 bucketName, String docName) {
```

```
        try {
            List<FeatureType> myList = new ArrayList<>();
            myList.add(FeatureType.TABLES);
            myList.add(FeatureType.FORMS);

            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            DocumentLocation location = DocumentLocation.builder()
                    .s3Object(s3Object)
                    .build();

            StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                    .documentLocation(location)
                    .featureTypes(myList)
                    .build();

            StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

            // Get the job ID
            String jobId = response.jobId();
            return jobId;

        } catch (TextractException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    private static String getJobResults(TextractClient textractClient, String
jobId) {
        boolean finished = false;
        int index = 0;
        String status = "";

        try {
            while (!finished) {
                GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
```

```
                        .jobId(jobId)
                        .maxResults(1000)
                        .build();

                GetDocumentAnalysisResponse response =
 textractClient.getDocumentAnalysis(analysisRequest);
                status = response.jobStatus().toString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(index + " status is: " + status);
                    Thread.sleep(1000);
                }
                index++;
            }

            return status;

        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte StartDocumentAnalysisa Referência AWS SDK for Java 2.x da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Iniciar um trabalho assíncrono para analisar um documento.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def start_analysis_job(
        self,
        bucket_name,
        document_file_name,
        feature_types,
        sns_topic_arn,
        sns_role_arn,
    ):
        """
        Starts an asynchronous job to detect text and additional elements, such
 as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract
 publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
 image.
        :param document_file_name: The name of the document image stored in
 Amazon S3.
        :param feature_types: The types of additional document features to
 detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
 topic
                               where job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
 (IAM)
                              role that can be assumed by Textract and grants
 permission
```

```
                                to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    "S3Object": {"Bucket": bucket_name, "Name":
document_file_name}
                },
                NotificationChannel={
                    "SNSTopicArn": sns_topic_arn,
                    "RoleArn": sns_role_arn,
                },
                FeatureTypes=feature_types,
            )
            job_id = response["JobId"]
            logger.info(
                "Started text analysis job %s on %s.", job_id, document_file_name
            )
        except ClientError:
            logger.exception("Couldn't analyze text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- Para obter detalhes da API, consulte a StartDocumentAnalysisReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Starts the asynchronous analysis of an input document for relationships"
   "between detected items such as key-value pairs, tables, and selection
elements."

   "Create ABAP objects for feature type."
   "Add TABLES to return information about the tables."
   "Add FORMS to return detected form data."
   "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).
   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).
   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

   "Start async document analysis."
   TRY.
       oo_result = lo_tex->startdocumentanalysis(      "oo_result is returned
for testing purposes."
         io_documentlocation     = lo_documentlocation
         it_featuretypes         = lt_featuretypes ).
       DATA(lv_jobid) = oo_result->get_jobid( ).

       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
     CATCH /aws1/cx_texinvalids3objectex.
       MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
```

```
      CATCH /aws1/cx_texlimitexceededex.
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartDocumentAnalysisreferência da API AWS SDK for SAP ABAP.

Use **`StartDocumentTextDetection`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartDocumentTextDetection`.

CLI

AWS CLI

Como começar a detectar texto em um documento com várias páginas

O exemplo de `start-document-text-detection` a seguir mostra como iniciar a detecção assíncrona de texto em um documento com várias páginas.

Linux/macOS:

```
aws textract start-document-text-detection \
        --document-location '{"S3Object":{"Bucket":"bucket","Name":"document"}}'
 \
        --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleARN"
```

Windows:

```
aws textract start-document-text-detection \
    --document-location "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document
\"}}" \
    --region region-name \
    --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"
```

Saída:

```
{
    "JobId": "57849a3dc627d4df74123dca269d69f7b89329c870c65bb16c9fd63409d200b9"
}
```

Para obter mais informações, consulte Detecção e análise de texto em Multi-Page documentos no Amazon Textract Developers Guide

- Para obter detalhes da API, consulte StartDocumentTextDetectionem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Iniciar um trabalho assíncrono para detectar texto em um documento.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def start_detection_job(
        self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn
    ):
        """
```

```
        Starts an asynchronous job to detect text elements in an image stored in
an
        Amazon S3 bucket. Textract publishes a notification to the specified
Amazon SNS
        topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
image.
        :param document_file_name: The name of the document image stored in
Amazon S3.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
topic
                               where the job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
(IAM)
                              role that can be assumed by Textract and grants
permission
                              to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_text_detection(
                DocumentLocation={
                    "S3Object": {"Bucket": bucket_name, "Name":
document_file_name}
                },
                NotificationChannel={
                    "SNSTopicArn": sns_topic_arn,
                    "RoleArn": sns_role_arn,
                },
            )
            job_id = response["JobId"]
            logger.info(
                "Started text detection job %s on %s.", job_id,
document_file_name
            )
        except ClientError:
            logger.exception("Couldn't detect text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- Para obter detalhes da API, consulte a StartDocumentTextDetectionReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Starts the asynchronous detection of text in a document."
   "Amazon Textract can detect lines of text and the words that make up a line
of text."

   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).
   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).
   "Start document analysis."
   TRY.
       oo_result = lo_tex->startdocumenttextdetection( io_documentlocation =
lo_documentlocation ).
       DATA(lv_jobid) = oo_result->get_jobid( ).             "oo_result is
returned for testing purposes."
       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
```

```
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidkmskeyex.
        MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texlimitexceededex.
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartDocumentTextDetectionreferência da API AWS SDK for SAP ABAP.

Cenários para o Amazon Textract usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Textract com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon Textract ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação de exploração do Amazon Textract
- Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
- Detecte entidades no texto extraído de uma imagem usando um AWS SDK
- Comece a usar a análise de documentos do Amazon Textract usando um AWS SDK
- Introdução à extração de texto de documentos

Criar uma aplicação de exploração do Amazon Textract

Os exemplos de código a seguir mostram como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.

- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio

Os exemplos de código a seguir mostram como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

.NET

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.

- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado nas funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );
```

```
  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
```

```
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
```

```
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Detecte entidades no texto extraído de uma imagem usando um AWS SDK

O exemplo de código a seguir mostra como usar o Amazon Comprehend para detectar entidades em texto extraído pelo Amazon Textract de uma imagem armazenada no Amazon S3.

Python

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) em um notebook Jupyter para detectar entidades no texto extraído de uma imagem. Este exemplo usa o Amazon Textract para extrair texto de uma imagem armazenada no Amazon Simple Storage Service (Amazon S3) e no Amazon Comprehend para detectar entidades no texto extraído.

Este exemplo é um caderno Jupyter e deve ser executado em um ambiente que possa hospedar blocos de anotações. Para obter instruções sobre como executar o exemplo usando o Amazon SageMaker AI, consulte as instruções em TextractAndComprehendNotebook.ipynb.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Comece a usar a análise de documentos do Amazon Textract usando um AWS SDK

O exemplo de código a seguir mostra como:

- Iniciar análise assíncrona.
- Obter análise de documento.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"Create ABAP objects for feature type."
"Add TABLES to return information about the tables."
"Add FORMS to return detected form data."
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
  ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
  ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).
```

```
   "Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

   "Start document analysis."
   TRY.
       DATA(lo_start_result) = lo_tex->startdocumentanalysis(
         io_documentlocation     = lo_documentlocation
         it_featuretypes         = lt_featuretypes ).
       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
     CATCH /aws1/cx_texinvalids3objectex.
       MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texlimitexceededex.
       MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
     CATCH /aws1/cx_texprovthruputexcdex.
       MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
     CATCH /aws1/cx_texthrottlingex.
       MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
     CATCH /aws1/cx_texunsupporteddocex.
       MESSAGE 'The document is not supported.' TYPE 'E'.
   ENDTRY.

   "Get job ID from the output."
   DATA(lv_jobid) = lo_start_result->get_jobid( ).
```

```
    "Wait for job to complete."
    oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).     "
oo_result is returned for testing purposes. "
    WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
      IF sy-index = 10.
        EXIT.               "Maximum 300 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).
    ENDWHILE.

    DATA(lt_blocks) = oo_result->get_blocks( ).
    LOOP AT lt_blocks INTO DATA(lo_block).
      IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
        MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
      ENDIF.
    ENDLOOP.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - GetDocumentAnalysis
  - StartDocumentAnalysis

Introdução à extração de texto de documentos

O exemplo de código a seguir mostra como:

- Criar um bucket do S3.
- Fazer upload de um documento no S3.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon Textract Getting Started Tutorial"
echo "====================================================="
echo "This script will guide you through using Amazon Textract to analyze
 document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*/\1=***REDACTED***/g'
        cleanup_on_error
        exit 1
```

```
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to
 delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ];
 then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
```

```
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
 default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
```

```
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
 2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
 {"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
 50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
```

```
UPLOAD_OUTPUT=$(aws s3 cp "./$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
 2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ./$DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi
```

```
# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "====================================================="
echo "Analysis Summary"
echo "====================================================="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
 '{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
 echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
 echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
 echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
 echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
 echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
 | wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
 wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""
```

```
# Cleanup confirmation
echo ""
echo "==================================================="
echo "RESOURCES CREATED"
echo "==================================================="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

echo ""
echo "==================================================="
echo "Tutorial complete!"
echo "==================================================="
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
```

```
echo ""

trap - EXIT
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AnalyzeDocument
  - Cp
  - Help
  - Mb
  - Rb
  - Rm

Exemplos de código para o Amazon Transcribe usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Transcribe com AWS um kit de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Transcribe: mais informações sobre o Amazon Transcribe.
- Referência da API Amazon Transcribe: detalhes sobre todas as ações disponíveis do Amazon Transcribe.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o Amazon Transcribe usando AWS SDKs

- Ações para o Amazon Transcribe usando AWS SDKs
  - Use CreateVocabularycom um AWS SDK ou CLI
  - Use DeleteMedicalTranscriptionJobcom um AWS SDK ou CLI
  - Use DeleteTranscriptionJobcom um AWS SDK ou CLI
  - Use DeleteVocabularycom um AWS SDK ou CLI
  - Use GetTranscriptionJobcom um AWS SDK ou CLI
  - Use GetVocabularycom um AWS SDK ou CLI
  - Use ListMedicalTranscriptionJobscom um AWS SDK ou CLI
  - Use ListTranscriptionJobscom um AWS SDK ou CLI
  - Use ListVocabulariescom um AWS SDK ou CLI
  - Use StartMedicalTranscriptionJobcom um AWS SDK ou CLI
  - Use StartTranscriptionJobcom um AWS SDK ou CLI
  - Use UpdateVocabularycom um AWS SDK ou CLI
- Cenários para o Amazon Transcribe usando AWS SDKs
  - Criar uma aplicação de transmissão do Amazon Transcribe
  - Converta texto em fala e volte para texto usando um AWS SDK
  - Crie e refine um vocabulário personalizado do Amazon Transcribe usando um AWS SDK
  - Transcreva áudio e obtenha dados de trabalho com o Amazon Transcribe usando um AWS SDK

Exemplos básicos para o Amazon Transcribe usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Transcribe com AWS SDKs.

Exemplos

- Ações para o Amazon Transcribe usando AWS SDKs
  - Use CreateVocabularycom um AWS SDK ou CLI
  - Use DeleteMedicalTranscriptionJobcom um AWS SDK ou CLI
  - Use DeleteTranscriptionJobcom um AWS SDK ou CLI
  - Use DeleteVocabularycom um AWS SDK ou CLI
  - Use GetTranscriptionJobcom um AWS SDK ou CLI
  - Use GetVocabularycom um AWS SDK ou CLI

- Use ListMedicalTranscriptionJobscom um AWS SDK ou CLI
- Use ListTranscriptionJobscom um AWS SDK ou CLI
- Use ListVocabulariescom um AWS SDK ou CLI
- Use StartMedicalTranscriptionJobcom um AWS SDK ou CLI
- Use StartTranscriptionJobcom um AWS SDK ou CLI
- Use UpdateVocabularycom um AWS SDK ou CLI

Ações para o Amazon Transcribe usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Transcribe com SDKs. AWS Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Transcribe e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Transcribe usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência da API do Amazon Transcribe.

Exemplos

- Use CreateVocabularycom um AWS SDK ou CLI
- Use DeleteMedicalTranscriptionJobcom um AWS SDK ou CLI
- Use DeleteTranscriptionJobcom um AWS SDK ou CLI
- Use DeleteVocabularycom um AWS SDK ou CLI
- Use GetTranscriptionJobcom um AWS SDK ou CLI
- Use GetVocabularycom um AWS SDK ou CLI
- Use ListMedicalTranscriptionJobscom um AWS SDK ou CLI
- Use ListTranscriptionJobscom um AWS SDK ou CLI
- Use ListVocabulariescom um AWS SDK ou CLI
- Use StartMedicalTranscriptionJobcom um AWS SDK ou CLI
- Use StartTranscriptionJobcom um AWS SDK ou CLI
- Use UpdateVocabularycom um AWS SDK ou CLI

Use **`CreateVocabulary`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateVocabulary`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e refinar um vocabulário personalizado

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a custom vocabulary using a list of phrases. Custom vocabularies
    /// improve transcription accuracy for one or more specific words.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> CreateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.CreateVocabularyAsync(
            new CreateVocabularyRequest
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- Para obter detalhes da API, consulte CreateVocabularya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como criar um vocabulário personalizado

O exemplo de `create-vocabulary` a seguir cria um vocabulário personalizado. Para criar um vocabulário personalizado, você deve ter criado um arquivo de texto com todos os termos que deseja transcrever com mais precisão. Em vocabulary-file-uri, especifique o URI do Amazon Simple Storage Service (Amazon S3) desse arquivo de texto. Em language-code, especifique um código de idioma correspondente ao idioma do vocabulário personalizado. Em vocabulary-name, especifique como você deseja chamar seu vocabulário personalizado.

```
aws transcribe create-vocabulary \
    --language-code language-code \
    --vocabulary-name cli-vocab-example \
    --vocabulary-file-uri s3://amzn-s3-demo-bucket/Amazon-S3-prefix/the-text-
file-for-the-custom-vocabulary.txt
```

Saída:

```
{
    "VocabularyName": "cli-vocab-example",
    "LanguageCode": "language-code",
    "VocabularyState": "PENDING"
}
```

Para obter mais informações, consulte Custom Vocabularies no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte CreateVocabularyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary
 processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
```

```
        logger.exception("Couldn't create custom vocabulary %s.",
vocabulary_name)
        raise
    else:
        return response
```

- Para obter detalhes da API, consulte a CreateVocabularyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->createvocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->createvocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Custom vocabulary created.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
```

```
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateVocabularyreferência da API AWS SDK for SAP ABAP.

Use **`DeleteMedicalTranscriptionJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteMedicalTranscriptionJob`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a medical transcription job. Also deletes the transcript
 associated with the job.
    /// </summary>
    /// <param name="jobName">Name of the medical transcription job to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMedicalTranscriptionJob(string jobName)
    {
        var response = await
 _amazonTranscribeService.DeleteMedicalTranscriptionJobAsync(
            new DeleteMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName
```

```
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteMedicalTranscriptionJoba Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir um trabalho de transcrição médica

O exemplo de `delete-medical-transcription-job` a seguir exclui um trabalho de transcrição médica.

```
aws transcribe delete-medical-transcription-job \
    --medical-transcription-job-name medical-transcription-job-name
```

Este comando não produz saída.

Para obter mais informações, consulte DeleteMedicalTranscriptionJobo Amazon Transcribe Developer Guide.

- Para obter detalhes da API, consulte DeleteMedicalTranscriptionJobem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Excluir um trabalho de transcrição médica.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteMedicalTranscriptionJobCommand } from "@aws-sdk/client-
transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // For example,
 'medical_transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteMedicalTranscriptionJoba Referência AWS SDK para JavaScript da API.

Use **`DeleteTranscriptionJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteTranscriptionJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e refinar um vocabulário personalizado

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a transcription job. Also deletes the transcript associated with
the job.
    /// </summary>
    /// <param name="jobName">Name of the transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTranscriptionJob(string jobName)
    {
        var response = await
_amazonTranscribeService.DeleteTranscriptionJobAsync(
            new DeleteTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTranscriptionJoba Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir um dos trabalhos de transcrição

O exemplo de `delete-transcription-job` a seguir exclui um dos trabalhos de transcrição.

```
aws transcribe delete-transcription-job \
    --transcription-job-name your-transcription-job
```

Este comando não produz saída.

Para obter mais informações, consulte DeleteTranscriptionJobo Amazon Transcribe Developer Guide.

- Para obter detalhes da API, consulte DeleteTranscriptionJobem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir um trabalho de transcrição.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME", // Required. For example, 'transciption_demo'
};

export const run = async () => {
```

```
  try {
    const data = await transcribeClient.send(
      new DeleteTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTranscriptionJoba Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_job(job_name, transcribe_client):
```

```
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteTranscriptionJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_tnb->deletetranscriptionjob( iv_job_name ).
        MESSAGE 'Transcription job deleted.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
```

```
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTranscriptionJobreferência da API AWS SDK for SAP ABAP.

Use **`DeleteVocabulary`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteVocabulary`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e refinar um vocabulário personalizado

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete an existing custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.DeleteVocabularyAsync(
        new DeleteVocabularyRequest
        {
            VocabularyName = vocabularyName
        });
    return response.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- Para obter detalhes da API, consulte DeleteVocabularya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como excluir um vocabulário personalizado

O exemplo de `delete-vocabulary` a seguir exclui um vocabulário personalizado.

```
aws transcribe delete-vocabulary \
    --vocabulary-name vocabulary-name
```

Este comando não produz saída.

Para obter mais informações, consulte Custom Vocabularies no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte DeleteVocabularyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.
```

```
    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteVocabularyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_tnb->deletevocabulary( iv_vocabulary_name ).
        MESSAGE 'Vocabulary deleted.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteVocabularyreferência da API AWS SDK for SAP ABAP.

Use **GetTranscriptionJob**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetTranscriptionJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e refinar um vocabulário personalizado
- Transcrever áudio e obter dados do trabalho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get details about a transcription job.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <returns>A TranscriptionJob instance with information on the requested
job.</returns>
    public async Task<TranscriptionJob> GetTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.GetTranscriptionJobAsync(
            new GetTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.TranscriptionJob;
    }
```

- Para obter detalhes da API, consulte GetTranscriptionJoba Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como obter informações sobre um trabalho de transcrição específico

O exemplo de `get-transcription-job` a seguir coleta informações sobre um trabalho de transcrição específico. Para acessar os resultados da transcrição, use o TranscriptFileUri parâmetro. Use o MediaFileUri parâmetro para ver qual arquivo de áudio você transcreveu com esse trabalho. É possível usar o objeto Configurações para ver os recursos opcionais habilitados no trabalho de transcrição.

```
aws transcribe get-transcription-job \
    --transcription-job-name your-transcription-job
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "your-transcription-job",
        "TranscriptionJobStatus": "COMPLETED",
        "LanguageCode": "language-code",
        "MediaSampleRateHertz": 48000,
        "MediaFormat": "mp4",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-
extension"
        },
        "Transcript": {
            "TranscriptFileUri": "https://Amazon-S3-file-location-of-
transcription-output"
        },
        "StartTime": "2020-09-18T22:27:23.970000+00:00",
        "CreationTime": "2020-09-18T22:27:23.948000+00:00",
        "CompletionTime": "2020-09-18T22:28:21.197000+00:00",
        "Settings": {
            "ChannelIdentification": false,
```

```
            "ShowAlternatives": false
        },
        "IdentifyLanguage": true,
        "IdentifiedLanguageScore": 0.8672199249267578
    }
}
```

Para obter mais informações, consulte Conceitos básicos (interface de linha de AWS comando) no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte GetTranscriptionJobem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job
```

- Para obter detalhes da API, consulte a GetTranscriptionJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_tnb->gettranscriptionjob( iv_job_name ).
    DATA(lo_job) = oo_result->get_transcriptionjob( ).
    MESSAGE 'Retrieved transcription job details.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
    RAISE EXCEPTION lo_bad_request_ex.
  CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
    MESSAGE lo_not_found_ex TYPE 'I'.
    RAISE EXCEPTION lo_not_found_ex.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetTranscriptionJobreferência da API AWS SDK for SAP ABAP.

Use **`GetVocabulary`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetVocabulary`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e refinar um vocabulário personalizado

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get information about a custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> GetCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.GetVocabularyAsync(
        new GetVocabularyRequest()
        {
            VocabularyName = vocabularyName
        });
    return response.VocabularyState;
}
```

- Para obter detalhes da API, consulte GetVocabularya Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como obter informações sobre um vocabulário personalizado

O exemplo de `get-vocabulary` a seguir coleta informações sobre um vocabulário personalizado criado anteriormente.

```
aws transcribe get-vocabulary \
    --vocabulary-name cli-vocab-1
```

Saída:

```
{
    "VocabularyName": "cli-vocab-1",
    "LanguageCode": "language-code",
    "VocabularyState": "READY",
    "LastModifiedTime": "2020-09-19T23:22:32.836000+00:00",
    "DownloadUri": "https://link-to-download-the-text-file-used-to-create-your-
custom-vocabulary"
}
```

Para obter mais informações, consulte Custom Vocabularies no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte GetVocabularyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
```

```
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response =
transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- Para obter detalhes da API, consulte a GetVocabularyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_tnb->getvocabulary( iv_vocabulary_name ).
        MESSAGE 'Retrieved vocabulary details.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
```

```
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetVocabularyreferência da API AWS SDK for SAP ABAP.

Use **`ListMedicalTranscriptionJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListMedicalTranscriptionJobs`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// List medical transcription jobs, optionally with a name filter.
   /// </summary>
   /// <param name="jobNameContains">Optional name filter for the medical
transcription jobs.</param>
   /// <returns>A list of summaries about medical transcription jobs.</returns>
   public async Task<List<MedicalTranscriptionJobSummary>>
ListMedicalTranscriptionJobs(
       string? jobNameContains = null)
   {
       var response = await
_amazonTranscribeService.ListMedicalTranscriptionJobsAsync(
           new ListMedicalTranscriptionJobsRequest()
           {
               JobNameContains = jobNameContains
           });
       return response.MedicalTranscriptionJobSummaries;
```

```
    }
```

- Para obter detalhes da API, consulte ListMedicalTranscriptionJobsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os trabalhos de transcrição médica

O `list-medical-transcription-jobs` exemplo a seguir lista os trabalhos de transcrição médica associados à sua AWS conta e região. Para obter mais informações sobre um trabalho de transcrição específico, copie o valor de um MedicalTranscriptionJobName parâmetro na saída da transcrição e especifique esse valor para a `MedicalTranscriptionJobName` opção do `get-medical-transcription-job` comando. Para ver mais trabalhos de transcrição, copie o valor do NextToken parâmetro, execute o `list-medical-transcription-jobs` comando novamente e especifique esse valor na `--next-token` opção.

```
aws transcribe list-medical-transcription-jobs
```

Saída:

```
{
    "NextToken": "3/PblzkiGhzjER3KHuQt2fmbPLF7cDYafjFMEoGn44ON/
gsuUSTIkGyanvRE6WMXFd/ZTEc2EZj+P9eii/
z1O2FDYli6RLI0WoRX4RwMisVrh9G0Kie0Y8ikBCdtqlZB10Wa9McC+ebOl
+LaDtZPC4u6ttoHLRlEfzqstHXSgapXg3tEBtm9piIaPB6MOM5BB6t86+qtmocTR/
qrteHZBBudhTfbCwhsxaqujHiiUvFdm3BQbKKWIW06yV9b+4f38oD2lVIan
+vfUs3gBYAl5VTDmXXzQPBQOHPjtwmFI+IWX15nSUjWuN3TUylHgPWzDaYT8qBtu0Z+3UG4V6b
+K2CC0XszXg5rBq9hYgNzy4XoFh/6s5DoSnzq49Q9xHgHdT2yBADFmvFK7myZBsj75+2vQZOSVpWUPy3WT/32zFAc
+mFYfUjtTZ8n/jq7aQEjQ42A
+X/7K6JgOcdVPtEg8PlDr5kgYYG3q3OmYXX37U3FZuJmnTI63VtIXsNnOU5eGoYObtpk00Nq9UkzgSJxqj84ZD5n
+S0EGy9ZUYBJRRcGeYUM3Q4DbSJfUwSAqcFdLIWZdp8qIREMQIBWy7BLwSdyqsQo2vRrd53hm5aWM7SVf6pPq6X/
IXR5+1eUOOD8/coaTT4ES2DerbV6RkV4o0VT1d0SdVX/
MmtkNG8nYj8PqU07w7988quh1ZP6D80veJS1q73tUUR9MjnGernW2tAnvnLNhdefBcD
+sZVfYq3iBMFY7wTy1P1G6NqW9GrYDYoX3tTPWlD7phpbVSyKrh/
PdYrps5UxnsGoA1b7L/FfAXDfUoGrGUB4N3JsPYXX9D++g+6gV1qBBs/
```

```
WfF934aKqfD6UTggm/zV3GAOWiBpfvAZRvEb924i6yGHyMC7y54O1ZAwSBupmI
+FFd13CaPO4kN1vJlth6aM5vUPXg4BpyUhtbRhwD/KxCvf9K0tLJGyL1A==",
    "MedicalTranscriptionJobSummaries": [
        {
            "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
            "CreationTime": "2020-09-21T21:17:27.016000+00:00",
            "StartTime": "2020-09-21T21:17:27.045000+00:00",
            "CompletionTime": "2020-09-21T21:17:59.561000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "DICTATION"
        },
        {
            "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
            "CreationTime": "2020-09-21T21:01:14.569000+00:00",
            "StartTime": "2020-09-21T21:01:14.592000+00:00",
            "CompletionTime": "2020-09-21T21:01:43.606000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "DICTATION"
        },
        {
            "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-21T19:09:18.171000+00:00",
            "StartTime": "2020-09-21T19:09:18.199000+00:00",
            "CompletionTime": "2020-09-21T19:10:22.516000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        },
        {
            "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-21T18:43:37.157000+00:00",
            "StartTime": "2020-09-21T18:43:37.265000+00:00",
```

```
            "CompletionTime": "2020-09-21T18:44:21.192000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        },
        {
            "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-20T23:46:44.053000+00:00",
            "StartTime": "2020-09-20T23:46:44.081000+00:00",
            "CompletionTime": "2020-09-20T23:47:35.851000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        }
    ]
}
```

Para obter mais informações, consulte https://docs.aws.amazon.com/transcribe/latest/dg/batch-med-transcription.html > no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte ListMedicalTranscriptionJobsem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
```

```
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Listar trabalhos de transcrição médica.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and
 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the
 same region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListMedicalTranscriptionJobsa Referência AWS SDK para JavaScript da API.

Use **`ListTranscriptionJobs`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListTranscriptionJobs`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List transcription jobs, optionally with a name filter.
    /// </summary>
    /// <param name="jobNameContains">Optional name filter for the transcription
jobs.</param>
    /// <returns>A list of transcription job summaries.</returns>
    public async Task<List<TranscriptionJobSummary>>
ListTranscriptionJobs(string? jobNameContains = null)
    {
        var response = await _amazonTranscribeService.ListTranscriptionJobsAsync(
            new ListTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
        return response.TranscriptionJobSummaries;
    }
```

- Para obter detalhes da API, consulte ListTranscriptionJobsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os trabalhos de transcrição

O `list-transcription-jobs` exemplo a seguir lista os trabalhos de transcrição associados à sua AWS conta e região.

```
aws transcribe list-transcription-jobs
```

Saída:

```
{
    "NextToken": "NextToken",
    "TranscriptionJobSummaries": [
        {
            "TranscriptionJobName": "speak-id-job-1",
            "CreationTime": "2020-08-17T21:06:15.391000+00:00",
            "StartTime": "2020-08-17T21:06:15.416000+00:00",
            "CompletionTime": "2020-08-17T21:07:05.098000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "job-1",
            "CreationTime": "2020-08-17T20:50:24.207000+00:00",
            "StartTime": "2020-08-17T20:50:24.230000+00:00",
            "CompletionTime": "2020-08-17T20:52:18.737000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "sdk-test-job-4",
            "CreationTime": "2020-08-17T20:32:27.917000+00:00",
            "StartTime": "2020-08-17T20:32:27.956000+00:00",
            "CompletionTime": "2020-08-17T20:33:15.126000+00:00",
```

```
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "Diarization-speak-id",
            "CreationTime": "2020-08-10T22:10:09.066000+00:00",
            "StartTime": "2020-08-10T22:10:09.116000+00:00",
            "CompletionTime": "2020-08-10T22:26:48.172000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "your-transcription-job-name",
            "CreationTime": "2020-07-29T17:45:09.791000+00:00",
            "StartTime": "2020-07-29T17:45:09.826000+00:00",
            "CompletionTime": "2020-07-29T17:46:20.831000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        }
    ]
}
```

Para obter mais informações, consulte Conceitos básicos (interface de linha de AWS comando) no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte ListTranscriptionJobsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class ListTranscriptionJobs {
```

```
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

            listTranscriptionJobs(transcribeClient);
        }

        public static void listTranscriptionJobs(TranscribeClient
transcribeClient) {
            ListTranscriptionJobsRequest listJobsRequest =
ListTranscriptionJobsRequest.builder()
                .build();


transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
                .flatMap(response ->
response.transcriptionJobSummaries().stream())
                .forEach(jobSummary -> {
                    System.out.println("Job Name: " +
jobSummary.transcriptionJobName());
                    System.out.println("Job Status: " +
jobSummary.transcriptionJobStatus());
                    System.out.println("Output Location: " +
jobSummary.outputLocationType());
                    // Add more information as needed

                    // Retrieve additional details for the job if necessary
                    GetTranscriptionJobResponse jobDetails =
transcribeClient.getTranscriptionJob(
                        GetTranscriptionJobRequest.builder()

.transcriptionJobName(jobSummary.transcriptionJobName())
                            .build());

                    // Display additional details
                    System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
                    System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
                    // Add more details as needed

                    System.out.println("--------------");
                });
```

```
        }
    }
```

- Para obter detalhes da API, consulte ListTranscriptionJobsa Referência AWS SDK for Java 2.x da API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar trabalhos de transcrição.

```
// Import the required AWS SDK clients and commands for Node.js

import { ListTranscriptionJobsCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  JobNameContains: "KEYWORD", // Not required. Returns only transcription
  // job names containing this string
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new ListTranscriptionJobsCommand(params),
    );
    console.log("Success", data.TranscriptionJobSummaries);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTranscriptionJobsa Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_jobs(job_filter, transcribe_client):
    """
    Lists summaries of the transcription jobs for the current AWS account.

    :param job_filter: The list of returned jobs must contain this string in
 their
                        names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved transcription job summaries.
    """
    try:
        response =
 transcribe_client.list_transcription_jobs(JobNameContains=job_filter)
```

```
        jobs = response["TranscriptionJobSummaries"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_transcription_jobs(
                JobNameContains=job_filter, NextToken=next_token
            )
            jobs += response["TranscriptionJobSummaries"]
            next_token = response.get("NextToken")
        logger.info("Got %s jobs with filter %s.", len(jobs), job_filter)
    except ClientError:
        logger.exception("Couldn't get jobs with filter %s.", job_filter)
        raise
    else:
        return jobs
```

- Para obter detalhes da API, consulte a ListTranscriptionJobsReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        IF iv_job_filter IS NOT INITIAL.
          oo_result = lo_tnb->listtranscriptionjobs( iv_jobnamecontains =
iv_job_filter ).
        ELSE.
          oo_result = lo_tnb->listtranscriptionjobs( ).
        ENDIF.
        MESSAGE 'Retrieved transcription jobs list.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
```

```
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTranscriptionJobsreferência da API AWS SDK for SAP ABAP.

Use **`ListVocabularies`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListVocabularies`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e refinar um vocabulário personalizado

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List custom vocabularies for the current account. Optionally specify a
name
    /// filter and a specific state to filter the vocabularies list.
    /// </summary>
    /// <param name="nameContains">Optional string the vocabulary name must
contain.</param>
    /// <param name="stateEquals">Optional state of the vocabulary.</param>
    /// <returns>List of information about the vocabularies.</returns>
    public async Task<List<VocabularyInfo>> ListCustomVocabularies(string?
nameContains = null,
```

```
        VocabularyState? stateEquals = null)
    {
        var response = await _amazonTranscribeService.ListVocabulariesAsync(
            new ListVocabulariesRequest()
            {
                NameContains = nameContains,
                StateEquals = stateEquals
            });
        return response.Vocabularies;
    }
```

- Para obter detalhes da API, consulte ListVocabularies a Referência AWS SDK para .NET da API.

CLI

AWS CLI

Para listar os vocabulários personalizados

O `list-vocabularies` exemplo a seguir lista os vocabulários personalizados associados à sua AWS conta e região.

```
aws transcribe list-vocabularies
```

Saída:

```
{
    "NextToken": "NextToken",
    "Vocabularies": [
        {
            "VocabularyName": "ards-test-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-27T22:00:27.330000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "sample-test",
            "LanguageCode": "language-code",
```

```
            "LastModifiedTime": "2020-04-24T23:04:11.044000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-test-3-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T22:12:22.277000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-test-2",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T21:53:50.455000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-1-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T21:39:33.356000+00:00",
            "VocabularyState": "READY"
        }
    ]
}
```

Para obter mais informações, consulte Custom Vocabularies no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte ListVocabulariesem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_vocabularies(vocabulary_filter, transcribe_client):
```

```
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string
in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs
```

- Para obter detalhes da API, consulte a ListVocabularies Referência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       IF iv_vocab_filter IS NOT INITIAL.
         oo_result = lo_tnb->listvocabularies( iv_namecontains =
iv_vocab_filter ).
       ELSE.
         oo_result = lo_tnb->listvocabularies( ).
       ENDIF.
       MESSAGE 'Retrieved vocabularies list.' TYPE 'I'.
     CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
       MESSAGE lo_bad_request_ex TYPE 'I'.
       RAISE EXCEPTION lo_bad_request_ex.
     CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
       MESSAGE lo_internal_ex TYPE 'I'.
       RAISE EXCEPTION lo_internal_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListVocabulariesreferência da API AWS SDK for SAP ABAP.

Use **`StartMedicalTranscriptionJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartMedicalTranscriptionJob`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start a medical transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the medical transcription job.</
param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon
 S3 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="outputBucketName">Location for the output, typically an
 Amazon S3 location.</param>
    /// <param name="transcriptionType">Conversation or dictation transcription
 type.</param>
    /// <returns>A MedicalTransactionJob instance with information on the new
 job.</returns>
    public async Task<MedicalTranscriptionJob> StartMedicalTranscriptionJob(
        string jobName, string mediaFileUri,
        MediaFormat mediaFormat, string outputBucketName,
 Amazon.TranscribeService.Type transcriptionType)
    {
        var response = await
 _amazonTranscribeService.StartMedicalTranscriptionJobAsync(
            new StartMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode =
```

```
                    LanguageCode
                        .EnUS, // The value must be en-US for medical
transcriptions.
                OutputBucketName = outputBucketName,
                OutputKey =
                    jobName, // The value is a key used to fetch the output of
the transcription.
                Specialty = Specialty.PRIMARYCARE, // The value PRIMARYCARE must
be set.
                Type = transcriptionType
            });
        return response.MedicalTranscriptionJob;
    }
```

- Para obter detalhes da API, consulte StartMedicalTranscriptionJoba Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: como transcrever um ditado médico armazenado como um arquivo de áudio

O exemplo de `start-medical-transcription-job` a seguir transcreve um arquivo de áudio. Especifique o local da saída da transcrição no parâmetro `OutputBucketName`.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfile.json
```

Conteúdo de `myfile.json`:

```
{
    "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    }
```

```
}
```

Saída:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-
job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T00:35:22.256000+00:00",
        "CreationTime": "2020-09-20T00:35:22.218000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

Para obter mais informações, consulte Batch Transcription Overview no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 2: como transcrever um diálogo entre médico e paciente armazenado como um arquivo de áudio

O exemplo de `start-medical-transcription-job` a seguir transcreve um arquivo de áudio contendo um diálogo entre médico e paciente. Você especifica a localização da saída da transcrição no OutputBucketName parâmetro.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysecondfile.json
```

Conteúdo de `mysecondfile.json`:

```
{
    "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
```

```
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    }
}
```

Saída:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T23:19:49.965000+00:00",
        "CreationTime": "2020-09-20T23:19:49.941000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

Para obter mais informações, consulte Batch Transcription Overview no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 3: como transcrever um arquivo de áudio multicanal de um diálogo entre médico e paciente

O exemplo de `start-medical-transcription-job` a seguir transcreve o áudio de cada canal no arquivo de áudio e mescla as transcrições separadas de cada canal em uma única saída de transcrição. Especifique o local da saída da transcrição no parâmetro `OutputBucketName`.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mythirdfile.json
```

Conteúdo de `mythirdfile.json`:

```
{
    "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
```

```
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
        "Media": {
          "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "Settings":{
          "ChannelIdentification": true
        }
}
```

Saída:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T23:46:44.081000+00:00",
        "CreationTime": "2020-09-20T23:46:44.053000+00:00",
        "Settings": {
            "ChannelIdentification": true
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

Para obter mais informações, consulte Channel Identification no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 4: como transcrever um arquivo de áudio de um diálogo entre médico e paciente e identificar quem está falando na saída da transcrição

O exemplo de `start-medical-transcription-job` a seguir transcreve um arquivo de áudio e rotula a fala de cada locutor na saída da transcrição. Especifique o local da saída da transcrição no parâmetro `OutputBucketName`.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfourthfile.json
```

Conteúdo de `myfourthfile.json`:

```
{
    "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
    "Settings":{
        "ShowSpeakerLabels": true,
        "MaxSpeakerLabels": 2
        }
}
```

Saída:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T18:43:37.265000+00:00",
        "CreationTime": "2020-09-21T18:43:37.157000+00:00",
        "Settings": {
            "ShowSpeakerLabels": true,
            "MaxSpeakerLabels": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

Para obter mais informações, consulte Identifying Speakers no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 5: como transcrever uma conversa médica armazenada como um arquivo de áudio com até duas alternativas de transcrição

O exemplo de `start-medical-transcription-job` a seguir cria até duas transcrições alternativas de um único arquivo de áudio. Cada transcrição tem um nível de confiança associado a ela. Por padrão, o Amazon Transcribe retorna a transcrição com o maior nível de confiança. É possível especificar que o Amazon Transcribe retorne transcrições adicionais com níveis de confiança mais baixos. Especifique o local da saída da transcrição no parâmetro `OutputBucketName`.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfifthfile.json
```

Conteúdo de `myfifthfile.json`:

```
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
        "ShowAlternatives": true,
        "MaxAlternatives": 2
    }
}
```

Saída:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
```

```
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T19:09:18.199000+00:00",
        "CreationTime": "2020-09-21T19:09:18.171000+00:00",
        "Settings": {
            "ShowAlternatives": true,
            "MaxAlternatives": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

Para obter mais informações, consulte Alternative Transcriptions no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 6: como transcrever um arquivo de áudio de um ditado médico com até duas transcrições alternativas

O exemplo de `start-medical-transcription-job` a seguir transcreve um arquivo de áudio e usa um filtro de vocabulário para mascarar qualquer palavra indesejada. Você especifica a localização da saída da transcrição no OutputBucketName parâmetro.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysixthfile.json
```

Conteúdo de `mysixthfile.json`:

```
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
          "ShowAlternatives": true,
```

```
            "MaxAlternatives": 2
    }
}
```

Saída:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T21:01:14.592000+00:00",
        "CreationTime": "2020-09-21T21:01:14.569000+00:00",
        "Settings": {
            "ShowAlternatives": true,
            "MaxAlternatives": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

Para obter mais informações, consulte Alternative Transcriptions no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 7: como transcrever um arquivo de áudio de um ditado médico com maior precisão usando um vocabulário personalizado

O exemplo de `start-medical-transcription-job` a seguir transcreve um arquivo de áudio e usa um vocabulário médico personalizado que você criou anteriormente para aumentar a precisão da transcrição. Especifique o local da saída da transcrição no parâmetro `OutputBucketName`.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myseventhfile.json
```

Conteúdo de `mysixthfile.json`:

```
{
    "MedicalTranscriptionJobName": "vocabulary-dictation-medical-transcription-
job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
        "VocabularyName": "cli-medical-vocab-1"
    }
}
```

Saída:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T21:17:27.045000+00:00",
        "CreationTime": "2020-09-21T21:17:27.016000+00:00",
        "Settings": {
            "VocabularyName": "cli-medical-vocab-1"
        },
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

Para obter mais informações, consulte Medical Custom Vocabularies no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte StartMedicalTranscriptionJobem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Iniciar um trabalho de transcrição médica.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and
 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the
 same region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
```

```
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte StartMedicalTranscriptionJoba Referência AWS SDK para JavaScript da API.

Use **`StartTranscriptionJob`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `StartTranscriptionJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Criar e refinar um vocabulário personalizado
- Transcrever áudio e obter dados do trabalho

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start a transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon
 S3 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="languageCode">The language code of the media file, such as
 en-US.</param>
    /// <param name="vocabularyName">Optional name of a custom vocabulary.</
param>
    /// <returns>A TranscriptionJob instance with information on the new job.</
returns>
    public async Task<TranscriptionJob> StartTranscriptionJob(string jobName,
 string mediaFileUri,
        MediaFormat mediaFormat, LanguageCode languageCode, string?
 vocabularyName)
    {
        var response = await _amazonTranscribeService.StartTranscriptionJobAsync(
            new StartTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode = languageCode,
                Settings = vocabularyName != null ? new Settings()
                {
                    VocabularyName = vocabularyName
                } : null
            });
        return response.TranscriptionJob;
    }
```

- Para obter detalhes da API, consulte StartTranscriptionJoba Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: como transcrever um arquivo de áudio

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfile.json
```

Conteúdo de `myfile.json`:

```
{
    "TranscriptionJobName": "cli-simple-transcription-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

Para obter mais informações, consulte Conceitos básicos (interface de linha de AWS comando) no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 2: como transcrever um arquivo de áudio multicanal

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio multicanal.

```
aws transcribe start-transcription-job \
    --cli-input-json file://mysecondfile.json
```

Conteúdo de `mysecondfile.json`:

```
{
    "TranscriptionJobName": "cli-channelid-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
```

```
    },
    "Settings":{
        "ChannelIdentification":true
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-channelid-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-17T16:07:56.817000+00:00",
        "CreationTime": "2020-09-17T16:07:56.784000+00:00",
        "Settings": {
            "ChannelIdentification": true
        }
    }
}
```

Para obter mais informações, consulte Transcrição de Multi-Channel áudio no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 3: como transcrever um arquivo de áudio e identificar quem está falando

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio e identifica os locutores na saída da transcrição.

```
aws transcribe start-transcription-job \
    --cli-input-json file://mythirdfile.json
```

Conteúdo de `mythirdfile.json`:

```
{
    "TranscriptionJobName": "cli-speakerid-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
```

```
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
    "ShowSpeakerLabels": true,
    "MaxSpeakerLabels": 2
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-speakerid-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-17T16:22:59.696000+00:00",
        "CreationTime": "2020-09-17T16:22:59.676000+00:00",
        "Settings": {
            "ShowSpeakerLabels": true,
            "MaxSpeakerLabels": 2
        }
    }
}
```

Para obter mais informações, consulte Identifying Speakers no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 4: como transcrever um arquivo de áudio e mascarar palavras indesejadas na saída da transcrição

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio e usa um filtro de vocabulário que você criou anteriormente para mascarar palavras indesejadas.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfourthfile.json
```

Conteúdo de `myfourthfile.json`:

```
{
    "TranscriptionJobName": "cli-filter-mask-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
          "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyFilterName": "your-vocabulary-filter",
        "VocabularyFilterMethod": "mask"
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-filter-mask-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyFilterName": "your-vocabulary-filter",
            "VocabularyFilterMethod": "mask"
        }
    }
}
```

Para obter mais informações, consulte Filtering Transcriptions o Guia do desenvolvedor do Amazon Transcribe.

Exemplo 5: como transcrever um arquivo de áudio e remover palavras indesejadas na saída da transcrição

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio e usa um filtro de vocabulário que você criou anteriormente para mascarar palavras indesejadas.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfifthfile.json
```

Conteúdo de `myfifthfile.json`:

```
{
    "TranscriptionJobName": "cli-filter-remove-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyFilterName": "your-vocabulary-filter",
        "VocabularyFilterMethod": "remove"
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-filter-remove-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyFilterName": "your-vocabulary-filter",
            "VocabularyFilterMethod": "remove"
        }
    }
}
```

Para obter mais informações, consulte Filtering Transcriptions o Guia do desenvolvedor do Amazon Transcribe.

Exemplo 6: como transcrever um arquivo de áudio com maior precisão usando um vocabulário personalizado

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio e usa um filtro de vocabulário que você criou anteriormente para mascarar palavras indesejadas.

```
aws transcribe start-transcription-job \
    --cli-input-json file://mysixthfile.json
```

Conteúdo de `mysixthfile.json`:

```
{
    "TranscriptionJobName": "cli-vocab-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyName": "your-vocabulary"
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-vocab-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyName": "your-vocabulary"
        }
    }
}
```

Para obter mais informações, consulte Filtering Transcriptions o Guia do desenvolvedor do Amazon Transcribe.

Exemplo 7: como identificar o idioma de um arquivo de áudio e transcrevê-lo

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio e usa um filtro de vocabulário que você criou anteriormente para mascarar palavras indesejadas.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myseventhfile.json
```

Conteúdo de `myseventhfile.json`:

```
{
    "TranscriptionJobName": "cli-identify-language-transcription-job",
    "IdentifyLanguage": true,
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-identify-language-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T22:27:23.970000+00:00",
        "CreationTime": "2020-09-18T22:27:23.948000+00:00",
        "IdentifyLanguage": true
    }
}
```

Para obter mais informações, consulte Identifying the Language no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 8: como transcrever um arquivo de áudio com informações de identificação pessoal censuradas

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio e oculta qualquer informação de identificação pessoal na saída da transcrição.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myeighthfile.json
```

Conteúdo de `myeigthfile.json`:

```
{
    "TranscriptionJobName": "cli-redaction-job",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
    },
    "ContentRedaction": {
        "RedactionOutput":"redacted",
        "RedactionType":"PII"
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-redaction-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-25T23:49:13.195000+00:00",
        "CreationTime": "2020-09-25T23:49:13.176000+00:00",
        "ContentRedaction": {
            "RedactionType": "PII",
            "RedactionOutput": "redacted"
        }
    }
}
```

Para obter mais informações, consulte Automatic Content Redaction no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 9: como gerar uma transcrição com informações de identificação pessoal (PII) censuradas e uma transcrição não censurada

O exemplo de `start-transcription-job` a seguir gera duas transcrições do arquivo de áudio, uma com as informações de identificação pessoal editadas e a outra sem nenhuma edição.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myninthfile.json
```

Conteúdo de `myninthfile.json`:

```
{
    "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
    "LanguageCode": "language-code",
    "Media": {
          "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
        },
    "ContentRedaction": {
        "RedactionOutput":"redacted_and_unredacted",
        "RedactionType":"PII"
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-25T23:59:47.677000+00:00",
        "CreationTime": "2020-09-25T23:59:47.653000+00:00",
        "ContentRedaction": {
            "RedactionType": "PII",
```

```
            "RedactionOutput": "redacted_and_unredacted"
        }
    }
}
```

Para obter mais informações, consulte Automatic Content Redaction no Guia do desenvolvedor do Amazon Transcribe.

Exemplo 10: como usar um modelo de linguagem personalizado criado anteriormente para transcrever um arquivo de áudio.

O exemplo de `start-transcription-job` a seguir transcreve o arquivo de áudio com um modelo de idioma personalizado que você criou anteriormente.

```
aws transcribe start-transcription-job \
    --cli-input-json file://mytenthfile.json
```

Conteúdo de `mytenthfile.json`:

```
{
    "TranscriptionJobName": "cli-clm-2-job-1",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-extension"
    },
    "ModelSettings": {
        "LanguageModelName":"cli-clm-2"
    }
}
```

Saída:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-clm-2-job-1",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-
extension"
        },
        "StartTime": "2020-09-28T17:56:01.835000+00:00",
```

```
        "CreationTime": "2020-09-28T17:56:01.801000+00:00",
        "ModelSettings": {
            "LanguageModelName": "cli-clm-2"
        }
    }
}
```

Para obter mais informações, consulte Melhorando a precisão da Domain-Specific transcrição com modelos de linguagem personalizados no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte StartTranscriptionJobem Referência de AWS CLI Comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Iniciar um trabalho de transcrição.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME",
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_LOCATION",
    // For example, "https://transcribe-demo.s3-REGION.amazonaws.com/
hello_world.wav"
  },
  OutputBucketName: "OUTPUT_BUCKET_NAME",
};
```

```
export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte StartTranscriptionJoba Referência AWS SDK para JavaScript da API.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def start_job(
```

```
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is
 started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
 transcribing
                             the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job
```

- Para obter detalhes da API, consulte a StartTranscriptionJobReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_media) = NEW /aws1/cl_tnbmedia( iv_mediafileuri = iv_media_uri ).
        DATA(lo_settings) = NEW /aws1/cl_tnbsettings( ).
        IF iv_vocabulary_name IS NOT INITIAL.
          lo_settings = NEW /aws1/cl_tnbsettings( iv_vocabularyname =
iv_vocabulary_name ).
        ENDIF.

        oo_result = lo_tnb->starttranscriptionjob(
          iv_transcriptionjobname = iv_job_name
          io_media = lo_media
          iv_mediaformat = iv_media_format
          iv_languagecode = iv_language_code
          io_settings = lo_settings ).

        MESSAGE 'Transcription job started.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
```

```
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartTranscriptionJobreferência da API AWS SDK for SAP ABAP.

Use **`UpdateVocabulary`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateVocabulary`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criar e refinar um vocabulário personalizado

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update a custom vocabulary with new values. Update overwrites all
existing information.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> UpdateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
```

```
        var response = await _amazonTranscribeService.UpdateVocabularyAsync(
            new UpdateVocabularyRequest()
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- Para obter detalhes da API, consulte UpdateVocabulary a Referência AWS SDK para .NET da API.

CLI

AWS CLI

Como atualizar um vocabulário personalizado com novos termos.

O exemplo de `update-vocabulary` a seguir substitui os termos usados para criar o vocabulário personalizado pelos novos termos fornecidos. Pré-requisito: para substituir os termos em um vocabulário personalizado, você precisa de um arquivo com novos termos.

```
aws transcribe update-vocabulary \
    --vocabulary-file-uri s3://amzn-s3-demo-bucket/Amazon-S3-Prefix/custom-vocabulary.txt \
    --vocabulary-name custom-vocabulary \
    --language-code language-code
```

Saída:

```
{
    "VocabularyName": "custom-vocabulary",
    "LanguageCode": "language",
    "VocabularyState": "PENDING"
}
```

Para obter mais informações, consulte Custom Vocabularies no Guia do desenvolvedor do Amazon Transcribe.

- Para obter detalhes da API, consulte UpdateVocabularyem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
```

```
        logger.exception("Couldn't update custom vocabulary %s.",
vocabulary_name)
        raise
```

- Para obter detalhes da API, consulte a UpdateVocabularyReferência da API AWS SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Vocabulary updated.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
```

```
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateVocabularyreferência da API AWS SDK for SAP ABAP.

Cenários para o Amazon Transcribe usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon AWS Transcribe com SDKs. Esses casos mostram como executar tarefas específicas chamando várias funções no Amazon Transcribe ou em combinação com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação de transmissão do Amazon Transcribe
- Converta texto em fala e volte para texto usando um AWS SDK
- Crie e refine um vocabulário personalizado do Amazon Transcribe usando um AWS SDK
- Transcreva áudio e obtenha dados de trabalho com o Amazon Transcribe usando um AWS SDK

Criar uma aplicação de transmissão do Amazon Transcribe

O exemplo de código a seguir mostra como construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Transcribe para construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Converta texto em fala e volte para texto usando um AWS SDK

O exemplo de código a seguir mostra como:

- Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio.
- Fazer upload do arquivo de áudio para um bucket do Amazon S3.
- Usar o Amazon Transcribe para converter o arquivo de áudio em texto.
- Exibir o texto.

Rust

SDK para Rust

Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio, fazer upload do arquivo de áudio em um bucket do Amazon S3, usar o Amazon Transcribe para converter esse arquivo de áudio em texto e exibir o texto.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Crie e refine um vocabulário personalizado do Amazon Transcribe usando um AWS SDK

O exemplo de código a seguir mostra como:

- Fazer upload de um arquivo de áudio para o Amazon S3.
- Executar um trabalho do Amazon Transcribe para transcrever o arquivo e obter os resultados.
- Criar e refinar um vocabulário personalizado para melhorar a precisão da transcrição.
- Executar trabalhos com vocabulários personalizados e obter os resultados.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Transcreva um arquivo de áudio que contenha uma leitura de Jabberwocky de Lewis Carroll. Comece criando funções que envolvam as ações do Amazon Transcribe.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
```

```
    Starts a transcription job. This function returns as soon as the job is
 started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
 transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job



def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
```

```
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job



def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise



def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary
 processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.
```

```
    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.",
 vocabulary_name)
        raise
    else:
        return response



def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response =
 transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
```

```
    else:
        return response



def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.",
 vocabulary_name)
        raise



def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string
 in
```

```
                                their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs



def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

Chame as funções do wrapper para transcrever áudio sem um vocabulário personalizado e, em seguida, com diferentes versões de um vocabulário personalizado para ver os melhores resultados.

```
def usage_demo():
    """Shows how to use the Amazon Transcribe service."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    s3_resource = boto3.resource("s3")
    transcribe_client = boto3.client("transcribe")

    print("-" * 88)
    print("Welcome to the Amazon Transcribe demo!")
    print("-" * 88)

    bucket_name = f"jabber-bucket-{time.time_ns()}"
    print(f"Creating bucket {bucket_name}.")
    bucket = s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": transcribe_client.meta.region_name
        },
    )
    media_file_name = ".media/Jabberwocky.mp3"
    media_object_key = "Jabberwocky.mp3"
    print(f"Uploading media file {media_file_name}.")
    bucket.upload_file(media_file_name, media_object_key)
    media_uri = f"s3://{bucket.name}/{media_object_key}"

    job_name_simple = f"Jabber-{time.time_ns()}"
    print(f"Starting transcription job {job_name_simple}.")
    start_job(
        job_name_simple,
        f"s3://{bucket_name}/{media_object_key}",
        "mp3",
        "en-US",
        transcribe_client,
    )
    transcribe_waiter = TranscribeCompleteWaiter(transcribe_client)
    transcribe_waiter.wait(job_name_simple)
    job_simple = get_job(job_name_simple, transcribe_client)
```

```
    transcript_simple = requests.get(
        job_simple["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_simple['jobName']}:")
    print(transcript_simple["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Creating a custom vocabulary that lists the nonsense words to try to "
        "improve the transcription."
    )
    vocabulary_name = f"Jabber-vocabulary-{time.time_ns()}"
    create_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        phrases=[
            "brillig",
            "slithy",
            "borogoves",
            "mome",
            "raths",
            "Jub-Jub",
            "frumious",
            "manxome",
            "Tumtum",
            "uffish",
            "whiffling",
            "tulgey",
            "thou",
            "frabjous",
            "callooh",
            "callay",
            "chortled",
        ],
    )
    vocabulary_ready_waiter = VocabularyReadyWaiter(transcribe_client)
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocabulary_list = f"Jabber-vocabulary-list-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocabulary_list}.")
    start_job(
        job_name_vocabulary_list,
        media_uri,
```

```
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocabulary_list)
    job_vocabulary_list = get_job(job_name_vocabulary_list, transcribe_client)
    transcript_vocabulary_list = requests.get(
        job_vocabulary_list["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocabulary_list['jobName']}:")
    print(transcript_vocabulary_list["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Updating the custom vocabulary with table data that provides additional
"
        "pronunciation hints."
    )
    table_vocab_file = "jabber-vocabulary-table.txt"
    bucket.upload_file(table_vocab_file, table_vocab_file)
    update_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        table_uri=f"s3://{bucket.name}/{table_vocab_file}",
    )
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocab_table = f"Jabber-vocab-table-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocab_table}.")
    start_job(
        job_name_vocab_table,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name=vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocab_table)
    job_vocab_table = get_job(job_name_vocab_table, transcribe_client)
    transcript_vocab_table = requests.get(
        job_vocab_table["Transcript"]["TranscriptFileUri"]
    ).json()
```

```
    print(f"Transcript for job {transcript_vocab_table['jobName']}:")
    print(transcript_vocab_table["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print("Getting data for jobs and vocabularies.")
    jabber_jobs = list_jobs("Jabber", transcribe_client)
    print(f"Found {len(jabber_jobs)} jobs:")
    for job_sum in jabber_jobs:
        job = get_job(job_sum["TranscriptionJobName"], transcribe_client)
        print(
            f"\t{job['TranscriptionJobName']}, {job['Media']['MediaFileUri']}, "
            f"{job['Settings'].get('VocabularyName')}"
        )

    jabber_vocabs = list_vocabularies("Jabber", transcribe_client)
    print(f"Found {len(jabber_vocabs)} vocabularies:")
    for vocab_sum in jabber_vocabs:
        vocab = get_vocabulary(vocab_sum["VocabularyName"], transcribe_client)
        vocab_content = requests.get(vocab["DownloadUri"]).text
        print(f"\t{vocab['VocabularyName']} contents:")
        print(vocab_content)

    print("-" * 88)
    print("Deleting demo jobs.")
    for job_name in [job_name_simple, job_name_vocabulary_list,
job_name_vocab_table]:
        delete_job(job_name, transcribe_client)
    print("Deleting demo vocabulary.")
    delete_vocabulary(vocabulary_name, transcribe_client)
    print("Deleting demo bucket.")
    bucket.objects.delete()
    bucket.delete()
    print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateVocabulary
  - DeleteTranscriptionJob
  - DeleteVocabulary

- GetTranscriptionJob
- GetVocabulary
- ListVocabularies
- StartTranscriptionJob
- UpdateVocabulary

Transcreva áudio e obtenha dados de trabalho com o Amazon Transcribe usando um AWS SDK

Os exemplos de código a seguir mostram como:

- Iniciar um trabalho de transcrição com o Amazon Transcribe.
- Aguardar a conclusão do trabalho.
- Obter o URI em que a transcrição está armazenada.

Para obter mais informações, consulte Começar a usar o Amazon Transcribe.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Transcreve um arquivo PCM.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }
```

```
    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
```

```
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
 positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
```

```
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
```

```
    }
}
```

Transcreve o streaming de áudio do microfone do computador.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
 {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
```

```
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
```

```
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
 positive"));
```

```
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }
```

```
            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetTranscriptionJob
  - StartTranscriptionJob

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import time
import boto3


def transcribe_file(job_name, file_uri, transcribe_client):
    transcribe_client.start_transcription_job(
        TranscriptionJobName=job_name,
        Media={"MediaFileUri": file_uri},
        MediaFormat="wav",
        LanguageCode="en-US",
    )
```

```
    max_tries = 60
    while max_tries > 0:
        max_tries -= 1
        job =
 transcribe_client.get_transcription_job(TranscriptionJobName=job_name)
        job_status = job["TranscriptionJob"]["TranscriptionJobStatus"]
        if job_status in ["COMPLETED", "FAILED"]:
            print(f"Job {job_name} is {job_status}.")
            if job_status == "COMPLETED":
                print(
                    f"Download the transcript from\n"
                    f"\t{job['TranscriptionJob']['Transcript']
['TranscriptFileUri']}."
                )
            break
        else:
            print(f"Waiting for {job_name}. Current status is {job_status}.")
        time.sleep(10)


def main():
    transcribe_client = boto3.client("transcribe")
    file_uri = "s3://test-transcribe/answer2.wav"
    transcribe_file("Example-job", file_uri, transcribe_client)


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - GetTranscriptionJob
  - StartTranscriptionJob

Exemplos de código para Amazon Transcribe Streaming usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Transcribe Streaming com AWS um kit de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Transcribe Streaming: mais informações sobre o Amazon Transcribe Streaming.
- Referência de API do Amazon Transcribe Streaming: detalhes sobre todas as ações disponíveis do Amazon Transcribe Streaming.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos do Amazon Transcribe Streaming usando AWS SDKs
  - Ações para Amazon Transcribe Streaming usando AWS SDKs
    - Use StartMedicalStreamTranscriptioncom um AWS SDK
    - Use StartStreamTranscriptioncom um AWS SDK
- Cenários para o Amazon Transcribe Streaming usando AWS SDKs
  - Gere uma transcrição de um arquivo usando um AWS SDK
  - Gere uma transcrição de um microfone usando um AWS SDK

Exemplos básicos do Amazon Transcribe Streaming usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Transcribe Streaming com AWS SDKs.

Exemplos

- Ações para Amazon Transcribe Streaming usando AWS SDKs
  - Use StartMedicalStreamTranscriptioncom um AWS SDK

- Use StartStreamTranscriptioncom um AWS SDK

Ações para Amazon Transcribe Streaming usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Transcribe Streaming com SDKs. AWS Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Transcribe Streaming e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Transcribe Streaming usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Transcribe Streaming.

Exemplos

- Use StartMedicalStreamTranscriptioncom um AWS SDK
- Use StartStreamTranscriptioncom um AWS SDK

Use **`StartMedicalStreamTranscription`**com um AWS SDK

O código de exemplo a seguir mostra como usar `StartMedicalStreamTranscription`.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/*
To run this AWS code example, ensure that you have set up your development
 environment, including your AWS credentials.

 For information, see this documentation topic:
```

```
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

This code demonstrates the process of starting a medical transcription job using
 the AWS Transcribe
Streaming service, including setting up the audio input stream, configuring the
 transcription request,
and handling the transcription response.
 */

public class TranscribeMedicalStreamingDemoApp {
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws ExecutionException, InterruptedException, LineUnavailableException
 {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .build();

        CompletableFuture<Void> result =
 client.startMedicalStreamTranscription(getMedicalRequest(16_000),
            new AudioStreamPublisher(getStreamFromMic()),
            getMedicalResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
 {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            throw new LineUnavailableException("The audio system microphone line
 is not supported.");
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
```

```
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartMedicalStreamTranscriptionRequest
getMedicalRequest(Integer mediaSampleRateHertz) {
        return StartMedicalStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString()) // For medical
transcription, EN_US is typically used.
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .specialty(Specialty.PRIMARYCARE) // Specify the medical specialty.
            .type(Type.CONVERSATION) // Set the type as CONVERSATION or
DICTATION.
            .build();
    }

    private static StartMedicalStreamTranscriptionResponseHandler
getMedicalResponseHandler() {
        return StartMedicalStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records streamed successfully ===");
            })
            .subscriber(event -> {
                List<MedicalResult> results = ((MedicalTranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
```

```
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }
```

```
        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);
            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
```

```
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                .audioChunk(SdkBytes.fromByteBuffer(bb))
                .build();
        }
    }
}
```

- Para obter detalhes da API, consulte StartMedicalStreamTranscriptiona Referência AWS SDK for Java 2.x da API.

Use **`StartStreamTranscription`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartStreamTranscription`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- Transcrever um arquivo de áudio
- Transcrever o áudio de um microfone

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if
 the SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
                [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
 &error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
        //SetTranscriptEventCallback called for every 'chunk' of file
 transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
```

```
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
                                  << std::endl;

                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
                    //the number of characters extracted by the last read()
operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
std::endl;
                            break;
                        }
```

```
                    }
                    else {
                        break;
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a
 file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
                    // Per the spec, we have to send an empty event (an event
 without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" <<
 std::endl;
                }
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };

        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
```

```
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- Para obter detalhes da API, consulte StartStreamTranscriptiona Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());
```

```
        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
{

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
```

```
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
```

```
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
```

```
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Para obter detalhes da API, consulte StartStreamTranscriptiona Referência AWS SDK for Java 2.x da API.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let client = TranscribeStreamingClient(
            config: try await
TranscribeStreamingClient.TranscribeStreamingClientConfiguration(
                region: region
            )
        )

        // Start the transcription running on the audio stream.

        let output = try await client.startStreamTranscription(
            input: StartStreamTranscriptionInput(
                audioStream: try await createAudioStream(),
                languageCode:
TranscribeStreamingClientTypes.LanguageCode(rawValue: lang),
                mediaEncoding: encoding,
                mediaSampleRateHertz: sampleRate
            )
        )
```

- Para obter detalhes da API, consulte StartStreamTranscriptiona referência da API AWS SDK for Swift.

Cenários para o Amazon Transcribe Streaming usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Transcribe Streaming com SDKs. AWS Esses casos mostram como executar tarefas específicas chamando várias funções no Amazon Transcribe Streaming ou em combinação com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Gere uma transcrição de um arquivo usando um AWS SDK
- Gere uma transcrição de um microfone usando um AWS SDK

Gere uma transcrição de um arquivo usando um AWS SDK

Os exemplos de código a seguir mostram como gerar uma transcrição de um arquivo de áudio de origem usando o streaming do Amazon Transcribe.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if
 the SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
```

```
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
                [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
 &error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
        //SetTranscriptEventCallback called for every 'chunk' of file
 transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
 std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
```

```
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
                                  << std::endl;

                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
                    //the number of characters extracted by the last read()
operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
std::endl;
                            break;
                        }
                    }
                    else {
                        break;
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a
file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
                    // Per the spec, we have to send an empty event (an event
without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" <<
std::endl;
                }
```

```
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };

        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- Para obter detalhes da API, consulte StartStreamTranscriptiona Referência AWS SDK para C++ da API.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
```

```
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
```

```
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);
```

```
        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];
```

```
            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Para obter detalhes da API, consulte StartStreamTranscriptiona Referência AWS SDK for Java 2.x da API.

Swift

SDK para Swift

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o streaming do Amazon Transcribe para transcrever o idioma falado em um arquivo de áudio.

```
/// An example that demonstrates how to watch an transcribe event stream to
/// transcribe audio from a file to the console.

import ArgumentParser
import AWSClientRuntime
import AWSTranscribeStreaming
import Foundation

/// Identify one of the media file formats supported by Amazon Transcribe.
enum TranscribeFormat: String, ExpressibleByArgument {
    case ogg = "ogg"
    case pcm = "pcm"
    case flac = "flac"
}

// -MARK: - Async command line tool

struct ExampleCommand: ParsableCommand {
    // -MARK: Command arguments
    @Flag(help: "Show partial results")
    var showPartial = false
    @Option(help: "Language code to transcribe into")
    var lang: String = "en-US"
    @Option(help: "Format of the source audio file")
    var format: TranscribeFormat
    @Option(help: "Sample rate of the source audio file in Hertz")
    var sampleRate: Int = 16000
    @Option(help: "Path of the source audio file")
    var path: String
    @Option(help: "Name of the Amazon S3 Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "tsevents",
        abstract: """
        This example shows how to use event streaming with Amazon Transcribe.
        """,
        discussion: """
        """
    )

    /// Create and return an Amazon Transcribe audio stream from the file
    /// specified in the arguments.
```

```
   ///
   /// - Throws: Errors from `TranscribeError`.
   ///
   /// - Returns:
`AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream, Error>`
   func createAudioStream() async throws
               ->
AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream, Error> {

       let fileURL: URL = URL(fileURLWithPath: path)
       let audioData = try Data(contentsOf: fileURL)

       // Properties defining the size of audio chunks and the total size of
       // the audio file in bytes. You should try to send chunks that last on
       // average 125 milliseconds.

       let chunkSizeInMilliseconds = 125.0
       let chunkSize = Int(chunkSizeInMilliseconds  / 1000.0 *
Double(sampleRate) * 2.0)
       let audioDataSize = audioData.count

       // Create an audio stream from the source data. The stream's job is
       // to send the audio in chunks to Amazon Transcribe as
       // `AudioStream.audioevent` events.

       let audioStream =
AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream,
                               Error> { continuation in
           Task {
               var currentStart = 0
               var currentEnd = min(chunkSize, audioDataSize - currentStart)

               // Generate and send chunks of audio data as `audioevent`
               // events until the entire file has been sent. Each event is
               // yielded to the SDK after being created.

               while currentStart < audioDataSize {
                   let dataChunk = audioData[currentStart ..< currentEnd]

                   let audioEvent =
TranscribeStreamingClientTypes.AudioStream.audioevent(
                       .init(audioChunk: dataChunk)
                   )
                   let yieldResult = continuation.yield(audioEvent)
```

```
                    switch yieldResult {
                        case .enqueued(_):
                            // The chunk was successfully enqueued into the
                            // stream. The `remaining` parameter estimates how
                            // much room is left in the queue, but is ignored
here.
                            break
                        case .dropped(_):
                            // The chunk was dropped because the queue buffer
                            // is full. This will cause transcription errors.
                            print("Warning: Dropped audio! The transcription will
be incomplete.")
                        case .terminated:
                            print("Audio stream terminated.")
                            continuation.finish()
                            return
                        default:
                            print("Warning: Unrecognized response during audio
streaming.")
                    }

                    currentStart = currentEnd
                    currentEnd = min(currentStart + chunkSize, audioDataSize)
                }

                // Let the SDK's continuation block know the stream is over.

                continuation.finish()
            }
        }

        return audioStream
    }

    /// Run the transcription process.
    ///
    /// - Throws: An error from `TranscribeError`.
    func transcribe(encoding: TranscribeStreamingClientTypes.MediaEncoding) async
throws {
        // Create the Transcribe Streaming client.

        let client = TranscribeStreamingClient(
            config: try await
TranscribeStreamingClient.TranscribeStreamingClientConfiguration(
```

```
                region: region
            )
        )

        // Start the transcription running on the audio stream.

        let output = try await client.startStreamTranscription(
            input: StartStreamTranscriptionInput(
                audioStream: try await createAudioStream(),
                languageCode:
TranscribeStreamingClientTypes.LanguageCode(rawValue: lang),
                mediaEncoding: encoding,
                mediaSampleRateHertz: sampleRate
            )
        )

        // Iterate over the events in the returned transcript result stream.
        // Each `transcriptevent` contains a list of result fragments which
        // need to be concatenated together to build the final transcript.
        for try await event in output.transcriptResultStream! {
            switch event {
            case .transcriptevent(let event):
            for result in event.transcript?.results ?? [] {
                guard let transcript = result.alternatives?.first?.transcript
else {
                    continue
                }

                // If showing partial results is enabled and the result is
                // partial, show it. Partial results may be incomplete, and
                // may be inaccurate, with upcoming audio making the
                // transcription complete or by giving more context to make
                // transcription make more sense.

                if (result.isPartial && showPartial) {
                    print("[Partial] \(transcript)")
                }

                // When the complete fragment of transcribed text is ready,
                // print it. This could just as easily be used to draw the
                // text as a subtitle over a playing video, though timing
                // would need to be managed.

                if !result.isPartial {
```

```
                    if (showPartial) {
                        print("[Final  ] ", terminator: "")
                    }
                    print(transcript)
                }
            }
            default:
                print("Error: Unexpected message from Amazon Transcribe:")
            }
        }
    }

    /// Convert the value of the `--format` command line option into the
    /// corresponding Transcribe Streaming `MediaEncoding` type.
    ///
    /// - Returns: The `MediaEncoding` equivalent of the format specified on
    ///   the command line.
    func getMediaEncoding() -> TranscribeStreamingClientTypes.MediaEncoding {
        let mediaEncoding: TranscribeStreamingClientTypes.MediaEncoding

        switch format {
        case .flac:
            mediaEncoding = .flac
        case .ogg:
            mediaEncoding = .oggOpus
        case .pcm:
            mediaEncoding = .pcm
        }

        return mediaEncoding
    }
}

// -MARK: - Entry point

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.transcribe(encoding: command.getMediaEncoding())
```

```
        } catch let error as TranscribeError {
            print("ERROR: \(error.errorDescription ?? "Unknown error")")
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}

/// Errors thrown by the example's functions.
enum TranscribeError: Error {
    /// No transcription stream available.
    case noTranscriptionStream
    /// The source media file couldn't be read.
    case readError

    var errorDescription: String? {
        switch self {
        case .noTranscriptionStream:
            return "No transcription stream returned by Amazon Transcribe."
        case .readError:
            return "Unable to read the source audio file."
        }
    }
}
```

- Para obter detalhes da API, consulte StartStreamTranscriptiona referência da API AWS SDK for Swift.

Gere uma transcrição de um microfone usando um AWS SDK

O exemplo de código a seguir mostra como gerar uma transcrição por um microfone usando o streaming do Amazon Transcribe.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
 {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
```

```
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
event).transcript().results();
                    if (results.size() > 0) {
```

```
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }
```

```
        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
```

```
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Para obter detalhes da API, consulte StartStreamTranscriptiona Referência AWS SDK for Java 2.x da API.

Exemplos de código para o Amazon Translate usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon Translate com um kit AWS de desenvolvimento de software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do Amazon Translate: mais informações sobre o Amazon Translate.
- Referência da API Amazon Translate: detalhes sobre todas as ações disponíveis do Amazon Translate.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.

- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para o uso do Amazon Translate AWS SDKs
  - Ações para o Amazon Translate usando AWS SDKs
    - Use DescribeTextTranslationJobcom um AWS SDK
    - Use ListTextTranslationJobscom um AWS SDK
    - Use StartTextTranslationJobcom um AWS SDK
    - Use StopTextTranslationJobcom um AWS SDK
    - Use TranslateTextcom um AWS SDK ou CLI
- Cenários para o Amazon Translate usando AWS SDKs
  - Criar uma aplicação de transmissão do Amazon Transcribe
  - Crie um chatbot do Amazon Lex para engajar os visitantes do site
  - Criar uma aplicação de publicação e assinatura que traduz mensagens
  - Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
  - Comece a usar trabalhos do Amazon Translate usando um AWS SDK

Exemplos básicos para o uso do Amazon Translate AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Translate com AWS SDKs.

Exemplos

- Ações para o Amazon Translate usando AWS SDKs
  - Use DescribeTextTranslationJobcom um AWS SDK
  - Use ListTextTranslationJobscom um AWS SDK
  - Use StartTextTranslationJobcom um AWS SDK
  - Use StopTextTranslationJobcom um AWS SDK
  - Use TranslateTextcom um AWS SDK ou CLI

Ações para o Amazon Translate usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais do Amazon Translate com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon Translate e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em Cenários para o Amazon Translate usando AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na Referência de API do Amazon Translate.

Exemplos

- Use DescribeTextTranslationJobcom um AWS SDK
- Use ListTextTranslationJobscom um AWS SDK
- Use StartTextTranslationJobcom um AWS SDK
- Use StopTextTranslationJobcom um AWS SDK
- Use TranslateTextcom um AWS SDK ou CLI

Use **`DescribeTextTranslationJob`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `DescribeTextTranslationJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de trabalhos de tradução

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
    /// The following example shows how to retrieve the details of
    /// a text translation job using Amazon Translate.
    /// </summary>
    public class DescribeTextTranslation
    {
        public static async Task Main()
        {
            var client = new AmazonTranslateClient();

            // The Job Id is generated when the text translation job is started
            // with a call to the StartTextTranslationJob method.
            var jobId = "1234567890abcdef01234567890abcde";

            var request = new DescribeTextTranslationJobRequest
            {
                JobId = jobId,
            };

            var jobProperties = await DescribeTranslationJobAsync(client,
 request);

            DisplayTranslationJobDetails(jobProperties);
        }

        /// <summary>
        /// Retrieve information about an Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client
 object.</param>
        /// <param name="request">The DescribeTextTranslationJobRequest object.</
param>
        /// <returns>The TextTranslationJobProperties object containing
        /// information about the text translation job..</returns>
        public static async Task<TextTranslationJobProperties>
 DescribeTranslationJobAsync(
            AmazonTranslateClient client,
            DescribeTextTranslationJobRequest request)
```

```
        {
            var response = await client.DescribeTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                return response.TextTranslationJobProperties;
            }
            else
            {
                return null;
            }
        }

        /// <summary>
        /// Displays the properties of the text translation job.
        /// </summary>
        /// <param name="jobProperties">The properties of the text translation
        /// job returned by the call to DescribeTextTranslationJobAsync.</param>
        public static void
DisplayTranslationJobDetails(TextTranslationJobProperties jobProperties)
        {
            if (jobProperties is null)
            {
                Console.WriteLine("No text translation job properties found.");
                return;
            }

            // Display the details of the text translation job.
            Console.WriteLine($"{jobProperties.JobId}: {jobProperties.JobName}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeTextTranslationJoba Referência AWS SDK para .NET da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Gets the properties associated with an asynchronous batch translation job."
   "Includes properties such as name, ID, status, source and target languages,
and input/output Amazon Simple Storage Service (Amazon S3) buckets."
   TRY.
       oo_result = lo_xl8->describetexttranslationjob(      "oo_result is
returned for testing purposes."
         iv_jobid        = iv_jobid ).
       MESSAGE 'Job description retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE
'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeTextTranslationJobreferência da API AWS SDK for SAP ABAP.

Use **`ListTextTranslationJobs`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `ListTextTranslationJobs`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// List Amazon Translate translation jobs, along with details about each
job.
/// </summary>
public class ListTranslationJobs
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
        var filter = new TextTranslationJobFilter
        {
            JobStatus = "COMPLETED",
        };

        var request = new ListTextTranslationJobsRequest
        {
            MaxResults = 10,
            Filter = filter,
        };

        await ListJobsAsync(client, request);
    }

    /// <summary>
    /// List Amazon Translate text translation jobs.
    /// </summary>
```

```
        /// <param name="client">The initialized Amazon Translate client
object.</param>
        /// <param name="request">An Amazon Translate
        /// ListTextTranslationJobsRequest object detailing which text
        /// translation jobs are of interest.</param>
        public static async Task ListJobsAsync(
            AmazonTranslateClient client,
            ListTextTranslationJobsRequest request)
        {
            ListTextTranslationJobsResponse response;

            do
            {
                response = await client.ListTextTranslationJobsAsync(request);

ShowTranslationJobDetails(response.TextTranslationJobPropertiesList);

                request.NextToken = response.NextToken;
            }
            while (response.NextToken is not null);
        }

        /// <summary>
        /// List existing translation job details.
        /// </summary>
        /// <param name="properties">A list of Amazon Translate text
        /// translation jobs.</param>
        public static void
ShowTranslationJobDetails(List<TextTranslationJobProperties> properties)
        {
            properties.ForEach(prop =>
            {
                Console.WriteLine($"{prop.JobId}: {prop.JobName}");
                Console.WriteLine($"Status: {prop.JobStatus}");
                Console.WriteLine($"Submitted time: {prop.SubmittedTime}");
            });
        }
    }
```

- Para obter detalhes da API, consulte ListTextTranslationJobsa Referência AWS SDK para .NET da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Gets a list of the batch translation jobs that you have submitted."

   DATA lo_filter TYPE REF TO /aws1/cl_xl8textxlationjobfilt.

   "Create an ABAP object for filtering using jobname."
   lo_filter = NEW #( iv_jobname = iv_jobname ).

   TRY.
       oo_result = lo_xl8->listtexttranslationjobs(      "oo_result is returned
for testing purposes."
         io_filter        = lo_filter ).
       MESSAGE 'Jobs retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE
'E'.
     CATCH /aws1/cx_xl8invalidfilterex.
       MESSAGE 'The filter specified for the operation is not valid. Specify a
different filter.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListTextTranslationJobsreferência da API AWS SDK for SAP ABAP.

Use **`StartTextTranslationJob`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StartTextTranslationJob`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de trabalhos de tradução

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   using System;
   using System.Collections.Generic;
   using System.Threading.Tasks;
   using Amazon.Translate;
   using Amazon.Translate.Model;

   /// <summary>
   /// This example shows how to use Amazon Translate to process the files in
   /// an Amazon Simple Storage Service (Amazon S3) bucket. The translated
results
   /// will also be stored in an Amazon S3 bucket.
   /// </summary>
   public class BatchTranslate
   {
       public static async Task Main()
       {
           var contentType = "text/plain";

           // Set this variable to an S3 bucket location with a folder."
           // Input files must be in a folder and not at the bucket root."
           var s3InputUri = "s3://amzn-s3-demo-bucket1/FOLDER/";
           var s3OutputUri = "s3://amzn-s3-demo-bucket2/";
```

```
            // This role must have permissions to read the source bucket and to
 read and
            // write to the destination bucket where the translated text will be
 stored.
            var dataAccessRoleArn = "arn:aws:iam::0123456789ab:role/
S3TranslateRole";

            var client = new AmazonTranslateClient();

            var inputConfig = new InputDataConfig
            {
                ContentType = contentType,
                S3Uri = s3InputUri,
            };

            var outputConfig = new OutputDataConfig
            {
                S3Uri = s3OutputUri,
            };

            var request = new StartTextTranslationJobRequest
            {
                JobName = "ExampleTranslationJob",
                DataAccessRoleArn = dataAccessRoleArn,
                InputDataConfig = inputConfig,
                OutputDataConfig = outputConfig,
                SourceLanguageCode = "en",
                TargetLanguageCodes = new List<string> { "fr" },
            };

            var response = await StartTextTranslationAsync(client, request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId}: {response.JobStatus}");
            }
        }

        /// <summary>
        /// Start the Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized AmazonTranslateClient object.</
param>
        /// <param name="request">The request object that includes details such
```

```
        /// as source and destination bucket names and the IAM Role that will
        /// be used to access the buckets.</param>
        /// <returns>The StartTextTranslationResponse object that includes the
        /// details of the request response.</returns>
        public static async Task<StartTextTranslationJobResponse>
 StartTextTranslationAsync(AmazonTranslateClient client,
 StartTextTranslationJobRequest request)
        {
            var response = await client.StartTextTranslationJobAsync(request);
            return response;
        }
    }
```

- Para obter detalhes da API, consulte StartTextTranslationJoba Referência AWS SDK para .NET da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Starts an asynchronous batch translation job."
    "Use batch translation jobs to translate large volumes of text across
 multiple documents at once."

    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    lo_inputdataconfig = NEW #( iv_s3uri = iv_input_data_s3uri
                                iv_contenttype = iv_input_data_contenttype ).
```

```
   "Create an ABAP object for the output data config."
   lo_outputdataconfig = NEW #( iv_s3uri = iv_output_data_s3uri ).

   "Create an internal table for target languages."
   lo_targetlanguagecodes = NEW #( iv_value = iv_targetlanguagecode ).
   INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

   TRY.
       oo_result = lo_xl8->starttexttranslationjob(      "oo_result is returned
for testing purposes."
         io_inputdataconfig = lo_inputdataconfig
           io_outputdataconfig = lo_outputdataconfig
           it_targetlanguagecodes = lt_targetlanguagecodes
           iv_dataaccessrolearn = iv_dataaccessrolearn
           iv_jobname = iv_jobname
           iv_sourcelanguagecode = iv_sourcelanguagecode ).
       MESSAGE 'Translation job started.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE
'E'.
     CATCH /aws1/cx_xl8invparamvalueex.
       MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
     CATCH /aws1/cx_xl8unsuppedlanguage00.
       MESSAGE 'Amazon Translate does not support translation from the language
of the source text into the requested target language.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a StartTextTranslationJobreferência da API AWS SDK for SAP ABAP.

Use **`StopTextTranslationJob`**com um AWS SDK

Os exemplos de código a seguir mostram como usar o `StopTextTranslationJob`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// Shows how to stop a running Amazon Translation Service text translation
/// job.
/// </summary>
public class StopTextTranslationJob
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new StopTextTranslationJobRequest
        {
            JobId = jobId,
        };

        await StopTranslationJobAsync(client, request);
    }

    /// <summary>
    /// Sends a request to stop a text translation job.
    /// </summary>
    /// <param name="client">Initialized AmazonTrnslateClient object.</param>
    /// <param name="request">The request object to be passed to the
    /// StopTextJobAsync method.</param>
    public static async Task StopTranslationJobAsync(
        AmazonTranslateClient client,
```

```
            StopTextTranslationJobRequest request)
        {
            var response = await client.StopTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId} as status:
{response.JobStatus}");
            }
        }
    }
```

- Para obter detalhes da API, consulte StopTextTranslationJoba Referência AWS SDK para .NET da API.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Stops an asynchronous batch translation job that is in progress."

    TRY.
        oo_result = lo_xl8->stoptexttranslationjob(      "oo_result is returned
for testing purposes."
          iv_jobid        = iv_jobid ).
        MESSAGE 'Translation job stopped.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StopTextTranslationJobreferência da API AWS SDK for SAP ABAP.

Use **`TranslateText`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `TranslateText`.

.NET

SDK para .NET

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// Take text from a file stored a Amazon Simple Storage Service (Amazon S3)
/// object and translate it using the Amazon Transfer Service.
/// </summary>
public class TranslateText
{
    public static async Task Main()
    {
        // If the region you want to use is different from the region
        // defined for the default user, supply it as a parameter to the
        // Amazon Translate client object constructor.
        var client = new AmazonTranslateClient();

        // Set the source language to "auto" to request Amazon Translate to
```

```
            // automatically detect te language of the source text.

            // You can get a list of the languages supposed by Amazon Translate
            // in the Amazon Translate Developer's Guide here:
            //      https://docs.aws.amazon.com/translate/latest/dg/what-is.html
            string srcLang = "en"; // English.
            string destLang = "fr"; // French.

            // The Amazon Simple Storage Service (Amazon S3) bucket where the
            // source text file is stored.
            string srcBucket = "amzn-s3-demo-bucket";
            string srcTextFile = "source.txt";

            var srcText = await GetSourceTextAsync(srcBucket, srcTextFile);
            var destText = await TranslatingTextAsync(client, srcLang, destLang,
srcText);

            ShowText(srcText, destText);
        }

        /// <summary>
        /// Use the Amazon S3 TransferUtility to retrieve the text to translate
        /// from an object in an S3 bucket.
        /// </summary>
        /// <param name="srcBucket">The name of the S3 bucket where the
        /// text is stored.
        /// </param>
        /// <param name="srcTextFile">The key of the S3 object that
        /// contains the text to translate.</param>
        /// <returns>A string representing the source text.</returns>
        public static async Task<string> GetSourceTextAsync(string srcBucket,
string srcTextFile)
        {
            string srcText = string.Empty;

            var s3Client = new AmazonS3Client();
            TransferUtility utility = new TransferUtility(s3Client);

            using var stream = await utility.OpenStreamAsync(srcBucket,
srcTextFile);

            StreamReader file = new System.IO.StreamReader(stream);

            srcText = file.ReadToEnd();
```

```
            return srcText;
        }

        /// <summary>
        /// Use the Amazon Translate Service to translate the document from the
        /// source language to the specified destination language.
        /// </summary>
        /// <param name="client">The Amazon Translate Service client used to
        /// perform the translation.</param>
        /// <param name="srcLang">The language of the source text.</param>
        /// <param name="destLang">The destination language for the translated
        /// text.</param>
        /// <param name="text">A string representing the text to ranslate.</
param>
        /// <returns>The text that has been translated to the destination
        /// language.</returns>
        public static async Task<string>
 TranslatingTextAsync(AmazonTranslateClient client, string srcLang, string
 destLang, string text)
        {
            var request = new TranslateTextRequest
            {
                SourceLanguageCode = srcLang,
                TargetLanguageCode = destLang,
                Text = text,
            };

            var response = await client.TranslateTextAsync(request);

            return response.TranslatedText;
        }

        /// <summary>
        /// Show the original text followed by the translated text.
        /// </summary>
        /// <param name="srcText">The original text to be translated.</param>
        /// <param name="destText">The translated text.</param>
        public static void ShowText(string srcText, string destText)
        {
            Console.WriteLine("Source text:");
            Console.WriteLine(srcText);
            Console.WriteLine();
            Console.WriteLine("Translated text:");
            Console.WriteLine(destText);
```

```
        }
    }
```

- Para obter detalhes da API, consulte TranslateTexta Referência AWS SDK para .NET da API.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: converte o texto em inglês especificado em francês. O texto a ser convertido também pode ser passado como o parâmetro -Text.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- Para obter detalhes da API, consulte TranslateTextem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: converte o texto em inglês especificado em francês. O texto a ser convertido também pode ser passado como o parâmetro -Text.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- Para obter detalhes da API, consulte TranslateTextem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Translates input text from the source language to the target language."
   TRY.
       oo_result = lo_xl8->translatetext(      "oo_result is returned for
testing purposes."
         iv_text        = iv_text
           iv_sourcelanguagecode = iv_sourcelanguagecode
           iv_targetlanguagecode = iv_targetlanguagecode ).
       MESSAGE 'Translation completed.' TYPE 'I'.
     CATCH /aws1/cx_xl8detectedlanguage00.
       MESSAGE 'The confidence that Amazon Comprehend accurately detected the
source language is low.' TYPE 'E'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8serviceunavailex.
       MESSAGE 'The Amazon Translate service is temporarily unavailable.' TYPE
'E'.
     CATCH /aws1/cx_xl8textsizelmtexcdex.
       MESSAGE 'The size of the text you submitted exceeds the size limit. '
TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
     CATCH /aws1/cx_xl8unsuppedlanguage00.
       MESSAGE 'Amazon Translate does not support translation from the language
of the source text into the requested target language. ' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a TranslateTextreferência da API AWS SDK for SAP ABAP.

Cenários para o Amazon Translate usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon Translate com AWS SDKs. Esses casos mostram como executar tarefas específicas chamando várias funções no Amazon Translate ou em combinação com outros Serviços da AWS. Cada cenário inclui um link

para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criar uma aplicação de transmissão do Amazon Transcribe
- Crie um chatbot do Amazon Lex para engajar os visitantes do site
- Criar uma aplicação de publicação e assinatura que traduz mensagens
- Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio
- Comece a usar trabalhos do Amazon Translate usando um AWS SDK

Criar uma aplicação de transmissão do Amazon Transcribe

O exemplo de código a seguir mostra como construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Transcribe para construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Crie um chatbot do Amazon Lex para engajar os visitantes do site

Os exemplos de código a seguir mostram como criar um chatbot para engajar os visitantes do seu site.

Java

SDK para Java 2.x

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o exemplo completo Criando um chatbot Amazon Lex no guia do AWS SDK para JavaScript desenvolvedor.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Criar uma aplicação de publicação e assinatura que traduz mensagens

Os exemplos de código a seguir mostram como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo que usa a API Java Async, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Kotlin

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon SNS para criar uma aplicação com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como criar um aplicativo web, veja o exemplo completo em GitHub.

Para ver o código-fonte completo e instruções sobre como criar um aplicativo Android nativo, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Criar uma aplicação que analise o feedback dos clientes e sintetize o áudio

Os exemplos de código a seguir mostram como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

.NET

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado nas funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );
```

```
  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
```

```
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
```

```
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Comece a usar trabalhos do Amazon Translate usando um AWS SDK

O exemplo de código a seguir mostra como:

- Iniciar um trabalho assíncrono de tradução em lote.
- Aguardar a conclusão do trabalho assíncrono.
- Descrever o trabalho assíncrono.

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    lo_inputdataconfig = NEW #( iv_s3uri = iv_input_data_s3uri
                                iv_contenttype = iv_input_data_contenttype ).

    "Create an ABAP object for the output data config."
    lo_outputdataconfig = NEW #( iv_s3uri = iv_output_data_s3uri ).

    "Create an internal table for target languages."
    lo_targetlanguagecodes = NEW #( iv_value = iv_targetlanguagecode ).
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
        DATA(lo_translationjob_result) = lo_xl8->starttexttranslationjob(
          io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode ).
        MESSAGE 'Translation job started.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE
 'E'.
      CATCH /aws1/cx_xl8invparamvalueex.
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
```

```
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time. '
 TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00.
        MESSAGE 'Amazon Translate does not support translation from the language
 of the source text into the requested target language.' TYPE 'E'.
    ENDTRY.

    "Get the job ID."
    DATA(lv_jobid) = lo_translationjob_result->get_jobid( ).

    "Wait for translate job to complete."
    DATA(lo_des_translation_result) = lo_xl8-
>describetexttranslationjob( iv_jobid = lv_jobid ).
    WHILE lo_des_translation_result->get_textxlationjobproperties( )-
>get_jobstatus( ) <> 'COMPLETED'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_des_translation_result = lo_xl8->describetexttranslationjob( iv_jobid =
 lv_jobid ).
    ENDWHILE.

    TRY.
        oo_result = lo_xl8->describetexttranslationjob(      "oo_result is
 returned for testing purposes."
          iv_jobid        = lv_jobid ).
        MESSAGE 'Job description retrieved.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE
 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
 TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - DescribeTextTranslationJob
  - StartTextTranslationJob

Exemplos de código para VPC Lattice usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon VPC Lattice com um kit de desenvolvimento de AWS software (SDK).

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- Guia do desenvolvedor do VPC Lattice — Mais informações sobre o VPC Lattice.
- Referência da API VPC Lattice — Detalhes sobre todas as ações disponíveis do VPC Lattice.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Cenários para VPC Lattice usando AWS SDKs
  - Criação e gerenciamento de uma malha de rede de serviços

Cenários para VPC Lattice usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no VPC Lattice com SDKs. AWS Esses cenários mostram como realizar tarefas específicas chamando várias funções no VPC Lattice ou combinadas com outras. Serviços da AWS Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criação e gerenciamento de uma malha de rede de serviços

Criação e gerenciamento de uma malha de rede de serviços

O exemplo de código a seguir mostra como:

- Criar uma rede de serviços.
- Criar um serviço.
- Listar as VPCs disponíveis.
- Listar grupos de segurança para a VPC selecionada.
- Listar associações de serviço.
- Listar associações de VPC.
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC Lattice Service Network Tutorial Script
# This script demonstrates how to create and manage a VPC Lattice service network

set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="vpc-lattice-tutorial.log"
touch "$LOG_FILE"
```

```
chmod 600 "$LOG_FILE"
echo "Starting VPC Lattice tutorial script at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_command() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    if [ "$1" -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        exit "$1"
    fi
}

# Function to validate AWS CLI is available
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" | tee -a
 "$LOG_FILE"
        exit 1
    fi
}

# Function to validate input parameters
validate_input() {
    local input="$1"
    local param_name="$2"

    if [[ -z "$input" ]]; then
        echo "ERROR: $param_name is empty" | tee -a "$LOG_FILE"
        return 1
    fi

    # Validate against common injection patterns
    if [[ "$input" =~ [\;\$\`\|\&\<\>\(\)\{\}] ]]; then
        echo "ERROR: $param_name contains invalid characters" | tee -a
 "$LOG_FILE"
        return 1
```

```
    fi

    return 0
}

# Function to wait for a resource to be in the desired state
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
    local desired_status="$3"
    local command="$4"
    local max_attempts=30
    local attempt=1
    local status=""

    validate_input "$resource_type" "resource_type" || return 1
    validate_input "$resource_id" "resource_id" || return 1
    validate_input "$desired_status" "desired_status" || return 1

    echo "Waiting for $resource_type $resource_id to be in state
 $desired_status..." | tee -a "$LOG_FILE"

    while [ "$attempt" -le "$max_attempts" ]; do
        echo "Attempt $attempt of $max_attempts..." >> "$LOG_FILE"

        # Run the command to get the status and capture the output
        status_output=$(eval "$command" 2>&1) || true
        echo "$status_output" >> "$LOG_FILE"

        # For service networks, they do not have a status field in the output
        # We'll consider them active if we can retrieve them
        if [[ "$resource_type" == "Service Network" ]]; then
            if [[ "$status_output" == *"$resource_id"* ]]; then
                echo "$resource_type $resource_id is now active" | tee -a
 "$LOG_FILE"
                return 0
            fi
        else
            # For other resources, extract the status field
            status=$(echo "$status_output" | grep -i "status" | awk -F'"' '{print
 $4}' || true)
            echo "Current status: $status" >> "$LOG_FILE"

            if [[ "$status" == "$desired_status" ]]; then
```

```
                echo "$resource_type $resource_id is now in state
 $desired_status" | tee -a "$LOG_FILE"
                return 0
            elif [[ "$status" == *"FAIL"* ]]; then
                echo "ERROR: $resource_type $resource_id failed to reach desired
 state. Current status: $status" | tee -a "$LOG_FILE"
                return 1
            fi
        fi

        echo "Waiting for status change... (attempt $attempt/$max_attempts)" >>
 "$LOG_FILE"
        sleep 10
        ((attempt++))
    done

    echo "ERROR: Timed out waiting for $resource_type $resource_id to reach state
 $desired_status" | tee -a "$LOG_FILE"
    return 1
}

# Cleanup function for trap
cleanup() {
    local exit_code=$?
    echo "Script interrupted or failed. Cleaning up..." | tee -a "$LOG_FILE"
    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Check prerequisites
check_aws_cli

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
SERVICE_NETWORK_NAME="lattice-network-${RANDOM_ID}"
SERVICE_NAME="lattice-service-${RANDOM_ID}"

# Store created resources for cleanup
declare -a CREATED_RESOURCES

echo "=== VPC Lattice Service Network Tutorial ===" | tee -a "$LOG_FILE"
echo "Random ID for this session: ${RANDOM_ID}" | tee -a "$LOG_FILE"
```

```
# Step 1: Create a VPC Lattice service network
echo -e "\n=== Step 1: Creating a VPC Lattice service network ===" | tee -a
 "$LOG_FILE"
echo "Creating service network: $SERVICE_NETWORK_NAME" | tee -a "$LOG_FILE"

SERVICE_NETWORK_OUTPUT=$(log_command "aws vpc-lattice create-service-
network --name '$SERVICE_NETWORK_NAME' --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-vpc-lattice-gs --output json")
check_error $?

# Extract the service network ID using jq for safety
SERVICE_NETWORK_ID=$(echo "$SERVICE_NETWORK_OUTPUT" | jq -r '.id // empty' 2>/
dev/null || true)
if [ -z "$SERVICE_NETWORK_ID" ]; then
    echo "ERROR: Failed to extract service network ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_NETWORK_ID" "SERVICE_NETWORK_ID" || exit 1

echo "Service network created with ID: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Network: $SERVICE_NETWORK_ID")

# Wait for the service network to be active
wait_for_resource "Service Network" "$SERVICE_NETWORK_ID" "ACTIVE" "aws vpc-
lattice get-service-network --service-network-identifier '$SERVICE_NETWORK_ID' --
output json"
check_error $?

# Step 2: Create a VPC Lattice service
echo -e "\n=== Step 2: Creating a VPC Lattice service ===" | tee -a "$LOG_FILE"
echo "Creating service: $SERVICE_NAME" | tee -a "$LOG_FILE"

SERVICE_OUTPUT=$(log_command "aws vpc-lattice create-service --name
 '$SERVICE_NAME' --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-
vpc-lattice-gs --output json")
check_error $?

# Extract the service ID using jq for safety
SERVICE_ID=$(echo "$SERVICE_OUTPUT" | jq -r '.id // empty' 2>/dev/null || true)
if [ -z "$SERVICE_ID" ]; then
    echo "ERROR: Failed to extract service ID" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
validate_input "$SERVICE_ID" "SERVICE_ID" || exit 1

echo "Service created with ID: $SERVICE_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service: $SERVICE_ID")

# Wait for the service to be active
wait_for_resource "Service" "$SERVICE_ID" "ACTIVE" "aws vpc-lattice get-service
 --service-identifier '$SERVICE_ID' --output json"
check_error $?

# Step 3: Associate the service with the service network
echo -e "\n=== Step 3: Associating service with service network ===" | tee -a
 "$LOG_FILE"

SERVICE_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
service-association --service-identifier '$SERVICE_ID' --service-network-
identifier '$SERVICE_NETWORK_ID' --output json")
check_error $?

# Extract the service association ID using jq for safety
SERVICE_ASSOC_ID=$(echo "$SERVICE_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
if [ -z "$SERVICE_ASSOC_ID" ]; then
    echo "ERROR: Failed to extract service association ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ASSOC_ID" "SERVICE_ASSOC_ID" || exit 1

echo "Service association created with ID: $SERVICE_ASSOC_ID" | tee -a
 "$LOG_FILE"
CREATED_RESOURCES+=("Service Association: $SERVICE_ASSOC_ID")

# Wait for the service association to be active
wait_for_resource "Service Association" "$SERVICE_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-service-association --service-network-service-
association-identifier '$SERVICE_ASSOC_ID' --output json"
check_error $?

# Step 4: List available VPCs to associate with the service network
echo -e "\n=== Step 4: Listing available VPCs ===" | tee -a "$LOG_FILE"
```

```
VPC_LIST=$(log_command "aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId,Tags[?Key==
\`Name\`].Value|[0]]' --output text")
check_error $?

echo "Available VPCs:" | tee -a "$LOG_FILE"
echo "$VPC_LIST" | tee -a "$LOG_FILE"

# Step 5: Auto-select first available VPC
echo -e "\n=== Step 5: Associate a VPC with the service network ===" | tee -a
 "$LOG_FILE"

VPC_ID=$(echo "$VPC_LIST" | head -n 1 | awk '{print $1}')

if [ -z "$VPC_ID" ]; then
    echo "WARNING: No VPC ID found" | tee -a "$LOG_FILE"
    echo "Skipping VPC association step" | tee -a "$LOG_FILE"
else
    validate_input "$VPC_ID" "VPC_ID" || {
        echo "ERROR: VPC_ID validation failed"
        exit 1
    }

    echo "Auto-selected VPC: $VPC_ID" | tee -a "$LOG_FILE"

    # Step 6: List security groups for the selected VPC
    echo -e "\n=== Step 6: Listing security groups for VPC $VPC_ID ===" | tee -a
 "$LOG_FILE"

    SG_LIST=$(log_command "aws ec2 describe-security-groups --filters Name=vpc-
id,Values='$VPC_ID' --query 'SecurityGroups[*].[GroupId,GroupName]' --output
 text")
    check_error $?

    echo "Available Security Groups for VPC $VPC_ID:" | tee -a "$LOG_FILE"
    echo "$SG_LIST" | tee -a "$LOG_FILE"

    # Step 7: Auto-select first available security group
    echo -e "\n=== Step 7: Select a security group for the VPC association ===" |
 tee -a "$LOG_FILE"

    SG_ID=$(echo "$SG_LIST" | head -n 1 | awk '{print $1}')

    if [ -z "$SG_ID" ]; then
        echo "WARNING: No Security Group ID found" | tee -a "$LOG_FILE"
```

```
        echo "Skipping VPC association step" | tee -a "$LOG_FILE"
    else
        validate_input "$SG_ID" "SG_ID" || {
            echo "ERROR: SG_ID validation failed"
            exit 1
        }

        echo "Auto-selected Security Group: $SG_ID" | tee -a "$LOG_FILE"

        # Step 8: Associate the VPC with the service network
        echo -e "\n=== Step 8: Associating VPC with service network ===" | tee -a
 "$LOG_FILE"

        VPC_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
vpc-association --vpc-identifier '$VPC_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --security-group-ids '$SG_ID' --output json")
        check_error $?

        # Extract the VPC association ID using jq for safety
        VPC_ASSOC_ID=$(echo "$VPC_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
        if [ -z "$VPC_ASSOC_ID" ]; then
            echo "ERROR: Failed to extract VPC association ID" | tee -a
 "$LOG_FILE"
        else
            validate_input "$VPC_ASSOC_ID" "VPC_ASSOC_ID" || exit 1

            echo "VPC association created with ID: $VPC_ASSOC_ID" | tee -a
 "$LOG_FILE"
            CREATED_RESOURCES+=("VPC Association: $VPC_ASSOC_ID")

            # Wait for the VPC association to be active
            wait_for_resource "VPC Association" "$VPC_ASSOC_ID" "ACTIVE"
 "aws vpc-lattice get-service-network-vpc-association --service-network-vpc-
association-identifier '$VPC_ASSOC_ID' --output json"
            check_error $?
        fi
    fi
fi

# Step 9: Display information about the created resources
echo -e "\n=== Step 9: Displaying information about created resources ===" | tee
 -a "$LOG_FILE"
```

```
echo "Service Network Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json"

echo "Service Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service --service-identifier '$SERVICE_ID' --
output json"

echo "Service Network Service Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-service-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

echo "Service Network VPC Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-vpc-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

# Step 10: Cleanup - Auto-confirm
echo -e "\n=== Step 10: Resource Cleanup ===" | tee -a "$LOG_FILE"
echo "Resources created in this tutorial:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
done

echo "Starting cleanup process..." | tee -a "$LOG_FILE"

# Delete resources in reverse order

# Delete VPC association if it was created
if [[ -n "${VPC_ASSOC_ID:-}" ]]; then
    echo "Deleting VPC association: $VPC_ASSOC_ID" | tee -a "$LOG_FILE"
    log_command "aws vpc-lattice delete-service-network-vpc-association --
service-network-vpc-association-identifier '$VPC_ASSOC_ID'" || true

    # Wait for the VPC association to be deleted
    echo "Waiting for VPC association to be deleted..." | tee -a "$LOG_FILE"
    sleep 30
fi

# Delete service association
echo "Deleting service association: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network-service-association --
service-network-service-association-identifier '$SERVICE_ASSOC_ID'" || true

# Wait for the service association to be deleted
```

```
echo "Waiting for service association to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service
echo "Deleting service: $SERVICE_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service --service-identifier '$SERVICE_ID'"
 || true

# Wait for the service to be deleted
echo "Waiting for service to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service network
echo "Deleting service network: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID'" || true

echo "Cleanup completed successfully!" | tee -a "$LOG_FILE"

echo -e "\n=== Tutorial completed! ===" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateService
  - CreateServiceNetwork
  - CreateServiceNetworkServiceAssociation
  - CreateServiceNetworkVpcAssociation
  - DeleteService
  - DeleteServiceNetwork
  - DeleteServiceNetworkServiceAssociation
  - DeleteServiceNetworkVpcAssociation
  - DescribeSecurityGroups
  - DescribeVpcs
  - GetService
  - GetServiceNetwork
  - GetServiceNetworkServiceAssociation
  - GetServiceNetworkVpcAssociation

- ListServiceNetworkServiceAssociations
- ListServiceNetworkVpcAssociations

Exemplos de código para AWS WAFV2 usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS WAFV2 com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- AWS WAFV2 Guia do desenvolvedor — Mais informações sobre AWS WAFV2.
- AWS WAFV2 Referência da API — Detalhes sobre todas as AWS WAFV2 ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos para AWS WAFV2 usar AWS SDKs
  - Ações para AWS WAFV2 usar AWS SDKs
    - Usar CreateWebAcl com uma CLI
- Cenários para AWS WAFV2 usar AWS SDKs
  - Conceitos básicos de firewalls do aplicativo Web

Exemplos básicos para AWS WAFV2 usar AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS WAFV2 com AWS SDKs.

Exemplos

- Ações para AWS WAFV2 usar AWS SDKs
  - Usar CreateWebAcl com uma CLI

Ações para AWS WAFV2 usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS WAFV2 ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a AWS WAFV2 API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para AWS WAFV2 usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS WAFV2.

Exemplos

- Usar CreateWebAcl com uma CLI

Usar **`CreateWebAcl`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateWebAcl`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Conceitos básicos de firewalls do aplicativo Web

CLI

AWS CLI

Para criar uma ACL da web

O comando `create-web-acl` a seguir cria uma ACL da web para uso regional. As instruções de regra para a Web ACL são fornecidas em um JSON-formatted arquivo.

```
aws wafv2 create-web-acl \
    --name TestWebAcl \
```

```
    --scope REGIONAL \
    --default-action Allow={} \
    --visibility-
config SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=TestWebAclMet
 \
    --rules file://waf-rule.json \
    --region us-west-2
```

Conteúdo de file://waf-rule.json:

```
[
    {
        "Name":"basic-rule",
        "Priority":0,
        "Statement":{
            "AndStatement":{
                "Statements":[
                    {
                        "ByteMatchStatement":{
                            "SearchString":"example.com",
                            "FieldToMatch":{
                                "SingleHeader":{
                                    "Name":"host"
                                }
                            },
                            "TextTransformations":[
                                {
                                    "Priority":0,
                                    "Type":"LOWERCASE"
                                }
                            ],
                            "PositionalConstraint":"EXACTLY"
                        }
                    },
                    {
                        "GeoMatchStatement":{
                            "CountryCodes":[
                                "US",
                                "IN"
                            ]
                        }
                    }
                ]
```

```
            }
        },
        "Action":{
            "Allow":{

            }
        },
        "VisibilityConfig":{
            "SampledRequestsEnabled":true,
            "CloudWatchMetricsEnabled":true,
            "MetricName":"basic-rule"
        }
    }
]
```

Saída:

```
{
    "Summary":{
        "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/webacl/TestWebAcl/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "Description":"",
        "Name":"TestWebAcl",
        "LockToken":"2294b3a1-eb60-4aa0-a86f-a3ae04329de9",
        "Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    }
}
```

Para obter mais informações, consulte Gerenciando e usando uma lista de controle de acesso à Web (Web ACL) no AWS WAF, no AWS Firewall Manager e no AWS Shield Advanced Developer Guide.

- Para obter detalhes da API, consulte CreateWebAclem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse comando cria uma ACL da web chamada “waf-test”. Observe que, de acordo com a documentação da API de serviço, 'DefaultAction' é uma propriedade obrigatória. Portanto, o valor de '- DefaultAction _Allow' '- and/or DefaultAction _Block' deve ser especificado. Como '- DefaultAction _Allow' e '- DefaultAction _Block 'não são as propriedades

obrigatórias, o valor '@ {}' pode ser usado como espaço reservado, conforme mostrado no exemplo acima.

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-
west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -
VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-
test" -Description "Test" -DefaultAction_Allow @{}
```

Saída:

```
ARN         : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-
test/19460b3f-db14-4b9a-8e23-a417e1eb007f
Description : Test
Id          : 19460b3f-db14-4b9a-8e23-a417e1eb007f
LockToken   : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab
Name        : waf-test
```

- Para obter detalhes da API, consulte CreateWebAclem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse comando cria uma ACL da web chamada “waf-test”. Observe que, de acordo com a documentação da API de serviço, 'DefaultAction' é uma propriedade obrigatória. Portanto, o valor de '- DefaultAction _Allow' '- and/or DefaultAction _Block' deve ser especificado. Como '- DefaultAction _Allow' e '- DefaultAction _Block 'não são as propriedades obrigatórias, o valor '@ {}' pode ser usado como espaço reservado, conforme mostrado no exemplo acima.

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-
west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -
VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-
test" -Description "Test" -DefaultAction_Allow @{}
```

Saída:

```
ARN         : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-
test/19460b3f-db14-4b9a-8e23-a417e1eb007f
Description : Test
Id          : 19460b3f-db14-4b9a-8e23-a417e1eb007f
```

```
LockToken   : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab
Name        : waf-test
```

- Para obter detalhes da API, consulte CreateWebAclem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para AWS WAFV2 usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns AWS WAFV2 com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas AWS WAFV2 ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Conceitos básicos de firewalls do aplicativo Web

Conceitos básicos de firewalls do aplicativo Web

O exemplo de código a seguir mostra como:

- Criar uma ACL da web.
- Adicionar uma regra de correspondência de string.
- Adicionar regras gerenciadas.
- Configurar o registro em log da
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
```

```
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
```

```
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken'
 2>/dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the
 AWS Console."
```

```
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi
```

```
# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "====================================================="
echo "STEP 2: Adding String Match Rule"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
```

```
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
```

```
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
```

```
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
```

```
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
```

```
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "====================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "====================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "====================================================="
```

```
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "====================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo
 "$first_dist" | grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution
 successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
```

```
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

Exemplos de código para WorkSpaces usar AWS SDKs

Os exemplos de código a seguir mostram como usar a Amazon WorkSpaces com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Mais atributos

- WorkSpaces Guia do desenvolvedor — Mais informações sobre WorkSpaces.
- WorkSpaces Referência da API — Detalhes sobre todas as WorkSpaces ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de WorkSpaces uso AWS SDKs
  - Ações para WorkSpaces usar AWS SDKs
    - Usar CreateTags com uma CLI
    - Usar CreateWorkspaces com uma CLI
    - Usar DeleteTags com uma CLI
    - Usar DescribeTags com uma CLI
    - Usar DescribeWorkspaceBundles com uma CLI
    - Usar DescribeWorkspaceDirectories com uma CLI
    - Usar DescribeWorkspaces com uma CLI
    - Usar DescribeWorkspacesConnectionStatus com uma CLI
    - Usar ModifyWorkspaceCreationProperties com uma CLI
    - Usar ModifyWorkspaceProperties com uma CLI
    - Usar ModifyWorkspaceState com uma CLI
    - Usar RebootWorkspaces com uma CLI
    - Usar RebuildWorkspaces com uma CLI
    - Usar RegisterWorkspaceDirectory com uma CLI

    - Usar StopWorkspaces com uma CLI
    - Usar TerminateWorkspaces com uma CLI
- Cenários para WorkSpaces usar AWS SDKs
  - Criação e gerenciamento de áreas de trabalho virtuais pessoais

Exemplos básicos de WorkSpaces uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico da Amazon WorkSpaces com AWS SDKs.

Exemplos

- Ações para WorkSpaces usar AWS SDKs
  - Usar CreateTags com uma CLI
  - Usar CreateWorkspaces com uma CLI
  - Usar DeleteTags com uma CLI
  - Usar DescribeTags com uma CLI
  - Usar DescribeWorkspaceBundles com uma CLI
  - Usar DescribeWorkspaceDirectories com uma CLI
  - Usar DescribeWorkspaces com uma CLI
  - Usar DescribeWorkspacesConnectionStatus com uma CLI
  - Usar ModifyWorkspaceCreationProperties com uma CLI
  - Usar ModifyWorkspaceProperties com uma CLI
  - Usar ModifyWorkspaceState com uma CLI
  - Usar RebootWorkspaces com uma CLI
  - Usar RebuildWorkspaces com uma CLI
  - Usar RegisterWorkspaceDirectory com uma CLI
  - Usar StopWorkspaces com uma CLI
  - Usar TerminateWorkspaces com uma CLI

Ações para WorkSpaces usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar WorkSpaces ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Esses trechos chamam a WorkSpaces API e são trechos de código de programas maiores que devem ser executados em contexto. É possível ver as ações em contexto em Cenários para WorkSpaces usar AWS SDKs.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Amazon WorkSpaces API Reference.

Exemplos

- Usar CreateTags com uma CLI
- Usar CreateWorkspaces com uma CLI
- Usar DeleteTags com uma CLI
- Usar DescribeTags com uma CLI
- Usar DescribeWorkspaceBundles com uma CLI
- Usar DescribeWorkspaceDirectories com uma CLI
- Usar DescribeWorkspaces com uma CLI
- Usar DescribeWorkspacesConnectionStatus com uma CLI
- Usar ModifyWorkspaceCreationProperties com uma CLI
- Usar ModifyWorkspaceProperties com uma CLI
- Usar ModifyWorkspaceState com uma CLI
- Usar RebootWorkspaces com uma CLI
- Usar RebuildWorkspaces com uma CLI
- Usar RegisterWorkspaceDirectory com uma CLI
- Usar StopWorkspaces com uma CLI
- Usar TerminateWorkspaces com uma CLI

Usar **`CreateTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateTags`.

CLI

AWS CLI

Para adicionar tags a um WorkSpace

O `create-tags` exemplo a seguir adiciona as tags especificadas às especificadas WorkSpace.

```
aws workspaces create-tags \
    --resource-id ws-dk1xzr417 \
    --tags Key=Department,Value=Finance
```

Este comando não produz saída.

Para obter mais informações, consulte WorkSpaces Recursos de tags no Guia de WorkSpaces administração da Amazon.

- Para obter detalhes da API, consulte CreateTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo adiciona uma nova tag a um espaço de trabalho chamado **`ws-wsname`**. A tag tem uma chave de “Name” e um valor-chave de **`AWS_Workspace`**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

Exemplo 2: esse exemplo adiciona várias tags a um espaço de trabalho chamado **`ws-wsname`**. Uma tag tem uma chave de “Name” e um valor-chave de **`AWS_Workspace`**; a outra tag tem uma chave de tag de “Stage” e um valor-chave de “Test”.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
```

```
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- Para obter detalhes da API, consulte CreateTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona uma nova tag a um espaço de trabalho chamado **`ws-wsname`**. A tag tem uma chave de “Name” e um valor-chave de **`AWS_Workspace`**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

Exemplo 2: esse exemplo adiciona várias tags a um espaço de trabalho chamado **`ws-wsname`**. Uma tag tem uma chave de “Name” e um valor-chave de **`AWS_Workspace`**; a outra tag tem uma chave de tag de “Stage” e um valor-chave de “Test”.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- Para obter detalhes da API, consulte CreateTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`CreateWorkspaces`** com uma CLI

Os exemplos de código a seguir mostram como usar o `CreateWorkspaces`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação e gerenciamento de áreas de trabalho virtuais pessoais

CLI

AWS CLI

Exemplo 1: Para criar um AlwaysOn WorkSpace

O `create-workspaces` exemplo a seguir cria um AlwaysOn WorkSpace para o usuário especificado, usando o diretório e o pacote especificados.

```
aws workspaces create-workspaces \
    --workspaces DirectoryId=d-926722edaf,UserName=Mateo,BundleId=wsb-0zsvgp8fc
```

Saída:

```
{
    "FailedRequests": [],
    "PendingRequests": [
        {
            "WorkspaceId": "ws-kcqms853t",
            "DirectoryId": "d-926722edaf",
            "UserName": "Mateo",
            "State": "PENDING",
            "BundleId": "wsb-0zsvgp8fc"
        }
    ]
}
```

Exemplo 2: Para criar um AutoStop WorkSpace

O `create-workspaces` exemplo a seguir cria um AutoStop WorkSpace para o usuário especificado, usando o diretório e o pacote especificados.

```
aws workspaces create-workspaces \
    --
workspaces DirectoryId=d-926722edaf,UserName=Mary,BundleId=wsb-0zsvgp8fc,WorkspaceProperti
```

Saída:

```
{
    "FailedRequests": [],
```

```
    "PendingRequests": [
        {
            "WorkspaceId": "ws-dk1xzr417",
            "DirectoryId": "d-926722edaf",
            "UserName": "Mary",
            "State": "PENDING",
            "BundleId": "wsb-0zsvgp8fc"
        }
    ]
}
```

Exemplo 3: Para criar um usuário desacoplado WorkSpace

O `create-workspaces` exemplo a seguir cria um usuário desacoplado WorkSpace definindo o nome de usuário como e especificando um WorkSpace nome`[UNDEFINED]`, ID do diretório e ID do pacote.

```
aws workspaces create-workspaces \
    --workspaces
 DirectoryId=d-926722edaf,UserName='"[UNDEFINED]"',WorkspaceName=MaryWorkspace1,BundleId=
```

Saída:

```
{
    "FailedRequests": [],
    "PendingRequests": [
        {
            "WorkspaceId": "ws-abcd1234",
            "DirectoryId": "d-926722edaf",
            "UserName": "[UNDEFINED]",
            "State": "PENDING",
            "BundleId": "wsb-0zsvgp8fc",
            "WorkspaceName": "MaryWorkspace1"
        }
    ]
}
```

Para obter mais informações, consulte Iniciar um desktop virtual no Amazon WorkSpaces Administration Guide.

- Para obter detalhes da API, consulte CreateWorkspacesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Crie um WorkSpace para o pacote, o diretório e o usuário fornecidos.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Exemplo 2: Este exemplo cria vários WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Para obter detalhes da API, consulte CreateWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Crie um WorkSpace para o pacote, o diretório e o usuário fornecidos.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Exemplo 2: Este exemplo cria vários WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Para obter detalhes da API, consulte CreateWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DeleteTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DeleteTags`.

CLI

AWS CLI

Para excluir uma tag de um WorkSpace

O `delete-tags` exemplo a seguir exclui a tag especificada da especificada WorkSpace.

```
aws workspaces delete-tags \
    --resource-id ws-dk1xzr417 \
    --tag-keys Department
```

Este comando não produz saída.

Para obter mais informações, consulte WorkSpaces Recursos de tags no Guia de WorkSpaces administração da Amazon.

- Para obter detalhes da API, consulte DeleteTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo remove a tag associada ao espaço de trabalho

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag associada ao espaço de trabalho

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes da API, consulte DeleteTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeTags`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeTags`.

CLI

AWS CLI

Para descrever as tags de um WorkSpace

O `describe-tags` exemplo a seguir descreve as tags para o especificado WorkSpace.

```
aws workspaces describe-tags \
    --resource-id ws-dk1xzr417
```

Saída:

```
{
    "TagList": [
        {
            "Key": "Department",
            "Value": "Finance"
        }
    ]
}
```

Para obter mais informações, consulte WorkSpaces Recursos de tags no Guia de WorkSpaces administração da Amazon.

- Para obter detalhes da API, consulte DescribeTagsem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo busca a tag para o espaço de trabalho fornecido

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Saída:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo busca a tag para o espaço de trabalho fornecido

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Saída:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- Para obter detalhes da API, consulte DescribeTagsem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeWorkspaceBundles`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeWorkspaceBundles`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação e gerenciamento de áreas de trabalho virtuais pessoais

CLI

AWS CLI

Para listar os pacotes fornecidos pela Amazon

O exemplo `describe-workspace-bundles` a seguir lista os nomes e IDs dos pacotes fornecidos pela Amazon, em formato de tabela e classificados por nome.

```
aws workspaces describe-workspace-bundles \
    --owner AMAZON \
    --query "Bundles[*].[Name, BundleId]"
```

Saída:

```
[
    [
        "Standard with Amazon Linux 2",
        "wsb-clj85qzj1"
    ],
    [
        "Performance with Windows 10 (Server 2016 based)",
        "wsb-gm4d5tx2v"
    ],
    [
        "PowerPro with Windows 7",
        "wsb-1pzkp0bx4"
    ],
    [
        "Power with Amazon Linux 2",
        "wsb-2bs6k5lgn"
    ],
    [
        "Graphics with Windows 10 (Server 2019 based)",
        "wsb-03gyjnfyy"
    ],
    ...
```

```
]
```

Para obter mais informações, consulte WorkSpaces pacotes e imagens no Guia de WorkSpaces administração da Amazon.

- Para obter detalhes da API, consulte DescribeWorkspaceBundlesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: essa amostra busca detalhes de todos os pacotes do Workspace na região atual

```
Get-WKSWorkspaceBundle
```

Saída:

```
BundleId        : wsb-sfhdgv342
ComputeType     : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId         : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name            : CustomBundleTest
Owner           : 233816212345
RootStorage     : Amazon.WorkSpaces.Model.RootStorage
UserStorage     : Amazon.WorkSpaces.Model.UserStorage
```

- Para obter detalhes da API, consulte DescribeWorkspaceBundlesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: essa amostra busca detalhes de todos os pacotes do Workspace na região atual

```
Get-WKSWorkspaceBundle
```

Saída:

```
BundleId        : wsb-sfhdgv342
ComputeType     : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
```

```
ImageId         : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name            : CustomBundleTest
Owner           : 233816212345
RootStorage     : Amazon.WorkSpaces.Model.RootStorage
UserStorage     : Amazon.WorkSpaces.Model.UserStorage
```

- Para obter detalhes da API, consulte DescribeWorkspaceBundlesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeWorkspaceDirectories`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeWorkspaceDirectories`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação e gerenciamento de áreas de trabalho virtuais pessoais

CLI

AWS CLI

Para descrever um diretório registrado

O exemplo `describe-workspace-directories` a seguir descreve o diretório registrado especificado.

```
aws workspaces describe-workspace-directories \
    --directory-ids d-926722edaf
```

Saída:

```
{
    "Directories": [
        {
            "DirectoryId": "d-926722edaf",
            "Alias": "d-926722edaf",
            "DirectoryName": "example.com",
            "RegistrationCode": "WSpdx+9RJ8JT",
            "SubnetIds": [
```

```
                "subnet-9d19c4c6",
                "subnet-500d5819"
            ],
            "DnsIpAddresses": [
                "172.16.1.140",
                "172.16.0.30"
            ],
            "CustomerUserName": "Administrator",
            "IamRoleId": "arn:aws:iam::123456789012:role/workspaces_DefaultRole",
            "DirectoryType": "SIMPLE_AD",
            "WorkspaceSecurityGroupId": "sg-0d89e927e5645d7c5",
            "State": "REGISTERED",
            "WorkspaceCreationProperties": {
                "EnableInternetAccess": false,
                "UserEnabledAsLocalAdministrator": true,
                "EnableMaintenanceMode": true
            },
            "WorkspaceAccessProperties": {
                "DeviceTypeWindows": "ALLOW",
                "DeviceTypeOsx": "ALLOW",
                "DeviceTypeWeb": "DENY",
                "DeviceTypeIos": "ALLOW",
                "DeviceTypeAndroid": "ALLOW",
                "DeviceTypeChromeOs": "ALLOW",
                "DeviceTypeZeroClient": "ALLOW",
                "DeviceTypeLinux": "DENY"
            },
            "Tenancy": "SHARED",
            "SelfservicePermissions": {
                "RestartWorkspace": "ENABLED",
                "IncreaseVolumeSize": "DISABLED",
                "ChangeComputeType": "DISABLED",
                "SwitchRunningMode": "DISABLED",
                "RebuildWorkspace": "DISABLED"
            }
        }
    ]
}
```

Para obter mais informações, consulte Gerenciar diretórios para o WorkSpaces Personal no Guia de WorkSpaces Administração da Amazon.

- Para obter detalhes da API, consulte DescribeWorkspaceDirectoriesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: essa amostra lista os detalhes do diretório para diretórios registrados

```
Get-WKSWorkspaceDirectory
```

Saída:

```
Alias                       : TestWorkspace
CustomerUserName            : Administrator
DirectoryId                 : d-123414a369
DirectoryName               : TestDirectory.com
DirectoryType               : MicrosoftAD
DnsIpAddresses              : {172.31.43.45, 172.31.2.97}
IamRoleId                   : arn:aws:iam::761234567801:role/
workspaces_RoleDefault
IpGroupIds                  : {}
RegistrationCode            : WSpdx+4RRT43
SelfservicePermissions      : Amazon.WorkSpaces.Model.SelfservicePermissions
State                       : REGISTERED
SubnetIds                   : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy                     : SHARED
WorkspaceAccessProperties   : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
 Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
WorkspaceSecurityGroupId    : sg-0ed2441234a123c43
```

- Para obter detalhes da API, consulte DescribeWorkspaceDirectoriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: essa amostra lista os detalhes do diretório para diretórios registrados

```
Get-WKSWorkspaceDirectory
```

Saída:

```
Alias                       : TestWorkspace
CustomerUserName            : Administrator
```

```
DirectoryId                  : d-123414a369
DirectoryName                : TestDirectory.com
DirectoryType                : MicrosoftAD
DnsIpAddresses               : {172.31.43.45, 172.31.2.97}
IamRoleId                    : arn:aws:iam::761234567801:role/
workspaces_RoleDefault
IpGroupIds                   : {}
RegistrationCode             : WSpdx+4RRT43
SelfservicePermissions       : Amazon.WorkSpaces.Model.SelfservicePermissions
State                        : REGISTERED
SubnetIds                    : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy                      : SHARED
WorkspaceAccessProperties    : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
 Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
WorkspaceSecurityGroupId     : sg-0ed2441234a123c43
```

- Para obter detalhes da API, consulte DescribeWorkspaceDirectoriesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeWorkspaces`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeWorkspaces`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação e gerenciamento de áreas de trabalho virtuais pessoais

CLI

AWS CLI

Para descrever um WorkSpace

O `describe-workspaces` exemplo a seguir descreve o especificado WorkSpace.

```
aws workspaces describe-workspaces \
    --workspace-ids ws-dk1xzr417
```

Saída:

```
{
    "Workspaces": [
        {
            "WorkspaceId": "ws-dk1xzr417",
            "DirectoryId": "d-926722edaf",
            "UserName": "Mary",
            "IpAddress": "172.16.0.175",
            "State": "STOPPED",
            "BundleId": "wsb-0zsvgp8fc",
            "SubnetId": "subnet-500d5819",
            "ComputerName": "WSAMZN-RBSLTTD9",
            "WorkspaceProperties": {
                "RunningMode": "AUTO_STOP",
                "RunningModeAutoStopTimeoutInMinutes": 60,
                "RootVolumeSizeGib": 80,
                "UserVolumeSizeGib": 10,
                "ComputeTypeName": "VALUE"
            },
            "ModificationStates": []
        }
    ]
}
```

Para obter mais informações, consulte Administrar seu WorkSpaces no Guia de WorkSpaces Administração da Amazon.

- Para obter detalhes da API, consulte DescribeWorkspacesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: recupera detalhes de todos os seus WorkSpaces para o pipeline.

```
Get-WKSWorkspace
```

Saída:

```
BundleId                    : wsb-1a2b3c4d
ComputerName                :
```

```
DirectoryId                 : d-1a2b3c4d
ErrorCode                   :
ErrorMessage                :
IpAddress                   :
RootVolumeEncryptionEnabled : False
State                       : PENDING
SubnetId                    :
UserName                    : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey         :
WorkspaceId                 : ws-1a2b3c4d
WorkspaceProperties         : Amazon.WorkSpaces.Model.WorkspaceProperties
```

Exemplo 2: esse comando mostra os valores das propriedades secundárias de **`WorkSpaceProperties`** para um espaço de trabalho na região **`us-west-2`**. Para obter mais informações sobre as propriedades secundárias do**`WorkSpaceProperties`**, consulte https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties
```

Saída:

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

Exemplo 3: esse comando mostra o valor da propriedade secundária **`RootVolumeSizeGib`** de **`WorkSpaceProperties`** para um espaço de trabalho na região **`us-west-2`**. O tamanho do volume raiz, em GiB, é 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Saída:

```
        80
```

- Para obter detalhes da API, consulte DescribeWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: recupera detalhes de todos os seus WorkSpaces para o pipeline.

```
Get-WKSWorkspace
```

Saída:

```
BundleId                    : wsb-1a2b3c4d
ComputerName                :
DirectoryId                 : d-1a2b3c4d
ErrorCode                   :
ErrorMessage                :
IpAddress                   :
RootVolumeEncryptionEnabled : False
State                       : PENDING
SubnetId                    :
UserName                    : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey         :
WorkspaceId                 : ws-1a2b3c4d
WorkspaceProperties         : Amazon.WorkSpaces.Model.WorkspaceProperties
```

Exemplo 2: esse comando mostra os valores das propriedades secundárias de **`WorkSpaceProperties`** para um espaço de trabalho na região **`us-west-2`**. Para obter mais informações sobre as propriedades secundárias do**`WorkSpaceProperties`**, consulte https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties
```

Saída:

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

Exemplo 3: esse comando mostra o valor da propriedade secundária **`RootVolumeSizeGib`** de **`WorkSpaceProperties`** para um espaço de trabalho na região **`us-west-2`**. O tamanho do volume raiz, em GiB, é 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Saída:

```
        80
```

- Para obter detalhes da API, consulte DescribeWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`DescribeWorkspacesConnectionStatus`** com uma CLI

Os exemplos de código a seguir mostram como usar o `DescribeWorkspacesConnectionStatus`.

CLI

AWS CLI

Para descrever o status da conexão de um WorkSpace

O `describe-workspaces-connection-status` exemplo a seguir descreve o status da conexão especificada WorkSpace.

```
aws workspaces describe-workspaces-connection-status \
    --workspace-ids ws-dk1xzr417
```

Saída:

```
{
    "WorkspacesConnectionStatus": [
        {
            "WorkspaceId": "ws-dk1xzr417",
            "ConnectionState": "CONNECTED",
            "ConnectionStateCheckTimestamp": 1662526214.744
        }
    ]
```

```
}
```

Para obter mais informações, consulte Administrar seu WorkSpaces no Guia de WorkSpaces Administração da Amazon.

- Para obter detalhes da API, consulte DescribeWorkspacesConnectionStatusem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: essa amostra busca o status da conexão para o espaço de trabalho especificado

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Para obter detalhes da API, consulte DescribeWorkspacesConnectionStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: essa amostra busca o status da conexão para o espaço de trabalho especificado

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Para obter detalhes da API, consulte DescribeWorkspacesConnectionStatusem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyWorkspaceCreationProperties`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyWorkspaceCreationProperties`.

CLI

AWS CLI

Para modificar uma propriedade de WorkSpace criação de um diretório

O exemplo `modify-workspace-creation-properties` a seguir ativa a propriedade `EnableInternetAccess` para o diretório especificado. Isso permite a atribuição automática de endereços IP públicos para o WorkSpaces criado para o diretório.

```
aws workspaces modify-workspace-creation-properties \
    --resource-id d-926722edaf \
    --workspace-creation-properties EnableInternetAccess=true
```

Este comando não produz saída.

Para obter mais informações, consulte Atualizar os detalhes do seu diretório WorkSpaces no Guia de WorkSpaces Administração da Amazon.

- Para obter detalhes da API, consulte ModifyWorkspaceCreationPropertiesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo permite que o Modo de Manutenção e Acesso à Internet seja verdadeiro como valores padrão ao criar um espaço de trabalho

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Para obter detalhes da API, consulte ModifyWorkspaceCreationPropertiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo permite que o Modo de Manutenção e Acesso à Internet seja verdadeiro como valores padrão ao criar um espaço de trabalho

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Para obter detalhes da API, consulte ModifyWorkspaceCreationPropertiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyWorkspaceProperties`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyWorkspaceProperties`.

CLI

AWS CLI

Para modificar o modo de execução de um WorkSpace

O `modify-workspace-properties` exemplo a seguir define o modo de execução do especificado WorkSpace para`AUTO_STOP`.

```
aws workspaces modify-workspace-properties \
    --workspace-id ws-dk1xzr417 \
    --workspace-properties RunningMode=AUTO_STOP
```

Este comando não produz saída.

Para obter mais informações, consulte Modificar a WorkSpace no Guia de WorkSpaces Administração da Amazon.

- Para obter detalhes da API, consulte ModifyWorkspacePropertiesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera a propriedade do modo de execução do espaço de trabalho para interrupção automática referente ao espaço de trabalho especificado

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- Para obter detalhes da API, consulte ModifyWorkspacePropertiesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera a propriedade do modo de execução do espaço de trabalho para interrupção automática referente ao espaço de trabalho especificado

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- Para obter detalhes da API, consulte ModifyWorkspaceProperties em Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`ModifyWorkspaceState`** com uma CLI

Os exemplos de código a seguir mostram como usar o `ModifyWorkspaceState`.

CLI

AWS CLI

Para modificar o estado de um WorkSpace

O `modify-workspace-state` exemplo a seguir define o estado do especificado WorkSpace para`ADMIN_MAINTENANCE`.

```
aws workspaces modify-workspace-state \
    --workspace-id ws-dk1xzr417 \
    --workspace-state ADMIN_MAINTENANCE
```

Este comando não produz saída.

Para obter mais informações, consulte a WorkSpace manutenção no Guia de WorkSpaces Administração da Amazon.

- Para obter detalhes da API, consulte ModifyWorkspaceState em Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo altera o estado do espaço de trabalho especificado para Disponível

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceState AVAILABLE
```

- Para obter detalhes da API, consulte ModifyWorkspaceState em Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o estado do espaço de trabalho especificado para Disponível

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceState AVAILABLE
```

- Para obter detalhes da API, consulte ModifyWorkspaceStateem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RebootWorkspaces`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RebootWorkspaces`.

CLI

AWS CLI

Para reinicializar um WorkSpace

O `reboot-workspaces` exemplo a seguir reinicializa o especificado WorkSpace.

```
aws workspaces reboot-workspaces \
    --reboot-workspace-requests ws-dk1xzr417
```

Saída:

```
{
    "FailedRequests": []
}
```

Para obter mais informações, consulte Reboot a WorkSpace no Amazon WorkSpaces Administration Guide.

- Para obter detalhes da API, consulte RebootWorkspacesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Reinicializa o especificado WorkSpace.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemplo 2: Reinicializa várias WorkSpaces.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Exemplo 3: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - de Restart-WKSWorkspace, fazendo com que o WorkSpaces seja reiniciado.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Para obter detalhes da API, consulte RebootWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Reinicializa o especificado WorkSpace.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemplo 2: Reinicializa várias WorkSpaces.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Exemplo 3: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - de Restart-WKSWorkspace, fazendo com que o WorkSpaces seja reiniciado.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Para obter detalhes da API, consulte RebootWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RebuildWorkspaces`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RebuildWorkspaces`.

CLI

AWS CLI

Para reconstruir um WorkSpace

O `rebuild-workspaces` exemplo a seguir reconstrói o especificado. WorkSpace

```
aws workspaces rebuild-workspaces \
    --rebuild-workspace-requests ws-dk1xzr417
```

Saída:

```
{
    "FailedRequests": []
}
```

Para obter mais informações, consulte Rebuild a WorkSpace no Amazon WorkSpaces Administration Guide.

- Para obter detalhes da API, consulte RebuildWorkspacesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: reconstrói o especificado. WorkSpace

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemplo 2: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - de Reset-WKSWorkspace, fazendo com que o WorkSpaces seja reconstruído.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Para obter detalhes da API, consulte RebuildWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: reconstrói o especificado. WorkSpace

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemplo 2: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - de Reset-WKSWorkspace, fazendo com que o WorkSpaces seja reconstruído.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Para obter detalhes da API, consulte RebuildWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`RegisterWorkspaceDirectory`** com uma CLI

Os exemplos de código a seguir mostram como usar o `RegisterWorkspaceDirectory`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação e gerenciamento de áreas de trabalho virtuais pessoais

CLI

AWS CLI

Para registrar um diretório

O `register-workspace-directory` exemplo a seguir registra o diretório especificado para uso com a Amazon WorkSpaces.

```
aws workspaces register-workspace-directory \
    --directory-id d-926722edaf
```

Este comando não produz saída.

Para obter mais informações, consulte Registrar um AWS diretório existente do Directory Service com WorkSpaces Personal no Amazon WorkSpaces Administration Guide.

- Para obter detalhes da API, consulte RegisterWorkspaceDirectoryem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo registra o diretório especificado para o Workspaces Service

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Para obter detalhes da API, consulte RegisterWorkspaceDirectoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra o diretório especificado para o Workspaces Service

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Para obter detalhes da API, consulte RegisterWorkspaceDirectoryem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`StopWorkspaces`** com uma CLI

Os exemplos de código a seguir mostram como usar o `StopWorkspaces`.

CLI

AWS CLI

Para parar um AutoStop WorkSpace

O `stop-workspaces` exemplo a seguir interrompe o especificado WorkSpace. Eles WorkSpace devem ter um modo de execução de`AutoStop`.

```
aws workspaces stop-workspaces \
```

```
    --stop-workspace-requests WorkspaceId=ws-dk1xzr417
```

Saída:

```
{
    "FailedRequests": []
}
```

Para obter mais informações, consulte Stop and start an AutoStop WorkSpace no Amazon WorkSpaces Administration Guide.

- Para obter detalhes da API, consulte StopWorkspacesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Interrompe várias WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Exemplo 2: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - Stop-WKSWorkspace que faz com que o WorkSpaces seja interrompido.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Exemplo 3: Este exemplo mostra como passar StopRequest objetos definindo o WorkSpaces a ser parado.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Stop-WKSWorkspace -Request $arrRequest
```

- Para obter detalhes da API, consulte StopWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Interrompe várias WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Exemplo 2: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - Stop-WKSWorkspace que faz com que o WorkSpaces seja interrompido.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Exemplo 3: Este exemplo mostra como passar StopRequest objetos definindo o WorkSpaces a ser parado.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Stop-WKSWorkspace -Request $arrRequest
```

- Para obter detalhes da API, consulte StopWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Usar **`TerminateWorkspaces`** com uma CLI

Os exemplos de código a seguir mostram como usar o `TerminateWorkspaces`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- Criação e gerenciamento de áreas de trabalho virtuais pessoais

CLI

AWS CLI

Para encerrar um WorkSpace

O exemplo `terminate-workspaces` a seguir encerra o WorkSpace especificado.

```
aws workspaces terminate-workspaces \
    --terminate-workspace-requests ws-dk1xzr417
```

Saída:

```
{
    "FailedRequests": []
}
```

Para obter mais informações, consulte Excluir um WorkSpace no Guia de WorkSpaces Administração da Amazon.

- Para obter detalhes da API, consulte TerminateWorkspacesem Referência de AWS CLI Comandos.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: Encerra vários WorkSpaces. O uso da opção -Force impede que o cmdlet solicite confirmação.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Exemplo 2: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - de Remove-WKSWorkspace, encerrando todos os. WorkSpaces O cmdlet avisará antes que cada um WorkSpace seja encerrado. Para ignorar a solicitação de confirmação, adicione a opção -Force.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Exemplo 3: Este exemplo mostra como passar TerminateRequest objetos definindo o WorkSpaces a ser encerrado. O cmdlet solicitará a confirmação antes de continuar, a menos que o parâmetro da opção -Force também seja especificado.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Remove-WKSWorkspace -Request $arrRequest
```

- Para obter detalhes da API, consulte TerminateWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V4).

Ferramentas para PowerShell V5

Exemplo 1: Encerra vários WorkSpaces. O uso da opção -Force impede que o cmdlet solicite confirmação.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Exemplo 2: recupera a coleção de todos os seus WorkSpaces e canaliza os IDs para o WorkSpaceId parâmetro - de Remove-WKSWorkspace, encerrando todos os. WorkSpaces O cmdlet avisará antes que cada um WorkSpace seja encerrado. Para ignorar a solicitação de confirmação, adicione a opção -Force.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Exemplo 3: Este exemplo mostra como passar TerminateRequest objetos definindo o WorkSpaces a ser encerrado. O cmdlet solicitará a confirmação antes de continuar, a menos que o parâmetro da opção -Force também seja especificado.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
```

```
$arrRequest += $request2
Remove-WKSWorkspace -Request $arrRequest
```

- Para obter detalhes da API, consulte TerminateWorkspacesem Referência de Ferramentas da AWS para PowerShell cmdlet (V5).

Cenários para WorkSpaces usar AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns WorkSpaces com AWS SDKs. Esses cenários mostram como realizar tarefas específicas chamando várias funções internas WorkSpaces ou combinadas com outras Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- Criação e gerenciamento de áreas de trabalho virtuais pessoais

Criação e gerenciamento de áreas de trabalho virtuais pessoais

O exemplo de código a seguir mostra como:

- Verifique WorkSpaces a disponibilidade em sua região
- Listar WorkSpaces pacotes disponíveis
- Crie um WorkSpace
- Verifique o status do seu WorkSpace
- Limpar os recursos

Bash

AWS CLI com script Bash

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a WorkSpace in WorkSpaces Personal
# This script follows the workflow described in the AWS documentation
# https://docs.aws.amazon.com/workspaces/latest/adminguide/create-workspaces-
personal.html

# Set up logging
LOG_FILE="workspaces_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting WorkSpaces creation script"
echo "==============================================="

# Initialize resource tracking array
declare -a CREATED_RESOURCES

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    exit 1
}

# Function to check if a command succeeded
check_command() {
    # Check for ResourceNotFound.User error specifically
    if echo "$1" | grep -q "ResourceNotFound.User"; then
        echo ""
```

```
        echo "ERROR: User not found in the directory."
        echo ""
        echo "This error occurs when the specified username doesn't exist in the
 directory."
        echo ""
        echo "To resolve this issue:"
        echo "1. Ensure the user exists in the directory before creating a
 WorkSpace."
        echo "2. For Simple AD and AWS Managed Microsoft AD:"
        echo "   - Connect to a directory-joined instance"
        echo "   - Use Active Directory tools to create the user"
        echo "   - See: https://docs.aws.amazon.com/workspaces/latest/adminguide/
manage-users.html"
        echo ""
        echo "3. For AD Connector:"
        echo "   - Create the user in your on-premises Active Directory"
        echo "   - Ensure proper synchronization with the AD Connector"
        echo ""
        echo "4. Alternatively, you can use the AWS Console to create a
 WorkSpace,"
        echo "   which can create the user automatically in some directory
 types."
        echo ""
        handle_error "User '$USERNAME' not found in directory '$DIRECTORY_ID'"
    # Check for other errors
    elif echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Step 0: Select AWS region
echo ""
echo "==============================================="
echo "AWS REGION SELECTION"
echo "==============================================="
echo "Enter the AWS region to use (e.g., us-east-1, us-west-2):"
AWS_REGION="${AWS_REGION:-us-east-1}"

if [ -z "$AWS_REGION" ]; then
    handle_error "Region cannot be empty"
fi

export AWS_DEFAULT_REGION="$AWS_REGION"
echo "Using AWS region: $AWS_REGION"
```

```
# Step 1: Prompt for directory ID
echo ""
echo "==============================================="
echo "DIRECTORY SELECTION"
echo "==============================================="
echo "Listing available directories..."

DIRECTORIES_OUTPUT=$(aws workspaces describe-workspace-directories --output json)
check_command "$DIRECTORIES_OUTPUT"
echo "$DIRECTORIES_OUTPUT"

# Extract directory IDs and display them
DIRECTORY_IDS=$(echo "$DIRECTORIES_OUTPUT" | grep -o '"DirectoryId": "[^"]*' |
 cut -d'"' -f4)

if [ -z "$DIRECTORY_IDS" ]; then
    echo "No directories found. Please create a directory first using AWS
 Directory Service."
    echo "For more information, see: https://docs.aws.amazon.com/workspaces/
latest/adminguide/register-deregister-directory.html"
    exit 1
fi

echo ""
echo "Available directory IDs:"
echo "$DIRECTORY_IDS"
echo ""
echo "Enter the directory ID you want to use:"
DIRECTORY_ID=$(echo "$DIRECTORY_IDS" | head -1)

# Validate directory ID
if ! echo "$DIRECTORY_IDS" | grep -q "$DIRECTORY_ID"; then
    echo "Directory ID $DIRECTORY_ID not found in the list of available
 directories."
    echo "Please check the ID and try again."
    exit 1
fi

echo "Selected directory ID: $DIRECTORY_ID"

# Step 2: Check if directory is registered with WorkSpaces
echo ""
echo "==============================================="
```

```
echo "CHECKING DIRECTORY REGISTRATION"
echo "==============================================="

REGISTERED=$(echo "$DIRECTORIES_OUTPUT" | grep -A 5 "\"DirectoryId\":
 \"$DIRECTORY_ID\"" | grep -c "\"State\": \"REGISTERED\"")

if [ "$REGISTERED" -eq 0 ]; then
    echo "Directory $DIRECTORY_ID is not registered with WorkSpaces. Registering
 now..."
    REGISTER_OUTPUT=$(aws workspaces register-workspace-directory --directory-id
 "$DIRECTORY_ID")
    check_command "$REGISTER_OUTPUT"
    echo "Directory registration initiated. This may take a few minutes."

    # Add to resource tracking
    CREATED_RESOURCES+=("Directory registration: $DIRECTORY_ID")

    # Wait for directory to be registered
    echo "Waiting for directory registration to complete..."
    sleep 30

    # Check registration status
    REGISTRATION_CHECK=$(aws workspaces describe-workspace-directories --
directory-ids "$DIRECTORY_ID")
    check_command "$REGISTRATION_CHECK"

    REGISTRATION_STATE=$(echo "$REGISTRATION_CHECK" | grep -o '"State": "[^"]*' |
 cut -d'"' -f4)
    if [ "$REGISTRATION_STATE" != "REGISTERED" ]; then
        echo "Directory registration is still in progress. Current state:
 $REGISTRATION_STATE"
        echo "Please check the AWS console for the final status."
        echo "You may need to wait a few minutes before proceeding."
    else
        echo "Directory successfully registered with WorkSpaces."
    fi
else
    echo "Directory $DIRECTORY_ID is already registered with WorkSpaces."
fi

# Get directory type to provide appropriate user guidance
DIRECTORY_TYPE=$(echo "$DIRECTORIES_OUTPUT" | grep -A 10 "\"DirectoryId\":
 \"$DIRECTORY_ID\"" | grep -o '"DirectoryType": "[^"]*' | cut -d'"' -f4)
echo "Directory type: $DIRECTORY_TYPE"
```

```
# Display user creation guidance based on directory type
echo ""
echo "==============================================="
echo "USER CREATION GUIDANCE"
echo "==============================================="
case "$DIRECTORY_TYPE" in
    "SimpleAD" | "MicrosoftAD")
        echo "For $DIRECTORY_TYPE, users must be created using Active Directory
 tools."
        echo "1. Connect to a directory-joined EC2 instance"
        echo "2. Use Active Directory Users and Computers to create users"
        echo "3. For detailed instructions, see: https://docs.aws.amazon.com/
workspaces/latest/adminguide/manage-users.html"
        ;;
    "ADConnector")
        echo "For AD Connector, users must exist in your on-premises Active
 Directory."
        echo "1. Create the user in your on-premises Active Directory"
        echo "2. Ensure the user is in an OU that is within the scope of your AD
 Connector"
        echo "3. For detailed instructions, see: https://docs.aws.amazon.com/
directoryservice/latest/admin-guide/ad_connector_management.html"
        ;;
    *)
        echo "For this directory type, ensure users exist before creating
 WorkSpaces."
        echo "For detailed instructions, see: https://docs.aws.amazon.com/
workspaces/latest/adminguide/manage-users.html"
        ;;
esac
echo ""

# Step 3: List available bundles
echo ""
echo "==============================================="
echo "BUNDLE SELECTION"
echo "==============================================="
echo "Listing available WorkSpace bundles..."

# Get bundles with a format that's easier to parse
BUNDLES_OUTPUT=$(aws workspaces describe-workspace-
bundles --owner AMAZON --output text --query "Bundles[*].
[BundleId,Name,ComputeType.Name,RootStorage.Capacity,UserStorage.Capacity]")
```

```
check_command "$BUNDLES_OUTPUT"

# Extract bundle information and display in a numbered list
echo "Available bundles:"
echo "-----------------"
echo "NUM | BUNDLE ID | NAME | COMPUTE TYPE | ROOT STORAGE | USER STORAGE"
echo "-------------------------------------------------------------------"

# Create arrays to store bundle information
declare -a BUNDLE_IDS
declare -a BUNDLE_NAMES

# Process the output to extract bundle information
COUNT=1
while IFS=$'\t' read -r BUNDLE_ID BUNDLE_NAME COMPUTE_TYPE ROOT_STORAGE
 USER_STORAGE || [[ -n "$BUNDLE_ID" ]]; do
    # Store in arrays
    BUNDLE_IDS[$COUNT]="$BUNDLE_ID"
    BUNDLE_NAMES[$COUNT]="$BUNDLE_NAME"

    # Display with number
    echo "$COUNT | $BUNDLE_ID | $BUNDLE_NAME | $COMPUTE_TYPE | $ROOT_STORAGE GB |
 $USER_STORAGE GB"

    ((COUNT++))
done <<< "$BUNDLES_OUTPUT"

# Prompt for selection
echo ""
echo "Enter the number of the bundle you want to use (1-$((COUNT-1))):"
BUNDLE_SELECTION="1"

# Validate selection
if ! [[ "$BUNDLE_SELECTION" =~ ^[0-9]+$ ]] || [ "$BUNDLE_SELECTION" -lt 1 ] ||
 [ "$BUNDLE_SELECTION" -ge "$COUNT" ]; then
    handle_error "Invalid bundle selection. Please enter a number between 1 and
 $((COUNT-1))."
fi

# Get the selected bundle ID
BUNDLE_ID="${BUNDLE_IDS[$BUNDLE_SELECTION]}"
BUNDLE_NAME="${BUNDLE_NAMES[$BUNDLE_SELECTION]}"

echo "Selected bundle: $BUNDLE_NAME (ID: $BUNDLE_ID)"
```

```
# Step 4: Prompt for username
echo ""
echo "==============================================="
echo "USER INFORMATION"
echo "==============================================="
echo "Enter the username for the WorkSpace:"
USERNAME="testuser-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n 1)"

echo "NOTE: The user must already exist in the directory for the WorkSpace
 creation to succeed."
echo "If you're using Simple AD or AWS Managed Microsoft AD, the user must be
 created using Active Directory tools."
echo "If you're using AD Connector, the user must exist in your on-premises
 Active Directory."
echo ""

echo "Enter the user's first name:"
FIRST_NAME="Test"

echo "Enter the user's last name:"
LAST_NAME="User"

echo "Enter the user's email address:"
EMAIL="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1)@example.com"

# Step 5: Choose running mode
echo ""
echo "==============================================="
echo "RUNNING MODE SELECTION"
echo "==============================================="
echo "Select running mode:"
echo "1. AlwaysOn (billed monthly)"
echo "2. AutoStop (billed hourly)"
RUNNING_MODE_CHOICE="1"

if [ "$RUNNING_MODE_CHOICE" = "1" ]; then
    RUNNING_MODE="ALWAYS_ON"
    AUTO_STOP_TIMEOUT=""
else
    RUNNING_MODE="AUTO_STOP"
    AUTO_STOP_TIMEOUT=60
fi
```

```
echo "Selected running mode: $RUNNING_MODE"

# Step 6: Add tags (optional)
echo ""
echo "==============================================="
echo "TAGS (OPTIONAL)"
echo "==============================================="
echo "Would you like to add tags to your WorkSpace? (y/n):"
ADD_TAGS="n"

TAGS_JSON=""
if [ "$ADD_TAGS" = "y" ] || [ "$ADD_TAGS" = "Y" ]; then
    echo "Enter tag key (e.g., Department):"
    read -r TAG_KEY

    echo "Enter tag value (e.g., IT):"
    read -r TAG_VALUE

    TAGS_JSON="[{\"Key\":\"$TAG_KEY\",\"Value\":\"$TAG_VALUE\"}]"
fi

# Step 7: Create the WorkSpace
echo ""
echo "==============================================="
echo "CREATING WORKSPACE"
echo "==============================================="
echo "Creating WorkSpace with the following parameters:"
echo "Directory ID: $DIRECTORY_ID"
echo "Username: $USERNAME"
echo "Bundle ID: $BUNDLE_ID"
echo "Running Mode: $RUNNING_MODE"
if [ -n "$TAGS_JSON" ]; then
    echo "Tags: $TAG_KEY=$TAG_VALUE"
fi

# Create JSON for workspace properties
if [ "$RUNNING_MODE" = "AUTO_STOP" ]; then
    PROPERTIES_JSON="{\"RunningMode\":\"$RUNNING_MODE\",
\"RunningModeAutoStopTimeoutInMinutes\":$AUTO_STOP_TIMEOUT}"
else
    PROPERTIES_JSON="{\"RunningMode\":\"$RUNNING_MODE\"}"
fi
```

```
# Create JSON for workspaces parameter
WORKSPACE_JSON="{\"DirectoryId\":\"$DIRECTORY_ID\",\"UserName\":\"$USERNAME\",
\"BundleId\":\"$BUNDLE_ID\",\"WorkspaceProperties\":$PROPERTIES_JSON"

# Add tags if specified
if [ -n "$TAGS_JSON" ]; then
    USER_TAGS=$(echo "$TAGS_JSON" | sed 's/^\[//;s/\]$//')
    WORKSPACE_JSON="$WORKSPACE_JSON,\"Tags\":[$USER_TAGS,{\"Key\":\"project\",
\"Value\":\"doc-smith\"},{\"Key\":\"tutorial\",\"Value\":\"workspaces-personal
\"}]"
else
    WORKSPACE_JSON="$WORKSPACE_JSON,\"Tags\":[{\"Key\":\"project\",\"Value\":
\"doc-smith\"},{\"Key\":\"tutorial\",\"Value\":\"workspaces-personal\"}]"
fi

# Close the JSON object
WORKSPACE_JSON="$WORKSPACE_JSON}"

# Construct the create-workspaces command
CREATE_COMMAND="aws workspaces create-workspaces --workspaces '$WORKSPACE_JSON'"

echo "Executing: $CREATE_COMMAND"
CREATE_OUTPUT=$(eval "$CREATE_COMMAND")
check_command "$CREATE_OUTPUT"
echo "$CREATE_OUTPUT"

# Extract WorkSpace ID
WORKSPACE_ID=$(echo "$CREATE_OUTPUT" | grep -o '"WorkspaceId": "[^"]*' | head -1
 | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    handle_error "Failed to extract WorkSpace ID from creation output."
fi

echo "WorkSpace creation initiated. WorkSpace ID: $WORKSPACE_ID"
CREATED_RESOURCES+=("WorkSpace: $WORKSPACE_ID")

# Step 8: Check WorkSpace status
echo ""
echo "================================================"
echo "CHECKING WORKSPACE STATUS"
echo "================================================"
echo "Checking status of WorkSpace $WORKSPACE_ID..."
```

```
# Initial status check
STATUS_OUTPUT=$(aws workspaces describe-workspaces --workspace-ids
 "$WORKSPACE_ID")
check_command "$STATUS_OUTPUT"
echo "$STATUS_OUTPUT"

WORKSPACE_STATE=$(echo "$STATUS_OUTPUT" | grep -o '"State": "[^"]*' | head -1 |
 cut -d'"' -f4)
echo "Current WorkSpace state: $WORKSPACE_STATE"

# Wait for WorkSpace to be available (this can take 20+ minutes)
echo ""
echo "WorkSpace creation is in progress. This can take 20+ minutes."
echo "The script will check the status every 60 seconds."
echo "Press Ctrl+C to exit the script at any time. The WorkSpace will continue to
 be created."

while [ "$WORKSPACE_STATE" = "PENDING" ]; do
    echo "$(date): WorkSpace state is still PENDING. Waiting 60 seconds before
 checking again..."
    sleep 60

    STATUS_OUTPUT=$(aws workspaces describe-workspaces --workspace-ids
 "$WORKSPACE_ID")
    check_command "$STATUS_OUTPUT"

    WORKSPACE_STATE=$(echo "$STATUS_OUTPUT" | grep -o '"State": "[^"]*' | head -1
 | cut -d'"' -f4)
    echo "$(date): Current WorkSpace state: $WORKSPACE_STATE"

    # If state is ERROR or UNHEALTHY, exit
    if [ "$WORKSPACE_STATE" = "ERROR" ] || [ "$WORKSPACE_STATE" = "UNHEALTHY" ];
 then
        handle_error "WorkSpace creation failed. Final state: $WORKSPACE_STATE"
    fi

    # If state is AVAILABLE, break the loop
    if [ "$WORKSPACE_STATE" = "AVAILABLE" ]; then
        break
    fi
done

# Step 9: Display WorkSpace information
echo ""
```

```
echo "==============================================="
echo "WORKSPACE CREATION COMPLETE"
echo "==============================================="
echo "WorkSpace has been successfully created!"
echo "WorkSpace ID: $WORKSPACE_ID"
echo "Directory ID: $DIRECTORY_ID"
echo "Username: $USERNAME"
echo "Running Mode: $RUNNING_MODE"

# Step 10: Remind about invitation emails
echo ""
echo "==============================================="
echo "INVITATION EMAILS"
echo "==============================================="
echo "IMPORTANT: If you're using AD Connector or a trust relationship, or if the
 user already exists in Active Directory,"
echo "invitation emails are not sent automatically. You'll need to manually send
 an invitation email."
echo "For more information, see: https://docs.aws.amazon.com/workspaces/latest/
adminguide/manage-workspaces-users.html#send-invitation"

# Step 11: Cleanup confirmation
echo ""
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

echo ""
echo "Do you want to clean up all created resources? (y/n):"
CLEANUP_CHOICE="y"

if [ "$CLEANUP_CHOICE" = "y" ] || [ "$CLEANUP_CHOICE" = "Y" ]; then
    echo ""
    echo "==============================================="
    echo "CLEANING UP RESOURCES"
    echo "==============================================="

    # Terminate WorkSpace
    if [ -n "$WORKSPACE_ID" ]; then
        echo "Terminating WorkSpace $WORKSPACE_ID..."
```

```
        TERMINATE_OUTPUT=$(aws workspaces terminate-workspaces --terminate-
workspace-requests WorkspaceId="$WORKSPACE_ID")
        check_command "$TERMINATE_OUTPUT"
        echo "$TERMINATE_OUTPUT"
        echo "WorkSpace termination initiated. This may take a few minutes."
    fi

    # Deregister directory (only if we registered it in this script)
    if [[ " ${CREATED_RESOURCES[*]} " == *"Directory registration:
 $DIRECTORY_ID"* ]]; then
        echo "Deregistering directory $DIRECTORY_ID from WorkSpaces..."
        DEREGISTER_OUTPUT=$(aws workspaces deregister-workspace-directory --
directory-id "$DIRECTORY_ID")
        check_command "$DEREGISTER_OUTPUT"
        echo "$DEREGISTER_OUTPUT"
        echo "Directory deregistration initiated. This may take a few minutes."
    fi

    echo "Cleanup completed."
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo ""
echo "================================================"
echo "SCRIPT COMPLETED"
echo "================================================"
echo "Log file: $LOG_FILE"
echo "Thank you for using the WorkSpaces creation script!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateWorkspaces
  - DeregisterWorkspaceDirectory
  - DescribeWorkspaceBundles
  - DescribeWorkspaceDirectories
  - DescribeWorkspaces
  - RegisterWorkspaceDirectory
  - TerminateWorkspaces

Exemplos de código para X-Ray usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS X-Ray com um kit de desenvolvimento de AWS software (SDK).

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Mais atributos

- X-Ray Guia do desenvolvedor — Mais informações sobre X-Ray.
- X-Ray Referência da API — Detalhes sobre todas as X-Ray ações disponíveis.
- AWS Centro do desenvolvedor — exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos básicos de X-Ray uso AWS SDKs
  - Ações para X-Ray usar AWS SDKs
    - Use CreateGroupcom um AWS SDK ou CLI
    - Use CreateSamplingRulecom um AWS SDK ou CLI
    - Use DeleteGroupcom um AWS SDK ou CLI
    - Use DeleteSamplingRulecom um AWS SDK ou CLI
    - Use GetGroupscom um AWS SDK ou CLI
    - Use GetSamplingRulescom um AWS SDK ou CLI
    - Use GetServiceGraphcom um AWS SDK ou CLI

Exemplos básicos de X-Ray uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS X-Ray com AWS SDKs.

Exemplos

- Ações para X-Ray usar AWS SDKs

- Use CreateGroupcom um AWS SDK ou CLI
- Use CreateSamplingRulecom um AWS SDK ou CLI
- Use DeleteGroupcom um AWS SDK ou CLI
- Use DeleteSamplingRulecom um AWS SDK ou CLI
- Use GetGroupscom um AWS SDK ou CLI
- Use GetSamplingRulescom um AWS SDK ou CLI
- Use GetServiceGraphcom um AWS SDK ou CLI

Ações para X-Ray usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar X-Ray ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS X-Ray.

Exemplos

- Use CreateGroupcom um AWS SDK ou CLI
- Use CreateSamplingRulecom um AWS SDK ou CLI
- Use DeleteGroupcom um AWS SDK ou CLI
- Use DeleteSamplingRulecom um AWS SDK ou CLI
- Use GetGroupscom um AWS SDK ou CLI
- Use GetSamplingRulescom um AWS SDK ou CLI
- Use GetServiceGraphcom um AWS SDK ou CLI

Use **`CreateGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateGroup`.

CLI

AWS CLI

Para criar um grupo

O exemplo `create-group` a seguir cria um recurso de grupo chamado `AdminGroup`. O grupo obtém uma expressão de filtro que define os critérios do grupo como um segmento relacionado a um serviço específico que causa uma falha ou um erro.

```
aws xray create-group \
   --group-name "AdminGroup" \
   --filter-expression "service(\"mydomain.com\") {fault OR error}"
```

Saída:

```
{
    "GroupName": "AdminGroup",
    "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/AdminGroup/123456789",
    "FilterExpression": "service(\"mydomain.com\") {fault OR error}"
}
```

Para obter mais informações, consulte Como definir configurações de amostragem, grupos e criptografia com a AWS X-Ray API no Guia do AWS X-Ray desenvolvedor.

- Para obter detalhes da API, consulte CreateGroupem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.CreateGroupRequest;
import software.amazon.awssdk.services.xray.model.CreateGroupResponse;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of the group to create\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        createNewGroup(xRayClient, groupName);
    }

    public static void createNewGroup(XRayClient xRayClient, String groupName) {
        try {
            CreateGroupRequest groupRequest = CreateGroupRequest.builder()
                    .filterExpression("fault = true AND http.url CONTAINS
 \"example/game\" AND responsetime >= 5")
                    .groupName(groupName)
                    .build();

            CreateGroupResponse groupResponse =
 xRayClient.createGroup(groupRequest);
            System.out.println("The Group ARN is " +
 groupResponse.group().groupARN());

        } catch (XRayException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewGroup(groupNameVal: String?) {
    val groupRequest =
        CreateGroupRequest {
            filterExpression = "fault = true AND http.url CONTAINS \"example/game
\" AND responsetime >= 5"
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val groupResponse = xRayClient.createGroup(groupRequest)
        println("The Group ARN is " + (groupResponse.group?.groupArn))
    }
}
```

- Para obter detalhes da API, consulte a CreateGroupreferência da API AWS SDK for Kotlin.

Use **`CreateSamplingRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateSamplingRule`.

CLI

AWS CLI

Para criar uma regra de amostragem

O exemplo `create-sampling-rule` a seguir cria uma regra para controlar o comportamento de amostragem para aplicações instrumentados. As regras são fornecidas por um arquivo JSON. A maioria dos campos da regra de amostragem é necessária para criar a regra.

```
aws xray create-sampling-rule \
    --cli-input-json file://9000-base-scorekeep.json
```

Conteúdo de `9000-base-scorekeep.json`:

```
{
    "SamplingRule": {
        "RuleName": "base-scorekeep",
        "ResourceARN": "*",
        "Priority": 9000,
        "FixedRate": 0.1,
        "ReservoirSize": 5,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1
    }
}
```

Saída:

```
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "base-scorekeep",
            "RuleARN": "arn:aws:xray:us-west-2:123456789012:sampling-rule/base-
scorekeep",
            "ResourceARN": "*",
            "Priority": 9000,
```

```
            "FixedRate": 0.1,
            "ReservoirSize": 5,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "*",
            "URLPath": "*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574410.0,
        "ModifiedAt": 1530574410.0
    }
}
```

Para obter mais informações, consulte Como definir configurações de amostragem, grupos e criptografia com a AWS X-Ray API no Guia do AWS X-Ray desenvolvedor.

- Para obter detalhes da API, consulte CreateSamplingRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.CreateSamplingRuleRequest;
import software.amazon.awssdk.services.xray.model.SamplingRule;
import software.amazon.awssdk.services.xray.model.XRayException;
import software.amazon.awssdk.services.xray.model.CreateSamplingRuleResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateSamplingRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <ruleName>

                Where:
                   ruleName - The name of the rule\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        createRule(xRayClient, ruleName);
    }

    public static void createRule(XRayClient xRayClient, String ruleName) {
        try {

            SamplingRule rule = SamplingRule.builder()
                    .ruleName(ruleName)
                    .priority(1)
                    .httpMethod("*")
                    .serviceType("*")
                    .serviceName("*")
                    .urlPath("*")
                    .version(1)
                    .host("*")
                    .resourceARN("*")
                    .build();
```

```
            CreateSamplingRuleRequest ruleRequest =
 CreateSamplingRuleRequest.builder()
                    .samplingRule(rule)
                    .build();

            CreateSamplingRuleResponse ruleResponse =
 xRayClient.createSamplingRule(ruleRequest);
            System.out.println(
                    "The ARN of the new rule is " +
 ruleResponse.samplingRuleRecord().samplingRule().ruleARN());

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateSamplingRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createRule(ruleNameVal: String?) {
    val rule =
        SamplingRule {
            ruleName = ruleNameVal
            priority = 1
            httpMethod = "*"
            serviceType = "*"
            serviceName = "*"
```

```
            urlPath = "*"
            version = 1
            host = "*"
            resourceArn = "*"
        }

    val ruleRequest =
        CreateSamplingRuleRequest {
            samplingRule = rule
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val ruleResponse: CreateSamplingRuleResponse =
 xRayClient.createSamplingRule(ruleRequest)
        println("The ARN of the new rule is
 ${ruleResponse.samplingRuleRecord?.samplingRule?.ruleArn}")
    }
}
```

- Para obter detalhes da API, consulte a CreateSamplingRulereferência da API AWS SDK for Kotlin.

Use **`DeleteGroup`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteGroup`.

CLI

AWS CLI

Para excluir um grupo

O exemplo `delete-group` a seguir exclui o recurso de grupo especificado.

```
aws xray delete-group \
    --group-name "AdminGroup" \
    --group-arn "arn:aws:xray:us-east-2:123456789012:group/AdminGroup/123456789"
```

Este comando não produz saída.

Para obter mais informações, consulte Como definir configurações de amostragem, grupos e criptografia com a AWS X-Ray API no Guia do AWS X-Ray desenvolvedor.

- Para obter detalhes da API, consulte DeleteGroupem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.DeleteGroupRequest;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of the group to delete\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        deleteSpecificGroup(xRayClient, groupName);
    }

    public static void deleteSpecificGroup(XRayClient xRayClient, String
 groupName) {
        try {
            DeleteGroupRequest groupRequest = DeleteGroupRequest.builder()
                    .groupName(groupName)
                    .build();

            xRayClient.deleteGroup(groupRequest);
            System.out.println(groupName + " was deleted!");

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteGroupa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSpecificGroup(groupNameVal: String) {
```

```
    val groupRequest =
        DeleteGroupRequest {
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteGroup(groupRequest)
        println("$groupNameVal was deleted!")
    }
}
```

- Para obter detalhes da API, consulte a DeleteGroupreferência da API AWS SDK for Kotlin.

Use **`DeleteSamplingRule`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteSamplingRule`.

CLI

AWS CLI

Para excluir uma regra de amostragem

O exemplo `delete-sampling-rule` a seguir exclui a regra de amostragem especificada. Você pode especificar o grupo usando o nome do grupo ou o ARN do grupo.

```
aws xray delete-sampling-rule \
    --rule-name polling-scorekeep
```

Saída:

```
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "polling-scorekeep",
            "RuleARN": "arn:aws:xray:us-west-2:123456789012:sampling-rule/
polling-scorekeep",
            "ResourceARN": "*",
            "Priority": 5000,
            "FixedRate": 0.003,
            "ReservoirSize": 0,
```

```
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "GET",
            "URLPath": "/api/state/*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574399.0,
        "ModifiedAt": 1530574399.0
    }
}
```

Para obter mais informações, consulte Como definir configurações de amostragem, grupos e criptografia com a AWS X-Ray API no Guia do AWS X-Ray desenvolvedor.

- Para obter detalhes da API, consulte DeleteSamplingRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.DeleteSamplingRuleRequest;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class DeleteSamplingRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <ruleName>

                Where:
                   ruleName - The name of the rule to delete\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        deleteRule(xRayClient, ruleName);
    }

    public static void deleteRule(XRayClient xRayClient, String ruleName) {
        try {
            DeleteSamplingRuleRequest ruleRequest =
 DeleteSamplingRuleRequest.builder()
                    .ruleName(ruleName)
                    .build();

            xRayClient.deleteSamplingRule(ruleRequest);
        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteSamplingRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteRule(ruleNameVal: String?) {
    val ruleRequest =
        DeleteSamplingRuleRequest {
            ruleName = ruleNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteSamplingRule(ruleRequest)
        println("$ruleNameVal was deleted")
    }
}
```

- Para obter detalhes da API, consulte a DeleteSamplingRulereferência da API AWS SDK for Kotlin.

Use **`GetGroups`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetGroups`.

CLI

AWS CLI

Para recuperar todos os grupos

O exemplo a seguir exibe detalhes de todos os grupos ativos.

```
aws xray get-groups
```

Saída:

```
{
    "Groups": [
        {
            "GroupName": "AdminGroup",
            "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/
AdminGroup/123456789",
            "FilterExpression": "service(\"example.com\") {fault OR error}"
        },
        {
            "GroupName": "SDETGroup",
            "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/
SDETGroup/987654321",
            "FilterExpression": "responsetime > 2"
        }
    ]
}
```

Para obter mais informações, consulte Como definir configurações de amostragem, grupos e criptografia com a AWS X-Ray API no Guia do AWS X-Ray desenvolvedor.

- Para obter detalhes da API, consulte GetGroupsem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetGroupsResponse;
import software.amazon.awssdk.services.xray.model.GroupSummary;
```

```
import software.amazon.awssdk.services.xray.model.XRayException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetGroups {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        getAllGroups(xRayClient);
    }

    public static void getAllGroups(XRayClient xRayClient) {
        try {
            GetGroupsResponse groupsResponse = xRayClient.getGroups();
            List<GroupSummary> groups = groupsResponse.groups();
            for (GroupSummary group : groups) {
                System.out.println("The AWS XRay group name is " +
 group.groupName());
            }

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetGroupsa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAllGroups() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getGroups(GetGroupsRequest {})
        response.groups?.forEach { group ->
            println("The AWS X-Ray group name is ${group.groupName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetGroupsreferência da API AWS SDK for Kotlin.

Use **`GetSamplingRules`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetSamplingRules`.

CLI

AWS CLI

Para recuperar todas as regras de amostragem

O exemplo `get-sampling-rules` a seguir exibe detalhes de todas as regras de amostragem disponíveis:

```
aws xray get-sampling-rules
```

Saída:

```
{
    "SamplingRuleRecords": [
```

```
        {
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.01,
                "ReservoirSize": 0,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1530558121.0
        },
        {
            "SamplingRule": {
                "RuleName": "base-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/base-
scorekeep",
                "ResourceARN": "*",
                "Priority": 9000,
                "FixedRate": 0.1,
                "ReservoirSize": 2,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530573954.0,
            "ModifiedAt": 1530920505.0
        },
        {
            "SamplingRule": {
                "RuleName": "polling-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/polling-
scorekeep",
```

```
                "ResourceARN": "*",
                "Priority": 5000,
                "FixedRate": 0.003,
                "ReservoirSize": 0,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "GET",
                "URLPath": "/api/state/*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530918163.0,
            "ModifiedAt": 1530918163.0
        }
    ]
}
```

Para obter mais informações, consulte Como usar regras de amostragem com a X-Ray API no Guia do AWS X-Ray desenvolvedor.

- Para obter detalhes da API, consulte GetSamplingRulesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetSamplingRulesResponse;
import software.amazon.awssdk.services.xray.model.SamplingRuleRecord;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetSamplingRules {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        getRules(xRayClient);
    }

    public static void getRules(XRayClient xRayClient) {
        try {
            GetSamplingRulesResponse response = xRayClient.getSamplingRules(r ->
 r.build());
            List<SamplingRuleRecord> records = response.samplingRuleRecords();

            for (SamplingRuleRecord record : records) {
                System.out.println("The rule name is: " +
 record.samplingRule().ruleName());
                System.out.println("The related service is: " +
 record.samplingRule().serviceName());
            }
        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetSamplingRulesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getRules() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getSamplingRules(GetSamplingRulesRequest {})
        response.samplingRuleRecords?.forEach { record ->
            println("The rule name is ${record.samplingRule?.ruleName}")
            println("The related service is:
 ${record.samplingRule?.serviceName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetSamplingRulesreferência da API AWS SDK for Kotlin.

Use **`GetServiceGraph`**com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `GetServiceGraph`.

CLI

AWS CLI

Para obter um gráfico de serviços

O exemplo a seguir exibe um documento dentro de um período especificado que descreve os serviços que processam solicitações de entrada e serviços downstream que elas chamam como um resultado:

```
aws xray get-service-graph \
    --start-time 1568835392.0
```

```
    --end-time 1568835446.0
```

Saída:

```
{
    "Services": [
        {
            "ReferenceId": 0,
            "Name": "Scorekeep",
            "Names": [
                "Scorekeep"
            ],
            "Root": true,
            "Type": "AWS::ElasticBeanstalk::Environment",
            "State": "active",
            "StartTime": 1568835392.0,
            "EndTime": 1568835446.0,
            "Edges": [
                {
                    "ReferenceId": 1,
                    "StartTime": 1568835392.0,
                    "EndTime": 1568835446.0,
                    "SummaryStatistics": {
                        "OkCount": 14,
                        "ErrorStatistics": {
                            "ThrottleCount": 0,
                            "OtherCount": 0,
                            "TotalCount": 0
                        },
                        "FaultStatistics": {
                            "OtherCount": 0,
                            "TotalCount": 0
                        },
                        "TotalCount": 14,
                        "TotalResponseTime": 0.13
                    },
                    "ResponseTimeHistogram": [
                        {
                            "Value": 0.008,
                            "Count": 1
                        },
                        {
                            "Value": 0.005,
```

```
                            "Count": 7
                        },
                        {
                            "Value": 0.009,
                            "Count": 1
                        },
                        {
                            "Value": 0.021,
                            "Count": 1
                        },
                        {
                            "Value": 0.038,
                            "Count": 1
                        },
                        {
                            "Value": 0.007,
                            "Count": 1
                        },
                        {
                            "Value": 0.006,
                            "Count": 2
                        }
                    ],
                    "Aliases": []
                },

                ... TRUNCATED FOR BREVITY ...

            ]
        }
    ],
    "StartTime": 1568835392.0,
    "EndTime": 1568835446.0,
    "ContainsOldGroupVersions": false
}
```

Para obter mais informações, consulte Como usar a AWS X-Ray API com a AWS CLI no Guia do AWS X-Ray desenvolvedor.

- Para obter detalhes da API, consulte GetServiceGraphem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetServiceGraphRequest;
import software.amazon.awssdk.services.xray.model.GetServiceGraphResponse;
import software.amazon.awssdk.services.xray.model.Service;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.time.LocalDateTime;
import java.time.Instant;
import java.time.ZoneId;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetServiceGraph {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of a group based on which you want to
 generate a graph.

                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();
        getGraph(xRayClient, groupName);
    }

    public static void getGraph(XRayClient xRayClient, String groupName) {
        try {
            // The Instant values have to be 6 hours apart.
            LocalDateTime localDateTime =
 LocalDateTime.parse("2022-03-09T06:00:00");
            Instant start = localDateTime.atZone(ZoneId.of("America/
New_York")).toInstant();

            LocalDateTime localDateTime2 =
 LocalDateTime.parse("2022-03-09T12:00:00");
            Instant end = localDateTime2.atZone(ZoneId.of("America/
New_York")).toInstant();

            GetServiceGraphRequest getServiceGraphRequest =
 GetServiceGraphRequest.builder()
                    .groupName(groupName)
                    .startTime(start)
                    .endTime(end)
                    .build();

            GetServiceGraphResponse graphResponse =
 xRayClient.getServiceGraph(getServiceGraphRequest);
            List<Service> services = graphResponse.services();
            for (Service service : services) {
                System.out.println("The name of the service is  " +
 service.name());
            }

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte GetServiceGraph a Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getGraph(groupNameVal: String?) {
    val time = aws.smithy.kotlin.runtime.time.Instant
    val getServiceGraphRequest =
        GetServiceGraphRequest {
            groupName = groupNameVal
            this.startTime = time.now()
            endTime = time.now()
        }
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getServiceGraph(getServiceGraphRequest)
        response.services?.forEach { service ->
            println("The name of the service is  ${service.name}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetServiceGraphreferência da API AWS SDK for Kotlin.

Exemplos de código por SDK usando AWS SDKs

Os exemplos de código a seguir mostram como usar Serviços da AWS com um kit de desenvolvimento de AWS software (SDK). Esta seção é organizada pela linguagem do SDK.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Exemplos de código

- Exemplos de código para SDK para .NET
- Exemplos de código para SDK para .NET (v4)
- Exemplos de código para AWS CLI com script Bash
- Exemplos de código para o SDK para C++
- Exemplos de código para SDK para Go V2
- Exemplos de código para SDK for JavaScript (v2)
- Exemplos de código para SDK for JavaScript (v3)
- Exemplos de código para SDK para Java 2.x
- Exemplos de código para SDK para Kotlin

- Exemplos de código para o SDK para PHP.
- Exemplos de código para Tools for PowerShell V5
- Exemplos de código para SDK para Python
- Exemplos de código para o SDK para Python (Boto3)
- Exemplos de código do SDK para Ruby
- Exemplos de código para SDK para Rust
- Exemplos de código para o SDK para SAP ABAP.
- Exemplos de código para o SDK para Swift.

Exemplos de código para SDK para .NET

Os exemplos de código a seguir mostram como usar o AWS SDK para .NET with AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- SDK para .NET Guia do desenvolvedor — Saiba mais sobre como usar o.NET com AWS o.
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos de ACM usando SDK para .NET

- Exemplos de API Gateway usando SDK para .NET
- Exemplos de Aurora usando SDK para .NET
- Exemplos de Auto Scaling usando SDK para .NET
- Exemplos do Amazon Bedrock Runtime usando SDK para .NET
- CloudWatch Exemplos de registros usando SDK para .NET
- Exemplos do Amazon Cognito Identity Provider usando SDK para .NET
- Exemplos do Amazon Comprehend usando SDK para .NET
- Exemplos do Amazon DocumentDB usando SDK para .NET
- Exemplos do DynamoDB usando SDK para .NET
- Exemplos do Amazon EC2 usando SDK para .NET
- Exemplos do Amazon ECS usando SDK para .NET
- Elastic Load Balancing - exemplos da versão 2 usando SDK para .NET
- EventBridge exemplos usando SDK para .NET
- EventBridge Exemplos de agendadores usando SDK para .NET
- Exemplos de uso do Amazon Glacier SDK para .NET
- AWS Glue exemplos usando SDK para .NET
- Exemplos de IAM usando SDK para .NET
- Exemplos de Amazon Keyspaces usando SDK para .NET
- Exemplos de Kinesis usando SDK para .NET
- AWS KMS exemplos usando SDK para .NET
- Exemplos de lambda usando SDK para .NET
- MediaConvert exemplos usando SDK para .NET
- Exemplos de uso do Amazon MSK SDK para .NET
- Exemplos de organizações usando SDK para .NET
- Exemplos do Partner Central usando SDK para .NET
- Exemplos do Amazon Pinpoint usando SDK para .NET
- Exemplos de uso do Amazon Polly SDK para .NET
- Exemplos do Amazon RDS usando SDK para .NET
- Exemplos do Amazon RDS Data Service usando SDK para .NET
- Exemplos do Amazon Rekognition usando SDK para .NET

- Exemplos de registro de domínio do Route 53 usando SDK para .NET
- Exemplos do Amazon S3 usando SDK para .NET
- SageMaker Exemplos de IA usando SDK para .NET
- Exemplos do Secrets Manager usando SDK para .NET
- Exemplos do Amazon SES usando SDK para .NET
- Exemplos da API Amazon SES v2 usando SDK para .NET
- Exemplos do Amazon SNS usando SDK para .NET
- Exemplos do Amazon SQS usando SDK para .NET
- Exemplos de funções de etapa usando SDK para .NET
- AWS STS exemplos usando SDK para .NET
- Suporte exemplos usando SDK para .NET
- Exemplos do Amazon Textract usando SDK para .NET
- Exemplos do Amazon Transcribe usando SDK para .NET
- Exemplos do Amazon Translate usando SDK para .NET

Exemplos de ACM usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o ACM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DescribeCertificate`

O código de exemplo a seguir mostra como usar `DescribeCertificate`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;

namespace DescribeCertificate
{
    class DescribeCertificate
    {
        // The following example retrieves and displays the metadata for a
        // certificate using the AWS Certificate Manager (ACM) service.

        // Specify your AWS Region (an example Region is shown).
        private static readonly RegionEndpoint ACMRegion = RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            _client = new
 Amazon.CertificateManager.AmazonCertificateManagerClient(ACMRegion);

            var describeCertificateReq = new DescribeCertificateRequest();
            // The ARN used here is just an example. Replace it with the ARN of
            // a certificate that exists on your account.
            describeCertificateReq.CertificateArn =
                "arn:aws:acm:us-
east-1:123456789012:certificate/8cfd7dae-9b6a-2d07-92bc-1c309EXAMPLE";

            var certificateDetailResp =
                DescribeCertificateResponseAsync(client: _client, request:
 describeCertificateReq);
            var certificateDetail = certificateDetailResp.Result.Certificate;
```

```
            if (certificateDetail is not null)
            {
                DisplayCertificateDetails(certificateDetail);
            }
        }

        /// <summary>
        /// Displays detailed metadata about a certificate retrieved
        /// using the ACM service.
        /// </summary>
        /// <param name="certificateDetail">The object that contains details
        /// returned from the call to DescribeCertificateAsync.</param>
        static void DisplayCertificateDetails(CertificateDetail certificateDetail)
        {
            Console.WriteLine("\nCertificate Details: ");
            Console.WriteLine($"Certificate Domain:
{certificateDetail.DomainName}");
            Console.WriteLine($"Certificate Arn:
{certificateDetail.CertificateArn}");
            Console.WriteLine($"Certificate Subject: {certificateDetail.Subject}");
            Console.WriteLine($"Certificate Status: {certificateDetail.Status}");
            foreach (var san in certificateDetail.SubjectAlternativeNames)
            {
                Console.WriteLine($"Certificate SubjectAlternativeName: {san}");
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with the ACM service certificate.
        /// </summary>
        /// <param name="client">An AmazonCertificateManagerClient object
        /// used to call DescribeCertificateResponse.</param>
        /// <param name="request">The DescribeCertificateRequest object that
        /// will be passed to the method call.</param>
        /// <returns></returns>
        static async Task<DescribeCertificateResponse>
DescribeCertificateResponseAsync(
            AmazonCertificateManagerClient client, DescribeCertificateRequest
request)
        {
            var response = new DescribeCertificateResponse();

            try
            {
```

```
                response = await client.DescribeCertificateAsync(request);
            }
            catch (InvalidArnException)
            {
                Console.WriteLine($"Error: The ARN specified is invalid.");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine($"Error: The specified certificate could not be
 found.");
            }

            return response;
        }
    }

}
```

- Para obter detalhes da API, consulte DescribeCertificate em Referência AWS SDK para .NET da API.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;
```

```
namespace ListCertificates
{
    // The following example retrieves and displays a list of the
    // certificates defined for the default account using the AWS
    // Certificate Manager (ACM) service.
    class ListCertificates
    {
        // Specify your AWS Region (an example Region is shown).

        private static readonly RegionEndpoint ACMRegion = RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            _client = new AmazonCertificateManagerClient(ACMRegion);
            var certificateList = ListCertificatesResponseAsync(client: _client);

            Console.WriteLine("Certificate Summary List\n");

            foreach (var certificate in
 certificateList.Result.CertificateSummaryList)
            {
                Console.WriteLine($"Certificate Domain: {certificate.DomainName}");
                Console.WriteLine($"Certificate ARN:
 {certificate.CertificateArn}\n");
            }
        }

        /// <summary>
        /// Retrieves a list of the certificates defined in this Region.
        /// </summary>
        /// <param name="client">The ACM client object passed to the
        /// ListCertificateResAsync method call.</param>
        /// <param name="request"></param>
        /// <returns>The ListCertificatesResponse.</returns>
        static async Task<ListCertificatesResponse> ListCertificatesResponseAsync(
            AmazonCertificateManagerClient client)
        {
            var request = new ListCertificatesRequest();

            var response = await client.ListCertificatesAsync(request);
            return response;
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte ListCertificates em Referência AWS SDK para .NET da API.

Exemplos de API Gateway usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com API Gateway.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários
- AWS contribuições da comunidade

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para .NET

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK .NET.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos de Aurora usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Aurora.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para .NET

Mostra como usar o AWS SDK para .NET para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful .NET.

- Integre um aplicativo web React com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos de Auto Scaling usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AttachLoadBalancerTargetGroups`

O código de exemplo a seguir mostra como usar `AttachLoadBalancerTargetGroups`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }
```

- Para obter detalhes da API, consulte AttachLoadBalancerTargetGroups em Referência AWS SDK para .NET da API.

`DeleteAutoScalingGroup`

O código de exemplo a seguir mostra como usar `DeleteAutoScalingGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize o tamanho mínimo de um grupo do Auto Scaling para zero, encerre todas as instâncias no grupo e exclua o grupo.

```
    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
```

```
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
```

```
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
```

```
    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
 _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroup em Referência AWS SDK para .NET da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper = host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
```

```
            "\nFor this demo, we'll use the AWS SDK for .NET to create several AWS
resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways to
make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs 'server_startup_script.sh'
when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the `server.py`
script. The web server\n"
```

```
            + "listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2 instances,
each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3, _autoScalerWrapper.GroupName,
zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or continue
with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the demo.");
        await _smParameterWrapper.Reset();
```

```
        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
 The target group\n"
            + "defines how the load balancer connects to instances. The load
 balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroupN
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.LoadB
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper.Lo
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port, ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);
```

```
            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for your
 default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management Console.
\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port, ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
 ipString);
                }
            }
            loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
 browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
```

```
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite of
these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
```

```
                            $"The table name is contained in a Systems Manager
 parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                            $"To simulate a failure of the recommendation service,
 let's set this parameter to name a non-existent table.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as\n" +
                            "healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service fails,
 the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
 "static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
 checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
 _smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in the
 target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
```

```
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile that
contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns either
a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo, a
deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not for
Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the Auto
Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load balancer
can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");
```

```
        Console.WriteLine($"\nNow, checking target health indicates that the
 instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two for
 the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
 endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
 rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by an
 auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler start
 a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
 instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a successful
 recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the load
 balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance typically
 takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
 instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
 checks mean all instances report as unhealthy.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
 continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
 return a static response rather than fail");
```

```
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account, we
can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBalanc
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGroup
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupName)
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
```

```
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
 incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
```

```
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
```

```
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
```

```
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
```

```
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
```

```
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
```

```
   /// the instance is started. This script installs the Python packages and starts
a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
```

```
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"An error occurred while listing availability zones.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with size
 {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
```

```
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
```

```
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
```

```
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
```

```
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
```

```
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
```

```
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the calling
computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP address
while running this example.
    /// If you do, be sure to remove public access when you're done.
```

```
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be open
to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }
```

```
    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
 targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
```

```
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
```

```
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
```

```
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
```

```
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
```

```
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
```

```
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
```

```
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

```
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books, movies,
 and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
```

```
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
```

```
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await File.ReadAllTextAsync(recommendationsPath);
        var records =
            JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
```

```
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-architecture-
table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
```

```
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Exemplos do Amazon Bedrock Runtime usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET Amazon Bedrock Runtime.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Cenários
- Amazônia Nova
- Amazon Nova Canvas
- Stable Image Core

Conceitos básicos

Olá, Amazon Bedrock Runtime

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock Runtime.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

namespace BedrockRuntimeActions;
```

```
/// <summary>
/// This example shows how to use the Converse API to send a text message
/// to Amazon Bedrock.
/// </summary>
internal class HelloBedrockRuntime
{
    static async Task Main(string[] args)
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // You can change the region to match your setup.
        var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USWest2);

        // Set the model ID, e.g., Claude Haiku.
        // The "global." prefix enables cross-region inference, allowing the request
        // to be routed to the nearest available region for the specified model.
        var modelId = "global.anthropic.claude-haiku-4-5-20251001-v1:0";

        // Define the user message.
        var userMessage = "Hi. In a short paragraph, explain what you can do.";

        // Create a request with the model ID, the user message, and an inference
 configuration.
        var request = new ConverseRequest
        {
            ModelId = modelId,
            Messages = new List<Message>
            {
                new Message
                {
                    Role = ConversationRole.User,
                    Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
                }
            }
        };

        try
        {
            // Send the request to the Bedrock Runtime and wait for the response.
            var response = await client.ConverseAsync(request);

            // Extract and print the response text.
            string responseText = response?.Output?.Message?.Content?[0]?.Text ??
 "";
```

```
            Console.WriteLine(responseText);
        }
        catch (AmazonBedrockRuntimeException e)
        {
            Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason:
 {e.Message}");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Cenários

Criar um aplicativo playground para interagir com os modelos de base do Amazon Bedrock

O exemplo de código a seguir mostra como criar playgrounds para interagir com os modelos de base do Amazon Bedrock por meio de diferentes modalidades.

SDK para .NET

O .NET Foundation Model (FM) Playground é um aplicativo de amostra do .NET MAUI Blazor que mostra como usar o Amazon Bedrock a partir do código C#. Este exemplo mostra como os desenvolvedores do.NET e C# podem usar o Amazon Bedrock para criar aplicativos AI-enabled generativos. É possível testar e interagir com os modelos de base do Amazon Bedrock usando os quatro playgrounds a seguir:

- Um playground de texto.
- Um playground de chat.
- Um playground de chat por voz.
- Um playground de imagens.

O exemplo também lista e exibe os modelos de base aos quais você tem acesso e respectivas características. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock Runtime

Uso de ferramenta com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;

namespace ConverseToolScenario;

public static class ConverseToolScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
```

```
    The script interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
    input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
   */

    public static BedrockActionsWrapper _bedrockActionsWrapper = null!;
    public static WeatherTool _weatherTool = null!;
    public static bool _interactive = true;

    // Change this string to use a different model with Converse API.
    private static string model_id = "amazon.nova-lite-v1:0";

    private static string system_prompt = @"
        You are a weather assistant that provides current weather data for user-
specified locations using only
        the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
        If the user specifies a state, country, or region, infer the locations of
 cities within that state.
        If the user provides coordinates, infer the approximate location and refer
 to it in your response.
        To use the tool, you strictly apply the provided tool specification.

        - Explain your step-by-step process, and give brief updates before each
 step.
        - Only use the Weather_Tool for data. Never guess or make up information.
        - Repeat the tool use for subsequent requests if necessary.
        - If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
        - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
          emojis where appropriate.
        - Only respond to weather queries. Remind off-topic users of your purpose.
        - Never claim to search online, access external data, or use tools besides
 Weather_Tool.
        - Complete the entire process until you have all required data before
 sending the complete response.
    "
    ;

    private static string default_prompt = "What is the weather like in Seattle?";

    // The maximum number of recursive calls allowed in the tool use function.
```

```
    // This helps prevent infinite loops and potential performance issues.
    private static int max_recursions = 5;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Error)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddHttpClient()
                    .AddSingleton<IAmazonBedrockRuntime>(_ => new
AmazonBedrockRuntimeClient(RegionEndpoint.USEast1)) // Specify a region that has
access to the chosen model.
                    .AddTransient<BedrockActionsWrapper>()
                    .AddTransient<WeatherTool>()
                    .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            await RunConversationAsync();

        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            Console.WriteLine(new string('-', 80));
        }
        finally
        {
            Console.WriteLine(
                "Amazon Bedrock Converse API with Tool Use Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
    }
```

```
    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _bedrockActionsWrapper =
host.Services.GetRequiredService<BedrockActionsWrapper>();
        _weatherTool = host.Services.GetRequiredService<WeatherTool>();
    }

    /// <summary>
    /// Starts the conversation with the user and handles the interaction with
Bedrock.
    /// </summary>
    /// <returns>The conversation array.</returns>
    public static async Task<List<Message>> RunConversationAsync()
    {
        // Print the greeting and a short user guide
        PrintHeader();

        // Start with an empty conversation
        var conversation = new List<Message>();

        // Get the first user input
        var userInput = await GetUserInputAsync();

        while (userInput != null)
        {
            // Create a new message with the user input and append it to the
conversation
            var message = new Message { Role = ConversationRole.User, Content = new
List<ContentBlock> { new ContentBlock { Text = userInput } } };
            conversation.Add(message);

            // Send the conversation to Amazon Bedrock
            var bedrockResponse = await SendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
            await ProcessModelResponseAsync(bedrockResponse, conversation,
max_recursions);

            // Repeat the loop until the user decides to exit the application
```

```
            userInput = await GetUserInputAsync();
        }

        PrintFooter();
        return conversation;
    }

    /// <summary>
    /// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
    /// </summary>
    /// <param name="conversation">The conversation history including the next
 message to send.</param>
    /// <returns>The response from Amazon Bedrock.</returns>
    private static async Task<ConverseResponse>
 SendConversationToBedrock(List<Message> conversation)
    {
        Console.WriteLine("\tCalling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and return
 the response
        return await _bedrockActionsWrapper.SendConverseRequestAsync(model_id,
 system_prompt, conversation, _weatherTool.GetToolSpec());
    }

    /// <summary>
    /// Processes the response received via Amazon Bedrock and performs the
 necessary actions based on the stop reason.
    /// </summary>
    /// <param name="modelResponse">The model's response returned via Amazon
 Bedrock.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls allowed.</
param>
    private static async Task ProcessModelResponseAsync(ConverseResponse
 modelResponse, List<Message> conversation, int maxRecursion)
    {
        if (maxRecursion <= 0)
        {
            // Stop the process, the number of recursive calls could indicate an
 infinite loop
            Console.WriteLine("\tWarning: Maximum number of recursions reached.
 Please try again.");
        }
```

```
        // Append the model's response to the ongoing conversation
        conversation.Add(modelResponse.Output.Message);

        if (modelResponse.StopReason == "tool_use")
        {
            // If the stop reason is "tool_use", forward everything to the tool use
 handler
            await HandleToolUseAsync(modelResponse.Output, conversation,
 maxRecursion - 1);
        }

        if (modelResponse.StopReason == "end_turn")
        {
            // If the stop reason is "end_turn", print the model's response text,
 and finish the process
            PrintModelResponse(modelResponse.Output.Message.Content[0].Text);
            if (!_interactive)
            {
                default_prompt = "x";
            }
        }
    }

    /// <summary>
    /// Handles the tool use case by invoking the specified tool and sending the
 tool's response back to Bedrock.
    /// The tool response is appended to the conversation, and the conversation is
 sent back to Amazon Bedrock for further processing.
    /// </summary>
    /// <param name="modelResponse">The model's response containing the tool use
 request.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls allowed.</
param>
    public static async Task HandleToolUseAsync(ConverseOutput modelResponse,
 List<Message> conversation, int maxRecursion)
    {
        // Initialize an empty list of tool results
        var toolResults = new List<ContentBlock>();

        // The model's response can consist of multiple content blocks
        foreach (var contentBlock in modelResponse.Message.Content)
        {
```

```
            if (!String.IsNullOrEmpty(contentBlock.Text))
            {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.Text);
            }

            if (contentBlock.ToolUse != null)
            {
                // If the content block is a tool use request, forward it to the
tool
                var toolResponse = await InvokeTool(contentBlock.ToolUse);

                // Add the tool use ID and the tool's response to the list of
results
                toolResults.Add(new ContentBlock
                {
                    ToolResult = new ToolResultBlock()
                    {
                        ToolUseId = toolResponse.ToolUseId,
                        Content = new List<ToolResultContentBlock>()
                            { new ToolResultContentBlock { Json =
toolResponse.Content } }
                    }
                });
            }
        }

        // Embed the tool results in a new user message
        var message = new Message() { Role = ConversationRole.User, Content =
toolResults };

        // Append the new message to the ongoing conversation
        conversation.Add(message);

        // Send the conversation to Amazon Bedrock
        var response = await SendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        await ProcessModelResponseAsync(response, conversation, maxRecursion);
    }

    /// <summary>
```

```
    /// Invokes the specified tool with the given payload and returns the tool's
 response.
    /// If the requested tool does not exist, an error message is returned.
    /// </summary>
    /// <param name="payload">The payload containing the tool name and input data.</
param>
    /// <returns>The tool's response or an error message.</returns>
    public static async Task<ToolResponse> InvokeTool(ToolUseBlock payload)
    {
        var toolName = payload.Name;

        if (toolName == "Weather_Tool")
        {
            var inputData = payload.Input.AsDictionary();
            PrintToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            var weatherResponse = await
 _weatherTool.FetchWeatherDataAsync(inputData["latitude"].ToString(),
 inputData["longitude"].ToString());
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
 weatherResponse };
        }
        else
        {
            var errorMessage = $"\tThe requested tool with name '{toolName}' does
 not exist.";
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content = new
 { error = true, message = errorMessage } };
        }
    }


    /// <summary>
    /// Prompts the user for input and returns the user's response.
    /// Returns null if the user enters 'x' to exit.
    /// </summary>
    /// <param name="prompt">The prompt to display to the user.</param>
    /// <returns>The user's input or null if the user chooses to exit.</returns>
    private static async Task<string?> GetUserInputAsync(string prompt = "\tYour
 weather info request:")
    {
        var userInput = default_prompt;
        if (_interactive)
```

```
        {
            Console.WriteLine(new string('*', 80));
            Console.WriteLine($"{prompt} (x to exit): \n\t");
            userInput = Console.ReadLine();
        }

        if (string.IsNullOrWhiteSpace(userInput))
        {
            prompt = "\tPlease enter your weather info request, e.g. the name of a
city";
            return await GetUserInputAsync(prompt);
        }

        if (userInput.ToLowerInvariant() == "x")
        {
            return null;
        }

        return userInput;
    }

    /// <summary>
    /// Logs the welcome message and usage guide for the tool use demo.
    /// </summary>
    public static void PrintHeader()
    {
        Console.WriteLine(@"
        ==================================================
        Welcome to the Amazon Bedrock Tool Use demo!
        ==================================================

        This assistant provides current weather information for user-specified
locations.
        You can ask for weather details by providing the location name or
coordinates. Weather information
        will be provided using a custom Tool and open-meteo API.

        Example queries:
        - What's the weather like in New York?
        - Current weather for latitude 40.70, longitude -74.01
        - Is it warmer in Rome or Barcelona today?

        To exit the program, simply type 'x' and press Enter.
```

```
        P.S.: You're not limited to single locations, or even to using English!
        Have fun and experiment with the app!
        ");
    }

    /// <summary>
    /// Logs the footer information for the tool use demo.
    /// </summary>
    public static void PrintFooter()
    {
        Console.WriteLine(@"
        ==================================================
        Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you
        learned something new, or got some inspiration for your own apps today!

        For more Bedrock examples in different programming languages, have a look
 at:
        https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
        ==================================================
        ");
    }

    /// <summary>
    /// Logs information about the tool use.
    /// </summary>
    /// <param name="toolName">The name of the tool being used.</param>
    /// <param name="inputData">The input data for the tool.</param>
    public static void PrintToolUse(string toolName, Dictionary<string, Document>
 inputData)
    {
        Console.WriteLine($"\n\tInvoking tool: {toolName} with input:
 {inputData["latitude"].ToString()}, {inputData["longitude"].ToString()}...\n");
    }

    /// <summary>
    /// Logs the model's response.
    /// </summary>
    /// <param name="message">The model's response message.</param>
    public static void PrintModelResponse(string message)
    {
        Console.WriteLine("\tThe model's response:\n");
        Console.WriteLine(message);
        Console.WriteLine();
```

```
    }
}
```

A ferramenta de meteorologia usada pela demonstração. Esse arquivo define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.Logging;

namespace ConverseToolScenario;

/// <summary>
/// Weather tool that will be invoked when requested by the Bedrock response.
/// </summary>
public class WeatherTool
{
    private readonly ILogger<WeatherTool> _logger;
    private readonly IHttpClientFactory _httpClientFactory;

    public WeatherTool(ILogger<WeatherTool> logger, IHttpClientFactory
 httpClientFactory)
    {
        _logger = logger;
        _httpClientFactory = httpClientFactory;
    }

    /// <summary>
    /// Returns the JSON Schema specification for the Weather tool. The tool
 specification
    /// defines the input schema and describes the tool's functionality.
    /// For more information, see https://json-schema.org/understanding-json-schema/
reference.
    /// </summary>
    /// <returns>The tool specification for the Weather tool.</returns>
    public ToolSpecification GetToolSpec()
    {
        ToolSpecification toolSpecification = new ToolSpecification();

        toolSpecification.Name = "Weather_Tool";
```

```
        toolSpecification.Description = "Get the current weather for a given
 location, based on its WGS84 coordinates.";

        Document toolSpecDocument = Document.FromObject(
            new
            {
                type = "object",
                properties = new
                {
                    latitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 latitude of the location."
                    },
                    longitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 longitude of the
 location."
                    }
                },
                required = new[] { "latitude", "longitude" }
            });

        toolSpecification.InputSchema = new ToolInputSchema() { Json =
 toolSpecDocument };
        return toolSpecification;
    }

    /// <summary>
    /// Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    /// Returns the weather data or an error message if the request fails.
    /// </summary>
    /// <param name="latitude">The latitude of the location.</param>
    /// <param name="longitude">The longitude of the location.</param>
    /// <returns>The weather data or an error message.</returns>
    public async Task<Document> FetchWeatherDataAsync(string latitude, string
 longitude)
    {
        string endpoint = "https://api.open-meteo.com/v1/forecast";

        try
        {
```

```
            var httpClient = _httpClientFactory.CreateClient();
            var response = await httpClient.GetAsync($"{endpoint}?
latitude={latitude}&longitude={longitude}&current_weather=True");
            response.EnsureSuccessStatusCode();
            var weatherData = await response.Content.ReadAsStringAsync();

            Document weatherDocument = Document.FromObject(
                new { weather_data = weatherData });

            return weatherDocument;
        }
        catch (HttpRequestException e)
        {
            _logger.LogError(e, "Error fetching weather data: {Message}",
 e.Message);
            throw;
        }
        catch (Exception e)
        {
            _logger.LogError(e, "Unexpected error fetching weather data: {Message}",
 e.Message);
            throw;
        }
    }
}
```

A ação da API Converse com uma configuração de ferramenta.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/> class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
```

```
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };
```

```
            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse request.");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Amazônia Nova

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Amazon Nova.

using System;
using System.Collections.Generic;
using Amazon;
```

```
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Amazon Nova Lite.
var modelId = "amazon.nova-lite-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
```

```
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

Envie uma conversa de mensagens para o Amazon Nova usando a API Converse do Bedrock com uma configuração de ferramenta.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/> class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
```

```
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse request.");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processar o fluxo de respostas em tempo real.

SDK para .NET

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
// Use the Converse API to send a text message to Amazon Nova
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Amazon Nova Lite.
var modelId = "amazon.nova-lite-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
```

```
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para .NET da API.

Cenário: uso de ferramentas com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;

namespace ConverseToolScenario;

public static class ConverseToolScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
```

```
    The script interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
    input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
   */

    public static BedrockActionsWrapper _bedrockActionsWrapper = null!;
    public static WeatherTool _weatherTool = null!;
    public static bool _interactive = true;

    // Change this string to use a different model with Converse API.
    private static string model_id = "amazon.nova-lite-v1:0";

    private static string system_prompt = @"
        You are a weather assistant that provides current weather data for user-
specified locations using only
        the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
        If the user specifies a state, country, or region, infer the locations of
 cities within that state.
        If the user provides coordinates, infer the approximate location and refer
 to it in your response.
        To use the tool, you strictly apply the provided tool specification.

        - Explain your step-by-step process, and give brief updates before each
 step.
        - Only use the Weather_Tool for data. Never guess or make up information.
        - Repeat the tool use for subsequent requests if necessary.
        - If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
        - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
          emojis where appropriate.
        - Only respond to weather queries. Remind off-topic users of your purpose.
        - Never claim to search online, access external data, or use tools besides
 Weather_Tool.
        - Complete the entire process until you have all required data before
 sending the complete response.
    "
    ;

    private static string default_prompt = "What is the weather like in Seattle?";

    // The maximum number of recursive calls allowed in the tool use function.
```

```
    // This helps prevent infinite loops and potential performance issues.
    private static int max_recursions = 5;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Error)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddHttpClient()
                    .AddSingleton<IAmazonBedrockRuntime>(_ => new
AmazonBedrockRuntimeClient(RegionEndpoint.USEast1)) // Specify a region that has
access to the chosen model.
                    .AddTransient<BedrockActionsWrapper>()
                    .AddTransient<WeatherTool>()
                    .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            await RunConversationAsync();

        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            Console.WriteLine(new string('-', 80));
        }
        finally
        {
            Console.WriteLine(
                "Amazon Bedrock Converse API with Tool Use Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
    }
```

```
    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _bedrockActionsWrapper =
host.Services.GetRequiredService<BedrockActionsWrapper>();
        _weatherTool = host.Services.GetRequiredService<WeatherTool>();
    }

    /// <summary>
    /// Starts the conversation with the user and handles the interaction with
Bedrock.
    /// </summary>
    /// <returns>The conversation array.</returns>
    public static async Task<List<Message>> RunConversationAsync()
    {
        // Print the greeting and a short user guide
        PrintHeader();

        // Start with an empty conversation
        var conversation = new List<Message>();

        // Get the first user input
        var userInput = await GetUserInputAsync();

        while (userInput != null)
        {
            // Create a new message with the user input and append it to the
conversation
            var message = new Message { Role = ConversationRole.User, Content = new
List<ContentBlock> { new ContentBlock { Text = userInput } } };
            conversation.Add(message);

            // Send the conversation to Amazon Bedrock
            var bedrockResponse = await SendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
            await ProcessModelResponseAsync(bedrockResponse, conversation,
max_recursions);

            // Repeat the loop until the user decides to exit the application
```

```
            userInput = await GetUserInputAsync();
        }

        PrintFooter();
        return conversation;
    }

    /// <summary>
    /// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
    /// </summary>
    /// <param name="conversation">The conversation history including the next
 message to send.</param>
    /// <returns>The response from Amazon Bedrock.</returns>
    private static async Task<ConverseResponse>
 SendConversationToBedrock(List<Message> conversation)
    {
        Console.WriteLine("\tCalling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and return
 the response
        return await _bedrockActionsWrapper.SendConverseRequestAsync(model_id,
 system_prompt, conversation, _weatherTool.GetToolSpec());
    }

    /// <summary>
    /// Processes the response received via Amazon Bedrock and performs the
 necessary actions based on the stop reason.
    /// </summary>
    /// <param name="modelResponse">The model's response returned via Amazon
 Bedrock.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls allowed.</
param>
    private static async Task ProcessModelResponseAsync(ConverseResponse
 modelResponse, List<Message> conversation, int maxRecursion)
    {
        if (maxRecursion <= 0)
        {
            // Stop the process, the number of recursive calls could indicate an
 infinite loop
            Console.WriteLine("\tWarning: Maximum number of recursions reached.
 Please try again.");
        }
```

```
        // Append the model's response to the ongoing conversation
        conversation.Add(modelResponse.Output.Message);

        if (modelResponse.StopReason == "tool_use")
        {
            // If the stop reason is "tool_use", forward everything to the tool use
 handler
            await HandleToolUseAsync(modelResponse.Output, conversation,
 maxRecursion - 1);
        }

        if (modelResponse.StopReason == "end_turn")
        {
            // If the stop reason is "end_turn", print the model's response text,
 and finish the process
            PrintModelResponse(modelResponse.Output.Message.Content[0].Text);
            if (!_interactive)
            {
                default_prompt = "x";
            }
        }
    }

    /// <summary>
    /// Handles the tool use case by invoking the specified tool and sending the
 tool's response back to Bedrock.
    /// The tool response is appended to the conversation, and the conversation is
 sent back to Amazon Bedrock for further processing.
    /// </summary>
    /// <param name="modelResponse">The model's response containing the tool use
 request.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls allowed.</
param>
    public static async Task HandleToolUseAsync(ConverseOutput modelResponse,
 List<Message> conversation, int maxRecursion)
    {
        // Initialize an empty list of tool results
        var toolResults = new List<ContentBlock>();

        // The model's response can consist of multiple content blocks
        foreach (var contentBlock in modelResponse.Message.Content)
        {
```

```
            if (!String.IsNullOrEmpty(contentBlock.Text))
            {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.Text);
            }

            if (contentBlock.ToolUse != null)
            {
                // If the content block is a tool use request, forward it to the
tool
                var toolResponse = await InvokeTool(contentBlock.ToolUse);

                // Add the tool use ID and the tool's response to the list of
results
                toolResults.Add(new ContentBlock
                {
                    ToolResult = new ToolResultBlock()
                    {
                        ToolUseId = toolResponse.ToolUseId,
                        Content = new List<ToolResultContentBlock>()
                            { new ToolResultContentBlock { Json =
toolResponse.Content } }
                    }
                });
            }
        }

        // Embed the tool results in a new user message
        var message = new Message() { Role = ConversationRole.User, Content =
toolResults };

        // Append the new message to the ongoing conversation
        conversation.Add(message);

        // Send the conversation to Amazon Bedrock
        var response = await SendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        await ProcessModelResponseAsync(response, conversation, maxRecursion);
    }

    /// <summary>
```

```
    /// Invokes the specified tool with the given payload and returns the tool's
 response.
    /// If the requested tool does not exist, an error message is returned.
    /// </summary>
    /// <param name="payload">The payload containing the tool name and input data.</
param>
    /// <returns>The tool's response or an error message.</returns>
    public static async Task<ToolResponse> InvokeTool(ToolUseBlock payload)
    {
        var toolName = payload.Name;

        if (toolName == "Weather_Tool")
        {
            var inputData = payload.Input.AsDictionary();
            PrintToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            var weatherResponse = await
 _weatherTool.FetchWeatherDataAsync(inputData["latitude"].ToString(),
 inputData["longitude"].ToString());
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
 weatherResponse };
        }
        else
        {
            var errorMessage = $"\tThe requested tool with name '{toolName}' does
 not exist.";
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content = new
 { error = true, message = errorMessage } };
        }
    }


    /// <summary>
    /// Prompts the user for input and returns the user's response.
    /// Returns null if the user enters 'x' to exit.
    /// </summary>
    /// <param name="prompt">The prompt to display to the user.</param>
    /// <returns>The user's input or null if the user chooses to exit.</returns>
    private static async Task<string?> GetUserInputAsync(string prompt = "\tYour
 weather info request:")
    {
        var userInput = default_prompt;
        if (_interactive)
```

```
        {
            Console.WriteLine(new string('*', 80));
            Console.WriteLine($"{prompt} (x to exit): \n\t");
            userInput = Console.ReadLine();
        }

        if (string.IsNullOrWhiteSpace(userInput))
        {
            prompt = "\tPlease enter your weather info request, e.g. the name of a
city";
            return await GetUserInputAsync(prompt);
        }

        if (userInput.ToLowerInvariant() == "x")
        {
            return null;
        }

        return userInput;
    }

    /// <summary>
    /// Logs the welcome message and usage guide for the tool use demo.
    /// </summary>
    public static void PrintHeader()
    {
        Console.WriteLine(@"
        ==================================================
        Welcome to the Amazon Bedrock Tool Use demo!
        ==================================================

        This assistant provides current weather information for user-specified
locations.
        You can ask for weather details by providing the location name or
coordinates. Weather information
        will be provided using a custom Tool and open-meteo API.

        Example queries:
        - What's the weather like in New York?
        - Current weather for latitude 40.70, longitude -74.01
        - Is it warmer in Rome or Barcelona today?

        To exit the program, simply type 'x' and press Enter.
```

```
        P.S.: You're not limited to single locations, or even to using English!
        Have fun and experiment with the app!
        ");
    }

    /// <summary>
    /// Logs the footer information for the tool use demo.
    /// </summary>
    public static void PrintFooter()
    {
        Console.WriteLine(@"
        ==================================================
        Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you
        learned something new, or got some inspiration for your own apps today!

        For more Bedrock examples in different programming languages, have a look
 at:
        https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
        ==================================================
        ");
    }

    /// <summary>
    /// Logs information about the tool use.
    /// </summary>
    /// <param name="toolName">The name of the tool being used.</param>
    /// <param name="inputData">The input data for the tool.</param>
    public static void PrintToolUse(string toolName, Dictionary<string, Document>
 inputData)
    {
        Console.WriteLine($"\n\tInvoking tool: {toolName} with input:
 {inputData["latitude"].ToString()}, {inputData["longitude"].ToString()}...\n");
    }

    /// <summary>
    /// Logs the model's response.
    /// </summary>
    /// <param name="message">The model's response message.</param>
    public static void PrintModelResponse(string message)
    {
        Console.WriteLine("\tThe model's response:\n");
        Console.WriteLine(message);
        Console.WriteLine();
```

```
    }
}
```

A ferramenta de meteorologia usada pela demonstração. Esse arquivo define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.Logging;

namespace ConverseToolScenario;

/// <summary>
/// Weather tool that will be invoked when requested by the Bedrock response.
/// </summary>
public class WeatherTool
{
    private readonly ILogger<WeatherTool> _logger;
    private readonly IHttpClientFactory _httpClientFactory;

    public WeatherTool(ILogger<WeatherTool> logger, IHttpClientFactory
 httpClientFactory)
    {
        _logger = logger;
        _httpClientFactory = httpClientFactory;
    }

    /// <summary>
    /// Returns the JSON Schema specification for the Weather tool. The tool
 specification
    /// defines the input schema and describes the tool's functionality.
    /// For more information, see https://json-schema.org/understanding-json-schema/
reference.
    /// </summary>
    /// <returns>The tool specification for the Weather tool.</returns>
    public ToolSpecification GetToolSpec()
    {
        ToolSpecification toolSpecification = new ToolSpecification();

        toolSpecification.Name = "Weather_Tool";
```

```
        toolSpecification.Description = "Get the current weather for a given
 location, based on its WGS84 coordinates.";

        Document toolSpecDocument = Document.FromObject(
            new
            {
                type = "object",
                properties = new
                {
                    latitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 latitude of the location."
                    },
                    longitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 longitude of the
 location."
                    }
                },
                required = new[] { "latitude", "longitude" }
            });

        toolSpecification.InputSchema = new ToolInputSchema() { Json =
 toolSpecDocument };
        return toolSpecification;
    }

    /// <summary>
    /// Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    /// Returns the weather data or an error message if the request fails.
    /// </summary>
    /// <param name="latitude">The latitude of the location.</param>
    /// <param name="longitude">The longitude of the location.</param>
    /// <returns>The weather data or an error message.</returns>
    public async Task<Document> FetchWeatherDataAsync(string latitude, string
 longitude)
    {
        string endpoint = "https://api.open-meteo.com/v1/forecast";

        try
        {
```

```
            var httpClient = _httpClientFactory.CreateClient();
            var response = await httpClient.GetAsync($"{endpoint}?
latitude={latitude}&longitude={longitude}&current_weather=True");
            response.EnsureSuccessStatusCode();
            var weatherData = await response.Content.ReadAsStringAsync();

            Document weatherDocument = Document.FromObject(
                new { weather_data = weatherData });

            return weatherDocument;
        }
        catch (HttpRequestException e)
        {
            _logger.LogError(e, "Error fetching weather data: {Message}",
 e.Message);
            throw;
        }
        catch (Exception e)
        {
            _logger.LogError(e, "Unexpected error fetching weather data: {Message}",
 e.Message);
            throw;
        }
    }
}
```

A ação da API Converse com uma configuração de ferramenta.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/> class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
```

```
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };
```

```
            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse request.");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Amazon Nova Canvas

InvokeModel

O exemplo de código a seguir mostra como invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
// Use the native inference API to create an image with Amazon Nova Canvas.

using System;
using System.IO;
using System.Text.Json;
```

```
using System.Text.Json.Nodes;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID.
var modelId = "amazon.nova-canvas-v1:0";

// Define the image generation prompt for the model.
var prompt = "A stylized picture of a cute old steampunk robot.";

// Create a random seed between 0 and 858,993,459
int seed = new Random().Next(0, 858993460);

//Format the request payload using the model's native structure.
var nativeRequest = JsonSerializer.Serialize(new
{
    taskType = "TEXT_IMAGE",
    textToImageParams = new
    {
        text = prompt
    },
    imageGenerationConfig = new
    {
        seed,
        quality = "standard",
        width = 512,
        height = 512,
        numberOfImages = 1
    }
});

// Create a request with the model ID and the model's native request payload.
var request = new InvokeModelRequest()
{
    ModelId = modelId,
    Body = new MemoryStream(System.Text.Encoding.UTF8.GetBytes(nativeRequest)),
    ContentType = "application/json"
};

try
```

```
{
    // Send the request to the Bedrock Runtime and wait for the response.
    var response = await client.InvokeModelAsync(request);

    // Decode the response body.
    var modelResponse = await JsonNode.ParseAsync(response.Body);

    // Extract the image data.
    var base64Image = modelResponse["images"]?[0].ToString() ?? "";

    // Save the image in a local folder
    string savedPath = AmazonNovaCanvas.InvokeModel.SaveBase64Image(base64Image);
    Console.WriteLine($"Image saved to: {savedPath}");
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK para .NET da API.

Stable Image Core

InvokeModel

O exemplo de código a seguir mostra como invocar o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Image Core.

```
        /// <summary>
        /// Asynchronously invokes the Stability.ai Stable Image Core model to run
 an inference based on the provided input.
        /// </summary>
        /// <param name="prompt">The prompt that describes the image Stability.ai
 Stable Image Core has to generate.</param>
        /// <returns>A base-64 encoded image generated by model</returns>
        /// <remarks>
        /// The different model providers have individual request and response
 formats.
        /// For the format, ranges, and default values for Stability.ai Stable Image
 Core, refer to:
        ///     https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-diffusion-stable-image-core-text-image-request-response.html
        /// </remarks>
        public static async Task<string?> InvokeStableImageCoreAsync(string prompt,
 int seed)
        {
            string stableImageCoreModelId = "stability.stable-image-core-v1:1";

            AmazonBedrockRuntimeClient client = new(RegionEndpoint.USWest2);

            string payload = new JsonObject()
            {
                { "prompt", prompt },
                { "aspect_ratio", "1:1" },
                { "seed", seed },
                { "output_format", "png" }
            }.ToJsonString();

            try
            {
                InvokeModelResponse response = await client.InvokeModelAsync(new
 InvokeModelRequest()
                {
                    ModelId = stableImageCoreModelId,
                    Body = AWSSDKUtils.GenerateMemoryStreamFromString(payload),
                    ContentType = "application/json",
                    Accept = "application/json"
                });

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
                {
                    var results = JsonNode.ParseAsync(response.Body).Result?
["images"]?.AsArray();

                    return results?[0]?.GetValue<string>();
                }
                else
                {
                    Console.WriteLine("InvokeModelAsync failed with status code " +
 response.HttpStatusCode);
                }
            }
            catch (AmazonBedrockRuntimeException e)
            {
                Console.WriteLine(e.Message);
            }
            return null;
        }
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK para .NET da API.

CloudWatch Exemplos de registros usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with CloudWatch Logs.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssociateKmsKey`

O código de exemplo a seguir mostra como usar `AssociateKmsKey`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to associate an AWS Key Management Service (AWS KMS) key with
    /// an Amazon CloudWatch Logs log group.
    /// </summary>
    public class AssociateKmsKey
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string kmsKeyId = "arn:aws:kms:us-west-2:<account-
number>:key/7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
            string groupName = "cloudwatchlogs-example-loggroup";

            var request = new AssociateKmsKeyRequest
            {
                KmsKeyId = kmsKeyId,
                LogGroupName = groupName,
            };
```

```
            var response = await client.AssociateKmsKeyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully associated KMS key ID: {kmsKeyId}
with log group: {groupName}.");
            }
            else
            {
                Console.WriteLine("Could not make the association between:
{kmsKeyId} and {groupName}.");
            }
        }
    }
```

- Para obter detalhes da API, consulte AssociateKmsKey em Referência AWS SDK para .NET da API.

`CancelExportTask`

O código de exemplo a seguir mostra como usar `CancelExportTask`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to cancel an Amazon CloudWatch Logs export task.
    /// </summary>
    public class CancelExportTask
```

```
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskId = "exampleTaskId";

            var request = new CancelExportTaskRequest
            {
                TaskId = taskId,
            };

            var response = await client.CancelExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{taskId} successfully canceled.");
            }
            else
            {
                Console.WriteLine($"{taskId} could not be canceled.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CancelExportTask em Referência AWS SDK para .NET da API.

`CreateExportTask`

O código de exemplo a seguir mostra como usar `CreateExportTask`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Export Task to export the contents of the Amazon
/// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon S3)
/// bucket.
/// </summary>
public class CreateExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskName = "export-task-example";
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string destination = "amzn-s3-demo-bucket";
        var fromTime = 1437584472382;
        var toTime = 1437584472833;

        var request = new CreateExportTaskRequest
        {
            From = fromTime,
            To = toTime,
            TaskName = taskName,
            LogGroupName = logGroupName,
            Destination = destination,
        };
```

```
            var response = await client.CreateExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The task, {taskName} with ID: " +
                                  $"{response.TaskId} has been created
successfully.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateExportTask em Referência AWS SDK para .NET da API.

`CreateLogGroup`

O código de exemplo a seguir mostra como usar `CreateLogGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs log group.
/// </summary>
public class CreateLogGroup
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
```

```
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string logGroupName = "cloudwatchlogs-example-loggroup";

            var request = new CreateLogGroupRequest
            {
                LogGroupName = logGroupName,
            };

            var response = await client.CreateLogGroupAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully create log group with ID:
{logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create log group.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateLogGroup em Referência AWS SDK para .NET da API.

`CreateLogStream`

O código de exemplo a seguir mostra como usar `CreateLogStream`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
/// log group.
/// </summary>
public class CreateLogStream
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string logStreamName = "cloudwatchlogs-example-logstream";

        var request = new CreateLogStreamRequest
        {
            LogGroupName = logGroupName,
            LogStreamName = logStreamName,
        };

        var response = await client.CreateLogStreamAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"{logStreamName} successfully created for
{logGroupName}.");
        }
        else
        {
            Console.WriteLine("Could not create stream.");
        }
    }
}
```

- Para obter detalhes da API, consulte CreateLogStream em Referência AWS SDK para .NET da API.

DeleteLogGroup

O código de exemplo a seguir mostra como usar `DeleteLogGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Uses the Amazon CloudWatch Logs Service to delete an existing
    /// CloudWatch Logs log group.
    /// </summary>
    public class DeleteLogGroup
    {
        public static async Task Main()
        {
            var client = new AmazonCloudWatchLogsClient();
            string logGroupName = "cloudwatchlogs-example-loggroup";

            var request = new DeleteLogGroupRequest
            {
                LogGroupName = logGroupName,
            };

            var response = await client.DeleteLogGroupAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted CloudWatch log group,
{logGroupName}.");
```

```
            }
        }
    }
```

- Para obter detalhes da API, consulte DeleteLogGroup em Referência AWS SDK para .NET da API.

`DescribeExportTasks`

O código de exemplo a seguir mostra como usar `DescribeExportTasks`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks.
/// </summary>
public class DescribeExportTasks
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeExportTasksRequest
```

```
            {
                Limit = 5,
            };

            var response = new DescribeExportTasksResponse();

            do
            {
                response = await client.DescribeExportTasksAsync(request);
                response.ExportTasks.ForEach(t =>
                {
                    Console.WriteLine($"{t.TaskName} with ID: {t.TaskId} has status:
{t.Status}");
                });
            }
            while (response.NextToken is not null);
        }
    }
```

- Para obter detalhes da API, consulte DescribeExportTasks em Referência AWS SDK para .NET da API.

`DescribeLogGroups`

O código de exemplo a seguir mostra como usar `DescribeLogGroups`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;
```

```
    /// <summary>
    /// Retrieves information about existing Amazon CloudWatch Logs log groups
    /// and displays the information on the console.
    /// </summary>
    public class DescribeLogGroups
    {
        public static async Task Main()
        {
            // Creates a CloudWatch Logs client using the default
            // user. If you need to work with resources in another
            // AWS Region than the one defined for the default user,
            // pass the AWS Region as a parameter to the client constructor.
            var client = new AmazonCloudWatchLogsClient();

            bool done = false;
            string newToken = null;

            var request = new DescribeLogGroupsRequest
            {
                Limit = 5,
            };

            DescribeLogGroupsResponse response;

            do
            {
                if (newToken is not null)
                {
                    request.NextToken = newToken;
                }

                response = await client.DescribeLogGroupsAsync(request);

                response.LogGroups.ForEach(lg =>
                {
                    Console.WriteLine($"{lg.LogGroupName} is associated with the
key: {lg.KmsKeyId}.");
                    Console.WriteLine($"Created on: {lg.CreationTime.Date.Date}");
                    Console.WriteLine($"Date for this group will be stored for:
{lg.RetentionInDays} days.\n");
                });

                if (response.NextToken is null)
                {
```

```
                    done = true;
                }
                else
                {
                    newToken = response.NextToken;
                }
            }
            while (!done);
        }
    }
```

- Para obter detalhes da API, consulte DescribeLogGroups em Referência AWS SDK para .NET da API.

StartLiveTail

O código de exemplo a seguir mostra como usar `StartLiveTail`.

SDK para .NET

Inclua os arquivos necessários.

```
using Amazon;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

Inicie a sessão do Live Tail.

```
            var client = new AmazonCloudWatchLogsClient();
            var request = new StartLiveTailRequest
            {
                LogGroupIdentifiers = logGroupIdentifiers,
                LogStreamNames = logStreamNames,
                LogEventFilterPattern = filterPattern,
            };

            var response = await client.StartLiveTailAsync(request);

            // Catch if request fails
```

```
            if (response.HttpStatusCode != System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Failed to start live tail session");
                return;
            }
```

Você pode lidar com os eventos da sessão do Live Tail de duas maneiras:

```
            /* Method 1
            * 1). Asynchronously loop through the event stream
            * 2). Set a timer to dispose the stream and stop the Live Tail session
at the end.
            */
            var eventStream = response.ResponseStream;
            var task = Task.Run(() =>
            {
                foreach (var item in eventStream)
                {
                    if (item is LiveTailSessionUpdate liveTailSessionUpdate)
                    {
                        foreach (var sessionResult in
liveTailSessionUpdate.SessionResults)
                        {
                            Console.WriteLine("Message : {0}",
sessionResult.Message);
                        }
                    }
                    if (item is LiveTailSessionStart)
                    {
                        Console.WriteLine("Live Tail session started");
                    }
                    // On-stream exceptions are processed here
                    if (item is CloudWatchLogsEventStreamException)
                    {
                        Console.WriteLine($"ERROR: {item}");
                    }
                }
            });
            // Close the stream to stop the session after a timeout
            if (!task.Wait(TimeSpan.FromSeconds(10))){
                eventStream.Dispose();
                Console.WriteLine("End of line");
```

```
            }
```

```
            /* Method 2
            * 1). Add event handlers to each event variable
            * 2). Start processing the stream and wait for a timeout using
AutoResetEvent
            */
            AutoResetEvent endEvent = new AutoResetEvent(false);
            var eventStream = response.ResponseStream;
            using (eventStream) // automatically disposes the stream to stop the
session after execution finishes
            {
                eventStream.SessionStartReceived += (sender, e) =>
                {
                    Console.WriteLine("LiveTail session started");
                };
                eventStream.SessionUpdateReceived += (sender, e) =>
                {
                    foreach (LiveTailSessionLogEvent logEvent in
e.EventStreamEvent.SessionResults){
                        Console.WriteLine("Message: {0}", logEvent.Message);
                    }
                };
                // On-stream exceptions are captured here
                eventStream.ExceptionReceived += (sender, e) =>
                {
                    Console.WriteLine($"ERROR: {e.EventStreamException.Message}");
                };

                eventStream.StartProcessing();
                // Stream events for this amount of time.
                endEvent.WaitOne(TimeSpan.FromSeconds(10));
                Console.WriteLine("End of line");
            }
```

- Para obter detalhes da API, consulte StartLiveTail em Referência AWS SDK para .NET da API.

Exemplos do Amazon Cognito Identity Provider usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AdminGetUser`

O código de exemplo a seguir mostra como usar `AdminGetUser`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with administrator
access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }
```

- Para obter detalhes da API, consulte AdminGetUser em Referência AWS SDK para .NET da API.

`AdminInitiateAuth`

O código de exemplo a seguir mostra como usar `AdminInitiateAuth`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
```

```
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
 userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }
```

- Para obter detalhes da API, consulte AdminInitiateAuth em Referência AWS SDK para .NET da API.

`AdminRespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `AdminRespondToAuthChallenge`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
```

```
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallenge em Referência AWS SDK para .NET da API.

`AssociateSoftwareToken`

O código de exemplo a seguir mostra como usar `AssociateSoftwareToken`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }
```

- Para obter detalhes da API, consulte AssociateSoftwareToken em Referência AWS SDK para .NET da API.

`ConfirmDevice`

O código de exemplo a seguir mostra como usar `ConfirmDevice`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string deviceKey,
 string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }
```

- Para obter detalhes da API, consulte ConfirmDevice em Referência AWS SDK para .NET da API.

`ConfirmSignUp`

O código de exemplo a seguir mostra como usar `ConfirmSignUp`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code, string
userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- Para obter detalhes da API, consulte ConfirmSignUp em Referência AWS SDK para .NET da API.

`InitiateAuth`

O código de exemplo a seguir mostra como usar `InitiateAuth`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- Para obter detalhes da API, consulte InitiateAuth em Referência AWS SDK para .NET da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get a list of users for the Amazon Cognito user pool.
/// </summary>
/// <param name="userPoolId">The user pool ID.</param>
/// <returns>A list of users.</returns>
public async Task<List<UserType>> ListUsersAsync(string userPoolId)
{
    var request = new ListUsersRequest
    {
        UserPoolId = userPoolId
    };

    var users = new List<UserType>();

    var usersPaginator = _cognitoService.Paginators.ListUsers(request);
    await foreach (var response in usersPaginator.Responses)
    {
        users.AddRange(response.Users);
    }

    return users;
}
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para .NET da API.

`ResendConfirmationCode`

O código de exemplo a seguir mostra como usar `ResendConfirmationCode`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
 {response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- Para obter detalhes da API, consulte ResendConfirmationCode em Referência AWS SDK para .NET da API.

SignUp

O código de exemplo a seguir mostra como usar SignUp.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
```

```
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte SignUp em Referência AWS SDK para .NET da API.

`VerifySoftwareToken`

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- Para obter detalhes da API, consulte VerifySoftwareToken em Referência AWS SDK para .NET da API.

Cenários

Inscrever um usuário em um grupo de usuários que exija MFA

O exemplo de código a seguir mostra como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
            )
```

```
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
```

```
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address: {email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();

        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);
```

```
        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        Console.WriteLine($"Setting up authenticator for {userName} in the user
 pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator: ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google Authenticator:
 ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}


using System.Net;

namespace CognitoActions;
```

```
/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
```

```
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
```

```
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };
```

```
        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }


    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }

    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</param>
```

```
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code, string
 userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
```

```
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }


    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string deviceKey,
 string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };
```

```
        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
 {response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with administrator
 access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
 poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }


    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
```

```
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers

- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Exemplos do Amazon Comprehend usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon Comprehend.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DetectDominantLanguage`

O código de exemplo a seguir mostra como usar `DetectDominantLanguage`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example calls the Amazon Comprehend service to determine the
    /// dominant language.
    /// </summary>
    public static class DetectDominantLanguage
    {
        /// <summary>
        /// Calls Amazon Comprehend to determine the dominant language used in
        /// the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle.";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            Console.WriteLine("Calling DetectDominantLanguage\n");
            var detectDominantLanguageRequest = new DetectDominantLanguageRequest()
            {
                Text = text,
            };

            var detectDominantLanguageResponse = await
comprehendClient.DetectDominantLanguageAsync(detectDominantLanguageRequest);
            foreach (var dl in detectDominantLanguageResponse.Languages)
            {
                Console.WriteLine($"Language Code: {dl.LanguageCode}, Score:
{dl.Score}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectDominantLanguage em Referência AWS SDK para .NET da API.

`DetectEntities`

O código de exemplo a seguir mostra como usar `DetectEntities`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use the AmazonComprehend service detect any
/// entities in submitted text.
/// </summary>
public static class DetectEntities
{
    /// <summary>
    /// The main method calls the DetectEntitiesAsync method to find any
    /// entities in the sample code.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new AmazonComprehendClient();

        Console.WriteLine("Calling DetectEntities\n");
        var detectEntitiesRequest = new DetectEntitiesRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
```

```
            var detectEntitiesResponse = await
comprehendClient.DetectEntitiesAsync(detectEntitiesRequest);

            foreach (var e in detectEntitiesResponse.Entities)
            {
                Console.WriteLine($"Text: {e.Text}, Type: {e.Type}, Score:
{e.Score}, BeginOffset: {e.BeginOffset}, EndOffset: {e.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectEntities em Referência AWS SDK para .NET da API.

`DetectKeyPhrases`

O código de exemplo a seguir mostra como usar `DetectKeyPhrases`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use the Amazon Comprehend service to
    /// search text for key phrases.
    /// </summary>
    public static class DetectKeyPhrase
    {
```

```
        /// <summary>
        /// This method calls the Amazon Comprehend method DetectKeyPhrasesAsync
        /// to detect any key phrases in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectKeyPhrases");
            var detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectKeyPhrasesResponse = await
comprehendClient.DetectKeyPhrasesAsync(detectKeyPhrasesRequest);
            foreach (var kp in detectKeyPhrasesResponse.KeyPhrases)
            {
                Console.WriteLine($"Text: {kp.Text}, Score: {kp.Score}, BeginOffset:
{kp.BeginOffset}, EndOffset: {kp.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectKeyPhrases em Referência AWS SDK para .NET da API.

`DetectPiiEntities`

O código de exemplo a seguir mostra como usar `DetectPiiEntities`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use the Amazon Comprehend service to find
/// personally identifiable information (PII) within text submitted to the
/// DetectPiiEntitiesAsync method.
/// </summary>
public class DetectingPII
{
    /// <summary>
    /// This method calls the DetectPiiEntitiesAsync method to locate any
    /// personally dientifiable information within the supplied text.
    /// </summary>
    public static async Task Main()
    {
        var comprehendClient = new AmazonComprehendClient();
        var text = @"Hello Paul Santos. The latest statement for your
                    credit card account 1111-0000-1111-0000 was
                    mailed to 123 Any Street, Seattle, WA 98109.";

        var request = new DetectPiiEntitiesRequest
        {
            Text = text,
            LanguageCode = "EN",
        };

        var response = await comprehendClient.DetectPiiEntitiesAsync(request);

        if (response.Entities.Count > 0)
        {
            foreach (var entity in response.Entities)
```

```
                {
                    var entityValue = text.Substring(entity.BeginOffset,
entity.EndOffset - entity.BeginOffset);
                    Console.WriteLine($"{entity.Type}: {entityValue}");
                }
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectPiiEntities em Referência AWS SDK para .NET da API.

`DetectSentiment`

O código de exemplo a seguir mostra como usar `DetectSentiment`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to detect the overall sentiment of the supplied
/// text using the Amazon Comprehend service.
/// </summary>
public static class DetectSentiment
{
    /// <summary>
    /// This method calls the DetetectSentimentAsync method to analyze the
    /// supplied text and determine the overal sentiment.
    /// </summary>
```

```
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectSentiment");
            var detectSentimentRequest = new DetectSentimentRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectSentimentResponse = await
comprehendClient.DetectSentimentAsync(detectSentimentRequest);
            Console.WriteLine($"Sentiment: {detectSentimentResponse.Sentiment}");
            Console.WriteLine("Done");
        }
    }
```

- Para obter detalhes da API, consulte DetectSentiment em Referência AWS SDK para .NET da API.

`DetectSyntax`

O código de exemplo a seguir mostra como usar `DetectSyntax`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
```

```
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use Amazon Comprehend to detect syntax
/// elements by calling the DetectSyntaxAsync method.
/// </summary>
public class DetectingSyntax
{
    /// <summary>
    /// This method calls DetectSynaxAsync to identify the syntax elements
    /// in the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new AmazonComprehendClient();

        // Call DetectSyntax API
        Console.WriteLine("Calling DetectSyntaxAsync\n");
        var detectSyntaxRequest = new DetectSyntaxRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        DetectSyntaxResponse detectSyntaxResponse = await
comprehendClient.DetectSyntaxAsync(detectSyntaxRequest);
        foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
        {
            Console.WriteLine($"Text: {s.Text}, PartOfSpeech:
{s.PartOfSpeech.Tag}, BeginOffset: {s.BeginOffset}, EndOffset: {s.EndOffset}");
        }

        Console.WriteLine("Done");
    }
}
```

- Para obter detalhes da API, consulte DetectSyntax em Referência AWS SDK para .NET da API.

StartTopicsDetectionJob

O código de exemplo a seguir mostra como usar `StartTopicsDetectionJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example scans the documents in an Amazon Simple Storage Service
    /// (Amazon S3) bucket and analyzes it for topics. The results are stored
    /// in another bucket and then the resulting job properties are displayed
    /// on the screen. This example was created using the AWS SDK for .NEt
    /// version 3.7 and .NET Core version 5.0.
    /// </summary>
    public static class TopicModeling
    {
        /// <summary>
        /// This methos calls a topic detection job by calling the Amazon
        /// Comprehend StartTopicsDetectionJobRequest.
        /// </summary>
        public static async Task Main()
        {
            var comprehendClient = new AmazonComprehendClient();

            string inputS3Uri = "s3://input bucket/input path";
            InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
            string outputS3Uri = "s3://output bucket/output path";
            string dataAccessRoleArn = "arn:aws:iam::account ID:role/data access
role";
            int numberOfTopics = 10;

            var startTopicsDetectionJobRequest = new
StartTopicsDetectionJobRequest()
```

```
            {
                InputDataConfig = new InputDataConfig()
                {
                    S3Uri = inputS3Uri,
                    InputFormat = inputDocFormat,
                },
                OutputDataConfig = new OutputDataConfig()
                {
                    S3Uri = outputS3Uri,
                },
                DataAccessRoleArn = dataAccessRoleArn,
                NumberOfTopics = numberOfTopics,
            };

            var startTopicsDetectionJobResponse = await
comprehendClient.StartTopicsDetectionJobAsync(startTopicsDetectionJobRequest);

            var jobId = startTopicsDetectionJobResponse.JobId;
            Console.WriteLine("JobId: " + jobId);

            var describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
            {
                JobId = jobId,
            };

            var describeTopicsDetectionJobResponse = await
comprehendClient.DescribeTopicsDetectionJobAsync(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);

            var listTopicsDetectionJobsResponse = await
comprehendClient.ListTopicsDetectionJobsAsync(new
ListTopicsDetectionJobsRequest());
            foreach (var props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
            {
                PrintJobProperties(props);
            }
        }

        /// <summary>
        /// This method is a helper method that displays the job properties
        /// from the call to StartTopicsDetectionJobRequest.
```

```
        /// </summary>
        /// <param name="props">A list of properties from the call to
        /// StartTopicsDetectionJobRequest.</param>
        private static void PrintJobProperties(TopicsDetectionJobProperties props)
        {
            Console.WriteLine($"JobId: {props.JobId}, JobName: {props.JobName},
JobStatus: {props.JobStatus}");
            Console.WriteLine($"NumberOfTopics: {props.NumberOfTopics}\nInputS3Uri:
{props.InputDataConfig.S3Uri}");
            Console.WriteLine($"InputFormat: {props.InputDataConfig.InputFormat},
OutputS3Uri: {props.OutputDataConfig.S3Uri}");
        }
    }
```

- Para obter detalhes da API, consulte StartTopicsDetectionJob em Referência AWS SDK para .NET da API.

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon DocumentDB usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Processando um evento do Amazon DocumentDB com o Lambda ao usar .NET.

```
using Amazon.Lambda.Core;
using System.Text.Json;
using System;
using System.Collections.Generic;
using System.Text.Json.Serialization;
//Assembly attribute to enable the Lambda function's JSON input to be converted into
 a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace LambdaDocDb;

public class Function
{

     /// <summary>
    /// Lambda function entry point to process Amazon DocumentDB events.
    /// </summary>
    /// <param name="event">The Amazon DocumentDB event.</param>
    /// <param name="context">The Lambda context object.</param>
    /// <returns>A string to indicate successful processing.</returns>
    public string FunctionHandler(Event evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Events)
        {
            ProcessDocumentDBEvent(record, context);
        }

        return "OK";
    }

     private void ProcessDocumentDBEvent(DocumentDBEventRecord record,
 ILambdaContext context)
    {

        var eventData = record.Event;
        var operationType = eventData.OperationType;
        var databaseName = eventData.Ns.Db;
        var collectionName = eventData.Ns.Coll;
        var fullDocument = JsonSerializer.Serialize(eventData.FullDocument, new
 JsonSerializerOptions { WriteIndented = true });
```

```
        context.Logger.LogLine($"Operation type: {operationType}");
        context.Logger.LogLine($"Database: {databaseName}");
        context.Logger.LogLine($"Collection: {collectionName}");
        context.Logger.LogLine($"Full document:\n{fullDocument}");
    }



    public class Event
    {
        [JsonPropertyName("eventSourceArn")]
        public string EventSourceArn { get; set; }

        [JsonPropertyName("events")]
        public List<DocumentDBEventRecord> Events { get; set; }

        [JsonPropertyName("eventSource")]
        public string EventSource { get; set; }
    }

    public class DocumentDBEventRecord
    {
        [JsonPropertyName("event")]
        public EventData Event { get; set; }
    }

    public class EventData
    {
        [JsonPropertyName("_id")]
        public IdData Id { get; set; }

        [JsonPropertyName("clusterTime")]
        public ClusterTime ClusterTime { get; set; }

        [JsonPropertyName("documentKey")]
        public DocumentKey DocumentKey { get; set; }

        [JsonPropertyName("fullDocument")]
        public Dictionary<string, object> FullDocument { get; set; }

        [JsonPropertyName("ns")]
        public Namespace Ns { get; set; }
```

```
        [JsonPropertyName("operationType")]
        public string OperationType { get; set; }
    }

    public class IdData
    {
        [JsonPropertyName("_data")]
        public string Data { get; set; }
    }

    public class ClusterTime
    {
        [JsonPropertyName("$timestamp")]
        public Timestamp Timestamp { get; set; }
    }

    public class Timestamp
    {
        [JsonPropertyName("t")]
        public long T { get; set; }

        [JsonPropertyName("i")]
        public int I { get; set; }
    }

    public class DocumentKey
    {
        [JsonPropertyName("_id")]
        public Id Id { get; set; }
    }

    public class Id
    {
        [JsonPropertyName("$oid")]
        public string Oid { get; set; }
    }

    public class Namespace
    {
        [JsonPropertyName("db")]
        public string Db { get; set; }

        [JsonPropertyName("coll")]
        public string Coll { get; set; }
```

```
    }
}
```

Exemplos do DynamoDB usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o DynamoDB.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários
- Exemplos sem servidor
- AWS contribuições da comunidade

Ações

`BatchExecuteStatement`

O código de exemplo a seguir mostra como usar `BatchExecuteStatement`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use lotes de instruções INSERT para adicionar itens.

```
        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table into which the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
```

```
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
```

```
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
            {
                return null!;
            }
        }
```

Use lotes de instruções SELECT para obter itens.

```
        /// <summary>
        /// Gets movies from the movie table by
        /// using an Amazon DynamoDB PartiQL SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year1">The year of the first movie.</param>
        /// <param name="year2">The year of the second movie.</param>
        /// <returns>True if successful.</returns>
        public static async Task<bool> GetBatch(
            string tableName,
            string title1,
            string title2,
            int year1,
            int year2)
        {
            var getBatch = $"SELECT * FROM {tableName} WHERE title = ? AND year
= ?";
```

```
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            if (response.Responses.Count > 0)
            {
                response.Responses.ForEach(r =>
                {
                    if (r.Item.Any())
                    {
                        Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
                    }
                });
                return true;
            }
            else
            {
                Console.WriteLine($"Couldn't find either {title1} or {title2}.");
                return false;
```

```
            }

        }
```

Use lotes de instruções UPDATE para atualizar itens.

```
        /// <summary>
        /// Updates information for multiple movies.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// movies to be updated.</param>
        /// <param name="producer1">The producer name for the first movie
        /// to update.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year that the first movie was released.</param>
        /// <param name="producer2">The producer name for the second
        /// movie to update.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year that the second movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the update.</
returns>
        public static async Task<bool> UpdateBatch(
            string tableName,
            string producer1,
            string title1,
            int year1,
            string producer2,
            string title2,
            int year2)
        {

            string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title = ?
 AND year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer1 },
                        new AttributeValue { S = title1 },
```

```
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer2 },
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
 BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Use lotes de instruções DELETE para excluir itens.

```
        /// <summary>
        /// Deletes multiple movies using a PartiQL BatchExecuteAsync
        /// statement.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// moves that will be deleted.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year the first movie was released.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year the second movie was released.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeleteBatch(
            string tableName,
            string title1,
```

```
            int year1,
            string title2,
            int year2)
        {

            string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para .NET da API.

`BatchGetItem`

O código de exemplo a seguir mostra como usar `BatchGetItem`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;

namespace LowLevelBatchGet
{
    public class LowLevelBatchGet
    {
        private static readonly string _table1Name = "Forum";
        private static readonly string _table2Name = "Thread";

        public static async void RetrieveMultipleItemsBatchGet(AmazonDynamoDBClient
 client)
        {
            var request = new BatchGetItemRequest
            {
                RequestItems = new Dictionary<string, KeysAndAttributes>()
            {
                { _table1Name,
                  new KeysAndAttributes
                  {
                      Keys = new List<Dictionary<string, AttributeValue> >()
                      {
                          new Dictionary<string, AttributeValue>()
                          {
                              { "Name", new AttributeValue {
                            S = "Amazon DynamoDB"
                        } }
                          },
                          new Dictionary<string, AttributeValue>()
```

```
                        {
                            { "Name", new AttributeValue {
                          S = "Amazon S3"
                      } }
                        }
                    }
                }},
            {
                _table2Name,
                new KeysAndAttributes
                {
                    Keys = new List<Dictionary<string, AttributeValue> >()
                    {
                        new Dictionary<string, AttributeValue>()
                        {
                            { "ForumName", new AttributeValue {
                                  S = "Amazon DynamoDB"
                              } },
                            { "Subject", new AttributeValue {
                                  S = "DynamoDB Thread 1"
                              } }
                        },
                        new Dictionary<string, AttributeValue>()
                        {
                            { "ForumName", new AttributeValue {
                                  S = "Amazon DynamoDB"
                              } },
                            { "Subject", new AttributeValue {
                                  S = "DynamoDB Thread 2"
                              } }
                        },
                        new Dictionary<string, AttributeValue>()
                        {
                            { "ForumName", new AttributeValue {
                                  S = "Amazon S3"
                              } },
                            { "Subject", new AttributeValue {
                                  S = "S3 Thread 1"
                              } }
                        }
                    }
                }
            }
        }
```

```
            };

            BatchGetItemResponse response;
            do
            {
                Console.WriteLine("Making request");
                response = await client.BatchGetItemAsync(request);

                // Check the response.
                var responses = response.Responses; // Attribute list in the
response.

                foreach (var tableResponse in responses)
                {
                    var tableResults = tableResponse.Value;
                    Console.WriteLine("Items retrieved from table {0}",
tableResponse.Key);
                    foreach (var item1 in tableResults)
                    {
                        PrintItem(item1);
                    }
                }

                // Any unprocessed keys? could happen if you exceed
ProvisionedThroughput or some other error.
                Dictionary<string, KeysAndAttributes> unprocessedKeys =
response.UnprocessedKeys;
                foreach (var unprocessedTableKeys in unprocessedKeys)
                {
                    // Print table name.
                    Console.WriteLine(unprocessedTableKeys.Key);
                    // Print unprocessed primary keys.
                    foreach (var key in unprocessedTableKeys.Value.Keys)
                    {
                        PrintItem(key);
                    }
                }

                request.RequestItems = unprocessedKeys;
            } while (response.UnprocessedKeys.Count > 0);
        }

        private static void PrintItem(Dictionary<string, AttributeValue>
attributeList)
```

```
        {
            foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)
            {
                string attributeName = kvp.Key;
                AttributeValue value = kvp.Value;

                Console.WriteLine(
                    attributeName + " " +
                    (value.S == null ? "" : "S=[" + value.S + "]") +
                    (value.N == null ? "" : "N=[" + value.N + "]") +
                    (value.SS == null ? "" : "SS=[" + string.Join(",",
 value.SS.ToArray()) + "]") +
                    (value.NS == null ? "" : "NS=[" + string.Join(",",
 value.NS.ToArray()) + "]")
                    );
            }
            Console.WriteLine("************************************************");
        }

        static void Main()
        {
            var client = new AmazonDynamoDBClient();

            RetrieveMultipleItemsBatchGet(client);
        }
    }
}
```

- Para obter detalhes da API, consulte BatchGetItem em Referência AWS SDK para .NET da API.

`DescribeTable`

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
private static async Task GetTableInformation()
{
    Console.WriteLine("\n*** Retrieving table information ***");

    var response = await Client.DescribeTableAsync(new DescribeTableRequest
    {
        TableName = ExampleTableName
    });

    var table = response.Table;
    Console.WriteLine($"Name: {table.TableName}");
    Console.WriteLine($"# of items: {table.ItemCount}");

}
```

- Para obter detalhes da API, consulte DescribeTable em Referência AWS SDK para .NET da API.

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use uma instrução INSERT para adicionar um item.

```
        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Use uma instrução SELECT para obter um item.

```
        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
```

```
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }
```

Use uma instrução SELECT para obter uma lista de itens.

```
        /// <summary>
        /// Retrieve multiple movies by year using a SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="year">The year the movies were released.</param>
        /// <returns></returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { N = year.ToString() },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
```

```
                Parameters = parameters,
            });

            return response.Items;
        }
```

Use uma instrução UPDATE para atualizar um item.

```
        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Use uma instrução DELETE para excluir um único filme.

```
        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para .NET da API.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
private static async Task ListMyTables()
{
    Console.WriteLine("\n*** Listing tables ***");

    string lastTableNameEvaluated = null;
    do
    {
        var response = await Client.ListTablesAsync(new ListTablesRequest
        {
            Limit = 2,
            ExclusiveStartTableName = lastTableNameEvaluated
        });

        foreach (var name in response.TableNames)
        {
            Console.WriteLine(name);
        }

        lastTableNameEvaluated = response.LastEvaluatedTableName;
    } while (lastTableNameEvaluated != null);
}
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para .NET da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para .NET

Mostra como usar a API .NET do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Before you run this example, download 'movies.json' from
// https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
GettingStarted.Js.02.html,
// and put it in the same folder as the example.

// Separator for the console display.
var SepBar = new string('-', 80);
const string tableName = "movie_table";
const string movieFileName = @"..\..\..\..\..\..\..\..\resources\sample_files
\movies.json";

DisplayInstructions();

// Create the table and wait for it to be active.
Console.WriteLine($"Creating the movie table: {tableName}");

var success = await DynamoDBMethods.CreateMovieTableAsync(tableName);
if (success)
{
    Console.WriteLine($"Successfully created table: {tableName}.");
}
```

```
WaitForEnter();

// Add movie information to the table from moviedata.json. See the
// instructions at the top of this file to download the JSON file.
Console.WriteLine($"Inserting movies into the new table. Please wait...");
success = await PartiQLBatchMethods.InsertMovies(tableName, movieFileName);
if (success)
{
    Console.WriteLine("Movies successfully added to the table.");
}
else
{
    Console.WriteLine("Movies could not be added to the table.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var title1 = "Star Wars";
var year1 = 1977;
var title2 = "Wizard of Oz";
var year2 = 1939;

Console.WriteLine($"Updating two movies with producer information: {title1} and
 {title2}.");
success = await PartiQLBatchMethods.GetBatch(tableName, title1, title2, year1,
 year2);
if (success)
{
    Console.WriteLine($"Successfully retrieved {title1} and {title2}.");
}
else
{
    Console.WriteLine("Select statement failed.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var producer1 = "LucasFilm";
var producer2 = "MGM";

Console.WriteLine($"Updating two movies with producer information: {title1} and
 {title2}.");
```

```
success = await PartiQLBatchMethods.UpdateBatch(tableName, producer1, title1, year1,
 producer2, title2, year2);
if (success)
{
    Console.WriteLine($"Successfully updated {title1} and {title2}.");
}
else
{
    Console.WriteLine("Update failed.");
}

WaitForEnter();

// Delete multiple movies by using the BatchExecute statement.
Console.WriteLine($"Now we will delete {title1} and {title2} from the table.");
success = await PartiQLBatchMethods.DeleteBatch(tableName, title1, year1, title2,
 year2);

if (success)
{
    Console.WriteLine($"Deleted {title1} and {title2}");
}
else
{
    Console.WriteLine($"could not delete {title1} or {title2}");
}

WaitForEnter();

// DNow that the PartiQL Batch scenario is complete, delete the movie table.
success = await DynamoDBMethods.DeleteTableAsync(tableName);

if (success)
{
    Console.WriteLine($"Successfully deleted {tableName}");
}
else
{
    Console.WriteLine($"Could not delete {tableName}");
}

/// <summary>
/// Displays the description of the application on the console.
/// </summary>
```

```
void DisplayInstructions()
{
    Console.Clear();
    Console.WriteLine();
    Console.Write(new string(' ', 24));
    Console.WriteLine("DynamoDB PartiQL Basics Example");
    Console.WriteLine(SepBar);
    Console.WriteLine("This demo application shows the basics of using Amazon
 DynamoDB with the AWS SDK for");
    Console.WriteLine(".NET version 3.7 and .NET 6.");
    Console.WriteLine(SepBar);
    Console.WriteLine("Creates a table by using the CreateTable method.");
    Console.WriteLine("Gets multiple movies by using a PartiQL SELECT statement.");
    Console.WriteLine("Updates multiple movies by using the ExecuteBatch method.");
    Console.WriteLine("Deletes multiple movies by using a PartiQL DELETE
 statement.");
    Console.WriteLine("Cleans up the resources created for the demo by deleting the
 table.");
    Console.WriteLine(SepBar);

    WaitForEnter();
}

/// <summary>
/// Simple method to wait for the <Enter> key to be pressed.
/// </summary>
void WaitForEnter()
{
    Console.WriteLine("\nPress <Enter> to continue.");
    Console.Write(SepBar);
    _ = Console.ReadLine();
}


        /// <summary>
        /// Gets movies from the movie table by
        /// using an Amazon DynamoDB PartiQL SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year1">The year of the first movie.</param>
        /// <param name="year2">The year of the second movie.</param>
        /// <returns>True if successful.</returns>
```

```
        public static async Task<bool> GetBatch(
            string tableName,
            string title1,
            string title2,
            int year1,
            int year2)
        {
            var getBatch = $"SELECT * FROM {tableName} WHERE title = ? AND year
= ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            if (response.Responses.Count > 0)
            {
                response.Responses.ForEach(r =>
                {
                    if (r.Item.Any())
                    {
```

```
                        Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
                    }
                });
                return true;
            }
            else
            {
                Console.WriteLine($"Couldn't find either {title1} or {title2}.");
                return false;
            }

        }

        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table into which the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
                    {
```

```
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
```

```
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
            {
                return null!;
            }
        }

        /// <summary>
        /// Updates information for multiple movies.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// movies to be updated.</param>
        /// <param name="producer1">The producer name for the first movie
        /// to update.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year that the first movie was released.</param>
        /// <param name="producer2">The producer name for the second
        /// movie to update.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year that the second movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the update.</
returns>
        public static async Task<bool> UpdateBatch(
            string tableName,
            string producer1,
            string title1,
            int year1,
            string producer2,
```

```
            string title2,
            int year2)
        {

            string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title = ?
AND year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer1 },
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer2 },
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }

        /// <summary>
        /// Deletes multiple movies using a PartiQL BatchExecuteAsync
        /// statement.
        /// </summary>
```

```
        /// <param name="tableName">The name of the table containing the
        /// moves that will be deleted.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year the first movie was released.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year the second movie was released.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeleteBatch(
            string tableName,
            string title1,
            int year1,
            string title2,
            int year2)
        {

            string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND year
 = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
 BatchExecuteStatementRequest
            {
```

```
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para .NET da API.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace PartiQL_Basics_Scenario
{
    public class PartiQLMethods
    {
        private static readonly AmazonDynamoDBClient Client = new
 AmazonDynamoDBClient();


        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
```

```
        /// <param name="tableName">The name of the table where the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
                        {
```

```
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
```

```
            else
            {
                return null!;
            }
        }



        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }



        /// <summary>
        /// Retrieve multiple movies by year using a SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="year">The year the movies were released.</param>
        /// <returns></returns>
```

```
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { N = year.ToString() },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }


        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });
```

```
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
```

```
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Displays the list of movies returned from a database query.
        /// </summary>
        /// <param name="items">The list of movie information to display.</param>
        private static void DisplayMovies(List<Dictionary<string, AttributeValue>>
items)
        {
            if (items.Count > 0)
            {
                Console.WriteLine($"Found {items.Count} movies.");
                items.ForEach(item =>
Console.WriteLine($"{item["year"].N}\t{item["title"].S}"));
            }
            else
            {
                Console.WriteLine($"Didn't find a movie that matched the supplied
criteria.");
            }
        }
```

```
    }
}



        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
 GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
 ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }



        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
```

```
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
```

```
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para .NET da API.

Usar um modelo de documento

O exemplo de código a seguir mostra como realizar operações de criação, leitura, atualização e exclusão (CRUD) e em lote usando um modelo de documento para o DynamoDB e um SDK. AWS

Para obter mais informações, consulte o modelo de documento.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute operações CRUD usando um modelo de documento.

```
/// <summary>
/// Performs CRUD operations on an Amazon DynamoDB table.
/// </summary>
public class MidlevelItemCRUD
{
    public static async Task Main()
    {
        var tableName = "ProductCatalog";
        var sampleBookId = 555;

        var client = new AmazonDynamoDBClient();
        var productCatalog = LoadTable(client, tableName);

        await CreateBookItem(productCatalog, sampleBookId);
        RetrieveBook(productCatalog, sampleBookId);

        // Couple of sample updates.
        UpdateMultipleAttributes(productCatalog, sampleBookId);
        UpdateBookPriceConditionally(productCatalog, sampleBookId);

        // Delete.
        await DeleteBook(productCatalog, sampleBookId);
    }

    /// <summary>
```

```
        /// Loads the contents of a DynamoDB table.
        /// </summary>
        /// <param name="client">An initialized DynamoDB client object.</param>
        /// <param name="tableName">The name of the table to load.</param>
        /// <returns>A DynamoDB table object.</returns>
        public static Table LoadTable(IAmazonDynamoDB client, string tableName)
        {
            Table productCatalog = Table.LoadTable(client, tableName);
            return productCatalog;
        }

        /// <summary>
        /// Creates an example book item and adds it to the DynamoDB table
        /// ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async Task CreateBookItem(Table productCatalog, int
sampleBookId)
        {
            Console.WriteLine("\n*** Executing CreateBookItem() ***");
            var book = new Document
            {
                ["Id"] = sampleBookId,
                ["Title"] = "Book " + sampleBookId,
                ["Price"] = 19.99,
                ["ISBN"] = "111-1111111111",
                ["Authors"] = new List<string> { "Author 1", "Author 2", "Author
3" },
                ["PageCount"] = 500,
                ["Dimensions"] = "8.5x11x.5",
                ["InPublication"] = new DynamoDBBool(true),
                ["InStock"] = new DynamoDBBool(false),
                ["QuantityOnHand"] = 0,
            };

            // Adds the book to the ProductCatalog table.
            await productCatalog.PutItemAsync(book);
        }

        /// <summary>
        /// Retrieves an item, a book, from the DynamoDB ProductCatalog table.
        /// </summary>
```

```
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void RetrieveBook(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing RetrieveBook() ***");

            // Optional configuration.
            var config = new GetItemOperationConfig
            {
                AttributesToGet = new List<string> { "Id", "ISBN", "Title",
"Authors", "Price" },
                ConsistentRead = true,
            };

            Document document = await productCatalog.GetItemAsync(sampleBookId,
config);
            Console.WriteLine("RetrieveBook: Printing book retrieved...");
            PrintDocument(document);
        }

        /// <summary>
        /// Updates multiple attributes for a book and writes the changes to the
        /// DynamoDB table ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void UpdateMultipleAttributes(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\nUpdating multiple attributes....");
            int partitionKey = sampleBookId;

            var book = new Document
            {
                ["Id"] = partitionKey,

                // List of attribute updates.
                // The following replaces the existing authors list.
                ["Authors"] = new List<string> { "Author x", "Author y" },
```

```
                ["newAttribute"] = "New Value",
                ["ISBN"] = null, // Remove it.
            };

            // Optional parameters.
            var config = new UpdateItemOperationConfig
            {
                // Gets updated item in response.
                ReturnValues = ReturnValues.AllNewAttributes,
            };

            Document updatedBook = await productCatalog.UpdateItemAsync(book,
config);
            Console.WriteLine("UpdateMultipleAttributes: Printing item after
updates ...");
            PrintDocument(updatedBook);
        }

        /// <summary>
        /// Updates a book item if it meets the specified criteria.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void UpdateBookPriceConditionally(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing UpdateBookPriceConditionally() ***");

            int partitionKey = sampleBookId;

            var book = new Document
            {
                ["Id"] = partitionKey,
                ["Price"] = 29.99,
            };

            // For conditional price update, creating a condition expression.
            var expr = new Expression
            {
                ExpressionStatement = "Price = :val",
            };
            expr.ExpressionAttributeValues[":val"] = 19.00;
```

```
            // Optional parameters.
            var config = new UpdateItemOperationConfig
            {
                ConditionalExpression = expr,
                ReturnValues = ReturnValues.AllNewAttributes,
            };

            Document updatedBook = await productCatalog.UpdateItemAsync(book,
config);
            Console.WriteLine("UpdateBookPriceConditionally: Printing item whose
price was conditionally updated");
            PrintDocument(updatedBook);
        }

        /// <summary>
        /// Deletes the book with the supplied Id value from the DynamoDB table
        /// ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async Task DeleteBook(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing DeleteBook() ***");

            // Optional configuration.
            var config = new DeleteItemOperationConfig
            {
                // Returns the deleted item.
                ReturnValues = ReturnValues.AllOldAttributes,
            };
            Document document = await productCatalog.DeleteItemAsync(sampleBookId,
config);
            Console.WriteLine("DeleteBook: Printing deleted just deleted...");

            PrintDocument(document);
        }

        /// <summary>
        /// Prints the information for the supplied DynamoDB document.
        /// </summary>
```

```
        /// <param name="updatedDocument">A DynamoDB document object.</param>
        public static void PrintDocument(Document updatedDocument)
        {
            if (updatedDocument is null)
            {
                return;
            }

            foreach (var attribute in updatedDocument.GetAttributeNames())
            {
                string stringValue = null;
                var value = updatedDocument[attribute];

                if (value is null)
                {
                    continue;
                }

                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                    select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}", attribute,
stringValue);
            }
        }
    }
```

Execute operações de gravação em lote usando um modelo de documento.

```
    /// <summary>
    /// Shows how to use mid-level Amazon DynamoDB API calls to perform batch
```

```
    /// operations.
    /// </summary>
    public class MidLevelBatchWriteItem
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            await SingleTableBatchWrite(client);
            await MultiTableBatchWrite(client);
        }

        /// <summary>
        /// Perform a batch operation on a single DynamoDB table.
        /// </summary>
        /// <param name="client">An initialized DynamoDB object.</param>
        public static async Task SingleTableBatchWrite(IAmazonDynamoDB client)
        {
            Table productCatalog = Table.LoadTable(client, "ProductCatalog");
            var batchWrite = productCatalog.CreateBatchWrite();

            var book1 = new Document
            {
                ["Id"] = 902,
                ["Title"] = "My book1 in batch write using .NET helper classes",
                ["ISBN"] = "902-11-11-1111",
                ["Price"] = 10,
                ["ProductCategory"] = "Book",
                ["Authors"] = new List<string> { "Author 1", "Author 2", "Author
3" },
                ["Dimensions"] = "8.5x11x.5",
                ["InStock"] = new DynamoDBBool(true),
                ["QuantityOnHand"] = new DynamoDBNull(), // Quantity is unknown at
this time.
            };

            batchWrite.AddDocumentToPut(book1);

            // Specify delete item using overload that takes PK.
            batchWrite.AddKeyToDelete(12345);
            Console.WriteLine("Performing batch write in SingleTableBatchWrite()");
            await batchWrite.ExecuteAsync();
        }
```

```
        /// <summary>
        /// Perform a batch operation involving multiple DynamoDB tables.
        /// </summary>
        /// <param name="client">An initialized DynamoDB client object.</param>
        public static async Task MultiTableBatchWrite(IAmazonDynamoDB client)
        {
            // Specify item to add in the Forum table.
            Table forum = Table.LoadTable(client, "Forum");
            var forumBatchWrite = forum.CreateBatchWrite();

            var forum1 = new Document
            {
                ["Name"] = "Test BatchWrite Forum",
                ["Threads"] = 0,
            };
            forumBatchWrite.AddDocumentToPut(forum1);

            // Specify item to add in the Thread table.
            Table thread = Table.LoadTable(client, "Thread");
            var threadBatchWrite = thread.CreateBatchWrite();

            var thread1 = new Document
            {
                ["ForumName"] = "S3 forum",
                ["Subject"] = "My sample question",
                ["Message"] = "Message text",
                ["KeywordTags"] = new List<string> { "S3", "Bucket" },
            };
            threadBatchWrite.AddDocumentToPut(thread1);

            // Specify item to delete from the Thread table.
            threadBatchWrite.AddKeyToDelete("someForumName", "someSubject");

            // Create multi-table batch.
            var superBatch = new MultiTableDocumentBatchWrite();
            superBatch.AddBatch(forumBatchWrite);
            superBatch.AddBatch(threadBatchWrite);
            Console.WriteLine("Performing batch write in MultiTableBatchWrite()");

            // Execute the batch.
            await superBatch.ExecuteAsync();
        }
    }
```

Verifique uma tabela usando um modelo de documento.

```
    /// <summary>
    /// Shows how to use mid-level Amazon DynamoDB API calls to scan a DynamoDB
    /// table for values.
    /// </summary>
    public class MidLevelScanOnly
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            Table productCatalogTable = Table.LoadTable(client, "ProductCatalog");

            await FindProductsWithNegativePrice(productCatalogTable);
            await FindProductsWithNegativePriceWithConfig(productCatalogTable);
        }

        /// <summary>
        /// Retrieves any products that have a negative price in a DynamoDB table.
        /// </summary>
        /// <param name="productCatalogTable">A DynamoDB table object.</param>
        public static async Task FindProductsWithNegativePrice(
          Table productCatalogTable)
        {
            // Assume there is a price error. So we scan to find items priced < 0.
            var scanFilter = new ScanFilter();
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

            Search search = productCatalogTable.Scan(scanFilter);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindProductsWithNegativePrice:
printing ............");

                foreach (var document in documentList)
                {
                    PrintDocument(document);
```

```
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Finds any items in the ProductCatalog table using a DynamoDB
        /// configuration object.
        /// </summary>
        /// <param name="productCatalogTable">A DynamoDB table object.</param>
        public static async Task FindProductsWithNegativePriceWithConfig(
          Table productCatalogTable)
        {
            // Assume there is a price error. So we scan to find items priced < 0.
            var scanFilter = new ScanFilter();
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

            var config = new ScanOperationConfig()
            {
                Filter = scanFilter,
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string> { "Title", "Id" },
            };

            Search search = productCatalogTable.Scan(config);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindProductsWithNegativePriceWithConfig:
printing ............");

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Displays the details of the passed DynamoDB document object on the
        /// console.
        /// </summary>
```

```
        /// <param name="document">A DynamoDB document object.</param>
        public static void PrintDocument(Document document)
        {
            Console.WriteLine();
            foreach (var attribute in document.GetAttributeNames())
            {
                string stringValue = null;
                var value = document[attribute];
                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                    select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}");
            }
        }
    }
```

Consulte e verifique uma tabela usando um modelo de documento.

```
    /// <summary>
    /// Shows how to perform mid-level query procedures on an Amazon DynamoDB
    /// table.
    /// </summary>
    public class MidLevelQueryAndScan
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            // Query examples.
            Table replyTable = Table.LoadTable(client, "Reply");
            string forumName = "Amazon DynamoDB";
```

```
            string threadSubject = "DynamoDB Thread 2";

            await FindRepliesInLast15Days(replyTable);
            await FindRepliesInLast15DaysWithConfig(replyTable, forumName,
threadSubject);
            await FindRepliesPostedWithinTimePeriod(replyTable, forumName,
threadSubject);

            // Get Example.
            Table productCatalogTable = Table.LoadTable(client, "ProductCatalog");
            int productId = 101;

            await GetProduct(productCatalogTable, productId);
        }

        /// <summary>
        /// Retrieves information about a product from the DynamoDB table
        /// ProductCatalog based on the product ID and displays the information
        /// on the console.
        /// </summary>
        /// <param name="tableName">The name of the table from which to retrieve
        /// product information.</param>
        /// <param name="productId">The ID of the product to retrieve.</param>
        public static async Task GetProduct(Table tableName, int productId)
        {
            Console.WriteLine("*** Executing GetProduct() ***");
            Document productDocument = await tableName.GetItemAsync(productId);
            if (productDocument != null)
            {
                PrintDocument(productDocument);
            }
            else
            {
                Console.WriteLine("Error: product " + productId + " does not
exist");
            }
        }

        /// <summary>
        /// Retrieves replies from the passed DynamoDB table object.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        public static async Task FindRepliesInLast15Days(
          Table table)
```

```
        {
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
            var filter = new QueryFilter("Id", QueryOperator.Equal, "Id");
            filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);

            // Use Query overloads that take the minimum required query parameters.
            Search search = table.Query(filter);

            do
            {
                var documentSet = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesInLast15Days:
printing ............");

                foreach (var document in documentSet)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Retrieve replies made during a specific time period.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadSubject">The subject of the thread, which we are
        /// searching for replies.</param>
        public static async Task FindRepliesPostedWithinTimePeriod(
          Table table,
          string forumName,
          string threadSubject)
        {
            DateTime startDate = DateTime.UtcNow.Subtract(new TimeSpan(21, 0, 0,
0));
            DateTime endDate = DateTime.UtcNow.Subtract(new TimeSpan(1, 0, 0, 0));

            var filter = new QueryFilter("Id", QueryOperator.Equal, forumName + "#"
+ threadSubject);
            filter.AddCondition("ReplyDateTime", QueryOperator.Between, startDate,
endDate);
```

```
            var config = new QueryOperationConfig()
            {
                Limit = 2, // 2 items/page.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
        {
          "Message",
          "ReplyDateTime",
          "PostedBy",
        },
                ConsistentRead = true,
                Filter = filter,
            };

            Search search = table.Query(config);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesPostedWithinTimePeriod: printing
replies posted within dates: {0} and {1} ............", startDate, endDate);

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Perform a query for replies made in the last 15 days using a DynamoDB
        /// QueryOperationConfig object.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadName">The bane of the thread that we are searching
        /// for replies.</param>
        public static async Task FindRepliesInLast15DaysWithConfig(
          Table table,
          string forumName,
          string threadName)
```

```
        {
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
            var filter = new QueryFilter("Id", QueryOperator.Equal, forumName + "#"
+ threadName);
            filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);

            var config = new QueryOperationConfig()
            {
                Filter = filter,

                // Optional parameters.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
                {
                  "Message",
                  "ReplyDateTime",
                  "PostedBy",
                },
                ConsistentRead = true,
            };

            Search search = table.Query(config);

            do
            {
                var documentSet = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesInLast15DaysWithConfig:
printing ............");

                foreach (var document in documentSet)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Displays the contents of the passed DynamoDB document on the console.
        /// </summary>
        /// <param name="document">A DynamoDB document to display.</param>
        public static void PrintDocument(Document document)
        {
```

```
            Console.WriteLine();
            foreach (var attribute in document.GetAttributeNames())
            {
                string stringValue = null;
                var value = document[attribute];

                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                    select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}");
            }
        }
    }
```

Usar um modelo de persistência de objetos de alto nível

O exemplo de código a seguir mostra como realizar operações de criação, leitura, atualização e exclusão (CRUD) e em lote usando um modelo de persistência de objetos para o DynamoDB e um SDK. AWS

Para obter mais informações, consulte Modelo de persistência de objetos.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Realize operações CRUD usando um modelo de persistência de objetos de alto nível.

```
    /// <summary>
    /// Shows how to perform high-level CRUD operations on an Amazon DynamoDB
    /// table.
    /// </summary>
    public class HighLevelItemCrud
    {
        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);
            await PerformCRUDOperations(context);
        }

        public static async Task PerformCRUDOperations(IDynamoDBContext context)
        {
            int bookId = 1001; // Some unique value.
            Book myBook = new Book
            {
                Id = bookId,
                Title = "object persistence-AWS SDK for.NET SDK-Book 1001",
                Isbn = "111-1111111001",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
            };

            // Save the book to the ProductCatalog table.
            await context.SaveAsync(myBook);

            // Retrieve the book from the ProductCatalog table.
            Book bookRetrieved = await context.LoadAsync<Book>(bookId);

            // Update some properties.
            bookRetrieved.Isbn = "222-2222221001";

            // Update existing authors list with the following values.
            bookRetrieved.BookAuthors = new List<string> { " Author 1", "Author
x" };
            await context.SaveAsync(bookRetrieved);

            // Retrieve the updated book. This time, add the optional
            // ConsistentRead parameter using DynamoDBContextConfig object.
            await context.LoadAsync<Book>(bookId, new DynamoDBContextConfig
            {
```

```
                ConsistentRead = true,
            });

            // Delete the book.
            await context.DeleteAsync<Book>(bookId);

            // Try to retrieve deleted book. It should return null.
            Book deletedBook = await context.LoadAsync<Book>(bookId, new
DynamoDBContextConfig
            {
                ConsistentRead = true,
            });

            if (deletedBook == null)
            {
                Console.WriteLine("Book is deleted");
            }
        }
    }
```

Realize operações de gravação em lote usando um modelo de persistência de objetos de alto nível.

```
    /// <summary>
    /// Performs high-level batch write operations to an Amazon DynamoDB table.
    /// This example was written using the AWS SDK for .NET version 3.7 and .NET
    /// Core 5.0.
    /// </summary>
    public class HighLevelBatchWriteItem
    {
        public static async Task SingleTableBatchWrite(IDynamoDBContext context)
        {
            Book book1 = new Book
            {
                Id = 902,
                InPublication = true,
                Isbn = "902-11-11-1111",
                PageCount = "100",
                Price = 10,
                ProductCategory = "Book",
```

```
                Title = "My book3 in batch write",
            };

            Book book2 = new Book
            {
                Id = 903,
                InPublication = true,
                Isbn = "903-11-11-1111",
                PageCount = "200",
                Price = 10,
                ProductCategory = "Book",
                Title = "My book4 in batch write",
            };

            var bookBatch = context.CreateBatchWrite<Book>();
            bookBatch.AddPutItems(new List<Book> { book1, book2 });

            Console.WriteLine("Adding two books to ProductCatalog table.");
            await bookBatch.ExecuteAsync();
        }

        public static async Task MultiTableBatchWrite(IDynamoDBContext context)
        {
            // New Forum item.
            Forum newForum = new Forum
            {
                Name = "Test BatchWrite Forum",
                Threads = 0,
            };
            var forumBatch = context.CreateBatchWrite<Forum>();
            forumBatch.AddPutItem(newForum);

            // New Thread item.
            Thread newThread = new Thread
            {
                ForumName = "S3 forum",
                Subject = "My sample question",
                KeywordTags = new List<string> { "S3", "Bucket" },
                Message = "Message text",
            };

            DynamoDBOperationConfig config = new DynamoDBOperationConfig();
            config.SkipVersionCheck = true;
            var threadBatch = context.CreateBatchWrite<Thread>(config);
```

```
            threadBatch.AddPutItem(newThread);
            threadBatch.AddDeleteKey("some partition key value", "some sort key
value");

            var superBatch = new MultiTableBatchWrite(forumBatch, threadBatch);

            Console.WriteLine("Performing batch write in MultiTableBatchWrite().");
            await superBatch.ExecuteAsync();
        }

        public static async Task Main()
        {
            AmazonDynamoDBClient client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);

            await SingleTableBatchWrite(context);
            await MultiTableBatchWrite(context);
        }
    }
```

Mapeie dados arbitrários em uma tabela usando um modelo de persistência de objetos de alto nível.

```
    /// <summary>
    /// Shows how to map arbitrary data to an Amazon DynamoDB table.
    /// </summary>
    public class HighLevelMappingArbitraryData
    {
        /// <summary>
        /// Creates a book, adds it to the DynamoDB ProductCatalog table, retrieves
        /// the new book from the table, updates the dimensions and writes the
        /// changed item back to the table.
        /// </summary>
        /// <param name="context">The DynamoDB context object used to write and
        /// read data from the table.</param>
        public static async Task AddRetrieveUpdateBook(IDynamoDBContext context)
        {
            // Create a book.
            DimensionType myBookDimensions = new DimensionType()
            {
```

```
                Length = 8M,
                Height = 11M,
                Thickness = 0.5M,
            };

            Book myBook = new Book
            {
                Id = 501,
                Title = "AWS SDK for .NET Object Persistence Model Handling
 Arbitrary Data",
                Isbn = "999-9999999999",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
                Dimensions = myBookDimensions,
            };

            // Add the book to the DynamoDB table ProductCatalog.
            await context.SaveAsync(myBook);

            // Retrieve the book.
            Book bookRetrieved = await context.LoadAsync<Book>(501);

            // Update the book dimensions property.
            bookRetrieved.Dimensions.Height += 1;
            bookRetrieved.Dimensions.Length += 1;
            bookRetrieved.Dimensions.Thickness += 0.2M;

            // Write the changed item to the table.
            await context.SaveAsync(bookRetrieved);
        }

        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);
            await AddRetrieveUpdateBook(context);
        }
    }
```

Consulte e verifique uma tabela usando um modelo de persistência de objetos de alto nível.

```
    /// <summary>
    /// Shows how to perform high-level query and scan operations to Amazon
    /// DynamoDB tables.
    /// </summary>
    public class HighLevelQueryAndScan
    {
        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();

            DynamoDBContext context = new DynamoDBContext(client);

            // Get an item.
            await GetBook(context, 101);

            // Sample forum and thread to test queries.
            string forumName = "Amazon DynamoDB";
            string threadSubject = "DynamoDB Thread 1";

            // Sample queries.
            await FindRepliesInLast15Days(context, forumName, threadSubject);
            await FindRepliesPostedWithinTimePeriod(context, forumName,
 threadSubject);

            // Scan table.
            await FindProductsPricedLessThanZero(context);
        }

        public static async Task GetBook(IDynamoDBContext context, int productId)
        {
            Book bookItem = await context.LoadAsync<Book>(productId);

            Console.WriteLine("\nGetBook: Printing result.....");
            Console.WriteLine($"Title: {bookItem.Title} \n ISBN:{bookItem.Isbn} \n
 No. of pages: {bookItem.PageCount}");
        }

        /// <summary>
        /// Queries a DynamoDB table to find replies posted within the last 15 days.
        /// </summary>
        /// <param name="context">The DynamoDB context used to perform the query.</
param>
        /// <param name="forumName">The name of the forum that we're interested
 in.</param>
```

```
        /// <param name="threadSubject">The thread object containing the query
 parameters.</param>
        public static async Task FindRepliesInLast15Days(
          IDynamoDBContext context,
          string forumName,
          string threadSubject)
        {
            string replyId = $"{forumName} #{threadSubject}";
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);

            List<object> times = new List<object>();
            times.Add(twoWeeksAgoDate);

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc = new ScanCondition("PostedBy", ScanOperator.GreaterThan,
 times.ToArray());
            scs.Add(sc);

            var cfg = new DynamoDBOperationConfig
            {
                QueryFilter = scs,
            };

            AsyncSearch<Reply> response = context.QueryAsync<Reply>(replyId, cfg);
            IEnumerable<Reply> latestReplies = await response.GetRemainingAsync();

            Console.WriteLine("\nReplies in last 15 days:");

            foreach (Reply r in latestReplies)
            {

 Console.WriteLine($"{r.Id}\t{r.PostedBy}\t{r.Message}\t{r.ReplyDateTime}");
            }
        }

        /// <summary>
        /// Queries for replies posted within a specific time period.
        /// </summary>
        /// <param name="context">The DynamoDB context used to perform the query.</
param>
        /// <param name="forumName">The name of the forum that we're interested
 in.</param>
        /// <param name="threadSubject">Information about the subject that we're
        /// interested in.</param>
```

```
        public static async Task FindRepliesPostedWithinTimePeriod(
          IDynamoDBContext context,
          string forumName,
          string threadSubject)
        {
            string forumId = forumName + "#" + threadSubject;
            Console.WriteLine("\nReplies posted within time period:");

            DateTime startDate = DateTime.UtcNow - TimeSpan.FromDays(30);
            DateTime endDate = DateTime.UtcNow - TimeSpan.FromDays(1);

            List<object> times = new List<object>();
            times.Add(startDate);
            times.Add(endDate);

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc = new ScanCondition("LastPostedBy", ScanOperator.Between,
times.ToArray());
            scs.Add(sc);

            var cfg = new DynamoDBOperationConfig
            {
                QueryFilter = scs,
            };

            AsyncSearch<Reply> response = context.QueryAsync<Reply>(forumId, cfg);
            IEnumerable<Reply> repliesInAPeriod = await
response.GetRemainingAsync();

            foreach (Reply r in repliesInAPeriod)
            {

Console.WriteLine("{r.Id}\t{r.PostedBy}\t{r.Message}\t{r.ReplyDateTime}");
            }
        }

        /// <summary>
        /// Queries the DynamoDB ProductCatalog table for products costing less
        /// than zero.
        /// </summary>
        /// <param name="context">The DynamoDB context object used to perform the
        /// query.</param>
        public static async Task FindProductsPricedLessThanZero(IDynamoDBContext
context)
```

```
        {
            int price = 0;

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc1 = new ScanCondition("Price", ScanOperator.LessThan, price);
            var sc2 = new ScanCondition("ProductCategory", ScanOperator.Equal,
"Book");
            scs.Add(sc1);
            scs.Add(sc2);

            AsyncSearch<Book> response = context.ScanAsync<Book>(scs);

            IEnumerable<Book> itemsWithWrongPrice = await
response.GetRemainingAsync();

            Console.WriteLine("\nFindProductsPricedLessThanZero: Printing
result.....");

            foreach (Book r in itemsWithWrongPrice)
            {
                Console.WriteLine($"{r.Id}\t{r.Title}\t{r.Price}\t{r.Isbn}");
            }
        }
    }
```

Exemplos sem servidor

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace AWSLambda_DDB;

public class Function
{
    public void FunctionHandler(DynamoDBEvent dynamoEvent, ILambdaContext context)
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");

        foreach (var record in dynamoEvent.Records)
        {
            context.Logger.LogInformation($"Event ID: {record.EventID}");
            context.Logger.LogInformation($"Event Name: {record.EventName}");

            context.Logger.LogInformation(JsonSerializer.Serialize(record));
        }

        context.Logger.LogInformation("Stream processing complete.");
    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace AWSLambda_DDB;

public class Function
{
    public StreamsEventResponse FunctionHandler(DynamoDBEvent dynamoEvent,
 ILambdaContext context)

    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>();
        StreamsEventResponse streamsEventResponse = new StreamsEventResponse();

        foreach (var record in dynamoEvent.Records)
        {
            try
            {
                var sequenceNumber = record.Dynamodb.SequenceNumber;
                context.Logger.LogInformation(sequenceNumber);
            }
```

```
            catch (Exception ex)
            {
                context.Logger.LogError(ex.Message);
                batchItemFailures.Add(new StreamsEventResponse.BatchItemFailure()
 { ItemIdentifier = record.Dynamodb.SequenceNumber });
            }
        }

        if (batchItemFailures.Count > 0)
        {
            streamsEventResponse.BatchItemFailures = batchItemFailures;
        }

        context.Logger.LogInformation("Stream processing complete.");
        return streamsEventResponse;
    }
}
```

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para .NET

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK .NET.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos do Amazon EC2 usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon EC2.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário em um prompt de comando.

```
/// <summary>
/// Show Amazon Elastic Compute Cloud (Amazon EC2) Basics actions.
/// </summary>
public class EC2Basics
{
    public static ILogger<EC2Basics> _logger = null!;
    public static EC2Wrapper _ec2Wrapper = null!;
    public static SsmWrapper _ssmWrapper = null!;
    public static UiMethods _uiMethods = null!;

    public static string associationId = null!;
    public static string allocationId = null!;
    public static string instanceId = null!;
    public static string keyPairName = null!;
    public static string groupName = null!;
    public static string tempFileName = null!;
    public static string secGroupId = null!;
    public static bool isInteractive = true;

    /// <summary>
    /// Perform the actions defined for the Amazon EC2 Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 and Amazon Simple Systems
        // Management (Amazon SSM) Service.
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
```

```
                    .AddTransient<EC2Wrapper>()
                    .AddTransient<SsmWrapper>()
            )
            .Build();

        SetUpServices(host);

        var uniqueName = Guid.NewGuid().ToString();
        keyPairName = "mvp-example-key-pair" + uniqueName;
        groupName = "ec2-scenario-group" + uniqueName;
        var groupDescription = "A security group created for the EC2 Basics
scenario.";

        try
        {
            // Start the scenario.
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(isInteractive);

            // Create the key pair.
            _uiMethods.DisplayTitle("Create RSA key pair");
            Console.Write("Let's create an RSA key pair that you can be use to ");
            Console.WriteLine("securely connect to your EC2 instance.");
            var keyPair = await _ec2Wrapper.CreateKeyPair(keyPairName);

            // Save key pair information to a temporary file.
            tempFileName = _ec2Wrapper.SaveKeyPair(keyPair);

            Console.WriteLine(
                $"Created the key pair: {keyPair.KeyName} and saved it to:
{tempFileName}");
            string? answer = "";
            if (isInteractive)
            {
                do
                {
                    Console.Write("Would you like to list your existing key pairs?
");
                    answer = Console.ReadLine();
                } while (answer!.ToLower() != "y" && answer.ToLower() != "n");
            }

            if (!isInteractive || answer == "y")
            {
```

```
                // List existing key pairs.
                _uiMethods.DisplayTitle("Existing key pairs");

                // Passing an empty string to the DescribeKeyPairs method will
return
                // a list of all existing key pairs.
                var keyPairs = await _ec2Wrapper.DescribeKeyPairs("");
                keyPairs.ForEach(kp =>
                {
                    Console.WriteLine(
                        $"{kp.KeyName} created at: {kp.CreateTime} Fingerprint:
{kp.KeyFingerprint}");
                });
            }

            _uiMethods.PressEnter(isInteractive);

            // Create the security group.
            Console.WriteLine(
                "Let's create a security group to manage access to your instance.");
            secGroupId = await _ec2Wrapper.CreateSecurityGroup(groupName,
groupDescription);
            Console.WriteLine(
                "Let's add rules to allow all HTTP and HTTPS inbound traffic and to
allow SSH only from your current IP address.");

            _uiMethods.DisplayTitle("Security group information");
            var secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);

            Console.WriteLine($"Created security group {groupName} in your default
VPC.");
            secGroups.ForEach(group =>
            {
                _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
            });
            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we'll authorize the security group we just created so that it
can");
            Console.WriteLine("access the EC2 instances you create.");
            await _ec2Wrapper.AuthorizeSecurityGroupIngress(groupName);

            secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);
```

```
            Console.WriteLine($"Now let's look at the permissions again.");
            secGroups.ForEach(group =>
            {
                _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
            });
            _uiMethods.PressEnter(isInteractive);

            // Get list of available Amazon Linux 2 Amazon Machine Images (AMIs).
            var parameters =
                await _ssmWrapper.GetParametersByPath(
                    "/aws/service/ami-amazon-linux-latest");

            List<string> imageIds = parameters.Select(param =>
param.Value).ToList();

            var images = await _ec2Wrapper.DescribeImages(imageIds);

            var i = 1;
            images.ForEach(image =>
            {
                Console.WriteLine($"\t{i++}\t{image.Description}");
            });

            int choice = 1;
            bool validNumber = false;
            if (isInteractive)
            {
                do
                {
                    Console.Write("Please select an image: ");
                    var selImage = Console.ReadLine();
                    validNumber = int.TryParse(selImage, out choice);
                } while (!validNumber);
            }

            var selectedImage = images[choice - 1];

            // Display available instance types.
            _uiMethods.DisplayTitle("Instance Types");
            var instanceTypes =
                await _ec2Wrapper.DescribeInstanceTypes(selectedImage.Architecture);

            i = 1;
            instanceTypes.ForEach(instanceType =>
```

```
            {
                Console.WriteLine($"\t{i++}\t{instanceType.InstanceType}");
            });
            if (isInteractive)
            {
                do
                {
                    Console.Write("Please select an instance type: ");
                    var selImage = Console.ReadLine();
                    validNumber = int.TryParse(selImage, out choice);
                } while (!validNumber);
            }

            var selectedInstanceType = instanceTypes[choice - 1].InstanceType;

            // Create an EC2 instance.
            _uiMethods.DisplayTitle("Creating an EC2 Instance");
            instanceId = await _ec2Wrapper.RunInstances(selectedImage.ImageId,
                selectedInstanceType, keyPairName, secGroupId);

            _uiMethods.PressEnter(isInteractive);

            var instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine(
                "\nYou can use SSH to connect to your instance. For example:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we'll stop the instance and then start it again to see what's
changed.");

            await _ec2Wrapper.StopInstances(instanceId);

            Console.WriteLine("Now let's start it up again.");
            await _ec2Wrapper.StartInstances(instanceId);

            Console.WriteLine("\nLet's see what changed.");
```

```
            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine("\nNotice the change in the SSH information:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we will stop the instance again. Then we will create and
associate an");
            Console.WriteLine("Elastic IP address to use with our instance.");

            await _ec2Wrapper.StopInstances(instanceId);
            _uiMethods.PressEnter(isInteractive);

            _uiMethods.DisplayTitle("Allocate Elastic IP address");
            Console.WriteLine(
                "You can allocate an Elastic IP address and associate it with your
instance\nto keep a consistent IP address even when your instance restarts.");
            var allocationResponse = await _ec2Wrapper.AllocateAddress();
            allocationId = allocationResponse.AllocationId;
            Console.WriteLine(
                "Now we will associate the Elastic IP address with our instance.");
            associationId = await _ec2Wrapper.AssociateAddress(allocationId,
instanceId);

            // Start the instance again.
            Console.WriteLine("Now let's start the instance again.");
            await _ec2Wrapper.StartInstances(instanceId);

            Console.WriteLine("\nLet's see what changed.");

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("Instance information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine("\nHere is the SSH information:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            Console.WriteLine("Let's stop and start the instance again.");
```

```
            _uiMethods.PressEnter(isInteractive);

            await _ec2Wrapper.StopInstances(instanceId);

            Console.WriteLine("\nThe instance has stopped.");

            Console.WriteLine("Now let's start it up again.");
            await _ec2Wrapper.StartInstances(instanceId);

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);
            Console.WriteLine("Note that the IP address did not change this time.");
            _uiMethods.PressEnter(isInteractive);

            await Cleanup();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, starting
cleanup.");
            await Cleanup();
        }

        _uiMethods.DisplayTitle("EC2 Basics Scenario completed.");
        _uiMethods.PressEnter(isInteractive);
    }

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host"></param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<EC2Basics>(loggerFactory);

        // Now the client is available for injection.
        _ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        _ssmWrapper = host.Services.GetRequiredService<SsmWrapper>();
        _uiMethods = new UiMethods();
```

```
    }

    /// <summary>
    /// Clean up any resources from the scenario.
    /// </summary>
    /// <returns></returns>
    public static async Task Cleanup()
    {
        _uiMethods.DisplayTitle("Clean up resources");
        Console.WriteLine("Now let's clean up the resources we created.");

        Console.WriteLine("Disassociate the Elastic IP address and release it.");
        // Disassociate the Elastic IP address.
        await _ec2Wrapper.DisassociateIp(associationId);

        // Delete the Elastic IP address.
        await _ec2Wrapper.ReleaseAddress(allocationId);

        // Terminate the instance.
        Console.WriteLine("Terminating the instance we created.");
        await _ec2Wrapper.TerminateInstances(instanceId);

        // Delete the security group.
        Console.WriteLine($"Deleting the Security Group: {groupName}.");
        await _ec2Wrapper.DeleteSecurityGroup(secGroupId);

        // Delete the RSA key pair.
        Console.WriteLine($"Deleting the key pair: {keyPairName}");
        await _ec2Wrapper.DeleteKeyPair(keyPairName);
        Console.WriteLine("Deleting the temporary file with the key information.");
        _ec2Wrapper.DeleteTempFile(tempFileName);
        _uiMethods.PressEnter(isInteractive);
    }
}
```

Defina uma classe que envolva as ações do EC2.

```
/// <summary>
/// Methods of this class perform Amazon Elastic Compute Cloud (Amazon EC2).
/// </summary>
public class EC2Wrapper
{
```

```
    private readonly IAmazonEC2 _amazonEC2;
    private readonly ILogger<EC2Wrapper> _logger;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EC2 client.</param>
    /// <param name="logger">The injected logger.</param>
    public EC2Wrapper(IAmazonEC2 amazonService, ILogger<EC2Wrapper> logger)
    {
        _amazonEC2 = amazonService;
        _logger = logger;
    }

    /// <summary>
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
    // instance. By using an Elastic IP address, you can keep the public IP address
    // constant even when you restart the associated instance.
    /// </summary>
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation ID
 {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-
addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
 exceeded. {ec2Exception.Message}");
            }

            throw;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Associates an Elastic IP address with an instance. When this association is
    /// created, the Elastic IP's public IP address is immediately used as the
 public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
 instanceId)
    {
        try
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to associate address.
 {ec2Exception.Message}");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer ingress to EC2 instances associated
    /// with the virtual private cloud (VPC) security group.
    /// </summary>
    /// <param name="groupName">The name of the security group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
    {
        try
        {
            // Get the IP address for the local computer.
            var ipAddress = await GetIpAddress();
            Console.WriteLine($"Your IP address is: {ipAddress}");
            var ipRanges =
                new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
            var permission = new IpPermission
            {
                Ipv4Ranges = ipRanges,
                IpProtocol = "tcp",
                FromPort = 22,
                ToPort = 22
            };
            var permissions = new List<IpPermission> { permission };
            var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
                new AuthorizeSecurityGroupIngressRequest(groupName, permissions));
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
            {
                _logger.LogError(
                    $"The ingress rule already exists. {ec2Exception.Message}");
            }
```

```
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while authorizing ingress.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer for ingress to
    /// the Amazon EC2 SecurityGroup.
    /// </summary>
    /// <returns>The IPv4 address of the computer running the scenario.</returns>
    private static async Task<string> GetIpAddress()
    {
        var httpClient = new HttpClient();
        var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

        // The IP address is returned with a new line
        // character on the end. Trim off the whitespace and
        // return the value to the caller.
        return ipString.Trim();
    }

    /// <summary>
    /// Create an Amazon EC2 key pair with a specified name.
    /// </summary>
    /// <param name="keyPairName">The name for the new key pair.</param>
    /// <returns>The Amazon EC2 key pair created.</returns>
    public async Task<KeyPair?> CreateKeyPair(string keyPairName)
    {
        try
        {
            var request = new CreateKeyPairRequest { KeyName = keyPairName, };

            var response = await _amazonEC2.CreateKeyPairAsync(request);

            var kp = response.KeyPair;
            // Return the key pair so it can be saved if needed.
```

```
            // Wait until the key pair exists.
            int retries = 5;
            while (retries-- > 0)
            {
                Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
                var keyPairs = await DescribeKeyPairs(keyPairName);
                if (keyPairs.Any())
                {
                    return kp;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
            throw new DoesNotExistException("KeyPair not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} already exists.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the key pair.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Save KeyPair information to a temporary file.
    /// </summary>
    /// <param name="keyPair">The name of the key pair.</param>
    /// <returns>The full path to the temporary file.</returns>
    public string SaveKeyPair(KeyPair keyPair)
    {
        var tempPath = Path.GetTempPath();
```

```
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");

        // Save the key pair to a file in a temporary folder.
        using var stream = new FileStream(pemFileName, FileMode.Create);
        using var writer = new StreamWriter(stream);
        writer.WriteLine(keyPair.KeyMaterial);

        return pemFileName;
    }

    /// <summary>
    /// Create an Amazon EC2 security group with a specified name and description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
    public async Task<string> CreateSecurityGroup(string groupName, string
 groupDescription)
    {
        try
        {
            var response = await _amazonEC2.CreateSecurityGroupAsync(
                new CreateSecurityGroupRequest(groupName, groupDescription));

            // Wait until the security group exists.
            int retries = 5;
            while (retries-- > 0)
            {
                var groups = await DescribeSecurityGroups(response.GroupId);
                if (groups.Any())
                {
                    return response.GroupId;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
            throw new DoesNotExistException("security group not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
```

```
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
            }
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the security group.:
{ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Create a new Amazon EC2 VPC.
    /// </summary>
    /// <param name="cidrBlock">The CIDR block for the new security group.</param>
    /// <returns>The VPC Id of the new VPC.</returns>
    public async Task<string?> CreateVPC(string cidrBlock)
    {

        try
        {
            var response = await _amazonEC2.CreateVpcAsync(new CreateVpcRequest
            {
                CidrBlock = cidrBlock,
            });

            Vpc vpc = response.Vpc;
            Console.WriteLine($"Created VPC with ID: {vpc.VpcId}.");
            return vpc.VpcId;
        }
        catch (AmazonEC2Exception ex)
        {
            Console.WriteLine($"Couldn't create VPC because: {ex.Message}");
            return null;
        }
    }
```

```
    /// <summary>
    /// Delete an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyPair(string keyPairName)
    {
        try
        {
            await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError($"KeyPair {keyPairName} does not exist and cannot
be deleted. Please verify the key pair name and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete the temporary file where the key pair information was saved.
    /// </summary>
    /// <param name="tempFileName">The path to the temporary file.</param>
    public void DeleteTempFile(string tempFileName)
    {
        if (File.Exists(tempFileName))
        {
            File.Delete(tempFileName);
        }
    }

    /// <summary>
```

```
    /// Delete an Amazon EC2 security group.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteSecurityGroup(string groupId)
    {
        try
        {
            var response =
                await _amazonEC2.DeleteSecurityGroupAsync(
                    new DeleteSecurityGroupRequest { GroupId = groupId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"Security Group {groupId} does not exist and cannot be deleted.
Please verify the ID and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 VPC.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteVpc(string vpcId)
    {
        var request = new DeleteVpcRequest
        {
            VpcId = vpcId,
        };

        var response = await _amazonEC2.DeleteVpcAsync(request);
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Get information about existing Amazon EC2 images.
    /// </summary>
    /// <returns>A list of image information.</returns>
    public async Task<List<Image>> DescribeImages(List<string>? imageIds)
    {
        var request = new DescribeImagesRequest();
        if (imageIds is not null)
        {
            // If the imageIds list is not null, add the list
            // to the request object.
            request.ImageIds = imageIds;
        }

        var response = await _amazonEC2.DescribeImagesAsync(request);
        return response.Images;
    }

    /// <summary>
    /// Display the information returned by DescribeImages.
    /// </summary>
    /// <param name="images">The list of image information to display.</param>
    public void DisplayImageInfo(List<Image> images)
    {
        images.ForEach(image =>
        {
            Console.WriteLine($"{image.Name} Created on: {image.CreationDate}");
        });

    }

    /// <summary>
    /// Get information about an Amazon EC2 instance.
    /// </summary>
    /// <param name="instanceId">The instance Id of the EC2 instance.</param>
    /// <returns>An EC2 instance.</returns>
    public async Task<Instance> DescribeInstance(string instanceId)
    {
        var response = await _amazonEC2.DescribeInstancesAsync(
```

```
            new DescribeInstancesRequest { InstanceIds = new List<string>
{ instanceId } });
        return response.Reservations[0].Instances[0];
    }

    /// <summary>
    /// Display EC2 instance information.
    /// </summary>
    /// <param name="instance">The instance Id of the EC2 instance.</param>
    public void DisplayInstanceInformation(Instance instance)
    {
        Console.WriteLine($"ID: {instance.InstanceId}");
        Console.WriteLine($"Image ID: {instance.ImageId}");
        Console.WriteLine($"{instance.InstanceType}");
        Console.WriteLine($"Key Name: {instance.KeyName}");
        Console.WriteLine($"VPC ID: {instance.VpcId}");
        Console.WriteLine($"Public IP: {instance.PublicIpAddress}");
        Console.WriteLine($"State: {instance.State.Name}");
    }

    /// <summary>
    /// Get information about EC2 instances with a particular state.
    /// </summary>
    /// <param name="tagName">The name of the tag to filter on.</param>
    /// <param name="tagValue">The value of the tag to look for.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> GetInstancesWithState(string state)
    {
        try
        {
            // Filters the results of the instance list.
            var filters = new List<Filter>
            {
                new Filter
                {
                    Name = $"instance-state-name",
                    Values = new List<string> { state, },
                },
            };
            var request = new DescribeInstancesRequest { Filters = filters, };

            Console.WriteLine($"\nShowing instances with state {state}");
            var paginator = _amazonEC2.Paginators.DescribeInstances(request);
```

```
            await foreach (var response in paginator.Responses)
            {
                foreach (var reservation in response.Reservations)
                {
                    foreach (var instance in reservation.Instances)
                    {
                        Console.Write($"Instance ID: {instance.InstanceId} ");
                        Console.WriteLine($"\tCurrent State:
{instance.State.Name}");
                    }
                }
            }

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Invalid parameter value for filtering instances.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list instances because: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Describe the instance types available.
    /// </summary>
    /// <returns>A list of instance type information.</returns>
    public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
    {
        try
        {
            var request = new DescribeInstanceTypesRequest();

            var filters = new List<Filter>
```

```
            {
                new Filter("processor-info.supported-architecture",
                    new List<string> { architecture.ToString() })
            };
            filters.Add(new Filter("instance-type", new() { "*.micro",
"*.small" }));

            request.Filters = filters;
            var instanceTypes = new List<InstanceTypeInfo>();

            var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
            await foreach (var instanceType in paginator.InstanceTypes)
            {
                instanceTypes.Add(instanceType);
            }

            return instanceTypes;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Parameters are invalid. Ensure architecture and size strings
conform to DescribeInstanceTypes API reference.");
            }

            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get information about an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair.</param>
    /// <returns>A list of key pair information.</returns>
    public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
    {
```

```
        try
        {
            var request = new DescribeKeyPairsRequest();
            if (!string.IsNullOrEmpty(keyPairName))
            {
                request = new DescribeKeyPairsRequest
                {
                    KeyNames = new List<string> { keyPairName }
                };
            }

            var response = await _amazonEC2.DescribeKeyPairsAsync(request);
            return response.KeyPairs.ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while describing the key pair.: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Retrieve information for one or all Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The optional Id of a specific Amazon EC2 security
 group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        try
        {
            var securityGroups = new List<SecurityGroup>();
```

```
            var request = new DescribeSecurityGroupsRequest();

            if (!string.IsNullOrEmpty(groupId))
            {
                var groupIds = new List<string> { groupId };
                request.GroupIds = groupIds;
            }

            var paginatorForSecurityGroups =
                _amazonEC2.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            return securityGroups;

        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"A security group {groupId} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while listing security groups. {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Display the information returned by the call to
    /// DescribeSecurityGroupsAsync.
    /// </summary>
    /// <param name="securityGroup">A list of security group information.</param>
    public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
```

```
    {
        Console.WriteLine($"{securityGroup.GroupName}");
        Console.WriteLine("Ingress permissions:");
        securityGroup.IpPermissions.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range => { Console.Write($"{range.CidrIp}
"); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
        Console.WriteLine("Egress permissions:");
        securityGroup.IpPermissionsEgress.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range => { Console.Write($"{range.CidrIp}
"); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
    }


    /// <summary>
```

```
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId = associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)
    {
        try
        {
            var request = new RebootInstancesRequest
            {
```

```
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.RebootInstancesAsync(request);

            // Wait for the instance to be running.
            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
 {ec2Exception.Message}");
            }
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while rebooting the instance {ec2InstanceId}.:
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Release an Elastic IP address. After the Elastic IP address is released,
    /// it can no longer be used.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
            var request = new ReleaseAddressRequest { AllocationId = allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
```

```
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while releasing the AllocationId
{allocationId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
    /// <param name="groupId">The Id of the Amazon EC2 security group that will be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
    {
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
```

```
                MinCount = 1,
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
            var instanceId = response.Reservation.Instances[0].InstanceId;

            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Start an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
    /// to start.</param>
    /// <returns>Async task.</returns>
    public async Task StartInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StartInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
```

```
            };

            await _amazonEC2.StartInstancesAsync(request);

            Console.Write("Waiting for instance to start. ");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while starting the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Stop an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance to
    /// stop.</param>
    /// <returns>Async task.</returns>
    public async Task StopInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StopInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StopInstancesAsync(request);
            Console.Write("Waiting for the instance to stop.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);
```

```
            Console.WriteLine("\nThe instance has stopped.");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while stopping the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Terminate an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance
    /// to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
    {
        try
        {
            var request = new TerminateInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId }
            };

            var response = await _amazonEC2.TerminateInstancesAsync(request);
            Console.Write("Waiting for the instance to terminate.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Terminated);

            Console.WriteLine($"\nThe instance {ec2InstanceId} has been
terminated.");
```

```
            return response.TerminatingInstances;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while terminating the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
```

```
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
    // instance. By using an Elastic IP address, you can keep the public IP address
    // constant even when you restart the associated instance.
    /// </summary>
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation ID
 {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-
addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
 exceeded. {ec2Exception.Message}");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
 {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte AllocateAddress em Referência AWS SDK para .NET da API.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Associates an Elastic IP address with an instance. When this association is
    /// created, the Elastic IP's public IP address is immediately used as the
 public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
 instanceId)
    {
        try
```

```
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to associate address.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte AssociateAddress em Referência AWS SDK para .NET da API.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Authorize the local computer ingress to EC2 instances associated
/// with the virtual private cloud (VPC) security group.
/// </summary>
/// <param name="groupName">The name of the security group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
{
    try
    {
        // Get the IP address for the local computer.
        var ipAddress = await GetIpAddress();
        Console.WriteLine($"Your IP address is: {ipAddress}");
        var ipRanges =
            new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
        var permission = new IpPermission
        {
            Ipv4Ranges = ipRanges,
            IpProtocol = "tcp",
            FromPort = 22,
            ToPort = 22
        };
        var permissions = new List<IpPermission> { permission };
        var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest(groupName, permissions));
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
        {
            _logger.LogError(
                $"The ingress rule already exists. {ec2Exception.Message}");
        }
```

```
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while authorizing ingress.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer for ingress to
    /// the Amazon EC2 SecurityGroup.
    /// </summary>
    /// <returns>The IPv4 address of the computer running the scenario.</returns>
    private static async Task<string> GetIpAddress()
    {
        var httpClient = new HttpClient();
        var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

        // The IP address is returned with a new line
        // character on the end. Trim off the whitespace and
        // return the value to the caller.
        return ipString.Trim();
    }
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngress em Referência AWS SDK para .NET da API.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an Amazon EC2 key pair with a specified name.
    /// </summary>
    /// <param name="keyPairName">The name for the new key pair.</param>
    /// <returns>The Amazon EC2 key pair created.</returns>
    public async Task<KeyPair?> CreateKeyPair(string keyPairName)
    {
        try
        {
            var request = new CreateKeyPairRequest { KeyName = keyPairName, };

            var response = await _amazonEC2.CreateKeyPairAsync(request);

            var kp = response.KeyPair;
            // Return the key pair so it can be saved if needed.

            // Wait until the key pair exists.
            int retries = 5;
            while (retries-- > 0)
            {
                Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
                var keyPairs = await DescribeKeyPairs(keyPairName);
                if (keyPairs.Any())
                {
                    return kp;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
            throw new DoesNotExistException("KeyPair not found");
        }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} already exists.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the key pair.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Save KeyPair information to a temporary file.
    /// </summary>
    /// <param name="keyPair">The name of the key pair.</param>
    /// <returns>The full path to the temporary file.</returns>
    public string SaveKeyPair(KeyPair keyPair)
    {
        var tempPath = Path.GetTempPath();
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");

        // Save the key pair to a file in a temporary folder.
        using var stream = new FileStream(pemFileName, FileMode.Create);
        using var writer = new StreamWriter(stream);
        writer.WriteLine(keyPair.KeyMaterial);

        return pemFileName;
    }
```

- Para obter detalhes da API, consulte CreateKeyPair em Referência AWS SDK para .NET da API.

`CreateLaunchTemplate`

O código de exemplo a seguir mostra como usar `CreateLaunchTemplate`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
   /// The launch template specifies a Bash script in its user data field that runs
after
   /// the instance is started. This script installs the Python packages and starts
a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
```

```
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                            {
                                Name = _instanceProfileName
                            },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateLaunchTemplate em Referência AWS SDK para .NET da API.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an Amazon EC2 security group with a specified name and description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
    public async Task<string> CreateSecurityGroup(string groupName, string
 groupDescription)
    {
        try
        {
            var response = await _amazonEC2.CreateSecurityGroupAsync(
                new CreateSecurityGroupRequest(groupName, groupDescription));

            // Wait until the security group exists.
            int retries = 5;
            while (retries-- > 0)
            {
                var groups = await DescribeSecurityGroups(response.GroupId);
                if (groups.Any())
                {
                    return response.GroupId;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
            throw new DoesNotExistException("security group not found");
        }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
            }
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the security group.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateSecurityGroup em Referência AWS SDK para .NET da API.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> DeleteKeyPair(string keyPairName)
    {
        try
        {
            await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError($"KeyPair {keyPairName} does not exist and cannot
be deleted. Please verify the key pair name and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete the temporary file where the key pair information was saved.
    /// </summary>
    /// <param name="tempFileName">The path to the temporary file.</param>
    public void DeleteTempFile(string tempFileName)
    {
        if (File.Exists(tempFileName))
        {
            File.Delete(tempFileName);
        }
    }
```

- Para obter detalhes da API, consulte DeleteKeyPair em Referência AWS SDK para .NET da API.

`DeleteLaunchTemplate`

O código de exemplo a seguir mostra como usar `DeleteLaunchTemplate`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
```

```
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DeleteLaunchTemplate em Referência AWS SDK para .NET da API.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an Amazon EC2 security group.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteSecurityGroup(string groupId)
    {
        try
        {
            var response =
                await _amazonEC2.DeleteSecurityGroupAsync(
                    new DeleteSecurityGroupRequest { GroupId = groupId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"Security Group {groupId} does not exist and cannot be deleted.
Please verify the ID and try again.");
```

```
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteSecurityGroup em Referência AWS SDK para .NET da API.

`DescribeAvailabilityZones`

O código de exemplo a seguir mostra como usar `DescribeAvailabilityZones`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability zones.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeAvailabilityZones em Referência AWS SDK para .NET da API.

`DescribeIamInstanceProfileAssociations`

O código de exemplo a seguir mostra como usar `DescribeIamInstanceProfileAssociations`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
```

```
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeIamInstanceProfileAssociations em Referência AWS SDK para .NET da API.

`DescribeInstanceTypes`

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the instance types available.
    /// </summary>
    /// <returns>A list of instance type information.</returns>
    public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
    {
        try
        {
            var request = new DescribeInstanceTypesRequest();

            var filters = new List<Filter>
            {
                new Filter("processor-info.supported-architecture",
                    new List<string> { architecture.ToString() })
            };
            filters.Add(new Filter("instance-type", new() { "*.micro",
"*.small" }));

            request.Filters = filters;
            var instanceTypes = new List<InstanceTypeInfo>();

            var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
            await foreach (var instanceType in paginator.InstanceTypes)
            {
                instanceTypes.Add(instanceType);
            }

            return instanceTypes;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Parameters are invalid. Ensure architecture and size strings
conform to DescribeInstanceTypes API reference.");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeInstanceTypes em Referência AWS SDK para .NET da API.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about EC2 instances with a particular state.
    /// </summary>
    /// <param name="tagName">The name of the tag to filter on.</param>
    /// <param name="tagValue">The value of the tag to look for.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> GetInstancesWithState(string state)
    {
        try
        {
            // Filters the results of the instance list.
            var filters = new List<Filter>
            {
                new Filter
                {
                    Name = $"instance-state-name",
                    Values = new List<string> { state, },
                },
            };
```

```
            var request = new DescribeInstancesRequest { Filters = filters, };

            Console.WriteLine($"\nShowing instances with state {state}");
            var paginator = _amazonEC2.Paginators.DescribeInstances(request);

            await foreach (var response in paginator.Responses)
            {
                foreach (var reservation in response.Reservations)
                {
                    foreach (var instance in reservation.Instances)
                    {
                        Console.Write($"Instance ID: {instance.InstanceId} ");
                        Console.WriteLine($"\tCurrent State:
{instance.State.Name}");
                    }
                }
            }

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Invalid parameter value for filtering instances.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list instances because: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DescribeInstances em Referência AWS SDK para .NET da API.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get information about an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair.</param>
/// <returns>A list of key pair information.</returns>
public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
{
    try
    {
        var request = new DescribeKeyPairsRequest();
        if (!string.IsNullOrEmpty(keyPairName))
        {
            request = new DescribeKeyPairsRequest
            {
                KeyNames = new List<string> { keyPairName }
            };
        }

        var response = await _amazonEC2.DescribeKeyPairsAsync(request);
        return response.KeyPairs.ToList();
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
        {
            _logger.LogError(
                $"A key pair called {keyPairName} does not exist.");
        }

        throw;
    }
```

```
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while describing the key pair.: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeKeyPairs em Referência AWS SDK para .NET da API.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve information for one or all Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        try
        {
            var securityGroups = new List<SecurityGroup>();
            var request = new DescribeSecurityGroupsRequest();

            if (!string.IsNullOrEmpty(groupId))
            {
                var groupIds = new List<string> { groupId };
```

```
                request.GroupIds = groupIds;
            }

            var paginatorForSecurityGroups =
                _amazonEC2.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            return securityGroups;

        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"A security group {groupId} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while listing security groups. {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Display the information returned by the call to
    /// DescribeSecurityGroupsAsync.
    /// </summary>
    /// <param name="securityGroup">A list of security group information.</param>
    public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
    {
        Console.WriteLine($"{securityGroup.GroupName}");
        Console.WriteLine("Ingress permissions:");
        securityGroup.IpPermissions.ForEach(permission =>
        {
```

```
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range => { Console.Write($"{range.CidrIp}
"); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
        Console.WriteLine("Egress permissions:");
        securityGroup.IpPermissionsEgress.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range => { Console.Write($"{range.CidrIp}
"); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
    }
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK para .NET da API.

`DescribeSubnets`

O código de exemplo a seguir mostra como usar `DescribeSubnets`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeSubnets em Referência AWS SDK para .NET da API.

`DescribeVpcs`

O código de exemplo a seguir mostra como usar `DescribeVpcs`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
```

```
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeVpcs em Referência AWS SDK para .NET da API.

`DisassociateAddress`

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId = associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DisassociateAddress em Referência AWS SDK para .NET da API.

`RebootInstances`

O código de exemplo a seguir mostra como usar `RebootInstances`.

SDK para .NET

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Reinicialize uma instância pelo ID.

```
    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)
    {
        try
        {
            var request = new RebootInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.RebootInstancesAsync(request);

            // Wait for the instance to be running.
            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
```

```
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
{ec2Exception.Message}");
            }
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while rebooting the instance {ec2InstanceId}.:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);
```

```
        return hasState;
    }
```

Substitua o perfil de uma instância, reinicialize e reinicie um servidor Web.

```
    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
```

```
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the template.:
 {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte RebootInstances em Referência AWS SDK para .NET da API.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Release an Elastic IP address. After the Elastic IP address is released,
    /// it can no longer be used.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
            var request = new ReleaseAddressRequest { AllocationId = allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while releasing the AllocationId
{allocationId}.: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte ReleaseAddress em Referência AWS SDK para .NET da API.

`ReplaceIamInstanceProfileAssociation`

O código de exemplo a seguir mostra como usar `ReplaceIamInstanceProfileAssociation`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
```

```
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
```

```
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ReplaceIamInstanceProfileAssociation em Referência AWS SDK para .NET da API.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create and run an EC2 instance.
   /// </summary>
   /// <param name="ImageId">The image Id of the image used as a basis for the
   /// EC2 instance.</param>
   /// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
   /// <param name="keyName">The name of the key pair to associate with the
   /// instance.</param>
   /// <param name="groupId">The Id of the Amazon EC2 security group that will be
   /// allowed to interact with the new EC2 instance.</param>
   /// <returns>The instance Id of the new EC2 instance.</returns>
   public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
   {
       try
       {
           var request = new RunInstancesRequest
           {
               ImageId = imageId,
               InstanceType = instanceType,
               KeyName = keyName,
               MinCount = 1,
               MaxCount = 1,
               SecurityGroupIds = new List<string> { groupId }
           };
           var response = await _amazonEC2.RunInstancesAsync(request);
           var instanceId = response.Reservation.Instances[0].InstanceId;

           Console.Write("Waiting for the instance to start.");
           await WaitForInstanceState(instanceId, InstanceStateName.Running);
```

```
            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte RunInstances em Referência AWS SDK para .NET da API.

StartInstances

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start an EC2 instance.
    /// </summary>
```

```
    /// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
    /// to start.</param>
    /// <returns>Async task.</returns>
    public async Task StartInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StartInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StartInstancesAsync(request);

            Console.Write("Waiting for instance to start. ");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while starting the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
```

```
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Para obter detalhes da API, consulte StartInstances em Referência AWS SDK para .NET da API.

`StopInstances`

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Stop an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance to
    /// stop.</param>
    /// <returns>Async task.</returns>
    public async Task StopInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StopInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StopInstancesAsync(request);
            Console.Write("Waiting for the instance to stop.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);

            Console.WriteLine("\nThe instance has stopped.");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while stopping the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Para obter detalhes da API, consulte StopInstances em Referência AWS SDK para .NET da API.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Terminate an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance
    /// to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
    {
        try
        {
            var request = new TerminateInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId }
            };

            var response = await _amazonEC2.TerminateInstancesAsync(request);
            Console.Write("Waiting for the instance to terminate.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Terminated);

            Console.WriteLine($"\nThe instance {ec2InstanceId} has been
terminated.");
            return response.TerminatingInstances;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while terminating the instance.: {ex.Message}");
            throw;
        }
    }
```

```
    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Para obter detalhes da API, consulte TerminateInstances em Referência AWS SDK para .NET da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
static async Task Main(string[] args)
{
    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
        .Build();
```

```
        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);
```

```
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper = host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
```

```
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several AWS
resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways to
make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));
```

```
        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs 'server_startup_script.sh'
when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the `server.py`
script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2 instances,
each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3, _autoScalerWrapper.GroupName,
zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or continue
with the demo.\n");

        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
 The target group\n"
            + "defines how the load balancer connects to instances. The load
 balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroupN
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.LoadB
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper.Lo
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();
```

```
            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port, ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for your
 default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management Console.
\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port, ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
 ipString);
                }
            }
            loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
```

```
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite of
these failures.");
        Console.WriteLine(new string('-', 88));
```

```
        Console.WriteLine("At the start, the load balancer endpoint returns
 recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
 recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
 parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
 let's set this parameter to name a non-existent table.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as\n" +
                          "healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service fails,
 the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
 "static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
 checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
 _smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in the
 target group so that it can't\n" +
```

```
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile that
contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns either
a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo, a
deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not for
Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the Auto
Scaling group when a dependent service fails.");
```

```
        Console.WriteLine("\nBy implementing deep health checks, the load balancer
can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two for
the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler start
a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a successful
recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the load
balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance typically
takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");
```

```
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
 continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
 return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account, we
 can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
 resources? (y/n) "))
        {
            await
 _elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBalanc
            await
 _elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGroup
            await
 _autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupName)
            await
 _autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
```

```
            await
 _autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
 _recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
 incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
```

```
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
```

```
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";
```

```
        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
```

```
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
```

```
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
```

```
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
    /// the instance is started. This script installs the Python packages and starts
a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
    /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
    /// <returns>The template object.</returns>
    public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
    {
        try
        {
            await CreateKeyPair(_keyPairName);
            await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
                _instanceProfileName, instancePolicyPath);

            var startServerText = await File.ReadAllTextAsync(startupScriptPath);
            var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

            var amiLatest = await _amazonSsm.GetParameterAsync(
                new GetParameterRequest() { Name = _amiParam });
            var amiId = amiLatest.Parameter.Value;
            var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new
```

```
LaunchTemplateIamInstanceProfileSpecificationRequest()
                            {
                                Name = _instanceProfileName
                            },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
 availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability zones.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with size
 {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
```

```
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
```

```
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
```

```
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }
```

```
    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
```

```
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
```

```
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
```

```
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
```

```
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
```

```
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
```

```
   /// Verify the default security group of a Vpc allows ingress from the calling
computer.
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP address
while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be open
to either this computer's IP\n" +
                                       "address, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
               }
               else
               {
```

```
                    break;
                }
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    /// The
    /// </summary>
```

```
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
 targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
```

```
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
```

```
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
```

```
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
```

```
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
```

```
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
```

```
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
```

```
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
```

```
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books, movies,
 and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
```

```
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");
```

```
            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await File.ReadAllTextAsync(recommendationsPath);
        var records =
            JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
```

```
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-architecture-
table";
```

```
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
```

```
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup

- UpdateAutoScalingGroup

Exemplos do Amazon ECS usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon ECS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`ListClusters`

O código de exemplo a seguir mostra como usar `ListClusters`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List cluster ARNs available.
```

```
    /// </summary>
    /// <returns>The ARN list of clusters.</returns>
    public async Task<List<string>> GetClusterARNSAsync()
    {

        Console.WriteLine("Getting a list of all the clusters in your AWS
account...");
        List<string> clusterArnList = new List<string>();
        // Get a list of all the clusters in your AWS account
        try
        {

            var listClustersResponse = _ecsClient.Paginators.ListClusters(new
ListClustersRequest
            {
            });

            var clusterArns = listClustersResponse.ClusterArns;

            // Print the ARNs of the clusters
            await foreach (var clusterArn in clusterArns)
            {
                clusterArnList.Add(clusterArn);
            }

            if (clusterArnList.Count == 0)
            {
                _logger.LogWarning("No clusters found in your AWS account.");
            }
            return clusterArnList;
        }
        catch (Exception e)
        {
            _logger.LogError($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
            throw new Exception($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
        }
    }
```

- Para obter detalhes da API, consulte ListClusters em Referência AWS SDK para .NET da API.

`ListServices`

O código de exemplo a seguir mostra como usar `ListServices`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// List service ARNs available.
/// </summary>
/// <param name="clusterARN">The arn of the ECS cluster.</param>
/// <returns>The ARN list of services in given cluster.</returns>
public async Task<List<string>> GetServiceARNSAsync(string clusterARN)
{
    List<string> serviceArns = new List<string>();

    var request = new ListServicesRequest
    {
        Cluster = clusterARN
    };
    // Call the ListServices API operation and get the list of service ARNs
    var serviceList = _ecsClient.Paginators.ListServices(request);

    await foreach (var serviceARN in serviceList.ServiceArns)
    {
        if (serviceARN is null)
            continue;

        serviceArns.Add(serviceARN);
    }

    if (serviceArns.Count == 0)
    {
        _logger.LogWarning($"No services found in cluster {clusterARN} .");
    }

    return serviceArns;
}
```

- Para obter detalhes da API, consulte ListServices em Referência AWS SDK para .NET da API.

`ListTasks`

O código de exemplo a seguir mostra como usar `ListTasks`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List task ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of tasks in given cluster.</returns>
    public async Task<List<string>> GetTaskARNsAsync(string clusterARN)
    {
        // Set up the request to describe the tasks in the service
        var listTasksRequest = new ListTasksRequest
        {
            Cluster = clusterARN
        };
        List<string> taskArns = new List<string>();

        // Call the ListTasks API operation and get the list of task ARNs
        var tasks = _ecsClient.Paginators.ListTasks(listTasksRequest);

        await foreach (var task in tasks.TaskArns)
        {
            if (task is null)
                continue;


            taskArns.Add(task);
```

```
        }

        if (taskArns.Count == 0)
        {
            _logger.LogWarning("No tasks found in cluster: " + clusterARN);
        }

        return taskArns;
    }
```

- Para obter detalhes da API, consulte ListTasks em Referência AWS SDK para .NET da API.

Cenários

Obter informações de ARN para clusters, serviços e tarefas

O exemplo de código a seguir mostra como:

- Obter uma lista de todos os clusters.
- Obter serviços para um cluster.
- Obter tarefas para um cluster.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
using Amazon.ECS;
using ECSActions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
```

```
namespace ECSScenario;

public class ECSScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.


    This .NET example performs the following tasks:
        1. List ECS Cluster ARNs.
        2. List services in every cluster
        3. List Task ARNs in every cluster.
    */

    private static ILogger logger = null!;
    private static ECSWrapper _ecsWrapper = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
        .Build();

        ILoggerFactory loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<ECSScenario>();

        var loggerECSWarpper = LoggerFactory.Create(builder =>
 { builder.AddConsole(); })
          .CreateLogger<ECSWrapper>();

        var amazonECSClient = new AmazonECSClient();

        _ecsWrapper = new ECSWrapper(amazonECSClient, loggerECSWarpper);
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon ECS example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            await ListClusterARNs();
            await ListServiceARNs();
            await ListTaskARNs();

        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// List ECS Cluster ARNs
    /// </summary>
    private static async Task ListClusterARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. List Cluster ARNs from ECS.");
        var arns = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var arn in arns)
        {
            Console.WriteLine($"Cluster arn: {arn}");
            Console.WriteLine($"Cluster name: {arn.Split("/").Last()}");
        }

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// List services in every cluster
    /// </summary>
    private static async Task ListServiceARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List Service ARNs in every cluster.");
```

```
        var clusterARNs = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var clusterARN in clusterARNs)
        {
            Console.WriteLine($"Getting services for cluster name:
{clusterARN.Split("/").Last()}");
            Console.WriteLine(new string('.', 5));


            var serviceARNs = await _ecsWrapper.GetServiceARNSAsync(clusterARN);

            foreach (var serviceARN in serviceARNs)
            {
                Console.WriteLine($"Service arn: {serviceARN}");
                Console.WriteLine($"Service name: {serviceARN.Split("/").Last()}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// List tasks in every cluster
    /// </summary>
    private static async Task ListTaskARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. List Task ARNs in every cluster.");
        var clusterARNs = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var clusterARN in clusterARNs)
        {
            Console.WriteLine($"Getting tasks for cluster name:
{clusterARN.Split("/").Last()}");
            Console.WriteLine(new string('.', 5));

            var taskARNs = await _ecsWrapper.GetTaskARNsAsync(clusterARN);

            foreach (var taskARN in taskARNs)
            {
                Console.WriteLine($"Task arn: {taskARN}");
            }
        }
```

```
        Console.WriteLine(new string('-', 80));
    }
}
```

Métodos de encapsulamento que são chamados pelo cenário para gerenciar as ações do Amazon ECS.

```
using Amazon.ECS;
using Amazon.ECS.Model;
using Microsoft.Extensions.Logging;

namespace ECSActions;

public class ECSWrapper
{
    private readonly AmazonECSClient _ecsClient;
    private readonly ILogger<ECSWrapper> _logger;

    /// <summary>
    /// Constructor for the ECS wrapper.
    /// </summary>
    /// <param name="ecsClient">The injected ECS client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public ECSWrapper(AmazonECSClient ecsClient, ILogger<ECSWrapper> logger)

    {
        _logger = logger;
        _ecsClient = ecsClient;
    }

    /// <summary>
    /// List cluster ARNs available.
    /// </summary>
    /// <returns>The ARN list of clusters.</returns>
    public async Task<List<string>> GetClusterARNSAsync()
    {

        Console.WriteLine("Getting a list of all the clusters in your AWS
 account...");
        List<string> clusterArnList = new List<string>();
        // Get a list of all the clusters in your AWS account
        try
```

```
        {

            var listClustersResponse = _ecsClient.Paginators.ListClusters(new
ListClustersRequest
            {
            });

            var clusterArns = listClustersResponse.ClusterArns;

            // Print the ARNs of the clusters
            await foreach (var clusterArn in clusterArns)
            {
                clusterArnList.Add(clusterArn);
            }

            if (clusterArnList.Count == 0)
            {
                _logger.LogWarning("No clusters found in your AWS account.");
            }
            return clusterArnList;
        }
        catch (Exception e)
        {
            _logger.LogError($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
            throw new Exception($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
        }
    }

    /// <summary>
    /// List service ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of services in given cluster.</returns>
    public async Task<List<string>> GetServiceARNSAsync(string clusterARN)
    {
        List<string> serviceArns = new List<string>();

        var request = new ListServicesRequest
        {
            Cluster = clusterARN
        };
        // Call the ListServices API operation and get the list of service ARNs
```

```
        var serviceList = _ecsClient.Paginators.ListServices(request);

        await foreach (var serviceARN in serviceList.ServiceArns)
        {
            if (serviceARN is null)
                continue;

            serviceArns.Add(serviceARN);
        }

        if (serviceArns.Count == 0)
        {
            _logger.LogWarning($"No services found in cluster {clusterARN} .");
        }

        return serviceArns;
    }

    /// <summary>
    /// List task ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of tasks in given cluster.</returns>
    public async Task<List<string>> GetTaskARNsAsync(string clusterARN)
    {
        // Set up the request to describe the tasks in the service
        var listTasksRequest = new ListTasksRequest
        {
            Cluster = clusterARN
        };
        List<string> taskArns = new List<string>();

        // Call the ListTasks API operation and get the list of task ARNs
        var tasks = _ecsClient.Paginators.ListTasks(listTasksRequest);

        await foreach (var task in tasks.TaskArns)
        {
            if (task is null)
                continue;


            taskArns.Add(task);
        }
```

```
        if (taskArns.Count == 0)
        {
            _logger.LogWarning("No tasks found in cluster: " + clusterARN);
        }

        return taskArns;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - ListClusters
  - ListServices
  - ListTasks

Elastic Load Balancing - exemplos da versão 2 usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with Elastic Load Balancing - Versão 2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateListener`

O código de exemplo a seguir mostra como usar `CreateListener`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
   /// and forwards requests to the specified target group.
   /// </summary>
   /// <param name="name">The name for the new load balancer.</param>
   /// <param name="subnetIds">Subnets for the load balancer.</param>
   /// <param name="targetGroup">Target group for forwarded requests.</param>
   /// <returns>The new LoadBalancer object.</returns>
   public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
   {
       var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
           new CreateLoadBalancerRequest()
           {
               Name = name,
               Subnets = subnetIds
           });
       var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

       // Wait for load balancer to be available.
       var loadBalancerReady = false;
       while (!loadBalancerReady)
       {
           try
           {
               var describeResponse =
                   await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
```

```
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }
```

- Para obter detalhes da API, consulte CreateListener em Referência AWS SDK para .NET da API.

`CreateLoadBalancer`

O código de exemplo a seguir mostra como usar `CreateLoadBalancer`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }
```

- Para obter detalhes da API, consulte CreateLoadBalancer em Referência AWS SDK para .NET da API.

`CreateTargetGroup`

O código de exemplo a seguir mostra como usar `CreateTargetGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
```

```
        return targetGroup;
    }
```

- Para obter detalhes da API, consulte CreateTargetGroup em Referência AWS SDK para .NET da API.

`DeleteLoadBalancer`

O código de exemplo a seguir mostra como usar `DeleteLoadBalancer`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
```

```
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }
```

- Para obter detalhes da API, consulte DeleteLoadBalancer em Referência AWS SDK para .NET da API.

`DeleteTargetGroup`

O código de exemplo a seguir mostra como usar `DeleteTargetGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
```

```
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

- Para obter detalhes da API, consulte DeleteTargetGroup em Referência AWS SDK para .NET da API.

`DescribeLoadBalancers`

O código de exemplo a seguir mostra como usar `DescribeLoadBalancers`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
```

```
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }
```

- Para obter detalhes da API, consulte DescribeLoadBalancers em Referência AWS SDK para .NET da API.

`DescribeTargetHealth`

O código de exemplo a seguir mostra como usar `DescribeTargetHealth`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
```

```
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }
```

- Para obter detalhes da API, consulte DescribeTargetHealth em Referência AWS SDK para .NET da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper = host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
```

```
            "\nFor this demo, we'll use the AWS SDK for .NET to create several AWS
resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways to
make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs 'server_startup_script.sh'
when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the `server.py`
script. The web server\n"
```

```
            + "listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2 instances,
each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3, _autoScalerWrapper.GroupName,
zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or continue
with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the demo.");
        await _smParameterWrapper.Reset();
```

```
        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
 The target group\n"
            + "defines how the load balancer connects to instances. The load
 balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroupN
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.LoadB
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper.Lo
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port, ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);
```

```
            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for your
 default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management Console.
\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port, ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
 ipString);
                }
            }
            loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
 browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
```

```
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite of
these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
```

```
                            $"The table name is contained in a Systems Manager
 parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                            $"To simulate a failure of the recommendation service,
 let's set this parameter to name a non-existent table.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as\n" +
                            "healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service fails,
 the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
 "static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
 checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
 _smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in the
 target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
```

```
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile that
contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns either
a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo, a
deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not for
Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the Auto
Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load balancer
can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");
```

```
        Console.WriteLine($"\nNow, checking target health indicates that the
 instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two for
 the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
 endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
 rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by an
 auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler start
 a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
 instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a successful
 recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the load
 balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance typically
 takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
 instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
 checks mean all instances report as unhealthy.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
 continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
 return a static response rather than fail");
```

```
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account, we
can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBalanc
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGroup
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupName)
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
```

```
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
 incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
```

```
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
```

```
   /// instance.The role has attached policies that specify the AWS permissions
granted to
   /// clients that run on the instance.
   /// </summary>
   /// <param name="policyName">Name to use for the policy.</param>
   /// <param name="roleName">Name to use for the role.</param>
   /// <param name="profileName">Name to use for the profile.</param>
   /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
   /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
   /// <returns>The Arn of the profile.</returns>
   public async Task<string> CreateInstanceProfileWithName(
       string policyName,
       string roleName,
       string profileName,
       string ssmOnlyPolicyFile,
       List<string>? awsManagedPolicies = null)
   {

       var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

       var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

       var policyArn = "";

       try
       {
           var createPolicyResult = await _amazonIam.CreatePolicyAsync(
               new CreatePolicyRequest
               {
                   PolicyName = policyName,
                   PolicyDocument = policyDocument
               });
```

```
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
```

```
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
```

```
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
```

```
   /// the instance is started. This script installs the Python packages and starts
a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
```

```
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                  $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"An error occurred while listing availability zones.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with size
 {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
```

```
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
```

```
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
```

```
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
```

```
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
```

```
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
```

```
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the calling
computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP address
while running this example.
    /// If you do, be sure to remove public access when you're done.
```

```
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be open
to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }
```

```
    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
 targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
```

```
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
```

```
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
```

```
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
```

```
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
```

```
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
```

```
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
```

```
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

```
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books, movies,
 and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
```

```
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
```

```
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await File.ReadAllTextAsync(recommendationsPath);
        var records =
            JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
```

```
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-architecture-
table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
```

```
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

EventBridge exemplos usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with EventBridge.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá EventBridge

O exemplo de código a seguir mostra como começar a usar o EventBridge.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.EventBridge;
using Amazon.EventBridge.Model;

namespace EventBridgeActions;
```

```
public static class HelloEventBridge
{
    static async Task Main(string[] args)
    {
        var eventBridgeClient = new AmazonEventBridgeClient();

        Console.WriteLine($"Hello Amazon EventBridge! Following are some of your
 EventBuses:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five event buses.
        var response = await eventBridgeClient.ListEventBusesAsync(
            new ListEventBusesRequest()
            {
                Limit = 5
            });

        foreach (var eventBus in response.EventBuses)
        {
            Console.WriteLine($"\tEventBus: {eventBus.Name}");
            Console.WriteLine($"\tArn: {eventBus.Arn}");
            Console.WriteLine($"\tPolicy: {eventBus.Policy}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte ListEventBuses em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma regra e adicionar um destino a ela.
- Habilitar e desabilitar regras.
- Listar e atualizar regras e destinos.

- Envie eventos e, em seguida, limpe os recursos.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
public class EventBridgeScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks with Amazon EventBridge:
    - Create a rule.
    - Add a target to a rule.
    - Enable and disable rules.
    - List rules and targets.
    - Update rules and targets.
    - Send events.
    - Delete the rule.
    */

    private static ILogger logger = null!;
    private static EventBridgeWrapper _eventBridgeWrapper = null!;
    private static IConfiguration _configuration = null!;

    private static IAmazonIdentityManagementService? _iamClient = null!;
    private static IAmazonSimpleNotificationService? _snsClient = null!;
    private static IAmazonS3 _s3Client = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonEventBridge>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddAWSService<IAmazonS3>()
            .AddAWSService<IAmazonSimpleNotificationService>()
            .AddTransient<EventBridgeWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<EventBridgeScenario>();

        ServicesSetup(host);

        string topicArn = "";
        string roleArn = "";

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            roleArn = await CreateRole();

            await CreateBucketWithEventBridgeEvents();

            await AddEventRule(roleArn);

            await ListEventRules();

            topicArn = await CreateSnsTopic();

            var email = await SubscribeToSnsTopic(topicArn);
```

```
            await AddSnsTarget(topicArn);

            await ListTargets();

            await ListRulesForTarget(topicArn);

            await UploadS3File(_s3Client);

            await ChangeRuleState(false);

            await GetRuleState();

            await UpdateSnsEventRule(topicArn);

            await ChangeRuleState(true);

            await UploadS3File(_s3Client);

            await UpdateToCustomRule(topicArn);

            await TriggerCustomRule(email);

            await CleanupResources(topicArn);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources(topicArn);
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon EventBridge example scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _eventBridgeWrapper =
host.Services.GetRequiredService<EventBridgeWrapper>();
```

```
        _snsClient =
host.Services.GetRequiredService<IAmazonSimpleNotificationService>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    }

    /// <summary>
    /// Create a role to be used by EventBridge.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateRole()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating a role to use with EventBridge and attaching
managed policy AmazonEventBridgeFullAccess.");
        Console.WriteLine(new string('-', 80));

        var roleName = _configuration["roleName"];

        var assumeRolePolicy = "{" +
                                   "\"Version\": \"2012-10-17\"," +
                                   "\"Statement\": [{" +
                                   "\"Effect\": \"Allow\"," +
                                   "\"Principal\": {" +
                                   $"\"Service\": \"events.amazonaws.com\"" +
                                   "}," +
                                   "\"Action\": \"sts:AssumeRole\"" +
                                   "}]" +
                                   "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = roleName
            });

        await _iamClient.AttachRolePolicyAsync(
            new AttachRolePolicyRequest()
            {
                PolicyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess",
                RoleName = roleName
```

```
            });
        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Create an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
events enabled.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateBucketWithEventBridgeEvents()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an S3 bucket with EventBridge events enabled.");

        var testBucketName = _configuration["testBucketName"];

        var bucketExists = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            testBucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = testBucketName,
                UseClientRegion = true
            });
        }

        await _s3Client.PutBucketNotificationAsync(new
PutBucketNotificationRequest()
        {
            BucketName = testBucketName,
            EventBridgeConfiguration = new EventBridgeConfiguration()
        });

        Console.WriteLine($"\tAdded bucket {testBucketName} with EventBridge events
enabled.");

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// Create and upload a file to an S3 bucket to trigger an event.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UploadS3File(IAmazonS3 s3Client)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Uploading a file to the test bucket. This will trigger a
subscription email.");

        var testBucketName = _configuration["testBucketName"];

        var fileName = $"example_upload_{DateTime.UtcNow.Ticks}.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for testing uploads.");
        }

        await s3Client.PutObjectAsync(new PutObjectRequest()
        {
            FilePath = fileName,
            BucketName = testBucketName
        });

        Console.WriteLine($"\tPress Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an Amazon Simple Notification Service (Amazon SNS) topic to use as an
EventBridge target.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> CreateSnsTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
```

```
            "Creating an Amazon Simple Notification Service (Amazon SNS) topic for
email subscriptions.");

        var topicName = _configuration["topicName"];

        string topicPolicy = "{" +
                             "\"Version\": \"2012-10-17\"," +
                             "\"Statement\": [{" +
                             "\"Sid\": \"EventBridgePublishTopic\"," +
                             "\"Effect\": \"Allow\"," +
                             "\"Principal\": {" +
                             $"\"Service\": \"events.amazonaws.com\"" +
                             "}," +
                             "\"Resource\": \"*\"," +
                             "\"Action\": \"sns:Publish\"" +
                             "}]" +
                             "}";

        var topicAttributes = new Dictionary<string, string>()
        {
            { "Policy", topicPolicy }
        };

        var topicResponse = await _snsClient!.CreateTopicAsync(new
CreateTopicRequest()
        {
            Name = topicName,
            Attributes = topicAttributes

        });

        Console.WriteLine($"\tAdded topic {topicName} for email subscriptions.");

        Console.WriteLine(new string('-', 80));

        return topicResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a user email to an SNS topic.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>The user's email.</returns>
    private static async Task<string> SubscribeToSnsTopic(string topicArn)
```

```
    {
        Console.WriteLine(new string('-', 80));


        string email = "";
        while (string.IsNullOrEmpty(email))
        {
            Console.WriteLine("Enter your email to subscribe to the Amazon SNS
topic:");
            email = Console.ReadLine()!;
        }

        var subscriptions = new List<string>();
        var paginatedSubscriptions =
_snsClient!.Paginators.ListSubscriptionsByTopic(
            new ListSubscriptionsByTopicRequest()
            {
                TopicArn = topicArn
            });

        // Get the entire list using the paginator.
        await foreach (var subscription in paginatedSubscriptions.Subscriptions)
        {
            subscriptions.Add(subscription.Endpoint);
        }

        if (subscriptions.Contains(email))
        {
            Console.WriteLine($"\tYour email is already subscribed.");
            Console.WriteLine(new string('-', 80));
            return email;
        }

        await _snsClient.SubscribeAsync(new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "email",
            Endpoint = email
        });

        Console.WriteLine($"Use the link in the email you received to confirm your
subscription, then press Enter to continue.");

        Console.ReadLine();
```

```
        Console.WriteLine(new string('-', 80));
        return email;
    }

    /// <summary>
    /// Add a rule which triggers when a file is uploaded to an S3 bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role used by EventBridge.</param>
    /// <returns>Async task.</returns>
    private static async Task AddEventRule(string roleArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an EventBridge event that sends an email when an
Amazon S3 object is created.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await _eventBridgeWrapper.PutS3UploadRule(roleArn, eventRuleName,
testBucketName);
        Console.WriteLine($"\tAdded event rule {eventRuleName} for bucket
{testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an SNS target to the rule.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task AddSnsTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Adding a target to the rule to that sends an email when
the rule is triggered.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];
        var topicName = _configuration["topicName"];
        await _eventBridgeWrapper.AddSnsTargetToRule(eventRuleName, topicArn);
        Console.WriteLine($"\tAdded event rule {eventRuleName} with Amazon SNS
target {topicName} for bucket {testBucketName}.");
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the event rules on the default event bus.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListEventRules()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Current event rules:");

        var rules = await _eventBridgeWrapper.ListAllRulesForEventBus();
        rules.ForEach(r => Console.WriteLine($"\tRule: {r.Name} Description:
{r.Description} State: {r.State}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the event target to use a transform.
    /// </summary>
    /// <param name="topicArn">The SNS topic ARN target to update.</param>
    /// <returns>Async task.</returns>
    private static async Task UpdateSnsEventRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Let's update the event target with a transform.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await
_eventBridgeWrapper.UpdateS3UploadRuleTargetWithTransform(eventRuleName, topicArn);
        Console.WriteLine($"\tUpdated event rule {eventRuleName} with Amazon SNS
target {topicArn} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the rule to use a custom event pattern.
    /// </summary>
```

```
    /// <returns>Async task.</returns>
    private static async Task UpdateToCustomRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Updating the event pattern to be triggered by a custom
event instead.");

        var eventRuleName = _configuration["eventRuleName"];

        await _eventBridgeWrapper.UpdateCustomEventPattern(eventRuleName);

        Console.WriteLine($"\tUpdated event rule {eventRuleName} to custom
pattern.");
        await _eventBridgeWrapper.UpdateCustomRuleTargetWithTransform(eventRuleName,
            topicArn);

        Console.WriteLine($"\tUpdated event target {topicArn}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Send rule events for a custom rule using the user's email address.
    /// </summary>
    /// <param name="email">The email address to include.</param>
    /// <returns>Async task.</returns>
    private static async Task TriggerCustomRule(string email)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Sending an event to trigger the rule. This will trigger a
subscription email.");

        await _eventBridgeWrapper.PutCustomEmailEvent(email);

        Console.WriteLine($"\tEvents have been sent. Press Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List all of the targets for a rule.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    private static async Task ListTargets()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the targets for a particular rule.");

        var eventRuleName = _configuration["eventRuleName"];
        var targets = await _eventBridgeWrapper.ListAllTargetsOnRule(eventRuleName);
        targets.ForEach(t => Console.WriteLine($"\tTarget: {t.Arn} Id: {t.Id} Input:
{t.Input}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List all of the rules for a particular target.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task ListRulesForTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the rules for a particular target.");

        var rules = await _eventBridgeWrapper.ListAllRuleNamesByTarget(topicArn);
        rules.ForEach(r => Console.WriteLine($"\tRule: {r}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Enable or disable a particular rule.
    /// </summary>
    /// <param name="isEnabled">True to enable the rule, otherwise false.</param>
    /// <returns>Async task.</returns>
    private static async Task ChangeRuleState(bool isEnabled)
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        if (!isEnabled)
        {
            Console.WriteLine($"Disabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.DisableRuleByName(eventRuleName);
        }
```

```
        else
        {
            Console.WriteLine($"Enabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.EnableRuleByName(eventRuleName);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the current state of the rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetRuleState()
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        var state = await _eventBridgeWrapper.GetRuleStateByRuleName(eventRuleName);
        Console.WriteLine($"Rule {eventRuleName} is in current state {state}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic to clean up.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        var eventRuleName = _configuration["eventRuleName"];
        if (GetYesNoResponse($"\tDelete all targets and event rule {eventRuleName}?
(y/n)"))
        {
            Console.WriteLine($"\tRemoving all targets from the event rule.");
            await _eventBridgeWrapper.RemoveAllTargetsFromRule(eventRuleName);

            Console.WriteLine($"\tDeleting event rule.");
            await _eventBridgeWrapper.DeleteRuleByName(eventRuleName);
        }
```

```
        var topicName = _configuration["topicName"];
        if (GetYesNoResponse($"\tDelete Amazon SNS subscription topic {topicName}?
(y/n)"))
        {
            Console.WriteLine($"\tDeleting topic.");
            await _snsClient!.DeleteTopicAsync(new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        }

        var bucketName = _configuration["testBucketName"];
        if (GetYesNoResponse($"\tDelete Amazon S3 bucket {bucketName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.S3Objects
                    .Select(o => new KeyVersion { Key = o.Key }).ToList()
            });
            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }

        var roleName = _configuration["roleName"];
        if (GetYesNoResponse($"\tDelete role {roleName}? (y/n)"))
        {
            Console.WriteLine($"\tDetaching policy and deleting role.");

            await _iamClient!.DetachRolePolicyAsync(new DetachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess",
```

```
            });

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = roleName
            });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Crie uma classe que envolva as EventBridge operações.

```
/// <summary>
/// Wrapper for Amazon EventBridge operations.
/// </summary>
public class EventBridgeWrapper
{
    private readonly IAmazonEventBridge _amazonEventBridge;
    private readonly ILogger<EventBridgeWrapper> _logger;

    /// <summary>
    /// Constructor for the EventBridge wrapper.
    /// </summary>
    /// <param name="amazonEventBridge">The injected EventBridge client.</param>
```

```
    /// <param name="logger">The injected logger for the wrapper.</param>
    public EventBridgeWrapper(IAmazonEventBridge amazonEventBridge,
ILogger<EventBridgeWrapper> logger)

    {
        _amazonEventBridge = amazonEventBridge;
        _logger = logger;
    }

    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }

    /// <summary>
    /// Enable a particular rule on an event bus.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
            new EnableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
```

```
    /// Disable a particular rule on an event bus.
    /// </summary
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
            new DisableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty, uses
 the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
 null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
```

```
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await _amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

```
    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created in
 a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]" +
                                        "}" +
                                    "}" +
                                "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }

    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the default
 event bus.</param>
    /// <returns>The ID of the target.</returns>
```

```
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string ruleName,
string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Update a custom rule with a transform on the target.
```

```
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the default
event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateCustomRuleTargetWithTransform(string ruleName,
string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputTemplate = "\"Notification: sample event was received.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
```

```
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }

    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
```

```
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }

    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string targetArn,
string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };

        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });

        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
```

```
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);

        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
```

```
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage}, code
 {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteRuleByName(string ruleName)
    {
        var response = await _amazonEventBridge.DeleteRuleAsync(
            new DeleteRuleRequest()
            {
                Name = ruleName
            });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Ações

`DeleteRule`

O código de exemplo a seguir mostra como usar `DeleteRule`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma regra pelo nome.

```
/// <summary>
/// Delete an event rule by name.
/// </summary>
/// <param name="ruleName">The name of the event rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteRuleByName(string ruleName)
{
    var response = await _amazonEventBridge.DeleteRuleAsync(
        new DeleteRuleRequest()
        {
            Name = ruleName
        });

    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DeleteRule em Referência AWS SDK para .NET da API.

`DescribeRule`

O código de exemplo a seguir mostra como usar `DescribeRule`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obter o estado de uma regra usando a descrição da regra.

```
    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }
```

- Para obter detalhes da API, consulte DescribeRule em Referência AWS SDK para .NET da API.

`DisableRule`

O código de exemplo a seguir mostra como usar `DisableRule`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Desabilitar uma regra pelo nome da regra.

```
/// <summary>
/// Disable a particular rule on an event bus.
/// </summary
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DisableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
        new DisableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DisableRule em Referência AWS SDK para .NET da API.

`EnableRule`

O código de exemplo a seguir mostra como usar `EnableRule`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Habilitar uma regra pelo nome da regra.

```
/// <summary>
/// Enable a particular rule on an event bus.
/// </summary>
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
        new EnableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte EnableRule em Referência AWS SDK para .NET da API.

`ListRuleNamesByTarget`

O código de exemplo a seguir mostra como usar `ListRuleNamesByTarget`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todos os nomes das regras usando o destino.

```
/// <summary>
/// List names of all rules matching a target.
/// </summary>
/// <param name="targetArn">The ARN of the target.</param>
/// <returns>The list of rule names.</returns>
public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
{
    var results = new List<string>();
```

```
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await _amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- Para obter detalhes da API, consulte ListRuleNamesByTarget em Referência AWS SDK para .NET da API.

`ListRules`

O código de exemplo a seguir mostra como usar `ListRules`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todas as regras para um barramento de eventos.

```
    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty, uses
the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
null)
```

```
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- Para obter detalhes da API, consulte ListRules em Referência AWS SDK para .NET da API.

`ListTargetsByRule`

O código de exemplo a seguir mostra como usar `ListTargetsByRule`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar todos os destinos para uma regra usando o nome da regra.

```
    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
```

```
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- Para obter detalhes da API, consulte ListTargetsByRule em Referência AWS SDK para .NET da API.

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie um evento que corresponda a um padrão personalizado para uma regra.

```
    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }
```

- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK para .NET da API.

`PutRule`

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma regra que seja acionada quando um objeto é adicionado a um bucket do Amazon Simple Storage Service.

```
    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created in
 a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]" +
                                        "}" +
                                    "}" +
                                "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }
```

Crie uma regra que utilize um padrão personalizado.

```
    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
```

```
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }
```

- Para obter detalhes da API, consulte PutRule em Referência AWS SDK para .NET da API.

`PutTargets`

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione um tópico do Amazon SNS como um destino para uma regra.

```
    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
```

```
    public async Task<string> AddSnsTargetToRule(string ruleName, string targetArn,
string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };

        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });

        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }
```

Adicione um transformador de entrada a um destino para uma regra.

```
    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
```

```
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the default
event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string ruleName,
string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
```

```
        return targetID;
    }
```

- Para obter detalhes da API, consulte PutTargets em Referência AWS SDK para .NET da API.

RemoveTargets

O código de exemplo a seguir mostra como usar `RemoveTargets`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Remover todos os destinos de uma regra usando o nome da regra.

```
    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);
```

```
        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte RemoveTargets em Referência AWS SDK para .NET da API.

EventBridge Exemplos de agendadores usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with EventBridge Scheduler.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá EventBridge Scheduler

O exemplo de código a seguir mostra como começar a usar o EventBridge Scheduler.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static class HelloScheduler
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the EventBridge Scheduler service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonScheduler>()
            ).Build();

        // Now the client is available for injection.
        var schedulerClient = host.Services.GetRequiredService<IAmazonScheduler>();

        // You can use await and any of the async methods to get a response, or a
 paginator to list schedules or groups.
        var results = new List<ScheduleSummary>();
        var paginateSchedules = schedulerClient.Paginators.ListSchedules(
            new ListSchedulesRequest());
        Console.WriteLine(
            $"Hello AWS Scheduler! Let's list schedules in your account.");
        // Get the entire list using the paginator.
```

```
        await foreach (var schedule in paginateSchedules.Schedules)
        {
            results.Add(schedule);
        }
        Console.WriteLine($"\tTotal of {results.Count} schedule(s) available.");
        results.ForEach(s => Console.WriteLine($"\tSchedule: {s.Name}"));
    }
}
```

- Para obter detalhes da API, consulte ListSchedules em Referência AWS SDK para .NET da API.

Ações

`CreateSchedule`

O código de exemplo a seguir mostra como usar `CreateSchedule`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a new schedule in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule.</param>
    /// <param name="scheduleExpression">The schedule expression that defines when
 the schedule should run.</param>
    /// <param name="scheduleGroupName">The name of the schedule group to which the
 schedule should be added.</param>
    /// <param name="deleteAfterCompletion">Indicates whether to delete the schedule
 after completion.</param>
    /// <param name="useFlexibleTimeWindow">Indicates whether to use a flexible time
 window for the schedule.</param>
    /// <param name="targetArn">ARN of the event target.</param>
    /// <param name="roleArn">Execution Role ARN.</param>
```

```
    /// <returns>True if the schedule was created successfully, false otherwise.</
returns>
    public async Task<bool> CreateScheduleAsync(
            string name,
            string scheduleExpression,
            string scheduleGroupName,
            string targetArn,
            string roleArn,
            string input,
            bool deleteAfterCompletion = false,
            bool useFlexibleTimeWindow = false)
    {
        try
        {
            int hoursToRun = 1;
            int flexibleTimeWindowMinutes = 10;

            var request = new CreateScheduleRequest
            {
                Name = name,
                ScheduleExpression = scheduleExpression,
                GroupName = scheduleGroupName,
                Target = new Target { Arn = targetArn, RoleArn = roleArn, Input =
 input },
                ActionAfterCompletion = deleteAfterCompletion
                    ? ActionAfterCompletion.DELETE
                    : ActionAfterCompletion.NONE,
                StartDate = DateTime.UtcNow, // Ignored for one-time schedules.
                EndDate =
                    DateTime.UtcNow
                        .AddHours(hoursToRun) // Ignored for one-time schedules.
            };
            // Allow a flexible time window if the caller specifies it.
            request.FlexibleTimeWindow = new FlexibleTimeWindow
            {
                Mode = useFlexibleTimeWindow
                    ? FlexibleTimeWindowMode.FLEXIBLE
                    : FlexibleTimeWindowMode.OFF,
                MaximumWindowInMinutes = useFlexibleTimeWindow
                    ? flexibleTimeWindowMinutes
                    : null
            };

            var response = await _amazonScheduler.CreateScheduleAsync(request);
```

```
            Console.WriteLine($"Successfully created schedule '{name}' " +
                              $"in schedule group '{scheduleGroupName}':
{response.ScheduleArn}.");
            return true;
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule '{name}' due to a conflict.
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating schedule '{name}' "
+
                             $"in schedule group '{scheduleGroupName}':
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateSchedule em Referência AWS SDK para .NET da API.

`CreateScheduleGroup`

O código de exemplo a seguir mostra como usar `CreateScheduleGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a new schedule group in Amazon EventBridge Scheduler.
```

```
    /// </summary>
    /// <param name="name">The name of the schedule group.</param>
    /// <returns>True if the schedule group was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleGroupAsync(string name)
    {
        try
        {
            var request = new CreateScheduleGroupRequest { Name = name };

            var response = await _amazonScheduler.CreateScheduleGroupAsync(request);

            Console.WriteLine($"Successfully created schedule group '{name}':
{response.ScheduleGroupArn}.");
            return true;

        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule group '{name}' due to a
conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating schedule group '{name}':
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateScheduleGroup em Referência AWS SDK para .NET da API.

`DeleteSchedule`

O código de exemplo a seguir mostra como usar `DeleteSchedule`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an existing schedule from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule to delete.</param>
    /// <param name="groupName">The group name of the schedule to delete.</param>
    /// <returns>True if the schedule was deleted successfully, false otherwise.</
returns>
    public async Task<bool> DeleteScheduleAsync(string name, string groupName)
    {
        try
        {
            var request = new DeleteScheduleRequest
            {
                Name = name,
                GroupName = groupName
            };

            await _amazonScheduler.DeleteScheduleAsync(request);

            Console.WriteLine($"Successfully deleted schedule with name '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule with ID '{name}' because the resource
 was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
```

```
                $"An error occurred while deleting schedule with ID '{name}':
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteSchedule em Referência AWS SDK para .NET da API.

`DeleteScheduleGroup`

O código de exemplo a seguir mostra como usar `DeleteScheduleGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an existing schedule group from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group to delete.</param>
    /// <returns>True if the schedule group was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleGroupAsync(string name)
    {
        try
        {
            var request = new DeleteScheduleGroupRequest { Name = name };

            await _amazonScheduler.DeleteScheduleGroupAsync(request);

            Console.WriteLine($"Successfully deleted schedule group '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
```

```
        {
            _logger.LogError(
                $"Failed to delete schedule group '{name}' because the resource was
not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule group '{name}':
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteScheduleGroup em Referência AWS SDK para .NET da API.

Cenários

Eventos agendados

O exemplo de código a seguir mostra como:

- Implante uma CloudFormation pilha com os recursos necessários.
- Crie um grupo de EventBridge agendamento do Scheduler.
- Crie uma programação única do EventBridge Scheduler com uma janela de tempo flexível.
- Crie uma programação recorrente do EventBridge Scheduler com uma taxa especificada.
- Exclua o EventBridge Scheduler, o agendamento e o grupo de agendamentos.
- Limpe os recursos e exclua a pilha.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário.

```
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.Scheduler;
using Amazon.Scheduler.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
using SchedulerActions;
using Exception = System.Exception;

namespace SchedulerScenario;

public class SchedulerWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:

    1. Prepare the Application:
       - Prompt the user for an email address to use for the subscription for the
 SNS topic subscription.
       - Prompt the user for a name for the Cloud Formation stack.
       - Deploy the Cloud Formation template in resources/cfn_template.yaml for
 resource creation.
       - Store the outputs of the stack into variables for use in the scenario.
       - Create a schedule group for all schedules.

    2. Create one-time Schedule:
       - Create a one-time schedule to send an initial event.
       - Use a Flexible Time Window and set the schedule to delete after completion.
       - Wait for the user to receive the event email from SNS.

    3. Create a time-based schedule:
       - Prompt the user for how many X times per Y hours a recurring event should
 be scheduled.
       - Create the scheduled event for X times per hour for Y hours.
       - Wait for the user to receive the event email from SNS.
```

```
       - Delete the schedule when the user is finished.

    4. Clean up:
       - Prompt the user for y/n answer if they want to destroy the stack and clean
 up all resources.
       - Delete the schedule group.
       - Destroy the Cloud Formation stack and wait until the stack has been
 removed.
    */

    public static ILogger<SchedulerWorkflow> _logger = null!;
    public static SchedulerWrapper _schedulerWrapper = null!;
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _roleArn = null!;
    private static string _snsTopicArn = null!;

    public static bool _interactive = true;
    private static string _stackName = "default-scheduler-scenario-stack-name";
    private static string _scheduleGroupName = "scenario-schedules-group";
    private static string _stackResourcePath = "../../../../../../scenarios/
features/eventbridge_scheduler/resources/cfn_template.yaml";

    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonScheduler>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<SchedulerWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<SchedulerWorkflow>();
```

```
            _schedulerWrapper =
host.Services.GetRequiredService<SchedulerWrapper>();
            _amazonCloudFormation =
host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge Scheduler Scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await CreateOneTimeSchedule();
                Console.WriteLine(new string('-', 80));

                Console.WriteLine(new string('-', 80));
                await CreateRecurringSchedule();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, initiating
cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("Amazon EventBridge Scheduler scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
```

```
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        try
        {
            // Prompt the user for an email address to use for the subscription.
            Console.WriteLine("\nThis example creates resources in a CloudFormation
stack, including an SNS topic" +
                              "\nthat will be subscribed to the EventBridge Scheduler
events. " +
                              "\n\nYou will need to confirm the subscription in order to
receive event emails. ");

            var emailAddress = PromptUserForEmail();

            // Prompt the user for a name for the CloudFormation stack
            _stackName = PromptUserForStackName();

            // Deploy the CloudFormation stack
            var deploySuccess = await DeployCloudFormationStack(_stackName,
emailAddress);

            if (deploySuccess)
            {
                // Create a schedule group for all schedules
                await
_schedulerWrapper.CreateScheduleGroupAsync(_scheduleGroupName);

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }
        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
```

```
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="email">The email to use for the subscription.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
string email)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = { Capability.CAPABILITY_NAMED_IAM }
            };

            // If an email is provided, set the parameter.
            if (!string.IsNullOrWhiteSpace(email))
            {
                request.Parameters = new List<Parameter>()
                {
                    new() { ParameterKey = "email", ParameterValue = email }
                };
            }

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                // Wait for the stack to be in CREATE_COMPLETE state
                bool stackCreated = await WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    // Retrieve the output values
                    var success = await GetStackOutputs(response.StackId);
                    return success;
                }
```

```
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already exists.
Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName, email);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="client">The CloudFormation client.</param>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 10;
        const int retryDelay = 30000; // 30 seconds.

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
```

```
            {
                StackName = stackId
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    Console.WriteLine("CloudFormation stack creation complete.");
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                        describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    Console.WriteLine("CloudFormation stack creation failed.");
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Retrieves the output values from the CloudFormation stack.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    private static async Task<bool> GetStackOutputs(string stackId)
    {
        try
        {
```

```
            var describeStacksRequest = new DescribeStacksRequest { StackName =
stackId };

            var describeStacksResponse =
                await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                var stack = describeStacksResponse.Stacks[0];
                _roleArn = GetStackOutputValue(stack, "RoleARN");
                _snsTopicArn = GetStackOutputValue(stack, "SNStopicARN");
                return true;
            }
            else
            {
                _logger.LogError($"No stack found for stack outputs: {stackId}");
                return false;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(
                ex, $"Failed to retrieve CloudFormation stack outputs: {stackId}");
            return false;
        }
    }

    /// <summary>
    /// Get an output value by key from a CloudFormation stack.
    /// </summary>
    /// <param name="stack">The CloudFormation stack.</param>
    /// <param name="outputKey">The key of the output.</param>
    /// <returns>The value as a string.</returns>
    private static string GetStackOutputValue(Stack stack, string outputKey)
    {
        var output = stack.Outputs.First(o => o.OutputKey == outputKey);
        var outputValue = output.OutputValue;
        Console.WriteLine($"Stack output {outputKey}: {outputValue}");
        return outputValue;
    }

    /// <summary>
    /// Creates a one-time schedule to send an initial event.
```

```
    /// </summary>
    /// <returns>True if the one-time schedule was created successfully.</returns>
    public static async Task<bool> CreateOneTimeSchedule()
    {
        var scheduleName =
            PromptUserForResourceName("Enter a name for the one-time schedule:");

        Console.WriteLine($"Creating a one-time schedule named '{scheduleName}' " +
                          $"\nto send an initial event in 1 minute with a flexible
time window...");
        try
        {
            // Create a one-time schedule with a flexible time
            // window set to delete after completion.
            // You may also set a timezone instead of using UTC.
            var scheduledTime = DateTime.UtcNow.AddMinutes(1).ToString("s");

            var createSuccess = await _schedulerWrapper.CreateScheduleAsync(
                scheduleName,
                $"at({scheduledTime})",
                _scheduleGroupName,
                _snsTopicArn,
                _roleArn,
                $"One time scheduled event test from schedule {scheduleName}.",
                true,
                useFlexibleTimeWindow: true);

            Console.WriteLine($"Subscription email will receive an email from this
event.");
            Console.WriteLine($"You must confirm your subscription to receive event
emails.");

            Console.WriteLine($"One-time schedule '{scheduleName}' created
successfully.");
            return createSuccess;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(ex, $"The target with ARN '{_snsTopicArn}' was not
found.");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the one-time
schedule '{scheduleName}'.");
            return false;
        }
    }

    /// <summary>
    /// Create a recurring schedule to send events at a specified rate in minutes.
    /// </summary>
    /// <returns>True if the recurring schedule was created successfully.</returns>
    public static async Task<bool> CreateRecurringSchedule()
    {
        Console.WriteLine("Creating a recurring schedule to send events for one
hour...");

        try
        {
            // Prompt the user for a schedule name.
            var scheduleName =
                PromptUserForResourceName("Enter a name for the recurring schedule:
");

            // Prompt the user for the schedule rate (in minutes).
            var scheduleRateInMinutes =
                PromptUserForInteger("Enter the desired schedule rate (in minutes):
");

            // Create the recurring schedule.
            var createSuccess = await _schedulerWrapper.CreateScheduleAsync(
                scheduleName,
                $"rate({scheduleRateInMinutes} minutes)",
                _scheduleGroupName,
                _snsTopicArn,
                _roleArn,
                $"Recurrent event test from schedule {scheduleName}.");

            Console.WriteLine($"Subscription email will receive an email from this
event.");
            Console.WriteLine($"You must confirm your subscription to receive event
emails.");

            // Delete the schedule when the user is finished.
            if (!_interactive || GetYesNoResponse($"Are you ready to delete the
'{scheduleName}' schedule? (y/n)"))
```

```
            {
                await _schedulerWrapper.DeleteScheduleAsync(scheduleName,
_scheduleGroupName);
            }

            return createSuccess;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(ex, $"The target with ARN '{_snsTopicArn}' was not
found.");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while creating the recurring
schedule.");
            return false;
        }
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    /// <returns>True if the cleanup was successful.</returns>
    public static async Task<bool> Cleanup()
    {
        // Prompt the user to confirm cleanup.
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete all resources created by this scenario? (y/n) ");
        if (cleanup)
        {
            try
            {
                // Delete the schedule group.
                var groupDeleteSuccess = await
_schedulerWrapper.DeleteScheduleGroupAsync(_scheduleGroupName);

                // Destroy the CloudFormation stack and wait for it to be removed.
                var stackDeleteSuccess = await DeleteCloudFormationStack(_stackName,
false);

                return groupDeleteSuccess && stackDeleteSuccess;
            }
```

```
            catch (Exception ex)
            {
                _logger.LogError(ex,
                    "An error occurred while cleaning up the resources.");
                return false;
            }
        }
        _logger.LogInformation("EventBridge Scheduler scenario is complete.");
        return true;
    }

    /// <summary>
    /// Delete the resources in the stack and wait for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the stack.</param>
    /// <param name="forceDelete">True to force delete the stack.</param>
    /// <returns>True if successful.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName, bool
forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{_stackName}' is being deleted.
This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(_stackName, forceDelete);

        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{_stackName}' has been
deleted.");
            return true;
        }
        else
        {
```

```
            _logger.LogError($"Failed to delete CloudFormation stack
'{_stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Wait for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the stack.</param>
    /// <param name="forceDelete">True to force delete the stack.</param>
    /// <returns>True if successful.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 10;
        const int retryDelay = 30000; // 30 seconds

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
describeStacksResponse.Stacks[0].StackStatus == StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_FAILED)
                {
                    // Try one time to force delete.
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
```

```
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                // Stack does not exist, so it has been successfully deleted.
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to be
deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack '{stackName}'
to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompt the user for a valid email address.
    /// </summary>
    /// <returns>The valid email address.</returns>
    private static string PromptUserForEmail()
    {
        if (_interactive)
        {
            Console.WriteLine("Enter an email address to use for event
subscriptions: ");

            string email = Console.ReadLine()!;
```

```
            if (!IsValidEmail(email))
            {
                Console.WriteLine("Invalid email address. Please try again.");
                return PromptUserForEmail();
            }
            return email;
        }
        // Used when running without user prompts.
        return "";
    }

    /// <summary>
    /// Prompt the user for a non-empty stack name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static string PromptUserForStackName()
    {
        Console.WriteLine("Enter a name for the AWS Cloud Formation Stack: ");
        if (_interactive)
        {
            string stackName = Console.ReadLine()!;
            var regex = "[a-zA-Z][-a-zA-Z0-9]|arn:[-a-zA-Z0-9:/._+]";
            if (!Regex.IsMatch(stackName, regex))
            {
                Console.WriteLine(
                    $"Invalid stack name. Please use a name that matches the pattern
{regex}.");
                return PromptUserForStackName();
            }

            return stackName;
        }
        // Used when running without user prompts.
        return _stackName;
    }

    /// <summary>
    /// Prompt the user for a non-empty resource name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static string PromptUserForResourceName(string prompt)
    {
        if (_interactive)
```

```
        {
            Console.WriteLine(prompt);
            string resourceName = Console.ReadLine()!;
            var regex = "[0-9a-zA-Z-_.]+";
            if (!Regex.IsMatch(resourceName, regex))
            {
                Console.WriteLine($"Invalid resource name. Please use a name that
matches the pattern {regex}.");
                return PromptUserForResourceName(prompt);
            }
            return resourceName!;
        }
        // Used when running without user prompts.
        return "resource-" + Guid.NewGuid();
    }

    /// <summary>
    /// Prompt the user for a non-empty resource name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static int PromptUserForInteger(string prompt)
    {
        if (_interactive)
        {
            Console.WriteLine(prompt);
            string stringResponse = Console.ReadLine()!;
            if (string.IsNullOrWhiteSpace(stringResponse) ||
                !Int32.TryParse(stringResponse, out var intResponse))
            {
                Console.WriteLine($"Invalid integer. ");
                return PromptUserForInteger(prompt);
            }
            return intResponse!;
        }
        // Used when running without user prompts.
        return 1;
    }

    /// <summary>
    /// Use System Mail to check for a valid email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email address.</returns>
    private static bool IsValidEmail(string email)
```

```
    {
        try
        {
            var mailAddress = new System.Net.Mail.MailAddress(email);
            return mailAddress.Address == email;
        }
        catch
        {
            // Invalid emails will cause an exception, return false.
            return false;
        }
    }
}
```

Wrapper para operações de serviço.

```
using Amazon.Scheduler;
using Amazon.Scheduler.Model;
using Microsoft.Extensions.Logging;

namespace SchedulerActions;

/// <summary>
/// Wrapper class for Amazon EventBridge Scheduler operations.
/// </summary>
public class SchedulerWrapper
{
    private readonly IAmazonScheduler _amazonScheduler;
    private readonly ILogger<SchedulerWrapper> _logger;

    /// <summary>
    /// Constructor for the SchedulerWrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EventBridge Scheduler client.</
param>
    /// <param name="logger">The injected logger.</param>
    public SchedulerWrapper(IAmazonScheduler amazonScheduler,
 ILogger<SchedulerWrapper> logger)
    {
        _amazonScheduler = amazonScheduler;
        _logger = logger;
    }
```

```
    /// <summary>
    /// Creates a new schedule in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule.</param>
    /// <param name="scheduleExpression">The schedule expression that defines when
 the schedule should run.</param>
    /// <param name="scheduleGroupName">The name of the schedule group to which the
 schedule should be added.</param>
    /// <param name="deleteAfterCompletion">Indicates whether to delete the schedule
 after completion.</param>
    /// <param name="useFlexibleTimeWindow">Indicates whether to use a flexible time
 window for the schedule.</param>
    /// <param name="targetArn">ARN of the event target.</param>
    /// <param name="roleArn">Execution Role ARN.</param>
    /// <returns>True if the schedule was created successfully, false otherwise.</
returns>
    public async Task<bool> CreateScheduleAsync(
            string name,
            string scheduleExpression,
            string scheduleGroupName,
            string targetArn,
            string roleArn,
            string input,
            bool deleteAfterCompletion = false,
            bool useFlexibleTimeWindow = false)
    {
        try
        {
            int hoursToRun = 1;
            int flexibleTimeWindowMinutes = 10;

            var request = new CreateScheduleRequest
            {
                Name = name,
                ScheduleExpression = scheduleExpression,
                GroupName = scheduleGroupName,
                Target = new Target { Arn = targetArn, RoleArn = roleArn, Input =
 input },
                ActionAfterCompletion = deleteAfterCompletion
                    ? ActionAfterCompletion.DELETE
                    : ActionAfterCompletion.NONE,
                StartDate = DateTime.UtcNow, // Ignored for one-time schedules.
                EndDate =
```

```
                    DateTime.UtcNow
                        .AddHours(hoursToRun) // Ignored for one-time schedules.
            };
            // Allow a flexible time window if the caller specifies it.
            request.FlexibleTimeWindow = new FlexibleTimeWindow
            {
                Mode = useFlexibleTimeWindow
                    ? FlexibleTimeWindowMode.FLEXIBLE
                    : FlexibleTimeWindowMode.OFF,
                MaximumWindowInMinutes = useFlexibleTimeWindow
                    ? flexibleTimeWindowMinutes
                    : null
            };

            var response = await _amazonScheduler.CreateScheduleAsync(request);

            Console.WriteLine($"Successfully created schedule '{name}' " +
                              $"in schedule group '{scheduleGroupName}':
{response.ScheduleArn}.");
            return true;
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule '{name}' due to a conflict.
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating schedule '{name}' "
+
                             $"in schedule group '{scheduleGroupName}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Creates a new schedule group in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group.</param>
    /// <returns>True if the schedule group was created successfully, false
otherwise.</returns>
```

```
    public async Task<bool> CreateScheduleGroupAsync(string name)
    {
        try
        {
            var request = new CreateScheduleGroupRequest { Name = name };

            var response = await _amazonScheduler.CreateScheduleGroupAsync(request);

            Console.WriteLine($"Successfully created schedule group '{name}':
 {response.ScheduleGroupArn}.");
            return true;

        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule group '{name}' due to a
 conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating schedule group '{name}':
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an existing schedule from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule to delete.</param>
    /// <param name="groupName">The group name of the schedule to delete.</param>
    /// <returns>True if the schedule was deleted successfully, false otherwise.</
returns>
    public async Task<bool> DeleteScheduleAsync(string name, string groupName)
    {
        try
        {
            var request = new DeleteScheduleRequest
            {
                Name = name,
                GroupName = groupName
```

```
            };

            await _amazonScheduler.DeleteScheduleAsync(request);

            Console.WriteLine($"Successfully deleted schedule with name '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule with ID '{name}' because the resource
was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule with ID '{name}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an existing schedule group from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group to delete.</param>
    /// <returns>True if the schedule group was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleGroupAsync(string name)
    {
        try
        {
            var request = new DeleteScheduleGroupRequest { Name = name };

            await _amazonScheduler.DeleteScheduleGroupAsync(request);

            Console.WriteLine($"Successfully deleted schedule group '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
```

```
            _logger.LogError(
                $"Failed to delete schedule group '{name}' because the resource was
 not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule group '{name}':
 {ex.Message}");
            return false;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Exemplos de uso do Amazon Glacier SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon Glacier.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Glacier

O exemplo de código a seguir mostra como começar a usar o Amazon Glacier.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.Glacier;
using Amazon.Glacier.Model;

namespace GlacierActions;

public static class HelloGlacier
{
    static async Task Main()
    {
        var glacierService = new AmazonGlacierClient();

        Console.WriteLine("Hello Amazon Glacier!");
        Console.WriteLine("Let's list your Glacier vaults:");

        // You can use await and any of the async methods to get a response.
        // Let's get the vaults using a paginator.
        var glacierVaultPaginator = glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });

        await foreach (var vault in glacierVaultPaginator.VaultList)
        {
            Console.WriteLine($"{vault.CreationDate}:{vault.VaultName}, ARN:
{vault.VaultARN}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListVaults em Referência AWS SDK para .NET da API.

Ações

`AddTagsToVault`

O código de exemplo a seguir mostra como usar `AddTagsToVault`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Add tags to the items in an Amazon S3 Glacier vault.
   /// </summary>
   /// <param name="vaultName">The name of the vault to add tags to.</param>
   /// <param name="key">The name of the object to tag.</param>
   /// <param name="value">The tag value to add.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> AddTagsToVaultAsync(string vaultName, string key, string
value)
   {
       var request = new AddTagsToVaultRequest
       {
           Tags = new Dictionary<string, string>
               {
                   { key, value },
               },
           AccountId = "-",
           VaultName = vaultName,
       };

       var response = await _glacierService.AddTagsToVaultAsync(request);
       return response.HttpStatusCode == HttpStatusCode.NoContent;
   }
```

- Para obter detalhes da API, consulte AddTagsToVault em Referência AWS SDK para .NET da API.

`CreateVault`

O código de exemplo a seguir mostra como usar `CreateVault`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create an Amazon S3 Glacier vault.
/// </summary>
/// <param name="vaultName">The name of the vault to create.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateVaultAsync(string vaultName)
{
    var request = new CreateVaultRequest
    {
        // Setting the AccountId to "-" means that
        // the account associated with the current
        // account will be used.
        AccountId = "-",
        VaultName = vaultName,
    };

    var response = await _glacierService.CreateVaultAsync(request);

    Console.WriteLine($"Created {vaultName} at: {response.Location}");

    return response.HttpStatusCode == HttpStatusCode.Created;
}
```

- Para obter detalhes da API, consulte CreateVault em Referência AWS SDK para .NET da API.

`DescribeVault`

O código de exemplo a seguir mostra como usar `DescribeVault`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to describe.</param>
    /// <returns>The Amazon Resource Name (ARN) of the vault.</returns>
    public async Task<string> DescribeVaultAsync(string vaultName)
    {
        var request = new DescribeVaultRequest
        {
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.DescribeVaultAsync(request);

        // Display the information about the vault.
        Console.WriteLine($"{response.VaultName}\tARN: {response.VaultARN}");
        Console.WriteLine($"Created on: {response.CreationDate}\tNumber of Archives:
{response.NumberOfArchives}\tSize (in bytes): {response.SizeInBytes}");
        if (response.LastInventoryDate != DateTime.MinValue)
        {
            Console.WriteLine($"Last inventory: {response.LastInventoryDate}");
        }

        return response.VaultARN;
    }
```

- Para obter detalhes da API, consulte DescribeVault em Referência AWS SDK para .NET da API.

InitiateJob

O código de exemplo a seguir mostra como usar `InitiateJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recupere um arquivo de um cofre. Este exemplo usa a ArchiveTransferManager classe. Para obter detalhes da API, consulte ArchiveTransferManager.

```
    /// <summary>
    /// Download an archive from an Amazon S3 Glacier vault using the Archive
    /// Transfer Manager.
    /// </summary>
    /// <param name="vaultName">The name of the vault containing the object.</param>
    /// <param name="archiveId">The Id of the archive to download.</param>
    /// <param name="localFilePath">The local directory where the file will
    /// be stored after download.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> DownloadArchiveWithArchiveManagerAsync(string vaultName,
string archiveId, string localFilePath)
    {
        try
        {
            var manager = new ArchiveTransferManager(_glacierService);

            var options = new DownloadOptions
            {
                StreamTransferProgress = Progress!,
            };

            // Download an archive.
            Console.WriteLine("Initiating the archive retrieval job and then polling
SQS queue for the archive to be available.");
```

```
            Console.WriteLine("When the archive is available, downloading will
begin.");
            await manager.DownloadAsync(vaultName, archiveId, localFilePath,
options);

            return true;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Event handler to track the progress of the Archive Transfer Manager.
    /// </summary>
    /// <param name="sender">The object that raised the event.</param>
    /// <param name="args">The argument values from the object that raised the
    /// event.</param>
    static void Progress(object sender, StreamTransferProgressArgs args)
    {
        if (args.PercentDone != _currentPercentage)
        {
            _currentPercentage = args.PercentDone;
            Console.WriteLine($"Downloaded {_currentPercentage}%");
        }
    }
```

- Para obter detalhes da API, consulte InitiateJob em Referência AWS SDK para .NET da API.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// List Amazon S3 Glacier jobs.
/// </summary>
/// <param name="vaultName">The name of the vault to list jobs for.</param>
/// <returns>A list of Amazon S3 Glacier jobs.</returns>
public async Task<List<GlacierJobDescription>> ListJobsAsync(string vaultName)
{
    var request = new ListJobsRequest
    {
        // Using a hyphen "-" for the Account Id will
        // cause the SDK to use the Account Id associated
        // with the current account.
        AccountId = "-",
        VaultName = vaultName,
    };

    var response = await _glacierService.ListJobsAsync(request);

    return response.JobList;
}
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para .NET da API.

`ListTagsForVault`

O código de exemplo a seguir mostra como usar `ListTagsForVault`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List tags for an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list tags for.</param>
    /// <returns>A dictionary listing the tags attached to each object in the
    /// vault and its tags.</returns>
    public async Task<Dictionary<string, string>> ListTagsForVaultAsync(string
vaultName)
    {
        var request = new ListTagsForVaultRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListTagsForVaultAsync(request);

        return response.Tags;
    }
```

- Para obter detalhes da API, consulte ListTagsForVault em Referência AWS SDK para .NET da API.

`ListVaults`

O código de exemplo a seguir mostra como usar `ListVaults`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// List the Amazon S3 Glacier vaults associated with the current account.
/// </summary>
/// <returns>A list containing information about each vault.</returns>
public async Task<List<DescribeVaultOutput>> ListVaultsAsync()
{
    var glacierVaultPaginator = _glacierService.Paginators.ListVaults(
        new ListVaultsRequest { AccountId = "-" });
    var vaultList = new List<DescribeVaultOutput>();

    await foreach (var vault in glacierVaultPaginator.VaultList)
    {
        vaultList.Add(vault);
    }

    return vaultList;
}
```

- Para obter detalhes da API, consulte ListVaults em Referência AWS SDK para .NET da API.

`UploadArchive`

O código de exemplo a seguir mostra como usar `UploadArchive`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Upload an object to an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the Amazon S3 Glacier vault to upload
    /// the archive to.</param>
    /// <param name="archiveFilePath">The file path of the archive to upload to the
vault.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<string> UploadArchiveWithArchiveManager(string vaultName,
string archiveFilePath)
    {
        try
        {
            var manager = new ArchiveTransferManager(_glacierService);

            // Upload an archive.
            var response = await manager.UploadAsync(vaultName, "upload archive
test", archiveFilePath);
            return response.ArchiveId;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return string.Empty;
        }
    }
```

- Para obter detalhes da API, consulte UploadArchive em Referência AWS SDK para .NET da API.

AWS Glue exemplos usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace GlueActions;

public class HelloGlue
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonGlue>()
                .AddTransient<GlueWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloGlue>();
        var glueClient = host.Services.GetRequiredService<IAmazonGlue>();

        var request = new ListJobsRequest();

        var jobNames = new List<string>();

        do
        {
            var response = await glueClient.ListJobsAsync(request);
            jobNames.AddRange(response.JobNames);
            request.NextToken = response.NextToken;
        }
        while (request.NextToken is not null);

        Console.Clear();
        Console.WriteLine("Hello, Glue. Let's list your existing Glue Jobs:");
        if (jobNames.Count == 0)
        {
            Console.WriteLine("You don't have any AWS Glue jobs.");
        }
        else
        {
            jobNames.ForEach(Console.WriteLine);
        }
    }
}
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que agrupe as AWS Glue funções que são usadas no cenário.

```
using System.Net;

namespace GlueActions;

public class GlueWrapper
{
    private readonly IAmazonGlue _amazonGlue;

    /// <summary>
    /// Constructor for the AWS Glue actions wrapper.
    /// </summary>
    /// <param name="amazonGlue"></param>
    public GlueWrapper(IAmazonGlue amazonGlue)
    {
```

```
        _amazonGlue = amazonGlue;
    }

    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be executed.</
param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service (Amazon
 S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
```

```
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName, string
bucketUrl, string jobName, string roleName, string description, string scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
```

```
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new DeleteCrawlerRequest
{ Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
```

```
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }
```

```
    /// <summary>
    /// Get information about the state of an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A value describing the state of the crawler.</returns>
    public async Task<CrawlerState> GetCrawlerStateAsync(string crawlerName)
    {
        var response = await _amazonGlue.GetCrawlerAsync(
            new GetCrawlerRequest { Name = crawlerName });
        return response.Crawler.State;
    }


    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }


    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
 { JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }
```

```
    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us behind
the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }


    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
```

```
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }


    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new ListJobsRequest
{ MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }


    /// <summary>
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }

}
```

Crie uma classe que execute o cenário.

```
global using Amazon.Glue;
global using GlueActions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
```

```
global using Microsoft.Extensions.Logging.Debug;



using Amazon.Glue.Model;
using Amazon.S3;
using Amazon.S3.Model;

namespace GlueBasics;

public class GlueBasics
{
    private static ILogger logger = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonGlue>()
            .AddTransient<GlueWrapper>()
            .AddTransient<UiWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
        .CreateLogger<GlueBasics>();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // These values are stored in settings.json
        // Once you have run the CDK script to deploy the resources,
        // edit the file to set "BucketName", "RoleName", and "ScriptURL"
```

```
        // to the appropriate values. Also set "CrawlerName" to the name
        // you want to give the crawler when it is created.
        string bucketName = _configuration["BucketName"]!;
        string bucketUrl = _configuration["BucketUrl"]!;
        string crawlerName = _configuration["CrawlerName"]!;
        string roleName = _configuration["RoleName"]!;
        string sourceData = _configuration["SourceData"]!;
        string dbName = _configuration["DbName"]!;
        string cron = _configuration["Cron"]!;
        string scriptUrl = _configuration["ScriptURL"]!;
        string jobName = _configuration["JobName"]!;

        var wrapper = host.Services.GetRequiredService<GlueWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UiWrapper>();

        uiWrapper.DisplayOverview();
        uiWrapper.PressEnter();

        // Create the crawler and wait for it to be ready.
        uiWrapper.DisplayTitle("Create AWS Glue crawler");
        Console.WriteLine("Let's begin by creating the AWS Glue crawler.");

        var crawlerDescription = "Crawler created for the AWS Glue Basics
scenario.";
        var crawlerCreated = await wrapper.CreateCrawlerAsync(crawlerName,
crawlerDescription, roleName, cron, sourceData, dbName);
        if (crawlerCreated)
        {
            Console.WriteLine($"The crawler: {crawlerName} has been created. Now
let's wait until it's ready.");
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for use.");
        }
        else
        {
            Console.WriteLine($"Couldn't create crawler {crawlerName}.");
            return; // Exit the application.
        }
```

```
        uiWrapper.DisplayTitle("Start AWS Glue crawler");
        Console.WriteLine("Now let's wait until the crawler has successfully
started.");
        var crawlerStarted = await wrapper.StartCrawlerAsync(crawlerName);
        if (crawlerStarted)
        {
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for use.");
        }
        else
        {
            Console.WriteLine($"Couldn't start the crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.PressEnter();

        Console.WriteLine($"\nLet's take a look at the database: {dbName}");
        var database = await wrapper.GetDatabaseAsync(dbName);

        if (database != null)
        {
            uiWrapper.DisplayTitle($"{database.Name} Details");
            Console.WriteLine($"{database.Name} created on {database.CreateTime}");
            Console.WriteLine(database.Description);
        }

        uiWrapper.PressEnter();

        var tables = await wrapper.GetTablesAsync(dbName);
        if (tables.Count > 0)
        {
            tables.ForEach(table =>
            {
                Console.WriteLine($"{table.Name}\tCreated:
{table.CreateTime}\tUpdated: {table.UpdateTime}");
            });
        }
```

```
        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create AWS Glue job");
        Console.WriteLine("Creating a new AWS Glue job.");
        var description = "An AWS Glue job created using the AWS SDK for .NET";
        await wrapper.CreateJobAsync(dbName, tables[0].Name, bucketUrl, jobName,
roleName, description, scriptUrl);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Starting AWS Glue job");
        Console.WriteLine("Starting the new AWS Glue job...");
        var jobRunId = await wrapper.StartJobRunAsync(jobName, dbName,
tables[0].Name, bucketName);
        var jobRunComplete = false;
        var jobRun = new JobRun();
        do
        {
            jobRun = await wrapper.GetJobRunAsync(jobName, jobRunId);
            if (jobRun.JobRunState == "SUCCEEDED" || jobRun.JobRunState == "STOPPED"
||
                jobRun.JobRunState == "FAILED" || jobRun.JobRunState == "TIMEOUT")
            {
                jobRunComplete = true;
            }
        } while (!jobRunComplete);

        uiWrapper.DisplayTitle($"Data in {bucketName}");

        // Get the list of data stored in the S3 bucket.
        var s3Client = new AmazonS3Client();

        var response = await s3Client.ListObjectsAsync(new ListObjectsRequest
{ BucketName = bucketName });
        response.S3Objects.ForEach(s3Object =>
        {
            Console.WriteLine(s3Object.Key);
        });

        uiWrapper.DisplayTitle("AWS Glue jobs");
        var jobNames = await wrapper.ListJobsAsync();
        jobNames.ForEach(jobName =>
        {
            Console.WriteLine(jobName);
```

```
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Get AWS Glue job run information");
        Console.WriteLine("Getting information about the AWS Glue job.");
        var jobRuns = await wrapper.GetJobRunsAsync(jobName);

        jobRuns.ForEach(jobRun =>
        {

 Console.WriteLine($"{jobRun.JobName}\t{jobRun.JobRunState}\t{jobRun.CompletedOn}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Deleting resources");
        Console.WriteLine("Deleting the AWS Glue job used by the example.");
        await wrapper.DeleteJobAsync(jobName);

        Console.WriteLine("Deleting the tables from the database.");
        tables.ForEach(async table =>
        {
            await wrapper.DeleteTableAsync(dbName, table.Name);
        });

        Console.WriteLine("Deleting the database.");
        await wrapper.DeleteDatabaseAsync(dbName);

        Console.WriteLine("Deleting the AWS Glue crawler.");
        await wrapper.DeleteCrawlerAsync(crawlerName);

        Console.WriteLine("The AWS Glue scenario has completed.");
        uiWrapper.PressEnter();
    }
}


namespace GlueBasics;

public class UiWrapper
{
    public readonly string SepBar = new string('-', Console.WindowWidth);
```

```
    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Amazon Glue: get started with crawlers and jobs");

        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t 1. Create a crawler, pass it the IAM role and the URL
to the public S3 bucket that contains the source data");
        Console.WriteLine("\t 2. Start the crawler.");
        Console.WriteLine("\t 3. Get the database created by the crawler and the
tables in the database.");
        Console.WriteLine("\t 4. Create a job.");
        Console.WriteLine("\t 5. Start a job run.");
        Console.WriteLine("\t 6. Wait for the job run to complete.");
        Console.WriteLine("\t 7. Show the data stored in the bucket.");
        Console.WriteLine("\t 8. List jobs for the account.");
        Console.WriteLine("\t 9. Get job run details for the job that was run.");
        Console.WriteLine("\t10. Delete the demo job.");
        Console.WriteLine("\t11. Delete the database and tables created for the
demo.");
        Console.WriteLine("\t12. Delete the crawler.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPlease press <Enter> to continue. ");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to center on the screen.</param>
    /// <returns>The string padded to make it center on the screen.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
```

```
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be executed.</
param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service (Amazon
 S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
```

```
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte CreateCrawler em Referência AWS SDK para .NET da API.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName, string
 bucketUrl, string jobName, string roleName, string description, string scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK para .NET da API.

`DeleteCrawler`

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new DeleteCrawlerRequest
{ Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteCrawler em Referência AWS SDK para .NET da API.

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteDatabase em Referência AWS SDK para .NET da API.

`DeleteJob`

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteJob em Referência AWS SDK para .NET da API.

DeleteTable

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para .NET da API.

GetCrawler

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }
```

- Para obter detalhes da API, consulte GetCrawler em Referência AWS SDK para .NET da API.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }
```

- Para obter detalhes da API, consulte GetDatabase em Referência AWS SDK para .NET da API.

`GetJobRun`

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }
```

- Para obter detalhes da API, consulte GetJobRun em Referência AWS SDK para .NET da API.

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
```

```
        };

        // No need to loop to get all the log groups--the SDK does it for us behind
the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }
```

- Para obter detalhes da API, consulte GetJobRuns em Referência AWS SDK para .NET da API.

`GetTables`

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
```

```
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }
```

- Para obter detalhes da API, consulte GetTables em Referência AWS SDK para .NET da API.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new ListJobsRequest
{ MaxResults = 10 });
```

```
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para .NET da API.

StartCrawler

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte StartCrawler em Referência AWS SDK para .NET da API.

StartJobRun

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Start an AWS Glue job run.
/// </summary>
/// <param name="jobName">The name of the job.</param>
/// <returns>A string representing the job run Id.</returns>
public async Task<string> StartJobRunAsync(
    string jobName,
    string inputDatabase,
    string inputTable,
    string bucketName)
{
    var request = new StartJobRunRequest
    {
        JobName = jobName,
        Arguments = new Dictionary<string, string>
        {
            {"--input_database", inputDatabase},
            {"--input_table", inputTable},
            {"--output_bucket_url", $"s3://{bucketName}/"}
        }
    };

    var response = await _amazonGlue.StartJobRunAsync(request);
    return response.JobRunId;
```

```
    }
```

- Para obter detalhes da API, consulte StartJobRun em Referência AWS SDK para .NET da API.

Exemplos de IAM usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();

        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
            Console.WriteLine($"Created:
 {policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;
```

```
public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
```

```
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }


    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
```

```
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName, string
 description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
 { UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new DeletePolicyRequest
{ PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
```

```
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM policy.</
param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {
```

```
        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
{ PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest { UserName
= userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>> ListAttachedRolePoliciesAsync(string
roleName)
    {
```

```
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new ListAttachedRolePoliciesRequest
{ RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
```

```
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator = _IAMService.Paginators.ListRolePolicies(new
ListRolePoliciesRequest { RoleName = roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
```

```
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
 ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
 ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string roleName,
 string policyDocument)
```

```
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM policy.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string policyName,
 string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;
```

```
        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId = accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}



using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
```

```
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
```

```
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon S3
buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();
```

```
        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");
```

```
        var assumedRoleCredentials = await s3Wrapper.AssumeS3RoleAsync("temporary-
session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
 {bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
 complete.");
        Console.WriteLine("Please wait while we clean up the resources we
 created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();
```

```
        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
```

```
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
```

```
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
 { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
```

```
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
(IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it full
access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries to
list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }
```

```
    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Attach an IAM policy to a role.
/// </summary>
/// <param name="policyArn">The policy to attach.</param>
/// <param name="roleName">The role that the policy will be attached to.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte AttachRolePolicy em Referência AWS SDK para .NET da API.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
        {
```

```
            UserName = userName,
        });

        return response.AccessKey;

    }
```

- Para obter detalhes da API, consulte CreateAccessKey em Referência AWS SDK para .NET da API.

`CreateInstanceProfile`

O código de exemplo a seguir mostra como usar `CreateInstanceProfile`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
```

```
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
```

```
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
```

```
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }
```

- Para obter detalhes da API, consulte CreateInstanceProfile em Referência AWS SDK para .NET da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK para .NET da API.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }
```

- Para obter detalhes da API, consulte CreateRole em Referência AWS SDK para .NET da API.

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName, string
 description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }
```

- Para obter detalhes da API, consulte CreateServiceLinkedRole em Referência AWS SDK para .NET da API.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }
```

- Para obter detalhes da API, consulte CreateUser em Referência AWS SDK para .NET da API.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
```

```
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteAccessKey em Referência AWS SDK para .NET da API.

`DeleteInstanceProfile`

O código de exemplo a seguir mostra como usar `DeleteInstanceProfile`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
```

```
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }
```

- Para obter detalhes da API, consulte DeleteInstanceProfile em Referência AWS SDK para .NET da API.

DeletePolicy

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new DeletePolicyRequest
{ PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeletePolicy em Referência AWS SDK para .NET da API.

DeleteRole

O código de exemplo a seguir mostra como usar `DeleteRole`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteRole em Referência AWS SDK para .NET da API.

`DeleteRolePolicy`

O código de exemplo a seguir mostra como usar `DeleteRolePolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
```

```
    /// <param name="policyName">The name of the IAM role policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteRolePolicy em Referência AWS SDK para .NET da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para .NET da API.

`DeleteUserPolicy`

O código de exemplo a seguir mostra como usar `DeleteUserPolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteUserPolicy em Referência AWS SDK para .NET da API.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM policy.</
param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DetachRolePolicy em Referência AWS SDK para .NET da API.

`GetAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicy em Referência AWS SDK para .NET da API.

`GetPolicy`

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
```

```
    /// <param name="policyArn">The IAM policy to retrieve information for.</param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
{ PolicyArn = policyArn });
        return response.Policy;
    }
```

- Para obter detalhes da API, consulte GetPolicy em Referência AWS SDK para .NET da API.

GetRole

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }
```

- Para obter detalhes da API, consulte GetRole em Referência AWS SDK para .NET da API.

GetUser

O código de exemplo a seguir mostra como usar `GetUser`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest { UserName
= userName });
        return response.User;
    }
```

- Para obter detalhes da API, consulte GetUser em Referência AWS SDK para .NET da API.

ListAttachedRolePolicies

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>> ListAttachedRolePoliciesAsync(string
roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new ListAttachedRolePoliciesRequest
{ RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }
```

- Para obter detalhes da API, consulte ListAttachedRolePolicies em Referência AWS SDK para .NET da API.

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

- Para obter detalhes da API, consulte ListGroups em Referência AWS SDK para .NET da API.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para .NET da API.

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
```

```
        var listRolePoliciesPaginator = _IAMService.Paginators.ListRolePolicies(new
ListRolePoliciesRequest { RoleName = roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
```

- Para obter detalhes da API, consulte ListRolePolicies em Referência AWS SDK para .NET da API.

`ListRoles`

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
```

```
            roles.AddRange(response.Roles);
        }

        return roles;
    }
```

- Para obter detalhes da API, consulte ListRoles em Referência AWS SDK para .NET da API.

Listar provedores de SAML

O código de exemplo a seguir mostra como usar `ListSAMLProviders`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK para .NET .

ListUsers

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para .NET da API.

`PutRolePolicy`

O código de exemplo a seguir mostra como usar `PutRolePolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string roleName,
 string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte PutRolePolicy em Referência AWS SDK para .NET da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.

- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
```

```
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
```

```
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper = host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several AWS
resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways to
make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");
```

```
        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs 'server_startup_script.sh'
when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the `server.py`
script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
```

```
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2 instances,
each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3, _autoScalerWrapper.GroupName,
zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or continue
with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");
```

```
        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroupN
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.LoadB
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper.Lo
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port, ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for your
 default VPC must\n"
```

```
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management Console.
\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port, ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
 ipString);
                }
            }
            loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
 browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
 endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
 configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
 endpoint:\n");
```

```
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite of
these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
```

```
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as\n" +
                          "healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service fails,
 the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
 "static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
 checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
 _smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in the
 target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
 _autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
```

```
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile that
contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns either
a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo, a
deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not for
Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the Auto
Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load balancer
can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two for
the load balancer to detect the unhealthy");
```

```
        Console.WriteLine("instance. Sending a GET request to the load balancer
 endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
 rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by an
 auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler start
 a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
 instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a successful
 recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the load
 balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance typically
 takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
 instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
 checks mean all instances report as unhealthy.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
 continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
 return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
```

```
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account, we
can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBalanc
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGroup
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupName)
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
```

```
                "Don't forget to delete them when you're done with them or you might
 incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;
```

```
    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
```

```
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                    "\"Version\": \"2012-10-17\"," +
                                    "\"Statement\": [{" +
                                         "\"Effect\": \"Allow\"," +
                                         "\"Principal\": {" +
                                         "\"Service\": [" +
                                             "\"ec2.amazonaws.com\"" +
                                         "]" +
                                         "}," +
                                    "\"Action\": \"sts:AssumeRole\"" +
                                    "}]" +
                                "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
```

```
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
```

```
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
    /// the instance is started. This script installs the Python packages and starts
a Python
    /// web server on the instance.
    /// </summary>
```

```
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability zones.:
{ex.Message}");
            throw;
        }
    }
```

```
    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with size
 {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
```

```
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
```

```
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
```

```
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
```

```
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
```

```
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }
```

```
   /// <summary>
   /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
   /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
   /// used to restart the Python web server.
   /// </summary>
   /// <param name="instanceId">The Id of the instance to update.</param>
   /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
   /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
   /// <returns>Async task.</returns>
   public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
   {
       try
       {
           await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
               new ReplaceIamInstanceProfileAssociationRequest()
               {
                   AssociationId = associationId,
                   IamInstanceProfile = new IamInstanceProfileSpecification()
                   {
                       Name = credsProfileName
                   }
               });
           // Allow time before resetting.
           Thread.Sleep(25000);

           await _amazonEc2.RebootInstancesAsync(
               new RebootInstancesRequest(new List<string>() { instanceId }));
           Thread.Sleep(25000);
           var instanceReady = false;
           var retries = 5;
           while (retries-- > 0 && !instanceReady)
           {
               var instancesPaginator =
                   _amazonSsm.Paginators.DescribeInstanceInformation(
                       new DescribeInstanceInformationRequest());
               // Get the entire list using the paginator.
               await foreach (var instance in
instancesPaginator.InstanceInformationList)
```

```
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }
```

```
    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
```

```
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
```

```
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the calling
 computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
 instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP address
 while running this example.
    /// If you do, be sure to remove public access when you're done.
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
```

```
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be open
to either this computer's IP\n" +
                                         "address, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
               }
               else
               {
                   break;
               }
           }
       }

       return portIsOpen;
   }

   /// <summary>
   /// Add an ingress rule to the specified security group that allows access on
the
   /// specified port from the specified IP address.
   /// </summary>
```

```
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
 targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
```

```
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
```

```
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }
```

```
        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
```

```
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
```

```
   /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
   /// and forwards requests to the specified target group.
   /// </summary>
   /// <param name="name">The name for the new load balancer.</param>
   /// <param name="subnetIds">Subnets for the load balancer.</param>
   /// <param name="targetGroup">Target group for forwarded requests.</param>
   /// <returns>The new LoadBalancer object.</returns>
   public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
   {
       var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
           new CreateLoadBalancerRequest()
           {
               Name = name,
               Subnets = subnetIds
           });
       var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

       // Wait for load balancer to be available.
       var loadBalancerReady = false;
       while (!loadBalancerReady)
       {
           try
           {
               var describeResponse =
                   await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                       new DescribeLoadBalancersRequest()
                       {
                           Names = new List<string>() { name }
                       });

               var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

               loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
           }
           catch (LoadBalancerNotFoundException)
           {
               loadBalancerReady = false;
           }
           Thread.Sleep(10000);
```

```
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
```

```
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }
```

```
    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
 {groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books, movies,
 and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
```

```
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
_amazonDynamoDb.DescribeTableAsync(request);
```

```
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await File.ReadAllTextAsync(recommendationsPath);
        var records =
            JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
```

```
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-architecture-
table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
```

```
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile

- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Exemplos de Amazon Keyspaces usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com Amazon Keyspaces.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Keyspaces

O exemplo de código a seguir mostra como começar a usar o Amazon Keyspaces.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace KeyspacesActions;

public class HelloKeyspaces
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Keyspaces (for Apache Cassandra).
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonKeyspaces>()
                .AddTransient<KeyspacesWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloKeyspaces>();

        var keyspacesClient = host.Services.GetRequiredService<IAmazonKeyspaces>();
        var keyspacesWrapper = new KeyspacesWrapper(keyspacesClient);

        Console.WriteLine("Hello, Amazon Keyspaces! Let's list your keyspaces:");
        await keyspacesWrapper.ListKeyspaces();
    }
}
```

- Para obter detalhes da API, consulte ListKeyspaces em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um keyspace e uma tabela. O esquema da tabela contém dados do filme e tem a recuperação para um ponto no tempo.
- Conectar-se ao keyspace usando uma conexão TLS segura com autenticação SigV4.
- Consultar a tabela. Adicionar, recuperar e atualizar dados do filme.
- Atualizar a tabela. Adicionar uma coluna para rastrear os filmes assistidos.
- Restaure a tabela ao estado anterior e limpe os recursos.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
global using System.Security.Cryptography.X509Certificates;
global using Amazon.Keyspaces;
global using Amazon.Keyspaces.Model;
global using KeyspacesActions;
global using KeyspacesScenario;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using Newtonsoft.Json;


namespace KeyspacesBasics;

/// <summary>
/// Amazon Keyspaces (for Apache Cassandra) scenario. Shows some of the basic
/// actions performed with Amazon Keyspaces.
/// </summary>
public class KeyspacesBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
```

```
            services.AddAWSService<IAmazonKeyspaces>()
            .AddTransient<KeyspacesWrapper>()
            .AddTransient<CassandraWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<KeyspacesBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var keyspacesWrapper = host.Services.GetRequiredService<KeyspacesWrapper>();
        var uiMethods = new UiMethods();

        var keyspaceName = configuration["KeyspaceName"];
        var tableName = configuration["TableName"];

        bool success; // Used to track the results of some operations.

        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        // Create the keyspace.
        var keyspaceArn = await keyspacesWrapper.CreateKeyspace(keyspaceName);

        // Wait for the keyspace to be available. GetKeyspace results in a
        // resource not found error until it is ready for use.
        try
        {
            var getKeyspaceArn = "";
            Console.Write($"Created {keyspaceName}. Waiting for it to become
available. ");
            do
            {
                getKeyspaceArn = await keyspacesWrapper.GetKeyspace(keyspaceName);
                Console.Write(". ");
            } while (getKeyspaceArn != keyspaceArn);
        }
        catch (ResourceNotFoundException)
```

```
        {
            Console.WriteLine("Waiting for keyspace to be created.");
        }

        Console.WriteLine($"\nThe keyspace {keyspaceName} is ready for use.");

        uiMethods.PressEnter();

        // Create the table.
        // First define the schema.
        var allColumns = new List<ColumnDefinition>
        {
            new ColumnDefinition { Name = "title", Type = "text" },
            new ColumnDefinition { Name = "year", Type = "int" },
            new ColumnDefinition { Name = "release_date", Type = "timestamp" },
            new ColumnDefinition { Name = "plot", Type = "text" },
        };

        var partitionKeys = new List<PartitionKey>
        {
            new PartitionKey { Name = "year", },
            new PartitionKey { Name = "title" },
        };

        var tableSchema = new SchemaDefinition
        {
            AllColumns = allColumns,
            PartitionKeys = partitionKeys,
        };

        var tableArn = await keyspacesWrapper.CreateTable(keyspaceName, tableSchema,
tableName);

        // Wait for the table to be active.
        try
        {
            var resp = new GetTableResponse();
            Console.Write("Waiting for the new table to be active. ");
            do
            {
                try
                {
                    resp = await keyspacesWrapper.GetTable(keyspaceName, tableName);
                    Console.Write(".");
```

```
                }
                catch (ResourceNotFoundException)
                {
                    Console.Write(".");
                }
            } while (resp.Status != TableStatus.ACTIVE);

            // Display the table's schema.
            Console.WriteLine($"\nTable {tableName} has been created in
{keyspaceName}");
            Console.WriteLine("Let's take a look at the schema.");
            uiMethods.DisplayTitle("All columns");
            resp.SchemaDefinition.AllColumns.ForEach(column =>
            {
                Console.WriteLine($"{column.Name,-40}\t{column.Type,-20}");
            });

            uiMethods.DisplayTitle("Cluster keys");
            resp.SchemaDefinition.ClusteringKeys.ForEach(clusterKey =>
            {

Console.WriteLine($"{clusterKey.Name,-40}\t{clusterKey.OrderBy,-20}");
            });

            uiMethods.DisplayTitle("Partition keys");
            resp.SchemaDefinition.PartitionKeys.ForEach(partitionKey =>
            {
                Console.WriteLine($"{partitionKey.Name}");
            });

            uiMethods.PressEnter();
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        // Access Apache Cassandra using the Cassandra drive for C#.
        var cassandraWrapper = host.Services.GetRequiredService<CassandraWrapper>();
        var movieFilePath = configuration["MovieFile"];

        Console.WriteLine("Let's add some movies to the table we created.");
        var inserted = await cassandraWrapper.InsertIntoMovieTable(keyspaceName,
tableName, movieFilePath);
```

```
        uiMethods.PressEnter();

        Console.WriteLine("Added the following movies to the table:");
        var rows = await cassandraWrapper.GetMovies(keyspaceName, tableName);
        uiMethods.DisplayTitle("All Movies");

        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            var plot = row.GetValue<string>("plot");
            var release_date = row.GetValue<DateTime>("release_date");
            Console.WriteLine($"{release_date}\t{title}\t{year}\n{plot}");
            Console.WriteLine(uiMethods.SepBar);
        }

        // Update the table schema
        uiMethods.DisplayTitle("Update table schema");
        Console.WriteLine("Now we will update the table to add a boolean field
called watched.");

        // First save the current time as a UTC Date so the original
        // table can be restored later.
        var timeChanged = DateTime.UtcNow;

        // Now update the schema.
        var resourceArn = await keyspacesWrapper.UpdateTable(keyspaceName,
tableName);
        uiMethods.PressEnter();

        Console.WriteLine("Now let's mark some of the movies as watched.");

        // Pick some files to mark as watched.
        var movieToWatch = rows[2].GetValue<string>("title");
        var watchedMovieYear = rows[2].GetValue<int>("year");
        var changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[6].GetValue<string>("title");
        watchedMovieYear = rows[6].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);
```

```
        movieToWatch = rows[9].GetValue<string>("title");
        watchedMovieYear = rows[9].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[10].GetValue<string>("title");
        watchedMovieYear = rows[10].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[13].GetValue<string>("title");
        watchedMovieYear = rows[13].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        uiMethods.DisplayTitle("Watched movies");
        Console.WriteLine("These movies have been marked as watched:");
        rows = await cassandraWrapper.GetWatchedMovies(keyspaceName, tableName);
        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            Console.WriteLine($"{title,-40}\t{year,8}");
        }
        uiMethods.PressEnter();

        Console.WriteLine("We can restore the table to its previous state but that
can take up to 20 minutes to complete.");
        string answer;
        do
        {
            Console.WriteLine("Do you want to restore the table? (y/n)");
            answer = Console.ReadLine();
        } while (answer.ToLower() != "y" && answer.ToLower() != "n");

        if (answer == "y")
        {
            var restoredTableName = $"{tableName}_restored";
            var restoredTableArn = await keyspacesWrapper.RestoreTable(
                keyspaceName,
                tableName,
                restoredTableName,
                timeChanged);
            // Loop and call GetTable until the table is gone. Once it has been
```

```
            // deleted completely, GetTable will raise a ResourceNotFoundException.
            bool wasRestored = false;

            try
            {
                do
                {
                    var resp = await keyspacesWrapper.GetTable(keyspaceName,
restoredTableName);
                    wasRestored = (resp.Status == TableStatus.ACTIVE);
                } while (!wasRestored);
            }
            catch (ResourceNotFoundException)
            {
                // If the restored table raised an error, it isn't
                // ready yet.
                Console.Write(".");
            }
        }

        uiMethods.DisplayTitle("Clean up resources.");

        // Delete the table.
        success = await keyspacesWrapper.DeleteTable(keyspaceName, tableName);

        Console.WriteLine($"Table {tableName} successfully deleted from
{keyspaceName}.");
        Console.WriteLine("Waiting for the table to be removed completely. ");

        // Loop and call GetTable until the table is gone. Once it has been
        // deleted completely, GetTable will raise a ResourceNotFoundException.
        bool wasDeleted = false;

        try
        {
            do
            {
                var resp = await keyspacesWrapper.GetTable(keyspaceName, tableName);
            } while (!wasDeleted);
        }
        catch (ResourceNotFoundException ex)
        {
            wasDeleted = true;
```

```
            Console.WriteLine($"{ex.Message} indicates that the table has been
 deleted.");
        }

        // Delete the keyspace.
        success = await keyspacesWrapper.DeleteKeyspace(keyspaceName);
        Console.WriteLine("The keyspace has been deleted and the demo is now
 complete.");
    }
}
```

```
namespace KeyspacesActions;

/// <summary>
/// Performs Amazon Keyspaces (for Apache Cassandra) actions.
/// </summary>
public class KeyspacesWrapper
{
    private readonly IAmazonKeyspaces _amazonKeyspaces;

    /// <summary>
    /// Constructor for the KeyspaceWrapper.
    /// </summary>
    /// <param name="amazonKeyspaces">An Amazon Keyspaces client object.</param>
    public KeyspacesWrapper(IAmazonKeyspaces amazonKeyspaces)
    {
        _amazonKeyspaces = amazonKeyspaces;
    }

    /// <summary>
    /// Create a new keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name for the new keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
    public async Task<string> CreateKeyspace(string keyspaceName)
    {
        var response =
            await _amazonKeyspaces.CreateKeyspaceAsync(
                new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }
```

```
    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace where the table will be created.</
param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
 schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
 PointInTimeRecoveryStatus.ENABLED }
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }


    /// <summary>
    /// Delete an existing keyspace.
    /// </summary>
    /// <param name="keyspaceName"></param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
            new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
```

```
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get data about a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
    public async Task<string> GetKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.GetKeyspaceAsync(
            new GetKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }


    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }


    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
 ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

 Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }


    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new ListTablesRequest
 { KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

 Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }


    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
```

```
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }


    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
        var newColumn = new ColumnDefinition { Name = "watched", Type = "boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }

}
```

```
using System.Net;
using Cassandra;

namespace KeyspacesScenario;
```

```
/// <summary>
/// Class to perform CRUD methods on an Amazon Keyspaces (for Apache Cassandra)
 database.
///
/// NOTE: This sample uses a plain text authenticator for example purposes only.
/// Recommended best practice is to use a SigV4 authentication plugin, if available.
/// </summary>
public class CassandraWrapper
{
    private readonly IConfiguration _configuration;
    private readonly string _localPathToFile;
    private const string _certLocation = "https://certs.secureserver.net/repository/
sf-class2-root.crt";
    private const string _certFileName = "sf-class2-root.crt";
    private readonly X509Certificate2Collection _certCollection;
    private X509Certificate2 _amazoncert;
    private Cluster _cluster;

    // User name and password for the service.
    private string _userName = null!;
    private string _pwd = null!;

    public CassandraWrapper()
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        _localPathToFile = Path.GetTempPath();

        // Get the Starfield digital certificate and save it locally.
        var client = new WebClient();
        client.DownloadFile(_certLocation, $"{_localPathToFile}/{_certFileName}");

        //var httpClient = new HttpClient();
        //var httpResult = httpClient.Get(fileUrl);
        //using var resultStream = await httpResult.Content.ReadAsStreamAsync();
        //using var fileStream = File.Create(pathToSave);
        //resultStream.CopyTo(fileStream);

        _certCollection = new X509Certificate2Collection();
```

```
        _amazoncert = new X509Certificate2($"{_localPathToFile}/{_certFileName}");

        // Get the user name and password stored in the configuration file.
        _userName = _configuration["UserName"]!;
        _pwd = _configuration["Password"]!;

        // For a list of Service Endpoints for Amazon Keyspaces, see:
        // https://docs.aws.amazon.com/keyspaces/latest/devguide/
programmatic.endpoints.html
        var awsEndpoint = _configuration["ServiceEndpoint"];

        _cluster = Cluster.Builder()
            .AddContactPoints(awsEndpoint)
            .WithPort(9142)
            .WithAuthProvider(new PlainTextAuthProvider(_userName, _pwd))
            .WithSSL(new SSLOptions().SetCertificateCollection(_certCollection))
            .WithQueryOptions(
                new QueryOptions()
                    .SetConsistencyLevel(ConsistencyLevel.LocalQuorum)
                    .SetSerialConsistencyLevel(ConsistencyLevel.LocalSerial))
            .Build();
    }

    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the Apache Cassandra table.
    /// </summary>
    /// <param name="movieFileName">The full path to the JSON file.</param>
    /// <returns>A list of movie objects.</returns>
    public List<Movie> ImportMoviesFromJson(string movieFileName, int numToImport =
 0)
    {
        if (!File.Exists(movieFileName))
        {
            return null!;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();

        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        // If numToImport = 0, return all movies in the collection.
        if (numToImport == 0)
```

```
        {
            // Now return the entire list of movies.
            return allMovies;
        }
        else
        {
            // Now return the first numToImport entries.
            return allMovies.GetRange(0, numToImport);
        }
    }

    /// <summary>
    /// Insert movies into the movie table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="movieTableName">The Amazon Keyspaces table.</param>
    /// <param name="movieFilePath">The path to the resource file containing
    /// movie data to insert into the table.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> InsertIntoMovieTable(string keyspaceName, string
 movieTableName, string movieFilePath, int numToImport = 20)
    {
        // Get some movie data from the movies.json file
        var movies = ImportMoviesFromJson(movieFilePath, numToImport);

        var session = _cluster.Connect(keyspaceName);

        string insertCql;

        RowSet rs;

        // Now we insert the numToImport movies into the table.
        foreach (var movie in movies)
        {
            // Escape single quote characters in the plot.
            insertCql = $"INSERT INTO {keyspaceName}.{movieTableName}
 (title, year, release_date, plot) values($${movie.Title}$$, {movie.Year},
 '{movie.Info.Release_Date.ToString("yyyy-MM-dd")}', $${movie.Info.Plot}$$)";
            rs = await session.ExecuteAsync(new SimpleStatement(insertCql));
        }

        return true;
    }
```

```
    /// <summary>
    /// Gets all of the movies in the movies table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing movie data.</returns>
    public async Task<List<Row>> GetMovies(string keyspaceName, string tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT * FROM
{keyspaceName}.{tableName}"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }

    /// <summary>
    /// Mark a movie in the movie table as watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <param name="title">The title of the movie to mark as watched.</param>
    /// <param name="year">The year the movie was released.</param>
    /// <returns>A set of rows containing the changed data.</returns>
    public async Task<List<Row>> MarkMovieAsWatched(string keyspaceName, string
tableName, string title, int year)
    {
        var session = _cluster.Connect();
        string updateCql = $"UPDATE {keyspaceName}.{tableName} SET watched=true
WHERE title = $${title}$$ AND year = {year};";
        var rs = await session.ExecuteAsync(new SimpleStatement(updateCql));
        var rows = rs.GetRows().ToList();
        return rows;
    }
```

```
    /// <summary>
    /// Retrieve the movies in the movies table where watched is true.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing information about movies
    /// where watched is true.</returns>
    public async Task<List<Row>> GetWatchedMovies(string keyspaceName, string
 tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT title,
 year, plot FROM {keyspaceName}.{tableName} WHERE watched = true ALLOW FILTERING"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable

- ListKeyspaces
- ListTables
- RestoreTable
- UpdateTable

Ações

`CreateKeyspace`

O código de exemplo a seguir mostra como usar `CreateKeyspace`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create a new keyspace.
/// </summary>
/// <param name="keyspaceName">The name for the new keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
public async Task<string> CreateKeyspace(string keyspaceName)
{
    var response =
        await _amazonKeyspaces.CreateKeyspaceAsync(
            new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.ResourceArn;
}
```

- Para obter detalhes da API, consulte CreateKeyspace em Referência AWS SDK para .NET da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace where the table will be created.</
param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
 schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
 PointInTimeRecoveryStatus.ENABLED }
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para .NET da API.

`DeleteKeyspace`

O código de exemplo a seguir mostra como usar `DeleteKeyspace`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete an existing keyspace.
/// </summary>
/// <param name="keyspaceName"></param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyspace(string keyspaceName)
{
    var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
        new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DeleteKeyspace em Referência AWS SDK para .NET da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para .NET da API.

`GetKeyspace`

O código de exemplo a seguir mostra como usar `GetKeyspace`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get data about a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
    public async Task<string> GetKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.GetKeyspaceAsync(
            new GetKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
```

```
    }
```

- Para obter detalhes da API, consulte GetKeyspace em Referência AWS SDK para .NET da API.

GetTable

O código de exemplo a seguir mostra como usar `GetTable`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }
```

- Para obter detalhes da API, consulte GetTable em Referência AWS SDK para .NET da API.

ListKeyspaces

O código de exemplo a seguir mostra como usar `ListKeyspaces`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }
```

- Para obter detalhes da API, consulte ListKeyspaces em Referência AWS SDK para .NET da API.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new ListTablesRequest
{ KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para .NET da API.

`RestoreTable`

O código de exemplo a seguir mostra como usar `RestoreTable`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }
```

- Para obter detalhes da API, consulte RestoreTable em Referência AWS SDK para .NET da API.

`UpdateTable`

O código de exemplo a seguir mostra como usar `UpdateTable`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
        var newColumn = new ColumnDefinition { Name = "watched", Type = "boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }
```

- Para obter detalhes da API, consulte UpdateTable em Referência AWS SDK para .NET da API.

Exemplos de Kinesis usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Kinesis.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Exemplos sem servidor

Ações

`AddTagsToStream`

O código de exemplo a seguir mostra como usar `AddTagsToStream`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to apply key/value pairs to an Amazon Kinesis
/// stream.
/// </summary>
public class TagStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        var tags = new Dictionary<string, string>
        {
            { "Project", "Sample Kinesis Project" },
            { "Application", "Sample Kinesis App" },
        };

        var success = await ApplyTagsToStreamAsync(client, streamName, tags);

        if (success)
        {
            Console.WriteLine($"Taggs successfully added to {streamName}.");
```

```
            }
            else
            {
                Console.WriteLine("Tags were not added to the stream.");
            }
        }

        /// <summary>
        /// Applies the set of tags to the named Kinesis stream.
        /// </summary>
        /// <param name="client">The initialized Kinesis client.</param>
        /// <param name="streamName">The name of the Kinesis stream to which
        /// the tags will be attached.</param>
        /// <param name="tags">A sictionary containing key/value pairs which
        /// will be used to create the Kinesis tags.</param>
        /// <returns>A Boolean value which represents the success or failure
        /// of AddTagsToStreamAsync.</returns>
        public static async Task<bool> ApplyTagsToStreamAsync(
            IAmazonKinesis client,
            string streamName,
            Dictionary<string, string> tags)
        {
            var request = new AddTagsToStreamRequest
            {
                StreamName = streamName,
                Tags = tags,
            };

            var response = await client.AddTagsToStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte AddTagsToStream em Referência AWS SDK para .NET da API.

`CreateStream`

O código de exemplo a seguir mostra como usar `CreateStream`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to create a new Amazon Kinesis stream.
/// </summary>
public class CreateStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        int shardCount = 1;

        var success = await CreateNewStreamAsync(client, streamName,
shardCount);
        if (success)
        {
            Console.WriteLine($"The stream, {streamName} successfully
created.");
        }
    }

    /// <summary>
    /// Creates a new Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client.</param>
    /// <param name="streamName">The name for the new stream.</param>
    /// <param name="shardCount">The number of shards the new stream will
    /// use. The throughput of the stream is a function of the number of
    /// shards; more shards are required for greater provisioned
```

```
        /// throughput.</param>
        /// <returns>A Boolean value indicating whether the stream was created.</
returns>
        public static async Task<bool> CreateNewStreamAsync(IAmazonKinesis client,
 string streamName, int shardCount)
        {
            var request = new CreateStreamRequest
            {
                StreamName = streamName,
                ShardCount = shardCount,
            };

            var response = await client.CreateStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte CreateStream em Referência AWS SDK para .NET da API.

`DeleteStream`

O código de exemplo a seguir mostra como usar `DeleteStream`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
```

```
    /// Shows how to delete an Amazon Kinesis stream.
    /// </summary>
    public class DeleteStream
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            string streamName = "AmazonKinesisStream";

            var success = await DeleteStreamAsync(client, streamName);

            if (success)
            {
                Console.WriteLine($"Stream, {streamName} successfully deleted.");
            }
            else
            {
                Console.WriteLine("Stream not deleted.");
            }
        }

        /// <summary>
        /// Deletes a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamName">The name of the string to delete.</param>
        /// <returns>A Boolean value representing the success of the operation.</
returns>
        public static async Task<bool> DeleteStreamAsync(IAmazonKinesis client,
 string streamName)
        {
            // If EnforceConsumerDeletion is true, any consumers
            // of this stream will also be deleted. If it is set
            // to false and this stream has any consumers, the
            // call will fail with a ResourceInUseException.
            var request = new DeleteStreamRequest
            {
                StreamName = streamName,
                EnforceConsumerDeletion = true,
            };

            var response = await client.DeleteStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
        }
    }
```

- Para obter detalhes da API, consulte DeleteStream em Referência AWS SDK para .NET da API.

`DeregisterStreamConsumer`

O código de exemplo a seguir mostra como usar `DeregisterStreamConsumer`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// Shows how to deregister a consumer from an Amazon Kinesis stream.
    /// </summary>
    public class DeregisterConsumer
    {
        public static async Task Main(string[] args)
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream";
            string consumerName = "CONSUMER_NAME";
            string consumerARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream/consumer/CONSUMER_NAME:000000000000";
```

```
            var success = await DeregisterConsumerAsync(client, streamARN,
 consumerARN, consumerName);

            if (success)
            {
                Console.WriteLine($"{consumerName} successfully deregistered.");
            }
            else
            {
                Console.WriteLine($"{consumerName} was not successfully
 deregistered.");
            }
        }

        /// <summary>
        /// Deregisters a consumer from a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamARN">The ARN of a Kinesis stream.</param>
        /// <param name="consumerARN">The ARN of the consumer.</param>
        /// <param name="consumerName">The name of the consumer.</param>
        /// <returns>A Boolean value representing the success of the operation.</
returns>
        public static async Task<bool> DeregisterConsumerAsync(
            IAmazonKinesis client,
            string streamARN,
            string consumerARN,
            string consumerName)
        {
            var request = new DeregisterStreamConsumerRequest
            {
                StreamARN = streamARN,
                ConsumerARN = consumerARN,
                ConsumerName = consumerName,
            };

            var response = await client.DeregisterStreamConsumerAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte DeregisterStreamConsumer em Referência AWS SDK para .NET da API.

`ListStreamConsumers`

O código de exemplo a seguir mostra como usar `ListStreamConsumers`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// List the consumers of an Amazon Kinesis stream.
    /// </summary>
    public class ListConsumers
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";
            int maxResults = 10;

            var consumers = await ListConsumersAsync(client, streamARN, maxResults);

            if (consumers.Count > 0)
            {
                consumers
                    .ForEach(c => Console.WriteLine($"Name: {c.ConsumerName} ARN:
 {c.ConsumerARN}"));
            }
```

```
            else
            {
                Console.WriteLine("No consumers found.");
            }
        }

        /// <summary>
        /// Retrieve a list of the consumers for a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamARN">The ARN of the stream for which we want to
        /// retrieve a list of clients.</param>
        /// <param name="maxResults">The maximum number of results to return.</
param>
        /// <returns>A list of Consumer objects.</returns>
        public static async Task<List<Consumer>> ListConsumersAsync(IAmazonKinesis
 client, string streamARN, int maxResults)
        {
            var request = new ListStreamConsumersRequest
            {
                StreamARN = streamARN,
                MaxResults = maxResults,
            };

            var response = await client.ListStreamConsumersAsync(request);

            return response.Consumers;
        }
    }
```

- Para obter detalhes da API, consulte ListStreamConsumers em Referência AWS SDK para .NET da API.

`ListStreams`

O código de exemplo a seguir mostra como usar `ListStreams`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Retrieves and displays a list of existing Amazon Kinesis streams.
/// </summary>
public class ListStreams
{
    public static async Task Main(string[] args)
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        var response = await client.ListStreamsAsync(new ListStreamsRequest());

        List<string> streamNames = response.StreamNames;

        if (streamNames.Count > 0)
        {
            streamNames
                .ForEach(s => Console.WriteLine($"Stream name: {s}"));
        }
        else
        {
            Console.WriteLine("No streams were found.");
        }
    }
}
```

- Para obter detalhes da API, consulte ListStreams em Referência AWS SDK para .NET da API.

`ListTagsForStream`

O código de exemplo a seguir mostra como usar `ListTagsForStream`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// Shows how to list the tags that have been attached to an Amazon Kinesis
    /// stream.
    /// </summary>
    public class ListTags
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            string streamName = "AmazonKinesisStream";

            await ListTagsAsync(client, streamName);
        }

        /// <summary>
        /// List the tags attached to a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamName">The name of the Kinesis stream for which you
        /// wish to display tags.</param>
        public static async Task ListTagsAsync(IAmazonKinesis client, string
streamName)
        {
            var request = new ListTagsForStreamRequest
            {
```

```
                StreamName = streamName,
                Limit = 10,
            };

            var response = await client.ListTagsForStreamAsync(request);
            DisplayTags(response.Tags);

            while (response.HasMoreTags)
            {
                request.ExclusiveStartTagKey = response.Tags[response.Tags.Count -
1].Key;
                response = await client.ListTagsForStreamAsync(request);
            }
        }

        /// <summary>
        /// Displays the items in a list of Kinesis tags.
        /// </summary>
        /// <param name="tags">A list of the Tag objects to be displayed.</param>
        public static void DisplayTags(List<Tag> tags)
        {
            tags
                .ForEach(t => Console.WriteLine($"Key: {t.Key} Value: {t.Value}"));
        }
    }
```

- Para obter detalhes da API, consulte ListTagsForStream em Referência AWS SDK para .NET da API.

`RegisterStreamConsumer`

O código de exemplo a seguir mostra como usar `RegisterStreamConsumer`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// This example shows how to register a consumer to an Amazon Kinesis
    /// stream.
    /// </summary>
    public class RegisterConsumer
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            string consumerName = "NEW_CONSUMER_NAME";
            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";

            var consumer = await RegisterConsumerAsync(client, consumerName,
 streamARN);

            if (consumer is not null)
            {
                Console.WriteLine($"{consumer.ConsumerName}");
            }
        }

        /// <summary>
        /// Registers the consumer to a Kinesis stream.
        /// </summary>
        /// <param name="client">The initialized Kinesis client object.</param>
        /// <param name="consumerName">A string representing the consumer.</param>
        /// <param name="streamARN">The ARN of the stream.</param>
        /// <returns>A Consumer object that contains information about the
 consumer.</returns>
        public static async Task<Consumer> RegisterConsumerAsync(IAmazonKinesis
 client, string consumerName, string streamARN)
        {
            var request = new RegisterStreamConsumerRequest
            {
                ConsumerName = consumerName,
                StreamARN = streamARN,
            };
```

```
            var response = await client.RegisterStreamConsumerAsync(request);
            return response.Consumer;
        }
    }
```

- Para obter detalhes da API, consulte RegisterStreamConsumer em Referência AWS SDK para .NET da API.

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali
```

```
namespace KinesisIntegrationSampleCode;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task FunctionHandler(KinesisEvent evnt, ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return;
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                throw;
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
```

```
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using System.Text.Json.Serialization;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace KinesisIntegration;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task<StreamsEventResponse> FunctionHandler(KinesisEvent evnt,
 ILambdaContext context)
    {
```

```
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return new StreamsEventResponse();
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
{record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                /* Since we are working with streams, we can return the failed item
immediately.
                   Lambda will immediately begin to retry processing from this
failed item onwards. */
                return new StreamsEventResponse
                {
                    BatchItemFailures = new
List<StreamsEventResponse.BatchItemFailure>
                    {
                        new StreamsEventResponse.BatchItemFailure { ItemIdentifier =
record.Kinesis.SequenceNumber }
                    }
                };
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
records.");
        return new StreamsEventResponse();
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
```

```
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}

public class StreamsEventResponse
{
    [JsonPropertyName("batchItemFailures")]
    public IList<BatchItemFailure> BatchItemFailures { get; set; }
    public class BatchItemFailure
    {
        [JsonPropertyName("itemIdentifier")]
        public string ItemIdentifier { get; set; }
    }
}
```

AWS KMS exemplos usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with AWS KMS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateAlias`

O código de exemplo a seguir mostra como usar `CreateAlias`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Creates an alias for an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class CreateAlias
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The alias name must start with alias/ and can be
        // up to 256 alphanumeric characters long.
        var aliasName = "alias/ExampleAlias";

        // The value supplied as the TargetKeyId can be either
        // the key ID or key Amazon Resource Name (ARN) of the
        // AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new CreateAliasRequest
        {
            AliasName = aliasName,
            TargetKeyId = keyId,
        };

        var response = await client.CreateAliasAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Alias, {aliasName}, successfully created.");
```

```
            }
            else
            {
                Console.WriteLine($"Could not create alias.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateAlias em Referência AWS SDK para .NET da API.

`CreateGrant`

O código de exemplo a seguir mostra como usar `CreateGrant`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();

            // The identity that is given permission to perform the operations
            // specified in the grant.
            var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

            // The identifier of the AWS KMS key to which the grant applies. You
            // can use the key ID or the Amazon Resource Name (ARN) of the KMS key.
            var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";

            var request = new CreateGrantRequest
            {
                GranteePrincipal = grantee,
                KeyId = keyId,
```

```
                // A list of operations that the grant allows.
                Operations = new List<string>
                {
                    "Encrypt",
                    "Decrypt",
                },
            };

            var response = await client.CreateGrantAsync(request);

            string grantId = response.GrantId; // The unique identifier of the
grant.
            string grantToken = response.GrantToken; // The grant token.

            Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
        }
    }
```

- Para obter detalhes da API, consulte CreateGrant em Referência AWS SDK para .NET da API.

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
```

```
    /// Shows how to create a new AWS Key Management Service (AWS KMS)
    /// key.
    /// </summary>
    public class CreateKey
    {
        public static async Task Main()
        {
            // Note that if you need to create a Key in an AWS Region
            // other than the Region defined for the default user, you need to
            // pass the Region to the client constructor.
            var client = new AmazonKeyManagementServiceClient();

            // The call to CreateKeyAsync will create a symmetrical AWS KMS
            // key. For more information about symmetrical and asymmetrical
            // keys, see:
            //
            // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
            var response = await client.CreateKeyAsync(new CreateKeyRequest());

            // The KeyMetadata object contains information about the new AWS KMS
 key.
            KeyMetadata keyMetadata = response.KeyMetadata;

            if (keyMetadata is not null)
            {
                Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
 created.");
            }
            else
            {
                Console.WriteLine("Could not create KMS Key.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateKey em Referência AWS SDK para .NET da API.

`DescribeKey`

O código de exemplo a seguir mostra como usar `DescribeKey`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Retrieve information about an AWS Key Management Service (AWS KMS) key.
/// You can supply either the key Id or the key Amazon Resource Name (ARN)
/// to the DescribeKeyRequest KeyId property.
/// </summary>
public class DescribeKey
{
    public static async Task Main()
    {
        var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
        var request = new DescribeKeyRequest
        {
            KeyId = keyId,
        };

        var client = new AmazonKeyManagementServiceClient();

        var response = await client.DescribeKeyAsync(request);
        var metadata = response.KeyMetadata;

        Console.WriteLine($"{metadata.KeyId} created on:
{metadata.CreationDate}");
        Console.WriteLine($"State: {metadata.KeyState}");
        Console.WriteLine($"{metadata.Description}");
    }
}
```

- Para obter detalhes da API, consulte DescribeKey em Referência AWS SDK para .NET da API.

DisableKey

O código de exemplo a seguir mostra como usar `DisableKey`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Disable an AWS Key Management Service (AWS KMS) key and then retrieve
/// the key's status to show that it has been disabled.
/// </summary>
public class DisableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new DisableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.DisableKeyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                // Retrieve information about the key to show that it has now
                // been disabled.
                var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- Para obter detalhes da API, consulte DisableKey em Referência AWS SDK para .NET da API.

`EnableKey`

O código de exemplo a seguir mostra como usar `EnableKey`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Enable an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class EnableKey
{
    public static async Task Main()
    {
```

```
            var client = new AmazonKeyManagementServiceClient();

            // The identifier of the AWS KMS key to enable. You can use the
            // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

            var request = new EnableKeyRequest
            {
                KeyId = keyId,
            };

            var response = await client.EnableKeyAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                // Retrieve information about the key to show that it has now
                // been enabled.
                var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- Para obter detalhes da API, consulte EnableKey em Referência AWS SDK para .NET da API.

`ListAliases`

O código de exemplo a seguir mostra como usar `ListAliases`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   using System;
   using System.Threading.Tasks;
   using Amazon.KeyManagementService;
   using Amazon.KeyManagementService.Model;

   /// <summary>
   /// List the AWS Key Management Service (AWS KMS) aliases that have been defined
for
   /// the keys in the same AWS Region as the default user. If you want to list
   /// the aliases in a different Region, pass the Region to the client
   /// constructor.
   /// </summary>
   public class ListAliases
   {
       public static async Task Main()
       {
           var client = new AmazonKeyManagementServiceClient();
           var request = new ListAliasesRequest();
           var response = new ListAliasesResponse();

           do
           {
               response = await client.ListAliasesAsync(request);

               response.Aliases.ForEach(alias =>
               {
                   Console.WriteLine($"Created: {alias.CreationDate} Last Update:
{alias.LastUpdatedDate} Name: {alias.AliasName}");
               });

               request.Marker = response.NextMarker;
           }
           while (response.Truncated);
       }
   }
```

- Para obter detalhes da API, consulte ListAliases em Referência AWS SDK para .NET da API.

`ListGrants`

O código de exemplo a seguir mostra como usar `ListGrants`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// List the AWS Key Management Service (AWS KMS) grants that are associated
with
    /// a specific key.
    /// </summary>
    public class ListGrants
    {
        public static async Task Main()
        {
            // The identifier of the AWS KMS key to disable. You can use the
            // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
            var client = new AmazonKeyManagementServiceClient();
            var request = new ListGrantsRequest
            {
                KeyId = keyId,
            };

            var response = new ListGrantsResponse();

            do
            {
                response = await client.ListGrantsAsync(request);

                response.Grants.ForEach(grant =>
                {
```

```
                    Console.WriteLine($"{grant.GrantId}");
                });

                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- Para obter detalhes da API, consulte ListGrants em Referência AWS SDK para .NET da API.

`ListKeys`

O código de exemplo a seguir mostra como usar `ListKeys`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// List the AWS Key Managements Service (AWS KMS) keys for the AWS Region
/// of the default user. To list keys in another AWS Region, supply the Region
/// as a parameter to the client constructor.
/// </summary>
public class ListKeys
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListKeysRequest();
        var response = new ListKeysResponse();
```

```
            do
            {
                response = await client.ListKeysAsync(request);

                response.Keys.ForEach(key =>
                {
                    Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
                });

                // Set the Marker property when response.Truncated is true
                // in order to get the next keys.
                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- Para obter detalhes da API, consulte ListKeys em Referência AWS SDK para .NET da API.

Exemplos de lambda usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor
- AWS contribuições da comunidade

Conceitos básicos

Olá, Lambda

O exemplo de código a seguir mostra como começar a usar o Lambda.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace LambdaActions;

using Amazon.Lambda;

public class HelloLambda
{
    static async Task Main(string[] args)
    {
        var lambdaClient = new AmazonLambdaClient();

        Console.WriteLine("Hello AWS Lambda");
        Console.WriteLine("Let's get started with AWS Lambda by listing your
 existing Lambda functions:");

        var response = await lambdaClient.ListFunctionsAsync();
        response.Functions.ForEach(function =>
        {
```

```
            Console.WriteLine($"{function.FunctionName}\t{function.Description}");
        });
    }
}
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie métodos que executem ações do Lambda.

```
namespace LambdaActions;

using Amazon.Lambda;
using Amazon.Lambda.Model;
```

```
/// <summary>
/// A class that implements AWS Lambda methods.
/// </summary>
public class LambdaWrapper
{
    private readonly IAmazonLambda _lambdaService;

    /// <summary>
    /// Constructor for the LambdaWrapper class.
    /// </summary>
    /// <param name="lambdaService">An initialized Lambda service client.</param>
    public LambdaWrapper(IAmazonLambda lambdaService)
    {
        _lambdaService = lambdaService;
    }

    /// <summary>
    /// Creates a new Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function.</param>
    /// <param name="s3Bucket">The Amazon Simple Storage Service (Amazon S3)
    /// bucket where the zip file containing the code is located.</param>
    /// <param name="s3Key">The Amazon S3 key of the zip file.</param>
    /// <param name="role">The Amazon Resource Name (ARN) of a role with the
    /// appropriate Lambda permissions.</param>
    /// <param name="handler">The name of the handler function.</param>
    /// <returns>The Amazon Resource Name (ARN) of the newly created
    /// Lambda function.</returns>
    public async Task<string> CreateLambdaFunctionAsync(
        string functionName,
        string s3Bucket,
        string s3Key,
        string role,
        string handler)
    {
        // Defines the location for the function code.
        // S3Bucket - The S3 bucket where the file containing
        //            the source code is stored.
        // S3Key    - The name of the file containing the code.
        var functionCode = new FunctionCode
        {
            S3Bucket = s3Bucket,
            S3Key = s3Key,
        };
```

```
        var createFunctionRequest = new CreateFunctionRequest
        {
            FunctionName = functionName,
            Description = "Created by the Lambda .NET API",
            Code = functionCode,
            Handler = handler,
            Runtime = Runtime.Dotnet6,
            Role = role,
        };

        var reponse = await
_lambdaService.CreateFunctionAsync(createFunctionRequest);
        return reponse.FunctionArn;
    }


    /// <summary>
    /// Delete an AWS Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// delete.</param>
    /// <returns>A Boolean value that indicates the success of the action.</returns>
    public async Task<bool> DeleteFunctionAsync(string functionName)
    {
        var request = new DeleteFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.DeleteFunctionAsync(request);

        // A return value of NoContent means that the request was processed.
        // In this case, the function was deleted, and the return value
        // is intentionally blank.
        return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
    }


    /// <summary>
    /// Gets information about a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function for
    /// which to retrieve information.</param>
```

```
    /// <returns>Async Task.</returns>
    public async Task<FunctionConfiguration> GetFunctionAsync(string functionName)
    {
        var functionRequest = new GetFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.GetFunctionAsync(functionRequest);
        return response.Configuration;
    }


    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// invoke.</param
    /// <param name="parameters">The parameter values that will be passed to the
function.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
        };

        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue = System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }


    /// <summary>
    /// Get a list of Lambda functions.
    /// </summary>
    /// <returns>A list of FunctionConfiguration objects.</returns>
    public async Task<List<FunctionConfiguration>> ListFunctionsAsync()
```

```
    {
        var functionList = new List<FunctionConfiguration>();

        var functionPaginator =
            _lambdaService.Paginators.ListFunctions(new ListFunctionsRequest());
        await foreach (var function in functionPaginator.Functions)
        {
            functionList.Add(function);
        }

        return functionList;
    }


    /// <summary>
    /// Update an existing Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to update.</
param>
    /// <param name="bucketName">The bucket where the zip file containing
    /// the Lambda function code is stored.</param>
    /// <param name="key">The key name of the source code file.</param>
    /// <returns>Async Task.</returns>
    public async Task UpdateFunctionCodeAsync(
        string functionName,
        string bucketName,
        string key)
    {
        var functionCodeRequest = new UpdateFunctionCodeRequest
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await
 _lambdaService.UpdateFunctionCodeAsync(functionCodeRequest);
        Console.WriteLine($"The Function was last modified at
 {response.LastModified}.");
    }


    /// <summary>
```

```
    /// Update the code of a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function to update.</param>
    /// <param name="functionHandler">The code that performs the function's
 actions.</param>
    /// <param name="environmentVariables">A dictionary of environment variables.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateFunctionConfigurationAsync(
        string functionName,
        string functionHandler,
        Dictionary<string, string> environmentVariables)
    {
        var request = new UpdateFunctionConfigurationRequest
        {
            Handler = functionHandler,
            FunctionName = functionName,
            Environment = new Amazon.Lambda.Model.Environment { Variables =
 environmentVariables },
        };

        var response = await
 _lambdaService.UpdateFunctionConfigurationAsync(request);

        Console.WriteLine(response.LastModified);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


}
```

Crie uma função que execute o cenário.

```
global using System.Threading.Tasks;
global using Amazon.IdentityManagement;
global using Amazon.Lambda;
global using LambdaActions;
global using LambdaScenarioCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
```

```
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


using Amazon.Lambda.Model;
using Microsoft.Extensions.Configuration;

namespace LambdaBasics;

public class LambdaBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonLambda>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<LambdaWrapper>()
            .AddTransient<LambdaRoleWrapper>()
            .AddTransient<UIWrapper>()
        )
        .Build();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();


        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<LambdaBasics>();

        var lambdaWrapper = host.Services.GetRequiredService<LambdaWrapper>();
```

```
        var lambdaRoleWrapper =
host.Services.GetRequiredService<LambdaRoleWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        string functionName = configuration["FunctionName"]!;
        string roleName = configuration["RoleName"]!;
        string policyDocument = "{" +
            " \"Version\": \"2012-10-17\"," +
            " \"Statement\": [ " +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Principal\": {" +
            "            \"Service\": \"lambda.amazonaws.com\" " +
            "    }," +
            "        \"Action\": \"sts:AssumeRole\" " +
            "    }" +
            "]" +
        "}";

        var incrementHandler = configuration["IncrementHandler"];
        var calculatorHandler = configuration["CalculatorHandler"];
        var bucketName = configuration["BucketName"];
        var incrementKey = configuration["IncrementKey"];
        var calculatorKey = configuration["CalculatorKey"];
        var policyArn = configuration["PolicyArn"];

        uiWrapper.DisplayLambdaBasicsOverview();

        // Create the policy to use with the AWS Lambda functions and then attach
the
        // policy to a new role.
        var roleArn = await lambdaRoleWrapper.CreateLambdaRoleAsync(roleName,
policyDocument);

        Console.WriteLine("Waiting for role to become active.");
        uiWrapper.WaitABit(15, "Wait until the role is active before trying to use
it.");

        // Attach the appropriate AWS Identity and Access Management (IAM) role
policy to the new role.
        var success = await lambdaRoleWrapper.AttachLambdaRolePolicyAsync(policyArn,
roleName);
        uiWrapper.WaitABit(10, "Allow time for the IAM policy to be attached to the
role.");
```

```
        // Create the Lambda function using a zip file stored in an Amazon Simple
Storage Service
        // (Amazon S3) bucket.
        uiWrapper.DisplayTitle("Create Lambda Function");
        Console.WriteLine($"Creating the AWS Lambda function: {functionName}.");
        var lambdaArn = await lambdaWrapper.CreateLambdaFunctionAsync(
            functionName,
            bucketName,
            incrementKey,
            roleArn,
            incrementHandler);

        Console.WriteLine("Waiting for the new function to be available.");
        Console.WriteLine($"The AWS Lambda ARN is {lambdaArn}");

        // Get the Lambda function.
        Console.WriteLine($"Getting the {functionName} AWS Lambda function.");
        FunctionConfiguration config;
        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.State != State.Active);

        Console.WriteLine($"\nThe function, {functionName} has been created.");
        Console.WriteLine($"The runtime of this Lambda function is
{config.Runtime}.");

        uiWrapper.PressEnter();

        // List the Lambda functions.
        uiWrapper.DisplayTitle("Listing all Lambda functions.");
        var functions = await lambdaWrapper.ListFunctionsAsync();
        DisplayFunctionList(functions);

        uiWrapper.DisplayTitle("Invoke increment function");
        Console.WriteLine("Now that it has been created, invoke the Lambda increment
function.");
        string? value;
        do
        {
            Console.Write("Enter a value to increment: ");
```

```
            value = Console.ReadLine();
        }
        while (string.IsNullOrEmpty(value));

        string functionParameters = "{" +
            "\"action\": \"increment\", " +
            "\"x\": \"" + value + "\"" +
        "}";
        var answer = await lambdaWrapper.InvokeFunctionAsync(functionName,
functionParameters);
        Console.WriteLine($"{value} + 1 = {answer}.");

        uiWrapper.DisplayTitle("Update function");
        Console.WriteLine("Now update the Lambda function code.");
        await lambdaWrapper.UpdateFunctionCodeAsync(functionName, bucketName,
calculatorKey);

        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

        await lambdaWrapper.UpdateFunctionConfigurationAsync(
            functionName,
            calculatorHandler,
            new Dictionary<string, string> { { "LOG_LEVEL", "DEBUG" } });

        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

        uiWrapper.DisplayTitle("Call updated function");
        Console.WriteLine("Now call the updated function...");

        bool done = false;

        do
        {
            string? opSelected;
```

```
            Console.WriteLine("Select the operation to perform:");
            Console.WriteLine("\t1. add");
            Console.WriteLine("\t2. subtract");
            Console.WriteLine("\t3. multiply");
            Console.WriteLine("\t4. divide");
            Console.WriteLine("\tOr enter \"q\" to quit.");
            Console.WriteLine("Enter the number (1, 2, 3, 4, or q) of the operation
you want to perform: ");
            do
            {
                Console.Write("Your choice? ");
                opSelected = Console.ReadLine();
            }
            while (opSelected == string.Empty);

            var operation = (opSelected) switch
            {
                "1" => "add",
                "2" => "subtract",
                "3" => "multiply",
                "4" => "divide",
                "q" => "quit",
                _ => "add",
            };

            if (operation == "quit")
            {
                done = true;
            }
            else
            {
                // Get two numbers and an action from the user.
                value = string.Empty;
                do
                {
                    Console.Write("Enter the first value: ");
                    value = Console.ReadLine();
                }
                while (value == string.Empty);

                string? value2;
                do
                {
```

```
                    Console.Write("Enter a second value: ");
                    value2 = Console.ReadLine();
                }
                while (value2 == string.Empty);

                functionParameters = "{" +
                    "\"action\": \"" + operation + "\", " +
                    "\"x\": \"" + value + "\"," +
                    "\"y\": \"" + value2 + "\"" +
                "}";

                answer = await lambdaWrapper.InvokeFunctionAsync(functionName,
functionParameters);
                Console.WriteLine($"The answer when we {operation} the two numbers
is: {answer}.");
            }

            uiWrapper.PressEnter();
        } while (!done);

        // Delete the function created earlier.

        uiWrapper.DisplayTitle("Clean up resources");
        // Detach the IAM policy from the IAM role.
        Console.WriteLine("First detach the IAM policy from the role.");
        success = await lambdaRoleWrapper.DetachLambdaRolePolicyAsync(policyArn,
roleName);
        uiWrapper.WaitABit(15, "Let's wait for the policy to be fully detached from
the role.");

        Console.WriteLine("Delete the AWS Lambda function.");
        success = await lambdaWrapper.DeleteFunctionAsync(functionName);
        if (success)
        {
            Console.WriteLine($"The {functionName} function was deleted.");
        }
        else
        {
            Console.WriteLine($"Could not remove the function {functionName}");
        }

        // Now delete the IAM role created for use with the functions
        // created by the application.
        Console.WriteLine("Now we can delete the role that we created.");
```

```
        success = await lambdaRoleWrapper.DeleteLambdaRoleAsync(roleName);
        if (success)
        {
            Console.WriteLine("The role has been successfully removed.");
        }
        else
        {
            Console.WriteLine("Couldn't delete the role.");
        }

        Console.WriteLine("The Lambda Scenario is now complete.");
        uiWrapper.PressEnter();

        // Displays a formatted list of existing functions returned by the
        // LambdaMethods.ListFunctions.
        void DisplayFunctionList(List<FunctionConfiguration> functions)
        {
            functions.ForEach(functionConfig =>
            {

 Console.WriteLine($"{functionConfig.FunctionName}\t{functionConfig.Description}");
            });
        }
    }
}


namespace LambdaActions;

using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

public class LambdaRoleWrapper
{
    private readonly IAmazonIdentityManagementService _lambdaRoleService;

    public LambdaRoleWrapper(IAmazonIdentityManagementService lambdaRoleService)
    {
        _lambdaRoleService = lambdaRoleService;
    }

    /// <summary>
    /// Attach an AWS Identity and Access Management (IAM) role policy to the
    /// IAM role to be assumed by the AWS Lambda functions created for the scenario.
```

```
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM policy.</
param>
    /// <param name="roleName">The name of the IAM role to attach the IAM policy
 to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachLambdaRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _lambdaRoleService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest { PolicyArn = policyArn, RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to create.</param>
    /// <param name="policyDocument">The policy document for the new IAM role.</
param>
    /// <returns>A string representing the ARN for newly created role.</returns>
    public async Task<string> CreateLambdaRoleAsync(string roleName, string
 policyDocument)
    {
        var request = new CreateRoleRequest
        {
            AssumeRolePolicyDocument = policyDocument,
            RoleName = roleName,
        };

        var response = await _lambdaRoleService.CreateRoleAsync(request);
        return response.Role.Arn;
    }

    /// <summary>
    /// Deletes an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> DeleteLambdaRoleAsync(string roleName)
    {
        var request = new DeleteRoleRequest
        {
            RoleName = roleName,
```

```
        };

        var response = await _lambdaRoleService.DeleteRoleAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    public async Task<bool> DetachLambdaRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _lambdaRoleService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest { PolicyArn = policyArn, RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}


namespace LambdaScenarioCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the AWS Lambda Basics scenario.
    /// </summary>
    public void DisplayLambdaBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to AWS Lambda Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an AWS Identity and Access Management (IAM)
 role that will be assumed by the functions we create.");
        Console.WriteLine("\t2. Attaches an IAM role policy that has Lambda
 permissions.");
        Console.WriteLine("\t3. Creates a Lambda function that increments the value
 passed to it.");
        Console.WriteLine("\t4. Calls the increment function and passes a value.");
        Console.WriteLine("\t5. Updates the code so that the function is a simple
 calculator.");
        Console.WriteLine("\t6. Calls the calculator function with the values
 entered.");
        Console.WriteLine("\t7. Deletes the Lambda function.");
        Console.WriteLine("\t7. Detaches the IAM role policy.");
```

```
        Console.WriteLine("\t8. Deletes the IAM role.");
        PressEnter();
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
```

```
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

Defina um manipulador do Lambda que aumente um número.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace LambdaIncrement;

public class Function
{

    /// <summary>
    /// A simple function increments the integer parameter.
    /// </summary>
    /// <param name="input">A JSON string containing an action, which must be
    /// "increment" and a string representing the value to increment.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</param>
    /// <returns>A string representing the incremented value of the parameter.</
returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
 context)
    {
        if (input["action"] == "increment")
        {
```

```
            int inputValue = Convert.ToInt32(input["x"]);
            return inputValue + 1;
        }
        else
        {
            return 0;
        }
    }
}
```

Defina um segundo manipulador do Lambda que faça operações aritméticas.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace LambdaCalculator;

public class Function
{

    /// <summary>
    /// A simple function that takes two number in string format and performs
    /// the requested arithmetic function.
    /// </summary>
    /// <param name="input">JSON data containing an action, and x and y values.
    /// Valid actions include: add, subtract, multiply, and divide.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</param>
    /// <returns>A string representing the results of the calculation.</returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
 context)
    {
        var action = input["action"];
        int x = Convert.ToInt32(input["x"]);
        int y = Convert.ToInt32(input["y"]);
        int result;
        switch (action)
```

```
        {
            case "add":
                result = x + y;
                break;
            case "subtract":
                result = x - y;
                break;
            case "multiply":
                result = x * y;
                break;
            case "divide":
                if (y == 0)
                {
                    Console.Error.WriteLine("Divide by zero error.");
                    result = 0;
                }
                else
                    result = x / y;
                break;
            default:
                Console.Error.WriteLine($"{action} is not a valid operation.");
                result = 0;
                break;
        }
        return result;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Creates a new Lambda function.
/// </summary>
/// <param name="functionName">The name of the function.</param>
/// <param name="s3Bucket">The Amazon Simple Storage Service (Amazon S3)
/// bucket where the zip file containing the code is located.</param>
/// <param name="s3Key">The Amazon S3 key of the zip file.</param>
/// <param name="role">The Amazon Resource Name (ARN) of a role with the
/// appropriate Lambda permissions.</param>
/// <param name="handler">The name of the handler function.</param>
/// <returns>The Amazon Resource Name (ARN) of the newly created
/// Lambda function.</returns>
public async Task<string> CreateLambdaFunctionAsync(
    string functionName,
    string s3Bucket,
    string s3Key,
    string role,
    string handler)
{
    // Defines the location for the function code.
    // S3Bucket - The S3 bucket where the file containing
    //            the source code is stored.
    // S3Key    - The name of the file containing the code.
    var functionCode = new FunctionCode
    {
        S3Bucket = s3Bucket,
        S3Key = s3Key,
    };
```

```
        var createFunctionRequest = new CreateFunctionRequest
        {
            FunctionName = functionName,
            Description = "Created by the Lambda .NET API",
            Code = functionCode,
            Handler = handler,
            Runtime = Runtime.Dotnet6,
            Role = role,
        };

        var reponse = await
_lambdaService.CreateFunctionAsync(createFunctionRequest);
        return reponse.FunctionArn;
    }
```

- Para obter detalhes da API, consulte CreateFunction em Referência AWS SDK para .NET da API.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an AWS Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// delete.</param>
    /// <returns>A Boolean value that indicates the success of the action.</returns>
    public async Task<bool> DeleteFunctionAsync(string functionName)
    {
```

```
        var request = new DeleteFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.DeleteFunctionAsync(request);

        // A return value of NoContent means that the request was processed.
        // In this case, the function was deleted, and the return value
        // is intentionally blank.
        return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
    }
```

- Para obter detalhes da API, consulte DeleteFunction em Referência AWS SDK para .NET da API.

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets information about a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function for
    /// which to retrieve information.</param>
    /// <returns>Async Task.</returns>
    public async Task<FunctionConfiguration> GetFunctionAsync(string functionName)
    {
        var functionRequest = new GetFunctionRequest
        {
            FunctionName = functionName,
```

```
        };

        var response = await _lambdaService.GetFunctionAsync(functionRequest);
        return response.Configuration;
    }
```

- Para obter detalhes da API, consulte GetFunction em Referência AWS SDK para .NET da API.

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// invoke.</param
    /// <param name="parameters">The parameter values that will be passed to the
function.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
        };
```

```
        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue = System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para .NET .

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of Lambda functions.
    /// </summary>
    /// <returns>A list of FunctionConfiguration objects.</returns>
    public async Task<List<FunctionConfiguration>> ListFunctionsAsync()
    {
        var functionList = new List<FunctionConfiguration>();

        var functionPaginator =
            _lambdaService.Paginators.ListFunctions(new ListFunctionsRequest());
        await foreach (var function in functionPaginator.Functions)
        {
            functionList.Add(function);
        }

        return functionList;
    }
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para .NET da API.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update an existing Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to update.</
param>
    /// <param name="bucketName">The bucket where the zip file containing
    /// the Lambda function code is stored.</param>
    /// <param name="key">The key name of the source code file.</param>
    /// <returns>Async Task.</returns>
    public async Task UpdateFunctionCodeAsync(
        string functionName,
        string bucketName,
        string key)
    {
        var functionCodeRequest = new UpdateFunctionCodeRequest
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await
 _lambdaService.UpdateFunctionCodeAsync(functionCodeRequest);
```

```
        Console.WriteLine($"The Function was last modified at
 {response.LastModified}.");
    }
```

- Para obter detalhes da API, consulte UpdateFunctionCode em Referência AWS SDK para .NET da API.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update the code of a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function to update.</param>
    /// <param name="functionHandler">The code that performs the function's
 actions.</param>
    /// <param name="environmentVariables">A dictionary of environment variables.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateFunctionConfigurationAsync(
        string functionName,
        string functionHandler,
        Dictionary<string, string> environmentVariables)
    {
        var request = new UpdateFunctionConfigurationRequest
        {
            Handler = functionHandler,
            FunctionName = functionName,
            Environment = new Amazon.Lambda.Model.Environment { Variables =
 environmentVariables },
```

```
        };

        var response = await
_lambdaService.UpdateFunctionConfigurationAsync(request);

        Console.WriteLine(response.LastModified);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte UpdateFunctionConfiguration em Referência AWS SDK para .NET da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Como transformar dados com o S3 Object Lambda

O exemplo de código a seguir mostra como transformar dados para sua aplicação com o S3 Object Lambda.

SDK para .NET

Mostra como adicionar código personalizado a solicitações GET padrão do S3 para modificar o objeto solicitado e recuperado do S3 e possibilitar que o objeto atenda às necessidades do cliente ou aplicação solicitante.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Lambda
- Amazon S3

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando .NET.

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace aws_rds;

public class InputModel
{
    public string key1 { get; set; }
```

```
    public string key2 { get; set; }
}

public class Function
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
 FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
 $"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

 $"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +

 $"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                 $"Pwd={authToken};";



        try
        {
            await using var connection = new MySqlConnection(connectionString);
            await connection.OpenAsync();

            const string sql = "SELECT @param1 + @param2 AS Sum";
```

```
            await using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
 "0"));
            command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
 "0"));

            await using var reader = await command.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                int result = reader.GetInt32("Sum");

                //Sample Response: {"statusCode":200,"body":"{\"message\":\"The sum
 is: 45\"}","isBase64Encoded":false}
                return new APIGatewayProxyResponse
                {
                    StatusCode = 200,
                    Body = JsonSerializer.Serialize(new { message = $"The sum is:
 {result}" })
                };
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = 500,
            Body = JsonSerializer.Serialize(new { error = "Internal server error" })
        };
    }
}
```

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace KinesisIntegrationSampleCode;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task FunctionHandler(KinesisEvent evnt, ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return;
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
```

```
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                throw;
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}
```

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace AWSLambda_DDB;

public class Function
{
    public void FunctionHandler(DynamoDBEvent dynamoEvent, ILambdaContext context)
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");

        foreach (var record in dynamoEvent.Records)
        {
            context.Logger.LogInformation($"Event ID: {record.EventID}");
            context.Logger.LogInformation($"Event Name: {record.EventName}");

            context.Logger.LogInformation(JsonSerializer.Serialize(record));
        }

        context.Logger.LogInformation("Stream processing complete.");
    }
}
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Processando um evento do Amazon DocumentDB com o Lambda ao usar .NET.

```
using Amazon.Lambda.Core;
using System.Text.Json;
using System;
using System.Collections.Generic;
using System.Text.Json.Serialization;
//Assembly attribute to enable the Lambda function's JSON input to be converted into
 a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace LambdaDocDb;

public class Function
{

     /// <summary>
    /// Lambda function entry point to process Amazon DocumentDB events.
    /// </summary>
    /// <param name="event">The Amazon DocumentDB event.</param>
    /// <param name="context">The Lambda context object.</param>
    /// <returns>A string to indicate successful processing.</returns>
    public string FunctionHandler(Event evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Events)
        {
            ProcessDocumentDBEvent(record, context);
        }

        return "OK";
    }
```

```
    private void ProcessDocumentDBEvent(DocumentDBEventRecord record,
ILambdaContext context)
   {

       var eventData = record.Event;
       var operationType = eventData.OperationType;
       var databaseName = eventData.Ns.Db;
       var collectionName = eventData.Ns.Coll;
       var fullDocument = JsonSerializer.Serialize(eventData.FullDocument, new
JsonSerializerOptions { WriteIndented = true });

       context.Logger.LogLine($"Operation type: {operationType}");
       context.Logger.LogLine($"Database: {databaseName}");
       context.Logger.LogLine($"Collection: {collectionName}");
       context.Logger.LogLine($"Full document:\n{fullDocument}");
   }



   public class Event
   {
       [JsonPropertyName("eventSourceArn")]
       public string EventSourceArn { get; set; }

       [JsonPropertyName("events")]
       public List<DocumentDBEventRecord> Events { get; set; }

       [JsonPropertyName("eventSource")]
       public string EventSource { get; set; }
   }

   public class DocumentDBEventRecord
   {
       [JsonPropertyName("event")]
       public EventData Event { get; set; }
   }

   public class EventData
   {
       [JsonPropertyName("_id")]
       public IdData Id { get; set; }

       [JsonPropertyName("clusterTime")]
       public ClusterTime ClusterTime { get; set; }
```

```
        [JsonPropertyName("documentKey")]
        public DocumentKey DocumentKey { get; set; }

        [JsonPropertyName("fullDocument")]
        public Dictionary<string, object> FullDocument { get; set; }

        [JsonPropertyName("ns")]
        public Namespace Ns { get; set; }

        [JsonPropertyName("operationType")]
        public string OperationType { get; set; }
    }

    public class IdData
    {
        [JsonPropertyName("_data")]
        public string Data { get; set; }
    }

    public class ClusterTime
    {
        [JsonPropertyName("$timestamp")]
        public Timestamp Timestamp { get; set; }
    }

    public class Timestamp
    {
        [JsonPropertyName("t")]
        public long T { get; set; }

        [JsonPropertyName("i")]
        public int I { get; set; }
    }

    public class DocumentKey
    {
        [JsonPropertyName("_id")]
        public Id Id { get; set; }
    }

    public class Id
    {
        [JsonPropertyName("$oid")]
```

```
        public string Oid { get; set; }
    }

    public class Namespace
    {
        [JsonPropertyName("db")]
        public string Db { get; set; }

        [JsonPropertyName("coll")]
        public string Coll { get; set; }
    }
}
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando .NET.

```
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KafkaEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace MSKLambda;
```

```
public class Function
{


    /// <param name="input">The event for the Lambda function handler to process.</
param>
    /// <param name="context">The ILambdaContext that provides methods for logging
 and describing the Lambda environment.</param>
    /// <returns></returns>
    public void FunctionHandler(KafkaEvent evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Records)
        {
            Console.WriteLine("Key:" + record.Key);
            foreach (var eventRecord in record.Value)
            {
                var valueBytes = eventRecord.Value.ToArray();
                var valueText = Encoding.UTF8.GetString(valueBytes);

                Console.WriteLine("Message:" + valueText);
            }
        }
    }


}
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
```

```
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request - {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record, ILambdaContext
 context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }

        context.Logger.LogInformation("done");
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
```

```
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }

    }
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using System.Text.Json.Serialization;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
```

```
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace KinesisIntegration;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task<StreamsEventResponse> FunctionHandler(KinesisEvent evnt,
 ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return new StreamsEventResponse();
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                return new StreamsEventResponse
                {
                    BatchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>
```

```
                    {
                        new StreamsEventResponse.BatchItemFailure { ItemIdentifier =
 record.Kinesis.SequenceNumber }
                    }
                };
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
        return new StreamsEventResponse();
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}

public class StreamsEventResponse
{
    [JsonPropertyName("batchItemFailures")]
    public IList<BatchItemFailure> BatchItemFailures { get; set; }
    public class BatchItemFailure
    {
        [JsonPropertyName("itemIdentifier")]
        public string ItemIdentifier { get; set; }
    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace AWSLambda_DDB;

public class Function
{
    public StreamsEventResponse FunctionHandler(DynamoDBEvent dynamoEvent,
 ILambdaContext context)

    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>();
        StreamsEventResponse streamsEventResponse = new StreamsEventResponse();

        foreach (var record in dynamoEvent.Records)
        {
            try
            {
                var sequenceNumber = record.Dynamodb.SequenceNumber;
                context.Logger.LogInformation(sequenceNumber);
            }
```

```
            catch (Exception ex)
            {
                context.Logger.LogError(ex.Message);
                batchItemFailures.Add(new StreamsEventResponse.BatchItemFailure()
 { ItemIdentifier = record.Dynamodb.SequenceNumber });
            }
        }

        if (batchItemFailures.Count > 0)
        {
            streamsEventResponse.BatchItemFailures = batchItemFailures;
        }

        context.Logger.LogInformation("Stream processing complete.");
        return streamsEventResponse;
    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
```

```
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali
namespace sqsSample;

public class Function
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
 ILambdaContext context)
    {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
 SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
        context.Logger.LogInformation($"Processed message {message.Body}");
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para .NET

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK .NET.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

MediaConvert exemplos usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with MediaConvert.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Olá MediaConvert

O exemplo de código a seguir mostra como começar a usar o AWS Elemental MediaConvert.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.MediaConvert;
using Amazon.MediaConvert.Model;

namespace MediaConvertActions;

public static class HelloMediaConvert
{
    static async Task Main(string[] args)
    {
        // Create the client using the default profile.
        var mediaConvertClient = new AmazonMediaConvertClient();

        Console.WriteLine($"Hello AWS Elemental MediaConvert! Your MediaConvert Jobs
 are:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get some MediaConvert jobs.
        var response = await mediaConvertClient.ListJobsAsync(
            new ListJobsRequest()
            {
                MaxResults = 10
            }
            );

        foreach (var job in response.Jobs)
        {
            Console.WriteLine($"\tJob: {job.Id} status {job.Status}");
```

```
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeEndpoints em Referência AWS SDK para .NET da API.

Ações

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o wrapper, o cliente e os locais do arquivo.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-
in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Creating job for input file {fileInput}.");
        var jobId = await wrapper.CreateJob(mediaConvertRole!, fileInput!,
 fileOutput!);
        Console.WriteLine($"Created job with Job ID: {jobId}");
        Console.WriteLine(new string('-', 80));
```

Criar o trabalho usando o método wrapper e retorne o ID do trabalho.

```
    /// <summary>
    /// Create a job to convert a media file.
    /// </summary>
    /// <param name="mediaConvertRole">The Amazon Resource Name (ARN) of the media
 convert role, as specified here:
    /// https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-in-
mediaconvert-configured.html</param>
    /// <param name="fileInput">The Amazon Simple Storage Service (Amazon S3)
 location of the input media file.</param>
    /// <param name="fileOutput">The Amazon S3 location for the output media file.</
param>
    /// <returns>The ID of the new job.</returns>
    public async Task<string> CreateJob(string mediaConvertRole, string fileInput,
        string fileOutput)
    {
        CreateJobRequest createJobRequest = new CreateJobRequest
        {
            Role = mediaConvertRole
        };

        createJobRequest.UserMetadata.Add("Customer", "Amazon");

        JobSettings jobSettings = new JobSettings
        {
            AdAvailOffset = 0,
            TimecodeConfig = new TimecodeConfig
            {
                Source = TimecodeSource.EMBEDDED
            }
        };
        createJobRequest.Settings = jobSettings;
```

```
        #region OutputGroup

        OutputGroup ofg = new OutputGroup
        {
            Name = "File Group",
            OutputGroupSettings = new OutputGroupSettings
            {
                Type = OutputGroupType.FILE_GROUP_SETTINGS,
                FileGroupSettings = new FileGroupSettings
                {
                    Destination = fileOutput
                }
            }
        };

        Output output = new Output
        {
            NameModifier = "_1"
        };

        #region VideoDescription

        VideoDescription vdes = new VideoDescription
        {
            ScalingBehavior = ScalingBehavior.DEFAULT,
            TimecodeInsertion = VideoTimecodeInsertion.DISABLED,
            AntiAlias = AntiAlias.ENABLED,
            Sharpness = 50,
            AfdSignaling = AfdSignaling.NONE,
            DropFrameTimecode = DropFrameTimecode.ENABLED,
            RespondToAfd = RespondToAfd.NONE,
            ColorMetadata = ColorMetadata.INSERT,
            CodecSettings = new VideoCodecSettings
            {
                Codec = VideoCodec.H_264
            }
        };
        output.VideoDescription = vdes;

        H264Settings h264 = new H264Settings
        {
            InterlaceMode = H264InterlaceMode.PROGRESSIVE,
            NumberReferenceFrames = 3,
            Syntax = H264Syntax.DEFAULT,
```

```
            Softness = 0,
            GopClosedCadence = 1,
            GopSize = 90,
            Slices = 1,
            GopBReference = H264GopBReference.DISABLED,
            SlowPal = H264SlowPal.DISABLED,
            SpatialAdaptiveQuantization = H264SpatialAdaptiveQuantization.ENABLED,
            TemporalAdaptiveQuantization = H264TemporalAdaptiveQuantization.ENABLED,
            FlickerAdaptiveQuantization = H264FlickerAdaptiveQuantization.DISABLED,
            EntropyEncoding = H264EntropyEncoding.CABAC,
            Bitrate = 5000000,
            FramerateControl = H264FramerateControl.SPECIFIED,
            RateControlMode = H264RateControlMode.CBR,
            CodecProfile = H264CodecProfile.MAIN,
            Telecine = H264Telecine.NONE,
            MinIInterval = 0,
            AdaptiveQuantization = H264AdaptiveQuantization.HIGH,
            CodecLevel = H264CodecLevel.AUTO,
            FieldEncoding = H264FieldEncoding.PAFF,
            SceneChangeDetect = H264SceneChangeDetect.ENABLED,
            QualityTuningLevel = H264QualityTuningLevel.SINGLE_PASS,
            FramerateConversionAlgorithm =
                H264FramerateConversionAlgorithm.DUPLICATE_DROP,
            UnregisteredSeiTimecode = H264UnregisteredSeiTimecode.DISABLED,
            GopSizeUnits = H264GopSizeUnits.FRAMES,
            ParControl = H264ParControl.SPECIFIED,
            NumberBFramesBetweenReferenceFrames = 2,
            RepeatPps = H264RepeatPps.DISABLED,
            FramerateNumerator = 30,
            FramerateDenominator = 1,
            ParNumerator = 1,
            ParDenominator = 1
        };
        output.VideoDescription.CodecSettings.H264Settings = h264;

        #endregion VideoDescription

        #region AudioDescription

        AudioDescription ades = new AudioDescription
        {
            LanguageCodeControl = AudioLanguageCodeControl.FOLLOW_INPUT,
            // This name matches one specified in the following Inputs.
            AudioSourceName = "Audio Selector 1",
```

```
            CodecSettings = new AudioCodecSettings
            {
                Codec = AudioCodec.AAC
            }
        };

        AacSettings aac = new AacSettings
        {
            AudioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.NORMAL,
            RateControlMode = AacRateControlMode.CBR,
            CodecProfile = AacCodecProfile.LC,
            CodingMode = AacCodingMode.CODING_MODE_2_0,
            RawFormat = AacRawFormat.NONE,
            SampleRate = 48000,
            Specification = AacSpecification.MPEG4,
            Bitrate = 64000
        };
        ades.CodecSettings.AacSettings = aac;
        output.AudioDescriptions.Add(ades);

        #endregion AudioDescription

        #region Mp4 Container

        output.ContainerSettings = new ContainerSettings
        {
            Container = ContainerType.MP4
        };
        Mp4Settings mp4 = new Mp4Settings
        {
            CslgAtom = Mp4CslgAtom.INCLUDE,
            FreeSpaceBox = Mp4FreeSpaceBox.EXCLUDE,
            MoovPlacement = Mp4MoovPlacement.PROGRESSIVE_DOWNLOAD
        };
        output.ContainerSettings.Mp4Settings = mp4;

        #endregion Mp4 Container

        ofg.Outputs.Add(output);
        createJobRequest.Settings.OutputGroups.Add(ofg);

        #endregion OutputGroup
```

```
        #region Input

        Input input = new Input
        {
            FilterEnable = InputFilterEnable.AUTO,
            PsiControl = InputPsiControl.USE_PSI,
            FilterStrength = 0,
            DeblockFilter = InputDeblockFilter.DISABLED,
            DenoiseFilter = InputDenoiseFilter.DISABLED,
            TimecodeSource = InputTimecodeSource.EMBEDDED,
            FileInput = fileInput
        };

        AudioSelector audsel = new AudioSelector
        {
            Offset = 0,
            DefaultSelection = AudioDefaultSelection.NOT_DEFAULT,
            ProgramSelection = 1,
            SelectorType = AudioSelectorType.TRACK
        };
        audsel.Tracks.Add(1);
        input.AudioSelectors.Add("Audio Selector 1", audsel);

        input.VideoSelector = new VideoSelector
        {
            ColorSpace = ColorSpace.FOLLOW
        };

        createJobRequest.Settings.Inputs.Add(input);

        #endregion Input

        CreateJobResponse createJobResponse =
            await _amazonMediaConvert.CreateJobAsync(createJobRequest);

        var jobId = createJobResponse.Job.Id;

        return jobId;
    }
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK para .NET da API.

`GetJob`

O código de exemplo a seguir mostra como usar `GetJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o wrapper, o cliente e os locais do arquivo.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-
in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

Obter um trabalho pelo ID.

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Getting job information for Job ID {jobId}");
        var job = await wrapper.GetJobById(jobId);
        Console.WriteLine($"Job {job.Id} created on {job.CreatedAt:d} has status
 {job.Status}.");
        Console.WriteLine(new string('-', 80));
```

```
    /// <summary>
```

```
    /// Get the job information for a job by its ID.
    /// </summary>
    /// <param name="jobId">The ID of the job.</param>
    /// <returns>The Job object.</returns>
    public async Task<Job> GetJobById(string jobId)
    {
        var jobResponse = await _amazonMediaConvert.GetJobAsync(
                new GetJobRequest
                {
                    Id = jobId
                });

        return jobResponse.Job;
    }
```

- Para obter detalhes da API, consulte GetJob em Referência AWS SDK para .NET da API.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o wrapper, o cliente e os locais do arquivo.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-
in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
```

```
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

Liste os trabalhos usando um status específico.

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Listing all complete jobs.");
        var completeJobs = await wrapper.ListAllJobsByStatus(JobStatus.COMPLETE);
        completeJobs.ForEach(j =>
        {
            Console.WriteLine($"Job {j.Id} created on {j.CreatedAt:d} has status
{j.Status}.");
        });
```

Liste os trabalhos usando um paginador.

```
    /// <summary>
    /// List all of the jobs with a particular status using a paginator.
    /// </summary>
    /// <param name="status">The status to use when listing jobs.</param>
    /// <returns>The list of jobs matching the status.</returns>
    public async Task<List<Job>> ListAllJobsByStatus(JobStatus? status = null)
    {
        var returnedJobs = new List<Job>();

        var paginatedJobs = _amazonMediaConvert.Paginators.ListJobs(
                new ListJobsRequest
                {
                    Status = status
                });

        // Get the entire list using the paginator.
        await foreach (var job in paginatedJobs.Jobs)
        {
            returnedJobs.Add(job);
        }

        return returnedJobs;
```

```
    }
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para .NET da API.

Exemplos de uso do Amazon MSK SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon MSK.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando .NET.

```
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KafkaEvents;
```

```
// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace MSKLambda;

public class Function
{


    /// <param name="input">The event for the Lambda function handler to process.</
param>
    /// <param name="context">The ILambdaContext that provides methods for logging
 and describing the Lambda environment.</param>
    /// <returns></returns>
    public void FunctionHandler(KafkaEvent evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Records)
        {
            Console.WriteLine("Key:" + record.Key);
            foreach (var eventRecord in record.Value)
            {
                var valueBytes = eventRecord.Value.ToArray();
                var valueText = Encoding.UTF8.GetString(valueBytes);

                Console.WriteLine("Message:" + valueText);
            }
        }
    }


}
```

Exemplos de organizações usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with Organizations.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AttachPolicy`

O código de exemplo a seguir mostra como usar `AttachPolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to attach an AWS Organizations policy to an organization,
/// an organizational unit, or an account.
/// </summary>
public class AttachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then calls the
    /// AttachPolicyAsync method to attach the policy to the root
    /// organization.
    /// </summary>
```

```
        public static async Task Main()
        {
            IAmazonOrganizations client = new AmazonOrganizationsClient();
            var policyId = "p-00000000";
            var targetId = "r-0000";

            var request = new AttachPolicyRequest
            {
                PolicyId = policyId,
                TargetId = targetId,
            };

            var response = await client.AttachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully attached Policy ID {policyId} to
Target ID: {targetId}.");
            }
            else
            {
                Console.WriteLine("Was not successful in attaching the policy.");
            }
        }
    }
```

- Para obter detalhes da API, consulte AttachPolicy em Referência AWS SDK para .NET da API.

`CreateAccount`

O código de exemplo a seguir mostra como usar `CreateAccount`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;

    /// <summary>
    /// Creates a new AWS Organizations account.
    /// </summary>
    public class CreateAccount
    {
        /// <summary>
        /// Initializes an Organizations client object and uses it to create
        /// the new account with the name specified in accountName.
        /// </summary>
        public static async Task Main()
        {
            IAmazonOrganizations client = new AmazonOrganizationsClient();
            var accountName = "ExampleAccount";
            var email = "someone@example.com";

            var request = new CreateAccountRequest
            {
                AccountName = accountName,
                Email = email,
            };

            var response = await client.CreateAccountAsync(request);
            var status = response.CreateAccountStatus;

            Console.WriteLine($"The staus of {status.AccountName} is
{status.State}.");
        }
    }
```

- Para obter detalhes da API, consulte CreateAccount em Referência AWS SDK para .NET da API.

`CreateOrganization`

O código de exemplo a seguir mostra como usar `CreateOrganization`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates an organization in AWS Organizations.
/// </summary>
public class CreateOrganization
{
    /// <summary>
    /// Creates an Organizations client object and then uses it to create
    /// a new organization with the default user as the administrator, and
    /// then displays information about the new organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.CreateOrganizationAsync(new
CreateOrganizationRequest
        {
            FeatureSet = "ALL",
        });

        Organization newOrg = response.Organization;

        Console.WriteLine($"Organization: {newOrg.Id} Main Accoount:
{newOrg.MasterAccountId}");
    }
}
```

- Para obter detalhes da API, consulte CreateOrganization em Referência AWS SDK para .NET da API.

`CreateOrganizationalUnit`

O código de exemplo a seguir mostra como usar `CreateOrganizationalUnit`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new organizational unit in AWS Organizations.
/// </summary>
public class CreateOrganizationalUnit
{
    /// <summary>
    /// Initializes an Organizations client object and then uses it to call
    /// the CreateOrganizationalUnit method. If the call succeeds, it
    /// displays information about the new organizational unit.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitName = "ProductDevelopmentUnit";

        var request = new CreateOrganizationalUnitRequest
        {
            Name = orgUnitName,
            ParentId = "r-0000",
        };
```

```
            var response = await client.CreateOrganizationalUnitAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully created organizational unit:
{orgUnitName}.");
                Console.WriteLine($"Organizational unit {orgUnitName} Details");
                Console.WriteLine($"ARN: {response.OrganizationalUnit.Arn} Id:
{response.OrganizationalUnit.Id}");
            }
            else
            {
                Console.WriteLine("Could not create new organizational unit.");
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateOrganizationalUnit em Referência AWS SDK para .NET da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;

    /// <summary>
```

```
    /// Creates a new AWS Organizations Policy.
    /// </summary>
    public class CreatePolicy
    {
        /// <summary>
        /// Initializes the AWS Organizations client object, uses it to
        /// create a new Organizations Policy, and then displays information
        /// about the newly created Policy.
        /// </summary>
        public static async Task Main()
        {
            IAmazonOrganizations client = new AmazonOrganizationsClient();
            var policyContent = "{" +
                "   \"Version\": \"2012-10-17\"," +
                " \"Statement\" : [{" +
                    " \"Action\" : [\"s3:*\"]," +
                    " \"Effect\" : \"Allow\"," +
                    " \"Resource\" : \"*\"" +
                "}]" +
            "}";

            try
            {
                var response = await client.CreatePolicyAsync(new
CreatePolicyRequest
                {
                    Content = policyContent,
                    Description = "Enables admins of attached accounts to delegate
all Amazon S3 permissions",
                    Name = "AllowAllS3Actions",
                    Type = "SERVICE_CONTROL_POLICY",
                });

                Policy policy = response.Policy;
                Console.WriteLine($"{policy.PolicySummary.Name} has the following
content: {policy.Content}");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK para .NET da API.

`DeleteOrganization`

O código de exemplo a seguir mostra como usar `DeleteOrganization`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing organization using the AWS
/// Organizations Service.
/// </summary>
public class DeleteOrganization
{
    /// <summary>
    /// Initializes the Organizations client and then calls
    /// DeleteOrganizationAsync to delete the organization.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.DeleteOrganizationAsync(new
DeleteOrganizationRequest());

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                Console.WriteLine("Successfully deleted organization.");
            }
            else
            {
                Console.WriteLine("Could not delete organization.");
            }
        }
    }
```

- Para obter detalhes da API, consulte DeleteOrganization em Referência AWS SDK para .NET da API.

`DeleteOrganizationalUnit`

O código de exemplo a seguir mostra como usar `DeleteOrganizationalUnit`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;

    /// <summary>
    /// Shows how to delete an existing AWS Organizations organizational unit.
    /// </summary>
    public class DeleteOrganizationalUnit
    {
        /// <summary>
        /// Initializes the Organizations client object and calls
        /// DeleteOrganizationalUnitAsync to delete the organizational unit
        /// with the selected ID.
```

```
        /// </summary>
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var orgUnitId = "ou-0000-00000000";

            var request = new DeleteOrganizationalUnitRequest
            {
                OrganizationalUnitId = orgUnitId,
            };

            var response = await client.DeleteOrganizationalUnitAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted the organizational unit
with ID: {orgUnitId}.");
            }
            else
            {
                Console.WriteLine($"Could not delete the organizational unit with
ID: {orgUnitId}.");
            }
        }
    }
```

- Para obter detalhes da API, consulte DeleteOrganizationalUnit em Referência AWS SDK para .NET da API.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Deletes an existing AWS Organizations policy.
/// </summary>
public class DeletePolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// delete the policy with the specified policyId.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";

        var request = new DeletePolicyRequest
        {
            PolicyId = policyId,
        };

        var response = await client.DeletePolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted Policy: {policyId}.");
        }
        else
        {
```

```
                Console.WriteLine($"Could not delete Policy: {policyId}.");
            }
        }
    }
```

- Para obter detalhes da API, consulte DeletePolicy em Referência AWS SDK para .NET da API.

`DetachPolicy`

O código de exemplo a seguir mostra como usar `DetachPolicy`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to detach a policy from an AWS Organizations organization,
/// organizational unit, or account.
/// </summary>
public class DetachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and uses it to call
    /// DetachPolicyAsync to detach the policy.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            var policyId = "p-00000000";
            var targetId = "r-0000";

            var request = new DetachPolicyRequest
            {
                PolicyId = policyId,
                TargetId = targetId,
            };

            var response = await client.DetachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully detached policy with Policy Id:
{policyId}.");
            }
            else
            {
                Console.WriteLine("Could not detach the policy.");
            }
        }
    }
```

- Para obter detalhes da API, consulte DetachPolicy em Referência AWS SDK para .NET da API.

`ListAccounts`

O código de exemplo a seguir mostra como usar `ListAccounts`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Uses the AWS Organizations service to list the accounts associated
/// with the default account.
/// </summary>
public class ListAccounts
{
    /// <summary>
    /// Creates the Organizations client and then calls its
    /// ListAccountsAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var request = new ListAccountsRequest
        {
            MaxResults = 5,
        };

        var response = new ListAccountsResponse();
        try
        {
            do
            {
                response = await client.ListAccountsAsync(request);
                response.Accounts.ForEach(a => DisplayAccounts(a));
                if (response.NextToken is not null)
                {
                    request.NextToken = response.NextToken;
                }
            }
            while (response.NextToken is not null);
        }
        catch (AWSOrganizationsNotInUseException ex)
        {
            Console.WriteLine(ex.Message);
        }
```

```
        }

        /// <summary>
        /// Displays information about an Organizations account.
        /// </summary>
        /// <param name="account">An Organizations account for which to display
        /// information on the console.</param>
        private static void DisplayAccounts(Account account)
        {
            string accountInfo = $"{account.Id} {account.Name}\t{account.Status}";

            Console.WriteLine(accountInfo);
        }
    }
```

- Para obter detalhes da API, consulte ListAccounts em Referência AWS SDK para .NET da API.

`ListOrganizationalUnitsForParent`

O código de exemplo a seguir mostra como usar `ListOrganizationalUnitsForParent`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;

    /// <summary>
    /// Lists the AWS Organizations organizational units that belong to an
    /// organization.
    /// </summary>
```

```
    public class ListOrganizationalUnitsForParent
    {
        /// <summary>
        /// Initializes the Organizations client object and then uses it to
        /// call the ListOrganizationalUnitsForParentAsync method to retrieve
        /// the list of organizational units.
        /// </summary>
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var parentId = "r-0000";

            var request = new ListOrganizationalUnitsForParentRequest
            {
                ParentId = parentId,
                MaxResults = 5,
            };

            var response = new ListOrganizationalUnitsForParentResponse();
            try
            {
                do
                {
                    response = await
client.ListOrganizationalUnitsForParentAsync(request);
                    response.OrganizationalUnits.ForEach(u =>
DisplayOrganizationalUnit(u));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about an Organizations organizational unit.
```

```
        /// </summary>
        /// <param name="unit">The OrganizationalUnit for which to display
        /// information.</param>
        public static void DisplayOrganizationalUnit(OrganizationalUnit unit)
        {
            string accountInfo = $"{unit.Id} {unit.Name}\t{unit.Arn}";

            Console.WriteLine(accountInfo);
        }
    }
```

- Para obter detalhes da API, consulte ListOrganizationalUnitsForParent em Referência AWS SDK para .NET da API.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to list the AWS Organizations policies associated with an
/// organization.
/// </summary>
public class ListPolicies
{
    /// <summary>
    /// Initializes an Organizations client object, and then calls its
```

```
        /// ListPoliciesAsync method.
        /// </summary>
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            // The value for the Filter parameter is required and must must be
            // one of the following:
            //     AISERVICES_OPT_OUT_POLICY
            //     BACKUP_POLICY
            //     SERVICE_CONTROL_POLICY
            //     TAG_POLICY
            var request = new ListPoliciesRequest
            {
                Filter = "SERVICE_CONTROL_POLICY",
                MaxResults = 5,
            };

            var response = new ListPoliciesResponse();
            try
            {
                do
                {
                    response = await client.ListPoliciesAsync(request);
                    response.Policies.ForEach(p => DisplayPolicies(p));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about the Organizations policies associated
        /// with an organization.
        /// </summary>
        /// <param name="policy">An Organizations policy summary to display
```

```
        /// information on the console.</param>
        private static void DisplayPolicies(PolicySummary policy)
        {
            string policyInfo = $"{policy.Id} {policy.Name}\t{policy.Description}";

            Console.WriteLine(policyInfo);
        }
    }
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para .NET da API.

Exemplos do Partner Central usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Partner Central.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateOpportunity`

O código de exemplo a seguir mostra como usar `CreateOpportunity`.

SDK para .NET

Crie uma oportunidade.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0
```

```
using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new CreateOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Origin = "Partner Referral",
                    Customer = new Customer
                    {
                        Account = new Account
                        {
                            Address = new Address
                            {
                                CountryCode = "US",
                                PostalCode = "99502",
                                StateOrRegion = "Alaska"
                            },
                            CompanyName = "TestCompanyName",
                            Duns = "123456789",
                            WebsiteUrl = "www.test.io",
                            Industry = "Automotive"
                        },
                        Contacts = new List<Contact>
```

```
                        {
                            new Contact
                            {
                                Email = "test@test.io",
                                FirstName = "John  ",
                                LastName = "Doe",
                                Phone = "+14444444444",
                                BusinessTitle = "test title"
                            }
                        }
                    },
                    LifeCycle = new LifeCycle
                    {
                        ReviewStatus = "Submitted",
                        TargetCloseDate = "2024-12-30"
                    },
                    Marketing = new Marketing
                    {
                        Source = "None"
                    },
                    OpportunityType = "Net New Business",
                    PrimaryNeedsFromAws = new List<string> { "Co-Sell -
Architectural Validation" },
                    Project = new Project
                    {
                        Title = "Moin Test UUID",
                        CustomerBusinessProblem = "Sandbox is not working as
expected",
                        CustomerUseCase = "AI Machine Learning and Analytics",
                        DeliveryModels = new List<string> { "SaaS or PaaS" },
                        ExpectedCustomerSpend = new List<ExpectedCustomerSpend>
                        {
                            new ExpectedCustomerSpend
                            {
                                Amount = "2000.0",
                                CurrencyCode = "USD",
                                Frequency = "Monthly",
                                TargetCompany = "Ibexlabs"
                            }
                        },
                        SalesActivities = new List<string> { "Initialized
discussions with customer" }
                    }
                };
```

```
                try
                {
                    var response = await client.CreateOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                }
                catch (ValidationException ex)
                {
                    Console.WriteLine("Validation error: " + ex.Message);
                }
                catch (AmazonPartnerCentralSellingException e)
                {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- Para obter detalhes da API, consulte CreateOpportunity em Referência AWS SDK para .NET da API.

`GetOpportunity`

O código de exemplo a seguir mostra como usar `GetOpportunity`.

SDK para .NET

Tenha uma oportunidade.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0
```

```
using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static readonly string identifier = "O1111111";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new GetOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Identifier = identifier
                };

                try {
                    var response = await client.GetOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
```

```
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- Para obter detalhes da API, consulte GetOpportunity em Referência AWS SDK para .NET da API.

`ListOpportunities`

O código de exemplo a seguir mostra como usar `ListOpportunities`.

SDK para .NET

Liste oportunidades.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "Sandbox";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
```

```
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                //var config = new AmazonPartnerCentralSellingConfig()
                //{
                //    ServiceURL = "https://partnercentral-selling.us-
east-1.api.aws",
                //};
                //var client = new AmazonPartnerCentralSellingClient(awsCredentials,
 config);
                var client = new AmazonPartnerCentralSellingClient(awsCredentials);
                var request = new ListOpportunitiesRequest
                {
                    Catalog = catalogToUse,
                    MaxResults = 2
                };

                try {
                    var response = await client.ListOpportunitiesAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    foreach (var opportunity in response.OpportunitySummaries)
                    {
                        Console.WriteLine("Opportunity id: " + opportunity.Id);
                    }
                    string formattedJson =
 JsonConvert.SerializeObject(response.OpportunitySummaries, Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- Para obter detalhes da API, consulte ListOpportunities em Referência AWS SDK para .NET da API.

Exemplos do Amazon Pinpoint usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendMessages`

O código de exemplo a seguir mostra como usar `SendMessages`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;
```

```
using Microsoft.Extensions.Configuration;

namespace SendEmailMessage;

public class SendEmailMainClass
{
    public static async Task Main(string[] args)
    {
        var configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load test settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();

        // The AWS Region that you want to use to send the email. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        string region = "us-east-1";

        // The "From" address. This address has to be verified in Amazon Pinpoint
        // in the region you're using to send email.
        string senderAddress = configuration["SenderAddress"]!;

        // The address on the "To" line. If your Amazon Pinpoint account is in
        // the sandbox, this address also has to be verified.
        string toAddress = configuration["ToAddress"]!;

        // The Amazon Pinpoint project/application ID to use when you send this
 message.
        // Make sure that the SMS channel is enabled for the project or application
        // that you choose.
        string appId = configuration["AppId"]!;

        try
        {
            await SendEmailMessage(region, appId, toAddress, senderAddress);
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
 ex.Message);
        }
    }
```

```
    public static async Task<MessageResponse> SendEmailMessage(
        string region, string appId, string toAddress, string senderAddress)
    {
        var client = new
 AmazonPinpointClient(RegionEndpoint.GetBySystemName(region));

        // The subject line of the email.
        string subject = "Amazon Pinpoint Email test";

        // The body of the email for recipients whose email clients don't
        // support HTML content.
        string textBody = @"Amazon Pinpoint Email Test (.NET)"
                          + "\n---------------------------------"
                          + "\nThis email was sent using the Amazon Pinpoint API
 using the AWS SDK for .NET.";

        // The body of the email for recipients whose email clients support
        // HTML content.
        string htmlBody = @"<html>"
                          + "\n<head></head>"
                          + "\n<body>"
                          + "\n  <h1>Amazon Pinpoint Email Test (AWS SDK for .NET)</
h1>"
                          + "\n  <p>This email was sent using the "
                          + "\n    <a href='https://aws.amazon.com/pinpoint/'>Amazon
 Pinpoint</a> API "
                          + "\n    using the <a href='https://aws.amazon.com/sdk-
for-net/'>AWS SDK for .NET</a>"
                          + "\n  </p>"
                          + "\n</body>"
                          + "\n</html>";

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        string charset = "UTF-8";

        var sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses = new Dictionary<string, AddressConfiguration>
                {
```

```
                    {
                        toAddress,
                        new AddressConfiguration
                        {
                            ChannelType = ChannelType.EMAIL
                        }
                    }
                },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    EmailMessage = new EmailMessage
                    {
                        FromAddress = senderAddress,
                        SimpleEmail = new SimpleEmail
                        {
                            HtmlPart = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = htmlBody
                            },
                            TextPart = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = textBody
                            },
                            Subject = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = subject
                            }
                        }
                    }
                }
            }
        };
        Console.WriteLine("Sending message...");
        SendMessagesResponse response = await client.SendMessagesAsync(sendRequest);
        Console.WriteLine("Message sent!");
        return response.MessageResponse;
    }
}
```

Envie uma mensagem SMS.

```
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;
using Microsoft.Extensions.Configuration;

namespace SendSmsMessage;

public class SendSmsMessageMainClass
{
    public static async Task Main(string[] args)
    {
        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // The AWS Region that you want to use to send the message. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        string region = "us-east-1";

        // The phone number or short code to send the message from. The phone number
        // or short code that you specify has to be associated with your Amazon
 Pinpoint
        // account. For best results, specify long codes in E.164 format.
        string originationNumber = configuration["OriginationNumber"]!;

        // The recipient's phone number.  For best results, you should specify the
        // phone number in E.164 format.
        string destinationNumber = configuration["DestinationNumber"]!;

        // The Pinpoint project/ application ID to use when you send this message.
        // Make sure that the SMS channel is enabled for the project or application
        // that you choose.
        string appId = configuration["AppId"]!;

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
```

```
        MessageType messageType = MessageType.TRANSACTIONAL;

        // The registered keyword associated with the originating short code.
        string? registeredKeyword = configuration["RegisteredKeyword"];

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        string? senderId = configuration["SenderId"];

        try
        {
            var response = await SendSmsMessage(region, appId, destinationNumber,
                originationNumber, registeredKeyword, senderId, messageType);
            Console.WriteLine($"Message sent to
 {response.MessageResponse.Result.Count} recipient(s).");
            foreach (var messageResultValue in
                     response.MessageResponse.Result.Select(r => r.Value))
            {
                Console.WriteLine($"{messageResultValue.MessageId} Status:
 {messageResultValue.DeliveryStatus}");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
 ex.Message);
        }
    }

    public static async Task<SendMessagesResponse> SendSmsMessage(
        string region, string appId, string destinationNumber, string
 originationNumber,
        string? keyword, string? senderId, MessageType messageType)
    {

        // The content of the SMS message.
        string message = "This message was sent through Amazon Pinpoint using" +
                         " the AWS SDK for .NET. Reply STOP to opt out.";


        var client = new
 AmazonPinpointClient(RegionEndpoint.GetBySystemName(region));
```

```
        SendMessagesRequest sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses =
                    new Dictionary<string, AddressConfiguration>
                    {
                        {
                            destinationNumber,
                            new AddressConfiguration { ChannelType =
 ChannelType.SMS }
                        }
                    },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    SMSMessage = new SMSMessage
                    {
                        Body = message,
                        MessageType = MessageType.TRANSACTIONAL,
                        OriginationNumber = originationNumber,
                        SenderId = senderId,
                        Keyword = keyword
                    }
                }
            }
        };
        SendMessagesResponse response = await client.SendMessagesAsync(sendRequest);
        return response;
    }
}
```

- Para obter detalhes da API, consulte SendMessages em Referência AWS SDK para .NET da API.

Exemplos de uso do Amazon Polly SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon Polly.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DeleteLexicon`

O código de exemplo a seguir mostra como usar `DeleteLexicon`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Deletes an existing Amazon Polly lexicon using the AWS SDK for .NET.
/// </summary>
public class DeleteLexicon
{
    public static async Task Main()
    {
```

```
            string lexiconName = "SampleLexicon";

            var client = new AmazonPollyClient();

            var success = await DeletePollyLexiconAsync(client, lexiconName);

            if (success)
            {
                Console.WriteLine($"Successfully deleted {lexiconName}.");
            }
            else
            {
                Console.WriteLine($"Could not delete {lexiconName}.");
            }
        }

        /// <summary>
        /// Deletes the named Amazon Polly lexicon.
        /// </summary>
        /// <param name="client">The initialized Amazon Polly client object.</param>
        /// <param name="lexiconName">The name of the Amazon Polly lexicon to
        /// delete.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeletePollyLexiconAsync(
            AmazonPollyClient client,
            string lexiconName)
        {
            var deleteLexiconRequest = new DeleteLexiconRequest()
            {
                Name = lexiconName,
            };

            var response = await client.DeleteLexiconAsync(deleteLexiconRequest);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Para obter detalhes da API, consulte DeleteLexicon em Referência AWS SDK para .NET da API.

`DescribeVoices`

O código de exemplo a seguir mostra como usar `DescribeVoices`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DescribeVoices
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();

        var allVoicesRequest = new DescribeVoicesRequest();
        var enUsVoicesRequest = new DescribeVoicesRequest()
        {
            LanguageCode = "en-US",
        };

        try
        {
            string nextToken;
            do
            {
                var allVoicesResponse = await
client.DescribeVoicesAsync(allVoicesRequest);
                nextToken = allVoicesResponse.NextToken;
                allVoicesRequest.NextToken = nextToken;

                Console.WriteLine("\nAll voices: ");
                allVoicesResponse.Voices.ForEach(voice =>
                {
                    DisplayVoiceInfo(voice);
```

```
                    });
                }
                while (nextToken is not null);

                do
                {
                    var enUsVoicesResponse = await
client.DescribeVoicesAsync(enUsVoicesRequest);
                    nextToken = enUsVoicesResponse.NextToken;
                    enUsVoicesRequest.NextToken = nextToken;

                    Console.WriteLine("\nen-US voices: ");
                    enUsVoicesResponse.Voices.ForEach(voice =>
                    {
                        DisplayVoiceInfo(voice);
                    });
                }
                while (nextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception caught: " + ex.Message);
            }
        }

        public static void DisplayVoiceInfo(Voice voice)
        {
            Console.WriteLine($" Name: {voice.Name}\tGender:
{voice.Gender}\tLanguageName: {voice.LanguageName}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeVoices em Referência AWS SDK para .NET da API.

GetLexicon

O código de exemplo a seguir mostra como usar `GetLexicon`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Retrieves information about a specific Amazon Polly lexicon.
/// </summary>
public class GetLexicon
{
    public static async Task Main(string[] args)
    {
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();

        await GetPollyLexiconAsync(client, lexiconName);
    }

    public static async Task GetPollyLexiconAsync(AmazonPollyClient client,
string lexiconName)
    {
        var getLexiconRequest = new GetLexiconRequest()
        {
            Name = lexiconName,
        };

        try
        {
            var response = await client.GetLexiconAsync(getLexiconRequest);
            Console.WriteLine($"Lexicon:\n Name: {response.Lexicon.Name}");
            Console.WriteLine($"Content: {response.Lexicon.Content}");
        }
        catch (Exception ex)
```

```
            {
                Console.WriteLine("Error: " + ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte GetLexicon em Referência AWS SDK para .NET da API.

`ListLexicons`

O código de exemplo a seguir mostra como usar `ListLexicons`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Lists the Amazon Polly lexicons that have been defined. By default,
/// lists the lexicons that are defined in the same AWS Region as the default
/// user. To view Amazon Polly lexicons that are defined in a different AWS
/// Region, supply it as a parameter to the Amazon Polly constructor.
/// </summary>
public class ListLexicons
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();
        var request = new ListLexiconsRequest();

        try
```

```
            {
                Console.WriteLine("All voices: ");

                do
                {
                    var response = await client.ListLexiconsAsync(request);
                    request.NextToken = response.NextToken;

                    response.Lexicons.ForEach(lexicon =>
                    {
                        var attributes = lexicon.Attributes;
                        Console.WriteLine($"Name: {lexicon.Name}");
                        Console.WriteLine($"\tAlphabet: {attributes.Alphabet}");
                        Console.WriteLine($"\tLanguageCode:
{attributes.LanguageCode}");
                        Console.WriteLine($"\tLastModified:
{attributes.LastModified}");
                        Console.WriteLine($"\tLexemesCount:
{attributes.LexemesCount}");
                        Console.WriteLine($"\tLexiconArn: {attributes.LexiconArn}");
                        Console.WriteLine($"\tSize: {attributes.Size}");
                    });
                }
                while (request.NextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- Para obter detalhes da API, consulte ListLexicons em Referência AWS SDK para .NET da API.

PutLexicon

O código de exemplo a seguir mostra como usar `PutLexicon`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Polly;
    using Amazon.Polly.Model;

    /// <summary>
    /// Creates a new Amazon Polly lexicon using the AWS SDK for .NET.
    /// </summary>
    public class PutLexicon
    {
        public static async Task Main()
        {
            string lexiconContent = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
                "<lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/
pronunciation-lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " +
                "xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-
lexicon http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" " +
                "alphabet=\"ipa\" xml:lang=\"en-US\">" +
                "<lexeme><grapheme>test1</grapheme><alias>test2</alias></lexeme>" +
                "</lexicon>";
            string lexiconName = "SampleLexicon";

            var client = new AmazonPollyClient();
            var putLexiconRequest = new PutLexiconRequest()
            {
                Name = lexiconName,
                Content = lexiconContent,
            };

            try
            {
                var response = await client.PutLexiconAsync(putLexiconRequest);
                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
```

```
                    Console.WriteLine($"Successfully created Lexicon:
{lexiconName}.");
                }
                else
                {
                    Console.WriteLine($"Could not create Lexicon: {lexiconName}.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception caught: " + ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte PutLexicon em Referência AWS SDK para .NET da API.

`SynthesizeSpeech`

O código de exemplo a seguir mostra como usar `SynthesizeSpeech`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeech
{
    public static async Task Main()
    {
        string outputFileName = "speech.mp3";
```

```
            string text = "Twas brillig, and the slithy toves did gyre and gimbol in
the wabe";

            var client = new AmazonPollyClient();
            var response = await PollySynthesizeSpeech(client, text);

            WriteSpeechToStream(response.AudioStream, outputFileName);
        }

        /// <summary>
        /// Calls the Amazon Polly SynthesizeSpeechAsync method to convert text
        /// to speech.
        /// </summary>
        /// <param name="client">The Amazon Polly client object used to connect
        /// to the Amazon Polly service.</param>
        /// <param name="text">The text to convert to speech.</param>
        /// <returns>A SynthesizeSpeechResponse object that includes an AudioStream
        /// object with the converted text.</returns>
        private static async Task<SynthesizeSpeechResponse>
PollySynthesizeSpeech(IAmazonPolly client, string text)
        {
            var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
            {
                OutputFormat = OutputFormat.Mp3,
                VoiceId = VoiceId.Joanna,
                Text = text,
            };

            var synthesizeSpeechResponse =
                await client.SynthesizeSpeechAsync(synthesizeSpeechRequest);

            return synthesizeSpeechResponse;
        }

        /// <summary>
        /// Writes the AudioStream returned from the call to
        /// SynthesizeSpeechAsync to a file in MP3 format.
        /// </summary>
        /// <param name="audioStream">The AudioStream returned from the
        /// call to the SynthesizeSpeechAsync method.</param>
        /// <param name="outputFileName">The full path to the file in which to
        /// save the audio stream.</param>
        private static void WriteSpeechToStream(Stream audioStream, string
outputFileName)
```

```
        {
            var outputStream = new FileStream(
                outputFileName,
                FileMode.Create,
                FileAccess.Write);
            byte[] buffer = new byte[2 * 1024];
            int readBytes;

            while ((readBytes = audioStream.Read(buffer, 0, 2 * 1024)) > 0)
            {
                outputStream.Write(buffer, 0, readBytes);
            }

            // Flushes the buffer to avoid losing the last second or so of
            // the synthesized text.
            outputStream.Flush();
            Console.WriteLine($"Saved {outputFileName} to disk.");
        }
    }
```

Sintetize a fala a partir do texto usando marcas de fala com o Amazon Polly usando um SDK. AWS

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Polly;
    using Amazon.Polly.Model;

    public class SynthesizeSpeechMarks
    {
        public static async Task Main()
        {
            var client = new AmazonPollyClient();
            string outputFileName = "speechMarks.json";

            var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
            {
                OutputFormat = OutputFormat.Json,
                SpeechMarkTypes = new List<string>
```

```
                {
                    SpeechMarkType.Viseme,
                    SpeechMarkType.Word,
                },
                VoiceId = VoiceId.Joanna,
                Text = "This is a sample text to be synthesized.",
            };

            try
            {
                using (var outputStream = new FileStream(outputFileName,
FileMode.Create, FileAccess.Write))
                {
                    var synthesizeSpeechResponse = await
client.SynthesizeSpeechAsync(synthesizeSpeechRequest);
                    var buffer = new byte[2 * 1024];
                    int readBytes;

                    var inputStream = synthesizeSpeechResponse.AudioStream;
                    while ((readBytes = inputStream.Read(buffer, 0, 2 * 1024)) > 0)
                    {
                        outputStream.Write(buffer, 0, readBytes);
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- Para obter detalhes da API, consulte SynthesizeSpeech em Referência AWS SDK para .NET da API.

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon RDS usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon RDS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon RDS

O exemplo de código a seguir mostra como começar a usar o Amazon RDS.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.RDS;
using Amazon.RDS.Model;

namespace RDSActions;
```

```
public static class HelloRds
{
    static async Task Main(string[] args)
    {
        var rdsClient = new AmazonRDSClient();

        Console.WriteLine($"Hello Amazon RDS! Following are some of your DB
 instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first twenty DB instances.
        var response = await rdsClient.DescribeDBInstancesAsync(
            new DescribeDBInstancesRequest()
            {
                MaxRecords = 20 // Must be between 20 and 100.
            });

        foreach (var instance in response.DBInstances)
        {
            Console.WriteLine($"\tDB name: {instance.DBName}");
            Console.WriteLine($"\tArn: {instance.DBInstanceArn}");
            Console.WriteLine($"\tIdentifier: {instance.DBInstanceIdentifier}");
            Console.WriteLine();
        }
    }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para .NET .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de banco de dados e definir os valores dos parâmetros.
- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.

- Exclua a instância e o grupo de parâmetros.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
/// <summary>
/// Scenario for RDS DB instance example.
/// </summary>
public class RDSInstanceScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Returns a list of the available DB engine families using the
 DescribeDBEngineVersionsAsync method.
    2.  Selects an engine family and creates a custom DB parameter group using the
 CreateDBParameterGroupAsync method.
    3.  Gets the parameter groups using the DescribeDBParameterGroupsAsync method.
    4.  Gets parameters in the group using the DescribeDBParameters method.
    5.  Parses and displays parameters in the group.
    6.  Modifies both the auto_increment_offset and auto_increment_increment
 parameters
        using the ModifyDBParameterGroupAsync method.
    7.  Gets and displays the updated parameters using the DescribeDBParameters
 method with a source of "user".
    8.  Gets a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Displays and selects from a list of micro instance classes available for the
 selected engine and version.
    10. Creates an RDS DB instance that contains a MySql database and uses the
 parameter group
        using the CreateDBInstanceAsync method.
```

```
   11. Waits for DB instance to be ready using the DescribeDBInstancesAsync method.
   12. Prints out the connection endpoint string for the new DB instance.
   13. Creates a snapshot of the DB instance using the CreateDBSnapshotAsync
method.
   14. Waits for DB snapshot to be ready using the DescribeDBSnapshots method.
   15. Deletes the DB instance using the DeleteDBInstanceAsync method.
   16. Waits for DB instance to be deleted using the DescribeDbInstances method.
   17. Deletes the parameter group using the DeleteDBParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static RDSWrapper rdsWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon RDS service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonRDS>()
                   .AddTransient<RDSWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger<RDSInstanceScenario>();

       rdsWrapper = host.Services.GetRequiredService<RDSWrapper>();

       Console.WriteLine(sepBar);
       Console.WriteLine(
           "Welcome to the Amazon Relational Database Service (Amazon RDS) DB
instance scenario example.");
       Console.WriteLine(sepBar);

       try
       {
```

```
            var parameterGroupFamily = await ChooseParameterGroupFamily();

            var parameterGroup = await CreateDbParameterGroup(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroup(parameterGroup.DBParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await ModifyParameters(parameterGroup.DBParameterGroupName, parameters);

            await DescribeUserSourceParameters(parameterGroup.DBParameterGroupName);

            var engineVersionChoice = await
ChooseDbEngineVersion(parameterGroupFamily);

            var instanceChoice = await ChooseDbInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            var newInstance = await CreateRdsNewInstance(parameterGroup, engine,
engineVersionChoice.EngineVersion,
                instanceChoice.DBInstanceClass, newInstanceIdentifier);
            if (newInstance != null)
            {
                DisplayConnectionString(newInstance);

                await CreateSnapshot(newInstance);

                await DeleteRdsInstance(newInstance);
            }

            await DeleteParameterGroup(parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }
```

```
    /// <summary>
    /// Choose the RDS DB parameter group family from a list of available options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamily()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await rdsWrapper.DescribeDBEngineVersions(engine);

        Console.WriteLine("1. The following is a list of available DB parameter
group families:");
        int i = 1;
        var parameterGroupFamilies = engines.GroupBy(e =>
e.DBParameterGroupFamily).ToList();
        foreach (var parameterGroupFamily in parameterGroupFamilies)
        {
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("Select an available DB parameter group family by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the new
DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBParameterGroup> CreateDbParameterGroup(string
dbParameterGroupFamily)
```

```
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

        var parameterGroup = await rdsWrapper.CreateDBParameterGroup(
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            dbParameterGroupFamily, "New example parameter group");

        var groupInfo =
            await rdsWrapper.DescribeDBParameterGroups(parameterGroup
                .DBParameterGroupName);

        Console.WriteLine(
            $"3. New DB parameter group: \n\t{groupInfo[0].Description}, \n\tARN
{groupInfo[0].DBParameterGroupArn}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameterNames">Optional specific names of parameters to
describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>> DescribeParametersInGroup(string
parameterGroupName, List<string>? parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
```

```
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParameters(string parameterGroupName,
List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        foreach (var p in parameters)
        {
            if (p.IsModifiable && p.DataType == "integer")
            {
                int newValue = 0;
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the allowed
values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    Int32.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        await rdsWrapper.ModifyDBParameterGroup(parameterGroupName, parameters);

        Console.WriteLine(sepBar);
```

```
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe user source parameters in the group.");

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName, "user");


        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }


    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDbEngineVersion(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await rdsWrapper.DescribeDBEngineVersions(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group family
{dbParameterGroupFamily}:");
```

```
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. Engine: {version.Engine} Version
 {version.EngineVersion}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by entering
 a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption> ChooseDbInstanceClass(string
 engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await rdsWrapper.DescribeOrderableDBInstanceOptions(engine,
 engineVersion);

        Console.WriteLine($"8. Available micro DB instance classes for engine
 {engine} and version {engineVersion}:");
        int i = 1;

        // Filter to micro instances for this example.
```

```
        allowedInstances = allowedInstances
            .Where(i => i.DBInstanceClass.Contains("micro")).ToList();

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("9. Select an available DB instance class by entering
 a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new RDS DB instance.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB instance.</
param>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateRdsNewInstance(DBParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string instanceClass, string
 instanceIdentifier)
    {
        Console.WriteLine(sepBar);
```

```
        Console.WriteLine($"10. Create a new DB instance with identifier
{instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await rdsWrapper.DescribeDBInstances();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
        }
        else
        {
            Console.WriteLine("Please enter an admin user name:");
            var username = Console.ReadLine();

            Console.WriteLine("Please enter an admin password:");
            var password = Console.ReadLine();

            newInstance = await rdsWrapper.CreateDBInstance(
                "ExampleInstance",
                instanceIdentifier,
                parameterGroup.DBParameterGroupName,
                engineName,
                engineVersion,
                instanceClass,
                20,
                username,
                password
            );

            // 11. Wait for the DB instance to be ready.

            Console.WriteLine("11. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                instances = await
rdsWrapper.DescribeDBInstances(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
```

```
                newInstance = instances.First();
                Thread.Sleep(30000);
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("12. New DB instance connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {instance.Endpoint.Address} -P {instance.Endpoint.Port}
 "
            + $"-u {instance.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an RDS DB instance.
    /// </summary>
    /// <param name="instance">DB instance to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBSnapshot> CreateSnapshot(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"13. Creating snapshot from DB instance
 {instance.DBInstanceIdentifier}.");
        var snapshot = await
 rdsWrapper.CreateDBSnapshot(instance.DBInstanceIdentifier, "ExampleSnapshot-" +
 DateTime.Now.Ticks);

        // Wait for the snapshot to be available
        bool isSnapshotReady = false;
```

```
        Console.WriteLine($"14. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            var snapshots = await
rdsWrapper.DescribeDBSnapshots(instance.DBInstanceIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
            Thread.Sleep(30000);
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Delete an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteRdsInstance(DBInstance newInstance)
    {
        Console.WriteLine(sepBar);
        // Delete the DB instance.
        Console.WriteLine($"15. Delete the DB instance
{newInstance.DBInstanceIdentifier}.");
        await rdsWrapper.DeleteDBInstance(newInstance.DBInstanceIdentifier);

        // Wait for the DB instance to delete.
        Console.WriteLine($"16. Waiting for the DB instance to delete...");
        bool isInstanceDeleted = false;

        while (!isInstanceDeleted)
        {
            var instance = await rdsWrapper.DescribeDBInstances();
            isInstanceDeleted = instance.All(i => i.DBInstanceIdentifier !=
newInstance.DBInstanceIdentifier);
            Thread.Sleep(30000);
        }

        Console.WriteLine("DB instance deleted.");
```

```
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Delete a DB parameter group.
    /// </summary>
    /// <param name="parameterGroup">The parameter group to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteParameterGroup(DBParameterGroup parameterGroup)
    {
        Console.WriteLine(sepBar);
        // Delete the parameter group.
        Console.WriteLine($"17. Delete the DB parameter group
 {parameterGroup.DBParameterGroupName}.");
        await
 rdsWrapper.DeleteDBParameterGroup(parameterGroup.DBParameterGroupName);

        Console.WriteLine(sepBar);
    }
```

Os métodos de wrapper usados pelo cenário para as ações de instância de banco de dados.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with DB
 instance operations.
/// </summary>
public partial class RDSWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public RDSWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }


    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family name.</
param>
```

```
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }



    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

```
    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }



    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB) to
allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
```

```
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }



    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

Os métodos de wrapper usados pelo cenário para o grupos de parâmetros de banco de dados.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 parameter groups.
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string name
 = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }



    /// <summary>
    /// Create a new DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
```

```
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }



    /// <summary>
    /// Update a DB parameter group. Use the action DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }



    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
 group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
```

```
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }



    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter group.</
param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
 dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

Os métodos de wrapper usados pelo cenário para as ações de snapshot de banco de dados.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 snapshots.
/// </summary>
```

```
public partial class RDSWrapper
{

    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
 string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }



    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
 dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
```

```
        return results;
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Ações

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB) to
allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para .NET .

`CreateDBParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBParameterGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a new DB parameter group. Use the action
DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }
```

- Para obter detalhes da API, consulte CreateDBParameterGroup em Referência AWS SDK para .NET da API.

`CreateDBSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBSnapshot`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para .NET .

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete a particular DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <returns>DB instance object.</returns>
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
{
    var response = await _amazonRDS.DeleteDBInstanceAsync(
        new DeleteDBInstanceRequest()
        {
            DBInstanceIdentifier = dbInstanceIdentifier,
            SkipFinalSnapshot = true,
            DeleteAutomatedBackups = true
        });

    return response.DBInstance;
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para .NET .

`DeleteDBParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBParameterGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteDBParameterGroup em Referência AWS SDK para .NET da API.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family name.</
param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions em Referência AWS SDK para .NET da API.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
   public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
   {
       var results = new List<DBInstance>();
       var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
           new DescribeDBInstancesRequest
           {
               DBInstanceIdentifier = dbInstanceIdentifier
           });
       // Get the entire list using the paginator.
       await foreach (var instances in instancesPaginator.DBInstances)
       {
           results.Add(instances);
       }
       return results;
   }
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para .NET .

`DescribeDBParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBParameterGroups`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string name
 = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }
```

- Para obter detalhes da API, consulte DescribeDBParameterGroups em Referência AWS SDK para .NET da API.

`DescribeDBParameters`

O código de exemplo a seguir mostra como usar `DescribeDBParameters`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter group.</
param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
 dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para .NET .

`DescribeDBSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBSnapshots`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
        return results;
    }
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência da API AWS SDK para .NET .

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Get a list of orderable DB instance options for a specific
   /// engine and engine version.
   /// </summary>
   /// <param name="engine">Name of the engine.</param>
   /// <param name="engineVersion">Version of the engine.</param>
   /// <returns>List of OrderableDBInstanceOptions.</returns>
   public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
   {
       // Use a paginator to get a list of DB instance options.
       var results = new List<OrderableDBInstanceOption>();
       var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
           new DescribeOrderableDBInstanceOptionsRequest()
           {
               Engine = engine,
               EngineVersion = engineVersion,
           });
       // Get the entire list using the paginator.
       await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
       {
           results.Add(instanceOptions);
       }
       return results;
   }
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptions em Referência AWS SDK para .NET da API.

`ModifyDBParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBParameterGroup`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update a DB parameter group. Use the action DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }
```

- Para obter detalhes da API, consulte ModifyDBParameterGroup em Referência AWS SDK para .NET da API.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para .NET

Mostra como usar o AWS SDK para .NET para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful .NET.

- Integre um aplicativo web React com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando .NET.

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace aws_rds;

public class InputModel
{
    public string key1 { get; set; }
    public string key2 { get; set; }
}

public class Function
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
 FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
```

```
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
$"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

$"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +

$"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                $"Pwd={authToken};";


        try
        {
            await using var connection = new MySqlConnection(connectionString);
            await connection.OpenAsync();

            const string sql = "SELECT @param1 + @param2 AS Sum";

            await using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
"0"));
            command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
"0"));

            await using var reader = await command.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                int result = reader.GetInt32("Sum");

                //Sample Response: {"statusCode":200,"body":"{\"message\":\"The sum
is: 45\"}","isBase64Encoded":false}
                return new APIGatewayProxyResponse
                {
                    StatusCode = 200,
```

```
                    Body = JsonSerializer.Serialize(new { message = $"The sum is:
 {result}" })
                };
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = 500,
            Body = JsonSerializer.Serialize(new { error = "Internal server error" })
        };
    }
}
```

Exemplos do Amazon RDS Data Service usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET Amazon RDS Data Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para .NET

Mostra como usar o AWS SDK para .NET para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful .NET.

- Integre um aplicativo web React com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon Rekognition usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon Rekognition.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CompareFaces`

O código de exemplo a seguir mostra como usar `CompareFaces`.

Para obter mais informações, consulte Comparação de faces em imagens.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to compare faces in two images.
/// </summary>
public class CompareFaces
{
    public static async Task Main()
    {
        float similarityThreshold = 70F;
        string sourceImage = "source.jpg";
```

```
            string targetImage = "target.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            Amazon.Rekognition.Model.Image imageSource = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageSource.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine($"Failed to load source image: {sourceImage}");
                return;
            }

            Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Failed to load target image: {targetImage}");
                Console.WriteLine(ex.Message);
                return;
            }

            var compareFacesRequest = new CompareFacesRequest
            {
                SourceImage = imageSource,
                TargetImage = imageTarget,
```

```
                SimilarityThreshold = similarityThreshold,
            };

            // Call operation
            var compareFacesResponse = await
rekognitionClient.CompareFacesAsync(compareFacesRequest);

            // Display results
            compareFacesResponse.FaceMatches.ForEach(match =>
            {
                ComparedFace face = match.Face;
                BoundingBox position = face.BoundingBox;
                Console.WriteLine($"Face at {position.Left} {position.Top} matches
with {match.Similarity}% confidence.");
            });

            Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
face(s) that did not match.");
        }
    }
```

- Para obter detalhes da API, consulte CompareFaces em Referência AWS SDK para .NET da API.

`CreateCollection`

O código de exemplo a seguir mostra como usar `CreateCollection`.

Para obter mais informações, consulte Criar uma coleção.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
```

```
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses Amazon Rekognition to create a collection to which you can add
    /// faces using the IndexFaces operation.
    /// </summary>
    public class CreateCollection
    {
        public static async Task Main()
        {
            var rekognitionClient = new AmazonRekognitionClient();

            string collectionId = "MyCollection";
            Console.WriteLine("Creating collection: " + collectionId);

            var createCollectionRequest = new CreateCollectionRequest
            {
                CollectionId = collectionId,
            };

            CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
            Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
            Console.WriteLine($"Status code :
{createCollectionResponse.StatusCode}");
        }
    }
```

- Para obter detalhes da API, consulte CreateCollection em Referência AWS SDK para .NET da API.

`DeleteCollection`

O código de exemplo a seguir mostra como usar `DeleteCollection`.

Para obter mais informações, consulte Excluir uma coleção.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete an existing collection.
/// </summary>
public class DeleteCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Deleting collection: " + collectionId);

        var deleteCollectionRequest = new DeleteCollectionRequest()
        {
            CollectionId = collectionId,
        };

        var deleteCollectionResponse = await
rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
        Console.WriteLine($"{collectionId}:
{deleteCollectionResponse.StatusCode}");
    }
}
```

- Para obter detalhes da API, consulte DeleteCollection em Referência AWS SDK para .NET da API.

`DeleteFaces`

O código de exemplo a seguir mostra como usar `DeleteFaces`.

Para obter mais informações, consulte Excluir faces de uma coleção.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete one or more faces from
/// a Rekognition collection.
/// </summary>
public class DeleteFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" };

        var rekognitionClient = new AmazonRekognitionClient();

        var deleteFacesRequest = new DeleteFacesRequest()
        {
            CollectionId = collectionId,
            FaceIds = faces,
        };

        DeleteFacesResponse deleteFacesResponse = await
rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
        deleteFacesResponse.DeletedFaces.ForEach(face =>
        {
```

```
                Console.WriteLine($"FaceID: {face}");
            });
        }
    }
```

- Para obter detalhes da API, consulte DeleteFaces em Referência AWS SDK para .NET da API.

DescribeCollection

O código de exemplo a seguir mostra como usar `DescribeCollection`.

Para obter mais informações, consulte Descrever uma coleção.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to describe the contents of a
/// collection.
/// </summary>
public class DescribeCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine($"Describing collection: {collectionId}");
```

```
            var describeCollectionRequest = new DescribeCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var describeCollectionResponse = await
rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
            Console.WriteLine($"Collection ARN:
{describeCollectionResponse.CollectionARN}");
            Console.WriteLine($"Face count:
{describeCollectionResponse.FaceCount}");
            Console.WriteLine($"Face model version:
{describeCollectionResponse.FaceModelVersion}");
            Console.WriteLine($"Created:
{describeCollectionResponse.CreationTimestamp}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeCollection em Referência AWS SDK para .NET da API.

`DetectFaces`

O código de exemplo a seguir mostra como usar `DetectFaces`.

Para obter mais informações, consulte Detectar faces em uma imagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;
```

```
    /// <summary>
    /// Uses the Amazon Rekognition Service to detect faces within an image
    /// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DetectFaces
    {
        public static async Task Main()
        {
            string photo = "input.jpg";
            string bucket = "amzn-s3-demo-bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectFacesRequest = new DetectFacesRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },

                // Attributes can be "ALL" or "DEFAULT".
                // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and Quality.
                // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
                Attributes = new List<string>() { "ALL" },
            };

            try
            {
                DetectFacesResponse detectFacesResponse = await
 rekognitionClient.DetectFacesAsync(detectFacesRequest);
                bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
                foreach (FaceDetail face in detectFacesResponse.FaceDetails)
                {
                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"Confidence: {face.Confidence}");
                    Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
```

```
                    Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                    Console.WriteLine($"Brightness:
{face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                    if (hasAll)
                    {
                        Console.WriteLine($"Estimated age is between
{face.AgeRange.Low} and {face.AgeRange.High} years old.");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Exiba as informações da caixa delimitadora de todas as faces em uma imagem.

```
    using System;
    using System.Collections.Generic;
    using System.Drawing;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to display the details of the
    /// bounding boxes around the faces detected in an image.
    /// </summary>
    public class ImageOrientationBoundingBox
    {
        public static async Task Main()
        {
            string photo = @"D:\Development\AWS-Examples\Rekognition\target.jpg"; //
"photo.jpg";

            var rekognitionClient = new AmazonRekognitionClient();
```

```
            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            int height;
            int width;

            // Used to extract original photo width/height
            using (var imageBitmap = new Bitmap(photo))
            {
                height = imageBitmap.Height;
                width = imageBitmap.Width;
            }

            Console.WriteLine("Image Information:");
            Console.WriteLine(photo);
            Console.WriteLine("Image Height: " + height);
            Console.WriteLine("Image Width: " + width);

            try
            {
                var detectFacesRequest = new DetectFacesRequest()
                {
                    Image = image,
                    Attributes = new List<string>() { "ALL" },
                };

                DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
                detectFacesResponse.FaceDetails.ForEach(face =>
                {
```

```
                    Console.WriteLine("Face:");
                    ShowBoundingBoxPositions(
                        height,
                        width,
                        face.BoundingBox,
                        detectFacesResponse.OrientationCorrection);

                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"The detected face is estimated to be between
 {face.AgeRange.Low} and {face.AgeRange.High} years old.\n");
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Display the bounding box information for an image.
        /// </summary>
        /// <param name="imageHeight">The height of the image.</param>
        /// <param name="imageWidth">The width of the image.</param>
        /// <param name="box">The bounding box for a face found within the image.</
param>
        /// <param name="rotation">The rotation of the face's bounding box.</param>
        public static void ShowBoundingBoxPositions(int imageHeight, int imageWidth,
 BoundingBox box, string rotation)
        {
            float left;
            float top;

            if (rotation == null)
            {
                Console.WriteLine("No estimated orientation. Check Exif data.");
                return;
            }

            // Calculate face position based on image orientation.
            switch (rotation)
            {
                case "ROTATE_0":
```

```
                    left = imageWidth * box.Left;
                    top = imageHeight * box.Top;
                    break;
                case "ROTATE_90":
                    left = imageHeight * (1 - (box.Top + box.Height));
                    top = imageWidth * box.Left;
                    break;
                case "ROTATE_180":
                    left = imageWidth - (imageWidth * (box.Left + box.Width));
                    top = imageHeight * (1 - (box.Top + box.Height));
                    break;
                case "ROTATE_270":
                    left = imageHeight * box.Top;
                    top = imageWidth * (1 - box.Left - box.Width);
                    break;
                default:
                    Console.WriteLine("No estimated orientation information. Check
Exif data.");
                    return;
            }

            // Display face location information.
            Console.WriteLine($"Left: {left}");
            Console.WriteLine($"Top: {top}");
            Console.WriteLine($"Face Width: {imageWidth * box.Width}");
            Console.WriteLine($"Face Height: {imageHeight * box.Height}");
        }
    }
```

- Para obter detalhes da API, consulte DetectFaces em Referência AWS SDK para .NET da API.

`DetectLabels`

O código de exemplo a seguir mostra como usar `DetectLabels`.

Para obter mais informações, consulte Detectar rótulos em uma imagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectLabels
{
    public static async Task Main()
    {
        string photo = "del_river_02092020_01.jpg"; // "input.jpg";
        string bucket = "amzn-s3-demo-bucket"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectlabelsRequest = new DetectLabelsRequest
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
            MaxLabels = 10,
            MinConfidence = 75F,
        };

        try
        {
```

```
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Detecte rótulos em um arquivo de imagem armazenado em seu computador.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to detect labels within an image
    /// stored locally.
    /// </summary>
    public class DetectLabelsLocalFile
    {
        public static async Task Main()
        {
            string photo = "input.jpg";

            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
```

```
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            var rekognitionClient = new AmazonRekognitionClient();

            var detectlabelsRequest = new DetectLabelsRequest
            {
                Image = image,
                MaxLabels = 10,
                MinConfidence = 77F,
            };

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine($"Detected labels for {photo}");
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"{label.Name}: {label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectLabels em Referência AWS SDK para .NET da API.

DetectModerationLabels

O código de exemplo a seguir mostra como usar `DetectModerationLabels`.

Para obter mais informações, consulte Detectar imagens impróprias.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect unsafe content in a
/// JPEG or PNG format image.
/// </summary>
public class DetectModerationLabels
{
    public static async Task Main(string[] args)
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectModerationLabelsRequest = new DetectModerationLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
            MinConfidence = 60F,
        };

        try
```

```
            {
                var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
                {
                    Console.WriteLine($"Label: {label.Name}");
                    Console.WriteLine($"Confidence: {label.Confidence}");
                    Console.WriteLine($"Parent: {label.ParentName}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectModerationLabels em Referência AWS SDK para .NET da API.

`DetectText`

O código de exemplo a seguir mostra como usar `DetectText`.

Para obter mais informações, consulte Detectar texto em uma imagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```

```
    /// <summary>
    /// Uses the Amazon Rekognition Service to detect text in an image. The
    /// example was created using the AWS SDK for .NET version 3.7 and .NET
    /// Core 5.0.
    /// </summary>
    public class DetectText
    {
        public static async Task Main()
        {
            string photo = "Dad_photographer.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectTextRequest = new DetectTextRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
            };

            try
            {
                DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
                Console.WriteLine($"Detected lines and words for {photo}");
                detectTextResponse.TextDetections.ForEach(text =>
                {
                    Console.WriteLine($"Detected: {text.DetectedText}");
                    Console.WriteLine($"Confidence: {text.Confidence}");
                    Console.WriteLine($"Id : {text.Id}");
                    Console.WriteLine($"Parent Id: {text.ParentId}");
                    Console.WriteLine($"Type: {text.Type}");
                });
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
```

```
            }
        }
    }
```

- Para obter detalhes da API, consulte DetectText em Referência AWS SDK para .NET da API.

GetCelebrityInfo

O código de exemplo a seguir mostra como usar `GetCelebrityInfo`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
/// celebrity identified by the supplied celebrity Id.
/// </summary>
public class CelebrityInfo
{
    public static async Task Main()
    {
        string celebId = "nnnnnnnn";

        var rekognitionClient = new AmazonRekognitionClient();

        var celebrityInfoRequest = new GetCelebrityInfoRequest
        {
            Id = celebId,
        };
```

```
            Console.WriteLine($"Getting information for celebrity: {celebId}");

            var celebrityInfoResponse = await
rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

            // Display celebrity information.
            Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
            Console.WriteLine("Further information (if available):");
            celebrityInfoResponse.Urls.ForEach(url =>
            {
                Console.WriteLine(url);
            });
        }
    }
```

- Para obter detalhes da API, consulte GetCelebrityInfo em Referência AWS SDK para .NET da API.

`IndexFaces`

O código de exemplo a seguir mostra como usar `IndexFaces`.

Para obter mais informações, consulte Adicionar faces a uma coleção.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
```

```
    /// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
    /// bucket and then adds the information to a collection.
    /// </summary>
    public class AddFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection2";
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Image
            {
                S3Object = new S3Object
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var indexFacesRequest = new IndexFacesRequest
            {
                Image = image,
                CollectionId = collectionId,
                ExternalImageId = photo,
                DetectionAttributes = new List<string>() { "ALL" },
            };

            IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

            Console.WriteLine($"{photo} added");
            foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            {
                Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
            }
        }
    }
```

- Para obter detalhes da API, consulte IndexFaces em Referência AWS SDK para .NET da API.

ListCollections

O código de exemplo a seguir mostra como usar `ListCollections`.

Para obter mais informações, consulte Listar coleções.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to list the collection IDs in the
/// current account.
/// </summary>
public class ListCollections
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        var listCollectionsRequest = new ListCollectionsRequest
        {
            MaxResults = limit,
        };

        var listCollectionsResponse = new ListCollectionsResponse();

        do
```

```
            {
                if (listCollectionsResponse is not null)
                {
                    listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
                }

                listCollectionsResponse = await
rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

                listCollectionsResponse.CollectionIds.ForEach(id =>
                {
                    Console.WriteLine(id);
                });
            }
            while (listCollectionsResponse.NextToken is not null);
        }
    }
```

- Para obter detalhes da API, consulte ListCollections em Referência AWS SDK para .NET da API.

`ListFaces`

O código de exemplo a seguir mostra como usar `ListFaces`.

Para obter mais informações, consulte Listar faces em uma coleção.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
```

```
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to retrieve the list of faces
    /// stored in a collection.
    /// </summary>
    public class ListFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection2";

            var rekognitionClient = new AmazonRekognitionClient();

            var listFacesResponse = new ListFacesResponse();
            Console.WriteLine($"Faces in collection {collectionId}");

            var listFacesRequest = new ListFacesRequest
            {
                CollectionId = collectionId,
                MaxResults = 1,
            };

            do
            {
                listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
                listFacesResponse.Faces.ForEach(face =>
                {
                    Console.WriteLine(face.FaceId);
                });

                listFacesRequest.NextToken = listFacesResponse.NextToken;
            }
            while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
        }
    }
```

- Para obter detalhes da API, consulte ListFaces em Referência AWS SDK para .NET da API.

`RecognizeCelebrities`

O código de exemplo a seguir mostra como usar `RecognizeCelebrities`.

Para obter mais informações, consulte Reconhecer celebridades em uma imagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// </summary>
public class CelebritiesInImage
{
    public static async Task Main(string[] args)
    {
        string photo = "moviestars.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

        var img = new Amazon.Rekognition.Model.Image();
        byte[] data = null;
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
        }
        catch (Exception)
```

```
            {
                Console.WriteLine($"Failed to load file {photo}");
                return;
            }

            img.Bytes = new MemoryStream(data);
            recognizeCelebritiesRequest.Image = img;

            Console.WriteLine($"Looking for celebrities in image {photo}\n");

            var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);

            Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
            recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
            {
                Console.WriteLine($"Celebrity recognized: {celeb.Name}");
                Console.WriteLine($"Celebrity ID: {celeb.Id}");
                BoundingBox boundingBox = celeb.Face.BoundingBox;
                Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
                Console.WriteLine("Further information (if available):");
                celeb.Urls.ForEach(url =>
                {
                    Console.WriteLine(url);
                });
            });


Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count} face(s)
were unrecognized.");
        }
    }
```

- Para obter detalhes da API, consulte RecognizeCelebrities em Referência AWS SDK para .NET da API.

`SearchFaces`

O código de exemplo a seguir mostra como usar `SearchFaces`.

Para obter mais informações, consulte Pesquisar uma face (face ID).

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to find faces in an image that
/// match the face Id provided in the method request.
/// </summary>
public class SearchFacesMatchingId
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

        var rekognitionClient = new AmazonRekognitionClient();

        // Search collection for faces matching the face id.
        var searchFacesRequest = new SearchFacesRequest
        {
            CollectionId = collectionId,
            FaceId = faceId,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

        Console.WriteLine("Face matching faceId " + faceId);
```

```
            Console.WriteLine("Matche(s): ");
            searchFacesResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
{face.Similarity}");
            });
        }
    }
```

- Para obter detalhes da API, consulte SearchFaces em Referência AWS SDK para .NET da API.

`SearchFacesByImage`

O código de exemplo a seguir mostra como usar `SearchFacesByImage`.

Para obter mais informações, consulte Pesquisar uma face (imagem).

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection.
/// </summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
```

```
            string collectionId = "MyCollection";
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            // Get an image object from S3 bucket.
            var image = new Image()
            {
                S3Object = new S3Object()
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var searchFacesByImageRequest = new SearchFacesByImageRequest()
            {
                CollectionId = collectionId,
                Image = image,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesByImageResponse searchFacesByImageResponse = await
rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

            Console.WriteLine("Faces matching largest face in image from " + photo);
            searchFacesByImageResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
{face.Similarity}");
            });
        }
    }
```

- Para obter detalhes da API, consulte SearchFacesByImage em Referência AWS SDK para .NET da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para .NET

Mostra como usar a API .NET do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Exemplos de registro de domínio do Route 53 usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o registro de domínio AWS SDK para .NET com o Route 53.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Registro de domínios do Olá, Route 53

O exemplo de código a seguir mostra como começar a usar o registro de domínio do Route 53.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static class HelloRoute53Domains
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon Route 53 domain registration service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRoute53Domains>()
            ).Build();

        // Now the client is available for injection.
        var route53Client =
 host.Services.GetRequiredService<IAmazonRoute53Domains>();

        // You can use await and any of the async methods to get a response.
        var response = await route53Client.ListPricesAsync(new ListPricesRequest
 { Tld = "com" });
        Console.WriteLine($"Hello Amazon Route 53 Domains! Following are prices
 for .com domain operations:");
        var comPrices = response.Prices.FirstOrDefault();
        if (comPrices != null)
        {
            Console.WriteLine($"\tRegistration: {comPrices.RegistrationPrice?.Price}
 {comPrices.RegistrationPrice?.Currency}");
            Console.WriteLine($"\tRenewal: {comPrices.RenewalPrice?.Price}
 {comPrices.RenewalPrice?.Currency}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListPrices em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Listar os domínios atuais e as operações do ano passado.
- Ver o faturamento do ano passado e os preços dos tipos de domínio.
- Receber sugestões de domínio.
- Verificar a disponibilidade e a transferibilidade de um domínio.
- Opcionalmente, solicitar o registro de um domínio.
- Obter os detalhes de uma operação.
- Opcionalmente, obtenha os detalhes de um domínio.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
public static class Route53DomainScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. List current domains.
        2. List operations in the past year.
        3. View billing for the account in the past year.
        4. View prices for domain types.
        5. Get domain suggestions.
        6. Check domain availability.
        7. Check domain transferability.
        8. Optionally, request a domain registration.
        9. Get an operation detail.
       10. Optionally, get a domain detail.
   */

    private static Route53Wrapper _route53Wrapper = null!;
```

```
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
                    .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonRoute53Domains>()
                .AddTransient<Route53Wrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        var logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger(typeof(Route53DomainScenario));

        _route53Wrapper = host.Services.GetRequiredService<Route53Wrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon Route 53 domains example
scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            await ListDomains();
            await ListOperations();
            await ListBillingRecords();
            await ListPrices();
            await ListDomainSuggestions();
            await CheckDomainAvailability();
```

```
            await CheckDomainTransferability();
            var operationId = await RequestDomainRegistration();
            await GetOperationalDetail(operationId);
            await GetDomainDetails();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon Route 53 domains example scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List account registered domains.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomains()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. List account domains.");
        var domains = await _route53Wrapper.ListDomains();
        for (int i = 0; i < domains.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {domains[i].DomainName}");
        }

        if (!domains.Any())
        {
            Console.WriteLine("\tNo domains found in this account.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List domain operations in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListOperations()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List account domain operations in the past year.");
        var operations = await _route53Wrapper.ListOperations(
            DateTime.Today.AddYears(-1));
        for (int i = 0; i < operations.Count; i++)
        {
            Console.WriteLine($"\tOperation Id: {operations[i].OperationId}");
            Console.WriteLine($"\tStatus: {operations[i].Status}");
            Console.WriteLine($"\tDate: {operations[i].SubmittedDate}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List billing in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListBillingRecords()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. View billing for the account in the past year.");
        var billingRecords = await _route53Wrapper.ViewBilling(
            DateTime.Today.AddYears(-1),
            DateTime.Today);
        for (int i = 0; i < billingRecords.Count; i++)
        {
            Console.WriteLine($"\tBill Date:
{billingRecords[i].BillDate.ToShortDateString()}");
            Console.WriteLine($"\tOperation: {billingRecords[i].Operation}");
            Console.WriteLine($"\tPrice: {billingRecords[i].Price}");
        }
        if (!billingRecords.Any())
        {
            Console.WriteLine("\tNo billing records found in this account for the
past year.");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List prices for a few domain types.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListPrices()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. View prices for domain types.");
        var domainTypes = new List<string> { "net", "com", "org", "co" };

        var prices = await _route53Wrapper.ListPrices(domainTypes);
        foreach (var pr in prices)
        {
            Console.WriteLine($"\tName: {pr.Name}");
            Console.WriteLine($"\tRegistration: {pr.RegistrationPrice?.Price}
{pr.RegistrationPrice?.Currency}");
            Console.WriteLine($"\tRenewal: {pr.RenewalPrice?.Price}
{pr.RenewalPrice?.Currency}");
            Console.WriteLine($"\tTransfer: {pr.TransferPrice?.Price}
{pr.TransferPrice?.Currency}");
            Console.WriteLine($"\tChange Ownership: {pr.ChangeOwnershipPrice?.Price}
{pr.ChangeOwnershipPrice?.Currency}");
            Console.WriteLine($"\tRestoration: {pr.RestorationPrice?.Price}
{pr.RestorationPrice?.Currency}");
            Console.WriteLine();
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List domain suggestions for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomainSuggestions()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Get domain suggestions.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to get available domain
suggestions.");
            domainName = Console.ReadLine();
        }

        var suggestions = await _route53Wrapper.GetDomainSuggestions(domainName,
true, 5);
        foreach (var suggestion in suggestions)
        {
```

```
            Console.WriteLine($"\tSuggestion Name: {suggestion.DomainName}");
            Console.WriteLine($"\tAvailability: {suggestion.Availability}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check availability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckDomainAvailability()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Check domain availability.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to check domain availability.");
            domainName = Console.ReadLine();
        }

        var availability = await
_route53Wrapper.CheckDomainAvailability(domainName);
        Console.WriteLine($"\tAvailability: {availability}");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check transferability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckDomainTransferability()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Check domain transferability.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to check domain
transferability.");
            domainName = Console.ReadLine();
        }
```

```
        var transferability = await
 _route53Wrapper.CheckDomainTransferability(domainName);
        Console.WriteLine($"\tTransferability: {transferability}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check transferability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> RequestDomainRegistration()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Optionally, request a domain registration.");

        Console.WriteLine($"\tNote: This example uses domain request settings in
 settings.json.");
        Console.WriteLine($"\tTo change the domain registration settings, set the
 values in that file.");
        Console.WriteLine($"\tRemember, registering an actual domain will incur an
 account billing cost.");
        Console.WriteLine($"\tWould you like to begin a domain registration? (y/
n)");
        var ynResponse = Console.ReadLine();
        if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
        {
            string domainName = _configuration["DomainName"];
            ContactDetail contact = new ContactDetail();
            contact.CountryCode =
 CountryCode.FindValue(_configuration["Contact:CountryCode"]);
            contact.ContactType =
 ContactType.FindValue(_configuration["Contact:ContactType"]);

            _configuration.GetSection("Contact").Bind(contact);

            var operationId = await _route53Wrapper.RegisterDomain(
                domainName,
                Convert.ToBoolean(_configuration["AutoRenew"]),
                Convert.ToInt32(_configuration["DurationInYears"]),
                contact);
            if (operationId != null)
            {
```

```
                Console.WriteLine(
                    $"\tRegistration requested. Operation Id: {operationId}");
            }

            return operationId;
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }

    /// <summary>
    /// Get details for an operation.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetOperationalDetail(string? operationId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Get an operation detail.");

        var operationDetails =
            await _route53Wrapper.GetOperationDetail(operationId);

        Console.WriteLine(operationDetails);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Optionally, get details for a registered domain.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> GetDomainDetails()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Get details on a domain.");

        Console.WriteLine($"\tNote: you must have a registered domain to get
details.");
        Console.WriteLine($"\tWould you like to get domain details? (y/n)");
        var ynResponse = Console.ReadLine();
        if (ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase))
        {
```

```
            string? domainName = null;
            while (domainName == null)
            {
                Console.WriteLine($"\tEnter a domain name to get details.");
                domainName = Console.ReadLine();
            }

            var domainDetails = await _route53Wrapper.GetDomainDetail(domainName);
            Console.WriteLine(domainDetails);
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }
}
```

Os métodos de wrapper usados pelo cenário para as ações do registro de domínios do Route 53.

```
public class Route53Wrapper
{
    private readonly IAmazonRoute53Domains _amazonRoute53Domains;
    private readonly ILogger<Route53Wrapper> _logger;
    public Route53Wrapper(IAmazonRoute53Domains amazonRoute53Domains,
 ILogger<Route53Wrapper> logger)
    {
        _amazonRoute53Domains = amazonRoute53Domains;
        _logger = logger;
    }


    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginators.ListPrices(new
 ListPricesRequest());
        // Get the entire list using the paginator.
```

```
        await foreach (var prices in paginatePrices.Prices)
        {
            results.Add(prices);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }


    /// <summary>
    /// Check the availability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for availability.</param>
    /// <returns>An availability result string.</returns>
    public async Task<string> CheckDomainAvailability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
            new CheckDomainAvailabilityRequest
            {
                DomainName = domain
            }
        );
        return result.Availability.Value;
    }


    /// <summary>
    /// Check the transferability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for transferability.</param>
    /// <returns>A transferability result string.</returns>
    public async Task<string> CheckDomainTransferability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
            new CheckDomainTransferabilityRequest
            {
                DomainName = domain
            }
        );
        return result.Transferability.Transferable.Value;
    }


    /// <summary>
    /// Get a list of suggestions for a given domain.
```

```
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
    /// <param name="onlyAvailable">If true, only returns available domains.</param>
    /// <param name="suggestionCount">The number of suggestions to return. Defaults
 to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
 bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
    }


    /// <summary>
    /// Get details for a domain action operation.
    /// </summary>
    /// <param name="operationId">The operational Id.</param>
    /// <returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
    {
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
                        OperationId = operationId
                    }
                );

            var details = $"\tOperation {operationId}:\n" +
                          $"\tFor domain {operationDetails.DomainName} on
 {operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                          $"\tMessage is {operationDetails.Message}.\n" +
```

```
                            $"\tStatus is {operationDetails.Status}.\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
    }


    /// <summary>
    /// Initiate a domain registration request.
    /// </summary>
    /// <param name="contact">Contact details.</param>
    /// <param name="domainName">The domain name to register.</param>
    /// <param name="autoRenew">True if the domain should automatically renew.</
param>
    /// <param name="duration">The duration in years for the domain registration.</
param>
    /// <returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew, int
 duration, ContactDetail contact)
    {
        // This example uses the same contact information for admin, registrant, and
 tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
```

```
        catch (InvalidInputException)
        {
            _logger.LogInformation($"Unable to request registration for domain
{domainName}");
            return null;
        }
    }


    /// <summary>
    /// View billing records for the account between a start and end date.
    /// </summary>
    /// <param name="startDate">The start date for billing results.</param>
    /// <param name="endDate">The end date for billing results.</param>
    /// <returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate, DateTime
endDate)
    {
        var results = new List<BillingRecord>();
        var paginateBilling = _amazonRoute53Domains.Paginators.ViewBilling(
            new ViewBillingRequest()
            {
                Start = startDate,
                End = endDate
            });

        // Get the entire list using the paginator.
        await foreach (var billingRecords in paginateBilling.BillingRecords)
        {
            results.Add(billingRecords);
        }
        return results;
    }


    /// <summary>
    /// List the domains for the account.
    /// </summary>
    /// <returns>A collection of domain summary records.</returns>
    public async Task<List<DomainSummary>> ListDomains()
    {
        var results = new List<DomainSummary>();
        var paginateDomains = _amazonRoute53Domains.Paginators.ListDomains(
            new ListDomainsRequest());
```

```
        // Get the entire list using the paginator.
        await foreach (var domain in paginateDomains.Domains)
        {
            results.Add(domain);
        }
        return results;
    }


    /// <summary>
    /// List operations for the account that are submitted after a specified date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime
submittedSince)
    {
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginators.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });

        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
        {
            results.Add(operations);
        }
        return results;
    }


    /// <summary>
    /// Get details for a domain.
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
    {
        try
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
```

```
                    DomainName = domainName
                });
            var details = $"\tDomain {domainName}:\n" +
                          $"\tCreated on {result.CreationDate.ToShortDateString()}.
\n" +
                          $"\tAdmin contact is {result.AdminContact.Email}.\n" +
                          $"\tAuto-renew is {result.AutoRenew}.\n";

            return details;
        }
        catch (InvalidInputException)
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Ações

`CheckDomainAvailability`

O código de exemplo a seguir mostra como usar `CheckDomainAvailability`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Check the availability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for availability.</param>
/// <returns>An availability result string.</returns>
public async Task<string> CheckDomainAvailability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
        new CheckDomainAvailabilityRequest
        {
            DomainName = domain
        }
    );
    return result.Availability.Value;
}
```

- Para obter detalhes da API, consulte CheckDomainAvailability em Referência AWS SDK para .NET da API.

`CheckDomainTransferability`

O código de exemplo a seguir mostra como usar `CheckDomainTransferability`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Check the transferability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for transferability.</param>
/// <returns>A transferability result string.</returns>
public async Task<string> CheckDomainTransferability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
        new CheckDomainTransferabilityRequest
        {
            DomainName = domain
        }
    );
    return result.Transferability.Transferable.Value;
}
```

- Para obter detalhes da API, consulte CheckDomainTransferability em Referência AWS SDK para .NET da API.

`GetDomainDetail`

O código de exemplo a seguir mostra como usar `GetDomainDetail`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get details for a domain.
/// </summary>
/// <returns>A string with detail information about the domain.</returns>
public async Task<string> GetDomainDetail(string domainName)
{
    try
```

```
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
                    DomainName = domainName
                });
            var details = $"\tDomain {domainName}:\n" +
                          $"\tCreated on {result.CreationDate.ToShortDateString()}.
\n" +
                          $"\tAdmin contact is {result.AdminContact.Email}.\n" +
                          $"\tAuto-renew is {result.AutoRenew}.\n";

            return details;
        }
        catch (InvalidInputException)
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
```

- Para obter detalhes da API, consulte GetDomainDetail em Referência AWS SDK para .NET da API.

`GetDomainSuggestions`

O código de exemplo a seguir mostra como usar `GetDomainSuggestions`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of suggestions for a given domain.
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
```

```
    /// <param name="onlyAvailable">If true, only returns available domains.</param>
    /// <param name="suggestionCount">The number of suggestions to return. Defaults
to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
    }
```

- Para obter detalhes da API, consulte GetDomainSuggestions em Referência AWS SDK para .NET da API.

`GetOperationDetail`

O código de exemplo a seguir mostra como usar `GetOperationDetail`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get details for a domain action operation.
    /// </summary>
    /// <param name="operationId">The operational Id.</param>
    /// <returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
```

```
    {
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
                        OperationId = operationId
                    }
                );

            var details = $"\tOperation {operationId}:\n" +
                          $"\tFor domain {operationDetails.DomainName} on
{operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                          $"\tMessage is {operationDetails.Message}.\n" +
                          $"\tStatus is {operationDetails.Status}.\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
    }
```

- Para obter detalhes da API, consulte GetOperationDetail em Referência AWS SDK para .NET da API.

`ListDomains`

O código de exemplo a seguir mostra como usar `ListDomains`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the domains for the account.
    /// </summary>
    /// <returns>A collection of domain summary records.</returns>
    public async Task<List<DomainSummary>> ListDomains()
    {
        var results = new List<DomainSummary>();
        var paginateDomains = _amazonRoute53Domains.Paginators.ListDomains(
            new ListDomainsRequest());

        // Get the entire list using the paginator.
        await foreach (var domain in paginateDomains.Domains)
        {
            results.Add(domain);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte ListDomains em Referência AWS SDK para .NET da API.

`ListOperations`

O código de exemplo a seguir mostra como usar `ListOperations`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List operations for the account that are submitted after a specified date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
```

```
    public async Task<List<OperationSummary>> ListOperations(DateTime
submittedSince)
    {
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginators.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });

        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
        {
            results.Add(operations);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte ListOperations em Referência AWS SDK para .NET da API.

`ListPrices`

O código de exemplo a seguir mostra como usar `ListPrices`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
```

```
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginators.ListPrices(new
 ListPricesRequest());
        // Get the entire list using the paginator.
        await foreach (var prices in paginatePrices.Prices)
        {
            results.Add(prices);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }
```

- Para obter detalhes da API, consulte ListPrices em Referência AWS SDK para .NET da API.

`RegisterDomain`

O código de exemplo a seguir mostra como usar `RegisterDomain`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Initiate a domain registration request.
    /// </summary>
    /// <param name="contact">Contact details.</param>
    /// <param name="domainName">The domain name to register.</param>
    /// <param name="autoRenew">True if the domain should automatically renew.</
param>
    /// <param name="duration">The duration in years for the domain registration.</
param>
    /// <returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew, int
 duration, ContactDetail contact)
    {
```

```
        // This example uses the same contact information for admin, registrant, and
tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
        catch (InvalidInputException)
        {
            _logger.LogInformation($"Unable to request registration for domain
{domainName}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte RegisterDomain em Referência AWS SDK para .NET da API.

ViewBilling

O código de exemplo a seguir mostra como usar `ViewBilling`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// View billing records for the account between a start and end date.
    /// </summary>
    /// <param name="startDate">The start date for billing results.</param>
    /// <param name="endDate">The end date for billing results.</param>
    /// <returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate, DateTime
endDate)
    {
        var results = new List<BillingRecord>();
        var paginateBilling = _amazonRoute53Domains.Paginators.ViewBilling(
            new ViewBillingRequest()
            {
                Start = startDate,
                End = endDate
            });

        // Get the entire list using the paginator.
        await foreach (var billingRecords in paginateBilling.BillingRecords)
        {
            results.Add(billingRecords);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte ViewBilling em Referência AWS SDK para .NET da API.

Exemplos do Amazon S3 usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon S3.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários
- Exemplos sem servidor

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie um objeto usando uma solicitação condicional.

```
    /// <summary>
    /// Copies an object from one Amazon S3 bucket to another with a conditional
request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
```

```
    /// <param name="conditionType">The type of condition to apply, e.g.
 'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
 'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string destKey,
 string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null, string?
 etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
            }
```

```
            await _amazonS3.CopyObjectAsync(copyObjectRequest);
            _logger.LogInformation($"Conditional copy successful for key {destKey}
in bucket {destBucket}.");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional copy failed: Precondition failed");
            }
            else if (e.ErrorCode == "304")
            {
                _logger.LogError("Conditional copy failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK para .NET da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket com o bloqueio de objetos habilitado.

```
    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the bucket.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
 {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK para .NET da API.

`DeleteBucketCors`

O código de exemplo a seguir mostra como usar `DeleteBucketCors`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Deletes a CORS configuration from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to delete the CORS configuration from the bucket.</param>
        private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
        {
            DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            await client.DeleteCORSConfigurationAsync(request);
        }
```

- Para obter detalhes da API, consulte DeleteBucketCors em Referência AWS SDK para .NET da API.

`DeleteBucketLifecycle`

O código de exemplo a seguir mostra como usar `DeleteBucketLifecycle`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// This method removes the Lifecycle configuration from the named
        /// S3 bucket.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the RemoveLifecycleConfigAsync method.</param>
        /// <param name="bucketName">A string representing the name of the
        /// S3 bucket from which the configuration will be removed.</param>
        public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client, string
bucketName)
        {
            var request = new DeleteLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
```

- Para obter detalhes da API, consulte DeleteBucketLifecycle em Referência AWS SDK para .NET da API.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um objeto de um bucket do S3 sem versionamento.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
```

```
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete an object from a non-versioned Amazon
    /// Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteObject
    {
        /// <summary>
        /// The Main method initializes the necessary variables and then calls
        /// the DeleteObjectNonVersionedBucketAsync method to delete the object
        /// named by the keyName parameter.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";
            const string keyName = "testfile.txt";

            // If the Amazon S3 bucket is located in an AWS Region other than the
            // Region of the default account, define the AWS Region for the
            // Amazon S3 bucket in your call to the AmazonS3Client constructor.
            // For example RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();
            await DeleteObjectNonVersionedBucketAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
        /// desired object from the named bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to delete
        /// an object from an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// object will be deleted.</param>
        /// <param name="keyName">The name of the object to delete.</param>
        public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
```

```
                };

                Console.WriteLine($"Deleting object: {keyName}");
                await client.DeleteObjectAsync(deleteObjectRequest);
                Console.WriteLine($"Object: {keyName} deleted from {bucketName}.");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' when deleting an object.");
            }
        }
    }
```

Exclua um objeto de um bucket do S3 com versionamento.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example creates an object in an Amazon Simple Storage Service
    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
```

```
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3 client,
string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName, keyName);

                // Delete the object by specifying an object key and a version ID.
                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
object
        /// will be created.</param>
```

```
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
        public static async Task<string> PutAnObject(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!",
            };

            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObject em Referência AWS SDK para .NET da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua vários objetos de um bucket do S3 sem versionamento.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;
```

```
    /// <summary>
    /// This example shows how to delete multiple objects from an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        /// <summary>
        /// The Main method initializes the Amazon S3 client and the name of
        /// the bucket and then passes those values to MultiObjectDeleteAsync.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";

            // If the Amazon S3 bucket from which you wish to delete objects is not
            // located in the same AWS Region as the default user, define the
            // AWS Region for the Amazon S3 bucket as a parameter to the client
            // constructor.
            IAmazonS3 s3Client = new AmazonS3Client();

            await MultiObjectDeleteAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method uses the passed Amazon S3 client to first create and then
        /// delete three files from the named bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// Amazon S3 methods.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where objects
        /// will be created and then deleted.</param>
        public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
        {
            // Create three sample objects which we will then delete.
            var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

            // Now perform the multi-object delete, passing the key names and
            // version IDs. Since we are working with a non-versioned bucket,
            // the object keys collection includes null version IDs.
            DeleteObjectsRequest multiObjectDeleteRequest = new DeleteObjectsRequest
            {
```

```
                BucketName = bucketName,
                Objects = keysAndVersions,
            };

            // You can add a specific object key to the delete request using the
            // AddKey method of the multiObjectDeleteRequest.
            try
            {
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
        }

        /// <summary>
        /// Prints the list of errors raised by the call to DeleteObjectsAsync.
        /// </summary>
        /// <param name="ex">A collection of exceptions returned by the call to
        /// DeleteObjectsAsync.</param>
        public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
        {
            DeleteObjectsResponse errorResponse = ex.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

            Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// This method creates simple text file objects that can be used in
```

```
        /// the delete method.
        /// </summary>
        /// <param name="client">The Amazon S3 client used to call PutObjectAsync.</
param>
        /// <param name="number">The number of objects to create.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created.</param>
        /// <returns>A list of keys (object keys) and versions that the calling
        /// method will use to delete the newly created files.</returns>
        public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3 client,
 int number, string bucketName)
        {
            List<KeyVersion> keys = new List<KeyVersion>();
            for (int i = 0; i < number; i++)
            {
                string key = "ExampleObject-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                PutObjectResponse response = await client.PutObjectAsync(request);

                // For non-versioned bucket operations, we only need the
                // object key.
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                };
                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

Exclua vários objetos de um bucket do S3 com versionamento.

```
    using System;
```

```
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete objects in a version-enabled Amazon
    /// Simple StorageService (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";

            // If the AWS Region for your Amazon S3 bucket is different from
            // the AWS Region of the default user, define the AWS Region for
            // the Amazon S3 bucket and pass it to the client constructor
            // like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 s3Client;

            s3Client = new AmazonS3Client();
            await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
        }

        /// <summary>
        /// This method removes multiple versions and objects from a
        /// version-enabled Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string bucketName)
        {
            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync(client, bucketName);

            // Delete objects (without specifying object version in the request).
```

```
            var deletedObjects = await DeleteObjectsAsync(client, bucketName);

            // Additional exercise - remove the delete markers Amazon S3 returned
from
            // the preceding response. This results in the objects reappearing
            // in the bucket (you can verify the appearance/disappearance of
            // objects in the console).
            await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
        }

        /// <summary>
        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects deleted.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>> DeleteObjectsAsync(IAmazonS3
client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker and
            // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }

        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client, string
bucketName)
        {
            // Upload the sample objects.
```

```
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }

        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
            Console.WriteLine($"No. of objects successfully deleted =
 {errorResponse.DeletedObjects.Count}");
            Console.WriteLine($"No. of objects failed to delete =
 {errorResponse.DeleteErrors.Count}");
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
 {deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// Delete multiple objects from a version-enabled bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        private static async Task VersionedDeleteAsync(IAmazonS3 client, string
 bucketName, List<KeyVersion> keys)
        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keys, // This includes the object keys and specific
 version IDs.
```

```
            };

            try
            {
                Console.WriteLine("Executing VersionedDelete...");
                DeleteObjectsResponse response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
 {response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
        private static async Task<List<DeletedObject>>
 NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion> keys)
        {
            // Create a request that includes only the object key names.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)
            {
                multiObjectDeleteRequest.AddKey(key.Key);
            }

            // Execute DeleteObjectsAsync.
            // The DeleteObjectsAsync method adds a delete marker for each
            // object deleted. You can verify that the objects were removed
```

```
            // using the Amazon S3 console.
            DeleteObjectsResponse response;
            try
            {
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
 response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
                throw; // Some deletions failed. Investigate before continuing.
            }

            // This response contains the DeletedObjects list which we use to delete
 the delete markers.
            return response.DeletedObjects;
        }

        /// <summary>
        /// Deletes the markers left after deleting the temporary objects.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="deletedObjects">A list of the objects that were deleted.</
param>
        private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client, string
 bucketName, List<DeletedObject> deletedObjects)
        {
            var keyVersionList = new List<KeyVersion>();

            foreach (var deletedObject in deletedObjects)
            {
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = deletedObject.Key,
                    VersionId = deletedObject.DeleteMarkerVersionId,
                };
```

```
                keyVersionList.Add(keyVersion);
            }

            // Create another request to delete the delete markers.
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keyVersionList,
            };

            // Now, delete the delete marker to bring your objects back to the
bucket.
            try
            {
                Console.WriteLine("Removing the delete markers .....");
                var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Create temporary Amazon S3 objects to show how object deletion wors in
an
        /// Amazon S3 bucket with versioning enabled.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PutObjectAsync to create temporary objects for the example.</param>
        /// <param name="bucketName">A string representing the name of the S3
        /// bucket where we will create the temporary objects.</param>
        /// <param name="number">The number of temporary objects to create.</param>
        /// <returns>A list of the KeyVersion objects.</returns>
        private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, string bucketName, int number)
        {
            var keys = new List<KeyVersion>();

            for (var i = 0; i < number; i++)
```

```
            {
                string key = "ObjectToDelete-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                var response = await client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    VersionId = response.VersionId,
                };

                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK para .NET da API.

`GetBucketAcl`

O código de exemplo a seguir mostra como usar `GetBucketAcl`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Get the access control list (ACL) for the new bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to get the
        /// access control list (ACL) of the bucket.</param>
        /// <param name="newBucketName">The name of the newly created bucket.</
param>
        /// <returns>An S3AccessControlList.</returns>
        public static async Task<S3AccessControlList> GetACLForBucketAsync(IAmazonS3
 client, string newBucketName)
        {
            // Retrieve bucket ACL to show that the ACL was properly applied to
            // the new bucket.
            GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = newBucketName,
            });

            return getACLResponse.AccessControlList;
        }
```

- Para obter detalhes da API, consulte GetBucketAcl em Referência AWS SDK para .NET da API.

`GetBucketCors`

O código de exemplo a seguir mostra como usar `GetBucketCors`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
```

```
        /// Retrieve the CORS configuration applied to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to retrieve the CORS configuration.</param>
        /// <returns>The created CORS configuration object.</returns>
        private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
        {
            GetCORSConfigurationRequest request = new GetCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            var response = await client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }
```

- Para obter detalhes da API, consulte GetBucketCors em Referência AWS SDK para .NET da API.

`GetBucketEncryption`

O código de exemplo a seguir mostra como usar `GetBucketEncryption`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get and print the encryption settings of a bucket.
    /// </summary>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task GetEncryptionSettings(string bucketName)
```

```
    {
        // Check and print the bucket encryption settings.
        Console.WriteLine($"Getting encryption settings for bucket {bucketName}.");

        try
        {
            var settings =
                await _s3Client.GetBucketEncryptionAsync(
                    new GetBucketEncryptionRequest() { BucketName = bucketName });

            foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
            {
                Console.WriteLine(
                    $"\tAlgorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
                Console.WriteLine(
                    $"\tKey:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementServiceK
            }
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                ? $"Bucket {bucketName} was not found."
                : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
        }
    }
```

- Para obter detalhes da API, consulte GetBucketEncryption em Referência AWS SDK para .NET da API.

`GetBucketLifecycleConfiguration`

O código de exemplo a seguir mostra como usar `GetBucketLifecycleConfiguration`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Returns a configuration object for the supplied bucket name.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the GetLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">The name of the S3 bucket for which a
        /// configuration will be created.</param>
        /// <returns>Returns a new LifecycleConfiguration object.</returns>
        public static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
        {
            var request = new GetLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }
```

- Para obter detalhes da API, consulte GetBucketLifecycleConfiguration em Referência AWS SDK para .NET da API.

`GetBucketWebsite`

O código de exemplo a seguir mostra como usar `GetBucketWebsite`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
                // Get the website configuration.
                GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
                Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- Para obter detalhes da API, consulte GetBucketWebsite em Referência AWS SDK para .NET da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um objeto usando uma solicitação condicional.

```
    /// <summary>
```

```
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch', 'IfNoneMatch',
 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
 sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null, string?
 etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
```

```
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read successful. Here are the first
20 bytes of the object:\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional read failed: Precondition failed");
            }
            else if (e.ErrorCode == "NotModified")
            {
                _logger.LogError("Conditional read failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK para .NET da API.

`GetObjectLegalHold`

O código de exemplo a seguir mostra como usar `GetObjectLegalHold`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in {bucketName}:
" +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- Para obter detalhes da API, consulte GetObjectLegalHold em Referência AWS SDK para .NET da API.

`GetObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `GetObjectLockConfiguration`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Get the object lock configuration details for an S3 bucket.
   /// </summary>
   /// <param name="bucketName">The bucket to get details.</param>
   /// <returns>The bucket's object lock configuration details.</returns>
   public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
   {
       try
       {
           var request = new GetObjectLockConfigurationRequest()
           {
               BucketName = bucketName
           };

           var response = await _amazonS3.GetObjectLockConfigurationAsync(request);
           Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                             $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                             $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

           return response.ObjectLockConfiguration;
       }
       catch (AmazonS3Exception ex)
       {
           Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
           return new ObjectLockConfiguration();
       }
   }
```

- Para obter detalhes da API, consulte GetObjectLockConfiguration em Referência AWS SDK para .NET da API.

GetObjectRetention

O código de exemplo a seguir mostra como usar `GetObjectRetention`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in {bucketName}:
" +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"\tUnable to fetch object lock retention:
 '{ex.Message}'");
            return new ObjectLockRetention();
        }
    }
```

- Para obter detalhes da API, consulte GetObjectRetention em Referência AWS SDK para .NET da API.

ListBuckets

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
```

```
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3 client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
        /// <param name="bucketList">A List of S3 bucket objects.</param>
        public static void DisplayBucketList(List<S3Bucket> bucketList)
        {
            bucketList
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
 created on: {b.CreationDate}"));
        }

        public static async Task Main()
        {
            // The client uses the AWS Region of the default user.
            // If the Region where the buckets were created is different,
            // pass the Region to the client constructor. For example:
            // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
            _s3Client = new AmazonS3Client();
            var response = await GetBuckets(_s3Client);
            DisplayBucketList(response.Buckets);
        }
    }
}
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para .NET da API.

`ListObjectVersions`

O código de exemplo a seguir mostra como usar `ListObjectVersions`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example lists the versions of the objects in a version enabled
    /// Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectVersions
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";

            // If the AWS Region where your bucket is defined is different from
            // the AWS Region where the Amazon S3 bucket is defined, pass the
 constant
            // for the AWS Region to the client constructor like this:
            //      var client = new AmazonS3Client(RegionEndpoint.USWest2);
            IAmazonS3 client = new AmazonS3Client();
            await GetObjectListWithAllVersionsAsync(client, bucketName);
        }

        /// <summary>
        /// This method lists all versions of the objects within an Amazon S3
        /// version enabled bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// ListVersionsAsync.</param>
        /// <param name="bucketName">The name of the version enabled Amazon S3
 bucket
        /// for which you want to list the versions of the contained objects.</
param>
```

```
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3 client,
string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    MaxKeys = 2,
                };

                do
                {
                    ListVersionsResponse response = await
client.ListVersionsAsync(request);

                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine($"key: {entry.Key} size: {entry.Size}");
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                }
                while (request != null);
            }
            catch (AmazonS3Exception ex)
            {
```

```
                Console.WriteLine($"Error: '{ex.Message}'");
            }
        }
    }
```

- Para obter detalhes da API, consulte ListObjectVersions em Referência AWS SDK para .NET da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste objetos com um paginador.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// The following example lists objects in an Amazon Simple Storage
/// Service (Amazon S3) bucket.
/// </summary>
public class ListObjectsPaginator
{
    private const string BucketName = "amzn-s3-demo-bucket";

    public static async Task Main()
    {
        IAmazonS3 s3Client = new AmazonS3Client();
```

```
            Console.WriteLine($"Listing the objects contained in {BucketName}:\n");
            await ListingObjectsAsync(s3Client, BucketName);
        }

        /// <summary>
        /// This method uses a paginator to retrieve the list of objects in an
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the S3 bucket whose objects
        /// you want to list.</param>
        public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
        {
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK para .NET API.

`PutBucketAccelerateConfiguration`

O código de exemplo a seguir mostra como usar `PutBucketAccelerateConfiguration`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "amzn-s3-demo-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which the
    /// method will be enabling acceleration.</param>
```

```
        private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
        {
            try
            {
                var putRequest = new PutBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                    AccelerateConfiguration = new AccelerateConfiguration
                    {
                        Status = BucketAccelerateStatus.Enabled,
                    },
                };
                await client.PutBucketAccelerateConfigurationAsync(putRequest);

                var getRequest = new GetBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                };
                var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

                Console.WriteLine($"Acceleration state = '{response.Status}' ");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
            }
        }
    }
```

- Para obter detalhes da API, consulte PutBucketAccelerateConfiguration em Referência AWS SDK para .NET da API.

`PutBucketAcl`

O código de exemplo a seguir mostra como usar `PutBucketAcl`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Creates an Amazon S3 bucket with an ACL to control access to the
        /// bucket and the objects stored in it.
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// an Amazon S3 bucket, with an ACL applied to the bucket.
        /// </param>
        /// <param name="region">The AWS Region where the bucket will be created.</
param>
        /// <param name="newBucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value indicating success or failure.</returns>
        public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
 client, S3Region region, string newBucketName)
        {
            try
            {
                // Create a new Amazon S3 bucket with Canned ACL.
                var putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = region,
                    CannedACL = S3CannedACL.LogDeliveryWrite,
                };

                PutBucketResponse putBucketResponse = await
 client.PutBucketAsync(putBucketRequest);

                return putBucketResponse.HttpStatusCode ==
 System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Amazon S3 error: {ex.Message}");
```

```
            }

            return false;
        }
```

- Para obter detalhes da API, consulte PutBucketAcl em Referência AWS SDK para .NET da API.

`PutBucketCors`

O código de exemplo a seguir mostra como usar `PutBucketCors`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Add CORS configuration to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to apply the CORS configuration to an Amazon S3 bucket.</param>
        /// <param name="configuration">The CORS configuration to apply.</param>
        private static async Task PutCORSConfigurationAsync(AmazonS3Client client,
CORSConfiguration configuration)
        {
            PutCORSConfigurationRequest request = new PutCORSConfigurationRequest()
            {
                BucketName = BucketName,
                Configuration = configuration,
            };

            _ = await client.PutCORSConfigurationAsync(request);
        }
```

- Para obter detalhes da API, consulte PutBucketCors em Referência AWS SDK para .NET da API.

`PutBucketEncryption`

O código de exemplo a seguir mostra como usar `PutBucketEncryption`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Set the bucket server side encryption to use AWSKMS with a customer-managed
key id.
   /// </summary>
   /// <param name="bucketName">Name of the bucket.</param>
   /// <param name="kmsKeyId">The Id of the KMS Key.</param>
   /// <returns>True if successful.</returns>
   public static async Task<bool> SetBucketServerSideEncryption(string bucketName,
string kmsKeyId)
   {
       var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
       {
           ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
           {
               new ServerSideEncryptionRule
               {
                   ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
                   {
                       ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
                       ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
                   }
               }
```

```
            }
        };
        try
        {
            var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
            {
                BucketName = bucketName,
                ServerSideEncryptionConfiguration = serverSideEncryptionByDefault,
            });

            return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "AccessDenied"
                ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
                : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
        }
        return false;
    }
```

- Para obter detalhes da API, consulte PutBucketEncryption em Referência AWS SDK para .NET da API.

`PutBucketLifecycleConfiguration`

O código de exemplo a seguir mostra como usar `PutBucketLifecycleConfiguration`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        /// <summary>
        /// Adds lifecycle configuration information to the S3 bucket named in
        /// the bucketName parameter.
        /// </summary>
        /// <param name="client">The S3 client used to call the
        /// PutLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">A string representing the S3 bucket to
        /// which configuration information will be added.</param>
        /// <param name="configuration">A LifecycleConfiguration object that
        /// will be applied to the S3 bucket.</param>
        public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
 string bucketName, LifecycleConfiguration configuration)
        {
            var request = new PutLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
                Configuration = configuration,
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }
```

- Para obter detalhes da API, consulte PutBucketLifecycleConfiguration em Referência AWS SDK para .NET da API.

`PutBucketLogging`

O código de exemplo a seguir mostra como usar `PutBucketLogging`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
```

```
    using Amazon.S3;
    using Amazon.S3.Model;
    using Microsoft.Extensions.Configuration;

    /// <summary>
    /// This example shows how to enable logging on an Amazon Simple Storage
    /// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
    /// this example. The first is the bucket for which you wish to enable
    /// logging, and the second is the location where you want to store the
    /// logs.
    /// </summary>
    public class ServerAccessLogging
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of logs
to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
```

```
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the source
bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" + logPrefix +
@"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
```

```
                                ""ArnLike"": { ""aws:SourceArn"": ""arn:aws:s3:::" +
sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be used
        /// to configure and apply logging to the selected Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
```

```
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load local
settings.
                .Build();
        }
    }
```

- Para obter detalhes da API, consulte PutBucketLogging em Referência AWS SDK para .NET da API.

`PutBucketNotificationConfiguration`

O código de exemplo a seguir mostra como usar `PutBucketNotificationConfiguration`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to enable notifications for an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class EnableNotifications
    {
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket1";
            const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
            const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

            IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
            await EnableNotificationAsync(client, bucketName, snsTopic, sqsQueue);
        }

        /// <summary>
        /// This method makes the call to the PutBucketNotificationAsync method.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to call
        /// the PutBucketNotificationAsync method.</param>
        /// <param name="bucketName">The name of the bucket for which
        /// notifications will be turned on.</param>
        /// <param name="snsTopic">The ARN for the Amazon Simple Notification
        /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
        /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
```

```
        /// (Amazon SQS) queue to which notifications will be pushed.</param>
        public static async Task EnableNotificationAsync(
            IAmazonS3 client,
            string bucketName,
            string snsTopic,
            string sqsQueue)
        {
            try
            {
                // The bucket for which we are setting up notifications.
                var request = new PutBucketNotificationRequest()
                {
                    BucketName = bucketName,
                };

                // Defines the topic to use when sending a notification.
                var topicConfig = new TopicConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic,
                };
                request.TopicConfigurations = new List<TopicConfiguration>
                {
                    topicConfig,
                };
                request.QueueConfigurations = new List<QueueConfiguration>
                {
                    new QueueConfiguration()
                    {
                        Events = new List<EventType> { EventType.ObjectCreatedPut },
                        Queue = sqsQueue,
                    },
                };

                // Now apply the notification settings to the bucket.
                PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- Para obter detalhes da API, consulte PutBucketNotificationConfiguration em Referência AWS SDK para .NET da API.

PutBucketWebsite

O código de exemplo a seguir mostra como usar `PutBucketWebsite`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
                // Put the website configuration.
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument,
                    },
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- Para obter detalhes da API, consulte PutBucketWebsite em Referência AWS SDK para .NET da API.

PutObject

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de um objeto com criptografia do lado do servidor.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket with server-side encryption enabled.
/// </summary>
public class ServerSideEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        await WritingAnObjectAsync(client, bucketName, keyName);
    }

    /// <summary>
    /// Upload a sample object include a setting for encryption.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// to upload a file and apply server-side encryption.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// encrypted object will reside.</param>
    /// <param name="keyName">The name for the object that you want to
```

```
        /// create in the supplied bucket.</param>
        public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256,
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

                Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}' when writing an object");
            }
        }
    }
```

Coloque um objeto usando uma solicitação condicional.

```
    /// <summary>
```

```
    /// Uploads an object to Amazon S3 with a conditional request. Prevents
 overwrite using an IfNoneMatch condition for the object key.
    /// </summary>
    /// <param name="objectKey">The key of the object to upload.</param>
    /// <param name="bucket">The source bucket of the object.</param>
    /// <param name="content">The content to upload as a string.</param>
    /// <returns>The ETag if the conditional write is successful, empty otherwise.</
returns>
    public async Task<string> PutObjectConditional(string objectKey, string bucket,
 string content)
    {
        try
        {
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
 {objectKey} in bucket {bucket}.");
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK para .NET da API.

`PutObjectLegalHold`

O código de exemplo a seguir mostra como usar `PutObjectLegalHold`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
```

```
        }
    }
```

- Para obter detalhes da API, consulte PutObjectLegalHold em Referência AWS SDK para .NET da API.

`PutObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `PutObjectLockConfiguration`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de Bloqueio de Objetos de um bucket.

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });
```

```
            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

Defina o período de retenção padrão de um bucket.

```
    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
```

```
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in days
or years but not both.
                        }
                    }
                }
            };

            var response = await _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte PutObjectLockConfiguration em Referência AWS SDK para .NET da API.

`PutObjectRetention`

O código de exemplo a seguir mostra como usar `PutObjectRetention`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte PutObjectRetention em Referência AWS SDK para .NET da API.

`RestoreObject`

O código de exemplo a seguir mostra como usar `RestoreObject`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
```

```
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            IAmazonS3 client = new AmazonS3Client(bucketRegion);
            RestoreObjectAsync(client, bucketName, objectKey).Wait();
        }

        /// <summary>
        /// This method restores an archived object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// RestoreObjectAsync.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object was located before it was archived.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object to restore.</param>
        public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
        {
            try
            {
                var restoreRequest = new RestoreObjectRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Days = 2,
                };
                RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Error: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// This method retrieves the status of the object's restoration.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
```

```
        /// GetObjectMetadataAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
        /// S3 bucket which contains the archived object.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object you want to restore.</param>
        public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
        {
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
            };

            GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

            var restStatus = response.RestoreInProgress ? "in-progress" : "finished
or failed";
            Console.WriteLine($"Restoration status: {restStatus}");
        }
    }
```

- Para obter detalhes da API, consulte RestoreObject em Referência AWS SDK para .NET da API.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere um URL pré-assinado que possa executar uma ação do Amazon S3 por tempo limitado.

```
    using System;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    public class GenPresignedUrl
    {
        public static void Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";
            const string objectKey = "sample.txt";

            // Specify how long the presigned URL lasts, in hours
            const double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

            string urlString = GeneratePresignedURL(s3Client, bucketName, objectKey,
 timeoutDuration);
```

```
            Console.WriteLine($"The generated URL is: {urlString}.");
        }

        /// <summary>
        /// Generate a presigned URL that can be used to access the file named
        /// in the objectKey parameter for the amount of time specified in the
        /// duration parameter.
        /// </summary>
        /// <param name="client">An initialized S3 client object used to call
        /// the GetPresignedUrl method.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// object for which to create the presigned URL.</param>
        /// <param name="objectKey">The name of the object to access with the
        /// presigned URL.</param>
        /// <param name="duration">The length of time for which the presigned
        /// URL will be valid.</param>
        /// <returns>A string representing the generated presigned URL.</returns>
        public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)
        {
            string urlString = string.Empty;
            try
            {
                var request = new GetPreSignedUrlRequest()
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration),
                };
                urlString = client.GetPreSignedURL(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:'{ex.Message}'");
            }

            return urlString;
        }
    }
```

Gere um URL pré-assinado e faça um upload usando esse URL.

```
    using System;
    using System.IO;
    using System.Net.Http;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket using a presigned URL. The code first
    /// creates a presigned URL and then uses it to upload an object to an
    /// Amazon S3 bucket using that URL.
    /// </summary>
    public class UploadUsingPresignedURL
    {
        private static HttpClient httpClient = new HttpClient();

        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "samplefile.txt";
            string filePath = $"source\\{keyName}";

            // Specify how long the signed URL will be valid in hours.
            double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);
```

```
            var url = GeneratePreSignedURL(client, bucketName, keyName,
timeoutDuration);
            var success = await UploadObject(filePath, url);

            if (success)
            {
                Console.WriteLine("Upload succeeded.");
            }
            else
            {
                Console.WriteLine("Upload failed.");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket using the presigned URL passed
in
        /// the url parameter.
        /// </summary>
        /// <param name="filePath">The path (including file name) to the local
        /// file you want to upload.</param>
        /// <param name="url">The presigned URL that will be used to upload the
        /// file to the Amazon S3 bucket.</param>
        /// <returns>A Boolean value indicating the success or failure of the
        /// operation, based on the HttpWebResponse.</returns>
        public static async Task<bool> UploadObject(string filePath, string url)
        {
            using var streamContent = new StreamContent(
                new FileStream(filePath, FileMode.Open, FileAccess.Read));

            var response = await httpClient.PutAsync(url, streamContent);
            return response.IsSuccessStatusCode;
        }

        /// <summary>
        /// Generates a presigned URL which will be used to upload an object to
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetPreSignedURL.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which the
        /// presigned URL will point.</param>
```

```
        /// <param name="objectKey">The name of the file that will be uploaded.</
param>
        /// <param name="duration">How long (in hours) the presigned URL will
        /// be valid.</param>
        /// <returns>The generated URL.</returns>
        public static string GeneratePreSignedURL(
            IAmazonS3 client,
            string bucketName,
            string objectKey,
            double duration)
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Verb = HttpVerb.PUT,
                Expires = DateTime.UtcNow.AddHours(duration),
            };

            string url = client.GetPreSignedURL(request);
            return url;
        }
    }
```

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para .NET

Mostra como usar a API .NET do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Conceitos básicos de criptografia

O exemplo de código a seguir mostra como começar a usar a criptografia de objetos do Amazon S3.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Create an encryption key.
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            // Upload the object.
```

```
                PutObjectRequest putObjectRequest = await UploadObjectAsync(client,
 bucketName, keyName, base64Key);

                // Download the object and verify that its contents match what you
 uploaded.
                await DownloadObjectAsync(client, bucketName, keyName, base64Key,
 putObjectRequest);

                // Get object metadata and verify that the object uses AES-256
 encryption.
                await GetObjectMetadataAsync(client, bucketName, keyName,
 base64Key);

                // Copy both the source and target objects using server-side
 encryption with
                // an encryption key.
                await CopyObjectAsync(client, bucketName, keyName,
 copyTargetKeyName, aesEncryption, base64Key);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which the
        /// object will be uploaded.</param>
        /// <param name="keyName">The name of the object to upload to the Amazon S3
        /// bucket.</param>
        /// <param name="base64Key">The encryption key.</param>
        /// <returns>The PutObjectRequest object for use by DownloadObjectAsync.</
returns>
        public static async Task<PutObjectRequest> UploadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
```

```
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
            PutObjectResponse putObjectResponse = await
 client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }

        /// <summary>
        /// Downloads an encrypted object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
        /// <param name="base64Key">The encryption key used to encrypt the
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key,
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
 S3.
```

```
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same as
we set");
                }
                else
                {
                    Console.WriteLine("Error...Object encryption method is not the
same as AES256 we set");
                }
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with an Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to call GetObjectMetadataAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket containing the
        /// object for which we want to retrieve metadata.</param>
        /// <param name="keyName">The name of the object for which we wish to
        /// retrieve the metadata.</param>
        /// <param name="base64Key">The encryption key associated with the
```

```
        /// object.</param>
        public static async Task GetObjectMetadataAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used is:
{0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }

        /// <summary>
        /// Copies an encrypted object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// CopyObjectAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket containing the object
        /// to copy.</param>
        /// <param name="keyName">The name of the object to copy.</param>
        /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the object
        /// will be copied.</param>
        /// <param name="aesEncryption">The encryption type to use.</param>
        /// <param name="base64Key">The encryption key to use.</param>
        public static async Task CopyObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string copyTargetKeyName,
```

```
            Aes aesEncryption,
            string base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,

                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CopyObject
  - GetObject
  - GetObjectMetadata

Conceitos básicos de etiquetas

O exemplo de código a seguir mostra como começar a usar etiquetas de objetos do Amazon S3.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "newobject.txt";
        string filePath = @"*** file path ***";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        var client = new AmazonS3Client(bucketRegion);
        await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
    }

    /// <summary>
    /// This method uploads an object with tags. It then shows the tag
    /// values, changes the tags, and shows the new tags.
    /// </summary>
    /// <param name="client">The Initialized Amazon S3 client object used
    /// to call the methods to create and change an objects tags.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object will be stored.</param>
```

```
        /// <param name="keyName">A string representing the key name of the
        /// object to be tagged.</param>
        /// <param name="filePath">The directory location and file name of the
        /// object to be uploaded to the Amazon S3 bucket.</param>
        public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
        {
            try
            {
                // Create an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Keyx1", Value = "Value1" },
                        new Tag { Key = "Keyx2", Value = "Value2" },
                    },
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);

                // Now retrieve the new object's tags.
                GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

                // Display the tag values.
                objectTags.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

                Tagging newTagSet = new Tagging()
                {
                    TagSet = new List<Tag>
```

```
                    {
                        new Tag { Key = "Key3", Value = "Value3" },
                        new Tag { Key = "Key4", Value = "Value4" },
                    },
                };

                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet,
                };

                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // Retrieve the tags again and show the values.
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

                objectTags2.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine(
                        $"Error: '{ex.Message}'");
            }
        }
    }
```

- Para obter detalhes da API, consulte GetObjectTagging em Referência AWS SDK para .NET da API.

Bloquear objetos do Amazon S3

O exemplo de código a seguir mostra como trabalhar com os recursos de bloqueio de objetos do S3.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do Amazon S3.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
 lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
        5. Clean up objects and buckets.
   */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
```

```
    public static IConfiguration _configuration = null!;
    private static string _resourcePrefix = null!;
    private static string noLockBucketName = null!;
    private static string lockEnabledBucketName = null!;
    private static string retentionAfterCreationBucketName = null!;
    private static List<string> bucketNames = new List<string>();
    private static List<string> fileNames = new List<string>();

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3ActionsWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ConfigurationSetup();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            await Setup(true);

            await DemoActionChoices();
```

```
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-
creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }
```

```
    // <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create several
S3\n" +
            "buckets and files to demonstrate working with S3 locking features.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without object
lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName, false);
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
true);
        await
_s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nA bucket can be configured to use object locking with a
default retention period.");
        await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);
```

```
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName) +
i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on individual
files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
                    var exampleFileName = fileNames[i];
```

```
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
                                    $"\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for 1
day.
                                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                        bucketName, exampleFileName,
                                        ObjectLockRetentionMode.Governance,
                                        DateTime.UtcNow.AddDays(1));
                                }
                                break;
                            }
                    }
                }
            }
        }
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

```
    // <summary>
    /// List all of the current buckets and objects.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>The list of buckets and objects.</returns>
    public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
    {
        var allObjects = new List<S3ObjectVersion>();
        foreach (var bucketName in bucketNames)
        {
            var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
            foreach (var objectKey in objectsInBucket.Versions)
            {
                allObjects.Add(objectKey);
            }
        }

        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
            "List all files in buckets.",
```

```
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the scenario."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
        {
            Console.WriteLine(new string('-', 80));
            choice = GetChoiceResponse(
                "\nExplore the S3 locking features by selecting one of the following
choices:"
                , choices);
            Console.WriteLine(new string('-', 80));
            switch (choice)
            {
                case 0:
                    {
                        await ListBucketsAndObjects(true);
                        break;
                    }
                case 1:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 2:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
```

```
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 3:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
overwrite:");
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        // Create the file if it does not already exist.
                        if (!File.Exists(allFiles[fileChoice].Key))
                        {
                            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
                            await sw.WriteLineAsync(
                                "This is a sample file for uploading to a bucket.");
                        }
                        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
                        break;
                    }
                case 4:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
                        break;
                    }
                case 5:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
view:");
```

```
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        break;
                    }
            }
        }
        return true;
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files and
buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
                // Check for a legal hold.
                var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
                var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
```

```
                await _s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName,
fileInfo.Key, hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
```

```
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }
}
```

Uma classe de wrapper para funções do S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
```

```
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the bucket.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
 {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
```

```
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
```

```
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in days
or years but not both.
                        }
                    }
                }
            };

            var response = await _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
```

```
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in {bucketName}:
" +
                                $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
```

```
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in {bucketName}:
" +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await _amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                              $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
```

```
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.<returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\tCould not upload {objectName} to {bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }
```

```
    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
 objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
 {bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
```

```
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Fazer solicitações condicionais

O exemplo de código a seguir mostra como adicionar pré-condições a solicitações do Amazon S3.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos de solicitação condicional do Amazon S3.

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This example demonstrates the use of conditional requests for S3 operations.
    You can use conditional requests to add preconditions to S3 read requests to
 return or copy
    an object based on its Entity tag (ETag), or last modified date.
    You can use a conditional write requests to prevent overwrites by ensuring
    there is no existing object with the same key.
   */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    public static string _resourcePrefix = null!;
    public static string _sourceBucketName = null!;
    public static string _destinationBucketName = null!;
    public static string _sampleObjectKey = null!;
```

```
    public static string _sampleObjectEtag = null!;
    public static bool _interactive = true;


    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3ActionsWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Conditional Requests Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            ConfigurationSetup();
            _sampleObjectEtag = await Setup(_sourceBucketName,
_destinationBucketName, _sampleObjectKey);

            await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,
_sampleObjectKey, _sampleObjectEtag, 0);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
```

```
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await CleanupScenario(_sourceBucketName, _destinationBucketName);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        _sourceBucketName = _resourcePrefix + "-source";
        _destinationBucketName = _resourcePrefix + "-dest";
        _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
    }

    /// <summary>
    /// Sets up the scenario by creating a source and destination bucket, and
uploading a test file to the source bucket.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
```

```
    /// <param name="objectKey">The name of the test file to add to the source
 bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string destBucket,
 string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create several
 S3\n" +
            "buckets and files to demonstrate working with S3 conditional requests.
\n" +
            "This example demonstrates the use of conditional requests for S3
 operations.\r\n" +
            "You can use conditional requests to add preconditions to S3 read
 requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r\n"
 +
            "You can use a conditional write requests to prevent overwrites by
 ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.\r
\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
 example.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();

        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
 sourceBucket,
            "Test file content.");

        return eTag;
    }

    /// <summary>
    /// Cleans up the scenario by deleting the source and destination buckets.
    /// </summary>
```

```
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task CleanupScenario(string sourceBucket, string destBucket)
    {
        await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
        await _s3ActionsWrapper.CleanupBucketByName(destBucket);
    }

    /// <summary>
    /// Displays a list of the objects in the test buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task DisplayBuckets(string sourceBucket, string destBucket)
    {
        await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
        await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
    }

    /// <summary>
    /// Displays the menu of conditional request options for the user.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The key of the test object in the source bucket.</
param>
    /// <param name="etag">The ETag of the test object in the source bucket.</param>
    public static async Task DisplayDemoChoices(string sourceBucket, string
 destBucket, string objectKey, string etag, int defaultChoice)
    {
        var actions = new[]
        {
            "Print a list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit."
        };

        var conditions = new[]
        {
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should fail.",
```

```
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail."
        };

        var conditionTypes = new[]
        {
            S3ConditionType.IfMatch,
            S3ConditionType.IfNoneMatch,
            S3ConditionType.IfModifiedSince,
            S3ConditionType.IfUnmodifiedSince,
        };

        var yesterdayDate = DateTime.UtcNow.AddDays(-1);

        int choice;
        while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
features by selecting one of the following choices:", actions, defaultChoice)) !=
4)
        {
            switch (choice)
            {
                case 0:
                    Console.WriteLine("Listing the objects and buckets.");
                    await DisplayBuckets(sourceBucket, destBucket);
                    break;
                case 1:
                    int conditionTypeIndex = GetChoiceResponse("Perform a
conditional read:", conditions, 1);
                    if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
                    {
                        await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
                    }
                    else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)
                    {
                        await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
                    }
                    break;
                case 2:
                    int copyConditionTypeIndex = GetChoiceResponse("Perform a
conditional copy:", conditions, 1);
```

```
                    string destKey = GetStringResponse("Enter an object key:",
"sampleObjectKey");
                    if (copyConditionTypeIndex == 0 || copyConditionTypeIndex == 1)
                    {
                        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
etag);
                    }
                    else if (copyConditionTypeIndex == 2 || copyConditionTypeIndex
== 3)
                    {
                        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
                    }
                    break;
                case 3:
                    Console.WriteLine("Perform a conditional write using IfNoneMatch
condition on the object key.");
                    Console.WriteLine("If the key is a duplicate, the write will
fail.");
                    string newObjectKey = GetStringResponse("Enter an object key:",
"newObjectKey");
                    await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
                    break;
            }

            if (!_interactive)
            {
                break;
            }
        }

        Console.WriteLine("Proceeding to cleanup.");
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
```

```
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files and
buckets? (y/n) "))
        {
            await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
            await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices, int
defaultChoice)
```

```
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        if (!_interactive)
            return defaultChoice;

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }

    /// <summary>
    /// Get a string response from the user.
    /// </summary>
    /// <param name="question">The question to print.</param>
    /// <param name="defaultAnswer">A default answer to use when not interactive.</
param>
    /// <returns>The string response.</returns>
    public static string GetStringResponse(string? question, string defaultAnswer)
    {
        string? answer = "";
        if (_interactive)
        {
            do
            {
                Console.WriteLine(question);
                answer = Console.ReadLine();
            } while (string.IsNullOrWhiteSpace(answer));
        }
        else
        {
```

```
            answer = defaultAnswer;
        }

        return answer;
    }
}
```

Uma classe de wrapper para funções do S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    /// <param name="logger">The class logger.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
    {
        _amazonS3 = amazonS3;
        _logger = logger;
    }

    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
```

```
    /// <param name="conditionType">The type of condition: 'IfMatch', 'IfNoneMatch',
 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
 sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null, string?
 etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
```

```
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read successful. Here are the first
 20 bytes of the object:\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional read failed: Precondition failed");
            }
            else if (e.ErrorCode == "NotModified")
            {
                _logger.LogError("Conditional read failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }

    /// <summary>
    /// Uploads an object to Amazon S3 with a conditional request. Prevents
 overwrite using an IfNoneMatch condition for the object key.
    /// </summary>
    /// <param name="objectKey">The key of the object to upload.</param>
    /// <param name="bucket">The source bucket of the object.</param>
    /// <param name="content">The content to upload as a string.</param>
    /// <returns>The ETag if the conditional write is successful, empty otherwise.</
returns>
    public async Task<string> PutObjectConditional(string objectKey, string bucket,
 string content)
    {
        try
        {
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
```

```
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
 {objectKey} in bucket {bucket}.");
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }

    /// <summary>
    /// Copies an object from one Amazon S3 bucket to another with a conditional
 request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
    /// <param name="conditionType">The type of condition to apply, e.g.
 'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
 'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string destKey,
 string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null, string?
 etagConditionalValue = null)
    {
```

```
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            await _amazonS3.CopyObjectAsync(copyObjectRequest);
            _logger.LogInformation($"Conditional copy successful for key {destKey}
in bucket {destBucket}.");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional copy failed: Precondition failed");
            }
            else if (e.ErrorCode == "304")
            {
```

```
                _logger.LogError("Conditional copy failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithName(string bucketName)
    {
        Console.WriteLine($"\tCreating bucket {bucketName}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true
            };

            await _amazonS3.PutBucketAsync(request);
            var bucketReady = false;
            var retries = 5;
            while (!bucketReady && retries > 0)
            {
                Thread.Sleep(5000);
                bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
                retries--;
            }

            return bucketReady;
        }
        catch (BucketAlreadyExistsException ex)
        {
            Console.WriteLine($"Bucket already exists: '{ex.Message}'");
```

```
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Cleans up objects and deletes the bucket by name.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public async Task CleanupBucketByName(string bucketName)
    {
        try
        {
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
            }
            await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest { BucketName =
bucketName });
            Console.WriteLine($"Cleaned up bucket: {bucketName}.");
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
            }
            else
            {
                Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
                throw;
            }
        }
    }
```

```
    /// <summary>
    /// List the contents of the bucket with their ETag.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
    {
        var results = new List<S3Object>();
        try
        {
            Console.WriteLine($"\t Items in bucket {bucketName}");
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            if (listObjectsResponse.S3Objects.Count == 0)
            {
                Console.WriteLine("\t\tNo objects found.");
            }
            else
            {
                foreach (var obj in listObjectsResponse.S3Objects)
                {
                    Console.WriteLine($"\t\t object: {obj.Key} ETag {obj.ETag}");
                }
            }
            results = listObjectsResponse.S3Objects;

        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                _logger.LogError($"Bucket {bucketName} does not exist.");
            }
            else
            {
                _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
                throw;
            }
        }

        return results;
    }
```

```
    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket by deleting the objects and then the bucket itself.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CleanUpBucketByName(string bucketName)
    {
        try
        {
            var allFiles = await ListBucketContentsByName(bucketName);

            foreach (var fileInfo in allFiles)
            {
                await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
```

```
            }

            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CopyObject
  - GetObject
  - PutObject

Gerenciar listas de controle de acesso (ACLs)

Os exemplos de código a seguir mostram como gerenciar listas de controle de acesso (ACLs) para buckets do Amazon S3.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
```

```
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to manage Amazon Simple Storage Service
    /// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
    /// access.
    /// </summary>
    public class ManageACLs
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket1";
            string newBucketName = "amzn-s3-demo-bucket2";
            string keyName = "sample-object.txt";
            string emailAddress = "someone@example.com";

            // If the AWS Region where your bucket is located is different from
            // the Region defined for the default user, pass the Amazon S3 bucket's
            // name to the client constructor. It should look like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
            IAmazonS3 client = new AmazonS3Client();

            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
```

```
        /// belonging to the person to whom access to the Amazon S3 bucket will be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
            IAmazonS3 client,
            string bucketName,
            string newBucketName,
            string keyName,
            string emailAddress)
        {
            try
            {
                // Create a new Amazon S3 bucket and specify canned ACL.
                var success = await CreateBucketWithCannedACLAsync(client,
 newBucketName);

                // Get the ACL on a bucket.
                await GetBucketACLAsync(client, bucketName);

                // Add (replace) the ACL on an object in a bucket.
                await AddACLToExistingObjectAsync(client, bucketName, keyName,
 emailAddress);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Exception: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL attached.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new Amazon S3 bucket.</param>
        /// <returns>Returns a boolean value indicating success or failure.</
returns>
        public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
 client, string newBucketName)
        {
            var request = new PutBucketRequest()
            {
                BucketName = newBucketName,
                BucketRegion = S3Region.EUWest1,
```

```
                // Add a canned ACL.
                CannedACL = S3CannedACL.LogDeliveryWrite,
            };

            var response = await client.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Retrieves the ACL associated with the Amazon S3 bucket name in the
        /// bucketName parameter.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket for which we want to get
 the
        /// ACL list.</param>
        /// <returns>Returns an S3AccessControlList returned from the call to
        /// GetACLAsync.</returns>
        public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
 client, string bucketName)
        {
            GetACLResponse response = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
            });

            return response.AccessControlList;
        }



        /// <summary>
        /// Adds a new ACL to an existing object in the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon S3
        /// bucket containing the object to which we want to apply a new ACL.</
param>
        /// <param name="keyName">A string representing the name of the object
        /// to which we want to apply the new ACL.</param>
```

```
        /// <param name="emailAddress">The email address of the person to whom
        /// we will be applying to whom access will be granted.</param>
        public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
 string bucketName, string keyName, string emailAddress)
        {
            // Retrieve the ACL for an object.
            GetACLResponse aclResponse = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
            });

            S3AccessControlList acl = aclResponse.AccessControlList;

            // Retrieve the owner.
            Owner owner = acl.Owner;

            // Clear existing grants.
            acl.Grants.Clear();

            // Add a grant to reset the owner's full permission
            // (the previous clear statement removed all permissions).
            var fullControlGrant = new S3Grant
            {
                Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
            };
            acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

            // Specify email to identify grantee for granting permissions.
            var grantUsingEmail = new S3Grant
            {
                Grantee = new S3Grantee { EmailAddress = emailAddress },
                Permission = S3Permission.WRITE_ACP,
            };

            // Specify log delivery group as grantee.
            var grantLogDeliveryGroup = new S3Grant
            {
                Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/s3/
LogDelivery" },
                Permission = S3Permission.WRITE,
            };

            // Create a new ACL.
```

```
            var newAcl = new S3AccessControlList
            {
                Grants = new List<S3Grant> { grantUsingEmail,
grantLogDeliveryGroup },
                Owner = owner,
            };

            // Set the new ACL. We're throwing away the response here.
            _ = await client.PutACLAsync(new PutACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
                AccessControlList = newAcl,
            });
        }

    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - GetBucketAcl
  - GetObjectAcl
  - PutBucketAcl
  - PutObjectAcl

Executar uma cópia multipart

O exemplo de código a seguir mostra como executar uma cópia multipart de um objeto do Amazon S3.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }

    /// <summary>
    /// This method uses the passed client object to perform a multipart
    /// copy operation.
    /// </summary>
    /// <param name="client">An Amazon S3 client object that will be used
    /// to perform the copy.</param>
    public static async Task MPUCopyObjectAsync(AmazonS3Client client)
    {
        // Create a list to store the copy part responses.
        var copyResponses = new List<CopyPartResponse>();

        // Setup information required to initiate the multipart upload.
        var initiateRequest = new InitiateMultipartUploadRequest
        {
            BucketName = TargetBucket,
            Key = TargetObjectKey,
```

```
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            string uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                var metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = SourceBucket,
                    Key = SourceObjectKey,
                };

                GetObjectMetadataResponse metadataResponse =
                    await client.GetObjectMetadataAsync(metadataRequest);
                var objectSize = metadataResponse.ContentLength; // Length in bytes.

                // Copy the parts.
                var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    var copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = TargetBucket,
                        DestinationKey = TargetObjectKey,
                        SourceBucket = SourceBucket,
                        SourceKey = SourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i,
                    };

                    copyResponses.Add(await client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
```

```
                }

                // Set up to complete the copy.
                var completeRequest = new CompleteMultipartUploadRequest
                {
                    BucketName = TargetBucket,
                    Key = TargetObjectKey,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{e.Message}' when writing an object");
            }
            catch (Exception e)
            {
                Console.WriteLine($"Unknown encountered on server.
Message:'{e.Message}' when writing an object");
            }
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - GetObjectMetadata
  - UploadPartCopy

Como transformar dados com o S3 Object Lambda

O exemplo de código a seguir mostra como transformar dados para sua aplicação com o S3 Object Lambda.

SDK para .NET

Mostra como adicionar código personalizado a solicitações GET padrão do S3 para modificar o objeto solicitado e recuperado do S3 e possibilitar que o objeto atenda às necessidades do cliente ou aplicação solicitante.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Lambda
- Amazon S3

Fazer upload ou download de arquivos grandes

O exemplo de código a seguir mostra como fazer upload ou download de arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte Carregar um objeto usando carregamento fracionado.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Chame funções que transferem arquivos de e para um bucket do S3 usando o Amazon TransferUtility S3.

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;



// This Amazon S3 client uses the default user credentials
// defined for this computer.
```

```
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
 $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil, bucketName,
 fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
 {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
```

```
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil, bucketName,
 keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
 {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
 keyName, localPath);
if (success)
{
    Console.WriteLine("$Successfully downloaded the file, {keyName} from
 {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
```

```
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil, bucketName,
 s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
 Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to an
\n\t  S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
```

```
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
 directory.");
    Console.WriteLine($"\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth - textToCenter.Length) /
 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
 s3Path)
{
    ListObjectsV2Request request = new()
    {
```

```
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

Carregue um único arquivo.

```
        /// <summary>
        /// Uploads a single file from the local computer to an S3 bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the file
        /// will be stored.</param>
        /// <param name="fileName">The name of the file to upload.</param>
        /// <param name="localPath">The local path where the file is stored.</param>
        /// <returns>A boolean value indicating the success of the action.</returns>
        public static async Task<bool> UploadSingleFileAsync(
            TransferUtility transferUtil,
            string bucketName,
            string fileName,
            string localPath)
        {
            if (File.Exists($"{localPath}\\{fileName}"))
            {
```

```
                try
                {
                    await transferUtil.UploadAsync(new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        Key = fileName,
                        FilePath = $"{localPath}\\{fileName}",
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Could not upload {fileName} from {localPath}
because:");
                    Console.WriteLine(s3Ex.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"{fileName} does not exist in {localPath}");
                return false;
            }
        }
```

Carregue um diretório local inteiro.

```
        /// <summary>
        /// Uploads all the files in a local directory to a directory in an S3
        /// bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the files
        /// will be stored.</param>
        /// <param name="keyPrefix">The key prefix is the S3 directory where
        /// the files will be stored.</param>
        /// <param name="localPath">The local directory that contains the files
        /// to be uploaded.</param>
```

```
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> UploadFullDirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string keyPrefix,
            string localPath)
        {
            if (Directory.Exists(localPath))
            {
                try
                {
                    await transferUtil.UploadDirectoryAsync(new
 TransferUtilityUploadDirectoryRequest
                    {
                        BucketName = bucketName,
                        KeyPrefix = keyPrefix,
                        Directory = localPath,
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Can't upload the contents of {localPath}
 because:");
                    Console.WriteLine(s3Ex?.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"The directory {localPath} does not exist.");
                return false;
            }
        }
```

Baixe um único arquivo.

```
        /// <summary>
```

```
        /// Download a single file from an S3 bucket to the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// file to download.</param>
        /// <param name="keyName">The name of the file to download.</param>
        /// <param name="localPath">The path on the local computer where the
        /// downloaded file will be saved.</param>
        /// <returns>A Boolean value indicating the results of the action.</returns>
        public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
            string bucketName,
            string keyName,
            string localPath)
        {
            await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = $"{localPath}\\{keyName}",
            });

            return (File.Exists($"{localPath}\\{keyName}"));
        }
```

Baixe o conteúdo de um bucket do S3.

```
        /// <summary>
        /// Downloads the contents of a directory in an S3 bucket to a
        /// directory on the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The bucket containing the files to download.</
param>
        /// <param name="s3Path">The S3 directory where the files are located.</
param>
        /// <param name="localPath">The local path to which the files will be
        /// saved.</param>
```

```
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> DownloadS3DirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string s3Path,
            string localPath)
        {
            int fileCount = 0;

            // If the directory doesn't exist, it will be created.
            if (Directory.Exists(s3Path))
            {
                var files = Directory.GetFiles(localPath);
                fileCount = files.Length;
            }

            await transferUtil.DownloadDirectoryAsync(new
 TransferUtilityDownloadDirectoryRequest
            {
                BucketName = bucketName,
                LocalDirectory = localPath,
                S3Directory = s3Path,
            });

            if (Directory.Exists(localPath))
            {
                var files = Directory.GetFiles(localPath);
                if (files.Length > fileCount)
                {
                    return true;
                }

                // No change in the number of files. Assume
                // the download failed.
                return false;
            }

            // The local directory doesn't exist. No files
            // were downloaded.
            return false;
        }
```

Acompanhe o progresso de um upload usando TransferUtility o.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }

    /// <summary>
    /// Starts an Amazon S3 multipart upload and assigns an event handler to
    /// track the progress of the upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// perform the multipart upload.</param>
    /// <param name="bucketName">The name of the bucket to which to upload
    /// the file.</param>
    /// <param name="filePath">The path, including the file name of the
    /// file to be uploaded to the Amazon S3 bucket.</param>
    /// <param name="keyName">The file name to be used in the
```

```
        /// destination Amazon S3 bucket.</param>
        public static async Task TrackMPUAsync(
            IAmazonS3 client,
            string bucketName,
            string filePath,
            string keyName)
        {
            try
            {
                var fileTransferUtility = new TransferUtility(client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName,
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>(
                        UploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:: {ex.Message}");
            }
        }

        /// <summary>
        /// Event handler to check the progress of the multipart upload.
        /// </summary>
        /// <param name="sender">The object that raised the event.</param>
        /// <param name="e">The object that contains multipart upload
        /// information.</param>
        public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
        {
            // Process event.
```

```
            Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
        }
    }
```

Faça upload de um objeto com criptografia.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
/// perform a multipart upload to an Amazon S3 bucket.
/// </summary>
public class SSECLowLevelMPUcopyObject
{
    public static async Task Main()
    {
        string existingBucketName = "amzn-s3-demo-bucket";
        string sourceKeyName = "sample_file.txt";
        string targetKeyName = "sample_file_copy.txt";
        string filePath = $"sample\\{targetKeyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USEast1.
        IAmazonS3 client = new AmazonS3Client();

        // Create the encryption key.
        var base64Key = CreateEncryptionKey();

        await CreateSampleObjUsingClientEncryptionKeyAsync(
            client,
            existingBucketName,
            sourceKeyName,
            filePath,
```

```
                base64Key);
        }

        /// <summary>
        /// Creates the encryption key to use with the multipart upload.
        /// </summary>
        /// <returns>A string containing the base64-encoded key for encrypting
        /// the multipart upload.</returns>
        public static string CreateEncryptionKey()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
            return base64Key;
        }

        /// <summary>
        /// Creates and uploads an object using a multipart upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object used to
        /// initialize and perform the multipart upload.</param>
        /// <param name="existingBucketName">The name of the bucket to which
        /// the object will be uploaded.</param>
        /// <param name="sourceKeyName">The source object name.</param>
        /// <param name="filePath">The location of the source object.</param>
        /// <param name="base64Key">The encryption key to use with the upload.</
param>
        public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
            IAmazonS3 client,
            string existingBucketName,
            string sourceKeyName,
            string filePath,
            string base64Key)
        {
            List<UploadPartResponse> uploadResponses = new
 List<UploadPartResponse>();

            InitiateMultipartUploadRequest initiateRequest = new
 InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
```

```
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
               await client.InitiateMultipartUploadAsync(initiateRequest);

            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key,
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };
```

```
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine($"Exception occurred: {exception.Message}");

                // If there was an error, abort the multipart upload.
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };

                await client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
```

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");
```

```
                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request - {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

SageMaker Exemplos de IA usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com SageMaker IA.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá SageMaker AI

O exemplo de código a seguir mostra como começar a usar a SageMaker IA.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.SageMaker;
using Amazon.SageMaker.Model;

namespace SageMakerActions;

public static class HelloSageMaker
{
    static async Task Main(string[] args)
    {
        var sageMakerClient = new AmazonSageMakerClient();

        Console.WriteLine($"Hello Amazon SageMaker! Let's list some of your notebook
 instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five notebook instances.
        var response = await sageMakerClient.ListNotebookInstancesAsync(
            new ListNotebookInstancesRequest()
            {
                MaxResults = 5
            });

        if (!response.NotebookInstances.Any())
        {
            Console.WriteLine($"No notebook instances found.");
            Console.WriteLine("See https://docs.aws.amazon.com/sagemaker/latest/dg/
howitworks-create-ws.html to create one.");
```

```
        }

        foreach (var notebookInstance in response.NotebookInstances)
        {
            Console.WriteLine($"\tInstance:
 {notebookInstance.NotebookInstanceName}");
            Console.WriteLine($"\tArn: {notebookInstance.NotebookInstanceArn}");
            Console.WriteLine($"\tCreation Date:
 {notebookInstance.CreationTime.ToShortDateString()}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte ListNotebookInstances em Referência AWS SDK para .NET da API.

Ações

`CreatePipeline`

O código de exemplo a seguir mostra como usar `CreatePipeline`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
```

```
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }
```

- Para obter detalhes da API, consulte CreatePipeline em Referência AWS SDK para .NET da API.

`DeletePipeline`

O código de exemplo a seguir mostra como usar `DeletePipeline`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete a SageMaker pipeline by name.
/// </summary>
/// <param name="pipelineName">The name of the pipeline to delete.</param>
/// <returns>The ARN of the pipeline.</returns>
public async Task<string> DeletePipelineByName(string pipelineName)
{
    var deleteResponse = await _amazonSageMaker.DeletePipelineAsync(
        new DeletePipelineRequest()
        {
            PipelineName = pipelineName
        });

    return deleteResponse.PipelineArn;
}
```

- Para obter detalhes da API, consulte DeletePipeline em Referência AWS SDK para .NET da API.

`DescribePipelineExecution`

O código de exemplo a seguir mostra como usar `DescribePipelineExecution`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Check the status of a run.
    /// </summary>
    /// <param name="pipelineExecutionArn">The ARN.</param>
    /// <returns>The status of the pipeline.</returns>
    public async Task<PipelineExecutionStatus> CheckPipelineExecutionStatus(string
pipelineExecutionArn)
    {
        var describeResponse = await
_amazonSageMaker.DescribePipelineExecutionAsync(
            new DescribePipelineExecutionRequest()
            {
                PipelineExecutionArn = pipelineExecutionArn
            });

        return describeResponse.PipelineExecutionStatus;
    }
```

- Para obter detalhes da API, consulte DescribePipelineExecution em Referência AWS SDK para .NET da API.

StartPipelineExecution

O código de exemplo a seguir mostra como usar `StartPipelineExecution`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Run a pipeline with input and output file locations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue to use for pipeline
callbacks.</param>
    /// <param name="inputLocationUrl">The input location in Amazon Simple Storage
Service (Amazon S3).</param>
```

```
    /// <param name="outputLocationUrl">The output location in Amazon S3.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="executionRoleArn">The ARN of the role.</param>
    /// <returns>The ARN of the pipeline run.</returns>
    public async Task<string> ExecutePipeline(
        string queueUrl,
        string inputLocationUrl,
        string outputLocationUrl,
        string pipelineName,
        string executionRoleArn)
    {
        var inputConfig = new VectorEnrichmentJobInputConfig()
        {
            DataSourceConfig = new()
            {
                S3Data = new VectorEnrichmentJobS3Data()
                {
                    S3Uri = inputLocationUrl
                }
            },
            DocumentType = VectorEnrichmentJobDocumentType.CSV
        };

        var exportConfig = new ExportVectorEnrichmentJobOutputConfig()
        {
            S3Data = new VectorEnrichmentJobS3Data()
            {
                S3Uri = outputLocationUrl
            }
        };

        var jobConfig = new VectorEnrichmentJobConfig()
        {
            ReverseGeocodingConfig = new ReverseGeocodingConfig()
            {
                XAttributeName = "Longitude",
                YAttributeName = "Latitude"
            }
        };

#pragma warning disable SageMaker1002 // Property value does not match required
 pattern is allowed here to match the pipeline definition.
        var startExecutionResponse = await
 _amazonSageMaker.StartPipelineExecutionAsync(
```

```
            new StartPipelineExecutionRequest()
            {
                PipelineName = pipelineName,
                PipelineExecutionDisplayName = pipelineName + "-example-execution",
                PipelineParameters = new List<Parameter>()
                {
                    new Parameter() { Name = "parameter_execution_role", Value =
 executionRoleArn },
                    new Parameter() { Name = "parameter_queue_url", Value =
 queueUrl },
                    new Parameter() { Name = "parameter_vej_input_config", Value =
 JsonSerializer.Serialize(inputConfig) },
                    new Parameter() { Name = "parameter_vej_export_config", Value =
 JsonSerializer.Serialize(exportConfig) },
                    new Parameter() { Name = "parameter_step_1_vej_config", Value =
 JsonSerializer.Serialize(jobConfig) }
                }
            });
#pragma warning restore SageMaker1002
        return startExecutionResponse.PipelineExecutionArn;
    }
```

- Para obter detalhes da API, consulte StartPipelineExecution em Referência AWS SDK para .NET da API.

`UpdatePipeline`

O código de exemplo a seguir mostra como usar `UpdatePipeline`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
```

```
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }
```

- Para obter detalhes da API, consulte UpdatePipeline em Referência AWS SDK para .NET da API.

Cenários

Conceitos básicos de trabalhos geoespaciais e pipelines

O exemplo de código a seguir mostra como:

- Configurar recursos para um pipeline.
- Configurar um pipeline que executa um trabalho geoespacial.
- Iniciar a execução de um pipeline.
- Monitorar o status da execução.
- Ver a saída do pipeline.
- Limpar recursos.

Para obter mais informações, consulte Criar e executar SageMaker pipelines usando AWS SDKs em. Community.aws

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as operações de SageMaker IA.

```
using System.Text.Json;
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
using Amazon.SageMakerGeospatial;
using Amazon.SageMakerGeospatial.Model;

namespace SageMakerActions;

/// <summary>
/// Wrapper class for Amazon SageMaker actions and logic.
/// </summary>
public class SageMakerWrapper
{
    private readonly IAmazonSageMaker _amazonSageMaker;
    public SageMakerWrapper(IAmazonSageMaker amazonSageMaker)
    {
        _amazonSageMaker = amazonSageMaker;
    }
```

```
    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }

    /// <summary>
    /// Run a pipeline with input and output file locations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue to use for pipeline
 callbacks.</param>
```

```
    /// <param name="inputLocationUrl">The input location in Amazon Simple Storage
 Service (Amazon S3).</param>
    /// <param name="outputLocationUrl">The output location in Amazon S3.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="executionRoleArn">The ARN of the role.</param>
    /// <returns>The ARN of the pipeline run.</returns>
    public async Task<string> ExecutePipeline(
        string queueUrl,
        string inputLocationUrl,
        string outputLocationUrl,
        string pipelineName,
        string executionRoleArn)
    {
        var inputConfig = new VectorEnrichmentJobInputConfig()
        {
            DataSourceConfig = new()
            {
                S3Data = new VectorEnrichmentJobS3Data()
                {
                    S3Uri = inputLocationUrl
                }
            },
            DocumentType = VectorEnrichmentJobDocumentType.CSV
        };

        var exportConfig = new ExportVectorEnrichmentJobOutputConfig()
        {
            S3Data = new VectorEnrichmentJobS3Data()
            {
                S3Uri = outputLocationUrl
            }
        };

        var jobConfig = new VectorEnrichmentJobConfig()
        {
            ReverseGeocodingConfig = new ReverseGeocodingConfig()
            {
                XAttributeName = "Longitude",
                YAttributeName = "Latitude"
            }
        };

#pragma warning disable SageMaker1002 // Property value does not match required
 pattern is allowed here to match the pipeline definition.
```

```
        var startExecutionResponse = await
 _amazonSageMaker.StartPipelineExecutionAsync(
            new StartPipelineExecutionRequest()
            {
                PipelineName = pipelineName,
                PipelineExecutionDisplayName = pipelineName + "-example-execution",
                PipelineParameters = new List<Parameter>()
                {
                    new Parameter() { Name = "parameter_execution_role", Value =
 executionRoleArn },
                    new Parameter() { Name = "parameter_queue_url", Value =
 queueUrl },
                    new Parameter() { Name = "parameter_vej_input_config", Value =
 JsonSerializer.Serialize(inputConfig) },
                    new Parameter() { Name = "parameter_vej_export_config", Value =
 JsonSerializer.Serialize(exportConfig) },
                    new Parameter() { Name = "parameter_step_1_vej_config", Value =
 JsonSerializer.Serialize(jobConfig) }
                }
            });
#pragma warning restore SageMaker1002
        return startExecutionResponse.PipelineExecutionArn;
    }

    /// <summary>
    /// Check the status of a run.
    /// </summary>
    /// <param name="pipelineExecutionArn">The ARN.</param>
    /// <returns>The status of the pipeline.</returns>
    public async Task<PipelineExecutionStatus> CheckPipelineExecutionStatus(string
 pipelineExecutionArn)
    {
        var describeResponse = await
 _amazonSageMaker.DescribePipelineExecutionAsync(
            new DescribePipelineExecutionRequest()
            {
                PipelineExecutionArn = pipelineExecutionArn
            });

        return describeResponse.PipelineExecutionStatus;
    }

    /// <summary>
    /// Delete a SageMaker pipeline by name.
```

```
    /// </summary>
    /// <param name="pipelineName">The name of the pipeline to delete.</param>
    /// <returns>The ARN of the pipeline.</returns>
    public async Task<string> DeletePipelineByName(string pipelineName)
    {
        var deleteResponse = await _amazonSageMaker.DeletePipelineAsync(
            new DeletePipelineRequest()
            {
                PipelineName = pipelineName
            });

        return deleteResponse.PipelineArn;
    }
}
```

Crie uma função que processe retornos de chamada do pipeline de SageMaker IA.

```
using System.Text.Json;
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
using Amazon.SageMakerGeospatial;
using Amazon.SageMakerGeospatial.Model;

// Assembly attribute to enable the AWS Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SageMakerLambda;

/// <summary>
/// The AWS Lambda function handler for the Amazon SageMaker pipeline.
/// </summary>
public class SageMakerLambdaFunction
{
    /// <summary>
    /// Default constructor. This constructor is used by AWS Lambda to construct the
 instance. When invoked in a Lambda environment
```

```
    /// the AWS credentials will come from the AWS Identity and Access Management
(IAM) role associated with the function. The AWS Region will be set to the
    /// Region that the Lambda function is running in.
    /// </summary>
    public SageMakerLambdaFunction()
    {
    }

    /// <summary>
    /// The AWS Lambda function handler that processes events from the SageMaker
pipeline and starts a job or export.
    /// </summary>
    /// <param name="request">The custom SageMaker pipeline request object.</param>
    /// <param name="context">The Lambda context.</param>
    /// <returns>The dictionary of output parameters.</returns>
    public async Task<Dictionary<string, string>> FunctionHandler(PipelineRequest
request, ILambdaContext context)
    {
        var geoSpatialClient = new AmazonSageMakerGeospatialClient();
        var sageMakerClient = new AmazonSageMakerClient();
        var responseDictionary = new Dictionary<string, string>();
        context.Logger.LogInformation("Function handler started with request: " +
JsonSerializer.Serialize(request));
        if (request.Records != null && request.Records.Any())
        {
            context.Logger.LogInformation("Records found, this is a queue event.
Processing the queue records.");
            foreach (var message in request.Records)
            {
                await ProcessMessageAsync(message, context, geoSpatialClient,
sageMakerClient);
            }
        }
        else if (!string.IsNullOrEmpty(request.vej_export_config))
        {
            context.Logger.LogInformation("Export configuration found, this is an
export. Start the Vector Enrichment Job (VEJ) export.");

            var outputConfig =
                JsonSerializer.Deserialize<ExportVectorEnrichmentJobOutputConfig>(
                    request.vej_export_config);

            var exportResponse = await
geoSpatialClient.ExportVectorEnrichmentJobAsync(
```

```
                new ExportVectorEnrichmentJobRequest()
                {
                    Arn = request.vej_arn,
                    ExecutionRoleArn = request.Role,
                    OutputConfig = outputConfig
                });
            context.Logger.LogInformation($"Export response:
{JsonSerializer.Serialize(exportResponse)}");
            responseDictionary = new Dictionary<string, string>
            {
                { "export_eoj_status", exportResponse.ExportStatus.ToString() },
                { "vej_arn", exportResponse.Arn }
            };
        }
        else if (!string.IsNullOrEmpty(request.vej_name))
        {
            context.Logger.LogInformation("Vector Enrichment Job name found,
starting the job.");
            var inputConfig =
                JsonSerializer.Deserialize<VectorEnrichmentJobInputConfig>(
                    request.vej_input_config);

            var jobConfig =
                JsonSerializer.Deserialize<VectorEnrichmentJobConfig>(
                    request.vej_config);

            var jobResponse = await geoSpatialClient.StartVectorEnrichmentJobAsync(
                new StartVectorEnrichmentJobRequest()
                {
                    ExecutionRoleArn = request.Role,
                    InputConfig = inputConfig,
                    Name = request.vej_name,
                    JobConfig = jobConfig

                });
            context.Logger.LogInformation("Job response: " +
JsonSerializer.Serialize(jobResponse));
            responseDictionary = new Dictionary<string, string>
            {
                { "vej_arn", jobResponse.Arn },
                { "statusCode", jobResponse.HttpStatusCode.ToString() }
            };
        }
        return responseDictionary;
```

```
    }

    /// <summary>
    /// Process a queue message and check the status of a SageMaker job.
    /// </summary>
    /// <param name="message">The queue message.</param>
    /// <param name="context">The Lambda context.</param>
    /// <param name="geoClient">The SageMaker GeoSpatial client.</param>
    /// <param name="sageMakerClient">The SageMaker client.</param>
    /// <returns>Async task.</returns>
    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
ILambdaContext context,
        AmazonSageMakerGeospatialClient geoClient, AmazonSageMakerClient
sageMakerClient)
    {
        context.Logger.LogInformation($"Processed message {message.Body}");

        // Get information about the SageMaker job.
        var payload = JsonSerializer.Deserialize<QueuePayload>(message.Body);
        context.Logger.LogInformation($"Payload token {payload!.token}");
        var token = payload.token;

        if (payload.arguments.ContainsKey("vej_arn"))
        {
            // Use the job ARN and the token to get the job status.
            var job_arn = payload.arguments["vej_arn"];
            context.Logger.LogInformation($"Token: {token}, arn {job_arn}");

            var jobInfo = geoClient.GetVectorEnrichmentJobAsync(
                new GetVectorEnrichmentJobRequest()
                {
                    Arn = job_arn
                });
            context.Logger.LogInformation("Job info: " +
JsonSerializer.Serialize(jobInfo));
            if (jobInfo.Result.Status == VectorEnrichmentJobStatus.COMPLETED)
            {
                context.Logger.LogInformation($"Status completed, resuming
pipeline...");
                await sageMakerClient.SendPipelineExecutionStepSuccessAsync(
                    new SendPipelineExecutionStepSuccessRequest()
                    {
                        CallbackToken = token,
                        OutputParameters = new List<OutputParameter>()
```

```
                        {
                            new OutputParameter()
                                { Name = "export_status", Value =
 jobInfo.Result.Status }
                        }
                    });
            }
            else if (jobInfo.Result.Status == VectorEnrichmentJobStatus.FAILED)
            {
                context.Logger.LogInformation($"Status failed, stopping
 pipeline...");
                await sageMakerClient.SendPipelineExecutionStepFailureAsync(
                    new SendPipelineExecutionStepFailureRequest()
                    {
                        CallbackToken = token,
                        FailureReason = jobInfo.Result.ErrorDetails.ErrorMessage
                    });
            }
            else if (jobInfo.Result.Status == VectorEnrichmentJobStatus.IN_PROGRESS)
            {
                // Put this message back in the queue to reprocess later.
                context.Logger.LogInformation(
                    $"Status still in progress, check back later.");
                throw new("Job still running.");
            }
        }
    }
}
```

Execute um cenário interativo em um prompt de comando.

```
public static class PipelineWorkflow
{
    public static IAmazonIdentityManagementService _iamClient = null!;
    public static SageMakerWrapper _sageMakerWrapper = null!;
    public static IAmazonSQS _sqsClient = null!;
    public static IAmazonS3 _s3Client = null!;
    public static IAmazonLambda _lambdaClient = null!;
    public static IConfiguration _configuration = null!;

    public static string lambdaFunctionName = "SageMakerExampleFunction";
    public static string sageMakerRoleName = "SageMakerExampleRole";
```

```
    public static string lambdaRoleName = "SageMakerExampleLambdaRole";

    private static string[] lambdaRolePolicies = null!;
    private static string[] sageMakerRolePolicies = null!;

    static async Task Main(string[] args)
    {
        var options = new AWSOptions() { Region = RegionEndpoint.USWest2 };
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>(options)
                    .AddAWSService<IAmazonEC2>(options)
                    .AddAWSService<IAmazonSageMaker>(options)
                    .AddAWSService<IAmazonSageMakerGeospatial>(options)
                    .AddAWSService<IAmazonSQS>(options)
                    .AddAWSService<IAmazonS3>(options)
                    .AddAWSService<IAmazonLambda>(options)
                    .AddTransient<SageMakerWrapper>()
        )
        .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ServicesSetup(host);
        string queueUrl = "";
        string queueName = _configuration["queueName"];
        string bucketName = _configuration["bucketName"];
        var pipelineName = _configuration["pipelineName"];

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(
```

```
                "Welcome to the Amazon SageMaker pipeline example scenario.");
            Console.WriteLine(
                "\nThis example scenario will guide you through setting up and
 running an" +
                "\nAmazon SageMaker pipeline. The pipeline uses an AWS Lambda
 function and an" +
                "\nAmazon SQS Queue. It runs a vector enrichment reverse geocode job
 to" +
                "\nreverse geocode addresses in an input file and store the results
 in an export file.");
            Console.WriteLine(new string('-', 80));

            Console.WriteLine(new string('-', 80));
            Console.WriteLine(
                "First, we will set up the roles, functions, and queue needed by the
 SageMaker pipeline.");
            Console.WriteLine(new string('-', 80));

            var lambdaRoleArn = await CreateLambdaRole();
            var sageMakerRoleArn = await CreateSageMakerRole();
            var functionArn = await SetupLambda(lambdaRoleArn, true);
            queueUrl = await SetupQueue(queueName);
            await SetupBucket(bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Now we can create and run our pipeline.");
            Console.WriteLine(new string('-', 80));

            await SetupPipeline(sageMakerRoleArn, functionArn, pipelineName);
            var executionArn = await ExecutePipeline(queueUrl, sageMakerRoleArn,
 pipelineName, bucketName);
            await WaitForPipelineExecution(executionArn);

            await GetOutputResults(bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("The pipeline has completed. To view the pipeline and
 runs " +
                              "in SageMaker Studio, follow these instructions:" +
                              "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/
pipelines-studio.html");
            Console.WriteLine(new string('-', 80));

            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await CleanupResources(true, queueUrl, pipelineName, bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("SageMaker pipeline scenario is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources(true, queueUrl, pipelineName, bucketName);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sageMakerWrapper = host.Services.GetRequiredService<SageMakerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _sqsClient = host.Services.GetRequiredService<IAmazonSQS>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _lambdaClient = host.Services.GetRequiredService<IAmazonLambda>();
    }

    /// <summary>
    /// Set up AWS Lambda, either by updating an existing function or creating a new
function.
    /// </summary>
    /// <param name="roleArn">The role Amazon Resource Name (ARN) to use for the
Lambda function.</param>
    /// <param name="askUser">True to ask the user before updating.</param>
    /// <returns>The ARN of the function.</returns>
    public static async Task<string> SetupLambda(string roleArn, bool askUser)
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Setting up the Lambda function for the pipeline.");
        var handlerName =
"SageMakerLambda::SageMakerLambda.SageMakerLambdaFunction::FunctionHandler";
        var functionArn = "";
        try
        {
            var functionInfo = await _lambdaClient.GetFunctionAsync(new
GetFunctionRequest()
            {
                FunctionName = lambdaFunctionName
            });

            var updateFunction = true;
            if (askUser)
            {
                updateFunction = GetYesNoResponse(
                    $"\tThe Lambda function {lambdaFunctionName} already exists, do
you want to update it?");
            }

            if (updateFunction)
            {
                // Update the Lambda function.
                using var zipMemoryStream = new MemoryStream(await
File.ReadAllBytesAsync("SageMakerLambda.zip"));
                await _lambdaClient.UpdateFunctionCodeAsync(
                    new UpdateFunctionCodeRequest()
                    {
                        FunctionName = lambdaFunctionName,
                        ZipFile = zipMemoryStream,
                    });
            }

            functionArn = functionInfo.Configuration.FunctionArn;
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"\tThe Lambda function {lambdaFunctionName} was not
found, creating the new function.");

            // Create the function if it does not already exist.
            using var zipMemoryStream = new MemoryStream(await
File.ReadAllBytesAsync("SageMakerLambda.zip"));
            var createResult = await _lambdaClient.CreateFunctionAsync(
```

```
                new CreateFunctionRequest()
                {
                    FunctionName = lambdaFunctionName,
                    Runtime = Runtime.Dotnet6,
                    Description = "SageMaker example function.",
                    Code = new FunctionCode()
                    {
                        ZipFile = zipMemoryStream
                    },
                    Handler = handlerName,
                    Role = roleArn,
                    Timeout = 30
                });

            functionArn = createResult.FunctionArn;
        }

        Console.WriteLine($"\tLambda ready with ARN {functionArn}.");
        Console.WriteLine(new string('-', 80));
        return functionArn;
    }

    /// <summary>
    /// Create a role to be used by AWS Lambda. Does not create the role if it
 already exists.
    /// </summary>
    /// <returns>The role ARN.</returns>
    public static async Task<string> CreateLambdaRole()
    {
        Console.WriteLine(new string('-', 80));

        lambdaRolePolicies = new string[]{
            "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess",
            "arn:aws:iam::aws:policy/AmazonSQSFullAccess",
            "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess",
            "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy",
            "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
        };

        var roleArn = await GetRoleArnIfExists(lambdaRoleName);
        if (!string.IsNullOrEmpty(roleArn))
```

```
        {
            return roleArn;
        }

        Console.WriteLine("\tCreating a role to for AWS Lambda to use.");

        var assumeRolePolicy = "{" +
                               "\"Version\": \"2012-10-17\"," +
                               "\"Statement\": [{" +
                                    "\"Effect\": \"Allow\"," +
                                    "\"Principal\": {" +
                                        $"\"Service\": [" +
                                            "\"sagemaker.amazonaws.com\"," +
                                            "\"sagemaker-geospatial.amazonaws.com
\"," +
                                            "\"lambda.amazonaws.com\"," +
                                            "\"s3.amazonaws.com\"" +
                                        "]" +
                                    "}," +
                                    "\"Action\": \"sts:AssumeRole\"" +
                               "}]" +
                           "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = lambdaRoleName
            });
        foreach (var policy in lambdaRolePolicies)
        {
            await _iamClient.AttachRolePolicyAsync(
                new AttachRolePolicyRequest()
                {
                    PolicyArn = policy,
                    RoleName = lambdaRoleName
                });
        }

        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        Console.WriteLine($"\tRole ready with ARN {roleResult.Role.Arn}.");
        Console.WriteLine(new string('-', 80));
```

```
        return roleResult.Role.Arn;
    }


    /// <summary>
    /// Create a role to be used by SageMaker.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateSageMakerRole()
    {
        Console.WriteLine(new string('-', 80));

        sageMakerRolePolicies = new string[]{
            "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess",
            "arn:aws:iam::aws:policy/AmazonSageMakerGeospatialFullAccess",
        };

        var roleArn = await GetRoleArnIfExists(sageMakerRoleName);
        if (!string.IsNullOrEmpty(roleArn))
        {
            return roleArn;
        }

        Console.WriteLine("\tCreating a role to use with SageMaker.");

        var assumeRolePolicy = "{" +
                                        "\"Version\": \"2012-10-17\"," +
                                        "\"Statement\": [{" +
                                            "\"Effect\": \"Allow\"," +
                                            "\"Principal\": {" +
                                                $"\"Service\": [" +
                                                    "\"sagemaker.amazonaws.com\"," +
                                                    "\"sagemaker-
geospatial.amazonaws.com\"," +
                                                    "\"lambda.amazonaws.com\"," +
                                                    "\"s3.amazonaws.com\"" +
                                                "]" +
                                            "}," +
                                            "\"Action\": \"sts:AssumeRole\"" +
                                        "}]" +
                                    "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
```

```
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = sageMakerRoleName
            });

        foreach (var policy in sageMakerRolePolicies)
        {
            await _iamClient.AttachRolePolicyAsync(
                new AttachRolePolicyRequest()
                {
                    PolicyArn = policy,
                    RoleName = sageMakerRoleName
                });
        }

        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        Console.WriteLine($"\tRole ready with ARN {roleResult.Role.Arn}.");
        Console.WriteLine(new string('-', 80));
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Set up the SQS queue to use with the pipeline.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <returns>The URL for the queue.</returns>
    public static async Task<string> SetupQueue(string queueName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up queue {queueName}.");

        try
        {
            var queueInfo = await _sqsClient.GetQueueUrlAsync(new
GetQueueUrlRequest()
            { QueueName = queueName });
            return queueInfo.QueueUrl;
        }
        catch (QueueDoesNotExistException)
        {
            var attrs = new Dictionary<string, string>
```

```
            {
                {
                    QueueAttributeName.DelaySeconds,
                    "5"
                },
                {
                    QueueAttributeName.ReceiveMessageWaitTimeSeconds,
                    "5"
                },
                {
                    QueueAttributeName.VisibilityTimeout,
                    "300"
                },
            };

            var request = new CreateQueueRequest
            {
                Attributes = attrs,
                QueueName = queueName,
            };

            var response = await _sqsClient.CreateQueueAsync(request);
            Thread.Sleep(10000);
            await ConnectLambda(response.QueueUrl);
            Console.WriteLine($"\tQueue ready with Url {response.QueueUrl}.");
            Console.WriteLine(new string('-', 80));
            return response.QueueUrl;
        }
    }

    /// <summary>
    /// Connect the queue to the Lambda function as an event source.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue.</param>
    /// <returns>Async task.</returns>
    public static async Task ConnectLambda(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Connecting the Lambda function and queue for the
pipeline.");

        var queueAttributes = await _sqsClient.GetQueueAttributesAsync(
            new GetQueueAttributesRequest() { QueueUrl = queueUrl, AttributeNames =
new List<string>() { "All" } });
```

```
        var queueArn = queueAttributes.QueueARN;

        var eventSource = await _lambdaClient.ListEventSourceMappingsAsync(
             new ListEventSourceMappingsRequest()
             {
                 FunctionName = lambdaFunctionName
             });

        if (!eventSource.EventSourceMappings.Any())
        {
            // Only add the event source mapping if it does not already exist.
            await _lambdaClient.CreateEventSourceMappingAsync(
                new CreateEventSourceMappingRequest()
                {
                    EventSourceArn = queueArn,
                    FunctionName = lambdaFunctionName,
                    Enabled = true
                });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the bucket to use for pipeline input and output.
    /// </summary>
    /// <param name="bucketName">The name for the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task SetupBucket(string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up bucket {bucketName}.");

        var bucketExists = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            bucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = bucketName,
                BucketRegion = S3Region.USWest2
            });
```

```
            Thread.Sleep(5000);

            await _s3Client.PutObjectAsync(new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = "samplefiles/latlongtest.csv",
                FilePath = "latlongtest.csv"
            });
        }

        Console.WriteLine($"\tBucket {bucketName} ready.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Display some results from the output directory.
    /// </summary>
    /// <param name="bucketName">The name for the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task<string> GetOutputResults(string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Getting output results {bucketName}.");
        string outputKey = "";
        Thread.Sleep(15000);
        var outputFiles = await _s3Client.ListObjectsAsync(
              new ListObjectsRequest()
              {
                  BucketName = bucketName,
                  Prefix = "outputfiles/"
              });

        if (outputFiles.S3Objects.Any())
        {
            var sampleOutput = outputFiles.S3Objects.OrderBy(s =>
s.LastModified).Last();
            Console.WriteLine($"\tOutput file: {sampleOutput.Key}");
            var outputSampleResponse = await _s3Client.GetObjectAsync(
                new GetObjectRequest()
                {
                    BucketName = bucketName,
                    Key = sampleOutput.Key
                });
```

```
            outputKey = sampleOutput.Key;
            StreamReader reader = new
 StreamReader(outputSampleResponse.ResponseStream);
            await reader.ReadLineAsync();
            Console.WriteLine("\tOutput file contents: \n");
            for (int i = 0; i < 10; i++)
            {
                if (!reader.EndOfStream)
                {
                    Console.WriteLine("\t" + await reader.ReadLineAsync());
                }
            }
        }

        Console.WriteLine(new string('-', 80));
        return outputKey;
    }

    /// <summary>
    /// Create a pipeline from the example pipeline JSON
    /// that includes the Lambda, callback, processing, and export jobs.
    /// </summary>
    /// <param name="roleArn">The ARN of the role for the pipeline.</param>
    /// <param name="functionArn">The ARN of the Lambda function for the pipeline.</
param>
    /// <param name="pipelineName">The name for the pipeline.</param>
    /// <returns>The ARN of the pipeline.</returns>
    public static async Task<string> SetupPipeline(string roleArn, string
 functionArn, string pipelineName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up the pipeline.");

        var pipelineJson = await File.ReadAllTextAsync("GeoSpatialPipeline.json");

        // Add the correct function ARN instead of the placeholder.
        pipelineJson = pipelineJson.Replace("*FUNCTION_ARN*", functionArn);

        var pipelineArn = await _sageMakerWrapper.SetupPipeline(pipelineJson,
 roleArn, pipelineName,
            "sdk example pipeline", pipelineName);

        Console.WriteLine($"\tPipeline set up with ARN {pipelineArn}.");
        Console.WriteLine(new string('-', 80));
```

```
        return pipelineArn;
    }

    /// <summary>
    /// Start a pipeline run with job configurations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue used in the pipeline.</param>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>The pipeline run ARN.</returns>
    public static async Task<string> ExecutePipeline(
        string queueUrl,
        string roleArn,
        string pipelineName,
        string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Starting pipeline execution.");

        var input = $"s3://{bucketName}/samplefiles/latlongtest.csv";
        var output = $"s3://{bucketName}/outputfiles/";

        var executionARN =
            await _sageMakerWrapper.ExecutePipeline(queueUrl, input, output,
                pipelineName, roleArn);

        Console.WriteLine($"\tRun started with ARN {executionARN}.");
        Console.WriteLine(new string('-', 80));

        return executionARN;
    }

    /// <summary>
    /// Wait for a pipeline run to complete.
    /// </summary>
    /// <param name="executionArn">The pipeline run ARN.</param>
    /// <returns>Async task.</returns>
    public static async Task WaitForPipelineExecution(string executionArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Waiting for pipeline to finish.");
```

```
        PipelineExecutionStatus status;
        do
        {
            status = await
 _sageMakerWrapper.CheckPipelineExecutionStatus(executionArn);
            Thread.Sleep(30000);
            Console.WriteLine($"\tStatus is {status}.");
        } while (status == PipelineExecutionStatus.Executing);

        Console.WriteLine($"\tPipeline finished with status {status}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="askUser">True to ask the user for cleanup.</param>
    /// <param name="queueUrl">The URL of the queue to clean up.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> CleanupResources(
        bool askUser,
        string queueUrl,
        string pipelineName,
        string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (!askUser || GetYesNoResponse($"\tDelete pipeline {pipelineName}? (y/
n)"))
        {
            Console.WriteLine($"\tDeleting pipeline.");
            // Delete the pipeline.
            await _sageMakerWrapper.DeletePipelineByName(pipelineName);
        }

        if (!string.IsNullOrEmpty(queueUrl) && (!askUser ||
 GetYesNoResponse($"\tDelete queue {queueUrl}? (y/n)")))
        {
            Console.WriteLine($"\tDeleting queue.");
            // Delete the queue.
            await _sqsClient.DeleteQueueAsync(new DeleteQueueRequest(queueUrl));
```

```
        }

        if (!askUser || GetYesNoResponse($"\tDelete Amazon S3 bucket {bucketName}?
 (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
 ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            if (deleteList.KeyCount > 0)
            {
                await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.S3Objects
                        .Select(o => new KeyVersion { Key = o.Key }).ToList()
                });
            }

            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete lambda {lambdaFunctionName}? (y/
n)"))
        {
            Console.WriteLine($"\tDeleting lambda function.");

            await _lambdaClient.DeleteFunctionAsync(new DeleteFunctionRequest()
            {
                FunctionName = lambdaFunctionName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete role {lambdaRoleName}? (y/n)"))
        {
            Console.WriteLine($"\tDetaching policies and deleting role.");
```

```
            foreach (var policy in lambdaRolePolicies)
            {
                await _iamClient!.DetachRolePolicyAsync(new
 DetachRolePolicyRequest()
                {
                    RoleName = lambdaRoleName,
                    PolicyArn = policy
                });
            }

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = lambdaRoleName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete role {sageMakerRoleName}? (y/
n)"))
        {
            Console.WriteLine($"\tDetaching policies and deleting role.");

            foreach (var policy in sageMakerRolePolicies)
            {
                await _iamClient!.DetachRolePolicyAsync(new
 DetachRolePolicyRequest()
                {
                    RoleName = sageMakerRoleName,
                    PolicyArn = policy
                });
            }

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = sageMakerRoleName
            });
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a role's ARN if it already exists.
    /// </summary>
```

```
    /// <param name="roleName">The name of the AWS Identity and Access Management
 (IAM) Role to look for.</param>
    /// <returns>The role ARN if it exists, otherwise an empty string.</returns>
    private static async Task<string> GetRoleArnIfExists(string roleName)
    {
        Console.WriteLine($"Checking for role named {roleName}.");

        try
        {
            var existingRole = await _iamClient.GetRoleAsync(new GetRoleRequest()
            {
                RoleName = lambdaRoleName
            });
            return existingRole.Role.Arn;
        }
        catch (NoSuchEntityException)
        {
            return string.Empty;
        }
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreatePipeline
  - DeletePipeline

- DescribePipelineExecution
- StartPipelineExecution
- UpdatePipeline

Exemplos do Secrets Manager usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetSecretValue`

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.SecretsManager;
using Amazon.SecretsManager.Model;
```

```
    /// <summary>
    /// This example uses the Amazon Web Service Secrets Manager to retrieve
    /// the secret value for the provided secret name.
    /// </summary>
    public class GetSecretValue
    {
        /// <summary>
        /// The main method initializes the necessary values and then calls
        /// the GetSecretAsync and DecodeString methods to get the decoded
        /// secret value for the secret named in secretName.
        /// </summary>
        public static async Task Main()
        {
            string secretName = "<<{{MySecretName}}>>";
            string secret;

            IAmazonSecretsManager client = new AmazonSecretsManagerClient();

            var response = await GetSecretAsync(client, secretName);

            if (response is not null)
            {
                secret = DecodeString(response);

                if (!string.IsNullOrEmpty(secret))
                {
                    Console.WriteLine($"The decoded secret value is: {secret}.");
                }
                else
                {
                    Console.WriteLine("No secret value was returned.");
                }
            }
        }

        /// <summary>
        /// Retrieves the secret value given the name of the secret to
        /// retrieve.
        /// </summary>
        /// <param name="client">The client object used to retrieve the secret
        /// value for the given secret name.</param>
        /// <param name="secretName">The name of the secret value to retrieve.</
param>
        /// <returns>The GetSecretValueReponse object returned by
```

```
        /// GetSecretValueAsync.</returns>
        public static async Task<GetSecretValueResponse> GetSecretAsync(
            IAmazonSecretsManager client,
            string secretName)
        {
            GetSecretValueRequest request = new GetSecretValueRequest()
            {
                SecretId = secretName,
                VersionStage = "AWSCURRENT", // VersionStage defaults to AWSCURRENT
if unspecified.
            };

            GetSecretValueResponse response = null;

            // For the sake of simplicity, this example handles only the most
            // general SecretsManager exception.
            try
            {
                response = await client.GetSecretValueAsync(request);
            }
            catch (AmazonSecretsManagerException e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }

            return response;
        }

        /// <summary>
        /// Decodes the secret returned by the call to GetSecretValueAsync and
        /// returns it to the calling program.
        /// </summary>
        /// <param name="response">A GetSecretValueResponse object containing
        /// the requested secret value returned by GetSecretValueAsync.</param>
        /// <returns>A string representing the decoded secret value.</returns>
        public static string DecodeString(GetSecretValueResponse response)
        {
            // Decrypts secret using the associated AWS Key Management Service
            // Customer Master Key (CMK.) Depending on whether the secret is a
            // string or binary value, one of these fields will be populated.
            if (response.SecretString is not null)
            {
                var secret = response.SecretString;
                return secret;
```

```
            }
            else if (response.SecretBinary is not null)
            {
                var memoryStream = response.SecretBinary;
                StreamReader reader = new StreamReader(memoryStream);
                string decodedBinarySecret =
System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(reader.ReadToEnd()));
                return decodedBinarySecret;
            }
            else
            {
                return string.Empty;
            }
        }
    }
```

- Para obter detalhes da API, consulte GetSecretValue em Referência AWS SDK para .NET da API.

Exemplos do Amazon SES usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateTemplate`

O código de exemplo a seguir mostra como usar `CreateTemplate`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an email template.
    /// </summary>
    /// <param name="name">Name of the template.</param>
    /// <param name="subject">Email subject.</param>
    /// <param name="text">Email body text.</param>
    /// <param name="html">Email HTML body text.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string name, string subject,
string text,
        string html)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.CreateTemplateAsync(
                new CreateTemplateRequest
                {
                    Template = new Template
                    {
                        TemplateName = name,
                        SubjectPart = subject,
                        TextPart = text,
                        HtmlPart = html
                    }
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine("CreateEmailTemplateAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

- Para obter detalhes da API, consulte CreateTemplate em Referência AWS SDK para .NET da API.

`DeleteIdentity`

O código de exemplo a seguir mostra como usar `DeleteIdentity`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an email identity.
    /// </summary>
    /// <param name="identityEmail">The identity email to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteIdentityAsync(string identityEmail)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteIdentityAsync(
                new DeleteIdentityRequest
                {
                    Identity = identityEmail
```

```
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteIdentityAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

- Para obter detalhes da API, consulte DeleteIdentity em Referência AWS SDK para .NET da API.

DeleteTemplate

O código de exemplo a seguir mostra como usar `DeleteTemplate`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an email template.
    /// </summary>
    /// <param name="templateName">Name of the template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteTemplateAsync(
```

```
                new DeleteTemplateRequest
                {
                    TemplateName = templateName
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteEmailTemplateAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

- Para obter detalhes da API, consulte DeleteTemplate em Referência AWS SDK para .NET da API.

`GetIdentityVerificationAttributes`

O código de exemplo a seguir mostra como usar `GetIdentityVerificationAttributes`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get identity verification status for an email.
    /// </summary>
    /// <returns>The verification status of the email.</returns>
    public async Task<VerificationStatus> GetIdentityStatusAsync(string email)
    {
        var result = VerificationStatus.TemporaryFailure;
        try
```

```
        {
            var response =
                await
_amazonSimpleEmailService.GetIdentityVerificationAttributesAsync(
                    new GetIdentityVerificationAttributesRequest
                    {
                        Identities = new List<string> { email }
                    });

            if (response.VerificationAttributes.ContainsKey(email))
                result = response.VerificationAttributes[email].VerificationStatus;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetIdentityStatusAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- Para obter detalhes da API, consulte GetIdentityVerificationAttributes em Referência AWS SDK para .NET da API.

`GetSendQuota`

O código de exemplo a seguir mostra como usar `GetSendQuota`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information on the current account's send quota.
```

```
    /// </summary>
    /// <returns>The send quota response data.</returns>
    public async Task<GetSendQuotaResponse> GetSendQuotaAsync()
    {
        var result = new GetSendQuotaResponse();
        try
        {
            var response = await _amazonSimpleEmailService.GetSendQuotaAsync(
                new GetSendQuotaRequest());
            result = response;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetSendQuotaAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- Para obter detalhes da API, consulte GetSendQuota em Referência AWS SDK para .NET da API.

`ListIdentities`

O código de exemplo a seguir mostra como usar `ListIdentities`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the identities of a specified type for the current account.
    /// </summary>
```

```
    /// <param name="identityType">IdentityType to list.</param>
    /// <returns>The list of identities.</returns>
    public async Task<List<string>> ListIdentitiesAsync(IdentityType identityType)
    {
        var result = new List<string>();
        try
        {
            var response = await _amazonSimpleEmailService.ListIdentitiesAsync(
                new ListIdentitiesRequest
                {
                    IdentityType = identityType
                });
            result = response.Identities;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListIdentitiesAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- Para obter detalhes da API, consulte ListIdentities em Referência AWS SDK para .NET da API.

`ListTemplates`

O código de exemplo a seguir mostra como usar `ListTemplates`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List email templates for the current account.
    /// </summary>
    /// <returns>A list of template metadata.</returns>
    public async Task<List<TemplateMetadata>> ListEmailTemplatesAsync()
    {
        var result = new List<TemplateMetadata>();
        try
        {
            var response = await _amazonSimpleEmailService.ListTemplatesAsync(
                new ListTemplatesRequest());
            result = response.TemplatesMetadata;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListEmailTemplatesAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- Para obter detalhes da API, consulte ListTemplates em Referência AWS SDK para .NET da API.

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    ///  Send an email by using Amazon SES.
    /// </summary>
    /// <param name="toAddresses">List of recipients.</param>
    /// <param name="ccAddresses">List of cc recipients.</param>
    /// <param name="bccAddresses">List of bcc recipients.</param>
    /// <param name="bodyHtml">Body of the email in HTML.</param>
    /// <param name="bodyText">Body of the email in plain text.</param>
    /// <param name="subject">Subject line of the email.</param>
    /// <param name="senderAddress">From address.</param>
    /// <returns>The messageId of the email.</returns>
    public async Task<string> SendEmailAsync(List<string> toAddresses,
        List<string> ccAddresses, List<string> bccAddresses,
        string bodyHtml, string bodyText, string subject, string senderAddress)
    {
        var messageId = "";
        try
        {
            var response = await _amazonSimpleEmailService.SendEmailAsync(
                new SendEmailRequest
                {
                    Destination = new Destination
                    {
                        BccAddresses = bccAddresses,
                        CcAddresses = ccAddresses,
                        ToAddresses = toAddresses
                    },
                    Message = new Message
                    {
                        Body = new Body
                        {
                            Html = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyHtml
                            },
                            Text = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyText
                            }
                        },
                        Subject = new Content
                        {
                            Charset = "UTF-8",
```

```
                            Data = subject
                        }
                    },
                    Source = senderAddress
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendEmailAsync failed with exception: " +
ex.Message);
        }

        return messageId;
    }
```

- Para obter detalhes da API, consulte SendEmail em Referência AWS SDK para .NET da API.

SendTemplatedEmail

O código de exemplo a seguir mostra como usar `SendTemplatedEmail`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Send an email using a template.
    /// </summary>
    /// <param name="sender">Address of the sender.</param>
    /// <param name="recipients">Addresses of the recipients.</param>
    /// <param name="templateName">Name of the email template.</param>
    /// <param name="templateDataObject">Data for the email template.</param>
    /// <returns>The messageId of the email.</returns>
```

```
    public async Task<string> SendTemplateEmailAsync(string sender, List<string>
recipients,
        string templateName, object templateDataObject)
    {
        var messageId = "";
        try
        {
            // Template data should be serialized JSON from either a class or a
dynamic object.
            var templateData = JsonSerializer.Serialize(templateDataObject);

            var response = await _amazonSimpleEmailService.SendTemplatedEmailAsync(
                new SendTemplatedEmailRequest
                {
                    Source = sender,
                    Destination = new Destination
                    {
                        ToAddresses = recipients
                    },
                    Template = templateName,
                    TemplateData = templateData
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendTemplateEmailAsync failed with exception: " +
ex.Message);
        }

        return messageId;
    }
```

- Para obter detalhes da API, consulte SendTemplatedEmail em Referência AWS SDK para .NET da API.

`VerifyEmailIdentity`

O código de exemplo a seguir mostra como usar `VerifyEmailIdentity`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Starts verification of an email identity. This request sends an email
    /// from Amazon SES to the specified email address. To complete
    /// verification, follow the instructions in the email.
    /// </summary>
    /// <param name="recipientEmailAddress">Email address to verify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyEmailIdentityAsync(string recipientEmailAddress)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.VerifyEmailIdentityAsync(
                new VerifyEmailIdentityRequest
                {
                    EmailAddress = recipientEmailAddress
                });

            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("VerifyEmailIdentityAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

- Para obter detalhes da API, consulte VerifyEmailIdentity em Referência AWS SDK para .NET da API.

Cenários

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para .NET

Mostra como usar a API .NET do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para .NET

Mostra como usar o AWS SDK para .NET para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful .NET.

- Integre um aplicativo web React com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Aurora.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para .NET

Mostra como usar a API .NET do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Exemplos da API Amazon SES v2 usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API v2 AWS SDK para .NET com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateContact`

O código de exemplo a seguir mostra como usar `CreateContact`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a contact and adds it to the specified contact list.
    /// </summary>
    /// <param name="emailAddress">The email address of the contact.</param>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
```

```
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }
```

- Para obter detalhes da API, consulte CreateContact em Referência AWS SDK para .NET da API.

`CreateContactList`

O código de exemplo a seguir mostra como usar `CreateContactList`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact list:
{ex.Message}");
        }
        return false;
    }
```

- Para obter detalhes da API, consulte CreateContactList em Referência AWS SDK para .NET da API.

`CreateEmailIdentity`

O código de exemplo a seguir mostra como usar `CreateEmailIdentity`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Creates an email identity (email address or domain) and starts the
verification process.
   /// </summary>
   /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
   /// <returns>The response from the CreateEmailIdentity operation.</returns>
   public async Task<CreateEmailIdentityResponse> CreateEmailIdentityAsync(string
emailIdentity)
   {
       var request = new CreateEmailIdentityRequest
       {
           EmailIdentity = emailIdentity
       };

       try
       {
           var response = await _sesClient.CreateEmailIdentityAsync(request);
           return response;
       }
       catch (AlreadyExistsException ex)
       {
           Console.WriteLine($"Email identity {emailIdentity} already exists.");
           Console.WriteLine(ex.Message);
           throw;
       }
       catch (ConcurrentModificationException ex)
```

```
        {
            Console.WriteLine($"The email identity {emailIdentity} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been exceeded.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email identity:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateEmailIdentity em Referência AWS SDK para .NET da API.

`CreateEmailTemplate`

O código de exemplo a seguir mostra como usar `CreateEmailTemplate`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
exists.");
            Console.WriteLine(ex.Message);
        }
        catch (LimitExceededException ex)
```

```
        {
            Console.WriteLine("The limit for email templates has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email template:
{ex.Message}");
        }

        return false;
    }
```

- Para obter detalhes da API, consulte CreateEmailTemplate em Referência AWS SDK para .NET da API.

`DeleteContactList`

O código de exemplo a seguir mostra como usar `DeleteContactList`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact list:
{ex.Message}");
        }

        return false;
    }
```

- Para obter detalhes da API, consulte DeleteContactList em Referência AWS SDK para .NET da API.

`DeleteEmailIdentity`

O código de exemplo a seguir mostra como usar `DeleteEmailIdentity`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
```

```
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email identity:
{ex.Message}");
        }

        return false;
    }
```

- Para obter detalhes da API, consulte DeleteEmailIdentity em Referência AWS SDK para .NET da API.

DeleteEmailTemplate

O código de exemplo a seguir mostra como usar `DeleteEmailTemplate`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };
```

```
        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email template:
{ex.Message}");
        }

        return false;
    }
```

- Para obter detalhes da API, consulte DeleteEmailTemplate em Referência AWS SDK para .NET da API.

`ListContacts`

O código de exemplo a seguir mostra como usar `ListContacts`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.ListContactsAsync(request);
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
 {ex.Message}");
        }

        return new List<Contact>();
    }
```

- Para obter detalhes da API, consulte ListContacts em Referência AWS SDK para .NET da API.

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email from.</
param>
    /// <param name="toEmailAddresses">The email addresses to send the email to.</
param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
 (optional).</param>
    /// <param name="templateData">The data to replace placeholders in the email
 template (optional).</param>
    /// <param name="contactListName">The name of the contact list for unsubscribe
 functionality (optional).</param>
    /// <returns>The MessageId response from the SendEmail operation.</returns>
    public async Task<string> SendEmailAsync(string fromEmailAddress, List<string>
 toEmailAddresses, string? subject,
        string? htmlContent, string? textContent, string? templateName = null,
 string? templateData = null, string? contactListName = null)
    {
        var request = new SendEmailRequest
        {
            FromEmailAddress = fromEmailAddress
        };

        if (toEmailAddresses.Any())
        {
            request.Destination = new Destination { ToAddresses =
 toEmailAddresses };
```

```
        }

        if (!string.IsNullOrEmpty(templateName))
        {
            request.Content = new EmailContent()
            {
                Template = new Template
                {
                    TemplateName = templateName,
                    TemplateData = templateData
                }
            };
        }
        else
        {
            request.Content = new EmailContent
            {
                Simple = new Message
                {
                    Subject = new Content { Data = subject },
                    Body = new Body
                    {
                        Html = new Content { Data = htmlContent },
                        Text = new Content { Data = textContent }
                    }
                }
            };
        }

        if (!string.IsNullOrEmpty(contactListName))
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
```

```
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
{ex.Message}");
        }

        return string.Empty;
    }
```

- Para obter detalhes da API, consulte SendEmail em Referência AWS SDK para .NET da API.

Cenários

Cenário de boletim informativo

O exemplo de código a seguir mostra como executar o cenário do boletim informativo da API v2 do Amazon SES.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace Sesv2Scenario;

public static class NewsletterWorkflow
{
    /*
      This scenario demonstrates how to use the Amazon Simple Email Service (SES) v2
 to send a coupon newsletter to a list of subscribers.
      The scenario performs the following tasks:

      1. Prepare the application:
         - Create a verified email identity for sending and replying to emails.
         - Create a contact list to store the subscribers' email addresses.
         - Create an email template for the coupon newsletter.

      2. Gather subscriber email addresses:
         - Prompt the user for a base email address.
```

```
        - Create 3 variants of the email address using subaddress extensions (e.g.,
user+ses-weekly-newsletter-1@example.com).
        - Add each variant as a contact to the contact list.
        - Send a welcome email to each new contact.

     3. Send the coupon newsletter:
        - Retrieve the list of contacts from the contact list.
        - Send the coupon newsletter using the email template to each contact.

     4. Monitor and review:
        - Provide instructions for the user to review the sending activity and
metrics in the AWS console.

     5. Clean up resources:
        - Delete the contact list (which also deletes all contacts within it).
        - Delete the email template.
        - Optionally delete the verified email identity.

   */

   public static SESv2Wrapper _sesv2Wrapper;
   public static string? _baseEmailAddress = null;
   public static string? _verifiedEmail = null;
   private static string _contactListName = "weekly-coupons-newsletter";
   private static string _templateName = "weekly-coupons";
   private static string _subject = "Weekly Coupons Newsletter";
   private static string _htmlContentFile = "coupon-newsletter.html";
   private static string _textContentFile = "coupon-newsletter.txt";
   private static string _htmlWelcomeFile = "welcome.html";
   private static string _textWelcomeFile = "welcome.txt";
   private static string _couponsDataFile = "sample_coupons.json";

   // Relative location of the resources folder.
   private static string _resourcesFilePathLocation = "../../../../resources/";

   public static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSimpleEmailServiceV2>()
                    .AddTransient<SESv2Wrapper>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon SES v2 Coupon Newsletter
Scenario.");
            Console.WriteLine("This scenario demonstrates how to use the Amazon
Simple Email Service (SES) v2 " +
                              "\r\nto send a coupon newsletter to a list of
subscribers.");

            // Prepare the application.
            var emailIdentity = await PrepareApplication();

            // Gather subscriber email addresses.
            await GatherSubscriberEmailAddresses(emailIdentity);

            // Send the coupon newsletter.
            await SendCouponNewsletter(emailIdentity);

            // Monitor and review.
            MonitorAndReview(true);

            // Clean up resources.
            await Cleanup(emailIdentity, true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon SES v2 Coupon Newsletter scenario is
complete.");
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sesv2Wrapper = host.Services.GetRequiredService<SESv2Wrapper>();
    }

    /// <summary>
    /// Set up the resources for the scenario.
    /// </summary>
    /// <returns>The email address of the verified identity.</returns>
    public static async Task<string?> PrepareApplication()
    {
        var htmlContent = await File.ReadAllTextAsync(_resourcesFilePathLocation +
_htmlContentFile);
        var textContent = await File.ReadAllTextAsync(_resourcesFilePathLocation +
_textContentFile);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("1. In this step, we will prepare the application:" +
                          "\r\n  - Create a verified email identity for sending and
replying to emails." +
                          "\r\n  - Create a contact list to store the subscribers'
email addresses." +
                          "\r\n  - Create an email template for the coupon
newsletter.\r\n");

        // Prompt the user for a verified email address.
        while (!IsEmail(_verifiedEmail))
        {
            Console.Write("Enter a verified email address or an email to verify: ");
            _verifiedEmail = Console.ReadLine();
        }

        try
        {
            // Create an email identity and start the verification process.
            await _sesv2Wrapper.CreateEmailIdentityAsync(_verifiedEmail);
            Console.WriteLine($"Identity {_verifiedEmail} created.");
        }
```

```
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Identity {_verifiedEmail} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email identity: {ex.Message}");
        }

        // Create a contact list.
        try
        {
            await _sesv2Wrapper.CreateContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Contact list {_contactListName} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating contact list: {ex.Message}");
        }

        // Create an email template.
        try
        {
            await _sesv2Wrapper.CreateEmailTemplateAsync(_templateName, _subject,
htmlContent, textContent);
            Console.WriteLine($"Email template {_templateName} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Email template {_templateName} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email template: {ex.Message}");
        }

        return _verifiedEmail;
    }

    /// <summary>
```

```
    /// Generate subscriber addresses and send welcome emails.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
 PrepareApplication.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> GatherSubscriberEmailAddresses(string
 fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("2. In Step 2, we will gather subscriber email addresses:"
 +
                          "\r\n  - Prompt the user for a base email address." +
                          "\r\n  - Create 3 variants of the email address using
 subaddress extensions (e.g., user+ses-weekly-newsletter-1@example.com)." +
                          "\r\n  - Add each variant as a contact to the contact
 list." +
                          "\r\n  - Send a welcome email to each new contact.\r\n");

        // Prompt the user for a base email address.
        while (!IsEmail(_baseEmailAddress))
        {
            Console.Write("Enter a base email address (e.g., user@example.com): ");
            _baseEmailAddress = Console.ReadLine();
        }

        // Create 3 variants of the email address using +ses-weekly-newsletter-1,
 +ses-weekly-newsletter-2, etc.
        var baseEmailAddressParts = _baseEmailAddress!.Split("@");
        for (int i = 1; i <= 3; i++)
        {
            string emailAddress = $"{baseEmailAddressParts[0]}+ses-weekly-
newsletter-{i}@{baseEmailAddressParts[1]}";

            try
            {
                // Create a contact with the email address in the contact list.
                await _sesv2Wrapper.CreateContactAsync(emailAddress,
 _contactListName);
                Console.WriteLine($"Contact {emailAddress} added to the
 {_contactListName} contact list.");
            }
            catch (AlreadyExistsException)
            {
```

```
                Console.WriteLine($"Contact {emailAddress} already exists in the
{_contactListName} contact list.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error creating contact {emailAddress}:
{ex.Message}");
                return false;
            }

            // Send a welcome email to the new contact.
            try
            {
                string subject = "Welcome to the Weekly Coupons Newsletter";
                string htmlContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _htmlWelcomeFile);
                string textContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _textWelcomeFile);

                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress, new
List<string> { emailAddress }, subject, htmlContent, textContent);
                Console.WriteLine($"Welcome email sent to {emailAddress}.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending welcome email to {emailAddress}:
{ex.Message}");
                return false;
            }

            // Wait 2 seconds before sending the next email (if the account is in
the SES Sandbox).
            await Task.Delay(2000);
        }

        return true;
    }

    /// <summary>
    ///  Send the coupon newsletter to the subscribers in the contact list.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <returns>True if successful.</returns>
```

```
    public static async Task<bool> SendCouponNewsletter(string fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("3. In this step, we will send the coupon newsletter:" +
                          "\r\n  - Retrieve the list of contacts from the contact
list." +
                          "\r\n  - Send the coupon newsletter using the email
template to each contact.\r\n");


        // Retrieve the list of contacts from the contact list.
        var contacts = await _sesv2Wrapper.ListContactsAsync(_contactListName);
        if (!contacts.Any())
        {
            Console.WriteLine($"No contacts found in the {_contactListName} contact
list.");
            return false;
        }

        // Load the coupon data from the sample_coupons.json file.
        string couponsData = await File.ReadAllTextAsync(_resourcesFilePathLocation
+ _couponsDataFile);

        // Send the coupon newsletter to each contact using the email template.
        try
        {
            foreach (var contact in contacts)
            {
                // To use the Contact List for list management, send to only one
address at a time.
                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress,
                    new List<string> { contact.EmailAddress },
                    null, null, null, _templateName, couponsData, _contactListName);
            }

            Console.WriteLine($"Coupon newsletter sent to contact list
{_contactListName}.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error sending coupon newsletter to contact list
{_contactListName}: {ex.Message}");
            return false;
        }
```

```
        return true;
    }

    /// <summary>
    /// Provide instructions for monitoring sending activity and metrics.
    /// </summary>
    /// <param name="interactive">True to run in interactive mode.</param>
    /// <returns>True if successful.</returns>
    public static bool MonitorAndReview(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("4. In step 4, we will monitor and review:" +
                          "\r\n  - Provide instructions for the user to review the
sending activity and metrics in the AWS console.\r\n");

        Console.WriteLine("Review your sending activity using the SES Homepage in
the AWS console.");
        Console.WriteLine("Press Enter to open the SES Homepage in your default
browser...");
        if (interactive)
        {
            Console.ReadLine();
            try
            {
                // Open the SES Homepage in the default browser.
                Process.Start(new ProcessStartInfo
                {
                    FileName = "https://console.aws.amazon.com/ses/home",
                    UseShellExecute = true
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error opening the SES Homepage: {ex.Message}");
                return false;
            }
        }

        Console.WriteLine("Review the sending activity and email metrics, then press
Enter to continue...");
        if (interactive)
            Console.ReadLine();
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources used in the scenario.
    /// </summary>
    /// <param name="verifiedEmailAddress">The verified email address from
 PrepareApplication.</param>
    /// <param name="interactive">True if interactive.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Cleanup(string verifiedEmailAddress, bool
 interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("5. Finally, we clean up resources:" +
                          "\r\n  - Delete the contact list (which also deletes all
 contacts within it)." +
                          "\r\n  - Delete the email template." +
                          "\r\n  - Optionally delete the verified email identity.\r
\n");

        Console.WriteLine("Cleaning up resources...");

        // Delete the contact list (this also deletes all contacts in the list).
        try
        {
            await _sesv2Wrapper.DeleteContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} deleted.");
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Contact list {_contactListName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting contact list {_contactListName}:
 {ex.Message}");
            return false;
        }

        // Delete the email template.
        try
        {
            await _sesv2Wrapper.DeleteEmailTemplateAsync(_templateName);
            Console.WriteLine($"Email template {_templateName} deleted.");
```

```
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Email template {_templateName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting email template {_templateName}:
{ex.Message}");
            return false;
        }

        // Ask the user if they want to delete the email identity.
        var deleteIdentity = !interactive ||
            GetYesNoResponse(
                $"Do you want to delete the email identity {verifiedEmailAddress}?
(y/n) ");
        if (deleteIdentity)
        {
            try
            {
                await _sesv2Wrapper.DeleteEmailIdentityAsync(verifiedEmailAddress);
                Console.WriteLine($"Email identity {verifiedEmailAddress}
deleted.");
            }
            catch (NotFoundException)
            {
                Console.WriteLine(
                    $"Email identity {verifiedEmailAddress} not found.");
            }
            catch (Exception ex)
            {
                Console.WriteLine(
                    $"Error deleting email identity {verifiedEmailAddress}:
{ex.Message}");
                return false;
            }
        }
        else
        {
            Console.WriteLine(
                $"Skipping deletion of email identity {verifiedEmailAddress}.");
        }
```

```
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Simple check to verify a string is an email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email.</returns>
    private static bool IsEmail(string? email)
    {
        if (string.IsNullOrEmpty(email))
            return false;
        return Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+\.[^@\s]+$",
 RegexOptions.IgnoreCase);
    }
}
```

Wrapper para operações de serviço.

```
using System.Net;
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;

namespace Sesv2Scenario;

/// <summary>
/// Wrapper class for Amazon Simple Email Service (SES) v2 operations.
/// </summary>
```

```
public class SESv2Wrapper
{

    private readonly IAmazonSimpleEmailServiceV2 _sesClient;

    /// <summary>
    /// Constructor for the SESv2Wrapper.
    /// </summary>
    /// <param name="sesClient">The injected SES v2 client.</param>
    public SESv2Wrapper(IAmazonSimpleEmailServiceV2 sesClient)
    {
        _sesClient = sesClient;
    }

    /// <summary>
    /// Creates a contact and adds it to the specified contact list.
    /// </summary>
    /// <param name="emailAddress">The email address of the contact.</param>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
 contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
 exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
```

```
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
```

```
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact list:
{ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates an email identity (email address or domain) and starts the
verification process.
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
    /// <returns>The response from the CreateEmailIdentity operation.</returns>
    public async Task<CreateEmailIdentityResponse> CreateEmailIdentityAsync(string
emailIdentity)
    {
        var request = new CreateEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.CreateEmailIdentityAsync(request);
            return response;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email identity {emailIdentity} already exists.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (ConcurrentModificationException ex)
```

```
        {
            Console.WriteLine($"The email identity {emailIdentity} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been exceeded.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email identity:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
```

```
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
exists.");
            Console.WriteLine(ex.Message);
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email template:
{ex.Message}");
        }

        return false;
    }

    /// <summary>
```

```
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being modified
 by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact list:
 {ex.Message}");
        }

        return false;
    }
```

```
    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email identity:
{ex.Message}");
        }

        return false;
    }
```

```
    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email template:
 {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
```

```
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.ListContactsAsync(request);
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
 {ex.Message}");
        }

        return new List<Contact>();
    }

    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email from.</
param>
    /// <param name="toEmailAddresses">The email addresses to send the email to.</
param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
 (optional).</param>
```

```
   /// <param name="templateData">The data to replace placeholders in the email
template (optional).</param>
   /// <param name="contactListName">The name of the contact list for unsubscribe
functionality (optional).</param>
   /// <returns>The MessageId response from the SendEmail operation.</returns>
   public async Task<string> SendEmailAsync(string fromEmailAddress, List<string>
toEmailAddresses, string? subject,
       string? htmlContent, string? textContent, string? templateName = null,
string? templateData = null, string? contactListName = null)
   {
       var request = new SendEmailRequest
       {
           FromEmailAddress = fromEmailAddress
       };

       if (toEmailAddresses.Any())
       {
           request.Destination = new Destination { ToAddresses =
toEmailAddresses };
       }

       if (!string.IsNullOrEmpty(templateName))
       {
           request.Content = new EmailContent()
           {
               Template = new Template
               {
                   TemplateName = templateName,
                   TemplateData = templateData
               }
           };
       }
       else
       {
           request.Content = new EmailContent
           {
               Simple = new Message
               {
                   Subject = new Content { Data = subject },
                   Body = new Body
                   {
                       Html = new Content { Data = htmlContent },
                       Text = new Content { Data = textContent }
                   }
```

```
                }
            };
        }

        if (!string.IsNullOrEmpty(contactListName))
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
 {ex.Message}");
        }

        return string.Empty;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Exemplos do Amazon SNS usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon SNS

O exemplo de código a seguir mostra como começar a usar o Amazon SNS.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
```

```
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para .NET da API.

Ações

`CheckIfPhoneNumberIsOptedOut`

O código de exemplo a seguir mostra como usar `CheckIfPhoneNumberIsOptedOut`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";
```

```
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" : "not
opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOut em Referência AWS SDK para .NET da API.

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico com um nome específico.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
```

```
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

Crie um tópico com um nome e atributos específicos de FIFO e desduplicação.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
```

```
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- Para obter detalhes da API, consulte CreateTopic em Referência AWS SDK para .NET da API.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um tópico por meio do respectivo ARN.

```
    /// <summary>
    /// Delete a topic by its topic ARN.
```

```
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTopic em Referência AWS SDK para .NET da API.

`GetTopicAttributes`

O código de exemplo a seguir mostra como usar `GetTopicAttributes`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-west-2:000000000000:ExampleSNSTopic";
```

```
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
        /// Given the ARN of the Amazon SNS topic, this method retrieves the topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
        public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- Para obter detalhes da API, consulte GetTopicAttributes em Referência AWS SDK para .NET da API.

`ListSubscriptions`

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example will retrieve a list of the existing Amazon Simple
/// Notification Service (Amazon SNS) subscriptions.
/// </summary>
public class ListSubscriptions
{
    public static async Task Main()
    {
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        Console.WriteLine("Enter a topic ARN to list subscriptions for a
specific topic, " +
                          "or press Enter to list subscriptions for all
topics.");
        var topicArn = Console.ReadLine();
        Console.WriteLine();

        var subscriptions = await GetSubscriptionsListAsync(client, topicArn);

        DisplaySubscriptionList(subscriptions);
```

```
        }

        /// <summary>
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults to
 null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string topicArn
 = null)
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
                }
            }
            else
            {
                var paginateAllSubscriptions =
 client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
 paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }
```

```
            return results;
        }

        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
                Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- Para obter detalhes da API, consulte ListSubscriptions em Referência AWS SDK para .NET da API.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// Lists the Amazon Simple Notification Service (Amazon SNS)
    /// topics for the current account.
    /// </summary>
    public class ListSNSTopics
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await GetTopicListAsync(client);
        }

        /// <summary>
        /// Retrieves the list of Amazon SNS topics in groups of up to 100
        /// topics.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the list of topics.</param>
        public static async Task GetTopicListAsync(IAmazonSimpleNotificationService
client)
        {
            // If there are more than 100 Amazon SNS topics, the call to
            // ListTopicsAsync will return a value to pass to the
            // method to retrieve the next 100 (or less) topics.
            string nextToken = string.Empty;

            do
            {
                var response = await client.ListTopicsAsync(nextToken);
                DisplayTopicsList(response.Topics);
                nextToken = response.NextToken;
            }
            while (!string.IsNullOrEmpty(nextToken));
        }
```

```
        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para .NET da API.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Publique uma mensagem em um tópico.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
```

```
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

Publique uma mensagem em um tópico com opções de grupo, duplicação e atributo.

```
    /// <summary>
    /// Publish messages using user settings.
```

```
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is a
sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must set
a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID for
this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID for
this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
```

```
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
 messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?", false);
        }
    }
```

Aplique as seleções do usuário à ação de publicação.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</param>
    /// <param name="attributeValue">The optional attribute value for the message.</
param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
```

```
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para .NET .

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
        /// <summary>
        /// Creates a new subscription to a topic.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object, used
        /// to create an Amazon SNS subscription.</param>
        /// <param name="topicArn">The ARN of the topic to subscribe to.</param>
        /// <returns>A SubscribeResponse object which includes the subscription
        /// ARN for the new subscription.</returns>
        public static async Task<SubscribeResponse> TopicSubscribeAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            SubscribeRequest request = new SubscribeRequest()
            {
                TopicArn = topicArn,
                ReturnSubscriptionArn = true,
                Protocol = "email",
                Endpoint = "recipient@example.com",
            };

            var response = await client.SubscribeAsync(request);

            return response;
        }
```

Inscreva uma fila em um tópico com filtros opcionais.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
```

```
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para .NET .

Cancelar assinatura

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cancele a assinatura de um tópico por meio de um ARN de assinatura.

```
    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
```

```
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK para .NET .

Cenários

Criação de uma aplicação do Amazon SNS

O exemplo de código a seguir mostra como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

SDK para .NET

Mostra como usar a API .NET do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
```

```
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para .NET .

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para .NET

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
/// <summary>
/// Console application to run a feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
```

```
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();
```

```
            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for configuring
the topic and the subscriptions for the 2 queues." +
```

```
                            $"\r\nYou can then post to the topic and see the results
 in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                            $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                            $"\r\nYou can then post to the topic and see the results
 in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended to
 the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic, deduplication
 is supported." +
                                $"\r\nDeduplication IDs are either set in the message
 or automatically generated " +
                                $"\r\nfrom content using a hash function.\r\n" +
                                $"\r\nIf a message is successfully published to an SNS
 FIFO topic, any message " +
                                $"\r\npublished and determined to have the same
 deduplication ID, " +
                                $"\r\nwithin the five-minute deduplication interval,
 is accepted but not delivered.\r\n" +
```

```
                                $"\r\nFor more information about deduplication, " +
                                $"\r\nsee https://docs.aws.amazon.com/sns/latest/dg/
fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName, _useFifoTopic,
 _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
 Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS queue:
 ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
 appended to the queue name.");
                }
```

```
                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
```

```
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
 queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
```

```
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" : "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }
```

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is a
sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must set
a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID for
this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID for
this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
```

```
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?", false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl, 10);
```

```
            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the queue
at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message> messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
```

```
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
{ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer = true)
```

```
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Crie uma classe que envolva operações do Amazon SQS.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
```

```
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;

    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
 useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
```

```
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await _amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string topicArn,
string queueUrl)
    {
```

```
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
                                                $"\"aws:SourceArn\": \"{topicArn}\""
 +
                                            "}" +
                                        "}" +
                                "}]" +
                            "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
```

```
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl, List<Message>
messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
```

```
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Crie uma classe que envolva operações do Amazon SNS.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };
```

```
        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
```

```
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
 { { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</param>
    /// <param name="attributeValue">The optional attribute value for the message.</
param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
```

```
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;
```

```
// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record, ILambdaContext
 context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Exemplos do Amazon SQS usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon SQS

O exemplo de código a seguir mostra como começar a usar o Amazon SQS.

SDK para .NET

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.SQS;
using Amazon.SQS.Model;
```

```
namespace SQSActions;

public static class HelloSQS
{
    static async Task Main(string[] args)
    {
        var sqsClient = new AmazonSQSClient();

        Console.WriteLine($"Hello Amazon SQS! Following are some of your queues:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five queues.
        var response = await sqsClient.ListQueuesAsync(
            new ListQueuesRequest()
            {
                MaxResults = 5
            });

        foreach (var queue in response.QueueUrls)
        {
            Console.WriteLine($"\tQueue Url: {queue}");
            Console.WriteLine();
        }
    }
}
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para .NET da API.

Ações

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma fila com um nome específico.

```
    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
```

```
        return createResponse.QueueUrl;
    }
```

Crie uma fila do Amazon SQS e envie uma mensagem para ela.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SQS;
    using Amazon.SQS.Model;

    public class CreateSendExample
    {
        // Specify your AWS Region (an example Region is shown).
        private static readonly string QueueName = "Example_Queue";
        private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
        private static IAmazonSQS client;

        public static async Task Main()
        {
            client = new AmazonSQSClient(ServiceRegion);
            var createQueueResponse = await CreateQueue(client, QueueName);

            string queueUrl = createQueueResponse.QueueUrl;

            Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
            {
                { "Title",   new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
                { "Author",  new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
                { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } },
            };

            string messageBody = "Information about current NY Times fiction
bestseller for week of 12/11/2016.";
```

```
            var sendMsgResponse = await SendMessage(client, queueUrl, messageBody,
 messageAttributes);
        }

        /// <summary>
        /// Creates a new Amazon SQS queue using the queue name passed to it
        /// in queueName.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueName">A string representing the name of the queue
        /// to create.</param>
        /// <returns>A CreateQueueResponse that contains information about the
        /// newly created queue.</returns>
        public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS client,
 string queueName)
        {
            var request = new CreateQueueRequest
            {
                QueueName = queueName,
                Attributes = new Dictionary<string, string>
                {
                    { "DelaySeconds", "60" },
                    { "MessageRetentionPeriod", "86400" },
                },
            };

            var response = await client.CreateQueueAsync(request);
            Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

            return response;
        }

        /// <summary>
        /// Sends a message to an SQS queue.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueUrl">The URL of the queue to which to send the
        /// message.</param>
        /// <param name="messageBody">A string representing the body of the
        /// message to be sent to the queue.</param>
        /// <param name="messageAttributes">Attributes for the message to be
        /// sent to the queue.</param>
```

```
        /// <returns>A SendMessageResponse object that contains information
        /// about the message that was sent.</returns>
        public static async Task<SendMessageResponse> SendMessage(
            IAmazonSQS client,
            string queueUrl,
            string messageBody,
            Dictionary<string, MessageAttributeValue> messageAttributes)
        {
            var sendMessageRequest = new SendMessageRequest
            {
                DelaySeconds = 10,
                MessageAttributes = messageAttributes,
                MessageBody = messageBody,
                QueueUrl = queueUrl,
            };

            var response = await client.SendMessageAsync(sendMessageRequest);
            Console.WriteLine($"Sent a message with id : {response.MessageId}");

            return response;
        }
    }
```

- Para obter detalhes da API, consulte CreateQueue em Referência AWS SDK para .NET da API.

`DeleteMessage`

O código de exemplo a seguir mostra como usar `DeleteMessage`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem de uma fila do Amazon SQS e, em seguida, exclua a mensagem.

```
        public static async Task Main()
        {
            // If the AWS Region you want to use is different from
            // the AWS Region defined for the default user, supply
            // the specify your AWS Region to the client constructor.
            var client = new AmazonSQSClient();
            string queueName = "Example_Queue";

            var queueUrl = await GetQueueUrl(client, queueName);
            Console.WriteLine($"The SQS queue's URL is {queueUrl}");

            var response = await ReceiveAndDeleteMessage(client, queueUrl);

            Console.WriteLine($"Message: {response.Messages[0]}");
        }

        /// <summary>
        /// Retrieve the queue URL for the queue named in the queueName
        /// property using the client object.
        /// </summary>
        /// <param name="client">The Amazon SQS client used to retrieve the
        /// queue URL.</param>
        /// <param name="queueName">A string representing  name of the queue
        /// for which to retrieve the URL.</param>
        /// <returns>The URL of the queue.</returns>
        public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
        {
            var request = new GetQueueUrlRequest
            {
                QueueName = queueName,
            };

            GetQueueUrlResponse response = await client.GetQueueUrlAsync(request);
            return response.QueueUrl;
        }

        /// <summary>
        /// Retrieves the message from the quque at the URL passed in the
        /// queueURL parameters using the client.
        /// </summary>
        /// <param name="client">The SQS client used to retrieve a message.</param>
        /// <param name="queueUrl">The URL of the queue from which to retrieve
```

```
        /// a message.</param>
        /// <returns>The response from the call to ReceiveMessageAsync.</returns>
        public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
        {
            // Receive a single message from the queue.
            var receiveMessageRequest = new ReceiveMessageRequest
            {
                AttributeNames = { "SentTimestamp" },
                MaxNumberOfMessages = 1,
                MessageAttributeNames = { "All" },
                QueueUrl = queueUrl,
                VisibilityTimeout = 0,
                WaitTimeSeconds = 0,
            };

            var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

            // Delete the received message from the queue.
            var deleteMessageRequest = new DeleteMessageRequest
            {
                QueueUrl = queueUrl,
                ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
            };

            await client.DeleteMessageAsync(deleteMessageRequest);

            return receiveMessageResponse;
        }
    }
```

- Para obter detalhes da API, consulte DeleteMessage em Referência AWS SDK para .NET da API.

DeleteMessageBatch

O código de exemplo a seguir mostra como usar `DeleteMessageBatch`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl, List<Message>
messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }
```

- Para obter detalhes da API, consulte DeleteMessageBatch em Referência AWS SDK para .NET da API.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma fila usando seu URL.

```
/// <summary>
/// Delete a queue by its URL.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteQueueByUrl(string queueUrl)
{
    var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
        new DeleteQueueRequest()
        {
            QueueUrl = queueUrl
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DeleteQueue em Referência AWS SDK para .NET da API.

`GetQueueAttributes`

O código de exemplo a seguir mostra como usar `GetQueueAttributes`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Get the ARN for a queue from its URL.
/// </summary>
/// <param name="queueUrl">The URL of the queue.</param>
/// <returns>The ARN of the queue.</returns>
public async Task<string> GetQueueArnByUrl(string queueUrl)
{
    var getAttributesRequest = new GetQueueAttributesRequest()
    {
        QueueUrl = queueUrl,
        AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
    };

    var getAttributesResponse = await _amazonSQSClient.GetQueueAttributesAsync(
        getAttributesRequest);

    return getAttributesResponse.QueueARN;
}
```

- Para obter detalhes da API, consulte GetQueueAttributes em Referência AWS SDK para .NET da API.

`GetQueueUrl`

O código de exemplo a seguir mostra como usar `GetQueueUrl`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class GetQueueUrl
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// GetQueueUrlAsync method to retrieve the URL of an Amazon SQS
    /// queue.
    /// </summary>
    public static async Task Main()
    {
        // If the Amazon SQS message queue is not in the same AWS Region as your
        // default user, you need to provide the AWS Region as a parameter to
the
        // client constructor.
        var client = new AmazonSQSClient();

        string queueName = "New-Example-Queue";

        try
        {
            var response = await client.GetQueueUrlAsync(queueName);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The URL for {queueName} is:
{response.QueueUrl}");
            }
        }
        catch (QueueDoesNotExistException ex)
        {
```

```
                Console.WriteLine(ex.Message);
                Console.WriteLine($"The queue {queueName} was not found.");
            }
        }
    }
```

- Para obter detalhes da API, consulte GetQueueUrl em Referência AWS SDK para .NET da API.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba mensagens de uma fila usando seu URL.

```
    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
```

```
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }
```

Receba uma mensagem de uma fila do Amazon SQS e, em seguida, exclua a mensagem.

```
        public static async Task Main()
        {
            // If the AWS Region you want to use is different from
            // the AWS Region defined for the default user, supply
            // the specify your AWS Region to the client constructor.
            var client = new AmazonSQSClient();
            string queueName = "Example_Queue";

            var queueUrl = await GetQueueUrl(client, queueName);
            Console.WriteLine($"The SQS queue's URL is {queueUrl}");

            var response = await ReceiveAndDeleteMessage(client, queueUrl);

            Console.WriteLine($"Message: {response.Messages[0]}");
        }

        /// <summary>
        /// Retrieve the queue URL for the queue named in the queueName
        /// property using the client object.
        /// </summary>
        /// <param name="client">The Amazon SQS client used to retrieve the
        /// queue URL.</param>
        /// <param name="queueName">A string representing  name of the queue
        /// for which to retrieve the URL.</param>
        /// <returns>The URL of the queue.</returns>
        public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
        {
            var request = new GetQueueUrlRequest
            {
                QueueName = queueName,
            };

            GetQueueUrlResponse response = await client.GetQueueUrlAsync(request);
            return response.QueueUrl;
```

```
        }

        /// <summary>
        /// Retrieves the message from the quque at the URL passed in the
        /// queueURL parameters using the client.
        /// </summary>
        /// <param name="client">The SQS client used to retrieve a message.</param>
        /// <param name="queueUrl">The URL of the queue from which to retrieve
        /// a message.</param>
        /// <returns>The response from the call to ReceiveMessageAsync.</returns>
        public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
        {
            // Receive a single message from the queue.
            var receiveMessageRequest = new ReceiveMessageRequest
            {
                AttributeNames = { "SentTimestamp" },
                MaxNumberOfMessages = 1,
                MessageAttributeNames = { "All" },
                QueueUrl = queueUrl,
                VisibilityTimeout = 0,
                WaitTimeSeconds = 0,
            };

            var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

            // Delete the received message from the queue.
            var deleteMessageRequest = new DeleteMessageRequest
            {
                QueueUrl = queueUrl,
                ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
            };

            await client.DeleteMessageAsync(deleteMessageRequest);

            return receiveMessageResponse;
        }
    }
```

- Para obter detalhes da API, consulte ReceiveMessage em Referência AWS SDK para .NET da API.

SendMessage

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para .NET

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma fila do Amazon SQS e envie uma mensagem para ela.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.SQS;
using Amazon.SQS.Model;

public class CreateSendExample
{
    // Specify your AWS Region (an example Region is shown).
    private static readonly string QueueName = "Example_Queue";
    private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
    private static IAmazonSQS client;

    public static async Task Main()
    {
        client = new AmazonSQSClient(ServiceRegion);
        var createQueueResponse = await CreateQueue(client, QueueName);

        string queueUrl = createQueueResponse.QueueUrl;

        Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
        {
```

```
                { "Title",   new MessageAttributeValue { DataType = "String",
 StringValue = "The Whistler" } },
                { "Author",  new MessageAttributeValue { DataType = "String",
 StringValue = "John Grisham" } },
                { "WeeksOn", new MessageAttributeValue { DataType = "Number",
 StringValue = "6" } },
            };

            string messageBody = "Information about current NY Times fiction
 bestseller for week of 12/11/2016.";

            var sendMsgResponse = await SendMessage(client, queueUrl, messageBody,
 messageAttributes);
        }

        /// <summary>
        /// Creates a new Amazon SQS queue using the queue name passed to it
        /// in queueName.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueName">A string representing the name of the queue
        /// to create.</param>
        /// <returns>A CreateQueueResponse that contains information about the
        /// newly created queue.</returns>
        public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS client,
 string queueName)
        {
            var request = new CreateQueueRequest
            {
                QueueName = queueName,
                Attributes = new Dictionary<string, string>
                {
                    { "DelaySeconds", "60" },
                    { "MessageRetentionPeriod", "86400" },
                },
            };

            var response = await client.CreateQueueAsync(request);
            Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

            return response;
        }
```

```
        /// <summary>
        /// Sends a message to an SQS queue.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueUrl">The URL of the queue to which to send the
        /// message.</param>
        /// <param name="messageBody">A string representing the body of the
        /// message to be sent to the queue.</param>
        /// <param name="messageAttributes">Attributes for the message to be
        /// sent to the queue.</param>
        /// <returns>A SendMessageResponse object that contains information
        /// about the message that was sent.</returns>
        public static async Task<SendMessageResponse> SendMessage(
            IAmazonSQS client,
            string queueUrl,
            string messageBody,
            Dictionary<string, MessageAttributeValue> messageAttributes)
        {
            var sendMessageRequest = new SendMessageRequest
            {
                DelaySeconds = 10,
                MessageAttributes = messageAttributes,
                MessageBody = messageBody,
                QueueUrl = queueUrl,
            };

            var response = await client.SendMessageAsync(sendMessageRequest);
            Console.WriteLine($"Sent a message with id : {response.MessageId}");

            return response;
        }
    }
```

- Para obter detalhes da API, consulte SendMessage em Referência AWS SDK para .NET da API.

`SetQueueAttributes`

O código de exemplo a seguir mostra como usar `SetQueueAttributes`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Definir o atributo de política de uma fila para um tópico.

```
    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string topicArn,
string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
                                                $"\"aws:SourceArn\": \"{topicArn}\""
+
                                            "}" +
                                        "}" +
                                "}]" +
                            "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
```

```
                Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte SetQueueAttributes em Referência AWS SDK para .NET da API.

Cenários

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
/// <summary>
/// Console application to run a feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;
```

```
    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
"sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
```

```
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
 subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for configuring
 the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the results
 in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the results
 in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended to
 the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic, deduplication
 is supported." +
```

```
                                $"\r\nDeduplication IDs are either set in the message
 or automatically generated " +
                                $"\r\nfrom content using a hash function.\r\n" +
                                $"\r\nIf a message is successfully published to an SNS
 FIFO topic, any message " +
                                $"\r\npublished and determined to have the same
 deduplication ID, " +
                                $"\r\nwithin the five-minute deduplication interval,
 is accepted but not delivered.\r\n" +
                                $"\r\nFor more information about deduplication, " +
                                $"\r\nsee https://docs.aws.amazon.com/sns/latest/dg/
fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName, _useFifoTopic,
 _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
 Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS queue:
 ", $"example-queue-{i}");
```

```
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
 queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
```

```
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" : "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
```

```
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is a
sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must set
a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID for
this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");
```

```
                    Console.WriteLine("Enter a deduplication ID for this message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID for
this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?", false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
```

```
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl, 10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the queue
at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message> messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
{ex.Message}.");
        }
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer = true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
```

```
    }
}
```

Crie uma classe que envolva operações do Amazon SQS.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;

    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
 useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
```

```
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await _amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }
```

```
    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string topicArn,
string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
                                                $"\"aws:SourceArn\": \"{topicArn}\""
+
                                            "}" +
                                        "}" +
                                "}]" +
                            "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
```

```
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl, List<Message>
 messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
 _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }
```

```
    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Crie uma classe que envolva operações do Amazon SNS.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
```

```
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
 subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
```

```
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
 filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
 { { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</param>
    /// <param name="attributeValue">The optional attribute value for the message.</
param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
```

```
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
```

```
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Use o comando AWS Estrutura de processamento de mensagens para.NET com Amazon SQS

O exemplo de código a seguir mostra como criar aplicativos que publicam e recebem mensagens do Amazon SQS usando o AWS Message Processing Framework for.NET.

SDK para .NET

Fornece um tutorial para a estrutura de processamento de AWS mensagens para.NET. O tutorial cria uma aplicação web que permite ao usuário publicar uma mensagem do Amazon SQS e uma aplicação de linha de comando que recebe a mensagem.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o tutorial completo no Guia do AWS SDK para .NET desenvolvedor e o exemplo em GitHub.

Serviços usados neste exemplo

- Amazon SQS

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }
```

```
        context.Logger.LogInformation("done");
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }

    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando o .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali
namespace sqsSample;

public class Function
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
 ILambdaContext context)
    {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
 SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
        context.Logger.LogInformation($"Processed message {message.Body}");
```

```
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

Exemplos de funções de etapa usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with Step Functions.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Step Functions

O exemplo de código a seguir mostra como começar a usar o Step Functions.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace StepFunctionsActions;

using Amazon.StepFunctions;
using Amazon.StepFunctions.Model;

public class HelloStepFunctions
{
    static async Task Main()
    {
        var stepFunctionsClient = new AmazonStepFunctionsClient();

        Console.Clear();
        Console.WriteLine("Welcome to AWS Step Functions");
        Console.WriteLine("Let's list up to 10 of your state machines:");
        var stateMachineListRequest = new ListStateMachinesRequest { MaxResults =
 10 };

        // Get information for up to 10 Step Functions state machines.
        var response = await
 stepFunctionsClient.ListStateMachinesAsync(stateMachineListRequest);

        if (response.StateMachines.Count > 0)
        {
            response.StateMachines.ForEach(stateMachine =>
            {
                Console.WriteLine($"State Machine Name: {stateMachine.Name}\tAmazon
 Resource Name (ARN): {stateMachine.StateMachineArn}");
            });
        }
        else
        {
            Console.WriteLine("\tNo state machines were found.");
        }
    }
}
```

- Para obter detalhes da API, consulte ListStateMachines em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma atividade.
- Criar uma máquina de estado a partir de uma definição da Amazon States Language que contenha a atividade criada anteriormente como uma etapa.
- Executar a máquina de estado e responder à atividade com entrada do usuário.
- Obtenha o status e a saída finais após a conclusão da execução e, em seguida, limpe os recursos.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
global using System.Text.Json;
global using Amazon.StepFunctions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using StepFunctionsActions;
global using LogLevel = Microsoft.Extensions.Logging.LogLevel;



using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using Amazon.StepFunctions.Model;

namespace StepFunctionsBasics;
```

```
public class StepFunctionsBasics
{
    private static ILogger _logger = null!;
    private static IConfigurationRoot _configuration = null!;
    private static IAmazonIdentityManagementService _iamService = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Step Functions.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonStepFunctions>()
                    .AddAWSService<IAmazonIdentityManagementService>()
                    .AddTransient<StepFunctionsWrapper>()
            )
            .Build();

        _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<StepFunctionsBasics>();

        // Load configuration settings.
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var activityName = _configuration["ActivityName"];
        var stateMachineName = _configuration["StateMachineName"];

        var roleName = _configuration["RoleName"];
        var repoBaseDir = _configuration["RepoBaseDir"];
        var jsonFilePath = _configuration["JsonFilePath"];
        var jsonFileName = _configuration["JsonFileName"];

        var uiMethods = new UiMethods();
```

```
        var stepFunctionsWrapper =
 host.Services.GetRequiredService<StepFunctionsWrapper>();

        _iamService =
 host.Services.GetRequiredService<IAmazonIdentityManagementService>();

        // Load definition for the state machine from a JSON file.
        var stateDefinitionJson = File.ReadAllText($"{repoBaseDir}{jsonFilePath}
{jsonFileName}");

        Console.Clear();
        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("Create activity");
        Console.WriteLine("Let's start by creating an activity.");
        string activityArn;
        string stateMachineArn;

        // Check to see if the activity already exists.
        var activityList = await stepFunctionsWrapper.ListActivitiesAsync();
        var existingActivity = activityList.FirstOrDefault(activity => activity.Name
 == activityName);
        if (existingActivity is not null)
        {
            activityArn = existingActivity.ActivityArn;
            Console.WriteLine($"Activity, {activityName}, already exists.");
        }
        else
        {
            activityArn = await stepFunctionsWrapper.CreateActivity(activityName);
        }

        // Swap the placeholder in the JSON file with the Amazon Resource Name (ARN)
        // of the recently created activity.
        var stateDefinition =
 stateDefinitionJson.Replace("{{DOC_EXAMPLE_ACTIVITY_ARN}}", activityArn);

        uiMethods.DisplayTitle("Create state machine");
        Console.WriteLine("Now we'll create a state machine.");

        // Find or create an IAM role that can be assumed by Step Functions.
        var role = await GetOrCreateStateMachineRole(roleName);
```

```
        // See if the state machine already exists.
        var stateMachineList = await stepFunctionsWrapper.ListStateMachinesAsync();
        var existingStateMachine =
            stateMachineList.FirstOrDefault(stateMachine => stateMachine.Name ==
stateMachineName);
        if (existingStateMachine is not null)
        {
            Console.WriteLine($"State machine, {stateMachineName}, already
exists.");
            stateMachineArn = existingStateMachine.StateMachineArn;
        }
        else
        {
            // Create the state machine.
            stateMachineArn =
                await stepFunctionsWrapper.CreateStateMachine(stateMachineName,
stateDefinition, role.Arn);
            uiMethods.PressEnter();
        }

        Console.WriteLine("The state machine has been created.");
        var describeStateMachineResponse = await
stepFunctionsWrapper.DescribeStateMachineAsync(stateMachineArn);


Console.WriteLine($"{describeStateMachineResponse.Name}\t{describeStateMachineResponse.Stat
        Console.WriteLine($"Current status: {describeStateMachineResponse.Status}");
        Console.WriteLine($"Amazon Resource Name (ARN) of the role assumed by the
state machine: {describeStateMachineResponse.RoleArn}");

        var userName = string.Empty;
        Console.Write("Before we start the state machine, tell me what should
ChatSFN call you? ");
        userName = Console.ReadLine();

        // Keep asking until the user enters a string value.
        while (string.IsNullOrEmpty(userName))
        {
            Console.Write("Enter your name: ");
            userName = Console.ReadLine();
        }

        var executionJson = @"{""name"": """ + userName + @"""}";
```

```
        // Start the state machine execution.
        Console.WriteLine("Now we'll start execution of the state machine.");
        var executionArn = await
stepFunctionsWrapper.StartExecutionAsync(executionJson, stateMachineArn);
        Console.WriteLine("State machine started.");

        Console.WriteLine($"Thank you, {userName}. Now let's get started...");
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("ChatSFN");

        var isDone = false;
        var response = new GetActivityTaskResponse();
        var taskToken = string.Empty;
        var userChoice = string.Empty;

        while (!isDone)
        {
            response = await stepFunctionsWrapper.GetActivityTaskAsync(activityArn,
"MvpWorker");
            taskToken = response.TaskToken;

            // Parse the returned JSON string.
            var taskJsonResponse = JsonDocument.Parse(response.Input);
            var taskJsonObject = taskJsonResponse.RootElement;
            var message = taskJsonObject.GetProperty("message").GetString();
            var actions =
taskJsonObject.GetProperty("actions").EnumerateArray().Select(x =>
x.ToString()).ToList();
            Console.WriteLine($"\n{message}\n");

            // Prompt the user for another choice.
            Console.WriteLine("ChatSFN: What would you like me to do?");
            actions.ForEach(action => Console.WriteLine($"\t{action}"));
            Console.Write($"\n{userName}, tell me your choice: ");
            userChoice = Console.ReadLine();
            if (userChoice?.ToLower() == "done")
            {
                isDone = true;
            }

            Console.WriteLine($"You have selected: {userChoice}");
            var jsonResponse = @"{""action"": """ + userChoice + @"""}";
```

```
            await stepFunctionsWrapper.SendTaskSuccessAsync(taskToken,
jsonResponse);
        }

        await stepFunctionsWrapper.StopExecution(executionArn);
        Console.WriteLine("Now we will wait for the execution to stop.");
        DescribeExecutionResponse executionResponse;
        do
        {
            executionResponse = await
stepFunctionsWrapper.DescribeExecutionAsync(executionArn);
        } while (executionResponse.Status == ExecutionStatus.RUNNING);

        Console.WriteLine("State machine stopped.");
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("State machine executions");
        Console.WriteLine("Now let's take a look at the execution values for the
state machine.");

        // List the executions.
        var executions = await
stepFunctionsWrapper.ListExecutionsAsync(stateMachineArn);

        uiMethods.DisplayTitle("Step function execution values");
        executions.ForEach(execution =>
        {
            Console.WriteLine($"{execution.Name}\t{execution.StartDate} to
{execution.StopDate}");
        });

        uiMethods.PressEnter();

        // Now delete the state machine and the activity.
        uiMethods.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the state machine...");

        await stepFunctionsWrapper.DeleteStateMachine(stateMachineArn);
        Console.WriteLine("State machine deleted.");

        Console.WriteLine("Deleting the activity...");
        await stepFunctionsWrapper.DeleteActivity(activityArn);
        Console.WriteLine("Activity deleted.");
```

```
        Console.WriteLine("The Amazon Step Functions scenario is now complete.");
    }

    static async Task<Role> GetOrCreateStateMachineRole(string roleName)
    {
        // Define the policy document for the role.
        var stateMachineRolePolicy = @"{
         ""Version"": ""2012-10-17"",
        ""Statement"": [{
            ""Sid"": """",
            ""Effect"": ""Allow"",
            ""Principal"": {
                ""Service"": ""states.amazonaws.com""},
            ""Action"": ""sts:AssumeRole""}]}";

        var role = new Role();
        var roleExists = false;

        try
        {
            var getRoleResponse = await _iamService.GetRoleAsync(new GetRoleRequest
{ RoleName = roleName });
            roleExists = true;
            role = getRoleResponse.Role;
        }
        catch (NoSuchEntityException)
        {
            // The role doesn't exist. Create it.
            Console.WriteLine($"Role, {roleName} doesn't exist. Creating it...");
        }

        if (!roleExists)
        {
            var request = new CreateRoleRequest
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = stateMachineRolePolicy,
            };

            var createRoleResponse = await _iamService.CreateRoleAsync(request);
            role = createRoleResponse.Role;
        }

        return role;
```

```
    }
}


namespace StepFunctionsBasics;

/// <summary>
/// Some useful methods to make screen display easier.
/// </summary>
public class UiMethods
{
    private readonly string _sepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Welcome to the AWS Step Functions Demo");

        Console.WriteLine("This example application will do the following:");
        Console.WriteLine("\t 1. Create an activity.");
        Console.WriteLine("\t 2. Create a state machine.");
        Console.WriteLine("\t 3. Start an execution.");
        Console.WriteLine("\t 4. Run the worker, then stop it.");
        Console.WriteLine("\t 5. List executions.");
        Console.WriteLine("\t 6. Clean up the resources created for the example.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue.");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter"></param>
    /// <returns></returns>
```

```
    private string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(_sepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(_sepBar);
    }
}
```

Defina uma classe que envolva ações de máquina de estado e atividade.

```
namespace StepFunctionsActions;

using Amazon.StepFunctions;
using Amazon.StepFunctions.Model;

/// <summary>
/// Wrapper that performs AWS Step Functions actions.
/// </summary>
public class StepFunctionsWrapper
{
    private readonly IAmazonStepFunctions _amazonStepFunctions;

    /// <summary>
    /// The constructor for the StepFunctionsWrapper. Initializes the
    /// client object passed to it.
    /// </summary>
    /// <param name="amazonStepFunctions">An initialized Step Functions client
 object.</param>
```

```
    public StepFunctionsWrapper(IAmazonStepFunctions amazonStepFunctions)
    {
        _amazonStepFunctions = amazonStepFunctions;
    }

    /// <summary>
    /// Create a Step Functions activity using the supplied name.
    /// </summary>
    /// <param name="activityName">The name for the new Step Functions activity.</
param>
    /// <returns>The Amazon Resource Name (ARN) for the new activity.</returns>
    public async Task<string> CreateActivity(string activityName)
    {
        var response = await _amazonStepFunctions.CreateActivityAsync(new
 CreateActivityRequest { Name = activityName });
        return response.ActivityArn;
    }


    /// <summary>
    /// Create a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineName">Name for the new Step Functions state
    /// machine.</param>
    /// <param name="definition">A JSON string that defines the Step Functions
    /// state machine.</param>
    /// <param name="roleArn">The Amazon Resource Name (ARN) of the role.</param>
    /// <returns></returns>
    public async Task<string> CreateStateMachine(string stateMachineName, string
 definition, string roleArn)
    {
        var request = new CreateStateMachineRequest
        {
            Name = stateMachineName,
            Definition = definition,
            RoleArn = roleArn
        };

        var response =
            await _amazonStepFunctions.CreateStateMachineAsync(request);
        return response.StateMachineArn;
    }
```

```
    /// <summary>
    /// Delete a Step Machine activity.
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the activity.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteActivity(string activityArn)
    {
        var response = await _amazonStepFunctions.DeleteActivityAsync(new
DeleteActivityRequest { ActivityArn = activityArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// state machine.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteStateMachine(string stateMachineArn)
    {
        var response = await _amazonStepFunctions.DeleteStateMachineAsync(new
DeleteStateMachineRequest
        { StateMachineArn = stateMachineArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Retrieve information about the specified Step Functions execution.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of the
    /// Step Functions execution.</param>
    /// <returns>The API response returned by the API.</returns>
    public async Task<DescribeExecutionResponse> DescribeExecutionAsync(string
executionArn)
    {
        var response = await _amazonStepFunctions.DescribeExecutionAsync(new
DescribeExecutionRequest { ExecutionArn = executionArn });
        return response;
    }
```

```
    /// <summary>
    /// Retrieve information about the specified Step Functions state machine.
    /// </summary>
    /// <param name="StateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine to retrieve.</param>
    /// <returns>Information about the specified Step Functions state machine.</
returns>
    public async Task<DescribeStateMachineResponse> DescribeStateMachineAsync(string
 StateMachineArn)
    {
        var response = await _amazonStepFunctions.DescribeStateMachineAsync(new
 DescribeStateMachineRequest { StateMachineArn = StateMachineArn });
        return response;
    }


    /// <summary>
    /// Retrieve a task with the specified Step Functions activity
    /// with the specified Amazon Resource Name (ARN).
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the Step Functions activity.</param>
    /// <param name="workerName">The name of the Step Functions worker.</param>
    /// <returns>The response from the Step Functions activity.</returns>
    public async Task<GetActivityTaskResponse> GetActivityTaskAsync(string
 activityArn, string workerName)
    {
        var response = await _amazonStepFunctions.GetActivityTaskAsync(new
 GetActivityTaskRequest
        { ActivityArn = activityArn, WorkerName = workerName });
        return response;
    }


    /// <summary>
    /// List the Step Functions activities for the current account.
    /// </summary>
    /// <returns>A list of ActivityListItems.</returns>
    public async Task<List<ActivityListItem>> ListActivitiesAsync()
    {
        var request = new ListActivitiesRequest();
        var activities = new List<ActivityListItem>();

        do
```

```
        {
            var response = await _amazonStepFunctions.ListActivitiesAsync(request);

            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }

            activities.AddRange(response.Activities);
        }
        while (request.NextToken is not null);

        return activities;
    }


    /// <summary>
    /// Retrieve information about executions of a Step Functions
    /// state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>A list of ExecutionListItem objects.</returns>
    public async Task<List<ExecutionListItem>> ListExecutionsAsync(string
stateMachineArn)
    {
        var executions = new List<ExecutionListItem>();
        ListExecutionsResponse response;
        var request = new ListExecutionsRequest { StateMachineArn =
stateMachineArn };

        do
        {
            response = await _amazonStepFunctions.ListExecutionsAsync(request);
            executions.AddRange(response.Executions);
            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);

        return executions;
    }
```

```
    /// <summary>
    /// Retrieve a list of Step Functions state machines.
    /// </summary>
    /// <returns>A list of StateMachineListItem objects.</returns>
    public async Task<List<StateMachineListItem>> ListStateMachinesAsync()
    {
        var stateMachines = new List<StateMachineListItem>();
        var listStateMachinesPaginator =
            _amazonStepFunctions.Paginators.ListStateMachines(new
 ListStateMachinesRequest());

        await foreach (var response in listStateMachinesPaginator.Responses)
        {
            stateMachines.AddRange(response.StateMachines);
        }

        return stateMachines;
    }


    /// <summary>
    /// Indicate that the Step Functions task, indicated by the
    /// task token, has completed successfully.
    /// </summary>
    /// <param name="taskToken">Identifies the task.</param>
    /// <param name="taskResponse">The response received from executing the task.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SendTaskSuccessAsync(string taskToken, string
 taskResponse)
    {
        var response = await _amazonStepFunctions.SendTaskSuccessAsync(new
 SendTaskSuccessRequest
        { TaskToken = taskToken, Output = taskResponse });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start execution of an AWS Step Functions state machine.
    /// </summary>
    /// <param name="executionName">The name to use for the execution.</param>
```

```
    /// <param name="executionJson">The JSON string to pass for execution.</param>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>The Amazon Resource Name (ARN) of the AWS Step Functions
    /// execution.</returns>
    public async Task<string> StartExecutionAsync(string executionJson, string
 stateMachineArn)
    {
        var executionRequest = new StartExecutionRequest
        {
            Input = executionJson,
            StateMachineArn = stateMachineArn
        };

        var response = await
 _amazonStepFunctions.StartExecutionAsync(executionRequest);
        return response.ExecutionArn;
    }


    /// <summary>
    /// Stop execution of a Step Functions workflow.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of
    /// the Step Functions execution to stop.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StopExecution(string executionArn)
    {
        var response =
            await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
 { ExecutionArn = executionArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateActivity

- CreateStateMachine
- DeleteActivity
- DeleteStateMachine
- DescribeExecution
- DescribeStateMachine
- GetActivityTask
- ListActivities
- ListStateMachines
- SendTaskSuccess
- StartExecution
- StopExecution

Ações

`CreateActivity`

O código de exemplo a seguir mostra como usar `CreateActivity`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a Step Functions activity using the supplied name.
    /// </summary>
    /// <param name="activityName">The name for the new Step Functions activity.</
param>
    /// <returns>The Amazon Resource Name (ARN) for the new activity.</returns>
    public async Task<string> CreateActivity(string activityName)
    {
        var response = await _amazonStepFunctions.CreateActivityAsync(new
 CreateActivityRequest { Name = activityName });
        return response.ActivityArn;
```

```
    }
```

- Para obter detalhes da API, consulte CreateActivity em Referência AWS SDK para .NET da API.

`CreateStateMachine`

O código de exemplo a seguir mostra como usar `CreateStateMachine`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineName">Name for the new Step Functions state
    /// machine.</param>
    /// <param name="definition">A JSON string that defines the Step Functions
    /// state machine.</param>
    /// <param name="roleArn">The Amazon Resource Name (ARN) of the role.</param>
    /// <returns></returns>
    public async Task<string> CreateStateMachine(string stateMachineName, string
definition, string roleArn)
    {
        var request = new CreateStateMachineRequest
        {
            Name = stateMachineName,
            Definition = definition,
            RoleArn = roleArn
        };

        var response =
            await _amazonStepFunctions.CreateStateMachineAsync(request);
        return response.StateMachineArn;
```

```
    }
```

- Para obter detalhes da API, consulte CreateStateMachine em Referência AWS SDK para .NET da API.

DeleteActivity

O código de exemplo a seguir mostra como usar `DeleteActivity`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a Step Machine activity.
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the activity.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteActivity(string activityArn)
    {
        var response = await _amazonStepFunctions.DeleteActivityAsync(new
DeleteActivityRequest { ActivityArn = activityArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteActivity em Referência AWS SDK para .NET da API.

DeleteStateMachine

O código de exemplo a seguir mostra como usar `DeleteStateMachine`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// state machine.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteStateMachine(string stateMachineArn)
    {
        var response = await _amazonStepFunctions.DeleteStateMachineAsync(new
DeleteStateMachineRequest
        { StateMachineArn = stateMachineArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteStateMachine em Referência AWS SDK para .NET da API.

`DescribeExecution`

O código de exemplo a seguir mostra como usar `DescribeExecution`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
```

```
    /// Retrieve information about the specified Step Functions execution.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of the
    /// Step Functions execution.</param>
    /// <returns>The API response returned by the API.</returns>
    public async Task<DescribeExecutionResponse> DescribeExecutionAsync(string
 executionArn)
    {
        var response = await _amazonStepFunctions.DescribeExecutionAsync(new
 DescribeExecutionRequest { ExecutionArn = executionArn });
        return response;
    }
```

- Para obter detalhes da API, consulte DescribeExecution em Referência AWS SDK para .NET da API.

`DescribeStateMachine`

O código de exemplo a seguir mostra como usar `DescribeStateMachine`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve information about the specified Step Functions state machine.
    /// </summary>
    /// <param name="StateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine to retrieve.</param>
    /// <returns>Information about the specified Step Functions state machine.</
returns>
    public async Task<DescribeStateMachineResponse> DescribeStateMachineAsync(string
 StateMachineArn)
    {
```

```
        var response = await _amazonStepFunctions.DescribeStateMachineAsync(new
DescribeStateMachineRequest { StateMachineArn = StateMachineArn });
        return response;
    }
```

- Para obter detalhes da API, consulte DescribeStateMachine em Referência AWS SDK para .NET da API.

`GetActivityTask`

O código de exemplo a seguir mostra como usar `GetActivityTask`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve a task with the specified Step Functions activity
    /// with the specified Amazon Resource Name (ARN).
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the Step Functions activity.</param>
    /// <param name="workerName">The name of the Step Functions worker.</param>
    /// <returns>The response from the Step Functions activity.</returns>
    public async Task<GetActivityTaskResponse> GetActivityTaskAsync(string
activityArn, string workerName)
    {
        var response = await _amazonStepFunctions.GetActivityTaskAsync(new
GetActivityTaskRequest
        { ActivityArn = activityArn, WorkerName = workerName });
        return response;
    }
```

- Para obter detalhes da API, consulte GetActivityTask em Referência AWS SDK para .NET da API.

`ListActivities`

O código de exemplo a seguir mostra como usar `ListActivities`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// List the Step Functions activities for the current account.
/// </summary>
/// <returns>A list of ActivityListItems.</returns>
public async Task<List<ActivityListItem>> ListActivitiesAsync()
{
    var request = new ListActivitiesRequest();
    var activities = new List<ActivityListItem>();

    do
    {
        var response = await _amazonStepFunctions.ListActivitiesAsync(request);

        if (response.NextToken is not null)
        {
            request.NextToken = response.NextToken;
        }

        activities.AddRange(response.Activities);
    }
    while (request.NextToken is not null);

    return activities;
}
```

- Para obter detalhes da API, consulte ListActivities em Referência AWS SDK para .NET da API.

`ListExecutions`

O código de exemplo a seguir mostra como usar `ListExecutions`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve information about executions of a Step Functions
    /// state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>A list of ExecutionListItem objects.</returns>
    public async Task<List<ExecutionListItem>> ListExecutionsAsync(string
stateMachineArn)
    {
        var executions = new List<ExecutionListItem>();
        ListExecutionsResponse response;
        var request = new ListExecutionsRequest { StateMachineArn =
stateMachineArn };

        do
        {
            response = await _amazonStepFunctions.ListExecutionsAsync(request);
            executions.AddRange(response.Executions);
            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);

        return executions;
    }
```

- Para obter detalhes da API, consulte ListExecutions em Referência AWS SDK para .NET da API.

`ListStateMachines`

O código de exemplo a seguir mostra como usar `ListStateMachines`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve a list of Step Functions state machines.
    /// </summary>
    /// <returns>A list of StateMachineListItem objects.</returns>
    public async Task<List<StateMachineListItem>> ListStateMachinesAsync()
    {
        var stateMachines = new List<StateMachineListItem>();
        var listStateMachinesPaginator =
            _amazonStepFunctions.Paginators.ListStateMachines(new
ListStateMachinesRequest());

        await foreach (var response in listStateMachinesPaginator.Responses)
        {
            stateMachines.AddRange(response.StateMachines);
        }

        return stateMachines;
    }
```

- Para obter detalhes da API, consulte ListStateMachines em Referência AWS SDK para .NET da API.

`SendTaskSuccess`

O código de exemplo a seguir mostra como usar `SendTaskSuccess`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Indicate that the Step Functions task, indicated by the
    /// task token, has completed successfully.
    /// </summary>
    /// <param name="taskToken">Identifies the task.</param>
    /// <param name="taskResponse">The response received from executing the task.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SendTaskSuccessAsync(string taskToken, string
 taskResponse)
    {
        var response = await _amazonStepFunctions.SendTaskSuccessAsync(new
 SendTaskSuccessRequest
        { TaskToken = taskToken, Output = taskResponse });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte SendTaskSuccess em Referência AWS SDK para .NET da API.

`StartExecution`

O código de exemplo a seguir mostra como usar `StartExecution`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start execution of an AWS Step Functions state machine.
    /// </summary>
    /// <param name="executionName">The name to use for the execution.</param>
    /// <param name="executionJson">The JSON string to pass for execution.</param>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>The Amazon Resource Name (ARN) of the AWS Step Functions
    /// execution.</returns>
    public async Task<string> StartExecutionAsync(string executionJson, string
stateMachineArn)
    {
        var executionRequest = new StartExecutionRequest
        {
            Input = executionJson,
            StateMachineArn = stateMachineArn
        };

        var response = await
_amazonStepFunctions.StartExecutionAsync(executionRequest);
        return response.ExecutionArn;
    }
```

- Para obter detalhes da API, consulte StartExecution em Referência AWS SDK para .NET da API.

AWS STS exemplos usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
```

```
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        // A region property may be used if the profile or credentials loaded do not
 specify a region,
        // or to use a specific region.
        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials, REGION);
```

```
            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- Para obter detalhes da API, consulte AssumeRole em Referência AWS SDK para .NET da API.

Suporte exemplos usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET with Suporte.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Suporte

O exemplo de código a seguir mostra como começar a usar o Suporte.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.AWSSupport;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

public static class HelloSupport
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        // You must have one of the following AWS Support plans: Business,
 Enterprise On-Ramp, or Enterprise. Otherwise, an exception will be thrown.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonAWSSupport>()
            ).Build();

        // Now the client is available for injection.
        var supportClient = host.Services.GetRequiredService<IAmazonAWSSupport>();

        // You can use await and any of the async methods to get a response.
        var response = await supportClient.DescribeServicesAsync();
        Console.WriteLine($"\tHello AWS Support! There are {response.Services.Count}
 services available.");
    }
}
```

- Para obter detalhes da API, consulte DescribeServices em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Obter e exibir os serviços disponíveis e os níveis de gravidade dos casos.
- Criar um caso de suporte usando um serviço, uma categoria e um nível de gravidade selecionados.
- Obter e exibir uma lista de casos em aberto para o dia atual.
- Adicionar um conjunto de anexos e uma comunicação ao novo caso.
- Descrever o novo anexo e a comunicação para o caso.
- Resolver o caso.
- Obtenha e exiba uma lista de casos resolvidos para o dia atual.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
/// <summary>
/// Hello AWS Support example.
/// </summary>
public static class SupportCaseScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    To use the AWS Support API, you must have one of the following AWS Support
 plans: Business, Enterprise On-Ramp, or Enterprise.

    This .NET example performs the following tasks:
    1.  Get and display services. Select a service from the list.
```

```
   2.  Select a category from the selected service.
   3.  Get and display severity levels and select a severity level from the list.
   4.  Create a support case using the selected service, category, and severity
level.
   5.  Get and display a list of open support cases for the current day.
   6.  Create an attachment set with a sample text file to add to the case.
   7.  Add a communication with the attachment to the support case.
   8.  List the communications of the support case.
   9.  Describe the attachment set.
   10. Resolve the support case.
   11. Get a list of resolved cases for the current day.
  */

   private static SupportWrapper _supportWrapper = null!;

   static async Task Main(string[] args)
   {
       // Set up dependency injection for the AWS Support service.
       // Use your AWS profile name, or leave it blank to use the default profile.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonAWSSupport>(new AWSOptions() { Profile
= "default" })
                   .AddTransient<SupportWrapper>()
           )
           .Build();

       var logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger(typeof(SupportCaseScenario));

       _supportWrapper = host.Services.GetRequiredService<SupportWrapper>();

       Console.WriteLine(new string('-', 80));
       Console.WriteLine("Welcome to the AWS Support case example scenario.");
       Console.WriteLine(new string('-', 80));

       try
```

```
        {
            var apiSupported = await _supportWrapper.VerifySubscription();
            if (!apiSupported)
            {
                logger.LogError("You must have a Business, Enterprise On-Ramp, or
Enterprise Support " +
                                "plan to use the AWS Support API. \n\tPlease
upgrade your subscription to run these examples.");
                return;
            }

            var service = await DisplayAndSelectServices();

            var category = DisplayAndSelectCategories(service);

            var severityLevel = await DisplayAndSelectSeverity();

            var caseId = await CreateSupportCase(service, category, severityLevel);

            await DescribeTodayOpenCases();

            var attachmentSetId = await CreateAttachmentSet();

            await AddCommunicationToCase(attachmentSetId, caseId);

            var attachmentId = await ListCommunicationsForCase(caseId);

            await DescribeCaseAttachment(attachmentId);

            await ResolveCase(caseId);

            await DescribeTodayResolvedCases();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("AWS Support case example scenario complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
```

```
    /// List some available services from AWS Support, and select a service for the
example.
    /// </summary>
    /// <returns>The selected service.</returns>
    private static async Task<Service> DisplayAndSelectServices()
    {
        Console.WriteLine(new string('-', 80));
        var services = await _supportWrapper.DescribeServices();
        Console.WriteLine($"AWS Support client returned {services.Count}
services.");

        Console.WriteLine($"1. Displaying first 10 services:");
        for (int i = 0; i < 10 && i < services.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {services[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > services.Count)
        {
            Console.WriteLine(
                "Select an example support service by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        Console.WriteLine(new string('-', 80));

        return services[choiceNumber - 1];
    }

    /// <summary>
    /// List the available categories for a service and select a category for the
example.
    /// </summary>
    /// <param name="service">Service to use for displaying categories.</param>
    /// <returns>The selected category.</returns>
    private static Category DisplayAndSelectCategories(Service service)
    {
        Console.WriteLine(new string('-', 80));

        Console.WriteLine($"2. Available support categories for Service
\"{service.Name}\":");
        for (int i = 0; i < service.Categories.Count; i++)
```

```
        {
            Console.WriteLine($"\t{i + 1}. {service.Categories[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > service.Categories.Count)
        {
            Console.WriteLine(
                "Select an example support category by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        Console.WriteLine(new string('-', 80));

        return service.Categories[choiceNumber - 1];
    }

    /// <summary>
    /// List available severity levels from AWS Support, and select a level for the
example.
    /// </summary>
    /// <returns>The selected severity level.</returns>
    private static async Task<SeverityLevel> DisplayAndSelectSeverity()
    {
        Console.WriteLine(new string('-', 80));
        var severityLevels = await _supportWrapper.DescribeSeverityLevels();

        Console.WriteLine($"3. Get and display available severity levels:");
        for (int i = 0; i < 10 && i < severityLevels.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {severityLevels[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > severityLevels.Count)
        {
            Console.WriteLine(
                "Select an example severity level by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
```

```
        Console.WriteLine(new string('-', 80));

        return severityLevels[choiceNumber - 1];
    }

    /// <summary>
    /// Create an example support case.
    /// </summary>
    /// <param name="service">Service to use for the new case.</param>
    /// <param name="category">Category to use for the new case.</param>
    /// <param name="severity">Severity to use for the new case.</param>
    /// <returns>The caseId of the new support case.</returns>
    private static async Task<string> CreateSupportCase(Service service,
        Category category, SeverityLevel severity)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Create an example support case" +
                          $" with the following settings:" +
                          $" \n\tService: {service.Name}, Category: {category.Name}
" +
                          $"and Severity Level: {severity.Name}.");
        var caseId = await _supportWrapper.CreateCase(service.Code, category.Code,
severity.Code,
            "Example case for testing, ignore.", "This is my example support
case.");

        Console.WriteLine($"\tNew case created with ID {caseId}");

        Console.WriteLine(new string('-', 80));

        return caseId;
    }

    /// <summary>
    /// List open cases for the current day.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeTodayOpenCases()
    {
        Console.WriteLine($"5. List the open support cases for the current day.");
        // Describe the cases. If it is empty, try again and allow time for the new
case to appear.
        List<CaseDetails> currentOpenCases = null!;
        while (currentOpenCases == null || currentOpenCases.Count == 0)
```

```
        {
            Thread.Sleep(1000);
            currentOpenCases = await _supportWrapper.DescribeCases(
                new List<string>(),
                null,
                false,
                false,
                DateTime.UtcNow.Date,
                DateTime.UtcNow);
        }

        foreach (var openCase in currentOpenCases)
        {
            Console.WriteLine($"\tCase: {openCase.CaseId} created
{openCase.TimeCreated}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an attachment set for a support case.
    /// </summary>
    /// <returns>The attachment set id.</returns>
    private static async Task<string> CreateAttachmentSet()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Create an attachment set for a support case.");
        var fileName = "example_attachment.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for attachment to a support case.");
        }

        await using var ms = new MemoryStream(await
File.ReadAllBytesAsync(fileName));

        var attachmentSetId = await _supportWrapper.AddAttachmentToSet(
            ms,
            fileName);
```

```
        Console.WriteLine($"\tNew attachment set created with id: \n
\t{attachmentSetId.Substring(0, 65)}...");

        Console.WriteLine(new string('-', 80));

        return attachmentSetId;
    }

    /// <summary>
    /// Add an attachment set and communication to a case.
    /// </summary>
    /// <param name="attachmentSetId">Id of the attachment set.</param>
    /// <param name="caseId">Id of the case to receive the attachment set.</param>
    /// <returns>Async task.</returns>
    private static async Task AddCommunicationToCase(string attachmentSetId, string
 caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Add attachment set and communication to {caseId}.");

        await _supportWrapper.AddCommunicationToCase(
            caseId,
            "This is an example communication added to a support case.",
            attachmentSetId);

        Console.WriteLine($"\tNew attachment set and communication added to
 {caseId}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the communications for a case.
    /// </summary>
    /// <param name="caseId">Id of the case to describe.</param>
    /// <returns>An attachment id.</returns>
    private static async Task<string> ListCommunicationsForCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. List communications for case {caseId}.");

        var communications = await _supportWrapper.DescribeCommunications(caseId);
        var attachmentId = "";
```

```
        foreach (var communication in communications)
        {
            Console.WriteLine(
                $"\tCommunication created on: {communication.TimeCreated} has
{communication.AttachmentSet.Count} attachments.");
            if (communication.AttachmentSet.Any())
            {
                attachmentId = communication.AttachmentSet.First().AttachmentId;
            }
        }

        Console.WriteLine(new string('-', 80));
        return attachmentId;
    }

    /// <summary>
    /// Describe an attachment by id.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment to describe.</param>
    /// <returns>Async task.</returns>
    private static async Task DescribeCaseAttachment(string attachmentId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Describe the attachment set.");

        var attachment = await _supportWrapper.DescribeAttachment(attachmentId);
        var data = Encoding.ASCII.GetString(attachment.Data.ToArray());
        Console.WriteLine($"\tAttachment includes {attachment.FileName} with data:
\n\t{data}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Resolve the support case.
    /// </summary>
    /// <param name="caseId">Id of the case to resolve.</param>
    /// <returns>Async task.</returns>
    private static async Task ResolveCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Resolve case {caseId}.");

        var status = await _supportWrapper.ResolveCase(caseId);
```

```
        Console.WriteLine($"\tCase {caseId} has final status {status}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List resolved cases for the current day.
    /// </summary>
    /// <returns>Async Task.</returns>
    private static async Task DescribeTodayResolvedCases()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. List the resolved support cases for the current
 day.");
        var currentCases = await _supportWrapper.DescribeCases(
            new List<string>(),
            null,
            false,
            true,
            DateTime.UtcNow.Date,
            DateTime.UtcNow);

        foreach (var currentCase in currentCases)
        {
            if (currentCase.Status == "resolved")
            {
                Console.WriteLine(
                    $"\tCase: {currentCase.CaseId}: status {currentCase.Status}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }
}
```

Métodos de invólucro usados pelo cenário para Suporte ações.

```
/// <summary>
/// Wrapper methods to use AWS Support for working with support cases.
/// </summary>
public class SupportWrapper
```

```
{
    private readonly IAmazonAWSSupport _amazonSupport;
    public SupportWrapper(IAmazonAWSSupport amazonSupport)
    {
        _amazonSupport = amazonSupport;
    }


    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
 are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
            {
                Language = language
            });
        return response.Services;
    }



    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
 are supported.</param>
    /// <returns>The list of support severity levels.</returns>
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language =
 "en")
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
            {
                Language = language
            });
        return response.SeverityLevels;
    }
```

```
    /// <summary>
    /// Create a new support case.
    /// </summary>
    /// <param name="serviceCode">Service code for the new case.</param>
    /// <param name="categoryCode">Category for the new case.</param>
    /// <param name="severityCode">Severity code for the new case.</param>
    /// <param name="subject">Subject of the new case.</param>
    /// <param name="body">Body text of the new case.</param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set for the new
case.</param>
    /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
    /// <returns>The caseId of the new support case.</returns>
    public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
        string body, string language = "en", string? attachmentSetId = null, string
issueType = "customer-service")
    {
        var response = await _amazonSupport.CreateCaseAsync(
            new CreateCaseRequest()
            {
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
                CommunicationBody = body
            });
        return response.CaseId;
    }



    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does not
exist.
```

```
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates a
new attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string fileName,
string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
                    }
                }
            });
        return response.AttachmentSetId;
    }



    /// <summary>
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }
```

```
    /// <summary>
    /// Add communication to a case, including optional attachment set ID and CC
 email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }



    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
 DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
 _amazonSupport.Paginators.DescribeCommunications(
            new DescribeCommunicationsRequest()
            {
                CaseId = caseId,
```

```
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }



    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
    /// <param name="displayId">Optional display ID.</param>
    /// <param name="includeCommunication">True to include communication. Defaults
 to true.</param>
    /// <param name="includeResolvedCases">True to include resolved cases. Defaults
 to false.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
 are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds, string?
 displayId = null, bool includeCommunication = true,
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
 beforeTime = null,
        string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginators.DescribeCases(
            new DescribeCasesRequest()
            {
                CaseIdList = caseIds,
                DisplayId = displayId,
                IncludeCommunications = includeCommunication,
                IncludeResolvedCases = includeResolvedCases,
```

```
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s"),
                Language = language
            });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }



    /// <summary>
    /// Resolve a support case by caseId.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <returns>The final status of the case after resolving.</returns>
    public async Task<string> ResolveCase(string caseId)
    {
        var response = await _amazonSupport.ResolveCaseAsync(
            new ResolveCaseRequest()
            {
                CaseId = caseId
            });
        return response.FinalCaseStatus;
    }


    /// <summary>
    /// Verify the support level for AWS Support API access.
    /// </summary>
    /// <returns>True if the subscription level supports API access.</returns>
    public async Task<bool> VerifySubscription()
    {
        try
        {
            var response = await _amazonSupport.DescribeServicesAsync(
                new DescribeServicesRequest()
                {
                    Language = "en"
                });
            return response.HttpStatusCode == HttpStatusCode.OK;
```

```
        }
        catch (Amazon.AWSSupport.AmazonAWSSupportException ex)
        {
            if (ex.ErrorCode == "SubscriptionRequiredException")
            {
                return false;
            }
            else throw;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Ações

`AddAttachmentsToSet`

O código de exemplo a seguir mostra como usar `AddAttachmentsToSet`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does not
exist.
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates a
new attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string fileName,
string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
                    }
                }
            });
        return response.AttachmentSetId;
    }
```

- Para obter detalhes da API, consulte AddAttachmentsToSet em Referência AWS SDK para .NET da API.

`AddCommunicationToCase`

O código de exemplo a seguir mostra como usar `AddCommunicationToCase`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add communication to a case, including optional attachment set ID and CC
email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }
```

- Para obter detalhes da API, consulte AddCommunicationToCase em Referência AWS SDK para .NET da API.

`CreateCase`

O código de exemplo a seguir mostra como usar `CreateCase`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Create a new support case.
   /// </summary>
   /// <param name="serviceCode">Service code for the new case.</param>
   /// <param name="categoryCode">Category for the new case.</param>
   /// <param name="severityCode">Severity code for the new case.</param>
   /// <param name="subject">Subject of the new case.</param>
   /// <param name="body">Body text of the new case.</param>
   /// <param name="language">Optional language support for your case.
   /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
   /// <param name="attachmentSetId">Optional Id for an attachment set for the new
case.</param>
   /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
   /// <returns>The caseId of the new support case.</returns>
   public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
       string body, string language = "en", string? attachmentSetId = null, string
issueType = "customer-service")
   {
       var response = await _amazonSupport.CreateCaseAsync(
           new CreateCaseRequest()
           {
               ServiceCode = serviceCode,
               CategoryCode = categoryCode,
               SeverityCode = severityCode,
               Subject = subject,
               Language = language,
               AttachmentSetId = attachmentSetId,
               IssueType = issueType,
               CommunicationBody = body
           });
```

```
        return response.CaseId;
    }
```

- Para obter detalhes da API, consulte CreateCase em Referência AWS SDK para .NET da API.

`DescribeAttachment`

O código de exemplo a seguir mostra como usar `DescribeAttachment`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }
```

- Para obter detalhes da API, consulte DescribeAttachment em Referência AWS SDK para .NET da API.

`DescribeCases`

O código de exemplo a seguir mostra como usar `DescribeCases`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
    /// <param name="displayId">Optional display ID.</param>
    /// <param name="includeCommunication">True to include communication. Defaults
 to true.</param>
    /// <param name="includeResolvedCases">True to include resolved cases. Defaults
 to false.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
 are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds, string?
 displayId = null, bool includeCommunication = true,
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
 beforeTime = null,
        string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginators.DescribeCases(
            new DescribeCasesRequest()
            {
                CaseIdList = caseIds,
                DisplayId = displayId,
                IncludeCommunications = includeCommunication,
```

```
                IncludeResolvedCases = includeResolvedCases,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s"),
                Language = language
            });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeCases em Referência AWS SDK para .NET da API.

`DescribeCommunications`

O código de exemplo a seguir mostra como usar `DescribeCommunications`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <returns>The list of communications for the case.</returns>
```

```
    public async Task<List<Communication>> DescribeCommunications(string caseId,
DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
_amazonSupport.Paginators.DescribeCommunications(
            new DescribeCommunicationsRequest()
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeCommunications em Referência AWS SDK para .NET da API.

`DescribeServices`

O código de exemplo a seguir mostra como usar `DescribeServices`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
```

```
    /// <param name="name">Optional language for services.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
            {
                Language = language
            });
        return response.Services;
    }
```

- Para obter detalhes da API, consulte DescribeServices em Referência AWS SDK para .NET da API.

`DescribeSeverityLevels`

O código de exemplo a seguir mostra como usar `DescribeSeverityLevels`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
    /// <returns>The list of support severity levels.</returns>
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language =
"en")
```

```
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
            {
                Language = language
            });
        return response.SeverityLevels;
    }
```

- Para obter detalhes da API, consulte DescribeSeverityLevels em Referência AWS SDK para .NET da API.

`ResolveCase`

O código de exemplo a seguir mostra como usar `ResolveCase`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Resolve a support case by caseId.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <returns>The final status of the case after resolving.</returns>
    public async Task<string> ResolveCase(string caseId)
    {
        var response = await _amazonSupport.ResolveCaseAsync(
            new ResolveCaseRequest()
            {
                CaseId = caseId
            });
        return response.FinalCaseStatus;
    }
```

- Para obter detalhes da API, consulte ResolveCase em Referência AWS SDK para .NET da API.

Exemplos do Amazon Textract usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon Textract.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.

- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon Transcribe usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon Transcribe.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateVocabulary`

O código de exemplo a seguir mostra como usar `CreateVocabulary`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a custom vocabulary using a list of phrases. Custom vocabularies
    /// improve transcription accuracy for one or more specific words.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> CreateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.CreateVocabularyAsync(
            new CreateVocabularyRequest
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- Para obter detalhes da API, consulte CreateVocabulary em Referência AWS SDK para .NET da API.

`DeleteMedicalTranscriptionJob`

O código de exemplo a seguir mostra como usar `DeleteMedicalTranscriptionJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a medical transcription job. Also deletes the transcript associated
 with the job.
    /// </summary>
    /// <param name="jobName">Name of the medical transcription job to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMedicalTranscriptionJob(string jobName)
    {
        var response = await
 _amazonTranscribeService.DeleteMedicalTranscriptionJobAsync(
            new DeleteMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteMedicalTranscriptionJob em Referência AWS SDK para .NET da API.

`DeleteTranscriptionJob`

O código de exemplo a seguir mostra como usar `DeleteTranscriptionJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a transcription job. Also deletes the transcript associated with the
job.
    /// </summary>
    /// <param name="jobName">Name of the transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.DeleteTranscriptionJobAsync(
            new DeleteTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteTranscriptionJob em Referência AWS SDK para .NET da API.

`DeleteVocabulary`

O código de exemplo a seguir mostra como usar `DeleteVocabulary`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an existing custom vocabulary.
    /// </summary>
    /// <param name="vocabularyName">Name of the vocabulary to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteCustomVocabulary(string vocabularyName)
    {
        var response = await _amazonTranscribeService.DeleteVocabularyAsync(
            new DeleteVocabularyRequest
            {
                VocabularyName = vocabularyName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteVocabulary em Referência AWS SDK para .NET da API.

`GetTranscriptionJob`

O código de exemplo a seguir mostra como usar `GetTranscriptionJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get details about a transcription job.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <returns>A TranscriptionJob instance with information on the requested
job.</returns>
    public async Task<TranscriptionJob> GetTranscriptionJob(string jobName)
```

```
    {
        var response = await _amazonTranscribeService.GetTranscriptionJobAsync(
            new GetTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.TranscriptionJob;
    }
```

- Para obter detalhes da API, consulte GetTranscriptionJob em Referência AWS SDK para .NET da API.

`GetVocabulary`

O código de exemplo a seguir mostra como usar `GetVocabulary`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information about a custom vocabulary.
    /// </summary>
    /// <param name="vocabularyName">Name of the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> GetCustomVocabulary(string vocabularyName)
    {
        var response = await _amazonTranscribeService.GetVocabularyAsync(
            new GetVocabularyRequest()
            {
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- Para obter detalhes da API, consulte GetVocabulary em Referência AWS SDK para .NET da API.

`ListMedicalTranscriptionJobs`

O código de exemplo a seguir mostra como usar `ListMedicalTranscriptionJobs`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// List medical transcription jobs, optionally with a name filter.
   /// </summary>
   /// <param name="jobNameContains">Optional name filter for the medical
transcription jobs.</param>
   /// <returns>A list of summaries about medical transcription jobs.</returns>
   public async Task<List<MedicalTranscriptionJobSummary>>
ListMedicalTranscriptionJobs(
       string? jobNameContains = null)
   {
       var response = await
_amazonTranscribeService.ListMedicalTranscriptionJobsAsync(
           new ListMedicalTranscriptionJobsRequest()
           {
               JobNameContains = jobNameContains
           });
       return response.MedicalTranscriptionJobSummaries;
   }
```

- Para obter detalhes da API, consulte ListMedicalTranscriptionJobs em Referência AWS SDK para .NET da API.

`ListTranscriptionJobs`

O código de exemplo a seguir mostra como usar `ListTranscriptionJobs`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// List transcription jobs, optionally with a name filter.
   /// </summary>
   /// <param name="jobNameContains">Optional name filter for the transcription
jobs.</param>
   /// <returns>A list of transcription job summaries.</returns>
   public async Task<List<TranscriptionJobSummary>> ListTranscriptionJobs(string?
jobNameContains = null)
   {
       var response = await _amazonTranscribeService.ListTranscriptionJobsAsync(
           new ListTranscriptionJobsRequest()
           {
               JobNameContains = jobNameContains
           });
       return response.TranscriptionJobSummaries;
   }
```

- Para obter detalhes da API, consulte ListTranscriptionJobs em Referência AWS SDK para .NET da API.

`ListVocabularies`

O código de exemplo a seguir mostra como usar `ListVocabularies`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List custom vocabularies for the current account. Optionally specify a name
    /// filter and a specific state to filter the vocabularies list.
    /// </summary>
    /// <param name="nameContains">Optional string the vocabulary name must
contain.</param>
    /// <param name="stateEquals">Optional state of the vocabulary.</param>
    /// <returns>List of information about the vocabularies.</returns>
    public async Task<List<VocabularyInfo>> ListCustomVocabularies(string?
nameContains = null,
        VocabularyState? stateEquals = null)
    {
        var response = await _amazonTranscribeService.ListVocabulariesAsync(
            new ListVocabulariesRequest()
            {
                NameContains = nameContains,
                StateEquals = stateEquals
            });
        return response.Vocabularies;
    }
```

- Para obter detalhes da API, consulte ListVocabularies em Referência AWS SDK para .NET da API.

StartMedicalTranscriptionJob

O código de exemplo a seguir mostra como usar `StartMedicalTranscriptionJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start a medical transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the medical transcription job.</
param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon S3
 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="outputBucketName">Location for the output, typically an Amazon
 S3 location.</param>
    /// <param name="transcriptionType">Conversation or dictation transcription
 type.</param>
    /// <returns>A MedicalTransactionJob instance with information on the new job.</
returns>
    public async Task<MedicalTranscriptionJob> StartMedicalTranscriptionJob(
        string jobName, string mediaFileUri,
        MediaFormat mediaFormat, string outputBucketName,
 Amazon.TranscribeService.Type transcriptionType)
    {
        var response = await
 _amazonTranscribeService.StartMedicalTranscriptionJobAsync(
            new StartMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
```

```
                LanguageCode =
                    LanguageCode
                        .EnUS, // The value must be en-US for medical
transcriptions.
                OutputBucketName = outputBucketName,
                OutputKey =
                    jobName, // The value is a key used to fetch the output of the
transcription.
                Specialty = Specialty.PRIMARYCARE, // The value PRIMARYCARE must be
set.
                Type = transcriptionType
            });
        return response.MedicalTranscriptionJob;
    }
```

- Para obter detalhes da API, consulte StartMedicalTranscriptionJob em Referência AWS SDK para .NET da API.

StartTranscriptionJob

O código de exemplo a seguir mostra como usar `StartTranscriptionJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Start a transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon S3
location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
```

```
    /// <param name="languageCode">The language code of the media file, such as en-
US.</param>
    /// <param name="vocabularyName">Optional name of a custom vocabulary.</param>
    /// <returns>A TranscriptionJob instance with information on the new job.</
returns>
    public async Task<TranscriptionJob> StartTranscriptionJob(string jobName, string
 mediaFileUri,
        MediaFormat mediaFormat, LanguageCode languageCode, string? vocabularyName)
    {
        var response = await _amazonTranscribeService.StartTranscriptionJobAsync(
            new StartTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode = languageCode,
                Settings = vocabularyName != null ? new Settings()
                {
                    VocabularyName = vocabularyName
                } : null
            });
        return response.TranscriptionJob;
    }
```

- Para obter detalhes da API, consulte StartTranscriptionJob em Referência AWS SDK para .NET da API.

`UpdateVocabulary`

O código de exemplo a seguir mostra como usar `UpdateVocabulary`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update a custom vocabulary with new values. Update overwrites all existing
information.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> UpdateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.UpdateVocabularyAsync(
            new UpdateVocabularyRequest()
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- Para obter detalhes da API, consulte UpdateVocabulary em Referência AWS SDK para .NET da API.

Exemplos do Amazon Translate usando SDK para .NET

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET com o Amazon Translate.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DescribeTextTranslationJob`

O código de exemplo a seguir mostra como usar `DescribeTextTranslationJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// The following example shows how to retrieve the details of
/// a text translation job using Amazon Translate.
/// </summary>
public class DescribeTextTranslation
{
    public static async Task Main()
```

```
        {
            var client = new AmazonTranslateClient();

            // The Job Id is generated when the text translation job is started
            // with a call to the StartTextTranslationJob method.
            var jobId = "1234567890abcdef01234567890abcde";

            var request = new DescribeTextTranslationJobRequest
            {
                JobId = jobId,
            };

            var jobProperties = await DescribeTranslationJobAsync(client, request);

            DisplayTranslationJobDetails(jobProperties);
        }

        /// <summary>
        /// Retrieve information about an Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client object.</
param>
        /// <param name="request">The DescribeTextTranslationJobRequest object.</
param>
        /// <returns>The TextTranslationJobProperties object containing
        /// information about the text translation job..</returns>
        public static async Task<TextTranslationJobProperties>
 DescribeTranslationJobAsync(
            AmazonTranslateClient client,
            DescribeTextTranslationJobRequest request)
        {
            var response = await client.DescribeTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                return response.TextTranslationJobProperties;
            }
            else
            {
                return null;
            }
        }

        /// <summary>
        /// Displays the properties of the text translation job.
```

```
        /// </summary>
        /// <param name="jobProperties">The properties of the text translation
        /// job returned by the call to DescribeTextTranslationJobAsync.</param>
        public static void DisplayTranslationJobDetails(TextTranslationJobProperties
jobProperties)
        {
            if (jobProperties is null)
            {
                Console.WriteLine("No text translation job properties found.");
                return;
            }

            // Display the details of the text translation job.
            Console.WriteLine($"{jobProperties.JobId}: {jobProperties.JobName}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeTextTranslationJob em Referência AWS SDK para .NET da API.

`ListTextTranslationJobs`

O código de exemplo a seguir mostra como usar `ListTextTranslationJobs`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
```

```
    /// List Amazon Translate translation jobs, along with details about each job.
    /// </summary>
    public class ListTranslationJobs
    {
        public static async Task Main()
        {
            var client = new AmazonTranslateClient();
            var filter = new TextTranslationJobFilter
            {
                JobStatus = "COMPLETED",
            };

            var request = new ListTextTranslationJobsRequest
            {
                MaxResults = 10,
                Filter = filter,
            };

            await ListJobsAsync(client, request);
        }

        /// <summary>
        /// List Amazon Translate text translation jobs.
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client object.</
param>
        /// <param name="request">An Amazon Translate
        /// ListTextTranslationJobsRequest object detailing which text
        /// translation jobs are of interest.</param>
        public static async Task ListJobsAsync(
            AmazonTranslateClient client,
            ListTextTranslationJobsRequest request)
        {
            ListTextTranslationJobsResponse response;

            do
            {
                response = await client.ListTextTranslationJobsAsync(request);

 ShowTranslationJobDetails(response.TextTranslationJobPropertiesList);

                request.NextToken = response.NextToken;
            }
            while (response.NextToken is not null);
```

```
        }

        /// <summary>
        /// List existing translation job details.
        /// </summary>
        /// <param name="properties">A list of Amazon Translate text
        /// translation jobs.</param>
        public static void
ShowTranslationJobDetails(List<TextTranslationJobProperties> properties)
        {
            properties.ForEach(prop =>
            {
                Console.WriteLine($"{prop.JobId}: {prop.JobName}");
                Console.WriteLine($"Status: {prop.JobStatus}");
                Console.WriteLine($"Submitted time: {prop.SubmittedTime}");
            });
        }
    }
```

- Para obter detalhes da API, consulte ListTextTranslationJobs em Referência AWS SDK para .NET da API.

StartTextTranslationJob

O código de exemplo a seguir mostra como usar `StartTextTranslationJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Translate;
    using Amazon.Translate.Model;
```

```
    /// <summary>
    /// This example shows how to use Amazon Translate to process the files in
    /// an Amazon Simple Storage Service (Amazon S3) bucket. The translated results
    /// will also be stored in an Amazon S3 bucket.
    /// </summary>
    public class BatchTranslate
    {
        public static async Task Main()
        {
            var contentType = "text/plain";

            // Set this variable to an S3 bucket location with a folder."
            // Input files must be in a folder and not at the bucket root."
            var s3InputUri = "s3://amzn-s3-demo-bucket1/FOLDER/";
            var s3OutputUri = "s3://amzn-s3-demo-bucket2/";

            // This role must have permissions to read the source bucket and to read
 and
            // write to the destination bucket where the translated text will be
 stored.
            var dataAccessRoleArn = "arn:aws:iam::0123456789ab:role/
S3TranslateRole";

            var client = new AmazonTranslateClient();

            var inputConfig = new InputDataConfig
            {
                ContentType = contentType,
                S3Uri = s3InputUri,
            };

            var outputConfig = new OutputDataConfig
            {
                S3Uri = s3OutputUri,
            };

            var request = new StartTextTranslationJobRequest
            {
                JobName = "ExampleTranslationJob",
                DataAccessRoleArn = dataAccessRoleArn,
                InputDataConfig = inputConfig,
                OutputDataConfig = outputConfig,
                SourceLanguageCode = "en",
```

```
                TargetLanguageCodes = new List<string> { "fr" },
            };

            var response = await StartTextTranslationAsync(client, request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId}: {response.JobStatus}");
            }
        }

        /// <summary>
        /// Start the Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized AmazonTranslateClient object.</
param>
        /// <param name="request">The request object that includes details such
        /// as source and destination bucket names and the IAM Role that will
        /// be used to access the buckets.</param>
        /// <returns>The StartTextTranslationResponse object that includes the
        /// details of the request response.</returns>
        public static async Task<StartTextTranslationJobResponse>
 StartTextTranslationAsync(AmazonTranslateClient client,
 StartTextTranslationJobRequest request)
        {
            var response = await client.StartTextTranslationJobAsync(request);
            return response;
        }
    }
```

- Para obter detalhes da API, consulte StartTextTranslationJob em Referência AWS SDK para .NET da API.

`StopTextTranslationJob`

O código de exemplo a seguir mostra como usar `StopTextTranslationJob`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// Shows how to stop a running Amazon Translation Service text translation
/// job.
/// </summary>
public class StopTextTranslationJob
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new StopTextTranslationJobRequest
        {
            JobId = jobId,
        };

        await StopTranslationJobAsync(client, request);
    }

    /// <summary>
    /// Sends a request to stop a text translation job.
    /// </summary>
    /// <param name="client">Initialized AmazonTrnslateClient object.</param>
    /// <param name="request">The request object to be passed to the
    /// StopTextJobAsync method.</param>
    public static async Task StopTranslationJobAsync(
        AmazonTranslateClient client,
        StopTextTranslationJobRequest request)
    {
```

```
            var response = await client.StopTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId} as status:
{response.JobStatus}");
            }
        }
    }
```

- Para obter detalhes da API, consulte StopTextTranslationJob em Referência AWS SDK para .NET da API.

`TranslateText`

O código de exemplo a seguir mostra como usar `TranslateText`.

SDK para .NET

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Transfer;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
    /// Take text from a file stored a Amazon Simple Storage Service (Amazon S3)
    /// object and translate it using the Amazon Transfer Service.
    /// </summary>
    public class TranslateText
    {
        public static async Task Main()
```

```
        {
            // If the region you want to use is different from the region
            // defined for the default user, supply it as a parameter to the
            // Amazon Translate client object constructor.
            var client = new AmazonTranslateClient();

            // Set the source language to "auto" to request Amazon Translate to
            // automatically detect te language of the source text.

            // You can get a list of the languages supposed by Amazon Translate
            // in the Amazon Translate Developer's Guide here:
            //      https://docs.aws.amazon.com/translate/latest/dg/what-is.html
            string srcLang = "en"; // English.
            string destLang = "fr"; // French.

            // The Amazon Simple Storage Service (Amazon S3) bucket where the
            // source text file is stored.
            string srcBucket = "amzn-s3-demo-bucket";
            string srcTextFile = "source.txt";

            var srcText = await GetSourceTextAsync(srcBucket, srcTextFile);
            var destText = await TranslatingTextAsync(client, srcLang, destLang,
srcText);

            ShowText(srcText, destText);
        }

        /// <summary>
        /// Use the Amazon S3 TransferUtility to retrieve the text to translate
        /// from an object in an S3 bucket.
        /// </summary>
        /// <param name="srcBucket">The name of the S3 bucket where the
        /// text is stored.
        /// </param>
        /// <param name="srcTextFile">The key of the S3 object that
        /// contains the text to translate.</param>
        /// <returns>A string representing the source text.</returns>
        public static async Task<string> GetSourceTextAsync(string srcBucket, string
srcTextFile)
        {
            string srcText = string.Empty;

            var s3Client = new AmazonS3Client();
            TransferUtility utility = new TransferUtility(s3Client);
```

```
            using var stream = await utility.OpenStreamAsync(srcBucket,
srcTextFile);

            StreamReader file = new System.IO.StreamReader(stream);

            srcText = file.ReadToEnd();
            return srcText;
        }

        /// <summary>
        /// Use the Amazon Translate Service to translate the document from the
        /// source language to the specified destination language.
        /// </summary>
        /// <param name="client">The Amazon Translate Service client used to
        /// perform the translation.</param>
        /// <param name="srcLang">The language of the source text.</param>
        /// <param name="destLang">The destination language for the translated
        /// text.</param>
        /// <param name="text">A string representing the text to ranslate.</param>
        /// <returns>The text that has been translated to the destination
        /// language.</returns>
        public static async Task<string> TranslatingTextAsync(AmazonTranslateClient
client, string srcLang, string destLang, string text)
        {
            var request = new TranslateTextRequest
            {
                SourceLanguageCode = srcLang,
                TargetLanguageCode = destLang,
                Text = text,
            };

            var response = await client.TranslateTextAsync(request);

            return response.TranslatedText;
        }

        /// <summary>
        /// Show the original text followed by the translated text.
        /// </summary>
        /// <param name="srcText">The original text to be translated.</param>
        /// <param name="destText">The translated text.</param>
        public static void ShowText(string srcText, string destText)
        {
```

```
            Console.WriteLine("Source text:");
            Console.WriteLine(srcText);
            Console.WriteLine();
            Console.WriteLine("Translated text:");
            Console.WriteLine(destText);
        }
    }
```

- Para obter detalhes da API, consulte TranslateText em Referência AWS SDK para .NET da API.

Cenários

Criação de uma aplicação do Amazon SNS

O exemplo de código a seguir mostra como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

SDK para .NET

Mostra como usar a API .NET do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para .NET

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos de código para SDK para .NET (v4)

Os exemplos de código a seguir mostram como usar o AWS SDK para .NET (v4) com AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- SDK para .NET (v4) Guia do desenvolvedor — Saiba mais sobre como usar o.NET com o. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos de Aurora usando SDK para .NET (v4)
- Exemplos de Auto Scaling usando SDK para .NET (v4)
- Exemplos do Amazon Bedrock usando SDK para .NET (v4)
- Exemplos do Amazon Bedrock Runtime usando SDK para .NET (v4)
- CloudFormation exemplos usando SDK para .NET (v4)
- CloudWatch exemplos usando SDK para .NET (v4)
- CloudWatch Exemplos de registros usando SDK para .NET (v4)
- Exemplos do Amazon Cognito Identity Provider usando SDK para .NET (v4)
- AWS Control Tower exemplos usando SDK para .NET (v4)
- Exemplos do DynamoDB usando SDK para .NET (v4)
- Exemplos do Amazon EC2 usando SDK para .NET (v4)
- Exemplos do Amazon ECS usando SDK para .NET (v4)
- AWS IoT exemplos usando SDK para .NET (v4)
- AWS IoT data exemplos usando SDK para .NET (v4)
- Exemplos do Amazon Redshift usando SDK para .NET (v4)
- Exemplos do Amazon S3 usando SDK para .NET (v4)

Exemplos de Aurora usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o Aurora.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Aurora

O exemplo de código a seguir mostra como começar a usar o Aurora.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
```

```
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here for
 example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para .NET .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de cluster do banco de dados do Aurora e definir os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Crie um snapshot do cluster do banco de dados e limpe os recursos.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using the
 DescribeDBEngineVersionsAsync method.
```

```
   2.  Select an engine family and create a custom DB cluster parameter group using
the CreateDBClusterParameterGroupAsync method.
   3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
method.
   4.  Get some parameters in the group using the DescribeDBClusterParametersAsync
method.
   5.  Parse and display some parameters in the group.
   6.  Modify the auto_increment_offset and auto_increment_increment parameters
       using the ModifyDBClusterParameterGroupAsync method.
   7.  Get and display the updated parameters using the
DescribeDBClusterParametersAsync method with a source of "user".
   8.  Get a list of allowed engine versions using the
DescribeDBEngineVersionsAsync method.
   9.  Create an Aurora DB cluster that contains a MySql database and uses the
parameter group.
       using the CreateDBClusterAsync method.
   10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
method.
   11. Display and select from a list of instance classes available for the
selected engine and version
       using the paginated DescribeOrderableDBInstanceOptions method.
   12. Create a database instance in the cluster using the CreateDBInstanceAsync
method.
   13. Wait for the DB instance to be ready using the DescribeDBInstances method.
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the CreateDBClusterSnapshotAsync
method.
   16. Wait for DB snapshot to be ready using the DescribeDBClusterSnapshotsAsync
method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
                    .AddTransient<AuroraWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger<AuroraScenario>();

        auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();

        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Aurora: get started with DB clusters example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName,
parameters);
```

```
            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            newInstance = await CreateNewInstance(
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
```

```
    /// Choose the Aurora DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamilyAsync()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

        Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

        var parameterGroupFamilies =
            engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
        for (var i = 1; i <= parameterGroupFamilies.Count; i++)
        {
            var parameterGroupFamily = parameterGroupFamilies[i - 1];
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("2. Select an available DB parameter group family by
entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the new
DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
```

```
    public static async Task<DBClusterParameterGroup>
 CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
 {dbParameterGroupFamily}:");

        var parameterGroup = await
 auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
 auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParameterG

        Console.WriteLine(
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</param>
    /// <param name="parameterNames">Optional specific names of parameters to
 describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>> DescribeParametersInGroupAsync(string
 parameterGroupName, List<string>? parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);

        var matchingParameters =
```

```
            parameters.Where(p => parameterNames == null ||
parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
group.");
```

```
        var parameters =
            await
auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName, "user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group family
{dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
```

```
            Console.WriteLine("8. Select an available DB engine version by entering
 a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
 cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
 {clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
 clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
 clusterIdentifier);
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
```

```
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
            );

            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
 auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }

        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption> ChooseDBInstanceClass(string
 engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
```

```
            await auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("11. Select an available DB instance class by entering
 a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateNewInstance(
        string clusterIdentifier,
        string engineName,
        string engineVersion,
```

```
        string instanceClass,
        string instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"12. Create a new DB instance with identifier
{instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
        }
        else
        {
            newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
                clusterIdentifier,
                instanceIdentifier,
                engineName,
                engineVersion,
                instanceClass
            );

            Console.WriteLine("13. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                instances = await
auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
                newInstance = instances.First();
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
```

```
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
 {cluster.DBClusterIdentifier}.");
        var snapshot = await auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
```

```
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Clean up resources from the scenario.
    /// </summary>
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
{newCluster.DBClusterIdentifier}.");
```

```
            await
auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
newCluster.DBClusterIdentifier);
            }

            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
{parameterGroup.DBClusterParameterGroupName}.");
            await
auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGroup
            Console.WriteLine("Parameter group deleted.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
```

```
        return response;
    }
```

Métodos de encapsulamento que são chamados pelo cenário para gerenciar as ações do Aurora.

```
using Amazon.RDS;
using Amazon.RDS.Model;

namespace AuroraActions;

/// <summary>
/// Wrapper for the Amazon Aurora cluster client operations.
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family name.</
param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }

    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
 DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
```

```
        do
        {
            response = await _amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }

    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }

    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string groupName,
List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
```

```
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the allowed
values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await _amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }


    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
```

```
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }

    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await _amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
```

```
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
 instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
 dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB cluster.</
param>
    /// <returns>List of DB clusters.</returns>
```

```
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name or
size.
```

```
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
```

```
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
```

```
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create a new cluster and database.
/// </summary>
/// <param name="dbName">The name of the new database.</param>
/// <param name="clusterIdentifier">The identifier of the cluster.</param>
/// <param name="parameterGroupName">The name of the parameter group.</param>
/// <param name="dbEngine">The engine to use for the new cluster.</param>
/// <param name="dbEngineVersion">The version of the engine to use.</param>
/// <param name="adminName">The admin username.</param>
/// <param name="adminPassword">The primary admin password.</param>
/// <returns>The cluster object.</returns>
public async Task<DBCluster> CreateDBClusterWithAdminAsync(
    string dbName,
    string clusterIdentifier,
    string parameterGroupName,
    string dbEngine,
    string dbEngineVersion,
    string adminName,
    string adminPassword)
{
    var request = new CreateDBClusterRequest
    {
        DatabaseName = dbName,
        DBClusterIdentifier = clusterIdentifier,
        DBClusterParameterGroupName = parameterGroupName,
        Engine = dbEngine,
        EngineVersion = dbEngineVersion,
        MasterUsername = adminName,
        MasterUserPassword = adminPassword,
    };

    var response = await _amazonRDS.CreateDBClusterAsync(request);
    return response.DBCluster;
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para .NET .

`CreateDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- Para obter detalhes da API, consulte CreateDBClusterParameterGroup em Referência AWS SDK para .NET da API.

`CreateDBClusterSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBClusterSnapshot`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }
```

- Para obter detalhes da API, consulte CreateDBClusterSnapshot em Referência AWS SDK para .NET da API.

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name or
size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
```

```
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para .NET .

DeleteDBCluster

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para .NET .

DeleteDBClusterParameterGroup

O código de exemplo a seguir mostra como usar `DeleteDBClusterParameterGroup`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete a particular parameter group by name.
/// </summary>
/// <param name="groupName">The name of the parameter group.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteClusterParameterGroupByNameAsync(string groupName)
{
    var request = new DeleteDBClusterParameterGroupRequest
    {
        DBClusterParameterGroupName = groupName,
    };

    var response = await _amazonRDS.DeleteDBClusterParameterGroupAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DeleteDBClusterParameterGroup em Referência AWS SDK para .NET da API.

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para .NET .

`DescribeDBClusterParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameterGroups`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
DescribeCustomDBClusterParameterGroupAsync(string name)
```

```
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterParameterGroups em Referência AWS SDK para .NET da API.

`DescribeDBClusterParameters`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameters`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
```

```
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterParameters em Referência AWS SDK para .NET da API.

`DescribeDBClusterSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBClusterSnapshots`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();
```

```
        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
 DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterSnapshots em Referência AWS SDK para .NET da API.

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB cluster.</
param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
 dbClusterIdentifier = null)
```

```
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para .NET .

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
```

```
    /// <param name="parameterGroupFamily">Optional parameter group family name.</
param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions em Referência AWS SDK para .NET da API.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
 instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
 dbInstanceIdentifier = null)
    {
```

```
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para .NET .

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
```

```
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptions em Referência AWS SDK para .NET da API.

`ModifyDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
```

```
    public async Task<string> ModifyIntegerParametersInGroupAsync(string groupName,
List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the allowed
values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await _amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- Para obter detalhes da API, consulte ModifyDBClusterParameterGroup em Referência AWS SDK para .NET da API.

Exemplos de Auto Scaling usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com Auto Scaling.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Auto Scaling

O exemplo de código a seguir mostra como começar a usar o Auto Scaling.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
```

```
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        if (response.AutoScalingGroups == null || response.AutoScalingGroups.Count
 == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
            return;
        }
        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availability
        });

    }
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo do Amazon EC2 Auto Scaling com um modelo de inicialização e zonas de disponibilidade e obter informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.

- Obtenha estatísticas para CloudWatch métricas e, em seguida, limpe os recursos.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.EC2;
using Microsoft.Extensions.Configuration;
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
```

```
            services.AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonCloudWatch>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalingWrapper>()
                .AddTransient<CloudWatchWrapper>()
                .AddTransient<EC2Wrapper>()
                .AddTransient<UIWrapper>()
            )
            .Build();


        var autoScalingWrapper =
host.Services.GetRequiredService<AutoScalingWrapper>();
        var cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();
        var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var imageId = configuration["ImageId"];
        var instanceType = configuration["InstanceType"];
        var launchTemplateName = configuration["LaunchTemplateName"];

        launchTemplateName += Guid.NewGuid().ToString();

        // The name of the Auto Scaling group.
        var groupName = configuration["GroupName"];

        uiWrapper.DisplayTitle("Auto Scaling Basics");
        uiWrapper.DisplayAutoScalingBasicsDescription();

        // Create the launch template and save the template Id to use when deleting
the
        // launch template at the end of the application.
        var launchTemplateId = await ec2Wrapper.CreateLaunchTemplateAsync(imageId!,
instanceType!, launchTemplateName);

        // Confirm that the template was created by asking for a description of it.
```

```
        await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

        uiWrapper.PressEnter();

        var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

        Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
        await autoScalingWrapper.CreateAutoScalingGroupAsync(
            groupName!,
            launchTemplateName,
            availabilityZones[0].ZoneName);

        // Keep checking the details of the new group until its lifecycle state
        // is "InService".
        Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

        List<AutoScalingInstanceDetails> instanceDetails;

        do
        {
            instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
        }
        while (instanceDetails.Count <= 0);

        Console.WriteLine($"Auto scaling group {groupName} successfully created.");
        Console.WriteLine($"{instanceDetails.Count} instances were created for the
group.");

        // Display the details of the Auto Scaling group.
        instanceDetails.ForEach(detail =>
        {
            Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Metrics collection");
        Console.WriteLine($"Enable metrics collection for {groupName}");
        await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);

        // Show the metrics that are collected for the group.

        // Update the maximum size of the group to three instances.
```

```
        Console.WriteLine("--- Update the Auto Scaling group to increase max size to
3 ---");
        int maxSize = 3;
        await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

        Console.WriteLine("--- Describe all Auto Scaling groups to show the current
state of the group ---");
        var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

        uiWrapper.DisplayGroupDetails(groups!);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Describe account limits");
        await autoScalingWrapper.DescribeAccountLimitsAsync();

        uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");

        uiWrapper.DisplayTitle("Set desired capacity");
        int desiredCapacity = 2;
        await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

        Console.WriteLine("Get the two instance Id values");

        // Empty the group before getting the details again.
        groups.Clear();
        groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
        if (groups.Any())
        {
            foreach (AutoScalingGroup group in groups)
            {
                Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
                Console.WriteLine($"The group ARN is {group.AutoScalingGroupARN}");
                var instances = group.Instances;
                foreach (Amazon.AutoScaling.Model.Instance instance in instances)
                {
                    Console.WriteLine($"The instance id is {instance.InstanceId}");
                    Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
```

```
                }
            }
        }

        uiWrapper.DisplayTitle("Scaling Activities");
        Console.WriteLine("Let's list the scaling activities that have occurred for
the group.");
        var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
        if (activities.Any())
        {
            activities.ForEach(activity =>
            {
                Console.WriteLine($"The activity Id is {activity.ActivityId}");
                Console.WriteLine($"The activity details are {activity.Details}");
            });
        }

        // Display the Amazon CloudWatch metrics that have been collected.
        var metrics = await cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
        if (metrics.Any())
        {
            Console.WriteLine($"Metrics collected for {groupName}:");
            metrics.ForEach(metric =>
            {
                Console.Write($"Metric name: {metric.MetricName}\t");
                Console.WriteLine($"Namespace: {metric.Namespace}");
            });
        }

        var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
        if (dataPoints.Any())
        {
            Console.WriteLine("Details for the metrics collected:");
            dataPoints.ForEach(detail => { Console.WriteLine(detail); });
        }

        // Disable metrics collection.
        Console.WriteLine("Disabling the collection of metrics for {groupName}.");
        var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

        if (success)
```

```
        {
            Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
        }

        // Terminate all instances in the group.
        uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
        Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

        if (groups is not null)
        {
            groups.ForEach(group =>
            {
                // Only delete instances in the AutoScaling group we created.
                if (group.AutoScalingGroupName == groupName)
                {
                    group.Instances.ForEach(async instance =>
                    {
                        await
autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                    });
                }
            });
        }

        // After all instances are terminated, delete the group.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the Auto Scaling group.");
        await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);

        // Delete the launch template.
        var deletedLaunchTemplateName = await
ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }
```

```
        Console.WriteLine("The demo is now concluded.");
    }
}


namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
    {
        Console.WriteLine("This code example performs the following operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
        Console.WriteLine(" 7. Changes the desired capacity of the Auto Scaling");
        Console.WriteLine("    group to use an additional instance.");
        Console.WriteLine(" 8. Shows that there are now instances in the group.");
        Console.WriteLine(" 9. Lists the scaling activities that have occurred for
 the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
 group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
```

```
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>
        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }
```

```
    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

Defina as funções que são chamadas pelo cenário para gerenciar modelos e métricas de lançamento. Essas funções abrangem o Auto Scaling, o Amazon EC2 e CloudWatch as ações.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
```

```
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
 client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)
    {
        _amazonAutoScaling = amazonAutoScaling;
    }


    /// <summary>
    /// Create a new Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name to use for the new Auto Scaling
    /// group.</param>
    /// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
    /// launch template to use to create instances in the group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        string availabilityZone)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
```

```
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };
        try
        {
            var response = await
 _amazonAutoScaling.CreateAutoScalingGroupAsync(request);
            Console.WriteLine($"{groupName} Auto Scaling Group created");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }



    /// <summary>
    /// Retrieve information about Amazon EC2 Auto Scaling quotas to the
    /// active AWS account.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DescribeAccountLimitsAsync()
    {
        var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
 response.MaxNumberOfAutoScalingGroups);
        Console.WriteLine("The current number of Auto Scaling groups is " +
 response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
```

```
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }



    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
 null)
                {
                    group.Instances.ForEach(instance =>
                    {
```

```
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }



    /// <summary>
    /// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
    public async Task<List<AutoScalingGroup>> DescribeAutoScalingGroupsAsync(
        string groupName)
    {
        var groups = new List<AutoScalingGroup>();
        var groupList = new List<string>
            {
                groupName,
            };

        var request = new DescribeAutoScalingGroupsRequest
        {
            AutoScalingGroupNames = groupList,
        };
```

```
        var response = await
 _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
        if (response.AutoScalingGroups != null)
        {
            groups = response.AutoScalingGroups;
        }

        return groups;
    }


    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
 _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }


    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
```

```
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
```

```
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
 _amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
 {desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }


    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
 {groupName}.");
            return true;
        }
```

```
        else
        {
            return false;
        }
    }

}


namespace AutoScalingActions;

using Amazon.EC2;
using Amazon.EC2.Model;

public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }

    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
```

```
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Delete an Amazon EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
    {
        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);
```

```
        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.Write($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
    }

    /// <summary>
    /// Retrieve the availability zones for the current region.
    /// </summary>
    /// <returns>A collection of availability zones.</returns>
    public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
    {
        var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());

        return response.AvailabilityZones;
    }
}


namespace AutoScalingActions;

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
```

```
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</returns>
    public async Task<List<Metric>> GetCloudWatchMetricsAsync(string groupName)
    {
        var metrics = new List<Metric>();
        var filter = new DimensionFilter
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        };

        var request = new ListMetricsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = new List<DimensionFilter> { filter },
            Namespace = "AWS/AutoScaling",
        };

        var response = await _amazonCloudWatch.ListMetricsAsync(request);
        if (response.Metrics != null)
        {
            metrics = response.Metrics;
        }
        return metrics;
    }

    /// <summary>
    /// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of data points.</returns>
    public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
    {
        var dataPoints = new List<Datapoint>();
```

```
        var metricDimensions = new List<Dimension>
            {
                new Dimension
                {
                    Name = "AutoScalingGroupName",
                    Value = $"{groupName}",
                },
            };

        // The start time will be yesterday.
        var startTime = DateTime.UtcNow.AddDays(-1);

        var request = new GetMetricStatisticsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = metricDimensions,
            Namespace = "AWS/AutoScaling",
            Period = 60, // 60 seconds.
            Statistics = new List<string>() { "Minimum" },
            StartTimeUtc = startTime,
            EndTimeUtc = DateTime.UtcNow,
        };

        var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);
        if (response.Datapoints != null)
        {
            dataPoints = response.Datapoints;
        }

        return dataPoints;
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances

- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Ações

`CreateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `CreateAutoScalingGroup`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };
```

```
        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };
        try
        {
            var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
            Console.WriteLine($"{groupName} Auto Scaling Group created");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }
```

- Para obter detalhes da API, consulte CreateAutoScalingGroup em Referência AWS SDK para .NET da API.

`DescribeAutoScalingGroups`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingGroups`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
 null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };
```

```
            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK para .NET da API.

`DescribeAutoScalingInstances`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
```

```
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstances em Referência AWS SDK para .NET da API.

`DescribeScalingActivities`

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }
```

- Para obter detalhes da API, consulte DescribeScalingActivities em Referência AWS SDK para .NET da API.

`DisableMetricsCollection`

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DisableMetricsCollection em Referência AWS SDK para .NET da API.

`EnableMetricsCollection`

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte EnableMetricsCollection em Referência AWS SDK para .NET da API.

`SetDesiredCapacity`

O código de exemplo a seguir mostra como usar `SetDesiredCapacity`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte SetDesiredCapacity em Referência AWS SDK para .NET da API.

`TerminateInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroup em Referência AWS SDK para .NET da API.

`UpdateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `UpdateAutoScalingGroup`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
```

```
            Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroup em Referência AWS SDK para .NET da API.

Exemplos do Amazon Bedrock usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Bedrock

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon;
using Amazon.Bedrock;
using Amazon.Bedrock.Model;

namespace BedrockActions;

/// <summary>
/// This example shows how to list foundation models.
/// </summary>
internal class HelloBedrock
{
    /// <summary>
    /// Main method to call the ListFoundationModelsAsync method.
    /// </summary>
    /// <param name="args"> The command line arguments. </param>
    static async Task Main(string[] args)
    {
        // Specify a region endpoint where Amazon Bedrock is available. For a list
 of supported region see https://docs.aws.amazon.com/bedrock/latest/userguide/what-
is-bedrock.html#bedrock-regions
        AmazonBedrockClient bedrockClient = new(RegionEndpoint.USWest2);

        await ListFoundationModelsAsync(bedrockClient);

    }


    /// <summary>
    /// List foundation models.
    /// </summary>
    /// <param name="bedrockClient"> The Amazon Bedrock client. </param>
    private static async Task ListFoundationModelsAsync(AmazonBedrockClient
 bedrockClient)
    {
```

```
        Console.WriteLine("List foundation models with no filter.");

        try
        {
            var response = await bedrockClient.ListFoundationModelsAsync(new
 ListFoundationModelsRequest()
            {
            });

            if (response?.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                foreach (var fm in response.ModelSummaries)
                {
                    WriteToConsole(fm);
                }
            }
            else
            {
                Console.WriteLine("Something wrong happened");
            }
        }
        catch (AmazonBedrockException e)
        {
            Console.WriteLine(e.Message);
        }
    }


    /// <summary>
    /// Write the foundation model summary to console.
    /// </summary>
    /// <param name="foundationModel"> The foundation model summary to write to
 console. </param>
    private static void WriteToConsole(FoundationModelSummary foundationModel)
    {
        Console.WriteLine($"{foundationModel.ModelId}, Customization:
 {string.Join(", ", foundationModel.CustomizationsSupported)}, Stream:
 {foundationModel.ResponseStreamingSupported}, Input: {string.Join(",
 ", foundationModel.InputModalities)}, Output: {string.Join(", ",
 foundationModel.OutputModalities)}");
    }
}
```

- Para obter detalhes da API, consulte ListFoundationModels em Referência AWS SDK para .NET da API.

Ações

`ListFoundationModels`

O código de exemplo a seguir mostra como usar `ListFoundationModels`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Bedrock disponíveis.

```
   /// <summary>
   /// List foundation models.
   /// </summary>
   /// <param name="bedrockClient"> The Amazon Bedrock client. </param>
   private static async Task ListFoundationModelsAsync(AmazonBedrockClient
bedrockClient)
   {
       Console.WriteLine("List foundation models with no filter.");

       try
       {
           var response = await bedrockClient.ListFoundationModelsAsync(new
ListFoundationModelsRequest()
           {
           });

           if (response?.HttpStatusCode == System.Net.HttpStatusCode.OK)
           {
               foreach (var fm in response.ModelSummaries)
               {
                   WriteToConsole(fm);
               }
```

```
            }
            else
            {
                Console.WriteLine("Something wrong happened");
            }
        }
        catch (AmazonBedrockException e)
        {
            Console.WriteLine(e.Message);
        }
    }
```

- Para obter detalhes da API, consulte ListFoundationModels em Referência AWS SDK para .NET da API.

Exemplos do Amazon Bedrock Runtime usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o Amazon Bedrock Runtime.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Claude da Anthropic
- Command da Cohere
- Llama da Meta
- Mistral AI
- Stable Image Core

Conceitos básicos

Olá, Amazon Bedrock Runtime

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock Runtime.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

namespace BedrockRuntimeActions;

/// <summary>
/// This example shows how to use the Converse API to send a text message
/// to Amazon Bedrock.
/// </summary>
internal class HelloBedrockRuntime
{
    static async Task Main(string[] args)
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // You can change the region to match your setup.
        var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USWest2);

        // Set the model ID, e.g., Claude Haiku.
        // The "global." prefix enables cross-region inference, allowing the request
        // to be routed to the nearest available region for the specified model.
        var modelId = "global.anthropic.claude-haiku-4-5-20251001-v1:0";

        // Define the user message.
        var userMessage = "Hi. In a short paragraph, explain what you can do.";

        // Create a request with the model ID, the user message, and an inference
 configuration.
        var request = new ConverseRequest
        {
            ModelId = modelId,
            Messages = new List<Message>
            {
```

```
                new Message
                {
                    Role = ConversationRole.User,
                    Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
                }
            }
        };

        try
        {
            // Send the request to the Bedrock Runtime and wait for the response.
            var response = await client.ConverseAsync(request);

            // Extract and print the response text.
            string responseText = response?.Output?.Message?.Content?[0]?.Text ??
 "";
            Console.WriteLine(responseText);
        }
        catch (AmazonBedrockRuntimeException e)
        {
            Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason:
 {e.Message}");
            throw;
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

Claude da Anthropic

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
```

```
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Anthropic Claude
```

```
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);
```

```
    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para .NET da API.

Command da Cohere

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Cohere Command usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Cohere Command.

using System;
using System.Collections.Generic;
```

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
```

```
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";
```

```
// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para .NET da API.

Llama da Meta

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

SDK para .NET (v4)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Meta Llama.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
```

```
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para .NET .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
```

```
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para .NET da API.

Mistral AI

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Mistral.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
```

```
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para .NET .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para a Mistral usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Mistral
```

```
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);
```

```
    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para .NET da API.

Stable Image Core

InvokeModel

O exemplo de código a seguir mostra como invocar o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Image Core.

```
        /// <summary>
        /// Asynchronously invokes the Stability.ai Stable Image Core model to run
 an inference based on the provided input.
```

```
        /// </summary>
        /// <param name="prompt">The prompt that describes the image Stability.ai
 Stable Image Core has to generate.</param>
        /// <returns>A base-64 encoded image generated by model</returns>
        /// <remarks>
        /// The different model providers have individual request and response
 formats.
        /// For the format, ranges, and default values for Stability.ai Stable Image
 Core, refer to:
        ///     https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-diffusion-stable-image-core-text-image-request-response.html
        /// </remarks>
        public static async Task<string?> InvokeStableImageCoreAsync(string prompt,
 int seed)
        {
            string stableImageCoreModelId = "stability.stable-image-core-v1:1";

            AmazonBedrockRuntimeClient client = new(RegionEndpoint.USWest2);

            string payload = new JsonObject()
            {
                { "prompt", prompt },
                { "aspect_ratio", "1:1" },
                { "seed", seed },
                { "output_format", "png" }
            }.ToJsonString();

            try
            {
                InvokeModelResponse response = await client.InvokeModelAsync(new
 InvokeModelRequest()
                {
                    ModelId = stableImageCoreModelId,
                    Body = AWSSDKUtils.GenerateMemoryStreamFromString(payload),
                    ContentType = "application/json",
                    Accept = "application/json"
                });

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    var results = JsonNode.ParseAsync(response.Body).Result?
["images"]?.AsArray();

                    return results?[0]?.GetValue<string>();
```

```
                }
                else
                {
                    Console.WriteLine("InvokeModelAsync failed with status code " +
response.HttpStatusCode);
                }
            }
            catch (AmazonBedrockRuntimeException e)
            {
                Console.WriteLine(e.Message);
            }
            return null;
        }
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK para .NET da API.

CloudFormation exemplos usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com CloudFormation.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Hello CloudFormation

O exemplo de código a seguir mostra como começar a usar o CloudFormation.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.Runtime;

namespace CloudFormationActions;

public static class HelloCloudFormation
{
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    static async Task Main(string[] args)
    {
        // Create the CloudFormation client
        _amazonCloudFormation = new AmazonCloudFormationClient();
        Console.WriteLine($"\nIn Region:
 {_amazonCloudFormation.Config.RegionEndpoint}");

        // List the resources for each stack
        await ListResources();
    }

    /// <summary>
    /// Method to list stack resources and other information.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> ListResources()
    {
        try
        {
            Console.WriteLine("Getting CloudFormation stack information...");

            // Get all stacks using the stack paginator.
            var paginatorForDescribeStacks =
                _amazonCloudFormation.Paginators.DescribeStacks(
```

```
                    new DescribeStacksRequest());
            if (paginatorForDescribeStacks.Stacks != null)
            {
                await foreach (Stack stack in paginatorForDescribeStacks.Stacks)
                {
                    // Basic information for each stack
                    Console.WriteLine(
                        "\n-----------------------------------------------");
                    Console.WriteLine($"\nStack: {stack.StackName}");
                    Console.WriteLine($"  Status: {stack.StackStatus.Value}");
                    Console.WriteLine($"  Created: {stack.CreationTime}");

                    // The tags of each stack (etc.)
                    if (stack.Tags != null && stack.Tags.Count > 0)
                    {
                        Console.WriteLine("  Tags:");
                        foreach (Tag tag in stack.Tags)
                            Console.WriteLine($"    {tag.Key}, {tag.Value}");
                    }

                    // The resources of each stack
                    DescribeStackResourcesResponse responseDescribeResources =
                        await _amazonCloudFormation.DescribeStackResourcesAsync(
                            new DescribeStackResourcesRequest
                            {
                                StackName = stack.StackName
                            });
                    if (responseDescribeResources.StackResources != null &&
responseDescribeResources.StackResources.Count > 0)
                    {
                        Console.WriteLine("  Resources:");
                        foreach (StackResource resource in responseDescribeResources
                                     .StackResources)
                            Console.WriteLine(
                                $"    {resource.LogicalResourceId}:
{resource.ResourceStatus}");
                    }
                }
            }

            Console.WriteLine("\n-----------------------------------------------");
            return true;
        }
        catch (AmazonCloudFormationException ex)
```

```
        {
            Console.WriteLine("Unable to get stack information:\n" + ex.Message);
            return false;
        }
        catch (AmazonServiceException ex)
        {
            if (ex.Message.Contains("Unable to get IAM security credentials"))
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine("If you are usnig SSO, be sure to install" +
                                  " the AWSSDK.SSO and AWSSDK.SSOOIDC packages.");
            }
            else
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }

            return false;
        }
        catch (ArgumentNullException ex)
        {
            if (ex.Message.Contains("Options property cannot be empty: ClientName"))
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine("If you are using SSO, have you logged in?");
            }
            else
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }

            return false;
        }
    }
```

- Para obter detalhes da API, consulte DescribeStackResources em Referência AWS SDK para .NET da API.

CloudWatch exemplos usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com CloudWatch.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá CloudWatch

O exemplo de código a seguir mostra como começar a usar o CloudWatch.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace CloudWatchActions;
```

```
public static class HelloCloudWatch
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon CloudWatch service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatch>()
            ).Build();

        // Now the client is available for injection.
        var cloudWatchClient =
 host.Services.GetRequiredService<IAmazonCloudWatch>();

        // You can use await and any of the async methods to get a response.
        var metricNamespace = "AWS/Billing";
        var response = await cloudWatchClient.ListMetricsAsync(new
 ListMetricsRequest
        {
            Namespace = metricNamespace
        });
        Console.WriteLine($"Hello Amazon CloudWatch! Following are some metrics
 available in the {metricNamespace} namespace:");
        Console.WriteLine();
        if (response.Metrics != null)
        {
            foreach (var metric in response.Metrics.Take(5))
            {
                Console.WriteLine($"\tMetric: {metric.MetricName}");
                Console.WriteLine($"\tNamespace: {metric.Namespace}");
                Console.WriteLine(
                    $"\tDimensions: {string.Join(", ", metric.Dimensions.Select(m =>
 $"{m.Name}:{m.Value}"))}");
                Console.WriteLine();
            }
        }
    }
}
```

- Para obter detalhes da API, consulte ListMetrics em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Liste CloudWatch namespaces e métricas.
- Obter estatísticas para uma métrica e para faturamento estimado.
- Criar e atualizar um painel.
- Criar e adicionar dados a uma métrica.
- Criar e acionar um alarme e, em seguida, visualizar o histórico de alarmes.
- Criar um detector de anomalias.
- Obter uma imagem de métrica e, em seguida, limpar os recursos.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
public class CloudWatchScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
    https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

    This .NET example performs the following tasks:
        1. List and select a CloudWatch namespace.
        2. List and select a CloudWatch metric.
        3. Get statistics for a CloudWatch metric.
```

```
        4. Get estimated billing statistics for the last week.
        5. Create a new CloudWatch dashboard with two metrics.
        6. List current CloudWatch dashboards.
        7. Create a CloudWatch custom metric and add metric data.
        8. Add the custom metric to the dashboard.
        9. Create a CloudWatch alarm for the custom metric.
       10. Describe current CloudWatch alarms.
       11. Get recent data for the custom metric.
       12. Add data to the custom metric to trigger the alarm.
       13. Wait for an alarm state.
       14. Get history for the CloudWatch alarm.
       15. Add an anomaly detector.
       16. Describe current anomaly detectors.
       17. Get and display a metric image.
       18. Clean up resources.
   */

   private static ILogger logger = null!;
   private static CloudWatchWrapper _cloudWatchWrapper = null!;
   private static IConfiguration _configuration = null!;
   private static readonly List<string> _statTypes = new List<string>
{ "SampleCount", "Average", "Sum", "Minimum", "Maximum" };
   private static SingleMetricAnomalyDetector? anomalyDetector = null!;

   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
           .ConfigureServices((_, services) =>
           services.AddAWSService<IAmazonCloudWatch>()
           .AddTransient<CloudWatchWrapper>()
       )
       .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
```

```
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CloudWatchScenario>();

        _cloudWatchWrapper = host.Services.GetRequiredService<CloudWatchWrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon CloudWatch example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            var selectedNamespace = await SelectNamespace();
            var selectedMetric = await SelectMetric(selectedNamespace);
            await GetAndDisplayMetricStatistics(selectedNamespace, selectedMetric);
            await GetAndDisplayEstimatedBilling();
            await CreateDashboardWithMetrics();
            await ListDashboards();
            await CreateNewCustomMetric();
            await AddMetricToDashboard();
            await CreateMetricAlarm();
            await DescribeAlarms();
            await GetCustomMetricData();
            await AddMetricDataForAlarm();
            await CheckForMetricAlarm();
            await GetAlarmHistory();
            anomalyDetector = await AddAnomalyDetector();
            await DescribeAnomalyDetectors();
            await GetAndOpenMetricImage();
            await CleanupResources();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources();
        }

    }

    /// <summary>
    /// Select a namespace.
    /// </summary>
    /// <returns>The selected namespace.</returns>
```

```
    private static async Task<string> SelectNamespace()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. Select a CloudWatch Namespace from a list of
Namespaces.");
        var metrics = await _cloudWatchWrapper.ListMetrics();
        // Get a distinct list of namespaces.
        var namespaces = metrics.Select(m => m.Namespace).Distinct().ToList();
        for (int i = 0; i < namespaces.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {namespaces[i]}");
        }

        var namespaceChoiceNumber = 0;
        while (namespaceChoiceNumber < 1 || namespaceChoiceNumber >
namespaces.Count)
        {
            Console.WriteLine(
                "Select a namespace by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out namespaceChoiceNumber);
        }

        var selectedNamespace = namespaces[namespaceChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedNamespace;
    }

    /// <summary>
    /// Select a metric from a namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <returns>The metric name.</returns>
    private static async Task<Metric> SelectMetric(string metricNamespace)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. Select a CloudWatch metric from a namespace.");

        var namespaceMetrics = await
_cloudWatchWrapper.ListMetrics(metricNamespace);

        for (int i = 0; i < namespaceMetrics.Count && i < 15; i++)
```

```
        {
            var dimensionsWithValues = namespaceMetrics[i].Dimensions
                .Where(d => !string.Equals("None", d.Value));
            Console.WriteLine($"\t{i + 1}. {namespaceMetrics[i].MetricName} " +
                              $"{string.Join(", :", dimensionsWithValues.Select(d =>
d.Value))}");
        }

        var metricChoiceNumber = 0;
        while (metricChoiceNumber < 1 || metricChoiceNumber >
namespaceMetrics.Count)
        {
            Console.WriteLine(
                "Select a metric by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out metricChoiceNumber);
        }

        var selectedMetric = namespaceMetrics[metricChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedMetric;
    }

    /// <summary>
    /// Get and display metric statistics for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayMetricStatistics(string metricNamespace,
Metric metric)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. Get CloudWatch metric statistics for the last day.");

        for (int i = 0; i < _statTypes.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {_statTypes[i]}");
        }

        var statisticChoiceNumber = 0;
```

```
        while (statisticChoiceNumber < 1 || statisticChoiceNumber >
_statTypes.Count)
        {
            Console.WriteLine(
                "Select a metric statistic by entering a number from the preceding
list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out statisticChoiceNumber);
        }

        var selectedStatistic = _statTypes[statisticChoiceNumber - 1];
        var statisticsList = new List<string> { selectedStatistic };

        var metricStatistics = await
_cloudWatchWrapper.GetMetricStatistics(metricNamespace, metric.MetricName,
statisticsList, metric.Dimensions, 1, 60);

        if (!metricStatistics.Any())
        {
            Console.WriteLine($"No {selectedStatistic} statistics found for {metric}
in namespace {metricNamespace}.");
        }

        metricStatistics = metricStatistics.OrderBy(s => s.Timestamp).ToList();
        for (int i = 0; i < metricStatistics.Count && i < 10; i++)
        {
            var metricStat = metricStatistics[i];
            var statValue =
metricStat.GetType().GetProperty(selectedStatistic)!.GetValue(metricStat, null);
            Console.WriteLine($"\t{i + 1}. Timestamp
{metricStatistics[i].Timestamp:G} {selectedStatistic}: {statValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get and display estimated billing statistics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayEstimatedBilling()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Get CloudWatch estimated billing for the last
week.");

        var billingStatistics = await SetupBillingStatistics();

        for (int i = 0; i < billingStatistics.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. Timestamp
{billingStatistics[i].Timestamp:G} : {billingStatistics[i].Maximum}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the past
seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Create a dashboard with metrics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CreateDashboardWithMetrics()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Create a new CloudWatch dashboard with metrics.");
        var dashboardName = _configuration["dashboardName"];
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);
        var newDashboardString = JsonSerializer.Serialize(
            newDashboard,
            new JsonSerializerOptions
            {
                DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull
            });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} was created.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List dashboards.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDashboards()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. List the CloudWatch dashboards in the current
account.");

        var dashboards = await _cloudWatchWrapper.ListDashboards();

        for (int i = 0; i < dashboards.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {dashboards[i].DashboardName}");
        }

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Create and add data for a new custom metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateNewCustomMetric()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Create and add data for a new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var customData = await PutRandomMetricData(customMetricName,
customMetricNamespace);

        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric {customMetricName}:
\n\t{valuesString}");

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes in
the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
```

```
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace, customData);
        return customData;
    }

    /// <summary>
    /// Add the custom metric to the dashboard.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricToDashboard()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Add the new custom metric to the dashboard.");

        var dashboardName = _configuration["dashboardName"];

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var validationMessages = await SetupDashboard(customMetricNamespace,
customMetricName, dashboardName);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} updated with metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
```

```
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
```

```
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Create a CloudWatch alarm for the new metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Create a CloudWatch alarm for the new metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var alarmName = _configuration["exampleAlarmName"];
        var accountId = _configuration["accountId"];
        var region = _configuration["region"];
        var emailTopic = _configuration["emailTopic"];
        var alarmActions = new List<string>();

        if (GetYesNoResponse(
                $"\tAdd an email action for topic {emailTopic} to alarm {alarmName}?
(y/n)"))
        {
            _cloudWatchWrapper.AddEmailAlarmAction(accountId, region, emailTopic,
alarmActions);
        }

        await _cloudWatchWrapper.PutMetricEmailAlarm(
            "Example metric alarm",
            alarmName,
            ComparisonOperator.GreaterThanOrEqualToThreshold,
            customMetricName,
            customMetricNamespace,
            100,
            alarmActions);

        Console.WriteLine($"\tAlarm {alarmName} added for metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Describe Alarms.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAlarms()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Describe CloudWatch alarms in the current
account.");

        var alarms = await _cloudWatchWrapper.DescribeAlarms();
        alarms = alarms.OrderByDescending(a => a.StateUpdatedTimestamp).ToList();

        for (int i = 0; i < alarms.Count && i < 10; i++)
        {
            var alarm = alarms[i];
            Console.WriteLine($"\t{i + 1}. {alarm.AlarmName}");
            Console.WriteLine($"\tState: {alarm.StateValue} for {alarm.MetricName}
{alarm.ComparisonOperator} {alarm.Threshold}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the recent data for the metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetCustomMetricData()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. Get current data for new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var accountId = _configuration["accountId"];

        var query = new List<MetricDataQuery>
        {
            new MetricDataQuery
            {
                AccountId = accountId,
```

```
                Id = "m1",
                Label = "Custom Metric Data",
                MetricStat = new MetricStat
                {
                    Metric = new Metric
                    {
                        MetricName = customMetricName,
                        Namespace = customMetricNamespace,
                    },
                    Period = 1,
                    Stat = "Maximum"
                }
            }
        };

        var metricData = await _cloudWatchWrapper.GetMetricData(
            20,
            true,
            DateTime.UtcNow.AddMinutes(1),
            20,
            query);

        for (int i = 0; i < metricData.Count; i++)
        {
            if (metricData[i].Values != null)
            {
                for (int j = 0; j < metricData[i].Values.Count; j++)
                {
                    Console.WriteLine(
                        $"\tTimestamp {metricData[i].Timestamps[j]:G} Value:
{metricData[i].Values[j]}");
                }
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add metric data to trigger an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricDataForAlarm()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"12. Add metric data to the custom metric to trigger an
alarm.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var nowUtc = DateTime.UtcNow;
        List<MetricDatum> customData = new List<MetricDatum>
        {
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-2)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-1)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc
            }
        };
        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric {customMetricName}:
\n\t{valuesString}");
        await _cloudWatchWrapper.PutMetricData(customMetricNamespace, customData);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check for a metric alarm using the DescribeAlarmsForMetric action.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckForMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"13. Checking for an alarm state.");
```

```
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var hasAlarm = false;
        var retries = 10;
        while (!hasAlarm && retries > 0)
        {
            var alarms = await
_cloudWatchWrapper.DescribeAlarmsForMetric(customMetricNamespace,
customMetricName);
            hasAlarm = alarms.Any(a => a.StateValue == StateValue.ALARM);
            retries--;
            Thread.Sleep(20000);
        }

        Console.WriteLine(hasAlarm
            ? $"\tAlarm state found for {customMetricName}."
            : $"\tNo Alarm state found for {customMetricName} after 10 retries.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get history for an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAlarmHistory()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"14. Get alarm history.");

        var exampleAlarmName = _configuration["exampleAlarmName"];

        var alarmHistory = await
_cloudWatchWrapper.DescribeAlarmHistory(exampleAlarmName, 2);

        for (int i = 0; i < alarmHistory.Count; i++)
        {
            var history = alarmHistory[i];
            Console.WriteLine($"\t{i + 1}. {history.HistorySummary}, time
{history.Timestamp:g}");
        }
        if (!alarmHistory.Any())
        {
```

```
            Console.WriteLine($"\tNo alarm history data found for
{exampleAlarmName}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an anomaly detector.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<SingleMetricAnomalyDetector> AddAnomalyDetector()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"15. Add an anomaly detector.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detector = new SingleMetricAnomalyDetector
        {
            MetricName = customMetricName,
            Namespace = customMetricNamespace,
            Stat = "Maximum"
        };
        await _cloudWatchWrapper.PutAnomalyDetector(detector);
        Console.WriteLine($"\tAdded anomaly detector for metric
{customMetricName}.");

        Console.WriteLine(new string('-', 80));
        return detector;
    }

    /// <summary>
    /// Describe anomaly detectors.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAnomalyDetectors()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"16. Describe anomaly detectors in the current
account.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
```

```
        var customMetricName = _configuration["customMetricName"];

        var detectors = await
_cloudWatchWrapper.DescribeAnomalyDetectors(customMetricNamespace,
customMetricName);

        for (int i = 0; i < detectors.Count; i++)
        {
            var detector = detectors[i];
            Console.WriteLine($"\t{i + 1}.
{detector.SingleMetricAnomalyDetector.MetricName}, state {detector.StateValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Fetch and open a metrics image for a CloudWatch metric and namespace.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAndOpenMetricImage()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("17. Get a metric image from CloudWatch.");

        Console.WriteLine($"\tGetting Image data for custom metric.");
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var memoryStream = await
_cloudWatchWrapper.GetTimeSeriesMetricImage(customMetricNamespace,
customMetricName, "Maximum", 10);
        var file = _cloudWatchWrapper.SaveMetricImage(memoryStream, "MetricImages");

        ProcessStartInfo info = new ProcessStartInfo();

        Console.WriteLine($"\tFile saved as {Path.GetFileName(file)}.");
        Console.WriteLine($"\tPress enter to open the image.");
        Console.ReadLine();
        info.FileName = Path.Combine("ms-photos://", file);
        info.UseShellExecute = true;
        info.CreateNoWindow = true;
        info.Verb = string.Empty;
```

```
        Process.Start(info);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up created resources.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"18. Clean up resources.");

        var dashboardName = _configuration["dashboardName"];
        if (GetYesNoResponse($"\tDelete dashboard {dashboardName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting dashboard.");
            var dashboardList = new List<string> { dashboardName };
            await _cloudWatchWrapper.DeleteDashboards(dashboardList);
        }

        var alarmName = _configuration["exampleAlarmName"];
        if (GetYesNoResponse($"\tDelete alarm {alarmName}? (y/n)"))
        {
            Console.WriteLine($"\tCleaning up alarms.");
            var alarms = new List<string> { alarmName };
            await _cloudWatchWrapper.DeleteAlarms(alarms);
        }

        if (GetYesNoResponse($"\tDelete anomaly detector? (y/n)") &&
anomalyDetector != null)
        {
            Console.WriteLine($"\tCleaning up anomaly detector.");

            await _cloudWatchWrapper.DeleteAnomalyDetector(
                anomalyDetector);
        }

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Métodos de invólucro usados pelo cenário para CloudWatch ações.

```
/// <summary>
/// Wrapper class for Amazon CloudWatch methods.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;
    private readonly ILogger<CloudWatchWrapper> _logger;

    /// <summary>
    /// Constructor for the CloudWatch wrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch,
 ILogger<CloudWatchWrapper> logger)

    {
        _logger = logger;
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
```

```
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter> { filter } :
null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions, int
days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
```

```
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints ?? new List<Datapoint>();
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard was
 created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages ?? new
 List<DashboardValidationMessage>();
    }


    /// <summary>
    /// Get information on a dashboard.
    /// </summary>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A JSON object with dashboard information.</returns>
    public async Task<string> GetDashboard(string dashboardName)
    {
```

```
        var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
            new GetDashboardRequest()
            {
                DashboardName = dashboardName
            });

        return dashboardResponse.DashboardBody;
    }


    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });
```

```
        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string metricNamespace,
string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString = JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
```

```
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }

    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData, bool
 useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        // Add the plus sign for positive offsets.
        timeZoneString = timeZoneString.StartsWith('-') ? timeZoneString : "+" +
 timeZoneString;
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
```

```
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        if (paginatedMetricData.MetricDataResults != null)
        {
            await foreach (var data in paginatedMetricData.MetricDataResults)
            {
                metricData.Add(data);
            }
        }

        return metricData;
    }

    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
    {
        try
        {
            var putEmailAlarmResponse = await _amazonCloudWatch.PutMetricAlarmAsync(
                new PutMetricAlarmRequest()
                {
                    AlarmActions = alarmActions,
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
                    ComparisonOperator = comparison,
                    Threshold = threshold,
```

```
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota has
already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to append
to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }

    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
```

```
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms = _amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }

    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms ?? new List<MetricAlarm>();
    }

    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string alarmName,
int historyDays)
```

```
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }

    /// <summary>
    /// Delete a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAlarms(List<string> alarmNames)
    {
        var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
            new DeleteAlarmsRequest()
            {
                AlarmNames = alarmNames
            });

        return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
```

```
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
```

```
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }

    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
 _amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Exemplos de valores de settings.json para o cenário.

```
{
  "dashboardName": "example-new-dashboard",
  "exampleAlarmName": "example-metric-alarm",
  "accountId": "1234567890",
  "region": "us-east-1",
  "emailTopic": "Default_CloudWatch_Alarms_Topic",
  "customMetricNamespace": "example-namespace",
  "customMetricName": "example-custom-metric",
  "dashboardExampleBody": {
    "widgets": [
      {
        "height": 6,
        "width": 6,
        "y": 0,
        "x": 0,
        "type": "text",
        "properties": {
          "markdown": "# Code Example Dashboard \nThis dashboard was created by
 example code.\n"
        }
      },
      {
```

```
        "height": 8,
        "width": 8,
        "y": 0,
        "x": 6,
        "type": "metric",
        "properties": {
          "metrics": [
            [
              "AWS/Billing",
              "EstimatedCharges",
              "Currency",
              "USD",
              { "region": "us-east-1" }
            ]
          ],
          "view": "timeSeries",
          "region": "us-east-1",
          "stat": "Maximum",
          "period": 86400,
          "yAxis": {
            "left": {
              "min": 0,
              "max": 100
            }
          },
          "stacked": false,
          "title": "Estimated Billing",
          "setPeriodToTimeRange": false,
          "liveData": true,
          "sparkline": true,
          "trend": true
        }
      },
      {
        "height": 8,
        "width": 8,
        "y": 0,
        "x": 14,
        "type": "metric",
        "properties": {
          "metrics": [
            [ "AWS/Usage", "CallCount", "Type", "API", "Resource", "ListMetrics",
"Service", "CloudWatch", "Class", "None" ],
            [ "...", "GetMetricStatistics", ".", ".", ".", "." ],
```

```
            [ "...", "GetMetricData", ".", ".", ".", "." ],
            [ "...", "PutDashboard", ".", ".", ".", "." ],
            [ "...", "PutMetricData", ".", ".", ".", "." ]
          ],
          "view": "timeSeries",
          "yAxis": {
            "left": {
              "min": 0,
              "max": 200
            }
          },
          "stacked": false,
          "region": "us-east-1",
          "stat": "Sum",
          "period": 300,
          "title": "CloudWatch Usage",
          "setPeriodToTimeRange": false,
          "liveData": true,
          "sparkline": true,
          "trend": true
        }
      }
    ]
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics

- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Ações

`DeleteAlarms`

O código de exemplo a seguir mostra como usar `DeleteAlarms`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Delete a list of alarms from CloudWatch.
/// </summary>
/// <param name="alarmNames">A list of names of alarms to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteAlarms(List<string> alarmNames)
{
    var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
        new DeleteAlarmsRequest()
        {
            AlarmNames = alarmNames
        });

    return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
}
```

- Para obter detalhes da API, consulte DeleteAlarms em Referência AWS SDK para .NET da API.

`DeleteAnomalyDetector`

O código de exemplo a seguir mostra como usar `DeleteAnomalyDetector`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteAnomalyDetector em Referência AWS SDK para .NET da API.

`DeleteDashboards`

O código de exemplo a seguir mostra como usar `DeleteDashboards`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
_amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DeleteDashboards em Referência AWS SDK para .NET da API.

`DescribeAlarmHistory`

O código de exemplo a seguir mostra como usar `DescribeAlarmHistory`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string alarmName,
int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }
```

- Para obter detalhes da API, consulte DescribeAlarmHistory em Referência AWS SDK para .NET da API.

`DescribeAlarms`

O código de exemplo a seguir mostra como usar `DescribeAlarms`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms = _amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }
```

- Para obter detalhes da API, consulte DescribeAlarms em Referência AWS SDK para .NET da API.

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
```

```
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms ?? new List<MetricAlarm>();
    }
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetric em Referência AWS SDK para .NET da API.

`DescribeAnomalyDetectors`

O código de exemplo a seguir mostra como usar `DescribeAnomalyDetectors`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
```

```
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }
```

- Para obter detalhes da API, consulte DescribeAnomalyDetectors em Referência AWS SDK para .NET da API.

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
```

```
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte DisableAlarmActions em Referência AWS SDK para .NET da API.

`EnableAlarmActions`

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- Para obter detalhes da API, consulte EnableAlarmActions em Referência AWS SDK para .NET da API.

GetDashboard

O código de exemplo a seguir mostra como usar `GetDashboard`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get information on a dashboard.
    /// </summary>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A JSON object with dashboard information.</returns>
    public async Task<string> GetDashboard(string dashboardName)
    {
        var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
            new GetDashboardRequest()
            {
                DashboardName = dashboardName
            });

        return dashboardResponse.DashboardBody;
    }
```

- Para obter detalhes da API, consulte GetDashboard em Referência AWS SDK para .NET da API.

`GetMetricData`

O código de exemplo a seguir mostra como usar `GetMetricData`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData, bool
 useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        // Add the plus sign for positive offsets.
        timeZoneString = timeZoneString.StartsWith('-') ? timeZoneString : "+" +
 timeZoneString;
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
```

```
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        if (paginatedMetricData.MetricDataResults != null)
        {
            await foreach (var data in paginatedMetricData.MetricDataResults)
            {
                metricData.Add(data);
            }
        }

        return metricData;
    }
```

- Para obter detalhes da API, consulte GetMetricData em Referência AWS SDK para .NET da API.

GetMetricStatistics

O código de exemplo a seguir mostra como usar `GetMetricStatistics`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
```

```
    {
        // Make a request for EstimatedCharges with a period of one day for the past
seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions, int
days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints ?? new List<Datapoint>();
```

```
    }
```

- Para obter detalhes da API, consulte GetMetricStatistics em Referência AWS SDK para .NET da API.

GetMetricWidgetImage

O código de exemplo a seguir mostra como usar `GetMetricWidgetImage`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string metricNamespace,
string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };
```

```
        var metricImageWidgetString = JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }
```

- Para obter detalhes da API, consulte GetMetricWidgetImage em Referência AWS SDK para .NET da API.

`ListDashboards`

O código de exemplo a seguir mostra como usar `ListDashboards`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }
```

- Para obter detalhes da API, consulte ListDashboards em Referência AWS SDK para .NET da API.

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
```

```
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter> { filter } :
null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }
```

- Para obter detalhes da API, consulte ListMetrics em Referência AWS SDK para .NET da API.

`PutAnomalyDetector`

O código de exemplo a seguir mostra como usar `PutAnomalyDetector`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte PutAnomalyDetector em Referência AWS SDK para .NET da API.

`PutDashboard`

O código de exemplo a seguir mostra como usar `PutDashboard`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string dashboardName)
    {
        // Get the dashboard model from configuration.
```

```
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
```

```
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard was
 created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages ?? new
 List<DashboardValidationMessage>();
    }
```

- Para obter detalhes da API, consulte PutDashboard em Referência AWS SDK para .NET da API.

`PutMetricAlarm`

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
```

```
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
    {
        try
        {
            var putEmailAlarmResponse = await _amazonCloudWatch.PutMetricAlarmAsync(
                new PutMetricAlarmRequest()
                {
                    AlarmActions = alarmActions,
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
                    ComparisonOperator = comparison,
                    Threshold = threshold,
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota has
already been reached.");
        }

        return false;
    }

    /// <summary>
```

```
    /// Add specific email actions to a list of action strings for a CloudWatch
alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to append
to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }
```

- Para obter detalhes da API, consulte PutMetricAlarm em Referência AWS SDK para .NET da API.

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
```

```
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes in
the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace, customData);
        return customData;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });
```

```
        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obter detalhes da API, consulte PutMetricData em Referência AWS SDK para .NET da API.

CloudWatch Exemplos de registros usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com CloudWatch registros.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`GetQueryResults`

O código de exemplo a seguir mostra como usar `GetQueryResults`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets the results of a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="queryId">The ID of the query.</param>
    /// <returns>The query results response.</returns>
    public async Task<GetQueryResultsResponse?> GetQueryResultsAsync(string queryId)
    {
        try
        {
            var request = new GetQueryResultsRequest
            {
                QueryId = queryId
            };

            var response = await
_amazonCloudWatchLogs.GetQueryResultsAsync(request);
            return response;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Query not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while getting query results:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte GetQueryResults em Referência AWS SDK para .NET da API.

StartQuery

O código de exemplo a seguir mostra como usar `StartQuery`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Starts a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="logGroupName">The name of the log group to query.</param>
    /// <param name="queryString">The CloudWatch Logs Insights query string.</param>
    /// <param name="startTime">The start time for the query (seconds since
 epoch).</param>
    /// <param name="endTime">The end time for the query (seconds since epoch).</
param>
    /// <param name="limit">The maximum number of results to return.</param>
    /// <returns>The query ID if successful, null otherwise.</returns>
    public async Task<string?> StartQueryAsync(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit = 10000)
    {
        try
        {
            var request = new StartQueryRequest
            {
                LogGroupName = logGroupName,
                QueryString = queryString,
                StartTime = startTime,
                EndTime = endTime,
                Limit = limit
            };

            var response = await _amazonCloudWatchLogs.StartQueryAsync(request);
            return response.QueryId;
        }
        catch (InvalidParameterException ex)
        {
```

```
            _logger.LogError($"Invalid parameter for query: {ex.Message}");
            return null;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Log group not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while starting query:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte StartQuery em Referência AWS SDK para .NET da API.

Cenários

Executar uma consulta grande

O exemplo de código a seguir mostra como usar o CloudWatch Logs para consultar mais de 10.000 registros.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o principal fluxo de trabalho que demonstra o cenário de grandes consultas.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

```
using CloudWatchLogsActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace CloudWatchLogsScenario;

public class LargeQueryWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the CloudWatch Logs
 Large Query workflow:

    1. Prepare the Application:
       - Prompt the user to deploy CloudFormation stack and generate sample logs.
       - Deploy the CloudFormation template for resource creation.
       - Generate 50,000 sample log entries using CloudWatch Logs API.
       - Wait 5 minutes for logs to be fully ingested.

    2. Execute Large Query:
       - Perform recursive queries to retrieve all logs using binary search.
       - Display progress for each query executed.
       - Show total execution time and logs found.

    3. Clean up:
       - Prompt the user to delete the CloudFormation stack and all resources.
       - Destroy the CloudFormation stack and wait until removed.
    */

    public static ILogger<LargeQueryWorkflow> _logger = null!;
    public static CloudWatchLogsWrapper _wrapper = null!;
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _logGroupName = "/workflows/cloudwatch-logs/large-query";
    private static string _logStreamName = "stream1";
    private static long _queryStartDate;
    private static long _queryEndDate;

    public static bool _interactive = true;
    public static string _stackName = "CloudWatchLargeQueryStack";
    private static string _stackResourcePath = "../../../../../../../scenarios/
features/cloudwatch_logs_large_query/resources/stack.yaml";
```

```
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter("Microsoft", LogLevel.Information))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatchLogs>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<CloudWatchLogsWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<LargeQueryWorkflow>();

            _wrapper = host.Services.GetRequiredService<CloudWatchLogsWrapper>();
            _amazonCloudFormation =
host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to retrieve
more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
```

```
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, initiating
cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Runs the scenario workflow. Used for testing.
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to retrieve
more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
```

```
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, initiating
cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        Console.WriteLine();

        try
        {
            var deployStack = !_interactive || GetYesNoResponse(
                "Would you like to deploy the CloudFormation stack and generate
sample logs? (y/n) ");

            if (deployStack)
            {
                if (_interactive)
                {
                    Console.Write(
                        $"Enter a path for the CloudFormation stack resource .yaml
file (or press Enter for default '{_stackResourcePath}'): ");
                    string? inputPath = Console.ReadLine();
                    if (!string.IsNullOrWhiteSpace(inputPath))
                    {
                        _stackResourcePath = inputPath;
                    }
```

```
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName);

                if (deploySuccess)
                {
                    Console.WriteLine();
                    Console.WriteLine("Generating 50,000 sample log entries...");
                    var generateSuccess = await GenerateSampleLogs();

                    if (generateSuccess)
                    {
                        Console.WriteLine();
                        Console.WriteLine("Sample logs created. Waiting 5 minutes
for logs to be fully ingested...");
                        await WaitWithCountdown(300);

                        Console.WriteLine("Application preparation complete.");
                        return true;
                    }
                }
            }
            else
            {
                _logGroupName = PromptUserForInput("Enter the log group name ",
_logGroupName);
                _logStreamName = PromptUserForInput("Enter the log stream name ",
_logStreamName);

                var startDateMs = PromptUserForLong("Enter the query start date
(milliseconds since epoch): ");
                var endDateMs = PromptUserForLong("Enter the query end date
(milliseconds since epoch): ");

                _queryStartDate = startDateMs / 1000;
                _queryEndDate = endDateMs / 1000;

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
```

```
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }

        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath)
            };

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created successfully.");
                    return true;
                }
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
```

```
                    return false;
                }
            }
            else
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already exists.
Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };
```

```
            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Generates sample logs directly using CloudWatch Logs API.
    /// Creates 50,000 log entries spanning 5 minutes.
    /// </summary>
    /// <returns>True if logs were generated successfully.</returns>
    private static async Task<bool> GenerateSampleLogs()
    {
        const int totalEntries = 50000;
        const int entriesPerBatch = 10000;
        const int fiveMinutesMs = 5 * 60 * 1000;

        try
        {
            // Calculate timestamps
```

```
            var startTimeMs = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();
            var timestampIncrement = fiveMinutesMs / totalEntries;

            Console.WriteLine($"Generating {totalEntries} log entries...");

            var entryCount = 0;
            var currentTimestamp = startTimeMs;
            var numBatches = totalEntries / entriesPerBatch;

            // Generate and upload logs in batches
            for (int batchNum = 0; batchNum < numBatches; batchNum++)
            {
                var logEvents = new List<InputLogEvent>();

                for (int i = 0; i < entriesPerBatch; i++)
                {
                    logEvents.Add(new InputLogEvent
                    {
                        Timestamp =
 DateTimeOffset.FromUnixTimeMilliseconds(currentTimestamp).UtcDateTime,
                        Message = $"Entry {entryCount}"
                    });

                    entryCount++;
                    currentTimestamp += timestampIncrement;
                }

                // Upload batch
                var success = await _wrapper.PutLogEventsAsync(_logGroupName,
 _logStreamName, logEvents);
                if (!success)
                {
                    _logger.LogError($"Failed to upload batch {batchNum + 1}/
{numBatches}");
                    return false;
                }

                Console.WriteLine($"Uploaded batch {batchNum + 1}/{numBatches}");
            }

            // Set query date range (convert milliseconds to seconds for query API)
            _queryStartDate = startTimeMs / 1000;
            _queryEndDate = (currentTimestamp - timestampIncrement) / 1000;
```

```
            Console.WriteLine($"Query start date:
{DateTimeOffset.FromUnixTimeSeconds(_queryStartDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Query end date:
{DateTimeOffset.FromUnixTimeSeconds(_queryEndDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Successfully uploaded {totalEntries} log entries");

            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while generating sample logs.");
            return false;
        }
    }

    /// <summary>
    /// Executes the large query workflow.
    /// </summary>
    public static async Task ExecuteLargeQuery()
    {
        Console.WriteLine("Starting recursive query to retrieve all logs...");
        Console.WriteLine();

        var queryLimit = PromptUserForInteger("Enter the query limit (max 10000) ",
10000);
        if (queryLimit > 10000) queryLimit = 10000;

        var queryString = "fields @timestamp, @message | sort @timestamp asc";

        var stopwatch = Stopwatch.StartNew();
        var allResults = await PerformLargeQuery(_logGroupName, queryString,
_queryStartDate, _queryEndDate, queryLimit);
        stopwatch.Stop();

        Console.WriteLine();
        Console.WriteLine($"Queries finished in {stopwatch.Elapsed.TotalSeconds:F3}
seconds.");
        Console.WriteLine($"Total logs found: {allResults.Count}");

        // Check for duplicates
        Console.WriteLine();
        Console.WriteLine("Checking for duplicate logs...");
        var duplicates = FindDuplicateLogs(allResults);
        if (duplicates.Count > 0)
```

```
        {
            Console.WriteLine($"WARNING: Found {duplicates.Count} duplicate log
entries!");
            Console.WriteLine("Duplicate entries (showing first 10):");
            foreach (var dup in duplicates.Take(10))
            {
                Console.WriteLine($"  [{dup.Timestamp}] {dup.Message} (appears
{dup.Count} times)");
            }

            var uniqueCount = allResults.Count - duplicates.Sum(d => d.Count - 1);
            Console.WriteLine($"Unique logs: {uniqueCount}");
        }
        else
        {
            Console.WriteLine("No duplicates found. All logs are unique.");
        }
        Console.WriteLine();

        var viewSample = !_interactive || GetYesNoResponse("Would you like to see a
sample of the logs? (y/n) ");
        if (viewSample)
        {
            Console.WriteLine();
            Console.WriteLine($"Sample logs (first 10 of {allResults.Count}):");
            for (int i = 0; i < Math.Min(10, allResults.Count); i++)
            {
                var timestamp = allResults[i].Find(f => f.Field ==
"@timestamp")?.Value ?? "N/A";
                var message = allResults[i].Find(f => f.Field ==
"@message")?.Value ?? "N/A";
                Console.WriteLine($"[{timestamp}] {message}");
            }
        }
    }

    /// <summary>
    /// Performs a large query using recursive binary search.
    /// </summary>
    private static async Task<List<List<ResultField>>> PerformLargeQuery(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
```

```
        int limit)
    {
        var queryId = await _wrapper.StartQueryAsync(logGroupName, queryString,
 startTime, endTime, limit);
        if (queryId == null)
        {
            return new List<List<ResultField>>();
        }

        var results = await PollQueryResults(queryId);
        if (results == null || results.Count == 0)
        {
            return new List<List<ResultField>>();
        }

        var startDate =
 DateTimeOffset.FromUnixTimeSeconds(startTime).ToString("yyyy-MM-ddTHH:mm:ss.fffZ");
        var endDate = DateTimeOffset.FromUnixTimeSeconds(endTime).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        Console.WriteLine($"Query date range: {startDate} ({startTime}s) to
 {endDate} ({endTime}s). Found {results.Count} logs.");

        if (results.Count < limit)
        {
            Console.WriteLine($"  -> Returning {results.Count} logs (less than limit
 of {limit})");
            return results;
        }

        Console.WriteLine($"  -> Hit limit of {limit}. Need to split and recurse.");

        // Get the timestamp of the last log (sorted to find the actual last one)
        var lastLogTimestamp = GetLastLogTimestamp(results);
        if (lastLogTimestamp == null)
        {
            Console.WriteLine($"  -> No timestamp found in results. Returning
 {results.Count} logs.");
            return results;
        }

        Console.WriteLine($"  -> Last log timestamp: {lastLogTimestamp}");

        // Parse the timestamp and add 1 millisecond to avoid querying the same log
 again
```

```
        var lastLogDate = DateTimeOffset.Parse(lastLogTimestamp + " +0000");
        Console.WriteLine($"  -> Last log as DateTimeOffset: {lastLogDate:yyyy-MM-
ddTHH:mm:ss.fffZ} ({lastLogDate.ToUnixTimeSeconds()}s)");

        var offsetLastLogDate = lastLogDate.AddMilliseconds(1);
        Console.WriteLine($"  -> Offset timestamp (last
 + 1ms): {offsetLastLogDate:yyyy-MM-ddTHH:mm:ss.fffZ}
 ({offsetLastLogDate.ToUnixTimeSeconds()}s)");

        // Convert to seconds, but round UP to the next second to avoid overlapping
 with logs in the same second
        // This ensures we don't re-query logs that share the same second as the
 last log
        var offsetLastLogTime = offsetLastLogDate.ToUnixTimeSeconds();
        if (offsetLastLogDate.Millisecond > 0)
        {
            offsetLastLogTime++; // Move to the next full second
            Console.WriteLine($"  -> Adjusted to next full second:
 {offsetLastLogTime}s ({DateTimeOffset.FromUnixTimeSeconds(offsetLastLogTime):yyyy-
MM-ddTHH:mm:ss.fffZ})");
        }

        Console.WriteLine($"  -> Comparing: offsetLastLogTime={offsetLastLogTime}s
 vs endTime={endTime}s");
        Console.WriteLine($"  -> End time as date:
 {DateTimeOffset.FromUnixTimeSeconds(endTime):yyyy-MM-ddTHH:mm:ss.fffZ}");

        // Check if there's any time range left to query
        if (offsetLastLogTime >= endTime)
        {
            Console.WriteLine($"  -> No time range left to query. Offset time
 ({offsetLastLogTime}s) >= end time ({endTime}s)");
            return results;
        }

        // Split the remaining date range in half
        var (range1Start, range1End, range2Start, range2End) =
 SplitDateRange(offsetLastLogTime, endTime);

        var range1StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range1Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range1EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range1End).ToString("yyyy-MM-ddTHH:mm:ss.fffZ");
```

```
        var range2StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range2Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range2End).ToString("yyyy-MM-ddTHH:mm:ss.fffZ");

        Console.WriteLine($"  -> Splitting remaining range:");
        Console.WriteLine($"     Range 1: {range1StartDate} ({range1Start}s) to
 {range1EndDate} ({range1End}s)");
        Console.WriteLine($"     Range 2: {range2StartDate} ({range2Start}s) to
 {range2EndDate} ({range2End}s)");

        // Query both halves recursively
        Console.WriteLine($"  -> Querying range 1...");
        var results1 = await PerformLargeQuery(logGroupName, queryString,
 range1Start, range1End, limit);
        Console.WriteLine($"  -> Range 1 returned {results1.Count} logs");

        Console.WriteLine($"  -> Querying range 2...");
        var results2 = await PerformLargeQuery(logGroupName, queryString,
 range2Start, range2End, limit);
        Console.WriteLine($"  -> Range 2 returned {results2.Count} logs");

        // Combine all results
        var allResults = new List<List<ResultField>>(results);
        allResults.AddRange(results1);
        allResults.AddRange(results2);

        Console.WriteLine($"  -> Combined total: {allResults.Count} logs
 ({results.Count} + {results1.Count} + {results2.Count})");

        return allResults;
    }

    /// <summary>
    /// Gets the timestamp string of the most recent log from a list of logs.
    /// Sorts timestamps to find the actual last one.
    /// </summary>
    private static string? GetLastLogTimestamp(List<List<ResultField>> logs)
    {
        var timestamps = logs
            .Select(log => log.Find(f => f.Field == "@timestamp")?.Value)
            .Where(t => !string.IsNullOrEmpty(t))
            .OrderBy(t => t)
```

```
            .ToList();

        if (timestamps.Count == 0)
        {
            return null;
        }

        return timestamps[timestamps.Count - 1];
    }

    /// <summary>
    /// Splits a date range in half.
    /// Range 2 starts at midpoint + 1 second to avoid overlap.
    /// </summary>
    private static (long range1Start, long range1End, long range2Start, long
range2End) SplitDateRange(long startTime, long endTime)
    {
        var midpoint = startTime + (endTime - startTime) / 2;
        // Range 2 starts at midpoint + 1 to avoid querying the same second twice
        return (startTime, midpoint, midpoint + 1, endTime);
    }

    /// <summary>
    /// Polls for query results until complete.
    /// </summary>
    private static async Task<List<List<ResultField>>?> PollQueryResults(string
queryId)
    {
        int retryCount = 0;
        const int maxRetries = 60;
        const int retryDelay = 1000;

        while (retryCount < maxRetries)
        {
            var response = await _wrapper.GetQueryResultsAsync(queryId);
            if (response == null)
            {
                return null;
            }

            if (response.Status == QueryStatus.Complete)
            {
                return response.Results;
            }
```

```
            if (response.Status == QueryStatus.Failed ||
                response.Status == QueryStatus.Cancelled ||
                response.Status == QueryStatus.Timeout ||
                response.Status == QueryStatus.Unknown)
            {
                _logger.LogError($"Query failed with status: {response.Status}");
                return null;
            }

            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for query results.");
        return null;
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    public static async Task<bool> Cleanup()
    {
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete the CloudFormation stack and all resources? (y/n)
");

        if (cleanup)
        {
            try
            {
                var stackDeleteSuccess = await DeleteCloudFormationStack(_stackName,
false);
                return stackDeleteSuccess;
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "An error occurred while cleaning up the
resources.");
                return false;
            }
        }
```

```
        Console.WriteLine($"Resources will remain. Stack name: {_stackName}, Log
group: {_logGroupName}");
        _logger.LogInformation("CloudWatch Logs Large Query scenario is complete.");
        return true;
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    private static async Task<bool> DeleteCloudFormationStack(string stackName, bool
forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(stackName, forceDelete);

        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
            return true;
        }
        else
        {
            _logger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
```

```
    /// </summary>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }

                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_FAILED)
                {
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to be
deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }
```

```
        _logger.LogError($"Timed out waiting for CloudFormation stack '{stackName}'
to be deleted.");
        return false;
    }

    /// <summary>
    /// Waits with a countdown display.
    /// </summary>
    private static async Task WaitWithCountdown(int seconds)
    {
        for (int i = seconds; i > 0; i--)
        {
            Console.Write($"\rWaiting: {i} seconds remaining...  ");
            await Task.Delay(1000);
        }
        Console.WriteLine("\rWait complete.                      ");
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (_interactive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press Enter
for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = "[a-zA-Z][-a-zA-Z0-9]*";
```

```
                if (!Regex.IsMatch(input, regex))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
{_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }

    /// <summary>
    /// Prompts the user for input with a default value.
    /// </summary>
    private static string PromptUserForInput(string prompt, string defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'): ");
            string? input = Console.ReadLine();
            return string.IsNullOrWhiteSpace(input) ? defaultValue : input;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for an integer value.
    /// </summary>
    private static int PromptUserForInteger(string prompt, int defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'): ");
            string? input = Console.ReadLine();
            if (string.IsNullOrWhiteSpace(input) || !int.TryParse(input, out var
result))
            {
                return defaultValue;
            }
            return result;
        }
        return defaultValue;
    }
```

```
    /// <summary>
    /// Prompts the user for a long value.
    /// </summary>
    private static long PromptUserForLong(string prompt)
    {
        if (_interactive)
        {
            Console.Write(prompt);
            string? input = Console.ReadLine();
            if (long.TryParse(input, out var result))
            {
                return result;
            }
        }
        return 0;
    }

    /// <summary>
    /// Finds duplicate log entries based on timestamp and message.
    /// </summary>
    private static List<(string Timestamp, string Message, int Count)>
FindDuplicateLogs(List<List<ResultField>> logs)
    {
        var logSignatures = new Dictionary<string, int>();

        foreach (var log in logs)
        {
            var timestamp = log.Find(f => f.Field == "@timestamp")?.Value ?? "";
            var message = log.Find(f => f.Field == "@message")?.Value ?? "";
            var signature = $"{timestamp}|{message}";

            if (logSignatures.ContainsKey(signature))
            {
                logSignatures[signature]++;
            }
            else
            {
                logSignatures[signature] = 1;
            }
        }

        return logSignatures
            .Where(kvp => kvp.Value > 1)
```

```
            .Select(kvp =>
            {
                var parts = kvp.Key.Split('|');
                return (Timestamp: parts[0], Message: parts[1], Count: kvp.Value);
            })
            .OrderByDescending(x => x.Count)
            .ToList();
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - GetQueryResults
  - StartQuery

Exemplos do Amazon Cognito Identity Provider usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListUserPools`

O código de exemplo a seguir mostra como usar `ListUserPools`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- Para obter detalhes da API, consulte ListUserPools em Referência AWS SDK para .NET da API.

AWS Control Tower exemplos usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com AWS Control Tower.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Control Tower

O exemplo de código a seguir mostra como começar a usar o AWS Control Tower.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
using LogLevel = Microsoft.Extensions.Logging.LogLevel;

namespace ControlTowerActions;

/// <summary>
/// A class that introduces the AWS Control Tower by listing the
```

```
/// available baselines for the account.
/// </summary>
public class HelloControlTower
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Control Tower.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonControlTower>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloControlTower>();

        var amazonClient = host.Services.GetRequiredService<IAmazonControlTower>();

        Console.Clear();
        Console.WriteLine("Hello, AWS Control Tower! Let's list available
 baselines:");
        Console.WriteLine();

        var baselines = new List<BaselineSummary>();

        try
        {
            var baselinesPaginator = amazonClient.Paginators.ListBaselines(new
 ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            Console.WriteLine($"{baselines.Count} baseline(s) retrieved.");
            foreach (var baseline in baselines)
```

```
            {
                Console.WriteLine($"\t{baseline.Name}");
            }
        }
        catch (Amazon.ControlTower.Model.AccessDeniedException)
        {
            Console.WriteLine("Access denied. Please ensure you have the necessary
 permissions.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListBaselines em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Liste as zonas de pouso.
- Liste, ative, obtenha, redefina e desative as linhas de base.
- Liste, ative, obtenha e desative controles.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS Control Tower recursos.

```
using Amazon.ControlCatalog;
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using Amazon.Organizations;
using Amazon.Organizations.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using ControlTowerActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace ControlTowerBasics;

/// <summary>
/// Scenario class for AWS Control Tower basics.
/// </summary>
public class ControlTowerBasics
{
    public static bool isInteractive = true;
    public static ILogger logger = null!;
    public static IAmazonOrganizations? orgClient = null;
    public static IAmazonSecurityTokenService? stsClient = null;
    public static ControlTowerWrapper? wrapper = null;
    private static string? ouArn;
    private static bool useLandingZone = false;

    /// <summary>
    /// Main entry point for the AWS Control Tower basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonControlTower>()
                .AddAWSService<IAmazonControlCatalog>()
                .AddAWSService<IAmazonOrganizations>()
                .AddAWSService<IAmazonSecurityTokenService>()
                .AddTransient<ControlTowerWrapper>()
            )
            .Build();
```

```
        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<ControlTowerBasics>();

        wrapper = host.Services.GetRequiredService<ControlTowerWrapper>();
        orgClient = host.Services.GetRequiredService<IAmazonOrganizations>();
        stsClient = host.Services.GetRequiredService<IAmazonSecurityTokenService>();

        await RunScenario();
    }

    /// <summary>
    /// Runs the example scenario.
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("\tWelcome to the AWS Control Tower with ControlCatalog
 example scenario.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("This demo will walk you through working with AWS Control
 Tower for landing zones,");
        Console.WriteLine("managing baselines, and working with controls.");

        try
        {
            var accountId = (await stsClient!.GetCallerIdentityAsync(new
 GetCallerIdentityRequest())).Account;
            Console.WriteLine($"\nAccount ID: {accountId}");

            Console.WriteLine("\nSome demo operations require the use of a landing
 zone.");
            Console.WriteLine("You can use an existing landing zone or opt out of
 these operations in the demo.");
            Console.WriteLine("For instructions on how to set up a landing zone,");
            Console.WriteLine("see https://docs.aws.amazon.com/controltower/latest/
userguide/getting-started-from-console.html");

            // List available landing zones
            var landingZones = await wrapper!.ListLandingZonesAsync();
            if (landingZones.Count > 0)
            {
                Console.WriteLine("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.Count; i++)
```

```
                {
                    Console.WriteLine($"{i + 1}. {landingZones[i].Arn}");
                }

                Console.Write($"\nDo you want to use the first landing zone in the
 list ({landingZones[0].Arn})? (y/n): ");
                if (GetUserConfirmation())
                {
                    useLandingZone = true;
                    Console.WriteLine($"Using landing zone: {landingZones[0].Arn}");
                    ouArn = await SetupOrganizationAsync();
                }
            }

            // Managing Baselines
            Console.WriteLine("\nManaging Baselines:");
            var baselines = await wrapper.ListBaselinesAsync();
            Console.WriteLine("\nListing available Baselines:");
            BaselineSummary? controlTowerBaseline = null;
            foreach (var baseline in baselines)
            {
                if (baseline.Name == "AWSControlTowerBaseline")
                    controlTowerBaseline = baseline;
                Console.WriteLine($"  - {baseline.Name}");
            }

            EnabledBaselineSummary? identityCenterBaseline = null;
            string? baselineArn = null;

            if (useLandingZone && ouArn != null)
            {
                Console.WriteLine("\nListing enabled baselines:");
                var enabledBaselines = await wrapper.ListEnabledBaselinesAsync();
                foreach (var baseline in enabledBaselines)
                {
                    if (baseline.BaselineIdentifier.Contains("baseline/
LN25R72TTG6IGPTQ"))
                        identityCenterBaseline = baseline;
                    Console.WriteLine($"  - {baseline.BaselineIdentifier}");
                }

                if (controlTowerBaseline != null)
                {
```

```
                    Console.Write("\nDo you want to enable the Control Tower
Baseline? (y/n): ");
                    if (GetUserConfirmation())
                    {
                        Console.WriteLine("\nEnabling Control Tower Baseline.");
                        var icBaselineArn = identityCenterBaseline?.Arn;
                        baselineArn = await wrapper.EnableBaselineAsync(ouArn,
                            controlTowerBaseline.Arn, "5.0", icBaselineArn ?? "");
                        var alreadyEnabled = false;
                        if (baselineArn != null)
                        {
                            Console.WriteLine($"Enabled baseline ARN:
{baselineArn}");
                        }
                        else
                        {
                            // Find the enabled baseline
                            foreach (var enabled in enabledBaselines)
                            {
                                if (enabled.BaselineIdentifier ==
controlTowerBaseline.Arn)
                                {
                                    baselineArn = enabled.Arn;
                                    alreadyEnabled = true;
                                    Console.WriteLine("No change, the selected
baseline was already enabled.");
                                    break;
                                }
                            }
                        }

                        if (baselineArn != null)
                        {
                            Console.Write("\nDo you want to reset the Control Tower
Baseline? (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine($"\nResetting Control Tower
Baseline: {baselineArn}");
                                var operationId = await
wrapper.ResetEnabledBaselineAsync(baselineArn);
                                Console.WriteLine($"Reset baseline operation id:
{operationId}");
                            }
```

```
                            Console.Write("\nDo you want to disable the Control
Tower Baseline? (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine($"Disabling baseline ARN:
{baselineArn}");
                                var operationId = await
wrapper.DisableBaselineAsync(baselineArn);
                                Console.WriteLine($"Disabled baseline operation id:
{operationId}");
                                if (alreadyEnabled)
                                {
                                    Console.WriteLine($"\nRe-enabling Control Tower
Baseline: {baselineArn}");
                                    // Re-enable the Control Tower baseline if it
was originally enabled.
                                    await wrapper.EnableBaselineAsync(ouArn,
                                        controlTowerBaseline.Arn, "5.0",
icBaselineArn ?? "");
                                }
                            }
                        }
                    }
                }
            }

            // Managing Controls
            Console.WriteLine("\nManaging Controls:");
            var controls = await wrapper.ListControlsAsync();
            Console.WriteLine("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.Min(5, controls.Count); i++)
            {
                Console.WriteLine($"{i + 1}. {controls[i].Name} -
{controls[i].Arn}");
            }

            if (useLandingZone && ouArn != null)
            {
                var enabledControls = await wrapper.ListEnabledControlsAsync(ouArn);
                Console.WriteLine("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.Count; i++)
                {
```

```
                    Console.WriteLine($"{i + 1}.
 {enabledControls[i].ControlIdentifier}");
                }

                // Find first non-enabled control
                var enabledControlArns = enabledControls.Select(c =>
 c.Arn).ToHashSet();
                var controlArn = controls.FirstOrDefault(c => !
enabledControlArns.Contains(c.Arn))?.Arn;

                if (controlArn != null)
                {
                    Console.Write($"\nDo you want to enable the control
 {controlArn}? (y/n): ");
                    if (GetUserConfirmation())
                    {
                        Console.WriteLine($"\nEnabling control: {controlArn}");
                        var operationId = await
 wrapper.EnableControlAsync(controlArn, ouArn);
                        if (operationId != null)
                        {
                            Console.WriteLine($"Enabled control with operation id:
 {operationId}");

                            Console.Write("\nDo you want to disable the control? (y/
n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine("\nDisabling the control...");
                                var disableOpId = await
 wrapper.DisableControlAsync(controlArn, ouArn);
                                Console.WriteLine($"Disable operation ID:
 {disableOpId}");
                            }
                        }
                    }
                }
            }

            Console.WriteLine("\nThis concludes the example scenario.");
            Console.WriteLine("Thanks for watching!");
            Console.WriteLine(new string('-', 88));
        }
        catch (Exception ex)
```

```
        {
            logger.LogError(ex, "An error occurred during the Control Tower
scenario.");
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }

    /// <summary>
    /// Sets up AWS Organizations and creates or finds a Sandbox OU.
    /// </summary>
    /// <returns>The ARN of the Sandbox organizational unit.</returns>
    private static async Task<string> SetupOrganizationAsync()
    {
        Console.WriteLine("\nChecking organization status...");

        try
        {
            var orgResponse = await orgClient!.DescribeOrganizationAsync(new
DescribeOrganizationRequest());
            var orgId = orgResponse.Organization.Id;
            Console.WriteLine($"Account is part of organization: {orgId}");
        }
        catch (AWSOrganizationsNotInUseException)
        {
            Console.WriteLine("No organization found. Creating a new
organization...");
            var createResponse = await orgClient!.CreateOrganizationAsync(new
CreateOrganizationRequest { FeatureSet = OrganizationFeatureSet.ALL });
            var orgId = createResponse.Organization.Id;
            Console.WriteLine($"Created new organization: {orgId}");
        }

        // Look for Sandbox OU
        var roots = await orgClient.ListRootsAsync(new ListRootsRequest());
        var rootId = roots.Roots[0].Id;

        Console.WriteLine("Checking for Sandbox OU...");
        var ous = await orgClient.ListOrganizationalUnitsForParentAsync(new
ListOrganizationalUnitsForParentRequest { ParentId = rootId });
        var sandboxOu = ous.OrganizationalUnits.FirstOrDefault(ou => ou.Name ==
"Sandbox");

        if (sandboxOu == null)
        {
```

```
            Console.WriteLine("Creating Sandbox OU...");
            var createOuResponse = await orgClient.CreateOrganizationalUnitAsync(new
 CreateOrganizationalUnitRequest { ParentId = rootId, Name = "Sandbox" });
            sandboxOu = createOuResponse.OrganizationalUnit;
            Console.WriteLine($"Created new Sandbox OU: {sandboxOu.Id}");
        }
        else
        {
            Console.WriteLine($"Found existing Sandbox OU: {sandboxOu.Id}");
        }

        return sandboxOu.Arn;
    }

    /// <summary>
    /// Gets user confirmation by waiting for input or returning true if not
 interactive.
    /// </summary>
    /// <returns>True if user enters 'y' or if isInteractive is false, otherwise
 false.</returns>
    private static bool GetUserConfirmation()
    {
        return Console.ReadLine()?.ToLower() == "y" || !isInteractive;
    }
}
```

Métodos de encapsulamento que são chamados pelo cenário para gerenciar as ações do Aurora.

```
using Amazon.ControlCatalog;
using Amazon.ControlCatalog.Model;
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using ValidationException = Amazon.ControlTower.Model.ValidationException;

namespace ControlTowerActions;

/// <summary>
/// Methods to perform AWS Control Tower actions.
/// </summary>
public class ControlTowerWrapper
```

```
{
    private readonly IAmazonControlTower _controlTowerService;
    private readonly IAmazonControlCatalog _controlCatalogService;

    /// <summary>
    /// Constructor for the wrapper class containing AWS Control Tower actions.
    /// </summary>
    /// <param name="controlTowerService">The AWS Control Tower client object.</
param>
    /// <param name="controlCatalogService">The AWS Control Catalog client object.</
param>
    public ControlTowerWrapper(IAmazonControlTower controlTowerService,
 IAmazonControlCatalog controlCatalogService)
    {
        _controlTowerService = controlTowerService;
        _controlCatalogService = controlCatalogService;
    }

    /// <summary>
    /// List the AWS Control Tower landing zones for an account.
    /// </summary>
    /// <returns>A list of LandingZoneSummary objects.</returns>
    public async Task<List<LandingZoneSummary>> ListLandingZonesAsync()
    {
        try
        {
            var landingZones = new List<LandingZoneSummary>();

            var landingZonesPaginator =
 _controlTowerService.Paginators.ListLandingZones(new ListLandingZonesRequest());

            await foreach (var response in landingZonesPaginator.Responses)
            {
                landingZones.AddRange(response.LandingZones);
            }

            return landingZones;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list landing zones. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
```

```
    }


    /// <summary>
    /// List all baselines.
    /// </summary>
    /// <returns>A list of baseline summaries.</returns>
    public async Task<List<BaselineSummary>> ListBaselinesAsync()
    {
        try
        {
            var baselines = new List<BaselineSummary>();

            var baselinesPaginator =
_controlTowerService.Paginators.ListBaselines(new ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            return baselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list baselines. Here's why: {ex.ErrorCode}:
{ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all enabled baselines.
    /// </summary>
    /// <returns>A list of enabled baseline summaries.</returns>
    public async Task<List<EnabledBaselineSummary>> ListEnabledBaselinesAsync()
    {
        try
        {
            var enabledBaselines = new List<EnabledBaselineSummary>();
```

```
            var enabledBaselinesPaginator =
 _controlTowerService.Paginators.ListEnabledBaselines(new
 ListEnabledBaselinesRequest());

            await foreach (var response in enabledBaselinesPaginator.Responses)
            {
                enabledBaselines.AddRange(response.EnabledBaselines);
            }

            return enabledBaselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled baselines. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Enable a baseline for the specified target.
    /// </summary>
    /// <param name="targetIdentifier">The ARN of the target.</param>
    /// <param name="baselineIdentifier">The identifier of baseline to enable.</
param>
    /// <param name="baselineVersion">The version of baseline to enable.</param>
    /// <param name="identityCenterBaseline">The identifier of identity center
 baseline if it is enabled.</param>
    /// <returns>The enabled baseline ARN or null.</returns>
    public async Task<string?> EnableBaselineAsync(string targetIdentifier, string
 baselineIdentifier, string baselineVersion, string identityCenterBaseline)
    {
        try
        {
            var parameters = new List<EnabledBaselineParameter>();
            if (!string.IsNullOrEmpty(identityCenterBaseline))
            {
                parameters.Add(
                    new EnabledBaselineParameter
                    {
                        Key = "IdentityCenterEnabledBaselineArn",
                        Value = identityCenterBaseline
                    });
```

```
            }
            var request = new EnableBaselineRequest
            {
                BaselineIdentifier = baselineIdentifier,
                BaselineVersion = baselineVersion,
                TargetIdentifier = targetIdentifier,
                Parameters = parameters
            };

            var response = await _controlTowerService.EnableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return response.Arn;
        }
        catch (ValidationException ex)
        {
            if (ex.Message.Contains("already enabled"))
                Console.WriteLine("Baseline is already enabled for this target");
            else { Console.WriteLine(ex.Message); }
            // Write the message and return null if baseline cannot be enabled.
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

```
    /// <summary>
    /// Disable a baseline for a specific target and wait for the operation to
complete.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the baseline to
disable.</param>
    /// <returns>The operation ID or null if there was a conflict.</returns>
    public async Task<string?> DisableBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new DisableBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await _controlTowerService.DisableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (ConflictException ex)
        {
            Console.WriteLine($"Conflict disabling baseline: {ex.Message}. Skipping
disable step.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
```

```
            Console.WriteLine($"Couldn't disable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Reset an enabled baseline for a specific target.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the enabled
baseline to reset.</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> ResetEnabledBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new ResetEnabledBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.ResetEnabledBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
```

```
            Console.WriteLine("Target not found, unable to reset enabled
baseline.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't reset enabled baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Get the status of a baseline operation.
    /// </summary>
    /// <param name="operationId">The ID of the baseline operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<BaselineOperationStatus> GetBaselineOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetBaselineOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetBaselineOperationAsync(request);
            return response.BaselineOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get baseline operation status. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

```
    /// <summary>
    /// List enabled controls for a target organizational unit.
    /// </summary>
    /// <param name="targetIdentifier">The target organizational unit identifier.</
param>
    /// <returns>A list of enabled control summaries.</returns>
    public async Task<List<EnabledControlSummary>> ListEnabledControlsAsync(string
 targetIdentifier)
    {
        try
        {
            var request = new ListEnabledControlsRequest
            {
                TargetIdentifier = targetIdentifier
            };

            var enabledControls = new List<EnabledControlSummary>();

            var enabledControlsPaginator =
 _controlTowerService.Paginators.ListEnabledControls(request);

            await foreach (var response in enabledControlsPaginator.Responses)
            {
                enabledControls.AddRange(response.EnabledControls);
            }

            return enabledControls;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
 (ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
 enabling controls.");
            return new List<EnabledControlSummary>();
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled controls. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

```
   /// <summary>
   /// Enable a control for a specified target.
   /// </summary>
   /// <param name="controlArn">The ARN of the control to enable.</param>
   /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
   /// <returns>The operation ID or null if already enabled.</returns>
   public async Task<string?> EnableControlAsync(string controlArn, string
targetIdentifier)
   {
       try
       {
           Console.WriteLine(controlArn);
           Console.WriteLine(targetIdentifier);

           var request = new EnableControlRequest
           {
               ControlIdentifier = controlArn,
               TargetIdentifier = targetIdentifier
           };

           var response = await _controlTowerService.EnableControlAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
           while (true)
           {
               var status = await GetControlOperationAsync(operationId);
               Console.WriteLine($"Control operation status: {status}");
               if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
               {
                   break;
               }
               await Task.Delay(30000); // Wait 30 seconds
           }

           return operationId;
       }
       catch (Amazon.ControlTower.Model.ValidationException ex) when
(ex.Message.Contains("already enabled"))
       {
```

```
            Console.WriteLine("Control is already enabled for this target");
            return null;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable control. Here's why: {ex.ErrorCode}:
{ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Disable a control for a specified target.
    /// </summary>
    /// <param name="controlArn">The ARN of the control to disable.</param>
    /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> DisableControlAsync(string controlArn, string
targetIdentifier)
    {
        try
        {
            var request = new DisableControlRequest
            {
                ControlIdentifier = controlArn,
                TargetIdentifier = targetIdentifier
            };

            var response = await _controlTowerService.DisableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
```

```
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Control not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Get the status of a control operation.
    /// </summary>
    /// <param name="operationId">The ID of the control operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<ControlOperationStatus> GetControlOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetControlOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetControlOperationAsync(request);
            return response.ControlOperation.Status;
        }
```

```
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get control operation status. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all controls in the Control Tower control catalog.
    /// </summary>
    /// <returns>A list of control summaries.</returns>
    public async Task<List<ControlSummary>> ListControlsAsync()
    {
        try
        {
            var controls = new List<ControlSummary>();

            var controlsPaginator =
 _controlCatalogService.Paginators.ListControls(new
 Amazon.ControlCatalog.Model.ListControlsRequest());

            await foreach (var response in controlsPaginator.Responses)
            {
                controls.AddRange(response.Controls);
            }

            return controls;
        }
        catch (AmazonControlCatalogException ex)
        {
            Console.WriteLine($"Couldn't list controls. Here's why: {ex.ErrorCode}:
 {ex.Message}");
            throw;
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Ações

`DisableBaseline`

O código de exemplo a seguir mostra como usar `DisableBaseline`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Disable a baseline for a specific target and wait for the operation to
complete.
```

```
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the baseline to
disable.</param>
    /// <returns>The operation ID or null if there was a conflict.</returns>
    public async Task<string?> DisableBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new DisableBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await _controlTowerService.DisableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (ConflictException ex)
        {
            Console.WriteLine($"Conflict disabling baseline: {ex.Message}. Skipping
disable step.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
```

```
    }
```

- Para obter detalhes da API, consulte DisableBaseline em Referência AWS SDK para .NET da API.

`DisableControl`

O código de exemplo a seguir mostra como usar `DisableControl`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Disable a control for a specified target.
   /// </summary>
   /// <param name="controlArn">The ARN of the control to disable.</param>
   /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
   /// <returns>The operation ID.</returns>
   public async Task<string> DisableControlAsync(string controlArn, string
targetIdentifier)
   {
       try
       {
           var request = new DisableControlRequest
           {
               ControlIdentifier = controlArn,
               TargetIdentifier = targetIdentifier
           };

           var response = await _controlTowerService.DisableControlAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
```

```
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Control not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DisableControl em Referência AWS SDK para .NET da API.

`EnableBaseline`

O código de exemplo a seguir mostra como usar `EnableBaseline`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Enable a baseline for the specified target.
    /// </summary>
    /// <param name="targetIdentifier">The ARN of the target.</param>
    /// <param name="baselineIdentifier">The identifier of baseline to enable.</
param>
    /// <param name="baselineVersion">The version of baseline to enable.</param>
    /// <param name="identityCenterBaseline">The identifier of identity center
 baseline if it is enabled.</param>
    /// <returns>The enabled baseline ARN or null.</returns>
    public async Task<string?> EnableBaselineAsync(string targetIdentifier, string
 baselineIdentifier, string baselineVersion, string identityCenterBaseline)
    {
        try
        {
            var parameters = new List<EnabledBaselineParameter>();
            if (!string.IsNullOrEmpty(identityCenterBaseline))
            {
                parameters.Add(
                    new EnabledBaselineParameter
                    {
                        Key = "IdentityCenterEnabledBaselineArn",
                        Value = identityCenterBaseline
                    });
            }
            var request = new EnableBaselineRequest
            {
                BaselineIdentifier = baselineIdentifier,
                BaselineVersion = baselineVersion,
                TargetIdentifier = targetIdentifier,
                Parameters = parameters
            };

            var response = await _controlTowerService.EnableBaselineAsync(request);
```

```
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return response.Arn;
        }
        catch (ValidationException ex)
        {
            if (ex.Message.Contains("already enabled"))
                Console.WriteLine("Baseline is already enabled for this target");
            else { Console.WriteLine(ex.Message); }
            // Write the message and return null if baseline cannot be enabled.
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte EnableBaseline em Referência AWS SDK para .NET da API.

EnableControl

O código de exemplo a seguir mostra como usar `EnableControl`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Enable a control for a specified target.
   /// </summary>
   /// <param name="controlArn">The ARN of the control to enable.</param>
   /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
   /// <returns>The operation ID or null if already enabled.</returns>
   public async Task<string?> EnableControlAsync(string controlArn, string
targetIdentifier)
   {
       try
       {
           Console.WriteLine(controlArn);
           Console.WriteLine(targetIdentifier);

           var request = new EnableControlRequest
           {
               ControlIdentifier = controlArn,
               TargetIdentifier = targetIdentifier
           };

           var response = await _controlTowerService.EnableControlAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
           while (true)
           {
               var status = await GetControlOperationAsync(operationId);
               Console.WriteLine($"Control operation status: {status}");
               if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
               {
                   break;
               }
```

```
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ValidationException ex) when
(ex.Message.Contains("already enabled"))
        {
            Console.WriteLine("Control is already enabled for this target");
            return null;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable control. Here's why: {ex.ErrorCode}:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte EnableControl em Referência AWS SDK para .NET da API.

`GetBaselineOperation`

O código de exemplo a seguir mostra como usar `GetBaselineOperation`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the status of a baseline operation.
    /// </summary>
    /// <param name="operationId">The ID of the baseline operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<BaselineOperationStatus> GetBaselineOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetBaselineOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetBaselineOperationAsync(request);
            return response.BaselineOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get baseline operation status. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte GetBaselineOperation em Referência AWS SDK para .NET da API.

`GetControlOperation`

O código de exemplo a seguir mostra como usar `GetControlOperation`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Get the status of a control operation.
   /// </summary>
   /// <param name="operationId">The ID of the control operation.</param>
   /// <returns>The operation status.</returns>
   public async Task<ControlOperationStatus> GetControlOperationAsync(string
operationId)
   {
       try
       {
           var request = new GetControlOperationRequest
           {
               OperationIdentifier = operationId
           };

           var response = await
_controlTowerService.GetControlOperationAsync(request);
           return response.ControlOperation.Status;
       }
       catch (Amazon.ControlTower.Model.ResourceNotFoundException)
       {
           Console.WriteLine("Operation not found.");
           throw;
       }
       catch (AmazonControlTowerException ex)
       {
           Console.WriteLine($"Couldn't get control operation status. Here's why:
{ex.ErrorCode}: {ex.Message}");
           throw;
       }
   }
```

- Para obter detalhes da API, consulte GetControlOperation em Referência AWS SDK para .NET da API.

ListBaselines

O código de exemplo a seguir mostra como usar `ListBaselines`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List all baselines.
    /// </summary>
    /// <returns>A list of baseline summaries.</returns>
    public async Task<List<BaselineSummary>> ListBaselinesAsync()
    {
        try
        {
            var baselines = new List<BaselineSummary>();

            var baselinesPaginator =
_controlTowerService.Paginators.ListBaselines(new ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            return baselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list baselines. Here's why: {ex.ErrorCode}:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListBaselines em Referência AWS SDK para .NET da API.

`ListEnabledBaselines`

O código de exemplo a seguir mostra como usar `ListEnabledBaselines`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List all enabled baselines.
    /// </summary>
    /// <returns>A list of enabled baseline summaries.</returns>
    public async Task<List<EnabledBaselineSummary>> ListEnabledBaselinesAsync()
    {
        try
        {
            var enabledBaselines = new List<EnabledBaselineSummary>();

            var enabledBaselinesPaginator =
_controlTowerService.Paginators.ListEnabledBaselines(new
ListEnabledBaselinesRequest());

            await foreach (var response in enabledBaselinesPaginator.Responses)
            {
                enabledBaselines.AddRange(response.EnabledBaselines);
            }

            return enabledBaselines;
        }
        catch (AmazonControlTowerException ex)
        {
```

```
            Console.WriteLine($"Couldn't list enabled baselines. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListEnabledBaselines em Referência AWS SDK para .NET da API.

`ListEnabledControls`

O código de exemplo a seguir mostra como usar `ListEnabledControls`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List enabled controls for a target organizational unit.
    /// </summary>
    /// <param name="targetIdentifier">The target organizational unit identifier.</
param>
    /// <returns>A list of enabled control summaries.</returns>
    public async Task<List<EnabledControlSummary>> ListEnabledControlsAsync(string
 targetIdentifier)
    {
        try
        {
            var request = new ListEnabledControlsRequest
            {
                TargetIdentifier = targetIdentifier
            };

            var enabledControls = new List<EnabledControlSummary>();
```

```
            var enabledControlsPaginator =
_controlTowerService.Paginators.ListEnabledControls(request);

            await foreach (var response in enabledControlsPaginator.Responses)
            {
                enabledControls.AddRange(response.EnabledControls);
            }

            return enabledControls;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return new List<EnabledControlSummary>();
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled controls. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListEnabledControls em Referência AWS SDK para .NET da API.

`ListLandingZones`

O código de exemplo a seguir mostra como usar `ListLandingZones`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List the AWS Control Tower landing zones for an account.
    /// </summary>
    /// <returns>A list of LandingZoneSummary objects.</returns>
    public async Task<List<LandingZoneSummary>> ListLandingZonesAsync()
    {
        try
        {
            var landingZones = new List<LandingZoneSummary>();

            var landingZonesPaginator =
_controlTowerService.Paginators.ListLandingZones(new ListLandingZonesRequest());

            await foreach (var response in landingZonesPaginator.Responses)
            {
                landingZones.AddRange(response.LandingZones);
            }

            return landingZones;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list landing zones. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListLandingZones em Referência AWS SDK para .NET da API.

`ResetEnabledBaseline`

O código de exemplo a seguir mostra como usar `ResetEnabledBaseline`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Reset an enabled baseline for a specific target.
   /// </summary>
   /// <param name="enabledBaselineIdentifier">The identifier of the enabled
baseline to reset.</param>
   /// <returns>The operation ID.</returns>
   public async Task<string> ResetEnabledBaselineAsync(string
enabledBaselineIdentifier)
   {
       try
       {
           var request = new ResetEnabledBaselineRequest
           {
               EnabledBaselineIdentifier = enabledBaselineIdentifier
           };

           var response = await
_controlTowerService.ResetEnabledBaselineAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
           while (true)
           {
               var status = await GetBaselineOperationAsync(operationId);
               Console.WriteLine($"Baseline operation status: {status}");
               if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
               {
                   break;
               }
               await Task.Delay(30000); // Wait 30 seconds
           }

           return operationId;
```

```
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Target not found, unable to reset enabled
baseline.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't reset enabled baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ResetEnabledBaseline em Referência AWS SDK para .NET da API.

Exemplos do DynamoDB usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
using Microsoft.Extensions.DependencyInjection;

namespace DynamoDBActions;

/// <summary>
/// A simple example that demonstrates basic DynamoDB operations.
/// </summary>
public class HelloDynamoDB
{
    /// <summary>
    /// HelloDynamoDB lists the existing DynamoDB tables for the default user.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon DynamoDB.
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonDynamoDB>()
            )
            .Build();

        // Now the client is available for injection.
```

```
        var dynamoDbClient = host.Services.GetRequiredService<IAmazonDynamoDB>();

        try
        {
            var request = new ListTablesRequest();
            var tableNames = new List<string>();

            var paginatorForTables = dynamoDbClient.Paginators.ListTables(request);

            await foreach (var tableName in paginatorForTables.TableNames)
            {
                tableNames.Add(tableName);
            }

            Console.WriteLine("Welcome to the DynamoDB Hello Service example. " +
                              "\nLet's list your DynamoDB tables:");
            tableNames.ForEach(table =>
            {
                Console.WriteLine($"Table: {table}");
            });
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB service error occurred while
 listing tables. {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing tables.
 {ex.Message}");
        }
    }
}
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// This example application performs the following basic Amazon DynamoDB
/// functions:
///     CreateTableAsync
///     PutItemAsync
///     UpdateItemAsync
///     BatchWriteItemAsync
///     GetItemAsync
///     DeleteItemAsync
///     Query
///     Scan
///     DeleteItemAsync.
/// </summary>
public class DynamoDbBasics
{
    public static bool IsInteractive = true;

    // Separator for the console display.
    private static readonly string SepBar = new string('-', 80);

    /// <summary>
    /// The main entry point for the DynamoDB Basics example application.
    /// </summary>
```

```
    /// <param name="args">Command line arguments.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon DynamoDB.
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonDynamoDB>()
                    .AddTransient<DynamoDbWrapper>())
            .Build();

        // Now the wrapper is available for injection.
        var dynamoDbWrapper = host.Services.GetRequiredService<DynamoDbWrapper>();

        var tableName = "movie_table";

        var movieFileName = @"movies.json";

        DisplayInstructions();

        // Create a new table and wait for it to be active.
        Console.WriteLine($"Creating the new table: {tableName}");

        var success = await dynamoDbWrapper.CreateMovieTableAsync(tableName);

        Console.WriteLine(success
            ? $"\nTable: {tableName} successfully created."
            : $"\nCould not create {tableName}.");

        WaitForEnter();

        // Add a single new movie to the table.
        var newMovie = new Movie
        {
            Year = 2021,
            Title = "Spider-Man: No Way Home",
        };

        success = await dynamoDbWrapper.PutItemAsync(newMovie, tableName);
        if (success)
        {
            Console.WriteLine($"Added {newMovie.Title} to the table.");
        }
```

```
        else
        {
            Console.WriteLine("Could not add movie to table.");
        }

        WaitForEnter();

        // Update the new movie by adding a plot and rank.
        var newInfo = new MovieInfo
        {
            Plot = "With Spider-Man's identity now revealed, Peter asks" +
                   "Doctor Strange for help. When a spell goes wrong, dangerous" +
                   "foes from other worlds start to appear, forcing Peter to" +
                   "discover what it truly means to be Spider-Man.",
            Rank = 9,
        };

        success = await dynamoDbWrapper.UpdateItemAsync(newMovie, newInfo,
tableName);
        if (success)
        {
            Console.WriteLine($"Successfully updated the movie: {newMovie.Title}");
        }
        else
        {
            Console.WriteLine("Could not update the movie.");
        }

        WaitForEnter();

        // Add a batch of movies to the DynamoDB table from a list of
        // movies in a JSON file.
        var itemCount = await dynamoDbWrapper.BatchWriteItemsAsync(movieFileName,
tableName);
        Console.WriteLine($"Added {itemCount} movies to the table.");

        WaitForEnter();

        // Get a movie by key. (partition + sort)
        var lookupMovie = new Movie
        {
            Title = "Jurassic Park",
            Year = 1993,
        };
```

```
        Console.WriteLine("Looking for the movie \"Jurassic Park\".");
        var item = await dynamoDbWrapper.GetItemAsync(lookupMovie, tableName);
        if (item?.Count > 0)
        {
            dynamoDbWrapper.DisplayItem(item);
        }
        else
        {
            Console.WriteLine($"Couldn't find {lookupMovie.Title}");
        }

        WaitForEnter();

        // Delete a movie.
        var movieToDelete = new Movie
        {
            Title = "The Town",
            Year = 2010,
        };

        success = await dynamoDbWrapper.DeleteItemAsync(tableName, movieToDelete);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {movieToDelete.Title}.");
        }
        else
        {
            Console.WriteLine($"Could not delete {movieToDelete.Title}.");
        }

        WaitForEnter();

        // Use Query to find all the movies released in 2010.
        int findYear = 2010;
        Console.WriteLine($"Movies released in {findYear}");
        var queryCount = await dynamoDbWrapper.QueryMoviesAsync(tableName,
findYear);
        Console.WriteLine($"Found {queryCount} movies released in {findYear}");

        WaitForEnter();

        // Use Scan to get a list of movies from 2001 to 2011.
```

```
        int startYear = 2001;
        int endYear = 2011;
        var scanCount = await dynamoDbWrapper.ScanTableAsync(tableName, startYear,
endYear);
        Console.WriteLine($"Found {scanCount} movies released between {startYear}
and {endYear}");

        WaitForEnter();

        // Delete the table.
        success = await dynamoDbWrapper.DeleteTableAsync(tableName);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {tableName}");
        }
        else
        {
            Console.WriteLine($"Could not delete {tableName}");
        }

        Console.WriteLine("The DynamoDB Basics example application is complete.");

        WaitForEnter();
    }

    /// <summary>
    /// Displays the description of the application on the console.
    /// </summary>
    private static void DisplayInstructions()
    {
        if (!IsInteractive)
        {
            return;
        }

        Console.Clear();
        Console.WriteLine();
        Console.Write(new string(' ', 28));
        Console.WriteLine("DynamoDB Basics Example");
        Console.WriteLine(SepBar);
        Console.WriteLine("This demo application shows the basics of using DynamoDB
with the AWS SDK.");
        Console.WriteLine(SepBar);
```

```
        Console.WriteLine("The application does the following:");
        Console.WriteLine("\t1. Creates a table with partition: year and
 sort:title.");
        Console.WriteLine("\t2. Adds a single movie to the table.");
        Console.WriteLine("\t3. Adds movies to the table from moviedata.json.");
        Console.WriteLine("\t4. Updates the rating and plot of the movie that was
 just added.");
        Console.WriteLine("\t5. Gets a movie using its key (partition + sort).");
        Console.WriteLine("\t6. Deletes a movie.");
        Console.WriteLine("\t7. Uses QueryAsync to return all movies released in a
 given year.");
        Console.WriteLine("\t8. Uses ScanAsync to return all movies released within
 a range of years.");
        Console.WriteLine("\t9. Finally, it deletes the table that was just
 created.");
        WaitForEnter();
    }

    /// <summary>
    /// Simple method to wait for the Enter key to be pressed.
    /// </summary>
    private static void WaitForEnter()
    {
        if (IsInteractive)
        {
            Console.WriteLine("\nPress <Enter> to continue.");
            Console.WriteLine(SepBar);
            _ = Console.ReadLine();
        }
    }
}
```

Usar o cliente injetado para operações de tabela.

```
using System.Text.Json;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DataModel;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.DynamoDBv2.Model;

namespace DynamoDBActions;
```

```
/// <summary>
/// Methods of this class perform Amazon DynamoDB operations.
/// </summary>
public class DynamoDbWrapper
{
    private readonly IAmazonDynamoDB _amazonDynamoDB;

    /// <summary>
    /// Constructor for the DynamoDbWrapper class.
    /// </summary>
    /// <param name="amazonDynamoDB">The injected DynamoDB client.</param>
    public DynamoDbWrapper(IAmazonDynamoDB amazonDynamoDB)
    {
        _amazonDynamoDB = amazonDynamoDB;
    }
```

Cria uma tabela para conter dados de filmes.

```
    /// <summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    /// </summary>
    /// <param name="tableName">The name of the table to create.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> CreateMovieTableAsync(string tableName)
    {
        try
        {
            var response = await _amazonDynamoDB.CreateTableAsync(new
 CreateTableRequest
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition
                    {
                        AttributeName = "title",
                        AttributeType = ScalarAttributeType.S,
                    },
                    new AttributeDefinition
```

```
                    {
                        AttributeName = "year",
                        AttributeType = ScalarAttributeType.N,
                    },
                },
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement
                    {
                        AttributeName = "year",
                        KeyType = KeyType.HASH,
                    },
                    new KeySchemaElement
                    {
                        AttributeName = "title",
                        KeyType = KeyType.RANGE,
                    },
                },
                BillingMode = BillingMode.PAY_PER_REQUEST,
            });

            // Wait until the table is ACTIVE and then report success.
            Console.Write("Waiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = response.TableDescription.TableName,
            };

            TableStatus status;

            int sleepDuration = 2000;

            do
            {
                Thread.Sleep(sleepDuration);

                var describeTableResponse = await
_amazonDynamoDB.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");
```

```
            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException ex)
        {
            Console.WriteLine($"Table {tableName} already exists. {ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while creating
 table {tableName}. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating table {tableName}.
 {ex.Message}");
            throw;
        }
    }
```

Adiciona um único filme à tabela.

```
    /// <summary>
    /// Adds a new item to the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing informtation for
    /// the movie to add to the table.</param>
    /// <param name="tableName">The name of the table where the item will be
 added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
returns>
    public async Task<bool> PutItemAsync(Movie newMovie, string tableName)
    {
        try
        {
            var item = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
```

```
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = item,
            };

            await _amazonDynamoDB.PutItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while putting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while putting item.
{ex.Message}");
            throw;
        }
    }
```

Atualiza um item único em uma tabela.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
```

```
    /// <param name="tableName">The name of the table that contains the movie.</
param>
    /// <returns>A Boolean value that indicates the success of the operation.</
returns>
    public async Task<bool> UpdateItemAsync(
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
            var updates = new Dictionary<string, AttributeValueUpdate>
            {
                ["info.plot"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { S = newInfo.Plot },
                },

                ["info.rating"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { N = newInfo.Rank.ToString() },
                },
            };

            var request = new UpdateItemRequest
            {
                AttributeUpdates = updates,
                Key = key,
                TableName = tableName,
            };

            await _amazonDynamoDB.UpdateItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
```

```
            Console.WriteLine($"Table {tableName} or item was not found.
{ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while updating
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while updating item.
{ex.Message}");
            throw;
        }
    }
```

Recupera um único item de uma tabela de filmes.

```
    /// <summary>
    /// Gets information about an existing movie from the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information about
    /// the movie to retrieve.</param>
    /// <param name="tableName">The name of the table containing the movie.</param>
    /// <returns>A Dictionary object containing information about the item
    /// retrieved.</returns>
    public async Task<Dictionary<string, AttributeValue>> GetItemAsync(Movie
newMovie, string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new GetItemRequest
```

```
            {
                Key = key,
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.GetItemAsync(request);
            return response.Item;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return new Dictionary<string, AttributeValue>();
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while getting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while getting item.
{ex.Message}");
            throw;
        }
    }
```

Grava um lote de itens na tabela de filmes.

```
    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the DynamoDB table.
    /// </summary>
    /// <param name="movieFileName">The name of the JSON file.</param>
    /// <returns>A generic list of movie objects.</returns>
    public List<Movie> ImportMovies(string movieFileName)
    {
        var moviesList = new List<Movie>();
        if (!File.Exists(movieFileName))
        {
```

```
            return moviesList;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();
        var allMovies = JsonSerializer.Deserialize<List<Movie>>(
            json,
            new JsonSerializerOptions
            {
                PropertyNameCaseInsensitive = true
            });

        // Now return the first 250 entries.
        if (allMovies != null && allMovies.Any())
        {
            moviesList = allMovies.GetRange(0, 250);
        }
        return moviesList;
    }

    /// <summary>
    /// Writes 250 items to the movie table.
    /// </summary>
    /// <param name="movieFileName">A string containing the full path to
    /// the JSON file containing movie data.</param>
    /// <param name="tableName">The name of the table to write items to.</param>
    /// <returns>A long integer value representing the number of movies
    /// imported from the JSON file.</returns>
    public async Task<long> BatchWriteItemsAsync(
        string movieFileName, string tableName)
    {
        try
        {
            var movies = ImportMovies(movieFileName);
            if (!movies.Any())
            {
                Console.WriteLine("Couldn't find the JSON file with movie data.");
                return 0;
            }

            var context = new DynamoDBContextBuilder()
                // Optional call to provide a specific instance of IAmazonDynamoDB
                .WithDynamoDBClient(() => _amazonDynamoDB)
                .Build();
```

```
            var movieBatch = context.CreateBatchWrite<Movie>(
                new BatchWriteConfig()
                {
                    OverrideTableName = tableName
                });
            movieBatch.AddPutItems(movies);

            Console.WriteLine("Adding imported movies to the table.");
            await movieBatch.ExecuteAsync();

            return movies.Count;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table was not found during batch write operation.
{ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred during batch write
operation. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during batch write operation.
{ex.Message}");
            throw;
        }
    }
```

Exclui um único item da tabela.

```
    /// <summary>
    /// Deletes a single item from a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table from which the item
    /// will be deleted.</param>
```

```
    /// <param name="movieToDelete">A movie object containing the title and
    /// year of the movie to delete.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// delete operation.</returns>
    public async Task<bool> DeleteItemAsync(
        string tableName,
        Movie movieToDelete)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = movieToDelete.Title },
                ["year"] = new AttributeValue { N = movieToDelete.Year.ToString() },
            };

            var request = new DeleteItemRequest { TableName = tableName, Key =
key, };

            await _amazonDynamoDB.DeleteItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting item.
{ex.Message}");
            throw;
        }
    }
```

Consulta a tabela de filmes lançados em determinado ano.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public async Task<int> QueryMoviesAsync(string tableName, int year)
{
    try
    {
        var movieTable = new TableBuilder(_amazonDynamoDB, tableName)
            .AddHashKey("year", DynamoDBEntryType.Numeric)
            .AddRangeKey("title", DynamoDBEntryType.String)
            .Build();

        var filter = new QueryFilter("year", QueryOperator.Equal, year);

        Console.WriteLine("\nFind movies released in: {year}:");

        var config = new QueryOperationConfig()
        {
            Limit = 10, // 10 items per page.
            Select = SelectValues.SpecificAttributes,
            AttributesToGet = new List<string>
            {
                "title",
                "year",
            },
            ConsistentRead = true,
            Filter = filter,
        };

        // Value used to track how many movies match the
        // supplied criteria.
        var moviesFound = 0;

        var search = movieTable.Query(config);
        do
        {
```

```
                var movieList = await search.GetNextSetAsync();
                moviesFound += movieList.Count;

                foreach (var movie in movieList)
                {
                    DisplayDocument(movie);
                }
            }
            while (!search.IsDone);

            return moviesFound;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while querying
movies. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while querying movies.
{ex.Message}");
            throw;
        }
    }
```

Busca na tabela os filmes lançados em um intervalo de anos.

```
    /// <summary>
    /// Scans the table for movies released between the specified years.
    /// </summary>
    /// <param name="tableName">The name of the table to scan.</param>
    /// <param name="startYear">The starting year for the range.</param>
    /// <param name="endYear">The ending year for the range.</param>
    /// <returns>The number of movies found in the specified year range.</returns>
    public async Task<int> ScanTableAsync(
```

```
        string tableName,
        int startYear,
        int endYear)
    {
        try
        {
            var request = new ScanRequest
            {
                TableName = tableName,
                ExpressionAttributeNames = new Dictionary<string, string>
                {
                    { "#yr", "year" },
                },
                ExpressionAttributeValues = new Dictionary<string, AttributeValue>
                {
                    { ":y_a", new AttributeValue { N = startYear.ToString() } },
                    { ":y_z", new AttributeValue { N = endYear.ToString() } },
                },
                FilterExpression = "#yr between :y_a and :y_z",
                ProjectionExpression = "#yr, title, info.actors[0], info.directors,
info.running_time_secs",
                Limit = 10 // Set a limit to demonstrate using the LastEvaluatedKey.
            };

            // Keep track of how many movies were found.
            int foundCount = 0;

            var response = new ScanResponse();
            do
            {
                response = await _amazonDynamoDB.ScanAsync(request);
                foundCount += response.Items.Count;
                response.Items.ForEach(i => DisplayItem(i));
                request.ExclusiveStartKey = response.LastEvaluatedKey;
            }
            while (response?.LastEvaluatedKey?.Count > 0);
            return foundCount;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
```

```
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while scanning
table. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while scanning table.
{ex.Message}");
            throw;
        }
    }
```

Exclui a tabela de filmes.

```
    /// <summary>
    /// Deletes a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> DeleteTableAsync(string tableName)
    {
        try
        {
            var request = new DeleteTableRequest
            {
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.DeleteTableAsync(request);

            Console.WriteLine($"Table {response.TableDescription.TableName}
successfully deleted.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found and cannot be
deleted. {ex.Message}");
            return false;
```

```
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
table {tableName}. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting table {tableName}.
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Grava um lote de itens na tabela de filmes.

```
/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The name of the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public List<Movie> ImportMovies(string movieFileName)
{
    var moviesList = new List<Movie>();
    if (!File.Exists(movieFileName))
    {
        return moviesList;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonSerializer.Deserialize<List<Movie>>(
        json,
        new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        });

    // Now return the first 250 entries.
    if (allMovies != null && allMovies.Any())
    {
        moviesList = allMovies.GetRange(0, 250);
    }
    return moviesList;
}

/// <summary>
```

```
    /// Writes 250 items to the movie table.
    /// </summary>
    /// <param name="movieFileName">A string containing the full path to
    /// the JSON file containing movie data.</param>
    /// <param name="tableName">The name of the table to write items to.</param>
    /// <returns>A long integer value representing the number of movies
    /// imported from the JSON file.</returns>
    public async Task<long> BatchWriteItemsAsync(
        string movieFileName, string tableName)
    {
        try
        {
            var movies = ImportMovies(movieFileName);
            if (!movies.Any())
            {
                Console.WriteLine("Couldn't find the JSON file with movie data.");
                return 0;
            }

            var context = new DynamoDBContextBuilder()
                // Optional call to provide a specific instance of IAmazonDynamoDB
                .WithDynamoDBClient(() => _amazonDynamoDB)
                .Build();

            var movieBatch = context.CreateBatchWrite<Movie>(
                new BatchWriteConfig()
                {
                    OverrideTableName = tableName
                });
            movieBatch.AddPutItems(movies);

            Console.WriteLine("Adding imported movies to the table.");
            await movieBatch.ExecuteAsync();

            return movies.Count;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table was not found during batch write operation.
{ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
```

```
            Console.WriteLine($"An Amazon DynamoDB error occurred during batch write
operation. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during batch write operation.
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte BatchWriteItem em Referência AWS SDK para .NET da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    /// </summary>
    /// <param name="tableName">The name of the table to create.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> CreateMovieTableAsync(string tableName)
    {
        try
        {
```

```
            var response = await _amazonDynamoDB.CreateTableAsync(new
CreateTableRequest
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition
                    {
                        AttributeName = "title",
                        AttributeType = ScalarAttributeType.S,
                    },
                    new AttributeDefinition
                    {
                        AttributeName = "year",
                        AttributeType = ScalarAttributeType.N,
                    },
                },
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement
                    {
                        AttributeName = "year",
                        KeyType = KeyType.HASH,
                    },
                    new KeySchemaElement
                    {
                        AttributeName = "title",
                        KeyType = KeyType.RANGE,
                    },
                },
                BillingMode = BillingMode.PAY_PER_REQUEST,
            });

            // Wait until the table is ACTIVE and then report success.
            Console.Write("Waiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = response.TableDescription.TableName,
            };

            TableStatus status;

            int sleepDuration = 2000;
```

```
            do
            {
                Thread.Sleep(sleepDuration);

                var describeTableResponse = await
_amazonDynamoDB.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException ex)
        {
            Console.WriteLine($"Table {tableName} already exists. {ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while creating
table {tableName}. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating table {tableName}.
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para .NET da API.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes a single item from a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table from which the item
    /// will be deleted.</param>
    /// <param name="movieToDelete">A movie object containing the title and
    /// year of the movie to delete.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// delete operation.</returns>
    public async Task<bool> DeleteItemAsync(
        string tableName,
        Movie movieToDelete)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = movieToDelete.Title },
                ["year"] = new AttributeValue { N = movieToDelete.Year.ToString() },
            };

            var request = new DeleteItemRequest { TableName = tableName, Key =
key, };

            await _amazonDynamoDB.DeleteItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
```

```
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting item.
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DeleteItem em Referência AWS SDK para .NET da API.

DeleteTable

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> DeleteTableAsync(string tableName)
    {
        try
        {
            var request = new DeleteTableRequest
            {
                TableName = tableName,
```

```
            };

            var response = await _amazonDynamoDB.DeleteTableAsync(request);

            Console.WriteLine($"Table {response.TableDescription.TableName}
successfully deleted.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found and cannot be
deleted. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
table {tableName}. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting table {tableName}.
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para .NET da API.

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets information about an existing movie from the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information about
    /// the movie to retrieve.</param>
    /// <param name="tableName">The name of the table containing the movie.</param>
    /// <returns>A Dictionary object containing information about the item
    /// retrieved.</returns>
    public async Task<Dictionary<string, AttributeValue>> GetItemAsync(Movie
newMovie, string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new GetItemRequest
            {
                Key = key,
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.GetItemAsync(request);
            return response.Item;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return new Dictionary<string, AttributeValue>();
        }
```

```
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while getting
 item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while getting item.
 {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte GetItem em Referência AWS SDK para .NET da API.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Adds a new item to the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing informtation for
    /// the movie to add to the table.</param>
    /// <param name="tableName">The name of the table where the item will be
 added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
returns>
    public async Task<bool> PutItemAsync(Movie newMovie, string tableName)
    {
```

```
        try
        {
            var item = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = item,
            };

            await _amazonDynamoDB.PutItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while putting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while putting item.
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK para .NET da API.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public async Task<int> QueryMoviesAsync(string tableName, int year)
{
    try
    {
        var movieTable = new TableBuilder(_amazonDynamoDB, tableName)
            .AddHashKey("year", DynamoDBEntryType.Numeric)
            .AddRangeKey("title", DynamoDBEntryType.String)
            .Build();

        var filter = new QueryFilter("year", QueryOperator.Equal, year);

        Console.WriteLine("\nFind movies released in: {year}:");

        var config = new QueryOperationConfig()
        {
            Limit = 10, // 10 items per page.
            Select = SelectValues.SpecificAttributes,
            AttributesToGet = new List<string>
            {
                "title",
                "year",
            },
            ConsistentRead = true,
            Filter = filter,
        };
```

```
            // Value used to track how many movies match the
            // supplied criteria.
            var moviesFound = 0;

            var search = movieTable.Query(config);
            do
            {
                var movieList = await search.GetNextSetAsync();
                moviesFound += movieList.Count;

                foreach (var movie in movieList)
                {
                    DisplayDocument(movie);
                }
            }
            while (!search.IsDone);

            return moviesFound;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while querying
movies. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while querying movies.
{ex.Message}");
            throw;
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para .NET .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Scans the table for movies released between the specified years.
    /// </summary>
    /// <param name="tableName">The name of the table to scan.</param>
    /// <param name="startYear">The starting year for the range.</param>
    /// <param name="endYear">The ending year for the range.</param>
    /// <returns>The number of movies found in the specified year range.</returns>
    public async Task<int> ScanTableAsync(
        string tableName,
        int startYear,
        int endYear)
    {
        try
        {
            var request = new ScanRequest
            {
                TableName = tableName,
                ExpressionAttributeNames = new Dictionary<string, string>
                {
                    { "#yr", "year" },
                },
                ExpressionAttributeValues = new Dictionary<string, AttributeValue>
                {
                    { ":y_a", new AttributeValue { N = startYear.ToString() } },
                    { ":y_z", new AttributeValue { N = endYear.ToString() } },
                },
                FilterExpression = "#yr between :y_a and :y_z",
                ProjectionExpression = "#yr, title, info.actors[0], info.directors,
info.running_time_secs",
                Limit = 10 // Set a limit to demonstrate using the LastEvaluatedKey.
            };
```

```
            // Keep track of how many movies were found.
            int foundCount = 0;

            var response = new ScanResponse();
            do
            {
                response = await _amazonDynamoDB.ScanAsync(request);
                foundCount += response.Items.Count;
                response.Items.ForEach(i => DisplayItem(i));
                request.ExclusiveStartKey = response.LastEvaluatedKey;
            }
            while (response?.LastEvaluatedKey?.Count > 0);
            return foundCount;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while scanning
table. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while scanning table.
{ex.Message}");
            throw;
        }
    }
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para .NET .

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
    /// <param name="tableName">The name of the table that contains the movie.</
param>
    /// <returns>A Boolean value that indicates the success of the operation.</
returns>
    public async Task<bool> UpdateItemAsync(
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
            var updates = new Dictionary<string, AttributeValueUpdate>
            {
                ["info.plot"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { S = newInfo.Plot },
                },

                ["info.rating"] = new AttributeValueUpdate
                {
```

```
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { N = newInfo.Rank.ToString() },
                },
            };

            var request = new UpdateItemRequest
            {
                AttributeUpdates = updates,
                Key = key,
                TableName = tableName,
            };

            await _amazonDynamoDB.UpdateItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} or item was not found.
{ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while updating
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while updating item.
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para .NET da API.

Exemplos do Amazon EC2 usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o Amazon EC2.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace EC2Actions;

public class HelloEc2
{
    /// <summary>
    /// HelloEc2 lists the existing security groups for the default users.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Elastic Compute Cloud (Amazon
 EC2).
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                .AddTransient<EC2Wrapper>()
            )
            .Build();

        // Now the client is available for injection.
        var ec2Client = host.Services.GetRequiredService<IAmazonEC2>();

        try
        {
            // Retrieve information for up to 10 Amazon EC2 security groups.
            var request = new DescribeSecurityGroupsRequest { MaxResults = 10 };
            var securityGroups = new List<SecurityGroup>();

            var paginatorForSecurityGroups =
                ec2Client.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            // Now print the security groups returned by the call to
            // DescribeSecurityGroupsAsync.
            Console.WriteLine("Welcome to the EC2 Hello Service example. " +
                              "\nLet's list your Security Groups:");
            securityGroups.ForEach(group =>
            {
                Console.WriteLine(
                    $"Security group: {group.GroupName} ID: {group.GroupId}");
            });
        }
        catch (AmazonEC2Exception ex)
        {
            Console.WriteLine($"An Amazon EC2 service error occurred while listing
security groups. {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing security groups.
{ex.Message}");
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK para .NET da API.

Exemplos do Amazon ECS usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o Amazon ECS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Olá, Amazon ECS

O exemplo de código a seguir mostra como começar a usar o Amazon ECS.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
using Amazon.ECS;
using Amazon.ECS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
```

```
namespace ECSActions;

/// <summary>
/// A class that introduces the Amazon ECS Client by listing the
/// cluster ARNs for the account.
/// </summary>
public class HelloECS
{
    static async System.Threading.Tasks.Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon ECS client.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonECS>()
            )
            .Build();

        var amazonECSClient = host.Services.GetRequiredService<IAmazonECS>();

        Console.WriteLine($"Hello Amazon ECS! Following are some cluster ARNS
 available in the your account");
        Console.WriteLine();

        var clusters = new List<string>();

        var clustersPaginator = amazonECSClient.Paginators.ListClusters(new
 ListClustersRequest());

        await foreach (var response in clustersPaginator.Responses)
        {
            clusters.AddRange(response.ClusterArns);
        }

        if (clusters.Count > 0)
        {
            clusters.ForEach(cluster =>
            {
```

```
                Console.WriteLine($"\tARN: {cluster}");
                Console.WriteLine($"Cluster Name: {cluster.Split("/").Last()}");
                Console.WriteLine();
            });
        }
        else
        {
            Console.WriteLine("No clusters were found.");
        }

    }
}
```

- Para obter detalhes da API, consulte ListClusters em Referência AWS SDK para .NET da API.

AWS IoT exemplos usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com AWS IoT.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Hello AWS IoT example.
/// </summary>
public class HelloIoT
{
    /// <summary>
    /// Main method to run the Hello IoT example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var iotClient = new AmazonIoTClient();

        try
        {
            Console.WriteLine("Hello AWS IoT! Let's list your IoT Things:");
            Console.WriteLine(new string('-', 80));

            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await iotClient.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;
```

```
            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the next
page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await iotClient.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            if (things is { Count: > 0 })
            {
                Console.WriteLine($"Found {things.Count} IoT Things:");
                foreach (var thing in things)
                {
                    Console.WriteLine($"- Thing Name: {thing.ThingName}");
                    Console.WriteLine($"  Thing ARN: {thing.ThingArn}");
                    Console.WriteLine($"  Thing Type: {thing.ThingTypeName ?? "No
type specified"}");
                    Console.WriteLine($"  Version: {thing.Version}");

                    if (thing.Attributes?.Count > 0)
                    {
                        Console.WriteLine("  Attributes:");
                        foreach (var attr in thing.Attributes)
                        {
                            Console.WriteLine($"    {attr.Key}: {attr.Value}");
                        }
                    }
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No IoT Things found in your account.");
                Console.WriteLine("You can create IoT Things using the IoT Basics
scenario example.");
```

```
            }

            Console.WriteLine("Hello IoT completed successfully.");
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            Console.WriteLine($"Request throttled, please try again later:
 {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list Things. Here's why: {ex.Message}");
        }
    }
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK para .NET .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie uma AWS IoT coisa.
- Gerar um certificado de dispositivo.
- Atualize AWS IoT qualquer coisa com atributos.
- Exibir um endpoint exclusivo.
- Liste seus AWS IoT certificados.
- Atualize uma AWS IoT sombra.
- Gravar informações do estado.
- Cria uma regra.
- Listar suas regras.
- Pesquisar coisas usando o nome da coisa.
- Exclua qualquer AWS IoT coisa.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS IoT recursos.

```
/// <summary>
/// Scenario class for AWS IoT basics.
/// </summary>
public class IoTBasics
{
    public static bool IsInteractive = true;
    public static IoTWrapper? Wrapper = null;
    public static IAmazonCloudFormation? CloudFormationClient = null;
    public static ILogger<IoTBasics> logger = null!;
    private static IoTWrapper _iotWrapper = null!;
    private static IAmazonCloudFormation _amazonCloudFormation = null!;
    private static ILogger<IoTBasics> _logger = null!;

    private static string _stackName = "IoTBasicsStack";
    private static string _stackResourcePath = "../../../../../../scenarios/basics/
iot/iot_usecase/resources/cfn_template.yaml";

    /// <summary>
    /// Main method for the IoT Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIoT>(new AWSOptions() { Region =
 RegionEndpoint.USEast1 })
                    .AddAWSService<IAmazonCloudFormation>()
                        .AddTransient<IoTWrapper>()
                        .AddLogging(builder => builder.AddConsole())
                        .AddSingleton<IAmazonIotData>(sp =>
```

```
                        {
                            var iotService = sp.GetRequiredService<IAmazonIoT>();
                            var request = new DescribeEndpointRequest
                            {
                                EndpointType = "iot:Data-ATS"
                            };
                            var response =
 iotService.DescribeEndpointAsync(request).Result;
                            return new AmazonIotDataClient($"https://
{response.EndpointAddress}/");
                        })
            )
            .Build();

        logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<IoTBasics>();

        Wrapper = host.Services.GetRequiredService<IoTWrapper>();
        CloudFormationClient =
 host.Services.GetRequiredService<IAmazonCloudFormation>();

        // Set the private fields for backwards compatibility
        _logger = logger;
        _iotWrapper = Wrapper;
        _amazonCloudFormation = CloudFormationClient;

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS IoT example scenario.");
        Console.WriteLine("This example program demonstrates various interactions
 with the AWS Internet of Things (IoT) Core service.");
        Console.WriteLine();
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }
        Console.WriteLine(new string('-', 80));

        try
        {
            await RunScenarioAsync();
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, "There was a problem running the scenario.");
            Console.WriteLine($"\nAn error occurred: {ex.Message}");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The AWS IoT scenario has successfully completed.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Run the IoT Basics scenario.
    /// </summary>
    /// <returns>A Task object.</returns>
    public static async Task RunScenarioAsync()
    {
        // Use static properties if available, otherwise use private fields
        var iotWrapper = Wrapper ?? _iotWrapper;
        var cloudFormationClient = CloudFormationClient ?? _amazonCloudFormation;
        var scenarioLogger = logger ?? _logger;

        await RunScenarioInternalAsync(iotWrapper, cloudFormationClient,
 scenarioLogger);
    }

    /// <summary>
    /// Internal method to run the IoT Basics scenario with injected dependencies.
    /// </summary>
    /// <param name="iotWrapper">The IoT wrapper instance.</param>
    /// <param name="cloudFormationClient">The CloudFormation client instance.</
param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    private static async Task RunScenarioInternalAsync(IoTWrapper iotWrapper,
 IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        string thingName = $"iot-thing-{Guid.NewGuid():N}";
        string certificateArn = "";
        string certificateId = "";
        string ruleName = $"iotruledefault";
        string snsTopicArn = "";

        try
        {
            // Step 1: Create an AWS IoT Thing
```

```
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("1. Create an AWS IoT Thing.");
            Console.WriteLine("An AWS IoT Thing represents a virtual entity in the
AWS IoT service that can be associated with a physical device.");
            Console.WriteLine();

            if (IsInteractive)
            {
                Console.Write("Enter Thing name: ");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    thingName = userInput;
            }
            else
            {
                Console.WriteLine($"Using default Thing name: {thingName}");
            }

            var thingArn = await iotWrapper.CreateThingAsync(thingName);
            Console.WriteLine($"{thingName} was successfully created. The ARN value
is {thingArn}");
            Console.WriteLine(new string('-', 80));

            // Step 1.1: List AWS IoT Things
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("2. List AWS IoT Things.");
            Console.WriteLine("Now let's list the IoT Things to see the Thing we
just created.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var things = await iotWrapper.ListThingsAsync();
            Console.WriteLine($"Found {things.Count} IoT Things:");
            foreach (var thing in things.Take(10)) // Show first 10 things
            {
                Console.WriteLine($"Thing Name: {thing.ThingName}");
                Console.WriteLine($"Thing ARN: {thing.ThingArn}");
                if (thing.Attributes != null && thing.Attributes.Any())
                {
                    Console.WriteLine("Attributes:");
```

```
                    foreach (var attr in thing.Attributes)
                    {
                        Console.WriteLine($"  {attr.Key}: {attr.Value}");
                    }
                }
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 2: Generate a Device Certificate
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("3. Generate a device certificate.");
            Console.WriteLine("A device certificate performs a role in securing the
communication between devices (Things) and the AWS IoT platform.");
            Console.WriteLine();

            var createCert = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to create a certificate for {thingName}?
(y/n)");
                createCert = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Creating certificate for {thingName}...");
            }

            if (createCert?.ToLower() == "y")
            {
                var certificateResult = await
iotWrapper.CreateKeysAndCertificateAsync();
                if (certificateResult.HasValue)
                {
                    var (certArn, certPem, certId) = certificateResult.Value;
                    certificateArn = certArn;
                    certificateId = certId;

                    Console.WriteLine($"\nCertificate:");
                    // Show only first few lines of certificate for brevity
                    var lines = certPem.Split('\n');
                    for (int i = 0; i < Math.Min(lines.Length, 5); i++)
                    {
```

```
                        Console.WriteLine(lines[i]);
                    }
                    if (lines.Length > 5)
                    {
                        Console.WriteLine("...");
                    }

                    Console.WriteLine($"\nCertificate ARN:");
                    Console.WriteLine(certificateArn);

                    // Step 3: Attach the Certificate to the AWS IoT Thing
                    Console.WriteLine("Attach the certificate to the AWS IoT
Thing.");
                    var attachResult = await
iotWrapper.AttachThingPrincipalAsync(thingName, certificateArn);
                    if (attachResult)
                    {
                        Console.WriteLine("Certificate attached to Thing
successfully.");
                    }
                    else
                    {
                        Console.WriteLine("Failed to attach certificate to Thing.");
                    }

                    Console.WriteLine("Thing Details:");
                    Console.WriteLine($"Thing Name: {thingName}");
                    Console.WriteLine($"Thing ARN: {thingArn}");
                }
                else
                {
                    Console.WriteLine("Failed to create certificate.");
                }
            }
            Console.WriteLine(new string('-', 80));

            // Step 4: Update an AWS IoT Thing with Attributes
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("4. Update an AWS IoT Thing with Attributes.");
            Console.WriteLine("IoT Thing attributes, represented as key-value pairs,
offer a pivotal advantage in facilitating efficient data");
            Console.WriteLine("management and retrieval within the AWS IoT
ecosystem.");
            Console.WriteLine();
```

```
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var attributes = new Dictionary<string, string>
            {
                { "Location", "Seattle" },
                { "DeviceType", "Sensor" },
                { "Firmware", "1.2.3" }
            };

            await iotWrapper.UpdateThingAsync(thingName, attributes);
            Console.WriteLine("Thing attributes updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 5: Return a unique endpoint specific to the Amazon Web Services
account
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("5. Return a unique endpoint specific to the Amazon
Web Services account.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var endpoint = await iotWrapper.DescribeEndpointAsync();
            if (endpoint != null)
            {
                var subdomain = endpoint.Split('.')[0];
                Console.WriteLine($"Extracted subdomain: {subdomain}");
                Console.WriteLine($"Full Endpoint URL: https://{endpoint}");
            }
            else
            {
                Console.WriteLine("Failed to retrieve endpoint.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 6: List your AWS IoT certificates
            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine("6. List your AWS IoT certificates");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var certificates = await iotWrapper.ListCertificatesAsync();
            foreach (var cert in certificates.Take(5)) // Show first 5 certificates
            {
                Console.WriteLine($"Cert id: {cert.CertificateId}");
                Console.WriteLine($"Cert Arn: {cert.CertificateArn}");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 7: Create an IoT shadow
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("7. Update an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowPayload = JsonSerializer.Serialize(new
            {
                state = new
                {
                    desired = new
                    {
                        temperature = 25,
                        humidity = 50
                    }
                }
            });

            await iotWrapper.UpdateThingShadowAsync(thingName, shadowPayload);
            Console.WriteLine("Thing Shadow updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 8: Write out the state information, in JSON format
```

```
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("8. Write out the state information, in JSON
format.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowData = await iotWrapper.GetThingShadowAsync(thingName);
            Console.WriteLine($"Received Shadow Data: {shadowData}");
            Console.WriteLine(new string('-', 80));

            // Step 9: Set up resources (SNS topic and IAM role) and create a rule
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("9. Set up resources and create a rule");
            Console.WriteLine();

            // Deploy CloudFormation stack to create SNS topic and IAM role
            Console.WriteLine("Deploying CloudFormation stack to create SNS topic
and IAM role...");

            var deployStack = !IsInteractive || GetYesNoResponse("Would you like to
deploy the CloudFormation stack? (y/n) ");
            if (deployStack)
            {
                if (IsInteractive)
                {
                    Console.Write(
                        $"Enter stack resource file path (or press Enter for default
'{_stackResourcePath}'): ");
                    var userResourcePath = Console.ReadLine();
                    if (!string.IsNullOrEmpty(userResourcePath))
                        _stackResourcePath = userResourcePath;
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);

                if (deploySuccess)
                {
                    // Get stack outputs
```

```
                    var stackOutputs = await GetStackOutputs(_stackName,
cloudFormationClient, scenarioLogger);
                    if (stackOutputs != null)
                    {
                        snsTopicArn = stackOutputs["SNSTopicArn"];
                        string roleArn = stackOutputs["RoleArn"];

                        Console.WriteLine($"Successfully deployed stack. SNS topic:
{snsTopicArn}");
                        Console.WriteLine($"Successfully deployed stack. IAM role:
{roleArn}");

                        if (IsInteractive)
                        {
                            Console.Write($"Enter Rule name (press Enter for default
'{ruleName}'): ");
                            var userRuleName = Console.ReadLine();
                            if (!string.IsNullOrEmpty(userRuleName))
                                ruleName = userRuleName;
                        }
                        else
                        {
                            Console.WriteLine($"Using default rule name:
{ruleName}");
                        }

                        // Now create the IoT rule with the CloudFormation outputs
                        var ruleResult = await
iotWrapper.CreateTopicRuleAsync(ruleName, snsTopicArn, roleArn);
                        if (ruleResult)
                        {
                            Console.WriteLine("IoT Rule created successfully.");
                        }
                        else
                        {
                            Console.WriteLine("Failed to create IoT rule.");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Failed to get stack outputs. Skipping
rule creation.");
                    }
                }
```

```
                else
                {
                    Console.WriteLine("Failed to deploy CloudFormation stack.
 Skipping rule creation.");
                }
            }
            else
            {
                Console.WriteLine("Skipping CloudFormation stack deployment and rule
 creation.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 10: List your rules
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("10. List your rules.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var rules = await iotWrapper.ListTopicRulesAsync();
            Console.WriteLine("List of IoT Rules:");
            foreach (var rule in rules.Take(5)) // Show first 5 rules
            {
                Console.WriteLine($"Rule Name: {rule.RuleName}");
                Console.WriteLine($"Rule ARN: {rule.RuleArn}");
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 11: Search things using the Thing name
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("11. Search things using the Thing name.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var searchResults = await iotWrapper.SearchIndexAsync($"thingName:
{thingName}");
```

```
            if (searchResults.Any())
            {
                Console.WriteLine($"Thing id found using search is
{searchResults.First().ThingId}");
            }
            else
            {
                Console.WriteLine($"No search results found for Thing:
{thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 12: Cleanup - Detach and delete certificate
            if (!string.IsNullOrEmpty(certificateArn))
            {
                Console.WriteLine(new string('-', 80));
                var deleteCert = "y";
                if (IsInteractive)
                {
                    Console.Write($"Do you want to detach and delete the certificate
for {thingName}? (y/n)");
                    deleteCert = Console.ReadLine();
                }
                else
                {
                    Console.WriteLine($"Detaching and deleting certificate for
{thingName}...");
                }

                if (deleteCert?.ToLower() == "y")
                {
                    Console.WriteLine("12. You selected to detach and delete the
certificate.");
                    if (IsInteractive)
                    {
                        Console.WriteLine("Press Enter to continue...");
                        Console.ReadLine();
                    }

                    await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                    Console.WriteLine($"{certificateArn} was successfully removed
from {thingName}");
```

```
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                    Console.WriteLine($"{certificateArn} was successfully
deleted.");
                }
                Console.WriteLine(new string('-', 80));
            }

            // Step 13: Delete the AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("13. Delete the AWS IoT Thing.");
            var deleteThing = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to delete the IoT Thing? (y/n)");
                deleteThing = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Deleting IoT Thing {thingName}...");
            }

            if (deleteThing?.ToLower() == "y")
            {
                await iotWrapper.DeleteThingAsync(thingName);
                Console.WriteLine($"Deleted Thing {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 14: Clean up CloudFormation stack
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                Console.WriteLine(new string('-', 80));
                Console.WriteLine("14. Clean up CloudFormation stack.");
                Console.WriteLine("Deleting the CloudFormation stack and all
resources...");

                var cleanup = !IsInteractive || GetYesNoResponse("Do you want to
delete the CloudFormation stack and all resources? (y/n) ");
                if (cleanup)
                {
                    var ruleCleanupSuccess = await
iotWrapper.DeleteTopicRuleAsync(ruleName);
```

```
                    var stackCleanupSuccess = await
 DeleteCloudFormationStack(_stackName, cloudFormationClient, scenarioLogger);
                    if (ruleCleanupSuccess && stackCleanupSuccess)
                    {
                        Console.WriteLine("Successfully cleaned up CloudFormation
 stack and all resources.");
                    }
                    else
                    {
                        Console.WriteLine("Some cleanup operations failed. Check the
 logs for details.");
                    }
                }
                else
                {
                    Console.WriteLine($"Resources will remain. Stack name:
 {_stackName}");
                }
                Console.WriteLine(new string('-', 80));
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "Error occurred during scenario
 execution.");

            // Cleanup on error
            if (!string.IsNullOrEmpty(certificateArn) && !
string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DetachThingPrincipalAsync(thingName,
 certificateArn);
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
                }
            }

            if (!string.IsNullOrEmpty(thingName))
            {
```

```
                try
                {
                    await iotWrapper.DeleteThingAsync(thingName);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during Thing
cleanup.");
                }
            }

            // Clean up CloudFormation stack on error
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                try
                {
                    await _iotWrapper.DeleteTopicRuleAsync(ruleName);
                    await DeleteCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during CloudFormation
stack cleanup.");
                }
            }

            throw;
        }
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
```

```
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = new List<string> { Capability.CAPABILITY_NAMED_IAM }
            };

            var response = await cloudFormationClient.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await WaitForStackCompletion(response.StackId,
cloudFormationClient, scenarioLogger);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created successfully.");
                    return true;
                }
                else
                {
                    scenarioLogger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                scenarioLogger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            scenarioLogger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName,
cloudFormationClient, scenarioLogger);
```

```
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
```

```
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError("Timed out waiting for CloudFormation stack creation
to complete.");
        return false;
    }

    /// <summary>
    /// Gets the outputs from the CloudFormation stack.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>A dictionary of stack outputs.</returns>
    private static async Task<Dictionary<string, string>?> GetStackOutputs(string
stackName, IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics>
scenarioLogger)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            var response = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (response.Stacks.Count > 0)
            {
                var outputs = new Dictionary<string, string>();
                foreach (var output in response.Stacks[0].Outputs)
                {
                    outputs[output.OutputKey] = output.OutputValue;
                }
```

```
                return outputs;
            }

            return null;
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"Failed to get stack outputs for
{stackName}");
            return null;
        }
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        try
        {
            var request = new DeleteStackRequest
            {
                StackName = stackName
            };

            await cloudFormationClient.DeleteStackAsync(request);
            Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

            bool stackDeleted = await WaitForStackDeletion(stackName,
cloudFormationClient, scenarioLogger);

            if (stackDeleted)
            {
                Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
                return true;
            }
            else
```

```
            {
                scenarioLogger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
                return false;
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deleting the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
```

```
                    return true;
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to be
deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (IsInteractive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press Enter
for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
```

```
                var regex = new System.Text.RegularExpressions.Regex("[a-zA-Z][-a-
zA-Z0-9]*");
                if (!regex.IsMatch(input))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
 {_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }
}
```

Uma classe de invólucro para métodos AWS IoT SDK.

```
/// <summary>
/// Wrapper methods to use Amazon IoT Core with .NET.
/// </summary>
public class IoTWrapper
{
    private readonly IAmazonIoT _amazonIoT;
    private readonly IAmazonIotData _amazonIotData;
    private readonly ILogger<IoTWrapper> _logger;

    /// <summary>
    /// Constructor for the IoT wrapper.
    /// </summary>
    /// <param name="amazonIoT">The injected IoT client.</param>
    /// <param name="amazonIotData">The injected IoT Data client.</param>
    /// <param name="logger">The injected logger.</param>
    public IoTWrapper(IAmazonIoT amazonIoT, IAmazonIotData amazonIotData,
 ILogger<IoTWrapper> logger)
    {
        _amazonIoT = amazonIoT;
        _amazonIotData = amazonIotData;
        _logger = logger;
    }

    /// <summary>
    /// Creates an AWS IoT Thing.
```

```
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
{response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or null
if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };
```

```
            var response = await _amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to Thing
{thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
```

```
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
```

```
    /// <returns>List of certificate information, or empty list if listing failed.</
returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
 certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
 {ex.Message}");
            return new List<Certificate>();
        }
    }

    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
 shadowPayload)
    {
        try
        {
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
 MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };
```

```
            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
{ex.Message}");
```

```
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };
```

```
            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteTopicRuleAsync(string ruleName)
    {
        try
        {
            var request = new DeleteTopicRuleRequest
            {
                RuleName = ruleName,
            };

            await _amazonIoT.DeleteTopicRuleAsync(request);
            _logger.LogInformation($"Deleted IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogWarning($"Rule {ruleName} not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete topic rule. Here's why:
{ex.Message}");
```

```
            return false;
        }
    }

    /// <summary>
    /// Lists all IoT topic rules.
    /// </summary>
    /// <returns>List of topic rules, or empty list if listing failed.</returns>
    public async Task<List<TopicRuleListItem>> ListTopicRulesAsync()
    {
        try
        {
            var request = new ListTopicRulesRequest();
            var response = await _amazonIoT.ListTopicRulesAsync(request);

            _logger.LogInformation($"Retrieved {response.Rules.Count} IoT rules");
            return response.Rules;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list topic rules. Here's why:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
    }

    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
```

```
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count} Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting for
the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
```

```
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });

            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready (attempt
{attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if it's
ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if (configResponse.ThingIndexingConfiguration?.ThingIndexingMode
== ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await _amazonIoT.SearchIndexAsync(request);
```

```
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready after
configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
certificateArn)
    {
        try
```

```
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
```

```
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
```

```
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the next
page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
```

```
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration

- UpdateThing

Ações

`AttachThingPrincipal`

O código de exemplo a seguir mostra como usar `AttachThingPrincipal`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Attaches a certificate to an IoT Thing.
   /// </summary>
   /// <param name="thingName">The name of the Thing.</param>
   /// <param name="certificateArn">The ARN of the certificate to attach.</param>
   /// <returns>True if successful, false otherwise.</returns>
   public async Task<bool> AttachThingPrincipalAsync(string thingName, string
certificateArn)
   {
       try
       {
           var request = new AttachThingPrincipalRequest
           {
               ThingName = thingName,
               Principal = certificateArn
           };

           await _amazonIoT.AttachThingPrincipalAsync(request);
           _logger.LogInformation($"Attached certificate {certificateArn} to Thing
{thingName}");
           return true;
       }
       catch (Amazon.IoT.Model.ResourceNotFoundException ex)
       {
```

```
            _logger.LogError($"Cannot attach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte AttachThingPrincipal em Referência AWS SDK para .NET da API.

`CreateKeysAndCertificate`

O código de exemplo a seguir mostra como usar `CreateKeysAndCertificate`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or null
if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
```

```
            };

            var response = await _amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte CreateKeysAndCertificate em Referência AWS SDK para .NET da API.

`CreateThing`

O código de exemplo a seguir mostra como usar `CreateThing`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
```

```
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
{response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte CreateThing em Referência AWS SDK para .NET da API.

`CreateTopicRule`

O código de exemplo a seguir mostra como usar `CreateTopicRule`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };
```

```
            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateTopicRule em Referência AWS SDK para .NET da API.

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
```

```
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteCertificate em Referência AWS SDK para .NET da API.

DeleteThing

O código de exemplo a seguir mostra como usar `DeleteThing`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteThing em Referência AWS SDK para .NET da API.

`DescribeEndpoint`

O código de exemplo a seguir mostra como usar `DescribeEndpoint`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte DescribeEndpoint em Referência AWS SDK para .NET da API.

`DetachThingPrincipal`

O código de exemplo a seguir mostra como usar `DetachThingPrincipal`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Detaches a certificate from an IoT Thing.
   /// </summary>
   /// <param name="thingName">The name of the Thing.</param>
   /// <param name="certificateArn">The ARN of the certificate to detach.</param>
   /// <returns>True if successful, false otherwise.</returns>
   public async Task<bool> DetachThingPrincipalAsync(string thingName, string
certificateArn)
   {
       try
       {
           var request = new DetachThingPrincipalRequest
           {
               ThingName = thingName,
               Principal = certificateArn
           };

           await _amazonIoT.DetachThingPrincipalAsync(request);
           _logger.LogInformation($"Detached certificate {certificateArn} from
Thing {thingName}");
           return true;
       }
       catch (Amazon.IoT.Model.ResourceNotFoundException ex)
       {
```

```
            _logger.LogError($"Cannot detach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DetachThingPrincipal em Referência AWS SDK para .NET da API.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing failed.</
returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);
```

```
            _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
{ex.Message}");
            return new List<Certificate>();
        }
    }
```

- Para obter detalhes da API, consulte ListCertificates em Referência AWS SDK para .NET da API.

`ListThings`

O código de exemplo a seguir mostra como usar `ListThings`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
```

```
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the next
page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }
```

- Para obter detalhes da API, consulte ListThings em Referência AWS SDK para .NET da API.

`SearchIndex`

O código de exemplo a seguir mostra como usar `SearchIndex`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count} Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
```

```
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting for
the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });
```

```
            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready (attempt
{attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if it's
ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if (configResponse.ThingIndexingConfiguration?.ThingIndexingMode
== ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await _amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
```

```
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready after
configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }
```

- Para obter detalhes da API, consulte SearchIndex em Referência AWS SDK para .NET da API.

`UpdateThing`

O código de exemplo a seguir mostra como usar `UpdateThing`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
```

```
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte UpdateThing em Referência AWS SDK para .NET da API.

AWS IoT data exemplos usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com AWS IoT data.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetThingShadow`

O código de exemplo a seguir mostra como usar `GetThingShadow`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Gets the Thing's shadow information.
/// </summary>
/// <param name="thingName">The name of the Thing.</param>
/// <returns>The shadow data as a string, or null if retrieval failed.</returns>
public async Task<string?> GetThingShadowAsync(string thingName)
{
    try
    {
        var request = new GetThingShadowRequest
        {
            ThingName = thingName
        };

        var response = await _amazonIotData.GetThingShadowAsync(request);
        using var reader = new StreamReader(response.Payload);
```

```
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte GetThingShadow em Referência AWS SDK para .NET da API.

`UpdateThingShadow`

O código de exemplo a seguir mostra como usar `UpdateThingShadow`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
```

```
   public async Task<bool> UpdateThingShadowAsync(string thingName, string
shadowPayload)
   {
       try
       {
           var request = new UpdateThingShadowRequest
           {
               ThingName = thingName,
               Payload = new
MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
           };

           await _amazonIotData.UpdateThingShadowAsync(request);
           _logger.LogInformation($"Updated shadow for Thing {thingName}");
           return true;
       }
       catch (Amazon.IotData.Model.ResourceNotFoundException ex)
       {
           _logger.LogError($"Cannot update Thing shadow - resource not found:
{ex.Message}");
           return false;
       }
       catch (Exception ex)
       {
           _logger.LogError($"Couldn't update Thing shadow. Here's why:
{ex.Message}");
           return false;
       }
   }
```

- Para obter detalhes da API, consulte UpdateThingShadow em Referência AWS SDK para .NET da API.

Exemplos do Amazon Redshift usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o Amazon Redshift.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Redshift

O exemplo de código a seguir mostra como começar a usar o Amazon Redshift.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Main method to run the Hello Amazon Redshift example.
/// </summary>
/// <param name="args">Command line arguments (not used).</param>
public static async Task Main(string[] args)
{
    var redshiftClient = new AmazonRedshiftClient();

    Console.WriteLine("Hello, Amazon Redshift! Let's list available clusters:");

    var clusters = new List<Cluster>();

    try
```

```
        {
            // Use pagination to retrieve all clusters.
            var clustersPaginator = redshiftClient.Paginators.DescribeClusters(new
DescribeClustersRequest());

            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");

            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }
        }
        catch (AmazonRedshiftException ex)
        {
            Console.WriteLine($"Couldn't list clusters. Here's why: {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
```

- Para obter detalhes da API, consulte DescribeClusters em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um cluster do Redshift.
- Listar bancos de dados no cluster.
- Criar uma tabela chamada Filmes.

- Preencher a tabela Filmes.
- Consultar a tabela Filmes por ano.
- Modificar o cluster do Redshift.
- Excluir o cluster do Amazon Redshift.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe de wrapper do Redshift para gerenciar as operações.

```
/// <summary>
/// Wrapper class for Amazon Redshift operations.
/// </summary>
public class RedshiftWrapper
{
    private readonly IAmazonRedshift _redshiftClient;
    private readonly IAmazonRedshiftDataAPIService _redshiftDataClient;

    /// <summary>
    /// Constructor for RedshiftWrapper.
    /// </summary>
    /// <param name="redshiftClient">Amazon Redshift client.</param>
    /// <param name="redshiftDataClient">Amazon Redshift Data API client.</param>
    public RedshiftWrapper(IAmazonRedshift redshiftClient,
 IAmazonRedshiftDataAPIService redshiftDataClient)
    {
        _redshiftClient = redshiftClient;
        _redshiftDataClient = redshiftDataClient;
    }

    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
    /// <param name="masterUsername">The master username.</param>
```

```
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier, string
databaseName,
        string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
    {
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create cluster. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe Amazon Redshift clusters.
    /// </summary>
    /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
    /// <returns>A list of clusters.</returns>
```

```
    public async Task<List<Cluster>> DescribeClustersAsync(string? clusterIdentifier
= null)
    {
        try
        {
            var clusters = new List<Cluster>();
            var request = new DescribeClustersRequest();
            if (!string.IsNullOrEmpty(clusterIdentifier))
            {
                request.ClusterIdentifier = clusterIdentifier;
            }

            var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
```

```
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance window.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
 preferredMaintenanceWindow)
    {
        try
        {
            var request = new ModifyClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                PreferredMaintenanceWindow = preferredMaintenanceWindow
            };

            var response = await _redshiftClient.ModifyClusterAsync(request);
            Console.WriteLine($"The modified cluster was successfully modified and
 has {response.Cluster.PreferredMaintenanceWindow} as the maintenance window");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't modify cluster. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete an Amazon Redshift cluster without a final snapshot.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
 clusterIdentifier)
    {
        try
```

```
        {
            var request = new DeleteClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                SkipFinalClusterSnapshot = true
            };

            var response = await _redshiftClient.DeleteClusterAsync(request);
            Console.WriteLine($"The {clusterIdentifier} was deleted");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete cluster. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();
```

```
            foreach (var database in response.Databases)
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create a table in the Redshift database.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> CreateTableAsync(string clusterIdentifier, string
database, string dbUser)
    {
        try
        {
            var sqlStatement = @"
                CREATE TABLE Movies (
                    id INTEGER PRIMARY KEY,
                    title VARCHAR(250) NOT NULL,
                    year INTEGER NOT NULL
                )";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
```

```
                DbUser = dbUser,
                Sql = sqlStatement
            };

            var response = await _redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine("Table created: Movies");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create table. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Insert a record into the Movies table using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="id">The movie ID.</param>
    /// <param name="title">The movie title.</param>
    /// <param name="year">The movie year.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> InsertMovieAsync(string clusterIdentifier, string
database, string dbUser,
        int id, string title, int year)
    {
        try
        {
            var sqlStatement = "INSERT INTO Movies (id, title, year) VALUES
(:id, :title, :year)";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
```

```
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "id", Value = id.ToString() },
                    new SqlParameter { Name = "title", Value = title },
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };

            var response = await _redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine($"Inserted: {title} ({year})");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't insert movie. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Query movies by year using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="year">The year to query.</param>
    /// <returns>A list of movie titles.</returns>
    public async Task<List<string>> QueryMoviesByYearAsync(string clusterIdentifier,
string database,
        string dbUser, int year)
    {
        try
        {
            var sqlStatement = "SELECT title FROM Movies WHERE year = :year";

            var request = new ExecuteStatementRequest
```

```
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };

            var response = await _redshiftDataClient.ExecuteStatementAsync(request);
            Console.WriteLine($"The identifier of the statement is {response.Id}");

            await WaitForStatementToCompleteAsync(response.Id);

            var results = await GetStatementResultAsync(response.Id);
            var movieTitles = new List<string>();

            foreach (var row in results)
            {
                if (row.Count > 0)
                {
                    var title = row[0].StringValue;
                    Console.WriteLine($"The Movie title field is {title}");
                    movieTitles.Add(title);
                }
            }

            return movieTitles;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't query movies. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
```

```
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
    public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
    {
        try
        {
            var request = new DescribeStatementRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get the results of a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A list of result rows.</returns>
    public async Task<List<List<Field>>> GetStatementResultAsync(string statementId)
    {
        try
        {
            var request = new GetStatementResultRequest
            {
                Id = statementId
            };
```

```
            var response = await
_redshiftDataClient.GetStatementResultAsync(request);
            return response.Records;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait for a statement to complete execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    private async Task WaitForStatementToCompleteAsync(string statementId)
    {
        var status = StatusString.SUBMITTED;
        DescribeStatementResponse? response = null;

        while (status == StatusString.SUBMITTED || status == StatusString.PICKED ||
status == StatusString.STARTED)
        {
            await Task.Delay(1000); // Wait 1 second
            response = await DescribeStatementAsync(statementId);
            status = response.Status;
            Console.WriteLine($"...{status}");
        }

        if (status == StatusString.FINISHED)
        {
            Console.WriteLine("The statement is finished!");
        }
        else
        {
            var errorMessage = response?.Error ?? "Unknown error";
```

```
            Console.WriteLine($"The statement failed with status: {status}");
            Console.WriteLine($"Error message: {errorMessage}");
        }
    }

    /// <summary>
    /// Wait for a cluster to become available.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public async Task WaitForClusterAvailableAsync(string clusterIdentifier)
    {
        Console.WriteLine($"Wait until {clusterIdentifier} is available. This may
 take a few minutes.");

        var startTime = DateTime.Now;
        var clusters = await DescribeClustersAsync(clusterIdentifier);

        while (clusters[0].ClusterStatus != "available")
        {
            var elapsed = DateTime.Now - startTime;
            Console.WriteLine($"Elapsed Time: {elapsed:mm\\:ss} - Waiting for
 cluster...");

            await Task.Delay(5000); // Wait 5 seconds
            clusters = await DescribeClustersAsync(clusterIdentifier);
        }

        var totalElapsed = DateTime.Now - startTime;
        Console.WriteLine($"Cluster is available! Total Elapsed Time:
 {totalElapsed:mm\\:ss}");
    }
}
```

Execute um cenário interativo que demonstra os fundamentos do Redshift.

```
/// <summary>
/// Amazon Redshift Getting Started Scenario.
/// </summary>
public class RedshiftBasics
{
    public static bool IsInteractive = true;
```

```
    public static RedshiftWrapper? Wrapper = null;
    public static ILogger logger = null!;
    private static readonly string _moviesFilePath = "../../../../../../resources/
sample_files/movies.json";

    /// <summary>
    /// Main method for the Amazon Redshift Getting Started scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRedshift>()
                    .AddAWSService<IAmazonRedshiftDataAPIService>()
                    .AddTransient<RedshiftWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<RedshiftBasics>();

        Wrapper = host.Services.GetRequiredService<RedshiftWrapper>();

        await RunScenarioAsync();
    }

    /// <summary>
    /// Run the complete Amazon Redshift scenario.
    /// </summary>
    public static async Task RunScenarioAsync()
    {
        // Set all variables to default values
        string userName = "awsuser";
        string userPassword = "AwsUser1000";
        string clusterIdentifier = "redshift-cluster-movies";
        var databaseName = "dev";
        int recordCount = 50;
        int year = 2013;
        try
        {
            Console.WriteLine(

 "================================================================================");
```

```
            Console.WriteLine("Welcome to the Amazon Redshift SDK Getting Started
 scenario.");
            Console.WriteLine(
                "This .NET program demonstrates how to interact with Amazon Redshift
 by using the AWS SDK for .NET.");
            Console.WriteLine("Let's get started...");
            Console.WriteLine(

 "================================================================================");

            // Step 1: Get user credentials (if interactive)
            if (IsInteractive)
            {
                Console.WriteLine("Please enter a user name for the cluster (default
 is awsuser):");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    userName = userInput;


 Console.WriteLine("=================================================================
                Console.WriteLine("Please enter a user password for the cluster
 (default is AwsUser1000):");
                var passwordInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(passwordInput))
                    userPassword = passwordInput;


 Console.WriteLine("=================================================================

                // Step 2: Get cluster identifier
                Console.WriteLine("Enter a cluster id value (default is redshift-
cluster-movies):");
                var clusterInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(clusterInput))
                    clusterIdentifier = clusterInput;
            }
            else
            {
                Console.WriteLine($"Using default values: userName={userName},
 clusterIdentifier={clusterIdentifier}");
            }

            // Step 3: Create Redshift cluster
```

```
            await Wrapper!.CreateClusterAsync(clusterIdentifier, databaseName,
userName, userPassword);

Console.WriteLine("==========================================================================

            // Step 4: Wait for cluster to become available

Console.WriteLine("==========================================================================
            await Wrapper.WaitForClusterAvailableAsync(clusterIdentifier);

Console.WriteLine("==========================================================================

            // Step 5: List databases

Console.WriteLine("==========================================================================
            Console.WriteLine($" When you created {clusterIdentifier}, the dev
database is created by default and used in this scenario.");
            Console.WriteLine(" To create a custom database, you need to have a
CREATEDB privilege.");
            Console.WriteLine(" For more information, see the documentation here:
https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

Console.WriteLine("==========================================================================


Console.WriteLine("==========================================================================
            Console.WriteLine($"List databases in {clusterIdentifier}");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.ListDatabasesAsync(clusterIdentifier, userName,
databaseName);

Console.WriteLine("==========================================================================

            // Step 6: Create Movies table
```

```
Console.WriteLine("==========================================================================
            Console.WriteLine("Now you will create a table named Movies.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.CreateTableAsync(clusterIdentifier, databaseName,
userName);

Console.WriteLine("==========================================================================

            // Step 7: Populate the Movies table

Console.WriteLine("==========================================================================
            Console.WriteLine("Populate the Movies table using the Movies.json
file.");

            if (IsInteractive)
            {
                Console.WriteLine("Specify the number of records you would like to
add to the Movies Table.");
                Console.WriteLine("Please enter a value between 50 and 200.");
                Console.Write("Enter a value: ");

                var recordCountInput = Console.ReadLine();
                if (int.TryParse(recordCountInput, out var inputCount) && inputCount
is >= 50 and <= 200)
                {
                    recordCount = inputCount;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{recordCount}.");
                }
            }
            else
            {
                Console.WriteLine($"Using default record count: {recordCount}");
            }
```

```
            await PopulateMoviesTableAsync(clusterIdentifier, databaseName,
userName, recordCount);
            Console.WriteLine($"{recordCount} records were added to the Movies
table.");

Console.WriteLine("==========================================================================

            // Step 8 & 9: Query movies by year

Console.WriteLine("==========================================================================
            Console.WriteLine("Query the Movies table by year. Enter a value between
2012-2014.");

            if (IsInteractive)
            {
                Console.Write("Enter a year: ");
                var yearInput = Console.ReadLine();
                if (int.TryParse(yearInput, out var inputYear) && inputYear is >=
2012 and <= 2014)
                {
                    year = inputYear;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{year}.");
                }
            }
            else
            {
                Console.WriteLine($"Using default year: {year}");
            }

            await Wrapper.QueryMoviesByYearAsync(clusterIdentifier, databaseName,
userName, year);

Console.WriteLine("==========================================================================

            // Step 10: Modify the cluster

Console.WriteLine("==========================================================================
            Console.WriteLine("Now you will modify the Redshift cluster.");
            if (IsInteractive)
            {
```

```
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.ModifyClusterAsync(clusterIdentifier, "wed:07:30-
wed:08:00");

 Console.WriteLine("==========================================================================

            // Step 11 & 12: Delete cluster confirmation

 Console.WriteLine("==========================================================================
            if (IsInteractive)
            {
                Console.WriteLine("Would you like to delete the Amazon Redshift
 cluster? (y/n)");
                var deleteResponse = Console.ReadLine();
                if (deleteResponse?.ToLower() == "y")
                {
                    await
 Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
                }
            }
            else
            {
                Console.WriteLine("Deleting the Amazon Redshift cluster...");
                await Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            }

 Console.WriteLine("==========================================================================


 Console.WriteLine("==========================================================================
            Console.WriteLine("This concludes the Amazon Redshift SDK Getting
 Started scenario.");

 Console.WriteLine("==========================================================================
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during the scenario:
 {ex.Message}");
            Console.WriteLine("Deleting the Amazon Redshift cluster...");
            await Wrapper!.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            throw;
```

```
        }
    }

    /// <summary>
    /// Populate the Movies table with data from the JSON file.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="recordCount">Number of records to insert.</param>
    private static async Task PopulateMoviesTableAsync(string clusterIdentifier,
string database, string dbUser, int recordCount)
    {
        if (!File.Exists(_moviesFilePath))
        {
            throw new FileNotFoundException($"Required movies data file not found
at: {_moviesFilePath}");
        }

        var jsonContent = await File.ReadAllTextAsync(_moviesFilePath);
        var options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };
        var movies = JsonSerializer.Deserialize<List<Movie>>(jsonContent, options);

        if (movies == null || movies.Count == 0)
        {
            throw new InvalidOperationException("Failed to parse movies JSON file or
file is empty.");
        }

        var insertCount = Math.Min(recordCount, movies.Count);

        for (int i = 0; i < insertCount; i++)
        {
            var movie = movies[i];
            await Wrapper!.InsertMovieAsync(clusterIdentifier, database, dbUser, i,
movie.Title, movie.Year);
        }
    }

    /// <summary>
    /// Movie data model.
```

```
    /// </summary>
    private class Movie
    {
        public string Title { get; set; } = string.Empty;
        public int Year { get; set; }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
```

```
    /// <param name="masterUsername">The master username.</param>
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier, string
databaseName,
        string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
    {
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create cluster. Here's why: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte CreateCluster em Referência AWS SDK para .NET da API.

`DeleteCluster`

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete an Amazon Redshift cluster without a final snapshot.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
clusterIdentifier)
    {
        try
        {
            var request = new DeleteClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                SkipFinalClusterSnapshot = true
            };

            var response = await _redshiftClient.DeleteClusterAsync(request);
            Console.WriteLine($"The {clusterIdentifier} was deleted");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete cluster. Here's why: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteCluster em Referência AWS SDK para .NET da API.

`DescribeClusters`

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe Amazon Redshift clusters.
    /// </summary>
    /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
    /// <returns>A list of clusters.</returns>
    public async Task<List<Cluster>> DescribeClustersAsync(string? clusterIdentifier
= null)
    {
        try
        {
            var clusters = new List<Cluster>();
            var request = new DescribeClustersRequest();
            if (!string.IsNullOrEmpty(clusterIdentifier))
            {
                request.ClusterIdentifier = clusterIdentifier;
            }

            var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
```

```
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeClusters em Referência AWS SDK para .NET da API.

`DescribeStatement`

O código de exemplo a seguir mostra como usar `DescribeStatement`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
    public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
    {
        try
        {
            var request = new DescribeStatementRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte DescribeStatement em Referência AWS SDK para .NET da API.

`GetStatementResult`

O código de exemplo a seguir mostra como usar `GetStatementResult`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Get the results of a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A list of result rows.</returns>
    public async Task<List<List<Field>>> GetStatementResultAsync(string statementId)
    {
        try
        {
            var request = new GetStatementResultRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.GetStatementResultAsync(request);
            return response.Records;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte GetStatementResult em Referência AWS SDK para .NET da API.

`ListDatabases`

O código de exemplo a seguir mostra como usar `ListDatabases`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();

            foreach (var database in response.Databases)
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
```

```
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why: {ex.Message}");
            throw;
        }
    }
```

- Para obter detalhes da API, consulte ListDatabases em Referência AWS SDK para .NET da API.

`ModifyCluster`

O código de exemplo a seguir mostra como usar `ModifyCluster`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance window.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
 preferredMaintenanceWindow)
    {
        try
        {
            var request = new ModifyClusterRequest
            {
```

```
                ClusterIdentifier = clusterIdentifier,
                PreferredMaintenanceWindow = preferredMaintenanceWindow
            };

            var response = await _redshiftClient.ModifyClusterAsync(request);
            Console.WriteLine($"The modified cluster was successfully modified and
has {response.Cluster.PreferredMaintenanceWindow} as the maintenance window");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't modify cluster. Here's why: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte ModifyCluster em Referência AWS SDK para .NET da API.

Exemplos do Amazon S3 usando SDK para .NET (v4)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para .NET (v4) com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Hello Amazon Simple Storage Service
// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var s3Client = new AmazonS3Client();

        try
        {
```

```
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
            Console.WriteLine(new string('-', 80));

            // Use the built-in paginator to list buckets
            var request = new ListBucketsRequest();
            var paginator = s3Client.Paginators.ListBuckets(request);

            var buckets = new List<S3Bucket>();

            await foreach (var response in paginator.Responses)
            {
                buckets.AddRange(response.Buckets);
            }

            if (buckets.Any())
            {
                Console.WriteLine($"Found {buckets.Count} S3 buckets:");
                Console.WriteLine();

                foreach (var bucket in buckets)
                {
                    Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
                    Console.WriteLine($"  Creation Date: {bucket.CreationDate:yyyy-
MM-dd HH:mm:ss UTC}");
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No S3 buckets found in your account.");
            }

            Console.WriteLine("Hello S3 completed successfully.");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"S3 service error occurred: {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list S3 buckets. Here's why:
 {ex.Message}");
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para .NET da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon S3.

```
public class S3_Basics
{
    public static bool IsInteractive = true;
    public static string BucketName = null!;
    public static string TempFilePath = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static ILogger<S3_Basics> _logger = null!;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddLogging(builder => builder.AddConsole()))
            .Build();

        _logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<S3_Basics>();

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();

        var sepBar = new string('-', 45);

        Console.WriteLine(sepBar);
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
        Console.WriteLine("procedures. This application will:");
        Console.WriteLine("\n\t1. Create a bucket");
        Console.WriteLine("\n\t2. Upload an object to the new bucket");
        Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
        Console.WriteLine("\n\t4. List the items in the new bucket");
        Console.WriteLine("\n\t5. Delete all the items in the bucket");
        Console.WriteLine("\n\t6. Delete the bucket");
        Console.WriteLine(sepBar);

        await RunScenario(_s3Wrapper, _logger);

        Console.WriteLine(sepBar);
        Console.WriteLine("The Amazon S3 scenario has successfully completed.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Run the S3 Basics scenario with injected dependencies.
    /// </summary>
    /// <param name="s3Wrapper">The S3 wrapper instance.</param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
scenarioLogger)
    {
        string bucketName = BucketName;
        string filePath = TempFilePath;
        string keyName = string.Empty;
```

```
        var sepBar = new string('-', 45);

        try
        {
            // Create a bucket.
            Console.WriteLine($"\n{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                Console.Write("Please enter a name for the new bucket: ");
                bucketName = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Using bucket name: {bucketName}");
            }

            var success = await s3Wrapper.CreateBucketAsync(bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                // Get the local path and filename for the file to upload.
                while (string.IsNullOrEmpty(filePath))
                {
                    Console.Write("Please enter the path and filename of the file to
upload: ");
                    filePath = Console.ReadLine();

                    // Confirm that the file exists on the local computer.
```

```
                    if (!File.Exists(filePath))
                    {
                        Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
                        filePath = string.Empty;
                    }
                }
            }
            else
            {
                // Use the public variable if set, otherwise create a temp file
                if (!string.IsNullOrEmpty(TempFilePath))
                {
                    filePath = TempFilePath;
                    Console.WriteLine($"Using provided test file: {filePath}");
                }
                else
                {
                    // Create a temporary test file for non-interactive mode
                    filePath = Path.GetTempFileName();
                    var testContent = "This is a test file for S3 basics scenario.
\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss UTC");
                    await File.WriteAllTextAsync(filePath, testContent);
                    Console.WriteLine($"Created temporary test file: {filePath}");
                }
            }

            // Get the file name from the full path.
            keyName = Path.GetFileName(filePath);

            success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
 filePath);

            if (success)
            {
                Console.WriteLine($"Successfully uploaded {keyName} from {filePath}
 to {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not upload {keyName}.\n");
            }

            // Set up download path
```

```
            string downloadPath = string.Empty;

            if (IsInteractive)
            {
                // Now get a new location where we can save the file.
                while (string.IsNullOrEmpty(downloadPath))
                {
                    // First get the path to which the file will be downloaded.
                    Console.Write("Please enter the path where the file will be
downloaded: ");
                    downloadPath = Console.ReadLine();

                    // Confirm that the file doesn't already exist on the local
computer.
                    if (File.Exists($"{downloadPath}\\{keyName}"))
                    {
                        Console.WriteLine($"Sorry, the file already exists in that
location.\n");
                        downloadPath = string.Empty;
                    }
                }
            }
            else
            {
                downloadPath = Path.GetTempPath();
                var downloadFile = Path.Combine(downloadPath, keyName);
                if (File.Exists(downloadFile))
                {
                    File.Delete(downloadFile);
                }

                Console.WriteLine($"Using download path: {downloadPath}");
            }

            // Download an object from a bucket.
            success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

            if (success)
            {
                Console.WriteLine($"Successfully downloaded {keyName}.\n");
            }
            else
            {
```

```
                Console.WriteLine($"Sorry, could not download {keyName}.\n");
            }

            // Copy the object to a different folder in the bucket.
            string folderName = string.Empty;

            if (IsInteractive)
            {
                while (string.IsNullOrEmpty(folderName))
                {
                    Console.Write("Please enter the name of the folder to copy your
 object to: ");
                    folderName = Console.ReadLine();
                }
            }
            else
            {
                folderName = "test-folder";
                Console.WriteLine($"Using folder name: {folderName}");
            }

            await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
 folderName);

            // List the objects in the bucket.
            await s3Wrapper.ListBucketContentsAsync(bucketName);

            // Delete the contents of the bucket.
            if (IsInteractive)
            {
                Console.WriteLine("Press <Enter> when you are ready to delete the
 bucket contents.");
                _ = Console.ReadLine();
            }

            var deleteContentsSuccess = await
 s3Wrapper.DeleteBucketContentsAsync(bucketName);
            if (deleteContentsSuccess)
            {
                Console.WriteLine($"Successfully deleted contents of {bucketName}.
\n");
            }
            else
            {
```

```
                Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
            }

            if (IsInteractive)
            {
                // Deleting the bucket too quickly after separately deleting its
contents can
                // cause an error that the bucket isn't empty. To delete contents
and bucket in one
                // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
                Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
                _ = Console.ReadLine();
            }
            else
            {
                // Add a small delay for non-interactive mode to ensure objects are
fully deleted.
                Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
                await Task.Delay(2000);
            }

            // Delete the bucket.
            var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
            if (deleteSuccess)
            {
                Console.WriteLine($"Successfully deleted {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
            }

            // Clean up temporary files in non-interactive mode
            if (!IsInteractive)
            {
                try
                {
                    if (File.Exists(filePath))
                    {
                        File.Delete(filePath);
                        Console.WriteLine("Cleaned up temporary test file.");
```

```
                    }

                    var downloadFile = Path.Combine(downloadPath, keyName);
                    if (File.Exists(downloadFile))
                    {
                        File.Delete(downloadFile);
                        Console.WriteLine("Cleaned up downloaded test file.");
                    }
                }
                catch (Exception ex)
                {
                    scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
                }
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

            // Clean up on error - delete bucket if it exists
            try
            {
                if (!string.IsNullOrEmpty(bucketName))
                {
                    await s3Wrapper.DeleteBucketContentsAsync(bucketName);
                    await s3Wrapper.DeleteBucketAsync(bucketName);
                }
            }
            catch (Exception cleanupEx)
            {
                scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
            }

            // Clean up temporary files in non-interactive mode
            if (!IsInteractive)
            {
                try
                {
                    if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
                    {
                        File.Delete(filePath);
                    }
```

```
                }
                catch (Exception fileCleanupEx)
                {
                    scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
 temporary files during error handling.");
                }
            }

            throw;
        }
    }
}
```

Uma classe de wrapper para os métodos de SDK do Amazon S3.

```
using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Initializes a new instance of the <see cref="S3Wrapper"/> class.
    /// </summary>
    /// <param name="amazonS3">An initialized Amazon S3 client object.</param>
    public S3Wrapper(IAmazonS3 amazonS3)
    {
        _amazonS3 = amazonS3;
    }


    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
```

```
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
    {
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
            };

            var response = await _amazonS3.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }



    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
```

```
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await _amazonS3.GetObjectAsync(request);

        try
        {
```

```
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }



    /// <summary>
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
    {
        try
        {
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
                MaxKeys = 5,
            };

            if (printList)
            {
                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");
            }

            var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });
            var s3Objects = new List<S3Object>();
            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                if (response.S3Objects != null)
                {
                    s3Objects.AddRange(response.S3Objects);
                }
            }
```

```
            if (printList)
            {
                Console.WriteLine($"Number of Objects: {s3Objects.Count}");
                foreach (var entry in s3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }

            return s3Objects;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server. Message:'{ex.Message}'
getting list of objects.");
            return null;
        }
    }



    /// <summary>
    /// Delete all of the objects stored in an existing Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket from which the
    /// contents will be deleted.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// deleting all of the objects in the bucket.</returns>
    public async Task<bool> DeleteBucketContentsAsync(string bucketName)
    {
        // Iterate over the contents of the bucket and delete all objects.
        try
        {
            // Delete all objects in the bucket.
            var deleteList = await ListBucketContentsAsync(bucketName, false);
            if (deleteList != null && deleteList.Any())
            {
                await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
                });
```

```
            }

            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Shows how to delete an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket to delete.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// the delete operation.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest { BucketName = bucketName, };

            await _amazonS3.DeleteBucketAsync(request);
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET .
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
```

```
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK para .NET da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
```

```
    {
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
            };

            var response = await _amazonS3.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK para .NET da API.

`CreatePresignedPost`

O código de exemplo a seguir mostra como usar `CreatePresignedPost`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um URL POST pré-assinado.

```
    /// <summary>
    /// Create a presigned POST URL with conditions.
    /// </summary>
    /// <param name="s3Client">The Amazon S3 client.</param>
```

```
    /// <param name="bucketName">The name of the bucket.</param>
    /// <param name="objectKey">The object key (path) where the uploaded file will
 be stored.</param>
    /// <param name="expires">When the presigned URL expires.</param>
    /// <param name="fields">Dictionary of fields to add to the form.</param>
    /// <param name="conditions">List of conditions to apply.</param>
    /// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
    public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
        IAmazonS3 s3Client,
        string bucketName,
        string objectKey,
        DateTime expires,
        Dictionary<string, string>? fields = null,
        List<S3PostCondition>? conditions = null)
    {
        var request = new CreatePresignedPostRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Expires = expires
        };

        // Add custom fields if provided
        if (fields != null)
        {
            foreach (var field in fields)
            {
                request.Fields.Add(field.Key, field.Value);
            }
        }

        // Add conditions if provided
        if (conditions != null)
        {
            foreach (var condition in conditions)
            {
                request.Conditions.Add(condition);
            }
        }

        return await s3Client.CreatePresignedPostAsync(request);
    }
```

- Para obter detalhes da API, consulte CreatePresignedPost em Referência AWS SDK para .NET da API.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
```

- Para obter detalhes da API, consulte DeleteBucket em Referência AWS SDK para .NET da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Delete all of the objects stored in an existing Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket from which the
    /// contents will be deleted.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// deleting all of the objects in the bucket.</returns>
    public async Task<bool> DeleteBucketContentsAsync(string bucketName)
    {
        // Iterate over the contents of the bucket and delete all objects.
        try
        {
            // Delete all objects in the bucket.
            var deleteList = await ListBucketContentsAsync(bucketName, false);
            if (deleteList != null && deleteList.Any())
            {
                await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
                });
            }

            return true;
        }
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK para .NET da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
```

```
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await _amazonS3.GetObjectAsync(request);

        try
        {
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK para .NET da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
```

```
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
    {
        try
        {
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
                MaxKeys = 5,
            };

            if (printList)
            {
                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");
            }

            var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });
            var s3Objects = new List<S3Object>();
            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                if (response.S3Objects != null)
                {
                    s3Objects.AddRange(response.S3Objects);
                }
            }

            if (printList)
            {
                Console.WriteLine($"Number of Objects: {s3Objects.Count}");
                foreach (var entry in s3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
```

```
            return s3Objects;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server. Message:'{ex.Message}'
getting list of objects.");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK para .NET API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
```

```
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK para .NET da API.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para .NET (v4)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie e use URLs POST pré-assinados para uploads diretos do navegador.

```
/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
    public static string? _bucketName;
    public static string? _objectKey;

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host">The IHost instance.</param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
        _httpClientFactory = host.Services.GetRequiredService<IHttpClientFactory>();
        _uiMethods = new UiMethods();
    }

    /// <summary>
    /// Perform the actions defined for the Amazon S3 Presigned POST scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        _isInteractive = !args.Contains("--non-interactive");
```

```
        // Set up dependency injection for Amazon S3
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddHttpClient()
            )
            .Build();

        SetUpServices(host);

        try
        {
            // Display overview
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(_isInteractive);

            // Step 1: Create bucket
            await CreateBucketAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 2: Create presigned URL
            _uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
            var response = await CreatePresignedPostAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 3: Display URL and fields
            _uiMethods.DisplayTitle("Step 3: Presigned POST URL details");
            DisplayPresignedPostFields(response);
            _uiMethods.PressEnter(_isInteractive);

            // Step 4: Upload file
            _uiMethods.DisplayTitle("Step 4: Upload test file using presigned POST
URL");
            await UploadFileAsync(response);
            _uiMethods.PressEnter(_isInteractive);

            // Step 5: Verify file exists
            await VerifyFileExistsAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 6: Cleanup
            _uiMethods.DisplayTitle("Step 6: Clean up resources");
```

```
            await CleanupAsync();

            _uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
 successfully!");
            _uiMethods.PressEnter(_isInteractive);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error in scenario");
            Console.WriteLine($"Error: {ex.Message}");

            // Attempt cleanup if there was an error
            if (!string.IsNullOrEmpty(_bucketName))
            {
                _uiMethods.DisplayTitle("Cleaning up resources after error");
                await _s3Wrapper.DeleteBucketAsync(_bucketName);
                Console.WriteLine($"Cleaned up bucket: {_bucketName}");
            }
        }
    }

    /// <summary>
    /// Create an S3 bucket for the scenario.
    /// </summary>
    private static async Task CreateBucketAsync()
    {
        _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

        // Generate a default bucket name for the scenario
        var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

        // Prompt user for bucket name or use default in non-interactive mode
        _bucketName = _uiMethods.GetUserInput(
            $"Enter S3 bucket name (or press Enter for '{defaultBucketName}'): ",
            defaultBucketName,
            _isInteractive);

        // Basic validation to ensure bucket name is not empty
        if (string.IsNullOrWhiteSpace(_bucketName))
        {
            _bucketName = defaultBucketName;
        }
```

```
        Console.WriteLine($"Creating bucket: {_bucketName}");

        await _s3Wrapper.CreateBucketAsync(_bucketName);

        Console.WriteLine($"Successfully created bucket: {_bucketName}");
    }


    /// <summary>
    /// Create a presigned POST URL.
    /// </summary>
    private static async Task<CreatePresignedPostResponse>
 CreatePresignedPostAsync()
    {
        _objectKey = "example-upload.txt";
        var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for the
 demo

        Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
        Console.WriteLine($"Expiration: {expiration} UTC");

        var s3Client = _s3Wrapper.GetS3Client();

        var response = await _s3Wrapper.CreatePresignedPostAsync(
            s3Client, _bucketName!, _objectKey, expiration);

        Console.WriteLine("Successfully created presigned POST URL");
        return response;
    }

    /// <summary>
    /// Upload a file using the presigned POST URL.
    /// </summary>
    private static async Task UploadFileAsync(CreatePresignedPostResponse response)
    {

        // Create a temporary test file to upload
        string testFilePath = Path.GetTempFileName();
        string testContent = "This is a test file for the S3 presigned POST
 scenario.";

        await File.WriteAllTextAsync(testFilePath, testContent);
        Console.WriteLine($"Created test file at: {testFilePath}");
```

```
        // Upload the file using the presigned POST URL
        Console.WriteLine("\nUploading file using the presigned POST URL...");
        var uploadResult = await UploadFileWithPresignedPostAsync(response,
testFilePath);

        // Display the upload result
        if (uploadResult.Success)
        {
            Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
        }
        else
        {
            Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
            Console.WriteLine($"Error: {uploadResult.Response}");
            throw new Exception("File upload failed");
        }

        // Clean up the temporary file
        File.Delete(testFilePath);
        Console.WriteLine("Temporary file deleted");
    }

    /// <summary>
    /// Helper method to upload a file using a presigned POST URL.
    /// </summary>
    private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
        CreatePresignedPostResponse response,
        string filePath)
    {
        try
        {
            _logger.LogInformation("Uploading file {filePath} using presigned POST
URL", filePath);

            using var httpClient = _httpClientFactory.CreateClient();
            using var formContent = new MultipartFormDataContent();

            // Add all the fields from the presigned POST response
            foreach (var field in response.Fields)
            {
```

```
                formContent.Add(new StringContent(field.Value), field.Key);
            }

            // Add the file content
            var fileStream = File.OpenRead(filePath);
            var fileName = Path.GetFileName(filePath);
            var fileContent = new StreamContent(fileStream);
            fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
            formContent.Add(fileContent, "file", fileName);

            // Send the POST request
            var httpResponse = await httpClient.PostAsync(response.Url,
 formContent);
            var responseContent = await httpResponse.Content.ReadAsStringAsync();

            // Log and return the result
            _logger.LogInformation("Upload completed with status code {statusCode}",
 httpResponse.StatusCode);

            return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
 responseContent);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error uploading file");
            return (false, HttpStatusCode.InternalServerError, ex.Message);
        }
    }

    /// <summary>
    /// Verify that the uploaded file exists in the S3 bucket.
    /// </summary>
    private static async Task VerifyFileExistsAsync()
    {
        _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

        Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

        try
        {
            var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
 _objectKey!);
```

```
            Console.WriteLine($"File verification successful! File exists in the
bucket.");
            Console.WriteLine($"File size: {metadata.ContentLength} bytes");
            Console.WriteLine($"File type: {metadata.Headers.ContentType}");
            Console.WriteLine($"Last modified: {metadata.LastModified}");
        }
        catch (AmazonS3Exception ex) when (ex.StatusCode ==
System.Net.HttpStatusCode.NotFound)
        {
            Console.WriteLine($"Error: File was not found in the bucket.");
            throw;
        }
    }

    private static void DisplayPresignedPostFields(CreatePresignedPostResponse
response)
    {
        Console.WriteLine($"Presigned POST URL: {response.Url}");
        Console.WriteLine("Form fields to include:");

        foreach (var field in response.Fields)
        {
            Console.WriteLine($"  {field.Key}: {field.Value}");
        }
    }

    /// <summary>
    /// Clean up resources created by the scenario.
    /// </summary>
    private static async Task CleanupAsync()
    {
        if (!string.IsNullOrEmpty(_bucketName))
        {
            Console.WriteLine($"Deleting bucket {_bucketName} and its contents...");
            bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

            if (result)
            {
                Console.WriteLine("Bucket deleted successfully");
            }
            else
            {
```

```
                Console.WriteLine("Failed to delete bucket - it may have been
 already deleted");
            }
        }
    }
}
```

Exemplos de código para AWS CLI com script Bash

Os exemplos de código a seguir mostram como usar o script AWS Command Line Interface with Bash com AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- AWS CLI com o Guia do Desenvolvedor Bash script : saiba mais sobre como usar o Bash com a AWS.
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos de API Gateway usando AWS CLI com script Bash
- Exemplos de Athena usando AWS CLI com script Bash

- AWS Batch exemplos usando AWS CLI com script Bash
- AWS Cloud Map exemplos usando AWS CLI com script Bash
- CloudFormation exemplos usando AWS CLI com script Bash
- CloudFront exemplos usando AWS CLI com script Bash
- CloudWatch exemplos usando AWS CLI com script Bash
- CloudWatch Exemplos de registros usando AWS CLI com script Bash
- Exemplos do Amazon Cognito Identity Provider usando AWS CLI com script Bash
- AWS Config exemplos usando AWS CLI com script Bash
- Exemplos de clientes do Connect usando AWS CLI com script Bash
- Direct Connect exemplos usando AWS CLI com script Bash
- Exemplos do Amazon DocumentDB usando AWS CLI com script Bash
- Exemplos do DynamoDB usando AWS CLI com script Bash
- Exemplos do Amazon EC2 usando AWS CLI com script Bash
- Exemplos do Amazon ECR usando AWS CLI com script Bash
- Exemplos do Amazon ECS usando AWS CLI com script Bash
- Exemplos do Amazon EKS usando AWS CLI com script Bash
- Elastic Load Balancing - exemplos da versão 2 usando AWS CLI com script Bash
- ElastiCache exemplos usando AWS CLI com script Bash
- Exemplos do Amazon EMR usando AWS CLI com script Bash
- OpenSearch Exemplos de serviços usando AWS CLI com script Bash
- AWS FIS exemplos usando AWS CLI com script Bash
- AWS Glue exemplos usando AWS CLI com script Bash
- HealthImaging exemplos usando AWS CLI com script Bash
- Exemplos de IAM usando AWS CLI com script Bash
- AWS IoT exemplos usando AWS CLI com script Bash
- AWS KMS exemplos usando AWS CLI com script Bash
- Exemplos de lambda usando AWS CLI com script Bash
- Exemplos de uso do Lightsail AWS CLI com script Bash
- Exemplos do Amazon Managed Grafana usando AWS CLI com script Bash
- MediaConnect exemplos usando AWS CLI com script Bash

- Exemplos de uso do Amazon MSK AWS CLI com script Bash
- Exemplos de Netuno usando AWS CLI com script Bash
- AWS Payment Cryptography exemplos usando AWS CLI com script Bash
- Exemplos do Amazon Pinpoint usando AWS CLI com script Bash
- Exemplos de uso do Amazon Polly AWS CLI com script Bash
- Exemplos do Amazon RDS usando AWS CLI com script Bash
- Exemplos do Amazon Redshift usando AWS CLI com script Bash
- Exemplos do Amazon S3 usando AWS CLI com script Bash
- SageMaker Exemplos de IA usando AWS CLI com script Bash
- Exemplos do Secrets Manager usando AWS CLI com script Bash
- Exemplos do Amazon SES usando AWS CLI com script Bash
- Exemplos do Amazon SNS usando AWS CLI com script Bash
- Exemplos de funções de etapa usando AWS CLI com script Bash
- AWS STS exemplos usando AWS CLI com script Bash
- Suporte exemplos usando AWS CLI com script Bash
- Exemplos do Systems Manager usando AWS CLI com script Bash
- Exemplos do Amazon Textract usando AWS CLI com script Bash
- Exemplos de VPC Lattice usando AWS CLI com script Bash
- AWS WAFV2 exemplos usando AWS CLI com script Bash
- WorkSpaces exemplos usando AWS CLI com script Bash

Exemplos de API Gateway usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o API Gateway.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma API REST com integração de proxy de função

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para execução do Lambda.
- Criar e implantar uma função do Lambda.
- Criar uma API REST
- Configurar a integração de proxy do Lambda.
- Implantar e testar a API
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
```

```
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
```

```
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
```

```
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query "items[?
pathPart=='helloworld'].id" --output text)
```

```
# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
```

```
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed to
 delete API" >&2
```

```
# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning: Failed
 to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Exemplos de Athena usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Athena.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução à análise de consulta

O exemplo de código a seguir mostra como:

- Criar um bucket do S3 para resultados de consulta.
- Criar um banco de dados.
- Criar uma tabela.
- Executar uma consulta.
- Criar e usar consultas nomeadas.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Athena Getting Started Script
```

```
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}
```

```
# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] || [ ${#bucket_name}
 -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
```

```
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~ ^[a-
z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION" 2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi
```

```
    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
athena-gs}]'

    # Security: Enable S3 bucket encryption with KMS validation
    echo "Enabling default encryption on S3 bucket..."
    if ! aws s3api put-bucket-encryption \
        --bucket "$S3_BUCKET" \
        --server-side-encryption-configuration '{
            "Rules": [{
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }]
        }' 2>&1; then
        echo "Warning: Could not enable encryption on bucket"
    fi

    # Security: Block public access
    echo "Blocking public access to S3 bucket..."
    if ! aws s3api put-public-access-block \
        --bucket "$S3_BUCKET" \
        --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1; then
        echo "Warning: Could not block public access on bucket"
    fi

    # Security: Enable versioning for data protection
    echo "Enabling versioning on S3 bucket..."
    if ! aws s3api put-bucket-versioning \
        --bucket "$S3_BUCKET" \
        --versioning-configuration Status=Enabled 2>&1; then
        echo "Warning: Could not enable versioning on bucket"
    fi

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
```

```
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
 "$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
```

```
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.$TABLE_NAME (
  \`Date\` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  \"input.regex\" = \"^(?!#)([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s
+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+[^\\\
\(]+[\\\\(]([^\\\\;]+).*\\\\%20([^\\\\/]+)[\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
```

```
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
```

```
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null ||
 echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
 "$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"
```

```
# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/dev/
null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -d'"' -
f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"
```

```
# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null ||
 echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"
```

```
# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
```

```
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
"$NAMED_QUERY_ID" \
        --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                    echo "Warning: Table deletion failed with status: $QUERY_STATUS"
                    break
                fi
                echo "Table deletion in progress, status: $QUERY_STATUS"
```

```
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \
        --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi
```

```
    # Empty and delete S3 bucket (only if not shared)
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi

        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
        if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
        else
            echo "S3 bucket deleted successfully."
        fi
    else
        echo "Skipping S3 bucket deletion (shared resource)"
    fi

    # Security: Remove downloaded query results
    if [ -f "./query-results.csv" ]; then
        if command -v shred &>/dev/null; then
            shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - Cp
  - CreateNamedQuery
  - DeleteNamedQuery
  - GetNamedQuery

- GetQueryExecution
- GetQueryResults
- ListDatabases
- ListNamedQueries
- ListTableMetadata
- Mb
- Rb
- Rm
- StartQueryExecution

AWS Batch exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS Batch.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução ao processamento em lote e aos contêineres com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um perfil de execução do IAM.
- Criar um ambiente de computação.
- Criar uma fila de trabalhos.
- Criar uma definição de trabalho.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Batch Fargate Getting Started Script - Security Hardened Version
# This script demonstrates creating AWS Batch resources with Fargate orchestration
#

set -euo pipefail  # Exit on any error, undefined variables, and pipe failures

# Configuration
SCRIPT_NAME="batch-fargate-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
RANDOM_SUFFIX=$(openssl rand -hex 6)
COMPUTE_ENV_NAME="batch-fargate-compute-${RANDOM_SUFFIX}"
JOB_QUEUE_NAME="batch-fargate-queue-${RANDOM_SUFFIX}"
JOB_DEF_NAME="batch-fargate-jobdef-${RANDOM_SUFFIX}"
JOB_NAME="batch-hello-world-${RANDOM_SUFFIX}"
ROLE_NAME="BatchEcsTaskExecutionRole-${RANDOM_SUFFIX}"
TRUST_POLICY_FILE="batch-trust-policy-${RANDOM_SUFFIX}.json"

# Security: Set restrictive umask
umask 0077

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Logging function with sanitization
log() {
    local message="${1//[$'\t\r\n']/}"
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] ${message}" | tee -a "${LOG_FILE}"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
```

```
    log "Attempting to clean up resources created so far..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error ${LINENO}' ERR

# Validate AWS credentials
validate_aws_credentials() {
    if ! aws sts get-caller-identity &>/dev/null; then
        log "ERROR: AWS credentials are not configured or invalid"
        exit 1
    fi
}

# Function to wait for resource to be ready
wait_for_compute_env() {
    local env_name="${1}"
    local max_attempts=60
    local attempt=0

    log "Waiting for compute environment ${env_name} to be VALID..."

    while [ ${attempt} -lt ${max_attempts} ]; do
        local status
        status=$(aws batch describe-compute-environments \
            --compute-environments "${env_name}" \
            --query 'computeEnvironments[0].status' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${status}" = "VALID" ]; then
            log "Compute environment ${env_name} is ready"
            return 0
        elif [ "${status}" = "INVALID" ] || [ "${status}" = "NOT_FOUND" ]; then
            log "ERROR: Compute environment ${env_name} failed to create properly"
            return 1
        fi

        log "Compute environment status: ${status}. Waiting 10 seconds..."
        sleep 10
        ((attempt++))
    done
```

```
    log "ERROR: Timeout waiting for compute environment ${env_name}"
    return 1
}

# Function to wait for job queue to be ready
wait_for_job_queue() {
    local queue_name="${1}"
    local max_attempts=60
    local attempt=0

    log "Waiting for job queue ${queue_name} to be VALID..."

    while [ ${attempt} -lt ${max_attempts} ]; do
        local state
        state=$(aws batch describe-job-queues \
            --job-queues "${queue_name}" \
            --query 'jobQueues[0].state' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${state}" = "ENABLED" ]; then
            log "Job queue ${queue_name} is ready"
            return 0
        elif [ "${state}" = "DISABLED" ] || [ "${state}" = "NOT_FOUND" ]; then
            log "ERROR: Job queue ${queue_name} failed to create properly"
            return 1
        fi

        log "Job queue state: ${state}. Waiting 10 seconds..."
        sleep 10
        ((attempt++))
    done

    log "ERROR: Timeout waiting for job queue ${queue_name}"
    return 1
}

# Function to wait for job completion
wait_for_job() {
    local job_id="${1}"
    local max_attempts=120
    local attempt=0

    log "Waiting for job ${job_id} to complete..."
```

```
    while [ ${attempt} -lt ${max_attempts} ]; do
        local status
        status=$(aws batch describe-jobs \
            --jobs "${job_id}" \
            --query 'jobs[0].status' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${status}" = "SUCCEEDED" ]; then
            log "Job ${job_id} completed successfully"
            return 0
        elif [ "${status}" = "FAILED" ]; then
            log "ERROR: Job ${job_id} failed"
            return 1
        fi

        log "Job status: ${status}. Waiting 30 seconds..."
        sleep 30
        ((attempt++))
    done

    log "ERROR: Timeout waiting for job ${job_id}"
    return 1
}

# Function to wait for resource state before deletion
wait_for_resource_state() {
    local resource_type="${1}"
    local resource_name="${2}"
    local expected_state="${3}"
    local max_attempts=30
    local attempt=0

    log "Waiting for ${resource_type} ${resource_name} to reach state:
 ${expected_state}"

    while [ ${attempt} -lt ${max_attempts} ]; do
        local current_state=""

        case "${resource_type}" in
            "JOB_QUEUE")
                current_state=$(aws batch describe-job-queues \
                    --job-queues "${resource_name}" \
                    --query 'jobQueues[0].state' \
                    --output text 2>/dev/null || echo "NOT_FOUND")
```

```
                ;;
            "COMPUTE_ENV")
                current_state=$(aws batch describe-compute-environments \
                    --compute-environments "${resource_name}" \
                    --query 'computeEnvironments[0].status' \
                    --output text 2>/dev/null || echo "NOT_FOUND")
                ;;
            *)
                log "WARNING: Unknown resource type: ${resource_type}"
                return 1
                ;;
        esac

        if [ "${current_state}" = "${expected_state}" ]; then
            log "${resource_type} ${resource_name} is now in state:
 ${expected_state}"
            return 0
        fi

        log "${resource_type} ${resource_name} state: ${current_state} (waiting for
 ${expected_state})"
        sleep 10
        ((attempt++))
    done

    log "WARNING: ${resource_type} ${resource_name} did not reach expected state
 after ${max_attempts} attempts"
    return 1
}

# Cleanup function
cleanup_resources() {
    log "Starting cleanup of created resources..."

    # Clean up in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local resource="${CREATED_RESOURCES[i]}"
        local resource_type
        local resource_name

        resource_type=$(echo "${resource}" | cut -d: -f1)
        resource_name=$(echo "${resource}" | cut -d: -f2-)

        log "Cleaning up ${resource_type}: ${resource_name}"
```

```
        case "${resource_type}" in
            "JOB_QUEUE")
                aws batch update-job-queue --job-queue "${resource_name}" --state
 DISABLED 2>/dev/null || true
                wait_for_resource_state "JOB_QUEUE" "${resource_name}" "DISABLED" ||
 true
                aws batch delete-job-queue --job-queue "${resource_name}" 2>/dev/
null || true
                ;;
            "COMPUTE_ENV")
                aws batch update-compute-environment --compute-environment
 "${resource_name}" --state DISABLED 2>/dev/null || true
                wait_for_resource_state "COMPUTE_ENV" "${resource_name}" "DISABLED"
 || true
                aws batch delete-compute-environment --compute-environment
 "${resource_name}" 2>/dev/null || true
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "${resource_name}" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/
null || true
                aws iam delete-role --role-name "${resource_name}" 2>/dev/null ||
 true
                ;;
            "FILE")
                rm -f "${resource_name}" 2>/dev/null || true
                ;;
            *)
                log "WARNING: Unknown resource type for cleanup: ${resource_type}"
                ;;
        esac
    done

    log "Cleanup completed"
}

# Validate input parameters
validate_inputs() {
    if [ -z "${ACCOUNT_ID:-}" ]; then
        log "ERROR: ACCOUNT_ID is not set"
        return 1
    fi
```

```
    if [ -z "${DEFAULT_VPC:-}" ]; then
        log "ERROR: DEFAULT_VPC is not set"
        return 1
    fi

    if [ -z "${SUBNETS:-}" ]; then
        log "ERROR: SUBNETS is not set"
        return 1
    fi
}

# Validate container image format
validate_container_image() {
    local image="${1}"

    # Check if image contains any shell metacharacters that could be dangerous
    if [[ "${image}" =~ [';$`|&<>()[]{}\\'] ]]; then
        log "ERROR: Container image contains potentially dangerous characters:
 ${image}"
        return 1
    fi

    # Basic ECR/Docker image format validation
    if ! [[ "${image}" =~ ^[a-zA-Z0-9._/:-]+$ ]]; then
        log "ERROR: Container image format is invalid: ${image}"
        return 1
    fi

    return 0
}

# Main script execution
main() {
    log "Starting AWS Batch Fargate tutorial script - Security Hardened Version"
    log "Log file: ${LOG_FILE}"

    # Validate AWS credentials
    validate_aws_credentials

    # Get AWS account ID
    log "Getting AWS account ID..."
    local ACCOUNT_ID
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    if [ -z "${ACCOUNT_ID}" ] || [ "${ACCOUNT_ID}" = "None" ]; then
```

```
        log "ERROR: Could not retrieve AWS account ID"
        exit 1
    fi
    log "Account ID: ${ACCOUNT_ID}"

    # Get default VPC and subnets
    log "Getting default VPC and subnets..."
    local DEFAULT_VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs \
        --filters "Name=is-default,Values=true" \
        --query 'Vpcs[0].VpcId' \
        --output text)

    if [ "${DEFAULT_VPC}" = "None" ] || [ "${DEFAULT_VPC}" = "null" ] || [ -z
 "${DEFAULT_VPC}" ]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi

    log "Default VPC: ${DEFAULT_VPC}"

    # Get subnets in the default VPC
    local SUBNETS
    SUBNETS=$(aws ec2 describe-subnets \
        --filters "Name=vpc-id,Values=${DEFAULT_VPC}" \
        --query 'Subnets[*].SubnetId' \
        --output text)

    if [ -z "${SUBNETS}" ]; then
        log "ERROR: No subnets found in default VPC"
        exit 1
    fi

    # Convert tab/space-separated subnets to JSON array format
    local SUBNET_ARRAY
    SUBNET_ARRAY=$(echo "${SUBNETS}" | tr '\t ' '\n' | sed 's/^/"/;s/$/"/' | paste -
sd ',' -)
    log "Subnets: ${SUBNETS}"
    log "Subnet array: [${SUBNET_ARRAY}]"

    # Get default security group for the VPC
    local DEFAULT_SG
    DEFAULT_SG=$(aws ec2 describe-security-groups \
```

```
        --filters "Name=vpc-id,Values=${DEFAULT_VPC}" "Name=group-
name,Values=default" \
        --query 'SecurityGroups[0].GroupId' \
        --output text)

    if [ "${DEFAULT_SG}" = "None" ] || [ "${DEFAULT_SG}" = "null" ] || [ -z
 "${DEFAULT_SG}" ]; then
        log "ERROR: No default security group found in VPC"
        exit 1
    fi

    log "Default security group: ${DEFAULT_SG}"

    # Step 1: Create IAM execution role
    log "Step 1: Creating IAM execution role..."

    # Create trust policy document with proper escaping
    cat > "${TRUST_POLICY_FILE}" << 'EOFPOLICY'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOFPOLICY
    CREATED_RESOURCES+=("FILE:${TRUST_POLICY_FILE}")

    # Validate trust policy file before using it
    if ! jq empty "${TRUST_POLICY_FILE}" 2>/dev/null; then
        log "ERROR: Trust policy file is not valid JSON"
        exit 1
    fi

    # Create the role
    aws iam create-role \
        --role-name "${ROLE_NAME}" \
        --assume-role-policy-document "file://${TRUST_POLICY_FILE}"
```

```
    aws iam tag-role --role-name "${ROLE_NAME}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=getting-started-batch-fargate
    CREATED_RESOURCES+=("IAM_ROLE:${ROLE_NAME}")

    # Attach policy
    aws iam attach-role-policy \
        --role-name "${ROLE_NAME}" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"

    log "IAM role created: ${ROLE_NAME}"

    # Wait for IAM role propagation
    log "Waiting for IAM role propagation (15 seconds)..."
    sleep 15

    # Step 2: Create compute environment
    log "Step 2: Creating Fargate compute environment..."

    aws batch create-compute-environment \
        --compute-environment-name "${COMPUTE_ENV_NAME}" \
        --type MANAGED \
        --state ENABLED \
        --tags project=doc-smith,tutorial=getting-started-batch-fargate \
        --compute-resources "{
            \"type\": \"FARGATE\",
            \"maxvCpus\": 256,
            \"subnets\": [${SUBNET_ARRAY}],
            \"securityGroupIds\": [\"${DEFAULT_SG}\"]
        }"
    CREATED_RESOURCES+=("COMPUTE_ENV:${COMPUTE_ENV_NAME}")

    # Wait for compute environment to be ready
    if ! wait_for_compute_env "${COMPUTE_ENV_NAME}"; then
        log "ERROR: Compute environment failed to reach VALID state"
        exit 1
    fi

    # Step 3: Create job queue
    log "Step 3: Creating job queue..."

    aws batch create-job-queue \
        --job-queue-name "${JOB_QUEUE_NAME}" \
        --state ENABLED \
```

```
        --priority 900 \
        --tags project=doc-smith,tutorial=getting-started-batch-fargate \
        --compute-environment-order "order=1,computeEnvironment=${COMPUTE_ENV_NAME}"
    CREATED_RESOURCES+=("JOB_QUEUE:${JOB_QUEUE_NAME}")

    # Wait for job queue to be ready
    if ! wait_for_job_queue "${JOB_QUEUE_NAME}"; then
        log "ERROR: Job queue failed to reach ENABLED state"
        exit 1
    fi

    # Step 4: Create job definition
    log "Step 4: Creating job definition..."

    local CONTAINER_IMAGE="busybox:latest"
    validate_container_image "${CONTAINER_IMAGE}"

    aws batch register-job-definition \
        --job-definition-name "${JOB_DEF_NAME}" \
        --type container \
        --platform-capabilities FARGATE \
        --container-properties "{
            \"image\": \"${CONTAINER_IMAGE}\",
            \"resourceRequirements\": [
                {\"type\": \"VCPU\", \"value\": \"0.25\"},
                {\"type\": \"MEMORY\", \"value\": \"512\"}
            ],
            \"command\": [\"echo\", \"hello world\"],
            \"networkConfiguration\": {
                \"assignPublicIp\": \"DISABLED\"
            },
            \"executionRoleArn\": \"arn:aws:iam::${ACCOUNT_ID}:role/${ROLE_NAME}\"
        }"

    log "Job definition created: ${JOB_DEF_NAME}"

    # Step 5: Submit job
    log "Step 5: Submitting job..."

    local JOB_ID
    JOB_ID=$(aws batch submit-job \
        --job-name "${JOB_NAME}" \
        --job-queue "${JOB_QUEUE_NAME}" \
        --job-definition "${JOB_DEF_NAME}" \
```

```
        --query 'jobId' \
        --output text)

    if [ -z "${JOB_ID}" ] || [ "${JOB_ID}" = "None" ]; then
        log "ERROR: Failed to submit job"
        exit 1
    fi

    log "Job submitted with ID: ${JOB_ID}"

    # Step 6: Wait for job completion and view output
    log "Step 6: Waiting for job completion..."
    if ! wait_for_job "${JOB_ID}"; then
        log "ERROR: Job failed or timed out"
        exit 1
    fi

    # Get log stream name
    log "Getting job logs..."
    local LOG_STREAM
    LOG_STREAM=$(aws batch describe-jobs \
        --jobs "${JOB_ID}" \
        --query 'jobs[0].attempts[0].taskProperties.containers[0].logStreamName' \
        --output text)

    if [ "${LOG_STREAM}" != "None" ] && [ "${LOG_STREAM}" != "null" ] && [ -n
"${LOG_STREAM}" ]; then
        log "Log stream: ${LOG_STREAM}"
        log "Job output:"
        aws logs get-log-events \
            --log-group-name "/aws/batch/job" \
            --log-stream-name "${LOG_STREAM}" \
            --query 'events[*].message' \
            --output text 2>/dev/null | tee -a "${LOG_FILE}" || true
    else
        log "No log stream available for job"
    fi

    log "Tutorial completed successfully!"

    # Show created resources
    echo ""
    echo "==========================================="
    echo "CREATED RESOURCES"
```

```
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${resource}"
    done
    echo ""
    echo "==========================================="
    echo "CLEANUP"
    echo "==========================================="

    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - RegisterJobDefinition
  - SubmitJob
  - UpdateComputeEnvironment
  - UpdateJobQueue

AWS Cloud Map exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS Cloud Map.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
```

```
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"
```

```
    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
```

```
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
```

```
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
```

```
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
```

```
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
```

```
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
```

```
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF
```

```
    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
```

```
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}
```

```
# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
```

```
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
```

```
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources
```

```
    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace

- DeleteRole
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeClusters
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Descoberta de serviço

O exemplo de código a seguir mostra como:

- Criar um namespace de DNS público com a integração de zona hospedada do Route 53.
- Criar serviços detectáveis por meio de consultas ao DNS e chamadas de API.
- Registrar instâncias de serviço com diferentes configurações de descoberta.
- Descobrir serviços usando consultas ao DNS e a API do Cloud Map.
- Verificar a descoberta de serviços usando o comando dig e chamadas de API.
- Limpar os recursos na ordem correta (instâncias, serviços e namespace).

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Cloud Map Private Namespace Tutorial Script
# This script demonstrates how to use AWS Cloud Map for service discovery
# with DNS queries and API calls

# Exit on error
set -e

# Configuration
REGION="us-east-2"
NAMESPACE_NAME="cloudmap-tutorial.com"
LOG_FILE="cloudmap-tutorial.log"
CREATOR_REQUEST_ID=$(date +%s)

# Function to log messages
log() {
    local message="$1"
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $message" | tee -a "$LOG_FILE"
}

# Function to check operation status
check_operation() {
    local operation_id="$1"
    local status=""

    log "Checking operation status for $operation_id..."

    while [[ "$status" != "SUCCESS" ]]; do
        sleep 5
        status=$(aws servicediscovery get-operation \
            --operation-id "$operation_id" \
            --region "$REGION" \
            --query "Operation.Status" \
```

```
            --output text)

        log "Operation status: $status"

        if [[ "$status" == "FAIL" ]]; then
            log "Operation failed. Exiting."
            exit 1
        fi
    done

    log "Operation completed successfully."
}

# Function to clean up resources
cleanup() {
    log "Starting cleanup process..."

    if [[ -n "$FIRST_INSTANCE_ID" ]]; then
        log "Deregistering first service instance..."
        aws servicediscovery deregister-instance \
            --service-id "$PUBLIC_SERVICE_ID" \
            --instance-id "$FIRST_INSTANCE_ID" \
            --region "$REGION" || log "Failed to deregister first instance"
    fi

    if [[ -n "$SECOND_INSTANCE_ID" ]]; then
        log "Deregistering second service instance..."
        aws servicediscovery deregister-instance \
            --service-id "$BACKEND_SERVICE_ID" \
            --instance-id "$SECOND_INSTANCE_ID" \
            --region "$REGION" || log "Failed to deregister second instance"
    fi

    if [[ -n "$PUBLIC_SERVICE_ID" ]]; then
        log "Deleting public service..."
        aws servicediscovery delete-service \
            --id "$PUBLIC_SERVICE_ID" \
            --region "$REGION" || log "Failed to delete public service"
    fi

    if [[ -n "$BACKEND_SERVICE_ID" ]]; then
        log "Deleting backend service..."
        aws servicediscovery delete-service \
            --id "$BACKEND_SERVICE_ID" \
```

```
            --region "$REGION" || log "Failed to delete backend service"
    fi

    if [[ -n "$NAMESPACE_ID" ]]; then
        log "Deleting namespace..."
        aws servicediscovery delete-namespace \
            --id "$NAMESPACE_ID" \
            --region "$REGION" || log "Failed to delete namespace"
    fi

    log "Cleanup completed."
}

# Set up trap for cleanup on script exit
trap cleanup EXIT INT TERM

# Initialize log file
> "$LOG_FILE"
log "Starting AWS Cloud Map tutorial script"

# Step 1: Create an AWS Cloud Map namespace
log "Creating AWS Cloud Map namespace: $NAMESPACE_NAME"
OPERATION_RESULT=$(aws servicediscovery create-public-dns-namespace \
    --name "$NAMESPACE_NAME" \
    --creator-request-id "cloudmap-tutorial-$CREATOR_REQUEST_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-discovery
 \
    --region "$REGION")

OPERATION_ID=$(echo "$OPERATION_RESULT" | jq -r '.OperationId')
log "Namespace creation initiated. Operation ID: $OPERATION_ID"

# Check operation status
check_operation "$OPERATION_ID"

# Get the namespace ID
log "Getting namespace ID..."
NAMESPACE_ID=$(aws servicediscovery list-namespaces \
    --region "$REGION" \
    --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" \
    --output text)

log "Namespace ID: $NAMESPACE_ID"
```

```
# Get the hosted zone ID
log "Getting Route 53 hosted zone ID..."
HOSTED_ZONE_ID=$(aws route53 list-hosted-zones-by-name \
    --dns-name "$NAMESPACE_NAME" \
    --query "HostedZones[0].Id" \
    --output text | sed 's|/hostedzone/||')

log "Hosted Zone ID: $HOSTED_ZONE_ID"

# Step 2: Create the AWS Cloud Map services
log "Creating public service..."
PUBLIC_SERVICE_RESULT=$(aws servicediscovery create-service \
    --name "public-service" \
    --namespace-id "$NAMESPACE_ID" \
    --dns-config "RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=A,TTL=300}]" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-discovery
 \
    --region "$REGION")

PUBLIC_SERVICE_ID=$(echo "$PUBLIC_SERVICE_RESULT" | jq -r '.Service.Id')
log "Public service created. Service ID: $PUBLIC_SERVICE_ID"

log "Creating backend service..."
BACKEND_SERVICE_RESULT=$(aws servicediscovery create-service \
    --name "backend-service" \
    --namespace-id "$NAMESPACE_ID" \
    --type "HTTP" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-discovery
 \
    --region "$REGION")

BACKEND_SERVICE_ID=$(echo "$BACKEND_SERVICE_RESULT" | jq -r '.Service.Id')
log "Backend service created. Service ID: $BACKEND_SERVICE_ID"

# Step 3: Register the AWS Cloud Map service instances
log "Registering first service instance..."
FIRST_INSTANCE_RESULT=$(aws servicediscovery register-instance \
    --service-id "$PUBLIC_SERVICE_ID" \
    --instance-id "first" \
    --attributes "AWS_INSTANCE_IPV4=192.168.2.1" \
    --region "$REGION")

FIRST_INSTANCE_ID="first"
FIRST_OPERATION_ID=$(echo "$FIRST_INSTANCE_RESULT" | jq -r '.OperationId')
```

```
log "First instance registration initiated. Operation ID: $FIRST_OPERATION_ID"

# Check operation status
check_operation "$FIRST_OPERATION_ID"

log "Registering second service instance..."
SECOND_INSTANCE_RESULT=$(aws servicediscovery register-instance \
    --service-id "$BACKEND_SERVICE_ID" \
    --instance-id "second" \
    --attributes "service-name=backend" \
    --region "$REGION")

SECOND_INSTANCE_ID="second"
SECOND_OPERATION_ID=$(echo "$SECOND_INSTANCE_RESULT" | jq -r '.OperationId')
log "Second instance registration initiated. Operation ID: $SECOND_OPERATION_ID"

# Check operation status
check_operation "$SECOND_OPERATION_ID"

# Step 4: Discover the AWS Cloud Map service instances
log "Getting Route 53 name servers..."
NAME_SERVERS=$(aws route53 get-hosted-zone \
    --id "$HOSTED_ZONE_ID" \
    --query "DelegationSet.NameServers[0]" \
    --output text)

log "Name server: $NAME_SERVERS"

log "Using dig to query DNS records (this will be simulated)..."
log "Command: dig @$NAME_SERVERS public-service.$NAMESPACE_NAME"
log "Expected output would show: public-service.$NAMESPACE_NAME. 300 IN A
 192.168.2.1"

log "Using AWS CLI to discover backend service instances..."
DISCOVER_RESULT=$(aws servicediscovery discover-instances \
    --namespace-name "$NAMESPACE_NAME" \
    --service-name "backend-service" \
    --region "$REGION")

log "Discovery result: $(echo "$DISCOVER_RESULT" | jq -c '.')"

# Display created resources
log "Resources created:"
log "- Namespace: $NAMESPACE_NAME (ID: $NAMESPACE_ID)"
```

```
log "- Public Service: public-service (ID: $PUBLIC_SERVICE_ID)"
log "- Backend Service: backend-service (ID: $BACKEND_SERVICE_ID)"
log "- Service Instance: first (Service: public-service)"
log "- Service Instance: second (Service: backend-service)"

# Ask user if they want to clean up resources
read -p "Do you want to clean up all created resources? (y/n): " CLEANUP_RESPONSE

if [[ "$CLEANUP_RESPONSE" == "y" || "$CLEANUP_RESPONSE" == "Y" ]]; then
    log "User confirmed cleanup. Proceeding with resource deletion."
    # Cleanup function will be called automatically on exit
else
    log "User chose not to clean up resources. Exiting without cleanup."
    trap - EXIT
    exit 0
fi
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreatePublicDnsNamespace
  - CreateService
  - DeleteNamespace
  - DeleteService
  - DeregisterInstance
  - DiscoverInstances
  - GetOperation
  - ListNamespaces
  - RegisterInstance

Atributos personalizados de descoberta de serviços

O exemplo de código a seguir mostra como:

- Criar um namespace do Cloud Map.
- Criar uma tabela do DynamoDB.
- Criar um serviço de dados do Cloud Map e registrar a tabela do DynamoDB.
- Criar um perfil do IAM para funções do Lambda.

- Criar a função do Lambda para gravar dados.
- Criar um serviço de aplicação do Cloud Map e registrar a função de gravação do Lambda.
- Criar a função do Lambda para ler dados.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Cloud Map Tutorial Script
# This script demonstrates how to use AWS Cloud Map for service discovery with
 custom attributes

set -euo pipefail

# Set up logging
LOG_FILE="cloudmap-tutorial.log"
echo "AWS Cloud Map Tutorial Script" > "$LOG_FILE"
echo "Started at $(date)" >> "$LOG_FILE"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log commands and their output
log_cmd() {
  echo "$ $1" | tee -a "$LOG_FILE"
  eval "$1" | tee -a "$LOG_FILE"
}

# Function to handle errors
handle_error() {
  local LINE=$1
  echo "An error occurred at line $LINE" | tee -a "$LOG_FILE"
  echo "Resources created so far:" | tee -a "$LOG_FILE"
  for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
```

```
  done
  echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"
  cleanup
  exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Helper function to wait for Cloud Map operations to complete
wait_for_operation() {
  local OPERATION_ID=$1
  local TIMEOUT=300  # 5 minutes timeout
  local START_TIME
  START_TIME=$(date +%s)

  while true; do
    local STATUS
    STATUS=$(aws servicediscovery get-operation --operation-id "$OPERATION_ID" --
query 'Operation.Status' --output text 2>/dev/null || echo "UNKNOWN")

    if [ "$STATUS" == "SUCCESS" ]; then
      echo "Operation completed successfully" | tee -a "$LOG_FILE"
      break
    elif [ "$STATUS" == "FAIL" ]; then
      echo "Operation failed" | tee -a "$LOG_FILE"
      return 1
    fi

    local CURRENT_TIME
    CURRENT_TIME=$(date +%s)
    if [ $((CURRENT_TIME - START_TIME)) -gt $TIMEOUT ]; then
      echo "Operation timed out" | tee -a "$LOG_FILE"
      return 1
    fi

    sleep 5
  done

  return 0
}

# Function to clean up resources
cleanup() {
```

```
 echo "Cleaning up resources..." | tee -a "$LOG_FILE"

 # Reverse the order of created resources for proper deletion
 for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
   resource="${CREATED_RESOURCES[$i]}"
   echo "Deleting $resource..." | tee -a "$LOG_FILE"

   if [[ $resource == "instance:"* ]]; then
     # Extract service ID and instance ID
     SERVICE_ID=$(echo "$resource" | cut -d':' -f2)
     INSTANCE_ID=$(echo "$resource" | cut -d':' -f3)

     # Check if instance exists before trying to deregister
     INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
"$SERVICE_ID" --query "Instances[?Id=='$INSTANCE_ID'].Id" --output text 2>/dev/null
|| echo "")
     if [[ -n "$INSTANCE_EXISTS" ]]; then
       OPERATION_ID=$(aws servicediscovery deregister-instance --service-id
"$SERVICE_ID" --instance-id "$INSTANCE_ID" --query 'OperationId' --output text)

       # Wait for deregistration to complete
       echo "Waiting for instance deregistration to complete..." | tee -a
"$LOG_FILE"
       wait_for_operation "$OPERATION_ID" || true
     else
       echo "Instance $INSTANCE_ID already deregistered" | tee -a "$LOG_FILE"
     fi
   elif [[ $resource == "lambda:"* ]]; then
     # Extract function name
     FUNCTION_NAME=$(echo "$resource" | cut -d':' -f2)
     aws lambda delete-function --function-name "$FUNCTION_NAME" 2>/dev/null ||
echo "Function already deleted" | tee -a "$LOG_FILE"
   elif [[ $resource == "role:"* ]]; then
     # Extract role name
     ROLE_NAME=$(echo "$resource" | cut -d':' -f2)

     # Detach all policies first
     for POLICY_ARN in $(aws iam list-attached-role-policies --role-name
"$ROLE_NAME" --query 'AttachedPolicies[*].PolicyArn' --output text); do
       aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$POLICY_ARN"
     done

     # Delete the role
```

```
      aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Role already
 deleted" | tee -a "$LOG_FILE"
    elif [[ $resource == "dynamodb:"* ]]; then
      # Extract table name
      TABLE_NAME=$(echo "$resource" | cut -d':' -f2)
      aws dynamodb delete-table --table-name "$TABLE_NAME" 2>/dev/null || echo
 "Table already deleted" | tee -a "$LOG_FILE"

      # Wait for table deletion to complete
      echo "Waiting for DynamoDB table deletion to complete..." | tee -a "$LOG_FILE"
      aws dynamodb wait table-not-exists --table-name "$TABLE_NAME" 2>/dev/null ||
 true
    fi
  done

  # Handle services separately to ensure all instances are deregistered first
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    if [[ $resource == "service:"* ]]; then
      # Extract service ID
      SERVICE_ID=$(echo "$resource" | cut -d':' -f2)
      echo "Deleting service $SERVICE_ID..." | tee -a "$LOG_FILE"

      # Make sure all instances are deregistered
      INSTANCES=$(aws servicediscovery list-instances --service-id "$SERVICE_ID" --
query 'Instances[*].Id' --output text 2>/dev/null || echo "")
      if [[ -n "$INSTANCES" ]]; then
        echo "Service still has instances. Waiting before deletion..." | tee -a
 "$LOG_FILE"
        sleep 10
      fi

      # Try to delete the service
      aws servicediscovery delete-service --id "$SERVICE_ID" 2>/dev/null || echo
 "Service already deleted" | tee -a "$LOG_FILE"
      sleep 5
    fi
  done

  # Handle namespaces last to ensure all services are deleted first
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    if [[ $resource == "namespace:"* ]]; then
      # Extract namespace ID
```

```
      NAMESPACE_ID=$(echo "$resource" | cut -d':' -f2)
      echo "Deleting namespace $NAMESPACE_ID..." | tee -a "$LOG_FILE"

      # Check if namespace still has services
      SERVICES=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query 'Services[*].Id' --
output text 2>/dev/null || echo "")
      if [[ -n "$SERVICES" ]]; then
        echo "Namespace still has services. Deleting them first..." | tee -a
 "$LOG_FILE"
        for SERVICE_ID in $SERVICES; do
          echo "Deleting service $SERVICE_ID..." | tee -a "$LOG_FILE"
          aws servicediscovery delete-service --id "$SERVICE_ID" 2>/dev/null || true
        done
        sleep 5
      fi

      # Try to delete the namespace
      OPERATION_ID=$(aws servicediscovery delete-namespace --id "$NAMESPACE_ID" --
query 'OperationId' --output text 2>/dev/null || echo "")
      if [[ -n "$OPERATION_ID" ]]; then
        echo "Waiting for namespace deletion to complete..." | tee -a "$LOG_FILE"
        wait_for_operation "$OPERATION_ID" || true
      else
        echo "Failed to delete namespace or namespace already deleted" | tee -a
 "$LOG_FILE"
      fi
    fi
  done

  echo "Cleanup complete" | tee -a "$LOG_FILE"
}

# Step 1: Create an AWS Cloud Map namespace
echo "Step 1: Creating AWS Cloud Map namespace..." | tee -a "$LOG_FILE"

# Check if namespace already exists
NAMESPACE_ID=$(aws servicediscovery list-namespaces --query "Namespaces[?
Name=='cloudmap-tutorial'].Id" --output text 2>/dev/null || echo "")

if [[ -z "$NAMESPACE_ID" || "$NAMESPACE_ID" == "None" ]]; then
  log_cmd "aws servicediscovery create-http-namespace --name cloudmap-tutorial --
creator-request-id namespace-request-\$(date +%s)"
```

```
  OPERATION_ID=$(aws servicediscovery create-http-namespace --name cloudmap-tutorial
 --creator-request-id "namespace-request-$(date +%s)" --tags Key=project,Value=doc-
smith Key=tutorial,Value=cloudmap-custom-attributes --query 'OperationId' --output
 text)

  # Wait for namespace creation to complete
  echo "Waiting for namespace creation to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"

  # Get the namespace ID
  NAMESPACE_ID=$(aws servicediscovery list-namespaces --query "Namespaces[?
Name=='cloudmap-tutorial'].Id" --output text)
  echo "Namespace created with ID: $NAMESPACE_ID" | tee -a "$LOG_FILE"
else
  echo "Namespace cloudmap-tutorial already exists with ID: $NAMESPACE_ID" | tee -a
 "$LOG_FILE"
fi

CREATED_RESOURCES+=("namespace:$NAMESPACE_ID")

# Step 2: Create a DynamoDB table
echo "Step 2: Creating DynamoDB table..." | tee -a "$LOG_FILE"

# Check if table already exists
TABLE_EXISTS=$(aws dynamodb describe-table --table-name cloudmap 2>&1 || echo
 "NOT_EXISTS")

if [[ $TABLE_EXISTS == *"ResourceNotFoundException"* || $TABLE_EXISTS ==
 "NOT_EXISTS" ]]; then
  log_cmd "aws dynamodb create-table --table-name cloudmap --attribute-definitions
 AttributeName=id,AttributeType=S --key-schema AttributeName=id,KeyType=HASH
 --billing-mode PAY_PER_REQUEST --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes"

  # Wait for DynamoDB table to become active
  echo "Waiting for DynamoDB table to become active..." | tee -a "$LOG_FILE"
  aws dynamodb wait table-exists --table-name cloudmap
else
  echo "DynamoDB table cloudmap already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("dynamodb:cloudmap")

# Step 3: Create an AWS Cloud Map data service
```

```
echo "Step 3: Creating AWS Cloud Map data service..." | tee -a "$LOG_FILE"

# Get all services in the namespace
echo "Listing all services in namespace $NAMESPACE_ID..." | tee -a "$LOG_FILE"
SERVICES=$(aws servicediscovery list-services --filters "Name=NAMESPACE_ID,Values=
$NAMESPACE_ID,Condition=EQ" --query 'Services[*].[Id,Name]' --output text 2>/dev/
null || echo "")
echo "Services found: $SERVICES" | tee -a "$LOG_FILE"

# Check if data service already exists
DATA_SERVICE_ID=""
while read -r id name || [[ -n "$id" ]]; do
  echo "Checking service: ID=$id, Name=$name" | tee -a "$LOG_FILE"
  if [[ "$name" == "data-service" ]]; then
    DATA_SERVICE_ID="$id"
    break
  fi
done <<< "$SERVICES"

if [[ -z "$DATA_SERVICE_ID" ]]; then
  echo "Data service does not exist, creating it..." | tee -a "$LOG_FILE"
  # Create the service and capture the ID directly
  echo "$ aws servicediscovery create-service --name data-service --namespace-id
 $NAMESPACE_ID --creator-request-id data-service-request-\$(date +%s)" | tee -a
 "$LOG_FILE"
  CREATE_OUTPUT=$(aws servicediscovery create-service --name data-service
 --namespace-id "$NAMESPACE_ID" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes --creator-request-id "data-service-
request-$(date +%s)")
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"

  # Extract the service ID using AWS CLI query
  DATA_SERVICE_ID=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query "Services[?
Name=='data-service'].Id" --output text)
  echo "Data service created with ID: $DATA_SERVICE_ID" | tee -a "$LOG_FILE"
else
  echo "Data service already exists with ID: $DATA_SERVICE_ID" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("service:$DATA_SERVICE_ID")

# Register DynamoDB table as a service instance
echo "Registering DynamoDB table as a service instance..." | tee -a "$LOG_FILE"
```

```
# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$DATA_SERVICE_ID" --query "Instances[?Id=='data-instance'].Id" --output text 2>/
dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  AWS_REGION=$(aws configure get region || echo "us-east-1")
  log_cmd "aws servicediscovery register-instance --service-id $DATA_SERVICE_ID --
instance-id data-instance --attributes tablename=cloudmap,region=$AWS_REGION"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$DATA_SERVICE_ID" --instance-id data-instance --attributes
 "tablename=cloudmap,region=$AWS_REGION" --query 'OperationId' --output text)

  # Wait for instance registration to complete
  echo "Waiting for instance registration to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance data-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$DATA_SERVICE_ID:data-instance")

# Step 4: Create an IAM role for Lambda
echo "Step 4: Creating IAM role for Lambda..." | tee -a "$LOG_FILE"

# Create a trust policy for Lambda
cat > lambda-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Check if role already exists
echo "Checking if IAM role already exists..." | tee -a "$LOG_FILE"
```

```
ROLE_EXISTS=$(aws iam get-role --role-name cloudmap-tutorial-role 2>&1 || echo
 "NOT_EXISTS")

if [[ $ROLE_EXISTS == *"NoSuchEntity"* || $ROLE_EXISTS == "NOT_EXISTS" ]]; then
    log_cmd "aws iam create-role --role-name cloudmap-tutorial-role --assume-role-
policy-document file://lambda-trust-policy.json"
    aws iam tag-role --role-name cloudmap-tutorial-role --tags
 Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-custom-attributes
else
    echo "Role cloudmap-tutorial-role already exists, using existing role" | tee -a
 "$LOG_FILE"
fi

# Create a custom policy with least privilege
cat > cloudmap-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:GetItem",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/cloudmap"
    },
    {
      "Effect": "Allow",
      "Action": [
        "servicediscovery:DiscoverInstances"
      ],
      "Resource": "*"
    }
  ]
```

```
}
EOF

# Check if policy already exists
POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='CloudMapTutorialPolicy'].Arn" --output text 2>/dev/null || echo "")

if [[ -z "$POLICY_ARN" ]]; then
  echo "Creating CloudMapTutorialPolicy..." | tee -a "$LOG_FILE"
  echo "$ aws iam create-policy --policy-name CloudMapTutorialPolicy --policy-
document file://cloudmap-policy.json" | tee -a "$LOG_FILE"
  CREATE_OUTPUT=$(aws iam create-policy --policy-name CloudMapTutorialPolicy --
policy-document file://cloudmap-policy.json)
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"
  POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='CloudMapTutorialPolicy'].Arn" --output text)
else
  echo "Policy CloudMapTutorialPolicy already exists with ARN: $POLICY_ARN" | tee -a
 "$LOG_FILE"
fi

echo "$ aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 $POLICY_ARN" | tee -a "$LOG_FILE"
aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 "$POLICY_ARN" | tee -a "$LOG_FILE"

echo "$ aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-
arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole" | tee -a
 "$LOG_FILE"
aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole | tee -a
 "$LOG_FILE"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

ROLE_ARN=$(aws iam get-role --role-name cloudmap-tutorial-role --query 'Role.Arn' --
output text)
CREATED_RESOURCES+=("role:cloudmap-tutorial-role")

# Step 5: Create an AWS Cloud Map app service
echo "Step 5: Creating AWS Cloud Map app service..." | tee -a "$LOG_FILE"
```

```
# Get all services in the namespace
SERVICES=$(aws servicediscovery list-services --filters "Name=NAMESPACE_ID,Values=
$NAMESPACE_ID,Condition=EQ" --query 'Services[*].[Id,Name]' --output text 2>/dev/
null || echo "")

# Check if app service already exists
APP_SERVICE_ID=""
while read -r id name || [[ -n "$id" ]]; do
  if [[ "$name" == "app-service" ]]; then
    APP_SERVICE_ID="$id"
    break
  fi
done <<< "$SERVICES"

if [[ -z "$APP_SERVICE_ID" ]]; then
  echo "App service does not exist, creating it..." | tee -a "$LOG_FILE"
  # Create the service and capture the ID directly
  echo "$ aws servicediscovery create-service --name app-service --namespace-id
 $NAMESPACE_ID --creator-request-id app-service-request-\$(date +%s)" | tee -a
 "$LOG_FILE"
  CREATE_OUTPUT=$(aws servicediscovery create-service --name app-service
 --namespace-id "$NAMESPACE_ID" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes --creator-request-id "app-service-
request-$(date +%s)")
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"

  # Extract the service ID using AWS CLI query
  APP_SERVICE_ID=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query "Services[?
Name=='app-service'].Id" --output text)
  echo "App service created with ID: $APP_SERVICE_ID" | tee -a "$LOG_FILE"
else
  echo "App service already exists with ID: $APP_SERVICE_ID" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("service:$APP_SERVICE_ID")

# Step 6: Create a Lambda function to write data
echo "Step 6: Creating Lambda function to write data..." | tee -a "$LOG_FILE"

# Create Lambda function code
cat > writefunction.py << 'EOF'
import boto3
import json
```

```
import random
import os

def lambda_handler(event, context):
    # Use AWS Cloud Map to discover the DynamoDB table
    serviceclient = boto3.client('servicediscovery')

    try:
        # Discover the data service instance
        response = serviceclient.discover_instances(
            NamespaceName='cloudmap-tutorial',
            ServiceName='data-service'
        )

        if not response.get('Instances'):
            return {
                'statusCode': 404,
                'body': json.dumps('No service instances found')
            }

        # Extract table name and region from the instance attributes
        attributes = response['Instances'][0].get('Attributes', {})
        tablename = attributes.get('tablename', 'cloudmap')
        region = attributes.get('region', os.environ.get('AWS_REGION', 'us-east-1'))

        # Create DynamoDB client in the specified region
        dynamodb = boto3.resource('dynamodb', region_name=region)
        table = dynamodb.Table(tablename)

        # Write data to the table
        table.put_item(
            Item={
                'id': str(random.randint(1, 100)),
                'todo': str(event)
            }
        )

        return {
            'statusCode': 200,
            'body': json.dumps('Data written successfully!')
        }
    except Exception as e:
        return {
            'statusCode': 500,
```

```
            'body': json.dumps(f'Error: {str(e)}')
        }
EOF

# Zip the function code
log_cmd "zip writefunction.zip writefunction.py"

# Create the Lambda function
FUNCTION_EXISTS=$(aws lambda list-functions --query "Functions[?
FunctionName=='writefunction'].FunctionName" --output text 2>/dev/null || echo "")
if [[ -z "$FUNCTION_EXISTS" ]]; then
  log_cmd "aws lambda create-function --function-name writefunction --runtime
 python3.12 --role $ROLE_ARN --handler writefunction.lambda_handler --zip-file
 fileb://writefunction.zip --architectures x86_64 --timeout 10 --tags project=doc-
smith,tutorial=cloudmap-custom-attributes"

  # Wait for the Lambda function to be active before updating
  echo "Waiting for Lambda function to become active..." | tee -a "$LOG_FILE"
  function_state="Pending"
  while [ "$function_state" == "Pending" ]; do
      sleep 5
      function_state=$(aws lambda get-function --function-name writefunction --query
 'Configuration.State' --output text)
      echo "Current function state: $function_state" | tee -a "$LOG_FILE"
  done
else
  echo "Lambda function writefunction already exists" | tee -a "$LOG_FILE"
fi
CREATED_RESOURCES+=("lambda:writefunction")

# Step 7: Register the Lambda write function as an AWS Cloud Map service instance
echo "Step 7: Registering Lambda write function as a service instance..." | tee -a
 "$LOG_FILE"

# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id "$APP_SERVICE_ID"
 --query "Instances[?Id=='write-instance'].Id" --output text 2>/dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  log_cmd "aws servicediscovery register-instance --service-id $APP_SERVICE_ID --
instance-id write-instance --attributes action=write,functionname=writefunction"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$APP_SERVICE_ID" --instance-id write-instance --attributes
 action=write,functionname=writefunction --query 'OperationId' --output text)
```

```
  # Wait for instance registration to complete
  echo "Waiting for write instance registration to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance write-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$APP_SERVICE_ID:write-instance")

# Step 8: Create a Lambda function to read data
echo "Step 8: Creating Lambda function to read data..." | tee -a "$LOG_FILE"

# Create Lambda function code
cat > readfunction.py << 'EOF'
import boto3
import json
import os

def lambda_handler(event, context):
    # Use AWS Cloud Map to discover the DynamoDB table
    serviceclient = boto3.client('servicediscovery')

    try:
        # Discover the data service instance
        response = serviceclient.discover_instances(
            NamespaceName='cloudmap-tutorial',
            ServiceName='data-service'
        )

        if not response.get('Instances'):
            return {
                'statusCode': 404,
                'body': json.dumps('No service instances found')
            }

        # Extract table name and region from the instance attributes
        attributes = response['Instances'][0].get('Attributes', {})
        tablename = attributes.get('tablename', 'cloudmap')
        region = attributes.get('region', os.environ.get('AWS_REGION', 'us-east-1'))

        # Create DynamoDB client in the specified region
        dynamodb = boto3.resource('dynamodb', region_name=region)
        table = dynamodb.Table(tablename)
```

```
        # Read data from the table
        response = table.scan()

        return {
            'statusCode': 200,
            'body': json.dumps(response.get('Items', []))
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps(f'Error: {str(e)}')
        }
EOF

# Zip the function code
log_cmd "zip readfunction.zip readfunction.py"

# Create the Lambda function
FUNCTION_EXISTS=$(aws lambda list-functions --query "Functions[?
FunctionName=='readfunction'].FunctionName" --output text 2>/dev/null || echo "")
if [[ -z "$FUNCTION_EXISTS" ]]; then
  log_cmd "aws lambda create-function --function-name readfunction --runtime
 python3.12 --role $ROLE_ARN --handler readfunction.lambda_handler --zip-file
 fileb://readfunction.zip --architectures x86_64 --timeout 10 --tags project=doc-
smith,tutorial=cloudmap-custom-attributes"

  # Wait for the Lambda function to be active before updating
  echo "Waiting for Lambda function to become active..." | tee -a "$LOG_FILE"
  function_state="Pending"
  while [ "$function_state" == "Pending" ]; do
      sleep 5
      function_state=$(aws lambda get-function --function-name readfunction --query
 'Configuration.State' --output text)
      echo "Current function state: $function_state" | tee -a "$LOG_FILE"
  done
else
  echo "Lambda function readfunction already exists" | tee -a "$LOG_FILE"
fi
CREATED_RESOURCES+=("lambda:readfunction")

# Step 9: Register the Lambda read function as an AWS Cloud Map service instance
echo "Step 9: Registering Lambda read function as a service instance..." | tee -a
 "$LOG_FILE"
```

```
# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id "$APP_SERVICE_ID"
 --query "Instances[?Id=='read-instance'].Id" --output text 2>/dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  log_cmd "aws servicediscovery register-instance --service-id $APP_SERVICE_ID --
instance-id read-instance --attributes action=read,functionname=readfunction"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$APP_SERVICE_ID" --instance-id read-instance --attributes
 action=read,functionname=readfunction --query 'OperationId' --output text)

  # Wait for read instance registration to complete
  echo "Waiting for read instance registration to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance read-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$APP_SERVICE_ID:read-instance")

# Step 10: Create Python clients to interact with the services
echo "Step 10: Creating Python clients..." | tee -a "$LOG_FILE"

cat > writeclient.py << 'EOF'
#!/usr/bin/env python3
import boto3
import json
import sys

try:
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='app-service',
        QueryParameters={'action': 'write'}
    )

    if not response.get('Instances'):
        print("No write service instances found", file=sys.stderr)
        sys.exit(1)

    functionname = response["Instances"][0]["Attributes"]["functionname"]
```

```
    lambdaclient = boto3.client('lambda')

    resp = lambdaclient.invoke(
        FunctionName=functionname,
        Payload=json.dumps('This is a test data'),
        InvocationType='RequestResponse'
    )

    print(resp["Payload"].read().decode())
except Exception as e:
    print(f"Error: {str(e)}", file=sys.stderr)
    sys.exit(1)
EOF

cat > readclient.py << 'EOF'
#!/usr/bin/env python3
import boto3
import json
import sys

try:
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='app-service',
        QueryParameters={'action': 'read'}
    )

    if not response.get('Instances'):
        print("No read service instances found", file=sys.stderr)
        sys.exit(1)

    functionname = response["Instances"][0]["Attributes"]["functionname"]

    lambdaclient = boto3.client('lambda')

    resp = lambdaclient.invoke(
        FunctionName=functionname,
        InvocationType='RequestResponse'
    )

    print(resp["Payload"].read().decode())
```

```
except Exception as e:
    print(f"Error: {str(e)}", file=sys.stderr)
    sys.exit(1)
EOF

chmod +x writeclient.py readclient.py

echo "Running write client..." | tee -a "$LOG_FILE"
log_cmd "python3 writeclient.py" || echo "Write client execution completed with
 warnings" | tee -a "$LOG_FILE"

echo "Running read client..." | tee -a "$LOG_FILE"
log_cmd "python3 readclient.py" || echo "Read client execution completed with
 warnings" | tee -a "$LOG_FILE"

# Step 11: Clean up resources
echo "Resources created:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]}"; do
  echo "- $resource" | tee -a "$LOG_FILE"
done

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"
cleanup

echo "Script completed at $(date)" | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateHttpNamespace
  - CreateService
  - DeleteNamespace
  - DeleteService
  - DeregisterInstance
  - GetOperation
  - ListNamespaces
  - ListServices
  - RegisterInstance

CloudFormation exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com CloudFormation.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criação da sua primeira pilha de infraestrutura

O exemplo de código a seguir mostra como:

- Crie um CloudFormation modelo
- Testar o servidor web.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudFormation Getting Started Script
# This script creates a CloudFormation stack with a web server and security group,
# monitors the stack creation, and provides cleanup options.

set -euo pipefail
```

```
# Set up logging with secure permissions
LOG_FILE="cloudformation-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS CloudFormation Getting Started Tutorial"
echo "====================================================="
echo "This script will create a CloudFormation stack with:"
echo "- An EC2 instance running a simple web server"
echo "- A security group allowing HTTP access from your IP"
echo ""
echo "Starting at: $(date)"
echo ""

# Function to validate IP address format
validate_ip() {
    local ip=$1
    if ! [[ "$ip" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        return 1
    fi

    local IFS=.
    local -a octets=($ip)
    for octet in "${octets[@]}"; do
        if ((octet > 255)); then
            return 1
        fi
    done
    return 0
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="

    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack: $STACK_NAME"
```

```
        aws cloudformation delete-stack --stack-name "$STACK_NAME" --region
 "${AWS_REGION:-us-east-1}" || true

        echo "Waiting for stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME" --
region "${AWS_REGION:-us-east-1}" 2>/dev/null || true

        echo "Stack deletion complete."
    fi

    if [ -f "${TEMPLATE_FILE:-}" ]; then
        echo "Removing local template file: $TEMPLATE_FILE"
        shred -vfz -n 3 "$TEMPLATE_FILE" 2>/dev/null || rm -f "$TEMPLATE_FILE"
    fi

    echo "Cleanup completed at: $(date)"
}

# Function to handle errors
handle_error() {
    echo ""
    echo "====================================================="
    echo "ERROR: $1"
    echo "====================================================="
    echo "Resources created before error:"
    if [ -n "${STACK_NAME:-}" ]; then
        echo "- CloudFormation stack: $STACK_NAME"
    fi
    echo ""

    echo "Cleaning up resources automatically..."
    cleanup

    exit 1
}

# Set up trap for script interruption
trap 'handle_error "Script interrupted"' INT TERM
trap 'cleanup' EXIT

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS_REGION format: $AWS_REGION"
```

```
fi

# Generate a unique stack name with timestamp
TIMESTAMP=$(date +%s)
STACK_NAME="MyTestStack-${TIMESTAMP}"
TEMPLATE_FILE="webserver-template-${TIMESTAMP}.yaml"

# Verify AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Verify curl is installed
if ! command -v curl &> /dev/null; then
    handle_error "curl is not installed or not in PATH"
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Step 1: Create the CloudFormation template file
echo "Creating CloudFormation template file: $TEMPLATE_FILE"
cat > "$TEMPLATE_FILE" << 'EOF'
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation Template for WebServer with Security Group and EC2
 Instance

Metadata:
  AWS::CloudFormation::Init:
    config:
      packages:
        yum:
          httpd: []
      services:
        sysvinit:
          httpd:
            enabled: true
            ensureRunning: true

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
```

```
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.

  MyIP:
    Description: Your IP address in CIDR format (e.g 203.0.113.1/32).
    Type: String
    MinLength: '9'
    MaxLength: '18'
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action: 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'

  WebServerInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
        - !Ref WebServerRole

  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP access via specified IP address
```

```
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
          Description: Allow HTTP outbound
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
          Description: Allow HTTPS outbound for package updates
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref MyIP
          Description: HTTP access from specified IP

  WebServer:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      IamInstanceProfile: !Ref WebServerInstanceProfile
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      Monitoring: true
      MetadataOptions:
        HttpEndpoint: enabled
        HttpTokens: required
        HttpPutResponseHopLimit: 1
      UserData: !Base64 |
        #!/bin/bash
        set -euo pipefail
        exec > >(tee /var/log/user-data.log)
        exec 2>&1
        yum update -y
        yum install -y httpd
        systemctl start httpd
        systemctl enable httpd
        echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html
        chmod 644 /var/www/html/index.html
```

```
Outputs:
  WebsiteURL:
    Value: !Join
      - ''
      - - http://
        - !GetAtt WebServer.PublicDnsName
    Description: Website URL
  InstanceId:
    Value: !Ref WebServer
    Description: EC2 Instance ID
  SecurityGroupId:
    Value: !Ref WebServerSecurityGroup
    Description: Security Group ID
EOF

chmod 600 "$TEMPLATE_FILE"

if [ ! -f "$TEMPLATE_FILE" ]; then
    handle_error "Failed to create template file"
fi

# Step 2: Validate the template
echo ""
echo "Validating CloudFormation template..."
if ! VALIDATION_RESULT=$(aws cloudformation validate-template \
    --template-body "file://$TEMPLATE_FILE" \
    --region "$AWS_REGION" 2>&1); then
    handle_error "Template validation failed: $VALIDATION_RESULT"
fi
echo "Template validation successful."

# Step 3: Get the user's public IP address
echo ""
echo "Retrieving your public IP address..."

MY_IP=""
for endpoint in "https://checkip.amazonaws.com" "https://api.ipify.org" "https://
icanhazip.com"; do
    if MY_IP=$(curl -s --max-time 5 "$endpoint" 2>/dev/null); then
        MY_IP="${MY_IP//[[:space:]]/}"
        if validate_ip "$MY_IP"; then
            break
        fi
        MY_IP=""
```

```
    fi
done

if [ -z "$MY_IP" ]; then
    handle_error "Failed to retrieve public IP address from multiple sources"
fi

MY_IP_CIDR="${MY_IP}/32"
echo "Your public IP address: $MY_IP_CIDR"

# Step 4: Create the CloudFormation stack
echo ""
echo "Creating CloudFormation stack: $STACK_NAME"
echo "This will create an EC2 instance and security group."
if ! CREATE_RESULT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-body "file://$TEMPLATE_FILE" \
  --parameters \
    ParameterKey=InstanceType,ParameterValue=t2.micro \
    ParameterKey=MyIP,ParameterValue="$MY_IP_CIDR" \
  --capabilities CAPABILITY_IAM \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudformation-gs \
  --region "$AWS_REGION" \
  --output text 2>&1); then
    handle_error "Stack creation failed: $CREATE_RESULT"
fi

STACK_ID="${CREATE_RESULT//[[:space:]]/}"
echo "Stack creation initiated. Stack ID: $STACK_ID"

# Step 5: Monitor stack creation
echo ""
echo "Monitoring stack creation..."
echo "This may take a few minutes."

if ! aws cloudformation wait stack-create-complete \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" 2>/dev/null; then
    STACK_STATUS=$(aws cloudformation describe-stacks \
        --stack-name "$STACK_NAME" \
        --query "Stacks[0].StackStatus" \
        --region "$AWS_REGION" \
        --output text 2>/dev/null || echo "UNKNOWN")
    if [[ "$STACK_STATUS" =~ ROLLBACK|FAILED ]]; then
```

```
        handle_error "Stack creation failed. Status: $STACK_STATUS"
    fi
fi

echo "Stack creation completed successfully."

# Step 6: List stack resources
echo ""
echo "Resources created by the stack:"
aws cloudformation list-stack-resources \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "StackResourceSummaries[*].{LogicalID:LogicalResourceId,
 Type:ResourceType, Status:ResourceStatus}" \
    --output table

# Step 7: Get stack outputs
echo ""
echo "Stack outputs:"
if ! OUTPUTS=$(aws cloudformation describe-stacks \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "Stacks[0].Outputs" \
    --output json 2>&1); then
    handle_error "Failed to retrieve stack outputs"
fi

echo "$OUTPUTS"

# Extract the WebsiteURL
WEBSITE_URL=$(aws cloudformation describe-stacks \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "Stacks[0].Outputs[?OutputKey=='WebsiteURL'].OutputValue" \
    --output text 2>/dev/null || echo "")

if [ -z "$WEBSITE_URL" ]; then
    handle_error "Failed to extract WebsiteURL from stack outputs"
fi

echo ""
echo "WebsiteURL: $WEBSITE_URL"
echo ""
echo "You can access the web server by opening the above URL in your browser."
```

```
echo "You should see a simple 'Hello World!' message."

# Step 8: Test the connection via CLI with timeout
echo ""
echo "Testing connection to the web server..."
MAX_RETRIES=5
RETRY_COUNT=0

while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
    if HTTP_RESPONSE=$(curl -s -o /dev/null -w "%{http_code}" --max-time 10
 "$WEBSITE_URL" 2>/dev/null); then
        if [ "$HTTP_RESPONSE" == "200" ]; then
            echo "Connection successful! HTTP status code: $HTTP_RESPONSE"
            break
        elif [ "$HTTP_RESPONSE" == "000" ]; then
            RETRY_COUNT=$((RETRY_COUNT + 1))
            if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
                echo "Connection test attempt $RETRY_COUNT/$MAX_RETRIES - retrying
 in 10 seconds..."
                sleep 10
            else
                echo "Warning: Connection test failed after $MAX_RETRIES attempts"
            fi
        else
            echo "Warning: Connection test returned HTTP status code:
 $HTTP_RESPONSE"
            break
        fi
    else
        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Connection test attempt $RETRY_COUNT/$MAX_RETRIES - retrying in 10
 seconds..."
            sleep 10
        else
            echo "Warning: Connection test failed after $MAX_RETRIES attempts"
        fi
    fi
done

# Step 9: Auto-confirm cleanup
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
```

```
echo "==================================================="
echo "Resources created:"
echo "- CloudFormation stack: $STACK_NAME"
echo "  - EC2 instance"
echo "  - Security group"
echo "  - IAM role and instance profile"
echo ""
echo "Proceeding with cleanup of all created resources..."

cleanup

echo ""
echo "==================================================="
echo "Tutorial completed at: $(date)"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateStack
  - DeleteStack
  - DescribeStacks
  - ListStackResources
  - ValidateTemplate
  - Wait

Conceitos básicos de clusters gerenciados de kubernetes

O exemplo de código a seguir mostra como:

- Criar uma VPC para um cluster do EKS.
- Criar perfis do IAM para um cluster do EKS.
- Criação do seu cluster de EKS
- Configurar o kubectl para se comunicar com o cluster.
- Crie um grupo de nós gerenciados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the AWS
 CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
```

```
        echo "To install kubectl, follow these instructions based on your operating
 system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/
kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
```

```
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --nodegroup-
name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
```

```
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources; exit
 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
```

```
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
```

```
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
```

```
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"
```

```
CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-kubeconfig
 --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
```

```
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn" --
output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register
```

```
# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl get
 nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or the
 nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl get
 all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
```

```
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --cluster-
name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --stack-
name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy

- CreateCluster
- CreateNodegroup
- CreateRole
- CreateStack
- DeleteCluster
- DeleteNodegroup
- DeleteRole
- DeleteStack
- DescribeCluster
- DescribeNodegroup
- DescribeStacks
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

CloudFront exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com CloudFront.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução a uma rede básica de distribuição de conteúdo

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Fazer upload do conteúdo no bucket.
- Crie uma CloudFront distribuição com o OAC
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudFront Getting Started Tutorial Script
# This script creates an S3 bucket, uploads sample content, creates a CloudFront
 distribution with OAC,
# and demonstrates how to access content through CloudFront.

set -euo pipefail

# Set up logging
LOG_FILE="cloudfront-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting CloudFront Getting Started Tutorial at $(date)"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created before error:"
    if [ -n "${BUCKET_NAME:-}" ]; then
        echo "- S3 Bucket: $BUCKET_NAME"
```

```
    fi
    if [ -n "${OAC_ID:-}" ]; then
        echo "- CloudFront Origin Access Control: $OAC_ID"
    fi
    if [ -n "${DISTRIBUTION_ID:-}" ]; then
        echo "- CloudFront Distribution: $DISTRIBUTION_ID"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "Cleaning up resources..."

    if [ -n "${DISTRIBUTION_ID:-}" ]; then
        echo "Disabling CloudFront distribution $DISTRIBUTION_ID..."

        # Get the current configuration and ETag in one call
        DIST_CONFIG=$(aws cloudfront get-distribution-config --id "$DISTRIBUTION_ID"
 2>/dev/null) || {
            echo "Failed to get distribution config. Continuing with cleanup..."
            DIST_CONFIG=""
        }

        if [ -n "$DIST_CONFIG" ]; then
            ETAG=$(echo "$DIST_CONFIG" | jq -r '.ETag')

            # Modify and update distribution in one pipeline
            if echo "$DIST_CONFIG" | jq '.DistributionConfig.Enabled = false' | \
                aws cloudfront update-distribution \
                    --id "$DISTRIBUTION_ID" \
                    --distribution-config "$(cat)" \
                    --if-match "$ETAG" 2>/dev/null; then

                echo "Waiting for distribution to be disabled (this may take several
 minutes)..."
                aws cloudfront wait distribution-deployed --id "$DISTRIBUTION_ID"
 2>/dev/null || {
                    echo "Distribution deployment wait timed out. Proceeding with
 deletion..."
                }
```

```
                # Get fresh ETag for deletion
                DIST_CONFIG=$(aws cloudfront get-distribution-config --id
 "$DISTRIBUTION_ID" 2>/dev/null) || {
                    echo "Failed to get updated config. Skipping distribution
 deletion..."
                    DIST_CONFIG=""
                }

                if [ -n "$DIST_CONFIG" ]; then
                    ETAG=$(echo "$DIST_CONFIG" | jq -r '.ETag')
                    aws cloudfront delete-distribution --id "$DISTRIBUTION_ID" --if-
match "$ETAG" 2>/dev/null && \
                        echo "CloudFront distribution deleted." || \
                        echo "Failed to delete distribution. You may need to delete
 it manually."
                fi
            else
                echo "Failed to disable distribution. Continuing with cleanup..."
            fi
        fi
    fi

    if [ -n "${OAC_ID:-}" ]; then
        echo "Deleting Origin Access Control $OAC_ID..."
        OAC_DATA=$(aws cloudfront get-origin-access-control --id "$OAC_ID" 2>/dev/
null) || {
            echo "Failed to get Origin Access Control. You may need to delete it
 manually."
            OAC_DATA=""
        }

        if [ -n "$OAC_DATA" ]; then
            OAC_ETAG=$(echo "$OAC_DATA" | jq -r '.ETag')
            aws cloudfront delete-origin-access-control --id "$OAC_ID" --if-match
 "$OAC_ETAG" 2>/dev/null && \
                echo "Origin Access Control deleted." || \
                echo "Failed to delete Origin Access Control. You may need to delete
 it manually."
        fi
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "$BUCKET_IS_SHARED" != "true" ]; then
        echo "Deleting S3 bucket $BUCKET_NAME and its contents..."
```

```
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || {
            echo "Failed to remove bucket contents. Continuing with bucket
 deletion..."
        }

        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null && \
            echo "S3 bucket deleted." || \
            echo "Failed to delete bucket. You may need to delete it manually."
    fi

    # Clean up temporary files
    rm -f temp_disabled_config.json distribution-config.json bucket-policy.json 2>/
dev/null || true
    rm -rf temp_content 2>/dev/null || true
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Validate AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Validate jq is available
if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi

# Validate AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Initialize variables
BUCKET_NAME=""
OAC_ID=""
DISTRIBUTION_ID=""
BUCKET_IS_SHARED=false

# Generate a random identifier for the bucket name using secure random
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
    handle_error "Failed to generate random identifier"
```

```
fi

# Check for shared prereq bucket and get account ID in parallel calls
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null) || PREREQ_BUCKET=""

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="cloudfront-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Get AWS account ID early
ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
if [ $? -ne 0 ]; then
    handle_error "Failed to get AWS account ID"
fi

# Validate account ID format
if ! [[ "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    handle_error "Invalid AWS account ID format: $ACCOUNT_ID"
fi

# Create a temporary directory for content with restrictive permissions
TEMP_DIR="temp_content"
mkdir -p "$TEMP_DIR/css"
chmod 700 "$TEMP_DIR"
if [ $? -ne 0 ]; then
    handle_error "Failed to create temporary directory"
fi

# Step 1: Create an S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" != "true" ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    aws s3 mb "s3://$BUCKET_NAME" --region us-east-1
    if [ $? -ne 0 ]; then
        handle_error "Failed to create S3 bucket"
    fi
```

```
    aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=cloudfront-
gettingstarted}]'

    # Batch bucket configuration calls for efficiency
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled &
    aws s3api put-public-access-block \
        --bucket "$BUCKET_NAME" \
        --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 &
    aws s3api put-bucket-encryption \
        --bucket "$BUCKET_NAME" \
        --server-side-encryption-configuration '{
            "Rules": [
                {
                    "ApplyServerSideEncryptionByDefault": {
                        "SSEAlgorithm": "AES256"
                    }
                }
            ]
        }' &
    wait
fi

# Step 2: Create sample content
echo "Creating sample content..."
cat > "$TEMP_DIR/index.html" << 'EOF'
<!DOCTYPE html>
<html>
<head>
    <title>Hello World</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>
EOF

cat > "$TEMP_DIR/css/styles.css" << 'EOF'
body {
```

```
    font-family: Arial, sans-serif;
    margin: 40px;
    background-color: #f5f5f5;
}
h1 {
    color: #333;
    text-align: center;
}
EOF

# Set restrictive permissions on content files
chmod 600 "$TEMP_DIR/index.html" "$TEMP_DIR/css/styles.css"

# Step 3: Upload content to the S3 bucket with encryption and metadata
echo "Uploading content to S3 bucket..."
aws s3 cp "$TEMP_DIR/" "s3://$BUCKET_NAME/" --recursive \
    --sse AES256 \
    --metadata "Source=CloudFrontTutorial"
if [ $? -ne 0 ]; then
    handle_error "Failed to upload content to S3 bucket"
fi

# Step 4: Create Origin Access Control
echo "Creating Origin Access Control..."
OAC_RESPONSE=$(aws cloudfront create-origin-access-control \
    --origin-access-control-config Name="oac-for-
$BUCKET_NAME",SigningProtocol=sigv4,SigningBehavior=always,OriginAccessControlOriginType=s3)

if [ $? -ne 0 ]; then
    handle_error "Failed to create Origin Access Control"
fi

OAC_ID=$(echo "$OAC_RESPONSE" | jq -r '.OriginAccessControl.Id')
if [ -z "$OAC_ID" ] || [ "$OAC_ID" = "null" ]; then
    handle_error "Failed to extract OAC ID from response"
fi

# Validate OAC ID format (alphanumeric and hyphens)
if ! [[ "$OAC_ID" =~ ^[A-Z0-9]+$ ]]; then
    handle_error "Invalid OAC ID format: $OAC_ID"
fi

echo "Created Origin Access Control with ID: $OAC_ID"
```

```
# Step 5: Create CloudFront distribution
echo "Creating CloudFront distribution..."

# Validate bucket name format
if ! [[ "$BUCKET_NAME" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]]; then
    handle_error "Invalid S3 bucket name format: $BUCKET_NAME"
fi

# Create distribution configuration with improved security settings
cat > distribution-config.json << EOF
{
    "CallerReference": "cli-tutorial-$(date +%s)",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "S3-$BUCKET_NAME",
                "DomainName": "$BUCKET_NAME.s3.amazonaws.com",
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                },
                "OriginAccessControlId": "$OAC_ID"
            }
        ]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "S3-$BUCKET_NAME",
        "ViewerProtocolPolicy": "redirect-to-https",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": ["GET", "HEAD"],
            "CachedMethods": {
                "Quantity": 2,
                "Items": ["GET", "HEAD"]
            }
        },
        "DefaultTTL": 86400,
        "MinTTL": 0,
        "MaxTTL": 31536000,
        "Compress": true,
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
```

```
            }
        }
    },
    "Comment": "CloudFront distribution for tutorial",
    "Enabled": true,
    "WebACLId": "",
    "HttpVersion": "http2and3"
}
EOF

# Set restrictive permissions on config file before passing credentials
chmod 600 distribution-config.json

DIST_RESPONSE=$(aws cloudfront create-distribution --distribution-config file://
distribution-config.json)
if [ $? -ne 0 ]; then
    handle_error "Failed to create CloudFront distribution"
fi

DISTRIBUTION_ID=$(echo "$DIST_RESPONSE" | jq -r '.Distribution.Id')
DOMAIN_NAME=$(echo "$DIST_RESPONSE" | jq -r '.Distribution.DomainName')

if [ -z "$DISTRIBUTION_ID" ] || [ "$DISTRIBUTION_ID" = "null" ] || [ -z
 "$DOMAIN_NAME" ] || [ "$DOMAIN_NAME" = "null" ]; then
    handle_error "Failed to extract distribution ID or domain name from response"
fi

# Validate distribution ID format
if ! [[ "$DISTRIBUTION_ID" =~ ^[A-Z0-9]+$ ]]; then
    handle_error "Invalid distribution ID format: $DISTRIBUTION_ID"
fi

echo "Created CloudFront distribution with ID: $DISTRIBUTION_ID"
echo "CloudFront domain name: $DOMAIN_NAME"

# Tag the CloudFront distribution
aws cloudfront tag-resource --resource "arn:aws:cloudfront::
$ACCOUNT_ID:distribution/$DISTRIBUTION_ID" --tags 'Items=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=cloudfront-gettingstarted}]'

# Step 6: Update S3 bucket policy
echo "Updating S3 bucket policy..."

cat > bucket-policy.json << EOF
```

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::$BUCKET_NAME/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn": "arn:aws:cloudfront::$ACCOUNT_ID:distribution/
$DISTRIBUTION_ID"
                }
            }
        }
    ]
}
EOF

# Set restrictive permissions on policy file
chmod 600 bucket-policy.json

aws s3api put-bucket-policy --bucket "$BUCKET_NAME" --policy file://bucket-
policy.json
if [ $? -ne 0 ]; then
    handle_error "Failed to update S3 bucket policy"
fi

# Step 7: Wait for distribution to deploy
echo "Waiting for CloudFront distribution to deploy (this may take 5-10 minutes)..."
aws cloudfront wait distribution-deployed --id "$DISTRIBUTION_ID" 2>/dev/null || {
    echo "Warning: Distribution deployment wait timed out. The distribution may
 still be deploying."
}

echo "CloudFront distribution is now deployed."

# Step 8: Display access information
echo ""
echo "===== CloudFront Distribution Setup Complete ====="
echo "You can access your content at: https://$DOMAIN_NAME/index.html"
```

```
echo ""
echo "Resources created:"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- CloudFront Origin Access Control: $OAC_ID"
echo "- CloudFront Distribution: $DISTRIBUTION_ID"
echo ""
echo "To clean up resources, run: cleanup"
echo ""

echo "Tutorial completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateDistribution
  - CreateOriginAccessControl
  - DeleteDistribution
  - DeleteOriginAccessControl
  - GetDistribution
  - GetDistributionConfig
  - GetOriginAccessControl
  - UpdateDistribution
  - WaitDistributionDeployed

Conceitos básicos de firewalls do aplicativo Web

O exemplo de código a seguir mostra como:

- Criar uma ACL da web.
- Adicionar uma regra de correspondência de string.
- Adicionar regras gerenciadas.
- Configurar o registro em log da
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
```

```
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
```

```
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/
dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the AWS
 Console."
                else
```

```
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
```

```
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "====================================================="
echo "STEP 2: Adding String Match Rule"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi
```

```
    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
|| true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
```

```
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true
```

```
    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
|| true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
```

```
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
```

```
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "====================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "====================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "====================================================="
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "====================================================="
```

```
    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo "$first_dist" |
 grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
```

```
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

CloudWatch exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com CloudWatch.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Como criar um painel de monitoramento com o nome da função como uma variável

O exemplo de código a seguir mostra como:

- Crie um CloudWatch painel
- Adicionar widgets de métricas do Lambda com uma variável de nome de função.
- Verificar o painel.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
```

```
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with 'aws
 configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
```

```
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi
```

```
    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-dynamicdash
 \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
```

```
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/null
 --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
```

```
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}", { "stat":
 "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
```

```
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
```

```
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
```

```
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of the
 dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
```

```
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --region
 "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to delete
 it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach it
 manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi
```

```
echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
```

```
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi
```

```
    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
```

```
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10
```

```
echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"
```

```
echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"
```

```
# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
```

```
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
```

```
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
```

```
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
```

```
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
```

```
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
"stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
```

```
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole

- DeleteAlarms
- DeleteExperimentTemplate
- DeleteInstanceProfile
- DeleteRole
- DeleteRolePolicy
- DescribeAlarms
- DescribeImages
- DetachRolePolicy
- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar funções do Lambda para monitoramento.
- Crie um CloudWatch painel
- Adicionar uma variável de propriedade ao painel.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
```

```
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
```

```
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/dev/
null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
```

```
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
```

```
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /dev/
null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /dev/
null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a "$LOG_FILE"

# Create dashboard body with proper escaping and validation
```

```
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name "$DASHBOARD_NAME"
 --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
 | tee -a "$LOG_FILE"
```

```
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME" --
output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
```

```
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

CloudWatch Exemplos de registros usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com CloudWatch Logs.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
```

```
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
```

```
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}
```

```
# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"
```

```
    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}
```

```
# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
```

```
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null
```

```
        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
```

```
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
```

```
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
```

```
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi
```

```
    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)
```

```
        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
```

```
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
```

```
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
```

```
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks

- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Como criar sua primeira função com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Lambda.
- Criar código da função.
- Criar uma função do Lambda
- Testar uma função do Lambda.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
```

```
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

###############################################################################
# Setup
###############################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

###############################################################################
# Helper functions
###############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"
```

```
        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could not
detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
```

```
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "==========================================="
    echo "ERROR: Script failed at $1"
    echo "==========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
```

```
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

################################################################################
# Runtime selection
################################################################################

echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
```

```
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !== 'number')
 {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
```

```
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

################################################################################
# Step 1: Create IAM execution role
################################################################################

echo ""
echo "==========================================="
echo "Step 1: Create IAM execution role"
echo "==========================================="
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi
```

```
echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/service-
role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
```

```
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --query
 State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
```

```
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

###############################################################################
# Step 4: View CloudWatch logs
###############################################################################

echo ""
echo "==========================================="
echo "Step 4: View CloudWatch Logs"
echo "==========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""
```

```
echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################
```

```
echo ""
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction
  - GetFunctionConfiguration
  - GetLogEvents
  - Invoke

Exemplos do Amazon Cognito Identity Provider usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon Cognito Identity Provider.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos dos grupos de usuários

O exemplo de código a seguir mostra como:

- Criar um grupo de usuários.
- Criar um cliente de aplicação.
- Configurar um domínio para o grupo de usuários.
- Criar um usuário como administrador.
- Habilitar o autorregistro.
- Listar usuários no grupo de usuários.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Cognito User Pools Getting Started Script
# This script creates and configures an Amazon Cognito user pool with an app client

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cognito-user-pool-setup.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Cognito User Pool setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -qi "error\|failed"; then
    echo "ERROR: Command failed: $cmd" >&2
    echo "Output: $output" >&2
    cleanup_on_error
    exit 1
  fi
}

# Function to check AWS CLI return code
check_aws_error() {
  local exit_code=$?
  local cmd=$1

  if [ $exit_code -ne 0 ]; then
    echo "ERROR: AWS CLI command failed with exit code $exit_code: $cmd" >&2
    cleanup_on_error
    exit "$exit_code"
  fi
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
  echo "Error encountered. Attempting to clean up resources..." >&2

  if [ -n "${DOMAIN_NAME:-}" ] && [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool domain: $DOMAIN_NAME" >&2
    aws cognito-idp delete-user-pool-domain \
      --user-pool-id "$USER_POOL_ID" \
      --domain "$DOMAIN_NAME" 2>/dev/null || true
  fi

  if [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool: $USER_POOL_ID" >&2
    aws cognito-idp delete-user-pool \
      --user-pool-id "$USER_POOL_ID" 2>/dev/null || true
  fi
}

# Set trap for cleanup on exit
trap cleanup_on_error EXIT ERR

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
  echo "ERROR: AWS CLI is not installed" >&2
  exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
  echo "ERROR: jq is not installed" >&2
  exit 1
fi

# Validate openssl is installed
if ! command -v openssl &> /dev/null; then
  echo "ERROR: openssl is not installed" >&2
  exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
  echo "ERROR: AWS CLI is not configured or credentials are invalid" >&2
  exit 1
fi
```

```
# Get the current AWS region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
  AWS_REGION="us-east-1" # Default region if not configured
fi
echo "Using AWS Region: $AWS_REGION"

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
  echo "ERROR: Invalid AWS region format: $AWS_REGION" >&2
  exit 1
fi

# Generate random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
  echo "ERROR: Failed to generate random identifier" >&2
  exit 1
fi

USER_POOL_NAME="MyUserPool-${RANDOM_ID}"
APP_CLIENT_NAME="MyAppClient-${RANDOM_ID}"
DOMAIN_NAME="my-auth-domain-${RANDOM_ID}"

# Validate resource names don't exceed limits
if [ ${#USER_POOL_NAME} -gt 128 ]; then
  echo "ERROR: User pool name exceeds maximum length of 128 characters" >&2
  exit 1
fi

if [ ${#APP_CLIENT_NAME} -gt 128 ]; then
  echo "ERROR: App client name exceeds maximum length of 128 characters" >&2
  exit 1
fi

echo "Using random identifier: $RANDOM_ID"
echo "User pool name: $USER_POOL_NAME"
echo "App client name: $APP_CLIENT_NAME"
echo "Domain name: $DOMAIN_NAME"

# Step 1: Create a User Pool
echo "Creating user pool..."
USER_POOL_OUTPUT=$(aws cognito-idp create-user-pool \
```

```
  --pool-name "$USER_POOL_NAME" \
  --auto-verified-attributes email \
  --username-attributes email \
  --policies '{"PasswordPolicy":
{"MinimumLength":12,"RequireUppercase":true,"RequireLowercase":true,"RequireNumbers":true,"R
 \
  --schema '[{"Name":"email","Required":true,"Mutable":true}]' \
  --mfa-configuration OFF \
  --user-attribute-update-settings '{"AttributesRequireVerificationBeforeUpdate":
["email"]}' \
  --account-recovery-setting 'RecoveryMechanisms=[{Name=verified_email,Priority=1}]'
 \
  --deletion-protection INACTIVE \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool"

# Extract the User Pool ID using jq for safety
USER_POOL_ID=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Id // empty')
if [ -z "$USER_POOL_ID" ]; then
  echo "ERROR: Failed to extract User Pool ID" >&2
  exit 1
fi

# Validate User Pool ID format
if ! [[ "$USER_POOL_ID" =~ ^[a-z]{2}-[a-z]+-[0-9]+_[a-zA-Z0-9]+$ ]]; then
  echo "ERROR: Invalid User Pool ID format: $USER_POOL_ID" >&2
  exit 1
fi

echo "User Pool created with ID: $USER_POOL_ID"

USER_POOL_ARN=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Arn // empty')
aws cognito-idp tag-resource \
  --resource-arn "$USER_POOL_ARN" \
  --tags project=doc-smith,tutorial=amazon-cognito-gs

# Wait for user pool to be ready
echo "Waiting for user pool to be ready..."
sleep 5

# Step 2: Create an App Client with enhanced security
echo "Creating app client..."
APP_CLIENT_OUTPUT=$(aws cognito-idp create-user-pool-client \
```

```
  --user-pool-id "$USER_POOL_ID" \
  --client-name "$APP_CLIENT_NAME" \
  --no-generate-secret \
  --explicit-auth-flows ALLOW_REFRESH_TOKEN_AUTH ALLOW_USER_PASSWORD_AUTH \
  --callback-urls '["https://localhost:3000/callback"]' \
  --allowed-o-auth-flows 'code' \
  --allowed-o-auth-scopes 'openid' 'email' 'profile' \
  --allowed-o-auth-flows-user-pool-client \
  --prevent-user-existence-errors ENABLED \
  --enable-token-revocation \
  --access-token-validity 1 \
  --id-token-validity 1 \
  --refresh-token-validity 30 \
  --token-validity-units 'AccessToken=hours,IdToken=hours,RefreshToken=days' \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-client"

# Extract the Client ID using jq for safety
CLIENT_ID=$(echo "$APP_CLIENT_OUTPUT" | jq -r '.UserPoolClient.ClientId // empty')
if [ -z "$CLIENT_ID" ]; then
  echo "ERROR: Failed to extract Client ID" >&2
  cleanup_on_error
  exit 1
fi

# Validate Client ID format
if ! [[ "$CLIENT_ID" =~ ^[a-z0-9]{26}$ ]]; then
  echo "ERROR: Invalid Client ID format: $CLIENT_ID" >&2
  cleanup_on_error
  exit 1
fi

echo "App Client created with ID: $CLIENT_ID"

# Step 3: Set Up a Domain for Your User Pool
echo "Setting up user pool domain..."
DOMAIN_OUTPUT=$(aws cognito-idp create-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-domain"
echo "Domain created: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"
```

```
# Step 4: View User Pool Details
echo "Retrieving user pool details..."
USER_POOL_DETAILS=$(aws cognito-idp describe-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool"
echo "User Pool details retrieved successfully"

# Step 5: View App Client Details
echo "Retrieving app client details..."
APP_CLIENT_DETAILS=$(aws cognito-idp describe-user-pool-client \
  --user-pool-id "$USER_POOL_ID" \
  --client-id "$CLIENT_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool-client"
echo "App Client details retrieved successfully"

# Step 6: Create a User (Admin)
echo "Creating admin user..."
ADMIN_USER_EMAIL="admin@example.com"
TEMP_PASSWORD="$(openssl rand -base64 12 | tr -d '\n')!@#"
if [ -z "$TEMP_PASSWORD" ]; then
  echo "ERROR: Failed to generate temporary password" >&2
  cleanup_on_error
  exit 1
fi

ADMIN_USER_OUTPUT=$(aws cognito-idp admin-create-user \
  --user-pool-id "$USER_POOL_ID" \
  --username "$ADMIN_USER_EMAIL" \
  --user-attributes Name=email,Value="$ADMIN_USER_EMAIL"
 Name=email_verified,Value=true \
  --temporary-password "$TEMP_PASSWORD" \
  --message-action SUPPRESS \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-create-user"
echo "Admin user created: $ADMIN_USER_EMAIL"

# Securely clear temporary password from memory
unset TEMP_PASSWORD
```

```
# Step 7: Self-Registration
echo "Demonstrating self-registration..."
USER_EMAIL="user@example.com"
USER_PASSWORD="SecurePassword123!"
SIGNUP_OUTPUT=$(aws cognito-idp sign-up \
  --client-id "$CLIENT_ID" \
  --username "$USER_EMAIL" \
  --password "$USER_PASSWORD" \
  --user-attributes Name=email,Value="$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "sign-up"
echo "User signed up: $USER_EMAIL"
echo "A confirmation code would be sent to the user's email in a real scenario"

# Securely clear password from memory
unset USER_PASSWORD

echo ""
echo "====================================================="
echo "IMPORTANT: In a real scenario, the user would receive"
echo "a confirmation code via email. For this demo, we'll"
echo "use admin-confirm-sign-up instead."
echo "====================================================="
echo ""

# Step 8: Confirm User Registration (using admin privileges for demo)
echo "Confirming user registration (admin method)..."
CONFIRM_OUTPUT=$(aws cognito-idp admin-confirm-sign-up \
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-confirm-sign-up"
echo "User confirmed: $USER_EMAIL"

# Step 9: Set permanent password for user
echo "Setting permanent password for user..."
SET_PASSWORD="SecureUserPassword123!"
SET_PASS_OUTPUT=$(aws cognito-idp admin-set-user-password \
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --password "$SET_PASSWORD" \
```

```
  --permanent \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-set-user-password"
echo "Permanent password set for user"

unset SET_PASSWORD

# Step 10: Authenticate a User
echo "Authenticating user..."
AUTH_PASSWORD="SecureUserPassword123!"
AUTH_OUTPUT=$(aws cognito-idp initiate-auth \
  --client-id "$CLIENT_ID" \
  --auth-flow USER_PASSWORD_AUTH \
  --auth-parameters "USERNAME=$USER_EMAIL,PASSWORD=$AUTH_PASSWORD" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "initiate-auth"
echo "User authenticated successfully"

unset AUTH_PASSWORD

# Extract auth tokens securely
ID_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.IdToken // empty')
ACCESS_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.AccessToken //
 empty')

# Validate tokens exist
if [ -z "$ID_TOKEN" ] || [ -z "$ACCESS_TOKEN" ]; then
  echo "WARNING: Failed to extract authentication tokens" >&2
else
  echo "Authentication tokens obtained successfully"
fi

# Securely clear tokens from memory
unset ID_TOKEN
unset ACCESS_TOKEN

# Step 11: List Users in the User Pool
echo "Listing users in the user pool..."
USERS_OUTPUT=$(aws cognito-idp list-users \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
```

```
check_aws_error "list-users"
echo "Users listed successfully"

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "User Pool ID: $USER_POOL_ID"
echo "User Pool Name: $USER_POOL_NAME"
echo "App Client ID: $CLIENT_ID"
echo "App Client Name: $APP_CLIENT_NAME"
echo "Domain: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"
echo "Admin User: $ADMIN_USER_EMAIL"
echo "Regular User: $USER_EMAIL"
echo "==================================================="
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting cleanup process..."

# Step 12: Clean Up Resources
echo "Deleting user pool domain..."
DELETE_DOMAIN_OUTPUT=$(aws cognito-idp delete-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "delete-user-pool-domain"
echo "Domain deleted successfully"

# Wait for domain deletion to complete
echo "Waiting for domain deletion to complete..."
sleep 5

echo "Deleting user pool (this will also delete the app client)..."
DELETE_POOL_OUTPUT=$(aws cognito-idp delete-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
```

```
check_aws_error "delete-user-pool"
echo "User pool deleted successfully"

echo "All resources have been cleaned up"

echo "Script completed at $(date)"

# Remove trap to prevent cleanup on successful exit
trap - EXIT ERR
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AdminConfirmSignUp
  - AdminCreateUser
  - CreateUserPool
  - CreateUserPoolClient
  - CreateUserPoolDomain
  - DeleteUserPool
  - DeleteUserPoolDomain
  - DescribeUserPool
  - DescribeUserPoolClient
  - InitiateAuth
  - ListUsers
  - SignUp

AWS Config exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS Config.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
```

```
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
```

```
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=========================================="
    echo "CREATED RESOURCES"
    echo "=========================================="
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
```

```
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
```

```
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
```

```
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
```

```
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
```

```
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
```

```
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources
```

```
# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel

- PutPublicAccessBlock
- PutRolePolicy
- Rm
- StartConfigurationRecorder
- StopConfigurationRecorder

Exemplos de clientes do Connect usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Connect Customer.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criação de uma instância de central de atendimento

O exemplo de código a seguir mostra como:

- Criar uma instância do Amazon Connect.
- Configurar um usuário administrador.
- Configurar opções de telefonia.
- Configurar um número de telefone.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create an Amazon Connect instance using AWS CLI
# This script follows the steps in the Amazon Connect instance creation tutorial

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="connect-instance-creation.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting Amazon Connect instance creation at $(date)" > "$LOG_FILE"

# Set default region
AWS_REGION="${AWS_REGION:-us-west-2}"
echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"

# Validate AWS CLI is installed and credentials are available
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid" | tee -a "$LOG_FILE"
    exit 1
fi

# Function to log commands and their output
log_cmd() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
```

```
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        return 1
    fi
    return 0
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 "$LOG_FILE"

    if [[ -n "${INSTANCE_ID:-}" ]]; then
        echo "Deleting Amazon Connect instance: $INSTANCE_ID" | tee -a "$LOG_FILE"
        log_cmd "aws connect delete-instance --instance-id '$INSTANCE_ID' --region
 '$AWS_REGION'" || true
    fi

    echo "Cleanup completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
}

# Set trap to clean up on error
trap cleanup_on_error ERR EXIT

# Function to wait for instance to be fully active
wait_for_instance() {
    local instance_id="$1"
    local max_attempts=30
    local attempt=1

    echo "Waiting for instance $instance_id to become fully active..." | tee -a
 "$LOG_FILE"

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt of $max_attempts: Checking instance status..." | tee
 -a "$LOG_FILE"
```

```
        # Try to describe the instance
        local result
        result=$(log_cmd "aws connect describe-instance --instance-id '$instance_id'
 --region '$AWS_REGION' --output json" 2>&1) || true

        # Check if the command was successful and instance status is ACTIVE
        if [[ $? -eq 0 && "$result" =~ "ACTIVE" ]]; then
            echo "Instance is now fully active and ready to use." | tee -a
 "$LOG_FILE"
            return 0
        fi

        echo "Instance not fully active yet. Waiting 30 seconds before next
 check..." | tee -a "$LOG_FILE"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for instance to become fully active." | tee -a
 "$LOG_FILE"
    return 1
}

# Function to check and handle existing instances
check_existing_instances() {
    echo "Checking for existing Amazon Connect instances..." | tee -a "$LOG_FILE"

    local instances
    instances=$(log_cmd "aws connect list-instances --region '$AWS_REGION' --output
 json" 2>&1) || true
    if [[ $? -ne 0 ]]; then
        echo "ERROR: Failed to list existing instances" | tee -a "$LOG_FILE"
        return 1
    fi

    # Check if there are any instances
    local instance_count
    instance_count=$(echo "$instances" | jq '.InstanceSummaryList | length' 2>/dev/
null || echo 0)

    if [[ $instance_count -gt 0 ]]; then
        echo "Found $instance_count existing Amazon Connect instance(s)" | tee -a
 "$LOG_FILE"
```

```
        echo "$instances" | jq '.InstanceSummaryList[] | {Id, Alias}' 2>/dev/null |
 tee -a "$LOG_FILE" || true

        echo ""
        echo "============================================"
        echo "EXISTING INSTANCES FOUND"
        echo "============================================"
        echo "Found $instance_count existing Amazon Connect instance(s)."
        echo "Auto-deleting existing instances to free up quota..." | tee -a
 "$LOG_FILE"

        echo "Deleting existing instances..." | tee -a "$LOG_FILE"

        # Extract instance IDs and delete each one
        local instance_ids
        instance_ids=$(echo "$instances" | jq -r '.InstanceSummaryList[].Id' 2>/dev/
null || echo "")

        while IFS= read -r id; do
            if [[ -n "$id" ]]; then
                echo "Deleting instance: $id" | tee -a "$LOG_FILE"
                log_cmd "aws connect delete-instance --instance-id '$id' --region
 '$AWS_REGION'" || {
                    echo "WARNING: Failed to delete instance $id" | tee -a
 "$LOG_FILE"
                }

                # Wait a bit between deletions
                sleep 5
            fi
        done <<< "$instance_ids"

        echo "Waiting for deletions to complete..." | tee -a "$LOG_FILE"
        sleep 30
    else
        echo "No existing Amazon Connect instances found" | tee -a "$LOG_FILE"
    fi

    return 0
}

# Check for existing instances before proceeding
check_existing_instances
```

```
# Generate a random instance alias to avoid naming conflicts
INSTANCE_ALIAS="connect-instance-$(openssl rand -hex 6 2>/dev/null || date +%s%N)"
echo "Using instance alias: $INSTANCE_ALIAS" | tee -a "$LOG_FILE"

# Step 1: Create Amazon Connect instance
echo "Step 1: Creating Amazon Connect instance..." | tee -a "$LOG_FILE"
INSTANCE_RESULT=$(log_cmd "aws connect create-instance --identity-management-type
 CONNECT_MANAGED --instance-alias '$INSTANCE_ALIAS' --inbound-calls-enabled --
outbound-calls-enabled --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$INSTANCE_RESULT" $? "Failed to create Amazon Connect instance";
 then
    # Check if the error is due to quota limit
    if [[ "$INSTANCE_RESULT" =~ "ServiceQuotaExceededException" ||
 "$INSTANCE_RESULT" =~ "Quota limit reached" ]]; then
        echo "Quota limit reached for Amazon Connect instances. Please delete
 existing instances or request a quota increase." | tee -a "$LOG_FILE"
    fi
    exit 1
fi

# Extract instance ID from the result using jq
INSTANCE_ID=$(echo "$INSTANCE_RESULT" | jq -r '.InstanceId' 2>/dev/null || echo "")
INSTANCE_ARN=$(echo "$INSTANCE_RESULT" | jq -r '.InstanceArn' 2>/dev/null || echo
 "")

if [[ -z "$INSTANCE_ID" ]]; then
    echo "ERROR: Failed to extract instance ID from the result" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Successfully created Amazon Connect instance with ID: $INSTANCE_ID" | tee -a
 "$LOG_FILE"
echo "Instance ARN: $INSTANCE_ARN" | tee -a "$LOG_FILE"

# Tag the instance
aws connect tag-resource --resource-arn "$INSTANCE_ARN" \
    --tags project=doc-smith,tutorial=connect-gs \
    --region "$AWS_REGION"

# Wait for the instance to be fully created and active
if ! wait_for_instance "$INSTANCE_ID"; then
    echo "ERROR: Instance did not become fully active within the timeout period" |
 tee -a "$LOG_FILE"
```

```
    exit 1
fi

# Step 2: Get security profiles to find the Admin profile ID
echo "Step 2: Getting security profiles..." | tee -a "$LOG_FILE"
SECURITY_PROFILES=$(log_cmd "aws connect list-security-profiles --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$SECURITY_PROFILES" $? "Failed to list security profiles"; then
    exit 1
fi

# Extract Admin security profile ID using jq
ADMIN_PROFILE_ID=$(echo "$SECURITY_PROFILES" | jq -r '.SecurityProfileSummaryList[]
 | select(.Name=="Admin") | .Id' 2>/dev/null | head -1 || echo "")

if [[ -z "$ADMIN_PROFILE_ID" ]]; then
    echo "ERROR: Failed to find Admin security profile ID" | tee -a "$LOG_FILE"
    echo "Available security profiles:" | tee -a "$LOG_FILE"
    echo "$SECURITY_PROFILES" | jq '.SecurityProfileSummaryList[] | {Id, Name}' 2>/
dev/null | tee -a "$LOG_FILE" || echo "$SECURITY_PROFILES" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Found Admin security profile ID: $ADMIN_PROFILE_ID" | tee -a "$LOG_FILE"

# Step 3: Get routing profiles to find a default routing profile ID
echo "Step 3: Getting routing profiles..." | tee -a "$LOG_FILE"
ROUTING_PROFILES=$(log_cmd "aws connect list-routing-profiles --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$ROUTING_PROFILES" $? "Failed to list routing profiles"; then
    exit 1
fi

# Extract the first routing profile ID using jq
ROUTING_PROFILE_ID=$(echo "$ROUTING_PROFILES" | jq -r
 '.RoutingProfileSummaryList[0].Id' 2>/dev/null || echo "")

if [[ -z "$ROUTING_PROFILE_ID" ]]; then
    echo "ERROR: Failed to find a routing profile ID" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
echo "Found routing profile ID: $ROUTING_PROFILE_ID" | tee -a "$LOG_FILE"

# Step 4: Create an admin user
echo "Step 4: Creating admin user..." | tee -a "$LOG_FILE"

# Generate a secure password
ADMIN_PASSWORD="Connect$(openssl rand -base64 12 2>/dev/null || head -c 12 /dev/
urandom | base64)"

USER_RESULT=$(log_cmd "aws connect create-user --instance-id '$INSTANCE_ID'
 --username admin --password '$ADMIN_PASSWORD' --identity-info
 FirstName=Admin,LastName=User,Email=admin@example.com --phone-config
 PhoneType=DESK_PHONE,AutoAccept=true,AfterContactWorkTimeLimit=30,DeskPhoneNumber=
+12065550100 --security-profile-ids '$ADMIN_PROFILE_ID' --routing-profile-id
 '$ROUTING_PROFILE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$USER_RESULT" $? "Failed to create admin user"; then
    exit 1
fi

# Extract user ID using jq
USER_ID=$(echo "$USER_RESULT" | jq -r '.UserId' 2>/dev/null || echo "")

if [[ -z "$USER_ID" ]]; then
    echo "ERROR: Failed to extract user ID from the result" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Successfully created admin user with ID: $USER_ID" | tee -a "$LOG_FILE"
echo "Admin password: $ADMIN_PASSWORD" | tee -a "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Step 5: Configure telephony options
echo "Step 5: Configuring telephony options..." | tee -a "$LOG_FILE"

# Enable early media
EARLY_MEDIA_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-id
 '$INSTANCE_ID' --attribute-type EARLY_MEDIA --value true --region '$AWS_REGION'"
 2>&1) || true

if ! check_error "$EARLY_MEDIA_RESULT" $? "Failed to enable early media"; then
    exit 1
fi
```

```
# Enable multi-party calls and enhanced monitoring for voice
MULTI_PARTY_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-
id '$INSTANCE_ID' --attribute-type MULTI_PARTY_CONFERENCE --value true --region
 '$AWS_REGION'" 2>&1) || true

if ! check_error "$MULTI_PARTY_RESULT" $? "Failed to enable multi-party calls"; then
    exit 1
fi

# Enable multi-party chats and enhanced monitoring for chat
MULTI_PARTY_CHAT_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-
id '$INSTANCE_ID' --attribute-type MULTI_PARTY_CHAT_CONFERENCE --value true --region
 '$AWS_REGION'" 2>&1) || true

if ! check_error "$MULTI_PARTY_CHAT_RESULT" $? "Failed to enable multi-party chats";
 then
    exit 1
fi

echo "Successfully configured telephony options" | tee -a "$LOG_FILE"

# Step 6: View storage configurations
echo "Step 6: Viewing storage configurations..." | tee -a "$LOG_FILE"

# List storage configurations for chat transcripts
STORAGE_CONFIGS=$(log_cmd "aws connect list-instance-storage-configs --instance-id
 '$INSTANCE_ID' --resource-type CHAT_TRANSCRIPTS --region '$AWS_REGION' --output
 json" 2>&1) || true

if ! check_error "$STORAGE_CONFIGS" $? "Failed to list storage configurations"; then
    exit 1
fi

echo "Successfully retrieved storage configurations" | tee -a "$LOG_FILE"

# Step 7: Verify instance details
echo "Step 7: Verifying instance details..." | tee -a "$LOG_FILE"
INSTANCE_DETAILS=$(log_cmd "aws connect describe-instance --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$INSTANCE_DETAILS" $? "Failed to describe instance"; then
    exit 1
fi
```

```
echo "Successfully verified instance details" | tee -a "$LOG_FILE"

# Step 8: Search for available phone numbers (optional)
echo "Step 8: Searching for available phone numbers..." | tee -a "$LOG_FILE"
PHONE_NUMBERS=$(log_cmd "aws connect search-available-phone-numbers --target-arn
 '$INSTANCE_ARN' --phone-number-type TOLL_FREE --phone-number-country-code US --max-
results 5 --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$PHONE_NUMBERS" $? "Failed to search for available phone numbers";
 then
    exit 1
fi

# Extract the first phone number if available using jq
PHONE_NUMBER=$(echo "$PHONE_NUMBERS" | jq -r '.AvailableNumbersList[0].PhoneNumber'
 2>/dev/null || echo "")

if [[ -n "$PHONE_NUMBER" ]]; then
    echo "Found available phone number: $PHONE_NUMBER" | tee -a "$LOG_FILE"

    echo ""
    echo "==========================================="
    echo "CLAIM PHONE NUMBER"
    echo "==========================================="
    echo "Auto-claiming available phone number $PHONE_NUMBER..." | tee -a
 "$LOG_FILE"

    CLAIM_RESULT=$(log_cmd "aws connect claim-phone-number --target-arn
 '$INSTANCE_ARN' --phone-number '$PHONE_NUMBER' --region '$AWS_REGION' --output
 json" 2>&1) || true

    if ! check_error "$CLAIM_RESULT" $? "Failed to claim phone number"; then
        echo "WARNING: Failed to claim phone number, but continuing with script" |
 tee -a "$LOG_FILE"
    else
        echo "Successfully claimed phone number" | tee -a "$LOG_FILE"
        # Extract the phone number ID from the claim result using jq
        PHONE_NUMBER_ID=$(echo "$CLAIM_RESULT" | jq -r '.PhoneNumberId' 2>/dev/null
 || echo "")
    fi
else
    echo "No available phone numbers found" | tee -a "$LOG_FILE"
fi
```

```
# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "Amazon Connect Instance ID: $INSTANCE_ID"
echo "Amazon Connect Instance ARN: $INSTANCE_ARN"
echo "Admin User ID: $USER_ID"
echo "Admin Username: admin"
echo "Admin Password: $ADMIN_PASSWORD"
if [[ -n "${PHONE_NUMBER:-}" ]]; then
    echo "Claimed Phone Number: $PHONE_NUMBER"
    if [[ -n "${PHONE_NUMBER_ID:-}" ]]; then
        echo "Claimed Phone Number ID: $PHONE_NUMBER_ID"
    fi
fi
echo "============================================"
echo ""

# Auto-confirm cleanup and clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Auto-starting cleanup..." | tee -a "$LOG_FILE"

# Release claimed phone number if applicable
if [[ -n "${PHONE_NUMBER_ID:-}" ]]; then
    echo "Releasing phone number: $PHONE_NUMBER_ID" | tee -a "$LOG_FILE"
    RELEASE_RESULT=$(log_cmd "aws connect release-phone-number --phone-number-id
 '$PHONE_NUMBER_ID' --region '$AWS_REGION'" 2>&1) || true

    if ! check_error "$RELEASE_RESULT" $? "Failed to release phone number"; then
        echo "WARNING: Failed to release phone number" | tee -a "$LOG_FILE"
    else
        echo "Successfully released phone number" | tee -a "$LOG_FILE"
    fi

    echo "Waiting for phone number release to complete..." | tee -a "$LOG_FILE"
    sleep 10
fi

# Delete the Amazon Connect instance (this will also delete all associated
 resources)
```

```
echo "Deleting Amazon Connect instance: $INSTANCE_ID" | tee -a "$LOG_FILE"
DELETE_RESULT=$(log_cmd "aws connect delete-instance --instance-id '$INSTANCE_ID' --
region '$AWS_REGION'" 2>&1) || true

if ! check_error "$DELETE_RESULT" $? "Failed to delete instance"; then
    echo "WARNING: Failed to delete instance" | tee -a "$LOG_FILE"
else
    echo "Successfully deleted instance" | tee -a "$LOG_FILE"
fi

echo "Cleanup completed. All resources have been deleted." | tee -a "$LOG_FILE"

echo "Script completed successfully. See $LOG_FILE for details." | tee -a
 "$LOG_FILE"

trap - ERR EXIT
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - ClaimPhoneNumber
  - CreateInstance
  - CreateUser
  - DeleteInstance
  - DescribeInstance
  - ListInstanceStorageConfigs
  - ListInstances
  - ListRoutingProfiles
  - ListSecurityProfiles
  - ReleasePhoneNumber
  - SearchAvailablePhoneNumbers
  - UpdateInstanceAttribute

Direct Connect exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com Direct Connect.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos de conexões de rede dedicadas

O exemplo de código a seguir mostra como:

- Use ec2 CreateVpnGateway
- Use ec2 DeleteVpnGateway
- Use ec2 DescribeVpnGateways
- Use conexão direta CreateConnection
- Use conexão direta CreatePrivateVirtualInterface
- Use conexão direta DeleteConnection

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Direct Connect Connection Management Script - Version 6
# This script demonstrates how to create and manage AWS Direct Connect connections
 using the AWS CLI
# This version includes fixes for user input handling and better error reporting

# Set up logging
```

```
LOG_FILE="directconnect-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting AWS Direct Connect script v6"

# Function to check for errors in command output
check_error() {
    local output=$1
    local command=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $command"
        echo "Output: $output"
        cleanup_resources
        exit 1
    fi
}

# Function to wait for VGW to be available
wait_for_vgw() {
    local vgw_id=$1
    local max_attempts=30
    local attempt=1

    echo "Waiting for virtual private gateway $vgw_id to become available..."

    while [ $attempt -le $max_attempts ]; do
        VGW_STATE=$(aws ec2 describe-vpn-gateways --vpn-gateway-ids "$vgw_id" --
query 'VpnGateways[0].State' --output text)

        if [ "$VGW_STATE" == "available" ]; then
            echo "Virtual private gateway is now available"
            return 0
        elif [ "$VGW_STATE" == "failed" ]; then
            echo "Virtual private gateway failed to become available"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: VGW state is $VGW_STATE, waiting 10
 seconds..."
        sleep 10
        attempt=$((attempt + 1))
    done
```

```
    echo "Timeout waiting for VGW to become available"
    return 1
}

# Function to wait for connection to be available
wait_for_connection() {
    local connection_id=$1
    local max_attempts=60
    local attempt=1

    echo "Waiting for connection $connection_id to become available..."
    echo "Note: This can take 30+ minutes in production as AWS provisions the
 physical connection"

    while [ $attempt -le $max_attempts ]; do
        CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$connection_id" --query 'connections[0].connectionState' --output text)

        if [ "$CONNECTION_STATE" == "available" ]; then
            echo "Connection is now available"
            return 0
        elif [ "$CONNECTION_STATE" == "rejected" ] || [ "$CONNECTION_STATE" ==
 "deleted" ]; then
            echo "Connection failed with state: $CONNECTION_STATE"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: Connection state is $CONNECTION_STATE,
 waiting 30 seconds..."
        sleep 30
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for connection to become available"
    return 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete virtual interfaces if they exist
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "Deleting private virtual interface: $PRIVATE_VIF_ID"
```

```
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PRIVATE_VIF_ID"
    fi

    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "Deleting public virtual interface: $PUBLIC_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PUBLIC_VIF_ID"
    fi

    # Delete connection if it exists
    if [ -n "$CONNECTION_ID" ]; then
        echo "Deleting connection: $CONNECTION_ID"
        aws directconnect delete-connection --connection-id "$CONNECTION_ID"
    fi

    # Delete VGW if it exists
    if [ -n "$VGW_ID" ]; then
        echo "Deleting virtual private gateway: $VGW_ID"
        aws ec2 delete-vpn-gateway --vpn-gateway-id "$VGW_ID"
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 6)
CONNECTION_NAME="DxConn-${RANDOM_ID}"

# Step 1: List available Direct Connect locations
echo "Listing available Direct Connect locations..."
LOCATIONS_OUTPUT=$(aws directconnect describe-locations)
check_error "$LOCATIONS_OUTPUT" "describe-locations"
echo "$LOCATIONS_OUTPUT"

# Extract the first location code for demonstration purposes
LOCATION_CODE=$(aws directconnect describe-locations --query
 'locations[0].locationCode' --output text)
if [ -z "$LOCATION_CODE" ] || [ "$LOCATION_CODE" == "None" ]; then
    echo "Error: Could not extract location code from the output."
    exit 1
fi
echo "Using location: $LOCATION_CODE"

# Step 2: Create a dedicated connection
```

```
echo "Creating a dedicated connection at location $LOCATION_CODE with bandwidth
 1Gbps..."
CONNECTION_OUTPUT=$(aws directconnect create-connection \
    --location "$LOCATION_CODE" \
    --bandwidth "1Gbps" \
    --connection-name "$CONNECTION_NAME" \
    --tags key=project,value=doc-smith key=tutorial,value=aws-direct-connect-gs)
check_error "$CONNECTION_OUTPUT" "create-connection"
echo "$CONNECTION_OUTPUT"

# Extract connection ID directly from the output
CONNECTION_ID=$(echo "$CONNECTION_OUTPUT" | grep -o '"connectionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$CONNECTION_ID" ]; then
    echo "Error: Could not extract connection ID from the output."
    exit 1
fi
echo "Connection created with ID: $CONNECTION_ID"

# Step 3: Describe the connection
echo "Retrieving connection details..."
DESCRIBE_OUTPUT=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID")
check_error "$DESCRIBE_OUTPUT" "describe-connections"
echo "$DESCRIBE_OUTPUT"

# Step 4: Update the connection name
NEW_CONNECTION_NAME="${CONNECTION_NAME}-updated"
echo "Updating connection name to $NEW_CONNECTION_NAME..."
UPDATE_OUTPUT=$(aws directconnect update-connection \
    --connection-id "$CONNECTION_ID" \
    --connection-name "$NEW_CONNECTION_NAME")
check_error "$UPDATE_OUTPUT" "update-connection"
echo "$UPDATE_OUTPUT"

# Step 5: Check if we can download the LOA-CFA
# Note: In a real scenario, the LOA-CFA might not be immediately available
echo "Attempting to download the LOA-CFA (this may not be available yet)..."
LOA_OUTPUT=$(aws directconnect describe-loa --connection-id "$CONNECTION_ID" 2>&1)
if echo "$LOA_OUTPUT" | grep -i "error" > /dev/null; then
    echo "LOA-CFA not available yet. This is expected for newly created
 connections."
    echo "The LOA-CFA will be available once AWS begins provisioning your
 connection."
```

```
else
    LOA_CONTENT=$(echo "$LOA_OUTPUT" | grep -o '"loaContent": "[^"]*' | cut -d'"' -
f4)
    echo "$LOA_CONTENT" | base64 --decode > "loa-cfa-${CONNECTION_ID}.pdf"
    echo "LOA-CFA downloaded to loa-cfa-${CONNECTION_ID}.pdf"
fi

# Step 6: Create a virtual private gateway (required for private virtual interface)
echo "Creating a virtual private gateway..."
VGW_OUTPUT=$(aws ec2 create-vpn-gateway --type ipsec.1 \
    --tag-specifications 'ResourceType=vpn-gateway,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=aws-direct-connect-gs}]')
check_error "$VGW_OUTPUT" "create-vpn-gateway"
echo "$VGW_OUTPUT"

# Extract VGW ID directly from the output
VGW_ID=$(echo "$VGW_OUTPUT" | grep -o '"VpnGatewayId": "[^"]*' | cut -d'"' -f4)
if [ -z "$VGW_ID" ]; then
    echo "Error: Could not extract VPN gateway ID from the output."
    exit 1
fi
echo "Virtual private gateway created with ID: $VGW_ID"

# Wait for VGW to become available
if ! wait_for_vgw "$VGW_ID"; then
    echo "Failed to wait for VGW to become available. Skipping virtual interface
 creation."
    VIF_CREATION_SKIPPED=true
else
    VIF_CREATION_SKIPPED=false
fi

# Step 7: Create a private virtual interface (only if VGW is available)
if [ "$VIF_CREATION_SKIPPED" = false ]; then
    echo "Creating a private virtual interface..."
    PRIVATE_VIF_OUTPUT=$(aws directconnect create-private-virtual-interface \
        --connection-id "$CONNECTION_ID" \
        --new-private-virtual-interface '{
            "virtualInterfaceName": "PrivateVIF-'"$RANDOM_ID"'",
            "vlan": 100,
            "asn": 65000,
            "authKey": "'"$RANDOM_ID"'key",
            "amazonAddress": "192.168.1.1/30",
            "customerAddress": "192.168.1.2/30",
```

```
            "addressFamily": "ipv4",
            "virtualGatewayId": "'"$VGW_ID"'"
        }' 2>&1)

    if echo "$PRIVATE_VIF_OUTPUT" | grep -i "error" > /dev/null; then
        echo "Could not create private virtual interface. This is expected if the
 connection is not yet available."
        echo "Error: $PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=""
    else
        echo "$PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=$(echo "$PRIVATE_VIF_OUTPUT" | grep -o '"virtualInterfaceId":
 "[^"]*' | cut -d'"' -f4)
        echo "Private virtual interface created with ID: $PRIVATE_VIF_ID"
    fi
else
    echo "Skipping private virtual interface creation due to VGW not being
 available"
    PRIVATE_VIF_ID=""
fi

# Step 8: Check connection state and provide guidance for public virtual interface
CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID" --query 'connections[0].connectionState' --output text)
echo "Current connection state: $CONNECTION_STATE"

if [ "$CONNECTION_STATE" != "available" ]; then
    echo ""
    echo "============================================"
    echo "CONNECTION NOT YET AVAILABLE"
    echo "============================================"
    echo "The connection is in '$CONNECTION_STATE' state."
    echo "In production, you would:"
    echo "1. Wait for AWS to provision the connection (can take 30+ minutes)"
    echo "2. Download the LOA-CFA when available"
    echo "3. Provide the LOA-CFA to your network provider for cross-connect"
    echo "4. Create virtual interfaces once connection is 'available'"
    echo ""

    # Ask if user wants to wait for connection to become available
    echo ""
    echo "============================================"
    echo "CONNECTION WAIT CONFIRMATION"
    echo "============================================"
```

```
    echo -n "Do you want to wait for the connection to become available? (y/n): "
    read -r WAIT_CHOICE

    if [[ "$WAIT_CHOICE" =~ ^[Yy]$ ]]; then
        if wait_for_connection "$CONNECTION_ID"; then
            echo "Connection is now available! You could now create virtual
 interfaces."
        else
            echo "Connection did not become available within the timeout period."
        fi
    else
        echo "Skipping wait for connection availability."
    fi
else
    echo "Connection is available! Virtual interfaces can be created."
fi

# Step 9: List all virtual interfaces
echo "Listing all virtual interfaces..."
VIF_LIST_OUTPUT=$(aws directconnect describe-virtual-interfaces)
check_error "$VIF_LIST_OUTPUT" "describe-virtual-interfaces"
echo "$VIF_LIST_OUTPUT"

# Step 10: Display important information about production usage
echo ""
echo "============================================"
echo "IMPORTANT PRODUCTION NOTES"
echo "============================================"
echo "1. Direct Connect connections take time to be provisioned by AWS"
echo "2. You cannot create virtual interfaces until the connection is 'available'"
echo "3. For public virtual interfaces, you must own the public IP addresses"
echo "4. LOA-CFA (Letter of Authorization) is needed for cross-connect at the
 facility"
echo "5. This demo creates resources that incur costs (~\$300/month for 1Gbps)"
echo "6. Always test connectivity before putting into production"
echo ""

# Step 11: Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo -n "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually delete them later."
    echo ""
    echo "Created resources:"
    echo "- Connection ID: $CONNECTION_ID"
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "- Private Virtual Interface ID: $PRIVATE_VIF_ID"
    fi
    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "- Public Virtual Interface ID: $PUBLIC_VIF_ID"
    fi
    echo "- Virtual Private Gateway ID: $VGW_ID"
    echo ""
    echo "Manual cleanup commands:"
    echo "aws directconnect delete-connection --connection-id $CONNECTION_ID"
    echo "aws ec2 delete-vpn-gateway --vpn-gateway-id $VGW_ID"
    echo ""
    echo "Remember: Direct Connect resources incur ongoing costs!"
fi

echo "$(date): Script completed"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateConnection
  - CreatePrivateVirtualInterface
  - CreateVpnGateway
  - DeleteConnection
  - DeleteVirtualInterface
  - DeleteVpnGateway
  - DescribeConnections
  - DescribeLoa
  - DescribeLocations
  - DescribeVirtualInterfaces
  - DescribeVpnGateways

- UpdateConnection

Exemplos do Amazon DocumentDB usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon DocumentDB.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução aos bancos de dados de documentos

O exemplo de código a seguir mostra como:

- Criar um grupo de sub-redes de banco de dados
- Criar um cluster do DocumentDB.
- Criar uma instância do DocumentDB.
- Configurar a segurança e a conectividade.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

###############################################################################
# Configuration
###############################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

###############################################################################
# Logging
###############################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""

###############################################################################
# Region pre-check
###############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
```

```
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

################################################################################
# Error handler
################################################################################
handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

################################################################################
# Wait function
################################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
```

```
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
```

```
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound\|
not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

################################################################################
# Cleanup
################################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
```

```
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG ingress
 rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed to
 delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}
```

```
################################################################################
# Step 1: Generate password and store in Secrets Manager
################################################################################
echo "============================================"
echo "Step 1: Create master password in Secrets Manager"
echo "============================================"
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

################################################################################
# Step 2: Find default VPC and subnets
################################################################################
echo "============================================"
```

```
echo "Step 2: Find default VPC and subnets"
echo "============================================"
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi
```

```
# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

##############################################################################
# Step 3: Create subnet group
##############################################################################
echo "==========================================="
echo "Step 3: Create DocumentDB subnet group"
echo "==========================================="
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

##############################################################################
# Step 4: Create DocumentDB cluster
##############################################################################
echo "==========================================="
echo "Step 4: Create DocumentDB cluster"
echo "==========================================="
```

```
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "============================================"
echo "Step 5: Create DocumentDB instance"
echo "============================================"
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
```

```
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

##############################################################################
# Step 6: Get cluster endpoint and security group
##############################################################################
echo "==========================================="
echo "Step 6: Get cluster endpoint and security group"
echo "==========================================="
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
```

```
# Step 7: Add security group ingress for port 27017 from user's IP
##############################################################################
echo "============================================="
echo "Step 7: Add security group ingress rule"
echo "============================================="
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null ||
 true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
```

```
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

##############################################################################
# Step 8: Download CA certificate
##############################################################################
echo "==========================================="
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "==========================================="
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

##############################################################################
# Step 9: Display connection information
##############################################################################
```

```
echo "==========================================="
echo "CONNECTION INFORMATION"
echo "==========================================="
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH} \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString --
output text)"
echo ""

###############################################################################
# Step 10: Cleanup
###############################################################################
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - RevokeSecurityGroupIngress
  - Wait

Exemplos do DynamoDB usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O cenário de conceitos básicos do DynamoDB.

```
###############################################################################
# function dynamodb_getting_started_movies
#
# Scenario to create an Amazon DynamoDB table and perform a series of operations on
 the table.
#
```

```
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function dynamodb_getting_started_movies() {

  source ./dynamodb_operations.sh

  key_schema_json_file="dynamodb_key_schema.json"
  attribute_definitions_json_file="dynamodb_attr_def.json"
  item_json_file="movie_item.json"
  key_json_file="movie_key.json"
  batch_json_file="batch.json"
  attribute_names_json_file="attribute_names.json"
  attributes_values_json_file="attribute_values.json"

  echo_repeat "*" 88
  echo
  echo "Welcome to the Amazon DynamoDB getting started demo."
  echo
  echo_repeat "*" 88
  echo

  local table_name
  echo -n "Enter a name for a new DynamoDB table: "
  get_input
  table_name=$get_input_result

  echo '[
  {"AttributeName": "year", "KeyType": "HASH"},
   {"AttributeName": "title", "KeyType": "RANGE"}
  ]' >"$key_schema_json_file"

  echo '[
  {"AttributeName": "year", "AttributeType": "N"},
   {"AttributeName": "title", "AttributeType": "S"}
  ]' >"$attribute_definitions_json_file"

  if dynamodb_create_table -n "$table_name" -a "$attribute_definitions_json_file" \
    -k "$key_schema_json_file" 1>/dev/null; then
    echo "Created a DynamoDB table named $table_name"
  else
    errecho "The table failed to create. This demo will exit."
    clean_up
```

```
    return 1
  fi

  echo "Waiting for the table to become active...."

  if dynamodb_wait_table_active -n "$table_name"; then
    echo "The table is now active."
  else
    errecho "The table failed to become active. This demo will exit."
    cleanup "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo -n "Enter the title of a movie you want to add to the table: "
  get_input
  local added_title
  added_title=$get_input_result

  local added_year
  get_int_input "What year was it released? "
  added_year=$get_input_result

  local rating
  get_float_input "On a scale of 1 - 10, how do you rate it? " "1" "10"
  rating=$get_input_result

  local plot
  echo -n "Summarize the plot for me: "
  get_input
  plot=$get_input_result

  echo '{
    "year": {"N" :"'"$added_year"'"},
    "title": {"S" :  "'"$added_title"'"},
    "info": {"M" : {"plot": {"S" : "'"$plot"'"}, "rating": {"N" :"'"$rating"'"} } }
   }' >"$item_json_file"

  if dynamodb_put_item -n "$table_name" -i "$item_json_file"; then
    echo "The movie '$added_title' was successfully added to the table
'$table_name'."
```

```
  else
    errecho "Put item failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo "Let's update your movie '$added_title'."
  get_float_input "You rated it $rating, what new rating would you give it? " "1"
"10"
  rating=$get_input_result

  echo -n "You summarized the plot as '$plot'."
  echo "What would you say now? "
  get_input
  plot=$get_input_result

  echo '{
    "year": {"N" :"'"$added_year"'"},
    "title": {"S" :  "'"$added_title"'"}
    }' >"$key_json_file"

  echo '{
    ":r": {"N" :"'"$rating"'"},
    ":p": {"S" : "'"$plot"'"}
   }' >"$item_json_file"

  local update_expression="SET info.rating = :r, info.plot = :p"

  if dynamodb_update_item -n "$table_name" -k "$key_json_file" -e
"$update_expression" -v "$item_json_file"; then
    echo "Updated '$added_title' with new attributes."
  else
    errecho "Update item failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo
```

```
  echo "We will now use batch write to upload 150 movie entries into the table."

  local batch_json
  for batch_json in movie_files/movies_*.json; do
    echo "{ \"$table_name\" : $(<"$batch_json") }" >"$batch_json_file"
    if dynamodb_batch_write_item -i "$batch_json_file" 1>/dev/null; then
      echo "Entries in $batch_json added to table."
    else
      errecho "Batch write failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi
  done

  local title="The Lord of the Rings: The Fellowship of the Ring"
  local year="2001"

  if get_yes_no_input "Let's move on...do you want to get info about '$title'? (y/n)
"; then
    echo '{
  "year": {"N" :"'"$year"'"},
  "title": {"S" :  "'"$title"'"}
  }' >"$key_json_file"
    local info
    info=$(dynamodb_get_item -n "$table_name" -k "$key_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
      errecho "Get item failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi

    echo "Here is what I found:"
    echo "$info"
  fi

  local ask_for_year=true
  while [[ "$ask_for_year" == true ]]; do
    echo "Let's get a list of movies released in a given year."
    get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
    year=$get_input_result
    echo '{
```

```
    "#n": "year"
    }' >"$attribute_names_json_file"

    echo '{
    ":v": {"N" :"'"$year"'"}
    }' >"$attributes_values_json_file"

    response=$(dynamodb_query -n "$table_name" -k "#n=:v" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
      errecho "Query table failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi

    echo "Here is what I found:"
    echo "$response"

    if ! get_yes_no_input "Try another year? (y/n) "; then
      ask_for_year=false
    fi
  done

  echo "Now let's scan for movies released in a range of years. Enter a year: "
  get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
  local start=$get_input_result

  get_int_input "Enter another year: " "1972" "2018"
  local end=$get_input_result

  echo '{
    "#n": "year"
    }' >"$attribute_names_json_file"

  echo '{
    ":v1": {"N" : "'"$start"'"},
    ":v2": {"N" : "'"$end"'"}
    }' >"$attributes_values_json_file"

  response=$(dynamodb_scan -n "$table_name" -f "#n BETWEEN :v1 AND :v2" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")
```

```
  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "Scan table failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo "Here is what I found:"
  echo "$response"

  echo
  echo_repeat "*" 88
  echo

  echo "Let's remove your movie '$added_title' from the table."

  if get_yes_no_input "Do you want to remove '$added_title'? (y/n) "; then
    echo '{
  "year": {"N" :"'"$added_year"'"},
  "title": {"S" :  "'"$added_title"'"}
  }' >"$key_json_file"

    if ! dynamodb_delete_item -n "$table_name" -k "$key_json_file"; then
      errecho "Delete item failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi
  fi

  if get_yes_no_input "Do you want to delete the table '$table_name'? (y/n) "; then
    if ! clean_up "$table_name"; then
      return 1
    fi
  else
    if ! clean_up; then
      return 1
    fi
  fi

  return 0
}
```

As funções do DynamoDB usadas nesse cenário.

```
###############################################################################
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to create.
#       -a attribute_definitions -- JSON file path of a list of attributes and their
 types.
#       -k key_schema -- JSON file path of a list of attributes and their key types.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_create_table() {
  local table_name attribute_definitions key_schema response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_create_table"
    echo "Creates an Amazon DynamoDB table with on-demand billing."
    echo " -n table_name  -- The name of the table to create."
    echo " -a attribute_definitions -- JSON file path of a list of attributes and
 their types."
    echo " -k key_schema -- JSON file path of a list of attributes and their key
 types."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:a:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      a) attribute_definitions="${OPTARG}" ;;
      k) key_schema="${OPTARG}" ;;
      h)
        usage
        return 0
```

```
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_definitions" ]]; then
    errecho "ERROR: You must provide an attribute definitions json file path the -a
parameter."
    usage
    return 1
  fi

  if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    attribute_definitions:   $attribute_definitions"
  iecho "    key_schema:   $key_schema"
  iecho ""

  response=$(aws dynamodb create-table \
    --table-name "$table_name" \
    --attribute-definitions file://"$attribute_definitions" \
    --billing-mode PAY_PER_REQUEST \
    --key-schema file://"$key_schema" )

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
  fi

  return 0
}

##############################################################################
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#
#  Response:
#       - TableStatus:
#     And:
#       0 - Table is active.
#       1 - If it fails.
##############################################################################
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_describe_table"
    echo "Describe the status of a DynamoDB table."
    echo "  -n table_name  -- The name of the table."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
```

```
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  local table_status
    table_status=$(
      aws dynamodb describe-table \
        --table-name "$table_name" \
        --output text \
        --query 'Table.TableStatus'
    )

   local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log "$error_code"
      errecho "ERROR: AWS reports describe-table operation failed.$table_status"
      return 1
    fi

  echo "$table_status"

  return 0
}

###############################################################################
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -i item  -- Path to json file containing the item values.
```

```
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_put_item() {
  local table_name item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_put_item"
    echo "Put an item into a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -i item  -- Path to json file containing the item values."
    echo ""
  }

  while getopts "n:i:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$item" ]]; then
```

```
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""
  iecho ""

  response=$(aws dynamodb put-item \
    --table-name "$table_name" \
    --item file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports put-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 update.
#       -e update expression  -- An expression that defines one or more attributes
 to be updated.
#       -v values  -- Path to json file containing the update values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
###########################################################################
function dynamodb_update_item() {
  local table_name keys update_expression values response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_update_item"
    echo "Update an item in a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
 to update."
    echo " -e update expression  -- An expression that defines one or more
 attributes to be updated."
    echo " -v values  -- Path to json file containing the update values."
    echo ""
  }

  while getopts "n:k:e:v:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      e) update_expression="${OPTARG}" ;;
      v) values="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi
  if [[ -z "$update_expression" ]]; then
    errecho "ERROR: You must provide an update expression with the -e parameter."
    usage
    return 1
  fi

  if [[ -z "$values" ]]; then
    errecho "ERROR: You must provide a values json file path the -v parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho "    update_expression:   $update_expression"
  iecho "    values:   $values"

  response=$(aws dynamodb update-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --update-expression "$update_expression" \
    --expression-attribute-values file://"$values")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_batch_write_item
```

```
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the items to write.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_batch_write_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_batch_write_item"
    echo "Write a batch of items into a DynamoDB table."
    echo " -i item  -- Path to json file containing the items to write."
    echo ""
  }
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi
```

```
  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""

  response=$(aws dynamodb batch-write-item \
    --request-items file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  return 0
}

###########################################################################
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 get.
#       [-q query]  -- Optional JMESPath query expression.
#
#  Returns:
#       The item as text output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_get_item() {
  local table_name keys query response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
```

```
  function usage() {
    echo "function dynamodb_get_item"
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
  }
  query=""
  while getopts "n:k:q:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi

  if [[ -n "$query" ]]; then
    response=$(aws dynamodb get-item \
      --table-name "$table_name" \
      --key file://"$keys" \
```

```
      --output text \
      --query "$query")
  else
    response=$(
      aws dynamodb get-item \
        --table-name "$table_name" \
        --key file://"$keys" \
        --output text
    )
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports get-item operation failed.$response"
    return 1
  fi

  if [[ -n "$query" ]]; then
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
 query inserts on some strings.
  else
    echo "$response"
  fi

  return 0
}

###########################################################################
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k key_condition_expression -- The key condition expression.
#       -a attribute_names -- Path to JSON file containing the attribute names.
#       -v attribute_values -- Path to JSON file containing the attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
```

```
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_query() {
  local table_name key_condition_expression attribute_names attribute_values
 projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_query"
    echo "Query a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k key_condition_expression -- The key condition expression."
    echo " -a attribute_names -- Path to JSON file containing the attribute names."
    echo " -v attribute_values -- Path to JSON file containing the attribute
 values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) key_condition_expression="${OPTARG}" ;;
      a) attribute_names="${OPTARG}" ;;
      v) attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
```

```
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_values" ]]; then
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values")
  else
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
```

```
    return 1
  fi

  echo "$response"

  return 0
}

###########################################################################
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -f filter_expression  -- The filter expression.
#       -a expression_attribute_names -- Path to JSON file containing the expression
 attribute names.
#       -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_scan() {
  local table_name filter_expression expression_attribute_names
 expression_attribute_values projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_scan"
    echo "Scan a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -f filter_expression  -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
 expression attribute names."
```

```
    echo " -v expression_attribute_values -- Path to JSON file containing the
expression attribute values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      f) filter_expression="${OPTARG}" ;;
      a) expression_attribute_names="${OPTARG}" ;;
      v) expression_attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_names" ]]; then
    errecho "ERROR: You must provide expression attribute names with the -a
parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
 parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values")
  else
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
```

```
#       -k keys  -- Path to json file containing the keys that identify the item to
 delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_delete_item() {
  local table_name keys response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_delete_item"
    echo "Delete an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
 to delete."
    echo ""
  }
  while getopts "n:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho ""

  response=$(aws dynamodb delete-item \
    --table-name "$table_name" \
    --key file://"$keys")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_delete_table() {
  local table_name response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function dynamodb_delete_table"
  echo "Deletes an Amazon DynamoDB table."
  echo " -n table_name  -- The name of the table to delete."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho ""

response=$(aws dynamodb delete-table \
  --table-name "$table_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-table operation failed.$response"
  return 1
```

```
  fi

  return 0
}
```

As funções utilitárias usadas nesse cenário.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
```

```
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`BatchGetItem`

O código de exemplo a seguir mostra como usar `BatchGetItem`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_batch_get_item
#
# This function gets a batch of items from a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the keys of the items to get.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_batch_get_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_batch_get_item"
    echo "Get a batch of items from a DynamoDB table."
    echo " -i item  -- Path to json file containing the keys of the items to get."
    echo ""
  }
```

```
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  response=$(aws dynamodb batch-get-item \
    --request-items file://"$item")
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-get-item operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
```

```
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte BatchGetItem em Referência de AWS CLI comandos.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_batch_write_item
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the items to write.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function dynamodb_batch_write_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_batch_write_item"
    echo "Write a batch of items into a DynamoDB table."
    echo " -i item  -- Path to json file containing the items to write."
    echo ""
  }
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""

  response=$(aws dynamodb batch-write-item \
    --request-items file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function iecho
```

```
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
```

```
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte BatchWriteItem em Referência de AWS CLI comandos.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to create.
#       -a attribute_definitions -- JSON file path of a list of attributes and their
 types.
#       -k key_schema -- JSON file path of a list of attributes and their key types.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
```

```
function dynamodb_create_table() {
  local table_name attribute_definitions key_schema response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_create_table"
    echo "Creates an Amazon DynamoDB table with on-demand billing."
    echo " -n table_name  -- The name of the table to create."
    echo " -a attribute_definitions -- JSON file path of a list of attributes and
 their types."
    echo " -k key_schema -- JSON file path of a list of attributes and their key
 types."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:a:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      a) attribute_definitions="${OPTARG}" ;;
      k) key_schema="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_definitions" ]]; then
```

```
    errecho "ERROR: You must provide an attribute definitions json file path the -a
 parameter."
    usage
    return 1
  fi

  if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    attribute_definitions:   $attribute_definitions"
  iecho "    key_schema:   $key_schema"
  iecho ""

  response=$(aws dynamodb create-table \
    --table-name "$table_name" \
    --attribute-definitions file://"$attribute_definitions" \
    --billing-mode PAY_PER_REQUEST \
    --key-schema file://"$key_schema" )

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
```

```
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
```

```
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte CreateTable em Referência de AWS CLI comandos.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_delete_item() {
  local table_name keys response
  local option OPTARG # Required to use getopts command in a function.
```

```
# ######################################
# Function usage explanation
######################################
function usage() {
  echo "function dynamodb_delete_item"
  echo "Delete an item from a DynamoDB table."
  echo " -n table_name  -- The name of the table."
  echo " -k keys  -- Path to json file containing the keys that identify the item
to delete."
  echo ""
}
while getopts "n:k:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    k) keys="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    keys:   $keys"
iecho ""
```

```
  response=$(aws dynamodb delete-item \
    --table-name "$table_name" \
    --key file://"$keys")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
  fi

  return 0

}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
```

```
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteItem em Referência de AWS CLI comandos.

DeleteTable

O código de exemplo a seguir mostra como usar `DeleteTable`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_delete_table() {
  local table_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function dynamodb_delete_table"
    echo "Deletes an Amazon DynamoDB table."
    echo " -n table_name  -- The name of the table to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho ""

  response=$(aws dynamodb delete-table \
    --table-name "$table_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-table operation failed.$response"
    return 1
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
```

```
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
```

```
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteTable em Referência de AWS CLI comandos.

DescribeTable

O código de exemplo a seguir mostra como usar `DescribeTable`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#
#  Response:
#       - TableStatus:
#     And:
#       0 - Table is active.
#       1 - If it fails.
###############################################################################
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
```

```
  echo "function dynamodb_describe_table"
  echo "Describe the status of a DynamoDB table."
  echo "  -n table_name  -- The name of the table."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

local table_status
  table_status=$(
    aws dynamodb describe-table \
      --table-name "$table_name" \
      --output text \
      --query 'Table.TableStatus'
  )

 local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log "$error_code"
    errecho "ERROR: AWS reports describe-table operation failed.$table_status"
    return 1
  fi
```

```
  echo "$table_status"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
```

```
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeTable em Referência de AWS CLI comandos.

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 get.
#       [-q query]  -- Optional JMESPath query expression.
#
#  Returns:
#       The item as text output.
#  And:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function dynamodb_get_item() {
  local table_name keys query response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_get_item"
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
 to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
  }
  query=""
  while getopts "n:k:q:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
```

```
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi

  if [[ -n "$query" ]]; then
    response=$(aws dynamodb get-item \
      --table-name "$table_name" \
      --key file://"$keys" \
      --output text \
      --query "$query")
  else
    response=$(
      aws dynamodb get-item \
        --table-name "$table_name" \
        --key file://"$keys" \
        --output text
    )
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports get-item operation failed.$response"
    return 1
  fi

  if [[ -n "$query" ]]; then
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
 query inserts on some strings.
  else
    echo "$response"
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
```

```
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte GetItem em Referência de AWS CLI comandos.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function dynamodb_list_tables
#
# This function lists all the tables in a DynamoDB.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#############################################################################
function dynamodb_list_tables() {
  response=$(aws dynamodb list-tables \
    --output text \
    --query "TableNames")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  echo "$response" | tr -s "[:space:]" "\n"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
```

```
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte ListTables em Referência de AWS CLI comandos.

PutItem

O código de exemplo a seguir mostra como usar `PutItem`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -i item  -- Path to json file containing the item values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_put_item() {
  local table_name item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
```

```
function usage() {
  echo "function dynamodb_put_item"
  echo "Put an item into a DynamoDB table."
  echo " -n table_name  -- The name of the table."
  echo " -i item  -- Path to json file containing the item values."
  echo ""
}

while getopts "n:i:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    i) item="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$item" ]]; then
  errecho "ERROR: You must provide an item with the -i parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    item:   $item"
iecho ""
iecho ""

response=$(aws dynamodb put-item \
```

```
    --table-name "$table_name" \
    --item file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports put-item operation failed.$response"
    return 1
  fi

  return 0

}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
```

```
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte PutItem em Referência de AWS CLI comandos.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k key_condition_expression -- The key condition expression.
#       -a attribute_names -- Path to JSON file containing the attribute names.
#       -v attribute_values -- Path to JSON file containing the attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_query() {
  local table_name key_condition_expression attribute_names attribute_values
 projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_query"
    echo "Query a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k key_condition_expression -- The key condition expression."
    echo " -a attribute_names -- Path to JSON file containing the attribute names."
    echo " -v attribute_values -- Path to JSON file containing the attribute
 values."
```

```
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) key_condition_expression="${OPTARG}" ;;
      a) attribute_names="${OPTARG}" ;;
      v) attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_values" ]]; then
```

```
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values")
  else
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
```

```
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Consulte detalhes da API em Query na Referência de comandos da AWS CLI .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -f filter_expression  -- The filter expression.
#       -a expression_attribute_names -- Path to JSON file containing the expression
 attribute names.
#       -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_scan() {
  local table_name filter_expression expression_attribute_names
 expression_attribute_values projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_scan"
    echo "Scan a DynamoDB table."
```

```
    echo " -n table_name  -- The name of the table."
    echo " -f filter_expression  -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
expression attribute names."
    echo " -v expression_attribute_values -- Path to JSON file containing the
expression attribute values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      f) filter_expression="${OPTARG}" ;;
      a) expression_attribute_names="${OPTARG}" ;;
      v) expression_attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_names" ]]; then
```

```
    errecho "ERROR: You must provide expression attribute names with the -a
 parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
 parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values")
  else
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- Consulte detalhes da API em Scan na Referência de comandos da AWS CLI .

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 update.
#       -e update expression  -- An expression that defines one or more attributes
 to be updated.
#       -v values  -- Path to json file containing the update values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_update_item() {
  local table_name keys update_expression values response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
```

```
 ######################################
 function usage() {
   echo "function dynamodb_update_item"
   echo "Update an item in a DynamoDB table."
   echo " -n table_name  -- The name of the table."
   echo " -k keys  -- Path to json file containing the keys that identify the item
to update."
   echo " -e update expression  -- An expression that defines one or more
attributes to be updated."
   echo " -v values  -- Path to json file containing the update values."
   echo ""
 }

 while getopts "n:k:e:v:h" option; do
   case "${option}" in
     n) table_name="${OPTARG}" ;;
     k) keys="${OPTARG}" ;;
     e) update_expression="${OPTARG}" ;;
     v) values="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$table_name" ]]; then
   errecho "ERROR: You must provide a table name with the -n parameter."
   usage
   return 1
 fi

 if [[ -z "$keys" ]]; then
   errecho "ERROR: You must provide a keys json file path the -k parameter."
   usage
   return 1
 fi
 if [[ -z "$update_expression" ]]; then
```

```
    errecho "ERROR: You must provide an update expression with the -e parameter."
    usage
    return 1
  fi

  if [[ -z "$values" ]]; then
    errecho "ERROR: You must provide a values json file path the -v parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho "    update_expression:   $update_expression"
  iecho "    values:   $values"

  response=$(aws dynamodb update-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --update-expression "$update_expression" \
    --expression-attribute-values file://"$values")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
  fi

  return 0

}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
```

```
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
```

```
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência de AWS CLI comandos.

Cenários

Cenários avançados do índice secundário global

O exemplo de código a seguir mostra como trabalhar com configurações de índice secundário global (GSI).

- Crie uma tabela com vários GSIs.
- Crie uma tabela com capacidade e GSI sob demanda.
- Insira itens em uma tabela com vários GSIs.
- Consulte vários GSIs com condições diferentes.

AWS CLI com script Bash

Crie uma tabela com vários GSIs.

```
# Create a table with multiple GSIs
aws dynamodb create-table \
    --table-name MusicLibrary \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=AlbumTitle,AttributeType=S \
        AttributeName=Genre,AttributeType=S \
        AttributeName=Year,AttributeType=N \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
```

```
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"AlbumIndex\",
                \"KeySchema\": [{\"AttributeName\":\"AlbumTitle\",\"KeyType\":\"HASH
\"}],
                \"Projection\": {\"ProjectionType\":\"ALL\"}
            },
            {
                \"IndexName\": \"GenreYearIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"Year\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"INCLUDE\",\"NonKeyAttributes
\":[\"Artist\",\"SongTitle\"]}
            }
        ]"
```

Crie uma tabela com capacidade e GSI sob demanda.

```
# Create a table with on-demand capacity and GSI
aws dynamodb create-table \
    --table-name MusicOnDemand \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=Genre,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"GenreIndex\",
                \"KeySchema\": [{\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"}],
                \"Projection\": {\"ProjectionType\":\"ALL\"}
            }
        ]"
```

Insira itens em uma tabela com vários GSIs.

```
# Add items to MusicLibrary table
aws dynamodb put-item \
    --table-name MusicLibrary \
    --item '{
        "Artist": {"S": "The Beatles"},
        "SongTitle": {"S": "Hey Jude"},
        "AlbumTitle": {"S": "Past Masters"},
        "Genre": {"S": "Rock"},
        "Year": {"N": "1968"}
    }'

aws dynamodb put-item \
    --table-name MusicLibrary \
    --item '{
        "Artist": {"S": "Miles Davis"},
        "SongTitle": {"S": "So What"},
        "AlbumTitle": {"S": "Kind of Blue"},
        "Genre": {"S": "Jazz"},
        "Year": {"N": "1959"}
    }'
```

Consulte os itens de uma tabela com vários GSIs.

```
# Query the AlbumIndex GSI
echo "Querying AlbumIndex GSI:"
aws dynamodb query \
    --table-name MusicLibrary \
    --index-name AlbumIndex \
    --key-condition-expression "AlbumTitle = :album" \
    --expression-attribute-values '{":album":{"S":"Kind of Blue"}}'

# Query the GenreYearIndex GSI with a range condition
echo "Querying GenreYearIndex GSI with range condition:"
aws dynamodb query \
    --table-name MusicLibrary \
    --index-name GenreYearIndex \
    --key-condition-expression "Genre = :genre AND #yr > :year" \
    --expression-attribute-names '{"#yr": "Year"}' \
    --expression-attribute-values '{":genre":{"S":"Rock"},":year":{"N":"1965"}}'
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - PutItem
  - Consulta

Criar e gerenciar tabelas globais com MRSC

O exemplo de código a seguir mostra como criar e gerenciar tabelas globais do DynamoDB com consistência Multi-Region forte (MRSC).

- Crie uma tabela com consistência Multi-Region forte.
- Verifique a configuração de MRSC e o status da réplica.
- Teste a consistência forte entre regiões com leituras imediatas.
- Faça gravações condicionais com garantia de MRSC.
- Limpe os recursos da tabela global com MRSC.

AWS CLI com script Bash

Crie uma tabela com consistência Multi-Region forte.

```
# Step 1: Create a new table in us-east-2 (primary region for MRSC)
# Note: Table must be empty when enabling MRSC
aws dynamodb create-table \
    --table-name MusicTable \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --region us-east-2

# Wait for table to become active
aws dynamodb wait table-exists --table-name MusicTable --region us-east-2

# Step 2: Add replica and witness with Multi-Region Strong Consistency
# MRSC requires exactly three replicas in supported regions
aws dynamodb update-table \
```

```
    --table-name MusicTable \
    --replica-updates '[{"Create": {"RegionName": "us-east-1"}}]' \
    --global-table-witness-updates '[{"Create": {"RegionName": "us-west-2"}}]' \
    --multi-region-consistency STRONG \
    --region us-east-2
```

Verifique a configuração de MRSC e o status da réplica.

```
# Verify the global table configuration and MRSC setting
aws dynamodb describe-table \
    --table-name MusicTable \
    --region us-east-2 \
    --query 'Table.
{TableName:TableName,TableStatus:TableStatus,MultiRegionConsistency:MultiRegionConsistency,R
{Region:RegionName,Status:ReplicaStatus}}'
```

Teste a consistência forte com leituras imediatas entre regiões.

```
# Write an item to the primary region
aws dynamodb put-item \
    --table-name MusicTable \
    --item '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"},"Album":
 {"S":"The Beatles 1967-1970"},"Year": {"N":"1968"}}' \
    --region us-east-2

# Read the item from replica region to verify strong consistency (cannot read or
 write to witness)
# No wait time needed - MRSC provides immediate consistency
echo "Reading from us-east-1 (immediate consistency):"
aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"}}' \
    --consistent-read \
    --region us-east-1
```

Faça gravações condicionais com garantia de MRSC.

```
# Perform a conditional update from a different region
# This demonstrates that conditions work consistently across all regions
```

```
aws dynamodb update-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"}}' \
    --update-expression "SET #rating = :rating" \
    --condition-expression "attribute_exists(Artist)" \
    --expression-attribute-names '{"#rating": "Rating"}' \
    --expression-attribute-values '{":rating": {"N":"5"}}' \
    --region us-east-1
```

Limpe os recursos da tabela global com MRSC.

```
# Remove replica tables (must be done before deleting the primary table)
aws dynamodb update-table \
    --table-name MusicTable \
    --replica-updates '[{"Delete": {"RegionName": "us-east-1"}}]' \
    --global-table-witness-updates '[{"Delete": {"RegionName": "us-west-2"}}]' \
    --region us-east-2

# Wait for replicas to be deleted
echo "Waiting for replicas to be deleted..."
sleep 30

# Delete the primary table
aws dynamodb delete-table \
    --table-name MusicTable \
    --region us-east-2
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

Conceitos básicos de bancos de dados NoSQL

O exemplo de código a seguir mostra como:

- Criar uma tabela no DynamoDB.
- Atualizar os dados em uma tabela do DynamoDB.
- Consultar os dados em uma tabela do DynamoDB.
- Excluir uma tabela do DynamoDB.

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# DynamoDB Getting Started Tutorial Script
# This script demonstrates basic operations with Amazon DynamoDB:
# - Creating a table
# - Writing data to the table
# - Reading data from the table
# - Updating data in the table
# - Querying data in the table
# - Deleting the table (cleanup)

set -uo pipefail

# Set up logging with secure permissions
LOG_DIR="${XDG_STATE_HOME:-.}/dynamodb-tutorial-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/dynamodb-tutorial-$(date +%Y%m%d-%H%M%S).log"
chmod 700 "$LOG_DIR"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting DynamoDB Getting Started Tutorial at $(date)"
```

```
echo "Logging to $LOG_FILE"

# Validate AWS CLI is configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed or not in PATH"
    exit 1
fi

# Check AWS credentials are available
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd_name=$2

    if echo "$output" | grep -qi "error\|failed"; then
        echo "ERROR detected in $cmd_name command:" >&2
        echo "$output" >&2
        return 1
    fi
    return 0
}

# Function to wait for table to be in ACTIVE state
wait_for_table_active() {
    local table_name=$1
    local max_attempts=60
    local attempt=0
    local status=""

    echo "Waiting for table $table_name to become ACTIVE..."

    while [[ "$status" != "ACTIVE" && $attempt -lt $max_attempts ]]; do
        sleep 5
        status=$(aws dynamodb describe-table --table-name "$table_name" --query
 "Table.TableStatus" --output text 2>/dev/null || echo "UNKNOWN")
        echo "Current status: $status"
        ((attempt++))
    done
```

```
    if [[ "$status" != "ACTIVE" ]]; then
        echo "ERROR: Table $table_name did not become ACTIVE within timeout period"
 >&2
        return 1
    fi

    echo "Table $table_name is now ACTIVE"
    return 0
}

# Track created resources for cleanup
declare -a RESOURCES=()

# Cleanup function
cleanup() {
    local exit_code=$?

    if [[ $exit_code -ne 0 ]]; then
        echo "Script encountered an error (exit code: $exit_code)" >&2
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP"
    echo "============================================"
    echo "Resources to clean up:"
    for resource in "${RESOURCES[@]+"${RESOURCES[@]}"}"; do
        echo "- $resource"
    done
    echo ""

    if [[ ${#RESOURCES[@]} -gt 0 ]]; then
        echo "Proceeding with cleanup of all created resources..."

        for resource in "${RESOURCES[@]+"${RESOURCES[@]}"}"; do
            if [[ "$resource" == Table:* ]]; then
                local table_name="${resource#Table:}"
                echo "Deleting table: $table_name"
                if aws dynamodb delete-table --table-name "$table_name" 2>/dev/null;
 then
                    echo "Waiting for table deletion to complete..."
                    aws dynamodb wait table-not-exists --table-name "$table_name"
 2>/dev/null || true
                else
```

```
                    echo "Warning: Failed to delete table $table_name" >&2
                fi
            fi
        done

        echo "Cleanup completed."
    fi

    return $exit_code
}

trap cleanup EXIT

# Validate table name
validate_table_name() {
    local name=$1
    if [[ ! $name =~ ^[a-zA-Z0-9._-]+$ ]] || [[ ${#name} -gt 255 ]]; then
        echo "ERROR: Invalid table name: $name" >&2
        return 1
    fi
    return 0
}

# Step 1: Create a table in DynamoDB
echo "Step 1: Creating Music table in DynamoDB..."

TABLE_NAME="Music"
validate_table_name "$TABLE_NAME"

CREATE_TABLE_OUTPUT=$(aws dynamodb create-table \
    --table-name "$TABLE_NAME" \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema AttributeName=Artist,KeyType=HASH
 AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --table-class STANDARD \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-dynamodb-gs 2>&1)
 || {
    echo "ERROR: Failed to create table" >&2
    exit 1
}
```

```
check_error "$CREATE_TABLE_OUTPUT" "create-table"
echo "$CREATE_TABLE_OUTPUT"

# Add table to resources list
RESOURCES+=("Table:$TABLE_NAME")

# Wait for table to be active
wait_for_table_active "$TABLE_NAME"

# Enable point-in-time recovery (best practice)
echo "Enabling point-in-time recovery for the $TABLE_NAME table..."

PITR_OUTPUT=$(aws dynamodb update-continuous-backups \
    --table-name "$TABLE_NAME" \
    --point-in-time-recovery-specification PointInTimeRecoveryEnabled=true 2>&1) ||
 {
    echo "ERROR: Failed to enable PITR" >&2
    exit 1
}

check_error "$PITR_OUTPUT" "update-continuous-backups"
echo "$PITR_OUTPUT"

# Step 2: Write data to the DynamoDB table
echo "Step 2: Writing data to the $TABLE_NAME table..."

# Use a temporary file for item data
ITEMS_TEMP=$(mktemp)
trap "rm -f '$ITEMS_TEMP'" EXIT

cat > "$ITEMS_TEMP" << 'EOF'
{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Call Me Today"},
 "AlbumTitle": {"S": "Somewhat Famous"}, "Awards": {"N": "1"}}
{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Howdy"}, "AlbumTitle":
 {"S": "Somewhat Famous"}, "Awards": {"N": "2"}}
{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}, "AlbumTitle": {"S":
 "Songs About Life"}, "Awards": {"N": "10"}}
{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "PartiQL Rocks"}, "AlbumTitle":
 {"S": "Another Album Title"}, "Awards": {"N": "8"}}
EOF

declare -i item_num=0
while IFS= read -r item_data; do
    ((item_num++))
```

```
    ITEM_OUTPUT=$(aws dynamodb put-item \
        --table-name "$TABLE_NAME" \
        --item "$item_data" 2>&1) || {
        echo "ERROR: Failed to put item $item_num" >&2
        exit 1
    }
    check_error "$ITEM_OUTPUT" "put-item (item $item_num)"
    echo "Item $item_num added successfully"
done < "$ITEMS_TEMP"

# Step 3: Read data from the DynamoDB table
echo "Step 3: Reading data from the $TABLE_NAME table..."

# Get a specific item
GET_ITEM_OUTPUT=$(aws dynamodb get-item --consistent-read \
    --table-name "$TABLE_NAME" \
    --key '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}}' 2>&1) ||
 {
    echo "ERROR: Failed to get item" >&2
    exit 1
}

check_error "$GET_ITEM_OUTPUT" "get-item"
echo "Retrieved item:"
echo "$GET_ITEM_OUTPUT"

# Step 4: Update data in the DynamoDB table
echo "Step 4: Updating data in the $TABLE_NAME table..."

# Update an item
UPDATE_ITEM_OUTPUT=$(aws dynamodb update-item \
    --table-name "$TABLE_NAME" \
    --key '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}}' \
    --update-expression "SET AlbumTitle = :newval" \
    --expression-attribute-values '{":newval": {"S": "Updated Album Title"}}' \
    --return-values ALL_NEW 2>&1) || {
    echo "ERROR: Failed to update item" >&2
    exit 1
}

check_error "$UPDATE_ITEM_OUTPUT" "update-item"
echo "Updated item:"
echo "$UPDATE_ITEM_OUTPUT"
```

```
# Step 5: Query data in the DynamoDB table
echo "Step 5: Querying data in the $TABLE_NAME table..."

# Query items by Artist
QUERY_OUTPUT=$(aws dynamodb query \
    --table-name "$TABLE_NAME" \
    --key-condition-expression "Artist = :name" \
    --expression-attribute-values '{":name": {"S": "Acme Band"}}' 2>&1) || {
    echo "ERROR: Failed to query table" >&2
    exit 1
}

check_error "$QUERY_OUTPUT" "query"
echo "Query results:"
echo "$QUERY_OUTPUT"

echo "DynamoDB Getting Started Tutorial completed successfully at $(date)"
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Consulta
  - UpdateContinuousBackups
  - UpdateItem
  - Wait

Gerenciar índices secundários globais

O exemplo de código a seguir mostra como gerenciar o ciclo de vida completo dos índices secundários globais.

- Crie uma tabela com um índice secundário global.
- Adicione um novo GSI a uma tabela existente.
- Atualize (aumente) o throughput a quente do GSI.

- Consulte dados usando GSIs.
- Exclua um GSI.

AWS CLI com script Bash

Crie uma tabela com um índice secundário global.

```
# Create a table with a GSI
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=AlbumTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "IndexName=AlbumIndex,\
        KeySchema=[{AttributeName=AlbumTitle,KeyType=HASH}],\
        Projection={ProjectionType=ALL}"
```

Adicione um novo GSI (sob demanda) a uma tabela existente.

```
# Add a new GSI to an existing table
aws dynamodb update-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Genre,AttributeType=S \
    --global-secondary-index-updates \
        "[{\"Create\":{\"IndexName\":\"GenreIndex\",\
        \"KeySchema\":[{\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"}],\
        \"Projection\":{\"ProjectionType\":\"ALL\"}}}]"
```

Atualize (aumente) o throughput a quente do GSI.

```
# Increase the warm throughput of a GSI (default values are 12k reads, 4k writes)
aws dynamodb update-table \
```

```
    --table-name MusicCollection \
    --global-secondary-index-updates \
        "[{\"Update\":{\"IndexName\":\"AlbumIndex\",\
        \"WarmThroughput\":{\"ReadUnitsPerSecond\":15000,\"WriteUnitsPerSecond
\":6000}}}]"
```

Consulte dados usando GSIs.

```
# Query the AlbumIndex GSI
aws dynamodb query \
    --table-name MusicCollection \
    --index-name AlbumIndex \
    --key-condition-expression "AlbumTitle = :album" \
    --expression-attribute-values '{":album":{"S":"Let It Be"}}'

# Query the GenreIndex GSI
aws dynamodb query \
    --table-name MusicCollection \
    --index-name GenreIndex \
    --key-condition-expression "Genre = :genre" \
    --expression-attribute-values '{":genre":{"S":"Jazz"}}'
```

Exclua um GSI.

```
# Delete a GSI from a table
aws dynamodb update-table \
    --table-name MusicCollection \
    --global-secondary-index-updates \
        "[{\"Delete\":{\"IndexName\":\"GenreIndex\"}}]"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteTable
  - Consulta
  - UpdateTable

Gerenciar políticas baseadas em recursos

O exemplo de código a seguir mostra como gerenciar o ciclo de vida completo das políticas baseadas em recursos para tabelas do DynamoDB.

- Crie uma tabela com uma política de recursos.
- Obtenha uma política de recursos.
- Atualize uma política de recursos.
- Exclua uma política de recursos.

AWS CLI com script Bash

Crie uma tabela com uma política de recursos.

```
# Step 1: Create a DynamoDB table
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST

# Step 2: Create a resource-based policy document
cat > policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/DynamoDBReadOnly"
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
```

```
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection"
    }
  ]
}
EOF

# Step 3: Attach the resource-based policy to the table
aws dynamodb put-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \
    --policy file://policy.json
```

Obtenha uma política de recursos.

```
# Get the resource-based policy attached to a table
aws dynamodb get-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

Atualize uma política de recursos.

```
# Step 1: Create an updated policy document
cat > updated-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:role/DynamoDBReadOnly",
          "arn:aws:iam::123456789012:role/DynamoDBAnalytics"
        ]
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection"
    }
  ]
```

```
}
EOF

# Step 2: Update the resource-based policy on the table
aws dynamodb put-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \
    --policy file://updated-policy.json
```

Exclua uma política de recursos.

```
# Delete the resource-based policy from a table
aws dynamodb delete-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteResourcePolicy
  - GetResourcePolicy
  - PutResourcePolicy

Configurar o controle de Attribute-Based acesso

O exemplo de código a seguir mostra como implementar o Controle de Attribute-Based Acesso (ABAC) para o DynamoDB.

- Crie uma política do IAM para o ABAC.
- Crie tabelas com etiquetas que representem diferentes departamentos.
- Liste e filtre tabelas com base nas etiquetas.

AWS CLI com script Bash

Crie uma política do IAM para o ABAC.

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

Crie tabelas com etiquetas que representem diferentes departamentos.

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

Liste e filtre tabelas com base nas etiquetas.

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text | xargs
 -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn" --output
 text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
```

```
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreatePolicy
  - CreateTable
  - ListTables

Trabalhar com índices secundários locais

O exemplo de código a seguir mostra como criar e consultar tabelas com índices secundários locais.

- Crie uma tabela com um índice secundário local (LSI).
- Crie uma tabela com vários LSIs com diferentes tipos de projeção.
- Consulte dados usando LSIs.

AWS CLI com script Bash

Crie uma tabela com um índice secundário local.

```
# Create a table with a Local Secondary Index
aws dynamodb create-table \
    --table-name CustomerOrders \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
        AttributeName=OrderID,AttributeType=S \
        AttributeName=OrderDate,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
```

```
        AttributeName=OrderID,KeyType=RANGE \
    --local-secondary-indexes \
        "IndexName=OrderDateIndex,\
        KeySchema=[{AttributeName=CustomerID,KeyType=HASH},
{AttributeName=OrderDate,KeyType=RANGE}],\
        Projection={ProjectionType=ALL}" \
    --billing-mode PAY_PER_REQUEST
```

Crie uma tabela com vários LSIs.

```
# Create a table with multiple Local Secondary Indexes
aws dynamodb create-table \
    --table-name CustomerDetails \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
        AttributeName=Name,AttributeType=S \
        AttributeName=Email,AttributeType=S \
        AttributeName=RegistrationDate,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
        AttributeName=Name,KeyType=RANGE \
    --local-secondary-indexes \
        "[
            {
                \"IndexName\": \"EmailIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"CustomerID\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"Email\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"INCLUDE\",\"NonKeyAttributes
\":[\"Address\",\"Phone\"]}
            },
            {
                \"IndexName\": \"RegistrationIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"CustomerID\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"RegistrationDate\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"KEYS_ONLY\"}
            }
        ]" \
    --billing-mode PAY_PER_REQUEST
```

Consulte dados usando LSIs.

```
# Query the OrderDateIndex LSI
aws dynamodb query \
    --table-name CustomerOrders \
    --index-name OrderDateIndex \
    --key-condition-expression "CustomerID = :custId AND OrderDate BETWEEN :date1
 AND :date2" \
    --expression-attribute-values '{
        ":custId": {"S": "C1"},
        ":date1": {"S": "2023-01-01"},
        ":date2": {"S": "2023-02-01"}
    }'

# Query with a filter expression
aws dynamodb query \
    --table-name CustomerOrders \
    --index-name OrderDateIndex \
    --key-condition-expression "CustomerID = :custId" \
    --filter-expression "Amount > :amount" \
    --expression-attribute-values '{
        ":custId": {"S": "C1"},
        ":amount": {"N": "150"}
    }'
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - Consulta

Trabalhe com Streams e Time-to-Live

O exemplo de código a seguir mostra como gerenciar streams e recursos do DynamoDB. Time-to-Live

- Crie uma tabela com a opção de fluxos habilitada.
- Descreva os fluxos.
- Crie uma função do Lambda para processar os fluxos.

- Habilite a vida útil (TTL) em uma tabela.
- Adicione itens com atributos de TTL.
- Descreva as configurações de TTL.

AWS CLI com script Bash

Crie uma tabela com a opção de fluxos habilitada.

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

Descreva os fluxos.

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)

echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

Crie uma função do Lambda para os fluxos.

```
# Step 1: Create an IAM role for the Lambda function
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
echo "    --starting-position LATEST"
```

Habilite a vida útil (TTL) em uma tabela.

```
# Create a table for TTL demonstration
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
```

```
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo

# Enable TTL on the table
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

Adicione itens com atributos de TTL.

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

Descreva as configurações de TTL.

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
```

```
    --table-name TTLDemo
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole
  - CreateTable
  - DescribeTable
  - DescribeTimeToLive
  - PutItem
  - UpdateTimeToLive

Trabalhar com tabelas globais e a consistência final de replicação mutirregional (MREC)

O exemplo de código a seguir mostra como gerenciar tabelas globais do DynamoDB com consistência final de replicação mutirregional (MREC).

- Crie uma tabela com replicação mutirregional (MREC).
- Insira e obtenha itens de tabelas-réplica.
- Remova as réplicas uma por uma.
- Exclua a tabela para fazer uma limpeza.

AWS CLI com script Bash

Crie uma tabela com replicação mutirregional.

```
# Step 1: Create a new table (MusicTable) in US East (Ohio), with DynamoDB Streams
 enabled (NEW_AND_OLD_IMAGES)
aws dynamodb create-table \
    --table-name MusicTable \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES \
```

```
    --region us-east-2

# Step 2: Create an identical MusicTable table in US East (N. Virginia)
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Create": {
        "RegionName": "us-east-1"
      }
    }
  ]
}' \
--region us-east-2

# Step 3: Create a table in Europe (Ireland)
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Create": {
        "RegionName": "eu-west-1"
      }
    }
  ]
}' \
--region us-east-2
```

Descreva a tabela multirregional.

```
# Step 4: View the list of replicas created using describe-table
aws dynamodb describe-table \
    --table-name MusicTable \
    --region us-east-2 \
    --query 'Table.
{TableName:TableName,TableStatus:TableStatus,MultiRegionConsistency:MultiRegionConsistency,R
{Region:RegionName,Status:ReplicaStatus}}'
```

Insira itens em uma tabela-réplica.

```
# Step 5: To verify that replication is working, add a new item to the Music table
 in US East (Ohio)
aws dynamodb put-item \
    --table-name MusicTable \
    --item '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region us-east-2
```

Obtenha itens de tabelas-réplica.

```
# Step 6: Wait for a few seconds, and then check to see whether the item has been
# successfully replicated to US East (N. Virginia) and Europe (Ireland)
aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region us-east-1

aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region eu-west-1
```

Remova as réplicas.

```
# Step 7: Delete the replica table in Europe (Ireland) Region
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Delete": {
        "RegionName": "eu-west-1"
      }
    }
  ]
}' \
--region us-east-2

# Delete the replica table in US East (N. Virginia) Region
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
```

```
  "ReplicaUpdates":
  [
    {
      "Delete": {
        "RegionName": "us-east-1"
      }
    }
  ]
}' \
--region us-east-2
```

Exclua a tabela para fazer uma limpeza.

```
# Clean up: Delete the primary table
aws dynamodb delete-table --table-name MusicTable --region us-east-2

echo "Global table demonstration complete."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateTable

Trabalhar com a marcação de recursos

O exemplo de código a seguir mostra como gerenciar tags para recursos do DynamoDB.

- Crie uma tabela com tags.
- Listar as etiquetas de um recurso.
- Adicione tags a um recurso.
- Remova as tags de um recurso.
- Filtre tabelas por tags.

AWS CLI com script Bash

Crie uma tabela com tags.

```
# Create a table with tags
aws dynamodb create-table \
    --table-name TaggedTable \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Environment,Value=Production \
        Key=Project,Value=Analytics \
        Key=Owner,Value=DataTeam
```

Listar as etiquetas de um recurso.

```
# Get the table ARN
TABLE_ARN=$(aws dynamodb describe-table \
    --table-name TaggedTable \
    --query "Table.TableArn" \
    --output text)

# List tags for the table
aws dynamodb list-tags-of-resource \
    --resource-arn $TABLE_ARN
```

Adicione tags a um recurso.

```
# Add tags to an existing table
aws dynamodb tag-resource \
    --resource-arn $TABLE_ARN \
    --tags \
        Key=CostCenter,Value=12345 \
        Key=BackupSchedule,Value=Daily
```

Remova as tags de um recurso.

```
# Remove tags from a table
aws dynamodb untag-resource \
    --resource-arn $TABLE_ARN \
    --tag-keys Owner BackupSchedule
```

Filtre tabelas por tags.

```
# Create another table with different tags
aws dynamodb create-table \
    --table-name AnotherTaggedTable \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Environment,Value=Development \
        Key=Project,Value=Testing

# Wait for table to become active
aws dynamodb wait table-exists --table-name AnotherTaggedTable

# List all tables
echo "All tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nFiltering tables by Environment=Production tag:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text | xargs
 -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn" --output
 text)

# Find tables with specific tag
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTable
  - ListTagsOfResource
  - TagResource
  - UntagResource

Trabalhar com a criptografia de tabelas

O exemplo de código a seguir mostra como gerenciar opções de criptografia para tabelas do DynamoDB.

- Crie uma tabela com a criptografia padrão.
- Crie uma tabela com uma chave gerenciada pelo cliente (CMK).
- Atualize as configurações de criptografia da tabela.
- Descreva a criptografia da tabela.

AWS CLI com script Bash

Crie uma tabela com a criptografia padrão.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Crie uma tabela com uma chave gerenciada pelo cliente (CMK).

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
```

```
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Atualize a criptografia da tabela.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Descreva a criptografia da tabela.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateKey
  - CreateTable
  - DescribeTable
  - UpdateTable

Exemplos do Amazon EC2 usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon EC2.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
###############################################################################
# function get_started_with_ec2_instances
#
# Runs an interactive scenario that shows how to get started using EC2 instances.
#
#     "EC2 access" permissions are needed to run this code.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function get_started_with_ec2_instances() {
  # Requires version 4 for mapfile.
  local required_version=4.0

  # Get the current Bash version
  # Check if BASH_VERSION is set
  local current_version
  if [[ -n "$BASH_VERSION" ]]; then
    # Convert BASH_VERSION to a number for comparison
    current_version=$BASH_VERSION
  else
    # Get the current Bash version using the bash command
    current_version=$(bash --version | head -n 1 | awk '{ print $4 }')
  fi

  # Convert version strings to numbers for comparison
  local required_version_num current_version_num
  required_version_num=$(echo "$required_version" | awk -F. '{ print ($1 * 10000) +
 ($2 * 100) + $3 }')
  current_version_num=$(echo "$current_version" | awk -F. '{ print ($1 * 10000) +
 ($2 * 100) + $3 }')
```

```
  # Compare versions
  if ((current_version_num < required_version_num)); then
    echo "Error: This script requires Bash version $required_version or higher."
    echo "Your current Bash version is number is $current_version."
    exit 1
  fi

  {
    if [ "$EC2_OPERATIONS_SOURCED" != "True" ]; then

      source ./ec2_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started with
instances demo."
  echo_repeat "*" 88
  echo

  echo "Let's create an RSA key pair that you can be use to securely connect to "
  echo "your EC2 instance."

  echo -n "Enter a unique name for your key: "
  get_input
  local key_name
  key_name=$get_input_result

  local temp_dir
  temp_dir=$(mktemp -d)
  local key_file_name="$temp_dir/${key_name}.pem"

  if ec2_create_keypair -n "${key_name}" -f "${key_file_name}"; then
    echo "Created a key pair $key_name and saved the private key to $key_file_name"
    echo
  else
    errecho "The key pair failed to create. This demo will exit."
    return 1
  fi

  chmod 400 "${key_file_name}"

  if yes_no_input "Do you want to list some of your key pairs? (y/n) "; then
    local keys_and_fingerprints
```

```
    keys_and_fingerprints="$(ec2_describe_key_pairs)" && {
      local image_name_and_id
      while IFS=$'\n' read -r image_name_and_id; do
        local entries
        IFS=$'\t' read -ra entries <<<"$image_name_and_id"
        echo "Found rsa key ${entries[0]} with fingerprint:"
        echo "     ${entries[1]}"
      done <<<"$keys_and_fingerprints"

    }
  fi

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's create a security group to manage access to your instance."
  echo -n "Enter a unique name for your security group: "
  get_input
  local security_group_name
  security_group_name=$get_input_result
  local security_group_id
  security_group_id=$(ec2_create_security_group -n "$security_group_name" \
    -d "Security group for EC2 instance") || {
    errecho "The security failed to create. This demo will exit."
    clean_up "$key_name" "$key_file_name"
    return 1
  }

  echo "Security group created with ID $security_group_id"
  echo

  local public_ip
  public_ip=$(curl -s http://checkip.amazonaws.com)

  echo "Let's add a rule to allow SSH only from your current IP address."
  echo "Your public IP address is $public_ip"
  echo -n "press return to add this rule to your security group."
  get_input

  if ! ec2_authorize_security_group_ingress -g "$security_group_id" -i "$public_ip"
-p tcp -f 22 -t 22; then
    errecho "The security group rules failed to update. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
```

```
  fi

  echo "Security group rules updated"

  local security_group_description
  security_group_description="$(ec2_describe_security_groups -g
 "${security_group_id}")" || {
    errecho "Failed to describe security groups. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  mapfile -t parameters <<<"$security_group_description"
  IFS=$'\t' read -ra entries <<<"${parameters[0]}"
  echo "Security group: ${entries[0]}"
  echo "    ID: ${entries[1]}"
  echo "    VPC: ${entries[2]}"
  echo "Inbound permissions:"
  IFS=$'\t' read -ra entries <<<"${parameters[1]}"
  echo "    IpProtocol: ${entries[0]}"
  echo "    FromPort: ${entries[1]}"
  echo "    ToPort: ${entries[2]}"
  echo "    CidrIp: ${parameters[2]}"

  local parameters
  parameters="$(ssm_get_parameters_by_path -p "/aws/service/ami-amazon-linux-
latest")" || {
    errecho "Failed to get parameters. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1

  }

  local image_ids=""
  mapfile -t parameters <<<"$parameters"
  for image_name_and_id in "${parameters[@]}"; do
    IFS=$'\t' read -ra values <<<"$image_name_and_id"
    if [[ "${values[0]}" == *"amzn2"* ]]; then
      image_ids+="${values[1]} "
    fi
  done

  local images
  images="$(ec2_describe_images -i "$image_ids")" || {
```

```
    errecho "Failed to describe images. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1

  }

  new_line_and_tab_to_list "$images"
  local images=("${list_result[@]}")

  # Get the size of the array
  local images_count=${#images[@]}

  if ((images_count == 0)); then
    errecho "No images found. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  fi

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's create an instance from an Amazon Linux 2 AMI. Here are some options:"
  for ((i = 0; i < images_count; i += 3)); do
    echo "$(((i / 3) + 1)) - ${images[$i]}"
  done

  integer_input "Please enter the number of the AMI you want to use: " 1
"$((images_count / 3))"
  local choice=$get_input_result
  choice=$(((choice - 1) * 3))

  echo "Great choice."
  echo

  local architecture=${images[$((choice + 1))]}
  local image_id=${images[$((choice + 2))]}
  echo "Here are some instance types that support the ${architecture} architecture
of the image:"
  response="$(ec2_describe_instance_types -a "${architecture}" -t
"*.micro,*.small")" || {
    errecho "Failed to describe instance types. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }
```

```
  local instance_types
  mapfile -t instance_types <<<"$response"

  # Get the size of the array
  local instance_types_count=${#instance_types[@]}

  echo "Here are some options:"
  for ((i = 0; i < instance_types_count; i++)); do
    echo "$((i + 1)) - ${instance_types[$i]}"
  done

  integer_input "Which one do you want to use? " 1 "${#instance_types[@]}
"
  choice=$get_input_result
  local instance_type=${instance_types[$((choice - 1))]}
  echo "Another great choice."
  echo

  echo "Creating your instance and waiting for it to start..."
  local instance_id
  instance_id=$(ec2_run_instances -i "$image_id" -t "$instance_type" -k "$key_name"
 -s "$security_group_id") || {
    errecho "Failed to run instance. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  ec2_wait_for_instance_running -i "$instance_id"
  echo "Your instance is ready:"
  echo

  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  echo
  print_instance_details "${instance_details}"

  local public_ip
  public_ip=$(echo "${instance_details}" | awk '{print $6}')
  echo
  echo "You can use SSH to connect to your instance"
  echo "If the connection attempt times out, you might have to manually update the
 SSH ingress rule"
```

```
echo "for your IP address in the AWS Management Console."
connect_to_instance "$key_file_name" "$public_ip"

echo -n "Press Enter when you're ready to continue the demo: "
get_input

echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's stop and start your instance to see what changes."
echo "Stopping your instance and waiting until it's stopped..."
ec2_stop_instances -i "$instance_id"
ec2_wait_for_instance_stopped -i "$instance_id"

echo "Your instance is stopped. Restarting..."

ec2_start_instances -i "$instance_id"
ec2_wait_for_instance_running -i "$instance_id"

echo "Your instance is running again."
local instance_details
instance_details="$(ec2_describe_instances -i "${instance_id}")"

print_instance_details "${instance_details}"

public_ip=$(echo "${instance_details}" | awk '{print $6}')

echo "Every time your instance is restarted, its public IP address changes"
connect_to_instance "$key_file_name" "$public_ip"

echo -n "Press Enter when you're ready to continue the demo: "
get_input

echo_repeat "*" 88
echo_repeat "*" 88

echo "You can allocate an Elastic IP address and associate it with your instance"
echo "to keep a consistent IP address even when your instance restarts."

local result
result=$(ec2_allocate_address -d vpc) || {
  errecho "Failed to allocate an address. This demo will exit."
  clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
  return 1
```

```
  }

  local elastic_ip allocation_id
  elastic_ip=$(echo "$result" | awk '{print $1}')
  allocation_id=$(echo "$result" | awk '{print $2}')

  echo "Allocated static Elastic IP address: $elastic_ip"

  local association_id
  association_id=$(ec2_associate_address -i "$instance_id" -a "$allocation_id") || {
    errecho "Failed to associate an address. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
"$allocation_id"
    return 1
  }

  echo "Associated your Elastic IP with your instance."
  echo "You can now use SSH to connect to your instance by using the Elastic IP."
  connect_to_instance "$key_file_name" "$elastic_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's stop and start your instance to see what changes."
  echo "Stopping your instance and waiting until it's stopped..."
  ec2_stop_instances -i "$instance_id"
  ec2_wait_for_instance_stopped -i "$instance_id"

  echo "Your instance is stopped. Restarting..."

  ec2_start_instances -i "$instance_id"
  ec2_wait_for_instance_running -i "$instance_id"

  echo "Your instance is running again."
  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  print_instance_details "${instance_details}"

  echo "Because you have associated an Elastic IP with your instance, you can"
  echo "connect by using a consistent IP address after the instance restarts."
```

```
  connect_to_instance "$key_file_name" "$elastic_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  if yes_no_input "Do you want to delete the resources created in this demo: (y/n)
 "; then
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id" \
      "$allocation_id" "$association_id"
  else
    echo "The following resources were not deleted."
    echo "Key pair: $key_name"
    echo "Key file: $key_file_name"
    echo "Security group: $security_group_id"
    echo "Instance: $instance_id"
    echo "Elastic IP address: $elastic_ip"
  fi
}

###############################################################################
# function clean_up
#
# This function cleans up the created resources.
#     $1 - The name of the ec2 key pair to delete.
#     $2 - The name of the key file to delete.
#     $3 - The ID of the security group to delete.
#     $4 - The ID of the instance to terminate.
#     $5 - The ID of the elastic IP address to release.
#     $6 - The ID of the elastic IP address to disassociate.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function clean_up() {
  local result=0
  local key_pair_name=$1
  local key_file_name=$2
  local security_group_id=$3
  local instance_id=$4
  local allocation_id=$5
```

```
  local association_id=$6

  if [ -n "$association_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_disassociate_address -a "$association_id"); then
      echo "Disassociated elastic IP address with ID $association_id"
    else
      errecho "The elastic IP address disassociation failed."
      result=1
    fi
  fi

  if [ -n "$allocation_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_release_address -a "$allocation_id"); then
      echo "Released elastic IP address with ID $allocation_id"
    else
      errecho "The elastic IP address release failed."
      result=1
    fi
  fi

  if [ -n "$instance_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_terminate_instances -i "$instance_id"); then
      echo "Started terminating instance with ID $instance_id"

      ec2_wait_for_instance_terminated -i "$instance_id"
    else
      errecho "The instance terminate failed."
      result=1
    fi
  fi

  if [ -n "$security_group_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_security_group -i "$security_group_id"); then
      echo "Deleted security group with ID $security_group_id"
    else
      errecho "The security group delete failed."
      result=1
    fi
  fi
```

```
  if [ -n "$key_pair_name" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_keypair -n "$key_pair_name"); then
      echo "Deleted key pair named $key_pair_name"
    else
      errecho "The key pair delete failed."
      result=1
    fi
  fi

  if [ -n "$key_file_name" ]; then
    rm -f "$key_file_name"
  fi

  return $result
}

###############################################################################
# function ssm_get_parameters_by_path
#
# This function retrieves one or more parameters from the AWS Systems Manager
 Parameter Store
# by specifying a parameter path.
#
# Parameters:
#       -p parameter_path - The path of the parameter(s) to retrieve.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ssm_get_parameters_by_path() {
  local parameter_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ssm_get_parameters_by_path"
    echo "Retrieves one or more parameters from the AWS Systems Manager Parameter
 Store by specifying a parameter path."
    echo "  -p parameter_path - The path of the parameter(s) to retrieve."
    echo ""
  }
```

```
  # Retrieve the calling parameters.
  while getopts "p:h" option; do
    case "${option}" in
      p) parameter_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$parameter_path" ]]; then
    errecho "ERROR: You must provide a parameter path with the -p parameter."
    usage
    return 1
  fi

  response=$(aws ssm get-parameters-by-path \
    --path "$parameter_path" \
    --query "Parameters[*].[Name, Value]" \
    --output text) || {
    aws_cli_error_log $?
    errecho "ERROR: AWS reports get-parameters-by-path operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function print_instance_details
#
# This function prints the details of an Amazon Elastic Compute Cloud (Amazon EC2)
 instance.
#
# Parameters:
```

```
#       instance_details - The instance details in the format "InstanceId ImageId
 InstanceType KeyName VpcId PublicIpAddress State.Name".
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function print_instance_details() {
  local instance_details="$1"

  if [[ -z "${instance_details}" ]]; then
    echo "Error: Missing required instance details argument."
    return 1
  fi

  local instance_id image_id instance_type key_name vpc_id public_ip state
  instance_id=$(echo "${instance_details}" | awk '{print $1}')
  image_id=$(echo "${instance_details}" | awk '{print $2}')
  instance_type=$(echo "${instance_details}" | awk '{print $3}')
  key_name=$(echo "${instance_details}" | awk '{print $4}')
  vpc_id=$(echo "${instance_details}" | awk '{print $5}')
  public_ip=$(echo "${instance_details}" | awk '{print $6}')
  state=$(echo "${instance_details}" | awk '{print $7}')

  echo "    ID: ${instance_id}"
  echo "    Image ID: ${image_id}"
  echo "    Instance type: ${instance_type}"
  echo "    Key name: ${key_name}"
  echo "    VPC ID: ${vpc_id}"
  echo "    Public IP: ${public_ip}"
  echo "    State: ${state}"

  return 0
}

##############################################################################
# function connect_to_instance
#
# This function displays the public IP address of an Amazon Elastic Compute Cloud
 (Amazon EC2) instance and prompts the user to connect to the instance via SSH.
#
# Parameters:
#       $1 - The name of the key file used to connect to the instance.
#       $2 - The public IP address of the instance.
```

```
#
# Returns:
#       None
###############################################################################
function connect_to_instance() {
  local key_file_name="$1"
  local public_ip="$2"

  # Validate the input parameters
  if [[ -z "$key_file_name" ]]; then
    echo "ERROR: You must provide a key file name as the first argument." >&2
    return 1
  fi

  if [[ -z "$public_ip" ]]; then
    echo "ERROR: You must provide a public IP address as the second argument." >&2
    return 1
  fi

  # Display the public IP address and connection command
  echo "To connect, run the following command:"
  echo "    ssh -i ${key_file_name} ec2-user@${public_ip}"

  # Prompt the user to connect to the instance
  if yes_no_input "Do you want to connect now? (y/n) "; then
    echo "After you have connected, you can return to this example by typing 'exit'"
    ssh -i "${key_file_name}" ec2-user@"${public_ip}"
  fi
}

###############################################################################
# function get_input
#
# This function gets user input from the command line.
#
# Outputs:
#   User input to stdout.
#
# Returns:
#       0
###############################################################################
function get_input() {

  if [ -z "${mock_input+x}" ]; then
```

```
    read -r get_input_result
  else

    if [ "$mock_input_array_index" -lt ${#mock_input_array[@]} ]; then
      get_input_result="${mock_input_array[$mock_input_array_index]}"
      # bashsupport disable=BP2001
      # shellcheck disable=SC2206
      ((mock_input_array_index++))
      echo -n "$get_input_result"
    else
      echo "MOCK_INPUT_ARRAY has no more elements" 1>&2
      return 1
    fi
  fi

  return 0
}

##############################################################################
# function yes_no_input
#
# This function requests a yes/no answer from the user, following to a prompt.
#
# Parameters:
#       $1 - The prompt.
#
# Returns:
#       0 - If yes.
#       1 - If no.
##############################################################################
function yes_no_input() {
  if [ -z "$1" ]; then
    echo "Internal error yes_no_input"
    return 1
  fi

  local index=0
  local response="N"
  while [[ $index -lt 10 ]]; do
    index=$((index + 1))
    echo -n "$1"
    if ! get_input; then
      return 1
    fi
```

```
    response=$(echo "$get_input_result" | tr '[:upper:]' '[:lower:]')
    if [ "$response" = "y" ] || [ "$response" = "n" ]; then
      break
    else
      echo -e "\nPlease enter or 'y' or 'n'."
    fi
  done

  echo

  if [ "$response" = "y" ]; then
    return 0
  else
    return 1
  fi
}

##############################################################################
# function integer_input
#
# This function prompts the user to enter an integer within a specified range
# and validates the input.
#
# Parameters:
#       $1 - The prompt message to display to the user.
#       $2 - The minimum value of the accepted range.
#       $3 - The maximum value of the accepted range.
#
# Returns:
#       The valid integer input from the user.
#       If the input is invalid or out of range, the function will continue
#       prompting the user until a valid input is provided.
##############################################################################
function integer_input() {
  local prompt="$1"
  local min_value="$2"
  local max_value="$3"
  local input=""

  while true; do
    # Display the prompt message and wait for user input
    echo -n "$prompt"

    if ! get_input; then
```

```
      return 1
    fi

    input="$get_input_result"

    # Check if the input is a valid integer
    if [[ "$input" =~ ^-?[0-9]+$ ]]; then
      # Check if the input is within the specified range
      if ((input >= min_value && input <= max_value)); then
        return 0
      else
        echo "Error: Input, $input, must be between $min_value and $max_value."
      fi
    else
      echo "Error: Invalid input- $input. Please enter an integer."
    fi
  done
}
##############################################################################
# function new_line_and_tab_to_list
#
# This function takes a string input containing newlines and tabs, and
# converts it into a list (array) of elements.
#
# Parameters:
#       $1 - The input string containing newlines and tabs.
#
# Returns:
#       The resulting list (array) is stored in the global variable
#       'list_result'.
##############################################################################
function new_line_and_tab_to_list() {
  local input=$1
  export list_result

  list_result=()
  mapfile -t lines <<<"$input"
  local line
  for line in "${lines[@]}"; do
    IFS=$'\t' read -ra parameters <<<"$line"
    list_result+=("${parameters[@]}")
  done
}
```

```
##############################################################################
# function echo_repeat
#
# This function prints a string 'n' times to stdout.
#
# Parameters:
#       $1 - The string.
#       $2 - Number of times to print the string.
#
# Outputs:
#   String 'n' times to stdout.
#
# Returns:
#       0
##############################################################################
function echo_repeat() {
  local end=$2
  for ((i = 0; i < end; i++)); do
    echo -n "$1"
  done
  echo
}
```

As funções do DynamoDB usadas nesse cenário.

```
##############################################################################
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#       -f file_path - File to store the key pair.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_create_keypair() {
  local key_pair_name file_path response
```

```
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_create_keypair"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-bit
RSA key pair"
    echo " and writes it to a file."
    echo "  -n key_pair_name - A key pair name."
    echo "  -f file_path - File to store the key pair."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:f:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      f) file_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
  fi

  response=$(aws ec2 create-key-pair \
```

```
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  }

  if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
  fi

  return 0
}

###############################################################################
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs.
#
# Parameters:
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_key_pairs() {
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_key_pairs"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
```

```
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Parameters:
#       -n security_group_name - The name of the security group.
#       -d security_group_description - The description of the security group.
#
# Returns:
#       The ID of the created security group, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
```

```
function ec2_create_security_group() {
  local security_group_name security_group_description response

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -n security_group_name - The name of the security group."
    echo "  -d security_group_description - The description of the security group."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "n:d:h" option; do
    case "${option}" in
      n) security_group_name="${OPTARG}" ;;
      d) security_group_description="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$security_group_name" ]]; then
    errecho "ERROR: You must provide a security group name with the -n parameter."
    return 1
  fi

  if [[ -z "$security_group_description" ]]; then
    errecho "ERROR: You must provide a security group description with the -d
 parameter."
    return 1
  fi

  # Create the security group
  response=$(aws ec2 create-security-group \
```

```
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups.
#
# Parameters:
#       -g security_group_id - The ID of the security group to describe (optional).
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_security_groups() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_security_groups"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) security
 groups."
    echo "  -g security_group_id - The ID of the security group to describe
 (optional)."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:h" option; do
    case "${option}" in
```

```
      g) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"

  if [[ -n "$security_group_id" ]]; then
    response=$(aws ec2 describe-security-groups --group-ids "$security_group_id" --
query "${query}" --output text)
  else
    response=$(aws ec2 describe-security-groups --query "${query}" --output text)
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports describe-security-groups operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
 (Amazon EC2) security group.
#
# Parameters:
```

```
#       -g security_group_id - The ID of the security group.
#       -i ip_address - The IP address or CIDR block to authorize.
#       -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#       -f from_port - The start of the port range to authorize.
#       -t to_port - The end of the port range to authorize.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_authorize_security_group_ingress() {
  local security_group_id ip_address protocol from_port to_port response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
 EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      i) ip_address="${OPTARG}" ;;
      p) protocol="${OPTARG}" ;;
      f) from_port="${OPTARG}" ;;
      t) to_port="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
  fi

  if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
 parameter."
    usage
    return 1
  fi

  if [[ -z "$protocol" ]]; then
    errecho "ERROR: You must provide a protocol with the -p parameter."
    usage
    return 1
  fi

  if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
  fi

  response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation failed.
$response"
```

```
    return 1
  }

  return 0
}

##############################################################################
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images.
#
# Parameters:
#       -i image_ids - A space-separated  list of image IDs (optional).
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_describe_images() {
  local image_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_images"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) image_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
```

```
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
  fi

  response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g., x86_64)
# -t, --type INSTANCE_TYPE         Comma-separated list of instance types (e.g.,
 t2.micro)
# -h, --help                       Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
###############################################################################
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""
```

```
  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-t|--
type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture
 (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE         Comma-separated list of instance types
 (e.g., t2.micro)"
    echo "  -h, --help                       Show this help message"
  }

  while [[ $# -gt 0 ]]; do
    case "$1" in
      -a | --architecture)
        architecture="$2"
        shift 2
        ;;
      -t | --type)
        instance_types="$2"
        shift 2
        ;;
      -h | --help)
        usage
        return 0
        ;;
      *)
        echo "Unknown argument: $1"
        return 1
        ;;
    esac
  done

  if [[ -z "$architecture" ]]; then
    errecho "Error: Architecture not specified."
    usage
    return 1
  fi

  if [[ -z "$instance_types" ]]; then
    errecho "Error: Instance type not specified."
    usage
    return 1
  fi
```

```
  local tmp_json_file="temp_ec2.json"
  echo -n '[
    {
      "Name": "processor-info.supported-architecture",
      "Values": [' >"$tmp_json_file"

  local items
  IFS=',' read -ra items <<<"$architecture"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done
  echo -n ']},
    {
    "Name": "instance-type",
      "Values": [' >>"$tmp_json_file"
  IFS=',' read -ra items <<<"$instance_types"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done

  echo -n ']}]' >>"$tmp_json_file"

  local response
  response=$(aws ec2 describe-instance-types --filters file://"$tmp_json_file" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

  local error_code=$?

  rm "$tmp_json_file"

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
```

```
    return 1
  fi

  echo "$response"
  return 0
}

###############################################################################
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#       -t instance_type - The instance type to use (e.g., t2.micro).
#       -k key_pair_name - The name of the key pair to use.
#       -s security_group_id - The ID of the security group to use.
#       -c count - The number of instances to launch (default: 1).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_run_instances() {
  local image_id instance_type key_pair_name security_group_id count response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_run_instances"
    echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
    echo "  -k key_pair_name - The name of the key pair to use."
    echo "  -s security_group_id - The ID of the security group to use."
    echo "  -c count - The number of instances to launch (default: 1)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:t:k:s:c:h" option; do
```

```
    case "${option}" in
      i) image_id="${OPTARG}" ;;
      t) instance_type="${OPTARG}" ;;
      k) key_pair_name="${OPTARG}" ;;
      s) security_group_id="${OPTARG}" ;;
      c) count="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$image_id" ]]; then
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
parameter."
    usage
    return 1
  fi

  if [[ -z "$instance_type" ]]; then
    errecho "ERROR: You must provide an instance type with the -t parameter."
    usage
    return 1
  fi

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -k parameter."
    usage
    return 1
  fi

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -s parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$count" ]]; then
    count=1
  fi

  response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_describe_instances
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID of the instance to describe (optional).
#       -q query - The query to filter the response (optional).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_instances() {
  local instance_id query response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_instances"
```

```
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
    echo "  -q query - The query to filter the response (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:q:h" option; do
    case "${option}" in
      i) instance_id="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
  fi

  local query_arg=""
  if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
  else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
```

```
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_stop_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_stop_instances"
    echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
    return 1
  }

  return 0
}

###############################################################################
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function ec2_start_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 start-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports start-instances operation failed with $response."
    return 1
  }
```

```
  return 0
}

##############################################################################
# function ec2_allocate_address
#
# This function allocates an Elastic IP address for use with Amazon Elastic Compute
 Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#       -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard').
#
# Returns:
#       The allocated Elastic IP address, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_allocate_address() {
  local domain response

  # Function to display usage information
  function usage() {
    echo "function ec2_allocate_address"
    echo "Allocates an Elastic IP address for use with Amazon Elastic Compute Cloud
 (Amazon EC2) instances in a specific AWS Region."
    echo "  -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard')."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "d:h" option; do
    case "${option}" in
      d) domain="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$domain" ]]; then
    errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc' or
 'standard')."
    return 1
  fi

  if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
    errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
    return 1
  fi

  # Allocate the Elastic IP address
  response=$(aws ec2 allocate-address \
    --domain "$domain" \
    --query "[PublicIp,AllocationId]" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

################################################################################
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to associate.
```

```
#       -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_associate_address() {
  local allocation_id instance_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 associate."
    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:i:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      i) instance_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
```

```
    return 1
  fi

  if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
    return 1
  fi

  # Associate the Elastic IP address
  response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a association_id - The association ID that represents the association of
 the Elastic IP address with an instance.
#
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
```

```
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the association
 of the Elastic IP address with an instance."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) association_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_release_address
```

```
#
# This function releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to release.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_release_address() {
  local allocation_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_release_address"
    echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 release."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
```

```
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 release-address \
    --allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
#       -i instance_ids - A space-separated list of instance IDs.
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_terminate_instances() {
  local instance_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_terminate_instances"
    echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_ids - A space-separated list of instance IDs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
```

```
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Check if instance ID is provided
  if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Parameters:
#       -i security_group_id - The ID of the security group to delete.
#
```

```
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_delete_security_group() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-security-group --group-id "$security_group_id" --output
 text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
```

```
  }

  return 0
}

###############################################################################
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_keypair() {
  local key_pair_name response

  local option OPTARG # Required to use getopts command in a function.
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-key-pair \
    --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas nesse cenário.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
```

```
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress

- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function ec2_allocate_address
#
# This function allocates an Elastic IP address for use with Amazon Elastic Compute
 Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#       -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard').
#
# Returns:
#       The allocated Elastic IP address, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###########################################################################
function ec2_allocate_address() {
  local domain response
```

```
# Function to display usage information
function usage() {
  echo "function ec2_allocate_address"
  echo "Allocates an Elastic IP address for use with Amazon Elastic Compute Cloud
(Amazon EC2) instances in a specific AWS Region."
  echo "  -d domain - The domain for the Elastic IP address (either 'vpc' or
'standard')."
  echo ""
}

# Parse the command-line arguments
while getopts "d:h" option; do
  case "${option}" in
    d) domain="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$domain" ]]; then
  errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc' or
'standard')."
  return 1
fi

if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
  errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
  return 1
fi

# Allocate the Elastic IP address
response=$(aws ec2 allocate-address \
  --domain "$domain" \
  --query "[PublicIp,AllocationId]" \
```

```
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
```

```
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte AllocateAddress em Referência de AWS CLI comandos.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to associate.
#       -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
#
##############################################################################
function ec2_associate_address() {
  local allocation_id instance_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 associate."
    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:i:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      i) instance_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
```

```
    return 1
  fi

  # Associate the Elastic IP address
  response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
```

```
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte AssociateAddress em Referência de AWS CLI comandos.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_authorize_security_group_ingress
#
```

```
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
 (Amazon EC2) security group.
#
# Parameters:
#       -g security_group_id - The ID of the security group.
#       -i ip_address - The IP address or CIDR block to authorize.
#       -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#       -f from_port - The start of the port range to authorize.
#       -t to_port - The end of the port range to authorize.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_authorize_security_group_ingress() {
  local security_group_id ip_address protocol from_port to_port response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
 EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      i) ip_address="${OPTARG}" ;;
      p) protocol="${OPTARG}" ;;
      f) from_port="${OPTARG}" ;;
      t) to_port="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
  fi

  if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
parameter."
    usage
    return 1
  fi

  if [[ -z "$protocol" ]]; then
    errecho "ERROR: You must provide a protocol with the -p parameter."
    usage
    return 1
  fi

  if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
  fi

  response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
```

```
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation failed.
$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
```

```
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngress em Referência de AWS CLI comandos.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#       -f file_path - File to store the key pair.
#
# And:
```

```
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_create_keypair() {
  local key_pair_name file_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_create_keypair"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-bit
 RSA key pair"
    echo " and writes it to a file."
    echo "  -n key_pair_name - A key pair name."
    echo "  -f file_path - File to store the key pair."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:f:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      f) file_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
```

```
    usage
    return 1
  fi

  response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  }

  if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
  fi

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
```

```
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte CreateKeyPair em Referência de AWS CLI comandos.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
```

```
#
# Parameters:
#       -n security_group_name - The name of the security group.
#       -d security_group_description - The description of the security group.
#
# Returns:
#       The ID of the created security group, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_create_security_group() {
  local security_group_name security_group_description response

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -n security_group_name - The name of the security group."
    echo "  -d security_group_description - The description of the security group."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "n:d:h" option; do
    case "${option}" in
      n) security_group_name="${OPTARG}" ;;
      d) security_group_description="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
```

```
  if [[ -z "$security_group_name" ]]; then
    errecho "ERROR: You must provide a security group name with the -n parameter."
    return 1
  fi

  if [[ -z "$security_group_description" ]]; then
    errecho "ERROR: You must provide a security group description with the -d
 parameter."
    return 1
  fi

  # Create the security group
  response=$(aws ec2 create-security-group \
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
```

```
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte CreateSecurityGroup em Referência de AWS CLI comandos.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_keypair() {
  local key_pair_name response

  local option OPTARG # Required to use getopts command in a function.
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-key-pair \
    --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
```

```
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteKeyPair em Referência de AWS CLI comandos.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
##############################################################################
```

```
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Parameters:
#       -i security_group_id - The ID of the security group to delete.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_security_group() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
```

```
  fi

  response=$(aws ec2 delete-security-group --group-id "$security_group_id" --output
 text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
```

```
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteSecurityGroup em Referência de AWS CLI comandos.

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images.
#
# Parameters:
#       -i image_ids - A space-separated  list of image IDs (optional).
#       -h - Display help.
#
```

```
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_describe_images() {
  local image_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_images"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) image_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
  fi

  response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
```

```
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
```

```
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeImages em Referência de AWS CLI comandos.

`DescribeInstanceTypes`

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g., x86_64)
# -t, --type INSTANCE_TYPE         Comma-separated list of instance types (e.g.,
 t2.micro)
# -h, --help                       Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
```

```
##############################################################################
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""

  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-t|--
type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture
 (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE         Comma-separated list of instance types
 (e.g., t2.micro)"
    echo "  -h, --help                       Show this help message"
  }

  while [[ $# -gt 0 ]]; do
    case "$1" in
      -a | --architecture)
        architecture="$2"
        shift 2
        ;;
      -t | --type)
        instance_types="$2"
        shift 2
        ;;
      -h | --help)
        usage
        return 0
        ;;
      *)
        echo "Unknown argument: $1"
        return 1
        ;;
    esac
  done

  if [[ -z "$architecture" ]]; then
    errecho "Error: Architecture not specified."
    usage
    return 1
  fi

  if [[ -z "$instance_types" ]]; then
```

```
    errecho "Error: Instance type not specified."
    usage
    return 1
  fi

  local tmp_json_file="temp_ec2.json"
  echo -n '[
    {
      "Name": "processor-info.supported-architecture",
      "Values": [' >"$tmp_json_file"

  local items
  IFS=',' read -ra items <<<"$architecture"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done
  echo -n ']},
    {
    "Name": "instance-type",
      "Values": [' >>"$tmp_json_file"
  IFS=',' read -ra items <<<"$instance_types"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done

  echo -n ']}]' >>"$tmp_json_file"

  local response
  response=$(aws ec2 describe-instance-types --filters file://"$tmp_json_file" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

  local error_code=$?

  rm "$tmp_json_file"
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
    return 1
  fi

  echo "$response"
  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
```

```
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeInstanceTypes em Referência de AWS CLI comandos.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function ec2_describe_instances
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID of the instance to describe (optional).
#       -q query - The query to filter the response (optional).
#       -h - Display help.
#
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_describe_instances() {
  local instance_id query response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_instances"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
    echo "  -q query - The query to filter the response (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:q:h" option; do
    case "${option}" in
      i) instance_id="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
  fi
```

```
  local query_arg=""
  if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
  else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
```

```
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeInstances em Referência de AWS CLI comandos.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs.
#
# Parameters:
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_key_pairs() {
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_key_pairs"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response
```

```
  response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
```

```
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeKeyPairs em Referência de AWS CLI comandos.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups.
#
# Parameters:
#       -g security_group_id - The ID of the security group to describe (optional).
#
```

```
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_describe_security_groups() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_security_groups"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) security
 groups."
    echo "  -g security_group_id - The ID of the security group to describe
 (optional)."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"

  if [[ -n "$security_group_id" ]]; then
    response=$(aws ec2 describe-security-groups --group-ids "$security_group_id" --
query "${query}" --output text)
  else
    response=$(aws ec2 describe-security-groups --query "${query}" --output text)
  fi
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports describe-security-groups operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
```

```
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência de AWS CLI comandos.

`DisassociateAddress`

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
```

```
#       -a association_id - The association ID that represents the association of
 the Elastic IP address with an instance.
#
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the association
 of the Elastic IP address with an instance."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) association_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
  fi
```

```
  response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
```

```
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DisassociateAddress em Referência de AWS CLI comandos.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function ec2_release_address
#
# This function releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to release.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
#
##############################################################################
function ec2_release_address() {
  local allocation_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_release_address"
    echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 release."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 release-address \
    --allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
```

```
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
```

```
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte ReleaseAddress em Referência de AWS CLI comandos.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#       -t instance_type - The instance type to use (e.g., t2.micro).
#       -k key_pair_name - The name of the key pair to use.
#       -s security_group_id - The ID of the security group to use.
#       -c count - The number of instances to launch (default: 1).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
```

```
function ec2_run_instances() {
  local image_id instance_type key_pair_name security_group_id count response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_run_instances"
    echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
    echo "  -k key_pair_name - The name of the key pair to use."
    echo "  -s security_group_id - The ID of the security group to use."
    echo "  -c count - The number of instances to launch (default: 1)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:t:k:s:c:h" option; do
    case "${option}" in
      i) image_id="${OPTARG}" ;;
      t) instance_type="${OPTARG}" ;;
      k) key_pair_name="${OPTARG}" ;;
      s) security_group_id="${OPTARG}" ;;
      c) count="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$image_id" ]]; then
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
 parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$instance_type" ]]; then
    errecho "ERROR: You must provide an instance type with the -t parameter."
    usage
    return 1
  fi

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -k parameter."
    usage
    return 1
  fi

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -s parameter."
    usage
    return 1
  fi

  if [[ -z "$count" ]]; then
    count=1
  fi

  response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte RunInstances em Referência de AWS CLI comandos.

StartInstances

O código de exemplo a seguir mostra como usar `StartInstances`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function ec2_start_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }
```

```
  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 start-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports start-instances operation failed with $response."
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte StartInstances em Referência de AWS CLI comandos.

`StopInstances`

O código de exemplo a seguir mostra como usar `StopInstances`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_stop_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_stop_instances"
    echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
```

```
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
```

```
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte StopInstances em Referência de AWS CLI comandos.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
#       -i instance_ids - A space-separated list of instance IDs.
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_terminate_instances() {
  local instance_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_terminate_instances"
    echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_ids - A space-separated list of instance IDs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
```

```
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Check if instance ID is provided
  if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
  }

  return 0
}
```

As funções utilitárias usadas neste exemplo.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Para obter detalhes da API, consulte TerminateInstances em Referência de AWS CLI comandos.

Cenários

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
```

```
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
```

```
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
```

```
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true
```

```
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
```

```
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
```

```
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
```

```
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
```

```
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
```

```
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
```

```
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
```

```
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
```

```
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"
```

```
    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
```

```
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity

- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Criar uma rede privada virtual básica

O exemplo de código a seguir mostra como:

- Criar uma VPC.
- Criar sub-redes.
- Configurar a conectividade com a internet.
- Criar um gateway NAT.
- Definir as configurações da sub-rede.
- Criar grupos de segurança.
- Verificar a configuração de sua VPC.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC Creation Script
# This script creates a VPC with public and private subnets, internet gateway, NAT
 gateway, and security groups
```

```
# Set up logging
LOG_FILE="vpc_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to handle errors
handle_error() {
  echo "ERROR: $1"
  echo "Resources created before error:"
  for resource in "${CREATED_RESOURCES[@]}"
  do
    echo "- $resource"
  done

  echo "Attempting to clean up resources..."
  cleanup_resources
  exit 1
}

# Function to clean up resources
cleanup_resources() {
  echo "Cleaning up resources in reverse order..."

  # Reverse the array to delete in reverse order of creation
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--))
  do
    resource="${CREATED_RESOURCES[$i]}"
    resource_type=$(echo "$resource" | cut -d':' -f1)
    resource_id=$(echo "$resource" | cut -d':' -f2)

    case "$resource_type" in
      "INSTANCE")
        echo "Terminating EC2 instance: $resource_id"
        aws ec2 terminate-instances --instance-ids "$resource_id" || echo "Failed to
 terminate instance: $resource_id"
        # Wait for instance to terminate
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$resource_id" || echo
 "Failed to wait for instance termination: $resource_id"
        ;;
      "KEY_PAIR")
        echo "Deleting key pair: $resource_id"
        aws ec2 delete-key-pair --key-name "$resource_id" || echo "Failed to delete
 key pair: $resource_id"
```

```
        # Remove the .pem file if it exists
        if [ -f "${resource_id}.pem" ]; then
          rm -f "${resource_id}.pem"
        fi
        ;;
      "NAT_GATEWAY")
        echo "Deleting NAT Gateway: $resource_id"
        aws ec2 delete-nat-gateway --nat-gateway-id "$resource_id" || echo "Failed
 to delete NAT Gateway: $resource_id"
        # NAT Gateway deletion takes time, wait for it to complete
        echo "Waiting for NAT Gateway to be deleted..."
        aws ec2 wait nat-gateway-deleted --nat-gateway-ids "$resource_id" || echo
 "Failed to wait for NAT Gateway deletion: $resource_id"
        ;;
      "EIP")
        echo "Releasing Elastic IP: $resource_id"
        aws ec2 release-address --allocation-id "$resource_id" || echo "Failed to
 release Elastic IP: $resource_id"
        ;;
      "ROUTE_TABLE_ASSOCIATION")
        echo "Disassociating Route Table: $resource_id"
        aws ec2 disassociate-route-table --association-id "$resource_id" || echo
 "Failed to disassociate Route Table: $resource_id"
        ;;
      "ROUTE_TABLE")
        echo "Deleting Route Table: $resource_id"
        aws ec2 delete-route-table --route-table-id "$resource_id" || echo "Failed
 to delete Route Table: $resource_id"
        ;;
      "INTERNET_GATEWAY")
        echo "Detaching Internet Gateway: $resource_id from VPC: $VPC_ID"
        aws ec2 detach-internet-gateway --internet-gateway-id "$resource_id" --vpc-
id "$VPC_ID" || echo "Failed to detach Internet Gateway: $resource_id"
        echo "Deleting Internet Gateway: $resource_id"
        aws ec2 delete-internet-gateway --internet-gateway-id "$resource_id" || echo
 "Failed to delete Internet Gateway: $resource_id"
        ;;
      "SECURITY_GROUP")
        echo "Deleting Security Group: $resource_id"
        aws ec2 delete-security-group --group-id "$resource_id" || echo "Failed to
 delete Security Group: $resource_id"
        ;;
      "SUBNET")
        echo "Deleting Subnet: $resource_id"
```

```
        aws ec2 delete-subnet --subnet-id "$resource_id" || echo "Failed to delete
 Subnet: $resource_id"
        ;;
      "VPC")
        echo "Deleting VPC: $resource_id"
        aws ec2 delete-vpc --vpc-id "$resource_id" || echo "Failed to delete VPC:
 $resource_id"
        ;;
    esac
  done
}

# Initialize array to track created resources
CREATED_RESOURCES=()

echo "Starting VPC creation script at $(date)"

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
aws configure list || handle_error "AWS CLI is not properly configured"

# Verify identity and permissions
echo "Verifying identity and permissions..."
if ! aws sts get-caller-identity; then
  echo "ERROR: Unable to verify AWS identity. This could be due to:"
  echo "  - Expired credentials"
  echo "  - Missing or invalid AWS credentials"
  echo "  - Insufficient permissions"
  echo ""
  echo "Please run 'aws configure' to update your credentials or check your IAM
 permissions."
  exit 1
fi

# Create VPC
echo "Creating VPC with CIDR block 10.0.0.0/16..."
VPC_ID=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 'ResourceType=vpc,Tags=[{Key=Name,Value=MyVPC},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'Vpc.VpcId' --output text)

if [ -z "$VPC_ID" ]; then
  handle_error "Failed to create VPC"
fi
```

```
CREATED_RESOURCES+=("VPC:$VPC_ID")
echo "VPC created with ID: $VPC_ID"

# Enable DNS support and hostnames
echo "Enabling DNS support and hostnames for VPC..."
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-support || handle_error
 "Failed to enable DNS support"
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-hostnames ||
 handle_error "Failed to enable DNS hostnames"

# Get available Availability Zones
echo "Getting available Availability Zones..."
AZ1=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[0].ZoneName' --
output text)
AZ2=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[1].ZoneName' --
output text)

if [ -z "$AZ1" ] || [ -z "$AZ2" ]; then
  handle_error "Failed to get Availability Zones"
fi

echo "Using Availability Zones: $AZ1 and $AZ2"

# Create public subnets
echo "Creating public subnet in $AZ1..."
PUBLIC_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.0.0/24 \
  --availability-zone "$AZ1" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ1},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ1" ]; then
  handle_error "Failed to create public subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ1")
echo "Public subnet created in $AZ1 with ID: $PUBLIC_SUBNET_AZ1"

echo "Creating public subnet in $AZ2..."
PUBLIC_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
```

```
  --cidr-block 10.0.1.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ2},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ2" ]; then
  handle_error "Failed to create public subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ2")
echo "Public subnet created in $AZ2 with ID: $PUBLIC_SUBNET_AZ2"

# Create private subnets
echo "Creating private subnet in $AZ1..."
PRIVATE_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.2.0/24 \
  --availability-zone "$AZ1" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ1},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PRIVATE_SUBNET_AZ1" ]; then
  handle_error "Failed to create private subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ1")
echo "Private subnet created in $AZ1 with ID: $PRIVATE_SUBNET_AZ1"

echo "Creating private subnet in $AZ2..."
PRIVATE_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.3.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ2},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PRIVATE_SUBNET_AZ2" ]; then
  handle_error "Failed to create private subnet in AZ2"
```

```
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ2")
echo "Private subnet created in $AZ2 with ID: $PRIVATE_SUBNET_AZ2"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_ID=$(aws ec2 create-internet-gateway \
  --tag-specifications 'ResourceType=internet-gateway,Tags=[{Key=Name,Value=MyIGW},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'InternetGateway.InternetGatewayId' \
  --output text)

if [ -z "$IGW_ID" ]; then
  handle_error "Failed to create Internet Gateway"
fi

CREATED_RESOURCES+=("INTERNET_GATEWAY:$IGW_ID")
echo "Internet Gateway created with ID: $IGW_ID"

# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id "$VPC_ID"
 || handle_error "Failed to attach Internet Gateway to VPC"

# Create public route table
echo "Creating public route table..."
PUBLIC_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Public-RT},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PUBLIC_RT" ]; then
  handle_error "Failed to create public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PUBLIC_RT")
echo "Public route table created with ID: $PUBLIC_RT"

# Add route to Internet Gateway
echo "Adding route to Internet Gateway in public route table..."
```

```
aws ec2 create-route --route-table-id "$PUBLIC_RT" --destination-cidr-block
 0.0.0.0/0 --gateway-id "$IGW_ID" || handle_error "Failed to add route to Internet
 Gateway"

# Associate public subnets with public route table
echo "Associating public subnet in $AZ1 with public route table..."
PUBLIC_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT" --
subnet-id "$PUBLIC_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_1" ]; then
  handle_error "Failed to associate public subnet in AZ1 with public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_1")

echo "Associating public subnet in $AZ2 with public route table..."
PUBLIC_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT" --
subnet-id "$PUBLIC_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_2" ]; then
  handle_error "Failed to associate public subnet in AZ2 with public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_2")

# Create private route table
echo "Creating private route table..."
PRIVATE_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Private-RT},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PRIVATE_RT" ]; then
  handle_error "Failed to create private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PRIVATE_RT")
echo "Private route table created with ID: $PRIVATE_RT"

# Associate private subnets with private route table
echo "Associating private subnet in $AZ1 with private route table..."
```

```
PRIVATE_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT" --
subnet-id "$PRIVATE_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_1" ]; then
  handle_error "Failed to associate private subnet in AZ1 with private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_1")

echo "Associating private subnet in $AZ2 with private route table..."
PRIVATE_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT" --
subnet-id "$PRIVATE_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_2" ]; then
  handle_error "Failed to associate private subnet in AZ2 with private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_2")

# Allocate Elastic IP for NAT Gateway
echo "Allocating Elastic IP for NAT Gateway..."
EIP_ALLOC=$(aws ec2 allocate-address --domain vpc --tag-specifications
 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'AllocationId' --output text)

if [ -z "$EIP_ALLOC" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

CREATED_RESOURCES+=("EIP:$EIP_ALLOC")
echo "Elastic IP allocated with ID: $EIP_ALLOC"

# Create NAT Gateway
echo "Creating NAT Gateway in public subnet in $AZ1..."
NAT_GW=$(aws ec2 create-nat-gateway \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --allocation-id "$EIP_ALLOC" \
  --tag-specifications 'ResourceType=natgateway,Tags=[{Key=Name,Value=MyNATGateway},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'NatGateway.NatGatewayId' \
  --output text)

if [ -z "$NAT_GW" ]; then
  handle_error "Failed to create NAT Gateway"
```

```
fi

CREATED_RESOURCES+=("NAT_GATEWAY:$NAT_GW")
echo "NAT Gateway created with ID: $NAT_GW"

# Wait for NAT Gateway to be available
echo "Waiting for NAT Gateway to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW" || handle_error "NAT
 Gateway did not become available"

# Add route to NAT Gateway in private route table
echo "Adding route to NAT Gateway in private route table..."
aws ec2 create-route --route-table-id "$PRIVATE_RT" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW" || handle_error "Failed to add route to NAT
 Gateway"

# Enable auto-assign public IP for instances in public subnets
echo "Enabling auto-assign public IP for instances in public subnet in $AZ1..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ1" --map-public-ip-on-
launch || handle_error "Failed to enable auto-assign public IP for public subnet in
 AZ1"

echo "Enabling auto-assign public IP for instances in public subnet in $AZ2..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ2" --map-public-ip-on-
launch || handle_error "Failed to enable auto-assign public IP for public subnet in
 AZ2"

# Create security group for web servers
echo "Creating security group for web servers..."
WEB_SG=$(aws ec2 create-security-group \
  --group-name "WebServerSG-$(date +%s)" \
  --description "Security group for web servers" \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'GroupId' \
  --output text)

if [ -z "$WEB_SG" ]; then
  handle_error "Failed to create security group for web servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$WEB_SG")
echo "Security group for web servers created with ID: $WEB_SG"
```

```
# Allow HTTP and HTTPS traffic
echo "Allowing HTTP traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --port
 80 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTP traffic"

echo "Allowing HTTPS traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --port
 443 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTPS traffic"

# Note: In a production environment, you should restrict the source IP ranges for
 security
echo "NOTE: In a production environment, you should restrict the source IP ranges
 for HTTP and HTTPS traffic"

# Create security group for database servers
echo "Creating security group for database servers..."
DB_SG=$(aws ec2 create-security-group \
  --group-name "DBServerSG-$(date +%s)" \
  --description "Security group for database servers" \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'GroupId' \
  --output text)

if [ -z "$DB_SG" ]; then
  handle_error "Failed to create security group for database servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$DB_SG")
echo "Security group for database servers created with ID: $DB_SG"

# Allow MySQL/Aurora traffic from web servers only
echo "Allowing MySQL/Aurora traffic from web servers to database servers..."
aws ec2 authorize-security-group-ingress --group-id "$DB_SG" --protocol tcp --
port 3306 --source-group "$WEB_SG" || handle_error "Failed to allow MySQL/Aurora
 traffic"

# Verify VPC configuration
echo "Verifying VPC configuration..."
echo "VPC:"
aws ec2 describe-vpcs --vpc-id "$VPC_ID" || handle_error "Failed to describe VPC"
```

```
echo "Subnets:"
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" || handle_error
 "Failed to describe subnets"

echo "Route tables:"
aws ec2 describe-route-tables --filters "Name=vpc-id,Values=$VPC_ID" || handle_error
 "Failed to describe route tables"

echo "Internet gateway:"
aws ec2 describe-internet-gateways --filters "Name=attachment.vpc-id,Values=$VPC_ID"
 || handle_error "Failed to describe Internet Gateway"

echo "NAT gateway:"
aws ec2 describe-nat-gateways --filter "Name=vpc-id,Values=$VPC_ID" || handle_error
 "Failed to describe NAT Gateway"

echo "Security groups:"
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$VPC_ID" ||
 handle_error "Failed to describe security groups"

echo ""
# Summary of created resources
echo "VPC creation completed successfully!"
echo "Summary of created resources:"
echo "- VPC: $VPC_ID"
echo "- Public Subnet in $AZ1: $PUBLIC_SUBNET_AZ1"
echo "- Public Subnet in $AZ2: $PUBLIC_SUBNET_AZ2"
echo "- Private Subnet in $AZ1: $PRIVATE_SUBNET_AZ1"
echo "- Private Subnet in $AZ2: $PRIVATE_SUBNET_AZ2"
echo "- Internet Gateway: $IGW_ID"
echo "- Public Route Table: $PUBLIC_RT"
echo "- Private Route Table: $PRIVATE_RT"
echo "- Elastic IP: $EIP_ALLOC"
echo "- NAT Gateway: $NAT_GW"
echo "- Web Servers Security Group: $WEB_SG"
echo "- Database Servers Security Group: $DB_SG"

# Deploy EC2 instances
echo ""
echo "Deploying EC2 instances..."

# Create key pair for SSH access
KEY_NAME="vpc-tutorial-key-$(date +%s)"
echo "Creating key pair $KEY_NAME..."
```

```
aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-specifications
 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-
gs}]' --query 'KeyMaterial' --output text > "${KEY_NAME}.pem" || handle_error
 "Failed to create key pair"
chmod 400 "${KEY_NAME}.pem"
echo "Key pair saved to ${KEY_NAME}.pem"
CREATED_RESOURCES+=("KEY_PAIR:$KEY_NAME")

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" --output text) ||
 handle_error "Failed to get AMI"
echo "Using AMI: $AMI_ID"

# Launch web server in public subnet
echo "Launching web server in public subnet..."
WEB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$WEB_SG" \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --associate-public-ip-address \
  --user-data '#!/bin/bash
    yum update -y
    yum install -y httpd
    systemctl start httpd
    systemctl enable httpd
    echo "<h1>Hello from $(hostname -f) in the public subnet</h1>" > /var/www/html/
index.html' \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=WebServer},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Instances[0].InstanceId' \
  --output text) || handle_error "Failed to launch web server"
echo "Web server instance created with ID: $WEB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$WEB_INSTANCE")

# Wait for web server to be running
echo "Waiting for web server to be running..."
aws ec2 wait instance-running --instance-ids "$WEB_INSTANCE"
```

```
# Get web server public IP
WEB_PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$WEB_INSTANCE" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' --output text)
echo "Web server public IP: $WEB_PUBLIC_IP"
echo "You can access the web server at: http://$WEB_PUBLIC_IP"

# Launch database server in private subnet
echo "Launching database server in private subnet..."
DB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$DB_SG" \
  --subnet-id "$PRIVATE_SUBNET_AZ1" \
  --user-data '#!/bin/bash
    yum update -y
    yum install -y mariadb-server
    systemctl start mariadb
    systemctl enable mariadb' \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=DBServer},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Instances[0].InstanceId' \
  --output text) || handle_error "Failed to launch database server"
echo "Database server instance created with ID: $DB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$DB_INSTANCE")

# Wait for database server to be running
echo "Waiting for database server to be running..."
aws ec2 wait instance-running --instance-ids "$DB_INSTANCE"

# Get database server private IP
DB_PRIVATE_IP=$(aws ec2 describe-instances --instance-ids "$DB_INSTANCE" \
  --query 'Reservations[0].Instances[0].PrivateIpAddress' --output text)
echo "Database server private IP: $DB_PRIVATE_IP"

echo "EC2 instances deployed successfully!"
echo "- Web Server (Public): $WEB_INSTANCE ($WEB_PUBLIC_IP)"
echo "- Database Server (Private): $DB_INSTANCE ($DB_PRIVATE_IP)"
echo ""
echo "Note: To connect to the web server: ssh -i ${KEY_NAME}.pem ec2-user@
$WEB_PUBLIC_IP"
```

```
echo "To connect to the database server, you must first connect to the web server,
 then use it as a bastion host."
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  echo "Cleaning up resources..."
  cleanup_resources
  echo "All resources have been cleaned up."
else
  echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateInternetGateway
  - CreateKeyPair
  - CreateNatGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - DeleteInternetGateway
  - DeleteKeyPair
  - DeleteNatGateway
  - DeleteRouteTable
  - DeleteSecurityGroup

- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeImages
- DescribeInstances
- DescribeInternetGateways
- DescribeNatGateways
- DescribeRouteTables
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachInternetGateway
- DisassociateRouteTable
- ModifySubnetAttribute
- ModifyVpcAttribute
- ReleaseAddress
- RunInstances
- TerminateInstances

Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um cluster.
- Criar uma definição de tarefa.
- Criar o serviço.
- Fazer a limpeza.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
```

```
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|AccessDenied
\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}
```

```
# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security group
 $security_group_id"
            echo "Waiting 10 seconds before next check..."
```

```
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
```

```
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]]; then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
 anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
```

```
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]]; then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete security
 group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix "$TASK_FAMILY"
 --query "taskDefinitionArns" --output text 2>/dev/null || echo "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
```

```
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME --
service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo "6. Deregister task definitions: aws ecs list-task-definitions --
family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --task-
definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
```

```
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --assume-
role-policy-document file://trust-policy.json" "Create ECS task execution role"
```

```
    aws iam tag-role --role-name ecsTaskExecutionRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
 "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-ecs-
fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
```

```
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-json
 file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
```

```
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --group-
name '$SECURITY_GROUP_NAME' --description 'Security group for ECS Fargate tutorial
 - HTTP access' --vpc-id '$VPC_ID' --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-fargate-
linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"' |
 head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi
```

```
# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP inbound
 rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
```

```
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-type
 FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets\":[\"$(echo
 "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups\":[\"$SECURITY_GROUP_ID
\"],\"assignPublicIp\":\"ENABLED\"}}' --tags key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN" --
query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --output
 text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
```

```
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID" --
query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
```

```
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetRole
  - ListTaskDefinitions
  - ListTasks
  - RegisterTaskDefinition
  - UpdateService

- Wait

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
```

```
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
```

```
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
"$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
"$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
"Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi
```

```
    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
```

```
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi
```

```
    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}
```

```
# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
```

```
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}
```

```
# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
```

```
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
```

```
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
```

```
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"
```

```
    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
```

```
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
```

```
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
```

```
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetParameters
  - GetRole
  - ListContainerInstances
  - ListServices
  - ListTasks
  - RegisterTaskDefinition
  - RunInstances

- StopTask
- TerminateInstances
- UpdateService
- Wait

Criação de uma instância de banco de dados relacional

O exemplo de código a seguir mostra como:

- Configurar componentes de rede.
- Criar um grupo de sub-redes de banco de dados
- Criar uma instância de banco de dados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/latest/
UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
```

```
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ]; then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
```

```
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found ${#SUBNET_IDS[@]}
 subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"
```

```
SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
```

```
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id "$SECRET_NAME"
 --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -d'"'
 -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -d'"'
 -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
```

```
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""
```

```
# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup

- DeleteDbInstance
- DeleteDbSubnetGroup
- DeleteSecret
- DeleteSecurityGroup
- DescribeDbInstances
- DescribeSubnets
- DescribeVpcs
- GetSecretValue
- Wait

Criação e gerenciamento de uma malha de rede de serviços

O exemplo de código a seguir mostra como:

- Criar uma rede de serviços.
- Criar um serviço.
- Listar as VPCs disponíveis.
- Listar grupos de segurança para a VPC selecionada.
- Listar associações de serviço.
- Listar associações de VPC.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC Lattice Service Network Tutorial Script
# This script demonstrates how to create and manage a VPC Lattice service network
```

```
set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="vpc-lattice-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting VPC Lattice tutorial script at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_command() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    if [ "$1" -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        exit "$1"
    fi
}

# Function to validate AWS CLI is available
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" | tee -a "$LOG_FILE"
        exit 1
    fi
}

# Function to validate input parameters
validate_input() {
    local input="$1"
    local param_name="$2"

    if [[ -z "$input" ]]; then
        echo "ERROR: $param_name is empty" | tee -a "$LOG_FILE"
        return 1
    fi

    # Validate against common injection patterns
```

```
    if [[ "$input" =~ [\;\$\`\|\&\<\>\(\)\{\}] ]]; then
        echo "ERROR: $param_name contains invalid characters" | tee -a "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to wait for a resource to be in the desired state
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
    local desired_status="$3"
    local command="$4"
    local max_attempts=30
    local attempt=1
    local status=""

    validate_input "$resource_type" "resource_type" || return 1
    validate_input "$resource_id" "resource_id" || return 1
    validate_input "$desired_status" "desired_status" || return 1

    echo "Waiting for $resource_type $resource_id to be in state $desired_status..."
 | tee -a "$LOG_FILE"

    while [ "$attempt" -le "$max_attempts" ]; do
        echo "Attempt $attempt of $max_attempts..." >> "$LOG_FILE"

        # Run the command to get the status and capture the output
        status_output=$(eval "$command" 2>&1) || true
        echo "$status_output" >> "$LOG_FILE"

        # For service networks, they do not have a status field in the output
        # We'll consider them active if we can retrieve them
        if [[ "$resource_type" == "Service Network" ]]; then
            if [[ "$status_output" == *"$resource_id"* ]]; then
                echo "$resource_type $resource_id is now active" | tee -a
 "$LOG_FILE"
                return 0
            fi
        else
            # For other resources, extract the status field
            status=$(echo "$status_output" | grep -i "status" | awk -F'"' '{print
 $4}' || true)
```

```
            echo "Current status: $status" >> "$LOG_FILE"

            if [[ "$status" == "$desired_status" ]]; then
                echo "$resource_type $resource_id is now in state $desired_status" |
 tee -a "$LOG_FILE"
                return 0
            elif [[ "$status" == *"FAIL"* ]]; then
                echo "ERROR: $resource_type $resource_id failed to reach desired
 state. Current status: $status" | tee -a "$LOG_FILE"
                return 1
            fi
        fi

        echo "Waiting for status change... (attempt $attempt/$max_attempts)" >>
 "$LOG_FILE"
        sleep 10
        ((attempt++))
    done

    echo "ERROR: Timed out waiting for $resource_type $resource_id to reach state
 $desired_status" | tee -a "$LOG_FILE"
    return 1
}

# Cleanup function for trap
cleanup() {
    local exit_code=$?
    echo "Script interrupted or failed. Cleaning up..." | tee -a "$LOG_FILE"
    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Check prerequisites
check_aws_cli

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
SERVICE_NETWORK_NAME="lattice-network-${RANDOM_ID}"
SERVICE_NAME="lattice-service-${RANDOM_ID}"

# Store created resources for cleanup
declare -a CREATED_RESOURCES
```

```
echo "=== VPC Lattice Service Network Tutorial ===" | tee -a "$LOG_FILE"
echo "Random ID for this session: ${RANDOM_ID}" | tee -a "$LOG_FILE"

# Step 1: Create a VPC Lattice service network
echo -e "\n=== Step 1: Creating a VPC Lattice service network ===" | tee -a
 "$LOG_FILE"
echo "Creating service network: $SERVICE_NETWORK_NAME" | tee -a "$LOG_FILE"

SERVICE_NETWORK_OUTPUT=$(log_command "aws vpc-lattice create-service-
network --name '$SERVICE_NETWORK_NAME' --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-vpc-lattice-gs --output json")
check_error $?

# Extract the service network ID using jq for safety
SERVICE_NETWORK_ID=$(echo "$SERVICE_NETWORK_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
if [ -z "$SERVICE_NETWORK_ID" ]; then
    echo "ERROR: Failed to extract service network ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_NETWORK_ID" "SERVICE_NETWORK_ID" || exit 1

echo "Service network created with ID: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Network: $SERVICE_NETWORK_ID")

# Wait for the service network to be active
wait_for_resource "Service Network" "$SERVICE_NETWORK_ID" "ACTIVE" "aws vpc-lattice
 get-service-network --service-network-identifier '$SERVICE_NETWORK_ID' --output
 json"
check_error $?

# Step 2: Create a VPC Lattice service
echo -e "\n=== Step 2: Creating a VPC Lattice service ===" | tee -a "$LOG_FILE"
echo "Creating service: $SERVICE_NAME" | tee -a "$LOG_FILE"

SERVICE_OUTPUT=$(log_command "aws vpc-lattice create-service --name '$SERVICE_NAME'
 --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-vpc-lattice-gs --
output json")
check_error $?

# Extract the service ID using jq for safety
SERVICE_ID=$(echo "$SERVICE_OUTPUT" | jq -r '.id // empty' 2>/dev/null || true)
if [ -z "$SERVICE_ID" ]; then
```

```
    echo "ERROR: Failed to extract service ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ID" "SERVICE_ID" || exit 1

echo "Service created with ID: $SERVICE_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service: $SERVICE_ID")

# Wait for the service to be active
wait_for_resource "Service" "$SERVICE_ID" "ACTIVE" "aws vpc-lattice get-service --
service-identifier '$SERVICE_ID' --output json"
check_error $?

# Step 3: Associate the service with the service network
echo -e "\n=== Step 3: Associating service with service network ===" | tee -a
 "$LOG_FILE"

SERVICE_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-service-
association --service-identifier '$SERVICE_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json")
check_error $?

# Extract the service association ID using jq for safety
SERVICE_ASSOC_ID=$(echo "$SERVICE_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/null
 || true)
if [ -z "$SERVICE_ASSOC_ID" ]; then
    echo "ERROR: Failed to extract service association ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ASSOC_ID" "SERVICE_ASSOC_ID" || exit 1

echo "Service association created with ID: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Association: $SERVICE_ASSOC_ID")

# Wait for the service association to be active
wait_for_resource "Service Association" "$SERVICE_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-service-association --service-network-service-
association-identifier '$SERVICE_ASSOC_ID' --output json"
check_error $?

# Step 4: List available VPCs to associate with the service network
echo -e "\n=== Step 4: Listing available VPCs ===" | tee -a "$LOG_FILE"
```

```
VPC_LIST=$(log_command "aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId,Tags[?Key==
\`Name\`].Value|[0]]' --output text")
check_error $?

echo "Available VPCs:" | tee -a "$LOG_FILE"
echo "$VPC_LIST" | tee -a "$LOG_FILE"

# Step 5: Auto-select first available VPC
echo -e "\n=== Step 5: Associate a VPC with the service network ===" | tee -a
 "$LOG_FILE"

VPC_ID=$(echo "$VPC_LIST" | head -n 1 | awk '{print $1}')

if [ -z "$VPC_ID" ]; then
    echo "WARNING: No VPC ID found" | tee -a "$LOG_FILE"
    echo "Skipping VPC association step" | tee -a "$LOG_FILE"
else
    validate_input "$VPC_ID" "VPC_ID" || {
        echo "ERROR: VPC_ID validation failed"
        exit 1
    }

    echo "Auto-selected VPC: $VPC_ID" | tee -a "$LOG_FILE"

    # Step 6: List security groups for the selected VPC
    echo -e "\n=== Step 6: Listing security groups for VPC $VPC_ID ===" | tee -a
 "$LOG_FILE"

    SG_LIST=$(log_command "aws ec2 describe-security-groups --filters Name=vpc-
id,Values='$VPC_ID' --query 'SecurityGroups[*].[GroupId,GroupName]' --output text")
    check_error $?

    echo "Available Security Groups for VPC $VPC_ID:" | tee -a "$LOG_FILE"
    echo "$SG_LIST" | tee -a "$LOG_FILE"

    # Step 7: Auto-select first available security group
    echo -e "\n=== Step 7: Select a security group for the VPC association ===" |
 tee -a "$LOG_FILE"

    SG_ID=$(echo "$SG_LIST" | head -n 1 | awk '{print $1}')

    if [ -z "$SG_ID" ]; then
        echo "WARNING: No Security Group ID found" | tee -a "$LOG_FILE"
```

```
        echo "Skipping VPC association step" | tee -a "$LOG_FILE"
    else
        validate_input "$SG_ID" "SG_ID" || {
            echo "ERROR: SG_ID validation failed"
            exit 1
        }

        echo "Auto-selected Security Group: $SG_ID" | tee -a "$LOG_FILE"

        # Step 8: Associate the VPC with the service network
        echo -e "\n=== Step 8: Associating VPC with service network ===" | tee -a
 "$LOG_FILE"

        VPC_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
vpc-association --vpc-identifier '$VPC_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --security-group-ids '$SG_ID' --output json")
        check_error $?

        # Extract the VPC association ID using jq for safety
        VPC_ASSOC_ID=$(echo "$VPC_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/null
 || true)
        if [ -z "$VPC_ASSOC_ID" ]; then
            echo "ERROR: Failed to extract VPC association ID" | tee -a "$LOG_FILE"
        else
            validate_input "$VPC_ASSOC_ID" "VPC_ASSOC_ID" || exit 1

            echo "VPC association created with ID: $VPC_ASSOC_ID" | tee -a
 "$LOG_FILE"
            CREATED_RESOURCES+=("VPC Association: $VPC_ASSOC_ID")

            # Wait for the VPC association to be active
            wait_for_resource "VPC Association" "$VPC_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-vpc-association --service-network-vpc-association-
identifier '$VPC_ASSOC_ID' --output json"
            check_error $?
        fi
    fi
fi

# Step 9: Display information about the created resources
echo -e "\n=== Step 9: Displaying information about created resources ===" | tee -a
 "$LOG_FILE"

echo "Service Network Details:" | tee -a "$LOG_FILE"
```

```
log_command "aws vpc-lattice get-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json"

echo "Service Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service --service-identifier '$SERVICE_ID' --output
 json"

echo "Service Network Service Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-service-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

echo "Service Network VPC Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-vpc-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

# Step 10: Cleanup - Auto-confirm
echo -e "\n=== Step 10: Resource Cleanup ===" | tee -a "$LOG_FILE"
echo "Resources created in this tutorial:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
done

echo "Starting cleanup process..." | tee -a "$LOG_FILE"

# Delete resources in reverse order

# Delete VPC association if it was created
if [[ -n "${VPC_ASSOC_ID:-}" ]]; then
    echo "Deleting VPC association: $VPC_ASSOC_ID" | tee -a "$LOG_FILE"
    log_command "aws vpc-lattice delete-service-network-vpc-association --service-
network-vpc-association-identifier '$VPC_ASSOC_ID'" || true

    # Wait for the VPC association to be deleted
    echo "Waiting for VPC association to be deleted..." | tee -a "$LOG_FILE"
    sleep 30
fi

# Delete service association
echo "Deleting service association: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network-service-association --service-
network-service-association-identifier '$SERVICE_ASSOC_ID'" || true

# Wait for the service association to be deleted
echo "Waiting for service association to be deleted..." | tee -a "$LOG_FILE"
```

```
sleep 30

# Delete service
echo "Deleting service: $SERVICE_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service --service-identifier '$SERVICE_ID'" ||
 true

# Wait for the service to be deleted
echo "Waiting for service to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service network
echo "Deleting service network: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID'" || true

echo "Cleanup completed successfully!" | tee -a "$LOG_FILE"

echo -e "\n=== Tutorial completed! ===" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateService
  - CreateServiceNetwork
  - CreateServiceNetworkServiceAssociation
  - CreateServiceNetworkVpcAssociation
  - DeleteService
  - DeleteServiceNetwork
  - DeleteServiceNetworkServiceAssociation
  - DeleteServiceNetworkVpcAssociation
  - DescribeSecurityGroups
  - DescribeVpcs
  - GetService
  - GetServiceNetwork
  - GetServiceNetworkServiceAssociation
  - GetServiceNetworkVpcAssociation
  - ListServiceNetworkServiceAssociations

- ListServiceNetworkVpcAssociations

Criação e gerenciamento de volumes de armazenamento em bloco

O exemplo de código a seguir mostra como:

- Criar um volume do EBS.
- Verificar o status do volume.
- Criar uma instância do EC2 (opcional).
- Anexar um volume a uma instância do EC2.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create and manage Amazon EBS volumes
# This script demonstrates how to create an EBS volume and attach it to an EC2
 instance
# It can also create a test EC2 instance if needed

# Set up logging
LOG_FILE="ebs-volume-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting EBS volume creation script at $(date)"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "$VOLUME_ID" ]; then
```

```
        echo "- EBS Volume: $VOLUME_ID"
    fi
    if [ -n "$INSTANCE_ID" ]; then
        echo "- EC2 Instance: $INSTANCE_ID"
    fi
    if [ -n "$SG_ID" ]; then
        echo "- Security Group: $SG_ID"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Do you want to clean up created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
    else
        echo "Resources were not cleaned up. You will need to delete them manually."
    fi

    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "$VOLUME_ID" ] && [ "$ATTACHED" = true ]; then
        echo "Detaching volume $VOLUME_ID..."
        aws ec2 detach-volume --volume-id "$VOLUME_ID"

        # Wait for volume to be detached
        echo "Waiting for volume to be detached..."
        aws ec2 wait volume-available --volume-ids "$VOLUME_ID"
    fi

    if [ -n "$VOLUME_ID" ]; then
        echo "Deleting volume $VOLUME_ID..."
        aws ec2 delete-volume --volume-id "$VOLUME_ID"
    fi

    if [ -n "$INSTANCE_ID" ] && [ "$CREATED_INSTANCE" = true ]; then
```

```
        echo "Terminating instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
    fi

    # Clean up security group if created
    if [ -n "$SG_ID" ] && [ "$CREATED_INSTANCE" = true ]; then
        echo "Deleting security group $SG_ID..."
        # Wait a bit for instance termination to complete
        sleep 10
        aws ec2 delete-security-group --group-id "$SG_ID" 2>/dev/null || echo
 "Security group may have dependencies, delete manually if needed"
    fi

    echo "Cleanup completed."
}

# Function to get available instance type
get_available_instance_type() {
    local region=$1

    # Try instance types in order of preference (cheapest first)
    local instance_types=("t3.nano" "t3.micro" "t2.micro" "t2.nano")

    for instance_type in "${instance_types[@]}"; do
        local available=$(aws ec2 describe-instance-type-offerings \
            --region "$region" \
            --filters "Name=instance-type,Values=$instance_type" \
            --query "length(InstanceTypeOfferings)" \
            --output text)

        if [ "$available" -gt 0 ]; then
            echo "$instance_type"
            return 0
        fi
    done

    # If none of the preferred types are available, get any available type
    local fallback_type=$(aws ec2 describe-instance-type-offerings \
        --region "$region" \
        --query "InstanceTypeOfferings[0].InstanceType" \
        --output text)
```

```
    if [ "$fallback_type" != "None" ] && [ -n "$fallback_type" ]; then
        echo "$fallback_type"
        return 0
    fi

    return 1
}

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 "AvailabilityZones[0].RegionName" --output text)
fi

echo "Using region: $REGION"

# Get available Availability Zones
echo "Retrieving available Availability Zones..."
AZ=$(aws ec2 describe-availability-zones --filters "Name=state,Values=available" --
query "AvailabilityZones[0].ZoneName" --output text)
if [ -z "$AZ" ]; then
    handle_error "Failed to retrieve Availability Zones"
fi
echo "Using Availability Zone: $AZ"

# Create a gp3 volume
echo "Creating a 10 GiB gp3 volume in $AZ..."
VOLUME_ID=$(aws ec2 create-volume \
    --volume-type gp3 \
    --size 10 \
    --availability-zone "$AZ" \
    --tag-specifications
 'ResourceType=volume,Tags=[{Key=Name,Value=EBSTutorialVolume},
{Key=Purpose,Value=Tutorial},{Key=project,Value=doc-smith},{Key=tutorial,Value=ebs-
gs-volumes}]' \
    --query 'VolumeId' \
    --output text)

if [ -z "$VOLUME_ID" ]; then
    handle_error "Failed to create EBS volume"
fi

echo "Volume created with ID: $VOLUME_ID"
```

```
# Wait for volume to become available
echo "Waiting for volume to become available..."
aws ec2 wait volume-available --volume-ids "$VOLUME_ID"
if [ $? -ne 0 ]; then
    handle_error "Volume did not become available"
fi

# Check volume details
echo "Retrieving volume details..."
aws ec2 describe-volumes --volume-ids "$VOLUME_ID"

# Ask if user wants to attach the volume to an instance
echo ""
echo "============================================"
echo "VOLUME ATTACHMENT"
echo "============================================"
echo "Do you want to attach this volume to an EC2 instance? (y/n): "
read -r ATTACH_CHOICE

ATTACHED=false
CREATED_INSTANCE=false
INSTANCE_ID=""
SG_ID=""

if [[ "$ATTACH_CHOICE" =~ ^[Yy]$ ]]; then
    # List available instances in the same AZ
    echo "Retrieving EC2 instances in $AZ..."
    INSTANCES_COUNT=$(aws ec2 describe-instances \
        --filters "Name=availability-zone,Values=$AZ" "Name=instance-state-
name,Values=running" \
        --query "length(Reservations[].Instances[])" \
        --output text)

    # Check if there are any running instances in the AZ
    if [ "$INSTANCES_COUNT" -eq 0 ]; then
        echo "No running instances found in $AZ."
        echo ""
        echo "Would you like to create a test EC2 instance? (y/n): "
        read -r CREATE_INSTANCE_CHOICE

        if [[ "$CREATE_INSTANCE_CHOICE" =~ ^[Yy]$ ]]; then
            # Get available instance type
            echo "Finding available instance type for region $REGION..."
```

```
            INSTANCE_TYPE=$(get_available_instance_type "$REGION")
            if [ $? -ne 0 ] || [ -z "$INSTANCE_TYPE" ]; then
                handle_error "No suitable instance type found in region $REGION"
            fi
            echo "Using instance type: $INSTANCE_TYPE"

            # Get the latest Amazon Linux 2 AMI
            echo "Finding the latest Amazon Linux 2 AMI..."
            AMI_ID=$(aws ec2 describe-images \
                --owners amazon \
                --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
                --query "sort_by(Images, &CreationDate)[-1].ImageId" \
                --output text)

            if [ -z "$AMI_ID" ]; then
                handle_error "Failed to find a suitable AMI"
            fi

            echo "Using AMI: $AMI_ID"

            # Check if a default VPC exists
            DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
                --filters "Name=isDefault,Values=true" \
                --query "Vpcs[0].VpcId" \
                --output text)

            if [ "$DEFAULT_VPC_ID" = "None" ] || [ -z "$DEFAULT_VPC_ID" ]; then
                handle_error "No default VPC found. Please create a VPC and subnet
 before running this script."
            fi

            # Get a subnet in the selected AZ
            SUBNET_ID=$(aws ec2 describe-subnets \
                --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=availability-
zone,Values=$AZ" \
                --query "Subnets[0].SubnetId" \
                --output text)

            if [ -z "$SUBNET_ID" ] || [ "$SUBNET_ID" = "None" ]; then
                handle_error "No subnet found in $AZ. Please create a subnet before
 running this script."
            fi
```

```
            echo "Using subnet: $SUBNET_ID"

            # Create a security group that allows SSH
            SG_NAME="EBSTutorialSG-$(date +%s)"
            SG_ID=$(aws ec2 create-security-group \
                --group-name "$SG_NAME" \
                --description "Security group for EBS tutorial" \
                --vpc-id "$DEFAULT_VPC_ID" \
                --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=ebs-gs-volumes}]' \
                --query "GroupId" \
                --output text)

            if [ -z "$SG_ID" ]; then
                handle_error "Failed to create security group"
            fi

            echo "Created security group: $SG_ID"

            # Add a rule to allow SSH
            aws ec2 authorize-security-group-ingress \
                --group-id "$SG_ID" \
                --protocol tcp \
                --port 22 \
                --cidr 0.0.0.0/0

            echo "Added SSH rule to security group"

            # Create the instance
            echo "Creating EC2 instance in $AZ with instance type $INSTANCE_TYPE..."
            INSTANCE_ID=$(aws ec2 run-instances \
                --image-id "$AMI_ID" \
                --instance-type "$INSTANCE_TYPE" \
                --subnet-id "$SUBNET_ID" \
                --security-group-ids "$SG_ID" \
                --tag-specifications
 'ResourceType=instance,Tags=[{Key=Name,Value=EBSTutorialInstance},
{Key=Purpose,Value=Tutorial},{Key=project,Value=doc-smith},{Key=tutorial,Value=ebs-
gs-volumes}]' \
                --query "Instances[0].InstanceId" \
                --output text)

            if [ -z "$INSTANCE_ID" ]; then
                handle_error "Failed to create EC2 instance"
```

```
            fi

            CREATED_INSTANCE=true
            echo "Instance created with ID: $INSTANCE_ID"

            # Wait for the instance to be running
            echo "Waiting for instance to be running..."
            aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

            # Wait a bit more for the instance to initialize
            echo "Waiting for instance initialization (30 seconds)..."
            sleep 30
        else
            echo "Skipping instance creation and volume attachment."
            INSTANCE_ID=""
        fi
    else
        # Display available instances
        echo "Available instances in $AZ:"
        aws ec2 describe-instances \
            --filters "Name=availability-zone,Values=$AZ" "Name=instance-state-
name,Values=running" \
            --query "Reservations[*].Instances[*].[InstanceId,Tags[?
Key=='Name'].Value|[0],InstanceType]" \
            --output table

        # Ask for instance ID
        echo ""
        echo "Enter the instance ID to attach the volume to (or press Enter to
 skip): "
        read -r INSTANCE_ID
    fi

    if [ -n "$INSTANCE_ID" ]; then
        # Attach volume to the instance
        echo "Attaching volume $VOLUME_ID to instance $INSTANCE_ID..."
        ATTACH_RESULT=$(aws ec2 attach-volume \
            --volume-id "$VOLUME_ID" \
            --instance-id "$INSTANCE_ID" \
            --device "/dev/sdf" \
            --query 'State' \
            --output text)

        if [ $? -ne 0 ] || [ -z "$ATTACH_RESULT" ]; then
```

```
            handle_error "Failed to attach volume to instance"
        fi

        ATTACHED=true
        echo "Volume attached successfully. Device: /dev/sdf"

        # Verify attachment
        echo "Verifying attachment..."
        aws ec2 describe-volumes \
            --volume-ids "$VOLUME_ID" \
            --query "Volumes[0].Attachments"
    else
        echo "Skipping volume attachment."
    fi
else
    echo "Skipping volume attachment."
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "Created resources:"
echo "- EBS Volume: $VOLUME_ID"
if [ "$ATTACHED" = true ]; then
    echo "  - Attached to: $INSTANCE_ID as /dev/sdf"
fi
if [ "$CREATED_INSTANCE" = true ]; then
    echo "- EC2 Instance: $INSTANCE_ID (type: $INSTANCE_TYPE)"
fi
if [ -n "$SG_ID" ]; then
    echo "- Security Group: $SG_ID"
fi

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
```

```
    cleanup_resources
else
    echo ""
    echo "Resources were not cleaned up. You can manually delete them later."
    if [ -n "$VOLUME_ID" ]; then
        if [ "$ATTACHED" = true ]; then
            echo "To detach the volume:"
            echo "  aws ec2 detach-volume --volume-id $VOLUME_ID"
        fi
        echo "To delete the volume:"
        echo "  aws ec2 delete-volume --volume-id $VOLUME_ID"
    fi
    if [ "$CREATED_INSTANCE" = true ]; then
        echo "To terminate the instance:"
        echo "  aws ec2 terminate-instances --instance-ids $INSTANCE_ID"
    fi
    if [ -n "$SG_ID" ]; then
        echo "To delete the security group (after instance termination):"
        echo "  aws ec2 delete-security-group --group-id $SG_ID"
    fi
fi

echo ""
echo "Script completed at $(date)"
echo "================================================"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachVolume
  - AuthorizeSecurityGroupIngress
  - CreateSecurityGroup
  - CreateVolume
  - DeleteSecurityGroup
  - DeleteVolume
  - DescribeAvailabilityZones
  - DescribeImages
  - DescribeInstanceTypeOfferings
  - DescribeInstances
  - DescribeSubnets

- DescribeVolumes
- DescribeVpcs
- DetachVolume
- RunInstances
- TerminateInstances
- Wait

Conceitos básicos de conexões de rede dedicadas

O exemplo de código a seguir mostra como:

- Use ec2 CreateVpnGateway
- Use ec2 DeleteVpnGateway
- Use ec2 DescribeVpnGateways
- Use conexão direta CreateConnection
- Use conexão direta CreatePrivateVirtualInterface
- Use conexão direta DeleteConnection

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Direct Connect Connection Management Script - Version 6
# This script demonstrates how to create and manage AWS Direct Connect connections
 using the AWS CLI
# This version includes fixes for user input handling and better error reporting

# Set up logging
LOG_FILE="directconnect-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "$(date): Starting AWS Direct Connect script v6"

# Function to check for errors in command output
check_error() {
    local output=$1
    local command=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $command"
        echo "Output: $output"
        cleanup_resources
        exit 1
    fi
}

# Function to wait for VGW to be available
wait_for_vgw() {
    local vgw_id=$1
    local max_attempts=30
    local attempt=1

    echo "Waiting for virtual private gateway $vgw_id to become available..."

    while [ $attempt -le $max_attempts ]; do
        VGW_STATE=$(aws ec2 describe-vpn-gateways --vpn-gateway-ids "$vgw_id" --
query 'VpnGateways[0].State' --output text)

        if [ "$VGW_STATE" == "available" ]; then
            echo "Virtual private gateway is now available"
            return 0
        elif [ "$VGW_STATE" == "failed" ]; then
            echo "Virtual private gateway failed to become available"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: VGW state is $VGW_STATE, waiting 10
 seconds..."
        sleep 10
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for VGW to become available"
    return 1
```

```
}

# Function to wait for connection to be available
wait_for_connection() {
    local connection_id=$1
    local max_attempts=60
    local attempt=1

    echo "Waiting for connection $connection_id to become available..."
    echo "Note: This can take 30+ minutes in production as AWS provisions the
 physical connection"

    while [ $attempt -le $max_attempts ]; do
        CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$connection_id" --query 'connections[0].connectionState' --output text)

        if [ "$CONNECTION_STATE" == "available" ]; then
            echo "Connection is now available"
            return 0
        elif [ "$CONNECTION_STATE" == "rejected" ] || [ "$CONNECTION_STATE" ==
 "deleted" ]; then
            echo "Connection failed with state: $CONNECTION_STATE"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: Connection state is $CONNECTION_STATE,
 waiting 30 seconds..."
        sleep 30
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for connection to become available"
    return 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete virtual interfaces if they exist
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "Deleting private virtual interface: $PRIVATE_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PRIVATE_VIF_ID"
```

```
    fi

    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "Deleting public virtual interface: $PUBLIC_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PUBLIC_VIF_ID"
    fi

    # Delete connection if it exists
    if [ -n "$CONNECTION_ID" ]; then
        echo "Deleting connection: $CONNECTION_ID"
        aws directconnect delete-connection --connection-id "$CONNECTION_ID"
    fi

    # Delete VGW if it exists
    if [ -n "$VGW_ID" ]; then
        echo "Deleting virtual private gateway: $VGW_ID"
        aws ec2 delete-vpn-gateway --vpn-gateway-id "$VGW_ID"
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 6)
CONNECTION_NAME="DxConn-${RANDOM_ID}"

# Step 1: List available Direct Connect locations
echo "Listing available Direct Connect locations..."
LOCATIONS_OUTPUT=$(aws directconnect describe-locations)
check_error "$LOCATIONS_OUTPUT" "describe-locations"
echo "$LOCATIONS_OUTPUT"

# Extract the first location code for demonstration purposes
LOCATION_CODE=$(aws directconnect describe-locations --query
 'locations[0].locationCode' --output text)
if [ -z "$LOCATION_CODE" ] || [ "$LOCATION_CODE" == "None" ]; then
    echo "Error: Could not extract location code from the output."
    exit 1
fi
echo "Using location: $LOCATION_CODE"

# Step 2: Create a dedicated connection
echo "Creating a dedicated connection at location $LOCATION_CODE with bandwidth
 1Gbps..."
CONNECTION_OUTPUT=$(aws directconnect create-connection \
```

```
    --location "$LOCATION_CODE" \
    --bandwidth "1Gbps" \
    --connection-name "$CONNECTION_NAME" \
    --tags key=project,value=doc-smith key=tutorial,value=aws-direct-connect-gs)
check_error "$CONNECTION_OUTPUT" "create-connection"
echo "$CONNECTION_OUTPUT"

# Extract connection ID directly from the output
CONNECTION_ID=$(echo "$CONNECTION_OUTPUT" | grep -o '"connectionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$CONNECTION_ID" ]; then
    echo "Error: Could not extract connection ID from the output."
    exit 1
fi
echo "Connection created with ID: $CONNECTION_ID"

# Step 3: Describe the connection
echo "Retrieving connection details..."
DESCRIBE_OUTPUT=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID")
check_error "$DESCRIBE_OUTPUT" "describe-connections"
echo "$DESCRIBE_OUTPUT"

# Step 4: Update the connection name
NEW_CONNECTION_NAME="${CONNECTION_NAME}-updated"
echo "Updating connection name to $NEW_CONNECTION_NAME..."
UPDATE_OUTPUT=$(aws directconnect update-connection \
    --connection-id "$CONNECTION_ID" \
    --connection-name "$NEW_CONNECTION_NAME")
check_error "$UPDATE_OUTPUT" "update-connection"
echo "$UPDATE_OUTPUT"

# Step 5: Check if we can download the LOA-CFA
# Note: In a real scenario, the LOA-CFA might not be immediately available
echo "Attempting to download the LOA-CFA (this may not be available yet)..."
LOA_OUTPUT=$(aws directconnect describe-loa --connection-id "$CONNECTION_ID" 2>&1)
if echo "$LOA_OUTPUT" | grep -i "error" > /dev/null; then
    echo "LOA-CFA not available yet. This is expected for newly created
 connections."
    echo "The LOA-CFA will be available once AWS begins provisioning your
 connection."
else
    LOA_CONTENT=$(echo "$LOA_OUTPUT" | grep -o '"loaContent": "[^"]*' | cut -d'"' -
f4)
```

```
    echo "$LOA_CONTENT" | base64 --decode > "loa-cfa-${CONNECTION_ID}.pdf"
    echo "LOA-CFA downloaded to loa-cfa-${CONNECTION_ID}.pdf"
fi

# Step 6: Create a virtual private gateway (required for private virtual interface)
echo "Creating a virtual private gateway..."
VGW_OUTPUT=$(aws ec2 create-vpn-gateway --type ipsec.1 \
    --tag-specifications 'ResourceType=vpn-gateway,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=aws-direct-connect-gs}]')
check_error "$VGW_OUTPUT" "create-vpn-gateway"
echo "$VGW_OUTPUT"

# Extract VGW ID directly from the output
VGW_ID=$(echo "$VGW_OUTPUT" | grep -o '"VpnGatewayId": "[^"]*' | cut -d'"' -f4)
if [ -z "$VGW_ID" ]; then
    echo "Error: Could not extract VPN gateway ID from the output."
    exit 1
fi
echo "Virtual private gateway created with ID: $VGW_ID"

# Wait for VGW to become available
if ! wait_for_vgw "$VGW_ID"; then
    echo "Failed to wait for VGW to become available. Skipping virtual interface
 creation."
    VIF_CREATION_SKIPPED=true
else
    VIF_CREATION_SKIPPED=false
fi

# Step 7: Create a private virtual interface (only if VGW is available)
if [ "$VIF_CREATION_SKIPPED" = false ]; then
    echo "Creating a private virtual interface..."
    PRIVATE_VIF_OUTPUT=$(aws directconnect create-private-virtual-interface \
        --connection-id "$CONNECTION_ID" \
        --new-private-virtual-interface '{
            "virtualInterfaceName": "PrivateVIF-'"$RANDOM_ID"'",
            "vlan": 100,
            "asn": 65000,
            "authKey": "'"$RANDOM_ID"'key",
            "amazonAddress": "192.168.1.1/30",
            "customerAddress": "192.168.1.2/30",
            "addressFamily": "ipv4",
            "virtualGatewayId": "'"$VGW_ID"'"
        }' 2>&1)
```

```
    if echo "$PRIVATE_VIF_OUTPUT" | grep -i "error" > /dev/null; then
        echo "Could not create private virtual interface. This is expected if the
 connection is not yet available."
        echo "Error: $PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=""
    else
        echo "$PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=$(echo "$PRIVATE_VIF_OUTPUT" | grep -o '"virtualInterfaceId":
 "[^"]*' | cut -d'"' -f4)
        echo "Private virtual interface created with ID: $PRIVATE_VIF_ID"
    fi
else
    echo "Skipping private virtual interface creation due to VGW not being
 available"
    PRIVATE_VIF_ID=""
fi

# Step 8: Check connection state and provide guidance for public virtual interface
CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID" --query 'connections[0].connectionState' --output text)
echo "Current connection state: $CONNECTION_STATE"

if [ "$CONNECTION_STATE" != "available" ]; then
    echo ""
    echo "============================================"
    echo "CONNECTION NOT YET AVAILABLE"
    echo "============================================"
    echo "The connection is in '$CONNECTION_STATE' state."
    echo "In production, you would:"
    echo "1. Wait for AWS to provision the connection (can take 30+ minutes)"
    echo "2. Download the LOA-CFA when available"
    echo "3. Provide the LOA-CFA to your network provider for cross-connect"
    echo "4. Create virtual interfaces once connection is 'available'"
    echo ""

    # Ask if user wants to wait for connection to become available
    echo ""
    echo "============================================"
    echo "CONNECTION WAIT CONFIRMATION"
    echo "============================================"
    echo -n "Do you want to wait for the connection to become available? (y/n): "
    read -r WAIT_CHOICE
```

```
    if [[ "$WAIT_CHOICE" =~ ^[Yy]$ ]]; then
        if wait_for_connection "$CONNECTION_ID"; then
            echo "Connection is now available! You could now create virtual
 interfaces."
        else
            echo "Connection did not become available within the timeout period."
        fi
    else
        echo "Skipping wait for connection availability."
    fi
else
    echo "Connection is available! Virtual interfaces can be created."
fi

# Step 9: List all virtual interfaces
echo "Listing all virtual interfaces..."
VIF_LIST_OUTPUT=$(aws directconnect describe-virtual-interfaces)
check_error "$VIF_LIST_OUTPUT" "describe-virtual-interfaces"
echo "$VIF_LIST_OUTPUT"

# Step 10: Display important information about production usage
echo ""
echo "============================================"
echo "IMPORTANT PRODUCTION NOTES"
echo "============================================"
echo "1. Direct Connect connections take time to be provisioned by AWS"
echo "2. You cannot create virtual interfaces until the connection is 'available'"
echo "3. For public virtual interfaces, you must own the public IP addresses"
echo "4. LOA-CFA (Letter of Authorization) is needed for cross-connect at the
 facility"
echo "5. This demo creates resources that incur costs (~\$300/month for 1Gbps)"
echo "6. Always test connectivity before putting into production"
echo ""

# Step 11: Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo -n "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
```

```
    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually delete them later."
    echo ""
    echo "Created resources:"
    echo "- Connection ID: $CONNECTION_ID"
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "- Private Virtual Interface ID: $PRIVATE_VIF_ID"
    fi
    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "- Public Virtual Interface ID: $PUBLIC_VIF_ID"
    fi
    echo "- Virtual Private Gateway ID: $VGW_ID"
    echo ""
    echo "Manual cleanup commands:"
    echo "aws directconnect delete-connection --connection-id $CONNECTION_ID"
    echo "aws ec2 delete-vpn-gateway --vpn-gateway-id $VGW_ID"
    echo ""
    echo "Remember: Direct Connect resources incur ongoing costs!"
fi

echo "$(date): Script completed"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateConnection
  - CreatePrivateVirtualInterface
  - CreateVpnGateway
  - DeleteConnection
  - DeleteVirtualInterface
  - DeleteVpnGateway
  - DescribeConnections
  - DescribeLoa
  - DescribeLocations
  - DescribeVirtualInterfaces
  - DescribeVpnGateways
  - UpdateConnection

Conceitos básicos de compras do mercado de software

O exemplo de código a seguir mostra como:

- Use ec2 AuthorizeSecurityGroupIngress
- Use ec2 CreateKeyPair
- Use ec2 CreateSecurityGroup

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Marketplace Buyer Getting Started Script
# This script demonstrates how to search for products in AWS Marketplace,
# launch an EC2 instance with a product AMI, and manage subscriptions.

set -euo pipefail

# Setup logging with secure permissions
LOG_FILE="marketplace-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS Marketplace Buyer Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. List available products in AWS Marketplace"
echo "2. Create resources needed to launch an EC2 instance"
echo "3. Launch an EC2 instance with an Amazon Linux 2 AMI"
echo "4. Show how to manage and terminate the instance"
echo "====================================================="
echo ""
```

```
# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it first."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured. Please configure them first."
    exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
    echo "ERROR: jq is not installed. Please install jq for safe JSON parsing."
    exit 1
fi

# Function to safely extract JSON values using jq
extract_json_value() {
    local json=$1
    local query=$2

    echo "$json" | jq -r "$query" 2>/dev/null || {
        echo "ERROR: Failed to parse JSON with query: $query" >&2
        return 1
    }
}

# Function to validate AWS permissions
validate_aws_permissions() {
    echo "Validating AWS permissions..."

    local identity
    identity=$(aws sts get-caller-identity --output json)
    local account_id
    account_id=$(extract_json_value "$identity" '.Account') || return 1
    local arn
    arn=$(extract_json_value "$identity" '.Arn') || return 1

    echo "AWS Account ID: $account_id"
    echo "AWS Principal ARN: $arn"
    echo "Note: This script requires EC2 permissions for key pair, security group,
 and instance management."
```

```
    echo ""
}

# Function to clean up resources
cleanup_resources() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="

    if [ -n "${INSTANCE_ID:-}" ]; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" > /dev/null 2>&1 || true

        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" 2>/dev/null || true
        echo "Instance terminated successfully."
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting before deleting security group..."
        sleep 5
        echo "Deleting security group: $SECURITY_GROUP_ID"
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID" --region
 "$AWS_REGION" > /dev/null 2>&1 || true
        echo "Security group deleted."
    fi

    if [ -n "${KEY_NAME:-}" ]; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" --region "$AWS_REGION" > /
dev/null 2>&1 || true

        # Remove the local key file if it exists with secure deletion
        if [ -f "${KEY_NAME}.pem" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "${KEY_NAME}.pem" 2>/dev/null || rm -f
 "${KEY_NAME}.pem"
            else
                rm -f "${KEY_NAME}.pem"
            fi
            echo "Local key file securely deleted."
```

```
        fi
    fi

    echo "Cleanup completed."
}

# Set trap to ensure cleanup on script exit
trap cleanup_resources EXIT

# Get the current AWS region
AWS_REGION=$(aws configure get region || echo "us-east-1")
if [ -z "$AWS_REGION" ] || [ "$AWS_REGION" = "None" ]; then
    AWS_REGION="us-east-1"
fi
echo "Using AWS Region: $AWS_REGION"
echo ""

# Validate permissions
validate_aws_permissions

# Generate random identifier for resource names using cryptographically secure
 method
RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
KEY_NAME="marketplace-key-${RANDOM_ID}"
SECURITY_GROUP_NAME="marketplace-sg-${RANDOM_ID}"

# Initialize variables to track created resources
INSTANCE_ID=""
SECURITY_GROUP_ID=""
AMI_ID=""

# Step 1: List available products in AWS Marketplace
echo "Listing available products in AWS Marketplace..."
echo "Note: In a real scenario, you would use marketplace-catalog commands to list
 and search for products."
echo "However, this requires specific permissions and product knowledge."
echo ""
echo "For this tutorial, we'll use a public Amazon Linux 2 AMI instead of an actual
 marketplace product."
echo "This is because subscribing to marketplace products requires accepting terms
 via the console."
echo ""

# Step 2: Create a key pair for SSH access
```

```
echo "Creating key pair: $KEY_NAME"
KEY_OUTPUT=$(aws ec2 create-key-pair \
  --key-name "$KEY_NAME" \
  --region "$AWS_REGION" \
  --query 'KeyMaterial' \
  --output text) || {
    echo "ERROR: Failed to create key pair" >&2
    exit 1
}

# Securely save the key with restricted permissions
if ! echo "$KEY_OUTPUT" > "${KEY_NAME}.pem" 2>/dev/null; then
    echo "ERROR: Failed to write key file ${KEY_NAME}.pem" >&2
    exit 1
fi
chmod 600 "${KEY_NAME}.pem" || {
    echo "ERROR: Failed to set permissions on key file" >&2
    rm -f "${KEY_NAME}.pem"
    exit 1
}
echo "Key pair created and saved to ${KEY_NAME}.pem with secure permissions (600)"

# Step 3: Create a security group
echo "Creating security group: $SECURITY_GROUP_NAME"
SG_OUTPUT=$(aws ec2 create-security-group \
  --group-name "$SECURITY_GROUP_NAME" \
  --description "Security group for AWS Marketplace tutorial" \
  --region "$AWS_REGION" \
  --output json) || {
    echo "ERROR: Failed to create security group" >&2
    exit 1
}

# Extract security group ID using jq for safe parsing
SECURITY_GROUP_ID=$(extract_json_value "$SG_OUTPUT" '.GroupId') || exit 1
if [ -z "$SECURITY_GROUP_ID" ] || [ "$SECURITY_GROUP_ID" = "null" ]; then
    echo "ERROR: Could not extract security group ID" >&2
    exit 1
fi
echo "Security group created with ID: $SECURITY_GROUP_ID"

# Add inbound rules for SSH and HTTP in parallel for better performance
echo "Configuring security group rules..."
{
```

```
    aws ec2 authorize-security-group-ingress \
      --group-id "$SECURITY_GROUP_ID" \
      --protocol tcp \
      --port 22 \
      --cidr 0.0.0.0/0 \
      --region "$AWS_REGION" > /dev/null 2>&1
} &
SSH_PID=$!

{
    aws ec2 authorize-security-group-ingress \
      --group-id "$SECURITY_GROUP_ID" \
      --protocol tcp \
      --port 80 \
      --cidr 0.0.0.0/0 \
      --region "$AWS_REGION" > /dev/null 2>&1
} &
HTTP_PID=$!

# Wait for both operations to complete
wait $SSH_PID || {
    echo "ERROR: Failed to add SSH ingress rule" >&2
    exit 1
}
wait $HTTP_PID || {
    echo "ERROR: Failed to add HTTP ingress rule" >&2
    exit 1
}

echo "Security group configured with SSH and HTTP access."
echo "WARNING: SSH rule allows access from any IP (0.0.0.0/0). Restrict this in
 production."
echo "WARNING: In a production environment, you should restrict access to specific
 IP ranges."
echo ""

# Step 4: Get the latest Amazon Linux 2 AMI ID - Use pagination to optimize costs
echo "Getting the latest Amazon Linux 2 AMI ID..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-2.0.*-x86_64-gp2"
 "Name=state,Values=available" \
  --region "$AWS_REGION" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
```

```
  --output text \
  --max-results 50) || {
    echo "ERROR: Failed to describe images" >&2
    exit 1
}

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    echo "ERROR: Could not find a suitable AMI ID" >&2
    exit 1
fi

echo "Using AMI ID: $AMI_ID"
echo "Note: In a real marketplace scenario, you would use the AMI ID from your
 subscribed product."
echo ""

# Step 5: Launch an EC2 instance with cost optimization
echo "Launching EC2 instance with the AMI..."
echo "Using t2.micro (eligible for AWS Free Tier if applicable)"
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --count 1 \
  --region "$AWS_REGION" \
  --monitoring Enabled=false \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=marketplace-buyer-gs}]' \
  --output json) || {
    echo "ERROR: Failed to launch instance" >&2
    exit 1
}

# Extract instance ID using jq for safe parsing
INSTANCE_ID=$(extract_json_value "$INSTANCE_OUTPUT" '.Instances[0].InstanceId') ||
 exit 1
if [ -z "$INSTANCE_ID" ] || [ "$INSTANCE_ID" = "null" ]; then
    echo "ERROR: Could not extract instance ID" >&2
    exit 1
fi
echo "Instance launched with ID: $INSTANCE_ID"

# Wait for the instance to be running
```

```
echo "Waiting for instance to be in running state..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID" --region "$AWS_REGION"
 || {
    echo "ERROR: Instance failed to reach running state" >&2
    exit 1
}
echo "Instance is now running."
echo ""

# Step 6: Get instance details
echo "Getting instance details..."
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --region "$AWS_REGION" \
  --output json) || {
    echo "ERROR: Failed to describe instance" >&2
    exit 1
}

echo "Instance details:"
extract_json_value "$INSTANCE_DETAILS" '.Reservations[0].Instances[0] | {InstanceId,
 State: .State.Name, PublicDnsName, PrivateIpAddress, LaunchTime, InstanceType}' ||
 exit 1

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Key Pair: $KEY_NAME"
echo "Security Group: $SECURITY_GROUP_NAME (ID: $SECURITY_GROUP_ID)"
echo "EC2 Instance: $INSTANCE_ID"
echo "Instance Type: t2.micro (cost-optimized)"
echo "AMI ID: $AMI_ID"
echo "Region: $AWS_REGION"
echo ""
echo "COST OPTIMIZATION NOTES:"
echo "- t2.micro instances are eligible for AWS Free Tier (750 hours/month for 12
 months)"
echo "- Detailed monitoring is disabled to reduce costs"
echo "- Consider using Spot Instances for non-production workloads"
echo "- Review AWS Pricing Calculator: https://calculator.aws/"
echo ""
echo "To connect to your instance (once it's fully initialized):"
```

```
echo "ssh -i ${KEY_NAME}.pem ec2-user@<public-dns-name>"
echo "Replace <public-dns-name> with the PublicDnsName from the instance details
 above."
echo ""

# Auto-confirm cleanup of resources
echo "====================================================="
echo "CLEANUP CONFIRMATION"
echo "====================================================="
echo "Cleaning up all created resources..."

echo ""
echo "Script completed. See $LOG_FILE for the complete log."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - RunInstances
  - TerminateInstances
  - Wait

Introdução aos clusters de processamento de big data

O exemplo de código a seguir mostra como:

- Criar um par de chaves do EC2.
- Configurar o armazenamento e preparar uma aplicação.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
```

```
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null || true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ]; then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi
```

```
# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-configuration
 Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
```

```
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to find
 the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as 's3://
emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-tutorial-
bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
```

```
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
```

```
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output text
 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
```

```
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"
```

```
        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"
```

```
# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID"
 --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv" |
 head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv" |
 head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"
```

```
# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb
  - Rb
  - Rm
  - ssh
  - TerminateClusters
  - Wait

Introdução aos bancos de dados de documentos

O exemplo de código a seguir mostra como:

- Criar um grupo de sub-redes de banco de dados
- Criar um cluster do DocumentDB.
- Criar uma instância do DocumentDB.
- Configurar a segurança e a conectividade.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

################################################################################
# Configuration
################################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
```

```
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

################################################################################
# Logging
################################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""

################################################################################
# Region pre-check
################################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

################################################################################
# Error handler
################################################################################
handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
```

```
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

##############################################################################
# Wait function
##############################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
```

```
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound\|
not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
```

```
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG ingress
 rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed to
 delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete cluster."
```

```
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}

##############################################################################
# Step 1: Generate password and store in Secrets Manager
##############################################################################
echo "============================================"
echo "Step 1: Create master password in Secrets Manager"
echo "============================================"
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
```

```
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "==========================================="
echo "Step 2: Find default VPC and subnets"
echo "==========================================="
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"
```

```
# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

################################################################################
# Step 3: Create subnet group
################################################################################
echo "==========================================="
echo "Step 3: Create DocumentDB subnet group"
echo "==========================================="
echo ""
```

```
SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

##############################################################################
# Step 4: Create DocumentDB cluster
##############################################################################
echo "============================================"
echo "Step 4: Create DocumentDB cluster"
echo "============================================"
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""
```

```
if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

################################################################################
# Step 5: Create DocumentDB instance
################################################################################
echo "==========================================="
echo "Step 5: Create DocumentDB instance"
echo "==========================================="
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

################################################################################
# Step 6: Get cluster endpoint and security group
################################################################################
```

```
echo "============================================"
echo "Step 6: Get cluster endpoint and security group"
echo "============================================"
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
##############################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null ||
 true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')
```

```
    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "==========================================="
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "==========================================="
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
```

```
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

################################################################################
# Step 9: Display connection information
################################################################################
echo "============================================"
echo "CONNECTION INFORMATION"
echo "============================================"
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH} \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
```

```
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString --
output text)"
echo ""

################################################################################
# Step 10: Cleanup
################################################################################
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances

- DescribeSubnets
- DescribeVpcs
- GetSecretValue
- RevokeSecurityGroupIngress
- Wait

Introdução aos bancos de dados de grafos

O exemplo de código a seguir mostra como:

- Criar uma VPC para um banco de dados do Neptune.
- Criar sub-redes em várias zonas de disponibilidade.
- Configurar a segurança do banco de dados do Neptune.
- Criar um grupo de sub-redes do Neptune.
- Criar uma instância e um cluster de banco de dados do Neptune.
- Adicionar dados a um banco de dados de grafos.
- Consultar um banco de dados de grafos.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Neptune Getting Started Script
# This script creates an Amazon Neptune database cluster and demonstrates basic
 operations

# Set up logging
LOG_FILE="neptune-setup.log"
echo "Starting Neptune setup at $(date)" > "$LOG_FILE"
```

```
# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..." | tee -a "$LOG_FILE"

    # Only attempt to delete resources that were successfully created
    if [[ -n "$DB_INSTANCE_ID" ]]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"
        log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"
    fi

    if [[ -n "$DB_CLUSTER_ID" ]]; then
        echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --skip-final-snapshot"
    fi

    if [[ -n "$DB_SUBNET_GROUP" ]]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"
    fi
```

```
    if [[ -n "$SECURITY_GROUP_ID" ]]; then
        echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
    fi

    if [[ -n "$SUBNET_IDS" ]]; then
        for SUBNET_ID in $SUBNET_IDS; do
            echo "Deleting subnet $SUBNET_ID..." | tee -a "$LOG_FILE"
            log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET_ID"
        done
    fi

    if [[ -n "$VPC_ID" ]]; then
        echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"
    fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
VPC_NAME="neptune-vpc-$RANDOM_ID"
DB_SUBNET_GROUP="neptune-subnet-group-$RANDOM_ID"
DB_CLUSTER_ID="neptune-cluster-$RANDOM_ID"
DB_INSTANCE_ID="neptune-instance-$RANDOM_ID"
SG_NAME="neptune-sg-$RANDOM_ID"

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "VPC Name: $VPC_NAME" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "DB Cluster ID: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "DB Instance ID: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "Security Group Name: $SG_NAME" | tee -a "$LOG_FILE"

# Step 1: Create VPC
echo "Creating VPC..." | tee -a "$LOG_FILE"
VPC_OUTPUT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=$VPC_NAME},{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]" --output json)
check_error "$VPC_OUTPUT" $? "Failed to create VPC"

VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID" | tee -a "$LOG_FILE"
```

```
# Enable DNS support for the VPC
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support"
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames"

# Step 2: Create Internet Gateway and attach to VPC
echo "Creating Internet Gateway..." | tee -a "$LOG_FILE"
IGW_OUTPUT=$(aws ec2 create-internet-gateway --tag-specifications
 'ResourceType=internet-gateway,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$IGW_OUTPUT" $? "Failed to create Internet Gateway"

IGW_ID=$(echo "$IGW_OUTPUT" | grep -o '"InternetGatewayId": "[^"]*' | cut -d'"' -f4)
echo "Internet Gateway created with ID: $IGW_ID" | tee -a "$LOG_FILE"

log_cmd "aws ec2 attach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"

# Step 3: Create subnets in different AZs
echo "Creating subnets..." | tee -a "$LOG_FILE"

# Get available AZs
AZ_OUTPUT=$(aws ec2 describe-availability-zones --output json)
check_error "$AZ_OUTPUT" $? "Failed to get availability zones"

# Extract first 3 AZ names
AZ1=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -1)
AZ2=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -2 |
 tail -1)
AZ3=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -3 |
 tail -1)

# Create 3 subnets in different AZs
SUBNET1_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.1.0/24 --availability-zone $AZ1 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET1_OUTPUT" $? "Failed to create subnet 1"
SUBNET1_ID=$(echo "$SUBNET1_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET2_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.2.0/24 --availability-zone $AZ2 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET2_OUTPUT" $? "Failed to create subnet 2"
```

```
SUBNET2_ID=$(echo "$SUBNET2_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET3_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.3.0/24 --availability-zone $AZ3 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET3_OUTPUT" $? "Failed to create subnet 3"
SUBNET3_ID=$(echo "$SUBNET3_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET_IDS="$SUBNET1_ID $SUBNET2_ID $SUBNET3_ID"
echo "Created subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"

# Step 4: Create route table and add route to Internet Gateway
echo "Creating route table..." | tee -a "$LOG_FILE"
ROUTE_TABLE_OUTPUT=$(aws ec2 create-route-table --vpc-id $VPC_ID --tag-
specifications 'ResourceType=route-table,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$ROUTE_TABLE_OUTPUT" $? "Failed to create route table"

ROUTE_TABLE_ID=$(echo "$ROUTE_TABLE_OUTPUT" | grep -o '"RouteTableId": "[^"]*' | cut
 -d'"' -f4)
echo "Route table created with ID: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"

# Add route to Internet Gateway
log_cmd "aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-cidr-
block 0.0.0.0/0 --gateway-id $IGW_ID"

# Associate route table with subnets
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET1_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET2_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET3_ID"

# Step 5: Create security group
echo "Creating security group..." | tee -a "$LOG_FILE"
SG_OUTPUT=$(aws ec2 create-security-group --group-name $SG_NAME --description
 "Security group for Neptune" --vpc-id $VPC_ID --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SG_OUTPUT" $? "Failed to create security group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
```

```
echo "Security group created with ID: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"

# Add inbound rule for Neptune port (8182)
# Note: In production, you should restrict this to specific IP ranges
echo "Adding security group rule for Neptune port 8182..." | tee -a "$LOG_FILE"
log_cmd "aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID --
protocol tcp --port 8182 --cidr 10.0.0.0/16"

# Step 6: Create DB subnet group
echo "Creating DB subnet group..." | tee -a "$LOG_FILE"
DB_SUBNET_GROUP_OUTPUT=$(aws neptune create-db-subnet-group --db-subnet-group-
name $DB_SUBNET_GROUP --db-subnet-group-description "Subnet group for Neptune" --
subnet-ids $SUBNET1_ID $SUBNET2_ID $SUBNET3_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_SUBNET_GROUP_OUTPUT" $? "Failed to create DB subnet group"
echo "DB subnet group created: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"

# Step 7: Create Neptune DB cluster
echo "Creating Neptune DB cluster..." | tee -a "$LOG_FILE"
DB_CLUSTER_OUTPUT=$(aws neptune create-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --engine neptune --vpc-security-group-ids $SECURITY_GROUP_ID
 --db-subnet-group-name $DB_SUBNET_GROUP --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_CLUSTER_OUTPUT" $? "Failed to create Neptune DB cluster"
echo "Neptune DB cluster created: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"

# Step 8: Create Neptune DB instance
echo "Creating Neptune DB instance..." | tee -a "$LOG_FILE"
DB_INSTANCE_OUTPUT=$(aws neptune create-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --db-instance-class db.r5.large --engine neptune --db-
cluster-identifier $DB_CLUSTER_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_INSTANCE_OUTPUT" $? "Failed to create Neptune DB instance"
echo "Neptune DB instance created: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"

# Step 9: Wait for the DB instance to become available
echo "Waiting for Neptune DB instance to become available..." | tee -a "$LOG_FILE"
log_cmd "aws neptune wait db-instance-available --db-instance-identifier
 $DB_INSTANCE_ID"

# Step 10: Get the Neptune endpoint
echo "Getting Neptune endpoint..." | tee -a "$LOG_FILE"
ENDPOINT_OUTPUT=$(aws neptune describe-db-clusters --db-cluster-identifier
 $DB_CLUSTER_ID --output json)
```

```
check_error "$ENDPOINT_OUTPUT" $? "Failed to get Neptune endpoint"

NEPTUNE_ENDPOINT=$(echo "$ENDPOINT_OUTPUT" | grep -o '"Endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Neptune endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"

# Step 11: Display information about how to connect to Neptune
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "NEPTUNE SETUP COMPLETE" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Neptune Endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"
echo "Port: 8182" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To query your Neptune database using Gremlin, you can use curl:" | tee -a
 "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.V().limit(1)\"}' https://
$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To add data to your graph:" | tee -a "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.addV(\\\"person\\\").property(\\\"name\\\",
 \\\"Howard\\\")\"}' https://$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Note: You may need to configure your client machine to access the Neptune
 instance within the VPC." | tee -a "$LOG_FILE"
echo "For production use, consider using an EC2 instance in the same VPC or setting
 up a bastion host." | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 12: List all created resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "RESOURCES CREATED" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "VPC: $VPC_ID" | tee -a "$LOG_FILE"
echo "Internet Gateway: $IGW_ID" | tee -a "$LOG_FILE"
echo "Subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"
echo "Route Table: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"
echo "Security Group: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "Neptune DB Cluster: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "Neptune DB Instance: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
```

```
# Step 13: Ask if user wants to clean up resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Do you want to clean up all created resources? (y/n): " | tee -a "$LOG_FILE"
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..." | tee -a "$LOG_FILE"

    # Delete DB instance
    echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-instance --db-instance-identifier $DB_INSTANCE_ID
 --skip-final-snapshot"

    # Wait for DB instance to be deleted
    echo "Waiting for DB instance to be deleted..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"

    # Delete DB cluster
    echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-cluster --db-cluster-identifier $DB_CLUSTER_ID --
skip-final-snapshot"

    # Wait for DB cluster to be deleted (no specific wait command for this, so we'll
 sleep)
    echo "Waiting for DB cluster to be deleted..." | tee -a "$LOG_FILE"
    sleep 60

    # Delete DB subnet group
    echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"

    # Delete security group
    echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"

    # Detach and delete internet gateway
    echo "Detaching and deleting internet gateway $IGW_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 detach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"
```

```
    log_cmd "aws ec2 delete-internet-gateway --internet-gateway-id $IGW_ID"

    # Delete subnets
    echo "Deleting subnets..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET1_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET2_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET3_ID"

    # Delete route table
    echo "Deleting route table $ROUTE_TABLE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-route-table --route-table-id $ROUTE_TABLE_ID"

    # Delete VPC
    echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"

    echo "Cleanup complete!" | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can delete them manually later." |
 tee -a "$LOG_FILE"
    echo "See the list of resources above for reference." | tee -a "$LOG_FILE"
fi

echo "Script completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateInternetGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet

- CreateVpc
- DeleteDbCluster
- DeleteDbInstance
- DeleteDbSubnetGroup
- DeleteInternetGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeDbClusters
- DetachInternetGateway
- ModifyVpcAttribute
- Wait

Introdução ao armazenamento em cache em memória

O exemplo de código a seguir mostra como:

- Configurar grupo de segurança para ElastiCache acesso
- Criar um cache sem servidor do Valkey.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ElastiCache Getting Started Script
```

```
# This script creates a Valkey serverless cache, configures security groups,
# and demonstrates how to connect to and use the cache.

set -uo pipefail

# Set up logging
LOG_FILE="elasticache_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting ElastiCache tutorial script. Logging to $LOG_FILE"
echo "============================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${CACHE_NAME:-}" ]; then
        echo "- ElastiCache serverless cache: $CACHE_NAME"
    fi
    if [ -n "${SG_RULE_6379:-}" ] || [ -n "${SG_RULE_6380:-}" ]; then
        echo "- Security group rules for ports 6379 and 6380"
    fi
    echo "Please clean up these resources manually."
    exit 1
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CACHE_NAME="valkey-cache-${RANDOM_ID}"

echo "Using cache name: $CACHE_NAME"

# Step 1: Set up security group for ElastiCache access
echo "Step 1: Setting up security group for ElastiCache access..."
```

```
# Get default security group ID
echo "Getting default security group ID..."
SG_ID=$(aws ec2 describe-security-groups \
  --filters Name=group-name,Values=default \
  --query "SecurityGroups[0].GroupId" \
  --output text 2>/dev/null || echo "")

if [[ -z "$SG_ID" || "$SG_ID" == "None" ]]; then
    handle_error "Failed to get default security group ID"
fi

echo "Default security group ID: $SG_ID"

# Add inbound rule for port 6379
echo "Adding inbound rule for port 6379..."
SG_RULE_6379=""
if SG_RULE_6379=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6379 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379" ==
 *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    fi
else
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379" ==
 *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    else
        handle_error "Failed to add security group rule for port 6379:
 $SG_RULE_6379"
    fi
fi

# Add inbound rule for port 6380
echo "Adding inbound rule for port 6380..."
SG_RULE_6380=""
if SG_RULE_6380=$(aws ec2 authorize-security-group-ingress \
```

```
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6380 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380" ==
 *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    fi
else
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380" ==
 *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    else
        handle_error "Failed to add security group rule for port 6380:
 $SG_RULE_6380"
    fi
fi

echo "Security group rules added successfully."
echo ""
echo "##  SECURITY WARNING: The security group rules created allow access from any
 IP address (0.0.0.0/0)."
echo "This is NOT RECOMMENDED for production environments. For production,"
echo "you should restrict access to specific IP ranges or security groups."
echo "Update the CIDR blocks in this script before using in production."
echo ""

# Step 2: Create a Valkey serverless cache
echo "Step 2: Creating Valkey serverless cache..."
if ! CREATE_RESULT=$(aws elasticache create-serverless-cache \
  --serverless-cache-name "$CACHE_NAME" \
  --engine valkey \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-elasticache-gs 2>&1);
 then
    handle_error "Failed to create serverless cache: $CREATE_RESULT"
fi

echo "Cache creation initiated. Waiting for cache to become available..."

# Step 3: Check the status of the cache creation
```

```
echo "Step 3: Checking cache status..."

# Wait for the cache to become active
MAX_ATTEMPTS=30
ATTEMPT=1
CACHE_STATUS=""

while [[ $ATTEMPT -le $MAX_ATTEMPTS ]]; do
    echo "Checking cache status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    if ! DESCRIBE_RESULT=$(aws elasticache describe-serverless-caches \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        handle_error "Failed to describe serverless cache: $DESCRIBE_RESULT"
    fi

    # Extract status using jq for reliable JSON parsing
    if command -v jq &> /dev/null; then
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | jq -r
 '.ServerlessCaches[0].Status // "UNKNOWN"' 2>/dev/null || echo "")
    else
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | grep -o '"Status": "[^"]*"' | awk -
F'"' '{print $4}' | head -n 1)
    fi

    echo "Current status: $CACHE_STATUS"

    if [[ "${CACHE_STATUS,,}" == "available" ]]; then
        echo "Cache is now available!"
        break
    elif [[ "${CACHE_STATUS,,}" == "create-failed" ]]; then
        handle_error "Cache creation failed. Please check the AWS console for
 details."
    fi

    if [[ $ATTEMPT -lt $MAX_ATTEMPTS ]]; then
        echo "Waiting 30 seconds..."
        sleep 30
    fi

    ((ATTEMPT++))
done

if [[ "${CACHE_STATUS,,}" != "available" ]]; then
```

```
    handle_error "Cache did not become available within the expected time. Last
 status: $CACHE_STATUS"
fi

# Step 4: Find your cache endpoint
echo "Step 4: Getting cache endpoint..."
if ! ENDPOINT=$(aws elasticache describe-serverless-caches \
  --serverless-cache-name "$CACHE_NAME" \
  --query "ServerlessCaches[0].Endpoint.Address" \
  --output text 2>&1); then
    handle_error "Failed to get cache endpoint: $ENDPOINT"
fi

if [[ -z "$ENDPOINT" || "$ENDPOINT" == "None" ]]; then
    handle_error "Failed to get cache endpoint"
fi

echo "Cache endpoint: $ENDPOINT"

# Step 5: Instructions for connecting to the cache
echo ""
echo "============================================================"
echo "Your Valkey serverless cache has been successfully created!"
echo "Cache Name: $CACHE_NAME"
echo "Endpoint: $ENDPOINT"
echo "============================================================"
echo ""
echo "To connect to your cache from an EC2 instance, follow these steps:"
echo ""
echo "1. Install valkey-cli on your EC2 instance:"
echo "   sudo amazon-linux-extras install epel -y"
echo "   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y"
echo "   wget https://github.com/valkey-io/valkey/archive/refs/tags/8.0.0.tar.gz"
echo "   tar xvzf 8.0.0.tar.gz"
echo "   cd valkey-8.0.0"
echo "   make BUILD_TLS=yes"
echo ""
echo "2. Connect to your cache using valkey-cli:"
echo "   src/valkey-cli -h $ENDPOINT --tls -p 6379"
echo ""
echo "3. Once connected, you can run commands like:"
echo "   set mykey \"Hello ElastiCache\""
echo "   get mykey"
echo ""
```

```
# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- ElastiCache serverless cache: $CACHE_NAME"
if [[ "${SG_RULE_6379:-}" != "existing" ]] || [[ "${SG_RULE_6380:-}" !=
 "existing" ]]; then
    echo "- Security group rules for ports 6379 and 6380"
fi
echo ""
echo "Proceeding with cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Step 7: Delete the cache
    echo "Deleting serverless cache $CACHE_NAME..."
    if ! DELETE_RESULT=$(aws elasticache delete-serverless-cache \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        echo "WARNING: Failed to delete serverless cache: $DELETE_RESULT"
        echo "Please delete the cache manually from the AWS console."
    else
        echo "Cache deletion initiated. This may take several minutes to complete."
    fi

    # Only attempt to remove security group rules if we created them
    if [[ "${SG_RULE_6379:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6379..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6379 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6379"
        fi
    fi

    if [[ "${SG_RULE_6380:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6380..."
```

```
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6380 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6380"
        fi
    fi

    echo "Cleanup completed."
fi

echo ""
echo "Script completed. See $LOG_FILE for the full log."
echo "============================================================"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateServerlessCache
  - DeleteServerlessCache
  - DescribeSecurityGroups
  - DescribeServerlessCaches
  - RevokeSecurityGroupIngress

Introdução ao balanceamento de carga

O exemplo de código a seguir mostra como:

- Criar um Application Load Balancer.
- Criar um grupo de destino.
- Criar um receptor.
- Verificar sua configuração.
- Adicionar um receptor HTTPS (opcional).
- Adicionar roteamento baseado em caminho (opcional).
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Elastic Load Balancing Getting Started Script - v2
# This script creates an Application Load Balancer with HTTP listener and target
 group

set -euo pipefail

# Set up logging
LOG_FILE="elb-script-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Elastic Load Balancing setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check AWS CLI command success
check_command() {
    local output="$1"
    if [[ -z "$output" ]] || [[ "$output" == "None" ]]; then
        handle_error "AWS CLI command returned empty or invalid output"
    fi
}

# Function to validate ARN format
validate_arn() {
    local arn="$1"
```

```
    if [[ ! "$arn" =~ ^arn:aws:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$ ]]; then
        handle_error "Invalid ARN format: $arn"
    fi
}

# Function to validate security group ID
validate_security_group_id() {
    local sg_id="$1"
    if [[ ! "$sg_id" =~ ^sg-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid security group ID format: $sg_id"
    fi
}

# Function to validate VPC ID
validate_vpc_id() {
    local vpc_id="$1"
    if [[ ! "$vpc_id" =~ ^vpc-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid VPC ID format: $vpc_id"
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    if [ -n "${LISTENER_ARN:-}" ]; then
        echo "Deleting listener: $LISTENER_ARN"
        aws elbv2 delete-listener --listener-arn "$LISTENER_ARN" 2>/dev/null || true
    fi

    if [ -n "${LOAD_BALANCER_ARN:-}" ]; then
        echo "Deleting load balancer: $LOAD_BALANCER_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LOAD_BALANCER_ARN" 2>/
dev/null || true

        # Wait for load balancer to be deleted before deleting target group
        echo "Waiting for load balancer to be deleted..."
        aws elbv2 wait load-balancers-deleted --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null || true
    fi

    if [ -n "${TARGET_GROUP_ARN:-}" ]; then
        echo "Deleting target group: $TARGET_GROUP_ARN"
```

```
        aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/
null || true
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting 30 seconds before deleting security group to ensure all
 dependencies are removed..."
        sleep 30

        echo "Deleting security group: $SECURITY_GROUP_ID"
        local sg_delete_output
        sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)

        local retry_count=0
        local max_retries=5
        while echo "$sg_delete_output" | grep -i "DependencyViolation" > /dev/null
 && [ $retry_count -lt $max_retries ]; do
            retry_count=$((retry_count+1))
            echo "Security group still has dependencies. Retrying in 30 seconds...
 (Attempt $retry_count of $max_retries)"
            sleep 30
            sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)
        done

        if echo "$sg_delete_output" | grep -i "error" > /dev/null; then
            echo "WARNING: Could not delete security group: $SECURITY_GROUP_ID" >&2
            echo "You may need to delete it manually using: aws ec2 delete-security-
group --group-id $SECURITY_GROUP_ID"
        else
            echo "Security group deleted successfully."
        fi
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
RESOURCE_PREFIX="elb-demo-${RANDOM_ID}"

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi
```

```
# Step 1: Verify AWS CLI support for Elastic Load Balancing
echo "Verifying AWS CLI support for Elastic Load Balancing..."
if ! aws elbv2 help > /dev/null 2>&1; then
    handle_error "AWS CLI does not support elbv2 commands. Please update your AWS
 CLI."
fi

# Step 2: Get VPC ID and subnet information
echo "Retrieving VPC information..."
VPC_INFO=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 "Vpcs[0].VpcId" --output text 2>/dev/null || echo "")
check_command "$VPC_INFO"
VPC_ID="$VPC_INFO"
validate_vpc_id "$VPC_ID"
echo "Using VPC: $VPC_ID"

# Get two subnets from different Availability Zones
echo "Retrieving subnet information..."
SUBNET_INFO=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[0:2].SubnetId" --output text 2>/dev/null || echo "")
check_command "$SUBNET_INFO"

# Convert space-separated list to array
read -r -a SUBNETS <<< "$SUBNET_INFO"
if [ ${#SUBNETS[@]} -lt 2 ]; then
    handle_error "Need at least 2 subnets in different Availability Zones. Found:
 ${#SUBNETS[@]}"
fi

echo "Using subnets: ${SUBNETS[0]} and ${SUBNETS[1]}"

# Step 3: Create a security group for the load balancer
echo "Creating security group for the load balancer..."
SG_INFO=$(aws ec2 create-security-group \
    --group-name "${RESOURCE_PREFIX}-sg" \
    --description "Security group for ELB demo" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=elastic-load-balancing-gs}]' \
    --query "GroupId" --output text 2>/dev/null || echo "")
check_command "$SG_INFO"
SECURITY_GROUP_ID="$SG_INFO"
validate_security_group_id "$SECURITY_GROUP_ID"
```

```
echo "Created security group: $SECURITY_GROUP_ID"

# Add inbound rule to allow HTTP traffic
echo "Adding inbound rule to allow HTTP traffic..."
aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 80 \
    --cidr "0.0.0.0/0" > /dev/null 2>&1 || handle_error "Failed to authorize
 security group ingress"

echo "WARNING: Security group allows HTTP from 0.0.0.0/0. In production, restrict to
 specific IP addresses." >&2

# Step 4: Create the load balancer
echo "Creating Application Load Balancer..."
LB_INFO=$(aws elbv2 create-load-balancer \
    --name "${RESOURCE_PREFIX}-lb" \
    --subnets "${SUBNETS[0]}" "${SUBNETS[1]}" \
    --security-groups "$SECURITY_GROUP_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "LoadBalancers[0].LoadBalancerArn" --output text 2>/dev/null || echo "")
check_command "$LB_INFO"
LOAD_BALANCER_ARN="$LB_INFO"
validate_arn "$LOAD_BALANCER_ARN"
echo "Created load balancer: $LOAD_BALANCER_ARN"

# Wait for the load balancer to be active
echo "Waiting for load balancer to become active..."
if ! aws elbv2 wait load-balancer-available --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null; then
    handle_error "Load balancer did not reach active state within timeout period"
fi

# Step 5: Create a target group
echo "Creating target group..."
TG_INFO=$(aws elbv2 create-target-group \
    --name "${RESOURCE_PREFIX}-targets" \
    --protocol HTTP \
    --port 80 \
    --vpc-id "$VPC_ID" \
    --target-type instance \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "TargetGroups[0].TargetGroupArn" --output text 2>/dev/null || echo "")
check_command "$TG_INFO"
TARGET_GROUP_ARN="$TG_INFO"
validate_arn "$TARGET_GROUP_ARN"
echo "Created target group: $TARGET_GROUP_ARN"

# Step 6: Find EC2 instances to register as targets
echo "Looking for available EC2 instances to register as targets..."
INSTANCES=$(aws ec2 describe-instances \
    --filters "Name=vpc-id,Values=$VPC_ID" "Name=instance-state-name,Values=running"
 \
    --query "Reservations[*].Instances[*].InstanceId" --output text 2>/dev/null ||
 echo "")

# Convert space-separated list to array
read -r -a INSTANCE_IDS <<< "$INSTANCES"

if [ ${#INSTANCE_IDS[@]} -eq 0 ]; then
    echo "No running instances found in VPC $VPC_ID."
    echo "You will need to register targets manually after launching instances."
else
    # Step 7: Register targets with the target group (up to 2 instances)
    echo "Registering targets with the target group..."
    target_args=()
    for i in "${!INSTANCE_IDS[@]}"; do
        if [ "$i" -lt 2 ]; then
            target_args+=("Id=${INSTANCE_IDS[$i]}")
        fi
    done

    if [ ${#target_args[@]} -gt 0 ]; then
        if aws elbv2 register-targets \
            --target-group-arn "$TARGET_GROUP_ARN" \
            --targets "${target_args[@]}" 2>/dev/null; then
            echo "Registered instances: ${target_args[*]}"
        else
            handle_error "Failed to register targets"
        fi
    fi
fi

# Step 8: Create a listener
```

```
echo "Creating HTTP listener..."
LISTENER_INFO=$(aws elbv2 create-listener \
    --load-balancer-arn "$LOAD_BALANCER_ARN" \
    --protocol HTTP \
    --port 80 \
    --default-actions Type=forward,TargetGroupArn="$TARGET_GROUP_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "Listeners[0].ListenerArn" --output text 2>/dev/null || echo "")
check_command "$LISTENER_INFO"
LISTENER_ARN="$LISTENER_INFO"
validate_arn "$LISTENER_ARN"
echo "Created listener: $LISTENER_ARN"

# Step 9: Verify target health
echo "Verifying target health..."
aws elbv2 describe-target-health --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/null
 || true

# Display load balancer DNS name
LB_DNS=$(aws elbv2 describe-load-balancers \
    --load-balancer-arns "$LOAD_BALANCER_ARN" \
    --query "LoadBalancers[0].DNSName" --output text 2>/dev/null || echo "")
check_command "$LB_DNS"

echo ""
echo "================================================"
echo "SETUP COMPLETE"
echo "================================================"
echo "Load Balancer DNS Name: $LB_DNS"
echo ""
echo "Resources created:"
echo "- Load Balancer: $LOAD_BALANCER_ARN"
echo "- Target Group: $TARGET_GROUP_ARN"
echo "- Listener: $LISTENER_ARN"
echo "- Security Group: $SECURITY_GROUP_ID"
echo ""

# Prompt for cleanup confirmation
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
read -p "Do you want to clean up all created resources? (y/n): " -r CLEANUP_CHOICE
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
    echo "To clean up later, run the following commands:"
    echo "aws elbv2 delete-listener --listener-arn $LISTENER_ARN"
    echo "aws elbv2 delete-load-balancer --load-balancer-arn $LOAD_BALANCER_ARN"
    echo "aws elbv2 wait load-balancers-deleted --load-balancer-arns
 $LOAD_BALANCER_ARN"
    echo "aws elbv2 delete-target-group --target-group-arn $TARGET_GROUP_ARN"
    echo "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateListener
  - CreateLoadBalancer
  - CreateSecurityGroup
  - CreateTargetGroup
  - DeleteListener
  - DeleteLoadBalancer
  - DeleteSecurityGroup
  - DeleteTargetGroup
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetHealth
  - DescribeVpcs
  - Help
  - RegisterTargets
  - Wait

Conceitos básicos de streaming gerenciado

O exemplo de código a seguir mostra como:

- Criar um cluster do Amazon MSK.
- Criar permissões do IAM para acessar o MSK.
- Criar uma máquina cliente
- Obter os agentes de bootstrap.
- Configurar a máquina cliente.
- Crie um tópico e produce/consume dados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
```

```
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group: $CLIENT_SG_ID";
 fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id" &>/
dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated" &>/
dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
```

```
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o "\"GroupId
\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing $sg_id,
 removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
```

```
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing $sg_id,
 removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No outbound
 all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
```

```
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed to
 terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group" "$CLIENT_SG_ID";
 then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
 seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
```

```
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete key
pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to delete
policy"
```

```
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az (subnet:
 $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
```

```
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "================================================"

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default VPC
 in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
```

```
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-az,Values=true"
 \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least 3
 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=$DEFAULT_VPC_ID"
 \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\", \"ClientSubnets
\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\", \"${SUBNET_ARRAY[2]}\"],
 \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-gs
 2>&1)

# Check if the command was successful
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --query
 "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"
```

```
# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
```

```
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi
```

```
echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@]; then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"
```

```
# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
```

```
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
```

```
echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-${RANDOM_SUFFIX}},
{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-streaming-for-
apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
```

```
    handle_error "Failed to extract instance ID from response: $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""
```

```
while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
```

```
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-
auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
```

```
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/$LATEST_VERSION/
aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
```

```
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
```

```
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "================================================"
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
```

```
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "==============================================="
echo ""

# Ask if user wants to clean up resources
```

```
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair
  - DeletePolicy
  - DeleteRole
  - DeleteSecurityGroup
  - DescribeAvailabilityZones

- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Introdução aos gateways de trânsito de rede

O exemplo de código a seguir mostra como:

- Criar um gateway de trânsito.
- Anexar as VPCs ao gateway de trânsito.
- Adicionar rotas entre o gateway de trânsito e as VPCs.
- Testar o gateway de trânsito.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon VPC Transit Gateway CLI Script
# This script demonstrates how to create a transit gateway and connect two VPCs
# Modified to work with older AWS CLI versions that don't support transit gateway
 wait commands
# Security improved: Added input validation, error handling, and credential checks

# Error handling
set -euo pipefail
LOG_FILE="transit-gateway-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Security: Check for required AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
  echo "ERROR: AWS credentials not configured or invalid. Please configure AWS
 credentials."
  exit 1
fi

# Function to validate AWS CLI output
validate_aws_output() {
  local output=$1
  local context=$2

  if [ -z "$output" ] || [ "$output" = "None" ]; then
    echo "ERROR: Failed to retrieve $context from AWS API"
    return 1
  fi
}

# Function to validate CIDR blocks
validate_cidr() {
  local cidr=$1
```

```
  if ! [[ "$cidr" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}$ ]]; then
    echo "ERROR: Invalid CIDR block format: $cidr"
    return 1
  fi
}

# Function to wait for transit gateway to be available
wait_for_tgw() {
  local tgw_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway $tgw_id to become available..."

  while [ $attempt -lt $max_attempts ]; do
    status=$(aws ec2 describe-transit-gateways \
      --transit-gateway-ids "$tgw_id" \
      --query "TransitGateways[0].State" \
      --output text 2>/dev/null || echo "failed")

    echo "Current status: $status"

    if [ "$status" = "available" ]; then
      echo "Transit Gateway is now available"
      return 0
    fi

    if [ "$status" = "failed" ]; then
      echo "ERROR: Transit Gateway creation failed"
      return 1
    fi

    echo "Waiting for transit gateway to become available. Current state: $status"
    sleep 10
    ((attempt++))
  done

  echo "ERROR: Timeout waiting for transit gateway to become available"
  return 1
}

# Function to wait for transit gateway attachment to be available
wait_for_tgw_attachment() {
  local attachment_id=$1
```

```
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway Attachment $attachment_id to become
 available..."

  while [ $attempt -lt $max_attempts ]; do
    status=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --transit-gateway-attachment-ids "$attachment_id" \
      --query "TransitGatewayVpcAttachments[0].State" \
      --output text 2>/dev/null || echo "failed")

    echo "Current status: $status"

    if [ "$status" = "available" ]; then
      echo "Transit Gateway Attachment is now available"
      return 0
    fi

    if [ "$status" = "failed" ]; then
      echo "ERROR: Transit Gateway Attachment creation failed"
      return 1
    fi

    echo "Waiting for transit gateway attachment to become available. Current state:
 $status"
    sleep 10
    ((attempt++))
  done

  echo "ERROR: Timeout waiting for transit gateway attachment to become available"
  return 1
}

# Function to wait for transit gateway attachment to be deleted
wait_for_tgw_attachment_deleted() {
  local attachment_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway Attachment $attachment_id to be deleted..."

  while [ $attempt -lt $max_attempts ]; do
    count=$(aws ec2 describe-transit-gateway-vpc-attachments \
```

```
      --filters "Name=transit-gateway-attachment-id,Values=$attachment_id" \
      --query "length(TransitGatewayVpcAttachments)" \
      --output text 2>/dev/null || echo "0")

    if [ "$count" = "0" ]; then
      echo "Transit Gateway Attachment has been deleted"
      return 0
    fi

    status=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --transit-gateway-attachment-ids "$attachment_id" \
      --query "TransitGatewayVpcAttachments[0].State" \
      --output text 2>/dev/null || echo "deleted")

    if [ "$status" = "deleted" ] || [ "$status" = "deleting" ]; then
      echo "Transit Gateway Attachment is being deleted. Current state: $status"
    fi

    echo "Waiting for transit gateway attachment to be deleted. Current state:
 $status"
    sleep 10
    ((attempt++))
  done

  echo "WARNING: Timeout waiting for transit gateway attachment to be deleted"
  return 0
}

# Function to clean up resources
cleanup() {
  local exit_code=$?
  echo "Error occurred (exit code: $exit_code). Cleaning up resources..."

  # Delete resources in reverse order
  if [ -n "${TGW_ATTACHMENT_1_ID:-}" ]; then
    echo "Deleting Transit Gateway VPC Attachment 1: $TGW_ATTACHMENT_1_ID"
    aws ec2 delete-transit-gateway-vpc-attachment \
      --transit-gateway-attachment-id "$TGW_ATTACHMENT_1_ID" &>/dev/null || true
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
  fi

  if [ -n "${TGW_ATTACHMENT_2_ID:-}" ]; then
    echo "Deleting Transit Gateway VPC Attachment 2: $TGW_ATTACHMENT_2_ID"
    aws ec2 delete-transit-gateway-vpc-attachment \
```

```
      --transit-gateway-attachment-id "$TGW_ATTACHMENT_2_ID" &>/dev/null || true
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true
  fi

  if [ -n "${TGW_ID:-}" ]; then
    echo "Deleting Transit Gateway: $TGW_ID"
    aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" &>/dev/null ||
 true
  fi

  exit "$exit_code"
}

# Set up trap for error handling
trap cleanup EXIT

echo "=== Amazon VPC Transit Gateway Tutorial ==="
echo "This script will create a transit gateway and connect two VPCs"
echo ""

# Get a valid availability zone dynamically
echo "Getting available AZ in current region..."
AZ=$(aws ec2 describe-availability-zones \
  --query "AvailabilityZones[0].ZoneName" \
  --output text)
validate_aws_output "$AZ" "availability zone" || exit 1
echo "Using availability zone: $AZ"

# Check if VPCs exist
echo "Checking for existing VPCs..."
VPC1_ID=$(aws ec2 describe-vpcs \
  --filters "Name=tag:Name,Values=VPC1" \
  --query "Vpcs[0].VpcId" \
  --output text)
VPC2_ID=$(aws ec2 describe-vpcs \
  --filters "Name=tag:Name,Values=VPC2" \
  --query "Vpcs[0].VpcId" \
  --output text)

if [ "$VPC1_ID" = "None" ] || [ -z "$VPC1_ID" ]; then
  echo "Creating VPC1..."
  VPC1_ID=$(aws ec2 create-vpc \
    --cidr-block 10.1.0.0/16 \
```

```
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
    --query Vpc.VpcId \
    --output text)
  validate_aws_output "$VPC1_ID" "VPC1" || exit 1
  echo "Created VPC1: $VPC1_ID"

  # Create a subnet in VPC1
  echo "Creating subnet in VPC1..."
  SUBNET1_ID=$(aws ec2 create-subnet \
    --vpc-id "$VPC1_ID" \
    --cidr-block 10.1.0.0/24 \
    --availability-zone "$AZ" \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
    --query Subnet.SubnetId \
    --output text)
  validate_aws_output "$SUBNET1_ID" "VPC1 subnet" || exit 1
  echo "Created subnet in VPC1: $SUBNET1_ID"
else
  echo "Using existing VPC1: $VPC1_ID"
  SUBNET1_ID=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$VPC1_ID" \
    --query "Subnets[0].SubnetId" \
    --output text)
  if [ "$SUBNET1_ID" = "None" ] || [ -z "$SUBNET1_ID" ]; then
    echo "Creating subnet in VPC1..."
    SUBNET1_ID=$(aws ec2 create-subnet \
      --vpc-id "$VPC1_ID" \
      --cidr-block 10.1.0.0/24 \
      --availability-zone "$AZ" \
      --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
      --query Subnet.SubnetId \
      --output text)
    validate_aws_output "$SUBNET1_ID" "VPC1 subnet" || exit 1
    echo "Created subnet in VPC1: $SUBNET1_ID"
  else
    echo "Using existing subnet in VPC1: $SUBNET1_ID"
  fi
fi

if [ "$VPC2_ID" = "None" ] || [ -z "$VPC2_ID" ]; then
  echo "Creating VPC2..."
```

```
  VPC2_ID=$(aws ec2 create-vpc \
    --cidr-block 10.2.0.0/16 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
    --query Vpc.VpcId \
    --output text)
  validate_aws_output "$VPC2_ID" "VPC2" || exit 1
  echo "Created VPC2: $VPC2_ID"

  # Create a subnet in VPC2
  echo "Creating subnet in VPC2..."
  SUBNET2_ID=$(aws ec2 create-subnet \
    --vpc-id "$VPC2_ID" \
    --cidr-block 10.2.0.0/24 \
    --availability-zone "$AZ" \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
    --query Subnet.SubnetId \
    --output text)
  validate_aws_output "$SUBNET2_ID" "VPC2 subnet" || exit 1
  echo "Created subnet in VPC2: $SUBNET2_ID"
else
  echo "Using existing VPC2: $VPC2_ID"
  SUBNET2_ID=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$VPC2_ID" \
    --query "Subnets[0].SubnetId" \
    --output text)
  if [ "$SUBNET2_ID" = "None" ] || [ -z "$SUBNET2_ID" ]; then
    echo "Creating subnet in VPC2..."
    SUBNET2_ID=$(aws ec2 create-subnet \
      --vpc-id "$VPC2_ID" \
      --cidr-block 10.2.0.0/24 \
      --availability-zone "$AZ" \
      --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
      --query Subnet.SubnetId \
      --output text)
    validate_aws_output "$SUBNET2_ID" "VPC2 subnet" || exit 1
    echo "Created subnet in VPC2: $SUBNET2_ID"
  else
    echo "Using existing subnet in VPC2: $SUBNET2_ID"
  fi
fi
```

```
# Get route tables for each VPC
RTB1_ID=$(aws ec2 describe-route-tables \
  --filters "Name=vpc-id,Values=$VPC1_ID" \
  --query "RouteTables[0].RouteTableId" \
  --output text)
validate_aws_output "$RTB1_ID" "VPC1 route table" || exit 1

RTB2_ID=$(aws ec2 describe-route-tables \
  --filters "Name=vpc-id,Values=$VPC2_ID" \
  --query "RouteTables[0].RouteTableId" \
  --output text)
validate_aws_output "$RTB2_ID" "VPC2 route table" || exit 1

echo "Route table for VPC1: $RTB1_ID"
echo "Route table for VPC2: $RTB2_ID"

# Step 1: Create the transit gateway
echo "Creating Transit Gateway..."
TGW_ID=$(aws ec2 create-transit-gateway \
  --description "My Transit Gateway" \
  --options
 "AmazonSideAsn=64512,AutoAcceptSharedAttachments=disable,DefaultRouteTableAssociation=enabl
 \
  --tag-specifications 'ResourceType=transit-
gateway,Tags=[{Key=Name,Value=MyTransitGateway},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGateway.TransitGatewayId \
  --output text)
validate_aws_output "$TGW_ID" "Transit Gateway" || exit 1
echo "Created Transit Gateway: $TGW_ID"

# Wait for the transit gateway to become available
wait_for_tgw "$TGW_ID" || exit 1

# Step 2: Attach VPCs to the transit gateway
echo "Attaching VPC1 to Transit Gateway..."
TGW_ATTACHMENT_1_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC1_ID" \
  --subnet-ids "$SUBNET1_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC1-Attachment},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
```

```
  --output text)
validate_aws_output "$TGW_ATTACHMENT_1_ID" "Transit Gateway VPC1 Attachment" || exit
 1
echo "Created Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"

echo "Attaching VPC2 to Transit Gateway..."
TGW_ATTACHMENT_2_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC2_ID" \
  --subnet-ids "$SUBNET2_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC2-Attachment},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)
validate_aws_output "$TGW_ATTACHMENT_2_ID" "Transit Gateway VPC2 Attachment" || exit
 1
echo "Created Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"

# Wait for the attachments to become available
wait_for_tgw_attachment "$TGW_ATTACHMENT_1_ID" || exit 1
wait_for_tgw_attachment "$TGW_ATTACHMENT_2_ID" || exit 1

# Step 3: Add routes between the transit gateway and VPCs
echo "Adding route from VPC1 to VPC2 via Transit Gateway..."
validate_cidr "10.2.0.0/16" || exit 1
aws ec2 create-route \
  --route-table-id "$RTB1_ID" \
  --destination-cidr-block 10.2.0.0/16 \
  --transit-gateway-id "$TGW_ID" || exit 1

echo "Adding route from VPC2 to VPC1 via Transit Gateway..."
validate_cidr "10.1.0.0/16" || exit 1
aws ec2 create-route \
  --route-table-id "$RTB2_ID" \
  --destination-cidr-block 10.1.0.0/16 \
  --transit-gateway-id "$TGW_ID" || exit 1

echo "Routes added successfully"

# Step 4: Display information for testing
echo ""
echo "=== Transit Gateway Setup Complete ==="
echo "Transit Gateway ID: $TGW_ID"
```

```
echo "VPC1 ID: $VPC1_ID"
echo "VPC2 ID: $VPC2_ID"
echo ""
echo "To test connectivity:"
echo "1. Launch an EC2 instance in each VPC"
echo "2. Configure security groups to allow ICMP traffic"
echo "3. Connect to one instance and ping the other instance's private IP"
echo ""

echo "Viewing created resources..."

echo ""
echo "=== Resources Created ===="
echo "Transit Gateway: $TGW_ID"
echo "VPC1: $VPC1_ID"
echo "VPC2: $VPC2_ID"
echo "Subnet in VPC1: $SUBNET1_ID"
echo "Subnet in VPC2: $SUBNET2_ID"
echo "Transit Gateway Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
echo "Transit Gateway Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
echo ""

echo "Starting cleanup..."

# Delete routes
echo "Deleting routes..."
aws ec2 delete-route --route-table-id "$RTB1_ID" --destination-cidr-block
 10.2.0.0/16 || true
aws ec2 delete-route --route-table-id "$RTB2_ID" --destination-cidr-block
 10.1.0.0/16 || true

# Delete transit gateway attachments
echo "Deleting Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
aws ec2 delete-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id "$TGW_ATTACHMENT_1_ID" || true

echo "Deleting Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
aws ec2 delete-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id "$TGW_ATTACHMENT_2_ID" || true

# Wait for attachments to be deleted
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true
```

```
# Delete transit gateway
echo "Deleting Transit Gateway: $TGW_ID"
aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" || true

echo "Cleanup completed successfully"

echo "Tutorial completed. See $LOG_FILE for detailed logs."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateRoute
  - CreateSubnet
  - CreateTransitGateway
  - CreateTransitGatewayVpcAttachment
  - CreateVpc
  - DeleteRoute
  - DeleteTransitGateway
  - DeleteTransitGatewayVpcAttachment
  - DescribeAvailabilityZones
  - DescribeRouteTables
  - DescribeSubnets
  - DescribeTransitGatewayAttachments
  - DescribeTransitGatewayVpcAttachments
  - DescribeTransitGateways
  - DescribeVpcs

Introdução à transmissão de transporte de vídeo

O exemplo de código a seguir mostra como:

- Verifique o acesso ao Elemental MediaConnect
- Criar um fluxo.
- Adicionar uma saída.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Elemental MediaConnect Getting Started Tutorial Script
# This script creates a MediaConnect flow, adds an output, grants an entitlement,
# and then cleans up the resources.

set -euo pipefail

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="mediaconnect-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Elemental MediaConnect tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Tags for all resources
TAGS_ARRAY=("Key=project,Value=doc-smith" "Key=tutorial,Value=aws-elemental-
mediaconnect-gs")

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI is available
validate_aws_cli() {
```

```
    if ! command -v aws &> /dev/null; then
        handle_error "AWS CLI is not installed or not in PATH"
    fi

    # Security: Verify AWS CLI version is recent
    local aws_version
    aws_version=$(aws --version 2>&1 | head -1)
    echo "AWS CLI version: $aws_version"

    if ! aws sts get-caller-identity &> /dev/null; then
        handle_error "AWS credentials are not configured or invalid"
    fi

    # Security: Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query Account --output text 2>/dev/
null)
    if [ -z "$account_id" ]; then
        handle_error "Failed to retrieve AWS account ID"
    fi
    echo "AWS Account ID: $account_id"
}

# Function to safely extract JSON values using jq (preferred) or fallback
extract_json_value() {
    local json_output="$1"
    local key="$2"

    if [ -z "$json_output" ]; then
        return 1
    fi

    # Security: Use jq if available for safer JSON parsing
    if command -v jq &> /dev/null; then
        echo "$json_output" | jq -r ".${key} // empty" 2>/dev/null || return 1
    else
        # Fallback with additional validation
        if ! echo "$json_output" | grep -q "\"$key\""; then
            return 1
        fi
        echo "$json_output" | grep -o "\"$key\": \"[^\"]*" | head -1 | cut -d'"' -f4
    fi
}
```

```
# Function to tag MediaConnect resource
tag_mediaconnect_resource() {
    local resource_arn="$1"
    echo "Tagging resource: $resource_arn"

    for tag in "${TAGS_ARRAY[@]}"; do
        if ! aws mediaconnect tag-resource --resource-arn "$resource_arn" --tags
 "$tag" 2>&1; then
            echo "WARNING: Failed to apply tag $tag to resource"
        fi
    done
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "${FLOW_ARN:-}" ]; then
        # Security: Validate ARN format before using it
        if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-
Z0-9:-]+$ ]]; then
            echo "WARNING: Invalid Flow ARN format, skipping cleanup: $FLOW_ARN"
            return 1
        fi

        # Check flow status before attempting to stop
        echo "Checking flow status..."
        local flow_status_output
        if flow_status_output=$(aws mediaconnect describe-flow --flow-arn
 "$FLOW_ARN" --query "Flow.Status" --output text 2>&1); then
            echo "Current flow status: $flow_status_output"

            if [ "$flow_status_output" == "ACTIVE" ] || [ "$flow_status_output" ==
 "UPDATING" ]; then
                echo "Stopping flow: $FLOW_ARN"
                if aws mediaconnect stop-flow --flow-arn "$FLOW_ARN" 2>&1; then
                    # Wait for flow to stop before deleting
                    echo "Waiting for flow to stop..."
                    sleep 10
                else
                    echo "WARNING: Failed to stop flow. Attempting to delete
 anyway..."
                fi
            else
```

```
                echo "Flow is not in ACTIVE or UPDATING state, skipping stop
 operation."
            fi

            # Delete the flow
            echo "Deleting flow: $FLOW_ARN"
            if aws mediaconnect delete-flow --flow-arn "$FLOW_ARN" 2>&1; then
                echo "Flow deleted successfully"
            else
                echo "WARNING: Failed to delete flow. You may need to manually
 delete it from the AWS console."
            fi
        else
            echo "WARNING: Could not check flow status"
        fi
    fi
}

# Set trap to cleanup on script exit
trap cleanup_resources EXIT

# Validate AWS CLI setup
validate_aws_cli

# Get the current AWS region
aws_region=""
if aws_region=$(aws configure get region 2>/dev/null); then
    if [ -z "$aws_region" ]; then
        handle_error "Failed to get AWS region. Please make sure AWS CLI is
 configured."
    fi
else
    handle_error "Failed to retrieve AWS region configuration"
fi

# Security: Validate region format
if [[ ! "$aws_region" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $aws_region"
fi

AWS_REGION="$aws_region"
echo "Using AWS Region: $AWS_REGION"

# Get available availability zones in the current region
```

```
echo "Getting available availability zones in region $AWS_REGION..."
az_output=""
if az_output=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --query
 "AvailabilityZones[0].ZoneName" --output text 2>&1); then
    AVAILABILITY_ZONE="$az_output"
    if [ -z "$AVAILABILITY_ZONE" ]; then
        handle_error "Failed to retrieve availability zones"
    fi

    # Security: Validate AZ format
    if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9][a-z]$ ]]; then
        handle_error "Invalid availability zone format: $AVAILABILITY_ZONE"
    fi

    echo "Using availability zone: $AVAILABILITY_ZONE"
else
    handle_error "Failed to get availability zones"
fi

# Generate a unique suffix for resource names
SUFFIX=$(date +%s | cut -c 6-10)
FLOW_NAME="AwardsNYCShow-${SUFFIX}"
SOURCE_NAME="AwardsNYCSource-${SUFFIX}"
OUTPUT_NAME="AwardsNYCOutput-${SUFFIX}"
ENTITLEMENT_NAME="PhillyTeam-${SUFFIX}"

echo "Using the following resource names:"
echo "Flow name: $FLOW_NAME"
echo "Source name: $SOURCE_NAME"
echo "Output name: $OUTPUT_NAME"
echo "Entitlement name: $ENTITLEMENT_NAME"

# Step 1: Verify access to MediaConnect
echo "Step 1: Verifying access to AWS Elemental MediaConnect..."
list_flows_output=""
if list_flows_output=$(aws mediaconnect list-flows 2>&1); then
    echo "$list_flows_output"
else
    handle_error "Failed to list flows. Please check your AWS credentials and
 permissions."
fi

# Step 2: Create a flow
echo "Step 2: Creating a flow..."
```

```
create_flow_output=""
if create_flow_output=$(aws mediaconnect create-flow \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "$FLOW_NAME" \
    --source "Name=$SOURCE_NAME,Protocol=zixi-
push,WhitelistCidr=10.24.34.0/23,StreamId=ZixiAwardsNYCFeed" 2>&1); then
    echo "$create_flow_output"
else
    handle_error "Failed to create flow"
fi

# Extract the flow ARN from the output
FLOW_ARN=$(echo "$create_flow_output" | jq -r '.Flow.FlowArn // empty' 2>/dev/null)
if [ -z "$FLOW_ARN" ]; then
    FLOW_ARN=$(echo "$create_flow_output" | grep -o '"FlowArn": "[^"]*' | head -1 |
 cut -d'"' -f4)
fi
if [ -z "$FLOW_ARN" ]; then
    handle_error "Failed to extract flow ARN from output"
fi
echo "Flow ARN: $FLOW_ARN"

# Validate flow ARN format
if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-Z0-9:-]+
$ ]]; then
    handle_error "Invalid Flow ARN format: $FLOW_ARN"
fi

# Tag the flow
tag_mediaconnect_resource "$FLOW_ARN"

# Step 3: Add an output
echo "Step 3: Adding an output to the flow..."
add_output_output=""
if add_output_output=$(aws mediaconnect add-flow-outputs \
    --flow-arn "$FLOW_ARN" \
    --outputs "Name=$OUTPUT_NAME,Protocol=zixi-
push,Destination=198.51.100.11,Port=1024,StreamId=ZixiAwardsOutput" 2>&1); then
    echo "$add_output_output"
else
    handle_error "Failed to add output to flow"
fi

# Extract the output ARN
```

```
output_arn=""
output_arn=$(echo "$add_output_output" | jq -r ".Output.OutputArn // empty" 2>/dev/
null)
if [ -z "$output_arn" ]; then output_arn=$(echo "$add_output_output" | grep -o
 '"OutputArn": "[^"]*' | head -1 | cut -d'"' -f4); fi
if [ -z "$output_arn" ]; then
    echo "WARNING: Failed to extract output ARN from output"
else
    OUTPUT_ARN="$output_arn"
    echo "Output ARN: $OUTPUT_ARN"
    tag_mediaconnect_resource "$OUTPUT_ARN"
fi

# Step 4: Grant an entitlement
echo "Step 4: Granting an entitlement..."
grant_entitlement_output=""
if grant_entitlement_output=$(aws mediaconnect grant-flow-entitlements \
    --flow-arn "$FLOW_ARN" \
    --entitlements "Name=$ENTITLEMENT_NAME,Subscribers=222233334444" 2>&1); then
    echo "$grant_entitlement_output"
else
    handle_error "Failed to grant entitlement"
fi

# Extract the entitlement ARN
entitlement_arn=""
entitlement_arn=$(echo "$grant_entitlement_output" | jq -r
 '.Entitlement.EntitlementArn // empty' 2>/dev/null)
if [ -z "$entitlement_arn" ]; then
    entitlement_arn=$(echo "$grant_entitlement_output" | grep -o '"EntitlementArn":
 "[^"]*' | head -1 | cut -d'"' -f4)
fi
if [ -z "$entitlement_arn" ]; then
    echo "WARNING: Failed to extract entitlement ARN from output"
else
    ENTITLEMENT_ARN="$entitlement_arn"
    echo "Entitlement ARN: $ENTITLEMENT_ARN"
    tag_mediaconnect_resource "$ENTITLEMENT_ARN"
fi

# Step 5: List entitlements to share with affiliates
echo "Step 5: Listing entitlements for the flow..."
describe_flow_output=""
```

```
if describe_flow_output=$(aws mediaconnect describe-flow --flow-arn "$FLOW_ARN" --
query "Flow.Entitlements" 2>&1); then
    echo "Entitlements for the flow:"
    echo "$describe_flow_output"
else
    handle_error "Failed to describe flow"
fi

# Display information to share with affiliates
echo ""
echo "Information to share with your Philadelphia affiliate:"
echo "Entitlement ARN: ${ENTITLEMENT_ARN:-N/A}"
echo "AWS Region: $AWS_REGION"

# Display resource summary
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "The following resources were created:"
echo "1. Flow: $FLOW_NAME (ARN: $FLOW_ARN)"
echo "2. Output: $OUTPUT_NAME (ARN: ${OUTPUT_ARN:-N/A})"
echo "3. Entitlement: $ENTITLEMENT_NAME (ARN: ${ENTITLEMENT_ARN:-N/A})"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddFlowOutputs
  - CreateFlow
  - DeleteFlow
  - DescribeAvailabilityZones
  - DescribeFlow
  - GrantFlowEntitlements
  - ListFlows
  - StopFlow

Introdução às máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um par de chaves.
- Criar um grupo de segurança.
- Interromper e iniciar uma instância.
- Testar a persistência de um IP elástico.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EC2 Basics Tutorial Script - Revised
# This script demonstrates the basics of working with EC2 instances using AWS CLI
# Updated to use Amazon Linux 2023 and enhanced security settings

# Set up logging
LOG_FILE="ec2_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to log messages
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

# Function to handle errors
handle_error() {
  log "ERROR: $1"
  log "Cleaning up resources..."
  cleanup
  exit 1
}
```

```
# Function to clean up resources
cleanup() {
  log "Resources created:"

  if [ -n "$ASSOCIATION_ID" ]; then
    log "- Elastic IP Association: $ASSOCIATION_ID"
  fi

  if [ -n "$ALLOCATION_ID" ]; then
    log "- Elastic IP Allocation: $ALLOCATION_ID (IP: $ELASTIC_IP)"
  fi

  if [ -n "$INSTANCE_ID" ]; then
    log "- EC2 Instance: $INSTANCE_ID"
  fi

  if [ -n "$SECURITY_GROUP_ID" ]; then
    log "- Security Group: $SECURITY_GROUP_ID"
  fi

  if [ -n "$KEY_NAME" ]; then
    log "- Key Pair: $KEY_NAME (File: $KEY_FILE)"
  fi

  REPLY=y

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    log "Starting cleanup..."

    # Track cleanup failures
    CLEANUP_FAILURES=0

    # Disassociate Elastic IP if it exists
    if [ -n "$ASSOCIATION_ID" ]; then
      log "Disassociating Elastic IP..."
      if ! aws ec2 disassociate-address --association-id "$ASSOCIATION_ID"; then
        log "Failed to disassociate Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Release Elastic IP if it exists
    if [ -n "$ALLOCATION_ID" ]; then
```

```
      log "Releasing Elastic IP..."
      if ! aws ec2 release-address --allocation-id "$ALLOCATION_ID"; then
        log "Failed to release Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Terminate instance if it exists
    if [ -n "$INSTANCE_ID" ]; then
      log "Terminating instance $INSTANCE_ID..."
      if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" > /dev/null;
then
        log "Failed to terminate instance"
        ((CLEANUP_FAILURES++))
      else
        log "Waiting for instance to terminate..."
        if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"; then
          log "Failed while waiting for instance to terminate"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Delete security group if it exists
    if [ -n "$SECURITY_GROUP_ID" ]; then
      log "Deleting security group..."
      if ! aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"; then
        log "Failed to delete security group"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ]; then
      log "Deleting key pair..."
      if ! aws ec2 delete-key-pair --key-name "$KEY_NAME"; then
        log "Failed to delete key pair"
        ((CLEANUP_FAILURES++))
      fi

      # Remove key file
      if [ -f "$KEY_FILE" ]; then
        log "Removing key file..."
        if ! rm -f "$KEY_FILE"; then
```

```
          log "Failed to remove key file"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Report cleanup status
    if [ $CLEANUP_FAILURES -eq 0 ]; then
      log "Cleanup completed successfully."
    else
      log "WARNING: Cleanup completed with $CLEANUP_FAILURES failures. Some
 resources may not have been deleted properly."
    fi
  else
    log "Resources were not deleted."
  fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
KEY_NAME="ec2-tutorial-key-$RANDOM_ID"
SG_NAME="ec2-tutorial-sg-$RANDOM_ID"

# Create a directory for the key file
KEY_DIR=$(mktemp -d)
KEY_FILE="$KEY_DIR/$KEY_NAME.pem"

log "Starting EC2 basics tutorial script"
log "Random identifier: $RANDOM_ID"
log "Key name: $KEY_NAME"
log "Security group name: $SG_NAME"

# Step 1: Create a key pair
log "Creating key pair..."
KEY_RESULT=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial' --
output text)

if [ $? -ne 0 ] || [ -z "$KEY_RESULT" ]; then
  handle_error "Failed to create key pair"
fi

echo "$KEY_RESULT" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
log "Created key pair and saved to $KEY_FILE"
```

```
# Step 2: Create a security group
log "Creating security group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --group-name "$SG_NAME" \
  --description "Security group for EC2 tutorial" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query "GroupId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$SECURITY_GROUP_ID" ]; then
  handle_error "Failed to create security group"
fi

log "Created security group: $SECURITY_GROUP_ID"

# Get current public IP address for SSH access
MY_IP=$(curl -s http://checkip.amazonaws.com)
if [ $? -ne 0 ] || [ -z "$MY_IP" ]; then
  handle_error "Failed to get current IP address"
fi

log "Adding SSH ingress rule for IP $MY_IP..."
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr "$MY_IP/32" > /dev/null

if [ $? -ne 0 ]; then
  handle_error "Failed to add security group ingress rule"
fi

log "Added SSH ingress rule for IP $MY_IP"

# Step 3: Find an Amazon Linux 2023 AMI (updated from AL2)
log "Finding latest Amazon Linux 2023 AMI..."
AMI_ID=$(aws ssm get-parameters-by-path \
  --path "/aws/service/ami-amazon-linux-latest" \
  --query "Parameters[?contains(Name, 'al2023-ami-kernel-default-x86_64')].Value" \
  --output text | head -1)

if [ $? -ne 0 ] || [ -z "$AMI_ID" ]; then
```

```
  handle_error "Failed to find Amazon Linux 2023 AMI"
fi

log "Selected AMI: $AMI_ID"

# Get the architecture of the AMI
log "Getting AMI architecture..."
AMI_ARCH=$(aws ec2 describe-images \
  --image-ids "$AMI_ID" \
  --query "Images[0].Architecture" \
  --output text)

if [ $? -ne 0 ] || [ -z "$AMI_ARCH" ]; then
  handle_error "Failed to get AMI architecture"
fi

log "AMI architecture: $AMI_ARCH"

# Find a compatible instance type
log "Finding compatible instance type..."
# Directly use t2.micro for simplicity
INSTANCE_TYPE="t2.micro"
log "Using instance type: $INSTANCE_TYPE"

# Step 4: Launch an EC2 instance with enhanced security
log "Launching EC2 instance with IMDSv2 and encryption enabled..."
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --metadata-options "HttpTokens=required,HttpEndpoint=enabled" \
  --block-device-mappings "DeviceName=/dev/xvda,Ebs={Encrypted=true}" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' 'ResourceType=volume,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --count 1 \
  --query 'Instances[0].InstanceId' \
  --output text)

if [ $? -ne 0 ] || [ -z "$INSTANCE_ID" ]; then
  handle_error "Failed to launch EC2 instance"
fi
```

```
log "Launched instance $INSTANCE_ID. Waiting for it to start..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get instance details
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].
{ID:InstanceId,Type:InstanceType,State:State.Name,PublicIP:PublicIpAddress}' \
  --output json)

if [ $? -ne 0 ]; then
  handle_error "Failed to get instance details"
fi

log "Instance details: $INSTANCE_DETAILS"

# Get the public IP address
PUBLIC_IP=$(echo "$INSTANCE_DETAILS" | grep -oP '"PublicIP": "\K[^"]+')
if [ -z "$PUBLIC_IP" ]; then
  handle_error "Failed to get instance public IP"
fi

log "Instance public IP: $PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$PUBLIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 6: Stop and Start the Instance
log "Stopping instance $INSTANCE_ID..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
```

```
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get the new public IP address
NEW_PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$NEW_PUBLIC_IP" ]; then
  handle_error "Failed to get new public IP"
fi

log "Instance restarted with new public IP: $NEW_PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$NEW_PUBLIC_IP"

# Step 7: Allocate and Associate an Elastic IP Address
log "Allocating Elastic IP address..."
ALLOCATION_RESULT=$(aws ec2 allocate-address \
  --domain vpc \
  --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' \
  --query '[PublicIp,AllocationId]' \
  --output text)

if [ $? -ne 0 ] || [ -z "$ALLOCATION_RESULT" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

ELASTIC_IP=$(echo "$ALLOCATION_RESULT" | awk '{print $1}')
ALLOCATION_ID=$(echo "$ALLOCATION_RESULT" | awk '{print $2}')

log "Allocated Elastic IP: $ELASTIC_IP with ID: $ALLOCATION_ID"
```

```
log "Associating Elastic IP with instance..."
ASSOCIATION_ID=$(aws ec2 associate-address \
  --instance-id "$INSTANCE_ID" \
  --allocation-id "$ALLOCATION_ID" \
  --query "AssociationId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$ASSOCIATION_ID" ]; then
  handle_error "Failed to associate Elastic IP"
fi

log "Associated Elastic IP with instance. Association ID: $ASSOCIATION_ID"
log "To connect to your instance using the Elastic IP, run: ssh -i $KEY_FILE ec2-
user@$ELASTIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 8: Test the Elastic IP by Stopping and Starting the Instance
log "Stopping instance $INSTANCE_ID to test Elastic IP persistence..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi
```

```
# Verify the Elastic IP is still associated
CURRENT_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$CURRENT_IP" ]; then
  handle_error "Failed to get current public IP"
fi

log "Current public IP address: $CURRENT_IP"
log "Elastic IP address: $ELASTIC_IP"

if [ "$CURRENT_IP" = "$ELASTIC_IP" ]; then
  log "Success! The Elastic IP is still associated with your instance."
else
  log "Something went wrong. The Elastic IP is not associated with your instance."
fi

log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$ELASTIC_IP"

# Step 9: Clean up resources
log "Tutorial completed successfully!"
cleanup

exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - DisassociateAddress
  - GetParametersByPath

- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- Wait

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi
```

```
    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"
```

```
# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
```

```
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
```

```
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
```

```
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")
```

```
# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi
```

```
echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
```

```
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
```

```
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi
```

```
echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
```

```
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
```

```
DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

# Terminate EC2 instance
echo "Terminating EC2 instance $INSTANCE_ID..."
TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
echo "Waiting for instance to terminate..."
aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

# Clean up IAM resources
echo "Removing role from instance profile..."
REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

echo "Deleting instance profile..."
DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

echo "Deleting FIS role policy..."
DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME")
check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

echo "Detaching policy from EC2 role..."
DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

echo "Deleting FIS role..."
DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

echo "Deleting EC2 role..."
DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
  --role-name "$EC2_ROLE_NAME")
check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"
```

```
    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances

- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Rede privada virtual com gerenciamento de endereços IP

O exemplo de código a seguir mostra como:

- Definir e configurar o Gerenciador de endereços IP (IPAM) da Amazon VPC usando a CLI.
- Criar um IPAM com regiões de operação (p. ex., us-east-1 e us-west-2).
- Recuperar o ID do escopo privado do IPAM.
- Criar uma estrutura hierárquica de grupos de IPv4 (grupos de nível superior, regionais e de desenvolvimento).
- Provisione blocos CIDR para cada pool (por exemplo, 10.0.0. 0/8, 10.0.0. 0/16, 10.0.0. 0/24).
- Criar uma VPC usando um CIDR alocado em um grupo do IPAM.
- Verificar as alocações do grupo do IPAM e a criação da VPC.
- Solucionar problemas comuns, como erros de permissão, falhas na alocação de CIDR e violações de dependências.
- Limpar os recursos do IPAM (VPC, grupos, CIDRs e IPAM) para evitar cobranças desnecessárias.
- Explorar as próximas etapas para recursos avançados do IPAM.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# IPAM Getting Started CLI Script - Version 7
# This script creates an IPAM, creates a hierarchy of IP address pools, and
 allocates a CIDR to a VPC
# Fixed to correctly identify the private scope ID, wait for resources to be
 available, add locale to development pool,
# use the correct parameter names for VPC creation, and wait for CIDR provisioning
 to complete

# Set up logging
LOG_FILE="ipam_script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting IPAM setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED:"
    echo "============================================"

    if [ -n "$VPC_ID" ]; then
        echo "VPC: $VPC_ID"
    fi

    if [ -n "$DEV_POOL_ID" ]; then
        echo "Development Pool: $DEV_POOL_ID"
    fi

    if [ -n "$REGIONAL_POOL_ID" ]; then
        echo "Regional Pool: $REGIONAL_POOL_ID"
```

```
    fi

    if [ -n "$TOP_POOL_ID" ]; then
        echo "Top-level Pool: $TOP_POOL_ID"
    fi

    if [ -n "$IPAM_ID" ]; then
        echo "IPAM: $IPAM_ID"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."

        # Delete resources in reverse order of creation to handle dependencies

        if [ -n "$VPC_ID" ]; then
            echo "Deleting VPC: $VPC_ID"
            aws ec2 delete-vpc --vpc-id "$VPC_ID" || echo "Failed to delete VPC"
            echo "Waiting for VPC to be deleted..."
            sleep 10
        fi

        if [ -n "$DEV_POOL_ID" ]; then
            echo "Deleting Development Pool: $DEV_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$DEV_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Development Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$DEV_POOL_ID" --
cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$DEV_POOL_ID" || echo "Failed
 to delete Development Pool"
            echo "Waiting for Development Pool to be deleted..."
            sleep 10
```

```
        fi

        if [ -n "$REGIONAL_POOL_ID" ]; then
            echo "Deleting Regional Pool: $REGIONAL_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$REGIONAL_POOL_ID"
 --query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Regional Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id
 "$REGIONAL_POOL_ID" --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$REGIONAL_POOL_ID" || echo
 "Failed to delete Regional Pool"
            echo "Waiting for Regional Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$TOP_POOL_ID" ]; then
            echo "Deleting Top-level Pool: $TOP_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$TOP_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Top-level Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$TOP_POOL_ID" --
cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$TOP_POOL_ID" || echo "Failed
 to delete Top-level Pool"
            echo "Waiting for Top-level Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$IPAM_ID" ]; then
            echo "Deleting IPAM: $IPAM_ID"
            aws ec2 delete-ipam --ipam-id "$IPAM_ID" || echo "Failed to delete IPAM"
        fi

        echo "Cleanup completed."
    else
        echo "Cleanup skipped. Resources will remain in your account."
```

```
    fi
}

# Function to wait for a pool to be in the 'create-complete' state
wait_for_pool() {
    local pool_id=$1
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for pool $pool_id to be available..."

    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 describe-ipam-pools --ipam-pool-ids "$pool_id" --query
 'IpamPools[0].State' --output text)

        if [ "$state" = "create-complete" ]; then
            echo "Pool $pool_id is now available (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: Pool $pool_id is in state: $state.
 Waiting..."
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for pool $pool_id to be available"
    return 1
}

# Function to wait for a CIDR to be fully provisioned
wait_for_cidr_provisioning() {
    local pool_id=$1
    local cidr=$2
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for CIDR $cidr to be fully provisioned in pool $pool_id..."

    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$pool_id" --query
 "IpamPoolCidrs[?Cidr=='$cidr'].State" --output text)
```

```
        if [ "$state" = "provisioned" ]; then
            echo "CIDR $cidr is now fully provisioned (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: CIDR $cidr is in state: $state.
 Waiting..."
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for CIDR $cidr to be provisioned"
    return 1
}

# Step 1: Create an IPAM
echo "Creating IPAM..."
IPAM_RESULT=$(aws ec2 create-ipam \
    --description "My IPAM" \
    --operating-regions RegionName=us-east-1 RegionName=us-west-2 \
    --tag-specifications 'ResourceType=ipam,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create IPAM"
fi

IPAM_ID=$(echo "$IPAM_RESULT" | grep -o '"IpamId": "[^"]*' | cut -d'"' -f4)
echo "IPAM created with ID: $IPAM_ID"

# Wait for IPAM to be created and available
echo "Waiting for IPAM to be available..."
sleep 20

# Step 2: Get the IPAM Scope ID - FIXED to correctly identify the private scope
echo "Getting IPAM Scope ID..."
SCOPE_RESULT=$(aws ec2 describe-ipams --ipam-id "$IPAM_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to get IPAM details"
fi

# Extract the private scope ID directly from the IPAM details
```

```
PRIVATE_SCOPE_ID=$(echo "$SCOPE_RESULT" | grep -o '"PrivateDefaultScopeId": "[^"]*'
 | cut -d'"' -f4)
echo "Private Scope ID: $PRIVATE_SCOPE_ID"

if [ -z "$PRIVATE_SCOPE_ID" ]; then
    handle_error "Failed to get Private Scope ID"
fi

# Step 3: Create a Top-Level IPv4 Pool
echo "Creating Top-level IPv4 Pool..."
TOP_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --address-family ipv4 \
    --description "Top-level pool" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Top-level Pool"
fi

TOP_POOL_ID=$(echo "$TOP_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -d'"' -
f4)
echo "Top-level Pool created with ID: $TOP_POOL_ID"

# Wait for the top-level pool to be available
if ! wait_for_pool "$TOP_POOL_ID"; then
    handle_error "Top-level Pool did not become available in time"
fi

# Provision CIDR to the top-level pool
echo "Provisioning CIDR to Top-level Pool..."
TOP_POOL_CIDR="10.0.0.0/8"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$TOP_POOL_ID" \
    --cidr "$TOP_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Top-level Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
```

```
if ! wait_for_cidr_provisioning "$TOP_POOL_ID" "$TOP_POOL_CIDR"; then
    handle_error "CIDR provisioning to Top-level Pool did not complete in time"
fi

# Step 4: Create a Regional IPv4 Pool
echo "Creating Regional IPv4 Pool..."
REGIONAL_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$TOP_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Regional pool in us-east-1" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Regional Pool"
fi

REGIONAL_POOL_ID=$(echo "$REGIONAL_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' |
 cut -d'"' -f4)
echo "Regional Pool created with ID: $REGIONAL_POOL_ID"

# Wait for the regional pool to be available
if ! wait_for_pool "$REGIONAL_POOL_ID"; then
    handle_error "Regional Pool did not become available in time"
fi

# Provision CIDR to the regional pool
echo "Provisioning CIDR to Regional Pool..."
REGIONAL_POOL_CIDR="10.0.0.0/16"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$REGIONAL_POOL_ID" \
    --cidr "$REGIONAL_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Regional Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$REGIONAL_POOL_ID" "$REGIONAL_POOL_CIDR"; then
    handle_error "CIDR provisioning to Regional Pool did not complete in time"
```

```
fi

# Step 5: Create a Development IPv4 Pool - FIXED to include locale
echo "Creating Development IPv4 Pool..."
DEV_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$REGIONAL_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Development pool" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Development Pool"
fi

DEV_POOL_ID=$(echo "$DEV_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -d'"' -
f4)
echo "Development Pool created with ID: $DEV_POOL_ID"

# Wait for the development pool to be available
if ! wait_for_pool "$DEV_POOL_ID"; then
    handle_error "Development Pool did not become available in time"
fi

# Provision CIDR to the development pool
echo "Provisioning CIDR to Development Pool..."
DEV_POOL_CIDR="10.0.0.0/24"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$DEV_POOL_ID" \
    --cidr "$DEV_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Development Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$DEV_POOL_ID" "$DEV_POOL_CIDR"; then
    handle_error "CIDR provisioning to Development Pool did not complete in time"
fi
```

```
# Step 6: Create a VPC Using an IPAM Pool CIDR - FIXED to use the correct parameter
 names and a smaller netmask length
echo "Creating VPC using IPAM Pool CIDR..."
VPC_RESULT=$(aws ec2 create-vpc \
    --ipv4-ipam-pool-id "$DEV_POOL_ID" \
    --ipv4-netmask-length 26 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=IPAM-VPC},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create VPC"
fi

VPC_ID=$(echo "$VPC_RESULT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID"

# Step 7: Verify the IPAM Pool Allocation
echo "Verifying IPAM Pool Allocation..."
ALLOCATION_RESULT=$(aws ec2 get-ipam-pool-allocations \
    --ipam-pool-id "$DEV_POOL_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to verify IPAM Pool Allocation"
fi

echo "IPAM Pool Allocation verified:"
echo "$ALLOCATION_RESULT" | grep -A 5 "Allocations"

echo ""
echo "IPAM setup completed successfully!"
echo ""

# Prompt for cleanup
cleanup_resources

echo "Script completed at $(date)"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateIpam
  - CreateIpamPool

- CreateVpc
- DeleteIpam
- DeleteIpamPool
- DeleteVpc
- DeprovisionIpamPoolCidr
- DescribeIpamPools
- DescribeIpams
- DescribeVpcs
- GetIpamPoolAllocations
- GetIpamPoolCidrs
- ProvisionIpamPoolCidr

Rede privada virtual com servidores privados

O exemplo de código a seguir mostra como:

- Criar uma VPC com sub-redes privadas e gateways NAT usando a CLI.
- Configurar os componentes necessários, como VPC, sub-redes, tabelas de rotas e gateways NAT.
- Configurar grupos de segurança e perfis do IAM para acesso e segurança adequados.
- Usar os comandos da CLI para automatizar a criação e a configuração desses recursos.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC with Private Subnets and NAT Gateways (IMDSv2 Compliant Version)
# This script creates a VPC with public and private subnets in two Availability
 Zones,
```

```
# NAT gateways, an internet gateway, route tables, a VPC endpoint for S3,
# security groups, a launch template, an Auto Scaling group, and an Application Load
 Balancer.

# Set up logging
LOG_FILE="vpc-private-subnets-nat.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Cleanup function to delete all created resources
cleanup_resources() {
  echo "Cleaning up resources..."

  # Delete Auto Scaling group if it exists
  if [ -n "${ASG_NAME:-}" ]; then
    echo "Deleting Auto Scaling group: $ASG_NAME"
    aws autoscaling delete-auto-scaling-group --auto-scaling-group-name "$ASG_NAME"
 --force-delete
    echo "Waiting for Auto Scaling group to be deleted..."
    aws autoscaling wait auto-scaling-groups-deleted --auto-scaling-group-names
 "$ASG_NAME"
  fi

  # Delete load balancer if it exists
  if [ -n "${LB_ARN:-}" ]; then
    echo "Deleting load balancer: $LB_ARN"
    aws elbv2 delete-load-balancer --load-balancer-arn "$LB_ARN"
    # Wait for load balancer to be deleted
    sleep 30
  fi

  # Delete target group if it exists
  if [ -n "${TARGET_GROUP_ARN:-}" ]; then
    echo "Deleting target group: $TARGET_GROUP_ARN"
    aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN"
  fi

  # Delete launch template if it exists
  if [ -n "${LAUNCH_TEMPLATE_NAME:-}" ]; then
    echo "Deleting launch template: $LAUNCH_TEMPLATE_NAME"
    aws ec2 delete-launch-template --launch-template-name "$LAUNCH_TEMPLATE_NAME"
  fi

  # Delete NAT Gateways if they exist
  if [ -n "${NAT_GW1_ID:-}" ]; then
```

```
  echo "Deleting NAT Gateway 1: $NAT_GW1_ID"
  aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW1_ID"
fi

if [ -n "${NAT_GW2_ID:-}" ]; then
  echo "Deleting NAT Gateway 2: $NAT_GW2_ID"
  aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW2_ID"
fi

# Wait for NAT Gateways to be deleted
if [ -n "${NAT_GW1_ID:-}" ] || [ -n "${NAT_GW2_ID:-}" ]; then
  echo "Waiting for NAT Gateways to be deleted..."
  sleep 60
fi

# Release Elastic IPs if they exist
if [ -n "${EIP1_ALLOC_ID:-}" ]; then
  echo "Releasing Elastic IP 1: $EIP1_ALLOC_ID"
  aws ec2 release-address --allocation-id "$EIP1_ALLOC_ID"
fi

if [ -n "${EIP2_ALLOC_ID:-}" ]; then
  echo "Releasing Elastic IP 2: $EIP2_ALLOC_ID"
  aws ec2 release-address --allocation-id "$EIP2_ALLOC_ID"
fi

# Delete VPC endpoint if it exists
if [ -n "${VPC_ENDPOINT_ID:-}" ]; then
  echo "Deleting VPC endpoint: $VPC_ENDPOINT_ID"
  aws ec2 delete-vpc-endpoints --vpc-endpoint-ids "$VPC_ENDPOINT_ID"
fi

# Delete security groups if they exist
if [ -n "${APP_SG_ID:-}" ]; then
  echo "Deleting application security group: $APP_SG_ID"
  aws ec2 delete-security-group --group-id "$APP_SG_ID"
fi

if [ -n "${LB_SG_ID:-}" ]; then
  echo "Deleting load balancer security group: $LB_SG_ID"
  aws ec2 delete-security-group --group-id "$LB_SG_ID"
fi

# Detach and delete Internet Gateway if it exists
```

```
  if [ -n "${IGW_ID:-}" ] && [ -n "${VPC_ID:-}" ]; then
    echo "Detaching Internet Gateway: $IGW_ID from VPC: $VPC_ID"
    aws ec2 detach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
"$VPC_ID"
    echo "Deleting Internet Gateway: $IGW_ID"
    aws ec2 delete-internet-gateway --internet-gateway-id "$IGW_ID"
  fi

  # Delete route table associations and route tables if they exist
  if [ -n "${PUBLIC_RT_ASSOC1_ID:-}" ]; then
    echo "Disassociating public route table from subnet 1: $PUBLIC_RT_ASSOC1_ID"
    aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC1_ID"
  fi

  if [ -n "${PUBLIC_RT_ASSOC2_ID:-}" ]; then
    echo "Disassociating public route table from subnet 2: $PUBLIC_RT_ASSOC2_ID"
    aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC2_ID"
  fi

  if [ -n "${PRIVATE_RT1_ASSOC_ID:-}" ]; then
    echo "Disassociating private route table 1: $PRIVATE_RT1_ASSOC_ID"
    aws ec2 disassociate-route-table --association-id "$PRIVATE_RT1_ASSOC_ID"
  fi

  if [ -n "${PRIVATE_RT2_ASSOC_ID:-}" ]; then
    echo "Disassociating private route table 2: $PRIVATE_RT2_ASSOC_ID"
    aws ec2 disassociate-route-table --association-id "$PRIVATE_RT2_ASSOC_ID"
  fi

  if [ -n "${PUBLIC_RT_ID:-}" ]; then
    echo "Deleting public route table: $PUBLIC_RT_ID"
    aws ec2 delete-route-table --route-table-id "$PUBLIC_RT_ID"
  fi

  if [ -n "${PRIVATE_RT1_ID:-}" ]; then
    echo "Deleting private route table 1: $PRIVATE_RT1_ID"
    aws ec2 delete-route-table --route-table-id "$PRIVATE_RT1_ID"
  fi

  if [ -n "${PRIVATE_RT2_ID:-}" ]; then
    echo "Deleting private route table 2: $PRIVATE_RT2_ID"
    aws ec2 delete-route-table --route-table-id "$PRIVATE_RT2_ID"
  fi
```

```
  # Delete subnets if they exist
  if [ -n "${PUBLIC_SUBNET1_ID:-}" ]; then
    echo "Deleting public subnet 1: $PUBLIC_SUBNET1_ID"
    aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET1_ID"
  fi

  if [ -n "${PUBLIC_SUBNET2_ID:-}" ]; then
    echo "Deleting public subnet 2: $PUBLIC_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET2_ID"
  fi

  if [ -n "${PRIVATE_SUBNET1_ID:-}" ]; then
    echo "Deleting private subnet 1: $PRIVATE_SUBNET1_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET1_ID"
  fi

  if [ -n "${PRIVATE_SUBNET2_ID:-}" ]; then
    echo "Deleting private subnet 2: $PRIVATE_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET2_ID"
  fi

  # Delete VPC if it exists
  if [ -n "${VPC_ID:-}" ]; then
    echo "Deleting VPC: $VPC_ID"
    aws ec2 delete-vpc --vpc-id "$VPC_ID"
  fi

  echo "Cleanup completed."
}

# Error handling function
handle_error() {
  echo "ERROR: $1"
  echo "Attempting to clean up resources..."
  cleanup_resources
  exit 1
}

# Function to check command success
check_command() {
  if [ $? -ne 0 ]; then
    handle_error "$1"
  fi
}
```

```
# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Create VPC
echo "Creating VPC..."
VPC_RESULT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=ProductionVPC-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create VPC"

VPC_ID=$(echo "$VPC_RESULT" | jq -r '.Vpc.VpcId')
echo "VPC created with ID: $VPC_ID"

# Get Availability Zones
echo "Getting Availability Zones..."
AZ_RESULT=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0:2].ZoneName' --output text)
check_command "Failed to get Availability Zones"

# Convert space-separated output to array
read -r -a AZS <<< "$AZ_RESULT"
AZ1=${AZS[0]}
AZ2=${AZS[1]}
echo "Using Availability Zones: $AZ1 and $AZ2"

# Create subnets
echo "Creating subnets..."
PUBLIC_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.0.0/24 --availability-zone "$AZ1" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create public subnet 1"
PUBLIC_SUBNET1_ID=$(echo "$PUBLIC_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.1.0/24 --availability-zone "$AZ1" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private subnet 1"
PRIVATE_SUBNET1_ID=$(echo "$PRIVATE_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')
```

```
PUBLIC_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.2.0/24 --availability-zone "$AZ2" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create public subnet 2"
PUBLIC_SUBNET2_ID=$(echo "$PUBLIC_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.3.0/24 --availability-zone "$AZ2" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private subnet 2"
PRIVATE_SUBNET2_ID=$(echo "$PRIVATE_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

echo "Subnets created with IDs:"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_RESULT=$(aws ec2 create-internet-gateway --tag-specifications
 "ResourceType=internet-gateway,Tags=[{Key=Name,Value=ProductionIGW-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create Internet Gateway"
IGW_ID=$(echo "$IGW_RESULT" | jq -r '.InternetGateway.InternetGatewayId')
echo "Internet Gateway created with ID: $IGW_ID"

# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id "$VPC_ID"
check_command "Failed to attach Internet Gateway to VPC"

# Create route tables
echo "Creating route tables..."
PUBLIC_RT_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PublicRouteTable-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-
gs}]")
check_command "Failed to create public route table"
PUBLIC_RT_ID=$(echo "$PUBLIC_RT_RESULT" | jq -r '.RouteTable.RouteTableId')
```

```
PRIVATE_RT1_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PrivateRouteTable1-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-
gs}]")
check_command "Failed to create private route table 1"
PRIVATE_RT1_ID=$(echo "$PRIVATE_RT1_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT2_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PrivateRouteTable2-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-
gs}]")
check_command "Failed to create private route table 2"
PRIVATE_RT2_ID=$(echo "$PRIVATE_RT2_RESULT" | jq -r '.RouteTable.RouteTableId')

echo "Route tables created with IDs:"
echo "Public Route Table: $PUBLIC_RT_ID"
echo "Private Route Table 1: $PRIVATE_RT1_ID"
echo "Private Route Table 2: $PRIVATE_RT2_ID"

# Add route to Internet Gateway in public route table
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT_ID" --destination-cidr-block
 0.0.0.0/0 --gateway-id "$IGW_ID"
check_command "Failed to add route to Internet Gateway"

# Associate subnets with route tables
echo "Associating subnets with route tables..."
PUBLIC_RT_ASSOC1_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET1_ID")
check_command "Failed to associate public subnet 1 with route table"
PUBLIC_RT_ASSOC1_ID=$(echo "$PUBLIC_RT_ASSOC1_RESULT" | jq -r '.AssociationId')

PUBLIC_RT_ASSOC2_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET2_ID")
check_command "Failed to associate public subnet 2 with route table"
PUBLIC_RT_ASSOC2_ID=$(echo "$PUBLIC_RT_ASSOC2_RESULT" | jq -r '.AssociationId')

PRIVATE_RT1_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PRIVATE_RT1_ID" --subnet-id "$PRIVATE_SUBNET1_ID")
check_command "Failed to associate private subnet 1 with route table"
PRIVATE_RT1_ASSOC_ID=$(echo "$PRIVATE_RT1_ASSOC_RESULT" | jq -r '.AssociationId')

PRIVATE_RT2_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PRIVATE_RT2_ID" --subnet-id "$PRIVATE_SUBNET2_ID")
```

```
check_command "Failed to associate private subnet 2 with route table"
PRIVATE_RT2_ASSOC_ID=$(echo "$PRIVATE_RT2_ASSOC_RESULT" | jq -r '.AssociationId')

echo "Route table associations created with IDs:"
echo "Public Subnet 1 Association: $PUBLIC_RT_ASSOC1_ID"
echo "Public Subnet 2 Association: $PUBLIC_RT_ASSOC2_ID"
echo "Private Subnet 1 Association: $PRIVATE_RT1_ASSOC_ID"
echo "Private Subnet 2 Association: $PRIVATE_RT2_ASSOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateways..."

# Allocate Elastic IPs for NAT Gateways
echo "Allocating Elastic IPs for NAT Gateways..."
EIP1_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
 "ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT1-EIP-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to allocate Elastic IP 1"
EIP1_ALLOC_ID=$(echo "$EIP1_RESULT" | jq -r '.AllocationId')

EIP2_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
 "ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT2-EIP-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to allocate Elastic IP 2"
EIP2_ALLOC_ID=$(echo "$EIP2_RESULT" | jq -r '.AllocationId')

echo "Elastic IPs allocated with IDs:"
echo "EIP 1 Allocation ID: $EIP1_ALLOC_ID"
echo "EIP 2 Allocation ID: $EIP2_ALLOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateway in public subnet 1..."
NAT_GW1_RESULT=$(aws ec2 create-nat-gateway --subnet-id
 "$PUBLIC_SUBNET1_ID" --allocation-id "$EIP1_ALLOC_ID" --tag-specifications
 "ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create NAT Gateway 1"
NAT_GW1_ID=$(echo "$NAT_GW1_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "Creating NAT Gateway in public subnet 2..."
NAT_GW2_RESULT=$(aws ec2 create-nat-gateway --subnet-id
 "$PUBLIC_SUBNET2_ID" --allocation-id "$EIP2_ALLOC_ID" --tag-specifications
 "ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
```

```
check_command "Failed to create NAT Gateway 2"
NAT_GW2_ID=$(echo "$NAT_GW2_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "NAT Gateways created with IDs:"
echo "NAT Gateway 1: $NAT_GW1_ID"
echo "NAT Gateway 2: $NAT_GW2_ID"

# Wait for NAT Gateways to be available
echo "Waiting for NAT Gateways to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW1_ID"
check_command "NAT Gateway 1 did not become available"
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW2_ID"
check_command "NAT Gateway 2 did not become available"
echo "NAT Gateways are now available"

# Add routes to NAT Gateways in private route tables
echo "Adding routes to NAT Gateways in private route tables..."
aws ec2 create-route --route-table-id "$PRIVATE_RT1_ID" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW1_ID"
check_command "Failed to add route to NAT Gateway 1"

aws ec2 create-route --route-table-id "$PRIVATE_RT2_ID" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW2_ID"
check_command "Failed to add route to NAT Gateway 2"

# Create VPC Endpoint for S3
echo "Creating VPC Endpoint for S3..."
S3_PREFIX_LIST_ID=$(aws ec2 describe-prefix-lists --filters "Name=prefix-
list-name,Values=com.amazonaws.$(aws configure get region).s3" --query
 'PrefixLists[0].PrefixListId' --output text)
check_command "Failed to get S3 prefix list ID"

VPC_ENDPOINT_RESULT=$(aws ec2 create-vpc-endpoint --vpc-id "$VPC_ID" --
service-name "com.amazonaws.$(aws configure get region).s3" --route-table-ids
 "$PRIVATE_RT1_ID" "$PRIVATE_RT2_ID" --tag-specifications "ResourceType=vpc-
endpoint,Tags=[{Key=Name,Value=S3-Endpoint-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create VPC endpoint for S3"
VPC_ENDPOINT_ID=$(echo "$VPC_ENDPOINT_RESULT" | jq -r '.VpcEndpoint.VpcEndpointId')
echo "VPC Endpoint created with ID: $VPC_ENDPOINT_ID"

# Create security groups
echo "Creating security groups..."
```

```
LB_SG_RESULT=$(aws ec2 create-security-group --group-name "LoadBalancerSG-
$RANDOM_ID" --description "Security group for the load balancer"
 --vpc-id "$VPC_ID" --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=LoadBalancerSG-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create load balancer security group"
LB_SG_ID=$(echo "$LB_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from anywhere to the load balancer
aws ec2 authorize-security-group-ingress --group-id "$LB_SG_ID" --protocol tcp --
port 80 --cidr 0.0.0.0/0
check_command "Failed to authorize ingress to load balancer security group"

APP_SG_RESULT=$(aws ec2 create-security-group --group-name "AppServerSG-$RANDOM_ID"
 --description "Security group for the application servers" --vpc-id "$VPC_ID" --
tag-specifications "ResourceType=security-group,Tags=[{Key=Name,Value=AppServerSG-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-
gs}]")
check_command "Failed to create application server security group"
APP_SG_ID=$(echo "$APP_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from the load balancer security group to the
 application servers
aws ec2 authorize-security-group-ingress --group-id "$APP_SG_ID" --protocol tcp --
port 80 --source-group "$LB_SG_ID"
check_command "Failed to authorize ingress to application server security group"

echo "Security groups created with IDs:"
echo "Load Balancer Security Group: $LB_SG_ID"
echo "Application Server Security Group: $APP_SG_ID"

# Create a launch template
echo "Creating launch template..."

# Create user data script with IMDSv2 support
cat > user-data.sh << 'EOF'
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd

# Use IMDSv2 with session token
```

```
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600")
AZ=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/
meta-data/placement/availability-zone)
HOSTNAME=$(hostname -f)

echo "<h1>Hello from $HOSTNAME in $AZ</h1>" > /var/www/html/index.html
EOF

# Encode user data
USER_DATA=$(base64 -w 0 user-data.sh)

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon --filters "Name=name,Values=amzn2-
ami-hvm-*-x86_64-gp2" "Name=state,Values=available" --query 'sort_by(Images,
 &CreationDate)[-1].ImageId' --output text)
check_command "Failed to get latest Amazon Linux 2 AMI"
echo "Using AMI: $AMI_ID"

# Create launch template with IMDSv2 required
LAUNCH_TEMPLATE_NAME="AppServerTemplate-$RANDOM_ID"
echo "Creating launch template: $LAUNCH_TEMPLATE_NAME"

aws ec2 create-launch-template \
  --launch-template-name "$LAUNCH_TEMPLATE_NAME" \
  --version-description "Initial version" \
  --tag-specifications "ResourceType=launch-template,Tags=[{Key=Name,Value=
$LAUNCH_TEMPLATE_NAME},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-
private-servers-gs}]" \
  --launch-template-data "{
    \"NetworkInterfaces\": [{
      \"DeviceIndex\": 0,
      \"Groups\": [\"$APP_SG_ID\"],
      \"DeleteOnTermination\": true
    }],
    \"ImageId\": \"$AMI_ID\",
    \"InstanceType\": \"t3.micro\",
    \"UserData\": \"$USER_DATA\",
    \"MetadataOptions\": {
      \"HttpTokens\": \"required\",
      \"HttpEndpoint\": \"enabled\"
    },
    \"TagSpecifications\": [{
```

```
      \"ResourceType\": \"instance\",
      \"Tags\": [{
        \"Key\": \"Name\",
        \"Value\": \"AppServer-$RANDOM_ID\"
      }]
    }]
  }"
check_command "Failed to create launch template"

# Create target group
echo "Creating target group..."
TARGET_GROUP_NAME="AppTargetGroup-$RANDOM_ID"
TARGET_GROUP_RESULT=$(aws elbv2 create-target-group \
  --name "$TARGET_GROUP_NAME" \
  --protocol HTTP \
  --port 80 \
  --vpc-id "$VPC_ID" \
  --target-type instance \
  --health-check-protocol HTTP \
  --health-check-path "/" \
  --health-check-port traffic-port \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=vpc-private-servers-gs)
check_command "Failed to create target group"
TARGET_GROUP_ARN=$(echo "$TARGET_GROUP_RESULT" | jq -r
 '.TargetGroups[0].TargetGroupArn')
echo "Target group created with ARN: $TARGET_GROUP_ARN"

# Create load balancer
echo "Creating load balancer..."
LB_NAME="AppLoadBalancer-$RANDOM_ID"
LB_RESULT=$(aws elbv2 create-load-balancer \
  --name "$LB_NAME" \
  --subnets "$PUBLIC_SUBNET1_ID" "$PUBLIC_SUBNET2_ID" \
  --security-groups "$LB_SG_ID" \
  --tags "Key=Name,Value=$LB_NAME" Key=project,Value=doc-smith
 Key=tutorial,Value=vpc-private-servers-gs)
check_command "Failed to create load balancer"
LB_ARN=$(echo "$LB_RESULT" | jq -r '.LoadBalancers[0].LoadBalancerArn')
echo "Load balancer created with ARN: $LB_ARN"

# Wait for load balancer to be active
echo "Waiting for load balancer to be active..."
aws elbv2 wait load-balancer-available --load-balancer-arns "$LB_ARN"
check_command "Load balancer did not become available"
```

```
# Create listener
echo "Creating listener..."
LISTENER_RESULT=$(aws elbv2 create-listener \
  --load-balancer-arn "$LB_ARN" \
  --protocol HTTP \
  --port 80 \
  --default-actions "Type=forward,TargetGroupArn=$TARGET_GROUP_ARN")
check_command "Failed to create listener"
LISTENER_ARN=$(echo "$LISTENER_RESULT" | jq -r '.Listeners[0].ListenerArn')
echo "Listener created with ARN: $LISTENER_ARN"

# Create Auto Scaling group
echo "Creating Auto Scaling group..."
ASG_NAME="AppAutoScalingGroup-$RANDOM_ID"
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name "$ASG_NAME" \
  --launch-template "LaunchTemplateName=$LAUNCH_TEMPLATE_NAME,Version=\$Latest" \
  --min-size 2 \
  --max-size 4 \
  --desired-capacity 2 \
  --vpc-zone-identifier "$PRIVATE_SUBNET1_ID,$PRIVATE_SUBNET2_ID" \
  --target-group-arns "$TARGET_GROUP_ARN" \
  --health-check-type ELB \
  --health-check-grace-period 300 \
  --tags "Key=Name,Value=AppServer-$RANDOM_ID,PropagateAtLaunch=true"
 Key=project,Value=doc-smith,PropagateAtLaunch=true Key=tutorial,Value=vpc-private-
servers-gs,PropagateAtLaunch=true
check_command "Failed to create Auto Scaling group"
echo "Auto Scaling group created with name: $ASG_NAME"

# Get load balancer DNS name
LB_DNS_NAME=$(aws elbv2 describe-load-balancers --load-balancer-arns "$LB_ARN" --
query 'LoadBalancers[0].DNSName' --output text)
check_command "Failed to get load balancer DNS name"

echo ""
echo "============================================"
echo "DEPLOYMENT COMPLETE"
echo "============================================"
echo "VPC ID: $VPC_ID"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID (AZ: $AZ1)"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID (AZ: $AZ1)"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID (AZ: $AZ2)"
```

```
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID (AZ: $AZ2)"
echo "NAT Gateway 1: $NAT_GW1_ID"
echo "NAT Gateway 2: $NAT_GW2_ID"
echo "Load Balancer: $LB_NAME"
echo "Auto Scaling Group: $ASG_NAME"
echo ""
echo "Your application will be available at: http://$LB_DNS_NAME"
echo "It may take a few minutes for the instances to launch and pass health checks."
echo ""

# Add health check monitoring
echo "============================================"
echo "MONITORING INSTANCE HEALTH AND LOAD BALANCER"
echo "============================================"
echo "Waiting for instances to launch and pass health checks..."
echo "This may take 3-5 minutes. Checking every 30 seconds..."

# Monitor instance health and load balancer accessibility
MAX_ATTEMPTS=10
ATTEMPT=1
HEALTHY_INSTANCES=0

while [ $ATTEMPT -le $MAX_ATTEMPTS ] && [ $HEALTHY_INSTANCES -lt 2 ]; do
  echo "Check attempt $ATTEMPT of $MAX_ATTEMPTS..."

  # Check Auto Scaling group instances
  echo "Checking Auto Scaling group instances..."
  ASG_INSTANCES=$(aws autoscaling describe-auto-scaling-groups --auto-
scaling-group-names "$ASG_NAME" --query 'AutoScalingGroups[0].Instances[*].
[InstanceId,HealthStatus]' --output json)
  echo "ASG Instances status:"
  echo "$ASG_INSTANCES" | jq -r '.[] | "Instance: \(.[0]), Health: \(.[1])"'

  # Check target group health
  echo "Checking target group health..."
  TARGET_HEALTH=$(aws elbv2 describe-target-health --target-group-arn
 "$TARGET_GROUP_ARN" --output json)
  echo "Target health status:"
  echo "$TARGET_HEALTH" | jq -r '.TargetHealthDescriptions[] | "Instance:
 \(.Target.Id), State: \(.TargetHealth.State), Reason: \(.TargetHealth.Reason // "N/
A"), Description: \(.TargetHealth.Description // "N/A")"'

  # Count healthy instances
```

```
 HEALTHY_INSTANCES=$(echo "$TARGET_HEALTH" | jq -r '[.TargetHealthDescriptions[] |
select(.TargetHealth.State=="healthy")] | length')
 echo "Number of healthy instances: $HEALTHY_INSTANCES of 2 expected"

 # Check if we have healthy instances
 if [ $HEALTHY_INSTANCES -ge 2 ]; then
   echo "All instances are healthy!"

   # Test load balancer accessibility
   echo "Testing load balancer accessibility..."
   HTTP_STATUS=$(curl -s -o /dev/null -w "%{http_code}" "http://$LB_DNS_NAME")

   if [ "$HTTP_STATUS" = "200" ]; then
     echo "Load balancer is accessible! HTTP Status: $HTTP_STATUS"
     echo "You can access your application at: http://$LB_DNS_NAME"

     # Try to get the content to verify IMDSv2 is working
     echo "Fetching content to verify IMDSv2 functionality..."
     CONTENT=$(curl -s "http://$LB_DNS_NAME")
     echo "Response from server:"
     echo "$CONTENT"

     # Check if the content contains the expected pattern
     if [[ "$CONTENT" == *"Hello from"* && "$CONTENT" == *"in"* ]]; then
       echo "IMDSv2 is working correctly! The instance was able to access metadata
using the token-based approach."
     else
       echo "Warning: Content doesn't match expected pattern. IMDSv2 functionality
could not be verified."
     fi

     break
   else
     echo "Load balancer returned HTTP status: $HTTP_STATUS"
     echo "Will try again in 30 seconds..."
   fi
 else
   echo "Waiting for instances to become healthy..."
   echo "Will check again in 30 seconds..."
 fi

 ATTEMPT=$((ATTEMPT+1))

 if [ $ATTEMPT -le $MAX_ATTEMPTS ]; then
```

```
    sleep 30
  fi
done

if [ $HEALTHY_INSTANCES -lt 2 ]; then
  echo "Warning: Not all instances are healthy after maximum attempts."
  echo "You may need to wait longer or check for configuration issues."
fi

echo "To test your application, run:"
echo "curl http://$LB_DNS_NAME"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  cleanup_resources
  echo "All resources have been deleted."
else
  echo "Resources will not be deleted. You can manually delete them later."
  echo "To delete resources, run this script again and choose to clean up."
fi
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateInternetGateway
  - CreateLaunchTemplate
  - CreateNatGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet

- CreateVpc
- CreateVpcEndpoint
- DeleteAutoScalingGroup
- DeleteInternetGateway
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteNatGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteTargetGroup
- DeleteVpc
- DeleteVpcEndpoints
- DescribeAvailabilityZones
- DescribeImages
- DescribePrefixLists
- DetachInternetGateway
- ReleaseAddress

Trabalho com criptografia, instantâneos e inicialização de volumes de armazenamento em bloco

O exemplo de código a seguir mostra como:

- Habilitar a criptografia do Amazon EBS por padrão.
- Criar snapshots do EBS.
- Criar e inicializar um volume por meio de um snapshot.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script for EBS operations: encryption, snapshots, and volume initialization
# This script demonstrates:
# 1. Enabling EBS encryption by default
# 2. Creating an EBS snapshot
# 3. Creating a volume from a snapshot
# Cost optimizations:
# - Reduced volume size from 1 GiB to 100 MiB for testing
# - Changed volume type to gp3 with cost-optimized IOPS/throughput
# - Added early cleanup to minimize storage duration
# - Removed unnecessary API calls for KMS key retrieval

set -euo pipefail

# Security: Restrict file permissions for log files
umask 0077

# Setup logging with secure permissions
LOG_FILE="ebs-operations-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting EBS operations script at $(date)"
echo "All operations will be logged to $LOG_FILE"

# Function to check command status
check_status() {
    local exit_code=$?
    if [ $exit_code -ne 0 ]; then
        echo "ERROR: $1 failed with exit code $exit_code. Exiting."
        cleanup_resources
        exit 1
    fi
}
```

```
# Function to cleanup resources with retry logic
cleanup_resources() {
    echo "Attempting to clean up resources..."
    local retry_count=0
    local max_retries=3

    if [ -n "${NEW_VOLUME_ID:-}" ]; then
        echo "Deleting new volume $NEW_VOLUME_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-volume --volume-id "$NEW_VOLUME_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted new volume $NEW_VOLUME_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $NEW_VOLUME_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete new volume $NEW_VOLUME_ID after
 $max_retries attempts"
                fi
            fi
        done
    fi

    if [ -n "${VOLUME_ID:-}" ]; then
        echo "Deleting original volume $VOLUME_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-volume --volume-id "$VOLUME_ID" --region "$AWS_REGION"
 2>/dev/null; then
                echo "Successfully deleted original volume $VOLUME_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $VOLUME_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete original volume $VOLUME_ID after
 $max_retries attempts"
                fi
            fi
```

```
        done
    fi

    if [ -n "${SNAPSHOT_ID:-}" ]; then
        echo "Deleting snapshot $SNAPSHOT_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-snapshot --snapshot-id "$SNAPSHOT_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted snapshot $SNAPSHOT_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $SNAPSHOT_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete snapshot $SNAPSHOT_ID after
 $max_retries attempts"
                fi
            fi
        done
    fi

    if [ "${ENCRYPTION_MODIFIED:-false}" = true ]; then
        echo "Restoring original encryption setting..."
        if [ "${ORIGINAL_ENCRYPTION:-}" = "False" ]; then
            aws ec2 disable-ebs-encryption-by-default --region "$AWS_REGION" 2>/dev/
null || echo "WARNING: Could not restore encryption setting"
        fi
    fi

    echo "Cleanup completed."
}

# Set trap for cleanup on exit
trap cleanup_resources EXIT

# Track created resources
VOLUME_ID=""
NEW_VOLUME_ID=""
SNAPSHOT_ID=""
ENCRYPTION_MODIFIED=false
ORIGINAL_ENCRYPTION=""
```

```
# Input validation function
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        exit 1
    fi

    # Verify AWS credentials are configured
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS credentials are not properly configured"
        exit 1
    fi
}

validate_aws_cli

# Security: Validate AWS region format
validate_region() {
    local region="$1"
    if [[ ! "$region" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
        echo "ERROR: Invalid AWS region format: $region"
        exit 1
    fi
}

# Get the current AWS region
AWS_REGION="${AWS_REGION:-$(aws configure get region)}"
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config. Using default: $AWS_REGION"
fi

validate_region "$AWS_REGION"
echo "Using AWS region: $AWS_REGION"

# Security: Validate volume ID format before use
validate_volume_id() {
    local volume_id="$1"
    if [[ ! "$volume_id" =~ ^vol-[a-z0-9]{17}$ ]]; then
        echo "ERROR: Invalid volume ID format: $volume_id"
        exit 1
    fi
}
```

```
# Security: Validate snapshot ID format before use
validate_snapshot_id() {
    local snapshot_id="$1"
    if [[ ! "$snapshot_id" =~ ^snap-[a-z0-9]{17}$ ]]; then
        echo "ERROR: Invalid snapshot ID format: $snapshot_id"
        exit 1
    fi
}

# Get availability zones in the region with caching
AVAILABILITY_ZONE=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --
query 'AvailabilityZones[0].ZoneName' --output text)
check_status "Getting availability zone"

# Security: Validate AZ format
if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}[a-z]$ ]]; then
    echo "ERROR: Invalid availability zone format: $AVAILABILITY_ZONE"
    exit 1
fi
echo "Using availability zone: $AVAILABILITY_ZONE"

# Step 1: Check and enable EBS encryption by default
echo "Step 1: Checking current EBS encryption by default setting..."
ORIGINAL_ENCRYPTION=$(aws ec2 get-ebs-encryption-by-default --region "$AWS_REGION"
 --query 'EbsEncryptionByDefault' --output text)
check_status "Checking encryption status"
echo "Current encryption by default setting: $ORIGINAL_ENCRYPTION"

if [ "$ORIGINAL_ENCRYPTION" = "False" ]; then
    echo "Enabling EBS encryption by default..."
    aws ec2 enable-ebs-encryption-by-default --region "$AWS_REGION"
    check_status "Enabling encryption by default"
    ENCRYPTION_MODIFIED=true
    echo "Updated encryption by default setting: True"
else
    echo "EBS encryption by default is already enabled."
fi

# Step 2: Create a test volume for snapshot with minimal size for cost optimization
echo "Step 2: Creating a test volume (1 GiB for testing)..."
VOLUME_ID=$(aws ec2 create-volume --region "$AWS_REGION" --availability-zone
 "$AVAILABILITY_ZONE" --size 1 --volume-type gp3 --iops 3000 --throughput 125
 --tag-specifications 'ResourceType=volume,Tags=[{Key=Name,Value=ebs-tutorial-
```

```
volume},{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-intermediate}]' --query 'VolumeId' --output text)
check_status "Creating test volume"

# Security: Validate volume ID
validate_volume_id "$VOLUME_ID"
echo "Created test volume: $VOLUME_ID"

# Wait for volume to become available with timeout
echo "Waiting for volume to become available..."
timeout 300 aws ec2 wait volume-available --region "$AWS_REGION" --volume-ids
 "$VOLUME_ID" || {
    echo "ERROR: Volume did not become available within timeout"
    exit 1
}
check_status "Waiting for volume"

# Step 3: Create a snapshot of the volume
echo "Step 3: Creating snapshot of the volume..."
SNAPSHOT_ID=$(aws ec2 create-snapshot --region "$AWS_REGION" --volume-id
 "$VOLUME_ID" --description "Snapshot for EBS tutorial - $(date +%Y-%m-%d)" --
tag-specifications 'ResourceType=snapshot,Tags=[{Key=Name,Value=ebs-tutorial-
snapshot},{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-
smith},{Key=tutorial,Value=ebs-intermediate}]' --query 'SnapshotId' --output text)
check_status "Creating snapshot"

# Security: Validate snapshot ID
validate_snapshot_id "$SNAPSHOT_ID"
echo "Created snapshot: $SNAPSHOT_ID"

# Wait for snapshot to complete with progress indication and timeout
echo "Waiting for snapshot to complete (this may take several minutes)..."
timeout 1800 aws ec2 wait snapshot-completed --region "$AWS_REGION" --snapshot-ids
 "$SNAPSHOT_ID" || {
    echo "ERROR: Snapshot did not complete within timeout"
    exit 1
}
check_status "Waiting for snapshot"
echo "Snapshot completed."

# Step 4: Create a new volume from the snapshot
echo "Step 4: Creating a new volume from the snapshot..."
NEW_VOLUME_ID=$(aws ec2 create-volume --region "$AWS_REGION" --
snapshot-id "$SNAPSHOT_ID" --availability-zone "$AVAILABILITY_ZONE"
```

```
 --volume-type gp3 --iops 3000 --throughput 125 --tag-specifications
 'ResourceType=volume,Tags=[{Key=Name,Value=ebs-tutorial-volume-from-snapshot},
{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-intermediate}]' --query 'VolumeId' --output text)
check_status "Creating new volume from snapshot"

# Security: Validate new volume ID
validate_volume_id "$NEW_VOLUME_ID"
echo "Created new volume from snapshot: $NEW_VOLUME_ID"

# Wait for new volume to become available with timeout
echo "Waiting for new volume to become available..."
timeout 300 aws ec2 wait volume-available --region "$AWS_REGION" --volume-ids
 "$NEW_VOLUME_ID" || {
    echo "ERROR: New volume did not become available within timeout"
    exit 1
}
check_status "Waiting for new volume"

# Display created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "Original Volume: $VOLUME_ID"
echo "Snapshot: $SNAPSHOT_ID"
echo "New Volume: $NEW_VOLUME_ID"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup process to minimize storage costs..."

# Delete the new volume immediately to reduce storage duration
echo "Deleting new volume $NEW_VOLUME_ID..."
aws ec2 delete-volume --region "$AWS_REGION" --volume-id "$NEW_VOLUME_ID"
check_status "Deleting new volume"

# Delete the original volume
echo "Deleting original volume $VOLUME_ID..."
aws ec2 delete-volume --region "$AWS_REGION" --volume-id "$VOLUME_ID"
```

```
check_status "Deleting original volume"

# Delete the snapshot
echo "Deleting snapshot $SNAPSHOT_ID..."
aws ec2 delete-snapshot --region "$AWS_REGION" --snapshot-id "$SNAPSHOT_ID"
check_status "Deleting snapshot"

# Restore original encryption setting if modified
if [ "${ENCRYPTION_MODIFIED:-false}" = true ]; then
    echo "Restoring original encryption setting..."
    if [ "${ORIGINAL_ENCRYPTION:-}" = "False" ]; then
        aws ec2 disable-ebs-encryption-by-default --region "$AWS_REGION"
        check_status "Disabling encryption by default"
    fi
fi

echo "Cleanup completed successfully."

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateSnapshot
  - CreateVolume
  - DeleteSnapshot
  - DeleteVolume
  - DescribeAvailabilityZones
  - DescribeVolumes
  - DetachVolume
  - DisableEbsEncryptionByDefault
  - EnableEbsEncryptionByDefault
  - GetEbsDefaultKmsKeyId
  - GetEbsEncryptionByDefault
  - Wait

Trabalho com conexões de emparelhamento de rede

O exemplo de código a seguir mostra como:

- Criar VPCs para emparelhamento.
- Criar uma conexão de emparelhamento da VPC.
- Atualizar tabelas de rotas.
- Verificar a conexão de emparelhamento da VPC.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC Peering Connection Script - Version 6 (Security Enhanced)
# This script establishes a VPC peering connection between two VPCs,
# creates subnets if needed, and configures the necessary route tables.
# It will use existing VPCs if available, or create new ones if needed.

set -euo pipefail

# Security: Set strict umask
umask 0077

# Initialize log file with restricted permissions
LOG_FILE="./vpc-peering-script-v6.log"
touch "$LOG_FILE"
chmod 0600 "$LOG_FILE"
echo "Starting VPC Peering script at $(date)" > "$LOG_FILE"

# Configuration
declare -r AWS_REGION="${AWS_REGION:-us-east-1}"
declare -r MAX_RETRIES=3
declare -r RETRY_DELAY=5

# Validate script is run from secure location
if [[ "$LOG_FILE" != /* ]] && [[ "$LOG_FILE" != ./* ]]; then
    echo "ERROR: Log file path must be absolute or relative starting with ./" >&2
```

```
    exit 1
fi

# Function to sanitize variable for safe command execution
sanitize_var() {
    local var="$1"
    if [[ ! "$var" =~ ^[a-zA-Z0-9_/.-]+$ ]]; then
        echo "ERROR: Invalid characters in variable: $var" | tee -a "$LOG_FILE"
        return 1
    fi
    echo "$var"
    return 0
}

# Function to escape string for safe use in commands
escape_string() {
    local string="$1"
    printf '%s\n' "$string" | sed -e 's/[\/&]/\\&/g'
}

# Function to log commands and their output securely
log_cmd() {
    local cmd="$1"

    # Validate command doesn't contain suspicious patterns
    if [[ "$cmd" =~ (\$\(|\`|;.*rm|;.*mv|;.*cp) ]]; then
        echo "ERROR: Suspicious command pattern detected" | tee -a "$LOG_FILE"
        return 1
    fi

    echo "$(date): COMMAND: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    local exit_code="$1"
    local error_msg="${2:-Command failed}"
    if [ "$exit_code" -ne 0 ]; then
        echo "ERROR: $error_msg (exit code: $exit_code)" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        cleanup_on_error
        exit "$exit_code"
```

```
    fi
}

# Function to validate AWS CLI is available and configured
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
        exit 1
    fi

    # Check AWS CLI version
    local aws_version
    aws_version=$(aws --version 2>&1 | cut -d' ' -f1 | cut -d'/' -f2)
    echo "AWS CLI version: $aws_version" >> "$LOG_FILE"

    if ! aws sts get-caller-identity --region "$AWS_REGION" &>/dev/null; then
        echo "ERROR: AWS CLI is not properly configured or credentials are invalid"
 | tee -a "$LOG_FILE"
        exit 1
    fi

    # Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/
null)
    if [[ ! "$account_id" =~ ^[0-9]{12}$ ]]; then
        echo "ERROR: Invalid AWS account ID" | tee -a "$LOG_FILE"
        exit 1
    fi
    echo "Authenticated as AWS Account: $account_id" | tee -a "$LOG_FILE"
}

# Function to validate CIDR blocks
validate_cidr() {
    local cidr="$1"
    if ! [[ "$cidr" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}$ ]]; then
        echo "ERROR: Invalid CIDR block format: $cidr" | tee -a "$LOG_FILE"
        return 1
    fi

    # Additional validation for IP octets
    local ip_part="${cidr%/*}"
    local mask_part="${cidr#*/}"
```

```
    IFS='.' read -r -a octets <<< "$ip_part"
    for octet in "${octets[@]}"; do
        if (( octet > 255 )); then
            echo "ERROR: Invalid octet value in CIDR: $cidr" | tee -a "$LOG_FILE"
            return 1
        fi
    done

    if (( mask_part > 32 || mask_part < 0 )); then
        echo "ERROR: Invalid CIDR mask value: $mask_part" | tee -a "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 "$LOG_FILE"

    # List created resources
    echo "Resources created:" | tee -a "$LOG_FILE"
    for resource in "${CREATED_RESOURCES[@]:-}"; do
        echo "- $resource" | tee -a "$LOG_FILE"
    done

    # Clean up in reverse order with retry logic
    for ((i=${#CLEANUP_COMMANDS[@]}-1; i>=0; i--)); do
        echo "Executing cleanup: ${CLEANUP_COMMANDS[$i]}" >> "$LOG_FILE"
        local retry_count=0
        while [ $retry_count -lt $MAX_RETRIES ]; do
            if eval "${CLEANUP_COMMANDS[$i]}" 2>&1 >> "$LOG_FILE"; then
                break
            else
                retry_count=$((retry_count + 1))
                if [ $retry_count -lt $MAX_RETRIES ]; then
                    echo "Cleanup command failed, retrying in ${RETRY_DELAY}s..." >>
 "$LOG_FILE"
                    sleep "$RETRY_DELAY"
                fi
            fi
        done
    done
```

```
}

# Array to store created resources and cleanup commands
declare -a CREATED_RESOURCES=()
declare -a CLEANUP_COMMANDS=()

# Trap errors and cleanup
trap cleanup_on_error EXIT

echo "Setting up VPC peering connection..."

# Validate AWS CLI
validate_aws_cli

# Check for existing VPCs
echo "Checking for existing VPCs..."
EXISTING_VPCS=$(aws ec2 describe-vpcs --region "$AWS_REGION" --query 'Vpcs[?
State==`available`].[VpcId,CidrBlock]' --output text 2>/dev/null || echo "")

if [ -z "$EXISTING_VPCS" ]; then
    echo "No existing VPCs found. Creating new VPCs..."
    CREATE_VPCS=true
else
    echo "Found existing VPCs:"
    echo "$EXISTING_VPCS"
    echo ""
    echo "Using existing VPCs..."
    CREATE_VPCS=false
    # Get the first two available VPCs
    VPC1_INFO=$(echo "$EXISTING_VPCS" | head -n 1)
    VPC2_INFO=$(echo "$EXISTING_VPCS" | head -n 2 | tail -n 1)

    if [ -z "$VPC2_INFO" ]; then
        echo "Only one VPC found. Creating a second VPC..."
        VPC1_ID=$(echo "$VPC1_INFO" | awk '{print $1}')
        VPC1_CIDR=$(echo "$VPC1_INFO" | awk '{print $2}')

        # Sanitize extracted values
        VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID format"
        VPC1_CIDR=$(sanitize_var "$VPC1_CIDR") || check_error 1 "Invalid VPC1_CIDR
 format"

        validate_cidr "$VPC1_CIDR" || check_error 1 "Invalid VPC1 CIDR"
        CREATE_VPC2_ONLY=true
```

```
    else
        VPC1_ID=$(echo "$VPC1_INFO" | awk '{print $1}')
        VPC1_CIDR=$(echo "$VPC1_INFO" | awk '{print $2}')
        VPC2_ID=$(echo "$VPC2_INFO" | awk '{print $1}')
        VPC2_CIDR=$(echo "$VPC2_INFO" | awk '{print $2}')

        # Sanitize extracted values
        VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID format"
        VPC1_CIDR=$(sanitize_var "$VPC1_CIDR") || check_error 1 "Invalid VPC1_CIDR
 format"
        VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID format"
        VPC2_CIDR=$(sanitize_var "$VPC2_CIDR") || check_error 1 "Invalid VPC2_CIDR
 format"

        validate_cidr "$VPC1_CIDR" || check_error 1 "Invalid VPC1 CIDR"
        validate_cidr "$VPC2_CIDR" || check_error 1 "Invalid VPC2 CIDR"
        CREATE_VPC2_ONLY=false
    fi
fi

# Create VPCs if needed
if [ "$CREATE_VPCS" = true ]; then
    echo "Creating VPC1..."
    VPC1_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.1.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC1"
    VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID returned"
    VPC1_CIDR="10.1.0.0/16"
    CREATED_RESOURCES+=("VPC1: $VPC1_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC1_ID'")
    echo "VPC1 created with ID: $VPC1_ID"

    echo "Creating VPC2..."
    VPC2_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.2.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC2"
    VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID returned"
    VPC2_CIDR="10.2.0.0/16"
    CREATED_RESOURCES+=("VPC2: $VPC2_ID")
```

```
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC2_ID'")
    echo "VPC2 created with ID: $VPC2_ID"

    # Wait for VPCs to be available
    echo "Waiting for VPCs to be available..."
    log_cmd "aws ec2 wait vpc-available --region '$AWS_REGION' --vpc-ids '$VPC1_ID'
 '$VPC2_ID'"
    check_error $? "Timeout waiting for VPCs to become available"

elif [ "$CREATE_VPC2_ONLY" = true ]; then
    echo "Creating VPC2..."
    VPC2_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.2.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC2"
    VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID returned"
    VPC2_CIDR="10.2.0.0/16"
    CREATED_RESOURCES+=("VPC2: $VPC2_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC2_ID'")
    echo "VPC2 created with ID: $VPC2_ID"

    # Wait for VPC2 to be available
    echo "Waiting for VPC2 to be available..."
    log_cmd "aws ec2 wait vpc-available --region '$AWS_REGION' --vpc-ids '$VPC2_ID'"
    check_error $? "Timeout waiting for VPC2 to become available"
fi

echo "Using the following VPCs:"
echo "VPC1: $VPC1_ID ($VPC1_CIDR)"
echo "VPC2: $VPC2_ID ($VPC2_CIDR)"

# Verify the VPCs exist and are available
echo "Verifying VPCs..."
log_cmd "aws ec2 describe-vpcs --region '$AWS_REGION' --vpc-ids '$VPC1_ID'
 '$VPC2_ID' --query 'Vpcs[*].[VpcId,State,CidrBlock]' --output table"
check_error $? "Failed to verify VPCs"

# Determine subnet CIDR blocks based on VPC CIDR blocks
VPC1_SUBNET_CIDR=$(echo "$VPC1_CIDR" | sed 's/0\.0\/16/1.0\/24/')
VPC2_SUBNET_CIDR=$(echo "$VPC2_CIDR" | sed 's/0\.0\/16/1.0\/24/')
```

```
# Sanitize subnet CIDR blocks
VPC1_SUBNET_CIDR=$(sanitize_var "$VPC1_SUBNET_CIDR") || check_error 1 "Invalid
 VPC1_SUBNET_CIDR format"
VPC2_SUBNET_CIDR=$(sanitize_var "$VPC2_SUBNET_CIDR") || check_error 1 "Invalid
 VPC2_SUBNET_CIDR format"

validate_cidr "$VPC1_SUBNET_CIDR" || check_error 1 "Invalid subnet CIDR for VPC1"
validate_cidr "$VPC2_SUBNET_CIDR" || check_error 1 "Invalid subnet CIDR for VPC2"

# Create subnets in both VPCs
echo "Creating subnet in VPC1..."
SUBNET1_ID=$(log_cmd "aws ec2 create-subnet --region '$AWS_REGION' --
vpc-id '$VPC1_ID' --cidr-block '$VPC1_SUBNET_CIDR' --tag-specifications
 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Peering-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'Subnet.SubnetId' --output text")
check_error $? "Failed to create subnet in VPC1"
SUBNET1_ID=$(sanitize_var "$SUBNET1_ID") || check_error 1 "Invalid SUBNET1_ID
 returned"
CREATED_RESOURCES+=("Subnet in VPC1: $SUBNET1_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET1_ID'")
echo "Subnet created in VPC1 with ID: $SUBNET1_ID (CIDR: $VPC1_SUBNET_CIDR)"

echo "Creating subnet in VPC2..."
SUBNET2_ID=$(log_cmd "aws ec2 create-subnet --region '$AWS_REGION' --
vpc-id '$VPC2_ID' --cidr-block '$VPC2_SUBNET_CIDR' --tag-specifications
 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Peering-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'Subnet.SubnetId' --output text")
check_error $? "Failed to create subnet in VPC2"
SUBNET2_ID=$(sanitize_var "$SUBNET2_ID") || check_error 1 "Invalid SUBNET2_ID
 returned"
CREATED_RESOURCES+=("Subnet in VPC2: $SUBNET2_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET2_ID'")
echo "Subnet created in VPC2 with ID: $SUBNET2_ID (CIDR: $VPC2_SUBNET_CIDR)"

# Create a VPC peering connection
echo "Creating VPC peering connection..."
PEERING_ID=$(log_cmd "aws ec2 create-vpc-peering-connection --region
 '$AWS_REGION' --vpc-id '$VPC1_ID' --peer-vpc-id '$VPC2_ID' --tag-specifications
 'ResourceType=vpc-peering-connection,Tags=[{Key=Name,Value=VPC1-VPC2-Peering},
```

```
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'VpcPeeringConnection.VpcPeeringConnectionId' --output text")
check_error $? "Failed to create VPC peering connection"
PEERING_ID=$(sanitize_var "$PEERING_ID") || check_error 1 "Invalid PEERING_ID
 returned"
CREATED_RESOURCES+=("VPC Peering Connection: $PEERING_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-vpc-peering-connection --region '$AWS_REGION' --
vpc-peering-connection-id '$PEERING_ID'")
echo "VPC Peering Connection created with ID: $PEERING_ID"

# Accept the VPC peering connection
echo "Accepting VPC peering connection..."
log_cmd "aws ec2 accept-vpc-peering-connection --region '$AWS_REGION' --vpc-peering-
connection-id '$PEERING_ID'"
check_error $? "Failed to accept VPC peering connection"
echo "VPC Peering Connection accepted"

# Wait for the peering connection to become active
echo "Waiting for peering connection to become active..."
log_cmd "aws ec2 wait vpc-peering-connection-exists --region '$AWS_REGION' --vpc-
peering-connection-ids '$PEERING_ID'"
check_error $? "Timeout waiting for peering connection to become active"

# Create a route table for VPC1
echo "Creating route table for VPC1..."
RTB1_ID=$(log_cmd "aws ec2 create-route-table --region '$AWS_REGION'
 --vpc-id '$VPC1_ID' --tag-specifications 'ResourceType=route-
table,Tags=[{Key=Name,Value=VPC1-RouteTable},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-peering}]' --query 'RouteTable.RouteTableId' --output text")
check_error $? "Failed to create route table for VPC1"
RTB1_ID=$(sanitize_var "$RTB1_ID") || check_error 1 "Invalid RTB1_ID returned"
CREATED_RESOURCES+=("Route Table for VPC1: $RTB1_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-route-table --region '$AWS_REGION' --route-table-
id '$RTB1_ID'")
echo "Route table created for VPC1 with ID: $RTB1_ID"

# Create a route from VPC1 to VPC2
echo "Creating route from VPC1 to VPC2..."
log_cmd "aws ec2 create-route --region '$AWS_REGION' --route-table-id '$RTB1_ID' --
destination-cidr-block '$VPC2_CIDR' --vpc-peering-connection-id '$PEERING_ID'"
check_error $? "Failed to create route from VPC1 to VPC2"
echo "Route created from VPC1 to VPC2"

# Associate the route table with the subnet in VPC1
```

```
echo "Associating route table with subnet in VPC1..."
RTB1_ASSOC_ID=$(log_cmd "aws ec2 associate-route-table --region '$AWS_REGION' --
route-table-id '$RTB1_ID' --subnet-id '$SUBNET1_ID' --query 'AssociationId' --output
 text")
check_error $? "Failed to associate route table with subnet in VPC1"
RTB1_ASSOC_ID=$(sanitize_var "$RTB1_ASSOC_ID") || check_error 1 "Invalid
 RTB1_ASSOC_ID returned"
CREATED_RESOURCES+=("Route Table Association for VPC1: $RTB1_ASSOC_ID")
CLEANUP_COMMANDS+=("aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB1_ASSOC_ID'")
echo "Route table associated with subnet in VPC1"

# Create a route table for VPC2
echo "Creating route table for VPC2..."
RTB2_ID=$(log_cmd "aws ec2 create-route-table --region '$AWS_REGION'
 --vpc-id '$VPC2_ID' --tag-specifications 'ResourceType=route-
table,Tags=[{Key=Name,Value=VPC2-RouteTable},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-peering}]' --query 'RouteTable.RouteTableId' --output text")
check_error $? "Failed to create route table for VPC2"
RTB2_ID=$(sanitize_var "$RTB2_ID") || check_error 1 "Invalid RTB2_ID returned"
CREATED_RESOURCES+=("Route Table for VPC2: $RTB2_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-route-table --region '$AWS_REGION' --route-table-
id '$RTB2_ID'")
echo "Route table created for VPC2 with ID: $RTB2_ID"

# Create a route from VPC2 to VPC1
echo "Creating route from VPC2 to VPC1..."
log_cmd "aws ec2 create-route --region '$AWS_REGION' --route-table-id '$RTB2_ID' --
destination-cidr-block '$VPC1_CIDR' --vpc-peering-connection-id '$PEERING_ID'"
check_error $? "Failed to create route from VPC2 to VPC1"
echo "Route created from VPC2 to VPC1"

# Associate the route table with the subnet in VPC2
echo "Associating route table with subnet in VPC2..."
RTB2_ASSOC_ID=$(log_cmd "aws ec2 associate-route-table --region '$AWS_REGION' --
route-table-id '$RTB2_ID' --subnet-id '$SUBNET2_ID' --query 'AssociationId' --output
 text")
check_error $? "Failed to associate route table with subnet in VPC2"
RTB2_ASSOC_ID=$(sanitize_var "$RTB2_ASSOC_ID") || check_error 1 "Invalid
 RTB2_ASSOC_ID returned"
CREATED_RESOURCES+=("Route Table Association for VPC2: $RTB2_ASSOC_ID")
CLEANUP_COMMANDS+=("aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB2_ASSOC_ID'")
echo "Route table associated with subnet in VPC2"
```

```
# Verify the VPC peering connection
echo "Verifying VPC peering connection..."
log_cmd "aws ec2 describe-vpc-peering-connections --region '$AWS_REGION' --
vpc-peering-connection-ids '$PEERING_ID' --query 'VpcPeeringConnections[0].
[VpcPeeringConnectionId,Status.Code,AccepterVpcInfo.VpcId,RequesterVpcInfo.VpcId]'
 --output table"
check_error $? "Failed to verify VPC peering connection"
echo "VPC peering connection verified"

# Display summary of created resources
echo ""
echo "================================================"
echo "SUMMARY OF CREATED RESOURCES"
echo "================================================"
echo "VPC1 ID: $VPC1_ID"
echo "VPC1 CIDR: $VPC1_CIDR"
echo "Subnet1 ID: $SUBNET1_ID (CIDR: $VPC1_SUBNET_CIDR)"
echo "VPC2 ID: $VPC2_ID"
echo "VPC2 CIDR: $VPC2_CIDR"
echo "Subnet2 ID: $SUBNET2_ID (CIDR: $VPC2_SUBNET_CIDR)"
echo "Peering Connection ID: $PEERING_ID"
echo "Route Table 1 ID: $RTB1_ID"
echo "Route Table 1 Association ID: $RTB1_ASSOC_ID"
echo "Route Table 2 ID: $RTB2_ID"
echo "Route Table 2 Association ID: $RTB2_ASSOC_ID"
echo ""
echo "Created resources:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "================================================"
echo ""

# Test connectivity (optional)
echo "================================================"
echo "CONNECTIVITY TEST"
echo "================================================"
echo "To test connectivity between VPCs, you would need to:"
echo "1. Launch EC2 instances in each subnet"
echo "2. Configure security groups to allow traffic"
echo "3. Test ping or other network connectivity"
echo ""
```

```
# Automatic cleanup
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Auto-confirming cleanup of all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Clean up in reverse order
    echo "Disassociating route table from subnet in VPC2..."
    log_cmd "aws ec2 disassociate-route-table --region '$AWS_REGION' --association-
id '$RTB2_ASSOC_ID'" || true

    echo "Disassociating route table from subnet in VPC1..."
    log_cmd "aws ec2 disassociate-route-table --region '$AWS_REGION' --association-
id '$RTB1_ASSOC_ID'" || true

    echo "Deleting route table for VPC2..."
    log_cmd "aws ec2 delete-route-table --region '$AWS_REGION' --route-table-id
 '$RTB2_ID'" || true

    echo "Deleting route table for VPC1..."
    log_cmd "aws ec2 delete-route-table --region '$AWS_REGION' --route-table-id
 '$RTB1_ID'" || true

    echo "Deleting VPC peering connection..."
    log_cmd "aws ec2 delete-vpc-peering-connection --region '$AWS_REGION' --vpc-
peering-connection-id '$PEERING_ID'" || true

    echo "Deleting subnet in VPC2..."
    log_cmd "aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id '$SUBNET2_ID'"
 || true

    echo "Deleting subnet in VPC1..."
    log_cmd "aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id '$SUBNET1_ID'"
 || true

    # Delete VPCs if they were created by this script
    if [ "$CREATE_VPCS" = true ]; then
        echo "Deleting VPC2..."
```

```
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC2_ID'" ||
 true

        echo "Deleting VPC1..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC1_ID'" ||
 true
    elif [ "$CREATE_VPC2_ONLY" = true ]; then
        echo "Deleting VPC2..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC2_ID'" ||
 true
    fi

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo ""
    echo "To manually clean up later, you can delete resources in this order:"
    echo "1. Route table associations"
    echo "2. Route tables"
    echo "3. VPC peering connection"
    echo "4. Subnets"
    if [ "$CREATE_VPCS" = true ] || [ "$CREATE_VPC2_ONLY" = true ]; then
        echo "5. VPCs (if created by this script)"
    fi
fi

echo "Script execution completed. See $LOG_FILE for detailed logs."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AcceptVpcPeeringConnection
  - AssociateRouteTable
  - CreateRoute
  - CreateRouteTable
  - CreateSubnet
  - CreateVpc
  - CreateVpcPeeringConnection
  - DeleteRouteTable
  - DeleteSubnet
  - DeleteVpc

- DeleteVpcPeeringConnection
- DescribeVpcPeeringConnections
- DescribeVpcs
- DisassociateRouteTable
- Wait

Exemplos do Amazon ECR usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon ECR.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos de registros de contêineres

O exemplo de código a seguir mostra como:

- Criar uma imagem do Docker.
- Criar um repositório do Amazon ECR.
- Excluir recursos.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "====================================================="

# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
 for installation instructions."
    exit 1
fi

# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure' to
 set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
```

```
    echo "ERROR: Docker is not installed. Please install Docker before running this
 script."
    echo "Visit https://docs.docker.com/get-docker/ for installation instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"

    echo "====================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "====================================================="

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "====================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
```

```
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force || echo
 "Failed to delete repository, it may not exist or may have already been deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local image,
 it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove tagged
 image, it may not exist or may have already been deleted."
    fi

    echo "Cleanup completed."
    echo "==================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
```

```
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build Docker
 image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list Docker
 images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"

# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials are
 configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"

# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
```

```
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --tags
 key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-registry-
gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username AWS --
password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" || handle_error
 "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"

# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image to
 ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged image"

# Pull the image with timeout
```

```
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "====================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "====================================================="

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi
```

```
    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "====================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "====================================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - BatchDeleteImage
  - CreateRepository
  - DeleteRepository
  - GetCallerIdentity
  - GetLoginPassword

Exemplos do Amazon ECS usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon ECS.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
```

```
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
```

```
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
```

```
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
```

```
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
```

```
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
```

```
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
```

```
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
```

```
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
```

```
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
```

```
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
```

```
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json
```

```
    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
```

```
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters

- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um cluster.
- Criar uma definição de tarefa.
- Criar o serviço.
- Fazer a limpeza.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "==========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "==========================================="

    local output
    local exit_code
```

```
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|AccessDenied
\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
```

```
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security group
 $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
```

```
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "CLEANUP PROCESS"
```

```
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]]; then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
 anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]]; then
            echo ""
            echo "Step 3: Deleting cluster..."
```

```
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete security
 group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix "$TASK_FAMILY"
 --query "taskDefinitionArns" --output text 2>/dev/null || echo "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "============================================"
        echo "CLEANUP COMPLETED"
        echo "============================================"
        echo "All resources have been cleaned up successfully!"
```

```
    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME --
service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo "6. Deregister task definitions: aws ecs list-task-definitions --
family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --task-
definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi
```

```
EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --assume-
role-policy-document file://trust-policy.json" "Create ECS task execution role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
 "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json
```

```
    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-ecs-
fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
```

```
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-json
 file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
```

```
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --group-
name '$SECURITY_GROUP_NAME' --description 'Security group for ECS Fargate tutorial
 - HTTP access' --vpc-id '$VPC_ID' --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-fargate-
linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"' |
 head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
```

```
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP inbound
 rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-type
 FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets\":[\"$(echo
 "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups\":[\"$SECURITY_GROUP_ID
\"],\"assignPublicIp\":\"ENABLED\"}}' --tags key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"
```

```
CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN" --
query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --output
 text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID" --
query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
```

```
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetRole
  - ListTaskDefinitions
  - ListTasks
  - RegisterTaskDefinition
  - UpdateService
  - Wait

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
```

```
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
```

```
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
"Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
```

```
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
```

```
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
```

```
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
```

```
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."
```

```
    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
```

```
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
```

```
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
```

```
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}
```

```
# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
```

```
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
```

```
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "==========================================="
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair

- CreateRole
- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteKeyPair
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Exemplos do Amazon EKS usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon EKS.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos de clusters gerenciados de kubernetes

O exemplo de código a seguir mostra como:

- Criar uma VPC para um cluster do EKS.
- Criar perfis do IAM para um cluster do EKS.
- Criação do seu cluster de EKS
- Configurar o kubectl para se comunicar com o cluster.
- Crie um grupo de nós gerenciados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the AWS
 CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your operating
 system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
```

```
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/
kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
```

```
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --nodegroup-
name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
```

```
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources; exit
 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."
```

```
# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
```

```
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
```

```
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-kubeconfig
 --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn" --
output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })
```

```
echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl get
 nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or the
 nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
```

```
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl get
 all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
```

```
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --cluster-
name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --stack-
name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity

- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

Elastic Load Balancing - exemplos da versão 2 usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Elastic Load Balancing - Versão 2.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução ao balanceamento de carga

O exemplo de código a seguir mostra como:

- Criar um Application Load Balancer.
- Criar um grupo de destino.
- Criar um receptor.
- Verificar sua configuração.
- Adicionar um receptor HTTPS (opcional).
- Adicionar roteamento baseado em caminho (opcional).
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Elastic Load Balancing Getting Started Script - v2
# This script creates an Application Load Balancer with HTTP listener and target
 group

set -euo pipefail

# Set up logging
LOG_FILE="elb-script-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Elastic Load Balancing setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check AWS CLI command success
check_command() {
    local output="$1"
    if [[ -z "$output" ]] || [[ "$output" == "None" ]]; then
        handle_error "AWS CLI command returned empty or invalid output"
    fi
}

# Function to validate ARN format
validate_arn() {
    local arn="$1"
```

```
    if [[ ! "$arn" =~ ^arn:aws:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$ ]]; then
        handle_error "Invalid ARN format: $arn"
    fi
}

# Function to validate security group ID
validate_security_group_id() {
    local sg_id="$1"
    if [[ ! "$sg_id" =~ ^sg-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid security group ID format: $sg_id"
    fi
}

# Function to validate VPC ID
validate_vpc_id() {
    local vpc_id="$1"
    if [[ ! "$vpc_id" =~ ^vpc-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid VPC ID format: $vpc_id"
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    if [ -n "${LISTENER_ARN:-}" ]; then
        echo "Deleting listener: $LISTENER_ARN"
        aws elbv2 delete-listener --listener-arn "$LISTENER_ARN" 2>/dev/null || true
    fi

    if [ -n "${LOAD_BALANCER_ARN:-}" ]; then
        echo "Deleting load balancer: $LOAD_BALANCER_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LOAD_BALANCER_ARN" 2>/
dev/null || true

        # Wait for load balancer to be deleted before deleting target group
        echo "Waiting for load balancer to be deleted..."
        aws elbv2 wait load-balancers-deleted --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null || true
    fi

    if [ -n "${TARGET_GROUP_ARN:-}" ]; then
        echo "Deleting target group: $TARGET_GROUP_ARN"
```

```
        aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/
null || true
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting 30 seconds before deleting security group to ensure all
 dependencies are removed..."
        sleep 30

        echo "Deleting security group: $SECURITY_GROUP_ID"
        local sg_delete_output
        sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)

        local retry_count=0
        local max_retries=5
        while echo "$sg_delete_output" | grep -i "DependencyViolation" > /dev/null
 && [ $retry_count -lt $max_retries ]; do
            retry_count=$((retry_count+1))
            echo "Security group still has dependencies. Retrying in 30 seconds...
 (Attempt $retry_count of $max_retries)"
            sleep 30
            sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)
        done

        if echo "$sg_delete_output" | grep -i "error" > /dev/null; then
            echo "WARNING: Could not delete security group: $SECURITY_GROUP_ID" >&2
            echo "You may need to delete it manually using: aws ec2 delete-security-
group --group-id $SECURITY_GROUP_ID"
        else
            echo "Security group deleted successfully."
        fi
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
RESOURCE_PREFIX="elb-demo-${RANDOM_ID}"

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi
```

```
# Step 1: Verify AWS CLI support for Elastic Load Balancing
echo "Verifying AWS CLI support for Elastic Load Balancing..."
if ! aws elbv2 help > /dev/null 2>&1; then
    handle_error "AWS CLI does not support elbv2 commands. Please update your AWS
 CLI."
fi

# Step 2: Get VPC ID and subnet information
echo "Retrieving VPC information..."
VPC_INFO=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 "Vpcs[0].VpcId" --output text 2>/dev/null || echo "")
check_command "$VPC_INFO"
VPC_ID="$VPC_INFO"
validate_vpc_id "$VPC_ID"
echo "Using VPC: $VPC_ID"

# Get two subnets from different Availability Zones
echo "Retrieving subnet information..."
SUBNET_INFO=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[0:2].SubnetId" --output text 2>/dev/null || echo "")
check_command "$SUBNET_INFO"

# Convert space-separated list to array
read -r -a SUBNETS <<< "$SUBNET_INFO"
if [ ${#SUBNETS[@]} -lt 2 ]; then
    handle_error "Need at least 2 subnets in different Availability Zones. Found:
 ${#SUBNETS[@]}"
fi

echo "Using subnets: ${SUBNETS[0]} and ${SUBNETS[1]}"

# Step 3: Create a security group for the load balancer
echo "Creating security group for the load balancer..."
SG_INFO=$(aws ec2 create-security-group \
    --group-name "${RESOURCE_PREFIX}-sg" \
    --description "Security group for ELB demo" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=elastic-load-balancing-gs}]' \
    --query "GroupId" --output text 2>/dev/null || echo "")
check_command "$SG_INFO"
SECURITY_GROUP_ID="$SG_INFO"
validate_security_group_id "$SECURITY_GROUP_ID"
```

```
echo "Created security group: $SECURITY_GROUP_ID"

# Add inbound rule to allow HTTP traffic
echo "Adding inbound rule to allow HTTP traffic..."
aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 80 \
    --cidr "0.0.0.0/0" > /dev/null 2>&1 || handle_error "Failed to authorize
 security group ingress"

echo "WARNING: Security group allows HTTP from 0.0.0.0/0. In production, restrict to
 specific IP addresses." >&2

# Step 4: Create the load balancer
echo "Creating Application Load Balancer..."
LB_INFO=$(aws elbv2 create-load-balancer \
    --name "${RESOURCE_PREFIX}-lb" \
    --subnets "${SUBNETS[0]}" "${SUBNETS[1]}" \
    --security-groups "$SECURITY_GROUP_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "LoadBalancers[0].LoadBalancerArn" --output text 2>/dev/null || echo "")
check_command "$LB_INFO"
LOAD_BALANCER_ARN="$LB_INFO"
validate_arn "$LOAD_BALANCER_ARN"
echo "Created load balancer: $LOAD_BALANCER_ARN"

# Wait for the load balancer to be active
echo "Waiting for load balancer to become active..."
if ! aws elbv2 wait load-balancer-available --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null; then
    handle_error "Load balancer did not reach active state within timeout period"
fi

# Step 5: Create a target group
echo "Creating target group..."
TG_INFO=$(aws elbv2 create-target-group \
    --name "${RESOURCE_PREFIX}-targets" \
    --protocol HTTP \
    --port 80 \
    --vpc-id "$VPC_ID" \
    --target-type instance \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "TargetGroups[0].TargetGroupArn" --output text 2>/dev/null || echo "")
check_command "$TG_INFO"
TARGET_GROUP_ARN="$TG_INFO"
validate_arn "$TARGET_GROUP_ARN"
echo "Created target group: $TARGET_GROUP_ARN"

# Step 6: Find EC2 instances to register as targets
echo "Looking for available EC2 instances to register as targets..."
INSTANCES=$(aws ec2 describe-instances \
    --filters "Name=vpc-id,Values=$VPC_ID" "Name=instance-state-name,Values=running"
 \
    --query "Reservations[*].Instances[*].InstanceId" --output text 2>/dev/null ||
 echo "")

# Convert space-separated list to array
read -r -a INSTANCE_IDS <<< "$INSTANCES"

if [ ${#INSTANCE_IDS[@]} -eq 0 ]; then
    echo "No running instances found in VPC $VPC_ID."
    echo "You will need to register targets manually after launching instances."
else
    # Step 7: Register targets with the target group (up to 2 instances)
    echo "Registering targets with the target group..."
    target_args=()
    for i in "${!INSTANCE_IDS[@]}"; do
        if [ "$i" -lt 2 ]; then
            target_args+=("Id=${INSTANCE_IDS[$i]}")
        fi
    done

    if [ ${#target_args[@]} -gt 0 ]; then
        if aws elbv2 register-targets \
            --target-group-arn "$TARGET_GROUP_ARN" \
            --targets "${target_args[@]}" 2>/dev/null; then
            echo "Registered instances: ${target_args[*]}"
        else
            handle_error "Failed to register targets"
        fi
    fi
fi

# Step 8: Create a listener
```

```
echo "Creating HTTP listener..."
LISTENER_INFO=$(aws elbv2 create-listener \
    --load-balancer-arn "$LOAD_BALANCER_ARN" \
    --protocol HTTP \
    --port 80 \
    --default-actions Type=forward,TargetGroupArn="$TARGET_GROUP_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "Listeners[0].ListenerArn" --output text 2>/dev/null || echo "")
check_command "$LISTENER_INFO"
LISTENER_ARN="$LISTENER_INFO"
validate_arn "$LISTENER_ARN"
echo "Created listener: $LISTENER_ARN"

# Step 9: Verify target health
echo "Verifying target health..."
aws elbv2 describe-target-health --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/null
 || true

# Display load balancer DNS name
LB_DNS=$(aws elbv2 describe-load-balancers \
    --load-balancer-arns "$LOAD_BALANCER_ARN" \
    --query "LoadBalancers[0].DNSName" --output text 2>/dev/null || echo "")
check_command "$LB_DNS"

echo ""
echo "================================================"
echo "SETUP COMPLETE"
echo "================================================"
echo "Load Balancer DNS Name: $LB_DNS"
echo ""
echo "Resources created:"
echo "- Load Balancer: $LOAD_BALANCER_ARN"
echo "- Target Group: $TARGET_GROUP_ARN"
echo "- Listener: $LISTENER_ARN"
echo "- Security Group: $SECURITY_GROUP_ID"
echo ""

# Prompt for cleanup confirmation
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
read -p "Do you want to clean up all created resources? (y/n): " -r CLEANUP_CHOICE
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
    echo "To clean up later, run the following commands:"
    echo "aws elbv2 delete-listener --listener-arn $LISTENER_ARN"
    echo "aws elbv2 delete-load-balancer --load-balancer-arn $LOAD_BALANCER_ARN"
    echo "aws elbv2 wait load-balancers-deleted --load-balancer-arns
 $LOAD_BALANCER_ARN"
    echo "aws elbv2 delete-target-group --target-group-arn $TARGET_GROUP_ARN"
    echo "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateListener
  - CreateLoadBalancer
  - CreateSecurityGroup
  - CreateTargetGroup
  - DeleteListener
  - DeleteLoadBalancer
  - DeleteSecurityGroup
  - DeleteTargetGroup
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetHealth
  - DescribeVpcs
  - Help
  - RegisterTargets
  - Wait

ElastiCache exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com ElastiCache.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução ao armazenamento em cache em memória

O exemplo de código a seguir mostra como:

- Configurar grupo de segurança para ElastiCache acesso
- Criar um cache sem servidor do Valkey.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ElastiCache Getting Started Script
# This script creates a Valkey serverless cache, configures security groups,
# and demonstrates how to connect to and use the cache.

set -uo pipefail
```

```
# Set up logging
LOG_FILE="elasticache_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting ElastiCache tutorial script. Logging to $LOG_FILE"
echo "============================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${CACHE_NAME:-}" ]; then
        echo "- ElastiCache serverless cache: $CACHE_NAME"
    fi
    if [ -n "${SG_RULE_6379:-}" ] || [ -n "${SG_RULE_6380:-}" ]; then
        echo "- Security group rules for ports 6379 and 6380"
    fi
    echo "Please clean up these resources manually."
    exit 1
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CACHE_NAME="valkey-cache-${RANDOM_ID}"

echo "Using cache name: $CACHE_NAME"

# Step 1: Set up security group for ElastiCache access
echo "Step 1: Setting up security group for ElastiCache access..."

# Get default security group ID
echo "Getting default security group ID..."
SG_ID=$(aws ec2 describe-security-groups \
```

```
  --filters Name=group-name,Values=default \
  --query "SecurityGroups[0].GroupId" \
  --output text 2>/dev/null || echo "")

if [[ -z "$SG_ID" || "$SG_ID" == "None" ]]; then
    handle_error "Failed to get default security group ID"
fi

echo "Default security group ID: $SG_ID"

# Add inbound rule for port 6379
echo "Adding inbound rule for port 6379..."
SG_RULE_6379=""
if SG_RULE_6379=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6379 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379" ==
 *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    fi
else
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379" ==
 *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    else
        handle_error "Failed to add security group rule for port 6379:
 $SG_RULE_6379"
    fi
fi

# Add inbound rule for port 6380
echo "Adding inbound rule for port 6380..."
SG_RULE_6380=""
if SG_RULE_6380=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6380 \
  --cidr 0.0.0.0/0 \
```

```
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380" ==
 *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    fi
else
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380" ==
 *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    else
        handle_error "Failed to add security group rule for port 6380:
 $SG_RULE_6380"
    fi
fi

echo "Security group rules added successfully."
echo ""
echo "##  SECURITY WARNING: The security group rules created allow access from any
 IP address (0.0.0.0/0)."
echo "This is NOT RECOMMENDED for production environments. For production,"
echo "you should restrict access to specific IP ranges or security groups."
echo "Update the CIDR blocks in this script before using in production."
echo ""

# Step 2: Create a Valkey serverless cache
echo "Step 2: Creating Valkey serverless cache..."
if ! CREATE_RESULT=$(aws elasticache create-serverless-cache \
  --serverless-cache-name "$CACHE_NAME" \
  --engine valkey \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-elasticache-gs 2>&1);
 then
    handle_error "Failed to create serverless cache: $CREATE_RESULT"
fi

echo "Cache creation initiated. Waiting for cache to become available..."

# Step 3: Check the status of the cache creation
echo "Step 3: Checking cache status..."

# Wait for the cache to become active
MAX_ATTEMPTS=30
```

```
ATTEMPT=1
CACHE_STATUS=""

while [[ $ATTEMPT -le $MAX_ATTEMPTS ]]; do
    echo "Checking cache status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    if ! DESCRIBE_RESULT=$(aws elasticache describe-serverless-caches \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        handle_error "Failed to describe serverless cache: $DESCRIBE_RESULT"
    fi

    # Extract status using jq for reliable JSON parsing
    if command -v jq &> /dev/null; then
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | jq -r
 '.ServerlessCaches[0].Status // "UNKNOWN"' 2>/dev/null || echo "")
    else
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | grep -o '"Status": "[^"]*"' | awk -
F'"' '{print $4}' | head -n 1)
    fi

    echo "Current status: $CACHE_STATUS"

    if [[ "${CACHE_STATUS,,}" == "available" ]]; then
        echo "Cache is now available!"
        break
    elif [[ "${CACHE_STATUS,,}" == "create-failed" ]]; then
        handle_error "Cache creation failed. Please check the AWS console for
 details."
    fi

    if [[ $ATTEMPT -lt $MAX_ATTEMPTS ]]; then
        echo "Waiting 30 seconds..."
        sleep 30
    fi

    ((ATTEMPT++))
done

if [[ "${CACHE_STATUS,,}" != "available" ]]; then
    handle_error "Cache did not become available within the expected time. Last
 status: $CACHE_STATUS"
fi

# Step 4: Find your cache endpoint
```

```
echo "Step 4: Getting cache endpoint..."
if ! ENDPOINT=$(aws elasticache describe-serverless-caches \
  --serverless-cache-name "$CACHE_NAME" \
  --query "ServerlessCaches[0].Endpoint.Address" \
  --output text 2>&1); then
    handle_error "Failed to get cache endpoint: $ENDPOINT"
fi

if [[ -z "$ENDPOINT" || "$ENDPOINT" == "None" ]]; then
    handle_error "Failed to get cache endpoint"
fi

echo "Cache endpoint: $ENDPOINT"

# Step 5: Instructions for connecting to the cache
echo ""
echo "============================================================"
echo "Your Valkey serverless cache has been successfully created!"
echo "Cache Name: $CACHE_NAME"
echo "Endpoint: $ENDPOINT"
echo "============================================================"
echo ""
echo "To connect to your cache from an EC2 instance, follow these steps:"
echo ""
echo "1. Install valkey-cli on your EC2 instance:"
echo "   sudo amazon-linux-extras install epel -y"
echo "   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y"
echo "   wget https://github.com/valkey-io/valkey/archive/refs/tags/8.0.0.tar.gz"
echo "   tar xvzf 8.0.0.tar.gz"
echo "   cd valkey-8.0.0"
echo "   make BUILD_TLS=yes"
echo ""
echo "2. Connect to your cache using valkey-cli:"
echo "   src/valkey-cli -h $ENDPOINT --tls -p 6379"
echo ""
echo "3. Once connected, you can run commands like:"
echo "   set mykey \"Hello ElastiCache\""
echo "   get mykey"
echo ""

# Auto-confirm cleanup
echo ""
echo "=========================================="
echo "CLEANUP CONFIRMATION"
```

```
echo "============================================"
echo "Resources created:"
echo "- ElastiCache serverless cache: $CACHE_NAME"
if [[ "${SG_RULE_6379:-}" != "existing" ]] || [[ "${SG_RULE_6380:-}" !=
 "existing" ]]; then
    echo "- Security group rules for ports 6379 and 6380"
fi
echo ""
echo "Proceeding with cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Step 7: Delete the cache
    echo "Deleting serverless cache $CACHE_NAME..."
    if ! DELETE_RESULT=$(aws elasticache delete-serverless-cache \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        echo "WARNING: Failed to delete serverless cache: $DELETE_RESULT"
        echo "Please delete the cache manually from the AWS console."
    else
        echo "Cache deletion initiated. This may take several minutes to complete."
    fi

    # Only attempt to remove security group rules if we created them
    if [[ "${SG_RULE_6379:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6379..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6379 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6379"
        fi
    fi

    if [[ "${SG_RULE_6380:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6380..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6380 \
          --cidr 0.0.0.0/0 2>&1; then
```

```
            echo "WARNING: Failed to remove security group rule for port 6380"
        fi
    fi

    echo "Cleanup completed."
fi

echo ""
echo "Script completed. See $LOG_FILE for the full log."
echo "============================================================"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateServerlessCache
  - DeleteServerlessCache
  - DescribeSecurityGroups
  - DescribeServerlessCaches
  - RevokeSecurityGroupIngress

Exemplos do Amazon EMR usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon EMR.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução aos clusters de processamento de big data

O exemplo de código a seguir mostra como:

- Criar um par de chaves do EC2.
- Configurar o armazenamento e preparar uma aplicação.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
```

```
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "==========================================="
    echo "CLEANUP IN PROGRESS"
    echo "==========================================="
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null || true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ]; then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi
```

```
# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-configuration
 Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 || true
```

```
# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to find
 the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as 's3://
emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-tutorial-
bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
```

```
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample data"
echo "Sample data downloaded and extracted successfully."
```

```
# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output text
 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
```

```
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout period"
        fi
```

```
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"
```

```
# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID"
 --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv" |
 head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv" |
 head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
```

```
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb
  - Rb
  - Rm
  - ssh
  - TerminateClusters
  - Wait

OpenSearch Exemplos de serviços usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com OpenSearch Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos dos mecanismos de pesquisa e analytics

O exemplo de código a seguir mostra como:

- Criar um domínio OpenSearch de serviço
- Carregar dados em seu domínio.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md
```

```
# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Track if domain was successfully created
DOMAIN_CREATED=false
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."
```

```
    if [[ "$DOMAIN_CREATED" == "true" ]]; then
        echo "Checking if domain $DOMAIN_NAME exists before attempting to delete..."

        # Check if domain exists before trying to delete
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/null
 2>&1; then
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
}

# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
```

```
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
# This policy restricts access to specific actions and should be further restricted
 to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack creation
 failed"

DOMAIN_CREATED=true
echo "Domain created successfully via CloudFormation."
```

```
# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120

# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo "No";
 fi)"

# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
```

```
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel", "Malahide,
 Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas", "Stellate, Nick",
 "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese", "Spielbauer, Bruce",
 "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett, Charlie"], "title": "U.S.
 Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955, "actor":
 ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal", "Backus, Jim",
 "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen, Corey", "Birch, Paul",
 "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh, Nelson", "Williams,
 Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira", "McMahon, David",
 "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"

    # Test 1: Basic authentication with root endpoint
```

```
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)

    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }

    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d:
-f2)

        if is_success_code "$HEALTH_HTTP_CODE"; then
```

```
            echo "# Cluster health authentication test successful! (HTTP
 $HEALTH_HTTP_CODE)"
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
            echo "Cluster health authentication also failed with HTTP code:
 $HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
 then
                echo "Error: Request is being treated as anonymous (authentication
 not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential backoff..."

            # Retry authentication test with exponential backoff
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT+1))/
$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)

                echo "Retry result (HTTP $HTTP_CODE):"
                echo "$AUTH_TEST_RESULT"
```

```
                if is_success_code "$HTTP_CODE"; then
                    echo "# Authentication test successful after retry! (HTTP
$HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" |
cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
(HTTP $HEALTH_HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding with
data operations."

        # Upload single document (matches tutorial exactly)
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
```

```
            --netrc-file "$CREDENTIALS_FILE" \
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
'"result"'; then
            echo "# Single document uploaded successfully! (HTTP $UPLOAD_HTTP_CODE)"
        else
            echo "# Warning: Single document upload may have failed (HTTP
$UPLOAD_HTTP_CODE)"
        fi

        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
'"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
$BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
        echo "Waiting for documents to be indexed..."
        sleep 5

        # Step 3: Search Documents (matches tutorial exactly)
```

```
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
'"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "==========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "==========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."

        # Debug domain configuration
```

```
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-name "$DOMAIN_NAME"
 --query 'DomainStatus.{AdvancedSecurityOptions: AdvancedSecurityOptions,
 AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi

# Display OpenSearch Dashboards URL (matches tutorial)
echo ""
```

```
echo "============================================"
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================"
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
```

```
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
echo "Cleaning up temporary files..."

# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - GetCallerIdentity

AWS FIS exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS FIS.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
```

```
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
```

```
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}
```

```
# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
```

```
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
```

```
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
```

```
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"
```

```
    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
```

```
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
```

```
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
```

```
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
```

```
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
```

```
DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

# Delete CloudWatch alarm
echo "Deleting CloudWatch alarm $ALARM_NAME..."
DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

# Terminate EC2 instance
echo "Terminating EC2 instance $INSTANCE_ID..."
TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
echo "Waiting for instance to terminate..."
aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

# Clean up IAM resources
echo "Removing role from instance profile..."
REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

echo "Deleting instance profile..."
DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

echo "Deleting FIS role policy..."
DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME")
check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

echo "Detaching policy from EC2 role..."
DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

echo "Deleting FIS role..."
DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"
```

```
    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment

- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

AWS Glue exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS Glue.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos do catálogo de dados

O exemplo de código a seguir mostra como:

- Criar um banco de dados.
- Criar uma tabela.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Glue Data Catalog Tutorial Script
# This script demonstrates how to create and manage AWS Glue Data Catalog resources
 using the AWS CLI
# Cost improvements: Reduced API calls, optimized queries, eliminated redundant
 operations
# Reliability improvements: Enhanced error handling, input validation, resource
 tracking

set -euo pipefail

# Setup logging
LOG_FILE="glue-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Glue Data Catalog tutorial script at $(date)"
echo "All operations will be logged to $LOG_FILE"

# Generate a unique identifier for resource names
UNIQUE_ID=$(openssl rand -hex 4)
DB_NAME="tutorial-db-${UNIQUE_ID}"
TABLE_NAME="flights-data-${UNIQUE_ID}"
TABLE_INPUT_FILE="table-input-${UNIQUE_ID}.json"

# Track created resources
declare -a CREATED_RESOURCES=()

# Set default region if not provided
AWS_REGION="${AWS_REGION:-us-east-1}"

# Flag to track if database was successfully created
DATABASE_CREATED=false
```

```
# Trap to ensure cleanup on exit
trap cleanup_resources EXIT

# Function to check command status
check_status() {
    if [ $? -ne 0 ]; then
        echo "ERROR: $1 failed." >&2
        exit 1
    fi
}

# Function to cleanup resources
cleanup_resources() {
    local exit_code=$?
    echo "Attempting to clean up resources..."

    # Delete resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource=${CREATED_RESOURCES[$i]}
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_name=$(echo "$resource" | cut -d':' -f2)

        echo "Deleting $resource_type: $resource_name"

        case $resource_type in
            "table")
                if [ "$DATABASE_CREATED" = true ]; then
                    aws glue delete-table \
                        --database-name "$DB_NAME" \
                        --name "$resource_name" \
                        --region "$AWS_REGION" \
                        2>/dev/null || echo "Warning: Failed to delete table
 $resource_name"
                fi
                ;;
            "database")
                aws glue delete-database \
                    --name "$resource_name" \
                    --region "$AWS_REGION" \
                    2>/dev/null || echo "Warning: Failed to delete database
 $resource_name"
                ;;
            *)
                echo "Unknown resource type: $resource_type" >&2
```

```
                ;;
        esac
    done

    # Clean up temporary files securely
    if [ -f "$TABLE_INPUT_FILE" ]; then
        if command -v shred &> /dev/null; then
            shred -vfz -n 3 "$TABLE_INPUT_FILE" 2>/dev/null || rm -f
 "$TABLE_INPUT_FILE"
        else
            rm -f "$TABLE_INPUT_FILE"
        fi
    fi

    echo "Cleanup completed."
    exit $exit_code
}

# Function to validate prerequisites
validate_prerequisites() {
    # Validate AWS CLI is available
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" >&2
        exit 1
    fi

    # Validate AWS CLI version
    local AWS_CLI_VERSION
    AWS_CLI_VERSION=$(aws --version 2>&1 | cut -d' ' -f1 | cut -d'/' -f2 | cut -d'.'
 -f1)
    if [ "$AWS_CLI_VERSION" -lt 1 ]; then
        echo "ERROR: AWS CLI is required" >&2
        exit 1
    fi

    # Validate jq is available for JSON validation
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH" >&2
        exit 1
    fi

    # Validate AWS credentials and get account identity in single call (cost
 optimization)
    local CALLER_IDENTITY
```

```
    CALLER_IDENTITY=$(aws sts get-caller-identity --region "$AWS_REGION" --query
 'Account' --output text 2>/dev/null) || {
        echo "ERROR: Failed to get AWS caller identity. Check credentials and
 permissions." >&2
        exit 1
    }

    if [ -z "$CALLER_IDENTITY" ] || [ "$CALLER_IDENTITY" == "None" ]; then
        echo "ERROR: Unable to determine AWS account identity" >&2
        exit 1
    fi
    echo "Using AWS Account: $CALLER_IDENTITY"
    echo "Using Region: $AWS_REGION"
}

# Function to create database with verification
create_database() {
    echo "Step 1: Creating a database named $DB_NAME"

    if ! aws glue create-database \
        --database-input "Name=$DB_NAME,Description=Database for AWS Glue tutorial"
 \
        --region "$AWS_REGION" \
        --output json > /dev/null 2>&1; then
        echo "ERROR: Failed to create database $DB_NAME" >&2
        exit 1
    fi

    ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
    aws glue tag-resource \
        --resource-arn "arn:aws:glue:${AWS_REGION}:${ACCOUNT_ID}:database/
${DB_NAME}" \
        --tags-to-add Key=project,Value=doc-smith Key=tutorial,Value=glue-gs \
        --region "$AWS_REGION" \
        2>/dev/null || true

    DATABASE_CREATED=true
    CREATED_RESOURCES+=("database:$DB_NAME")
    echo "Database $DB_NAME created successfully."
}

# Function to prepare table input JSON
prepare_table_input() {
    # Create a temporary JSON file for table input with restricted permissions
```

```
    if ! touch "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Failed to create temporary file $TABLE_INPUT_FILE" >&2
        exit 1
    fi

    if ! chmod 600 "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Failed to set permissions on $TABLE_INPUT_FILE" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    cat > "$TABLE_INPUT_FILE" << 'EOF'
{
  "Name": "TABLE_NAME_PLACEHOLDER",
  "StorageDescriptor": {
    "Columns": [
      {
        "Name": "year",
        "Type": "bigint"
      },
      {
        "Name": "quarter",
        "Type": "bigint"
      }
    ],
    "Location": "s3://crawler-public-us-west-2/flight/2016/csv",
    "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
    "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
    "Compressed": false,
    "NumberOfBuckets": -1,
    "SerdeInfo": {
      "SerializationLibrary": "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
      "Parameters": {
        "field.delim": ",",
        "serialization.format": ","
      }
    }
  },
  "PartitionKeys": [
    {
      "Name": "mon",
      "Type": "string"
    }
  ],
```

```
  "TableType": "EXTERNAL_TABLE",
  "Parameters": {
    "EXTERNAL": "TRUE",
    "classification": "csv",
    "columnsOrdered": "true",
    "compressionType": "none",
    "delimiter": ",",
    "skip.header.line.count": "1",
    "typeOfData": "file"
  }
}
EOF

    # Replace placeholder with actual table name
    if ! sed -i "s/TABLE_NAME_PLACEHOLDER/$TABLE_NAME/g" "$TABLE_INPUT_FILE" 2>/dev/
null; then
        echo "ERROR: Failed to substitute table name in JSON file" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    # Validate JSON syntax before using it
    if ! jq empty "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Invalid JSON in table input file" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi
}

# Function to create table
create_table() {
    echo "Step 2: Creating a table named $TABLE_NAME in database $DB_NAME"

    prepare_table_input

    local TABLE_ARN
    if ! aws glue create-table \
        --database-name "$DB_NAME" \
        --table-input "file://${TABLE_INPUT_FILE}" \
        --region "$AWS_REGION" \
        --output json > /dev/null 2>&1; then
        echo "ERROR: Failed to create table $TABLE_NAME" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
```

```
    fi

    aws glue tag-resource \
        --resource-arn "arn:aws:glue:${AWS_REGION}:${ACCOUNT_ID}:table/${DB_NAME}/
${TABLE_NAME}" \
        --tags-to-add Key=project,Value=doc-smith Key=tutorial,Value=glue-gs \
        --region "$AWS_REGION" \
        2>/dev/null || true

    CREATED_RESOURCES+=("table:$TABLE_NAME")
    echo "Table $TABLE_NAME created successfully."
}

# Function to get and display table details
display_table_details() {
    echo "Step 3: Getting details of table $TABLE_NAME"

    if ! aws glue get-table \
        --database-name "$DB_NAME" \
        --name "$TABLE_NAME" \
        --region "$AWS_REGION" \
        --output json; then
        echo "ERROR: Failed to retrieve table details" >&2
        exit 1
    fi
}

# Function to display summary
display_summary() {
    echo ""
    echo "==========================================="
    echo "RESOURCES CREATED"
    echo "==========================================="
    echo "Database: $DB_NAME"
    echo "Table: $TABLE_NAME"
    echo "==========================================="
}

# Main execution flow
validate_prerequisites
create_database
create_table
display_table_details
display_summary
```

```
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Starting cleanup process..."

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateDatabase
  - CreateTable
  - DeleteDatabase
  - DeleteTable
  - GetDatabase
  - GetTable

HealthImaging exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com HealthImaging.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateDatastore`

O código de exemplo a seguir mostra como usar `CreateDatastore`.

AWS CLI com script Bash

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_create_datastore
#
# This function creates an AWS HealthImaging data store for importing DICOM P10
 files.
#
# Parameters:
#       -n data_store_name - The name of the data store.
#
# Returns:
#       The datastore ID.
#    And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_create_datastore() {
  local datastore_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_create_datastore"
    echo "Creates an AWS HealthImaging data store for importing DICOM P10 files."
    echo "  -n data_store_name - The name of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) datastore_name="${OPTARG}" ;;
      h)
        usage
```

```
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_name" ]]; then
    errecho "ERROR: You must provide a data store name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging create-datastore \
    --datastore-name "$datastore_name" \
    --output text \
    --query 'datastoreId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging create-datastore operation failed.
$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte CreateDatastore em Referência de AWS CLI comandos.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteDatastore`

O código de exemplo a seguir mostra como usar `DeleteDatastore`.

AWS CLI com script Bash

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_delete_datastore
#
# This function deletes an AWS HealthImaging data store.
#
# Parameters:
#       -i datastore_id - The ID of the data store.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_delete_datastore() {
  local datastore_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_delete_datastore"
    echo "Deletes an AWS HealthImaging data store."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
```

```
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging delete-datastore \
    --datastore-id "$datastore_id")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging delete-datastore operation failed.
$response"
    return 1
  fi

  return 0
}
```

- Para obter detalhes da API, consulte DeleteDatastore em Referência de AWS CLI comandos.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDatastore`

O código de exemplo a seguir mostra como usar `GetDatastore`.

AWS CLI com script Bash

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_get_datastore
#
# Get a data store's properties.
#
# Parameters:
#       -i data_store_id - The ID of the data store.
#
# Returns:
#       [datastore_name, datastore_id, datastore_status, datastore_arn,  created_at,
 updated_at]
#    And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_get_datastore() {
  local datastore_id option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_get_datastore"
    echo "Gets a data store's properties."
```

```
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  local response

  response=$(
    aws medical-imaging get-datastore \
      --datastore-id "$datastore_id" \
      --output text \
      --query "[ datastoreProperties.datastoreName,
datastoreProperties.datastoreId, datastoreProperties.datastoreStatus,
datastoreProperties.datastoreArn,  datastoreProperties.createdAt,
datastoreProperties.updatedAt]"
  )
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
```

```
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte GetDatastore em Referência de AWS CLI comandos.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDatastores`

O código de exemplo a seguir mostra como usar `ListDatastores`.

AWS CLI com script Bash

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_list_datastores
#
# List the HealthImaging data stores in the account.
#
# Returns:
#       [[datastore_name, datastore_id, datastore_status]]
#    And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_list_datastores() {
```

```
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_list_datastores"
    echo "Lists the AWS HealthImaging data stores in the account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response
  response=$(aws medical-imaging list-datastores \
    --output text \
    --query "datastoreSummaries[*][datastoreName, datastoreId, datastoreStatus]")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte ListDatastores em Referência de AWS CLI comandos.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exemplos de IAM usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then
```

```
    source ./iam_operations.sh
  fi
}

echo_repeat "*" 88
echo "Welcome to the IAM create user and assume role demo."
echo
echo "This demo will create an IAM user, create an IAM role, and apply the role to
the user."
echo_repeat "*" 88
echo

echo -n "Enter a name for a new IAM user: "
get_input
user_name=$get_input_result

local user_arn
user_arn=$(iam_create_user -u "$user_name")

# shellcheck disable=SC2181
if [[ ${?} == 0 ]]; then
  echo "Created demo IAM user named $user_name"
else
  errecho "$user_arn"
  errecho "The user failed to create. This demo will exit."
  return 1
fi

local access_key_response
access_key_response=$(iam_create_user_access_key -u "$user_name")
# shellcheck disable=SC2181
if [[ ${?} != 0 ]]; then
  errecho "The access key failed to create. This demo will exit."
  clean_up "$user_name"
  return 1
fi

IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
local key_name=${access_key_values[0]}
local key_secret=${access_key_values[1]}

echo "Created access key named $key_name"

echo "Wait 10 seconds for the user to be ready."
```

```
sleep 10
echo_repeat "*" 88
echo

local iam_role_name
iam_role_name=$(generate_random_name "test-role")
echo "Creating a role named $iam_role_name with user $user_name as the principal."

local assume_role_policy_document="{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
      \"Effect\": \"Allow\",
      \"Principal\": {\"AWS\": \"$user_arn\"},
      \"Action\": \"sts:AssumeRole\"
      }]
  }"

local role_arn
role_arn=$(iam_create_role -n "$iam_role_name" -p "$assume_role_policy_document")

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
  echo "Created IAM role named $iam_role_name"
else
  errecho "The role failed to create. This demo will exit."
  clean_up "$user_name" "$key_name"
  return 1
fi

local policy_name
policy_name=$(generate_random_name "test-policy")
local policy_document="{
              \"Version\": \"2012-10-17\",
              \"Statement\": [{
                  \"Effect\": \"Allow\",
                  \"Action\": \"s3:ListAllMyBuckets\",
                  \"Resource\": \"arn:aws:s3:::*\"}]}"

local policy_arn
policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
# shellcheck disable=SC2181
if [[ ${?} == 0 ]]; then
  echo "Created  IAM policy named $policy_name"
else
```

```
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
  fi

  if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
  else
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
  fi

  local assume_role_policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"$role_arn\"}]}"

  local assume_role_policy_name
  assume_role_policy_name=$(generate_random_name "test-assume-role-")

  # shellcheck disable=SC2181
  local assume_role_policy_arn
  assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
    return 1
  fi

  echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
  sleep 10
  echo_repeat "*" 88
  echo

  echo "Try to list buckets without the new user assuming the role."
  echo_repeat "*" 88
```

```
  echo

  # Set the environment variables for the created user.
  # bashsupport disable=BP2001
  export AWS_ACCESS_KEY_ID=$key_name
  # bashsupport disable=BP2001
  export AWS_SECRET_ACCESS_KEY=$key_secret

  local buckets
  buckets=$(s3_list_buckets)

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. This should not have
happened."
  else
    errecho "Because the role with permissions has not been assumed, listing buckets
failed."
  fi

  echo
  echo_repeat "*" 88
  echo "Now assume the role $iam_role_name and list the buckets."
  echo_repeat "*" 88
  echo

  local credentials

  credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Assumed role $iam_role_name"
  else
    errecho "Failed to assume role."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"
    return 1
  fi

  IFS=$'\t ' read -r -a credentials <<<"$credentials"
```

```
  export AWS_ACCESS_KEY_ID=${credentials[0]}
  export AWS_SECRET_ACCESS_KEY=${credentials[1]}
  # bashsupport disable=BP2001
  export AWS_SESSION_TOKEN=${credentials[2]}

  buckets=$(s3_list_buckets)

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets succeeded
because of "
    echo "the assumed role."
  else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"
    return 1
  fi

  local result=0
  export AWS_ACCESS_KEY_ID=""
  export AWS_SECRET_ACCESS_KEY=""

  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
```

```
}
```

As funções do IAM usadas neste cenário.

```
##############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
##############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}
```

```
###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must supply
 a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
```

```
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
```

```
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function iam_create_user_access_key"
  echo "Creates an AWS Identity and Access Management (IAM) role."
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_json -- The assume role policy document."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
```

```
    --query Role.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
```

```
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}
```

```
###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
```

```
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
```

```
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
```

```
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi
```

```
  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
```

```
  echo "  -k access_key   The access key to delete."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:k:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    k) access_key="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must supply
 a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
```

```
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte AttachRolePolicy em Referência de AWS CLI comandos.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
```

```
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:f:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    f) file_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

response=$(aws iam create-access-key \
  --user-name "$user_name" \
  --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-access-key operation failed.$response"
  return 1
fi

if [[ -n "$file_name" ]]; then
  echo "$response" >"$file_name"
fi

local key_id key_secret
```

```
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}
```

- Para obter detalhes da API, consulte CreateAccessKey em Referência de AWS CLI comandos.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
```

```
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}
```

- Para obter detalhes da API, consulte CreatePolicy em Referência de AWS CLI comandos.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
```

```
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte CreateRole em Referência de AWS CLI comandos.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
```

```
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must supply
 a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
```

```
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte CreateUser em Referência de AWS CLI comandos.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function errecho
```

```
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Username:   $user_name"
  iecho "    Access key:   $access_key"
  iecho ""

  response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}
```

- Para obter detalhes da API, consulte DeleteAccessKey em Referência de AWS CLI comandos.

DeletePolicy

O código de exemplo a seguir mostra como usar `DeletePolicy`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
```

```
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""
```

```
  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}
```

- Para obter detalhes da API, consulte DeletePolicy em Referência de AWS CLI comandos.

`DeleteRole`

O código de exemplo a seguir mostra como usar `DeleteRole`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
```

```
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
```

```
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}
```

- Para obter detalhes da API, consulte DeleteRole em Referência de AWS CLI comandos.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
```

```
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must supply
a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi
```

```
  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- Para obter detalhes da API, consulte DeleteUser em Referência de AWS CLI comandos.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte DetachRolePolicy em Referência de AWS CLI comandos.

`GetUser`

O código de exemplo a seguir mostra como usar `GetUser`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}
```

```
  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}
```

- Para obter detalhes da API, consulte GetUser em Referência de AWS CLI comandos.

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#
```

```
# Parameters:
#       -u user_name -- The name of the IAM user.
#
# Returns:
#       access_key_ids
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_list_access_keys() {

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_access_keys"
    echo "Lists the AWS Identity and Access Management (IAM) access key IDs for the
 specified user."
    echo "  -u user_name   The name of the IAM user."
    echo ""
  }

  local user_name response
  local option OPTARG # Required to use getopts command in a function.
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi
```

```
  response=$(aws iam list-access-keys \
    --user-name "$user_name" \
    --output text \
    --query 'AccessKeyMetadata[].AccessKeyId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-access-keys operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte ListAccessKeys em Referência de AWS CLI comandos.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
##############################################################################
# function iam_list_users
#
# List the IAM users in the account.
#
# Returns:
#       The list of users names
#    And:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
##############################################################################
function iam_list_users() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_users"
    echo "Lists the AWS Identity and Access Management (IAM) user in the account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws iam list-users \
    --output text \
    --query "Users[].UserName")
  error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-users operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte ListUsers em Referência de AWS CLI comandos.

`UpdateAccessKey`

O código de exemplo a seguir mostra como usar `UpdateAccessKey`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iam_update_access_key
#
# This function can activate or deactivate an IAM access key for the specified IAM
 user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to update.
#       -a            -- Activate the selected access key.
#       -d            -- Deactivate the selected access key.
#
# Example:
#       # To deactivate the selected access key for IAM user Bob
#       iam_update_access_key -u Bob -k AKIAIOSFODNN7EXAMPLE -d
```

```
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_update_access_key() {
  local user_name access_key status response
  local option OPTARG # Required to use getopts command in a function.
  local activate_flag=false deactivate_flag=false

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_update_access_key"
    echo "Updates the status of an AWS Identity and Access Management (IAM) access
 key for the specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to update."
    echo "  -a              Activate the access key."
    echo "  -d              Deactivate the access key."
    echo ""
  }

  # Retrieve the calling parameters.
    while getopts "u:k:adh" option; do
      case "${option}" in
        u) user_name="${OPTARG}" ;;
        k) access_key="${OPTARG}" ;;
        a) activate_flag=true ;;
        d) deactivate_flag=true ;;
        h)
          usage
          return 0
          ;;
        \?)
          echo "Invalid parameter"
          usage
          return 1
          ;;
      esac
    done
    export OPTIND=1

   # Validate input parameters
    if [[ -z "$user_name" ]]; then
```

```
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
  fi

  # Ensure that only -a or -d is specified
  if [[ "$activate_flag" == true && "$deactivate_flag" == true ]]; then
    errecho "ERROR: You cannot specify both -a (activate) and -d (deactivate) at
the same time."
    usage
    return 1
  fi

  # If neither -a nor -d is provided, return an error
  if [[ "$activate_flag" == false && "$deactivate_flag" == false ]]; then
    errecho "ERROR: You must specify either -a (activate) or -d (deactivate)."
    usage
    return 1
  fi

  # Determine the status based on the flag
  if [[ "$activate_flag" == true ]]; then
    status="Active"
  elif [[ "$deactivate_flag" == true ]]; then
    status="Inactive"
  fi

  iecho "Parameters:\n"
  iecho "    Username:   $user_name"
  iecho "    Access key: $access_key"
  iecho "    New status: $status"
  iecho ""

  # Update the access key status
  response=$(aws iam update-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key" \
    --status "$status" 2>&1)
```

```
    local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log $error_code
      errecho "ERROR: AWS reports update-access-key operation failed.\n$response"
      return 1
    fi

    iecho "update-access-key response: $response"
    iecho

    return 0
}
```

- Para obter detalhes da API, consulte UpdateAccessKey em Referência de AWS CLI comandos.

Cenários

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
```

```
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
```

```
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
```

```
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
```

```
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"
```

```
    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi
```

```
    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
```

```
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}
```

```
# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
```

```
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
```

```
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
```

```
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "==========================================="
    echo "CREATED RESOURCES"
    echo "==========================================="
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "==========================================="
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
```

```
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")
```

```
                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster
```

```
    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNamespaces
  - ListServices
  - ListTaskDefinitions
  - ListTasks
  - RegisterTaskDefinition
  - UpdateService
  - Wait

Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um cluster.
- Criar uma definição de tarefa.
- Criar o serviço.
- Fazer a limpeza.

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"
```

```
# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"
```

```
    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|AccessDenied
\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
```

```
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security group
 $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
```

```
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "CLEANUP PROCESS"
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]]; then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
```

```
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
 anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]]; then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete security
 group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo ""
```

```
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix "$TASK_FAMILY"
 --query "taskDefinitionArns" --output text 2>/dev/null || echo "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME --
service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo "6. Deregister task definitions: aws ecs list-task-definitions --
family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --task-
definition <ARN>"
        fi
```

```
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --assume-
role-policy-document file://trust-policy.json" "Create ECS task execution role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
 "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-ecs-
fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"
```

```
# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-json
 file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --group-
name '$SECURITY_GROUP_NAME' --description 'Security group for ECS Fargate tutorial
```

```
 - HTTP access' --vpc-id '$VPC_ID' --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-fargate-
linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"' |
 head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP inbound
 rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
```

```
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-type
 FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets\":[\"$(echo
 "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups\":[\"$SECURITY_GROUP_ID
\"],\"assignPublicIp\":\"ENABLED\"}}' --tags key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi
```

```
echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN" --
query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --output
 text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID" --
query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
```

```
echo "==========================================="
echo "SERVICE INFORMATION"
echo "==========================================="
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "==========================================="
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "==========================================="
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices

- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Criar uma API REST com integração de proxy de função

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para execução do Lambda.
- Criar e implantar uma função do Lambda.
- Criar uma API REST
- Configurar a integração de proxy do Lambda.
- Implantar e testar a API
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
```

```
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
```

```
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
```

```
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || {
```

```
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)
```

```
# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query "items[?
pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
```

```
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
```

```
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed to
 delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning: Failed
 to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole

- DeleteFunction
- DeleteRestApi
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetResources
- GetRestApis
- PutIntegration
- PutMethod

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error
```

```
# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}
```

```
# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
 "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
```

```
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
```

```
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
 retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
 attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."
```

```
    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"
```

```
    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)
```

```
    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."
```

```
        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole
```

```
        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
```

```
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}
```

```
# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
```

```
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)
```

```
    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table
```

```
    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "============================================"
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "============================================"
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
```

```
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
```

```
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks

- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Como criar um espaço de trabalho de monitoramento gerenciado

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para seu espaço de trabalho.
- Criar um espaço de trabalho do Grafana.
- Configurar autenticação
- Definir configurações opcionais.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
```

```
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
```

```
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"
```

```
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-grafana-
gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0
```

```
while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -d'"' -
f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "==========================================="
echo "WORKSPACE INFORMATION"
echo "==========================================="
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
```

```
echo "Note: Since SAML authentication is used, you need to configure SAML settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "==========================================="

# Step 6: Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 | grep
 -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
```

```
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace

- DetachRolePolicy
- GetCallerIdentity

Como criar um painel de monitoramento com o nome da função como uma variável

O exemplo de código a seguir mostra como:

- Crie um CloudWatch painel
- Adicionar widgets de métricas do Lambda com uma variável de nome de função.
- Verificar o painel.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
```

```
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with 'aws
 configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"
```

```
# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

```
    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-dynamicdash
 \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
```

```
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/null
 --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
```

```
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}", { "stat":
 "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
```

```
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
```

```
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
```

```
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of the
 dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --region
 "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
```

```
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to delete
 it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach it
 manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AttachRolePolicy
- CreateFunction
- CreateRole
- DeleteDashboards
- DeleteFunction
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetDashboard
- Invoke
- ListDashboards
- ListFunctions
- PutDashboard

Como criar sua primeira função com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Lambda.
- Criar código da função.
- Criar uma função do Lambda
- Testar uma função do Lambda.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
```

```
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

###############################################################################
# Setup
###############################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

###############################################################################
# Helper functions
###############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
```

```
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could not
detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
IAM role ${NAME}."
                ;;
        esac
    done
```

```
    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
```

```
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

################################################################################
# Runtime selection
################################################################################

echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"
```

```
case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !== 'number')
 {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
```

```
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

################################################################################
# Step 1: Create IAM execution role
################################################################################

echo ""
echo "============================================"
echo "Step 1: Create IAM execution role"
echo "============================================"
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
```

```
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/service-
role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "============================================"
echo "Step 2: Create Lambda function"
echo "============================================"
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
```

```
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --query
 State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
```

```
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

##############################################################################
# Step 4: View CloudWatch logs
##############################################################################

echo ""
echo "============================================"
echo "Step 4: View CloudWatch Logs"
echo "============================================"
echo ""
```

```
echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
```

```
# Summary and cleanup
################################################################################

echo ""
echo "============================================"
echo "SUMMARY"
echo "============================================"
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction
  - GetFunctionConfiguration
  - GetLogEvents
  - Invoke

Conceitos básicos de data warehouses de tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Use o gerenciador de segredos CreateSecret
- Use o gerenciador de segredos DeleteSecret
- Use o gerenciador de segredos GetSecretValue
- Use redshift CreateNamespace
- Use redshift CreateWorkgroup
- Use redshift DeleteNamespace
- Use iam CreateRole

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq dependency)
# This script creates a Redshift Serverless environment, loads sample data, and runs
 queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2
```

```
  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi
```

```
    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
```

```
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status: $status.
 Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
```

```
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "============================================"
  echo "CLEANUP CONFIRMATION"
  echo "============================================"
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
```

```
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --workgroup-
name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --namespace-
name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
```

```
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin credentials
 for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
```

```
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless get-
namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
```

```
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless get-
workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
```

```
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
```

```
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
```

```
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
```

```
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID" 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
```

```
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time total
 sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price desc)
 as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID" 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
```

```
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""
```

```
# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole
  - GetSecretValue
  - GetWorkgroup
  - PutRolePolicy
  - UpdateNamespace

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.

- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
```

```
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi
```

```
        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
```

```
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
```

```
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
```

```
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
```

```
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Conceitos básicos da proteção de dispositivos da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM obrigatórios.
- Habiltiar verificações de auditoria do IoT Device Defender.
- Execute uma On-Demand auditoria
- Criar uma ação de mitigação.
- Aplicar ações de mitigação às descobertas.
- Configurar notificações do SNS (opcional).
- Habilitar o registro em log de IoT

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "==================================================="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /dev/
null; then
```

```
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/
null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
```

```
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
```

```
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi
```

```
        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
```

```
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
```

```
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "====================================================="
echo "Step 1: Creating required IAM roles"
echo "====================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")
```

```
# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "====================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "====================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1) ||
 true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "====================================================="
echo "Step 3: Running an on-demand audit"
```

```
echo "====================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
```

```
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null || echo
 "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
```

```
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "==================================================="
echo "Step 4: Creating a mitigation action"
echo "==================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting it
 first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1
 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true
```

```
if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 ||
 true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "====================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "====================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi
```

```
    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1) ||
 true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
```

```
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --output
 json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
```

```
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "==================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "==================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
```

```
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2 methods,
 continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved but
 could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
```

```
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."
```

```
cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --output
 json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null ||
 echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"
```

```
echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies
  - ListTopics
  - PutRolePolicy
  - SetLoggingOptions
  - SetV2LoggingOptions
  - StartAuditMitigationActionsTask
  - StartOnDemandAuditTask

- UpdateAccountAuditConfiguration

Conceitos básicos de arquivos de atributos de machine learning

O exemplo de código a seguir mostra como:

- Configurar permissões do IAM.
- Crie uma função SageMaker de execução
- Criar grupo de atributos.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
```

```
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)
```

```
        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
```

```
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi
```

```
    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
```

```
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
```

```
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
```

```
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
```

```
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"
```

```
CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
```

```
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
```

```
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS" |
 grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
```

```
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket

- CreateFeatureGroup
- CreateRole
- DeleteBucket
- DeleteFeatureGroup
- DeleteRole
- DescribeFeatureGroup
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListFeatureGroups
- PutPublicAccessBlock
- Rm

Conceitos básicos de clusters gerenciados de kubernetes

O exemplo de código a seguir mostra como:

- Criar uma VPC para um cluster do EKS.
- Criar perfis do IAM para um cluster do EKS.
- Criação do seu cluster de EKS
- Configurar o kubectl para se comunicar com o cluster.
- Crie um grupo de nós gerenciados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the AWS
 CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your operating
 system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
```

```
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/
kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --nodegroup-
name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
```

```
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources; exit
 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")
```

```
echo "Waiting for CloudFormation stack to complete (this may take a few minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"
```

```
# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"
```

```
# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
```

```
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-kubeconfig
 --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"
```

```
NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn" --
output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl get
 nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or the
 nodes are still registering."
```

```
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl get
 all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
```

```
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --cluster-
name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --stack-
name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole

- DeleteStack
- DescribeCluster
- DescribeNodegroup
- DescribeStacks
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

Conceitos básicos de streaming gerenciado

O exemplo de código a seguir mostra como:

- Criar um cluster do Amazon MSK.
- Criar permissões do IAM para acessar o MSK.
- Criar uma máquina cliente
- Obter os agentes de bootstrap.
- Configurar a máquina cliente.
- Crie um tópico e produce/consume dados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
```

```
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group: $CLIENT_SG_ID";
 fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
```

```
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id" &>/
dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated" &>/
dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
```

```
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o "\"GroupId
\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing $sg_id,
 removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing $sg_id,
 removing them..."
```

```
                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No outbound
 all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed to
 terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group" "$CLIENT_SG_ID";
 then
        remove_security_group_references "$CLIENT_SG_ID"
```

```
        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete key
pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
```

```
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to delete
 policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
```

```
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az (subnet:
 $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "================================================"

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
```

```
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default VPC
 in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-az,Values=true"
 \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least 3
 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=$DEFAULT_VPC_ID"
 \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"
```

```
# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\", \"ClientSubnets
\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\", \"${SUBNET_ARRAY[2]}\"],
 \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-gs
 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --query
 "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
```

```
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
```

```
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
```

```
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "================================================"

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@]; then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi
```

```
echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)
```

```
if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
```

```
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
```

```
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-${RANDOM_SUFFIX}},
{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-streaming-for-
apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response: $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)
```

```
    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
```

```
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi
```

```
echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-
auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/$LATEST_VERSION/
aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
```

```
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
```

```
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
```

```
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
```

```
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile

- CreateKeyPair
- CreatePolicy
- CreateRole
- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances

- TerminateInstances

- Wait

Conceitos básicos de clusters de data warehouse provisionados

O exemplo de código a seguir mostra como:

- Criar um cluster do Redshift.
- Criar um perfil do IAM para acessar o S3.
- Criar tabelas e carregar dados.
- Executar consultas de exemplo.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
```

```
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-final-
cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/dev/
null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
```

```
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""

    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --query
 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id "$statement_id"
 --query 'Error' --output text 2>/dev/null || echo "Unknown error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
```

```
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus" \
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and try
 again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi
```

```
# Validate password meets requirements
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi

# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
```

```
cat > redshift-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed to
 attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
```

```
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer not
 null, eventid integer not null, dateid smallint not null sortkey, qtysold smallint
 not null, pricepaid decimal(8,2), commission decimal(8,2), saletime timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint not
 null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi
```

```
echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt' DELIMITER
 '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"
```

```
# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid AND
 caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
```

```
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

Conceitos básicos da orquestração de fluxo de trabalaho

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Step Functions.
- Criar sua primeira máquina de estado.
- Iniciar a execução da máquina de estado.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS Step
 Functions Getting Started tutorial

set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
done

# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT

echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"
```

```
# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be less
 reliable."
    echo "Consider installing jq for better error handling: brew install jq (macOS)
 or apt-get install jq (Ubuntu)"
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
export AWS_REGION="$CURRENT_REGION"

# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
```

```
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}

# Function to extract JSON field efficiently
extract_json_field() {
```

```
    local json="$1"
    local field="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] || [ "$wait_time"
 -gt 300 ]; then
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"

    if [[ "$USE_JQ" == "true" ]]; then
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
```

```
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."

    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
 $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for detach operations to complete
    wait 2>/dev/null || true
```

```
    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed to
 delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
        "input.json"
        "sentiment-input.json"
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
            fi
        fi
    done
}
```

```
# Security: Set trap to cleanup on script exit
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
```

```
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
```

```
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)

check_api_error "$ROLE_RESULT" "Create IAM role"
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"
```

```
# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")

if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)

if [ $? -ne 0 ]; then
```

```
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi

# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)

check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"

# Wait for execution to complete (the workflow has a 10-second wait state)
```

```
echo "Waiting for execution to complete (12 seconds)..."
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
```

```
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
```

```
}
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)

check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"

# Wait for execution to complete (the workflow has a 5-second wait state)
```

```
echo "Waiting for execution to complete (8 seconds)..."
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")

    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
    fi
    echo "Policy ARN: $POLICY_ARN"
```

```
    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
```

```
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
      },
      "Next": "SuccessState"
```

```
    },
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"

    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)
```

```
    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3

    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
```

```
fi
echo "==========================================="

# Cleanup
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy
  - StartExecution
  - UpdateStateMachine

Como mover segredos codificados para o armazenamento seguro de segredos

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Criar um segredo no Secrets Manager.
- Atualizar o código de uma aplicação.
- Atualizar o segredo.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}
```

```
# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name "$RUNTIME_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi
```

```
    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME" --
query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-name
 "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10
```

```
# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_urls
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/null
 || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
```

```
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').token_

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi
```

```
echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy
  - GetCallerIdentity
  - GetSecretValue

- PutResourcePolicy
- UpdateSecret

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
```

```
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi
```

```
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
```

```
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
```

```
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")
```

```
# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
```

```
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
```

```
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
```

```
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
```

```
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
```

```
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
```

```
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
```

```
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
```

```
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
```

```
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances
  - StartExperiment
  - StopExperiment

- TerminateInstances
- Wait

Configurar o controle de Attribute-Based acesso

O exemplo de código a seguir mostra como implementar o Controle de Attribute-Based Acesso (ABAC) para o DynamoDB.

- Crie uma política do IAM para o ABAC.
- Crie tabelas com etiquetas que representem diferentes departamentos.
- Liste e filtre tabelas com base nas etiquetas.

AWS CLI com script Bash

Crie uma política do IAM para o ABAC.

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
```

```
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

Crie tabelas com etiquetas que representem diferentes departamentos.

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
```

```
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

Liste e filtre tabelas com base nas etiquetas.

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text | xargs
 -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn" --output
 text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreatePolicy
  - CreateTable
  - ListTables

Como configurar o gerenciamento de sistemas

O exemplo de código a seguir mostra como:

- Criar permissões do IAM para o Systems Manager.
- Criar um perfil do IAM para o Systems Manager
- Configurar o Systems Manager.
- Verificar a configuração.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
```

```
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"
```

```
    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id --
policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --manager-
arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null
 || true
```

```
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and region."
echo "It will create IAM policies and roles, then enable Systems Manager features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
```

```
          "ssm:ListAssociations",
          "ssm:DescribeAssociation",
          "ssm:ListDocuments",
          "ssm:ListResourceDataSync",
          "ssm:DescribePatchBaselines",
          "ssm:GetPatchBaseline",
          "ssm:DescribeMaintenanceWindows",
          "ssm:DescribeMaintenanceWindowTasks"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
```

```
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
```

```
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
```

```
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
```

```
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/accountdiscovery.ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM_AccountDiscovery",
```

```
        "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
        "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
      ]
    }
  ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
```

```
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
```

```
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --policy-
arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
```

```
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-arn
 $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration manager"
```

```
echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up later." |
 tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole

- DeleteConfigurationManager
- DeletePolicy
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetConfigurationManager

Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar funções do Lambda para monitoramento.
- Crie um CloudWatch painel
- Adicionar uma variável de propriedade ao painel.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
```

```
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
```

```
    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/dev/
null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
```

```
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"
```

```
# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"
```

```
# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"
```

```
# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /dev/
null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /dev/
null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a "$LOG_FILE"

# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
```

```
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name "$DASHBOARD_NAME"
 --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
 | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME" --
output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
```

```
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

Trabalhe com Streams e Time-to-Live

O exemplo de código a seguir mostra como gerenciar streams e recursos do DynamoDB. Time-to-Live

- Crie uma tabela com a opção de fluxos habilitada.
- Descreva os fluxos.
- Crie uma função do Lambda para processar os fluxos.
- Habilite a vida útil (TTL) em uma tabela.
- Adicione itens com atributos de TTL.
- Descreva as configurações de TTL.

AWS CLI com script Bash

Crie uma tabela com a opção de fluxos habilitada.

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

Descreva os fluxos.

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)
```

```
echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

Crie uma função do Lambda para os fluxos.

```
# Step 1: Create an IAM role for the Lambda function
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
```

```
echo "    --starting-position LATEST"
```

Habilite a vida útil (TTL) em uma tabela.

```
# Create a table for TTL demonstration
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo

# Enable TTL on the table
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

Adicione itens com atributos de TTL.

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
```

```
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

Descreva as configurações de TTL.

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
    --table-name TTLDemo
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole
  - CreateTable
  - DescribeTable
  - DescribeTimeToLive
  - PutItem
  - UpdateTimeToLive

AWS IoT exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS IoT.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos da proteção de dispositivos da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM obrigatórios.
- Habiltiar verificações de auditoria do IoT Device Defender.
- Execute uma On-Demand auditoria
- Criar uma ação de mitigação.
- Aplicar ações de mitigação às descobertas.
- Configurar notificações do SNS (opcional).
- Habilitar o registro em log de IoT

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "====================================================="
echo "Starting script execution at $(date)"
```

```
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /dev/
null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/
null) || true
```

```
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
```

```
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)
```

```
        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
```

```
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}
```

```
# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "====================================================="
echo "Step 1: Creating required IAM roles"
echo "====================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")
```

```
# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
```

```
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "====================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "====================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1) ||
 true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"
```

```
# Step 3: Run an on-demand audit
echo ""
echo "====================================================="
echo "Step 3: Running an on-demand audit"
echo "====================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi
```

```
    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null || echo
 "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi
```

```
# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "====================================================="
echo "Step 4: Creating a mitigation action"
echo "====================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting it
 first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1
 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi
```

```
MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 ||
 true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "==================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "==================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)
```

```
    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1) ||
 true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
```

```
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --output
 json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
```

```
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "====================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "====================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi
```

```
    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2 methods,
 continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved but
 could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10
```

```
# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi
```

```
# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --output
 json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null ||
 echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}
```

```
cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies
  - ListTopics
  - PutRolePolicy
  - SetLoggingOptions

- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Conceitos básicos de mensagens da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar recursos de IoT
- Configurar o dispositivo
- Executar os aplicativos de exemplo
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
}
```

```
# Function to check for errors
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a $LOG_FILE
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --
new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
}
```

```
# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "==================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a $LOG_FILE
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
    echo "AWS region not configured. Please run 'aws configure' to set your region."
 | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE

# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE

# Create IoT policy
```

```
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-document file://
iot-policy.json --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-core-
gs"
check_error $?

# Create IoT thing
```

```
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn' --
output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?

# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active --
certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile $CERTS_DIR/
public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target $CERTIFICATE_ARN"
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
check_error $?
```

```
# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." | tee
 -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-python-
v2.git"
    check_error $?
else
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
fi
```

```
# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
```

```
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE

# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
```

```
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE

    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"

    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"

    # Delete policies
    echo "Deleting policies..." | tee -a $LOG_FILE
```

```
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-status
 INACTIVE" | tee -a $LOG_FILE
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachPolicy
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreatePolicy
  - CreateThing
  - DeleteCertificate
  - DeletePolicy
  - DeleteThing
  - DescribeEndpoint
  - DetachPolicy

- DetachThingPrincipal
- GetCallerIdentity
- UpdateCertificate

AWS KMS exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS KMS.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Trabalhar com a criptografia de tabelas

O exemplo de código a seguir mostra como gerenciar opções de criptografia para tabelas do DynamoDB.

- Crie uma tabela com a criptografia padrão.
- Crie uma tabela com uma chave gerenciada pelo cliente (CMK).
- Atualize as configurações de criptografia da tabela.
- Descreva a criptografia da tabela.

AWS CLI com script Bash

Crie uma tabela com a criptografia padrão.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
```

```
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Crie uma tabela com uma chave gerenciada pelo cliente (CMK).

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Atualize a criptografia da tabela.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Descreva a criptografia da tabela.

```
# Describe the table to see encryption settings
```

```
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateKey
  - CreateTable
  - DescribeTable
  - UpdateTable

Exemplos de lambda usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Lambda.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma API REST com integração de proxy de função

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para execução do Lambda.
- Criar e implantar uma função do Lambda.
- Criar uma API REST
- Configurar a integração de proxy do Lambda.
- Implantar e testar a API
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
```

```
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
```

```
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
```

```
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
```

```
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query "items[?
pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
```

```
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
```

```
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed to
 delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning: Failed
 to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json
```

```
echo "Cleanup completed!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Como criar um painel de monitoramento com o nome da função como uma variável

O exemplo de código a seguir mostra como:

- Crie um CloudWatch painel
- Adicionar widgets de métricas do Lambda com uma variável de nome de função.
- Verificar o painel.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
```

```
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with 'aws
 configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure permissions
```

```
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-dynamicdash
 \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
```

```
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/null
 --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
```

```
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
```

```
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}", { "stat":
 "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
```

```
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
```

```
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of the
 dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""
```

```
# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --region
 "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to delete
 it manually."
        fi

        echo "Detaching role policy..."
```

```
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach it
 manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards

- ListFunctions
- PutDashboard

Como criar sua primeira função com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Lambda.
- Criar código da função.
- Criar uma função do Lambda
- Testar uma função do Lambda.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

###############################################################################
# Setup
###############################################################################
```

```
UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

###############################################################################
# Helper functions
###############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
```

```
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could not
 detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
```

```
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}
```

```
##############################################################################
# Region pre-check
##############################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

##############################################################################
# Runtime selection
##############################################################################

echo ""
echo "============================================"
echo "AWS Lambda - Create Your First Function"
echo "============================================"
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
```

```
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !== 'number')
 {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

##############################################################################
# Step 1: Create IAM execution role
##############################################################################
```

```
echo ""
echo "==========================================="
echo "Step 1: Create IAM execution role"
echo "==========================================="
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
```

```
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/service-
role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

################################################################################
# Step 2: Create Lambda function
################################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
```

```
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --query
 State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
```

```
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

###############################################################################
# Step 4: View CloudWatch logs
###############################################################################

echo ""
echo "==========================================="
echo "Step 4: View CloudWatch Logs"
echo "==========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
```

```
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################

echo ""
echo "============================================"
echo "SUMMARY"
echo "============================================"
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
```

```
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction
  - GetFunctionConfiguration
  - GetLogEvents
  - Invoke

Como usar variáveis de propriedade nos painéis de monitoramento para monitorar várias funções com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar funções do Lambda para monitoramento.
- Crie um CloudWatch painel
- Adicionar uma variável de propriedade ao painel.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
```

```
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
```

```
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/dev/
null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
```

```
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
```

```
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /dev/
null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /dev/
null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a "$LOG_FILE"

# Create dashboard body with proper escaping and validation
```

```
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name "$DASHBOARD_NAME"
 --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
 | tee -a "$LOG_FILE"
```

```
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME" --
output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
```

```
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

Exemplos de uso do Lightsail AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Lightsail.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução aos servidores privados virtuais simples

O exemplo de código a seguir mostra como:

- Criar uma instância do
- Adicionar armazenamento à sua instância
- Criar um snapshot
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Lightsail Getting Started CLI Script
# This script demonstrates how to create and manage Lightsail resources using the
 AWS CLI

set -euo pipefail

# Set AWS region
export AWS_REGION="${AWS_REGION:-us-west-2}"
echo "Using AWS region: $AWS_REGION"

# Set up logging
LOG_FILE="lightsail-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Lightsail Getting Started script at $(date)"

# Error handling function
handle_error() {
  echo "ERROR: $1" >&2
```

```
  echo "Attempting to clean up resources..."
  cleanup_resources || true
  exit 1
}

# Function to check if a command succeeded
check_status() {
  if [ $? -ne 0 ]; then
    handle_error "$1"
  fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
INSTANCE_NAME="LightsailInstance-${RANDOM_ID}"
DISK_NAME="LightsailDisk-${RANDOM_ID}"
SNAPSHOT_NAME="LightsailSnapshot-${RANDOM_ID}"

# Array to track created resources
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
  CREATED_RESOURCES+=("$1:$2")
  echo "Created $1: $2"
}

# Function to clean up resources
cleanup_resources() {
  echo "Resources created by this script:"
  for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "  $resource"
  done

  # Reverse the array to delete resources in reverse order
  for (( idx=${#CREATED_RESOURCES[@]}-1 ; idx>=0 ; idx-- )); do
    IFS=':' read -r type name <<< "${CREATED_RESOURCES[idx]}"

    case "$type" in
      "instance_snapshot")
        echo "Deleting instance snapshot: $name"
        aws lightsail delete-instance-snapshot --instance-snapshot-name "$name" --
region "$AWS_REGION" 2>/dev/null || true
        ;;
```

```
      "disk_snapshot")
        echo "Deleting disk snapshot: $name"
        aws lightsail delete-disk-snapshot --disk-snapshot-name "$name" --region
 "$AWS_REGION" 2>/dev/null || true
        ;;
      "disk")
        echo "Detaching disk: $name"
        aws lightsail detach-disk --disk-name "$name" --region "$AWS_REGION" 2>/dev/
null || true
        sleep 10
        echo "Deleting disk: $name"
        aws lightsail delete-disk --disk-name "$name" --region "$AWS_REGION" 2>/dev/
null || true
        ;;
      "instance")
        echo "Deleting instance: $name"
        INSTANCE_STATE=$(aws lightsail get-instance-state --instance-name "$name"
 --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null || echo
 "unknown")
        if [ "$INSTANCE_STATE" == "pending" ]; then
          echo "Instance is in pending state. Waiting for it to be ready before
 deleting..."
          MAX_WAIT=30
          WAITED=0
          while [ "$INSTANCE_STATE" == "pending" ] && [ $WAITED -lt $MAX_WAIT ]; do
            sleep 10
            WAITED=$((WAITED+1))
            INSTANCE_STATE=$(aws lightsail get-instance-state --instance-name
 "$name" --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null ||
 echo "unknown")
            echo "Instance state: $INSTANCE_STATE"
          done
        fi
        aws lightsail delete-instance --instance-name "$name" --region "$AWS_REGION"
 2>/dev/null || true
        ;;
    esac
  done

  echo "Cleanup completed"
}

# Step 1: Verify AWS CLI configuration
echo "Step 1: Verifying AWS CLI configuration"
```

```
aws configure list
check_status "Failed to verify AWS CLI configuration"

# Step 2: Get available blueprints and bundles
echo "Step 2: Getting available blueprints and bundles"
echo "Available blueprints (showing first 5):"
aws lightsail get-blueprints --region "$AWS_REGION" --query 'blueprints[0:5].
[blueprintId,name]' --output table
check_status "Failed to get blueprints"

echo "Available bundles (showing first 5):"
aws lightsail get-bundles --region "$AWS_REGION" --query 'bundles[0:5].
[bundleId,name,price]' --output table
check_status "Failed to get bundles"

# Get available regions and availability zones
echo "Getting available regions and availability zones"
# Use a specific availability zone in us-west-2 region
AVAILABILITY_ZONE="us-west-2a"
echo "Using availability zone: $AVAILABILITY_ZONE"

# Step 3: Create a Lightsail instance
echo "Step 3: Creating Lightsail instance: $INSTANCE_NAME"
aws lightsail create-instances \
  --instance-names "$INSTANCE_NAME" \
  --availability-zone "$AVAILABILITY_ZONE" \
  --blueprint-id amazon_linux_2023 \
  --bundle-id nano_3_0 \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create Lightsail instance"
track_resource "instance" "$INSTANCE_NAME"

# Wait for the instance to be in a running state
echo "Waiting for instance to be in running state..."
MAX_ATTEMPTS=30
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  STATUS=$(aws lightsail get-instance-state --instance-name "$INSTANCE_NAME"
 --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null || echo
 "unknown")
  if [ "$STATUS" == "running" ]; then
    echo "Instance is now running"
    break
```

```
  fi
  echo "Instance status: $STATUS. Waiting..."
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  handle_error "Instance failed to reach running state after 5 minutes"
fi

# Get instance details
echo "Getting instance details"
INSTANCE_IP=$(aws lightsail get-instance --instance-name "$INSTANCE_NAME" --region
 "$AWS_REGION" --query 'instance.publicIpAddress' --output text)
check_status "Failed to get instance IP address"
echo "Instance IP address: $INSTANCE_IP"

# Validate IP address format
if ! [[ "$INSTANCE_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
  handle_error "Invalid IP address format: $INSTANCE_IP"
fi

# Step 4: Download the default key pair
echo "Step 4: Downloading default key pair"
KEY_FILE="lightsail_key_${RANDOM_ID}.pem"
umask 077
aws lightsail download-default-key-pair --region "$AWS_REGION" --output text >
 "$KEY_FILE"
check_status "Failed to download key pair"

if [ ! -f "$KEY_FILE" ] || [ ! -s "$KEY_FILE" ]; then
  handle_error "Key pair file was not created or is empty"
fi

chmod 400 "$KEY_FILE"
check_status "Failed to set permissions on key pair"
echo "Key pair downloaded to $KEY_FILE"

echo "To connect to your instance, use:"
echo "ssh -i $KEY_FILE ec2-user@$INSTANCE_IP"

# Step 5: Create a block storage disk
echo "Step 5: Creating block storage disk: $DISK_NAME"
aws lightsail create-disk \
```

```
  --disk-name "$DISK_NAME" \
  --availability-zone "$AVAILABILITY_ZONE" \
  --size-in-gb 8 \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create disk"
track_resource "disk" "$DISK_NAME"

# Wait for the disk to be available using polling
echo "Waiting for disk to be available..."
MAX_ATTEMPTS=30
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  DISK_STATE=$(aws lightsail get-disk --disk-name "$DISK_NAME" --region
 "$AWS_REGION" --query 'disk.state' --output text 2>/dev/null || echo "unknown")
  if [ "$DISK_STATE" == "available" ]; then
    echo "Disk is now available"
    break
  fi
  echo "Disk status: $DISK_STATE. Waiting..."
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  handle_error "Disk failed to become available after 5 minutes"
fi

# Attach the disk to the instance
echo "Attaching disk to instance"
aws lightsail attach-disk \
  --disk-name "$DISK_NAME" \
  --instance-name "$INSTANCE_NAME" \
  --disk-path /dev/xvdf \
  --region "$AWS_REGION"
check_status "Failed to attach disk to instance"

echo "Disk attached. To format and mount the disk, connect to your instance and
 run:"
echo "sudo mkfs -t ext4 /dev/xvdf"
echo "sudo mkdir -p /mnt/my-data"
echo "sudo mount /dev/xvdf /mnt/my-data"
echo "sudo chown ec2-user:ec2-user /mnt/my-data"
```

```
# Step 6: Create a snapshot of the instance
echo "Step 6: Creating snapshot of the instance: $SNAPSHOT_NAME"
aws lightsail create-instance-snapshot \
  --instance-name "$INSTANCE_NAME" \
  --instance-snapshot-name "$SNAPSHOT_NAME" \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create instance snapshot"
track_resource "instance_snapshot" "$SNAPSHOT_NAME"

# Wait for the snapshot to complete using polling
echo "Waiting for snapshot to complete... (this may take several minutes)"
MAX_ATTEMPTS=60
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  SNAPSHOT_STATE=$(aws lightsail get-instance-snapshot --instance-snapshot-name
 "$SNAPSHOT_NAME" --region "$AWS_REGION" --query 'instanceSnapshot.state' --output
 text 2>/dev/null || echo "unknown")
  if [ "$SNAPSHOT_STATE" == "available" ]; then
    echo "Snapshot creation completed"
    break
  fi
  echo "Snapshot status: $SNAPSHOT_STATE. Waiting... ($ATTEMPTS/$MAX_ATTEMPTS)"
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  echo "Warning: Snapshot creation is taking longer than expected but will continue
 in the background."
  echo "You can check its status later with: aws lightsail get-instance-snapshot --
instance-snapshot-name $SNAPSHOT_NAME --region $AWS_REGION"
fi

# Step 7: Clean up resources
echo "Step 7: Clean up resources"
echo "The script has created the following resources:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
  echo "  $resource"
done

echo "Cleaning up these resources automatically..."
cleanup_resources
```

```
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachDisk
  - CreateDisk
  - CreateInstanceSnapshot
  - CreateInstances
  - DeleteDisk
  - DeleteInstance
  - DeleteInstanceSnapshot
  - DetachDisk
  - DownloadDefaultKeyPair
  - GetBlueprints
  - GetBundles
  - GetDisk
  - GetInstance
  - GetInstanceSnapshot
  - GetInstanceState

Exemplos do Amazon Managed Grafana usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon Managed Grafana.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Como criar um espaço de trabalho de monitoramento gerenciado

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para seu espaço de trabalho.
- Criar um espaço de trabalho do Grafana.
- Configurar autenticação
- Definir configurações opcionais.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
```

```
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"
```

```
# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
```

```
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
```

```
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-grafana-
gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
```

```
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -d'"' -
f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 | grep
 -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
```

```
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

MediaConnect exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com MediaConnect.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução à transmissão de transporte de vídeo

O exemplo de código a seguir mostra como:

- Verifique o acesso ao Elemental MediaConnect
- Criar um fluxo.
- Adicionar uma saída.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Elemental MediaConnect Getting Started Tutorial Script
# This script creates a MediaConnect flow, adds an output, grants an entitlement,
# and then cleans up the resources.

set -euo pipefail

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="mediaconnect-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Elemental MediaConnect tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
```

```
# Tags for all resources
TAGS_ARRAY=("Key=project,Value=doc-smith" "Key=tutorial,Value=aws-elemental-
mediaconnect-gs")

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI is available
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        handle_error "AWS CLI is not installed or not in PATH"
    fi

    # Security: Verify AWS CLI version is recent
    local aws_version
    aws_version=$(aws --version 2>&1 | head -1)
    echo "AWS CLI version: $aws_version"

    if ! aws sts get-caller-identity &> /dev/null; then
        handle_error "AWS credentials are not configured or invalid"
    fi

    # Security: Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query Account --output text 2>/dev/
null)
    if [ -z "$account_id" ]; then
        handle_error "Failed to retrieve AWS account ID"
    fi
    echo "AWS Account ID: $account_id"
}

# Function to safely extract JSON values using jq (preferred) or fallback
extract_json_value() {
    local json_output="$1"
    local key="$2"

    if [ -z "$json_output" ]; then
```

```
        return 1
    fi

    # Security: Use jq if available for safer JSON parsing
    if command -v jq &> /dev/null; then
        echo "$json_output" | jq -r ".${key} // empty" 2>/dev/null || return 1
    else
        # Fallback with additional validation
        if ! echo "$json_output" | grep -q "\"$key\""; then
            return 1
        fi
        echo "$json_output" | grep -o "\"$key\": \"[^\"]*" | head -1 | cut -d'"' -f4
    fi
}

# Function to tag MediaConnect resource
tag_mediaconnect_resource() {
    local resource_arn="$1"
    echo "Tagging resource: $resource_arn"

    for tag in "${TAGS_ARRAY[@]}"; do
        if ! aws mediaconnect tag-resource --resource-arn "$resource_arn" --tags
 "$tag" 2>&1; then
            echo "WARNING: Failed to apply tag $tag to resource"
        fi
    done
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "${FLOW_ARN:-}" ]; then
        # Security: Validate ARN format before using it
        if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-
Z0-9:-]+$ ]]; then
            echo "WARNING: Invalid Flow ARN format, skipping cleanup: $FLOW_ARN"
            return 1
        fi

        # Check flow status before attempting to stop
        echo "Checking flow status..."
        local flow_status_output
```

```
        if flow_status_output=$(aws mediaconnect describe-flow --flow-arn
 "$FLOW_ARN" --query "Flow.Status" --output text 2>&1); then
            echo "Current flow status: $flow_status_output"

            if [ "$flow_status_output" == "ACTIVE" ] || [ "$flow_status_output" ==
 "UPDATING" ]; then
                echo "Stopping flow: $FLOW_ARN"
                if aws mediaconnect stop-flow --flow-arn "$FLOW_ARN" 2>&1; then
                    # Wait for flow to stop before deleting
                    echo "Waiting for flow to stop..."
                    sleep 10
                else
                    echo "WARNING: Failed to stop flow. Attempting to delete
 anyway..."
                fi
            else
                echo "Flow is not in ACTIVE or UPDATING state, skipping stop
 operation."
            fi

            # Delete the flow
            echo "Deleting flow: $FLOW_ARN"
            if aws mediaconnect delete-flow --flow-arn "$FLOW_ARN" 2>&1; then
                echo "Flow deleted successfully"
            else
                echo "WARNING: Failed to delete flow. You may need to manually
 delete it from the AWS console."
            fi
        else
            echo "WARNING: Could not check flow status"
        fi
    fi
}

# Set trap to cleanup on script exit
trap cleanup_resources EXIT

# Validate AWS CLI setup
validate_aws_cli

# Get the current AWS region
aws_region=""
if aws_region=$(aws configure get region 2>/dev/null); then
    if [ -z "$aws_region" ]; then
```

```
        handle_error "Failed to get AWS region. Please make sure AWS CLI is
 configured."
    fi
else
    handle_error "Failed to retrieve AWS region configuration"
fi

# Security: Validate region format
if [[ ! "$aws_region" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $aws_region"
fi

AWS_REGION="$aws_region"
echo "Using AWS Region: $AWS_REGION"

# Get available availability zones in the current region
echo "Getting available availability zones in region $AWS_REGION..."
az_output=""
if az_output=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --query
 "AvailabilityZones[0].ZoneName" --output text 2>&1); then
    AVAILABILITY_ZONE="$az_output"
    if [ -z "$AVAILABILITY_ZONE" ]; then
        handle_error "Failed to retrieve availability zones"
    fi

    # Security: Validate AZ format
    if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9][a-z]$ ]]; then
        handle_error "Invalid availability zone format: $AVAILABILITY_ZONE"
    fi

    echo "Using availability zone: $AVAILABILITY_ZONE"
else
    handle_error "Failed to get availability zones"
fi

# Generate a unique suffix for resource names
SUFFIX=$(date +%s | cut -c 6-10)
FLOW_NAME="AwardsNYCShow-${SUFFIX}"
SOURCE_NAME="AwardsNYCSource-${SUFFIX}"
OUTPUT_NAME="AwardsNYCOutput-${SUFFIX}"
ENTITLEMENT_NAME="PhillyTeam-${SUFFIX}"

echo "Using the following resource names:"
echo "Flow name: $FLOW_NAME"
```

```
echo "Source name: $SOURCE_NAME"
echo "Output name: $OUTPUT_NAME"
echo "Entitlement name: $ENTITLEMENT_NAME"

# Step 1: Verify access to MediaConnect
echo "Step 1: Verifying access to AWS Elemental MediaConnect..."
list_flows_output=""
if list_flows_output=$(aws mediaconnect list-flows 2>&1); then
    echo "$list_flows_output"
else
    handle_error "Failed to list flows. Please check your AWS credentials and
 permissions."
fi

# Step 2: Create a flow
echo "Step 2: Creating a flow..."
create_flow_output=""
if create_flow_output=$(aws mediaconnect create-flow \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "$FLOW_NAME" \
    --source "Name=$SOURCE_NAME,Protocol=zixi-
push,WhitelistCidr=10.24.34.0/23,StreamId=ZixiAwardsNYCFeed" 2>&1); then
    echo "$create_flow_output"
else
    handle_error "Failed to create flow"
fi

# Extract the flow ARN from the output
FLOW_ARN=$(echo "$create_flow_output" | jq -r '.Flow.FlowArn // empty' 2>/dev/null)
if [ -z "$FLOW_ARN" ]; then
    FLOW_ARN=$(echo "$create_flow_output" | grep -o '"FlowArn": "[^"]*' | head -1 |
 cut -d'"' -f4)
fi
if [ -z "$FLOW_ARN" ]; then
    handle_error "Failed to extract flow ARN from output"
fi
echo "Flow ARN: $FLOW_ARN"

# Validate flow ARN format
if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-Z0-9:-]+
$ ]]; then
    handle_error "Invalid Flow ARN format: $FLOW_ARN"
fi
```

```
# Tag the flow
tag_mediaconnect_resource "$FLOW_ARN"

# Step 3: Add an output
echo "Step 3: Adding an output to the flow..."
add_output_output=""
if add_output_output=$(aws mediaconnect add-flow-outputs \
    --flow-arn "$FLOW_ARN" \
    --outputs "Name=$OUTPUT_NAME,Protocol=zixi-
push,Destination=198.51.100.11,Port=1024,StreamId=ZixiAwardsOutput" 2>&1); then
    echo "$add_output_output"
else
    handle_error "Failed to add output to flow"
fi

# Extract the output ARN
output_arn=""
output_arn=$(echo "$add_output_output" | jq -r ".Output.OutputArn // empty" 2>/dev/
null)
if [ -z "$output_arn" ]; then output_arn=$(echo "$add_output_output" | grep -o
 '"OutputArn": "[^"]*' | head -1 | cut -d'"' -f4); fi
if [ -z "$output_arn" ]; then
    echo "WARNING: Failed to extract output ARN from output"
else
    OUTPUT_ARN="$output_arn"
    echo "Output ARN: $OUTPUT_ARN"
    tag_mediaconnect_resource "$OUTPUT_ARN"
fi

# Step 4: Grant an entitlement
echo "Step 4: Granting an entitlement..."
grant_entitlement_output=""
if grant_entitlement_output=$(aws mediaconnect grant-flow-entitlements \
    --flow-arn "$FLOW_ARN" \
    --entitlements "Name=$ENTITLEMENT_NAME,Subscribers=222233334444" 2>&1); then
    echo "$grant_entitlement_output"
else
    handle_error "Failed to grant entitlement"
fi

# Extract the entitlement ARN
entitlement_arn=""
entitlement_arn=$(echo "$grant_entitlement_output" | jq -r
 '.Entitlement.EntitlementArn // empty' 2>/dev/null)
```

```
if [ -z "$entitlement_arn" ]; then
    entitlement_arn=$(echo "$grant_entitlement_output" | grep -o '"EntitlementArn":
 "[^"]*' | head -1 | cut -d'"' -f4)
fi
if [ -z "$entitlement_arn" ]; then
    echo "WARNING: Failed to extract entitlement ARN from output"
else
    ENTITLEMENT_ARN="$entitlement_arn"
    echo "Entitlement ARN: $ENTITLEMENT_ARN"
    tag_mediaconnect_resource "$ENTITLEMENT_ARN"
fi

# Step 5: List entitlements to share with affiliates
echo "Step 5: Listing entitlements for the flow..."
describe_flow_output=""
if describe_flow_output=$(aws mediaconnect describe-flow --flow-arn "$FLOW_ARN" --
query "Flow.Entitlements" 2>&1); then
    echo "Entitlements for the flow:"
    echo "$describe_flow_output"
else
    handle_error "Failed to describe flow"
fi

# Display information to share with affiliates
echo ""
echo "Information to share with your Philadelphia affiliate:"
echo "Entitlement ARN: ${ENTITLEMENT_ARN:-N/A}"
echo "AWS Region: $AWS_REGION"

# Display resource summary
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "The following resources were created:"
echo "1. Flow: $FLOW_NAME (ARN: $FLOW_ARN)"
echo "2. Output: $OUTPUT_NAME (ARN: ${OUTPUT_ARN:-N/A})"
echo "3. Entitlement: $ENTITLEMENT_NAME (ARN: ${ENTITLEMENT_ARN:-N/A})"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."
```

```
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddFlowOutputs
  - CreateFlow
  - DeleteFlow
  - DescribeAvailabilityZones
  - DescribeFlow
  - GrantFlowEntitlements
  - ListFlows
  - StopFlow

Exemplos de uso do Amazon MSK AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon MSK.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos de streaming gerenciado

O exemplo de código a seguir mostra como:

- Criar um cluster do Amazon MSK.
- Criar permissões do IAM para acessar o MSK.
- Criar uma máquina cliente
- Obter os agentes de bootstrap.

- Configurar a máquina cliente.
- Crie um tópico e produce/consume dados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group: $CLIENT_SG_ID";
 fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi
```

```
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id" &>/
dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated" &>/
dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
```

```
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o "\"GroupId
\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing $sg_id,
 removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
```

```
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing $sg_id,
 removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No outbound
 all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed to
 terminate instance"
        echo "Waiting for instance to terminate..."
```

```
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
"Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group" "$CLIENT_SG_ID";
then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete key
pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
```

```
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
 profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
 "$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
 delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
 "$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to delete
 policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")
```

```
    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az (subnet:
 $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
```

```
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "================================================"

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default VPC
 in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-az,Values=true"
 \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
```

```
    handle_error "Not enough subnets available in the default VPC. Need at least 3
 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=$DEFAULT_VPC_ID"
 \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\", \"ClientSubnets
\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\", \"${SUBNET_ARRAY[2]}\"],
 \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-gs
 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLUSTER_ARN" ]; then
```

```
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --query
 "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi
```

```
if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
```

```
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs
```

```
# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "================================================"

# Step 3: Create a client machine
```

```
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@]; then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' 2>&1)
```

```
# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"
```

```
# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
```

```
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-${RANDOM_SUFFIX}},
{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-streaming-for-
apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response: $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
```

```
if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
```

```
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
```

```
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-
auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/$LATEST_VERSION/
aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
```

```
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
```

```
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "================================================"

# Display summary of created resources
echo ""
echo "================================================"
echo "RESOURCE SUMMARY"
echo "================================================"
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
```

```
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
```

```
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
```

```
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair
  - DeletePolicy
  - DeleteRole
  - DeleteSecurityGroup
  - DescribeAvailabilityZones
  - DescribeCluster
  - DescribeImages
  - DescribeInstanceTypeOfferings
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups

- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Exemplos de Netuno usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Neptune.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução aos bancos de dados de grafos

O exemplo de código a seguir mostra como:

- Criar uma VPC para um banco de dados do Neptune.
- Criar sub-redes em várias zonas de disponibilidade.
- Configurar a segurança do banco de dados do Neptune.
- Criar um grupo de sub-redes do Neptune.
- Criar uma instância e um cluster de banco de dados do Neptune.
- Adicionar dados a um banco de dados de grafos.
- Consultar um banco de dados de grafos.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Neptune Getting Started Script
# This script creates an Amazon Neptune database cluster and demonstrates basic
 operations

# Set up logging
LOG_FILE="neptune-setup.log"
echo "Starting Neptune setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
```

```
    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..." | tee -a "$LOG_FILE"

    # Only attempt to delete resources that were successfully created
    if [[ -n "$DB_INSTANCE_ID" ]]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"
        log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"
    fi

    if [[ -n "$DB_CLUSTER_ID" ]]; then
        echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --skip-final-snapshot"
    fi

    if [[ -n "$DB_SUBNET_GROUP" ]]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"
    fi

    if [[ -n "$SECURITY_GROUP_ID" ]]; then
        echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
    fi

    if [[ -n "$SUBNET_IDS" ]]; then
        for SUBNET_ID in $SUBNET_IDS; do
            echo "Deleting subnet $SUBNET_ID..." | tee -a "$LOG_FILE"
            log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET_ID"
        done
    fi
```

```
    if [[ -n "$VPC_ID" ]]; then
        echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"
    fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
VPC_NAME="neptune-vpc-$RANDOM_ID"
DB_SUBNET_GROUP="neptune-subnet-group-$RANDOM_ID"
DB_CLUSTER_ID="neptune-cluster-$RANDOM_ID"
DB_INSTANCE_ID="neptune-instance-$RANDOM_ID"
SG_NAME="neptune-sg-$RANDOM_ID"

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "VPC Name: $VPC_NAME" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "DB Cluster ID: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "DB Instance ID: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "Security Group Name: $SG_NAME" | tee -a "$LOG_FILE"

# Step 1: Create VPC
echo "Creating VPC..." | tee -a "$LOG_FILE"
VPC_OUTPUT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=$VPC_NAME},{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]" --output json)
check_error "$VPC_OUTPUT" $? "Failed to create VPC"

VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID" | tee -a "$LOG_FILE"

# Enable DNS support for the VPC
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support"
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames"

# Step 2: Create Internet Gateway and attach to VPC
echo "Creating Internet Gateway..." | tee -a "$LOG_FILE"
IGW_OUTPUT=$(aws ec2 create-internet-gateway --tag-specifications
 'ResourceType=internet-gateway,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$IGW_OUTPUT" $? "Failed to create Internet Gateway"

IGW_ID=$(echo "$IGW_OUTPUT" | grep -o '"InternetGatewayId": "[^"]*' | cut -d'"' -f4)
echo "Internet Gateway created with ID: $IGW_ID" | tee -a "$LOG_FILE"
```

```
log_cmd "aws ec2 attach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"

# Step 3: Create subnets in different AZs
echo "Creating subnets..." | tee -a "$LOG_FILE"

# Get available AZs
AZ_OUTPUT=$(aws ec2 describe-availability-zones --output json)
check_error "$AZ_OUTPUT" $? "Failed to get availability zones"

# Extract first 3 AZ names
AZ1=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -1)
AZ2=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -2 |
 tail -1)
AZ3=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -3 |
 tail -1)

# Create 3 subnets in different AZs
SUBNET1_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.1.0/24 --availability-zone $AZ1 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET1_OUTPUT" $? "Failed to create subnet 1"
SUBNET1_ID=$(echo "$SUBNET1_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET2_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.2.0/24 --availability-zone $AZ2 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET2_OUTPUT" $? "Failed to create subnet 2"
SUBNET2_ID=$(echo "$SUBNET2_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET3_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.3.0/24 --availability-zone $AZ3 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET3_OUTPUT" $? "Failed to create subnet 3"
SUBNET3_ID=$(echo "$SUBNET3_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET_IDS="$SUBNET1_ID $SUBNET2_ID $SUBNET3_ID"
echo "Created subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"

# Step 4: Create route table and add route to Internet Gateway
```

```
echo "Creating route table..." | tee -a "$LOG_FILE"
ROUTE_TABLE_OUTPUT=$(aws ec2 create-route-table --vpc-id $VPC_ID --tag-
specifications 'ResourceType=route-table,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$ROUTE_TABLE_OUTPUT" $? "Failed to create route table"

ROUTE_TABLE_ID=$(echo "$ROUTE_TABLE_OUTPUT" | grep -o '"RouteTableId": "[^"]*' | cut
 -d'"' -f4)
echo "Route table created with ID: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"

# Add route to Internet Gateway
log_cmd "aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-cidr-
block 0.0.0.0/0 --gateway-id $IGW_ID"

# Associate route table with subnets
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET1_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET2_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET3_ID"

# Step 5: Create security group
echo "Creating security group..." | tee -a "$LOG_FILE"
SG_OUTPUT=$(aws ec2 create-security-group --group-name $SG_NAME --description
 "Security group for Neptune" --vpc-id $VPC_ID --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SG_OUTPUT" $? "Failed to create security group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
echo "Security group created with ID: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"

# Add inbound rule for Neptune port (8182)
# Note: In production, you should restrict this to specific IP ranges
echo "Adding security group rule for Neptune port 8182..." | tee -a "$LOG_FILE"
log_cmd "aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID --
protocol tcp --port 8182 --cidr 10.0.0.0/16"

# Step 6: Create DB subnet group
echo "Creating DB subnet group..." | tee -a "$LOG_FILE"
DB_SUBNET_GROUP_OUTPUT=$(aws neptune create-db-subnet-group --db-subnet-group-
name $DB_SUBNET_GROUP --db-subnet-group-description "Subnet group for Neptune" --
```

```
subnet-ids $SUBNET1_ID $SUBNET2_ID $SUBNET3_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_SUBNET_GROUP_OUTPUT" $? "Failed to create DB subnet group"
echo "DB subnet group created: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"

# Step 7: Create Neptune DB cluster
echo "Creating Neptune DB cluster..." | tee -a "$LOG_FILE"
DB_CLUSTER_OUTPUT=$(aws neptune create-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --engine neptune --vpc-security-group-ids $SECURITY_GROUP_ID
 --db-subnet-group-name $DB_SUBNET_GROUP --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_CLUSTER_OUTPUT" $? "Failed to create Neptune DB cluster"
echo "Neptune DB cluster created: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"

# Step 8: Create Neptune DB instance
echo "Creating Neptune DB instance..." | tee -a "$LOG_FILE"
DB_INSTANCE_OUTPUT=$(aws neptune create-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --db-instance-class db.r5.large --engine neptune --db-
cluster-identifier $DB_CLUSTER_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_INSTANCE_OUTPUT" $? "Failed to create Neptune DB instance"
echo "Neptune DB instance created: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"

# Step 9: Wait for the DB instance to become available
echo "Waiting for Neptune DB instance to become available..." | tee -a "$LOG_FILE"
log_cmd "aws neptune wait db-instance-available --db-instance-identifier
 $DB_INSTANCE_ID"

# Step 10: Get the Neptune endpoint
echo "Getting Neptune endpoint..." | tee -a "$LOG_FILE"
ENDPOINT_OUTPUT=$(aws neptune describe-db-clusters --db-cluster-identifier
 $DB_CLUSTER_ID --output json)
check_error "$ENDPOINT_OUTPUT" $? "Failed to get Neptune endpoint"

NEPTUNE_ENDPOINT=$(echo "$ENDPOINT_OUTPUT" | grep -o '"Endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Neptune endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"

# Step 11: Display information about how to connect to Neptune
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "NEPTUNE SETUP COMPLETE" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Neptune Endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"
```

```
echo "Port: 8182" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To query your Neptune database using Gremlin, you can use curl:" | tee -a
 "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.V().limit(1)\"}' https://
$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To add data to your graph:" | tee -a "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.addV(\\\"person\\\").property(\\\"name\\\",
 \\\"Howard\\\")\"}' https://$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Note: You may need to configure your client machine to access the Neptune
 instance within the VPC." | tee -a "$LOG_FILE"
echo "For production use, consider using an EC2 instance in the same VPC or setting
 up a bastion host." | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 12: List all created resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "RESOURCES CREATED" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "VPC: $VPC_ID" | tee -a "$LOG_FILE"
echo "Internet Gateway: $IGW_ID" | tee -a "$LOG_FILE"
echo "Subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"
echo "Route Table: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"
echo "Security Group: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "Neptune DB Cluster: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "Neptune DB Instance: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 13: Ask if user wants to clean up resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Do you want to clean up all created resources? (y/n): " | tee -a "$LOG_FILE"
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..." | tee -a "$LOG_FILE"

    # Delete DB instance
```

```
    echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-instance --db-instance-identifier $DB_INSTANCE_ID
 --skip-final-snapshot"

    # Wait for DB instance to be deleted
    echo "Waiting for DB instance to be deleted..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"

    # Delete DB cluster
    echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-cluster --db-cluster-identifier $DB_CLUSTER_ID --
skip-final-snapshot"

    # Wait for DB cluster to be deleted (no specific wait command for this, so we'll
 sleep)
    echo "Waiting for DB cluster to be deleted..." | tee -a "$LOG_FILE"
    sleep 60

    # Delete DB subnet group
    echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"

    # Delete security group
    echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"

    # Detach and delete internet gateway
    echo "Detaching and deleting internet gateway $IGW_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 detach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"
    log_cmd "aws ec2 delete-internet-gateway --internet-gateway-id $IGW_ID"

    # Delete subnets
    echo "Deleting subnets..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET1_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET2_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET3_ID"

    # Delete route table
    echo "Deleting route table $ROUTE_TABLE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-route-table --route-table-id $ROUTE_TABLE_ID"
```

```
    # Delete VPC
    echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"

    echo "Cleanup complete!" | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can delete them manually later." |
 tee -a "$LOG_FILE"
    echo "See the list of resources above for reference." | tee -a "$LOG_FILE"
fi

echo "Script completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateInternetGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteInternetGateway
  - DeleteRouteTable
  - DeleteSecurityGroup
  - DeleteSubnet
  - DeleteVpc

- DescribeAvailabilityZones
- DescribeDbClusters
- DetachInternetGateway
- ModifyVpcAttribute
- Wait

AWS Payment Cryptography exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS Payment Cryptography.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução à criptografia de segurança de pagamento

O exemplo de código a seguir mostra como:

- Criar uma chave.
- Verificar o valor CVV2.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Payment Cryptography Getting Started Script
# This script demonstrates how to use AWS Payment Cryptography to create a key,
# generate and verify CVV2 values, and clean up resources.

set -euo pipefail

# Initialize log file with secure permissions
LOG_FILE="payment-cryptography-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "AWS Payment Cryptography Tutorial - $(date)" > "$LOG_FILE"

# Function to log messages
log() {
    local message="$1"
    echo "$(date +"%Y-%m-%d %H:%M:%S") - $message" | tee -a "$LOG_FILE"
}

# Function to handle errors
handle_error() {
    local error_message="$1"
    log "ERROR: $error_message"
    log "Script failed. Please check the log file: $LOG_FILE"

    echo ""
    echo "============================================"
    echo "ERROR ENCOUNTERED"
    echo "============================================"
    echo "The script encountered an error: $error_message"
    echo "Resources created will be listed below."
    echo ""

    if [ -n "${KEY_ARN:-}" ]; then
        echo "Key ARN: $KEY_ARN"
    fi

    exit 1
}

# Function to check command output for errors
check_error() {
```

```
    local output="$1"
    local command="$2"

    if echo "$output" | grep -iq "error\|exception\|fail"; then
        handle_error "Command failed: $command"
    fi
}

# Validate AWS CLI is available and credentials are configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not properly configured"
fi

log "Starting AWS Payment Cryptography tutorial"

# Step 1: Create a key
log "Step 1: Creating a card verification key (CVK)"
if ! KEY_OUTPUT=$(aws payment-cryptography create-key \
  --exportable \
  --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyMod
 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-payment-cryptography-gs
 2>&1); then
    handle_error "Failed to create key"
fi

echo "$KEY_OUTPUT"
check_error "$KEY_OUTPUT" "create-key"

# Extract the Key ARN from the output
KEY_ARN=$(echo "$KEY_OUTPUT" | grep -o '"KeyArn": "[^"]*' | cut -d'"' -f4)

if [ -z "$KEY_ARN" ]; then
    handle_error "Failed to extract Key ARN from output"
fi

log "Successfully created key with ARN: $KEY_ARN"

# Step 2: Generate a CVV2 value
```

```
log "Step 2: Generating a CVV2 value"
if ! CVV2_OUTPUT=$(aws payment-cryptography-data generate-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --generation-attributes CardVerificationValue2={CardExpiryDate=0123} 2>&1); then
    handle_error "Failed to generate CVV2 value"
fi

echo "$CVV2_OUTPUT"
check_error "$CVV2_OUTPUT" "generate-card-validation-data"

# Extract the CVV2 value from the output - updated to use ValidationData instead of
 CardDataValue
CVV2_VALUE=$(echo "$CVV2_OUTPUT" | grep -o '"ValidationData": "[^"]*' | cut -d'"' -
f4)

if [ -z "$CVV2_VALUE" ]; then
    handle_error "Failed to extract CVV2 value from output"
fi

log "Successfully generated CVV2 value"

# Step 3: Verify the CVV2 value
log "Step 3: Verifying the CVV2 value"
if ! VERIFY_OUTPUT=$(aws payment-cryptography-data verify-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data "$CVV2_VALUE" 2>&1); then
    handle_error "Failed to verify CVV2 value"
fi

echo "$VERIFY_OUTPUT"
check_error "$VERIFY_OUTPUT" "verify-card-validation-data"

log "Successfully verified CVV2 value"

# Step 4: Perform a negative test
log "Step 4: Performing a negative test with incorrect CVV2"
if aws payment-cryptography-data verify-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data 999 2>&1; then
```

```
    handle_error "Negative test did not fail as expected"
fi

log "Negative test completed successfully (verification failed as expected)"

# Display created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "Key ARN: $KEY_ARN"
echo ""

# Auto-confirm cleanup
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Proceeding with cleanup of all created resources..."

log "Step 5: Cleaning up resources"

# Delete the key
log "Deleting key: $KEY_ARN"
if ! DELETE_OUTPUT=$(aws payment-cryptography delete-key \
  --key-identifier "$KEY_ARN" 2>&1); then
    handle_error "Failed to delete key"
fi

echo "$DELETE_OUTPUT"
check_error "$DELETE_OUTPUT" "delete-key"

log "Key scheduled for deletion. Default waiting period is 7 days."
log "To cancel deletion before the waiting period ends, use:"
log "aws payment-cryptography restore-key --key-identifier $KEY_ARN"

echo ""
echo "============================================"
echo "CLEANUP COMPLETE"
echo "============================================"
echo "The key has been scheduled for deletion after the default waiting period (7
 days)."
echo "To cancel deletion before the waiting period ends, use:"
echo "aws payment-cryptography restore-key --key-identifier $KEY_ARN"
```

```
log "Tutorial completed successfully"
echo ""
echo "Tutorial completed successfully. See $LOG_FILE for details."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateKey
  - DeleteKey
  - RestoreKey

Exemplos do Amazon Pinpoint usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon Pinpoint.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos das notificações push

O exemplo de código a seguir mostra como:

- Criar uma aplicação.
- Habilitar canais de notificações por push.
- Enviar notificações por push
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail

# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
cleanup() {
    local exit_code=$?
```

```
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
        echo "ERROR: Command failed: $cmd" >&2
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
}
```

```
# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi

    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
    echo ""

    return 0
}

# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
        echo "jq is not available, using grep for JSON parsing"
```

```
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."

    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
        echo "  - mobiletargeting:CreateApp" >&2
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
    local app_name=$1
```

```
    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters" >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi

check_json_tools
validate_permissions

# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
fi
```

```
echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi

echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "==========================================="
echo "ENABLING FCM (GCM) CHANNEL"
echo "==========================================="
echo "Note: This is using a placeholder API key for demonstration purposes only."
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
```

```
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-demo-
only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real certificates)
echo ""
echo "============================================"
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================"
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --apns-
channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
    --application-id "$APP_ID" \
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
```

```
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-channel
 command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "==========================================="
echo "CREATING MESSAGE FILES"
echo "==========================================="

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
        "key1": "value1",
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
```

```
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF

echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
echo "To send real messages, you would need to replace the sample device tokens with
 actual ones."

# Step 5: Demonstrate how to send messages (this will fail with placeholder tokens)
echo ""
echo "============================================"
echo "DEMONSTRATING MESSAGE SENDING"
echo "============================================"
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
```

```
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
echo "Retrieving application details..."

echo "Executing: aws pinpoint get-app --application-id $APP_ID"
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
```

```
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
echo "2. Attempting to enable FCM and APNS channels (with placeholder credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
echo "5. Retrieving application details"
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
```

```
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps
  - GetCallerIdentity
  - SendMessages
  - UpdateApnsChannel
  - UpdateGcmChannel

Exemplos de uso do Amazon Polly AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon Polly.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução à síntese de conversão de texto em fala

O exemplo de código a seguir mostra como:

- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Polly Getting Started Script
# This script demonstrates how to use Amazon Polly with the AWS CLI

set -euo pipefail

# Set up logging
LOG_FILE="polly-tutorial.log"
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
WORK_DIR=$(mktemp -d)
trap 'cleanup_temp' EXIT

cleanup_temp() {
    rm -rf "$WORK_DIR"
}

echo "Starting Amazon Polly tutorial at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    # Use bash array to safely handle arguments
    bash -c "$1" 2>&1 | tee -a "$LOG_FILE" || return $?
}
```

```
# Function to check for errors
check_error() {
    if echo "$1" | grep -iq "error"; then
        echo "ERROR detected in output. Exiting script." | tee -a "$LOG_FILE"
        echo "$1" | tee -a "$LOG_FILE"
        return 1
    fi
    return 0
}

# Function to handle errors and cleanup
handle_error() {
    local line_number=$1
    echo "Error occurred at line $line_number. Attempting cleanup..." | tee -a
 "$LOG_FILE"
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "" | tee -a "$LOG_FILE"
    echo "============================================================" | tee -a
 "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "============================================================" | tee -a
 "$LOG_FILE"

    # Delete lexicon if it exists
    if [[ -n "${LEXICON_NAME:-}" ]]; then
        echo "Deleting lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
        if aws polly delete-lexicon --name "$LEXICON_NAME" 2>&1 | tee -a
 "$LOG_FILE"; then
            echo "Lexicon deleted successfully." | tee -a "$LOG_FILE"
        else
            echo "Warning: Failed to delete lexicon." | tee -a "$LOG_FILE"
        fi
    fi

    # Remove audio files
    for file in output.mp3 ssml-output.mp3 lexicon-output.mp3 example.pls; do
        if [[ -f "$file" ]]; then
            rm -f "$file"
            echo "Removed $file" | tee -a "$LOG_FILE"
```

```
        fi
    done

    echo "Cleanup complete." | tee -a "$LOG_FILE"
}

# Trap errors with line number
trap 'handle_error ${LINENO}' ERR

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    echo "AWS CLI is not installed. Please install it first." | tee -a "$LOG_FILE"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "AWS credentials are not configured. Please configure them first." | tee -a
 "$LOG_FILE"
    exit 1
fi

# Step 1: Verify Amazon Polly is available
echo "Step 1: Verifying Amazon Polly availability" | tee -a "$LOG_FILE"
if aws polly describe-voices --query 'Voices[0].Name' --output text &> /dev/null;
 then
    echo "Amazon Polly is available. Proceeding with tutorial." | tee -a "$LOG_FILE"
else
    echo "Amazon Polly is not available in your AWS CLI installation or region." |
 tee -a "$LOG_FILE"
    echo "Please update your AWS CLI to the latest version or check your region." |
 tee -a "$LOG_FILE"
    exit 1
fi

# Step 2: List available voices
echo "" | tee -a "$LOG_FILE"
echo "Step 2: Listing available voices" | tee -a "$LOG_FILE"
log_cmd "aws polly describe-voices --language-code en-US --output text --query
 'Voices[0:3].[Id, LanguageCode, Gender]'" || true

# Step 3: Basic text-to-speech conversion
echo "" | tee -a "$LOG_FILE"
echo "Step 3: Converting text to speech" | tee -a "$LOG_FILE"
```

```
OUTPUT_FILE="${WORK_DIR}/output.mp3"
POLLY_TEXT="Hello, welcome to Amazon Polly. This is a sample text to speech
 conversion."
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --text
 '$POLLY_TEXT' '$OUTPUT_FILE'" || true

if [[ -f "$OUTPUT_FILE" ]]; then
    echo "Successfully created output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$OUTPUT_FILE" output.mp3
else
    echo "Failed to create output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Step 4: Using SSML for enhanced speech
echo "" | tee -a "$LOG_FILE"
echo "Step 4: Using SSML for enhanced speech" | tee -a "$LOG_FILE"
SSML_OUTPUT="${WORK_DIR}/ssml-output.mp3"
SSML_TEXT='<speak>Hello! <break time="1s"/> This is a sample of <emphasis>SSML
 enhanced speech</emphasis>.</speak>'
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Matthew --text-
type ssml --text '$SSML_TEXT' '$SSML_OUTPUT'" || true

if [[ -f "$SSML_OUTPUT" ]]; then
    echo "Successfully created ssml-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$SSML_OUTPUT" ssml-output.mp3
else
    echo "Failed to create ssml-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Step 5: Working with lexicons
echo "" | tee -a "$LOG_FILE"
echo "Step 5: Working with lexicons" | tee -a "$LOG_FILE"

# Generate a random identifier for the lexicon (max 20 chars, alphanumeric only)
LEXICON_NAME="example$(openssl rand -hex 6 | cut -c 1-10)"
echo "Using lexicon name: $LEXICON_NAME" | tee -a "$LOG_FILE"

# Create a lexicon file
```

```
echo "Creating lexicon file..." | tee -a "$LOG_FILE"
LEXICON_FILE="${WORK_DIR}/example.pls"
cat > "$LEXICON_FILE" << 'EOF'
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
      xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
      alphabet="ipa"
      xml:lang="en-US">
  <lexeme>
    <grapheme>AWS</grapheme>
    <alias>Amazon Web Services</alias>
  </lexeme>
</lexicon>
EOF

# Upload the lexicon
echo "Uploading lexicon..." | tee -a "$LOG_FILE"
LEXICON_ARN=$(aws polly put-lexicon --name "$LEXICON_NAME" --content
 file://"$LEXICON_FILE" --query 'LexiconArn' --output text 2>&1) || true
if [[ -n "$LEXICON_ARN" && "$LEXICON_ARN" != "" && ! "$LEXICON_ARN" =~ error ]];
 then
    echo "Lexicon uploaded with ARN: $LEXICON_ARN" | tee -a "$LOG_FILE"
    aws polly tag-resource --resource-arn "$LEXICON_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-polly-gs 2>&1 | tee -a
 "$LOG_FILE" || true
else
    echo "Lexicon uploaded." | tee -a "$LOG_FILE"
fi

# List available lexicons
echo "Listing available lexicons..." | tee -a "$LOG_FILE"
log_cmd "aws polly list-lexicons --output text --query 'Lexicons[*].[Name]'" || true

# Get details about the lexicon
echo "Getting details about the lexicon..." | tee -a "$LOG_FILE"
log_cmd "aws polly get-lexicon --name '$LEXICON_NAME' --output text --query
 'Lexicon.Name'" || true

# Use the lexicon when synthesizing speech
echo "Using the lexicon for speech synthesis..." | tee -a "$LOG_FILE"
LEXICON_OUTPUT="${WORK_DIR}/lexicon-output.mp3"
```

```
LEXICON_TEXT="I work with AWS every day."
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --
lexicon-names '$LEXICON_NAME' --text '$LEXICON_TEXT' '$LEXICON_OUTPUT'" || true

if [[ -f "$LEXICON_OUTPUT" ]]; then
    echo "Successfully created lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$LEXICON_OUTPUT" lexicon-output.mp3
else
    echo "Failed to create lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Summary of created resources
echo "" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "TUTORIAL SUMMARY" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "Created resources:" | tee -a "$LOG_FILE"
echo "1. Lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
echo "2. Audio files:" | tee -a "$LOG_FILE"
echo "   - output.mp3" | tee -a "$LOG_FILE"
echo "   - ssml-output.mp3" | tee -a "$LOG_FILE"
echo "   - lexicon-output.mp3" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Cleanup with auto-confirmation
echo "" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"
cleanup

echo "" | tee -a "$LOG_FILE"
echo "Tutorial completed successfully!" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- DeleteLexicon
- DescribeVoices
- GetLexicon
- Help
- ListLexicons
- PutLexicon
- SynthesizeSpeech

Exemplos do Amazon RDS usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon RDS.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criação de uma instância de banco de dados relacional

O exemplo de código a seguir mostra como:

- Configurar componentes de rede.
- Criar um grupo de sub-redes de banco de dados
- Criar uma instância de banco de dados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/latest/
UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
```

```
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ]; then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi
```

```
echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found ${#SUBNET_IDS[@]}
 subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
```

```
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id "$SECRET_NAME"
 --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -d'"'
 -f4)
```

```
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -d'"'
 -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
```

```
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
```

```
    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - Wait

Exemplos do Amazon Redshift usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon Redshift.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos de data warehouses de tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Use o gerenciador de segredos CreateSecret
- Use o gerenciador de segredos DeleteSecret
- Use o gerenciador de segredos GetSecretValue
- Use redshift CreateNamespace
- Use redshift CreateWorkgroup
- Use redshift DeleteNamespace
- Use iam CreateRole

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq dependency)
# This script creates a Redshift Serverless environment, loads sample data, and runs
 queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)
```

```
    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")
```

```
    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status: $status.
 Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
```

```
  echo "============================================"
  echo "CLEANUP CONFIRMATION"
  echo "============================================"
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --workgroup-
name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --namespace-
name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
```

```
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin credentials
 for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()
```

```
echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
```

```
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless get-
namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
```

```
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless get-
workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
```

```
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
```

```
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
```

```
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
```

```
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID" 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
```

```
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time total
 sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price desc)
 as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
```

```
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID" 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
```

```
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole

- GetSecretValue
- GetWorkgroup
- PutRolePolicy
- UpdateNamespace

Conceitos básicos de clusters de data warehouse provisionados

O exemplo de código a seguir mostra como:

- Criar um cluster do Redshift.
- Criar um perfil do IAM para acessar o S3.
- Criar tabelas e carregar dados.
- Executar consultas de exemplo.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
```

```
# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-final-
cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/dev/
null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}
```

```
# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""

    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --query
 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id "$statement_id"
 --query 'Error' --output text 2>/dev/null || echo "Unknown error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1
```

```
    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus" \
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and try
 again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
```

```
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi

# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"
```

```
echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed to
 attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi
```

```
echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer not
 null, eventid integer not null, dateid smallint not null sortkey, qtysold smallint
 not null, pricepaid decimal(8,2), commission decimal(8,2), saletime timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint not
 null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"
```

```
# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi

echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt' DELIMITER
 '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
```

```
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid AND
 caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Cleaning up all created resources..."
```

```
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

Exemplos do Amazon S3 usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function s3_getting_started() {
  {
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
      cd bucket-lifecycle-operations || exit
```

```
    source ./bucket_operations.sh
    cd ..
  fi
}

echo_repeat "*" 88
echo "Welcome to the Amazon S3 getting started demo."
echo_repeat "*" 88
  echo "A unique bucket will be created by appending a Universally Unique
Identifier to a bucket name prefix."
  echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
  get_input
  bucket_name_prefix=$get_input_result
local bucket_name
bucket_name=$(generate_random_name "$bucket_name_prefix")

local region_code
region_code=$(aws configure get region)

if create_bucket -b "$bucket_name" -r "$region_code"; then
  echo "Created demo bucket named $bucket_name"
else
  errecho "The bucket failed to create. This demo will exit."
  return 1
fi

local file_name
while [ -z "$file_name" ]; do
  echo -n "Enter a file you want to upload to your bucket: "
  get_input
  file_name=$get_input_result

  if [ ! -f "$file_name" ]; then
    echo "Could not find file $file_name. Are you sure it exists?"
    file_name=""
  fi
done

local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
```

```
    echo "Uploaded file $file_name into bucket $bucket_name with key $key."
  else
    result=1
  fi

  local destination_file
  destination_file="$file_name.download"
  if yes_no_input "Would you like to download $key to the file $destination_file?
 (y/n) "; then
    if download_object_from_bucket "$bucket_name" "$destination_file" "$key"; then
      echo "Downloaded $key in the bucket $bucket_name to the file
 $destination_file."
    else
      result=1
    fi
  fi

  if yes_no_input "Would you like to copy $key a new object key in your bucket? (y/
n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
      echo "Copied $key in the bucket $bucket_name to the  $to_key."
    else
      result=1
    fi
  fi

  local bucket_items
  bucket_items=$(list_items_in_bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    result=1
  fi

  echo "Your bucket contains the following items."
  echo -e "Name\t\tSize"
  echo "$bucket_items"

  if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
 (y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)
```

```
    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
      echo "The following items were deleted from the bucket $bucket_name"
      echo "$bucket_items"
    else
      result=1
    fi

    if delete_bucket "$bucket_name"; then
      echo "Deleted the bucket $bucket_name"
    else
      result=1
    fi
  fi

  return $result
}
```

As funções do Amazon S3 usadas nesse cenário.

```
###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
```

```
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
```

```
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}

##############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
```

```
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

##############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

##############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
```

```
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response
```

```
  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
```

```
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n$response"
    return 1
  fi
}

##############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects

- GetObject
- ListObjectsV2
- PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte CopyObject em Referência de AWS CLI comandos.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
```

```
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
```

```
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)
```

```
  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte CreateBucket em Referência de AWS CLI comandos.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte DeleteBucket em Referência de AWS CLI comandos.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
```

```
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - The key (file name) in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_item_in_bucket() {
  local bucket_name=$1
  local key=$2
  local response

  response=$(aws s3api delete-object \
    --bucket "$bucket_name" \
    --key "$key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte DeleteObject em Referência de AWS CLI comandos.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n$response"
```

```
    return 1
  fi
}
```

- Para obter detalhes da API, consulte DeleteObjects em Referência de AWS CLI comandos.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
##############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte GetObject em Referência de AWS CLI comandos.

`HeadBucket`

O código de exemplo a seguir mostra como usar `HeadBucket`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
##############################################################################
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#       $1 - The name of the bucket to check.
```

```
#
# Returns:
#       0 - If the bucket already exists.
#       1 - If the bucket doesn't exist.
###############################################################################
function bucket_exists() {
  local bucket_name
  bucket_name=$1

  # Check whether the bucket already exists.
  # We suppress all output - we're interested only in the return code.

  if aws s3api head-bucket \
    --bucket "$bucket_name" \
    >/dev/null 2>&1; then
    return 0 # 0 in Bash script means true.
  else
    return 1 # 1 in Bash script means false.
  fi
}
```

- Para obter detalhes da API, consulte HeadBucket em Referência de AWS CLI comandos.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte ListObjects V2 na Referência de AWS CLI Comandos.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")
```

```
  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- Para obter detalhes da API, consulte PutObject em Referência de AWS CLI comandos.

Cenários

Introdução aos clusters de processamento de big data

O exemplo de código a seguir mostra como:

- Criar um par de chaves do EC2.
- Configurar o armazenamento e preparar uma aplicação.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
```

```
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null || true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ]; then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
```

```
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
```

```
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-configuration
 Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to find
 the top 10 establishments
    with the most Red violations from 2006 to 2020.
```

```
    :param data_source: The URI of your food establishment data CSV, such as 's3://
emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-tutorial-
bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
```

```
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output text
 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
```

```
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi
```

```
echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])
```

```
# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID"
 --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv" |
 head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
```

```
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv" |
 head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddSteps
  - Cp

- CreateCluster
- CreateDefaultRoles
- CreateKeyPair
- DescribeCluster
- DescribeKeyPairs
- DescribeStep
- Ls
- Mb
- Rb
- Rm
- ssh
- TerminateClusters
- Wait

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
```

```
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
```

```
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
```

```
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
```

```
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
```

```
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
```

```
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
```

```
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
```

```
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
```

```
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic

- DeleteBucket
- DeleteConfigurationRecorder
- DeleteDeliveryChannel
- DeleteRole
- DeleteRolePolicy
- DeleteTopic
- DescribeConfigurationRecorderStatus
- DescribeConfigurationRecorders
- DescribeDeliveryChannels
- DetachRolePolicy
- GetCallerIdentity
- PutConfigurationRecorder
- PutDeliveryChannel
- PutPublicAccessBlock
- PutRolePolicy
- Rm
- StartConfigurationRecorder
- StopConfigurationRecorder

Introdução à extração de texto de documentos

O exemplo de código a seguir mostra como:

- Criar um bucket do S3.
- Fazer upload de um documento no S3.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "Amazon Textract Getting Started Tutorial"
echo "==================================================="
echo "This script will guide you through using Amazon Textract to analyze document
 text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*/\1=***REDACTED***/g'
        cleanup_on_error
        exit 1
    fi
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to delete
 document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ]; then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\).*/
\1=***REDACTED***/g'
    exit 1
fi
```

```
# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
 default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
 2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
```

```
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
 {"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text 50,50
 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "./$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256 2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
```

```
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ./$DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
```

```
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "==================================================="
echo "Analysis Summary"
echo "==================================================="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk '{print
 $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l || echo
 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l || echo
 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l || echo
 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
 echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l || echo
 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"' |
 wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' | wc -
l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "==================================================="
echo "RESOURCES CREATED"
```

```
echo "====================================================="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "====================================================="
echo "CLEANUP CONFIRMATION"
echo "====================================================="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-results.json)
 has been kept."

echo ""
echo "====================================================="
echo "Tutorial complete!"
echo "====================================================="
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AnalyzeDocument
  - Cp
  - Help
  - Mb
  - Rb
  - Rm

Conceitos básicos de arquivos de atributos de machine learning

O exemplo de código a seguir mostra como:

- Configurar permissões do IAM.
- Crie uma função SageMaker de execução
- Criar grupo de atributos.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""
```

```
# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
```

```
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
```

```
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
```

```
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi
```

```
# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
```

```
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"
```

```
CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
```

```
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)
```

```
if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
```

```
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
```

```
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS" |
 grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
```

```
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================="
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================="
    echo "CLEANUP CONFIRMATION"
    echo "============================================="
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi
```

```
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Introdução ao armazenamento de objetos

O exemplo de código a seguir mostra como:

- Criar seu primeiro bucket do S3.
- Fazer upload de um objeto.
- Habilitar o versionamento.
- Configurar a criptografia padrão.
- Adicionar tags a um bucket.
- Listar objetos e versões.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail

# ===============================================================================
# Prerequisites check
# ===============================================================================

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
 AWS_DEFAULT_REGION."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
    exit 1
fi

# ===============================================================================
# Setup: logging, temp directory, resource tracking
# ===============================================================================

UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
```

```
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || true)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="s3api-${UNIQUE_ID}"
fi

TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
LOG_FILE="${TEMP_DIR}/s3-gettingstarted.log"
CREATED_RESOURCES=()

exec > >(tee -a "$LOG_FILE") 2>&1

echo "============================================="
echo "S3 Getting Started"
echo "============================================="
echo "Bucket name: ${BUCKET_NAME}"
echo "Temp directory: ${TEMP_DIR}"
echo "Log file: ${LOG_FILE}"
echo ""

# ============================================================================
# Helper functions
# ============================================================================

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0
```

```
    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
 then
        return 0
    fi

    echo "$versions" | jq -r '.[] | "\(.Key)\t\(.VersionId)"' 2>/dev/null | while
 IFS=$'\t' read -r key version_id; do
        if [ -n "$key" ] && [ "$key" != "null" ]; then
            aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
 "$version_id" >/dev/null 2>&1 || true
        fi
    done

    return 0
}

# ============================================================================
# Error handling and cleanup functions
# ============================================================================

cleanup() {
    echo ""
    echo "============================================="
    echo "CLEANUP"
    echo "============================================="

    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[].
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[].
{Key:Key,VersionId:VersionId}" || true

        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
```

```
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null;
 then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "============================================="
    echo "ERROR on line ${line_number}"
    echo "============================================="
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@]}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# ==============================================================================
# Step 1: Create a bucket
# ==============================================================================

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
```

```
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME"
            exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
 2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
        fi
    fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag bucket"
    fi
fi
echo ""

# ==============================================================================
# Step 2: Upload a sample text file
# ==============================================================================

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
```

```
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# ==============================================================================
# Step 3: Download the object
# ==============================================================================

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then
    echo "ERROR: Failed to download sample.txt"
    exit 1
fi
echo "Downloaded to: ${DOWNLOAD_FILE}"
echo "Contents:"
cat "$DOWNLOAD_FILE"
echo ""

# ==============================================================================
# Step 4: Copy the object to a folder prefix
# ==============================================================================

echo "Step 4: Copying object to a folder prefix..."

if ! aws s3api copy-object \
    --bucket "$BUCKET_NAME" \
    --copy-source "${BUCKET_NAME}/sample.txt" \
    --key "backup/sample.txt" \
```

```
    --server-side-encryption AES256 \
    --metadata-directive COPY >/dev/null 2>&1; then
    echo "ERROR: Failed to copy object to backup/sample.txt"
    exit 1
fi
echo "Object copied to backup/sample.txt."
echo ""

# ==============================================================================
# Step 5: Enable versioning and upload a second version
# ==============================================================================

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \
    --versioning-configuration Status=Enabled >/dev/null 2>&1; then
    echo "ERROR: Failed to enable versioning"
    exit 1
fi
echo "Versioning enabled."

echo "Uploading a second version of sample.txt..."
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is version 2 of the sample file.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# ==============================================================================
# Step 6: Configure SSE-S3 encryption
# ==============================================================================

echo "Step 6: Configuring SSE-S3 default encryption..."
```

```
if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi
echo "SSE-S3 encryption configured."
echo ""

# ==============================================================================
# Step 7: Block all public access
# ==============================================================================

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# ==============================================================================
# Step 8: Configure bucket logging
# ==============================================================================

echo "Step 8: Configuring bucket logging..."
```

```
LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
 true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
 2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \
        --bucket "$LOG_TARGET_BUCKET" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag log bucket"
    fi

    aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-write
 2>/dev/null || true

    if ! aws s3api put-bucket-logging \
        --bucket "$BUCKET_NAME" \
        --bucket-logging-status '{
            "LoggingEnabled": {
                "TargetBucket": "'$LOG_TARGET_BUCKET'",
                "TargetPrefix": "logs/"
            }
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to configure bucket logging"
```

```
    else
        echo "Bucket logging configured."
    fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

# ==============================================================================
# Step 9: Tag the bucket
# ==============================================================================

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            },
            {
                "Key": "Environment",
                "Value": "Tutorial"
            },
            {
                "Key": "Project",
                "Value": "S3-GettingStarted"
            },
            {
                "Key": "ManagedBy",
                "Value": "Bash-Tutorial"
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to tag bucket"
    exit 1
fi
echo "Bucket tagged."
```

```
echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# ==============================================================================
# Step 10: List objects and versions
# ==============================================================================

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# ==============================================================================
# Step 11: Cleanup
# ==============================================================================

echo ""
echo "============================================"
echo "TUTORIAL COMPLETE"
echo "============================================"
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "============================================"
```

```
echo "CLEANUP"
echo "==========================================="
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - HeadObject
  - ListObjectVersions
  - ListObjectsV2
  - PutBucketEncryption
  - PutBucketTagging
  - PutBucketVersioning
  - PutObject
  - PutPublicAccessBlock

Introdução à análise de consulta

O exemplo de código a seguir mostra como:

- Criar um bucket do S3 para resultados de consulta.
- Criar um banco de dados.
- Criar uma tabela.
- Executar uma consulta.
- Criar e usar consultas nomeadas.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
```

```
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] || [ ${#bucket_name}
 -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi
```

```
# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
```

```
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~ ^[a-
z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION" 2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
athena-gs}]'

    # Security: Enable S3 bucket encryption with KMS validation
    echo "Enabling default encryption on S3 bucket..."
    if ! aws s3api put-bucket-encryption \
        --bucket "$S3_BUCKET" \
        --server-side-encryption-configuration '{
            "Rules": [{
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }]
        }' 2>&1; then
        echo "Warning: Could not enable encryption on bucket"
    fi

    # Security: Block public access
    echo "Blocking public access to S3 bucket..."
    if ! aws s3api put-public-access-block \
        --bucket "$S3_BUCKET" \
        --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1; then
        echo "Warning: Could not block public access on bucket"
    fi
```

```
    # Security: Enable versioning for data protection
    echo "Enabling versioning on S3 bucket..."
    if ! aws s3api put-bucket-versioning \
        --bucket "$S3_BUCKET" \
        --versioning-configuration Status=Enabled 2>&1; then
        echo "Warning: Could not enable versioning on bucket"
    fi

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
```

```
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
 "$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.$TABLE_NAME (
  \`Date\` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  \"input.regex\" = \"^(?!#)([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s
+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+[^\\\
\(]+[\\\\(]([^\\\\;]+).*\\\\%20([^\\\\/]+)[\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
```

```
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
```

```
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null ||
 echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
```

```
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
 "$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
```

```
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/dev/
null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -d'"' -
f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
```

```
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null ||
 echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
```

```
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
 "$NAMED_QUERY_ID" \
        --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
```

```
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Table deletion failed with status: $QUERY_STATUS"
                    break
                fi
                echo "Table deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \
        --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
```

```
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Empty and delete S3 bucket (only if not shared)
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi

        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
        if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
        else
            echo "S3 bucket deleted successfully."
        fi
    else
        echo "Skipping S3 bucket deletion (shared resource)"
    fi

    # Security: Remove downloaded query results
    if [ -f "./query-results.csv" ]; then
        if command -v shred &>/dev/null; then
            shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
```

```
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - Cp
  - CreateNamedQuery
  - DeleteNamedQuery
  - GetNamedQuery
  - GetQueryExecution
  - GetQueryResults
  - ListDatabases
  - ListNamedQueries
  - ListTableMetadata
  - Mb
  - Rb
  - Rm
  - StartQueryExecution

SageMaker Exemplos de IA usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com SageMaker IA.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos de arquivos de atributos de machine learning

O exemplo de código a seguir mostra como:

- Configurar permissões do IAM.
- Crie uma função SageMaker de execução
- Criar grupo de atributos.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
```

```
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"
```

```
        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
```

```
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
```

```
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"
```

```
# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
```

```
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
```

```
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
```

```
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"
```

```
CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
```

```
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
```

```
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS" |
 grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
```

```
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup

- CreateRole
- DeleteBucket
- DeleteFeatureGroup
- DeleteRole
- DescribeFeatureGroup
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListFeatureGroups
- PutPublicAccessBlock
- Rm

Exemplos do Secrets Manager usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Secrets Manager.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criação de uma instância de banco de dados relacional

O exemplo de código a seguir mostra como:

- Configurar componentes de rede.
- Criar um grupo de sub-redes de banco de dados
- Criar uma instância de banco de dados

- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/latest/
UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
```

```
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ]; then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
```

```
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found ${#SUBNET_IDS[@]}
 subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
```

```
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id "$SECRET_NAME"
 --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -d'"'
 -f4)
```

```
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -d'"'
 -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
```

```
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
```

```
    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - Wait

Conceitos básicos de data warehouses de tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Use o gerenciador de segredos CreateSecret
- Use o gerenciador de segredos DeleteSecret
- Use o gerenciador de segredos GetSecretValue
- Use redshift CreateNamespace
- Use redshift CreateWorkgroup
- Use redshift DeleteNamespace
- Use iam CreateRole

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq dependency)
# This script creates a Redshift Serverless environment, loads sample data, and runs
 queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
```

```
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done
```

```
  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
```

```
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status: $status.
 Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5
```

```
  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "============================================"
  echo "CLEANUP CONFIRMATION"
  echo "============================================"
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --workgroup-
name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
```

```
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --namespace-
name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
```

```
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin credentials
 for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"
```

```
# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless get-
namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless get-
workgroup --workgroup-name $WORKGROUP_NAME"
```

```
# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)
```

```
# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
```

```
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
```

```
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
```

```
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID" 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time total
 sales..."
```

```
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price desc)
 as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID" 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
```

```
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources
```

```
echo "Tutorial completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole
  - GetSecretValue
  - GetWorkgroup
  - PutRolePolicy
  - UpdateNamespace

Introdução aos bancos de dados de documentos

O exemplo de código a seguir mostra como:

- Criar um grupo de sub-redes de banco de dados
- Criar um cluster do DocumentDB.
- Criar uma instância do DocumentDB.
- Configurar a segurança e a conectividade.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

################################################################################
# Configuration
################################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

################################################################################
# Logging
################################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""
```

```
##############################################################################
# Region pre-check
##############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

##############################################################################
# Error handler
##############################################################################
handle_error() {
    echo ""
    echo "==========================================="
    echo "ERROR at $1"
    echo "==========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

##############################################################################
# Wait function
##############################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
```

```
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
```

```
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound\|
not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
```

```
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG ingress
 rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed to
 delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
```

```
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}

################################################################################
# Step 1: Generate password and store in Secrets Manager
################################################################################
echo "==========================================="
echo "Step 1: Create master password in Secrets Manager"
echo "==========================================="
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi
```

```
SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "============================================"
echo "Step 2: Find default VPC and subnets"
echo "============================================"
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
```

```
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

################################################################################
# Step 3: Create subnet group
################################################################################
echo "==========================================="
echo "Step 3: Create DocumentDB subnet group"
echo "==========================================="
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi
```

```
CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

##############################################################################
# Step 4: Create DocumentDB cluster
##############################################################################
echo "==========================================="
echo "Step 4: Create DocumentDB cluster"
echo "==========================================="
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""
```

```
##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "============================================="
echo "Step 5: Create DocumentDB instance"
echo "============================================="
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

##############################################################################
# Step 6: Get cluster endpoint and security group
##############################################################################
echo "============================================="
echo "Step 6: Get cluster endpoint and security group"
echo "============================================="
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
```

```
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
##############################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null ||
 true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
```

```
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "============================================"
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "============================================"
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
```

```
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

##############################################################################
# Step 9: Display connection information
##############################################################################
echo "============================================"
echo "CONNECTION INFORMATION"
echo "============================================"
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH} \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString --
output text)"
echo ""

##############################################################################
# Step 10: Cleanup
##############################################################################
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo ""
echo "Resources created:"
```

```
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - RevokeSecurityGroupIngress
  - Wait

Como mover segredos codificados para o armazenamento seguro de segredos

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.

- Criar um segredo no Secrets Manager.
- Atualizar o código de uma aplicação.
- Atualizar o segredo.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
```

```
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name "$RUNTIME_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
```

```
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME" --
query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-name
 "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."
```

```
# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
```

```
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_urls
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
```

```
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/null
 || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
```

```
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').token_

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
```

```
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole

- DeleteSecret
- DetachRolePolicy
- GetCallerIdentity
- GetSecretValue
- PutResourcePolicy
- UpdateSecret

Exemplos do Amazon SES usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon SES.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Configuração de serviços de envio de e-mails

O exemplo de código a seguir mostra como:

- Verificar um endereço de e-mail.
- Verificar um domínio (opcional).
- Verificar limites de envio.
- Enviar um e-mail de teste.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SES Setup Script (v2)
# This script helps you set up Amazon SES for sending emails

# Initialize log file
LOG_FILE="ses-setup.log"
echo "Starting Amazon SES setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
    local ignore_error="${4:-false}"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        if [[ "$ignore_error" != "true" ]]; then
            cleanup_resources
            exit 1
        fi
    fi
}

# Function to clean up resources
cleanup_resources() {
```

```
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"

    # No physical resources to clean up for SES setup
    # Email identities can be deleted if needed
    if [[ -n "$EMAIL_ADDRESS" ]]; then
        echo "Deleting email identity: $EMAIL_ADDRESS" | tee -a "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$EMAIL_ADDRESS\""
    fi

    if [[ -n "$RECIPIENT_EMAIL" && "$RECIPIENT_EMAIL" != "$EMAIL_ADDRESS" ]]; then
        echo "Deleting recipient email identity: $RECIPIENT_EMAIL" | tee -a
 "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$RECIPIENT_EMAIL\""
    fi

    if [[ -n "$DOMAIN_NAME" ]]; then
        echo "Deleting domain identity: $DOMAIN_NAME" | tee -a "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$DOMAIN_NAME\""
    fi
}

# Track created resources
CREATED_RESOURCES=()

# Welcome message
echo "==============================================="
echo "Amazon SES Setup Script"
echo "==============================================="
echo "This script will help you set up Amazon SES for sending emails."
echo "You'll need to verify at least one email address that you own."
echo ""
echo "NOTE: New SES accounts are placed in the sandbox environment."
echo "In the sandbox, both sender AND recipient email addresses must be verified."
echo ""

# Get email address to verify
echo "Please enter an email address that you own and can access:"
EMAIL_ADDRESS="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1)@example.com"
echo "$EMAIL_ADDRESS"

# Verify email identity
echo "Verifying email address: $EMAIL_ADDRESS" | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses verify-email-identity --email-address \"$EMAIL_ADDRESS\"")
```

```
check_error "$OUTPUT" $? "Failed to verify email address"

CREATED_RESOURCES+=("Email identity: $EMAIL_ADDRESS")
echo "A verification email has been sent to $EMAIL_ADDRESS."
echo "Please check your inbox and click the verification link before continuing."
echo ""
echo "Press Enter after you've verified your email address..."
sleep 1

# Check verification status
echo "Checking verification status..." | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses list-identities --identity-type EmailAddress")
check_error "$OUTPUT" $? "Failed to list identities"

OUTPUT=$(log_cmd "aws ses get-identity-verification-attributes --identities
 \"$EMAIL_ADDRESS\"")
check_error "$OUTPUT" $? "Failed to get verification attributes"

# Check if the email is verified
VERIFICATION_STATUS=$(echo "$OUTPUT" | grep -o '"VerificationStatus": "[^"]*' | cut
 -d'"' -f4)
if [[ "$VERIFICATION_STATUS" != "Success" ]]; then
    echo "Email address $EMAIL_ADDRESS is not verified yet. Please check your inbox
 and verify before continuing."
    echo "Exiting script..."
    exit 1
fi

# Ask if user wants to verify a domain
echo ""
echo "Do you want to verify a domain for sending emails? (y/n):"
VERIFY_DOMAIN="y"
echo "$VERIFY_DOMAIN"

if [[ "$VERIFY_DOMAIN" =~ ^[Yy] ]]; then
    echo "Please enter the domain name you want to verify:"
    DOMAIN_NAME="test-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1).example.com"
    echo "$DOMAIN_NAME"

    # Verify domain identity
    echo "Verifying domain: $DOMAIN_NAME" | tee -a "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses verify-domain-identity --domain \"$DOMAIN_NAME\"")
    check_error "$OUTPUT" $? "Failed to verify domain identity"
```

```
    # Extract verification token
    VERIFICATION_TOKEN=$(echo "$OUTPUT" | grep -o '"VerificationToken": "[^"]*' |
cut -d'"' -f4)

    CREATED_RESOURCES+=("Domain identity: $DOMAIN_NAME")

    echo ""
    echo "============================================="
    echo "Domain Verification Instructions"
    echo "============================================="
    echo "To verify your domain ownership, you need to add a TXT record"
    echo "to your domain's DNS settings with the following values:"
    echo ""
    echo "Record Type: TXT"
    echo "Record Name: _amazonses.$DOMAIN_NAME"
    echo "Record Value: $VERIFICATION_TOKEN"
    echo ""
    echo "After adding this DNS record, verification may take up to 72 hours."
    echo ""

    # Set up DKIM for the domain
    echo "Setting up DKIM for domain: $DOMAIN_NAME" | tee -a "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses verify-domain-dkim --domain \"$DOMAIN_NAME\"")
    check_error "$OUTPUT" $? "Failed to set up DKIM"

    # Extract DKIM tokens
    DKIM_TOKENS=$(echo "$OUTPUT" | grep -o '"DkimTokens": \[[^]]*\]' | sed
's/"DkimTokens": \[\|\]//g' | sed 's/,//g' | sed 's/"//g')

    echo "============================================="
    echo "DKIM Configuration Instructions"
    echo "============================================="
    echo "To configure DKIM for your domain, add the following CNAME records"
    echo "to your domain's DNS settings:"
    echo ""

    for token in $DKIM_TOKENS; do
        echo "Record Type: CNAME"
        echo "Record Name: ${token}._domainkey.$DOMAIN_NAME"
        echo "Record Value: ${token}.dkim.amazonses.com"
        echo ""
    done
```

```
    echo "After adding these DNS records, DKIM verification may take up to 72
 hours."
    echo ""
fi

# Check sending limits
echo "Checking your SES sending limits..." | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses get-send-quota")
check_error "$OUTPUT" $? "Failed to get sending quota"

# Extract quota information
MAX_SEND_RATE=$(echo "$OUTPUT" | grep -o '"MaxSendRate": [0-9.]*' | cut -d' ' -f2)
MAX_24_HOUR_SEND=$(echo "$OUTPUT" | grep -o '"Max24HourSend": [0-9.]*' | cut -d' ' -
f2)
SENT_LAST_24_HOURS=$(echo "$OUTPUT" | grep -o '"SentLast24Hours": [0-9.]*' | cut -d'
 ' -f2)

echo ""
echo "==============================================="
echo "Your SES Sending Limits"
echo "==============================================="
echo "Maximum send rate: $MAX_SEND_RATE emails/second"
echo "Maximum 24-hour send: $MAX_24_HOUR_SEND emails"
echo "Sent in the last 24 hours: $SENT_LAST_24_HOURS emails"
echo ""

# Ask if user wants to send a test email
echo "Do you want to send a test email? (y/n):"
SEND_TEST="y"
echo "$SEND_TEST"

if [[ "$SEND_TEST" =~ ^[Yy] ]]; then
    echo ""
    echo "==============================================="
    echo "SANDBOX ENVIRONMENT NOTICE"
    echo "==============================================="
    echo "Your account is likely in the SES sandbox environment."
    echo "In the sandbox, you can only send emails to verified email addresses."
    echo ""
    echo "Do you want to:"
    echo "1. Send a test email to the same verified address ($EMAIL_ADDRESS)"
    echo "2. Verify another email address to use as recipient"
    echo ""
    echo "Enter your choice (1 or 2):"
```

```
    RECIPIENT_CHOICE="1"
    echo "$RECIPIENT_CHOICE"

    if [[ "$RECIPIENT_CHOICE" == "1" ]]; then
        RECIPIENT_EMAIL="$EMAIL_ADDRESS"
    else
        echo "Please enter the recipient email address you want to verify:"
        RECIPIENT_EMAIL="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 |
 head -n 1)@example.com"
        echo "$RECIPIENT_EMAIL"

        # Verify recipient email identity if different from sender
        if [[ "$RECIPIENT_EMAIL" != "$EMAIL_ADDRESS" ]]; then
            echo "Verifying recipient email address: $RECIPIENT_EMAIL" | tee -a
 "$LOG_FILE"
            OUTPUT=$(log_cmd "aws ses verify-email-identity --email-address
 \"$RECIPIENT_EMAIL\"")
            check_error "$OUTPUT" $? "Failed to verify recipient email address"

            CREATED_RESOURCES+=("Email identity: $RECIPIENT_EMAIL")
            echo "A verification email has been sent to $RECIPIENT_EMAIL."
            echo "Please check the inbox and click the verification link before
 continuing."
            echo ""
            echo "Press Enter after you've verified the recipient email address..."
            sleep 1

            # Check recipient verification status
            OUTPUT=$(log_cmd "aws ses get-identity-verification-attributes --
identities \"$RECIPIENT_EMAIL\"")
            check_error "$OUTPUT" $? "Failed to get recipient verification
 attributes"

            # Check if the recipient email is verified
            RECIPIENT_VERIFICATION_STATUS=$(echo "$OUTPUT" | grep -o
 '"VerificationStatus": "[^"]*' | cut -d'"' -f4)
            if [[ "$RECIPIENT_VERIFICATION_STATUS" != "Success" ]]; then
                echo "Recipient email address $RECIPIENT_EMAIL is not verified yet."
                echo "You can try sending the email anyway, but it may fail."
            fi
        fi
    fi
```

```
    echo "Sending test email from $EMAIL_ADDRESS to $RECIPIENT_EMAIL..." | tee -a
 "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses send-email \
        --from \"$EMAIL_ADDRESS\" \
        --destination \"ToAddresses=$RECIPIENT_EMAIL\" \
        --message \"Subject={Data=SES Test
 Email,Charset=UTF-8},Body={Text={Data=This is a test email sent from Amazon SES
 using the AWS CLI,Charset=UTF-8}}\"")

    # Don't exit on send email error, just report it
    check_error "$OUTPUT" $? "Failed to send test email" "true"

    # Check if the email was sent successfully
    if [[ "$OUTPUT" =~ "MessageId" ]]; then
        # Extract message ID
        MESSAGE_ID=$(echo "$OUTPUT" | grep -o '"MessageId": "[^"]*' | cut -d'"' -f4)
        echo "Test email sent successfully! Message ID: $MESSAGE_ID"
    else
        echo "Failed to send test email. This is likely because your account is in
 the sandbox environment."
        echo "In the sandbox, both sender AND recipient email addresses must be
 verified."
    fi
    echo ""
fi

# Summary of created resources
echo ""
echo "==============================================="
echo "Setup Complete - Resources Created"
echo "==============================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo ""

# Ask if user wants to clean up resources
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
echo "Do you want to clean up all created resources? (y/n):"
CLEANUP_CHOICE="y"
echo "$CLEANUP_CHOICE"
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
fi

echo ""
echo "==============================================="
echo "Amazon SES Setup Complete"
echo "==============================================="
echo "For production use, you may need to request to be moved out of the SES
 sandbox."
echo "Visit the SES console and navigate to 'Account dashboard' to request
 production access."
echo ""
echo "Log file: $LOG_FILE"
echo "==============================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - DeleteIdentity
  - GetIdentityVerificationAttributes
  - GetSendQuota
  - ListIdentities
  - SendEmail
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Exemplos do Amazon SNS usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon SNS.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar um tópico e publicar mensagens

O exemplo de código a seguir mostra como:

- Criar um tópico do Amazon SNS.
- Inscrever um endpoint de e-mail no tópico.
- Verificar sua assinatura.
- Publicar uma mensagem no tópico.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SNS Getting Started Script
# This script demonstrates how to create an SNS topic, subscribe to it, publish a
 message,
# and clean up resources.

set -euo pipefail

# Set up logging with secure file permissions
LOG_FILE="sns-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon SNS Getting Started Tutorial..."
echo "$(date)"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    if [ -n "${SUBSCRIPTION_ARN:-}" ] && [ "$SUBSCRIPTION_ARN" != "pending
 confirmation" ] && [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ]; then
        echo "Deleting subscription: $SUBSCRIPTION_ARN"
        if ! aws sns unsubscribe --subscription-arn "$SUBSCRIPTION_ARN" --region
 "$AWS_REGION" 2>/dev/null; then
            echo "Warning: Failed to delete subscription" >&2
        fi
    fi

    if [ -n "${TOPIC_ARN:-}" ]; then
        echo "Deleting topic: $TOPIC_ARN"
        if ! aws sns delete-topic --topic-arn "$TOPIC_ARN" --region "$AWS_REGION"
 2>/dev/null; then
            echo "Warning: Failed to delete topic" >&2
        fi
    fi

    return $exit_code
}

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi
```

```
# Set trap to cleanup on exit
trap cleanup_resources EXIT

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi

if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random topic name suffix using secure method
RANDOM_SUFFIX=$(openssl rand -hex 4)
TOPIC_NAME="my-topic-${RANDOM_SUFFIX}"

# Validate topic name length (max 256 characters)
if [ ${#TOPIC_NAME} -gt 256 ]; then
    handle_error "Topic name exceeds maximum length of 256 characters"
fi

# Step 1: Create an SNS topic
echo "Creating SNS topic: $TOPIC_NAME"
TOPIC_RESULT=$(aws sns create-topic \
    --name "$TOPIC_NAME" \
    --region "$AWS_REGION" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
    --output json) || handle_error "Failed to create SNS topic"

# Extract the topic ARN using jq for reliable parsing
TOPIC_ARN=$(echo "$TOPIC_RESULT" | jq -r '.TopicArn // empty') || handle_error
 "Failed to parse topic result"

if [ -z "$TOPIC_ARN" ]; then
    handle_error "Failed to extract topic ARN from result: $TOPIC_RESULT"
fi

# Validate ARN format
if [[ ! "$TOPIC_ARN" =~ ^arn:aws:sns:[a-z0-9-]+:[0-9]{12}:[a-zA-Z0-9_-]+$ ]]; then
```

```
    handle_error "Invalid SNS topic ARN format: $TOPIC_ARN"
fi

echo "Successfully created topic with ARN: $TOPIC_ARN"

# Step 2: Subscribe to the topic using Email-JSON protocol to reduce costs
echo ""
echo "==============================================="
echo "EMAIL SUBSCRIPTION"
echo "==============================================="
EMAIL_ADDRESS="test-${RANDOM_SUFFIX}@example.com"

# Validate email format (basic validation)
if [[ ! "$EMAIL_ADDRESS" =~ ^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]]; then
    handle_error "Invalid email format: $EMAIL_ADDRESS"
fi

echo "Subscribing email: $EMAIL_ADDRESS to topic using Email-JSON protocol"
SUBSCRIPTION_RESULT=$(aws sns subscribe \
    --topic-arn "$TOPIC_ARN" \
    --protocol email-json \
    --notification-endpoint "$EMAIL_ADDRESS" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to create subscription"

# Extract the subscription ARN using jq
SUBSCRIPTION_ARN=$(echo "$SUBSCRIPTION_RESULT" | jq -r '.SubscriptionArn // empty')
 || handle_error "Failed to parse subscription result"

echo "Subscription created: $SUBSCRIPTION_ARN"
echo "A confirmation email has been sent to $EMAIL_ADDRESS"
echo ""

# Tag the subscription
if [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ] && [ "$SUBSCRIPTION_ARN" !=
 "pending confirmation" ]; then
    aws sns tag-resource \
        --resource-arn "$SUBSCRIPTION_ARN" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
        --region "$AWS_REGION" 2>/dev/null || echo "Warning: Failed to tag
 subscription"
fi
```

```
# Step 3: List subscriptions to verify
echo "Listing subscriptions for topic: $TOPIC_ARN"
SUBSCRIPTIONS=$(aws sns list-subscriptions-by-topic --topic-arn "$TOPIC_ARN" --
region "$AWS_REGION" --output json) || handle_error "Failed to list subscriptions"

echo "Current subscriptions:"
echo "$SUBSCRIPTIONS" | jq '.'

# Get the confirmed subscription ARN with optimized jq query and improved error
 handling
CONFIRMED_SUBSCRIPTION=$(echo "$SUBSCRIPTIONS" | jq -r '.Subscriptions[]? |
 select(.SubscriptionArn != "PendingConfirmation") | .SubscriptionArn' 2>/dev/null |
 head -n 1)

if [ -n "$CONFIRMED_SUBSCRIPTION" ]; then
    SUBSCRIPTION_ARN="$CONFIRMED_SUBSCRIPTION"
else
    echo "Warning: No confirmed subscription found. You may not have confirmed the
 subscription yet."
    echo "The script will continue, but you may not receive the test message."
fi

# Step 4: Publish a message to the topic
echo ""
echo "Publishing a test message to the topic"
MESSAGE="Hello from Amazon SNS! This is a test message sent at $(date)."

# Validate message length (max 256 KB for SNS)
if [ ${#MESSAGE} -gt 262144 ]; then
    handle_error "Message exceeds maximum size of 256 KB"
fi

PUBLISH_RESULT=$(aws sns publish \
    --topic-arn "$TOPIC_ARN" \
    --message "$MESSAGE" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to publish message"

MESSAGE_ID=$(echo "$PUBLISH_RESULT" | jq -r '.MessageId // empty') || handle_error
 "Failed to parse publish result"

if [ -z "$MESSAGE_ID" ]; then
    handle_error "No message ID returned from publish operation"
fi
```

```
# Validate message ID format (UUID v4)
if [[ ! "$MESSAGE_ID" =~ ^[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]
{12}$ ]]; then
    handle_error "Unexpected message ID format: $MESSAGE_ID"
fi

echo "Message published successfully with ID: $MESSAGE_ID"
echo "Check your email for the message."

# Pause to allow the user to check their email
echo ""
echo "Pausing for 3 seconds to allow message delivery..."
sleep 3

# Step 5: Clean up resources
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Resources created:"
echo "- SNS Topic: $TOPIC_ARN"
echo "- Subscription: ${SUBSCRIPTION_ARN:-N/A}"
echo ""
echo "Cleaning up resources to avoid unnecessary charges..."

echo ""
echo "Tutorial completed successfully!"
echo "$(date)"
echo "================================================"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateTopic
  - DeleteTopic
  - ListSubscriptionsByTopic
  - Publicar
  - Assinar
  - Cancelar assinatura

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}
```

```
# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
```

```
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
```

```
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "==========================================="
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"
```

```
# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
```

```
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
```

```
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
```

```
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "==========================================="
```

```
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy

- Rm
- StartConfigurationRecorder
- StopConfigurationRecorder

Conceitos básicos da proteção de dispositivos da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM obrigatórios.
- Habiltiar verificações de auditoria do IoT Device Defender.
- Execute uma On-Demand auditoria
- Criar uma ação de mitigação.
- Aplicar ações de mitigação às descobertas.
- Configurar notificações do SNS (opcional).
- Habilitar o registro em log de IoT

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "====================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "====================================================="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /dev/
null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
```

```
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/
null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
```

```
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
```

```
    ]
}
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
```

```
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
```

```
    echo "Role ARN: $ROLE_ARN"
    return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "====================================================="
echo "Step 1: Creating required IAM roles"
echo "====================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
```

```
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "====================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "====================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1) ||
 true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
```

```
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "==================================================="
echo "Step 3: Running an on-demand audit"
echo "==================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
```

```
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null || echo
 "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
```

```
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "==================================================="
echo "Step 4: Creating a mitigation action"
echo "==================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting it
 first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1
 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)
```

```
if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 ||
 true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "====================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "====================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
```

```
  "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1) ||
 true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
```

```
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --output
 json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
```

```
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "==================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "==================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)
```

```
    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2 methods,
 continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved but
 could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
```

```
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true
```

```
# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --output
 json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null ||
 echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
```

```
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies

- ListTopics
- PutRolePolicy
- SetLoggingOptions
- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Exemplos de funções de etapa usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com Step Functions.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos da orquestração de fluxo de trabalaho

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para o Step Functions.
- Criar sua primeira máquina de estado.
- Iniciar a execução da máquina de estado.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS Step
 Functions Getting Started tutorial

set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
done
```

```
# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT

echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be less
 reliable."
    echo "Consider installing jq for better error handling: brew install jq (macOS)
 or apt-get install jq (Ubuntu)"
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
export AWS_REGION="$CURRENT_REGION"
```

```
# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
```

```
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}

# Function to extract JSON field efficiently
extract_json_field() {
    local json="$1"
    local field="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] || [ "$wait_time"
 -gt 300 ]; then
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"

    if [[ "$USE_JQ" == "true" ]]; then
```

```
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."

    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN" 2>/dev/null &
```

```
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
$ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for detach operations to complete
    wait 2>/dev/null || true

    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed to
delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
        "input.json"
        "sentiment-input.json"
```

```
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
            fi
        fi
    done
}

# Security: Set trap to cleanup on script exit
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
```

```
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
```

```
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)

check_api_error "$ROLE_RESULT" "Create IAM role"
```

```
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"

# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")
```

```
if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)

if [ $? -ne 0 ]; then
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi

# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)
```

```
check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"

# Wait for execution to complete (the workflow has a 10-second wait state)
echo "Waiting for execution to complete (12 seconds)..."
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
```

```
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
```

```
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)
```

```
check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"

# Wait for execution to complete (the workflow has a 5-second wait state)
echo "Waiting for execution to complete (8 seconds)..."
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
```

```
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")

    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
    fi
    echo "Policy ARN: $POLICY_ARN"

    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
```

```
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
```

```
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
```

```
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"

    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)

    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3

    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
```

```
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
fi
echo "============================================"

# Cleanup
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy
  - StartExecution
  - UpdateStateMachine

AWS STS exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
```

```
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
##############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
```

```
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Para obter detalhes da API, consulte AssumeRole em Referência de AWS CLI comandos.

Cenários

Configurar a conectividade do serviço de contêiner

O exemplo de código a seguir mostra como:

- Criar a infraestrutura da VPC.
- Configurar registro em log
- Criar o cluster do ECS.
- Configurar perfis do IAM.
- Criar o serviço com o Service Connect.
- Verificar a implantação
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
```

```
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
```

```
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
```

```
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true
```

```
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
```

```
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
```

```
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
```

```
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
```

```
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
```

```
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
```

```
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
```

```
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
```

```
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"
```

```
    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
```

```
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy

  - GetCallerIdentity

- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Criar uma tarefa de contêiner para o tipo de inicialização com tecnologia sem servidor

O exemplo de código a seguir mostra como:

- Criar um cluster.
- Criar uma definição de tarefa.
- Criar o serviço.
- Fazer a limpeza.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error
```

```
# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "==========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "==========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
```

```
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|AccessDenied
\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
```

```
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security group
 $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
```

```
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."
```

```
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]]; then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
 anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]]; then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete security
 group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."
```

```
            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix "$TASK_FAMILY"
 --query "taskDefinitionArns" --output text 2>/dev/null || echo "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "============================================"
        echo "CLEANUP COMPLETED"
        echo "============================================"
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
```

```
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME --
service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo "6. Deregister task definitions: aws ecs list-task-definitions --
family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --task-
definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
```

```
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --assume-
role-policy-document file://trust-policy.json" "Create ECS task execution role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
 "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"
```

```
CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-ecs-
fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "logConfiguration": {
```

```
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-json
 file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
```

```
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --group-
name '$SECURITY_GROUP_NAME' --description 'Security group for ECS Fargate tutorial
 - HTTP access' --vpc-id '$VPC_ID' --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-fargate-
linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"' |
 head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP inbound
 rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
```

```
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-type
 FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets\":[\"$(echo
 "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups\":[\"$SECURITY_GROUP_ID
\"],\"assignPublicIp\":\"ENABLED\"}}' --tags key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"
```

```
echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN" --
query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --output
 text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID" --
query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
```

```
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole

- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeNetworkInterfaces
- DescribeSecurityGroups
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Criar uma API REST com integração de proxy de função

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para execução do Lambda.
- Criar e implantar uma função do Lambda.
- Criar uma API REST
- Configurar a integração de proxy do Lambda.
- Implantar e testar a API
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
```

```
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
```

```
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
```

```
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
```

```
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query "items[?
pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
```

```
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
```

```
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed to
 delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning: Failed
 to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json
```

```
echo "Cleanup completed!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
```

```
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
```

```
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
"Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
```

```
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
```

```
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
```

```
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
```

```
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."
```

```
    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
```

```
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
```

```
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
```

```
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}
```

```
# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
```

```
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
```

```
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "==========================================="
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair

- CreateRole
- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteKeyPair
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Como criar um espaço de trabalho de monitoramento gerenciado

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM para seu espaço de trabalho.

- Criar um espaço de trabalho do Grafana.
- Configurar autenticação
- Definir configurações opcionais.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
```

```
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
```

```
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-grafana-
gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"
```

```
echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
```

```
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -d'"' -
f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"
```

```
    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 | grep
 -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
```

```
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Como criar um painel de monitoramento com o nome da função como uma variável

O exemplo de código a seguir mostra como:

- Crie um CloudWatch painel
- Adicionar widgets de métricas do Lambda com uma variável de nome de função.
- Verificar o painel.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}
```

```
# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with 'aws
 configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
```

```
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-dynamicdash
 \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
```

```
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/null
 --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"
```

```
# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}", { "stat":
 "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
```

```
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi
```

```
# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
```

```
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of the
 dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================="
```

```
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --region
 "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to delete
 it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach it
 manually."
        fi
```

```
        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Conceitos básicos do gerenciamento de configurações

O exemplo de código a seguir mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR
```

```
# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi
```

```
        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "==========================================="
    echo "CREATED RESOURCES"
    echo "==========================================="
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
```

```
    fi
    echo "============================================="
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
```

```
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
```

```
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
```

```
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"
```

```
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm

- StartConfigurationRecorder
- StopConfigurationRecorder

Conceitos básicos de registros de contêineres

O exemplo de código a seguir mostra como:

- Criar uma imagem do Docker.
- Criar um repositório do Amazon ECR.
- Excluir recursos.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "====================================================="
```

```
# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
 for installation instructions."
    exit 1
fi

# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure' to
 set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running this
 script."
    echo "Visit https://docs.docker.com/get-docker/ for installation instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"
```

```
    echo "====================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "====================================================="

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "====================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force || echo
 "Failed to delete repository, it may not exist or may have already been deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local image,
 it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove tagged
 image, it may not exist or may have already been deleted."
    fi
```

```
    echo "Cleanup completed."
    echo "==================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build Docker
 image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list Docker
 images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"
```

```
# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials are
 configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"

# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --tags
 key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-registry-
gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username AWS --
password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" || handle_error
 "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"
```

```
# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image to
 ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "====================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "====================================================="

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
```

```
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "==================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- BatchDeleteImage
- CreateRepository
- DeleteRepository
- GetCallerIdentity
- GetLoginPassword

Conceitos básicos de mensagens da Internet das Coisas

O exemplo de código a seguir mostra como:

- Criar recursos de IoT
- Configurar o dispositivo
- Executar os aplicativos de exemplo
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
```

```
}

# Function to check for errors
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a $LOG_FILE
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --
new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
```

```
}

# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "==================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a $LOG_FILE
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
    echo "AWS region not configured. Please run 'aws configure' to set your region."
 | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE

# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE
```

```
# Create IoT policy
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-document file://
iot-policy.json --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-core-
gs"
check_error $?
```

```
# Create IoT thing
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn' --
output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?

# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active --
certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile $CERTS_DIR/
public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target $CERTIFICATE_ARN"
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
```

```
check_error $?

# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." | tee
 -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-python-
v2.git"
    check_error $?
else
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
```

```
fi

# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
```

```
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE

# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
```

```
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE

    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"

    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"

    # Delete policies
```

```
    echo "Deleting policies..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-status
 INACTIVE" | tee -a $LOG_FILE
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachPolicy
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreatePolicy
  - CreateThing
  - DeleteCertificate
  - DeletePolicy
  - DeleteThing
  - DescribeEndpoint

- DetachPolicy
- DetachThingPrincipal
- GetCallerIdentity
- UpdateCertificate

Conceitos básicos de arquivos de atributos de machine learning

O exemplo de código a seguir mostra como:

- Configurar permissões do IAM.
- Crie uma função SageMaker de execução
- Criar grupo de atributos.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()
```

```
# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
```

```
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
```

```
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)
```

```
    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
```

```
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
```

```
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
```

```
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
```

```
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3:////'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
```

```
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi
```

```
echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
```

```
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS" |
 grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
```

```
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AttachRolePolicy
- CreateBucket
- CreateFeatureGroup
- CreateRole
- DeleteBucket
- DeleteFeatureGroup
- DeleteRole
- DescribeFeatureGroup
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListFeatureGroups
- PutPublicAccessBlock
- Rm

Conceitos básicos de clusters gerenciados de kubernetes

O exemplo de código a seguir mostra como:

- Criar uma VPC para um cluster do EKS.
- Criar perfis do IAM para um cluster do EKS.
- Criação do seu cluster de EKS
- Configurar o kubectl para se comunicar com o cluster.
- Crie um grupo de nós gerenciados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the AWS
 CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your operating
 system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
```

```
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/
kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"
```

```
# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --nodegroup-
name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
```

```
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources; exit
 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
```

```
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
```

```
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
```

```
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
```

```
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-kubeconfig
 --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"
```

```
NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn" --
output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl get
 nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or the
 nodes are still registering."
```

```
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl get
 all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
```

```
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --cluster-
name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --stack-
name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole

- DeleteStack
- DescribeCluster
- DescribeNodegroup
- DescribeStacks
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

Conceitos básicos de streaming gerenciado

O exemplo de código a seguir mostra como:

- Criar um cluster do Amazon MSK.
- Criar permissões do IAM para acessar o MSK.
- Criar uma máquina cliente
- Obter os agentes de bootstrap.
- Configurar a máquina cliente.
- Crie um tópico e produce/consume dados
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
```

```
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group: $CLIENT_SG_ID";
 fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
```

```
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id" &>/
dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated" &>/
dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
```

```
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o "\"GroupId
\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing $sg_id,
 removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing $sg_id,
 removing them..."
```

```
                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No outbound
 all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed to
 terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group" "$CLIENT_SG_ID";
 then
        remove_security_group_references "$CLIENT_SG_ID"
```

```
        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete key
pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
```

```
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to delete
 policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
```

```
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az (subnet:
 $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
```

```
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default VPC
 in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-az,Values=true"
 \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least 3
 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=$DEFAULT_VPC_ID"
 \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"
```

```
# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\", \"ClientSubnets
\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\", \"${SUBNET_ARRAY[2]}\"],
 \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-gs
 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --query
 "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
```

```
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "================================================"
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
```

```
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
```

```
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "================================================"

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@]; then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi
```

```
echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)
```

```
if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
```

```
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
```

```
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-${RANDOM_SUFFIX}},
{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-streaming-for-
apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response: $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)
```

```
    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
```

```
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi
```

```
echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-
auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/$LATEST_VERSION/
aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
```

```
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "================================================"

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
```

```
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "================================================"

# Display summary of created resources
echo ""
echo "================================================"
echo "RESOURCE SUMMARY"
echo "================================================"
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "================================================"
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
```

```
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
```

```
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile

- CreateKeyPair
- CreatePolicy
- CreateRole
- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances

- TerminateInstances

- Wait

Conceitos básicos das notificações push

O exemplo de código a seguir mostra como:

- Criar uma aplicação.
- Habilitar canais de notificações por push.
- Enviar notificações por push
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail

# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
```

```
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
cleanup() {
    local exit_code=$?
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
```

```
        echo "ERROR: Command failed: $cmd" >&2
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
}

# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi

    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
```

```
    echo ""

    return 0
}

# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
        echo "jq is not available, using grep for JSON parsing"
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."

    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
```

```
        echo "  - mobiletargeting:CreateApp" >&2
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
    local app_name=$1

    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters" >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi
```

```
check_json_tools
validate_permissions

# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
fi

echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi

echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "==========================================="
echo "ENABLING FCM (GCM) CHANNEL"
```

```
echo "============================================"
echo "Note: This is using a placeholder API key for demonstration purposes only."
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-demo-
only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real certificates)
echo ""
echo "============================================"
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================"
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --apns-
channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
```

```
    --application-id "$APP_ID" \
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-channel
 command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "============================================"
echo "CREATING MESSAGE FILES"
echo "============================================"

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
```

```
        "key1": "value1",
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF

echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
echo "To send real messages, you would need to replace the sample device tokens with
 actual ones."

# Step 5: Demonstrate how to send messages (this will fail with placeholder tokens)
```

```
echo ""
echo "============================================"
echo "DEMONSTRATING MESSAGE SENDING"
echo "============================================"
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
echo "Retrieving application details..."
```

```
echo "Executing: aws pinpoint get-app --application-id $APP_ID"
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
echo "2. Attempting to enable FCM and APNS channels (with placeholder credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
```

```
echo "5. Retrieving application details"
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps
  - GetCallerIdentity
  - SendMessages
  - UpdateApnsChannel
  - UpdateGcmChannel

Conceitos básicos dos mecanismos de pesquisa e analytics

O exemplo de código a seguir mostra como:

- Criar um domínio OpenSearch de serviço

- Carregar dados em seu domínio.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md

# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
```

```
# Track if domain was successfully created
DOMAIN_CREATED=false
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [[ "$DOMAIN_CREATED" == "true" ]]; then
        echo "Checking if domain $DOMAIN_NAME exists before attempting to delete..."

        # Check if domain exists before trying to delete
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/null
 2>&1; then
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
```

```
}

# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
```

```
# This policy restricts access to specific actions and should be further restricted
 to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack creation
 failed"

DOMAIN_CREATED=true
echo "Domain created successfully via CloudFormation."

# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120

# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo "No";
 fi)"
```

```
# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel", "Malahide,
 Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas", "Stellate, Nick",
 "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese", "Spielbauer, Bruce",
 "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett, Charlie"], "title": "U.S.
 Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955, "actor":
 ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal", "Backus, Jim",
 "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen, Corey", "Birch, Paul",
 "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh, Nelson", "Williams,
```

```
 Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira", "McMahon, David",
 "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"

    # Test 1: Basic authentication with root endpoint
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)

    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }
```

```
    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d:
-f2)

        if is_success_code "$HEALTH_HTTP_CODE"; then
            echo "# Cluster health authentication test successful! (HTTP
$HEALTH_HTTP_CODE)"
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
            echo "Cluster health authentication also failed with HTTP code:
$HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
then
                echo "Error: Request is being treated as anonymous (authentication
not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential backoff..."

            # Retry authentication test with exponential backoff
```

```
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT+1))/
$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)

                echo "Retry result (HTTP $HTTP_CODE):"
                echo "$AUTH_TEST_RESULT"

                if is_success_code "$HTTP_CODE"; then
                    echo "# Authentication test successful after retry! (HTTP
 $HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" |
 cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
 (HTTP $HEALTH_HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
```

```
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding with
data operations."

        # Upload single document (matches tutorial exactly)
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
'"result"'; then
            echo "# Single document uploaded successfully! (HTTP $UPLOAD_HTTP_CODE)"
        else
            echo "# Warning: Single document upload may have failed (HTTP
$UPLOAD_HTTP_CODE)"
        fi

        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
```

```
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
'"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
$BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
        echo "Waiting for documents to be indexed..."
        sleep 5

        # Step 3: Search Documents (matches tutorial exactly)
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
```

```
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
 '"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "==========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "==========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."

        # Debug domain configuration
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-name "$DOMAIN_NAME"
 --query 'DomainStatus.{AdvancedSecurityOptions: AdvancedSecurityOptions,
 AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
```

```
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi

# Display OpenSearch Dashboards URL (matches tutorial)
echo ""
echo "============================================"
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================"
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
```

```
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
echo "Cleaning up temporary files..."

# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true
```

```
echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - GetCallerIdentity

Conceitos básicos de firewalls do aplicativo Web

O exemplo de código a seguir mostra como:

- Criar uma ACL da web.
- Adicionar uma regra de correspondência de string.
- Adicionar regras gerenciadas.
- Configurar o registro em log da
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
```

```
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi
```

```
    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/
dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the AWS
 Console."
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
```

```
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"
```

```
# Step 2: Add a String Match Rule
echo ""
echo "====================================================="
echo "STEP 2: Adding String Match Rule"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
 || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
```

```
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
"SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
\
        --region us-east-1 2>&1) || true
```

```
    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi
```

```
    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
|| true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
```

```
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
        --visibility-config
"SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
\
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
attempts"
```

```
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "====================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "====================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "====================================================="
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "====================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo "$first_dist" |
 grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
```

```
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
```

```
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

Como mover segredos codificados para o armazenamento seguro de segredos

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.
- Criar um segredo no Secrets Manager.
- Atualizar o código de uma aplicação.
- Atualizar o segredo.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
```

```
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name "$RUNTIME_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME" --
query 'PolicyNames[]' --output text 2>/dev/null || true); do
```

```
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-name
 "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
```

```
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_urls
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
```

```
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/null
 || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').token_

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
```

```
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
```

```
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy
  - GetCallerIdentity
  - GetSecretValue
  - PutResourcePolicy
  - UpdateSecret

Executar testes de estresse da CPU em instâncias da máquina virtual usando a injeção de falhas

O exemplo de código a seguir mostra como:

- Criar perfis do IAM.

- Crie um CloudWatch alarme
- Criar um modelo de experimento
- Executar o experimento.
- Verificar os resultados.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
```

```
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi
```

```
    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
```

```
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
```

```
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
```

```
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60
```

```
# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
```

```
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF
```

```
# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail
```

```
    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")
```

```
# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
```

```
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
```

```
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .

- AddRoleToInstanceProfile
- AttachRolePolicy
- CreateExperimentTemplate
- CreateInstanceProfile
- CreateRole
- DeleteAlarms
- DeleteExperimentTemplate
- DeleteInstanceProfile
- DeleteRole
- DeleteRolePolicy
- DescribeAlarms
- DescribeImages
- DetachRolePolicy
- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Como configurar o gerenciamento de sistemas

O exemplo de código a seguir mostra como:

- Criar permissões do IAM para o Systems Manager.

- Criar um perfil do IAM para o Systems Manager
- Configurar o Systems Manager.
- Verificar a configuração.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
```

```
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
```

```
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id --
policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --manager-
arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null
 || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
```

```
echo "This script will set up AWS Systems Manager for a single account and region."
echo "It will create IAM policies and roles, then enable Systems Manager features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
```

```
          "ssm:GetPatchBaseline",
          "ssm:DescribeMaintenanceWindows",
          "ssm:DescribeMaintenanceWindowTasks"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
```

```
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
```

```
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
```

```
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
```

```
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/accountdiscovery.ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
```

```
      }
    ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"
```

```
# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
```

```
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --policy-
arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
```

```
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-arn
 $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
```

```
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up later." |
 tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole

- DetachRolePolicy
- GetCallerIdentity
- GetConfigurationManager

Suporte exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com Suporte.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução ao suporte técnico

O exemplo de código a seguir mostra como:

- Verificar os serviços disponíveis e os níveis de gravidade.
- Criar um caso de suporte.
- Adicionar comunicações a um caso.

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash
```

```
# AWS Support CLI Tutorial Script
# This script demonstrates how to use AWS Support API through AWS CLI

set -euo pipefail

# Security: Validate script location and permissions
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
readonly SCRIPT_DIR

# Security: Use secure temporary directory
readonly TEMP_DIR="$(mktemp -d)" || { echo "Failed to create temp directory"; exit
 1; }
trap "rm -rf '$TEMP_DIR'" EXIT

# Set up logging with secure permissions
LOG_FILE="${TEMP_DIR}/aws-support-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Validate AWS CLI is available
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed or not in PATH" >&2
    exit 1
fi

# Security: Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not properly configured" >&2
    exit 1
fi

{
    echo "Starting AWS Support Tutorial at $(date)"
    echo "Script: $0"
    echo "User: $(whoami)"
    echo "---"
} >> "$LOG_FILE"

# Function to log commands and their outputs securely
log_cmd() {
    local cmd="$1"
    echo "$(date): Running command" >> "$LOG_FILE"
    # Don't echo the actual command to prevent credential leakage
```

```
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
    local is_fatal="${4:-true}"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command returned status: $cmd_status" >> "$LOG_FILE"

        # Check for subscription error
        if [[ "$cmd_output" =~ "SubscriptionRequiredException" ]]; then
            {
                echo ""
                echo "====================================================="
                echo "IMPORTANT: This account does not have the required AWS Support
 plan."
                echo "You need a Business, Enterprise On-Ramp, or Enterprise Support
 plan"
                echo "to use the AWS Support API."
                echo ""
                echo "This script will now demonstrate the commands that would be
 run"
                echo "if you had the appropriate support plan, but will not execute
 them."
                echo "====================================================="
            } | tee -a "$LOG_FILE"

            DEMO_MODE=true
            return 0
        fi

        if [[ "$is_fatal" == "true" ]]; then
            cleanup_resources
            exit 1
        fi
    fi
}
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    echo "No persistent resources were created that need cleanup." | tee -a
 "$LOG_FILE"
}

# Function to run a command in demo mode
demo_cmd() {
    local cmd="$1"
    local description="$2"

    {
        echo ""
        echo "DEMO: $description"
        echo "Command that would be executed:"
        echo "  [Command hidden for security]"
        echo ""
    } | tee -a "$LOG_FILE"
}

# Function to safely extract JSON values
extract_json_value() {
    local json_output="$1"
    local key="$2"

    echo "$json_output" | grep -o "\"$key\": \"[^\"]*\"" | head -1 | cut -d'"' -f4
 || echo ""
}

# Array to track created resources
declare -a CREATED_RESOURCES

# Initialize demo mode flag
DEMO_MODE=false

# Security: Validate input parameters
if [[ $# -gt 0 ]]; then
    echo "ERROR: This script does not accept parameters" >&2
    exit 1
fi

{
    echo "====================================================="
```

```
    echo "AWS Support CLI Tutorial"
    echo "====================================================="
    echo "This script demonstrates how to use AWS Support API"
    echo "Note: You must have a Business, Enterprise On-Ramp,"
    echo "or Enterprise Support plan to use the AWS Support API."
    echo "====================================================="
    echo ""
} | tee -a "$LOG_FILE"

# Step 1: Check available services
echo "Step 1: Checking available AWS Support services..." | tee -a "$LOG_FILE"
SERVICES_OUTPUT=$(log_cmd "aws support describe-services --language en" 2>&1) ||
 SERVICES_OUTPUT=""
check_error "$SERVICES_OUTPUT" $? "Failed to retrieve AWS Support services"

# If we're in demo mode, set default values
if [[ "$DEMO_MODE" == "true" ]]; then
    SERVICE_CODE="general-info"
    echo "Using demo service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
else
    # Extract a service code for demonstration using safer method
    SERVICE_CODE=$(extract_json_value "$SERVICES_OUTPUT" "code") || SERVICE_CODE=""
    if [[ -z "$SERVICE_CODE" ]]; then
        SERVICE_CODE="general-info"
        echo "Using default service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
    else
        echo "Found service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
    fi
fi

# Step 2: Check available severity levels
echo "Step 2: Checking available severity levels..." | tee -a "$LOG_FILE"
if [[ "$DEMO_MODE" == "true" ]]; then
    demo_cmd "aws support describe-severity-levels --language en" "Check available
 severity levels"
    SEVERITY_CODE="low"
    echo "Using demo severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
else
    SEVERITY_OUTPUT=$(log_cmd "aws support describe-severity-levels --language en"
 2>&1) || SEVERITY_OUTPUT=""
    check_error "$SEVERITY_OUTPUT" $? "Failed to retrieve severity levels"

    SEVERITY_CODE=$(extract_json_value "$SEVERITY_OUTPUT" "code") ||
 SEVERITY_CODE=""
```

```
    if [[ -z "$SEVERITY_CODE" ]]; then
        SEVERITY_CODE="low"
        echo "Using default severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
    else
        echo "Found severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
    fi
fi

# Step 3: Create a test support case
{
    echo ""
    echo "====================================================="
    echo "SUPPORT CASE CREATION"
    echo "====================================================="
} | tee -a "$LOG_FILE"

if [[ "$DEMO_MODE" == "true" ]]; then
    {
        echo "DEMO MODE: The following steps would create and manage a support case"
        echo "if you had a Business, Enterprise On-Ramp, or Enterprise Support
 plan."
        echo ""
    } | tee -a "$LOG_FILE"

    USER_EMAIL="example@example.com"

    demo_cmd "aws support create-case" "Create a support case"

    CASE_ID="case-12345678910-2013-c4c1d2bf33c5cf47"
    echo "Demo case ID: $CASE_ID" | tee -a "$LOG_FILE"

    demo_cmd "aws support describe-cases" "List support cases"
    demo_cmd "aws support add-communication-to-case" "Add communication to case"
    demo_cmd "aws support describe-communications" "View case communications"
    demo_cmd "aws support resolve-case" "Resolve the support case"

else
    echo "Creating a test support case..." | tee -a "$LOG_FILE"

    USER_EMAIL="example@example.com"
    CC_EMAIL_PARAM="--cc-email-addresses $USER_EMAIL"

    # Create the case
```

```
    CASE_OUTPUT=$(log_cmd "aws support create-case --subject 'AWS CLI Tutorial Test
 Case' --service-code '$SERVICE_CODE' --category-code 'using-aws' --communication-
body 'This is a test case created as part of an AWS CLI tutorial.' --severity-code
 '$SEVERITY_CODE' --language 'en' $CC_EMAIL_PARAM" 2>&1) || CASE_OUTPUT=""

    check_error "$CASE_OUTPUT" $? "Failed to create support case"

    # Extract the case ID safely
    CASE_ID=$(extract_json_value "$CASE_OUTPUT" "caseId") || CASE_ID=""

    if [[ -n "$CASE_ID" ]]; then
        echo "Successfully created support case with ID: $CASE_ID" | tee -a
 "$LOG_FILE"
        CREATED_RESOURCES+=("Support Case: $CASE_ID")

        # Step 4: List the case we just created
        echo "" | tee -a "$LOG_FILE"
        echo "Step 4: Listing the support case we just created..." | tee -a
 "$LOG_FILE"
        CASES_OUTPUT=$(log_cmd "aws support describe-cases --case-id-list '$CASE_ID'
 --include-resolved-cases false --language 'en'" 2>&1) || CASES_OUTPUT=""

        check_error "$CASES_OUTPUT" $? "Failed to retrieve case details"

        # Step 5: Add a communication to the case
        echo "" | tee -a "$LOG_FILE"
        echo "Step 5: Adding a communication to the support case..." | tee -a
 "$LOG_FILE"
        COMM_OUTPUT=$(log_cmd "aws support add-communication-to-case --case-id
 '$CASE_ID' --communication-body 'This is an additional communication for the test
 case.' $CC_EMAIL_PARAM" 2>&1) || COMM_OUTPUT=""

        check_error "$COMM_OUTPUT" $? "Failed to add communication to case"

        # Step 6: View communications for the case
        echo "" | tee -a "$LOG_FILE"
        echo "Step 6: Viewing communications for the support case..." | tee -a
 "$LOG_FILE"
        COMMS_OUTPUT=$(log_cmd "aws support describe-communications --case-id
 '$CASE_ID' --language 'en'" 2>&1) || COMMS_OUTPUT=""

        check_error "$COMMS_OUTPUT" $? "Failed to retrieve case communications"

        # Step 7: Resolve the case
```

```
        {
            echo ""
            echo "====================================================="
            echo "CASE RESOLUTION"
            echo "====================================================="
            echo "Resolving the support case..."
        } | tee -a "$LOG_FILE"

        RESOLVE_OUTPUT=$(log_cmd "aws support resolve-case --case-id '$CASE_ID'"
 2>&1) || RESOLVE_OUTPUT=""

        check_error "$RESOLVE_OUTPUT" $? "Failed to resolve case"
        echo "Successfully resolved support case: $CASE_ID" | tee -a "$LOG_FILE"
    else
        echo "Could not extract case ID from the response." | tee -a "$LOG_FILE"
    fi
fi

# Display summary of created resources
{
    echo ""
    echo "====================================================="
    echo "TUTORIAL SUMMARY"
    echo "====================================================="
} | tee -a "$LOG_FILE"

if [[ "$DEMO_MODE" == "true" ]]; then
    {
        echo "This was a demonstration in DEMO MODE."
        echo "No actual AWS Support cases were created."
        echo "To use the AWS Support API, you need a Business, Enterprise On-Ramp,"
        echo "or Enterprise Support plan."
    } | tee -a "$LOG_FILE"
else
    {
        echo "Resources created during this tutorial:"
        if [[ ${#CREATED_RESOURCES[@]} -eq 0 ]]; then
            echo "No resources were created."
        else
            for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
                echo "- $resource"
            done
        fi
    } | tee -a "$LOG_FILE"
```

```
fi

{
    echo ""
    echo "Tutorial completed successfully!"
    echo "Log file: $LOG_FILE"
    echo "====================================================="
} | tee -a "$LOG_FILE"

# Display log file path to user
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddCommunicationToCase
  - CreateCase
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Exemplos do Systems Manager usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Systems Manager.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Como criar um serviço de contêiner para instâncias de máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um cluster do ECS.
- Criar e monitorar um serviço.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"
```

```
# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
```

```
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
"$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
"$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
"Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi
```

```
    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
```

```
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi
```

```
    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}
```

```
# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
```

```
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}
```

```
# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
```

```
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
```

```
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
```

```
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"
```

```
    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
```

```
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
```

```
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
```

```
main "$@"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetParameters
  - GetRole
  - ListContainerInstances
  - ListServices
  - ListTasks
  - RegisterTaskDefinition
  - RunInstances

- StopTask
- TerminateInstances
- UpdateService
- Wait

Introdução às máquinas virtuais

O exemplo de código a seguir mostra como:

- Criar um par de chaves.
- Criar um grupo de segurança.
- Interromper e iniciar uma instância.
- Testar a persistência de um IP elástico.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# EC2 Basics Tutorial Script - Revised
# This script demonstrates the basics of working with EC2 instances using AWS CLI
# Updated to use Amazon Linux 2023 and enhanced security settings

# Set up logging
LOG_FILE="ec2_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to log messages
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}
```

```
# Function to handle errors
handle_error() {
  log "ERROR: $1"
  log "Cleaning up resources..."
  cleanup
  exit 1
}

# Function to clean up resources
cleanup() {
  log "Resources created:"

  if [ -n "$ASSOCIATION_ID" ]; then
    log "- Elastic IP Association: $ASSOCIATION_ID"
  fi

  if [ -n "$ALLOCATION_ID" ]; then
    log "- Elastic IP Allocation: $ALLOCATION_ID (IP: $ELASTIC_IP)"
  fi

  if [ -n "$INSTANCE_ID" ]; then
    log "- EC2 Instance: $INSTANCE_ID"
  fi

  if [ -n "$SECURITY_GROUP_ID" ]; then
    log "- Security Group: $SECURITY_GROUP_ID"
  fi

  if [ -n "$KEY_NAME" ]; then
    log "- Key Pair: $KEY_NAME (File: $KEY_FILE)"
  fi

  REPLY=y

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    log "Starting cleanup..."

    # Track cleanup failures
    CLEANUP_FAILURES=0

    # Disassociate Elastic IP if it exists
    if [ -n "$ASSOCIATION_ID" ]; then
      log "Disassociating Elastic IP..."
      if ! aws ec2 disassociate-address --association-id "$ASSOCIATION_ID"; then
```

```
        log "Failed to disassociate Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Release Elastic IP if it exists
    if [ -n "$ALLOCATION_ID" ]; then
      log "Releasing Elastic IP..."
      if ! aws ec2 release-address --allocation-id "$ALLOCATION_ID"; then
        log "Failed to release Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Terminate instance if it exists
    if [ -n "$INSTANCE_ID" ]; then
      log "Terminating instance $INSTANCE_ID..."
      if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" > /dev/null;
then
        log "Failed to terminate instance"
        ((CLEANUP_FAILURES++))
      else
        log "Waiting for instance to terminate..."
        if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"; then
          log "Failed while waiting for instance to terminate"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Delete security group if it exists
    if [ -n "$SECURITY_GROUP_ID" ]; then
      log "Deleting security group..."
      if ! aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"; then
        log "Failed to delete security group"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ]; then
      log "Deleting key pair..."
      if ! aws ec2 delete-key-pair --key-name "$KEY_NAME"; then
        log "Failed to delete key pair"
```

```
        ((CLEANUP_FAILURES++))
      fi

      # Remove key file
      if [ -f "$KEY_FILE" ]; then
        log "Removing key file..."
        if ! rm -f "$KEY_FILE"; then
          log "Failed to remove key file"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Report cleanup status
    if [ $CLEANUP_FAILURES -eq 0 ]; then
      log "Cleanup completed successfully."
    else
      log "WARNING: Cleanup completed with $CLEANUP_FAILURES failures. Some
 resources may not have been deleted properly."
    fi
  else
    log "Resources were not deleted."
  fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
KEY_NAME="ec2-tutorial-key-$RANDOM_ID"
SG_NAME="ec2-tutorial-sg-$RANDOM_ID"

# Create a directory for the key file
KEY_DIR=$(mktemp -d)
KEY_FILE="$KEY_DIR/$KEY_NAME.pem"

log "Starting EC2 basics tutorial script"
log "Random identifier: $RANDOM_ID"
log "Key name: $KEY_NAME"
log "Security group name: $SG_NAME"

# Step 1: Create a key pair
log "Creating key pair..."
KEY_RESULT=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial' --
output text)
```

```
if [ $? -ne 0 ] || [ -z "$KEY_RESULT" ]; then
  handle_error "Failed to create key pair"
fi

echo "$KEY_RESULT" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
log "Created key pair and saved to $KEY_FILE"

# Step 2: Create a security group
log "Creating security group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --group-name "$SG_NAME" \
  --description "Security group for EC2 tutorial" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query "GroupId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$SECURITY_GROUP_ID" ]; then
  handle_error "Failed to create security group"
fi

log "Created security group: $SECURITY_GROUP_ID"

# Get current public IP address for SSH access
MY_IP=$(curl -s http://checkip.amazonaws.com)
if [ $? -ne 0 ] || [ -z "$MY_IP" ]; then
  handle_error "Failed to get current IP address"
fi

log "Adding SSH ingress rule for IP $MY_IP..."
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr "$MY_IP/32" > /dev/null

if [ $? -ne 0 ]; then
  handle_error "Failed to add security group ingress rule"
fi

log "Added SSH ingress rule for IP $MY_IP"

# Step 3: Find an Amazon Linux 2023 AMI (updated from AL2)
```

```
log "Finding latest Amazon Linux 2023 AMI..."
AMI_ID=$(aws ssm get-parameters-by-path \
  --path "/aws/service/ami-amazon-linux-latest" \
  --query "Parameters[?contains(Name, 'al2023-ami-kernel-default-x86_64')].Value" \
  --output text | head -1)

if [ $? -ne 0 ] || [ -z "$AMI_ID" ]; then
  handle_error "Failed to find Amazon Linux 2023 AMI"
fi

log "Selected AMI: $AMI_ID"

# Get the architecture of the AMI
log "Getting AMI architecture..."
AMI_ARCH=$(aws ec2 describe-images \
  --image-ids "$AMI_ID" \
  --query "Images[0].Architecture" \
  --output text)

if [ $? -ne 0 ] || [ -z "$AMI_ARCH" ]; then
  handle_error "Failed to get AMI architecture"
fi

log "AMI architecture: $AMI_ARCH"

# Find a compatible instance type
log "Finding compatible instance type..."
# Directly use t2.micro for simplicity
INSTANCE_TYPE="t2.micro"
log "Using instance type: $INSTANCE_TYPE"

# Step 4: Launch an EC2 instance with enhanced security
log "Launching EC2 instance with IMDSv2 and encryption enabled..."
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --metadata-options "HttpTokens=required,HttpEndpoint=enabled" \
  --block-device-mappings "DeviceName=/dev/xvda,Ebs={Encrypted=true}" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' 'ResourceType=volume,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --count 1 \
```

```
  --query 'Instances[0].InstanceId' \
  --output text)

if [ $? -ne 0 ] || [ -z "$INSTANCE_ID" ]; then
  handle_error "Failed to launch EC2 instance"
fi

log "Launched instance $INSTANCE_ID. Waiting for it to start..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get instance details
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].
{ID:InstanceId,Type:InstanceType,State:State.Name,PublicIP:PublicIpAddress}' \
  --output json)

if [ $? -ne 0 ]; then
  handle_error "Failed to get instance details"
fi

log "Instance details: $INSTANCE_DETAILS"

# Get the public IP address
PUBLIC_IP=$(echo "$INSTANCE_DETAILS" | grep -oP '"PublicIP": "\K[^"]+')
if [ -z "$PUBLIC_IP" ]; then
  handle_error "Failed to get instance public IP"
fi

log "Instance public IP: $PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$PUBLIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 6: Stop and Start the Instance
log "Stopping instance $INSTANCE_ID..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
```

```
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get the new public IP address
NEW_PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$NEW_PUBLIC_IP" ]; then
  handle_error "Failed to get new public IP"
fi

log "Instance restarted with new public IP: $NEW_PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$NEW_PUBLIC_IP"

# Step 7: Allocate and Associate an Elastic IP Address
log "Allocating Elastic IP address..."
ALLOCATION_RESULT=$(aws ec2 allocate-address \
  --domain vpc \
  --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' \
  --query '[PublicIp,AllocationId]' \
  --output text)

if [ $? -ne 0 ] || [ -z "$ALLOCATION_RESULT" ]; then
```

```
  handle_error "Failed to allocate Elastic IP"
fi

ELASTIC_IP=$(echo "$ALLOCATION_RESULT" | awk '{print $1}')
ALLOCATION_ID=$(echo "$ALLOCATION_RESULT" | awk '{print $2}')

log "Allocated Elastic IP: $ELASTIC_IP with ID: $ALLOCATION_ID"

log "Associating Elastic IP with instance..."
ASSOCIATION_ID=$(aws ec2 associate-address \
  --instance-id "$INSTANCE_ID" \
  --allocation-id "$ALLOCATION_ID" \
  --query "AssociationId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$ASSOCIATION_ID" ]; then
  handle_error "Failed to associate Elastic IP"
fi

log "Associated Elastic IP with instance. Association ID: $ASSOCIATION_ID"
log "To connect to your instance using the Elastic IP, run: ssh -i $KEY_FILE ec2-
user@$ELASTIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 8: Test the Elastic IP by Stopping and Starting the Instance
log "Stopping instance $INSTANCE_ID to test Elastic IP persistence..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi
```

```
log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Verify the Elastic IP is still associated
CURRENT_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$CURRENT_IP" ]; then
  handle_error "Failed to get current public IP"
fi

log "Current public IP address: $CURRENT_IP"
log "Elastic IP address: $ELASTIC_IP"

if [ "$CURRENT_IP" = "$ELASTIC_IP" ]; then
  log "Success! The Elastic IP is still associated with your instance."
else
  log "Something went wrong. The Elastic IP is not associated with your instance."
fi

log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$ELASTIC_IP"

# Step 9: Clean up resources
log "Tutorial completed successfully!"
cleanup

exit 0
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair

- DeleteSecurityGroup
- DescribeImages
- DescribeInstances
- DisassociateAddress
- GetParametersByPath
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- Wait

Como configurar o gerenciamento de sistemas

O exemplo de código a seguir mostra como:

- Criar permissões do IAM para o Systems Manager.
- Criar um perfil do IAM para o Systems Manager
- Configurar o Systems Manager.
- Verificar a configuração.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
```

```
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id --
policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
```

```
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --manager-
arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null
 || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and region."
echo "It will create IAM policies and roles, then enable Systems Manager features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
```

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
         "ssm:GetPatchBaseline",
         "ssm:DescribeMaintenanceWindows",
         "ssm:DescribeMaintenanceWindowTasks"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmDocument",
       "Effect": "Allow",
       "Action": [
         "ssm:GetDocument",
         "ssm:DescribeDocument"
       ],
       "Resource": [
         "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
         "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
       ]
     },
     {
       "Sid": "SsmEnableExplorer",
       "Effect": "Allow",
       "Action": "ssm:StartAutomationExecution",
```

```
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
```

```
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
```

```
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
```

```
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
```

```
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/accountdiscovery.ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
      }
    ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
```

```
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --policy-
arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"
```

```
# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
```

```
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-arn
 $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up later." |
 tee -a "$LOG_FILE"
```

```
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetConfigurationManager

Exemplos do Amazon Textract usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o Amazon Textract.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Introdução à extração de texto de documentos

O exemplo de código a seguir mostra como:

- Criar um bucket do S3.
- Fazer upload de um documento no S3.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "Amazon Textract Getting Started Tutorial"
echo "==================================================="
echo "This script will guide you through using Amazon Textract to analyze document
 text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
```

```
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*/\1=***REDACTED***/g'
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to delete
 document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ]; then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
```

```
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\).*/
\1=***REDACTED***/g'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
 default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
```

```
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
 2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
 {"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text 50,50
 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
```

```
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "./$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256 2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ./$DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
```

```
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "==================================================="
echo "Analysis Summary"
echo "==================================================="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk '{print
 $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l || echo
 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l || echo
 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l || echo
 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
 echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l || echo
 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"' |
 wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' | wc -
l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
```

```
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "==================================================="
echo "RESOURCES CREATED"
echo "==================================================="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-results.json)
 has been kept."

echo ""
```

```
echo "==================================================="
echo "Tutorial complete!"
echo "==================================================="
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AnalyzeDocument
  - Cp
  - Help
  - Mb
  - Rb
  - Rm

Exemplos de VPC Lattice usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com o VPC Lattice.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criação e gerenciamento de uma malha de rede de serviços

O exemplo de código a seguir mostra como:

- Criar uma rede de serviços.

- Criar um serviço.
- Listar as VPCs disponíveis.
- Listar grupos de segurança para a VPC selecionada.
- Listar associações de serviço.
- Listar associações de VPC.
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# VPC Lattice Service Network Tutorial Script
# This script demonstrates how to create and manage a VPC Lattice service network

set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="vpc-lattice-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting VPC Lattice tutorial script at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_command() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    if [ "$1" -ne 0 ]; then
```

```
        echo "ERROR: Command failed with exit code $1" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        exit "$1"
    fi
}

# Function to validate AWS CLI is available
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" | tee -a "$LOG_FILE"
        exit 1
    fi
}

# Function to validate input parameters
validate_input() {
    local input="$1"
    local param_name="$2"

    if [[ -z "$input" ]]; then
        echo "ERROR: $param_name is empty" | tee -a "$LOG_FILE"
        return 1
    fi

    # Validate against common injection patterns
    if [[ "$input" =~ [\;\$\`\|\&\<\>\(\)\{\}] ]]; then
        echo "ERROR: $param_name contains invalid characters" | tee -a "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to wait for a resource to be in the desired state
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
    local desired_status="$3"
    local command="$4"
    local max_attempts=30
    local attempt=1
    local status=""

    validate_input "$resource_type" "resource_type" || return 1
```

```
   validate_input "$resource_id" "resource_id" || return 1
   validate_input "$desired_status" "desired_status" || return 1

   echo "Waiting for $resource_type $resource_id to be in state $desired_status..."
| tee -a "$LOG_FILE"

   while [ "$attempt" -le "$max_attempts" ]; do
       echo "Attempt $attempt of $max_attempts..." >> "$LOG_FILE"

       # Run the command to get the status and capture the output
       status_output=$(eval "$command" 2>&1) || true
       echo "$status_output" >> "$LOG_FILE"

       # For service networks, they do not have a status field in the output
       # We'll consider them active if we can retrieve them
       if [[ "$resource_type" == "Service Network" ]]; then
           if [[ "$status_output" == *"$resource_id"* ]]; then
               echo "$resource_type $resource_id is now active" | tee -a
"$LOG_FILE"
               return 0
           fi
       else
           # For other resources, extract the status field
           status=$(echo "$status_output" | grep -i "status" | awk -F'"' '{print
$4}' || true)
           echo "Current status: $status" >> "$LOG_FILE"

           if [[ "$status" == "$desired_status" ]]; then
               echo "$resource_type $resource_id is now in state $desired_status" |
tee -a "$LOG_FILE"
               return 0
           elif [[ "$status" == *"FAIL"* ]]; then
               echo "ERROR: $resource_type $resource_id failed to reach desired
state. Current status: $status" | tee -a "$LOG_FILE"
               return 1
           fi
       fi

       echo "Waiting for status change... (attempt $attempt/$max_attempts)" >>
"$LOG_FILE"
       sleep 10
       ((attempt++))
   done
```

```
    echo "ERROR: Timed out waiting for $resource_type $resource_id to reach state
 $desired_status" | tee -a "$LOG_FILE"
    return 1
}

# Cleanup function for trap
cleanup() {
    local exit_code=$?
    echo "Script interrupted or failed. Cleaning up..." | tee -a "$LOG_FILE"
    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Check prerequisites
check_aws_cli

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
SERVICE_NETWORK_NAME="lattice-network-${RANDOM_ID}"
SERVICE_NAME="lattice-service-${RANDOM_ID}"

# Store created resources for cleanup
declare -a CREATED_RESOURCES

echo "=== VPC Lattice Service Network Tutorial ===" | tee -a "$LOG_FILE"
echo "Random ID for this session: ${RANDOM_ID}" | tee -a "$LOG_FILE"

# Step 1: Create a VPC Lattice service network
echo -e "\n=== Step 1: Creating a VPC Lattice service network ===" | tee -a
 "$LOG_FILE"
echo "Creating service network: $SERVICE_NETWORK_NAME" | tee -a "$LOG_FILE"

SERVICE_NETWORK_OUTPUT=$(log_command "aws vpc-lattice create-service-
network --name '$SERVICE_NETWORK_NAME' --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-vpc-lattice-gs --output json")
check_error $?

# Extract the service network ID using jq for safety
SERVICE_NETWORK_ID=$(echo "$SERVICE_NETWORK_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
if [ -z "$SERVICE_NETWORK_ID" ]; then
    echo "ERROR: Failed to extract service network ID" | tee -a "$LOG_FILE"
    exit 1
```

```
fi

validate_input "$SERVICE_NETWORK_ID" "SERVICE_NETWORK_ID" || exit 1

echo "Service network created with ID: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Network: $SERVICE_NETWORK_ID")

# Wait for the service network to be active
wait_for_resource "Service Network" "$SERVICE_NETWORK_ID" "ACTIVE" "aws vpc-lattice
 get-service-network --service-network-identifier '$SERVICE_NETWORK_ID' --output
 json"
check_error $?

# Step 2: Create a VPC Lattice service
echo -e "\n=== Step 2: Creating a VPC Lattice service ===" | tee -a "$LOG_FILE"
echo "Creating service: $SERVICE_NAME" | tee -a "$LOG_FILE"

SERVICE_OUTPUT=$(log_command "aws vpc-lattice create-service --name '$SERVICE_NAME'
 --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-vpc-lattice-gs --
output json")
check_error $?

# Extract the service ID using jq for safety
SERVICE_ID=$(echo "$SERVICE_OUTPUT" | jq -r '.id // empty' 2>/dev/null || true)
if [ -z "$SERVICE_ID" ]; then
    echo "ERROR: Failed to extract service ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ID" "SERVICE_ID" || exit 1

echo "Service created with ID: $SERVICE_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service: $SERVICE_ID")

# Wait for the service to be active
wait_for_resource "Service" "$SERVICE_ID" "ACTIVE" "aws vpc-lattice get-service --
service-identifier '$SERVICE_ID' --output json"
check_error $?

# Step 3: Associate the service with the service network
echo -e "\n=== Step 3: Associating service with service network ===" | tee -a
 "$LOG_FILE"
```

```
SERVICE_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-service-
association --service-identifier '$SERVICE_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json")
check_error $?

# Extract the service association ID using jq for safety
SERVICE_ASSOC_ID=$(echo "$SERVICE_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/null
 || true)
if [ -z "$SERVICE_ASSOC_ID" ]; then
    echo "ERROR: Failed to extract service association ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ASSOC_ID" "SERVICE_ASSOC_ID" || exit 1

echo "Service association created with ID: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Association: $SERVICE_ASSOC_ID")

# Wait for the service association to be active
wait_for_resource "Service Association" "$SERVICE_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-service-association --service-network-service-
association-identifier '$SERVICE_ASSOC_ID' --output json"
check_error $?

# Step 4: List available VPCs to associate with the service network
echo -e "\n=== Step 4: Listing available VPCs ===" | tee -a "$LOG_FILE"

VPC_LIST=$(log_command "aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId,Tags[?Key==
\`Name\`].Value|[0]]' --output text")
check_error $?

echo "Available VPCs:" | tee -a "$LOG_FILE"
echo "$VPC_LIST" | tee -a "$LOG_FILE"

# Step 5: Auto-select first available VPC
echo -e "\n=== Step 5: Associate a VPC with the service network ===" | tee -a
 "$LOG_FILE"

VPC_ID=$(echo "$VPC_LIST" | head -n 1 | awk '{print $1}')

if [ -z "$VPC_ID" ]; then
    echo "WARNING: No VPC ID found" | tee -a "$LOG_FILE"
    echo "Skipping VPC association step" | tee -a "$LOG_FILE"
else
```

```
    validate_input "$VPC_ID" "VPC_ID" || {
        echo "ERROR: VPC_ID validation failed"
        exit 1
    }

    echo "Auto-selected VPC: $VPC_ID" | tee -a "$LOG_FILE"

    # Step 6: List security groups for the selected VPC
    echo -e "\n=== Step 6: Listing security groups for VPC $VPC_ID ===" | tee -a
 "$LOG_FILE"

    SG_LIST=$(log_command "aws ec2 describe-security-groups --filters Name=vpc-
id,Values='$VPC_ID' --query 'SecurityGroups[*].[GroupId,GroupName]' --output text")
    check_error $?

    echo "Available Security Groups for VPC $VPC_ID:" | tee -a "$LOG_FILE"
    echo "$SG_LIST" | tee -a "$LOG_FILE"

    # Step 7: Auto-select first available security group
    echo -e "\n=== Step 7: Select a security group for the VPC association ===" |
 tee -a "$LOG_FILE"

    SG_ID=$(echo "$SG_LIST" | head -n 1 | awk '{print $1}')

    if [ -z "$SG_ID" ]; then
        echo "WARNING: No Security Group ID found" | tee -a "$LOG_FILE"
        echo "Skipping VPC association step" | tee -a "$LOG_FILE"
    else
        validate_input "$SG_ID" "SG_ID" || {
            echo "ERROR: SG_ID validation failed"
            exit 1
        }

        echo "Auto-selected Security Group: $SG_ID" | tee -a "$LOG_FILE"

        # Step 8: Associate the VPC with the service network
        echo -e "\n=== Step 8: Associating VPC with service network ===" | tee -a
 "$LOG_FILE"

        VPC_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
vpc-association --vpc-identifier '$VPC_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --security-group-ids '$SG_ID' --output json")
        check_error $?
```

```
        # Extract the VPC association ID using jq for safety
        VPC_ASSOC_ID=$(echo "$VPC_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/null
 || true)
        if [ -z "$VPC_ASSOC_ID" ]; then
            echo "ERROR: Failed to extract VPC association ID" | tee -a "$LOG_FILE"
        else
            validate_input "$VPC_ASSOC_ID" "VPC_ASSOC_ID" || exit 1

            echo "VPC association created with ID: $VPC_ASSOC_ID" | tee -a
 "$LOG_FILE"
            CREATED_RESOURCES+=("VPC Association: $VPC_ASSOC_ID")

            # Wait for the VPC association to be active
            wait_for_resource "VPC Association" "$VPC_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-vpc-association --service-network-vpc-association-
identifier '$VPC_ASSOC_ID' --output json"
            check_error $?
        fi
    fi
fi

# Step 9: Display information about the created resources
echo -e "\n=== Step 9: Displaying information about created resources ===" | tee -a
 "$LOG_FILE"

echo "Service Network Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json"

echo "Service Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service --service-identifier '$SERVICE_ID' --output
 json"

echo "Service Network Service Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-service-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

echo "Service Network VPC Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-vpc-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

# Step 10: Cleanup - Auto-confirm
echo -e "\n=== Step 10: Resource Cleanup ===" | tee -a "$LOG_FILE"
echo "Resources created in this tutorial:" | tee -a "$LOG_FILE"
```

```
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
done

echo "Starting cleanup process..." | tee -a "$LOG_FILE"

# Delete resources in reverse order

# Delete VPC association if it was created
if [[ -n "${VPC_ASSOC_ID:-}" ]]; then
    echo "Deleting VPC association: $VPC_ASSOC_ID" | tee -a "$LOG_FILE"
    log_command "aws vpc-lattice delete-service-network-vpc-association --service-
network-vpc-association-identifier '$VPC_ASSOC_ID'" || true

    # Wait for the VPC association to be deleted
    echo "Waiting for VPC association to be deleted..." | tee -a "$LOG_FILE"
    sleep 30
fi

# Delete service association
echo "Deleting service association: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network-service-association --service-
network-service-association-identifier '$SERVICE_ASSOC_ID'" || true

# Wait for the service association to be deleted
echo "Waiting for service association to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service
echo "Deleting service: $SERVICE_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service --service-identifier '$SERVICE_ID'" ||
 true

# Wait for the service to be deleted
echo "Waiting for service to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service network
echo "Deleting service network: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID'" || true

echo "Cleanup completed successfully!" | tee -a "$LOG_FILE"
```

```
echo -e "\n=== Tutorial completed! ===" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateService
  - CreateServiceNetwork
  - CreateServiceNetworkServiceAssociation
  - CreateServiceNetworkVpcAssociation
  - DeleteService
  - DeleteServiceNetwork
  - DeleteServiceNetworkServiceAssociation
  - DeleteServiceNetworkVpcAssociation
  - DescribeSecurityGroups
  - DescribeVpcs
  - GetService
  - GetServiceNetwork
  - GetServiceNetworkServiceAssociation
  - GetServiceNetworkVpcAssociation
  - ListServiceNetworkServiceAssociations
  - ListServiceNetworkVpcAssociations

AWS WAFV2 exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com AWS WAFV2.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Conceitos básicos de firewalls do aplicativo Web

O exemplo de código a seguir mostra como:

- Criar uma ACL da web.
- Adicionar uma regra de correspondência de string.
- Adicionar regras gerenciadas.
- Configurar o registro em log da
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
```

```
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="
```

```
    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/
dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
```

```
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the AWS
 Console."
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
```

```
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "====================================================="
echo "STEP 2: Adding String Match Rule"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
```

```
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
|| true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
```

```
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
```

```
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
 || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi
```

```
    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
```

```
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "====================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "====================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""
```

```
if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "====================================================="
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "====================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo "$first_dist" |
 grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution successfully."
```

```
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl

- ListDistributions
- UpdateWebAcl

WorkSpaces exemplos usando AWS CLI com script Bash

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o script AWS Command Line Interface with Bash com WorkSpaces.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criação e gerenciamento de áreas de trabalho virtuais pessoais

O exemplo de código a seguir mostra como:

- Verifique WorkSpaces a disponibilidade em sua região
- Listar WorkSpaces pacotes disponíveis
- Crie um WorkSpace
- Verifique o status do seu WorkSpace
- Limpar os recursos

AWS CLI com script Bash

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Sample developer tutorials.

```
#!/bin/bash

# Script to create a WorkSpace in WorkSpaces Personal
# This script follows the workflow described in the AWS documentation
# https://docs.aws.amazon.com/workspaces/latest/adminguide/create-workspaces-
personal.html

# Set up logging
LOG_FILE="workspaces_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting WorkSpaces creation script"
echo "==============================================="

# Initialize resource tracking array
declare -a CREATED_RESOURCES

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    exit 1
}

# Function to check if a command succeeded
check_command() {
    # Check for ResourceNotFound.User error specifically
    if echo "$1" | grep -q "ResourceNotFound.User"; then
        echo ""
        echo "ERROR: User not found in the directory."
        echo ""
        echo "This error occurs when the specified username doesn't exist in the
 directory."
        echo ""
        echo "To resolve this issue:"
        echo "1. Ensure the user exists in the directory before creating a
 WorkSpace."
        echo "2. For Simple AD and AWS Managed Microsoft AD:"
        echo "   - Connect to a directory-joined instance"
        echo "   - Use Active Directory tools to create the user"
```

```
        echo "   - See: https://docs.aws.amazon.com/workspaces/latest/adminguide/
manage-users.html"
        echo ""
        echo "3. For AD Connector:"
        echo "   - Create the user in your on-premises Active Directory"
        echo "   - Ensure proper synchronization with the AD Connector"
        echo ""
        echo "4. Alternatively, you can use the AWS Console to create a WorkSpace,"
        echo "   which can create the user automatically in some directory types."
        echo ""
        handle_error "User '$USERNAME' not found in directory '$DIRECTORY_ID'"
    # Check for other errors
    elif echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Step 0: Select AWS region
echo ""
echo "==============================================="
echo "AWS REGION SELECTION"
echo "==============================================="
echo "Enter the AWS region to use (e.g., us-east-1, us-west-2):"
AWS_REGION="${AWS_REGION:-us-east-1}"

if [ -z "$AWS_REGION" ]; then
    handle_error "Region cannot be empty"
fi

export AWS_DEFAULT_REGION="$AWS_REGION"
echo "Using AWS region: $AWS_REGION"

# Step 1: Prompt for directory ID
echo ""
echo "==============================================="
echo "DIRECTORY SELECTION"
echo "==============================================="
echo "Listing available directories..."

DIRECTORIES_OUTPUT=$(aws workspaces describe-workspace-directories --output json)
check_command "$DIRECTORIES_OUTPUT"
echo "$DIRECTORIES_OUTPUT"

# Extract directory IDs and display them
```

```
DIRECTORY_IDS=$(echo "$DIRECTORIES_OUTPUT" | grep -o '"DirectoryId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$DIRECTORY_IDS" ]; then
    echo "No directories found. Please create a directory first using AWS Directory
 Service."
    echo "For more information, see: https://docs.aws.amazon.com/workspaces/latest/
adminguide/register-deregister-directory.html"
    exit 1
fi

echo ""
echo "Available directory IDs:"
echo "$DIRECTORY_IDS"
echo ""
echo "Enter the directory ID you want to use:"
DIRECTORY_ID=$(echo "$DIRECTORY_IDS" | head -1)

# Validate directory ID
if ! echo "$DIRECTORY_IDS" | grep -q "$DIRECTORY_ID"; then
    echo "Directory ID $DIRECTORY_ID not found in the list of available
 directories."
    echo "Please check the ID and try again."
    exit 1
fi

echo "Selected directory ID: $DIRECTORY_ID"

# Step 2: Check if directory is registered with WorkSpaces
echo ""
echo "==============================================="
echo "CHECKING DIRECTORY REGISTRATION"
echo "==============================================="

REGISTERED=$(echo "$DIRECTORIES_OUTPUT" | grep -A 5 "\"DirectoryId\":
 \"$DIRECTORY_ID\"" | grep -c "\"State\": \"REGISTERED\"")

if [ "$REGISTERED" -eq 0 ]; then
    echo "Directory $DIRECTORY_ID is not registered with WorkSpaces. Registering
 now..."
    REGISTER_OUTPUT=$(aws workspaces register-workspace-directory --directory-id
 "$DIRECTORY_ID")
    check_command "$REGISTER_OUTPUT"
    echo "Directory registration initiated. This may take a few minutes."
```

```
    # Add to resource tracking
    CREATED_RESOURCES+=("Directory registration: $DIRECTORY_ID")

    # Wait for directory to be registered
    echo "Waiting for directory registration to complete..."
    sleep 30

    # Check registration status
    REGISTRATION_CHECK=$(aws workspaces describe-workspace-directories --directory-
ids "$DIRECTORY_ID")
    check_command "$REGISTRATION_CHECK"

    REGISTRATION_STATE=$(echo "$REGISTRATION_CHECK" | grep -o '"State": "[^"]*' |
 cut -d'"' -f4)
    if [ "$REGISTRATION_STATE" != "REGISTERED" ]; then
        echo "Directory registration is still in progress. Current state:
 $REGISTRATION_STATE"
        echo "Please check the AWS console for the final status."
        echo "You may need to wait a few minutes before proceeding."
    else
        echo "Directory successfully registered with WorkSpaces."
    fi
else
    echo "Directory $DIRECTORY_ID is already registered with WorkSpaces."
fi

# Get directory type to provide appropriate user guidance
DIRECTORY_TYPE=$(echo "$DIRECTORIES_OUTPUT" | grep -A 10 "\"DirectoryId\":
 \"$DIRECTORY_ID\"" | grep -o '"DirectoryType": "[^"]*' | cut -d'"' -f4)
echo "Directory type: $DIRECTORY_TYPE"

# Display user creation guidance based on directory type
echo ""
echo "==============================================="
echo "USER CREATION GUIDANCE"
echo "==============================================="
case "$DIRECTORY_TYPE" in
    "SimpleAD" | "MicrosoftAD")
        echo "For $DIRECTORY_TYPE, users must be created using Active Directory
 tools."
        echo "1. Connect to a directory-joined EC2 instance"
        echo "2. Use Active Directory Users and Computers to create users"
```

```
        echo "3. For detailed instructions, see: https://docs.aws.amazon.com/
workspaces/latest/adminguide/manage-users.html"
        ;;
    "ADConnector")
        echo "For AD Connector, users must exist in your on-premises Active
 Directory."
        echo "1. Create the user in your on-premises Active Directory"
        echo "2. Ensure the user is in an OU that is within the scope of your AD
 Connector"
        echo "3. For detailed instructions, see: https://docs.aws.amazon.com/
directoryservice/latest/admin-guide/ad_connector_management.html"
        ;;
    *)
        echo "For this directory type, ensure users exist before creating
 WorkSpaces."
        echo "For detailed instructions, see: https://docs.aws.amazon.com/
workspaces/latest/adminguide/manage-users.html"
        ;;
esac
echo ""

# Step 3: List available bundles
echo ""
echo "==============================================="
echo "BUNDLE SELECTION"
echo "==============================================="
echo "Listing available WorkSpace bundles..."

# Get bundles with a format that's easier to parse
BUNDLES_OUTPUT=$(aws workspaces describe-workspace-bundles --owner AMAZON --output
 text --query "Bundles[*].
[BundleId,Name,ComputeType.Name,RootStorage.Capacity,UserStorage.Capacity]")
check_command "$BUNDLES_OUTPUT"

# Extract bundle information and display in a numbered list
echo "Available bundles:"
echo "-----------------"
echo "NUM | BUNDLE ID | NAME | COMPUTE TYPE | ROOT STORAGE | USER STORAGE"
echo "-------------------------------------------------------------------"

# Create arrays to store bundle information
declare -a BUNDLE_IDS
declare -a BUNDLE_NAMES
```

```
# Process the output to extract bundle information
COUNT=1
while IFS=$'\t' read -r BUNDLE_ID BUNDLE_NAME COMPUTE_TYPE ROOT_STORAGE USER_STORAGE
 || [[ -n "$BUNDLE_ID" ]]; do
    # Store in arrays
    BUNDLE_IDS[$COUNT]="$BUNDLE_ID"
    BUNDLE_NAMES[$COUNT]="$BUNDLE_NAME"

    # Display with number
    echo "$COUNT | $BUNDLE_ID | $BUNDLE_NAME | $COMPUTE_TYPE | $ROOT_STORAGE GB |
 $USER_STORAGE GB"

    ((COUNT++))
done <<< "$BUNDLES_OUTPUT"

# Prompt for selection
echo ""
echo "Enter the number of the bundle you want to use (1-$((COUNT-1))):"
BUNDLE_SELECTION="1"

# Validate selection
if ! [[ "$BUNDLE_SELECTION" =~ ^[0-9]+$ ]] || [ "$BUNDLE_SELECTION" -lt 1 ] ||
 [ "$BUNDLE_SELECTION" -ge "$COUNT" ]; then
    handle_error "Invalid bundle selection. Please enter a number between 1 and
 $((COUNT-1))."
fi

# Get the selected bundle ID
BUNDLE_ID="${BUNDLE_IDS[$BUNDLE_SELECTION]}"
BUNDLE_NAME="${BUNDLE_NAMES[$BUNDLE_SELECTION]}"

echo "Selected bundle: $BUNDLE_NAME (ID: $BUNDLE_ID)"

# Step 4: Prompt for username
echo ""
echo "==============================================="
echo "USER INFORMATION"
echo "==============================================="
echo "Enter the username for the WorkSpace:"
USERNAME="testuser-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n 1)"

echo "NOTE: The user must already exist in the directory for the WorkSpace creation
 to succeed."
```

```
echo "If you're using Simple AD or AWS Managed Microsoft AD, the user must be
 created using Active Directory tools."
echo "If you're using AD Connector, the user must exist in your on-premises Active
 Directory."
echo ""

echo "Enter the user's first name:"
FIRST_NAME="Test"

echo "Enter the user's last name:"
LAST_NAME="User"

echo "Enter the user's email address:"
EMAIL="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1)@example.com"

# Step 5: Choose running mode
echo ""
echo "================================================"
echo "RUNNING MODE SELECTION"
echo "================================================"
echo "Select running mode:"
echo "1. AlwaysOn (billed monthly)"
echo "2. AutoStop (billed hourly)"
RUNNING_MODE_CHOICE="1"

if [ "$RUNNING_MODE_CHOICE" = "1" ]; then
    RUNNING_MODE="ALWAYS_ON"
    AUTO_STOP_TIMEOUT=""
else
    RUNNING_MODE="AUTO_STOP"
    AUTO_STOP_TIMEOUT=60
fi

echo "Selected running mode: $RUNNING_MODE"

# Step 6: Add tags (optional)
echo ""
echo "================================================"
echo "TAGS (OPTIONAL)"
echo "================================================"
echo "Would you like to add tags to your WorkSpace? (y/n):"
ADD_TAGS="n"
```

```
TAGS_JSON=""
if [ "$ADD_TAGS" = "y" ] || [ "$ADD_TAGS" = "Y" ]; then
    echo "Enter tag key (e.g., Department):"
    read -r TAG_KEY

    echo "Enter tag value (e.g., IT):"
    read -r TAG_VALUE

    TAGS_JSON="[{\"Key\":\"$TAG_KEY\",\"Value\":\"$TAG_VALUE\"}]"
fi

# Step 7: Create the WorkSpace
echo ""
echo "==============================================="
echo "CREATING WORKSPACE"
echo "==============================================="
echo "Creating WorkSpace with the following parameters:"
echo "Directory ID: $DIRECTORY_ID"
echo "Username: $USERNAME"
echo "Bundle ID: $BUNDLE_ID"
echo "Running Mode: $RUNNING_MODE"
if [ -n "$TAGS_JSON" ]; then
    echo "Tags: $TAG_KEY=$TAG_VALUE"
fi

# Create JSON for workspace properties
if [ "$RUNNING_MODE" = "AUTO_STOP" ]; then
    PROPERTIES_JSON="{\"RunningMode\":\"$RUNNING_MODE\",
\"RunningModeAutoStopTimeoutInMinutes\":$AUTO_STOP_TIMEOUT}"
else
    PROPERTIES_JSON="{\"RunningMode\":\"$RUNNING_MODE\"}"
fi

# Create JSON for workspaces parameter
WORKSPACE_JSON="{\"DirectoryId\":\"$DIRECTORY_ID\",\"UserName\":\"$USERNAME\",
\"BundleId\":\"$BUNDLE_ID\",\"WorkspaceProperties\":$PROPERTIES_JSON"

# Add tags if specified
if [ -n "$TAGS_JSON" ]; then
    USER_TAGS=$(echo "$TAGS_JSON" | sed 's/^\[//;s/\]$//')
    WORKSPACE_JSON="$WORKSPACE_JSON,\"Tags\":[$USER_TAGS,{\"Key\":\"project\",
\"Value\":\"doc-smith\"},{\"Key\":\"tutorial\",\"Value\":\"workspaces-personal\"}]"
else
```

```
    WORKSPACE_JSON="$WORKSPACE_JSON,\"Tags\":[{\"Key\":\"project\",\"Value\":\"doc-
smith\"},{\"Key\":\"tutorial\",\"Value\":\"workspaces-personal\"}]"
fi

# Close the JSON object
WORKSPACE_JSON="$WORKSPACE_JSON}"

# Construct the create-workspaces command
CREATE_COMMAND="aws workspaces create-workspaces --workspaces '$WORKSPACE_JSON'"

echo "Executing: $CREATE_COMMAND"
CREATE_OUTPUT=$(eval "$CREATE_COMMAND")
check_command "$CREATE_OUTPUT"
echo "$CREATE_OUTPUT"

# Extract WorkSpace ID
WORKSPACE_ID=$(echo "$CREATE_OUTPUT" | grep -o '"WorkspaceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    handle_error "Failed to extract WorkSpace ID from creation output."
fi

echo "WorkSpace creation initiated. WorkSpace ID: $WORKSPACE_ID"
CREATED_RESOURCES+=("WorkSpace: $WORKSPACE_ID")

# Step 8: Check WorkSpace status
echo ""
echo "==============================================="
echo "CHECKING WORKSPACE STATUS"
echo "==============================================="
echo "Checking status of WorkSpace $WORKSPACE_ID..."

# Initial status check
STATUS_OUTPUT=$(aws workspaces describe-workspaces --workspace-ids "$WORKSPACE_ID")
check_command "$STATUS_OUTPUT"
echo "$STATUS_OUTPUT"

WORKSPACE_STATE=$(echo "$STATUS_OUTPUT" | grep -o '"State": "[^"]*' | head -1 | cut
 -d'"' -f4)
echo "Current WorkSpace state: $WORKSPACE_STATE"

# Wait for WorkSpace to be available (this can take 20+ minutes)
echo ""
```

```
echo "WorkSpace creation is in progress. This can take 20+ minutes."
echo "The script will check the status every 60 seconds."
echo "Press Ctrl+C to exit the script at any time. The WorkSpace will continue to be
 created."

while [ "$WORKSPACE_STATE" = "PENDING" ]; do
    echo "$(date): WorkSpace state is still PENDING. Waiting 60 seconds before
 checking again..."
    sleep 60

    STATUS_OUTPUT=$(aws workspaces describe-workspaces --workspace-ids
 "$WORKSPACE_ID")
    check_command "$STATUS_OUTPUT"

    WORKSPACE_STATE=$(echo "$STATUS_OUTPUT" | grep -o '"State": "[^"]*' | head -1 |
 cut -d'"' -f4)
    echo "$(date): Current WorkSpace state: $WORKSPACE_STATE"

    # If state is ERROR or UNHEALTHY, exit
    if [ "$WORKSPACE_STATE" = "ERROR" ] || [ "$WORKSPACE_STATE" = "UNHEALTHY" ];
 then
        handle_error "WorkSpace creation failed. Final state: $WORKSPACE_STATE"
    fi

    # If state is AVAILABLE, break the loop
    if [ "$WORKSPACE_STATE" = "AVAILABLE" ]; then
        break
    fi
done

# Step 9: Display WorkSpace information
echo ""
echo "================================================"
echo "WORKSPACE CREATION COMPLETE"
echo "================================================"
echo "WorkSpace has been successfully created!"
echo "WorkSpace ID: $WORKSPACE_ID"
echo "Directory ID: $DIRECTORY_ID"
echo "Username: $USERNAME"
echo "Running Mode: $RUNNING_MODE"

# Step 10: Remind about invitation emails
echo ""
echo "================================================"
```

```
echo "INVITATION EMAILS"
echo "==============================================="
echo "IMPORTANT: If you're using AD Connector or a trust relationship, or if the
 user already exists in Active Directory,"
echo "invitation emails are not sent automatically. You'll need to manually send an
 invitation email."
echo "For more information, see: https://docs.aws.amazon.com/workspaces/latest/
adminguide/manage-workspaces-users.html#send-invitation"

# Step 11: Cleanup confirmation
echo ""
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

echo ""
echo "Do you want to clean up all created resources? (y/n):"
CLEANUP_CHOICE="y"

if [ "$CLEANUP_CHOICE" = "y" ] || [ "$CLEANUP_CHOICE" = "Y" ]; then
    echo ""
    echo "==============================================="
    echo "CLEANING UP RESOURCES"
    echo "==============================================="

    # Terminate WorkSpace
    if [ -n "$WORKSPACE_ID" ]; then
        echo "Terminating WorkSpace $WORKSPACE_ID..."
        TERMINATE_OUTPUT=$(aws workspaces terminate-workspaces --terminate-
workspace-requests WorkspaceId="$WORKSPACE_ID")
        check_command "$TERMINATE_OUTPUT"
        echo "$TERMINATE_OUTPUT"
        echo "WorkSpace termination initiated. This may take a few minutes."
    fi

    # Deregister directory (only if we registered it in this script)
    if [[ " ${CREATED_RESOURCES[*]} " == *"Directory registration:
 $DIRECTORY_ID"* ]]; then
        echo "Deregistering directory $DIRECTORY_ID from WorkSpaces..."
```

```
        DEREGISTER_OUTPUT=$(aws workspaces deregister-workspace-directory --
directory-id "$DIRECTORY_ID")
        check_command "$DEREGISTER_OUTPUT"
        echo "$DEREGISTER_OUTPUT"
        echo "Directory deregistration initiated. This may take a few minutes."
    fi

    echo "Cleanup completed."
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo ""
echo "================================================"
echo "SCRIPT COMPLETED"
echo "================================================"
echo "Log file: $LOG_FILE"
echo "Thank you for using the WorkSpaces creation script!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI .
  - CreateWorkspaces
  - DeregisterWorkspaceDirectory
  - DescribeWorkspaceBundles
  - DescribeWorkspaceDirectories
  - DescribeWorkspaces
  - RegisterWorkspaceDirectory
  - TerminateWorkspaces

Exemplos de código para o SDK para C++

Os exemplos de código a seguir mostram como usar o AWS SDK para C++ with AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para C++ — Saiba mais sobre como usar o C++ com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos do ACM usando o SDK para C++
- Exemplos da API Gateway usando o SDK para C++
- Exemplos do Aurora usando o SDK para C++
- Exemplos do Auto Scaling usando o SDK para C++
- CloudTrail exemplos de uso do SDK para C++
- CloudWatch exemplos de uso do SDK para C++
- CloudWatch Exemplos de registros usando o SDK para C++
- CodeBuild exemplos de uso do SDK para C++
- Exemplos do Provedor de Identidade do Amazon Cognito usando o SDK para C++
- Exemplos do DynamoDB usando o SDK para C++
- Exemplos do Amazon EC2 usando o SDK para C++
- EventBridge exemplos de uso do SDK para C++
- AWS Glue exemplos de uso do SDK para C++
- HealthImaging exemplos de uso do SDK para C++
- Exemplos do IAM usando o SDK para C++
- AWS IoT exemplos de uso do SDK para C++
- AWS IoT data exemplos de uso do SDK para C++
- Exemplos do Lambda usando o SDK para C++

- MediaConvert exemplos de uso do SDK para C++
- Exemplos do Amazon RDS usando o SDK para C++
- Exemplos do Amazon RDS Data Service usando o SDK para C++
- Exemplos do Amazon Rekognition usando o SDK para C++
- Exemplos do Amazon S3 usando o SDK para C++
- Exemplos do Secrets Manager usando o SDK para C++
- Exemplos do Amazon SES usando o SDK para C++
- Exemplos do Amazon SNS usando o SDK para C++
- Exemplos do Amazon SQS usando o SDK para C++
- AWS STS exemplos de uso do SDK para C++
- Exemplos do Amazon Transcribe Streaming usando o SDK para C++

Exemplos do ACM usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o ACM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AddTagsToCertificate`

O código de exemplo a seguir mostra como usar `AddTagsToCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Add tags to an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param tagKey: The key for the tag.
  \param tagValue: The value for the tag.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::addTagsToCertificate(const Aws::String &certificateArn,
                                       const Aws::String &tagKey,
                                       const Aws::String &tagValue,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::AddTagsToCertificateRequest request;
    Aws::Vector<Aws::ACM::Model::Tag> tags;
    Aws::ACM::Model::Tag tag;

    tag.WithKey(tagKey).WithValue(tagValue);
    tags.push_back(tag);

    request.WithCertificateArn(certificateArn).WithTags(tags);

    Aws::ACM::Model::AddTagsToCertificateOutcome outcome =
            acmClient.AddTagsToCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: addTagsToCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Tag with key '" << tagKey <<
                  "' and value '" << tagValue <<
```

```
                    "' added to certificate with ARN '" <<
                    certificateArn << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AddTagsToCertificate em Referência AWS SDK para C++ da API.

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::deleteCertificate(const Aws::String &certificateArn,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::DeleteCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DeleteCertificateOutcome outcome =
            acmClient.DeleteCertificate(request);

    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "Error: DeleteCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: The certificate with the ARN '" <<
                  certificateArn << "' is deleted." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteCertificate em Referência AWS SDK para C++ da API.

`DescribeCertificate`

O código de exemplo a seguir mostra como usar `DescribeCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::describeCertificate(const Aws::String &certificateArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::DescribeCertificateRequest request;
    request.WithCertificateArn(certificateArn);
```

```
    Aws::ACM::Model::DescribeCertificateOutcome outcome =
            acm_client.DescribeCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DescribeCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::ACM::Model::CertificateDetail certificate =
                outcome.GetResult().GetCertificate();

        std::cout << "Success: Information about certificate "
                     "with ARN '" << certificateArn << "':" << std::endl <<
std::endl;

        std::cout << "ARN:                 " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << "Authority ARN:       " <<
                  certificate.GetCertificateAuthorityArn() << std::endl;
        std::cout << "Created at (GMT):    " <<
                  certificate.GetCreatedAt().ToGmtString(
                          Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
        std::cout << "Domain name:         " << certificate.GetDomainName()
                  << std::endl;

        Aws::Vector<Aws::ACM::Model::DomainValidation> options =
                certificate.GetDomainValidationOptions();

        if (!options.empty()) {
            std::cout << std::endl << "Domain validation information: "
                      << std::endl << std::endl;

            for (auto &validation: options) {
                std::cout << "  Domain name:              " <<
                          validation.GetDomainName() << std::endl;

                const Aws::ACM::Model::ResourceRecord &record =
                        validation.GetResourceRecord();

                std::cout << "  Resource record name:     " <<
                          record.GetName() << std::endl;

                Aws::ACM::Model::RecordType recordType = record.GetType();
```

```
                Aws::String type;

                switch (recordType) {
                    case Aws::ACM::Model::RecordType::CNAME:
                        type = "CNAME";
                        break;
                    case Aws::ACM::Model::RecordType::NOT_SET:
                        type = "Not set";
                        break;
                    default:
                        type = "Cannot determine.";
                        break;
                }

                std::cout << "  Resource record type:     " << type <<
                          std::endl;

                std::cout << "  Resource record value:    " <<
                          record.GetValue() << std::endl;

                std::cout << "  Validation domain:        " <<
                          validation.GetValidationDomain() << std::endl;

                Aws::Vector<Aws::String> emails =
                        validation.GetValidationEmails();

                if (!emails.empty()) {
                    std::cout << "  Validation emails:" << std::endl <<
                              std::endl;

                    for (auto &email: emails) {
                        std::cout << "    " << email << std::endl;
                    }

                    std::cout << std::endl;
                }

                Aws::ACM::Model::ValidationMethod validationMethod =
                        validation.GetValidationMethod();
                Aws::String method;

                switch (validationMethod) {
                    case Aws::ACM::Model::ValidationMethod::DNS:
                        method = "DNS";
```

```
                    break;
                case Aws::ACM::Model::ValidationMethod::EMAIL:
                    method = "Email";
                    break;
                case Aws::ACM::Model::ValidationMethod::NOT_SET:
                    method = "Not set";
                    break;
                default:
                    method = "Cannot determine";
            }

            std::cout << "  Validation method:        " <<
                      method << std::endl;

            Aws::ACM::Model::DomainStatus domainStatus =
                    validation.GetValidationStatus();
            Aws::String status;

            switch (domainStatus) {
                case Aws::ACM::Model::DomainStatus::FAILED:
                    status = "Failed";
                    break;
                case Aws::ACM::Model::DomainStatus::NOT_SET:
                    status = "Not set";
                    break;
                case Aws::ACM::Model::DomainStatus::PENDING_VALIDATION:
                    status = "Pending validation";
                    break;
                case Aws::ACM::Model::DomainStatus::SUCCESS:
                    status = "Success";
                    break;
                default:
                    status = "Cannot determine";
            }

            std::cout << "  Domain validation status: " << status <<
                      std::endl << std::endl;

        }
    }

    Aws::Vector<Aws::ACM::Model::ExtendedKeyUsage> usages =
            certificate.GetExtendedKeyUsages();
```

```
        if (!usages.empty()) {
            std::cout << std::endl << "Extended key usages:" <<
                      std::endl << std::endl;

            for (auto &usage: usages) {
                Aws::ACM::Model::ExtendedKeyUsageName usageName =
                        usage.GetName();
                Aws::String name;

                switch (usageName) {
                    case Aws::ACM::Model::ExtendedKeyUsageName::ANY:
                        name = "Any";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::CODE_SIGNING:
                        name = "Code signing";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::CUSTOM:
                        name = "Custom";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::EMAIL_PROTECTION:
                        name = "Email protection";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_END_SYSTEM:
                        name = "IPSEC end system";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_TUNNEL:
                        name = "IPSEC tunnel";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_USER:
                        name = "IPSEC user";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::NONE:
                        name = "None";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::NOT_SET:
                        name = "Not set";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::OCSP_SIGNING:
                        name = "OCSP signing";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::TIME_STAMPING:
                        name = "Time stamping";
                        break;
```

```
                    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_CLIENT_AUTHENTICATION:
                        name = "TLS web client authentication";
                        break;
                    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_SERVER_AUTHENTICATION:
                        name = "TLS web server authentication";
                        break;
                    default:
                        name = "Cannot determine";
                }

                std::cout << "  Name: " << name << std::endl;
                std::cout << "  OID:  " << usage.GetOID() <<
                          std::endl << std::endl;
            }

            std::cout << std::endl;
        }

        Aws::ACM::Model::CertificateStatus certificateStatus =
                certificate.GetStatus();
        Aws::String status;

        switch (certificateStatus) {
            case Aws::ACM::Model::CertificateStatus::EXPIRED:
                status = "Expired";
                break;
            case Aws::ACM::Model::CertificateStatus::FAILED:
                status = "Failed";
                break;
            case Aws::ACM::Model::CertificateStatus::INACTIVE:
                status = "Inactive";
                break;
            case Aws::ACM::Model::CertificateStatus::ISSUED:
                status = "Issued";
                break;
            case Aws::ACM::Model::CertificateStatus::NOT_SET:
                status = "Not set";
                break;
            case Aws::ACM::Model::CertificateStatus::PENDING_VALIDATION:
                status = "Pending validation";
                break;
            case Aws::ACM::Model::CertificateStatus::REVOKED:
```

```
                status = "Revoked";
                break;
            case Aws::ACM::Model::CertificateStatus::VALIDATION_TIMED_OUT:
                status = "Validation timed out";
                break;
            default:
                status = "Cannot determine";
        }

        std::cout << "Status:              " << status << std::endl;

        if (certificate.GetStatus() ==
            Aws::ACM::Model::CertificateStatus::FAILED) {
            Aws::ACM::Model::FailureReason failureReason =
                    certificate.GetFailureReason();
            Aws::String reason;

            switch (failureReason) {
                case
 Aws::ACM::Model::FailureReason::ADDITIONAL_VERIFICATION_REQUIRED:
                    reason = "Additional verification required";
                    break;
                case Aws::ACM::Model::FailureReason::CAA_ERROR:
                    reason = "CAA error";
                    break;
                case Aws::ACM::Model::FailureReason::DOMAIN_NOT_ALLOWED:
                    reason = "Domain not allowed";
                    break;
                case Aws::ACM::Model::FailureReason::DOMAIN_VALIDATION_DENIED:
                    reason = "Domain validation denied";
                    break;
                case Aws::ACM::Model::FailureReason::INVALID_PUBLIC_DOMAIN:
                    reason = "Invalid public domain";
                    break;
                case Aws::ACM::Model::FailureReason::NOT_SET:
                    reason = "Not set";
                    break;
                case Aws::ACM::Model::FailureReason::NO_AVAILABLE_CONTACTS:
                    reason = "No available contacts";
                    break;
                case Aws::ACM::Model::FailureReason::OTHER:
                    reason = "Other";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_ACCESS_DENIED:
```

```
                    reason = "PCA access denied";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_ARGS:
                    reason = "PCA invalid args";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_ARN:
                    reason = "PCA invalid ARN";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_DURATION:
                    reason = "PCA invalid duration";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_STATE:
                    reason = "PCA invalid state";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_LIMIT_EXCEEDED:
                    reason = "PCA limit exceeded";
                    break;
                case
Aws::ACM::Model::FailureReason::PCA_NAME_CONSTRAINTS_VALIDATION:
                    reason = "PCA name constraints validation";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_REQUEST_FAILED:
                    reason = "PCA request failed";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_RESOURCE_NOT_FOUND:
                    reason = "PCA resource not found";
                    break;
                default:
                    reason = "Cannot determine";
            }

            std::cout << "Failure reason:      " << reason << std::endl;
        }

        if (certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::REVOKED)
{
            std::cout << "Revoked at (GMT):    " <<
                      certificate.GetRevokedAt().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;

            Aws::ACM::Model::RevocationReason revocationReason =
                    certificate.GetRevocationReason();
            Aws::String reason;
```

```
        switch (revocationReason) {
            case Aws::ACM::Model::RevocationReason::AFFILIATION_CHANGED:
                reason = "Affiliation changed";
                break;
            case Aws::ACM::Model::RevocationReason::A_A_COMPROMISE:
                reason = "AA compromise";
                break;
            case Aws::ACM::Model::RevocationReason::CA_COMPROMISE:
                reason = "CA compromise";
                break;
            case Aws::ACM::Model::RevocationReason::CERTIFICATE_HOLD:
                reason = "Certificate hold";
                break;
            case Aws::ACM::Model::RevocationReason::CESSATION_OF_OPERATION:
                reason = "Cessation of operation";
                break;
            case Aws::ACM::Model::RevocationReason::KEY_COMPROMISE:
                reason = "Key compromise";
                break;
            case Aws::ACM::Model::RevocationReason::NOT_SET:
                reason = "Not set";
                break;
            case Aws::ACM::Model::RevocationReason::PRIVILEGE_WITHDRAWN:
                reason = "Privilege withdrawn";
                break;
            case Aws::ACM::Model::RevocationReason::REMOVE_FROM_CRL:
                reason = "Revoke from CRL";
                break;
            case Aws::ACM::Model::RevocationReason::SUPERCEDED:
                reason = "Superceded";
                break;
            case Aws::ACM::Model::RevocationReason::UNSPECIFIED:
                reason = "Unspecified";
                break;
            default:
                reason = "Cannot determine";
        }

        std::cout << "Revocation reason:   " << reason << std::endl;
    }

    if (certificate.GetType() == Aws::ACM::Model::CertificateType::IMPORTED) {
        std::cout << "Imported at (GMT):   " <<
```

```
                    certificate.GetImportedAt().ToGmtString(
                            Aws::Utils::DateFormat::ISO_8601)
                    << std::endl;
        }

        Aws::Vector<Aws::String> inUseBys = certificate.GetInUseBy();

        if (!inUseBys.empty()) {
            std::cout << std::endl << "In use by:" << std::endl << std::endl;

            for (auto &in_use_by: inUseBys) {
                std::cout << "  " << in_use_by << std::endl;
            }

            std::cout << std::endl;
        }

        if (certificate.GetType() == Aws::ACM::Model::CertificateType::AMAZON_ISSUED
&&
            certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::ISSUED) {
            std::cout << "Issued at (GMT):     " <<
                    certificate.GetIssuedAt().ToGmtString(
                            Aws::Utils::DateFormat::ISO_8601)
                    << std::endl;
        }

        std::cout << "Issuer:              " << certificate.GetIssuer() <<
                  std::endl;

        Aws::ACM::Model::KeyAlgorithm keyAlgorithm =
                certificate.GetKeyAlgorithm();
        Aws::String algorithm;

        switch (keyAlgorithm) {
            case Aws::ACM::Model::KeyAlgorithm::EC_prime256v1:
                algorithm = "P-256 (secp256r1, prime256v1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::EC_secp384r1:
                algorithm = "P-384 (secp384r1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::EC_secp521r1:
                algorithm = "P-521 (secp521r1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::NOT_SET:
```

```
                algorithm = "Not set";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_1024:
                algorithm = "RSA 1024";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_2048:
                algorithm = "RSA 2048";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_4096:
                algorithm = "RSA 4096";
                break;
            default:
                algorithm = "Cannot determine";
        }

        std::cout << "Key algorithm:       " << algorithm << std::endl;

        if (certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::ISSUED) {
            std::cout << "Not valid after (GMT): " <<
                      certificate.GetNotAfter().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
            std::cout << "Not valid before (GMT): " <<
                      certificate.GetNotBefore().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
        }

        Aws::ACM::Model::CertificateTransparencyLoggingPreference loggingPreference
=

certificate.GetOptions().GetCertificateTransparencyLoggingPreference();
        Aws::String preference;

        switch (loggingPreference) {
            case
Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED:
                preference = "Disabled";
                break;
            case Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED:
                preference = "Enabled";
                break;
            case Aws::ACM::Model::CertificateTransparencyLoggingPreference::NOT_SET:
                preference = "Not set";
```

```
                break;
            default:
                preference = "Cannot determine";
        }

        std::cout << "Logging preference:  " << preference << std::endl;

        std::cout << "Serial:              " << certificate.GetSerial() <<
                  std::endl;
        std::cout << "Signature algorithm: "
                  << certificate.GetSignatureAlgorithm() << std::endl;
        std::cout << "Subject:             " << certificate.GetSubject() <<
                  std::endl;

        Aws::ACM::Model::CertificateType certificateType = certificate.GetType();
        Aws::String type;

        switch (certificateType) {
            case Aws::ACM::Model::CertificateType::AMAZON_ISSUED:
                type = "Amazon issued";
                break;
            case Aws::ACM::Model::CertificateType::IMPORTED:
                type = "Imported";
                break;
            case Aws::ACM::Model::CertificateType::NOT_SET:
                type = "Not set";
                break;
            case Aws::ACM::Model::CertificateType::PRIVATE_:
                type = "Private";
                break;
            default:
                type = "Cannot determine";
        }

        std::cout << "Type:                " << type << std::endl;

        Aws::Vector<Aws::String> altNames =
                certificate.GetSubjectAlternativeNames();

        if (!altNames.empty()) {
            std::cout << std::endl << "Alternative names:" <<
                      std::endl << std::endl;

            for (auto &alt_name: altNames) {
```

```
                std::cout << "  " << alt_name << std::endl;
            }

            std::cout << std::endl;
        }
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeCertificate em Referência AWS SDK para C++ da API.

`ExportCertificate`

O código de exemplo a seguir mostra como usar `ExportCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Export an AWS Certificate Manager (ACM)  certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param passphrase: A passphrase to decrypt the exported certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::exportCertificate(const Aws::String &certificateArn,
                                    const Aws::String &passphrase,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ExportCertificateRequest request;
    Aws::Utils::CryptoBuffer cryptoBuffer(
```

```
            reinterpret_cast<const unsigned char *>(passphrase.c_str()),
            passphrase.length());
    request.WithCertificateArn(certificateArn).WithPassphrase(cryptoBuffer);

    Aws::ACM::Model::ExportCertificateOutcome outcome =
            acm_client.ExportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ExportCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate:       " << std::endl << std::endl <<
                  result.GetCertificate() << std::endl << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
                  result.GetCertificateChain() << std::endl << std::endl;
        std::cout << "Private key:       " << std::endl << std::endl <<
                  result.GetPrivateKey() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ExportCertificate em Referência AWS SDK para C++ da API.

`GetCertificate`

O código de exemplo a seguir mostra como usar `GetCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::getCertificate(const Aws::String &certificateArn,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::GetCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::GetCertificateOutcome outcome =
            acmClient.GetCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: GetCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate: " << std::endl << std::endl <<
                  result.GetCertificate() << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
                  result.GetCertificateChain() << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte GetCertificate em Referência AWS SDK para C++ da API.

`ImportCertificate`

O código de exemplo a seguir mostra como usar `ImportCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Import an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateFile: Path to certificate to import.
  \param privateKeyFile: Path to file containing a private key.
  \param certificateChainFile: Path to file containing a PEM encoded certificate
 chain.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::importCertificate(const Aws::String &certificateFile,
                                    const Aws::String &privateKeyFile,
                                    const Aws::String &certificateChainFile,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    std::ifstream certificateInStream(certificateFile.c_str());
    if (!certificateInStream) {
        std::cerr << "Error: The certificate file '" << certificateFile <<
                  "' does not exist." << std::endl;

        return false;
    }

    std::ifstream privateKeyInstream(privateKeyFile.c_str());
```

```
    if (!privateKeyInstream) {
        std::cerr << "Error: The private key file '" << privateKeyFile <<
                  "' does not exist." << std::endl;

        return false;
    }

    std::ifstream certificateChainInStream(certificateChainFile.c_str());
    if (!certificateChainInStream) {
        std::cerr << "Error: The certificate chain file '"
                  << certificateChainFile << "' does not exist." << std::endl;

        return false;
    }

    Aws::String certificate;
    certificate.assign(std::istreambuf_iterator<char>(certificateInStream),
                       std::istreambuf_iterator<char>());

    Aws::String privateKey;
    privateKey.assign(std::istreambuf_iterator<char>(privateKeyInstream),
                      std::istreambuf_iterator<char>());

    Aws::String certificateChain;

certificateChain.assign(std::istreambuf_iterator<char>(certificateChainInStream),
                            std::istreambuf_iterator<char>());

    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ImportCertificateRequest request;

    request.WithCertificate(Aws::Utils::ByteBuffer((unsigned char *)
                                                           certificate.c_str(),
                                                   certificate.size()))
            .WithPrivateKey(Aws::Utils::ByteBuffer((unsigned char *)
                                                           privateKey.c_str(),
                                                   privateKey.size()))
            .WithCertificateChain(Aws::Utils::ByteBuffer((unsigned char *)

certificateChain.c_str(),
                                                         certificateChain.size()));

    Aws::ACM::Model::ImportCertificateOutcome outcome =
```

```
            acmClient.ImportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ImportCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Certificate associated with ARN '" <<
                  outcome.GetResult().GetCertificateArn() << "' imported."
                  << std::endl;

        return true;
    }
}
```

- Para obter detalhes da API, consulte ImportCertificate em Referência AWS SDK para C++ da API.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the AWS Certificate Manager (ACM) certificates in an account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);
```

```
    Aws::ACM::Model::ListCertificatesRequest request;
    Aws::Vector<Aws::ACM::Model::CertificateSummary> allCertificates;
    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::ACM::Model::ListCertificatesOutcome outcome =
                acmClient.ListCertificates(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListCertificates: " <<
                      outcome.GetError().GetMessage() << std::endl;

            return false;
        }
        else {
            const Aws::ACM::Model::ListCertificatesResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::ACM::Model::CertificateSummary> &certificates =
                    result.GetCertificateSummaryList();
            allCertificates.insert(allCertificates.end(), certificates.begin(),
                                   certificates.end());

            nextToken = result.GetNextToken();
        }
    } while (!nextToken.empty());

    if (!allCertificates.empty()) {
        for (const Aws::ACM::Model::CertificateSummary &certificate:
allCertificates) {
            std::cout << "Certificate ARN: " <<
                      certificate.GetCertificateArn() << std::endl;
            std::cout << "Domain name:     " <<
                      certificate.GetDomainName() << std::endl << std::endl;
        }
    }
    else {
        std::cout << "No available certificates found in account."
                  << std::endl;
    }
```

```
    return true;
}
```

- Para obter detalhes da API, consulte ListCertificates em Referência AWS SDK para C++ da API.

ListTagsForCertificate

O código de exemplo a seguir mostra como usar `ListTagsForCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the tags for an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listTagsForCertificate(const Aws::String &certificateArn,
                                         const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ListTagsForCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::ListTagsForCertificateOutcome outcome =
            acm_client.ListTagsForCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cout << "Error: ListTagsForCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
        return false;
    }
    else {
        std::cout << "Success: Information about tags for "
                     "certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        Aws::Vector<Aws::ACM::Model::Tag> tags =
                result.GetTags();

        if (tags.size() > 0) {
            for (const Aws::ACM::Model::Tag &tag: tags) {
                std::cout << "Key:   " << tag.GetKey() << std::endl;
                std::cout << "Value: " << tag.GetValue()
                          << std::endl << std::endl;
            }
        }
        else {
            std::cout << "No tags found." << std::endl;
        }

        return true;
    }
}
```

- Para obter detalhes da API, consulte ListTagsForCertificate em Referência AWS SDK para C++ da API.

`RemoveTagsFromCertificate`

O código de exemplo a seguir mostra como usar `RemoveTagsFromCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Remove a tag from an ACM certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param tagKey: The key for the tag.
  \param tagValue: The value for the tag.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::removeTagsFromCertificate(const Aws::String &certificateArn,
                                            const Aws::String &tagKey,
                                            const Aws::String &tagValue,
                                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::Vector<Aws::ACM::Model::Tag> tags;

    Aws::ACM::Model::Tag tag;
    tag.SetKey(tagKey);

    tags.push_back(tag);

    Aws::ACM::Model::RemoveTagsFromCertificateRequest request;
    request.WithCertificateArn(certificateArn)
            .WithTags(tags);

    Aws::ACM::Model::RemoveTagsFromCertificateOutcome outcome =
            acmClient.RemoveTagsFromCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RemoveTagFromCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Tag with key '" << tagKey << "' removed from "
                  << "certificate with ARN '" << certificateArn << "'." <<
 std::endl;

        return true;
    }
}
```

- Para obter detalhes da API, consulte RemoveTagsFromCertificate em Referência AWS SDK para C++ da API.

`RenewCertificate`

O código de exemplo a seguir mostra como usar `RenewCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Renew an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::renewCertificate(const Aws::String &certificateArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RenewCertificateRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::RenewCertificateOutcome outcome =
            acmClient.RenewCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RenewCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
```

```
        std::cout << "Success: Renewed certificate with ARN '"
                  << certificateArn << "'." << std::endl;

        return true;
    }
}
```

- Para obter detalhes da API, consulte RenewCertificate em Referência AWS SDK para C++ da API.

`RequestCertificate`

O código de exemplo a seguir mostra como usar `RequestCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Request an AWS Certificate Manager (ACM) certificate.
/*!
  \param domainName: A fully qualified domain name.
  \param idempotencyToken: Customer chosen string for idempotency.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::requestCertificate(const Aws::String &domainName,
                                     const Aws::String &idempotencyToken,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RequestCertificateRequest request;
    request.WithDomainName(domainName)
            .WithIdempotencyToken(idempotencyToken);

    Aws::ACM::Model::RequestCertificateOutcome outcome =
```

```
            acmClient.RequestCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "RequestCertificate error: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The newly requested certificate's "
                     "ARN is '" <<
                  outcome.GetResult().GetCertificateArn() <<
                  "'." << std::endl;

        return true;
    }
}
```

- Para obter detalhes da API, consulte RequestCertificate em Referência AWS SDK para C++ da API.

`ResendValidationEmail`

O código de exemplo a seguir mostra como usar `ResendValidationEmail`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Resend the email that requests domain ownership validation.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param domainName: A fully qualified domain name.
  \param validationDomain: The base validation domain that will act as the suffix
                           of the email addresses.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::resendValidationEmail(const Aws::String &certificateArn,
                                        const Aws::String &domainName,
                                        const Aws::String &validationDomain,
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ResendValidationEmailRequest request;
    request.WithCertificateArn(certificateArn)
            .WithDomain(domainName)
            .WithValidationDomain(validationDomain);

    Aws::ACM::Model::ResendValidationEmailOutcome outcome =
            acmClient.ResendValidationEmail(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "ResendValidationEmail error: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The validation email has been resent."
                  << std::endl;

        return true;
    }
}
```

- Para obter detalhes da API, consulte ResendValidationEmail em Referência AWS SDK para C ++ da API.

UpdateCertificateOptions

O código de exemplo a seguir mostra como usar `UpdateCertificateOptions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update an AWS Certificate Manager (ACM) certificate option.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param loggingEnabled: Boolean specifying logging enabled.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::updateCertificateOption(const Aws::String &certificateArn,
                                          bool loggingEnabled,
                                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::UpdateCertificateOptionsRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::CertificateOptions options;

    if (loggingEnabled) {
        options.SetCertificateTransparencyLoggingPreference(
                Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED);
    }
    else {
        options.SetCertificateTransparencyLoggingPreference(

 Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED);
    }

    request.SetOptions(options);

    Aws::ACM::Model::UpdateCertificateOptionsOutcome outcome =
            acmClient.UpdateCertificateOptions(request);

    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "UpdateCertificateOption error: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The option '"
                  << (loggingEnabled ? "enabled" : "disabled") << "' has been set
 for "
                                                                  "the certificate
 with the ARN '"
                  << certificateArn << "'."
                  << std::endl;

        return true;
    }
}
```

- Para obter detalhes da API, consulte UpdateCertificateOptions em Referência AWS SDK para C++ da API.

Exemplos da API Gateway usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com API Gateway.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos do Aurora usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Aurora.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Aurora

O exemplo de código a seguir mostra como começar a usar o Aurora.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
```

```
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());
```

```
        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para C++ .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de cluster do banco de dados do Aurora e definir os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Criar um snapshot do cluster do banco de dados e limpar os recursos.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
```

```
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
 std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " << DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family << std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB cluster parameter group.
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully created."
```

```
                    << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter group."
              << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB cluster parameter group.
    if (!getDBCLusterParameters(CLUSTER_PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX,
                                NO_SOURCE,
                                autoIncrementParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value between ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
```

```
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a source
of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the DB cluster parameter group.
    if (!getDBCLusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                                userParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
```

```
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
    else {
        std::cout << "Let's create a DB cluster." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters): ");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of engine versions for the parameter group family.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
                                 client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group family are:"
                  << std::endl;
```

```
        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
{
            std::cout << "  " << index << ": " << engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                                1];

        engineName = engineVersion.GetEngine();
        engineVersionName = engineVersion.GetEngineVersion();
        std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB cluster is configured to use your custom cluster
parameter group '"
                  << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
                  << "selected engine version " << engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB cluster to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB cluster to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for cluster to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
```

```
                        client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
                  << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql' shell
to the database.
    displayConnection(dbCluster);
```

```
    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
        Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for the snapshot to become available." << std::endl;

        Aws::RDS::Model::DBClusterSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
```

```
                        << counter
                        << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                        << outcome.GetError().GetMessage()
                        << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                        << snapshot.GetStatus()
                        << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
```

```
            "Do you want to delete the DB cluster, DB instance, and parameter group
 (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }
```

```
    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,
                                            Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
```

```
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);
```

```
    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
```

```
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
 {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
```

```
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
 parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);
```

```
            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " << counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
```

```
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." << std::endl;
            }

            if (clusterDeleting) {
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para C++ .

`CreateDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte CreateDBClusterParameterGroup em Referência AWS SDK para C++ da API.

`CreateDBClusterSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBClusterSnapshot`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Para obter detalhes da API, consulte CreateDBClusterSnapshot em Referência AWS SDK para C++ da API.

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para C++ .

`DeleteDBCluster`

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para C++ .

`DeleteDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBClusterParameterGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(parameterGroupName);

    Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
            client.DeleteDBClusterParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully deleted."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
```

- Para obter detalhes da API, consulte DeleteDBClusterParameterGroup em Referência AWS SDK para C++ da API.

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para C++ .

`DescribeDBClusterParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameterGroups`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
std::endl;
            dbParameterGroupFamily =
outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte DescribeDBClusterParameterGroups em Referência AWS SDK para C++ da API.

`DescribeDBClusterParameters`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameters`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,
                                            Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Para obter detalhes da API, consulte DescribeDBClusterParameters em Referência AWS SDK para C++ da API.

`DescribeDBClusterSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBClusterSnapshots`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Para obter detalhes da API, consulte DescribeDBClusterSnapshots em Referência AWS SDK para C++ da API.

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
```

```
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para C++ .

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
```

```
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions em Referência AWS SDK para C++ da API.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);
```

```
    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para C ++ .

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);
```

```
//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
 {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptions em Referência AWS SDK para C++ da API.

`ModifyDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);
```

```
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte ModifyDBClusterParameterGroup em Referência AWS SDK para C++ da API.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar uma API REST C++ que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora

- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Auto Scaling usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com Auto Scaling.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Auto Scaling

O exemplo de código a seguir mostra como começar a usar o Auto Scaling.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)
```

```
target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_autoscalling.cpp.

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2 Auto
 Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
```

```
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
 &autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
 std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK para C++ da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo do Amazon EC2 Auto Scaling com um modelo de inicialização e zonas de disponibilidade e obter informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas para CloudWatch métricas e, em seguida, limpe os recursos.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
  \sa groupsAndInstancesScenario()
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout
```

```
            << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
Scaling "
            << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
              << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
            "Would you like to use an existing EC2 launch template (y/n)?  ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
                "Enter the name of the existing EC2 launch template.  ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateNames(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
                ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                      << "' exists by calling DescribeLaunchTemplate." << std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    else { // 2.  Or create a new EC2 launch template.
        templateName = askQuestion("Enter the name for a new EC2 launch template:
");

        Aws::EC2::Model::CreateLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
        requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

        request.SetLaunchTemplateData(requestLaunchTemplateData);

        Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
```

```
                ec2Client.CreateLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << " was
created."
                      << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
                 "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already exists"
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
            "Enter a name for the Auto Scaling group:  ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
                ec2Client.DescribeAvailabilityZones(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "EC2 instances can be created in the following Availability
Zones:"
                    << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "   " << i + 1 << ".  "
                          << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    int availabilityZoneChoice = askQuestionForIntRange(
            "Choose an Availability Zone:  ", 1,
            static_cast<int>(availabilityZones.size()));
    // 4. Create an Auto Scaling group with the specified Availability Zone.
    {
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
```

```
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        std::cout << "Here is the Auto Scaling group description." << std::endl;
        if (!autoScalingGroups.empty()) {
            logAutoScalingGroupInfo(autoScalingGroups);
        }
    }
    else {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    std::cout
            << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
            << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    bool enableMetrics = askYesNoQuestion(
            "Do you want to collect metrics about the A"
            "Auto Scaling group during this demo (y/n)?  ");
    // 7. Optionally enable metrics collection for the Auto Scaling group.
    if (enableMetrics) {
        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
```

```
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
              "' from 1 to 3." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 8. Update the Auto Scaling group, setting a new maximum size.
    {
        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            const auto &instances = autoScalingGroups[0].GetInstances();
            std::cout
                    << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                    << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
```

```
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
              << std::endl;
    std::cout << "Let's update the desired capacity in '" << groupName <<
              "' from 1 to 2." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    //  9. Update the Auto Scaling group, setting a new desired capacity.
    {
        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            std::cout
                    << "Here is the current state of the group." << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
```

```
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
              << std::endl;

    std::cout << "Let's terminate one of the EC2 instances in " << groupName << "."
              << std::endl;
    std::cout << "Because the desired capacity is 2, another EC2 instance will start
"
              << "to replace the terminated EC2 instance."
              << std::endl;
    std::cout << "The currently running EC2 instances are:" << std::endl;

    if (autoScalingGroups.empty()) {
        std::cerr << "Error describing groups. No groups returned." << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    int instanceNumber = 1;
    Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
            autoScalingGroups[0].GetInstances());
    for (const Aws::String &instanceID: instanceIDs) {
        std::cout << "   " << instanceNumber << ". " << instanceID << std::endl;
        ++instanceNumber;
    }

    instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to stop?
",
                                            1,
                                            static_cast<int>(instanceIDs.size()));

    // 10. Terminate an EC2 instance in the Auto Scaling group.
    {
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);
```

```
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
=
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
              << std::endl;
    std::cout << "Let's get a report of scaling activities for EC2 launch group '"
              << groupName << "'."
              << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 11. Get a description of activities for the Auto Scaling group.
    {
        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
                        outcome.GetResult().GetActivities();
```

```
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
    }

    if (enableMetrics) {
        if (!logAutoScalingMetrics(groupName, clientConfig)) {
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    std::cout << "Let's  clean up." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);

    // 13. Disable metrics collection if enabled.
    if (enableMetrics) {
        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
```

```
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    return cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                            const Aws::AutoScaling::AutoScalingClient
 &client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
```

```
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
                break;
            }
            else {
                if ((count % 5) == 0) {
                    std::cout << "No instance IDs returned for group." << std::endl;
                }

                continue;
            }
        }

        // 6.  Check lifecycle state of the instances using
DescribeAutoScalingInstances.
        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();
            ready = instancesDetails.size() >= desiredCapacity;
            for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails &details:
instancesDetails) {
                if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                    ready = false;
                    break;
                }
            }
            // Log the status while waiting.
            if (((count % 5) == 1) || ready) {
                logInstancesLifecycleState(instancesDetails);
            }
```

```
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
\return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const Aws::AutoScaling::AutoScalingClient
 &autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client) {
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
        (askYesNoQuestion(
                Aws::String("Delete the Auto Scaling group '") + groupName +
                "'  (y/n)?"))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
```

```
                    autoScalingClient.UpdateAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout
                        << "The minimum size and desired capacity of the Auto
Scaling group "
                        << "was set to zero before terminating the instances."
                        << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                          << outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
        }

        Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
        if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                               autoScalingClient)) {
            if (!autoScalingGroups.empty()) {
                Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                        autoScalingGroups[0].GetInstances());
                for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                    request.SetInstanceId(instanceID);
                    request.SetShouldDecrementDesiredCapacity(true);


Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =
                            autoScalingClient.TerminateInstanceInAutoScalingGroup(
                                    request);

                    if (outcome.IsSuccess()) {
                        std::cout << "Initiating termination of EC2 instance '"
                                  << instanceID << "'." << std::endl;
                    }
                    else {
                        std::cerr
                                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                                << outcome.GetError().GetMessage() << std::endl;
                        result = false;
                    }
```

```
                }
            }

            std::cout
                    << "Waiting for the EC2 instances to terminate before deleting
the "
                    << "Auto Scaling group..." << std::endl;
            waitForInstances(groupName, autoScalingGroups, autoScalingClient);
        }

        {
            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    // 15. Delete the EC2 launch template.
    if (!templateName.empty() && (askYesNoQuestion(
            Aws::String("Delete the EC2 launch template '") + templateName +
            "' (y/n)?"))) {
        Aws::EC2::Model::DeleteLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
                ec2Client.DeleteLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "EC2 launch template '" << templateName << "' was deleted."
                      << std::endl;
        }
```

```
        else {
            std::cerr << "Error with EC2::DeleteLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
                                         const Aws::AutoScaling::AutoScalingClient
 &client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
            client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Ações

`CreateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `CreateAutoScalingGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
```

```
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Para obter detalhes da API, consulte CreateAutoScalingGroup em Referência AWS SDK para C++ da API.

`DeleteAutoScalingGroup`

O código de exemplo a seguir mostra como usar `DeleteAutoScalingGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                autoScalingClient.DeleteAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling group '" << groupName << "' was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroup em Referência AWS SDK para C++ da API.

`DescribeAutoScalingGroups`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingGroups`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK para C++ da API.

`DescribeAutoScalingInstances`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();

        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstances em Referência AWS SDK para C++ da API.

`DescribeScalingActivities`

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
```

```
                << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
```

- Para obter detalhes da API, consulte DescribeScalingActivities em Referência AWS SDK para C++ da API.

`DisableMetricsCollection`

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;

    }
```

- Para obter detalhes da API, consulte DisableMetricsCollection em Referência AWS SDK para C++ da API.

`EnableMetricsCollection`

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    request.AddMetrics("GroupMinSize");
    request.AddMetrics("GroupMaxSize");
    request.AddMetrics("GroupDesiredCapacity");
    request.AddMetrics("GroupInServiceInstances");
    request.AddMetrics("GroupTotalInstances");
    request.SetGranularity("1Minute");

    Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
            autoScalingClient.EnableMetricsCollection(request);
    if (outcome.IsSuccess()) {
        std::cout << "Auto Scaling metrics have been enabled."
```

```
                    << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Para obter detalhes da API, consulte EnableMetricsCollection em Referência AWS SDK para C++ da API.

SetDesiredCapacity

O código de exemplo a seguir mostra como usar `SetDesiredCapacity`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
```

```
                    << std::endl;

        }
```

- Para obter detalhes da API, consulte SetDesiredCapacity em Referência AWS SDK para C++ da API.

`TerminateInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
=
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
```

```
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroup em Referência AWS SDK para C++ da API.

`UpdateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `UpdateAutoScalingGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
        }
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroup em Referência AWS SDK para C++ da API.

CloudTrail exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with CloudTrail.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateTrail`

O código de exemplo a seguir mostra como usar `CreateTrail`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Routine which creates an AWS CloudTrail trail.
/*!
  \param trailName: The name of the CloudTrail trail.
  \param bucketName: The Amazon S3 bucket designate for publishing logs.
```

```
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::createTrail(const Aws::String trailName,
                                     const Aws::String bucketName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);
    Aws::CloudTrail::Model::CreateTrailRequest request;
    request.SetName(trailName);
    request.SetS3BucketName(bucketName);

    Aws::CloudTrail::Model::CreateTrailOutcome outcome = trailClient.CreateTrail(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Failed to create trail " << trailName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTrail em Referência AWS SDK para C++ da API.

`DeleteTrail`

O código de exemplo a seguir mostra como usar `DeleteTrail`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Routine which deletes an AWS CloudTrail trail.
/*!
```

```
  \param trailName: The name of the CloudTrail trail.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::deleteTrail(const Aws::String trailName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);

    Aws::CloudTrail::Model::DeleteTrailRequest request;
    request.SetName(trailName);

    auto outcome = trailClient.DeleteTrail(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Error deleting trail " << trailName << " " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTrail em Referência AWS SDK para C++ da API.

`DescribeTrail`

O código de exemplo a seguir mostra como usar `DescribeTrail`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Routine which describes the AWS CloudTrail trails in an account.
/*!
  \param clientConfig: Aws client configuration.
```

```
  \return bool: Function succeeded.
*/

bool AwsDoc::CloudTrail::describeTrails(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudTrailClient(clientConfig);
    Aws::CloudTrail::Model::DescribeTrailsRequest request;

    auto outcome = cloudTrailClient.DescribeTrails(request);
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::CloudTrail::Model::Trail> &trails =
 outcome.GetResult().GetTrailList();
        std::cout << trails.size() << " trail(s) found." << std::endl;
        for (const Aws::CloudTrail::Model::Trail &trail: trails) {
            std::cout << trail.GetName() << std::endl;
        }
    }
    else {
        std::cerr << "Failed to describe trails." << outcome.GetError().GetMessage()
                  << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeTrail em Referência AWS SDK para C++ da API.

`LookupEvents`

O código de exemplo a seguir mostra como usar `LookupEvents`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Routine which looks up events captured by AWS CloudTrail.
```

```
/*!
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::lookupEvents(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudtrail(clientConfig);

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::CloudTrail::Model::Event> allEvents;

    Aws::CloudTrail::Model::LookupEventsRequest request;

    size_t count = 0;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::CloudTrail::Model::LookupEventsOutcome outcome =
 cloudtrail.LookupEvents(
                request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::CloudTrail::Model::Event> &events =
 outcome.GetResult().GetEvents();
            count += events.size();
            allEvents.insert(allEvents.end(), events.begin(), events.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!nextToken.empty() && count <= 50); // Limit to 50 events.

    std::cout << "Found " << allEvents.size() << " event(s)." << std::endl;

    for (auto &event: allEvents) {
        std::cout << "Event name: " << event.GetEventName() << std::endl;
        std::cout << "Event source: " << event.GetEventSource() << std::endl;
        std::cout << "Event id: " << event.GetEventId() << std::endl;
        std::cout << "Resources: " << std::endl;
        for (auto &resource: event.GetResources()) {
            std::cout << "  " << resource.GetResourceName() << std::endl;
```

```
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte LookupEvents em Referência AWS SDK para C++ da API.

CloudWatch exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteAlarms`

O código de exemplo a seguir mostra como usar `DeleteAlarms`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

Excluir o alarme.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DeleteAlarmsRequest request;
        request.AddAlarmNames(alarm_name);

        auto outcome = cw.DeleteAlarms(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- Para obter detalhes da API, consulte DeleteAlarms em Referência AWS SDK para C++ da API.

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

Descreva os alarmes.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DescribeAlarmsRequest request;
        request.SetMaxRecords(1);

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.DescribeAlarms(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to describe CloudWatch alarms:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left <<
                    std::setw(32) << "Name" <<
                    std::setw(64) << "Arn" <<
                    std::setw(64) << "Description" <<
                    std::setw(20) << "LastUpdated" <<
                    std::endl;
                header = true;
            }

            const auto &alarms = outcome.GetResult().GetMetricAlarms();
            for (const auto &alarm : alarms)
            {
                std::cout << std::left <<
                    std::setw(32) << alarm.GetAlarmName() <<
                    std::setw(64) << alarm.GetAlarmArn() <<
```

```
                    std::setw(64) << alarm.GetAlarmDescription() <<
                    std::setw(20) <<
                    alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                        SIMPLE_DATE_FORMAT_STR) <<
                    std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetric em Referência AWS SDK para C++ da API.

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Desabilite as ações de alarme.

```
        Aws::CloudWatch::CloudWatchClient cw;
```

```
        Aws::CloudWatch::Model::DisableAlarmActionsRequest
disableAlarmActionsRequest;
        disableAlarmActionsRequest.AddAlarmNames(alarm_name);

        auto disableAlarmActionsOutcome =
cw.DisableAlarmActions(disableAlarmActionsRequest);
        if (!disableAlarmActionsOutcome.IsSuccess())
        {
            std::cout << "Failed to disable actions for alarm " << alarm_name <<
                ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully disabled actions for alarm " <<
                alarm_name << std::endl;
        }
```

- Para obter detalhes da API, consulte DisableAlarmActions em Referência AWS SDK para C++ da API.

`EnableAlarmActions`

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Habilite as ações de alarme.

```
    Aws::CloudWatch::CloudWatchClient cw;
    Aws::CloudWatch::Model::PutMetricAlarmRequest request;
    request.SetAlarmName(alarm_name);
    request.SetComparisonOperator(
        Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
    request.SetEvaluationPeriods(1);
    request.SetMetricName("CPUUtilization");
    request.SetNamespace("AWS/EC2");
    request.SetPeriod(60);
    request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
    request.SetThreshold(70.0);
    request.SetActionsEnabled(false);
    request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
    request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
    request.AddAlarmActions(actionArn);

    Aws::CloudWatch::Model::Dimension dimension;
    dimension.SetName("InstanceId");
    dimension.SetValue(instanceId);
    request.AddDimensions(dimension);

    auto outcome = cw.PutMetricAlarm(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create CloudWatch alarm:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
    enable_request.AddAlarmNames(alarm_name);

    auto enable_outcome = cw.EnableAlarmActions(enable_request);
    if (!enable_outcome.IsSuccess())
    {
        std::cout << "Failed to enable alarm actions:" <<
            enable_outcome.GetError().GetMessage() << std::endl;
        return;
    }
```

```
    std::cout << "Successfully created alarm " << alarm_name <<
        " and enabled actions on it." << std::endl;
```

- Para obter detalhes da API, consulte EnableAlarmActions em Referência AWS SDK para C++ da API.

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

Liste as métricas.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::ListMetricsRequest request;

        if (argc > 1)
        {
            request.SetMetricName(argv[1]);
        }

        if (argc > 2)
```

```
        {
            request.SetNamespace(argv[2]);
        }

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.ListMetrics(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to list CloudWatch metrics:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left << std::setw(48) << "MetricName" <<
                    std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
                    std::endl;
                header = true;
            }

            const auto &metrics = outcome.GetResult().GetMetrics();
            for (const auto &metric : metrics)
            {
                std::cout << std::left << std::setw(48) <<
                    metric.GetMetricName() << std::setw(32) <<
                    metric.GetNamespace();
                const auto &dimensions = metric.GetDimensions();
                for (auto iter = dimensions.cbegin();
                    iter != dimensions.cend(); ++iter)
                {
                    const auto &dimkv = *iter;
                    std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
                    if (iter + 1 != dimensions.cend())
                    {
                        std::cout << ", ";
                    }
                }
                std::cout << std::endl;
            }
```

```
            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Para obter detalhes da API, consulte ListMetrics em Referência AWS SDK para C++ da API.

PutMetricAlarm

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Crie o alarme para vigiar a métrica.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::PutMetricAlarmRequest request;
        request.SetAlarmName(alarm_name);
        request.SetComparisonOperator(
            Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
        request.SetEvaluationPeriods(1);
        request.SetMetricName("CPUUtilization");
        request.SetNamespace("AWS/EC2");
        request.SetPeriod(60);
        request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
        request.SetThreshold(70.0);
```

```
        request.SetActionsEnabled(false);
        request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
        request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("InstanceId");
        dimension.SetValue(instanceId);

        request.AddDimensions(dimension);

        auto outcome = cw.PutMetricAlarm(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- Para obter detalhes da API, consulte PutMetricAlarm em Referência AWS SDK para C++ da API.

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
```

```
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

Insira dados na métrica.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("UNIQUE_PAGES");
        dimension.SetValue("URLS");

        Aws::CloudWatch::Model::MetricDatum datum;
        datum.SetMetricName("PAGES_VISITED");
        datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
        datum.SetValue(data_point);
        datum.AddDimensions(dimension);

        Aws::CloudWatch::Model::PutMetricDataRequest request;
        request.SetNamespace("SITE/TRAFFIC");
        request.AddMetricData(datum);

        auto outcome = cw.PutMetricData(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to put sample metric data:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully put sample metric data" << std::endl;
        }
```

- Para obter detalhes da API, consulte PutMetricData em Referência AWS SDK para C++ da API.

CloudWatch Exemplos de registros usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with CloudWatch Logs.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteSubscriptionFilter`

O código de exemplo a seguir mostra como usar `DeleteSubscriptionFilter`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Excluir o filtro de assinatura.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetLogGroupName(log_group);
```

```
        auto outcome = cwl.DeleteSubscriptionFilter(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Failed to delete CloudWatch log subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        } else {
            std::cout << "Successfully deleted CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- Para obter detalhes da API, consulte DeleteSubscriptionFilter em Referência AWS SDK para C++ da API.

`DescribeSubscriptionFilters`

O código de exemplo a seguir mostra como usar `DescribeSubscriptionFilters`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

Liste os filtros de assinatura.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
```

```
        Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
        request.SetLogGroupName(log_group);
        request.SetLimit(1);

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = cwl.DescribeSubscriptionFilters(
                    request);
            if (!outcome.IsSuccess()) {
                std::cout << "Failed to describe CloudWatch subscription filters "
                    << "for log group " << log_group << ": " <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(32) << "Name" <<
                    std::setw(64) << "FilterPattern" << std::setw(64) <<
                    "DestinationArn" << std::endl;
                header = true;
            }

            const auto &filters = outcome.GetResult().GetSubscriptionFilters();
            for (const auto &filter : filters) {
                std::cout << std::left << std::setw(32) <<
                    filter.GetFilterName() << std::setw(64) <<
                    filter.GetFilterPattern() << std::setw(64) <<
                    filter.GetDestinationArn() << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Para obter detalhes da API, consulte DescribeSubscriptionFilters em Referência AWS SDK para C++ da API.

PutSubscriptionFilter

O código de exemplo a seguir mostra como usar `PutSubscriptionFilter`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Crie o filtro de assinatura.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetFilterPattern(filter_pattern);
        request.SetLogGroupName(log_group);
        request.SetDestinationArn(dest_arn);
        auto outcome = cwl.PutSubscriptionFilter(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch logs subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- Para obter detalhes da API, consulte PutSubscriptionFilter em Referência AWS SDK para C++ da API.

CodeBuild exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with CodeBuild.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListBuilds`

O código de exemplo a seguir mostra como usar `ListBuilds`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the CodeBuild builds.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
```

```
    listBuildsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Used for pagination.

    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
codeBuildClient.ListBuilds(
                listBuildsRequest);

        if (listBuildsOutcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &ids =
listBuildsOutcome.GetResult().GetIds();
            if (!ids.empty()) {

                std::cout << "Information about each build:" << std::endl;
                Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
                getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
                Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
codeBuildClient.BatchGetBuilds(
                        getBuildsRequest);

                if (getBuildsOutcome.IsSuccess()) {
                    const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
getBuildsOutcome.GetResult().GetBuilds();
                    std::cout << builds.size() << " build(s) found." << std::endl;
                    for (auto val: builds) {
                        std::cout << val.GetId() << std::endl;
                    }
                } else {
                    std::cerr << "Error getting builds"
                              << getBuildsOutcome.GetError().GetMessage() <<
std::endl;
                    return false;
                }
            } else {
                std::cout << "No builds found." << std::endl;
            }

            // Get the next token for pagination.
```

```
            nextToken = listBuildsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error listing builds"
                      << listBuildsOutcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!nextToken.

            empty()

            );

    return true;
}
```

- Para obter detalhes da API, consulte ListBuilds em Referência AWS SDK para C++ da API.

`ListProjects`

O código de exemplo a seguir mostra como usar `ListProjects`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the CodeBuild projects.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType sortType,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Next token for pagination.
    Aws::Vector<Aws::String> allProjects;

    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
 codeBuildClient.ListProjects(
                listProjectsRequest);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
 outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(), projects.end());
            nextToken = outcome.GetResult().GetNextToken();
        }

        else {
            std::cerr << "Error listing projects" << outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (!nextToken.empty());

    std::cout << allProjects.size() << " project(s) found." << std::endl;
    for (auto project: allProjects) {
        std::cout << project << std::endl;
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListProjects em Referência AWS SDK para C++ da API.

StartBuild

O código de exemplo a seguir mostra como usar `StartBuild`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Start an AWS CodeBuild project build.
/*!
  \param projectName: A CodeBuild project name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome = codeBuildClient.StartBuild(
            startBuildRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
                  << std::endl;
    }

    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte StartBuild em Referência AWS SDK para C++ da API.

Exemplos do Provedor de Identidade do Amazon Cognito usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon Cognito

O exemplo de código a seguir mostra como começar a usar o Amazon Cognito.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

# Set this project's name.
project("hello_cognito")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)
```

```
target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_cognito.cpp.

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito client
 and lists the Amazon Cognito
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
 listUserPoolsRequest;
            if (!nextToken.empty()) {
```

```
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
listUserPoolsOutcome.GetResult().GetUserPools()) {

                    userPools.push_back(userPool.GetName());
                }

                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "ListUserPools error: " <<
listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListUserPools em Referência AWS SDK para C++ da API.

Ações

`AdminGetUser`

O código de exemplo a seguir mostra como usar `AdminGetUser`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                        outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() << std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Para obter detalhes da API, consulte AdminGetUser em Referência AWS SDK para C++ da API.

`AdminInitiateAuth`

O código de exemplo a seguir mostra como usar `AdminInitiateAuth`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Para obter detalhes da API, consulte AdminInitiateAuth em Referência AWS SDK para C++ da API.

`AdminRespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `AdminRespondToAuthChallenge`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
        request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;
```

```
            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallenge em Referência AWS SDK para C++ da API.

`AssociateSoftwareToken`

O código de exemplo a seguir mostra como usar `AssociateSoftwareToken`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome outcome =
                client.AssociateSoftwareToken(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for example
 Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte AssociateSoftwareToken em Referência AWS SDK para C++ da API.

`ConfirmSignUp`

O código de exemplo a seguir mostra como usar `ConfirmSignUp`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte ConfirmSignUp em Referência AWS SDK para C++ da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para C++ da API.

`ResendConfirmationCode`

O código de exemplo a seguir mostra como usar `ResendConfirmationCode`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

       Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest request;
       request.SetUsername(userName);
       request.SetClientId(clientID);

       Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome outcome =
               client.ResendConfirmationCode(request);

       if (outcome.IsSuccess()) {
           std::cout
                   << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                   << std::endl;
       }
       else {
           std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
```

- Para obter detalhes da API, consulte ResendConfirmationCode em Referência AWS SDK para C++ da API.

SignUp

O código de exemplo a seguir mostra como usar `SignUp`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte SignUp em Referência AWS SDK para C++ da API.

`VerifySoftwareToken`

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte VerifySoftwareToken em Referência AWS SDK para C++ da API.

Cenários

Inscrever um usuário em um grupo de usuários que exija MFA

O exemplo de código a seguir mostra como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
            << "Welcome to the Amazon Cognito example scenario."
            << std::endl;
    printAsterisksLine();

    std::cout
```

```
            << "This scenario will add a user to an Amazon Cognito user pool."
            << std::endl;
    const Aws::String userName = askQuestion("Enter a new username: ");
    const Aws::String password = askQuestion("Enter a new password: ");
    const Aws::String email = askQuestion("Enter a valid email for the user: ");

    std::cout << "Signing up " << userName << std::endl;

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
    bool userExists = false;
    do {
        // 1. Add a user with a username, password, and email address.
        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (userExists);
```

```
    printAsterisksLine();
    std::cout << "Retrieving status of " << userName << " in the user pool."
              << std::endl;
    // 2. Confirm that the user was added to the user pool.
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    std::cout << "A confirmation code was sent to " << email << "." << std::endl;

    bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
    if (resend) {
        // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();

    {
        // 4. Send the confirmation code that's received in the email.
(ConfirmSignUp)
        const Aws::String confirmationCode = askQuestion(
                "Enter the confirmation code that was emailed: ");
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
```

```
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
              << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout << "Initiating authorization using the username and password."
              << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
session, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-factor
authentication (MFA)."
            << std::endl;
```

```
    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for example
 Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
    askQuestion("Type enter to continue...", alwaysTrueTest);

    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");
```

```
        //  7. Send the MFA code copied from an authenticator app.
(VerifySoftwareToken)
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
        Aws::String mfaCode = askQuestion(
                "Re-enter the 6 digit code displayed in the authenticator app: ");

        // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        request.SetChallengeName(

 Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
 outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

 outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;

            accessToken =
 outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

        std::cout << "You have successfully added a user to Amazon Cognito."
                  << std::endl;
    }

    if (askYesNoQuestion("Would you like to delete the user that you just added? (y/
n) ")) {
        // 10. Delete the user that you just added. (DeleteUser)
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
```

```
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                        outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() << std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

```
//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

 Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Exemplos do DynamoDB usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS dynamodb)

# Set this project's name.
project("hello_dynamodb")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_dynamodb.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_dynamodb.cpp.

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ListTablesRequest.h>
#include <iostream>

/*
 *  A "Hello DynamoDB" starter application which initializes an Amazon DynamoDB
 (DynamoDB) client and lists the
 *  DynamoDB tables.
 *
 *  main function
 *
 *  Usage: 'hello_dynamodb'
 *
 */
```

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.

    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::DynamoDB::DynamoDBClient dynamodbClient(clientConfig);
        Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
        listTablesRequest.SetLimit(50);
        do {
            const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
 dynamodbClient.ListTables(
                    listTablesRequest);
            if (!outcome.IsSuccess()) {
                std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
                result = 1;
                break;
            }

            for (const auto &tableName: outcome.GetResult().GetTableNames()) {
                std::cout << tableName << std::endl;
            }

            listTablesRequest.SetExclusiveStartTableName(
                    outcome.GetResult().GetLastEvaluatedTableName());

        } while (!listTablesRequest.GetExclusiveStartTableName().empty());
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para C++ da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    {
        Aws::Client::ClientConfiguration clientConfig;
        //  1. Create a table with partition: year (N) and sort: title (S).
 (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::dynamodbGettingStartedScenario(clientConfig);

            // 9. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }
    }

//! Scenario to modify and query a DynamoDB table.
/*!
  \sa dynamodbGettingStartedScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout << "Welcome to the Amazon DynamoDB getting started demo." <<
 std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie.
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
 ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::PutItemRequest putItemRequest;
        putItemRequest.SetTableName(MOVIE_TABLE_NAME);

        putItemRequest.AddItem(YEAR_KEY,
                               Aws::DynamoDB::Model::AttributeValue().SetN(year));
        putItemRequest.AddItem(TITLE_KEY,
                               Aws::DynamoDB::Model::AttributeValue().SetS(title));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
 Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
 Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
```

```
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);

        putItemRequest.AddItem(INFO_KEY, infoMapAttribute);

        Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
                putItemRequest);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add an item: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
              << std::endl;

    // 3. Update the rating and plot of the movie by using an update expression.
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);
        plot = askQuestion(Aws::String("You summarized the plot as '") + plot +
                           "'.\nWhat would you say now? ");

        Aws::DynamoDB::Model::UpdateItemRequest request;
        request.SetTableName(MOVIE_TABLE_NAME);
        request.AddKey(TITLE_KEY,
Aws::DynamoDB::Model::AttributeValue().SetS(title));
        request.AddKey(YEAR_KEY, Aws::DynamoDB::Model::AttributeValue().SetN(year));
        std::stringstream expressionStream;
        expressionStream << "set " << INFO_KEY << "." << RATING_KEY << " =:r, "
                         << INFO_KEY << "." << PLOT_KEY << " =:p";
        request.SetUpdateExpression(expressionStream.str());
        request.SetExpressionAttributeValues({
                                                        {":r",
Aws::DynamoDB::Model::AttributeValue().SetN(
                                                                rating)},
                                                        {":p",
Aws::DynamoDB::Model::AttributeValue().SetS(
                                                                plot)}
```

```
                                                    });

        request.SetReturnValues(Aws::DynamoDB::Model::ReturnValue::UPDATED_NEW);

        const Aws::DynamoDB::Model::UpdateItemOutcome &result =
dynamoClient.UpdateItem(
                request);
        if (!result.IsSuccess()) {
            std::cerr << "Error updating movie " + result.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

    // 4. Put 250 movies in the table from moviedata.json.
    {
        std::cout << "Adding movies from a json file to the database." << std::endl;
        const size_t MAX_SIZE_FOR_BATCH_WRITE = 25;
        const size_t MOVIES_TO_WRITE = 10 * MAX_SIZE_FOR_BATCH_WRITE;
        Aws::String jsonString = getMovieJSON();
        if (!jsonString.empty()) {
            Aws::Utils::Json::JsonValue json(jsonString);
            Aws::Utils::Array<Aws::Utils::Json::JsonView> movieJsons =
json.View().AsArray();
            Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;

            // To add movies with a cross-section of years, use an appropriate
increment
            // value for iterating through the database.
            size_t increment = movieJsons.GetLength() / MOVIES_TO_WRITE;
            for (size_t i = 0; i < movieJsons.GetLength(); i += increment) {
                writeRequests.push_back(Aws::DynamoDB::Model::WriteRequest());
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> putItems
= movieJsonViewToAttributeMap(
                        movieJsons[i]);
                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(putItems);
                writeRequests.back().SetPutRequest(putRequest);
                if (writeRequests.size() == MAX_SIZE_FOR_BATCH_WRITE) {
                    Aws::DynamoDB::Model::BatchWriteItemRequest request;
                    request.AddRequestItems(MOVIE_TABLE_NAME, writeRequests);
```

```
                    const Aws::DynamoDB::Model::BatchWriteItemOutcome &outcome =
dynamoClient.BatchWriteItem(
                            request);
                    if (!outcome.IsSuccess()) {
                        std::cerr << "Unable to batch write movie data: "
                                  << outcome.GetError().GetMessage()
                                  << std::endl;
                        writeRequests.clear();
                        break;
                    }
                    else {
                        std::cout << "Added batch of " << writeRequests.size()
                                  << " movies to the database."
                                  << std::endl;
                    }
                    writeRequests.clear();
                }
            }
        }
    }

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;

    // 5. Get a movie by Key (partition + sort).
    {
        Aws::String titleToGet("King Kong");
        Aws::String answer = askQuestion(Aws::String(
                "Let's move on...Would you like to get info about '" + titleToGet +
                "'? (y/n) "));
        if (answer == "y") {
            Aws::DynamoDB::Model::GetItemRequest request;
            request.SetTableName(MOVIE_TABLE_NAME);
            request.AddKey(TITLE_KEY,
                           Aws::DynamoDB::Model::AttributeValue().SetS(titleToGet));
            request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(1933));

            const Aws::DynamoDB::Model::GetItemOutcome &result =
dynamoClient.GetItem(
                    request);
            if (!result.IsSuccess()) {
                std::cerr << "Error " << result.GetError().GetMessage();
            }
```

```
            else {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = result.GetResult().GetItem();
                if (!item.empty()) {
                    std::cout << "\nHere's what I found:" << std::endl;
                    printMovieInfo(item);
                }
                else {
                    std::cout << "\nThe movie was not found in the database."
                              << std::endl;
                }
            }
        }
    }

    // 6. Use Query with a key condition expression to return all movies
    //    released in a given year.
    Aws::String doAgain = "n";
    do {
        Aws::DynamoDB::Model::QueryRequest req;

        req.SetTableName(MOVIE_TABLE_NAME);

        // "year" is a DynamoDB reserved keyword and must be replaced with an
        // expression attribute name.
        req.SetKeyConditionExpression("#dynobase_year = :valueToMatch");
        req.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

        int yearToMatch = askQuestionForIntRange(
                "\nLet's get a list of movies released in"
                " a given year. Enter a year between 1972 and 2018 ",
                1972, 2018);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
        attributeValues.emplace(":valueToMatch",
                                Aws::DynamoDB::Model::AttributeValue().SetN(
                                        yearToMatch));
        req.SetExpressionAttributeValues(attributeValues);

        const Aws::DynamoDB::Model::QueryOutcome &result = dynamoClient.Query(req);
        if (result.IsSuccess()) {
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
            if (!items.empty()) {
                std::cout << "\nThere were " << items.size()
```

```
                          << " movies in the database from "
                          << yearToMatch << "." << std::endl;
                for (const auto &item: items) {
                    printMovieInfo(item);
                }
                doAgain = "n";
            }
            else {
                std::cout << "\nNo movies from " << yearToMatch
                          << " were found in the database"
                          << std::endl;
                doAgain = askQuestion(Aws::String("Try another year? (y/n) "));
            }
        }
        else {
            std::cerr << "Failed to Query items: " << result.GetError().GetMessage()
                      << std::endl;
        }

    } while (doAgain == "y");

    //  7. Use Scan to return movies released within a range of years.
    //     Show how to paginate data using ExclusiveStartKey. (Scan +
FilterExpression)
    {
        int startYear = askQuestionForIntRange("\nNow let's scan a range of years "
                                               "for movies in the database. Enter a
start year: ",
                                               1972, 2018);
        int endYear = askQuestionForIntRange("\nEnter an end year: ",
                                             startYear, 2018);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
exclusiveStartKey;
        do {
            Aws::DynamoDB::Model::ScanRequest scanRequest;
            scanRequest.SetTableName(MOVIE_TABLE_NAME);
            scanRequest.SetFilterExpression(
                    "#dynobase_year >= :startYear AND #dynobase_year <= :endYear");
            scanRequest.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

            Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
            attributeValues.emplace(":startYear",
                                    Aws::DynamoDB::Model::AttributeValue().SetN(
```

```
                                        startYear));
            attributeValues.emplace(":endYear",
                                    Aws::DynamoDB::Model::AttributeValue().SetN(
                                        endYear));
            scanRequest.SetExpressionAttributeValues(attributeValues);

            if (!exclusiveStartKey.empty()) {
                scanRequest.SetExclusiveStartKey(exclusiveStartKey);
            }

            const Aws::DynamoDB::Model::ScanOutcome &result = dynamoClient.Scan(
                    scanRequest);
            if (result.IsSuccess()) {
                const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
                if (!items.empty()) {
                    std::stringstream stringStream;
                    stringStream << "\nFound " << items.size() << " movies in one
scan."
                                 << " How many would you like to see? ";
                    size_t count = askQuestionForInt(stringStream.str());
                    for (size_t i = 0; i < count && i < items.size(); ++i) {
                        printMovieInfo(items[i]);
                    }
                }
                else {
                    std::cout << "\nNo movies in the database between " << startYear
<<
                              " and " << endYear << "." << std::endl;
                }

                exclusiveStartKey = result.GetResult().GetLastEvaluatedKey();
                if (!exclusiveStartKey.empty()) {
                    std::cout << "Not all movies were retrieved. Scanning for more."
                              << std::endl;
                }
                else {
                    std::cout << "All movies were retrieved with this scan."
                              << std::endl;
                }
            }
            else {
                std::cerr << "Failed to Scan movies: "
                          << result.GetError().GetMessage() << std::endl;
```

```
            }
        } while (!exclusiveStartKey.empty());
    }

    // 8. Delete a movie. (DeleteItem)
    {
        std::stringstream stringStream;
        stringStream << "\nWould you like to delete the movie " << title
                     << " from the database? (y/n) ";
        Aws::String answer = askQuestion(stringStream.str());
        if (answer == "y") {
            Aws::DynamoDB::Model::DeleteItemRequest request;
            request.AddKey(YEAR_KEY,
 Aws::DynamoDB::Model::AttributeValue().SetN(year));
            request.AddKey(TITLE_KEY,
                           Aws::DynamoDB::Model::AttributeValue().SetS(title));
            request.SetTableName(MOVIE_TABLE_NAME);

            const Aws::DynamoDB::Model::DeleteItemOutcome &result =
 dynamoClient.DeleteItem(
                    request);
            if (result.IsSuccess()) {
                std::cout << "\nRemoved \"" << title << "\" from the database."
                          << std::endl;
            }
            else {
                std::cerr << "Failed to delete the movie: "
                          << result.GetError().GetMessage()
                          << std::endl;
            }
        }
    }

    return true;
}

//! Routine to convert a JsonView object to an attribute map.
/*!
  \sa movieJsonViewToAttributeMap()
  \param jsonView: Json view object.
  \return map: Map that can be used in a DynamoDB request.
 */
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
AwsDoc::DynamoDB::movieJsonViewToAttributeMap(
```

```
        const Aws::Utils::Json::JsonView &jsonView) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> result;

    if (jsonView.KeyExists(YEAR_KEY)) {
        result[YEAR_KEY].SetN(jsonView.GetInteger(YEAR_KEY));
    }
    if (jsonView.KeyExists(TITLE_KEY)) {
        result[TITLE_KEY].SetS(jsonView.GetString(TITLE_KEY));
    }
    if (jsonView.KeyExists(INFO_KEY)) {
        Aws::Map<Aws::String, const
 std::shared_ptr<Aws::DynamoDB::Model::AttributeValue>> infoMap;
        Aws::Utils::Json::JsonView infoView = jsonView.GetObject(INFO_KEY);
        if (infoView.KeyExists(RATING_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
 std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetN(infoView.GetDouble(RATING_KEY));
            infoMap.emplace(std::make_pair(RATING_KEY, attributeValue));
        }
        if (infoView.KeyExists(PLOT_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
 std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetS(infoView.GetString(PLOT_KEY));
            infoMap.emplace(std::make_pair(PLOT_KEY, attributeValue));
        }

        result[INFO_KEY].SetM(infoMap);
    }

    return result;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;
```

```
    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
```

```
                            << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
```

```
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
```

```
    }
    return false;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`BatchExecuteStatement`

O código de exemplo a seguir mostra como usar `BatchExecuteStatement`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use lotes de instruções INSERT para adicionar itens.

```
    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
```

```
    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
                "On a scale of 1 - 10, how do you rate it? ",
                1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
"));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
              << (titles.size() == 1 ? " movie " : " movies ")
              << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
```

```
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

           // Create attribute for the info map.
           Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

           std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                   ALLOCATION_TAG.c_str());
           ratingAttribute->SetN(ratings[i]);
           infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

           std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                   ALLOCATION_TAG.c_str());
           plotAttribute->SetS(plots[i]);
           infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
           attributes.push_back(infoMapAttribute);
           statements[i].SetParameters(attributes);
       }

       Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

       request.SetStatements(statements);

       Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
               request);
       if (!outcome.IsSuccess()) {
           std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   }
```

Use lotes de instruções SELECT para obter itens.

```
   // 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
   {
       Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
```

```
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
```

```
    }
```

Use lotes de instruções UPDATE para atualizar itens.

```
    // 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

    for (size_t i = 0; i < titles.size(); ++i) {
        ratings[i] = askQuestionForFloatRange(
                Aws::String("\nLet's update your the movie, \"") + titles[i] +
                ".\nYou rated it  " + std::to_string(ratings[i])
                + ", what new rating would you give it? ", 1, 10);
    }

    std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";


        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }
```

```
        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

Use lotes de instruções DELETE para excluir itens.

```
    // 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para C++ da API.

`BatchGetItem`

O código de exemplo a seguir mostra como usar `BatchGetItem`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Batch get items from different Amazon DynamoDB tables.
/*!
  \sa batchGetItem()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::batchGetItem(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchGetItemRequest request;

    // Table1: Forum.
    Aws::String table1Name = "Forum";
    Aws::DynamoDB::Model::KeysAndAttributes table1KeysAndAttributes;

    // Table1: Projection expression.
```

```
    table1KeysAndAttributes.SetProjectionExpression("#n, Category, Messages, #v");

    // Table1: Expression attribute names.
    Aws::Http::HeaderValueCollection headerValueCollection;
    headerValueCollection.emplace("#n", "Name");
    headerValueCollection.emplace("#v", "Views");
    table1KeysAndAttributes.SetExpressionAttributeNames(headerValueCollection);

    // Table1: Set key name, type, and value to search.
    std::vector<Aws::String> nameValues = {"Amazon DynamoDB", "Amazon S3"};
    for (const Aws::String &name: nameValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetS(name);
        keys.emplace("Name", key);
        table1KeysAndAttributes.AddKeys(keys);
    }

    Aws::Map<Aws::String, Aws::DynamoDB::Model::KeysAndAttributes> requestItems;
    requestItems.emplace(table1Name, table1KeysAndAttributes);

    // Table2: ProductCatalog.
    Aws::String table2Name = "ProductCatalog";
    Aws::DynamoDB::Model::KeysAndAttributes table2KeysAndAttributes;
    table2KeysAndAttributes.SetProjectionExpression("Title, Price, Color");

    // Table2: Set key name, type, and value to search.
    std::vector<Aws::String> idValues = {"102", "103", "201"};
    for (const Aws::String &id: idValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetN(id);
        keys.emplace("Id", key);
        table2KeysAndAttributes.AddKeys(keys);
    }

    requestItems.emplace(table2Name, table2KeysAndAttributes);

    bool result = true;
    do {  // Use a do loop to handle pagination.
        request.SetRequestItems(requestItems);
        const Aws::DynamoDB::Model::BatchGetItemOutcome &outcome =
dynamoClient.BatchGetItem(
                request);
```

```
        if (outcome.IsSuccess()) {
            for (const auto &responsesMapEntry: outcome.GetResult().GetResponses())
{
                Aws::String tableName = responsesMapEntry.first;
                const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &tableResults = responsesMapEntry.second;
                std::cout << "Retrieved " << tableResults.size()
                          << " responses for table '" << tableName << "'.\n"
                          << std::endl;
                if (tableName == "Forum") {

                    std::cout << "Name | Category | Message | Views" << std::endl;
                    for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Name").GetS() << " | ";
                        std::cout << item.at("Category").GetS() << " | ";
                        std::cout << (item.count("Message") == 0 ? "" : item.at(
                                "Messages").GetN()) << " | ";
                        std::cout << (item.count("Views") == 0 ? "" : item.at(
                                "Views").GetN()) << std::endl;
                    }
                }
                else {
                    std::cout << "Title | Price | Color" << std::endl;
                    for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Title").GetS() << " | ";
                        std::cout << (item.count("Price") == 0 ? "" : item.at(
                                "Price").GetN());
                        if (item.count("Color")) {
                            std::cout << " | ";
                            for (const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> &listItem: item.at(
                                        "Color").GetL())
                                std::cout << listItem->GetS() << " ";
                        }
                        std::cout << std::endl;
                    }
                }
                std::cout << std::endl;
            }

            // If necessary, repeat request for remaining items.
```

```
            requestItems = outcome.GetResult().GetUnprocessedKeys();
        }
        else {
            std::cerr << "Batch get item failed: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
            break;
        }
    } while (!requestItems.empty());

    return result;
}
```

- Para obter detalhes da API, consulte BatchGetItem em Referência AWS SDK para C++ da API.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Batch write items from a JSON file.
/*!
  \sa batchWriteItem()
  \param jsonFilePath: JSON file path.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

/*
 * The input for this routine is a JSON file that you can download from the
 following URL:
```

```
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.html.
 *
 * The JSON data uses the BatchWriteItem API request syntax. The JSON strings are
 * converted to AttributeValue objects. These AttributeValue objects will then
 generate
 * JSON strings when constructing the BatchWriteItem request, essentially outputting
 * their input.
 *
 * This is perhaps an artificial example, but it demonstrates the APIs.
 */

bool AwsDoc::DynamoDB::batchWriteItem(const Aws::String &jsonFilePath,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    std::ifstream fileStream(jsonFilePath);

    if (!fileStream) {
        std::cerr << "Error: could not open file '" << jsonFilePath << "'."
                  << std::endl;
    }

    std::stringstream stringStream;
    stringStream << fileStream.rdbuf();
    Aws::Utils::Json::JsonValue jsonValue(stringStream);

    Aws::DynamoDB::Model::BatchWriteItemRequest batchWriteItemRequest;
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> level1Map =
 jsonValue.View().GetAllObjects();
    for (const auto &level1Entry: level1Map) {
        const Aws::Utils::Json::JsonView &entriesView = level1Entry.second;
        const Aws::String &tableName = level1Entry.first;
        // The JSON entries at this level are as follows:
        //  key - table name
        //  value - list of request objects
        if (!entriesView.IsListType()) {
            std::cerr << "Error: JSON file entry '"
                      << tableName << "' is not a list." << std::endl;
            continue;
        }

        Aws::Utils::Array<Aws::Utils::Json::JsonView> entries =
 entriesView.AsArray();

        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;
```

```
        if (AwsDoc::DynamoDB::addWriteRequests(tableName, entries,
                                               writeRequests)) {
            batchWriteItemRequest.AddRequestItems(tableName, writeRequests);
        }
    }

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchWriteItemOutcome outcome =
 dynamoClient.BatchWriteItem(
            batchWriteItemRequest);

    if (outcome.IsSuccess()) {
        std::cout << "DynamoDB::BatchWriteItem was successful." << std::endl;
    }
    else {
        std::cerr << "Error with DynamoDB::BatchWriteItem. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}

//! Convert requests in JSON format to a vector of WriteRequest objects.
/*!
  \sa addWriteRequests()
  \param tableName: Name of the table for the write operations.
  \param requestsJson: Request data in JSON format.
  \param writeRequests: Vector to receive the WriteRequest objects.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::addWriteRequests(const Aws::String &tableName,
                                        const
 Aws::Utils::Array<Aws::Utils::Json::JsonView> &requestsJson,

 Aws::Vector<Aws::DynamoDB::Model::WriteRequest> &writeRequests) {
    for (size_t i = 0; i < requestsJson.GetLength(); ++i) {
        const Aws::Utils::Json::JsonView &requestsEntry = requestsJson[i];
        if (!requestsEntry.IsObject()) {
            std::cerr << "Error: incorrect requestsEntry type "
                      << requestsEntry.WriteReadable() << std::endl;
            return false;
```

```
        }

        Aws::Map<Aws::String, Aws::Utils::Json::JsonView> requestsMap =
 requestsEntry.GetAllObjects();

        for (const auto &request: requestsMap) {
            const Aws::String &requestType = request.first;
            const Aws::Utils::Json::JsonView &requestJsonView = request.second;

            if (requestType == "PutRequest") {
                if (!requestJsonView.ValueExists("Item")) {
                    std::cerr << "Error: item key missing for requests "
                              << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 attributes;
                if (!getAttributeObjectsMap(requestJsonView.GetObject("Item"),
                                            attributes)) {
                    std::cerr << "Error getting attributes "
                              << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }

                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(attributes);
                writeRequests.push_back(
                        Aws::DynamoDB::Model::WriteRequest().WithPutRequest(
                                putRequest));
            }
            else {
                std::cerr << "Error: unimplemented request type '" << requestType
                          << "'." << std::endl;
            }
        }
    }

    return true;
}

//! Generate a map of AttributeValue objects from JSON records.
/*!
  \sa getAttributeObjectsMap()
  \param jsonView: JSONView of attribute records.
```

```
  \param writeRequests: Map to receive the AttributeValue objects.
  \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::getAttributeObjectsMap(const Aws::Utils::Json::JsonView &jsonView,
                                         Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue> &attributes) {
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> objectsMap =
 jsonView.GetAllObjects();
    for (const auto &entry: objectsMap) {
        const Aws::String &attributeKey = entry.first;
        const Aws::Utils::Json::JsonView &attributeJsonView = entry.second;

        if (!attributeJsonView.IsObject()) {
            std::cerr << "Error: attribute not an object "
                      << attributeJsonView.WriteReadable() << std::endl;
            return false;
        }

        attributes.emplace(attributeKey,
                           Aws::DynamoDB::Model::AttributeValue(attributeJsonView));
    }

    return true;
}
```

- Para obter detalhes da API, consulte BatchWriteItem em Referência AWS SDK para C++ da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon DynamoDB table.
/*!
  \sa createTable()
  \param tableName: Name for the DynamoDB table.
  \param primaryKey: Primary key for the DynamoDB table.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createTable(const Aws::String &tableName,
                                   const Aws::String &primaryKey,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
              " with a simple primary key: \"" << primaryKey << "\"." << std::endl;

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey;
    hashKey.SetAttributeName(primaryKey);
    hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey);

    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement;
    keySchemaElement.WithAttributeName(primaryKey).WithKeyType(
            Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(keySchemaElement);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::CreateTableOutcome &outcome =
 dynamoClient.CreateTable(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Table \""
                  << outcome.GetResult().GetTableDescription().GetTableName() <<
                  " created!" << std::endl;
    }
    else {
```

```
        std::cerr << "Failed to create table: " << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
```

```
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para C++ da API.

DeleteItem

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an item from an Amazon DynamoDB table.
/*!
  \sa deleteItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::deleteItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
```

```
    Aws::DynamoDB::Model::DeleteItemRequest request;

    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteItemOutcome &outcome =
 dynamoClient.DeleteItem(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Item \"" << partitionValue << "\" deleted!" << std::endl;
    }
    else {
        std::cerr << "Failed to delete item: " << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
```

```
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte DeleteItem em Referência AWS SDK para C++ da API.

DeleteTable

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon DynamoDB table.
/*!
  \sa deleteTable()
```

```
  \param tableName: The DynamoDB table name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteTable(const Aws::String &tableName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para C++ da API.

DescribeTable

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe an Amazon DynamoDB table.
/*!
  \sa describeTable()
  \param tableName: The DynamoDB table name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::describeTable(const Aws::String &tableName,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DescribeTableOutcome &outcome =
 dynamoClient.DescribeTable(
            request);

    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::TableDescription &td =
 outcome.GetResult().GetTable();
        std::cout << "Table name  : " << td.GetTableName() << std::endl;
        std::cout << "Table ARN   : " << td.GetTableArn() << std::endl;
        std::cout << "Status      : "
                  << Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(
                          td.GetTableStatus()) << std::endl;
        std::cout << "Item count  : " << td.GetItemCount() << std::endl;
        std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;

        const Aws::DynamoDB::Model::ProvisionedThroughputDescription &ptd =
 td.GetProvisionedThroughput();
        std::cout << "Throughput" << std::endl;
        std::cout << "  Read Capacity : " << ptd.GetReadCapacityUnits() <<
 std::endl;
        std::cout << "  Write Capacity: " << ptd.GetWriteCapacityUnits() <<
 std::endl;

        const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition> &ad =
 td.GetAttributeDefinitions();
        std::cout << "Attributes" << std::endl;
        for (const auto &a: ad)
            std::cout << "  " << a.GetAttributeName() << " (" <<
```

```
 Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(
                              a.GetAttributeType()) <<
                       ")" << std::endl;
    }
    else {
        std::cerr << "Failed to describe table: " <<
 outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeTable em Referência AWS SDK para C++ da API.

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use uma instrução INSERT para adicionar um item.

```
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
```

```
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    }
```

Use uma instrução SELECT para obter um item.

```
    //  3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            // Print the retrieved movie information.
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were retrieved. "
                          << " There should be only one movie." << std::endl;
            }
```

```
        }
    }
```

Use uma instrução UPDATE para atualizar um item.

```
    //  4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update a movie: "
                      << outcome.GetError().GetMessage();
            return false;
        }
    }
```

Use uma instrução DELETE para excluir um item.

```
    // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para C++ da API.

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get an item from an Amazon DynamoDB table.
```

```
/*!
  \sa getItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::getItem(const Aws::String &tableName,
                               const Aws::String &partitionKey,
                               const Aws::String &partitionValue,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::GetItemRequest request;

    // Set up the request.
    request.SetTableName(tableName);
    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));

    // Retrieve the item's fields and values.
    const Aws::DynamoDB::Model::GetItemOutcome &outcome =
 dynamoClient.GetItem(request);
    if (outcome.IsSuccess()) {
        // Reference the retrieved fields/values.
        const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item =
 outcome.GetResult().GetItem();
        if (!item.empty()) {
            // Output each retrieved field and its value.
            for (const auto &i: item)
                std::cout << "Values: " << i.first << ": " << i.second.GetS()
                          << std::endl;
        }
        else {
            std::cout << "No item found with the key " << partitionKey << std::endl;
        }
    }
    else {
        std::cerr << "Failed to get item: " << outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte GetItem em Referência AWS SDK para C++ da API.

ListTables

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the Amazon DynamoDB tables for the current AWS account.
/*!
  \sa listTables()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::listTables(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
    listTablesRequest.SetLimit(50);
    do {
        const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
 dynamoClient.ListTables(
                listTablesRequest);
        if (!outcome.IsSuccess()) {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        for (const auto &tableName: outcome.GetResult().GetTableNames())
            std::cout << tableName << std::endl;
        listTablesRequest.SetExclusiveStartTableName(
```

```
                outcome.GetResult().GetLastEvaluatedTableName());

    } while (!listTablesRequest.GetExclusiveStartTableName().empty());

    return true;
}
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para C++ da API.

PutItem

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Put an item in an Amazon DynamoDB table.
/*!
  \sa putItem()
  \param tableName: The table name.
  \param artistKey: The artist key. This is the partition key for the table.
  \param artistValue: The artist value.
  \param albumTitleKey: The album title key.
  \param albumTitleValue: The album title value.
  \param awardsKey: The awards key.
  \param awardsValue: The awards value.
  \param songTitleKey: The song title key.
  \param songTitleValue: The song title value.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::putItem(const Aws::String &tableName,
                               const Aws::String &artistKey,
                               const Aws::String &artistValue,
                               const Aws::String &albumTitleKey,
                               const Aws::String &albumTitleValue,
```

```
                                const Aws::String &awardsKey,
                                const Aws::String &awardsValue,
                                const Aws::String &songTitleKey,
                                const Aws::String &songTitleValue,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::PutItemRequest putItemRequest;
    putItemRequest.SetTableName(tableName);

    putItemRequest.AddItem(artistKey, Aws::DynamoDB::Model::AttributeValue().SetS(
            artistValue)); // This is the hash key.
    putItemRequest.AddItem(albumTitleKey,
 Aws::DynamoDB::Model::AttributeValue().SetS(
            albumTitleValue));
    putItemRequest.AddItem(awardsKey,

 Aws::DynamoDB::Model::AttributeValue().SetS(awardsValue));
    putItemRequest.AddItem(songTitleKey,

 Aws::DynamoDB::Model::AttributeValue().SetS(songTitleValue));

    const Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
            putItemRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully added Item!" << std::endl;
    }
    else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
```

```
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK para C++ da API.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Perform a query on an Amazon DynamoDB Table and retrieve items.
/*!
  \sa queryItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param projectionExpression: The projections expression, which is ignored if
 empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */

/*
 * The partition key attribute is searched with the specified value. By default, all
 fields and values
 * contained in the item are returned. If an optional projection expression is
 * specified on the command line, only the specified fields and values are
 * returned.
 */

bool AwsDoc::DynamoDB::queryItems(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::String &projectionExpression,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::QueryRequest request;

    request.SetTableName(tableName);
```

```
    if (!projectionExpression.empty()) {
        request.SetProjectionExpression(projectionExpression);
    }

    // Set query key condition expression.
    request.SetKeyConditionExpression(partitionKey + "= :valueToMatch");

    // Set Expression AttributeValues.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
    attributeValues.emplace(":valueToMatch", partitionValue);

    request.SetExpressionAttributeValues(attributeValues);

    bool result = true;

    // "exclusiveStartKey" is used for pagination.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> exclusiveStartKey;
    do {
        if (!exclusiveStartKey.empty()) {
            request.SetExclusiveStartKey(exclusiveStartKey);
            exclusiveStartKey.clear();
        }
        // Perform Query operation.
        const Aws::DynamoDB::Model::QueryOutcome &outcome =
dynamoClient.Query(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            if (!items.empty()) {
                std::cout << "Number of items retrieved from Query: " <<
items.size()
                          << std::endl;
                // Iterate each item and print.
                for (const auto &item: items) {
                    std::cout
                            <<
"*****************************************************"
                            << std::endl;
                    // Output each retrieved field and its value.
                    for (const auto &i: item)
                        std::cout << i.first << ": " << i.second.GetS() <<
std::endl;
                }
```

```
            }
            else {
                std::cout << "No item found in table: " << tableName << std::endl;
            }

            exclusiveStartKey = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Query items: " <<
 outcome.GetError().GetMessage();
            result = false;
            break;
        }
    } while (!exclusiveStartKey.empty());

    return result;
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para C++ .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Scan an Amazon DynamoDB table.
/*!
  \sa scanTable()
  \param tableName: Name for the DynamoDB table.
  \param projectionExpression: An optional projection expression, ignored if empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::DynamoDB::scanTable(const Aws::String &tableName,
                                 const Aws::String &projectionExpression,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::ScanRequest request;
    request.SetTableName(tableName);

    if (!projectionExpression.empty())
        request.SetProjectionExpression(projectionExpression);

    Aws::Vector<Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>>
 all_items;
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 last_evaluated_key; // Used for pagination;
    do {
        if (!last_evaluated_key.empty()) {
            request.SetExclusiveStartKey(last_evaluated_key);
        }
        const Aws::DynamoDB::Model::ScanOutcome &outcome =
 dynamoClient.Scan(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            all_items.insert(all_items.end(), items.begin(), items.end());

            last_evaluated_key = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Scan items: " << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!last_evaluated_key.empty());

    if (!all_items.empty()) {
        std::cout << "Number of items retrieved from scan: " << all_items.size()
                  << std::endl;
        // Iterate each item and print.
        for (const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 &itemMap: all_items) {
            std::cout << "*****************************************************"
```

```
                        << std::endl;
            // Output each retrieved field and its value.
            for (const auto &itemEntry: itemMap)
                std::cout << itemEntry.first << ": " << itemEntry.second.GetS()
                          << std::endl;
        }
    }

    else {
        std::cout << "No items found in table: " << tableName << std::endl;
    }

    return true;
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para C++ .

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update an Amazon DynamoDB table item.
/*!
  \sa updateItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param attributeKey: The key for the attribute to be updated.
  \param attributeValue: The value for the attribute to be updated.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */
```

```
/*
 *  The example code only sets/updates an attribute value. It processes
 *  the attribute value as a string, even if the value could be interpreted
 *  as a number. Also, the example code does not remove an existing attribute
 *  from the key value.
 */

bool AwsDoc::DynamoDB::updateItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::String &attributeKey,
                                  const Aws::String &attributeValue,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // *** Define UpdateItem request arguments.
    // Define TableName argument.
    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(tableName);

    // Define KeyName argument.
    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(partitionValue);
    request.AddKey(partitionKey, attribValue);

    // Construct the SET update expression argument.
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument.
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = attributeKey;
    request.SetExpressionAttributeNames(expressionAttributeNames);

    // Construct attribute value argument.
    Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
    attributeUpdatedValue.SetS(attributeValue);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 expressionAttributeValues;
    expressionAttributeValues[":valueA"] = attributeUpdatedValue;
    request.SetExpressionAttributeValues(expressionAttributeValues);
```

```
    // Update the item.
    const Aws::DynamoDB::Model::UpdateItemOutcome &outcome =
 dynamoClient.UpdateItem(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Item was updated" << std::endl;
    } else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
```

```
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para C++ da API.

`UpdateTable`

O código de exemplo a seguir mostra como usar `UpdateTable`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update a DynamoDB table.
/*!
  \sa updateTable()
  \param tableName: Name for the DynamoDB table.
  \param readCapacity: Provisioned read capacity.
  \param writeCapacity: Provisioned write capacity.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::updateTable(const Aws::String &tableName,
                                   long long readCapacity, long long writeCapacity,
```

```
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
   Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

   std::cout << "Updating " << tableName << " with new provisioned throughput
values"
             << std::endl;
   std::cout << "Read capacity : " << readCapacity << std::endl;
   std::cout << "Write capacity: " << writeCapacity << std::endl;

   Aws::DynamoDB::Model::UpdateTableRequest request;
   Aws::DynamoDB::Model::ProvisionedThroughput provisionedThroughput;

provisionedThroughput.WithReadCapacityUnits(readCapacity).WithWriteCapacityUnits(
           writeCapacity);

request.WithProvisionedThroughput(provisionedThroughput).WithTableName(tableName);

   const Aws::DynamoDB::Model::UpdateTableOutcome &outcome =
dynamoClient.UpdateTable(
           request);
   if (outcome.IsSuccess()) {
       std::cout << "Successfully updated the table." << std::endl;
   } else {
       const Aws::DynamoDB::DynamoDBError &error = outcome.GetError();
       if (error.GetErrorType() == Aws::DynamoDB::DynamoDBErrors::VALIDATION &&
           error.GetMessage().find("The provisioned throughput for the table will
not change") != std::string::npos) {
           std::cout << "The provisioned throughput for the table will not change."
<< std::endl;
       } else {
           std::cerr << outcome.GetError().GetMessage() << std::endl;
           return false;
       }
   }

   return waitTableActive(tableName, dynamoClient);
}
```

Código que aguarda a tabela se tornar ativa.

```
//! Query a newly created DynamoDB table until it is active.
```

```
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte UpdateTable em Referência AWS SDK para C++ da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        //  1. Create a table. (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::partiqlBatchExecuteScenario(clientConfig);

            // 7. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }

//! Scenario to modify and query a DynamoDB table using PartiQL batch statements.
/*!
  \sa partiqlBatchExecuteScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::partiqlBatchExecuteScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
```

```
                "On a scale of 1 - 10, how do you rate it? ",
                1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
"));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
              << (titles.size() == 1 ? " movie " : " movies ")
              << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

            // Create attribute for the info map.
            Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            ratingAttribute->SetN(ratings[i]);
            infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);
```

```
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            plotAttribute->SetS(plots[i]);
            infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
            attributes.push_back(infoMapAttribute);
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Retrieving the movie data with a batch \"SELECT\" statement."
              << std::endl;

    // 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
```

```
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    // 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

    for (size_t i = 0; i < titles.size(); ++i) {
        ratings[i] = askQuestionForFloatRange(
                Aws::String("\nLet's update your the movie, \"") + titles[i] +
                ".\nYou rated it  " + std::to_string(ratings[i])
                + ", what new rating would you give it? ", 1, 10);
    }
```

```
   std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

   {
       Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
               titles.size());

       std::stringstream sqlStream;
       sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                 << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                 << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";


       std::string sql(sqlStream.str());

       for (size_t i = 0; i < statements.size(); ++i) {
           statements[i].SetStatement(sql);

           Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
           attributes.push_back(
                   Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
           attributes.push_back(
                   Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
           statements[i].SetParameters(attributes);
       }

       Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

       request.SetStatements(statements);
       Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
               request);
       if (!outcome.IsSuccess()) {
           std::cerr << "Failed to update movie information: "
                     << outcome.GetError().GetMessage() << std::endl;
           return false;
       }
   }

   std::cout << "Retrieving the updated movie data with a batch \"SELECT\"
statement."
             << std::endl;
```

```
   // 5. Get the updated data for multiple movies using "Select" statements.
(BatchExecuteStatement)
   {
       Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
               titles.size());
       std::stringstream sqlStream;
       sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                 << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

       std::string sql(sqlStream.str());

       for (size_t i = 0; i < statements.size(); ++i) {
           statements[i].SetStatement(sql);
           Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
           attributes.push_back(
                   Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
           statements[i].SetParameters(attributes);
       }

       Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

       request.SetStatements(statements);

       Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
               request);
       if (outcome.IsSuccess()) {
           const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

           const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

           for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
               const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

               printMovieInfo(item);
           }
       }
```

```
        else {
            std::cerr << "Failed to retrieve the movies information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Deleting the movie data with a batch \"DELETE\" statement."
              << std::endl;

    // 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

```
    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
```

```
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
 dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
```

```
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
```

```
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para C++ da API.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        //  1. Create a table. (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::partiqlExecuteScenario(clientConfig);

            // 7. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }

//! Scenario to modify and query a DynamoDB table using single PartiQL statements.
/*!
  \sa partiqlExecuteScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::partiqlExecuteScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
 ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
```

```
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
              << std::endl;

    //  3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());
```

```
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            // Print the retrieved movie information.
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }

    //  4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
```

```
                << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update a movie: "
                      << outcome.GetError().GetMessage();
            return false;
        }
    }

    std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

    //  5. Get the updated data for the movie using a "Select" statement.
(ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve the movie information: "
```

```
                    << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }

    std::cout << "Deleting the movie" << std::endl;

    // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
```

```
    }

    std::cout << "Movie successfully deleted." << std::endl;
    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
```

```
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
 dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
```

```
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para C++ da API.

Exemplos do Amazon EC2 usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Amazon EC2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS ec2)

# Set this project's name.
project("hello_ec2")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_ec2.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo fonte hello_ec2.cpp.

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <iomanip>
#include <iostream>

/*
 *  A "Hello EC2" starter application which initializes an Amazon Elastic Compute
 Cloud (Amazon EC2) client and describes
 *  the Amazon EC2 instances.
 *
 *  main function
 *
 *  Usage: 'hello_ec2'
 *
 */

int main(int argc, char **argv) {
    (void)argc;
    (void)argv;

    Aws::SDKOptions options;
```

```
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::EC2::EC2Client ec2Client(clientConfig);
        Aws::EC2::Model::DescribeInstancesRequest request;
        bool header = false;
        bool done = false;
        while (!done) {
            Aws::EC2::Model::DescribeInstancesOutcome outcome =
 ec2Client.DescribeInstances(request);
            if (outcome.IsSuccess()) {
                if (!header) {
                    std::cout << std::left <<
                              std::setw(48) << "Name" <<
                              std::setw(20) << "ID" <<
                              std::setw(25) << "Ami" <<
                              std::setw(15) << "Type" <<
                              std::setw(15) << "State" <<
                              std::setw(15) << "Monitoring" << std::endl;
                    header = true;
                }

                const std::vector<Aws::EC2::Model::Reservation> &reservations =
                        outcome.GetResult().GetReservations();

                for (const auto &reservation: reservations) {
                    const std::vector<Aws::EC2::Model::Instance> &instances =
                            reservation.GetInstances();
                    for (const auto &instance: instances) {
                        Aws::String instanceStateString =

 Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                        instance.GetState().GetName());

                        Aws::String typeString =

 Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                                        instance.GetInstanceType());
```

```
                        Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                                        instance.GetMonitoring().GetState());
                        Aws::String name = "Unknown";

                        const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                        auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                                                     [](const Aws::EC2::Model::Tag
&tag) {
                                                         return tag.GetKey() ==
"Name";
                                                     });
                        if (nameIter != tags.cend()) {
                            name = nameIter->GetValue();
                        }
                        std::cout <<
                                  std::setw(48) << name <<
                                  std::setw(20) << instance.GetInstanceId() <<
                                  std::setw(25) << instance.GetImageId() <<
                                  std::setw(15) << typeString <<
                                  std::setw(15) << instanceStateString <<
                                  std::setw(15) << monitorString << std::endl;
                    }
                }

                if (!outcome.GetResult().GetNextToken().empty()) {
                    request.SetNextToken(outcome.GetResult().GetNextToken());
                } else {
                    done = true;
                }
            } else {
                std::cerr << "Failed to describe EC2 instances:" <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
```

```
    return result;
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK para C++ da API.

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Allocate an Elastic IP address and associate it with an Amazon Elastic Compute
 Cloud
//! (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param[out] publicIPAddress: String to return the public IP address.
  \param[out] allocationID: String to return the allocation ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::allocateAndAssociateAddress(const Aws::String &instanceId,
 Aws::String &publicIPAddress,
                                               Aws::String &allocationID,
                                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AllocateAddressRequest request;
    request.SetDomain(Aws::EC2::Model::DomainType::vpc);
```

```
    const Aws::EC2::Model::AllocateAddressOutcome outcome =
            ec2Client.AllocateAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to allocate Elastic IP address:" <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    const Aws::EC2::Model::AllocateAddressResponse &response = outcome.GetResult();
    allocationID = response.GetAllocationId();
    publicIPAddress = response.GetPublicIp();


    return true;
}
```

- Para obter detalhes da API, consulte AllocateAddress em Referência AWS SDK para C++ da API.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

//! Associate an Elastic IP address with an EC2 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param allocationId: An Elastic IP allocation ID.
  \param[out] associationID: String to receive the association ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: True if the address was associated with the instance; otherwise,
 false.
```

```
 */
bool AwsDoc::EC2::associateAddress(const Aws::String &instanceId, const Aws::String
 &allocationId,
                                    Aws::String &associationID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AssociateAddressRequest request;
    request.SetInstanceId(instanceId);
    request.SetAllocationId(allocationId);

    Aws::EC2::Model::AssociateAddressOutcome outcome =
 ec2Client.AssociateAddress(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to associate address " << allocationId <<
                  " with instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully associated address " << allocationId <<
                  " with instance " << instanceId << std::endl;
        associationID = outcome.GetResult().GetAssociationId();
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AssociateAddress em Referência AWS SDK para C++ da API.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Authorize ingress to an Amazon Elastic Compute Cloud (Amazon EC2) group.
/*!
  \param groupID: The EC2 group ID.
  \param clientConfiguration: The ClientConfiguration object.
  \return bool: True if the operation was successful, false otherwise.
 */
bool
AwsDoc::EC2::authorizeSecurityGroupIngress(const Aws::String &groupID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest
 authorizeSecurityGroupIngressRequest;
    authorizeSecurityGroupIngressRequest.SetGroupId(groupID);
    buildSampleIngressRule(authorizeSecurityGroupIngressRequest);

    Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome
 authorizeSecurityGroupIngressOutcome =

 ec2Client.AuthorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);

    if (authorizeSecurityGroupIngressOutcome.IsSuccess()) {
        std::cout << "Successfully authorized security group ingress." << std::endl;
    } else {
        std::cerr << "Error authorizing security group ingress: "
                  << authorizeSecurityGroupIngressOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return authorizeSecurityGroupIngressOutcome.IsSuccess();
}
```

Função utilitária para criar uma regra de entrada.

```
//! Build a sample ingress rule.
/*!
  \param authorize_request: An 'AuthorizeSecurityGroupIngressRequest' instance.
  \return void:
 */
void buildSampleIngressRule(
        Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest &authorize_request) {
    Aws::String ingressIPRange = "203.0.113.0/24";  // Configure this for your
 allowed IP range.
    Aws::EC2::Model::IpRange ip_range;
    ip_range.SetCidrIp(ingressIPRange);

    Aws::EC2::Model::IpPermission permission1;
    permission1.SetIpProtocol("tcp");
    permission1.SetToPort(80);
    permission1.SetFromPort(80);
    permission1.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission1);

    Aws::EC2::Model::IpPermission permission2;
    permission2.SetIpProtocol("tcp");
    permission2.SetToPort(22);
    permission2.SetFromPort(22);
    permission2.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission2);
}
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngress em Referência AWS SDK para C++ da API.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
  \param keyPairName: A name for a key pair.
  \param keyFilePath: File path where the credentials are stored. Ignored if it is
 an empty string;
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createKeyPair(const Aws::String &keyPairName, const Aws::String
 &keyFilePath,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(keyPairName);

    Aws::EC2::Model::CreateKeyPairOutcome outcome =
 ec2Client.CreateKeyPair(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create key pair - "  << keyPairName << ". " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully created key pair named " <<
                  keyPairName << std::endl;
        if (!keyFilePath.empty()) {
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
                      keyFilePath << std::endl;
        }

    }
```

```
    return outcome.IsSuccess();

}
```

- Para obter detalhes da API, consulte CreateKeyPair em Referência AWS SDK para C++ da API.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create a security group.
/*!
  \param groupName: A security group name.
  \param description: A description.
  \param vpcID: A virtual private cloud (VPC) ID.
  \param[out] groupIDResult: A string to receive the group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createSecurityGroup(const Aws::String &groupName,
                                      const Aws::String &description,
                                      const Aws::String &vpcID,
                                      Aws::String &groupIDResult,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(groupName);
    request.SetDescription(description);
```

```
    request.SetVpcId(vpcID);

    const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
            ec2Client.CreateSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create security group:" <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    std::cout << "Successfully created security group named " << groupName <<
              std::endl;


    groupIDResult = outcome.GetResult().GetGroupId();

    return true;
}
```

- Para obter detalhes da API, consulte CreateSecurityGroup em Referência AWS SDK para C++ da API.

`CreateTags`

O código de exemplo a seguir mostra como usar `CreateTags`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Add or overwrite only the specified tags for the specified Amazon Elastic
 Compute Cloud (Amazon EC2) resource or resources.
/*!
  \param resources: The resources for the tags.
  \param tags: Vector of tags.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createTags(const Aws::Vector<Aws::String> &resources,
                             const Aws::Vector<Aws::EC2::Model::Tag> &tags,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateTagsRequest createTagsRequest;
    createTagsRequest.SetResources(resources);
    createTagsRequest.SetTags(tags);

    Aws::EC2::Model::CreateTagsOutcome outcome =
 ec2Client.CreateTags(createTagsRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created tags for resources" << std::endl;
    } else {
        std::cerr << "Failed to create tags for resources, " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTags em Referência AWS SDK para C++ da API.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
```

```
/*!
  \param keyPairName: A name for a key pair.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::EC2::deleteKeyPair(const Aws::String &keyPairName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteKeyPairRequest request;

    request.SetKeyName(keyPairName);
    const Aws::EC2::Model::DeleteKeyPairOutcome outcome = ec2Client.DeleteKeyPair(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete key pair " << keyPairName <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted key pair named " << keyPairName <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteKeyPair em Referência AWS SDK para C++ da API.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete a security group.
/*!
  \param securityGroupID: A security group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::deleteSecurityGroup(const Aws::String &securityGroupID,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteSecurityGroupRequest request;

    request.SetGroupId(securityGroupID);
    Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =
 ec2Client.DeleteSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete security group " << securityGroupID <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted security group " << securityGroupID <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteSecurityGroup em Referência AWS SDK para C++ da API.

`DescribeAddresses`

O código de exemplo a seguir mostra como usar `DescribeAddresses`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Elastic IP addresses.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeAddresses(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAddressesRequest request;
    Aws::EC2::Model::DescribeAddressesOutcome outcome =
 ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
                  std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
                  std::setw(30) << "Allocation ID" << std::setw(25) <<
                  "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
 outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
                    Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                            address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                      address.GetInstanceId() << std::setw(15) <<
                      address.GetPublicIp() << std::setw(10) << domainString <<
                      std::setw(30) << address.GetAllocationId() << std::setw(25)
                      << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeAddresses em Referência AWS SDK para C++ da API.

`DescribeAvailabilityZones`

O código de exemplo a seguir mostra como usar `DescribeAvailabilityZones`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! DescribeAvailabilityZones
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
int AwsDoc::EC2::describeAvailabilityZones(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
 ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "ZoneName" <<
                  std::setw(20) << "State" <<
                  std::setw(32) << "Region" << std::endl;

        const auto &zones =
                outcome.GetResult().GetAvailabilityZones();

        for (const auto &zone: zones) {
            Aws::String stateString =

 Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                            zone.GetState());
            std::cout << std::left <<
                      std::setw(32) << zone.GetZoneName() <<
                      std::setw(20) << stateString <<
                      std::setw(32) << zone.GetRegionName() << std::endl;
```

```
        }
    } else {
        std::cerr << "Failed to describe availability zones:" <<
                  outcome.GetError().GetMessage() << std::endl;

    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeAvailabilityZones em Referência AWS SDK para C++ da API.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instances associated with
 an account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeInstances(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
 ec2Client.DescribeInstances(request);
```

```
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<
                          std::setw(48) << "Name" <<
                          std::setw(20) << "ID" <<
                          std::setw(25) << "Ami" <<
                          std::setw(15) << "Type" <<
                          std::setw(15) << "State" <<
                          std::setw(15) << "Monitoring" << std::endl;
                header = true;
            }

            const std::vector<Aws::EC2::Model::Reservation> &reservations =
                    outcome.GetResult().GetReservations();

            for (const auto &reservation: reservations) {
                const std::vector<Aws::EC2::Model::Instance> &instances =
                        reservation.GetInstances();
                for (const auto &instance: instances) {
                    Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                    instance.GetState().GetName());

                    Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                                    instance.GetInstanceType());

                    Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                                    instance.GetMonitoring().GetState());
                    Aws::String name = "Unknown";

                    const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                    auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                                                 [](const Aws::EC2::Model::Tag &tag)
{
                                                     return tag.GetKey() == "Name";
                                                 });
                    if (nameIter != tags.cend()) {
                        name = nameIter->GetValue();
```

```
                    }
                    std::cout <<
                              std::setw(48) << name <<
                              std::setw(20) << instance.GetInstanceId() <<
                              std::setw(25) << instance.GetImageId() <<
                              std::setw(15) << typeString <<
                              std::setw(15) << instanceStateString <<
                              std::setw(15) << monitorString << std::endl;
                }
            }

            if (!outcome.GetResult().GetNextToken().empty()) {
                request.SetNextToken(outcome.GetResult().GetNextToken());
            } else {
                done = true;
            }
        } else {
            std::cerr << "Failed to describe EC2 instances:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte DescribeInstances em Referência AWS SDK para C++ da API.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instance key pairs.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeKeyPairs(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeKeyPairsRequest request;

    Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
 ec2Client.DescribeKeyPairs(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "Name" <<
                  std::setw(64) << "Fingerprint" << std::endl;

        const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
                outcome.GetResult().GetKeyPairs();
        for (const auto &key_pair: key_pairs) {
            std::cout << std::left <<
                      std::setw(32) << key_pair.GetKeyName() <<
                      std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe key pairs:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeKeyPairs em Referência AWS SDK para C++ da API.

`DescribeRegions`

O código de exemplo a seguir mostra como usar `DescribeRegions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) Regions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeRegions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::DescribeRegionsRequest request;
    Aws::EC2::Model::DescribeRegionsOutcome outcome =
 ec2Client.DescribeRegions(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "RegionName" <<
                  std::setw(64) << "Endpoint" << std::endl;

        const auto &regions = outcome.GetResult().GetRegions();
        for (const auto &region: regions) {
            std::cout << std::left <<
                      std::setw(32) << region.GetRegionName() <<
                      std::setw(64) << region.GetEndpoint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe regions:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    std::cout << std::endl;

    return outcome.IsSuccess();

}
```

- Para obter detalhes da API, consulte DescribeRegions em Referência AWS SDK para C++ da API.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) security groups, or a
 specific group.
/*!
  \param groupID: A group ID, ignored if empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeSecurityGroups(const Aws::String &groupID,
                                         const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeSecurityGroupsRequest request;

    if (!groupID.empty()) {
        request.AddGroupIds(groupID);
    }

    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
 ec2Client.DescribeSecurityGroups(request);
        if (outcome.IsSuccess()) {
            std::cout << std::left <<
```

```
                    std::setw(32) << "Name" <<
                    std::setw(30) << "GroupId" <<
                    std::setw(30) << "VpcId" <<
                    std::setw(64) << "Description" << std::endl;

            const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
                    outcome.GetResult().GetSecurityGroups();

            for (const auto &securityGroup: securityGroups) {
                std::cout << std::left <<
                          std::setw(32) << securityGroup.GetGroupName() <<
                          std::setw(30) << securityGroup.GetGroupId() <<
                          std::setw(30) << securityGroup.GetVpcId() <<
                          std::setw(64) << securityGroup.GetDescription() <<
                          std::endl;
            }
        } else {
            std::cerr << "Failed to describe security groups:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return true;
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK para C++ da API.

`MonitorInstances`

O código de exemplo a seguir mostra como usar `MonitorInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Enable detailed monitoring for an Amazon Elastic Compute Cloud (Amazon EC2)
 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::enableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
 ec2Client.MonitorInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to enable monitoring on instance. A dry run
 should trigger an error."
                <<
                std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
                  instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                  std::endl;
        return false;
    }

    request.SetDryRun(false);
```

```
    Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
 ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
                  instanceId << ": " <<
                  monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully enabled monitoring on instance " <<
                  instanceId << std::endl;
    }

    return monitorInstancesOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte MonitorInstances em Referência AWS SDK para C++ da API.

`RebootInstances`

O código de exemplo a seguir mostra como usar `RebootInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::rebootInstance(const Aws::String &instanceId,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
```

```
    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
 ec2Client.RebootInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to reboot on instance. A dry run should trigger
 an error."
                <<
                std::endl;
        return false;
    } else if (dry_run_outcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to reboot instance " << instanceId << ": "
                  << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::RebootInstancesOutcome outcome =
 ec2Client.RebootInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to reboot instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully rebooted instance " << instanceId <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte RebootInstances em Referência AWS SDK para C++ da API.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Release an Elastic IP address.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::releaseAddress(const Aws::String &allocationID,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2(clientConfiguration);

    Aws::EC2::Model::ReleaseAddressRequest request;
    request.SetAllocationId(allocationID);

    Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to release Elastic IP address " <<
                  allocationID << ":" << outcome.GetError().GetMessage() <<
                  std::endl;
    } else {
        std::cout << "Successfully released Elastic IP address " <<
                  allocationID << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ReleaseAddress em Referência AWS SDK para C++ da API.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceName: A name for the EC2 instance.
  \param amiId: An Amazon Machine Image (AMI) identifier.
  \param[out] instanceID: String to return the instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::runInstance(const Aws::String &instanceName,
                              const Aws::String &amiId,
                              Aws::String &instanceID,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RunInstancesRequest runRequest;
    runRequest.SetImageId(amiId);
    runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    runRequest.SetMinCount(1);
    runRequest.SetMaxCount(1);

    Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
            runRequest);
    if (!runOutcome.IsSuccess()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
                  " based on ami " << amiId << ":" <<
                  runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const Aws::Vector<Aws::EC2::Model::Instance> &instances =
 runOutcome.GetResult().GetInstances();
    if (instances.empty()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
```

```
                  " based on ami " << amiId << ":" <<
                  runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    instanceID = instances[0].GetInstanceId();

    return true;
}
```

- Para obter detalhes da API, consulte RunInstances em Referência AWS SDK para C++ da API.

StartInstances

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::startInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest startRequest;
    startRequest.AddInstanceIds(instanceId);
    startRequest.SetDryRun(true);
```

```
    Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
 ec2Client.StartInstances(startRequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to start instance. A dry run should trigger an
 error."
                << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
    Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
 ec2Client.StartInstances(startRequest);

    if (!startInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
                  startInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully started instance " << instanceId <<
                  std::endl;
    }

    return startInstancesOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte StartInstances em Referência AWS SDK para C++ da API.

`StopInstances`

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Stop an EC2 instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::stopInstance(const Aws::String &instanceId,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
 ec2Client.StopInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to stop instance. A dry run should trigger an
 error."
                << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::StopInstancesOutcome outcome =
 ec2Client.StopInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully stopped instance " << instanceId <<
                    std::endl;
    }

    return outcome.IsSuccess();
}

void PrintUsage() {
    std::cout << "Usage: run_start_stop_instance <instance_id> <start|stop>" <<
              std::endl;
}
```

- Para obter detalhes da API, consulte StopInstances em Referência AWS SDK para C++ da API.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::terminateInstances(const Aws::String &instanceID,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
```

```
    Aws::EC2::Model::TerminateInstancesRequest request;
    request.SetInstanceIds({instanceID});

    Aws::EC2::Model::TerminateInstancesOutcome outcome =
            ec2Client.TerminateInstances(request);
    if (outcome.IsSuccess()) {
        std::cout << "Ec2 instance '" << instanceID <<
                  "' was terminated." << std::endl;
    } else {
        std::cerr << "Failed to terminate ec2 instance " << instanceID <<
                  ", " <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte TerminateInstances em Referência AWS SDK para C++ da API.

`UnmonitorInstances`

O código de exemplo a seguir mostra como usar `UnmonitorInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Disable monitoring for an EC2 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::disableMonitoring(const Aws::String &instanceId,
```

```
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
   Aws::EC2::EC2Client ec2Client(clientConfiguration);
   Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
   unrequest.AddInstanceIds(instanceId);
   unrequest.SetDryRun(true);

   Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
ec2Client.UnmonitorInstances(unrequest);
   if (dryRunOutcome.IsSuccess()) {
       std::cerr
               << "Failed dry run to disable monitoring on instance. A dry run
should trigger an error."
               <<
               std::endl;
       return false;
   } else if (dryRunOutcome.GetError().GetErrorType() !=
              Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
       std::cout << "Failed dry run to disable monitoring on instance " <<
                 instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                 std::endl;
       return false;
   }

   unrequest.SetDryRun(false);
   Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
ec2Client.UnmonitorInstances(unrequest);
   if (!unmonitorInstancesOutcome.IsSuccess()) {
       std::cout << "Failed to disable monitoring on instance " << instanceId
                 << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
                 std::endl;
   } else {
       std::cout << "Successfully disable monitoring on instance " <<
                 instanceId << std::endl;
   }

   return unmonitorInstancesOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UnmonitorInstances em Referência AWS SDK para C++ da API.

EventBridge exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with EventBridge.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Enviar o evento.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
        event_entry.SetDetail(MakeDetails(event_key, event_value));
        event_entry.SetDetailType("sampleSubmitted");
        event_entry.AddResources(resource_arn);
        event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

        Aws::CloudWatchEvents::Model::PutEventsRequest request;
        request.AddEntries(event_entry);

        auto outcome = cwe.PutEvents(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to post CloudWatch event: " <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully posted CloudWatch event" << std::endl;
        }
```

- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK para C++ da API.

`PutRule`

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
```

```
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Crie a regra.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;
        Aws::CloudWatchEvents::Model::PutRuleRequest request;
        request.SetName(rule_name);
        request.SetRoleArn(role_arn);
        request.SetScheduleExpression("rate(5 minutes)");
        request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

        auto outcome = cwe.PutRule(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events rule " <<
                rule_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch events rule " <<
                rule_name << " with resulting Arn " <<
                outcome.GetResult().GetRuleArn() << std::endl;
        }
```

- Para obter detalhes da API, consulte PutRule em Referência AWS SDK para C++ da API.

`PutTargets`

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inclua os arquivos necessários.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Adicione o destino.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::Target target;
        target.SetArn(lambda_arn);
        target.SetId(target_id);

        Aws::CloudWatchEvents::Model::PutTargetsRequest request;
        request.SetRule(rule_name);
        request.AddTargets(target);

        auto putTargetsOutcome = cwe.PutTargets(request);
        if (!putTargetsOutcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout <<
                "Successfully created CloudWatch events target for rule "
                << rule_name << std::endl;
```

```
        }
```

- Para obter detalhes da API, consulte PutTargets em Referência AWS SDK para C++ da API.

AWS Glue exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_glue.cpp)
```

```
target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_glue.cpp.

```
#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 *  A "Hello Glue" starter application which initializes an AWS Glue client and
 lists the
 *  AWS Glue job definitions.
 *
 *  main function
 *
 *  Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken;  // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
```

```
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = glueClient.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());

                nextToken = listRunsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Your account has " << jobs.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < jobs.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << jobs[i] << std::endl;
        }
    }
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para C++ da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.

- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Scenario which demonstrates using AWS Glue to add a crawler and run a job.
/*!
 \\sa runGettingStartedWithGlueScenario()
 \param bucketName: An S3 bucket created in the setup.
 \param roleName: An AWS Identity and Access Management (IAM) role created in the
 setup.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String &bucketName,
                                                     const Aws::String &roleName,
                                                     const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::Glue::GlueClient client(clientConfig);

    Aws::String roleArn;
    if (!getRoleArn(roleName, roleArn, clientConfig)) {
        std::cerr << "Error getting role ARN for role." << std::endl;
        return false;
    }

    // 1. Upload the job script to the S3 bucket.
    {
        std::cout << "Uploading the job script '"
                  << AwsDoc::Glue::PYTHON_SCRIPT
                  << "'." << std::endl;
```

```
        if (!AwsDoc::Glue::uploadFile(bucketName,
                                      AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                      AwsDoc::Glue::PYTHON_SCRIPT,
                                      clientConfig)) {
            std::cerr << "Error uploading the job file." << std::endl;
            return false;
        }
    }

    // 2. Create a crawler.
    {
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName, clientConfig);
            return false;
        }
    }

    // 3. Get a crawler.
    {
        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);
```

```
        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<
                      Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 4. Start a crawler.
    {
        Aws::Glue::Model::StartCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


        if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                    outcome.GetError().GetErrorType())) {
            if (!outcome.IsSuccess()) {
                std::cout << "Crawler was already started." << std::endl;
            }
            else {
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
```

```
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                    crawlerState)
                            << ". After " << iterations
                            << " seconds elapsed."
                            << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                            << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

```
    }

    // 5. Get a database.
    {
        Aws::Glue::Model::GetDatabaseRequest request;
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome = client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 6. Get tables.
    Aws::String tableName;
    {
        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        std::vector<Aws::Glue::Model::Table> all_tables;
        Aws::String nextToken; // Used for pagination.
        do {
            Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
                all_tables.insert(all_tables.end(), tables.begin(), tables.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting the tables. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " << all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex - 1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
    }

    // 7. Create a job.
    {
        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 8. Start a job run.
    {
        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName + "/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome = client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
```

```
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT)) {
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10 seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                         jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                         clientConfig);
            return false;
        }
    }

    // 9. List the output data stored in the S3 bucket.
    {
        Aws::S3::S3Client s3Client;
        Aws::S3::Model::ListObjectsV2Request request;
        request.SetBucket(bucketName);
        request.SetPrefix(OUTPUT_FILE_PREFIX);

        Aws::String continuationToken; // Used for pagination.
        std::vector<Aws::S3::Model::Object> allObjects;
        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = s3Client.ListObjectsV2(
                    request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::S3::Model::Object> &objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(), objects.end());
                continuationToken = outcome.GetResult().GetNextContinuationToken();
            }
            else {
                std::cerr << "Error listing objects. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!continuationToken.empty());

        std::cout << "Data from your job is in " << allObjects.size() <<
                  " files in the S3 bucket, " << bucketName << "." << std::endl;

        for (size_t i = 0; i < allObjects.size(); ++i) {
            std::cout << "    " << i + 1 << ". " << allObjects[i].GetKey()
                      << std::endl;
        }
```

```
        int objectIndex = askQuestionForIntRange(
                std::string(
                        "Enter the number of a block to download it and see the
first ") +
                std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
                " lines of JSON output in the block: ", 1,
                static_cast<int>(allObjects.size()));

        Aws::String objectKey = allObjects[objectIndex - 1].GetKey();

        std::stringstream stringStream;
        if (getObjectFromBucket(bucketName, objectKey, stringStream,
                                clientConfig)) {
            for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream; ++i) {
                std::string line;
                std::getline(stringStream, line);
                std::cout << "    " << line << std::endl;
            }
        }
        else {
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                         clientConfig);
            return false;
        }
    }

    // 10. List all the jobs.
    Aws::String jobName;
    {
        Aws::Glue::Model::ListJobsRequest listJobsRequest;

        Aws::String nextToken;
        std::vector<Aws::String> allJobNames;

        do {
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
```

```
                allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
            }
        } while (!nextToken.empty());
        std::cout << "Your account has " << allJobNames.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < allJobNames.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobNames[i] << std::endl;
        }
        int jobIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobNames.size()) +
                " to see the list of runs for a job: ",
                1, static_cast<int>(allJobNames.size()));

        jobName = allJobNames[jobIndex - 1];
    }

    // 11. Get the job runs for a job.
    Aws::String jobRunID;
    if (!jobName.empty()) {
        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(), jobRuns.end());
```

```
                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "There are " << allJobRuns.size() << " runs in the job '"
                  <<
                  jobName << "'." << std::endl;

        for (size_t i = 0; i < allJobRuns.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobRuns[i].GetJobName()
                      << std::endl;
        }

        int runIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobRuns.size()) +
                " to see details for a run: ",
                1, static_cast<int>(allJobRuns.size()));
        jobRunID = allJobRuns[runIndex - 1].GetId();
    }

    // 12. Get a single job run.
    if (!jobRunID.empty()) {
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
```

```
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                        clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
 \\sa deleteAssets()
 \param crawler: Name of an AWS Glue crawler.
 \param database: The name of an AWS Glue database.
 \param job: The name of an AWS Glue job.
 \param bucketName: The name of an S3 bucket.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
 &database,
                                const Aws::String &job, const Aws::String
 &bucketName,
                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::Glue::GlueClient client(clientConfig);
    bool result = true;

    // 13. Delete a job.
    if (!job.empty()) {
        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
```

```
            result = false;
        }
    }

    // 14. Delete a database.
    if (!database.empty()) {
        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    // 15. Delete a crawler.
    if (!crawler.empty()) {
        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    // 16. Delete the job script and run data from the S3 bucket.
    result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
                                                       clientConfig);
```

```
    return result;
}

//! Routine which uploads a file to an S3 bucket.
/*!
 \\sa uploadFile()
 \param bucketName: An S3 bucket created in the setup.
 \param filePath: The path of the file to upload.
 \param fileName The name for the uploaded file.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                         const Aws::String &filePath,
                         const Aws::String &fileName,
                         const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          filePath.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << filePath << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
```

```
        std::cout << "Added object '" << filePath << "' to bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
 \\sa deleteAllObjectsInS3Bucket()
 \param bucketName: The S3 bucket name.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                              const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsV2Request listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    bool result = true;
    do {
        if (!continuationToken.empty()) {
            listObjectsRequest.SetContinuationToken(continuationToken);
        }

        Aws::S3::Model::ListObjectsV2Outcome listObjectsOutcome =
 client.ListObjectsV2(
                listObjectsRequest);

        if (listObjectsOutcome.IsSuccess()) {
            const std::vector<Aws::S3::Model::Object> &objects =
 listObjectsOutcome.GetResult().GetContents();
            if (!objects.empty()) {
                Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
                deleteObjectsRequest.SetBucket(bucketName);

                std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
                for (const Aws::S3::Model::Object &object: objects) {
                    objectIdentifiers.push_back(
                            Aws::S3::Model::ObjectIdentifier().WithKey(
                                    object.GetKey()));
```

```
                }
                Aws::S3::Model::Delete objectsDelete;
                objectsDelete.SetObjects(objectIdentifiers);
                objectsDelete.SetQuiet(true);
                deleteObjectsRequest.SetDelete(objectsDelete);

                Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                        client.DeleteObjects(deleteObjectsRequest);

                if (!deleteObjectsOutcome.IsSuccess()) {
                    std::cerr << "Error deleting objects. " <<
                              deleteObjectsOutcome.GetError().GetMessage() <<
 std::endl;
                    result = false;
                    break;
                }
                else {
                    std::cout << "Successfully deleted the objects." << std::endl;

                }
            }
            else {
                std::cout << "No objects to delete in '" << bucketName << "'."
                          << std::endl;
            }

            continuationToken =
 listObjectsOutcome.GetResult().GetNextContinuationToken();
        }
        else {
            std::cerr << "Error listing objects. "
                      << listObjectsOutcome.GetError().GetMessage() << std::endl;
            result = false;
            break;
        }
    } while (!continuationToken.empty());

    return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
 \\sa getObjectFromBucket()
 \param bucketName: The S3 bucket name.
```

```
 \param objectKey: The object's name.
 \param objectStream: A stream to receive the retrieved data.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);


    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." << std::endl;
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
    }
    else {
        std::cerr << "Error retrieving object. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable

- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para C++

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);
```

```
        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName, clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte CreateCrawler em Referência AWS SDK para C++ da API.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK para C++ da API.

`DeleteCrawler`

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
```

- Para obter detalhes da API, consulte DeleteCrawler em Referência AWS SDK para C++ da API.

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Para obter detalhes da API, consulte DeleteDatabase em Referência AWS SDK para C++ da API.

`DeleteJob`

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Para obter detalhes da API, consulte DeleteJob em Referência AWS SDK para C++ da API.

GetCrawler

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);
```

```
        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<
                      Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte GetCrawler em Referência AWS SDK para C++ da API.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetDatabaseRequest request;
```

```
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome = client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte GetDatabase em Referência AWS SDK para C++ da API.

`GetJobRun`

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte GetJobRun em Referência AWS SDK para C++ da API.

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(), jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());
```

- Para obter detalhes da API, consulte GetJobRuns em Referência AWS SDK para C++ da API.

`GetTables`

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        std::vector<Aws::Glue::Model::Table> all_tables;
        Aws::String nextToken; // Used for pagination.
        do {
            Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
                all_tables.insert(all_tables.end(), tables.begin(), tables.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting the tables. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " << all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
```

```
            std::cout << all_tables[tableIndex - 1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
```

- Para obter detalhes da API, consulte GetTables em Referência AWS SDK para C++ da API.

ListJobs

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::ListJobsRequest listJobsRequest;

        Aws::String nextToken;
        std::vector<Aws::String> allJobNames;

        do {
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
```

```
                const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
                allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
            }
        } while (!nextToken.empty());
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para C++ da API.

`StartCrawler`

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::StartCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);
```

```
        if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                    outcome.GetError().GetErrorType())) {
            if (!outcome.IsSuccess()) {
                std::cout << "Crawler was already started." << std::endl;
            }
            else {
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                        crawlerState)
                                << ". After " << iterations
                                << " seconds elapsed."
                                << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                                << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }
```

```
            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte StartCrawler em Referência AWS SDK para C++ da API.

`StartJobRun`

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);
```

```
        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName + "/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome = client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT)) {
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
```

```
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10 seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                        jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " << outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                         clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte StartJobRun em Referência AWS SDK para C++ da API.

HealthImaging exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with HealthImaging.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá HealthImaging

O exemplo de código a seguir mostra como começar a usar o HealthImaging.

SDK para C++

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS medical-imaging)

# Set this project's name.
project("hello_health-imaging")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you may
 need to uncomment this
    # and set the proper subdirectory to the executable location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_health_imaging.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_health_imaging.cpp.

```
#include <aws/core/Aws.h>
#include <aws/medical-imaging/MedicalImagingClient.h>
#include <aws/medical-imaging/model/ListDatastoresRequest.h>

#include <iostream>

/*
 *  A "Hello HealthImaging" starter application which initializes an AWS
 HealthImaging (HealthImaging) client
 *  and lists the HealthImaging data stores in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_health-imaging'
 *
 */
```

```
#include <aws/core/auth/AWSCredentialsProviderChain.h>
#include <aws/core/platform/Environment.h>

int main(int argc, char **argv) {
    (void) argc;
    (void) argv;
    Aws::SDKOptions options;
    //   Optional: change the log level for debugging.
    //   options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;

    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::MedicalImaging::MedicalImagingClient
 medicalImagingClient(clientConfig);
        Aws::MedicalImaging::Model::ListDatastoresRequest listDatastoresRequest;

        Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
 allDataStoreSummaries;
        Aws::String nextToken; // Used for paginated results.
        do {
            if (!nextToken.empty()) {
                listDatastoresRequest.SetNextToken(nextToken);
            }
            Aws::MedicalImaging::Model::ListDatastoresOutcome listDatastoresOutcome
 =
                    medicalImagingClient.ListDatastores(listDatastoresRequest);
            if (listDatastoresOutcome.IsSuccess()) {
                const Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
 &dataStoreSummaries =
                        listDatastoresOutcome.GetResult().GetDatastoreSummaries();
                allDataStoreSummaries.insert(allDataStoreSummaries.cend(),
                                             dataStoreSummaries.cbegin(),
                                             dataStoreSummaries.cend());
                nextToken = listDatastoresOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "ListDatastores error: "
                          << listDatastoresOutcome.GetError().GetMessage() <<
 std::endl;
                break;
```

```
            }
        } while (!nextToken.empty());

        std::cout << allDataStoreSummaries.size() << " HealthImaging data "
                  << ((allDataStoreSummaries.size() == 1) ?
                      "store was retrieved." : "stores were retrieved.") <<
 std::endl;

        for (auto const &dataStoreSummary: allDataStoreSummaries) {
            std::cout << "  Datastore: " << dataStoreSummary.GetDatastoreName()
                      << std::endl;
            std::cout << "  Datastore ID: " << dataStoreSummary.GetDatastoreId()
                      << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListDatastores em Referência AWS SDK para C++ da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Ações

`DeleteImageSet`

O código de exemplo a seguir mostra como usar `DeleteImageSet`.

SDK para C++

```
//! Routine which deletes an AWS HealthImaging image set.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
```

```
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::deleteImageSet(
        const Aws::String &dataStoreID, const Aws::String &imageSetID,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::DeleteImageSetRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    Aws::MedicalImaging::Model::DeleteImageSetOutcome outcome =
 client.DeleteImageSet(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted image set " << imageSetID
                  << " from data store " << dataStoreID << std::endl;
    }
    else {
        std::cerr << "Error deleting image set " << imageSetID << " from data store
 "
                  << dataStoreID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteImageSet em Referência AWS SDK para C++ da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDICOMImportJob`

O código de exemplo a seguir mostra como usar `GetDICOMImportJob`.

SDK para C++

```
//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

- Para obter detalhes da API, consulte GetDICOMImportJob em Referência AWS SDK para C++ da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageFrame`

O código de exemplo a seguir mostra como usar `GetImageFrame`.

SDK para C++

```
//! Routine which downloads an AWS HealthImaging image frame.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param frameID: The image frame ID.
  \param jphFile: File to store the downloaded frame.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageFrame(const Aws::String &dataStoreID,
                                            const Aws::String &imageSetID,
                                            const Aws::String &frameID,
                                            const Aws::String &jphFile,
                                            const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);

    Aws::MedicalImaging::Model::GetImageFrameRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);

    Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
    imageFrameInformation.SetImageFrameId(frameID);
    request.SetImageFrameInformation(imageFrameInformation);

    Aws::MedicalImaging::Model::GetImageFrameOutcome outcome = client.GetImageFrame(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved image frame." << std::endl;
        auto &buffer = outcome.GetResult().GetImageFrameBlob();

        std::ofstream outfile(jphFile, std::ios::binary);
        outfile << buffer.rdbuf();
    }
    else {
        std::cout << "Error retrieving image frame." <<
 outcome.GetError().GetMessage()
                  << std::endl;

    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetImageFrame em Referência AWS SDK para C++ da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSetMetadata`

O código de exemplo a seguir mostra como usar `GetImageSetMetadata`.

SDK para C++

Função de utilitário para obter metadados do conjunto de imagens.

```
//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
```

```
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Obter metadados do conjunto de imagens sem versão.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
 "", outputFilePath, clientConfig))
        {
            std::cout << "Successfully retrieved image set metadata." << std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

Obter metadados do conjunto de imagens com versão.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
 versionID, outputFilePath, clientConfig))
        {
            std::cout << "Successfully retrieved image set metadata." << std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

- Para obter detalhes da API, consulte GetImageSetMetadata em Referência AWS SDK para C ++ da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`SearchImageSets`

O código de exemplo a seguir mostra como usar `SearchImageSets`.

SDK para C++

A função de utilitário para pesquisar conjuntos de imagens.

```
//! Routine which searches for image sets based on defined input attributes.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param searchCriteria: A search criteria instance.
  \param imageSetResults: Vector to receive the image set IDs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::searchImageSets(const Aws::String &dataStoreID,
                                              const
 Aws::MedicalImaging::Model::SearchCriteria &searchCriteria,
                                              Aws::Vector<Aws::String>
 &imageSetResults,
                                              const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::SearchImageSetsRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetSearchCriteria(searchCriteria);

    Aws::String nextToken; // Used for paginated results.
    bool result = true;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::MedicalImaging::Model::SearchImageSetsOutcome outcome =
 client.SearchImageSets(
```

```
                request);
        if (outcome.IsSuccess()) {
            for (auto &imageSetMetadataSummary:
 outcome.GetResult().GetImageSetsMetadataSummaries()) {
                imageSetResults.push_back(imageSetMetadataSummary.GetImageSetId());
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    } while (!nextToken.empty());

    return result;
}
```

Caso de uso nº 1: operador EQUAL.

```
        Aws::Vector<Aws::String> imageIDsForPatientID;
        Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
        Aws::Vector<Aws::MedicalImaging::Model::SearchFilter> patientIDSearchFilters
 = {

 Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Operato

 .WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(patien
        };

        searchCriteriaEqualsPatientID.SetFilters(patientIDSearchFilters);
        bool result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

 searchCriteriaEqualsPatientID,
                                                                imageIDsForPatientID,
                                                                clientConfig);
        if (result) {
            std::cout << imageIDsForPatientID.size() << " image sets found for the
 patient with ID '"
            <<  patientID << "'." << std::endl;
            for (auto &imageSetResult : imageIDsForPatientID) {
                std::cout << "  Image set with ID '" << imageSetResult << std::endl;
```

```
            }
        }
```

Caso de uso #2: operador BETWEEN usando DICOMStudyDate DICOMStudyTime e.

```
         Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

 useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTim
        .WithDICOMStudyDate("19990101")
        .WithDICOMStudyTime("000000.000"));

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

 useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTime(

 .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeStrin
%m%d"))
        .WithDICOMStudyTime("000000.000"));

        Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
        useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

 useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;
        useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

        Aws::Vector<Aws::String> usesCase2Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase2SearchCriteria,
                                                          usesCase2Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase2Results.size() << " image sets found for between
 1999/01/01 and present."
                      <<  std::endl;
            for (auto &imageSetResult : usesCase2Results) {
                std::cout << "  Image set with ID '" << imageSetResult << std::endl;
            }
        }
```

Caso de uso nº 3: operador BETWEEN usando o createdAt. Os estudos de tempo foram previamente persistidos.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase3StartDate;

useCase3StartDate.SetCreatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::DateF

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase3EndDate;

useCase3EndDate.SetCreatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

        Aws::MedicalImaging::Model::SearchFilter useCase3SearchFilter;
        useCase3SearchFilter.SetValues({useCase3StartDate, useCase3EndDate});

useCase3SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchCriteria useCase3SearchCriteria;
        useCase3SearchCriteria.SetFilters({useCase3SearchFilter});

        Aws::Vector<Aws::String> usesCase3Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase3SearchCriteria,
                                                          usesCase3Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase3Results.size() << " image sets found for created
between 2023/11/30 and present."
                      <<  std::endl;
            for (auto &imageSetResult : usesCase3Results) {
                std::cout << "  Image set with ID '" << imageSetResult << std::endl;
            }
        }
```

Caso de uso #4: operador EQUAL on DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifique a resposta na ordem ASC no campo updatedAt.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase4StartDate;

useCase4StartDate.SetUpdatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::DateF

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase4EndDate;
```

```
useCase4EndDate.SetUpdatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

       Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterBetween;
       useCase4SearchFilterBetween.SetValues({useCase4StartDate, useCase4EndDate});

useCase4SearchFilterBetween.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

       Aws::MedicalImaging::Model::SearchByAttributeValue seriesInstanceUID;
       seriesInstanceUID.SetDICOMSeriesInstanceUID(dicomSeriesInstanceUID);

       Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterEqual;
       useCase4SearchFilterEqual.SetValues({seriesInstanceUID});

useCase4SearchFilterEqual.SetOperator(Aws::MedicalImaging::Model::Operator::EQUAL);

       Aws::MedicalImaging::Model::SearchCriteria useCase4SearchCriteria;
       useCase4SearchCriteria.SetFilters({useCase4SearchFilterBetween,
useCase4SearchFilterEqual});

       Aws::MedicalImaging::Model::Sort useCase4Sort;
       useCase4Sort.SetSortField(Aws::MedicalImaging::Model::SortField::updatedAt);
       useCase4Sort.SetSortOrder(Aws::MedicalImaging::Model::SortOrder::ASC);

       useCase4SearchCriteria.SetSort(useCase4Sort);

       Aws::Vector<Aws::String> usesCase4Results;
       result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                         useCase4SearchCriteria,
                                                         usesCase4Results,
                                                         clientConfig);
       if (result) {
           std::cout << usesCase4Results.size() << " image sets found for EQUAL
operator "
           << "on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response
\n"
           <<  "in ASC order on updatedAt field." <<  std::endl;
           for (auto &imageSetResult : usesCase4Results) {
               std::cout << "  Image set with ID '" << imageSetResult << std::endl;
           }
       }
```

- Para obter detalhes da API, consulte SearchImageSets em Referência AWS SDK para C++ da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

StartDICOMImportJob

O código de exemplo a seguir mostra como usar `StartDICOMImportJob`.

SDK para C++

```
//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory + "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
```

```
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte StartDICOMImportJob em Referência AWS SDK para C ++ da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cenários

Começar a usar conjuntos de imagens e quadros de imagem

O exemplo de código a seguir mostra como importar arquivos DICOM e baixar quadros de imagem em HealthImaging.

A implementação é estruturada como uma aplicação da linha de comando.

- Configure recursos para uma importação DICOM.
- Importe arquivos DICOM para um armazenamento de dados.
- Recupere os IDs do conjunto de imagens para o trabalho de importação.

- Recupere os IDs do quadro de imagem para o conjunto de imagens.
- Baixe, decodifique e verifique os quadros de imagem.
- Limpe recursos.

SDK para C++

Crie uma CloudFormation pilha com os recursos necessários.

```
   Aws::String inputBucketName;
   Aws::String outputBucketName;
   Aws::String dataStoreId;
   Aws::String roleArn;
   Aws::String stackName;

   if (askYesNoQuestion(
           "Would you like to let this workflow create the resources for you? (y/n)
")) {
       stackName = askQuestion(
               "Enter a name for the AWS CloudFormation stack to create. ");
       Aws::String dataStoreName = askQuestion(
               "Enter a name for the HealthImaging datastore to create. ");

       Aws::Map<Aws::String, Aws::String> outputs = createCloudFormationStack(
               stackName,
               dataStoreName,
               clientConfiguration);

       if (!retrieveOutputs(outputs, dataStoreId, inputBucketName,
outputBucketName,
                            roleArn)) {
           return false;
       }

       std::cout << "The following resources have been created." << std::endl;
       std::cout << "A HealthImaging datastore with ID: " << dataStoreId << "."
                 << std::endl;
       std::cout << "An Amazon S3 input bucket named: " << inputBucketName << "."
                 << std::endl;
       std::cout << "An Amazon S3 output bucket named: " << outputBucketName << "."
                 << std::endl;
       std::cout << "An IAM role with the ARN: " << roleArn << "." << std::endl;
       askQuestion("Enter return to continue.", alwaysTrueTest);
```

```
    }
    else {
        std::cout << "You have chosen to use preexisting resources:" << std::endl;
        dataStoreId = askQuestion(
                "Enter the data store ID of the HealthImaging datastore you wish to
use: ");
        inputBucketName = askQuestion(
                "Enter the name of the S3 input bucket you wish to use: ");
        outputBucketName = askQuestion(
                "Enter the name of the S3 output bucket you wish to use: ");
        roleArn = askQuestion(
                "Enter the ARN for the IAM role with the proper permissions to
import a DICOM series: ");
    }
```

Copie arquivos DICOM para o bucket de importação do Amazon S3.

```
    std::cout
            << "This workflow uses DICOM files from the National Cancer Institute
Imaging Data\n"
            << "Commons (IDC) Collections." << std::endl;
    std::cout << "Here is the link to their website." << std::endl;
    std::cout << "https://registry.opendata.aws/nci-imaging-data-commons/" <<
std::endl;
    std::cout << "We will use DICOM files stored in an S3 bucket managed by the
IDC."
              << std::endl;
    std::cout
            << "First one of the DICOM folders in the IDC collection must be copied
to your\n"
               "input S3 bucket."
            << std::endl;
    std::cout << "You have the choice of one of the following "
              << IDC_ImageChoices.size() << " folders to copy." << std::endl;

    int index = 1;
    for (auto &idcChoice: IDC_ImageChoices) {
        std::cout << index << " - " << idcChoice.mDescription << std::endl;
        index++;
    }
    int choice = askQuestionForIntRange("Choose DICOM files to import: ", 1, 4);
```

```
    Aws::String fromDirectory = IDC_ImageChoices[choice - 1].mDirectory;
    Aws::String inputDirectory = "input";

    std::cout << "The files in the directory '" << fromDirectory << "' in the bucket
 '"
              << IDC_S3_BucketName << "' will be copied " << std::endl;
    std::cout << "to the folder '" << inputDirectory << "/" << fromDirectory
              << "' in the bucket '" << inputBucketName << "'." << std::endl;
    askQuestion("Enter return to start the copy.", alwaysTrueTest);

    if (!AwsDoc::Medical_Imaging::copySeriesBetweenBuckets(
            IDC_S3_BucketName,
            fromDirectory,
            inputBucketName,
            inputDirectory, clientConfiguration)) {
        std::cerr << "This workflow will exit because of an error." << std::endl;
        cleanup(stackName, dataStoreId, clientConfiguration);
        return false;
    }
```

Importe os arquivos DICOM para o armazenamento de dados do Amazon S3.

```
bool AwsDoc::Medical_Imaging::startDicomImport(const Aws::String &dataStoreID,
                                               const Aws::String &inputBucketName,
                                               const Aws::String &inputDirectory,
                                               const Aws::String &outputBucketName,
                                               const Aws::String &outputDirectory,
                                               const Aws::String &roleArn,
                                               Aws::String &importJobId,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = false;
    if (startDICOMImportJob(dataStoreID, inputBucketName, inputDirectory,
                            outputBucketName, outputDirectory, roleArn, importJobId,
                            clientConfiguration)) {
        std::cout << "DICOM import job started with job ID " << importJobId << "."
                  << std::endl;
        result = waitImportJobCompleted(dataStoreID, importJobId,
 clientConfiguration);
        if (result) {
            std::cout << "DICOM import job completed." << std::endl;
```

```
        }
    }

    return result;
}

//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory + "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
```

```
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}


//! Routine which waits for a DICOM import job to complete.
/*!
 * @param dataStoreID: The HealthImaging data store ID.
 * @param importJobId: The import job ID.
 * @param clientConfiguration : Aws client configuration.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::waitImportJobCompleted(const Aws::String &datastoreID,
                                                     const Aws::String &importJobId,
                                                     const
 Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::JobStatus jobStatus =
 Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS;
    while (jobStatus == Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS) {
        std::this_thread::sleep_for(std::chrono::seconds(1));

        Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
 getDicomImportJobOutcome = getDICOMImportJob(
                datastoreID, importJobId,
                clientConfiguration);

        if (getDicomImportJobOutcome.IsSuccess()) {
            jobStatus =
 getDicomImportJobOutcome.GetResult().GetJobProperties().GetJobStatus();

            std::cout << "DICOM import job status: " <<

 Aws::MedicalImaging::Model::JobStatusMapper::GetNameForJobStatus(
                              jobStatus) << std::endl;
        }
        else {
            std::cerr << "Failed to get import job status because "
                      << getDicomImportJobOutcome.GetError().GetMessage() <<
 std::endl;
```

```
            return false;
        }
    }

    return jobStatus == Aws::MedicalImaging::Model::JobStatus::COMPLETED;
}

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

Obtenha conjuntos de imagens criados pelo trabalho de importação DICOM.

```
bool
AwsDoc::Medical_Imaging::getImageSetsForDicomImportJob(const Aws::String
 &datastoreID,
                                                       const Aws::String
 &importJobId,
```

```
                                                        Aws::Vector<Aws::String>
&imageSets,
                                                        const
Aws::Client::ClientConfiguration &clientConfiguration) {
   Aws::MedicalImaging::Model::GetDICOMImportJobOutcome getDicomImportJobOutcome =
getDICOMImportJob(
           datastoreID, importJobId, clientConfiguration);
   bool result = false;
   if (getDicomImportJobOutcome.IsSuccess()) {
       auto outputURI =
getDicomImportJobOutcome.GetResult().GetJobProperties().GetOutputS3Uri();
       Aws::Http::URI uri(outputURI);
       const Aws::String &bucket = uri.GetAuthority();
       Aws::String key = uri.GetPath();

       Aws::S3::S3Client s3Client(clientConfiguration);
       Aws::S3::Model::GetObjectRequest objectRequest;
       objectRequest.SetBucket(bucket);
       objectRequest.SetKey(key + "/" + IMPORT_JOB_MANIFEST_FILE_NAME);

       auto getObjectOutcome = s3Client.GetObject(objectRequest);
       if (getObjectOutcome.IsSuccess()) {
           auto &data = getObjectOutcome.GetResult().GetBody();

           std::stringstream stringStream;
           stringStream << data.rdbuf();

           try {
               // Use JMESPath to extract the image set IDs.
               // https://jmespath.org/specification.html
               std::string jmesPathExpression =
"jobSummary.imageSetsSummary[].imageSetId";
               jsoncons::json doc = jsoncons::json::parse(stringStream.str());

               jsoncons::json imageSetsJson = jsoncons::jmespath::search(doc,

jmesPathExpression);\
               for (auto &imageSet: imageSetsJson.array_range()) {
                   imageSets.push_back(imageSet.as_string());
               }

               result = true;
           }
           catch (const std::exception &e) {
```

```
                std::cerr << e.what() << '\n';
            }

        }
        else {
            std::cerr << "Failed to get object because "
                      << getObjectOutcome.GetError().GetMessage() << std::endl;
        }

    }
    else {
        std::cerr << "Failed to get import job status because "
                  << getDicomImportJobOutcome.GetError().GetMessage() << std::endl;
    }

    return result;
}
```

Obtenha informações sobre os quadros de imagem de conjuntos de imagens.

```
bool AwsDoc::Medical_Imaging::getImageFramesForImageSet(const Aws::String
 &dataStoreID,
                                                         const Aws::String
 &imageSetID,
                                                         const Aws::String
 &outDirectory,
                                                         Aws::Vector<ImageFrameInfo>
 &imageFrames,
                                                         const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::String fileName = outDirectory + "/" + imageSetID + "_metadata.json.gzip";
    bool result = false;
    if (getImageSetMetadata(dataStoreID, imageSetID, "", // Empty string for version
 ID.
                            fileName, clientConfiguration)) {
        try {
            std::string metadataGZip;
            {
                std::ifstream inFileStream(fileName.c_str(), std::ios::binary);
                if (!inFileStream) {
                    throw std::runtime_error("Failed to open file " + fileName);
                }
```

```
                std::stringstream stringStream;
                stringStream << inFileStream.rdbuf();
                metadataGZip = stringStream.str();
            }
            std::string metadataJson = gzip::decompress(metadataGZip.data(),
                                                        metadataGZip.size());
            // Use JMESPath to extract the image set IDs.
            // https://jmespath.org/specification.html
            jsoncons::json doc = jsoncons::json::parse(metadataJson);
            std::string jmesPathExpression = "Study.Series.*.Instances[].*[]";
            jsoncons::json instances = jsoncons::jmespath::search(doc,

jmesPathExpression);
            for (auto &instance: instances.array_range()) {
                jmesPathExpression = "DICOM.RescaleSlope";
                std::string rescaleSlope = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();
                jmesPathExpression = "DICOM.RescaleIntercept";
                std::string rescaleIntercept = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();

                jmesPathExpression = "ImageFrames[][]";
                jsoncons::json imageFramesJson =
jsoncons::jmespath::search(instance,

jmesPathExpression);

                for (auto &imageFrame: imageFramesJson.array_range()) {
                    ImageFrameInfo imageFrameIDs;
                    imageFrameIDs.mImageSetId = imageSetID;
                    imageFrameIDs.mImageFrameId = imageFrame.find(
                            "ID")->value().as_string();
                    imageFrameIDs.mRescaleIntercept = rescaleIntercept;
                    imageFrameIDs.mRescaleSlope = rescaleSlope;
                    imageFrameIDs.MinPixelValue = imageFrame.find(
                            "MinPixelValue")->value().as_string();
                    imageFrameIDs.MaxPixelValue = imageFrame.find(
                            "MaxPixelValue")->value().as_string();

                    jmesPathExpression =
"max_by(PixelDataChecksumFromBaseToFullResolution, &Width).Checksum";
```

```
                    jsoncons::json checksumJson =
 jsoncons::jmespath::search(imageFrame,

 jmesPathExpression);
                    imageFrameIDs.mFullResolutionChecksum =
 checksumJson.as_integer<uint32_t>();

                    imageFrames.emplace_back(imageFrameIDs);
                }
            }

            result = true;
        }
        catch (const std::exception &e) {
            std::cerr << "getImageFramesForImageSet failed because " << e.what()
                      << std::endl;
        }
    }

    return result;
}

//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
```

```
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Baixe, decodifique e verifique os quadros de imagem.

```
bool AwsDoc::Medical_Imaging::downloadDecodeAndCheckImageFrames(
        const Aws::String &dataStoreID,
        const Aws::Vector<ImageFrameInfo> &imageFrames,
        const Aws::String &outDirectory,
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::Client::ClientConfiguration clientConfiguration1(clientConfiguration);
    clientConfiguration1.executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(
            clientConfiguration1);

    Aws::Utils::Threading::Semaphore semaphore(0, 1);
    std::atomic<size_t> count(imageFrames.size());

    bool result = true;
    for (auto &imageFrame: imageFrames) {
        Aws::MedicalImaging::Model::GetImageFrameRequest getImageFrameRequest;
        getImageFrameRequest.SetDatastoreId(dataStoreID);
        getImageFrameRequest.SetImageSetId(imageFrame.mImageSetId);

        Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
        imageFrameInformation.SetImageFrameId(imageFrame.mImageFrameId);
```

```
        getImageFrameRequest.SetImageFrameInformation(imageFrameInformation);

        auto getImageFrameAsyncLambda = [&semaphore, &result, &count, imageFrame,
 outDirectory](
                const Aws::MedicalImaging::MedicalImagingClient *client,
                const Aws::MedicalImaging::Model::GetImageFrameRequest &request,
                Aws::MedicalImaging::Model::GetImageFrameOutcome outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext>
 &context) {

                if (!handleGetImageFrameResult(outcome, outDirectory, imageFrame)) {
                    std::cerr << "Failed to download and convert image frame: "
                              << imageFrame.mImageFrameId << " from image set: "
                              << imageFrame.mImageSetId << std::endl;
                    result = false;
                }

                count--;
                if (count <= 0) {

                    semaphore.ReleaseAll();
                }
        }; // End of 'getImageFrameAsyncLambda' lambda.

        medicalImagingClient.GetImageFrameAsync(getImageFrameRequest,
                                                getImageFrameAsyncLambda);
    }

    if (count > 0) {
        semaphore.WaitOne();
    }

    if (result) {
        std::cout << imageFrames.size() << " image files were downloaded."
                  << std::endl;
    }

    return result;
}

bool AwsDoc::Medical_Imaging::decodeJPHFileAndValidateWithChecksum(
        const Aws::String &jphFile,
        uint32_t crc32Checksum) {
    opj_image_t *outputImage = jphImageToOpjBitmap(jphFile);
```

```
    if (!outputImage) {
        return false;
    }

    bool result = true;
    if (!verifyChecksumForImage(outputImage, crc32Checksum)) {
        std::cerr << "The checksum for the image does not match the expected value."
                  << std::endl;
        std::cerr << "File :" << jphFile << std::endl;
        result = false;
    }

    opj_image_destroy(outputImage);

    return result;
}

opj_image *
AwsDoc::Medical_Imaging::jphImageToOpjBitmap(const Aws::String &jphFile) {
    opj_stream_t *inFileStream = nullptr;
    opj_codec_t *decompressorCodec = nullptr;
    opj_image_t *outputImage = nullptr;
    try {
        std::shared_ptr<opj_dparameters> decodeParameters =
 std::make_shared<opj_dparameters>();
        memset(decodeParameters.get(), 0, sizeof(opj_dparameters));

        opj_set_default_decoder_parameters(decodeParameters.get());

        decodeParameters->decod_format = 1; // JP2 image format.
        decodeParameters->cod_format = 2; // BMP image format.

        std::strncpy(decodeParameters->infile, jphFile.c_str(),
                     OPJ_PATH_LEN);

        inFileStream = opj_stream_create_default_file_stream(
                decodeParameters->infile, true);
        if (!inFileStream) {
            throw std::runtime_error(
                    "Unable to create input file stream for file '" + jphFile +
 "'.");
        }

        decompressorCodec = opj_create_decompress(OPJ_CODEC_JP2);
```

```
        if (!decompressorCodec) {
            throw std::runtime_error("Failed to create decompression codec.");
        }

        int decodeMessageLevel = 1;
        if (!setupCodecLogging(decompressorCodec, &decodeMessageLevel)) {
            std::cerr << "Failed to setup codec logging." << std::endl;
        }

        if (!opj_setup_decoder(decompressorCodec, decodeParameters.get())) {
            throw std::runtime_error("Failed to setup decompression codec.");
        }
        if (!opj_codec_set_threads(decompressorCodec, 4)) {
            throw std::runtime_error("Failed to set decompression codec threads.");
        }

        if (!opj_read_header(inFileStream, decompressorCodec, &outputImage)) {
            throw std::runtime_error("Failed to read header.");
        }

        if (!opj_decode(decompressorCodec, inFileStream,
                        outputImage)) {
            throw std::runtime_error("Failed to decode.");
        }

        if (DEBUGGING) {
            std::cout << "image width : " << outputImage->x1 - outputImage->x0
                      << std::endl;
            std::cout << "image height : " << outputImage->y1 - outputImage->y0
                      << std::endl;
            std::cout << "number of channels: " << outputImage->numcomps
                      << std::endl;
            std::cout << "colorspace : " << outputImage->color_space << std::endl;
        }

    } catch (const std::exception &e) {
        std::cerr << e.what() << std::endl;
        if (outputImage) {
            opj_image_destroy(outputImage);
            outputImage = nullptr;
        }
    }
    if (inFileStream) {
        opj_stream_destroy(inFileStream);
```

```
    }
    if (decompressorCodec) {
        opj_destroy_codec(decompressorCodec);
    }

    return outputImage;
}

//! Template function which converts a planar image bitmap to an interleaved image
 bitmap and
//! then verifies the checksum of the bitmap.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
template<class myType>
bool verifyChecksumForImageForType(opj_image_t *image, uint32_t crc32Checksum) {
    uint32_t width = image->x1 - image->x0;
    uint32_t height = image->y1 - image->y0;
    uint32_t numOfChannels = image->numcomps;

    // Buffer for interleaved bitmap.
    std::vector<myType> buffer(width * height * numOfChannels);

    // Convert planar bitmap to interleaved bitmap.
    for (uint32_t channel = 0; channel < numOfChannels; channel++) {
        for (uint32_t row = 0; row < height; row++) {
            uint32_t fromRowStart = row / image->comps[channel].dy * width /
                                    image->comps[channel].dx;
            uint32_t toIndex = (row * width) * numOfChannels + channel;

            for (uint32_t col = 0; col < width; col++) {
                uint32_t fromIndex = fromRowStart + col / image->comps[channel].dx;

                buffer[toIndex] = static_cast<myType>(image-
>comps[channel].data[fromIndex]);

                toIndex += numOfChannels;
            }
        }
    }

    // Verify checksum.
```

```
    boost::crc_32_type crc32;
    crc32.process_bytes(reinterpret_cast<char *>(buffer.data()),
                        buffer.size() * sizeof(myType));

    bool result = crc32.checksum() == crc32Checksum;
    if (!result) {
        std::cerr << "verifyChecksumForImage, checksum mismatch, expected - "
                  << crc32Checksum << ", actual - " << crc32.checksum()
                  << std::endl;
    }

    return result;
}

//! Routine which verifies the checksum of an OpenJPEG image struct.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::verifyChecksumForImage(opj_image_t *image,
                                                     uint32_t crc32Checksum) {
    uint32_t channels = image->numcomps;
    bool result = false;
    if (0 < channels) {
        // Assume the precision is the same for all channels.
        uint32_t precision = image->comps[0].prec;
        bool signedData = image->comps[0].sgnd;
        uint32_t bytes = (precision + 7) / 8;

        if (signedData) {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<int8_t>(image,
                                                                   crc32Checksum);
                    break;
                case 2 :
                    result = verifyChecksumForImageForType<int16_t>(image,
                                                                    crc32Checksum);
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<int32_t>(image,
                                                                    crc32Checksum);
                    break;
```

```
                default:
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
 signed bytes - "
                            << bytes << std::endl;
                    break;
            }
        }
        else {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<uint8_t>(image,
                                                                    crc32Checksum);
                    break;
                case 2 :
                    result = verifyChecksumForImageForType<uint16_t>(image,
                                                                     crc32Checksum);
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<uint32_t>(image,
                                                                     crc32Checksum);
                    break;
                default:
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
 unsigned bytes - "
                            << bytes << std::endl;
                    break;
            }
        }

        if (!result) {
            std::cerr << "verifyChecksumForImage, error bytes " << bytes
                      << " signed "
                      << signedData << std::endl;
        }
    }
    else {
        std::cerr << "'verifyChecksumForImage', no channels in the image."
                  << std::endl;
    }
    return result;
}
```

Limpe recursos.

```
bool AwsDoc::Medical_Imaging::cleanup(const Aws::String &stackName,
                                      const Aws::String &dataStoreId,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    bool result = true;

    if (!stackName.empty() && askYesNoQuestion(
            "Would you like to delete the stack " + stackName + "? (y/n)")) {
        std::cout << "Deleting the image sets in the stack." << std::endl;
        result &= emptyDatastore(dataStoreId, clientConfiguration);
        printAsterisksLine();
        std::cout << "Deleting the stack." << std::endl;
        result &= deleteStack(stackName, clientConfiguration);
    }
    return result;
}

bool AwsDoc::Medical_Imaging::emptyDatastore(const Aws::String &datastoreID,
                                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {

    Aws::MedicalImaging::Model::SearchCriteria emptyCriteria;
    Aws::Vector<Aws::String> imageSetIDs;
    bool result = false;
    if (searchImageSets(datastoreID, emptyCriteria, imageSetIDs,
                        clientConfiguration)) {
        result = true;
        for (auto &imageSetID: imageSetIDs) {
            result &= deleteImageSet(datastoreID, imageSetID, clientConfiguration);
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .

- DeleteImageSet
- GetDICOMImportJob
- GetImageFrame
- GetImageSetMetadata
- SearchImageSets
- StartDICOMImportJob

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exemplos do IAM usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem iam.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 *  A "Hello IAM" starter application which initializes an AWS Identity and Access
 Management (IAM) client
 *  and lists the IAM policies.
 *
 *  main function
 *
 *  Usage: 'hello_iam'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
```

```
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
                          std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                          std::setw(64) << "Description" << std::setw(12) <<
                          "CreateDate" << std::endl;
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                          policy.GetPolicyName() << std::setw(30) <<
                          policy.GetPolicyId() << std::setw(80) << policy.GetArn()
<<
                          std::setw(64) << policy.GetDescription() << std::setw(12)
<<
                          policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
                          std::endl;
            }

            if (outcome.GetResult().GetIsTruncated()) {
                request.SetMarker(outcome.GetResult().GetMarker());
            } else {
                done = true;
```

```
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para C++ da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para C++

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
```

```
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                                Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName << std::endl;
        }

        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;
```

```
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.
        request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }
```

```
    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome = client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " << policyName
<<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }
```

```
    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);

        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
```

```
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                          listBucketsOutcome.GetError().GetMessage() << std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome = client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
```

```
                        outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." << std::endl;
        }
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() << std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
```

```
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
 client.DetachRolePolicy(
                    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
 std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
 client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
```

```
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
        }

    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
```

```
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
```

```
                                    const Aws::String &policyArn,
                                    const Aws::Client::ClientConfiguration
&clientConfig) {
   Aws::IAM::IAMClient iam(clientConfig);

   Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
   list_request.SetRoleName(roleName);

   bool done = false;
   while (!done) {
       auto list_outcome = iam.ListAttachedRolePolicies(list_request);
       if (!list_outcome.IsSuccess()) {
           std::cerr << "Failed to list attached policies of role " <<
                     roleName << ": " << list_outcome.GetError().GetMessage() <<
                     std::endl;
           return false;
       }

       const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
       if (std::any_of(policies.cbegin(), policies.cend(),
                       [=](const Aws::IAM::Model::AttachedPolicy &policy) {
                               return policy.GetPolicyArn() == policyArn;
                       })) {
           std::cout << "Policy " << policyArn <<
                     " is already attached to role " << roleName << std::endl;
           return true;
       }

       done = !list_outcome.GetResult().GetIsTruncated();
       list_request.SetMarker(list_outcome.GetResult().GetMarker());
   }

   Aws::IAM::Model::AttachRolePolicyRequest request;
   request.SetRoleName(roleName);
   request.SetPolicyArn(policyArn);

   Aws::IAM::Model::AttachRolePolicyOutcome outcome =
iam.AttachRolePolicy(request);
   if (!outcome.IsSuccess()) {
       std::cerr << "Failed to attach policy " << policyArn << " to role " <<
                 roleName << ": " << outcome.GetError().GetMessage() << std::endl;
   }
   else {
       std::cout << "Successfully attached policy " << policyArn << " to role " <<
```

```
                    roleName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AttachRolePolicy em Referência AWS SDK para C++ da API.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
    Aws::IAM::Model::CreateAccessKeyOutcome outcome = iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
        std::cout << "Successfully created access key for IAM user " <<
                  userName << std::endl << "  aws_access_key_id = " <<
                  accessKey.GetAccessKeyId() << std::endl <<
```

```
                  " aws_secret_access_key = " << accessKey.GetSecretAccessKey() <<
                  std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}
```

- Para obter detalhes da API, consulte CreateAccessKey em Referência AWS SDK para C++ da API.

`CreateAccountAlias`

O código de exemplo a seguir mostra como usar `CreateAccountAlias`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
                  std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateAccountAlias em Referência AWS SDK para C++ da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                      const Aws::String &rsrcArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
                  std::endl;
    }
```

```
    return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
                 << "  \"Version\": \"2012-10-17\","
                 << "  \"Statement\": ["
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": \"logs:CreateLogGroup\","
                 << "        \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    },"
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": ["
                 << "            \"dynamodb:DeleteItem\","
                 << "            \"dynamodb:GetItem\","
                 << "            \"dynamodb:PutItem\","
                 << "            \"dynamodb:Scan\","
                 << "            \"dynamodb:UpdateItem\""
                 << "        ],"
                 << "        \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    }"
                 << "   ]"
                 << "}";

    return stringStream.str();
}
```

- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK para C++ da API.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::createIamRole(
        const Aws::String &roleName,
        const Aws::String &policy,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateRole em Referência AWS SDK para C++ da API.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateUserRequest create_request;
    create_request.SetUserName(userName);

    auto create_outcome = iam.CreateUser(create_request);
    if (!create_outcome.IsSuccess()) {
        std::cerr << "Error creating IAM user " << userName << ":" <<
                  create_outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created IAM user " << userName << std::endl;
    }

    return create_outcome.IsSuccess();
```

- Para obter detalhes da API, consulte CreateUser em Referência AWS SDK para C++ da API.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
```

```
                                    const Aws::String &accessKeyID,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
                  << userName << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                  << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteAccessKey em Referência AWS SDK para C++ da API.

`DeleteAccountAlias`

O código de exemplo a seguir mostra como usar `DeleteAccountAlias`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
```

```
                                        const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteAccountAlias em Referência AWS SDK para C++ da API.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration &clientConfig)
 {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
```

```
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeletePolicy em Referência AWS SDK para C++ da API.

`DeleteServerCertificate`

O código de exemplo a seguir mostra como usar `DeleteServerCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
```

```
            std::cerr << "Error deleting server certificate " << certificateName <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " << certificateName
                  << std::endl;
    }

    return result;
}
```

- Para obter detalhes da API, consulte DeleteServerCertificate em Referência AWS SDK para C ++ da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteUserRequest request;
    request.SetUserName(userName);
    auto outcome = iam.DeleteUser(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting IAM user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;;
```

```
    }
    else {
        std::cout << "Successfully deleted IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
```

- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para C++ da API.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
    detachRequest.SetRoleName(roleName);
    detachRequest.SetPolicyArn(policyArn);

    auto detachOutcome = iam.DetachRolePolicy(detachRequest);
    if (!detachOutcome.IsSuccess()) {
        std::cerr << "Failed to detach policy " << policyArn << " from role "
                  << roleName << ": " << detachOutcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully detached policy " << policyArn << " from role "
                  << roleName << std::endl;
    }

    return detachOutcome.IsSuccess();
```

- Para obter detalhes da API, consulte DetachRolePolicy em Referência AWS SDK para C++ da API.

GetAccessKeyLastUsed

O código de exemplo a seguir mostra como usar `GetAccessKeyLastUsed`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome = iam.GetAccessKeyLastUsed(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
                  secretKeyID << ":" << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
                outcome.GetResult()
                        .GetAccessKeyLastUsed()
                        .GetLastUsedDate()
                        .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
                  lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte GetAccessKeyLastUsed em Referência AWS SDK para C++ da API.

GetPolicy

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                            const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
                  "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
                  policy.GetArn() << std::endl << "Description: " <<
                  policy.GetDescription() << std::endl << "CreateDate: " <<

 policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte GetPolicy em Referência AWS SDK para C++ da API.

GetServerCertificate

O código de exemplo a seguir mostra como usar `GetServerCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<
```

```
 certificate.GetServerCertificateMetadata().GetServerCertificateName()
                  << std::endl << "Body: " << certificate.GetCertificateBody() <<
                  std::endl << "Chain: " << certificate.GetCertificateChain() <<
                  std::endl;
    }

    return result;
}
```

- Para obter detalhes da API, consulte GetServerCertificate em Referência AWS SDK para C++ da API.

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                 const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                      << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
```

```
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                      std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                    Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                            key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                      std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                      statusString << std::setw(20) <<
                      key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
 std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListAccessKeys em Referência AWS SDK para C++ da API.

`ListAccountAliases`

O código de exemplo a seguir mostra como usar `ListAccountAliases`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
                break;
            }
            std::cout << std::left << std::setw(32) << "Alias" << std::endl;
            header = true;
        }

        for (const auto &alias: aliases) {
            std::cout << std::left << std::setw(32) << alias << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
```

```
            done = true;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListAccountAliases em Referência AWS SDK para C++ da API.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration &clientConfig)
 {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
```

```
                        std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                        std::setw(64) << "Description" << std::setw(12) <<
                        "CreateDate" << std::endl;
            header = true;
        }

        const auto &policies = outcome.GetResult().GetPolicies();
        for (const auto &policy: policies) {
            std::cout << std::left << std::setw(55) <<
                        policy.GetPolicyName() << std::setw(30) <<
                        policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
                        std::setw(64) << policy.GetDescription() << std::setw(12) <<
                        policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
                        std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para C++ da API.

`ListServerCertificates`

O código de exemplo a seguir mostra como usar `ListServerCertificates`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::listServerCertificates(
        const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(14) << "UploadDate" << std::setw(14) <<
                      "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
                outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                      certificate.GetServerCertificateName() << std::setw(30) <<
                      certificate.GetServerCertificateId() << std::setw(80) <<
                      certificate.GetArn() << std::setw(14) <<
                      certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str())
 <<
                      std::setw(14) <<
                      certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str())
 <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
```

```
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListServerCertificates em Referência AWS SDK para C++ da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
```

```
            std::cout << std::left << std::setw(32) << "Name" <<
                      std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &users = outcome.GetResult().GetUsers();
        for (const auto &user: users) {
            std::cout << std::left << std::setw(32) << user.GetUserName() <<
                      std::setw(30) << user.GetUserId() << std::setw(64) <<
                      user.GetArn() << std::setw(20) <<
                      user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
                      << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para C++ da API.

`PutRolePolicy`

O código de exemplo a seguir mostra como usar `PutRolePolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::putRolePolicy(
```

```
        const Aws::String &roleName,
        const Aws::String &policyName,
        const Aws::String &policyDocument,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
 iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte PutRolePolicy em Referência AWS SDK para C++ da API.

`UpdateAccessKey`

O código de exemplo a seguir mostra como usar `UpdateAccessKey`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
```

```
                                    const Aws::String &accessKeyID,
                                    Aws::IAM::Model::StatusType status,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateAccessKey em Referência AWS SDK para C++ da API.

`UpdateServerCertificate`

O código de exemplo a seguir mostra como usar `UpdateServerCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String &currentCertificateName,
```

```
                                                const Aws::String &newCertificateName,
                                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName << ":"
 <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                      << "' not found." << std::endl;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte UpdateServerCertificate em Referência AWS SDK para C++ da API.

`UpdateUser`

O código de exemplo a seguir mostra como usar `UpdateUser`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                  " successfully updated with new user name " << newUserName <<
                  std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateUser em Referência AWS SDK para C++ da API.

AWS IoT exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with AWS IoT.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

SDK para C++

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
```

```
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_iot.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */
```

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
 iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
 listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                          << listThingsOutcome.GetError().GetMessage() << std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }
```

```
    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK para C++ .

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie uma AWS IoT coisa.
- Gerar um certificado de dispositivo.
- Atualize AWS IoT qualquer coisa com atributos.
- Exibir um endpoint exclusivo.
- Liste seus AWS IoT certificados.
- Atualize uma AWS IoT sombra.
- Gravar informações do estado.
- Cria uma regra.
- Listar suas regras.
- Pesquisar coisas usando o nome da coisa.
- Exclua qualquer AWS IoT coisa.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie qualquer AWS IoT coisa.

```
    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }
```

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

Gerar e anexar um certificado de dispositivo.

```
    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing? (y/
n) ")) {
        Aws::String outputFolder;
        if (askYesNoQuestion(
                "Would you like to save the certificate and keys to file? (y/n) "))
 {
            outputFolder = std::filesystem::current_path();
            outputFolder += "/device_keys_and_certificates";

            std::filesystem::create_directories(outputFolder);

            std::cout << "The certificate and keys will be saved to the folder: "
                      << outputFolder << std::endl;
        }

        if (!createKeysAndCertificate(outputFolder, certificateARN, certificateID,
                                      clientConfiguration)) {
            std::cerr << "Exiting because createKeysAndCertificate failed."
                      << std::endl;
            cleanup(thingName, "", "", "", "", false, clientConfiguration);
            return false;
        }

        std::cout << "\nNext, the certificate will be attached to the thing.\n"
                  << std::endl;
        if (!attachThingPrincipal(certificateARN, thingName, clientConfiguration)) {
            std::cerr << "Exiting because attachThingPrincipal failed." <<
std::endl;
            cleanup(thingName, certificateARN, certificateID, "", "",
                    false,
                    clientConfiguration);
            return false;
        }
    }
```

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string is
 empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ", certificate
 ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
```

```
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" << publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Execute várias operações na AWS IoT coisa.

```
    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
 "v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now an endpoint will be retrieved for your account.\n" <<
 std::endl;
    std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
 Locator that serves as the entry point\n"
    << "for communication between IoT devices and the AWS IoT service." <<
 std::endl;
```

```
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String endpoint;
    if (!describeEndpoint(endpoint, clientConfiguration)) {
        std::cerr << "Exiting because getEndpoint failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
    printAsterisksLine();

    std::cout << "Now the certificates in your account will be listed." <<
 std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    if (!listCertificates(clientConfiguration)) {
        std::cerr << "Exiting because listCertificates failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
    std::cout << "A thing shadow refers to a feature that enables you to create a
 virtual representation, or \"shadow,\"\n"
    << "of a physical device or thing. The thing shadow allows you to synchronize
 and control the state of a device between\n"
    << "the cloud and the device itself. and the AWS IoT service. For example, you
 can write and retrieve JSON data from a thing shadow." << std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
        std::cerr << "Exiting because updateThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
```

```
   std::cout << "Now, the state information for the shadow will be retrieved.\n" <<
std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);

   Aws::String shadowState;
   if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
       std::cerr << "Exiting because getThingShadow failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }
   std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

   printAsterisksLine();

   std::cout << "A rule with now be added to to the thing.\n" << std::endl;
   std::cout << "Any user who has permission to create rules will be able to access
data processed by the rule." << std::endl;
   std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
   std::cout << "These resources will be created using a CloudFormation template."
<< std::endl;
   std::cout << "Stack creation may take a few minutes." << std::endl;

   askQuestion("Press Enter to continue: ", alwaysTrueTest);
   Aws::Map<Aws::String, Aws::String> outputs
=createCloudFormationStack(STACK_NAME,clientConfiguration);
   if (outputs.empty()) {
       std::cerr << "Exiting because createCloudFormationStack failed." <<
std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }

   // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
   auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
   auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
   if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
       std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
       "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."  <<
std::endl;
       cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
```

```
                false,
                clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM '";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += "'";

    printAsterisksLine();

    std::cout << "Now a rule will be created.\n" << std::endl;
    std::cout << "Rules are an administrator-level action. Any user who has
 permission\n"
                << "to create rules will be able to access data processed by the
 rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
 std::endl;
    std::cout << "For more information on IoT SQL, see https://docs.aws.amazon.com/
iot/latest/developerguide/iot-sql-reference.html" << std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
 clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }
```

```
    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
 account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
       << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/latest/
developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account? (y/
n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

 thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGISTR

 thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnectiv
        // The ThingGroupIndexingConfiguration object is ignored if not set.
        Aws::IoT::Model::ThingGroupIndexingConfiguration
 thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
 thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                    ruleName, false,
                    clientConfiguration);
            return false;
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
```

```
    }
```

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
```

```
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome = iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);
```

```
        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
```

```
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
```

```
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
```

```
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome = iotClient.ListTopicRules(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
 outcome.GetResult();
            allRules.insert(allRules.end(),
                            result.GetRules().cbegin(),
                            result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "ListTopicRules error: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
              << std::endl;
    for (auto &rule: allRules) {
        std::cout << "  Rule name: " << rule.GetRuleName() << ", rule ARN: "
                  << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
```

```
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result = outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());
```

```
    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

Limpe recursos.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
 &certificateARN,
                     const Aws::String &certificateID, const Aws::String &stackName,
                     const Aws::String &ruleName, bool askForConfirmation,
                     const Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
                              askYesNoQuestion("Delete the rule '" + ruleName +
                                               "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
    }

    Aws::CloudFormation::CloudFormationClient
 cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
                              askYesNoQuestion(
                                      "Delete the CloudFormation stack '" +
 stackName +
                                      "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
                                    askYesNoQuestion("Delete the certificate '" +
                                                     certificateARN + "'? (y/n) ")))
 {
        result &= detachThingPrincipal(certificateARN, thingName,
 clientConfiguration);
```

```
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                               askYesNoQuestion("Delete the thing '" + thingName +
                                                "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from thing
 "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome = iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
```

```
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .

- AttachThingPrincipal
- CreateKeysAndCertificate
- CreateThing
- CreateTopicRule
- DeleteCertificate
- DeleteThing
- DeleteTopicRule
- DescribeEndpoint
- DescribeThing
- DetachThingPrincipal
- ListCertificates
- ListThings
- SearchIndex
- UpdateIndexingConfiguration
- UpdateThing

Ações

`AttachThingPrincipal`

O código de exemplo a seguir mostra como usar `AttachThingPrincipal`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AttachThingPrincipal em Referência AWS SDK para C++ da API.

`CreateKeysAndCertificate`

O código de exemplo a seguir mostra como usar `CreateKeysAndCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create keys and certificate for an Aws IoT device.
```

```
//! This routine will save certificates and keys to an output folder, if provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string is
 empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ", certificate
 ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
```

```
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" << publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateKeysAndCertificate em Referência AWS SDK para C++ da API.

`CreateThing`

O código de exemplo a seguir mostra como usar `CreateThing`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateThing em Referência AWS SDK para C++ da API.

`CreateTopicRule`

O código de exemplo a seguir mostra como usar `CreateTopicRule`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});
```

```
    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTopicRule em Referência AWS SDK para C++ da API.

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
```

```
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome = iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteCertificate em Referência AWS SDK para C++ da API.

`DeleteThing`

O código de exemplo a seguir mostra como usar `DeleteThing`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteThing em Referência AWS SDK para C++ da API.

`DeleteTopicRule`

O código de exemplo a seguir mostra como usar `DeleteTopicRule`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
```

```
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTopicRule em Referência AWS SDK para C++ da API.

`DescribeEndpoint`

O código de exemplo a seguir mostra como usar `DescribeEndpoint`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome = iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeEndpoint em Referência AWS SDK para C++ da API.

`DescribeThing`

O código de exemplo a seguir mostra como usar `DescribeThing`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
 iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "'" <<
 std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
                  << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
 attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeThing em Referência AWS SDK para C++ da API.

`DetachThingPrincipal`

O código de exemplo a seguir mostra como usar `DetachThingPrincipal`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from thing
 "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DetachThingPrincipal em Referência AWS SDK para C++ da API.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
```

```
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListCertificates em Referência AWS SDK para C++ da API.

`SearchIndex`

O código de exemplo a seguir mostra como usar `SearchIndex`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Query the AWS IoT fleet index.
```

```
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result = outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
```

```
                    << std::endl;
    }
    return true;
}
```

- Para obter detalhes da API, consulte SearchIndex em Referência AWS SDK para C++ da API.

`UpdateIndexingConfiguration`

O código de exemplo a seguir mostra como usar `UpdateIndexingConfiguration`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update the indexing configuration.
/*!
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration object
 which is ignored if not set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateIndexingConfiguration(
        const Aws::IoT::Model::ThingIndexingConfiguration
 &thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
 &thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }
```

```
    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {
        request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
 iotClient.UpdateIndexingConfiguration(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateIndexingConfiguration em Referência AWS SDK para C++ da API.

`UpdateThing`

O código de exemplo a seguir mostra como usar `UpdateThing`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateThing em Referência AWS SDK para C++ da API.

AWS IoT data exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with AWS IoT data.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetThingShadow`

O código de exemplo a seguir mostra como usar `GetThingShadow`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetThingShadow em Referência AWS SDK para C++ da API.

`UpdateThingShadow`

O código de exemplo a seguir mostra como usar `UpdateThingShadow`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateThingShadow em Referência AWS SDK para C++ da API.

Exemplos do Lambda usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Lambda

O exemplo de código a seguir mostra como começar a usar o Lambda.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS lambda)

# Set this project's name.
project("hello_lambda")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_lambda.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_lambda.cpp.

```
#include <aws/core/Aws.h>
#include <aws/lambda/LambdaClient.h>
#include <aws/lambda/model/ListFunctionsRequest.h>
#include <iostream>

/*
 *  A "Hello Lambda" starter application which initializes an AWS Lambda (Lambda)
 client and lists the Lambda functions.
 *
 *  main function
 *
 *  Usage: 'hello_lambda'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
```

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Lambda::LambdaClient lambdaClient(clientConfig);
        std::vector<Aws::String> functions;
        Aws::String marker; // Used for pagination.

        do {
            Aws::Lambda::Model::ListFunctionsRequest request;
            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::Lambda::Model::ListFunctionsOutcome outcome =
lambdaClient.ListFunctions(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Lambda::Model::ListFunctionsResult &listFunctionsResult =
outcome.GetResult();
                std::cout << listFunctionsResult.GetFunctions().size()
                          << " lambda functions were retrieved." << std::endl;

                for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: listFunctionsResult.GetFunctions()) {
                    functions.push_back(functionConfiguration.GetFunctionName());
                    std::cout << functions.size() << "  "
                              << functionConfiguration.GetDescription() <<
std::endl;
                    std::cout << "   "
                              <<
Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                      functionConfiguration.GetRuntime()) << ": "
                              << functionConfiguration.GetHandler()
                              << std::endl;
                }
                marker = listFunctionsResult.GetNextMarker();
            } else {
                std::cerr << "Error with Lambda::ListFunctions. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
```

```
            }
        } while (!marker.empty());
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para C++ da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get started with functions scenario.
/*!
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
```

```
 */
bool AwsDoc::Lambda::getStartedWithFunctionsScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::Lambda::LambdaClient client(clientConfig);

    // 1. Create an AWS Identity and Access Management (IAM) role for Lambda
 function.
    Aws::String roleArn;
    if (!getIamRoleArn(roleArn, clientConfig)) {
        return false;
    }

    // 2. Create a Lambda function.
    int seconds = 0;
    do {
        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#if USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);

        // Assume the AWS Lambda function was built in Docker with same architecture
        // as this code.
#if  defined(__x86_64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif
        request.SetRole(roleArn);
        request.SetHandler(LAMBDA_HANDLER_NAME);
        request.SetPublish(true);
        Aws::Lambda::Model::FunctionCode code;
        std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
```

```
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();

        code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
 buffer.str().c_str(),
                                                buffer.str().length()));
        request.SetCode(code);

        Aws::Lambda::Model::CreateFunctionOutcome outcome = client.CreateFunction(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda function was successfully created. " << seconds
                      << " seconds elapsed." << std::endl;
            break;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::Lambda::LambdaErrors::INVALID_PARAMETER_VALUE &&
                 outcome.GetError().GetMessage().find("role") >= 0) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout
                        << "Waiting for the IAM role to become available as a
 CreateFunction parameter. "
                        << seconds
                        << " seconds elapsed." << std::endl;

                std::cout << outcome.GetError().GetMessage() << std::endl;
            }
        }
        else {
            std::cerr << "Error with CreateFunction. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (60 > seconds);

    std::cout << "The current Lambda function increments 1 by an input." <<
std::endl;

    // 3.  Invoke the Lambda function.
    {
        int increment = askQuestionForInt("Enter an increment integer: ");

        Aws::Lambda::Model::InvokeResult invokeResult;
        Aws::Utils::Json::JsonValue jsonPayload;
        jsonPayload.WithString("action", "increment");
        jsonPayload.WithInteger("number", increment);
        if (invokeLambdaFunction(jsonPayload, Aws::Lambda::Model::LogType::Tail,
                                 invokeResult, client)) {
            Aws::Utils::Json::JsonValue jsonValue(invokeResult.GetPayload());
            Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
                    jsonValue.View().GetAllObjects();
            auto iter = values.find("result");
            if (iter != values.end() && iter->second.IsIntegerType()) {
                {
                    std::cout << INCREMENT_RESUlT_PREFIX
                              << iter->second.AsInteger() << std::endl;
                }
            }
            else {
                std::cout << "There was an error in execution. Here is the log."
                          << std::endl;
                Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
                        invokeResult.GetLogResult());
                std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
            }
        }
    }

    std::cout
```

```
            << "The Lambda function will now be updated with new code. Press return
 to continue, ";
    Aws::String answer;
    std::getline(std::cin, answer);

    // 4.  Update the Lambda function code.
    {
        Aws::Lambda::Model::UpdateFunctionCodeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
                Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                       buffer.str().length()));
        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
 client.UpdateFunctionCode(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
        }
    }

    std::cout
            << "This function uses an environment variable to control the logging
level."
            << std::endl;
    std::cout
            << "UpdateFunctionConfiguration will be used to set the LOG_LEVEL to
DEBUG."
            << std::endl;
    seconds = 0;

    // 5.  Update the Lambda function configuration.
    do {
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        Aws::Lambda::Model::Environment environment;
        environment.AddVariables("LOG_LEVEL", "DEBUG");
        request.SetEnvironment(environment);

        Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda configuration was successfully updated."
                      << std::endl;
            break;
        }

            // RESOURCE_IN_USE: function code update not completed.
        else if (outcome.GetError().GetErrorType() !=
                 Aws::Lambda::LambdaErrors::RESOURCE_IN_USE) {
            if ((seconds % 10) == 0) { // Log status every 10 seconds.
                std::cout << "Lambda function update in progress . After " <<
seconds
                          << " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
```

```
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (0 < seconds);

    if (0 > seconds) {
        std::cerr << "Function failed to become active." << std::endl;
    }
    else {
        std::cout << "Updated function active after " << seconds << " seconds."
                  << std::endl;
    }

    std::cout
            << "\nThe new code applies an arithmetic operator to two variables, x an
y."
            << std::endl;
    std::vector<Aws::String> operators = {"plus", "minus", "times", "divided-by"};
    for (size_t i = 0; i < operators.size(); ++i) {
        std::cout << "   " << i + 1 << " " << operators[i] << std::endl;
    }

    // 6.  Invoke the updated Lambda function.
    do {
        int operatorIndex = askQuestionForIntRange("Select an operator index 1 - 4
", 1,
                                                   4);
        int x = askQuestionForInt("Enter an integer for the x value ");
        int y = askQuestionForInt("Enter an integer for the y value ");

        Aws::Utils::Json::JsonValue calculateJsonPayload;
        calculateJsonPayload.WithString("action", operators[operatorIndex - 1]);
        calculateJsonPayload.WithInteger("x", x);
        calculateJsonPayload.WithInteger("y", y);
        Aws::Lambda::Model::InvokeResult calculatedResult;
        if (invokeLambdaFunction(calculateJsonPayload,
                                 Aws::Lambda::Model::LogType::Tail,
                                 calculatedResult, client)) {
            Aws::Utils::Json::JsonValue jsonValue(calculatedResult.GetPayload());
            Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
                    jsonValue.View().GetAllObjects();
            auto iter = values.find("result");
            if (iter != values.end() && iter->second.IsIntegerType()) {
```

```
                std::cout << ARITHMETIC_RESUlT_PREFIX << x << " "
                          << operators[operatorIndex - 1] << " "
                          << y << " is " << iter->second.AsInteger() << std::endl;
            }
            else if (iter != values.end() && iter->second.IsFloatingPointType()) {
                std::cout << ARITHMETIC_RESUlT_PREFIX << x << " "
                          << operators[operatorIndex - 1] << " "
                          << y << " is " << iter->second.AsDouble() << std::endl;
            }
            else {
                std::cout << "There was an error in execution. Here is the log."
                          << std::endl;
                Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
                        calculatedResult.GetLogResult());
                std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
            }
        }

        answer = askQuestion("Would you like to try another operation? (y/n) ");
    } while (answer == "y");

    std::cout
            << "A list of the lambda functions will be retrieved. Press return to
continue, ";
    std::getline(std::cin, answer);

    // 7.  List the Lambda functions.

    std::vector<Aws::String> functions;
    Aws::String marker;

    do {
        Aws::Lambda::Model::ListFunctionsRequest request;
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
```

```
            std::cout << result.GetFunctions().size()
                      << " lambda functions were retrieved." << std::endl;

            for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
                functions.push_back(functionConfiguration.GetFunctionName());
                std::cout << functions.size() << "  "
                          << functionConfiguration.GetDescription() << std::endl;
                std::cout << "   "
                          << Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                  functionConfiguration.GetRuntime()) << ": "
                          << functionConfiguration.GetHandler()
                          << std::endl;
            }
            marker = result.GetNextMarker();
        }
        else {
            std::cerr << "Error with Lambda::ListFunctions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    // 8.  Get a Lambda function.
    if (!functions.empty()) {
        std::stringstream question;
        question << "Choose a function to retrieve between 1 and " <<
functions.size()
                 << " ";
        int functionIndex = askQuestionForIntRange(question.str(), 1,

static_cast<int>(functions.size()));

        Aws::String functionName = functions[functionIndex - 1];

        Aws::Lambda::Model::GetFunctionRequest request;
        request.SetFunctionName(functionName);

        Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

        if (outcome.IsSuccess()) {
            std::cout << "Function retrieve.\n" <<
```

```
 outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
                      << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::GetFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout << "The resources will be deleted. Press return to continue, ";
    std::getline(std::cin, answer);

    // 9.  Delete the Lambda function.
    bool result = deleteLambdaFunction(client);

    // 10. Delete the IAM role.
    return result && deleteIamRole(clientConfig);
}

//! Routine which invokes a Lambda function and returns the result.
/*!
 \param jsonPayload: Payload for invoke function.
 \param logType: Log type setting for invoke function.
 \param invokeResult: InvokeResult object to receive the result.
 \param client: Lambda client.
 \return bool: Successful completion.
 */
bool
AwsDoc::Lambda::invokeLambdaFunction(const Aws::Utils::Json::JsonValue &jsonPayload,
                                     Aws::Lambda::Model::LogType logType,
                                     Aws::Lambda::Model::InvokeResult &invokeResult,
                                     const Aws::Lambda::LambdaClient &client) {
    int seconds = 0;
    bool result = false;
    /*
     * In this example, the Invoke function can be called before recently created
 resources are
     * available.  The Invoke function is called repeatedly until the resources are
     * available.
     */
    do {
        Aws::Lambda::Model::InvokeRequest request;
```

```
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload = Aws::MakeShared<Aws::StringStream>(
                "FunctionTest");
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
            invokeResult = std::move(outcome.GetResult());
            result = true;
            break;
        }

            // ACCESS_DENIED: because the role is not available yet.
            // RESOURCE_CONFLICT: because the Lambda function is being created or
 updated.
        else if ((outcome.GetError().GetErrorType() ==
                  Aws::Lambda::LambdaErrors::ACCESS_DENIED) ||
                 (outcome.GetError().GetErrorType() ==
                  Aws::Lambda::LambdaErrors::RESOURCE_CONFLICT)) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout << "Waiting for the invoke api to be available, status "
 <<
                          ((outcome.GetError().GetErrorType() ==
                            Aws::Lambda::LambdaErrors::ACCESS_DENIED ?
                            "ACCESS_DENIED" : "RESOURCE_CONFLICT")) << ". " <<
 seconds
                          << " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error with Lambda::InvokeRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            break;
        }
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (seconds < 60);

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
 (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#if USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
```

```
        request.SetTimeout(15);
        request.SetMemorySize(128);

        // Assume the AWS Lambda function was built in Docker with same architecture
        // as this code.
#if  defined(__x86_64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif
        request.SetRole(roleArn);
        request.SetHandler(LAMBDA_HANDLER_NAME);
        request.SetPublish(true);
        Aws::Lambda::Model::FunctionCode code;
        std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();

        code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
 buffer.str().c_str(),
                                               buffer.str().length()));
        request.SetCode(code);

        Aws::Lambda::Model::CreateFunctionOutcome outcome = client.CreateFunction(
```

```
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda function was successfully created. " << seconds
                      << " seconds elapsed." << std::endl;
            break;
        }

        else {
            std::cerr << "Error with CreateFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
```

- Para obter detalhes da API, consulte CreateFunction em Referência AWS SDK para C++ da API.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::DeleteFunctionRequest request;
    request.SetFunctionName(LAMBDA_NAME);
```

```
    Aws::Lambda::Model::DeleteFunctionOutcome outcome = client.DeleteFunction(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda function was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::DeleteFunction. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Para obter detalhes da API, consulte DeleteFunction em Referência AWS SDK para C++ da API.

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::GetFunctionRequest request;
        request.SetFunctionName(functionName);

        Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "Function retrieve.\n" <<

outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
                      << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::GetFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte GetFunction em Referência AWS SDK para C++ da API.

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload = Aws::MakeShared<Aws::StringStream>(
                "FunctionTest");
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
```

```
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
            invokeResult = std::move(outcome.GetResult());
            result = true;
            break;
        }

        else {
            std::cerr << "Error with Lambda::InvokeRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            break;
        }
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para C++ .

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    std::vector<Aws::String> functions;
    Aws::String marker;
```

```
    do {
        Aws::Lambda::Model::ListFunctionsRequest request;
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
            std::cout << result.GetFunctions().size()
                      << " lambda functions were retrieved." << std::endl;

            for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
                functions.push_back(functionConfiguration.GetFunctionName());
                std::cout << functions.size() << "  "
                          << functionConfiguration.GetDescription() << std::endl;
                std::cout << "   "
                          << Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                  functionConfiguration.GetRuntime()) << ": "
                          << functionConfiguration.GetHandler()
                          << std::endl;
            }
            marker = result.GetNextMarker();
        }
        else {
            std::cerr << "Error with Lambda::ListFunctions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para C++ da API.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
 (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::UpdateFunctionCodeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
                Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                       buffer.str().length()));
        request.SetPublish(true);
```

```
        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
client.UpdateFunctionCode(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte UpdateFunctionCode em Referência AWS SDK para C++ da API.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        Aws::Lambda::Model::Environment environment;
        environment.AddVariables("LOG_LEVEL", "DEBUG");
        request.SetEnvironment(environment);
```

```
        Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda configuration was successfully updated."
                      << std::endl;
            break;
        }

        else {
            std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte UpdateFunctionConfiguration em Referência AWS SDK para C++ da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

MediaConvert exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with MediaConvert.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an AWS Elemental MediaConvert job.
/*!
  \param mediaConvertRole: An Amazon Resource Name (ARN) for the AWS Identity and
                           Access Management (IAM) role for the job.
```

```
 \param fileInput: A URI to an input file that is stored in Amazon Simple Storage
Service
                   (Amazon S3) or on an HTTP(S) server.
 \param fileOutput: A URI for an Amazon S3 output location and the output file name
base.
 \param jobSettingsFile: An optional JSON settings file.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/

bool AwsDoc::MediaConvert::createJob(const Aws::String &mediaConvertRole,
                                     const Aws::String &fileInput,
                                     const Aws::String &fileOutput,
                                     const Aws::String &jobSettingsFile,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::MediaConvert::Model::CreateJobRequest createJobRequest;

    createJobRequest.SetRole(mediaConvertRole);
    Aws::Http::HeaderValueCollection hvc;
    hvc.emplace("Customer", "Amazon");
    createJobRequest.SetUserMetadata(hvc);

    if (!jobSettingsFile.empty()) // Use a JSON file for the job settings.
    {
        std::ifstream jobSettingsStream(jobSettingsFile, std::ios::ate);
        if (!jobSettingsStream) {
            std::cerr << "Unable to open the job template file." << std::endl;
            return false;
        }
        std::vector<char> buffer(jobSettingsStream.tellg());
        jobSettingsStream.seekg(0);
        jobSettingsStream.read(buffer.data(), buffer.size());
        std::string jobSettingsJSON(buffer.data(), buffer.size());
        size_t pos = jobSettingsJSON.find(INPUT_FILE_PLACEHOLDER);
        if (pos != std::string::npos) {
            jobSettingsJSON.replace(pos, strlen(INPUT_FILE_PLACEHOLDER), fileInput);
        }

        pos = jobSettingsJSON.find(OUTPUT_FILE_PLACEHOLDER);
        if (pos != std::string::npos) {
            jobSettingsJSON.replace(pos, strlen(OUTPUT_FILE_PLACEHOLDER),
fileOutput);
        }
```

```
        Aws::Utils::Json::JsonValue jsonValue(jobSettingsJSON);
        Aws::MediaConvert::Model::JobSettings jobSettings(jsonValue);

        createJobRequest.SetSettings(jobSettings);
    }
    else { // Configure the job settings programmatically.
        Aws::MediaConvert::Model::JobSettings jobSettings;
        jobSettings.SetAdAvailOffset(0);
        Aws::MediaConvert::Model::TimecodeConfig timecodeConfig;

timecodeConfig.SetSource(Aws::MediaConvert::Model::TimecodeSource::EMBEDDED);
        jobSettings.SetTimecodeConfig(timecodeConfig);

        // Configure the output group.
        Aws::MediaConvert::Model::OutputGroup outputGroup;
        outputGroup.SetName("File Group");
        Aws::MediaConvert::Model::OutputGroupSettings outputGroupSettings;
        outputGroupSettings.SetType(
                Aws::MediaConvert::Model::OutputGroupType::FILE_GROUP_SETTINGS);
        Aws::MediaConvert::Model::FileGroupSettings fileGroupSettings;
        fileGroupSettings.SetDestination(fileOutput);
        outputGroupSettings.SetFileGroupSettings(fileGroupSettings);
        outputGroup.SetOutputGroupSettings(outputGroupSettings);

        Aws::MediaConvert::Model::Output output;
        output.SetNameModifier("_1");

        Aws::MediaConvert::Model::VideoDescription videoDescription;
        videoDescription.SetScalingBehavior(
                Aws::MediaConvert::Model::ScalingBehavior::DEFAULT);
        videoDescription.SetTimecodeInsertion(
                Aws::MediaConvert::Model::VideoTimecodeInsertion::DISABLED);
        videoDescription.SetAntiAlias(Aws::MediaConvert::Model::AntiAlias::ENABLED);
        videoDescription.SetSharpness(50);

videoDescription.SetAfdSignaling(Aws::MediaConvert::Model::AfdSignaling::NONE);
        videoDescription.SetDropFrameTimecode(
                Aws::MediaConvert::Model::DropFrameTimecode::ENABLED);

videoDescription.SetRespondToAfd(Aws::MediaConvert::Model::RespondToAfd::NONE);
        videoDescription.SetColorMetadata(
                Aws::MediaConvert::Model::ColorMetadata::INSERT);

        Aws::MediaConvert::Model::VideoCodecSettings videoCodecSettings;
```

```
        videoCodecSettings.SetCodec(Aws::MediaConvert::Model::VideoCodec::H_264);
        Aws::MediaConvert::Model::H264Settings h264Settings;
        h264Settings.SetNumberReferenceFrames(3);
        h264Settings.SetSyntax(Aws::MediaConvert::Model::H264Syntax::DEFAULT);
        h264Settings.SetSoftness(0);
        h264Settings.SetGopClosedCadence(1);
        h264Settings.SetGopSize(90);
        h264Settings.SetSlices(1);
        h264Settings.SetGopBReference(
                Aws::MediaConvert::Model::H264GopBReference::DISABLED);
        h264Settings.SetSlowPal(Aws::MediaConvert::Model::H264SlowPal::DISABLED);
        h264Settings.SetSpatialAdaptiveQuantization(
                Aws::MediaConvert::Model::H264SpatialAdaptiveQuantization::ENABLED);
        h264Settings.SetTemporalAdaptiveQuantization(

Aws::MediaConvert::Model::H264TemporalAdaptiveQuantization::ENABLED);
        h264Settings.SetFlickerAdaptiveQuantization(

Aws::MediaConvert::Model::H264FlickerAdaptiveQuantization::DISABLED);
        h264Settings.SetEntropyEncoding(
                Aws::MediaConvert::Model::H264EntropyEncoding::CABAC);
        h264Settings.SetBitrate(5000000);
        h264Settings.SetFramerateControl(
                Aws::MediaConvert::Model::H264FramerateControl::SPECIFIED);
        h264Settings.SetRateControlMode(
                Aws::MediaConvert::Model::H264RateControlMode::CBR);

h264Settings.SetCodecProfile(Aws::MediaConvert::Model::H264CodecProfile::MAIN);
        h264Settings.SetTelecine(Aws::MediaConvert::Model::H264Telecine::NONE);
        h264Settings.SetMinIInterval(0);
        h264Settings.SetAdaptiveQuantization(
                Aws::MediaConvert::Model::H264AdaptiveQuantization::HIGH);
        h264Settings.SetCodecLevel(Aws::MediaConvert::Model::H264CodecLevel::AUTO);
        h264Settings.SetFieldEncoding(
                Aws::MediaConvert::Model::H264FieldEncoding::PAFF);
        h264Settings.SetSceneChangeDetect(
                Aws::MediaConvert::Model::H264SceneChangeDetect::ENABLED);
        h264Settings.SetQualityTuningLevel(
                Aws::MediaConvert::Model::H264QualityTuningLevel::SINGLE_PASS);
        h264Settings.SetFramerateConversionAlgorithm(

Aws::MediaConvert::Model::H264FramerateConversionAlgorithm::DUPLICATE_DROP);
        h264Settings.SetUnregisteredSeiTimecode(
                Aws::MediaConvert::Model::H264UnregisteredSeiTimecode::DISABLED);
```

```
        h264Settings.SetGopSizeUnits(
                Aws::MediaConvert::Model::H264GopSizeUnits::FRAMES);

h264Settings.SetParControl(Aws::MediaConvert::Model::H264ParControl::SPECIFIED);
        h264Settings.SetNumberBFramesBetweenReferenceFrames(2);

h264Settings.SetRepeatPps(Aws::MediaConvert::Model::H264RepeatPps::DISABLED);
        h264Settings.SetFramerateNumerator(30);
        h264Settings.SetFramerateDenominator(1);
        h264Settings.SetParNumerator(1);
        h264Settings.SetParDenominator(1);
        videoCodecSettings.SetH264Settings(h264Settings);
        videoDescription.SetCodecSettings(videoCodecSettings);
        output.SetVideoDescription(videoDescription);

        Aws::MediaConvert::Model::AudioDescription audioDescription;
        audioDescription.SetLanguageCodeControl(
                Aws::MediaConvert::Model::AudioLanguageCodeControl::FOLLOW_INPUT);
        audioDescription.SetAudioSourceName(AUDIO_SOURCE_NAME);
        Aws::MediaConvert::Model::AudioCodecSettings audioCodecSettings;
        audioCodecSettings.SetCodec(Aws::MediaConvert::Model::AudioCodec::AAC);
        Aws::MediaConvert::Model::AacSettings aacSettings;
        aacSettings.SetAudioDescriptionBroadcasterMix(

Aws::MediaConvert::Model::AacAudioDescriptionBroadcasterMix::NORMAL);
        aacSettings.SetRateControlMode(
                Aws::MediaConvert::Model::AacRateControlMode::CBR);
        aacSettings.SetCodecProfile(Aws::MediaConvert::Model::AacCodecProfile::LC);
        aacSettings.SetCodingMode(
                Aws::MediaConvert::Model::AacCodingMode::CODING_MODE_2_0);
        aacSettings.SetRawFormat(Aws::MediaConvert::Model::AacRawFormat::NONE);
        aacSettings.SetSampleRate(48000);

aacSettings.SetSpecification(Aws::MediaConvert::Model::AacSpecification::MPEG4);
        aacSettings.SetBitrate(64000);
        audioCodecSettings.SetAacSettings(aacSettings);
        audioDescription.SetCodecSettings(audioCodecSettings);
        Aws::Vector<Aws::MediaConvert::Model::AudioDescription> audioDescriptions;
        audioDescriptions.emplace_back(audioDescription);
        output.SetAudioDescriptions(audioDescriptions);

        Aws::MediaConvert::Model::ContainerSettings mp4container;
        mp4container.SetContainer(Aws::MediaConvert::Model::ContainerType::MP4);
        Aws::MediaConvert::Model::Mp4Settings mp4Settings;
```

```
        mp4Settings.SetCslgAtom(Aws::MediaConvert::Model::Mp4CslgAtom::INCLUDE);

mp4Settings.SetFreeSpaceBox(Aws::MediaConvert::Model::Mp4FreeSpaceBox::EXCLUDE);
        mp4Settings.SetMoovPlacement(
                Aws::MediaConvert::Model::Mp4MoovPlacement::PROGRESSIVE_DOWNLOAD);
        mp4container.SetMp4Settings(mp4Settings);
        output.SetContainerSettings(mp4container);

        outputGroup.AddOutputs(output);
        jobSettings.AddOutputGroups(outputGroup);

        // Configure inputs.
        Aws::MediaConvert::Model::Input input;
        input.SetFilterEnable(Aws::MediaConvert::Model::InputFilterEnable::AUTO);
        input.SetPsiControl(Aws::MediaConvert::Model::InputPsiControl::USE_PSI);
        input.SetFilterStrength(0);

input.SetDeblockFilter(Aws::MediaConvert::Model::InputDeblockFilter::DISABLED);

input.SetDenoiseFilter(Aws::MediaConvert::Model::InputDenoiseFilter::DISABLED);
        input.SetTimecodeSource(
                Aws::MediaConvert::Model::InputTimecodeSource::EMBEDDED);
        input.SetFileInput(fileInput);

        Aws::MediaConvert::Model::AudioSelector audioSelector;
        audioSelector.SetOffset(0);
        audioSelector.SetDefaultSelection(
                Aws::MediaConvert::Model::AudioDefaultSelection::NOT_DEFAULT);
        audioSelector.SetProgramSelection(1);
        audioSelector.SetSelectorType(
                Aws::MediaConvert::Model::AudioSelectorType::TRACK);
        audioSelector.AddTracks(1);
        input.AddAudioSelectors(AUDIO_SOURCE_NAME, audioSelector);

        Aws::MediaConvert::Model::VideoSelector videoSelector;
        videoSelector.SetColorSpace(Aws::MediaConvert::Model::ColorSpace::FOLLOW);
        input.SetVideoSelector(videoSelector);

        jobSettings.AddInputs(input);

        createJobRequest.SetSettings(jobSettings);
    }

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);
```

```
    Aws::MediaConvert::Model::CreateJobOutcome outcome = client.CreateJob(
            createJobRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Job successfully created with ID - "
                  << outcome.GetResult().GetJob().GetId() << std::endl;
    }
    else {
        std::cerr << "Error CreateJob - " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK para C++ da API.

GetJob

O código de exemplo a seguir mostra como usar `GetJob`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve the information for a specific completed transcoding job.
/*!
  \param jobID: A job ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::getJob(const Aws::String &jobID,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    Aws::MediaConvert::Model::GetJobRequest request;
    request.SetId(jobID);
```

```
    const Aws::MediaConvert::Model::GetJobOutcome outcome = client.GetJob(
            request);
    if (outcome.IsSuccess()) {
        std::cout << outcome.GetResult().GetJob().Jsonize().View().WriteReadable()
                  << std::endl;
    }
    else {
        std::cerr << "DescribeEndpoints error - " << outcome.GetError().GetMessage()
                  << std::endl;
    }


    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetJob em Referência AWS SDK para C++ da API.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve a list of created jobs.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::listJobs(
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    bool result = true;
    Aws::String nextToken; // Used to handle paginated results.
```

```
    do {
        Aws::MediaConvert::Model::ListJobsRequest request;
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        const Aws::MediaConvert::Model::ListJobsOutcome outcome = client.ListJobs(
                request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::MediaConvert::Model::Job> &jobs =
                    outcome.GetResult().GetJobs();
            std::cout << jobs.size() << " jobs retrieved." << std::endl;
            for (const Aws::MediaConvert::Model::Job &job: jobs) {
                std::cout << "  " << job.Jsonize().View().WriteReadable() <<
std::endl;
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "DescribeEndpoints error - " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
            break;

        }
    } while (!nextToken.empty());


    return result;
}
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para C++ da API.

Exemplos do Amazon RDS usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Amazon RDS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon RDS

O exemplo de código a seguir mostra como começar a usar o Amazon RDS.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
```

```
project("hello_rds")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                       # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rds.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_rds.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
```

```
#include <aws/rds/model/DescribeDBInstancesRequest.h>
#include <iostream>

/*
 *  A "Hello Rds" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client and
 *  describes the Amazon RDS instances.
 *
 *  main function
 *
 *  Usage: 'hello_rds'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);
        Aws::String marker;
        std::vector<Aws::String> instanceDBIDs;

        do {
            Aws::RDS::Model::DescribeDBInstancesRequest request;

            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
                    rdsClient.DescribeDBInstances(request);

            if (outcome.IsSuccess()) {
                for (auto &instance: outcome.GetResult().GetDBInstances()) {
                    instanceDBIDs.push_back(instance.GetDBInstanceIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
```

```
            } else {
                result = 1;
                std::cerr << "Error with RDS::DescribeDBInstances. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << instanceDBIDs.size() << " RDS instances found." << std::endl;
        for (auto &instanceDBID: instanceDBIDs) {
            std::cout << "   Instance: " << instanceDBID << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para C++ .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de banco de dados e definir os valores dos parâmetros.
- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.
- Exclua a instância e o grupo de parâmetros.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon RDS instance and demonstrates several operations
//! on that instance.
/*!
 \sa gettingStartedWithDBInstances()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::gettingStartedWithDBInstances(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon RDS)"
              << std::endl;
    std::cout << "get started with DB instances demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB parameter group named '" <<
              PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB parameter group already exists.
        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
```

```
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' does not exist." << std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available engine versions for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available database engine versions for " << DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family << std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
```

```
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB parameter group.
        Aws::RDS::Model::CreateDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example parameter group.");

        Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
                client.CreateDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your parameter group."
              << std::endl;

    Aws::String marker;
    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB parameter group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX, NO_SOURCE,
                         autoIncrementParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
```

```
                    autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value in the range ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the group.
        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
```

```
   std::cout
           << "You can get a list of parameters you've set by specifying a source
of 'user'."
           << std::endl;

   Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
   // 6.  Display the modified parameters in the group.
   if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
userParameters,
                        client)) {
       cleanUpResources(PARAMETER_GROUP_NAME, "", client);
       return false;
   }

   for (const auto &userParameter: userParameters) {
       std::cout << "  " << userParameter.GetParameterName() << ", " <<
                 userParameter.GetDescription() << ", parameter value - "
                 << userParameter.GetParameterValue() << std::endl;
   }

   printAsterisksLine();
   std::cout << "Checking for an existing DB instance." << std::endl;

   Aws::RDS::Model::DBInstance dbInstance;
   // 7.  Check if the DB instance already exists.
   if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
       cleanUpResources(PARAMETER_GROUP_NAME, "", client);
       return false;
   }

   if (dbInstance.DbInstancePortHasBeenSet()) {
       std::cout << "The DB instance already exists." << std::endl;
   }
   else {
       std::cout << "Let's create a DB instance." << std::endl;
       const Aws::String administratorName = askQuestion(
               "Enter an administrator username for the database: ");
       const Aws::String administratorPassword = askQuestion(
               "Enter a password for the administrator (at least 8 characters): ");
       Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

       // 8.  Get a list of available engine versions.
       if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
                                client)) {
```

```
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group are:" <<
std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
{
            std::cout << "  " << index << ": " << engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                                1];

        Aws::String dbInstanceClass;
        // 9.  Get a list of micro instance classes.
        if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
                                        engineVersion.GetEngineVersion(),
                                        dbInstanceClass,
                                        client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB instance is configured to use your custom parameter
group '"
                  << PARAMETER_GROUP_NAME << "',\n"
                  << "selected engine version " << engineVersion.GetEngineVersion()
                  << ",\n"
                  << "selected DB instance class '" << dbInstanceClass << "',"
                  << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
                  << " storage.\nThis typically takes several minutes." <<
std::endl;
```

```
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
```

```
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    printAsterisksLine();

    // 12. Display the connection string that can be used to connect a 'mysql' shell
 to the database.
    displayConnection(dbInstance);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB instance (y/n)? ")) {
        Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
 std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 13. Create a snapshot of the DB instance.
            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
```

```
            }
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for snapshot to become available." << std::endl;

        Aws::RDS::Model::DBSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 14. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
```

```
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB instance and parameter group (y/n)? ")) {
        result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
    }

    return result;
}


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
```

```
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
```

```
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,
                                      Aws::Vector<Aws::RDS::Model::DBEngineVersion>
 &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    } while (!marker.empty());


    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

```
//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
 {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
```

```
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::cleanUpResources(const Aws::String &parameterGroupName,
                                   const Aws::String &dbInstanceIdentifier,
                                   const Aws::RDS::RDSClient &client) {
    bool result = true;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 15. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
```

```
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        std::cout
                << "Waiting for DB instance to delete before deleting the parameter
group."
                << std::endl;
        std::cout << "This may take a while." << std::endl;

        int counter = 0;
        Aws::RDS::Model::DBInstance dbInstance;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 800) {
                std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                          << " seconds." << std::endl;
                return false;
            }

            dbInstance = Aws::RDS::Model::DBInstance();
            // 16. Wait for the DB instance to be deleted.
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }

            if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) == 0)
{
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
```

```
        } while (dbInstance.DBInstanceIdentifierHasBeenSet());
    }

    if (!parameterGroupName.empty()) {
        // 17. Delete the parameter group.
        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots

- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Ações

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBName(DB_NAME);
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetEngine(engineVersion.GetEngine());
    request.SetEngineVersion(engineVersion.GetEngineVersion());
    request.SetDBInstanceClass(dbInstanceClass);
    request.SetStorageType(DB_STORAGE_TYPE);
    request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
            client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
                  << std::endl;
```

```
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para C++ .

`CreateDBParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBParameterGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example parameter group.");

        Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
                client.CreateDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully created."
                      << std::endl;
        }
```

```
        else {
            std::cerr << "Error with RDS::CreateDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte CreateDBParameterGroup em Referência AWS SDK para C++ da API.

`CreateDBSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBSnapshot`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
```

```
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para C++ .

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
            else {
```

```
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para C++ .

`DeleteDBParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBParameterGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBParameterGroupRequest request;
    request.SetDBParameterGroupName(parameterGroupName);

    Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
            client.DeleteDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully deleted."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
```

```
        }
```

- Para obter detalhes da API, consulte DeleteDBParameterGroup em Referência AWS SDK para C++ da API.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,
                                      Aws::Vector<Aws::RDS::Model::DBEngineVersion>
 &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
```

```
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions em Referência AWS SDK para C++ da API.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
```

```
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para C ++ .

`DescribeDBParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBParameterGroups`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Para obter detalhes da API, consulte DescribeDBParameterGroups em Referência AWS SDK para C++ da API.

`DescribeDBParameters`

O código de exemplo a seguir mostra como usar `DescribeDBParameters`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
```

```
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para C++ .

`DescribeDBSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBSnapshots`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
```

```
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência da API AWS SDK para C++ .

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
```

```
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
 {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database engine
 are:"
              << std::endl;
```

```
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptions em Referência AWS SDK para C++ da API.

`ModifyDBParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBParameterGroup`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
```

```
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obter detalhes da API, consulte ModifyDBParameterGroup em Referência AWS SDK para C++ da API.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar uma API REST C++ que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon RDS Data Service usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ Amazon RDS Data Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar uma API REST C++ que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon Rekognition usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Amazon Rekognition.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon Rekognition

O exemplo de código a seguir mostra como começar a usar o Amazon Rekognition.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
```

```
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_rekognition.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rekognition/RekognitionClient.h>
#include <aws/rekognition/model/ListCollectionsRequest.h>
#include <iostream>

/*
 *  A "Hello Rekognition" starter application which initializes an Amazon
 Rekognition client and
 *  lists the Amazon Rekognition collections in the current account and region.
 *
 *  main function
 *
 *  Usage: 'hello_rekognition'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Rekognition::RekognitionClient rekognitionClient(clientConfig);
        Aws::Rekognition::Model::ListCollectionsRequest request;
        Aws::Rekognition::Model::ListCollectionsOutcome outcome =
                rekognitionClient.ListCollections(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String>& collectionsIds =
 outcome.GetResult().GetCollectionIds();
            if (!collectionsIds.empty()) {
                std::cout << "collectionsIds: " << std::endl;
                for (auto &collectionId : collectionsIds) {
                    std::cout << "- " << collectionId << std::endl;
                }
            } else {
                std::cout << "No collections found" << std::endl;
            }
        } else {
```

```
            std::cerr << "Error with ListCollections: " << outcome.GetError()
                      << std::endl;
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListCollections em Referência AWS SDK para C++ da API.

Ações

`DetectLabels`

O código de exemplo a seguir mostra como usar `DetectLabels`.

Para obter mais informações, consulte Detectar rótulos em uma imagem.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Detect instances of real-world entities within an image by using Amazon
 Rekognition
/*!
  \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
 containing an image.
  \param imageKey: The Amazon S3 key of an image object.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
                                       const Aws::String &imageKey,
```

```
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3Object;
    s3Object.SetBucket(imageBucket);
    s3Object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3Object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
 rekognitionClient.DetectLabels(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
 outcome.GetResult().GetLabels();
        if (labels.empty()) {
            std::cout << "No labels detected" << std::endl;
        } else {
            for (const Aws::Rekognition::Model::Label &label: labels) {
                std::cout << label.GetName() << ": " << label.GetConfidence() <<
 std::endl;
            }
        }
    } else {
        std::cerr << "Error while detecting labels: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DetectLabels em Referência AWS SDK para C++ da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos do Amazon S3 usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
```

```
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_s3.cpp.

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;
```

```
/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
 *  Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the S3
 service permits anonymous requests, thus the s3Client will return "success" and 0
 buckets even if you are unauthenticated, which can be confusing to a new user.
        auto provider = Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-
tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
            result = 1;
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                      << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
```

```
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para C++ da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
```

```
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param client: An S3 client.
          \return bool: Function succeeded.
        */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param key: The key for the object in the S3 bucket.
          \param client: An S3 client.
          \return bool: Function succeeded.
         */
        static bool
        deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
 &key,
                               Aws::S3::S3Client &client);
    }
}


//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
  \param bucketNamePrefix: A prefix for a bucket name.
  \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
  \param saveFilePath: Path for saving a downloaded S3 object.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
        const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = bucketNamePrefix +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                             clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                      "', in the region, '" << clientConfig.region << "'." <<
 std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
```

```
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
                Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                              uploadFilePath,
                                              std::ios_base::in |
                                              std::ios_base::binary);

        if (!input_data->is_open()) {
            std::cerr << "Error: unable to open file, '" << uploadFilePath << "'."
                      << std::endl;
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        }

        request.SetBody(input_data);

        Aws::S3::Model::PutObjectOutcome outcome =
                client.PutObject(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: putObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
            AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        } else {
            std::cout << "Added the object with the key, '" << key
                      << "', to the bucket, '"
                      << bucketName << "'." << std::endl;
        }
    }

    // 3. Download the object to a local file.
    {
        Aws::S3::Model::GetObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        Aws::S3::Model::GetObjectOutcome outcome =
```

```
                client.GetObject(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: getObject: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        } else {
            std::cout << "Downloaded the object with the key, '" << key
                      << "', in the bucket, '"
                      << bucketName << "'." << std::endl;

            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                    GetBody();
            Aws::OFStream outStream(saveFilePath,
                                    std::ios_base::out | std::ios_base::binary);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath << "'."
                          << std::endl;
            } else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file '"
                          << saveFilePath << "'." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
                .WithKey(copiedToKey)
                .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
                client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: copyObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Copied the object with the key, '" << key
                      << "', to the key, '" << copiedToKey
                      << ", in the bucket, '" << bucketName << "'." << std::endl;
```

```
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsV2Request request;
        request.WithBucket(bucketName);

        Aws::String continuationToken;
        Aws::Vector<Aws::S3::Model::Object> allObjects;

        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
                    request);

            if (!outcome.IsSuccess()) {
                std::cerr << "Error: ListObjects: " <<
                          outcome.GetError().GetMessage() << std::endl;
                break;
            } else {
                Aws::Vector<Aws::S3::Model::Object> objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(), objects.end());
                continuationToken = outcome.GetResult().GetContinuationToken();
            }
        } while (!continuationToken.empty());

        std::cout << allObjects.size() << " objects in the bucket, '" << bucketName
                  << "':" << std::endl;

        for (Aws::S3::Model::Object &object: allObjects) {
            std::cout << "     '" << object.GetKey() << "'" << std::endl;
        }
    }

    // 6. Delete all objects in the bucket.
    // All objects in the bucket must be deleted before deleting the bucket.
    AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
    AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

    // 7. Delete the bucket.
```

```
    return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
                  << "', from the bucket, '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações

`AbortMultipartUpload`

O código de exemplo a seguir mostra como usar `AbortMultipartUpload`.

SDK para C++

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
```

```
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AbortMultipartUpload em Referência AWS SDK para C++ da API.

`CompleteMultipartUpload`

O código de exemplo a seguir mostra como usar `CompleteMultipartUpload`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
```

```
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

 const Aws::String &key,

 const Aws::String &uploadID,

 const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

 const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}
```

- Para obter detalhes da API, consulte CompleteMultipartUpload em Referência AWS SDK para C++ da API.

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
 &fromBucket, const Aws::String &toBucket,
                            const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

    } else {
        std::cout << "Successfully copied " << objectKey << " from " << fromBucket
 <<
                  " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK para C++ da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                        clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
                  " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK para C++ da API.

`CreateMultipartUpload`

O código de exemplo a seguir mostra como usar `CreateMultipartUpload`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return uploadID;
}
```

- Para obter detalhes da API, consulte CreateMultipartUpload em Referência AWS SDK para C ++ da API.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteBucket em Referência AWS SDK para C++ da API.

`DeleteBucketPolicy`

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
 client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteBucketPolicy em Referência AWS SDK para C++ da API.

`DeleteBucketWebsite`

O código de exemplo a seguir mostra como usar `DeleteBucketWebsite`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
            client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteBucketWebsite em Referência AWS SDK para C++ da API.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
            .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteObject em Referência AWS SDK para C++ da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

 deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
            client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }

        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK para C++ da API.

GetBucketAcl

O código de exemplo a seguir mostra como usar `GetBucketAcl`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
            s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();
```

```
            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) << std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
*/
```

```
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}
```

- Para obter detalhes da API, consulte GetBucketAcl em Referência AWS SDK para C++ da API.

`GetBucketPolicy`

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                 const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
            s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n" <<
```

```
                    policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetBucketPolicy em Referência AWS SDK para C++ da API.

GetBucketWebsite

O código de exemplo a seguir mostra como usar `GetBucketWebsite`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
            s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult = outcome.GetResult();
```

```
        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
                  << websiteResult.GetErrorDocument().GetKey()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetBucketWebsite em Referência AWS SDK para C++ da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
```

```
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK para C++ da API.

`GetObjectAcl`

O código de exemplo a seguir mostra como usar `GetObjectAcl`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
            s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
```

```
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                      << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) << std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
```

```
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                   "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
```

```
        default:
            return "Permission unknown";
    }
}
```

- Para obter detalhes da API, consulte GetObjectAcl em Referência AWS SDK para C++ da API.

GetObjectAttributes

O código de exemplo a seguir mostra como usar `GetObjectAttributes`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of the
 object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
```

```
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
```

```
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
```

```
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object parts."
 <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}
```

- Para obter detalhes da API, consulte GetObjectAttributes em Referência AWS SDK para C++ da API.

`ListBuckets`

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
 buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para C++ da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
```

```
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                    outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK para C++ API.

`PutBucketAcl`

O código de exemplo a seguir mostra como usar `PutBucketAcl`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
 &ownerID,
                              const Aws::String &granteePermission,
                              const Aws::String &granteeType, const Aws::String
 &granteeID,
                              const Aws::String &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
```

```
    Aws::S3::Model::PutBucketAclOutcome outcome =
            s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
*/

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
```

```
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- Para obter detalhes da API, consulte PutBucketAcl em Referência AWS SDK para C++ da API.

`PutBucketPolicy`

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
            Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
            s3Client.PutBucketPolicy(request);
```

```
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}


//! Build a policy JSON string.
/*!
  \param userArn: Aws user Amazon Resource Name (ARN).
      For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
  \param bucketName: Name of a bucket.
  \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
            "{\n"
            "   \"Version\":\"2012-10-17\",\n"
            "   \"Statement\":[\n"
            "       {\n"
            "           \"Sid\": \"1\",\n"
            "           \"Effect\": \"Allow\",\n"
            "           \"Principal\": {\n"
            "               \"AWS\": \""
            + userArn +
            "\"\n""           },\n"
            "           \"Action\": [ \"s3:getObject\" ],\n"
            "           \"Resource\": [ \"arn:aws:s3:::"
            + bucketName +
            "/*\" ]\n"
            "       }\n"
            "   ]\n"
            "}";
}
```

- Para obter detalhes da API, consulte PutBucketPolicy em Referência AWS SDK para C++ da API.

`PutBucketWebsite`

O código de exemplo a seguir mostra como usar `PutBucketWebsite`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
            client.PutBucketWebsite(request);
```

```
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte PutBucketWebsite em Referência AWS SDK para C++ da API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
 needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
```

```
                                        fileName.c_str(),
                                        std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Added object '" << fileName << "' to bucket '"
                  << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK para C++ da API.

`PutObjectAcl`

O código de exemplo a seguir mostra como usar `PutObjectAcl`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
 &objectKey, const Aws::String &ownerID,
```

```
                                const Aws::String &granteePermission, const
Aws::String &granteeType,
                                const Aws::String &granteeID, const Aws::String
&granteeEmailAddress,
                                const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
   Aws::S3::S3Client s3Client(clientConfig);

   Aws::S3::Model::Owner owner;
   owner.SetID(ownerID);

   Aws::S3::Model::Grantee grantee;
   grantee.SetType(setGranteeType(granteeType));

   if (!granteeEmailAddress.empty()) {
       grantee.SetEmailAddress(granteeEmailAddress);
   }

   if (!granteeID.empty()) {
       grantee.SetID(granteeID);
   }

   if (!granteeURI.empty()) {
       grantee.SetURI(granteeURI);
   }

   Aws::S3::Model::Grant grant;
   grant.SetGrantee(grantee);
   grant.SetPermission(setGranteePermission(granteePermission));

   Aws::Vector<Aws::S3::Model::Grant> grants;
   grants.push_back(grant);

   Aws::S3::Model::AccessControlPolicy acp;
   acp.SetOwner(owner);
   acp.SetGrants(grants);

   Aws::S3::Model::PutObjectAclRequest request;
   request.SetAccessControlPolicy(acp);
   request.SetBucket(bucketName);
   request.SetKey(objectKey);

   Aws::S3::Model::PutObjectAclOutcome outcome =
           s3Client.PutObjectAcl(request);
```

```
    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
                  << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param access: Human readable string.
 \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
```

```
    return Aws::S3::Model::Type::NOT_SET;
}
```

- Para obter detalhes da API, consulte PutObjectAcl em Referência AWS SDK para C++ da API.

UploadPart

O código de exemplo a seguir mostra como usar `UploadPart`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the checksum
 algorithm,
                           ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
 &uploadID,
                                                         int partNumber,
```

```
 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                        const Aws::String
 &calculatedHash,
                                                        const
 std::shared_ptr<Aws::IOStream> &body,
                                                        const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}
```

- Para obter detalhes da API, consulte UploadPart em Referência AWS SDK para C++ da API.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere um URL pré-assinado para baixar um objeto.

```
//! Routine which demonstrates creating a pre-signed URL to download an object from
 an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param expirationSeconds: Expiration in seconds for pre-signed URL.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String &bucketName,
                                                      const Aws::String &key,
                                                      uint64_t expirationSeconds,
                                                      const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_GET,
                                       expirationSeconds);
}
```

Baixe usando a libcurl.

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                 Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }
```

```
    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
 std::endl;
        return false;
    }

    return true;
}
```

Gere um URL pré-assinado para carregar um objeto.

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String &bucketName,
                                                      const Aws::String &key,
                                                      uint64_t expirationSeconds,
                                                      const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_PUT,
                                       expirationSeconds);
}
```

Carregue usando a libcurl.

```
static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to put an object in a bucket.
  \param data: Body of the putObject request.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                 const Aws::String &data) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
```

```
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
```

```
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                  << std::endl;
        return false;
    }
}
```

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Trabalhar com a integridade de objetos do Amazon S3

O exemplo de código a seguir mostra como trabalhar com os recursos de integridade de objetos do S3.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstra os recursos de integridade do Amazon S3.

```
//! Routine which runs the S3 object integrity workflow.
/*!
   \param clientConfig: Aws client configuration.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
        const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
 std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
 std::endl;
```

```
        return false;
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
              << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << "  " << hashMethod << " - " << stringForHashMethod(hashMethod)
                  << std::endl;
    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                                          SHA256);


    gUseCalculatedChecksum = !askYesNoQuestion(
            "Let the SDK calculate the checksum for you? (y/n) ");

    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
              << std::endl;
    std::cout << "Object integrity will be verified using the "
              << stringForHashMethod(chosenHashMethod) << " algorithm."
              << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout
                << "A checksum computed by this workflow will be used for object
integrity verification,"
                << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout
                << "A checksum computed by the SDK will be used for object integrity
verification."
                << std::endl;
    }
```

```
    pressEnterToContinue();
    printAsterisksLine();

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                           inputData, chosenHashMethod == DEFAULT,
                           *client)) {
        std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
```

```
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                            putObjectHashMethod, retrievedHash,
                            nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                         "PutObject upload", putObjectHashMethod);


    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32;  // The default hash method for the
TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "
                  << stringForHashMethod(transferManagerHashMethod)
                  << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
chosenHashMethod == DEFAULT,
                                 client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
```

```
   std::vector<Aws::String> retrievedTransferManagerPartHashes;
   Aws::String retrievedTransferManagerFinalHash;

   // Retrieve all the hashes for the TransferManager upload.
   if (!retrieveObjectHash(bucketName, key,
                           transferManagerHashMethod,
                           retrievedTransferManagerFinalHash,
                           &retrievedTransferManagerPartHashes, *client)) {
       std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   AwsDoc::S3::Hasher locallyCalculatedFinalHash;
   std::vector<Aws::String> locallyCalculatedPartHashes;

   // Calculate the hashes locally to demonstrate how TransferManager hashes are
computed.
   if (!calculatePartHashesForFile(transferManagerHashMethod, MULTI_PART_TEST_FILE,
                                   UPLOAD_BUFFER_SIZE,
                                   locallyCalculatedFinalHash,
                                   locallyCalculatedPartHashes)) {
       std::cerr << "Exiting because of an error in calculatePartHashesForFile." <<
std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   verifyHashingResults(retrievedTransferManagerFinalHash,
                        locallyCalculatedFinalHash, "TransferManager upload",
                        transferManagerHashMethod,
                        retrievedTransferManagerPartHashes,
                        locallyCalculatedPartHashes);

   printAsterisksLine();

   key = "mp_";
   key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

   multiPartUploadExplanations(key, chosenHashMethod);

   pressEnterToContinue();
```

```
   std::shared_ptr<Aws::IOStream> largeFileInputData =
           Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                         MULTI_PART_TEST_FILE,
                                         std::ios_base::in |
                                         std::ios_base::binary);

   if (!largeFileInputData->good()) {
       std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
   if (multipartUploadHashMethod == DEFAULT) {
       multipartUploadHashMethod = MD5;  // The default hash method for multipart
uploads is MD5.

       std::cout << "The default checksum algorithm for multipart upload is "
                 << stringForHashMethod(putObjectHashMethod)
                 << std::endl;
   }

   AwsDoc::S3::Hasher hashData;
   std::vector<Aws::String> partHashes;

   if (!doMultipartUpload(bucketName, key,
                          multipartUploadHashMethod,
                          largeFileInputData, chosenHashMethod == DEFAULT,
                          hashData,
                          partHashes,
                          *client)) {
       std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   std::cout << "Finished multipart upload of with hash method " <<
             stringForHashMethod(multipartUploadHashMethod) << std::endl;

   std::cout << "Now we will retrieve the checksums from the server." << std::endl;

   retrievedHash.clear();
```

```
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                            multipartUploadHashMethod,
                            retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
 multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                         multipartUploadHashMethod,
                         retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
 workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
 std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object integrity
 hashing methods.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashData: The hash value that will be associated with the uploaded object.
   \param hashMethod: The hashing algorithm to use when calculating the hash value.
   \param body: The data content of the object being uploaded.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
 &key,
                                   const Aws::String &hashData,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &body,
```

```
                                    bool useDefaultHashMethod,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {

request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {
        std::cout << "Object successfully uploaded." << std::endl;
    } else {
        std::cerr << "Error uploading object." <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
```

```
}


// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of the
 object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
```

```
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
```

```
                const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
 result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
 NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object parts."
 <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
 \param retrievedHash The hash value retrieved from the remote source.
 \param localHash The hash value calculated locally.
```

```
 \param uploadtype The type of upload (e.g., "multipart", "single-part").
 \param hashMethod The hashing method used (e.g., MD5, SHA-256).
 \param retrievedPartHashes (Optional) The list of hashes for the individual parts
 retrieved from the remote source.
 \param localPartHashes (Optional) The list of hashes for the individual parts
 calculated locally.
 */
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                      const Hasher &localHash,
                                      const Aws::String &uploadtype,
                                      HASH_METHOD hashMethod,
                                      const std::vector<Aws::String>
 &retrievedPartHashes,
                                      const std::vector<Aws::String>
 &localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash <<
 std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
 retrieved."
                  << std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << "  Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
 std::endl;
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }
```

```
    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do not
 match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
 "'"
                      << std::endl;
            std::cerr << "Remote number of hashes - '"
                      << retrievedPartHashes.size()
                      << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not match
 for part " << i + 1
                          << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                          << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                          << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
 hashes all match!" << std::endl;
    }

}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &,
                                         const
 Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" << std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                          const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &handle) {
```

```
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
 std::endl;
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's Transfer
 Manager.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashMethod: The hashing algorithm to use when calculating the hash value.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    bool useDefaultHashMethod,
                                    const std::shared_ptr<Aws::S3::S3Client>
 &client) {
    std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = client;
    transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
    if (!useDefaultHashMethod) {
        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
            transfer_config.checksumAlgorithm = getChecksumAlgorithmForHashMethod(
                    hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback = transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
 Aws::Transfer::TransferManager::Create(
            transfer_config);
```

```
    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle = transfer_manager-
>UploadFile(MULTI_PART_TEST_FILE,

           bucket, key,

           "text/plain",

           Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
            uploadHandle->GetStatus() == Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle->GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being uploaded
 to an S3 bucket.
/*!
   \param hashMethod: The hashing algorithm to use when calculating the hash values.
   \param fileName: The path to the file for which the part hashes will be
 calculated.
   \param bufferSize: The size of the buffer to use when reading the file.
   \param[out] hashDataResult: The Hasher object that will store the concatenated
 hash value.
   \param[out] partHashes: The vector that will store the calculated hash values for
 each part of the file.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                            const Aws::String &fileName,
                                            size_t bufferSize,
                                            AwsDoc::S3::Hasher &hashDataResult,
                                            std::vector<Aws::String> &partHashes) {
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
```

```
    size_t uploadedBytes = 0;


    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
 static_cast<std::streamsize>(std::min(buffer.size(), objectSize - uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
 preallocatedStreamBuf(buffer.data(),

 bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
                Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                                &preallocatedStreamBuf);
        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
 hashBuffer.GetUnderlyingData(),
                               hashBuffer.GetUnderlyingData() +
 hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
```

```
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the checksum
 algorithm,
                           ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String &bucket,
                                                         const Aws::String &key,
```

```
                                                                const Aws::String
 &uploadID,
                                                                int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                                const Aws::String
 &calculatedHash,
                                                                const
 std::shared_ptr<Aws::IOStream> &body,
                                                                const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}
```

```
//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

 const Aws::String &key,
```

```
 const Aws::String &uploadID,

 const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

 const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

//! Routine which performs a multi-part upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the concatenated
 hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &ioStream,
```

```
                                    bool useDefaultHashMethod,
                                    AwsDoc::S3::Hasher &hashDataResult,
                                    std::vector<Aws::String> &partHashes,
                                    const Aws::S3::S3Client &client) {
   // Get object size.
   ioStream->seekg(0, ioStream->end);
   size_t objectSize = ioStream->tellg();
   ioStream->seekg(0, ioStream->beg);

   Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
   if (!useDefaultHashMethod) {
       if (hashMethod != MD5) {
           checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
       }
   }
   Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
client);
   if (uploadID.empty()) {
       return false;
   }

   std::vector<unsigned char> totalHashBuffer;
   bool uploadSucceeded = true;
   std::streamsize uploadedBytes = 0;
   int partNumber = 1;
   Aws::Vector<Aws::S3::Model::CompletedPart> parts;
   while (uploadedBytes < objectSize) {
       std::cout << "Uploading part " << partNumber << "." << std::endl;

       std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
       std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),

objectSize - uploadedBytes));
       ioStream->read((char *) buffer.data(), bytesToRead);
       Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
       std::shared_ptr<Aws::IOStream> body =
               Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                              &preallocatedStreamBuf);
```

```
        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            uploadSucceeded = false;
            break;
        }

        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                               hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

        Aws::String calculatedHash;
        if (gUseCalculatedChecksum) {
            calculatedHash = base64HashString;
        }
        Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,
                                                                        client);
        if (uploadPartOutcome.IsSuccess()) {
            const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
            Aws::S3::Model::CompletedPart completedPart;
            completedPart.SetETag(uploadPartResult.GetETag());
            completedPart.SetPartNumber(partNumber);
            switch (hashMethod) {
                case AwsDoc::S3::MD5:
                    break; // Do nothing.
                case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:
```

```
completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                    break;
            }

            parts.push_back(completedPart);
        } else {
            std::cerr << "Error uploading part. " <<
                      uploadPartOutcome.GetError().GetMessage() << std::endl;
            uploadSucceeded = false;
            break;
        }

        uploadedBytes += bytesToRead;
        partNumber++;
    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                 key,

                                 uploadID,

                                 parts,

                                 client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod)) {
```

```
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
        } else {
            std::cerr << "Error completing multipart upload." <<
                      completeMultipartUploadOutcome.GetError().GetMessage()
                      << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            return "Default";
        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
```

```
    Aws::S3::Model::ChecksumAlgorithm result =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not valid."
 << std::endl;
            break;  // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;

    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                         const Aws::S3::S3ClientConfiguration &clientConfiguration)
 {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
```

```
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                               clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
  \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
            << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
 integrity workflow."
            << std::endl;
    printAsterisksLine();
    std::cout
            << "This workflow demonstrates how Amazon S3 uses checksum values to
 verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
            << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
            << "By default it calculates a checksum that is uploaded with an object.
\n"
            << "The default checksum algorithm for PutObject and MultiPart upload is
 an MD5 hash.\n"
            << "The default checksum algorithm for TransferManager uploads is a
 CRC32 checksum."
            << std::endl;
    std::cout
            << "You can override the default behavior, requiring one of the
 following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting the
 SDK calculate the value."
              << std::endl;
    std::cout
```

```
            << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
            << std::endl;

    std::cout
            << "This workflow will locally compute checksums for files uploaded to
 an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
            << "This is done to provide demonstration code for how the checksums are
 calculated."
            << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
 object uploads."
              << std::endl;
}

//! Console interaction which explains the PutObject results.
/*!
*/
void AwsDoc::S3::explainPutObjectResults() {

    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have failed."
              << std::endl;
    std::cout
            << "The checksums calculated by the server have been retrieved using the
 GetObjectAttributes."
            << std::endl;
    std::cout
            << "The locally calculated checksums have been verified against the
 retrieved checksums."
            << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
  \param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
        const Aws::String &objectKey) {
    std::cout
            << "Now the workflow will demonstrate object integrity for
 TransferManager multi-part uploads."
```

```
            << std::endl;
    std::cout
            << "The AWS C++ SDK has a TransferManager class which simplifies
 multipart uploads."
            << std::endl;
    std::cout
            << "The following code lets the TransferManager handle much of the
 checksum configuration."
            << std::endl;

    std::cout << "An object with the key '" << objectKey
              << " will be uploaded by the TransferManager using a "
              << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
 calculate the hash value."
                  << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
  \param objectKey: The key for the object being uploaded.
  \param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                             HASH_METHOD chosenHashMethod) {
    std::cout
            << "Now we will provide an in-depth demonstration of multi-part
 uploading by calling the multi-part upload APIs directly."
            << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when uploading
 large files."
              << std::endl;
    std::cout
            << "In the following code, the checksums are also calculated locally and
 then compared."
            << std::endl;
    std::cout
            << "For multi-part uploads, a checksum is uploaded with each part. The
 final checksum is a concatenation of"
```

```
            << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
            << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,\""
            << " in the section \"Using part-level checksums for multipart uploads
\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
              stringForHashMethod(chosenHashMethod) << " uploading to with object
 key\n"
              << "'" << objectKey << "'," << std::endl;

}

//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
 file.
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                                | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
 MULTI_PART_TEST_FILE <<
                  std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in

                                  | std::ios::binary);
    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
                  std::endl;
        return false;
    }
```

```
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();

    return true;
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - AbortMultipartUpload
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - DeleteObject
  - GetObjectAttributes
  - PutObject
  - UploadPart

Exemplos do Secrets Manager usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetSecretValue`

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve an AWS Secrets Manager encrypted secret.
/*!
  \param secretID: The ID for the secret.
  \return bool: Function succeeded.
 */
bool AwsDoc::SecretsManager::getSecretValue(const Aws::String &secretID,
                                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SecretsManager::SecretsManagerClient
 secretsManagerClient(clientConfiguration);

    Aws::SecretsManager::Model::GetSecretValueRequest request;
    request.SetSecretId(secretID);

    Aws::SecretsManager::Model::GetSecretValueOutcome getSecretValueOutcome =
 secretsManagerClient.GetSecretValue(
            request);
    if (getSecretValueOutcome.IsSuccess()) {
        std::cout << "Secret is: "
                  << getSecretValueOutcome.GetResult().GetSecretString() <<
 std::endl;
    }
    else {
        std::cerr << "Failed with Error: " << getSecretValueOutcome.GetError()
                  << std::endl;
    }
```

```
    return getSecretValueOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetSecretValue em Referência AWS SDK para C++ da API.

Exemplos do Amazon SES usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateReceiptFilter`

O código de exemplo a seguir mostra como usar `CreateReceiptFilter`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt filter..
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param cidr: IP address or IP address range in Classless Inter-Domain Routing
 (CIDR) notation.
  \param policy: Block or allow enum of type ReceiptFilterPolicy.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::String &cidr,
                                      Aws::SES::Model::ReceiptFilterPolicy policy,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::CreateReceiptFilterRequest createReceiptFilterRequest;
    Aws::SES::Model::ReceiptFilter receiptFilter;
    Aws::SES::Model::ReceiptIpFilter receiptIpFilter;
    receiptIpFilter.SetCidr(cidr);
    receiptIpFilter.SetPolicy(policy);
    receiptFilter.SetName(receiptFilterName);
    receiptFilter.SetIpFilter(receiptIpFilter);
    createReceiptFilterRequest.SetFilter(receiptFilter);
    Aws::SES::Model::CreateReceiptFilterOutcome createReceiptFilterOutcome =
 sesClient.CreateReceiptFilter(
            createReceiptFilterRequest);
    if (createReceiptFilterOutcome.IsSuccess()) {
        std::cout << "Successfully created receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt filter: " <<
                  createReceiptFilterOutcome.GetError().GetMessage() << std::endl;
    }

    return createReceiptFilterOutcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateReceiptFilter em Referência AWS SDK para C++ da API.

`CreateReceiptRule`

O código de exemplo a seguir mostra como usar `CreateReceiptRule`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
  \param receiptRuleName: The name for the receipt rule.
  \param s3BucketName: The name of the S3 bucket for incoming mail.
  \param s3ObjectKeyPrefix: The prefix for the objects in the S3 bucket.
  \param ruleSetName: The name of the rule set where the receipt rule is added.
  \param recipients: Aws::Vector of recipients.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &s3BucketName,
                                    const Aws::String &s3ObjectKeyPrefix,
                                    const Aws::String &ruleSetName,
                                    const Aws::Vector<Aws::String> &recipients,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleRequest createReceiptRuleRequest;

    Aws::SES::Model::S3Action s3Action;
    s3Action.SetBucketName(s3BucketName);
    s3Action.SetObjectKeyPrefix(s3ObjectKeyPrefix);

    Aws::SES::Model::ReceiptAction receiptAction;
    receiptAction.SetS3Action(s3Action);

    Aws::SES::Model::ReceiptRule receiptRule;
    receiptRule.SetName(receiptRuleName);
    receiptRule.WithRecipients(recipients);
```

```
    Aws::Vector<Aws::SES::Model::ReceiptAction> receiptActionList;
    receiptActionList.emplace_back(receiptAction);
    receiptRule.SetActions(receiptActionList);

    createReceiptRuleRequest.SetRuleSetName(ruleSetName);
    createReceiptRuleRequest.SetRule(receiptRule);

    auto outcome = sesClient.CreateReceiptRule(createReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateReceiptRule em Referência AWS SDK para C++ da API.

`CreateReceiptRuleSet`

O código de exemplo a seguir mostra como usar `CreateReceiptRuleSet`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
```

```
  \param ruleSetName: The name of the rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRuleSet(const Aws::String &ruleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleSetRequest createReceiptRuleSetRequest;

    createReceiptRuleSetRequest.SetRuleSetName(ruleSetName);

    Aws::SES::Model::CreateReceiptRuleSetOutcome outcome =
 sesClient.CreateReceiptRuleSet(
            createReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule set." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateReceiptRuleSet em Referência AWS SDK para C ++ da API.

`CreateTemplate`

O código de exemplo a seguir mostra como usar `CreateTemplate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateTemplateRequest createTemplateRequest;
    Aws::SES::Model::Template aTemplate;

    aTemplate.SetTemplateName(templateName);
    aTemplate.SetHtmlPart(htmlPart);
    aTemplate.SetSubjectPart(subjectPart);
    aTemplate.SetTextPart(textPart);

    createTemplateRequest.SetTemplate(aTemplate);

    Aws::SES::Model::CreateTemplateOutcome outcome = sesClient.CreateTemplate(
            createTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created template." << templateName << "."
                  << std::endl;
    }
```

```
    else {
        std::cerr << "Error creating template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTemplate em Referência AWS SDK para C++ da API.

DeleteIdentity

O código de exemplo a seguir mostra como usar `DeleteIdentity`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete the specified identity (an email address or a domain).
/*!
  \param identity: The identity to delete.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteIdentity(const Aws::String &identity,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteIdentityRequest deleteIdentityRequest;

    deleteIdentityRequest.SetIdentity(identity);

    Aws::SES::Model::DeleteIdentityOutcome outcome = sesClient.DeleteIdentity(
```

```
            deleteIdentityRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted identity." << std::endl;
    }
    else {
        std::cerr << "Error deleting identity. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteIdentity em Referência AWS SDK para C++ da API.

`DeleteReceiptFilter`

O código de exemplo a seguir mostra como usar `DeleteReceiptFilter`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt filter.
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
```

```
    Aws::SES::Model::DeleteReceiptFilterRequest deleteReceiptFilterRequest;

    deleteReceiptFilterRequest.SetFilterName(receiptFilterName);

    Aws::SES::Model::DeleteReceiptFilterOutcome outcome =
 sesClient.DeleteReceiptFilter(
            deleteReceiptFilterRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error deleting receipt filter. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteReceiptFilter em Referência AWS SDK para C++ da API.

`DeleteReceiptRule`

O código de exemplo a seguir mostra como usar `DeleteReceiptRule`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
  \param receiptRuleName: The name for the receipt rule.
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &receiptRuleSetName,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleRequest deleteReceiptRuleRequest;

    deleteReceiptRuleRequest.SetRuleName(receiptRuleName);
    deleteReceiptRuleRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleOutcome outcome = sesClient.DeleteReceiptRule(
            deleteReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule." << std::endl;
    }
    else {
        std::cout << "Error deleting receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteReceiptRule em Referência AWS SDK para C++ da API.

`DeleteReceiptRuleSet`

O código de exemplo a seguir mostra como usar `DeleteReceiptRuleSet`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRuleSet(const Aws::String &receiptRuleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleSetRequest deleteReceiptRuleSetRequest;

    deleteReceiptRuleSetRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleSetOutcome outcome =
 sesClient.DeleteReceiptRuleSet(
            deleteReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule set." << std::endl;
    }

    else {
        std::cerr << "Error deleting receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteReceiptRuleSet em Referência AWS SDK para C++ da API.

DeleteTemplate

O código de exemplo a seguir mostra como usar `DeleteTemplate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteTemplate(const Aws::String &templateName,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteTemplateRequest deleteTemplateRequest;

    deleteTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::DeleteTemplateOutcome outcome = sesClient.DeleteTemplate(
            deleteTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted template." << std::endl;
    }
    else {
        std::cerr << "Error deleting template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte DeleteTemplate em Referência AWS SDK para C++ da API.

GetTemplate

O código de exemplo a seguir mostra como usar `GetTemplate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Get a template's attributes.
/*!
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::getTemplate(const Aws::String &templateName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::GetTemplateRequest getTemplateRequest;

    getTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::GetTemplateOutcome outcome = sesClient.GetTemplate(
            getTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully got template." << std::endl;
    }
```

```
    else {
        std::cerr << "Error getting template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetTemplate em Referência AWS SDK para C++ da API.

ListIdentities

O código de exemplo a seguir mostra como usar `ListIdentities`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the identities associated with this account.
/*!
  \param identityType: The identity type enum. "NOT_SET" is a valid option.
  \param identities; A vector to receive the retrieved identities.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::listIdentities(Aws::SES::Model::IdentityType identityType,
                                 Aws::Vector<Aws::String> &identities,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::ListIdentitiesRequest listIdentitiesRequest;

    if (identityType != Aws::SES::Model::IdentityType::NOT_SET) {
```

```
        listIdentitiesRequest.SetIdentityType(identityType);
    }

    Aws::String nextToken; // Used for paginated results.
    do {
        if (!nextToken.empty()) {
            listIdentitiesRequest.SetNextToken(nextToken);
        }
        Aws::SES::Model::ListIdentitiesOutcome outcome = sesClient.ListIdentities(
                listIdentitiesRequest);

        if (outcome.IsSuccess()) {
            const auto &retrievedIdentities = outcome.GetResult().GetIdentities();
            if (!retrievedIdentities.empty()) {
                identities.insert(identities.cend(), retrievedIdentities.cbegin(),
                                  retrievedIdentities.cend());
            }
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error listing identities. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    return true;
}
```

- Para obter detalhes da API, consulte ListIdentities em Referência AWS SDK para C++ da API.

`ListReceiptFilters`

O código de exemplo a seguir mostra como usar `ListReceiptFilters`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List the receipt filters associated with this account.
/*!
  \param filters; A vector of "ReceiptFilter" to receive the retrieved filters.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SES::listReceiptFilters(Aws::Vector<Aws::SES::Model::ReceiptFilter>
 &filters,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::ListReceiptFiltersRequest listReceiptFiltersRequest;

    Aws::SES::Model::ListReceiptFiltersOutcome outcome =
 sesClient.ListReceiptFilters(
            listReceiptFiltersRequest);
    if (outcome.IsSuccess()) {
        auto &retrievedFilters = outcome.GetResult().GetFilters();
        if (!retrievedFilters.empty()) {
            filters.insert(filters.cend(), retrievedFilters.cbegin(),
                           retrievedFilters.cend());
        }
    }
    else {
        std::cerr << "Error retrieving IP address filters: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ListReceiptFilters em Referência AWS SDK para C++ da API.

SendEmail

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Send an email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param subject: Email subject.
  \param htmlBody: Email body as HTML. At least one body data is required.
  \param textBody: Email body as plain text. At least one body data is required.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendEmail(const Aws::Vector<Aws::String> &recipients,
                            const Aws::String &subject,
                            const Aws::String &htmlBody,
                            const Aws::String &textBody,
                            const Aws::String &senderEmailAddress,
                            const Aws::Vector<Aws::String> &ccAddresses,
                            const Aws::String &replyToAddress,
                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
```

```
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::Body message_body;
    if (!htmlBody.empty()) {
        message_body.SetHtml(
                Aws::SES::Model::Content().WithCharset("UTF-8").WithData(htmlBody));
    }

    if (!textBody.empty()) {
        message_body.SetText(
                Aws::SES::Model::Content().WithCharset("UTF-8").WithData(textBody));
    }

    Aws::SES::Model::Message message;
    message.SetBody(message_body);
    message.SetSubject(
            Aws::SES::Model::Content().WithCharset("UTF-8").WithData(subject));

    Aws::SES::Model::SendEmailRequest sendEmailRequest;
    sendEmailRequest.SetDestination(destination);
    sendEmailRequest.SetMessage(message);
    if (!senderEmailAddress.empty()) {
        sendEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendEmail(sendEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending message. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte SendEmail em Referência AWS SDK para C++ da API.

SendTemplatedEmail

O código de exemplo a seguir mostra como usar `SendTemplatedEmail`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Send a templated email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param templateName: The name of the template to use.
  \param templateData: Map of key-value pairs for replacing text in template.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendTemplatedEmail(const Aws::Vector<Aws::String> &recipients,
                                     const Aws::String &templateName,
                                     const Aws::Map<Aws::String, Aws::String>
 &templateData,
                                     const Aws::String &senderEmailAddress,
                                     const Aws::Vector<Aws::String> &ccAddresses,
                                     const Aws::String &replyToAddress,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
```

```
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::SendTemplatedEmailRequest sendTemplatedEmailRequest;
    sendTemplatedEmailRequest.SetDestination(destination);
    sendTemplatedEmailRequest.SetTemplate(templateName);

    std::ostringstream templateDataStream;
    templateDataStream << "{";
    size_t dataCount = 0;
    for (auto &pair: templateData) {
        templateDataStream << "\"" << pair.first << "\":\"" << pair.second << "\"";
        dataCount++;
        if (dataCount < templateData.size()) {
            templateDataStream << ",";
        }
    }
    templateDataStream << "}";

    sendTemplatedEmailRequest.SetTemplateData(templateDataStream.str());

    if (!senderEmailAddress.empty()) {
        sendTemplatedEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendTemplatedEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendTemplatedEmail(sendTemplatedEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent templated message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending templated message. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte SendTemplatedEmail em Referência AWS SDK para C++ da API.

`UpdateTemplate`

O código de exemplo a seguir mostra como usar `UpdateTemplate`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::updateTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Template templateValues;

    templateValues.SetTemplateName(templateName);
    templateValues.SetSubjectPart(subjectPart);
    templateValues.SetHtmlPart(htmlPart);
    templateValues.SetTextPart(textPart);
```

```
    Aws::SES::Model::UpdateTemplateRequest updateTemplateRequest;
    updateTemplateRequest.SetTemplate(templateValues);

    Aws::SES::Model::UpdateTemplateOutcome outcome =
 sesClient.UpdateTemplate(updateTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated template." << std::endl;
    } else {
        std::cerr << "Error updating template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateTemplate em Referência AWS SDK para C++ da API.

`VerifyEmailIdentity`

O código de exemplo a seguir mostra como usar `VerifyEmailIdentity`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Add an email address to the list of identities associated with this account and
//! initiate verification.
/*!
  \param emailAddress; The email address to add.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::verifyEmailIdentity(const Aws::String &emailAddress,
```

```
                        const Aws::Client::ClientConfiguration
 &clientConfiguration)
{
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::VerifyEmailIdentityRequest verifyEmailIdentityRequest;

    verifyEmailIdentityRequest.SetEmailAddress(emailAddress);

    Aws::SES::Model::VerifyEmailIdentityOutcome outcome =
 sesClient.VerifyEmailIdentity(verifyEmailIdentityRequest);

    if (outcome.IsSuccess())
    {
        std::cout << "Email verification initiated." << std::endl;
    }

    else
    {
        std::cerr << "Error initiating email verification. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte VerifyEmailIdentity em Referência AWS SDK para C++ da API.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para C++

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Aurora Sem Servidor.

Para obter o código-fonte completo e instruções sobre como configurar uma API REST C++ que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon SNS usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon SNS

O exemplo de código a seguir mostra como começar a usar o Amazon SNS.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_sns.cpp.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << " topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
```

```
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para C++ da API.

Ações

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);
```

```
    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateTopic em Referência AWS SDK para C++ da API.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteTopic em Referência AWS SDK para C++ da API.

`GetSMSAttributes`

O código de exemplo a seguir mostra como usar `GetSMSAttributes`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve the default settings for sending SMS messages from your AWS account by
 using
//! Amazon Simple Notification Service (Amazon SNS).
```

```
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Consulte detalhes da API em GetSMSAttributes na Referência da API AWS SDK para C++ .

`GetTopicAttributes`

O código de exemplo a seguir mostra como usar `GetTopicAttributes`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetTopicAttributes em Referência AWS SDK para C++ da API.

ListSubscriptions

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
```

```
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
                      << outcome.GetError().GetMessage()
                      <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- Para obter detalhes da API, consulte ListSubscriptions em Referência AWS SDK para C++ da API.

ListTopics

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " << outcome.GetError().GetMessage()
 <<
                      std::endl;
            result = false;
            break;
```

```
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para C++ da API.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Publicar uma mensagem com um atributo.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                        "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
```

```
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para C++ .

`SetSMSAttributes`

O código de exemplo a seguir mostra como usar `SetSMSAttributes`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Como usar o Amazon SNS para definir o atributo DefaultSMSType.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para receber detalhes da API, consulte SetSMSAttributes na Referência da API AWS SDK para C++ .

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Inscrever uma aplicação móvel em um tópico.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
```

```
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Inscrever uma função do Lambda em um tópico.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Assinar uma fila do SQS em um tópico.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
```

```
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN << "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

Assinar com um filtro em um tópico.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
```

```
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then only
 the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
 filter "
                                << "will be added to this subscription." <<
 std::endl;
                    }
                }
```

```
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN << "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
```

```
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para C++ .

Cancelar assinatura

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK para C++ .

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
```

```
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an SMS
 text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account is
 in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction that
 you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
```

```
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para C++ .

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
```

```
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the queues."
              << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication and
message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or automatically
generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic, any
message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
```

```
                << "within the five-minute deduplication interval, is accepted but
 not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a deduplication
 ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);
```

```
        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " << std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

  request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
```

```
                                        "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue, '.fifo'
must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is "
```

```
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
```

```
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
```

```
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then only
 the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
```

```
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN << "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }
```

```
    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received in
the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
```

```
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
```

```
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
```

```
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }
```

```
    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
```

```
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

```
//! Create an IAM policy that gives an SQS queue permission to receive messages from
 an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Publicar
- ReceiveMessage
- SetQueueAttributes
- Assinar
- Cancelar assinatura

Exemplos do Amazon SQS usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon SQS

O exemplo de código a seguir mostra como começar a usar o Amazon SQS.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para o arquivo CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sqs)

# Set this project's name.
project("hello_sqs")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if(WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.
```

```
    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif()

add_executable(${PROJECT_NAME}
        hello_sqs.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_sqs.cpp.

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>

/*
 *  A "Hello SQS" starter application that initializes an Amazon Simple Queue
 Service
 *  (Amazon SQS) client and lists the SQS queues in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sqs'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SQS::SQSClient sqsClient(clientConfig);

        Aws::Vector<Aws::String> allQueueUrls;
```

```
        Aws::String nextToken; // Next token is used to handle a paginated response.
        do {
            Aws::SQS::Model::ListQueuesRequest request;

            Aws::SQS::Model::ListQueuesOutcome outcome =
 sqsClient.ListQueues(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::String> &pageOfQueueUrls =
 outcome.GetResult().GetQueueUrls();
                if (!pageOfQueueUrls.empty()) {
                    allQueueUrls.insert(allQueueUrls.cend(),
 pageOfQueueUrls.cbegin(),
                                        pageOfQueueUrls.cend());
                }
            }
            else {
                std::cerr << "Error with SQS::ListQueues. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());


        std::cout << "Hello Amazon SQS! You have " << allQueueUrls.size() << "
 queue"
                  << (allQueueUrls.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allQueueUrls.empty()) {
            std::cout << "Here are your queue URLs." << std::endl;
            for (const Aws::String &queueUrl: allQueueUrls) {
                std::cout << "  * " << queueUrl << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para C++ da API.

Ações

`ChangeMessageVisibility`

O código de exemplo a seguir mostra como usar `ChangeMessageVisibility`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Changes the visibility timeout of a message in an Amazon Simple Queue Service
//! (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageReceiptHandle: A message receipt handle.
  \param visibilityTimeoutSeconds: Visibility timeout in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::changeMessageVisibility(
        const Aws::String &queue_url,
        const Aws::String &messageReceiptHandle,
        int visibilityTimeoutSeconds,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ChangeMessageVisibilityRequest request;
    request.SetQueueUrl(queue_url);
    request.SetReceiptHandle(messageReceiptHandle);
    request.SetVisibilityTimeout(visibilityTimeoutSeconds);
```

```
    auto outcome = sqsClient.ChangeMessageVisibility(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully changed visibility of message " <<
                  messageReceiptHandle << " from queue " << queue_url << std::endl;
    }
    else {
        std::cout << "Error changing visibility of message from queue "
                  << queue_url << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ChangeMessageVisibility em Referência AWS SDK para C++ da API.

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Create an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueName: An Amazon SQS queue name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::createQueue(const Aws::String &queueName,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::CreateQueueRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::CreateQueueOutcome outcome =
 sqsClient.CreateQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created queue " << queueName << " with a queue
 URL "
                  << outcome.GetResult().GetQueueUrl() << "." << std::endl;
    }
    else {
        std::cerr << "Error creating queue " << queueName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateQueue em Referência AWS SDK para C++ da API.

`DeleteMessage`

O código de exemplo a seguir mostra como usar `DeleteMessage`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
//! Delete a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageReceiptHandle: A message receipt handle.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteMessage(const Aws::String &queueUrl,
                                const Aws::String &messageReceiptHandle,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetReceiptHandle(messageReceiptHandle);

    const Aws::SQS::Model::DeleteMessageOutcome outcome = sqsClient.DeleteMessage(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted message from queue " << queueUrl
                  << std::endl;
    }
    else {
        std::cerr << "Error deleting message from queue " << queueUrl << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteMessage em Referência AWS SDK para C++ da API.

`DeleteMessageBatch`

O código de exemplo a seguir mostra como usar `DeleteMessageBatch`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::DeleteMessageBatchRequest request;
        request.SetQueueUrl(queueURLS[i]);
        int id = 1; // Ids must be unique within a batch delete request.
        for (const Aws::String &receiptHandle: receiptHandles) {
            Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
            entry.SetId(std::to_string(id));
            ++id;
            entry.SetReceiptHandle(receiptHandle);
            request.AddEntries(entry);
        }

        Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                sqsClient.DeleteMessageBatch(request);

        if (outcome.IsSuccess()) {
            std::cout << "The batch deletion of messages was successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with SQS::DeleteMessageBatch. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);
```

```
                return false;
            }
```

- Para obter detalhes da API, consulte DeleteMessageBatch em Referência AWS SDK para C++ da API.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Delete an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueURL: An Amazon SQS queue URL.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteQueue(const Aws::String &queueURL,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::SQS::Model::DeleteQueueRequest request;
    request.SetQueueUrl(queueURL);

    const Aws::SQS::Model::DeleteQueueOutcome outcome =
 sqsClient.DeleteQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted queue with url " << queueURL <<
                  std::endl;
```

```
    }
    else {
        std::cerr << "Error deleting queue " << queueURL << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteQueue em Referência AWS SDK para C++ da API.

`GetQueueAttributes`

O código de exemplo a seguir mostra como usar `GetQueueAttributes`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

 request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
 &attributes =
```

```
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }

            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;


            }
```

- Para obter detalhes da API, consulte GetQueueAttributes em Referência AWS SDK para C++ da API.

`GetQueueUrl`

O código de exemplo a seguir mostra como usar `GetQueueUrl`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Get the URL for an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
```

```
  \param queueName: An Amazon SQS queue name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::getQueueUrl(const Aws::String &queueName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::GetQueueUrlRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::GetQueueUrlOutcome outcome =
 sqsClient.GetQueueUrl(request);
    if (outcome.IsSuccess()) {
        std::cout << "Queue " << queueName << " has url " <<
                  outcome.GetResult().GetQueueUrl() << std::endl;
    }
    else {
        std::cerr << "Error getting url for queue " << queueName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte GetQueueUrl em Referência AWS SDK para C++ da API.

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! List the Amazon Simple Queue Service (Amazon SQS) queues within an AWS account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SQS::listQueues(const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ListQueuesRequest listQueuesRequest;

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::String> allQueueUrls;

    do {
        if (!nextToken.empty()) {
            listQueuesRequest.SetNextToken(nextToken);
        }
        const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
                listQueuesRequest);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &queueUrls =
 outcome.GetResult().GetQueueUrls();
            allQueueUrls.insert(allQueueUrls.end(),
                                queueUrls.begin(),
                                queueUrls.end());

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing queues: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." << std::endl;
```

```
    for (const auto &iter: allQueueUrls) {
        std::cout << " " << iter << std::endl;
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para C++ da API.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Receive a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::receiveMessage(const Aws::String &queueUrl,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ReceiveMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMaxNumberOfMessages(1);
```

```
    const Aws::SQS::Model::ReceiveMessageOutcome outcome = sqsClient.ReceiveMessage(
            request);
    if (outcome.IsSuccess()) {

        const Aws::Vector<Aws::SQS::Model::Message> &messages =
                outcome.GetResult().GetMessages();
        if (!messages.empty()) {
            const Aws::SQS::Model::Message &message = messages[0];
            std::cout << "Received message:" << std::endl;
            std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
            std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
 std::endl;
            std::cout << "  Body: " << message.GetBody() << std::endl << std::endl;
        }
        else {
            std::cout << "No messages received from queue " << queueUrl <<
                      std::endl;

        }
    }
    else {
        std::cerr << "Error receiving message from queue " << queueUrl << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte ReceiveMessage em Referência AWS SDK para C++ da API.

SendMessage

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Send a message to an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageBody: A message body.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::sendMessage(const Aws::String &queueUrl,
                              const Aws::String &messageBody,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SendMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMessageBody(messageBody);

    const Aws::SQS::Model::SendMessageOutcome outcome =
 sqsClient.SendMessage(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message to " << queueUrl <<
                  std::endl;
    }
    else {
        std::cerr << "Error sending message to " << queueUrl << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte SendMessage em Referência AWS SDK para C++ da API.

`SetQueueAttributes`

O código de exemplo a seguir mostra como usar `SetQueueAttributes`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Set the value for an attribute in an Amazon Simple Queue Service (Amazon SQS)
 queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param attributeName: An attribute name enum.
  \param attribute: The attribute value as a string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setQueueAttributes(const Aws::String &queueURL,
                                     Aws::SQS::Model::QueueAttributeName
 attributeName,
                                     const Aws::String &attribute,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
            attributeName,
            attribute);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
 sqsClient.SetQueueAttributes(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set the attribute  " <<

 Aws::SQS::Model::QueueAttributeNameMapper::GetNameForQueueAttributeName(
```

```
                            attributeName)
                    << " with value " << attribute << " in queue " <<
                    queueURL << "." << std::endl;
    }
    else {
        std::cout << "Error setting attribute for  queue " <<
                    queueURL << ": " << outcome.GetError().GetMessage() <<
                    std::endl;
    }

    return outcome.IsSuccess();
}
```

Configurar uma dead-letter queue.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Connect an Amazon Simple Queue Service (Amazon SQS) queue to an associated
//! dead-letter queue.
/*!
  \param srcQueueUrl: An Amazon SQS queue URL.
  \param deadLetterQueueARN: The Amazon Resource Name (ARN) of an Amazon SQS dead-
letter queue.
  \param maxReceiveCount: The max receive count of a message before it is sent to
 the dead-letter queue.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setDeadLetterQueue(const Aws::String &srcQueueUrl,
                                     const Aws::String &deadLetterQueueARN,
                                     int maxReceiveCount,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String redrivePolicy = MakeRedrivePolicy(deadLetterQueueARN,
 maxReceiveCount);

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(srcQueueUrl);
```

```
    request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
            redrivePolicy);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set dead letter queue for queue  " <<
                  srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
    }
    else {
        std::cerr << "Error setting dead letter queue for queue " <<
                  srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }

    return outcome.IsSuccess();
}

//! Make a redrive policy for a dead-letter queue.
/*!
  \param queueArn: An Amazon SQS ARN for the dead-letter queue.
  \param maxReceiveCount: The max receive count of a message before it is sent to
 the dead-letter queue.
  \return Aws::String: Policy as JSON string.
 */
Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queueArn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(maxReceiveCount);

    Aws::Utils::Json::JsonValue policy_map;
    policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
    policy_map.WithObject("maxReceiveCount", max_msg_entry);

    return policy_map.View().WriteReadable();
}
```

Configurar uma fila do Amazon SQS para usar sondagem longa.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Set the wait time for an Amazon Simple Queue Service (Amazon SQS) queue poll.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param pollTimeSeconds: The receive message wait time in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setQueueLongPollingAttribute(const Aws::String &queueURL,
                                               const Aws::String &pollTimeSeconds,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
            pollTimeSeconds);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
 sqsClient.SetQueueAttributes(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated long polling time for queue " <<
                  queueURL << " to " << pollTimeSeconds << std::endl;
    }
    else {
        std::cout << "Error updating long polling time for queue " <<
                  queueURL << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte SetQueueAttributes em Referência AWS SDK para C++ da API.

Cenários

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the queues."
```

```
              << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication and
 message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or automatically
 generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic, any
 message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted but
 not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a deduplication
 ID? (y/n) ");
    }
```

```
    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " << std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue, '.fifo'
must "
                            << "be appended to the queue name." << std::endl;
                }
            }
```

```
            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is "
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
```

```
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
```

```
        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
```

```
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then only
 the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
 filter "
                                << "will be added to this subscription." <<
 std::endl;
                    }
                }
```

```
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN << "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
```

```
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received in
the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
```

```
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
```

```
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
```

```
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
```

```
                                        const Aws::Vector<Aws::String>
&subscriptionARNS,
                                        const Aws::SNS::SNSClient &snsClient,
                                        const Aws::SQS::SQSClient &sqsClient,
                                        bool askUser) {
   bool result = true;
   printAsterisksLine();
   if (!queueURLS.empty() && askUser &&
       askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

       for (const auto &queueURL: queueURLS) {
           // 9.  Delete an SQS queue.
           Aws::SQS::Model::DeleteQueueRequest request;
           request.SetQueueUrl(queueURL);

           Aws::SQS::Model::DeleteQueueOutcome outcome =
                   sqsClient.DeleteQueue(request);

           if (outcome.IsSuccess()) {
               std::cout << "The queue with URL '" << queueURL
                         << "' was successfully deleted." << std::endl;
           }
           else {
               std::cerr << "Error with SQS::DeleteQueue. "
                         << outcome.GetError().GetMessage()
                         << std::endl;
               result = false;
           }
       }

       for (const auto &subscriptionARN: subscriptionARNS) {
           // 10. Unsubscribe an SNS subscription.
           Aws::SNS::Model::UnsubscribeRequest request;
           request.SetSubscriptionArn(subscriptionARN);

           Aws::SNS::Model::UnsubscribeOutcome outcome =
                   snsClient.Unsubscribe(request);

           if (outcome.IsSuccess()) {
               std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                         << "' was successful." << std::endl;
           }
           else {
               std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
```

```
                            << outcome.GetError().GetMessage()
                            << std::endl;
                    result = false;
                }
            }
        }

        printAsterisksLine();
        if (!topicARN.empty() && askUser &&
            askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

            // 11. Delete an SNS topic.
            Aws::SNS::Model::DeleteTopicRequest request;
            request.SetTopicArn(topicARN);

            Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

            if (outcome.IsSuccess()) {
                std::cout << "The topic with ARN '" << topicARN
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        return result;
    }

    //! Create an IAM policy that gives an SQS queue permission to receive messages from
 an SNS topic.
    /*!
     \sa createPolicyForQueue()
     \param queueARN: The SQS queue Amazon Resource Name (ARN).
     \param topicARN: The SNS topic ARN.
     \return Aws::String: The policy as JSON.
     */
    Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
```

```
                                                                const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++ .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

AWS STS exemplos de uso do SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ with AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);
```

```
    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AssumeRole em Referência AWS SDK para C++ da API.

Exemplos do Amazon Transcribe Streaming usando o SDK para C++

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para C++ Amazon Transcribe Streaming.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`StartStreamTranscription`

O código de exemplo a seguir mostra como usar `StartStreamTranscription`.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
 SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;
```

```
        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
                [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
&error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
        //SetTranscriptEventCallback called for every 'chunk' of file transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
```

```
                                << std::endl;

                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
 characters in the given string. It returns
                    //the number of characters extracted by the last read()
 operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
 std::endl;
                            break;
                        }
                    }
                    else {
                        break;
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
                    // Per the spec, we have to send an empty event (an event
 without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" << std::endl;
                }
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
```

```
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };

        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- Para obter detalhes da API, consulte StartStreamTranscription em Referência AWS SDK para C++ da API.

Cenários

Transcrever um arquivo de áudio

O exemplo de código a seguir mostra como gerar uma transcrição de um arquivo de áudio de origem usando o streaming do Amazon Transcribe.

SDK para C++

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
 SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
                []( const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
 &error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
        //SetTranscriptEventCallback called for every 'chunk' of file transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
```

```
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
                                  << std::endl;

                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
```

```
                    //the number of characters extracted by the last read()
 operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
 std::endl;
                            break;
                        }
                    }
                    else {
                        break;
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
                    // Per the spec, we have to send an empty event (an event
 without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" << std::endl;
                }
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };
```

```
        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- Para obter detalhes da API, consulte StartStreamTranscription em Referência AWS SDK para C++ da API.

Exemplos de código para SDK para Go V2

Os exemplos de código a seguir mostram como usar a AWS SDK para Go V2 com AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK for Go V2 — Saiba mais sobre como usar o Go with. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.

- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos do API Gateway usando o SDK para o Go V2
- Exemplos do Aurora usando o SDK para Go V2
- Exemplos do Amazon Bedrock usando o SDK para Go V2
- Exemplos do Amazon Bedrock Runtime usando o SDK para Go V2
- CloudFormation exemplos de uso do SDK para Go V2
- CloudWatch Exemplos de registros usando o SDK para Go V2
- Exemplos do Provedor de Identidade do Amazon Cognito usando o SDK para Go V2
- Exemplos do Amazon DocumentDB usando o SDK para Go V2
- Exemplos do DynamoDB usando o SDK para Go V2
- Exemplos do IAM usando o SDK para Go V2
- Exemplos do Kinesis usando o SDK para Go V2
- Exemplos do Lambda usando o SDK para Go V2
- Exemplos do Amazon MSK usando o SDK para Go V2
- Exemplos da Central de Parceiros usando o SDK para Go V2
- Exemplos do Amazon RDS usando o SDK para Go V2
- Exemplos do Amazon Redshift usando o SDK para Go V2
- Exemplos do Amazon S3 usando o SDK para Go V2
- Exemplos do Amazon SNS usando o SDK para Go V2
- Exemplos do Amazon SQS usando o SDK para Go V2

Exemplos do API Gateway usando o SDK para o Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o API Gateway.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- AWS contribuições da comunidade

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Go V2

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Go.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos do Aurora usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Aurora.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Aurora

O exemplo de código a seguir mostra como começar a usar o Aurora.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up to
 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
```

```
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Go .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de cluster do banco de dados do Aurora e definir os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.

- Crie um snapshot do cluster do banco de dados e limpe os recursos.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/google/uuid"
)

// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
```

```
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service (Amazon
 RDS)
// client and create wrappers for the actions used in the scenario.
func NewGetStartedClusters(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine, parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
```

```
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a specified
// database engine and create a DB cluster parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?\n",
 families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
```

```
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName, "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName, updateParams)
 if err != nil {
```

```
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source of
 'user'.")
 userParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a database
// of a specified type. The database is also configured to use a custom DB cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

 log.Println("Checking for an existing DB cluster.")
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
```

```
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?\n",
 engineChoices)
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName, dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
   log.Println("Cluster created and available.")
  }
 }
 log.Println("Cluster data:")
 log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
 log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
 log.Printf("\tStatus: %v\n", *cluster.Status)
 log.Printf("\tEngine: %v\n", *cluster.Engine)
 log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
 log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
 log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
 log.Println(strings.Repeat("-", 88))
 return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
 log.Println("Checking for an existing database instance.")
```

```
dbInstance, err := scenario.dbClusters.GetInstance(ctx,
*cluster.DBClusterIdentifier)
if err != nil {
 panic(err)
}
if dbInstance == nil {
 log.Println("Let's create a database instance in your DB cluster.")
 log.Println("First, choose a DB instance type:")
 instOpts, err := scenario.dbClusters.GetOrderableInstances(
  ctx, *cluster.Engine, *cluster.EngineVersion)
 if err != nil {
  panic(err)
 }
 var instChoices []string
 for _, opt := range instOpts {
  instChoices = append(instChoices, *opt.DBInstanceClass)
 }
 slices.Sort(instChoices)
 instChoices = slices.Compact(instChoices)
 instIndex := scenario.questioner.AskChoice(
  "Which DB instance class do you want to use?\n", instChoices)
 log.Println("Creating a database instance. This typically takes several minutes.")
 dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
  ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier, *cluster.Engine,
  instChoices[instIndex])
 if err != nil {
  panic(err)
 }
 for *dbInstance.DBInstanceStatus != "available" {
  scenario.helper.Pause(30)
  dbInstance, err = scenario.dbClusters.GetInstance(ctx,
*cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
 }
}
log.Println("Instance data:")
log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
log.Printf("\tEngine: %v\n", *dbInstance.Engine)
log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
log.Println(strings.Repeat("-", 88))
```

```
 return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster and
 tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
 log.Println(
  "You can now connect to your database using your favorite MySql client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n" +
   "that is running in the same VPC as your database cluster. Pass the endpoint,\n"
 +
   "port, and administrator user name to 'mysql' and enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
 log.Println("For more information, see the User Guide for Aurora:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora.Co
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context, clusterName
 string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.\n",
 snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
```

```
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster parameter
 group.
// Before the DB cluster parameter group can be deleted, all associated DB instances
 and
// DB clusters must first be deleted.
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
  log.Printf("Deleting database instance %v.\n", *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx, *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
```

```
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Aurora.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}


// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
```

```
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
```

```
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context, parameterGroupName
 string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
```

```
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified engine
 and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName string,
 parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
```

```
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName string)
 error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context, clusterName
 string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
```

```
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context, clusterName
 string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string) (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
```

```
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName string)
 error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
```

```
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateDBCluster

- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBInstance
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified engine
 and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName string,
 parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Go .

`CreateDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
```

```
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

- Para obter detalhes da API, consulte CreateDBClusterParameterGroup em Referência AWS SDK para Go da API.

`CreateDBClusterSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBClusterSnapshot`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context, clusterName
 string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}
```

- Para obter detalhes da API, consulte CreateDBClusterSnapshot em Referência AWS SDK para Go da API.

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context, clusterName
 string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Go .

`DeleteDBCluster`

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName string)
 error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Go .

`DeleteDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBClusterParameterGroup`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
```

```
  return nil
 }
}
```

- Para obter detalhes da API, consulte DeleteDBClusterParameterGroup em Referência AWS SDK para Go da API.

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName string)
 error {
```

```
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Go .

`DescribeDBClusterParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameterGroups`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
```

```
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- Para obter detalhes da API, consulte DescribeDBClusterParameterGroups em Referência AWS SDK para Go da API.

`DescribeDBClusterParameters`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameters`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Para obter detalhes da API, consulte DescribeDBClusterParameters em Referência AWS SDK para Go da API.

`DescribeDBClusterSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBClusterSnapshots`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
```

```
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- Para obter detalhes da API, consulte DescribeDBClusterSnapshots em Referência AWS SDK para Go da API.

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Go .

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
```

```
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions em Referência AWS SDK para Go da API.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string) (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
```

```
  return &output.DBInstances[0], nil
 }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para Go .

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
```

```
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptions em Referência AWS SDK para Go da API.

`ModifyDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context, parameterGroupName
 string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obter detalhes da API, consulte ModifyDBClusterParameterGroup em Referência AWS SDK para Go da API.

Exemplos do Amazon Bedrock usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Bedrock

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/bedrock"
)

const region = "us-east-1"

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock client and
// list the available foundation models in your account and the chosen region.
```

```
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(region))
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 bedrockClient := bedrock.NewFromConfig(sdkConfig)
 result, err := bedrockClient.ListFoundationModels(ctx,
 &bedrock.ListFoundationModelsInput{})
 if err != nil {
  fmt.Printf("Couldn't list foundation models. Here's why: %v\n", err)
  return
 }
 if len(result.ModelSummaries) == 0 {
  fmt.Println("There are no foundation models.")
 }
 for _, modelSummary := range result.ModelSummaries {
  fmt.Println(*modelSummary.ModelId)
 }
}
```

- Para obter detalhes da API, consulte ListFoundationModels em Referência AWS SDK para Go da API.

Ações

`ListFoundationModels`

O código de exemplo a seguir mostra como usar `ListFoundationModels`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Bedrock disponíveis.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/bedrock"
 "github.com/aws/aws-sdk-go-v2/service/bedrock/types"
)

// FoundationModelWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock service client that is used to perform foundation model
 actions.
type FoundationModelWrapper struct {
 BedrockClient *bedrock.Client
}



// ListPolicies lists Bedrock foundation models that you can use.
func (wrapper FoundationModelWrapper) ListFoundationModels(ctx context.Context)
 ([]types.FoundationModelSummary, error) {

 var models []types.FoundationModelSummary

 result, err := wrapper.BedrockClient.ListFoundationModels(ctx,
 &bedrock.ListFoundationModelsInput{})

 if err != nil {
  log.Printf("Couldn't list foundation models. Here's why: %v\n", err)
 } else {
  models = result.ModelSummaries
 }
 return models, err
```

```
}
```

- Para obter detalhes da API, consulte ListFoundationModels em Referência AWS SDK para Go da API.

Exemplos do Amazon Bedrock Runtime usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon Bedrock Runtime.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Amazônia Nova
- Gerador de Imagens do Amazon Titan
- Claude da Anthropic

Conceitos básicos

Olá, Amazon Bedrock Runtime

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock Runtime.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main
```

```
import (
 "context"
 "flag"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock Runtime client
// and invokes Anthropic Claude using the Converse API.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {

 region := flag.String("region", "us-east-1", "The AWS region")
 flag.Parse()

 fmt.Printf("Using AWS region: %s\n", *region)

 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(*region))
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }

 client := bedrockruntime.NewFromConfig(sdkConfig)

 // Set the model ID, e.g., Claude Haiku.
 // The "global." prefix enables cross-region inference, allowing the request
 // to be routed to the nearest available region for the specified model.
 modelId := "global.anthropic.claude-haiku-4-5-20251001-v1:0"

 // Start a conversation with the user message.
 prompt := "Hello. In a short paragraph, explain what you can do."

 message := types.Message{
  Content: []types.ContentBlock{
```

```
    &types.ContentBlockMemberText{Value: prompt},
   },
   Role: types.ConversationRoleUser,
  }

  fmt.Printf("Model: %s\n", modelId)
  fmt.Printf("Prompt: %s\n\n", prompt)

  result, err := client.Converse(ctx, &bedrockruntime.ConverseInput{
   ModelId:  aws.String(modelId),
   Messages: []types.Message{message},
  })

  if err != nil {
   log.Fatalf("ERROR: Can't invoke '%s'. Reason: %v\n", modelId, err)
  }

  // Extract and print the response text.
  responseMessage, ok := result.Output.(*types.ConverseOutputMemberMessage)
  if !ok {
   log.Fatal("ERROR: Unexpected output type from Converse API")
  }
  responseContentBlock := responseMessage.Value.Content[0]
  text, ok := responseContentBlock.(*types.ContentBlockMemberText)
  if !ok {
   log.Fatal("ERROR: Unexpected content block type from Converse API")
  }
  fmt.Printf("Response: %s\n", text.Value)
 }
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Go .

Amazônia Nova

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
func (wrapper ConverseWrapper) ConverseNova(ctx context.Context, prompt string)
 (string, error) {
 var content = types.ContentBlockMemberText{
  Value: prompt,
 }
 var message = types.Message{
  Content: []types.ContentBlock{&content},
  Role:    "user",
 }
 modelId := "amazon.nova-lite-v1:0"
 var converseInput = bedrockruntime.ConverseInput{
  ModelId:  aws.String(modelId),
  Messages: []types.Message{message},
 }
 response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
 if err != nil {
  ProcessError(err, modelId)
 }

 responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
 responseContentBlock := responseText.Value.Content[0]
 text, _ := responseContentBlock.(*types.ContentBlockMemberText)
 return text.Value, nil
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Go .

Gerador de Imagens do Amazon Titan

InvokeModel

O exemplo de código a seguir mostra como invocar o Amazon Titan Image no Amazon Bedrock para gerar uma imagem.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Gerador de Imagens do Amazon Titan.

```
import (
 "context"
 "encoding/json"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
)

// InvokeModelWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke foundation models.
type InvokeModelWrapper struct {
 BedrockRuntimeClient *bedrockruntime.Client
}



type TitanImageRequest struct {
 TaskType              string                `json:"taskType"`
 TextToImageParams     TextToImageParams     `json:"textToImageParams"`
 ImageGenerationConfig ImageGenerationConfig `json:"imageGenerationConfig"`
}
type TextToImageParams struct {
 Text string `json:"text"`
```

```
}
type ImageGenerationConfig struct {
 NumberOfImages int     `json:"numberOfImages"`
 Quality        string  `json:"quality"`
 CfgScale       float64 `json:"cfgScale"`
 Height         int     `json:"height"`
 Width          int     `json:"width"`
 Seed           int64   `json:"seed"`
}

type TitanImageResponse struct {
 Images []string `json:"images"`
}

// Invokes the Titan Image model to create an image using the input provided
// in the request body.
func (wrapper InvokeModelWrapper) InvokeTitanImage(ctx context.Context, prompt
 string, seed int64) (string, error) {
 modelId := "amazon.titan-image-generator-v2:0"

 body, err := json.Marshal(TitanImageRequest{
  TaskType: "TEXT_IMAGE",
  TextToImageParams: TextToImageParams{
   Text: prompt,
  },
  ImageGenerationConfig: ImageGenerationConfig{
   NumberOfImages: 1,
   Quality:        "standard",
   CfgScale:       8.0,
   Height:         512,
   Width:          512,
   Seed:           seed,
  },
 })

 if err != nil {
  log.Fatal("failed to marshal", err)
 }

 output, err := wrapper.BedrockRuntimeClient.InvokeModel(ctx,
 &bedrockruntime.InvokeModelInput{
  ModelId:     aws.String(modelId),
  ContentType: aws.String("application/json"),
  Body:        body,
```

```
 })

 if err != nil {
  ProcessError(err, modelId)
 }

 var response TitanImageResponse
 if err := json.Unmarshal(output.Body, &response); err != nil {
  log.Fatal("failed to unmarshal", err)
 }

 base64ImageData := response.Images[0]

 return base64ImageData, nil

}
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK para Go da API.

Claude da Anthropic

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
import (
 "context"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// ConverseWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke Bedrock.
type ConverseWrapper struct {
 BedrockRuntimeClient *bedrockruntime.Client
}



func (wrapper ConverseWrapper) ConverseClaude(ctx context.Context, prompt string)
 (string, error) {
 var content = types.ContentBlockMemberText{
  Value: prompt,
 }
 var message = types.Message{
  Content: []types.ContentBlock{&content},
  Role:    "user",
 }
 modelId := "anthropic.claude-3-haiku-20240307-v1:0"
 var converseInput = bedrockruntime.ConverseInput{
  ModelId:  aws.String(modelId),
  Messages: []types.Message{message},
 }
 response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
 if err != nil {
  ProcessError(err, modelId)
 }

 responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
 responseContentBlock := responseText.Value.Content[0]
 text, _ := responseContentBlock.(*types.ContentBlockMemberText)
 return text.Value, nil

}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Go .

CloudFormation exemplos de uso do SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com CloudFormation.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DescribeStacks`

O código de exemplo a seguir mostra como usar `DescribeStacks`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string
```

```
type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

- Para obter detalhes da API, consulte DescribeStacks em Referência AWS SDK para Go da API.

CloudWatch Exemplos de registros usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com CloudWatch registros.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

StartLiveTail

O código de exemplo a seguir mostra como usar `StartLiveTail`.

SDK para Go V2

Inclua os arquivos necessários.

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

Gerencie os eventos da sessão do Live Tail.

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
 eventsChan := stream.Events()
 for {
  event := <-eventsChan
  switch e := event.(type) {
  case *types.StartLiveTailResponseStreamMemberSessionStart:
   log.Println("Received SessionStart event")
  case *types.StartLiveTailResponseStreamMemberSessionUpdate:
   for _, logEvent := range e.Value.SessionResults {
    log.Println(*logEvent.Message)
   }
  default:
   // Handle on-stream exceptions
   if err := stream.Err(); err != nil {
    log.Fatalf("Error occured during streaming: %v", err)
   } else if event == nil {
    log.Println("Stream is Closed")
```

```
    return
   } else {
    log.Fatalf("Unknown event type: %T", e)
   }
  }
 }
}
```

Inicie a sessão do Live Tail.

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
 panic("configuration error, " + err.Error())
}
client := cloudwatchlogs.NewFromConfig(cfg)

request := &cloudwatchlogs.StartLiveTailInput{
 LogGroupIdentifiers:   logGroupIdentifiers,
 LogStreamNames:        logStreamNames,
 LogEventFilterPattern: logEventFilterPattern,
}

response, err := client.StartLiveTail(context.TODO(), request)
// Handle pre-stream Exceptions
if err != nil {
 log.Fatalf("Failed to start streaming: %v", err)
}

// Start a Goroutine to handle events over stream
stream := response.GetStream()
go handleEventStreamAsync(stream)
```

Interrompa a sessão do Live Tail após um período decorrido.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- Para obter detalhes da API, consulte StartLiveTail em Referência AWS SDK para Go da API.

Exemplos do Provedor de Identidade do Amazon Cognito usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon Cognito

O exemplo de código a seguir mostra como começar a usar o Amazon Cognito.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
```

```
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
```

```
}
```

- Para obter detalhes da API, consulte ListUserPools em Referência AWS SDK para Go da API.

Ações

`AdminCreateUser`

O código de exemplo a seguir mostra como usar `AdminCreateUser`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
```

```
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- Para obter detalhes da API, consulte AdminCreateUser em Referência AWS SDK para Go da API.

`AdminSetUserPassword`

O código de exemplo a seguir mostra como usar `AdminSetUserPassword`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- Para obter detalhes da API, consulte AdminSetUserPassword em Referência AWS SDK para Go da API.

`ConfirmForgotPassword`

O código de exemplo a seguir mostra como usar `ConfirmForgotPassword`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
```

```
    log.Println(*invalidPassword.Message)
   } else {
    log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
   }
  }
  return err
}
```

- Para obter detalhes da API, consulte ConfirmForgotPassword em Referência AWS SDK para Go da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para Go da API.

`ForgotPassword`

O código de exemplo a seguir mostra como usar `ForgotPassword`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
```

```
 CognitoClient *cognitoidentityprovider.Client
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

- Para obter detalhes da API, consulte ForgotPassword em Referência AWS SDK para Go da API.

`InitiateAuth`

O código de exemplo a seguir mostra como usar `InitiateAuth`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
```

```
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

- Para obter detalhes da API, consulte InitiateAuth em Referência AWS SDK para Go da API.

`ListUserPools`

O código de exemplo a seguir mostra como usar `ListUserPools`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
```

```
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- Para obter detalhes da API, consulte ListUserPools em Referência AWS SDK para Go da API.

`SignUp`

O código de exemplo a seguir mostra como usar `SignUp`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

- Para obter detalhes da API, consulte SignUp em Referência AWS SDK para Go da API.

`UpdateUserPool`

O código de exemplo a seguir mostra como usar `UpdateUserPool`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
```

```
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte UpdateUserPool em Referência AWS SDK para Go da API.

Cenários

Confirme automaticamente usuários conhecidos com uma função do Lambda

O exemplo de código a seguir mostra como confirmar automaticamente usuários conhecidas do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PreSignUp`.
- Inscreva-se para ser um usuário no Amazon Cognito.
- A função do Lambda verifica uma tabela do DynamoDB e confirma automaticamente os usuários conhecidos.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
```

```
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you specify.
```

```
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's email
 matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least eight
 characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
```

```
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string, userName
 string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))
```

```
 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `PreSignUp` com uma função do Lambda.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}
```

```
type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the response
 from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
```

```
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match supplied
 UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.\n",
 user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)
```

```
// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}
```

```
// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
```

```
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
```

```
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
```

```
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
```

```
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
```

```
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
```

```
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
```

```
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
```

```
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
```

```
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
```

```
    }
    log.Println("Deleted user.")
   }
   triggerList := make([]actions.TriggerInfo, len(resources.triggers))
   for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
  nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Migre automaticamente usuários conhecidos com uma função do Lambda

O exemplo de código a seguir mostra como migrar automaticamente usuários conhecidas do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `MigrateUser`.
- Faça login no Amazon Cognito com um nome de usuário e e-mail que não estejam no grupo de usuários.

- A função do Lambda verifica uma tabela do DynamoDB e migra automaticamente os usuários conhecidos para o grupo de usuários.
- Realize um fluxo de senha esquecida para redefinir a senha para o usuário migrado.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
```

```
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function take
 action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username and
 email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user is
 migrated to the Cognito user pool.")
```

```
 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This email
 will be used to confirm user migration\n" +
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.", user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
```

```
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string, user
 actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user to
 Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
  log.Println("To complete this example and successfully migrate a user to Cognito,
 you must enter an email\n" +
   "you own that can receive a confirmation code.")
  return
 }
 codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
 user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("\nA confirmation code has been sent to %v.", *codeDelivery.Destination)
 code := runner.questioner.Ask("Check your email and enter it here:")

 confirmed := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !confirmed {
  log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
 user.UserName, password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   confirmed = true
  }
 }
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
 password)
 if err != nil {
  panic(err)
```

```
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
```

```
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `MigrateUser` com uma função do Lambda.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser) (events.CognitoEventUserPoolsMigrateUser,
 error) {
 log.Printf("Received migrate trigger from %v for user '%v'", event.TriggerSource,
 event.UserName)
```

```
if event.TriggerSource != "UserMigration_Authentication" {
 return event, nil
}
tableName := os.Getenv(TABLE_NAME)
user := UserInfo{
 UserName: event.UserName,
}
log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
if err != nil {
 log.Printf("Error building expression to query for user '%v'.\n", user.UserName)
 return event, err
}
output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
 TableName:                 aws.String(tableName),
 FilterExpression:          expr.Filter(),
 ExpressionAttributeNames:  expr.Names(),
 ExpressionAttributeValues: expr.Values(),
})
if err != nil {
 log.Printf("Error looking up user '%v'.\n", user.UserName)
 return event, err
}
if len(output.Items) == 0 {
 log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
 return event, err
}

var users []UserInfo
err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
if err != nil {
 log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
 return event, err
}

user = users[0]
log.Printf("UserName '%v' found with email %v. User is migrated and must reset
password.\n", user.UserName, user.UserEmail)
event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes = map[string]string{
 "email":          user.UserEmail,
 "email_verified": "true", // email_verified is required for the forgot password
flow.
}
```

```
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus = "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
```

```
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}
```

```
// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
```

```
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
```

```
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

```
// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
```

```
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
```

```
    log.Println(*invalidPassword.Message)
   } else {
    log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
   }
  }
  return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
```

```
    log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
   }
  }
  return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
```

```
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
```

```
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
```

```
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
```

```
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)
```

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
```

```
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
```

```
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Grave dados de atividades personalizados com uma função do Lambda após a autenticação do usuário do Amazon Cognito

O exemplo de código a seguir mostra como gravar dados de atividade personalizados com uma função do Lambda depois da autenticação do usuário do Amazon Cognito.

- Use as funções de administrador para adicionar um usuário a um grupo de usuários.
- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PostAuthentication`.
- Faça login do novo usuário no Amazon Cognito.
- A função Lambda grava informações personalizadas em CloudWatch Logs e em uma tabela do DynamoDB.
- Obtenha e veja dados personalizados da tabela do DynamoDB e, em seguida, limpe os recursos.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
```

```
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))
```

```
 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication trigger
 from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function take
 action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle PostAuthentication
 Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string, userName
 string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
```

```
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the DynamoDB
 table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])
```

```
 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `PostAuthentication` com uma função do Lambda.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
```

```
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to the
 logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)
```

```
 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
```

```
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}
```

```
// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
```

```
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
```

```
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
```

```
    lambdaConfig.PostAuthentication = trigger.HandlerArn
   }
  }
  _, err = actor.CognitoClient.UpdateUserPool(ctx,
  &cognitoidentityprovider.UpdateUserPoolInput{
   UserPoolId:   aws.String(userPoolId),
   LambdaConfig: lambdaConfig,
  })
  if err != nil {
   log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
  }
  return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

```
// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

```
// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
```

```
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
```

```
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
```

```
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
```

```
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
```

```
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
```

```
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Exemplos do Amazon DocumentDB usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Go.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"

 "github.com/aws/aws-lambda-go/lambda"
)
```

```
type Event struct {
 Events []Record `json:"events"`
}

type Record struct {
 Event struct {
  OperationType string `json:"operationType"`
  NS            struct {
   DB   string `json:"db"`
   Coll string `json:"coll"`
  } `json:"ns"`
  FullDocument interface{} `json:"fullDocument"`
 } `json:"event"`
}

func main() {
 lambda.Start(handler)
}

func handler(ctx context.Context, event Event) (string, error) {
 fmt.Println("Loading function")
 for _, record := range event.Events {
  logDocumentDBEvent(record)
 }

 return "OK", nil
}

func logDocumentDBEvent(record Record) {
 fmt.Printf("Operation type: %s\n", record.Event.OperationType)
 fmt.Printf("db: %s\n", record.Event.NS.DB)
 fmt.Printf("collection: %s\n", record.Event.NS.Coll)
 docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")
 fmt.Printf("Full document: %s\n", string(docBytes))
}
```

Exemplos do DynamoDB usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes em todo o mundo AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor
- AWS contribuições da comunidade

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo para criar a tabela e executar ações nela.

```
import (
 "context"
 "fmt"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunMovieScenario is an interactive example that shows you how to use the AWS SDK
 for Go
// to create and use an Amazon DynamoDB table that stores data about movies.
//
//  1. Create a table that can hold movie data.
//  2. Put, get, and update a single movie in the table.
//  3. Write movie data to the table from a sample JSON file.
//  4. Query for movies that were released in a given year.
//  5. Scan for movies that were released in a range of years.
//  6. Delete a movie from the table.
//  7. Delete the table.
//
// This example creates a DynamoDB service client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
```

```
// The specified movie sampler is used to get sample data from a URL that is loaded
// into the named table.
func RunMovieScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner,
 tableName string,
 movieSampler actions.IMovieSampler) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB getting started demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{TableName: tableName,
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig)}

 exists, err := tableBasics.TableExists(ctx)
 if err != nil {
  panic(err)
 }
 if !exists {
  log.Printf("Creating table %v...\n", tableName)
  _, err = tableBasics.CreateMovieTable(ctx)
  if err != nil {
   panic(err)
  } else {
   log.Printf("Created table %v.\n", tableName)
  }
 } else {
  log.Printf("Table %v already exists.\n", tableName)
 }

 var customMovie actions.Movie
 customMovie.Title = questioner.Ask("Enter a movie title to add to the table:",
  demotools.NotEmpty{})
 customMovie.Year = questioner.AskInt("What year was it released?",
  demotools.NotEmpty{}, demotools.InIntRange{Lower: 1900, Upper: 2030})
 customMovie.Info = map[string]interface{}{}
 customMovie.Info["rating"] = questioner.AskFloat64(
  "Enter a rating between 1 and 10:",
  demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
```

```
customMovie.Info["plot"] = questioner.Ask("What's the plot? ",
 demotools.NotEmpty{})
err = tableBasics.AddMovie(ctx, customMovie)
if err == nil {
 log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's update your movie. You previously rated it %v.\n",
customMovie.Info["rating"])
customMovie.Info["rating"] = questioner.AskFloat64(
 "What new rating would you give it?",
 demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
log.Printf("You summarized the plot as '%v'.\n", customMovie.Info["plot"])
customMovie.Info["plot"] = questioner.Ask("What would you say now?",
 demotools.NotEmpty{})
attributes, err := tableBasics.UpdateMovie(ctx, customMovie)
if err == nil {
 log.Printf("Updated %v with new values.\n", customMovie.Title)
 for _, attVal := range attributes {
  for valKey, val := range attVal {
   log.Printf("\t%v: %v\n", valKey, val)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting movie data from %v and adding 250 movies to the table...\n",
 movieSampler.GetURL())
movies := movieSampler.GetSampleMovies()
written, err := tableBasics.AddMovieBatch(ctx, movies, 250)
if err != nil {
 panic(err)
} else {
 log.Printf("Added %v movies to the table.\n", written)
}

show := 10
if show > written {
 show = written
}
log.Printf("The first %v movies in the table are:", show)
for index, movie := range movies[:show] {
 log.Printf("\t%v. %v\n", index+1, movie.Title)
```

```
}
movieIndex := questioner.AskInt(
 "Enter the number of a movie to get info about it: ",
 demotools.InIntRange{Lower: 1, Upper: show},
)
movie, err := tableBasics.GetMovie(ctx, movies[movieIndex-1].Title,
movies[movieIndex-1].Year)
if err == nil {
 log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Println("Let's get a list of movies released in a given year.")
releaseYear := questioner.AskInt("Enter a year between 1972 and 2018: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018},
)
releases, err := tableBasics.Query(ctx, releaseYear)
if err == nil {
 if len(releases) == 0 {
  log.Printf("I couldn't find any movies released in %v!\n", releaseYear)
 } else {
  for _, movie = range releases {
   log.Println(movie)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Println("Now let's scan for movies released in a range of years.")
startYear := questioner.AskInt("Enter a year: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018})
endYear := questioner.AskInt("Enter another year: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018})
releases, err = tableBasics.Scan(ctx, startYear, endYear)
if err == nil {
 if len(releases) == 0 {
  log.Printf("I couldn't find any movies released between %v and %v!\n", startYear,
endYear)
 } else {
  log.Printf("Found %v movies. In this list, the plot is <nil> because "+
   "we used a projection expression when scanning for items to return only "+
   "the title, year, and rating.\n", len(releases))
  for _, movie = range releases {
   log.Println(movie)
```

```
    }
   }
  }
  log.Println(strings.Repeat("-", 88))

  var tables []string
  if questioner.AskBool("Do you want to list all of your tables? (y/n) ", "y") {
   tables, err = tableBasics.ListTables(ctx)
   if err == nil {
    log.Printf("Found %v tables:", len(tables))
    for _, table := range tables {
     log.Printf("\t%v", table)
    }
   }
  }
  log.Println(strings.Repeat("-", 88))

  log.Printf("Let's remove your movie '%v'.\n", customMovie.Title)
  if questioner.AskBool("Do you want to delete it from the table? (y/n) ", "y") {
   err = tableBasics.DeleteMovie(ctx, customMovie)
  }
  if err == nil {
   log.Printf("Deleted %v.\n", customMovie.Title)
  }

  if questioner.AskBool("Delete the table, too? (y/n)", "y") {
   err = tableBasics.DeleteTable(ctx)
  } else {
   log.Println("Don't forget to delete the table when you're done or you might " +
    "incur charges on your account.")
  }
  if err == nil {
   log.Printf("Deleted table %v.\n", tableBasics.TableName)
  }

  log.Println(strings.Repeat("-", 88))
  log.Println("Thanks for watching!")
  log.Println(strings.Repeat("-", 88))
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
```

```
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Crie uma estrutura e métodos que chamem ações do DynamoDB.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
 exists := true
 _, err := basics.DynamoDbClient.DescribeTable(
  ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
 )
 if err != nil {
  var notFoundEx *types.ResourceNotFoundException
  if errors.As(err, &notFoundEx) {
   log.Printf("Table %v does not exist.\n", basics.TableName)
   err = nil
  } else {
```

```
    log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
 basics.TableName, err)
  }
  exists = false
 }
 return exists, err
}



// CreateMovieTable creates a DynamoDB table with a composite primary key defined as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
 (*types.TableDescription, error) {
 var tableDesc *types.TableDescription
 table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
  AttributeDefinitions: []types.AttributeDefinition{{
   AttributeName: aws.String("year"),
   AttributeType: types.ScalarAttributeTypeN,
  }, {
   AttributeName: aws.String("title"),
   AttributeType: types.ScalarAttributeTypeS,
  }},
  KeySchema: []types.KeySchemaElement{{
   AttributeName: aws.String("year"),
   KeyType:       types.KeyTypeHash,
  }, {
   AttributeName: aws.String("title"),
   KeyType:       types.KeyTypeRange,
  }},
  TableName:   aws.String(basics.TableName),
  BillingMode: types.BillingModePayPerRequest,
 })
 if err != nil {
  log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
 } else {
  waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
  err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
   TableName: aws.String(basics.TableName)}, 5*time.Minute)
  if err != nil {
   log.Printf("Wait for table exists failed. Here's why: %v\n", err)
  }
```

```
  tableDesc = table.TableDescription
  log.Printf("Ccreating table test")
 }
 return tableDesc, err
}



// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
 var tableNames []string
 var output *dynamodb.ListTablesOutput
 var err error
 tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
 &dynamodb.ListTablesInput{})
 for tablePaginator.HasMorePages() {
  output, err = tablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't list tables. Here's why: %v\n", err)
   break
  } else {
   tableNames = append(tableNames, output.TableNames...)
  }
 }
 return tableNames, err
}



// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
 item, err := attributevalue.MarshalMap(movie)
 if err != nil {
  panic(err)
 }
 _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
  TableName: aws.String(basics.TableName), Item: item,
 })
 if err != nil {
  log.Printf("Couldn't add item to table. Here's why: %v\n", err)
 }
 return err
}
```

```
// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
 (map[string]map[string]interface{}, error) {
 var err error
 var response *dynamodb.UpdateItemOutput
 var attributeMap map[string]map[string]interface{}
 update := expression.Set(expression.Name("info.rating"),
 expression.Value(movie.Info["rating"]))
 update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
 expr, err := expression.NewBuilder().WithUpdate(update).Build()
 if err != nil {
  log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
 } else {
  response, err = basics.DynamoDbClient.UpdateItem(ctx, &dynamodb.UpdateItemInput{
   TableName:                aws.String(basics.TableName),
   Key:                      movie.GetKey(),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   UpdateExpression:         expr.Update(),
   ReturnValues:             types.ReturnValueUpdatedNew,
  })
  if err != nil {
   log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
  } else {
   err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
   if err != nil {
    log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
   }
  }
 }
 return attributeMap, err
}



// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
 maxMovies int) (int, error) {
```

```
 var err error
 var item map[string]types.AttributeValue
 written := 0
 batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
 start := 0
 end := start + batchSize
 for start < maxMovies && start < len(movies) {
  var writeReqs []types.WriteRequest
  if end > len(movies) {
   end = len(movies)
  }
  for _, movie := range movies[start:end] {
   item, err = attributevalue.MarshalMap(movie)
   if err != nil {
    log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
 movie.Title, err)
   } else {
    writeReqs = append(
     writeReqs,
     types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
   }
  }
  _, err = basics.DynamoDbClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
   RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
  if err != nil {
   log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
 basics.TableName, err)
  } else {
   written += len(writeReqs)
  }
  start = end
  end += batchSize
 }

 return written, err
}



// GetMovie gets movie data from the DynamoDB table by using the primary composite
 key
// made of title and year.
```

```
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 movie := Movie{Title: title, Year: year}
 response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key: movie.GetKey(), TableName: aws.String(basics.TableName),
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Item, &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}



// Query gets all movies in the DynamoDB table that were released in the specified
 year.
// The function uses the `expression` package to build the key condition expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
 error) {
 var err error
 var response *dynamodb.QueryOutput
 var movies []Movie
 keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
 expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
 if err != nil {
  log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
 } else {
  queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
 &dynamodb.QueryInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   KeyConditionExpression:    expr.KeyCondition(),
  })
  for queryPaginator.HasMorePages() {
   response, err = queryPaginator.NextPage(ctx)
   if err != nil {
```

```
    log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
 releaseYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}



// Scan gets all movies in the DynamoDB table that were released in a range of years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
 ([]Movie, error) {
 var movies []Movie
 var err error
 var response *dynamodb.ScanOutput
 filtEx := expression.Name("year").Between(expression.Value(startYear),
 expression.Value(endYear))
 projEx := expression.NamesList(
  expression.Name("year"), expression.Name("title"), expression.Name("info.rating"))
 expr, err :=
 expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
 if err != nil {
  log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
 } else {
  scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
 &dynamodb.ScanInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   FilterExpression:          expr.Filter(),
```

```
   ProjectionExpression:      expr.Projection(),
  })
  for scanPaginator.HasMorePages() {
   response, err = scanPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v
\n",
     startYear, endYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}



// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
 _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
  TableName: aws.String(basics.TableName), Key: movie.GetKey(),
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}



// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
 _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
  TableName: aws.String(basics.TableName)})
```

```
 if err != nil {
  log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`BatchExecuteStatement`

O código de exemplo a seguir mostra como usar `BatchExecuteStatement`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina uma estrutura de receptor de função para o exemplo.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

Use lotes de instruções INSERT para adicionar itens.

```
// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies to
 the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie) error
 {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(fmt.Sprintf(
    "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
 runner.TableName)),
   Parameters: params,
```

```
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n", err)
 }
 return err
}
```

Use lotes de instruções SELECT para obter itens.

```
// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
 from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
 ([]Movie, error) {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
   Parameters: params,
  }
 }

 output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 var outMovies []Movie
 if err != nil {
  log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
```

```
 } else {
  for _, response := range output.Responses {
   var movie Movie
   err = attributevalue.UnmarshalMap(response.Item, &movie)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    outMovies = append(outMovies, movie)
   }
  }
 }
 return outMovies, err
}
```

Use lotes de instruções UPDATE para atualizar itens.

```
// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating
 of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
 ratings []float64) error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{ratings[index],
 movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
```

```
 if err != nil {
  log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

Use lotes de instruções DELETE para excluir itens.

```
// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
 movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
```

```
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para Go da API.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

```
// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
 maxMovies int) (int, error) {
 var err error
 var item map[string]types.AttributeValue
 written := 0
 batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
 start := 0
 end := start + batchSize
 for start < maxMovies && start < len(movies) {
  var writeReqs []types.WriteRequest
  if end > len(movies) {
   end = len(movies)
  }
  for _, movie := range movies[start:end] {
   item, err = attributevalue.MarshalMap(movie)
   if err != nil {
    log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
 movie.Title, err)
   } else {
    writeReqs = append(
     writeReqs,
     types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
   }
  }
  _, err = basics.DynamoDbClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
   RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
  if err != nil {
   log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
 basics.TableName, err)
  } else {
   written += len(writeReqs)
  }
  start = end
  end += batchSize
 }

 return written, err
```

```
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
```

```
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte BatchWriteItem em Referência AWS SDK para Go da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)
```

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// CreateMovieTable creates a DynamoDB table with a composite primary key defined as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
 (*types.TableDescription, error) {
 var tableDesc *types.TableDescription
 table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
  AttributeDefinitions: []types.AttributeDefinition{{
   AttributeName: aws.String("year"),
   AttributeType: types.ScalarAttributeTypeN,
  }, {
   AttributeName: aws.String("title"),
   AttributeType: types.ScalarAttributeTypeS,
  }},
  KeySchema: []types.KeySchemaElement{{
   AttributeName: aws.String("year"),
   KeyType:       types.KeyTypeHash,
  }, {
   AttributeName: aws.String("title"),
   KeyType:       types.KeyTypeRange,
  }},
  TableName:   aws.String(basics.TableName),
  BillingMode: types.BillingModePayPerRequest,
 })
 if err != nil {
  log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
 } else {
  waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
  err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
   TableName: aws.String(basics.TableName)}, 5*time.Minute)
  if err != nil {
   log.Printf("Wait for table exists failed. Here's why: %v\n", err)
  }
```

```
  tableDesc = table.TableDescription
  log.Printf("Ccreating table test")
 }
 return tableDesc, err
}
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para Go da API.

DeleteItem

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
```

```
}



// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
 _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
  TableName: aws.String(basics.TableName), Key: movie.GetKey(),
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
```

```
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte DeleteItem em Referência AWS SDK para Go da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
```

```
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
 _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
  TableName: aws.String(basics.TableName)})
 if err != nil {
  log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para Go da API.

`DescribeTable`

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
 exists := true
 _, err := basics.DynamoDbClient.DescribeTable(
  ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
 )
 if err != nil {
  var notFoundEx *types.ResourceNotFoundException
  if errors.As(err, &notFoundEx) {
   log.Printf("Table %v does not exist.\n", basics.TableName)
```

```
    err = nil
   } else {
    log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
 basics.TableName, err)
   }
   exists = false
  }
  return exists, err
}
```

- Para obter detalhes da API, consulte DescribeTable em Referência AWS SDK para Go da API.

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina uma estrutura de receptor de função para o exemplo.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
```

```
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

Use uma instrução INSERT para adicionar um item.

```
// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
 }
 return err
}
```

Use uma instrução SELECT para obter um item.

```
// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table
 by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
```

```
 var movie Movie
 params, err := attributevalue.MarshalList([]interface{}{title, year})
 if err != nil {
  panic(err)
 }
 response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Items[0], &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}
```

Use uma instrução SELECT para obter uma lista de itens e projetar os resultados.

```
// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
 table.
// pageSize is not typically required and is used to show how to paginate the
 results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
 ([]map[string]interface{}, error) {
 var output []map[string]interface{}
 var response *dynamodb.ExecuteStatementOutput
 var err error
 var nextToken *string
 for moreData := true; moreData; {
  response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
   Statement: aws.String(
```

```
    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
   Limit:     aws.Int32(pageSize),
   NextToken: nextToken,
  })
  if err != nil {
   log.Printf("Couldn't get movies. Here's why: %v\n", err)
   moreData = false
  } else {
   var pageOutput []map[string]interface{}
   err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    log.Printf("Got a page of length %v.\n", len(response.Items))
    output = append(output, pageOutput...)
   }
   nextToken = response.NextToken
   moreData = nextToken != nil
  }
 }
 return output, err
}
```

Use uma instrução UPDATE para atualizar um item.

```
// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
 float64) error {
 params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
```

```
 })
 if err != nil {
  log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
 }
 return err
}
```

Use uma instrução DELETE para excluir um item.

```
// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
 table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
```

```
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para Go da API.

GetItem

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// GetMovie gets movie data from the DynamoDB table by using the primary composite
 key
// made of title and year.
```

```
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 movie := Movie{Title: title, Year: year}
 response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key: movie.GetKey(), TableName: aws.String(basics.TableName),
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Item, &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
```

```
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte GetItem em Referência AWS SDK para Go da API.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
```

```
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
 var tableNames []string
 var output *dynamodb.ListTablesOutput
 var err error
 tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
 &dynamodb.ListTablesInput{})
 for tablePaginator.HasMorePages() {
  output, err = tablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't list tables. Here's why: %v\n", err)
   break
  } else {
   tableNames = append(tableNames, output.TableNames...)
  }
 }
 return tableNames, err
}
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para Go da API.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
 item, err := attributevalue.MarshalMap(movie)
 if err != nil {
  panic(err)
 }
 _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
```

```
  TableName: aws.String(basics.TableName), Item: item,
 })
 if err != nil {
  log.Printf("Couldn't add item to table. Here's why: %v\n", err)
 }
 return err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
```

```
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK para Go da API.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
```

```
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// Query gets all movies in the DynamoDB table that were released in the specified
 year.
// The function uses the `expression` package to build the key condition expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
 error) {
 var err error
 var response *dynamodb.QueryOutput
 var movies []Movie
 keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
 expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
 if err != nil {
  log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
 } else {
  queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
 &dynamodb.QueryInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   KeyConditionExpression:    expr.KeyCondition(),
  })
  for queryPaginator.HasMorePages() {
   response, err = queryPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
 releaseYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
```

```
      log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
      break
     } else {
      movies = append(movies, moviePage...)
     }
    }
   }
  }
 return movies, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
```

```
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Go .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// Scan gets all movies in the DynamoDB table that were released in a range of years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
 ([]Movie, error) {
 var movies []Movie
 var err error
 var response *dynamodb.ScanOutput
 filtEx := expression.Name("year").Between(expression.Value(startYear),
 expression.Value(endYear))
 projEx := expression.NamesList(
  expression.Name("year"), expression.Name("title"), expression.Name("info.rating"))
 expr, err :=
 expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
 if err != nil {
  log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
 } else {
  scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
 &dynamodb.ScanInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   FilterExpression:          expr.Filter(),
   ProjectionExpression:      expr.Projection(),
  })
  for scanPaginator.HasMorePages() {
   response, err = scanPaginator.NextPage(ctx)
```

```
   if err != nil {
    log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v
\n",
     startYear, endYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
```

```
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Go .

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
 (map[string]map[string]interface{}, error) {
 var err error
 var response *dynamodb.UpdateItemOutput
 var attributeMap map[string]map[string]interface{}
 update := expression.Set(expression.Name("info.rating"),
 expression.Value(movie.Info["rating"]))
 update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
 expr, err := expression.NewBuilder().WithUpdate(update).Build()
 if err != nil {
  log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
 } else {
  response, err = basics.DynamoDbClient.UpdateItem(ctx, &dynamodb.UpdateItemInput{
   TableName:                 aws.String(basics.TableName),
   Key:                       movie.GetKey(),
   ExpressionAttributeNames:  expr.Names(),
```

```
    ExpressionAttributeValues: expr.Values(),
    UpdateExpression:          expr.Update(),
    ReturnValues:              types.ReturnValueUpdatedNew,
   })
   if err != nil {
    log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
   } else {
    err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
    if err != nil {
     log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
    }
   }
  }
  return attributeMap, err
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
```

```
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para Go da API.

Cenários

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário que crie uma tabela e execute lotes de consultas do PartiQL.

```
import (
 "context"
 "fmt"
 "log"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunPartiQLBatchScenario shows you how to use the AWS SDK for Go
// to run batches of PartiQL statements to query a table that stores data about
 movies.
//
//   - Use batches of PartiQL statements to add, get, update, and delete data for
//     individual movies.
//
// This example creates an Amazon DynamoDB service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLBatchScenario(ctx context.Context, sdkConfig aws.Config, tableName
 string) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()
```

```
log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon DynamoDB PartiQL batch demo.")
log.Println(strings.Repeat("-", 88))

tableBasics := actions.TableBasics{
 DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
 TableName:      tableName,
}
runner := actions.PartiQLRunner{
 DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
 TableName:      tableName,
}

exists, err := tableBasics.TableExists(ctx)
if err != nil {
 panic(err)
}
if !exists {
 log.Printf("Creating table %v...\n", tableName)
 _, err = tableBasics.CreateMovieTable(ctx)
 if err != nil {
  panic(err)
 } else {
  log.Printf("Created table %v.\n", tableName)
 }
} else {
 log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovies := []actions.Movie{{
 Title: "House PartiQL",
 Year:  currentYear - 5,
 Info: map[string]interface{}{
  "plot":   "Wacky high jinks result from querying a mysterious database.",
  "rating": 8.5}}, {
 Title: "House PartiQL 2",
 Year:  currentYear - 3,
 Info: map[string]interface{}{
  "plot":   "Moderate high jinks result from querying another mysterious
database.",
  "rating": 6.5}}, {
 Title: "House PartiQL 3",
```

```
 Year:  currentYear - 1,
 Info: map[string]interface{}{
  "plot":   "Tepid high jinks result from querying yet another mysterious
database.",
  "rating": 2.5},
},
}

log.Printf("Inserting a batch of movies into table '%v'.\n", tableName)
err = runner.AddMovieBatch(ctx, customMovies)
if err == nil {
 log.Printf("Added %v movies to the table.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting data for a batch of movies.")
movies, err := runner.GetMovieBatch(ctx, customMovies)
if err == nil {
 for _, movie := range movies {
  log.Println(movie)
 }
}
log.Println(strings.Repeat("-", 88))

newRatings := []float64{7.7, 4.4, 1.1}
log.Println("Updating a batch of movies with new ratings.")
err = runner.UpdateMovieBatch(ctx, customMovies, newRatings)
if err == nil {
 log.Printf("Updated %v movies with new ratings.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting projected data from the table to verify our update.")
log.Println("Using a page size of 2 to demonstrate paging.")
projections, err := runner.GetAllMovies(ctx, 2)
if err == nil {
 log.Println("All movies:")
 for _, projection := range projections {
  log.Println(projection)
 }
}
log.Println(strings.Repeat("-", 88))

log.Println("Deleting a batch of movies.")
```

```
 err = runner.DeleteMovieBatch(ctx, customMovies)
 if err == nil {
  log.Printf("Deleted %v movies.\n", len(customMovies))
 }

 err = tableBasics.DeleteTable(ctx)
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}
```

```
// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Crie uma estrutura e métodos que executem declarações PartiQL.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
```

```
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies to
 the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie) error
 {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(fmt.Sprintf(
    "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n", err)
 }
 return err
}



// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
 from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
 ([]Movie, error) {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
```

```
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
   Parameters: params,
  }
 }

 output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 var outMovies []Movie
 if err != nil {
  log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
 } else {
  for _, response := range output.Responses {
   var movie Movie
   err = attributevalue.UnmarshalMap(response.Item, &movie)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    outMovies = append(outMovies, movie)
   }
  }
 }
 return outMovies, err
}



// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
 table.
// pageSize is not typically required and is used to show how to paginate the
 results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
 ([]map[string]interface{}, error) {
 var output []map[string]interface{}
 var response *dynamodb.ExecuteStatementOutput
```

```
 var err error
 var nextToken *string
 for moreData := true; moreData; {
  response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
   Statement: aws.String(
    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
   Limit:     aws.Int32(pageSize),
   NextToken: nextToken,
  })
  if err != nil {
   log.Printf("Couldn't get movies. Here's why: %v\n", err)
   moreData = false
  } else {
   var pageOutput []map[string]interface{}
   err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    log.Printf("Got a page of length %v.\n", len(response.Items))
    output = append(output, pageOutput...)
   }
   nextToken = response.NextToken
   moreData = nextToken != nil
  }
 }
 return output, err
}



// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating
 of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
 ratings []float64) error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{ratings[index],
 movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
```

```
    Statement: aws.String(
     fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
 runner.TableName)),
    Parameters: params,
   }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
 }
 return err
}



// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
 movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
 }
```

```
 return err
}
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para Go da API.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário que crie uma tabela e execute consultas do PartiQL.

```
import (
 "context"
 "fmt"
 "log"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)
```

```
// RunPartiQLSingleScenario shows you how to use the AWS SDK for Go
// to use PartiQL to query a table that stores data about movies.
//
// * Use PartiQL statements to add, get, update, and delete data for individual
 movies.
//
// This example creates an Amazon DynamoDB service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLSingleScenario(ctx context.Context, sdkConfig aws.Config, tableName
 string) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB PartiQL single action demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }
 runner := actions.PartiQLRunner{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }

 exists, err := tableBasics.TableExists(ctx)
 if err != nil {
  panic(err)
 }
 if !exists {
  log.Printf("Creating table %v...\n", tableName)
  _, err = tableBasics.CreateMovieTable(ctx)
  if err != nil {
   panic(err)
  } else {
   log.Printf("Created table %v.\n", tableName)
  }
```

```
} else {
 log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovie := actions.Movie{
 Title: "24 Hour PartiQL People",
 Year:  currentYear,
 Info: map[string]interface{}{
  "plot":   "A group of data developers discover a new query language they can't
stop using.",
  "rating": 9.9,
 },
}

log.Printf("Inserting movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
err = runner.AddMovie(ctx, customMovie)
if err == nil {
 log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting data for movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
movie, err := runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
if err == nil {
 log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

newRating := 6.6
log.Printf("Updating movie '%v' with a rating of %v.", customMovie.Title,
newRating)
err = runner.UpdateMovie(ctx, customMovie, newRating)
if err == nil {
 log.Printf("Updated %v with a new rating.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting data again to verify the update.")
movie, err = runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
if err == nil {
```

```
  log.Println(movie)
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Deleting movie '%v'.\n", customMovie.Title)
 err = runner.DeleteMovie(ctx, customMovie)
 if err == nil {
  log.Printf("Deleted %v.\n", customMovie.Title)
 }

 err = tableBasics.DeleteTable(ctx)
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina uma estrutura de filme usada neste exemplo.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
```

```
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Crie uma estrutura e métodos que executem declarações PartiQL.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)
```

```
// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
 }
 return err
}



// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table
 by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 var movie Movie
 params, err := attributevalue.MarshalList([]interface{}{title, year})
 if err != nil {
  panic(err)
 }
```

```
 response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Items[0], &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}



// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
 float64) error {
 params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
 }
 return err
}
```

```
// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
 table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para Go da API.

Exemplos sem servidor

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-lambda-go/events"
 "fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string, error)
 {
 if len(event.Records) == 0 {
  return nil, fmt.Errorf("received empty event")
 }

 for _, record := range event.Records {
   LogDynamoDBRecord(record)
 }

 message := fmt.Sprintf("Records processed: %d", len(event.Records))
 return &message, nil
}

func main() {
 lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
 fmt.Println(record.EventID)
 fmt.Println(record.EventName)
 fmt.Printf("%+v\n", record.Change)
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
 ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
 BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*BatchResult,
 error) {
 var batchItemFailures []BatchItemFailure
 curRecordSequenceNumber := ""

 for _, record := range event.Records {
  // Process your record
  curRecordSequenceNumber = record.Change.SequenceNumber
 }

 if curRecordSequenceNumber != "" {
  batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
 curRecordSequenceNumber})
 }
```

```
 batchResult := BatchResult{
  BatchItemFailures: batchItemFailures,
 }

 return &batchResult, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Go V2

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Go.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos do IAM usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access Management
 (IAM)
```

```
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 iamClient := iam.NewFromConfig(sdkConfig)
 const maxPols = 10
 fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
 result, err := iamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPols),
 })
 if err != nil {
  fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Policies) == 0 {
  fmt.Println("You don't have any policies!")
 } else {
  for _, policy := range result.Policies {
   fmt.Printf("\t%v\n", *policy.PolicyName)
  }
 }
}
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para Go da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/credentials"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/sts"
```

```
 "github.com/aws/smithy-go"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/iam/actions"
)

// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
//  1. Create a user who has no permissions.
//  2. Create a role that grants permission to list Amazon Simple Storage Service
//     (Amazon S3) buckets for the account.
//  3. Add a policy to let the user assume the role.
//  4. Try and fail to list buckets without permissions.
//  5. Assume the role and list S3 buckets using temporary credentials.
//  6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig      aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper  actions.PolicyWrapper
 roleWrapper    actions.RoleWrapper
 userWrapper    actions.UserWrapper
 questioner     demotools.IQuestioner
 helper         IScenarioHelper
 isTestRun      bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
 return AssumeRoleScenario{
  sdkConfig:      sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
```

```
}

// addTestOptions appends the API options specified in the original configuration to
// another configuration. This is used to attach the middleware stubber to clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
   log.Println(r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
 demo.")
 log.Println(strings.Repeat("-", 88))

 user := scenario.CreateUser(ctx)
 accessKey := scenario.CreateAccessKey(ctx, user)
 role := scenario.CreateRoleAndPolicies(ctx, user)
 noPermsConfig := scenario.ListBucketsWithoutPermissions(ctx, accessKey)
 scenario.ListBucketsWithAssumedRole(ctx, noPermsConfig, role)
 scenario.Cleanup(ctx, user, role)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser(ctx context.Context) *types.User {
 log.Println("Let's create an example user with no permissions.")
 userName := scenario.questioner.Ask("Enter a name for the example user:",
 demotools.NotEmpty{})
 user, err := scenario.userWrapper.GetUser(ctx, userName)
```

```
 if err != nil {
  panic(err)
 }
 if user == nil {
  user, err = scenario.userWrapper.CreateUser(ctx, userName)
  if err != nil {
   panic(err)
  }
  log.Printf("Created user %v.\n", *user.UserName)
 } else {
  log.Printf("User %v already exists.\n", *user.UserName)
 }
 log.Println(strings.Repeat("-", 88))
 return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(ctx context.Context, user
 *types.User) *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(ctx, *user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3 buckets
 for
// the current account and attaches the policy to a newly created role. It also adds
 an
// inline policy to the specified user that grants the user permission to assume the
 role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(ctx context.Context, user
 *types.User) *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err := scenario.roleWrapper.CreateRole(ctx,
 scenario.helper.GetName(), *user.Arn)
 if err != nil {
```

```
  panic(err)
 }
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  ctx, scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
 if err != nil {
  panic(err)
 }
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(ctx, *listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {
  panic(err)
 }
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(ctx, *user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's access
 key
// credentials and tries to list buckets for the account. Because the user does not
 have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(ctx
 context.Context, accessKey *types.AccessKey) *aws.Config {
 log.Println("Let's try to list buckets without permissions. This should return an
 AccessDenied error.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 noPermsConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
```

```
    *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
   ))
  if err != nil {
   panic(err)
  }

  // Add test options if this is a test run. This is needed only for testing
  purposes.
  scenario.addTestOptions(&noPermsConfig)

  s3Client := s3.NewFromConfig(noPermsConfig)
  _, err = s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
  if err != nil {
   // The SDK for Go does not model the AccessDenied error, so check ErrorCode
  directly.
   var ae smithy.APIError
   if errors.As(err, &ae) {
    switch ae.ErrorCode() {
    case "AccessDenied":
     log.Println("Got AccessDenied error, which is the expected result because\n" +
      "the ListBuckets call was made without permissions.")
    default:
     log.Println("Expected AccessDenied, got something else.")
     panic(err)
    }
   }
  } else {
   log.Println("Expected AccessDenied error when calling ListBuckets without
  permissions,\n" +
    "but the call succeeded. Continuing the example anyway...")
  }
  log.Println(strings.Repeat("-", 88))
  return &noPermsConfig
 }

 // ListBucketsWithAssumedRole performs the following actions:
 //
 //  1. Creates an AWS Security Token Service (AWS STS) client from the config
  created from
 //     the user's access key credentials.
 //  2. Gets temporary credentials by assuming the role that grants permission to
  list the
 //     buckets.
 //  3. Creates an Amazon S3 client from the temporary credentials.
```

```
//  4. Lists buckets for the account. Because the temporary credentials are
 generated by
//     assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(ctx context.Context,
 noPermsConfig *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and try
 again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(ctx, &sts.AssumeRoleInput{
  RoleArn:         role.Arn,
  RoleSessionName: aws.String("AssumeRoleExampleSession"),
  DurationSeconds: aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
 )
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n" +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
```

```
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(ctx context.Context, user *types.User,
 role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
  policies, err := scenario.roleWrapper.ListAttachedRolePolicies(ctx,
 *role.RoleName)
  if err != nil {
   panic(err)
  }
  for _, policy := range policies {
   err = scenario.roleWrapper.DetachRolePolicy(ctx, *role.RoleName,
 *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   err = scenario.policyWrapper.DeletePolicy(ctx, *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   log.Printf("Detached policy %v from role %v and deleted the policy.\n",
    *policy.PolicyName, *role.RoleName)
  }
  err = scenario.roleWrapper.DeleteRole(ctx, *role.RoleName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted role %v.\n", *role.RoleName)

  userPols, err := scenario.userWrapper.ListUserPolicies(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, userPol := range userPols {
   err = scenario.userWrapper.DeleteUserPolicy(ctx, *user.UserName, userPol)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
```

```
  }
  keys, err := scenario.userWrapper.ListAccessKeys(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, key := range keys {
   err = scenario.userWrapper.DeleteAccessKey(ctx, *user.UserName, *key.AccessKeyId)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
 *user.UserName)
  }
  err = scenario.userWrapper.DeleteUser(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted user %v.\n", *user.UserName)
  log.Println(strings.Repeat("-", 88))
 }

}

// IScenarioHelper abstracts input and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
 Pause(secs int)
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper *ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
```

```
 time.Sleep(time.Duration(secs) * time.Second)
}
```

Defina um struct que encapsule as ações de conta.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

```
// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

Defina um struct que encapsule as ações de política.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
```

```
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
```

```
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
```

```
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

Defina um struct que encapsule as ações de perfil.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
```

```
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
```

```
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context, serviceName
 string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n", roleName,
 err)
```

```
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn string,
 roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
```

```
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName, err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx, &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName, err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

Defina um struct que encapsule as ações de usuário.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
```

```
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a policy
 that
```

```
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx, &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName, err)
 } else {
  policies = result.PolicyNames
```

```
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
```

```
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn string,
 roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte AttachRolePolicy em Referência AWS SDK para Go da API.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
```

```
 }
 return key, err
}
```

- Para obter detalhes da API, consulte CreateAccessKey em Referência AWS SDK para Go da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}
```

```
// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
```

```
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK para Go da API.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}
```

```
// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- Para obter detalhes da API, consulte CreateRole em Referência AWS SDK para Go da API.

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context, serviceName
 string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
```

```
 } else {
  role = result.Role
 }
 return role, err
}
```

- Para obter detalhes da API, consulte CreateServiceLinkedRole em Referência AWS SDK para Go da API.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}
```

```
// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}
```

- Para obter detalhes da API, consulte CreateUser em Referência AWS SDK para Go da API.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteAccessKey em Referência AWS SDK para Go da API.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeletePolicy em Referência AWS SDK para Go da API.

DeleteRole

O código de exemplo a seguir mostra como usar `DeleteRole`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteRole em Referência AWS SDK para Go da API.

`DeleteServiceLinkedRole`

O código de exemplo a seguir mostra como usar `DeleteServiceLinkedRole`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
```

```
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteServiceLinkedRole em Referência AWS SDK para Go da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}
```

```
// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para Go da API.

`DeleteUserPolicy`

O código de exemplo a seguir mostra como usar `DeleteUserPolicy`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)
```

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteUserPolicy em Referência AWS SDK para Go da API.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
```

```
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DetachRolePolicy em Referência AWS SDK para Go da API.

GetAccountPasswordPolicy

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicy em Referência AWS SDK para Go da API.

`GetPolicy`

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// GetPolicy gets data about a policy.
```

```
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- Para obter detalhes da API, consulte GetPolicy em Referência AWS SDK para Go da API.

`GetRole`

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
```

```
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- Para obter detalhes da API, consulte GetRole em Referência AWS SDK para Go da API.

`GetUser`

O código de exemplo a seguir mostra como usar `GetUser`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
```

```
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}
```

- Para obter detalhes da API, consulte GetUser em Referência AWS SDK para Go da API.

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
```

```
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- Para obter detalhes da API, consulte ListAccessKeys em Referência AWS SDK para Go da API.

`ListAttachedRolePolicies`

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}
```

```
// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}
```

- Para obter detalhes da API, consulte ListAttachedRolePolicies em Referência AWS SDK para Go da API.

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
 IamClient *iam.Client
}



// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(ctx context.Context, maxGroups int32)
 ([]types.Group, error) {
 var groups []types.Group
 result, err := wrapper.IamClient.ListGroups(ctx, &iam.ListGroupsInput{
  MaxItems: aws.Int32(maxGroups),
 })
 if err != nil {
  log.Printf("Couldn't list groups. Here's why: %v\n", err)
 } else {
  groups = result.Groups
 }
 return groups, err
}
```

- Para obter detalhes da API, consulte ListGroups em Referência AWS SDK para Go da API.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
```

```
}
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para Go da API.

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
```

```
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx, &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName, err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

- Para obter detalhes da API, consulte ListRolePolicies em Referência AWS SDK para Go da API.

`ListRoles`

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
```

```
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}
```

- Para obter detalhes da API, consulte ListRoles em Referência AWS SDK para Go da API.

Listar provedores de SAML

O código de exemplo a seguir mostra como usar `ListSAMLProviders`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
```

```
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK para Go .

`ListUserPolicies`

O código de exemplo a seguir mostra como usar `ListUserPolicies`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx, &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName, err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
```

```
}
```

- Para obter detalhes da API, consulte ListUserPolicies em Referência AWS SDK para Go da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
```

```
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para Go da API.

`PutUserPolicy`

O código de exemplo a seguir mostra como usar `PutUserPolicy`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)
```

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUserPolicy adds an inline policy to a user. This example creates a policy
 that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte PutUserPolicy em Referência AWS SDK para Go da API.

Exemplos do Kinesis usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Kinesis.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"
```

```
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
 if len(kinesisEvent.Records) == 0 {
  log.Printf("empty Kinesis event received")
  return nil
 }

 for _, record := range kinesisEvent.Records {
  log.Printf("processed Kinesis event with EventId: %v", record.EventID)
  recordDataBytes := record.Kinesis.Data
  recordDataText := string(recordDataBytes)
  log.Printf("record data: %v", recordDataText)
  // TODO: Do interesting work based on the new data
 }
 log.Printf("successfully processed %v records", len(kinesisEvent.Records))
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, record := range kinesisEvent.Records {
  curRecordSequenceNumber := ""

  // Process your record
  if /* Your record processing condition here */ {
   curRecordSequenceNumber = record.Kinesis.SequenceNumber
  }

  // Add a condition to check if the record processing failed
  if curRecordSequenceNumber != "" {
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": curRecordSequenceNumber})
  }
 }

 kinesisBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return kinesisBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Exemplos do Lambda usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor
- AWS contribuições da comunidade

Conceitos básicos

Olá, Lambda

O exemplo de código a seguir mostra como começar a usar o Lambda.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
)

// main uses the AWS SDK for Go (v2) to create an AWS Lambda client and list up to
 10
// functions in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 lambdaClient := lambda.NewFromConfig(sdkConfig)

 maxItems := 10
 fmt.Printf("Let's list up to %v functions for your account.\n", maxItems)
 result, err := lambdaClient.ListFunctions(ctx, &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 if err != nil {
```

```
  fmt.Printf("Couldn't list functions for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Functions) == 0 {
  fmt.Println("You don't have any functions!")
 } else {
  for _, function := range result.Functions {
   fmt.Printf("\t%v\n", *function.FunctionName)
  }
 }
}
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para Go da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um cenário interativo para mostrar como começar a usar funções do Lambda.

```
import (
 "archive/zip"
 "bytes"
 "context"
 "encoding/base64"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "os"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 iamtypes "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/lambda/actions"
)

// GetStartedFunctionsScenario shows you how to use AWS Lambda to perform the
 following
// actions:
//
//  1. Create an AWS Identity and Access Management (IAM) role and Lambda function,
 then upload handler code.
//  2. Invoke the function with a single parameter and get results.
//  3. Update the function code and configure with an environment variable.
//  4. Invoke the function with new parameters and get results. Display the returned
 execution log.
//  5. List the functions for your account, then clean up resources.
type GetStartedFunctionsScenario struct {
 sdkConfig       aws.Config
 functionWrapper actions.FunctionWrapper
 questioner      demotools.IQuestioner
 helper          IScenarioHelper
 isTestRun       bool
}

// NewGetStartedFunctionsScenario constructs a GetStartedFunctionsScenario instance
 from a configuration.
```

```
// It uses the specified config to get a Lambda client and create wrappers for the
 actions
// used in the scenario.
func NewGetStartedFunctionsScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedFunctionsScenario {
 lambdaClient := lambda.NewFromConfig(sdkConfig)
 return GetStartedFunctionsScenario{
  sdkConfig:       sdkConfig,
  functionWrapper: actions.FunctionWrapper{LambdaClient: lambdaClient},
  questioner:      questioner,
  helper:          helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedFunctionsScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Lambda get started with functions demo.")
 log.Println(strings.Repeat("-", 88))

 role := scenario.GetOrCreateRole(ctx)
 funcName := scenario.CreateFunction(ctx, role)
 scenario.InvokeIncrement(ctx, funcName)
 scenario.UpdateFunction(ctx, funcName)
 scenario.InvokeCalculator(ctx, funcName)
 scenario.ListFunctions(ctx)
 scenario.Cleanup(ctx, role, funcName)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// GetOrCreateRole checks whether the specified role exists and returns it if it
 does.
// Otherwise, a role is created that specifies Lambda as a trusted principal.
```

```
// The AWSLambdaBasicExecutionRole managed policy is attached to the role and the
 role
// is returned.
func (scenario GetStartedFunctionsScenario) GetOrCreateRole(ctx context.Context)
 *iamtypes.Role {
 var role *iamtypes.Role
 iamClient := iam.NewFromConfig(scenario.sdkConfig)
 log.Println("First, we need an IAM role that Lambda can assume.")
 roleName := scenario.questioner.Ask("Enter a name for the role:",
 demotools.NotEmpty{})
 getOutput, err := iamClient.GetRole(ctx, &iam.GetRoleInput{
  RoleName: aws.String(roleName)})
 if err != nil {
  var noSuch *iamtypes.NoSuchEntityException
  if errors.As(err, &noSuch) {
   log.Printf("Role %v doesn't exist. Creating it....\n", roleName)
  } else {
   log.Panicf("Couldn't check whether role %v exists. Here's why: %v\n",
    roleName, err)
  }
 } else {
  role = getOutput.Role
  log.Printf("Found role %v.\n", *role.RoleName)
 }
 if role == nil {
  trustPolicy := PolicyDocument{
   Version: "2012-10-17",
   Statement: []PolicyStatement{{
    Effect:    "Allow",
    Principal: map[string]string{"Service": "lambda.amazonaws.com"},
    Action:    []string{"sts:AssumeRole"},
   }},
  }
  policyArn := "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
  createOutput, err := iamClient.CreateRole(ctx, &iam.CreateRoleInput{
   AssumeRolePolicyDocument: aws.String(trustPolicy.String()),
   RoleName:                 aws.String(roleName),
  })
  if err != nil {
   log.Panicf("Couldn't create role %v. Here's why: %v\n", roleName, err)
  }
  role = createOutput.Role
  _, err = iamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
   PolicyArn: aws.String(policyArn),
```

```
    RoleName:  aws.String(roleName),
   })
   if err != nil {
    log.Panicf("Couldn't attach a policy to role %v. Here's why: %v\n", roleName,
  err)
   }
   log.Printf("Created role %v.\n", *role.RoleName)
   log.Println("Let's give AWS a few seconds to propagate resources...")
   scenario.helper.Pause(10)
  }
  log.Println(strings.Repeat("-", 88))
  return role
}

// CreateFunction creates a Lambda function and uploads a handler written in Python.
// The code for the Python handler is packaged as a []byte in .zip format.
func (scenario GetStartedFunctionsScenario) CreateFunction(ctx context.Context, role
 *iamtypes.Role) string {
 log.Println("Let's create a function that increments a number.\n" +
  "The function uses the 'lambda_handler_basic.py' script found in the \n" +
  "'handlers' directory of this project.")
 funcName := scenario.questioner.Ask("Enter a name for the Lambda function:",
 demotools.NotEmpty{})
 zipPackage := scenario.helper.CreateDeploymentPackage("lambda_handler_basic.py",
 fmt.Sprintf("%v.py", funcName))
 log.Printf("Creating function %v and waiting for it to be ready.", funcName)
 funcState := scenario.functionWrapper.CreateFunction(ctx, funcName,
 fmt.Sprintf("%v.lambda_handler", funcName),
  role.Arn, zipPackage)
 log.Printf("Your function is %v.", funcState)
 log.Println(strings.Repeat("-", 88))
 return funcName
}

// InvokeIncrement invokes a Lambda function that increments a number. The function
// parameters are contained in a Go struct that is used to serialize the parameters
 to
// a JSON payload that is passed to the function.
// The result payload is deserialized into a Go struct that contains an int value.
func (scenario GetStartedFunctionsScenario) InvokeIncrement(ctx context.Context,
 funcName string) {
 parameters := actions.IncrementParameters{Action: "increment"}
 log.Println("Let's invoke our function. This function increments a number.")
```

```
 parameters.Number = scenario.questioner.AskInt("Enter a number to increment:",
 demotools.NotEmpty{})
 log.Printf("Invoking %v with %v...\n", funcName, parameters.Number)
 invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, parameters, false)
 var payload actions.LambdaResultInt
 err := json.Unmarshal(invokeOutput.Payload, &payload)
 if err != nil {
  log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
   funcName, err)
 }
 log.Printf("Invoking %v with %v returned %v.\n", funcName, parameters.Number,
 payload)
 log.Println(strings.Repeat("-", 88))
}

// UpdateFunction updates the code for a Lambda function by uploading a simple
 arithmetic
// calculator written in Python. The code for the Python handler is packaged as a
// []byte in .zip format.
// After the code is updated, the configuration is also updated with a new log
// level that instructs the handler to log additional information.
func (scenario GetStartedFunctionsScenario) UpdateFunction(ctx context.Context,
 funcName string) {
 log.Println("Let's update the function to an arithmetic calculator.\n" +
  "The function uses the 'lambda_handler_calculator.py' script found in the \n" +
  "'handlers' directory of this project.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 log.Println("Creating deployment package...")
 zipPackage :=
 scenario.helper.CreateDeploymentPackage("lambda_handler_calculator.py",
  fmt.Sprintf("%v.py", funcName))
 log.Println("...and updating the Lambda function and waiting for it to be ready.")
 funcState := scenario.functionWrapper.UpdateFunctionCode(ctx, funcName, zipPackage)
 log.Printf("Updated function %v. Its current state is %v.", funcName, funcState)
 log.Println("This function uses an environment variable to control logging level.")
 log.Println("Let's set it to DEBUG to get the most logging.")
 scenario.functionWrapper.UpdateFunctionConfiguration(ctx, funcName,
  map[string]string{"LOG_LEVEL": "DEBUG"})
 log.Println(strings.Repeat("-", 88))
}

// InvokeCalculator invokes the Lambda calculator function. The parameters are
 stored in a
```

```
// Go struct that is used to serialize the parameters to a JSON payload. That
 payload is then passed
// to the function.
// The result payload is deserialized to a Go struct that stores the result as
 either an
// int or float32, depending on the kind of operation that was specified.
func (scenario GetStartedFunctionsScenario) InvokeCalculator(ctx context.Context,
 funcName string) {
 wantInvoke := true
 choices := []string{"plus", "minus", "times", "divided-by"}
 for wantInvoke {
  choice := scenario.questioner.AskChoice("Select an arithmetic operation:\n",
 choices)
  x := scenario.questioner.AskInt("Enter a value for x:", demotools.NotEmpty{})
  y := scenario.questioner.AskInt("Enter a value for y:", demotools.NotEmpty{})
  log.Printf("Invoking %v %v %v...", x, choices[choice], y)
  calcParameters := actions.CalculatorParameters{
   Action: choices[choice],
   X:      x,
   Y:      y,
  }
  invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, calcParameters,
 true)
  var payload any
  if choice == 3 { // divide-by results in a float.
   payload = actions.LambdaResultFloat{}
  } else {
   payload = actions.LambdaResultInt{}
  }
  err := json.Unmarshal(invokeOutput.Payload, &payload)
  if err != nil {
   log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
    funcName, err)
  }
  log.Printf("Invoking %v with %v %v %v returned %v.\n", funcName,
   calcParameters.X, calcParameters.Action, calcParameters.Y, payload)
  scenario.questioner.Ask("Press Enter to see the logs from the call.")
  logRes, err := base64.StdEncoding.DecodeString(*invokeOutput.LogResult)
  if err != nil {
   log.Panicf("Couldn't decode log result. Here's why: %v\n", err)
  }
  log.Println(string(logRes))
  wantInvoke = scenario.questioner.AskBool("Do you want to calculate again? (y/n)",
 "y")
```

```
 }
 log.Println(strings.Repeat("-", 88))
}

// ListFunctions lists up to the specified number of functions for your account.
func (scenario GetStartedFunctionsScenario) ListFunctions(ctx context.Context) {
 count := scenario.questioner.AskInt(
  "Let's list functions for your account. How many do you want to see?",
 demotools.NotEmpty{})
 functions := scenario.functionWrapper.ListFunctions(ctx, count)
 log.Printf("Found %v functions:", len(functions))
 for _, function := range functions {
  log.Printf("\t%v", *function.FunctionName)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup removes the IAM and Lambda resources created by the example.
func (scenario GetStartedFunctionsScenario) Cleanup(ctx context.Context, role
 *iamtypes.Role, funcName string) {
 if scenario.questioner.AskBool("Do you want to clean up resources created for this
 example? (y/n)",
  "y") {
  iamClient := iam.NewFromConfig(scenario.sdkConfig)
  policiesOutput, err := iamClient.ListAttachedRolePolicies(ctx,
   &iam.ListAttachedRolePoliciesInput{RoleName: role.RoleName})
  if err != nil {
   log.Panicf("Couldn't get policies attached to role %v. Here's why: %v\n",
    *role.RoleName, err)
  }
  for _, policy := range policiesOutput.AttachedPolicies {
   _, err = iamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
    PolicyArn: policy.PolicyArn, RoleName: role.RoleName,
   })
   if err != nil {
    log.Panicf("Couldn't detach policy %v from role %v. Here's why: %v\n",
     *policy.PolicyArn, *role.RoleName, err)
   }
  }
  _, err = iamClient.DeleteRole(ctx, &iam.DeleteRoleInput{RoleName: role.RoleName})
  if err != nil {
   log.Panicf("Couldn't delete role %v. Here's why: %v\n", *role.RoleName, err)
  }
  log.Printf("Deleted role %v.\n", *role.RoleName)
```

```
  scenario.functionWrapper.DeleteFunction(ctx, funcName)
  log.Printf("Deleted function %v.\n", funcName)
 } else {
  log.Println("Okay. Don't forget to delete the resources when you're done with
 them.")
 }
}

// IScenarioHelper abstracts I/O and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 CreateDeploymentPackage(sourceFile string, destinationFile string) *bytes.Buffer
}

// ScenarioHelper lets the caller specify the path to Lambda handler functions.
type ScenarioHelper struct {
 HandlerPath string
}

// Pause waits for the specified number of seconds.
func (helper *ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// CreateDeploymentPackage creates an AWS Lambda deployment package from a source
 file. The
// deployment package is stored in .zip format in a bytes.Buffer. The buffer can be
// used to pass a []byte to Lambda when creating the function.
// The specified destinationFile is the name to give the file when it's deployed to
 Lambda.
func (helper *ScenarioHelper) CreateDeploymentPackage(sourceFile string,
 destinationFile string) *bytes.Buffer {
 var err error
 buffer := &bytes.Buffer{}
 writer := zip.NewWriter(buffer)
 zFile, err := writer.Create(destinationFile)
 if err != nil {
  log.Panicf("Couldn't create destination archive %v. Here's why: %v\n",
 destinationFile, err)
 }
 sourceBody, err := os.ReadFile(fmt.Sprintf("%v/%v", helper.HandlerPath,
 sourceFile))
```

```
 if err != nil {
  log.Panicf("Couldn't read handler source file %v. Here's why: %v\n",
   sourceFile, err)
 } else {
  _, err = zFile.Write(sourceBody)
  if err != nil {
   log.Panicf("Couldn't write handler %v to zip archive. Here's why: %v\n",
    sourceFile, err)
  }
 }
 err = writer.Close()
 if err != nil {
  log.Panicf("Couldn't close zip writer. Here's why: %v\n", err)
 }
 return buffer
}
```

Crie uma estrutura que encapsule ações individuais do Lambda.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}


// GetFunction gets data about the Lambda function specified by functionName.
```

```
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName string)
 types.State {
 var state types.State
 funcOutput, err := wrapper.LambdaClient.GetFunction(ctx, &lambda.GetFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
 } else {
  state = funcOutput.Configuration.State
 }
 return state
}



// CreateFunction creates a new Lambda function from code contained in the
 zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
 until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
 string, handlerName string,
 iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
  Code:         &types.FunctionCode{ZipFile: zipPackage.Bytes()},
  FunctionName: aws.String(functionName),
  Role:         iamRoleArn,
  Handler:      aws.String(handlerName),
  Publish:      true,
  Runtime:      types.RuntimePython39,
 })
 if err != nil {
  var resConflict *types.ResourceConflictException
  if errors.As(err, &resConflict) {
   log.Printf("Function %v already exists.\n", functionName)
   state = types.StateActive
  } else {
   log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
```

```
  }
 } else {
  waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}



// UpdateFunctionCode updates the code for the Lambda function specified by
 functionName.
// The existing code for the Lambda function is entirely replaced by the code in the
// zipPackage buffer. After the update action is called, a
 lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context, functionName
 string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
 &lambda.UpdateFunctionCodeInput{
  FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
 })
 if err != nil {
  log.Panicf("Couldn't update code for function %v. Here's why: %v\n", functionName,
 err)
 } else {
  waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
```

```
 return state
}



// UpdateFunctionConfiguration updates a map of environment variables configured for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
 functionName string, envVars map[string]string) {
 _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
 &lambda.UpdateFunctionConfigurationInput{
  FunctionName: aws.String(functionName),
  Environment:  &types.Environment{Variables: envVars},
 })
 if err != nil {
  log.Panicf("Couldn't update configuration for %v. Here's why: %v", functionName,
 err)
 }
}



// ListFunctions lists up to maxItems functions for the account. This function uses
 a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
 []types.FunctionConfiguration {
 var functions []types.FunctionConfiguration
 paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
 &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 for paginator.HasMorePages() && len(functions) < maxItems {
  pageOutput, err := paginator.NextPage(ctx)
  if err != nil {
   log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
  }
  functions = append(functions, pageOutput.Functions...)
 }
 return functions
}
```

```
// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
 string) {
 _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
 }
}



// Invoke invokes the Lambda function specified by functionName, passing the
 parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
 tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
 parameters any, getLog bool) *lambda.InvokeOutput {
 logType := types.LogTypeNone
 if getLog {
  logType = types.LogTypeTail
 }
 payload, err := json.Marshal(parameters)
 if err != nil {
  log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
 }
 invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
  FunctionName: aws.String(functionName),
  LogType:      logType,
  Payload:      payload,
 })
 if err != nil {
  log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
 }
 return invokeOutput
}



// IncrementParameters is used to serialize parameters to the increment Lambda
 handler.
type IncrementParameters struct {
```

```
 Action string `json:"action"`
 Number int    `json:"number"`
}

// CalculatorParameters is used to serialize parameters to the calculator Lambda
 handler.
type CalculatorParameters struct {
 Action string `json:"action"`
 X      int    `json:"x"`
 Y      int    `json:"y"`
}

// LambdaResultInt is used to deserialize an int result from a Lambda handler.
type LambdaResultInt struct {
 Result int `json:"result"`
}

// LambdaResultFloat is used to deserialize a float32 result from a Lambda handler.
type LambdaResultFloat struct {
 Result float32 `json:"result"`
}
```

Defina um manipulador do Lambda que aumente um número.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)


def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
 number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
```

```
    result = None
    action = event.get("action")
    if action == "increment":
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Defina um segundo manipulador do Lambda que faça operações aritméticas.

```
import logging
import os


logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
 function.
ACTIONS = {
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}


def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
```

```
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)

    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {"result": result}
    return response
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// CreateFunction creates a new Lambda function from code contained in the
 zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
 until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
 string, handlerName string,
 iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
```

```
 var state types.State
 _, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
  Code:         &types.FunctionCode{ZipFile: zipPackage.Bytes()},
  FunctionName: aws.String(functionName),
  Role:         iamRoleArn,
  Handler:      aws.String(handlerName),
  Publish:      true,
  Runtime:      types.RuntimePython39,
 })
 if err != nil {
  var resConflict *types.ResourceConflictException
  if errors.As(err, &resConflict) {
   log.Printf("Function %v already exists.\n", functionName)
   state = types.StateActive
  } else {
   log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
  }
 } else {
  waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}
```

- Para obter detalhes da API, consulte CreateFunction em Referência AWS SDK para Go da API.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
 string) {
 _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
 }
}
```

- Para obter detalhes da API, consulte DeleteFunction em Referência AWS SDK para Go da API.

GetFunction

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// GetFunction gets data about the Lambda function specified by functionName.
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName string)
 types.State {
 var state types.State
 funcOutput, err := wrapper.LambdaClient.GetFunction(ctx, &lambda.GetFunctionInput{
```

```
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
 } else {
  state = funcOutput.Configuration.State
 }
 return state
}
```

- Para obter detalhes da API, consulte GetFunction em Referência AWS SDK para Go da API.

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
```

```
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// Invoke invokes the Lambda function specified by functionName, passing the
 parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
 tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
 parameters any, getLog bool) *lambda.InvokeOutput {
 logType := types.LogTypeNone
 if getLog {
  logType = types.LogTypeTail
 }
 payload, err := json.Marshal(parameters)
 if err != nil {
  log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
 }
 invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
  FunctionName: aws.String(functionName),
  LogType:      logType,
  Payload:      payload,
 })
 if err != nil {
  log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
 }
 return invokeOutput
}
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para Go .

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// ListFunctions lists up to maxItems functions for the account. This function uses
 a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
 []types.FunctionConfiguration {
 var functions []types.FunctionConfiguration
 paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
 &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 for paginator.HasMorePages() && len(functions) < maxItems {
  pageOutput, err := paginator.NextPage(ctx)
```

```
  if err != nil {
   log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
  }
  functions = append(functions, pageOutput.Functions...)
 }
 return functions
}
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para Go da API.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
```

```
 LambdaClient *lambda.Client
}



// UpdateFunctionCode updates the code for the Lambda function specified by
 functionName.
// The existing code for the Lambda function is entirely replaced by the code in the
// zipPackage buffer. After the update action is called, a
 lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context, functionName
 string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
 &lambda.UpdateFunctionCodeInput{
  FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
 })
 if err != nil {
  log.Panicf("Couldn't update code for function %v. Here's why: %v\n", functionName,
 err)
 } else {
  waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}
```

- Para obter detalhes da API, consulte UpdateFunctionCode em Referência AWS SDK para Go da API.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// UpdateFunctionConfiguration updates a map of environment variables configured for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
 functionName string, envVars map[string]string) {
 _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
 &lambda.UpdateFunctionConfigurationInput{
  FunctionName: aws.String(functionName),
  Environment:  &types.Environment{Variables: envVars},
 })
 if err != nil {
  log.Panicf("Couldn't update configuration for %v. Here's why: %v", functionName,
 err)
```

```
 }
}
```

- Para obter detalhes da API, consulte UpdateFunctionConfiguration em Referência AWS SDK para Go da API.

Cenários

Confirme automaticamente usuários conhecidos com uma função do Lambda

O exemplo de código a seguir mostra como confirmar automaticamente usuários conhecidas do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PreSignUp`.
- Inscreva-se para ser um usuário no Amazon Cognito.
- A função do Lambda verifica uma tabela do DynamoDB e confirma automaticamente os usuários conhecidos.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
```

```
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's email
 matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least eight
 characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
```

```
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string, userName
 string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
```

```
 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `PreSignUp` com uma função do Lambda.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}
```

```
// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the response
 from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
```

```
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match supplied
 UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.\n",
 user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
```

```
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
```

```
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
```

```
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
```

```
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
```

```
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

```
// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

```
// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
```

```
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
```

```
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
```

```
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
```

```
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
```

```
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
```

```
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Migre automaticamente usuários conhecidos com uma função do Lambda

O exemplo de código a seguir mostra como migrar automaticamente usuários conhecidas do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `MigrateUser`.
- Faça login no Amazon Cognito com um nome de usuário e e-mail que não estejam no grupo de usuários.
- A função do Lambda verifica uma tabela do DynamoDB e migra automaticamente os usuários conhecidos para o grupo de usuários.
- Realize um fluxo de senha esquecida para redefinir a senha para o usuário migrado.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
```

```
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function take
 action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
```

```
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username and
 email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user is
 migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This email
 will be used to confirm user migration\n" +
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.", user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }
```

```
 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string, user
 actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user to
 Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
  log.Println("To complete this example and successfully migrate a user to Cognito,
 you must enter an email\n" +
   "you own that can receive a confirmation code.")
  return
 }
 codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
 user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("\nA confirmation code has been sent to %v.", *codeDelivery.Destination)
 code := runner.questioner.Ask("Check your email and enter it here:")

 confirmed := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !confirmed {
  log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
 user.UserName, password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   confirmed = true
  }
 }
```

```
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
 password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
```

```
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `MigrateUser` com uma função do Lambda.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
```

```
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser) (events.CognitoEventUserPoolsMigrateUser,
 error) {
 log.Printf("Received migrate trigger from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n", user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
```

```
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes = map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus = "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)
```

```
// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}
```

```
// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
```

```
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
```

```
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
```

```
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
```

```
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
```

```
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
```

```
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
```

```
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
```

```
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
```

```
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
```

```
    }
    log.Println("Deleted user.")
   }
   triggerList := make([]actions.TriggerInfo, len(resources.triggers))
   for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
  nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Grave dados de atividades personalizados com uma função do Lambda após a autenticação do usuário do Amazon Cognito

O exemplo de código a seguir mostra como gravar dados de atividade personalizados com uma função do Lambda depois da autenticação do usuário do Amazon Cognito.

- Use as funções de administrador para adicionar um usuário a um grupo de usuários.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PostAuthentication`.
- Faça login do novo usuário no Amazon Cognito.
- A função Lambda grava informações personalizadas em CloudWatch Logs e em uma tabela do DynamoDB.
- Obtenha e veja dados personalizados da tabela do DynamoDB e, em seguida, limpe os recursos.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}
```

```
// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
```

```
    } else {
     panic(err)
    }
   } else {
    pwSet = true
   }
  }

  log.Println(strings.Repeat("-", 88))

  return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication trigger
 from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function take
 action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle PostAuthentication
 Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string, userName
 string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
```

```
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the DynamoDB
 table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")
```

```
 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Aborde o acionador `PostAuthentication` com uma função do Lambda.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
```

```
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to the
 logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
```

```
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Crie uma struct que realize tarefas comuns.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
```

```
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
```

```
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Crie uma struct que encapsule ações do Amazon Cognito.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
```

```
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
```

```
    log.Println(*invalidPassword.Message)
   } else {
    log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
   }
  } else {
   confirmed = output.UserConfirmed
  }
  return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
```

```
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
```

```
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
```

```
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Crie uma struct que encapsule ações do DynamoDB.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
```

```
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
```

```
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Crie uma estrutura que envolva as ações do CloudWatch Logs.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
```

```
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Crie uma estrutura que envolva as ações. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
```

```
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}
```

```
// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Go.

```
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"
```

```
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
 var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
 var region string = os.Getenv("AWS_REGION")

 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error: " + err.Error())
 }

 authenticationToken, err := auth.BuildAuthToken(
  context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
 if err != nil {
  panic("failed to create authentication token: " + err.Error())
 }

 dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
  dbUser, authenticationToken, dbEndpoint, dbName,
 )

 db, err := sql.Open("mysql", dsn)
 if err != nil {
  panic(err)
 }

 defer db.Close()

 var sum int
 err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
 if err != nil {
```

```
  panic(err)
 }
 s := fmt.Sprint(sum)
 message := fmt.Sprintf("The selected sum is: %s", s)

 messageBytes, err := json.Marshal(message)
 if err != nil {
  return nil, err
 }

 messageString := string(messageBytes)
 return map[string]interface{}{
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
 if len(kinesisEvent.Records) == 0 {
  log.Printf("empty Kinesis event received")
  return nil
 }

 for _, record := range kinesisEvent.Records {
  log.Printf("processed Kinesis event with EventId: %v", record.EventID)
  recordDataBytes := record.Kinesis.Data
  recordDataText := string(recordDataBytes)
  log.Printf("record data: %v", recordDataText)
  // TODO: Do interesting work based on the new data
 }
 log.Printf("successfully processed %v records", len(kinesisEvent.Records))
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-lambda-go/events"
 "fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string, error)
 {
 if len(event.Records) == 0 {
  return nil, fmt.Errorf("received empty event")
 }

 for _, record := range event.Records {
   LogDynamoDBRecord(record)
 }

 message := fmt.Sprintf("Records processed: %d", len(event.Records))
 return &message, nil
}

func main() {
 lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
 fmt.Println(record.EventID)
 fmt.Println(record.EventName)
 fmt.Printf("%+v\n", record.Change)
```

```
}
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Go.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"

 "github.com/aws/aws-lambda-go/lambda"
)

type Event struct {
 Events []Record `json:"events"`
}

type Record struct {
 Event struct {
  OperationType string `json:"operationType"`
  NS            struct {
   DB   string `json:"db"`
   Coll string `json:"coll"`
  } `json:"ns"`
  FullDocument interface{} `json:"fullDocument"`
```

```
 } `json:"event"`
}

func main() {
 lambda.Start(handler)
}

func handler(ctx context.Context, event Event) (string, error) {
 fmt.Println("Loading function")
 for _, record := range event.Events {
  logDocumentDBEvent(record)
 }

 return "OK", nil
}

func logDocumentDBEvent(record Record) {
 fmt.Printf("Operation type: %s\n", record.Event.OperationType)
 fmt.Printf("db: %s\n", record.Event.NS.DB)
 fmt.Printf("collection: %s\n", record.Event.NS.Coll)
 docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")
 fmt.Printf("Full document: %s\n", string(docBytes))
}
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando Go.

```
package main

import (
 "encoding/base64"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
 for key, records := range event.Records {
  fmt.Println("Key:", key)

  for _, record := range records {
   fmt.Println("Record:", record)

   decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
   message := string(decodedValue)
   fmt.Println("Message:", message)
  }
 }
}

func main() {
 lambda.Start(handler)
}
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
```

```
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
```

```
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
```

```
)

func handler(event events.SQSEvent) error {
 for _, record := range event.Records {
  err := processMessage(record)
  if err != nil {
   return err
  }
 }
 fmt.Println("done")
 return nil
}

func processMessage(record events.SQSMessage) error {
 fmt.Printf("Processed message %s\n", record.Body)
 // TODO: Do interesting work based on the new message
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, record := range kinesisEvent.Records {
  curRecordSequenceNumber := ""

  // Process your record
  if /* Your record processing condition here */ {
   curRecordSequenceNumber = record.Kinesis.SequenceNumber
  }

  // Add a condition to check if the record processing failed
  if curRecordSequenceNumber != "" {
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": curRecordSequenceNumber})
  }
 }

 kinesisBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return kinesisBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
 ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
 BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*BatchResult,
 error) {
 var batchItemFailures []BatchItemFailure
 curRecordSequenceNumber := ""

 for _, record := range event.Records {
  // Process your record
  curRecordSequenceNumber = record.Change.SequenceNumber
```

```
 }

 if curRecordSequenceNumber != "" {
  batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
 curRecordSequenceNumber})
 }

 batchResult := BatchResult{
  BatchItemFailures: batchItemFailures,
 }

 return &batchResult, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
```

```
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent) (map[string]interface{},
 error) {
 batchItemFailures := []map[string]interface{}{}

 for _, message := range sqsEvent.Records {
  if len(message.Body) > 0 {
   // Your message processing condition here
   fmt.Printf("Successfully processed message: %s\n", message.Body)
  } else {
   // Message processing failed
   fmt.Printf("Failed to process message %s\n", message.MessageId)
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": message.MessageId})
  }
 }

 sqsBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return sqsBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Go V2

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Go.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos do Amazon MSK usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon MSK.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando Go.

```
package main

import (
 "encoding/base64"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
 for key, records := range event.Records {
  fmt.Println("Key:", key)

  for _, record := range records {
   fmt.Println("Record:", record)

   decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
   message := string(decodedValue)
   fmt.Println("Message:", message)
  }
 }
}

func main() {
 lambda.Start(handler)
}
```

Exemplos da Central de Parceiros usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Partner Central.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetOpportunity`

O código de exemplo a seguir mostra como usar `GetOpportunity`.

SDK para Go V2

Tenha uma oportunidade.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.GetOpportunity(context.TODO(),
 &partnercentralselling.GetOpportunityInput{
  Identifier: aws.String("O1111111"),
  Catalog:    aws.String("AWS"),
 })
```

```
 if err != nil {
  log.Fatal(err)
 }
 log.Println("printing opportuniy...\n")

 jsonOutput, err := json.MarshalIndent(output, "", "    ")

 fmt.Println(string(jsonOutput))
}
```

- Para obter detalhes da API, consulte GetOpportunity em Referência AWS SDK para Go da API.

`ListOpportunities`

O código de exemplo a seguir mostra como usar `ListOpportunities`.

SDK para Go V2

Liste oportunidades.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }
```

```
 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.ListOpportunities(context.TODO(),
 &partnercentralselling.ListOpportunitiesInput{
  MaxResults: aws.Int32(2),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }

 jsonOutput, err := json.MarshalIndent(output, "", "    ")
 fmt.Println(string(jsonOutput))
}
```

- Para obter detalhes da API, consulte ListOpportunities em Referência AWS SDK para Go da API.

Exemplos do Amazon RDS usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon RDS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos

- Ações
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon RDS

O exemplo de código a seguir mostra como começar a usar o Amazon RDS.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Relational Database Service
 (Amazon RDS)
// client and list up to 20 DB instances in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
```

```
  return
 }
 rdsClient := rds.NewFromConfig(sdkConfig)
 const maxInstances = 20
 fmt.Printf("Let's list up to %v DB instances.\n", maxInstances)
 output, err := rdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{MaxRecords: aws.Int32(maxInstances)})
 if err != nil {
  fmt.Printf("Couldn't list DB instances: %v\n", err)
  return
 }
 if len(output.DBInstances) == 0 {
  fmt.Println("No DB instances found.")
 } else {
  for _, instance := range output.DBInstances {
   fmt.Printf("DB instance %v has database %v.\n", *instance.DBInstanceIdentifier,
    *instance.DBName)
  }
 }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para Go .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de banco de dados e definir os valores dos parâmetros.
- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.
- Exclua a instância e o grupo de parâmetros.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "fmt"
 "log"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/rds/actions"
 "github.com/google/uuid"
)

// GetStartedInstances is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Relation Database Service (Amazon RDS) to do the following:
//
//  1. Create a custom DB parameter group and set parameter values.
//  2. Create a DB instance that is configured to use the parameter group. The DB
 instance
//     also contains a database.
//  3. Take a snapshot of the DB instance.
//  4. Delete the DB instance and parameter group.
type GetStartedInstances struct {
 sdkConfig  aws.Config
 instances  actions.DbInstances
 questioner demotools.IQuestioner
 helper     IScenarioHelper
```

```
 isTestRun  bool
}

// NewGetStartedInstances constructs a GetStartedInstances instance from a
 configuration.
// It uses the specified config to get an Amazon RDS
// client and create wrappers for the actions used in the scenario.
func NewGetStartedInstances(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) GetStartedInstances {
 rdsClient := rds.NewFromConfig(sdkConfig)
 return GetStartedInstances{
  sdkConfig:  sdkConfig,
  instances:  actions.DbInstances{RdsClient: rdsClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedInstances) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 instanceName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Relational Database Service (Amazon RDS) DB
 Instance demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine, parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 instance := scenario.CreateInstance(ctx, instanceName, dbEngine, dbName,
 parameterGroup)
 scenario.DisplayConnection(instance)
 scenario.CreateSnapshot(ctx, instance)
 scenario.Cleanup(ctx, instance, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
```

```
}

// CreateParameterGroup shows how to get available engine versions for a specified
// database engine and create a DB parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedInstances) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBParameterGroup {

 log.Printf("Checking for an existing DB parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?\n",
 families)
  log.Println("Creating a DB parameter group.")
  _, err = scenario.instances.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
```

```
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup
}

// SetUserParameters shows how to get the parameters contained in a custom parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedInstances) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName, "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.instances.UpdateParameters(ctx, parameterGroupName, updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("To get a list of parameters that you set previously, specify a source
 of 'user'.")
 userParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
 "user")
```

```
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateInstance shows how to create a DB instance that contains a database of a
// specified type. The database is also configured to use a custom DB parameter
 group.
func (scenario GetStartedInstances) CreateInstance(ctx context.Context, instanceName
 string, dbEngine string,
 dbName string, parameterGroup *types.DBParameterGroup) *types.DBInstance {

 log.Println("Checking for an existing DB instance.")
 instance, err := scenario.instances.GetInstance(ctx, instanceName)
 if err != nil {
  panic(err)
 }
 if instance == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator username for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.AskPassword(
   "Enter a password for the administrator (at least 8 characters): ", 7)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine,
   *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  engineIndex := scenario.questioner.AskChoice(
   "The available engines for your parameter group are:\n", engineChoices)
  engineSelection := engineVersions[engineIndex]
  instOpts, err := scenario.instances.GetOrderableInstances(ctx,
 *engineSelection.Engine,
   *engineSelection.EngineVersion)
  if err != nil {
   panic(err)
```

```
 }
 optSet := map[string]struct{}{}
 for _, opt := range instOpts {
  if strings.Contains(*opt.DBInstanceClass, "micro") {
   optSet[*opt.DBInstanceClass] = struct{}{}
  }
 }
 var optChoices []string
 for opt := range optSet {
  optChoices = append(optChoices, opt)
 }
 sort.Strings(optChoices)
 optIndex := scenario.questioner.AskChoice(
  "The available micro DB instance classes for your database engine are:\n",
optChoices)
 storageType := "standard"
 allocatedStorage := int32(5)
 log.Printf("Creating a DB instance named %v and database %v.\n"+
  "The DB instance is configured to use your custom parameter group %v,\n"+
  "selected engine %v,\n"+
  "selected DB instance class %v,"+
  "and %v GiB of %v storage.\n"+
  "This typically takes several minutes.",
  instanceName, dbName, *parameterGroup.DBParameterGroupName,
*engineSelection.EngineVersion,
  optChoices[optIndex], allocatedStorage, storageType)
 instance, err = scenario.instances.CreateInstance(
  ctx, instanceName, dbName, *engineSelection.Engine,
*engineSelection.EngineVersion,
  *parameterGroup.DBParameterGroupName, optChoices[optIndex], storageType,
  allocatedStorage, adminUsername, adminPassword)
 if err != nil {
  panic(err)
 }
 for *instance.DBInstanceStatus != "available" {
  scenario.helper.Pause(30)
  instance, err = scenario.instances.GetInstance(ctx, instanceName)
  if err != nil {
   panic(err)
  }
 }
 log.Println("Instance created and available.")
}
log.Println("Instance data:")
```

```
 log.Printf("\tDBInstanceIdentifier: %v\n", *instance.DBInstanceIdentifier)
 log.Printf("\tARN: %v\n", *instance.DBInstanceArn)
 log.Printf("\tStatus: %v\n", *instance.DBInstanceStatus)
 log.Printf("\tEngine: %v\n", *instance.Engine)
 log.Printf("\tEngine version: %v\n", *instance.EngineVersion)
 log.Println(strings.Repeat("-", 88))
 return instance
}

// DisplayConnection displays connection information about a DB instance and tips
// on how to connect to it.
func (scenario GetStartedInstances) DisplayConnection(instance *types.DBInstance) {
 log.Println(
  "You can now connect to your database by using your favorite MySQL client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n" +
   "that is running in the same VPC as your DB instance. Pass the endpoint,\n" +
   "port, and administrator username to 'mysql'. Then, enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *instance.Endpoint.Address, instance.Endpoint.Port, *instance.MasterUsername)
 log.Println("For more information, see the User Guide for RDS:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB instance snapshot and wait until it's
 available.
func (scenario GetStartedInstances) CreateSnapshot(ctx context.Context, instance
 *types.DBInstance) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB instance (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", *instance.DBInstanceIdentifier,
 scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.\n",
 snapshotId)
  snapshot, err := scenario.instances.CreateSnapshot(ctx,
 *instance.DBInstanceIdentifier, snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.instances.GetSnapshot(ctx, snapshotId)
```

```
    if err != nil {
     panic(err)
    }
   }
   log.Println("Snapshot data:")
   log.Printf("\tDBSnapshotIdentifier: %v\n", *snapshot.DBSnapshotIdentifier)
   log.Printf("\tARN: %v\n", *snapshot.DBSnapshotArn)
   log.Printf("\tStatus: %v\n", *snapshot.Status)
   log.Printf("\tEngine: %v\n", *snapshot.Engine)
   log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
   log.Printf("\tDBInstanceIdentifier: %v\n", *snapshot.DBInstanceIdentifier)
   log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
   log.Println(strings.Repeat("-", 88))
  }
 }

// Cleanup shows how to clean up a DB instance and DB parameter group.
// Before the DB parameter group can be deleted, all associated DB instances must
 first be deleted.
func (scenario GetStartedInstances) Cleanup(
 ctx context.Context, instance *types.DBInstance, parameterGroup
 *types.DBParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance and parameter group (y/n)? ", "y")
 {
  log.Printf("Deleting database instance %v.\n", *instance.DBInstanceIdentifier)
  err := scenario.instances.DeleteInstance(ctx, *instance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance to delete. This typically takes several minutes.")
  for instance != nil {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(ctx,
 *instance.DBInstanceIdentifier)
   if err != nil {
    panic(err)
   }
  }
  log.Printf("Deleting parameter group %v.", *parameterGroup.DBParameterGroupName)
  err = scenario.instances.DeleteParameterGroup(ctx,
 *parameterGroup.DBParameterGroupName)
```

```
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Amazon RDS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}
```

```
// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}



// CreateParameterGroup creates a DB parameter group that is based on the specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
```

```
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
```

```
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName string)
 (*types.DBSnapshot, error) {
```

```
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string, dbInstanceClass
 string,
 storageType string, allocatedStorage int32, adminName string, adminPassword string)
 (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

```
// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

```
// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions, output.OrderableDBInstanceOptions...)
```

```
  }
 }
 return instanceOptions, err
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Ações

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string, dbInstanceClass
 string,
 storageType string, allocatedStorage int32, adminName string, adminPassword string)
 (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
```

```
 }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Go .

`CreateDBParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBParameterGroup`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateParameterGroup creates a DB parameter group that is based on the specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBParameterGroup, error) {
```

```
 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}
```

- Para obter detalhes da API, consulte CreateDBParameterGroup em Referência AWS SDK para Go da API.

`CreateDBSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBSnapshot`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
```

```
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para Go .

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Go .

`DeleteDBParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBParameterGroup`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Para obter detalhes da API, consulte DeleteDBParameterGroup em Referência AWS SDK para Go da API.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
```

```
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions em Referência AWS SDK para Go da API.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}
```

```
// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para Go .

`DescribeDBParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBParameterGroups`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}
```

- Para obter detalhes da API, consulte DescribeDBParameterGroups em Referência AWS SDK para Go da API.

`DescribeDBParameters`

O código de exemplo a seguir mostra como usar `DescribeDBParameters`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
```

```
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para Go .

`DescribeDBSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBSnapshots`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
```

```
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName string)
 (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência da API AWS SDK para Go .

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions, output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptions em Referência AWS SDK para Go da API.

ModifyDBParameterGroup

O código de exemplo a seguir mostra como usar `ModifyDBParameterGroup`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
```

```
}
```

- Para obter detalhes da API, consulte ModifyDBParameterGroup em Referência AWS SDK para Go da API.

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Go.

```
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"

 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
```

```
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
 var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
 var region string = os.Getenv("AWS_REGION")

 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error: " + err.Error())
 }

 authenticationToken, err := auth.BuildAuthToken(
  context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
 if err != nil {
  panic("failed to create authentication token: " + err.Error())
 }

 dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
  dbUser, authenticationToken, dbEndpoint, dbName,
 )

 db, err := sql.Open("mysql", dsn)
 if err != nil {
  panic(err)
 }

 defer db.Close()

 var sum int
 err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
 if err != nil {
  panic(err)
 }
 s := fmt.Sprint(sum)
```

```
 message := fmt.Sprintf("The selected sum is: %s", s)

 messageBytes, err := json.Marshal(message)
 if err != nil {
  return nil, err
 }

 messageString := string(messageBytes)
 return map[string]interface{}{
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Exemplos do Amazon Redshift usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon Redshift.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Redshift

O exemplo de código a seguir mostra como começar a usar o Amazon Redshift.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
)

// main uses the AWS SDK for Go V2 to create a Redshift client
// and list up to 10 clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 redshiftClient := redshift.NewFromConfig(sdkConfig)
 count := 20
 fmt.Printf("Let's list up to %v clusters for your account.\n", count)
```

```
 result, err := redshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  MaxRecords: aws.Int32(int32(count)),
 })
 if err != nil {
  fmt.Printf("Couldn't list clusters for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Clusters) == 0 {
  fmt.Println("You don't have any clusters!")
  return
 }
 for _, cluster := range result.Clusters {
  fmt.Printf("\t%v : %v\n", *cluster.ClusterIdentifier, *cluster.ClusterStatus)
 }
}
```

- Para obter detalhes da API, consulte DescribeClusters em Referência AWS SDK para Go da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um cluster do Redshift.
- Listar bancos de dados no cluster.
- Criar uma tabela chamada Filmes.
- Preencher a tabela Filmes.
- Consultar a tabela Filmes por ano.
- Modificar o cluster do Redshift.
- Excluir o cluster do Amazon Redshift.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package scenarios

import (
 "context"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 redshift_types "github.com/aws/aws-sdk-go-v2/service/redshift/types"
 redshiftdata_types "github.com/aws/aws-sdk-go-v2/service/redshiftdata/types"
 "github.com/aws/aws-sdk-go-v2/service/secretsmanager"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/redshift/actions"

 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshiftdata"
)

// IScenarioHelper abstracts input and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
```

```
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// RedshiftBasicsScenario separates the steps of this scenario into individual
 functions so that
// they are simpler to read and understand.
type RedshiftBasicsScenario struct {
 sdkConfig         aws.Config
 helper            IScenarioHelper
 questioner        demotools.IQuestioner
 pauser            demotools.IPausable
 filesystem        demotools.IFileSystem
 redshiftActor     *actions.RedshiftActions
 redshiftDataActor *actions.RedshiftDataActions
 secretsmanager    *SecretsManager
}

// SecretsManager is used to retrieve username and password information from a
 secure service.
type SecretsManager struct {
 SecretsManagerClient *secretsmanager.Client
}

// RedshiftBasics constructs a new Redshift Basics runner.
func RedshiftBasics(sdkConfig aws.Config, questioner demotools.IQuestioner, pauser
 demotools.IPausable, filesystem demotools.IFileSystem, helper IScenarioHelper)
 RedshiftBasicsScenario {
 scenario := RedshiftBasicsScenario{
  sdkConfig:         sdkConfig,
  helper:            helper,
  questioner:        questioner,
  pauser:            pauser,
  filesystem:        filesystem,
  secretsmanager:    &SecretsManager{SecretsManagerClient:
 secretsmanager.NewFromConfig(sdkConfig)},
  redshiftActor:     &actions.RedshiftActions{RedshiftClient:
 redshift.NewFromConfig(sdkConfig)},
  redshiftDataActor: &actions.RedshiftDataActions{RedshiftDataClient:
 redshiftdata.NewFromConfig(sdkConfig)},
```

```
 }
 return scenario
}


// Movie makes it easier to use Movie objects given in json format.
type Movie struct {
 ID    int    `json:"id"`
 Title string `json:"title"`
 Year  int    `json:"year"`
}


// User makes it easier to get the User data back from SecretsManager and use it
 later.
type User struct {
 Username string `json:"userName"`
 Password string `json:"userPassword"`
}

// Run runs the RedshiftBasics interactive example that shows you how to use Amazon
// Redshift and how to interact with its common endpoints.
//
// 0. Retrieve username and password information to access Redshift.
// 1. Create a cluster.
// 2. Wait for the cluster to become available.
// 3. List the available databases in the region.
// 4. Create a table named "Movies" in the "dev" database.
// 5. Populate the movies table from the "movies.json" file.
// 6. Query the movies table by year.
// 7. Modify the cluster's maintenance window.
// 8. Optionally clean up all resources created during this demo.
//
// This example creates an Amazon Redshift service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func (runner *RedshiftBasicsScenario) Run(ctx context.Context) {

 user := User{}
 secretId := "s3express/basics/secrets"
```

```
clusterId := "demo-cluster-1"
maintenanceWindow := "wed:07:30-wed:08:00"
databaseName := "dev"
tableName := "Movies"
fileName := "Movies.json"
nodeType := "ra3.xlplus"
clusterType := "single-node"

defer func() {
 if r := recover(); r != nil {
  log.Println("Something went wrong with the demo.")
  _, isMock := runner.questioner.(*demotools.MockQuestioner)
  if isMock || runner.questioner.AskBool("Do you want to see the full error message
(y/n)?", "y") {
   log.Println(r)
  }
  runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username,
runner.questioner)
 }
}()

// Retrieve the userName and userPassword from SecretsManager
output, err := runner.secretsmanager.SecretsManagerClient.GetSecretValue(ctx,
&secretsmanager.GetSecretValueInput{
 SecretId: aws.String(secretId),
})
if err != nil {
 log.Printf("There was a problem getting the secret value: %s", err)
 log.Printf("Please make sure to create a secret named 's3express/basics/secrets'
with keys of 'userName' and 'userPassword'.")
 panic(err)
}

err = json.Unmarshal([]byte(*output.SecretString), &user)
if err != nil {
 log.Printf("There was a problem parsing the secret value from JSON: %s", err)
 panic(err)
}

// Create the Redshift cluster
_, err = runner.redshiftActor.CreateCluster(ctx, clusterId, user.Username,
user.Password, nodeType, clusterType, true)
if err != nil {
 var clusterAlreadyExistsFault *redshift_types.ClusterAlreadyExistsFault
```

```
 if errors.As(err, &clusterAlreadyExistsFault) {
  log.Println("Cluster already exists. Continuing.")
 } else {
  log.Println("Error creating cluster.")
  panic(err)
 }
}

// Wait for the cluster to become available
waiter := redshift.NewClusterAvailableWaiter(runner.redshiftActor.RedshiftClient)
err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
 ClusterIdentifier: aws.String(clusterId),
}, 5*time.Minute)
if err != nil {
 log.Println("An error occurred waiting for the cluster.")
 panic(err)
}

// Get some info about the cluster
describeOutput, err := runner.redshiftActor.DescribeClusters(ctx, clusterId)
if err != nil {
 log.Println("Something went wrong trying to get information about the cluster.")
 panic(err)
}
log.Println("Here's some information about the cluster.")
log.Printf("The cluster's status is %s", *describeOutput.Clusters[0].ClusterStatus)
log.Printf("The cluster was created at %s",
*describeOutput.Clusters[0].ClusterCreateTime)

// List databases
log.Println("List databases in", clusterId)
runner.questioner.Ask("Press Enter to continue...")
err = runner.redshiftDataActor.ListDatabases(ctx, clusterId, databaseName,
user.Username)
if err != nil {
 log.Printf("Failed to list databases: %v\n", err)
 panic(err)
}

// Create the "Movies" table
log.Println("Now you will create a table named " + tableName + ".")
runner.questioner.Ask("Press Enter to continue...")
err = nil
```

```
 result, err := runner.redshiftDataActor.CreateTable(ctx, clusterId, databaseName,
 tableName, user.Username, runner.pauser, []string{"title VARCHAR(256)", "year
 INT"})
 if err != nil {
  log.Printf("Failed to create table: %v\n", err)
  panic(err)
 }

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }
 query := actions.RedshiftQuery{
  Context: ctx,
  Input:   describeInput,
  Result:  result,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute query: %v\n", err)
  panic(err)
 }
 log.Printf("Successfully executed query\n")

 // Populate the "Movies" table
 runner.PopulateMoviesTable(ctx, clusterId, databaseName, tableName, user.Username,
 fileName)

 // Query the "Movies" table by year
 log.Println("Query the Movies table by year.")
 year := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 2012, 2014),
  demotools.InIntRange{Lower: 2012, Upper: 2014})
 runner.QueryMoviesByYear(ctx, clusterId, databaseName, tableName, user.Username,
 year)

 // Modify the cluster's maintenance window
 runner.redshiftActor.ModifyCluster(ctx, clusterId, maintenanceWindow)

 // Delete the Redshift cluster if confirmed
 runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username,
 runner.questioner)

 log.Println("Thanks for watching!")
}
```

```
// cleanUpResources asks the user if they would like to delete each resource created
 during the scenario, from most
// impactful to least impactful. If any choice to delete is made, further deletion
 attempts are skipped.
func (runner *RedshiftBasicsScenario) cleanUpResources(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string,
 questioner demotools.IQuestioner) {
 deleted := false
 var err error = nil
 if questioner.AskBool("Do you want to delete the entire cluster? This will clean up
 all resources. (y/n)", "y") {
  deleted, err = runner.redshiftActor.DeleteCluster(ctx, clusterId)
  if err != nil {
   log.Printf("Error deleting cluster: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete the dev table? This will
 clean up all inserted records but keep your cluster intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteTable(ctx, clusterId, databaseName,
 tableName, userName)
  if err != nil {
   log.Printf("Error deleting movies table: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete all rows in the Movies
 table? This will clean up all inserted records but keep your cluster and table
 intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteDataRows(ctx, clusterId,
 databaseName, tableName, userName, runner.pauser)
  if err != nil {
   log.Printf("Error deleting data rows: %v", err)
  }
 }
 if !deleted {
  log.Print("Please manually delete any unwanted resources.")
 }
}


// loadMoviesFromJSON takes the <fileName> file and populates a slice of Movie
 objects.
func (runner *RedshiftBasicsScenario) loadMoviesFromJSON(fileName string, filesystem
 demotools.IFileSystem) ([]Movie, error) {
```

```
 file, err := filesystem.OpenFile("../../resources/sample_files/" + fileName)
 if err != nil {
  return nil, err
 }
 defer filesystem.CloseFile(file)

 var movies []Movie
 err = json.NewDecoder(file).Decode(&movies)
 if err != nil {
  return nil, err
 }

 return movies, nil
}



// PopulateMoviesTable reads data from the <fileName> file and inserts records into
 the "Movies" table.
func (runner *RedshiftBasicsScenario) PopulateMoviesTable(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string, fileName
 string) {
 log.Println("Populate the " + tableName + " table using the " + fileName + "
 file.")
 numRecords := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 10, 100),
  demotools.InIntRange{Lower: 10, Upper: 100})

 movies, err := runner.loadMoviesFromJSON(fileName, runner.filesystem)
 if err != nil {
  log.Printf("Failed to load movies from JSON: %v\n", err)
  panic(err)
 }

 var sqlStatements []string

 for i, movie := range movies {
  if i >= numRecords {
   break
  }

  sqlStatement := fmt.Sprintf(`INSERT INTO %s (title, year) VALUES ('%s', %d);`,
   tableName,
```

```
    strings.Replace(movie.Title, "'", "''", -1), // Double any single quotes to
  escape them
    movie.Year)

   sqlStatements = append(sqlStatements, sqlStatement)
  }

  input := &redshiftdata.BatchExecuteStatementInput{
   ClusterIdentifier: aws.String(clusterId),
   Database:          aws.String(databaseName),
   DbUser:            aws.String(userName),
   Sqls:              sqlStatements,
  }

  result, err := runner.redshiftDataActor.ExecuteBatchStatement(ctx, *input)
  if err != nil {
   log.Printf("Failed to execute batch statement: %v\n", err)
   panic(err)
  }

  describeInput := redshiftdata.DescribeStatementInput{
   Id: result.Id,
  }

  query := actions.RedshiftQuery{
   Context: ctx,
   Result:  result,
   Input:   describeInput,
  }
  err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
  if err != nil {
   log.Printf("Failed to execute batch insert query: %v\n", err)
   return
  }
  log.Printf("Successfully executed batch statement\n")

  log.Printf("%d records were added to the Movies table.\n", numRecords)
 }



 // QueryMoviesByYear retrieves only movies from the "Movies" table which match the
  given year.
```

```
func (runner *RedshiftBasicsScenario) QueryMoviesByYear(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string, year int)
 {

 sqlStatement := fmt.Sprintf(`SELECT title FROM %s WHERE year = %d;`, tableName,
 year)

 input := &redshiftdata.ExecuteStatementInput{
  ClusterIdentifier: aws.String(clusterId),
  Database:          aws.String(databaseName),
  DbUser:            aws.String(userName),
  Sql:               aws.String(sqlStatement),
 }

 result, err := runner.redshiftDataActor.ExecuteStatement(ctx, *input)
 if err != nil {
  log.Printf("Failed to query movies: %v\n", err)
  panic(err)
 }

 log.Println("The identifier of the statement is ", *result.Id)

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }

 query := actions.RedshiftQuery{
  Context: ctx,
  Input:   describeInput,
  Result:  result,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute query: %v\n", err)
  panic(err)
 }
 log.Printf("Successfully executed query\n")

 getResultOutput, err := runner.redshiftDataActor.GetStatementResult(ctx,
 *result.Id)
 if err != nil {
  log.Printf("Failed to query movies: %v\n", err)
  panic(err)
 }
```

```
 for _, row := range getResultOutput.Records {
  for _, col := range row {
   title, ok := col.(*redshiftdata_types.FieldMemberStringValue)
   if !ok {
    log.Println("Failed to parse the field")
   } else {
    log.Printf("The Movie title field is %s\n", title.Value)
   }
  }
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// CreateCluster sends a request to create a cluster with the given clusterId using
 the provided credentials.
func (actor RedshiftActions) CreateCluster(ctx context.Context, clusterId string,
 userName string, userPassword string, nodeType string, clusterType string,
 publiclyAccessible bool) (*redshift.CreateClusterOutput, error) {
 // Create a new Redshift cluster
 input := &redshift.CreateClusterInput{
  ClusterIdentifier:  aws.String(clusterId),
  MasterUserPassword: aws.String(userPassword),
  MasterUsername:     aws.String(userName),
  NodeType:           aws.String(nodeType),
  ClusterType:        aws.String(clusterType),
  PubliclyAccessible: aws.Bool(publiclyAccessible),
 }
 var opErr *types.ClusterAlreadyExistsFault
 output, err := actor.RedshiftClient.CreateCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster already exists")
  return nil, nil
 } else if err != nil {
  log.Printf("Failed to create Redshift cluster: %v\n", err)
  return nil, err
```

```
 }

 log.Printf("Created cluster %s\n", *output.Cluster.ClusterIdentifier)
 return output, nil
}
```

- Para obter detalhes da API, consulte CreateCluster em Referência AWS SDK para Go da API.

`DeleteCluster`

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}
```

```
// DeleteCluster deletes the given cluster.
func (actor RedshiftActions) DeleteCluster(ctx context.Context, clusterId string)
 (bool, error) {
 input := redshift.DeleteClusterInput{
  ClusterIdentifier:        aws.String(clusterId),
  SkipFinalClusterSnapshot: aws.Bool(true),
 }
 _, err := actor.RedshiftClient.DeleteCluster(ctx, &input)
 var opErr *types.ClusterNotFoundFault
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster was not found. Where could it be?")
  return false, err
 } else if err != nil {
  log.Printf("Failed to delete Redshift cluster: %v\n", err)
  return false, err
 }
 waiter := redshift.NewClusterDeletedWaiter(actor.RedshiftClient)
 err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 }, 5*time.Minute)
 if err != nil {
  log.Printf("Wait time exceeded for deleting cluster, continuing: %v\n", err)
 }
 log.Printf("The cluster %s was deleted\n", clusterId)
 return true, nil
}
```

- Para obter detalhes da API, consulte DeleteCluster em Referência AWS SDK para Go da API.

`DescribeClusters`

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// DescribeClusters returns information about the given cluster.
func (actor RedshiftActions) DescribeClusters(ctx context.Context, clusterId string)
 (*redshift.DescribeClustersOutput, error) {
 input, err := actor.RedshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 })
 var opErr *types.AccessToClusterDeniedFault
 if errors.As(err, &opErr) {
  println("Access to cluster denied.")
  panic(err)
 } else if err != nil {
  println("Failed to describe Redshift clusters.")
  return nil, err
 }
 return input, nil
}
```

- Para obter detalhes da API, consulte DescribeClusters em Referência AWS SDK para Go da API.

`ModifyCluster`

O código de exemplo a seguir mostra como usar `ModifyCluster`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// ModifyCluster sets the preferred maintenance window for the given cluster.
func (actor RedshiftActions) ModifyCluster(ctx context.Context, clusterId string,
 maintenanceWindow string) *redshift.ModifyClusterOutput {
 // Modify the cluster's maintenance window
 input := &redshift.ModifyClusterInput{
  ClusterIdentifier:          aws.String(clusterId),
  PreferredMaintenanceWindow: aws.String(maintenanceWindow),
 }

 var opErr *types.InvalidClusterStateFault
```

```
 output, err := actor.RedshiftClient.ModifyCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster is in an invalid state.")
  panic(err)
 } else if err != nil {
  log.Printf("Failed to modify Redshift cluster: %v\n", err)
  panic(err)
 }

 log.Printf("The cluster was successfully modified and now has %s as the maintenance
 window\n", *output.Cluster.PreferredMaintenanceWindow)
 return output
}
```

- Para obter detalhes da API, consulte ModifyCluster em Referência AWS SDK para Go da API.

Exemplos do Amazon S3 usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "errors"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
```

```
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  var ae smithy.APIError
  if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
   fmt.Println("You don't have permission to list buckets for this account.")
  } else {
   fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  }
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para Go da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.

- Exclir os objetos do bucket e o bucket.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina um struct que envolva as ações de bucket e objeto usadas pelo cenário.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
```

```
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket, error)
 {
 var err error
 var output *s3.ListBucketsOutput
 var buckets []types.Bucket
 bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
 &s3.ListBucketsInput{})
 for bucketPaginator.HasMorePages() {
  output, err = bucketPaginator.NextPage(ctx)
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
   } else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
   }
   break
  } else {
   buckets = append(buckets, output.Buckets...)
  }
 }
 return buckets, err
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
 (bool, error) {
 _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
```

```
    log.Printf("Either you don't have access to bucket %v or another error occurred.
 "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
 string) error {
 _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", name)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", name)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
  }
 }
 return err
}
```

```
// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
     "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
     "or the multipart upload API (5TB max).", bucketName)
   } else {
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
     fileName, bucketName, objectKey, err)
   }
  } else {
   err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
   }
  }
 }
 return err
}



// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName string,
 objectKey string, largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
```

```
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
 objectKey, bucketName)
```

```
    err = noKey
   } else {
    log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName, objectKey,
  err)
   }
   return err
  }
  defer result.Body.Close()
  file, err := os.Create(fileName)
  if err != nil {
   log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
   return err
  }
  defer file.Close()
  body, err := io.ReadAll(result.Body)
  if err != nil {
   log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey, err)
  }
  _, err = file.Write(body)
  return err
}



// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
```

```
}



// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
 objectKey string, folderName string) error {
 objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(bucketName),
  CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
  Key:        aws.String(objectDest),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, bucketName)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectDest)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
  }
 }
 return err
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
 destinationBucket string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
```

```
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, sourceBucket)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }
 return err
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
 ([]types.Object, error) {
 var err error
 var output *s3.ListObjectsV2Output
 input := &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 }
 var objects []types.Object
 objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
 for objectPaginator.HasMorePages() {
  output, err = objectPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
   break
  } else {
   objects = append(objects, output.Contents...)
  }
 }
 return objects, err
}
```

```
// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
 objectKeys []string) error {
 var objectIds []types.ObjectIdentifier
 for _, key := range objectKeys {
  objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
 }
 output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
  Bucket: aws.String(bucketName),
  Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
 })
 if err != nil || len(output.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucketName)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
  } else if len(output.Errors) > 0 {
   for _, outErr := range output.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *output.Errors[0].Message)
  }
 } else {
  for _, delObjs := range output.Deleted {
   err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
 time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to be deleted.\n",
 *delObjs.Key)
   } else {
    log.Printf("Deleted %s.\n", *delObjs.Key)
   }
  }
 }
 return err
}
```

```
// DeleteBucket deletes a bucket. The bucket must be empty or an error is returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
 error {
 _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucketName)
   err = noBucket
  } else {
   log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
  }
 } else {
  err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucketName)
  } else {
   log.Printf("Deleted %s.\n", bucketName)
  }
 }
 return err
}
```

Execute um cenário interativo que mostre como trabalhar com buckets e objetos do S3.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
```

```
)

// RunGetStartedScenario is an interactive example that shows you how to use Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
 objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 getting started demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 count := 10
 log.Printf("Let's list up to %v buckets for your account:", count)
 buckets, err := bucketBasics.ListBuckets(ctx)
```

```
if err != nil {
 panic(err)
}
if len(buckets) == 0 {
 log.Println("You don't have any buckets!")
} else {
 if count > len(buckets) {
  count = len(buckets)
 }
 for _, bucket := range buckets[:count] {
  log.Printf("\t%v\n", *bucket.Name)
 }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
 demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
```

```
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
 panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
 panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
 panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
 objKeys = append(objKeys, *object.Key)
 log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
 "contents? (y/n)", "y") {
 log.Println("Deleting objects.")
 err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
 if err != nil {
  panic(err)
 }
 log.Println("Deleting bucket.")
 err = bucketBasics.DeleteBucket(ctx, bucketName)
 if err != nil {
  panic(err)
 }
```

```
  log.Printf("Deleting downloaded file %v.\n", downloadFileName)
  err = os.Remove(downloadFileName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
 destinationBucket string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
```

```
  Key:        aws.String(objectKey),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, sourceBucket)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }
 return err
}
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK para Go da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar um bucket com a configuração padrão.

```
import (
 "bytes"
```

```
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
 string) error {
 _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", name)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", name)
   err = exists
  }
```

```
 } else {
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
  }
 }
 return err
}
```

Crie um bucket com bloqueio de objetos e espere que ele exista.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
```

```
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}
```

- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK para Go da API.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
 error {
 _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
```

```
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucketName)
   err = noBucket
  } else {
   log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
  }
 } else {
  err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucketName)
  } else {
   log.Printf("Deleted %s.\n", bucketName)
  }
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteBucket em Referência AWS SDK para Go da API.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
```

```
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key string,
 versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
```

```
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
 time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.\n",
 key, bucket)
  } else {
   deleted = true
  }
 }
 return deleted, err
}
```

- Para obter detalhes da API, consulte DeleteObject em Referência AWS SDK para Go da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
```

```
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
```

```
    }
   } else if len(delOut.Errors) > 0 {
    for _, outErr := range delOut.Errors {
     log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
    }
    err = fmt.Errorf("%s", *delOut.Errors[0].Message)
   }
  } else {
   for _, delObjs := range delOut.Deleted {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
     ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
  time.Minute)
    if err != nil {
     log.Printf("Failed attempt to wait for object %s to be deleted.\n",
  *delObjs.Key)
    } else {
     log.Printf("Deleted %s.\n", *delObjs.Key)
    }
   }
  }
  return err
}
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK para Go da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
```

```
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
 objectKey, bucketName)
   err = noKey
  } else {
   log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName, objectKey,
 err)
  }
  return err
 }
```

```
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey, err)
 }
 _, err = file.Write(body)
 return err
}
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK para Go da API.

`GetObjectLegalHold`

O código de exemplo a seguir mostra como usar `GetObjectLegalHold`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
```

```
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }
```

```
 return status, err
}
```

- Para obter detalhes da API, consulte GetObjectLegalHold em Referência AWS SDK para Go da API.

`GetObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `GetObjectLockConfiguration`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}
```

- Para obter detalhes da API, consulte GetObjectLockConfiguration em Referência AWS SDK para Go da API.

`GetObjectRetention`

O código de exemplo a seguir mostra como usar `GetObjectRetention`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectRetention retrieves the object retention configuration for an S3 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
```

```
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}
```

- Para obter detalhes da API, consulte GetObjectRetention em Referência AWS SDK para Go da API.

`HeadBucket`

O código de exemplo a seguir mostra como usar `HeadBucket`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
```

```
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
 (bool, error) {
 _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error occurred.
 "+
```

```
      "Here's what happened: %v\n", bucketName, err)
    }
   }
  } else {
   log.Printf("Bucket %v exists and you already own it.", bucketName)
  }

  return exists, err
}
```

- Para obter detalhes da API, consulte HeadBucket em Referência AWS SDK para Go da API.

`ListBuckets`

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
```

```
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket, error)
 {
 var err error
 var output *s3.ListBucketsOutput
 var buckets []types.Bucket
 bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
 &s3.ListBucketsInput{})
 for bucketPaginator.HasMorePages() {
  output, err = bucketPaginator.NextPage(ctx)
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
   } else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
   }
   break
  } else {
   buckets = append(buckets, output.Buckets...)
  }
 }
 return buckets, err
}
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para Go da API.

`ListObjectVersions`

O código de exemplo a seguir mostra como usar `ListObjectVersions`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
```

```
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

- Para obter detalhes da API, consulte ListObjectVersions em Referência AWS SDK para Go da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
```

```
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
 ([]types.Object, error) {
 var err error
 var output *s3.ListObjectsV2Output
 input := &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 }
 var objects []types.Object
 objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
 for objectPaginator.HasMorePages() {
  output, err = objectPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
   break
  } else {
   objects = append(objects, output.Contents...)
  }
 }
```

```
 return objects, err
}
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK para Go API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloque um objeto em um bucket usando a API de baixo nível.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
```

```
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
     "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
     "or the multipart upload API (5TB max).", bucketName)
   } else {
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
     fileName, bucketName, objectKey, err)
   }
  } else {
   err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
   }
  }
 }
 return err
}
```

Faça upload de um objeto em um bucket usando um gerenciador de transferência.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key string,
 contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
```

```
    err = noBucket
   }
  } else {
   err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
    Bucket: aws.String(bucket),
    Key:    aws.String(key),
   }, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key, bucket)
   } else {
    outKey = *output.Key
   }
  }
  return outKey, err
}
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK para Go da API.

`PutObjectLegalHold`

O código de exemplo a seguir mostra como usar `PutObjectLegalHold`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- Para obter detalhes da API, consulte PutObjectLegalHold em Referência AWS SDK para Go da API.

`PutObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `PutObjectLockConfiguration`.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de Bloqueio de Objetos de um bucket.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
```

```
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket string)
 error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

Defina o período de retenção padrão de um bucket.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ModifyDefaultBucketRetention modifies the default retention period of an existing
 bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
```

```
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

- Para obter detalhes da API, consulte PutObjectLockConfiguration em Referência AWS SDK para Go da API.

`PutObjectRetention`

O código de exemplo a seguir mostra como usar `PutObjectRetention`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)
```

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- Para obter detalhes da API, consulte PutObjectRetention em Referência AWS SDK para Go da API.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que envolvam ações de pré-assinatura do S3.

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
 actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
 S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
 client.
type Presigner struct {
 PresignClient *s3.PresignClient
}



// GetObject makes a presigned request that can be used to get an object from a
 bucket.
```

```
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
 ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignGetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// PutObject makes a presigned request that can be used to put an object in a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
 ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignPutObject(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// DeleteObject makes a presigned request that can be used to delete an object from
 a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
 objectKey string) (*v4.PresignedHTTPRequest, error) {
```

```
 request, err := presigner.PresignClient.PresignDeleteObject(ctx,
 &s3.DeleteObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to delete object %v. Here's why: %v
\n", objectKey, err)
 }
 return request, err
}



func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName string,
 objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error) {
 request, err := presigner.PresignClient.PresignPostObject(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(options *s3.PresignPostOptions) {
  options.Expires = time.Duration(lifetimeSecs) * time.Second
 })
 if err != nil {
  log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v\n",
 bucketName, objectKey, err)
 }
 return request, nil
}
```

Execute um exemplo interativo que gera e usa URLs pré-assinados para fazer upload, fazer download e excluir um objeto do S3.

```
import (
 "bytes"
 "context"
 "io"
 "log"
 "mime/multipart"
 "net/http"
```

```
 "os"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)



// RunPresigningScenario is an interactive example that shows you how to get
 presigned
// HTTP requests that you can use to move data into and out of Amazon Simple Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local file
 to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the object
 to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
 mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner, httpRequester IHttpRequester) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
```

```
    if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
     log.Println(r)
    }
   }
  }()

  log.Println(strings.Repeat("-", 88))
  log.Println("Welcome to the Amazon S3 presigning demo.")
  log.Println(strings.Repeat("-", 88))

  s3Client := s3.NewFromConfig(sdkConfig)
  bucketBasics := actions.BucketBasics{S3Client: s3Client}
  presignClient := s3.NewPresignClient(s3Client)
  presigner := actions.Presigner{PresignClient: presignClient}

  bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
   "you own or one you want to create:", demotools.NotEmpty{})
  bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
  if err != nil {
   panic(err)
  }
  if !bucketExists {
   err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
   if err != nil {
    panic(err)
   } else {
    log.Println("Bucket created.")
   }
  }
  log.Println(strings.Repeat("-", 88))

  log.Printf("Let's presign a request to upload a file to your bucket.")
  uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
   demotools.NotEmpty{})
  uploadKey := questioner.Ask("What would you like to name the uploaded object?",
   demotools.NotEmpty{})
  uploadFile, err := os.Open(uploadFilename)
  if err != nil {
   panic(err)
  }
  defer uploadFile.Close()
  presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
  if err != nil {
```

```
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
 presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
 panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
 uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
 presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
 uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
```

```
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName, uploadKey,
60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
 panic(err)
}
defer uploadFile.Close()
multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
if err != nil {
 panic(err)
}
log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
multiPartResponse.StatusCode)

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
 presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
```

```
  uploadKey, delResponse.StatusCode)
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina um wrapper de solicitação HTTP usado pelo exemplo para fazer solicitações HTTP.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
 during
// unit testing.
type IHttpRequester interface {
 Get(url string) (resp *http.Response, err error)
 Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
 Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
 error)
 Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
 scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
 return http.Get(url)
}
func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
 *http.Response, err error) {
 postRequest, err := http.NewRequest("POST", url, body)
 if err != nil {
  return nil, err
 }
 postRequest.Header.Set("Content-Type", contentType)
 return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
 (resp *http.Response, err error) {
 putRequest, err := http.NewRequest("PUT", url, body)
```

```
 if err != nil {
  return nil, err
 }
 putRequest.ContentLength = contentLength
 return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error) {
 delRequest, err := http.NewRequest("DELETE", url, nil)
 if err != nil {
  return nil, err
 }
 return http.DefaultClient.Do(delRequest)
}
```

Bloquear objetos do Amazon S3

O exemplo de código a seguir mostra como trabalhar com os recursos de bloqueio de objetos do S3.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do Amazon S3.

```
import (
 "context"
 "fmt"
 "log"
 "strings"

 "s3_object_lock/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
```

```
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
 questioner demotools.IQuestioner
 resources  Resources
 s3Actions  *actions.S3Actions
 sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ObjectLockScenario {
 scenario := ObjectLockScenario{
  questioner: questioner,
  resources:  Resources{},
  s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
  sdkConfig:  sdkConfig,
 }
 scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
 scenario.resources.init(scenario.s3Actions, questioner)
 return scenario
}

type nameLocked struct {
 name   string
 locked bool
}

var createInfo = []nameLocked{
 {"standard-bucket", false},
 {"lock-bucket", true},
 {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
 log.Println("Let's create some S3 buckets to use for this workflow.")
 success := false
 for !success {
  prefix := scenario.questioner.Ask(
```

```
    "This example creates three buckets. Enter a prefix to name your buckets
 (remember bucket names must be globally unique):")

  for _, info := range createInfo {
   log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
   bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx, fmt.Sprintf("%s.
%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
   if err != nil {
    switch err.(type) {
    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
     log.Printf("Couldn't create bucket %s.\n", bucketName)
    default:
     panic(err)
    }
    break
   }
   scenario.resources.demoBuckets[info.name] = &DemoBucket{
    name:       bucketName,
    objectKeys: []string{},
   }
   log.Printf("Created bucket %s.\n", bucketName)
  }

  if len(scenario.resources.demoBuckets) < len(createInfo) {
   scenario.resources.deleteBuckets(ctx)
  } else {
   success = true
  }
 }

 log.Println("S3 buckets created.")
 log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking.")
 scenario.questioner.Ask("Press Enter to continue.")

 var err error
 bucket := scenario.resources.demoBuckets["retention-bucket"]
 err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
```

```
  case *types.NoSuchBucket:
   log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Object locking enabled on bucket %s.", bucket.name)
 }

 log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking with a default
 retention period.")

 bucket := scenario.resources.demoBuckets["retention-bucket"]
 retentionPeriod := scenario.questioner.AskInt("Enter the default retention period
 in days: ")
 err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
 bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
 types.ObjectLockRetentionModeGovernance)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't configure a default retention period on bucket %s.\n",
 bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Default retention policy set on bucket %s with %d day retention
 period.", bucket.name, retentionPeriod)
  bucket.retentionEnabled = true
 }

 log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
 log.Println("Uploading test objects to S3 buckets.")
```

```
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  for i := 0; i < 2; i++ {
   key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
 fmt.Sprintf("example-%d", i),
    fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
   if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
     log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
   }
  }
 }

 scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
 log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx context.Context)
 {
 log.Println("Now let's set object lock configurations on individual objects.")

 buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
 scenario.resources.demoBuckets["retention-bucket"]}
 for _, bucket := range buckets {
  for index, objKey := range bucket.objectKeys {
   switch index {
   case 0:
    if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold to
 %s in %s (y/n)? ", objKey, bucket.name), "y") {
     err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
 types.ObjectLockLegalHoldStatusOn)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set legal hold on %s.\n", objKey)
      default:
```

```
        panic(err)
       }
      } else {
       log.Printf("Legal hold set on %s.\n", objKey)
      }
     }
    case 1:
     q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to %s
 in %s?\n"+
      "Reminder: Only a user with the s3:BypassGovernanceRetention permission is able
 to delete this object\n"+
      "or its bucket until the retention period has expired. (y/n) ", objKey,
 bucket.name)
     if scenario.questioner.AskBool(q, "y") {
      err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
 types.ObjectLockRetentionModeGovernance, 1)
      if err != nil {
       switch err.(type) {
       case *types.NoSuchKey:
        log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
 bucket.name)
       default:
        panic(err)
       }
      } else {
       log.Printf("Retention period set to 1 for %s.", objKey)
       bucket.retentionEnabled = true
      }
     }
    }
   }
 }
 log.Println(strings.Repeat("-", 88))
}

const (
 ListAll = iota
 DeleteObject
 DeleteRetentionObject
 OverwriteObject
 ViewRetention
 ViewLegalHold
 Finish
)
```

```
// InteractWithObjects allows the user to interact with the objects and test the
 object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
 log.Println("Now you can interact with the objects to explore the object lock
 configurations.")
 interactiveChoices := []string{
  "List all objects and buckets.",
  "Attempt to delete an object.",
  "Attempt to delete an object with retention period bypass.",
  "Attempt to overwrite a file.",
  "View the retention settings for an object.",
  "View the legal hold settings for an object.",
  "Finish the workflow."}

 choice := ListAll
 for choice != Finish {
  objList := scenario.GetAllObjects(ctx)
  objChoices := scenario.makeObjectChoiceList(objList)
  choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
 interactiveChoices)
  switch choice {
  case ListAll:
   log.Println("The current objects in the example buckets are:")
   for _, objChoice := range objChoices {
    log.Println("\t", objChoice)
   }
  case DeleteObject, DeleteRetentionObject:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
 delete:\n", objChoices)
   obj := objList[objChoice]
   deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
 obj.versionId, choice == DeleteRetentionObject)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Println("Nothing to delete.")
    default:
     panic(err)
    }
   } else if deleted {
    log.Printf("Object %s deleted.\n", obj.key)
   }
  case OverwriteObject:
```

```
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
  obj := objList[objChoice]
  _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Println("Couldn't upload to nonexistent bucket.")
   default:
    panic(err)
   }
  } else {
   log.Printf("Uploaded new content to object %s.\n", obj.key)
  }
 case ViewRetention:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
  obj := objList[objChoice]
  retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket, obj.key)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Can't get retention configuration for %s.\n", obj.key)
   default:
    panic(err)
   }
  } else if retention != nil {
   log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
  } else {
   log.Printf("Object %s does not have object retention configured.\n", obj.key)
  }
 case ViewLegalHold:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
  obj := objList[objChoice]
  legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket, obj.key,
obj.versionId)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
   default:
```

```
      panic(err)
     }
    } else if legalHold != nil {
     log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
    } else {
     log.Printf("Object %s does not have legal hold configured.", obj.key)
    }
   case Finish:
    log.Println("Let's clean up.")
   }
   log.Println(strings.Repeat("-", 88))
  }
}

type BucketKeyVersionId struct {
 bucket    string
 key       string
 versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns them
 in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
 []BucketKeyVersionId {
 var objectList []BucketKeyVersionId
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't get object versions for %s.\n", bucket.name)
   default:
    panic(err)
   }
  } else {
   for _, version := range versions {
    objectList = append(objectList,
     BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
 *version.VersionId})
   }
  }
 }
 return objectList
```

```
}

// makeObjectChoiceList makes the object version list into a list of strings that
 are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
 []BucketKeyVersionId) []string {
 choices := make([]string, len(bucketObjects))
 for i := 0; i < len(bucketObjects); i++ {
  choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
   bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
 }
 return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := scenario.questioner.(*demotools.MockQuestioner)
   if isMock || scenario.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   scenario.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
 log.Println(strings.Repeat("-", 88))

 scenario.CreateBuckets(ctx)
 scenario.EnableLockOnBucket(ctx)
 scenario.SetDefaultRetentionPolicy(ctx)
 scenario.UploadTestObjects(ctx)
 scenario.SetObjectLockConfigurations(ctx)
 scenario.InteractWithObjects(ctx)

 scenario.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
```

```
 log.Println(strings.Repeat("-", 88))
}
```

Defina um struct que envolva as ações do S3 usadas neste exemplo.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
```

```
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
```

```
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }
```

```
 return lockConfig, err
}



// GetObjectRetention retrieves the object retention configuration for an S3 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
```

```
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// ModifyDefaultBucketRetention modifies the default retention period of an existing
 bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
```

```
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket string)
 error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
```

```
  }
 }

 return err
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key string,
 contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
```

```
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key, bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
```

```
  }
 }
 return versions, err
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key string,
 versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
```

```
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
 time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.\n",
 key, bucket)
  } else {
   deleted = true
  }
 }
 return deleted, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
```

```
    err = fmt.Errorf("%s", *delOut.Errors[0].Message)
   }
  } else {
   for _, delObjs := range delOut.Deleted {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
     ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
  time.Minute)
    if err != nil {
     log.Printf("Failed attempt to wait for object %s to be deleted.\n",
  *delObjs.Key)
    } else {
     log.Printf("Deleted %s.\n", *delObjs.Key)
    }
   }
  }
  return err
}
```

Limpe recursos.

```
import (
 "context"
 "log"
 "s3_object_lock/actions"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
 name             string
 retentionEnabled bool
 objectKeys       []string
}
```

```
// Resources keeps track of AWS resources created during the ObjectLockScenario and
 handles
// cleanup when the scenario finishes.
type Resources struct {
 demoBuckets map[string]*DemoBucket

 s3Actions  *actions.S3Actions
 questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
 demotools.IQuestioner) {
 resources.s3Actions = s3Actions
 resources.questioner = questioner
 resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources " +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if !wantDelete {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
  return
 }

 log.Println("Removing objects from S3 buckets and deleting buckets...")
 resources.deleteBuckets(ctx)
 //resources.deleteRetentionObjects(resources.retentionBucket,
 resources.retentionObjects)

 log.Println("Cleanup complete.")
}
```

```
// deleteBuckets empties and then deletes all buckets created during the
 ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  resources.deleteObjects(ctx, bucket)
  _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucket.name),
  })
  if err != nil {
   panic(err)
  }
 }
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucket.name)
  } else {
   log.Printf("Deleted %s.\n", bucket.name)
  }
 }
 resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket *DemoBucket) {
 lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx, bucket.name)
 if err != nil {
  panic(err)
 }
 versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("No objects to get from %s.\n", bucket.name)
  default:
   panic(err)
  }
 }
 delObjects := make([]types.ObjectIdentifier, len(versions))
 for i, version := range versions {
```

```
 if lockConfig != nil && lockConfig.ObjectLockEnabled ==
types.ObjectLockEnabledEnabled {
  status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
   default:
    panic(err)
   }
  } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
   err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
    default:
     panic(err)
    }
   }
  }
 }
 delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
bucket.retentionEnabled)
if err != nil {
 switch err.(type) {
 case *types.NoSuchBucket:
  log.Println("Nothing to delete.")
 default:
  panic(err)
 }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Fazer upload ou download de arquivos grandes

O exemplo de código a seguir mostra como fazer upload ou download de arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte Carregar um objeto usando carregamento fracionado.

SDK para Go V2

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que usam gerenciadores de upload e download para dividir os dados em partes e transferi-los simultaneamente.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName string,
 objectKey string, largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
```

```
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}



// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

Execute um cenário interativo que mostre como usar os gerenciadores de upload e download em contexto.

```
import (
 "context"
 "crypto/rand"
 "log"
 "strings"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 large object demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}
```

```
bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
 demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
_, _ = rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName, largeKey)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
 "contents? (y/n)", "y") {
```

```
  log.Println("Deleting object.")
  err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
  if err != nil {
   panic(err)
  }
  log.Println("Deleting bucket.")
  err = bucketBasics.DeleteBucket(ctx, bucketName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Exemplos do Amazon SNS usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon SNS

O exemplo de código a seguir mostra como começar a usar o Amazon SNS.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
```

```
 }
}
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para Go da API.

Ações

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}
```

```
// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Para obter detalhes da API, consulte CreateTopic em Referência AWS SDK para Go da API.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteTopic em Referência AWS SDK para Go da API.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
```

```
    break
   } else {
    topics = append(topics, output.Topics...)
   }
  }
  if len(topics) == 0 {
   fmt.Println("You don't have any topics!")
  } else {
   for _, topic := range topics {
    fmt.Printf("\t%v\n", *topic.TopicArn)
   }
  }
}
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para Go da API.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)
```

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent to
 all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered according
 to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string) error
 {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
 }
 return err
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para Go .

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscreva uma fila em um tópico com filtros opcionais.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
```

```
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para Go .

Cenários

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.

- Pesquise as filas para ver as mensagens recebidas.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
 "strings"
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ScenarioRunner struct {
```

```
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string, bool,
 bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or standard.
\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.\n"
 +
   "Deduplication IDs are either set in the message or are automatically generated
\n" +
   "from content using a hash function. If a message is successfully published to\n"
 +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted\n"
 +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
```

```
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN) \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
 isFifoTopic bool) (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS queue.
 ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string, topicArn
 string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
```

```
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
 "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
 if ordinal == "first" {
  log.Println("Subscriptions to a FIFO topic can have filters.\n" +
   "If you add a filter to this subscription, then only the filtered messages\n" +
   "will be received in the queue.\n" +
   "For information about message filtering, see\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
   "For this example, you can filter messages by a \"tone\" attribute.")
 }

 wantFiltering := runner.questioner.AskBool(
  fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
   "from the %v topic? (y/n) ", queueName, topicName), "y")
 if wantFiltering {
  log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

  var toneSelections []string
  askAboutTones := true
  for askAboutTones {
   toneIndex := runner.questioner.AskChoice(
    "Enter the number of the tone you want to filter by:\n", ToneChoices)
   toneSelections = append(toneSelections, ToneChoices[toneIndex])
   askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
  }
  log.Printf("Your subscription will be filtered to only pass the following tones:
%v\n", toneSelections)
  filterPolicy = map[string][]string{TONE_KEY: toneSelections}
 }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)
if err != nil {
 panic(err)
}
log.Printf("The queue %v is now subscribed to the topic %v with the subscription
ARN %v.\n",
```

```
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn string,
 isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group ID.
\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))
```

```
  publishMore = runner.questioner.AskBool("Do you want to publish another messsage?
 (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages), queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
   if err != nil {
    panic(err)
   }
```

```
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to the
\n"+
  "topic. You can select from several options for configuring the topic and the\n"+
  "subscriptions for the queues. You can then post to the topic and see the results
\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
```

```
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.\n",
 queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources created
 for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina um struct que envolva as ações do Amazon SNS usadas neste exemplo.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
```

```
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
```

```
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent to
 all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered according
 to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string) error
 {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
```

```
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
 }
 return err
}
```

Defina um struct que envolva as ações do Amazon SQS usadas neste exemplo.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
```

```
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
 error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to
 an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to
 the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
```

```
    Principal: map[string]string{"Service": "sns.amazonaws.com"},
    Resource:  aws.String(queueArn),
    Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string
```

```
// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages
 from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 }
 return err
}
```

```
// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Limpe recursos.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management Console
\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
```

```
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}
```

```
func main() {
 lambda.Start(handler)
}
```

Exemplos do Amazon SQS usando o SDK para Go V2

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para Go V2 com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon SQS

O exemplo de código a seguir mostra como começar a usar o Amazon SQS.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 sqsClient := sqs.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the queues for your account.")
 var queueUrls []string
 paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get queues. Here's why: %v\n", err)
   break
```

```
  } else {
   queueUrls = append(queueUrls, output.QueueUrls...)
  }
 }
 if len(queueUrls) == 0 {
  fmt.Println("You don't have any queues!")
 } else {
  for _, queueUrl := range queueUrls {
   fmt.Printf("\t%v\n", queueUrl)
  }
 }
}
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para Go da API.

Ações

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
```

```
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}
```

- Para obter detalhes da API, consulte CreateQueue em Referência AWS SDK para Go da API.

DeleteMessageBatch

O código de exemplo a seguir mostra como usar `DeleteMessageBatch`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages
 from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
```

```
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteMessageBatch em Referência AWS SDK para Go da API.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}
```

```
// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- Para obter detalhes da API, consulte DeleteQueue em Referência AWS SDK para Go da API.

`GetQueueAttributes`

O código de exemplo a seguir mostra como usar `GetQueueAttributes`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)
```

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
 error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}
```

- Para obter detalhes da API, consulte GetQueueAttributes em Referência AWS SDK para Go da API.

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 sqsClient := sqs.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the queues for your account.")
 var queueUrls []string
 paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get queues. Here's why: %v\n", err)
   break
```

```
  } else {
   queueUrls = append(queueUrls, output.QueueUrls...)
  }
 }
 if len(queueUrls) == 0 {
  fmt.Println("You don't have any queues!")
 } else {
  for _, queueUrl := range queueUrls {
   fmt.Printf("\t%v\n", queueUrl)
  }
 }
}
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para Go da API.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)
```

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  messages = result.Messages
 }
 return messages, err
}
```

- Para obter detalhes da API, consulte ReceiveMessage em Referência AWS SDK para Go da API.

`SetQueueAttributes`

O código de exemplo a seguir mostra como usar `SetQueueAttributes`.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to
 an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to
 the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
```

```
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string
```

- Para obter detalhes da API, consulte SetQueueAttributes em Referência AWS SDK para Go da API.

Cenários

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
 "strings"
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}
```

```
// ScenarioRunner separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string, bool,
 bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or standard.
\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.\n"
 +
   "Deduplication IDs are either set in the message or are automatically generated
\n" +
   "from content using a hash function. If a message is successfully published to\n"
 +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted\n"
 +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to\n"+
   "the topic name.", FIFO_SUFFIX)
 }
```

```
 topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN) \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
 isFifoTopic bool) (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS queue.
 ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string, topicArn
 string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)
```

```
err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
if err != nil {
 panic(err)
}
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
 "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
 if ordinal == "first" {
  log.Println("Subscriptions to a FIFO topic can have filters.\n" +
   "If you add a filter to this subscription, then only the filtered messages\n" +
   "will be received in the queue.\n" +
   "For information about message filtering, see\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
   "For this example, you can filter messages by a \"tone\" attribute.")
 }

 wantFiltering := runner.questioner.AskBool(
  fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
   "from the %v topic? (y/n) ", queueName, topicName), "y")
 if wantFiltering {
  log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

  var toneSelections []string
  askAboutTones := true
  for askAboutTones {
   toneIndex := runner.questioner.AskChoice(
    "Enter the number of the tone you want to filter by:\n", ToneChoices)
   toneSelections = append(toneSelections, ToneChoices[toneIndex])
   askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
  }
  log.Printf("Your subscription will be filtered to only pass the following tones:
%v\n", toneSelections)
  filterPolicy = map[string][]string{TONE_KEY: toneSelections}
 }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)
if err != nil {
```

```
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn string,
 isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group ID.
\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
```

```
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another messsage?
 (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages), queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
```

```
    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
     panic(err)
    }
   }
  }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to the
\n"+
  "topic. You can select from several options for configuring the topic and the\n"+
  "subscriptions for the queues. You can then post to the topic and see the results
\n"+
  "in the queues.\n", queueCount)
```

```
log.Println(strings.Repeat("-", 88))

runner := ScenarioRunner{
 questioner: questioner,
 snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
 sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
}
resources.snsActor = runner.snsActor
resources.sqsActor = runner.sqsActor

topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
runner.CreateTopic(ctx)
resources.topicArn = topicArn
log.Println(strings.Repeat("-", 88))

log.Printf("Now you will create %v SQS queues and subscribe them to the topic.\n",
queueCount)
ordinals := []string{"first", "next"}
usingFilters := false
for _, ordinal := range ordinals {
 queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
 resources.queueUrls = append(resources.queueUrls, queueUrl)

 _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl, topicName,
topicArn, ordinal, isFifoTopic)
 usingFilters = usingFilters || filtering
}

log.Println(strings.Repeat("-", 88))
runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
usingFilters)
log.Println(strings.Repeat("-", 88))
runner.PollForMessages(ctx, resources.queueUrls)

log.Println(strings.Repeat("-", 88))

wantCleanup := questioner.AskBool("Do you want to remove all AWS resources created
for this scenario? (y/n) ", "y")
if wantCleanup {
 log.Println("Cleaning up resources...")
 resources.Cleanup(ctx)
}
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Defina um struct que envolva as ações do Amazon SNS usadas neste exemplo.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
```

```
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
```

```
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent to
 all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered according
 to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string) error
 {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
```

```
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
 }
 return err
}
```

Defina um struct que envolva as ações do Amazon SQS usadas neste exemplo.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
```

```
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
 error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to
 an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to
 the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
```

```
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}
```

```
// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages
 from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 }
 return err
```

```
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Limpe recursos.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management Console
\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
```

```
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
 for _, record := range event.Records {
  err := processMessage(record)
  if err != nil {
   return err
  }
 }
 fmt.Println("done")
 return nil
}

func processMessage(record events.SQSMessage) error {
 fmt.Printf("Processed message %s\n", record.Body)
 // TODO: Do interesting work based on the new message
```

```
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Go V2

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent) (map[string]interface{},
 error) {
 batchItemFailures := []map[string]interface{}{}

 for _, message := range sqsEvent.Records {
  if len(message.Body) > 0 {
```

```
    // Your message processing condition here
    fmt.Printf("Successfully processed message: %s\n", message.Body)
   } else {
    // Message processing failed
    fmt.Printf("Failed to process message %s\n", message.MessageId)
    batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": message.MessageId})
   }
  }

  sqsBatchResponse := map[string]interface{}{
   "batchItemFailures": batchItemFailures,
  }
  return sqsBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Exemplos de código para SDK for JavaScript (v2)

Os exemplos de código a seguir mostram como usar o AWS SDK para JavaScript (v2) com AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para JavaScript (v2) — Saiba mais sobre como usar JavaScript com. AWS

- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- CloudWatch exemplos de uso do SDK para JavaScript (v2)
- CloudWatch Exemplos de eventos usando o SDK for JavaScript (v2)
- CloudWatch Exemplos de registros usando o SDK for JavaScript (v2)
- Exemplos do DynamoDB usando o SDK for JavaScript (v2)
- AWS Entity Resolution exemplos de uso do SDK para JavaScript (v2)
- EventBridge exemplos de uso do SDK para JavaScript (v2)
- Exemplos do Amazon Glacier usando o SDK para JavaScript (v2)
- Exemplos de IAM usando SDK para JavaScript (v2)
- Exemplos de Lambda usando SDK para JavaScript (v2)
- Exemplos do Amazon Pinpoint usando o SDK para JavaScript (v2)
- Exemplos da API de SMS e voz do Amazon Pinpoint usando o SDK para JavaScript (v2)
- Exemplos do Amazon SNS usando o SDK para JavaScript (v2)
- Exemplos do Amazon SQS usando o SDK para JavaScript (v2)
- AWS STS exemplos de uso do SDK para JavaScript (v2)

CloudWatch exemplos de uso do SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteAlarms`

O código de exemplo a seguir mostra como usar `DeleteAlarms`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmNames: ["Web_Server_CPU_Utilization"],
};

cw.deleteAlarms(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

- Para obter detalhes da API, consulte DeleteAlarms em Referência AWS SDK para JavaScript da API.

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.describeAlarms({ StateValue: "INSUFFICIENT_DATA" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeAlarmsForMetric em Referência AWS SDK para JavaScript da API.

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.disableAlarmActions(
  { AlarmNames: ["Web_Server_CPU_Utilization"] },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DisableAlarmActions em Referência AWS SDK para JavaScript da API.

`EnableAlarmActions`

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"],
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
```

```
    console.log("Alarm action added", data);
    var paramsEnableAlarmAction = {
      AlarmNames: [params.AlarmName],
    };
    cw.enableAlarmActions(paramsEnableAlarmAction, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Alarm action enabled", data);
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte EnableAlarmActions em Referência AWS SDK para JavaScript da API.

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  Dimensions: [
```

```
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};

cw.listMetrics(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListMetrics em Referência AWS SDK para JavaScript da API.

`PutMetricAlarm`

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });
```

```
var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutMetricAlarm em Referência AWS SDK para JavaScript da API.

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
    {
      MetricName: "PAGES_VISITED",
      Dimensions: [
        {
          Name: "UNIQUE_PAGES",
          Value: "URLS",
        },
      ],
      Unit: "None",
      Value: 1.0,
    },
  ],
  Namespace: "SITE/TRAFFIC",
};

cw.putMetricData(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", JSON.stringify(data));
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutMetricData em Referência AWS SDK para JavaScript da API.

CloudWatch Exemplos de eventos usando o SDK for JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com CloudWatch Eventos.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};

cwevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK para JavaScript da API.

`PutRule`

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
```

```
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

cwevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutRule em Referência AWS SDK para JavaScript da API.

`PutTargets`

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
```

```
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myCloudWatchEventsTarget",
    },
  ],
};

cwevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutTargets em Referência AWS SDK para JavaScript da API.

CloudWatch Exemplos de registros usando o SDK for JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com CloudWatch registros.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteSubscriptionFilter`

O código de exemplo a seguir mostra como usar `DeleteSubscriptionFilter`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  filterName: "FILTER",
  logGroupName: "LOG_GROUP",
};

cwl.deleteSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteSubscriptionFilter em Referência AWS SDK para JavaScript da API.

`DescribeSubscriptionFilters`

O código de exemplo a seguir mostra como usar `DescribeSubscriptionFilters`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  logGroupName: "GROUP_NAME",
  limit: 5,
};

cwl.describeSubscriptionFilters(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeSubscriptionFilters em Referência AWS SDK para JavaScript da API.

`PutSubscriptionFilter`

O código de exemplo a seguir mostra como usar `PutSubscriptionFilter`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  destinationArn: "LAMBDA_FUNCTION_ARN",
  filterName: "FILTER_NAME",
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP",
};

cwl.putSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutSubscriptionFilter em Referência AWS SDK para JavaScript da API.

Exemplos do DynamoDB usando o SDK for JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com o DynamoDB.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`BatchGetItem`

O código de exemplo a seguir mostra como usar `BatchGetItem`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  RequestItems: {
    TABLE_NAME: {
```

```
      Keys: [
        { KEY_NAME: { N: "KEY_VALUE_1" } },
        { KEY_NAME: { N: "KEY_VALUE_2" } },
        { KEY_NAME: { N: "KEY_VALUE_3" } },
      ],
      ProjectionExpression: "KEY_NAME, ATTRIBUTE",
    },
  },
};

ddb.batchGetItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    data.Responses.TABLE_NAME.forEach(function (element, index, array) {
      console.log(element);
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte BatchGetItem em Referência AWS SDK para JavaScript da API.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  RequestItems: {
    TABLE_NAME: [
      {
        PutRequest: {
          Item: {
            KEY: { N: "KEY_VALUE" },
            ATTRIBUTE_1: { S: "ATTRIBUTE_1_VALUE" },
            ATTRIBUTE_2: { N: "ATTRIBUTE_2_VALUE" },
          },
        },
      },
      {
        PutRequest: {
          Item: {
            KEY: { N: "KEY_VALUE" },
            ATTRIBUTE_1: { S: "ATTRIBUTE_1_VALUE" },
            ATTRIBUTE_2: { N: "ATTRIBUTE_2_VALUE" },
          },
        },
      },
    ],
  },
};

ddb.batchWriteItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte BatchWriteItem em Referência AWS SDK para JavaScript da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  AttributeDefinitions: [
    {
      AttributeName: "CUSTOMER_ID",
      AttributeType: "N",
    },
    {
      AttributeName: "CUSTOMER_NAME",
      AttributeType: "S",
    },
  ],
  KeySchema: [
    {
      AttributeName: "CUSTOMER_ID",
      KeyType: "HASH",
    },
    {
      AttributeName: "CUSTOMER_NAME",
      KeyType: "RANGE",
    },
  ],
  ProvisionedThroughput: {
    ReadCapacityUnits: 1,
```

```
    WriteCapacityUnits: 1,
  },
  TableName: "CUSTOMER_LIST",
  StreamSpecification: {
    StreamEnabled: false,
  },
};

// Call DynamoDB to create the table
ddb.createTable(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Table Created", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para JavaScript da API.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Exclua um item de uma tabela.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Key: {
    KEY_NAME: { N: "VALUE" },
  },
};

// Call DynamoDB to delete the item from the table
ddb.deleteItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Exclua um item de uma tabela usando o cliente de documento do DynamoDB.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  Key: {
    HASH_KEY: VALUE,
  },
  TableName: "TABLE",
};

docClient.delete(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
```

```
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteItem em Referência AWS SDK para JavaScript da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: process.argv[2],
};

// Call DynamoDB to delete the specified table
ddb.deleteTable(params, function (err, data) {
  if (err && err.code === "ResourceNotFoundException") {
    console.log("Error: Table not found");
  } else if (err && err.code === "ResourceInUseException") {
    console.log("Error: Table in use");
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para JavaScript da API.

`DescribeTable`

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: process.argv[2],
};

// Call DynamoDB to retrieve the selected table descriptions
ddb.describeTable(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Table.KeySchema);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

- Para obter detalhes da API, consulte DescribeTable em Referência AWS SDK para JavaScript da API.

GetItem

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Obtenha um item de uma tabela.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Key: {
    KEY_NAME: { N: "001" },
  },
  ProjectionExpression: "ATTRIBUTE_NAME",
};

// Call DynamoDB to read the item from the table
ddb.getItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

Obtenha um item de uma tabela usando o cliente de documento do DynamoDB.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  TableName: "EPISODES_TABLE",
  Key: { KEY_NAME: VALUE },
};

docClient.get(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetItem em Referência AWS SDK para JavaScript da API.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

// Call DynamoDB to retrieve the list of tables
ddb.listTables({ Limit: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err.code);
  } else {
    console.log("Table names are ", data.TableNames);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para JavaScript da API.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloque um item em uma tabela.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "CUSTOMER_LIST",
  Item: {
    CUSTOMER_ID: { N: "001" },
    CUSTOMER_NAME: { S: "Richard Roe" },
  },
};

// Call DynamoDB to add the item to the table
ddb.putItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Coloque um item em uma tabela usando o cliente de documento do DynamoDB.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Item: {
    HASHKEY: VALUE,
    ATTRIBUTE_1: "STRING_VALUE",
    ATTRIBUTE_2: VALUE_2,
  },
};

docClient.put(params, function (err, data) {
  if (err) {
```

```
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutItem em Referência AWS SDK para JavaScript da API.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  ExpressionAttributeValues: {
    ":s": 2,
    ":e": 9,
    ":topic": "PHRASE",
  },
  KeyConditionExpression: "Season = :s and Episode > :e",
  FilterExpression: "contains (Subtitle, :topic)",
  TableName: "EPISODES_TABLE",
};
```

```
docClient.query(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Items);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js.
var AWS = require("aws-sdk");
// Set the AWS Region.
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object.
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

const params = {
  // Specify which items in the results are returned.
  FilterExpression: "Subtitle = :topic AND Season = :s AND Episode = :e",
  // Define the expression attribute value, which are substitutes for the values you
 want to compare.
  ExpressionAttributeValues: {
    ":topic": { S: "SubTitle2" },
    ":s": { N: 1 },
```

```
    ":e": { N: 2 },
  },
  // Set the projection expression, which are the attributes that you want.
  ProjectionExpression: "Season, Episode, Title, Subtitle",
  TableName: "EPISODES_TABLE",
};

ddb.scan(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
    data.Items.forEach(function (element, index, array) {
      console.log(
        "printing",
        element.Title.S + " (" + element.Subtitle.S + ")"
      );
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Scan na Referência da API AWS SDK para JavaScript .

Cenários

Invocar uma função do Lambda em um navegador

O exemplo de código a seguir mostra como invocar uma AWS Lambda função em um navegador.

SDK para JavaScript (v2)

Você pode criar um aplicativo baseado em navegador que usa uma AWS Lambda função para atualizar uma tabela do Amazon DynamoDB com as seleções do usuário.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB

- Lambda

AWS Entity Resolution exemplos de uso do SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com AWS Entity Resolution.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie mapeamento de esquemas.
- Crie um AWS Entity Resolution fluxo de trabalho.
- Inicie o trabalho correspondente para o fluxo de trabalho.
- Obtenha detalhes do trabalho correspondente.
- Obtenha o mapeamento de esquemas.
- Liste todos os mapeamentos de esquemas.
- Marque o recurso de mapeamento de esquemas.
- Exclua os AWS Entity Resolution ativos.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS Entity Resolution recursos.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
} from "@aws-sdk/client-cloudformation";
import {
  EntityResolutionClient,
  CreateSchemaMappingCommand,
  CreateMatchingWorkflowCommand,
  GetMatchingJobCommand,
  StartMatchingJobCommand,
  GetSchemaMappingCommand,
  ListSchemaMappingsCommand,
  TagResourceCommand,
  DeleteMatchingWorkflowCommand,
  DeleteSchemaMappingCommand,
  ConflictException,
  ValidationException,
} from "@aws-sdk/client-entityresolution";
import {
  DeleteObjectsCommand,
  DeleteBucketCommand,
  PutObjectCommand,
```

```
  S3Client,
  ListObjectsCommand,
} from "@aws-sdk/client-s3";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { readFile } from "node:fs/promises";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = `${data.inputs.entityResolutionStack}`;

/*The inputs for this example can be edited in the ../input.json.*/
import data from "../inputs.json" with { type: "json" };
const skipWhenErrors = (state) => state.errors.length > 0;
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
/* v8 ignore next 3 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  verbose: "false",
  skipWhen: skipWhenErrors,
});

const region = "eu-west-1";

const entityResolutionClient = new EntityResolutionClient({ region: region });
const cloudFormationClient = new CloudFormationClient({ region: region });
const s3Client = new S3Client({ region: region });

const greet = new ScenarioOutput(
  "greet",
  "AWS Entity Resolution is a fully-managed machine learning service provided by " +
    "Amazon Web Services (AWS) that helps organizations extract, link, and " +
    "organize information from multiple data sources. It leverages natural " +
    "language processing and deep learning models to identify and resolve " +
    "entities, such as people, places, organizations, and products, " +
    "across structured and unstructured data.\n" +
    "\n" +
```

```
    "With Entity Resolution, customers can build robust data integration " +
    "pipelines to combine and reconcile data from multiple systems, databases, " +
    "and documents. The service can handle ambiguous, incomplete, or conflicting " +
    "information, and provide a unified view of entities and their relationships. "
 +
    "This can be particularly valuable in applications such as customer 360, " +
    "fraud detection, supply chain management, and knowledge management, where " +
    "accurate entity identification is crucial.\n" +
    "\n" +
    "The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x " +
    "provides a set of methods to programmatically interact with the AWS Entity " +
    "Resolution service. This allows developers to automate the entity extraction, "
 +
    "linking, and deduplication process as part of their data processing workflows.
 " +
    "With Entity Resolution, organizations can unlock the value of their data, " +
    "improve decision-making, and enhance customer experiences by having a reliable,
 " +
    "comprehensive view of their key entities.",

  { header: true },
);
const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "To prepare the AWS resources needed for this scenario application, the next step
 uploads " +
    "a CloudFormation template whose resulting stack creates the following
 resources:\n" +
    "- An AWS Glue Data Catalog table \n" +
    "- An AWS IAM role \n" +
    "- An AWS S3 bucket \n" +
    "- An AWS Entity Resolution Schema \n" +
    "It can take a couple minutes for the Stack to finish creating the resources.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../resources/cfn/entity-resolution-basics/entity-
resolution-basics-template.yml`,
        "utf8",
      );
```

```
      await cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: cloudFormationClient },
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );

      state.entityResolutionRole = stack.Stacks[0].Outputs[1];
      state.jsonGlueTable = stack.Stacks[0].Outputs[2];
      state.CSVGlueTable = stack.Stacks[0].Outputs[3];
      state.glueDataBucket = stack.Stacks[0].Outputs[0];
      state.stackName = stack.StackName;
      console.log(state.glueDataBucket);
      console.log(
        `The  ARN of the EntityResolution Role is
${state.entityResolutionRole.OutputValue}`,
      );
      console.log(
        `The ARN of the Json Glue Table is ${state.jsonGlueTable.OutputValue}`,
      );
      console.log(
        `The ARN of the CSV Glue Table is ${state.CSVGlueTable.OutputValue}`,
      );
      console.log(
        `The name of the Glue Data Bucket is ${state.glueDataBucket.OutputValue}\n`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
```

```
    try {
      console.log(
        `Uploading the following JSON in ../data.json to the
 ${state.glueDataBucket.OutputValue} S3 bucket...`,
      );
      const bucketName = state.glueDataBucket.OutputValue;

      const putObjectParams = {
        Bucket: bucketName,
        Key: "jsonData/data.json",
        Body: await readFileSync(
          `${__dirname}/../../../../javascriptv3/example_code/entityresolution/
data.json`,
        ),
      };
      const command = new PutObjectCommand(putObjectParams);
      const response = await s3Client.send(command);
      console.log(
        `../data.json file data uploaded to the ${state.glueDataBucket.OutputValue}
 S3 bucket.\n`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
    try {
      console.log(
        `Uploading the CSV data in ../data.csv to the
 ${state.glueDataBucket.OutputValue} S3 bucket...`,
      );

      const bucketName = state.glueDataBucket.OutputValue;
      const putObjectParams = {
        Bucket: bucketName,
        Key: "csvData/data.csv",
        Body: await readFileSync(
          `${__dirname}/../../../../javascriptv3/example_code/entityresolution/
data.csv`,
        ),
      };
      const command = new PutObjectCommand(putObjectParams);
      const response = await s3Client.send(command);
      console.log(
```

```
        `../data.csv file data uploaded to the ${state.glueDataBucket.OutputValue}
 S3 bucket.`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateSchemaMapping = new ScenarioOutput(
  "displayCreateSchemaMapping",
  "1. Create Schema Mapping" +
    "Entity Resolution schema mapping aligns and integrates data from " +
    "multiple sources by identifying and matching corresponding entities " +
    "like customers or products. It unifies schemas, resolves conflicts, " +
    "and uses machine learning to link related entities, enabling a " +
    "consolidated, accurate view for improved data quality and decision-making." +
    "\n" +
    "In this example, the schema mapping lines up with the fields in the JSON and
 CSV objects. That is, " +
    " it contains these fields: id, name, and email. ",
);

const sdkCreateSchemaMapping = new ScenarioAction(
  "sdkCreateSchemaMapping",
  async (/** @type {State} */ state) => {
    const createSchemaMappingParamsJson = {
      schemaName: `${data.inputs.schemaNameJson}`,
      mappedInputFields: [
        {
          fieldName: "id",
          type: "UNIQUE_ID",
        },
        {
          fieldName: "name",
          type: "NAME",
        },
        {
          fieldName: "email",
          type: "EMAIL_ADDRESS",
        },
      ],
    };
```

```
    const createSchemaMappingParamsCSV = {
      schemaName: `${data.inputs.schemaNameCSV}`,
      mappedInputFields: [
        {
          fieldName: "id",
          type: "UNIQUE_ID",
        },
        {
          fieldName: "name",
          type: "NAME",
        },
        {
          fieldName: "email",
          type: "EMAIL_ADDRESS",
        },
        {
          fieldName: "phone",
          type: "PROVIDER_ID",
          subType: "STRING",
        },
      ],
    };
    try {
      const command = new CreateSchemaMappingCommand(
        createSchemaMappingParamsJson,
      );
      const response = await entityResolutionClient.send(command);
      state.schemaNameJson = response.schemaName;
      state.schemaArn = response.schemaArn;
      state.idOutputAttribute = response.mappedInputFields[0].fieldName;
      state.nameOutputAttribute = response.mappedInputFields[1].fieldName;
      state.emailOutputAttribute = response.mappedInputFields[2].fieldName;

      console.log("The JSON schema mapping name is ", state.schemaNameJson);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema mapping already exists: ${caught.message} \n Exiting
program.`,
        );
        return;
      }
    }
    try {
```

```
      const command = new CreateSchemaMappingCommand(
        createSchemaMappingParamsCSV,
      );
      const response = await entityResolutionClient.send(command);
      state.schemaNameCSV = response.schemaName;
      state.phoneOutputAttribute = response.mappedInputFields[3].fieldName;
      console.log("The CSV schema mapping name is ", state.schemaNameCSV);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An unexpected error occurred while creating the geofence collection:
 ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayCreateMatchingWorkflow = new ScenarioOutput(
  "displayCreateMatchingWorkflow",
  "2. Create an AWS Entity Resolution Workflow. " +
    "An Entity Resolution matching workflow identifies and links records " +
    "across datasets that represent the same real-world entity, such as " +
    "customers or products. Using techniques like schema mapping, " +
    "data profiling, and machine learning algorithms, " +
    "it evaluates attributes like names or emails to detect duplicates " +
    "or relationships, even with variations or inconsistencies. " +
    "The workflow outputs consolidated, de-duplicated data." +
    "\n" +
    "We will use the machine learning-based matching technique.",
);

const sdkCreateMatchingWorkflow = new ScenarioAction(
  "sdkCreateMatchingWorkflow",
  async (/** @type {State} */ state) => {
    const createMatchingWorkflowParams = {
      roleArn: `${state.entityResolutionRole.OutputValue}`,
      workflowName: `${data.inputs.workflowName}`,
      description: "Created by using the AWS SDK for JavaScript (v3).",
      inputSourceConfig: [
        {
          inputSourceARN: `${state.jsonGlueTable.OutputValue}`,
          schemaName: `${data.inputs.schemaNameJson}`,
          applyNormalization: false,
```

```
        },
        {
          inputSourceARN: `${state.CSVGlueTable.OutputValue}`,
          schemaName: `${data.inputs.schemaNameCSV}`,
          applyNormalization: false,
        },
      ],
      outputSourceConfig: [
        {
          outputS3Path: `s3://${state.glueDataBucket.OutputValue}/eroutput`,
          output: [
            {
              name: state.idOutputAttribute,
            },
            {
              name: state.nameOutputAttribute,
            },
            {
              name: state.emailOutputAttribute,
            },
            {
              name: state.phoneOutputAttribute,
            },
          ],
          applyNormalization: false,
        },
      ],
      resolutionTechniques: { resolutionType: "ML_MATCHING" },
    };
    try {
      const command = new CreateMatchingWorkflowCommand(
        createMatchingWorkflowParams,
      );
      const response = await entityResolutionClient.send(command);
      state.workflowArn = response.workflowArn;
      console.log(
        `Workflow created successfully.\n The workflow ARN is:
${response.workflowArn}`,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The matching workflow already exists: ${caught.message} \n Exiting
program.`,
```

```
        );
        return;
      }
      if (caught instanceof ValidationException) {
        console.error(
          `There was a validation exception: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayMatchingJobOfWorkflow = new ScenarioOutput(
  "displayMatchingJobOfWorkflow",
  "3. Start the matching job of the workflow",
);

const sdkMatchingJobOfWorkflow = new ScenarioAction(
  "sdk",
  async (/** @type {State} */ state) => {
    const matchingJobOfWorkflowParams = {
      workflowName: `${data.inputs.workflowName}`,
    };
    try {
      const command = new StartMatchingJobCommand(matchingJobOfWorkflowParams);
      const response = await entityResolutionClient.send(command);
      state.jobID = response.jobId;
      console.log(`Job ID: ${state.jobID} \n
The matching job was successfully started.`);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The matching workflow already exists: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
    }
  },
);

const displayGetDetailsforJob = new ScenarioOutput(
  "displayGetDetailsforJob",
```

```
  `4. While the matching job is running, let's look at other API methods. First,
 let's get details for the job `,
);

const sdkGetDetailsforJob = new ScenarioAction(
  "sdkGetDetailsforJob",
  async (/** @type {State} */ state) => {
    const getDetailsforJobParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${state.jobID}`,
    };
    try {
      const command = new GetMatchingJobCommand(getDetailsforJobParams);
      const response = await entityResolutionClient.send(command);
      state.Status = response.status;
      state.response = response;
      console.log(`Job status: ${state.Status} `);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayGetSchemaMappingJson = new ScenarioOutput(
  "displayGetSchemaMappingJson",
  "5. Get the schema mapping for the JSON data.",
);

const sdkGetSchemaMappingJson = new ScenarioAction(
  "sdkGetSchemaMappingJson",
  async (/** @type {State} */ state) => {
    const getSchemaMappingJsonParams = {
      schemaName: `${data.inputs.schemaNameJson}`,
    };
    try {
      const command = new GetSchemaMappingCommand(getSchemaMappingJsonParams);
      const response = await entityResolutionClient.send(command);
      console.log("Schema·mapping·ARN·is:·", response.schemaArn);
      const resultMappings = response.mappedInputFields;
      const noOfResultMappings = resultMappings.length;
      for (let i = 0; i < noOfResultMappings; i++) {
        console.log(
          `Attribute name: ${resultMappings[i].fieldName} `,
```

```
          `Attribute type: ${resultMappings[i].type}`,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayListSchemaMappings = new ScenarioOutput(
  "displayListSchemaMappings",
  "6. List Schema Mappings.",
);

const sdkListSchemaMappings = new ScenarioAction(
  "sdkListSchemaMappings",
  async (/** @type {State} */ state) => {
    try {
      const command = new ListSchemaMappingsCommand({});
      const response = await entityResolutionClient.send(command);
      const noOfSchemas = response.schemaList.length;
      for (let i = 0; i < noOfSchemas; i++) {
        console.log(
          `Schema Mapping Name: ${response.schemaList[i].schemaName} `,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayTagTheJsonSchema = new ScenarioOutput(
  "display",
  "7. Tag the resource. \n" +
    "Tags can help you organize and categorize your Entity Resolution resources. " +
    "You can also use them to scope user permissions by granting a user permission "
 +
    "to access or change only resources with certain tag values. " +
    "In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged. For
 this example, " +
    "the SchemaMapping is tagged.",
```

```
);

const sdkTagTheJsonSchema = new ScenarioAction(
  "sdkGetSchemaMappingJson",
  async (/** @type {State} */ state) => {
    const tagResourceCommandParams = {
      resourceArn: state.schemaArn,
      tags: {
        tag1: "tag1Value",
        tag2: "tag2Value",
      },
    };
    try {
      const command = new TagResourceCommand(tagResourceCommandParams);
      const response = await entityResolutionClient.send(command);
      console.log("Successfully tagged the resource.");
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayGetJobInfo = new ScenarioOutput(
  "displayGetJobInfo",
  "8. View the results of the AWS Entity Resolution Workflow.\n " +
    "Please perform this task manually in the AWS Management Console. ",
);

const displayDeleteResources = new ScenarioOutput(
  "displayDeleteResources",
  "9. Delete the resources \n" +
    "You cannot delete a workflow that is in a running state. So this will take ~30
 minutes.\n" +
    "If you don't want to delete the resources, simply exit this application.",
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    console.log(
      "You selected to delete the resources. This will take about 30 minutes.",
    );
    await wait(1800);
```

```
    const bucketName = state.glueDataBucket.OutputValue;
    try {
      const emptyBucket = async ({ bucketName }) => {
        const listObjectsCommand = new ListObjectsCommand({
          Bucket: bucketName,
        });
        const { Contents } = await s3Client.send(listObjectsCommand);
        const keys = Contents.map((c) => c.Key);

        const deleteObjectsCommand = new DeleteObjectsCommand({
          Bucket: bucketName,
          Delete: { Objects: keys.map((key) => ({ Key: key })) },
        });
        await s3Client.send(deleteObjectsCommand);
        console.log(`Bucket ${bucketName} emptied successfully.\n`);
      };
      await emptyBucket({ bucketName });
    } catch (error) {
      console.log("error ", error);
    }
    try {
      const deleteBucket = async ({ bucketName }) => {
        const command = new DeleteBucketCommand({ Bucket: bucketName });
        await s3Client.send(command);
        console.log(`Bucket ${bucketName} deleted successfully.\n`);
      };
      await deleteBucket({ bucketName });
    } catch (error) {
      console.log("error ", error);
    }
    try {
      console.log(
        "Now we will delete the CloudFormation stack, which deletes the resources
that were created at the beginning of the scenario.",
      );
      const deleteStackParams = { StackName: `${state.stackName}` };
      const command = new DeleteStackCommand(deleteStackParams);
      const response = await cloudFormationClient.send(command);
      console.log("CloudFormation stack deleted successfully.");
    } catch (error) {
      console.log("error ", error);
    }
    try {
      const deleteWorkflowParams = {
```

```
        workflowName: `${data.inputs.workflowName}`,
      };
      const command = new DeleteMatchingWorkflowCommand(deleteWorkflowParams);
      const response = await entityResolutionClient.send(command);
      console.log("Workflow deleted successfully!");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `Job associated with workflow ${data.inputs.workflowName} is still
running, so can't be deleted.
          Neither can schemas ${data.inputs.schemaNameJson} and
${data.inputs.schemaNameCSV} associated with it. Please confirm this workflow is
finished in the AWS Management Console, then delete it manually.`,
        );
        throw caught;
      }
    }
    try {
      const deleteJSONschemaMapping = {
        schemaName: `${data.inputs.schemaNameJson}`,
      };
      const command = new DeleteSchemaMappingCommand(deleteJSONschemaMapping);
      const response = await entityResolutionClient.send(command);
      console.log("Schema mapping deleted successfully. ");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema ${data.inputs.schemaNameJson} can't be deleted because it is
associated with workflow
           ${data.inputs.workflowName}, which is still running. Please confirm this
workflow is finished in the AWS Management Console, then delete it manually.`,
        );
        throw caught;
      }
    }
    try {
      const deleteCSVschemaMapping = {
        schemaName: `${data.inputs.schemaNameCSV}`,
      };
      const command = new DeleteSchemaMappingCommand(deleteCSVschemaMapping);
      const response = await entityResolutionClient.send(command);
      console.log("Schema mapping deleted successfully.");
    } catch (caught) {
      if (caught instanceof ConflictException) {
```

```
        console.error(
          `The schema ${data.inputs.schemaNameCSV} can't be deleted because it
 is associated with workflow ${data.inputs.workflowName}, which is still running.
 Please confirm this workflow is finished in the AWS Management Console, then delete
 it manually.`,
        );
        throw caught;
      }
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.confirmDeleteResources === "",
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Location Service Use demo. We hope you " +
    "learned something new, or got some inspiration for your own apps today!" +
    " For more Amazon Location Services examples in different programming languages,
 have a look at: " +
    "https://docs.aws.amazon.com/code-library/latest/ug/
location_code_examples.html",
);

const myScenario = new Scenario("Entity Resolution Basics Scenario", [
  greet,
  pressEnter,
  displayBuildCloudFormationStack,
  sdkBuildCloudFormationStack,
  pressEnter,
  displayCreateSchemaMapping,
  sdkCreateSchemaMapping,
  pressEnter,
  displayCreateMatchingWorkflow,
  sdkCreateMatchingWorkflow,
  pressEnter,
  displayMatchingJobOfWorkflow,
  sdkMatchingJobOfWorkflow,
  pressEnter,
  displayGetDetailsforJob,
  sdkGetDetailsforJob,
  pressEnter,
```

```
  displayGetSchemaMappingJson,
  sdkGetSchemaMappingJson,
  pressEnter,
  displayListSchemaMappings,
  sdkListSchemaMappings,
  pressEnter,
  displayTagTheJsonSchema,
  sdkTagTheJsonSchema,
  pressEnter,
  displayGetJobInfo,
  pressEnter,
  displayDeleteResources,
  pressEnter,
  sdkDeleteResources,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateMatchingWorkflow
  - CreateSchemaMapping
  - DeleteMatchingWorkflow

- DeleteSchemaMapping
- GetMatchingJob
- GetSchemaMapping
- ListMatchingWorkflows
- ListSchemaMappings
- StartMatchingJob
- TagResource

EventBridge exemplos de uso do SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com EventBridge.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};

ebevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK para JavaScript da API.

`PutRule`

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

ebevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- Para obter detalhes da API, consulte PutRule em Referência AWS SDK para JavaScript da API.

`PutTargets`

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
```

```
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myEventBridgeTarget",
    },
  ],
};

ebevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter detalhes da API, consulte PutTargets em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon Glacier usando o SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com o Amazon Glacier.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateVault`

O código de exemplo a seguir mostra como usar `CreateVault`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
// Call Glacier to create the vault
glacier.createVault({ vaultName: "YOUR_VAULT_NAME" }, function (err) {
  if (!err) {
    console.log("Created vault!");
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateVault em Referência AWS SDK para JavaScript da API.

`UploadArchive`

O código de exemplo a seguir mostra como usar `UploadArchive`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object and buffer
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = { vaultName: "YOUR_VAULT_NAME", body: buffer };
// Call Glacier to upload the archive.
glacier.uploadArchive(params, function (err, data) {
  if (err) {
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UploadArchive em Referência AWS SDK para JavaScript da API.

`UploadMultipartPart`

O código de exemplo a seguir mostra como usar `UploadMultipartPart`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um carregamento fracionado com pedaços de 1 megabyte de um objeto Buffer.

```
// Create a new service object and some supporting variables
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" }),
  vaultName = "YOUR_VAULT_NAME",
  buffer = new Buffer(2.5 * 1024 * 1024), // 2.5MB buffer
  partSize = 1024 * 1024, // 1MB chunks,
  numPartsLeft = Math.ceil(buffer.length / partSize),
  startTime = new Date(),
  params = { vaultName: vaultName, partSize: partSize.toString() };

// Compute the complete SHA-256 tree hash so we can pass it
// to completeMultipartUpload request at the end
var treeHash = glacier.computeChecksums(buffer).treeHash;

// Initiate the multipart upload
console.log("Initiating upload to", vaultName);
// Call Glacier to initiate the upload.
glacier.initiateMultipartUpload(params, function (mpErr, multipart) {
  if (mpErr) {
    console.log("Error!", mpErr.stack);
    return;
  }
  console.log("Got upload ID", multipart.uploadId);

  // Grab each partSize chunk and upload it as a part
  for (var i = 0; i < buffer.length; i += partSize) {
    var end = Math.min(i + partSize, buffer.length),
      partParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        range: "bytes " + i + "-" + (end - 1) + "/*",
        body: buffer.slice(i, end),
      };
```

```
    // Send a single part
    console.log("Uploading part", i, "=", partParams.range);
    glacier.uploadMultipartPart(partParams, function (multiErr, mData) {
      if (multiErr) return;
      console.log("Completed part", this.request.params.range);
      if (--numPartsLeft > 0) return; // complete only when all parts uploaded

      var doneParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        archiveSize: buffer.length.toString(),
        checksum: treeHash, // the computed tree hash
      };

      console.log("Completing upload...");
      glacier.completeMultipartUpload(doneParams, function (err, data) {
        if (err) {
          console.log("An error occurred while uploading the archive");
          console.log(err);
        } else {
          var delta = (new Date() - startTime) / 1000;
          console.log("Completed upload in", delta, "seconds");
          console.log("Archive ID:", data.archiveId);
          console.log("Checksum:  ", data.checksum);
        }
      });
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UploadMultipartPart em Referência AWS SDK para JavaScript da API.

Exemplos de IAM usando SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com o IAM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
```

```
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    var params = {
      PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
      RoleName: process.argv[2],
    };
    iam.attachRolePolicy(params, function (err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte AttachRolePolicy em Referência AWS SDK para JavaScript da API.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccessKey({ UserName: "IAM_USER_NAME" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateAccessKey em Referência AWS SDK para JavaScript da API.

`CreateAccountAlias`

O código de exemplo a seguir mostra como usar `CreateAccountAlias`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
```

```
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateAccountAlias em Referência AWS SDK para JavaScript da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var myManagedPolicy = {
  Version: "2012-10-17",
  Statement: [
    {
      Effect: "Allow",
      Action: "logs:CreateLogGroup",
      Resource: "RESOURCE_ARN",
    },
    {
```

```
      Effect: "Allow",
      Action: [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem",
      ],
      Resource: "RESOURCE_ARN",
    },
  ],
};

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "myDynamoDBPolicy",
};

iam.createPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK para JavaScript da API.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    iam.createUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log(
      "User " + process.argv[2] + " already exists",
      data.User.UserId
    );
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateUser em Referência AWS SDK para JavaScript da API.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  UserName: "USER_NAME",
};

iam.deleteAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAccessKey em Referência AWS SDK para JavaScript da API.

`DeleteAccountAlias`

O código de exemplo a seguir mostra como usar `DeleteAccountAlias`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAccountAlias em Referência AWS SDK para JavaScript da API.

`DeleteServerCertificate`

O código de exemplo a seguir mostra como usar `DeleteServerCertificate`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteServerCertificate em Referência AWS SDK para JavaScript da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para JavaScript da API.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
```

```
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        var params = {
          PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
          RoleName: process.argv[2],
        };
        iam.detachRolePolicy(params, function (err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DetachRolePolicy em Referência AWS SDK para JavaScript da API.

`GetAccessKeyLastUsed`

O código de exemplo a seguir mostra como usar `GetAccessKeyLastUsed`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getAccessKeyLastUsed(
  { AccessKeyId: "ACCESS_KEY_ID" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.AccessKeyLastUsed);
    }
  }
);
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetAccessKeyLastUsed em Referência AWS SDK para JavaScript da API.

`GetPolicy`

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  PolicyArn: "arn:aws:iam::aws:policy/AWSLambdaExecute",
};

iam.getPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetPolicy em Referência AWS SDK para JavaScript da API.

`GetServerCertificate`

O código de exemplo a seguir mostra como usar `GetServerCertificate`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetServerCertificate em Referência AWS SDK para JavaScript da API.

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME",
};

iam.listAccessKeys(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListAccessKeys em Referência AWS SDK para JavaScript da API.

`ListAccountAliases`

O código de exemplo a seguir mostra como usar `ListAccountAliases`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listAccountAliases({ MaxItems: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListAccountAliases em Referência AWS SDK para JavaScript da API.

`ListServerCertificates`

O código de exemplo a seguir mostra como usar `ListServerCertificates`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listServerCertificates({}, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListServerCertificates em Referência AWS SDK para JavaScript da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });
```

```
var params = {
  MaxItems: 10,
};

iam.listUsers(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para JavaScript da API.

`UpdateAccessKey`

O código de exemplo a seguir mostra como usar `UpdateAccessKey`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });
```

```
var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  Status: "Active",
  UserName: "USER_NAME",
};

iam.updateAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateAccessKey em Referência AWS SDK para JavaScript da API.

`UpdateServerCertificate`

O código de exemplo a seguir mostra como usar `UpdateServerCertificate`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  ServerCertificateName: "CERTIFICATE_NAME",
```

```
  NewServerCertificateName: "NEW_CERTIFICATE_NAME",
};

iam.updateServerCertificate(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateServerCertificate em Referência AWS SDK para JavaScript da API.

`UpdateUser`

O código de exemplo a seguir mostra como usar `UpdateUser`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3],
};
```

```
iam.updateUser(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateUser em Referência AWS SDK para JavaScript da API.

Exemplos de Lambda usando SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com o Lambda.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Invocar uma função do Lambda em um navegador

O exemplo de código a seguir mostra como invocar uma AWS Lambda função em um navegador.

SDK para JavaScript (v2)

Você pode criar um aplicativo baseado em navegador que usa uma AWS Lambda função para atualizar uma tabela do Amazon DynamoDB com as seleções do usuário.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda

Exemplos do Amazon Pinpoint usando o SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendMessages`

O código de exemplo a seguir mostra como usar `SendMessages`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
"use strict";

const AWS = require("aws-sdk");
```

```
// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/
const aws_region = "us-west-2";

// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
const senderAddress = "sender@example.com";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
var toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
const appId = "ce796be37f32f178af652b26eexample";

// The subject line of the email.
var subject = "Amazon Pinpoint (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
----------------------------------------------------
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
 Node.js.
For more information, see https:\/\/aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint API</a> using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding the you want to use for the subject line and
// message body of the email.
var charset = "UTF-8";
```

```
// Specify that you're using a shared credentials file.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: appId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: senderAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

//Try to send the email.
pinpoint.sendMessages(params, function (err, data) {
```

```
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
  } else {
    console.log(
      "Email sent! Message ID: ",
      data["MessageResponse"]["Result"][toAddress]["MessageId"]
    );
  }
});
```

Envie uma mensagem SMS.

```
"use strict";

var AWS = require("aws-sdk");

// The AWS Region that you want to use to send the message. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/.
var aws_region = "us-east-1";

// The phone number or short code to send the message from. The phone number
// or short code that you specify has to be associated with your Amazon Pinpoint
// account. For best results, specify long codes in E.164 format.
var originationNumber = "+12065550199";

// The recipient's phone number.  For best results, you should specify the
// phone number in E.164 format.
var destinationNumber = "+14255550142";

// The content of the SMS message.
var message =
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out.";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
```

```
var applicationId = "ce796be37f32f178af652b26eexample";

// The type of SMS message that you want to send. If you plan to send
// time-sensitive content, specify TRANSACTIONAL. If you plan to send
// marketing-related content, specify PROMOTIONAL.
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
var senderId = "MySenderID";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: applicationId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
```

```
  },
};

//Try to send the message.
pinpoint.sendMessages(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
    // Otherwise, show the unique ID for the message.
  } else {
    console.log(
      "Message sent! " +
        data["MessageResponse"]["Result"][destinationNumber]["StatusMessage"]
    );
  }
});
```

- Para obter detalhes da API, consulte SendMessages em Referência AWS SDK para JavaScript da API.

Exemplos da API de SMS e voz do Amazon Pinpoint usando o SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a AWS SDK para JavaScript (v2) com a API de SMS e voz do Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendVoiceMessage`

O código de exemplo a seguir mostra como usar `SendVoiceMessage`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"use strict";

var AWS = require("aws-sdk");

// The AWS Region that you want to use to send the voice message. For a list of
// AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see
// https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/
var aws_region = "us-east-1";

// The phone number that the message is sent from. The phone number that you
// specify has to be associated with your Amazon Pinpoint account. For best results,
 you
// should specify the phone number in E.164 format.
var originationNumber = "+12065550110";

// The recipient's phone number. For best results, you should specify the phone
// number in E.164 format.
var destinationNumber = "+12065550142";

// The language to use when sending the message. For a list of supported
// languages, see https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
var languageCode = "en-US";

// The Amazon Polly voice that you want to use to send the message. For a list
// of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
var voiceId = "Matthew";

// The content of the message. This example uses SSML to customize and control
```

```
// certain aspects of the message, such as the volume or the speech rate.
// The message can't contain any line breaks.
var ssmlMessage =
  "<speak>" +
  "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> " +
  "using the <break strength='weak'/>AWS SDK for JavaScript in Node.js. " +
  "<amazon:effect phonation='soft'>Thank you for listening." +
  "</amazon:effect>" +
  "</speak>";

// The phone number that you want to appear on the recipient's device. The phone
// number that you specify has to be associated with your Amazon Pinpoint account.
var callerId = "+12065550199";

// The configuration set that you want to use to send the message.
var configurationSet = "ConfigSet";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpointsmsvoice = new AWS.PinpointSMSVoice();

var params = {
  CallerId: callerId,
  ConfigurationSetName: configurationSet,
  Content: {
    SSMLMessage: {
      LanguageCode: languageCode,
      Text: ssmlMessage,
      VoiceId: voiceId,
    },
  },
  DestinationPhoneNumber: destinationNumber,
  OriginationPhoneNumber: originationNumber,
};

//Try to send the message.
pinpointsmsvoice.sendVoiceMessage(params, function (err, data) {
```

```
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
    // Otherwise, show the unique ID for the message.
  } else {
    console.log("Message sent! Message ID: " + data["MessageId"]);
  }
});
```

- Para obter detalhes da API, consulte SendVoiceMessage em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon SNS usando o SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetTopicAttributes`

O código de exemplo a seguir mostra como usar `GetTopicAttributes`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetTopicAttributes em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon SQS usando o SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ChangeMessageVisibility`

O código de exemplo a seguir mostra como usar `ChangeMessageVisibility`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem do Amazon SQS e altere sua visibilidade de tempo limite.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region to us-west-2
AWS.config.update({ region: "us-west-2" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
```

```
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else {
    // Make sure we have a message
    if (data.Messages != null) {
      var visibilityParams = {
        QueueUrl: queueURL,
        ReceiptHandle: data.Messages[0].ReceiptHandle,
        VisibilityTimeout: 20, // 20 second timeout
      };
      sqs.changeMessageVisibility(visibilityParams, function (err, data) {
        if (err) {
          console.log("Delete Error", err);
        } else {
          console.log("Timeout Changed", data);
        }
      });
    } else {
      console.log("No messages to change");
    }
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ChangeMessageVisibility em Referência AWS SDK para JavaScript da API.

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma fila padrão do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    DelaySeconds: "60",
    MessageRetentionPeriod: "86400",
  },
};

sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

Crie uma fila do Amazon SQS que aguarda a chegada de uma mensagem.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
};
```

```
sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateQueue em Referência AWS SDK para JavaScript da API.

`DeleteMessage`

O código de exemplo a seguir mostra como usar `DeleteMessage`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receber e excluir mensagens do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 10,
```

```
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else if (data.Messages) {
    var deleteParams = {
      QueueUrl: queueURL,
      ReceiptHandle: data.Messages[0].ReceiptHandle,
    };
    sqs.deleteMessage(deleteParams, function (err, data) {
      if (err) {
        console.log("Delete Error", err);
      } else {
        console.log("Message Deleted", data);
      }
    });
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteMessage em Referência AWS SDK para JavaScript da API.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueUrl: "SQS_QUEUE_URL",
};

sqs.deleteQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteQueue em Referência AWS SDK para JavaScript da API.

`GetQueueUrl`

O código de exemplo a seguir mostra como usar `GetQueueUrl`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha o URL para uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
};

sqs.getQueueUrl(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetQueueUrl em Referência AWS SDK para JavaScript da API.

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar filas do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
```

```
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {};

sqs.listQueues(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrls);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para JavaScript da API.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
```

```
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  WaitTimeSeconds: 20,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ReceiveMessage em Referência AWS SDK para JavaScript da API.

`SendMessage`

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Enviar uma mensagem para uma fila do Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
```

```
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  // Remove DelaySeconds parameter and value for FIFO queues
  DelaySeconds: 10,
  MessageAttributes: {
    Title: {
      DataType: "String",
      StringValue: "The Whistler",
    },
    Author: {
      DataType: "String",
      StringValue: "John Grisham",
    },
    WeeksOn: {
      DataType: "Number",
      StringValue: "6",
    },
  },
  MessageBody:
    "Information about current NY Times fiction bestseller for week of 12/11/2016.",
  // MessageDeduplicationId: "TheWhistler",  // Required for FIFO queues
  // MessageGroupId: "Group1",  // Required for FIFO queues
  QueueUrl: "SQS_QUEUE_URL",
};

sqs.sendMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.MessageId);
  }
});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte SendMessage em Referência AWS SDK para JavaScript da API.

AWS STS exemplos de uso do SDK para JavaScript (v2)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v2) com AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para JavaScript (v2)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;
```

```
// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- Para obter detalhes da API, consulte AssumeRole em Referência AWS SDK para JavaScript da API.

Exemplos de código para SDK for JavaScript (v3)

Os exemplos de código a seguir mostram como usar o AWS SDK para JavaScript (v3) com AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para JavaScript (v3) — Saiba mais sobre como usar JavaScript com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos de API Gateway usando SDK para JavaScript (v3)
- Exemplos do Aurora usando o SDK para JavaScript (v3)
- Exemplos de Auto Scaling usando o SDK for JavaScript (v3)
- Exemplos do Amazon Bedrock usando o SDK para JavaScript (v3)
- Exemplos do Amazon Bedrock Runtime usando o SDK para JavaScript (v3)
- Exemplos de Amazon Bedrock Agents usando o SDK para JavaScript (v3)
- Exemplos do Amazon Bedrock Agents Runtime usando o SDK para JavaScript (v3)
- CloudWatch exemplos de uso do SDK para JavaScript (v3)
- CloudWatch Exemplos de eventos usando o SDK for JavaScript (v3)
- CloudWatch Exemplos de registros usando o SDK for JavaScript (v3)
- CodeBuild exemplos de uso do SDK para JavaScript (v3)
- Exemplos de identidade do Amazon Cognito usando o SDK para JavaScript (v3)
- Exemplos do Amazon Cognito Identity Provider usando o SDK para JavaScript (v3)
- Exemplos do Amazon Comprehend usando o SDK para JavaScript (v3)

- Exemplos do Amazon DocumentDB usando o SDK para JavaScript (v3)
- Exemplos do DynamoDB usando o SDK for JavaScript (v3)
- Exemplos do Amazon EC2 usando o SDK para JavaScript (v3)
- Elastic Load Balancing - exemplos da versão 2 usando o SDK para JavaScript (v3)
- AWS Entity Resolution exemplos de uso do SDK para JavaScript (v3)
- EventBridge exemplos de uso do SDK para JavaScript (v3)
- Exemplos do Amazon Glacier usando o SDK para JavaScript (v3)
- AWS Glue exemplos de uso do SDK para JavaScript (v3)
- HealthImaging exemplos de uso do SDK para JavaScript (v3)
- Exemplos de IAM usando SDK para JavaScript (v3)
- AWS IoT SiteWise exemplos de uso do SDK para JavaScript (v3)
- Exemplos do Kinesis usando o SDK para JavaScript (v3)
- Exemplos de Lambda usando SDK para JavaScript (v3)
- Exemplos do Amazon Lex usando o SDK para JavaScript (v3)
- Exemplos de localização da Amazon usando o SDK para JavaScript (v3)
- AWS Marketplace Exemplos de API de contrato usando SDK para JavaScript (v3)
- Exemplos do Amazon MSK usando o SDK para JavaScript (v3)
- Exemplos do Amazon Personalize usando o SDK para JavaScript (v3)
- Exemplos de eventos do Amazon Personalize usando o SDK para JavaScript (v3)
- Exemplos do Amazon Personalize Runtime usando o SDK para JavaScript (v3)
- Exemplos do Amazon Pinpoint usando o SDK para JavaScript (v3)
- Exemplos do Amazon Polly usando o SDK para JavaScript (v3)
- Exemplos do Amazon RDS usando o SDK para JavaScript (v3)
- Exemplos do Amazon RDS Data Service usando o SDK para JavaScript (v3)
- Exemplos do Amazon Redshift usando o SDK para JavaScript (v3)
- Exemplos do Amazon Rekognition usando o SDK para (v3) JavaScript
- Exemplos do Amazon S3 usando o SDK para JavaScript (v3)
- SageMaker Exemplos de IA usando SDK para JavaScript (v3)
- Exemplos do Secrets Manager usando o SDK para JavaScript (v3)

- Exemplos do Amazon SES usando o SDK para JavaScript (v3)
- Exemplos do Amazon SNS usando o SDK para JavaScript (v3)
- Exemplos do Amazon SQS usando o SDK para JavaScript (v3)
- Exemplos do Step Functions usando o SDK for JavaScript (v3)
- AWS STS exemplos de uso do SDK para JavaScript (v3)
- Suporte exemplos de uso do SDK para JavaScript (v3)
- Exemplos do Systems Manager usando o SDK para JavaScript (v3)
- Exemplos do Amazon Textract usando o SDK para JavaScript (v3)
- Exemplos do Amazon Transcribe usando o SDK para JavaScript (v3)
- Exemplos do Amazon Translate usando o SDK para JavaScript (v3)

Exemplos de API Gateway usando SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o API Gateway.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para JavaScript (v3)

> Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon SNS

Exemplos do Aurora usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Aurora.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um Node.js back-end Express.

- Integre um aplicativo React.js web com Serviços da AWS.
- Liste, adicione e atualize itens em uma tabela do Aurora.
- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos de Auto Scaling usando o SDK for JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AttachLoadBalancerTargetGroups`

O código de exemplo a seguir mostra como usar `AttachLoadBalancerTargetGroups`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new AutoScalingClient({});
await client.send(
  new AttachLoadBalancerTargetGroupsCommand({
    AutoScalingGroupName: NAMES.autoScalingGroupName,
    TargetGroupARNs: [state.targetGroupArn],
  }),
);
```

- Para obter detalhes da API, consulte AttachLoadBalancerTargetGroups em Referência AWS SDK para JavaScript da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
```

```
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Criar etapas para implantar todos os recursos.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
```

```
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
```

```
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
```

```
    new ScenarioOutput(
      "populatingTable",
      MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("populateTable", () => {
      const client = new DynamoDBClient({});
      /**
       * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item'][] }}
       */
      const recommendations = JSON.parse(
        readFileSync(join(RESOURCES_PATH, "recommendations.json")),
      );

      return client.send(
        new BatchWriteItemCommand({
          RequestItems: {
            [NAMES.tableName]: recommendations.map((item) => ({
              PutRequest: { Item: item },
            })),
          },
        }),
      );
    }),
    new ScenarioOutput(
      "populatedTable",
      MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioOutput(
      "creatingKeyPair",
      MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
    ),
    new ScenarioAction("createKeyPair", async () => {
      const client = new EC2Client({});
      const { KeyMaterial } = await client.send(
        new CreateKeyPairCommand({
          KeyName: NAMES.keyPairName,
        }),
      );

      writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
    }),
    new ScenarioOutput(
      "createdKeyPair",
      MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
```

```
    ),
    new ScenarioOutput(
      "creatingInstancePolicy",
      MESSAGES.creatingInstancePolicy.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      ),
    ),
    new ScenarioAction("createInstancePolicy", async (state) => {
      const client = new IAMClient({});
      const {
        Policy: { Arn },
      } = await client.send(
        new CreatePolicyCommand({
          PolicyName: NAMES.instancePolicyName,
          PolicyDocument: readFileSync(
            join(RESOURCES_PATH, "instance_policy.json"),
          ),
        }),
      );
      state.instancePolicyArn = Arn;
    }),
    new ScenarioOutput("createdInstancePolicy", (state) =>
      MESSAGES.createdInstancePolicy
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
    ),
    new ScenarioOutput(
      "creatingInstanceRole",
      MESSAGES.creatingInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      ),
    ),
    new ScenarioAction("createInstanceRole", () => {
      const client = new IAMClient({});
      return client.send(
        new CreateRoleCommand({
          RoleName: NAMES.instanceRoleName,
          AssumeRolePolicyDocument: readFileSync(
            join(ROOT, "assume-role-policy.json"),
          ),
        }),
      );
```

```
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
```

```
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
```

```
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
```

```
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
```

```
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
```

```
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
```

```
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}}
state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
```

```
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
```

```
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
```

```
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Criar etapas para executar a demonstração.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
```

```
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
```

```
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
```

```
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
```

```
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
```

```
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-scaling').Instance }}
state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
```

```
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
```

```
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}}
 state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
```

```
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
```

```
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
```

```
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );
```

```
  return InstanceProfile;
}
```

Criar etapas para destruir todos os recursos.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
```

```
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
```

```
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
```

```
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
```

```
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
```

```
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
```

```
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
```

```
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
```

```
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
```

```
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
```

```
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate

- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Exemplos do Amazon Bedrock usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Bedrock

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  ListFoundationModelsCommand,
} from "@aws-sdk/client-bedrock";

const REGION = "us-east-1";
const client = new BedrockClient({ region: REGION });

export const main = async () => {
  const command = new ListFoundationModelsCommand({});

  const response = await client.send(command);
  const models = response.modelSummaries;

  console.log("Listing the available Bedrock foundation models:");

  for (const model of models) {
    console.log("=".repeat(42));
    console.log(` Model: ${model.modelId}`);
```

```
    console.log("-".repeat(42));
    console.log(` Name: ${model.modelName}`);
    console.log(` Provider: ${model.providerName}`);
    console.log(` Model ARN: ${model.modelArn}`);
    console.log(` Input modalities: ${model.inputModalities}`);
    console.log(` Output modalities: ${model.outputModalities}`);
    console.log(` Supported customizations: ${model.customizationsSupported}`);
    console.log(` Supported inference types: ${model.inferenceTypesSupported}`);
    console.log(` Lifecycle status: ${model.modelLifecycle.status}`);
    console.log(`${"=".repeat(42)}\n`);
  }

  const active = models.filter(
    (m) => m.modelLifecycle.status === "ACTIVE",
  ).length;
  const legacy = models.filter(
    (m) => m.modelLifecycle.status === "LEGACY",
  ).length;

  console.log(
    `There are ${active} active and ${legacy} legacy foundation models in
 ${REGION}.`,
  );

  return response;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await main();
}
```

- Para obter detalhes da API, consulte ListFoundationModels em Referência AWS SDK para JavaScript da API.

Ações

`GetFoundationModel`

O código de exemplo a seguir mostra como usar `GetFoundationModel`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha detalhes de um modelo de base.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  GetFoundationModelCommand,
} from "@aws-sdk/client-bedrock";

/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @return {FoundationModelDetails} - The list of available bedrock foundation
 models.
 */
export const getFoundationModel = async () => {
  const client = new BedrockClient();

  const command = new GetFoundationModelCommand({
    modelIdentifier: "amazon.titan-embed-text-v1",
  });

  const response = await client.send(command);

  return response.modelDetails;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const model = await getFoundationModel();
  console.log(model);
}
```

- Para obter detalhes da API, consulte GetFoundationModel em Referência AWS SDK para JavaScript da API.

`ListFoundationModels`

O código de exemplo a seguir mostra como usar `ListFoundationModels`.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os modelos de base disponíveis.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  ListFoundationModelsCommand,
} from "@aws-sdk/client-bedrock";

/**
 * List the available Amazon Bedrock foundation models.
 *
 * @return {FoundationModelSummary[]} - The list of available bedrock foundation
 models.
 */
export const listFoundationModels = async () => {
  const client = new BedrockClient();

  const input = {
    // byProvider: 'STRING_VALUE',
    // byCustomizationType: 'FINE_TUNING' || 'CONTINUED_PRE_TRAINING',
    // byOutputModality: 'TEXT' || 'IMAGE' || 'EMBEDDING',
    // byInferenceType: 'ON_DEMAND' || 'PROVISIONED',
  };

  const command = new ListFoundationModelsCommand(input);
```

```
  const response = await client.send(command);

  return response.modelSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const models = await listFoundationModels();
  console.log(models);
}
```

- Para obter detalhes da API, consulte ListFoundationModels em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon Bedrock Runtime usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Bedrock Runtime.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Cenários
- Amazônia Nova
- Amazon Nova Canvas
- Claude da Anthropic
- Command da Cohere
- Llama da Meta
- Mistral AI

Conceitos básicos

Olá, Amazon Bedrock Runtime

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock Runtime.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Send a prompt to Amazon Bedrock using the Converse API.

const AWS_REGION = "us-east-1";

// Set the model ID, e.g., Claude Haiku.
// The "global." prefix enables cross-region inference, allowing the request
// to be routed to the nearest available region for the specified model.
const MODEL_ID = "global.anthropic.claude-haiku-4-5-20251001-v1:0";
const PROMPT = "Hi. In a short paragraph, explain what you can do.";

const hello = async () => {
  console.log("=".repeat(35));
  console.log("Welcome to the Amazon Bedrock demo!");
  console.log("=".repeat(35));

  console.log("Model: Anthropic Claude Haiku");
  console.log(`Prompt: ${PROMPT}\n`);
  console.log("Invoking model...\n");

  // Create a new Bedrock Runtime client instance.
  const client = new BedrockRuntimeClient({ region: AWS_REGION });
```

```
  // Create the command with the model ID, the user message, and a basic
 configuration.
  const command = new ConverseCommand({
    modelId: MODEL_ID,
    messages: [
      {
        role: "user",
        content: [{ text: PROMPT }],
      },
    ],
  });

  // Send the command to the model and wait for the response.
  try {
    const response = await client.send(command);

    // Extract and print the response text.
    const responseText = response.output.message.content[0].text;
    console.log(`Response: ${responseText}`);
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "BedrockRuntimeServiceException"
    ) {
      console.error(
        `ERROR: Can't invoke '${MODEL_ID}'. Reason: ${caught.message}`,
      );
      throw caught;
    }
    throw caught;
  }
};

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await hello();
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

Cenários

Uso de ferramenta com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
/* Before running this JavaScript code example, set up your development environment,
 including your credentials.
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";
```

```
import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";
import data from "./questions.json" with { type: "json" };
import toolConfig from "./tool_config.json" with { type: "json" };

const __filename = fileURLToPath(import.meta.url);

const systemPrompt = [
  {
    text:
      "You are a weather assistant that provides current weather data for user-
specified locations using only\n" +
      "the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.\n" +
      "If the user provides coordinates, infer the approximate location and refer to
 it in your response.\n" +
      "To use the tool, you strictly apply the provided tool specification.\n" +
      "If the user specifies a state, country, or region, infer the locations of
 cities within that state.\n" +
      "\n" +
      "- Explain your step-by-step process, and give brief updates before each step.
\n" +
      "- Only use the Weather_Tool for data. Never guess or make up information. \n"
 +
      "- Repeat the tool use for subsequent requests if necessary.\n" +
      "- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.\n" +
      "- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use\n" +
      "  emojis where appropriate.\n" +
      "- Only respond to weather queries. Remind off-topic users of your purpose.
 \n" +
      "- Never claim to search online, access external data, or use tools besides
 Weather_Tool.\n" +
      "- Complete the entire process until you have all required data before sending
 the complete response.",
  },
];
const tools_config = toolConfig;

/// Starts the conversation with the user and handles the interaction with Bedrock.
async function askQuestion(userMessage) {
  // The maximum number of recursive calls allowed in the tool use function.
  // This helps prevent infinite loops and potential performance issues.
```

```
  const max_recursions = 5;
  const messages = [
    {
      role: "user",
      content: [{ text: userMessage }],
    },
  ];
  try {
    const response = await SendConversationtoBedrock(messages);
    await ProcessModelResponseAsync(response, messages, max_recursions);
  } catch (error) {
    console.log("error ", error);
  }
}

// Sends the conversation, the system prompt, and the tool spec to Amazon Bedrock,
 and returns the response.
// param "messages" - The conversation history including the next message to send.
// return - The response from Amazon Bedrock.
async function SendConversationtoBedrock(messages) {
  const bedRockRuntimeClient = new BedrockRuntimeClient({
    region: "us-east-1",
  });
  try {
    const modelId = "amazon.nova-lite-v1:0";
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: messages,
        system: systemPrompt,
        toolConfig: tools_config,
      }),
    );
    return response;
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        "`${caught.name}` - Model not ready, please wait and try again.",
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        '`${caught.name}` - "Error occurred while sending Converse request.',
```

```
      );
      throw caught;
    }
  }
}

// Processes the response received via Amazon Bedrock and performs the necessary
 actions based on the stop reason.
// param "response" - The model's response returned via Amazon Bedrock.
// param "messages" - The conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function ProcessModelResponseAsync(response, messages, max_recursions) {
  if (max_recursions <= 0) {
    await HandleToolUseAsync(response, messages);
  }
  if (response.stopReason === "tool_use") {
    await HandleToolUseAsync(response, messages, max_recursions - 1);
  }
  if (response.stopReason === "end_turn") {
    const messageToPrint = response.output.message.content[0].text;
    console.log(messageToPrint.replace(/<[^>]+>/g, ""));
  }
}
// Handles the tool use case by invoking the specified tool and sending the tool's
 response back to Bedrock.
// The tool response is appended to the conversation, and the conversation is sent
 back to Amazon Bedrock for further processing.
// param "response" - the model's response containing the tool use request.
// param "messages" - the conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function HandleToolUseAsync(response, messages, max_recursions) {
  const toolResultFinal = [];
  try {
    const output_message = response.output.message;
    messages.push(output_message);
    const toolRequests = output_message.content;
    const toolMessage = toolRequests[0].text;
    console.log(toolMessage.replace(/<[^>]+>/g, ""));
    for (const toolRequest of toolRequests) {
      if (Object.hasOwn(toolRequest, "toolUse")) {
        const toolUse = toolRequest.toolUse;
        const latitude = toolUse.input.latitude;
        const longitude = toolUse.input.longitude;
        const toolUseID = toolUse.toolUseId;
```

```
        console.log(
          `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
        );
        if (toolUse.name === "Weather_Tool") {
          try {
            const current_weather = await callWeatherTool(
              longitude,
              latitude,
            ).then((current_weather) => current_weather);
            const currentWeather = current_weather;
            const toolResult = {
              toolResult: {
                toolUseId: toolUseID,
                content: [{ json: currentWeather }],
              },
            };
            toolResultFinal.push(toolResult);
          } catch (err) {
            console.log("An error occurred. ", err);
          }
        }
      }
    }

    const toolResultMessage = {
      role: "user",
      content: toolResultFinal,
    };
    messages.push(toolResultMessage);
    // Send the conversation to Amazon Bedrock
    await ProcessModelResponseAsync(
      await SendConversationtoBedrock(messages),
      messages,
    );
  } catch (error) {
    console.log("An error occurred. ", error);
  }
}
// Call the Weathertool.
// param = longitude of location
// param = latitude of location
async function callWeatherTool(longitude, latitude) {
  // Open-Meteo API endpoint
```

```
  const apiUrl = `https://api.open-meteo.com/v1/forecast?latitude=
${latitude}&longitude=${longitude}&current_weather=true`;

  // Fetch the weather data.
  return fetch(apiUrl)
    .then((response) => {
      return response.json().then((current_weather) => {
        return current_weather;
      });
    })
    .catch((error) => {
      console.error("Error fetching weather data:", error);
    });
}
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  default: "",
});

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Bedrock Tool Use demo! \n" +
    "This assistant provides current weather information for user-specified
 locations. " +
    "You can ask for weather details by providing the location name or coordinates."
 +
    "Weather information will be provided using a custom Tool and open-meteo API." +
    "For the purposes of this example, we'll use in order the questions in ./
questions.json :\n" +
    "What's the weather like in Seattle? " +
    "What's the best kind of cat? " +
    "Where is the warmest city in Washington State right now? " +
    "What's the warmest city in California right now?\n" +
    "To exit the program, simply type 'x' and press Enter.\n" +
    "Have fun and experiment with the app by editing the questions in ./
questions.json! " +
    "P.S.: You're not limited to single locations, or even to using English! ",

  { header: true },
);
```

```
const displayAskQuestion1 = new ScenarioOutput(
  "displayAskQuestion1",
  "Press enter to ask question number 1 (default is 'What's the weather like in
 Seattle?')",
);

const askQuestion1 = new ScenarioAction(
  "askQuestion1",
  async (/** @type {State} */ state) => {
    const userMessage1 = data.questions["question-1"];
    await askQuestion(userMessage1);
  },
);

const displayAskQuestion2 = new ScenarioOutput(
  "displayAskQuestion2",
  "Press enter to ask question number 2 (default is 'What's the best kind of
 cat?')",
);

const askQuestion2 = new ScenarioAction(
  "askQuestion2",
  async (/** @type {State} */ state) => {
    const userMessage2 = data.questions["question-2"];
    await askQuestion(userMessage2);
  },
);
const displayAskQuestion3 = new ScenarioOutput(
  "displayAskQuestion3",
  "Press enter to ask question number 3 (default is 'Where is the warmest city in
 Washington State right now?')",
);

const askQuestion3 = new ScenarioAction(
  "askQuestion3",
  async (/** @type {State} */ state) => {
    const userMessage3 = data.questions["question-3"];
    await askQuestion(userMessage3);
  },
);

const displayAskQuestion4 = new ScenarioOutput(
  "displayAskQuestion4",
```

```
  "Press enter to ask question number 4 (default is 'What's the warmest city in
 California right now?')",
);

const askQuestion4 = new ScenarioAction(
  "askQuestion4",
  async (/** @type {State} */ state) => {
    const userMessage4 = data.questions["question-4"];
    await askQuestion(userMessage4);
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you\n" +
    "learned something new, or got some inspiration for your own apps today!\n" +
    "For more Bedrock examples in different programming languages, have a look at:
\n" +
    "https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html",
);

const myScenario = new Scenario("Converse Tool Scenario", [
  greet,
  pressEnter,
  displayAskQuestion1,
  askQuestion1,
  pressEnter,
  displayAskQuestion2,
  askQuestion2,
  pressEnter,
  displayAskQuestion3,
  askQuestion3,
  pressEnter,
  displayAskQuestion4,
  askQuestion4,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};
```

```
// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

Amazônia Nova

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
// This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Create a message
// - Configure and send a request
// - Process the response
```

```
import {
  BedrockRuntimeClient,
  ConversationRole,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client
// Credentials will be automatically loaded from the environment.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Step 2: Specify which model to use:
// Available Amazon Nova models and their characteristics:
// - Amazon Nova Micro: Text-only model optimized for lowest latency and cost
// - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video, and text
// - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed, and
 cost
//
// For the most current model IDs, see:
// https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
const modelId = "amazon.nova-lite-v1:0";

// Step 3: Create the message
// The message includes the text prompt and specifies that it comes from the user
const inputText =
  "Describe the purpose of a 'hello world' program in one line.";
const message = {
  content: [{ text: inputText }],
  role: ConversationRole.USER,
};

// Step 4: Configure the request
// Optional parameters to control the model's response:
// - maxTokens: maximum number of tokens to generate
// - temperature: randomness (max: 1.0, default: 0.7)
//   OR
// - topP: diversity of word choice (max: 1.0, default: 0.9)
// Note: Use either temperature OR topP, but not both
const request = {
  modelId,
  messages: [message],
  inferenceConfig: {
    maxTokens: 500, // The maximum response length
    temperature: 0.5, // Using temperature for randomness control
    //topP: 0.9,        // Alternative: use topP instead of temperature
```

```
  },
};

// Step 5: Send and process the request
// - Send the request to the model
// - Extract and return the generated text from the response
try {
  const response = await client.send(new ConverseCommand(request));
  console.log(response.output.message.content[0].text);
} catch (error) {
  console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
  throw error;
}
```

Envie uma conversa de mensagens para o Amazon Nova usando a API Converse do Bedrock com uma configuração de ferramenta.

```
// This example demonstrates how to send a conversation of messages to Amazon Nova
 using Bedrock's Converse API with a tool configuration.
// It shows how to:
// - 1. Set up the Amazon Bedrock runtime client
// - 2. Define the parameters required enable Amazon Bedrock to use a tool when
 formulating its response (model ID, user input, system prompt, and the tool spec)
// - 3. Send the request to Amazon Bedrock, and returns the response.
// - 4. Add the tool response to the conversation, and send it back to Amazon
 Bedrock.
// - 5. Publish the response.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client

// Credentials will be automatically loaded from the environment
const bedRockRuntimeClient = new BedrockRuntimeClient({
  region: "us-east-1",
});
```

```
// Step 2. Define the parameters required enable Amazon Bedrock to use a tool when
 formulating its response.

// The Bedrock Model ID.
const modelId = "amazon.nova-lite-v1:0";

// The system prompt to help Amazon Bedrock craft it's response.
const system_prompt = [
  {
    text:
      "You are a music expert that provides the most popular song played on a radio
 station, using only the\n" +
      "the top_song tool, which he call sign for the radio station for which you
 want the most popular song. " +
      "Example calls signs are WZPZ and WKRP. \n" +
      "- Only use the top_song tool. Never guess or make up information. \n" +
      "- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.\n" +
      "- Only respond to queries about the most popular song played on a radio
 station\n" +
      "Remind off-topic users of your purpose. \n" +
      "- Never claim to search online, access external data, or use tools besides
 the top_song tool.\n",
  },
];
//  The user's question.
const message = [
  {
    role: "user",
    content: [{ text: "What is the most popular song on WZPZ?" }],
  },
];
// The tool specification. In this case, it uses an example schema for
// a tool that gets the most popular song played on a radio station.
const tool_config = {
  tools: [
    {
      toolSpec: {
        name: "top_song",
        description: "Get the most popular song played on a radio station.",
        inputSchema: {
          json: {
            type: "object",
            properties: {
```

```
              sign: {
                type: "string",
                description:
                  "The call sign for the radio station for which you want the most
 popular song. Example calls signs are WZPZ and WKRP.",
              },
            },
            required: ["sign"],
          },
        },
      },
    },
  ],
};

// Helper function to return the song and artist from top_song tool.
async function get_top_song(call_sign) {
  try {
    if (call_sign === "WZPZ") {
      const song = "Elemental Hotel";
      const artist = "8 Storey Hike";
      return { song, artist };
    }
  } catch (error) {
    console.log(`${error.message}`);
  }
}

// 3. Send the request to Amazon Bedrock, and returns the response.
export async function SendConversationtoBedrock(
  modelId,
  message,
  system_prompt,
  tool_config,
) {
  try {
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: message,
        system: system_prompt,
        toolConfig: tool_config,
      }),
    );
```

```
    if (response.stopReason === "tool_use") {
      const toolResultFinal = [];
      try {
        const output_message = response.output.message;
        message.push(output_message);
        const toolRequests = output_message.content;
        const toolMessage = toolRequests[0].text;
        console.log(toolMessage.replace(/<[^>]+>/g, ""));
        for (const toolRequest of toolRequests) {
          if (Object.hasOwn(toolRequest, "toolUse")) {
            const toolUse = toolRequest.toolUse;
            const sign = toolUse.input.sign;
            const toolUseID = toolUse.toolUseId;
            console.log(
              `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
            );
            if (toolUse.name === "top_song") {
              const toolResult = [];
              try {
                const top_song = await get_top_song(toolUse.input.sign).then(
                  (top_song) => top_song,
                );
                const toolResult = {
                  toolResult: {
                    toolUseId: toolUseID,
                    content: [
                      {
                        json: { song: top_song.song, artist: top_song.artist },
                      },
                    ],
                  },
                };
                toolResultFinal.push(toolResult);
              } catch (err) {
                const toolResult = {
                  toolUseId: toolUseID,
                  content: [{ json: { text: err.message } }],
                  status: "error",
                };
              }
            }
          }
        }
        const toolResultMessage = {
```

```
          role: "user",
          content: toolResultFinal,
        };
        // Step 4. Add the tool response to the conversation, and send it back to
 Amazon Bedrock.

        message.push(toolResultMessage);
        await SendConversationtoBedrock(
          modelId,
          message,
          system_prompt,
          tool_config,
        );
      } catch (caught) {
        console.error(`${caught.message}`);
        throw caught;
      }
    }

    // 4. Publish the response.
    if (response.stopReason === "end_turn") {
      const finalMessage = response.output.message.content[0].text;
      const messageToPrint = finalMessage.replace(/<[^>]+>/g);
      console.log(messageToPrint.replace(/<[^>]+>/g));
      return messageToPrint;
    }
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        `${caught.name} - Model not ready, please wait and try again.`,
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        `${caught.name} - Error occurred while sending Converse request`,
      );
      throw caught;
    }
  }
}
await SendConversationtoBedrock(modelId, message, system_prompt, tool_config);
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processar o fluxo de respostas em tempo real.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
// This example demonstrates how to use the Amazon Nova foundation models
// to generate streaming text responses.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Create a message
// - Configure a streaming request
// - Process the streaming response

import {
  BedrockRuntimeClient,
  ConversationRole,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client
// Credentials will be automatically loaded from the environment
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Step 2: Specify which model to use
// Available Amazon Nova models and their characteristics:
// - Amazon Nova Micro: Text-only model optimized for lowest latency and cost
// - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video, and text
```

```
// - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed, and
 cost
//
// For the most current model IDs, see:
// https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
const modelId = "amazon.nova-lite-v1:0";

// Step 3: Create the message
// The message includes the text prompt and specifies that it comes from the user
const inputText =
  "Describe the purpose of a 'hello world' program in one paragraph";
const message = {
  content: [{ text: inputText }],
  role: ConversationRole.USER,
};

// Step 4: Configure the streaming request
// Optional parameters to control the model's response:
// - maxTokens: maximum number of tokens to generate
// - temperature: randomness (max: 1.0, default: 0.7)
//   OR
// - topP: diversity of word choice (max: 1.0, default: 0.9)
// Note: Use either temperature OR topP, but not both
const request = {
  modelId,
  messages: [message],
  inferenceConfig: {
    maxTokens: 500, // The maximum response length
    temperature: 0.5, // Using temperature for randomness control
    //topP: 0.9,        // Alternative: use topP instead of temperature
  },
};

// Step 5: Send and process the streaming request
// - Send the request to the model
// - Process each chunk of the streaming response
try {
  const response = await client.send(new ConverseStreamCommand(request));

  for await (const chunk of response.stream) {
    if (chunk.contentBlockDelta) {
      // Print each text chunk as it arrives
      process.stdout.write(chunk.contentBlockDelta.delta?.text || "");
    }
```

```
  }
} catch (error) {
  console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
  process.exitCode = 1;
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para JavaScript da API.

Cenário: uso de ferramentas com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
/* Before running this JavaScript code example, set up your development environment,
 including your credentials.
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.*/

import {
  Scenario,
```

```
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";
import data from "./questions.json" with { type: "json" };
import toolConfig from "./tool_config.json" with { type: "json" };

const __filename = fileURLToPath(import.meta.url);

const systemPrompt = [
  {
    text:
      "You are a weather assistant that provides current weather data for user-
specified locations using only\n" +
      "the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.\n" +
      "If the user provides coordinates, infer the approximate location and refer to
 it in your response.\n" +
      "To use the tool, you strictly apply the provided tool specification.\n" +
      "If the user specifies a state, country, or region, infer the locations of
 cities within that state.\n" +
      "\n" +
      "- Explain your step-by-step process, and give brief updates before each step.
\n" +
      "- Only use the Weather_Tool for data. Never guess or make up information. \n"
 +
      "- Repeat the tool use for subsequent requests if necessary.\n" +
      "- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.\n" +
      "- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use\n" +
      "  emojis where appropriate.\n" +
      "- Only respond to weather queries. Remind off-topic users of your purpose.
 \n" +
      "- Never claim to search online, access external data, or use tools besides
 Weather_Tool.\n" +
```

```
      "- Complete the entire process until you have all required data before sending
 the complete response.",
  },
];
const tools_config = toolConfig;

/// Starts the conversation with the user and handles the interaction with Bedrock.
async function askQuestion(userMessage) {
  // The maximum number of recursive calls allowed in the tool use function.
  // This helps prevent infinite loops and potential performance issues.
  const max_recursions = 5;
  const messages = [
    {
      role: "user",
      content: [{ text: userMessage }],
    },
  ];
  try {
    const response = await SendConversationtoBedrock(messages);
    await ProcessModelResponseAsync(response, messages, max_recursions);
  } catch (error) {
    console.log("error ", error);
  }
}

// Sends the conversation, the system prompt, and the tool spec to Amazon Bedrock,
 and returns the response.
// param "messages" - The conversation history including the next message to send.
// return - The response from Amazon Bedrock.
async function SendConversationtoBedrock(messages) {
  const bedRockRuntimeClient = new BedrockRuntimeClient({
    region: "us-east-1",
  });
  try {
    const modelId = "amazon.nova-lite-v1:0";
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: messages,
        system: systemPrompt,
        toolConfig: tools_config,
      }),
    );
    return response;
```

```
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        "`${caught.name}` - Model not ready, please wait and try again.",
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        '`${caught.name}` - "Error occurred while sending Converse request.',
      );
      throw caught;
    }
  }
}

// Processes the response received via Amazon Bedrock and performs the necessary
 actions based on the stop reason.
// param "response" - The model's response returned via Amazon Bedrock.
// param "messages" - The conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function ProcessModelResponseAsync(response, messages, max_recursions) {
  if (max_recursions <= 0) {
    await HandleToolUseAsync(response, messages);
  }
  if (response.stopReason === "tool_use") {
    await HandleToolUseAsync(response, messages, max_recursions - 1);
  }
  if (response.stopReason === "end_turn") {
    const messageToPrint = response.output.message.content[0].text;
    console.log(messageToPrint.replace(/<[^>]+>/g, ""));
  }
}
// Handles the tool use case by invoking the specified tool and sending the tool's
 response back to Bedrock.
// The tool response is appended to the conversation, and the conversation is sent
 back to Amazon Bedrock for further processing.
// param "response" - the model's response containing the tool use request.
// param "messages" - the conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function HandleToolUseAsync(response, messages, max_recursions) {
  const toolResultFinal = [];
  try {
    const output_message = response.output.message;
```

```
    messages.push(output_message);
    const toolRequests = output_message.content;
    const toolMessage = toolRequests[0].text;
    console.log(toolMessage.replace(/<[^>]+>/g, ""));
    for (const toolRequest of toolRequests) {
      if (Object.hasOwn(toolRequest, "toolUse")) {
        const toolUse = toolRequest.toolUse;
        const latitude = toolUse.input.latitude;
        const longitude = toolUse.input.longitude;
        const toolUseID = toolUse.toolUseId;
        console.log(
          `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
        );
        if (toolUse.name === "Weather_Tool") {
          try {
            const current_weather = await callWeatherTool(
              longitude,
              latitude,
            ).then((current_weather) => current_weather);
            const currentWeather = current_weather;
            const toolResult = {
              toolResult: {
                toolUseId: toolUseID,
                content: [{ json: currentWeather }],
              },
            };
            toolResultFinal.push(toolResult);
          } catch (err) {
            console.log("An error occurred. ", err);
          }
        }
      }
    }

    const toolResultMessage = {
      role: "user",
      content: toolResultFinal,
    };
    messages.push(toolResultMessage);
    // Send the conversation to Amazon Bedrock
    await ProcessModelResponseAsync(
      await SendConversationtoBedrock(messages),
      messages,
    );
```

```
  } catch (error) {
    console.log("An error occurred. ", error);
  }
}
// Call the Weathertool.
// param = longitude of location
// param = latitude of location
async function callWeatherTool(longitude, latitude) {
  // Open-Meteo API endpoint
  const apiUrl = `https://api.open-meteo.com/v1/forecast?latitude=
${latitude}&longitude=${longitude}&current_weather=true`;

  // Fetch the weather data.
  return fetch(apiUrl)
    .then((response) => {
      return response.json().then((current_weather) => {
        return current_weather;
      });
    })
    .catch((error) => {
      console.error("Error fetching weather data:", error);
    });
}
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  default: "",
});

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Bedrock Tool Use demo! \n" +
    "This assistant provides current weather information for user-specified
 locations. " +
    "You can ask for weather details by providing the location name or coordinates."
 +
    "Weather information will be provided using a custom Tool and open-meteo API." +
    "For the purposes of this example, we'll use in order the questions in ./
questions.json :\n" +
    "What's the weather like in Seattle? " +
    "What's the best kind of cat? " +
```

```
    "Where is the warmest city in Washington State right now? " +
    "What's the warmest city in California right now?\n" +
    "To exit the program, simply type 'x' and press Enter.\n" +
    "Have fun and experiment with the app by editing the questions in ./
questions.json! " +
    "P.S.: You're not limited to single locations, or even to using English! ",

  { header: true },
);
const displayAskQuestion1 = new ScenarioOutput(
  "displayAskQuestion1",
  "Press enter to ask question number 1 (default is 'What's the weather like in
 Seattle?')",
);

const askQuestion1 = new ScenarioAction(
  "askQuestion1",
  async (/** @type {State} */ state) => {
    const userMessage1 = data.questions["question-1"];
    await askQuestion(userMessage1);
  },
);

const displayAskQuestion2 = new ScenarioOutput(
  "displayAskQuestion2",
  "Press enter to ask question number 2 (default is 'What's the best kind of
 cat?')",
);

const askQuestion2 = new ScenarioAction(
  "askQuestion2",
  async (/** @type {State} */ state) => {
    const userMessage2 = data.questions["question-2"];
    await askQuestion(userMessage2);
  },
);
const displayAskQuestion3 = new ScenarioOutput(
  "displayAskQuestion3",
  "Press enter to ask question number 3 (default is 'Where is the warmest city in
 Washington State right now?')",
);

const askQuestion3 = new ScenarioAction(
  "askQuestion3",
```

```
  async (/** @type {State} */ state) => {
    const userMessage3 = data.questions["question-3"];
    await askQuestion(userMessage3);
  },
);

const displayAskQuestion4 = new ScenarioOutput(
  "displayAskQuestion4",
  "Press enter to ask question number 4 (default is 'What's the warmest city in
 California right now?')",
);

const askQuestion4 = new ScenarioAction(
  "askQuestion4",
  async (/** @type {State} */ state) => {
    const userMessage4 = data.questions["question-4"];
    await askQuestion(userMessage4);
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you\n" +
    "learned something new, or got some inspiration for your own apps today!\n" +
    "For more Bedrock examples in different programming languages, have a look at:
\n" +
    "https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html",
);

const myScenario = new Scenario("Converse Tool Scenario", [
  greet,
  pressEnter,
  displayAskQuestion1,
  askQuestion1,
  pressEnter,
  displayAskQuestion2,
  askQuestion2,
  pressEnter,
  displayAskQuestion3,
  askQuestion3,
  pressEnter,
  displayAskQuestion4,
  askQuestion4,
```

```
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

Amazon Nova Canvas

InvokeModel

O exemplo de código a seguir mostra como invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
import {
  BedrockRuntimeClient,
  InvokeModelCommand,
} from "@aws-sdk/client-bedrock-runtime";
import { saveImage } from "../../utils/image-creation.js";
import { fileURLToPath } from "node:url";

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 *
 * @returns {Promise<string>} Base64-encoded image data
 */
export const invokeModel = async () => {
  // Step 1: Create the Amazon Bedrock runtime client
  // Credentials will be automatically loaded from the environment
  const client = new BedrockRuntimeClient({ region: "us-east-1" });

  // Step 2: Specify which model to use
  // For the latest available models, see:
  // https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
  const modelId = "amazon.nova-canvas-v1:0";

  // Step 3: Configure the request payload
  // First, set the main parameters:
  // - prompt: Text description of the image to generate
  // - seed: Random number for reproducible generation (0 to 858,993,459)
  const prompt = "A stylized picture of a cute old steampunk robot";
  const seed = Math.floor(Math.random() * 858993460);

  // Then, create the payload using the following structure:
  // - taskType: TEXT_IMAGE (specifies text-to-image generation)
  // - textToImageParams: Contains the text prompt
  // - imageGenerationConfig: Contains optional generation settings (seed, quality,
 etc.)
  // For a list of available request parameters, see:
  // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
```

```
  const payload = {
    taskType: "TEXT_IMAGE",
    textToImageParams: {
      text: prompt,
    },
    imageGenerationConfig: {
      seed,
      quality: "standard",
    },
  };

  // Step 4: Send and process the request
  // - Embed the payload in a request object
  // - Send the request to the model
  // - Extract and return the generated image data from the response
  try {
    const request = {
      modelId,
      body: JSON.stringify(payload),
    };
    const response = await client.send(new InvokeModelCommand(request));

    const decodedResponseBody = new TextDecoder().decode(response.body);
    // The response includes an array of base64-encoded PNG images
    /** @type {{images: string[]}} */
    const responseBody = JSON.parse(decodedResponseBody);
    return responseBody.images[0]; // Base64-encoded image data
  } catch (error) {
    console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
    throw error;
  }
};

// If run directly, execute the example and save the generated image
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  console.log("Generating image. This may take a few seconds...");
  invokeModel()
    .then(async (imageData) => {
      const imagePath = await saveImage(imageData, "nova-canvas");
      // Example path: javascriptv3/example_code/bedrock-runtime/output/nova-canvas/
image-01.png
      console.log(`Image saved to: ${imagePath}`);
    })
    .catch((error) => {
```

```
      console.error("Execution failed:", error);
      process.exitCode = 1;
    });
}
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK para JavaScript da API.

Claude da Anthropic

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Anthropic Claude.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Claude 3 Haiku.
const modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
```

```
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Conversation API to send a text message to Anthropic Claude.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Claude 3 Haiku.
const modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
```

```
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para JavaScript da API.

Command da Cohere

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Cohere Command usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Cohere Command.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Command R.
const modelId = "cohere.command-r-v1:0";

// Start a conversation with the user message.
```

```
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Conversation API to send a text message to Cohere Command.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Command R.
const modelId = "cohere.command-r-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
```

```
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para JavaScript da API.

Llama da Meta

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Meta Llama.

import {
  BedrockRuntimeClient,
  ConverseCommand,
```

```
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Llama 3 8b Instruct.
const modelId = "meta.llama3-8b-instruct-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para JavaScript .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Conversation API to send a text message to Meta Llama.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Llama 3 8b Instruct.
const modelId = "meta.llama3-8b-instruct-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
```

```
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para JavaScript da API.

Mistral AI

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Mistral.
```

```
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Mistral Large.
const modelId = "mistral.mistral-large-2402-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK para JavaScript .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para a Mistral usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Conversation API to send a text message to Mistral.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Mistral Large.
const modelId = "mistral.mistral-large-2402-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];
```

```
// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK para JavaScript da API.

Exemplos de Amazon Bedrock Agents usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com os agentes Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Bedrock Agents

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock Agents.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentClient,
  GetAgentCommand,
  paginateListAgents,
} from "@aws-sdk/client-bedrock-agent";

/**
 * @typedef {Object} AgentSummary
 */

/**
 * A simple scenario to demonstrate basic setup and interaction with the Bedrock
 Agents Client.
 *
 * This function first initializes the Amazon Bedrock Agents client for a specific
 region.
 * It then retrieves a list of existing agents using the streamlined paginator
 approach.
 * For each agent found, it retrieves detailed information using a command object.
 *
 * Demonstrates:
```

```
 * - Use of the Bedrock Agents client to initialize and communicate with the AWS
 service.
 * - Listing resources in a paginated response pattern.
 * - Accessing an individual resource using a command object.
 *
 * @returns {Promise<void>} A promise that resolves when the function has completed
 execution.
 */
export const main = async () => {
  const region = "us-east-1";

  console.log("=".repeat(68));

  console.log(`Initializing Amazon Bedrock Agents client for ${region}...`);
  const client = new BedrockAgentClient({ region });

  console.log("Retrieving the list of existing agents...");
  const paginatorConfig = { client };
  const pages = paginateListAgents(paginatorConfig, {});

  /** @type {AgentSummary[]} */
  const agentSummaries = [];
  for await (const page of pages) {
    agentSummaries.push(...page.agentSummaries);
  }

  console.log(`Found ${agentSummaries.length} agents in ${region}.`);

  if (agentSummaries.length > 0) {
    for (const agentSummary of agentSummaries) {
      const agentId = agentSummary.agentId;
      console.log("=".repeat(68));
      console.log(`Retrieving agent with ID: ${agentId}:`);
      console.log("-".repeat(68));

      const command = new GetAgentCommand({ agentId });
      const response = await client.send(command);
      const agent = response.agent;

      console.log(` Name: ${agent.agentName}`);
      console.log(` Status: ${agent.agentStatus}`);
      console.log(` ARN: ${agent.agentArn}`);
      console.log(` Foundation model: ${agent.foundationModel}`);
    }
```

```
  }
  console.log("=".repeat(68));
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await main();
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - GetAgent
  - ListAgents

Ações

`CreateAgent`

O código de exemplo a seguir mostra como usar `CreateAgent`.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um agente do .

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  CreateAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
```

```
 * Creates an Amazon Bedrock Agent.
 *
 * @param {string} agentName - A name for the agent that you create.
 * @param {string} foundationModel - The foundation model to be used by the agent
 you create.
 * @param {string} agentResourceRoleArn - The ARN of the IAM role with permissions
 required by the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").Agent>} An object
 containing details of the created agent.
 */
export const createAgent = async (
  agentName,
  foundationModel,
  agentResourceRoleArn,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  const command = new CreateAgentCommand({
    agentName,
    foundationModel,
    agentResourceRoleArn,
  });
  const response = await client.send(command);

  return response.agent;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentName and accountId, and roleName with a
 unique name for the new agent,
  // the id of your AWS account, and the name of an existing execution role that the
 agent can use inside your account.
  // For foundationModel, specify the desired model. Ensure to remove the brackets
 '[]' before adding your data.

  // A string (max 100 chars) that can include letters, numbers, dashes '-', and
 underscores '_'.
  const agentName = "[your-bedrock-agent-name]";

  // Your AWS account id.
  const accountId = "[123456789012]";
```

```
  // The name of the agent's execution role. It must be prefixed by
 `AmazonBedrockExecutionRoleForAgents_`.
  const roleName = "[AmazonBedrockExecutionRoleForAgents_your-role-name]";

  // The ARN for the agent's execution role.
  // Follow the ARN format: 'arn:aws:iam::account-id:role/role-name'
  const roleArn = `arn:aws:iam::${accountId}:role/${roleName}`;

  // Specify the model for the agent. Change if a different model is preferred.
  const foundationModel = "anthropic.claude-v2";

  // Check for unresolved placeholders in agentName and roleArn.
  checkForPlaceholders([agentName, roleArn]);

  console.log("Creating a new agent...");

  const agent = await createAgent(agentName, foundationModel, roleArn);
  console.log(agent);
}
```

- Para obter detalhes da API, consulte CreateAgent em Referência AWS SDK para JavaScript da API.

`DeleteAgent`

O código de exemplo a seguir mostra como usar `DeleteAgent`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um agente.

```
import { fileURLToPath } from "node:url";
```

```
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  DeleteAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Deletes an Amazon Bedrock Agent.
 *
 * @param {string} agentId - The unique identifier of the agent to delete.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-
agent").DeleteAgentCommandOutput>} An object containing the agent id, the status,
 and some additional metadata.
 */
export const deleteAgent = (agentId, region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });
  const command = new DeleteAgentCommand({ agentId });
  return client.send(command);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId with an existing agent's id.
  // Ensure to remove the brackets (`[]`) before adding your data.

  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // Check for unresolved placeholders in agentId.
  checkForPlaceholders([agentId]);

  console.log(`Deleting agent with ID ${agentId}...`);

  const response = await deleteAgent(agentId);
  console.log(response);
}
```

- Para obter detalhes da API, consulte DeleteAgent em Referência AWS SDK para JavaScript da API.

GetAgent

O código de exemplo a seguir mostra como usar `GetAgent`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um agente.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  GetAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves the details of an Amazon Bedrock Agent.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").Agent>} An object
 containing the agent details.
 */
export const getAgent = async (agentId, region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  const command = new GetAgentCommand({ agentId });
  const response = await client.send(command);
  return response.agent;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId with an existing agent's id.
  // Ensure to remove the brackets '[]' before adding your data.
```

```
  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // Check for unresolved placeholders in agentId.
  checkForPlaceholders([agentId]);

  console.log(`Retrieving agent with ID ${agentId}...`);

  const agent = await getAgent(agentId);
  console.log(agent);
}
```

- Para obter detalhes da API, consulte GetAgent em Referência AWS SDK para JavaScript da API.

`ListAgentActionGroups`

O código de exemplo a seguir mostra como usar `ListAgentActionGroups`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os grupos de ação de um agente.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  ListAgentActionGroupsCommand,
  paginateListAgentActionGroups,
} from "@aws-sdk/client-bedrock-agent";

/**
```

```
 * Retrieves a list of Action Groups of an agent utilizing the paginator function.
 *
 * This function leverages a paginator, which abstracts the complexity of
 pagination, providing
 * a straightforward way to handle paginated results inside a `for await...of` loop.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} agentVersion - The version of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<ActionGroupSummary[]>} An array of action group summaries.
 */
export const listAgentActionGroupsWithPaginator = async (
  agentId,
  agentVersion,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  // Create a paginator configuration
  const paginatorConfig = {
    client,
    pageSize: 10, // optional, added for demonstration purposes
  };

  const params = { agentId, agentVersion };

  const pages = paginateListAgentActionGroups(paginatorConfig, params);

  // Paginate until there are no more results
  const actionGroupSummaries = [];
  for await (const page of pages) {
    actionGroupSummaries.push(...page.actionGroupSummaries);
  }

  return actionGroupSummaries;
};

/**
 * Retrieves a list of Action Groups of an agent utilizing the
 ListAgentActionGroupsCommand.
 *
 * This function demonstrates the manual approach, sending a command to the client
 and processing the response.
```

```
 * Pagination must manually be managed. For a simplified approach that abstracts
 away pagination logic, see
 * the `listAgentActionGroupsWithPaginator()` example below.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} agentVersion - The version of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<ActionGroupSummary[]>} An array of action group summaries.
 */
export const listAgentActionGroupsWithCommandObject = async (
  agentId,
  agentVersion,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  let nextToken;
  const actionGroupSummaries = [];
  do {
    const command = new ListAgentActionGroupsCommand({
      agentId,
      agentVersion,
      nextToken,
      maxResults: 10, // optional, added for demonstration purposes
    });

    /** @type {{actionGroupSummaries: ActionGroupSummary[], nextToken?: string}} */
    const response = await client.send(command);

    for (const actionGroup of response.actionGroupSummaries || []) {
      actionGroupSummaries.push(actionGroup);
    }

    nextToken = response.nextToken;
  } while (nextToken);

  return actionGroupSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId and agentVersion with an existing agent's
 id and version.
  // Ensure to remove the brackets '[]' before adding your data.
```

```
  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // A string either containing `DRAFT` or a number with 1-5 digits (e.g., '123' or
 'DRAFT').
  const agentVersion = "[DRAFT]";

  // Check for unresolved placeholders in agentId and agentVersion.
  checkForPlaceholders([agentId, agentVersion]);

  console.log("=".repeat(68));
  console.log(
    "Listing agent action groups using ListAgentActionGroupsCommand:",
  );

  for (const actionGroup of await listAgentActionGroupsWithCommandObject(
    agentId,
    agentVersion,
  )) {
    console.log(actionGroup);
  }

  console.log("=".repeat(68));
  console.log(
    "Listing agent action groups using the paginateListAgents function:",
  );
  for (const actionGroup of await listAgentActionGroupsWithPaginator(
    agentId,
    agentVersion,
  )) {
    console.log(actionGroup);
  }
}
```

- Para obter detalhes da API, consulte ListAgentActionGroups em Referência AWS SDK para JavaScript da API.

`ListAgents`

O código de exemplo a seguir mostra como usar `ListAgents`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os agentes que pertencem a uma conta.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentClient,
  ListAgentsCommand,
  paginateListAgents,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves a list of available Amazon Bedrock agents utilizing the paginator
 function.
 *
 * This function leverages a paginator, which abstracts the complexity of
 pagination, providing
 * a straightforward way to handle paginated results inside a `for await...of` loop.
 *
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<AgentSummary[]>} An array of agent summaries.
 */
export const listAgentsWithPaginator = async (region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  const paginatorConfig = {
    client,
    pageSize: 10, // optional, added for demonstration purposes
  };

  const pages = paginateListAgents(paginatorConfig, {});

  // Paginate until there are no more results
  const agentSummaries = [];
  for await (const page of pages) {
```

```
    agentSummaries.push(...page.agentSummaries);
  }

  return agentSummaries;
};

/**
 * Retrieves a list of available Amazon Bedrock agents utilizing the
 ListAgentsCommand.
 *
 * This function demonstrates the manual approach, sending a command to the client
 and processing the response.
 * Pagination must manually be managed. For a simplified approach that abstracts
 away pagination logic, see
 * the `listAgentsWithPaginator()` example below.
 *
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<AgentSummary[]>} An array of agent summaries.
 */
export const listAgentsWithCommandObject = async (region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  let nextToken;
  const agentSummaries = [];
  do {
    const command = new ListAgentsCommand({
      nextToken,
      maxResults: 10, // optional, added for demonstration purposes
    });

    /** @type {{agentSummaries: AgentSummary[], nextToken?: string}} */
    const paginatedResponse = await client.send(command);

    agentSummaries.push(...(paginatedResponse.agentSummaries || []));

    nextToken = paginatedResponse.nextToken;
  } while (nextToken);

  return agentSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  console.log("=".repeat(68));
```

```
  console.log("Listing agents using ListAgentsCommand:");
  for (const agent of await listAgentsWithCommandObject()) {
    console.log(agent);
  }

  console.log("=".repeat(68));
  console.log("Listing agents using the paginateListAgents function:");
  for (const agent of await listAgentsWithPaginator()) {
    console.log(agent);
  }
}
```

- Para obter detalhes da API, consulte ListAgents em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon Bedrock Agents Runtime usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Bedrock Agents Runtime.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`InvokeAgent`

O código de exemplo a seguir mostra como usar `InvokeAgent`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  BedrockAgentRuntimeClient,
  InvokeAgentCommand,
} from "@aws-sdk/client-bedrock-agent-runtime";

/**
 * @typedef {Object} ResponseBody
 * @property {string} completion
 */

/**
 * Invokes a Bedrock agent to run an inference using the input
 * provided in the request body.
 *
 * @param {string} prompt - The prompt that you want the Agent to complete.
 * @param {string} sessionId - An arbitrary identifier for the session.
 */
export const invokeBedrockAgent = async (prompt, sessionId) => {
  const client = new BedrockAgentRuntimeClient({ region: "us-east-1" });
  // const client = new BedrockAgentRuntimeClient({
  //   region: "us-east-1",
  //   credentials: {
  //     accessKeyId: "accessKeyId", // permission to invoke agent
  //     secretAccessKey: "accessKeySecret",
  //   },
  // });

  const agentId = "AJBHXXILZN";
  const agentAliasId = "AVKP1ITZAA";

  const command = new InvokeAgentCommand({
    agentId,
    agentAliasId,
```

```
    sessionId,
    inputText: prompt,
  });

  try {
    let completion = "";
    const response = await client.send(command);

    if (response.completion === undefined) {
      throw new Error("Completion is undefined");
    }

    for await (const chunkEvent of response.completion) {
      const chunk = chunkEvent.chunk;
      console.log(chunk);
      const decodedResponse = new TextDecoder("utf-8").decode(chunk.bytes);
      completion += decodedResponse;
    }

    return { sessionId: sessionId, completion };
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const result = await invokeBedrockAgent("I need help.", "123");
  console.log(result);
}
```

- Para obter detalhes da API, consulte InvokeAgent em Referência AWS SDK para JavaScript da API.

`InvokeFlow`

O código de exemplo a seguir mostra como usar `InvokeFlow`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentRuntimeClient,
  InvokeFlowCommand,
} from "@aws-sdk/client-bedrock-agent-runtime";

/**
 * Invokes an alias of a flow to run the inputs that you specify and return
 * the output of each node as a stream.
 *
 * @param {{
 *  flowIdentifier: string,
 *  flowAliasIdentifier: string,
 *  prompt?: string,
 *  region?: string
 * }} options
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").FlowNodeOutput>} An
 object containing information about the output from flow invocation.
 */
export const invokeBedrockFlow = async ({
  flowIdentifier,
  flowAliasIdentifier,
  prompt = "Hi, how are you?",
  region = "us-east-1",
}) => {
  const client = new BedrockAgentRuntimeClient({ region });

  const command = new InvokeFlowCommand({
    flowIdentifier,
    flowAliasIdentifier,
    inputs: [
      {
```

```
        content: {
          document: prompt,
        },
        nodeName: "FlowInputNode",
        nodeOutputName: "document",
      },
    ],
  });

  let flowResponse = {};
  const response = await client.send(command);

  for await (const chunkEvent of response.responseStream) {
    const { flowOutputEvent, flowCompletionEvent } = chunkEvent;

    if (flowOutputEvent) {
      flowResponse = { ...flowResponse, ...flowOutputEvent };
      console.log("Flow output event:", flowOutputEvent);
    } else if (flowCompletionEvent) {
      flowResponse = { ...flowResponse, ...flowCompletionEvent };
      console.log("Flow completion event:", flowCompletionEvent);
    }
  }

  return flowResponse;
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    flowIdentifier: {
      type: "string",
      required: true,
    },
    flowAliasIdentifier: {
      type: "string",
      required: true,
    },
```

```
    prompt: {
      type: "string",
    },
    region: {
      type: "string",
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    invokeBedrockFlow(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte InvokeFlow em Referência AWS SDK para JavaScript da API.

CloudWatch exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteAlarms`

O código de exemplo a seguir mostra como usar `DeleteAlarms`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAlarms em Referência AWS SDK para JavaScript da API.

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DescribeAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
```

```
export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeAlarmsForMetric em Referência AWS SDK para JavaScript da API.

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DisableAlarmActionsCommand({
    AlarmNames: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DisableAlarmActions em Referência AWS SDK para JavaScript da API.

`EnableAlarmActions`

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { EnableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new EnableAlarmActionsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};
```

```
export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte EnableAlarmActions em Referência AWS SDK para JavaScript da API.

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import {
  CloudWatchServiceException,
  ListMetricsCommand,
} from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

export const main = async () => {
  // Use the AWS console to see available namespaces and metric names. Custom
 metrics can also be created.
  // https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
viewing_metrics_with_cloudwatch.html
```

```
  const command = new ListMetricsCommand({
    Dimensions: [
      {
        Name: "LogGroupName",
      },
    ],
    MetricName: "IncomingLogEvents",
    Namespace: "AWS/Logs",
  });

  try {
    const response = await client.send(command);
    console.log(`Metrics count: ${response.Metrics?.length}`);
    return response;
  } catch (caught) {
    if (caught instanceof CloudWatchServiceException) {
      console.error(`Error from CloudWatch. ${caught.name}: ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListMetrics em Referência AWS SDK para JavaScript da API.

`PutMetricAlarm`

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // This alarm triggers when CPUUtilization exceeds 70% for one minute.
  const command = new PutMetricAlarmCommand({
    AlarmName: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
    ComparisonOperator: "GreaterThanThreshold",
    EvaluationPeriods: 1,
    MetricName: "CPUUtilization",
    Namespace: "AWS/EC2",
    Period: 60,
    Statistic: "Average",
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: "Alarm when server CPU exceeds 70%",
    Dimensions: [
      {
        Name: "InstanceId",
        Value: process.env.EC2_INSTANCE_ID, // Set the value of EC_INSTANCE_ID to
 the Id of an existing Amazon EC2 instance.
      },
    ],
    Unit: "Percent",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};
```

```
export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutMetricAlarm em Referência AWS SDK para JavaScript da API.

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // See https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/
API_PutMetricData.html#API_PutMetricData_RequestParameters
  // and https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
publishingMetrics.html
  // for more information about the parameters in this command.
  const command = new PutMetricDataCommand({
    MetricData: [
```

```
      {
        MetricName: "PAGES_VISITED",
        Dimensions: [
          {
            Name: "UNIQUE_PAGES",
            Value: "URLS",
          },
        ],
        Unit: "None",
        Value: 1.0,
      },
    ],
    Namespace: "SITE/TRAFFIC",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutMetricData em Referência AWS SDK para JavaScript da API.

CloudWatch Exemplos de eventos usando o SDK for JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com CloudWatch Eventos.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutEventsCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutEventsCommand({
    // The list of events to send to Amazon CloudWatch Events.
    Entries: [
      {
        // The name of the application or service that is sending the event.
        Source: "my.app",

        // The name of the event that is being sent.
        DetailType: "My Custom Event",

        // The data that is sent with the event.
```

```
        Detail: JSON.stringify({ timeOfEvent: new Date().toISOString() }),
      },
    ],
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK para JavaScript da API.

`PutRule`

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutRuleCommand } from "@aws-sdk/client-cloudwatch-events";
```

```
import { client } from "../libs/client.js";

const run = async () => {
  // Request parameters for PutRule.
  // https://docs.aws.amazon.com/eventbridge/latest/APIReference/
API_PutRule.html#API_PutRule_RequestParameters
  const command = new PutRuleCommand({
    Name: process.env.CLOUDWATCH_EVENTS_RULE,

    // The event pattern for the rule.
    //  Example: {"source": ["my.app"]}
    EventPattern: process.env.CLOUDWATCH_EVENTS_RULE_PATTERN,

    // The state of the rule. Valid values: ENABLED, DISABLED
    State: "ENABLED",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutRule em Referência AWS SDK para JavaScript da API.

`PutTargets`

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { PutTargetsCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutTargetsCommand({
    // The name of the Amazon CloudWatch Events rule.
    Rule: process.env.CLOUDWATCH_EVENTS_RULE,

    // The targets to add to the rule.
    Targets: [
      {
        Arn: process.env.CLOUDWATCH_EVENTS_TARGET_ARN,
        // The ID of the target. Choose a unique ID for each target.
        Id: process.env.CLOUDWATCH_EVENTS_TARGET_ID,
      },
    ],
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Crie o cliente em um módulo separado e exporte-o.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
```

```
export const client = new CloudWatchEventsClient({});
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutTargets em Referência AWS SDK para JavaScript da API.

CloudWatch Exemplos de registros usando o SDK for JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com CloudWatch registros.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateLogGroup`

O código de exemplo a seguir mostra como usar `CreateLogGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new CreateLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte CreateLogGroup em Referência AWS SDK para JavaScript da API.

`DeleteLogGroup`

O código de exemplo a seguir mostra como usar `DeleteLogGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
```

```
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte DeleteLogGroup em Referência AWS SDK para JavaScript da API.

`DeleteSubscriptionFilter`

O código de exemplo a seguir mostra como usar `DeleteSubscriptionFilter`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteSubscriptionFilterCommand({
    // The name of the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
```

```
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte DeleteSubscriptionFilter em Referência AWS SDK para JavaScript da API.

`DescribeLogGroups`

O código de exemplo a seguir mostra como usar `DescribeLogGroups`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  paginateDescribeLogGroups,
  CloudWatchLogsClient,
} from "@aws-sdk/client-cloudwatch-logs";

const client = new CloudWatchLogsClient({});

export const main = async () => {
  const paginatedLogGroups = paginateDescribeLogGroups({ client }, {});
  const logGroups = [];

  for await (const page of paginatedLogGroups) {
    if (page.logGroups?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
```

```
};
```

- Para obter detalhes da API, consulte DescribeLogGroups em Referência AWS SDK para JavaScript da API.

`DescribeSubscriptionFilters`

O código de exemplo a seguir mostra como usar `DescribeSubscriptionFilters`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  // This will return a list of all subscription filters in your account
  // matching the log group name.
  const command = new DescribeSubscriptionFiltersCommand({
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
    limit: 1,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte DescribeSubscriptionFilters em Referência AWS SDK para JavaScript da API.

GetQueryResults

O código de exemplo a seguir mostra como usar `GetQueryResults`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
  return this.client.send(new GetQueryResultsCommand({ queryId }));
}
```

- Para obter detalhes da API, consulte GetQueryResults em Referência AWS SDK para JavaScript da API.

PutSubscriptionFilter

O código de exemplo a seguir mostra como usar `PutSubscriptionFilter`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { PutSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutSubscriptionFilterCommand({
    // An ARN of a same-account Kinesis stream, Kinesis Firehose
    // delivery stream, or Lambda function.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
SubscriptionFilters.html
    destinationArn: process.env.CLOUDWATCH_LOGS_DESTINATION_ARN,

    // A name for the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,

    // A filter pattern for subscribing to a filtered stream of log events.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
FilterAndPatternSyntax.html
    filterPattern: process.env.CLOUDWATCH_LOGS_FILTER_PATTERN,

    // The name of the log group. Messages in this group matching the filter pattern
    // will be sent to the destination ARN.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Para obter detalhes da API, consulte PutSubscriptionFilter em Referência AWS SDK para JavaScript da API.

StartLiveTail

O código de exemplo a seguir mostra como usar `StartLiveTail`.

SDK para JavaScript (v3)

Inclua os arquivos necessários.

```
import { CloudWatchLogsClient, StartLiveTailCommand } from "@aws-sdk/client-
cloudwatch-logs";
```

Gerencie os eventos da sessão do Live Tail.

```
async function handleResponseAsync(response) {
    try {
      for await (const event of response.responseStream) {
        if (event.sessionStart !== undefined) {
          console.log(event.sessionStart);
        } else if (event.sessionUpdate !== undefined) {
          for (const logEvent of event.sessionUpdate.sessionResults) {
            const timestamp = logEvent.timestamp;
            const date = new Date(timestamp);
            console.log("[" + date + "] " + logEvent.message);
          }
        } else {
            console.error("Unknown event type");
        }
      }
    } catch (err) {
        // On-stream exceptions are captured here
        console.error(err)
    }
}
```

Inicie a sessão do Live Tail.

```
    const client = new CloudWatchLogsClient();

    const command = new StartLiveTailCommand({
        logGroupIdentifiers: logGroupIdentifiers,
        logStreamNames: logStreamNames,
        logEventFilterPattern: filterPattern
    });
    try{
        const response = await client.send(command);
```

```
        handleResponseAsync(response);
    } catch (err){
        // Pre-stream exceptions are captured here
        console.log(err);
    }
```

Interrompa a sessão do Live Tail após um período decorrido.

```
    /* Set a timeout to close the client. This will stop the Live Tail session. */
    setTimeout(function() {
        console.log("Client timeout");
        client.destroy();
      }, 10000);
```

- Para obter detalhes da API, consulte StartLiveTail em Referência AWS SDK para JavaScript da API.

`StartQuery`

O código de exemplo a seguir mostra como usar `StartQuery`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
```

```
        new StartQueryCommand({
          logGroupNames: this.logGroupNames,
          queryString: "fields @timestamp, @message | sort @timestamp asc",
          startTime: startDate.valueOf(),
          endTime: endDate.valueOf(),
          limit: maxLogs,
        }),
      );
    } catch (err) {
      /** @type {string} */
      const message = err.message;
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }
```

- Para obter detalhes da API, consulte StartQuery em Referência AWS SDK para JavaScript da API.

Cenários

Executar uma consulta grande

O exemplo de código a seguir mostra como usar o CloudWatch Logs para consultar mais de 10.000 registros.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o ponto de entrada.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
import { CloudWatchQuery } from "./cloud-watch-query.js";

console.log("Starting a recursive query...");

if (!process.env.QUERY_START_DATE || !process.env.QUERY_END_DATE) {
  throw new Error(
    "QUERY_START_DATE and QUERY_END_DATE environment variables are required.",
  );
}

const cloudWatchQuery = new CloudWatchQuery(new CloudWatchLogsClient({}), {
  logGroupNames: ["/workflows/cloudwatch-logs/large-query"],
  dateRange: [
    new Date(Number.parseInt(process.env.QUERY_START_DATE)),
    new Date(Number.parseInt(process.env.QUERY_END_DATE)),
  ],
});

await cloudWatchQuery.run();

console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs found:
 ${cloudWatchQuery.results.length}`,
);
```

Essa é uma classe que divide as consultas em várias etapas, se necessário.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  StartQueryCommand,
  GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}

export class CloudWatchQuery {
```

```
  /**
   * Run a query for all CloudWatch Logs within a certain date range.
   * CloudWatch logs return a max of 10,000 results. This class
   * performs a binary search across all of the logs in the provided
   * date range if a query returns the maximum number of results.
   *
   * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient} client
   * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
{ limit: number } }} config
   */
  constructor(client, { logGroupNames, dateRange, queryConfig }) {
    this.client = client;
    /**
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;

    /**
     * The inclusive date range that is queried.
     */
    this.dateRange = dateRange;

    /**
     * CloudWatch Logs never returns more than 10,000 logs.
     */
    this.limit = queryConfig?.limit ?? 10000;

    /**
     * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
     */
    this.results = [];
  }

  /**
   * Run the query.
   */
  async run() {
    this.secondsElapsed = 0;
    const start = new Date();
    this.results = await this._largeQuery(this.dateRange);
    const end = new Date();
    this.secondsElapsed = (end - start) / 1000;
    return this.results;
  }
```

```
/**
 * Recursively query for logs.
 * @param {[Date, Date]} dateRange
 * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[][]>}
 */
async _largeQuery(dateRange) {
  const logs = await this._query(dateRange, this.limit);

  console.log(
    `Query date range: ${dateRange
      .map((d) => d.toISOString())
      .join(" to ")}. Found ${logs.length} logs.`,
  );

  if (logs.length < this.limit) {
    return logs;
  }

  const lastLogDate = this._getLastLogDate(logs);
  const offsetLastLogDate = new Date(lastLogDate);
  offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
  const subDateRange = [offsetLastLogDate, dateRange[1]];
  const [r1, r2] = splitDateRange(subDateRange);
  const results = await Promise.all([
    this._largeQuery(r1),
    this._largeQuery(r2),
  ]);
  return [logs, ...results].flat();
}

/**
 * Find the most recent log in a list of logs.
 * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
 */
_getLastLogDate(logs) {
  const timestamps = logs
    .map(
      (log) =>
        log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,
    )
    .filter((t) => !!t)
    .map((t) => `${t}Z`)
    .sort();
```

```
    if (!timestamps.length) {
      throw new Error("No timestamp found in logs.");
    }

    return new Date(timestamps[timestamps.length - 1]);
  }

  /**
   * Simple wrapper for the GetQueryResultsCommand.
   * @param {string} queryId
   */
  _getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
  }

  /**
   * Starts a query and waits for it to complete.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   */
  async _query(dateRange, maxLogs) {
    try {
      const { queryId } = await this._startQuery(dateRange, maxLogs);
      const { results } = await this._waitUntilQueryDone(queryId);
      return results ?? [];
    } catch (err) {
      /**
       * This error is thrown when StartQuery returns an error indicating
       * that the query's start or end date occur before the log group was
       * created.
       */
      if (err instanceof DateOutOfBoundsError) {
        return [];
      }
      throw err;
    }
  }

  /**
   * Wrapper for the StartQueryCommand. Uses a static query string
   * for consistency.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
```

```
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
      new StartQueryCommand({
        logGroupNames: this.logGroupNames,
        queryString: "fields @timestamp, @message | sort @timestamp asc",
        startTime: startDate.valueOf(),
        endTime: endDate.valueOf(),
        limit: maxLogs,
      }),
    );
  } catch (err) {
    /** @type {string} */
    const message = err.message;
    if (message.startsWith("Query's end date and time")) {
      // This error indicates that the query's start or end date occur
      // before the log group was created.
      throw new DateOutOfBoundsError(message);
    }

    throw err;
  }
}

/**
 * Call GetQueryResultsCommand until the query is done.
 * @param {string} queryId
 */
_waitUntilQueryDone(queryId) {
  const getResults = async () => {
    const results = await this._getQueryResults(queryId);
    const queryDone = [
      "Complete",
      "Failed",
      "Cancelled",
      "Timeout",
      "Unknown",
    ].includes(results.status);

    return { queryDone, results };
  };
```

```
    return retry(
      { intervalInMs: 1000, maxRetries: 60, quiet: true },
      async () => {
        const { queryDone, results } = await getResults();
        if (!queryDone) {
          throw new Error("Query not done.");
        }

        return results;
      },
    );
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - GetQueryResults
  - StartQuery

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

CodeBuild exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com CodeBuild.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateProject`

O código de exemplo a seguir mostra como usar `CreateProject`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um projeto.

```
import {
  ArtifactsType,
  CodeBuildClient,
  ComputeType,
  CreateProjectCommand,
  EnvironmentType,
  SourceType,
} from "@aws-sdk/client-codebuild";

// Create the AWS CodeBuild project.
export const createProject = async (
  projectName = "MyCodeBuilder",
  roleArn = "arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin",
  buildOutputBucket = "xxxx",
  githubUrl = "https://...",
) => {
  const codeBuildClient = new CodeBuildClient({});

  const response = await codeBuildClient.send(
    new CreateProjectCommand({
      artifacts: {
        // The destination of the build artifacts.
        type: ArtifactsType.S3,
        location: buildOutputBucket,
      },
      // Information about the build environment. The combination of "computeType"
 and "type" determines the
      // requirements for the environment such as CPU, memory, and disk space.
      environment: {
        // Build environment compute types.
        // https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
compute-types.html
        computeType: ComputeType.BUILD_GENERAL1_SMALL,
        // Docker image identifier.
        // See https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
available.html
        image: "aws/codebuild/standard:7.0",
        // Build environment type.
        type: EnvironmentType.LINUX_CONTAINER,
      },
      name: projectName,
      // A role ARN with permission to create a CodeBuild project, write to the
 artifact location, and write CloudWatch logs.
```

```
      serviceRole: roleArn,
      source: {
        // The type of repository that contains the source code to be built.
        type: SourceType.GITHUB,
        // The location of the repository that contains the source code to be built.
        location: githubUrl,
      },
    }),
  );
  console.log(response);
  //   {
  //     '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: 'b428b244-777b-49a6-a48d-5dffedced8e7',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //     },
  //     project: {
  //       arn: 'arn:aws:codebuild:us-east-1:xxxxxxxxxxxx:project/MyCodeBuilder',
  //       artifacts: {
  //         encryptionDisabled: false,
  //         location: 'xxxxxx-xxxxxxx-xxxxxx',
  //         name: 'MyCodeBuilder',
  //         namespaceType: 'NONE',
  //         packaging: 'NONE',
  //         type: 'S3'
  //       },
  //       badge: { badgeEnabled: false },
  //       cache: { type: 'NO_CACHE' },
  //       created: 2023-08-18T14:46:48.979Z,
  //       encryptionKey: 'arn:aws:kms:us-east-1:xxxxxxxxxxxx:alias/aws/s3',
  //       environment: {
  //         computeType: 'BUILD_GENERAL1_SMALL',
  //         environmentVariables: [],
  //         image: 'aws/codebuild/standard:7.0',
  //         imagePullCredentialsType: 'CODEBUILD',
  //         privilegedMode: false,
  //         type: 'LINUX_CONTAINER'
  //       },
  //       lastModified: 2023-08-18T14:46:48.979Z,
  //       name: 'MyCodeBuilder',
  //       projectVisibility: 'PRIVATE',
```

```
  //       queuedTimeoutInMinutes: 480,
  //       serviceRole: 'arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin',
  //       source: {
  //         insecureSsl: false,
  //         location: 'https://...',
  //         reportBuildStatus: false,
  //         type: 'GITHUB'
  //       },
  //       timeoutInMinutes: 60
  //     }
  //   }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateProject em Referência AWS SDK para JavaScript da API.

Exemplos de identidade do Amazon Cognito usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Cognito Identity.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Exemplos do Amazon Cognito Identity Provider usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon Cognito

O exemplo de código a seguir mostra como começar a usar o Amazon Cognito.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});

  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- Para obter detalhes da API, consulte ListUserPools em Referência AWS SDK para JavaScript da API.

Ações

`AdminGetUser`

O código de exemplo a seguir mostra como usar `AdminGetUser`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const adminGetUser = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte AdminGetUser em Referência AWS SDK para JavaScript da API.

`AdminInitiateAuth`

O código de exemplo a seguir mostra como usar `AdminInitiateAuth`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte AdminInitiateAuth em Referência AWS SDK para JavaScript da API.

`AdminRespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `AdminRespondToAuthChallenge`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const adminRespondToAuthChallenge = ({
  userPoolId,
  clientId,
```

```
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallenge em Referência AWS SDK para JavaScript da API.

`AssociateSoftwareToken`

O código de exemplo a seguir mostra como usar `AssociateSoftwareToken`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const associateSoftwareToken = (session) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });
```

```
  return client.send(command);
};
```

- Para obter detalhes da API, consulte AssociateSoftwareToken em Referência AWS SDK para JavaScript da API.

`ConfirmDevice`

O código de exemplo a seguir mostra como usar `ConfirmDevice`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const confirmDevice = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt,
    },
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ConfirmDevice em Referência AWS SDK para JavaScript da API.

`ConfirmSignUp`

O código de exemplo a seguir mostra como usar `ConfirmSignUp`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ConfirmSignUp em Referência AWS SDK para JavaScript da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para JavaScript da API.

`InitiateAuth`

O código de exemplo a seguir mostra como usar `InitiateAuth`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const initiateAuth = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
```

```
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte InitiateAuth em Referência AWS SDK para JavaScript da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const listUsers = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para JavaScript da API.

`ResendConfirmationCode`

O código de exemplo a seguir mostra como usar `ResendConfirmationCode`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ResendConfirmationCode em Referência AWS SDK para JavaScript da API.

`RespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `RespondToAuthChallenge`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const respondToAuthChallenge = ({
```

```
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte RespondToAuthChallenge em Referência AWS SDK para JavaScript da API.

`SignUp`

O código de exemplo a seguir mostra como usar `SignUp`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte SignUp em Referência AWS SDK para JavaScript da API.

`UpdateUserPool`

O código de exemplo a seguir mostra como usar `UpdateUserPool`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });
```

```
    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Para obter detalhes da API, consulte UpdateUserPool em Referência AWS SDK para JavaScript da API.

`VerifySoftwareToken`

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
```

```
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte VerifySoftwareToken em Referência AWS SDK para JavaScript da API.

Cenários

Confirme automaticamente usuários conhecidos com uma função do Lambda

O exemplo de código a seguir mostra como confirmar automaticamente usuários conhecidas do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PreSignUp`.
- Inscreva-se para ser um usuário no Amazon Cognito.
- A função do Lambda verifica uma tabela do DynamoDB e confirma automaticamente os usuários conhecidos.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure uma execução interativa de “Cenário”. Os exemplos JavaScript (v3) compartilham um executor de cenários para simplificar exemplos complexos. O código-fonte completo está ativado GitHub.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      UserEmail: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
```

```
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

Esse cenário demonstra a confirmação automática de um usuário conhecido. Ele orquestra as etapas do exemplo.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
```

```
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
```

```
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
```

```
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
```

```
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
    const signUp = (password) =>
      signUpUser({
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
```

```
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
    if (logStreamErr) {
      state.errors.push(logStreamErr);
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);
```

```
const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with: ${state.token.slice(0,
 11)}`,
  { skipWhen: skipWhenErrors },
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
```

```
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
```

```
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

Essas são etapas compartilhadas com outros cenários.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
```

```
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
  { skipWhen: skipWhenErrors },
);
```

Um manipulador do gatilho `PreSignUp` com uma função do Lambda.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
```

```
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user '${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
 '${storedUserInfo.UserName}' does not match supplied UserName '${event.userName}'.
 Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
```

```
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

Módulo de ações de CloudWatch registros.

```
import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
```

```
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[] |
 null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Módulo de ações do Amazon Cognito.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
```

```
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password: string }}
 config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
```

```
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
```

```
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

Módulo de ações do DynamoDB.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string, unknown>[] }}
 config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
```

```
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Inscrever um usuário em um grupo de usuários que exija MFA

O exemplo de código a seguir mostra como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para obter a melhor experiência, clone o GitHub repositório e execute este exemplo. O código a seguir representa uma amostra da aplicação de exemplo completa.

```
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";
```

```
const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-up'
 command.`,
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Signing up.");
    await signUp({ clientId, username, password, email });
    logger.log(`Signed up. A confirmation email has been sent to: ${email}.`);
    logger.log(
      `Run 'confirm-sign-up ${username} <code>' to confirm your account.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the 'confirm-
sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
```

```
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Confirming user.");
    await confirmSignUp({ clientId, username, code });
    logger.log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { confirmSignUpHandler };

const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
```

```
const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-software-
token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-initiate-
auth' command.`,
    );
  }
};
```

```
const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    logger.log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      logger.log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      logger.log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    logger.error(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });
```

```
  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-auth-
challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
```

```
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    logger.log("Successfully authenticated.");
  } catch (err) {
    logger.error(err);
  }
};

export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
```

```
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    logger.log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    logger.log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    logger.error(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Exemplos do Amazon Comprehend usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Comprehend.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação de transmissão do Amazon Transcribe

O exemplo de código a seguir mostra como construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail.

SDK para JavaScript (v3)

Mostra como usar o Amazon Transcribe para construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Criar um chatbot Amazon Lex

O exemplo de código a seguir mostra como criar um chatbot para engajar os visitantes do seu site.

SDK para JavaScript (v3)

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o exemplo completo Como criar um chatbot Amazon Lex no guia do AWS SDK para JavaScript desenvolvedor.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado dentro das funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;
```

```
  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);
```

```
  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });
```

```
  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon DocumentDB usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento Amazon DocumentDB com o Lambda usando. JavaScript

```
console.log('Loading function');
exports.handler = async (event, context) => {
    event.events.forEach(record => {
        logDocumentDBEvent(record);
    });
    return 'OK';
};

const logDocumentDBEvent = (record) => {
    console.log('Operation type: ' + record.event.operationType);
```

```
    console.log('db: ' + record.event.ns.db);
    console.log('collection: ' + record.event.ns.coll);
    console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

Consumindo um evento do Amazon DocumentDB com o Lambda usando TypeScript

```
import { DocumentDBEventRecord, DocumentDBEventSubscriptionContext } from 'aws-
lambda';

console.log('Loading function');

export const handler = async (
  event: DocumentDBEventSubscriptionContext,
  context: any
): Promise<string> => {
  event.events.forEach((record: DocumentDBEventRecord) => {
    logDocumentDBEvent(record);
  });
  return 'OK';
};

const logDocumentDBEvent = (record: DocumentDBEventRecord): void => {
  console.log('Operation type: ' + record.event.operationType);
  console.log('db: ' + record.event.ns.db);
  console.log('collection: ' + record.event.ns.coll);
  console.log('Full document:', JSON.stringify(record.event.fullDocument, null, 2));
};
```

Exemplos do DynamoDB usando o SDK for JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para obter mais detalhes sobre como trabalhar com o DynamoDB em AWS SDK para JavaScript, consulte Programando o DynamoDB com. JavaScript

```
import { ListTablesCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
```

```
  const command = new ListTablesCommand({});

  const response = await client.send(command);
  console.log(response.TableNames.join("\n"));
  return response;
};
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { readFileSync } from "node:fs";
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DynamoDBClient,
```

```
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";

/**
 * This module is a convenience library. It abstracts Amazon DynamoDB's data type
 * descriptors (such as S, N, B, and BOOL) by marshalling JavaScript objects into
 * AttributeValue shapes.
 */
import {
  BatchWriteCommand,
  DeleteCommand,
  DynamoDBDocumentClient,
  GetCommand,
  PutCommand,
  UpdateCommand,
  paginateQuery,
  paginateScan,
} from "@aws-sdk/lib-dynamodb";

// These modules are local to our GitHub repository. We recommend cloning
// the project from GitHub if you want to run this example.
// For more information, see https://github.com/awsdocs/aws-doc-sdk-examples.
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { chunkArray } from "@aws-doc-sdk-examples/lib/utils/util-array.js";

const dirname = dirnameFromMetaUrl(import.meta.url);
const tableName = getUniqueName("Movies");
const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);

export const main = async () => {
  /**
   * Create a table.
   */

  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
```

```
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "year",
        // 'N' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "N",
      },
      { AttributeName: "title", AttributeType: "S" },
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [
      // The way your data is accessed determines how you structure your keys.
      // The movies table will be queried for movies by year. It makes sense
      // to make year our partition (HASH) key.
      { AttributeName: "year", KeyType: "HASH" },
      { AttributeName: "title", KeyType: "RANGE" },
    ],
  });

  log("Creating a table.");
  const createTableResponse = await client.send(createTableCommand);
  log(`Table created: ${JSON.stringify(createTableResponse.TableDescription)}`);

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Add a movie to the table.
   */

  log("Adding a single movie to the table.");
  // PutCommand is the first example usage of 'lib-dynamodb'.
  const putCommand = new PutCommand({
    TableName: tableName,
```

```
    Item: {
      // In 'client-dynamodb', the AttributeValue would be required (`year: { N:
 1981 }`)
      // 'lib-dynamodb' simplifies the usage ( `year: 1981` )
      year: 1981,
      // The preceding KeySchema defines 'title' as our sort (RANGE) key, so 'title'
      // is required.
      title: "The Evil Dead",
      // Every other attribute is optional.
      info: {
        genres: ["Horror"],
      },
    },
  });
  await docClient.send(putCommand);
  log("The movie was added.");

  /**
   * Get a movie from the table.
   */

  log("Getting a single movie from the table.");
  const getCommand = new GetCommand({
    TableName: tableName,
    // Requires the complete primary key. For the movies table, the primary key
    // is only the id (partition key).
    Key: {
      year: 1981,
      title: "The Evil Dead",
    },
    // Set this to make sure that recent writes are reflected.
    // For more information, see https://docs.aws.amazon.com/amazondynamodb/latest/
developerguide/HowItWorks.ReadConsistency.html.
    ConsistentRead: true,
  });
  const getResponse = await docClient.send(getCommand);
  log(`Got the movie: ${JSON.stringify(getResponse.Item)}`);

  /**
   * Update a movie in the table.
   */

  log("Updating a single movie in the table.");
  const updateCommand = new UpdateCommand({
```

```
    TableName: tableName,
    Key: { year: 1981, title: "The Evil Dead" },
    // This update expression appends "Comedy" to the list of genres.
    // For more information on update expressions, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.UpdateExpressions.html
    UpdateExpression: "set #i.#g = list_append(#i.#g, :vals)",
    ExpressionAttributeNames: { "#i": "info", "#g": "genres" },
    ExpressionAttributeValues: {
      ":vals": ["Comedy"],
    },
    ReturnValues: "ALL_NEW",
  });
  const updateResponse = await docClient.send(updateCommand);
  log(`Movie updated: ${JSON.stringify(updateResponse.Attributes)}`);

  /**
   * Delete a movie from the table.
   */

  log("Deleting a single movie from the table.");
  const deleteCommand = new DeleteCommand({
    TableName: tableName,
    Key: { year: 1981, title: "The Evil Dead" },
  });
  await docClient.send(deleteCommand);
  log("Movie deleted.");

  /**
   * Upload a batch of movies.
   */

  log("Adding movies from local JSON file.");
  const file = readFileSync(
    `${dirname}../../../../resources/sample_files/movies.json`,
  );
  const movies = JSON.parse(file.toString());
  // chunkArray is a local convenience function. It takes an array and returns
  // a generator function. The generator function yields every N items.
  const movieChunks = chunkArray(movies, 25);
  // For every chunk of 25 movies, make one BatchWrite request.
  for (const chunk of movieChunks) {
    const putRequests = chunk.map((movie) => ({
      PutRequest: {
```

```
        Item: movie,
      },
    }));

    const command = new BatchWriteCommand({
      RequestItems: {
        [tableName]: putRequests,
      },
    });

    await docClient.send(command);
  }
  log("Movies added.");

  /**
   * Query for movies by year.
   */

  log("Querying for all movies from 1981.");
  const paginatedQuery = paginateQuery(
    { client: docClient },
    {
      TableName: tableName,
      //For more information about query expressions, see
      // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Query.html#Query.KeyConditionExpressions
      KeyConditionExpression: "#y = :y",
      // 'year' is a reserved word in DynamoDB. Indicate that it's an attribute
      // name by using an expression attribute name.
      ExpressionAttributeNames: { "#y": "year" },
      ExpressionAttributeValues: { ":y": 1981 },
      ConsistentRead: true,
    },
  );
  /**
   * @type { Record<string, any>[] };
   */
  const movies1981 = [];
  for await (const page of paginatedQuery) {
    movies1981.push(...page.Items);
  }
  log(`Movies: ${movies1981.map((m) => m.title).join(", ")}`);

  /**
```

```
    * Scan the table for movies between 1980 and 1990.
    */

  log("Scan for movies released between 1980 and 1990");
  // A 'Scan' operation always reads every item in the table. If your design
 requires
  // the use of 'Scan', consider indexing your table or changing your design.
  // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-query-
scan.html
  const paginatedScan = paginateScan(
    { client: docClient },
    {
      TableName: tableName,
      // Scan uses a filter expression instead of a key condition expression. Scan
 will
      // read the entire table and then apply the filter.
      FilterExpression: "#y between :y1 and :y2",
      ExpressionAttributeNames: { "#y": "year" },
      ExpressionAttributeValues: { ":y1": 1980, ":y2": 1990 },
      ConsistentRead: true,
    },
  );
  /**
   * @type { Record<string, any>[] };
   */
  const movies1980to1990 = [];
  for await (const page of paginatedScan) {
    movies1980to1990.push(...page.Items);
  }
  log(
    `Movies: ${movies1980to1990
      .map((m) => `${m.title} (${m.year})`)
      .join(", ")}`,
  );

  /**
   * Delete the table.
   */

  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  log(`Deleting table ${tableName}.`);
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`BatchExecuteStatement`

O código de exemplo a seguir mostra como usar `BatchExecuteStatement`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um lote de itens usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
```

```
const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const breakfastFoods = ["Eggs", "Bacon", "Sausage"];
  const command = new BatchExecuteStatementCommand({
    Statements: breakfastFoods.map((food) => ({
      Statement: `INSERT INTO BreakfastFoods value {'Name':?}`,
      Parameters: [food],
    })),
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Obtenha um lote de itens usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchExecuteStatementCommand({
    Statements: [
      {
        Statement: "SELECT * FROM PepperMeasurements WHERE Unit=?",
        Parameters: ["Teaspoons"],
        ConsistentRead: true,
      },
      {
        Statement: "SELECT * FROM PepperMeasurements WHERE Unit=?",
        Parameters: ["Grams"],
        ConsistentRead: true,
```

```
      },
    ],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Atualize um lote de itens usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const eggUpdates = [
    ["duck", "fried"],
    ["chicken", "omelette"],
  ];
  const command = new BatchExecuteStatementCommand({
    Statements: eggUpdates.map((change) => ({
      Statement: "UPDATE Eggs SET Style=? where Variety=?",
      Parameters: [change[1], change[0]],
    })),
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Exclua um lote de itens usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
```

```
import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchExecuteStatementCommand({
    Statements: [
      {
        Statement: "DELETE FROM Flavors where Name=?",
        Parameters: ["Grape"],
      },
      {
        Statement: "DELETE FROM Flavors where Name=?",
        Parameters: ["Strawberry"],
      },
    ],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para JavaScript da API.

`BatchGetItem`

O código de exemplo a seguir mostra como usar `BatchGetItem`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte BatchGet.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { BatchGetCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchGetCommand({
    // Each key in this object is the name of a table. This example refers
    // to a Books table.
    RequestItems: {
      Books: {
        // Each entry in Keys is an object that specifies a primary key.
        Keys: [
          {
            Title: "How to AWS",
          },
          {
            Title: "DynamoDB for DBAs",
          },
        ],
        // Only return the "Title" and "PageCount" attributes.
        ProjectionExpression: "Title, PageCount",
      },
    },
  });

  const response = await docClient.send(command);
  console.log(response.Responses.Books);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte BatchGetItem em Referência AWS SDK para JavaScript da API.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte BatchWrite.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";
import { readFileSync } from "node:fs";

// These modules are local to our GitHub repository. We recommend cloning
// the project from GitHub if you want to run this example.
// For more information, see https://github.com/awsdocs/aws-doc-sdk-examples.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { chunkArray } from "@aws-doc-sdk-examples/lib/utils/util-array.js";

const dirname = dirnameFromMetaUrl(import.meta.url);

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
```

```
  const file = readFileSync(
    `${dirname}../../../../../resources/sample_files/movies.json`,
  );

  const movies = JSON.parse(file.toString());

  // chunkArray is a local convenience function. It takes an array and returns
  // a generator function. The generator function yields every N items.
  const movieChunks = chunkArray(movies, 25);

  // For every chunk of 25 movies, make one BatchWrite request.
  for (const chunk of movieChunks) {
    const putRequests = chunk.map((movie) => ({
      PutRequest: {
        Item: movie,
      },
    }));

    const command = new BatchWriteCommand({
      RequestItems: {
        // An existing table is required. A composite key of 'title' and 'year' is
 recommended
        // to account for duplicate titles.
        BatchWriteMoviesTable: putRequests,
      },
    });

    await docClient.send(command);
  }
};
```

- Para obter detalhes da API, consulte BatchWriteItem em Referência AWS SDK para JavaScript da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import { CreateTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new CreateTableCommand({
    TableName: "EspressoDrinks",
    // For more information about data types,
    // see https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
HowItWorks.NamingRulesDataTypes.html#HowItWorks.DataTypes and
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
    AttributeDefinitions: [
      {
        AttributeName: "DrinkName",
        AttributeType: "S",
      },
    ],
    KeySchema: [
      {
        AttributeName: "DrinkName",
        KeyType: "HASH",
      },
    ],
    BillingMode: "PAY_PER_REQUEST",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para JavaScript da API.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte DeleteCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, DeleteCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new DeleteCommand({
    TableName: "Sodas",
    Key: {
      Flavor: "Cola",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteItem em Referência AWS SDK para JavaScript da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new DeleteTableCommand({
    TableName: "DecafCoffees",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para JavaScript da API.

`DescribeTable`

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import { DescribeTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new DescribeTableCommand({
    TableName: "Pastries",
  });

  const response = await client.send(command);
  console.log(`TABLE NAME: ${response.Table.TableName}`);
  console.log(`TABLE ITEM COUNT: ${response.Table.ItemCount}`);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DescribeTable em Referência AWS SDK para JavaScript da API.

`DescribeTimeToLive`

O código de exemplo a seguir mostra como usar `DescribeTimeToLive`.

SDK para JavaScript (v3)

Descreva a configuração de TTL em uma tabela existente do DynamoDB usando o AWS SDK para JavaScript.

```
import { DynamoDBClient, DescribeTimeToLiveCommand } from "@aws-sdk/client-
dynamodb";

export const describeTTL = async (tableName, region) => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    try {
        const ttlDescription = await client.send(new
 DescribeTimeToLiveCommand({ TableName: tableName }));
```

```
        if (ttlDescription.TimeToLiveDescription.TimeToLiveStatus === 'ENABLED') {
            console.log("TTL is enabled for table %s.", tableName);
        } else {
            console.log("TTL is not enabled for table %s.", tableName);
        }

        return ttlDescription;
    } catch (e) {
        console.error(`Error describing table: ${e}`);
        throw e;
    }
}

// Example usage (commented out for testing)
// describeTTL('your-table-name', 'us-east-1');
```

- Para obter detalhes da API, consulte DescribeTimeToLive em Referência AWS SDK para JavaScript da API.

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um item usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
```

```
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: `INSERT INTO Flowers value {'Name':?}`,
    Parameters: ["Rose"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Obtenha um item usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "SELECT * FROM CloudTypes WHERE IsStorm=?",
    Parameters: [false],
    ConsistentRead: true,
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Atualize um item usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
```

```
import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "UPDATE EyeColors SET IsRecessive=? where Color=?",
    Parameters: [true, "blue"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Exclua um item usando o PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "DELETE FROM PaintColors where Name=?",
    Parameters: ["Purple"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para JavaScript da API.

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte GetCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, GetCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new GetCommand({
    TableName: "AngryAnimals",
    Key: {
      CommonName: "Shoebill",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte GetItem em Referência AWS SDK para JavaScript da API.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import { ListTablesCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new ListTablesCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para JavaScript da API.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte PutCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { PutCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new PutCommand({
    TableName: "HappyAnimals",
    Item: {
      CommonName: "Shiba Inu",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK para JavaScript da API.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte QueryCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { QueryCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new QueryCommand({
    TableName: "CoffeeCrop",
    KeyConditionExpression:
      "OriginCountry = :originCountry AND RoastDate > :roastDate",
    ExpressionAttributeValues: {
      ":originCountry": "Ethiopia",
      ":roastDate": "2023-05-01",
    },
    ConsistentRead: true,
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte ScanCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, ScanCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ScanCommand({
    ProjectionExpression: "#Name, Color, AvgLifeSpan",
    ExpressionAttributeNames: { "#Name": "Name" },
    TableName: "Birds",
  });

  const response = await docClient.send(command);
  for (const bird of response.Items) {
    console.log(`${bird.Name} - (${bird.Color}, ${bird.AvgLifeSpan})`);
  }
  return response;
};
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para JavaScript .

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Neste exemplo, é usado o cliente de documentos para simplificar o trabalho com itens no DynamoDB. Para obter detalhes da API, consulte UpdateCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, UpdateCommand } from "@aws-sdk/lib-dynamodb";
```

```
const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new UpdateCommand({
    TableName: "Dogs",
    Key: {
      Breed: "Labrador",
    },
    UpdateExpression: "set Color = :color",
    ExpressionAttributeValues: {
      ":color": "black",
    },
    ReturnValues: "ALL_NEW",
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

`UpdateTimeToLive`

O código de exemplo a seguir mostra como usar `UpdateTimeToLive`.

SDK para JavaScript (v3)

Habilite a TTL em uma tabela existente do DynamoDB.

```
import { DynamoDBClient, UpdateTimeToLiveCommand } from "@aws-sdk/client-dynamodb";

export const enableTTL = async (tableName, ttlAttribute, region = 'us-east-1') => {

    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const params = {
```

```
        TableName: tableName,
        TimeToLiveSpecification: {
            Enabled: true,
            AttributeName: ttlAttribute
        }
    };

    try {
        const response = await client.send(new UpdateTimeToLiveCommand(params));
        if (response.$metadata.httpStatusCode === 200) {
            console.log(`TTL enabled successfully for table ${tableName}, using
 attribute name ${ttlAttribute}.`);
        } else {
            console.log(`Failed to enable TTL for table ${tableName}, response
 object: ${response}`);
        }
        return response;
    } catch (e) {
        console.error(`Error enabling TTL: ${e}`);
        throw e;
    }
};

// Example usage (commented out for testing)
// enableTTL('ExampleTable', 'exampleTtlAttribute');
```

Desabilite a TTL em uma tabela existente do DynamoDB.

```
import { DynamoDBClient, UpdateTimeToLiveCommand } from "@aws-sdk/client-dynamodb";

export const disableTTL = async (tableName, ttlAttribute, region = 'us-east-1') => {

    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const params = {
        TableName: tableName,
        TimeToLiveSpecification: {
            Enabled: false,
            AttributeName: ttlAttribute
        }
```

```
    };

    try {
        const response = await client.send(new UpdateTimeToLiveCommand(params));
        if (response.$metadata.httpStatusCode === 200) {
            console.log(`TTL disabled successfully for table ${tableName}, using
 attribute name ${ttlAttribute}.`);
        } else {
            console.log(`Failed to disable TTL for table ${tableName}, response
 object: ${response}`);
        }
        return response;
    } catch (e) {
        console.error(`Error disabling TTL: ${e}`);
        throw e;
    }
};

// Example usage (commented out for testing)
// disableTTL('ExampleTable', 'exampleTtlAttribute');
```

- Para obter detalhes da API, consulte UpdateTimeToLive em Referência AWS SDK para JavaScript da API.

Cenários

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

SDK para JavaScript (v3)

Este exemplo mostra como criar uma aplicação que permite que os usuários enviem dados para uma tabela do Amazon DynamoDB e enviem uma mensagem de texto ao administrador usando o Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Comparar vários valores com um único atributo

O exemplo de código a seguir mostra como comparar vários valores com um único atributo no DynamoDB.

- Use o operador IN para comparar vários valores com um único atributo.
- Compare o operador IN com várias condições OR.
- Saiba quais são os benefícios de desempenho e complexidade de expressão oferecidos pelo IN.

SDK para JavaScript (v3)

Compare vários valores com um único atributo usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  ScanCommand,
  QueryCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Query or scan a DynamoDB table to find items where an attribute matches any value
 from a list.
 *
 * This function demonstrates the use of the IN operator to compare a single
 attribute
 * against multiple possible values, which is more efficient than using multiple OR
 conditions.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} attributeName - The name of the attribute to compare against the
 values list
 * @param {Array} valuesList - List of values to compare the attribute against
 * @param {string} [partitionKeyName] - Optional name of the partition key attribute
 for query operations
```

```
 * @param {string} [partitionKeyValue] - Optional value of the partition key to
 query
 * @returns {Promise<Object>} - The response from DynamoDB containing the matching
 items
 */
async function compareMultipleValues(
  config,
  tableName,
  attributeName,
  valuesList,
  partitionKeyName,
  partitionKeyValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the filter expression using the IN operator
  const filterExpression = `${attributeName} IN (${valuesList.map((_, index) =>
 `:val${index}`).join(', ')})`;

  // Create expression attribute values for the values list
  const expressionAttributeValues = valuesList.reduce((acc, val, index) => {
    acc[`:val${index}`] = val;
    return acc;
  }, {});

  // If partition key is provided, perform a query operation
  if (partitionKeyName && partitionKeyValue) {
    const keyCondition = `${partitionKeyName} = :partitionKey`;
    expressionAttributeValues[':partitionKey'] = partitionKeyValue;

    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        KeyConditionExpression: keyCondition,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };
```

```
      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous query
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new QueryCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  } else {
    // Otherwise, perform a scan operation
    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous scan
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new ScanCommand(params));

      // Add the items from this page to our collection
```

```
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  }
}

/**
 * Alternative implementation using multiple OR conditions instead of the IN
 operator.
 *
 * This function is provided for comparison to show why using the IN operator is
 preferable.
 * With many values, this approach becomes verbose and less efficient.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} attributeName - The name of the attribute to compare against the
 values list
 * @param {Array} valuesList - List of values to compare the attribute against
 * @param {string} [partitionKeyName] - Optional name of the partition key attribute
 for query operations
 * @param {string} [partitionKeyValue] - Optional value of the partition key to
 query
 * @returns {Promise<Object>} - The response from DynamoDB containing the matching
 items
 */
async function compareWithOrConditions(
  config,
  tableName,
  attributeName,
  valuesList,
  partitionKeyName,
  partitionKeyValue
) {
```

```
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // If no values provided, return empty result
  if (!valuesList || valuesList.length === 0) {
    return {
      Items: [],
      Count: 0
    };
  }

  // Create the filter expression using multiple OR conditions
  const filterConditions = valuesList.map((_, index) => `${attributeName} = :val
${index}`);
  const filterExpression = filterConditions.join(' OR ');

  // Create expression attribute values for the values list
  const expressionAttributeValues = valuesList.reduce((acc, val, index) => {
    acc[`:val${index}`] = val;
    return acc;
  }, {});

  // If partition key is provided, perform a query operation
  if (partitionKeyName && partitionKeyValue) {
    const keyCondition = `${partitionKeyName} = :partitionKey`;
    expressionAttributeValues[':partitionKey'] = partitionKeyValue;

    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        KeyConditionExpression: keyCondition,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous query
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
```

```
      }

      const response = await docClient.send(new QueryCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  } else {
    // Otherwise, perform a scan operation
    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous scan
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new ScanCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }
```

```
      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  }
}
```

Exemplo de uso de comparação de vários valores com AWS SDK para JavaScript.

```
/**
 * Example of how to use the compareMultipleValues function.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const attributeName = "Category";
  const valuesList = ["Electronics", "Computers", "Accessories"];

  console.log(`Searching for products in any of these categories:
 ${valuesList.join(', ')}`);

  try {
    // Using the IN operator (recommended approach)
    console.log("\nApproach 1: Using the IN operator");
    const response = await compareMultipleValues(
      config,
      tableName,
      attributeName,
      valuesList
    );

    console.log(`Found ${response.Count} products in the specified categories`);

    // Using multiple OR conditions (alternative approach)
    console.log("\nApproach 2: Using multiple OR conditions");
    const response2 = await compareWithOrConditions(
```

```
      config,
      tableName,
      attributeName,
      valuesList
    );

    console.log(`Found ${response2.Count} products in the specified categories`);

    // Example with a query operation
    console.log("\nQuerying a specific manufacturer's products in multiple
 categories");
    const partitionKeyName = "Manufacturer";
    const partitionKeyValue = "Acme";

    const response3 = await compareMultipleValues(
      config,
      tableName,
      attributeName,
      valuesList,
      partitionKeyName,
      partitionKeyValue
    );

    console.log(`Found ${response3.Count} Acme products in the specified
 categories`);

    // Explain the benefits of using the IN operator
    console.log("\nBenefits of using the IN operator:");
    console.log("1. More concise expression compared to multiple OR conditions");
    console.log("2. Better readability and maintainability");
    console.log("3. Potentially better performance with large value lists");
    console.log("4. Simpler code that's less prone to errors");
    console.log("5. Easier to modify when adding or removing values");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - Query

- Scan

Atualizar condicionalmente a TTL de um item

O exemplo de código a seguir mostra como atualizar condicionalmente a TTL de um item.

SDK para JavaScript (v3)

Atualize a TTL em um item do DynamoDB existente em uma tabela, com uma condição.

```
import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const updateItemConditional = async (tableName, partitionKey, sortKey, region
 = 'us-east-1', newAttribute = 'default-value') => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);

    const params = {
        TableName: tableName,
        Key: marshall({
            artist: partitionKey,
            album: sortKey
        }),
        UpdateExpression: "SET newAttribute = :newAttribute",
        ConditionExpression: "expireAt > :expiration",
        ExpressionAttributeValues: marshall({
            ':newAttribute': newAttribute,
            ':expiration': currentTime
        }),
        ReturnValues: "ALL_NEW"
    };

    try {
        const response = await client.send(new UpdateItemCommand(params));
        const responseData = unmarshall(response.Attributes);
        console.log("Item updated successfully: ", responseData);
        return responseData;
    } catch (error) {
```

```
        if (error.name === "ConditionalCheckFailedException") {
            console.log("Condition check failed: Item's 'expireAt' is expired.");
        } else {
            console.error("Error updating item: ", error);
        }
        throw error;
    }
};

// Example usage (commented out for testing)
// updateItemConditional('your-table-name', 'your-partition-key-value', 'your-sort-
key-value');
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

Contar operadores de expressão

O exemplo de código a seguir mostra como contar operadores de expressão no DynamoDB.

- Entenda o limite de 300 operadores do DynamoDB.
- Conte operadores em expressões complexas.
- Otimize as expressões para permanecer dentro dos limites.

SDK para JavaScript (v3)

Demonstre a contagem de operadores de expressão usando o AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  QueryCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Create a complex filter expression with a specified number of conditions.
 *
 * This function demonstrates how to generate a complex expression with
 * a specific number of operators to test the 300 operator limit.
```

```
 *
 * @param {number} conditionsCount - Number of conditions to include
 * @param {boolean} useAnd - Whether to use AND (true) or OR (false) between
 conditions
 * @returns {Object} - Object containing the filter expression and attribute values
 */
function createComplexFilterExpression(conditionsCount, useAnd = true) {
  // Initialize the expression parts and attribute values
  const conditions = [];
  const expressionAttributeValues = {};

  // Generate the specified number of conditions
  for (let i = 0; i < conditionsCount; i++) {
    // Alternate between different comparison operators for variety
    let condition;
    const valueKey = `:val${i}`;

    switch (i % 5) {
      case 0:
        condition = `attribute${i} = ${valueKey}`;
        expressionAttributeValues[valueKey] = `value${i}`;
        break;
      case 1:
        condition = `attribute${i} > ${valueKey}`;
        expressionAttributeValues[valueKey] = i;
        break;
      case 2:
        condition = `attribute${i} < ${valueKey}`;
        expressionAttributeValues[valueKey] = i * 10;
        break;
      case 3:
        condition = `contains(attribute${i}, ${valueKey})`;
        expressionAttributeValues[valueKey] = `substring${i}`;
        break;
      case 4:
        condition = `attribute_exists(attribute${i})`;
        break;
    }

    conditions.push(condition);
  }

  // Join the conditions with AND or OR
  const operator = useAnd ? " AND " : " OR ";
```

```
  const filterExpression = conditions.join(operator);

  // Calculate the operator count
  // Each condition has 1 operator (=, >, <, contains, attribute_exists)
  // Each AND or OR between conditions is 1 operator
  const operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount -
 1 : 0);

  return {
    filterExpression,
    expressionAttributeValues,
    operatorCount
  };
}

/**
 * Create a complex update expression with a specified number of operations.
 *
 * This function demonstrates how to generate a complex update expression with
 * a specific number of operators to test the 300 operator limit.
 *
 * @param {number} operationsCount - Number of operations to include
 * @returns {Object} - Object containing the update expression and attribute values
 */
function createComplexUpdateExpression(operationsCount) {
  // Initialize the expression parts and attribute values
  const setOperations = [];
  const expressionAttributeValues = {};

  // Generate the specified number of SET operations
  for (let i = 0; i < operationsCount; i++) {
    // Alternate between different types of SET operations
    let operation;
    const valueKey = `:val${i}`;

    switch (i % 3) {
      case 0:
        // Simple assignment (1 operator: =)
        operation = `attribute${i} = ${valueKey}`;
        expressionAttributeValues[valueKey] = `value${i}`;
        break;
      case 1:
        // Addition (2 operators: = and +)
        operation = `attribute${i} = attribute${i} + ${valueKey}`;
```

```
        expressionAttributeValues[valueKey] = i;
        break;
      case 2:
        // Conditional assignment with if_not_exists (2 operators: = and
 if_not_exists)
        operation = `attribute${i} = if_not_exists(attribute${i}, ${valueKey})`;
        expressionAttributeValues[valueKey] = i * 10;
        break;
    }

    setOperations.push(operation);
  }

  // Create the update expression
  const updateExpression = `SET ${setOperations.join(", ")}`;

  // Calculate the operator count
  // Each operation has 1-2 operators as noted above
  let operatorCount = 0;
  for (let i = 0; i < operationsCount; i++) {
    operatorCount += (i % 3 === 0) ? 1 : 2;
  }

  return {
    updateExpression,
    expressionAttributeValues,
    operatorCount
  };
}

/**
 * Test the operator limit by attempting an operation with a complex expression.
 *
 * This function demonstrates what happens when an expression approaches or
 * exceeds the 300 operator limit.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {number} operatorCount - Target number of operators to include
 * @returns {Promise<Object>} - Result of the operation attempt
 */
async function testOperatorLimit(
  config,
```

```
  tableName,
  key,
  operatorCount
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create a complex update expression with the specified operator count
  const { updateExpression, expressionAttributeValues, operatorCount: actualCount }
=
    createComplexUpdateExpression(Math.ceil(operatorCount / 1.5)); // Adjust to get
close to target count

  console.log(`Generated update expression with approximately ${actualCount}
operators`);

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Attempt the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return {
      success: true,
      message: `Operation succeeded with ${actualCount} operators`,
      data: response
    };
  } catch (error) {
    // Check if the error is due to exceeding the operator limit
    if (error.name === "ValidationException" &&
        error.message.includes("too many operators")) {
      return {
        success: false,
        message: `Operation failed: ${error.message}`,
        operatorCount: actualCount
      };
    }
```

```
    // Return other errors
    return {
      success: false,
      message: `Operation failed: ${error.message}`,
      error
    };
  }
}

/**
 * Break down a complex expression into multiple simpler operations.
 *
 * This function demonstrates how to handle expressions that would exceed
 * the 300 operator limit by breaking them into multiple operations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {number} totalOperations - Total number of operations to perform
 * @returns {Promise<Object>} - Result of the operations
 */
async function breakDownComplexExpression(
  config,
  tableName,
  key,
  totalOperations
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Calculate how many operations we can safely include in each batch
  // Using 150 as a conservative limit (well below 300)
  const operationsPerBatch = 100;
  const batchCount = Math.ceil(totalOperations / operationsPerBatch);

  console.log(`Breaking down ${totalOperations} operations into ${batchCount}
 batches`);

  const results = [];

  // Process each batch
  for (let batch = 0; batch < batchCount; batch++) {
```

```
    // Calculate the operations for this batch
    const batchStart = batch * operationsPerBatch;
    const batchEnd = Math.min(batchStart + operationsPerBatch, totalOperations);
    const batchSize = batchEnd - batchStart;

    console.log(`Processing batch ${batch + 1}/${batchCount} with ${batchSize}
operations`);

    // Create an update expression for this batch
    const { updateExpression, expressionAttributeValues, operatorCount } =
      createComplexUpdateExpression(batchSize);

    // Define the update parameters
    const params = {
      TableName: tableName,
      Key: key,
      UpdateExpression: updateExpression,
      ExpressionAttributeValues: expressionAttributeValues,
      ReturnValues: "UPDATED_NEW"
    };

    try {
      // Perform the update operation for this batch
      const response = await docClient.send(new UpdateCommand(params));

      results.push({
        batch: batch + 1,
        success: true,
        operatorCount,
        attributes: response.Attributes
      });
    } catch (error) {
      results.push({
        batch: batch + 1,
        success: false,
        operatorCount,
        error: error.message
      });

      // Stop processing if an error occurs
      break;
    }
  }
```

```
  return {
    totalBatches: batchCount,
    results
  };
}

/**
 * Count operators in a DynamoDB expression based on the rules in the documentation.
 *
 * This function demonstrates how operators are counted according to the
 * DynamoDB documentation.
 *
 * @param {string} expression - The DynamoDB expression to analyze
 * @returns {Object} - Breakdown of operator counts
 */
function countOperatorsInExpression(expression) {
  // Initialize counters for different operator types
  const counts = {
    comparisonOperators: 0,
    logicalOperators: 0,
    functions: 0,
    arithmeticOperators: 0,
    specialOperators: 0,
    total: 0
  };

  // Count comparison operators (=, <>, <, <=, >, >=)
  const comparisonRegex = /[^<>]=[^=]|<>|<=|>=|[^<]>[^=]|[^>]<[^=]/g;
  const comparisonMatches = expression.match(comparisonRegex) || [];
  counts.comparisonOperators = comparisonMatches.length;

  // Count logical operators (AND, OR, NOT)
  const andMatches = expression.match(/\bAND\b/g) || [];
  const orMatches = expression.match(/\bOR\b/g) || [];
  const notMatches = expression.match(/\bNOT\b/g) || [];
  counts.logicalOperators = andMatches.length + orMatches.length +
 notMatches.length;

  // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
  const functionRegex = /\b(attribute_exists|attribute_not_exists|attribute_type|
begins_with|contains|size|if_not_exists)\(/g;
  const functionMatches = expression.match(functionRegex) || [];
  counts.functions = functionMatches.length;
```

```
  // Count arithmetic operators (+ and -)
  const arithmeticMatches = expression.match(/[a-zA-Z0-9_)\]]\s*[\+\-]\s*[a-zA-
Z0-9_(:]/g) || [];
  counts.arithmeticOperators = arithmeticMatches.length;

  // Count special operators (BETWEEN, IN)
  const betweenMatches = expression.match(/\bBETWEEN\b/g) || [];
  const inMatches = expression.match(/\bIN\b/g) || [];
  counts.specialOperators = betweenMatches.length + inMatches.length;

  // Add extra operators for BETWEEN (each BETWEEN includes an AND)
  counts.logicalOperators += betweenMatches.length;

  // Calculate total
  counts.total = counts.comparisonOperators +
                 counts.logicalOperators +
                 counts.functions +
                 counts.arithmeticOperators +
                 counts.specialOperators;

  return counts;
}
```

Exemplo de uso da contagem de operadores de expressão com AWS SDK para JavaScript.

```
/**
 * Example of how to work with expression operator counting.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating DynamoDB expression operator counting and the 300
 operator limit");

  try {
    // Example 1: Analyze a simple expression
    console.log("\nExample 1: Analyzing a simple expression");
```

```
   const simpleExpression = "Price = :price AND Rating > :rating AND Category IN
(:cat1, :cat2, :cat3)";
   const simpleCount = countOperatorsInExpression(simpleExpression);

   console.log(`Expression: ${simpleExpression}`);
   console.log("Operator count breakdown:");
   console.log(`- Comparison operators: ${simpleCount.comparisonOperators}`);
   console.log(`- Logical operators: ${simpleCount.logicalOperators}`);
   console.log(`- Functions: ${simpleCount.functions}`);
   console.log(`- Arithmetic operators: ${simpleCount.arithmeticOperators}`);
   console.log(`- Special operators: ${simpleCount.specialOperators}`);
   console.log(`- Total operators: ${simpleCount.total}`);

   // Example 2: Analyze a complex expression
   console.log("\nExample 2: Analyzing a complex expression");
   const complexExpression =
     "(attribute_exists(Category) AND Size BETWEEN :min AND :max) OR " +
     "(Price > :price AND contains(Description, :keyword) AND " +
     "(Rating >= :minRating OR Reviews > :minReviews))";
   const complexCount = countOperatorsInExpression(complexExpression);

   console.log(`Expression: ${complexExpression}`);
   console.log("Operator count breakdown:");
   console.log(`- Comparison operators: ${complexCount.comparisonOperators}`);
   console.log(`- Logical operators: ${complexCount.logicalOperators}`);
   console.log(`- Functions: ${complexCount.functions}`);
   console.log(`- Arithmetic operators: ${complexCount.arithmeticOperators}`);
   console.log(`- Special operators: ${complexCount.specialOperators}`);
   console.log(`- Total operators: ${complexCount.total}`);

   // Example 3: Test approaching the operator limit
   console.log("\nExample 3: Testing an expression approaching the operator
limit");
   const approachingLimit = await testOperatorLimit(config, tableName, key, 290);
   console.log(approachingLimit.message);

   // Example 4: Test exceeding the operator limit
   console.log("\nExample 4: Testing an expression exceeding the operator limit");
   const exceedingLimit = await testOperatorLimit(config, tableName, key, 310);
   console.log(exceedingLimit.message);

   // Example 5: Breaking down a complex expression
   console.log("\nExample 5: Breaking down a complex expression into multiple
operations");
```

```
    const breakdownResult = await breakDownComplexExpression(config, tableName, key,
500);
    console.log(`Processed ${breakdownResult.results.length} of
${breakdownResult.totalBatches} batches`);

    // Explain the operator counting rules
    console.log("\nKey points about DynamoDB expression operator counting:");
    console.log("1. The maximum number of operators in any expression is 300");
    console.log("2. Each comparison operator (=, <>, <, <=, >, >=) counts as 1
operator");
    console.log("3. Each logical operator (AND, OR, NOT) counts as 1 operator");
    console.log("4. Each function call (attribute_exists, contains, etc.) counts as
1 operator");
    console.log("5. Each arithmetic operator (+ or -) counts as 1 operator");
    console.log("6. BETWEEN counts as 2 operators (BETWEEN itself and the AND within
it)");
    console.log("7. IN counts as 1 operator regardless of the number of values");
    console.log("8. Parentheses for grouping and attribute paths don't count as
operators");
    console.log("9. When you exceed the limit, the error always reports '301
operators'");
    console.log("10. For complex operations, break them into multiple smaller
operations");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma tabela com o throughput a quente habilitado

O exemplo de código a seguir mostra como criar uma tabela com o throughput a quente habilitado.

SDK para JavaScript (v3)

Crie uma tabela do DynamoDB com uma configuração de throughput a quente usando o AWS SDK para JavaScript.

```
import { DynamoDBClient, CreateTableCommand } from "@aws-sdk/client-dynamodb";

export async function createDynamoDBTableWithWarmThroughput(
  tableName,
  partitionKey,
  sortKey,
  miscKeyAttr,
  nonKeyAttr,
  tableProvisionedReadUnits,
  tableProvisionedWriteUnits,
  tableWarmReads,
  tableWarmWrites,
  indexName,
  indexProvisionedReadUnits,
  indexProvisionedWriteUnits,
  indexWarmReads,
  indexWarmWrites,
```

```
  region = "us-east-1"
) {
  try {
    const ddbClient = new DynamoDBClient({ region: region });
    const command = new CreateTableCommand({
      TableName: tableName,
      AttributeDefinitions: [
          { AttributeName: partitionKey, AttributeType: "S" },
          { AttributeName: sortKey, AttributeType: "S" },
          { AttributeName: miscKeyAttr, AttributeType: "N" },
      ],
      KeySchema: [
          { AttributeName: partitionKey, KeyType: "HASH" },
          { AttributeName: sortKey, KeyType: "RANGE" },
      ],
      ProvisionedThroughput: {
          ReadCapacityUnits: tableProvisionedReadUnits,
          WriteCapacityUnits: tableProvisionedWriteUnits,
      },
      WarmThroughput: {
          ReadUnitsPerSecond: tableWarmReads,
          WriteUnitsPerSecond: tableWarmWrites,
      },
      GlobalSecondaryIndexes: [
          {
            IndexName: indexName,
            KeySchema: [
                { AttributeName: sortKey, KeyType: "HASH" },
                { AttributeName: miscKeyAttr, KeyType: "RANGE" },
            ],
            Projection: {
                ProjectionType: "INCLUDE",
                NonKeyAttributes: [nonKeyAttr],
            },
            ProvisionedThroughput: {
                ReadCapacityUnits: indexProvisionedReadUnits,
                WriteCapacityUnits: indexProvisionedWriteUnits,
            },
            WarmThroughput: {
                ReadUnitsPerSecond: indexWarmReads,
                WriteUnitsPerSecond: indexWarmWrites,
            },
          },
      ],
```

```
    });
    const response = await ddbClient.send(command);
    console.log(response);
    return response;
  } catch (error) {
    console.error(`Error creating table: ${error}`);
    throw error;
  }
}

// Example usage (commented out for testing)
/*
createDynamoDBTableWithWarmThroughput(
  'example-table',
  'pk',
  'sk',
  'gsiKey',
  'data',
  10, 10, 5, 5,
  'example-index',
  5, 5, 2, 2
);
*/
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para JavaScript da API.

Criar um item com TTL

O exemplo de código a seguir mostra como criar um item com TTL.

SDK para JavaScript (v3)

```
import { DynamoDBClient, PutItemCommand } from "@aws-sdk/client-dynamodb";

export function createDynamoDBItem(table_name, region, partition_key, sort_key) {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    // Get the current time in epoch second format
```

```
    const current_time = Math.floor(new Date().getTime() / 1000);

    // Calculate the expireAt time (90 days from now) in epoch second format
    const expire_at = Math.floor((new Date().getTime() + 90 * 24 * 60 * 60 * 1000) /
 1000);

    // Create DynamoDB item
    const item = {
        'partitionKey': {'S': partition_key},
        'sortKey': {'S': sort_key},
        'createdAt': {'N': current_time.toString()},
        'expireAt': {'N': expire_at.toString()}
    };

    const putItemCommand = new PutItemCommand({
        TableName: table_name,
        Item: item,
        ProvisionedThroughput: {
            ReadCapacityUnits: 1,
            WriteCapacityUnits: 1,
        },
    });

    client.send(putItemCommand, function(err, data) {
        if (err) {
            console.log("Exception encountered when creating item %s, here's what
 happened: ", data, err);
            throw err;
        } else {
            console.log("Item created successfully: %s.", data);
            return data;
        }
    });
}

// Example usage (commented out for testing)
// createDynamoDBItem('your-table-name', 'us-east-1', 'your-partition-key-value',
 'your-sort-key-value');
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK para JavaScript da API.

Excluir dados usando DELETE do PartiQL

O exemplo de código a seguir mostra como excluir dados usando declarações DELETE do PartiQL.

SDK para JavaScript (v3)

Exclua itens de uma tabela do DynamoDB usando instruções PartiQL DELETE com. AWS SDK para JavaScript

```
/**
 * This example demonstrates how to delete items from a DynamoDB table using
 PartiQL.
 * It shows different ways to delete documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Delete a single item by its partition key using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemByPartitionKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ?`,
    Parameters: [partitionKeyValue],
  };

  try {
```

```
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item:", err);
    throw err;
  }
};

/**
 * Delete an item by its composite key (partition key + sort key) using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemByCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${sortKeyName} = ?`,
    Parameters: [partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item:", err);
    throw err;
  }
};
```

```
/**
 * Delete an item with a condition to ensure the delete only happens if a condition
 is met.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param conditionAttribute - The attribute to check in the condition
 * @param conditionValue - The value to compare against in the condition
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemWithCondition = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  conditionAttribute: string,
  conditionValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${conditionAttribute} = ?`,
    Parameters: [partitionKeyValue, conditionValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted with condition successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item with condition:", err);
    throw err;
  }
};

/**
 * Batch delete multiple items using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param keys - Array of objects containing key information
 * @returns The response from the BatchExecuteStatementCommand
```

```
 */
export const batchDeleteItems = async (
  tableName: string,
  keys: Array<{
    partitionKeyName: string;
    partitionKeyValue: string | number;
    sortKeyName?: string;
    sortKeyValue?: string | number;
  }>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each delete
  const statements = keys.map((key) => {
    if (key.sortKeyName && key.sortKeyValue !== undefined) {
      return {
        Statement: `DELETE FROM "${tableName}" WHERE ${key.partitionKeyName} = ? AND
 ${key.sortKeyName} = ?`,
        Parameters: [key.partitionKeyValue, key.sortKeyValue],
      };
    } else {
      return {
        Statement: `DELETE FROM "${tableName}" WHERE ${key.partitionKeyName} = ?`,
        Parameters: [key.partitionKeyValue],
      };
    }
  });

  const params = {
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items batch deleted successfully");
    return data;
  } catch (err) {
    console.error("Error batch deleting items:", err);
    throw err;
  }
};

/**
```

```
 * Delete multiple items that match a filter condition.
 * Note: This performs a scan operation which can be expensive on large tables.
 *
 * @param tableName - The name of the DynamoDB table
 * @param filterAttribute - The attribute to filter on
 * @param filterValue - The value to filter by
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemsByFilter = async (
  tableName: string,
  filterAttribute: string,
  filterValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${filterAttribute} = ?`,
    Parameters: [filterValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items deleted by filter successfully");
    return data;
  } catch (err) {
    console.error("Error deleting items by filter:", err);
    throw err;
  }
};

/**
 * Example usage showing how to delete items with different index types
 */
export const deleteExamples = async () => {
  // Delete an item by partition key (simple primary key)
  await deleteItemByPartitionKey("UsersTable", "userId", "user123");

  // Delete an item by composite key (partition key + sort key)
  await deleteItemByCompositeKey(
    "OrdersTable",
    "orderId",
    "order456",
    "productId",
```

```
    "prod789"
  );

  // Delete with a condition
  await deleteItemWithCondition(
    "UsersTable",
    "userId",
    "user789",
    "userStatus",
    "inactive"
  );

  // Batch delete multiple items
  await batchDeleteItems("UsersTable", [
    { partitionKeyName: "userId", partitionKeyValue: "user234" },
    { partitionKeyName: "userId", partitionKeyValue: "user345" },
  ]);

  // Batch delete items with composite keys
  await batchDeleteItems("OrdersTable", [
    {
      partitionKeyName: "orderId",
      partitionKeyValue: "order567",
      sortKeyName: "productId",
      sortKeyValue: "prod123",
    },
    {
      partitionKeyName: "orderId",
      partitionKeyValue: "order678",
      sortKeyName: "productId",
      sortKeyValue: "prod456",
    },
  ]);

  // Delete items by filter (use with caution)
  await deleteItemsByFilter("UsersTable", "userStatus", "deleted");
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchExecuteStatement

- ExecuteStatement

Inserir dados usando INSERT do PartiQL

O exemplo de código a seguir mostra como inserir dados usando declarações INSERT do PartiQL.

SDK para JavaScript (v3)

Insira itens em uma tabela do DynamoDB usando instruções PartiQL INSERT com. AWS SDK para JavaScript

```
/**
 * This example demonstrates how to insert items into a DynamoDB table using
 PartiQL.
 * It shows different ways to insert documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Insert a single item into a DynamoDB table using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param item - The item to insert
 * @returns The response from the ExecuteStatementCommand
 */
export const insertItem = async (tableName: string, item: Record<string, any>) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Convert the item to a string representation for PartiQL
  const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');

  const params = {
    Statement: `INSERT INTO "${tableName}" VALUE ${itemString}`,
  };

  try {
```

```
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item inserted successfully");
    return data;
  } catch (err) {
    console.error("Error inserting item:", err);
    throw err;
  }
};

/**
 * Insert multiple items into a DynamoDB table using PartiQL batch operation.
 * This is more efficient than inserting items one by one.
 *
 * @param tableName - The name of the DynamoDB table
 * @param items - Array of items to insert
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchInsertItems = async (tableName: string, items: Record<string,
 any>[]) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each item
  const statements = items.map((item) => {
    const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');
    return {
      Statement: `INSERT INTO "${tableName}" VALUE ${itemString}`,
    };
  });

  const params = {
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items inserted successfully");
    return data;
  } catch (err) {
    console.error("Error batch inserting items:", err);
    throw err;
  }
};
```

```
/**
 * Insert an item with a condition to prevent overwriting existing items.
 * This is useful for ensuring you don't accidentally overwrite data.
 *
 * @param tableName - The name of the DynamoDB table
 * @param item - The item to insert
 * @param partitionKeyName - The name of the partition key attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const insertItemWithCondition = async (
  tableName: string,
  item: Record<string, any>,
  partitionKeyName: string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');
  const partitionKeyValue = JSON.stringify(item[partitionKeyName]);

  const params = {
    Statement: `INSERT INTO "${tableName}" VALUE ${itemString} WHERE
 attribute_not_exists(${partitionKeyName})`,
    Parameters: [{ S: partitionKeyValue }],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item inserted with condition successfully");
    return data;
  } catch (err) {
    console.error("Error inserting item with condition:", err);
    throw err;
  }
};

/**
 * Example usage showing how to insert items with different index types
 */
export const insertExamples = async () => {
  // Example table with a simple primary key (just partition key)
  const simpleKeyItem = {
    userId: "user123",
    name: "John Doe",
```

```
    email: "john@example.com",
  };
  await insertItem("UsersTable", simpleKeyItem);

  // Example table with composite key (partition key + sort key)
  const compositeKeyItem = {
    orderId: "order456",
    productId: "prod789",
    quantity: 2,
    price: 29.99,
  };
  await insertItem("OrdersTable", compositeKeyItem);

  // Example with Global Secondary Index (GSI)
  // The GSI might be on the email attribute
  const gsiItem = {
    userId: "user789",
    email: "jane@example.com",
    name: "Jane Smith",
    userType: "premium",  // This could be part of a GSI
  };
  await insertItem("UsersTable", gsiItem);

  // Example with Local Secondary Index (LSI)
  // LSI uses the same partition key but different sort key
  const lsiItem = {
    orderId: "order567",  // Partition key
    productId: "prod123", // Sort key for the table
    orderDate: "2023-11-15", // Potential sort key for an LSI
    quantity: 1,
    price: 19.99,
  };
  await insertItem("OrdersTable", lsiItem);

  // Batch insert example with multiple items
  const batchItems = [
    {
      userId: "user234",
      name: "Alice Johnson",
      email: "alice@example.com",
    },
    {
      userId: "user345",
      name: "Bob Williams",
```

```
      email: "bob@example.com",
    },
  ];
  await batchInsertItems("UsersTable", batchItems);
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchExecuteStatement
  - ExecuteStatement

Invocar uma função do Lambda em um navegador

O exemplo de código a seguir mostra como invocar uma AWS Lambda função em um navegador.

SDK para JavaScript (v3)

Você pode criar um aplicativo baseado em navegador que usa uma AWS Lambda função para atualizar uma tabela do Amazon DynamoDB com as seleções do usuário. Este aplicativo usa AWS SDK para JavaScript v3.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda

Executar operações de consulta avançada

O exemplo de código a seguir mostra como realizar operações de consulta avançada no DynamoDB.

- Consulte tabelas usando várias técnicas de filtragem e condição.
- Implemente a paginação para grandes conjuntos de resultados.
- Use índices secundários globais para padrões de acesso alternativos.
- Aplique controles de consistência com base nos requisitos da aplicação.

SDK para JavaScript (v3)

Consulte com leituras fortemente consistentes usando AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with configurable read consistency
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {boolean} useConsistentRead - Whether to use strongly consistent reads
 * @returns {Promise<Object>} - The query response
 */
async function queryWithConsistentRead(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  useConsistentRead = false
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue }
      },
      ConsistentRead: useConsistentRead
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
```

```
    console.error(`Error querying with consistent read: ${error}`);
    throw error;
  }
}
```

Consulte usando um Índice Secundário Global com AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table using the primary key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} userId - The user ID to query by (partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryTable(
  config,
  tableName,
  userId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the base table
    const input = {
      TableName: tableName,
      KeyConditionExpression: "user_id = :userId",
      ExpressionAttributeValues: {
        ":userId": { S: userId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying table: ${error}`);
    throw error;
  }
```

```
}

/**
 * Queries a DynamoDB Global Secondary Index (GSI)
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} indexName - The name of the GSI to query
 * @param {string} gameId - The game ID to query by (GSI partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryGSI(
  config,
  tableName,
  indexName,
  gameId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the GSI
    const input = {
      TableName: tableName,
      IndexName: indexName,
      KeyConditionExpression: "game_id = :gameId",
      ExpressionAttributeValues: {
        ":gameId": { S: gameId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying GSI: ${error}`);
    throw error;
  }
}
```

Consulta com paginação usando. AWS SDK para JavaScript

```
/**
 * Example demonstrating how to handle large query result sets in DynamoDB using
 pagination
 *
 * This example shows:
 * - How to use pagination to handle large result sets
 * - How to use LastEvaluatedKey to retrieve the next page of results
 * - How to construct subsequent query requests using ExclusiveStartKey
 */
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with pagination to handle large result sets
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number} pageSize - Number of items per page
 * @returns {Promise<Array>} - All items from the query
 */
async function queryWithPagination(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  pageSize = 25
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize variables for pagination
    let lastEvaluatedKey = undefined;
    const allItems = [];
    let pageCount = 0;

    // Loop until all pages are retrieved
    do {
      // Construct the query input
      const input = {
        TableName: tableName,
```

```
        KeyConditionExpression: "#pk = :pkValue",
        Limit: pageSize,
        ExpressionAttributeNames: {
          "#pk": partitionKeyName
        },
        ExpressionAttributeValues: {
          ":pkValue": { S: partitionKeyValue }
        }
      };

      // Add ExclusiveStartKey if we have a LastEvaluatedKey from a previous query
      if (lastEvaluatedKey) {
        input.ExclusiveStartKey = lastEvaluatedKey;
      }

      // Execute the query
      const command = new QueryCommand(input);
      const response = await client.send(command);

      // Process the current page of results
      pageCount++;
      console.log(`Processing page ${pageCount} with ${response.Items.length}
 items`);

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems.push(...response.Items);
      }

      // Get the LastEvaluatedKey for the next page
      lastEvaluatedKey = response.LastEvaluatedKey;

    } while (lastEvaluatedKey); // Continue until there are no more pages

    console.log(`Query complete. Retrieved ${allItems.length} items in ${pageCount}
 pages.`);
    return allItems;
  } catch (error) {
    console.error(`Error querying with pagination: ${error}`);
    throw error;
  }
}

/**
```

```
 * Example usage:
 *
 * // Query all items in the "AWS DynamoDB" forum with pagination
 * const allItems = await queryWithPagination(
 *   { region: "us-west-2" },
 *   "ForumThreads",
 *   "ForumName",
 *   "AWS DynamoDB",
 *   25 // 25 items per page
 * );
 *
 * console.log(`Total items retrieved: ${allItems.length}`);
 *
 * // Notes on pagination:
 * // - LastEvaluatedKey contains the primary key of the last evaluated item
 * // - When LastEvaluatedKey is undefined/null, there are no more items to retrieve
 * // - ExclusiveStartKey tells DynamoDB where to start the next page
 * // - Pagination helps manage memory usage for large result sets
 * // - Each page requires a separate network request to DynamoDB
 */

module.exports = { queryWithPagination };
```

Consulte com filtros complexos usando AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with a complex filter expression
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number|string} minViews - Minimum number of views for filtering
 * @param {number|string} minReplies - Minimum number of replies for filtering
 * @param {string} requiredTag - Tag that must be present in the item's tags set
 * @returns {Promise<Object>} - The query response
 */
async function queryWithComplexFilter(
  config,
  tableName,
```

```
  partitionKeyName,
  partitionKeyValue,
  minViews,
  minReplies,
  requiredTag
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      FilterExpression: "views >= :minViews AND replies >= :minReplies AND
 contains(tags, :tag)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":minViews": { N: minViews.toString() },
        ":minReplies": { N: minReplies.toString() },
        ":tag": { S: requiredTag }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with complex filter: ${error}`);
    throw error;
  }
}
```

Consulte com uma expressão de filtro construída dinamicamente usando AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

async function queryWithDynamicFilter(
```

```
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  sortKeyValue,
  filterParams = {}
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize filter expression components
    let filterExpressions = [];
    const expressionAttributeValues = {
      ":pkValue": { S: partitionKeyValue },
      ":skValue": { S: sortKeyValue }
    };
    const expressionAttributeNames = {
      "#pk": partitionKeyName,
      "#sk": sortKeyName
    };

    // Add status filter if provided
    if (filterParams.status) {
      filterExpressions.push("status = :status");
      expressionAttributeValues[":status"] = { S: filterParams.status };
    }

    // Add minimum views filter if provided
    if (filterParams.minViews !== undefined) {
      filterExpressions.push("views >= :minViews");
      expressionAttributeValues[":minViews"] = { N:
 filterParams.minViews.toString() };
    }

    // Add author filter if provided
    if (filterParams.author) {
      filterExpressions.push("author = :author");
      expressionAttributeValues[":author"] = { S: filterParams.author };
    }

    // Construct the query input
    const input = {
```

```
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND #sk = :skValue"
    };

    // Add filter expression if any filters were provided
    if (filterExpressions.length > 0) {
      input.FilterExpression = filterExpressions.join(" AND ");
    }

    // Add expression attribute names and values
    input.ExpressionAttributeNames = expressionAttributeNames;
    input.ExpressionAttributeValues = expressionAttributeValues;

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with dynamic filter: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Executar operações de lista

O exemplo de código a seguir mostra como realizar operações de lista no DynamoDB.

- Adicione elementos a um atributo de lista.
- Remova elementos de um atributo de lista.
- Atualize elementos específicos em uma lista por índice.
- Use as funções list.append e list.index.

SDK para JavaScript (v3)

Demonstre as operações de lista usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
```

```
  UpdateCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Append elements to a list attribute.
 *
 * This function demonstrates how to use the list_append function to add elements
 * to the end of a list.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} values - The values to append to the list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function appendToList(
  config,
  tableName,
  key,
  listName,
  values
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} =
 list_append(if_not_exists(${listName}, :empty_list), :values)`,
    ExpressionAttributeValues: {
      ":empty_list": [],
      ":values": values
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));
```

```
  return response;
}

/**
 * Prepend elements to a list attribute.
 *
 * This function demonstrates how to use the list_append function to add elements
 * to the beginning of a list.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} values - The values to prepend to the list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function prependToList(
  config,
  tableName,
  key,
  listName,
  values
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  // Note: To prepend, we put the new values first in the list_append function
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} = list_append(:values,
 if_not_exists(${listName}, :empty_list))`,
    ExpressionAttributeValues: {
      ":empty_list": [],
      ":values": values
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));
```

```
  return response;
}

/**
 * Update a specific element in a list by index.
 *
 * This function demonstrates how to update a specific element in a list
 * using the index notation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} index - The index of the element to update
 * @param {any} value - The new value for the element
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateListElement(
  config,
  tableName,
  key,
  listName,
  index,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using index notation
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName}[${index}] = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));
```

```
  return response;
}

/**
 * Remove an element from a list by index.
 *
 * This function demonstrates how to remove a specific element from a list
 * using the REMOVE action with index notation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} index - The index of the element to remove
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function removeListElement(
  config,
  tableName,
  key,
  listName,
  index
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using REMOVE with index notation
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `REMOVE ${listName}[${index}]`,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Concatenate two lists.
 *
```

```
 * This function demonstrates how to concatenate two lists using the list_append
 function.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName1 - The name of the first list attribute
 * @param {string} listName2 - The name of the second list attribute
 * @param {string} resultListName - The name of the attribute to store the
 concatenated list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function concatenateLists(
  config,
  tableName,
  key,
  listName1,
  listName2,
  resultListName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${resultListName} =
 list_append(if_not_exists(${listName1}, :empty_list),
 if_not_exists(${listName2}, :empty_list))`,
    ExpressionAttributeValues: {
      ":empty_list": []
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
```

```
 * Create a nested list structure.
 *
 * This function demonstrates how to create and work with nested lists.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} nestedLists - An array of arrays to create a nested list structure
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function createNestedList(
  config,
  tableName,
  key,
  listName,
  nestedLists
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters to create a nested list
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} = :nested_lists`,
    ExpressionAttributeValues: {
      ":nested_lists": nestedLists
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update an element in a nested list.
 *
 * This function demonstrates how to update an element in a nested list
 * using multiple index notations.
```

```
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} outerIndex - The index in the outer list
 * @param {number} innerIndex - The index in the inner list
 * @param {any} value - The new value for the element
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateNestedListElement(
  config,
  tableName,
  key,
  listName,
  outerIndex,
  innerIndex,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using multiple index notations
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName}[${outerIndex}][${innerIndex}] = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
```

```
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Exemplo de uso de operações de lista com AWS SDK para JavaScript.

```
/**
 * Example of how to work with lists in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "UserProfiles";
  const key = { UserId: "U12345" };

  console.log("Demonstrating list operations in DynamoDB");

  try {
```

```
    // Example 1: Append elements to a list
    console.log("\nExample 1: Appending elements to a list");
    const response1 = await appendToList(
      config,
      tableName,
      key,
      "RecentSearches",
      ["laptop", "headphones", "monitor"]
    );

    console.log("Appended to list:", response1.Attributes);

    // Example 2: Prepend elements to a list
    console.log("\nExample 2: Prepending elements to a list");
    const response2 = await prependToList(
      config,
      tableName,
      key,
      "RecentSearches",
      ["keyboard", "mouse"]
    );

    console.log("Prepended to list:", response2.Attributes);

    // Get the current state of the item
    let currentItem = await getItem(config, tableName, key);
    console.log("\nCurrent state of RecentSearches:", currentItem?.RecentSearches);

    // Example 3: Update a specific element in a list
    console.log("\nExample 3: Updating a specific element in a list");
    const response3 = await updateListElement(
      config,
      tableName,
      key,
      "RecentSearches",
      0, // Update the first element
      "mechanical keyboard" // New value
    );

    console.log("Updated list element:", response3.Attributes);

    // Example 4: Remove an element from a list
    console.log("\nExample 4: Removing an element from a list");
    const response4 = await removeListElement(
```

```
    config,
    tableName,
    key,
    "RecentSearches",
    2 // Remove the third element
  );

  console.log("List after removing element:", response4.Attributes);

  // Example 5: Create and concatenate lists
  console.log("\nExample 5: Creating and concatenating lists");

  // First, create two separate lists
  await updateWithMultipleActions(
    config,
    tableName,
    key,
    "SET WishList = :wishlist, SavedItems = :saveditems",
    null,
    {
      ":wishlist": ["gaming laptop", "wireless earbuds"],
      ":saveditems": ["smartphone", "tablet"]
    }
  );

  // Then, concatenate them
  const response5 = await concatenateLists(
    config,
    tableName,
    key,
    "WishList",
    "SavedItems",
    "AllItems"
  );

  console.log("Concatenated lists:", response5.Attributes);

  // Example 6: Create a nested list structure
  console.log("\nExample 6: Creating a nested list structure");
  const response6 = await createNestedList(
    config,
    tableName,
    key,
    "Categories",
```

```
      [
        ["Electronics", "Computers", "Accessories"],
        ["Books", "Magazines", "E-books"],
        ["Clothing", "Shoes", "Watches"]
      ]
    );

    console.log("Created nested list:", response6.Attributes);

    // Example 7: Update an element in a nested list
    console.log("\nExample 7: Updating an element in a nested list");
    const response7 = await updateNestedListElement(
      config,
      tableName,
      key,
      "Categories",
      0, // First inner list
      1, // Second element in that list
      "Laptops" // New value
    );

    console.log("Updated nested list element:", response7.Attributes);

    // Get the final state of the item
    currentItem = await getItem(config, tableName, key);
    console.log("\nFinal state of the item:", JSON.stringify(currentItem, null, 2));

    // Explain list operations
    console.log("\nKey points about list operations in DynamoDB:");
    console.log("1. Use list_append to add elements to a list");
    console.log("2. To append elements, use list_append(existingList,
newElements)");
    console.log("3. To prepend elements, use list_append(newElements,
existingList)");
    console.log("4. Use if_not_exists to handle cases where the list might not exist
yet");
    console.log("5. Use index notation (list[0]) to access or update specific
elements");
    console.log("6. Use REMOVE with index notation to remove elements from a list");
    console.log("7. Lists can contain elements of different types");
    console.log("8. Lists can be nested (lists of lists)");
    console.log("9. Use multiple index notations (list[0][1]) to access nested list
elements");
```

```
  } catch (error) {
    console.error("Error:", error);
  }
}

/**
 * Helper function for the examples.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} updateExpression - The update expression
 * @param {Object} expressionAttributeNames - Expression attribute name placeholders
 * @param {Object} expressionAttributeValues - Expression attribute value
 placeholders
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleActions(
  config,
  tableName,
  key,
  updateExpression,
  expressionAttributeNames,
  expressionAttributeValues
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Prepare the update parameters
  const updateParams = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ReturnValues: "UPDATED_NEW"
  };

  // Add expression attribute names if provided
  if (expressionAttributeNames) {
    updateParams.ExpressionAttributeNames = expressionAttributeNames;
  }

  // Add expression attribute values if provided
  if (expressionAttributeValues) {
```

```
    updateParams.ExpressionAttributeValues = expressionAttributeValues;
  }

  // Execute the update
  const response = await docClient.send(new UpdateCommand(updateParams));

  return response;
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

Executar operações de mapa

O exemplo de código a seguir mostra como realizar operações de mapa no DynamoDB.

- Adicione e atualize atributos aninhados em estruturas de mapa.
- Remova campos específicos dos mapas.
- Trabalhe com atributos de mapa profundamente aninhados.

SDK para JavaScript (v3)

Demonstre as operações do mapa usando AWS SDK para JavaScript.

```
/**
 * Example of updating map attributes in DynamoDB.
 *
 * This module demonstrates how to update map attributes that may not exist,
 * how to update nested attributes, and how to handle various map update scenarios.
 */

const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
```

```
 * Update a map attribute safely, handling the case where the map might not exist.
 *
 * This function demonstrates using the if_not_exists function to safely update
 * a map attribute that might not exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {string} mapKey - The key within the map to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMapAttributeSafe(
  config,
  tableName,
  key,
  mapName,
  mapKey,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with if_not_exists
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName}.${mapKey} = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return response;
  } catch (error) {
    // If the error is because the map doesn't exist, create it
    if (error.name === "ValidationException" &&
```

```
        error.message.includes("The document path provided in the update expression
 is invalid")) {

      // Create the map with the specified key-value pair
      const createParams = {
        TableName: tableName,
        Key: key,
        UpdateExpression: `SET ${mapName} = :map`,
        ExpressionAttributeValues: {
          ":map": { [mapKey]: value }
        },
        ReturnValues: "UPDATED_NEW"
      };

      return await docClient.send(new UpdateCommand(createParams));
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Update a map attribute using the if_not_exists function.
 *
 * This function demonstrates a more elegant approach using if_not_exists
 * to handle the case where the map doesn't exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {string} mapKey - The key within the map to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMapAttributeWithIfNotExists(
  config,
  tableName,
  key,
  mapName,
  mapKey,
  value
) {
```

```
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // A single UpdateExpression can't reference both a map and a path inside that
  // same map (for example `${mapName}` and `${mapName}.${mapKey}`): DynamoDB
  // rejects overlapping document paths with a ValidationException. Instead,
  // perform the update in two steps.

  // Step 1: create the map with an empty value if it doesn't already exist.
  await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName} = if_not_exists(${mapName}, :emptyMap)`,
    ExpressionAttributeValues: {
      ":emptyMap": {}
    }
  }));

  // Step 2: set the key within the now-guaranteed-to-exist map.
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName}.${mapKey} = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Add a value to a deeply nested map, creating parent maps if they don't exist.
 *
 * This function demonstrates how to update a deeply nested attribute,
 * creating any parent maps that don't exist along the way.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {Object} key - The key of the item to update
 * @param {string[]} path - The path to the nested attribute as an array of keys
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function addToNestedMap(
  config,
  tableName,
  key,
  path,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = "SET";
  const expressionAttributeValues = {};

  // For each level in the path, create a map if it doesn't exist
  for (let i = 0; i < path.length; i++) {
    const currentPath = path.slice(0, i + 1).join(".");
    const parentPath = i > 0 ? path.slice(0, i).join(".") : null;

    if (parentPath) {
      updateExpression += ` ${parentPath} = if_not_exists(${parentPath}, :emptyMap
${i}),`;
      expressionAttributeValues[`:emptyMap${i}`] = {};
    }
  }

  // Set the final value
  const fullPath = path.join(".");
  updateExpression += ` ${fullPath} = :value`;
  expressionAttributeValues[":value"] = value;

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
```

```
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update multiple fields in a map attribute in a single operation.
 *
 * This function demonstrates how to update multiple fields in a map
 * in a single DynamoDB operation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {Object} updates - Object containing key-value pairs to update
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMultipleMapFields(
  config,
  tableName,
  key,
  mapName,
  updates
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = `SET ${mapName} = if_not_exists(${mapName}, :emptyMap)`;
  const expressionAttributeValues = {
    ":emptyMap": {}
  };

  // Add each update to the expression
  Object.entries(updates).forEach(([field, value], index) => {
    updateExpression += `, ${mapName}.${field} = :val${index}`;
    expressionAttributeValues[`:val${index}`] = value;
  });
```

```
  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
```

```
  return response.Item || null;
}

/**
 * Example of how to use the map attribute update functions.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Users";
  const key = { UserId: "U12345" };

  console.log("Demonstrating different approaches to update map attributes in
 DynamoDB");

  try {
    // Example 1: Update a map attribute that might not exist (two-step approach)
    console.log("\nExample 1: Updating a map attribute that might not exist (two-
step approach)");
    const response1 = await updateMapAttributeSafe(
      config,
      tableName,
      key,
      "Preferences",
      "Theme",
      "Dark"
    );

    console.log("Updated preferences:", response1.Attributes);

    // Example 2: Update a map attribute using if_not_exists (elegant approach)
    console.log("\nExample 2: Updating a map attribute using if_not_exists (elegant
 approach)");
    const response2 = await updateMapAttributeWithIfNotExists(
      config,
      tableName,
      key,
      "Settings",
      "NotificationsEnabled",
      true
    );

    console.log("Updated settings:", response2.Attributes);
```

```
    // Example 3: Update a deeply nested attribute
    console.log("\nExample 3: Updating a deeply nested attribute");
    const response3 = await addToNestedMap(
      config,
      tableName,
      key,
      ["Profile", "Address", "City"],
      "Seattle"
    );

    console.log("Updated nested attribute:", response3.Attributes);

    // Example 4: Update multiple fields in a map
    console.log("\nExample 4: Updating multiple fields in a map");
    const response4 = await updateMultipleMapFields(
      config,
      tableName,
      key,
      "ContactInfo",
      {
        Email: "user@example.com",
        Phone: "555-123-4567",
        PreferredContact: "Email"
      }
    );

    console.log("Updated multiple fields:", response4.Attributes);

    // Get the final state of the item
    console.log("\nFinal state of the item:");
    const item = await getItem(config, tableName, key);
    console.log(JSON.stringify(item, null, 2));

    // Explain the benefits of different approaches
    console.log("\nKey points about updating map attributes:");
    console.log("1. Use if_not_exists to handle maps that might not exist");
    console.log("2. Multiple updates can be combined in a single operation");
    console.log("3. Deeply nested attributes require creating parent maps");
    console.log("4. DynamoDB expressions are atomic - the entire update succeeds or
fails");
    console.log("5. Using a single operation is more efficient than multiple
separate updates");

  } catch (error) {
```

```
    console.error("Error:", error);
  }
}

// Export the functions
module.exports = {
  updateMapAttributeSafe,
  updateMapAttributeWithIfNotExists,
  addToNestedMap,
  updateMultipleMapFields,
  getItem,
  exampleUsage
};

// Run the example if this file is executed directly
if (require.main === module) {
  exampleUsage();
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

Executar operações de conjunto

O exemplo de código a seguir mostra como realizar operações de conjunto no DynamoDB.

- Adicione elementos a um atributo de conjunto.
- Remova elementos de um atributo de conjunto.
- Use as operações ADD e DELETE com conjuntos.

SDK para JavaScript (v3)

Demonstre as operações de conjunto usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");
```

```
/**
 * Add elements to a set attribute.
 *
 * This function demonstrates using the ADD operation to add elements to a set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The values to add to the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function addToSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using ADD
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `ADD ${setName} :values`,
    ExpressionAttributeValues: {
```

```
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Remove elements from a set attribute.
 *
 * This function demonstrates using the DELETE operation to remove elements from a
 set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The values to remove from the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function removeFromSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
```

```
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using DELETE
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `DELETE ${setName} :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Create a new set attribute with initial values.
 *
 * This function demonstrates using the SET operation to create a new set attribute.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The initial values for the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function createSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
```

```
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using SET
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${setName} = :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Replace an entire set attribute with a new set of values.
 *
 * This function demonstrates using the SET operation to replace an entire set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The new values for the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
```

```
 */
async function replaceSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // This is the same as createSet, but included for clarity of intent
  return await createSet(config, tableName, key, setName, values, setType);
}

/**
 * Remove the last element from a set and handle the empty set case.
 *
 * This function demonstrates what happens when you delete the last element of a
 set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @returns {Promise<Object>} - The result of the operation
 */
async function removeLastElementFromSet(
  config,
  tableName,
  key,
  setName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // First, get the current item to check the set
  const currentItem = await getItem(config, tableName, key);

  // Check if the set exists and has elements
  if (!currentItem || !currentItem[setName] || currentItem[setName].size === 0) {
    return {
      success: false,
      message: "Set doesn't exist or is already empty",
      item: currentItem
```

```
    };
  }

  // Get the set values
  const setValues = Array.from(currentItem[setName]);

  // If there's only one element left, remove the attribute entirely
  if (setValues.length === 1) {
    // Define the update parameters to remove the attribute
    const params = {
      TableName: tableName,
      Key: key,
      UpdateExpression: `REMOVE ${setName}`,
      ReturnValues: "UPDATED_NEW"
    };

    // Perform the update operation
    await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Last element removed, attribute has been deleted",
      removedValue: setValues[0]
    };
  } else {
    // Otherwise, remove just the last element
    // Create a set with just the last element
    const lastElement = setValues[setValues.length - 1];
    const setType = typeof lastElement === 'number' ? 'number' : 'string';

    // Remove the last element
    const response = await removeFromSet(
      config,
      tableName,
      key,
      setName,
      [lastElement],
      setType
    );

    return {
      success: true,
      message: "Last element removed, set still contains elements",
      removedValue: lastElement,
```

```
      remainingSet: response.Attributes[setName]
    };
  }
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Exemplo de uso de operações definidas com AWS SDK para JavaScript.

```
/**
 * Example of how to work with sets in DynamoDB.
 */
```

```
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Users";
  const key = { UserId: "U12345" };

  console.log("Demonstrating set operations in DynamoDB");

  try {
    // Example 1: Create a string set
    console.log("\nExample 1: Creating a string set");
    const response1 = await createSet(
      config,
      tableName,
      key,
      "Interests",
      ["Reading", "Hiking", "Cooking"],
      "string"
    );

    console.log("Created set:", response1.Attributes);

    // Example 2: Add elements to a set
    console.log("\nExample 2: Adding elements to a set");
    const response2 = await addToSet(
      config,
      tableName,
      key,
      "Interests",
      ["Photography", "Travel"],
      "string"
    );

    console.log("Updated set after adding elements:", response2.Attributes);

    // Example 3: Remove elements from a set
    console.log("\nExample 3: Removing elements from a set");
    const response3 = await removeFromSet(
      config,
      tableName,
      key,
      "Interests",
      ["Cooking"],
      "string"
```

```
    );

    console.log("Updated set after removing elements:", response3.Attributes);

    // Example 4: Create a number set
    console.log("\nExample 4: Creating a number set");
    const response4 = await createSet(
      config,
      tableName,
      key,
      "FavoriteNumbers",
      [7, 42, 99],
      "number"
    );

    console.log("Created number set:", response4.Attributes);

    // Example 5: Replace an entire set
    console.log("\nExample 5: Replacing an entire set");
    const response5 = await replaceSet(
      config,
      tableName,
      key,
      "Interests",
      ["Gaming", "Movies", "Music"],
      "string"
    );

    console.log("Replaced set:", response5.Attributes);

    // Example 6: Remove the last element from a set
    console.log("\nExample 6: Removing the last element from a set");

    // First, create a set with just one element
    await createSet(
      config,
      tableName,
      { UserId: "U67890" },
      "Tags",
      ["LastTag"],
      "string"
    );

    // Then, remove the last element
```

```
    const response6 = await removeLastElementFromSet(
      config,
      tableName,
      { UserId: "U67890" },
      "Tags"
    );

    console.log(response6.message);
    console.log("Removed value:", response6.removedValue);

    // Get the final state of the items
    console.log("\nFinal state of the items:");
    const item1 = await getItem(config, tableName, key);
    console.log("User U12345:", JSON.stringify(item1, null, 2));

    const item2 = await getItem(config, tableName, { UserId: "U67890" });
    console.log("User U67890:", JSON.stringify(item2, null, 2));

    // Explain set operations
    console.log("\nKey points about set operations in DynamoDB:");
    console.log("1. Use ADD to add elements to a set (duplicates are automatically
 removed)");
    console.log("2. Use DELETE to remove elements from a set");
    console.log("3. Use SET to create a new set or replace an existing one");
    console.log("4. DynamoDB supports three types of sets: string sets, number sets,
 and binary sets");
    console.log("5. When you delete the last element from a set, the attribute
 remains as an empty set");
    console.log("6. To remove an empty set, use the REMOVE operation");
    console.log("7. Sets automatically maintain unique values (no duplicates)");
    console.log("8. You cannot mix data types within a set");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute instruções PartiQL em lote.

```
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DescribeTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);
const tableName = "Cities";

export const main = async (confirmAll = false) => {
  /**
```

```
    * Delete table if it exists.
    */
  try {
    await client.send(new DescribeTableCommand({ TableName: tableName }));
    // If no error was thrown, the table exists.
    const input = new ScenarioInput(
      "deleteTable",
      `A table named ${tableName} already exists. If you choose not to delete
this table, the scenario cannot continue. Delete it?`,
      { type: "confirm", confirmAll },
    );
    const deleteTable = await input.handle({}, { confirmAll });
    if (deleteTable) {
      await client.send(new DeleteTableCommand({ tableName }));
    } else {
      console.warn(
        "Scenario could not run. Either delete ${tableName} or provide a unique
 table name.",
      );
      return;
    }
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceNotFoundException"
    ) {
      // Do nothing. This means the table is not there.
    } else {
      throw caught;
    }
  }

  /**
   * Create a table.
   */

  log("Creating a table.");
  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
```

```
    AttributeDefinitions: [
      {
        AttributeName: "name",
        // 'S' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "S",
      },
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [{ AttributeName: "name", KeyType: "HASH" }],
  });
  await client.send(createTableCommand);
  log(`Table created: ${tableName}.`);

  /**
   * Wait until the table is active.
   */

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Insert items.
   */

  log("Inserting cities into the table.");
  const addItemsStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.insert.html
    Statements: [
      {
        Statement: `INSERT INTO ${tableName} value {'name':?, 'population':?}`,
        Parameters: ["Alachua", 10712],
      },
      {
```

```
        Statement: `INSERT INTO ${tableName} value {'name':?, 'population':?}`,
        Parameters: ["High Springs", 6415],
      },
    ],
  });
  await docClient.send(addItemsStatementCommand);
  log("Cities inserted.");

  /**
   * Select items.
   */

  log("Selecting cities from the table.");
  const selectItemsStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.select.html
    Statements: [
      {
        Statement: `SELECT * FROM ${tableName} WHERE name=?`,
        Parameters: ["Alachua"],
      },
      {
        Statement: `SELECT * FROM ${tableName} WHERE name=?`,
        Parameters: ["High Springs"],
      },
    ],
  });
  const selectItemResponse = await docClient.send(selectItemsStatementCommand);
  log(
    `Got cities: ${selectItemResponse.Responses.map(
      (r) => `${r.Item.name} (${r.Item.population})`,
    ).join(", ")}`,
  );

  /**
   * Update items.
   */

  log("Modifying the populations.");
  const updateItemStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.update.html
    Statements: [
      {
```

```
        Statement: `UPDATE ${tableName} SET population=? WHERE name=?`,
        Parameters: [10, "Alachua"],
      },
      {
        Statement: `UPDATE ${tableName} SET population=? WHERE name=?`,
        Parameters: [5, "High Springs"],
      },
    ],
  });
  await docClient.send(updateItemStatementCommand);
  log("Updated cities.");

  /**
   * Delete the items.
   */

  log("Deleting the cities.");
  const deleteItemStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.delete.html
    Statements: [
      {
        Statement: `DELETE FROM ${tableName} WHERE name=?`,
        Parameters: ["Alachua"],
      },
      {
        Statement: `DELETE FROM ${tableName} WHERE name=?`,
        Parameters: ["High Springs"],
      },
    ],
  });
  await docClient.send(deleteItemStatementCommand);
  log("Cities deleted.");

  /**
   * Delete the table.
   */

  log("Deleting the table.");
  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para JavaScript da API.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute instruções PartiQL individuais.

```
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DescribeTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);
```

```
const log = (msg) => console.log(`[SCENARIO] ${msg}`);
const tableName = "SingleOriginCoffees";

export const main = async (confirmAll = false) => {
  /**
   * Delete table if it exists.
   */
  try {
    await client.send(new DescribeTableCommand({ TableName: tableName }));
    // If no error was thrown, the table exists.
    const input = new ScenarioInput(
      "deleteTable",
      `A table named ${tableName} already exists. If you choose not to delete
this table, the scenario cannot continue. Delete it?`,
      { type: "confirm", confirmAll },
    );
    const deleteTable = await input.handle({});
    if (deleteTable) {
      await client.send(new DeleteTableCommand({ tableName }));
    } else {
      console.warn(
        "Scenario could not run. Either delete ${tableName} or provide a unique
 table name.",
      );
      return;
    }
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceNotFoundException"
    ) {
      // Do nothing. This means the table is not there.
    } else {
      throw caught;
    }
  }

  /**
   * Create a table.
   */

  log("Creating a table.");
  const createTableCommand = new CreateTableCommand({
```

```
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "varietal",
        // 'S' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "S",
      },
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [{ AttributeName: "varietal", KeyType: "HASH" }],
  });
  await client.send(createTableCommand);
  log(`Table created: ${tableName}.`);

  /**
   * Wait until the table is active.
   */

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Insert an item.
   */

  log("Inserting a coffee into the table.");
  const addItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.insert.html
```

```
    Statement: `INSERT INTO ${tableName} value {'varietal':?, 'profile':?}`,
    Parameters: ["arabica", ["chocolate", "floral"]],
  });
  await client.send(addItemStatementCommand);
  log("Coffee inserted.");

  /**
   * Select an item.
   */

  log("Selecting the coffee from the table.");
  const selectItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.select.html
    Statement: `SELECT * FROM ${tableName} WHERE varietal=?`,
    Parameters: ["arabica"],
  });
  const selectItemResponse = await docClient.send(selectItemStatementCommand);
  log(`Got coffee: ${JSON.stringify(selectItemResponse.Items[0])}`);

  /**
   * Update the item.
   */

  log("Add a flavor profile to the coffee.");
  const updateItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.update.html
    Statement: `UPDATE ${tableName} SET profile=list_append(profile, ?) WHERE
 varietal=?`,
    Parameters: [["fruity"], "arabica"],
  });
  await client.send(updateItemStatementCommand);
  log("Updated coffee");

  /**
   * Delete the item.
   */

  log("Deleting the coffee.");
  const deleteItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.delete.html
    Statement: `DELETE FROM ${tableName} WHERE varietal=?`,
```

```
    Parameters: ["arabica"],
  });
  await docClient.send(deleteItemStatementCommand);
  log("Coffee deleted.");

  /**
   * Delete the table.
   */

  log("Deleting the table.");
  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para JavaScript da API.

Consultar uma tabela usando um índice secundário global

O exemplo de código a seguir mostra como consultar uma tabela usando um índice secundário global.

- Consulte uma tabela do DynamoDB usando a respectiva chave primária.
- Consulte um índice secundário global (GSI) para obter padrões de acesso alternativos.
- Compare consultas de tabela e consultas de GSI.

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB usando a chave primária com. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table using the primary key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} userId - The user ID to query by (partition key)
 * @returns {Promise<Object>} - The query response
```

```
 */
async function queryTable(
  config,
  tableName,
  userId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the base table
    const input = {
      TableName: tableName,
      KeyConditionExpression: "user_id = :userId",
      ExpressionAttributeValues: {
        ":userId": { S: userId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying table: ${error}`);
    throw error;
  }
}
```

Consulte um Índice Secundário Global (GSI) do DynamoDB com. AWS SDK para JavaScript

```
/**
 * Queries a DynamoDB Global Secondary Index (GSI)
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} indexName - The name of the GSI to query
 * @param {string} gameId - The game ID to query by (GSI partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryGSI(
  config,
  tableName,
```

```
  indexName,
  gameId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the GSI
    const input = {
      TableName: tableName,
      IndexName: indexName,
      KeyConditionExpression: "game_id = :gameId",
      ExpressionAttributeValues: {
        ":gameId": { S: gameId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying GSI: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consultar uma tabela usando uma condição begins_with

O exemplo de código a seguir mostra como consultar uma tabela usando uma condição begins_with.

- Use a função begins_with em uma expressão de condição de chave.
- Filtre itens com base em um padrão de prefixo na chave de classificação.

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB usando uma condição begins_with na chave de classificação por meio do AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");
```

```
/**
 * Queries a DynamoDB table for items where the sort key begins with a specific
 prefix
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key
 * @param {string} prefix - The prefix to match at the beginning of the sort key
 * @returns {Promise<Object>} - The query response
 */
async function queryWithBeginsWith(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  prefix
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND begins_with(#sk, :prefix)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":prefix": { S: prefix }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with begins_with: ${error}`);
```

```
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consulte uma tabela usando um intervalo de datas

O exemplo de código a seguir mostra como consultar uma tabela usando um intervalo de datas na chave de classificação.

- Consulte itens em um intervalo de datas específico.
- Use operadores de comparação em chaves de classificação formatadas por data.

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB em busca de itens dentro de um intervalo de datas com. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range on the sort key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key (must be a date/time
 attribute)
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRangeOnSortKey(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
```

```
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: '#pk = :pkValue AND #sk BETWEEN :startDate
 AND :endDate',
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range on sort key: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consultar uma tabela com uma expressão de filtro complexa

O exemplo de código a seguir mostra como consultar uma tabela com uma expressão de filtro complexa.

- Aplique expressões de filtro complexas aos resultados da consulta.
- Combine várias condições usando operadores lógicos.
- Filtre itens com base em atributos não chave.

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com uma expressão de filtro complexa usando. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with a complex filter expression
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number|string} minViews - Minimum number of views for filtering
 * @param {number|string} minReplies - Minimum number of replies for filtering
 * @param {string} requiredTag - Tag that must be present in the item's tags set
 * @returns {Promise<Object>} - The query response
 */
async function queryWithComplexFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  minViews,
  minReplies,
  requiredTag
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      FilterExpression: "views >= :minViews AND replies >= :minReplies AND
 contains(tags, :tag)",
```

```
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":minViews": { N: minViews.toString() },
        ":minReplies": { N: minReplies.toString() },
        ":tag": { S: requiredTag }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with complex filter: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consultar uma tabela com uma expressão de filtro dinâmica

O exemplo de código a seguir mostra como consultar uma tabela com uma expressão de filtro dinâmica.

- Crie expressões de filtro dinamicamente no tempo de execução.
- Crie condições de filtro com base na entrada do usuário ou no estado da aplicação.
- Adicione ou remova critérios de filtro condicionalmente.

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com uma expressão de filtro construída dinamicamente usando. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

async function queryWithDynamicFilter(
  config,
```

```
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  sortKeyValue,
  filterParams = {}
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize filter expression components
    let filterExpressions = [];
    const expressionAttributeValues = {
      ":pkValue": { S: partitionKeyValue },
      ":skValue": { S: sortKeyValue }
    };
    const expressionAttributeNames = {
      "#pk": partitionKeyName,
      "#sk": sortKeyName
    };

    // Add status filter if provided
    if (filterParams.status) {
      filterExpressions.push("status = :status");
      expressionAttributeValues[":status"] = { S: filterParams.status };
    }

    // Add minimum views filter if provided
    if (filterParams.minViews !== undefined) {
      filterExpressions.push("views >= :minViews");
      expressionAttributeValues[":minViews"] = { N:
 filterParams.minViews.toString() };
    }

    // Add author filter if provided
    if (filterParams.author) {
      filterExpressions.push("author = :author");
      expressionAttributeValues[":author"] = { S: filterParams.author };
    }

    // Construct the query input
    const input = {
      TableName: tableName,
```

```
      KeyConditionExpression: "#pk = :pkValue AND #sk = :skValue"
    };

    // Add filter expression if any filters were provided
    if (filterExpressions.length > 0) {
      input.FilterExpression = filterExpressions.join(" AND ");
    }

    // Add expression attribute names and values
    input.ExpressionAttributeNames = expressionAttributeNames;
    input.ExpressionAttributeValues = expressionAttributeValues;

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with dynamic filter: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consultar uma tabela com atributos aninhados

O exemplo de código a seguir mostra como consultar uma tabela com atributos aninhados.

- Acesse e filtre por atributos aninhados nos itens do DynamoDB.
- Use expressões de caminho de documento para se referir a elementos aninhados.

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com atributos aninhados usando. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table filtering on a nested attribute
 *
 * @param {Object} config - AWS SDK configuration object
```

```
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} productId - The product ID to query by (partition key)
 * @param {string} category - The category to filter by (nested attribute)
 * @returns {Promise<Object>} - The query response
 */
async function queryWithNestedAttribute(
  config,
  tableName,
  productId,
  category
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "product_id = :productId",
      FilterExpression: "details.category = :category",
      ExpressionAttributeValues: {
        ":productId": { S: productId },
        ":category": { S: category }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with nested attribute: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consultar uma tabela com paginação

O exemplo de código a seguir mostra como consultar uma tabela com paginação.

- Implemente a paginação para os resultados da consulta do DynamoDB.

- Use o LastEvaluatedKey para recuperar as páginas subsequentes.
- Controle o número de itens por página com o parâmetro Limit.

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com paginação usando. AWS SDK para JavaScript

```
/**
 * Example demonstrating how to handle large query result sets in DynamoDB using
 pagination
 *
 * This example shows:
 * - How to use pagination to handle large result sets
 * - How to use LastEvaluatedKey to retrieve the next page of results
 * - How to construct subsequent query requests using ExclusiveStartKey
 */
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with pagination to handle large result sets
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number} pageSize - Number of items per page
 * @returns {Promise<Array>} - All items from the query
 */
async function queryWithPagination(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  pageSize = 25
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize variables for pagination
    let lastEvaluatedKey = undefined;
```

```
    const allItems = [];
    let pageCount = 0;

    // Loop until all pages are retrieved
    do {
      // Construct the query input
      const input = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pkValue",
        Limit: pageSize,
        ExpressionAttributeNames: {
          "#pk": partitionKeyName
        },
        ExpressionAttributeValues: {
          ":pkValue": { S: partitionKeyValue }
        }
      };

      // Add ExclusiveStartKey if we have a LastEvaluatedKey from a previous query
      if (lastEvaluatedKey) {
        input.ExclusiveStartKey = lastEvaluatedKey;
      }

      // Execute the query
      const command = new QueryCommand(input);
      const response = await client.send(command);

      // Process the current page of results
      pageCount++;
      console.log(`Processing page ${pageCount} with ${response.Items.length}
items`);

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems.push(...response.Items);
      }

      // Get the LastEvaluatedKey for the next page
      lastEvaluatedKey = response.LastEvaluatedKey;

    } while (lastEvaluatedKey); // Continue until there are no more pages

    console.log(`Query complete. Retrieved ${allItems.length} items in ${pageCount}
pages.`);
```

```
    return allItems;
  } catch (error) {
    console.error(`Error querying with pagination: ${error}`);
    throw error;
  }
}

/**
 * Example usage:
 *
 * // Query all items in the "AWS DynamoDB" forum with pagination
 * const allItems = await queryWithPagination(
 *   { region: "us-west-2" },
 *   "ForumThreads",
 *   "ForumName",
 *   "AWS DynamoDB",
 *   25 // 25 items per page
 * );
 *
 * console.log(`Total items retrieved: ${allItems.length}`);
 *
 * // Notes on pagination:
 * // - LastEvaluatedKey contains the primary key of the last evaluated item
 * // - When LastEvaluatedKey is undefined/null, there are no more items to retrieve
 * // - ExclusiveStartKey tells DynamoDB where to start the next page
 * // - Pagination helps manage memory usage for large result sets
 * // - Each page requires a separate network request to DynamoDB
 */

module.exports = { queryWithPagination };
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consultar uma tabela com leituras altamente consistentes

O exemplo de código a seguir mostra como consultar uma tabela com leituras altamente consistentes.

- Configure o nível de consistência das consultas do DynamoDB.
- Use leituras altamente consistentes para obter os dados mais atualizados.
- Entenda as vantagens e desvantagens entre consistência final e consistência forte.

SDK para JavaScript (v3)

Consulte uma tabela do DynamoDB com consistência de leitura configurável usando. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with configurable read consistency
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {boolean} useConsistentRead - Whether to use strongly consistent reads
 * @returns {Promise<Object>} - The query response
 */
async function queryWithConsistentRead(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  useConsistentRead = false
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue }
      },
      ConsistentRead: useConsistentRead
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
```

```
  } catch (error) {
    console.error(`Error querying with consistent read: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consultar dados usando SELECT do PartiQL

O exemplo de código a seguir mostra como consultar dados usando declarações SELECT do PartiQL.

SDK para JavaScript (v3)

Consulte itens de uma tabela do DynamoDB usando instruções PartiQL SELECT com. AWS SDK para JavaScript

```
/**
 * This example demonstrates how to query items from a DynamoDB table using PartiQL.
 * It shows different ways to select data with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Select all items from a DynamoDB table using PartiQL.
 * Note: This should be used with caution on large tables.
 *
 * @param tableName - The name of the DynamoDB table
 * @returns The response from the ExecuteStatementCommand
 */
export const selectAllItems = async (tableName: string) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
```

```
    Statement: `SELECT * FROM "${tableName}"`,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select an item by its primary key using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemByPartitionKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${partitionKeyName} = ?`,
    Parameters: [partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving item:", err);
    throw err;
  }
};
```

```
/**
 * Select an item by its composite key (partition key + sort key) using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemByCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${sortKeyName} = ?`,
    Parameters: [partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving item:", err);
    throw err;
  }
};

/**
 * Select items using a filter condition with PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param filterAttribute - The attribute to filter on
 * @param filterValue - The value to filter by
 * @returns The response from the ExecuteStatementCommand
 */
```

```
export const selectItemsWithFilter = async (
  tableName: string,
  filterAttribute: string,
  filterValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${filterAttribute} = ?`,
    Parameters: [filterValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select items using a begins_with function for prefix matching.
 * This is useful for querying hierarchical data.
 *
 * @param tableName - The name of the DynamoDB table
 * @param attributeName - The attribute to check for prefix
 * @param prefix - The prefix to match
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsByPrefix = async (
  tableName: string,
  attributeName: string,
  prefix: string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE
 begins_with(${attributeName}, ?)`,
    Parameters: [prefix],
```

```
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select items using a between condition for range queries.
 *
 * @param tableName - The name of the DynamoDB table
 * @param attributeName - The attribute to check for range
 * @param startValue - The start value of the range
 * @param endValue - The end value of the range
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsByRange = async (
  tableName: string,
  attributeName: string,
  startValue: number | string,
  endValue: number | string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${attributeName} BETWEEN ? AND ?
`,
    Parameters: [startValue, endValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
```

```
};

/**
 * Example usage showing how to select items with different index types
 */
export const selectExamples = async () => {
  // Select all items from a table (use with caution on large tables)
  await selectAllItems("UsersTable");

  // Select by partition key (simple primary key)
  await selectItemByPartitionKey("UsersTable", "userId", "user123");

  // Select by composite key (partition key + sort key)
  await selectItemByCompositeKey("OrdersTable", "orderId", "order456", "productId",
 "prod789");

  // Select with a filter condition (can use any attribute)
  await selectItemsWithFilter("UsersTable", "userType", "premium");

  // Select items with a prefix (useful for hierarchical data)
  await selectItemsByPrefix("ProductsTable", "category", "electronics");

  // Select items within a range (useful for numeric or date ranges)
  await selectItemsByRange("OrdersTable", "orderDate", "2023-01-01", "2023-12-31");
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchExecuteStatement
  - ExecuteStatement

Consultar itens com TTL

O exemplo de código a seguir mostra como consultar itens com TTL.

SDK para JavaScript (v3)

Consulte a expressão filtrada para reunir itens TTL em uma tabela do DynamoDB usando. AWS SDK para JavaScript

```
import { DynamoDBClient, QueryCommand } from "@aws-sdk/client-dynamodb";
```

```
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const queryFiltered = async (tableName, primaryKey, region = 'us-east-1') =>
 {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);

    const params = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pk",
        FilterExpression: "#ea > :ea",
        ExpressionAttributeNames: {
            "#pk": "primaryKey",
            "#ea": "expireAt"
        },
        ExpressionAttributeValues: marshall({
            ":pk": primaryKey,
            ":ea": currentTime
        })
    };

    try {
        const { Items } = await client.send(new QueryCommand(params));
        Items.forEach(item => {
            console.log(unmarshall(item))
        });
        return Items;
    } catch (err) {
        console.error(`Error querying items: ${err}`);
        throw err;
    }
}

// Example usage (commented out for testing)
// queryFiltered('your-table-name', 'your-partition-key-value');
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Consultar tabelas usando padrões de data e hora

O exemplo de código a seguir mostra como consultar tabelas usando padrões de data e hora.

- Armazene e consulte date/time valores no DynamoDB.
- Implemente consultas de intervalo de datas usando chaves de classificação.
- Formate strings de data para obter uma consulta eficaz.

SDK para JavaScript (v3)

Consulte usando intervalos de datas em chaves de classificação com AWS SDK para JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range on the sort key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key (must be a date/time
 attribute)
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRangeOnSortKey(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
```

```
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: '#pk = :pkValue AND #sk BETWEEN :startDate
 AND :endDate',
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range on sort key: ${error}`);
    throw error;
  }
}
```

Consulte usando variáveis de data e hora com. AWS SDK para JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} dateKeyName - The name of the date attribute to filter on
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
```

```
 */
async function queryByDateRange(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  dateKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: `#pk = :pkValue AND #dateAttr BETWEEN :startDate
 AND :endDate`,
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#dateAttr": dateKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range: ${error}`);
    throw error;
  }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para JavaScript .

Como funciona a ordem da expressão de atualização

O exemplo de código a seguir mostra como funciona a ordem da expressão de atualização.

- Saiba como o DynamoDB processa expressões de atualização.
- Entenda como funciona a ordem das operações nas expressões de atualização.
- Evite resultados inesperados entendendo a avaliação da expressão.

SDK para JavaScript (v3)

Demonstre a ordem das expressões de atualização usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update an item with multiple actions in a single update expression.
 *
 * This function demonstrates how to use multiple actions in a single update
 expression
 * and how DynamoDB processes these actions.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {string} updateExpression - The update expression with multiple actions
 * @param {Object} [expressionAttributeNames] - Expression attribute name
 placeholders
 * @param {Object} [expressionAttributeValues] - Expression attribute value
 placeholders
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleActions(
  config,
```

```
  tableName,
  key,
  updateExpression,
  expressionAttributeNames,
  expressionAttributeValues
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Prepare the update parameters
  const updateParams = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ReturnValues: "UPDATED_NEW"
  };

  // Add expression attribute names if provided
  if (expressionAttributeNames) {
    updateParams.ExpressionAttributeNames = expressionAttributeNames;
  }

  // Add expression attribute values if provided
  if (expressionAttributeValues) {
    updateParams.ExpressionAttributeValues = expressionAttributeValues;
  }

  // Execute the update
  const response = await docClient.send(new UpdateCommand(updateParams));

  return response;
}

/**
 * Demonstrate that variables hold copies of existing values before modifications.
 *
 * This function creates an item with initial values, then updates it with an
 expression
 * that uses the values of attributes before they are modified in the same
 expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateValueCopying(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = { ...key, a: 1, b: 2, c: 3 };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with an expression that uses values before they are
 modified
  // This expression removes 'a', then sets 'b' to the value of 'a', and 'c' to the
 value of 'b'
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: "REMOVE a SET b = a, c = b",
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
```

```
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}

/**
 * Demonstrate the order in which different action types are processed.
 *
 * This function creates an item with initial values, then updates it with an
 expression
 * that includes multiple action types (SET, REMOVE, ADD, DELETE) to show the order
 * in which they are processed.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateActionOrder(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = {
    ...key,
    counter: 10,
    set_attr: new Set(["A", "B", "C"]),
    to_remove: "This will be removed",
    to_modify: "Original value"
  };
```

```
  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with multiple action types
  // The actions will be processed in this order: REMOVE, SET, ADD, DELETE
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: "REMOVE to_remove SET to_modify = :new_value ADD
 counter :increment DELETE set_attr :elements",
    ExpressionAttributeValues: {
      ":new_value": "Updated value",
      ":increment": 5,
      ":elements": new Set(["B"])
    },
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}
```

```
/**
 * Update multiple attributes with a single SET action.
 *
 * This function demonstrates how to update multiple attributes in a single SET
 action,
 * which is more efficient than using multiple separate update operations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {Object} attributes - The attributes to update and their new values
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleSetActions(
  config,
  tableName,
  key,
  attributes
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = "SET ";
  const expressionAttributeValues = {};

  // Add each attribute to the update expression
  Object.entries(attributes).forEach(([attrName, attrValue], index) => {
    const valuePlaceholder = `:val${index}`;

    if (index > 0) {
      updateExpression += ", ";
    }
    updateExpression += `${attrName} = ${valuePlaceholder}`;

    expressionAttributeValues[valuePlaceholder] = attrValue;
  });

  // Execute the update
  const response = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
```

```
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  }));

  return response;
}

/**
 * Update an attribute with a value from another attribute or a default value.
 *
 * This function demonstrates how to use if_not_exists to conditionally copy a value
 * from one attribute to another, or use a default value if the source doesn't
 exist.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {string} sourceAttribute - The attribute to copy the value from
 * @param {string} targetAttribute - The attribute to update
 * @param {any} defaultValue - The default value to use if the source attribute
 doesn't exist
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithConditionalValueCopying(
  config,
  tableName,
  key,
  sourceAttribute,
  targetAttribute,
  defaultValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Use if_not_exists to conditionally copy the value
  const response = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${targetAttribute} =
 if_not_exists(${sourceAttribute}, :default)`,
    ExpressionAttributeValues: {
      ":default": defaultValue
    },
```

```
    ReturnValues: "UPDATED_NEW"
  }));

  return response;
}

/**
 * Demonstrate complex update expressions with multiple operations on the same
 attribute.
 *
 * This function shows how DynamoDB processes multiple operations on the same
 attribute
 * in a single update expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateMultipleOperationsOnSameAttribute(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = {
    ...key,
    counter: 10,
    list_attr: [1, 2, 3],
    map_attr: {
      nested1: "value1",
      nested2: "value2"
    }
  };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));
```

```
// Step 2: Get the item to verify initial state
const responseBefore = await docClient.send(new GetCommand({
  TableName: tableName,
  Key: key
}));

const itemBefore = responseBefore.Item || {};

// Step 3: Update the item with multiple operations on the same attributes
const updateResponse = await docClient.send(new UpdateCommand({
  TableName: tableName,
  Key: key,
  UpdateExpression: `
    SET counter = counter + :inc1,
        counter = counter + :inc2,
        map_attr.nested1 = :new_val1,
        map_attr.nested3 = :new_val3,
        list_attr[0] = list_attr[1],
        list_attr[1] = list_attr[2]
  `,
  ExpressionAttributeValues: {
    ":inc1": 5,
    ":inc2": 3,
    ":new_val1": "updated_value1",
    ":new_val3": "new_value3"
  },
  ReturnValues: "UPDATED_NEW"
}));

// Step 4: Get the item to verify final state
const responseAfter = await docClient.send(new GetCommand({
  TableName: tableName,
  Key: key
}));

const itemAfter = responseAfter.Item || {};

// Return the results
return {
  initialState: itemBefore,
  updateResponse: updateResponse,
  finalState: itemAfter
};
```

```
}
```

Exemplo de uso da ordem de atualização da expressão com AWS SDK para JavaScript.

```
/**
 * Example of how to use update expression order of operations in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "OrderProcessing";

  console.log("Demonstrating update expression order of operations in DynamoDB");

  try {
    // Example 1: Demonstrating value copying in update expressions
    console.log("\nExample 1: Demonstrating value copying in update expressions");
    const results1 = await demonstrateValueCopying(
      config,
      tableName,
      { OrderId: "order123" }
    );

    console.log("Initial state:", JSON.stringify(results1.initialState, null, 2));
    console.log("Update response:", JSON.stringify(results1.updateResponse, null,
 2));
    console.log("Final state:", JSON.stringify(results1.finalState, null, 2));

    console.log("\nExplanation:");
    console.log("1. The initial state had a=1, b=2, c=3");
    console.log("2. The update expression 'REMOVE a SET b = a, c = b' did the
 following:");
    console.log("   - Copied the value of 'a' (which was 1) to be used for 'b'");
    console.log("   - Copied the value of 'b' (which was 2) to be used for 'c'");
    console.log("   - Removed the attribute 'a'");
    console.log("3. The final state has b=1, c=2, and 'a' is removed");
    console.log("4. This demonstrates that DynamoDB uses the values of attributes as
 they were BEFORE any modifications");

    // Example 2: Demonstrating the order of different action types
    console.log("\nExample 2: Demonstrating the order of different action types");
    const results2 = await demonstrateActionOrder(
```

```
      config,
      tableName,
      { OrderId: "order456" }
    );

    console.log("Initial state:", JSON.stringify(results2.initialState, null, 2));
    console.log("Update response:", JSON.stringify(results2.updateResponse, null,
2));
    console.log("Final state:", JSON.stringify(results2.finalState, null, 2));

    console.log("\nExplanation:");
    console.log("1. The update expression contained multiple action types: REMOVE,
SET, ADD, DELETE");
    console.log("2. DynamoDB processes these actions in this order: REMOVE, SET,
ADD, DELETE");
    console.log("3. First, 'to_remove' was removed");
    console.log("4. Then, 'to_modify' was set to a new value");
    console.log("5. Next, 'counter' was incremented by 5");
    console.log("6. Finally, 'B' was removed from the set attribute");

    // Example 3: Updating multiple attributes in a single SET action
    console.log("\nExample 3: Updating multiple attributes in a single SET action");
    const response3 = await updateWithMultipleSetActions(
      config,
      tableName,
      { OrderId: "order789" },
      {
        Status: "Shipped",
        ShippingDate: "2025-05-28",
        TrackingNumber: "1Z999AA10123456784"
      }
    );

    console.log("Multiple attributes updated successfully:",
JSON.stringify(response3.Attributes, null, 2));

    // Example 4: Conditional value copying with if_not_exists
    console.log("\nExample 4: Conditional value copying with if_not_exists");
    const response4 = await updateWithConditionalValueCopying(
      config,
      tableName,
      { OrderId: "order101" },
      "PreferredShippingMethod",
      "ShippingMethod",
```

```
      "Standard"
    );

    console.log("Conditional value copying result:",
JSON.stringify(response4.Attributes, null, 2));

    // Example 5: Multiple operations on the same attribute
    console.log("\nExample 5: Multiple operations on the same attribute");
    const results5 = await demonstrateMultipleOperationsOnSameAttribute(
      config,
      tableName,
      { OrderId: "order202" }
    );

    console.log("Initial state:", JSON.stringify(results5.initialState, null, 2));
    console.log("Update response:", JSON.stringify(results5.updateResponse, null,
2));
    console.log("Final state:", JSON.stringify(results5.finalState, null, 2));

    console.log("\nExplanation:");
    console.log("1. The counter was incremented twice (first by 5, then by 3) for a
total of +8");
    console.log("2. The map attribute had one value updated and a new nested
attribute added");
    console.log("3. The list attribute had values shifted (value at index 1 moved to
index 0, value at index 2 moved to index 1)");
    console.log("4. All operations within the SET action are processed from left to
right");

    // Key points about update expression order of operations
    console.log("\nKey Points About Update Expression Order of Operations:");
    console.log("1. Variables in expressions hold copies of attribute values as they
existed BEFORE any modifications");
    console.log("2. Multiple actions in an update expression are processed in this
order: REMOVE, SET, ADD, DELETE");
    console.log("3. Within each action type, operations are processed from left to
right");
    console.log("4. You can reference the same attribute multiple times in an
expression");
    console.log("5. You can use if_not_exists() to conditionally set values based on
attribute existence");
    console.log("6. Using a single update expression with multiple actions is more
efficient than multiple separate updates");
```

```
    console.log("7. The update expression is atomic - either all actions succeed or
 none do");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

Atualizar a configuração de throughput a quente de uma tabela

O exemplo de código a seguir mostra como atualizar a configuração de throughput a quente de uma tabela.

SDK para JavaScript (v3)

Atualize a configuração de throughput a quente em uma tabela existente do DynamoDB usando o AWS SDK para JavaScript.

```
import { DynamoDBClient, UpdateTableCommand } from "@aws-sdk/client-dynamodb";

export async function updateDynamoDBTableWarmThroughput(
  tableName,
  tableReadUnits,
  tableWriteUnits,
  gsiName,
  gsiReadUnits,
  gsiWriteUnits,
  region = "us-east-1"
) {
  try {
    const ddbClient = new DynamoDBClient({ region: region });

    // Construct the update table request
    const updateTableRequest = {
      TableName: tableName,
      GlobalSecondaryIndexUpdates: [
        {
            Update: {
```

```
                IndexName: gsiName,
                WarmThroughput: {
                    ReadUnitsPerSecond: gsiReadUnits,
                    WriteUnitsPerSecond: gsiWriteUnits,
                },
            },
        },
      ],
      WarmThroughput: {
          ReadUnitsPerSecond: tableReadUnits,
          WriteUnitsPerSecond: tableWriteUnits,
      },
    };

    const command = new UpdateTableCommand(updateTableRequest);
    const response = await ddbClient.send(command);
    console.log(`Table updated successfully! Response:
 ${JSON.stringify(response)}`);
    return response;
  } catch (error) {
    console.error(`Error updating table: ${error}`);
    throw error;
  }
}

// Example usage (commented out for testing)
/*
updateDynamoDBTableWarmThroughput(
  'example-table',
  5, 5,
  'example-index',
  2, 2
);
*/
```

- Para obter detalhes da API, consulte UpdateTable em Referência AWS SDK para JavaScript da API.

Atualiza a TTL de um item

O exemplo de código a seguir mostra como atualizar a TTL de um item.

SDK para JavaScript (v3)

```
import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const updateItem = async (tableName, partitionKey, sortKey, region = 'us-
east-1') => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);
    const expireAt = Math.floor((Date.now() + 90 * 24 * 60 * 60 * 1000) / 1000);

    const params = {
        TableName: tableName,
        Key: marshall({
            partitionKey: partitionKey,
            sortKey: sortKey
        }),
        UpdateExpression: "SET updatedAt = :c, expireAt = :e",
        ExpressionAttributeValues: marshall({
            ":c": currentTime,
            ":e": expireAt
        }),
    };

    try {
        const data = await client.send(new UpdateItemCommand(params));
        const responseData = unmarshall(data.Attributes);
        console.log("Item updated successfully: %s", responseData);
        return responseData;
    } catch (err) {
        console.error("Error updating item:", err);
        throw err;
    }
}

// Example usage (commented out for testing)
// updateItem('your-table-name', 'your-partition-key-value', 'your-sort-key-value');
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

Atualizar dados usando UPDATE do PartiQL

O exemplo de código a seguir mostra como atualizar dados usando declarações UPDATE do PartiQL.

SDK para JavaScript (v3)

Atualize itens em uma tabela do DynamoDB usando instruções PartiQL UPDATE com. AWS SDK para JavaScript

```
/**
 * This example demonstrates how to update items in a DynamoDB table using PartiQL.
 * It shows different ways to update documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Update a single attribute of an item using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const updateSingleAttribute = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeName: string,
  attributeValue: any
) => {
```

```
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ?`,
    Parameters: [attributeValue, partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update multiple attributes of an item using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeUpdates - Object containing attribute names and their new values
 * @returns The response from the ExecuteStatementCommand
 */
export const updateMultipleAttributes = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeUpdates: Record<string, any>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create SET clause for each attribute
  const setClause = Object.keys(attributeUpdates)
    .map((attr, index) => `${attr} = ?`)
    .join(", ");

  // Create parameters array with attribute values followed by the partition key
 value
```

```
  const parameters = [...Object.values(attributeUpdates), partitionKeyValue];

  const params = {
    Statement: `UPDATE "${tableName}" SET ${setClause} WHERE ${partitionKeyName} = ?
`,
    Parameters: parameters,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update an item identified by a composite key (partition key + sort key) using
 PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const updateItemWithCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number,
  attributeName: string,
  attributeValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
```

```
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ? AND ${sortKeyName} = ?`,
    Parameters: [attributeValue, partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update an item with a condition to ensure the update only happens if a condition
 is met.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @param conditionAttribute - The attribute to check in the condition
 * @param conditionValue - The value to compare against in the condition
 * @returns The response from the ExecuteStatementCommand
 */
export const updateItemWithCondition = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeName: string,
  attributeValue: any,
  conditionAttribute: string,
  conditionValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ? AND ${conditionAttribute} = ?`,
    Parameters: [attributeValue, partitionKeyValue, conditionValue],
```

```
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated with condition successfully");
    return data;
  } catch (err) {
    console.error("Error updating item with condition:", err);
    throw err;
  }
};

/**
 * Batch update multiple items using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param updates - Array of objects containing key and update information
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchUpdateItems = async (
  tableName: string,
  updates: Array<{
    partitionKeyName: string;
    partitionKeyValue: string | number;
    attributeName: string;
    attributeValue: any;
  }>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each update
  const statements = updates.map((update) => {
    return {
      Statement: `UPDATE "${tableName}" SET ${update.attributeName} = ? WHERE
 ${update.partitionKeyName} = ?`,
      Parameters: [update.attributeValue, update.partitionKeyValue],
    };
  });

  const params = {
    Statements: statements,
  };
```

```
  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items batch updated successfully");
    return data;
  } catch (err) {
    console.error("Error batch updating items:", err);
    throw err;
  }
};

/**
 * Example usage showing how to update items with different index types
 */
export const updateExamples = async () => {
  // Update a single attribute using a simple primary key
  await updateSingleAttribute("UsersTable", "userId", "user123", "email",
 "newemail@example.com");

  // Update multiple attributes at once
  await updateMultipleAttributes("UsersTable", "userId", "user123", {
    email: "newemail@example.com",
    name: "John Smith",
    lastLogin: new Date().toISOString(),
  });

  // Update an item with a composite key (partition key + sort key)
  await updateItemWithCompositeKey(
    "OrdersTable",
    "orderId",
    "order456",
    "productId",
    "prod789",
    "quantity",
    5
  );

  // Update with a condition
  await updateItemWithCondition(
    "UsersTable",
    "userId",
    "user123",
    "userStatus",
    "active",
    "userType",
```

```
    "premium"
  );

  // Batch update multiple items
  await batchUpdateItems("UsersTable", [
    {
      partitionKeyName: "userId",
      partitionKeyValue: "user123",
      attributeName: "lastLogin",
      attributeValue: new Date().toISOString(),
    },
    {
      partitionKeyName: "userId",
      partitionKeyValue: "user456",
      attributeName: "lastLogin",
      attributeValue: new Date().toISOString(),
    },
  ]);
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchExecuteStatement
  - ExecuteStatement

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar operações de contador atômico

O exemplo de código a seguir mostra como usar operações de contador atômico no DynamoDB.

- Incremente os contadores atomicamente usando as operações ADD e SET.
- Incremente seguramente contadores que talvez não existam.
- Implemente um bloqueio positivo para operações de contador.

SDK para JavaScript (v3)

Demonstre operações de contador atômico usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Increment a counter using the ADD operation.
 *
 * This function demonstrates using the ADD operation for atomic increments.
 * The ADD operation is atomic and is the recommended way to increment counters.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterWithAdd(
  config,
  tableName,
  key,
  counterName,
  incrementValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using ADD
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `ADD ${counterName} :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter using the SET operation with an expression.
 *
 * This function demonstrates using the SET operation with an expression for
 increments.
 * While this approach works, it's less idiomatic for simple increments than using
 ADD.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
```

```
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterWithSet(
  config,
  tableName,
  key,
  counterName,
  incrementValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with an expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = ${counterName} + :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter safely, handling the case where the counter might not exist.
 *
 * This function demonstrates using the if_not_exists function with SET to safely
 * increment a counter that might not exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @param {number} defaultValue - The default value if the counter doesn't exist
```

```
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterSafely(
  config,
  tableName,
  key,
  counterName,
  incrementValue,
  defaultValue = 0
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with if_not_exists
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = if_not_exists(${counterName}, :default)
 + :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue,
      ":default": defaultValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter with optimistic locking to prevent race conditions.
 *
 * This function demonstrates using a condition expression to implement optimistic
 * locking, which prevents race conditions when multiple processes try to update
 * the same counter.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
```

```
 * @param {number} incrementValue - The value to increment by
 * @param {number} expectedValue - The expected current value of the counter
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterWithLocking(
  config,
  tableName,
  key,
  counterName,
  incrementValue,
  expectedValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = ${counterName} + :increment`,
    ConditionExpression: `${counterName} = :expected`,
    ExpressionAttributeValues: {
      ":increment": incrementValue,
      ":expected": expectedValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return {
      success: true,
      data: response
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        error: "Optimistic locking failed: the counter value has changed"
      };
    }
```

```
    // Re-throw other errors
    throw error;
  }
}

/**
 * Get the current value of a counter.
 *
 * Helper function to retrieve the current value of a counter attribute.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @param {string} counterName - The name of the counter attribute
 * @returns {Promise<number|null>} - The current counter value or null if not found
 */
async function getCounterValue(
  config,
  tableName,
  key,
  counterName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the counter value if it exists, otherwise null
  return response.Item && counterName in response.Item
    ? response.Item[counterName]
    : null;
}
```

Exemplo de uso de operações de contador atômico com AWS SDK para JavaScript.

```
/**
 * Example of how to use the atomic counter operations.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };
  const counterName = "ViewCount";
  const incrementValue = 1;

  console.log("Demonstrating different approaches to increment counters in
 DynamoDB");

  try {
    // Example 1: Using ADD operation (recommended for simple increments)
    console.log("\nExample 1: Incrementing counter with ADD operation");
    const response1 = await incrementCounterWithAdd(
      config,
      tableName,
      key,
      counterName,
      incrementValue
    );

    console.log(`Counter incremented to: ${response1.Attributes[counterName]}`);

    // Example 2: Using SET operation with an expression
    console.log("\nExample 2: Incrementing counter with SET operation");
    const response2 = await incrementCounterWithSet(
      config,
      tableName,
      key,
      counterName,
      incrementValue
    );

    console.log(`Counter incremented to: ${response2.Attributes[counterName]}`);

    // Example 3: Safely incrementing a counter that might not exist
    console.log("\nExample 3: Safely incrementing counter that might not exist");
    const newKey = { ProductId: "P67890" };
    const response3 = await incrementCounterSafely(
```

```
      config,
      tableName,
      newKey,
      counterName,
      incrementValue,
      0
    );

    console.log(`Counter initialized and incremented to:
${response3.Attributes[counterName]}`);

    // Example 4: Incrementing with optimistic locking
    console.log("\nExample 4: Incrementing with optimistic locking");

    // First, get the current counter value
    const currentValue = await getCounterValue(config, tableName, key, counterName);
    console.log(`Current counter value: ${currentValue}`);

    // Then, try to increment with optimistic locking
    const response4 = await incrementCounterWithLocking(
      config,
      tableName,
      key,
      counterName,
      incrementValue,
      currentValue
    );

    if (response4.success) {
      console.log(`Counter successfully incremented to:
${response4.data.Attributes[counterName]}`);
    } else {
      console.log(response4.error);
    }

    // Explain the differences between ADD and SET
    console.log("\nKey differences between ADD and SET for counter operations:");
    console.log("1. ADD is more concise and idiomatic for simple increments");
    console.log("2. SET with expressions is more flexible for complex operations");
    console.log("3. Both operations are atomic and safe for concurrent updates");
    console.log("4. SET with if_not_exists is required when the attribute might not
exist");
    console.log("5. Optimistic locking can be added to either approach for
additional safety");
```

```
  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para JavaScript da API.

Usar operações condicionais

O exemplo de código a seguir mostra como usar operações condicionais no DynamoDB.

- Implemente gravações condicionais para evitar a substituição de dados.
- Use expressões condicionais para impor regras de negócios.
- Lide com falhas de verificação condicional com tranquilidade.

SDK para JavaScript (v3)

Demonstre operações condicionais usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  DeleteCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Perform a conditional update operation.
 *
 * This function demonstrates how to update an item only if a condition is met.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} conditionAttribute - The attribute to check in the condition
 * @param {any} conditionValue - The value to compare against
```

```
 * @param {string} updateAttribute - The attribute to update
 * @param {any} updateValue - The new value to set
 * @returns {Promise<Object>} - Result of the operation
 */
async function conditionalUpdate(
  config,
  tableName,
  key,
  conditionAttribute,
  conditionValue,
  updateAttribute,
  updateValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${updateAttribute} = :value`,
    ConditionExpression: `${conditionAttribute} = :condition`,
    ExpressionAttributeValues: {
      ":value": updateValue,
      ":condition": conditionValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Condition was met and update was performed",
      updatedAttributes: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
```

```
        message: "Condition was not met, update was not performed",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Perform a conditional delete operation.
 *
 * This function demonstrates how to delete an item only if a condition is met.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to delete
 * @param {string} conditionAttribute - The attribute to check in the condition
 * @param {any} conditionValue - The value to compare against
 * @returns {Promise<Object>} - Result of the operation
 */
async function conditionalDelete(
  config,
  tableName,
  key,
  conditionAttribute,
  conditionValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the delete parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    ConditionExpression: `${conditionAttribute} = :condition`,
    ExpressionAttributeValues: {
      ":condition": conditionValue
    },
    ReturnValues: "ALL_OLD"
  };
```

```
  try {
    // Perform the delete operation
    const response = await docClient.send(new DeleteCommand(params));

    return {
      success: true,
      message: "Condition was met and item was deleted",
      deletedItem: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Condition was not met, item was not deleted",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Implement optimistic locking with a version number.
 *
 * This function demonstrates how to use a version number for optimistic locking
 * to prevent race conditions when multiple processes update the same item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {Object} updates - The attributes to update
 * @param {number} expectedVersion - The expected current version number
 * @returns {Promise<Object>} - Result of the operation
 */
async function updateWithOptimisticLocking(
  config,
  tableName,
  key,
  updates,
  expectedVersion
) {
```

```
// Initialize the DynamoDB client
const client = new DynamoDBClient(config);
const docClient = DynamoDBDocumentClient.from(client);

// Build the update expression
const updateExpressions = [];
const expressionAttributeValues = {
  ":expectedVersion": expectedVersion,
  ":newVersion": expectedVersion + 1
};

// Add each update to the expression
Object.entries(updates).forEach(([attribute, value], index) => {
  updateExpressions.push(`${attribute} = :val${index}`);
  expressionAttributeValues[`:val${index}`] = value;
});

// Add the version update
updateExpressions.push("version = :newVersion");

// Define the update parameters with a condition expression
const params = {
  TableName: tableName,
  Key: key,
  UpdateExpression: `SET ${updateExpressions.join(", ")}`,
  ConditionExpression: "version = :expectedVersion",
  ExpressionAttributeValues: expressionAttributeValues,
  ReturnValues: "UPDATED_NEW"
};

try {
  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return {
    success: true,
    message: "Update succeeded with optimistic locking",
    newVersion: expectedVersion + 1,
    updatedAttributes: response.Attributes
  };
} catch (error) {
  // Check if the error is due to the condition check failing
  if (error.name === "ConditionalCheckFailedException") {
    return {
```

```
        success: false,
        message: "Optimistic locking failed: the item was modified by another
 process",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Implement a conditional write that creates an item only if it doesn't exist.
 *
 * This function demonstrates how to use attribute_not_exists to create an item
 * only if it doesn't already exist (similar to an "INSERT IF NOT EXISTS"
 operation).
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} item - The item to create
 * @returns {Promise<Object>} - Result of the operation
 */
async function createIfNotExists(
  config,
  tableName,
  item
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Extract the primary key attributes
  const keyAttributes = Object.keys(item).filter(attr =>
    attr === "id" || attr === "ID" || attr === "Id" ||
    attr.endsWith("Id") || attr.endsWith("ID") ||
    attr.endsWith("Key")
  );

  if (keyAttributes.length === 0) {
    throw new Error("Could not determine primary key attributes");
  }
```

```
  // Create a condition expression that checks if the item doesn't exist
  const conditionExpression = `attribute_not_exists(${keyAttributes[0]})`;

  // Define the put parameters with a condition expression
  const params = {
    TableName: tableName,
    Item: item,
    ConditionExpression: conditionExpression
  };

  try {
    // Perform the put operation
    await docClient.send(new PutCommand(params));

    return {
      success: true,
      message: "Item was created because it didn't exist",
      item
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Item already exists, creation was skipped",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
```

```
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Exemplo de uso de operações condicionais com AWS SDK para JavaScript.

```
/**
 * Example of how to use conditional operations.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating conditional operations in DynamoDB");

  try {
    // Example 1: Conditional update based on attribute value
    console.log("\nExample 1: Conditional update based on attribute value");
    const updateResult = await conditionalUpdate(
      config,
      tableName,
      key,
```

```
    "Category",
    "Electronics",
    "Price",
    299.99
  );

  console.log(`Result: ${updateResult.message}`);
  if (updateResult.success) {
    console.log("Updated attributes:", updateResult.updatedAttributes);
  }

  // Example 2: Conditional delete based on attribute value
  console.log("\nExample 2: Conditional delete based on attribute value");
  const deleteResult = await conditionalDelete(
    config,
    tableName,
    key,
    "InStock",
    false
  );

  console.log(`Result: ${deleteResult.message}`);
  if (deleteResult.success) {
    console.log("Deleted item:", deleteResult.deletedItem);
  }

  // Example 3: Optimistic locking with version number
  console.log("\nExample 3: Optimistic locking with version number");

  // First, get the current item to check its version
  const currentItem = await getItem(config, tableName, { ProductId: "P67890" });
  const currentVersion = currentItem ? (currentItem.version || 0) : 0;

  console.log(`Current version: ${currentVersion}`);

  // Then, update with optimistic locking
  const lockingResult = await updateWithOptimisticLocking(
    config,
    tableName,
    { ProductId: "P67890" },
    {
      Name: "Updated Product Name",
      Description: "This is an updated description"
    },
```

```
      currentVersion
    );

    console.log(`Result: ${lockingResult.message}`);
    if (lockingResult.success) {
      console.log(`New version: ${lockingResult.newVersion}`);
      console.log("Updated attributes:", lockingResult.updatedAttributes);
    }

    // Example 4: Create item only if it doesn't exist
    console.log("\nExample 4: Create item only if it doesn't exist");
    const createResult = await createIfNotExists(
      config,
      tableName,
      {
        ProductId: "P99999",
        Name: "New Product",
        Category: "Accessories",
        Price: 19.99,
        InStock: true
      }
    );

    console.log(`Result: ${createResult.message}`);
    if (createResult.success) {
      console.log("Created item:", createResult.item);
    }

    // Explain conditional operations
    console.log("\nKey points about conditional operations:");
    console.log("1. Conditional operations only succeed if the condition is met");
    console.log("2. ConditionalCheckFailedException indicates the condition wasn't
met");
    console.log("3. Optimistic locking prevents race conditions in concurrent
updates");
    console.log("4. attribute_exists and attribute_not_exists are useful for
checking if attributes are present");
    console.log("5. Conditional operations are atomic - they either succeed
completely or fail completely");
    console.log("6. You can use any valid comparison operators and functions in
condition expressions");
    console.log("7. Conditional operations don't consume write capacity if the
condition fails");
```

```
  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteItem
  - PutItem
  - UpdateItem

Usar nomes de atributo de expressão

O exemplo de código a seguir mostra como usar nomes de atributo de expressão no DynamoDB.

- Trabalhe com palavras reservadas nas expressões do DynamoDB.
- Use espaços reservados para nomes de atributo de expressão.
- Manipule caracteres especiais em nomes de atributo.

SDK para JavaScript (v3)

Demonstre nomes de atributos de expressão usando AWS SDK para JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  QueryCommand,
  ScanCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update an attribute that is a reserved word in DynamoDB.
 *
 * This function demonstrates how to use expression attribute names to update
 * attributes that are reserved words in DynamoDB.
 *
 * @param {Object} config - AWS configuration object
```

```
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} reservedWordAttribute - The reserved word attribute to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateReservedWordAttribute(
  config,
  tableName,
  key,
  reservedWordAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using expression attribute names
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: "SET #attr = :value",
    ExpressionAttributeNames: {
      "#attr": reservedWordAttribute
    },
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update an attribute that contains special characters.
 *
 * This function demonstrates how to use expression attribute names to update
 * attributes that contain special characters.
 *
 * @param {Object} config - AWS configuration object
```

```
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} specialCharAttribute - The attribute with special characters to
 update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateSpecialCharacterAttribute(
  config,
  tableName,
  key,
  specialCharAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using expression attribute names
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: "SET #attr = :value",
    ExpressionAttributeNames: {
      "#attr": specialCharAttribute
    },
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Query items using an attribute that is a reserved word.
 *
 * This function demonstrates how to use expression attribute names in a query
 * when the attribute is a reserved word.
 *
```

```
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key attribute
 * @param {any} partitionKeyValue - The value of the partition key
 * @param {string} reservedWordAttribute - The reserved word attribute to filter on
 * @param {any} value - The value to compare against
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function queryWithReservedWordAttribute(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  reservedWordAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the query parameters using expression attribute names
  const params = {
    TableName: tableName,
    KeyConditionExpression: "#pkName = :pkValue",
    FilterExpression: "#attr = :value",
    ExpressionAttributeNames: {
      "#pkName": partitionKeyName,
      "#attr": reservedWordAttribute
    },
    ExpressionAttributeValues: {
      ":pkValue": partitionKeyValue,
      ":value": value
    }
  };

  // Perform the query operation
  const response = await docClient.send(new QueryCommand(params));

  return response;
}

/**
 * Update a nested attribute with a path that contains reserved words.
 *
```

```
 * This function demonstrates how to use expression attribute names to update
 * nested attributes where the path contains reserved words.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string[]} attributePath - The path to the nested attribute as an array
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateNestedReservedWordAttribute(
  config,
  tableName,
  key,
  attributePath,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create expression attribute names for each part of the path
  const expressionAttributeNames = {};
  for (let i = 0; i < attributePath.length; i++) {
    expressionAttributeNames[`#attr${i}`] = attributePath[i];
  }

  // Build the attribute path using the expression attribute names
  const attributePathExpression = attributePath
    .map((_, i) => `#attr${i}`)
    .join(".");

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${attributePathExpression} = :value`,
    ExpressionAttributeNames: expressionAttributeNames,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };
```

```
  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Scan a table with multiple attribute name placeholders.
 *
 * This function demonstrates how to use multiple expression attribute names
 * in a complex filter expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} filters - Object mapping attribute names to filter values
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function scanWithMultipleAttributeNames(
  config,
  tableName,
  filters
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create expression attribute names and values
  const expressionAttributeNames = {};
  const expressionAttributeValues = {};
  const filterConditions = [];

  // Build the filter expression
  Object.entries(filters).forEach(([attrName, value], index) => {
    const nameKey = `#attr${index}`;
    const valueKey = `:val${index}`;

    expressionAttributeNames[nameKey] = attrName;
    expressionAttributeValues[valueKey] = value;
    filterConditions.push(`${nameKey} = ${valueKey}`);
  });

  // Join the filter conditions with AND
  const filterExpression = filterConditions.join(" AND ");
```

```
  // Define the scan parameters
  const params = {
    TableName: tableName,
    FilterExpression: filterExpression,
    ExpressionAttributeNames: expressionAttributeNames,
    ExpressionAttributeValues: expressionAttributeValues
  };

  // Perform the scan operation
  const response = await docClient.send(new ScanCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
```

```
}
```

Exemplo de uso de nomes de atributos de expressão com AWS SDK para JavaScript.

```
/**
 * Example of how to use expression attribute names.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating expression attribute names in DynamoDB");

  try {
    // Example 1: Update an attribute that is a reserved word
    console.log("\nExample 1: Updating an attribute that is a reserved word");
    const response1 = await updateReservedWordAttribute(
      config,
      tableName,
      key,
      "Size", // "SIZE" is a reserved word in DynamoDB
      "Large"
    );

    console.log("Updated attribute:", response1.Attributes);

    // Example 2: Update an attribute with special characters
    console.log("\nExample 2: Updating an attribute with special characters");
    const response2 = await updateSpecialCharacterAttribute(
      config,
      tableName,
      key,
      "Product-Type", // Contains a hyphen, which is a special character
      "Electronics"
    );

    console.log("Updated attribute:", response2.Attributes);

    // Example 3: Query with a reserved word attribute
    console.log("\nExample 3: Querying with a reserved word attribute");
```

```
    const response3 = await queryWithReservedWordAttribute(
      config,
      tableName,
      "Category",
      "Electronics",
      "Count", // "COUNT" is a reserved word in DynamoDB
      10
    );

    console.log(`Found ${response3.Items.length} items`);

    // Example 4: Update a nested attribute with reserved words in the path
    console.log("\nExample 4: Updating a nested attribute with reserved words in the
path");
    const response4 = await updateNestedReservedWordAttribute(
      config,
      tableName,
      key,
      ["Dimensions", "Size", "Height"], // "SIZE" is a reserved word
      30
    );

    console.log("Updated nested attribute:", response4.Attributes);

    // Example 5: Scan with multiple attribute name placeholders
    console.log("\nExample 5: Scanning with multiple attribute name placeholders");
    const response5 = await scanWithMultipleAttributeNames(
      config,
      tableName,
      {
        "Size": "Large",
        "Count": 10,
        "Product-Type": "Electronics"
      }
    );

    console.log(`Found ${response5.Items.length} items`);

    // Get the final state of the item
    console.log("\nFinal state of the item:");
    const item = await getItem(config, tableName, key);
    console.log(JSON.stringify(item, null, 2));

    // Show some common reserved words
```

```
    console.log("\nSome common DynamoDB reserved words:");
    const commonReservedWords = [
      "ABORT", "ABSOLUTE", "ACTION", "ADD", "ALL", "ALTER", "AND", "ANY", "AS",
      "ASC", "BETWEEN", "BY", "CASE", "CAST", "COLUMN", "CONNECT", "COUNT",
      "CREATE", "CURRENT", "DATE", "DELETE", "DESC", "DROP", "ELSE", "EXISTS",
      "FOR", "FROM", "GRANT", "GROUP", "HAVING", "IN", "INDEX", "INSERT", "INTO",
      "IS", "JOIN", "KEY", "LEVEL", "LIKE", "LIMIT", "LOCAL", "MAX", "MIN", "NAME",
      "NOT", "NULL", "OF", "ON", "OR", "ORDER", "OUTER", "REPLACE", "RETURN",
      "SELECT", "SET", "SIZE", "TABLE", "THEN", "TO", "UPDATE", "USER", "VALUES",
      "VIEW", "WHERE"
    ];
    console.log(commonReservedWords.join(", "));

    // Explain expression attribute names
    console.log("\nKey points about expression attribute names:");
    console.log("1. Use expression attribute names (#name) for reserved words");
    console.log("2. Use expression attribute names for attributes with special
 characters");
    console.log("3. Special characters include: spaces, hyphens, dots, and other
 non-alphanumeric characters");
    console.log("4. Expression attribute names are required for nested attributes
 with reserved words");
    console.log("5. You can use multiple expression attribute names in a single
 expression");
    console.log("6. Expression attribute names are case-sensitive");
    console.log("7. Expression attribute names are only used in expressions, not in
 the actual data");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - Consulta
  - UpdateItem

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos sem servidor

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do DynamoDB com o Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
};

const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Consumindo um evento do DynamoDB com o Lambda usando. TypeScript

```
export const handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
}
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatando falhas de itens de lote do DynamoDB com o uso do Lambda. JavaScript

```
export const handler = async (event) => {
  const records = event.Records;
  let curRecordSequenceNumber = "";

  for (const record of records) {
    try {
      // Process your record
      curRecordSequenceNumber = record.dynamodb.SequenceNumber;
    } catch (e) {
      // Return failed record's sequence number
      return { batchItemFailures: [{ itemIdentifier: curRecordSequenceNumber }] };
    }
  }

  return { batchItemFailures: [] };
};
```

Relatando falhas de itens de lote do DynamoDB com o uso do Lambda. TypeScript

```
import {
  DynamoDBBatchResponse,
  DynamoDBBatchItemFailure,
  DynamoDBStreamEvent,
} from "aws-lambda";
```

```
export const handler = async (
  event: DynamoDBStreamEvent
): Promise<DynamoDBBatchResponse> => {
  const batchItemFailures: DynamoDBBatchItemFailure[] = [];
  let curRecordSequenceNumber;

  for (const record of event.Records) {
    curRecordSequenceNumber = record.dynamodb?.SequenceNumber;

    if (curRecordSequenceNumber) {
      batchItemFailures.push({
        itemIdentifier: curRecordSequenceNumber,
      });
    }
  }

  return { batchItemFailures: batchItemFailures };
};
```

Exemplos do Amazon EC2 usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon EC2.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos

- Ações
- Cenários

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

// Call DescribeSecurityGroups and display the result.
export const main = async () => {
  const client = new EC2Client();
  try {
    const { SecurityGroups } = await client.send(
      new DescribeSecurityGroupsCommand({}),
    );

    const securityGroupList = SecurityGroups.slice(0, 9)
      .map((sg) => ` • ${sg.GroupId}: ${sg.GroupName}`)
      .join("\n");

    console.log(
      "Hello, Amazon EC2! Let's list up to 10 of your security groups:",
    );
    console.log(securityGroupList);
  } catch (err) {
    console.error(err);
  }
};
```

```
// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo contém uma lista de ações comuns usadas com o EC2. As etapas são construídas com um framework de cenário que simplifica a execução de um exemplo interativo. Para ver o contexto completo, visite o GitHub repositório.

```
import { tmpdir } from "node:os";
import { writeFile, mkdtemp, rm } from "node:fs/promises";
import { join } from "node:path";
import { get } from "node:http";
```

```
import {
  AllocateAddressCommand,
  AssociateAddressCommand,
  AuthorizeSecurityGroupIngressCommand,
  CreateKeyPairCommand,
  CreateSecurityGroupCommand,
  DeleteKeyPairCommand,
  DeleteSecurityGroupCommand,
  DisassociateAddressCommand,
  paginateDescribeImages,
  paginateDescribeInstances,
  paginateDescribeInstanceTypes,
  ReleaseAddressCommand,
  RunInstancesCommand,
  StartInstancesCommand,
  StopInstancesCommand,
  TerminateInstancesCommand,
  waitUntilInstanceStatusOk,
  waitUntilInstanceStopped,
  waitUntilInstanceTerminated,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

import { paginateGetParametersByPath, SSMClient } from "@aws-sdk/client-ssm";

/**
 * @typedef {{
 *   ec2Client: import('@aws-sdk/client-ec2').EC2Client,
 *   errors: Error[],
 *   keyPairId?: string,
 *   tmpDirectory?: string,
 *   securityGroupId?: string,
 *   ipAddress?: string,
 *   images?: import('@aws-sdk/client-ec2').Image[],
 *   image?: import('@aws-sdk/client-ec2').Image,
 *   instanceTypes?: import('@aws-sdk/client-ec2').InstanceTypeInfo[],
 *   instanceId?: string,
 *   instanceIpAddress?: string,
```

```
 *   allocationId?: string,
 *   allocatedIpAddress?: string,
 *   associationId?: string,
 * }} State
 */

/**
 * A skip function provided to the `skipWhen` of a Step when you want
 * to ignore that step if any errors have occurred.
 * @param {State} state
 */
const skipWhenErrors = (state) => state.errors.length > 0;

const MAX_WAITER_TIME_IN_SECONDS = 60 * 8;

export const confirm = new ScenarioInput("confirmContinue", "Continue?", {
  type: "confirm",
  skipWhen: skipWhenErrors,
});

export const exitOnNoConfirm = new ScenarioAction(
  "exitOnConfirmContinueFalse",
  (/** @type { { earlyExit: boolean } & Record<string, any>} */ state) => {
    if (!state[confirm.name]) {
      state.earlyExit = true;
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

export const greeting = new ScenarioOutput(
  "greeting",
  `

Welcome to the Amazon EC2 basic usage scenario.

Before you launch an instances, you'll need to provide a few things:
 - A key pair - This is for SSH access to your EC2 instance. You only need to
 provide the name.
 - A security group - This is used for configuring access to your instance. Again,
 only the name is needed.
 - An IP address - Your public IP address will be fetched.
```

```
 - An Amazon Machine Image (AMI)
 - A compatible instance type`,
  { header: true, preformatted: true, skipWhen: skipWhenErrors },
);

export const provideKeyPairName = new ScenarioInput(
  "keyPairName",
  "Provide a name for a new key pair.",
  { type: "input", default: "ec2-example-key-pair", skipWhen: skipWhenErrors },
);

export const createKeyPair = new ScenarioAction(
  "createKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Create a key pair in Amazon EC2.
      const { KeyMaterial, KeyPairId } = await state.ec2Client.send(
        // A unique name for the key pair. Up to 255 ASCII characters.
        new CreateKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );

      state.keyPairId = KeyPairId;

      // Save the private key in a temporary location.
      state.tmpDirectory = await mkdtemp(join(tmpdir(), "ec2-scenario-tmp"));
      await writeFile(
        `${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem`,
        KeyMaterial,
        {
          mode: 0o400,
        },
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidKeyPair.Duplicate"
      ) {
        caught.message = `${caught.message}. Try another key name.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
```

```
);

export const logKeyPair = new ScenarioOutput(
  "logKeyPair",
  (/** @type {State} */ state) =>
    `Created the key pair ${state[provideKeyPairName.name]}.`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteKeyPair = new ScenarioInput(
  "confirmDeleteKeyPair",
  "Do you want to delete the key pair?",
  {
    type: "confirm",
    // Don't do anything when a key pair was never created.
    skipWhen: (/** @type {State} */ state) => !state.keyPairId,
  },
);

export const maybeDeleteKeyPair = new ScenarioAction(
  "deleteKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Delete a key pair by name from EC2
      await state.ec2Client.send(
        new DeleteKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        // Occurs when a required parameter (e.g. KeyName) is undefined.
        caught.name === "MissingParameter"
      ) {
        caught.message = `${caught.message}. Did you provide the required value?`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no key pair to delete or the user chooses
    // to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.keyPairId || !state[confirmDeleteKeyPair.name],
  },
```

```
);

export const provideSecurityGroupName = new ScenarioInput(
  "securityGroupName",
  "Provide a name for a new security group.",
  { type: "input", default: "ec2-scenario-sg", skipWhen: skipWhenErrors },
);

export const createSecurityGroup = new ScenarioAction(
  "createSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Create a new security group that will be used to configure ingress/egress
 for
      // an EC2 instance.
      const { GroupId } = await state.ec2Client.send(
        new CreateSecurityGroupCommand({
          GroupName: state[provideSecurityGroupName.name],
          Description: "A security group for the Amazon EC2 example.",
        }),
      );
      state.securityGroupId = GroupId;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidGroup.Duplicate") {
        caught.message = `${caught.message}. Please provide a different name for
 your security group.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroup = new ScenarioOutput(
  "logSecurityGroup",
  (/** @type {State} */ state) =>
    `Created the security group ${state.securityGroupId}.`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteSecurityGroup = new ScenarioInput(
  "confirmDeleteSecurityGroup",
  "Do you want to delete the security group?",
```

```
  {
    type: "confirm",
    // Don't do anything when a security group was never created.
    skipWhen: (/** @type {State} */ state) => !state.securityGroupId,
  },
);

export const maybeDeleteSecurityGroup = new ScenarioAction(
  "deleteSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Delete the security group if the 'skipWhen' condition below is not met.
      await state.ec2Client.send(
        new DeleteSecurityGroupCommand({
          GroupId: state.securityGroupId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no security group to delete
    // or the user chooses to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.securityGroupId || !state[confirmDeleteSecurityGroup.name],
  },
);

export const authorizeSecurityGroupIngress = new ScenarioAction(
  "authorizeSecurity",
  async (/** @type {State} */ state) => {
    try {
      // Get the public IP address of the machine running this example.
      const ipAddress = await new Promise((res, rej) => {
        get("http://checkip.amazonaws.com", (response) => {
          let data = "";
          response.on("data", (chunk) => {
```

```
            data += chunk;
          });
          response.on("end", () => res(data.trim()));
        }).on("error", (err) => {
          rej(err);
        });
      });
      state.ipAddress = ipAddress;
      // Allow ingress from the IP address above to the security group.
      // This will allow you to SSH into the EC2 instance.
      const command = new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.securityGroupId,
        IpPermissions: [
          {
            IpProtocol: "tcp",
            FromPort: 22,
            ToPort: 22,
            IpRanges: [{ CidrIp: `${ipAddress}/32` }],
          },
        ],
      });

      await state.ec2Client.send(command);
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroupIngress = new ScenarioOutput(
  "logSecurityGroupIngress",
  (/** @type {State} */ state) =>
    `Allowed SSH access from your public IP: ${state.ipAddress}.`,
  { skipWhen: skipWhenErrors },
);
```

```
export const getImages = new ScenarioAction(
  "images",
  async (/** @type {State} */ state) => {
    const AMIs = [];
    // Some AWS services publish information about common artifacts as AWS Systems
 Manager (SSM)
    // public parameters. For example, the Amazon Elastic Compute Cloud (Amazon EC2)
    // service publishes information about Amazon Machine Images (AMIs) as public
 parameters.

    // Create the paginator for getting images. Actions that return multiple pages
 of
    // results have paginators to simplify those calls.
    const getParametersByPathPaginator = paginateGetParametersByPath(
      {
        // Not storing this client in state since it's only used once.
        client: new SSMClient({}),
      },
      {
        // The path to the public list of the latest amazon-linux instances.
        Path: "/aws/service/ami-amazon-linux-latest",
      },
    );

    try {
      for await (const page of getParametersByPathPaginator) {
        for (const param of page.Parameters) {
          // Filter by Amazon Linux 2
          if (param.Name.includes("amzn2")) {
            AMIs.push(param.Value);
          }
        }
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidFilterValue") {
        caught.message = `${caught.message} Please provide a valid filter value for
 paginateGetParametersByPath.`;
      }
      state.errors.push(caught);
      return;
    }

    const imageDetails = [];
    const describeImagesPaginator = paginateDescribeImages(
```

```
      { client: state.ec2Client },
      // The images found from the call to SSM.
      { ImageIds: AMIs },
    );

    try {
      // Get more details for the images found above.
      for await (const page of describeImagesPaginator) {
        imageDetails.push(...(page.Images || []));
      }

      // Store the image details for later use.
      state.images = imageDetails;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidAMIID.NotFound") {
        caught.message = `${caught.message}. Please provide a valid image id.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const provideImage = new ScenarioInput(
  "image",
  "Select one of the following images.",
  {
    type: "select",
    choices: (/** @type { State } */ state) =>
      state.images.map((image) => ({
        name: `${image.Description}`,
        value: image,
      })),
    default: (/** @type { State } */ state) => state.images[0],
    skipWhen: skipWhenErrors,
  },
);

export const getCompatibleInstanceTypes = new ScenarioAction(
  "getCompatibleInstanceTypes",
  async (/** @type {State} */ state) => {
    // Get more details about instance types that match the architecture of
    // the provided image.
```

```
    const paginator = paginateDescribeInstanceTypes(
      { client: state.ec2Client, pageSize: 25 },
      {
        Filters: [
          {
            Name: "processor-info.supported-architecture",
            // The value selected from provideImage()
            Values: [state.image.Architecture],
          },
          // Filter for smaller, less expensive, types.
          { Name: "instance-type", Values: ["*.micro", "*.small"] },
        ],
      },
    );

    const instanceTypes = [];

    try {
      for await (const page of paginator) {
        if (page.InstanceTypes.length) {
          instanceTypes.push(...(page.InstanceTypes || []));
        }
      }

      if (!instanceTypes.length) {
        state.errors.push(
          "No instance types matched the instance type filters.",
        );
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check the provided values and
 try again.`;
      }

      state.errors.push(caught);
    }

    state.instanceTypes = instanceTypes;
  },
  { skipWhen: skipWhenErrors },
);

export const provideInstanceType = new ScenarioInput(
```

```
  "instanceType",
  "Select an instance type.",
  {
    choices: (/** @type {State} */ state) =>
      state.instanceTypes.map((instanceType) => ({
        name: `${instanceType.InstanceType} - Memory:
${instanceType.MemoryInfo.SizeInMiB}`,
        value: instanceType.InstanceType,
      })),
    type: "select",
    default: (/** @type {State} */ state) =>
      state.instanceTypes[0].InstanceType,
    skipWhen: skipWhenErrors,
  },
);

export const runInstance = new ScenarioAction(
  "runInstance",
  async (/** @type { State } */ state) => {
    const { Instances } = await state.ec2Client.send(
      new RunInstancesCommand({
        KeyName: state[provideKeyPairName.name],
        SecurityGroupIds: [state.securityGroupId],
        ImageId: state.image.ImageId,
        InstanceType: state[provideInstanceType.name],
        // Availability Zones have capacity limitations that may impact your ability
 to launch instances.
        // The `RunInstances` operation will only succeed if it can allocate at
 least the `MinCount` of instances.
        // However, EC2 will attempt to launch up to the `MaxCount` of instances,
 even if the full request cannot be satisfied.
        // If you need a specific number of instances, use `MinCount` and `MaxCount`
 set to the same value.
        // If you want to launch up to a certain number of instances, use `MaxCount`
 and let EC2 provision as many as possible.
        // If you require a minimum number of instances, but do not want to exceed a
 maximum, use both `MinCount` and `MaxCount`.
        MinCount: 1,
        MaxCount: 1,
      }),
    );

    state.instanceId = Instances[0].InstanceId;
```

```
    try {
      // Poll `DescribeInstanceStatus` until status is "ok".
      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [Instances[0].InstanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the
 waiter.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logRunInstance = new ScenarioOutput(
  "logRunInstance",
  "The next step is to run your EC2 instance for the first time. This can take a few
 minutes.",
  { header: true, skipWhen: skipWhenErrors },
);

export const describeInstance = new ScenarioAction(
  "describeInstance",
  async (/** @type { State } */ state) => {
    /** @type { import("@aws-sdk/client-ec2").Instance[] } */
    const instances = [];

    try {
      const paginator = paginateDescribeInstances(
        {
          client: state.ec2Client,
        },
        {
          // Only get our created instance.
          InstanceIds: [state.instanceId],
        },
      );
```

```
      for await (const page of paginator) {
        for (const reservation of page.Reservations) {
          instances.push(...reservation.Instances);
        }
      }
      if (instances.length !== 1) {
        throw new Error(`Instance ${state.instanceId} not found.`);
      }

      // The only info we need is the IP address for SSH purposes.
      state.instanceIpAddress = instances[0].PublicIpAddress;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check provided values and try
 again.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSSHConnectionInfo = new ScenarioOutput(
  "logSSHConnectionInfo",
  (/** @type { State } */ state) =>
    `You can now SSH into your instance using the following command:
ssh -i ${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem ec2-user@
${state.instanceIpAddress}`,
  { preformatted: true, skipWhen: skipWhenErrors },
);

export const logStopInstance = new ScenarioOutput(
  "logStopInstance",
  "Stopping your EC2 instance.",
  { skipWhen: skipWhenErrors },
);

export const stopInstance = new ScenarioAction(
  "stopInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
```

```
        new StopInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStopped(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the
 waiter.`;
      }

      state.errors.push(caught);
    }
  },
  // Don't try to stop an instance that doesn't exist.
  { skipWhen: (/** @type { State } */ state) => !state.instanceId },
);

export const logIpAddressBehavior = new ScenarioOutput(
  "logIpAddressBehavior",
  [
    "When you run an instance, by default it's assigned an IP address.",
    "That IP address is not static. It will change every time the instance is
 restarted.",
    "The next step is to stop and restart your instance to demonstrate this
 behavior.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const logStartInstance = new ScenarioOutput(
  "logStartInstance",
  (/** @type { State } */ state) => `Starting instance ${state.instanceId}`,
  { skipWhen: skipWhenErrors },
);

export const startInstance = new ScenarioAction(
```

```
  "startInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StartInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the
 waiter.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logIpAllocation = new ScenarioOutput(
  "logIpAllocation",
  [
    "It is possible to have a static IP address.",
    "To demonstrate this, an IP will be allocated and associated to your EC2
 instance.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const allocateIp = new ScenarioAction(
  "allocateIp",
  async (/** @type { State } */ state) => {
    try {
      // An Elastic IP address is allocated to your AWS account, and is yours until
 you release it.
```

```
      const { AllocationId, PublicIp } = await state.ec2Client.send(
        new AllocateAddressCommand({}),
      );
      state.allocationId = AllocationId;
      state.allocatedIpAddress = PublicIp;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "MissingParameter") {
        caught.message = `${caught.message}. Did you provide these values?`;
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const associateIp = new ScenarioAction(
  "associateIp",
  async (/** @type { State } */ state) => {
    try {
      // Associate an allocated IP address to an EC2 instance. An IP address can be
 allocated
      // with the AllocateAddress action.
      const { AssociationId } = await state.ec2Client.send(
        new AssociateAddressCommand({
          AllocationId: state.allocationId,
          InstanceId: state.instanceId,
        }),
      );
      state.associationId = AssociationId;
      // Update the IP address that is being tracked to match
      // the one just associated.
      state.instanceIpAddress = state.allocatedIpAddress;
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
        caught.message = `${caught.message}. Did you provide the ID of a valid
 Elastic IP address AllocationId?`;
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
```

```
);

export const logStaticIpProof = new ScenarioOutput(
  "logStaticIpProof",
  "The IP address should remain the same even after stopping and starting the
 instance.",
  { header: true, skipWhen: skipWhenErrors },
);

export const logCleanUp = new ScenarioOutput(
  "logCleanUp",
  "That's it! You can choose to clean up the resources now, or clean them up on your
 own later.",
  { header: true, skipWhen: skipWhenErrors },
);

export const confirmDisassociateAddress = new ScenarioInput(
  "confirmDisassociateAddress",
  "Do you want to disassociate and release the static IP address created earlier?",
  {
    type: "confirm",
    skipWhen: (/** @type { State } */ state) => !state.associationId,
  },
);

export const maybeDisassociateAddress = new ScenarioAction(
  "maybeDisassociateAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new DisassociateAddressCommand({
          AssociationId: state.associationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAssociationID.NotFound"
      ) {
        caught.message = `${caught.message}. Please provide a valid association
 ID.`;
      }
      state.errors.push(caught);
    }
```

```
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.associationId,
  },
);

export const maybeReleaseAddress = new ScenarioAction(
  "maybeReleaseAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new ReleaseAddressCommand({
          AllocationId: state.allocationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
        caught.message = `${caught.message}. Please provide a valid AllocationID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.allocationId,
  },
);

export const confirmTerminateInstance = new ScenarioInput(
  "confirmTerminateInstance",
  "Do you want to terminate the instance?",
  // Don't do anything when an instance was never run.
  {
    skipWhen: (/** @type { State } */ state) => !state.instanceId,
    type: "confirm",
  },
);

export const maybeTerminateInstance = new ScenarioAction(
  "terminateInstance",
```

```
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new TerminateInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );
      await waitUntilInstanceTerminated(
        { client: state.ec2Client },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the
 waiter.`;
      }

      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no instance to terminate or the
    // use chooses not to terminate.
    skipWhen: (/** @type { State } */ state) =>
      !state.instanceId || !state[confirmTerminateInstance.name],
  },
);

export const deleteTemporaryDirectory = new ScenarioAction(
  "deleteTemporaryDirectory",
  async (/** @type { State } */ state) => {
    try {
      await rm(state.tmpDirectory, { recursive: true });
    } catch (caught) {
      state.errors.push(caught);
    }
  },
);

export const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
```

```
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    preformatted: true,
    header: true,
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { AllocateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Allocates an Elastic IP address to your AWS account.
 */
export const main = async () => {
  const client = new EC2Client({});
  const command = new AllocateAddressCommand({});

  try {
    const { AllocationId, PublicIp } = await client.send(command);
    console.log("A new IP address has been allocated to your account:");
    console.log(`ID: ${AllocationId} Public IP: ${PublicIp}`);
    console.log(
      "You can view your IP addresses in the AWS Management Console for Amazon EC2.
 Look under Network & Security > Elastic IPs",
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
import { fileURLToPath } from "node:url";
// Call function if run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
```

```
  main();
}
```

- Para obter detalhes da API, consulte AllocateAddress em Referência AWS SDK para JavaScript da API.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { AssociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Associates an Elastic IP address, or carrier IP address (for instances that are
 in subnets in Wavelength Zones)
 * with an instance or a network interface.
 * @param {{ instanceId: string, allocationId: string }} options
 */
export const main = async ({ instanceId, allocationId }) => {
  const client = new EC2Client({});
  const command = new AssociateAddressCommand({
    // You need to allocate an Elastic IP address before associating it with an
 instance.
    // You can do that with the AllocateAddressCommand.
    AllocationId: allocationId,
    // You need to create an EC2 instance before an IP address can be associated
 with it.
    // You can do that with the RunInstancesCommand.
    InstanceId: instanceId,
  });

  try {
```

```
    const { AssociationId } = await client.send(command);
    console.log(
      `Address with allocation ID ${allocationId} is now associated with instance
 ${instanceId}.`,
      `The association ID is ${AssociationId}.`,
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Did you provide the ID of a valid Elastic IP address
 AllocationId?`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte AssociateAddress em Referência AWS SDK para JavaScript da API.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  AuthorizeSecurityGroupIngressCommand,
  EC2Client,
} from "@aws-sdk/client-ec2";
```

```
/**
 * Adds the specified inbound (ingress) rules to a security group.
 * @param {{ groupId: string, ipAddress: string }} options
 */
export const main = async ({ groupId, ipAddress }) => {
  const client = new EC2Client({});
  const command = new AuthorizeSecurityGroupIngressCommand({
    // Use a group ID from the AWS console or
    // the DescribeSecurityGroupsCommand.
    GroupId: groupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        // The IP address to authorize.
        // For more information on this notation, see
        // https://en.wikipedia.org/wiki/Classless_Inter-
Domain_Routing#CIDR_notation
        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
      },
    ],
  });

  try {
    const { SecurityGroupRules } = await client.send(command);
    console.log(JSON.stringify(SecurityGroupRules, null, 2));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngress em Referência AWS SDK para JavaScript da API.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates an ED25519 or 2048-bit RSA key pair with the specified name and in the
 specified PEM or PPK format.
 * Amazon EC2 stores the public key and displays the private key for you to save to
 a file.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new CreateKeyPairCommand({
    KeyName: keyName,
  });

  try {
    const { KeyMaterial, KeyName } = await client.send(command);
    console.log(KeyName);
    console.log(KeyMaterial);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidKeyPair.Duplicate") {
      console.warn(`${caught.message}. Try another key name.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateKeyPair em Referência AWS SDK para JavaScript da API.

`CreateLaunchTemplate`

O código de exemplo a seguir mostra como usar `CreateLaunchTemplate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const ssmClient = new SSMClient({});
const { Parameter } = await ssmClient.send(
  new GetParameterCommand({
    Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
  }),
);
const ec2Client = new EC2Client({});
await ec2Client.send(
  new CreateLaunchTemplateCommand({
    LaunchTemplateName: NAMES.launchTemplateName,
    LaunchTemplateData: {
      InstanceType: "t3.micro",
      ImageId: Parameter.Value,
      IamInstanceProfile: { Name: NAMES.instanceProfileName },
      UserData: readFileSync(
        join(RESOURCES_PATH, "server_startup_script.sh"),
      ).toString("base64"),
      KeyName: NAMES.keyPairName,
    },
  }),
```

- Para obter detalhes da API, consulte CreateLaunchTemplate em Referência AWS SDK para JavaScript da API.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates a security group.
 * @param {{ groupName: string, description: string }} options
 */
export const main = async ({ groupName, description }) => {
  const client = new EC2Client({});
  const command = new CreateSecurityGroupCommand({
    // Up to 255 characters in length. Cannot start with sg-.
    GroupName: groupName,
    // Up to 255 characters in length.
    Description: description,
  });

  try {
    const { GroupId } = await client.send(command);
    console.log(GroupId);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateSecurityGroup em Referência AWS SDK para JavaScript da API.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes the specified key pair, by removing the public key from Amazon EC2.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new DeleteKeyPairCommand({
    KeyName: keyName,
  });

  try {
    await client.send(command);
    console.log("Successfully deleted key pair.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide the required value?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteKeyPair em Referência AWS SDK para JavaScript da API.

`DeleteLaunchTemplate`

O código de exemplo a seguir mostra como usar `DeleteLaunchTemplate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    await client.send(
      new DeleteLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
      }),
    );
```

- Para obter detalhes da API, consulte DeleteLaunchTemplate em Referência AWS SDK para JavaScript da API.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes a security group.
 * @param {{ groupId: string }} options
 */
```

```
export const main = async ({ groupId }) => {
  const client = new EC2Client({});
  const command = new DeleteSecurityGroupCommand({
    GroupId: groupId,
  });

  try {
    await client.send(command);
    console.log("Security group deleted successfully.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteSecurityGroup em Referência AWS SDK para JavaScript da API.

`DescribeAddresses`

O código de exemplo a seguir mostra como usar `DescribeAddresses`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeAddressesCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified Elastic IP addresses or all of your Elastic IP addresses.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
```

```
  const client = new EC2Client({});
  const command = new DescribeAddressesCommand({
    // You can omit this property to show all addresses.
    AllocationIds: [allocationId],
  });

  try {
    const { Addresses } = await client.send(command);
    const addressList = Addresses.map((address) => ` • ${address.PublicIp}`);
    console.log("Elastic IP addresses:");
    console.log(addressList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationId.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeAddresses em Referência AWS SDK para JavaScript da API.

`DescribeIamInstanceProfileAssociations`

O código de exemplo a seguir mostra como usar `DescribeIamInstanceProfileAssociations`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
```

```
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
```

- Para obter detalhes da API, consulte DescribeIamInstanceProfileAssociations em Referência AWS SDK para JavaScript da API.

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, paginateDescribeImages } from "@aws-sdk/client-ec2";

/**
 * Describes the specified images (AMIs, AKIs, and ARIs) available to you or all of
 the images available to you.
 * @param {{ architecture: string, pageSize: number }} options
 */
export const main = async ({ architecture, pageSize }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the base command.
  const paginator = paginateDescribeImages(
    // Without limiting the page size, this call can take a long time. pageSize is
 just sugar for
    // the MaxResults property in the base command.
    { client, pageSize },
    {
```

```
      // There are almost 70,000 images available. Be specific with your filtering
      // to increase efficiency.
      // See https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
ec2/interfaces/describeimagescommandinput.html#filters
      Filters: [{ Name: "architecture", Values: [architecture] }],
    },
  );

  /**
   * @type {import('@aws-sdk/client-ec2').Image[]}
   */
  const images = [];
  let recordsScanned = 0;

  try {
    for await (const page of paginator) {
      recordsScanned += pageSize;
      if (page.Images.length) {
        images.push(...page.Images);
        break;
      }
      console.log(
        `No matching image found yet. Searched ${recordsScanned} records.`,
      );
    }

    if (images.length) {
      console.log(
        `Found ${images.length} images:\n\n${images.map((image) =>
 image.Name).join("\n")}\n`,
      );
    } else {
      console.log(
        `No matching images found. Searched ${recordsScanned} records.\n`,
      );
    }

    return images;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
      return [];
    }
    throw caught;
```

```
  }
};
```

- Para obter detalhes da API, consulte DescribeImages em Referência AWS SDK para JavaScript da API.

DescribeInstanceTypes

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, paginateDescribeInstanceTypes } from "@aws-sdk/client-ec2";

/**
 * Describes the specified instance types. By default, all instance types for the
 * current Region are described. Alternatively, you can filter the results.
 * @param {{ pageSize: string, supportedArch: string[], freeTier: boolean }} options
 */
export const main = async ({ pageSize, supportedArch, freeTier }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the underlying command.
  const paginator = paginateDescribeInstanceTypes(
    // Without limiting the page size, this call can take a long time. pageSize is
 just sugar for
    // the MaxResults property in the underlying command.
    { client, pageSize },
    {
      Filters: [
        {
          Name: "processor-info.supported-architecture",
          Values: supportedArch,
```

```
        },
        { Name: "free-tier-eligible", Values: [freeTier ? "true" : "false"] },
      ],
    },
  );

  try {
    /**
     * @type {import('@aws-sdk/client-ec2').InstanceTypeInfo[]}
     */
    const instanceTypes = [];

    for await (const page of paginator) {
      if (page.InstanceTypes.length) {
        instanceTypes.push(...page.InstanceTypes);

        // When we have at least 1 result, we can stop.
        if (instanceTypes.length >= 1) {
          break;
        }
      }
    }
    console.log(
      `Memory size in MiB for matching instance types:\n\n${instanceTypes.map((it)
 => `${it.InstanceType}: ${it.MemoryInfo.SizeInMiB} MiB`).join("\n")}`,
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
      return [];
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte DescribeInstanceTypes em Referência AWS SDK para JavaScript da API.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, paginateDescribeInstances } from "@aws-sdk/client-ec2";

/**
 * List all of your EC2 instances running with the provided architecture that
 * were launched in the past month.
 * @param {{ pageSize: string, architectures: string[] }} options
 */
export const main = async ({ pageSize, architectures }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});
  const d = new Date();
  const year = d.getFullYear();
  const month = `0${d.getMonth() + 1}`.slice(-2);
  const launchTimePattern = `${year}-${month}-*`;

  const paginator = paginateDescribeInstances(
    {
      client,
      pageSize,
    },
    {
      Filters: [
        { Name: "architecture", Values: architectures },
        { Name: "instance-state-name", Values: ["running"] },
        {
          Name: "launch-time",
          Values: [launchTimePattern],
        },
      ],
    },
  );

  try {
    /**
```

```
     * @type {import('@aws-sdk/client-ec2').Instance[]}
     */
    const instanceList = [];
    for await (const page of paginator) {
      const { Reservations } = page;
      for (const reservation of Reservations) {
        instanceList.push(...reservation.Instances);
      }
    }
    console.log(
      `Running instances launched this month:\n\n${instanceList.map((instance) =>
 instance.InstanceId).join("\n")}`,
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeInstances em Referência AWS SDK para JavaScript da API.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeKeyPairsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
```

```
 * List all key pairs in the current AWS account.
 * @param {{ dryRun: boolean }}
 */
export const main = async ({ dryRun }) => {
  const client = new EC2Client({});
  const command = new DescribeKeyPairsCommand({ DryRun: dryRun });

  try {
    const { KeyPairs } = await client.send(command);
    const keyPairList = KeyPairs.map(
      (kp) => ` • ${kp.KeyPairId}: ${kp.KeyName}`,
    ).join("\n");
    console.log("The following key pairs were found in your account:");
    console.log(keyPairList);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeKeyPairs em Referência AWS SDK para JavaScript da API.

`DescribeRegions`

O código de exemplo a seguir mostra como usar `DescribeRegions`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeRegionsCommand, EC2Client } from "@aws-sdk/client-ec2";
```

```
/**
 * List all available AWS regions.
 * @param {{ regionNames: string[], includeOptInRegions: boolean }} options
 */
export const main = async ({ regionNames, includeOptInRegions }) => {
  const client = new EC2Client({});
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions is true, even the regions that require opt-in will be
 returned.
    AllRegions: includeOptInRegions,
    // You can omit the Filters property if you want to get all regions.
    Filters: regionNames?.length
      ? [
          {
            Name: "region-name",
            // You can specify multiple values for a filter.
            // You can also use '*' as a wildcard. This will return all
            // of the regions that start with `us-east-`.
            Values: regionNames,
          },
        ]
      : undefined,
  });

  try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeRegions em Referência AWS SDK para JavaScript da API.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified security groups or all of your security groups.
 * @param {{ groupIds: string[] }} options
 */
export const main = async ({ groupIds = [] }) => {
  const client = new EC2Client({});
  const command = new DescribeSecurityGroupsCommand({
    GroupIds: groupIds,
  });

  try {
    const { SecurityGroups } = await client.send(command);
    const sgList = SecurityGroups.map(
      (sg) => `• ${sg.GroupName} (${sg.GroupId}): ${sg.Description}`,
    ).join("\n");
    if (sgList.length) {
      console.log(`Security groups:\n${sgList}`);
    } else {
      console.log("No security groups found.");
    }
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else if (
      caught instanceof Error &&
      caught.name === "InvalidGroup.NotFound"
    ) {
      console.warn(caught.message);
    } else {
```

```
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK para JavaScript da API.

DescribeSubnets

O código de exemplo a seguir mostra como usar `DescribeSubnets`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new EC2Client({});
const { Subnets } = await client.send(
  new DescribeSubnetsCommand({
    Filters: [
      { Name: "vpc-id", Values: [state.defaultVpc] },
      { Name: "availability-zone", Values: state.availabilityZoneNames },
      { Name: "default-for-az", Values: ["true"] },
    ],
  }),
);
```

- Para obter detalhes da API, consulte DescribeSubnets em Referência AWS SDK para JavaScript da API.

DescribeVpcs

O código de exemplo a seguir mostra como usar `DescribeVpcs`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
```

- Para obter detalhes da API, consulte DescribeVpcs em Referência AWS SDK para JavaScript da API.

`DisassociateAddress`

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DisassociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Disassociate an Elastic IP address from an instance.
 * @param {{ associationId: string }} options
 */
export const main = async ({ associationId }) => {
  const client = new EC2Client({});
  const command = new DisassociateAddressCommand({
```

```
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AssociationId: associationId,
  });

  try {
    await client.send(command);
    console.log("Successfully disassociated address");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAssociationID.NotFound"
    ) {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DisassociateAddress em Referência AWS SDK para JavaScript da API.

`MonitorInstances`

O código de exemplo a seguir mostra como usar `MonitorInstances`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, MonitorInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Turn on detailed monitoring for the selected instance.
 * By default, metrics are sent to Amazon CloudWatch every 5 minutes.
```

```
 * For a cost you can enable detailed monitoring which sends metrics every minute.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new MonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instancesBeingMonitored = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
 ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instancesBeingMonitored.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte MonitorInstances em Referência AWS SDK para JavaScript da API.

`RebootInstances`

O código de exemplo a seguir mostra como usar `RebootInstances`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, RebootInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Requests a reboot of the specified instances. This operation is asynchronous;
 * it only queues a request to reboot the specified instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new RebootInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    await client.send(command);
    console.log("Instance rebooted successfully.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Please provide the InstanceId of a valid instance to
 reboot.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte RebootInstances em Referência AWS SDK para JavaScript da API.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ReleaseAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Release an Elastic IP address.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new ReleaseAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AllocationId: allocationId,
  });

  try {
    await client.send(command);
    console.log("Successfully released address.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationID.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte ReleaseAddress em Referência AWS SDK para JavaScript da API.

`ReplaceIamInstanceProfileAssociation`

O código de exemplo a seguir mostra como usar `ReplaceIamInstanceProfileAssociation`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );
```

- Para obter detalhes da API, consulte ReplaceIamInstanceProfileAssociation em Referência AWS SDK para JavaScript da API.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, RunInstancesCommand } from "@aws-sdk/client-ec2";

/**
```

```
 * Create new EC2 instances.
 * @param {{
 *  keyName: string,
 *  securityGroupIds: string[],
 *  imageId: string,
 *  instanceType: import('@aws-sdk/client-ec2')._InstanceType,
 *  minCount?: number,
 *  maxCount?: number }} options
 */
export const main = async ({
  keyName,
  securityGroupIds,
  imageId,
  instanceType,
  minCount = "1",
  maxCount = "1",
}) => {
  const client = new EC2Client({});
  minCount = Number.parseInt(minCount);
  maxCount = Number.parseInt(maxCount);
  const command = new RunInstancesCommand({
    // Your key pair name.
    KeyName: keyName,
    // Your security group.
    SecurityGroupIds: securityGroupIds,
    // An Amazon Machine Image (AMI). There are multiple ways to search for AMIs.
 For more information, see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html
    ImageId: imageId,
    // An instance type describing the resources provided to your instance. There
 are multiple
    // ways to search for instance types. For more information see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-discovery.html
    InstanceType: instanceType,
    // Availability Zones have capacity limitations that may impact your ability to
 launch instances.
    // The `RunInstances` operation will only succeed if it can allocate at least
 the `MinCount` of instances.
    // However, EC2 will attempt to launch up to the `MaxCount` of instances, even
 if the full request cannot be satisfied.
    // If you need a specific number of instances, use `MinCount` and `MaxCount` set
 to the same value.
    // If you want to launch up to a certain number of instances, use `MaxCount` and
 let EC2 provision as many as possible.
```

```
    // If you require a minimum number of instances, but do not want to exceed a
 maximum, use both `MinCount` and `MaxCount`.
    MinCount: minCount,
    MaxCount: maxCount,
  });

  try {
    const { Instances } = await client.send(command);
    const instanceList = Instances.map(
      (instance) => `• ${instance.InstanceId}`,
    ).join("\n");
    console.log(`Launched instances:\n${instanceList}`);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceCountExceeded") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte RunInstances em Referência AWS SDK para JavaScript da API.

StartInstances

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, StartInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
```

```
/**
 * Starts an Amazon EBS-backed instance that you've previously stopped.
 * @param {{ instanceIds }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StartInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StartingInstances } = await client.send(command);
    const instanceIdList = StartingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Starting instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte StartInstances em Referência AWS SDK para JavaScript da API.

StopInstances

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, StopInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Stop one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StopInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte StopInstances em Referência AWS SDK para JavaScript da API.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, TerminateInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Terminate one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new TerminateInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { TerminatingInstances } = await client.send(command);
    const instanceList = TerminatingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Terminating instances:");
    console.log(instanceList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
```

```
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte TerminateInstances em Referência AWS SDK para JavaScript da API.

`UnmonitorInstances`

O código de exemplo a seguir mostra como usar `UnmonitorInstances`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { EC2Client, UnmonitorInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Turn off detailed monitoring for the selected instance.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new UnmonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instanceMonitoringsList = InstanceMonitorings.map(
      (im) =>
```

```
        ` • Detailed monitoring state for ${im.InstanceId} is
 ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instanceMonitoringsList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte UnmonitorInstances em Referência AWS SDK para JavaScript da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.

- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
```

```
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Criar etapas para implantar todos os recursos.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
```

```
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
```

```
    new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
    new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
      type: "confirm",
    }),
    new ScenarioAction(
      "handleConfirmDeployment",
      (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
      "creatingTable",
      MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
      const client = new DynamoDBClient({});
      await client.send(
        new CreateTableCommand({
          TableName: NAMES.tableName,
          ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
          },
          AttributeDefinitions: [
            {
              AttributeName: "MediaType",
              AttributeType: "S",
            },
            {
              AttributeName: "ItemId",
              AttributeType: "N",
            },
          ],
          KeySchema: [
            {
              AttributeName: "MediaType",
              KeyType: "HASH",
            },
            {
              AttributeName: "ItemId",
              KeyType: "RANGE",
            },
          ],
        }),
      );
      await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
```

```
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item'][] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );
```

```
    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
```

```
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
```

```
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
```

```
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
```

```
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
```

```
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
```

```
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
```

```
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}}
state
     */
```

```
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
```

```
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
```

```
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Criar etapas para executar a demonstração.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
```

```
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
```

```
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);
```

```
const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
```

```
    new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
    new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
    ...statusSteps,
    new ScenarioInput(
      "brokenDependencyConfirmation",
      MESSAGES.demoBrokenDependencyConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("brokenDependency", async (state) => {
      if (!state.brokenDependencyConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        state.badTableName = `fake-table-${Date.now()}`;
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmTableNameKey,
            Value: state.badTableName,
            Overwrite: true,
            Type: "String",
          }),
        );
      }
    }),
    new ScenarioOutput("testBrokenDependency", (state) =>
      MESSAGES.demoTestBrokenDependency.replace(
        "${TABLE_NAME}",
        state.badTableName,
      ),
    ),
    ...statusSteps,
    new ScenarioInput(
      "staticResponseConfirmation",
      MESSAGES.demoStaticResponseConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("staticResponse", async (state) => {
      if (!state.staticResponseConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmFailureResponseKey,
```

```
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-scaling').Instance }}
state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
```

```
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
        InstanceIds: [state.targetInstance.InstanceId],
```

```
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}}
state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
```

```
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
```

```
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
```

```
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
```

```
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Criar etapas para destruir todos os recursos.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
```

```
import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
```

```
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
```

```
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
```

```
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
```

```
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
```

```
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
```

```
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
```

```
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
```

```
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
```

```
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
```

```
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}
```

```
/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachLoadBalancerTargetGroups

- CreateAutoScalingGroup
- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Elastic Load Balancing - exemplos da versão 2 usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Elastic Load Balancing - Versão 2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Elastic Load Balancing

O exemplo de código a seguir mostra como começar a usar o Elastic Load Balancing.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  ElasticLoadBalancingV2Client,
  DescribeLoadBalancersCommand,
} from "@aws-sdk/client-elastic-load-balancing-v2";

export async function main() {
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
    new DescribeLoadBalancersCommand({}),
  );
```

```
  const loadBalancersList = LoadBalancers.map(
    (lb) => `• ${lb.LoadBalancerName}: ${lb.DNSName}`,
  ).join("\n");
  console.log(
    "Hello, Elastic Load Balancing! Let's list some of your load balancers:\n",
    loadBalancersList,
  );
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancers em Referência AWS SDK para JavaScript da API.

Ações

`CreateListener`

O código de exemplo a seguir mostra como usar `CreateListener`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
```

```
      ],
    }),
  );
```

- Para obter detalhes da API, consulte CreateListener em Referência AWS SDK para JavaScript da API.

`CreateLoadBalancer`

O código de exemplo a seguir mostra como usar `CreateLoadBalancer`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
const { LoadBalancers } = await client.send(
  new CreateLoadBalancerCommand({
    Name: NAMES.loadBalancerName,
    Subnets: state.subnets,
  }),
);
state.loadBalancerDns = LoadBalancers[0].DNSName;
state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
await waitUntilLoadBalancerAvailable(
  { client },
  { Names: [NAMES.loadBalancerName] },
);
```

- Para obter detalhes da API, consulte CreateLoadBalancer em Referência AWS SDK para JavaScript da API.

`CreateTargetGroup`

O código de exemplo a seguir mostra como usar `CreateTargetGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
const { TargetGroups } = await client.send(
  new CreateTargetGroupCommand({
    Name: NAMES.loadBalancerTargetGroupName,
    Protocol: "HTTP",
    Port: 80,
    HealthCheckPath: "/healthcheck",
    HealthCheckIntervalSeconds: 10,
    HealthCheckTimeoutSeconds: 5,
    HealthyThresholdCount: 2,
    UnhealthyThresholdCount: 2,
    VpcId: state.defaultVpc,
  }),
);
```

- Para obter detalhes da API, consulte CreateTargetGroup em Referência AWS SDK para JavaScript da API.

`DeleteLoadBalancer`

O código de exemplo a seguir mostra como usar `DeleteLoadBalancer`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
await client.send(
  new DeleteLoadBalancerCommand({
    LoadBalancerArn: loadBalancer.LoadBalancerArn,
  }),
);
await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
  const lb = await findLoadBalancer(NAMES.loadBalancerName);
  if (lb) {
    throw new Error("Load balancer still exists.");
  }
});
```

- Para obter detalhes da API, consulte DeleteLoadBalancer em Referência AWS SDK para JavaScript da API.

`DeleteTargetGroup`

O código de exemplo a seguir mostra como usar `DeleteTargetGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new ElasticLoadBalancingV2Client({});
```

```
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
```

- Para obter detalhes da API, consulte DeleteTargetGroup em Referência AWS SDK para JavaScript da API.

`DescribeLoadBalancers`

O código de exemplo a seguir mostra como usar `DescribeLoadBalancers`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  ElasticLoadBalancingV2Client,
  DescribeLoadBalancersCommand,
} from "@aws-sdk/client-elastic-load-balancing-v2";

export async function main() {
```

```
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
    new DescribeLoadBalancersCommand({}),
  );
  const loadBalancersList = LoadBalancers.map(
    (lb) => `• ${lb.LoadBalancerName}: ${lb.DNSName}`,
  ).join("\n");
  console.log(
    "Hello, Elastic Load Balancing! Let's list some of your load balancers:\n",
    loadBalancersList,
  );
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancers em Referência AWS SDK para JavaScript da API.

`DescribeTargetGroups`

O código de exemplo a seguir mostra como usar `DescribeTargetGroups`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );
```

- Para obter detalhes da API, consulte DescribeTargetGroups em Referência AWS SDK para JavaScript da API.

`DescribeTargetHealth`

O código de exemplo a seguir mostra como usar `DescribeTargetHealth`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const { TargetHealthDescriptions } = await client.send(
  new DescribeTargetHealthCommand({
    TargetGroupArn: TargetGroups[0].TargetGroupArn,
  }),
);
```

- Para obter detalhes da API, consulte DescribeTargetHealth em Referência AWS SDK para JavaScript da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.

- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
```

```
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Criar etapas para implantar todos os recursos.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
```

```
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
```

```
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
```

```
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item'][] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
```

```
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
```

```
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
```

```
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
```

```
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
```

```
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
```

```
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
```

```
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
```

```
    .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
```

```
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}}
state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
```

```
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
```

```
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Criar etapas para executar a demonstração.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
```

```
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
```

```
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
```

```
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
```

```
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
```

```
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-scaling').Instance }}
state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
```

```
    const autoScalingClient = new AutoScalingClient({});
    const { AutoScalingGroups } = await autoScalingClient.send(
      new DescribeAutoScalingGroupsCommand({
        AutoScalingGroupNames: [NAMES.autoScalingGroupName],
      }),
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
```

```
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}}
state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
```

```
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
```

```
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
```

```
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
```

```
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Criar etapas para destruir todos os recursos.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
```

```
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
```

```
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
```

```
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
```

```
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
```

```
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
```

```
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
```

```
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
```

```
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
```

```
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
```

```
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
```

```
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
```

```
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
```

```
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

AWS Entity Resolution exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com AWS Entity Resolution.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Entity Resolution

O exemplo de código a seguir mostra como começar a usar o AWS Entity Resolution.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  EntityResolutionClient,
  ListMatchingWorkflowsCommand,
} from "@aws-sdk/client-entityresolution";

export const main = async () => {
  const region = "eu-west-1";
  const erClient = new EntityResolutionClient({ region: region });
  try {
```

```
    const command = new ListMatchingWorkflowsCommand({});
    const response = await erClient.send(command);
    const workflowSummaries = response.workflowSummaries;
    for (const workflowSummary of workflowSummaries) {
      console.log(`Attribute name: ${workflowSummaries[0].workflowName} `);
    }
    if (workflowSummaries.length === 0) {
      console.log("No matching workflows found.");
    }
  } catch (error) {
    console.error(
      `An error occurred in listing the workflow summaries: ${error.message} \n
 Exiting program.`,
    );
    return;
  }
};
```

- Para obter detalhes da API, consulte ListMatchingWorkflows em Referência AWS SDK para JavaScript da API.

Ações

`CreateMatchingWorkflow`

O código de exemplo a seguir mostra como usar `CreateMatchingWorkflow`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";
```

```
import {
  CreateMatchingWorkflowCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const createMatchingWorkflowParams = {
    roleArn: `${data.inputs.roleArn}`,
    workflowName: `${data.inputs.workflowName}`,
    description: "Created by using the AWS SDK for JavaScript (v3).",
    inputSourceConfig: [
      {
        inputSourceARN: `${data.inputs.JSONinputSourceARN}`,
        schemaName: `${data.inputs.schemaNameJson}`,
        applyNormalization: false,
      },
      {
        inputSourceARN: `${data.inputs.CSVinputSourceARN}`,
        schemaName: `${data.inputs.schemaNameCSV}`,
        applyNormalization: false,
      },
    ],
    outputSourceConfig: [
      {
        outputS3Path: `s3://${data.inputs.myBucketName}/eroutput`,
        output: [
          {
            name: "id",
          },
          {
            name: "name",
          },
          {
            name: "email",
          },
          {
            name: "phone",
          },
        ],
```

```
        applyNormalization: false,
      },
    ],
    resolutionTechniques: { resolutionType: "ML_MATCHING" },
  };
  try {
    const command = new CreateMatchingWorkflowCommand(
      createMatchingWorkflowParams,
    );
    const response = await erClient.send(command);

    console.log(
      `Workflow created successfully.\n The workflow ARN is:
 ${response.workflowArn}`,
    );
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte CreateMatchingWorkflow em Referência AWS SDK para JavaScript da API.

`CreateSchemaMapping`

O código de exemplo a seguir mostra como usar `CreateSchemaMapping`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.
```

```
import { fileURLToPath } from "node:url";

import {
  CreateSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const createSchemaMappingParamsJson = {
    schemaName: `${data.inputs.schemaNameJson}`,
    mappedInputFields: [
      {
        fieldName: "id",
        type: "UNIQUE_ID",
      },
      {
        fieldName: "name",
        type: "NAME",
      },
      {
        fieldName: "email",
        type: "EMAIL_ADDRESS",
      },
    ],
  };
  const createSchemaMappingParamsCSV = {
    schemaName: `${data.inputs.schemaNameCSV}`,
    mappedInputFields: [
      {
        fieldName: "id",
        type: "UNIQUE_ID",
      },
      {
        fieldName: "name",
        type: "NAME",
      },
      {
        fieldName: "email",
        type: "EMAIL_ADDRESS",
      },
```

```
      {
        fieldName: "phone",
        type: "PROVIDER_ID",
        subType: "STRING",
      },
    ],
  };
  try {
    const command = new CreateSchemaMappingCommand(
      createSchemaMappingParamsJson,
    );
    const response = await erClient.send(command);
    console.log("The JSON schema mapping name is ", response.schemaName);
  } catch (error) {
    console.log("error ", error.message);
  }
};
```

- Para obter detalhes da API, consulte CreateSchemaMapping em Referência AWS SDK para JavaScript da API.

`DeleteMatchingWorkflow`

O código de exemplo a seguir mostra como usar `DeleteMatchingWorkflow`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
```

```
  DeleteMatchingWorkflowCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  try {
    const deleteWorkflowParams = {
      workflowName: `${data.inputs.workflowName}`,
    };
    const command = new DeleteMatchingWorkflowCommand(deleteWorkflowParams);
    const response = await erClient.send(command);
    console.log("Workflow deleted successfully!", response);
  } catch (error) {
    console.log("error ", error);
  }
};
```

- Para obter detalhes da API, consulte DeleteMatchingWorkflow em Referência AWS SDK para JavaScript da API.

`DeleteSchemaMapping`

O código de exemplo a seguir mostra como usar `DeleteSchemaMapping`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.
```

```
import { fileURLToPath } from "node:url";

import {
  DeleteSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const deleteSchemaMapping = {
    schemaName: `${data.inputs.schemaNameJson}`,
  };
  try {
    const command = new DeleteSchemaMappingCommand(deleteSchemaMapping);
    const response = await erClient.send(command);
    console.log("Schema mapping deleted successfully. ", response);
  } catch (error) {
    console.log("error ", error);
  }
};
```

- Para obter detalhes da API, consulte DeleteSchemaMapping em Referência AWS SDK para JavaScript da API.

`GetMatchingJob`

O código de exemplo a seguir mostra como usar `GetMatchingJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  GetMatchingJobCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  async function getInfo() {
    const getJobInfoParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${data.inputs.jobId}`,
    };
    try {
      const command = new GetMatchingJobCommand(getJobInfoParams);
      const response = await erClient.send(command);
      console.log(`Job status: ${response.status}`);
    } catch (error) {
      console.log("error ", error.message);
    }
  }
};
```

- Para obter detalhes da API, consulte GetMatchingJob em Referência AWS SDK para JavaScript da API.

`GetSchemaMapping`

O código de exemplo a seguir mostra como usar `GetSchemaMapping`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  GetSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const getSchemaMappingJsonParams = {
    schemaName: `${data.inputs.schemaNameJson}`,
  };
  try {
    const command = new GetSchemaMappingCommand(getSchemaMappingJsonParams);
    const response = await erClient.send(command);
    console.log(response);
    console.log(
      `Schema mapping for the JSON data:\n ${response.mappedInputFields[0]}`,
    );
    console.log("Schema mapping ARN is: ", response.schemaArn);
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte GetSchemaMapping em Referência AWS SDK para JavaScript da API.

`ListSchemaMappings`

O código de exemplo a seguir mostra como usar `ListSchemaMappings`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  ListSchemaMappingsCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  async function getInfo() {
    const listSchemaMappingsParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${data.inputs.jobId}`,
    };
    try {
      const command = new ListSchemaMappingsCommand(listSchemaMappingsParams);
      const response = await erClient.send(command);
      const noOfSchemas = response.schemaList.length;
      for (let i = 0; i < noOfSchemas; i++) {
        console.log(
          `Schema Mapping Name: ${response.schemaList[i].schemaName} `,
```

```
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  }
```

- Para obter detalhes da API, consulte ListSchemaMappings em Referência AWS SDK para JavaScript da API.

StartMatchingJob

O código de exemplo a seguir mostra como usar `StartMatchingJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";
import {
  StartMatchingJobCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const matchingJobOfWorkflowParams = {
    workflowName: `${data.inputs.workflowName}`,
```

```
  };
  try {
    const command = new StartMatchingJobCommand(matchingJobOfWorkflowParams);
    const response = await erClient.send(command);
    console.log(`Job ID: ${response.jobID} \n
The matching job was successfully started.`);
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte StartMatchingJob em Referência AWS SDK para JavaScript da API.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  TagResourceCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
```

```
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const tagResourceCommandParams = {
    resourceArn: `${data.inputs.schemaArn}`,
    tags: {
      tag1: "tag1Value",
      tag2: "tag2Value",
    },
  };
  try {
    const command = new TagResourceCommand(tagResourceCommandParams);
    const response = await erClient.send(command);
    console.log("Successfully tagged the resource.");
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte TagResource em Referência AWS SDK para JavaScript da API.

EventBridge exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com EventBridge.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import {
  EventBridgeClient,
  PutEventsCommand,
} from "@aws-sdk/client-eventbridge";

export const putEvents = async (
  source = "eventbridge.integration.test",
  detailType = "greeting",
  resources = [],
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutEventsCommand({
      Entries: [
        {
          Detail: JSON.stringify({ greeting: "Hello there." }),
          DetailType: detailType,
          Resources: resources,
          Source: source,
        },
      ],
```

```
    }),
  );

  console.log("PutEvents response:");
  console.log(response);
  // PutEvents response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3d0df73d-dcea-4a23-ae0d-f5556a3ac109',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Entries: [ { EventId: '51620841-5af4-6402-d9bc-b77734991eb5' } ],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK para JavaScript da API.

`PutRule`

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import { EventBridgeClient, PutRuleCommand } from "@aws-sdk/client-eventbridge";
```

```
export const putRule = async (
  ruleName = "some-rule",
  source = "some-source",
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutRuleCommand({
      Name: ruleName,
      EventPattern: JSON.stringify({ source: [source] }),
      State: "ENABLED",
      EventBusName: "default",
    }),
  );

  console.log("PutRule response:");
  console.log(response);
  // PutRule response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd7292ced-1544-421b-842f-596326bc7072',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   RuleArn: 'arn:aws:events:us-east-1:xxxxxxxxxxxx:rule/
EventBridgeTestRule-1696280037720'
  // }
  return response;
};
```

- Para obter detalhes da API, consulte PutRule em Referência AWS SDK para JavaScript da API.

PutTargets

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importe o SDK e os módulos do cliente e chame a API.

```
import {
  EventBridgeClient,
  PutTargetsCommand,
} from "@aws-sdk/client-eventbridge";

export const putTarget = async (
  existingRuleName = "some-rule",
  targetArn = "arn:aws:lambda:us-east-1:000000000000:function:test-func",
  uniqueId = Date.now().toString(),
) => {
  const client = new EventBridgeClient({});
  const response = await client.send(
    new PutTargetsCommand({
      Rule: existingRuleName,
      Targets: [
        {
          Arn: targetArn,
          Id: uniqueId,
        },
      ],
    }),
  );

  console.log("PutTargets response:");
  console.log(response);
  // PutTargets response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'f5b23b9a-2c17-45c1-ad5c-f926c3692e3d',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   FailedEntries: [],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- Para obter detalhes da API, consulte PutTargets em Referência AWS SDK para JavaScript da API.

Cenários

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge

- Lambda
- Amazon SNS

Exemplos do Amazon Glacier usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Glacier.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateVault`

O código de exemplo a seguir mostra como usar `CreateVault`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Crie o cofre.

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateVault em Referência AWS SDK para JavaScript da API.

`UploadArchive`

O código de exemplo a seguir mostra como usar `UploadArchive`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Faça upload do arquivo.

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME

// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
  try {
    const data = await glacierClient.send(new UploadArchiveCommand(params));
    console.log("Archive ID", data.archiveId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error uploading archive!", err);
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UploadArchive em Referência AWS SDK para JavaScript da API.

AWS Glue exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ListJobsCommand, GlueClient } from "@aws-sdk/client-glue";

const client = new GlueClient({});

export const main = async () => {
  const command = new ListJobsCommand({});

  const { JobNames } = await client.send(command);
  const formattedJobNames = JobNames.join("\n");
  console.log("Job names: ");
  console.log(formattedJobNames);
```

```
  return JobNames;
};
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie e execute um rastreador que rastreia um bucket público do Amazon Simple Storage Service (Amazon S3) e gera um banco de dados de metadados que descreve os dados encontrados. CSV-formatted

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
```

```
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const crawlerExists = async ({ getCrawler }, crawlerName) => {
  try {
    await getCrawler(crawlerName);
    return true;
  } catch {
    return false;
  }
};

/**
 * @param {{ createCrawler: import('../../../actions/create-
crawler.js').createCrawler}} actions
 */
```

```
const makeCreateCrawlerStep = (actions) => async (context) => {
  if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
    log("Crawler already exists. Skipping creation.");
  } else {
    await actions.createCrawler(
      process.env.CRAWLER_NAME,
      process.env.ROLE_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_PREFIX,
      process.env.S3_TARGET_PATH,
    );

    log("Crawler created successfully.", { type: "success" });
  }

  return { ...context };
};

/**
 * @param {(name: string) => Promise<import('@aws-sdk/client-
glue').GetCrawlerCommandOutput>} getCrawler
 * @param {string} crawlerName
 */
const waitForCrawler = async (getCrawler, crawlerName) => {
  const waitTimeInSeconds = 30;
  const { Crawler } = await getCrawler(crawlerName);

  if (!Crawler) {
    throw new Error(`Crawler with name ${crawlerName} not found.`);
  }

  if (Crawler.State === "READY") {
    return;
  }

  log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
  await wait(waitTimeInSeconds);
  return waitForCrawler(getCrawler, crawlerName);
};

const makeStartCrawlerStep =
  ({ startCrawler, getCrawler }) =>
  async (context) => {
    log("Starting crawler.");
```

```
    await startCrawler(process.env.CRAWLER_NAME);
    log("Crawler started.", { type: "success" });

    log("Waiting for crawler to finish running. This can take a while.");
    await waitForCrawler(getCrawler, process.env.CRAWLER_NAME);
    log("Crawler ready.", { type: "success" });

    return { ...context };
  };
```

Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};

const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
  };
```

```
/**
 * @param {{ getTables: () => Promise<import('@aws-sdk/client-
glue').GetTablesCommandOutput}} config
 */
const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
  };
```

Crie e execute um trabalho que extraia dados CSV do bucket de origem do Amazon S3, os transforme removendo e renomeando campos e carregue a JSON-formatted saída em outro bucket do Amazon S3.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
```

```
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY,
    );
    log("Job created.", { type: "success" });

    return { ...context };
  };

/**
 * @param {(name: string, runId: string) => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput> }  getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "TIMEOUT":
    case "STOPPED":
    case "ERROR":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`,
      );
```

```
    case "SUCCEEDED":
      return;
    default:
      break;
  }

  log(
    `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...`,
  );
  await wait(waitTimeInSeconds);
  return waitForJobRun(getJobRun, jobName, jobRunId);
};

/**
 * @param {{ prompter: { prompt: () => Promise<{ shouldOpen: boolean }>} }} context
 */
const promptToOpen = async (context) => {
  const { shouldOpen } = await context.prompter.prompt({
    name: "shouldOpen",
    type: "confirm",
    message: "Open the output bucket in your browser?",
  });

  if (shouldOpen) {
    return open(
      `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME} to
 view the output.`,
    );
  }
};

const makeStartJobRunStep =
  ({ startJobRun, getJobRun }) =>
  async (context) => {
    log("Starting job.");
    const { JobRunId } = await startJobRun(
      process.env.JOB_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_NAME,
      process.env.BUCKET_NAME,
    );
    log("Job started.", { type: "success" });

    log("Waiting for job to finish running. This can take a while.");
```

```
    await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
    log("Job run succeeded.", { type: "success" });

    await promptToOpen(context);

    return { ...context };
  };
```

Liste informações sobre execuções de tarefas e visualize alguns dos dados transformados.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};

/**
 * @typedef {{ prompter: { prompt: () => Promise<{jobName: string}> } }} Context
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-glue').GetJobRunCommandOutput>}
 getJobRun
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-glue').GetJobRunsCommandOutput}
 getJobRuns
 */
```

```
/**
 *
 * @param {getJobRun} getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
  const { JobRun } = await getJobRun(jobName, jobRunId);
  log(JobRun, { type: "object" });
};

/**
 *
 * @param {{getJobRuns: getJobRuns, getJobRun: getJobRun }} funcs
 */
const makePickJobRunStep =
  ({ getJobRuns, getJobRun }) =>
  async (/** @type { Context } */ context) => {
    if (context.selectedJobName) {
      const { JobRuns } = await getJobRuns(context.selectedJobName);

      const { jobRunId } = await context.prompter.prompt({
        name: "jobRunId",
        type: "list",
        message: "Select a job run to see details.",
        choices: JobRuns.map((run) => run.Id),
      });

      logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
    }

    return { ...context };
  };
```

Exclua todos os recursos criados pela demonstração.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
```

```
  });

  return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};

const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};

const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};

/**
 *
 * @param {import('../../../actions/delete-job.js').deleteJob} deleteJobFn
 * @param {string[]} jobNames
 * @param {{ prompter: { prompt: () => Promise<any> }}} context
 */
const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
  /**
```

```
    * @type {{ selectedJobNames: string[] }}
    */
  const { selectedJobNames } = await context.prompter.prompt({
    name: "selectedJobNames",
    type: "checkbox",
    message: "Let's clean up jobs. Select jobs to delete.",
    choices: jobNames,
  });

  if (selectedJobNames.length === 0) {
    log("No jobs selected.");
  } else {
    log("Deleting jobs.");
    await Promise.all(
      selectedJobNames.map((n) => deleteJobFn(n).catch(console.error)),
    );
    log("Jobs deleted.", { type: "success" });
  }
};

/**
 * @param {{
 *   listJobs: import('../../../actions/list-jobs.js').listJobs,
 *   deleteJob: import('../../../actions/delete-job.js').deleteJob
 * }} config
 */
const makeCleanUpJobsStep =
  ({ listJobs, deleteJob }) =>
  async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
      await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
  };

/**
 * @param {import('../../../actions/delete-table.js').deleteTable} deleteTable
 * @param {string} databaseName
 * @param {string[]} tableNames
 */
const deleteTables = (deleteTable, databaseName, tableNames) =>
  Promise.all(
```

```
    tableNames.map((tableName) =>
      deleteTable(databaseName, tableName).catch(console.error),
    ),
  );

/**
 * @param {{
 *  getTables: import('../../../actions/get-tables.js').getTables,
 *  deleteTable: import('../../../actions/delete-table.js').deleteTable
 * }} config
 */
const makeCleanUpTablesStep =
  ({ getTables, deleteTable }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}}} context
   */
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
      () => ({ TableList: null }),
    );

    if (TableList && TableList.length > 0) {
      /**
       * @type {{ tableNames: string[] }}
       */
      const { tableNames } = await context.prompter.prompt({
        name: "tableNames",
        type: "checkbox",
        message: "Let's clean up tables. Select tables to delete.",
        choices: TableList.map((t) => t.Name),
      });

      if (tableNames.length === 0) {
        log("No tables selected.");
      } else {
        log("Deleting tables.");
        await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
        log("Tables deleted.", { type: "success" });
      }
    }

    return { ...context };
  };
```

```
/**
 * @param {import('../../../actions/delete-database.js').deleteDatabase}
 deleteDatabase
 * @param {string[]} databaseNames
 */
const deleteDatabases = (deleteDatabase, databaseNames) =>
  Promise.all(
    databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error)),
  );

/**
 * @param {{
 *   getDatabases: import('../../../actions/get-databases.js').getDatabases
 *   deleteDatabase: import('../../../actions/delete-database.js').deleteDatabase
 * }} config
 */
const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}} context
   */
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      /** @type {{ dbNames: string[] }} */
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
        message: "Let's clean up databases. Select databases to delete.",
        choices: DatabaseList.map((db) => db.Name),
      });

      if (dbNames.length === 0) {
        log("No databases selected.");
      } else {
        log("Deleting databases.");
        await deleteDatabases(deleteDatabase, dbNames);
        log("Databases deleted.", { type: "success" });
      }
    }

    return { ...context };
  };
```

```
const cleanUpCrawlerStep = async (context) => {
  log("Deleting crawler.");

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log("Crawler is already deleted.");
    } else {
      throw err;
    }
  }

  return { ...context };
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler

- StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte CreateCrawler em Referência AWS SDK para JavaScript da API.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK para JavaScript da API.

`DeleteCrawler`

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteCrawler em Referência AWS SDK para JavaScript da API.

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteDatabase em Referência AWS SDK para JavaScript da API.

`DeleteJob`

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteJob em Referência AWS SDK para JavaScript da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const deleteTable = (databaseName, tableName) => {
```

```
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para JavaScript da API.

`GetCrawler`

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetCrawler em Referência AWS SDK para JavaScript da API.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetDatabase em Referência AWS SDK para JavaScript da API.

`GetDatabases`

O código de exemplo a seguir mostra como usar `GetDatabases`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getDatabases = () => {
```

```
  const client = new GlueClient({});

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetDatabases em Referência AWS SDK para JavaScript da API.

GetJob

O código de exemplo a seguir mostra como usar `GetJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getJob = (jobName) => {
  const client = new GlueClient({});

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJob em Referência AWS SDK para JavaScript da API.

GetJobRun

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJobRun em Referência AWS SDK para JavaScript da API.

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });
```

```
  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetJobRuns em Referência AWS SDK para JavaScript da API.

GetTables

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetTables em Referência AWS SDK para JavaScript da API.

ListJobs

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const listJobs = () => {
  const client = new GlueClient({});

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para JavaScript da API.

StartCrawler

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });
```

```
  return client.send(command);
};
```

- Para obter detalhes da API, consulte StartCrawler em Referência AWS SDK para JavaScript da API.

StartJobRun

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte StartJobRun em Referência AWS SDK para JavaScript da API.

HealthImaging exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com HealthImaging.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá HealthImaging

O exemplo de código a seguir mostra como começar a usar o HealthImaging.

SDK para JavaScript (v3)

```
import {
  ListDatastoresCommand,
  MedicalImagingClient,
} from "@aws-sdk/client-medical-imaging";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new MedicalImagingClient({});

export const helloMedicalImaging = async () => {
  const command = new ListDatastoresCommand({});

  const { datastoreSummaries } = await client.send(command);
```

```
  console.log("Datastores: ");
  console.log(datastoreSummaries.map((item) => item.datastoreName).join("\n"));
  return datastoreSummaries;
};
```

- Para obter detalhes da API, consulte ListDatastores em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Ações

`CopyImageSet`

O código de exemplo a seguir mostra como usar `CopyImageSet`.

SDK para JavaScript (v3)

Função de utilitário para copiar um conjunto de imagens.

```
import { CopyImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The source image set ID.
 * @param {string} sourceVersionId - The source version ID.
 * @param {string} destinationImageSetId - The optional ID of the destination image
 set.
 * @param {string} destinationVersionId - The optional version ID of the destination
 image set.
 * @param {boolean} force - Force the copy action.
 * @param {[string]} copySubsets - A subset of instance IDs to copy.
 */
export const copyImageSet = async (
  datastoreId = "xxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
  sourceVersionId = "1",
```

```
  destinationImageSetId = "",
  destinationVersionId = "",
  force = false,
  copySubsets = [],
) => {
  try {
    const params = {
      datastoreId: datastoreId,
      sourceImageSetId: imageSetId,
      copyImageSetInformation: {
        sourceImageSet: { latestVersionId: sourceVersionId },
      },
      force: force,
    };
    if (destinationImageSetId !== "" && destinationVersionId !== "") {
      params.copyImageSetInformation.destinationImageSet = {
        imageSetId: destinationImageSetId,
        latestVersionId: destinationVersionId,
      };
    }

    if (copySubsets.length > 0) {
      let copySubsetsJson;
      copySubsetsJson = {
        SchemaVersion: 1.1,
        Study: {
          Series: {
            imageSetId: {
              Instances: {},
            },
          },
        },
      };

      for (let i = 0; i < copySubsets.length; i++) {
        copySubsetsJson.Study.Series.imageSetId.Instances[copySubsets[i]] = {};
      }

      params.copyImageSetInformation.dicomCopies = copySubsetsJson;
    }

    const response = await medicalImagingClient.send(
      new CopyImageSetCommand(params),
    );
```

```
    console.log(response);
    // {
    //     '$metadata': {
    //         httpStatusCode: 200,
    //         requestId: 'd9b219ce-cc48-4a44-a5b2-c5c3068f1ee8',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    //      },
    //       datastoreId: 'xxxxxxxxxxxxxx',
    //       destinationImageSetProperties: {
    //             createdAt: 2023-09-27T19:46:21.824Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxxx:datastore/xxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING',
    //             latestVersionId: '1',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //       },
    //       sourceImageSetProperties: {
    //             createdAt: 2023-09-22T14:49:26.427Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxxx:datastore/xxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING_WITH_READ_ONLY_ACCESS',
    //             latestVersionId: '4',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //      }
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Copiar um conjunto de imagens sem um destino.

```
  await copyImageSet(
```

```
  "12345678901234567890123456789012",
  "12345678901234567890123456789012",
  "1",
);
```

Copiar um conjunto de imagens com um destino.

```
await copyImageSet(
  "12345678901234567890123456789012",
  "12345678901234567890123456789012",
  "1",
  "12345678901234567890123456789012",
  "1",
  false,
);
```

Copie um subconjunto de um conjunto de imagens com um destino e force a cópia.

```
await copyImageSet(
  "12345678901234567890123456789012",
  "12345678901234567890123456789012",
  "1",
  "12345678901234567890123456789012",
  "1",
  true,
  ["12345678901234567890123456789012", "11223344556677889900112233445566"],
);
```

- Para obter detalhes da API, consulte CopyImageSet em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`CreateDatastore`

O código de exemplo a seguir mostra como usar `CreateDatastore`.

SDK para JavaScript (v3)

```
import { CreateDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreName - The name of the data store to create.
 */
export const createDatastore = async (datastoreName = "DATASTORE_NAME") => {
  const response = await medicalImagingClient.send(
    new CreateDatastoreCommand({ datastoreName: datastoreName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: 'a71cd65f-2382-49bf-b682-f9209d8d399b',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //    datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //    datastoreStatus: 'CREATING'
  // }
  return response;
};
```

- Para obter detalhes da API, consulte CreateDatastore em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteDatastore`

O código de exemplo a seguir mostra como usar `DeleteDatastore`.

SDK para JavaScript (v3)

```
import { DeleteDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store to delete.
 */
export const deleteDatastore = async (datastoreId = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new DeleteDatastoreCommand({ datastoreId }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //           httpStatusCode: 200,
  //           requestId: 'f5beb409-678d-48c9-9173-9a001ee1ebb1',
  //           extendedRequestId: undefined,
  //           cfId: undefined,
  //           attempts: 1,
  //           totalRetryDelay: 0
  //        },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     datastoreStatus: 'DELETING'
  // }

  return response;
};
```

- Para obter detalhes da API, consulte DeleteDatastore em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteImageSet`

O código de exemplo a seguir mostra como usar `DeleteImageSet`.

SDK para JavaScript (v3)

```
import { DeleteImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store ID.
 * @param {string} imageSetId - The image set ID.
 */
export const deleteImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new DeleteImageSetCommand({
      datastoreId: datastoreId,
      imageSetId: imageSetId,
    }),
  );
  console.log(response);
  // {
  //    '$metadata': {
  //        httpStatusCode: 200,
  //        requestId: '6267bbd2-eaa5-4a50-8ee8-8fddf535cf73',
  //        extendedRequestId: undefined,
  //        cfId: undefined,
  //        attempts: 1,
  //        totalRetryDelay: 0
  //     },
  //     datastoreId: 'xxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'LOCKED',
  //     imageSetWorkflowStatus: 'DELETING'
```

```
  // }
  return response;
};
```

- Para obter detalhes da API, consulte DeleteImageSet em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDICOMImportJob`

O código de exemplo a seguir mostra como usar `GetDICOMImportJob`.

SDK para JavaScript (v3)

```
import { GetDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} jobId - The ID of the import job.
 */
export const getDICOMImportJob = async (
  datastoreId = "xxxxxxxxxxxxxxxxxxxx",
  jobId = "xxxxxxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new GetDICOMImportJobCommand({ datastoreId: datastoreId, jobId: jobId }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: 'a2637936-78ea-44e7-98b8-7a87d95dfaee',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
```

```
  //         totalRetryDelay: 0
  // },
  //     jobProperties: {
  //         dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-19T17:29:21.753Z,
  //             inputS3Uri: 's3://healthimaging-source/CTStudy/',
  //             jobId: ''xxxxxxxxxxxxxxxxxxxxxxxxx'',
  //             jobName: 'job_1',
  //             jobStatus: 'COMPLETED',
  //             outputS3Uri: 's3://health-imaging-dest/
ouput_ct/'xxxxxxxxxxxxxxxxxxxxxxxxx'-DicomImport-'xxxxxxxxxxxxxxxxxxxxxxxxx'/',
  //             submittedAt: 2023-09-19T17:27:25.143Z
  //     }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte GetDICOMImportJob em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDatastore`

O código de exemplo a seguir mostra como usar `GetDatastore`.

SDK para JavaScript (v3)

```
import { GetDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreID - The ID of the data store.
 */
export const getDatastore = async (datastoreID = "DATASTORE_ID") => {
```

```
  const response = await medicalImagingClient.send(
    new GetDatastoreCommand({ datastoreId: datastoreID }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '55ea7d2e-222c-4a6a-871e-4f591f40cadb',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //   datastoreProperties: {
  //        createdAt: 2023-08-04T18:50:36.239Z,
  //         datastoreArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreName: 'my_datastore',
  //         datastoreStatus: 'ACTIVE',
  //         updatedAt: 2023-08-04T18:50:36.239Z
  //   }
  // }
  return response.datastoreProperties;
};
```

- Para obter detalhes da API, consulte GetDatastore em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageFrame`

O código de exemplo a seguir mostra como usar `GetImageFrame`.

SDK para JavaScript (v3)

```
import { GetImageFrameCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} imageFrameFileName - The name of the file for the HTJ2K-encoded
 image frame.
 * @param {string} datastoreID - The data store's ID.
 * @param {string} imageSetID - The image set's ID.
 * @param {string} imageFrameID - The image frame's ID.
 */
export const getImageFrame = async (
  imageFrameFileName = "image.jph",
  datastoreID = "DATASTORE_ID",
  imageSetID = "IMAGE_SET_ID",
  imageFrameID = "IMAGE_FRAME_ID",
) => {
  const response = await medicalImagingClient.send(
    new GetImageFrameCommand({
      datastoreId: datastoreID,
      imageSetId: imageSetID,
      imageFrameInformation: { imageFrameId: imageFrameID },
    }),
  );
  const buffer = await response.imageFrameBlob.transformToByteArray();
  writeFileSync(imageFrameFileName, buffer);

  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'e4ab42a5-25a3-4377-873f-374ecf4380e1',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     contentType: 'application/octet-stream',
  //     imageFrameBlob: <ref *1> IncomingMessage {}
  // }
  return response;
};
```

- Para obter detalhes da API, consulte GetImageFrame em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSet`

O código de exemplo a seguir mostra como usar `GetImageSet`.

SDK para JavaScript (v3)

```
import { GetImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 * @param {string} imageSetVersion - The optional version of the image set.
 *
 */
export const getImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxx",
  imageSetVersion = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imageSetId };
  if (imageSetVersion !== "") {
    params.imageSetVersion = imageSetVersion;
  }
  const response = await medicalImagingClient.send(
    new GetImageSetCommand(params),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
```

```
  //         requestId: '0615c161-410d-4d06-9d8c-6e1241bb0a5a',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     createdAt: 2023-09-22T14:49:26.427Z,
  //     datastoreId: 'xxxxxxxxxxxxxxx',
  //     imageSetArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'ACTIVE',
  //     imageSetWorkflowStatus: 'CREATED',
  //     updatedAt: 2023-09-22T14:49:26.427Z,
  //     versionId: '1'
  // }

  return response;
};
```

- Para obter detalhes da API, consulte GetImageSet em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSetMetadata`

O código de exemplo a seguir mostra como usar `GetImageSetMetadata`.

SDK para JavaScript (v3)

Função de utilitário para obter metadados do conjunto de imagens.

```
import { GetImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
import { writeFileSync } from "node:fs";
```

```
/**
 * @param {string} metadataFileName - The name of the file for the gzipped metadata.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imagesetId - The ID of the image set.
 * @param {string} versionID - The optional version ID of the image set.
 */
export const getImageSetMetadata = async (
  metadataFileName = "metadata.json.gzip",
  datastoreId = "xxxxxxxxxxxxxx",
  imagesetId = "xxxxxxxxxxxxxx",
  versionID = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imagesetId };

  if (versionID) {
    params.versionID = versionID;
  }

  const response = await medicalImagingClient.send(
    new GetImageSetMetadataCommand(params),
  );
  const buffer = await response.imageSetMetadataBlob.transformToByteArray();
  writeFileSync(metadataFileName, buffer);

  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '5219b274-30ff-4986-8cab-48753de3a599',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     contentType: 'application/json',
  //     contentEncoding: 'gzip',
  //     imageSetMetadataBlob: <ref *1> IncomingMessage {}
  // }

  return response;
};
```

Obter metadados do conjunto de imagens sem versão.

```
try {
  await getImageSetMetadata(
    "metadata.json.gzip",
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
  );
} catch (err) {
  console.log("Error", err);
}
```

Obter metadados do conjunto de imagens com versão.

```
try {
  await getImageSetMetadata(
    "metadata2.json.gzip",
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
  );
} catch (err) {
  console.log("Error", err);
}
```

- Para obter detalhes da API, consulte GetImageSetMetadata em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDICOMImportJobs`

O código de exemplo a seguir mostra como usar `ListDICOMImportJobs`.

SDK para JavaScript (v3)

```
import { paginateListDICOMImportJobs } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 */
export const listDICOMImportJobs = async (
  datastoreId = "xxxxxxxxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId: datastoreId };
  const paginator = paginateListDICOMImportJobs(paginatorConfig, commandParams);

  const jobSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    jobSummaries.push(...page.jobSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '3c20c66e-0797-446a-a1d8-91b742fd15a0',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobSummaries: [
  //         {
  //             dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-22T14:49:51.351Z,
  //             jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             jobName: 'test-1',
  //             jobStatus: 'COMPLETED',
  //             submittedAt: 2023-09-22T14:48:45.767Z
```

```
  // }
  // ]}

  return jobSummaries;
};
```

- Para obter detalhes da API, consulte ListDICOMImportJobs em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDatastores`

O código de exemplo a seguir mostra como usar `ListDatastores`.

SDK para JavaScript (v3)

```
import { paginateListDatastores } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

export const listDatastores = async () => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {};
  const paginator = paginateListDatastores(paginatorConfig, commandParams);

  /**
   * @type {import("@aws-sdk/client-medical-imaging").DatastoreSummary[]}
   */
  const datastoreSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    datastoreSummaries.push(...page.datastoreSummaries);
```

```
    console.log(page);
  }
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '6aa99231-d9c2-4716-a46e-edb830116fa3',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //   },
  //   datastoreSummaries: [
  //     {
  //       createdAt: 2023-08-04T18:49:54.429Z,
  //       datastoreArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreName: 'my_datastore',
  //       datastoreStatus: 'ACTIVE',
  //       updatedAt: 2023-08-04T18:49:54.429Z
  //     }
  //     ...
  //   ]
  // }

  return datastoreSummaries;
};
```

- Para obter detalhes da API, consulte ListDatastores em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListImageSetVersions`

O código de exemplo a seguir mostra como usar `ListImageSetVersions`.

SDK para JavaScript (v3)

```
import { paginateListImageSetVersions } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 */
export const listImageSetVersions = async (
  datastoreId = "xxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId, imageSetId };
  const paginator = paginateListImageSetVersions(
    paginatorConfig,
    commandParams,
  );

  const imageSetPropertiesList = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetPropertiesList.push(...page.imageSetPropertiesList);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '74590b37-a002-4827-83f2-3c590279c742',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetPropertiesList: [
  //         {
  //             ImageSetWorkflowStatus: 'CREATED',
  //             createdAt: 2023-09-22T14:49:26.427Z,
```

```
  //             imageSetId: 'xxxxxxxxxxxxxxxxxxxxxx',
  //             imageSetState: 'ACTIVE',
  //             versionId: '1'
  //         }]
  // }
  return imageSetPropertiesList;
};
```

- Para obter detalhes da API, consulte ListImageSetVersions em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListTagsForResource`

O código de exemplo a seguir mostra como usar `ListTagsForResource`.

SDK para JavaScript (v3)

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
```

```
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte ListTagsForResource em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`SearchImageSets`

O código de exemplo a seguir mostra como usar `SearchImageSets`.

SDK para JavaScript (v3)

A função de utilitário para pesquisar conjuntos de imagens.

```
import { paginateSearchImageSets } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store's ID.
 * @param { import('@aws-sdk/client-medical-imaging').SearchFilter[] } filters - The
 search criteria filters.
 * @param { import('@aws-sdk/client-medical-imaging').Sort } sort - The search
 criteria sort.
 */
export const searchImageSets = async (
  datastoreId = "xxxxxxxx",
  searchCriteria = {},
) => {
```

```
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {
    datastoreId: datastoreId,
    searchCriteria: searchCriteria,
  };

  const paginator = paginateSearchImageSets(paginatorConfig, commandParams);

  const imageSetsMetadataSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetsMetadataSummaries.push(...page.imageSetsMetadataSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'f009ea9c-84ca-4749-b5b6-7164f00a5ada',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetsMetadataSummaries: [
  //         {
  //             DICOMTags: [Object],
  //             createdAt: "2023-09-19T16:59:40.551Z",
  //             imageSetId: '7f75e1b5c0f40eac2b24cf712f485f50',
  //             updatedAt: "2023-09-19T16:59:40.551Z",
  //             version: 1
  //         }]
  // }

  return imageSetsMetadataSummaries;
};
```

Caso de uso nº 1: operador EQUAL.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [{ DICOMPatientId: "1234567" }],
        operator: "EQUAL",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Caso de uso #2: operador BETWEEN usando DICOMStudyDate DICOMStudyTime e.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "19900101",
              DICOMStudyTime: "000000",
            },
          },
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "20230901",
              DICOMStudyTime: "000000",
            },
          },
        ],
        operator: "BETWEEN",
      },
    ],
```

```
    };

    await searchImageSets(datastoreId, searchCriteria);
  } catch (err) {
    console.error(err);
  }
```

Caso de uso nº 3: operador BETWEEN usando o createdAt. Os estudos de tempo foram previamente persistidos.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { createdAt: new Date("1985-04-12T23:20:50.52Z") },
          { createdAt: new Date() },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Caso de uso #4: operador EQUAL on DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifique a resposta na ordem ASC no campo updatedAt.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { updatedAt: new Date("1985-04-12T23:20:50.52Z") },
```

```
            { updatedAt: new Date() },
          ],
          operator: "BETWEEN",
        },
        {
          values: [
            {
              DICOMSeriesInstanceUID:
                "1.1.123.123456.1.12.1.1234567890.1234.12345678.123",
            },
          ],
          operator: "EQUAL",
        },
      ],
      sort: {
        sortOrder: "ASC",
        sortField: "updatedAt",
      },
    };

    await searchImageSets(datastoreId, searchCriteria);
  } catch (err) {
    console.error(err);
  }
```

- Para obter detalhes da API, consulte SearchImageSets em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`StartDICOMImportJob`

O código de exemplo a seguir mostra como usar `StartDICOMImportJob`.

SDK para JavaScript (v3)

```
import { StartDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
```

```
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} jobName - The name of the import job.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} dataAccessRoleArn - The Amazon Resource Name (ARN) of the role
 that grants permission.
 * @param {string} inputS3Uri - The URI of the S3 bucket containing the input files.
 * @param {string} outputS3Uri - The URI of the S3 bucket where the output files are
 stored.
 */
export const startDicomImportJob = async (
  jobName = "test-1",
  datastoreId = "12345678901234567890123456789012",
  dataAccessRoleArn = "arn:aws:iam::xxxxxxxxxxxx:role/ImportJobDataAccessRole",
  inputS3Uri = "s3://medical-imaging-dicom-input/dicom_input/",
  outputS3Uri = "s3://medical-imaging-output/job_output/",
) => {
  const response = await medicalImagingClient.send(
    new StartDICOMImportJobCommand({
      jobName: jobName,
      datastoreId: datastoreId,
      dataAccessRoleArn: dataAccessRoleArn,
      inputS3Uri: inputS3Uri,
      outputS3Uri: outputS3Uri,
    }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '6e81d191-d46b-4e48-a08a-cdcc7e11eb79',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobStatus: 'SUBMITTED',
  //     submittedAt: 2023-09-22T14:48:45.767Z
  // }
  return response;
};
```

- Para obter detalhes da API, consulte StartDICOMImportJob em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para JavaScript (v3)

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
```

```
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte TagResource em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`UntagResource`

O código de exemplo a seguir mostra como usar `UntagResource`.

SDK para JavaScript (v3)

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
```

```
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte UntagResource em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`UpdateImageSetMetadata`

O código de exemplo a seguir mostra como usar `UpdateImageSetMetadata`.

SDK para JavaScript (v3)

```
import { UpdateImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the HealthImaging data store.
 * @param {string} imageSetId - The ID of the HealthImaging image set.
 * @param {string} latestVersionId - The ID of the HealthImaging image set version.
 * @param {{}} updateMetadata - The metadata to update.
 * @param {boolean} force - Force the update.
 */
export const updateImageSetMetadata = async (
  datastoreId = "xxxxxxxxxx",
  imageSetId = "xxxxxxxxxx",
  latestVersionId = "1",
```

```
  updateMetadata = "{}",
  force = false,
) => {
  try {
    const response = await medicalImagingClient.send(
      new UpdateImageSetMetadataCommand({
        datastoreId: datastoreId,
        imageSetId: imageSetId,
        latestVersionId: latestVersionId,
        updateImageSetMetadataUpdates: updateMetadata,
        force: force,
      }),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //     httpStatusCode: 200,
    //         requestId: '7966e869-e311-4bff-92ec-56a61d3003ea',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    // },
    //     createdAt: 2023-09-22T14:49:26.427Z,
    //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetState: 'LOCKED',
    //     imageSetWorkflowStatus: 'UPDATING',
    //     latestVersionId: '4',
    //     updatedAt: 2023-09-27T19:41:43.494Z
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Caso de uso 1: insira ou atualize um atributo e force a atualização.

```
    const insertAttributes = JSON.stringify({
      SchemaVersion: 1.1,
      Study: {
```

```
      DICOM: {
        StudyDescription: "CT CHEST",
      },
    },
  });

  const updateMetadata = {
    DICOMUpdates: {
      updatableAttributes: new TextEncoder().encode(insertAttributes),
    },
  };

  await updateImageSetMetadata(
    datastoreID,
    imageSetID,
    versionID,
    updateMetadata,
    true,
  );
```

Caso de uso 2: remova um atributo.

```
  // Attribute key and value must match the existing attribute.
  const remove_attribute = JSON.stringify({
    SchemaVersion: 1.1,
    Study: {
      DICOM: {
        StudyDescription: "CT CHEST",
      },
    },
  });

  const updateMetadata = {
    DICOMUpdates: {
      removableAttributes: new TextEncoder().encode(remove_attribute),
    },
  };

  await updateImageSetMetadata(
    datastoreID,
    imageSetID,
    versionID,
```

```
  updateMetadata,
);
```

Caso de uso 3: remova uma instância.

```
const remove_instance = JSON.stringify({
  SchemaVersion: 1.1,
  Study: {
    Series: {
      "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
        Instances: {
          "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {},
        },
      },
    },
  },
});

const updateMetadata = {
  DICOMUpdates: {
    removableAttributes: new TextEncoder().encode(remove_instance),
  },
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

Caso de uso 4: reverta para uma versão anterior.

```
const updateMetadata = {
  revertToVersionId: "1",
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
```

```
    updateMetadata,
  );
```

- Para obter detalhes da API, consulte UpdateImageSetMetadata em Referência AWS SDK para JavaScript da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cenários

Começar a usar conjuntos de imagens e quadros de imagem

O exemplo de código a seguir mostra como importar arquivos DICOM e baixar quadros de imagem em HealthImaging.

A implementação é estruturada como uma aplicação da linha de comando.

- Configure recursos para uma importação DICOM.
- Importe arquivos DICOM para um armazenamento de dados.
- Recupere os IDs do conjunto de imagens para o trabalho de importação.
- Recupere os IDs do quadro de imagem para o conjunto de imagens.
- Baixe, decodifique e verifique os quadros de imagem.
- Limpe recursos.

SDK para JavaScript (v3)

Orquestre etapas (index.js).

```
import {
  parseScenarioArgs,
  Scenario,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
```

```
  saveState,
  loadState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import {
  createStack,
  deployStack,
  getAccountId,
  getDatastoreName,
  getStackName,
  outputState,
  waitForStackCreation,
} from "./deploy-steps.js";
import {
  doCopy,
  selectDataset,
  copyDataset,
  outputCopiedObjects,
} from "./dataset-steps.js";
import {
  doImport,
  outputImportJobStatus,
  startDICOMImport,
  waitForImportJobCompletion,
} from "./import-steps.js";
import {
  getManifestFile,
  outputImageSetIds,
  parseManifestFile,
} from "./image-set-steps.js";
import {
  getImageSetMetadata,
  outputImageFrameIds,
} from "./image-frame-steps.js";
import { decodeAndVerifyImages, doVerify } from "./verify-steps.js";
import {
  confirmCleanup,
  deleteImageSets,
  deleteStack,
} from "./clean-up-steps.js";

const context = {};

const scenarios = {
```

```
  deploy: new Scenario(
    "Deploy Resources",
    [
      deployStack,
      getStackName,
      getDatastoreName,
      getAccountId,
      createStack,
      waitForStackCreation,
      outputState,
      saveState,
    ],
    context,
  ),
  demo: new Scenario(
    "Run Demo",
    [
      loadState,
      doCopy,
      selectDataset,
      copyDataset,
      outputCopiedObjects,
      doImport,
      startDICOMImport,
      waitForImportJobCompletion,
      outputImportJobStatus,
      getManifestFile,
      parseManifestFile,
      outputImageSetIds,
      getImageSetMetadata,
      outputImageFrameIds,
      doVerify,
      decodeAndVerifyImages,
      saveState,
    ],
    context,
  ),
  destroy: new Scenario(
    "Clean Up Resources",
    [loadState, confirmCleanup, deleteImageSets, deleteStack],
    context,
  ),
};
```

```
// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Health Imaging Workflow",
    description:
      "Work with DICOM images using an AWS Health Imaging data store.",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Implante recursos (deploy-steps.js).

```
import fs from "node:fs/promises";
import path from "node:path";

import {
  CloudFormationClient,
  CreateStackCommand,
  DescribeStacksCommand,
} from "@aws-sdk/client-cloudformation";
import { STSClient, GetCallerIdentityCommand } from "@aws-sdk/client-sts";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const cfnClient = new CloudFormationClient({});
const stsClient = new STSClient({});

const __dirname = path.dirname(new URL(import.meta.url).pathname);
const cfnTemplatePath = path.join(
  __dirname,
  "../../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml",
);
```

```
export const deployStack = new ScenarioInput(
  "deployStack",
  "Do you want to deploy the CloudFormation stack?",
  { type: "confirm" },
);

export const getStackName = new ScenarioInput(
  "getStackName",
  "Enter a name for the CloudFormation stack:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getDatastoreName = new ScenarioInput(
  "getDatastoreName",
  "Enter a name for the HealthImaging datastore:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getAccountId = new ScenarioAction(
  "getAccountId",
  async (/** @type {{}} */ state) => {
    const command = new GetCallerIdentityCommand({});
    const response = await stsClient.send(command);
    state.accountId = response.Account;
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const createStack = new ScenarioAction(
  "createStack",
  async (/** @type {{}} */ state) => {
    const stackName = state.getStackName;
    const datastoreName = state.getDatastoreName;
    const accountId = state.accountId;

    const command = new CreateStackCommand({
      StackName: stackName,
      TemplateBody: await fs.readFile(cfnTemplatePath, "utf8"),
      Capabilities: ["CAPABILITY_IAM"],
      Parameters: [
        {
          ParameterKey: "datastoreName",
```

```
          ParameterValue: datastoreName,
        },
        {
          ParameterKey: "userAccountID",
          ParameterValue: accountId,
        },
      ],
    });

    const response = await cfnClient.send(command);
    state.stackId = response.StackId;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const waitForStackCreation = new ScenarioAction(
  "waitForStackCreation",
  async (/** @type {{}} */ state) => {
    const command = new DescribeStacksCommand({
      StackName: state.stackId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await cfnClient.send(command);
      const stack = response.Stacks?.find(
        (s) => s.StackName === state.getStackName,
      );
      if (!stack || stack.StackStatus === "CREATE_IN_PROGRESS") {
        throw new Error("Stack creation is still in progress");
      }
      if (stack.StackStatus === "CREATE_COMPLETE") {
        state.stackOutputs = stack.Outputs?.reduce((acc, output) => {
          acc[output.OutputKey] = output.OutputValue;
          return acc;
        }, {});
      } else {
        throw new Error(
          `Stack creation failed with status: ${stack.StackStatus}`,
        );
      }
    });
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
```

```
  },
);

export const outputState = new ScenarioOutput(
  "outputState",
  (/** @type {{}} */ state) => {
    /**
     * @type {{ stackOutputs: { DatastoreID: string, BucketName: string, RoleArn:
 string }}}
     */
    const { stackOutputs } = state;
    return `Stack creation completed. Output values:
Datastore ID: ${stackOutputs?.DatastoreID}
Bucket Name: ${stackOutputs?.BucketName}
Role ARN: ${stackOutputs?.RoleArn}
    `;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);
```

Copie arquivos DICOM (dataset-steps.js).

```
import {
  S3Client,
  CopyObjectCommand,
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const s3Client = new S3Client({});

const datasetOptions = [
  {
    name: "CT of chest (2 images)",
    value: "00029d25-fb18-4d42-aaa5-a0897d1ac8f7",
  },
  {
```

```
    name: "CT of pelvis (57 images)",
    value: "00025d30-ef8f-4135-a35a-d83eff264fc1",
  },
  {
    name: "MRI of head (192 images)",
    value: "0002d261-8a5d-4e63-8e2e-0cbfac87b904",
  },
  {
    name: "MRI of breast (92 images)",
    value: "0002dd07-0b7f-4a68-a655-44461ca34096",
  },
];

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   doCopy: boolean
 * }}} State
 */

export const selectDataset = new ScenarioInput(
  "selectDataset",
  (state) => {
    if (!state.doCopy) {
      process.exit(0);
    }
    return "Select a DICOM dataset to import:";
  },
  {
    type: "select",
    choices: datasetOptions,
  },
);

export const doCopy = new ScenarioInput(
  "doCopy",
  "Do you want to copy images from the public dataset into your bucket?",
  {
    type: "confirm",
  },
);

export const copyDataset = new ScenarioAction(
```

```
  "copyDataset",
  async (/** @type { State } */ state) => {
    const inputBucket = state.stackOutputs.BucketName;
    const inputPrefix = "input/";
    const selectedDatasetId = state.selectDataset;

    const sourceBucket = "idc-open-data";
    const sourcePrefix = `${selectedDatasetId}`;

    const listObjectsCommand = new ListObjectsV2Command({
      Bucket: sourceBucket,
      Prefix: sourcePrefix,
    });

    const objects = await s3Client.send(listObjectsCommand);

    const copyPromises = objects.Contents.map((object) => {
      const sourceKey = object.Key;
      const destinationKey = `${inputPrefix}${sourceKey
        .split("/")
        .slice(1)
        .join("/")}`;

      const copyCommand = new CopyObjectCommand({
        Bucket: inputBucket,
        CopySource: `/${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });

      return s3Client.send(copyCommand);
    });

    const results = await Promise.all(copyPromises);
    state.copiedObjects = results.length;
  },
);

export const outputCopiedObjects = new ScenarioOutput(
  "outputCopiedObjects",
  (state) => `${state.copiedObjects} DICOM files were copied.`,
);
```

Inicie a importação para o datastore (import-steps.js).

```
import {
  MedicalImagingClient,
  StartDICOMImportJobCommand,
  GetDICOMImportJobCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioOutput,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }}} State
 */

export const doImport = new ScenarioInput(
  "doImport",
  "Do you want to import DICOM images into your datastore?",
  {
    type: "confirm",
    default: true,
  },
);

export const startDICOMImport = new ScenarioAction(
  "startDICOMImport",
  async (/** @type {State} */ state) => {
    if (!state.doImport) {
      process.exit(0);
    }
    const medicalImagingClient = new MedicalImagingClient({});
    const inputS3Uri = `s3://${state.stackOutputs.BucketName}/input/`;
    const outputS3Uri = `s3://${state.stackOutputs.BucketName}/output/`;

    const command = new StartDICOMImportJobCommand({
      dataAccessRoleArn: state.stackOutputs.RoleArn,
```

```
      datastoreId: state.stackOutputs.DatastoreID,
      inputS3Uri,
      outputS3Uri,
    });

    const response = await medicalImagingClient.send(command);
    state.importJobId = response.jobId;
  },
);

export const waitForImportJobCompletion = new ScenarioAction(
  "waitForImportJobCompletion",
  async (/** @type {State} */ state) => {
    const medicalImagingClient = new MedicalImagingClient({});
    const command = new GetDICOMImportJobCommand({
      datastoreId: state.stackOutputs.DatastoreID,
      jobId: state.importJobId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await medicalImagingClient.send(command);
      const jobStatus = response.jobProperties?.jobStatus;
      if (!jobStatus || jobStatus === "IN_PROGRESS") {
        throw new Error("Import job is still in progress");
      }
      if (jobStatus === "COMPLETED") {
        state.importJobOutputS3Uri = response.jobProperties.outputS3Uri;
      } else {
        throw new Error(`Import job failed with status: ${jobStatus}`);
      }
    });
  },
);

export const outputImportJobStatus = new ScenarioOutput(
  "outputImportJobStatus",
  (state) =>
    `DICOM import job completed. Output location: ${state.importJobOutputS3Uri}`,
);
```

Obtenha IDs do conjunto de imagens (image-set-steps.js -).

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, importJobId: string,
 * importJobOutputS3Uri: string,
 * imageSetIds: string[],
 * manifestContent: { jobSummary: { imageSetsSummary: { imageSetId: string }[] } }
 * }} State
 */

const s3Client = new S3Client({});

export const getManifestFile = new ScenarioAction(
  "getManifestFile",
  async (/** @type {State} */ state) => {
    const bucket = state.stackOutputs.BucketName;
    const prefix = `output/${state.stackOutputs.DatastoreID}-DicomImport-
${state.importJobId}/`;
    const key = `${prefix}job-output-manifest.json`;

    const command = new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    });

    const response = await s3Client.send(command);
    const manifestContent = await response.Body.transformToString();
    state.manifestContent = JSON.parse(manifestContent);
  },
);

export const parseManifestFile = new ScenarioAction(
  "parseManifestFile",
  (/** @type {State} */ state) => {
```

```
    const imageSetIds =
      state.manifestContent.jobSummary.imageSetsSummary.reduce((ids, next) => {
        return Object.assign({}, ids, {
          [next.imageSetId]: next.imageSetId,
        });
      }, {});
    state.imageSetIds = Object.keys(imageSetIds);
  },
);

export const outputImageSetIds = new ScenarioOutput(
  "outputImageSetIds",
  (/** @type {State} */ state) =>
    `The image sets created by this import job are: \n${state.imageSetIds
      .map((id) => `Image set: ${id}`)
      .join("\n")}`,
);
```

Obtenha IDs do quadro de imagens (image-frame-steps.js).

```
import {
  MedicalImagingClient,
  GetImageSetMetadataCommand,
} from "@aws-sdk/client-medical-imaging";
import { gunzip } from "node:zlib";
import { promisify } from "node:util";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const gunzipAsync = promisify(gunzip);

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
```

```
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
```

```
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetIds: string[] }} State
 */

const medicalImagingClient = new MedicalImagingClient({});

export const getImageSetMetadata = new ScenarioAction(
  "getImageSetMetadata",
  async (/** @type {State} */ state) => {
    const outputMetadata = [];

    for (const imageSetId of state.imageSetIds) {
      const command = new GetImageSetMetadataCommand({
        datastoreId: state.stackOutputs.DatastoreID,
        imageSetId,
      });

      const response = await medicalImagingClient.send(command);
      const compressedMetadataBlob =
        await response.imageSetMetadataBlob.transformToByteArray();
      const decompressedMetadata = await gunzipAsync(compressedMetadataBlob);
      const imageSetMetadata = JSON.parse(decompressedMetadata.toString());

      outputMetadata.push(imageSetMetadata);
    }

    state.imageSetMetadata = outputMetadata;
  },
);

export const outputImageFrameIds = new ScenarioOutput(
  "outputImageFrameIds",
  (/** @type {State & { imageSetMetadata: ImageSetMetadata[] }} */ state) => {
    let output = "";

    for (const metadata of state.imageSetMetadata) {
      const imageSetId = metadata.ImageSetID;
      /** @type {DICOMMetadata[]} */
      const instances = Object.values(metadata.Study.Series).flatMap(
        (series) => {
          return Object.values(series.Instances);
        },
```

```
      );
      const imageFrameIds = instances.flatMap((instance) =>
        instance.ImageFrames.map((frame) => frame.ID),
      );

      output += `Image set ID: ${imageSetId}\nImage frame IDs:\n
${imageFrameIds.join(
        "\n",
      )}\n\n`;
    }

    return output;
  },
);
```

Verifique os quadros de imagens (verify-steps.js). A biblioteca AWS HealthImaging Pixel Data Verification foi usada para verificação.

```
import { spawn } from "node:child_process";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */
```

```
/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

export const doVerify = new ScenarioInput(
  "doVerify",
  "Do you want to verify the imported images?",
```

```
  {
    type: "confirm",
    default: true,
  },
);

export const decodeAndVerifyImages = new ScenarioAction(
  "decodeAndVerifyImages",
  async (/** @type {State} */ state) => {
    if (!state.doVerify) {
      process.exit(0);
    }
    const verificationTool = "./pixel-data-verification/index.js";

    for (const metadata of state.imageSetMetadata) {
      const datastoreId = state.stackOutputs.DatastoreID;
      const imageSetId = metadata.ImageSetID;

      for (const [seriesInstanceUid, series] of Object.entries(
        metadata.Study.Series,
      )) {
        for (const [sopInstanceUid, _] of Object.entries(series.Instances)) {
          console.log(
            `Verifying image set ${imageSetId} with series ${seriesInstanceUid} and
 sop ${sopInstanceUid}`,
          );
          const child = spawn(
            "node",
            [
              verificationTool,
              datastoreId,
              imageSetId,
              seriesInstanceUid,
              sopInstanceUid,
            ],
            { stdio: "inherit" },
          );

          await new Promise((resolve, reject) => {
            child.on("exit", (code) => {
              if (code === 0) {
                resolve();
              } else {
                reject(
```

```
                  new Error(
                    `Verification tool exited with code ${code} for image set
 ${imageSetId}`,
                  ),
                );
              }
            });
          });
        }
      }
    }
  },
);
```

Destrua recursos (clean-up-steps.js).

```
import {
  CloudFormationClient,
  DeleteStackCommand,
} from "@aws-sdk/client-cloudformation";
import {
  MedicalImagingClient,
  DeleteImageSetCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
```

```
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
```

```
 */

const cfnClient = new CloudFormationClient({});
const medicalImagingClient = new MedicalImagingClient({});

export const confirmCleanup = new ScenarioInput(
  "confirmCleanup",
  "Do you want to delete the created resources?",
  { type: "confirm" },
);

export const deleteImageSets = new ScenarioAction(
  "deleteImageSets",
  async (/** @type {State} */ state) => {
    const datastoreId = state.stackOutputs.DatastoreID;

    for (const metadata of state.imageSetMetadata) {
      const command = new DeleteImageSetCommand({
        datastoreId,
        imageSetId: metadata.ImageSetID,
      });

      try {
        await medicalImagingClient.send(command);
        console.log(`Successfully deleted image set ${metadata.ImageSetID}`);
      } catch (e) {
        if (e instanceof Error) {
          if (e.name === "ConflictException") {
            console.log(`Image set ${metadata.ImageSetID} already deleted`);
          }
        }
      }
    }
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);

export const deleteStack = new ScenarioAction(
  "deleteStack",
  async (/** @type {State} */ state) => {
    const stackName = state.getStackName;
```

```
    const command = new DeleteStackCommand({
      StackName: stackName,
    });

    await cfnClient.send(command);
    console.log(`Stack ${stackName} deletion initiated`);
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Marcar um datastore

O exemplo de código a seguir mostra como marcar um armazenamento HealthImaging de dados.

SDK para JavaScript (v3)

Marcar um datastore.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
```

```
    const tags = {
      Deployment: "Development",
    };
    await tagResource(datastoreArn, tags);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para marcar um recurso.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

Listar tags para um datastore.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(datastoreArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para listar as tags de um recurso.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

Desmarcar um datastore.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(datastoreArn, keys);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para desmarcar um recurso.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
```

```
  // }

  return response;
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Marcar um conjunto de imagens

O exemplo de código a seguir mostra como marcar um conjunto de HealthImaging imagens.

SDK para JavaScript (v3)

Marcar um conjunto de imagens

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(imagesetArn, tags);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para marcar um recurso.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

Listar tags para um conjunto de imagens

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(imagesetArn);
    console.log(tags);
  } catch (e) {
```

```
    console.log(e);
  }
```

A função de utilitário para listar as tags de um recurso.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

Desmarcar um conjunto de imagens

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
    const keys = ["Deployment"];
    await untagResource(imagesetArn, keys);
  } catch (e) {
    console.log(e);
  }
```

A função de utilitário para desmarcar um recurso.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .

- ListTagsForResource
- TagResource
- UntagResource

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exemplos de IAM usando SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
  /**
   * In v3, the clients expose paginateOperationName APIs that are written using
 async generators so that you can use async iterators in a for await..of loop.
   * https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
   */
  const paginator = paginateListPolicies(
    { client, pageSize: 10 },
    // List only customer managed policies.
    { Scope: "Local" },
  );

  console.log("IAM policies defined in your account:");
  let policyCount = 0;
  for await (const page of paginator) {
    if (page.Policies) {
      for (const policy of page.Policies) {
        console.log(`${policy.PolicyName}`);
        policyCount++;
      }
    }
  }
  console.log(`Found ${policyCount} policies.`);
};
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário e um perfil do IAM que conceda permissão para listar os buckets do Amazon S3. O usuário só tem direitos para assumir a função. Após assumir a função, use credenciais temporárias para listar os buckets para a conta.

```
import {
  CreateUserCommand,
  GetUserCommand,
```

```
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "iam_basic_test_username";
const policyName = "iam_basic_test_policy";
const roleName = "iam_basic_test_role";

/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
```

```
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    }
    throw caught;
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }

  // Create an access key. This key is used to authenticate the new user to
  // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service (AWS
 STS).
  // It's not best practice to use access keys. For more information, see https://
aws.amazon.com/iam/resources/best-practices/.
  const createAccessKeyResponse = await iamClient.send(
    new CreateAccessKeyCommand({ UserName: userName }),
  );

  if (
    !createAccessKeyResponse.AccessKey?.AccessKeyId ||
    !createAccessKeyResponse.AccessKey?.SecretAccessKey
  ) {
    throw new Error("Access key not created");
  }

  const {
    AccessKey: { AccessKeyId, SecretAccessKey },
  } = createAccessKeyResponse;
```

```
  let s3Client = new S3Client({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
  // thrown while the user and access keys are still stabilizing.
  await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
    try {
      return await listBuckets(s3Client);
    } catch (err) {
      if (err instanceof Error && err.name === "InvalidAccessKeyId") {
        throw err;
      }
    }
  });

  // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
  // is thrown while the user and access keys are still stabilizing.
  const { Role } = await retry(
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
                  // Allow the previously created user to assume this role.
                  AWS: User.Arn,
                },
                Action: "sts:AssumeRole",
              },
            ],
          }),
```

```
        RoleName: roleName,
      }),
    ),
);

if (!Role) {
  throw new Error("Role not created");
}

// Create a policy that allows the user to list S3 buckets.
const { Policy: listBucketPolicy } = await iamClient.send(
  new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: ["s3:ListAllMyBuckets"],
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  }),
);

if (!listBucketPolicy) {
  throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
  new AttachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

// Assume the role.
const stsClient = new STSClient({
  credentials: {
    accessKeyId: AccessKeyId,
    secretAccessKey: SecretAccessKey,
  },
```

```
});

// Retry the assume role operation until it succeeds.
const { Credentials } = await retry(
  { intervalInMs: 2000, maxRetries: 60 },
  () =>
    stsClient.send(
      new AssumeRoleCommand({
        RoleArn: Role.Arn,
        RoleSessionName: `iamBasicScenarioSession-${Math.floor(
          Math.random() * 1000000,
        )}`,
        DurationSeconds: 900,
      }),
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 120 }, () =>
  listBuckets(s3Client),
);

// Clean up.
await iamClient.send(
  new DetachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);
```

```
  await iamClient.send(
    new DeletePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
    }),
  );

  await iamClient.send(
    new DeleteRoleCommand({
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeleteAccessKeyCommand({
      UserName: userName,
      AccessKeyId,
    }),
  );

  await iamClient.send(
    new DeleteUserCommand({
      UserName: userName,
    }),
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Anexe a política.

```
import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
```

```
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte AttachRolePolicy em Referência AWS SDK para JavaScript da API.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a chave de acesso.

```
import { CreateAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 */
export const createAccessKey = (userName) => {
```

```
  const command = new CreateAccessKeyCommand({ UserName: userName });
  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateAccessKey em Referência AWS SDK para JavaScript da API.

`CreateAccountAlias`

O código de exemplo a seguir mostra como usar `CreateAccountAlias`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar o alias da conta.

```
import { CreateAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias - A unique name for the account alias.
 * @returns
 */
export const createAccountAlias = (alias) => {
  const command = new CreateAccountAliasCommand({
    AccountAlias: alias,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateAccountAlias em Referência AWS SDK para JavaScript da API.

`CreateGroup`

O código de exemplo a seguir mostra como usar `CreateGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const createGroup = async (groupName) => {
  const command = new CreateGroupCommand({ GroupName: groupName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte CreateGroup em Referência AWS SDK para JavaScript da API.

`CreateInstanceProfile`

O código de exemplo a seguir mostra como usar `CreateInstanceProfile`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
```

- Para obter detalhes da API, consulte CreateInstanceProfile em Referência AWS SDK para JavaScript da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a política .

```
import { CreatePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} policyName
 */
export const createPolicy = (policyName) => {
  const command = new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "*",
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK para JavaScript da API.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a função.

```
import { CreateRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const createRole = (roleName) => {
  const command = new CreateRoleCommand({
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            Service: "lambda.amazonaws.com",
          },
          Action: "sts:AssumeRole",
        },
      ],
    }),
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte CreateRole em Referência AWS SDK para JavaScript da API.

`CreateSAMLProvider`

O código de exemplo a seguir mostra como usar `CreateSAMLProvider`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
import * as path from "node:path";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";

const client = new IAMClient({});

/**
 * This sample document was generated using Auth0.
 * For more information on generating this document,
 see https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_providers_create_saml.html#samlstep1.
 */
const sampleMetadataDocument = readFileSync(
  path.join(
    dirnameFromMetaUrl(import.meta.url),
    "../../../../resources/sample_files/sample_saml_metadata.xml",
  ),
);

/**
 *
 * @param {*} providerName
 * @returns
 */
export const createSAMLProvider = async (providerName) => {
  const command = new CreateSAMLProviderCommand({
    Name: providerName,
    SAMLMetadataDocument: sampleMetadataDocument.toString(),
  });

  const response = await client.send(command);
  console.log(response);
  return response;
```

```
};
```

- Consulte detalhes da API em CreateSAMLProvider na Referência da API do AWS SDK para JavaScript .

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma função vinculada ao serviço.

```
import {
  CreateServiceLinkedRoleCommand,
  GetRoleCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} serviceName
 */
export const createServiceLinkedRole = async (serviceName) => {
  const command = new CreateServiceLinkedRoleCommand({
    // For a list of AWS services that support service-linked roles,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-
that-work-with-iam.html.
    //
    // For a list of AWS service endpoints, see https://docs.aws.amazon.com/general/
latest/gr/aws-service-information.html.
    AWSServiceName: serviceName,
  });
```

```
  try {
    const response = await client.send(command);
    console.log(response);
    return response;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInputException" &&
      caught.message.includes(
        "Service role name AWSServiceRoleForElasticBeanstalk has been taken in this
 account",
      )
    ) {
      console.warn(caught.message);
      return client.send(
        new GetRoleCommand({ RoleName: "AWSServiceRoleForElasticBeanstalk" }),
      );
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte CreateServiceLinkedRole em Referência AWS SDK para JavaScript da API.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o usuário.

```
import { CreateUserCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const createUser = (name) => {
  const command = new CreateUserCommand({ UserName: name });
  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateUser em Referência AWS SDK para JavaScript da API.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a chave de acesso.

```
import { DeleteAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const deleteAccessKey = (userName, accessKeyId) => {
  const command = new DeleteAccessKeyCommand({
```

```
    AccessKeyId: accessKeyId,
    UserName: userName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAccessKey em Referência AWS SDK para JavaScript da API.

`DeleteAccountAlias`

O código de exemplo a seguir mostra como usar `DeleteAccountAlias`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua o alias da conta.

```
import { DeleteAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias
 */
export const deleteAccountAlias = (alias) => {
  const command = new DeleteAccountAliasCommand({ AccountAlias: alias });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteAccountAlias em Referência AWS SDK para JavaScript da API.

`DeleteGroup`

O código de exemplo a seguir mostra como usar `DeleteGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const deleteGroup = async (groupName) => {
  const command = new DeleteGroupCommand({
    GroupName: groupName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte DeleteGroup em Referência AWS SDK para JavaScript da API.

`DeleteInstanceProfile`

O código de exemplo a seguir mostra como usar `DeleteInstanceProfile`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const client = new IAMClient({});
await client.send(
  new DeleteInstanceProfileCommand({
    InstanceProfileName: NAMES.instanceProfileName,
  }),
);
```

- Para obter detalhes da API, consulte DeleteInstanceProfile em Referência AWS SDK para JavaScript da API.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a política.

```
import { DeletePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} policyArn
 */
export const deletePolicy = (policyArn) => {
  const command = new DeletePolicyCommand({ PolicyArn: policyArn });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeletePolicy em Referência AWS SDK para JavaScript da API.

`DeleteRole`

O código de exemplo a seguir mostra como usar `DeleteRole`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua o perfil.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteRole em Referência AWS SDK para JavaScript da API.

`DeleteRolePolicy`

O código de exemplo a seguir mostra como usar `DeleteRolePolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 */
export const deleteRolePolicy = (roleName, policyName) => {
  const command = new DeleteRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
  });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteRolePolicy em Referência AWS SDK para JavaScript da API.

`DeleteSAMLProvider`

O código de exemplo a seguir mostra como usar `DeleteSAMLProvider`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} providerArn
 * @returns
 */
export const deleteSAMLProvider = async (providerArn) => {
  const command = new DeleteSAMLProviderCommand({
    SAMLProviderArn: providerArn,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Consulte detalhes da API em DeleteSAMLProvider na Referência da API do AWS SDK para JavaScript .

`DeleteServerCertificate`

O código de exemplo a seguir mostra como usar `DeleteServerCertificate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir um certificado de servidor.

```
import { DeleteServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 */
export const deleteServerCertificate = (certName) => {
  const command = new DeleteServerCertificateCommand({
    ServerCertificateName: certName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteServerCertificate em Referência AWS SDK para JavaScript da API.

`DeleteServiceLinkedRole`

O código de exemplo a seguir mostra como usar `DeleteServiceLinkedRole`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteServiceLinkedRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteServiceLinkedRole = (roleName) => {
  const command = new DeleteServiceLinkedRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteServiceLinkedRole em Referência AWS SDK para JavaScript da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua o usuário.

```
import { DeleteUserCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const deleteUser = (name) => {
  const command = new DeleteUserCommand({ UserName: name });
  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para JavaScript da API.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Desanexe a política.

```
import { DetachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const detachRolePolicy = (policyArn, roleName) => {
```

```
  const command = new DetachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DetachRolePolicy em Referência AWS SDK para JavaScript da API.

`GetAccessKeyLastUsed`

O código de exemplo a seguir mostra como usar `GetAccessKeyLastUsed`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a chave de acesso.

```
import { GetAccessKeyLastUsedCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} accessKeyId
 */
export const getAccessKeyLastUsed = async (accessKeyId) => {
  const command = new GetAccessKeyLastUsedCommand({
    AccessKeyId: accessKeyId,
  });

  const response = await client.send(command);
```

```
  if (response.AccessKeyLastUsed?.LastUsedDate) {
    console.log(`
    ${accessKeyId} was last used by ${response.UserName} via
    the ${response.AccessKeyLastUsed.ServiceName} service on
    ${response.AccessKeyLastUsed.LastUsedDate.toISOString()}
    `);
  }

  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetAccessKeyLastUsed em Referência AWS SDK para JavaScript da API.

`GetAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política de senha da conta.

```
import {
  GetAccountPasswordPolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const getAccountPasswordPolicy = async () => {
  const command = new GetAccountPasswordPolicyCommand({});
```

```
  const response = await client.send(command);
  console.log(response.PasswordPolicy);
  return response;
};
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicy em Referência AWS SDK para JavaScript da API.

`GetPolicy`

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política.

```
import { GetPolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const getPolicy = (policyArn) => {
  const command = new GetPolicyCommand({
    PolicyArn: policyArn,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

- Para obter detalhes da API, consulte GetPolicy em Referência AWS SDK para JavaScript da API.

GetRole

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a função.

```
import { GetRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const getRole = (roleName) => {
  const command = new GetRoleCommand({
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetRole em Referência AWS SDK para JavaScript da API.

GetServerCertificate

O código de exemplo a seguir mostra como usar `GetServerCertificate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um certificado do servidor.

```
import { GetServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 * @returns
 */
export const getServerCertificate = async (certName) => {
  const command = new GetServerCertificateCommand({
    ServerCertificateName: certName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetServerCertificate em Referência AWS SDK para JavaScript da API.

`GetServiceLinkedRoleDeletionStatus`

O código de exemplo a seguir mostra como usar `GetServiceLinkedRoleDeletionStatus`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetServiceLinkedRoleDeletionStatusCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} deletionTaskId
 */
export const getServiceLinkedRoleDeletionStatus = (deletionTaskId) => {
  const command = new GetServiceLinkedRoleDeletionStatusCommand({
    DeletionTaskId: deletionTaskId,
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetServiceLinkedRoleDeletionStatus em Referência AWS SDK para JavaScript da API.

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as chaves de acesso.

```
import { ListAccessKeysCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 * @param {string} userName
 */
export async function* listAccessKeys(userName) {
  const command = new ListAccessKeysCommand({
    MaxItems: 5,
    UserName: userName,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListAccessKeysCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.AccessKeyMetadata?.length) {
    for (const key of response.AccessKeyMetadata) {
      yield key;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccessKeysCommand({
          Marker: response.Marker,
        }),
```

```
      );
    } else {
      break;
    }
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListAccessKeys em Referência AWS SDK para JavaScript da API.

`ListAccountAliases`

O código de exemplo a seguir mostra como usar `ListAccountAliases`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os aliases de conta.

```
import { ListAccountAliasesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 */
export async function* listAccountAliases() {
  const command = new ListAccountAliasesCommand({ MaxItems: 5 });

  let response = await client.send(command);
```

```
  while (response.AccountAliases?.length) {
    for (const alias of response.AccountAliases) {
      yield alias;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccountAliasesCommand({
          Marker: response.Marker,
          MaxItems: 5,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListAccountAliases em Referência AWS SDK para JavaScript da API.

`ListAttachedRolePolicies`

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Lista as políticas que estão anexadas a uma função.

```
import {
  ListAttachedRolePoliciesCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 * @param {string} roleName
 */
export async function* listAttachedRolePolicies(roleName) {
  const command = new ListAttachedRolePoliciesCommand({
    RoleName: roleName,
  });

  let response = await client.send(command);

  while (response.AttachedPolicies?.length) {
    for (const policy of response.AttachedPolicies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAttachedRolePoliciesCommand({
          RoleName: roleName,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListAttachedRolePolicies em Referência AWS SDK para JavaScript da API.

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os grupos.

```
import { ListGroupsCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 */
export async function* listGroups() {
  const command = new ListGroupsCommand({
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.Groups?.length) {
    for (const group of response.Groups) {
      yield group;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListGroupsCommand({
          Marker: response.Marker,
          MaxItems: 10,
        }),
      );
    } else {
      break;
    }
  }
```

```
}
```

- Para obter detalhes da API, consulte ListGroups em Referência AWS SDK para JavaScript da API.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as políticas.

```
import { ListPoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 */
export async function* listPolicies() {
  const command = new ListPoliciesCommand({
    MaxItems: 10,
    OnlyAttached: false,
    // List only the customer managed policies in your Amazon Web Services account.
    Scope: "Local",
  });

  let response = await client.send(command);
```

```
  while (response.Policies?.length) {
    for (const policy of response.Policies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListPoliciesCommand({
          Marker: response.Marker,
          MaxItems: 10,
          OnlyAttached: false,
          Scope: "Local",
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para JavaScript da API.

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as políticas.

```
import { ListRolePoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 * @param {string} roleName
 */
export async function* listRolePolicies(roleName) {
  const command = new ListRolePoliciesCommand({
    RoleName: roleName,
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.PolicyNames?.length) {
    for (const policyName of response.PolicyNames) {
      yield policyName;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolePoliciesCommand({
          RoleName: roleName,
          MaxItems: 10,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListRolePolicies em Referência AWS SDK para JavaScript da API.

`ListRoles`

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os perfis.

```
import { ListRolesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 */
export async function* listRoles() {
  const command = new ListRolesCommand({
    MaxItems: 10,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListRolesCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.Roles?.length) {
    for (const role of response.Roles) {
      yield role;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolesCommand({
          Marker: response.Marker,
        }),
      );
    } else {
```

```
      break;
    }
  }
}
```

- Para obter detalhes da API, consulte ListRoles em Referência AWS SDK para JavaScript da API.

Listar provedores de SAML

O código de exemplo a seguir mostra como usar `ListSAMLProviders`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os provedores SAML.

```
import { ListSAMLProvidersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listSamlProviders = async () => {
  const command = new ListSAMLProvidersCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API do AWS SDK para JavaScript .

`ListServerCertificates`

O código de exemplo a seguir mostra como usar `ListServerCertificates`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os certificados.

```
import { ListServerCertificatesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 */
export async function* listServerCertificates() {
  const command = new ListServerCertificatesCommand({});
  let response = await client.send(command);

  while (response.ServerCertificateMetadataList?.length) {
    for await (const cert of response.ServerCertificateMetadataList) {
      yield cert;
    }

    if (response.IsTruncated) {
      response = await client.send(new ListServerCertificatesCommand({}));
    } else {
      break;
    }
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListServerCertificates em Referência AWS SDK para JavaScript da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os usuários.

```
import { ListUsersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listUsers = async () => {
  const command = new ListUsersCommand({ MaxItems: 10 });

  const response = await client.send(command);

  for (const { UserName, CreateDate } of response.Users) {
    console.log(`${UserName} created on: ${CreateDate}`);
  }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para JavaScript da API.

PutRolePolicy

O código de exemplo a seguir mostra como usar `PutRolePolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { PutRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const examplePolicyDocument = JSON.stringify({
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "VisualEditor0",
      Effect: "Allow",
      Action: [
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts",
      ],
      Resource: "arn:aws:s3:::amzn-s3-demo-bucket",
    },
    {
      Sid: "VisualEditor1",
      Effect: "Allow",
      Action: [
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:ListJobs",
        "s3:ListMultiRegionAccessPoints",
      ],
      Resource: "*",
    },
  ],
});
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 * @param {string} policyDocument
 */
export const putRolePolicy = async (roleName, policyName, policyDocument) => {
  const command = new PutRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
    PolicyDocument: policyDocument,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte PutRolePolicy em Referência AWS SDK para JavaScript da API.

`UpdateAccessKey`

O código de exemplo a seguir mostra como usar `UpdateAccessKey`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize a chave de acesso.

```
import {
  UpdateAccessKeyCommand,
```

```
  IAMClient,
  StatusType,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const updateAccessKey = (userName, accessKeyId) => {
  const command = new UpdateAccessKeyCommand({
    AccessKeyId: accessKeyId,
    Status: StatusType.Inactive,
    UserName: userName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateAccessKey em Referência AWS SDK para JavaScript da API.

`UpdateServerCertificate`

O código de exemplo a seguir mostra como usar `UpdateServerCertificate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize um certificado do servidor.

```
import { UpdateServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} currentName
 * @param {string} newName
 */
export const updateServerCertificate = (currentName, newName) => {
  const command = new UpdateServerCertificateCommand({
    ServerCertificateName: currentName,
    NewServerCertificateName: newName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateServerCertificate em Referência AWS SDK para JavaScript da API.

`UpdateUser`

O código de exemplo a seguir mostra como usar `UpdateUser`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize o usuário.

```
import { UpdateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
```

```
 *
 * @param {string} currentUserName
 * @param {string} newUserName
 */
export const updateUser = (currentUserName, newUserName) => {
  const command = new UpdateUserCommand({
    UserName: currentUserName,
    NewUserName: newUserName,
  });

  return client.send(command);
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte UpdateUser em Referência AWS SDK para JavaScript da API.

`UploadServerCertificate`

O código de exemplo a seguir mostra como usar `UploadServerCertificate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { UploadServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import * as path from "node:path";

const client = new IAMClient({});

const certMessage = `Generate a certificate and key with the following command, or
 the equivalent for your system.

openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
```

```
-keyout example.key -out example.crt -subj "/CN=example.com" \
-addext "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
`;

const getCertAndKey = () => {
  try {
    const cert = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.crt"),
    );
    const key = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.key"),
    );
    return { cert, key };
  } catch (err) {
    if (err.code === "ENOENT") {
      throw new Error(
        `Certificate and/or private key not found. ${certMessage}`,
      );
    }

    throw err;
  }
};

/**
 *
 * @param {string} certificateName
 */
export const uploadServerCertificate = (certificateName) => {
  const { cert, key } = getCertAndKey();
  const command = new UploadServerCertificateCommand({
    ServerCertificateName: certificateName,
    CertificateBody: cert.toString(),
    PrivateKey: key.toString(),
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte UploadServerCertificate em Referência AWS SDK para JavaScript da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
```

```
}
```

Criar etapas para implantar todos os recursos.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
```

```
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
```

```
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']][] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
```

```
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
```

```
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
```

```
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
```

```
    new ScenarioAction("addRoleToInstanceProfile", () => {
      const client = new IAMClient({});
      return client.send(
        new AddRoleToInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    }),
    new ScenarioOutput(
      "addedRoleToInstanceProfile",
      MESSAGES.addedRoleToInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
    ),
    ...initParamsSteps,
    new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
    new ScenarioAction("createLaunchTemplate", async () => {
      const ssmClient = new SSMClient({});
      const { Parameter } = await ssmClient.send(
        new GetParameterCommand({
          Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        }),
      );
      const ec2Client = new EC2Client({});
      await ec2Client.send(
        new CreateLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
          LaunchTemplateData: {
            InstanceType: "t3.micro",
            ImageId: Parameter.Value,
            IamInstanceProfile: { Name: NAMES.instanceProfileName },
            UserData: readFileSync(
              join(RESOURCES_PATH, "server_startup_script.sh"),
            ).toString("base64"),
            KeyName: NAMES.keyPairName,
          },
        }),
      );
    }),
    new ScenarioOutput(
      "createdLaunchTemplate",
      MESSAGES.createdLaunchTemplate.replace(
        "${LAUNCH_TEMPLATE_NAME}",
```

```
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
```

```
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
```

```
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
```

```
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
```

```
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}}
state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
```

```
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }
```

```
      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Criar etapas para executar a demonstração.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
```

```
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
```

```
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
```

```
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
```

```
    new ScenarioOutput("testBrokenDependency", (state) =>
      MESSAGES.demoTestBrokenDependency.replace(
        "${TABLE_NAME}",
        state.badTableName,
      ),
    ),
    ...statusSteps,
    new ScenarioInput(
      "staticResponseConfirmation",
      MESSAGES.demoStaticResponseConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("staticResponse", async (state) => {
      if (!state.staticResponseConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmFailureResponseKey,
            Value: "static",
            Overwrite: true,
            Type: "String",
          }),
        );
      }
    }),
    new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
    ...statusSteps,
    new ScenarioInput(
      "badCredentialsConfirmation",
      MESSAGES.demoBadCredentialsConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("badCredentialsExit", (state) => {
      if (!state.badCredentialsConfirmation) {
        process.exit();
      }
    }),
    new ScenarioAction("fixDynamoDBName", async () => {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
```

```
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-scaling').Instance }}
state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
```

```
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}}
state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
```

```
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
```

```
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
```

```
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
```

```
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Criar etapas para destruir todos os recursos.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
```

```
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
```

```
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);
```

```
      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
```

```
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
```

```
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
```

```
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
```

```
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
```

```
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
```

```
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
```

```
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
```

```
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
 state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
```

```
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
```

```
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers

- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

AWS IoT SiteWise exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com AWS IoT SiteWise.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT SiteWise

O exemplo de código a seguir mostra como começar a usar o AWS IoT SiteWise.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  paginateListAssetModels,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new IoTSiteWiseClient();
  const listAssetModelsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListAssetModels({ client }, { maxResults: 5 });
    for await (const page of paginator) {
      listAssetModelsPaginated.push(...page.assetModelSummaries);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }
  for (const { name, creationDate } of listAssetModelsPaginated) {
    console.log(`${name} - ${creationDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte ListAssetModels em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um modelo AWS IoT SiteWise de ativos.
- Crie um AWS IoT SiteWise ativo.
- Recuperar os valores de ID da propriedade.
- Envie dados para um AWS IoT SiteWise ativo.
- Recupere o valor da propriedade AWS IoT SiteWise Asset.
- Crie um AWS IoT SiteWise portal.
- Crie um AWS IoT SiteWise gateway.
- Descreva o AWS IoT SiteWise Gateway.
- Exclua os AWS IoT SiteWise ativos.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
  //} from "@aws-doc-sdk-examples/lib/scenario/index.js";
} from "../../libs/scenario/index.js";
import {
  IoTSiteWiseClient,
  CreateAssetModelCommand,
```

```
  CreateAssetCommand,
  ListAssetModelPropertiesCommand,
  BatchPutAssetPropertyValueCommand,
  GetAssetPropertyValueCommand,
  CreatePortalCommand,
  DescribePortalCommand,
  CreateGatewayCommand,
  DescribeGatewayCommand,
  DeletePortalCommand,
  DeleteGatewayCommand,
  DeleteAssetCommand,
  DeleteAssetModelCommand,
  DescribeAssetModelCommand,
} from "@aws-sdk/client-iotsitewise";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
  waitUntilStackDeleteComplete,
} from "@aws-sdk/client-cloudformation";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = "SiteWiseBasicsStack";

/**
 * @typedef {{
 *   iotSiteWiseClient: import('@aws-sdk/client-iotsitewise').IotSiteWiseClient,
 *   cloudFormationClient: import('@aws-sdk/client-
cloudformation').CloudFormationClient,
 *   stackName,
 *   stack,
 *   askToDeleteResources: true,
 *   asset: {assetName: "MyAsset1"},
 *   assetModel: {assetModelName: "MyAssetModel1"},
 *   portal: {portalName: "MyPortal1"},
```

```
 *   gateway: {gatewayName: "MyGateway1"},
 *   propertyIds: [],
 *   contactEmail: "user@mydomain.com",
 *   thing: "MyThing1",
 *   sampleData: { temperature: 23.5, humidity: 65.0}
 * }} State
 */

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `AWS IoT SiteWise is a fully managed industrial software-as-a-service (SaaS)
 that makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes. It is designed to help industrial and manufacturing
 organizations collect data from their equipment and processes, and use that data to
 make informed decisions about their operations.
One of the key features of AWS IoT SiteWise is its ability to connect to a wide
 range of industrial equipment and systems, including programmable logic controllers
 (PLCs), sensors, and other industrial devices. It can collect data from these
 devices and organize it into a unified data model, making it easier to analyze and
 gain insights from the data. AWS IoT SiteWise also provides tools for visualizing
 the data, setting up alarms and alerts, and generating reports.
Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data. It can collect and store data from thousands of devices and
 process millions of data points per second, making it suitable for large-scale
 industrial operations. Additionally, AWS IoT SiteWise is designed to be secure
 and compliant, with features like role-based access controls, data encryption,
 and integration with other AWS services for additional security and compliance
 features.

Let's get started...`,
  { header: true },
);

const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
```

```
  "This scenario uses AWS CloudFormation to create an IAM role that is required for
 this scenario. The stack will now be deployed.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../resources/cfn/iotsitewise_basics/SitewiseRoles-
template.yml`,
        "utf8",
      );
      await state.cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await state.cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );
      state.stack = stack.Stacks[0].Outputs[0];
      console.log(`The ARN of the IAM role is ${state.stack.OutputValue}`);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateAWSSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSSiteWiseAssetModel",
```

```
  `1. Create an AWS SiteWise Asset Model
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such
 as equipment, processes, and systems, that exist in an industrial environment.
 This model provides a structured and hierarchical representation of these assets,
 allowing users to define the relationships and properties of each asset.

This scenario creates two asset model properties: temperature and humidity.`,
);

const sdkCreateAWSSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    let assetModelResponse;
    try {
      assetModelResponse = await state.iotSiteWiseClient.send(
        new CreateAssetModelCommand({
          assetModelName: state.assetModel.assetModelName,
          assetModelProperties: [
            {
              name: "Temperature",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
            {
              name: "Humidity",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
          ],
        }),
      );
      state.assetModel.assetModelId = assetModelResponse.assetModelId;
      console.log(
        `Asset Model successfully created. Asset Model ID:
 ${state.assetModel.assetModelId}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceAlreadyExistsException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} already exists.`,
```

```
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateAWSIoTSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSIoTSiteWiseAssetModel",
  `2. Create an AWS IoT SiteWise Asset
The IoT SiteWise model that we just created defines the structure and metadata for
 your physical assets. Now we create an asset from the asset model.

Let's wait 30 seconds for the asset to be ready.`,
);

const waitThirtySeconds = new ScenarioAction("waitThirtySeconds", async () => {
  await wait(30); // wait 30 seconds
  console.log("Time's up! Let's check the asset's status.");
});

const sdkCreateAWSIoTSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSIoTSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    try {
      const assetResponse = await state.iotSiteWiseClient.send(
        new CreateAssetCommand({
          assetModelId: state.assetModel.assetModelId,
          assetName: state.asset.assetName,
        }),
      );
      state.asset.assetId = assetResponse.assetId;
      console.log(`Asset created with ID: ${state.asset.assetId}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset ${state.assetModel.assetModelName} was not found.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
```

```
    }
  },
);

const displayRetrievePropertyId = new ScenarioOutput(
  "displayRetrievePropertyId",
  `3. Retrieve the property ID values

To send data to an asset, we need to get the property ID values. In this scenario,
 we access the temperature and humidity property ID values.`,
);

const sdkRetrievePropertyId = new ScenarioAction(
  "sdkRetrievePropertyId",
  async (state) => {
    try {
      const retrieveResponse = await state.iotSiteWiseClient.send(
        new ListAssetModelPropertiesCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
      for (const retrieveResponseKey in
 retrieveResponse.assetModelPropertySummaries) {
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Humidity"
        ) {
          state.propertyIds.Humidity =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Temperature"
        ) {
          state.propertyIds.Temperature =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
      }
      console.log(`The Humidity propertyId is ${state.propertyIds.Humidity}`);
      console.log(
```

```
        `The Temperature propertyId is ${state.propertyIds.Temperature}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem retrieving the properties: ${caught.message}`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displaySendDataToIoTSiteWiseAsset = new ScenarioOutput(
  "displaySendDataToIoTSiteWiseAsset",
  `4. Send data to an AWS IoT SiteWise Asset

By sending data to an IoT SiteWise Asset, you can aggregate data from multiple
 sources, normalize the data into a standard format, and store it in a centralized
 location. This makes it easier to analyze and gain insights from the data.

In this example, we generate sample temperature and humidity data and send it to the
 AWS IoT SiteWise asset.`,
);

const sdkSendDataToIoTSiteWiseAsset = new ScenarioAction(
  "sdkSendDataToIoTSiteWiseAsset",
  async (state) => {
    try {
      const sendResponse = await state.iotSiteWiseClient.send(
        new BatchPutAssetPropertyValueCommand({
          entries: [
            {
              entryId: "entry-3",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Humidity,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.humidity,
                  },
                  timestamp: {
```

```
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
            {
              entryId: "entry-4",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Temperature,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.temperature,
                  },
                  timestamp: {
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
          ],
        }),
      );
      console.log("The data was sent successfully.");
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrieveValueOfIoTSiteWiseAsset = new ScenarioOutput(
  "displayRetrieveValueOfIoTSiteWiseAsset",
  `5. Retrieve the value of the IoT SiteWise Asset property

IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data from connected equipment and sensors. One of the key benefits of
 reading an IoT SiteWise property is the ability to gain valuable insights from your
 industrial data.`,
);
```

```
const sdkRetrieveValueOfIoTSiteWiseAsset = new ScenarioAction(
  "sdkRetrieveValueOfIoTSiteWiseAsset",
  async (/** @type {State} */ state) => {
    try {
      const temperatureResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Temperature,
        }),
      );
      const humidityResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Humidity,
        }),
      );
      console.log(
        `The property value for Temperature is
 ${temperatureResponse.propertyValue.value.doubleValue}`,
      );
      console.log(
        `The property value for Humidity is
 ${humidityResponse.propertyValue.value.doubleValue}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateIoTSiteWisePortal = new ScenarioOutput(
  "displayCreateIoTSiteWisePortal",
  `6. Create an IoT SiteWise Portal

An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources, such as sensors, equipment, and control systems, into a centralized
 platform.`,
);
```

```
const sdkCreateIoTSiteWisePortal = new ScenarioAction(
  "sdkCreateIoTSiteWisePortal",
  async (/** @type {State} */ state) => {
    try {
      const createPortalResponse = await state.iotSiteWiseClient.send(
        new CreatePortalCommand({
          portalName: state.portal.portalName,
          portalContactEmail: state.contactEmail,
          roleArn: state.stack.OutputValue,
        }),
      );
      state.portal = { ...state.portal, ...createPortalResponse };
      await wait(5); // Allow the portal to properly propagate.
      console.log(
        `Portal created successfully. Portal ID ${createPortalResponse.portalId}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the Portal: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribePortal = new ScenarioOutput(
  "displayDescribePortal",
  `7. Describe the Portal

In this step, we get a description of the portal and display the portal URL.`,
);

const sdkDescribePortal = new ScenarioAction(
  "sdkDescribePortal",
  async (/** @type {State} */ state) => {
    try {
      const describePortalResponse = await state.iotSiteWiseClient.send(
        new DescribePortalCommand({
          portalId: state.portal.portalId,
```

```
        }),
      );
      console.log(`Portal URL: ${describePortalResponse.portalStartUrl}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateIoTSiteWiseGateway = new ScenarioOutput(
  "displayCreateIoTSiteWiseGateway",
  `8. Create an IoT SiteWise Gateway

IoT SiteWise Gateway serves as the bridge between industrial equipment, sensors, and
 the cloud-based IoT SiteWise service. It is responsible for securely collecting,
 processing, and transmitting data from various industrial assets to the IoT
 SiteWise platform, enabling real-time monitoring, analysis, and optimization of
 industrial operations.`,
);

const sdkCreateIoTSiteWiseGateway = new ScenarioAction(
  "sdkCreateIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const createGatewayResponse = await state.iotSiteWiseClient.send(
        new CreateGatewayCommand({
          gatewayName: state.gateway.gatewayName,
          gatewayPlatform: {
            greengrassV2: {
              coreDeviceThingName: state.thing,
            },
          },
        }),
      );
      console.log(
        `Gateway creation completed successfully. ID is
 ${createGatewayResponse.gatewayId}`,
      );
      state.gateway.gatewayId = createGatewayResponse.gatewayId;
```

```
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the gateway: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribeIoTSiteWiseGateway = new ScenarioOutput(
  "displayDescribeIoTSiteWiseGateway",
  "9. Describe the IoT SiteWise Gateway",
);

const sdkDescribeIoTSiteWiseGateway = new ScenarioAction(
  "sdkDescribeIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const describeGatewayResponse = await state.iotSiteWiseClient.send(
        new DescribeGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
      console.log("Gateway creation completed successfully.");
      console.log(`Gateway Name: ${describeGatewayResponse.gatewayName}`);
      console.log(`Gateway ARN: ${describeGatewayResponse.gatewayArn}`);
      console.log(
        `Gateway Platform: ${Object.keys(describeGatewayResponse.gatewayPlatform)}`,
      );
      console.log(
        `Gateway Creation Date: ${describeGatewayResponse.creationDate}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
```

```
  },
);

const askToDeleteResources = new ScenarioInput(
  "askToDeleteResources",
  `10. Delete the AWS IoT SiteWise Assets

Before you can delete the Asset Model, you must delete the assets.`,
  { type: "confirm" },
);

const displayConfirmDeleteResources = new ScenarioAction(
  "displayConfirmDeleteResources",
  async (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You selected to delete the SiteWise assets.";
    }
    return "The resources will not be deleted. Please delete them manually to avoid
 charges.";
  },
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    await wait(10); // Give the portal status time to catch up.
    try {
      await state.iotSiteWiseClient.send(
        new DeletePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(
        `Portal ${state.portal.portalName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
      } else {
        console.log(`When trying to delete the portal: ${caught.message}`);
      }
    }

    try {
```

```
    await state.iotSiteWiseClient.send(
      new DeleteGatewayCommand({
        gatewayId: state.gateway.gatewayId,
      }),
    );
    console.log(
      `Gateway ${state.gateway.gatewayName} was deleted successfully.`,
    );
  } catch (caught) {
    if (caught.name === "ResourceNotFoundException") {
      console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
    } else {
      console.log(`When trying to delete the gateway: ${caught.message}`);
    }
  }

  try {
    await state.iotSiteWiseClient.send(
      new DeleteAssetCommand({
        assetId: state.asset.assetId,
      }),
    );
    await wait(5); // Allow the delete to finish.
    console.log(`Asset ${state.asset.assetName} was deleted successfully.`);
  } catch (caught) {
    if (caught.name === "ResourceNotFoundException") {
      console.log(`The Asset ${state.asset.assetName} was not found.`);
    } else {
      console.log(`When deleting the asset: ${caught.message}`);
    }
  }

  await wait(30); // Allow asset deletion to finish.
  try {
    await state.iotSiteWiseClient.send(
      new DeleteAssetModelCommand({
        assetModelId: state.assetModel.assetModelId,
      }),
    );
    console.log(
      `Asset Model ${state.assetModel.assetModelName} was deleted successfully.`,
    );
  } catch (caught) {
    if (caught.name === "ResourceNotFoundException") {
```

```
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} was not found.`,
        );
      } else {
        console.log(`When deleting the asset model: ${caught.message}`);
      }
    }

    try {
      await state.cloudFormationClient.send(
        new DeleteStackCommand({
          StackName: stackName,
        }),
      );
      await waitUntilStackDeleteComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      console.log("The stack was deleted successfully.");
    } catch (caught) {
      console.log(
        `${caught.message}. The stack was NOT deleted. Please clean up the resources
 manually.`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the IoT Sitewise Basics scenario for the AWS Javascript SDK v3.
 Thank you!",
);

const myScenario = new Scenario(
  "IoTSiteWise Basics",
  [
    greet,
    pressEnter,
    displayBuildCloudFormationStack,
    sdkBuildCloudFormationStack,
    pressEnter,
    displayCreateAWSSiteWiseAssetModel,
```

```
    sdkCreateAWSSiteWiseAssetModel,
    displayCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    waitThirtySeconds,
    sdkCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    displayRetrievePropertyId,
    sdkRetrievePropertyId,
    pressEnter,
    displaySendDataToIoTSiteWiseAsset,
    sdkSendDataToIoTSiteWiseAsset,
    pressEnter,
    displayRetrieveValueOfIoTSiteWiseAsset,
    sdkRetrieveValueOfIoTSiteWiseAsset,
    pressEnter,
    displayCreateIoTSiteWisePortal,
    sdkCreateIoTSiteWisePortal,
    pressEnter,
    displayDescribePortal,
    sdkDescribePortal,
    pressEnter,
    displayCreateIoTSiteWiseGateway,
    sdkCreateIoTSiteWiseGateway,
    pressEnter,
    displayDescribeIoTSiteWiseGateway,
    sdkDescribeIoTSiteWiseGateway,
    pressEnter,
    askToDeleteResources,
    displayConfirmDeleteResources,
    sdkDeleteResources,
    goodbye,
  ],
  {
    iotSiteWiseClient: new IoTSiteWiseClient({}),
    cloudFormationClient: new CloudFormationClient({}),
    asset: { assetName: "MyAsset1" },
    assetModel: { assetModelName: "MyAssetModel1" },
    portal: { portalName: "MyPortal1" },
    gateway: { gatewayName: "MyGateway1" },
    propertyIds: [],
    contactEmail: "user@mydomain.com",
    thing: "MyThing1",
    sampleData: { temperature: 23.5, humidity: 65.0 },
  },
```

```
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties
  - ListAssetModels

Ações

`BatchPutAssetPropertyValue`

O código de exemplo a seguir mostra como usar `BatchPutAssetPropertyValue`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  BatchPutAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Batch put asset property values.
 * @param {{ entries : array }}
 */
export const main = async ({ entries }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new BatchPutAssetPropertyValueCommand({
        entries: entries,
      }),
    );
    console.log("Asset properties batch put successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(`${caught.message}. A resource could not be found.`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte BatchPutAssetPropertyValue em Referência AWS SDK para JavaScript da API.

`CreateAsset`

O código de exemplo a seguir mostra como usar `CreateAsset`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  CreateAssetCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetName, assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateAssetCommand({
        assetName: assetName, // The name to give the Asset.
        assetModelId: assetModelId, // The ID of the asset model from which to
 create the asset.
      }),
    );
    console.log("Asset created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
```

```
      console.warn(
        `${caught.message}. The asset model could not be found. Please check the
 asset model id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateAsset em Referência AWS SDK para JavaScript da API.

`CreateAssetModel`

O código de exemplo a seguir mostra como usar `CreateAssetModel`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  CreateAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset Model.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetModelName, assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateAssetModelCommand({
```

```
        assetModelName: assetModelName, // The name to give the Asset Model.
      }),
    );
    console.log("Asset model created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the asset model.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateAssetModel em Referência AWS SDK para JavaScript da API.

`CreateGateway`

O código de exemplo a seguir mostra como usar `CreateGateway`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  CreateGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create a Gateway.
 * @param {{  }}
```

```
 */
export const main = async ({ gatewayName }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateGatewayCommand({
        gatewayName: gatewayName, // The name to give the created Gateway.
      }),
    );
    console.log("Gateway created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the Gateway.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateGateway em Referência AWS SDK para JavaScript da API.

`DeleteAsset`

O código de exemplo a seguir mostra como usar `DeleteAsset`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DeleteAssetCommand,
  IoTSiteWiseClient,
```

```
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset.
 * @param {{ assetId : string }}
 */
export const main = async ({ assetId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteAssetCommand({
        assetId: assetId, // The model id to delete.
      }),
    );
    console.log("Asset deleted successfully.");
    return { assetDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteAsset em Referência AWS SDK para JavaScript da API.

`DeleteAssetModel`

O código de exemplo a seguir mostra como usar `DeleteAssetModel`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DeleteAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteAssetModelCommand({
        assetModelId: assetModelId, // The model id to delete.
      }),
    );
    console.log("Asset model deleted successfully.");
    return { assetModelDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset model.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteAssetModel em Referência AWS SDK para JavaScript da API.

`DeleteGateway`

O código de exemplo a seguir mostra como usar `DeleteGateway`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DeleteGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Gateway deleted successfully.");
    return { gatewayDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the Gateway
 Id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteGateway em Referência AWS SDK para JavaScript da API.

`DescribeAssetModel`

O código de exemplo a seguir mostra como usar `DescribeAssetModel`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DescribeAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { assetModelDescription } = await client.send(
      new DescribeAssetModelCommand({
        assetModelId: assetModelId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset model information retrieved successfully.");
    return { assetModelDescription: assetModelDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
```

```
        `${caught.message}. The asset model could not be found. Please check the
 asset model id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeAssetModel em Referência AWS SDK para JavaScript da API.

`DescribeGateway`

O código de exemplo a seguir mostra como usar `DescribeGateway`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  DescribeGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { gatewayDescription } = await client.send(
      new DescribeGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
```

```
      }),
    );
    console.log("Gateway information retrieved successfully.");
    return { gatewayDescription: gatewayDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the Gateway
 Id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DescribeGateway em Referência AWS SDK para JavaScript da API.

`GetAssetPropertyValue`

O código de exemplo a seguir mostra como usar `GetAssetPropertyValue`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe an asset property value.
 * @param {{ entryId : string }}
```

```
 */
export const main = async ({ entryId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new GetAssetPropertyValueCommand({
        entryId: entryId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset property information retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset property entry could not be found. Please
 check the entry id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte GetAssetPropertyValue em Referência AWS SDK para JavaScript da API.

`ListAssetModels`

O código de exemplo a seguir mostra como usar `ListAssetModels`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  ListAssetModelsCommand,
```

```
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * List asset models.
 * @param {{ assetModelTypes : array }}
 */
export const main = async ({ assetModelTypes = [] }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new ListAssetModelsCommand({
        assetModelTypes: assetModelTypes, // The model types to list
      }),
    );
    console.log("Asset model types retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem listing the asset model types.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte ListAssetModels em Referência AWS SDK para JavaScript da API.

Exemplos do Kinesis usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Kinesis.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Exemplos sem servidor

Ações

`PutRecords`

O código de exemplo a seguir mostra como usar `PutRecords`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { PutRecordsCommand, KinesisClient } from "@aws-sdk/client-kinesis";

/**
 * Put multiple records into a Kinesis stream.
 * @param {{ streamArn: string }} config
 */
export const main = async ({ streamArn }) => {
  const client = new KinesisClient({});
  try {
    await client.send(
      new PutRecordsCommand({
        StreamARN: streamArn,
        Records: [
          {
            Data: new Uint8Array(),
            /**
             * Determines which shard in the stream the data record is assigned to.
             * Partition keys are Unicode strings with a maximum length limit of 256
```

```
             * characters for each key. Amazon Kinesis Data Streams uses the
 partition
             * key as input to a hash function that maps the partition key and
             * associated data to a specific shard.
             */
            PartitionKey: "TEST_KEY",
          },
          {
            Data: new Uint8Array(),
            PartitionKey: "TEST_KEY",
          },
        ],
      }),
    );
  } catch (caught) {
    if (caught instanceof Error) {
      //
    } else {
      throw caught;
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    streamArn: {
      type: "string",
      description: "The ARN of the stream.",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- Para obter detalhes da API, consulte PutRecords em Referência AWS SDK para JavaScript da API.

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do Kinesis com o Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      throw err;
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Consumindo um evento do Kinesis com o Lambda usando. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      throw err;
    }
    logger.info(`Successfully processed ${event.Records.length} records.`);
  }
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Javascript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed item
 onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};
```

```
async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Relatando falhas de itens de lote do Kinesis com o uso do Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
  KinesisStreamBatchResponse,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<KinesisStreamBatchResponse> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed item
 onwards. */
```

```
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  logger.info(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Exemplos de Lambda usando SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Lambda

O exemplo de código a seguir mostra como começar a usar o Lambda.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { LambdaClient, paginateListFunctions } from "@aws-sdk/client-lambda";

const client = new LambdaClient({});

export const helloLambda = async () => {
  const paginator = paginateListFunctions({ client }, {});
  const functions = [];

  for await (const page of paginator) {
    const funcNames = page.Functions.map((f) => f.FunctionName);
    functions.push(...funcNames);
  }

  console.log("Functions:");
  console.log(functions.join("\n"));
  return functions;
};
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma função AWS Identity and Access Management (IAM) que conceda permissão ao Lambda para gravar em registros.

```
    logger.log(`Creating role (${NAME_ROLE_LAMBDA})...`);
    const response = await createRole(NAME_ROLE_LAMBDA);

import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
```

```
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

Crie uma função do Lambda e carregue o código de manipulador.

```
const createFunction = async (funcName, roleArn) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${funcName}.zip`);

  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

Invoque essa função com um único parâmetro e receba resultados.

```
const invoke = async (funcName, payload) => {
  const client = new LambdaClient({});
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

Atualize o código de função e configure seu ambiente do Lambda usando uma variável de ambiente.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${newFunc}.zip`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};

const updateFunctionConfiguration = async (funcName) => {
  const client = new LambdaClient({});
  const config = readFileSync(`${dirname}../functions/config.json`).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  const result = await client.send(command);
  await waitForFunctionUpdated({ FunctionName: funcName });
  return result;
};
```

Liste as funções para a sua conta.

```
const listFunctions = () => {
  const client = new LambdaClient({});
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

Exclua o perfil do IAM e a função do Lambda.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};

/**
 * @param {string} funcName
 */
const deleteFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const createFunction = async (funcName, roleArn) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${funcName}.zip`);

  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte CreateFunction em Referência AWS SDK para JavaScript da API.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * @param {string} funcName
 */
const deleteFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte DeleteFunction em Referência AWS SDK para JavaScript da API.

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const getFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new GetFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Para obter detalhes da API, consulte GetFunction em Referência AWS SDK para JavaScript da API.

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const invoke = async (funcName, payload) => {
  const client = new LambdaClient({});
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para JavaScript .

ListFunctions

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const listFunctions = () => {
  const client = new LambdaClient({});
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para JavaScript da API.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${newFunc}.zip`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
```

```
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- Para obter detalhes da API, consulte UpdateFunctionCode em Referência AWS SDK para JavaScript da API.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
const updateFunctionConfiguration = async (funcName) => {
  const client = new LambdaClient({});
  const config = readFileSync(`${dirname}../functions/config.json`).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  const result = await client.send(command);
  await waitForFunctionUpdated({ FunctionName: funcName });
  return result;
};
```

- Para obter detalhes da API, consulte UpdateFunctionConfiguration em Referência AWS SDK para JavaScript da API.

Cenários

Confirme automaticamente usuários conhecidos com uma função do Lambda

O exemplo de código a seguir mostra como confirmar automaticamente usuários conhecidas do Amazon Cognito com uma função do Lambda.

- Configure um grupo de usuários para chamar uma função do Lambda para o acionador `PreSignUp`.
- Inscreva-se para ser um usuário no Amazon Cognito.
- A função do Lambda verifica uma tabela do DynamoDB e confirma automaticamente os usuários conhecidos.
- Faça login como o novo usuário e, em seguida, limpe os recursos.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure uma execução interativa de “Cenário”. Os exemplos JavaScript (v3) compartilham um executor de cenários para simplificar exemplos complexos. O código-fonte completo está ativado GitHub.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      UserEmail: "test_email_1@example.com",
    },
```

```
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

Esse cenário demonstra a confirmação automática de um usuário conhecido. Ele orquestra as etapas do exemplo.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
```

```
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
```

```
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
```

```
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });
```

```
    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
    const signUp = (password) =>
      signUpUser({
```

```
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
    if (logStreamErr) {
      state.errors.push(logStreamErr);
```

```
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
```

```
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with: ${state.token.slice(0,
 11)}`,
  { skipWhen: skipWhenErrors },
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
```

```
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

Essas são etapas compartilhadas com outros cenários.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
```

```
  { skipWhen: skipWhenErrors },
);
```

Um manipulador do gatilho `PreSignUp` com uma função do Lambda.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user '${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
```

```
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
 '${storedUserInfo.UserName}' does not match supplied UserName '${event.userName}'.
 Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

Módulo de ações de CloudWatch registros.

```
import {
```

```
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[] |
 null, unknown]>}
```

```
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Módulo de ações do Amazon Cognito.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
```

```
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
```

```
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password: string }}
 config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
```

```
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

Módulo de ações do DynamoDB.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
```

```
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string, unknown>[] }}
 config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado dentro das funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);
```

```
  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
```

```
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Invocar uma função do Lambda em um navegador

O exemplo de código a seguir mostra como invocar uma AWS Lambda função em um navegador.

SDK para JavaScript (v3)

Você pode criar um aplicativo baseado em navegador que usa uma AWS Lambda função para atualizar uma tabela do Amazon DynamoDB com as seleções do usuário. Este aplicativo usa AWS SDK para JavaScript v3.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectando-se a um banco de dados do Amazon RDS em uma função do Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
```

```
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}

export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

Conectando-se a um banco de dados do Amazon RDS em uma função do Lambda usando. TypeScript

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that the
 DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
```

```
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}

async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number; body:
 string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
```

```
    };
};
```

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do Kinesis com o Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      throw err;
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Consumindo um evento do Kinesis com o Lambda usando. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      throw err;
    }
    logger.info(`Successfully processed ${event.Records.length} records.`);
  }
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
```

```
  return data;
}
```

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do DynamoDB com o Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
};

const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Consumindo um evento do DynamoDB com o Lambda usando. TypeScript

```
export const handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
```

```
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
}
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento Amazon DocumentDB com o Lambda usando. JavaScript

```
console.log('Loading function');
exports.handler = async (event, context) => {
    event.events.forEach(record => {
        logDocumentDBEvent(record);
    });
    return 'OK';
};

const logDocumentDBEvent = (record) => {
    console.log('Operation type: ' + record.event.operationType);
    console.log('db: ' + record.event.ns.db);
    console.log('collection: ' + record.event.ns.coll);
    console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

Consumindo um evento do Amazon DocumentDB com o Lambda usando TypeScript

```
import { DocumentDBEventRecord, DocumentDBEventSubscriptionContext } from 'aws-
lambda';

console.log('Loading function');

export const handler = async (
  event: DocumentDBEventSubscriptionContext,
  context: any
): Promise<string> => {
  event.events.forEach((record: DocumentDBEventRecord) => {
    logDocumentDBEvent(record);
  });
  return 'OK';
};

const logDocumentDBEvent = (record: DocumentDBEventRecord): void => {
  console.log('Operation type: ' + record.event.operationType);
  console.log('db: ' + record.event.ns.db);
  console.log('collection: ' + record.event.ns.coll);
  console.log('Full document:', JSON.stringify(record.event.fullDocument, null, 2));
};
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento Amazon MSK com o Lambda usando. JavaScript

```
exports.handler = async (event) => {
    // Iterate through keys
    for (let key in event.records) {
      console.log('Key: ', key)
      // Iterate through records
      event.records[key].map((record) => {
        console.log('Record: ', record)
        // Decode base64
        const msg = Buffer.from(record.value, 'base64').toString()
        console.log('Message:', msg)
      })
    }
}
```

Consumindo um evento Amazon MSK com o Lambda usando. TypeScript

```
import { MSKEvent, Context } from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "msk-handler-sample",
});

export const handler = async (
  event: MSKEvent,
  context: Context
): Promise<void> => {
  for (const [topic, topicRecords] of Object.entries(event.records)) {
    logger.info(`Processing key: ${topic}`);

    // Process each record in the partition
    for (const record of topicRecords) {
      try {
        // Decode the message value from base64
        const decodedMessage = Buffer.from(record.value, 'base64').toString();

        logger.info({
          message: decodedMessage
```

```
        });
      }
      catch (error) {
        logger.error('Error processing event', { error });
        throw error;
      }
    };
  }
}
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do S3 com o Lambda usando. JavaScript

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
```

```
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Consumindo um evento do S3 com o Lambda usando. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
```

```
    console.log(message);
    throw new Error(message);
  }
};
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento SNS com o Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Consumindo um evento SNS com o Lambda usando. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento SQS com o Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message) {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Consumindo um evento SQS com o Lambda usando. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";

export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
```

```
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Javascript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed item
 onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
```

```
      };
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Relatando falhas de itens de lote do Kinesis com o uso do Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
  KinesisStreamBatchResponse,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<KinesisStreamBatchResponse> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
```

```
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed item
 onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  logger.info(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatando falhas de itens de lote do DynamoDB com o uso do Lambda. JavaScript

```
export const handler = async (event) => {
```

```
  const records = event.Records;
  let curRecordSequenceNumber = "";

  for (const record of records) {
    try {
      // Process your record
      curRecordSequenceNumber = record.dynamodb.SequenceNumber;
    } catch (e) {
      // Return failed record's sequence number
      return { batchItemFailures: [{ itemIdentifier: curRecordSequenceNumber }] };
    }
  }

  return { batchItemFailures: [] };
};
```

Relatando falhas de itens de lote do DynamoDB com o uso do Lambda. TypeScript

```
import {
  DynamoDBBatchResponse,
  DynamoDBBatchItemFailure,
  DynamoDBStreamEvent,
} from "aws-lambda";

export const handler = async (
  event: DynamoDBStreamEvent
): Promise<DynamoDBBatchResponse> => {
  const batchItemFailures: DynamoDBBatchItemFailure[] = [];
  let curRecordSequenceNumber;

  for (const record of event.Records) {
    curRecordSequenceNumber = record.dynamodb?.SequenceNumber;

    if (curRecordSequenceNumber) {
      batchItemFailures.push({
        itemIdentifier: curRecordSequenceNumber,
      });
    }
  }

  return { batchItemFailures: batchItemFailures };
};
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatando falhas de itens de lote do SQS com o uso do Lambda. JavaScript

```
// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
    console.log(`Processed message: ${record.body}`);
}
```

Relatando falhas de itens de lote do SQS com o uso do Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord } from
 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
 Promise<SQSBatchResponse> => {
    const batchItemFailures: SQSBatchItemFailure[] = [];

    for (const record of event.Records) {
        try {
            await processMessageAsync(record);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }

    return {batchItemFailures: batchItemFailures};
};

async function processMessageAsync(record: SQSRecord): Promise<void> {
    if (record.body && record.body.includes("error")) {
        throw new Error('There is an error in the SQS Message.');
    }
    console.log(`Processed message ${record.body}`);
}
```

Exemplos do Amazon Lex usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Lex.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar um chatbot Amazon Lex

O exemplo de código a seguir mostra como criar um chatbot para engajar os visitantes do seu site.

SDK para JavaScript (v3)

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o exemplo completo Como criar um chatbot Amazon Lex no guia do AWS SDK para JavaScript desenvolvedor.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Exemplos de localização da Amazon usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Location.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos

- Ações

Conceitos básicos

Olá, Amazon Location

O exemplo de código a seguir mostra como começar a usar o Amazon Location Service.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  LocationClient,
  ListGeofenceCollectionsCommand,
} from "@aws-sdk/client-location";

/**
 * Lists geofences from a specified geofence collection asynchronously.
 */
export const main = async () => {
  const region = "eu-west-1";
  const locationClient = new LocationClient({ region: region });
  const listGeofenceCollParams = {
    MaxResults: 100,
  };
  try {
    const command = new ListGeofenceCollectionsCommand(listGeofenceCollParams);
    const response = await locationClient.send(command);
    const geofenceEntries = response.Entries;
    if (geofenceEntries.length === 0) {
      console.log("No Geofences were found in the collection.");
    } else {
      for (const geofenceEntry of geofenceEntries) {
        console.log(`Geofence ID: ${geofenceEntry.CollectionName}`);
      }
```

```
    }
  } catch (error) {
    console.error(
      `A validation error occurred while creating geofence: ${error} \n Exiting
 program.`,
    );
    return;
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - ListGeofenceCollections
  - ListGeofences

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um mapa do Amazon Location.
- Criar uma chave de API do Amazon Location.
- Exibir o URL do mapa.
- Criar uma coleção de geocercas
- Armazenar uma geometria de geocerca.
- Criar um recurso de rastreador.
- Atualizar a posição de um dispositivo.
- Recuperar a atualização de posição mais recente de um dispositivo específico.
- Criar uma calculadora de rotas.
- Determinar a distância entre Seattle e Vancouver.
- Usar as APIs de nível superior do Amazon Location.
- Excluir os ativos do Amazon Location.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/*
Before running this JavaScript code example, set up your development environment,
 including your credentials.
This demo illustrates how to use the AWS SDK for JavaScript (v3) to work with Amazon
 Location Service.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/getting-
started.html
*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

import {
  CreateMapCommand,
  CreateGeofenceCollectionCommand,
  PutGeofenceCommand,
  CreateTrackerCommand,
  BatchUpdateDevicePositionCommand,
  GetDevicePositionCommand,
  CreateRouteCalculatorCommand,
  CalculateRouteCommand,
  LocationClient,
  ConflictException,
  ResourceNotFoundException,
  DeleteGeofenceCollectionCommand,
  DeleteRouteCalculatorCommand,
```

```
  DeleteTrackerCommand,
  DeleteMapCommand,
} from "@aws-sdk/client-location";

import {
  GeoPlacesClient,
  ReverseGeocodeCommand,
  SearchNearbyCommand,
  SearchTextCommand,
  GetPlaceCommand,
  ValidationException,
} from "@aws-sdk/client-geo-places";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";

/*The inputs for this example can be edited in the ./input.json.*/
import data from "./inputs.json" with { type: "json" };

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
/* v8 ignore next 3 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
  verbose: "false",
});

const pressEnterConfirm = new ScenarioInput(
  "confirm",
  "Press Enter to continue",
  {
    type: "confirm",
    verbose: "false",
  },
);

const region = "eu-west-1";

const locationClient = new LocationClient({ region: region });

const greet = new ScenarioOutput(
  "greet",
```

```
  "Welcome to the Amazon Location Use demo! \n" +
    "AWS Location Service is a fully managed service offered by Amazon Web Services
 (AWS) that " +
    "provides location-based services for developers. This service simplifies " +
    "the integration of location-based features into applications, making it " +
    "Maps: The service provides access to high-quality maps, satellite imagery, " +
    "and geospatial data from various providers, allowing developers to " +
    "easily embed maps into their applications:\n" +
    "Tracking: The Location Service enables real-time tracking of mobile devices, "
 +
    "assets, or other entities, allowing developers to build applications " +
    "that can monitor the location of people, vehicles, or other objects.\n" +
    "Geocoding: The service provides the ability to convert addresses or " +
    "location names into geographic coordinates (latitude and longitude), " +
    "and vice versa, enabling developers to integrate location-based search " +
    "and routing functionality into their applications. " +
    "Please define values ./inputs.json for each user-defined variable used in this
 app. Otherwise the default is used:\n" +
    "- mapName: The name of the map to be create (default is 'AWSMap').\n" +
    "- keyName: The name of the API key to create (default is ' AWSApiKey')\n" +
    "- collectionName: The name of the geofence collection (default is
 'AWSLocationCollection')\n" +
    "- geoId: The geographic identifier used for the geofence or map (default is
 'geoId')\n" +
    "- trackerName: The name of the tracker (default is 'geoTracker')\n" +
    "- calculatorName: The name of the route calculator (default is
 'AWSRouteCalc')\n" +
    "- deviceId: The ID of the device (default is 'iPhone-112356')",

  { header: true },
);
const displayCreateAMap = new ScenarioOutput(
  "displayCreateAMap",
  "1. Create a map\n" +
    "An AWS Location map can enhance the user experience of your " +
    " application by providing accurate and personalized location-based " +
    " features. For example, you could use the geocoding capabilities to " +
    " allow users to search for and locate businesses, landmarks, or " +
    " other points of interest within a specific region.",
);

const sdkCreateAMap = new ScenarioAction(
  "sdkCreateAMap",
  async (/** @type {State} */ state) => {
```

```
    const createMapParams = {
      MapName: `${data.inputs.mapName}`,
      Configuration: { style: "VectorEsriNavigation" },
    };
    try {
      const command = new CreateMapCommand(createMapParams);
      const response = await locationClient.send(command);
      state.MapName = response.MapName;
      console.log("Map created. Map ARN is: ", state.MapName);
    } catch (error) {
      console.error("Error creating map: ", error);
      throw error;
    }
  },
);

const displayMapUrl = new ScenarioOutput(
  "displayMapUrl",
  "2. Display Map URL\n" +
    "When you embed a map in a web app or website, the API key is " +
    "included in the map tile URL to authenticate requests. You can " +
    "restrict API keys to specific AWS Location operations (e.g., only " +
    "maps, not geocoding). API keys can expire, ensuring temporary " +
    "access control.\n" +
    "In order to get the MAP URL you need to create and get the API Key value. " +
    "You can create and get the key value using the AWS Management Console under " +
    "Location Services. These operations cannot be completed using the " +
    "AWS SDK. For more information about getting the key value, see " +
    "the AWS Location Documentation.",
);

const sdkDisplayMapUrl = new ScenarioAction(
  "sdkDisplayMapUrl",
  async (/** @type {State} */ state) => {
    const mapURL = `https://maps.geo.aws.amazon.com/maps/v0/maps/${state.MapName}/
tiles/{z}/{x}/{y}?key=API_KEY_VALUE`;
    state.mapURL = mapURL;
    console.log(
      `Replace \'API_KEY_VALUE\' in the following URL with the value for the API key
 you create and get from the AWS Management Console under Location Services. This is
 then the Map URL you can embed this URL in your Web app:\n
${state.mapURL}`,
    );
  },
```

```
);
const displayCreateGeoFenceColl = new ScenarioOutput(
  "displayCreateGeoFenceColl",
  "3. Create a geofence collection, which manages and stores geofences.",
);

const sdkCreateGeoFenceColl = new ScenarioAction(
  "sdkCreateGeoFenceColl",
  async (/** @type {State} */ state) => {
    // Creates a new geofence collection.
    const geoFenceCollParams = {
      CollectionName: `${data.inputs.collectionName}`,
    };
    try {
      const command = new CreateGeofenceCollectionCommand(geoFenceCollParams);
      const response = await locationClient.send(command);
      state.CollectionName = response.CollectionName;
      console.log(
        `The geofence collection was successfully created: ${state.CollectionName}`,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An unexpected error occurred while creating the geofence collection:
 ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayStoreGeometry = new ScenarioOutput(
  "displayStoreGeometry",
  "4. Store a geofence geometry in a given geofence collection. " +
    "An AWS Location geofence is a virtual boundary that defines a geographic area "
 +
    "on a map. It is a useful feature for tracking the location of " +
    "assets or monitoring the movement of objects within a specific region. " +
    "To define a geofence, you need to specify the coordinates of a " +
    "polygon that represents the area of interest. The polygon must be " +
    "defined in a counter-clockwise direction, meaning that the points of " +
    "the polygon must be listed in a counter-clockwise order. " +
    "This is a requirement for the AWS Location service to correctly " +
    "interpret the geofence and ensure that the location data is " +
```

```
    "accurately processed within the defined area.",
);

const sdkStoreGeometry = new ScenarioAction(
  "sdkStoreGeometry",
  async (/** @type {State} */ state) => {
    const geoFenceGeoParams = {
      CollectionName: `${data.inputs.collectionName}`,
      GeofenceId: `${data.inputs.geoId}`,
      Geometry: {
        Polygon: [
          [
            [-122.3381, 47.6101],
            [-122.3281, 47.6101],
            [-122.3281, 47.6201],
            [-122.3381, 47.6201],
            [-122.3381, 47.6101],
          ],
        ],
      },
    };
    try {
      const command = new PutGeofenceCommand(geoFenceGeoParams);
      const response = await locationClient.send(command);
      state.GeoFencId = response.GeofenceId;
      console.log("GeoFence created. GeoFence ID is: ", state.GeoFencId);
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `A validation error occurred while creating geofence: ${caught.message} \n
 Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayCreateTracker = new ScenarioOutput(
  "displayCreateTracker",
  "5. Create a tracker resource which lets you retrieve current and historical
 location of devices.",
);

const sdkCreateTracker = new ScenarioAction(
```

```
  "sdkCreateTracker",
  async (/** @type {State} */ state) => {
    //Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
    const createTrackerParams = {
      TrackerName: `${data.inputs.trackerName}`,
      Description: "Created using the JavaScript V3 SDK",
      PositionFiltering: "TimeBased",
    };
    try {
      const command = new CreateTrackerCommand(createTrackerParams);
      const response = await locationClient.send(command);
      state.trackerName = response.TrackerName;
      console.log("Tracker created. Tracker name is : ", state.trackerName);
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `A validation error occurred while creating geofence: ${caught.message} \n
 Exiting program.`,
        );
      } else {
        `An unexpected error error occurred: ${caught.message} \n Exiting program.`;
      }
      return;
    }
  },
);
const displayUpdatePosition = new ScenarioOutput(
  "displayUpdatePosition",
  "6. Update the position of a device in the location tracking system." +
    "The AWS Location Service does not enforce a strict format for deviceId, but it
 must:\n " +
    "- Be a string (case-sensitive).\n" +
    "- Be 1-100 characters long.\n" +
    "- Contain only: Alphanumeric characters (A-Z, a-z, 0-9); Underscores (_);
 Hyphens (-); and be the same ID used when sending and retrieving positions.",
);

const sdkUpdatePosition = new ScenarioAction(
  "sdkUpdatePosition",
  async (/** @type {State} */ state) => {
    // Updates the position of a device in the location tracking system.

    const updateDevicePosParams = {
```

```
      TrackerName: `${data.inputs.trackerName}`,
      Updates: [
        {
          DeviceId: `${data.inputs.deviceId}`,
          SampleTime: new Date(),
          Position: [-122.4194, 37.7749],
        },
      ],
    };
    try {
      const command = new BatchUpdateDevicePositionCommand(
        updateDevicePosParams,
      );
      const response = await locationClient.send(command);
      console.log(
        `Device with id ${data.inputs.deviceId} was successfully updated in the
 location tracking system. `,
      );
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `A validation error occurred while updating the device: ${caught.message}
 \n Exiting program.`,
        );
      }
    }
  },
);
const displayRetrievePosition = new ScenarioOutput(
  "displayRetrievePosition",
  "7. Retrieve the most recent position update for a specified device.",
);

const sdkRetrievePosition = new ScenarioAction(
  "sdkRetrievePosition",
  async (/** @type {State} */ state) => {
    const devicePositionParams = {
      TrackerName: `${data.inputs.trackerName}`,
      DeviceId: `${data.inputs.deviceId}`,
    };
    try {
      const command = new GetDevicePositionCommand(devicePositionParams);
      const response = await locationClient.send(command);
      state.position = response.Position;
```

```
      console.log("Successfully fetched device position: : ", state.position);
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `"The resource was not found: ${caught.message} \n Exiting program.`,
        );
      } else {
        `An unexpected error error occurred: ${caught.message} \n Exiting program.`;
      }
      return;
    }
  },
);
const displayCreateRouteCalc = new ScenarioOutput(
  "displayCreateRouteCalc",
  "8. Create a route calculator.",
);

const sdkCreateRouteCalc = new ScenarioAction(
  "sdkCreateRouteCalc",
  async (/** @type {State} */ state) => {
    const routeCalcParams = {
      CalculatorName: `${data.inputs.calculatorName}`,
      DataSource: "Esri",
    };
    try {
      // Creates a new route calculator with the specified name and data source.
      const command = new CreateRouteCalculatorCommand(routeCalcParams);
      const response = await locationClient.send(command);
      state.CalculatorName = response.CalculatorName;
      console.log(
        "Route calculator created successfully. Calculator name is: ",
        state.CalculatorName,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
```

```
const displayDetermineDist = new ScenarioOutput(
  "displayDetermineDist",
  "9. Determine the distance between Seattle and Vancouver using the route
 calculator.",
);

const sdkDetermineDist = new ScenarioAction(
  "sdkDetermineDist",
  async (/** @type {State} */ state) => {
    // Calculates the distance between two locations asynchronously.
    const determineDist = {
      CalculatorName: `${data.inputs.calculatorName}`,
      DeparturePosition: [-122.3321, 47.6062],
      DestinationPosition: [-123.1216, 49.2827],
      TravelMode: "Car",
      DistanceUnit: "Kilometers",
    };
    try {
      const command = new CalculateRouteCommand(determineDist);
      const response = await locationClient.send(command);

      console.log(
        "Successfully calculated route. The distance in kilometers is : ",
        response.Summary.Distance,
      );
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `Failed to calculate route: ${caught.message} \n Exiting program.`,
        );
      }
      return;
    }
  },
);
const displayUseGeoPlacesClient = new ScenarioOutput(
  "displayUseGeoPlacesClient",
  "10. Use the GeoPlacesAsyncClient to perform additional operations. " +
    "This scenario will show use of the GeoPlacesClient that enables" +
    "location search and geocoding capabilities for your applications. " +
    "We are going to use this client to perform these AWS Location tasks: \n" +
    " - Reverse Geocoding (reverseGeocode): Converts geographic coordinates into
 addresses.\n " +
    " - Place Search (searchText): Finds places based on search queries.\n " +
```

```
    " - Nearby Search (searchNearby): Finds places near a specific location.\n " +
    "First we will perform a Reverse Geocoding operation",
);

const sdkUseGeoPlacesClient = new ScenarioAction(
  "sdkUseGeoPlacesClient",
  async (/** @type {State} */ state) => {
    const geoPlacesClient = new GeoPlacesClient({ region: region });

    const reverseGeoCodeParams = {
      QueryPosition: [-122.4194, 37.7749],
    };
    const searchTextParams = {
      QueryText: "coffee shop",
      BiasPosition: [-122.4194, 37.7749], //San Fransisco
    };
    const searchNearbyParams = {
      QueryPosition: [-122.4194, 37.7749],
      QueryRadius: Number("1000"),
    };
    try {
      /*   Performs reverse geocoding using the AWS Geo Places API.
     Reverse geocoding is the process of converting geographic coordinates (latitude
 and longitude) to a human-readable address.
     This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.*/

      console.log("Use latitude 37.7749 and longitude -122.4194.");
      const command = new ReverseGeocodeCommand(reverseGeoCodeParams);
      const response = await geoPlacesClient.send(command);
      console.log(
        "Successfully calculated route. The distance in kilometers is : ",
        response.ResultItems[0].Distance,
      );
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
    try {
      console.log(
```

```
        "Now we are going to perform a text search using coffee shop",
      );

      /*Searches for a place using the provided search query and prints the detailed
information of the first result.
 @param searchTextParams the search query to be used for the place search (ex,
coffee shop)*/

      const command = new SearchTextCommand(searchTextParams);
      const response = await geoPlacesClient.send(command);
      const placeId = response.ResultItems[0].PlaceId.toString();
      const getPlaceCommand = new GetPlaceCommand({
        PlaceId: placeId,
      });
      const getPlaceResponse = await geoPlacesClient.send(getPlaceCommand);
      console.log(
        `Detailed Place Information: \n Name and address:
${getPlaceResponse.Address.Label}`,
      );

      const foodTypes = getPlaceResponse.FoodTypes;
      if (foodTypes.length) {
        console.log("Food Types: ");
        for (const foodType of foodTypes) {
          console.log("- ", foodType.LocalizedName);
        }
      } else {
        console.log("No food types available.");
      }
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
    try {
      console.log("\nNow we are going to perform a nearby search.");
      const command = new SearchNearbyCommand(searchNearbyParams);
      const response = await geoPlacesClient.send(command);
      const resultItems = response.ResultItems;
      console.log("\nSuccessfully performed nearby search.");
      for (const resultItem of resultItems) {
```

```
        console.log("Name and address: ", resultItem.Address.Label);
        console.log("Distance: ", resultItem.Distance);
      }
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);

const displayDeleteResources = new ScenarioOutput(
  "displayDeleteResources",
  "11. Delete the AWS Location Services resources. " +
    "Would you like to delete the AWS Location Services resources? (y/n)",
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    const deleteGeofenceCollParams = {
      CollectionName: `${state.CollectionName}`,
    };
    const deleteRouteCalculatorParams = {
      CalculatorName: `${state.CalculatorName}`,
    };
    const deleteTrackerParams = { TrackerName: `${state.trackerName}` };
    const deleteMapParams = { MapName: `${state.MapName}` };
    try {
      const command = new DeleteMapCommand(deleteMapParams);
      const response = await locationClient.send(command);
      console.log("Map deleted.");
    } catch (error) {
      console.log("Error deleting map: ", error);
    }
    try {
      const command = new DeleteGeofenceCollectionCommand(
        deleteGeofenceCollParams,
      );
      const response = await locationClient.send(command);
      console.log("Geofence collection deleted.");
```

```
    } catch (error) {
      console.log("Error deleting geofence collection: ", error);
    }
    try {
      const command = new DeleteRouteCalculatorCommand(
        deleteRouteCalculatorParams,
      );
      const response = await locationClient.send(command);
      console.log("Route calculator deleted.");
    } catch (error) {
      console.log("Error deleting route calculator: ", error);
    }
    try {
      const command = new DeleteTrackerCommand(deleteTrackerParams);
      const response = await locationClient.send(command);
      console.log("Tracker deleted.");
    } catch (error) {
      console.log("Error deleting tracker: ", error);
    }
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Location Service Use demo. We hope you " +
    "learned something new, or got some inspiration for your own apps today!" +
    " For more Amazon Location Services examples in different programming languages,
 have a look at: " +
    "https://docs.aws.amazon.com/code-library/latest/ug/
location_code_examples.html",
);

const myScenario = new Scenario("Location Services Scenario", [
  greet,
  pressEnter,
  displayCreateAMap,
  sdkCreateAMap,
  pressEnter,
  displayMapUrl,
  sdkDisplayMapUrl,
  pressEnter,
  displayCreateGeoFenceColl,
  sdkCreateGeoFenceColl,
  pressEnter,
```

```
  displayStoreGeometry,
  sdkStoreGeometry,
  pressEnter,
  displayCreateTracker,
  sdkCreateTracker,
  pressEnter,
  displayUpdatePosition,
  sdkUpdatePosition,
  pressEnter,
  displayRetrievePosition,
  sdkRetrievePosition,
  pressEnter,
  displayCreateRouteCalc,
  sdkCreateRouteCalc,
  pressEnter,
  displayDetermineDist,
  sdkDetermineDist,
  pressEnter,
  displayUseGeoPlacesClient,
  sdkUseGeoPlacesClient,
  pressEnter,
  displayDeleteResources,
  pressEnterConfirm,
  sdkDeleteResources,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

Ações

`BatchUpdateDevicePosition`

O código de exemplo a seguir mostra como usar `BatchUpdateDevicePosition`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
```

```
  BatchUpdateDevicePositionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });
const updateDevicePosParams = {
  TrackerName: `${data.inputs.trackerName}`,
  Updates: [
    {
      DeviceId: `${data.inputs.deviceId}`,
      SampleTime: new Date(),
      Position: [-122.4194, 37.7749],
    },
  ],
};
export const main = async () => {
  try {
    const command = new BatchUpdateDevicePositionCommand(updateDevicePosParams);
    const response = await locationClient.send(command);
    //console.log("response ", response.Errors[0].Error);

    console.log(
      `Device with id ${data.inputs.deviceId} was successfully updated in the
 location tracking system. `,
      response,
    );
  } catch (error) {
    console.log("error ", error);
  }
};
```

- Para obter detalhes da API, consulte BatchUpdateDevicePosition em Referência AWS SDK para JavaScript da API.

`CalculateRoute`

O código de exemplo a seguir mostra como usar `CalculateRoute`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  CalculateRouteCommand,
  ResourceNotFoundException,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });

export const main = async () => {
  const routeCalcParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
    DeparturePosition: [-122.3321, 47.6062],
    DestinationPosition: [-123.1216, 49.2827],
    TravelMode: "Car",
    DistanceUnit: "Kilometers",
  };
  try {
    const command = new CalculateRouteCommand(routeCalcParams);
    const response = await locationClient.send(command);

    console.log(
      "Successfully calculated route. The distance in kilometers is : ",
      response.Summary.Distance,
    );
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `An conflict occurred: ${caught.message} \n Exiting program.`,
      );
      return;
    }
```

```
  }
};
```

- Para obter detalhes da API, consulte CalculateRoute em Referência AWS SDK para JavaScript da API.

`CreateGeofenceCollection`

O código de exemplo a seguir mostra como usar `CreateGeofenceCollection`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  ConflictException,
  CreateGeofenceCollectionCommand,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const geoFenceCollParams = {
    CollectionName: `${data.inputs.collectionName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateGeofenceCollectionCommand(geoFenceCollParams);
    const response = await locationClient.send(command);
    console.log(
      "Collection created. Collection name is: ",
      response.CollectionName,
```

```
    );
  } catch (caught) {
    if (caught instanceof ConflictException) {
      console.error("A conflict occurred. Exiting program.");
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateGeofenceCollection em Referência AWS SDK para JavaScript da API.

`CreateMap`

O código de exemplo a seguir mostra como usar `CreateMap`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import { CreateMapCommand, LocationClient } from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const CreateMapCommandInput = {
    MapName: `${data.inputs.mapName}`,
    Configuration: { style: "VectorEsriNavigation" },
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateMapCommand(CreateMapCommandInput);
    const response = await locationClient.send(command);
```

```
    console.log("Map created. Map ARN is : ", response.MapArn);
  } catch (error) {
    console.error("Error creating map: ", error);
    throw error;
  }
};
```

- Para obter detalhes da API, consulte CreateMap em Referência AWS SDK para JavaScript da API.

`CreateRouteCalculator`

O código de exemplo a seguir mostra como usar `CreateRouteCalculator`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  ConflictException,
  CreateRouteCalculatorCommand,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });

export const main = async () => {
  const routeCalcParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
    DataSource: "Esri",
  };
  try {
```

```
    const command = new CreateRouteCalculatorCommand(routeCalcParams);
    const response = await locationClient.send(command);

    console.log(
      "Route calculator created successfully. Calculator name is ",
      response.CalculatorName,
    );
  } catch (caught) {
    if (caught instanceof ConflictException) {
      console.error(
        `An conflict occurred: ${caught.message} \n Exiting program.`,
      );
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateRouteCalculator em Referência AWS SDK para JavaScript da API.

`CreateTracker`

O código de exemplo a seguir mostra como usar `CreateTracker`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import { CreateTrackerCommand, LocationClient } from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
```

```
  const createTrackerParams = {
    TrackerName: `${data.inputs.trackerName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateTrackerCommand(createTrackerParams);
    const response = await locationClient.send(command);
    //state.trackerName - response.TrackerName;
    console.log("Tracker created. Tracker name is : ", response.TrackerName);
  } catch (error) {
    console.error("Error creating map: ", error);
    throw error;
  }
};
```

- Para obter detalhes da API, consulte CreateTracker em Referência AWS SDK para JavaScript da API.

`DeleteGeofenceCollection`

O código de exemplo a seguir mostra como usar `DeleteGeofenceCollection`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  DeleteGeofenceCollectionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
```

```
export const main = async () => {
  const deleteGeofenceCollParams = {
    CollectionName: `${data.inputs.collectionName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new DeleteGeofenceCollectionCommand(
      deleteGeofenceCollParams,
    );
    const response = await locationClient.send(command);
    console.log("Collection deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `${data.inputs.collectionName} Geofence collection not found.`,
      );
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteGeofenceCollection em Referência AWS SDK para JavaScript da API.

`DeleteMap`

O código de exemplo a seguir mostra como usar `DeleteMap`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
```

```
  DeleteMapCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteMapParams = {
    MapName: `${data.inputs.mapName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteMapCommand(deleteMapParams);
    const response = await locationClient.send(command);
    console.log("Map deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(`${data.inputs.mapName} map not found.`);
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteMap em Referência AWS SDK para JavaScript da API.

`DeleteRouteCalculator`

O código de exemplo a seguir mostra como usar `DeleteRouteCalculator`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  DeleteRouteCalculatorCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteRouteCalculatorParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteRouteCalculatorCommand(
      deleteRouteCalculatorParams,
    );
    const response = await locationClient.send(command);
    console.log("Route calculator deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `${data.inputs.calculatorName} route calculator not found.`,
      );
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteRouteCalculator em Referência AWS SDK para JavaScript da API.

`DeleteTracker`

O código de exemplo a seguir mostra como usar `DeleteTracker`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  DeleteTrackerCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteTrackerParams = {
    TrackerName: `${data.inputs.trackerName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteTrackerCommand(deleteTrackerParams);
    const response = await locationClient.send(command);
    console.log("Tracker deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(`${data.inputs.trackerName} tracker not found.`);
      return;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteTracker em Referência AWS SDK para JavaScript da API.

`GetDevicePosition`

O código de exemplo a seguir mostra como usar `GetDevicePosition`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  GetDevicePositionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const locationClient = new LocationClient({ region: region });
  const deviceId = `${data.inputs.deviceId}`;
  const trackerName = `${data.inputs.trackerName}`;

  const devicePositionParams = {
    DeviceId: deviceId,
    TrackerName: trackerName,
  };
  try {
    const command = new GetDevicePositionCommand(devicePositionParams);
    const response = await locationClient.send(command);
    //state.position = response.position;
    console.log("Successfully fetched device position: ", response);
  } catch (error) {
    console.log("Error ", error);
    /*  if (caught instanceof ResourceNotFoundException) {
      console.error(
        `"The resource was not found: ${caught.message} \n Exiting program.`,
      );
    } else {
```

```
      `An unexpected error error occurred: ${caught.message} \n Exiting program.`;
    }
    return;*/
  }
};
```

- Para obter detalhes da API, consulte GetDevicePosition em Referência AWS SDK para JavaScript da API.

`PutGeofence`

O código de exemplo a seguir mostra como usar `PutGeofence`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { fileURLToPath } from "node:url";
import {
  PutGeofenceCommand,
  LocationClient,
  ValidationException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });
export const main = async () => {
  const geoFenceGeoParams = {
    CollectionName: `${data.inputs.collectionName}`,
    GeofenceId: `${data.inputs.geoId}`,
    Geometry: {
      Polygon: [
        [
          [-122.3381, 47.6101],
```

```
          [-122.3281, 47.6101],
          [-122.3281, 47.6201],
          [-122.3381, 47.6201],
          [-122.3381, 47.6101],
        ],
      ],
    },
  };
  try {
    const command = new PutGeofenceCommand(geoFenceGeoParams);
    const response = await locationClient.send(command);
    console.log("GeoFence created. GeoFence ID is: ", response.GeofenceId);
  } catch (error) {
    console.error(
      `A validation error occurred while creating geofence: ${error} \n Exiting
 program.`,
    );
    return;
  }
};
```

- Para obter detalhes da API, consulte PutGeofence em Referência AWS SDK para JavaScript da API.

AWS Marketplace Exemplos de API de contrato usando SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS SDK para JavaScript (v3) with AWS Marketplace Agreement.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Contratos

Contratos

Aceitar uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como aceitar uma solicitação de cancelamento de contrato iniciada pelo vendedor.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    AcceptAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";

async function acceptAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new AcceptAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Description: " + response.description);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
```

```
    console.log("Updated At: " + response.updatedAt);
}

acceptAgreementCancellationRequest();
```

- Para obter detalhes da API, consulte AcceptAgreementCancellationRequest em Referência AWS SDK para JavaScript da API.

Aceitar uma solicitação de pagamento por contrato

O exemplo de código a seguir mostra como aceitar uma solicitação de pagamento por contrato iniciada pelo vendedor.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    AcceptAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
const PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER REFERENCE HERE>";

async function acceptAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new AcceptAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
            purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
        })
```

```
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Name: " + response.name);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

acceptAgreementPaymentRequest();
```

- Para obter detalhes da API, consulte AcceptAgreementPaymentRequest em Referência AWS SDK para JavaScript da API.

Alterar um termo de renovação do contrato de SaaS

O exemplo de código a seguir mostra como alterar um contrato de SaaS para adicionar ou atualizar seu prazo de renovação.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
```

```
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model and then
 turn on
 * the auto-renewal setting using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product using a public offer that supports
 * auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
 * auto-renewal via the RenewalTerm configuration.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P1M" for 1 month).
 *   - DIMENSION_1_KEY — the dimension key defined in the offer.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
const SELECTOR_VALUE = "<your-selector-value>";

// The dimension key defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-key>";

// Quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the RenewalTerm in your offer.
const RENEWAL_TERM_ID = "<your-renewal-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the SupportTerm in your offer.
const SUPPORT_TERM_ID = "<your-support-term-id>";

/**
```

```
 * Full end-to-end flow:
 * 1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
 disabled.
 * 2. Wait for entitlements to become active.
 * 3. Amend the agreement to enable auto-renewal.
 */
async function amendSaaSContractAgreementRenewalTerm() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                ],
            },
        },
    };

    // Initial agreement: auto-renewal disabled.
    const renewalTerm = {
        id: RENEWAL_TERM_ID,
        configuration: {
            renewalTermConfiguration: {
                enableAutoRenew: false,
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };
    const supportTerm = { id: SUPPORT_TERM_ID };

    // --- Create and accept the initial SaaS agreement request with CONTRACT
 pricing model ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, renewalTerm, legalTerm,
 supportTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
```

```
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId);

    // Wait for entitlements to become active before amending.
    console.log("Waiting for entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
    console.log("Entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Amend: enable auto-renewal ---
    const renewalTermAmended = {
        id: RENEWAL_TERM_ID,
        configuration: {
            renewalTermConfiguration: {
                enableAutoRenew: true,
            },
        },
    };

    // Use Intent.AMEND and sourceAgreementIdentifier to target the existing
agreement.
    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "AMEND",
            requestedTerms: [configurableUpfrontPricingTerm, renewalTermAmended,
legalTerm, supportTerm],
            sourceAgreementIdentifier: acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Amend agreement request created. AgreementRequestId: " +
carResponse.agreementRequestId);
```

```
    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    console.log("Amendment accepted. Auto-renewal enabled. New AgreementId: " +
 aarResponse.agreementId);
}

amendSaaSContractAgreementRenewalTerm();
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK para JavaScript da API.

Alterar uma AMI ConfigurableUpfrontPricingTerm para o modelo de preços de uso

O exemplo de código a seguir mostra como alterar um contrato de AMI ConfigurableUpfrontPricingTerm atualizando a quantidade da dimensão para o modelo de preços de uso.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create an AMI agreement with ConfigurableUpfrontPricingTerm
 and then amend the dimension quantity
```

```
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: An AMI product with USAGE pricing requires two agreements:
 *   1. An agreement with usageBasedPricingTerm (UBPT) — accepted first to establish
 the base agreement.
 *   2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
 UBPT agreement entitlements are active.
 *
 * Once both agreement entitlements are available, this sample shows how to amend
 the agreement
 * with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P365D" for one year).
 *   - DIMENSION_1_KEY — the dimension key defined in the offer (e.g., instance
 type).
 *   - DIMENSION_1_VALUE — initial quantity; NEW_DIMENSION_1_VALUE — amended
 quantity.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P365D" for 365 days).
const SELECTOR_VALUE = "<your-selector-value>";

// The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
const DIMENSION_1_KEY = "<your-dimension-key>";

// Initial quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the UsageBasedPricingTerm in your offer.
const USAGE_TERM_ID = "<your-usage-term-id>";
```

```
// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in your offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// New quantity to use when amending the dimension of CUPT.
const NEW_DIMENSION_1_VALUE = 5;

/**
 * Full end-to-end flow:
 * 1. Create and accept an agreement request with usageBasedPricingTerm.
 * 2. Wait for entitlements to become active, then create and accept an agreement
 request with configurableUpfrontPricingTerm (CUPT).
 * 3. Wait for CUPT entitlements to become active, then amend the dimension
 quantity.
 */
async function amendAmiCUPTAgreement() {
    const client = new MarketplaceAgreementClient();

    const usageTerm = { id: USAGE_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Agreement with UBPT ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageTerm, legalTerm, validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with UBPT accepted. AgreementId: " +
 acceptAgreementRequestResponse.agreementId);
```

```
   // Wait for entitlements to become active before creating the agreement with
CUPT.
   console.log("Waiting for UBPT agreement entitlements to become active...");
   const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
   console.log("UBPT agreement entitlements are now active.");
   formatOutput(entitlementsResponse);

   // --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
   const configurableUpfrontPricingTerm = {
       id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
       configuration: {
           configurableUpfrontPricingTermConfiguration: {
               selectorValue: SELECTOR_VALUE,
               dimensions: [
                   { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
               ],
           },
       },
   };

   const carResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "NEW",
           requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
           agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
       })
   );
   console.log("Agreement request with CUPT created. AgreementRequestId: " +
carResponse.agreementRequestId);

   const aarResponse = await client.send(
       new AcceptAgreementRequestCommand({
           agreementRequestId: carResponse.agreementRequestId,
       })
   );
   const cuptAgreementId = aarResponse.agreementId;
   console.log("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

   // Wait for entitlements to become active before amending.
```

```
    console.log("Waiting for CUPT agreement entitlements to become active...");
    const cuptEntitlementsResponse = await pollUntilEntitlementsAvailable(client,
 cuptAgreementId);
    console.log("CUPT agreement entitlements are now active.");
    formatOutput(cuptEntitlementsResponse);

    // --- Amend Agreement with CUPT ---
    // Increase the dimension quantity using Intent.AMEND and
 sourceAgreementIdentifier.
    const newConfig = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 NEW_DIMENSION_1_VALUE }, // Increase quantity for this dimension key
                ],
            },
        },
    };

    const carAmendResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "AMEND",
            requestedTerms: [newConfig, legalTerm, validityTerm],
            sourceAgreementIdentifier: cuptAgreementId,
        })
    );
    console.log("Amendment of CUPT agreement request created. AgreementRequestId: "
 + carAmendResponse.agreementRequestId);

    const aarAmendResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carAmendResponse.agreementRequestId,
        })
    );
    console.log("Amendment accepted. New AgreementId: " +
 aarAmendResponse.agreementId);
}

amendAmiCUPTAgreement();
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK para JavaScript da API.

Crie um novo contrato de teste gratuito da AMI

O exemplo de código a seguir mostra como criar um novo contrato de teste gratuito da AMI.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create an AMI Free Trial agreement
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to an AMI product that offers a free trial period.
 * The free trial includes a FreeTrialPricingTerm alongside a UsageBasedPricingTerm.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";
```

```
// Term ID for the FreeTrialPricingTerm in your offer.
const FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>";

// Term ID for the UsageBasedPricingTerm in your offer (applies after the trial
 ends).
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the SupportTerm in your offer.
const SUPPORT_TERM_ID = "<your-support-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

/**
 * Creates an AMI Free Trial agreement.
 * The FreeTrialPricingTerm grants access at no cost for the trial period.
 * The UsageBasedPricingTerm defines the charges that apply once the trial ends.
 */
async function createAndAcceptAmiFreeTrialAgreementRequest() {
    const client = new MarketplaceAgreementClient();

    const freeTrialPricingTerm = { id: FREE_TRIAL_PRICING_TERM_ID };
    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const supportTerm = { id: SUPPORT_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };

    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [freeTrialPricingTerm, usageBasedPricingTerm,
 supportTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
```

```
    console.log("Agreement request with freeTrialPricingTerm accepted. AgreementId:
 " + acceptAgreementRequestResponse.agreementId);
}

createAndAcceptAmiFreeTrialAgreementRequest();
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK para JavaScript da API.

Crie um novo contrato de SaaS com pagamento adiantado

O exemplo de código a seguir mostra como criar um novo contrato de SaaS com pagamento adiantado.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
 * selecting an agreement duration and specifying quantities for multiple
 dimensions.
 * Tax estimation is enabled at the time of agreement creation.
```

```
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P12M" for 12 months).
 *   - DIMENSION_1_KEY, DIMENSION_2_KEY — dimension keys defined in the offer.
 *   - DIMENSION_1_VALUE, DIMENSION_2_VALUE — quantities for each dimension.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P12M" for 12 months).
const SELECTOR_VALUE = "<your-selector-value>";

// First dimension key and quantity defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-1-key>";
const DIMENSION_1_VALUE = 10;

// Second dimension key and quantity defined in your offer.
const DIMENSION_2_KEY = "<your-dimension-2-key>";
const DIMENSION_2_VALUE = 20;

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Tax estimation setting: "ENABLED" to include estimated taxes in the agreement.
const TAX_ESTIMATION = "ENABLED";

/**
 * Creates a SaaS agreement with CONTRACT pricing model with configurable upfront
 pricing
 * for multiple dimensions and tax estimation.
 */
async function createSaaSContractAgreement() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
```

```
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                    { dimensionKey: DIMENSION_2_KEY, dimensionValue:
 DIMENSION_2_VALUE },
                ],
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };

    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm],
            taxConfiguration: { taxEstimation: TAX_ESTIMATION },
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("SaaS agreement request with CONTRACT pricing model accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId);
}

createSaaSContractAgreement();
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK para JavaScript da API.

Obtenha detalhes de uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como obter detalhes de uma solicitação específica de cancelamento de contrato.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    GetAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";

async function getAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new GetAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}
```

```
getAgreementCancellationRequest();
```

- Para obter detalhes da API, consulte GetAgreementCancellationRequest em Referência AWS SDK para JavaScript da API.

Obtenha detalhes de uma solicitação de pagamento por contrato

O exemplo de código a seguir mostra como obter detalhes de uma solicitação de pagamento de um contrato específico.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    GetAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";

async function getAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new GetAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
```

```
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Name: " + response.name);
    console.log("Charge ID: " + response.chargeId);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

getAgreementPaymentRequest();
```

- Para obter detalhes da API, consulte GetAgreementPaymentRequest em Referência AWS SDK para JavaScript da API.

Listar solicitações de cancelamento de contratos

O exemplo de código a seguir mostra como listar solicitações de cancelamento de contratos dos quais participo como aceitante.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    ListAgreementCancellationRequestsCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const PARTY_TYPE = "Proposer";

async function listAgreementCancellationRequests() {
    const client = new MarketplaceAgreementClient();

    let nextToken = null;
```

```
    do {
        const response = await client.send(
            new ListAgreementCancellationRequestsCommand({
                partyType: PARTY_TYPE,
                nextToken: nextToken,
            })
        );

        for (const summary of response.items) {
            console.log("Cancellation Request ID: " +
 summary.agreementCancellationRequestId);
            console.log("Agreement ID: " + summary.agreementId);
            console.log("Status: " + summary.status);
            console.log("Reason Code: " + summary.reasonCode);
            console.log("Agreement Type: " + summary.agreementType);
            console.log("Catalog: " + summary.catalog);
            console.log("Created At: " + summary.createdAt);
            console.log("Updated At: " + summary.updatedAt);
            console.log("---");
        }

        nextToken = response.nextToken;
    } while (nextToken != null);
}

listAgreementCancellationRequests();
```

- Para obter detalhes da API, consulte ListAgreementCancellationRequests em Referência AWS SDK para JavaScript da API.

Listar solicitações de pagamento de contratos

O exemplo de código a seguir mostra como listar solicitações de pagamento de contratos dos quais participo como aceitante.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    ListAgreementPaymentRequestsCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const PARTY_TYPE = "Proposer";

async function listAgreementPaymentRequests() {
    const client = new MarketplaceAgreementClient();

    let nextToken = null;

    do {
        const response = await client.send(
            new ListAgreementPaymentRequestsCommand({
                partyType: PARTY_TYPE,
                nextToken: nextToken,
            })
        );

        for (const summary of response.items) {
            console.log("Payment Request ID: " + summary.paymentRequestId);
            console.log("Agreement ID: " + summary.agreementId);
            console.log("Status: " + summary.status);
            console.log("Name: " + summary.name);
            console.log("Charge ID: " + summary.chargeId);
            console.log("Charge Amount: " + summary.chargeAmount);
            console.log("Currency Code: " + summary.currencyCode);
            console.log("Created At: " + summary.createdAt);
            console.log("Updated At: " + summary.updatedAt);
            console.log("---");
        }
```

```
        nextToken = response.nextToken;
    } while (nextToken != null);
}

listAgreementPaymentRequests();
```

- Para obter detalhes da API, consulte ListAgreementPaymentRequests em Referência AWS SDK para JavaScript da API.

Rejeitar uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como rejeitar uma solicitação de cancelamento de contrato iniciada pelo vendedor.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    RejectAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";
const REJECTION_REASON = "<REJECTION REASON HERE>";

async function rejectAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new RejectAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
```

```
            rejectionReason: REJECTION_REASON,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

rejectAgreementCancellationRequest();
```

- Para obter detalhes da API, consulte RejectAgreementCancellationRequest em Referência AWS SDK para JavaScript da API.

Rejeitar uma solicitação de pagamento de contrato

O exemplo de código a seguir mostra como rejeitar uma solicitação de pagamento de contrato iniciada pelo vendedor.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    RejectAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
```

```
const REJECTION_REASON = "<REJECTION REASON HERE>";

async function rejectAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new RejectAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
            rejectionReason: REJECTION_REASON,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Name: " + response.name);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

rejectAgreementPaymentRequest();
```

- Para obter detalhes da API, consulte RejectAgreementPaymentRequest em Referência AWS SDK para JavaScript da API.

Substitua um contrato de SaaS por uma oferta baseada em contrato

O exemplo de código a seguir mostra como substituir um contrato de SaaS por uma nova oferta baseada em contrato.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
 * Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
 * Agreement-Based Offer from the seller. The buyer replaces the existing agreement
 * with the new ABO offer, which may include updated pricing terms and payment
 schedule.
 *
 * Note: Unlike other Replace samples, this example starts from an already-active
 * EXISTING_AGREEMENT_ID rather than creating a new agreement first. Use this
 * pattern when you already have an agreement ID and want to replace it directly.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offers:
 *   - EXISTING_AGREEMENT_ID — the agreement ID of the active agreement to replace
 (starts with "agmt-").
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new ABO
 offer.
 *   - Term IDs (starting with "term-") — found in the new offer's term list.
 */
```

```
// The agreement ID of the active SaaS agreement with CONTRACT pricing model to
 replace (starts with "agmt-").
const EXISTING_AGREEMENT_ID = "<your-existing-agreement-id>";

// The agreementProposalId from the new Agreement-Based Offer.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the LegalTerm in the new offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in the new offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the FixedUpfrontPricingTerm in the new offer.
const FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>";

// Term ID for the PaymentScheduleTerm in the new offer.
const PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>";

/**
 * Replaces an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.
 * Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing agreement
 ID.
 */
async function replaceExistingAgreement() {
    const client = new MarketplaceAgreementClient();

    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const fixedUpfrontPricingTerm = { id: FIXED_UPFRONT_PRICING_TERM_ID };
    const paymentScheduleTerm = { id: PAYMENT_SCHEDULE_TERM_ID };

    // Replace the agreement with the new offer
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [fixedUpfrontPricingTerm, paymentScheduleTerm,
 legalTerm, validityTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: EXISTING_AGREEMENT_ID,
        })
```

```
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement replaced with ABO. New AgreementId: " +
 acceptAgreementRequestResponse.agreementId);
}

replaceExistingAgreement();
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK para JavaScript da API.

Substitua um teste gratuito de SaaS por um contrato com preços de consumo

O exemplo de código a seguir mostra como substituir um teste gratuito de SaaS por um Contrato com Preços de Consumo (CCP).

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");
```

```
/**
 * Demonstrates how to create a SaaS free trial agreement and then replace it with a
 * paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.
 *
 * Scenario: A buyer first starts a free trial on a SaaS product. Once the trial is
 active,
 * they decide to convert by replacing it with a paid offer that includes a
 * FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.
 *
 * Flow:
 *   1. Create a SaaS free trial agreement with freeTrialPricingTerm.
 *   2. Wait for freeTrialPricingTerm agreement entitlements to become active.
 *   3. Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the free trial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the paid CCP
 offer.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 */

// The agreementProposalId from the free trial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-agreement-proposal-
identifier>";

// Term ID for the FreeTrialPricingTerm in the free trial offer.
const FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>";

// Term ID for the LegalTerm in the free trial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// The agreementProposalId from the paid CCP offer to convert to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-agreement-proposal-
identifier>";

// Term ID for the FixedUpfrontPricingTerm in the CCP offer.
const FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>";

// Term ID for the PaymentScheduleTerm in the CCP offer.
```

```
const PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>";

// Term ID for the UsageBasedPricingTerm in the CCP offer.
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the CCP offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the LegalTerm in the CCP offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS free trial agreement with freeTrialPricingTerm.
 * 2. Wait for freeTrialPricingTerm agreement entitlements to become active.
 * 3. Replace the free trial agreement with the paid CCP offer.
 */
async function createSaaSFreeTrialAndReplaceWithCCP() {
    const client = new MarketplaceAgreementClient();

    const legalTerm = { id: LEGAL_TERM_ID };
    const freeTrialPricingTerm = { id: FREE_TRIAL_PRICING_TERM_ID };

    // --- Step 1: Agreement with freeTrialPricingTerm ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [freeTrialPricingTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with freeTrialPricingTerm created.
 AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with freeTrialPricingTerm accepted. AgreementId:
 " + acceptAgreementRequestResponse.agreementId);
```

```
    // Wait for freeTrialPricingTerm agreement entitlements to become active before
 replacing.
    console.log("Waiting for freeTrialPricingTerm agreement entitlements to become
 active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
 acceptAgreementRequestResponse.agreementId);
    console.log("freeTrialPricingTerm agreement entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP Offer
 ---
    // Use Intent.REPLACE and sourceAgreementIdentifier to replace the free trial
 agreement.
    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const fixedUpfrontPricingTerm = { id: FIXED_UPFRONT_PRICING_TERM_ID };
    const paymentScheduleTerm = { id: PAYMENT_SCHEDULE_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };

    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [usageBasedPricingTerm, fixedUpfrontPricingTerm,
 paymentScheduleTerm, validityTerm, newLegalTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    console.log("Agreement with freeTrialPricingTerm replaced with paid CCP offer.
 New AgreementId: " + aarResponse.agreementId);
}

createSaaSFreeTrialAndReplaceWithCCP();
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK para JavaScript da API.

Substitua um termo de preço baseado no uso do SaaS e cancele o contrato anterior

O exemplo de código a seguir mostra como substituir um termo de preço baseado no uso do SaaS e cancelar o contrato anterior.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
    CancelAgreementCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT) and
 then replace it
 * with a new offer using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
 * The buyer then converts to a different offer by replacing the existing agreement.
 *
 * Flow:
 *   1. Create and accept the initial agreement request with UBPT.
 *   2. Wait for entitlements to become active.
 *   3. Replace the agreement with a new offer using Intent.REPLACE.
 *   4. Cancel the new agreement using CancelAgreement.
 *
```

```
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 */

// The agreementProposalId from the initial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the UsageBasedPricingTerm in the initial offer.
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the initial offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the LegalTerm in the initial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// The agreementProposalId from the new offer to replace to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the UsageBasedPricingTerm in the new offer.
const NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the new offer.
const NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>";

// Term ID for the LegalTerm in the new offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create and accept the initial agreement request with UsageBasedPricingTerm.
 * 2. Wait for entitlements to become active.
 * 3. Replace the agreement with a new offer.
 * 4. Cancel the new agreement.
 */
async function replaceSaaSUbptAndCancel() {
    const client = new MarketplaceAgreementClient();
```

```
   const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
   const validityTerm = { id: VALIDITY_TERM_ID };
   const legalTerm = { id: LEGAL_TERM_ID };

   // --- Step 1: Create and accept the initial UBPT agreement request ---
   const createAgreementRequestResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "NEW",
           requestedTerms: [usageBasedPricingTerm, validityTerm, legalTerm],
           agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
       })
   );
   console.log("Agreement request with UBPT created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

   const acceptAgreementRequestResponse = await client.send(
       new AcceptAgreementRequestCommand({
           agreementRequestId: createAgreementRequestResponse.agreementRequestId,
       })
   );
   console.log("Agreement request with UBPT accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId);

   // Wait for entitlements to become active before replacing.
   console.log("Waiting for entitlements to become active...");
   const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
   console.log("Entitlements are now active.");
   formatOutput(entitlementsResponse);

   // --- Step 2: Replace the UBPT agreement with a new offer ---
   // Use Intent.REPLACE and sourceAgreementIdentifier to replace the existing
agreement.
   const newUsageBasedPricingTerm = { id: NEW_USAGE_BASED_PRICING_TERM_ID };
   const newValidityTerm = { id: NEW_VALIDITY_TERM_ID };
   const newLegalTerm = { id: NEW_LEGAL_TERM_ID };

   const carResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "REPLACE",
```

```
            requestedTerms: [newUsageBasedPricingTerm, newValidityTerm,
 newLegalTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const agreementIdWithTheNewOffer = aarResponse.agreementId;
    console.log("UBPT agreement replaced. New AgreementId: " +
 agreementIdWithTheNewOffer);

    // --- Step 3: Cancel the new agreement ---
    await client.send(
        new CancelAgreementCommand({
            agreementId: agreementIdWithTheNewOffer,
        })
    );
    console.log("The new agreement has been cancelled. AgreementId: " +
 agreementIdWithTheNewOffer);
}

replaceSaaSUbptAndCancel();
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK para JavaScript da API.

Substitua um termo de preço baseado no uso da AMI, mas mantenha o ConfigurableUpfrontPricingTerm

O exemplo de código a seguir mostra como substituir um termo de preço baseado no uso da AMI e, ao mesmo tempo, manter o existente. ConfigurableUpfrontPricingTerm

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
 * the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active, using
 the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: An AMI product with USAGE pricing requires two agreements:
 *   1. An agreement with usageBasedPricingTerm — accepted first to establish the
 base agreement.
 *   2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
 usageBasedPricingTerm agreement entitlements are active.
 *
 * This sample shows how to replace only the agreement with usageBasedPricingTerm
 with a
 * new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
 *
 * Flow:
 *   1. Create and accept the initial agreement request with usageBasedPricingTerm.
 *   2. Create and accept the agreement request with configurableUpfrontPricingTerm
 (CUPT) (after usageBasedPricingTerm agreement entitlements are active).
 *   3. Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.
 *
```

```
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P365D").
 *   - DIMENSION_1_KEY — dimension key defined in the CUPT term.
 */

// The agreementProposalId from the initial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P365D" for 365 days).
const SELECTOR_VALUE = "<your-selector-value>";

// Dimension key defined in the CUPT term.
const DIMENSION_1_KEY = "<your-dimension-key>";

// Quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the UsageBasedPricingTerm in the initial offer.
const USAGE_TERM_ID = "<your-usage-term-id>";

// Term ID for the LegalTerm in the initial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in the initial offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// The agreementProposalId from the new offer to replace to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the UsageBasedPricingTerm in the new offer.
const NEW_USAGE_TERM_ID = "<your-new-usage-term-id>";
```

```
// Term ID for the LegalTerm in the new offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

// Term ID for the ValidityTerm in the new offer.
const NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create and accept an agreement request with usageBasedPricingTerm, then wait
 for entitlements.
 * 2. Create and accept an agreement request with CUPT, then wait for entitlements.
 * 3. Replace the agreement with usageBasedPricingTerm with a new offer (agreement
 with CUPT is unaffected).
 */
async function replaceAmiUsageWhenCuptExists() {
    const client = new MarketplaceAgreementClient();

    const usageTerm = { id: USAGE_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Step 1: Agreement with UBPT ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageTerm, legalTerm, validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    const usageAgreementId = acceptAgreementRequestResponse.agreementId;
    console.log("Agreement request with UBPT accepted. AgreementId: " +
 usageAgreementId);

    // Wait for UBPT agreement entitlements to become active before creating the
 agreement with CUPT.
```

```
   console.log("Waiting for UBPT agreement entitlements to become active...");
   const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
usageAgreementId);
   console.log("UBPT agreement entitlements are now active.");
   formatOutput(entitlementsResponse);

   // --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
   const configurableUpfrontPricingTerm = {
       id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
       configuration: {
           configurableUpfrontPricingTermConfiguration: {
               selectorValue: SELECTOR_VALUE,
               dimensions: [
                   { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
               ],
           },
       },
   };

   const carResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "NEW",
           requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
           agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
       })
   );
   console.log("Agreement request with CUPT created. AgreementRequestId: " +
carResponse.agreementRequestId);

   const aarResponse = await client.send(
       new AcceptAgreementRequestCommand({
           agreementRequestId: carResponse.agreementRequestId,
       })
   );
   const cuptAgreementId = aarResponse.agreementId;
   console.log("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

   // Wait for CUPT agreement entitlements to become active before replacing usage.
   console.log("Waiting for CUPT agreement entitlements to become active...");
```

```
    const cuptEntitlementsResponse = await pollUntilEntitlementsAvailable(client,
 cuptAgreementId);
    console.log("CUPT agreement entitlements are now active.");
    formatOutput(cuptEntitlementsResponse);

    // --- Step 3: Replace Agreement with usageBasedPricingTerm with a new offer ---
    const newUsageTerm = { id: NEW_USAGE_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };
    const newValidityTerm = { id: NEW_VALIDITY_TERM_ID };

    // Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
 usageBasedPricingTerm agreement only.
    // The agreement with configurableUpfrontPricingTerm (CUPT) is NOT affected by
 this replacement.
    const carReplaceResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [newUsageTerm, newLegalTerm, newValidityTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: usageAgreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carReplaceResponse.agreementRequestId);

    const aarReplaceResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carReplaceResponse.agreementRequestId,
        })
    );
    console.log("Agreement with UBPT replaced. New AgreementId: " +
 aarReplaceResponse.agreementId);
}

replaceAmiUsageWhenCuptExists();
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK para JavaScript da API.

Atualizar pedidos de compra após a aceitação do contrato

O exemplo de código a seguir mostra como atualizar pedidos de compra após a aceitação do contrato.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
    ListAgreementChargesCommand,
    UpdatePurchaseOrdersCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT pricing model
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model
 and provides a purchase order
 * reference in AcceptAgreementRequest. After acceptance, the buyer lists the
 resulting
 * charges via ListAgreementCharges and associates the purchase order reference with
 a
 * specific charge via UpdatePurchaseOrders.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement.
```

```
 *   - DIMENSION_1_KEY — dimension key defined in the offer.
 *   - PURCHASE_ORDER_REFERENCE — your internal purchase order number (e.g.,
 "po-123456").
 */

// Your internal purchase order reference number (e.g., "po-123456").
const PURCHASE_ORDER_REFERENCE = "po-123456";

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P366D" for 366 days).
const SELECTOR_VALUE = "<your-selector-value>";

// Dimension key and quantity defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-key>";
const DIMENSION_1_VALUE = 1;

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in your offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS agreement with CONTRACT pricing model with a purchase order
 reference.
 * 2. List charges to retrieve charge IDs and revisions.
 * 3. Associate the purchase order reference with a specific charge via
 UpdatePurchaseOrders.
 * 4. List charges again to confirm the update.
 */
async function listAgreementChargesAndUpdatePurchaseOrders() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
```

```
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Create Agreement ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

    // --- Accept Agreement Request with Purchase Order ---
    // The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
    const chargeId =
createAgreementRequestResponse.chargeSummary.expectedCharges[0].id;
    const purchaseOrderAtAcceptance = {
        chargeId: chargeId,
        purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
    };
    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
            purchaseOrders: [purchaseOrderAtAcceptance],
        })
    );
    const agreementId = acceptAgreementRequestResponse.agreementId;
    console.log("Agreement request accepted with purchase order reference '" +
PURCHASE_ORDER_REFERENCE
        + "'. AgreementId: " + agreementId);
```

```
    // --- List Agreement Charges ---
    const listAgreementChargesResponse = await client.send(
        new ListAgreementChargesCommand({
            agreementId: agreementId,
        })
    );

    console.log("All charges for agreement " + agreementId + ":");
    formatOutput(listAgreementChargesResponse);

    // --- Update Purchase Order ---
    const firstCharge = listAgreementChargesResponse.items[0];
    const purchaseOrder = {
        agreementId: agreementId,
        purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
        chargeRevision: firstCharge.revision,
        chargeId: firstCharge.id,
    };
    await client.send(
        new UpdatePurchaseOrdersCommand({
            purchaseOrders: [purchaseOrder],
        })
    );
    console.log("Purchase order reference '" + PURCHASE_ORDER_REFERENCE
        + "' updated for ChargeId: " + firstCharge.id);

    // --- Verify Update ---
    const lacResponse = await client.send(
        new ListAgreementChargesCommand({
            agreementId: agreementId,
        })
    );
    console.log("Verified updated charge:");
    formatOutput(lacResponse.items[0]);
}

listAgreementChargesAndUpdatePurchaseOrders();
```

- Para obter detalhes da API, consulte UpdatePurchaseOrders em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon MSK usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon MSK.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento Amazon MSK com o Lambda usando. JavaScript

```
exports.handler = async (event) => {
    // Iterate through keys
    for (let key in event.records) {
      console.log('Key: ', key)
      // Iterate through records
      event.records[key].map((record) => {
        console.log('Record: ', record)
        // Decode base64
        const msg = Buffer.from(record.value, 'base64').toString()
```

```
        console.log('Message:', msg)
      })
    }
}
```

Consumindo um evento Amazon MSK com o Lambda usando. TypeScript

```
import { MSKEvent, Context } from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "msk-handler-sample",
});

export const handler = async (
  event: MSKEvent,
  context: Context
): Promise<void> => {
  for (const [topic, topicRecords] of Object.entries(event.records)) {
    logger.info(`Processing key: ${topic}`);

    // Process each record in the partition
    for (const record of topicRecords) {
      try {
        // Decode the message value from base64
        const decodedMessage = Buffer.from(record.value, 'base64').toString();

        logger.info({
          message: decodedMessage
        });
      }
      catch (error) {
        logger.error('Error processing event', { error });
        throw error;
      }
    };
  }
}
```

Exemplos do Amazon Personalize usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Personalize.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateBatchInferenceJob`

O código de exemplo a seguir mostra como usar `CreateBatchInferenceJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchInferenceJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch inference job's parameters.

export const createBatchInferenceJobParam = {
```

```
  jobName: "JOB_NAME",
  jobInput: {
    s3DataSource: {
      path: "INPUT_PATH",
    },
  },
  jobOutput: {
    s3DataDestination: {
      path: "OUTPUT_PATH",
    },
  },
  roleArn: "ROLE_ARN",
  solutionVersionArn: "SOLUTION_VERSION_ARN",
  numResults: 20,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateBatchInferenceJobCommand(createBatchInferenceJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateBatchInferenceJob em Referência AWS SDK para JavaScript da API.

`CreateBatchSegmentJob`

O código de exemplo a seguir mostra como usar `CreateBatchSegmentJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchSegmentJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch segment job's parameters.

export const createBatchSegmentJobParam = {
  jobName: "NAME",
  jobInput: {
    s3DataSource: {
      path: "INPUT_PATH",
    },
  },
  jobOutput: {
    s3DataDestination: {
      path: "OUTPUT_PATH",
    },
  },
  roleArn: "ROLE_ARN",
  solutionVersionArn: "SOLUTION_VERSION_ARN",
  numResults: 20,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateBatchSegmentJobCommand(createBatchSegmentJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
```

```
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateBatchSegmentJob em Referência AWS SDK para JavaScript da API.

`CreateCampaign`

O código de exemplo a seguir mostra como usar `CreateCampaign`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.

import { CreateCampaignCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the campaign's parameters.
export const createCampaignParam = {
  solutionVersionArn: "SOLUTION_VERSION_ARN" /* required */,
  name: "NAME" /* required */,
  minProvisionedTPS: 1 /* optional integer */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateCampaignCommand(createCampaignParam),
    );
```

```
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateCampaign em Referência AWS SDK para JavaScript da API.

`CreateDataset`

O código de exemplo a seguir mostra como usar `CreateDataset`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  datasetType: "DATASET_TYPE" /* required */,
  name: "NAME" /* required */,
  schemaArn: "SCHEMA_ARN" /* required */,
};

export const run = async () => {
  try {
```

```
    const response = await personalizeClient.send(
      new CreateDatasetCommand(createDatasetParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateDataset em Referência AWS SDK para JavaScript da API.

`CreateDatasetExportJob`

O código de exemplo a seguir mostra como usar `CreateDatasetExportJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetExportJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the export job parameters.
export const datasetExportJobParam = {
  datasetArn: "DATASET_ARN" /* required */,
  jobOutput: {
    s3DataDestination: {
      path: "S3_DESTINATION_PATH" /* required */,
      //kmsKeyArn: 'ARN'  /* include if your bucket uses AWS KMS for encryption
```

```
    },
  },
  jobName: "NAME" /* required */,
  roleArn: "ROLE_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetExportJobCommand(datasetExportJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateDatasetExportJob em Referência AWS SDK para JavaScript da API.

`CreateDatasetGroup`

O código de exemplo a seguir mostra como usar `CreateDatasetGroup`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.

import { CreateDatasetGroupCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
```

```
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset group parameters.
export const createDatasetGroupParam = {
  name: "NAME" /* required */,
};

export const run = async (createDatasetGroupParam) => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetGroupCommand(createDatasetGroupParam),
    );
    console.log("Success", response);
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run(createDatasetGroupParam);
```

Criar um grupo de conjunto de dados de domínio.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetGroupCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: "NAME" /* required */,
  domain:
    "DOMAIN" /* required for a domain dsg, specify ECOMMERCE or VIDEO_ON_DEMAND */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetGroupCommand(domainDatasetGroupParams),
    );
    console.log("Success", response);
```

```
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateDatasetGroup em Referência AWS SDK para JavaScript da API.

`CreateDatasetImportJob`

O código de exemplo a seguir mostra como usar `CreateDatasetImportJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetImportJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: "DATASET_ARN" /* required */,
  dataSource: {
    /* required */
    dataLocation: "S3_PATH",
  },
  jobName: "NAME" /* required */,
  roleArn: "ROLE_ARN" /* required */,
};
```

```
export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetImportJobCommand(datasetImportJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateDatasetImportJob em Referência AWS SDK para JavaScript da API.

`CreateEventTracker`

O código de exemplo a seguir mostra como usar `CreateEventTracker`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateEventTrackerCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the event tracker's parameters.
export const createEventTrackerParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  name: "NAME" /* required */,
};
```

```
export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateEventTrackerCommand(createEventTrackerParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateEventTracker em Referência AWS SDK para JavaScript da API.

`CreateFilter`

O código de exemplo a seguir mostra como usar `CreateFilter`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateFilterCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the filter's parameters.
export const createFilterParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  name: "NAME" /* required */,
  filterExpression: "FILTER_EXPRESSION" /*required */,
```

```
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateFilterCommand(createFilterParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateFilter em Referência AWS SDK para JavaScript da API.

`CreateRecommender`

O código de exemplo a seguir mostra como usar `CreateRecommender`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateRecommenderCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the recommender's parameters.
export const createRecommenderParam = {
  name: "NAME" /* required */,
```

```
  recipeArn: "RECIPE_ARN" /* required */,
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateRecommenderCommand(createRecommenderParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateRecommender em Referência AWS SDK para JavaScript da API.

`CreateSchema`

O código de exemplo a seguir mostra como usar `CreateSchema`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from "node:fs";
```

```
const schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = "TEST"; // For unit tests.
}
// Set the schema parameters.
export const createSchemaParam = {
  name: "NAME" /* required */,
  schema: mySchema /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSchemaCommand(createSchemaParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Crie um esquema com um domínio.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from "node:fs";

const schemaFilePath = "SCHEMA_PATH";
let mySchema = "";
```

```
try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = "TEST"; // for unit tests.
}

// Set the domain schema parameters.
export const createDomainSchemaParam = {
  name: "NAME" /* required */,
  schema: mySchema /* required */,
  domain:
    "DOMAIN" /* required for a domain dataset group, specify ECOMMERCE or
 VIDEO_ON_DEMAND */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSchemaCommand(createDomainSchemaParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateSchema em Referência AWS SDK para JavaScript da API.

`CreateSolution`

O código de exemplo a seguir mostra como usar `CreateSolution`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution parameters.
export const createSolutionParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  name: "NAME" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionCommand(createSolutionParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateSolution em Referência AWS SDK para JavaScript da API.

`CreateSolutionVersion`

O código de exemplo a seguir mostra como usar `CreateSolutionVersion`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionVersionCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution version parameters.
export const solutionVersionParam = {
  solutionArn: "SOLUTION_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionVersionCommand(solutionVersionParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateSolutionVersion em Referência AWS SDK para JavaScript da API.

Exemplos de eventos do Amazon Personalize usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Personalize Events.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { PutEventsCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Convert your UNIX timestamp to a Date.
const sentAtDate = new Date(1613443801 * 1000); // 1613443801 is a testing value.
 Replace it with your sentAt timestamp in UNIX format.
```

```
// Set put events parameters.
const putEventsParam = {
  eventList: [
    /* required */
    {
      eventType: "EVENT_TYPE" /* required */,
      sentAt: sentAtDate /* required, must be a Date with js */,
      eventId: "EVENT_ID" /* optional */,
      itemId: "ITEM_ID" /* optional */,
    },
  ],
  sessionId: "SESSION_ID" /* required */,
  trackingId: "TRACKING_ID" /* required */,
  userId: "USER_ID" /* required */,
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutEventsCommand(putEventsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK para JavaScript da API.

`PutItems`

O código de exemplo a seguir mostra como usar `PutItems`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { PutItemsCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Set the put items parameters. For string properties and values, use the \
 character to escape quotes.
const putItemsParam = {
  datasetArn: "DATASET_ARN" /* required */,
  items: [
    /* required */
    {
      itemId: "ITEM_ID" /*  required */,
      properties:
        '{"PROPERTY1_NAME": "PROPERTY1_VALUE", "PROPERTY2_NAME": "PROPERTY2_VALUE",
 "PROPERTY3_NAME": "PROPERTY3_VALUE"}' /* optional */,
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutItemsCommand(putItemsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte PutItems em Referência AWS SDK para JavaScript da API.

PutUsers

O código de exemplo a seguir mostra como usar PutUsers.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { PutUsersCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Set the put users parameters. For string properties and values, use the \
 character to escape quotes.
const putUsersParam = {
  datasetArn: "DATASET_ARN",
  users: [
    {
      userId: "USER_ID",
      properties: '{"PROPERTY1_NAME": "PROPERTY1_VALUE"}',
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutUsersCommand(putUsersParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

- Para obter detalhes da API, consulte PutUsers em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon Personalize Runtime usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Personalize Runtime.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetPersonalizedRanking`

O código de exemplo a seguir mostra como usar `GetPersonalizedRanking`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
```

```
import { GetPersonalizedRankingCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the ranking request parameters.
export const getPersonalizedRankingParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  inputList: ["ITEM_ID_1", "ITEM_ID_2", "ITEM_ID_3", "ITEM_ID_4"],
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetPersonalizedRankingCommand(getPersonalizedRankingParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte GetPersonalizedRanking em Referência AWS SDK para JavaScript da API.

`GetRecommendations`

O código de exemplo a seguir mostra como usar `GetRecommendations`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";

import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Obtenha recomendações com um filtro (grupo de conjunto de dados personalizados).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  recommenderArn: "RECOMMENDER_ARN" /* required */,
  userId: "USER_ID" /* required */,
```

```
  numResults: 15 /* optional */,
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Obtenha recomendações filtradas de um recomendador criado em um grupo de conjunto de dados de domínio.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here:
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
  filterArn: "FILTER_ARN" /* required to filter recommendations */,
  filterValues: {
    PROPERTY:
      '"VALUE"' /* Only required if your filter has a placeholder parameter */,
  },
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
```

```
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte GetRecommendations em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon Pinpoint usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendMessages`

O código de exemplo a seguir mostra como usar `SendMessages`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { PinpointClient } from "@aws-sdk/client-pinpoint";
// Set the AWS Region.
const REGION = "us-east-1";
export const pinClient = new PinpointClient({ region: REGION });
```

Envie uma mensagem de e-mail.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

// The FromAddress must be verified in SES.
const fromAddress = "FROM_ADDRESS";
const toAddress = "TO_ADDRESS";
const projectId = "PINPOINT_PROJECT_ID";

// The subject line of the email.
const subject = "Amazon Pinpoint Test (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
const body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
----------------------------------------------------
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
 Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
const body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
```

```
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint Email API</a>
 using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding for the subject line and message body of the email.
const charset = "UTF-8";

const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: fromAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

const run = async () => {
  try {
    const { MessageResponse } = await pinClient.send(
```

```
      new SendMessagesCommand(params),
    );

    if (!MessageResponse) {
      throw new Error("No message response.");
    }

    if (!MessageResponse.Result) {
      throw new Error("No message result.");
    }

    const recipientResult = MessageResponse.Result[toAddress];

    if (recipientResult.StatusCode !== 200) {
      throw new Error(recipientResult.StatusMessage);
    }
    console.log(recipientResult.MessageId);
  } catch (err) {
    console.log(err.message);
  }
};

run();
```

Envie uma mensagem SMS.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

/* The phone number or short code to send the message from. The phone number
 or short code that you specify has to be associated with your Amazon Pinpoint
account. For best results, specify long codes in E.164 format. */
const originationNumber = "SENDER_NUMBER"; //e.g., +1XXXXXXXXXX

// The recipient's phone number.  For best results, you should specify the phone
 number in E.164 format.
const destinationNumber = "RECEIVER_NUMBER"; //e.g., +1XXXXXXXXXX

// The content of the SMS message.
const message =
```

```
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out.";

/*The Amazon Pinpoint project/application ID to use when you send this message.
Make sure that the SMS channel is enabled for the project or application
that you choose.*/
const projectId = "PINPOINT_PROJECT_ID"; //e.g., XXXXXXXX66e4e9986478cXXXXXXXXX

/* The type of SMS message that you want to send. If you plan to send
time-sensitive content, specify TRANSACTIONAL. If you plan to send
marketing-related content, specify PROMOTIONAL.*/
const messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
const registeredKeyword = "myKeyword";

/* The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html.*/

const senderId = "MySenderID";

// Specify the parameters to pass to the API.
const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
  },
};
```

```
const run = async () => {
  try {
    const data = await pinClient.send(new SendMessagesCommand(params));
    console.log(
      `Message sent!
 ${data.MessageResponse.Result[destinationNumber].StatusMessage}`,
    );
  } catch (err) {
    console.log(err);
  }
};
run();
```

- Para obter detalhes da API, consulte SendMessages em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon Polly usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Polly.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado dentro das funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;
```

```
  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);
```

```
  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });
```

```
  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon RDS usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon RDS.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários
- Exemplos sem servidor

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um Node.js back-end Express.

- Integre um aplicativo React.js web com Serviços da AWS.
- Liste, adicione e atualize itens em uma tabela do Aurora.
- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora

- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectando-se a um banco de dados do Amazon RDS em uma função do Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,
```

```
  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}

export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

Conectando-se a um banco de dados do Amazon RDS em uma função do Lambda usando. TypeScript

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that the
 DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}

async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
```

```
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number; body:
 string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
    };
};
```

Exemplos do Amazon RDS Data Service usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon RDS Data Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um Node.js back-end Express.

- Integre um aplicativo React.js web com Serviços da AWS.
- Liste, adicione e atualize itens em uma tabela do Aurora.
- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon Redshift usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Redshift.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Crie o cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME", // Required
  NodeType: "NODE_TYPE", //Required
  MasterUsername: "MASTER_USER_NAME", // Required - must be lowercase
```

```
  MasterUserPassword: "MASTER_USER_PASSWORD", // Required - must contain at least
 one uppercase letter, and one number
  ClusterType: "CLUSTER_TYPE", // Required
  IAMRoleARN: "IAM_ROLE_ARN", // Optional - the ARN of an IAM role with permissions
 your cluster needs to access other AWS services on your behalf, such as Amazon S3.
  ClusterSubnetGroupName: "CLUSTER_SUBNET_GROUPNAME", //Optional - the name of a
 cluster subnet group to be associated with this cluster. Defaults to 'default' if
 not specified.
  DBName: "DATABASE_NAME", // Optional - defaults to 'dev' if not specified
  Port: "PORT_NUMBER", // Optional - defaults to '5439' if not specified
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new CreateClusterCommand(params));
    console.log(
      `Cluster ${data.Cluster.ClusterIdentifier} successfully created`,
    );
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte CreateCluster em Referência AWS SDK para JavaScript da API.

`DeleteCluster`

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Crie o cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  SkipFinalClusterSnapshot: false,
  FinalClusterSnapshotIdentifier: "CLUSTER_SNAPSHOT_ID",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DeleteClusterCommand(params));
    console.log("Success, cluster deleted. ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte DeleteCluster em Referência AWS SDK para JavaScript da API.

`DescribeClusters`

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Descreva os clusters.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeClustersCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DescribeClustersCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter detalhes da API, consulte DescribeClusters em Referência AWS SDK para JavaScript da API.

`ModifyCluster`

O código de exemplo a seguir mostra como usar `ModifyCluster`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Modificar um cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { ModifyClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

// Set the parameters
const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  MasterUserPassword: "NEW_MASTER_USER_PASSWORD",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new ModifyClusterCommand(params));
    console.log("Success was modified.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

- Para obter detalhes da API, consulte ModifyCluster em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon Rekognition usando o SDK para (v3) JavaScript

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Rekognition.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para JavaScript (v3)

Mostra como usar o Amazon Rekognition com o AWS SDK para JavaScript para criar um aplicativo que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Aprenda como:

- Criar um usuário não autenticado usando o Amazon Cognito.
- Analisar imagens em busca de objetos usando o Amazon Rekognition.
- Verificar um endereço de e-mail para o Amazon SES.
- Enviar uma notificação por e-mail usando o Amazon SES.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Exemplos do Amazon S3 usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {
  // When no region or credentials are provided, the SDK will use the
  // region and credentials from the local AWS config.
  const client = new S3Client({});

  try {
    /**
     * @type { import("@aws-sdk/client-s3").Bucket[] }
     */
    const buckets = [];

    for await (const page of paginateListBuckets({ client }, {})) {
      buckets.push(...page.Buckets);
    }
    console.log("Buckets: ");
    console.log(buckets.map((bucket) => bucket.Name).join("\n"));
    return buckets;
  } catch (caught) {
    // ListBuckets does not throw any modeled errors. Any error caught
    // here will be something generic like `AccessDenied`.
    if (caught instanceof S3ServiceException) {
      console.error(`${caught.name}: ${caught.message}`);
    } else {
      // Something besides S3 failed.
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Primeiro, importe todos os módulos necessários.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
```

```
} from "@aws-sdk/client-s3";
```

As importações anteriores fazem referência a alguns utilitários auxiliares. Esses utilitários são locais para o GitHub repositório vinculado no início desta seção. Para sua referência, consulte as implementações a seguir desses utilitários.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}} options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {{ message: string }} options
   */
  password(options) {
    return password({ ...options, mask: true });
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && `${prompt} `;
    const ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
```

```
  };

  /**
   * @param {{ message: string }} options
   */
  confirm(options) {
    return confirm(options);
  }

  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}} options
   */
  checkbox(options) {
    return checkbox(options);
  }
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

Os objetos são armazenados em “buckets”. Vamos definir uma função para criar um bucket.

```
export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};
```

Os buckets contêm “objetos”. Essa função faz upload do conteúdo de um diretório para seu bucket como objetos.

```
export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
```

```
    const filePath = `${folderPath}/${key}`;
    const fileContent = readFileSync(filePath);
    return {
      Key: key,
      Body: fileContent,
    };
  });

  for (const file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
};
```

Depois de fazer upload dos objetos, confira se eles foram carregados corretamente. Você pode usar ListObjects para isso. Você usará a propriedade “Key”, mas também há outras propriedades úteis na resposta.

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(`${contentsList}\n`);
};
```

Às vezes é necessário copiar um objeto de um bucket em outro. Use o CopyObject comando para isso.

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });
```

```
  if (!proceed) {
    return;
  }
  const copy = async () => {
    try {
      const sourceBucket = await prompter.input({
        message: "Enter source bucket name:",
      });
      const sourceKey = await prompter.input({
        message: "Enter source key:",
      });
      const destinationKey = await prompter.input({
        message: "Enter destination key:",
      });

      const command = new CopyObjectCommand({
        Bucket: destinationBucket,
        CopySource: `${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });
      await s3Client.send(command);
      await copyFileFromBucket({ destinationBucket });
    } catch (err) {
      console.error("Copy error.");
      console.error(err);
      const retryAnswer = await prompter.confirm({ message: "Try again?" });
      if (retryAnswer) {
        await copy();
      }
    }
  };
  await copy();
};
```

Não há um método de SDK para obter vários objetos de um bucket. Em vez disso, você criará uma lista de objetos para baixar e iterar sobre eles.

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
```

```
    message: "Enter destination path for files:",
  });

  for (const content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

É hora de limpar seus recursos. Um bucket deve estar vazio para poder ser excluído. Essas duas funções esvaziam e excluem o bucket.

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

A função “principal” reúne tudo. Se você executar esse arquivo diretamente, a função principal será chamada.

```
const main = async () => {
```

```
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to
 provide your own.",
    );
    await uploadFilesToBucket({
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie o objeto.

```
import {
  S3Client,
  CopyObjectCommand,
  ObjectNotInActiveTierError,
  waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{
 *   sourceBucket: string,
 *   sourceKey: string,
 *   destinationBucket: string,
 *   destinationKey: string }} config
 */
```

```
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
  destinationKey,
}) => {
  const client = new S3Client({});

  try {
    await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucket}/${sourceKey}`,
        Bucket: destinationBucket,
        Key: destinationKey,
      }),
    );
    await waitUntilObjectExists(
      { client },
      { Bucket: destinationBucket, Key: destinationKey },
    );
    console.log(
      `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/
${destinationKey}`,
    );
  } catch (caught) {
    if (caught instanceof ObjectNotInActiveTierError) {
      console.error(
        `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the
 active tier.`,
      );
    } else {
      throw caught;
    }
  }
};
```

Copie o objeto caso seu ETag não corresponda ao fornecido.

```
import {
  CopyObjectCommand,
  NoSuchKey,
```

```
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string, destinationBucketName:
 string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;
  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket "${sourceBucketName}":
 ${caught.name}: ${caught.message}`,
      );
    } else {
```

```
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copie o objeto caso seu ETag não corresponda ao fornecido.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string, destinationBucketName:
 string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfNoneMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
```

```
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket "${sourceBucketName}":
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
```

```
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copie o objeto usando a condição de que ele tenha sido criado ou modificado em determinado período.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string, destinationBucketName:
 string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const name = data.name;
  const client = new S3Client({});
```

```
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
  const copiedKey = name + sourceKeyName;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfModifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
```

```
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copie o objeto usando a condição de que ele não tenha sido criado ou modificado em determinado período.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string, destinationBucketName:
 string }}
```

```
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);
  const client = new S3Client({});
  const name = data.name;
  const copiedKey = name + sourceKeyName;
  const copySource = `${sourceBucketName}/${sourceKeyName}`;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfUnmodifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
```

```
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK para JavaScript da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o bucket.

```
import {
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  CreateBucketCommand,
  S3Client,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
 constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket
 names must be globally unique.`,
      );
```

```
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK para JavaScript da API.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua o bucket.

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
```

```
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting the bucket. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteBucket em Referência AWS SDK para JavaScript da API.

`DeleteBucketPolicy`

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a política de bucket.

```
import {
  DeleteBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Bucket policy deleted from "${bucketName}".`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
```

```
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteBucketPolicy em Referência AWS SDK para JavaScript da API.

`DeleteBucketWebsite`

O código de exemplo a seguir mostra como usar `DeleteBucketWebsite`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a configuração de site do bucket.

```
import {
  DeleteBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the website configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
```

```
      new DeleteBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    // The response code will be successful for both removed configurations and
    // configurations that did not exist in the first place.
    console.log(
      `The bucket "${bucketName}" is not longer configured as a website, or it never
 was.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}. The
 bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteBucketWebsite em Referência AWS SDK para JavaScript da API.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um objeto.

```
import {
  DeleteObjectCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
 exist.`,
    );
  } catch (caught) {
    if (
```

```
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteObject em Referência AWS SDK para JavaScript da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua vários objetos.

```
import {
  DeleteObjectsCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";
```

```
/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
        { client },
        { Bucket: bucketName, Key: key },
      );
    }
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
 objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
```

```
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK para JavaScript da API.

`GetBucketAcl`

O código de exemplo a seguir mostra como usar `GetBucketAcl`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha as permissões de ACL.

```
import {
  GetBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketAclCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`ACL for bucket "${bucketName}":`);
```

```
    console.log(JSON.stringify(response, null, 2));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetBucketAcl em Referência AWS SDK para JavaScript da API.

`GetBucketCors`

O código de exemplo a seguir mostra como usar `GetBucketCors`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política de CORS para o bucket.

```
import {
```

```
  GetBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-".repeat(10)}`,
        `\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\nAllowedMethods: ${cr.AllowedMethods}`,
        `\nAllowedOrigins: ${cr.AllowedOrigins}`,
        `\nExposeHeaders: ${cr.ExposeHeaders}`,
        `\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
```

```
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetBucketCors em Referência AWS SDK para JavaScript da API.

`GetBucketPolicy`

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política de bucket.

```
import {
  GetBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Policy } = await client.send(
      new GetBucketPolicyCommand({
```

```
        Bucket: bucketName,
      }),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetBucketPolicy em Referência AWS SDK para JavaScript da API.

`GetBucketWebsite`

O código de exemplo a seguir mostra como usar `GetBucketWebsite`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a configuração do site.

```
import {
  GetBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:\n
${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}. The
 bucket isn't configured as a website.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte GetBucketWebsite em Referência AWS SDK para JavaScript da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Baixe o objeto.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
```

```
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Baixe o objeto com a condição de que sua ETag corresponda à fornecida.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
```

```
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
```

```
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Baixe o objeto com a condição de que sua ETag não corresponda à fornecida.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
```

```
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
```

```
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Baixe o objeto usando a condição de que ele tenha sido criado ou modificado em determinado período.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfModifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
```

```
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
```

```
  } else {
    console.error(errors.join("\n"));
  }
}
```

Baixe o objeto usando a condição de que ele não tenha sido criado ou modificado em determinado período.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfUnmodifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
```

```
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetObject em Referência AWS SDK para JavaScript da API.

GetObjectLegalHold

O código de exemplo a seguir mostra como usar `GetObjectLegalHold`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
```

```
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in ${bucketName}.
 The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in ${bucketName}
 from ${bucketName}.  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
```

```
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte GetObjectLegalHold em Referência AWS SDK para JavaScript da API.

`GetObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `GetObjectLockConfiguration`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { ObjectLockConfiguration } = await client.send(
      new GetObjectLockConfigurationCommand({
        Bucket: bucketName,
```

```
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the bucket>",
      }),
    );
    console.log(
      `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting object lock configuration for ${bucketName}.
 The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object lock configuration for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
```

```
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte GetObjectLockConfiguration em Referência AWS SDK para JavaScript da API.

`GetObjectRetention`

O código de exemplo a seguir mostra como usar `GetObjectRetention`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
```

```
  try {
    const { Retention } = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    console.log(
      `${key} in ${bucketName} will be retained until ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
        `The object "${key}" in the bucket "${bucketName}" does not have an
 ObjectLock configuration.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object retention settings for "${bucketName}".
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
```

```
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte GetObjectRetention em Referência AWS SDK para JavaScript da API.

`ListBuckets`

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os buckets.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * List the Amazon S3 buckets in your account.
 */
export const main = async () => {
  const client = new S3Client({});
  /** @type {?import('@aws-sdk/client-s3').Owner} */
  let Owner = null;

  /** @type {import('@aws-sdk/client-s3').Bucket[]} */
  const Buckets = [];

  try {
    const paginator = paginateListBuckets({ client }, {});

    for await (const page of paginator) {
      if (!Owner) {
        Owner = page.Owner;
      }

      Buckets.push(...page.Buckets);
    }

    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
      }:`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (caught) {
    if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing buckets.  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para JavaScript da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todos os objetos no bucket. Se houver mais de um objeto, IsTruncated NextContinuationToken ele será usado para iterar a lista completa.

```
import {
  S3Client,
  S3ServiceException,
  // This command supersedes the ListObjectsCommand and is the recommended way to
 list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );

    for await (const page of paginator) {
      objects.push(page.Contents.map((o) => o.Key));
    }
    objects.forEach((objectList, pageNum) => {
      console.log(
```

```
        `Page ${pageNum + 1}\n------\n${objectList.map((o) => `•
 ${o}`).join("\n")}\n`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}". The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}".  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK para JavaScript API.

`PutBucketAcl`

O código de exemplo a seguir mostra como usar `PutBucketAcl`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloque a ACL do bucket.

```
import {
  PutBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant read access to a user using their canonical AWS account ID.
 *
 * Most Amazon S3 use cases don't require the use of access control lists (ACLs).
 * We recommend that you disable ACLs, except in unusual circumstances where
 * you need to control access for each object individually. Consider a policy
 instead.
 * For more information see https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucket-policies.html.
 * @param {{ bucketName: string, granteeCanonicalUserId: string,
 ownerCanonicalUserId }}
 */
export const main = async ({
  bucketName,
  granteeCanonicalUserId,
  ownerCanonicalUserId,
}) => {
  const client = new S3Client({});
  const command = new PutBucketAclCommand({
    Bucket: bucketName,
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of your
 AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/finding-canonical-user-id.html.
            ID: granteeCanonicalUserId,
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "READ",
        },
```

```
      ],
      Owner: {
        ID: ownerCanonicalUserId,
      },
    },
  });

  try {
    await client.send(command);
    console.log(`Granted READ access to ${bucketName}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutBucketAcl em Referência AWS SDK para JavaScript da API.

`PutBucketCors`

O código de exemplo a seguir mostra como usar `PutBucketCors`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione uma regra de CORS.

```
import {
  PutBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutBucketCorsCommand({
        Bucket: bucketName,
        CORSConfiguration: {
          CORSRules: [
            {
              // Allow all headers to be sent to this bucket.
              AllowedHeaders: ["*"],
              // Allow only GET and PUT methods to be sent to this bucket.
              AllowedMethods: ["GET", "PUT"],
              // Allow only requests from the specified origin.
              AllowedOrigins: ["https://www.example.com"],
              // Allow the entity tag (ETag) header to be returned in the response.
 The ETag header
              // The entity tag represents a specific version of the object. The
 ETag reflects
              // changes only to the contents of an object, not its metadata.
              ExposeHeaders: ["ETag"],
```

```
              // How long the requesting browser should cache the preflight
 response. After
              // this time, the preflight request will have to be made again.
              MaxAgeSeconds: 3600,
            },
          ],
        },
      },
    );
    console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting CORS rules for ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting CORS rules for ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutBucketCors em Referência AWS SDK para JavaScript da API.

`PutBucketPolicy`

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione a política.

```
import {
  PutBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
  const client = new S3Client({});
  const command = new PutBucketPolicyCommand({
    // This is a resource-based policy. For more information on resource-based
 policies,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/
access_policies.html#policies_resource-based.
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            AWS: iamRoleArn,
          },
          Action: "s3:GetObject",
          Resource: `arn:aws:s3:::${bucketName}/*`,
        },
      ],
    }),
    // Apply the preceding policy to this bucket.
    Bucket: bucketName,
```

```
  });

  try {
    await client.send(command);
    console.log(
      `GetObject access to the bucket "${bucketName}" was granted to the provided
 IAM role.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "MalformedPolicy"
    ) {
      console.error(
        `Error from S3 while setting the bucket policy for the bucket
 "${bucketName}". The policy was malformed.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting the bucket policy for the bucket
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutBucketPolicy em Referência AWS SDK para JavaScript da API.

`PutBucketWebsite`

O código de exemplo a seguir mostra como usar `PutBucketWebsite`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração do site.

```
import {
  PutBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/
WebsiteAccessPermissionsReqd.html.
 *
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request when the request is
        // for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
```

```
    await client.send(command);
    console.log(
      `The bucket "${bucketName}" has been configured as a static website.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static
 website. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static
 website. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutBucketWebsite em Referência AWS SDK para JavaScript da API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Carregue o objeto.

```
import { readFile } from "node:fs/promises";

import {
  PutObjectCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "EntityTooLarge"
    ) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console
 (160GB max) \
or the multipart upload API (5TB max).`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
```

```
  }
};
```

Carregue o objeto com a condição de que sua ETag corresponda à fornecida.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
```

```
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte PutObject em Referência AWS SDK para JavaScript da API.

`PutObjectLegalHold`

O código de exemplo a seguir mostra como usar `PutObjectLegalHold`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  PutObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" | "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'.",
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
```

```
    },
  });

  try {
    await client.send(command);
    console.log(
      `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in
 "${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
 "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    objectKey: {
      type: "string",
```

```
      required: true,
    },
    legalHoldStatus: {
      type: "string",
      default: "ON",
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte PutObjectLegalHold em Referência AWS SDK para JavaScript da API.

`PutObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `PutObjectLockConfiguration`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de Bloqueio de Objetos de um bucket.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
```

```
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the bucket
 "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the bucket
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
```

```
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Defina o período de retenção padrão de um bucket.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
export const main = async ({ bucketName, retentionDays }) => {
  const client = new S3Client({});
```

```
 try {
   await client.send(
     new PutObjectLockConfigurationCommand({
       Bucket: bucketName,
       // The Object Lock configuration that you want to apply to the specified
bucket.
       ObjectLockConfiguration: {
         ObjectLockEnabled: "Enabled",
         Rule: {
           // The default Object Lock retention mode and period that you want to
apply
           // to new objects placed in the specified bucket. Bucket settings
require
           // both a mode and a period. The period can be either Days or Years but
           // you must select one.
           DefaultRetention: {
             // In governance mode, users can't overwrite or delete an object
version
             // or alter its lock settings unless they have special permissions.
With
             // governance mode, you protect objects against being deleted by most
users,
             // but you can still grant some users permission to alter the
retention settings
             // or delete the objects if necessary.
             Mode: "GOVERNANCE",
             Days: Number.parseInt(retentionDays),
           },
         },
       },
     }),
   );
   console.log(
     `Set default retention mode to "GOVERNANCE" with a retention period of
${retentionDays} day(s).`,
   );
 } catch (caught) {
   if (
     caught instanceof S3ServiceException &&
     caught.name === "NoSuchBucket"
   ) {
     console.error(
       `Error from S3 while setting the default object retention for a bucket. The
bucket doesn't exist.`,
```

```
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting the default object retention for a bucket.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    retentionDays: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte PutObjectLockConfiguration em Referência AWS SDK para JavaScript da API.

`PutObjectRetention`

O código de exemplo a seguir mostra como usar `PutObjectRetention`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  PutObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most users,
      // but you can still grant some users permission to alter the retention
 settings
      // or delete the objects if necessary.
      Mode: "GOVERNANCE",
```

```
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
  });

  try {
    await client.send(command);
    console.log("Object Retention settings updated.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period on
 an object. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period on
 an object. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
```

```
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Para obter detalhes da API, consulte PutObjectRetention em Referência AWS SDK para JavaScript da API.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um URL pré-assinado para carregar um objeto em um bucket.

```
import https from "node:https";

import { XMLParser } from "fast-xml-parser";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
```

```
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Make a PUT request to the provided URL.
 *
 * @param {string} url
 * @param {string} data
 */
const put = (url, data) => {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
```

```
        res.on("end", () => {
          const parser = new XMLParser();
          if (res.statusCode >= 200 && res.statusCode <= 299) {
            resolve(parser.parse(responseBody, true));
          } else {
            reject(parser.parse(responseBody, true));
          }
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
};

/**
 * Create two presigned urls for uploading an object to an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");
```

```
    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
 configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Crie um URL pré-assinado para baixar um objeto de um bucket.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
```

```
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      region,
      key,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
 configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma página da web que oferece uma lista de objetos do Amazon S3

O exemplo de código a seguir mostra como listar objetos do Amazon S3 em uma página da web.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O código a seguir é o componente React relevante que faz chamadas para o AWS SDK. Uma versão executável do aplicativo contendo esse componente pode ser encontrada no link anterior GitHub .

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach a
 role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
 traffic from
      // this example site. Here's an example configuration that allows all origins.
 Don't
      // do this in production.
      //[
      //  {
      //    "AllowedHeaders": ["*"],
      //    "AllowedMethods": ["GET"],
      //    "AllowedOrigins": ["*"],
      //    "ExposeHeaders": [],
      //  },
      //]
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
```

```
      }),
    });
    const command = new ListObjectsCommand({ Bucket: "bucket-name" });
    client.send(command).then(({ Contents }) => setObjects(Contents || []));
  }, []);

  return (
    <div className="App">
      {objects.map((o) => (
        <div key={o.ETag}>{o.Key}</div>
      ))}
    </div>
  );
}

export default App;
```

- Para obter detalhes da API, consulte ListObjects em Referência AWS SDK para JavaScript da API.

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Excluir todos os objetos em um bucket

Os exemplos de código a seguir mostram como excluir todos os objetos de um bucket do Amazon S3.

SDK para JavaScript (v3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua todos os objetos de um bucket específico do Amazon S3.

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 *
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  try {
    console.log(`Deleting all objects in bucket: ${bucketName}`);

    const paginator = paginateListObjectsV2(
      { client },
      {
        Bucket: bucketName,
```

```
      },
    );

    const objectKeys = [];
    for await (const { Contents } of paginator) {
      objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
    }

    const deleteCommand = new DeleteObjectsCommand({
      Bucket: bucketName,
      Delete: { Objects: objectKeys },
    });

    await client.send(deleteCommand);

    console.log(`All objects deleted from bucket: ${bucketName}`);
  } catch (caught) {
    if (caught instanceof Error) {
      console.error(
        `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    bucketName: {
      type: "string",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - DeleteObjects

- ListObjectsV2

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para JavaScript (v3)

Mostra como usar o Amazon Rekognition com o AWS SDK para JavaScript para criar um aplicativo que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Aprenda como:

- Criar um usuário não autenticado usando o Amazon Cognito.
- Analisar imagens em busca de objetos usando o Amazon Rekognition.
- Verificar um endereço de e-mail para o Amazon SES.
- Enviar uma notificação por e-mail usando o Amazon SES.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Bloquear objetos do Amazon S3

O exemplo de código a seguir mostra como trabalhar com os recursos de bloqueio de objetos do S3.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Ponto de entrada do cenário (index.js). Isso orquestra todas as etapas. Visite GitHub para ver os detalhes da implementação do Cenário ScenarioInput ScenarioOutput,, ScenarioAction e.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "./setup.steps.js";
```

```
/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
```

```
    demo: new scenarios.Scenario(
      "S3 Object Locking - Demo",
      [loadState, replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Object Locking - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
 features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Envie mensagens de boas-vindas para o console (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */
```

```
/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature
 Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
 several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Implante buckets, objetos e configurações de arquivos (setup.steps.js).

```
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
  GetBucketVersioningCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
```

```
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
         ${state.bucketPrefix}-no-lock with object lock False.
         ${state.bucketPrefix}-lock-enabled with object lock True.
         ${state.bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });
```

```
/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${state.bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

    try {
      await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
      await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: lockEnabledBucketName,
          ObjectLockEnabledForBucket: true,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: lockEnabledBucketName },
      );
      await client.send(
        new CreateBucketCommand({ Bucket: retentionBucketName }),
      );
      await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

      state.noLockBucketName = noLockBucketName;
      state.lockEnabledBucketName = lockEnabledBucketName;
      state.retentionBucketName = retentionBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
      }
    }
  });
```

```
/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
         file0.txt in ${state.bucketPrefix}-no-lock.
         file1.txt in ${state.bucketPrefix}-no-lock.
         file0.txt in ${state.bucketPrefix}-lock-enabled.
         file1.txt in ${state.bucketPrefix}-lock-enabled.
         file0.txt in ${state.bucketPrefix}-retention-after-creation.
         file1.txt in ${state.bucketPrefix}-retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
```

```
          Bucket: state.noLockBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
        );
```

```
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    (state) => `A bucket can be configured to use object locking with a default
 retention period.
A default retention period will be configured for ${state.bucketPrefix}-retention-
after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );
```

```
    const getBucketVersioning = new GetBucketVersioningCommand({
      Bucket: state.retentionBucketName,
    });

    await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
      const { Status } = await client.send(getBucketVersioning);
      if (Status !== "Enabled") {
        throw new Error("Bucket versioning is not enabled.");
      }
    });

    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
```

```
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
```

```
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.lockEnabledBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be able
 to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
```

```
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
```

```
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be able
 to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
```

```
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
  getBucketPrefix,
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

Visualize e exclua arquivos nos buckets (repl.steps.js).

```
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
```

```
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
```

```
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);
```

```
    const fileInput = new scenarios.ScenarioInput(
      "selectedFile",
      "Select a file:",
      {
        type: "select",
        choices: files.map((file, index) => ({
          name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
            file.version
          })`,
          value: index,
        })),
      },
    );

    const { replChoice } = state;

    switch (replChoice) {
      case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [
          state.noLockBucketName,
          state.lockEnabledBucketName,
          state.retentionBucketName,
        ]);
        state.replOutput = files
          .map(
            (file) =>
              `${file.bucket}: ${file.key} (version: ${file.version})`,
          )
          .join("\n");
        break;
      }
      case choices.DELETE_FILE: {
        /** @type {number} */
        const fileToDelete = await fileInput.handle(state);
        const selectedFile = files[fileToDelete];
        try {
          await client.send(
            new DeleteObjectCommand({
              Bucket: selectedFile.bucket,
              Key: selectedFile.key,
              VersionId: selectedFile.version,
            }),
          );
```

```
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.DELETE_FILE_WITH_RETENTION: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.OVERWRITE_FILE: {
          /** @type {number} */
          const fileToOverwrite = await fileInput.handle(state);
          const selectedFile = files[fileToOverwrite];
          try {
            await client.send(
              new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
              }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
```

```
          } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
 bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.VIEW_RETENTION_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const retention = await client.send(
              new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            const bucketConfig = await client.send(
              new GetObjectLockConfigurationCommand({
                Bucket: selectedFile.bucket,
              }),
            );
            state.replOutput = `Object retention for ${selectedFile.key}
 in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
 ${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
 ${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
          } catch (err) {
            state.replOutput = `Unable to fetch object lock retention:
 '${err.message}'`;
          }
          break;
        }
        case choices.VIEW_LEGAL_HOLD_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const legalHold = await client.send(
              new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
```

```
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Object legal hold for ${selectedFile.key} in
 ${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
          } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
          }
          break;
        }
        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
          "REPL output",
          (state) => state.replOutput,
          { preformatted: true },
        ),
      },
    },
  );

export { replInput, replAction, choices };
```

Destrua todos os recursos criados (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
```

```
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
```

```
    }
    throw e;
  }

  for (const version of objectsResponse.Versions || []) {
    const { Key, VersionId } = version;

    try {
      const legalHold = await client.send(
        new GetObjectLegalHoldCommand({
          Bucket: bucket,
          Key,
          VersionId,
        }),
      );

      if (legalHold.LegalHold?.Status === "ON") {
        await client.send(
          new PutObjectLegalHoldCommand({
            Bucket: bucket,
            Key,
            VersionId,
            LegalHold: {
              Status: "OFF",
            },
          }),
        );
      }
    } catch (err) {
      console.log(
        `Unable to fetch legal hold for ${Key} in ${bucket}: '${err.message}'`,
      );
    }

    try {
      const retention = await client.send(
        new GetObjectRetentionCommand({
          Bucket: bucket,
          Key,
          VersionId,
        }),
      );

      if (retention.Retention?.Mode === "GOVERNANCE") {
```

```
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
                BypassGovernanceRetention: true,
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch object lock retention for ${Key} in ${bucket}:
 '${err.message}'`,
          );
        }

        await client.send(
          new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Fazer solicitações condicionais

O exemplo de código a seguir mostra como adicionar pré-condições a solicitações do Amazon S3.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Ponto de entrada do fluxo de trabalho (index.js). Isso orquestra todas as etapas. Visite GitHub para ver os detalhes da implementação do Cenário ScenarioInput ScenarioOutput,, ScenarioAction e.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
```

```
    deploy: new scenarios.Scenario(
      "S3 Conditional Requests - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Conditional Requests - Demo",
      [loadState, welcome(scenarios), replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Conditional Requests - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
```

```
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
 features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Envie mensagens de boas-vindas para o console (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3 operations." +
      " You can use conditional requests to add preconditions to S3 read requests to
 return " +
      "or copy an object based on its Entity tag (ETag), or last modified date.You
 can use " +
      "a conditional write requests to prevent overwrites by ensuring there is no
 existing " +
      "object with the same key.\n" +
      "This example will enable you to perform conditional reads and writes that
 will succeed " +
      "or fail based on your selected options.\n" +
      "Sample buckets and a sample object will be created as part of the example.\n"
 +
      "Some steps require a key name prefix to be defined by the user. Before you
 begin, you can " +
      "optionally edit this prefix in ./object_name.json. If you do so, please
 reload the scenario before you begin.",
    { header: true },
```

```
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Implante buckets e objetos (setup.steps.js).

```
import {
  ChecksumAlgorithm,
  CreateBucketCommand,
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
```

```
  );
/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
         ${state.bucketPrefix}-source-bucket.
         ${state.bucketPrefix}-destination-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const sourceBucketName = `${state.bucketPrefix}-source-bucket`;
    const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;

    try {
      await client.send(
        new CreateBucketCommand({
          Bucket: sourceBucketName,
        }),
      );
      await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: destinationBucketName,
        }),
      );
      await waitUntilBucketExists(
        { client },
```

```
        { Bucket: destinationBucketName },
      );

      state.sourceBucketName = sourceBucketName;
      state.destinationBucketName = destinationBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
         file01.txt in ${state.bucketPrefix}-source-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
```

```
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.sourceBucketName,
          Key: "file01.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
 ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });

export {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
};
```

Obtenha, copie e coloque objetos usando solicitações condicionais do S3 (repl.steps.js).

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
```

```
  CopyObjectCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";
import data from "./object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
    "Explore the S3 conditional request features by selecting one of the following
 choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
```

```
        {
          name: "Perform a conditional copy. These examples use the key name prefix
 defined in ./object_name.json.",
          value: choices.CONDITIONAL_COPY,
        },
        {
          name: "Perform a conditional write. This example use the sample file ./
text02.txt.",
          value: choices.CONDITIONAL_WRITE,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key } = version;
      files.push({ bucket, key: Key });
    }
  }
  return files;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
  const objectsResponse = await client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: key,
```

```
    }),
  );
  return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
        },
      );
      const condReadOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional read action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should fail.",
```

```
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
          ],
        },
      );
      const condCopyOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional copy action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
          ],
        },
      );
      const condWriteOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional write action would you like to take?",
        {
          type: "select",
          choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate, the
write will fail.",
          ],
        },
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.sourceBucketName,
            state.destinationBucketName,
          ]);
          state.replOutput = files
            .map(
```

```
              (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
            )
            .join("\n");
          break;
        }
        case choices.CONDITIONAL_READ:
          {
            const selectedCondRead = await condReadOptions.handle(state);
            if (
              selectedCondRead ===
              "If-Match: using the object's ETag. This condition should succeed."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              const ETag = await getEtag(client, bucket, key);

              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfMatch: ETag,
                  }),
                );
                state.replOutput = `${key} in bucket ${state.sourceBucketName} read
because ETag provided matches the object's ETag.`;
              } catch (err) {
                state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-None-Match: using the object's ETag. This condition should fail."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              const ETag = await getEtag(client, bucket, key);

              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
```

```
                    Key: key,
                    IfNoneMatch: ETag,
                  }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
              } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-Modified-Since: using yesterday's date. This condition should
succeed."
            ) {
              const date = new Date();
              date.setDate(date.getDate() - 1);

              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfModifiedSince: date,
                  }),
                );
                state.replOutput = `${key} in bucket ${state.sourceBucketName} read
because it has been created or modified in the last 24 hours.`;
              } catch (err) {
                state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-Unmodified-Since: using yesterday's date. This condition should
fail."
            ) {
              const bucket = state.sourceBucketName;
```

```
              const key = "file01.txt";

              const date = new Date();
              date.setDate(date.getDate() - 1);
              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfUnmodifiedSince: date,
                  }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was read.`;
              } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
              }
              break;
            }
          }
          break;
        case choices.CONDITIONAL_COPY: {
          const selectedCondCopy = await condCopyOptions.handle(state);
          if (
            selectedCondCopy ===
            "If-Match: using the object's ETag. This condition should succeed."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);

            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;
            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfMatch: ETag,
                }),
```

```
              );
              state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's ETag.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-None-Match: using the object's ETag. This condition should fail."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfNoneMatch: ETag,
                }),
              );
              state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
            } catch (err) {
              state.replOutput = `Unable to copy object as ${key} as as ${copiedKey}
to bucket ${state.destinationBucketName}: ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-Modified-Since: using yesterday's date. This condition should
succeed."
          ) {
```

```
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            const date = new Date();
            date.setDate(date.getDate() - 1);

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfModifiedSince: date,
                }),
              );
              state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the last
24 hours.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-Unmodified-Since: using yesterday's date. This condition should
fail."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            const date = new Date();
            date.setDate(date.getDate() - 1);
```

```
          try {
            await client.send(
              new CopyObjectCommand({
                CopySource: copySource,
                Bucket: state.destinationBucketName,
                Key: copiedKey,
                CopySourceIfUnmodifiedSince: date,
              }),
            );
            state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in the
last 24 hours.`;
          } catch (err) {
            state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
          }
        }
        break;
      }
      case choices.CONDITIONAL_WRITE:
        {
          const selectedCondWrite = await condWriteOptions.handle(state);
          if (
            selectedCondWrite ===
            "IfNoneMatch condition on the object key: If the key is a duplicate,
the write will fail."
          ) {
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const key = "text02.txt";
            const __filename = fileURLToPath(import.meta.url);
            const __dirname = dirname(__filename);
            const filePath = path.join(__dirname, "text02.txt");
            try {
              await client.send(
                new PutObjectCommand({
                  Bucket: `${state.destinationBucketName}`,
                  Key: `${key}`,
                  Body: await readFile(filePath),
                  IfNoneMatch: "*",
                }),
              );
```

```
                state.replOutput = `${key} uploaded to bucket
 ${state.destinationBucketName} because the key is not a duplicate.`;
              } catch (err) {
                state.replOutput = `Unable to upload object to bucket
 ${state.destinationBucketName}:${err.message}`;
              }
              break;
            }
          }
          break;

        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
          preformatted: true,
        }),
      },
    },
  );

export { replInput, choices };
```

Destrua todos os recursos criados (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
```

```
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          } catch (err) {
            console.log(`An error occurred: ${err.message} `);
          }
        }
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Objects and buckets have already been deleted.");
```

```
          continue;
        }
        throw e;
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CopyObject
  - GetObject
  - PutObject

Fazer upload ou download de arquivos grandes

O exemplo de código a seguir mostra como fazer upload ou download de arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte Carregar um objeto usando carregamento fracionado.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de um arquivo grande.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
```

```
  ProgressBar,
  logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
        Bucket: bucketName,
        Key: key,
        Body: buffer,
      },
    });

    upload.on("httpUploadProgress", ({ loaded, total }) => {
      progressBar.update({ current: loaded, total });
    });

    await upload.done();
  } catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
      logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
      throw caught;
    }
```

```
  }
};
```

Baixe um arquivo grande.

```
import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));
```

```
  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento do S3 com o Lambda usando. JavaScript

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
```

```
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Consumindo um evento do S3 com o Lambda usando. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

SageMaker Exemplos de IA usando SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com SageMaker IA.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá SageMaker AI

O exemplo de código a seguir mostra como começar a usar a SageMaker IA.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  SageMakerClient,
  ListNotebookInstancesCommand,
} from "@aws-sdk/client-sagemaker";

const client = new SageMakerClient({
```

```
  region: "us-west-2",
});

export const helloSagemaker = async () => {
  const command = new ListNotebookInstancesCommand({ MaxResults: 5 });

  const response = await client.send(command);
  console.log(
    "Hello Amazon SageMaker! Let's list some of your notebook instances:",
  );

  const instances = response.NotebookInstances || [];

  if (instances.length === 0) {
    console.log(
      "• No notebook instances found. Try creating one in the AWS Management Console
 or with the CreateNotebookInstanceCommand.",
    );
  } else {
    console.log(
      instances
        .map(
          (i) =>
            `• Instance: ${i.NotebookInstanceName}\n  Arn:${
              i.NotebookInstanceArn
            } \n  Creation Date: ${i.CreationTime.toISOString()}`,
        )
        .join("\n"),
    );
  }

  return response;
};
```

- Para obter detalhes da API, consulte ListNotebookInstances em Referência AWS SDK para JavaScript da API.

Ações

`CreatePipeline`

O código de exemplo a seguir mostra como usar `CreatePipeline`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Uma função que cria um pipeline de SageMaker IA usando uma definição JSON fornecida localmente.

```
/**
 * Create the Amazon SageMaker pipeline using a JSON pipeline definition. The
 definition
 * can also be provided as an Amazon S3 object using PipelineDefinitionS3Location.
 * @param {{roleArn: string, name: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient}} props
 */
export async function createSagemakerPipeline({
  // Assumes an AWS IAM role has been created for this pipeline.
  roleArn,
  name,
  // Assumes an AWS Lambda function has been created for this pipeline.
  functionArn,
  sagemakerClient,
}) {
  const pipelineDefinition = readFileSync(
    // dirnameFromMetaUrl is a local utility function. You can find its
 implementation
    // on GitHub.
    `${dirnameFromMetaUrl(
      import.meta.url,
    )}../../../../../scenarios/features/sagemaker_pipelines/resources/
GeoSpatialPipeline.json`,
  )
    .toString()
    .replace(/\*FUNCTION_ARN\*/g, functionArn);

  let arn = null;

  const createPipeline = () =>
    sagemakerClient.send(
```

```
      new CreatePipelineCommand({
        PipelineName: name,
        PipelineDefinition: pipelineDefinition,
        RoleArn: roleArn,
      }),
    );

  try {
    const { PipelineArn } = await createPipeline();
    arn = PipelineArn;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ValidationException" &&
      caught.message.includes(
        "Pipeline names must be unique within an AWS account and region",
      )
    ) {
      const { PipelineArn } = await sagemakerClient.send(
        new DescribePipelineCommand({ PipelineName: name }),
      );
      arn = PipelineArn;
    } else {
      throw caught;
    }
  }

  return {
    arn,
    cleanUp: async () => {
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
    },
  };
}
```

- Para obter detalhes da API, consulte CreatePipeline em Referência AWS SDK para JavaScript da API.

`DeletePipeline`

O código de exemplo a seguir mostra como usar `DeletePipeline`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A sintaxe para excluir um pipeline de SageMaker IA. Esse código faz parte de uma função maior. Consulte “Criar um pipeline” ou o GitHub repositório para obter mais contexto.

```
    await sagemakerClient.send(
      new DeletePipelineCommand({ PipelineName: name }),
    );
```

- Para obter detalhes da API, consulte DeletePipeline em Referência AWS SDK para JavaScript da API.

`DescribePipelineExecution`

O código de exemplo a seguir mostra como usar `DescribePipelineExecution`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Espere que a execução de um pipeline de SageMaker IA seja bem-sucedida, falhe ou pare.

```
/**
 * Poll the executing pipeline until the status is 'SUCCEEDED', 'STOPPED', or
 'FAILED'.
```

```
 * @param {{ arn: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient, wait: (ms: number) => Promise<void>}} props
 */
export async function waitForPipelineComplete({ arn, sagemakerClient, wait }) {
  const command = new DescribePipelineExecutionCommand({
    PipelineExecutionArn: arn,
  });

  let complete = false;
  const intervalInSeconds = 15;
  const COMPLETION_STATUSES = [
    PipelineExecutionStatus.FAILED,
    PipelineExecutionStatus.STOPPED,
    PipelineExecutionStatus.SUCCEEDED,
  ];

  do {
    const { PipelineExecutionStatus: status, FailureReason } =
      await sagemakerClient.send(command);

    complete = COMPLETION_STATUSES.includes(status);

    if (!complete) {
      console.log(
        `Pipeline is ${status}. Waiting ${intervalInSeconds} seconds before checking
 again.`,
      );
      await wait(intervalInSeconds);
    } else if (status === PipelineExecutionStatus.FAILED) {
      throw new Error(`Pipeline failed because: ${FailureReason}`);
    } else if (status === PipelineExecutionStatus.STOPPED) {
      throw new Error("Pipeline was forcefully stopped.");
    } else {
      console.log(`Pipeline execution ${status}.`);
    }
  } while (!complete);
}
```

- Para obter detalhes da API, consulte DescribePipelineExecution em Referência AWS SDK para JavaScript da API.

StartPipelineExecution

O código de exemplo a seguir mostra como usar `StartPipelineExecution`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicie a execução de um pipeline de SageMaker IA.

```
/**
 * Start the execution of the Amazon SageMaker pipeline. Parameters that are
 * passed in are used in the AWS Lambda function.
 * @param {{
 *   name: string,
 *   sagemakerClient: import('@aws-sdk/client-sagemaker').SageMakerClient,
 *   roleArn: string,
 *   queueUrl: string,
 *   s3InputBucketName: string,
 * }} props
 */
export async function startPipelineExecution({
  sagemakerClient,
  name,
  bucketName,
  roleArn,
  queueUrl,
}) {
  /**
   * The Vector Enrichment Job requests CSV data. This configuration points to a CSV
   * file in an Amazon S3 bucket.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobInputConfig}
   */
  const inputConfig = {
    DataSourceConfig: {
      S3Data: {
        S3Uri: `s3://${bucketName}/input/sample_data.csv`,
      },
```

```
    },
    DocumentType: VectorEnrichmentJobDocumentType.CSV,
  };

  /**
   * The Vector Enrichment Job adds additional data to the source CSV. This
 configuration points
   * to an Amazon S3 prefix where the output will be stored.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").ExportVectorEnrichmentJobOutputConfig}
   */
  const outputConfig = {
    S3Data: {
      S3Uri: `s3://${bucketName}/output/`,
    },
  };

  /**
   * This job will be a Reverse Geocoding Vector Enrichment Job. Reverse Geocoding
 requires
   * latitude and longitude values.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobConfig}
   */
  const jobConfig = {
    ReverseGeocodingConfig: {
      XAttributeName: "Longitude",
      YAttributeName: "Latitude",
    },
  };

  const { PipelineExecutionArn } = await sagemakerClient.send(
    new StartPipelineExecutionCommand({
      PipelineName: name,
      PipelineExecutionDisplayName: `${name}-example-execution`,
      PipelineParameters: [
        { Name: "parameter_execution_role", Value: roleArn },
        { Name: "parameter_queue_url", Value: queueUrl },
        {
          Name: "parameter_vej_input_config",
          Value: JSON.stringify(inputConfig),
        },
        {
          Name: "parameter_vej_export_config",
```

```
          Value: JSON.stringify(outputConfig),
        },
        {
          Name: "parameter_step_1_vej_config",
          Value: JSON.stringify(jobConfig),
        },
      ],
    }),
  );

  return {
    arn: PipelineExecutionArn,
  };
}
```

- Para obter detalhes da API, consulte StartPipelineExecution em Referência AWS SDK para JavaScript da API.

Cenários

Conceitos básicos de trabalhos geoespaciais e pipelines

O exemplo de código a seguir mostra como:

- Configurar recursos para um pipeline.
- Configurar um pipeline que executa um trabalho geoespacial.
- Iniciar a execução de um pipeline.
- Monitorar o status da execução.
- Ver a saída do pipeline.
- Limpar recursos.

Para obter mais informações, consulte Criar e executar SageMaker pipelines usando AWS SDKs em. Community.aws

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O trecho de arquivo a seguir contém funções que usam o cliente de SageMaker IA para gerenciar um pipeline.

```
import { readFileSync } from "node:fs";

import {
  CreateRoleCommand,
  DeleteRoleCommand,
  CreatePolicyCommand,
  DeletePolicyCommand,
  AttachRolePolicyCommand,
  DetachRolePolicyCommand,
  GetRoleCommand,
  ListPoliciesCommand,
} from "@aws-sdk/client-iam";

import {
  PublishLayerVersionCommand,
  DeleteLayerVersionCommand,
  CreateFunctionCommand,
  Runtime,
  DeleteFunctionCommand,
  CreateEventSourceMappingCommand,
  DeleteEventSourceMappingCommand,
  GetFunctionCommand,
} from "@aws-sdk/client-lambda";

import {
  PutObjectCommand,
  CreateBucketCommand,
  DeleteBucketCommand,
  DeleteObjectCommand,
  GetObjectCommand,
  ListObjectsV2Command,
```

```
} from "@aws-sdk/client-s3";

import {
  CreatePipelineCommand,
  DeletePipelineCommand,
  DescribePipelineCommand,
  DescribePipelineExecutionCommand,
  PipelineExecutionStatus,
  StartPipelineExecutionCommand,
} from "@aws-sdk/client-sagemaker";

import { VectorEnrichmentJobDocumentType } from "@aws-sdk/client-sagemaker-
geospatial";

import {
  CreateQueueCommand,
  DeleteQueueCommand,
  GetQueueAttributesCommand,
  GetQueueUrlCommand,
} from "@aws-sdk/client-sqs";

import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * Create the AWS IAM role that will be assumed by AWS Lambda.
 * @param {{ name: string, iamClient: import('@aws-sdk/client-iam').IAMClient }}
 props
 */
export async function createLambdaExecutionRole({ name, iamClient }) {
  const createRole = () =>
    iamClient.send(
      new CreateRoleCommand({
        RoleName: name,
        AssumeRolePolicyDocument: JSON.stringify({
          Version: "2012-10-17",
          Statement: [
            {
              Effect: "Allow",
              Action: ["sts:AssumeRole"],
              Principal: { Service: ["lambda.amazonaws.com"] },
            },
          ],
        }),
```

```
      }),
    );

  let role = null;

  try {
    const { Role } = await createRole();
    role = Role;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Role } = await iamClient.send(
        new GetRoleCommand({ RoleName: name }),
      );
      role = Role;
    } else {
      throw caught;
    }
  }

  return {
    arn: role.Arn,
    cleanUp: async () => {
      await iamClient.send(new DeleteRoleCommand({ RoleName: name }));
    },
  };
}

/**
 * Create an AWS IAM policy that will be attached to the AWS IAM role assumed by the
 AWS Lambda function.
 * The policy grants permission to work with Amazon SQS, Amazon CloudWatch, and
 Amazon SageMaker.
 * @param {{name: string, iamClient: import('@aws-sdk/client-iam').IAMClient,
 pipelineExecutionRoleArn: string}} props
 */
export async function createLambdaExecutionPolicy({
  name,
  iamClient,
  pipelineExecutionRoleArn,
}) {
  const policyConfig = {
```

```
    Version: "2012-10-17",
    Statement: [
      {
        Effect: "Allow",
        Action: [
          "sqs:ReceiveMessage",
          "sqs:DeleteMessage",
          "sqs:GetQueueAttributes",
          "logs:CreateLogGroup",
          "logs:CreateLogStream",
          "logs:PutLogEvents",
          "sagemaker-geospatial:StartVectorEnrichmentJob",
          "sagemaker-geospatial:GetVectorEnrichmentJob",
          "sagemaker:SendPipelineExecutionStepFailure",
          "sagemaker:SendPipelineExecutionStepSuccess",
          "sagemaker-geospatial:ExportVectorEnrichmentJob",
        ],
        Resource: "*",
      },
      {
        Effect: "Allow",
        // The AWS Lambda function needs permission to pass the pipeline execution
 role to
        // the StartVectorEnrichmentCommand. This restriction prevents an AWS Lambda
 function
        // from elevating privileges. For more information, see:
        // https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_use_passrole.html
        Action: ["iam:PassRole"],
        Resource: `${pipelineExecutionRoleArn}`,
        Condition: {
          StringEquals: {
            "iam:PassedToService": [
              "sagemaker.amazonaws.com",
              "sagemaker-geospatial.amazonaws.com",
            ],
          },
        },
      },
    ],
  };

  const createPolicy = () =>
    iamClient.send(
```

```
      new CreatePolicyCommand({
        PolicyDocument: JSON.stringify(policyConfig),
        PolicyName: name,
      }),
    );

  let policy = null;

  try {
    const { Policy } = await createPolicy();
    policy = Policy;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Policies } = await iamClient.send(new ListPoliciesCommand({}));
      if (Policies) {
        policy = Policies.find((p) => p.PolicyName === name);
      } else {
        throw new Error("No policies found.");
      }
    } else {
      throw caught;
    }
  }

  return {
    arn: policy?.Arn,
    policyConfig,
    cleanUp: async () => {
      await iamClient.send(new DeletePolicyCommand({ PolicyArn: policy?.Arn }));
    },
  };
}

/**
 * Attach an AWS IAM policy to an AWS IAM role.
 * @param {{roleName: string, policyArn: string, iamClient: import('@aws-sdk/client-
iam').IAMClient}} props
 */
export async function attachPolicy({ roleName, policyArn, iamClient }) {
  const attachPolicyCommand = new AttachRolePolicyCommand({
    RoleName: roleName,
```

```
    PolicyArn: policyArn,
  });

  await iamClient.send(attachPolicyCommand);
  return {
    cleanUp: async () => {
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: roleName,
          PolicyArn: policyArn,
        }),
      );
    },
  };
}

/**
 * Create an AWS Lambda layer that contains the Amazon SageMaker and Amazon
 SageMaker Geospatial clients
 * in the runtime. The default runtime supports v3.188.0 of the JavaScript SDK. The
 Amazon SageMaker
 * Geospatial client wasn't introduced until v3.221.0.
 * @param {{ name: string, lambdaClient: import('@aws-sdk/client-
lambda').LambdaClient }} props
 */
export async function createLambdaLayer({ name, lambdaClient }) {
  const layerPath = `${dirnameFromMetaUrl(import.meta.url)}lambda/nodejs.zip`;
  const { LayerVersionArn, Version } = await lambdaClient.send(
    new PublishLayerVersionCommand({
      LayerName: name,
      Content: {
        ZipFile: Uint8Array.from(readFileSync(layerPath)),
      },
    }),
  );

  return {
    versionArn: LayerVersionArn,
    version: Version,
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteLayerVersionCommand({
          LayerName: name,
          VersionNumber: Version,
```

```
        }),
      );
    },
  };
}

/**
 * Deploy the AWS Lambda function that will be used to respond to Amazon SageMaker
 pipeline
 * execution steps.
 * @param {{roleArn: string, name: string, lambdaClient: import('@aws-sdk/client-
lambda').LambdaClient, layerVersionArn: string}} props
 */
export async function createLambdaFunction({
  name,
  roleArn,
  lambdaClient,
  layerVersionArn,
}) {
  const lambdaPath = `${dirnameFromMetaUrl(
    import.meta.url,
  )}lambda/dist/index.mjs.zip`;

  // If a function of the same name already exists, return that
  // function's ARN instead. By default this is
  // "sagemaker-wkflw-lambda-function", so collisions are
  // unlikely.
  const createFunction = async () => {
    try {
      return await lambdaClient.send(
        new CreateFunctionCommand({
          Code: {
            ZipFile: Uint8Array.from(readFileSync(lambdaPath)),
          },
          Runtime: Runtime.nodejs18x,
          Handler: "index.handler",
          Layers: [layerVersionArn],
          FunctionName: name,
          Role: roleArn,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
```

```
        caught.name === "ResourceConflictException"
      ) {
        const { Configuration } = await lambdaClient.send(
          new GetFunctionCommand({ FunctionName: name }),
        );
        return Configuration;
      }
      throw caught;
    }
  };

  // Function creation fails if the Role is not ready. This retries
  // function creation until it succeeds or it times out.
  const { FunctionArn } = await retry(
    { intervalInMs: 1000, maxRetries: 60 },
    createFunction,
  );

  return {
    arn: FunctionArn,
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteFunctionCommand({ FunctionName: name }),
      );
    },
  };
}

/**
 * This uploads some sample coordinate data to an Amazon S3 bucket.
 * The Amazon SageMaker Geospatial vector enrichment job will take the simple Lat/
Long
 * coordinates in this file and augment them with more detailed location data.
 * @param {{bucketName: string, s3Client: import('@aws-sdk/client-s3').S3Client}}
 props
 */
export async function uploadCSVDataToS3({ bucketName, s3Client }) {
  const s3Path = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../../scenarios/features/sagemaker_pipelines/resources/
latlongtest.csv`;

  await s3Client.send(
    new PutObjectCommand({
```

```
      Bucket: bucketName,
      Key: "input/sample_data.csv",
      Body: readFileSync(s3Path),
    }),
  );
}

/**
 * Create the AWS IAM role that will be assumed by the Amazon SageMaker pipeline.
 * @param {{name: string, iamClient: import('@aws-sdk/client-iam').IAMClient, wait:
 (ms: number) => Promise<void>}} props
 */
export async function createSagemakerRole({ name, iamClient, wait }) {
  let role = null;

  const createRole = () =>
    iamClient.send(
      new CreateRoleCommand({
        RoleName: name,
        AssumeRolePolicyDocument: JSON.stringify({
          Version: "2012-10-17",
          Statement: [
            {
              Effect: "Allow",
              Action: ["sts:AssumeRole"],
              Principal: {
                Service: [
                  "sagemaker.amazonaws.com",
                  "sagemaker-geospatial.amazonaws.com",
                ],
              },
            },
          ],
        }),
      }),
    );

  try {
    const { Role } = await createRole();
    role = Role;
    // Wait for the role to be ready.
    await wait(10);
  } catch (caught) {
    if (
```

```
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Role } = await iamClient.send(
        new GetRoleCommand({ RoleName: name }),
      );
      role = Role;
    } else {
      throw caught;
    }
  }

  return {
    arn: role.Arn,
    cleanUp: async () => {
      await iamClient.send(new DeleteRoleCommand({ RoleName: name }));
    },
  };
}

/**
 * Create the Amazon SageMaker execution policy. This policy grants permission to
 * invoke the AWS Lambda function, read/write to the Amazon S3 bucket, and send
 messages to
 * the Amazon SQS queue.
 * @param {{ name: string, sqsQueueArn: string, lambdaArn: string, iamClient:
 import('@aws-sdk/client-iam').IAMClient, s3BucketName: string}} props
 */
export async function createSagemakerExecutionPolicy({
  sqsQueueArn,
  lambdaArn,
  iamClient,
  name,
  s3BucketName,
}) {
  const policyConfig = {
    Version: "2012-10-17",
    Statement: [
      {
        Effect: "Allow",
        Action: ["lambda:InvokeFunction"],
        Resource: lambdaArn,
      },
      {
```

```
        Effect: "Allow",
        Action: ["s3:*"],
        Resource: [
          `arn:aws:s3:::${s3BucketName}`,
          `arn:aws:s3:::${s3BucketName}/*`,
        ],
      },
      {
        Effect: "Allow",
        Action: ["sqs:SendMessage"],
        Resource: sqsQueueArn,
      },
    ],
  };

  const createPolicy = () =>
    iamClient.send(
      new CreatePolicyCommand({
        PolicyDocument: JSON.stringify(policyConfig),
        PolicyName: name,
      }),
    );

  let policy = null;

  try {
    const { Policy } = await createPolicy();
    policy = Policy;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Policies } = await iamClient.send(new ListPoliciesCommand({}));
      if (Policies) {
        policy = Policies.find((p) => p.PolicyName === name);
      } else {
        throw new Error("No policies found.");
      }
    } else {
      throw caught;
    }
  }
```

```
  return {
    arn: policy?.Arn,
    policyConfig,
    cleanUp: async () => {
      await iamClient.send(new DeletePolicyCommand({ PolicyArn: policy?.Arn }));
    },
  };
}

/**
 * Create the Amazon SageMaker pipeline using a JSON pipeline definition. The
 definition
 * can also be provided as an Amazon S3 object using PipelineDefinitionS3Location.
 * @param {{roleArn: string, name: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient}} props
 */
export async function createSagemakerPipeline({
  // Assumes an AWS IAM role has been created for this pipeline.
  roleArn,
  name,
  // Assumes an AWS Lambda function has been created for this pipeline.
  functionArn,
  sagemakerClient,
}) {
  const pipelineDefinition = readFileSync(
    // dirnameFromMetaUrl is a local utility function. You can find its
 implementation
    // on GitHub.
    `${dirnameFromMetaUrl(
      import.meta.url,
    )}../../../../../scenarios/features/sagemaker_pipelines/resources/
GeoSpatialPipeline.json`,
  )
    .toString()
    .replace(/\*FUNCTION_ARN\*/g, functionArn);

  let arn = null;

  const createPipeline = () =>
    sagemakerClient.send(
      new CreatePipelineCommand({
        PipelineName: name,
        PipelineDefinition: pipelineDefinition,
        RoleArn: roleArn,
```

```
      }),
    );

  try {
    const { PipelineArn } = await createPipeline();
    arn = PipelineArn;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ValidationException" &&
      caught.message.includes(
        "Pipeline names must be unique within an AWS account and region",
      )
    ) {
      const { PipelineArn } = await sagemakerClient.send(
        new DescribePipelineCommand({ PipelineName: name }),
      );
      arn = PipelineArn;
    } else {
      throw caught;
    }
  }

  return {
    arn,
    cleanUp: async () => {
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
    },
  };
}

/**
 * Create an Amazon SQS queue. The Amazon SageMaker pipeline will send messages
 * to this queue that are then processed by the AWS Lambda function.
 * @param {{name: string, sqsClient: import('@aws-sdk/client-sqs').SQSClient}} props
 */
export async function createSQSQueue({ name, sqsClient }) {
  const createSqsQueue = () =>
    sqsClient.send(
      new CreateQueueCommand({
        QueueName: name,
        Attributes: {
```

```
          DelaySeconds: "5",
          ReceiveMessageWaitTimeSeconds: "5",
          VisibilityTimeout: "300",
        },
      }),
    );

  let queueUrl = null;
  try {
    const { QueueUrl } = await createSqsQueue();
    queueUrl = QueueUrl;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "QueueNameExists") {
      const { QueueUrl } = await sqsClient.send(
        new GetQueueUrlCommand({ QueueName: name }),
      );
      queueUrl = QueueUrl;
    } else {
      throw caught;
    }
  }

  const { Attributes } = await retry(
    { intervalInMs: 1000, maxRetries: 60 },
    () =>
      sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: queueUrl,
          AttributeNames: ["QueueArn"],
        }),
      ),
  );

  return {
    queueUrl,
    queueArn: Attributes.QueueArn,
    cleanUp: async () => {
      await sqsClient.send(new DeleteQueueCommand({ QueueUrl: queueUrl }));
    },
  };
}

/**
 * Configure the AWS Lambda function to long poll for messages from the Amazon SQS
```

```
 * queue.
 * @param {{
 *   paginateListEventSourceMappings: () => Generator<import('@aws-sdk/client-
lambda').ListEventSourceMappingsCommandOutput>,
 *   lambdaName: string,
 *   queueArn: string,
 *   lambdaClient: import('@aws-sdk/client-lambda').LambdaClient}} props
 */
export async function configureLambdaSQSEventSource({
  lambdaName,
  queueArn,
  lambdaClient,
  paginateListEventSourceMappings,
}) {
  let uuid = null;
  const createEvenSourceMapping = () =>
    lambdaClient.send(
      new CreateEventSourceMappingCommand({
        EventSourceArn: queueArn,
        FunctionName: lambdaName,
      }),
    );

  try {
    const { UUID } = await createEvenSourceMapping();
    uuid = UUID;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceConflictException"
    ) {
      const paginator = paginateListEventSourceMappings(
        { client: lambdaClient },
        {},
      );
      /**
       * @type {import('@aws-sdk/client-lambda').EventSourceMappingConfiguration[]}
       */
      const eventSourceMappings = [];
      for await (const page of paginator) {
        eventSourceMappings.concat(page.EventSourceMappings || []);
      }

      const { Configuration } = await lambdaClient.send(
```

```
        new GetFunctionCommand({ FunctionName: lambdaName }),
      );

      uuid = eventSourceMappings.find(
        (mapping) =>
          mapping.EventSourceArn === queueArn &&
          mapping.FunctionArn === Configuration.FunctionArn,
      ).UUID;
    } else {
      throw caught;
    }
  }

  return {
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteEventSourceMappingCommand({
          UUID: uuid,
        }),
      );
    },
  };
}

/**
 * Create an Amazon S3 bucket that will store the simple coordinate file as input
 * and the output of the Amazon SageMaker Geospatial vector enrichment job.
 * @param {{
 *   s3Client: import('@aws-sdk/client-s3').S3Client,
 *   name: string,
 *   paginateListObjectsV2: () => Generator<import('@aws-sdk/client-
s3').ListObjectsCommandOutput>
 * }} props
 */
export async function createS3Bucket({
  name,
  s3Client,
  paginateListObjectsV2,
}) {
  await s3Client.send(new CreateBucketCommand({ Bucket: name }));

  return {
    cleanUp: async () => {
      const paginator = paginateListObjectsV2(
```

```
        { client: s3Client },
        { Bucket: name },
      );
      for await (const page of paginator) {
        const objects = page.Contents;
        if (objects) {
          for (const object of objects) {
            await s3Client.send(
              new DeleteObjectCommand({ Bucket: name, Key: object.Key }),
            );
          }
        }
      }
      await s3Client.send(new DeleteBucketCommand({ Bucket: name }));
    },
  };
}

/**
 * Start the execution of the Amazon SageMaker pipeline. Parameters that are
 * passed in are used in the AWS Lambda function.
 * @param {{
 *   name: string,
 *   sagemakerClient: import('@aws-sdk/client-sagemaker').SageMakerClient,
 *   roleArn: string,
 *   queueUrl: string,
 *   s3InputBucketName: string,
 * }} props
 */
export async function startPipelineExecution({
  sagemakerClient,
  name,
  bucketName,
  roleArn,
  queueUrl,
}) {
  /**
   * The Vector Enrichment Job requests CSV data. This configuration points to a CSV
   * file in an Amazon S3 bucket.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobInputConfig}
   */
  const inputConfig = {
    DataSourceConfig: {
```

```
      S3Data: {
        S3Uri: `s3://${bucketName}/input/sample_data.csv`,
      },
    },
    DocumentType: VectorEnrichmentJobDocumentType.CSV,
  };

  /**
   * The Vector Enrichment Job adds additional data to the source CSV. This
 configuration points
   * to an Amazon S3 prefix where the output will be stored.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").ExportVectorEnrichmentJobOutputConfig}
   */
  const outputConfig = {
    S3Data: {
      S3Uri: `s3://${bucketName}/output/`,
    },
  };

  /**
   * This job will be a Reverse Geocoding Vector Enrichment Job. Reverse Geocoding
 requires
   * latitude and longitude values.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobConfig}
   */
  const jobConfig = {
    ReverseGeocodingConfig: {
      XAttributeName: "Longitude",
      YAttributeName: "Latitude",
    },
  };

  const { PipelineExecutionArn } = await sagemakerClient.send(
    new StartPipelineExecutionCommand({
      PipelineName: name,
      PipelineExecutionDisplayName: `${name}-example-execution`,
      PipelineParameters: [
        { Name: "parameter_execution_role", Value: roleArn },
        { Name: "parameter_queue_url", Value: queueUrl },
        {
          Name: "parameter_vej_input_config",
          Value: JSON.stringify(inputConfig),
```

```
        },
        {
          Name: "parameter_vej_export_config",
          Value: JSON.stringify(outputConfig),
        },
        {
          Name: "parameter_step_1_vej_config",
          Value: JSON.stringify(jobConfig),
        },
      ],
    }),
  );

  return {
    arn: PipelineExecutionArn,
  };
}

/**
 * Poll the executing pipeline until the status is 'SUCCEEDED', 'STOPPED', or
 'FAILED'.
 * @param {{ arn: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient, wait: (ms: number) => Promise<void>}} props
 */
export async function waitForPipelineComplete({ arn, sagemakerClient, wait }) {
  const command = new DescribePipelineExecutionCommand({
    PipelineExecutionArn: arn,
  });

  let complete = false;
  const intervalInSeconds = 15;
  const COMPLETION_STATUSES = [
    PipelineExecutionStatus.FAILED,
    PipelineExecutionStatus.STOPPED,
    PipelineExecutionStatus.SUCCEEDED,
  ];

  do {
    const { PipelineExecutionStatus: status, FailureReason } =
      await sagemakerClient.send(command);

    complete = COMPLETION_STATUSES.includes(status);

    if (!complete) {
```

```
      console.log(
        `Pipeline is ${status}. Waiting ${intervalInSeconds} seconds before checking
 again.`,
      );
      await wait(intervalInSeconds);
    } else if (status === PipelineExecutionStatus.FAILED) {
      throw new Error(`Pipeline failed because: ${FailureReason}`);
    } else if (status === PipelineExecutionStatus.STOPPED) {
      throw new Error("Pipeline was forcefully stopped.");
    } else {
      console.log(`Pipeline execution ${status}.`);
    }
  } while (!complete);
}

/**
 * Return the string value of an Amazon S3 object.
 * @param {{ bucket: string, key: string, s3Client: import('@aws-sdk/client-
s3').S3Client}} param0
 */
export async function getObject({ bucket, s3Client }) {
  const prefix = "output/";
  const { Contents } = await s3Client.send(
    new ListObjectsV2Command({ MaxKeys: 1, Bucket: bucket, Prefix: prefix }),
  );

  if (!Contents.length) {
    throw new Error("No objects found in bucket.");
  }

  // Find the CSV file.
  const outputObject = Contents.find((obj) => obj.Key.endsWith(".csv"));

  if (!outputObject) {
    throw new Error(`No CSV file found in bucket with the prefix "${prefix}".`);
  }

  const { Body } = await s3Client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: outputObject.Key,
    }),
  );
```

```
  return Body.transformToString();
}
```

Essa função é um trecho de um arquivo que usa as funções de biblioteca anteriores para configurar um pipeline de SageMaker IA, executá-lo e excluir todos os recursos criados.

```
import { retry, wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  attachPolicy,
  configureLambdaSQSEventSource,
  createLambdaExecutionPolicy,
  createLambdaExecutionRole,
  createLambdaFunction,
  createLambdaLayer,
  createS3Bucket,
  createSQSQueue,
  createSagemakerExecutionPolicy,
  createSagemakerPipeline,
  createSagemakerRole,
  getObject,
  startPipelineExecution,
  uploadCSVDataToS3,
  waitForPipelineComplete,
} from "./lib.js";
import { MESSAGES } from "./messages.js";

export class SageMakerPipelinesWkflw {
  names = {
    LAMBDA_EXECUTION_ROLE: "sagemaker-wkflw-lambda-execution-role",
    LAMBDA_EXECUTION_ROLE_POLICY:
      "sagemaker-wkflw-lambda-execution-role-policy",
    LAMBDA_FUNCTION: "sagemaker-wkflw-lambda-function",
    LAMBDA_LAYER: "sagemaker-wkflw-lambda-layer",
    SAGE_MAKER_EXECUTION_ROLE: "sagemaker-wkflw-pipeline-execution-role",
    SAGE_MAKER_EXECUTION_ROLE_POLICY:
      "sagemaker-wkflw-pipeline-execution-role-policy",
    SAGE_MAKER_PIPELINE: "sagemaker-wkflw-pipeline",
    SQS_QUEUE: "sagemaker-wkflw-sqs-queue",
    S3_BUCKET: `sagemaker-wkflw-s3-bucket-${Date.now()}`,
  };

  cleanUpFunctions = [];
```

```
  /**
   * @param {import("@aws-doc-sdk-examples/lib/prompter.js").Prompter} prompter
   * @param {import("@aws-doc-sdk-examples/lib/logger.js").Logger} logger
   * @param {{ IAM: import("@aws-sdk/client-iam").IAMClient, Lambda: import("@aws-
sdk/client-lambda").LambdaClient, SageMaker: import("@aws-sdk/client-
sagemaker").SageMakerClient, S3: import("@aws-sdk/client-s3").S3Client, SQS:
 import("@aws-sdk/client-sqs").SQSClient }} clients
   */
  constructor(prompter, logger, clients) {
    this.prompter = prompter;
    this.logger = logger;
    this.clients = clients;
  }

  async run() {
    try {
      await this.startWorkflow();
    } catch (err) {
      console.error(err);
      throw err;
    } finally {
      this.logger.logSeparator();
      const doCleanUp = await this.prompter.confirm({
        message: "Clean up resources?",
      });
      if (doCleanUp) {
        await this.cleanUp();
      }
    }
  }

  async cleanUp() {
    // Run all of the clean up functions. If any fail, we log the error and
 continue.
    // This ensures all clean up functions are run.
    for (let i = this.cleanUpFunctions.length - 1; i >= 0; i--) {
      await retry(
        { intervalInMs: 1000, maxRetries: 60, swallowError: true },
        this.cleanUpFunctions[i],
      );
    }
  }
```

```
 async startWorkflow() {
   this.logger.logSeparator(MESSAGES.greetingHeader);
   await this.logger.log(MESSAGES.greeting);

   this.logger.logSeparator();
   await this.logger.log(
     MESSAGES.creatingRole.replace(
       "${ROLE_NAME}",
       this.names.LAMBDA_EXECUTION_ROLE,
     ),
   );

   // Create an IAM role that will be assumed by the AWS Lambda function. This
function
   // is triggered by Amazon SQS messages and calls SageMaker and SageMaker
GeoSpatial actions.
   const { arn: lambdaExecutionRoleArn, cleanUp: lambdaExecutionRoleCleanUp } =
     await createLambdaExecutionRole({
       name: this.names.LAMBDA_EXECUTION_ROLE,
       iamClient: this.clients.IAM,
     });
   // Add a clean up step to a stack for every resource created.
   this.cleanUpFunctions.push(lambdaExecutionRoleCleanUp);

   await this.logger.log(
     MESSAGES.roleCreated.replace(
       "${ROLE_NAME}",
       this.names.LAMBDA_EXECUTION_ROLE,
     ),
   );

   this.logger.logSeparator();

   await this.logger.log(
     MESSAGES.creatingRole.replace(
       "${ROLE_NAME}",
       this.names.SAGE_MAKER_EXECUTION_ROLE,
     ),
   );

   // Create an IAM role that will be assumed by the SageMaker pipeline. The
pipeline
   // sends messages to an Amazon SQS queue and puts/retrieves Amazon S3 objects.
   const {
```

```
      arn: pipelineExecutionRoleArn,
      cleanUp: pipelineExecutionRoleCleanUp,
    } = await createSagemakerRole({
      iamClient: this.clients.IAM,
      name: this.names.SAGE_MAKER_EXECUTION_ROLE,
      wait,
    });
    this.cleanUpFunctions.push(pipelineExecutionRoleCleanUp);

    await this.logger.log(
      MESSAGES.roleCreated.replace(
        "${ROLE_NAME}",
        this.names.SAGE_MAKER_EXECUTION_ROLE,
      ),
    );

    this.logger.logSeparator();

    // Create an IAM policy that allows the AWS Lambda function to invoke SageMaker
APIs.
    const {
      arn: lambdaExecutionPolicyArn,
      policy: lambdaPolicy,
      cleanUp: lambdaExecutionPolicyCleanUp,
    } = await createLambdaExecutionPolicy({
      name: this.names.LAMBDA_EXECUTION_ROLE_POLICY,
      s3BucketName: this.names.S3_BUCKET,
      iamClient: this.clients.IAM,
      pipelineExecutionRoleArn,
    });
    this.cleanUpFunctions.push(lambdaExecutionPolicyCleanUp);

    console.log(JSON.stringify(lambdaPolicy, null, 2), "\n");

    await this.logger.log(
      MESSAGES.attachPolicy
        .replace("${POLICY_NAME}", this.names.LAMBDA_EXECUTION_ROLE_POLICY)
        .replace("${ROLE_NAME}", this.names.LAMBDA_EXECUTION_ROLE),
    );

    await this.prompter.checkContinue();

    // Attach the Lambda execution policy to the execution role.
    const { cleanUp: lambdaExecutionRolePolicyCleanUp } = await attachPolicy({
```

```
      roleName: this.names.LAMBDA_EXECUTION_ROLE,
      policyArn: lambdaExecutionPolicyArn,
      iamClient: this.clients.IAM,
    });
    this.cleanUpFunctions.push(lambdaExecutionRolePolicyCleanUp);

    await this.logger.log(MESSAGES.policyAttached);

    this.logger.logSeparator();

    // Create Lambda layer for SageMaker packages.
    const { versionArn: layerVersionArn, cleanUp: lambdaLayerCleanUp } =
      await createLambdaLayer({
        name: this.names.LAMBDA_LAYER,
        lambdaClient: this.clients.Lambda,
      });
    this.cleanUpFunctions.push(lambdaLayerCleanUp);

    await this.logger.log(
      MESSAGES.creatingFunction.replace(
        "${FUNCTION_NAME}",
        this.names.LAMBDA_FUNCTION,
      ),
    );

    // Create the Lambda function with the execution role.
    const { arn: lambdaArn, cleanUp: lambdaCleanUp } =
      await createLambdaFunction({
        roleArn: lambdaExecutionRoleArn,
        lambdaClient: this.clients.Lambda,
        name: this.names.LAMBDA_FUNCTION,
        layerVersionArn,
      });
    this.cleanUpFunctions.push(lambdaCleanUp);

    await this.logger.log(
      MESSAGES.functionCreated.replace(
        "${FUNCTION_NAME}",
        this.names.LAMBDA_FUNCTION,
      ),
    );

    this.logger.logSeparator();
```

```
    await this.logger.log(
      MESSAGES.creatingSQSQueue.replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    // Create an SQS queue for the SageMaker pipeline.
    const {
      queueUrl,
      queueArn,
      cleanUp: queueCleanUp,
    } = await createSQSQueue({
      name: this.names.SQS_QUEUE,
      sqsClient: this.clients.SQS,
    });
    this.cleanUpFunctions.push(queueCleanUp);

    await this.logger.log(
      MESSAGES.sqsQueueCreated.replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.configuringLambdaSQSEventSource
        .replace("${LAMBDA_NAME}", this.names.LAMBDA_FUNCTION)
        .replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    // Configure the SQS queue as an event source for the Lambda.
    const { cleanUp: lambdaSQSEventSourceCleanUp } =
      await configureLambdaSQSEventSource({
        lambdaArn,
        lambdaName: this.names.LAMBDA_FUNCTION,
        queueArn,
        sqsClient: this.clients.SQS,
        lambdaClient: this.clients.Lambda,
      });
    this.cleanUpFunctions.push(lambdaSQSEventSourceCleanUp);

    await this.logger.log(
      MESSAGES.lambdaSQSEventSourceConfigured
        .replace("${LAMBDA_NAME}", this.names.LAMBDA_FUNCTION)
        .replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );
```

```
    this.logger.logSeparator();

    // Create an IAM policy that allows the SageMaker pipeline to invoke AWS Lambda
    // and send messages to the Amazon SQS queue.
    const {
      arn: pipelineExecutionPolicyArn,
      policy: sagemakerPolicy,
      cleanUp: pipelineExecutionPolicyCleanUp,
    } = await createSagemakerExecutionPolicy({
      sqsQueueArn: queueArn,
      lambdaArn,
      iamClient: this.clients.IAM,
      name: this.names.SAGE_MAKER_EXECUTION_ROLE_POLICY,
      s3BucketName: this.names.S3_BUCKET,
    });
    this.cleanUpFunctions.push(pipelineExecutionPolicyCleanUp);

    console.log(JSON.stringify(sagemakerPolicy, null, 2));

    await this.logger.log(
      MESSAGES.attachPolicy
        .replace("${POLICY_NAME}", this.names.SAGE_MAKER_EXECUTION_ROLE_POLICY)
        .replace("${ROLE_NAME}", this.names.SAGE_MAKER_EXECUTION_ROLE),
    );

    await this.prompter.checkContinue();

    // Attach the SageMaker execution policy to the execution role.
    const { cleanUp: pipelineExecutionRolePolicyCleanUp } = await attachPolicy({
      roleName: this.names.SAGE_MAKER_EXECUTION_ROLE,
      policyArn: pipelineExecutionPolicyArn,
      iamClient: this.clients.IAM,
    });
    this.cleanUpFunctions.push(pipelineExecutionRolePolicyCleanUp);
    // Wait for the role to be ready. If the role is used immediately,
    // the pipeline will fail.
    await wait(5);

    await this.logger.log(MESSAGES.policyAttached);

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingPipeline.replace(
```

```
        "${PIPELINE_NAME}",
        this.names.SAGE_MAKER_PIPELINE,
      ),
    );

    // Create the SageMaker pipeline.
    const { cleanUp: pipelineCleanUp } = await createSagemakerPipeline({
      roleArn: pipelineExecutionRoleArn,
      functionArn: lambdaArn,
      sagemakerClient: this.clients.SageMaker,
      name: this.names.SAGE_MAKER_PIPELINE,
    });
    this.cleanUpFunctions.push(pipelineCleanUp);

    await this.logger.log(
      MESSAGES.pipelineCreated.replace(
        "${PIPELINE_NAME}",
        this.names.SAGE_MAKER_PIPELINE,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingS3Bucket.replace("${BUCKET_NAME}", this.names.S3_BUCKET),
    );

    // Create an S3 bucket for storing inputs and outputs.
    const { cleanUp: s3BucketCleanUp } = await createS3Bucket({
      name: this.names.S3_BUCKET,
      s3Client: this.clients.S3,
    });
    this.cleanUpFunctions.push(s3BucketCleanUp);

    await this.logger.log(
      MESSAGES.s3BucketCreated.replace("${BUCKET_NAME}", this.names.S3_BUCKET),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.uploadingInputData.replace(
        "${BUCKET_NAME}",
        this.names.S3_BUCKET,
```

```
    ),
  );

  // Upload CSV Lat/Long data to S3.
  await uploadCSVDataToS3({
    bucketName: this.names.S3_BUCKET,
    s3Client: this.clients.S3,
  });

  await this.logger.log(MESSAGES.inputDataUploaded);

  this.logger.logSeparator();

  await this.prompter.checkContinue(MESSAGES.executePipeline);

  // Execute the SageMaker pipeline.
  const { arn: pipelineExecutionArn } = await startPipelineExecution({
    name: this.names.SAGE_MAKER_PIPELINE,
    sagemakerClient: this.clients.SageMaker,
    roleArn: pipelineExecutionRoleArn,
    bucketName: this.names.S3_BUCKET,
    queueUrl,
  });

  // Wait for the pipeline execution to finish.
  await waitForPipelineComplete({
    arn: pipelineExecutionArn,
    sagemakerClient: this.clients.SageMaker,
    wait,
  });

  this.logger.logSeparator();

  await this.logger.log(MESSAGES.outputDelay);

  // The getOutput function will throw an error if the output is not
  // found. The retry function will retry a failed function call once
  // ever 10 seconds for 2 minutes.
  const output = await retry({ intervalInMs: 10000, maxRetries: 12 }, () =>
    getObject({
      bucket: this.names.S3_BUCKET,
      s3Client: this.clients.S3,
    }),
  );
```

```
    this.logger.logSeparator();
    await this.logger.log(MESSAGES.outputDataRetrieved);
    console.log(output.split("\n").slice(0, 6).join("\n"));
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Exemplos do Secrets Manager usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetSecretValue`

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  GetSecretValueCommand,
  SecretsManagerClient,
} from "@aws-sdk/client-secrets-manager";

export const getSecretValue = async (secretName = "SECRET_NAME") => {
  const client = new SecretsManagerClient();
  const response = await client.send(
    new GetSecretValueCommand({
      SecretId: secretName,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '584eb612-f8b0-48c9-855e-6d246461b604',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   ARN: 'arn:aws:secretsmanager:us-east-1:xxxxxxxxxxxx:secret:binary-
secret-3873048-xxxxxx',
  //   CreatedDate: 2023-08-08T19:29:51.294Z,
  //   Name: 'binary-secret-3873048',
  //   SecretBinary: Uint8Array(11) [
  //      98, 105, 110, 97, 114,
  //     121,  32, 100, 97, 116,
  //      97
  //   ],
  //   VersionId: '712083f4-0d26-415e-8044-16735142cd6a',
  //   VersionStages: [ 'AWSCURRENT' ]
  // }
```

```
  if (response.SecretString) {
    return response.SecretString;
  }

  if (response.SecretBinary) {
    return response.SecretBinary;
  }
};
```

- Para obter detalhes da API, consulte GetSecretValue em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon SES usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateReceiptFilter`

O código de exemplo a seguir mostra como usar `CreateReceiptFilter`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  CreateReceiptFilterCommand,
  ReceiptFilterPolicy,
} from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const createCreateReceiptFilterCommand = ({ policy, ipOrRange, name }) => {
  return new CreateReceiptFilterCommand({
    Filter: {
      IpFilter: {
        Cidr: ipOrRange, // string, either a single IP address (10.0.0.1) or an IP
 address range in CIDR notation (10.0.0.1/24)).
        Policy: policy, // enum ReceiptFilterPolicy, email traffic from the filtered
 addressesOptions.
      },
      /*
        The name of the IP address filter. Only ASCII letters, numbers, underscores,
 or dashes.
        Must be less than 64 characters and start and end with a letter or number.
       */
      Name: name,
    },
  });
};

const FILTER_NAME = getUniqueName("ReceiptFilter");

const run = async () => {
  const createReceiptFilterCommand = createCreateReceiptFilterCommand({
    policy: ReceiptFilterPolicy.Allow,
    ipOrRange: "10.0.0.1",
    name: FILTER_NAME,
  });
```

```
  try {
    return await sesClient.send(createReceiptFilterCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte CreateReceiptFilter em Referência AWS SDK para JavaScript da API.

`CreateReceiptRule`

O código de exemplo a seguir mostra como usar `CreateReceiptRule`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateReceiptRuleCommand, TlsPolicy } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");
const RULE_NAME = getUniqueName("RuleName");
const S3_BUCKET_NAME = getUniqueName("S3BucketName");

const createS3ReceiptRuleCommand = ({
  bucketName,
  emailAddresses,
```

```
  name,
  ruleSet,
}) => {
  return new CreateReceiptRuleCommand({
    Rule: {
      Actions: [
        {
          S3Action: {
            BucketName: bucketName,
            ObjectKeyPrefix: "email",
          },
        },
      ],
      Recipients: emailAddresses,
      Enabled: true,
      Name: name,
      ScanEnabled: false,
      TlsPolicy: TlsPolicy.Optional,
    },
    RuleSetName: ruleSet, // Required
  });
};

const run = async () => {
  const s3ReceiptRuleCommand = createS3ReceiptRuleCommand({
    bucketName: S3_BUCKET_NAME,
    emailAddresses: ["email@example.com"],
    name: RULE_NAME,
    ruleSet: RULE_SET_NAME,
  });

  try {
    return await sesClient.send(s3ReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to create S3 receipt rule.", err);
    throw err;
  }
};
```

- Para obter detalhes da API, consulte CreateReceiptRule em Referência AWS SDK para JavaScript da API.

`CreateReceiptRuleSet`

O código de exemplo a seguir mostra como usar `CreateReceiptRuleSet`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createCreateReceiptRuleSetCommand = (ruleSetName) => {
  return new CreateReceiptRuleSetCommand({ RuleSetName: ruleSetName });
};

const run = async () => {
  const createReceiptRuleSetCommand =
    createCreateReceiptRuleSetCommand(RULE_SET_NAME);

  try {
    return await sesClient.send(createReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to create receipt rule set", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte CreateReceiptRuleSet em Referência AWS SDK para JavaScript da API.

`CreateTemplate`

O código de exemplo a seguir mostra como usar `CreateTemplate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateTemplateCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const TEMPLATE_NAME = getUniqueName("TestTemplateName");

const createCreateTemplateCommand = () => {
  return new CreateTemplateCommand({
    /**
     * The template feature in Amazon SES is based on the Handlebars template
 system.
     */
    Template: {
      /**
       * The name of an existing template in Amazon SES.
       */
      TemplateName: TEMPLATE_NAME,
      HtmlPart: `
        <h1>Hello, {{contact.firstName}}!</h1>
        <p>
        Did you know Amazon has a mascot named Peccy?
        </p>
      `,
      SubjectPart: "Amazon Tip",
    },
  });
};

const run = async () => {
  const createTemplateCommand = createCreateTemplateCommand();

  try {
    return await sesClient.send(createTemplateCommand);
```

```
  } catch (err) {
    console.log("Failed to create template.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte CreateTemplate em Referência AWS SDK para JavaScript da API.

`DeleteIdentity`

O código de exemplo a seguir mostra como usar `DeleteIdentity`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const IDENTITY_EMAIL = "fake@example.com";

const createDeleteIdentityCommand = (identityName) => {
  return new DeleteIdentityCommand({
    Identity: identityName,
  });
};

const run = async () => {
  const deleteIdentityCommand = createDeleteIdentityCommand(IDENTITY_EMAIL);

  try {
    return await sesClient.send(deleteIdentityCommand);
  } catch (err) {
    console.log("Failed to delete identity.", err);
    return err;
```

```
  }
};
```

- Para obter detalhes da API, consulte DeleteIdentity em Referência AWS SDK para JavaScript da API.

`DeleteReceiptFilter`

O código de exemplo a seguir mostra como usar `DeleteReceiptFilter`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteReceiptFilterCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RECEIPT_FILTER_NAME = getUniqueName("ReceiptFilterName");

const createDeleteReceiptFilterCommand = (filterName) => {
  return new DeleteReceiptFilterCommand({ FilterName: filterName });
};

const run = async () => {
  const deleteReceiptFilterCommand =
    createDeleteReceiptFilterCommand(RECEIPT_FILTER_NAME);

  try {
    return await sesClient.send(deleteReceiptFilterCommand);
  } catch (err) {
    console.log("Error deleting receipt filter.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteReceiptFilter em Referência AWS SDK para JavaScript da API.

`DeleteReceiptRule`

O código de exemplo a seguir mostra como usar `DeleteReceiptRule`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteReceiptRuleCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_NAME = getUniqueName("RuleName");
const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleCommand = () => {
  return new DeleteReceiptRuleCommand({
    RuleName: RULE_NAME,
    RuleSetName: RULE_SET_NAME,
  });
};

const run = async () => {
  const deleteReceiptRuleCommand = createDeleteReceiptRuleCommand();
  try {
    return await sesClient.send(deleteReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteReceiptRule em Referência AWS SDK para JavaScript da API.

`DeleteReceiptRuleSet`

O código de exemplo a seguir mostra como usar `DeleteReceiptRuleSet`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleSetCommand = () => {
  return new DeleteReceiptRuleSetCommand({ RuleSetName: RULE_SET_NAME });
};

const run = async () => {
  const deleteReceiptRuleSetCommand = createDeleteReceiptRuleSetCommand();

  try {
    return await sesClient.send(deleteReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule set.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteReceiptRuleSet em Referência AWS SDK para JavaScript da API.

`DeleteTemplate`

O código de exemplo a seguir mostra como usar `DeleteTemplate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createDeleteTemplateCommand = (templateName) =>
  new DeleteTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const deleteTemplateCommand = createDeleteTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(deleteTemplateCommand);
  } catch (err) {
    console.log("Failed to delete template.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte DeleteTemplate em Referência AWS SDK para JavaScript da API.

`GetTemplate`

O código de exemplo a seguir mostra como usar `GetTemplate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { GetTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createGetTemplateCommand = (templateName) =>
  new GetTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const getTemplateCommand = createGetTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(getTemplateCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte GetTemplate em Referência AWS SDK para JavaScript da API.

`ListIdentities`

O código de exemplo a seguir mostra como usar `ListIdentities`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ListIdentitiesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListIdentitiesCommand = () =>
  new ListIdentitiesCommand({ IdentityType: "EmailAddress", MaxItems: 10 });

const run = async () => {
  const listIdentitiesCommand = createListIdentitiesCommand();

  try {
    return await sesClient.send(listIdentitiesCommand);
  } catch (err) {
    console.log("Failed to list identities.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte ListIdentities em Referência AWS SDK para JavaScript da API.

`ListReceiptFilters`

O código de exemplo a seguir mostra como usar `ListReceiptFilters`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ListReceiptFiltersCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListReceiptFiltersCommand = () => new ListReceiptFiltersCommand({});

const run = async () => {
  const listReceiptFiltersCommand = createListReceiptFiltersCommand();

  return await sesClient.send(listReceiptFiltersCommand);
};
```

- Para obter detalhes da API, consulte ListReceiptFilters em Referência AWS SDK para JavaScript da API.

`ListTemplates`

O código de exemplo a seguir mostra como usar `ListTemplates`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ListTemplatesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListTemplatesCommand = (maxItems) =>
  new ListTemplatesCommand({ MaxItems: maxItems });

const run = async () => {
  const listTemplatesCommand = createListTemplatesCommand(10);

  try {
    return await sesClient.send(listTemplatesCommand);
  } catch (err) {
    console.log("Failed to list templates.", err);
    return err;
```

```
  }
};
```

- Para obter detalhes da API, consulte ListTemplates em Referência AWS SDK para JavaScript da API.

`SendBulkTemplatedEmail`

O código de exemplo a seguir mostra como usar `SendBulkTemplatedEmail`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SendBulkTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL_1 = postfix(getUniqueName("Bilbo"), "@example.com");
const VERIFIED_EMAIL_2 = postfix(getUniqueName("Frodo"), "@example.com");

const USERS = [
  { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL_1 },
  { firstName: "Frodo", emailAddress: VERIFIED_EMAIL_2 },
];
```

```
/**
 *
 * @param { { emailAddress: string, firstName: string }[] } users
 * @param { string } templateName the name of an existing template in SES
 * @returns { SendBulkTemplatedEmailCommand }
 */
const createBulkReminderEmailCommand = (users, templateName) => {
  return new SendBulkTemplatedEmailCommand({
    /**
     * Each 'Destination' uses a corresponding set of replacement data. We can map
 each user
     * to a 'Destination' and provide user specific replacement data to create
 personalized emails.
     *
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{name}},</h1><p>Don't forget about the party gifts!</p>
     * Destination 1: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!</
p>
     * Destination 2: <h1>Hello Frodo,</h1><p>Don't forget about the party gifts!</
p>
     */
    Destinations: users.map((user) => ({
      Destination: { ToAddresses: [user.emailAddress] },
      ReplacementTemplateData: JSON.stringify({ name: user.firstName }),
    })),
    DefaultTemplateData: JSON.stringify({ name: "Shireling" }),
    Source: VERIFIED_EMAIL_1,
    Template: templateName,
  });
};

const run = async () => {
  const sendBulkTemplateEmailCommand = createBulkReminderEmailCommand(
    USERS,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendBulkTemplateEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
```

```
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte SendBulkTemplatedEmail em Referência AWS SDK para JavaScript da API.

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SendEmailCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createSendEmailCommand = (toAddress, fromAddress) => {
  return new SendEmailCommand({
    Destination: {
      /* required */
      CcAddresses: [
        /* more items */
      ],
      ToAddresses: [
        toAddress,
        /* more To-email addresses */
      ],
    },
    Message: {
      /* required */
      Body: {
        /* required */
        Html: {
```

```
          Charset: "UTF-8",
          Data: "HTML_FORMAT_BODY",
        },
        Text: {
          Charset: "UTF-8",
          Data: "TEXT_FORMAT_BODY",
        },
      },
      Subject: {
        Charset: "UTF-8",
        Data: "EMAIL_SUBJECT",
      },
    },
    Source: fromAddress,
    ReplyToAddresses: [
      /* more items */
    ],
  });
};

const run = async () => {
  const sendEmailCommand = createSendEmailCommand(
    "recipient@example.com",
    "sender@example.com",
  );

  try {
    return await sesClient.send(sendEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte SendEmail em Referência AWS SDK para JavaScript da API.

`SendRawEmail`

O código de exemplo a seguir mostra como usar `SendRawEmail`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use nodemailer para enviar um e-mail com anexo.

```
import sesClientModule from "@aws-sdk/client-ses";
/**
 * nodemailer wraps the SES SDK and calls SendRawEmail. Use this for more advanced
 * functionality like adding attachments to your email.
 *
 * https://nodemailer.com/transports/ses
 */
import nodemailer from "nodemailer";

/**
 * @param {string} from An Amazon SES verified email address.
 * @param {*} to An Amazon SES verified email address.
 */
export const sendEmailWithAttachments = (
  from = "from@example.com",
  to = "to@example.com",
) => {
  const ses = new sesClientModule.SESClient({});
  const transporter = nodemailer.createTransport({
    SES: { ses, aws: sesClientModule },
  });

  return new Promise((resolve, reject) => {
    transporter.sendMail(
      {
        from,
        to,
        subject: "Hello World",
        text: "Greetings from Amazon SES!",
```

```
        attachments: [{ content: "Hello World!", filename: "hello.txt" }],
      },
      (err, info) => {
        if (err) {
          reject(err);
        } else {
          resolve(info);
        }
      },
    );
  });
};
```

- Para obter detalhes da API, consulte SendRawEmail em Referência AWS SDK para JavaScript da API.

`SendTemplatedEmail`

O código de exemplo a seguir mostra como usar `SendTemplatedEmail`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SendTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");
```

```
/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL = postfix(getUniqueName("Bilbo"), "@example.com");

const USER = { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL };

/**
 *
 * @param { { emailAddress: string, firstName: string } } user
 * @param { string } templateName - The name of an existing template in Amazon SES.
 * @returns { SendTemplatedEmailCommand }
 */
const createReminderEmailCommand = (user, templateName) => {
  return new SendTemplatedEmailCommand({
    /**
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{contact.firstName}},</h1><p>Don't forget about the
 party gifts!</p>
     * Destination: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!</p>
     */
    Destination: { ToAddresses: [user.emailAddress] },
    TemplateData: JSON.stringify({ contact: { firstName: user.firstName } }),
    Source: VERIFIED_EMAIL,
    Template: templateName,
  });
};

const run = async () => {
  const sendReminderEmailCommand = createReminderEmailCommand(
    USER,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendReminderEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte SendTemplatedEmail em Referência AWS SDK para JavaScript da API.

`UpdateTemplate`

O código de exemplo a seguir mostra como usar `UpdateTemplate`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { UpdateTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");
const HTML_PART = "<h1>Hello, World!</h1>";

const createUpdateTemplateCommand = () => {
  return new UpdateTemplateCommand({
    Template: {
      TemplateName: TEMPLATE_NAME,
      HtmlPart: HTML_PART,
      SubjectPart: "Example",
      TextPart: "Updated template text.",
    },
  });
};

const run = async () => {
  const updateTemplateCommand = createUpdateTemplateCommand();

  try {
    return await sesClient.send(updateTemplateCommand);
  } catch (err) {
```

```
    console.log("Failed to update template.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte UpdateTemplate em Referência AWS SDK para JavaScript da API.

`VerifyDomainIdentity`

O código de exemplo a seguir mostra como usar `VerifyDomainIdentity`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { VerifyDomainIdentityCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * You must have access to the domain's DNS settings to complete the
 * domain verification process.
 */
const DOMAIN_NAME = postfix(getUniqueName("Domain"), ".example.com");

const createVerifyDomainIdentityCommand = () => {
  return new VerifyDomainIdentityCommand({ Domain: DOMAIN_NAME });
};

const run = async () => {
  const VerifyDomainIdentityCommand = createVerifyDomainIdentityCommand();
```

```
  try {
    return await sesClient.send(VerifyDomainIdentityCommand);
  } catch (err) {
    console.log("Failed to verify domain.", err);
    return err;
  }
};
```

- Para obter detalhes da API, consulte VerifyDomainIdentity em Referência AWS SDK para JavaScript da API.

`VerifyEmailIdentity`

O código de exemplo a seguir mostra como usar `VerifyEmailIdentity`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Import required AWS SDK clients and commands for Node.js
import { VerifyEmailIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const EMAIL_ADDRESS = "name@example.com";

const createVerifyEmailIdentityCommand = (emailAddress) => {
  return new VerifyEmailIdentityCommand({ EmailAddress: emailAddress });
};

const run = async () => {
  const verifyEmailIdentityCommand =
    createVerifyEmailIdentityCommand(EMAIL_ADDRESS);
  try {
    return await sesClient.send(verifyEmailIdentityCommand);
  } catch (err) {
    console.log("Failed to verify email identity.", err);
```

```
    return err;
  }
};
```

- Para obter detalhes da API, consulte VerifyEmailIdentity em Referência AWS SDK para JavaScript da API.

Cenários

Criar uma aplicação de transmissão do Amazon Transcribe

O exemplo de código a seguir mostra como construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail.

SDK para JavaScript (v3)

Mostra como usar o Amazon Transcribe para construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript (v3) para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon Aurora e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um Node.js back-end Express.

- Integre um aplicativo React.js web com Serviços da AWS.
- Liste, adicione e atualize itens em uma tabela do Aurora.
- Use o Amazon SES para enviar um relatório por e-mail dos itens de trabalho filtrados.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para JavaScript (v3)

Mostra como usar o Amazon Rekognition com o AWS SDK para JavaScript para criar um aplicativo que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Aprenda como:

- Criar um usuário não autenticado usando o Amazon Cognito.
- Analisar imagens em busca de objetos usando o Amazon Rekognition.

- Verificar um endereço de e-mail para o Amazon SES.
- Enviar uma notificação por e-mail usando o Amazon SES.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Exemplos do Amazon SNS usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon SNS

O exemplo de código a seguir mostra como começar a usar o Amazon SNS.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicialize um cliente SNS e liste tópicos em sua conta.

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});

  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
    }
  }

  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para JavaScript da API.

Ações

`CheckIfPhoneNumberIsOptedOut`

O código de exemplo a seguir mostra como usar `CheckIfPhoneNumberIsOptedOut`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
```

```
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOut em Referência AWS SDK para JavaScript da API.

`ConfirmSubscription`

O código de exemplo a seguir mostra como usar `ConfirmSubscription`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
```

```
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm a
 subscription.
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
      // If this is true, the subscriber cannot unsubscribe while unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ConfirmSubscription em Referência AWS SDK para JavaScript da API.

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateTopic em Referência AWS SDK para JavaScript da API.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
```

```
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTopic em Referência AWS SDK para JavaScript da API.

`GetSMSAttributes`

O código de exemplo a seguir mostra como usar `GetSMSAttributes`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
```

```
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
    // the DefaultSMSType is undefined. For this example, it was set to
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em GetSMSAttributes na Referência da API AWS SDK para JavaScript .

`GetTopicAttributes`

O código de exemplo a seguir mostra como usar `GetTopicAttributes`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { GetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDelayRe
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetTopicAttributes em Referência AWS SDK para JavaScript da API.

`ListSubscriptions`

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
```

```
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.

- Para obter detalhes da API, consulte ListSubscriptions em Referência AWS SDK para JavaScript da API.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { ListTopicsCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
```

```
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para JavaScript da API.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
```

```
 *                                                    if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Publique uma mensagem em um tópico com opções de grupo, duplicação e atributo.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
```

```
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );
```

```
    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Publish na Referência da API AWS SDK para JavaScript .

`SetSMSAttributes`

O código de exemplo a seguir mostra como usar `SetSMSAttributes`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";
```

```
/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional - (Default) Noncritical messages, such as marketing messages.
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para receber detalhes da API, consulte SetSMSAttributes na Referência da API AWS SDK para JavaScript .

`SetTopicAttributes`

O código de exemplo a seguir mostra como usar `SetTopicAttributes`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
```

```
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte SetTopicAttributes em Referência AWS SDK para JavaScript da API.

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";
```

```
/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm a
 subscription.
 * @param {string} emailAddress - The email address that is subscribed to the topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

Inscrever uma aplicação móvel em um tópico.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint is
 created
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
```

```
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Inscrever uma função do Lambda em um tópico.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
```

```
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Assinar uma fila do SQS em um tópico.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
```

```
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Assinar com um filtro em um tópico.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute set
 to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para JavaScript .

Cancelar assinatura

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cliente em um módulo separado e exporte-o.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importe o SDK e os módulos do cliente e chame a API.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
```

```
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-xxxx-
xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Consulte detalhes da API em Unsubscribe na Referência da API do AWS SDK para JavaScript .

Cenários

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

SDK para JavaScript (v3)

Este exemplo mostra como criar uma aplicação que permite que os usuários enviem dados para uma tabela do Amazon DynamoDB e enviem uma mensagem de texto ao administrador usando o Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o ponto de entrada para esse cenário.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

O código anterior fornece as dependências necessárias e inicia o cenário. A próxima seção contém a maior parte do exemplo.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
```

```
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
    this.sqsClient = sqsClient;
    this.prompter = prompter;
    this.logger = logger;
  }

  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
      await this.logger.log(MESSAGES.deduplicationNotice);
      await this.logger.log(MESSAGES.deduplicationDescription);
      this.autoDedup = await this.prompter.confirm({
        message: MESSAGES.deduplicationPrompt,
      });
    }
  }

  async createTopic() {
    await this.logger.log(MESSAGES.creatingTopics);
    this.topicName = await this.prompter.input({
```

```
      message: MESSAGES.topicNamePrompt,
    });
    if (this.isFifo) {
      this.topicName += ".fifo";
      this.logger.logSeparator(MESSAGES.headerFifoNaming);
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
```

```
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
```

```
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }
```

```
  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
```

```
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
    );
  }
}

async publishMessages() {
  const message = await this.prompter.input({
    message: MESSAGES.publishMessagePrompt,
  });

  let groupId;
  let deduplicationId;
  let choices;

  if (this.isFifo) {
    await this.logger.log(MESSAGES.groupIdNotice);
    groupId = await this.prompter.input({
      message: MESSAGES.groupIdPrompt,
    });

    if (this.autoDedup === false) {
      await this.logger.log(MESSAGES.deduplicationIdNotice);
      deduplicationId = await this.prompter.input({
        message: MESSAGES.deduplicationIdPrompt,
      });
    }

    choices = await this.prompter.checkbox({
      message: MESSAGES.messageAttributesPrompt,
      choices: toneChoices,
    });
  }

  await this.snsClient.send(
    new PublishCommand({
      TopicArn: this.topicArn,
      Message: message,
      ...(groupId
        ? {
            MessageGroupId: groupId,
          }
        : {}),
      ...(deduplicationId
```

```
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
```

```
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
```

```
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes

- Publicar
- ReceiveMessage
- SetQueueAttributes
- Assinar
- Cancelar assinatura

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para JavaScript (v3)

Mostra como criar uma AWS Lambda função usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para JavaScript (v3)

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de tempo de JavaScript execução do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Esse exemplo também está disponível no Guia do desenvolvedor do AWS SDK para JavaScript v3.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento SNS com o Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Consumindo um evento SNS com o Lambda usando. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
```

```
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Exemplos do Amazon SQS usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon SQS

O exemplo de código a seguir mostra como começar a usar o Amazon SQS.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicializar um cliente Amazon SQS e listar as filas.

```
import { SQSClient, paginateListQueues } from "@aws-sdk/client-sqs";

export const helloSqs = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SQSClient({});

  // You can also use `ListQueuesCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListQueuesCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedQueues = paginateListQueues({ client }, {});
  const queues = [];

  for await (const page of paginatedQueues) {
    if (page.QueueUrls?.length) {
      queues.push(...page.QueueUrls);
    }
  }

  const suffix = queues.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SQS! You have ${queues.length} queue${suffix} in your account.`,
  );
  console.log(queues.map((t) => `  * ${t}`).join("\n"));
};
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para JavaScript da API.

Ações

`ChangeMessageVisibility`

O código de exemplo a seguir mostra como usar `ChangeMessageVisibility`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem do Amazon SQS e altere sua visibilidade de tempo limite.

```
import {
  ReceiveMessageCommand,
  ChangeMessageVisibilityCommand,
  SQSClient,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 1,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 1,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  const response = await client.send(
    new ChangeMessageVisibilityCommand({
      QueueUrl: queueUrl,
```

```
      ReceiptHandle: Messages[0].ReceiptHandle,
      VisibilityTimeout: 20,
    }),
  );
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte ChangeMessageVisibility em Referência AWS SDK para JavaScript da API.

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma fila padrão do Amazon SQS.

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (sqsQueueName = SQS_QUEUE_NAME) => {
  const command = new CreateQueueCommand({
    QueueName: sqsQueueName,
    Attributes: {
      DelaySeconds: "60",
      MessageRetentionPeriod: "86400",
    },
  });

  const response = await client.send(command);
```

```
  console.log(response);
  return response;
};
```

Crie uma fila do Amazon SQS com sondagem longa.

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "queue_name";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const response = await client.send(
    new CreateQueueCommand({
      QueueName: queueName,
      Attributes: {
        // When the wait time for the ReceiveMessage API action is greater than 0,
        // long polling is in effect. The maximum long polling wait time is 20
        // seconds. Long polling helps reduce the cost of using Amazon SQS by,
        // eliminating the number of empty responses and false empty responses.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        ReceiveMessageWaitTimeSeconds: "20",
      },
    }),
  );
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte CreateQueue em Referência AWS SDK para JavaScript da API.

`DeleteMessage`

O código de exemplo a seguir mostra como usar `DeleteMessage`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receber e excluir mensagens do Amazon SQS.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
```

```
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- Para obter detalhes da API, consulte DeleteMessage em Referência AWS SDK para JavaScript da API.

`DeleteMessageBatch`

O código de exemplo a seguir mostra como usar `DeleteMessageBatch`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
```

```
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- Para obter detalhes da API, consulte DeleteMessageBatch em Referência AWS SDK para JavaScript da API.

DeleteQueue

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma fila do Amazon SQS.

```
import { DeleteQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "test-queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new DeleteQueueCommand({ QueueUrl: queueUrl });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteQueue em Referência AWS SDK para JavaScript da API.

GetQueueAttributes

O código de exemplo a seguir mostra como usar `GetQueueAttributes`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { GetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const getQueueAttributes = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new GetQueueAttributesCommand({
    QueueUrl: queueUrl,
    AttributeNames: ["DelaySeconds"],
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '747a1192-c334-5682-a508-4cd5e8dc4e79',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: { DelaySeconds: '1' }
  // }
  return response;
};
```

- Para obter detalhes da API, consulte GetQueueAttributes em Referência AWS SDK para JavaScript da API.

`GetQueueUrl`

O código de exemplo a seguir mostra como usar `GetQueueUrl`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha o URL para uma fila do Amazon SQS.

```
import { GetQueueUrlCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const command = new GetQueueUrlCommand({ QueueName: queueName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte GetQueueUrl em Referência AWS SDK para JavaScript da API.

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar filas do Amazon SQS.

```
import { paginateListQueues, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});

export const main = async () => {
  const paginatedListQueues = paginateListQueues({ client }, {});

  /** @type {string[]} */
  const urls = [];
  for await (const page of paginatedListQueues) {
    const nextUrls = page.QueueUrls?.filter((qurl) => !!qurl) || [];
    urls.push(...nextUrls);
    for (const url of urls) {
      console.log(url);
    }
  }

  return urls;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para JavaScript da API.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem de uma fila do Amazon SQS.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
```

```
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
import { ReceiveMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new ReceiveMessageCommand({
    AttributeNames: ["SentTimestamp"],
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ["All"],
    QueueUrl: queueUrl,
    // The duration (in seconds) for which the call waits for a message
    // to arrive in the queue before returning. If a message is available,
    // the call returns sooner than WaitTimeSeconds. If no messages are
    // available and the wait time expires, the call returns successfully
    // with an empty list of messages.
    // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/
API_ReceiveMessage.html#API_ReceiveMessage_RequestSyntax
    WaitTimeSeconds: 20,
  });

  const response = await client.send(command);
  console.log(response);
```

```
  return response;
};
```

- Para obter detalhes da API, consulte ReceiveMessage em Referência AWS SDK para JavaScript da API.

SendMessage

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Enviar uma mensagem para uma fila do Amazon SQS.

```
import { SendMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

export const main = async (sqsQueueUrl = SQS_QUEUE_URL) => {
  const command = new SendMessageCommand({
    QueueUrl: sqsQueueUrl,
    DelaySeconds: 10,
    MessageAttributes: {
      Title: {
        DataType: "String",
        StringValue: "The Whistler",
      },
      Author: {
        DataType: "String",
        StringValue: "John Grisham",
      },
      WeeksOn: {
        DataType: "Number",
```

```
        StringValue: "6",
      },
    },
    MessageBody:
      "Information about current NY Times fiction bestseller for week of
 12/11/2016.",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte SendMessage em Referência AWS SDK para JavaScript da API.

`SetQueueAttributes`

O código de exemplo a seguir mostra como usar `SetQueueAttributes`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    QueueUrl: queueUrl,
    Attributes: {
      DelaySeconds: "1",
    },
```

```
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Configurar uma fila do Amazon SQS para usar sondagem longa.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    Attributes: {
      ReceiveMessageWaitTimeSeconds: "20",
    },
    QueueUrl: queueUrl,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Configurar uma dead-letter queue.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";
const DEAD_LETTER_QUEUE_ARN = "dead_letter_queue_arn";

export const main = async (
  queueUrl = SQS_QUEUE_URL,
  deadLetterQueueArn = DEAD_LETTER_QUEUE_ARN,
) => {
  const command = new SetQueueAttributesCommand({
    Attributes: {
```

```
      RedrivePolicy: JSON.stringify({
        // Amazon SQS supports dead-letter queues (DLQ), which other
        // queues (source queues) can target for messages that can't
        // be processed (consumed) successfully.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-dead-letter-queues.html
        deadLetterTargetArn: deadLetterQueueArn,
        maxReceiveCount: "10",
      }),
    },
    QueueUrl: queueUrl,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Para obter detalhes da API, consulte SetQueueAttributes em Referência AWS SDK para JavaScript da API.

Cenários

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse é o ponto de entrada para esse cenário.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;
```

```
  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

O código anterior fornece as dependências necessárias e inicia o cenário. A próxima seção contém a maior parte do exemplo.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
```

```
 */
constructor(snsClient, sqsClient, prompter, logger) {
  this.snsClient = snsClient;
  this.sqsClient = sqsClient;
  this.prompter = prompter;
  this.logger = logger;
}

async welcome() {
  await this.logger.log(MESSAGES.description);
}

async confirmFifo() {
  await this.logger.log(MESSAGES.snsFifoDescription);
  this.isFifo = await this.prompter.confirm({
    message: MESSAGES.snsFifoPrompt,
  });

  if (this.isFifo) {
    this.logger.logSeparator(MESSAGES.headerDedup);
    await this.logger.log(MESSAGES.deduplicationNotice);
    await this.logger.log(MESSAGES.deduplicationDescription);
    this.autoDedup = await this.prompter.confirm({
      message: MESSAGES.deduplicationPrompt,
    });
  }
}

async createTopic() {
  await this.logger.log(MESSAGES.creatingTopics);
  this.topicName = await this.prompter.input({
    message: MESSAGES.topicNamePrompt,
  });
  if (this.isFifo) {
    this.topicName += ".fifo";
    this.logger.logSeparator(MESSAGES.headerFifoNaming);
    await this.logger.log(MESSAGES.appendFifoNotice);
  }

  const response = await this.snsClient.send(
    new CreateTopicCommand({
      Name: this.topicName,
      Attributes: {
        FifoTopic: this.isFifo ? "true" : "false",
```

```
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
```

```
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
```

```
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];
```

```
      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
```

```
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
```

```
        : {}),
    }),
  );

  const publishAnother = await this.prompter.confirm({
    message: MESSAGES.publishAnother,
  });

  if (publishAnother) {
    await this.publishMessages();
  }
}

async receiveAndDeleteMessages() {
  for (const queue of this.queues) {
    const { Messages } = await this.sqsClient.send(
      new ReceiveMessageCommand({
        QueueUrl: queue.queueUrl,
      }),
    );

    if (Messages) {
      await this.logger.log(
        MESSAGES.messagesReceivedNotice.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      );
      console.log(Messages);

      await this.sqsClient.send(
        new DeleteMessageBatchCommand({
          QueueUrl: queue.queueUrl,
          Entries: Messages.map((message) => ({
            Id: message.MessageId,
            ReceiptHandle: message.ReceiptHandle,
          })),
        }),
      );
    } else {
      await this.logger.log(
        MESSAGES.noMessagesReceivedNotice.replace(
          "${QUEUE_NAME}",
          queue.queueName,
```

```
        ),
      );
    }
  }

  const deleteAndPoll = await this.prompter.confirm({
    message: MESSAGES.deleteAndPollConfirmation,
  });

  if (deleteAndPoll) {
    await this.receiveAndDeleteMessages();
  }
}

async destroyResources() {
  for (const subscriptionArn of this.subscriptionArns) {
    await this.snsClient.send(
      new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
    );
  }

  for (const queue of this.queues) {
    await this.sqsClient.send(
      new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
    );
  }

  if (this.topicArn) {
    await this.snsClient.send(
      new DeleteTopicCommand({ TopicArn: this.topicArn }),
    );
  }
}

async start() {
  console.clear();

  try {
    this.logger.logSeparator(MESSAGES.headerWelcome);
    await this.welcome();
    this.logger.logSeparator(MESSAGES.headerFifo);
    await this.confirmFifo();
    this.logger.logSeparator(MESSAGES.headerCreateTopic);
    await this.createTopic();
```

```
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumindo um evento SQS com o Lambda usando. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message) {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Consumindo um evento SQS com o Lambda usando. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";

export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatando falhas de itens de lote do SQS com o uso do Lambda. JavaScript

```
// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
    console.log(`Processed message: ${record.body}`);
}
```

Relatando falhas de itens de lote do SQS com o uso do Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord } from
 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
 Promise<SQSBatchResponse> => {
    const batchItemFailures: SQSBatchItemFailure[] = [];

    for (const record of event.Records) {
        try {
            await processMessageAsync(record);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }

    return {batchItemFailures: batchItemFailures};
};

async function processMessageAsync(record: SQSRecord): Promise<void> {
```

```
    if (record.body && record.body.includes("error")) {
        throw new Error('There is an error in the SQS Message.');
    }
    console.log(`Processed message ${record.body}`);
}
```

Exemplos do Step Functions usando o SDK for JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com Step Functions.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`StartExecution`

O código de exemplo a seguir mostra como usar `StartExecution`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SFNClient, StartExecutionCommand } from "@aws-sdk/client-sfn";
```

```
/**
 * @param {{ sfnClient: SFNClient, stateMachineArn: string }} config
 */
export async function startExecution({ sfnClient, stateMachineArn }) {
  const response = await sfnClient.send(
    new StartExecutionCommand({
      stateMachineArn,
    }),
  );
  console.log(response);
  // Example response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '202a9309-c16a-454b-adeb-c4d19afe3bf2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   executionArn: 'arn:aws:states:us-
east-1:000000000000:execution:MyStateMachine:aaaaaaaa-f787-49fb-a20c-1b61c64eafe6',
  //   startDate: 2024-01-04T15:54:08.362Z
  // }
  return response;
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  startExecution({ sfnClient: new SFNClient({}), stateMachineArn: "ARN" });
}
```

- Para obter detalhes da API, consulte StartExecution em Referência AWS SDK para JavaScript da API.

AWS STS exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

Assuma um perfil do IAM.

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
```

```
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte AssumeRole em Referência AWS SDK para JavaScript da API.

Suporte exemplos de uso do SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com Suporte.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Conceitos básicos
- Ações

Conceitos básicos

Hello Suporte

O exemplo de código a seguir mostra como começar a usar o Suporte.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Invoque `main()` para executar o exemplo.

```
import {
  DescribeServicesCommand,
  SupportClient,
} from "@aws-sdk/client-support";

// Change the value of 'region' to your preferred AWS Region.
const client = new SupportClient({ region: "us-east-1" });

const getServiceCount = async () => {
  try {
    const { services } = await client.send(new DescribeServicesCommand({}));
    return services.length;
  } catch (err) {
    if (err.name === "SubscriptionRequiredException") {
      throw new Error(
        "You must be subscribed to the AWS Support plan to use this feature.",
      );
    }
    throw err;
  }
};

export const main = async () => {
```

```
  try {
    const count = await getServiceCount();
    console.log(`Hello, AWS Support! There are ${count} services available.`);
  } catch (err) {
    console.error("Failed to get service count: ", err.message);
  }
};
```

- Para obter detalhes da API, consulte DescribeServices em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Obter e exibir os serviços disponíveis e os níveis de gravidade dos casos.
- Criar um caso de suporte usando um serviço, uma categoria e um nível de gravidade selecionados.
- Obter e exibir uma lista de casos em aberto para o dia atual.
- Adicionar um conjunto de anexos e uma comunicação ao novo caso.
- Descrever o novo anexo e a comunicação para o caso.
- Resolver o caso.
- Obtenha e exiba uma lista de casos resolvidos para o dia atual.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo no terminal.

```
import {
```

```
  AddAttachmentsToSetCommand,
  AddCommunicationToCaseCommand,
  CreateCaseCommand,
  DescribeAttachmentCommand,
  DescribeCasesCommand,
  DescribeCommunicationsCommand,
  DescribeServicesCommand,
  DescribeSeverityLevelsCommand,
  ResolveCaseCommand,
  SupportClient,
} from "@aws-sdk/client-support";
import * as inquirer from "@inquirer/prompts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};

const client = new SupportClient({ region: "us-east-1" });

// Verify that the account has a Support plan.
export const verifyAccount = async () => {
  const command = new DescribeServicesCommand({});

  try {
    await client.send(command);
  } catch (err) {
    if (err.name === "SubscriptionRequiredException") {
      throw new Error(
        "You must be subscribed to the AWS Support plan to use this feature.",
      );
    }
    throw err;
  }
};

/**
 * Select a service from the list returned from DescribeServices.
 */
export const getService = async () => {
  const { services } = await client.send(new DescribeServicesCommand({}));
  const selectedService = await inquirer.select({
    message:
```

```
      "Select a service. Your support case will be created for this service. The
 list of services is truncated for readability.",
    choices: services.slice(0, 10).map((s) => ({ name: s.name, value: s })),
  });
  return selectedService;
};

/**
 * @param {{ categories: import('@aws-sdk/client-support').Category[]}} service
 */
export const getCategory = async (service) => {
  const selectedCategory = await inquirer.select({
    message: "Select a category.",
    choices: service.categories.map((c) => ({ name: c.name, value: c })),
  });
  return selectedCategory;
};

// Get the available severity levels for the account.
export const getSeverityLevel = async () => {
  const command = new DescribeSeverityLevelsCommand({});
  const { severityLevels } = await client.send(command);
  const selectedSeverityLevel = await inquirer.select({
    message: "Select a severity level.",
    choices: severityLevels.map((s) => ({ name: s.name, value: s })),
  });
  return selectedSeverityLevel;
};

/**
 * Create a new support case
 * @param {{
 *  selectedService: import('@aws-sdk/client-support').Service
 *  selectedCategory: import('@aws-sdk/client-support').Category
 *  selectedSeverityLevel: import('@aws-sdk/client-support').SeverityLevel
 * }} selections
 * @returns
 */
export const createCase = async ({
  selectedService,
  selectedCategory,
  selectedSeverityLevel,
}) => {
  const command = new CreateCaseCommand({
```

```
    subject: "IGNORE: Test case",
    communicationBody: "This is a test. Please ignore.",
    serviceCode: selectedService.code,
    categoryCode: selectedCategory.code,
    severityCode: selectedSeverityLevel.code,
  });
  const { caseId } = await client.send(command);
  return caseId;
};

// Get a list of open support cases created today.
export const getTodaysOpenCases = async () => {
  const d = new Date();
  const startOfToday = new Date(d.getFullYear(), d.getMonth(), d.getDate());
  const command = new DescribeCasesCommand({
    includeCommunications: false,
    afterTime: startOfToday.toISOString(),
  });

  const { cases } = await client.send(command);

  if (cases.length === 0) {
    throw new Error(
      "Unexpected number of cases. Expected more than 0 open cases.",
    );
  }
  return cases;
};

// Create an attachment set.
export const createAttachmentSet = async () => {
  const command = new AddAttachmentsToSetCommand({
    attachments: [
      {
        fileName: "example.txt",
        data: new TextEncoder().encode("some example text"),
      },
    ],
  });
  const { attachmentSetId } = await client.send(command);
  return attachmentSetId;
};

export const linkAttachmentSetToCase = async (attachmentSetId, caseId) => {
```

```
  const command = new AddCommunicationToCaseCommand({
    attachmentSetId,
    caseId,
    communicationBody: "Adding attachment set to case.",
  });
  await client.send(command);
};

// Get all communications for a support case.
export const getCommunications = async (caseId) => {
  const command = new DescribeCommunicationsCommand({
    caseId,
  });
  const { communications } = await client.send(command);
  return communications;
};

/**
 * @param {import('@aws-sdk/client-support').Communication[]} communications
 */
export const getFirstAttachment = (communications) => {
  const firstCommWithAttachment = communications.find(
    (c) => c.attachmentSet.length > 0,
  );
  return firstCommWithAttachment?.attachmentSet[0].attachmentId;
};

// Get an attachment.
export const getAttachment = async (attachmentId) => {
  const command = new DescribeAttachmentCommand({
    attachmentId,
  });
  const { attachment } = await client.send(command);
  return attachment;
};

// Resolve the case matching the given case ID.
export const resolveCase = async (caseId) => {
  const shouldResolve = await inquirer.confirm({
    message: `Do you want to resolve ${caseId}?`,
  });

  if (shouldResolve) {
    const command = new ResolveCaseCommand({
```

```
      caseId: caseId,
    });

    await client.send(command);
    return true;
  }
  return false;
};

/**
 * Find a specific case in the list of provided cases by case ID.
 * If the case is not found, and the results are paginated, continue
 * paging through the results.
 * @param {{
 *   caseId: string,
 *   cases: import('@aws-sdk/client-support').CaseDetails[]
 *   nextToken: string
 * }} options
 * @returns
 */
export const findCase = async ({ caseId, cases, nextToken }) => {
  const foundCase = cases.find((c) => c.caseId === caseId);

  if (foundCase) {
    return foundCase;
  }

  if (nextToken) {
    const response = await client.send(
      new DescribeCasesCommand({
        nextToken,
        includeResolvedCases: true,
      }),
    );
    return findCase({
      caseId,
      cases: response.cases,
      nextToken: response.nextToken,
    });
  }

  throw new Error(`${caseId} not found.`);
};
```

```
// Get all cases created today.
export const getTodaysResolvedCases = async (caseIdToWaitFor) => {
  const d = new Date("2023-01-18");
  const startOfToday = new Date(d.getFullYear(), d.getMonth(), d.getDate());
  const command = new DescribeCasesCommand({
    includeCommunications: false,
    afterTime: startOfToday.toISOString(),
    includeResolvedCases: true,
  });
  const { cases, nextToken } = await client.send(command);
  await findCase({ cases, caseId: caseIdToWaitFor, nextToken });
  return cases.filter((c) => c.status === "resolved");
};

const main = async () => {
  let caseId;
  try {
    console.log(wrapText("Welcome to the AWS Support basic usage scenario."));

    // Verify that the account is subscribed to support.
    await verifyAccount();

    // Provided a truncated list of services and prompt the user to select one.
    const selectedService = await getService();

    // Provided the categories for the selected service and prompt the user to
 select one.
    const selectedCategory = await getCategory(selectedService);

    // Provide the severity available severity levels for the account and prompt the
 user to select one.
    const selectedSeverityLevel = await getSeverityLevel();

    // Create a support case.
    console.log("\nCreating a support case.");
    caseId = await createCase({
      selectedService,
      selectedCategory,
      selectedSeverityLevel,
    });
    console.log(`Support case created: ${caseId}`);

    // Display a list of open support cases created today.
    const todaysOpenCases = await retry(
```

```
      { intervalInMs: 1000, maxRetries: 15 },
      getTodaysOpenCases,
    );
    console.log(
      `\nOpen support cases created today: ${todaysOpenCases.length}`,
    );
    console.log(todaysOpenCases.map((c) => `${c.caseId}`).join("\n"));

    // Create an attachment set.
    console.log("\nCreating an attachment set.");
    const attachmentSetId = await createAttachmentSet();
    console.log(`Attachment set created: ${attachmentSetId}`);

    // Add the attachment set to the support case.
    console.log(`\nAdding attachment set to ${caseId}`);
    await linkAttachmentSetToCase(attachmentSetId, caseId);
    console.log(`Attachment set added to ${caseId}`);

    // List the communications for a support case.
    console.log(`\nListing communications for ${caseId}`);
    const communications = await getCommunications(caseId);
    console.log(
      communications
        .map(
          (c) =>
            `Communication created on ${c.timeCreated}. Has
${c.attachmentSet.length} attachments.`,
        )
        .join("\n"),
    );

    // Describe the first attachment.
    console.log(`\nDescribing attachment ${attachmentSetId}`);
    const attachmentId = getFirstAttachment(communications);
    const attachment = await getAttachment(attachmentId);
    console.log(
      `Attachment is the file '${
        attachment.fileName
      }' with data: \n${new TextDecoder().decode(attachment.data)}`,
    );

    // Confirm that the support case should be resolved.
    const isResolved = await resolveCase(caseId);
    if (isResolved) {
```

```
      // List the resolved cases and include the one previously created.
      // Resolved cases can take a while to appear.
      console.log(
        "\nWaiting for case status to be marked as resolved. This can take some
 time.",
      );
      const resolvedCases = await retry(
        { intervalInMs: 20000, maxRetries: 15 },
        () => getTodaysResolvedCases(caseId),
      );
      console.log("Resolved cases:");
      console.log(resolvedCases.map((c) => c.caseId).join("\n"));
    }
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Ações

`AddAttachmentsToSet`

O código de exemplo a seguir mostra como usar `AddAttachmentsToSet`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { AddAttachmentsToSetCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a new attachment set or add attachments to an existing set.
    // Provide an 'attachmentSetId' value to add attachments to an existing set.
    // Use AddCommunicationToCase or CreateCase to associate an attachment set with
 a support case.
    const response = await client.send(
      new AddAttachmentsToSetCommand({
        // You can add up to three attachments per set. The size limit is 5 MB per
 attachment.
        attachments: [
          {
            fileName: "example.txt",
            data: new TextEncoder().encode("some example text"),
          },
        ],
      }),
    );
    // Use this ID in AddCommunicationToCase or CreateCase.
    console.log(response.attachmentSetId);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte AddAttachmentsToSet em Referência AWS SDK para JavaScript da API.

`AddCommunicationToCase`

O código de exemplo a seguir mostra como usar `AddCommunicationToCase`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { AddCommunicationToCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  let attachmentSetId;

  try {
    // Add a communication to a case.
    const response = await client.send(
      new AddCommunicationToCaseCommand({
        communicationBody: "Adding an attachment.",
        // Set value to an existing support case id.
        caseId: "CASE_ID",
        // Optional. Set value to an existing attachment set id to add attachments
 to the case.
        attachmentSetId,
      }),
    );
    console.log(response);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte AddCommunicationToCase em Referência AWS SDK para JavaScript da API.

`CreateCase`

O código de exemplo a seguir mostra como usar `CreateCase`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a new case and log the case id.
    // Important: This creates a real support case in your account.
    const response = await client.send(
      new CreateCaseCommand({
        // The subject line of the case.
        subject: "IGNORE: Test case",
        // Use DescribeServices to find available service codes for each service.
        serviceCode: "service-quicksight-end-user",
        // Use DescribeSecurityLevels to find available severity codes for your
 support plan.
        severityCode: "low",
        // Use DescribeServices to find available category codes for each service.
        categoryCode: "end-user-support",
        // The main description of the support case.
        communicationBody: "This is a test. Please ignore.",
      }),
    );
    console.log(response.caseId);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte CreateCase em Referência AWS SDK para JavaScript da API.

`DescribeAttachment`

O código de exemplo a seguir mostra como usar `DescribeAttachment`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeAttachmentCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get the metadata and content of an attachment.
    const response = await client.send(
      new DescribeAttachmentCommand({
        // Set value to an existing attachment id.
        // Use DescribeCommunications or DescribeCases to find an attachment id.
        attachmentId: "ATTACHMENT_ID",
      }),
    );
    console.log(response.attachment?.fileName);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte DescribeAttachment em Referência AWS SDK para JavaScript da API.

`DescribeCases`

O código de exemplo a seguir mostra como usar `DescribeCases`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeCasesCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get all of the unresolved cases in your account.
    // Filter or expand results by providing parameters to the DescribeCasesCommand.
 Refer
    // to the TypeScript definition and the API doc for more information on possible
 parameters.
    // https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
support/interfaces/describecasescommandinput.html
    const response = await client.send(new DescribeCasesCommand({}));
    const caseIds = response.cases.map((supportCase) => supportCase.caseId);
    console.log(caseIds);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte DescribeCases em Referência AWS SDK para JavaScript da API.

`DescribeCommunications`

O código de exemplo a seguir mostra como usar `DescribeCommunications`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeCommunicationsCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get all communications for the support case.
    // Filter results by providing parameters to the DescribeCommunicationsCommand.
 Refer
    // to the TypeScript definition and the API doc for more information on possible
 parameters.
    // https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
support/interfaces/describecommunicationscommandinput.html
    const response = await client.send(
      new DescribeCommunicationsCommand({
        // Set value to an existing case id.
        caseId: "CASE_ID",
      }),
    );
    const text = response.communications.map((item) => item.body).join("\n");
    console.log(text);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte DescribeCommunications em Referência AWS SDK para JavaScript da API.

`DescribeSeverityLevels`

O código de exemplo a seguir mostra como usar `DescribeSeverityLevels`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DescribeSeverityLevelsCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get the list of severity levels.
    // The available values depend on the support plan for the account.
    const response = await client.send(new DescribeSeverityLevelsCommand({}));
    console.log(response.severityLevels);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte DescribeSeverityLevels em Referência AWS SDK para JavaScript da API.

`ResolveCase`

O código de exemplo a seguir mostra como usar `ResolveCase`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { ResolveCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

const main = async () => {
  try {
    const response = await client.send(
      new ResolveCaseCommand({
        caseId: "CASE_ID",
      }),
    );

    console.log(response.finalCaseStatus);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Para obter detalhes da API, consulte ResolveCase em Referência AWS SDK para JavaScript da API.

Exemplos do Systems Manager usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Systems Manager.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Systems Manager

O exemplo de código a seguir mostra como começar a usar o Systems Manager.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { paginateListDocuments, SSMClient } from "@aws-sdk/client-ssm";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new SSMClient();
  const listDocumentsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListDocuments({ client }, { MaxResults: 5 });
    for await (const page of paginator) {
```

```
      listDocumentsPaginated.push(...page.DocumentIdentifiers);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }

  for (const { Name, DocumentFormat, CreatedDate } of listDocumentsPaginated) {
    console.log(`${Name} - ${DocumentFormat} - ${CreatedDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Para obter detalhes da API, consulte ListDocuments em Referência AWS SDK para JavaScript da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma janela de manutenção.
- Modificar a programação da janela de manutenção.
- Criar um documento.
- Enviar um comando para uma instância do EC2 especificada.
- Crie um OpsItem.
- Atualize e resolva OpsItem o.
- Exclua a janela de manutenção OpsItem e o documento.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { fileURLToPath } from "node:url";
import {
  CreateDocumentCommand,
  CreateMaintenanceWindowCommand,
  CreateOpsItemCommand,
  DeleteDocumentCommand,
  DeleteMaintenanceWindowCommand,
  DeleteOpsItemCommand,
  DescribeOpsItemsCommand,
  DocumentAlreadyExists,
  OpsItemStatus,
  waitUntilCommandExecuted,
  CancelCommandCommand,
  paginateListCommandInvocations,
  SendCommandCommand,
  UpdateMaintenanceWindowCommand,
  UpdateOpsItemCommand,
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * @typedef {{
 *   ssmClient: import('@aws-sdk/client-ssm').SSMClient,
 *   documentName?: string
 *   maintenanceWindow?: string
 *   winId?: int
 *   ec2InstanceId?: string
```

```
 *   requestedDateTime?: Date
 *   opsItemId?: string
 *   askToDeleteResources?: boolean
 * }} State
 */

const defaultMaintenanceWindow = "ssm-maintenance-window";
const defaultDocumentName = "ssmdocument";
// The timeout duration is highly dependent on the specific setup and environment
 necessary. This example handles only the most common error cases, and uses a much
 shorter duration than most productions systems would use.
const COMMAND_TIMEOUT_DURATION_SECONDS = 30; // 30 seconds

const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `Welcome to the AWS Systems Manager SDK Getting Started scenario.
    This program demonstrates how to interact with Systems Manager using the AWS SDK
 for JavaScript V3.
    Systems Manager is the operations hub for your AWS applications and resources
 and a secure end-to-end management solution.
    The program's primary functions include creating a maintenance window, creating
 a document, sending a command to a document,
    listing documents, listing commands, creating an OpsItem, modifying an OpsItem,
 and deleting Systems Manager resources.
    Upon completion of the program, all AWS resources are cleaned up.
    Let's get started...`,
  { header: true },
);

const createMaintenanceWindow = new ScenarioOutput(
  "createMaintenanceWindow",
  "Step 1: Create a Systems Manager maintenance window.",
);

const getMaintenanceWindow = new ScenarioInput(
  "maintenanceWindow",
  "Please enter the maintenance window name:",
  { type: "input", default: defaultMaintenanceWindow },
);
```

```
export const sdkCreateMaintenanceWindow = new ScenarioAction(
  "sdkCreateMaintenanceWindow",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          Schedule: "cron(0 10 ? * MON-FRI *)", //The schedule of the maintenance
 window in the form of a cron or rate expression.
          Duration: 2, //The duration of the maintenance window in hours.
          Cutoff: 1, //The number of hours before the end of the maintenance window
 that Amazon Web Services Systems Manager stops scheduling new tasks for execution.
          AllowUnassociatedTargets: true, //Allow the maintenance window to run on
 managed nodes, even if you haven't registered those nodes as targets.
        }),
      );
      state.winId = response.WindowId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const modifyMaintenanceWindow = new ScenarioOutput(
  "modifyMaintenanceWindow",
  "Modify the maintenance window by changing the schedule.",
);

const sdkModifyMaintenanceWindow = new ScenarioAction(
  "sdkModifyMaintenanceWindow",
  async (/** @type {State} */ state) => {
    try {
      await state.ssmClient.send(
        new UpdateMaintenanceWindowCommand({
          WindowId: state.winId,
          Schedule: "cron(0 0 ? * MON *)",
        }),
      );
    } catch (caught) {
```

```
      console.error(caught.message);
      console.log(
        `An error occurred while modifying the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const createSystemsManagerActions = new ScenarioOutput(
  "createSystemsManagerActions",
  "Create a document that defines the actions that Systems Manager performs on your
 EC2 instance.",
);

const getDocumentName = new ScenarioInput(
  "documentName",
  "Please enter the document: ",
  { type: "input", default: defaultDocumentName },
);

const sdkCreateSSMDoc = new ScenarioAction(
  "sdkCreateSSMDoc",
  async (/** @type {State} */ state) => {
    const contentData = `{
                "schemaVersion": "2.2",
                "description": "Run a simple shell command",
                "mainSteps": [
                    {
                        "action": "aws:runShellScript",
                        "name": "runEchoCommand",
                        "inputs": {
                          "runCommand": [
                            "echo 'Hello, world!'"
                          ]
                        }
                    }
                ]
            }`;
    try {
      await state.ssmClient.send(
        new CreateDocumentCommand({
          Content: contentData,
```

```
          Name: state.documentName,
          DocumentType: "Command",
        }),
      );
    } catch (caught) {
      console.log(`Exception type: (${typeof caught})`);
      if (caught instanceof DocumentAlreadyExists) {
        console.log("Document already exists. Continuing...\n");
      } else {
        console.error(caught.message);
        console.log(
          `An error occurred while creating the document. Please fix the error and
 try again. Error message: ${caught.message}`,
        );
        throw caught;
      }
    }
  },
);

const ec2HelloWorld = new ScenarioOutput(
  "ec2HelloWorld",
  `Now you have the option of running a command on an EC2 instance that echoes
 'Hello, world!'. In order to run this command, you must provide the instance ID
 of a Linux EC2 instance. If you do not already have a running Linux EC2 instance
 in your account, you can create one using the AWS console. For information about
 creating an EC2 instance, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
ec2-launch-instance-wizard.html.`,
);

const enterIdOrSkipEC2HelloWorld = new ScenarioInput(
  "enterIdOrSkipEC2HelloWorld",
  "Enter your EC2 InstanceId or press enter to skip this step: ",
  { type: "input", default: "" },
);

const sdkEC2HelloWorld = new ScenarioAction(
  "sdkEC2HelloWorld",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new SendCommandCommand({
          DocumentName: state.documentName,
          InstanceIds: [state.ec2InstanceId],
```

```
          TimeoutSeconds: COMMAND_TIMEOUT_DURATION_SECONDS,
        }),
      );
      state.CommandId = response.Command.CommandId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while sending the command. Please fix the error and try
 again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const sdkGetCommandTime = new ScenarioAction(
  "sdkGetCommandTime",
  async (/** @type {State} */ state) => {
    const listInvocationsPaginated = [];
    console.log(
      "Let's get the time when the specific command was sent to the specific managed
 node.",
    );

    console.log(
      `First, we'll wait for the command to finish executing. This may take up to
 ${COMMAND_TIMEOUT_DURATION_SECONDS} seconds.`,
    );
    const commandExecutedResult = waitUntilCommandExecuted(
      { client: state.ssmClient },
      {
        CommandId: state.CommandId,
        InstanceId: state.ec2InstanceId,
      },
    );
    // This is necessary because the TimeoutSeconds of SendCommandCommand is only
 for the delivery, not execution.
    try {
      await new Promise((_, reject) =>
        setTimeout(
```

```
          reject,
          COMMAND_TIMEOUT_DURATION_SECONDS * 1000,
          new Error("Command Timed Out"),
        ),
      );
    } catch (caught) {
      if (caught.message === "Command Timed Out") {
        commandExecutedResult.state = "TIMED_OUT";
      } else {
        throw caught;
      }
    }

    if (commandExecutedResult.state !== "SUCCESS") {
      console.log(
        `The command with id: ${state.CommandId} did not execute in the allotted
time. Canceling command.`,
      );
      state.ssmClient.send(
        new CancelCommandCommand({
          CommandId: state.CommandId,
        }),
      );
      state.enterIdOrSkipEC2HelloWorld === "";
      return;
    }

    for await (const page of paginateListCommandInvocations(
      { client: state.ssmClient },
      { CommandId: state.CommandId },
    )) {
      listInvocationsPaginated.push(...page.CommandInvocations);
    }
    /**
     * @type {import('@aws-sdk/client-ssm').CommandInvocation}
     */
    const commandInvocation = listInvocationsPaginated.shift(); // Because the call
was made with CommandId, there's only one result, so shift it off.
    state.requestedDateTime = commandInvocation.RequestedDateTime;

    console.log(
      `The command invocation happened at: ${state.requestedDateTime}.`,
    );
  },
```

```
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const createSSMOpsItem = new ScenarioOutput(
  "createSSMOpsItem",
  `Now we will create a Systems Manager OpsItem. An OpsItem is a feature provided by
 the Systems Manager service. It is a type of operational data item that allows you
 to manage and track various operational issues, events, or tasks within your AWS
 environment.
You can create OpsItems to track and manage operational issues as they arise. For
 example, you could create an OpsItem whenever your application detects a critical
 error or an anomaly in your infrastructure.`,
);

const sdkCreateSSMOpsItem = new ScenarioAction(
  "sdkCreateSSMOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateOpsItemCommand({
          Description: "Created by the System Manager Javascript API",
          Title: "Disk Space Alert",
          Source: "EC2",
          Category: "Performance",
          Severity: "2",
        }),
      );
      state.opsItemId = response.OpsItemId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the ops item. Please fix the error and try
 again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const updateOpsItem = new ScenarioOutput(
  "updateOpsItem",
```

```
  (/** @type {State} */ state) =>
    `Now we will update the OpsItem: ${state.opsItemId}`,
);

const sdkUpdateOpsItem = new ScenarioAction(
  "sdkUpdateOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Description: `An update to ${state.opsItemId}`,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and try
 again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const getOpsItemStatus = new ScenarioOutput(
  "getOpsItemStatus",
  (/** @type {State} */ state) =>
    `Now we will get the status of the OpsItem: ${state.opsItemId}`,
);

const sdkOpsItemStatus = new ScenarioAction(
  "sdkGetOpsItemStatus",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new DescribeOpsItemsCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      state.opsItemStatus = response.OpsItemStatus;
    } catch (caught) {
      console.error(caught.message);
      console.log(
```

```
        `An error occurred while describing the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const resolveOpsItem = new ScenarioOutput(
  "resolveOpsItem",
  (/** @type {State} */ state) =>
    `Now we will resolve the OpsItem: ${state.opsItemId}`,
);

const sdkResolveOpsItem = new ScenarioAction(
  "sdkResolveOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Status: OpsItemStatus.RESOLVED,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and try
 again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
  "askToDeleteResources",
  "Would you like to delete the Systems Manager resources created during this
 example run?",
  { type: "confirm" },
);

const confirmDeleteChoice = new ScenarioOutput(
  "confirmDeleteChoice",
```

```
  (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You chose to delete the resources.";
    }
    return "The Systems Manager resources will not be deleted. Please delete them
 manually to avoid charges.";
  },
);

export const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    try {
      await state.ssmClient.send(
        new DeleteOpsItemCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      console.log(`The ops item: ${state.opsItemId} was successfully deleted.`);
    } catch (caught) {
      console.log(
        `There was a problem deleting the ops item: ${state.opsItemId}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          WindowId: state.winId,
        }),
      );
      console.log(
        `The maintenance window: ${state.maintenanceWindow} was successfully
 deleted.`,
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the maintenance window: ${state.opsItemId}.
 Please delete it manually. Error: ${caught.message}`,
      );
    }
```

```
    try {
      await state.ssmClient.send(
        new DeleteDocumentCommand({
          Name: state.documentName,
        }),
      );
      console.log(
        `The document: ${state.documentName} was successfully deleted.`,
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the document: ${state.documentName}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the Systems Manager Basics scenario for the AWS Javascript SDK v3.
 Thank you!",
);

const myScenario = new Scenario(
  "SSM Basics",
  [
    greet,
    pressEnter,
    createMaintenanceWindow,
    getMaintenanceWindow,
    sdkCreateMaintenanceWindow,
    modifyMaintenanceWindow,
    pressEnter,
    sdkModifyMaintenanceWindow,
    createSystemsManagerActions,
    getDocumentName,
    sdkCreateSSMDoc,
    ec2HelloWorld,
    enterIdOrSkipEC2HelloWorld,
    sdkEC2HelloWorld,
    sdkGetCommandTime,
    pressEnter,
```

```
    createSSMOpsItem,
    pressEnter,
    sdkCreateSSMOpsItem,
    updateOpsItem,
    pressEnter,
    sdkUpdateOpsItem,
    getOpsItemStatus,
    pressEnter,
    sdkOpsItemStatus,
    resolveOpsItem,
    pressEnter,
    sdkResolveOpsItem,
    askToDeleteResources,
    confirmDeleteChoice,
    sdkDeleteResources,
    goodbye,
  ],
  { ssmClient: new SSMClient({}) },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript .
  - CreateDocument

- CreateMaintenanceWindow
- CreateOpsItem
- DeleteMaintenanceWindow
- ListCommandInvocations
- SendCommand
- UpdateOpsItem

Ações

`CreateDocument`

O código de exemplo a seguir mostra como usar `CreateDocument`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ content, name, documentType }) => {
  const client = new SSMClient({});
  try {
    const { documentDescription } = await client.send(
      new CreateDocumentCommand({
        Content: content, // The content for the new SSM document. The content must
 not exceed 64KB.
        Name: name,
        DocumentType: documentType, // Document format type can be JSON, YAML, or
 TEXT. The default format is JSON.
      }),
```

```
    );
    console.log("Document created successfully.");
    return { DocumentDescription: documentDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DocumentAlreadyExists") {
      console.warn(`${caught.message}. Did you provide a new document name?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateDocument em Referência AWS SDK para JavaScript da API.

`CreateMaintenanceWindow`

O código de exemplo a seguir mostra como usar `CreateMaintenanceWindow`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM maintenance window.
 * @param {{ name: string, allowUnassociatedTargets: boolean, duration: number,
 cutoff: number, schedule: string, description?: string }}
 */
export const main = async ({
  name,
  allowUnassociatedTargets, // Allow the maintenance window to run on managed nodes,
 even if you haven't registered those nodes as targets.
  duration, // The duration of the maintenance window in hours.
```

```
  cutoff, // The number of hours before the end of the maintenance window that
 Amazon Web Services Systems Manager stops scheduling new tasks for execution.
  schedule, // The schedule of the maintenance window in the form of a cron or rate
 expression.
  description = undefined,
}) => {
  const client = new SSMClient({});

  try {
    const { windowId } = await client.send(
      new CreateMaintenanceWindowCommand({
        Name: name,
        Description: description,
        AllowUnassociatedTargets: allowUnassociatedTargets, // Allow the maintenance
 window to run on managed nodes, even if you haven't registered those nodes as
 targets.
        Duration: duration, // The duration of the maintenance window in hours.
        Cutoff: cutoff, // The number of hours before the end of the maintenance
 window that Amazon Web Services Systems Manager stops scheduling new tasks for
 execution.
        Schedule: schedule, // The schedule of the maintenance window in the form of
 a cron or rate expression.
      }),
    );
    console.log(`Maintenance window created with Id: ${windowId}`);
    return { WindowId: windowId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateMaintenanceWindow em Referência AWS SDK para JavaScript da API.

`CreateOpsItem`

O código de exemplo a seguir mostra como usar `CreateOpsItem`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { CreateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM OpsItem.
 * @param {{ title: string, source: string, category?: string, severity?: string }}
 */
export const main = async ({
  title,
  source,
  category = undefined,
  severity = undefined,
}) => {
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new CreateOpsItemCommand({
        Title: title,
        Source: source, // The origin of the OpsItem, such as Amazon EC2 or Systems
 Manager.
        Category: category,
        Severity: severity,
      }),
    );
    console.log(`Ops item created with id: ${opsItemId}`);
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte CreateOpsItem em Referência AWS SDK para JavaScript da API.

`DeleteDocument`

O código de exemplo a seguir mostra como usar `DeleteDocument`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM document.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(new DeleteDocumentCommand({ Name: documentName }));
    console.log(`Document '${documentName}' deleted.`);
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteDocument em Referência AWS SDK para JavaScript da API.

`DeleteMaintenanceWindow`

O código de exemplo a seguir mostra como usar `DeleteMaintenanceWindow`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { DeleteMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM maintenance window.
 * @param {{ windowId: string }}
 */
export const main = async ({ windowId }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new DeleteMaintenanceWindowCommand({ WindowId: windowId }),
    );
    console.log(`Maintenance window '${windowId}' deleted.`);
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte DeleteMaintenanceWindow em Referência AWS SDK para JavaScript da API.

`DescribeOpsItems`

O código de exemplo a seguir mostra como usar `DescribeOpsItems`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import {
  OpsItemFilterOperator,
  OpsItemFilterKey,
  paginateDescribeOpsItems,
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Describe SSM OpsItems.
 * @param {{ opsItemId: string }}
 */
export const main = async ({ opsItemId }) => {
  const client = new SSMClient({});
  try {
    const describeOpsItemsPaginated = [];
    for await (const page of paginateDescribeOpsItems(
      { client },
      {
        OpsItemFilters: {
          Key: OpsItemFilterKey.OPSITEM_ID,
          Operator: OpsItemFilterOperator.EQUAL,
          Values: opsItemId,
        },
      },
    )) {
      describeOpsItemsPaginated.push(...page.OpsItemSummaries);
```

```
    }
    console.log("Here are the ops items:");
    console.log(describeOpsItemsPaginated);
    return { OpsItemSummaries: describeOpsItemsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte DescribeOpsItems em Referência AWS SDK para JavaScript da API.

`ListCommandInvocations`

O código de exemplo a seguir mostra como usar `ListCommandInvocations`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { paginateListCommandInvocations, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * List SSM command invocations on an instance.
 * @param {{ instanceId: string }}
 */
export const main = async ({ instanceId }) => {
  const client = new SSMClient({});
  try {
    const listCommandInvocationsPaginated = [];
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListCommandInvocations(
```

```
      { client },
      {
        InstanceId: instanceId,
      },
    );
    for await (const page of paginator) {
      listCommandInvocationsPaginated.push(...page.CommandInvocations);
    }
    console.log("Here is the list of command invocations:");
    console.log(listCommandInvocationsPaginated);
    return { CommandInvocations: listCommandInvocationsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid instance ID?`);
    }
    throw caught;
  }
};
```

- Para obter detalhes da API, consulte ListCommandInvocations em Referência AWS SDK para JavaScript da API.

`SendCommand`

O código de exemplo a seguir mostra como usar `SendCommand`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { SendCommandCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Send an SSM command to a managed node.
 * @param {{ documentName: string }}
```

```
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new SendCommandCommand({
        DocumentName: documentName,
      }),
    );
    console.log("Command sent successfully.");
    return { Success: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid document name?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte SendCommand em Referência AWS SDK para JavaScript da API.

`UpdateMaintenanceWindow`

O código de exemplo a seguir mostra como usar `UpdateMaintenanceWindow`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { UpdateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Update an SSM maintenance window.
```

```
 * @param {{ windowId: string, allowUnassociatedTargets?: boolean, duration?:
 number, enabled?: boolean, name?: string, schedule?: string }}
 */
export const main = async ({
  windowId,
  allowUnassociatedTargets = undefined, //Allow the maintenance window to run on
 managed nodes, even if you haven't registered those nodes as targets.
  duration = undefined, //The duration of the maintenance window in hours.
  enabled = undefined,
  name = undefined,
  schedule = undefined, //The schedule of the maintenance window in the form of a
 cron or rate expression.
}) => {
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new UpdateMaintenanceWindowCommand({
        WindowId: windowId,
        AllowUnassociatedTargets: allowUnassociatedTargets,
        Duration: duration,
        Enabled: enabled,
        Name: name,
        Schedule: schedule,
      }),
    );
    console.log("Maintenance window updated.");
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Are these values correct?`);
    } else {
      throw caught;
    }
  }
};
```

- Para obter detalhes da API, consulte UpdateMaintenanceWindow em Referência AWS SDK para JavaScript da API.

`UpdateOpsItem`

O código de exemplo a seguir mostra como usar `UpdateOpsItem`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import { UpdateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Update an SSM OpsItem.
 * @param {{ opsItemId: string, status?: OpsItemStatus }}
 */
export const main = async ({
  opsItemId,
  status = undefined, // The OpsItem status. Status can be Open, In Progress, or
 Resolved
}) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new UpdateOpsItemCommand({
        OpsItemId: opsItemId,
        Status: status,
      }),
    );
    console.log("Ops item updated.");
    return { Success: true };
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "OpsItemLimitExceededException"
    ) {
      console.warn(
        `Couldn't create ops item because you have exceeded your open OpsItem limit.
 ${caught.message}.`,
      );
    } else {
      throw caught;
    }
```

```
  }
};
```

- Para obter detalhes da API, consulte UpdateOpsItem em Referência AWS SDK para JavaScript da API.

Exemplos do Amazon Textract usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Textract.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar um aplicativo React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado dentro das funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
```

```
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
```

```
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
```

```
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);
```

```
  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon Transcribe usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Transcribe.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DeleteMedicalTranscriptionJob`

O código de exemplo a seguir mostra como usar `DeleteMedicalTranscriptionJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Excluir um trabalho de transcrição médica.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // For example,
 'medical_transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteMedicalTranscriptionJob em Referência AWS SDK para JavaScript da API.

`DeleteTranscriptionJob`

O código de exemplo a seguir mostra como usar `DeleteTranscriptionJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir um trabalho de transcrição.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME", // Required. For example, 'transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte DeleteTranscriptionJob em Referência AWS SDK para JavaScript da API.

`ListMedicalTranscriptionJobs`

O código de exemplo a seguir mostra como usar `ListMedicalTranscriptionJobs`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Listar trabalhos de transcrição médica.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the same
 region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListMedicalTranscriptionJobs em Referência AWS SDK para JavaScript da API.

`ListTranscriptionJobs`

O código de exemplo a seguir mostra como usar `ListTranscriptionJobs`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar trabalhos de transcrição.

```
// Import the required AWS SDK clients and commands for Node.js

import { ListTranscriptionJobsCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  JobNameContains: "KEYWORD", // Not required. Returns only transcription
  // job names containing this string
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new ListTranscriptionJobsCommand(params),
    );
    console.log("Success", data.TranscriptionJobSummaries);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
```

```
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte ListTranscriptionJobs em Referência AWS SDK para JavaScript da API.

StartMedicalTranscriptionJob

O código de exemplo a seguir mostra como usar `StartMedicalTranscriptionJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e veja como configurar e executar no Repositório de exemplos de código da AWS.

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Iniciar um trabalho de transcrição médica.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
```

```
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the same
 region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte StartMedicalTranscriptionJob em Referência AWS SDK para JavaScript da API.

StartTranscriptionJob

O código de exemplo a seguir mostra como usar `StartTranscriptionJob`.

SDK para JavaScript (v3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Iniciar um trabalho de transcrição.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME",
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_LOCATION",
    // For example, "https://transcribe-demo.s3-REGION.amazonaws.com/
hello_world.wav"
  },
  OutputBucketName: "OUTPUT_BUCKET_NAME",
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Crie o cliente.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para JavaScript.
- Para obter detalhes da API, consulte StartTranscriptionJob em Referência AWS SDK para JavaScript da API.

Cenários

Criar uma aplicação de transmissão do Amazon Transcribe

O exemplo de código a seguir mostra como construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail.

SDK para JavaScript (v3)

Mostra como usar o Amazon Transcribe para construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Exemplos do Amazon Translate usando o SDK para JavaScript (v3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para JavaScript (v3) com o Amazon Translate.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação de transmissão do Amazon Transcribe

O exemplo de código a seguir mostra como construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail.

SDK para JavaScript (v3)

Mostra como usar o Amazon Transcribe para construir uma aplicação que registra, transcreve e traduz áudio ao vivo em tempo real, e envia os resultados por e-mail usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Criar um chatbot Amazon Lex

O exemplo de código a seguir mostra como criar um chatbot para engajar os visitantes do seu site.

SDK para JavaScript (v3)

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o exemplo completo Como criar um chatbot Amazon Lex no guia do AWS SDK para JavaScript desenvolvedor.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para JavaScript (v3)

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub. Os trechos a seguir mostram como o AWS SDK para JavaScript é usado dentro das funções do Lambda.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
```

```
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
```

```
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
```

```
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });
```

```
  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos de código para SDK para Java 2.x

Os exemplos de código a seguir mostram como usar o AWS SDK for Java 2.x with AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para Java 2.x — Saiba mais sobre como usar o Java com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos do ACM usando o SDK para Java 2.x
- Exemplos da API Gateway usando o SDK para Java 2.x
- Exemplos do Application Auto Scaling usando o SDK para Java 2.x
- Exemplos do Application Recovery Controller usando o SDK para Java 2.x
- Exemplos do Aurora usando o SDK para Java 2.x
- Exemplos do Auto Scaling usando o SDK para Java 2.x
- AWS Batch exemplos de uso do SDK para Java 2.x
- Exemplos do Amazon Bedrock usando o SDK para Java 2.x
- Exemplos do Amazon Bedrock Runtime usando o SDK para Java 2.x
- CloudFront exemplos de uso do SDK para Java 2.x
- CloudWatch exemplos de uso do SDK para Java 2.x
- CloudWatch Exemplos de eventos usando o SDK para Java 2.x
- CloudWatch Exemplos de registros usando o SDK para Java 2.x
- Exemplos do Amazon Cognito Identity usando o SDK para Java 2.x
- Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Java 2.x
- Exemplos do Amazon Comprehend usando o SDK para Java 2.x
- AWS Control Tower exemplos de uso do SDK para Java 2.x
- Exemplos do Firehose usando o SDK para Java 2.x
- Exemplos do Amazon DocumentDB usando o SDK para Java 2.x
- Exemplos do DynamoDB usando o SDK para Java 2.x
- Exemplos do Amazon EC2 usando o SDK para Java 2.x
- Exemplos do Amazon ECR usando o SDK para Java 2.x
- Exemplos do Amazon ECS usando o SDK para Java 2.x
- Exemplos do Elastic Load Balancing Versão 2 usando o SDK para Java 2.x
- MediaStore exemplos de uso do SDK para Java 2.x
- AWS Entity Resolution exemplos de uso do SDK para Java 2.x
- OpenSearch Exemplos de serviços usando o SDK para Java 2.x
- EventBridge exemplos de uso do SDK para Java 2.x

- EventBridge Exemplos de agendador usando o SDK para Java 2.x
- Exemplos do Forecast usando o SDK para Java 2.x
- Exemplos do Amazon Glacier usando o SDK para Java 2.x
- AWS Glue exemplos de uso do SDK para Java 2.x
- HealthImaging exemplos de uso do SDK para Java 2.x
- Exemplos do IAM usando o SDK para Java 2.x
- Exemplos do Amazon Inspector usando o SDK para Java 2.x
- AWS IoT exemplos de uso do SDK para Java 2.x
- AWS IoT data exemplos de uso do SDK para Java 2.x
- AWS IoT SiteWise exemplos de uso do SDK para Java 2.x
- Exemplos do Amazon Keyspaces usando o SDK para Java 2.x
- Exemplos do Kinesis usando o SDK para Java 2.x
- AWS KMS exemplos de uso do SDK para Java 2.x
- Exemplos do Lambda usando o SDK para Java 2.x
- Exemplos do Amazon Lex usando o SDK para Java 2.x
- Exemplos do Amazon Location usando o SDK para Java 2.x
- Exemplos do Location Service Places usando o SDK para Java 2.x
- AWS Marketplace Exemplos de API de catálogo usando o SDK para Java 2.x
- AWS Marketplace Exemplos de API de contrato usando o SDK para Java 2.x
- MediaConvert exemplos de uso do SDK para Java 2.x
- Exemplos do Migration Hub usando o SDK para Java 2.x
- Exemplos do Amazon MSK usando o SDK para Java 2.x
- Exemplos do Neptune usando o SDK para Java 2.x
- Exemplos da Central de Parceiros usando o SDK para Java 2.x
- Exemplos do Amazon Personalize usando o SDK para Java 2.x
- Exemplos de eventos do Amazon Personalize usando o SDK para Java 2.x
- Exemplos do Amazon Personalize Runtime usando o SDK para Java 2.x
- Exemplos do Amazon Pinpoint usando o SDK para Java 2.x
- Exemplos da API SMS and Voice do Amazon Pinpoint usando o SDK para Java 2.x

- Exemplos do Amazon Polly usando o SDK para Java 2.x
- Exemplos do Amazon RDS usando o SDK para Java 2.x
- Exemplos do Amazon RDS Data Service usando o SDK para Java 2.x
- Exemplos do Amazon Redshift usando o SDK para Java 2.x
- Exemplos do Amazon Rekognition usando o SDK para Java 2.x
- Exemplos para registro de domínios do Route 53 usando SDKs para Java 2.x
- Exemplos do Amazon S3 usando o SDK para Java 2.x
- Exemplos do Amazon S3 Control usando o SDK para Java 2.x
- Exemplos de buckets de diretório do S3 usando o SDK para Java 2.x
- SageMaker Exemplos de IA usando o SDK para Java 2.x
- Exemplos do Secrets Manager usando o SDK para Java 2.x
- Exemplos do Amazon SES usando o SDK para Java 2.x
- Exemplos do API v2 do Amazon SES usando o SDK para Java 2.x
- Exemplos do Amazon SNS usando o SDK para Java 2.x
- Exemplos do Amazon SQS usando o SDK para Java 2.x
- Exemplos do Step Functions usando o SDK para Java 2.x
- AWS STS exemplos de uso do SDK para Java 2.x
- Suporte exemplos de uso do SDK para Java 2.x
- Exemplos do Systems Manager usando o SDK para Java 2.x
- Exemplos do Amazon Textract usando o SDK para Java 2.x
- Exemplos do Amazon Transcribe usando o SDK para Java 2.x
- Exemplos do Amazon Transcribe Streaming usando o SDK para Java 2.x
- Exemplos do Amazon Translate usando o SDK para Java 2.x
- X-Ray exemplos de uso do SDK para Java 2.x

Exemplos do ACM usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o ACM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AddTagsToCertificate`

O código de exemplo a seguir mostra como usar `AddTagsToCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTagsToCertificate {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
```

```
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        addTags(certArn);
    }

    /**
     * Adds tags to a certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to add tags
 to
     */
    public static void addTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        AddTagsToCertificateRequest addTagsToCertificateRequest =
 AddTagsToCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.addTagsToCertificate(addTagsToCertificateRequest);
            System.out.println("Successfully added tags to a certificate");
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte AddTagsToCertificate em Referência AWS SDK for Java 2.x da API.

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        deleteCertificate(certArn);
    }

    /**
     * Deletes an SSL/TLS certificate from the AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to be
 deleted
     */
    public static void deleteCertificate( String certArn) {
```

```
        AcmClient acmClient = AcmClient.create();
        DeleteCertificateRequest request = DeleteCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.deleteCertificate(request);
            System.out.println("The certificate was deleted");

        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteCertificate em Referência AWS SDK for Java 2.x da API.

`DescribeCertificate`

O código de exemplo a seguir mostra como usar `DescribeCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class DescribeCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        describeCertificate(certArn);
    }

    /**
     * Describes the details of an SSL/TLS certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to describe
     * @throws AcmException if an error occurs while describing the certificate
     */
    public static void describeCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();
        DescribeCertificateRequest req = DescribeCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            DescribeCertificateResponse response =
 acmClient.describeCertificate(req);

            // Print the certificate details.
            System.out.println("Certificate ARN: " +
 response.certificate().certificateArn());
            System.out.println("Domain Name: " +
 response.certificate().domainName());
            System.out.println("Issued By: " + response.certificate().issuer());
            System.out.println("Issued On: " + response.certificate().issuedAt());
            System.out.println("Status: " + response.certificate().status());
        } catch (AcmException e) {
```

```
            System.out.println(e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeCertificate em Referência AWS SDK for Java 2.x da API.

`ExportCertificate`

O código de exemplo a seguir mostra como usar `ExportCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ExportCertificate {

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
```

```
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        exportCert(certArn);
    }

    /**
     * Exports an SSL/TLS certificate and its associated private key and certificate
chain from AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate that you
want to export.
     * @throws IOException If an I/O error occurs while reading the private key
passphrase file or exporting the certificate.
     */
    public static void exportCert(String certArn) throws IOException {
        AcmClient acmClient = AcmClient.create();

        // Initialize a file descriptor for the passphrase file.
        RandomAccessFile filePassphrase = null;
        ByteBuffer bufPassphrase = null;

        // Create a file stream for reading the private key passphrase.
        try {
            filePassphrase = new RandomAccessFile("C:\\AWS\\password.txt", "r");
        } catch (IllegalArgumentException | SecurityException |
FileNotFoundException ex) {
            throw ex;
        }

        // Create a channel to map the file.
        FileChannel channelPassphrase = filePassphrase.getChannel();

        // Map the file to the buffer.
        try {
            bufPassphrase = channelPassphrase.map(FileChannel.MapMode.READ_ONLY, 0,
channelPassphrase.size());
            channelPassphrase.close();
            filePassphrase.close();
        } catch (IOException ex) {
            throw ex;
        }
```

```
        // Create a request object.
        ExportCertificateRequest req = ExportCertificateRequest.builder()
            .certificateArn(certArn)
            .passphrase(SdkBytes.fromByteBuffer(bufPassphrase))
            .build();

        // Export the certificate.
        ExportCertificateResponse result = null;
        try {
            result = acmClient.exportCertificate(req);
        } catch (InvalidArnException | InvalidTagException |
 ResourceNotFoundException ex) {
            throw ex;
        }

        // Clear the buffer.
        bufPassphrase.clear();

        // Display the certificate and certificate chain.
        String certificate = result.certificate();
        System.out.println(certificate);

        String certificateChain = result.certificateChain();
        System.out.println(certificateChain);

        // This example retrieves but does not display the private key.
        String privateKey = result.privateKey();
        System.out.println("The example is complete");
    }
}
```

- Para obter detalhes da API, consulte ExportCertificate em Referência AWS SDK for Java 2.x da API.

`ImportCertificate`

O código de exemplo a seguir mostra como usar `ImportCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportCert {

    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <certificateKey> <privateKeyKey>

            Where:
                bucketName - The name of the S3 bucket containing the certificate
 and private key.
                certificateKey - The object key for the SSL/TLS certificate file in
 S3.
                privateKeyKey - The object key for the private key file in S3.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String certificateKey = args[1];
        String privateKeyKey = args[2];

        String certificateArn = importCertificate(bucketName, certificateKey,
 privateKeyKey);
        System.out.println("Certificate imported with ARN: " + certificateArn);
```

```
    }

    /**
     * Imports an SSL/TLS certificate and private key from S3 into AWS Certificate
Manager (ACM).
     *
     * @param bucketName     The name of the S3 bucket.
     * @param certificateKey The key for the SSL/TLS certificate file in S3.
     * @param privateKeyKey  The key for the private key file in S3.
     * @return The ARN of the imported certificate.
     */
    public static String importCertificate(String bucketName, String certificateKey,
String privateKeyKey) {
        AcmClient acmClient = AcmClient.create();
        S3Client s3Client = S3Client.create();

        try {
            byte[] certificateBytes = downloadFileFromS3(s3Client, bucketName,
certificateKey);
            byte[] privateKeyBytes = downloadFileFromS3(s3Client, bucketName,
privateKeyKey);

            ImportCertificateRequest request = ImportCertificateRequest.builder()

.certificate(SdkBytes.fromByteBuffer(ByteBuffer.wrap(certificateBytes)))

.privateKey(SdkBytes.fromByteBuffer(ByteBuffer.wrap(privateKeyBytes)))
                    .build();

            ImportCertificateResponse response =
acmClient.importCertificate(request);
            return response.certificateArn();

        } catch (IOException e) {
            System.err.println("Error downloading certificate or private key from
S3: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 error: " + e.awsErrorDetails().errorMessage());
        }
        return "";
    }

    /**
     * Downloads a file from Amazon S3 and returns its contents as a byte array.
```

```
     *
     * @param s3Client   The S3 client.
     * @param bucketName The name of the S3 bucket.
     * @param objectKey  The key of the object in S3.
     * @return The file contents as a byte array.
     * @throws IOException If an I/O error occurs.
     */
    private static byte[] downloadFileFromS3(S3Client s3Client, String bucketName,
 String objectKey) throws IOException {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

        try (ResponseInputStream<GetObjectResponse> s3Object =
 s3Client.getObject(getObjectRequest);
             ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream()) {
            IoUtils.copy(s3Object, byteArrayOutputStream);
            return byteArrayOutputStream.toByteArray();
        }
    }
}
```

- Para obter detalhes da API, consulte ImportCertificate em Referência AWS SDK for Java 2.x da API.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCerts {
    public static void main(String[] args) {
        listCertificates();
    }

    /**
     * Lists all the certificates managed by AWS Certificate Manager (ACM) that have
 a status of "ISSUED".
     */
    public static void listCertificates() {
        AcmClient acmClient = AcmClient.create();
        try {
            ListCertificatesRequest listRequest = ListCertificatesRequest.builder()
                .certificateStatuses(CertificateStatus.ISSUED)
                .maxItems(100)
                .build();
            ListCertificatesIterable listResponse =
 acmClient.listCertificatesPaginator(listRequest);

            // Print the certificate details using streams
            listResponse.certificateSummaryList().stream()
                .forEach(certificate -> {
                    System.out.println("Certificate ARN: " +
 certificate.certificateArn());
                    System.out.println("Certificate Domain Name: " +
 certificate.domainName());
                    System.out.println("Certificate Status: " +
 certificate.statusAsString());
                    System.out.println("---");
                });

        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ListCertificates em Referência AWS SDK for Java 2.x da API.

`ListTagsForCertificate`

O código de exemplo a seguir mostra como usar `ListTagsForCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCertTags {

    public static void main(String[] args) {

        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }
```

```
        String certArn = args[0];
        listCertTags(certArn);
    }

    /**
     * Lists the tags associated with an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the ACM certificate
     */
    public static void listCertTags(String certArn) {
        AcmClient acmClient = AcmClient.create();

        ListTagsForCertificateRequest request =
 ListTagsForCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        ListTagsForCertificateResponse response =
 acmClient.listTagsForCertificate(request);
        List<Tag> tagList = response.tags();
        tagList.forEach(tag -> {
            System.out.println("Key: " + tag.key());
            System.out.println("Value: " + tag.value());
        });
    }
}
```

- Para obter detalhes da API, consulte ListTagsForCertificate em Referência AWS SDK for Java 2.x da API.

`RemoveTagsFromCertificate`

O código de exemplo a seguir mostra como usar `RemoveTagsFromCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RemoveTagsFromCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        removeTags(certArn);
    }

    /**
     * Removes tags from an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate from which
 to remove tags
```

```
     */
    public static void removeTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        RemoveTagsFromCertificateRequest req =
 RemoveTagsFromCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.removeTagsFromCertificate(req);
            System.out.println("Successfully removed tags from the certificate");
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte RemoveTagsFromCertificate em Referência AWS SDK for Java 2.x da API.

`RenewCertificate`

O código de exemplo a seguir mostra como usar `RenewCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
```

```
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RenewCert {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        renewCertificate(certArn);
    }

    /**
     * Renews an existing SSL/TLS certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate to be
renewed.
     * @throws AcmException If there is an error renewing the certificate.
     */
    public static void renewCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();

        RenewCertificateRequest certificateRequest =
RenewCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.renewCertificate(certificateRequest);
            System.out.println("The certificate was renewed");
        } catch(AcmException e){
            System.out.println(e.getMessage());
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte RenewCertificate em Referência AWS SDK for Java 2.x da API.

`RequestCertificate`

O código de exemplo a seguir mostra como usar `RequestCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RequestCert {

    public static void main(String[] args) {
        requestCertificate();
    }

    /**
     * Requests a certificate from the AWS Certificate Manager (ACM) service.
     */
    public static void requestCertificate() {
        AcmClient acmClient = AcmClient.create();
        ArrayList<String> san = new ArrayList<>();
        san.add("www.example.com");
```

```
        RequestCertificateRequest req = RequestCertificateRequest.builder()
            .domainName("example.com")
            .idempotencyToken("1Aq25pTy")
            .subjectAlternativeNames(san)
            .build();

        try {
            RequestCertificateResponse response = acmClient.requestCertificate(req);
            System.out.println("Cert ARN IS " + response.certificateArn());
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte RequestCertificate em Referência AWS SDK for Java 2.x da API.

Exemplos da API Gateway usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com API Gateway.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

- AWS contribuições da comunidade

Ações

`CreateDeployment`

O código de exemplo a seguir mostra como usar `CreateDeployment`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

       try {
           CreateDeploymentRequest request = CreateDeploymentRequest.builder()
                   .restApiId(restApiId)
                   .description("Created using the AWS API Gateway Java API")
                   .stageName(stageName)
                   .build();

           CreateDeploymentResponse response =
apiGateway.createDeployment(request);
           System.out.println("The id of the deployment is " + response.id());
           return response.id();

       } catch (ApiGatewayException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte CreateDeployment em Referência AWS SDK for Java 2.x da API.

`CreateRestApi`

O código de exemplo a seguir mostra como usar `CreateRestApi`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createAPI(ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

       try {
           CreateRestApiRequest request = CreateRestApiRequest.builder()
                   .cloneFrom(restApiId)
                   .description("Created using the Gateway Java API")
                   .name(restApiName)
                   .build();

           CreateRestApiResponse response = apiGateway.createRestApi(request);
           System.out.println("The id of the new api is " + response.id());
           return response.id();

       } catch (ApiGatewayException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte CreateRestApi em Referência AWS SDK for Java 2.x da API.

`DeleteDeployment`

O código de exemplo a seguir mostra como usar `DeleteDeployment`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String
restApiId, String deploymentId) {

       try {
           DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
                   .restApiId(restApiId)
                   .deploymentId(deploymentId)
                   .build();

           apiGateway.deleteDeployment(request);
           System.out.println("Deployment was deleted");

       } catch (ApiGatewayException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DeleteDeployment em Referência AWS SDK for Java 2.x da API.

`DeleteRestApi`

O código de exemplo a seguir mostra como usar `DeleteRestApi`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void deleteAPI(ApiGatewayClient apiGateway, String restApiId) {

        try {
            DeleteRestApiRequest request = DeleteRestApiRequest.builder()
                    .restApiId(restApiId)
                    .build();

            apiGateway.deleteRestApi(request);
            System.out.println("The API was successfully deleted");

        } catch (ApiGatewayException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteRestApi em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Java 2.x

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Java.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos do Application Auto Scaling usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Application Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteScalingPolicy`

O código de exemplo a seguir mostra como usar `DeleteScalingPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReques
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """
```

```
            Usage:
               <tableId> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).\s
               policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }

    public static void deletePolicy(ApplicationAutoScalingClient
 appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
 ServiceNamespace ns, String tableId) {
        try {
            DeleteScalingPolicyRequest delSPRequest =
 DeleteScalingPolicyRequest.builder()
                .policyName(policyName)
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();
```

```
            appAutoScalingClient.deleteScalingPolicy(delSPRequest);
            System.out.println(policyName +" was deleted successfully.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the scaling policy was deleted
    public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        try {
            DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deregisterScalableTarget(targetRequest);
            System.out.println("The scalable target was deregistered.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
```

```
    public static void verifyTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
 DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- Para obter detalhes da API, consulte DeleteScalingPolicy em Referência AWS SDK for Java 2.x da API.

`RegisterScalableTarget`

O código de exemplo a seguir mostra como usar `RegisterScalableTarget`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
```

```
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyCon
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <roleARN> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).
               roleARN - The ARN of the role that has ApplicationAutoScaling
 permissions.
               policyName - The name of the policy to create.
```

```
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table, which
is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }

    public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
        try {
            RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .roleARN(roleARN)
                .minCapacity(5)
                .maxCapacity(10)
                .build();

            appAutoScalingClient.registerScalableTarget(targetRequest);
            System.out.println("You have registered " + tableId);
```

```
        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the target was created.
    public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    // Configure a scaling policy.
    public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
        // Check if the policy exists before creating a new one.
        DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
            .serviceNamespace(ns)
            .resourceId(tableId)
            .scalableDimension(tableWCUs)
            .build());

        if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
            // If policies exist, consider updating an existing policy instead of
creating a new one.
            System.out.println("Policy already exists. Consider updating it
instead.");
            List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
            for (ScalingPolicy pol : polList) {
                System.out.println("Policy name:" +pol.policyName());
            }
```

```
        } else {
            // If no policies exist, proceed with creating a new policy.
            PredefinedMetricSpecification specification =
 PredefinedMetricSpecification.builder()

 .predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
                .build();

            TargetTrackingScalingPolicyConfiguration policyConfiguration =
 TargetTrackingScalingPolicyConfiguration.builder()
                .predefinedMetricSpecification(specification)
                .targetValue(50.0)
                .scaleInCooldown(60)
                .scaleOutCooldown(60)
                .build();

            PutScalingPolicyRequest putScalingPolicyRequest =
 PutScalingPolicyRequest.builder()
                .targetTrackingScalingPolicyConfiguration(policyConfiguration)
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .policyName(policyName)
                .policyType(PolicyType.TARGET_TRACKING_SCALING)
                .build();

            try {
                appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
                System.out.println("You have successfully created a scaling policy
 for an Application Auto Scaling scalable target");
            } catch (ApplicationAutoScalingException e) {
                System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            }
        }
    }
}
```

- Para obter detalhes da API, consulte RegisterScalableTarget em Referência AWS SDK for Java 2.x da API.

Exemplos do Application Recovery Controller usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Application Recovery Controller.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetRoutingControlState`

O código de exemplo a seguir mostra como usar `GetRoutingControlState`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static GetRoutingControlStateResponse
 getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
```

```
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region())).build();
                return client.getRoutingControlState(
                        GetRoutingControlStateRequest.builder()
                                .routingControlArn(routingControlArn).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- Para obter detalhes da API, consulte GetRoutingControlState em Referência AWS SDK for Java 2.x da API.

`UpdateRoutingControlState`

O código de exemplo a seguir mostra como usar `UpdateRoutingControlState`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static UpdateRoutingControlStateResponse
 updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn,
            String routingControlState) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
```

```
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region()))
                        .build();
                return client.updateRoutingControlState(
                        UpdateRoutingControlStateRequest.builder()

.routingControlArn(routingControlArn).routingControlState(routingControlState).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- Para obter detalhes da API, consulte UpdateRoutingControlState em Referência AWS SDK for Java 2.x da API.

Exemplos do Aurora usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Aurora.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Aurora

O exemplo de código a seguir mostra como começar a usar o Aurora.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
```

```
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + " Arn
 = " + cluster.dbClusterArn()));
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK for Java 2.x .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de cluster do banco de dados do Aurora e definir os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Crie um snapshot do cluster do banco de dados e limpe os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
```

```
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
```

```
                "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group family
name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster identifier
value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n" +
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\"\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
```

```
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();

            rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();
```

```
            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
            String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();
```

```
            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .build();

           String endpoint = "";
           while (!instanceReady) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
               List<DBInstance> instanceList = response.dbInstances();
               for (DBInstance instance : instanceList) {
                   instanceReadyStr = instance.dbInstanceStatus();
                   if (instanceReadyStr.contains("available")) {
                       endpoint = instance.endpoint().address();
                       instanceReady = true;
                   } else {
                       System.out.print(".");
                       Thread.sleep(sleepTime * 1000);
                   }
               }
           }
           System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

       } catch (RdsException | InterruptedException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static String createDBInstanceCluster(RdsClient rdsClient,
           String dbInstanceIdentifier,
           String dbInstanceClusterIdentifier,
           String instanceClass) {
       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
```

```
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   // Modify the auto_increment_offset parameter.
   public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
       try {
           Parameter parameter1 = Parameter.builder()
                   .parameterName("auto_increment_offset")
                   .applyMethod("immediate")
                   .parameterValue("5")
                   .build();

           List<Parameter> paraList = new ArrayList<>();
           paraList.add(parameter1);
           ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dClusterGroupName)
                   .parameters(paraList)
                   .build();

           ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
           System.out.println(
                   "The parameter group " + response.dbClusterParameterGroupName()
+ " was successfully modified");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
```

```
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
```

```
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();
```

```
            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances

- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
           String dbClusterIdentifier, String userName, String password) {
       try {
           CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
                   .databaseName(dbName)
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .dbClusterParameterGroupName(dbParameterGroupFamily)
                   .engine("aurora-mysql")
                   .masterUsername(userName)
                   .masterUserPassword(password)
                   .build();

           CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
           return response.dbCluster().dbClusterArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK for Java 2.x .

`CreateDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
           String dbParameterGroupFamily) {
       try {
           CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CreateDBClusterParameterGroup em Referência AWS SDK for Java 2.x da API.

`CreateDBClusterSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBClusterSnapshot`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
           CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
           System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CreateDBClusterSnapshot em Referência AWS SDK for Java 2.x da API.

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createDBInstanceCluster(RdsClient rdsClient,
           String dbInstanceIdentifier,
           String dbInstanceClusterIdentifier,
           String instanceClass) {
       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .engine("aurora-mysql")
                   .dbInstanceClass(instanceClass)
                   .build();

           CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
           System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
           return response.dbInstance().dbInstanceArn();

       } catch (RdsException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
       return "";
   }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK for Java 2.x .

`DeleteDBCluster`

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK for Java 2.x .

`DeleteDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBClusterParameterGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(clusterDBARN) == 0) {
                       System.out.println(clusterDBARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
                   index++;
               }
           }

           DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                   .builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .build();

           rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
           System.out.println(dbClusterGroupName + " was deleted.");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteDBClusterParameterGroup em Referência AWS SDK for Java 2.x da API.

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK for Java 2.x .

`DescribeDBClusterParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameterGroups`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeDBClusterParameterGroups em Referência AWS SDK for Java 2.x da API.

`DescribeDBClusterParameters`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") == 0)) {
```

```
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterParameters em Referência AWS SDK for Java 2.x da API.

`DescribeDBClusterSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBClusterSnapshots`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
            String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");
```

```
            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBClusterSnapshots em Referência AWS SDK for Java 2.x da API.

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") == 0)) {
                   System.out.println("*** The parameter name is  " + paraName);
                   System.out.println("*** The parameter value is  " +
para.parameterValue());
                   System.out.println("*** The parameter data type is " +
para.dataType());
                   System.out.println("*** The parameter description is " +
para.description());
```

```
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK for Java 2.x .

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
```

```
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
                System.out.println("The version number of the database engine " +
engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions em Referência AWS SDK for Java 2.x da API.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
```

```
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK for Java 2.x .

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
                System.out.println("The version number of the database engine " +
engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptions em Referência AWS SDK for Java 2.x da API.

`ModifyDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ModifyDBClusterParameterGroup em Referência AWS SDK for Java 2.x da API.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Auto Scaling usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com Auto Scaling.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Auto Scaling

O exemplo de código a seguir mostra como começar a usar o Auto Scaling.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
        });
    }
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo do Amazon EC2 Auto Scaling com um modelo de inicialização e zonas de disponibilidade e obter informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas para CloudWatch métricas e, em seguida, limpe os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String ROLES_STACK = "MyCdkAutoScaleStack";
    public static void main(String[] args) throws InterruptedException {
```

```
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        String groupName = "MyAutoScalingGroup2";
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_WEST_2)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("First, we will create a launch template using a
CloudFormation script");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String launchTemplateName = stackOutputs.get("LaunchTemplateNameOutput");
        String vpcZoneId = getVPC(ec2);
        updateTemlate(ec2, launchTemplateName );
        System.out.println("The VPC zone id created by the CloudFormation stack
is"+vpcZoneId);

        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, ec2, groupName,
launchTemplateName, vpcZoneId);

        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group and cloud formation
resources");
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static String getVPC(Ec2Client ec2) {
        try {
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .filters(f -> f.name("isDefault").values("true"))
```

```
                    .build();

            DescribeVpcsResponse response = ec2.describeVpcs(request);

            if (!response.vpcs().isEmpty()) {
                Vpc defaultVpc = response.vpcs().get(0);
                System.out.println("Default VPC ID: " + defaultVpc.vpcId());
                return defaultVpc.vpcId();
            } else {
                System.out.println("No default VPC found.");
                return null; // Return null if no default VPC is found
            }

        } catch (Ec2Exception e) {
            System.err.println("EC2 error: " + e.awsErrorDetails().errorMessage());
            return null; // Return null in case of an error
        }
    }


    public static void updateTemlate(Ec2Client ec2, String launchTemplateName ) {
        // Step 1: Create new launch template version
        String newAmiId = "ami-0025f0db847eb6254";
        RequestLaunchTemplateData launchTemplateData =
RequestLaunchTemplateData.builder()
                .imageId(newAmiId)
                .build();

        CreateLaunchTemplateVersionRequest createVersionRequest =
CreateLaunchTemplateVersionRequest.builder()
                .launchTemplateName(launchTemplateName)
                .versionDescription("Updated with valid AMI")
                .sourceVersion("1")
                .launchTemplateData(launchTemplateData)
                .build();

        CreateLaunchTemplateVersionResponse createResponse =
ec2.createLaunchTemplateVersion(createVersionRequest);
        int newVersionNumber =
createResponse.launchTemplateVersion().versionNumber().intValue();

        // Step 2: Modify default version
        ModifyLaunchTemplateRequest modifyRequest =
ModifyLaunchTemplateRequest.builder()
```

```
                .launchTemplateName(launchTemplateName)
                .defaultVersion(String.valueOf(newVersionNumber))
                .build();

        ec2.modifyLaunchTemplate(modifyRequest);
        System.out.println("Updated launch template to version " + newVersionNumber
+ " with AMI " + newAmiId);
    }


    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
```

```
            System.out.println("You have set the DesiredCapacity to 2");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                              Ec2Client ec2Client,
                                              String groupName,
                                              String launchTemplateName,
                                              String vpcId) {
        try {
            // Step 1: Get one subnet ID in the given VPC
            DescribeSubnetsRequest subnetRequest = DescribeSubnetsRequest.builder()
                    .filters(Filter.builder().name("vpc-id").values(vpcId).build())
                    .build();

            DescribeSubnetsResponse subnetResponse =
ec2Client.describeSubnets(subnetRequest);

            if (subnetResponse.subnets().isEmpty()) {
                throw new RuntimeException("No subnets found in VPC: " + vpcId);
            }

            String subnetId = subnetResponse.subnets().get(0).subnetId(); // Use
first subnet
            System.out.println("Using subnet: " + subnetId);

            // Step 2: Create launch template reference
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            // Step 3: Create Auto Scaling group
            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .minSize(1)
                    .maxSize(1)
                    .vpcZoneIdentifier(subnetId)  // Correct: subnet ID, not VPC ID
```

```
                    .build();

            autoScalingClient.createAutoScalingGroup(request);

            // Step 4: Wait until group is created
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
                    waiter.waitUntilGroupExists(groupsRequest);

            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (Ec2Exception | AutoScalingException e) {
            System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }

    public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .maxRecords(10)
                    .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
```

```
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }

            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
```

```
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
       try {
           DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
           System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
           System.out.println("The current number of auto scaling groups is " +
response.numberOfAutoScalingGroups());

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
           String launchTemplateName) {
       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                   .launchTemplateName(launchTemplateName)
                   .build();

           UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                   .maxSize(3)
                   .autoScalingGroupName(groupName)
                   .launchTemplate(templateSpecification)
```

```
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group  " +
groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
```

```
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Ações

`CreateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `CreateAutoScalingGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
 <vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
```

```
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {

        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
```

```
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateAutoScalingGroup em Referência AWS SDK for Java 2.x da API.

`DeleteAutoScalingGroup`

O código de exemplo a seguir mostra como usar `DeleteAutoScalingGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
```

```
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
 String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroup em Referência AWS SDK for Java 2.x da API.

`DescribeAutoScalingGroups`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingGroups`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient, String
 groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
 group.autoScalingGroupName());
                System.out.println("The group ARN is " +
 group.autoScalingGroupARN());
```

```
                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK for Java 2.x da API.

`DescribeAutoScalingInstances`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient
```

```
.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstances em Referência AWS SDK for Java 2.x da API.

`DescribeScalingActivities`

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
```

```
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeScalingActivities em Referência AWS SDK for Java 2.x da API.

`DisableMetricsCollection`

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
```

```
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DisableMetricsCollection em Referência AWS SDK for Java 2.x da API.

`EnableMetricsCollection`

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte EnableMetricsCollection em Referência AWS SDK for Java 2.x da API.

`SetDesiredCapacity`

O código de exemplo a seguir mostra como usar `SetDesiredCapacity`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                   .autoScalingGroupName(groupName)
                   .desiredCapacity(2)
                   .build();

           autoScalingClient.setDesiredCapacity(capacityRequest);
           System.out.println("You have set the DesiredCapacity to 2");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte SetDesiredCapacity em Referência AWS SDK for Java 2.x da API.

`TerminateInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
       try {
           TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                   .instanceId(instanceId)
                   .shouldDecrementDesiredCapacity(false)
                   .build();

           autoScalingClient.terminateInstanceInAutoScalingGroup(request);
           System.out.println("You have terminated instance " + instanceId);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroup em Referência AWS SDK for Java 2.x da API.

`UpdateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `UpdateAutoScalingGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
           String launchTemplateName) {
       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                   .launchTemplateName(launchTemplateName)
                   .build();

           UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                   .maxSize(3)
                   .autoScalingGroupName(groupName)
                   .launchTemplate(templateSpecification)
                   .build();

           autoScalingClient.updateAutoScalingGroup(groupRequest);
           DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                   .autoScalingGroupNames(groupName)
                   .build();

           WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                   .waitUntilGroupInService(groupsRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("You successfully updated the auto scaling group  " +
groupName);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
    }
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroup em Referência AWS SDK for Java 2.x da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
public class Main {
```

```
    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }
```

```
   // Deletes the AWS resources used in this example.
   private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
           throws IOException, InterruptedException {
       loadBalancer.deleteLoadBalancer(lbName);
       System.out.println("*** Wait 30 secs for resource to be deleted");
       TimeUnit.SECONDS.sleep(30);
       loadBalancer.deleteTargetGroup(targetGroupName);
       autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
       autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
       autoScaler.deleteTemplate(templateName);
       database.deleteTable(tableName);
   }

   private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
       Scanner in = new Scanner(System.in);
       System.out.println(
               """
                       For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                       to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                       against various kinds of failures.

                       Some of the resources create by this demo are:
                       \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                       \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                       \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                       \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                       """);

       System.out.println("Press Enter when you're ready.");
       in.nextLine();
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Creating and populating a DynamoDB table named " +
tableName);
```

```
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
```

```
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);
```

```
                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
```

```
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
```

```
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                         \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                         healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns a
static response.
                The service still reports as healthy because health checks are still
shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
```

```
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
```

```
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }
```

```
   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
           System.out.println("See the current state of the service by selecting
one of the following choices:");
           for (int i = 0; i < actions.length; i++) {
               System.out.println(i + ": " + actions[i]);
           }

           try {
               System.out.print("\nWhich action would you like to take? ");
               int choice = scanner.nextInt();
               System.out.println("-".repeat(88));

               switch (choice) {
                   case 0 -> {
                       System.out.println("Request:\n");
                       System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                       CloseableHttpClient httpClient =
HttpClients.createDefault();

                       // Create an HTTP GET request to the ELB.
                       HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                       // Execute the request and get the response.
                       HttpResponse response = httpClient.execute(httpGet);
                       int statusCode = response.getStatusLine().getStatusCode();
                       System.out.println("HTTP Status Code: " + statusCode);

                       // Display the JSON response
                       BufferedReader reader = new BufferedReader(
                               new
InputStreamReader(response.getEntity().getContent()));
```

```
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                        target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
```

```
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
```

```
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
```

```
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
```

```
    * and deletes all the resources.
    */
   public void deleteInstanceProfile(String roleName, String profileName) {
       try {
           software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                   .builder()
                   .instanceProfileName(profileName)
                   .build();

           GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
           String name = response.instanceProfile().instanceProfileName();
           System.out.println(name);

           RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .roleName(roleName)
                   .build();

           getIAMClient().removeRoleFromInstanceProfile(profileRequest);
           DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .build();

           getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
           System.out.println("Deleted instance profile " + profileName);

           DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                   .roleName(roleName)
                   .build();

           // List attached role policies.
           ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                   .listAttachedRolePolicies(role -> role.roleName(roleName));
           List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
           for (AttachedPolicy attachedPolicy : attachedPolicies) {
               DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                       .roleName(roleName)
                       .policyArn(attachedPolicy.policyArn())
```

```
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
```

```
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
```

```
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();
```

```
        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```

```
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
```

```
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
```

```
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
 available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```

```
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();
```

```
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
{
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
```

```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

AWS Batch exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS Batch.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Batch

O exemplo de código a seguir mostra como começar a usar o AWS Batch.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

```
public class HelloBatch {
    private static BatchAsyncClient batchClient;

    public static void main(String[] args) {
        List<JobSummary> jobs = listJobs("my-job-queue");
        jobs.forEach(job ->
            System.out.printf("Job ID: %s, Job Name: %s, Job Status: %s%n",
                job.jobId(), job.jobName(), job.status())
        );
    }

    public static List<JobSummary> listJobs(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
 getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });

        future.join();
        return jobSummaries;
    }

    private static BatchAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)  // Increase max concurrency to handle more
 simultaneous connections.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();
```

```
        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
 call attempt timeout.
            .retryPolicy(RetryPolicy.builder()  // Add a retry policy to handle
 transient errors.
                .numRetries(3)  // Number of retry attempts.
                .build())
            .build();

        if (batchClient == null) {
            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }
}
```

- Para obter detalhes da API, consulte a lista JobsPaginator em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um ambiente AWS Batch computacional.
- Verificar o status do ambiente de computação.
- Configure uma fila de AWS Batch tarefas e uma definição de tarefas.
- Registrar uma definição de tarefa.
- Envie uma AWS Batch vaga.
- Obter uma lista de trabalhos aplicáveis à fila de trabalhos.
- Conferir o status do trabalho.

- Exclua AWS Batch recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS Batch recursos.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.ClientException;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Subnet;
import software.amazon.awssdk.services.ec2.model.Vpc;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * NOTE
 * This scenario submits a job that pulls a Docker image named echo-text from Amazon
 ECR to Amazon Fargate.
 *
 * To place this Docker image on Amazon ECR, run the following Basics scenario.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ecr
 *
 */
public class BatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    // Define two stacks used in this Basics Scenario.
    private static final String ROLES_STACK = "RolesStack";
    private static String defaultSubnet;
    private static String defaultSecurityGroup;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchScenario.class);

    public static void main(String[] args) throws InterruptedException {

        BatchActions batchActions = new BatchActions();
        Scanner scanner = new Scanner(System.in);
        String computeEnvironmentName = "my-compute-environment";
        String jobQueueName = "my-job-queue";
        String jobDefinitionName = "my-job-definition";


        // See the NOTE in this Java code example (at start).
        String dockerImage = "dkr.ecr.us-east-1.amazonaws.com/echo-text:echo-text";

        logger.info("""
            AWS Batch is a fully managed batch processing service that dynamically
 provisions the required compute
            resources for batch computing workloads. The Java V2 `BatchAsyncClient`
 allows
            developers to automate the submission, monitoring, and management of
 batch jobs.
```

```
            This scenario provides an example of setting up a compute environment,
 job queue and job definition,
            and then submitting a job.

            This scenario submits a job that pulls a Docker image named echo-text
 from Amazon ECR to Amazon Fargate.

            To place this Docker image on Amazon ECR, run the following Basics
 scenario.

            https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/
example_code/ecr

            Let's get started...

            You have two choices:

            1 - Run the entire program.
            2 - Delete an existing Compute Environment (created from a previous
 execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                logger.info("Continuing with the program...");
               // logger.info("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String jobQueueARN = String.valueOf(batchActions.
 describeJobQueueAsync(computeEnvironmentName));
                if (!jobQueueARN.isEmpty()) {
                    batchActions.disableJobQueueAsync(jobQueueARN);
                    countdown(1);
                    batchActions.deleteJobQueueAsync(jobQueueARN);
                }

                try {

 batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                        .exceptionally(ex -> {
                            logger.info("Disable compute environment failed: " +
 ex.getMessage());
```

```
                            return null;
                        })
                        .join();
                } catch (CompletionException ex) {
                    logger.info("Failed to disable compute environment: " +
ex.getMessage());
                }
                countdown(2);

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                return;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
        System.out.println(DASHES);

        waitForInputToContinue(scanner);
        // Get an AWS Account id used to retrieve the docker image from Amazon ECR.
        // Create a single-element array to store the `accountId` value.
        String[] accId = new String[1];
        CompletableFuture<String> accountIdFuture = batchActions.getAccountId();
        accountIdFuture.thenAccept(accountId -> {
            logger.info("Account ID: " + accountId);
            accId[0] = accountId;
        }).join();

        dockerImage = accId[0]+"."+dockerImage;

        // Get a default subnet and default security associated with the default
VPC.
        getSubnetSecurityGroup();

        logger.info("Use AWS CloudFormation to create two IAM roles that are
required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(ROLES_STACK);
        String batchIAMRole = stackOutputs.get("BatchRoleArn");
        String executionRoleARN = stackOutputs.get("EcsRoleArn");
```

```
        logger.info("The IAM role needed to interact with AWS Batch is
"+batchIAMRole);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("1. Create a Batch compute environment");
        logger.info("""
            A compute environment is a resource where you can run your batch jobs.
            After creating a compute environment, you can define job queues and job
definitions to submit jobs for
            execution.

            The benefit of creating a compute environment is it allows you to easily
configure and manage the compute
            resources that will be used to run your Batch jobs. By separating the
compute environment from the job definitions,
            you can easily scale your compute resources up or down as needed,
without having to modify your job definitions.
            This makes it easier to manage your Batch workloads and ensures that
your jobs have the necessary
            compute resources to run efficiently.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateComputeEnvironmentResponse> future =
batchActions.createComputeEnvironmentAsync(computeEnvironmentName, batchIAMRole,
defaultSubnet, defaultSecurityGroup);
            CreateComputeEnvironmentResponse response = future.join();
            logger.info("Compute Environment ARN: " +
response.computeEnvironmentArn());
        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The compute environment '" + computeEnvironmentName
+ "' already exists. Moving on...");
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
```

```
            } else {
                    logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rte.getMessage()));
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Check the status of the "+computeEnvironmentName +" Compute
Environment.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName);
            String status = future.join();
            logger.info("Compute Environment Status: " + status);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a job queue");
        logger.info("""
             A job queue is an essential component that helps manage the execution
of your batch jobs.
             It acts as a buffer, where jobs are placed and then scheduled for
execution based on their
             priority and the available resources in the compute environment.
             """);
        waitForInputToContinue(scanner);
```

```
        String jobQueueArn = null;
        try {
            CompletableFuture<String> jobQueueFuture =
batchActions.createJobQueueAsync(jobQueueName, computeEnvironmentName);
            jobQueueArn = jobQueueFuture.join();
            logger.info("Job Queue ARN: " + jobQueueArn);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof BatchException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The job queue '" + jobQueueName + "' already
exists. Moving on...");
                    // Retrieve the ARN of the job queue.
                    CompletableFuture<String> jobQueueArnFuture =
batchActions.getJobQueueARN(jobQueueName);
                    jobQueueArn = jobQueueArnFuture.join();
                    logger.info("Job Queue ARN: " + jobQueueArn);
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return; // End the execution
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Register a Job Definition.");
        logger.info("""
            Registering a job in AWS Batch using the Fargate launch type ensures
that all
            necessary parameters, such as the execution role, command to run, and so
on
            are specified and reused across multiple job submissions.

             The job definition pulls a Docker image from Amazon ECR and executes
the Docker image.
```

```
            """);

        waitForInputToContinue(scanner);
        String jobARN;
        try {
            String platform = "";
            while (true) {
                logger.info("""
                    On which platform/CPU architecture combination did you build the
Docker image?:
                    1. Windows       X86_64
                    2. Mac or Linux  ARM64
                    3. Mac or Linux  X86_64

                    Please select 1, 2, or 3.
                    """);
                String platAns = scanner.nextLine().trim();
                if (platAns.equals("1")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("2")) {
                    platform = "ARM64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("3")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else {
                    System.out.println("Invalid input. Please select either 1 or
2.");
                }
            }

            jobARN = batchActions.registerJobDefinitionAsync(jobDefinitionName,
executionRoleARN, dockerImage, platform)
                .exceptionally(ex -> {
                    System.err.println("Register job definition failed: " +
ex.getMessage());
                    return null;
                })
                .join();
            if (jobARN != null) {
                logger.info("Job ARN: " + jobARN);
            }
        } catch (RuntimeException rte) {
```

```
            logger.error("A Batch exception occurred while registering the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Submit an AWS Batch job from a job definition.");
        waitForInputToContinue(scanner);
        String jobId;
        try {
            jobId = batchActions.submitJobAsync(jobDefinitionName, jobQueueName,
jobARN)
                .exceptionally(ex -> {
                    System.err.println("Submit job failed: " + ex.getMessage());
                    return null;
                })
                .join();

            logger.info("The job id is "+jobId);
            logger.info("Let's wait 2 minutes for the job to complete");
            countdown(2);

        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info(DASHES);
        logger.info("6. Get a list of jobs applicable to the job queue.");

        waitForInputToContinue(scanner);
        try {
            List<JobSummary> jobs = batchActions.listJobsAsync(jobQueueName);
            jobs.forEach(job ->
                logger.info("Job ID: {}, Job Name: {}, Job Status: {}", job.jobId(),
job.jobName(), job.status()));

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
```

```
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Check the status of job "+jobId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future = batchActions.describeJobAsync(jobId);
            String jobStatus = future.join();
            logger.info("Job Status: " + jobStatus);

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info("8. Delete Batch resources");
        logger.info(
            """
            When deleting an AWS Batch compute environment, it does not happen
instantaneously.
            There is typically a delay, similar to some other AWS resources.
            AWS Batch starts the deletion process.
            """);
        logger.info("Would you like to delete the AWS Batch resources such as the
compute environment? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS ECR resources.");
            logger.info("First, we will deregister the Job Definition.");
            waitForInputToContinue(scanner);
            try {
                batchActions.deregisterJobDefinitionAsync(jobARN)
                    .exceptionally(ex -> {
                        logger.info("Deregister job definition failed: " +
ex.getMessage());
                        return null;
```

```
                    })
                    .join();
                logger.info(jobARN + " was deregistered");
            } catch (RuntimeException rte) {
                logger.error("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            logger.info("Second, we will disable and then delete the Job Queue.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableJobQueueAsync(jobQueueArn)
                    .exceptionally(ex -> {
                        logger.info("Disable job queue failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobQueueArn + " was disabled");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
            try {
                CompletableFuture<Void> future =
batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
                future.join();
                logger.info("Job queue is now disabled.");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            try {
                batchActions.deleteJobQueueAsync(jobQueueArn);
                logger.info(jobQueueArn +" was deleted");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
```

```
                return;
            }
            logger.info("Let's wait 2 minutes for the job queue to be deleted");
            countdown(2);
            waitForInputToContinue(scanner);

            logger.info("Third, we will delete the Compute Environment.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                    .exceptionally(ex -> {
                        System.err.println("Disable compute environment failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info("Compute environment disabled") ;
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }


batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName).thenAccept(state
-> {
                logger.info("Current State: " + state);
            }).join();

            logger.info("Lets wait 1 min for the compute environment to be
deleted");
            countdown(1);

            try {

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                logger.info(computeEnvironmentName +" was deleted.");

            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            waitForInputToContinue(scanner);
```

```
            CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Batch SDK scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int minutes) throws InterruptedException {
        int seconds = 0;
        for (int i = minutes * 60 + seconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.print(String.format("\r%02d:%02d", displayMinutes,
displaySeconds));
            Thread.sleep(1000); // Wait for 1 second
        }
        logger.info("Countdown complete!");
    }

    private static void getSubnetSecurityGroup() {
        try (Ec2AsyncClient ec2Client = Ec2AsyncClient.create()) {
            CompletableFuture<Vpc> defaultVpcFuture =
ec2Client.describeVpcs(DescribeVpcsRequest.builder()
                            .filters(Filter.builder()
                                    .name("is-default")
                                    .values("true")
```

```
                                        .build())
                                .build())
                        .thenApply(response -> response.vpcs().stream()
                                .findFirst()
                                .orElseThrow(() -> new RuntimeException("Default VPC not
found")));

            CompletableFuture<String> defaultSubnetFuture = defaultVpcFuture
                        .thenCompose(vpc ->
ec2Client.describeSubnets(DescribeSubnetsRequest.builder()
                                        .filters(Filter.builder()
                                                        .name("vpc-id")
                                                        .values(vpc.vpcId())
                                                        .build(),
                                                Filter.builder()
                                                        .name("default-for-az")
                                                        .values("true")
                                                        .build())
                                        .build())
                                .thenApply(DescribeSubnetsResponse::subnets)
                                .thenApply(subnets -> subnets.stream()
                                        .findFirst()
                                        .map(Subnet::subnetId)
                                        .orElseThrow(() -> new RuntimeException("No
default subnet found"))));

            CompletableFuture<String> defaultSecurityGroupFuture = defaultVpcFuture
                        .thenCompose(vpc ->
ec2Client.describeSecurityGroups(DescribeSecurityGroupsRequest.builder()
                                        .filters(Filter.builder()
                                                        .name("group-name")
                                                        .values("default")
                                                        .build(),
                                                Filter.builder()
                                                        .name("vpc-id")
                                                        .values(vpc.vpcId())
                                                        .build())
                                        .build())

.thenApply(DescribeSecurityGroupsResponse::securityGroups)
                                .thenApply(securityGroups -> securityGroups.stream()
                                        .findFirst()
                                        .map(SecurityGroup::groupId)
```

```
                                    .orElseThrow(() -> new RuntimeException("No
 default security group found"))));

            defaultSubnet = defaultSubnetFuture.join();
            defaultSecurityGroup = defaultSecurityGroupFuture.join();
        }
    }
}
```

Uma classe de invólucro para métodos AWS Batch SDK.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.BatchClient;
import software.amazon.awssdk.services.batch.model.AssignPublicIp;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.CEState;
import software.amazon.awssdk.services.batch.model.CEType;
import software.amazon.awssdk.services.batch.model.CRType;
import software.amazon.awssdk.services.batch.model.ComputeEnvironmentOrder;
import software.amazon.awssdk.services.batch.model.ComputeResource;
import software.amazon.awssdk.services.batch.model.ContainerProperties;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.CreateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueResponse;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionResponse;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsRequest;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesRequest;
```

```
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobsRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobsResponse;
import software.amazon.awssdk.services.batch.model.JQState;
import software.amazon.awssdk.services.batch.model.JobDefinitionType;
import software.amazon.awssdk.services.batch.model.JobDetail;
import software.amazon.awssdk.services.batch.model.JobQueueDetail;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionResponse;
import software.amazon.awssdk.services.batch.model.NetworkConfiguration;
import software.amazon.awssdk.services.batch.model.PlatformCapability;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.ResourceRequirement;
import software.amazon.awssdk.services.batch.model.ResourceType;
import software.amazon.awssdk.services.batch.model.RuntimePlatform;
import software.amazon.awssdk.services.batch.model.SubmitJobRequest;
import software.amazon.awssdk.services.batch.model.CreateJobQueueResponse;
import java.time.Duration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicBoolean;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.services.batch.model.SubmitJobResponse;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueResponse;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import software.amazon.awssdk.services.sts.StsAsyncClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;

public class BatchActions {
    private static BatchAsyncClient batchClient;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchActions.class);

    private static BatchAsyncClient getAsyncClient() {
```

```
        if (batchClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }

    /**
     * Asynchronously creates a new compute environment in AWS Batch.
     *
     * @param computeEnvironmentName the name of the compute environment to create
     * @param batchIAMRole the IAM role to be used by the compute environment
     * @param subnet the subnet ID to be used for the compute environment
     * @param secGroup the security group ID to be used for the compute environment
     * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
     *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
     * @throws BatchException if there is an error creating the compute environment
     * @throws RuntimeException if there is an unexpected error during the operation
     */
    public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
        String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
```

```
        CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
            .computeEnvironmentName(computeEnvironmentName)
            .type(CEType.MANAGED)
            .state(CEState.ENABLED)
            .computeResources(ComputeResource.builder()
                .type(CRType.FARGATE)
                .maxvCpus(256)
                .subnets(Collections.singletonList(subnet))
                .securityGroupIds(Collections.singletonList(secGroup))
                .build())
            .serviceRole(batchIAMRole)
            .build();

        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
            }
        });

        return response;
    }

    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
```

```
                }
            });
    }

    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }

        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();

        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }

    /**
     * Creates a job queue asynchronously.
     *
     * @param jobQueueName the name of the job queue to create
```

```
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
           .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
               .computeEnvironment(computeEnvironmentName)
               .order(1)
               .build())
           .build();

       CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
           }
       });

       return response.thenApply(CreateJobQueueResponse::jobQueueArn);
   }

   /**
    * Asynchronously lists the jobs in the specified job queue with the given job
status.
    *
    * @param jobQueue the name of the job queue to list jobs from
    * @return a List<JobSummary> that contains the jobs that succeeded
```

```
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
    }

    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
     * When using Fargate as the compute environment, it is crucial to set the
     * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
     * ensure proper networking configuration for the Fargate tasks. This
     * allows the tasks to communicate with external services, access the
     * internet, or communicate within a VPC.
     *
     * @param jobDefinitionName the name of the job definition to be registered
     * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
     *                         that provides permissions for the containers in the
job
     * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
     * @return a CompletableFuture that completes with the ARN of the registered
     *         job definition upon successful execution, or completes exceptionally
with
     *         an error if the registration fails
     */
    public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
        NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
```

```
                .assignPublicIp(AssignPublicIp.ENABLED)
                .build();

        ContainerProperties containerProperties = ContainerProperties.builder()
                .image(image)
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
```

```
    }

    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has been
deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
            .build();

        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();
```

```
        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }

    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
     * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
     *         The future completes when the job queue has been successfully deleted
or if an error occurs.
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .build();

        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }

    /**
     * Asynchronously describes the job queue associated with the specified compute
environment.
     *
     * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
```

```
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
               String jobQueueARN;
               for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                   for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                       String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                       String name = getComputeEnvironmentName(computeEnvironment);
                       if (name.equals(computeEnvironmentName)) {
                           jobQueueARN = jobQueueDetail.jobQueueArn();
                           logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                       }
                   }
               }
           } else {
               throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(describeJobQueuesResponse -> {
           String jobQueueARN = "";
           for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
               for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                   String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                   String name = getComputeEnvironmentName(computeEnvironment);
                   if (name.equals(computeEnvironmentName)) {
                       jobQueueARN = jobQueueDetail.jobQueueArn();
```

```
                    }
                }
            }
            return jobQueueARN;
        });
    }

    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
    public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
        UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .state(CEState.DISABLED)
            .build();

        CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Submits a job asynchronously to the AWS Batch service.
     *
     * @param jobDefinitionName the name of the job definition to use
     * @param jobQueueName the name of the job queue to submit the job to
     * @param jobARN the Amazon Resource Name (ARN) of the job definition
     * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
     */
```

```
    public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
        SubmitJobRequest jobRequest = SubmitJobRequest.builder()
            .jobDefinition(jobARN)
            .jobName(jobDefinitionName)
            .jobQueue(jobQueueName)
            .build();

        CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }

    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }

    /**
     * Disables the specific job queue using the asynchronous Java client.
     *
```

```
    * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to wait
for
    * @return a {@link CompletableFuture} that completes when the job queue is
disabled
    */
   public CompletableFuture<Void> waitForJobQueueToBeDisabledAsync(String
jobQueueArn) {
       AtomicBoolean isDisabled = new AtomicBoolean(false);
       return CompletableFuture.runAsync(() -> {
           while (!isDisabled.get()) {
               DescribeJobQueuesRequest describeRequest =
DescribeJobQueuesRequest.builder()
                   .jobQueues(jobQueueArn)
                   .build();

               CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeRequest);
               responseFuture.whenComplete((describeResponse, ex) -> {
                   if (describeResponse != null) {
                       for (JobQueueDetail jobQueue : describeResponse.jobQueues())
{
                           if (jobQueue.jobQueueArn().equals(jobQueueArn) &&
jobQueue.state() == JQState.DISABLED) {
                               isDisabled.set(true);
                               break;
                           }
                       }
                   } else {
                       throw new RuntimeException("Error describing job queues",
ex);
                   }
               }).join();

               if (!isDisabled.get()) {
                   try {
                       logger.info("Waiting for job queue to be disabled...");
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       Thread.currentThread().interrupt();
                       throw new RuntimeException("Thread interrupted while waiting
for job queue to be disabled", e);
                   }
               }
           }
```

```
        }).whenComplete((result, throwable) -> {
            if (throwable != null) {
                throw new RuntimeException("Error while waiting for job queue to be
disabled", throwable);
            }
        });
    }

    public CompletableFuture<String> getJobQueueARN(String jobQueueName) {
        // Describe the job queue asynchronously
        CompletableFuture<DescribeJobQueuesResponse> describeJobQueuesFuture =
batchClient.describeJobQueues(
            DescribeJobQueuesRequest.builder()
                .jobQueues(jobQueueName)
                .build()
        );

        // Handle the asynchronous response and return the Job Queue ARN in the
CompletableFuture<String>
        CompletableFuture<String> jobQueueArnFuture = new CompletableFuture<>();
        describeJobQueuesFuture.whenComplete((response, error) -> {
            if (error != null) {
                if (error instanceof BatchException) {
                    logger.info("Batch error: " + ((BatchException)
error).awsErrorDetails().errorMessage());
                } else {
                    logger.info("Error describing job queue: " +
error.getMessage());
                }
                jobQueueArnFuture.completeExceptionally(new RuntimeException("Failed
to retrieve Job Queue ARN", error));
            } else {
                if (response.jobQueues().isEmpty()) {
                    jobQueueArnFuture.completeExceptionally(new
RuntimeException("Job queue not found: " + jobQueueName));
                } else {
                    // Assuming only one job queue is returned for the given name
                    String jobQueueArn = response.jobQueues().get(0).jobQueueArn();
                    jobQueueArnFuture.complete(jobQueueArn);
                }
            }
        });

        return jobQueueArnFuture;
```

```
    }

    private static String getComputeEnvironmentName(String computeEnvironment) {
        String[] parts = computeEnvironment.split("/");
        if (parts.length == 2) {
            return parts[1];
        }
        return null;
    }

    public CompletableFuture<String> getAccountId() {
        StsAsyncClient stsAsyncClient = StsAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        return stsAsyncClient.getCallerIdentity()
            .thenApply(GetCallerIdentityResponse::account);
    }


}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DeregisterJobDefinition
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - ListJobsPaginator
  - RegisterJobDefinition
  - SubmitJob
  - UpdateComputeEnvironment
  - UpdateJobQueue

Ações

`CreateComputeEnvironment`

O código de exemplo a seguir mostra como usar `CreateComputeEnvironment`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously creates a new compute environment in AWS Batch.
    *
    * @param computeEnvironmentName the name of the compute environment to create
    * @param batchIAMRole the IAM role to be used by the compute environment
    * @param subnet the subnet ID to be used for the compute environment
    * @param secGroup the security group ID to be used for the compute environment
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
    *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
    * @throws BatchException if there is an error creating the compute environment
    * @throws RuntimeException if there is an unexpected error during the operation
    */
   public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
       String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
       CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
           .computeEnvironmentName(computeEnvironmentName)
           .type(CEType.MANAGED)
           .state(CEState.ENABLED)
           .computeResources(ComputeResource.builder()
               .type(CRType.FARGATE)
               .maxvCpus(256)
               .subnets(Collections.singletonList(subnet))
               .securityGroupIds(Collections.singletonList(secGroup))
               .build())
```

```
            .serviceRole(batchIAMRole)
            .build();

        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
            }
        });

        return response;
    }
```

- Para obter detalhes da API, consulte CreateComputeEnvironment em Referência AWS SDK for Java 2.x da API.

`CreateJobQueue`

O código de exemplo a seguir mostra como usar `CreateJobQueue`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a job queue asynchronously.
     *
     * @param jobQueueName the name of the job queue to create
     * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
     * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
     */
```

```
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
           .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
               .computeEnvironment(computeEnvironmentName)
               .order(1)
               .build())
           .build();

       CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
           }
       });

       return response.thenApply(CreateJobQueueResponse::jobQueueArn);
   }
```

- Para obter detalhes da API, consulte CreateJobQueue em Referência AWS SDK for Java 2.x da API.

DeleteComputeEnvironment

O código de exemplo a seguir mostra como usar `DeleteComputeEnvironment`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
       DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .build();

       return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
           .whenComplete((response, ex) -> {
               if (ex != null) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof BatchException) {
                       throw new RuntimeException(cause);
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               }
           });
   }
```

- Para obter detalhes da API, consulte DeleteComputeEnvironment em Referência AWS SDK for Java 2.x da API.

`DeleteJobQueue`

O código de exemplo a seguir mostra como usar `DeleteJobQueue`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a Batch job queue asynchronously.
    *
    * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
    * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
    *         The future completes when the job queue has been successfully deleted
or if an error occurs.
    *         If successful, the future will be completed with a {@code Void}
value.
    *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
    */
   public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
       DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .build();

       CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
       return responseFuture.whenComplete((deleteResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(deleteResponse -> null);
   }
```

- Para obter detalhes da API, consulte DeleteJobQueue em Referência AWS SDK for Java 2.x da API.

`DeregisterJobDefinition`

O código de exemplo a seguir mostra como usar `DeregisterJobDefinition`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deregisters a job definition asynchronously.
    *
    * @param jobDefinition the name of the job definition to be deregistered
    * @return a CompletableFuture that completes when the job definition has been
deregistered
    * or an exception has occurred
    */
   public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
       DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
           .jobDefinition(jobDefinition)
           .build();

       CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }
```

- Para obter detalhes da API, consulte DeregisterJobDefinition em Referência AWS SDK for Java 2.x da API.

`DescribeComputeEnvironments`

O código de exemplo a seguir mostra como usar `DescribeComputeEnvironments`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Checks the status of the specified compute environment.
    *
    * @param computeEnvironmentName the name of the compute environment to check
    * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
    */
   public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
           .computeEnvironments(computeEnvironmentName)
           .build();

       CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
           }
       });

       return response.thenApply(resp -> resp.computeEnvironments().stream()
           .map(env -> env.statusAsString())
```

```
            .findFirst()
            .orElse("UNKNOWN"));
    }
```

- Para obter detalhes da API, consulte DescribeComputeEnvironments em Referência AWS SDK for Java 2.x da API.

`DescribeJobQueues`

O código de exemplo a seguir mostra como usar `DescribeJobQueues`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the job queue associated with the specified compute
environment.
    *
    * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
               String jobQueueARN;
```

```
                for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                    for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                        String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                        String name = getComputeEnvironmentName(computeEnvironment);
                        if (name.equals(computeEnvironmentName)) {
                            jobQueueARN = jobQueueDetail.jobQueueArn();
                            logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                        }
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }
```

- Para obter detalhes da API, consulte DescribeJobQueues em Referência AWS SDK for Java 2.x da API.

`DescribeJobs`

O código de exemplo a seguir mostra como usar `DescribeJobs`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }
```

- Para obter detalhes da API, consulte DescribeJobs em Referência AWS SDK for Java 2.x da API.

`ListJobsPaginator`

O código de exemplo a seguir mostra como usar `ListJobsPaginator`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists the jobs in the specified job queue with the given job
status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
    }
```

- Para obter detalhes da API, consulte ListJobsPaginator em Referência AWS SDK for Java 2.x da API.

`RegisterJobDefinition`

O código de exemplo a seguir mostra como usar `RegisterJobDefinition`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Registers a new job definition asynchronously in AWS Batch.
    * <p>
    * When using Fargate as the compute environment, it is crucial to set the
    * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
    * ensure proper networking configuration for the Fargate tasks. This
    * allows the tasks to communicate with external services, access the
    * internet, or communicate within a VPC.
    *
    * @param jobDefinitionName the name of the job definition to be registered
    * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
    *                         that provides permissions for the containers in the
job
    * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
    * @return a CompletableFuture that completes with the ARN of the registered
    *         job definition upon successful execution, or completes exceptionally
with
    *         an error if the registration fails
    */
   public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
       NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
               .assignPublicIp(AssignPublicIp.ENABLED)
               .build();

       ContainerProperties containerProperties = ContainerProperties.builder()
               .image(image)
               .executionRoleArn(executionRoleARN)
               .resourceRequirements(
                       Arrays.asList(
```

```
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }
```

- Para obter detalhes da API, consulte RegisterJobDefinition em Referência AWS SDK for Java 2.x da API.

SubmitJob

O código de exemplo a seguir mostra como usar `SubmitJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Submits a job asynchronously to the AWS Batch service.
    *
    * @param jobDefinitionName the name of the job definition to use
    * @param jobQueueName the name of the job queue to submit the job to
    * @param jobARN the Amazon Resource Name (ARN) of the job definition
    * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
    */
   public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
       SubmitJobRequest jobRequest = SubmitJobRequest.builder()
           .jobDefinition(jobARN)
           .jobName(jobDefinitionName)
           .jobQueue(jobQueueName)
           .build();

       CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture.thenApply(SubmitJobResponse::jobId);
   }
```

- Para obter detalhes da API, consulte SubmitJob em Referência AWS SDK for Java 2.x da API.

`UpdateComputeEnvironment`

O código de exemplo a seguir mostra como usar `UpdateComputeEnvironment`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Disables the specified compute environment asynchronously.
    *
    * @param computeEnvironmentName the name of the compute environment to disable
    * @return a CompletableFuture that completes when the compute environment is
disabled
    */
   public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
       UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .state(CEState.DISABLED)
           .build();

       CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }
```

- Para obter detalhes da API, consulte UpdateComputeEnvironment em Referência AWS SDK for Java 2.x da API.

`UpdateJobQueue`

O código de exemplo a seguir mostra como usar `UpdateJobQueue`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Disables the specified job queue asynchronously.
    *
    * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
    * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
    *         or completes exceptionally if an error occurs during the operation
    */
   public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
       UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .state(JQState.DISABLED)
           .build();

       CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
       return responseFuture.whenComplete((updateResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(updateResponse -> null);
   }
```

- Para obter detalhes da API, consulte UpdateJobQueue em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Bedrock usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetFoundationModel`

O código de exemplo a seguir mostra como usar `GetFoundationModel`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha detalhes de um modelo de base usando o cliente síncrono Amazon Bedrock.

```
/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @param bedrockClient   The service client for accessing Amazon Bedrock.
 * @param modelIdentifier The model identifier.
```

```
     * @return An object containing the foundation model's details.
     */
    public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
        try {
            GetFoundationModelResponse response = bedrockClient.getFoundationModel(
                    r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = response.modelDetails();

            System.out.println(" Model ID:                     " + model.modelId());
            System.out.println(" Model ARN:                    " +
model.modelArn());
            System.out.println(" Model Name:                   " +
model.modelName());
            System.out.println(" Provider Name:                " +
model.providerName());
            System.out.println(" Lifecycle status:             " +
model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:             " +
model.inputModalities());
            System.out.println(" Output modalities:            " +
model.outputModalities());
            System.out.println(" Supported customizations:     " +
model.customizationsSupported());
            System.out.println(" Supported inference types:    " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ValidationException e) {
            throw new IllegalArgumentException(e.getMessage());
        } catch (SdkException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

Obtenha detalhes de um modelo de base usando o cliente assíncrono Amazon Bedrock.

```
    /**
     * Get details about an Amazon Bedrock foundation model.
     *
     * @param bedrockClient   The async service client for accessing Amazon Bedrock.
     * @param modelIdentifier The model identifier.
     * @return An object containing the foundation model's details.
     */
    public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
        try {
            CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
                    r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = future.get().modelDetails();

            System.out.println(" Model ID:                     " + model.modelId());
            System.out.println(" Model ARN:                    " +
model.modelArn());
            System.out.println(" Model Name:                   " +
model.modelName());
            System.out.println(" Provider Name:                " +
model.providerName());
            System.out.println(" Lifecycle status:             " +
model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:             " +
model.inputModalities());
            System.out.println(" Output modalities:            " +
model.outputModalities());
            System.out.println(" Supported customizations:     " +
model.customizationsSupported());
            System.out.println(" Supported inference types:    " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ExecutionException e) {
            if (e.getMessage().contains("ValidationException")) {
                throw new IllegalArgumentException(e.getMessage());
            } else {
```

```
                System.err.println(e.getMessage());
                throw new RuntimeException(e);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte GetFoundationModel em Referência AWS SDK for Java 2.x da API.

`ListFoundationModels`

O código de exemplo a seguir mostra como usar `ListFoundationModels`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os modelos de base do Amazon Bedrock disponíveis usando o cliente de base do Amazon Bedrock síncrono.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
     * @param bedrockClient The service client for accessing Amazon Bedrock.
     * @return A list of objects containing the foundation models' details
     */
    public static List<FoundationModelSummary> listFoundationModels(BedrockClient
bedrockClient) {

        try {
            ListFoundationModelsResponse response =
bedrockClient.listFoundationModels(r -> {});
```

```
            List<FoundationModelSummary> models = response.modelSummaries();

            if (models.isEmpty()) {
                System.out.println("No available foundation models in " +
region.toString());
            } else {
                for (FoundationModelSummary model : models) {
                    System.out.println("Model ID: " + model.modelId());
                    System.out.println("Provider: " + model.providerName());
                    System.out.println("Name:     " + model.modelName());
                    System.out.println();
                }
            }

            return models;

        } catch (SdkClientException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

Liste os modelos de base do Amazon Bedrock disponíveis usando o cliente de base do Amazon Bedrock assíncrono.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
     * @param bedrockClient The async service client for accessing Amazon Bedrock.
     * @return A list of objects containing the foundation models' details
     */
    public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
        try {
            CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

            List<FoundationModelSummary> models = future.get().modelSummaries();

            if (models.isEmpty()) {
```

```
                System.out.println("No available foundation models in " +
region.toString());
            } else {
                for (FoundationModelSummary model : models) {
                    System.out.println("Model ID: " + model.modelId());
                    System.out.println("Provider: " + model.providerName());
                    System.out.println("Name:     " + model.modelName());
                    System.out.println();
                }
            }

            return models;

        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        } catch (ExecutionException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte ListFoundationModels em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Bedrock Runtime usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon Bedrock Runtime.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários
- Amazônia Nova

- Amazon Nova Canvas
- Gerador de Imagens do Amazon Titan
- Incorporações de texto Amazon Titan
- Claude da Anthropic
- Command da Cohere
- Llama da Meta
- Mistral AI
- Stable Image Core

Cenários

Criar um aplicativo playground para interagir com os modelos de base do Amazon Bedrock

O exemplo de código a seguir mostra como criar playgrounds para interagir com os modelos de base do Amazon Bedrock por meio de diferentes modalidades.

SDK para Java 2.x

O Java Foundation Model (FM) Playground é um aplicativo de amostra da Spring Boot que mostra como usar o Amazon Bedrock com Java. Este exemplo mostra como os desenvolvedores de Java podem usar o Amazon Bedrock para criar aplicativos generativos AI-enabled . É possível testar e interagir com os modelos de base do Amazon Bedrock usando os três playgrounds a seguir:

- Um playground de texto.
- Um playground de chat.
- Um playground de imagens.

O exemplo também lista e exibe os modelos de base aos quais você tem acesso e respectivas características. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock Runtime

Gerar vídeos com base em prompts de texto usando o Amazon Bedrock

O exemplo de código a seguir mostra como criar um aplicativo Spring Boot que gera vídeos a partir de solicitações de texto usando o Amazon Bedrock e o Nova-Reel modelo.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere vídeos a partir de solicitações de texto usando o Amazon Bedrock e. Nova-Reel

```
import org.springframework.stereotype.Service;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

@Service
public class VideoGenerationService {

    public GenerateVideoResponse generateVideo(String prompt) {

        // add S3 bucket you want to store your generated videos
        String s3Bucket = "s3://mygeneratedvidoenovatest";


        //Create json request as an instance of Document class
        Document novaRequest = prepareDocument(prompt);

        // Create request
        StartAsyncInvokeRequest request = StartAsyncInvokeRequest.builder()
                .modelId("amazon.nova-reel-v1:0")
                .modelInput(novaRequest)
                .outputDataConfig(AsyncInvokeOutputDataConfig.builder()

 .s3OutputDataConfig(AsyncInvokeS3OutputDataConfig.builder().s3Uri(s3Bucket).build())
```

```
                        .build())
                .build();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            CompletableFuture<StartAsyncInvokeResponse>
startAsyncInvokeResponseCompletableFuture =
bedrockClient.startAsyncInvoke(request);

            //blocking operation to wait for the AWS API response
            StartAsyncInvokeResponse startAsyncInvokeResponse =
startAsyncInvokeResponseCompletableFuture.get();
            System.out.println("invocation ARN: " +
startAsyncInvokeResponse.invocationArn());

            GenerateVideoResponse response = new GenerateVideoResponse();
            response.setStatus("inProgress");
            response.setExecutionArn(startAsyncInvokeResponse.invocationArn());

            return response;
        } catch (Exception e) {
            System.out.println(e);
            throw new RuntimeException(e);
        }

    }

    public GenerateVideoResponse checkGenerationStatus(String invocationArn) {
        GenerateVideoResponse response = new GenerateVideoResponse();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            //creating async request to fetch status by invocation Arn
            GetAsyncInvokeRequest asyncRequest =
GetAsyncInvokeRequest.builder().invocationArn(invocationArn).build();

            CompletableFuture<GetAsyncInvokeResponse> asyncInvoke =
bedrockClient.getAsyncInvoke(asyncRequest);

            //blocking operation to wait for the AWS API response
            GetAsyncInvokeResponse asyncInvokeResponse = asyncInvoke.get();
            System.out.println("Invocation status =" +
asyncInvokeResponse.statusAsString());
```

```
            response.setExecutionArn(invocationArn);
            response.setStatus(asyncInvokeResponse.statusAsString());
            return response;
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }

    }

    private static BedrockRuntimeAsyncClient getBedrockRuntimeAsyncClient() {
        BedrockRuntimeAsyncClient bedrockClient =
 BedrockRuntimeAsyncClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
        return bedrockClient;
    }

    private static Document prepareDocument(String prompt) {
        Document textToVideoParams = Document.mapBuilder()
                .putString("text", prompt)
                .build();

        Document videoGenerationConfig = Document.mapBuilder()
                .putNumber("durationSeconds", 6)
                .putNumber("fps", 24)
                .putString("dimension", "1280x720")
                .build();

        Document novaRequest = Document.mapBuilder()
                .putString("taskType", "TEXT_VIDEO")
                .putDocument("textToVideoParams", textToVideoParams)
                .putDocument("videoGenerationConfig", videoGenerationConfig)
                .build();
        return novaRequest;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetAsyncInvoke

- StartAsyncInvoke

Usar ferramentas com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;
```

```
    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?
```

```
                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
```

```
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
```

```
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
```

```
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);
```

```
            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
```

```
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
        System.out.println("Calling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and return
the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
```

```
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

A ferramenta de meteorologia usada pela demonstração. Esse arquivo define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
```

```
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
```

```
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
 string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
```

```
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

A ação da API Converse com uma configuração de ferramenta.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
```

```
    * @param conversation a list of messages representing the conversation history
    * @param toolSpec     the specification of the tool to be used in the converse
request
    * @return the converse response received from the AI model
    */
   public ConverseResponse sendConverseRequestAsync(String modelId, String
systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
       List<Tool> toolList = new ArrayList<>();
       Tool tool = Tool.builder()
           .toolSpec(toolSpec)
           .build();

       toolList.add(tool);

       ToolConfiguration configuration = ToolConfiguration.builder()
           .tools(toolList)
           .build();

       SystemContentBlock block = SystemContentBlock.builder()
           .text(systemPrompt)
           .build();

       ConverseRequest request = ConverseRequest.builder()
           .modelId(modelId)
           .system(block)
           .messages(conversation)
           .toolConfig(configuration)
           .build();

       try {
           ConverseResponse response = getClient().converse(request).join();
           return response;

       } catch (ModelNotReadyException ex) {
           throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
       } catch (BedrockRuntimeException ex) {
           throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
       }
   }
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

Amazônia Nova

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock com o cliente Java assíncrono.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with an asynchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class ConverseAsync {

    public static String converseAsync() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
```

```
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
```

```
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request asynchronously
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request);
            return asyncResponse.thenApply(
                    response -> response.output().message().content().get(0).text()
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converseAsync();
        System.out.println(response);
    }
}
```

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with a synchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
```

```
 * - Configure and send a request
 * - Process the response
 */
public class Converse {

    public static String converse() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
```

```
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            ConverseResponse response = client.converse(request);
            return response.output().message().content().get(0).text();

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converse();
        System.out.println(response);
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processar o fluxo de respostas em tempo real.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;

/**
 * This example demonstrates how to use the Amazon Nova foundation models with an
 * asynchronous Amazon Bedrock runtime client to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ConverseStream {

    public static void converseStream() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
```

```
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 paragraph";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Set up the stream handler
        // The stream handler processes chunks of the response as they arrive
        // - onContentBlockDelta: Processes each text chunk
```

```
        // - onError: Handles any errors during streaming
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            System.out.print(chunk.delta().text());
                            System.out.flush();  // Ensure immediate output of each
 chunk
                        }).build())
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }

    public static void main(String[] args) {
        converseStream();
    }
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK for Java 2.x da API.

Cenário: uso de ferramentas com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A execução primária do fluxo do cenário. Esse cenário orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
```

```
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);
```

```
        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);
```

```
            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
```

```
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
```

```
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
```

```
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
        System.out.println("Calling Bedrock...");
```

```
        // Send the conversation, system prompt, and tool configuration, and return
the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    public static void printFooter() {
```

```
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

A ferramenta de meteorologia usada pela demonstração. Esse arquivo define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
```

```
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
```

```
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
```

```
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

A ação da API Converse com uma configuração de ferramenta.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
     * @param conversation a list of messages representing the conversation history
     * @param toolSpec     the specification of the tool to be used in the converse
 request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();
```

```
        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
        }
    }
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

Amazon Nova Canvas

InvokeModel

O exemplo de código a seguir mostra como invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.InvokeModelResponse;

import java.security.SecureRandom;
import java.util.Base64;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 */
public class InvokeModel {

    public static byte[] invokeModel() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
```

```
                .build();

        // Step 2: Specify which model to use
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-canvas-v1:0";

        // Step 3: Configure the generation parameters and create the request
        // First, set the main parameters:
        // - prompt: Text description of the image to generate
        // - seed: Random number for reproducible generation (0 to 858,993,459)
        String prompt = "A stylized picture of a cute old steampunk robot";
        int seed = new SecureRandom().nextInt(858_993_460);

        // Then, create the request using a template with the following structure:
        // - taskType: TEXT_IMAGE (specifies text-to-image generation)
        // - textToImageParams: Contains the text prompt
        // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
        // For a list of available request parameters, see:
        // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
        String request = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": {
                        "text": "{{prompt}}"
                    },
                    "imageGenerationConfig": {
                        "seed": {{seed}},
                        "quality": "standard"
                    }
                }"""
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", String.valueOf(seed));

        // Step 4: Send and process the request
        // - Send the request to the model using InvokeModelResponse
        // - Extract the Base64-encoded image from the JSON response
        // - Convert the encoded image to a byte array and return it
        try {
            InvokeModelResponse response = client.invokeModel(builder -> builder
                    .modelId(modelId)
```

```
                    .body(SdkBytes.fromUtf8String(request))
            );

            JSONObject responseBody = new
 JSONObject(response.body().asUtf8String());
            // Convert the Base64 string to byte array for better handling
            return Base64.getDecoder().decode(
                    new JSONPointer("/images/0").queryFrom(responseBody).toString()
            );

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s%n", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");
        byte[] imageData = invokeModel();
        displayImage(imageData);
    }
}
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK for Java 2.x da API.

Gerador de Imagens do Amazon Titan

InvokeModel

O exemplo de código a seguir mostra como invocar o Amazon Titan Image no Amazon Bedrock para gerar uma imagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Gerador de Imagens do Amazon Titan.

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Image G2.
        var modelId = "amazon.titan-image-generator-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
```

```
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
        var nativeRequestTemplate = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": { "text": "{{prompt}}" },
                    "imageGenerationConfig": { "seed": {{seed}} }
                }""";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 31-bit seed for the image generation (max. 2,147,483,647).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        var nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
```

```
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }
}
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK for Java 2.x da API.

Incorporações de texto Amazon Titan

InvokeModel

O exemplo de código a seguir mostra como:

- Começar a criar sua primeira incorporação.
- Criar incorporações configurando o número de dimensões e a normalização (somente v2).

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a primeira incorporação com o Incorporador de Texto do Titan v2.

```
// Generate and print an embedding with Amazon Titan Text Embeddings.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
```

```
public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Embeddings V2.
        var modelId = "amazon.titan-embed-text-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-embed-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{inputText}}\" }";

        // The text to convert into an embedding.
        var inputText = "Please recommend books with a theme similar to the movie
 'Inception'.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{inputText}}",
 inputText);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/
embedding").queryFrom(responseBody).toString();
```

```
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

Invoque o Incorporador de Texto do Titan v2 configurando o número de dimensões e a normalização.

```
    /**
     * Invoke Amazon Titan Text Embeddings V2 with additional inference parameters.
     *
     * @param inputText  - The text to convert to an embedding.
     * @param dimensions - The number of dimensions the output embeddings should
 have.
     *                   Values accepted by the model: 256, 512, 1024.
     * @param normalize  - A flag indicating whether or not to normalize the output
 embeddings.
     * @return The {@link JSONObject} representing the model's response.
     */
    public static JSONObject invokeModel(String inputText, int dimensions, boolean
 normalize) {

        // Create a Bedrock Runtime client in the AWS Region of your choice.
        var client = BedrockRuntimeClient.builder()
                .region(Region.US_WEST_2)
                .build();

        // Set the model ID, e.g., Titan Embed Text v2.0.
        var modelId = "amazon.titan-embed-text-v2:0";
```

```
        // Create the request for the model.
        var nativeRequest = """
                {
                    "inputText": "%s",
                    "dimensions": %d,
                    "normalize": %b
                }
                """.formatted(inputText, dimensions, normalize);

        // Encode and send the request.
        var response = client.invokeModel(request -> {
            request.body(SdkBytes.fromUtf8String(nativeRequest));
            request.modelId(modelId);
        });

        // Decode the model's response.
        var modelResponse = new JSONObject(response.body().asUtf8String());

        // Extract and print the generated embedding and the input text token count.
        var embedding = modelResponse.getJSONArray("embedding");
        var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
        System.out.println("Embedding: " + embedding);
        System.out.println("\nInput token count: " + inputTokenCount);

        // Return the model's native response.
        return modelResponse;
    }
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK for Java 2.x da API.

Claude da Anthropic

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
```

```
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().getFirst().text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock com o cliente Java assíncrono.

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
```

```
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().getFirst().text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
```

```
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
 handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK for Java 2.x da API.

Reasoning

O exemplo de código a seguir mostra como usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a capacidade de raciocínio do Anthropic Claude 3.7 Sonnet com o cliente de runtime assíncrono do Bedrock.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with an asynchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock async runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send an asynchronous request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class ReasoningAsync {

    public static ReasoningResponse reasoningAsync() {

        // Create the Amazon Bedrock runtime client
```

```
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );

            // Process the response asynchronously
            return asyncResponse.thenApply(response -> {

                        var content = response.output().message().content();
                        ReasoningContentBlock reasoning = null;
                        String text = null;

                        // Process each content block to find reasoning and response
 text
                        for (ContentBlock block : content) {
```

```
                            if (block.reasoningContent() != null) {
                                reasoning = block.reasoningContent();
                            } else if (block.text() != null) {
                                text = block.text();
                            }
                        }

                        return new ReasoningResponse(reasoning, text);
                    }
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoningAsync();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

Use a capacidade de raciocínio do Anthropic Claude 3.7 Sonnet com o cliente de runtime síncrono do Bedrock.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
```

```
 * with the synchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send a request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class Reasoning {

    public static ReasoningResponse reasoning() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            ConverseResponse bedrockResponse = client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
```

```
            );


            // Extract both reasoning and final response
            var content = bedrockResponse.output().message().content();
            ReasoningContentBlock reasoning = null;
            String text = null;

            // Process each content block to find reasoning and response text
            for (ContentBlock block : content) {
                if (block.reasoningContent() != null) {
                    reasoning = block.reasoningContent();
                } else if (block.text() != null) {
                    text = block.text();
                }
            }

            return new ReasoningResponse(reasoning, text);

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoning();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

Raciocínio com uma resposta em streaming

O exemplo de código a seguir mostra como usar a capacidade de raciocínio do Claude 3.7 Sonnet da Anthropic no Amazon Bedrock.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a capacidade de raciocínio do Anthropic Claude 3.7 Sonnet para gerar respostas de streaming de texto.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;
import java.util.concurrent.atomic.AtomicReference;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 * capability to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ReasoningStream {

    public static ReasoningResponse reasoningStream() {

        // Create the Amazon Bedrock runtime client
```

```
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        // Configure the request with the message, model ID, and reasoning config
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .additionalModelRequestFields(reasoningConfig)
                .messages(message)
                .modelId(modelId)
                .build();

        StringBuilder reasoning = new StringBuilder();
        StringBuilder text = new StringBuilder();
        AtomicReference<ReasoningResponse> finalresponse = new AtomicReference<>();

        // Set up the stream handler to processes chunks of the response as they
 arrive
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            ContentBlockDelta delta = chunk.delta();
                            if (delta.reasoningContent() != null) {
                                if (reasoning.isEmpty()) {
```

```
                                    System.out.println("\n<thinking>");
                                }
                                if (delta.reasoningContent().text() != null) {

System.out.print(delta.reasoningContent().text());

reasoning.append(delta.reasoningContent().text());
                                }
                            } else if (delta.text() != null) {
                                if (text.isEmpty()) {
                                    System.out.println("\n</thinking>\n");
                                }
                                System.out.print(delta.text());
                                text.append(delta.text());
                            }
                            System.out.flush();  // Ensure immediate output of each
chunk
                        }).build())
                .onComplete(() -> finalresponse.set(new ReasoningResponse(
                        ReasoningContentBlock.fromReasoningText(t ->
t.text(reasoning.toString())),
                        text.toString()
                )))
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
            return finalresponse.get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
            throw new RuntimeException(e);
        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        reasoningStream();
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

Command da Cohere

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Cohere Command usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Cohere Command.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock com o cliente Java assíncrono.

```
// Use the Converse API to send a text message to Cohere Command
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
```

```
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
```

```
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK for Java 2.x da API.

Llama da Meta

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Meta Llama.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock com o cliente Java assíncrono.

```
// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
```

```
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
```

```
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK for Java 2.x .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
handler.
            client.converseStream(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK for Java 2.x da API.

Mistral AI

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock.

```
// Use the Converse API to send a text message to Mistral.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }

    }
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock com o cliente Java assíncrono.

```
// Use the Converse API to send a text message to Mistral
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
```

```
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
```

```
        converseAsync();
    }
}
```

- Para obter detalhes da API, consulte Converse na Referência da API do AWS SDK for Java 2.x .

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para a Mistral usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();
```

```
        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStream em Referência AWS SDK for Java 2.x da API.

Stable Image Core

InvokeModel

O exemplo de código a seguir mostra como invocar o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Diffusion.

```
// Create an image with Stability AI Stable Image Core.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;
```

```
public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_WEST_2)
                .build();

        // Set the model ID, e.g., Stable Image Core.
        var modelId = "stability.stable-image-core-v1:1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
diffusion-stable-image-core-text-image-request-response.html
        var nativeRequestTemplate = """
                {
                    "prompt": "{{prompt}}",
                    "aspect_ratio": "1:1",
                    "seed": {{seed}},
                    "output_format": "png"
                }""";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random seed for the image generation (max. 4,294,967,294).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        String nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
```

```
                .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/images/0")
                    .queryFrom(responseBody)
                    .toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }

}
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK for Java 2.x da API.

CloudFront exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with CloudFront.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateDistribution`

O código de exemplo a seguir mostra como usar `CreateDistribution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo a seguir usa um bucket do Amazon Simple Storage Service (Amazon S3) como origem de conteúdo.

Depois de criar a distribuição, o código cria um sinal de espera CloudFrontWaiter até que a distribuição seja implantada antes de retornar a distribuição.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
```

```
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

        private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

        public static Distribution createDistribution(CloudFrontClient
 cloudFrontClient, S3Client s3Client,
                        final String bucketName, final String keyGroupId, final
 String originAccessControlId) {

                final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                                .get("x-amz-bucket-region").get(0);
                final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
                String originId = originDomain; // Use the originDomain value for
 the originId.

                // The service API requires some deprecated methods, such as
                // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
                CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                                .distributionConfig(b1 -> b1
                                                .origins(b2 -> b2
                                                                .quantity(1)
                                                                .items(b3 -> b3

 .domainName(originDomain)

 .id(originId)

 .s3OriginConfig(builder4 -> builder4

            .originAccessIdentity(

                            ""))

 .originAccessControlId(
```

```
            originAccessControlId)))
                                                .defaultCacheBehavior(b2 -> b2

  .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

  .targetOriginId(originId)
                                                                    .minTTL(200L)
                                                                    .forwardedValues(b5
  -> b5

  .cookies(cp -> cp

            .forward(ItemSelection.NONE))

  .queryString(true))
                                                                    .trustedKeyGroups(b3
  -> b3

  .quantity(1)

  .items(keyGroupId)

  .enabled(true))
                                                                    .allowedMethods(b4 -
> b4

  .quantity(2)

  .items(Method.HEAD, Method.GET)

  .cachedMethods(b5 -> b5

            .quantity(2)

            .items(Method.HEAD,

                           Method.GET))))
                                                .cacheBehaviors(b -> b
                                                                    .quantity(1)
                                                                    .items(b2 -> b2

  .pathPattern("/index.html")
```

```
.viewerProtocolPolicy(

           ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

           .quantity(1)

           .items(keyGroupId)

           .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

           .cookies(cp -> cp

                           .forward(ItemSelection.NONE))

           .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

           .items(Method.HEAD,

                           Method.GET)

           .cachedMethods(b6 -> b6

                           .quantity(2)

                           .items(Method.HEAD,

                                           Method.GET)))))
                                               .enabled(true)
                                               .comment("Distribution built with
java")

.callerReference(Instant.now().toString())));
```

```
                final Distribution distribution = createDistResponse.distribution();
                logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
 distribution.domainName(),
                                distribution.id());
                logger.info("Waiting for distribution to be deployed ...");
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                                        .matched();
                        responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                        distribution.id());
                }
                return distribution;
        }
}
```

- Para obter detalhes da API, consulte CreateDistribution em Referência AWS SDK for Java 2.x da API.

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <functionName> <filePath>

                Where:
                    functionName - The name of the function to create.\s
                    filePath - The path to a file that contains the application
 logic for the function.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
```

```
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient, String
 functionName, String filePath) {
        try {
            InputStream fileIs =
 CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                    .comment("Created by using the CloudFront Java API")
                    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .name(functionName)
                    .functionCode(functionCode)
                    .functionConfig(config)
                    .build();

            CreateFunctionResponse response =
 cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateFunction em Referência AWS SDK for Java 2.x da API.

`CreateKeyGroup`

O código de exemplo a seguir mostra como usar `CreateKeyGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Um grupo de chaves exige pelo menos uma chave pública usada para verificar URLs ou cookies assinados.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
 LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b -> b.keyGroupConfig(c
 -> c
                .items(publicKeyId)
                .name("JavaKeyGroup" + UUID.randomUUID()))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- Para obter detalhes da API, consulte CreateKeyGroup em Referência AWS SDK for Java 2.x da API.

`CreatePublicKey`

O código de exemplo a seguir mostra como usar `CreatePublicKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo de código a seguir lê uma chave pública e a carrega na Amazon CloudFront.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
 LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
 publicKeyFileName) {
        try (InputStream is =
 CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                    .createPublicKey(b -> b.publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte CreatePublicKey em Referência AWS SDK for Java 2.x da API.

`DeleteDistribution`

O código de exemplo a seguir mostra como usar `DeleteDistribution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo de código a seguir atualiza uma distribuição para desativada, usa um waiter que aguarda a implantação da alteração e, em seguida, exclui a distribuição.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
        private static final Logger logger =
 LoggerFactory.getLogger(DeleteDistribution.class);

        public static void deleteDistribution(final CloudFrontClient
 cloudFrontClient, final String distributionId) {
                // First, disable the distribution by updating it.
                GetDistributionResponse response =
 cloudFrontClient.getDistribution(b -> b
                                .id(distributionId));
                String etag = response.eTag();
                DistributionConfig distConfig =
 response.distribution().distributionConfig();
```

```
                cloudFrontClient.updateDistribution(builder -> builder
                                .id(distributionId)
                                .distributionConfig(builder1 -> builder1

.cacheBehaviors(distConfig.cacheBehaviors())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .enabled(false)
                                                .origins(distConfig.origins())
                                                .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .priceClass(distConfig.priceClass())
                                                .aliases(distConfig.aliases())
                                                .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                                                .webACLId(distConfig.webACLId())

.originGroups(distConfig.originGroups()))
                                .ifMatch(etag));

                logger.info("Distribution [{}] is DISABLED, waiting for deployment
before deleting ...",
                                distributionId);
                GetDistributionResponse distributionResponse;
                try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
```

```
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distributionId)).matched();
                        distributionResponse = responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Could not disable distribution"));
                }

                DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient
                                .deleteDistribution(builder -> builder
                                                .id(distributionId)

 .ifMatch(distributionResponse.eTag()));
                if (deleteDistributionResponse.sdkHttpResponse().isSuccessful()) {
                        logger.info("Distribution [{}] DELETED", distributionId);
                }
        }
}
```

- Para obter detalhes da API, consulte DeleteDistribution em Referência AWS SDK for Java 2.x da API.

`UpdateDistribution`

O código de exemplo a seguir mostra como usar `UpdateDistribution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
```

```
import software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <id>\s

                Where:
                    id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
 idVal) {
        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                    .id(idVal)
                    .build();
```

```
            GetDistributionResponse response =
 cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();

            // Create a new DistributionConfig object and add new values to comment
 and
            // aliases
            DistributionConfig config1 = DistributionConfig.builder()
                    .aliases(config.aliases()) // You can pass in new values here
                    .comment("New Comment")
                    .cacheBehaviors(config.cacheBehaviors())
                    .priceClass(config.priceClass())
                    .defaultCacheBehavior(config.defaultCacheBehavior())
                    .enabled(config.enabled())
                    .callerReference(config.callerReference())
                    .logging(config.logging())
                    .originGroups(config.originGroups())
                    .origins(config.origins())
                    .restrictions(config.restrictions())
                    .defaultRootObject(config.defaultRootObject())
                    .webACLId(config.webACLId())
                    .httpVersion(config.httpVersion())
                    .viewerCertificate(config.viewerCertificate())
                    .customErrorResponses(config.customErrorResponses())
                    .build();

            UpdateDistributionRequest updateDistributionRequest =
 UpdateDistributionRequest.builder()
                    .distributionConfig(config1)
                    .id(disObject.id())
                    .ifMatch(response.eTag())
                    .build();

            cloudFrontClient.updateDistribution(updateDistributionRequest);

        } catch (CloudFrontException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte UpdateDistribution em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma distribuição multilocatária e um locatário de distribuição

O exemplo a seguir mostra como criar uma distribuição multilocatária e um locatário de distribuição com várias configurações.

SDK para Java 2.x

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo a seguir demonstra como criar uma distribuição multilocatária com parâmetros e certificado curinga.

```
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.ConnectionMode;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.HttpVersion;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.SSLSupportMethod;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateMultiTenantDistribution {
    public static Distribution
 CreateMultiTenantDistributionWithCert(CloudFrontClient cloudFrontClient,
```

```
                                                                        S3Client
s3Client,
                                                                        final String
bucketName,
                                                                        final String
certificateArn) {
       // fetch the origin info if necessary
       final String region = s3Client.headBucket(b ->
b.bucket(bucketName)).sdkHttpResponse().headers()
               .get("x-amz-bucket-region").get(0);
       final String originDomain = bucketName + ".s3." + region + ".amazonaws.com";
       String originId = originDomain; // Use the originDomain value for the
originId.

       CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
               .distributionConfig(b1 -> b1
                       .httpVersion(HttpVersion.HTTP2)
                       .enabled(true)
                       .comment("Template Distribution with cert built with java")
                       .connectionMode(ConnectionMode.TENANT_ONLY)
                       .callerReference(Instant.now().toString())
                       .viewerCertificate(certBuilder -> certBuilder
                               .acmCertificateArn(certificateArn)
                               .sslSupportMethod(SSLSupportMethod.SNI_ONLY))
                       .origins(b2 -> b2
                               .quantity(1)
                               .items(b3 -> b3
                                       .domainName(originDomain)
                                       .id(originId)
                                       .originPath("/{{tenantName}}")
                                       .s3OriginConfig(builder4 -> builder4
                                               .originAccessIdentity(
                                                       "")))
                       .tenantConfig(b5 -> b5
                               .parameterDefinitions(b6 -> b6
                                       .name("tenantName")
                                       .definition(b7 -> b7
                                               .stringSchema(b8 -> b8
                                                       .comment("tenantName value")
                                                       .defaultValue("root")
                                                       .required(false)))))
                       .defaultCacheBehavior(b2 -> b2
```

```
 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
 cfWaiter
                    .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
 created"));
        }
        return distribution;
    }

    public static Distribution CreateMultiTenantDistributionNoCert(CloudFrontClient
 cloudFrontClient,
                                                                   S3Client s3Client,
                                                                   final String
 bucketName) {
        // fetch the origin info if necessary
        final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region + ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the
 originId.

        CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with java")
```

```
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        "")))))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
 cfWaiter
                    .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
 created"));
        }
        return distribution;
    }
```

```
}
```

O exemplo a seguir demonstra como criar um locatário de distribuição associado a esse modelo, incluindo a utilização do parâmetro que declaramos acima. Observe que não precisamos adicionar informações do certificado aqui porque nosso domínio já está coberto pelo modelo principal.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant createDistributionTenantNoCert(CloudFrontClient
 cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("no-cert-tenant")
```

```
        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection group
 that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

Se o certificado do visualizador fosse omitido do modelo principal, você precisaria adicionar informações do certificado sobre os inquilinos associados a ele. O exemplo a seguir demonstra como fazer isso por meio de um certificado ACM arn que cobre o domínio necessário para o inquilino.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
```

```
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantWithCert(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId,
                                                                    String
 certificateArn) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .enabled(false)
                .name("tenant-with-cert")
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .customizations(b3 -> b3
                        .certificate(b4 -> b4
                                .arn(certificateArn))) // NOTE: Cert must be in Us-
East-1 and cover the domain provided in this request

        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();
```

```
        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection group
 that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

O exemplo a seguir demonstra como fazer isso com uma solicitação de certificado CloudFront-hosted gerenciado. Isso é ideal se você ainda não tiver tráfego para o seu domínio. Nesse caso, criamos um ConnectionGroup para gerar um RoutingEndpoint. Em seguida, usamos isso RoutingEndpoint para criar registros DNS que verificam a propriedade do domínio e apontam para CloudFront. CloudFront em seguida, fornecerá automaticamente um token para validar a propriedade do domínio e criar um certificado gerenciado.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
```

```
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantCfHosted(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId) throws InterruptedException {
        CreateConnectionGroupResponse createConnectionGroupResponse =
 cloudFrontClient.createConnectionGroup(builder -> builder
                .ipv6Enabled(true)
                .name("cf-hosted-connection-group")
                .enabled(true));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("cf-hosted domain validation record")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type(RRType.CNAME)
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(createConnectionGroupResponse.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));

        // Give the R53 record time to propagate, if it isn't being returned by
 servers yet, the following call will fail
        Thread.sleep(60000);
```

```
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))

 .connectionGroupId(createConnectionGroupResponse.connectionGroup().id())
                .enabled(false)
                .name("cf-hosted-tenant")
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.CLOUDFRONT)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

O exemplo a seguir demonstra como fazer isso com uma solicitação de certificado gerenciado auto-hospedado. Isso é ideal se você tiver tráfego para o seu domínio e não puder tolerar o tempo de inatividade durante uma migração. No final deste exemplo, o inquilino será criado em um estado aguardando a validação do domínio e a configuração do DNS. Siga as etapas [aqui](https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/managed-cloudfront-certificates.html#complete -domain-ownership) para concluir a configuração quando estiver pronto para migrar o tráfego.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;
```

```
import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantSelfHosted(CloudFrontClient cloudFrontClient,
                                                                        String
 distributionId,
                                                                        String
 domain) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("self-hosted-tenant")
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.SELF_HOSTED)
                        .primaryDomainName(domain)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDistribution
  - CreateDistributionTenant

Excluir recursos de assinatura

O exemplo de código a seguir mostra como excluir recursos que são usados para obter acesso a conteúdo restrito em um bucket do Amazon Simple Storage Service (Amazon S3).

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient,
            final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
                .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
 String keyGroupId) {
```

```
        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
 cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
 final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteKeyGroup
  - DeleteOriginAccessControl
  - DeletePublicKey

Assinar URLs e cookies

O exemplo de código a seguir mostra como criar URLs e cookies assinados que permitem acesso a recursos restritos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a CannedSignerRequest classe para assinar URLs ou cookies com uma política predefinida.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
    }
}
```

Use a CustomSignerRequest classe para assinar URLs ou cookies com uma política personalizada. O `activeDate` e `ipRange` são métodos opcionais.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                // .resourceUrlPattern("https://*.example.com/*")  // Optional.
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

O exemplo a seguir demonstra o uso da CloudFrontUtilities classe para produzir cookies e URLs assinados. Veja este exemplo de código em GitHub.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
 getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }
```

```
    public static CookiesForCustomPolicy
 getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- Para obter detalhes da API, consulte CloudFrontUtilities em Referência AWS SDK for Java 2.x da API.

CloudWatch exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with CloudWatch.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos

- Ações
- Cenários

Conceitos básicos

Olá CloudWatch

O exemplo de código a seguir mostra como começar a usar o CloudWatch.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> System.out.println(" Retrieved metric is: "
 + metrics.metricName()));

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListMetrics em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Liste CloudWatch namespaces e métricas.
- Obter estatísticas para uma métrica e para faturamento estimado.
- Criar e atualizar um painel.
- Criar e adicionar dados a uma métrica.
- Criar e acionar um alarme e, em seguida, visualizar o histórico de alarmes.
- Criar um detector de anomalias.
- Obter uma imagem de métrica e, em seguida, limpar os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os CloudWatch recursos.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardInvalidInputErrorException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorResponse;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.LimitExceededException;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataResponse;
import java.io.IOException;
```

```
import java.util.ArrayList;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    static CloudWatchActions cwActions = new CloudWatchActions();
```

```
   private static final Logger logger =
LoggerFactory.getLogger(CloudWatchScenario.class);
   static Scanner scanner = new Scanner(System.in);
   public static void main(String[] args) throws Throwable {

       final String usage = """

           Usage:
             <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
<settings> <metricImage> \s

           Where:
             myDate - The start date to use to get metric statistics. (For example,
2023-01-11T18:35:24.00Z.)\s
             costDateWeek - The start date to use to get AWS/Billing statistics.
(For example, 2023-01-11T18:35:24.00Z.)\s
             dashboardName - The name of the dashboard to create.\s
             dashboardJson - The location of a JSON file to use to create a
dashboard. (See jsonWidgets.json in javav2/example_code/cloudwatch.)\s
             dashboardAdd - The location of a JSON file to use to update a
dashboard. (See CloudDashboard.json in javav2/example_code/cloudwatch.)\s
             settings - The location of a JSON file from which various values are
read. (See settings.json in javav2/example_code/cloudwatch.)\s
             metricImage - The location of a BMP file that is used to create a
graph.\s
           """;

       if (args.length != 7) {
           logger.info(usage);
           return;
       }
       String myDate = args[0];
       String costDateWeek = args[1];
       String dashboardName = args[2];
       String dashboardJson = args[3];
       String dashboardAdd = args[4];
       String settings = args[5];
       String metricImage = args[6];

       logger.info(DASHES);
       logger.info("Welcome to the Amazon CloudWatch Basics scenario.");
       logger.info("""
           Amazon CloudWatch is a comprehensive monitoring and observability
service
```

```
            provided by Amazon Web Services (AWS). It is designed to help you
monitor your
            AWS resources, applications, and services, as well as on-premises
resources,
            in real-time.

            CloudWatch collects and tracks various types of data, including
metrics,
            logs, and events, from your AWS and on-premises resources. It allows you
to set
            alarms and automatically respond to changes in your environment,
            enabling you to quickly identify and address issues before they impact
your
            applications or services.

            With CloudWatch, you can gain visibility into your entire
infrastructure, from the cloud
            to the edge, and use this information to make informed decisions and
optimize your
            resource utilization.

            This scenario guides you through how to perform Amazon CloudWatch tasks
by using the
            AWS SDK for Java v2. Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
            runScenario(myDate, costDateWeek, dashboardName, dashboardJson,
dashboardAdd, settings, metricImage);
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
        logger.info(DASHES);
    }

    private static void runScenario(String myDate, String costDateWeek, String
dashboardName, String dashboardJson, String dashboardAdd, String settings, String
metricImage ) throws Throwable {
        Double dataPoint = Double.parseDouble("10.0");
        logger.info(DASHES);
        logger.info("""
        1. List at least five available unique namespaces from Amazon CloudWatch.
        Select one from the list.
```

```
        """);
        String selectedNamespace;
        String selectedMetrics;
        int num;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listNameSpacesAsync();
            ArrayList<String> list = future.join();
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". {}", list.get(z));
            }

            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedNamespace = list.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedNamespace);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List available metrics within the selected namespace.");
        logger.info("""
            A metric is a measure of the performance or health of your AWS
resources,
            applications, or custom resources. Metrics are the basic building blocks
of CloudWatch
            and provide data points that represent a specific aspect of your system
or application over time.
```

```
            Select a metric from the list.
            """);

        Dimension myDimension = null;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listMetsAsync(selectedNamespace);
            ArrayList<String> metList = future.join();
            logger.info("Metrics successfully retrieved. Total metrics: {}",
metList.size());
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". " + metList.get(z));
            }
            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedMetrics = metList.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedMetrics);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            myDimension = cwActions.getSpecificMetAsync(selectedNamespace).join();
            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Get statistics for the selected metric over the last day.");
        logger.info("""
            Statistics refer to the various mathematical calculations that can be
performed on the
            collected metrics to derive meaningful insights. Statistics provide a
way to summarize and
            analyze the data collected for a specific metric over a specified time
period.
            """);
        waitForInputToContinue(scanner);
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            logger.info("    " + (t + 1) + ". {}", statTypes.get(t));
        }
        logger.info("Select a metric statistic by entering a number from the
preceding list:");
        num = Integer.parseInt(scanner.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            logger.info("You did not select a valid option.");
            return;
        }
        logger.info("You selected " + metricOption);
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getAndDisplayMetricStatisticsAsync(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
            future.join();
            logger.info("Metric statistics retrieved successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Get CloudWatch estimated billing for the last week.");
        waitForInputToContinue(scanner);
         try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getMetricStatisticsAsync(costDateWeek);
            future.join();

            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
             throw cause;
         }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Create a new CloudWatch dashboard with metrics.");
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.createDashboardWithMetricsAsync(dashboardName, dashboardJson);
            future.join();

        } catch (RuntimeException | IOException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List dashboards using a paginator.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.listDashboardsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create a new custom metric by adding data to it.");
        logger.info("""
            The primary benefit of using a custom metric in Amazon CloudWatch is the
ability to
```

```
            monitor and collect data that is specific to your application or
infrastructure.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.createNewCustomMetricAsync(dataPoint);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Add an additional metric to the dashboard.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.addMetricToDashboardAsync(dashboardAdd, dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Create an alarm for the custom metric.");
```

```
        waitForInputToContinue(scanner);
        String alarmName = "" ;
        try {
            CompletableFuture<String> future = cwActions.createAlarmAsync(settings);
            alarmName = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof LimitExceededException cwEx) {
                logger.info("The quota for alarms has been reached: Error message:
{}, Error code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Describe ten current alarms.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.describeAlarmsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Get current data for new custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.getCustomMetricDataAsync(settings);
```

```
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Push data into the custom metric to trigger the alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.addMetricDataForAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.checkForMetricAlarmAsync(settings);
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Get alarm history for the new alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.getAlarmHistoryAsync(settings, myDate);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Add an anomaly detector for the custom metric.");
        logger.info("""
            An anomaly detector is a feature that automatically detects unusual
patterns or deviations in your
            monitored metrics. It uses machine learning algorithms to analyze the
historical behavior
            of your metrics and establish a baseline.
```

```
            The anomaly detector then compares the current metric values against
this baseline and
            identifies any anomalies or outliers that may indicate potential issues
or unexpected changes
            in your system's performance or behavior.

            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.addAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Describe current anomaly detectors.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.describeAnomalyDetectorsAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Get a metric image for the custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.downloadAndSaveMetricImageAsync(metricImage);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("18. Clean up the Amazon CloudWatch resources.");

        try {
            logger.info(". Delete the Dashboard.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteDashboardsResponse> future =
cwActions.deleteDashboardAsync(dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
```

```
            logger.info("Delete the alarm.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAlarmsResponse> future =
cwActions.deleteCWAlarmAsync(alarmName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the anomaly detector.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAnomalyDetectorResponse> future =
cwActions.deleteAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("The Amazon CloudWatch example scenario is complete.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
```

```
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de invólucro para métodos CloudWatch SDK.

```
public class CloudWatchActions {

    private static CloudWatchAsyncClient cloudWatchAsyncClient;

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchActions.class);

    /**
     * Retrieves an asynchronous CloudWatch client instance.
     *
     * <p>
     * This method ensures that the CloudWatch client is initialized with the
 following configurations:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
     * </p>
     *
     * @return the asynchronous CloudWatch client instance
```

```
     */
    private static CloudWatchAsyncClient getAsyncClient() {
        if (cloudWatchAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            cloudWatchAsyncClient = CloudWatchAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return cloudWatchAsyncClient;
    }

    /**
     * Deletes an Anomaly Detector.
     *
     * @param fileName the name of the file containing the Anomaly Detector
configuration
     * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
     */
    public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
```

```
        });

        return readFileFuture.thenCompose(rootNode -> {
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }

    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
```

```
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }

    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }


    /**
     * Retrieves and saves a custom metric image to a file.
     *
```

```
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
       logger.info("Getting Image data for custom metric.");
       String myJSON = """
             {
                 "title": "Example Metric Graph",
                 "view": "timeSeries",
                 "stacked ": false,
                 "period": 10,
                 "width": 1400,
                 "height": 600,
                 "metrics": [
                     [
                     "AWS/Billing",
                     "EstimatedCharges",
                     "Currency",
                     "USD"
                    ]
                 ]
             }
           """;

       GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
           .metricWidget(myJSON)
           .build();

       return getAsyncClient().getMetricWidgetImage(imageRequest)
           .thenCompose(response -> {
               SdkBytes sdkBytes = response.metricWidgetImage();
               byte[] bytes = sdkBytes.asByteArray();
               return CompletableFuture.runAsync(() -> {
                   try {
                       File outputFile = new File(fileName);
                       try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                           outputStream.write(bytes);
                       }
                   } catch (IOException e) {
```

```
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }


    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
     * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
     * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
     *                          or when describing the anomaly detectors
     */
    public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser);
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }


    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
     */
    public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
```

```
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

                PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
```

```
        });
    }


    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
```

```
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                        logger.info("History summary: {}", item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }



    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
```

```
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified metric.
```

```
    *
    * @param metricRequest    the request to describe the alarms for the specified
metric
    * @param customMetricName the name of the custom metric to check for an alarm
    * @param retries          the number of retries to perform if no alarm is found
    * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
    */
   private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
       if (retries == 0) {
           return CompletableFuture.completedFuture(null).thenRun(() ->
               logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
           );
       }

       return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
           if (response.hasMetricAlarms()) {
               logger.info("Alarm state found for {}", customMetricName);
               return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
           } else {
               return CompletableFuture.runAsync(() -> {
                   try {
                       Thread.sleep(20000);
                       logger.info(".");
                   } catch (InterruptedException e) {
                       throw new RuntimeException("Interrupted while waiting to
retry", e);
                   }
               }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
           }
       }));
   }


   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
```

```
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
               Instant instant = Instant.now();

               // Create MetricDatum objects.
               MetricDatum datum1 = MetricDatum.builder()
                   .metricName(customMetricName)
                   .unit(StandardUnit.NONE)
                   .value(1001.00)
                   .timestamp(instant)
                   .build();

               MetricDatum datum2 = MetricDatum.builder()
                   .metricName(customMetricName)
                   .unit(StandardUnit.NONE)
                   .value(1002.00)
                   .timestamp(instant)
                   .build();
```

```
                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }


    /**
     * Retrieves custom metric data from the AWS CloudWatch service.
     *
     * @param fileName the name of the file containing the custom metric information
     * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
     */
    public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                // Read values from the JSON file.
```

```
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();
```

```
                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }


    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
```

```
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }

    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }

        // Extract values from the JSON node.
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName = rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
```

```
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }

    /**
     * Adds a metric to a dashboard asynchronously.
     *
     * @param fileName      the name of the file containing the dashboard content
     * @param dashboardName the name of the dashboard to be updated
```

```
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with a
    * {@link PutDashboardResponse} when the dashboard is successfully updated
    */
   public CompletableFuture<PutDashboardResponse> addMetricToDashboardAsync(String
fileName, String dashboardName) {
       String dashboardBody;
       try {
           dashboardBody = readFileAsString(fileName);
       } catch (IOException e) {
           throw new RuntimeException("Failed to read the dashboard file", e);
       }

       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to update dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Error updating dashboard", ex);
               } else {
                   logger.info("{} was successfully updated.", dashboardName);
                   return response;
               }
           });
   }

   /**
    * Creates a new custom metric.
    *
    * @param dataPoint the data point to be added to the custom metric
    * @return a {@link CompletableFuture} representing the asynchronous operation
of adding the custom metric
    */
   public CompletableFuture<PutMetricDataResponse>
createNewCustomMetricAsync(Double dataPoint) {
       Dimension dimension = Dimension.builder()
           .name("UNIQUE_PAGES")
           .value("URLS")
           .build();
```

```
        // Set an Instant object for the current time in UTC.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        // Create the MetricDatum.
        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum)
            .build();

        return getAsyncClient().putMetricData(request)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    throw new RuntimeException("Error adding custom metric", ex);
                } else {
                    logger.info("Successfully added metric values for
PAGES_VISITED.");
                }
            });
    }

    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing the
dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
```

```
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
    }


    /**
     * Creates a new dashboard with the specified name and metrics from the given
file.
     *
     * @param dashboardName the name of the dashboard to be created
     * @param fileName      the name of the file containing the dashboard body
     * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
     * @throws IOException if there is an error reading the dashboard body from the
file
     */
    public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
        String dashboardBody = readFileAsString(fileName);
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(dashboardBody)
            .build();

        return getAsyncClient().putDashboard(dashboardRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create dashboard: {}", ex.getMessage());
                    throw new RuntimeException("Dashboard creation failed", ex);
                } else {
                    // Handle the normal response case
                    logger.info("{} was successfully created.", dashboardName);
                    List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                    if (messages.isEmpty()) {
                        logger.info("There are no messages in the new Dashboard.");
```

```
                    } else {
                        for (DashboardValidationMessage message : messages) {
                            logger.info("Message: {}", message.message());
                        }
                    }
                    return response; // Return the response for further use
                }
            });
    }


    /**
     * Retrieves the metric statistics for the "EstimatedCharges" metric in the
"AWS/Billing" namespace.
     *
     * @param costDateWeek the start date for the metric statistics, in the format
of an ISO-8601 date string (e.g., "2023-04-05")
     * @return a {@link CompletableFuture} that, when completed, contains the {@link
GetMetricStatisticsResponse} with the retrieved metric statistics
     * @throws RuntimeException if the metric statistics cannot be retrieved
successfully
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getMetricStatisticsAsync(String costDateWeek) {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();

        // Define dimension
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
            .statistics(Statistic.MAXIMUM)
            .startTime(start)
            .endTime(endDate)
```

```
            .period(86400) // One day period
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {})",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    throw new RuntimeException("Failed to get metric statistics: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
     * @param metVal       the name of the metric
     * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
     * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
     * @param myDimension  the dimension(s) to filter the metric statistics by
     * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

         String metricOption, String date, Dimension myDimension) {

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
```

```
        // Building the request for metric statistics.
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400) // 1 day period
            .statistics(Statistic.fromValue(metricOption))
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
            });
    }


    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
```

```
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists the available namespaces for the current AWS account.
     *
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of the available namespace names.
     * @throws RuntimeException if an error occurs while listing the namespaces.
     */
    public CompletableFuture<ArrayList<String>> listNameSpacesAsync() {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder().build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String namespace = metric.namespace();
                if (!nameSpaceList.contains(namespace)) {
                    nameSpaceList.add(namespace);
                }
            });
```

```
        });

        return future
            .thenApply(ignored -> nameSpaceList)
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list namespaces: " +
exception.getMessage(), exception);
            });
    }
    /**
     * Retrieves the specific metric asynchronously.
     *
     * @param namespace the namespace of the metric to retrieve
     * @return a CompletableFuture that completes with the first dimension of the
first metric found in the specified namespace,
     * or throws a RuntimeException if an error occurs or no metrics or dimensions
are found
     */
    public CompletableFuture<Dimension> getSpecificMetAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        return getAsyncClient().listMetrics(request).handle((response, exception) ->
{
            if (exception != null) {
                logger.info("Error occurred while listing metrics: {} ",
exception.getMessage());
                throw new RuntimeException("Failed to retrieve specific metric
dimension", exception);
            } else {
                List<Metric> myList = response.metrics();
                if (!myList.isEmpty()) {
                    Metric metric = myList.get(0);
                    if (!metric.dimensions().isEmpty()) {
                        return metric.dimensions().get(0); // Return the first
dimension
                    }
                }
                throw new RuntimeException("No metrics or dimensions found");
            }
        });
    }
```

```
    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

Ações

`DeleteAlarms`

O código de exemplo a seguir mostra como usar `DeleteAlarms`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a CloudWatch alarm.
    *
    * @param alarmName the name of the alarm to be deleted
    * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
    * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
    * or a {@link RuntimeException} is thrown if the operation fails
    */
   public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
       DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
           .alarmNames(alarmName)
           .build();

       return getAsyncClient().deleteAlarms(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
               } else {
                   logger.info("Successfully deleted alarm {} ", alarmName);
               }
           });
   }
```

- Para obter detalhes da API, consulte DeleteAlarms em Referência AWS SDK for Java 2.x da API.

`DeleteAnomalyDetector`

O código de exemplo a seguir mostra como usar `DeleteAnomalyDetector`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes an Anomaly Detector.
    *
    * @param fileName the name of the file containing the Anomaly Detector
configuration
    * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
    */
   public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
               .metricName(customMetricName)
               .namespace(customMetricNamespace)
```

```
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }
```

- Para obter detalhes da API, consulte DeleteAnomalyDetector em Referência AWS SDK for Java 2.x da API.

`DeleteDashboards`

O código de exemplo a seguir mostra como usar `DeleteDashboards`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
```

```
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteDashboards em Referência AWS SDK for Java 2.x da API.

`DescribeAlarmHistory`

O código de exemplo a seguir mostra como usar `DescribeAlarmHistory`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
```

```
    * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
    * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
    */
   public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       // Use the alarm name to describe alarm history with a paginator.
       return readFileFuture.thenCompose(alarmName -> {
           try {
               Instant start = Instant.parse(date);
               Instant endDate = Instant.now();
               DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                   .startDate(start)
                   .endDate(endDate)
                   .alarmName(alarmName)
                   .historyItemType(HistoryItemType.ACTION)
                   .build();

               // Use the paginator to paginate through alarm history pages.
               DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
               CompletableFuture<Void> future = historyPublisher
                   .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                       logger.info("History summary: {}", item.historySummary());
                       logger.info("Timestamp: {}", item.timestamp());
                   }))
                   .whenComplete((result, exception) -> {
```

```
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }
```

- Para obter detalhes da API, consulte DescribeAlarmHistory em Referência AWS SDK for Java 2.x da API.

`DescribeAlarms`

O código de exemplo a seguir mostra como usar `DescribeAlarms`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }
```

- Para obter detalhes da API, consulte DescribeAlarms em Referência AWS SDK for Java 2.x da API.

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Checks for a metric alarm in AWS CloudWatch.
    *
    * @param fileName the name of the file containing the JSON configuration for
the custom metric
    * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
    */
   public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified metric.
     *
     * @param metricRequest    the request to describe the alarms for the specified
metric
     * @param customMetricName the name of the custom metric to check for an alarm
     * @param retries          the number of retries to perform if no alarm is found
     * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
     */
    private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
        if (retries == 0) {
            return CompletableFuture.completedFuture(null).thenRun(() ->
                logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
            );
        }

        return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
            if (response.hasMetricAlarms()) {
```

```
                logger.info("Alarm state found for {}", customMetricName);
                return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
            } else {
                return CompletableFuture.runAsync(() -> {
                    try {
                        Thread.sleep(20000);
                        logger.info(".");
                    } catch (InterruptedException e) {
                        throw new RuntimeException("Interrupted while waiting to
retry", e);
                    }
                }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
            }
        }));
    }
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetric em Referência AWS SDK for Java 2.x da API.

`DescribeAnomalyDetectors`

O código de exemplo a seguir mostra como usar `DescribeAnomalyDetectors`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
```

```
    * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
    * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
    *                          or when describing the anomaly detectors
    */
   public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser);
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                   .maxResults(10)
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

               return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                   List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                   for (AnomalyDetector detector : anomalyDetectorList) {
                       logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                       logger.info("State: {} ", detector.stateValue());
                   }
               });
           } catch (RuntimeException e) {
```

```
                throw new RuntimeException("Failed to describe anomaly detectors",
 e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
 exception);
            }
        });
    }
```

- Para obter detalhes da API, consulte DescribeAnomalyDetectors em Referência AWS SDK for Java 2.x da API.

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DisableAlarmActions {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DisableAlarmActions em Referência AWS SDK for Java 2.x da API.

`EnableAlarmActions`

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to enable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String alarm = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
                    .alarmNames(alarm)
                    .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte EnableAlarmActions em Referência AWS SDK for Java 2.x da API.

`GetMetricData`

O código de exemplo a seguir mostra como usar `GetMetricData`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves custom metric data from the AWS CloudWatch service.
    *
    * @param fileName the name of the file containing the custom metric information
    * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
    */
   public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               // Read values from the JSON file.
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               // Parse the JSON string to extract relevant values.
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
```

```
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }
```

- Para obter detalhes da API, consulte GetMetricData em Referência AWS SDK for Java 2.x da API.

`GetMetricStatistics`

O código de exemplo a seguir mostra como usar `GetMetricStatistics`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
     * @param metVal       the name of the metric
     * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
     * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
     * @param myDimension  the dimension(s) to filter the metric statistics by
     * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,
```

```
        String metricOption, String date, Dimension myDimension) {

       Instant start = Instant.parse(date);
       Instant endDate = Instant.now();

       // Building the request for metric statistics.
       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .endTime(endDate)
           .startTime(start)
           .dimensions(myDimension)
           .metricName(metVal)
           .namespace(nameSpace)
           .period(86400) // 1 day period
           .statistics(Statistic.fromValue(metricOption))
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                       }
                   } else {
                       logger.info("The returned data list is empty");
                   }
               } else {
                   logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
               }
           })
           .exceptionally(exception -> {
               throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
           });
   }
```

- Para obter detalhes da API, consulte GetMetricStatistics em Referência AWS SDK for Java 2.x da API.

`GetMetricWidgetImage`

O código de exemplo a seguir mostra como usar `GetMetricWidgetImage`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves and saves a custom metric image to a file.
    *
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
       logger.info("Getting Image data for custom metric.");
       String myJSON = """
             {
                 "title": "Example Metric Graph",
                 "view": "timeSeries",
                 "stacked ": false,
                 "period": 10,
                 "width": 1400,
                 "height": 600,
                 "metrics": [
                     [
                     "AWS/Billing",
                     "EstimatedCharges",
                     "Currency",
                     "USD"
                    ]
                 ]
```

```
              }
            """;

        GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

        return getAsyncClient().getMetricWidgetImage(imageRequest)
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }
```

- Para obter detalhes da API, consulte GetMetricWidgetImage em Referência AWS SDK for Java 2.x da API.

`ListDashboards`

O código de exemplo a seguir mostra como usar `ListDashboards`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Lists the available dashboards.
    *
    * @return a {@link CompletableFuture} that completes when the operation is
finished.
    * The future will complete exceptionally if an error occurs while listing the
dashboards.
    */
   public CompletableFuture<Void> listDashboardsAsync() {
       ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
       ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
       return paginator.subscribe(response -> {
           response.dashboardEntries().forEach(entry -> {
               logger.info("Dashboard name is: {} ", entry.dashboardName());
               logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
           });
       }).exceptionally(ex -> {
           logger.info("Failed to list dashboards: {} ", ex.getMessage());
           throw new RuntimeException("Error occurred while listing dashboards",
ex);
       });
   }
```

- Para obter detalhes da API, consulte ListDashboards em Referência AWS SDK for Java 2.x da API.

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves a list of metric names for the specified namespace.
    *
    * @param namespace the namespace for which to retrieve the metric names
    * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
    * the metric names in the specified namespace
    * @throws RuntimeException if an error occurs while listing the metrics
    */
   public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
       ListMetricsRequest request = ListMetricsRequest.builder()
           .namespace(namespace)
           .build();

       ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
       Set<String> metSet = new HashSet<>();
       CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
           response.metrics().forEach(metric -> {
               String metricName = metric.metricName();
               metSet.add(metricName);
           });
       });

       return future
           .thenApply(ignored -> new ArrayList<>(metSet))
           .exceptionally(exception -> {
               throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
           });
   }
```

- Para obter detalhes da API, consulte ListMetrics em Referência AWS SDK for Java 2.x da API.

`PutAnomalyDetector`

O código de exemplo a seguir mostra como usar `PutAnomalyDetector`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Adds an anomaly detector for the given file.
    *
    * @param fileName the name of the file containing the anomaly detector
configuration
    * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
    */
   public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

                PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }
```

- Para obter detalhes da API, consulte PutAnomalyDetector em Referência AWS SDK for Java 2.x da API.

`PutDashboard`

O código de exemplo a seguir mostra como usar `PutDashboard`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new dashboard with the specified name and metrics from the given
file.
    *
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from the
file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                   if (messages.isEmpty()) {
                       logger.info("There are no messages in the new Dashboard.");
                   } else {
```

```
                        for (DashboardValidationMessage message : messages) {
                            logger.info("Message: {}", message.message());
                        }
                    }
                    return response; // Return the response for further use
                }
            });
    }
```

- Para obter detalhes da API, consulte PutDashboard em Referência AWS SDK for Java 2.x da API.

`PutMetricAlarm`

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
```

```
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }

        // Extract values from the JSON node.
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName = rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
```

```
            });
    }
```

- Para obter detalhes da API, consulte PutMetricAlarm em Referência AWS SDK for Java 2.x da API.

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
```

```
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

                MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
```

```
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }
```

- Para obter detalhes da API, consulte PutMetricData em Referência AWS SDK for Java 2.x da API.

Cenários

Monitoramento do desempenho do DynamoDB

O exemplo de código, apresentado a seguir, mostra como configurar o uso do DynamoDB por uma aplicação para monitorar o desempenho.

SDK para Java 2.x

Este exemplo mostra como configurar uma aplicação em Java para monitorar o desempenho do DynamoDB. O aplicativo envia dados métricos para CloudWatch onde você pode monitorar o desempenho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch
- DynamoDB

CloudWatch Exemplos de eventos usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with CloudWatch Events.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <resourceArn>

                Where:
                   resourceArn - An Amazon Resource Name (ARN) related to the
 events.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\":
 \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                    .detail(EVENT_DETAILS)
                    .detailType("sampleSubmitted")
                    .resources(resourceArn)
                    .source("aws-sdk-java-cloudwatch-example")
                    .build();

            PutEventsRequest request = PutEventsRequest.builder()
                    .entries(requestEntry)
                    .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");
```

```
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK for Java 2.x da API.

PutRule

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                  <ruleName> roleArn>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  roleArn - A role ARN value (for example,
 arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWRule(cwe, ruleName, roleArn);
        cwe.close();
    }

    public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
 roleArn) {
        try {
            PutRuleRequest request = PutRuleRequest.builder()
                    .name(ruleName)
                    .roleArn(roleArn)
                    .scheduleExpression("rate(5 minutes)")
                    .state(RuleState.ENABLED)
                    .build();

            PutRuleResponse response = cwe.putRule(request);
            System.out.printf(
                    "Successfully created CloudWatch events rule %s with arn %s",
                    roleArn, response.ruleArn());

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutRule em Referência AWS SDK for Java 2.x da API.

`PutTargets`

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> <functionArn> <targetId>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  functionArn - An AWS Lambda function ARN (for example,
 arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                  targetId - A target id value.
                """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
        String targetId = args[2];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWTargets(cwe, ruleName, functionArn, targetId);
        cwe.close();
    }

    public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
 String functionArn, String targetId) {
        try {
            Target target = Target.builder()
                    .arn(functionArn)
                    .id(targetId)
                    .build();

            PutTargetsRequest request = PutTargetsRequest.builder()
                    .targets(target)
                    .rule(ruleName)
                    .build();

            cwe.putTargets(request);
            System.out.printf(
                    "Successfully created CloudWatch events target for rule %s",
                    ruleName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutTargets em Referência AWS SDK for Java 2.x da API.

CloudWatch Exemplos de registros usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with CloudWatch Logs.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DeleteSubscriptionFilter`

O código de exemplo a seguir mostra como usar `DeleteSubscriptionFilter`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
```

```
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <logGroup>

                Where:
                  filter - The name of the subscription filter (for example,
 MyFilter).
                  logGroup - The name of the log group. (for example, testgroup).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
 String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
 DeleteSubscriptionFilterRequest.builder()
                    .filterName(filter)
```

```
                    .logGroupName(logGroup)
                    .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
 filter %s", filter);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteSubscriptionFilter em Referência AWS SDK for Java 2.x da API.

`DescribeLogStreams`

O código de exemplo a seguir mostra como usar `DescribeLogStreams`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Pesquisa fluxos de log em um grupo de logs especificado que correspondam a um determinado prefixo.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 WeathertopJavaContainerLogs).
                  logStreamName - The name of the log stream (for example,
 weathertop-java-stream).
                  pattern - the pattern to use (for example, INFO)

                """;

        if (args.length != 3) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = args[0] ;
        String logStreamName = args[1] ;
        String pattern = args[2] ;

        CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
 pattern);
    }

    /**
     * Searches for log streams with a specific prefix within a log group and
 filters log events based on a specified pattern.
     *
     * @param cwlClient       the CloudWatchLogsClient used to interact with AWS
 CloudWatch Logs
     * @param logGroupName    the name of the log group to search within
     * @param logStreamPrefix the prefix of the log streams to search for
     * @param pattern         the pattern to filter log events by
     */
```

```
    public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
 cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
        DescribeLogStreamsRequest describeLogStreamsRequest =
 DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .logStreamNamePrefix(logStreamPrefix)
                .build();

        DescribeLogStreamsResponse describeLogStreamsResponse =
 cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
 FilterLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNames(logStreamName)
                    .filterPattern(pattern)
                    .build();

            FilterLogEventsResponse filterLogEventsResponse =
 cwlClient.filterLogEvents(filterLogEventsRequest);

            for (FilteredLogEvent event : filterLogEventsResponse.events()) {
                System.out.println(event.message());
            }


 System.out.println("--------------------------------------------------"); //
 Separator for better readability
        }
    }
}
```

Imprime metadados sobre o fluxo de logs mais recente em um grupo de logs especificado.

```
/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = """
                Usage:
                  <logGroupName>

                Where:
                  logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
                """;

        if (args.length != 1) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
        CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeMostRecentLogStream(logsClient, logGroupName);
    }

    /**
     * Describes and prints metadata about the most recent log stream in the
 specified log group.
     *
     * @param logsClient   the CloudWatchLogsClient used to interact with AWS
 CloudWatch Logs
     * @param logGroupName the name of the log group
     */
    public static void describeMostRecentLogStream(CloudWatchLogsClient logsClient,
 String logGroupName) {
        DescribeLogStreamsRequest streamsRequest =
 DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .orderBy(OrderBy.LAST_EVENT_TIME)
```

```
                .descending(true)
                .limit(1)
                .build();

        try {
            DescribeLogStreamsResponse streamsResponse =
 logsClient.describeLogStreams(streamsRequest);
            List<LogStream> logStreams = streamsResponse.logStreams();

            if (logStreams.isEmpty()) {
                System.out.println("No log streams found for log group: " +
 logGroupName);
                return;
            }

            LogStream stream = logStreams.get(0);
            System.out.println("Most Recent Log Stream:");
            System.out.println("  Name: " + stream.logStreamName());
            System.out.println("  ARN: " + stream.arn());
            System.out.println("  Creation Time: " + stream.creationTime());
            System.out.println("  First Event Time: " +
 stream.firstEventTimestamp());
            System.out.println("  Last Event Time: " + stream.lastEventTimestamp());
            System.out.println("  Stored Bytes: " + stream.storedBytes());
            System.out.println("  Upload Sequence Token: " +
 stream.uploadSequenceToken());

        } catch (CloudWatchLogsException e) {
            System.err.println("Failed to describe log stream: " +
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeLogStreams em Referência AWS SDK for Java 2.x da API.

`DescribeSubscriptionFilters`

O código de exemplo a seguir mostra como usar `DescribeSubscriptionFilters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroup>

                Where:
                  logGroup - A log group name (for example, myloggroup).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        describeFilters(logs, logGroup);
        logs.close();
    }

    public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
        try {
            boolean done = false;
            String newToken = null;

            while (!done) {
                DescribeSubscriptionFiltersResponse response;
                if (newToken == null) {
                    DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                            .logGroupName(logGroup)
                            .limit(1).build();

                    response = logs.describeSubscriptionFilters(request);
                } else {
                    DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                            .nextToken(newToken)
                            .logGroupName(logGroup)
                            .limit(1).build();
                    response = logs.describeSubscriptionFilters(request);
                }

                for (SubscriptionFilter filter : response.subscriptionFilters()) {
                    System.out.printf("Retrieved filter with name %s, " + "pattern
%s " + "and destination arn %s",
                            filter.filterName(),
                            filter.filterPattern(),
                            filter.destinationArn());
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
```

```
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Done");
    }
}
```

- Para obter detalhes da API, consulte DescribeSubscriptionFilters em Referência AWS SDK for Java 2.x da API.

`GetLogEvents`

O código de exemplo a seguir mostra como usar `GetLogEvents`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 myloggroup).
                  logStreamName - The name of the log stream (for example,
 mystream).

                """;

       // if (args.length != 2) {
       //     System.out.print(usage);
       //     System.exit(1);
//        }

        String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
        String logStreamName = "weathertop-java-stream" ; //args[1];

        Region region = Region.US_EAST_1 ;
        CloudWatchLogsClient cloudWatchLogsClient = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
        cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                      String logGroupName,
                                      String logStreamPrefix) {
```

```
        try {
            // First, find the exact log stream name
            DescribeLogStreamsRequest describeRequest =
DescribeLogStreamsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNamePrefix(logStreamPrefix)
                    .limit(1) // get the first matching stream
                    .build();

            DescribeLogStreamsResponse describeResponse =
cloudWatchLogsClient.describeLogStreams(describeRequest);

            if (describeResponse.logStreams().isEmpty()) {
                System.out.println("No matching log streams found for prefix: " +
logStreamPrefix);
                return;
            }

            String exactLogStreamName =
describeResponse.logStreams().get(0).logStreamName();
            System.out.println("Using exact log stream: " + exactLogStreamName);

            long startTime = Instant.now().minus(7, ChronoUnit.DAYS).toEpochMilli();
            long endTime = Instant.now().toEpochMilli();

            GetLogEventsRequest getLogEventsRequest = GetLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamName(exactLogStreamName) // <-- exact name, not prefix
                    .startTime(startTime)
                    .endTime(endTime)
                    .startFromHead(true)
                    .build();

            GetLogEventsResponse response =
cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

            if (response.events().isEmpty()) {
                System.out.println("No log events found in the past 7 days.");
            } else {
                response.events().forEach(e -> System.out.println(e.message()));
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetLogEvents em Referência AWS SDK for Java 2.x da API.

PutSubscriptionFilter

O código de exemplo a seguir mostra como usar `PutSubscriptionFilter`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
```

```
 *
 *
 * Also, before running this Java V2 code example,set up your development
 * environment,including your credentials.
 *
 * For more information,see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <pattern> <logGroup> <functionArn>\s

                Where:
                  filter - A filter name (for example, myfilter).
                  pattern - A filter pattern (for example, ERROR).
                  logGroup - A log group name (testgroup).
                  functionArn - An AWS Lambda function ARN (for example,
 arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }
```

```
    public static void putSubFilters(CloudWatchLogsClient cwl,
            String filter,
            String pattern,
            String logGroup,
            String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
 PutSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .filterPattern(pattern)
                    .logGroupName(logGroup)
                    .destinationArn(functionArn)
                    .build();

            cwl.putSubscriptionFilter(request);
            System.out.printf(
                    "Successfully created CloudWatch logs subscription filter %s",
                    filter);

        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutSubscriptionFilter em Referência AWS SDK for Java 2.x da API.

StartLiveTail

O código de exemplo a seguir mostra como usar `StartLiveTail`.

SDK para Java 2.x

Inclua os arquivos necessários.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Gerencie os eventos da sessão do Live Tail.

```
    private static StartLiveTailResponseHandler
 getStartLiveTailResponseStreamHandler(
            AtomicReference<Subscription> subscriptionAtomicReference) {
        return StartLiveTailResponseHandler.builder()
            .onResponse(r -> System.out.println("Received initial response"))
            .onError(throwable -> {
                CloudWatchLogsException e = (CloudWatchLogsException)
 throwable.getCause();
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            })
            .subscriber(() -> new FlowableSubscriber<>() {
                @Override
                public void onSubscribe(@NonNull Subscription s) {
                    subscriptionAtomicReference.set(s);
                    s.request(Long.MAX_VALUE);
                }

                @Override
                public void onNext(StartLiveTailResponseStream event) {
                    if (event instanceof LiveTailSessionStart) {
                        LiveTailSessionStart sessionStart = (LiveTailSessionStart)
 event;
                        System.out.println(sessionStart);
                    } else if (event instanceof LiveTailSessionUpdate) {
```

```
                        LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                        List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                        logEvents.forEach(e -> {
                            long timestamp = e.timestamp();
                            Date date = new Date(timestamp);
                            System.out.println("[" + date + "] " + e.message());
                        });
                    } else {
                        throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                    }
                }

                @Override
                public void onError(Throwable throwable) {
                    System.out.println(throwable.getMessage());
                    System.exit(1);
                }

                @Override
                public void onComplete() {
                    System.out.println("Completed Streaming Session");
                }
            })
            .build();
    }
```

Inicie a sessão do Live Tail.

```
        CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
                CloudWatchLogsAsyncClient.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();

        StartLiveTailRequest request =
                StartLiveTailRequest.builder()
                    .logGroupIdentifiers(logGroupIdentifiers)
                    .logStreamNames(logStreamNames)
                    .logEventFilterPattern(logEventFilterPattern)
                    .build();
```

```
        /* Create a reference to store the subscription */
        final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

        cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

Interrompa a sessão do Live Tail após um período decorrido.

```
        /* Set a timeout for the session and cancel the subscription. This will:
         * 1). Close the stream
         * 2). Stop the Live Tail session
         */
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        if (subscriptionAtomicReference.get() != null) {
            subscriptionAtomicReference.get().cancel();
            System.out.println("Subscription to stream closed");
        }
```

- Para obter detalhes da API, consulte StartLiveTail em Referência AWS SDK for Java 2.x da API.

Cenários

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Java 2.x

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime

de Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos do Amazon Cognito Identity usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Cognito Identity.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateIdentityPool`

O código de exemplo a seguir mostra como usar `CreateIdentityPool`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateIdentityPool em Referência AWS SDK for Java 2.x da API.

`DeleteIdentityPool`

O código de exemplo a seguir mostra como usar `DeleteIdentityPool`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
```

```
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteIdentityPool em Referência AWS SDK for Java 2.x da API.

`GetCredentialsForIdentity`

O código de exemplo a seguir mostra como usar `GetCredentialsForIdentity`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }
```

```
    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetCredentialsForIdentity em Referência AWS SDK for Java 2.x da API.

`ListIdentityPools`

O código de exemplo a seguir mostra como usar `ListIdentityPools`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
```

```
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte ListIdentityPools em Referência AWS SDK for Java 2.x da API.

Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon Cognito

O exemplo de código a seguir mostra como começar a usar o Amazon Cognito.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
 {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
```

```
                    .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
 userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListUserPools em Referência AWS SDK for Java 2.x da API.

Ações

`AdminGetUser`

O código de exemplo a seguir mostra como usar `AdminGetUser`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();
```

```
            AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte AdminGetUser em Referência AWS SDK for Java 2.x da API.

`AdminInitiateAuth`

O código de exemplo a seguir mostra como usar `AdminInitiateAuth`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId) {
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
```

```
                    .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " + response.challengeName());
            return response;

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Para obter detalhes da API, consulte AdminInitiateAuth em Referência AWS SDK for Java 2.x da API.

`AdminRespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `AdminRespondToAuthChallenge`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallenge em Referência AWS SDK for Java 2.x da API.

`AssociateSoftwareToken`

O código de exemplo a seguir mostra como usar `AssociateSoftwareToken`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
```

```
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }
```

- Para obter detalhes da API, consulte AssociateSoftwareToken em Referência AWS SDK for Java 2.x da API.

`ConfirmSignUp`

O código de exemplo a seguir mostra como usar `ConfirmSignUp`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
           String userName) {
       try {
           ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                   .clientId(clientId)
                   .confirmationCode(code)
                   .username(userName)
                   .build();

           identityProviderClient.confirmSignUp(signUpRequest);
           System.out.println(userName + " was confirmed");

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ConfirmSignUp em Referência AWS SDK for Java 2.x da API.

`CreateUserPool`

O código de exemplo a seguir mostra como usar `CreateUserPool`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
```

```
                    userPoolName - The name to give your user pool when it's
created.
               """;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
       }

       String userPoolName = args[0];
       CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
               .region(Region.US_EAST_1)
               .build();

       String id = createPool(cognitoClient, userPoolName);
       System.out.println("User pool ID: " + id);
       cognitoClient.close();
   }

   public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
       try {
           CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                   .poolName(userPoolName)
                   .build();

           CreateUserPoolResponse response = cognitoClient.createUserPool(request);
           return response.userPool().id();

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
}
```

- Para obter detalhes da API, consulte CreateUserPool em Referência AWS SDK for Java 2.x da API.

`CreateUserPoolClient`

O código de exemplo a seguir mostra como usar `CreateUserPoolClient`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s
```

```
                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
 String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " + response.userPoolClient().clientName()
 + " created. ID: "
                    + response.userPoolClient().clientId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateUserPoolClient em Referência AWS SDK for Java 2.x da API.

`ListUserPools`

O código de exemplo a seguir mostra como usar `ListUserPools`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
 {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
 userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListUserPools em Referência AWS SDK for Java 2.x da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
```

```
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                        + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username() + "
Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK for Java 2.x da API.

`ResendConfirmationCode`

O código de exemplo a seguir mostra como usar `ResendConfirmationCode`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
           String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                   .clientId(clientId)
                   .username(userName)
                   .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ResendConfirmationCode em Referência AWS SDK for Java 2.x da API.

SignUp

O código de exemplo a seguir mostra como usar SignUp.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
           String password, String email) {
       AttributeType userAttrs = AttributeType.builder()
               .name("email")
               .value(email)
               .build();

       List<AttributeType> userAttrsList = new ArrayList<>();
       userAttrsList.add(userAttrs);
       try {
           SignUpRequest signUpRequest = SignUpRequest.builder()
                   .userAttributes(userAttrsList)
                   .username(userName)
                   .clientId(clientId)
                   .password(password)
                   .build();

           identityProviderClient.signUp(signUpRequest);
           System.out.println("User has been signed up ");

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte SignUp em Referência AWS SDK for Java 2.x da API.

`VerifySoftwareToken`

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte VerifySoftwareToken em Referência AWS SDK for Java 2.x da API.

Cenários

Inscrever um usuário em um grupo de usuários que exija MFA

O exemplo de código a seguir mostra como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
```

```
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the AWS
 CDK script.
                    poolId - The pool Id that you can get from the AWS CDK script.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();
```

```
        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
```

```
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId) {
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);

           AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                   .clientId(clientId)
                   .userPoolId(userPoolId)
                   .authParameters(authParameters)
                   .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                   .build();

           AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
           System.out.println("Result Challenge is : " + response.challengeName());
           return response;

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }

   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
       AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
               .session(session)
               .build();

       AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
               .associateSoftwareToken(softwareTokenRequest);
       String secretCode = tokenResponse.secretCode();
       System.out.println("Enter this token into Google Authenticator");
       System.out.println(secretCode);
       return tokenResponse.session();
   }
```

```
    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
```

```
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Exemplos do Amazon Comprehend usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Comprehend.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateDocumentClassifier`

O código de exemplo a seguir mostra como usar `CreateDocumentClassifier`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
 software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-
amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
```

```
                 dataAccessRoleArn - The ARN value of the role used for this
operation.
                 s3Uri - The Amazon S3 bucket that contains the CSV file.
                 documentClassifierName - The name of the document classifier.
               """;

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

       String dataAccessRoleArn = args[0];
       String s3Uri = args[1];
       String documentClassifierName = args[2];

       Region region = Region.US_EAST_1;
       ComprehendClient comClient = ComprehendClient.builder()
               .region(region)
               .build();

       createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
       comClient.close();
   }

   public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri,
           String documentClassifierName) {
       try {
           DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
                   .s3Uri(s3Uri)
                   .build();

           CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
                   .documentClassifierName(documentClassifierName)
                   .dataAccessRoleArn(dataAccessRoleArn)
                   .languageCode("en")
                   .inputDataConfig(config)
                   .build();

           CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient
```

```
                    .createDocumentClassifier(createDocumentClassifierRequest);
            String documentClassifierArn =
 createDocumentClassifierResult.documentClassifierArn();
            System.out.println("Document Classifier ARN: " + documentClassifierArn);

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateDocumentClassifier em Referência AWS SDK for Java 2.x da API.

`DetectDominantLanguage`

O código de exemplo a seguir mostra como usar `DetectDominantLanguage`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient, String
 text) {
        try {
            DetectDominantLanguageRequest request =
 DetectDominantLanguageRequest.builder()
                    .text(text)
                    .build();

            DetectDominantLanguageResponse resp =
 comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectDominantLanguage em Referência AWS SDK for Java 2.x da API.

`DetectEntities`

O código de exemplo a seguir mostra como usar `DetectEntities`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();
```

```
        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    public static void detectAllEntities(ComprehendClient comClient, String text) {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
 DetectEntitiesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectEntitiesResponse detectEntitiesResult =
 comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectEntities em Referência AWS SDK for Java 2.x da API.

`DetectKeyPhrases`

O código de exemplo a seguir mostra como usar `DetectKeyPhrases`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String text)
 {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
 DetectKeyPhrasesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
 comClient.detectKeyPhrases(detectKeyPhrasesRequest);
```

```
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectKeyPhrases em Referência AWS SDK for Java 2.x da API.

`DetectSentiment`

O código de exemplo a seguir mostra como usar `DetectSentiment`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }

    public static void detectSentiments(ComprehendClient comClient, String text) {
        try {
            DetectSentimentRequest detectSentimentRequest =
 DetectSentimentRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSentimentResponse detectSentimentResult =
 comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
 detectSentimentResult.sentimentScore().neutral());

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectSentiment em Referência AWS SDK for Java 2.x da API.

`DetectSyntax`

O código de exemplo a seguir mostra como usar `DetectSyntax`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }
```

```
    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSyntaxResponse detectSyntaxResult =
 comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
                System.out.println("Language is " + token.text());
                System.out.println("Part of speech is " +
 token.partOfSpeech().tagAsString());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectSyntax em Referência AWS SDK for Java 2.x da API.

Cenários

Criar um chatbot Amazon Lex

O exemplo de código a seguir mostra como criar um chatbot para engajar os visitantes do seu site.

SDK para Java 2.x

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Crie um aplicativo de mensagem

O exemplo de código a seguir mostra como criar uma aplicação de mensagens usando o Amazon SQS.

SDK para Java 2.x

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.

- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

AWS Control Tower exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS Control Tower.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Control Tower

O exemplo de código a seguir mostra como começar a usar o AWS Control Tower.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class HelloControlTower {

    public static void main(String[] args) {
        try {
            ControlTowerClient controlTowerClient = ControlTowerClient.builder()
                .build() ;
            helloControlTower(controlTowerClient);
        } catch (ControlTowerException e) {
            System.out.println("Control Tower error occurred: " +
 e.awsErrorDetails().errorMessage());
        }
    }

    /**
     * Use the AWS SDK for Java (v2) to create an AWS Control Tower client
     * and list all available baselines.
     * This example uses the default settings specified in your shared credentials
     * and config files.
     *
     * @param controlTowerClient A ControlTowerClient object. This object wraps
     *                           the low-level AWS Control Tower service API.
     */
    public static void helloControlTower(ControlTowerClient controlTowerClient) {
        System.out.println("Hello, AWS Control Tower! Let's list available
 baselines:\n");

        ListBaselinesIterable paginator = controlTowerClient.listBaselinesPaginator(
                ListBaselinesRequest.builder().build());
        List<String> baselineNames = new ArrayList<>();

        try {
            paginator.stream()
                .flatMap(response -> response.baselines().stream())
                    .forEach(baseline -> baselineNames.add(baseline.name()));
```

```
            System.out.println(baselineNames.size() + " baseline(s) retrieved.");
            for (String baselineName : baselineNames) {
                System.out.println("\t" + baselineName);
            }

        } catch (ControlTowerException e) {
            if ("AccessDeniedException".equals(e.awsErrorDetails().errorCode())) {
                System.out.println("Access denied. Please ensure you have the
 necessary permissions.");
            } else {
                System.out.println("An error occurred: " + e.getMessage());
            }
        }
    }
}
```

- Para obter detalhes da API, consulte ListBaselines em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Liste as zonas de pouso.
- Liste, ative, obtenha, redefina e desative as linhas de base.
- Liste, ative, obtenha e desative controles.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS Control Tower recursos.

```
public class ControlTowerScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Scanner scanner = new Scanner(in);

    private static OrganizationsClient orgClient;
    private static ControlCatalogClient catClient;

    private static String ouId = null;
    private static String ouArn = null;
    private static String landingZoneArn = null;
    private static boolean useLandingZone = false;

    private String stack = null;
    private String accountId = null;

    public static void main(String[] args) {

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Control Tower basics scenario!");
        System.out.println(DASHES);

        try {
            runScenarioAsync();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // ----------------------------
    // Utilities
    // ----------------------------
    private static boolean askYesNo(String msg) {
        System.out.println(msg);
        return scanner.nextLine().trim().toLowerCase().startsWith("y");
    }

    private static void runScenarioAsync() {
        try {
            ControlTowerActions actions = new ControlTowerActions();

            // ----------------------------
            // Step 1: Landing Zones
            // ----------------------------
```

```
            System.out.println(DASHES);
            System.out.println("""
        Some demo operations require the use of a landing zone.
        You can use an existing landing zone or opt out of these operations in the
 demo.
        For instructions on how to set up a landing zone,
        see https://docs.aws.amazon.com/controltower/latest/userguide/getting-
started-from-console.html
        """);

            System.out.println("Step 1: Listing landing zones...");
            waitForInputToContinue(scanner);

            List<LandingZoneSummary> landingZones =
                    actions.listLandingZonesAsync().join();

            if (landingZones.isEmpty()) {
                System.out.println("No landing zones found. Landing-zone-dependent
 steps will be skipped.");
                useLandingZone = false;
                waitForInputToContinue(scanner);
            } else {
                System.out.println("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.size(); i++) {
                    System.out.printf("%d) %s%n", i + 1, landingZones.get(i).arn());
                }

                if (askYesNo("Do you want to use the first landing zone in the list
 (" +
                        landingZones.get(0).arn() + ")? (y/n): ")) {
                    useLandingZone = true;
                    landingZoneArn = landingZones.get(0).arn();
                } else if (askYesNo("Do you want to use a different existing Landing
 Zone for this demo? (y/n): ")) {
                    useLandingZone = true;
                    System.out.println("Enter landing zone ARN: ");
                    landingZoneArn = scanner.nextLine().trim();
                } else {
                    System.out.println("Proceeding without a landing zone.");
                    useLandingZone = false;
                    waitForInputToContinue(scanner);
                }
            }
```

```
            // -----------------------------
            // Setup Organization + Sandbox OU
            // -----------------------------
            if (useLandingZone) {
                System.out.println("Using landing zone ARN: " + landingZoneArn);

                ControlTowerActions.OrgSetupResult result =
                        actions.setupOrganizationAsync().join();

                ouArn = result.sandboxOuArn();
                ouId  = result.sandboxOuArn();

                System.out.println("Organization ID: " + result.orgId());
                System.out.println("Using Sandbox OU ARN: " + ouArn);
            }

            // -----------------------------
            // Step 2: Baselines
            // -----------------------------
            System.out.println(DASHES);
            System.out.println("Step 2: Listing available baselines...");
            System.out.println("""
In this step, the program lists available AWS Control Tower baselines and may
 perform
baseline-related operations (enable, disable, reset) if requested.

NOTE:
AWS Control Tower enforces governance through baselines and mandatory controls
(guardrails). Mandatory controls are required for landing zone governance and may
restrict certain operations depending on the account, region, or organizational
 policy.

For more information, see:
- Types of baselines in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/userguide/types-of-baselines.html
- Mandatory controls (guardrails) in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/controlreference/mandatory-
controls.html
- Baseline API examples:
  https://docs.aws.amazon.com/controltower/latest/userguide/baseline-api-
examples.html
""");
```

```
            waitForInputToContinue(scanner);
            List<BaselineSummary> baselines =
                    actions.listBaselinesAsync().join();

            BaselineSummary controlTowerBaseline = null;
            for (BaselineSummary b : baselines) {
                System.out.println("Baseline: " + b.name());
                System.out.println("  ARN: " + b.arn());
                if ("AWSControlTowerBaseline".equals(b.name())) {
                    controlTowerBaseline = b;
                }
            }

            waitForInputToContinue(scanner);

            if (useLandingZone && controlTowerBaseline != null) {

                System.out.println("\nListing enabled baselines:");
                List<EnabledBaselineSummary> enabledBaselines =
                        actions.listEnabledBaselinesAsync().join();

                String enabledBaselineArn = null;
                for (EnabledBaselineSummary eb : enabledBaselines) {
                    System.out.println("Checking enabled baseline ARN: " +
 eb.arn());
                    if (eb.baselineIdentifier().equals(controlTowerBaseline.arn()))
 {
                        enabledBaselineArn = eb.arn(); // correct enabled ARN for
 this baseline
                        break; // stop after finding the matching one
                    }
                }

                if (enabledBaselineArn == null) {
                    System.out.println("No enabled baseline found for " +
 controlTowerBaseline.arn());
                } else {
                    System.out.println("Selected enabled baseline ARN for reset/
disable: " + enabledBaselineArn);
                }


                // Enable the Baseline
```

```
                if (askYesNo("Do you want to enable the Control Tower Baseline? (y/
n): ")) {
                    System.out.println("\nEnabling Control Tower Baseline...");

                    String baselineId = controlTowerBaseline.arn();
                    String enabledBaselineId =
                            actions.enableBaselineAsync(
                                    ouArn,                  // targetIdentifier #
 the OU or account ARN
                                    baselineId,             // baselineIdentifier #
 the Control Tower baseline ARN
                                    "5.0"                   // baselineVersion #
 version string
                            ).join();


                    System.out.println("Enabled baseline operation ID: " +
 enabledBaselineId);
                    if (enabledBaselineId == null) {
                       enabledBaselineId = enabledBaselineArn;
                    }

                    // Reset the Baseline
                    if (askYesNo("Do you want to reset the Control Tower Baseline?
 (y/n): ")) {
                        String operationId =

 actions.resetEnabledBaselineAsync(enabledBaselineId).join();
                        System.out.println("Reset baseline operation ID: " +
 operationId);
                    }

                    if (askYesNo("Do you want to disable the Control Tower Baseline?
 (y/n): ")) {
                        String operationId =

 actions.disableBaselineAsync(enabledBaselineId).join();
                        System.out.println("Disabled baseline operation ID: " +
 operationId);

                        System.out.println("Now we will re-enable the baseline and
 wait 1 minute before making the call...");
                        try {
                            Thread.sleep(Duration.ofMinutes(1).toMillis());
```

```
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                            System.out.println("Wait interrupted");
                        }
                        String reEnabledBaselineId = actions.enableBaselineAsync(
                                ouArn,
                                baselineId,  // reuse baseline definition ARN
                                "5.0"
                        ).join();

                        System.out.println("Re-enabled baseline operation ID: " +
reEnabledBaselineId);
                    }
                }
            }

            // ----------------------------
            // Step 3: Controls
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("Step 3: Managing Controls:");
            waitForInputToContinue(scanner);

            List<ControlSummary> controls =
                    actions.listControlsAsync().join();

            System.out.println("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.min(5, controls.size()); i++) {
                ControlSummary c = controls.get(i);
                System.out.println("%d. %s - %s".formatted(i + 1, c.name(),
c.arn()));
            }

            if (useLandingZone) {
                waitForInputToContinue(scanner);

                List<EnabledControlSummary> enabledControls =
                        actions.listEnabledControlsAsync(ouArn).join();

                System.out.println("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.size(); i++) {
                    System.out.println("%d. %s".formatted(i + 1,
enabledControls.get(i).controlIdentifier()));
                }
```

```
                String controlArnToEnable = null;
                for (ControlSummary control : controls) {
                    boolean enabled = enabledControls.stream()
                            .anyMatch(ec ->
ec.controlIdentifier().equals(control.arn()));
                    if (!enabled) {
                        controlArnToEnable = control.arn();
                        break;
                    }
                }

                waitForInputToContinue(scanner);
                if (controlArnToEnable != null &&
                        askYesNo("Do you want to enable the control " +
controlArnToEnable + "? (y/n): ")) {

                    String operationId =
                            actions.enableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Enabled control with operation ID: " +
operationId);
                }

                waitForInputToContinue(scanner);

                if (controlArnToEnable != null &&
                        askYesNo("Do you want to disable the control? (y/n): ")) {

                    String operationId =
                            actions.disableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Disable operation ID: " + operationId);
                }
            }
            System.out.println("\nThis concludes the example scenario.");
            System.out.println("Thanks for watching!");
            System.out.println(DASHES);

        } catch (CompletionException e) {
            Throwable cause = e.getCause() != null ? e.getCause() : e;
            System.out.println("Scenario failed: " + cause.getMessage());
```

```
            throw e; // bubble up for tests / callers
        } catch (Exception e) {
            System.out.println("Unexpected error running scenario: " +
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    private static void waitForInputToContinue(Scanner sc) {
        System.out.println("\nEnter 'c' then <ENTER> to continue:");
        while (true) {
            String input = sc.nextLine();
            if ("c".equalsIgnoreCase(input.trim())) {
                System.out.println("Continuing...");
                break;
            }
        }
    }
}

public class ControlTowerActions {
    private static ControlCatalogAsyncClient controlCatalogAsyncClient;
    private static ControlTowerAsyncClient controlTowerAsyncClient;
    private static OrganizationsAsyncClient orgAsyncClient;

    private static OrganizationsAsyncClient getAsyncOrgClient() {
        if (orgAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(50)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .build();

            orgAsyncClient = OrganizationsAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
```

```
        }
        return orgAsyncClient;
    }

    private static ControlCatalogAsyncClient getAsyncCatClient() {
        if (controlCatalogAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            controlCatalogAsyncClient = ControlCatalogAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return controlCatalogAsyncClient;
    }

    private static ControlTowerAsyncClient getAsyncClient() {
        if (controlTowerAsyncClient == null) {

            SdkAsyncHttpClient httpClient =
                    AwsCrtAsyncHttpClient.builder()
                            .maxConcurrency(100)
                            .connectionTimeout(Duration.ofSeconds(60))
                            .build();

            ClientOverrideConfiguration overrideConfig =
                    ClientOverrideConfiguration.builder()
                            .apiCallTimeout(Duration.ofMinutes(2))
                            .apiCallAttemptTimeout(Duration.ofSeconds(90))
                            .retryStrategy(RetryMode.STANDARD)
                            .build();
```

```
            controlTowerAsyncClient =
                    ControlTowerAsyncClient.builder()
                            .httpClient(httpClient)
                            .overrideConfiguration(overrideConfig)
                            .build();
        }

        return controlTowerAsyncClient;
    }

    public record OrgSetupResult(String orgId, String sandboxOuArn) {
    }

    public CompletableFuture<OrgSetupResult> setupOrganizationAsync() {
        System.out.println("Starting organization setup…");

        OrganizationsAsyncClient client = getAsyncOrgClient();

        // Step 1: Describe or create organization
        CompletableFuture<Organization> orgFuture = client.describeOrganization()
                .thenApply(desc -> {
                    System.out.println("Organization exists: " +
desc.organization().id());
                    return desc.organization();
                })
                .exceptionallyCompose(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof AwsServiceException awsEx &&

"AWSOrganizationsNotInUseException".equals(awsEx.awsErrorDetails().errorCode())) {
                        System.out.println("No organization found. Creating one…");
                        return
client.createOrganization(CreateOrganizationRequest.builder()
                                        .featureSet(OrganizationFeatureSet.ALL)
                                        .build())
                                .thenApply(createResp -> {
                                    System.out.println("Created organization: {}" +
createResp.organization().id());
                                    return createResp.organization();
                                });
                    }
                    return CompletableFuture.failedFuture(
                            new CompletionException("Failed to describe or create
organization", cause)
```

```
                    );
                });

        // Step 2: Locate Sandbox OU
        return orgFuture.thenCompose(org -> {
            String orgId = org.id();
            System.out.println("Organization ID: {}" + orgId);

            return client.listRoots()
                    .thenCompose(rootsResp -> {
                        if (rootsResp.roots().isEmpty()) {
                            return CompletableFuture.failedFuture(
                                    new RuntimeException("No root found in
organization")
                            );
                        }
                        String rootId = rootsResp.roots().get(0).id();

                        ListOrganizationalUnitsForParentRequest ouRequest =
                                ListOrganizationalUnitsForParentRequest.builder()
                                        .parentId(rootId)
                                        .build();

                        ListOrganizationalUnitsForParentPublisher paginator =

client.listOrganizationalUnitsForParentPaginator(ouRequest);

                        AtomicReference<String> sandboxOuArnRef = new
AtomicReference<>();
                        return paginator.subscribe(page -> {
                                    for (OrganizationalUnit ou :
page.organizationalUnits()) {
                                        if ("Sandbox".equals(ou.name())) {
                                            sandboxOuArnRef.set(ou.arn());
                                            System.out.println("Found Sandbox OU: "
+ ou.id());
                                            break;
                                        }
                                    }
                                })
                                .thenApply(v -> {
                                    String sandboxArn = sandboxOuArnRef.get();
                                    if (sandboxArn == null) {
                                        System.out.println("Sandbox OU not found.");
```

```
                                    }
                                    return new OrgSetupResult(orgId, sandboxArn);
                                });
                    });
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
            System.out.println("Failed to setup organization: {}" +
 cause.getMessage());
            throw new CompletionException(cause);
        });
    }

    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request = ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
```

```
                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing zones: "
 + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
 e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
 cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
 cause);
                });
    }


    /**
     * Lists all available baselines using pagination to retrieve complete results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
```

```
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total: "
+ baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
%s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }
```

```
    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
        System.out.println("Starting list enabled baselines paginator…");

        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();

        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {

                        response.enabledBaselines().forEach(baseline -> {
                            enabledBaselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no enabled baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println(
                                "Successfully listed enabled baselines. Total: " +
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
```

```
                            throw new CompletionException(
                                    "Access denied when listing enabled baselines:
%s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }


    /**
     * Asynchronously enables a baseline for the specified target if not already
enabled.
     *
     * @param targetIdentifier       The ARN of the target (OU or account).
     * @param baselineIdentifier     The baseline definition ARN to enable.
     * @param baselineVersion        The baseline version to enable.
     * @return A CompletableFuture containing the enabled baseline ARN, or null if
already enabled.
     */
    public CompletableFuture<String> enableBaselineAsync(
            String targetIdentifier,
            String baselineIdentifier,
            String baselineVersion
    ) {
        EnableBaselineRequest request = EnableBaselineRequest.builder()
```

```
                .baselineIdentifier(baselineIdentifier)
                .baselineVersion(baselineVersion)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableBaseline(request)
                .handle((resp, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ControlTowerException e) {
                            String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                            String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled for
this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN synchronously
                            }

                            throw new RuntimeException("Error enabling baseline: " +
code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
```

```
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op = opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED) {
                                    return enabledBaselineArn;
                                } else if (status == BaselineOperationStatus.FAILED)
{
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline operation
failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {
                                    Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
                });
    }
```

```
   /**
    * Fetches the ARN of an already-enabled baseline for the target asynchronously.
    */
   private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
       return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
               .thenApply(listResp -> {
                   for (EnabledBaselineSummary eb : listResp.enabledBaselines()) {
                       if (baselineIdentifier.equals(eb.baselineIdentifier())
                               && targetIdentifier.equals(eb.targetIdentifier())) {
                           return eb.arn();
                       }
                   }
                   return null; // not yet available
               });
   }


   /**
    * Disables a baseline for a specified target.
    *
    * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
    * @return the operation identifier
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting disable of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       DisableBaselineRequest request = DisableBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().disableBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Disable baseline operation ID: " +
operationId);
```

```
                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
```

```
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }



    /**
     * Gets the status of a baseline operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
            String operationIdentifier) {

        GetBaselineOperationRequest request = GetBaselineOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getBaselineOperation(request)
                .whenComplete((response, exception) -> {
```

```
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null
                                ? exception.getCause()
                                : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Baseline operation not found: %s"
                                                .formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }
```

```
    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
 listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target " +
 targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +" enabled
 controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
```

```
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to work
with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
%s".formatted(e.getMessage()), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled controls",
cause);
                });
    }


    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
```

```
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
```

```
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ? e.getMessage() :
"";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for this
target");
                            return null;
                        }

                        if ("ValidationException".equals(errorCode)
                                && message.contains("enforced by default")) {
                            System.out.println("Control is enforced by default and
cannot be enabled manually.");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS Control
Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }
```

```
                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }


    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
```

```
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling: " +
e.getMessage());
                            return null;
                        }
```

```
                        throw new CompletionException(
                                "Error disabling control: " + e.getMessage(), e);
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error disabling control: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }



    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
```

```
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }


    /**
     * Lists all controls in the Control Tower control catalog.
     *
     * @return a list of controls
     * @throws SdkException if a service-specific error occurs
     */
    public CompletableFuture<List<ControlSummary>> listControlsAsync() {
        System.out.println("Starting list controls paginator…");

        ListControlsRequest request = ListControlsRequest.builder().build();
        ListControlsPublisher paginator =
getAsyncCatClient().listControlsPaginator(request);
        List<ControlSummary> controls = new ArrayList<>();

        return paginator.subscribe(response -> {
```

```
                    if (response.controls() != null && !
response.controls().isEmpty()) {
                        response.controls().forEach(control -> {
                            controls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no controls.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved " +
 controls.size() +" controls."))
                .thenApply(v -> controls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof SdkException sdkEx) {
                        if (sdkEx.getMessage() != null &&
 sdkEx.getMessage().contains("AccessDeniedException")) {
                            throw new CompletionException(
                                    "Access denied when listing controls. Please
 ensure you have the necessary permissions.",
                                    sdkEx
                            );
                        } else {
                            throw new CompletionException(
                                    "SDK error listing controls:
 %s".formatted(sdkEx.getMessage()),
                                    sdkEx
                            );
                        }
                    }

                    throw new CompletionException("Failed to list controls", cause);
                });
    }

    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
 reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
```

```
    */
   public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting reset of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       ResetEnabledBaselineRequest request = ResetEnabledBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().resetEnabledBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Reset enabled baseline operation ID: " +
operationId);

                   // Polling loop
                   CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                   Runnable poller = new Runnable() {
                       @Override
                       public void run() {
                           getBaselineOperationAsync(operationId)
                                   .thenAccept(statusObj -> {
                                       String status = statusObj.toString(); //
Convert enum/status to string for printing
                                       System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                       if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                           System.out.println("Baseline operation
finished with status: " + status);
                                           resultFuture.complete(operationId);
                                       } else {
                                           // Schedule next poll in 15 seconds
                                           CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                   .execute(this);
                                       }
                                   })
```

```
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
 operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code=" +
 errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
 sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline: " +
 cause.getMessage());
                    return null;
                });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Ações

`DisableBaseline`

O código de exemplo a seguir mostra como usar `DisableBaseline`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Disables a baseline for a specified target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
```

```
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting disable of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        DisableBaselineRequest request = DisableBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().disableBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable baseline operation ID: " +
operationId);

                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
```

```
                                                CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                        .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }
```

- Para obter detalhes da API, consulte DisableBaseline em Referência AWS SDK for Java 2.x da API.

`DisableControl`

O código de exemplo a seguir mostra como usar `DisableControl`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Disables a control for a specified target.
 *
 * @param controlIdentifier the identifier of the control to disable
 * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
 * @return the operation identifier
 * @throws ControlTowerException if a service-specific error occurs
 * @throws SdkException          if an SDK error occurs
 */
public CompletableFuture<String> disableControlAsync(
        String controlIdentifier,
        String targetIdentifier) {

    DisableControlRequest request = DisableControlRequest.builder()
            .controlIdentifier(controlIdentifier)
            .targetIdentifier(targetIdentifier)
            .build();

    return getAsyncClient().disableControl(request)
            .thenCompose(response -> {
                String operationId = response.operationIdentifier();
```

```
                    System.out.println("Disable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
```

```
                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling: " +
e.getMessage());
                            return null;
                        }

                        throw new CompletionException(
                                "Error disabling control: " + e.getMessage(), e);
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error disabling control: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }
```

- Para obter detalhes da API, consulte DisableControl em Referência AWS SDK for Java 2.x da API.

`EnableBaseline`

O código de exemplo a seguir mostra como usar `EnableBaseline`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Asynchronously enables a baseline for the specified target if not already
enabled.
    *
    * @param targetIdentifier       The ARN of the target (OU or account).
    * @param baselineIdentifier     The baseline definition ARN to enable.
    * @param baselineVersion        The baseline version to enable.
    * @return A CompletableFuture containing the enabled baseline ARN, or null if
already enabled.
    */
   public CompletableFuture<String> enableBaselineAsync(
           String targetIdentifier,
           String baselineIdentifier,
           String baselineVersion
   ) {
       EnableBaselineRequest request = EnableBaselineRequest.builder()
               .baselineIdentifier(baselineIdentifier)
               .baselineVersion(baselineVersion)
               .targetIdentifier(targetIdentifier)
               .build();

       return getAsyncClient().enableBaseline(request)
               .handle((resp, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ControlTowerException e) {
                           String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                           String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                           if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                               System.out.println("Baseline is already enabled for
this target # fetching ARN...");
                               return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                       .join(); // fetch existing ARN synchronously
                           }

                           throw new RuntimeException("Error enabling baseline: " +
code + " - " + msg, e);
                       }
```

```
                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op = opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED) {
                                    return enabledBaselineArn;
                                } else if (status == BaselineOperationStatus.FAILED)
{
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline operation
failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {
                                    Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
```

```
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines()) {
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                && targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
    }
```

- Para obter detalhes da API, consulte EnableBaseline em Referência AWS SDK for Java 2.x da API.

`EnableControl`

O código de exemplo a seguir mostra como usar `EnableControl`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
```

```
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ? e.getMessage() :
"";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for this
target");
                            return null;
                        }

                        if ("ValidationException".equals(errorCode)
                                && message.contains("enforced by default")) {
```

```
                            System.out.println("Control is enforced by default and
cannot be enabled manually.");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS Control
Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }
```

- Para obter detalhes da API, consulte EnableControl em Referência AWS SDK for Java 2.x da API.

GetBaselineOperation

O código de exemplo a seguir mostra como usar `GetBaselineOperation`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Gets the status of a baseline operation.
 *
 * @param operationIdentifier the identifier of the operation
 * @return the operation status
 * @throws ControlTowerException if a service-specific error occurs
 * @throws SdkException          if an SDK error occurs
 */
public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
        String operationIdentifier) {

    GetBaselineOperationRequest request = GetBaselineOperationRequest.builder()
            .operationIdentifier(operationIdentifier)
            .build();

    return getAsyncClient().getBaselineOperation(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause() != null
                            ? exception.getCause()
                            : exception;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Baseline operation not found: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }
```

```
                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }
```

- Para obter detalhes da API, consulte GetBaselineOperation em Referência AWS SDK for Java 2.x da API.

GetControlOperation

O código de exemplo a seguir mostra como usar `GetControlOperation`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
```

```
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }
```

- Para obter detalhes da API, consulte GetControlOperation em Referência AWS SDK for Java 2.x da API.

`ListBaselines`

O código de exemplo a seguir mostra como usar `ListBaselines`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all available baselines using pagination to retrieve complete results.
     *
```

```
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total: "
 + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
 %s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
 %s".formatted(e.getMessage()),
                                e
```

```
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }
```

- Para obter detalhes da API, consulte ListBaselines em Referência AWS SDK for Java 2.x da API.

`ListEnabledBaselines`

O código de exemplo a seguir mostra como usar `ListEnabledBaselines`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
```

```
        System.out.println("Starting list enabled baselines paginator…");

        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();

        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {

                        response.enabledBaselines().forEach(baseline -> {
                            enabledBaselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no enabled baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println(
                                "Successfully listed enabled baselines. Total: " +
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled baselines:
%s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
```

```
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }
```

- Para obter detalhes da API, consulte ListEnabledBaselines em Referência AWS SDK for Java 2.x da API.

`ListEnabledControls`

O código de exemplo a seguir mostra como usar `ListEnabledControls`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
```

```
     */
    public CompletableFuture<List<EnabledControlSummary>>
 listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target " +
 targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +" enabled
 controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
 e.getMessage().contains("not registered with AWS Control Tower")) {
```

```
                                    throw new CompletionException(
                                            "Control Tower must be enabled to work
with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
%s".formatted(e.getMessage()), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled controls",
cause);
                });
    }
```

- Para obter detalhes da API, consulte ListEnabledControls em Referência AWS SDK for Java 2.x da API.

`ListLandingZones`

O código de exemplo a seguir mostra como usar `ListLandingZones`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request = ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing zones: "
 + e.getMessage(), e);
                            default:
                                throw new CompletionException(
```

```
                                        "Error listing landing zones: " +
e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
cause);
                });
    }
```

- Para obter detalhes da API, consulte ListLandingZones em Referência AWS SDK for Java 2.x da API.

`ResetEnabledBaseline`

O código de exemplo a seguir mostra como usar `ResetEnabledBaseline`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
```

```
   public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting reset of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       ResetEnabledBaselineRequest request = ResetEnabledBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().resetEnabledBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Reset enabled baseline operation ID: " +
operationId);

                   // Polling loop
                   CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                   Runnable poller = new Runnable() {
                       @Override
                       public void run() {
                           getBaselineOperationAsync(operationId)
                                   .thenAccept(statusObj -> {
                                       String status = statusObj.toString(); //
Convert enum/status to string for printing
                                       System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                       if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                           System.out.println("Baseline operation
finished with status: " + status);
                                           resultFuture.complete(operationId);
                                       } else {
                                           // Schedule next poll in 15 seconds
                                           CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                   .execute(this);
                                       }
                                   })
                                   .exceptionally(ex -> {
```

```
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code=" +
errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline: " +
cause.getMessage());
                    return null;
                });
    }
```

- Para obter detalhes da API, consulte ResetEnabledBaseline em Referência AWS SDK for Java 2.x da API.

Exemplos do Firehose usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Firehose.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`PutRecord`

O código de exemplo a seguir mostra como usar `PutRecord`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
     *
     * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
```

```
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
               .build();

           PutRecordRequest putRecordRequest = PutRecordRequest.builder()
               .deliveryStreamName(deliveryStreamName)
               .record(firehoseRecord)
               .build();

           getFirehoseClient().putRecord(putRecordRequest);
           System.out.println("Record sent: " + jsonRecord);
       } catch (Exception e) {
           throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
       }
   }
```

- Para obter detalhes da API, consulte PutRecord em Referência AWS SDK for Java 2.x da API.

`PutRecordBatch`

O código de exemplo a seguir mostra como usar `PutRecordBatch`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
    *
    * @param records           a list of maps representing the records to be sent
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord = objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
```

```
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose record",
e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }
```

- Para obter detalhes da API, consulte PutRecordBatch em Referência AWS SDK for Java 2.x da API.

Cenários

Inserção de registros no Firehose

O exemplo de código a seguir mostra como usar o Firehose para processar registros individuais e em lote.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo insere registros individuais e em lote no Firehose.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
 objectMapper.readValue(jsonContent, new TypeReference<>() {});
```

```
            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return cloudWatchClient;
    }
```

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
    *
    * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
               .build();

           PutRecordRequest putRecordRequest = PutRecordRequest.builder()
               .deliveryStreamName(deliveryStreamName)
               .record(firehoseRecord)
               .build();

           getFirehoseClient().putRecord(putRecordRequest);
           System.out.println("Record sent: " + jsonRecord);
       } catch (Exception e) {
           throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
       }
   }


   /**
    * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
    *
```

```
    * @param records           a list of maps representing the records to be sent
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord = objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose record",
e);
                   }
               }).collect(Collectors.toList());

               PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                   .deliveryStreamName(deliveryStreamName)
                   .records(batchRecords)
                   .build();

               PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);
```

```
                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime, Instant
endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)

.dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).build(
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
```

```
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
 e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
 FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8)) {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - PutRecord
  - PutRecordBatch

Exemplos do Amazon DocumentDB usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
```

```
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

Exemplos do DynamoDB usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor
- AWS contribuições da comunidade

Conceitos básicos

Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
```

```
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
```

```
        }
        System.out.println("\nDone!");
    }
}
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma tabela do DynamoDB.

```
    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
```

```
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
```

```
    }
```

Obtenha um item de uma tabela.

```
    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Exemplo completo.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the
 * Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */

public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        String tableName = "Movies";
        String fileName = "../../../resources/sample_files/movies.json";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "1. Creating an Amazon DynamoDB table named Movies with a key named year
 and a sort key named title.");
        createTable(ddb, tableName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();
```

```
        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
```

```
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Query the table.
    public static void queryTable(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            QueryConditional queryConditional = QueryConditional
                .keyEqualTo(Key.builder()
                    .partitionValue(2013)
                    .build());

            // Get items in the table and write out the ID value.
            Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
            String result = "";

            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The title of the movie is " + rec.getTitle());
                System.out.println("The movie information  is " + rec.getInfo());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Scan the table.
    public static void scanMovies(DynamoDbClient ddb, String tableName) {
        System.out.println("******* Scanning all movies.\n");
```

```
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            Iterator<Movies> results = custTable.scan().items().iterator();
            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The movie title is " + rec.getTitle());
                System.out.println("The movie year is " + rec.getYear());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();
```

```
            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName) {
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s("{\"directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack\"]")
                .build())
            .action(AttributeAction.PUT)
            .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(itemKey)
            .attributeUpdates(updatedValues)
            .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Item was updated!");
    }
```

```
    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    public static void putRecord(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

            // Populate the Table.
            Movies record = new Movies();
            record.setYear(2020);
            record.setTitle("My Movie2");
            record.setInfo("no info");
            table.putItem(record);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Added a new movie to the table.");
    }

    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
```

```
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable

- GetItem
- PutItem
- Query
- Scan
- UpdateItem

Ações

`BatchGetItem`

O código de exemplo a seguir mostra como usar `BatchGetItem`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Mostra como receber itens em lote usando o cliente de serviço.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class BatchReadItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItems(dynamoDbClient, tableName);
    }

    public static void getBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Make the batchGetItem request.
        BatchGetItemResponse batchGetItemResponse =
 dynamoDbClient.batchGetItem(batchGetItemRequest);
```

```
        // Extract and print the retrieved items.
        Map<String, List<Map<String, AttributeValue>>> responses =
 batchGetItemResponse.responses();
        if (responses.containsKey(tableName)) {
            List<Map<String, AttributeValue>> musicItems = responses.get(tableName);
            for (Map<String, AttributeValue> item : musicItems) {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            }
        } else {
            System.out.println("No items retrieved.");
        }
    }
}
```

Mostra como receber itens em lote usando o cliente de serviço e um paginador.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
```

```
            .region(region)
            .build();

        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Use batchGetItemPaginator for paginated requests.
        dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
            .flatMap(response -> response.responses().getOrDefault(tableName,
 Collections.emptyList()).stream())
            .forEach(item -> {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            });
    }
}
```

- Para obter detalhes da API, consulte BatchGetItem em Referência AWS SDK for Java 2.x da API.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Insere vários itens em uma tabela usando o cliente de serviço.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
```

```
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """";

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        addBatchItems(dynamoDbClient, tableName);
    }

    public static void addBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Specify the updates you want to perform.
        List<WriteRequest> writeRequests = new ArrayList<>();

        // Set item 1.
        Map<String, AttributeValue> item1Attributes = new HashMap<>();
        item1Attributes.put("Artist",
 AttributeValue.builder().s("Artist1").build());
        item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
        item1Attributes.put("Comments", AttributeValue.builder().s("Great
 song!").build());
        item1Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle1").build());

 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attribut

        // Set item 2.
        Map<String, AttributeValue> item2Attributes = new HashMap<>();
        item2Attributes.put("Artist",
 AttributeValue.builder().s("Artist2").build());
        item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
        item2Attributes.put("Comments", AttributeValue.builder().s("Nice
 melody.").build());
        item2Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle2").build());

 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attribut

        try {
            // Create the BatchWriteItemRequest.
```

```
            BatchWriteItemRequest batchWriteItemRequest =
 BatchWriteItemRequest.builder()
                .requestItems(Map.of(tableName, writeRequests))
                .build();

            // Execute the BatchWriteItem operation.
            BatchWriteItemResponse batchWriteItemResponse =
 dynamoDbClient.batchWriteItem(batchWriteItemRequest);

            // Process the response.
            System.out.println("Batch write successful: " + batchWriteItemResponse);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Insira vários itens em uma tabela usando o cliente aprimorado.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 with these columns:
 *   - id - the id of the record that is the key
 *   - custName - the customer name
```

```
 *   - email - the email value
 *   - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnhancedBatchWriteItems {
        public static void main(String[] args) {
                Region region = Region.US_EAST_1;
                DynamoDbClient ddb = DynamoDbClient.builder()
                                .region(region)
                                .build();
                DynamoDbEnhancedClient enhancedClient =
 DynamoDbEnhancedClient.builder()
                                .dynamoDbClient(ddb)
                                .build();
                putBatchRecords(enhancedClient);
                ddb.close();
        }

        public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
                try {
                        DynamoDbTable<Customer> customerMappedTable =
 enhancedClient.table("Customer",
                                        TableSchema.fromBean(Customer.class));
                        DynamoDbTable<Music> musicMappedTable =
 enhancedClient.table("Music",
                                        TableSchema.fromBean(Music.class));
                        LocalDate localDate = LocalDate.parse("2020-04-07");
                        LocalDateTime localDateTime = localDate.atStartOfDay();
                        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

                        Customer record2 = new Customer();
                        record2.setCustName("Fred Pink");
                        record2.setId("id110");
                        record2.setEmail("fredp@noserver.com");
                        record2.setRegistrationDate(instant);

                        Customer record3 = new Customer();
                        record3.setCustName("Susan Pink");
```

```
                        record3.setId("id120");
                        record3.setEmail("spink@noserver.com");
                        record3.setRegistrationDate(instant);

                        Customer record4 = new Customer();
                        record4.setCustName("Jerry orange");
                        record4.setId("id101");
                        record4.setEmail("jorange@noserver.com");
                        record4.setRegistrationDate(instant);

                        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
= BatchWriteItemEnhancedRequest
                                        .builder()
                                        .writeBatches(

WriteBatch.builder(Customer.class) // add items to the Customer

      // table

.mappedTableResource(customerMappedTable)

.addPutItem(builder -> builder.item(record2))

.addPutItem(builder -> builder.item(record3))

.addPutItem(builder -> builder.item(record4))
                                                                        .build(),

WriteBatch.builder(Music.class) // delete an item from the Music

   // table

.mappedTableResource(musicMappedTable)

.addDeleteItem(builder -> builder.key(

   Key.builder().partitionValue(

                    "Famous Band")

                    .build()))
                                                                        .build())
                                        .build();
```

```
                        // Add three items to the Customer table and delete one item
 from the Music
                        // table.

 enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
                        System.out.println("done");

                } catch (DynamoDbException e) {
                        System.err.println(e.getMessage());
                        System.exit(1);
                }
        }
}
```

- Para obter detalhes da API, consulte BatchWriteItem em Referência AWS SDK for Java 2.x da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.BillingMode;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.OnDemandThroughput;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTable {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
 Music3).
                key - The key for the Amazon DynamoDB table (for example, Artist).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
 a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
```

```
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
 key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
            .attributeDefinitions(AttributeDefinition.builder()
                .attributeName(key)
                .attributeType(ScalarAttributeType.S)
                .build())
            .keySchema(KeySchemaElement.builder()
                .attributeName(key)
                .keyType(KeyType.HASH)
                .build())
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
 scales based on traffic.
            .tableName(tableName)
            .build();

        String newTable;
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            newTable = response.tableDescription().tableName();
            return newTable;

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK for Java 2.x da API.

DeleteItem

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyval>

                Where:
                    tableName - The Amazon DynamoDB table to delete the item from
 (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
```

```
                    keyval - The key value that represents the item to delete (for
 example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
 String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(keyToGet)
                .build();

        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteItem em Referência AWS SDK for Java 2.x da API.

DeleteTable

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to delete (for example,
 Music3).

                **Warning** This program will delete the table that you specify!
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
}
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK for Java 2.x da API.

`DescribeTable`

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
 software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information about
 (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String tableName = args[0];
        System.out.format("Getting description for %s\n\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        describeDymamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDymamoDBTable(DynamoDbClient ddb, String tableName) {
        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
                System.out.format("Table name  : %s\n", tableInfo.tableName());
                System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
                System.out.format("Status      : %s\n", tableInfo.tableStatus());
                System.out.format("Item count  : %d\n", tableInfo.itemCount());
                System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

                ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
                System.out.println("Throughput");
                System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits());
                System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits());

                List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
                System.out.println("Attributes");
                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
                }
            }

        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
}
```

- Para obter detalhes da API, consulte DescribeTable em Referência AWS SDK for Java 2.x da API.

`DescribeTimeToLive`

O código de exemplo a seguir mostra como usar `DescribeTimeToLive`.

SDK para Java 2.x

Descreva a configuração de TTL em uma tabela existente do DynamoDB usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.logging.Level;
import java.util.logging.Logger;

    public DescribeTimeToLiveResponse describeTTL(final String tableName, final
 Region region) {
        final DescribeTimeToLiveRequest request =
            DescribeTimeToLiveRequest.builder().tableName(tableName).build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.describeTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DescribeTimeToLive em Referência AWS SDK for Java 2.x da API.

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtém um item de uma tabela usando DynamoDbClient o.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
```

```
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal>

                Where:
                    tableName - The Amazon DynamoDB table from which an item is
 retrieved (for example, Music3).\s
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
                    keyval - The key value that represents the item to get (for
 example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
 tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        getDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
 key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());
```

```
        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName(tableName)
                .build();

        try {
            // If there is no matching item, GetItem does not return any data.
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
            if (returnedItem.isEmpty())
                System.out.format("No item found with the key %s!\n", key);
            else {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");
                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetItem em Referência AWS SDK for Java 2.x da API.

ListTables

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
```

```
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
}
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK for Java 2.x da API.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloca um item em uma tabela usando DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedPutItem example.
 */
public class PutItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
 <awardsval> <Songtitle> <songtitleval>

                Where:
                    tableName - The Amazon DynamoDB table in which an item is placed
 (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).
                    keyval - The key value that represents the item to get (for
 example, Famous Band).
                    albumTitle - The Album title (for example, AlbumTitle).
                    AlbumTitleValue - The name of the album (for example, Songs
 About Life ).
                    Awards - The awards column (for example, Awards).
                    AwardVal - The value of the awards (for example, 10).
                    SongTitle - The song title (for example, SongTitle).
                    SongTitleVal - The value of the song title (for example, Happy
 Day).
                **Warning** This program will  place an item that you specify into a
 table!
                """;
```

```
        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String albumTitle = args[3];
        String albumTitleValue = args[4];
        String awards = args[5];
        String awardVal = args[6];
        String songTitle = args[7];
        String songTitleVal = args[8];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
                songTitleVal);
        System.out.println("Done!");
        ddb.close();
    }

    public static void putItemInTable(DynamoDbClient ddb,
            String tableName,
            String key,
            String keyVal,
            String albumTitle,
            String albumTitleValue,
            String awards,
            String awardVal,
            String songTitle,
            String songTitleVal) {

        HashMap<String, AttributeValue> itemValues = new HashMap<>();
        itemValues.put(key, AttributeValue.builder().s(keyVal).build());
        itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
        itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());
```

```
        PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item(itemValues)
                .build();

        try {
            PutItemResponse response = ddb.putItem(request);
            System.out.println(tableName + " was successfully updated. The request
 id is "
                    + response.responseMetadata().requestId());

        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.err.println("Be sure that it exists and that you've typed its
 name correctly!");
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK for Java 2.x da API.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Consulta uma tabela usando DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedQueryRecords example.
 */
public class Query {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <partitionKeyName> <partitionKeyVal>

                Where:
                    tableName - The Amazon DynamoDB table to put the item in (for
 example, Music3).
                    partitionKeyName - The partition key name of the Amazon DynamoDB
 table (for example, Artist).
                    partitionKeyVal - The value of the partition key that should
 match (for example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String partitionKeyName = args[1];
```

```
        String partitionKeyVal = args[2];

        // For more information about an alias, see:
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
        String partitionAlias = "#a";

        System.out.format("Querying %s", tableName);
        System.out.println("");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
 partitionAlias);
        System.out.println("There were " + count + "  record(s) returned");
        ddb.close();
    }

    public static int queryTable(DynamoDbClient ddb, String tableName, String
 partitionKeyName, String partitionKeyVal,
            String partitionAlias) {
        // Set up an alias for the partition key name in case it's a reserved word.
        HashMap<String, String> attrNameAlias = new HashMap<String, String>();
        attrNameAlias.put(partitionAlias, partitionKeyName);

        // Set up mapping of the partition name with the value.
        HashMap<String, AttributeValue> attrValues = new HashMap<>();
        attrValues.put(":" + partitionKeyName, AttributeValue.builder()
                .s(partitionKeyVal)
                .build());

        QueryRequest queryReq = QueryRequest.builder()
                .tableName(tableName)
                .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
                .expressionAttributeNames(attrNameAlias)
                .expressionAttributeValues(attrValues)
                .build();

        try {
            QueryResponse response = ddb.query(queryReq);
            return response.count();
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return -1;
    }
}
```

Consulta uma tabela usando o `DynamoDbClient` e um índice secundário.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
 * **year-index**. For more information, see:
 *
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
 */
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
```

```
        queryIndex(ddb, tableName);
        ddb.close();
    }

    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(":yearValue",
 AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                    .tableName(tableName)
                    .indexName("year-index")
                    .keyConditionExpression("#year = :yearValue")
                    .expressionAttributeNames(expressionAttributesNames)
                    .expressionAttributeValues(expressionAttributeValues)
                    .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                    .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Escaneia uma tabela do Amazon DynamoDB usando. DynamoDbClient

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */

public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information from
 (for example, Music3).
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        scanItems(ddb, tableName);
        ddb.close();
    }

    public static void scanItems(DynamoDbClient ddb, String tableName) {
        try {
            ScanRequest scanRequest = ScanRequest.builder()
                    .tableName(tableName)
                    .build();

            ScanResponse response = ddb.scan(scanRequest);
            for (Map<String, AttributeValue> item : response.items()) {
                Set<String> keys = item.keySet();
                for (String key : keys) {
                    System.out.println("The key name is " + key + "\n");
                    System.out.println("The value is " + item.get(key).s());
                }
            }

        } catch (DynamoDbException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK for Java 2.x .

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualiza um item em uma tabela usando DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
 * practice to use the
 * Enhanced Client, See the EnhancedModifyItem example.
 */
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <name> <updateVal>

                Where:
```

```
                    tableName - The Amazon DynamoDB table (for example, Music3).
                    key - The name of the key in the table (for example, Artist).
                    keyVal - The value of the key (for example, Famous Band).
                    name - The name of the column where the value is updated (for
example, Awards).
                    updateVal - The value used to update an item (for example, 14).
                 Example:
                    UpdateItem Music3 Artist Famous Band Awards 14
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String name = args[3];
        String updateVal = args[4];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
        ddb.close();
    }

    public static void updateTableItem(DynamoDbClient ddb,
            String tableName,
            String key,
            String keyVal,
            String name,
            String updateVal) {

        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put(name, AttributeValueUpdate.builder()
                .value(AttributeValue.builder().s(updateVal).build())
```

```
                .action(AttributeAction.PUT)
                .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(itemKey)
                .attributeUpdates(updatedValues)
                .build();

        try {
            ddb.updateItem(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("The Amazon DynamoDB table was updated!");
    }
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

`UpdateTimeToLive`

O código de exemplo a seguir mostra como usar `UpdateTimeToLive`.

SDK para Java 2.x

Habilite o TTL em uma tabela existente do DynamoDB usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;
```

```
    public UpdateTimeToLiveResponse enableTTL(final String tableName, final String
 attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(true)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

Desabilite o TTL em uma tabela existente do DynamoDB. usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse disableTTL(
        final String tableName, final String attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
```

```
            .enabled(false)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte UpdateTimeToLive em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

SDK para Java 2.x

Mostra como criar uma aplicação Web dinâmica que envia dados usando a API Java do Amazon DynamoDB e envia uma mensagem de texto usando a API Java do Amazon Simple Notification Service.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Comparar vários valores com um único atributo

O exemplo de código a seguir mostra como comparar vários valores com um único atributo no DynamoDB.

- Use o operador IN para comparar vários valores com um único atributo.
- Compare o operador IN com várias condições OR.
- Saiba quais são os benefícios de desempenho e complexidade de expressão oferecidos pelo IN.

SDK para Java 2.x

Compare vários valores com um único atributo no DynamoDB usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

    /**
     * Queries a table using the IN operator to compare multiple values with a
 single attribute.
     *
     * <p>This method demonstrates how to use the IN operator in a filter expression
     * to match an attribute against multiple values.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
```

```
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareMultipleValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
```

```
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Queries a table using multiple OR conditions to compare multiple values with
a single attribute.
     *
     * <p>This method demonstrates the alternative approach to using the IN
operator,
     * by using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareWithOrConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for OR conditions
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
```

```
        }

        // Build the OR conditions
        StringBuilder orConditions = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                orConditions.append(" OR ");
            }
            orConditions.append("#attrName = :val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression(orConditions.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Compares the performance of using the IN operator versus multiple OR
conditions.
     *
     * <p>This method demonstrates the performance difference between using the IN
operator
     * and using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return Map containing the performance comparison results
     */
    public static Map<String, Object> comparePerformance(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
```

```
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Measure performance of IN operator
            long inStartTime = System.nanoTime();
            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long inEndTime = System.nanoTime();
            long inDuration = inEndTime - inStartTime;

            // Measure performance of OR conditions
            long orStartTime = System.nanoTime();
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long orEndTime = System.nanoTime();
            long orDuration = orEndTime - orStartTime;

            // Record results
            results.put("inOperatorDuration", inDuration);
            results.put("orConditionsDuration", orDuration);
            results.put("inOperatorItems", inResponse.count());
            results.put("orConditionsItems", orResponse.count());
            results.put("inOperatorExpression", "IN operator with " +
valuesList.size() + " values");
            results.put("orConditionsExpression", valuesList.size() + " OR
conditions");
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Scans a table using the IN operator with a large number of values.
```

```
     *
     * <p>This method demonstrates how to use the IN operator with a large number of
values,
     * which can help stay within the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The scan response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithLargeInClause(
        DynamoDbClient dynamoDbClient, String tableName, String attributeName,
List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();
```

```
        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }

    /**
     * Generates a list of sample values for testing.
     *
     * <p>Helper method to generate a list of sample values for testing.
     *
     * @param valueType The type of values to generate (string, number, or boolean)
     * @param count The number of values to generate
     * @return List of generated attribute values
     */
    public static List<AttributeValue> generateSampleValues(String valueType, int
count) {
        List<AttributeValue> values = new ArrayList<>();

        for (int i = 0; i < count; i++) {
            AttributeValue value;

            switch (valueType.toLowerCase(Locale.ROOT)) {
                case "string":
                    value = AttributeValue.builder().s("Value" + i).build();
                    break;
                case "number":
                    value = AttributeValue.builder().n(String.valueOf(i)).build();
                    break;
                case "boolean":
                    value = AttributeValue.builder().bool(i % 2 == 0).build();
                    break;
                default:
                    throw new IllegalArgumentException("Unsupported value type: " +
valueType);
            }

            values.add(value);
        }

        return values;
    }
```

Exemplo de uso de comparação de vários valores com AWS SDK for Java 2.x.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
       System.out.println("Demonstrating how to compare multiple values with a
single attribute in DynamoDB");

       try {
           // Example 1: Using the IN operator
           System.out.println("\nExample 1: Using the IN operator");
           List<AttributeValue> categories = List.of(
               AttributeValue.builder().s("Electronics").build(),
               AttributeValue.builder().s("Computers").build(),
               AttributeValue.builder().s("Accessories").build());

           QueryResponse inResponse = compareMultipleValues(
               dynamoDbClient,
               tableName,
               "Department",
               AttributeValue.builder().s("Retail").build(),
               "Category",
               categories);

           System.out.println("Found " + inResponse.count() + " items using IN
operator");
           System.out.println("Items: " + inResponse.items());

           // Example 2: Using multiple OR conditions
           System.out.println("\nExample 2: Using multiple OR conditions");
           QueryResponse orResponse = compareWithOrConditions(
               dynamoDbClient,
               tableName,
               "Department",
               AttributeValue.builder().s("Retail").build(),
               "Category",
               categories);

           System.out.println("Found " + orResponse.count() + " items using OR
conditions");
           System.out.println("Items: " + orResponse.items());

           // Example 3: Performance comparison
           System.out.println("\nExample 3: Performance comparison");
           Map<String, Object> perfComparison = comparePerformance(
```

```
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            if ((boolean) perfComparison.get("success")) {
                System.out.println("IN operator duration: " +
perfComparison.get("inOperatorDuration") + " ns");
                System.out.println("OR conditions duration: " +
perfComparison.get("orConditionsDuration") + " ns");
                System.out.println("IN operator found " +
perfComparison.get("inOperatorItems") + " items");
                System.out.println("OR conditions found " +
perfComparison.get("orConditionsItems") + " items");
                System.out.println("Expression complexity comparison:");
                System.out.println("  IN operator: " +
perfComparison.get("inOperatorExpression"));
                System.out.println("  OR conditions: " +
perfComparison.get("orConditionsExpression"));
            } else {
                System.out.println("Performance comparison failed: " +
perfComparison.get("error"));
            }

            // Example 4: Using IN with a large number of values
            System.out.println("\nExample 4: Using IN with a large number of
values");
            List<AttributeValue> productIds = generateSampleValues("string", 20);

            ScanResponse largeInResponse = scanWithLargeInClause(dynamoDbClient,
tableName, "ProductId", productIds);

            System.out.println(
                "Found " + largeInResponse.count() + " items using IN with " +
productIds.size() + " values");

            // Explain the benefits of using IN
            System.out.println("\nKey points about using the IN operator in
DynamoDB:");
            System.out.println("1. The IN operator allows comparing a single
attribute against multiple values");
```

```
            System.out.println("2. IN is more concise than using multiple OR
 conditions");
            System.out.println("3. IN counts as only 1 operator regardless of the
 number of values");
            System.out.println("4. Multiple OR conditions count as 1 operator per
 condition plus 1 per OR");
            System.out.println("5. Using IN helps stay within the 300 operator limit
 for complex expressions");
            System.out.println("6. IN can be used in filter expressions and
 condition expressions");
            System.out.println("7. The IN operator supports up to 100 comparison
 values");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - Query
  - Scan

Atualizar condicionalmente a TTL de um item

O exemplo de código a seguir mostra como atualizar condicionalmente a TTL de um item.

SDK para Java 2.x

Atualize a TTL em um item do DynamoDB existente em uma tabela, com uma condição.

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
```

```
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Map;
import java.util.Optional;

/**
 * Updates an item in a DynamoDB table with TTL attributes using a conditional
 expression.
 * This class demonstrates how to conditionally update TTL expiration timestamps.
 */
public class UpdateTTLConditional {

    private static final String USAGE =
        """
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash or
 partition key.
                sortKey - The name of the sort key. Also known as the range
 attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table is
 located in. (Default: us-east-1)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String UPDATED_AT_ATTR = "updatedAt";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String UPDATE_EXPRESSION = "SET " + UPDATED_AT_ATTR + "=:c,
 " + EXPIRE_AT_ATTR + "=:e";
    private static final String CONDITION_EXPRESSION = EXPIRE_AT_ATTR + " > :c";
    private static final String SUCCESS_MESSAGE = "%s UpdateItem operation with TTL
 successful.";
    private static final String CONDITION_FAILED_MESSAGE = "Condition check failed.
 The item may have already expired.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
 table \"%s\" can't be found.";
```

```
    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs an UpdateTTLConditional with a default DynamoDB client.
     */
    public UpdateTTLConditional() {
        this.dynamoDbClient = null;
    }

    /**
     * Constructs an UpdateTTLConditional with the specified DynamoDB client.
     *
     * @param dynamoDbClient The DynamoDB client to use
     */
    public UpdateTTLConditional(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Main method to demonstrate conditionally updating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new UpdateTTLConditional().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and conditionally update an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
```

```
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = Map.of(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKey).build(),
            SORT_KEY_ATTR, AttributeValue.builder().s(sortKey).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = Map.of(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build(),
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .conditionExpression(CONDITION_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return 0;
        } catch (ConditionalCheckFailedException e) {
            System.err.println(CONDITION_FAILED_MESSAGE);
            throw e;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
```

```
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

Contar operadores de expressão

O exemplo de código a seguir mostra como contar operadores de expressão no DynamoDB.

- Entenda o limite de 300 operadores do DynamoDB.
- Conte operadores em expressões complexas.
- Otimize as expressões para permanecer dentro dos limites.

SDK para Java 2.x

Demonstre a contagem de operadores de expressão usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

    /**
     * Creates a complex filter expression with a specified number of conditions.
     *
     * <p>This method demonstrates how to generate a complex expression with
     * a specific number of operators to test the 300 operator limit.
     *
```

```
    * @param conditionsCount Number of conditions to include
    * @param useAnd Whether to use AND (true) or OR (false) between conditions
    * @return Map containing the filter expression, attribute values, and operator
count
    */
   public static Map<String, Object> createComplexFilterExpression(int
conditionsCount, boolean useAnd) {
       // Initialize the expression parts and attribute values
       StringBuilder filterExpression = new StringBuilder();
       Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

       // Generate the specified number of conditions
       for (int i = 0; i < conditionsCount; i++) {
           // Add the operator between conditions (except for the first one)
           if (i > 0) {
               filterExpression.append(useAnd ? " AND " : " OR ");
           }

           // Alternate between different comparison operators for variety
           String valueKey = ":val" + i;

           switch (i % 5) {
               case 0:
                   filterExpression.append("attribute").append(i).append(" =
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey, AttributeValue.builder().s("value" + i).build());
                   break;
               case 1:
                   filterExpression.append("attribute").append(i).append(" >
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                   break;
               case 2:
                   filterExpression.append("attribute").append(i).append(" <
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey,
                       AttributeValue.builder().n(String.valueOf(i * 10)).build());
                   break;
               case 3:
                   filterExpression
```

```
                        .append("contains(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("substring" +
i).build());
                    break;
                case 4:
                    filterExpression
                        .append("attribute_exists(attribute")
                        .append(i)
                        .append(")");
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each condition has 1 operator (=, >, <, contains, attribute_exists)
        // Each AND or OR between conditions is 1 operator
        int operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount
- 1 : 0);

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("filterExpression", filterExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
     * Creates a complex update expression with a specified number of operations.
     *
     * <p>This method demonstrates how to generate a complex update expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param operationsCount Number of operations to include
```

```
    * @return Map containing the update expression, attribute values, and operator
count
    */
   public static Map<String, Object> createComplexUpdateExpression(int
operationsCount) {
       // Initialize the expression parts and attribute values
       StringBuilder updateExpression = new StringBuilder("SET ");
       Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

       // Generate the specified number of SET operations
       for (int i = 0; i < operationsCount; i++) {
           // Add comma between operations (except for the first one)
           if (i > 0) {
               updateExpression.append(", ");
           }

           // Alternate between different types of SET operations
           String valueKey = ":val" + i;

           switch (i % 3) {
               case 0:
                   // Simple assignment (1 operator: =)
                   updateExpression.append("attribute").append(i).append(" =
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey, AttributeValue.builder().s("value" + i).build());
                   break;
               case 1:
                   // Addition (2 operators: = and +)
                   updateExpression
                       .append("attribute")
                       .append(i)
                       .append(" = attribute")
                       .append(i)
                       .append(" + ")
                       .append(valueKey);
                   expressionAttributeValues.put(
                       valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                   break;
               case 2:
                   // Conditional assignment with if_not_exists (2 operators: = and
if_not_exists)
                   updateExpression
```

```
                        .append("attribute")
                        .append(i)
                        .append(" = if_not_exists(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each operation has 1-2 operators as noted above
        int operatorCount = 0;
        for (int i = 0; i < operationsCount; i++) {
            operatorCount += (i % 3 == 0) ? 1 : 2;
        }

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("updateExpression", updateExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
     * Test the operator limit by attempting an operation with a complex expression.
     *
     * <p>This method demonstrates what happens when an expression approaches or
     * exceeds the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param operatorCount Target number of operators to include
```

```
     * @return Map containing the result of the operation attempt
     */
    public static Map<String, Object> testOperatorLimit(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int operatorCount) {

        // Create a complex update expression with the specified operator count
        Map<String, Object> expressionData =
            createComplexUpdateExpression((int) Math.ceil(operatorCount / 1.5)); //
Adjust to get close to target count

        String updateExpression = (String) expressionData.get("updateExpression");
        @SuppressWarnings("unchecked")
        Map<String, AttributeValue> expressionAttributeValues =
            (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
        int actualCount = (int) expressionData.get("operatorCount");

        System.out.println("Generated update expression with approximately " +
actualCount + " operators");

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression(updateExpression)
            .expressionAttributeValues(expressionAttributeValues)
            .returnValues("UPDATED_NEW")
            .build();

        try {
            // Attempt the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            Map<String, Object> result = new HashMap<>();
            result.put("success", true);
            result.put("message", "Operation succeeded with " + actualCount + "
operators");
            result.put("data", response);
            return result;

        } catch (DynamoDbException e) {
            // Check if the error is due to exceeding the operator limit
            if (e.getMessage().contains("too many operators")) {
```

```
                Map<String, Object> result = new HashMap<>();
                result.put("success", false);
                result.put("message", "Operation failed: " + e.getMessage());
                result.put("operatorCount", actualCount);
                return result;
            }

            // Return other errors
            Map<String, Object> result = new HashMap<>();
            result.put("success", false);
            result.put("message", "Operation failed: " + e.getMessage());
            result.put("error", e);
            return result;
        }
    }

    /**
     * Break down a complex expression into multiple simpler operations.
     *
     * <p>This method demonstrates how to handle expressions that would exceed
     * the 300 operator limit by breaking them into multiple operations.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param totalOperations Total number of operations to perform
     * @return Map containing the results of the operations
     */
    public static Map<String, Object> breakDownComplexExpression(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int totalOperations) {

        // Calculate how many operations we can safely include in each batch
        // Using 150 as a conservative limit (well below 300)
        final int operationsPerBatch = 100;
        final int batchCount = (int) Math.ceil((double) totalOperations /
operationsPerBatch);

        System.out.println("Breaking down " + totalOperations + " operations into "
+ batchCount + " batches");

        Map<String, Object> results = new HashMap<>();
        results.put("totalBatches", batchCount);
```

```
        Map<Integer, Map<String, Object>> batchResults = new HashMap<>();

        // Process each batch
        for (int batch = 0; batch < batchCount; batch++) {
            // Calculate the operations for this batch
            int batchStart = batch * operationsPerBatch;
            int batchEnd = Math.min(batchStart + operationsPerBatch,
totalOperations);
            int batchSize = batchEnd - batchStart;

            System.out.println(
                "Processing batch " + (batch + 1) + "/" + batchCount + " with " +
batchSize + " operations");

            // Create an update expression for this batch
            Map<String, Object> expressionData =
createComplexUpdateExpression(batchSize);

            String updateExpression = (String)
expressionData.get("updateExpression");
            @SuppressWarnings("unchecked")
            Map<String, AttributeValue> expressionAttributeValues =
                (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
            int operatorCount = (int) expressionData.get("operatorCount");

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression(updateExpression)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues("UPDATED_NEW")
                .build();

            try {
                // Perform the update operation for this batch
                UpdateItemResponse response = dynamoDbClient.updateItem(request);

                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", true);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("attributes", response.attributes());
```

```
                batchResults.put(batch, batchResult);

            } catch (DynamoDbException e) {
                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", false);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("error", e.getMessage());

                batchResults.put(batch, batchResult);

                // Continue with next batch instead of breaking
                continue;
            }
        }

        results.put("results", batchResults);
        return results;
    }

    /**
     * Count operators in a DynamoDB expression based on the rules in the
documentation.
     *
     * <p>This method demonstrates how operators are counted according to the
     * DynamoDB documentation.
     *
     * @param expression The DynamoDB expression to analyze
     * @return Map containing the breakdown of operator counts
     */
    public static Map<String, Integer> countOperatorsInExpression(String expression)
{
        // Initialize counters for different operator types
        Map<String, Integer> counts = new HashMap<>();
        counts.put("comparisonOperators", 0);
        counts.put("logicalOperators", 0);
        counts.put("functions", 0);
        counts.put("arithmeticOperators", 0);
        counts.put("specialOperators", 0);
        counts.put("total", 0);

        // Count comparison operators (=, <>, <, <=, >, >=)
        // This is a simplified approach and may not catch all cases
```

```
        int comparisonCount = 0;
        Pattern comparisonPattern = Pattern.compile("(=|<>|<=|>=|<|>)");
        Matcher comparisonMatcher = comparisonPattern.matcher(expression);
        while (comparisonMatcher.find()) {
            comparisonCount++;
        }
        counts.put("comparisonOperators", comparisonCount);

        // Count logical operators (AND, OR, NOT)
        int andCount = countOccurrences(expression, "\\bAND\\b");
        int orCount = countOccurrences(expression, "\\bOR\\b");
        int notCount = countOccurrences(expression, "\\bNOT\\b");
        counts.put("logicalOperators", andCount + orCount + notCount);

        // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
        int functionCount = countOccurrences(
            expression,
            "\\b(attribute_exists|attribute_not_exists|attribute_type|begins_with|
contains|size|if_not_exists)\\(");
        counts.put("functions", functionCount);

        // Count arithmetic operators (+ and -)
        // This is a simplified approach and may not catch all cases
        int arithmeticCount = 0;
        Pattern arithmeticPattern = Pattern.compile("[a-zA-Z0-9_)\\]]\\s*[\\+\\-]\
\s*[a-zA-Z0-9_:(]");
        Matcher arithmeticMatcher = arithmeticPattern.matcher(expression);
        while (arithmeticMatcher.find()) {
            arithmeticCount++;
        }
        counts.put("arithmeticOperators", arithmeticCount);

        // Count special operators (BETWEEN, IN)
        int betweenCount = countOccurrences(expression, "\\bBETWEEN\\b");
        int inCount = countOccurrences(expression, "\\bIN\\b");
        counts.put("specialOperators", betweenCount + inCount);

        // Add extra operators for BETWEEN (each BETWEEN includes an AND)
        int currentLogicalOps = counts.getOrDefault("logicalOperators", 0);
        counts.put("logicalOperators", currentLogicalOps + betweenCount);

        // Calculate total
        int total = counts.getOrDefault("comparisonOperators", 0)
```

```
            + counts.getOrDefault("logicalOperators", 0)
            + counts.getOrDefault("functions", 0)
            + counts.getOrDefault("arithmeticOperators", 0)
            + counts.getOrDefault("specialOperators", 0);
        counts.put("total", total);

        return counts;
    }

    /**
     * Helper method to count occurrences of a pattern in a string.
     *
     * @param text The text to search in
     * @param regex The regular expression pattern to search for
     * @return The number of occurrences
     */
    private static int countOccurrences(String text, String regex) {
        final Pattern pattern = Pattern.compile(regex);
        final Matcher matcher = pattern.matcher(text);
        int count = 0;
        while (matcher.find()) {
            count++;
        }
        return count;
    }
```

Exemplo de uso da contagem de operadores de expressão com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating DynamoDB expression operator counting and
the 300 operator limit");

        try {
            // Example 1: Analyze a simple expression
            System.out.println("\nExample 1: Analyzing a simple expression");
            String simpleExpression = "Price = :price AND Rating > :rating AND
Category IN (:cat1, :cat2, :cat3)";
```

```
            Map<String, Integer> simpleCount =
countOperatorsInExpression(simpleExpression);

            System.out.println("Expression: " + simpleExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
simpleCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
simpleCount.get("logicalOperators"));
            System.out.println("- Functions: " + simpleCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
simpleCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
simpleCount.get("specialOperators"));
            System.out.println("- Total operators: " + simpleCount.get("total"));

            // Example 2: Analyze a complex expression
            System.out.println("\nExample 2: Analyzing a complex expression");
            String complexExpression = "(attribute_exists(Category) AND Size
BETWEEN :min AND :max) OR "
                + "(Price > :price AND contains(Description, :keyword) AND "
                + "(Rating >= :minRating OR Reviews > :minReviews))";
            Map<String, Integer> complexCount =
countOperatorsInExpression(complexExpression);

            System.out.println("Expression: " + complexExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
complexCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
complexCount.get("logicalOperators"));
            System.out.println("- Functions: " + complexCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
complexCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
complexCount.get("specialOperators"));
            System.out.println("- Total operators: " + complexCount.get("total"));

            // Example 3: Test approaching the operator limit
            System.out.println("\nExample 3: Testing an expression approaching the
operator limit");
            Map<String, Object> approachingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 290);
            System.out.println(approachingLimit.get("message"));
```

```
            // Example 4: Test exceeding the operator limit
            System.out.println("\nExample 4: Testing an expression exceeding the
operator limit");
            Map<String, Object> exceedingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 310);
            System.out.println(exceedingLimit.get("message"));

            // Example 5: Breaking down a complex expression
            System.out.println("\nExample 5: Breaking down a complex expression into
multiple operations");
            Map<String, Object> breakdownResult =
breakDownComplexExpression(dynamoDbClient, tableName, key, 500);
            @SuppressWarnings("unchecked")
            Map<Integer, Map<String, Object>> results =
                (Map<Integer, Map<String, Object>>) breakdownResult.get("results");
            System.out.println(
                "Processed " + results.size() + " of " +
breakdownResult.get("totalBatches") + " batches");

            // Explain the operator counting rules
            System.out.println("\nKey points about DynamoDB expression operator
counting:");
            System.out.println("1. The maximum number of operators in any expression
is 300");
            System.out.println("2. Each comparison operator (=, <>, <, <=, >, >=)
counts as 1 operator");
            System.out.println("3. Each logical operator (AND, OR, NOT) counts as 1
operator");
            System.out.println("4. Each function call (attribute_exists, contains,
etc.) counts as 1 operator");
            System.out.println("5. Each arithmetic operator (+ or -) counts as 1
operator");
            System.out.println("6. BETWEEN counts as 2 operators (BETWEEN itself and
the AND within it)");
            System.out.println("7. IN counts as 1 operator regardless of the number
of values");
            System.out.println("8. Parentheses for grouping and attribute paths
don't count as operators");
            System.out.println("9. When you exceed the limit, the error always
reports '301 operators'");
            System.out.println("10. For complex operations, break them into multiple
smaller operations");
```

```
        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma tabela com um índice secundário global

O código de exemplo a seguir mostra como criar uma tabela com um índice secundário global.

SDK para Java 2.x

Crie uma tabela do DynamoDB com o Global Secondary Index usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProjectionType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public void createTable() {
        try {
            // Attribute definitions
            final List<AttributeDefinition> attributeDefinitions = new
 ArrayList<>();
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(ISSUE_ID_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(TITLE_ATTR)
                .attributeType(ScalarAttributeType.S)
```

```
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(CREATE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(DUE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());

            // Key schema for table
            final List<KeySchemaElement> tableKeySchema = new ArrayList<>();
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.HASH)
                .build()); // Partition key
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.RANGE)
                .build()); // Sort key

            // Initial provisioned throughput settings for the indexes
            final ProvisionedThroughput ptIndex = ProvisionedThroughput.builder()
                .readCapacityUnits(1L)
                .writeCapacityUnits(1L)
                .build();

            // CreateDateIndex
            final List<KeySchemaElement> createDateKeySchema = new ArrayList<>();
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(CREATE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection createDateProjection = Projection.builder()
                .projectionType(ProjectionType.INCLUDE)
                .nonKeyAttributes(DESCRIPTION_ATTR, STATUS_ATTR)
                .build();
```

```
            final GlobalSecondaryIndex createDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(CREATE_DATE_INDEX)
                .keySchema(createDateKeySchema)
                .projection(createDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // TitleIndex
            final List<KeySchemaElement> titleKeySchema = new ArrayList<>();
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection titleProjection =

Projection.builder().projectionType(ProjectionType.KEYS_ONLY).build();

            final GlobalSecondaryIndex titleIndex = GlobalSecondaryIndex.builder()
                .indexName(TITLE_INDEX)
                .keySchema(titleKeySchema)
                .projection(titleProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // DueDateIndex
            final List<KeySchemaElement> dueDateKeySchema = new ArrayList<>();
            dueDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(DUE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());

            final Projection dueDateProjection =
                Projection.builder().projectionType(ProjectionType.ALL).build();

            final GlobalSecondaryIndex dueDateIndex = GlobalSecondaryIndex.builder()
                .indexName(DUE_DATE_INDEX)
                .keySchema(dueDateKeySchema)
                .projection(dueDateProjection)
```

```
                .provisionedThroughput(ptIndex)
                .build();

            final CreateTableRequest createTableRequest =
CreateTableRequest.builder()
                .tableName(TABLE_NAME)
                .keySchema(tableKeySchema)
                .attributeDefinitions(attributeDefinitions)
                .globalSecondaryIndexes(createDateIndex, titleIndex, dueDateIndex)
                .provisionedThroughput(ProvisionedThroughput.builder()
                    .readCapacityUnits(1L)
                    .writeCapacityUnits(1L)
                    .build())
                .build();

            System.out.println("Creating table " + TABLE_NAME + "...");
            dynamoDbClient.createTable(createTableRequest);

            // Wait for table to become active
            System.out.println("Waiting for " + TABLE_NAME + " to become
ACTIVE...");
            final DynamoDbWaiter waiter = dynamoDbClient.waiter();
            final DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder().tableName(TABLE_NAME).build();

            final WaiterResponse<DescribeTableResponse> waiterResponse =
                waiter.waitUntilTableExists(describeTableRequest);
            waiterResponse.matched().response().ifPresent(response ->
System.out.println("Table is now ready for use"));

        } catch (DynamoDbException e) {
            System.err.println("Error creating table: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK for Java 2.x da API.

Criar uma tabela com o throughput a quente habilitado

O exemplo de código a seguir mostra como criar uma tabela com o throughput a quente habilitado.

SDK para Java 2.x

Crie uma tabela do DynamoDB com uma configuração de throughput a quente usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Builds a ProvisionedThroughput object with the specified read and write
 capacity units.
     *
     * @param readCapacityUnits The read capacity units
     * @param writeCapacityUnits The write capacity units
     * @return A configured ProvisionedThroughput object
     */
    public static ProvisionedThroughput buildProvisionedThroughput(
        final Long readCapacityUnits, final Long writeCapacityUnits) {
        return ProvisionedThroughput.builder()
            .readCapacityUnits(readCapacityUnits)
            .writeCapacityUnits(writeCapacityUnits)
            .build();
    }

    /**
     * Builds an AttributeDefinition with the specified name and type.
     *
```

```
     * @param attributeName The attribute name
     * @param scalarAttributeType The attribute type
     * @return A configured AttributeDefinition
     */
    private static AttributeDefinition buildAttributeDefinition(
        final String attributeName, final ScalarAttributeType scalarAttributeType) {
        return AttributeDefinition.builder()
            .attributeName(attributeName)
            .attributeType(scalarAttributeType)
            .build();
    }

    /**
     * Builds a KeySchemaElement with the specified name and key type.
     *
     * @param attributeName The attribute name
     * @param keyType The key type (HASH or RANGE)
     * @return A configured KeySchemaElement
     */
    private static KeySchemaElement buildKeySchemaElement(final String
 attributeName, final KeyType keyType) {
        return KeySchemaElement.builder()
            .attributeName(attributeName)
            .keyType(keyType)
            .build();
    }

    /**
     * Creates a DynamoDB table with the specified configuration including warm
 throughput settings.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to create
     * @param partitionKey The partition key attribute name
     * @param sortKey The sort key attribute name
     * @param miscellaneousKeyAttribute Additional key attribute name for GSI
     * @param nonKeyAttribute Non-key attribute to include in GSI projection
     * @param tableReadCapacityUnits Read capacity units for the table
     * @param tableWriteCapacityUnits Write capacity units for the table
     * @param tableWarmReadUnitsPerSecond Warm read units per second for the table
     * @param tableWarmWriteUnitsPerSecond Warm write units per second for the table
     * @param globalSecondaryIndexName The name of the GSI to create
     * @param globalSecondaryIndexReadCapacityUnits Read capacity units for the GSI
```

```
    * @param globalSecondaryIndexWriteCapacityUnits Write capacity units for the
GSI
    * @param globalSecondaryIndexWarmReadUnitsPerSecond Warm read units per second
for the GSI
    * @param globalSecondaryIndexWarmWriteUnitsPerSecond Warm write units per
second for the GSI
    */
   public static void createDynamoDBTable(
       final DynamoDbClient ddb,
       final String tableName,
       final String partitionKey,
       final String sortKey,
       final String miscellaneousKeyAttribute,
       final String nonKeyAttribute,
       final Long tableReadCapacityUnits,
       final Long tableWriteCapacityUnits,
       final Long tableWarmReadUnitsPerSecond,
       final Long tableWarmWriteUnitsPerSecond,
       final String globalSecondaryIndexName,
       final Long globalSecondaryIndexReadCapacityUnits,
       final Long globalSecondaryIndexWriteCapacityUnits,
       final Long globalSecondaryIndexWarmReadUnitsPerSecond,
       final Long globalSecondaryIndexWarmWriteUnitsPerSecond) {

       // Define the table attributes
       final AttributeDefinition partitionKeyAttribute =
buildAttributeDefinition(partitionKey, ScalarAttributeType.S);
       final AttributeDefinition sortKeyAttribute =
buildAttributeDefinition(sortKey, ScalarAttributeType.S);
       final AttributeDefinition miscellaneousKeyAttributeDefinition =
           buildAttributeDefinition(miscellaneousKeyAttribute,
ScalarAttributeType.N);
       final AttributeDefinition[] attributeDefinitions = {
           partitionKeyAttribute, sortKeyAttribute,
miscellaneousKeyAttributeDefinition
       };

       // Define the table key schema
       final KeySchemaElement partitionKeyElement =
buildKeySchemaElement(partitionKey, KeyType.HASH);
       final KeySchemaElement sortKeyElement = buildKeySchemaElement(sortKey,
KeyType.RANGE);
       final KeySchemaElement[] keySchema = {partitionKeyElement, sortKeyElement};
```

```
        // Define the provisioned throughput for the table
        final ProvisionedThroughput provisionedThroughput =
            buildProvisionedThroughput(tableReadCapacityUnits,
tableWriteCapacityUnits);

        // Define the Global Secondary Index (GSI)
        final KeySchemaElement globalSecondaryIndexPartitionKeyElement =
buildKeySchemaElement(sortKey, KeyType.HASH);
        final KeySchemaElement globalSecondaryIndexSortKeyElement =
            buildKeySchemaElement(miscellaneousKeyAttribute, KeyType.RANGE);
        final KeySchemaElement[] gsiKeySchema = {
            globalSecondaryIndexPartitionKeyElement,
globalSecondaryIndexSortKeyElement
        };

        final Projection gsiProjection = Projection.builder()
            .projectionType(PROJECTION_TYPE_INCLUDE)
            .nonKeyAttributes(nonKeyAttribute)
            .build();

        final ProvisionedThroughput gsiProvisionedThroughput =
            buildProvisionedThroughput(globalSecondaryIndexReadCapacityUnits,
globalSecondaryIndexWriteCapacityUnits);

        // Define the warm throughput for the Global Secondary Index (GSI)
        final WarmThroughput gsiWarmThroughput = buildWarmThroughput(
            globalSecondaryIndexWarmReadUnitsPerSecond,
globalSecondaryIndexWarmWriteUnitsPerSecond);

        final GlobalSecondaryIndex globalSecondaryIndex =
GlobalSecondaryIndex.builder()
            .indexName(globalSecondaryIndexName)
            .keySchema(gsiKeySchema)
            .projection(gsiProjection)
            .provisionedThroughput(gsiProvisionedThroughput)
            .warmThroughput(gsiWarmThroughput)
            .build();

        // Define the warm throughput for the table
        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableWarmReadUnitsPerSecond,
tableWarmWriteUnitsPerSecond);

        final CreateTableRequest request = CreateTableRequest.builder()
```

```
            .tableName(tableName)
            .attributeDefinitions(attributeDefinitions)
            .keySchema(keySchema)
            .provisionedThroughput(provisionedThroughput)
            .globalSecondaryIndexes(globalSecondaryIndex)
            .warmThroughput(tableWarmThroughput)
            .build();

        final CreateTableResponse response = ddb.createTable(request);
        System.out.println(response);
    }
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK for Java 2.x da API.

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Java 2.x

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Criar um item com TTL

O exemplo de código a seguir mostra como criar um item com TTL.

SDK para Java 2.x

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

/**
 * Creates an item in a DynamoDB table with TTL attributes.
 * This class demonstrates how to add TTL expiration timestamps to DynamoDB items.
 */
public class CreateTTL {

    private static final String USAGE =
        """
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash or
 partition key.
                sortKey - The name of the sort key. Also known as the range
 attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table is
 located in. (Default: us-east-1)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String CREATION_DATE_ATTR = "creationDate";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String SUCCESS_MESSAGE = "%s PutItem operation with TTL
 successful.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
 table \"%s\" can't be found.";
```

```
    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs a CreateTTL instance with the specified DynamoDB client.
     *
     * @param dynamoDbClient The DynamoDB client to use
     */
    public CreateTTL(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Constructs a CreateTTL with a default DynamoDB client.
     */
    public CreateTTL() {
        this.dynamoDbClient = null;
    }

    /**
     * Main method to demonstrate creating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new CreateTTL().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and create an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
```

```
        // Argument validation (remove or replace this line when reusing this code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final CreateTTL createTTL = new CreateTTL(ddb);
            createTTL.createItemWithTTL(tableName, primaryKey, sortKey);
            return 0;
        } catch (Exception e) {
            throw e;
        }
    }

    /**
     * Creates an item in the specified table with TTL attributes.
     *
     * @param tableName The name of the table
     * @param primaryKeyValue The value for the primary key
     * @param sortKeyValue The value for the sort key
     * @return The response from the PutItem operation
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     */
    public PutItemResponse createItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
sortKeyValue) {
        // Get current time in epoch second format
        final long createDate = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = createDate + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        final Map<String, AttributeValue> itemMap = new HashMap<>();
        itemMap.put(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKeyValue).build());
```

```
        itemMap.put(SORT_KEY_ATTR,
 AttributeValue.builder().s(sortKeyValue).build());
        itemMap.put(
            CREATION_DATE_ATTR,
            AttributeValue.builder().n(String.valueOf(createDate)).build());
        itemMap.put(
            EXPIRE_AT_ATTR,
            AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final PutItemRequest request =
            PutItemRequest.builder().tableName(tableName).item(itemMap).build();

        try {
            final PutItemResponse response = dynamoDbClient.putItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK for Java 2.x da API.

Criar e gerenciar tabelas globais com MRSC

O exemplo de código a seguir mostra como criar e gerenciar tabelas globais do DynamoDB com consistência Multi-Region forte (MRSC).

- Crie uma tabela com consistência Multi-Region forte.
- Verifique a configuração de MRSC e o status da réplica.
- Teste a consistência forte entre regiões com leituras imediatas.
- Faça gravações condicionais com garantia de MRSC.
- Limpe os recursos da tabela global com MRSC.

SDK para Java 2.x

Crie uma tabela regional pronta para a conversão do MRSC usando AWS SDK for Java 2.x.

```
   public static CreateTableResponse createRegionalTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }

       try {
           LOGGER.info("Creating regional table: " + tableName + " (must be empty
for MRSC)");

           CreateTableRequest createTableRequest = CreateTableRequest.builder()
               .tableName(tableName)
               .attributeDefinitions(
                   AttributeDefinition.builder()
                       .attributeName("Artist")
                       .attributeType(ScalarAttributeType.S)
                       .build(),
                   AttributeDefinition.builder()
                       .attributeName("SongTitle")
                       .attributeType(ScalarAttributeType.S)
                       .build())
               .keySchema(
                   KeySchemaElement.builder()
                       .attributeName("Artist")
                       .keyType(KeyType.HASH)
                       .build(),
                   KeySchemaElement.builder()
                       .attributeName("SongTitle")
                       .keyType(KeyType.RANGE)
                       .build())
               .billingMode(BillingMode.PAY_PER_REQUEST)
               .build();

           CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
```

```
            LOGGER.info("Regional table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create regional table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to create regional table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Converta uma tabela regional em MRSC com réplicas e uso de testemunhas. AWS SDK for Java 2.x

```
    public static UpdateTableResponse convertToMRSCWithWitness(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Converting table to MRSC with replica in " +
replicaRegion.id() + " and witness in "
                + witnessRegion.id());
```

```
            // Create replica update using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate = ReplicationGroupUpdate.builder()
                .create(CreateReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Create witness update
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .create(CreateGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .multiRegionConsistency(MultiRegionConsistency.STRONG)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC conversion initiated. Status: "
                + response.tableDescription().tableStatus());
            LOGGER.info("UpdateTableResponse full object: " + response);
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to convert table to MRSC: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to convert table to MRSC: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Descreva uma configuração de tabela global do MRSC usando AWS SDK for Java 2.x.

```
   public static DescribeTableResponse describeMRSCTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }

       try {
           LOGGER.info("Describing MRSC global table: " + tableName);

           DescribeTableRequest request =
               DescribeTableRequest.builder().tableName(tableName).build();

           DescribeTableResponse response = dynamoDbClient.describeTable(request);

           LOGGER.info("Table status: " + response.table().tableStatus());
           LOGGER.info("Multi-region consistency: " +
response.table().multiRegionConsistency());

           if (response.table().replicas() != null
               && !response.table().replicas().isEmpty()) {
               LOGGER.info("Number of replicas: " +
response.table().replicas().size());
               response.table()
                   .replicas()
                   .forEach(replica -> LOGGER.info(
                       "Replica region: " + replica.regionName() + ", Status: " +
replica.replicaStatus()));
           }

           if (response.table().globalTableWitnesses() != null
               && !response.table().globalTableWitnesses().isEmpty()) {
               LOGGER.info("Number of witnesses: "
                   + response.table().globalTableWitnesses().size());
               response.table()
                   .globalTableWitnesses()
                   .forEach(witness -> LOGGER.info(
                       "Witness region: " + witness.regionName() + ", Status: " +
witness.witnessStatus()));
```

```
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " + e.getMessage());
            throw DynamoDbException.builder()
                .message("Table not found: " + tableName)
                .cause(e)
                .build();
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to describe table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Adicione itens de teste para verificar a forte consistência do MRSC usando AWS SDK for Java 2.x.

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String album,
        final String year) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
```

```
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to MRSC global table: " + tableName);

            Map<String, AttributeValue> item = new HashMap<>();
            item.put("Artist", AttributeValue.builder().s(artist).build());
            item.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            if (album != null && !album.trim().isEmpty()) {
                item.put("Album", AttributeValue.builder().s(album).build());
            }
            if (year != null && !year.trim().isEmpty()) {
                item.put("Year", AttributeValue.builder().n(year).build());
            }

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully with strong consistency");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to add test item to table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Leia itens com leituras consistentes de réplicas do MRSC usando. AWS SDK for Java 2.x

```
    public static GetItemResponse getItemWithConsistentRead(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {
```

```
        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Reading item from MRSC global table with consistent read: "
+ tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            GetItemRequest getItemRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .consistentRead(true)
                .build();

            GetItemResponse response = dynamoDbClient.getItem(getItemRequest);

            if (response.hasItem()) {
                LOGGER.info("Item found with strong consistency - no wait time
needed");
            } else {
                LOGGER.info("Item not found");
            }

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to read item from table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
```

```
                .message("Failed to read item from table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Execute atualizações condicionais com as garantias do MRSC usando. AWS SDK for Java 2.x

```
    public static UpdateItemResponse performConditionalUpdate(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String rating) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }
        if (rating == null || rating.trim().isEmpty()) {
            throw new IllegalArgumentException("Rating cannot be null or empty");
        }

        try {
            LOGGER.info("Performing conditional update on MRSC global table: " +
tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            Map<String, String> expressionAttributeNames = new HashMap<>();
```

```
            expressionAttributeNames.put("#rating", "Rating");

            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(
                ":rating", AttributeValue.builder().n(rating).build());

            UpdateItemRequest updateItemRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #rating = :rating")
                .conditionExpression("attribute_exists(Artist)")
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            UpdateItemResponse response =
dynamoDbClient.updateItem(updateItemRequest);
            LOGGER.info("Conditional update successful - demonstrates strong
consistency");

            return response;

        } catch (ConditionalCheckFailedException e) {
            LOGGER.warning("Conditional check failed: " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to perform conditional update: " + tableName + " -
" + e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to perform conditional update: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Aguarde até que as réplicas e testemunhas do MRSC se tornem ativas usando. AWS SDK for Java 2.x

```
    public static void waitForMRSCReplicasActive(
        final DynamoDbClient dynamoDbClient, final String tableName, final int
maxWaitTimeSeconds)
        throws InterruptedException {
```

```
        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (maxWaitTimeSeconds <= 0) {
            throw new IllegalArgumentException("Max wait time must be positive");
        }

        try {
            LOGGER.info("Waiting for MRSC replicas and witnesses to become active: "
+ tableName);

            final long startTime = System.currentTimeMillis();
            final long maxWaitTimeMillis = maxWaitTimeSeconds * 1000L;
            int backoffSeconds = 5; // Start with 5 second intervals
            final int maxBackoffSeconds = 30; // Cap at 30 seconds

            while (System.currentTimeMillis() - startTime < maxWaitTimeMillis) {
                DescribeTableResponse response = describeMRSCTable(dynamoDbClient,
tableName);

                boolean allActive = true;
                StringBuilder statusReport = new StringBuilder();

                if (response.table().multiRegionConsistency() == null
                    || !MultiRegionConsistency.STRONG
                        .toString()

.equals(response.table().multiRegionConsistency().toString())) {
                    allActive = false;
                    statusReport
                        .append("MultiRegionConsistency: ")
                        .append(response.table().multiRegionConsistency())
                        .append(" ");
                }
                if (response.table().replicas() == null
                    || response.table().replicas().isEmpty()) {
                    allActive = false;
                    statusReport.append("No replicas found. ");
                }
```

```
                if (response.table().globalTableWitnesses() == null
                    || response.table().globalTableWitnesses().isEmpty()) {
                    allActive = false;
                    statusReport.append("No witnesses found. ");
                }

                // Check table status
                if (!"ACTIVE".equals(response.table().tableStatus().toString())) {
                    allActive = false;
                    statusReport
                        .append("Table: ")
                        .append(response.table().tableStatus())
                        .append(" ");
                }

                // Check replica status
                if (response.table().replicas() != null) {
                    for (var replica : response.table().replicas()) {
                        if (!"ACTIVE".equals(replica.replicaStatus().toString())) {
                            allActive = false;
                            statusReport
                                .append("Replica(")
                                .append(replica.regionName())
                                .append("): ")
                                .append(replica.replicaStatus())
                                .append(" ");
                        }
                    }
                }

                // Check witness status
                if (response.table().globalTableWitnesses() != null) {
                    for (var witness : response.table().globalTableWitnesses()) {
                        if (!"ACTIVE".equals(witness.witnessStatus().toString())) {
                            allActive = false;
                            statusReport
                                .append("Witness(")
                                .append(witness.regionName())
                                .append("): ")
                                .append(witness.witnessStatus())
                                .append(" ");
                        }
                    }
                }
```

```
                if (allActive) {
                    LOGGER.info("All MRSC replicas and witnesses are now active: " +
tableName);
                    return;
                }

                LOGGER.info("Waiting for MRSC components to become active. Status: "
+ statusReport.toString());
                LOGGER.info("Next check in " + backoffSeconds + " seconds...");

                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, maxBackoffSeconds);
            }

            throw DynamoDbException.builder()
                .message("Timeout waiting for MRSC replicas to become active after "
+ maxWaitTimeSeconds + " seconds")
                .build();

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("Failed to wait for MRSC replicas to become active: " +
tableName + " - " + e.getMessage());
            throw e;
        }
    }
```

Limpe réplicas e testemunhas do MRSC usando. AWS SDK for Java 2.x

```
    public static UpdateTableResponse cleanupMRSCReplicas(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
```

```
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Cleaning up MRSC replicas and witnesses for table: " +
tableName);

            // Remove replica using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate = ReplicationGroupUpdate.builder()
                .delete(DeleteReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Remove witness
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .delete(DeleteGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC cleanup initiated - removing replica and witness.
Response: " + response);

            return response;

        } catch (DynamoDbException e) {
```

```
            LOGGER.severe("Failed to cleanup MRSC replicas: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to cleanup MRSC replicas: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Conclua a demonstração do fluxo de trabalho do MRSC usando AWS SDK for Java 2.x.

```
    public static void demonstrateCompleteMRSCWorkflow(
        final DynamoDbClient primaryClient,
        final DynamoDbClient replicaClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion)
        throws InterruptedException {

        if (primaryClient == null) {
            throw new IllegalArgumentException("Primary DynamoDB client cannot be
null");
        }
        if (replicaClient == null) {
            throw new IllegalArgumentException("Replica DynamoDB client cannot be
null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("=== Starting Complete MRSC Workflow Demonstration ===");

            // Step 1: Create an empty single-Region table
```

```
            LOGGER.info("Step 1: Creating empty single-Region table");
            createRegionalTable(primaryClient, tableName);

            // Use the existing GlobalTableOperations method for basic table waiting
            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "] to
become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 2: Convert to MRSC with replica and witness
            LOGGER.info("Step 2: Converting to MRSC with replica and witness");
            convertToMRSCWithWitness(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for MRSC conversion to complete using MRSC-specific waiter
            LOGGER.info("Waiting for MRSC conversion to complete...");
            waitForMRSCReplicasActive(primaryClient, tableName);

            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "] to
become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 3: Verify MRSC configuration
            LOGGER.info("Step 3: Verifying MRSC configuration");
            describeMRSCTable(primaryClient, tableName);

            // Step 4: Test strong consistency with data operations
            LOGGER.info("Step 4: Testing strong consistency with data operations");

            // Add test item to primary region
            putTestItem(primaryClient, tableName, "The Beatles", "Hey Jude", "The
Beatles 1967-1970", "1968");

            // Immediately read from replica region (no wait needed with MRSC)
            LOGGER.info("Reading from replica region immediately (strong
consistency):");
            GetItemResponse getResponse =
                getItemWithConsistentRead(replicaClient, tableName, "The Beatles",
"Hey Jude");

            if (getResponse.hasItem()) {
                LOGGER.info("# Strong consistency verified - item immediately
available in replica region");
            } else {
                LOGGER.warning("# Item not found in replica region");
```

```
            }

            // Test conditional update from replica region
            LOGGER.info("Testing conditional update from replica region:");
            performConditionalUpdate(replicaClient, tableName, "The Beatles", "Hey
Jude", "5");
            LOGGER.info("# Conditional update successful - demonstrates strong
consistency");

            // Step 5: Cleanup
            LOGGER.info("Step 5: Cleaning up resources");
            cleanupMRSCReplicas(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for cleanup to complete using basic table waiter
            LOGGER.info("Waiting for replica cleanup to complete...");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // "Halt" until replica/witness cleanup is complete
            DescribeTableResponse cleanupVerification =
describeMRSCTable(primaryClient, tableName);
            int backoffSeconds = 5; // Start with 5 second intervals
            while (cleanupVerification.table().multiRegionConsistency() != null) {
                LOGGER.info("Waiting additional time (" + backoffSeconds + "
seconds) for MRSC cleanup to complete...");
                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, 30);
                cleanupVerification = describeMRSCTable(primaryClient, tableName);
            }

            // Delete the primary table
            deleteTable(primaryClient, tableName);

            LOGGER.info("=== MRSC Workflow Demonstration Complete ===");
            LOGGER.info("");
            LOGGER.info("Key benefits of Multi-Region Strong Consistency (MRSC):");
            LOGGER.info("- Immediate consistency across all regions (no eventual
consistency delays)");
            LOGGER.info("- Simplified application logic (no need to handle eventual
consistency)");
            LOGGER.info("- Support for conditional writes and transactions across
regions");
```

```
            LOGGER.info("- Consistent read operations from any region without
waiting");

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("MRSC workflow failed: " + e.getMessage());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

Crie e gerencie tabelas globais demonstrando o MREC

O exemplo de código a seguir mostra como criar e gerenciar tabelas globais do DynamoDB com réplicas entre várias regiões.

- Crie uma tabela com um índice secundário global e DynamoDB Streams.
- Adicione réplicas em diferentes regiões para criar uma tabela global.
- Remova réplicas de uma tabela global.
- Adicione itens de teste para verificar a replicação entre regiões.
- Descreva a configuração da tabela global e o status da réplica.

SDK para Java 2.x

Crie uma tabela com o Global Secondary Index e o DynamoDB Streams usando. AWS SDK for Java 2.x

```
    public static CreateTableResponse createTableWithGSI(
```

```
        final DynamoDbClient dynamoDbClient, final String tableName, final String
indexName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (indexName == null || indexName.trim().isEmpty()) {
            throw new IllegalArgumentException("Index name cannot be null or
empty");
        }

        try {
            LOGGER.info("Creating table: " + tableName + " with GSI: " + indexName);

            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                .tableName(tableName)
                .attributeDefinitions(
                    AttributeDefinition.builder()
                        .attributeName("Artist")
                        .attributeType(ScalarAttributeType.S)
                        .build(),
                    AttributeDefinition.builder()
                        .attributeName("SongTitle")
                        .attributeType(ScalarAttributeType.S)
                        .build())
                .keySchema(
                    KeySchemaElement.builder()
                        .attributeName("Artist")
                        .keyType(KeyType.HASH)
                        .build(),
                    KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .globalSecondaryIndexes(GlobalSecondaryIndex.builder()
                    .indexName(indexName)
                    .keySchema(KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.HASH)
```

```
                        .build())
                    .projection(

Projection.builder().projectionType(ProjectionType.ALL).build())
                    .build())
                .streamSpecification(StreamSpecification.builder()
                    .streamEnabled(true)
                    .streamViewType(StreamViewType.NEW_AND_OLD_IMAGES)
                    .build())
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Aguarde até que uma tabela se torne ativa usando AWS SDK for Java 2.x.

```
    public static void waitForTableActive(final DynamoDbClient dynamoDbClient, final
String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Waiting for table to become active: " + tableName);

            try (DynamoDbWaiter waiter =
```

```
                DynamoDbWaiter.builder().client(dynamoDbClient).build()) {
                DescribeTableRequest request =
                    DescribeTableRequest.builder().tableName(tableName).build();

                waiter.waitUntilTableExists(request);
                LOGGER.info("Table is now active: " + tableName);
            }

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to wait for table to become active: " + tableName
+ " - " + e.getMessage());
            throw e;
        }
    }
```

Adicione uma réplica para criar ou estender uma tabela global usando AWS SDK for Java 2.x.

```
    public static UpdateTableResponse addReplica(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final String indexName,
        final Long readCapacity) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (indexName == null || indexName.trim().isEmpty()) {
            throw new IllegalArgumentException("Index name cannot be null or
empty");
        }
        if (readCapacity == null || readCapacity <= 0) {
            throw new IllegalArgumentException("Read capacity must be a positive
number");
        }
```

```
        try {
            LOGGER.info("Adding replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for adding a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .create(builder -> builder.regionName(replicaRegion.id())
                    .globalSecondaryIndexes(ReplicaGlobalSecondaryIndex.builder()
                        .indexName(indexName)

.provisionedThroughputOverride(ProvisionedThroughputOverride.builder()
                            .readCapacityUnits(readCapacity)
                            .build())
                        .build())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica addition initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add replica in region: " + replicaRegion.id() +
" - " + e.getMessage());
            throw e;
        }
    }
```

Remova uma réplica de uma tabela global usando AWS SDK for Java 2.x.

```
    public static UpdateTableResponse removeReplica(
```

```
        final DynamoDbClient dynamoDbClient, final String tableName, final Region
replicaRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }

        try {
            LOGGER.info("Removing replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for removing a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .delete(builder -> builder.regionName(replicaRegion.id()).build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica removal initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to remove replica in region: " +
replicaRegion.id() + " - " + e.getMessage());
            throw e;
        }
    }
```

Adicione itens de teste para verificar a replicação usando AWS SDK for Java 2.x.

```
   public static PutItemResponse putTestItem(
       final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }
       if (artist == null || artist.trim().isEmpty()) {
           throw new IllegalArgumentException("Artist cannot be null or empty");
       }
       if (songTitle == null || songTitle.trim().isEmpty()) {
           throw new IllegalArgumentException("Song title cannot be null or
empty");
       }

       try {
           LOGGER.info("Adding test item to table: " + tableName);

           Map<String,
software.amazon.awssdk.services.dynamodb.model.AttributeValue> item = new
HashMap<>();
           item.put(
               "Artist",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                   .s(artist)
                   .build());
           item.put(
               "SongTitle",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                   .s(songTitle)
                   .build());

           PutItemRequest putItemRequest =
               PutItemRequest.builder().tableName(tableName).item(item).build();

           PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
```

```
            LOGGER.info("Test item added successfully");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Descreva a configuração global da tabela e as réplicas usando AWS SDK for Java 2.x.

```
    public static DescribeTableResponse describeTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Describing table: " + tableName);

            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();

            DescribeTableResponse response = dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
                        "Replica region: " + replica.regionName() + ", Status: " +
replica.replicaStatus()));
```

```
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Exemplo completo de operações globais de tabela usando AWS SDK for Java 2.x.

```
    public static void exampleUsage(final Region sourceRegion, final Region
replicaRegion) {

        String tableName = "Music";
        String indexName = "SongTitleIndex";
        Long readCapacity = 15L;

        // Create DynamoDB client for the source region
        try (DynamoDbClient dynamoDbClient =
            DynamoDbClient.builder().region(sourceRegion).build()) {

            try {
                // Step 1: Create the initial table with GSI and streams
                LOGGER.info("Step 1: Creating table in source region: " +
sourceRegion.id());
                createTableWithGSI(dynamoDbClient, tableName, indexName);

                // Step 2: Wait for table to become active
                LOGGER.info("Step 2: Waiting for table to become active");
                waitForTableActive(dynamoDbClient, tableName);

                // Step 3: Add replica in destination region
                LOGGER.info("Step 3: Adding replica in region: " +
replicaRegion.id());
                addReplica(dynamoDbClient, tableName, replicaRegion, indexName,
readCapacity);
```

```
                // Step 4: Wait a moment for replica creation to start
                Thread.sleep(5000);

                // Step 5: Describe table to view replica information
                LOGGER.info("Step 5: Describing table to view replicas");
                describeTable(dynamoDbClient, tableName);

                // Step 6: Add a test item to verify replication
                LOGGER.info("Step 6: Adding test item to verify replication");
                putTestItem(dynamoDbClient, tableName, "TestArtist", "TestSong");

                LOGGER.info("Global table setup completed successfully!");
                LOGGER.info("You can verify replication by checking the item in
region: " + replicaRegion.id());

                // Step 7: Remove replica and clean up table
                LOGGER.info("Step 7: Removing replica from region: " +
replicaRegion.id());
                removeReplica(dynamoDbClient, tableName, replicaRegion);
                DeleteTableResponse deleteTableResponse =
dynamoDbClient.deleteTable(
                    DeleteTableRequest.builder().tableName(tableName).build());
                LOGGER.info("MREC global table demonstration completed
successfully!");

            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException("Thread was interrupted", e);
            } catch (DynamoDbException e) {
                LOGGER.severe("DynamoDB operation failed: " + e.getMessage());
                throw e;
            }
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTable
  - DescribeTable
  - PutItem

- UpdateTable

Detectar EPI em imagens

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Monitoramento do desempenho do DynamoDB

O exemplo de código, apresentado a seguir, mostra como configurar o uso do DynamoDB por uma aplicação para monitorar o desempenho.

SDK para Java 2.x

Este exemplo mostra como configurar uma aplicação em Java para monitorar o desempenho do DynamoDB. O aplicativo envia dados métricos para CloudWatch onde você pode monitorar o desempenho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch

- DynamoDB

Executar operações de consulta avançada

O exemplo de código a seguir mostra como realizar operações de consulta avançada no DynamoDB.

- Consulte tabelas usando várias técnicas de filtragem e condição.
- Implemente a paginação para grandes conjuntos de resultados.
- Use índices secundários globais para padrões de acesso alternativos.
- Aplique controles de consistência com base nos requisitos da aplicação.

SDK para Java 2.x

Consulte com leituras fortemente consistentes usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
```

```
        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
 response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

Consulte usando um Índice Secundário Global com AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
```

```
   public QueryResponse queryTable(
       final String tableName, final String partitionKeyName, final String
partitionKeyValue) {

       CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);

       // Create expression attribute names for the column names
       final Map<String, String> expressionAttributeNames = new HashMap<>();
       expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

       // Create expression attribute values for the column values
       final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
       expressionAttributeValues.put(
           EXPRESSION_ATTRIBUTE_VALUE_PK,
           AttributeValue.builder().s(partitionKeyValue).build());

       // Create the query request
       final QueryRequest queryRequest = QueryRequest.builder()
           .tableName(tableName)
           .keyConditionExpression(KEY_CONDITION_EXPRESSION)
           .expressionAttributeNames(expressionAttributeNames)
           .expressionAttributeValues(expressionAttributeValues)
           .build();

       try {
           final QueryResponse response = dynamoDbClient.query(queryRequest);
           System.out.println("Query on base table successful. Found " +
response.count() + " items");
           return response;
       } catch (ResourceNotFoundException e) {
           System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
           throw new DynamoDbQueryException("Table not found: " + tableName, e);
       } catch (DynamoDbException e) {
           System.err.println("Error querying base table: " + e.getMessage());
           throw new DynamoDbQueryException("Failed to execute query on base
table", e);
       }
   }
```

```
    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
+ " items");
```

```
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
 found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
 tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

Consulta com paginação usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
```

```
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }
```

```
                System.out.println("Page " + pageNumber + ": Retrieved " +
 pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
 " + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

Consulte com filtros complexos usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
```

```
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
 priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
 categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
 Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
 Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
 statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
 priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
 categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
```

```
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

Consulte com uma expressão de filtro construída dinamicamente usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {
```

```
        validateParameters(tableName, partitionKeyName, partitionKeyValue,
filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
```

```
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
```

```
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
 multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

Consulta com uma expressão de filtro e limite de uso AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
```

```
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();
```

```
        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
 {0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
 filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
 e.getMessage());
            throw e;
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Executar operações de lista

O exemplo de código a seguir mostra como realizar operações de lista no DynamoDB.

- Adicione elementos a um atributo de lista.
- Remova elementos de um atributo de lista.
- Atualize elementos específicos em uma lista por índice.
- Use as funções list.append e list.index.

SDK para Java 2.x

Demonstre as operações de lista usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Appends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the end of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToAppend The items to append to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse appendToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToAppend) {

        // Create a list value from the items to append
        AttributeValue listValue =
 AttributeValue.builder().l(itemsToAppend).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName =
 list_append(if_not_exists(#attrName, :emptyList), :newItems)")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
```

```
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Prepends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the beginning of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToPrepend The items to prepend to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse prependToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToPrepend) {

        // Create a list value from the items to prepend
        AttributeValue listValue =
AttributeValue.builder().l(itemsToPrepend).build();

        // Define the update parameters
        // Note: To prepend, we put the new items first in the list_append function
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName = list_append(:newItems,
if_not_exists(#attrName, :emptyList))")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
```

```
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates a specific element in a list attribute.
     *
     * <p>This method demonstrates how to update a specific element in a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to update
     * @param newValue The new value for the element
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index,
        AttributeValue newValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName[" + index + "] = :newValue")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(":newValue", newValue))
            .returnValues("UPDATED_NEW")
            .build();
```

```
        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes a specific element from a list attribute.
     *
     * <p>This method demonstrates how to remove a specific element from a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #attrName[" + index + "]")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a list attribute.
     *
     * <p>Helper method to retrieve the current value of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
```

```
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param listAttributeName The name of the list attribute
     * @return The list attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static List<AttributeValue> getListAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String listAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(listAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the list attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(listAttributeName)) {
                return response.item().get(listAttributeName).l();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get list attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Exemplo de uso de operações de lista com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
```

```
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating list operations in DynamoDB");

        try {
            // Example 1: Append items to a list
            System.out.println("\nExample 1: Appending items to a list");
            List<AttributeValue> tagsToAppend = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Gadget").build());

            UpdateItemResponse appendResponse = appendToList(dynamoDbClient,
tableName, key, "Tags", tagsToAppend);

            System.out.println("Updated list attribute: " +
appendResponse.attributes());

            // Example 2: Prepend items to a list
            System.out.println("\nExample 2: Prepending items to a list");
            List<AttributeValue> tagsToPrepend = List.of(
                AttributeValue.builder().s("Featured").build(),
                AttributeValue.builder().s("New").build());

            UpdateItemResponse prependResponse = prependToList(dynamoDbClient,
tableName, key, "Tags", tagsToPrepend);

            System.out.println("Updated list attribute: " +
prependResponse.attributes());

            // Example 3: Update a specific element in a list
            System.out.println("\nExample 3: Updating a specific element in a
list");
            UpdateItemResponse updateResponse = updateListElement(
                dynamoDbClient,
                tableName,
                key,
                "Tags",
                0,
                AttributeValue.builder().s("BestSeller").build());

            System.out.println("Updated list attribute: " +
updateResponse.attributes());

            // Example 4: Remove a specific element from a list
```

```
            System.out.println("\nExample 4: Removing a specific element from a
list");
            UpdateItemResponse removeResponse = removeListElement(dynamoDbClient,
tableName, key, "Tags", 1);

            System.out.println("Updated list attribute: " +
removeResponse.attributes());

            // Example 5: Get the current value of a list attribute
            System.out.println("\nExample 5: Getting the current value of a list
attribute");
            List<AttributeValue> currentList = getListAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentList != null) {
                System.out.println("Current list attribute:");
                for (int i = 0; i < currentList.size(); i++) {
                    System.out.println("  [" + i + "]: " + currentList.get(i).s());
                }
            } else {
                System.out.println("List attribute not found");
            }

            // Explain list operations
            System.out.println("\nKey points about DynamoDB list operations:");
            System.out.println("1. Lists are ordered collections of attributes");
            System.out.println("2. Use list_append to add items to a list");
            System.out.println("3. To append items, use list_append(existingList,
newItems)");
            System.out.println("4. To prepend items, use list_append(newItems,
existingList)");
            System.out.println("5. Use index notation (list[0]) to access or update
specific elements");
            System.out.println("6. Use REMOVE to delete elements from a list");
            System.out.println("7. List indices are zero-based");
            System.out.println("8. Use if_not_exists to handle the case where the
list doesn't exist yet");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

Executar operações de mapa

O exemplo de código a seguir mostra como realizar operações de mapa no DynamoDB.

- Adicione e atualize atributos aninhados em estruturas de mapa.
- Remova campos específicos dos mapas.
- Trabalhe com atributos de mapa profundamente aninhados.

SDK para Java 2.x

Demonstre as operações do mapa usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates a map attribute that may not exist.
     *
     * <p>This method demonstrates how to safely update a map attribute
     * by using if_not_exists to handle the case where the map doesn't exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to update
     * @param value The value to set
```

```
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateMapAttributeSafe(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey,
        AttributeValue value) {

        // A single UpdateExpression can't reference both a map and a path inside
        // that same map (for example "#mapName" and "#mapName.#mapKey"): DynamoDB
        // rejects overlapping document paths with a ValidationException. Instead,
        // perform the update in two steps.

        // Step 1: create the map with an empty value if it doesn't already exist.
        dynamoDbClient.updateItem(UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = if_not_exists(#mapName, :emptyMap)")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(
                ":emptyMap", AttributeValue.builder().m(new HashMap<>()).build()))
            .build());

        // Step 2: set the key within the now-guaranteed-to-exist map.
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName.#mapKey = :value")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds an attribute to a nested map.
```

```
     *
     * <p>This method demonstrates how to update a nested attribute without
     * overwriting the entire map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param path The path to the nested attribute as a list
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNestedMap(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> path,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < path.size(); i++) {
            expressionAttributeNames.put("#attr" + i, path.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < path.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();
```

```
        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes an attribute from a map.
     *
     * <p>This method demonstrates how to remove a specific attribute from a map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeMapAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #mapName.#mapKey")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Creates a map with multiple attributes in a single operation.
     *
     * <p>This method demonstrates how to create a map with multiple attributes
```

```
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param attributes The attributes to set in the map
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createMapWithAttributes(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        Map<String, AttributeValue> attributes) {

        // Create a map value from the attributes
        AttributeValue mapValue = AttributeValue.builder().m(attributes).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = :mapValue")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(":mapValue", mapValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a map attribute.
     *
     * <p>Helper method to retrieve the current value of a map attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param mapName The name of the map attribute
     * @return The map attribute value or null if not found
```

```
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getMapAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String mapName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(mapName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the map attribute if it exists, otherwise null
            if (response.item() != null && response.item().containsKey(mapName)) {
                return response.item().get(mapName).m();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get map attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Exemplo de uso de operações de mapa com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating map operations in DynamoDB");

        try {
```

```
            // Example 1: Create a map with multiple attributes
            System.out.println("\nExample 1: Creating a map with multiple
attributes");
            Map<String, AttributeValue> productDetails = new HashMap<>();
            productDetails.put("Color", AttributeValue.builder().s("Red").build());
            productDetails.put("Weight", AttributeValue.builder().n("2.5").build());
            productDetails.put(
                "Dimensions", AttributeValue.builder().s("10x20x5").build());

            UpdateItemResponse createResponse =
                createMapWithAttributes(dynamoDbClient, tableName, key, "Details",
productDetails);

            System.out.println("Created map attribute: " +
createResponse.attributes());

            // Example 2: Update a specific attribute in a map
            System.out.println("\nExample 2: Updating a specific attribute in a
map");
            UpdateItemResponse updateResponse = updateMapAttributeSafe(
                dynamoDbClient,
                tableName,
                key,
                "Details",
                "Color",
                AttributeValue.builder().s("Blue").build());

            System.out.println("Updated map attribute: " +
updateResponse.attributes());

            // Example 3: Add an attribute to a nested map
            System.out.println("\nExample 3: Adding an attribute to a nested map");
            UpdateItemResponse nestedResponse = addToNestedMap(
                dynamoDbClient,
                tableName,
                key,
                List.of("Specifications", "Technical", "Resolution"),
                AttributeValue.builder().s("1920x1080").build());

            System.out.println("Added to nested map: " +
nestedResponse.attributes());

            // Example 4: Remove an attribute from a map
            System.out.println("\nExample 4: Removing an attribute from a map");
```

```
            UpdateItemResponse removeResponse =
                removeMapAttribute(dynamoDbClient, tableName, key, "Details",
"Dimensions");

            System.out.println("Updated map after removal: " +
removeResponse.attributes());

            // Example 5: Get the current value of a map attribute
            System.out.println("\nExample 5: Getting the current value of a map
attribute");
            Map<String, AttributeValue> currentMap = getMapAttribute(dynamoDbClient,
tableName, key, "Details");

            if (currentMap != null) {
                System.out.println("Current map attribute:");
                for (Map.Entry<String, AttributeValue> entry :
currentMap.entrySet()) {
                    System.out.println("  " + entry.getKey() + ": " +
entry.getValue());
                }
            } else {
                System.out.println("Map attribute not found");
            }

            // Explain map operations
            System.out.println("\nKey points about DynamoDB map operations:");
            System.out.println("1. Maps are unordered collections of name-value
pairs");
            System.out.println("2. Use dot notation (map.key) to access or update
specific attributes");
            System.out.println("3. You can update individual attributes without
overwriting the entire map");
            System.out.println("4. Maps can be nested to create complex data
structures");
            System.out.println("5. Use REMOVE to delete attributes from a map");
            System.out.println("6. You can create a map with multiple attributes in
a single operation");
            System.out.println("7. Map keys are case-sensitive");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

Executar operações de conjunto

O exemplo de código a seguir mostra como realizar operações de conjunto no DynamoDB.

- Adicione elementos a um atributo de conjunto.
- Remova elementos de um atributo de conjunto.
- Use as operações ADD e DELETE com conjuntos.

SDK para Java 2.x

Demonstre as operações de conjunto usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

    /**
     * Adds values to a string set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a string set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
```

```
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToStringSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<String> valuesToAdd) {

        // Create a string set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ss(valuesToAdd).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds values to a number set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a number set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNumberSet(
        DynamoDbClient dynamoDbClient,
```

```
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<Number> valuesToAdd) {

        // Convert numbers to strings for DynamoDB
        Set<String> stringValues = new HashSet<>();
        for (Number value : valuesToAdd) {
            stringValues.add(value.toString());
        }

        // Create a number set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ns(stringValues).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes values from a set attribute.
     *
     * <p>This method demonstrates how to use the DELETE operation to remove values
     * from a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToRemove The values to remove from the set
     * @param isNumberSet Whether the set is a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
```

```
    public static UpdateItemResponse removeFromSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> valuesToRemove,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("DELETE #setAttr :valuesToRemove")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToRemove", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
```

```
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Checks if a value exists in a set attribute.
     *
     * <p>This method demonstrates how to use the contains function to check
     * if a value exists in a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to check
     * @param setAttributeName The name of the set attribute
     * @param valueToCheck The value to check for
     * @return Map containing the result of the check
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> checkIfValueInSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        String valueToCheck) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters with a condition expression
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #tempAttr = :tempVal")
                .conditionExpression("contains(#setAttr, :valueToCheck)")
                .expressionAttributeNames(Map.of("#setAttr", setAttributeName,
"#tempAttr", "TempAttribute"))
                .expressionAttributeValues(Map.of(
                    ":valueToCheck",
AttributeValue.builder().s(valueToCheck).build(),
                    ":tempVal", AttributeValue.builder().s("TempValue").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Attempt the update operation
            dynamoDbClient.updateItem(request);
```

```
            // If we get here, the condition was met
            result.put("exists", true);
            result.put("message", "Value '" + valueToCheck + "' exists in the set");

            // Clean up the temporary attribute
            UpdateItemRequest cleanupRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("REMOVE #tempAttr")
                .expressionAttributeNames(Map.of("#tempAttr", "TempAttribute"))
                .build();

            dynamoDbClient.updateItem(cleanupRequest);

        } catch (DynamoDbException e) {
            if (e.getMessage().contains("ConditionalCheckFailed")) {
                // The condition was not met
                result.put("exists", false);
                result.put("message", "Value '" + valueToCheck + "' does not exist
in the set");
            } else {
                // Some other error occurred
                result.put("exists", false);
                result.put("message", "Error checking set: " + e.getMessage());
                result.put("error", e.getClass().getSimpleName());
            }
        }

        return result;
    }

    /**
     * Creates a set with multiple values in a single operation.
     *
     * <p>This method demonstrates how to create a set with multiple values
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param setValues The values to include in the set
```

```
    * @param isNumberSet Whether to create a number set (true) or string set
(false)
    * @return The response from DynamoDB
    * @throws DynamoDbException if an error occurs during the operation
    */
   public static UpdateItemResponse createSetWithValues(
       DynamoDbClient dynamoDbClient,
       String tableName,
       Map<String, AttributeValue> key,
       String setAttributeName,
       Set<?> setValues,
       boolean isNumberSet) {

       AttributeValue setValue;

       if (isNumberSet) {
           // Convert numbers to strings for DynamoDB
           Set<String> stringValues = new HashSet<>();
           for (Object value : setValues) {
               if (value instanceof Number) {
                   stringValues.add(value.toString());
               } else {
                   throw new IllegalArgumentException("Values must be numbers for a
number set");
               }
           }

           setValue = AttributeValue.builder().ns(stringValues).build();
       } else {
           // Convert objects to strings for DynamoDB
           Set<String> stringValues = new HashSet<>();
           for (Object value : setValues) {
               stringValues.add(value.toString());
           }

           setValue = AttributeValue.builder().ss(stringValues).build();
       }

       // Define the update parameters
       UpdateItemRequest request = UpdateItemRequest.builder()
           .tableName(tableName)
           .key(key)
           .updateExpression("SET #setAttr = :setValue")
           .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
```

```
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a set attribute.
     *
     * <p>Helper method to retrieve the current value of a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param setAttributeName The name of the set attribute
     * @return The set attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static AttributeValue getSetAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String setAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(setAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the set attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(setAttributeName)) {
                return response.item().get(setAttributeName);
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
```

```
                .message("Failed to get set attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Exemplo de uso de operações definidas com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating set operations in DynamoDB");

        try {
            // Example 1: Create a string set with multiple values
            System.out.println("\nExample 1: Creating a string set with multiple
values");
            Set<String> tags = new HashSet<>();
            tags.add("Electronics");
            tags.add("Gadget");
            tags.add("Smartphone");

            UpdateItemResponse createResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Tags", tags, false // Not a number
set
                );

            System.out.println("Created set attribute: " +
createResponse.attributes());

            // Example 2: Add values to a string set
            System.out.println("\nExample 2: Adding values to a string set");
            Set<String> additionalTags = new HashSet<>();
            additionalTags.add("Mobile");
            additionalTags.add("Wireless");

            UpdateItemResponse addResponse = addToStringSet(dynamoDbClient,
tableName, key, "Tags", additionalTags);
```

```
            System.out.println("Updated set attribute: " +
addResponse.attributes());

            // Example 3: Create a number set with multiple values
            System.out.println("\nExample 3: Creating a number set with multiple
values");
            Set<Number> ratings = new HashSet<>();
            ratings.add(4);
            ratings.add(5);
            ratings.add(4.5);

            UpdateItemResponse createNumberSetResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Ratings", ratings, true // Is a
number set
                );

            System.out.println("Created number set attribute: " +
createNumberSetResponse.attributes());

            // Example 4: Add values to a number set
            System.out.println("\nExample 4: Adding values to a number set");
            Set<Number> additionalRatings = new HashSet<>();
            additionalRatings.add(3.5);
            additionalRatings.add(4.2);

            UpdateItemResponse addNumberResponse =
                addToNumberSet(dynamoDbClient, tableName, key, "Ratings",
additionalRatings);

            System.out.println("Updated number set attribute: " +
addNumberResponse.attributes());

            // Example 5: Remove values from a set
            System.out.println("\nExample 5: Removing values from a set");
            Set<String> tagsToRemove = new HashSet<>();
            tagsToRemove.add("Gadget");

            UpdateItemResponse removeResponse = removeFromSet(
                dynamoDbClient, tableName, key, "Tags", tagsToRemove, false // Not a
number set
                );

            System.out.println("Updated set after removal: " +
removeResponse.attributes());
```

```
            // Example 6: Check if a value exists in a set
            System.out.println("\nExample 6: Checking if a value exists in a set");
            Map<String, Object> checkResult = checkIfValueInSet(dynamoDbClient,
tableName, key, "Tags", "Electronics");

            System.out.println("Check result: " + checkResult.get("message"));

            // Example 7: Get the current value of a set attribute
            System.out.println("\nExample 7: Getting the current value of a set
attribute");
            AttributeValue currentStringSet = getSetAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentStringSet != null && currentStringSet.ss() != null) {
                System.out.println("Current string set values: " +
currentStringSet.ss());
            } else {
                System.out.println("String set attribute not found");
            }

            AttributeValue currentNumberSet = getSetAttribute(dynamoDbClient,
tableName, key, "Ratings");

            if (currentNumberSet != null && currentNumberSet.ns() != null) {
                System.out.println("Current number set values: " +
currentNumberSet.ns());
            } else {
                System.out.println("Number set attribute not found");
            }

            // Explain set operations
            System.out.println("\nKey points about DynamoDB set operations:");
            System.out.println(
                "1. DynamoDB supports three set types: string sets (SS), number sets
(NS), and binary sets (BS)");
            System.out.println("2. Sets can only contain elements of the same
type");
            System.out.println("3. Use ADD to add elements to a set");
            System.out.println("4. Use DELETE to remove elements from a set");
            System.out.println("5. Sets automatically remove duplicate values");
            System.out.println("6. Sets are unordered collections");
            System.out.println("7. Use the contains function to check if a value
exists in a set");
```

```
            System.out.println("8. You can create a set with multiple values in a
 single operation");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("Creating an Amazon DynamoDB table named " + tableName
```

```
            + " with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Adding multiple records into the " + tableName
            + " table using a batch command.");
        putRecordBatch(ddb);

        // Update multiple movies by using the BatchExecute statement.
        String title1 = "Star Wars";
        int year1 = 1977;
        String title2 = "Wizard of Oz";
        int year2 = 1939;

        System.out.println("Query two movies.");
        getBatch(ddb, tableName, title1, title2, year1, year2);

        System.out.println("Updating multiple records using a batch command.");
        updateTableItemBatch(ddb);

        System.out.println("Deleting multiple records using a batch command.");
        deleteItemBatch(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static boolean getBatch(DynamoDbClient ddb, String tableName, String
title1, String title2, int year1, int year2) {
        String getBatch = "SELECT * FROM " + tableName + " WHERE title = ? AND year
= ?";

        List<BatchStatementRequest> statements = new ArrayList<>();
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title1).build(),
                AttributeValue.builder().n(String.valueOf(year1)).build())
            .build());
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title2).build(),
                AttributeValue.builder().n(String.valueOf(year2)).build())
            .build());
```

```
        BatchExecuteStatementRequest batchExecuteStatementRequest =
BatchExecuteStatementRequest.builder()
            .statements(statements)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchExecuteStatementRequest);
            if (!response.responses().isEmpty()) {
                response.responses().forEach(r -> {
                    System.out.println(r.item().get("title") + "\\t" +
r.item().get("year"));
                });
                return true;
            } else {
                System.out.println("Couldn't find either " + title1 + " or " +
title2 + ".");
                return false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            return false;
        }
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
```

```
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse = dbWaiter
                .waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
```

```
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n("1977")
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("Star Wars")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parameters)
                .build();

            // Set data for Movie 2.
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                .n("1939")
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("Wizard of Oz")
                .build();

            AttributeValue attMovie2B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie2.add(attMovie2);
            parametersMovie2.add(attMovie2A);
            parametersMovie2.add(attMovie2B);

            BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
```

```
                .statement(sqlStatement)
                .parameters(parametersMovie2)
                .build();

            // Set data for Movie 3.
            List<AttributeValue> parametersMovie3 = new ArrayList<>();
            AttributeValue attMovie3 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie3A = AttributeValue.builder()
                .s("My Movie 3")
                .build();

            AttributeValue attMovie3B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie3.add(attMovie3);
            parametersMovie3.add(attMovie3A);
            parametersMovie3.add(attMovie3B);

            BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie3)
                .build();

            // Add all three movies to the list.
            List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
            myBatchStatementList.add(statementRequestMovie1);
            myBatchStatementList.add(statementRequestMovie2);
            myBatchStatementList.add(statementRequestMovie3);

            BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                .statements(myBatchStatementList)
                .build();

            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Added new movies using a batch command.");
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "UPDATE MoviesPartiQBatch SET info = 'directors\":
[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Update three records.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Update record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
```

```
            .build();

        // Update record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);
        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }
```

```
    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Specify record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Specify record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();
```

```
        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);

        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
 ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }
}
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK for Java 2.x da API.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        String fileName = "../../../resources/sample_files/movies.json";
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(
            "******* Creating an Amazon DynamoDB table named MoviesPartiQ with a key
 named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("Getting data from the MoviesPartiQ table.");
        getItem(ddb);

        System.out.println("Putting a record into the MoviesPartiQ table.");
        putRecord(ddb);

        System.out.println("Updating a record.");
        updateTableItem(ddb);

        System.out.println("Querying the movies released in 2013.");
        queryTable(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());
```

```
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //Scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        List<AttributeValue> parameters = new ArrayList<>();
        while (iter.hasNext()) {

            // Add 200 movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf(year))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s(title)
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s(info)
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            // Insert the movie into the Amazon DynamoDB table.
```

```
            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added Movie " + title);

            parameters.remove(att1);
            parameters.remove(att2);
            parameters.remove(att3);
            t++;
        }
    }

    public static void getItem(DynamoDbClient ddb) {

        String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n("2012")
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The Perks of Being a Wallflower")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();
```

```
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2020"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added new movie.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItem(DynamoDbClient ddb) {

        String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The East")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
 ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }

    private static void processResults(ExecuteStatementResponse
 executeStatementResult) {
        System.out.println("ExecuteStatement successful: " +
 executeStatementResult.toString());
    }
}
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK for Java 2.x da API.

Consultar uma tabela usando um índice secundário global

O exemplo de código a seguir mostra como consultar uma tabela usando um índice secundário global.

- Consulte uma tabela do DynamoDB usando a respectiva chave primária.
- Consulte um índice secundário global (GSI) para obter padrões de acesso alternativos.
- Compare consultas de tabela e consultas de GSI.

SDK para Java 2.x

Consulte uma tabela do DynamoDB usando sua chave primária e um Índice Secundário Global (GSI) com. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
 response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
```

```
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
```

```
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
+ " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

Compare a consulta direta de uma tabela com a consulta de um GSI com. AWS SDK for Java 2.x

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <basePartitionKeyName> <basePartitionKeyValue>
<gsiName> <gsiPartitionKeyName> <gsiPartitionKeyValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    basePartitionKeyName - The name of the base table partition key
attribute.
                    basePartitionKeyValue - The value of the base table partition
key to query.
                    gsiName - The name of the Global Secondary Index.
                    gsiPartitionKeyName - The name of the GSI partition key
attribute.
                    gsiPartitionKeyValue - The value of the GSI partition key to
query.
```

```
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String basePartitionKeyName = args[1];
        final String basePartitionKeyValue = args[2];
        final String gsiName = args[3];
        final String gsiPartitionKeyName = args[4];
        final String gsiPartitionKeyValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        try (DynamoDbClient ddb = DynamoDbClient.builder().region(region).build()) {
            final QueryTableAndGSI queryHelper = new QueryTableAndGSI(ddb);

            // Query the base table
            System.out.println("Querying base table where " + basePartitionKeyName +
" = " + basePartitionKeyValue);
            final QueryResponse tableResponse =
                queryHelper.queryTable(tableName, basePartitionKeyName,
basePartitionKeyValue);

            System.out.println("Found " + tableResponse.count() + " items in base
table:");
            tableResponse.items().forEach(item -> System.out.println(item));

            // Query the GSI
            System.out.println(
                "\nQuerying GSI '" + gsiName + "' where " + gsiPartitionKeyName + "
= " + gsiPartitionKeyValue);
            final QueryResponse gsiResponse =
                queryHelper.queryGlobalSecondaryIndex(tableName, gsiName,
gsiPartitionKeyName, gsiPartitionKeyValue);

            System.out.println("Found " + gsiResponse.count() + " items in GSI:");
            gsiResponse.items().forEach(item -> System.out.println(item));

            // Explain the differences between querying a table and a GSI
```

```
            System.out.println("\nKey differences between querying a table and a
GSI:");
            System.out.println("1. When querying a GSI, you must specify the
indexName parameter");
            System.out.println("2. GSIs may not contain all attributes from the base
table (projection)");
            System.out.println("3. GSIs consume read capacity units from the GSI's
capacity, not the base table's");
            System.out.println("4. GSIs may have eventually consistent data (cannot
use ConsistentRead=true)");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table or index not found: " + e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar uma tabela usando uma condição begins_with

O exemplo de código a seguir mostra como consultar uma tabela usando uma condição begins_with.

- Use a função begins_with em uma expressão de condição de chave.
- Filtre itens com base em um padrão de prefixo na chave de classificação.

SDK para Java 2.x

Consulte uma tabela do DynamoDB usando uma condição begins_with na chave de classificação por meio do AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

```
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithBeginsWithCondition(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String sortKeyName,
        final String sortKeyPrefix) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Sort key name", sortKeyName);
        CodeSampleUtils.validateStringParameter("Sort key prefix", sortKeyPrefix);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, sortKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_SK_PREFIX,
            AttributeValue.builder().s(sortKeyPrefix).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
```

```
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with begins_with condition successful.
Found {0} items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with begins_with condition",
e);
            throw e;
        }
    }
```

Demonstre o uso de begins_with com diferentes comprimentos de prefixo por meio do AWS SDK for Java 2.x.

```
    public static void main(String[] args) {
        try {
            CodeSampleUtils.BeginsWithQueryConfig config =
CodeSampleUtils.BeginsWithQueryConfig.fromArgs(args);
            LOGGER.log(Level.INFO, "Querying items where {0} = {1} and {2} begins
with ''{3}'", new Object[] {
                config.getPartitionKeyName(),
                config.getPartitionKeyValue(),
                config.getSortKeyName(),
                config.getSortKeyPrefix()
            });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new BeginsWithQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withSortKeyName(config.getSortKeyName())
                .withSortKeyPrefix(config.getSortKeyPrefix())
                .withRegion(config.getRegion())
                .execute();
```

```
            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> LOGGER.info(item.toString()));

            // Demonstrate with a different prefix
            if (!config.getSortKeyPrefix().isEmpty()) {
                String shorterPrefix = config.getSortKeyPrefix()
                    .substring(0, Math.max(1, config.getSortKeyPrefix().length() /
2));
                LOGGER.log(Level.INFO, "\nNow querying with a shorter prefix:
''{0}'''", shorterPrefix);

                final QueryResponse response2 = new BeginsWithQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withSortKeyName(config.getSortKeyName())
                    .withSortKeyPrefix(shorterPrefix)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with shorter prefix:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }
        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            printUsage();
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found", e);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error", e);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error", e);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consulte uma tabela usando um intervalo de datas

O exemplo de código a seguir mostra como consultar uma tabela usando um intervalo de datas na chave de classificação.

- Consulte itens em um intervalo de datas específico.
- Use operadores de comparação em chaves de classificação formatadas por data.

SDK para Java 2.x

Consulte uma tabela do DynamoDB em busca de itens dentro de um intervalo de datas com. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
```

```
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
```

```
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

Demonstra como consultar uma tabela do DynamoDB com filtragem de intervalo de datas.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <dateKeyName>
<startDate> <endDate> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    dateKeyName - The name of the date attribute to filter on.
                    startDate - The start date for the range query (YYYY-MM-DD).
                    endDate - The end date for the range query (YYYY-MM-DD).
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        try {
            // Parse command line arguments into a config object
            CodeSampleUtils.DateRangeQueryConfig config =
CodeSampleUtils.DateRangeQueryConfig.fromArgs(args);

            LOGGER.log(
                Level.INFO, "Querying items from {0} to {1}", new Object[]
{config.getStartDate(), config.getEndDate()
                });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DateRangeQueryBuilder()
```

```
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withDateKeyName(config.getDateKeyName())
                .withStartDate(config.getStartDate())
                .withEndDate(config.getEndDate())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> {
                LOGGER.info(item.toString());

                // Extract and display the date attribute for clarity
                if (item.containsKey(config.getDateKeyName())) {
                    LOGGER.log(
                        Level.INFO,
                        "  Date attribute: {0}",
                        item.get(config.getDateKeyName()).s());
                }
            });

            // Demonstrate with a different date range
            LocalDate narrowerStartDate = config.getStartDate().plusDays(1);
            LocalDate narrowerEndDate = config.getEndDate().minusDays(1);

            if (!narrowerStartDate.isAfter(narrowerEndDate)) {
                LOGGER.log(Level.INFO, "\nNow querying with a narrower date range:
{0} to {1}", new Object[] {
                    narrowerStartDate, narrowerEndDate
                });

                final QueryResponse response2 = new DateRangeQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withDateKeyName(config.getDateKeyName())
                    .withStartDate(narrowerStartDate)
                    .withEndDate(narrowerEndDate)
                    .withRegion(config.getRegion())
                    .execute();
```

```
                LOGGER.log(Level.INFO, "Found {0} items with narrower date range:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }

            LOGGER.info("\nNote: When storing dates in DynamoDB:");
            LOGGER.info("1. Use ISO format (YYYY-MM-DD) for lexicographical
ordering");
            LOGGER.info("2. Use the BETWEEN operator for inclusive date range
queries");
            LOGGER.info("3. Consider using ISO-8601 format for timestamps with time
components");

        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error: {0}", e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error: {0}", e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar uma tabela com uma expressão de filtro complexa

O exemplo de código a seguir mostra como consultar uma tabela com uma expressão de filtro complexa.

- Aplique expressões de filtro complexas aos resultados da consulta.
- Combine várias condições usando operadores lógicos.
- Filtre itens com base em atributos não chave.

SDK para Java 2.x

Consulte uma tabela do DynamoDB com uma expressão de filtro complexa usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
 priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
 categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
 Double.MAX_VALUE);
```

```
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar uma tabela com uma expressão de filtro dinâmica

O exemplo de código a seguir mostra como consultar uma tabela com uma expressão de filtro dinâmica.

- Crie expressões de filtro dinamicamente no tempo de execução.
- Crie condições de filtro com base na entrada do usuário ou no estado da aplicação.
- Adicione ou remova critérios de filtro condicionalmente.

SDK para Java 2.x

Consulte uma tabela do DynamoDB com uma expressão de filtro construída dinamicamente usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;
```

```
        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }
```

Demonstra como usar expressões de filtro dinâmico com AWS SDK for Java 2.x.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
```

```
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
```

```
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar uma tabela com uma expressão de filtro e limite

O exemplo de código a seguir mostra como consultar uma tabela com uma expressão de filtro e limite.

- Aplique expressões de filtro aos resultados da consulta com um limite nos itens avaliados.
- Entenda como o limite afeta os resultados da consulta filtrada.
- Controle o número máximo de itens processados em uma consulta.

SDK para Java 2.x

Consulte uma tabela do DynamoDB com uma expressão de filtro e limite de uso. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
 filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
 filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
 filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
```

```
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar uma tabela com atributos aninhados

O exemplo de código a seguir mostra como consultar uma tabela com atributos aninhados.

- Acesse e filtre por atributos aninhados nos itens do DynamoDB.

- Use expressões de caminho de documento para se referir a elementos aninhados.

SDK para Java 2.x

Consulte uma tabela do DynamoDB com atributos aninhados usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithNestedAttributes(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String nestedPath,
        final String nestedAttr,
        final String nestedValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Nested path", nestedPath);
        CodeSampleUtils.validateStringParameter("Nested attribute", nestedAttr);
        CodeSampleUtils.validateStringParameter("Nested value", nestedValue);

        // Split the nested path into components
        final String[] pathComponents = nestedPath.split("\\.");

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Build the nested attribute reference using document path notation
        final StringBuilder nestedAttributeRef = new StringBuilder();
        for (int i = 0; i < pathComponents.length; i++) {
            final String aliasName = "#n" + i;
```

```
            expressionAttributeNames.put(aliasName, pathComponents[i]);

            if (i > 0) {
                nestedAttributeRef.append(".");
            }
            nestedAttributeRef.append(aliasName);
        }

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_NESTED,
            AttributeValue.builder().s(nestedValue).build());

        // Create the filter expression using the nested attribute reference
        final String filterExpression = nestedAttributeRef + " = :nestedValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query with nested attribute filter successful. Found
" + response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with nested attribute filter: " +
e.getMessage());
            throw e;
        }
```

```
    }
```

Demonstra como consultar uma tabela do DynamoDB com atributos aninhados.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <nestedPath>
<nestedAttr> <nestedValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    nestedPath - The path to the nested map attribute (e.g.,
"address").
                    nestedAttr - The name of the nested attribute (e.g., "city").
                    nestedValue - The value to filter by (e.g., "Seattle").
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String nestedPath = args[3];
        final String nestedAttr = args[4];
        final String nestedValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        System.out.println("Querying items where " + partitionKeyName + " = " +
partitionKeyValue + " and " + nestedPath
            + "." + nestedAttr + " = " + nestedValue);

        try {
            // Using the builder pattern to create and execute the query
```

```
            final QueryResponse response = new NestedAttributeQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withNestedPath(nestedPath)
                .withNestedAttribute(nestedAttr)
                .withNestedValue(nestedValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> {
                System.out.println(item);

                // Extract and display the nested attribute for clarity
                if (item.containsKey(nestedPath) && item.get(nestedPath).hasM()) {
                    Map<String, AttributeValue> nestedMap =
item.get(nestedPath).m();
                    if (nestedMap.containsKey(nestedAttr)) {
                        System.out.println("  Nested attribute " + nestedPath + "."
+ nestedAttr + ": "
                            + formatAttributeValue(nestedMap.get(nestedAttr)));
                    }
                }
            });

            System.out.println("\nNote: When working with nested attributes in
DynamoDB:");
            System.out.println("1. Use dot notation in filter expressions to access
nested attributes");
            System.out.println("2. Use expression attribute names for each component
of the path");
            System.out.println("3. Check if the nested attribute exists before
accessing it");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
```

```
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar uma tabela com paginação

O exemplo de código a seguir mostra como consultar uma tabela com paginação.

- Implemente a paginação para os resultados da consulta do DynamoDB.
- Use o LastEvaluatedKey para recuperar as páginas subsequentes.
- Controle o número de itens por página com o parâmetro Limit.

SDK para Java 2.x

Consulte uma tabela do DynamoDB com paginação usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
```

```
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);
```

```
                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
" + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

Demonstra como consultar uma tabela do DynamoDB com paginação.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> [pageSize]
[region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
```

```
                    pageSize (optional) - The maximum number of items to return per
page. (Default: 10)
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final int pageSize = args.length > 3 ? Integer.parseInt(args[3]) : 10;
        final Region region = args.length > 4 ? Region.of(args[4]) :
Region.US_EAST_1;

        System.out.println("Querying items with pagination (page size: " + pageSize
+ ")");

        try {
            // Using the builder pattern to create and execute the query
            final List<Map<String, AttributeValue>> allItems = new
PaginationQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withPageSize(pageSize)
                .withRegion(region)
                .executeWithPagination();

            // Process the results
            System.out.println("\nSummary: Retrieved a total of " + allItems.size()
+ " items");

            // Display the first few items as a sample
            final int sampleSize = Math.min(5, allItems.size());
            if (sampleSize > 0) {
                System.out.println("\nSample of retrieved items (first " +
sampleSize + "):");
                for (int i = 0; i < sampleSize; i++) {
                    System.out.println(allItems.get(i));
                }
```

```
                if (allItems.size() > sampleSize) {
                    System.out.println("... and " + (allItems.size() - sampleSize) +
 " more items");
                }
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar uma tabela com leituras altamente consistentes

O exemplo de código a seguir mostra como consultar uma tabela com leituras altamente consistentes.

- Configure o nível de consistência das consultas do DynamoDB.
- Use leituras altamente consistentes para obter os dados mais atualizados.
- Entenda as vantagens e desvantagens entre consistência final e consistência forte.

SDK para Java 2.x

Consulte uma tabela do DynamoDB com consistência de leitura configurável usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
 response.count());
            return response;
```

```
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar itens com TTL

O exemplo de código a seguir mostra como consultar itens com TTL.

SDK para Java 2.x

Consulte a expressão filtrada para reunir itens TTL em uma tabela do DynamoDB usando. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Map;
import java.util.Optional;

        final QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
```

```
            final QueryResponse response = ddb.query(request);
            System.out.println("Query successful. Found " + response.count() + "
 items that have not expired yet.");

            // Print each item
            response.items().forEach(item -> {
                System.out.println("Item: " + item);
            });

            return 0;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Consultar tabelas usando padrões de data e hora

O exemplo de código a seguir mostra como consultar tabelas usando padrões de data e hora.

- Armazene e consulte date/time valores no DynamoDB.
- Implemente consultas de intervalo de datas usando chaves de classificação.
- Formate strings de data para obter uma consulta eficaz.

SDK para Java 2.x

Consulte usando intervalos de datas em chaves de classificação com AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
```

```
import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
```

```
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
 items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
 e.getMessage());
            throw e;
        }
    }
```

Consulte usando variáveis de data e hora com. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTime(
```

```
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final String startDate,
        final String endDate) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateDateRangeParameters(dateKeyName, startDate,
endDate);
        CodeSampleUtils.validateDateFormat("Start date", startDate);
        CodeSampleUtils.validateDateFormat("End date", endDate);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate", AttributeValue.builder().s(startDate).build());
        expressionAttributeValues.put(
            ":endDate", AttributeValue.builder().s(endDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
items");
            return response;
```

```
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with date range: " + e.getMessage());
            throw e;
        }
    }
```

Consulte dentro dos intervalos de datas nos carimbos de data/hora de época do Unix com o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeEpoch(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final long startEpoch,
        final long endEpoch) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        CodeSampleUtils.validateEpochTimestamp("Start epoch", startEpoch);
        CodeSampleUtils.validateEpochTimestamp("End epoch", endEpoch);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
```

```
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate",
AttributeValue.builder().n(String.valueOf(startEpoch)).build());
        expressionAttributeValues.put(
            ":endDate",
AttributeValue.builder().n(String.valueOf(endEpoch)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with epoch timestamps: " +
e.getMessage());
            throw e;
        }
    }
```

Consulte dentro de intervalos de datas usando LocalDateTime objetos com AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeLocalDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDateTime startDateTime,
        final LocalDateTime endDateTime) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        if (startDateTime == null || endDateTime == null) {
            throw new IllegalArgumentException("Start and end LocalDateTime must not
 be null");
        }

        // Convert LocalDateTime to ISO-8601 strings in UTC with the correct format
        final String startDate =
 startDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);
        final String endDate =
 endDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);

        return queryWithDateTime(tableName, partitionKeyName, partitionKeyValue,
 dateKeyName, startDate, endDate);
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK for Java 2.x .

Como funciona a ordem da expressão de atualização

O exemplo de código a seguir mostra como funciona a ordem da expressão de atualização.

- Saiba como o DynamoDB processa expressões de atualização.
- Entenda como funciona a ordem das operações nas expressões de atualização.
- Evite resultados inesperados entendendo a avaliação da expressão.

SDK para Java 2.x

Demonstre a ordem das expressões de atualização usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Demonstrates the effect of update expression order.
     *
     * <p>This method shows how the order of operations in an update expression
     * affects the result of the update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the results of different update orders
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> demonstrateUpdateOrder(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
 key) {
```

```
        Map<String, Object> results = new HashMap<>();

        try {
            // Initialize the item with a counter
            UpdateItemRequest initRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :zero, OldCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            dynamoDbClient.updateItem(initRequest);

            // Example 1: SET first, then ADD
            UpdateItemRequest setFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value ADD OldCounter :increment")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse setFirstResponse =
dynamoDbClient.updateItem(setFirstRequest);
            results.put("setFirstResponse", setFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 2: ADD first, then SET
            UpdateItemRequest addFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("ADD Counter :increment SET OldCounter = :value")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();
```

```
            UpdateItemResponse addFirstResponse =
dynamoDbClient.updateItem(addFirstRequest);
            results.put("addFirstResponse", addFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 3: SET with multiple attributes
            UpdateItemRequest multiSetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value, OldCounter = Counter")
                .expressionAttributeValues(
                    Map.of(":value", AttributeValue.builder().n("10").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse multiSetResponse =
dynamoDbClient.updateItem(multiSetRequest);
            results.put("multiSetResponse", multiSetResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 4: SET with expression using the same attribute
            UpdateItemRequest selfReferenceRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = Counter + :increment, OldCounter =
Counter")
                .expressionAttributeValues(
                    Map.of(":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse selfReferenceResponse =
dynamoDbClient.updateItem(selfReferenceRequest);
            results.put("selfReferenceResponse", selfReferenceResponse);

            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
```

```
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Updates an item with SET first, then REMOVE.
     *
     * <p>This method demonstrates updating an item with SET operation first,
     * followed by a REMOVE operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithSetFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue REMOVE #removeAttr")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
```

```
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with SET first: " + e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with REMOVE first, then SET.
     *
     * <p>This method demonstrates updating an item with REMOVE operation first,
     * followed by a SET operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithRemoveFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #removeAttr SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
```

```
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with REMOVE first: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with all operation types in a specific order.
     *
     * <p>This method demonstrates using all operation types (SET, REMOVE, ADD,
DELETE)
     * in a specific order in a single update expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithAllOperationTypes(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #stringAttr = :stringVal, #mapAttr.#nestedAttr
= :nestedVal " + "REMOVE #oldAttr "
                + "ADD #counterAttr :increment "
                + "DELETE #stringSetAttr :stringSetVal")
            .expressionAttributeNames(Map.of(
                "#stringAttr", "StringAttribute",
                "#mapAttr", "MapAttribute",
                "#nestedAttr", "NestedAttribute",
                "#oldAttr", "OldAttribute",
                "#counterAttr", "CounterAttribute",
                "#stringSetAttr", "StringSetAttribute"))
            .expressionAttributeValues(Map.of(
```

```
                ":stringVal", AttributeValue.builder().s("New Value").build(),
                ":nestedVal", AttributeValue.builder().s("Nested Value").build(),
                ":increment", AttributeValue.builder().n("1").build(),
                ":stringSetVal", AttributeValue.builder().ss("Value1").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with all operation types: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Gets the current state of an item.
     *
     * <p>Helper method to retrieve the current state of an item.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @return The item or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getItem(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the get parameters
        GetItemRequest request =
            GetItemRequest.builder().tableName(tableName).key(key).build();

        // Perform the get operation
        try {
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the item if it exists, otherwise null
            return response.item();
```

```
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get item: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Exemplo de uso da ordem de atualização da expressão com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating update expression order in DynamoDB");

        try {
            // Example 1: Demonstrate update order effects
            System.out.println("\nExample 1: Demonstrating update order effects");
            Map<String, Object> orderResults =
demonstrateUpdateOrder(dynamoDbClient, tableName, key);

            if ((boolean) orderResults.get("success")) {
                System.out.println("SET first, then ADD:");
                System.out.println("  " + orderResults.get("setFirstResponse"));

                System.out.println("ADD first, then SET:");
                System.out.println("  " + orderResults.get("addFirstResponse"));

                System.out.println("SET with multiple attributes:");
                System.out.println("  " + orderResults.get("multiSetResponse"));

                System.out.println("SET with self-reference:");
                System.out.println("  " +
orderResults.get("selfReferenceResponse"));
            } else {
                System.out.println("Error: " + orderResults.get("error"));
            }

            // Example 2: Update with SET first, then REMOVE
```

```
            System.out.println("\nExample 2: Update with SET first, then REMOVE");
            UpdateItemResponse setFirstResponse = updateWithSetFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Active").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
setFirstResponse.attributes());

            // Example 3: Update with REMOVE first, then SET
            System.out.println("\nExample 3: Update with REMOVE first, then SET");
            UpdateItemResponse removeFirstResponse = updateWithRemoveFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Inactive").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
removeFirstResponse.attributes());

            // Example 4: Update with all operation types
            System.out.println("\nExample 4: Update with all operation types");
            UpdateItemResponse allOpsResponse =
updateWithAllOperationTypes(dynamoDbClient, tableName, key);

            System.out.println("Updated attributes: " +
allOpsResponse.attributes());

            // Example 5: Get the current state of the item
            System.out.println("\nExample 5: Current state of the item");
            Map<String, AttributeValue> item = getItem(dynamoDbClient, tableName,
key);

            if (item != null) {
                System.out.println("Item: " + item);
            } else {
                System.out.println("Item not found");
            }
```

```
            // Explain update expression order
            System.out.println("\nKey points about update expression order in
DynamoDB:");
            System.out.println("1. Update expressions are processed in this order:
SET, REMOVE, ADD, DELETE");
            System.out.println("2. Within each clause, operations are processed from
left to right");
            System.out.println("3. SET operations use the item state before any
updates in the expression");
            System.out.println("4. When an attribute is referenced multiple times,
the first operation wins");
            System.out.println("5. To reference a new value, split the update into
multiple operations");
            System.out.println("6. The order of clauses in the expression doesn't
change the evaluation order");
            System.out.println("7. For complex updates, consider using multiple
separate update operations");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

Atualizar a configuração de throughput a quente de uma tabela

O exemplo de código a seguir mostra como atualizar a configuração de throughput a quente de uma tabela.

SDK para Java 2.x

Atualize a configuração de throughput a quente em uma tabela existente do DynamoDB usando o AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndexUpdate;
```

```
import
 software.amazon.awssdk.services.dynamodb.model.UpdateGlobalSecondaryIndexAction;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Updates a DynamoDB table with warm throughput settings for both the table and
 a global secondary index.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to update
     * @param tableReadUnitsPerSecond Read units per second for the table
     * @param tableWriteUnitsPerSecond Write units per second for the table
     * @param globalSecondaryIndexName The name of the global secondary index to
 update
     * @param globalSecondaryIndexReadUnitsPerSecond Read units per second for the
 GSI
     * @param globalSecondaryIndexWriteUnitsPerSecond Write units per second for the
 GSI
     */
    public static void updateDynamoDBTable(
        final DynamoDbClient ddb,
        final String tableName,
        final Long tableReadUnitsPerSecond,
        final Long tableWriteUnitsPerSecond,
        final String globalSecondaryIndexName,
        final Long globalSecondaryIndexReadUnitsPerSecond,
        final Long globalSecondaryIndexWriteUnitsPerSecond) {

        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableReadUnitsPerSecond, tableWriteUnitsPerSecond);
        final WarmThroughput gsiWarmThroughput =
            buildWarmThroughput(globalSecondaryIndexReadUnitsPerSecond,
 globalSecondaryIndexWriteUnitsPerSecond);
```

```
        final GlobalSecondaryIndexUpdate globalSecondaryIndexUpdate =
GlobalSecondaryIndexUpdate.builder()
            .update(UpdateGlobalSecondaryIndexAction.builder()
                .indexName(globalSecondaryIndexName)
                .warmThroughput(gsiWarmThroughput)
                .build())
            .build();

        final UpdateTableRequest request = UpdateTableRequest.builder()
            .tableName(tableName)
            .globalSecondaryIndexUpdates(globalSecondaryIndexUpdate)
            .warmThroughput(tableWarmThroughput)
            .build();

        try {
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }

        System.out.println(SUCCESS_MESSAGE);
    }
```

- Para obter detalhes da API, consulte UpdateTable em Referência AWS SDK for Java 2.x da API.

Atualiza a TTL de um item

O exemplo de código a seguir mostra como atualizar a TTL de um item.

SDK para Java 2.x

Atualize a TTL em um item do DynamoDB existente em uma tabela.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

    public UpdateItemResponse updateItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
 sortKeyValue) {
        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = new HashMap<>();
        keyMap.put(PRIMARY_KEY_ATTR,
 AttributeValue.builder().s(primaryKeyValue).build());
        keyMap.put(SORT_KEY_ATTR, AttributeValue.builder().s(sortKeyValue).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build());
        expressionAttributeValues.put(
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final UpdateItemResponse response = dynamoDbClient.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
```

```
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar Step Functions para invocar funções do Lambda

O exemplo de código a seguir mostra como criar uma máquina de AWS Step Functions estados que invoca AWS Lambda funções em sequência.

SDK para Java 2.x

Mostra como criar um fluxo de trabalho AWS sem servidor usando AWS Step Functions e o. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Funções de etapa

Usar operações de contador atômico

O exemplo de código a seguir mostra como usar operações de contador atômico no DynamoDB.

- Incremente os contadores atomicamente usando as operações ADD e SET.
- Incremente seguramente contadores que talvez não existam.
- Implemente um bloqueio positivo para operações de contador.

SDK para Java 2.x

Demonstre operações de contador atômico usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
```

```
    /**
     * Increments a counter using the ADD operation.
     *
     * <p>This method demonstrates how to use the ADD operation to atomically
     * increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithAdd(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #counterName :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter using the SET operation.
     *
     * <p>This method demonstrates how to use the SET operation with an expression
     * to increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
```

```
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter safely, handling the case where the counter doesn't
exist yet.
     *
     * <p>This method demonstrates how to use if_not_exists to safely increment a
counter
     * that may not exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
```

```
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = if_not_exists(#counterName, :zero)
+ :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build(),
                ":zero", AttributeValue.builder().n("0").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Decrements a counter safely, ensuring it doesn't go below zero.
     *
     * <p>This method demonstrates how to use a condition expression to safely
     * decrement a counter without going below zero.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param decrementValue The value to decrement by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation or if the
counter would go below zero
     */
    public static UpdateItemResponse decrementCounterSafely(
```

```
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int decrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName - :decrement")
            .conditionExpression("#counterName >= :decrement")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":decrement",
                AttributeValue.builder().n(String.valueOf(decrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Compares the ADD and SET approaches for incrementing counters.
     *
     * <p>This method demonstrates the differences between using ADD and SET
     * for incrementing counters in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the comparison results
     */
    public static Map<String, Object> compareAddVsSet(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Reset counters to ensure a fair comparison
            UpdateItemRequest resetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
```

```
                .key(key)
                .updateExpression("SET AddCounter = :zero, SetCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .build();

            dynamoDbClient.updateItem(resetRequest);

            // Increment with ADD
            long addStartTime = System.nanoTime();
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "AddCounter", 1);
            long addEndTime = System.nanoTime();
            long addDuration = addEndTime - addStartTime;

            // Increment with SET
            long setStartTime = System.nanoTime();
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "SetCounter", 1);
            long setEndTime = System.nanoTime();
            long setDuration = setEndTime - setStartTime;

            // Record results
            results.put("addResponse", addResponse);
            results.put("setResponse", setResponse);
            results.put("addDuration", addDuration);
            results.put("setDuration", setDuration);
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Gets the current value of a counter attribute.
     *
     * <p>Helper method to retrieve the current value of a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
```

```
     * @param key The key of the item to get
     * @param counterName The name of the counter attribute
     * @return The counter value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Integer getCounterValue(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String counterName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(counterName)
            .build();

        // Perform the get operation
        GetItemResponse response = dynamoDbClient.getItem(request);

        // Return the counter value if it exists, otherwise null
        if (response.item() != null && response.item().containsKey(counterName)) {
            return Integer.parseInt(response.item().get(counterName).n());
        }

        return null;
    }
```

Exemplo de uso de operações de contador atômico com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating atomic counter operations in DynamoDB");

        try {
            // Example 1: Increment a counter using ADD
            System.out.println("\nExample 1: Incrementing a counter using ADD");
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "ViewCount", 1);
```

```
            System.out.println("Updated counter: " + addResponse.attributes());

            // Example 2: Increment a counter using SET
            System.out.println("\nExample 2: Incrementing a counter using SET");
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "LikeCount", 1);

            System.out.println("Updated counter: " + setResponse.attributes());

            // Example 3: Increment a counter safely
            System.out.println("\nExample 3: Incrementing a counter safely");
            UpdateItemResponse safeResponse = incrementCounterSafely(dynamoDbClient,
tableName, key, "ShareCount", 1);

            System.out.println("Updated counter: " + safeResponse.attributes());

            // Example 4: Decrement a counter safely
            System.out.println("\nExample 4: Decrementing a counter safely");
            try {
                UpdateItemResponse decrementResponse =
                    decrementCounterSafely(dynamoDbClient, tableName, key,
"InventoryCount", 1);

                System.out.println("Updated counter: " +
decrementResponse.attributes());
            } catch (DynamoDbException e) {
                if (e.getMessage().contains("ConditionalCheckFailed")) {
                    System.out.println("Cannot decrement counter below zero");
                } else {
                    throw e;
                }
            }

            // Example 5: Compare ADD vs SET
            System.out.println("\nExample 5: Comparing ADD vs SET");
            Map<String, Object> comparison = compareAddVsSet(dynamoDbClient,
tableName, key);

            if ((boolean) comparison.get("success")) {
                System.out.println("ADD duration: " + comparison.get("addDuration")
+ " ns");
                System.out.println("SET duration: " + comparison.get("setDuration")
+ " ns");
```

```
                System.out.println("ADD response: " +
comparison.get("addResponse"));
                System.out.println("SET response: " +
comparison.get("setResponse"));
            } else {
                System.out.println("Comparison failed: " + comparison.get("error"));
            }

            // Explain atomic counter operations
            System.out.println("\nKey points about DynamoDB atomic counter
operations:");
            System.out.println("1. Both ADD and SET can be used for atomic
counters");
            System.out.println("2. ADD is more concise for simple increments");
            System.out.println("3. SET with an expression is more flexible for
complex operations");
            System.out.println("4. Use if_not_exists to handle the case where the
counter doesn't exist yet");
            System.out.println("5. Use condition expressions to prevent counters
from going below zero");
            System.out.println("6. Atomic operations are guaranteed to be isolated
from other writes");
            System.out.println("7. ADD can only be used with number and set data
types");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK for Java 2.x da API.

Usar operações condicionais

O exemplo de código a seguir mostra como usar operações condicionais no DynamoDB.

- Implemente gravações condicionais para evitar a substituição de dados.
- Use expressões condicionais para impor regras de negócios.
- Lide com falhas de verificação condicional com tranquilidade.

SDK para Java 2.x

Demonstre operações condicionais usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Performs a conditional update on an item.
     *
     * <p>This method demonstrates how to use a condition expression to update an
 item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdate(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue,
        String updateAttribute,
        AttributeValue updateValue) {
```

```
        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of(
                    "#condAttr", conditionAttribute,
                    "#updateAttr", updateAttribute))
                .expressionAttributeValues(Map.of(
                    ":condVal", conditionValue,
                    ":updateVal", updateValue))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and update was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, update was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }
```

```
    /**
     * Performs a conditional delete on an item.
     *
     * <p>This method demonstrates how to use a condition expression to delete an
item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to delete
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalDelete(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the delete parameters
            DeleteItemRequest request = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of("#condAttr", conditionAttribute))
                .expressionAttributeValues(Map.of(":condVal", conditionValue))
                .returnValues(ReturnValue.ALL_OLD)
                .build();

            // Perform the delete operation
            DeleteItemResponse response = dynamoDbClient.deleteItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and delete was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
```

```
            result.put("success", false);
            result.put("message", "Condition was not met, delete was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Demonstrates optimistic locking using a version attribute.
     *
     * <p>This method shows how to implement optimistic locking by using a version
     * attribute that is incremented with each update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param versionAttribute The name of the version attribute
     * @return Map containing the operation result
     */
    public static Map<String, Object> optimisticLockingExample(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String versionAttribute) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Get the current version of the item
            GetItemRequest getRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .projectionExpression(versionAttribute)
                .build();

            GetItemResponse getResponse = dynamoDbClient.getItem(getRequest);

            // Check if the item exists
```

```
            if (getResponse.item() == null || !
getResponse.item().containsKey(versionAttribute)) {
                // Item doesn't exist or doesn't have a version attribute
                // Initialize with version 1
                UpdateItemRequest initRequest = UpdateItemRequest.builder()
                    .tableName(tableName)
                    .key(key)
                    .updateExpression("SET #verAttr = :newVer, #dataAttr = :data")
                    .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
 "#dataAttr", "Data"))
                    .expressionAttributeValues(Map.of(
                        ":newVer", AttributeValue.builder().n("1").build(),
                        ":data", AttributeValue.builder().s("Initial
 data").build()))
                    .returnValues(ReturnValue.UPDATED_NEW)
                    .build();

                UpdateItemResponse initResponse =
 dynamoDbClient.updateItem(initRequest);

                result.put("operation", "initialize");
                result.put("success", true);
                result.put("attributes", initResponse.attributes());

                return result;
            }

            // Get the current version number
            int currentVersion =
                Integer.parseInt(getResponse.item().get(versionAttribute).n());
            int newVersion = currentVersion + 1;

            // Update the item with a condition on the version
            UpdateItemRequest updateRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #verAttr = :newVer, #dataAttr = :newData")
                .conditionExpression("#verAttr = :curVer")
                .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
 "#dataAttr", "Data"))
                .expressionAttributeValues(Map.of(
                    ":curVer",
                        AttributeValue.builder()
                            .n(String.valueOf(currentVersion))
```

```
                            .build(),
                    ":newVer",

AttributeValue.builder().n(String.valueOf(newVersion)).build(),
                    ":newData",
                        AttributeValue.builder()
                            .s("Updated data at version " + newVersion)
                            .build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse updateResponse =
dynamoDbClient.updateItem(updateRequest);

            // Record success result
            result.put("operation", "update");
            result.put("success", true);
            result.put("oldVersion", currentVersion);
            result.put("newVersion", newVersion);
            result.put("attributes", updateResponse.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to version mismatch
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Version mismatch, another process may have
updated the item");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Performs a conditional update with multiple conditions.
     *
```

```
    * <p>This method demonstrates how to use multiple conditions in a condition
expression.
    *
    * @param dynamoDbClient The DynamoDB client
    * @param tableName The name of the DynamoDB table
    * @param key The key of the item to update
    * @param conditions Map of attribute names to values for conditions
    * @param updateAttribute The attribute to update
    * @param updateValue The new value to set
    * @return Map containing the operation result and status
    */
   public static Map<String, Object> conditionalUpdateWithMultipleConditions(
       DynamoDbClient dynamoDbClient,
       String tableName,
       Map<String, AttributeValue> key,
       Map<String, AttributeValue> conditions,
       String updateAttribute,
       AttributeValue updateValue) {

       Map<String, Object> result = new HashMap<>();

       try {
           // Build the condition expression and attribute names/values
           StringBuilder conditionExpression = new StringBuilder();
           Map<String, String> expressionAttributeNames = new HashMap<>();
           Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

           // Add update attribute
           expressionAttributeNames.put("#updateAttr", updateAttribute);
           expressionAttributeValues.put(":updateVal", updateValue);

           // Add conditions
           int i = 0;
           for (Map.Entry<String, AttributeValue> condition :
conditions.entrySet()) {
               String attrName = condition.getKey();
               AttributeValue attrValue = condition.getValue();

               String nameKey = "#cond" + i;
               String valueKey = ":val" + i;

               expressionAttributeNames.put(nameKey, attrName);
               expressionAttributeValues.put(valueKey, attrValue);
```

```
                // Add AND between conditions (except for the first one)
                if (i > 0) {
                    conditionExpression.append(" AND ");
                }

                conditionExpression.append(nameKey).append(" = ").append(valueKey);
                i++;
            }

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression(conditionExpression.toString())
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "All conditions were met and update was
performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "One or more conditions were not met, update was
not performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
```

```
    }
```

Exemplo de uso de operações condicionais com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating conditional operations in DynamoDB");

        try {
            // Example 1: Conditional update
            System.out.println("\nExample 1: Conditional update");
            Map<String, Object> updateResult = conditionalUpdate(
                dynamoDbClient,
                tableName,
                key,
                "InStock",
                AttributeValue.builder().bool(true).build(),
                "Status",
                AttributeValue.builder().s("Available").build());

            System.out.println("Update result: " + updateResult.get("message"));
            if ((boolean) updateResult.get("success")) {
                System.out.println("Updated attributes: " +
updateResult.get("attributes"));
            }

            // Example 2: Conditional delete
            System.out.println("\nExample 2: Conditional delete");
            Map<String, Object> deleteResult = conditionalDelete(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Discontinued").build());

            System.out.println("Delete result: " + deleteResult.get("message"));
            if ((boolean) deleteResult.get("success")) {
```

```
                System.out.println("Deleted item: " +
deleteResult.get("attributes"));
            }

            // Example 3: Optimistic locking
            System.out.println("\nExample 3: Optimistic locking");
            Map<String, Object> lockingResult =
optimisticLockingExample(dynamoDbClient, tableName, key, "Version");

            System.out.println("Optimistic locking result:");
            System.out.println("  Operation: " + lockingResult.get("operation"));
            System.out.println("  Success: " + lockingResult.get("success"));
            if (lockingResult.get("operation").equals("update") && (boolean)
lockingResult.get("success")) {
                System.out.println("  Old version: " +
lockingResult.get("oldVersion"));
                System.out.println("  New version: " +
lockingResult.get("newVersion"));
            }
            System.out.println("  Attributes: " + lockingResult.get("attributes"));

            // Example 4: Multiple conditions
            System.out.println("\nExample 4: Multiple conditions");
            Map<String, AttributeValue> conditions = new HashMap<>();
            conditions.put("Price", AttributeValue.builder().n("199.99").build());
            conditions.put("Category",
AttributeValue.builder().s("Electronics").build());

            Map<String, Object> multiConditionResult =
conditionalUpdateWithMultipleConditions(
                dynamoDbClient,
                tableName,
                key,
                conditions,
                "OnSale",
                AttributeValue.builder().bool(true).build());

            System.out.println("Multiple conditions result: " +
multiConditionResult.get("message"));
            if ((boolean) multiConditionResult.get("success")) {
                System.out.println("Updated attributes: " +
multiConditionResult.get("attributes"));
            }
```

```
            // Explain conditional operations
            System.out.println("\nKey points about DynamoDB conditional
operations:");
            System.out.println("1. Conditional operations only succeed if the
condition is met");
            System.out.println("2. ConditionalCheckFailedException is thrown when
the condition fails");
            System.out.println("3. No changes are made to the item if the condition
fails");
            System.out.println("4. Conditions can be used with update, delete, and
put operations");
            System.out.println("5. Multiple conditions can be combined with AND and
OR");
            System.out.println("6. Optimistic locking can be implemented using a
version attribute");
            System.out.println(
                "7. Conditional operations consume the same amount of write capacity
whether they succeed or fail");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteItem
  - PutItem
  - UpdateItem

Usar nomes de atributo de expressão

O exemplo de código a seguir mostra como usar nomes de atributo de expressão no DynamoDB.

- Trabalhe com palavras reservadas nas expressões do DynamoDB.
- Use espaços reservados para nomes de atributo de expressão.
- Manipule caracteres especiais em nomes de atributo.

SDK para Java 2.x

Demonstre nomes de atributos de expressão usando AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates an attribute that is a reserved word in DynamoDB.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that are reserved words in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param reservedWordAttribute The reserved word attribute to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
```

```
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", reservedWordAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates an attribute that contains special characters.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that contain special characters.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param specialCharAttribute The attribute with special characters to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateSpecialCharacterAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String specialCharAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", specialCharAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }
```

```
    /**
     * Queries items using an attribute that is a reserved word.
     *
     * <p>This method demonstrates how to use expression attribute names in a query
     * when the attribute is a reserved word.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key
     * @param reservedWordAttribute The reserved word attribute to filter on
     * @param value The value to compare against
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse queryWithReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the query parameters using expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attr", reservedWordAttribute);

        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);
        expressionAttributeValues.put(":value", value);

        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attr = :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }
```

```
    /**
     * Updates a nested attribute with a path that contains reserved words.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * nested attributes where the path contains reserved words.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributePath The path to the nested attribute as an array
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateNestedReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> attributePath,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < attributePath.size(); i++) {
            expressionAttributeNames.put("#attr" + i, attributePath.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < attributePath.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
```

```
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Scans a table with multiple attribute name placeholders.
     *
     * <p>This method demonstrates how to use multiple expression attribute names
     * in a complex filter expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param filters Object mapping attribute names to filter values
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithMultipleAttributeNames(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
filters) {

        // Create expression attribute names and values
        Map<String, String> expressionAttributeNames = new HashMap<>();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        StringBuilder filterExpression = new StringBuilder();

        // Build the filter expression
        int index = 0;
        for (Map.Entry<String, AttributeValue> entry : filters.entrySet()) {
            String attrName = entry.getKey();
            AttributeValue attrValue = entry.getValue();

            String nameKey = "#attr" + index;
            String valueKey = ":val" + index;

            expressionAttributeNames.put(nameKey, attrName);
            expressionAttributeValues.put(valueKey, attrValue);

            // Add AND between conditions (except for the first one)
            if (index > 0) {
                filterExpression.append(" AND ");
```

```
            }

            filterExpression.append(nameKey).append(" = ").append(valueKey);
            index++;
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression(filterExpression.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }
```

Exemplo de uso de nomes de atributos de expressão com AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating expression attribute names in DynamoDB");

        try {
            // Example 1: Update an attribute that is a reserved word
            System.out.println("\nExample 1: Updating an attribute that is a
reserved word");
            UpdateItemResponse response1 = updateReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Size", // "SIZE" is a reserved word in DynamoDB
                AttributeValue.builder().s("Large").build());

            System.out.println("Updated attribute: " + response1.attributes());

            // Example 2: Update an attribute with special characters
```

```
            System.out.println("\nExample 2: Updating an attribute with special
characters");
            UpdateItemResponse response2 = updateSpecialCharacterAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Product-Type", // Contains a hyphen, which is a special character
                AttributeValue.builder().s("Electronics").build());

            System.out.println("Updated attribute: " + response2.attributes());

            // Example 3: Query with a reserved word attribute
            System.out.println("\nExample 3: Querying with a reserved word
attribute");
            QueryResponse response3 = queryWithReservedWordAttribute(
                dynamoDbClient,
                tableName,
                "Category",
                AttributeValue.builder().s("Electronics").build(),
                "Count", // "COUNT" is a reserved word in DynamoDB
                AttributeValue.builder().n("10").build());

            System.out.println("Found " + response3.count() + " items");

            // Example 4: Update a nested attribute with reserved words in the path
            System.out.println("\nExample 4: Updating a nested attribute with
reserved words in the path");
            UpdateItemResponse response4 = updateNestedReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                Arrays.asList("Dimensions", "Size", "Height"), // "SIZE" is a
reserved word
                AttributeValue.builder().n("30").build());

            System.out.println("Updated nested attribute: " +
response4.attributes());

            // Example 5: Scan with multiple attribute name placeholders
            System.out.println("\nExample 5: Scanning with multiple attribute name
placeholders");
            Map<String, AttributeValue> filters = new HashMap<>();
            filters.put("Size", AttributeValue.builder().s("Large").build());
            filters.put("Count", AttributeValue.builder().n("10").build());
```

```
            filters.put(
                "Product-Type", AttributeValue.builder().s("Electronics").build());

            ScanResponse response5 = scanWithMultipleAttributeNames(dynamoDbClient,
tableName, filters);

            System.out.println("Found " + response5.count() + " items");

            // Show some common reserved words
            System.out.println("\nSome common DynamoDB reserved words:");
            List<String> commonReservedWords = getDynamoDBReservedWords();
            System.out.println(String.join(", ", commonReservedWords));

            // Explain expression attribute names
            System.out.println("\nKey points about expression attribute names:");
            System.out.println("1. Use expression attribute names (#name) for
reserved words");
            System.out.println("2. Use expression attribute names for attributes
with special characters");
            System.out.println(
                "3. Special characters include: spaces, hyphens, dots, and other
non-alphanumeric characters");
            System.out.println("4. Expression attribute names are required for
nested attributes with reserved words");
            System.out.println("5. You can use multiple expression attribute names
in a single expression");
            System.out.println("6. Expression attribute names are case-sensitive");
            System.out.println("7. Expression attribute names are only used in
expressions, not in the actual data");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - Consulta
  - UpdateItem

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Java 2.x

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos sem servidor

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do DynamoDB com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
 {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
```

```
}
```

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Java 2.x

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Java.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos do Amazon EC2 usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon EC2.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
```

```
            .build();

        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário em um prompt de comando.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ssm.model.Parameter;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
```

```
 * 6. Gets additional information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI’s
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger = LoggerFactory.getLogger(EC2Scenario.class);
    public static void main(String[] args) throws InterruptedException,
 UnknownHostException {

        logger.info("""
            Usage:
               <keyName> <fileName> <groupName> <groupDesc>

            Where:
               keyName -  A key pair name (for example, TestKeyPair).\s
               fileName -  A file name where the key information is written to.\s
               groupName - The name of the security group.\s
               groupDesc - The description of the security group.\s
            """);

        Scanner scanner = new Scanner(System.in);
        EC2Actions ec2Actions = new EC2Actions();

        String keyName = "TestKeyPair7" ;
        String fileName = "ec2Key.pem";
        String groupName = "TestSecGroup7" ;
        String groupDesc = "Test Group" ;
        String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
        InetAddress localAddress = InetAddress.getLocalHost();
        String myIpAddress = localAddress.getHostAddress();
```

```
        logger.info("""
            Amazon Elastic Compute Cloud (EC2) is a web service that provides
 secure, resizable compute
            capacity in the cloud. It allows developers and organizations to easily
 launch and manage
            virtual server instances, known as EC2 instances, to run their
 applications.

            EC2 provides a wide range of instance types, each with different
 compute, memory,
            and storage capabilities, to meet the diverse needs of various
 workloads. Developers
            can choose the appropriate instance type based on their application's
 requirements,
            such as high-performance computing, memory-intensive tasks, or GPU-
accelerated workloads.

            The `Ec2AsyncClient` interface in the AWS SDK for Java 2.x provides a
 set of methods to
            programmatically interact with the Amazon EC2 service. This allows
 developers to
            automate the provisioning, management, and monitoring of EC2 instances
 as part of their
            application deployment pipelines. With EC2, teams can focus on building
 and deploying
            their applications without having to worry about the underlying
 infrastructure
            required to host and manage physical servers.

            This scenario walks you through how to perform key operations for this
 service.
            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an RSA key pair and save the private key material as
 a .pem file.");
        logger.info("""
            An RSA key pair for Amazon EC2 is a security mechanism used to
 authenticate and secure
```

```
            access to your EC2 instances. It consists of a public key and a private
key,
            which are generated as a pair.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateKeyPairResponse> future =
ec2Actions.createKeyPairAsync(keyName, fileName);
            CreateKeyPairResponse response = future.join();
            logger.info("Key Pair successfully created. Key Fingerprint: " +
response.keyFingerprint());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.getMessage().contains("already exists")) {
                    // Key pair already exists.
                    logger.info("The key pair '" + keyName + "' already exists.
Moving on...");
                } else {
                    logger.info("EC2 error occurred: Error message: {}, Error code
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();
            DescribeKeyPairsResponse keyPairsResponse = future.join();
            keyPairsResponse.keyPairs().forEach(keyPair -> logger.info(
                "Found key pair with name {} and fingerprint {}",
                keyPair.keyName(),
                keyPair.keyFingerprint()));
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Error message: {}, Error code {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a security group.");
        logger.info("""
            An AWS EC2 Security Group is a virtual firewall that controls the
            inbound and outbound traffic to an EC2 instance. It acts as a first
line
            of defense for your EC2 instances, allowing you to specify the rules
that
            govern the network traffic entering and leaving your instances.
           """);
        waitForInputToContinue(scanner);
        String groupId = "";
        try {
            CompletableFuture<String> future =
ec2Actions.createSecurityGroupAsync(groupName, groupDesc, vpcId, myIpAddress);
            future.join();
            logger.info("Created security group") ;

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {
                    logger.info("The Security Group already exists. Moving on...");
                } else {
                    logger.error("An unexpected error occurred: {}",
ec2Ex.awsErrorDetails().errorMessage());
                    return;
                }
```

```
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Display security group information for the new security
group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeSecurityGroupArnByNameAsync(groupName);
            groupId = future.join();
            logger.info("The security group Id is "+groupId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                String errorCode = ec2Ex.awsErrorDetails().errorCode();
                if ("InvalidGroup.NotFound".equals(errorCode)) {
                    logger.info("Security group '{}' does not exist. Error Code:
{}", groupName, errorCode);
                } else {
                    logger.info("EC2 error occurred: Message {}, Error Code: {}",
ec2Ex.getMessage(), errorCode);
                }
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2
in the name.");
        logger.info("""
            An Amazon EC2 AMI (Amazon Machine Image) is a pre-configured virtual
machine image that
```

```
            serves as a template for launching EC2 instances. It contains all the
necessary software and
            configurations required to run an application or operating system on an
EC2 instance.
            """);
        waitForInputToContinue(scanner);
        String instanceAMI="";
        try {
            CompletableFuture<GetParametersByPathResponse> future =
ec2Actions.getParaValuesAsync();
            GetParametersByPathResponse pathResponse = future.join();
            List<Parameter> parameterList = pathResponse.parameters();
            for (Parameter para : parameterList) {
                if (filterName(para.name())) {
                    instanceAMI = para.value();
                    break;
                }
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The AMI value with amzn2 is: {}", instanceAMI);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
        logger.info("""
           An AMI value represents a specific version of a virtual machine (VM) or
server image.
           It uniquely identifies a particular version of an EC2 instance, including
its operating system,
           pre-installed software, and any custom configurations. This allows you to
consistently deploy the same
           VM image across your infrastructure.
```

```
            """);
        waitForInputToContinue(scanner);
        String amiValue;
        try {
            CompletableFuture<String> future =
ec2Actions.describeImageAsync(instanceAMI);
            amiValue = future.join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Retrieves an instance type available in the current AWS
region.");
        waitForInputToContinue(scanner);
        String instanceType;
        try {
            CompletableFuture<String> future = ec2Actions.getInstanceTypesAsync();
            instanceType = future.join();
            if (!instanceType.isEmpty()) {
                logger.info("Found instance type: " + instanceType);
            } else {
                logger.info("Desired instance type not found.");
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
```

```
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an Amazon EC2 instance using the key pair, the
instance type, the security group, and the EC2 AMI value.");
        logger.info("Once the EC2 instance is created, it is placed into a running
state.");
        waitForInputToContinue(scanner);
        String newInstanceId;
        try {
            CompletableFuture<String> future =
ec2Actions.runInstanceAsync(instanceType, keyName, groupName, amiValue);
            newInstanceId = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                switch (ec2Ex.awsErrorDetails().errorCode()) {
                    case "InvalidParameterValue":
                        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InsufficientInstanceCapacity":
                        // Handle insufficient instance capacity.
                        logger.info("Insufficient instance capacity: {}, {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InvalidGroup.NotFound":
                        // Handle security group not found.
                        logger.info("Security group not found: {},{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    default:
                        logger.info("EC2 error occurred: {} (Code: {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                }
                return;
            } else {
```

```
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        logger.info("The instance Id is " + newInstanceId);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Display information about the running instance. ");

        waitForInputToContinue(scanner);
        String publicIp;
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP {}", publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }

        }
        logger.info("You can SSH to the instance using this command:");
        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Stop the instance using a waiter (this may take a few
mins).");
        // Remove the 2nd one
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.stopInstanceAsync(newInstanceId);
```

```
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Start the instance using a waiter (this may take a few
mins).");
        try {
            CompletableFuture<Void> future =
ec2Actions.startInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Allocate an Elastic IP address and associate it with the
instance.");
        logger.info("""
```

```
            An Elastic IP address is a static public IP address that you can
associate with your EC2 instance.
            This allows you to have a fixed, predictable IP address that remains the
same even if your instance
            is stopped, terminated, or replaced.
            This is particularly useful for applications or services that need to be
accessed consistently from a
            known IP address.

            An EC2 Allocation ID (also known as a Reserved Instance Allocation
ID) is a unique identifier associated with a Reserved Instance (RI) that you have
purchased in AWS.

            When you purchase a Reserved Instance, AWS assigns a unique Allocation
ID to it.
            This Allocation ID is used to track and identify the specific RI you
have purchased,
            and it is important for managing and monitoring your Reserved Instances.

            """);

        waitForInputToContinue(scanner);
        String allocationId;
        try {
            CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
            allocationId = future.join();
            logger.info("Successfully allocated address with ID: " +allocationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The allocation Id value is " + allocationId);
        waitForInputToContinue(scanner);
        String associationId;
        try {
            CompletableFuture<String> future =
ec2Actions.associateAddressAsync(newInstanceId, allocationId);
```

```
            associationId = future.join();
            logger.info("Successfully associated address with ID: " +associationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Describe the instance again. Note that the public IP
address has changed");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP: " + publicIp);
            logger.info("You can SSH to the instance using this command:");
            logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Disassociate and release the Elastic IP address.");
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
            future.join();
            logger.info("Address successfully disassociated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
            future.join(); // Wait for the operation to complete
            logger.info("Elastic IP address successfully released.");
        } catch (RuntimeException rte) {
            logger.info("An unexpected error occurred: {}", rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Terminate the instance and use a waiter (this may take a
few mins).");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
ec2Actions.terminateEC2Async(newInstanceId);
            future.join();
            logger.info("EC2 instance successfully terminated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
```

```
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Delete the security group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
            future.join();
            logger.info("Security group successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Delete the key.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DeleteKeyPairResponse> future =
ec2Actions.deleteKeysAsync(keyName);
            future.join();
            logger.info("Successfully deleted key pair named " + keyName);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
```

```
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
 ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon EC2 scenario.");
        logger.info(DASHES);
    }
    public static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Defina uma classe que envolva as ações do EC2.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
import
 software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.InstanceTypeInfo;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.IpRange;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Vpc;
import software.amazon.awssdk.services.ec2.paginators.DescribeImagesPublisher;
```

```
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesPublisher;
import
 software.amazon.awssdk.services.ec2.paginators.DescribeSecurityGroupsPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeVpcsPublisher;
import software.amazon.awssdk.services.ec2.waiters.Ec2AsyncWaiter;
import software.amazon.awssdk.services.ssm.SsmAsyncClient;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathRequest;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesResponse;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.concurrent.atomic.AtomicReference;

public class EC2Actions {
    private static final Logger logger = LoggerFactory.getLogger(EC2Actions.class);
    private static Ec2AsyncClient ec2AsyncClient;

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static Ec2AsyncClient getAsyncClient() {
        if (ec2AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
```

```
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
            } else if (resp == null) {
```

```
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }

    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }

    /**
     * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
     *
     * @param instanceId the ID of the EC2 instance to terminate
     * @return a {@link CompletableFuture} that completes when the instance has been
terminated
     * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
     */
    public CompletableFuture<Object> terminateEC2Async(String instanceId) {
        TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
```

```
            .build();

        CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
        return responseFuture.thenCompose(terminateResponse -> {
            if (terminateResponse == null) {
                throw new RuntimeException("No response received for terminating
instance " + instanceId);
            }
            System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
            return getAsyncClient().waiter()
                .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
                .thenApply(waiterResponse -> null);
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Releases an Elastic IP address asynchronously.
     *
     * @param allocId the allocation ID of the Elastic IP address to be released
     * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
     */
    public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP address",
ex);
            }
        });

        return response;
```

```
    }

    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
     *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
     *         finished.
     * @throws RuntimeException if the disassociation of the address fails.
     */
    public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }

    /**
     * Associates an Elastic IP address with an EC2 instance asynchronously.
     *
     * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
     * @param allocationId  the allocation ID of the Elastic IP address to associate
     * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
     *         or throws a {@link RuntimeException} if the operation fails
     */
```

```
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
       return responseFuture.thenApply(response -> {
           if (response.associationId() != null) {
               return response.associationId();
           } else {
               throw new RuntimeException("Association ID is null after associating
address.");
           }
       }).whenComplete((result, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to associate address", ex);
           }
       });
   }

   /**
    * Allocates an Elastic IP address asynchronously in the VPC domain.
    *
    * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
    */
   public CompletableFuture<String> allocateAddressAsync() {
       AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
           .domain(DomainType.VPC)
           .build();

       CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
       return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to allocate address", ex);
           }
       });
   }
```

```
    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
    public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
        AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.reservations().stream()
                .flatMap(reservation -> reservation.instances().stream())
                .filter(instance -> instance.instanceId().equals(newInstanceId))
                .findFirst()
                .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
        }).thenApply(v -> {
            String publicIpAddress = publicIpAddressRef.get();
            if (publicIpAddress == null) {
                throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
            }
            return publicIpAddress;
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
        });
    }

    /**
     * Runs an EC2 instance asynchronously.
     *
     * @param instanceType The instance type to use for the EC2 instance.
     * @param keyName The name of the key pair to associate with the EC2 instance.
```

```
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
       RunInstancesRequest runRequest = RunInstancesRequest.builder()
           .instanceType(instanceType)
           .keyName(keyName)
           .securityGroups(groupName)
           .maxCount(1)
           .minCount(1)
           .imageId(amiId)
           .build();

       CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
       return responseFuture.thenCompose(response -> {
           String instanceIdVal = response.instances().get(0).instanceId();
           System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
           return getAsyncClient().waiter()
               .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
               .thenCompose(waitResponse -> getAsyncClient().waiter()
                   .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                   .thenApply(runningResponse -> instanceIdVal));
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
       });
   }

   /**
    * Asynchronously retrieves the instance types available in the current AWS
region.
    * <p>
    * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
```

```
    * and then processes the response. It logs the memory information, network
information,
    * and instance type for each instance type returned. Additionally, it returns a
    * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
    * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
    * found, an empty string is returned.
    * </p>
    *
    * @return a {@link CompletableFuture} that resolves to the instance type string
for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
   public CompletableFuture<String> getInstanceTypesAsync() {
       DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
           .maxResults(10)
           .build();

       CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
               for (InstanceTypeInfo type : instanceTypes) {
                   logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                   logger.info("Network information is " +
type.networkInfo().toString());
                   logger.info("Instance type is " +
type.instanceType().toString());
               }
           } else {
               throw (RuntimeException) ex;
           }
       });

       return response.thenApply(resp -> {
           for (InstanceTypeInfo type : resp.instanceTypes()) {
               String instanceType = type.instanceType().toString();
               if (instanceType.equals("t2.2xlarge")) {
                   return instanceType;
```

```
                }
            }
            return "";
        });
    }

    /**
     * Asynchronously describes an AWS EC2 image with the specified image ID.
     *
     * @param imageId the ID of the image to be described
     * @return a {@link CompletableFuture} that, when completed, contains the ID of
the described image
     * @throws RuntimeException if no images are found with the provided image ID,
or if an error occurs during the AWS API call
     */
    public CompletableFuture<String> describeImageAsync(String imageId) {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(imageId)
            .build();

        AtomicReference<String> imageIdRef = new AtomicReference<>();
        DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
        return paginator.subscribe(response -> {
            response.images().stream()
                .filter(image -> image.imageId().equals(imageId))
                .findFirst()
                .ifPresent(image -> {
                    logger.info("The description of the image is " +
image.description());
                    logger.info("The name of the image is " + image.name());
                    imageIdRef.set(image.imageId());
                });
        }).thenApply(v -> {
            String id = imageIdRef.get();
            if (id == null) {
                throw new RuntimeException("No images found with the provided image
ID.");
            }
            return id;
        }).exceptionally(ex -> {
            logger.info("Failed to describe image: " + ex.getMessage());
            throw new RuntimeException("Failed to describe image", ex);
        });
```

```
    }

    /**
     * Retrieves the parameter values asynchronously using the AWS Systems Manager
(SSM) API.
     *
     * @return a {@link CompletableFuture} that holds the response from the SSM API
call to get parameters by path
     */
    public CompletableFuture<GetParametersByPathResponse> getParaValuesAsync() {
        SsmAsyncClient ssmClient = SsmAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        // Create a CompletableFuture to hold the final result.
        CompletableFuture<GetParametersByPathResponse> responseFuture = new
CompletableFuture<>();
        ssmClient.getParametersByPath(parameterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    responseFuture.completeExceptionally(new
RuntimeException("Failed to get parameters by path", exception));
                } else {
                    responseFuture.complete(response);
                }
            });

        return responseFuture;
    }


    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
```

```
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupNames(groupName)
            .build();

        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }

    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
```

```
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
                   .build();

               IpPermission ipPerm = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(80)
                   .fromPort(80)
                   .ipRanges(ipRange)
                   .build();

               IpPermission ipPerm2 = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(22)
                   .fromPort(22)
                   .ipRanges(ipRange)
                   .build();

               AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                   .groupName(groupName)
                   .ipPermissions(ipPerm, ipPerm2)
                   .build();

               return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                   .thenApply(authResponse -> groupId);
           })
```

```
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Creates a new key pair asynchronously.
     *
     * @param keyName the name of the key pair to create
     * @param fileName the name of the file to write the key material to
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of creating the key pair and writing the key material to a file
     */
```

```
   public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
       CreateKeyPairRequest request = CreateKeyPairRequest.builder()
           .keyName(keyName)
           .build();

       CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
       responseFuture.whenComplete((response, exception) -> {
           if (response != null) {
               try {
                   BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                   writer.write(response.keyMaterial());
                   writer.close();
               } catch (IOException e) {
                   throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
               }
           } else {
               throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
           }
       });

       return responseFuture;
   }

   /**
    * Describes the first default VPC asynchronously and using a paginator.
    *
    * @return a {@link CompletableFuture} that, when completed, contains the first
default VPC found.\
    */
   public CompletableFuture<Vpc> describeFirstEC2VpcAsync() {
       Filter myFilter = Filter.builder()
           .name("is-default")
           .values("true")
           .build();

       DescribeVpcsRequest request = DescribeVpcsRequest.builder()
           .filters(myFilter)
           .build();
```

```
        DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
        AtomicReference<Vpc> vpcRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.vpcs().stream()
                .findFirst()
                .ifPresent(vpcRef::set);
        }).thenApply(v -> {
            Vpc vpc = vpcRef.get();
            if (vpc == null) {
                throw new RuntimeException("Default VPC not found");
            }
            return vpc;
        }).exceptionally(ex -> {
            logger.info("Failed to describe VPCs: " + ex.getMessage());
            throw new RuntimeException("Failed to describe VPCs", ex);
        });
    }

    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
```

```
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }

    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();
```

```
        logger.info("Starting instance " + instanceId + " and waiting for it to
 run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
 start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress

- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Allocates an Elastic IP address asynchronously in the VPC domain.
    *
    * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
    */
   public CompletableFuture<String> allocateAddressAsync() {
       AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
           .domain(DomainType.VPC)
           .build();

       CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
       return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to allocate address", ex);
```

```
            }
        });
    }
```

- Para obter detalhes da API, consulte AllocateAddress em Referência AWS SDK for Java 2.x da API.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
    * @param allocationId  the allocation ID of the Elastic IP address to associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
       return responseFuture.thenApply(response -> {
           if (response.associationId() != null) {
```

```
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after associating
address.");
            }
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }
```

- Para obter detalhes da API, consulte AssociateAddress em Referência AWS SDK for Java 2.x da API.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
```

```
    * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
                   .build();

               IpPermission ipPerm = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(80)
                   .fromPort(80)
                   .ipRanges(ipRange)
                   .build();

               IpPermission ipPerm2 = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(22)
                   .fromPort(22)
                   .ipRanges(ipRange)
                   .build();

               AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                   .groupName(groupName)
                   .ipPermissions(ipPerm, ipPerm2)
                   .build();

               return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                   .thenApply(authResponse -> groupId);
           })
           .whenComplete((result, exception) -> {
               if (exception != null) {
```

```
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngress em Referência AWS SDK for Java 2.x da API.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new key pair asynchronously.
     *
     * @param keyName the name of the key pair to create
     * @param fileName the name of the file to write the key material to
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of creating the key pair and writing the key material to a file
     */
    public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();
```

```
        CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
        responseFuture.whenComplete((response, exception) -> {
            if (response != null) {
                try {
                    BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                    writer.write(response.keyMaterial());
                    writer.close();
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- Para obter detalhes da API, consulte CreateKeyPair em Referência AWS SDK for Java 2.x da API.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security group
    * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
                   .build();

               IpPermission ipPerm = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(80)
                   .fromPort(80)
                   .ipRanges(ipRange)
                   .build();

               IpPermission ipPerm2 = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(22)
                   .fromPort(22)
                   .ipRanges(ipRange)
                   .build();
```

```
                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateSecurityGroup em Referência AWS SDK for Java 2.x da API.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a key pair asynchronously.
```

```
    *
    * @param keyPair the name of the key pair to delete
    * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
    *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
    *         that provides the result of the key pair deletion operation.
    */
   public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
       DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
           .keyName(keyPair)
           .build();

       // Initiate the asynchronous request to delete the key pair.
       CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
           } else if (resp == null) {
               throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
           }
       });
   }
```

- Para obter detalhes da API, consulte DeleteKeyPair em Referência AWS SDK for Java 2.x da API.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }
```

- Para obter detalhes da API, consulte DeleteSecurityGroup em Referência AWS SDK for Java 2.x da API.

`DescribeInstanceTypes`

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously retrieves the instance types available in the current AWS
region.
    * <p>
    * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
    * and then processes the response. It logs the memory information, network
information,
    * and instance type for each instance type returned. Additionally, it returns a
    * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
    * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
    * found, an empty string is returned.
    * </p>
    *
    * @return a {@link CompletableFuture} that resolves to the instance type string
for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
   public CompletableFuture<String> getInstanceTypesAsync() {
       DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
           .maxResults(10)
           .build();

       CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
               for (InstanceTypeInfo type : instanceTypes) {
```

```
                    logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                    logger.info("Network information is " +
type.networkInfo().toString());
                    logger.info("Instance type is " +
type.instanceType().toString());
                }
            } else {
                throw (RuntimeException) ex;
            }
        });

        return response.thenApply(resp -> {
            for (InstanceTypeInfo type : resp.instanceTypes()) {
                String instanceType = type.instanceType().toString();
                if (instanceType.equals("t2.2xlarge")) {
                    return instanceType;
                }
            }
            return "";
        });
    }
```

- Para obter detalhes da API, consulte DescribeInstanceTypes em Referência AWS SDK for Java 2.x da API.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously describes the state of an EC2 instance.
```

```
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
    public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
        AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.reservations().stream()
                .flatMap(reservation -> reservation.instances().stream())
                .filter(instance -> instance.instanceId().equals(newInstanceId))
                .findFirst()
                .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
        }).thenApply(v -> {
            String publicIpAddress = publicIpAddressRef.get();
            if (publicIpAddress == null) {
                throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
            }
            return publicIpAddress;
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
        });
    }
```

- Para obter detalhes da API, consulte DescribeInstances em Referência AWS SDK for Java 2.x da API.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the key pairs associated with the current AWS
account.
    *
    * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
    * information about the key pairs.
    */
   public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
       CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
       responseFuture.whenComplete((response, exception) -> {
           if (exception != null) {
             throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
           }
       });

       return responseFuture;
   }
```

- Para obter detalhes da API, consulte DescribeKeyPairs em Referência AWS SDK for Java 2.x da API.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
               .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
               .findFirst()
               .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
       }).thenApply(v -> {
           String groupId = groupIdRef.get();
           if (groupId == null) {
               throw new RuntimeException("No security group found with the name: "
+ groupName);
           }
```

```
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK for Java 2.x da API.

`DisassociateAddress`

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Disassociates an Elastic IP address from an instance asynchronously.
    *
    * @param associationId The ID of the association you want to disassociate.
    * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
    *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
    *         finished.
    * @throws RuntimeException if the disassociation of the address fails.
    */
   public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
       Ec2AsyncClient ec2 = getAsyncClient();
       DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
           .associationId(associationId)
           .build();
```

```
        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
 ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }
```

- Para obter detalhes da API, consulte DisassociateAddress em Referência AWS SDK for Java 2.x da API.

`GetPasswordData`

O código de exemplo a seguir mostra como usar `GetPasswordData`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class GetPasswordData {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <instanceId>

                Where:
                   instanceId - An instance id value that you can obtain from the
 AWS Management Console.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String instanceId = args[0];
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
 instanceId);
            future.join();
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Fetches the password data for the specified EC2 instance asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the request
     * @param instanceId instanceId the ID of the EC2 instance for which you want to
 fetch the password data
     * @return a {@link CompletableFuture} that completes when the password data has
 been fetched
     * @throws RuntimeException if there was a failure in fetching the password data
     */
```

```
    public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
 ec2AsyncClient, String instanceId) {
        GetPasswordDataRequest getPasswordDataRequest =
 GetPasswordDataRequest.builder()
            .instanceId(instanceId)
            .build();


        CompletableFuture<GetPasswordDataResponse> response =
 ec2AsyncClient.getPasswordData(getPasswordDataRequest);
        response.whenComplete((getPasswordDataResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get password data for
 instance: " + instanceId, ex);
            } else if (getPasswordDataResponse == null ||
 getPasswordDataResponse.passwordData().isEmpty()) {
                throw new RuntimeException("No password data found for instance: " +
 instanceId);
            } else {
                String encryptedPasswordData =
 getPasswordDataResponse.passwordData();
                System.out.println("Encrypted Password Data: " +
 encryptedPasswordData);
            }
        });

        return response.thenApply(resp -> null);
    }
}
```

- Para obter detalhes da API, consulte GetPasswordData em Referência AWS SDK for Java 2.x da API.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Releases an Elastic IP address asynchronously.
    *
    * @param allocId the allocation ID of the Elastic IP address to be released
    * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
    */
   public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
       ReleaseAddressRequest request = ReleaseAddressRequest.builder()
           .allocationId(allocId)
           .build();

       CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to release Elastic IP address",
ex);
           }
       });

       return response;
   }
```

- Para obter detalhes da API, consulte ReleaseAddress em Referência AWS SDK for Java 2.x da API.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2 instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
       RunInstancesRequest runRequest = RunInstancesRequest.builder()
           .instanceType(instanceType)
           .keyName(keyName)
           .securityGroups(groupName)
           .maxCount(1)
           .minCount(1)
           .imageId(amiId)
           .build();

       CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
       return responseFuture.thenCompose(response -> {
           String instanceIdVal = response.instances().get(0).instanceId();
           System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
           return getAsyncClient().waiter()
               .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
               .thenCompose(waitResponse -> getAsyncClient().waiter()
                   .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
```

```
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }
```

- Para obter detalhes da API, consulte RunInstances em Referência AWS SDK for Java 2.x da API.

`StartInstances`

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();
```

```
        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
```

- Para obter detalhes da API, consulte StartInstances em Referência AWS SDK for Java 2.x da API.

`StopInstances`

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
    *
    * @param instanceId the ID of the EC2 instance to stop
    * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
    */
   public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
       StopInstancesRequest stopRequest = StopInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
           .client(getAsyncClient())
           .build();

       CompletableFuture<Void> resultFuture = new CompletableFuture<>();
       logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
       getAsyncClient().stopInstances(stopRequest)
           .thenCompose(response -> {
               if (response.stoppingInstances().isEmpty()) {
                   return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
               }
               return ec2Waiter.waitUntilInstanceStopped(describeRequest);
           })
           .thenAccept(waiterResponse -> {
               logger.info("Successfully stopped instance " + instanceId);
               resultFuture.complete(null);
           })
           .exceptionally(throwable -> {
               logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
               resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
               return null;
           });
```

```
        return resultFuture;
    }
```

- Para obter detalhes da API, consulte StopInstances em Referência AWS SDK for Java 2.x da API.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
     *
     * @param instanceId the ID of the EC2 instance to terminate
     * @return a {@link CompletableFuture} that completes when the instance has been
terminated
     * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
     */
    public CompletableFuture<Object> terminateEC2Async(String instanceId) {
        TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
        return responseFuture.thenCompose(terminateResponse -> {
            if (terminateResponse == null) {
                throw new RuntimeException("No response received for terminating
instance " + instanceId);
```

```
            }
            System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
            return getAsyncClient().waiter()
                .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
                .thenApply(waiterResponse -> null);
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }
```

- Para obter detalhes da API, consulte TerminateInstances em Referência AWS SDK for Java 2.x da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
   public static void main(String[] args) throws IOException, InterruptedException
{
       Scanner in = new Scanner(System.in);
       Database database = new Database();
       AutoScaler autoScaler = new AutoScaler();
       LoadBalancer loadBalancer = new LoadBalancer();

       System.out.println(DASHES);
       System.out.println("Welcome to the demonstration of How to Build and Manage
a Resilient Service!");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("A - SETUP THE RESOURCES");
       System.out.println("Press Enter when you're ready to start deploying
resources.");
       in.nextLine();
       deploy(loadBalancer);
       System.out.println(DASHES);
       System.out.println(DASHES);
       System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
       System.out.println("Press Enter when you're ready.");
       in.nextLine();
       demo(loadBalancer);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("C - DELETE THE RESOURCES");
       System.out.println("""
               This concludes the demo of how to build and manage a resilient
service.
               To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
               that were created for this demo.
               """);

       System.out.println("\n Do you want to delete the resources (y/n)? ");
       String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

       if (userInput.equals("y")) {
           // Delete resources here
           deleteResources(loadBalancer, autoScaler, database);
           System.out.println("Resources deleted.");
       } else {
```

```
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
```

```
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
```

```
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
```

```
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
```

```
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.
```

```
                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       System.out.println(
               """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns a
static response.
               The service still reports as healthy because health checks are still
shallow.
               """);
       demoChoices(loadBalancer);

       System.out.println("Let's reinstate the recommendation service.");
       paramHelper.put(paramHelper.tableName, paramHelper.dyntable);
```

```
        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
```

```
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
```

```
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();
```

```
                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                        target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
```

```
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }
```

```
    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }
```

```
   /**
    * Replaces the profile associated with a running instance. After the profile is
    * replaced, the instance is rebooted to ensure that it uses the new profile.
    * When
    * the instance is ready, Systems Manager is used to restart the Python web
    * server.
    */
   public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
           throws InterruptedException {
       // Create an IAM instance profile specification.
       software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
               .builder()
               .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                             // name.
               .build();

       // Replace the IAM instance profile association for the EC2 instance.
       ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
               .builder()
               .iamInstanceProfile(iamInstanceProfile)
               .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
               .build();

       try {
           getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
           // Handle the response as needed.
       } catch (Ec2Exception e) {
           // Handle exceptions, log, or report the error.
           System.err.println("Error: " + e.getMessage());
       }
       System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
               newInstanceProfileName);
       TimeUnit.SECONDS.sleep(15);
       boolean instReady = false;
       int tries = 0;

       // Reboot after 60 seconds
```

```
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
```

```
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
```

```
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }
```

```
    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
```

```
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
```

```
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
```

```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
```

```
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
```

```
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
 DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
```

```
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
" + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
```

```
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
```

```
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
```

```
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
```

```
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
```

```
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
```

```
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
```

```
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
```

```
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
{
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();
```

```
            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
```

```
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Exemplos do Amazon ECR usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon ECR.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Amazon ECR

O exemplo de código a seguir mostra como começar a usar o Amazon ECR.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
```

```
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();

        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Consulte detalhes da API em listImages na Referência de APIs do AWS SDK for Java 2.x .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um repositório do Amazon ECR.
- Defina políticas de repositório.
- Recupere URIs de repositórios.
- Obtenha tokens de autorização do Amazon ECR.
- Defina políticas de ciclo de vida para repositórios do Amazon ECR.
- Envie por push uma imagem do Docker para um repositório do Amazon ECR.
- Verifique a existência de uma imagem em um repositório do Amazon ECR.
- Liste os repositórios do Amazon ECR da conta e verifique os detalhes sobre eles.
- Exclua repositórios do Amazon ECR.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon ECR.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 *
 * This Java code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text. Without
 a local image,
 * this Java program will not successfully run. For more information including how
 to create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = """
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
 container registry
```

```
            service provided by AWS. It allows developers and organizations to
securely
            store, manage, and deploy Docker container images.
            ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
            from building and testing to production deployment.\s

            The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
            programmatically interact with the Amazon ECR service. This allows
developers to
            automate the storage, retrieval, and management of container images as
part of their application
            deployment pipelines. With ECR, teams can focus on building and
deploying their
            applications without having to worry about the underlying
infrastructure required to
            host and manage a container registry.

           This scenario walks you through how to perform key operations for this
service.
           Let's get started...

           You have two choices:
           1 - Run the entire program.
           2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
           this program that did not complete).
           """);

       while (true) {
           String input = scanner.nextLine();
           if (input.trim().equalsIgnoreCase("1")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else if (input.trim().equalsIgnoreCase("2")) {
               String repoName = "echo-text";
               ecrActions.deleteECRRepository(repoName);
               return;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
```

```
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("""
           1. Create an ECR repository.

           The first task is to ensure we have a local Docker image named echo-
text.
           If this image exists, then an Amazon ECR repository is created.

           An ECR repository is a private Docker container repository provided
           by Amazon Web Services (AWS). It is a managed service that makes it easy
           to store, manage, and deploy Docker container images.\s
           """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
 " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
```

```
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function is
crucial for maintaining
        the security and integrity of your container images. The repository policy
allows you to
        define specific rules and restrictions for accessing and managing the images
stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
```

```
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
        valid authorization token, which is required to authenticate your requests
to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the authorization
token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.
```

```
        The URI  of an Amazon ECR repository is important. When you want to deploy a
container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime to
pull the
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images, ensuring
that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
```

```
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by making a
call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message indicating
that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker image
to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
```

```
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the image
 in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
 CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository and
 have Amazon ECR delete all of its contents
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("An error occurred while deleting the Docker
 image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
```

```
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper para os métodos do SDK do Amazon ECR.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
```

```
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
 string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<CreateRepositoryResponse> response =
 getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
 successfully.");
```

```
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }

    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
```

```
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " + repoName +
" repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }



    private static DockerClient getDockerClient() {
        String osName = System.getProperty("os.name");
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
            DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build();
        }
        return dockerClient;
    }

    /**
```

```
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static EcrAsyncClient getAsyncClient() {

        /*
         The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
         and it is designed to provide a high-performance, asynchronous HTTP client
for interacting with AWS services.
         It uses the Netty framework to handle the underlying network communication
and the Java NIO API to
         provide a non-blocking, event-driven approach to HTTP requests and
responses.
         */
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(50)  // Adjust as needed.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
```

```
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
 StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
 getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
 successfully.");
```

```
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                } else {
```

```
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
     * @throws CompletionException if the asynchronous operation completes
 exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
 getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
 asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
```

```
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
     * If the operation is successful, the method prints the token to the console.
     * If an exception occurs, the method handles the exception and prints the error
message.
     *
     * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
     * @throws RuntimeException if there is an unexpected error during the
operation.
     */
    public void getAuthToken() {
        CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
                AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
```

```
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }

    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     * @throws EcrException if an AWS error occurs while getting the repository
policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
```

```
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }

    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does not
exist.
     * @throws EcrException                      if there is an unexpected error
setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
        /*
          This example policy document grants the specified AWS principal the
permission to perform the
          `ecr:BatchGetImage` action. This policy is designed to allow the specified
principal
          to retrieve Docker images from the ECR repository.
         */
        String policyDocumentTemplate = """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "%s"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }
             """;

        String policyDocument = String.format(policyDocumentTemplate, iamRole);
```

```
        SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .policyText(policyDocument)
            .build();

        CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy set successfully.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof RepositoryPolicyNotFoundException) {
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);
```

```
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
```

```
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Ações

`CreateRepository`

O código de exemplo a seguir mostra como usar `CreateRepository`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
    * @throws IllegalArgumentException     If repository name is invalid.
    * @throws RuntimeException             if an error occurs while creating the
repository.
    */
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
```

```
            } else {
                throw new RuntimeException("Unexpected response type");
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateRepository em Referência AWS SDK for Java 2.x da API.

`DeleteRepository`

O código de exemplo a seguir mostra como usar `DeleteRepository`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes an ECR (Elastic Container Registry) repository.
    *
    * @param repoName the name of the repository to delete.
    * @throws IllegalArgumentException if the repository name is null or empty.
    * @throws EcrException if there is an error deleting the repository.
    * @throws RuntimeException if an unexpected error occurs during the deletion
process.
    */
   public void deleteECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
           .force(true)
           .repositoryName(repoName)
           .build();

       CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
       response.whenComplete((deleteRepositoryResponse, ex) -> {
           if (deleteRepositoryResponse != null) {
               System.out.println("You have successfully deleted the " + repoName +
" repository");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
               }
           }
       });

       // Wait for the CompletableFuture to complete
       response.join();
   }
```

- Para obter detalhes da API, consulte DeleteRepository em Referência AWS SDK for Java 2.x da API.

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
```

```
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Para obter detalhes da API, consulte DescribeImages em Referência AWS SDK for Java 2.x da API.

`DescribeRepositories`

O código de exemplo a seguir mostra como usar `DescribeRepositories`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
```

```
    * @throws EcrException        if there is an error retrieving the repository
information.
    * @throws CompletionException if the asynchronous operation completes
exceptionally.
    */
   public void getRepositoryURI(String repoName) {
       DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
           .repositoryNames(repoName)
           .build();

       CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
       response.whenComplete((describeRepositoriesResponse, ex) -> {
           if (ex != null) {
               Throwable cause = ex.getCause();
               if (cause instanceof InterruptedException) {
                   Thread.currentThread().interrupt();
                   String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
           } else {
               if (describeRepositoriesResponse != null) {
                   if (!describeRepositoriesResponse.repositories().isEmpty()) {
                       String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                       System.out.println("Repository URI found: " +
repositoryUri);
                   } else {
                       System.out.println("No repositories found for the given
name.");
                   }
               } else {
                   System.err.println("No response received from
describeRepositories.");
               }
           }
       });
       response.join();
```

```
    }
```

- Para obter detalhes da API, consulte DescribeRepositories em Referência AWS SDK for Java 2.x da API.

`GetAuthorizationToken`

O código de exemplo a seguir mostra como usar `GetAuthorizationToken`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the console.
    * If an exception occurs, the method handles the exception and prints the error
message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
```

```
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }
```

- Para obter detalhes da API, consulte GetAuthorizationToken em Referência AWS SDK for Java 2.x da API.

`GetRepositoryPolicy`

O código de exemplo a seguir mostra como usar `GetRepositoryPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     * @throws EcrException if an AWS error occurs while getting the repository
policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }
```

- Para obter detalhes da API, consulte GetRepositoryPolicy em Referência AWS SDK for Java 2.x da API.

`PushImageCmd`

O código de exemplo a seguir mostra como usar `PushImageCmd`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName  the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   public void pushDockerImage(String repoName, String imageName) {
       System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
       CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
           .thenApply(response -> {
               String token =
response.authorizationData().get(0).authorizationToken();
               String decodedToken = new String(Base64.getDecoder().decode(token));
               String password = decodedToken.substring(4);

               DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
               Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
               assert repoData != null;
               String registryURL = repoData.repositoryUri().split("/")[0];

               AuthConfig authConfig = new AuthConfig()
                   .withUsername("AWS")
                   .withPassword(password)
                   .withRegistryAddress(registryURL);
               return authConfig;
           })
           .thenCompose(authConfig -> {
```

```
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- Para obter detalhes da API, consulte PushImageCmd em Referência AWS SDK for Java 2.x da API.

`SetRepositoryPolicy`

O código de exemplo a seguir mostra como usar `SetRepositoryPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Sets the repository policy for the specified ECR repository.
     *
```

```
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does not
exist.
    * @throws EcrException                     if there is an unexpected error
setting the repository policy.
    */
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the specified
principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
            {
             "Version":"2012-10-17",
             "Statement" : [ {
               "Sid" : "new statement",
               "Effect" : "Allow",
               "Principal" : {
                 "AWS" : "%s"
               },
               "Action" : "ecr:BatchGetImage"
             } ]
           }
            """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
```

```
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }
```

- Para obter detalhes da API, consulte SetRepositoryPolicy em Referência AWS SDK for Java 2.x da API.

`StartLifecyclePolicyPreview`

O código de exemplo a seguir mostra como usar `StartLifecyclePolicyPreview`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
```

```
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Para obter detalhes da API, consulte StartLifecyclePolicyPreview em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon ECS usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon ECS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <clusterName>\s

                Where:
                   clusterName - The name of the ECS cluster to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        String clusterArn = createGivenCluster(ecsClient, clusterName);
        System.out.println("The cluster ARN is " + clusterArn);
        ecsClient.close();
    }

    public static String createGivenCluster(EcsClient ecsClient, String clusterName)
 {
        try {
            ExecuteCommandConfiguration commandConfiguration =
 ExecuteCommandConfiguration.builder()
                    .logging(ExecuteCommandLogging.DEFAULT)
                    .build();

            ClusterConfiguration clusterConfiguration =
 ClusterConfiguration.builder()
                    .executeCommandConfiguration(commandConfiguration)
                    .build();

            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
                    .clusterName(clusterName)
                    .configuration(clusterConfiguration)
                    .build();
```

```
            CreateClusterResponse response =
 ecsClient.createCluster(clusterRequest);
            return response.cluster().clusterArn();

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateCluster em Referência AWS SDK for Java 2.x da API.

`CreateService`

O código de exemplo a seguir mostra como usar `CreateService`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateService {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                  <clusterName> <serviceName> <securityGroups>
 <subnets> <taskDefinition>

                                Where:
                                  clusterName - The name of the ECS cluster.
                                  serviceName - The name of the ECS service to
 create.
                                  securityGroups - The name of the security group.
                                  subnets - The name of the subnet.
                                  taskDefinition - The name of the task definition.
                                """;

                if (args.length != 5) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String clusterName = args[0];
                String serviceName = args[1];
                String securityGroups = args[2];
                String subnets = args[3];
                String taskDefinition = args[4];
                Region region = Region.US_EAST_1;
                EcsClient ecsClient = EcsClient.builder()
                                .region(region)
                                .build();

                String serviceArn = createNewService(ecsClient, clusterName,
 serviceName, securityGroups, subnets,
                                taskDefinition);
                System.out.println("The ARN of the service is " + serviceArn);
                ecsClient.close();
        }
```

```
        public static String createNewService(EcsClient ecsClient,
                        String clusterName,
                        String serviceName,
                        String securityGroups,
                        String subnets,
                        String taskDefinition) {

                try {
                        AwsVpcConfiguration vpcConfiguration =
 AwsVpcConfiguration.builder()
                                        .securityGroups(securityGroups)
                                        .subnets(subnets)
                                        .build();

                        NetworkConfiguration configuration =
 NetworkConfiguration.builder()
                                        .awsvpcConfiguration(vpcConfiguration)
                                        .build();

                        CreateServiceRequest serviceRequest =
 CreateServiceRequest.builder()
                                        .cluster(clusterName)
                                        .networkConfiguration(configuration)
                                        .desiredCount(1)
                                        .launchType(LaunchType.FARGATE)
                                        .serviceName(serviceName)
                                        .taskDefinition(taskDefinition)
                                        .build();

                        CreateServiceResponse response =
 ecsClient.createService(serviceRequest);
                        return response.service().serviceArn();

                } catch (EcsException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
                return "";
        }
}
```

- Para obter detalhes da API, consulte CreateService em Referência AWS SDK for Java 2.x da API.

`DeleteService`

O código de exemplo a seguir mostra como usar `DeleteService`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterName> <serviceArn>\s

                Where:
                  clusterName - The name of the ECS cluster.
                  serviceArn - The ARN of the ECS service.
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        deleteSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void deleteSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .build();

            ecsClient.deleteService(serviceRequest);
            System.out.println("The Service was successfully deleted");

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteService em Referência AWS SDK for Java 2.x da API.

`DescribeClusters`

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> \s

                Where:
                  clusterArn - The ARN of the ECS cluster to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
```

```
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
            DescribeClustersRequest clustersRequest =
 DescribeClustersRequest.builder()
                    .clusters(clusterArn)
                    .build();

            DescribeClustersResponse response =
 ecsClient.describeClusters(clustersRequest);
            List<Cluster> clusters = response.clusters();
            for (Cluster cluster : clusters) {
                System.out.println("The cluster name is " + cluster.clusterName());
            }

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeClusters em Referência AWS SDK for Java 2.x da API.

`DescribeTasks`

O código de exemplo a seguir mostra como usar `DescribeTasks`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> <taskId>\s

                Where:
                  clusterArn - The ARN of an ECS cluster.
                  taskId - The task Id value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
```

```
        String taskId = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        getAllTasks(ecsClient, clusterArn, taskId);
        ecsClient.close();
    }

    public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
 taskId) {
        try {
            DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
                    .cluster(clusterArn)
                    .tasks(taskId)
                    .build();

            DescribeTasksResponse response = ecsClient.describeTasks(tasksRequest);
            List<Task> tasks = response.tasks();
            for (Task task : tasks) {
                System.out.println("The task ARN is " + task.taskDefinitionArn());
            }

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeTasks em Referência AWS SDK for Java 2.x da API.

`ListClusters`

O código de exemplo a seguir mostra como usar `ListClusters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        listAllClusters(ecsClient);
        ecsClient.close();
    }

    public static void listAllClusters(EcsClient ecsClient) {
        try {
            ListClustersResponse response = ecsClient.listClusters();
            List<String> clusters = response.clusterArns();
            for (String cluster : clusters) {
                System.out.println("The cluster arn is " + cluster);
            }
```

```
        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListClusters em Referência AWS SDK for Java 2.x da API.

`UpdateService`

O código de exemplo a seguir mostra como usar `UpdateService`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class UpdateService {

    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                   <clusterName> <serviceArn>\s

                Where:
                  clusterName - The cluster name.
                  serviceArn - The service ARN value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        updateSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void updateSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .desiredCount(0)
                    .build();

            ecsClient.updateService(serviceRequest);
            System.out.println("The service was modified");

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte UpdateService em Referência AWS SDK for Java 2.x da API.

Exemplos do Elastic Load Balancing Versão 2 usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Elastic Load Balancing - Versão 2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Elastic Load Balancing

O exemplo de código a seguir mostra como começar a usar o Elastic Load Balancing.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class HelloLoadBalancer {

        public static void main(String[] args) {
                ElasticLoadBalancingV2Client loadBalancingV2Client =
 ElasticLoadBalancingV2Client.builder()
                                .region(Region.US_EAST_1)
                                .build();

                DescribeLoadBalancersResponse loadBalancersResponse =
 loadBalancingV2Client
                                .describeLoadBalancers(r -> r.pageSize(10));
                List<LoadBalancer> loadBalancerList =
 loadBalancersResponse.loadBalancers();
                for (LoadBalancer lb : loadBalancerList)
                        System.out.println("Load Balancer DNS name = " +
 lb.dnsName());
        }
}
```

- Para obter detalhes da API, consulte DescribeLoadBalancers em Referência AWS SDK for Java 2.x da API.

Ações

`CreateListener`

O código de exemplo a seguir mostra como usar `CreateListener`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
```

```
    */
   public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                   .subnets(subnetIdStrings)
                   .name(lbName)
                   .scheme("internet-facing")
                   .build();

           // Create and wait for the load balancer to become available.
           CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
           String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(lbARN)
                   .build();

           System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancerAvailable(request);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("Load Balancer " + lbName + " is available.");

           // Get the DNS name (endpoint) of the load balancer.
           String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
           System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

           // Create a listener for the load balance.
           Action action = Action.builder()
                   .targetGroupArn(targetGroupARN)
                   .type("forward")
```

```
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateListener em Referência AWS SDK for Java 2.x da API.

`CreateLoadBalancer`

O código de exemplo a seguir mostra como usar `CreateLoadBalancer`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /*
```

```
    * Creates an Elastic Load Balancing load balancer that uses the specified
    * subnets
    * and forwards requests to the specified target group.
    */
   public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                   .subnets(subnetIdStrings)
                   .name(lbName)
                   .scheme("internet-facing")
                   .build();

           // Create and wait for the load balancer to become available.
           CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
           String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(lbARN)
                   .build();

           System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancerAvailable(request);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("Load Balancer " + lbName + " is available.");

           // Get the DNS name (endpoint) of the load balancer.
           String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
           System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

           // Create a listener for the load balance.
```

```
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateLoadBalancer em Referência AWS SDK for Java 2.x da API.

`CreateTargetGroup`

O código de exemplo a seguir mostra como usar `CreateTargetGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }
```

- Para obter detalhes da API, consulte CreateTargetGroup em Referência AWS SDK for Java 2.x da API.

`DeleteLoadBalancer`

O código de exemplo a seguir mostra como usar `DeleteLoadBalancer`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Deletes a load balancer.
   public void deleteLoadBalancer(String lbName) {
       try {
           // Use a waiter to delete the Load Balancer.
           DescribeLoadBalancersResponse res = getLoadBalancerClient()
                   .describeLoadBalancers(describe -> describe.names(lbName));
           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                   .build();

           getLoadBalancerClient().deleteLoadBalancer(
                   builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancersDeleted(request);
           waiterResponse.matched().response().ifPresent(System.out::println);

       } catch (ElasticLoadBalancingV2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
       System.out.println(lbName + " was deleted.");
   }
```

- Para obter detalhes da API, consulte DeleteLoadBalancer em Referência AWS SDK for Java 2.x da API.

`DeleteTargetGroup`

O código de exemplo a seguir mostra como usar `DeleteTargetGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

- Para obter detalhes da API, consulte DeleteTargetGroup em Referência AWS SDK for Java 2.x da API.

`DescribeTargetHealth`

O código de exemplo a seguir mostra como usar `DescribeTargetHealth`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Checks the health of the instances in the target group.
   public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
       DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
               .names(targetGroupName)
               .build();

       DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

       DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
               .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
               .build();

       DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
       return healthResponse.targetHealthDescriptions();
   }
```

- Para obter detalhes da API, consulte DescribeTargetHealth em Referência AWS SDK for Java 2.x da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
```

```
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
```

```
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }
```

```
   private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
       Scanner in = new Scanner(System.in);
       System.out.println(
               """
                       For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                       to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                       against various kinds of failures.

                       Some of the resources create by this demo are:
                       \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                       \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                       \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                       \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                       """);

       System.out.println("Press Enter when you're ready.");
       in.nextLine();
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Creating and populating a DynamoDB table named " +
tableName);
       Database database = new Database();
       database.createTable(tableName, fileName);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("""
               Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
               This script starts a Python web server defined in the `server.py`
script. The web server
               listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
               For demo purposes, this server is run as the root user. In
production, the best practice is to
               run a web server, such as Apache, with least-privileged credentials.
```

```
                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
```

```
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns a
static response.
                The service still reports as healthy because health checks are still
shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
```

```
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
```

```
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
       autoScaler.terminateInstance(badInstanceId);

       System.out.println("""
               Even while the instance is terminating and the new instance is
starting, sending a GET
               request to the web service continues to get a successful
recommendation response because
               the load balancer routes requests to the healthy instances. After
the replacement instance
               starts and reports as healthy, it is included in the load balancing
rotation.
               Note that terminating and replacing an instance typically takes
several minutes, during which time you
               can see the changing health check status until the new instance is
running and healthy.
               """);

       demoChoices(loadBalancer);
       System.out.println(
               "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       demoChoices(loadBalancer);
       paramHelper.reset();
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
           System.out.println("See the current state of the service by selecting
one of the following choices:");
           for (int i = 0; i < actions.length; i++) {
```

```
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
```

```
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
 targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                        target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
 check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {
```

```
    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
```

```
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();
```

```
        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
```

```
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);
```

```
            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();
```

```
            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();
```

```
        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();
```

```
        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();
```

```
        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.
```

```
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
```

```
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
```

```
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
```

```
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
```

```
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
```

```
    * Creates an Elastic Load Balancing load balancer that uses the specified
    * subnets
    * and forwards requests to the specified target group.
    */
   public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                   .subnets(subnetIdStrings)
                   .name(lbName)
                   .scheme("internet-facing")
                   .build();

           // Create and wait for the load balancer to become available.
           CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
           String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(lbARN)
                   .build();

           System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancerAvailable(request);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("Load Balancer " + lbName + " is available.");

           // Get the DNS name (endpoint) of the load balancer.
           String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
           System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

           // Create a listener for the load balance.
```

```
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }
```

```
    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
```

```
                                .attributeType(ScalarAttributeType.N)
                                .build())
                .keySchema(
                        KeySchemaElement.builder()
                                .attributeName("MediaType")
                                .keyType(KeyType.HASH)
                                .build(),
                        KeySchemaElement.builder()
                                .attributeName("ItemId")
                                .keyType(KeyType.RANGE)
                                .build())
                .provisionedThroughput(
                        ProvisionedThroughput.builder()
                                .readCapacityUnits(5L)
                                .writeCapacityUnits(5L)
                                .build())
                .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
```

```
    public void populateTable(String fileName, String tableName) throws IOException
 {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
```

```
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups

- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

MediaStore exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with MediaStore.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateContainer`

O código de exemplo a seguir mostra como usar `CreateContainer`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
```

```
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateContainer em Referência AWS SDK for Java 2.x da API.

`DeleteContainer`

O código de exemplo a seguir mostra como usar `DeleteContainer`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteContainer em Referência AWS SDK for Java 2.x da API.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName>

                Where:
                   completePath - The path (including the container) of the item to
 delete.
                   containerName - The name of the container.
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .path(completePath)
                    .build();

            mediaStoreData.deleteObject(deleteObjectRequest);

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();
```

```
        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- Para obter detalhes da API, consulte DeleteObject em Referência AWS SDK for Java 2.x da API.

`DescribeContainer`

O código de exemplo a seguir mostra como usar `DescribeContainer`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <containerName>

                Where:
                   containerName - The name of the container to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
DescribeContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            DescribeContainerResponse containerResponse =
mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
containerResponse.container().name());
            System.out.println("The container ARN is " +
containerResponse.container().arn());
            return containerResponse.container().status().toString();

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
```

```
    }
}
```

- Para obter detalhes da API, consulte DescribeContainer em Referência AWS SDK for Java 2.x da API.

GetObject

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName> <savePath>

                Where:
                   completePath - The path of the object in the container (for
 example, Videos5/sampleVideo.mp4).
                   containerName - The name of the container.
                   savePath - The path on the local drive where the file is saved,
 including the file name (for example, C:/AWS/myvid.mp4).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }

    public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
 completePath, String savePath) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .path(completePath)
                    .build();
```

```
            // Write out the data to a file.
            ResponseInputStream<GetObjectResponse> data =
 mediaStoreData.getObject(objectRequest);
            byte[] buffer = new byte[data.available()];
            data.read(buffer);

            File targetFile = new File(savePath);
            OutputStream outStream = new FileOutputStream(targetFile);
            outStream.write(buffer);
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK for Java 2.x da API.

`ListContainers`

O código de exemplo a seguir mostra como usar `ListContainers`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
 mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
```

```
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListContainers em Referência AWS SDK for Java 2.x da API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = """

                To run this example, supply the name of a container, a file location
 to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
```

```
                    .path(completePath)
                    .contentType("video/mp4")
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK for Java 2.x da API.

AWS Entity Resolution exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS Entity Resolution.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Entity Resolution

O exemplo de código a seguir mostra como começar a usar o AWS Entity Resolution.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloEntityResoultion {
```

```
   private static final Logger logger =
LoggerFactory.getLogger(HelloEntityResoultion.class);

   private static EntityResolutionAsyncClient entityResolutionAsyncClient;
   public static void main(String[] args) {
       listMatchingWorkflows();
   }

   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
           and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return entityResolutionAsyncClient;
```

```
    }

    /**
     * Lists all matching workflows using an asynchronous paginator.
     * <p>
     * This method requests a paginated list of matching workflows from the
     * AWS Entity Resolution service and logs the names of the retrieved workflows.
     * It uses an asynchronous approach with a paginator and waits for the operation
     * to complete using {@code CompletableFuture#join()}.
     * </p>
     */
    public static void listMatchingWorkflows() {
        ListMatchingWorkflowsRequest request =
 ListMatchingWorkflowsRequest.builder().build();

        ListMatchingWorkflowsPublisher paginator =
            getResolutionAsyncClient().listMatchingWorkflowsPaginator(request);

        // Iterate through the paginated results asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.workflowSummaries().forEach(workflow ->
                logger.info("Matching Workflow Name: " + workflow.workflowName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- Para obter detalhes da API, consulte ListMatchingWorkflows em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie mapeamento de esquemas.
- Crie um AWS Entity Resolution fluxo de trabalho.

- Inicie o trabalho correspondente para o fluxo de trabalho.
- Obtenha detalhes do trabalho correspondente.
- Obtenha o mapeamento de esquemas.
- Liste todos os mapeamentos de esquemas.
- Marque o recurso de mapeamento de esquemas.
- Exclua os AWS Entity Resolution ativos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS Entity Resolution recursos.

```
public class EntityResScenario {
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResScenario.class);
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "EntityResolutionCdkStack2";
    private static final String ENTITY_RESOLUTION_ROLE_ARN_KEY =
 "EntityResolutionRoleArn";
    private static final String GLUE_DATA_BUCKET_NAME_KEY = "GlueDataBucketName";
    private static final String JSON_GLUE_TABLE_ARN_KEY = "JsonErGlueTableArn";
    private static final String CSV_GLUE_TABLE_ARN_KEY = "CsvErGlueTableArn";
    private static String glueBucketName;
    private static String workflowName = "workflow-" + UUID.randomUUID();

    private static String jsonSchemaMappingName = "jsonschema-" + UUID.randomUUID();
    private static String jsonSchemaMappingArn = null;
    private static String csvSchemaMappingName = "csv-" + UUID.randomUUID();
    private static String roleARN;
    private static String csvGlueTableArn;
    private static String jsonGlueTableArn;
    private static Scanner scanner = new Scanner(System.in);

    private static EntityResActions actions = new EntityResActions();
```

```
    public static void main(String[] args) throws InterruptedException {

        logger.info("Welcome to the AWS Entity Resolution Scenario.");
        logger.info("""
            AWS Entity Resolution is a fully-managed machine learning service
provided by
            Amazon Web Services (AWS) that helps organizations extract, link, and
            organize information from multiple data sources. It leverages natural
            language processing and deep learning models to identify and resolve
            entities, such as people, places, organizations, and products,
            across structured and unstructured data.

            With Entity Resolution, customers can build robust data integration
            pipelines to combine and reconcile data from multiple systems,
databases,
            and documents. The service can handle ambiguous, incomplete, or
conflicting
            information, and provide a unified view of entities and their
relationships.
            This can be particularly valuable in applications such as customer 360,
            fraud detection, supply chain management, and knowledge management,
where
            accurate entity identification is crucial.

            The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x
            provides a set of methods to programmatically interact with the AWS
Entity
            Resolution service. This allows developers to automate the entity
extraction,
            linking, and deduplication process as part of their data processing
workflows.
            With Entity Resolution, organizations can unlock the value of their
data,
            improve decision-making, and enhance customer experiences by having a
reliable,
            comprehensive view of their key entities.
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("""
```

```
            To prepare the AWS resources needed for this scenario application, the
next step uploads
            a CloudFormation template whose resulting stack creates the following
resources:
            - An AWS Glue Data Catalog table
            - An AWS IAM role
            - An AWS S3 bucket
            - An AWS Entity Resolution Schema

            It can take a couple minutes for the Stack to finish creating the
resources.
            """);
        waitForInputToContinue(scanner);
        logger.info("Generating resources...");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);
        Map<String, String> outputsMap =
CloudFormationHelper.getStackOutputsAsync(STACK_NAME).join();
        roleARN = outputsMap.get(ENTITY_RESOLUTION_ROLE_ARN_KEY);
        glueBucketName = outputsMap.get(GLUE_DATA_BUCKET_NAME_KEY);
        csvGlueTableArn = outputsMap.get(CSV_GLUE_TABLE_ARN_KEY);
        jsonGlueTableArn = outputsMap.get(JSON_GLUE_TABLE_ARN_KEY);
        logger.info(DASHES);
        waitForInputToContinue(scanner);

        try {
            runScenario();

        } catch (Exception ce) {
            Throwable cause = ce.getCause();
            logger.error("An exception happened: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
        }
    }

    private static void runScenario() throws InterruptedException {
        /*
         This JSON is a valid input for the AWS Entity Resolution service.
         The JSON represents an array of three objects, each containing an "id",
"name", and "email"
         property. This format aligns with the expected input structure for the
         Entity Resolution service.
         */
        String json = """
            {"id":"1","name":"Jane Doe","email":"jane.doe@example.com"}
```

```
            {"id":"2","name":"John Doe","email":"john.doe@example.com"}
            {"id":"3","name":"Jorge Souza","email":"jorge_souza@example.com"}
            """;
        logger.info("Upload the following JSON objects to the {} S3 bucket.",
glueBucketName);
        logger.info(json);
        String csv = """
            id,name,email,phone
            1,Jane B.,Doe,jane.doe@example.com,555-876-9846
            2,John Doe Jr.,john.doe@example.com,555-654-3210
            3,María García,maría_garcia@company.com,555-567-1234
            4,Mary Major,mary_major@company.com,555-222-3333
            """;
        logger.info("Upload the following CSV data to the {} S3 bucket.",
glueBucketName);
        logger.info(csv);
        waitForInputToContinue(scanner);
        try {
            actions.uploadInputData(glueBucketName, json, csv);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to upload input data: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ResourceNotFoundException) {
                logger.error("Failed to upload input data as the resource was not
found: {}", cause.getMessage(), cause);
            }
            return;
        }
        logger.info("The JSON and CSV objects have been uploaded to the S3
bucket.");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create Schema Mapping");
        logger.info("""
            Entity Resolution schema mapping aligns and integrates data from
            multiple sources by identifying and matching corresponding entities
            like customers or products. It unifies schemas, resolves conflicts,
```

```
            and uses machine learning to link related entities, enabling a
            consolidated, accurate view for improved data quality and decision-
making.

            In this example, the schema mapping lines up with the fields in the JSON
 and CSV objects. That is,
            it contains these fields: id, name, and email.
            """);
        try {
            CreateSchemaMappingResponse response =
 actions.createSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingName = response.schemaName();
            logger.info("The JSON schema mapping name is " + jsonSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to create JSON schema mapping: {}",
 ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
 cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating schema mapping: {}",
 cause.getMessage(), cause);
            }
            return;
        }

        try {
            CreateSchemaMappingResponse response =
 actions.createSchemaMappingAsync(csvSchemaMappingName).join();
            csvSchemaMappingName = response.schemaName();
            logger.info("The CSV schema mapping name is " + csvSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause == null) {
                logger.error("Failed to create CSV schema mapping: {}",
 ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
```

```
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating CSV schema mapping:
{}", cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Entity Resolution Workflow. ");
        logger.info("""
            An Entity Resolution matching workflow identifies and links records
            across datasets that represent the same real-world entity, such as
            customers or products. Using techniques like schema mapping,
            data profiling, and machine learning algorithms,
            it evaluates attributes like names or emails to detect duplicates
            or relationships, even with variations or inconsistencies.
            The workflow outputs consolidated, de-duplicated data.

            We will use the machine learning-based matching technique.
            """);
        waitForInputToContinue(scanner);
        try {
            String workflowArn = actions.createMatchingWorkflowAsync(
                roleARN, workflowName, glueBucketName, jsonGlueTableArn,
                jsonSchemaMappingName, csvGlueTableArn,
csvSchemaMappingName).join();

            logger.info("The workflow ARN is: " + workflowArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("An unexpected error occurred: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ValidationException) {
                logger.error("Validation error: {}", cause.getMessage(), cause);
            } else if (cause instanceof ConflictException) {
```

```
                logger.error("Workflow conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error: {}", cause.getMessage(), cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        logger.info("3. Start the matching job of the " + workflowName + "
workflow.");
        waitForInputToContinue(scanner);
        String jobId = null;
        try {
            jobId = actions.startMatchingJobAsync(workflowName).join();
            logger.info("The matching job was successfully started.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("Job conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error while starting the job: {}",
ce.getMessage(), ce);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. While the matching job is running, let's look at other API
methods. First, let's get details for job " + jobId);
        waitForInputToContinue(scanner);
        try {
            actions.getMatchingJobAsync(jobId, workflowName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The matching job not found: {}", cause.getMessage(),
cause);
            } else {
```

```
                logger.error("Failed to start matching job: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get the schema mapping for the JSON data.");
        waitForInputToContinue(scanner);
        try {
            GetSchemaMappingResponse response =
actions.getSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingArn = response.schemaArn();
            logger.info("Schema mapping ARN is " + jsonSchemaMappingArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("Schema mapping not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Error retrieving the specific schema mapping: " +
ce.getCause().getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List Schema Mappings.");
        try {
            actions.ListSchemaMappings();
        } catch (CompletionException ce) {
            logger.error("Error retrieving schema mappings: " +
ce.getCause().getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Tag the {} resource.", jsonSchemaMappingName);
        logger.info("""
```

```
            Tags can help you organize and categorize your Entity Resolution
resources.
            You can also use them to scope user permissions by granting a user
permission
            to access or change only resources with certain tag values.
            In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged.
For this example,
            the SchemaMapping is tagged.
                """);
        try {
            actions.tagEntityResource(jsonSchemaMappingArn).join();
        } catch (CompletionException ce) {
            logger.error("Error tagging the resource: " +
ce.getCause().getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. View the results of the AWS Entity Resolution Workflow.");
        logger.info("""
            You cannot view the result of the workflow that is in a running state.
            In order to view the results, you need to wait for the workflow that we
started in step 3 to complete.

            If you choose not to wait, you cannot view the results. You can perform

            this task manually in the AWS Management Console.

            This can take up to 30 mins (y/n).
            """);
        String viewAns = scanner.nextLine().trim();
        boolean isComplete = false;
        if (viewAns.equalsIgnoreCase("y")) {
            logger.info("You selected to view the Entity Resolution Workflow
results.");
            countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
            isComplete = true;
            try {
                JobMetrics metrics = actions.getJobInfo(workflowName, jobId).join();
                logger.info("Number of input records: {}", metrics.inputRecords());
                logger.info("Number of match ids: {}", metrics.matchIDs());
```

```
                logger.info("Number of records not processed: {}",
metrics.recordsNotProcessed());
                logger.info("Number of total records processed: {}",
metrics.totalRecordsProcessed());
                logger.info("The following represents the output data generated by
the Entity Resolution workflow based on the JSON and CSV input data. The output
data is stored in the {} bucket.", glueBucketName);
                actions.printData(glueBucketName);

                logger.info("""

                    Note that each of the last 2 records are considered a match even
though the 'name' differs between the records;
                    For example 'John Doe Jr.' compared to 'John Doe'.
                    The confidence level is a value between 0 and 1, where 1
indicates a perfect match.

                    """);

            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The job not found: {}", cause.getMessage(),
cause);
                } else {
                    logger.error("Error retrieving job information: " +
ce.getCause().getMessage());
                }
                return;
            }
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Do you want to delete the resources, including the workflow?
(y/n)");
        logger.info("""
            You cannot delete the workflow that is in a running state.
            In order to delete the workflow, you need to wait for the workflow to
complete.
```

```
            You can delete the workflow manually in the AWS Management Console at a
later time.

            If you already waited for the workflow to complete in the previous
step,
            the workflow is completed and you can delete it.

            If the workflow is not completed, this can take up to 30 mins (y/n).
            """);
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                if (!isComplete) {
                    countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
                }
                actions.deleteMatchingWorkflowAsync(workflowName).join();
                logger.info("Workflow deleted successfully!");
            } catch (CompletionException ce) {
                logger.info("Error deleting the workflow: {} ", ce.getMessage());
                return;
            }

            try {
                // Delete both schema mappings.
                actions.deleteSchemaMappingAsync(jsonSchemaMappingName).join();
                actions.deleteSchemaMappingAsync(csvSchemaMappingName).join();
                logger.info("Both schema mappings were deleted successfully!");
            } catch (CompletionException ce) {
                logger.error("Error deleting schema mapping: {}", ce.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            logger.info(DASHES);
            logger.info("""
                Now we delete the CloudFormation stack, which deletes
                the resources that were created at the beginning of this scenario.
                """);
            waitForInputToContinue(scanner);
            logger.info(DASHES);
            try {
                deleteCloudFormationStack();
            } catch (RuntimeException e) {
                logger.error("Failed to delete the stack: {}", e.getMessage());
```

```
                return;
            }

        } else {
            logger.info("You can delete the AWS resources in the AWS Management
Console.");
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the AWS Entity Resolution scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdownWithWorkflowCheck(EntityResActions actions, int
totalSeconds, String jobId, String workflowName) throws InterruptedException {
        int secondsElapsed = 0;

        while (true) {
            // Calculate display minutes and seconds.
            int remainingTime = totalSeconds - secondsElapsed;
            int displayMinutes = remainingTime / 60;
            int displaySeconds = remainingTime % 60;

            // Print the countdown.
```

```
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
            secondsElapsed++;

            // Check workflow status every 60 seconds.
            if (secondsElapsed % 60 == 0 || remainingTime <= 0) {
                GetMatchingJobResponse response =
 actions.checkWorkflowStatusCompleteAsync(jobId, workflowName).join();
                if (response != null &&
 "SUCCEEDED".equalsIgnoreCase(String.valueOf(response.status()))) {
                    logger.info(""); // Move to the next line after countdown.
                    logger.info("Countdown complete: Workflow is in Completed
 state!");
                    break; // Break out of the loop if the status is "SUCCEEDED"
                }
            }

            // If countdown reaches zero, reset it for continuous countdown.
            if (remainingTime <= 0) {
                secondsElapsed = 0;
            }
        }
    }

    private static void deleteCloudFormationStack() {
        try {
            CloudFormationHelper.emptyS3Bucket(glueBucketName);
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
            logger.info("Resources deleted successfully!");
        } catch (CloudFormationException e) {
            throw new RuntimeException("Failed to delete CloudFormation stack: " +
 e.getMessage(), e);
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to empty S3 bucket: " +
 e.getMessage(), e);
        }
    }
}
```

Uma classe de invólucro para métodos AWS Entity Resolution SDK.

```
public class EntityResActions {
```

```
   private static final String PREFIX = "eroutput/";
   private static final Logger logger =
LoggerFactory.getLogger(EntityResActions.class);

   private static EntityResolutionAsyncClient entityResolutionAsyncClient;

   private static S3AsyncClient s3AsyncClient;

   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
           and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
```

```
        return entityResolutionAsyncClient;
    }

    public static S3AsyncClient getS3AsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }

    /**
     * Deletes the schema mapping asynchronously.
     *
```

```
    * @param schemaName the name of the schema to delete
    * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
       DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().deleteSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully deleted the schema mapping, log the success
message.
                   logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
               } else {
                   // Ensure exception is not null before accessing its cause.
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
               }
           });
   }

   /**
    * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
    * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
    */
```

```
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();

        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }

    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
    public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
        DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().deleteMatchingWorkflow(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("{} was deleted", workflowName );
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                    }
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
            );
        }
```

```
        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Retrieves the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to retrieve the mapping for
     * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
     * is complete
     * @throws RuntimeException if the schema mapping retrieval fails
     */
    public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
        GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();
```

```
        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema mapping
was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
     */
    public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
```

```
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully fetched the matching job details, log the job
status.
                    logger.info("Job status: " + response.status());
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could not
be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the matching
job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
```

```
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }

    /**
     * Checks the status of a workflow asynchronously.
     *
     * @param jobId        the ID of the job to check
     * @param workflowName the name of the workflow to check
     * @return a CompletableFuture that resolves to a boolean value indicating
whether the workflow has completed
     * successfully
     */
    public CompletableFuture<GetMatchingJobResponse>
checkWorkflowStatusCompleteAsync(String jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Process the response and log the job status.
                    logger.info("Job status: " + response.status());
                } else {
```

```
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while checking job status.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested resource was
not found while checking the job status.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to check job status: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
     *
     * @param roleARN                 the AWS IAM role ARN to be used for the
workflow execution
     * @param workflowName            the name of the workflow to be created
     * @param outputBucket            the S3 bucket path where the workflow output
will be stored
     * @param jsonGlueTableArn        the ARN of the Glue Data Catalog table to be
used as the input source
     * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
     * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
     */
    public CompletableFuture<String> createMatchingWorkflowAsync(
        String roleARN
        , String workflowName
        , String outputBucket
        , String jsonGlueTableArn
        , String jsonErSchemaMappingName
        , String csvGlueTableArn
        , String csvErSchemaMappingName) {
```

```
        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
```

```
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }

    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
```

```
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was not
found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
                }
            });
    }

    public CompletableFuture<JobMetrics> getJobInfo(String workflowName, String
jobId) {
        return getResolutionAsyncClient().getMatchingJob(b -> b
                .workflowName(workflowName)
                .jobId(jobId))
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Job metrics fetched successfully for jobId: " +
jobId);
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("Invalid request: Job id was
not found.", cause);
                    }
                    throw new CompletionException("Failed to fetch job info: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.metrics()); // Extract job metrics
    }

    /**
```

```
    * Uploads data to an Amazon S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to upload the data to
    * @param jsonData   the JSON data to be uploaded
    * @param csvData    the CSV data to be uploaded
    * @return a {@link CompletableFuture} representing both asynchronous operation
of uploading the data
    * @throws RuntimeException if an error occurs during the file upload
    */

   public void uploadInputData(String bucketName, String jsonData, String csvData)
{
       // Upload JSON data.
       String jsonKey = "jsonData/data.json";
       PutObjectRequest jsonUploadRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(jsonKey)
           .contentType("application/json")
           .build();

       CompletableFuture<PutObjectResponse> jsonUploadResponse =
getS3AsyncClient().putObject(jsonUploadRequest,
AsyncRequestBody.fromString(jsonData));

       // Upload CSV data.
       String csvKey = "csvData/data.csv";
       PutObjectRequest csvUploadRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(csvKey)
           .contentType("text/csv")
           .build();
       CompletableFuture<PutObjectResponse> csvUploadResponse =
getS3AsyncClient().putObject(csvUploadRequest,
AsyncRequestBody.fromString(csvData));

       CompletableFuture.allOf(jsonUploadResponse, csvUploadResponse)
           .whenComplete((result, ex) -> {
               if (ex != null) {
                   // Wrap an AWS exception.
                   throw new CompletionException("Failed to upload files", ex);
               }
           }).join();

   }
```

```
    /**
     * Finds the latest file in the S3 bucket that starts with "run-" in any depth
of subfolders
     */
    private CompletableFuture<String> findLatestMatchingFile(String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(PREFIX) // Searches within the given folder
            .build();

        return getS3AsyncClient().listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .filter(key -> key.matches(".*?/run-[0-9a-zA-Z\\-]+")) // Matches
files like run-XXXXX in any subfolder
                .max(String::compareTo) // Gets the latest file
                .orElse(null))
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Latest matching file found: " + result);
                    } else {
                        logger.info("No matching files found.");
                    }
                } else {
                    throw new CompletionException("Failed to find latest matching
file: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prints the data located in the file in the S3 bucket that starts with "run-"
in any depth of subfolders
     */
    public void printData(String bucketName) {
        try {
            // Find the latest file with "run-" prefix in any depth of subfolders.
            String s3Key = findLatestMatchingFile(bucketName).join();
            if (s3Key == null) {
                logger.error("No matching files found in S3.");
                return;
            }
```

```
            logger.info("Downloading file: " + s3Key);

            // Read CSV file as String.
            String csvContent = readCSVFromS3Async(bucketName, s3Key).join();
            if (csvContent.isEmpty()) {
                logger.error("File is empty.");
                return;
            }

            // Process CSV content.
            List<String[]> records = parseCSV(csvContent);
            printTable(records);

        } catch (RuntimeException | IOException | CsvException e) {
            logger.error("Error processing CSV file from S3: " + e.getMessage());
            e.printStackTrace();
        }
    }

    /**
     * Reads a CSV file from S3 and returns it as a String.
     */
    private static CompletableFuture<String> readCSVFromS3Async(String bucketName,
String s3Key) {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(s3Key)
            .build();

        // Initiating the asynchronous request to get the file as bytes
        return getS3AsyncClient().getObject(getObjectRequest,
AsyncResponseTransformer.toBytes())
            .thenApply(responseBytes -> responseBytes.asUtf8String()) // Convert
bytes to UTF-8 string
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Failed to read CSV from S3: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Successfully fetched CSV file content from S3.");
                }
            });
    }
```

```
    /**
     * Parses CSV content from a String into a list of records.
     */
    private static List<String[]> parseCSV(String csvContent) throws IOException,
CsvException {
        try (CSVReader csvReader = new CSVReader(new StringReader(csvContent))) {
            return csvReader.readAll();
        }
    }

    /**
     * Prints the given CSV data in a formatted table
     */
    private static void printTable(List<String[]> records) {
        if (records.isEmpty()) {
            System.out.println("No records found.");
            return;
        }

        String[] headers = records.get(0);
        List<String[]> rows = records.subList(1, records.size());

        // Determine column widths dynamically based on longest content
        int[] columnWidths = new int[headers.length];
        for (int i = 0; i < headers.length; i++) {
            final int columnIndex = i;
            int maxWidth = Math.max(headers[i].length(), rows.stream()
                .map(row -> row.length > columnIndex ? row[columnIndex].length() :
0)
                .max(Integer::compareTo)
                .orElse(0));
            columnWidths[i] = Math.min(maxWidth, 25); // Limit max width for better
readability
        }

        // Enable ANSI Console for colored output
        AnsiConsole.systemInstall();

        // Print table header
        System.out.println(ansi().fgYellow().a("=== CSV Data from S3 ===").reset());
        printRow(headers, columnWidths, true);

        // Print rows
```

```
        rows.forEach(row -> printRow(row, columnWidths, false));

        // Restore console to normal
        AnsiConsole.systemUninstall();
    }

    private static void printRow(String[] row, int[] columnWidths, boolean isHeader)
 {
        String border = IntStream.range(0, columnWidths.length)
            .mapToObj(i -> "-".repeat(columnWidths[i] + 2))
            .collect(Collectors.joining("+", "+", "+"));

        if (isHeader) {
            System.out.println(border);
        }

        System.out.print("|");
        for (int i = 0; i < columnWidths.length; i++) {
            String cell = (i < row.length && row[i] != null) ? row[i] : "";
            System.out.printf(" %-" + columnWidths[i] + "s |", isHeader ?
 ansi().fgBrightBlue().a(cell).reset() : cell);
        }
        System.out.println();

        if (isHeader) {
            System.out.println(border);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateMatchingWorkflow
  - CreateSchemaMapping
  - DeleteMatchingWorkflow
  - DeleteSchemaMapping
  - GetMatchingJob
  - GetSchemaMapping
  - ListMatchingWorkflows

- ListSchemaMappings
- StartMatchingJob
- TagResource

Ações

`CreateMatchingWorkflow`

O código de exemplo a seguir mostra como usar `CreateMatchingWorkflow`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
    *
    * @param roleARN                 the AWS IAM role ARN to be used for the
workflow execution
    * @param workflowName            the name of the workflow to be created
    * @param outputBucket            the S3 bucket path where the workflow output
will be stored
    * @param jsonGlueTableArn        the ARN of the Glue Data Catalog table to be
used as the input source
    * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
    * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
    */
   public CompletableFuture<String> createMatchingWorkflowAsync(
       String roleARN
       , String workflowName
       , String outputBucket
       , String jsonGlueTableArn
       , String jsonErSchemaMappingName
       , String csvGlueTableArn
```

```
        , String csvErSchemaMappingName) {

        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
```

```
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }
```

- Para obter detalhes da API, consulte CreateMatchingWorkflow em Referência AWS SDK for Java 2.x da API.

`CreateSchemaMapping`

O código de exemplo a seguir mostra como usar `CreateSchemaMapping`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to create
    * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
    */
   public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
       List<SchemaInputAttribute> schemaAttributes = null;
       if (schemaName.startsWith("json")) {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
           );
       } else {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
           );
       }

       CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
           .schemaName(schemaName)
```

```
            .mappedInputFields(schemaAttributes)
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateSchemaMapping em Referência AWS SDK for Java 2.x da API.

`DeleteMatchingWorkflow`

O código de exemplo a seguir mostra como usar `DeleteMatchingWorkflow`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously deletes a workflow with the specified name.
    *
    * @param workflowName the name of the workflow to be deleted
    * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
    * @throws RuntimeException if the deletion of the workflow fails
    */
   public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
       DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().deleteMatchingWorkflow(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("{} was deleted", workflowName );
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The workflow to delete was
not found.", cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
               }
           });
   }
```

- Para obter detalhes da API, consulte DeleteMatchingWorkflow em Referência AWS SDK for Java 2.x da API.

DeleteSchemaMapping

O código de exemplo a seguir mostra como usar `DeleteSchemaMapping`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes the schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to delete
    * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
       DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().deleteSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully deleted the schema mapping, log the success
message.
                   logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
               } else {
                   // Ensure exception is not null before accessing its cause.
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
```

```
                        throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteSchemaMapping em Referência AWS SDK for Java 2.x da API.

`GetMatchingJob`

O código de exemplo a seguir mostra como usar `GetMatchingJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
     */
    public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
```

```
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully fetched the matching job details, log the job
status.
                    logger.info("Job status: " + response.status());
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could not
be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte GetMatchingJob em Referência AWS SDK for Java 2.x da API.

GetSchemaMapping

O código de exemplo a seguir mostra como usar `GetSchemaMapping`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to retrieve the mapping for
    * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
    * is complete
    * @throws RuntimeException if the schema mapping retrieval fails
    */
   public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
       GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   response.mappedInputFields().forEach(attribute ->
                       logger.info("Attribute Name: " + attribute.fieldName() +
                           ", Attribute Type: " + attribute.type().toString()));
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested schema mapping
was not found.", cause);
                   }
```

```
                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte GetSchemaMapping em Referência AWS SDK for Java 2.x da API.

`ListSchemaMappings`

O código de exemplo a seguir mostra como usar `ListSchemaMappings`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
     * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
     */
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();

        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
```

```
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
```

- Para obter detalhes da API, consulte ListSchemaMappings em Referência AWS SDK for Java 2.x da API.

StartMatchingJob

O código de exemplo a seguir mostra como usar `StartMatchingJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the matching
job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
```

```
                if (response != null) {
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }
```

- Para obter detalhes da API, consulte StartMatchingJob em Referência AWS SDK for Java 2.x da API.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Tags the specified schema mapping ARN.
```

```
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was not
found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte TagResource em Referência AWS SDK for Java 2.x da API.

OpenSearch Exemplos de serviços usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with OpenSearch Service.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello OpenSearch Service

O exemplo de código a seguir mostra como começar a usar o OpenSearch Service.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.ListVersionsRequest;
import java.util.List;
import java.util.concurrent.CompletableFuture;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloOpenSearch {
    public static void main(String[] args) {
        try {
            CompletableFuture<Void> future = listVersionsAsync();
            future.join();
            System.out.println("Versions listed successfully.");
        } catch (RuntimeException e) {
            System.err.println("Error occurred while listing versions: " +
 e.getMessage());
        }
    }

    private static OpenSearchAsyncClient getAsyncClient() {
        return OpenSearchAsyncClient.builder().build();
    }

    public static CompletableFuture<Void> listVersionsAsync() {
        ListVersionsRequest request = ListVersionsRequest.builder()
            .maxResults(10)
            .build();

        return getAsyncClient().listVersions(request).thenAccept(response -> {
            List<String> versionList = response.versions();
            for (String version : versionList) {
                System.out.println("Version info: " + version);
            }
        }).exceptionally(ex -> {
            // Handle the exception, or propagate it as a RuntimeException
            throw new RuntimeException("Failed to list versions", ex);
        });
    }
}
```

- Para obter detalhes da API, consulte ListVersions em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Saiba mais sobre as principais operações do OpenSearch Service

O exemplo de código a seguir mostra como:

- Crie um domínio OpenSearch de serviço.
- Fornece informações detalhadas sobre um domínio OpenSearch de serviço específico.
- Lista todos os domínios OpenSearch de serviço pertencentes à conta.
- Espera até que o status de alteração do domínio do OpenSearch serviço atinja um estado completo.
- Modifica a configuração de um domínio de OpenSearch serviço existente.
- Adicione uma tag ao domínio do OpenSearch serviço.
- Lista as tags associadas a um domínio OpenSearch de serviço.
- Remove tags de um domínio OpenSearch de serviço.
- Exclui o domínio do OpenSearch serviço.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos OpenSearch do serviço.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.opensearch.model.*;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

public class OpenSearchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static OpenSearchActions openSearchActions = new OpenSearchActions();

    public static void main(String[] args) throws Throwable {
        logger.info("""
            Welcome to the Amazon OpenSearch Service Basics Scenario.

            Use the Amazon OpenSearch Service API to create, configure, and manage
 OpenSearch Service domains.

            The operations exposed by the AWS OpenSearch Service client are focused
 on managing the OpenSearch Service domains
            and their configurations, not the data within the domains (such as
 indexing or querying documents).
            For document management, you typically interact directly with the
 OpenSearch REST API or use other libraries,
            such as the OpenSearch Java client (https://opensearch.org/docs/latest/
clients/java/).

            Let's get started...
        """);
        waitForInputToContinue(scanner);
        try {
            runScenario();
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
```

```
            }
        }
    }

    private static void runScenario() throws Throwable {
        String currentTimestamp = String.valueOf(System.currentTimeMillis());
        String domainName = "test-domain-" + currentTimestamp;

        logger.info(DASHES);
        logger.info("1. Create an Amazon OpenSearch domain");
        logger.info("""
            An Amazon OpenSearch domain is a managed instance of the OpenSearch
engine,
            which is an open-source search and analytics engine derived from
Elasticsearch.
            An OpenSearch domain is essentially a cluster of compute resources and
storage that hosts
            one or more OpenSearch indexes, enabling you to perform full-text
searches, data analysis, and
            visualizations.

            In this step, we'll initiate the creation of the domain. We'll check on
the progress in a later step.
        """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
openSearchActions.createNewDomainAsync(domainName);
            String domainId = future.join();
            logger.info("Domain successfully created with ID: {}", domainId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause != null) {
                if (cause instanceof OpenSearchException openSearchEx) {
                    logger.error("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
                } else {
                    logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage());
```

```
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("2. Describe the Amazon OpenSearch domain");
        logger.info("In this step, we get back the Domain ARN which is used in an
upcoming step.");
        waitForInputToContinue(scanner);

        String arn = "";
        try {
            CompletableFuture<String> future =
openSearchActions.describeDomainAsync(domainName);
            arn = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("3. List the domains in your account");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<List<DomainInfo>> future =
openSearchActions.listAllDomainsAsync();
            List<DomainInfo> domainInfoList = future.join();
            for (DomainInfo domain : domainInfoList) {
                logger.info("Domain name is: " + domain.domainName());
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
```

```
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we check on the change status of the domain that we
initiated in Step 1.
            Until we reach a COMPLETED state, we stay in a loop by sending a
DescribeDomainChangeProgressRequest.

            The time it takes for a change to an OpenSearch domain to reach a
completed state can range
            from a few minutes to several hours. In this case the change is creating
a new domain that we initiated in Step 1.
            The time varies depending on the complexity of the change and the
current load on
            the OpenSearch service. In general, simple changes, such as scaling the
number of data nodes or
            updating the OpenSearch version, may take 10-30 minutes.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
```

```
            while (cause.getCause() != null && !(cause instanceof
ResourceNotFoundException)) {
                cause = cause.getCause();
            }
            if (cause instanceof ResourceNotFoundException
resourceNotFoundException) {
                logger.info("The specific AWS resource was not found: Error message:
{}, Error code {}", resourceNotFoundException.awsErrorDetails().errorMessage(),
resourceNotFoundException.awsErrorDetails().errorCode());

                if (cause instanceof OpenSearchException ex) {
                    logger.info("An OpenSearch error occurred: Error message: " +
ex.getMessage());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Modify the domain");
        logger.info("""
            You can change your OpenSearch domain's settings, like the number of
instances, without starting over from scratch.
            This makes it easy to adjust your domain as your needs change, allowing
you to scale up or
            down quickly without recreating everything.

            We modify the domain in this step by changing the number of instances.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<UpdateDomainConfigResponse> future =
openSearchActions.updateSpecificDomainAsync(domainName);
            UpdateDomainConfigResponse updateResponse = future.join();
            logger.info("Domain update status: " +
updateResponse.domainConfig().changeProgressDetails().configChangeStatusAsString());
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
```

```
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we poll the status until the domain's change status
reaches a completed state.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException ex) {
                logger.info("EC2 error occurred: Error message: " +ex.getMessage());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("7. Tag the Domain");
        logger.info("""
            Tags let you assign arbitrary information to an Amazon OpenSearch
Service domain so you can
            categorize and filter on that information. A tag is a key-value pair
that you define and
```

```
            associate with an OpenSearch Service domain. You can use these tags to
track costs by grouping
            expenses for similarly tagged resources.

            In this scenario, we create tags with keys "service" and "instances".
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<AddTagsResponse> future =
openSearchActions.addDomainTagsAsync(arn);
            future.join();
            logger.info("Domain tags added successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                if (cause != null) {
                    if (cause instanceof OpenSearchException) {
                        logger.error("OpenSearch error occurred: Error message: " +
cause.getMessage(), cause);
                    } else {
                        logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                    }
                } else {
                    logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info("8. List Domain tags");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<ListTagsResponse> future =
openSearchActions.listDomainTagsAsync(arn);
            ListTagsResponse listTagsResponse = future.join();
            listTagsResponse.tagList().forEach(tag -> logger.info("Tag Key: " +
tag.key() + ", Tag Value: " + tag.value()));
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Delete the domain");
        logger.info("""
            In this step, we'll delete the Amazon OpenSearch domain that we created
in Step 1.
            Deleting a domain will remove all data and configuration for that
domain.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<DeleteDomainResponse> future =
openSearchActions.deleteSpecificDomainAsync(domainName);
            future.join();
            logger.info("Domain successfully deleted.");
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
 OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
 {}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
 openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("Scenario complete!");
    }
 }
```

Uma classe de invólucro para métodos OpenSearch do Service SDK.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.AddTagsRequest;
import software.amazon.awssdk.services.opensearch.model.AddTagsResponse;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressRequest;
```

```
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.DomainStatus;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsResponse;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.Tag;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class OpenSearchActions {
    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchActions.class);
    private static OpenSearchAsyncClient openSearchClientAsyncClient;
    private static OpenSearchAsyncClient getAsyncClient() {
        if (openSearchClientAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            openSearchClientAsyncClient = OpenSearchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
```

```
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return openSearchClientAsyncClient;
    }

    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
```

```
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }

    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has been
deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }

    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the domain
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
```

```
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }

    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
    }
```

```
    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }

    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
            long startTime = System.currentTimeMillis();
```

```
            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response = getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check domain
progress", ex);
                            }
                            return resp;
                        }).join();

                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }

    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
```

```
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();

       return getAsyncClient().addTags(addTagsRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
               } else {
                   logger.info("Added Tags");
               }
           });
   }


   /**
    * Asynchronously lists the tags associated with the specified Amazon Resource
Name (ARN).
    * @param arn the Amazon Resource Name (ARN) of the resource for which to list
the tags
```

```
     * @return a {@link CompletableFuture} that, when completed, will contain a list
 of the tags associated with the
     * specified ARN
     * @throws RuntimeException if there is an error listing the tags
     */
    public CompletableFuture<ListTagsResponse> listDomainTagsAsync(String arn) {
        ListTagsRequest tagsRequest = ListTagsRequest.builder()
            .arn(arn)
            .build();

        return getAsyncClient().listTags(tagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list domain tags",
 exception);
                }

                List<Tag> tagList = response.tagList();
                for (Tag tag : tagList) {
                    logger.info("Tag key is " + tag.key());
                    logger.info("Tag value is " + tag.value());
                }
            });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AddTags
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - DescribeDomainChangeProgress
  - ListDomainNames
  - ListTags
  - UpdateDomainConfig

Ações

`AddTags`

O código de exemplo a seguir mostra como usar `AddTags`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously adds tags to an Amazon OpenSearch Service domain.
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();

       return getAsyncClient().addTags(addTagsRequest)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- Para obter detalhes da API, consulte AddTags em Referência AWS SDK for Java 2.x da API.

`ChangeProgress`

O código de exemplo a seguir mostra como usar `ChangeProgress`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
            long startTime = System.currentTimeMillis();
```

```
            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response = getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check domain
progress", ex);
                            }
                            return resp;
                        }).join();

                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }
```

- Para obter detalhes da API, consulte ChangeProgress em Referência AWS SDK for Java 2.x da API.

`CreateDomain`

O código de exemplo a seguir mostra como usar `CreateDomain`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
```

```
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }
```

- Para obter detalhes da API, consulte CreateDomain em Referência AWS SDK for Java 2.x da API.

`DeleteDomain`

O código de exemplo a seguir mostra como usar `DeleteDomain`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
```

```
     * @return a {@link CompletableFuture} that completes when the domain has been
deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteDomain em Referência AWS SDK for Java 2.x da API.

`DescribeDomain`

O código de exemplo a seguir mostra como usar `DescribeDomain`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the domain
```

```
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }
```

- Para obter detalhes da API, consulte DescribeDomain em Referência AWS SDK for Java 2.x da API.

`ListDomainNames`

O código de exemplo a seguir mostra como usar `ListDomainNames`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
    }
```

- Para obter detalhes da API, consulte ListDomainNames em Referência AWS SDK for Java 2.x da API.

`ListTags`

O código de exemplo a seguir mostra como usar `ListTags`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
```

```
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();

       return getAsyncClient().addTags(addTagsRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
               } else {
                   logger.info("Added Tags");
               }
           });
   }
```

- Para obter detalhes da API, consulte ListTags em Referência AWS SDK for Java 2.x da API.

UpdateDomainConfig

O código de exemplo a seguir mostra como usar `UpdateDomainConfig`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates the configuration of a specific domain asynchronously.
    * @param domainName the name of the domain to update
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
    */
   public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
       ClusterConfig clusterConfig = ClusterConfig.builder()
           .instanceCount(3)
           .build();

       UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
           .domainName(domainName)
           .clusterConfig(clusterConfig)
           .build();

       return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to update the domain
configuration", exception);
               }
               // Handle success if needed (e.g., logging or additional actions)
           });
   }
```

- Para obter detalhes da API, consulte UpdateDomainConfig em Referência AWS SDK for Java 2.x da API.

EventBridge exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with EventBridge.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá EventBridge

O exemplo de código a seguir mostra como começar a usar o EventBridge.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                    .limit(10)
                    .build();

            ListEventBusesResponse response =
 eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
            for (EventBus bus : buses) {
                System.out.println("The name of the event bus is: " + bus.name());
                System.out.println("The ARN of the event bus is: " + bus.arn());
            }
```

```
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListEventBuses em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma regra e adicionar um destino a ela.
- Habilitar e desabilitar regras.
- Listar e atualizar regras e destinos.
- Enviar eventos e, em seguida, limpar os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
```

```
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
 *
 */
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
 {
        final String usage = """

                Usage:
                    <roleName> <bucketName> <topicName> <eventRuleName>

                Where:
                    roleName - The name of the role to create.
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
 bucket name to create.
                    topicName - The name of the Amazon Simple Notification Service
 (Amazon SNS) topic to create.
                    eventRuleName - The Amazon EventBridge rule name to create.
```

```
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        SnsClient snsClient = SnsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EventBridge example scenario.");
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out
                .println("1. Create an AWS Identity and Access Management (IAM) role
to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
```

```
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
                "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email when
an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to the
S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule " + eventRuleName + " to use a custom
pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
```

```
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
            IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
                .policyArn(policyArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
```

```
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3Client.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

        for (S3Object myValue : objects) {
            toDelete.add(ObjectIdentifier.builder()
                    .key(myValue.key())
                    .build());
        }

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(Delete.builder()
                        .objects(toDelete).build())
                .build();

        s3Client.deleteObjects(dor);

        // Delete the S3 bucket.
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("You have deleted the bucket and the objects");
    }

    // Delete the SNS topic.
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
                .name(ruleName)
                .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Successfully deleted the rule");
    }

    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }

    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
                "\"UserEmail\": \"" + email + "\"," +
                "\"Message\": \"This event was generated by example code.\"," +
```

```
                "\"UtcTime\": \"Now.\"" +
                "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
                .source("ExampleSource")
                .detail(json)
                .detailType("ExampleType")
                .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
       String customEventsPattern = "{" +
               "\"source\": [\"ExampleSource\"]," +
               "\"detail-type\": [\"ExampleType\"]" +
               "}";

       PutRuleRequest request = PutRuleRequest.builder()
               .name(ruleName)
               .description("Custom test rule")
               .eventPattern(customEventsPattern)
               .build();

       eventBrClient.putRule(request);
   }

   // Update an Amazon S3 object created rule with a transform on the target.
   public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
       String targetId = java.util.UUID.randomUUID().toString();
       Map<String, String> myMap = new HashMap<>();
       myMap.put("bucket", "$.detail.bucket.name");
       myMap.put("time", "$.time");

       InputTransformer inputTransformer = InputTransformer.builder()
               .inputTemplate("\"Notification: an object was uploaded to bucket
<bucket> at <time>.\"")
               .inputPathsMap(myMap)
               .build();

       Target target = Target.builder()
               .id(targetId)
               .arn(topicArn)
               .inputTransformer(inputTransformer)
               .build();

       try {
           PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                   .rule(ruleName)
                   .targets(target)
                   .eventBusName(null)
                   .build();

           eventBrClient.putTargets(targetsRequest);
```

```
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        try {
            DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

            DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
            System.out.println("The state of the rule is " +
response.stateAsString());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }

    // Create and upload a file to an S3 bucket to trigger an event.
    public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
        // Create a unique file name.
        String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
        String fileName = "TextFile" + fileSuffix + ".txt";

        File myFile = new File(fileName);
        FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(fileName)
                    .build();

            s3Client.putObject(putOb, RequestBody.fromFile(myFile));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
        ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
                .targetArn(topicArn)
                .build();

        ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
```

```
        }
    }

    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
                + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
```

```
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
 rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
```

```
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
    }

    // Create a new event rule that triggers when an Amazon S3 object is created in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
```

```
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
```

```
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
```

```
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Ações

`DeleteRule`

O código de exemplo a seguir mostra como usar `DeleteRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
       DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
               .name(ruleName)
               .build();

       eventBrClient.deleteRule(ruleRequest);
       System.out.println("Successfully deleted the rule");
   }
```

- Para obter detalhes da API, consulte DeleteRule em Referência AWS SDK for Java 2.x da API.

`DescribeRule`

O código de exemplo a seguir mostra como usar `DescribeRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
       try {
           DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                   .name(eventRuleName)
                   .build();

           DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
           System.out.println("The state of the rule is " +
response.stateAsString());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeRule em Referência AWS SDK for Java 2.x da API.

`DisableRule`

O código de exemplo a seguir mostra como usar `DisableRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Desabilitar uma regra usando o nome da regra.

```
   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
       try {
           if (!isEnabled) {
               System.out.println("Disabling the rule: " + eventRuleName);
               DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
```

```
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DisableRule em Referência AWS SDK for Java 2.x da API.

`EnableRule`

O código de exemplo a seguir mostra como usar `EnableRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Habilitar uma regra usando o nome da regra.

```
    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
```

```
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte EnableRule em Referência AWS SDK for Java 2.x da API.

`ListRuleNamesByTarget`

O código de exemplo a seguir mostra como usar `ListRuleNamesByTarget`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar todos os nomes das regras usando o destino.

```
   public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
       ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
               .targetArn(topicArn)
               .build();
```

```
        ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }
```

- Para obter detalhes da API, consulte ListRuleNamesByTarget em Referência AWS SDK for Java 2.x da API.

`ListRules`

O código de exemplo a seguir mostra como usar `ListRules`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Habilitar uma regra usando o nome da regra.

```
    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }
```

```
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListRules em Referência AWS SDK for Java 2.x da API.

`ListTargetsByRule`

O código de exemplo a seguir mostra como usar `ListTargetsByRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todos os destinos de uma regra usando o nome da regra.

```
    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }
```

- Para obter detalhes da API, consulte ListTargetsByRule em Referência AWS SDK for Java 2.x da API.

PutEvents

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
       String json = "{" +
               "\"UserEmail\": \"" + email + "\"," +
               "\"Message\": \"This event was generated by example code.\"," +
               "\"UtcTime\": \"Now.\"" +
               "}";

       PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
               .source("ExampleSource")
               .detail(json)
               .detailType("ExampleType")
               .build();

       PutEventsRequest eventsRequest = PutEventsRequest.builder()
               .entries(entry)
               .build();

       eventBrClient.putEvents(eventsRequest);
   }
```

- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK for Java 2.x da API.

PutRule

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma regra agendada

```
   public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
       try {
           PutRuleRequest ruleRequest = PutRuleRequest.builder()
                   .name(ruleName)
                   .eventBusName("default")
                   .scheduleExpression(cronExpression)
                   .state("ENABLED")
                   .description("A test rule that runs on a schedule created by the
Java API")
                   .build();

           PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
           System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Crie uma regra que seja acionada quando um objeto é adicionado a um bucket do Amazon Simple Storage Service.

```
   // Create a new event rule that triggers when an Amazon S3 object is created in
   // a bucket.
   public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
           String eventRuleName) {
       String pattern = "{\n" +
```

```
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte PutRule em Referência AWS SDK for Java 2.x da API.

`PutTargets`

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione um tópico do Amazon SNS como um destino para uma regra.

```
   // Add a rule which triggers an SNS target when a file is uploaded to an S3
   // bucket.
   public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
           String topicName, String eventRuleName, String bucketName) {
       String targetID = java.util.UUID.randomUUID().toString();
       Target myTarget = Target.builder()
               .id(targetID)
               .arn(topicArn)
               .build();

       List<Target> targets = new ArrayList<>();
       targets.add(myTarget);
       PutTargetsRequest request = PutTargetsRequest.builder()
               .eventBusName(null)
               .targets(targets)
               .rule(ruleName)
               .build();

       eventBrClient.putTargets(request);
       System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
               + bucketName + ".");
   }
```

Adicione um transformador de entrada a um destino para uma regra.

```
   public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
           String ruleName) {
       String targetId = java.util.UUID.randomUUID().toString();
       InputTransformer inputTransformer = InputTransformer.builder()
               .inputTemplate("\"Notification: sample event was received.\"")
               .build();

       Target target = Target.builder()
               .id(targetId)
               .arn(topicArn)
               .inputTransformer(inputTransformer)
               .build();
```

```
        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte PutTargets em Referência AWS SDK for Java 2.x da API.

`RemoveTargets`

O código de exemplo a seguir mostra como usar `RemoveTargets`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Remover todos os destinos de uma regra usando o nome da regra.

```
    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
```

```
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }
```

- Para obter detalhes da API, consulte RemoveTargets em Referência AWS SDK for Java 2.x da API.

Cenários

Enviar notificações de eventos para EventBridge

O exemplo de código a seguir mostra como habilitar um bucket para enviar notificações de eventos do S3 EventBridge e encaminhá-las para um tópico do Amazon SNS e uma fila do Amazon SQS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
     * to route the S3 object created events to a topic and a queue.
     *
     * @param bucketName Name of existing bucket
     * @param topicArn ARN of existing topic to receive S3 event notifications
     * @param queueArn ARN of existing queue to receive S3 event notifications
```

```
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
                           logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                           throw new RuntimeException(t.getCause().getMessage(),
t);
                       }
                       logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                   }).join();
```

```
            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Java 2.x

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime

de Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

EventBridge Exemplos de agendador usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with EventBridge Scheduler.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá EventBridge Scheduler

O exemplo de código a seguir mostra como começar a usar o EventBridge Scheduler.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ListSchedulesRequest;
import software.amazon.awssdk.services.scheduler.model.ScheduleSummary;
import software.amazon.awssdk.services.scheduler.paginators.ListSchedulesPublisher;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloScheduler {

    public static void main(String [] args) {
        listSchedulesAsync();
    }

    /**
     * Lists all the schedules available.
     * <p>
     * This method uses the {@link SchedulerAsyncClient} to make an asynchronous
 request to
     * list all the schedules available. The method uses the {@link
 ListSchedulesPublisher}
     * to fetch the schedules in a paginated manner, and then processes the
 responses
     * asynchronously.
     */
    public static void listSchedulesAsync() {
        SchedulerAsyncClient schedulerAsyncClient = SchedulerAsyncClient.create();
```

```
        // Build the request to list schedules
        ListSchedulesRequest listSchedulesRequest =
 ListSchedulesRequest.builder().build();

        // Use the paginator to fetch all schedules asynchronously.
        ListSchedulesPublisher paginator =
 schedulerAsyncClient.listSchedulesPaginator(listSchedulesRequest);
        List<ScheduleSummary> results = new ArrayList<>();

        // Subscribe to the paginator to process the response asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schedules().forEach(schedule -> {
                results.add(schedule);
                System.out.printf("Schedule: %s%n", schedule.name());
            });
        });

        // Wait for the asynchronous operation to complete.
        future.join();

        // After all schedules are fetched, print the total count.
        System.out.printf("Total of %d schedule(s) available.%n", results.size());
    }
}
```

- Para obter detalhes da API, consulte ListSchedules em Referência AWS SDK for Java 2.x da API.

Ações

`CreateSchedule`

O código de exemplo a seguir mostra como usar `CreateSchedule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new schedule for a target task.
    *
    * @param name                  the name of the schedule
    * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
    * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
    * @param targetArn             the Amazon Resource Name (ARN) of the target
task
    * @param roleArn               the ARN of the IAM role to be used for the
schedule
    * @param input                 the input data for the target task
    * @param deleteAfterCompletion whether to delete the schedule after it's
executed
    * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
    * @return true if the schedule was successfully created, false otherwise
    */
   public CompletableFuture<Boolean> createScheduleAsync(
       String name,
       String scheduleExpression,
       String scheduleGroupName,
       String targetArn,
       String roleArn,
       String input,
       boolean deleteAfterCompletion,
       boolean useFlexibleTimeWindow) {

       int hoursToRun = 1;
       int flexibleTimeWindowMinutes = 10;

       Target target = Target.builder()
           .arn(targetArn)
           .roleArn(roleArn)
           .input(input)
           .build();

       FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
           .mode(useFlexibleTimeWindow
               ? FlexibleTimeWindowMode.FLEXIBLE
               : FlexibleTimeWindowMode.OFF)
```

```
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateSchedule em Referência AWS SDK for Java 2.x da API.

`CreateScheduleGroup`

O código de exemplo a seguir mostra como usar `CreateScheduleGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new schedule group.
    *
    * @param name the name of the schedule group to be created
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
    */
   public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
       CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
           .name(name)
           .build();

       logger.info("Initiating createScheduleGroup call for group: {}", name);
       CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
       futureResponse.whenComplete((response, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                   // Rethrow the ConflictException
                   throw (ConflictException) ex.getCause();
               } else {
                   throw new CompletionException("Failed to create schedule group:
" + name, ex);
               }
```

```
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }
```

- Para obter detalhes da API, consulte CreateScheduleGroup em Referência AWS SDK for Java 2.x da API.

`DeleteSchedule`

O código de exemplo a seguir mostra como usar `DeleteSchedule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
```

```
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while deleting
schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.", ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
```

- Para obter detalhes da API, consulte DeleteSchedule em Referência AWS SDK for Java 2.x da API.

`DeleteScheduleGroup`

O código de exemplo a seguir mostra como usar `DeleteScheduleGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes the specified schedule group.
     *
```

```
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteScheduleGroup em Referência AWS SDK for Java 2.x da API.

Cenários

Eventos agendados

O exemplo de código a seguir mostra como:

- Implante uma CloudFormation pilha com os recursos necessários.
- Crie um grupo de EventBridge agendamento do Scheduler.
- Crie uma programação única do EventBridge Scheduler com uma janela de tempo flexível.

- Crie uma programação recorrente do EventBridge Scheduler com uma taxa especificada.
- Exclua o EventBridge Scheduler, a programação e o grupo de agendamentos.
- Limpe os recursos e exclua a pilha.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.scheduler.model.SchedulerException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * This Java code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:
 * <p>
 * 1. Prepare the Application:
 * - Prompt the user for an email address to use for the subscription for the SNS
 topic subscription.
 * - Deploy the Cloud Formation template in resources/cfn_template.yaml for resource
 creation.
 * - Store the outputs of the stack into variables for use in the workflow.
 * - Create a schedule group for all workflow schedules.
 * <p>
 * 2. Create one-time Schedule:
 * - Create a one-time schedule to send an initial event.
 * - Use a Flexible Time Window and set the schedule to delete after completion.
```

```
 * - Wait for the user to receive the event email from SNS.
 * <p>
 * 3. Create a time-based schedule:
 * - Prompt the user for how many X times per Y hours a recurring event should be
 scheduled.
 * - Create the scheduled event for X times per hour for Y hours.
 * - Wait for the user to receive the event email from SNS.
 * - Delete the schedule when the user is finished.
 * <p>
 * 4. Clean up:
 * - Prompt the user for y/n answer if they want to destroy the stack and clean up
 all resources.
 * - Delete the schedule group.
 * - Destroy the Cloud Formation stack and wait until the stack has been removed.
 */

public class EventbridgeSchedulerScenario {

    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerScenario.class);
    private static final Scanner scanner = new Scanner(System.in);
    private static String STACK_NAME = "workflow-stack-name";
    private static final String scheduleGroupName = "schedules-group";

    private static String recurringScheduleName = "";

    private static String oneTimeScheduleName = "";

    private static final EventbridgeSchedulerActions eventbridgeActions = new
 EventbridgeSchedulerActions();

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static String roleArn = "";
    public static String snsTopicArn = "";

    public static void main(String[] args) {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon EventBridge Scheduler Workflow.");
        logger.info("""
            Amazon EventBridge Scheduler is a fully managed service that helps you
 schedule and execute
            a wide range of tasks and events in the cloud. It's designed to simplify
 the process of
```

```
            scheduling and managing recurring or one-time events, making it easier
for developers and
            businesses to automate various workflows and processes.

            One of the key features of Amazon EventBridge Scheduler is its ability
to schedule events
            based on a variety of triggers, including time-based schedules, custom
event patterns, or
            even integration with other AWS services. For example, you can use
EventBridge Scheduler
            to schedule a report generation task to run every weekday at 9 AM, or to
trigger a
            Lambda function when a specific Amazon S3 object is created.

            This flexibility allows you to build complex and dynamic event-driven
architectures
            that adapt to your business needs.

            Lets get started...
            """);
        waitForInputToContinue();
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Prepare the application.");
        waitForInputToContinue();
        try {
            boolean prepareSuccess = prepareApplication();
            logger.info(DASHES);

            if (prepareSuccess) {
                logger.info("2. Create one-time schedule.");
                logger.info("""
                    A one-time schedule in Amazon EventBridge Scheduler is an event
trigger that allows
                    you to schedule a one-time event to run at a specific date and
time. This is useful for
                    executing a specific task or workflow at a predetermined time,
without the need for recurring
                    or complex scheduling.
                    """);
                waitForInputToContinue();
                createOneTimeSchedule();
```

```
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans = scanner.nextLine().trim();
                if (ans.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(oneTimeScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);

                logger.info("3. Create a recurring schedule.");
                logger.info("""
                    A recurring schedule is a feature that allows you to schedule
and manage the execution
                    of your serverless applications or workloads on a recurring
basis. For example,
                    with EventBridge Scheduler, you can create custom schedules for
your AWS Lambda functions,
                    AWS Step Functions, and other supported event sources, enabling
you to automate tasks and
                    workflows without the need for complex infrastructure
management.
                    """);
                waitForInputToContinue();
                createRecurringSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(recurringScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);
            }
        } catch (Exception ex) {
            logger.info("There was a problem with the workflow {}, initiating
cleanup...", ex.getMessage());
            cleanUp();
        }

        logger.info(DASHES);
        logger.info("4. Clean up the resources.");
        logger.info("Do you want to delete these AWS resources (y/n) ?");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
```

```
            cleanUp();
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        logger.info("Amazon EventBridge Scheduler workflow completed.");
        logger.info(DASHES);
    }

    /**
     * Cleans up the resources associated with the EventBridge scheduler.
     * If any errors occur during the cleanup process, the corresponding error
messages are logged.
     */
    public static void cleanUp() {
        logger.info("First, delete the schedule group.");
        logger.info("When the schedule group is deleted, schedules that are part of
that group are deleted.");
        waitForInputToContinue();
        try {
            eventbridgeActions.deleteScheduleGroupAsync(scheduleGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof SchedulerException schedulerException) {
                logger.error("Scheduler error occurred: Error message: {}, Error
code {}",
                    schedulerException.getMessage(),
schedulerException.awsErrorDetails().errorCode(), schedulerException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }

        logger.info("Destroy the CloudFormation stack");
        waitForInputToContinue();
        CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
    }

    /**
     * Prepares the application by creating resources in a CloudFormation stack,
including an SNS topic
```

```
    * that will be subscribed to the EventBridge Scheduler events. The user will
need to confirm the subscription
    * in order to receive event emails.
    *
    * @return true if the application preparation was successful, false otherwise
    */
   public static boolean prepareApplication() {
       logger.info("""
           This example creates resources in a CloudFormation stack, including an
SNS topic
           that will be subscribed to the EventBridge Scheduler events.
           You will need to confirm the subscription in order to receive event
emails.
            """);

       String emailAddress = promptUserForEmail();
       logger.info("You entered {}", emailAddress);

       logger.info("Do you want to use a custom Stack name (y/n) ?");
       String ans = scanner.nextLine().trim();
       if (ans.equalsIgnoreCase("y")) {
           String newStackName = scanner.nextLine();
           logger.info("You entered {} for the new stack name", newStackName);
           waitForInputToContinue();
           STACK_NAME = newStackName;
       }

       logger.info("Get the roleArn and snsTopicArn values using a Cloudformation
template.");
       waitForInputToContinue();
       CloudFormationHelper.deployCloudFormationStack(STACK_NAME, emailAddress);
       Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
       roleArn = stackOutputs.get("RoleARN");
       snsTopicArn = stackOutputs.get("SNStopicARN");

       logger.info("The roleARN is {}", roleArn);
       logger.info("The snsTopicArn is {}", snsTopicArn);

       try {
           eventbridgeActions.createScheduleGroup(scheduleGroupName).join();
           logger.info("createScheduleGroupAsync completed successfully.");

       } catch (RuntimeException e) {
```

```
            logger.error("Error occurred: {} ", e.getMessage());
            return false;
        }
        logger.info("Application preparation complete.");
        return true;
    }

    /**
     * Waits for the user to enter 'c' followed by <ENTER> to continue the program.
     * This method is used to pause the program execution and wait for user input
before
     * proceeding.
     */
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    /**
     * Prompts the user to enter an email address and validates the input.
     * If the provided email address is invalid, the method will prompt the user to
try again.
     *
     * @return the valid email address entered by the user
     */
    private static String promptUserForEmail() {
        logger.info("Enter an email address to use for event subscriptions: ");
        String email = scanner.nextLine();
        if (!isValidEmail(email)) {
            logger.info("Invalid email address. Please try again.");
            return promptUserForEmail();
        }
```

```
        return email;
    }

    /**
     * Checks if the given email address is valid.
     *
     * @param email the email address to be validated
     * @return {@code true} if the email address is valid, {@code false} otherwise
     */
    private static boolean isValidEmail(String email) {
        try {
            InternetAddress emailAddress = new InternetAddress(email);
            emailAddress.validate();
            return true;

        } catch (AddressException e) {
            return false;
        }
    }

    /**
     * Creates a one-time schedule to send an initial event in 1 minute with a
 flexible time window.
     *
     * @return {@code true} if the schedule was created successfully, {@code false}
 otherwise
     */
    public static Boolean createOneTimeSchedule() {
        oneTimeScheduleName = promptUserForResourceName("Enter a name for the one-
time schedule:");
        logger.info("Creating a one-time schedule named {} to send an initial event
 in 1 minute with a flexible time window...", oneTimeScheduleName);
        LocalDateTime scheduledTime = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd'T'HH:mm:ss");

        String scheduleExpression = "at(" + scheduledTime.format(formatter) + ")";
        return eventbridgeActions.createScheduleAsync(
            oneTimeScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "One time scheduled event test from schedule",
```

```
            true,
            true).join();
    }


    /**
     * Creates a recurring schedule to send events based on a specific time.
     *
     * @return A {@link CompletableFuture} that completes with a boolean value
indicating the success or failure of the operation.
     */
    public static Boolean createRecurringSchedule() {
        logger.info("Creating a recurring schedule to send events for one hour...");
        recurringScheduleName = promptUserForResourceName("Enter a name for the
recurring schedule:");

        // Prompt the user for the schedule rate (in minutes).
        int scheduleRateInMinutes = promptUserForInteger("Enter the desired schedule
rate (in minutes): ");
        String scheduleExpression = "rate(" + scheduleRateInMinutes + " minutes)";
        return eventbridgeActions.createScheduleAsync(
            recurringScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "Recurrent event test from schedule " + recurringScheduleName,
            true,
            true).join();
    }

    /**
     * Prompts the user for a resource name and validates the input.
     *
     * @param prompt the message to display to the user when prompting for the
resource name
     * @return the valid resource name entered by the user
     */
    private static String promptUserForResourceName(String prompt) {
        logger.info(prompt);
        String resourceName = scanner.nextLine();
        String regex = "[0-9a-zA-Z-_.]+";
        if (!resourceName.matches(regex)) {
```

```
            logger.info("Invalid resource name. Please use a name that matches the
 pattern " + regex + ".");
            return promptUserForResourceName(prompt);
        }
        return resourceName;
    }

    /**
     * Prompts the user for an integer input and returns the integer value.
     *
     * @param prompt the message to be displayed to the user when prompting for
 input
     * @return the integer value entered by the user
     */
    private static int promptUserForInteger(String prompt) {
        logger.info(prompt);
        String stringResponse = scanner.nextLine();
        if (stringResponse == null || stringResponse.trim().isEmpty() || !
isInteger(stringResponse)) {
            logger.info("Invalid integer.");
            return promptUserForInteger(prompt);
        }
        return Integer.parseInt(stringResponse);
    }

    /**
     * Checks if the given string represents a valid integer.
     *
     * @param str the string to be checked
     * @return {@code true} if the string represents a valid integer, {@code false}
 otherwise
     */
    private static boolean isInteger(String str) {
        try {
            Integer.parseInt(str);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }
}
```

Wrapper para operações de serviço.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ActionAfterCompletion;
import software.amazon.awssdk.services.scheduler.model.ConflictException;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupResponse;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleResponse;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindow;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindowMode;
import software.amazon.awssdk.services.scheduler.model.ResourceNotFoundException;
import software.amazon.awssdk.services.scheduler.model.Target;

import java.time.Instant;
import java.util.concurrent.CompletableFuture;
import java.time.Duration;
import java.util.concurrent.CompletionException;

public class EventbridgeSchedulerActions {

    private static SchedulerAsyncClient schedulerClient;
    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerActions.class);

    public static SchedulerAsyncClient getAsyncClient() {
        if (schedulerClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
```

```
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           schedulerClient = SchedulerAsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return schedulerClient;
   }


   /**
    * Creates a new schedule group.
    *
    * @param name the name of the schedule group to be created
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
    */
   public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
       CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
           .name(name)
           .build();
```

```
        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
                } else {
                    throw new CompletionException("Failed to create schedule group:
" + name, ex);
                }
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }


    /**
     * Creates a new schedule for a target task.
     *
     * @param name                  the name of the schedule
     * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
     * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
     * @param targetArn             the Amazon Resource Name (ARN) of the target
task
     * @param roleArn               the ARN of the IAM role to be used for the
schedule
     * @param input                 the input data for the target task
     * @param deleteAfterCompletion whether to delete the schedule after it's
executed
     * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
     * @return true if the schedule was successfully created, false otherwise
```

```
     */
    public CompletableFuture<Boolean> createScheduleAsync(
        String name,
        String scheduleExpression,
        String scheduleGroupName,
        String targetArn,
        String roleArn,
        String input,
        boolean deleteAfterCompletion,
        boolean useFlexibleTimeWindow) {

        int hoursToRun = 1;
        int flexibleTimeWindowMinutes = 10;

        Target target = Target.builder()
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();
```

```
        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
```

```
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while deleting
schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.", ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
```

```
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Exemplos do Forecast usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Forecast.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateDataset`

O código de exemplo a seguir mostra como usar `CreateDataset`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name>\s

                Where:
                    name - The name of the data set.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
        forecast.close();
```

```
    }

    public static String createForecastDataSet(ForecastClient forecast, String name)
{
        try {
            Schema schema = Schema.builder()
                    .attributes(getSchema())
                    .build();

            CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
                    .datasetName(name)
                    .domain("CUSTOM")
                    .datasetType("RELATED_TIME_SERIES")
                    .dataFrequency("D")
                    .schema(schema)
                    .build();

            CreateDatasetResponse response = forecast.createDataset(datasetRequest);
            return response.datasetArn();

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }

    // Create a SchemaAttribute list required to create a data set.
    private static List<SchemaAttribute> getSchema() {

        List<SchemaAttribute> schemaList = new ArrayList<>();
        SchemaAttribute att1 = SchemaAttribute.builder()
                .attributeName("item_id")
                .attributeType("string")
                .build();

        SchemaAttribute att2 = SchemaAttribute.builder()
                .attributeName("timestamp")
                .attributeType("timestamp")
                .build();

        SchemaAttribute att3 = SchemaAttribute.builder()
                .attributeName("target_value")
```

```
                .attributeType("float")
                .build();

        // Push the SchemaAttribute objects to the List.
        schemaList.add(att1);
        schemaList.add(att2);
        schemaList.add(att3);
        return schemaList;
    }
}
```

- Para obter detalhes da API, consulte CreateDataset em Referência AWS SDK for Java 2.x da API.

`CreateForecast`

O código de exemplo a seguir mostra como usar `CreateForecast`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class CreateForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name> <predictorArn>\s

                Where:
                    name - The name of the forecast.\s
                    predictorArn - The arn of the predictor to use.\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }

    public static String createNewForecast(ForecastClient forecast, String name,
 String predictorArn) {
        try {
            CreateForecastRequest forecastRequest = CreateForecastRequest.builder()
                    .forecastName(name)
                    .predictorArn(predictorArn)
                    .build();

            CreateForecastResponse response =
 forecast.createForecast(forecastRequest);
            return response.forecastArn();

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateForecast em Referência AWS SDK for Java 2.x da API.

DeleteDataset

O código de exemplo a seguir mostra como usar `DeleteDataset`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <datasetARN>\s

                Where:
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteDataset em Referência AWS SDK for Java 2.x da API.

`DeleteForecast`

O código de exemplo a seguir mostra como usar `DeleteForecast`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <datasetARN>\s

                Where:
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
```

```
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteForecast em Referência AWS SDK for Java 2.x da API.

`DescribeForecast`

O código de exemplo a seguir mostra como usar `DescribeForecast`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <forecastarn>\s

                Where:
                    forecastarn - The arn of the forecast (for example,
 "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String forecastarn = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        describe(forecast, forecastarn);
        forecast.close();
    }

    public static void describe(ForecastClient forecast, String forecastarn) {
        try {
```

```
            DescribeForecastRequest request = DescribeForecastRequest.builder()
                    .forecastArn(forecastarn)
                    .build();

            DescribeForecastResponse response = forecast.describeForecast(request);
            System.out.println("The name of the forecast is " +
 response.forecastName());

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeForecast em Referência AWS SDK for Java 2.x da API.

`ListDatasetGroups`

O código de exemplo a seguir mostra como usar `ListDatasetGroups`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                    .maxResults(10)
                    .build();

            ListDatasetGroupsResponse response = forecast.listDatasetGroups(group);
            List<DatasetGroupSummary> groups = response.datasetGroups();
            for (DatasetGroupSummary myGroup : groups) {
                System.out.println("The Data Set name is " +
 myGroup.datasetGroupName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListDatasetGroups em Referência AWS SDK for Java 2.x da API.

`ListForecasts`

O código de exemplo a seguir mostra como usar `ListForecasts`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListForecasts {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listAllForeCasts(forecast);
        forecast.close();
    }

    public static void listAllForeCasts(ForecastClient forecast) {
        try {
            ListForecastsRequest request = ListForecastsRequest.builder()
```

```
                    .maxResults(10)
                    .build();

            ListForecastsResponse response = forecast.listForecasts(request);
            List<ForecastSummary> forecasts = response.forecasts();
            for (ForecastSummary forecastSummary : forecasts) {
                System.out.println("The name of the forecast is " +
 forecastSummary.forecastName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListForecasts em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Glacier usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Glacier.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateVault`

O código de exemplo a seguir mostra como usar `CreateVault`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateVault {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            CreateVaultResponse createVaultResult =
 glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
 createVaultResult.location());

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateVault em Referência AWS SDK for Java 2.x da API.

`DeleteArchive`

O código de exemplo a seguir mostra como usar `DeleteArchive`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
```

```
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
                   vaultName - The name of the vault that contains the archive to
 delete.
                   accountId - The account ID value.
                   archiveId - The archive ID value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String archiveId = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
        glacier.close();
    }

    public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
 String accountId,
            String archiveId) {
        try {
```

```
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                    .vaultName(vaultName)
                    .accountId(accountId)
                    .archiveId(archiveId)
                    .build();

            glacier.deleteArchive(delArcRequest);
            System.out.println("The archive was deleted.");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteArchive em Referência AWS SDK for Java 2.x da API.

`DeleteVault`

O código de exemplo a seguir mostra como usar `DeleteVault`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteVault {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteVault em Referência AWS SDK for Java 2.x da API.

`InitiateJob`

O código de exemplo a seguir mostra como usar `InitiateJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recupere um inventário do cofre.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName> <accountId> <path>

                Where:
                   vaultName - The name of the vault.
                   accountId - The account ID value.
                   path - The path where the file is written to.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String jobNum = createJob(glacier, vaultName, accountId);
        checkJob(glacier, jobNum, vaultName, accountId, path);
        glacier.close();
    }

    public static String createJob(GlacierClient glacier, String vaultName, String
 accountId) {
        try {
            JobParameters job = JobParameters.builder()
                    .type("inventory-retrieval")
                    .build();

            InitiateJobRequest initJob = InitiateJobRequest.builder()
                    .jobParameters(job)
                    .accountId(accountId)
                    .vaultName(vaultName)
                    .build();
```

```
            InitiateJobResponse response = glacier.initiateJob(initJob);
            System.out.println("The job ID is: " + response.jobId());
            System.out.println("The relative URI path of the job is: " +
response.location());
            return response.jobId();

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);

        }
        return "";
    }

    // Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
    // Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {
        try {
            boolean finished = false;
            String jobStatus;
            int yy = 0;

            while (!finished) {
                DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                        .jobId(jobId)
                        .accountId(account)
                        .vaultName(name)
                        .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

                if (jobStatus.compareTo("Succeeded") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + jobStatus);
                    Thread.sleep(1000);
                }
                yy++;
            }

            System.out.println("Job has Succeeded");
            GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
```

```
                    .jobId(jobId)
                    .vaultName(name)
                    .accountId(account)
                    .build();

            ResponseBytes<GetJobOutputResponse> objectBytes =
 glacier.getJobOutputAsBytes(jobOutputRequest);
            // Write the data to a local file.
            byte[] data = objectBytes.asByteArray();
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from a Glacier vault");
            os.close();

        } catch (GlacierException | InterruptedException | IOException e) {
            System.out.println(e.getMessage());
            System.exit(1);

        }
    }
}
```

- Para obter detalhes da API, consulte InitiateJob em Referência AWS SDK for Java 2.x da API.

`ListVaults`

O código de exemplo a seguir mostra como usar `ListVaults`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
```

```
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v : vaultList) {
```

```
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }

                // Check for further results.
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListVaults em Referência AWS SDK for Java 2.x da API.

`UploadArchive`

O código de exemplo a seguir mostra como usar `UploadArchive`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
```

```
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = """

                Usage:   <strPath> <vaultName>\s

                Where:
                   strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
                   vaultName - The name of the vault.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
            return res.archiveId();

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }
```

```
    public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
```

```
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }
```

```
            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

- Para obter detalhes da API, consulte UploadArchive em Referência AWS SDK for Java 2.x da API.

AWS Glue exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
```

```
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri> <bucketNameSc>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3 permissions.
\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, s3://<bucket name>/read).
                cron - A cron expression used to specify the schedule  (i.e.,
 cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
```

```
                jobName - The name you assign to this job definition.
                scriptLocation - The Amazon S3 path to a script that runs a job.
                locationUri - The location of the database (you can find this file
in resources folder).
                bucketNameSc - The Amazon S3 bucket name used when creating a job
                """;

        if (args.length != 9) {
            System.out.println(usage);
            return;
        }
        Scanner scanner = new Scanner(System.in);
        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println("""
            AWS Glue is a fully managed extract, transform, and load (ETL) service
provided by Amazon
            Web Services (AWS). It is designed to simplify the process of building,
running, and maintaining
            ETL pipelines, which are essential for data integration and data
warehousing tasks.

            One of the key features of AWS Glue is its ability to automatically
discover and catalog data
            stored in various sources, such as Amazon S3, Amazon RDS, Amazon
Redshift, and other databases.
            This cataloging process creates a central metadata repository, known as
the AWS Glue Data Catalog,
            which provides a unified view of an organization's data assets. This
metadata can then be used to
```

```
            create ETL jobs, which can be scheduled and run on-demand or on a
regular basis.

            Lets get started.

            """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        try {
            createDatabase(glueClient, dbName, locationUri);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().equals("Database already
exists.")) {
                System.out.println("Database " + dbName + " already exists. Skipping
creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                return;
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        try {
            createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().contains("already exists")) {
                System.out.println("Crawler " + crawlerName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("3. Get a crawler.");
        try {
            getSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        try {
            startSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        try {
            getSpecificDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName;
        try {
            myTableName = getGlueTables(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        try {
            createJob(glueClient, jobName, iam, scriptLocation);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        try {
            startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        try {
            getAllJobs(glueClient);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        try {
            getJobRuns(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        try {
            deleteJob(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        try {
            deleteDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        try {
            deleteSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }


    /**
```

```
     * Creates a Glue database with the specified name and location URI.
     *
     * @param glueClient  The Glue client to use for the database creation.
     * @param dbName      The name of the database to create.
     * @param locationUri The location URI for the database.
     */
    public static void createDatabase(GlueClient glueClient, String dbName, String
 locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
                .locationUri(locationUri)
                .build();

            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                .databaseInput(input)
                .build();

            glueClient.createDatabase(request);
            System.out.println(dbName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue crawler using the AWS Glue Java API.
     *
     * @param glueClient  the AWS Glue client used to interact with the AWS Glue
 service
     * @param iam         the IAM role that the crawler will use to access the data
 source
     * @param s3Path      the S3 path that the crawler will scan for data
     * @param cron        the cron expression that defines the crawler's schedule
     * @param dbName      the name of the AWS Glue database where the crawler will
 store the metadata
     * @param crawlerName the name of the crawler to be created
     */
    public static void createGlueCrawler(GlueClient glueClient,
                                         String iam,
                                         String s3Path,
```

```
                                        String cron,
                                        String dbName,
                                        String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
```

```
                .build();

            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }

    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the specific database from the AWS Glue service.
     *
```

```
    * @param glueClient   an instance of the AWS Glue client used to interact with
the service
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
    */
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " + createDate);

       } catch (GlueException e) {
           throw e;
       }
   }


   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table names
from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
```

```
                .databaseName(dbName)
                .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }


    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                                String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
```

```
                .jobName(jobName)
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");
```

```
        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Retrieves and prints information about all the jobs in the Glue data catalog.
     *
     * @param glueClient the Glue client used to interact with the AWS Glue service
     */
    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                .maxResults(10)
                .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();
```

```
            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * Deletes a Glue job.
     *
```

```
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Deletes a specific AWS Glue crawler.
     *
```

```
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
 DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateCrawler

- CreateJob
- DeleteCrawler
- DeleteDatabase
- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new AWS Glue crawler using the AWS Glue Java API.
 *
```

```
     * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
     * @param iam         the IAM role that the crawler will use to access the data
source
     * @param s3Path      the S3 path that the crawler will scan for data
     * @param cron        the cron expression that defines the crawler's schedule
     * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
     * @param crawlerName the name of the crawler to be created
     */
    public static void createGlueCrawler(GlueClient glueClient,
                                         String iam,
                                         String s3Path,
                                         String cron,
                                         String dbName,
                                         String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CreateCrawler em Referência AWS SDK for Java 2.x da API.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new AWS Glue job.
    *
    * @param glueClient     the AWS Glue client to use for the operation
    * @param jobName        the name of the job to create
    * @param iam            the IAM role to associate with the job
    * @param scriptLocation the location of the script to be used by the job
    * @throws GlueException if there is an error creating the job
    */
   public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
       try {
           JobCommand command = JobCommand.builder()
               .pythonVersion("3")
               .name("glueetl")
               .scriptLocation(scriptLocation)
               .build();

           CreateJobRequest jobRequest = CreateJobRequest.builder()
               .description("A Job created by using the AWS SDK for Java V2")
               .glueVersion("2.0")
               .workerType(WorkerType.G_1_X)
               .numberOfWorkers(10)
```

```
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK for Java 2.x da API.

DeleteCrawler

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
                .name(crawlerName)
```

```
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteCrawler em Referência AWS SDK for Java 2.x da API.

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
```

```
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteDatabase em Referência AWS SDK for Java 2.x da API.

`DeleteJob`

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
```

```
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteJob em Referência AWS SDK for Java 2.x da API.

GetCrawler

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }
```

```
            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte GetCrawler em Referência AWS SDK for Java 2.x da API.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact with
the service
     * @param databaseName the name of the database to retrieve
     * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
     */
    public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                .name(databaseName)
                .build();

            GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
```

```
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                .withLocale(Locale.US)
                .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " + createDate);

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte GetDatabase em Referência AWS SDK for Java 2.x da API.

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
```

```
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte GetJobRuns em Referência AWS SDK for Java 2.x da API.

`GetTables`

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table names
from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
               .databaseName(dbName)
               .build();

           GetTablesResponse response = glueClient.getTables(tableRequest);
           List<Table> tables = response.tableList();
           if (tables.isEmpty()) {
               System.out.println("No tables were returned");
           } else {
               for (Table table : tables) {
                   myTableName = table.name();
```

```
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }
```

- Para obter detalhes da API, consulte GetTables em Referência AWS SDK for Java 2.x da API.

StartCrawler

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");
```

```
        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte StartCrawler em Referência AWS SDK for Java 2.x da API.

StartJobRun

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                                String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);
```

```
            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
                .jobName(jobName)
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte StartJobRun em Referência AWS SDK for Java 2.x da API.

HealthImaging exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with HealthImaging.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CopyImageSet`

O código de exemplo a seguir mostra como usar `CopyImageSet`.

SDK para Java 2.x

```
   /**
    * Copy an AWS HealthImaging image set.
    *
    * @param medicalImagingClient  - The AWS HealthImaging client object.
    * @param datastoreId           - The datastore ID.
    * @param imageSetId            - The image set ID.
    * @param latestVersionId       - The version ID.
    * @param destinationImageSetId - The optional destination image set ID, ignored
if null.
    * @param destinationVersionId  - The optional destination version ID, ignored
if null.
    * @param force                 - The force flag.
    * @param subsets               - The optional subsets to copy, ignored if null.
    * @return                      - The image set ID of the copy.
    * @throws MedicalImagingException - Base exception for all service exceptions
thrown by AWS HealthImaging.
    */
   public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
                                            String datastoreId,
                                            String imageSetId,
                                            String latestVersionId,
                                            String destinationImageSetId,
                                            String destinationVersionId,
                                            boolean force,
                                            Vector<String> subsets) {

       try {
           CopySourceImageSetInformation.Builder copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
                   .latestVersionId(latestVersionId);

           // Optionally copy a subset of image instances.
           if (subsets != null) {
```

```
                String subsetInstanceToCopy = getCopiableAttributesJSON(imageSetId,
subsets);
                copySourceImageSetInformation.dicomCopies(MetadataCopies.builder()
                        .copiableAttributes(subsetInstanceToCopy)
                        .build());
            }

            CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                    .sourceImageSet(copySourceImageSetInformation.build());

            // Optionally designate a destination image set.
            if (destinationImageSetId != null) {
                copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                        .imageSetId(destinationImageSetId)
                        .latestVersionId(destinationVersionId)
                        .build());
            }

            CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .force(force)
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Função utilitária para criar atributos copiáveis.

```
    /**
     * Create a JSON string of copiable image instances.
```

```
     *
     * @param imageSetId - The image set ID.
     * @param subsets    - The subsets to copy.
     * @return A JSON string of copiable image instances.
     */
    private static String getCopiableAttributesJSON(String imageSetId,
 Vector<String> subsets) {
        StringBuilder subsetInstanceToCopy = new StringBuilder(
                """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
                                "
                                 """
        );

        subsetInstanceToCopy.append(imageSetId);

        subsetInstanceToCopy.append(
                """
                                ": {
                                "Instances": {
                        """
        );

        for (String subset : subsets) {
            subsetInstanceToCopy.append('"' + subset + "\": {},");
        }
        subsetInstanceToCopy.deleteCharAt(subsetInstanceToCopy.length() - 1);
        subsetInstanceToCopy.append("""
                         }
                       }
                     }
                   }
                 }
                 """);
        return subsetInstanceToCopy.toString();
    }
```

- Para obter detalhes da API, consulte CopyImageSet em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`CreateDatastore`

O código de exemplo a seguir mostra como usar `CreateDatastore`.

SDK para Java 2.x

```
   public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
           String datastoreName) {
       try {
           CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
                   .datastoreName(datastoreName)
                   .build();
           CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
           return response.datastoreId();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return "";
   }
```

- Para obter detalhes da API, consulte CreateDatastore em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteDatastore`

O código de exemplo a seguir mostra como usar `DeleteDatastore`.

SDK para Java 2.x

```
   public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
           String datastoreID) {
       try {
           DeleteDatastoreRequest datastoreRequest =
DeleteDatastoreRequest.builder()
                   .datastoreId(datastoreID)
                   .build();
           medicalImagingClient.deleteDatastore(datastoreRequest);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DeleteDatastore em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteImageSet`

O código de exemplo a seguir mostra como usar `DeleteImageSet`.

SDK para Java 2.x

```
   public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String imagesetId) {
```

```
        try {
            DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId)
                    .build();

            medicalImagingClient.deleteImageSet(deleteImageSetRequest);

            System.out.println("The image set was deleted.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteImageSet em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDICOMImportJob`

O código de exemplo a seguir mostra como usar `GetDICOMImportJob`.

SDK para Java 2.x

```
    public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
            String datastoreId,
            String jobId) {

        try {
            GetDicomImportJobRequest getDicomImportJobRequest =
GetDicomImportJobRequest.builder()
                    .datastoreId(datastoreId)
                    .jobId(jobId)
```

```
                    .build();
            GetDicomImportJobResponse response =
medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
            return response.jobProperties();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Para obter detalhes da API, consulte GetDICOMImportJob em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDatastore`

O código de exemplo a seguir mostra como usar `GetDatastore`.

SDK para Java 2.x

```
    public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
            String datastoreID) {
        try {
            GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
                    .datastoreId(datastoreID)
                    .build();
            GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
            return response.datastoreProperties();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return null;
    }
```

- Para obter detalhes da API, consulte GetDatastore em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageFrame`

O código de exemplo a seguir mostra como usar `GetImageFrame`.

SDK para Java 2.x

```
        public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
                        String destinationPath,
                        String datastoreId,
                        String imagesetId,
                        String imageFrameId) {

                try {
                        GetImageFrameRequest getImageSetMetadataRequest =
GetImageFrameRequest.builder()
                                        .datastoreId(datastoreId)
                                        .imageSetId(imagesetId)

.imageFrameInformation(ImageFrameInformation.builder()
                                                        .imageFrameId(imageFrameId)
                                                        .build())
                                        .build();

medicalImagingClient.getImageFrame(getImageSetMetadataRequest,

FileSystems.getDefault().getPath(destinationPath));
```

```
                        System.out.println("Image frame downloaded to " +
destinationPath);
                } catch (MedicalImagingException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
```

- Para obter detalhes da API, consulte GetImageFrame em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSet`

O código de exemplo a seguir mostra como usar `GetImageSet`.

SDK para Java 2.x

```
   public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
medicalImagingClient,
            String datastoreId,
            String imagesetId,
            String versionId) {
        try {
            GetImageSetRequest.Builder getImageSetRequestBuilder =
GetImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId);

            if (versionId != null) {
                getImageSetRequestBuilder =
getImageSetRequestBuilder.versionId(versionId);
            }

            return
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
```

```
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Para obter detalhes da API, consulte GetImageSet em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSetMetadata`

O código de exemplo a seguir mostra como usar `GetImageSetMetadata`.

SDK para Java 2.x

```
   public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
           String destinationPath,
           String datastoreId,
           String imagesetId,
           String versionId) {

       try {
           GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
GetImageSetMetadataRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId);

           if (versionId != null) {
               getImageSetMetadataRequestBuilder =
getImageSetMetadataRequestBuilder.versionId(versionId);
           }
```

```
medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
                    FileSystems.getDefault().getPath(destinationPath));

            System.out.println("Metadata downloaded to " + destinationPath);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte GetImageSetMetadata em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDICOMImportJobs`

O código de exemplo a seguir mostra como usar `ListDICOMImportJobs`.

SDK para Java 2.x

```
    public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
            String datastoreId) {

        try {
            ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                    .datastoreId(datastoreId)
                    .build();
            ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
            return response.jobSummaries();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }

        return new ArrayList<>();
    }
```

- Para obter detalhes da API, consulte ListDICOMImportJobs em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDatastores`

O código de exemplo a seguir mostra como usar `ListDatastores`.

SDK para Java 2.x

```
   public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
       try {
           ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
                   .build();
           ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
           List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

           responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

           return datastoreSummaries;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Para obter detalhes da API, consulte ListDatastores em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListImageSetVersions`

O código de exemplo a seguir mostra como usar `ListImageSetVersions`.

SDK para Java 2.x

```
   public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
           String datastoreId,
           String imagesetId) {
       try {
           ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId)
                   .build();

           ListImageSetVersionsIterable responses = medicalImagingClient
                   .listImageSetVersionsPaginator(getImageSetRequest);
           List<ImageSetProperties> imageSetProperties = new ArrayList<>();
           responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

           return imageSetProperties;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Para obter detalhes da API, consulte ListImageSetVersions em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListTagsForResource`

O código de exemplo a seguir mostra como usar `ListTagsForResource`.

SDK para Java 2.x

```
   public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
           String resourceArn) {
       try {
           ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .build();

           return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Para obter detalhes da API, consulte ListTagsForResource em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`SearchImageSets`

O código de exemplo a seguir mostra como usar `SearchImageSets`.

SDK para Java 2.x

A função de utilitário para pesquisar conjuntos de imagens.

```
   public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
           MedicalImagingClient medicalImagingClient,
           String datastoreId, SearchCriteria searchCriteria) {
       try {
           SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
                   .datastoreId(datastoreId)
                   .searchCriteria(searchCriteria)
                   .build();
           SearchImageSetsIterable responses = medicalImagingClient
                   .searchImageSetsPaginator(datastoreRequest);
           List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();

           responses.stream().forEach(response -> imageSetsMetadataSummaries
                   .addAll(response.imageSetsMetadataSummaries()));

           return imageSetsMetadataSummaries;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

Caso de uso nº 1: operador EQUAL.

```
        List<SearchFilter> searchFilters =
Collections.singletonList(SearchFilter.builder()
                .operator(Operator.EQUAL)
                .values(SearchByAttributeValue.builder()
                        .dicomPatientId(patientId)
                        .build())
                .build());

        SearchCriteria searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
searchMedicalImagingImageSets(
                medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets for patient " + patientId + " are:\n"
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Caso de uso #2: operador BETWEEN usando DICOMStudyDate DICOMStudyTime e.

```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate("19990101")
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build(),
                        SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate((LocalDate.now()
                                                .format(formatter)))
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build())
```

```
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println(
                    "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
                            +
                            imageSetsMetadataSummaries);
            System.out.println();
        }
```

Caso de uso nº 3: operador BETWEEN usando o createdAt. Os estudos de tempo foram previamente persistidos.

```
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()
                                .createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                                .build(),
                        SearchByAttributeValue.builder()
                                .createdAt(Instant.now())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();
        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with BETWEEN operator using
createdAt are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
```

```
        }
```

Caso de uso #4: operador EQUAL on DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifique a resposta na ordem ASC no campo updatedAt.

```
        Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
        Instant endDate = Instant.now();

        searchFilters = Arrays.asList(
                SearchFilter.builder()
                        .operator(Operator.EQUAL)
                        .values(SearchByAttributeValue.builder()
                                .dicomSeriesInstanceUID(seriesInstanceUID)
                                .build())
                        .build(),
                SearchFilter.builder()
                        .operator(Operator.BETWEEN)
                        .values(

SearchByAttributeValue.builder().updatedAt(startDate).build(),

SearchByAttributeValue.builder().updatedAt(endDate).build()
                        ).build());

        Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .sort(sort)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
                    "in ASC order on updatedAt field are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

- Para obter detalhes da API, consulte SearchImageSets em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

StartDICOMImportJob

O código de exemplo a seguir mostra como usar `StartDICOMImportJob`.

SDK para Java 2.x

```
   public static String startDicomImportJob(MedicalImagingClient
medicalImagingClient,
           String jobName,
           String datastoreId,
           String dataAccessRoleArn,
           String inputS3Uri,
           String outputS3Uri) {

       try {
           StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
                   .jobName(jobName)
                   .datastoreId(datastoreId)
                   .dataAccessRoleArn(dataAccessRoleArn)
                   .inputS3Uri(inputS3Uri)
                   .outputS3Uri(outputS3Uri)
                   .build();
           StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
           return response.jobId();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
        return "";
    }
```

- Para obter detalhes da API, consulte StartDICOMImportJob em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para Java 2.x

```
   public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Map<String, String> tags) {
       try {
           TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .tags(tags)
                   .build();

           medicalImagingClient.tagResource(tagResourceRequest);

           System.out.println("Tags have been added to the resource.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte TagResource em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`UntagResource`

O código de exemplo a seguir mostra como usar `UntagResource`.

SDK para Java 2.x

```
   public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Collection<String> tagKeys) {
       try {
           UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .tagKeys(tagKeys)
                   .build();

           medicalImagingClient.untagResource(untagResourceRequest);

           System.out.println("Tags have been removed from the resource.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte UntagResource em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

UpdateImageSetMetadata

O código de exemplo a seguir mostra como usar `UpdateImageSetMetadata`.

SDK para Java 2.x

```
   /**
    * Update the metadata of an AWS HealthImaging image set.
    *
    * @param medicalImagingClient - The AWS HealthImaging client object.
    * @param datastoreId          - The datastore ID.
    * @param imageSetId           - The image set ID.
    * @param versionId            - The version ID.
    * @param metadataUpdates      - A MetadataUpdates object containing the
updates.
    * @param force                - The force flag.
    * @throws MedicalImagingException - Base exception for all service exceptions
thrown by AWS HealthImaging.
    */
   public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imageSetId,
                                                    String versionId,
                                                    MetadataUpdates
metadataUpdates,
                                                    boolean force) {
       try {
           UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                   .builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imageSetId)
                   .latestVersionId(versionId)
                   .updateImageSetMetadataUpdates(metadataUpdates)
                   .force(force)
                   .build();

           UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

           System.out.println("The image set metadata was updated" + response);
       } catch (MedicalImagingException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Caso de uso 1: insira ou atualize um atributo.

```
                final String insertAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .updatableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(insertAttributes
                                                .getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataInsertUpdates, force);
```

Caso de uso 2: remova um atributo.

```
                final String removeAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
```

```
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeAttributes
                                                .getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Caso de uso 3: remova uma instância.

```
                final String removeInstance = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
                              "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1":
{
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                              }
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeInstance
                                                .getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Caso de uso 4: reverta para uma versão anterior.

```
                // In this case, revert to previous version.
                String revertVersionId =
Integer.toString(Integer.parseInt(versionid) - 1);
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .revertToVersionId(revertVersionId)
                        .build();
                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

- Para obter detalhes da API, consulte UpdateImageSetMetadata em Referência AWS SDK for Java 2.x da API.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cenários

Marcar um datastore

O exemplo de código a seguir mostra como marcar um armazenamento HealthImaging de dados.

SDK para Java 2.x

Marcar um datastore.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                ImmutableMap.of("Deployment", "Development"));
```

A função de utilitário para marcar um recurso.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Listar tags para um datastore.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                datastoreArn);
                if (result != null) {
                        System.out.println("Tags for resource: " + result.tags());
                }
```

A função de utilitário para listar as tags de um recurso.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
```

```
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

Desmarcar um datastore.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                Collections.singletonList("Deployment"));
```

A função de utilitário para desmarcar um recurso.

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Marcar um conjunto de imagens

O exemplo de código a seguir mostra como marcar um conjunto de HealthImaging imagens.

SDK para Java 2.x

Marcar um conjunto de imagens

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                ImmutableMap.of("Deployment", "Development"));
```

A função de utilitário para marcar um recurso.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
```

```
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Listar tags para um conjunto de imagens

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                imageSetArn);
                if (result != null) {
                        System.out.println("Tags for resource: " + result.tags());
                }
```

A função de utilitário para listar as tags de um recurso.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
```

```
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

Desmarcar um conjunto de imagens

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                Collections.singletonList("Deployment"));
```

A função de utilitário para desmarcar um recurso.

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exemplos do IAM usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
```

```
        });
    }
}
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações do usuário do IAM.

```
/*
  To run this Java V2 code example, set up your development environment, including
 your credentials.
```

```
  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3 Service
 client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = """

                Usage:
                    <username> <policyName> <roleName> <roleSessionName>
 <bucketName>\s

                Where:
                    username - The name of the IAM user to create.\s
                    policyName - The name of the policy to create.\s
```

```
                    roleName - The name of the role to create.\s
                    roleSessionName - The name of the session required for the
assumeRole operation.\s
                    bucketName - The name of the Amazon S3 bucket from which objects
are read.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        String policyName = args[1];
        String roleName = args[2];
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
```

```
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }
```

```
    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

```
    public static String createIAMRole(IamClient iam, String rolename, String json)
{

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " + response.role().arn());
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {
```

```
        // Use the creds of the new IAM user that was created in this code example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
```

```
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRole(IamClient iam, String roleName, String polArn) {

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                    .policyArn(polArn)
                    .roleName(roleName)
                    .build();

            iam.detachRolePolicy(rolePolicyRequest);

            // Delete the policy.
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(polArn)
                    .build();

            iam.deletePolicy(request);
            System.out.println("*** Successfully deleted " + polArn);

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKey(IamClient iam, String username, String accessKey) {
        try {
```

```
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey

- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AttachRolePolicy {
```

```
   public static void main(String[] args) {
       final String usage = """

               Usage:
                   <roleName> <policyArn>\s

               Where:
                   roleName - A role name that you can obtain from the AWS
Management Console.\s
                   policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
               """;

       if (args.length != 2) {
           System.out.println(usage);
           System.exit(1);
       }

       String roleName = args[0];
       String policyArn = args[1];

       Region region = Region.AWS_GLOBAL;
       IamClient iam = IamClient.builder()
               .region(region)
               .build();

       attachIAMRolePolicy(iam, roleName, policyArn);
       iam.close();
   }

   public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
       try {
           ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                   .roleName(roleName)
                   .build();

           ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
           List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

           // Ensure that the policy is not attached to this role
           String polArn = "";
```

```
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
 this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                    " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte AttachRolePolicy em Referência AWS SDK for Java 2.x da API.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <user>\s

                Where:
                   user - An AWS IAM user that you can obtain from the AWS
 Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
```

```
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateAccessKey em Referência AWS SDK for Java 2.x da API.

`CreateAccountAlias`

O código de exemplo a seguir mostra como usar `CreateAccountAlias`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to create (for example, myawsaccount).
\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
```

```
                    .accountAlias(alias)
                    .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateAccountAlias em Referência AWS SDK for Java 2.x da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static void main(String[] args) {

        final String usage = """
                Usage:
                    CreatePolicy <policyName>\s

                Where:
                    policyName - A unique policy name.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
```

```
                .build();

        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: " +
 result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument)
                    .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);

 waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK for Java 2.x da API.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
*   This example requires a trust policy document. For more information, see:
*   https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
*
*
*  In addition, set up your development environment, including your credentials.
*
*  For information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = """
                Usage:
                    <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that represents
 the trust policy.\s
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
 fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(jsonObject.toJSONString())
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " + response.role().arn());

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static Object readJsonSimpleDemo(String filename) throws Exception {
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
```

```
}
```

- Para obter detalhes da API, consulte CreateRole em Referência AWS SDK for Java 2.x da API.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <username>\s

                Where:
                    username - The name of the user to create.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMUser(iam, username);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created.
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

        } catch (IamException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateUser em Referência AWS SDK for Java 2.x da API.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <username> <accessKey>\s

                Where:
                    username - The name of the user.\s
                    accessKey - The access key ID for the secret access key you want
 to delete.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteAccessKey em Referência AWS SDK for Java 2.x da API.

`DeleteAccountAlias`

O código de exemplo a seguir mostra como usar `DeleteAccountAlias`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte DeleteAccountAlias em Referência AWS SDK for Java 2.x da API.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <policyARN>\s

                Where:
                    policyARN - A policy ARN value to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
```

```
        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(policyARN)
                    .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte DeletePolicy em Referência AWS SDK for Java 2.x da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
```

```
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
```

```
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK for Java 2.x da API.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                    " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte DetachRolePolicy em Referência AWS SDK for Java 2.x da API.

ListAccessKeys

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
```

```
                    userName - The name of the user for which access keys are
retrieved.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void listKeys(IamClient iam, String userName) {
        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

                if (newMarker == null) {
                    ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                            .userName(userName)
                            .build();

                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                            .userName(userName)
                            .marker(newMarker)
                            .build();

                    response = iam.listAccessKeys(request);
                }
```

```
                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
 metadata.accessKeyId());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListAccessKeys em Referência AWS SDK for Java 2.x da API.

`ListAccountAliases`

O código de exemplo a seguir mostra como usar `ListAccountAliases`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListAccountAliases em Referência AWS SDK for Java 2.x da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
```

```
                if (newMarker == null) {
                    ListUsersRequest request = ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
 user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",

 permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK for Java 2.x da API.

UpdateAccessKey

O código de exemplo a seguir mostra como usar `UpdateAccessKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessId> <status>\s

                Where:
                    username - The name of the user whose key you want to update.\s
                    accessId - The access key ID of the secret access key you want
 to update.\s
                    status - The status you want to assign to the secret access key.
\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
        iam.close();
    }

    public static void updateKey(IamClient iam, String username, String accessId,
 String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                    .accessKeyId(accessId)
                    .userName(username)
                    .status(statusType)
                    .build();

            iam.updateAccessKey(request);
            System.out.printf("Successfully updated the status of access key %s to"
 +
                    "status %s for user %s", accessId, status, username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte UpdateAccessKey em Referência AWS SDK for Java 2.x da API.

`UpdateUser`

O código de exemplo a seguir mostra como usar `UpdateUser`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <curName> <newName>\s

                Where:
                    curName - The current user name.\s
                    newName - An updated user name.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String newName)
 {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s", newName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte UpdateUser em Referência AWS SDK for Java 2.x da API.

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
```

```
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
```

```
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
```

```
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
```

```
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }
```

```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
```

```
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns a
static response.
               The service still reports as healthy because health checks are still
shallow.
               """);
       demoChoices(loadBalancer);

       System.out.println("Let's reinstate the recommendation service.");
       paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

       System.out.println("""
               Let's also substitute bad credentials for one of the instances in
the target group so that it can't
               access the DynamoDB recommendation table. We will get an instance id
value.
               """);

       LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
       AutoScaler autoScaler = new AutoScaler();

       // Create a new instance profile based on badCredsProfileName.
       templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
       String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
       System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

       String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
       System.out.println("The association Id value is " + profileAssociationId);
```

```
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
```

```
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
```

```
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();
```

```
                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
```

```
    }
}
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
```

```
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                              // name.
                .build();
```

```
        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
```

```
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
```

```
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
```

```
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
```

```
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {
```

```
        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }
```

```
    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }
```

```
   // Gets data about the instances in the EC2 Auto Scaling group.
   public String getBadInstance(String groupName) {
       DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
               .autoScalingGroupNames(groupName)
               .build();

       DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
       AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
       List<String> instanceIds = autoScalingGroup.instances().stream()
               .map(instance -> instance.instanceId())
               .collect(Collectors.toList());

       String[] instanceIdArray = instanceIds.toArray(new String[0]);
       for (String instanceId : instanceIdArray) {
           System.out.println("Instance ID: " + instanceId);
           return instanceId;
       }
       return "";
   }

   // Gets data about the profile associated with an instance.
   public String getInstanceProfile(String instanceId) {
       Filter filter = Filter.builder()
               .name("instance-id")
               .values(instanceId)
               .build();

       DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
               .builder()
               .filters(filter)
               .build();

       DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
               .describeIamInstanceProfileAssociations(associationsRequest);
       return response.iamInstanceProfileAssociations().get(0).associationId();
   }

   public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```

```
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
```

```
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
```

```
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
```

```
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
```

```
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
```

```
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;
```

```
    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
```

```
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }
```

```
    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
 {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Crie uma classe que envolva as ações do Systems Manager.

```
public class ParameterHelper {
```

```
    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup

- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Trabalhar com a API IAM Policy Builder

O exemplo de código a seguir mostra como:

- Criar políticas do IAM usando a API orientada por objetos.
- Usar a API IAM Policy Builder com o serviço do IAM.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Os exemplos usam as importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Crie uma política com base no tempo.

```
        public String timeBasedPolicyExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addResource(IamResource.ALL)
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_GREATER_THAN)

 .key("aws:CurrentTime")

 .value("2020-04-01T00:00:00Z"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_LESS_THAN)

 .key("aws:CurrentTime")
```

```
 .value("2020-06-30T23:59:59Z")))
                                .build();

               // Use an IamPolicyWriter to write out the JSON string to a more
 readable
               // format.
               return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true)
                                .build());
       }
```

Crie uma política com várias condições.

```
       public String multipleConditionsExample() {
               IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addAction("dynamodb:BatchGetItem")
                                                .addAction("dynamodb:Query")
                                                .addAction("dynamodb:PutItem")
                                                .addAction("dynamodb:UpdateItem")
                                                .addAction("dynamodb:DeleteItem")

 .addAction("dynamodb:BatchWriteItem")

 .addResource("arn:aws:dynamodb:*:*:table/table-name")

 .addConditions(IamConditionOperator.STRING_EQUALS

 .addPrefix("ForAllValues:"),

 "dynamodb:Attributes",
                                                                List.of("column-
name1", "column-name2", "column-name3"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS

 .addSuffix("IfExists"))
```

```
 .key("dynamodb:Select")

 .value("SPECIFIC_ATTRIBUTES")))
                                .build();

                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Use entidades principais em uma política.

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.DENY)
                                                .addAction("s3:*")
                                                .addPrincipal(IamPrincipal.ALL)
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket/*")
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.ARN_NOT_EQUALS)

 .key("aws:PrincipalArn")

 .value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Permitir o acesso entre contas ao .

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
```

```
                                                    .addPrincipal(IamPrincipalType.AWS,
 "111122223333")
                                                    .addAction("s3:PutObject")
                                                    .addResource("arn:aws:s3:::amzn-s3-
demo-bucket/*")
                                                    .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                        .key("s3:x-amz-acl")
                                                                        .value("bucket-
owner-full-control")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Crie e carregue uma `IamPolicy`.

```
        public String createAndUploadPolicyExample(IamClient iam, String accountID,
 String policyName) {
                // Build the policy.
                IamPolicy policy = IamPolicy.builder() // 'version' defaults to
 "2012-10-17".
                                .addStatement(IamStatement.builder()
                                                    .effect(IamEffect.ALLOW)
                                                    .addAction("dynamodb:PutItem")
                                                    .addResource("arn:aws:dynamodb:us-
east-1:" + accountID
                                                                        + ":table/
exampleTableName")
                                                    .build())
                                .build();
                // Upload the policy.
                iam.createPolicy(r ->
 r.policyName(policyName).policyDocument(policy.toJson()));
                return
 policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

Baixe e trabalhe com uma `IamPolicy`.

```
        public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
                        String newPolicyName) {

                String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
                GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

                String policyVersion =
getPolicyResponse.policy().defaultVersionId();
                GetPolicyVersionResponse getPolicyVersionResponse = iam
                                .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

                // Create an IamPolicy instance from the JSON string returned from
IAM.
                String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                                StandardCharsets.UTF_8);
                IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

                /*
                 * All IamPolicy components are immutable, so use the copy method
that creates a
                 * new instance that
                 * can be altered in the same method call.
                 *
                 * Add the ability to get an item from DynamoDB as an additional
action.
                 */
                IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

                // Create a new statement that replaces the original statement.
                IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

                // Upload the new policy. IAM now has both policies.
                iam.createPolicy(r -> r.policyName(newPolicyName)
                                .policyDocument(newPolicy.toJson()));
```

```
                return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK for Java 2.x.
- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

Exemplos do Amazon Inspector usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Inspector.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello

O exemplo de código a seguir mostra como começar a usar o .

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloInspector {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloInspector.class);

    public static void main(String[] args) {
        logger.info("Hello Amazon Inspector!");

        try (Inspector2Client inspectorClient = Inspector2Client.builder().build())
 {

            logger.info("Listing member accounts for this Inspector administrator
 account...");
            listMembers(inspectorClient);

            logger.info("The Hello Inspector example completed successfully.");

        } catch (Inspector2Exception e) {
            logger.error("Error: {}", e.getMessage());
            logger.info("Troubleshooting:");
            logger.info("1. Verify AWS credentials are configured");
            logger.info("2. Check IAM permissions for Inspector2");
            logger.info("3. Ensure Inspector2 is enabled in your account");
            logger.info("4. Verify you're using a supported region");
        }
    }
```

```
    /**
     * Lists all member accounts associated with the current Inspector administrator
 account.
     *
     * @param inspectorClient The Inspector2Client used to interact with AWS
 Inspector.
     */
    public static void listMembers(Inspector2Client inspectorClient) {
        try {
            ListMembersRequest request = ListMembersRequest.builder()
                    .maxResults(50) // optional: limit results
                    .build();

            ListMembersResponse response = inspectorClient.listMembers(request);
            List<Member> members = response.members();

            if (members == null || members.isEmpty()) {
                logger.info("No member accounts found for this Inspector
 administrator account.");
                return;
            }

            logger.info("Found {} member account(s):", members.size());
            for (Member member : members) {
                logger.info(" - Account ID: {}, Status: {}",
                        member.accountId(),
                        member.relationshipStatusAsString());
            }

        } catch (Inspector2Exception e) {
            logger.error("Failed to list members: {}",
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ListMembers em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Verifique o status da conta do Inspetor.
- Certifique-se de que o Inspetor esteja ativado.
- Analise as descobertas de segurança.
- Verifique a cobertura do escaneamento.
- Crie um filtro de descobertas.
- Liste os filtros existentes.
- Verifique o uso e os custos.
- Obtenha estatísticas de cobertura.
- Exclua um filtro.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Executar um cenário interativo demonstrando os recursos de .

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class InspectorScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
   private static final Logger logger =
LoggerFactory.getLogger(InspectorScenario.class);
   private static final Scanner scanner = new Scanner(System.in);

   public static void main(String[] args) {
       InspectorActions inspectorActions = new InspectorActions();
       logger.info("Amazon Inspector Basics Scenario");

       logger.info("""
                   Amazon Inspector is a security assessment service provided by
Amazon Web Services (AWS) that helps
                   improve the security and compliance of applications deployed on
AWS. It automatically assesses
                   applications for vulnerabilities or deviations from best
practices. By leveraging Amazon Inspector,
                   users can gain insights into the overall security state of their
application and identify potential
                   security risks.

                   This service operates by conducting both network and host-based
assessments, allowing it to detect a
                   wide range of security issues, including those related to
operating systems, network configurations,
                   and application dependencies.
                   """);

       waitForInputToContinue();

       try {
           runScenario(inspectorActions);

           logger.info("");
           logger.info("Scenario completed successfully!");
           logger.info("");
           logger.info("What you learned:");
           logger.info("  - How to check Inspector account status");
           logger.info("  - How to enable Inspector");
           logger.info("  - How to list and analyze findings");
           logger.info("  - How to check coverage information");
           logger.info("  - How to create and manage filters");
           logger.info("  - How to track usage and costs");
           logger.info("  - How to clean up resources");
           logger.info("");
```

```
        } catch (Exception ex) {
            logger.error("Scenario failed due to unexpected error: {}",
ex.getMessage(), ex);

        } finally {
            scanner.close();
            logger.info("Exiting...");
        }
    }

    /**
     * Runs the Inspector scenario in a step-by-step sequence.
     *
     * All InspectorActions methods are asynchronous and return CompletableFutures.
     * Each step ends with .join(). Any async exception thrown during .join() will
bubble up
     *
     */
    public static void runScenario(InspectorActions actions) {
        String filterArn = null;
        boolean inspectorEnabled = false;

        try {
            // Step 1
            logger.info(DASHES);
            logger.info("Step 1: Checking Inspector account status...");
            String status = actions.getAccountStatusAsync().join();
            logger.info(status);
            waitForInputToContinue();

            // Step 2
            logger.info(DASHES);
            logger.info("Step 2: Enabling Inspector...");
            String message = actions.enableInspectorAsync(null).join();
            logger.info(message);
            inspectorEnabled = true;  // track that Inspector was enabled
            waitForInputToContinue();

            // Step 3
            logger.info(DASHES);
            logger.info("Step 3: Listing LOW severity findings...");

            // Call the service method
```

```
            List<String> allFindings =
actions.listLowSeverityFindingsAsync().join();

            if (!allFindings.isEmpty()) {
                // Only proceed if there are findings
                String lastArn = allFindings.get(allFindings.size() - 1);
                logger.info("Look up details on: {}", lastArn);
                waitForInputToContinue();
                String details = actions.getFindingDetailsAsync(lastArn).join();
                logger.info(details);
            } else {
                logger.info("No LOW severity findings found.");
            }

            waitForInputToContinue();

            // Step 4
            logger.info(DASHES);
            logger.info("Step 4: Listing coverage...");
            String coverage = actions.listCoverageAsync(5).join();
            logger.info(coverage);
            waitForInputToContinue();

            // Step 5
            logger.info(DASHES);
            logger.info("Step 5: Creating filter...");
            String filterName = "suppress-low-" + System.currentTimeMillis();
            filterArn = actions.createLowSeverityFilterAsync(filterName, "Suppress
low severity findings").join();
            logger.info("Created filter: {}", filterArn);
            waitForInputToContinue();

            // Step 6
            logger.info(DASHES);
            logger.info("Step 6: Listing filters...");
            String filters = actions.listFiltersAsync(10).join();
            logger.info(filters);
            waitForInputToContinue();

            // Step 7
            logger.info(DASHES);
            logger.info("Step 7: Usage totals...");
            String usage = actions.listUsageTotalsAsync(null, 10).join();
            logger.info(usage);
```

```
            waitForInputToContinue();

            // Step 8
            logger.info(DASHES);
            logger.info("Step 8: Coverage statistics...");
            String stats = actions.listCoverageStatisticsAsync().join();
            logger.info(stats);
            waitForInputToContinue();

            // Step 9
            logger.info(DASHES);
            logger.info("Step 9: Delete filter?");
            logger.info("Filter ARN: {}", filterArn);
            logger.info("Delete the filter and disable Inspector? (y/n)");

            if (scanner.nextLine().trim().equalsIgnoreCase("y")) {
                actions.deleteFilterAsync(filterArn).join();
                logger.info("Filter deleted.");
                String disableMsg = actions.disableInspectorAsync(null).join();
                logger.info(disableMsg);
                inspectorEnabled = false; // track that Inspector was disabled
            }

            waitForInputToContinue();

        } catch (Exception ex) {
            logger.error("Scenario encountered an error: {}", ex.getMessage(), ex);
            // Rethrow the exception
            throw ex;

        } finally {
            // Cleanup in case of an exception
            if (filterArn != null) {
                try {
                    actions.deleteFilterAsync(filterArn).join();
                    logger.info("Cleanup: Filter deleted.");
                } catch (Exception e) {
                    logger.warn("Failed to delete filter during cleanup: {}",
e.getMessage(), e);
                }
            }

            if (inspectorEnabled) {
                try {
```

```
                    actions.disableInspectorAsync(null).join();
                    logger.info("Cleanup: Inspector disabled.");
                } catch (Exception e) {
                    logger.warn("Failed to disable Inspector during cleanup: {}",
 e.getMessage(), e);
                }
            }
        }
    }

    // Utility Method
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' to continue:");
            String input = scanner.nextLine().trim();
            if (input.equalsIgnoreCase("c")) break;
            logger.info("Invalid input, try again.");
        }
    }
}
```

Uma classe de wrapper para métodos do SDK de .

```
public class InspectorActions {
    private static Inspector2AsyncClient inspectorAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorActions.class);

    private static Inspector2AsyncClient getAsyncClient() {
        if (inspectorAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
```

```
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            inspectorAsyncClient = Inspector2AsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return inspectorAsyncClient;
    }


    /**
     * Enables AWS Inspector for the provided account(s) and default resource types.
     *
     * @param accountIds Optional list of AWS account IDs.
     */
    public CompletableFuture<String> enableInspectorAsync(List<String> accountIds) {

        // The resource types to enable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        EnableRequest.Builder requestBuilder = EnableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        EnableRequest request = requestBuilder.build();
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
```

```
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

```
    /**
     * Retrieves and prints the coverage statistics using a paginator.
     */
    public CompletableFuture<String> listCoverageStatisticsAsync() {
        ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
                .build();

        return getAsyncClient().listCoverageStatistics(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();

                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Validation error listing coverage statistics:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
```

```
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !response.totals().isEmpty()) {
```

```
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." : summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }



    /**
     * Retrieves the account status using the Inspector2Client.
     */
    public CompletableFuture<String> getAccountStatusAsync() {
        BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
                .accountIds(Collections.emptyList())
```

```
                .build();

        return getAsyncClient().batchGetAccountStatus(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof AccessDeniedException) {
                            throw new CompletionException(
                                    "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {
```

```
                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID: ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !=
null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }
```

```
        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }


    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
     * @param description a descriptive string explaining the purpose of the filter
     * @return a CompletableFuture that completes with the ARN of the created filter
     * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
     */
    public CompletableFuture<String> createLowSeverityFilterAsync(
            String filterName,
```

```
            String description) {

        // Define a filter to match LOW severity findings.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        // Create filter criteria.
        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the filter creation request.
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .filterCriteria(filterCriteria)
                .action(FilterAction.SUPPRESS)
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        // Unexpected async error
```

```
                        throw new CompletionException(
                                "Unexpected error creating filter:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }


    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
```

```
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead of
a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
findings", ex);
                });
    }



    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();
```

```
            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
 ").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
 ").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
```

```
                throw new CompletionException(
                        "Validation error listing coverage: " + cause.getMessage(),
cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }


    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter: " +
cause, cause);
                    }
                    return null;
                });
    }
```

```
   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }

                   StringBuilder sb = new StringBuilder();
                   response.findingDetails().forEach(detail -> {
                       sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                               .append("Risk Score:
").append(detail.riskScore()).append("\n");

                       // ExploitObserved timings
                       if (detail.exploitObserved() != null) {
                           sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                   .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                       }

                       // Reference URLs
                       if (detail.hasReferenceUrls()) {
                           sb.append("Reference URLs:\n");
                           detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
```

```
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }
```

```
                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
 ARN: " + findingArn, cause);
                });
    }


    /**
     * Asynchronously disables AWS Inspector for the specified accounts and resource
 types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
 Inspector;
     *                   may be {@code null} or empty to target the current account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to validation
 errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String> accountIds)
 {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }
```

```
        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
```

```
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - BatchGetAccountStatus
  - BatchGetFindingDetails
  - CreateFilter
  - DeleteFilter
  - Habilitar
  - ListCoverage
  - ListCoverageStatistics
  - ListFilters
  - ListFindings
  - ListUsageTotals

Ações

`BatchGetAccountStatus`

O código de exemplo a seguir mostra como usar `BatchGetAccountStatus`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the account status using the Inspector2Client.
    */
   public CompletableFuture<String> getAccountStatusAsync() {
       BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
               .accountIds(Collections.emptyList())
               .build();

       return getAsyncClient().batchGetAccountStatus(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof AccessDeniedException) {
                           throw new CompletionException(
                                   "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                   cause
                           );

                       }

                       if (cause instanceof Inspector2Exception) {
                           Inspector2Exception e = (Inspector2Exception) cause;

                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                   e
                           );
                       }

                       throw new CompletionException(
                               "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                               exception
                       );
                   }
               })
               .thenApply(response -> {

                   StringBuilder sb = new StringBuilder();
```

```
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {

                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID: ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !=
null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }
```

- Para obter detalhes da API, consulte BatchGetAccountStatus em Referência AWS SDK for Java 2.x da API.

BatchGetFindingDetails

O código de exemplo a seguir mostra como usar `BatchGetFindingDetails`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }

                   StringBuilder sb = new StringBuilder();
                   response.findingDetails().forEach(detail -> {
                       sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                               .append("Risk Score:
").append(detail.riskScore()).append("\n");
```

```
                        // ExploitObserved timings
                        if (detail.exploitObserved() != null) {
                            sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                    .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                        }

                        // Reference URLs
                        if (detail.hasReferenceUrls()) {
                            sb.append("Reference URLs:\n");
                            detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
```

```
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
ARN: " + findingArn, cause);
                });
    }
```

- Para obter detalhes da API, consulte BatchGetFindingDetails em Referência AWS SDK for Java 2.x da API.

`CreateFilter`

O código de exemplo a seguir mostra como usar `CreateFilter`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
```

```
    * @param description a descriptive string explaining the purpose of the filter
    * @return a CompletableFuture that completes with the ARN of the created filter
    * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
    */
   public CompletableFuture<String> createLowSeverityFilterAsync(
           String filterName,
           String description) {

       // Define a filter to match LOW severity findings.
       StringFilter severityFilter = StringFilter.builder()
               .value(Severity.LOW.toString())
               .comparison(StringComparison.EQUALS)
               .build();

       // Create filter criteria.
       FilterCriteria filterCriteria = FilterCriteria.builder()
               .severity(Collections.singletonList(severityFilter))
               .build();

       // Build the filter creation request.
       CreateFilterRequest request = CreateFilterRequest.builder()
               .name(filterName)
               .filterCriteria(filterCriteria)
               .action(FilterAction.SUPPRESS)
               .description(description)
               .build();

       return getAsyncClient().createFilter(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                       if (cause instanceof ValidationException ve) {
                           throw new CompletionException(
                                   "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                   ve
                           );
                       }

                       if (cause instanceof Inspector2Exception e) {
                           throw new CompletionException(
```

```
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        // Unexpected async error
                        throw new CompletionException(
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }
```

- Para obter detalhes da API, consulte CreateFilter em Referência AWS SDK for Java 2.x da API.

`DeleteFilter`

O código de exemplo a seguir mostra como usar `DeleteFilter`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
```

```
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter: " +
cause, cause);
                    }
                    return null;
                });
    }
```

- Para obter detalhes da API, consulte DeleteFilter em Referência AWS SDK for Java 2.x da API.

Disable (Desabilitar)

O código de exemplo a seguir mostra como usar `Disable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
     * Asynchronously disables AWS Inspector for the specified accounts and resource
 types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
 Inspector;
     *                   may be {@code null} or empty to target the current account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to validation
 errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String> accountIds)
 {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
```

```
                                    "Inspector may already be disabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- Para obter detalhes da API, consulte Desativar na referência AWS SDK for Java 2.x da API.

Enable (Habilitar)

O código de exemplo a seguir mostra como usar `Enable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Enables AWS Inspector for the provided account(s) and default resource types.
 *
 * @param accountIds Optional list of AWS account IDs.
 */
public CompletableFuture<String> enableInspectorAsync(List<String> accountIds) {

    // The resource types to enable.
    List<ResourceScanType> resourceTypes = List.of(
            ResourceScanType.EC2,
            ResourceScanType.ECR,
            ResourceScanType.LAMBDA,
            ResourceScanType.LAMBDA_CODE
    );

    // Build the request.
    EnableRequest.Builder requestBuilder = EnableRequest.builder()
            .resourceTypes(resourceTypes);

    if (accountIds != null && !accountIds.isEmpty()) {
        requestBuilder.accountIds(accountIds);
    }

    EnableRequest request = requestBuilder.build();
    return getAsyncClient().enable(request)
```

```
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
```

```
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- Para obter detalhes da API, consulte Habilitar na referência AWS SDK for Java 2.x da API.

`ListCoverage`

O código de exemplo a seguir mostra como usar `ListCoverage`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();

            if (coveredResources == null || coveredResources.isEmpty()) {
```

```
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
 ").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
 ").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
```

```
                        "Validation error listing coverage: " + cause.getMessage(),
cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }
```

- Para obter detalhes da API, consulte ListCoverage em Referência AWS SDK for Java 2.x da API.

`ListCoverageStatistics`

O código de exemplo a seguir mostra como usar `ListCoverageStatistics`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves and prints the coverage statistics using a paginator.
     */
    public CompletableFuture<String> listCoverageStatisticsAsync() {
        ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
                .build();

        return getAsyncClient().listCoverageStatistics(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
```

```
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Validation error listing coverage statistics:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
```

```
                });
    }
```

- Para obter detalhes da API, consulte ListCoverageStatistics em Referência AWS SDK for Java 2.x da API.

`ListFilters`

O código de exemplo a seguir mostra como usar `ListFilters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }

        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();
```

```
        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }
```

- Para obter detalhes da API, consulte ListFilters em Referência AWS SDK for Java 2.x da API.

`ListFindings`

O código de exemplo a seguir mostra como usar `ListFindings`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
```

```
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead of
 a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
```

```
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
findings", ex);
                });
    }
```

- Para obter detalhes da API, consulte ListFindings em Referência AWS SDK for Java 2.x da API.

`ListUsageTotals`

O código de exemplo a seguir mostra como usar `ListUsageTotals`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");
```

```
        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." : summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }
```

```
                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }
```

- Para obter detalhes da API, consulte ListUsageTotals em Referência AWS SDK for Java 2.x da API.

AWS IoT exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS IoT.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK for Java 2.x .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie uma AWS IoT coisa.
- Gerar um certificado de dispositivo.
- Atualize AWS IoT qualquer coisa com atributos.
- Exibir um endpoint exclusivo.
- Liste seus AWS IoT certificados.
- Atualize uma AWS IoT sombra.
- Gravar informações do estado.
- Cria uma regra.
- Listar suas regras.
- Pesquisar coisas usando o nome da coisa.
- Exclua qualquer AWS IoT coisa.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS IoT recursos.

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
 with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series of
steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality for
creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
        System.out.println(DASHES);

        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service that
can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
```

```
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
```

```
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("The AWS IoT workflow has successfully completed.");
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de invólucro para métodos AWS IoT SDK.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
```

```
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
```

```
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
```

```
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT certificate.
     * If the request is successful, it prints the certificate details and returns
the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });
```

```
        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
```

```
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to an
IoT Thing.
    * If the request is successful, it prints a confirmation message and additional
information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn) {
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing successfully.");

               // Print additional information about the Thing.
               describeThing(thingName);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " + cause.getMessage());
               } else {
                   System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                       attachResponse.sdkHttpResponse().statusCode());
               }
           }
       });

       future.join();
   }

   /**
```

```
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
     * If the request is successful, it prints a confirmation message.
```

```
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument, StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
```

```
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }

    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);
```

```
        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
```

```
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
```

```
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest = CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
```

```
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
                List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
```

```
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed from
" + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
```

```
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();
```

```
        CompletableFuture<DeleteThingResponse> future =
 getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint

- DescribeThing
- DetachThingPrincipal
- ListCertificates
- ListThings
- SearchIndex
- UpdateIndexingConfiguration
- UpdateThing

Ações

`AttachThingPrincipal`

O código de exemplo a seguir mostra como usar `AttachThingPrincipal`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to an
IoT Thing.
    * If the request is successful, it prints a confirmation message and additional
information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn) {
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
```

```
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte AttachThingPrincipal em Referência AWS SDK for Java 2.x da API.

`CreateKeysAndCertificate`

O código de exemplo a seguir mostra como usar `CreateKeysAndCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an IoT certificate asynchronously.
    *
    * @return The ARN of the created certificate.
    * <p>
    * This method initiates an asynchronous request to create an IoT certificate.
    * If the request is successful, it prints the certificate details and returns
the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String certificatePem = response.certificatePem();
               certificateArn[0] = response.certificateArn();

               // Print the details.
               System.out.println("\nCertificate:");
               System.out.println(certificatePem);
               System.out.println("\nCertificate ARN:");
               System.out.println(certificateArn[0]);

           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + cause.getMessage());
               }
```

```
            }
        });

        future.join();
        return certificateArn[0];
    }
```

- Para obter detalhes da API, consulte CreateKeysAndCertificate em Referência AWS SDK for Java 2.x da API.

`CreateThing`

O código de exemplo a seguir mostra como usar `CreateThing`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
```

```
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte CreateThing em Referência AWS SDK for Java 2.x da API.

`CreateTopicRule`

O código de exemplo a seguir mostra como usar `CreateTopicRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
```

```
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest = CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
```

```
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte CreateTopicRule em Referência AWS SDK for Java 2.x da API.

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
```

```
                System.out.println(certificateArn + " was successfully deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DeleteCertificate em Referência AWS SDK for Java 2.x da API.

`DeleteThing`

O código de exemplo a seguir mostra como usar `DeleteThing`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
```

```
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DeleteThing em Referência AWS SDK for Java 2.x da API.

`DescribeEndpoint`

O código de exemplo a seguir mostra como usar `DescribeEndpoint`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
```

```
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
 IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- Para obter detalhes da API, consulte DescribeEndpoint em Referência AWS SDK for Java 2.x da API.

`DescribeThing`

O código de exemplo a seguir mostra como usar `DescribeThing`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
```

```
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DescribeThing em Referência AWS SDK for Java 2.x da API.

`DetachThingPrincipal`

O código de exemplo a seguir mostra como usar `DetachThingPrincipal`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();
```

```
        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed from
" + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DetachThingPrincipal em Referência AWS SDK for Java 2.x da API.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
```

```
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte ListCertificates em Referência AWS SDK for Java 2.x da API.

`SearchIndex`

O código de exemplo a seguir mostra como usar `SearchIndex`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
```

```
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte SearchIndex em Referência AWS SDK for Java 2.x da API.

AWS IoT data exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS IoT data.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetThingShadow`

O código de exemplo a seguir mostra como usar `GetThingShadow`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte GetThingShadow em Referência AWS SDK for Java 2.x da API.

UpdateThingShadow

O código de exemplo a seguir mostra como usar UpdateThingShadow.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates the shadow of an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
    * If the request is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void updateShadowThing(String thingName) {
       // Create Thing Shadow State Document.
       String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
       SdkBytes data = SdkBytes.fromString(stateDocument, StandardCharsets.UTF_8);
       UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
           .thingName(thingName)
           .payload(data)
           .build();

       CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
       future.whenComplete((updateResponse, ex) -> {
           if (updateResponse != null) {
               System.out.println("Thing Shadow updated successfully.");
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
```

```
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte UpdateThingShadow em Referência AWS SDK for Java 2.x da API.

AWS IoT SiteWise exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS IoT SiteWise.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT SiteWise

O exemplo de código a seguir mostra como começar a usar o AWS IoT SiteWise.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class HelloSitewise {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
         fetchAssetModels();
    }

    /**
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
 IoTSiteWiseAsyncClient.create();
        ListAssetModelsRequest assetModelsRequest = ListAssetModelsRequest.builder()
            .assetModelTypes(AssetModelType.ASSET_MODEL)
            .build();

        // Asynchronous paginator - process paginated results.
        ListAssetModelsPublisher listModelsPaginator =
 siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
        CompletableFuture<Void> future = listModelsPaginator.subscribe(response -> {
            response.assetModelSummaries().forEach(assetSummary ->
                logger.info("Asset Model Name: {} ", assetSummary.name())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- Para obter detalhes da API, consulte ListAssetModels em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um modelo AWS IoT SiteWise de ativos.
- Crie um AWS IoT SiteWise ativo.
- Recuperar os valores de ID da propriedade.
- Envie dados para um AWS IoT SiteWise ativo.
- Recupere o valor da propriedade AWS IoT SiteWise Asset.
- Crie um AWS IoT SiteWise portal.
- Crie um AWS IoT SiteWise gateway.
- Descreva o AWS IoT SiteWise Gateway.
- Exclua os AWS IoT SiteWise ativos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS IoT SiteWise recursos.

```
public class SitewiseScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static final String ROLES_STACK = "RoleSitewise";

    static SitewiseActions sitewiseActions = new SitewiseActions();
```

```
   public static void main(String[] args) throws Throwable {
       Scanner scanner = new Scanner(System.in);
       String assetModelName = "MyAssetModel1";
       String assetName = "MyAsset1" ;
       String gatewayName = "MyGateway1" ;
       String myThing =  "MyThing1" ;

       logger.info("""
           AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
           makes it easy to collect, store, organize, and monitor data from
industrial equipment and processes.
           It is designed to help industrial and manufacturing organizations
collect data from their equipment and
           processes, and use that data to make informed decisions about their
operations.

           One of the key features of AWS IoT SiteWise is its ability to connect to
a wide range of industrial
           equipment and systems, including programmable logic controllers (PLCs),
sensors, and other
           industrial devices. It can collect data from these devices and organize
it into a unified data model,
           making it easier to analyze and gain insights from the data. AWS IoT
SiteWise also provides tools for
           visualizing the data, setting up alarms and alerts, and generating
reports.

           Another key feature of AWS IoT SiteWise is its ability to scale to
handle large volumes of data.
           It can collect and store data from thousands of devices and process
millions of data points per second,
           making it suitable for large-scale industrial operations. Additionally,
AWS IoT SiteWise is designed
           to be secure and compliant, with features like role-based access
controls, data encryption,
           and integration with other AWS services for additional security and
compliance features.

           Let's get started...
           """);

       waitForInputToContinue(scanner);
       logger.info(DASHES);
```

```
        try {
            runScenario(assetModelName, assetName, gatewayName, myThing);
        } catch (RuntimeException e) {
           logger.info(e.getMessage());
        }
    }

    public static void runScenario(String assetModelName, String assetName, String
gatewayName, String myThing) throws Throwable {
        logger.info("Use AWS CloudFormation to create an IAM role that is required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String iamRole = stackOutputs.get("SitewiseRoleArn");
        logger.info("The ARN of the IAM role is {}",iamRole);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an AWS SiteWise Asset Model");
        logger.info("""
             An AWS IoT SiteWise Asset Model is a way to represent the physical
assets, such as equipment,
             processes, and systems, that exist in an industrial environment. This
model provides a structured and
             hierarchical representation of these assets, allowing users to define
the relationships and properties
             of each asset.

             This scenario creates two asset model properties: temperature and
humidity.
            """);
        waitForInputToContinue(scanner);
        String assetModelId = null;
        try {
            CreateAssetModelResponse response =
sitewiseActions.createAssetModelAsync(assetModelName).join();
            assetModelId = response.assetModelId();
            logger.info("Asset Model successfully created. Asset Model ID: {}. ",
assetModelId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                try {
```

```
                    assetModelId =
sitewiseActions.getAssetModelIdAsync(assetModelName).join();
                    logger.info("The Asset Model {} already exists. The id of the
existing model is {}. Moving on...", assetModelName, assetModelId);
                } catch (CompletionException cex) {
                    logger.error("Exception thrown acquiring the asset model id:
{}", cex.getCause().getCause(), cex);
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + cause.getMessage(),
cause);
                return;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("2. Create an AWS IoT SiteWise Asset");
        logger.info("""
             The IoT SiteWise model that we just created defines the structure and
metadata for your physical assets.
             Now we create an asset from the asset model.

            """);
        logger.info("Let's wait 30 seconds for the asset to be ready.");
        countdown(30);
        waitForInputToContinue(scanner);
        String assetId;
        try {
            CreateAssetResponse response =
sitewiseActions.createAssetAsync(assetName, assetModelId).join();
            assetId = response.assetId();
            logger.info("Asset created with ID: {}", assetId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The asset model id was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Retrieve the property ID values");
        logger.info("""
             To send data to an asset, we need to get the property ID values. In
this scenario, we access the
             temperature and humidity property ID values.
            """);
        waitForInputToContinue(scanner);
        Map<String, String>  propertyIds = null;
        try {
            propertyIds = sitewiseActions.getPropertyIds(assetModelId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException) {
                logger.error("IoTSiteWiseException occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        String humPropId =  propertyIds.get("Humidity");
        logger.info("The Humidity property Id is {}", humPropId);
        String tempPropId = propertyIds.get("Temperature");
        logger.info("The Temperature property Id is {}", tempPropId);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Send data to an AWS IoT SiteWise Asset");
        logger.info("""
            By sending data to an IoT SiteWise Asset, you can aggregate data from
            multiple sources, normalize the data into a standard format, and store
it in a
            centralized location. This makes it easier to analyze and gain insights
from the data.
```

```
            In this example, we generate sample temperature and humidity data and
send it to the AWS IoT SiteWise asset.

            """);
        waitForInputToContinue(scanner);
        try {
            sitewiseActions.sendDataToSiteWiseAsync(assetId, tempPropId,
humPropId).join();
            logger.info("Data sent successfully.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The AWS resource was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Retrieve the value of the IoT SiteWise Asset property");
        logger.info("""
            IoT SiteWise is an AWS service that allows you to collect, process, and
analyze industrial data
            from connected equipment and sensors. One of the key benefits of reading
an IoT SiteWise property
            is the ability to gain valuable insights from your industrial data.

            """);
        waitForInputToContinue(scanner);
        try {
            Double assetVal = sitewiseActions.getAssetPropValueAsync(tempPropId,
assetId).join();
            logger.info("The property name is: {}", "Temperature");
            logger.info("The value of this property is: {}", assetVal);

            waitForInputToContinue(scanner);

            assetVal = sitewiseActions.getAssetPropValueAsync(humPropId,
assetId).join();
```

```
            logger.info("The property name is: {}", "Humidity");
            logger.info("The value of this property is: {}", assetVal);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.info("The AWS resource was not found: {}",
cause.getMessage(), cause);
                } else {
                    logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
                }
                return;
            }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Create an IoT SiteWise Gateway");
        logger.info(
            """
                IoT SiteWise Gateway serves as the bridge between industrial
equipment, sensors, and the
                cloud-based IoT SiteWise service. It is responsible for securely
collecting, processing, and
                transmitting data from various industrial assets to the IoT SiteWise
platform,
                enabling real-time monitoring, analysis, and optimization of
industrial operations.

                """);
        waitForInputToContinue(scanner);
        String gatewayId = "";
        try {
            gatewayId = sitewiseActions.createGatewayAsync(gatewayName,
myThing).join();
            logger.info("Gateway creation completed successfully. id is {}",
gatewayId );
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException siteWiseEx) {
                logger.error("IoT SiteWise error occurred: Error message: {}, Error
code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
```

```
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
        logger.info(DASHES);
        logger.info(DASHES);

        logger.info("7. Describe the IoT SiteWise Gateway");
         waitForInputToContinue(scanner);
        try {
            sitewiseActions.describeGatewayAsync(gatewayId)
                .thenAccept(response -> {
                    logger.info("Gateway Name: {}", response.gatewayName());
                    logger.info("Gateway ARN: {}", response.gatewayArn());
                    logger.info("Gateway Platform: {}", response.gatewayPlatform());
                    logger.info("Gateway Creation Date: {}",
response.creationDate());
                }).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error message:
{}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the AWS IoT SiteWise Assets");
        logger.info(
            """
            Before you can delete the Asset Model, you must delete the assets.

            """);
        logger.info("Would you like to delete the IoT SiteWise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
```

```
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the SiteWise assets.");

            try {
                sitewiseActions.deleteGatewayAsync(gatewayId).join();
                logger.info("Gateway {} was deleted successfully.", gatewayId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }

            try {
                sitewiseActions.deleteAssetAsync(assetId).join();
                logger.info("Request to delete asset {} sent successfully",
assetId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            logger.info("Let's wait 1 minute for the asset to be deleted.");
            countdown(60);
            waitForInputToContinue(scanner);
            logger.info("Delete the AWS IoT SiteWise Asset Model");
            try {
                sitewiseActions.deleteAssetModelAsync(assetModelId).join();
                logger.info("Asset model deleted successfully.");
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
```

```
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            waitForInputToContinue(scanner);

        } else {
            logger.info("The resources will not be deleted.");
        }
        logger.info(DASHES);

        logger.info(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        logger.info("This concludes the AWS IoT SiteWise Scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int totalSeconds) throws InterruptedException {
        for (int i = totalSeconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
```

```
            Thread.sleep(1000); // Wait for 1 second
        }
        System.out.println(); // Move to the next line after countdown
        logger.info("Countdown complete!");
    }
}
```

Uma classe de invólucro para métodos AWS IoT SiteWise SDK.

```
public class SitewiseActions {

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseActions.class);

    private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

    private static IoTSiteWiseAsyncClient getAsyncClient() {
        if (ioTSiteWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ioTSiteWiseAsyncClient;
    }


    /**
```

```
    * Creates an asset model.
    *
    * @param name the name of the asset model to create.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
```

```
            .assetModelProperties(temperatureProperty, humidityProperty)
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }


    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
```

```
                }
            });
    }

    /**
     * Sends data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
     * @param humidityPropId the ID of the humidity property.
     * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
     *         calling code can attach callbacks, then handle the result or
exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
        Map<String, Double> sampleData = generateSampleData();
        long timestamp = Instant.now().toEpochMilli();

        TimeInNanos time = TimeInNanos.builder()
            .timeInSeconds(timestamp / 1000)
            .offsetInNanos((int) ((timestamp % 1000) * 1000000))
            .build();

        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
```

```
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Fetches the value of an asset property.
     *
     * @param propId  the ID of the asset property to fetch.
     * @param assetId the ID of the asset to fetch the property value for.
     * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
     *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
```

```
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }

   /**
    * Retrieves the property IDs associated with a specific asset model.
    *
    * @param assetModelId the ID of the asset model that defines the properties.
    * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
    *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
    *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
```

```
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }

    /**
     * Deletes an asset.
     *
     * @param assetId the ID of the asset to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
        DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
            .assetId(assetId)
            .build();

        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
```

```
                }
            });
    }

    /**
     * Deletes an Asset Model with the specified ID.
     *
     * @param assetModelId the ID of the Asset Model to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
        DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
            .assetModelId(assetModelId)
            .build();

        return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
                }
            });
    }

    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
```

```
    *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
       ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
       return getAsyncClient().listAssetModels(listAssetModelsRequest)
               .handle((listAssetModelsResponse, exception) -> {
                   if (exception != null) {
                       logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                       if (assetModelSummary.name().equals(assetModelName)) {
                           return assetModelSummary.id();
                       }
                   }
                   return null;
               });
   }


   /**
    * Creates a new IoT Sitewise gateway.
    *
    * @param gatewayName The name of the gateway to create.
    * @param myThing     The name of the core device thing to associate with the
gateway.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
```

```
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
       GreengrassV2 gg = GreengrassV2.builder()
           .coreDeviceThingName(myThing)
           .build();

       GatewayPlatform platform = GatewayPlatform.builder()
           .greengrassV2(gg)
           .build();

       Map<String, String> tag = new HashMap<>();
       tag.put("Environment", "Production");

       CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
           .gatewayName(gatewayName)
           .gatewayPlatform(platform)
           .tags(tag)
           .build();

       return getAsyncClient().createGateway(createGatewayRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.error("Error creating the gateway.");
                   throw (CompletionException) exception;
               }
               logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
               return response.gatewayId();
           });
   }

   /**
    * Deletes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
```

```
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
               }
           });
   }

   /**
    * Describes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to describe.
    * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
 gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
 method: {}", exception.getCause().getMessage());
                }
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties

- ListAssetModels

Ações

`BatchPutAssetPropertyValue`

O código de exemplo a seguir mostra como usar `BatchPutAssetPropertyValue`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Sends data to the SiteWise service.
    *
    * @param assetId        the ID of the asset to which the data will be sent.
    * @param tempPropertyId the ID of the temperature property.
    * @param humidityPropId the ID of the humidity property.
    * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
    *         calling code can attach callbacks, then handle the result or
exception by calling
    *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();
```

```
        TimeInNanos time = TimeInNanos.builder()
            .timeInSeconds(timestamp / 1000)
            .offsetInNanos((int) ((timestamp % 1000) * 1000000))
            .build();

        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
```

```
    }
```

- Para obter detalhes da API, consulte BatchPutAssetPropertyValue em Referência AWS SDK for Java 2.x da API.

`CreateAsset`

O código de exemplo a seguir mostra como usar `CreateAsset`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an asset with the specified name and asset model Id.
    *
    * @param assetName    the name of the asset to create.
    * @param assetModelId the Id of the asset model to associate with the asset.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
       CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
           .assetModelId(assetModelId)
```

```
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateAsset em Referência AWS SDK for Java 2.x da API.

`CreateAssetModel`

O código de exemplo a seguir mostra como usar `CreateAssetModel`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an asset model.
     *
     * @param name the name of the asset model to create.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
```

```
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
           .build();

       return getAsyncClient().createAssetModel(createAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
               }
```

```
            });
    }
```

- Para obter detalhes da API, consulte CreateAssetModel em Referência AWS SDK for Java 2.x da API.

`CreateGateway`

O código de exemplo a seguir mostra como usar `CreateGateway`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new IoT Sitewise gateway.
     *
     * @param gatewayName The name of the gateway to create.
     * @param myThing     The name of the core device thing to associate with the
gateway.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
```

```
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }
```

- Para obter detalhes da API, consulte CreateGateway em Referência AWS SDK for Java 2.x da API.

`DeleteAsset`

O código de exemplo a seguir mostra como usar `DeleteAsset`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes an asset.
    *
    * @param assetId the ID of the asset to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
       DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
           .assetId(assetId)
           .build();

       return getAsyncClient().deleteAsset(deleteAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred deleting asset with id: {}",
assetId);
               }
           });
   }
```

- Para obter detalhes da API, consulte DeleteAsset em Referência AWS SDK for Java 2.x da API.

`DeleteAssetModel`

O código de exemplo a seguir mostra como usar `DeleteAssetModel`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes an Asset Model with the specified ID.
    *
    * @param assetModelId the ID of the Asset Model to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
```

```
    }
```

- Para obter detalhes da API, consulte DeleteAssetModel em Referência AWS SDK for Java 2.x da API.

`DeleteGateway`

O código de exemplo a seguir mostra como usar `DeleteGateway`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
        DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();
```

```
        return getAsyncClient().deleteGateway(deleteGatewayRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- Para obter detalhes da API, consulte DeleteGateway em Referência AWS SDK for Java 2.x da API.

`DescribeAssetModel`

O código de exemplo a seguir mostra como usar `DescribeAssetModel`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
```

- Para obter detalhes da API, consulte DescribeAssetModel em Referência AWS SDK for Java 2.x da API.

`DescribeGateway`

O código de exemplo a seguir mostra como usar `DescribeGateway`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
```

```
    * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
       DescribeGatewayRequest request = DescribeGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().describeGateway(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
               }
           });
   }
```

- Para obter detalhes da API, consulte DescribeGateway em Referência AWS SDK for Java 2.x da API.

`GetAssetPropertyValue`

O código de exemplo a seguir mostra como usar `GetAssetPropertyValue`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Fetches the value of an asset property.
    *
    * @param propId  the ID of the asset property to fetch.
    * @param assetId the ID of the asset to fetch the property value for.
    * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
```

```
                });
    }
```

- Para obter detalhes da API, consulte GetAssetPropertyValue em Referência AWS SDK for Java 2.x da API.

`ListAssetModels`

O código de exemplo a seguir mostra como usar `ListAssetModels`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
     *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
     *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
```

```
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
                    return null;
                });
    }
```

- Para obter detalhes da API, consulte ListAssetModels em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Keyspaces usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com Amazon Keyspaces.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Keyspaces

O exemplo de código a seguir mostra como começar a usar o Amazon Keyspaces.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
```

```
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesResponse response =
 keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace : keyspaces) {
                System.out.println("The name of the keyspace is " +
 keyspace.keyspaceName());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListKeyspaces em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um keyspace e uma tabela. O esquema da tabela contém dados do filme e tem a recuperação para um ponto no tempo.
- Conectar-se ao keyspace usando uma conexão TLS segura com autenticação SigV4.
- Consultar a tabela. Adicionar, recuperar e atualizar dados do filme.
- Atualizar a tabela. Adicionar uma coluna para rastrear os filmes assistidos.
- Restaurar a tabela ao estado anterior e limpar os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
 * 8. Get all records from the Movie table.
 * 9. Get a specific Movie.
 * 10. Get a UTC timestamp for the current time.
 * 11. Update the table schema to add a ‘watched’ Boolean column.
 * 12. Update an item as watched.
 * 13. Query for items with watched = True.
 * 14. Restore the table back to the previous state using the timestamp.
```

```
 * 15. Check for completion of the restore action.
 * 16. Delete the table.
 * 17. Confirm that both tables are deleted.
 * 18. Delete the keyspace.
 */

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException, IOException
 {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
        String tableNameRestore = "MovieRestore";
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        DriverConfigLoader loader =
 DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
                .withConfigLoader(loader)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Keyspaces example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a keyspace.");
        createKeySpace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        Thread.sleep(5000);
        System.out.println("2. Check for keyspace existence.");
        checkKeyspaceExistence(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state using
the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            String status;
            GetTableResponse response;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            // Keep looping until table cannot be found and a
ResourceNotFoundException is
```

```
            // thrown.
            while (true) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);
                Thread.sleep(500);
            }

        } catch (ResourceNotFoundException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println("The table is deleted");
    }

    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
 String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRestoredTable(KeyspacesClient keyClient, String
 keyspaceName, String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
```

```
                status = response.statusAsString();
                System.out.println("The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session
                .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
                + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
                .setString("k0", titleUpdate)
                .setInt("k1", yearUpdate)
                .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }

    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();

            keyClient.updateTable(tableRequest);
```

```
        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificMovie(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute(
                "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
Family' ALLOW FILTERING ;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Get records from the Movie table.
    public static void getMovieData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Load data into the table.
    public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
        String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {

            // Add 20 movies to the table.
            if (t == 20)
                break;
```

```
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String plot = currentNode.path("info").path("plot").toString();

            // Insert the data into the Amazon Keyspaces table.
            BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
            builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
            PreparedStatement preparedStatement = session.prepare(sqlStatement);
            builder.addStatement(preparedStatement.boundStatementBuilder()
                    .setString("k0", title)
                    .setInt("k1", year)
                    .setString("k2", plot)
                    .build());

            BatchStatement batchStatement = builder.build();
            session.execute(batchStatement);
            t++;
        }

        System.out.println("You have added " + t + " records successfully!");
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           while (!tableStatus) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println(". The table status is " + status);

               if (status.compareTo("ACTIVE") == 0) {
                   tableStatus = true;
               }
               Thread.sleep(500);
           }

           List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
           for (ColumnDefinition def : cols) {
               System.out.println("The column name is " + def.name());
               System.out.println("The column type is " + def.type());
           }

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           // Set the columns.
           ColumnDefinition defTitle = ColumnDefinition.builder()
                   .name("title")
                   .type("text")
                   .build();
```

```
            ColumnDefinition defYear = ColumnDefinition.builder()
                    .name("year")
                    .type("int")
                    .build();

            ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                    .name("release_date")
                    .type("timestamp")
                    .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                    .name("plot")
                    .type("text")
                    .build();

            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();
```

```
            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
```

```
            System.out.println("The " + name + " KeySpace is ready");

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeySpace(KeyspacesClient keyClient, String
 keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
 keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
 response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable

- UpdateTable

Ações

`CreateKeyspace`

O código de exemplo a seguir mostra como usar `CreateKeyspace`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
           System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CreateKeyspace em Referência AWS SDK for Java 2.x da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           // Set the columns.
           ColumnDefinition defTitle = ColumnDefinition.builder()
                   .name("title")
                   .type("text")
                   .build();

           ColumnDefinition defYear = ColumnDefinition.builder()
                   .name("year")
                   .type("int")
                   .build();

           ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                   .name("release_date")
                   .type("timestamp")
                   .build();

           ColumnDefinition defPlot = ColumnDefinition.builder()
                   .name("plot")
                   .type("text")
                   .build();

           List<ColumnDefinition> colList = new ArrayList<>();
           colList.add(defTitle);
           colList.add(defYear);
           colList.add(defReleaseDate);
           colList.add(defPlot);

           // Set the keys.
```

```
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK for Java 2.x da API.

`DeleteKeyspace`

O código de exemplo a seguir mostra como usar `DeleteKeyspace`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           keyClient.deleteKeyspace(deleteKeyspaceRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DeleteKeyspace em Referência AWS SDK for Java 2.x da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
       try {
           DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           keyClient.deleteTable(tableRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK for Java 2.x da API.

`GetKeyspace`

O código de exemplo a seguir mostra como usar `GetKeyspace`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
           String name = response.keyspaceName();
           System.out.println("The " + name + " KeySpace is ready");

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte GetKeyspace em Referência AWS SDK for Java 2.x da API.

`GetTable`

O código de exemplo a seguir mostra como usar `GetTable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
```

```
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte GetTable em Referência AWS SDK for Java 2.x da API.

`ListKeyspaces`

O código de exemplo a seguir mostra como usar `ListKeyspaces`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListKeyspaces em Referência AWS SDK for Java 2.x da API.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();
```

```
            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK for Java 2.x da API.

`RestoreTable`

O código de exemplo a seguir mostra como usar `RestoreTable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();
```

```
            RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte RestoreTable em Referência AWS SDK for Java 2.x da API.

`UpdateTable`

O código de exemplo a seguir mostra como usar `UpdateTable`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();
```

```
            keyClient.updateTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte UpdateTable em Referência AWS SDK for Java 2.x da API.

Exemplos do Kinesis usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Kinesis.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Exemplos sem servidor

Ações

`CreateStream`

O código de exemplo a seguir mostra como usar `CreateStream`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
```

```
                .region(region)
                .build();
        createStream(kinesisClient, streamName);
        System.out.println("Done");
        kinesisClient.close();
    }

    public static void createStream(KinesisClient kinesisClient, String streamName)
 {
        try {
            CreateStreamRequest streamReq = CreateStreamRequest.builder()
                    .streamName(streamName)
                    .shardCount(1)
                    .build();

            kinesisClient.createStream(streamReq);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateStream em Referência AWS SDK for Java 2.x da API.

`DeleteStream`

O código de exemplo a seguir mostra como usar `DeleteStream`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataStream {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }

    public static void deleteStream(KinesisClient kinesisClient, String streamName)
 {
        try {
```

```
            DeleteStreamRequest delStream = DeleteStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            kinesisClient.deleteStream(delStream);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteStream em Referência AWS SDK for Java 2.x da API.

`GetRecords`

O código de exemplo a seguir mostra como usar `GetRecords`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetRecords {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to read from (for
 example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
 streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
                .streamName(streamName)
```

```
                .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
        do {
            streamRes = kinesisClient.describeStream(describeStreamRequest);
            shards.addAll(streamRes.streamDescription().shards());

            if (shards.size() > 0) {
                lastShardId = shards.get(shards.size() - 1).shardId();
            }
        } while (streamRes.streamDescription().hasMoreShards());

        GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
                .streamName(streamName)
                .shardIteratorType("TRIM_HORIZON")
                .shardId(lastShardId)
                .build();

        GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
        shardIterator = shardIteratorResult.shardIterator();

        // Continuously read data records from shard.
        List<Record> records;

        // Create new GetRecordsRequest with existing shardIterator.
        // Set maximum records to return to 1000.
        GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
                .shardIterator(shardIterator)
                .limit(1000)
                .build();

        GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

        // Put result into record list. Result may be empty.
        records = result.records();

        // Print records
        for (Record record : records) {
            SdkBytes byteBuffer = record.data();
            System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte GetRecords em Referência AWS SDK for Java 2.x da API.

PutRecord

O código de exemplo a seguir mostra como usar `PutRecord`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>
```

```
                Where:
                    streamName - The Amazon Kinesis data stream to which records are
written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        // Ensure that the Kinesis Stream is valid.
        validateStream(kinesisClient, streamName);
        setStockData(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void setStockData(KinesisClient kinesisClient, String streamName)
{
        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x = 0; x < index; x++) {
                StockTrade trade = stockTradeGenerator.getRandomTrade();
                sendStockTrade(trade, kinesisClient, streamName);
                Thread.sleep(100);
            }

        } catch (KinesisException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
```

```
    private static void sendStockTrade(StockTrade trade, KinesisClient
 kinesisClient,
            String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON serialization
 by
        // the Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
                .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol
 as the partition key, explained in
                                                        // the Supplemental
 Information section below.
                .streamName(streamName)
                .data(SdkBytes.fromByteArray(bytes))
                .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String
 streamName) {
        try {
            DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            DescribeStreamResponse describeStreamResponse =
 kinesisClient.describeStream(describeStreamRequest);

            if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
 {
```

```
                System.err.println("Stream " + streamName + " is not active. Please
 wait a few moments and try again.");
                System.exit(1);
            }

        } catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
 streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutRecord em Referência AWS SDK for Java 2.x da API.

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }

}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

AWS KMS exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS KMS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS KMS

O exemplo de código a seguir mostra como começar a usar o AWS Key Management Service.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
```

```
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
 is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
```

```
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ListKeys em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma chave do KMS.
- Listar chaves KMS para sua conta e obter detalhes sobre elas.
- Habilitar e desabilitar chaves do KMS.
- Gerar uma chave de dados simétrica que possa ser usada para criptografia do lado do cliente.
- Gere uma chave assimétrica usada para assinar dados digitalmente.
- Marcar chaves com tags.
- Excluir chaves do KMS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário em um prompt de comando.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String accountId = "";

    private static final Logger logger = LoggerFactory.getLogger(KMSScenario.class);

    static KMSActions kmsActions = new KMSActions();

    static Scanner scanner = new Scanner(System.in);

    static String aliasName = "alias/dev-encryption-key";

    public static void main(String[] args) {
        final String usage = """
            Usage: <granteePrincipal>

            Where:
               granteePrincipal - The principal (user, service account, or group) to
 whom the grant or permission is being given.
```

```
            """;

        if (args.length != 1) {
            logger.info(usage);
            return;
        }
        String granteePrincipal = args[0];
        String policyName = "default";

        accountId = kmsActions.getAccountId();
        String keyDesc = "Created by the AWS KMS API";

        logger.info(DASHES);
        logger.info("""
            Welcome to the AWS Key Management SDK Basics scenario.

            This program demonstrates how to interact with AWS Key Management using
the AWS SDK for Java (v2).
            The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
            and manage AWS KMS keys and control their use across a wide range of AWS
services and applications.
            KMS provides a centralized and unified approach to managing encryption
keys, making it easier to meet your
            data protection and regulatory compliance requirements.

            This Basics scenario creates two key types:

            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
```

```
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create a symmetric KMS key\n");
        logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId;
        try {
            CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
            targetKeyId = futureKeyId.join();
            logger.info("A symmetric key was successfully created " + targetKeyId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            2. Enable a KMS key

            By default, when the SDK creates an AWS key, it is enabled. The next bit
of code checks to
            determine if the key is enabled.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled;
```

```
        try {
            CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
            isEnabled = futureIsKeyEnabled.join();
            logger.info("Is the key enabled? {}", isEnabled);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
```

```
        SdkBytes encryptedData;
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
             """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
```

```
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            By default, when the SDK enables automatic rotation of a KMS key,
            KMS rotates the key material of the KMS key one year (approximately 365
days) from the enable date and every year
            thereafter.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
```

```
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            7. Create a grant

            A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
            It also can allow them to view a KMS key (DescribeKey) and create and
manage grants.
            When authorizing access to a KMS key, grants are considered along with
key policies and IAM policies.
            """);

        waitForInputToContinue(scanner);
        String grantId = null;
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
            The revocation of a grant immediately removes the permissions and access
that the grant had provided.
            This means that any principal (user, role, or service) that was granted
access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
```

```
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            Lets decrypt the data that was encrypted in an early step.
            The code uses the same key to decrypt the string that we encrypted
earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptedData = "";
        try {
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
```

```
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
            A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
            access to KMS keys. Every KMS key must have exactly one key policy. The
statements in the key policy
            determine who has permission to use the KMS key and how they can use
it.
            You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
            must have a key policy.

            By default, when you create a key by using the SDK, a policy is created
that
            gives the AWS account that owns the KMS key full access to the KMS key.

            Let's try to replace the automatically created policy with the following
policy.

                "Version":"2012-10-17",
                "Statement": [{
                "Effect": "Allow",
                "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
```

```
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make sure
it exists.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             Signing your data with an AWS key can provide several benefits that
make it an attractive option
```

```
            for your data signing needs. By using an AWS KMS key, you can leverage
the
            security controls and compliance features provided by AWS,
            which can help you meet various regulatory requirements and enhance the
overall security posture
            of your organization.
           """);
       waitForInputToContinue(scanner);
       try {
           CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
           boolean success = future.join();
           if (success) {
               logger.info("Sign and verify data operation succeeded.");
           } else {
               logger.error("Sign and verify data operation failed.");
           }

       } catch (RuntimeException rt) {
           Throwable cause = rt.getCause();
           if (cause instanceof KmsException kmsEx) {
               logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
           } else {
               logger.info("An unexpected error occurred: " + rt.getMessage());
           }
           deleteAliasName(aliasName);
           deleteKey(targetKeyId);
           throw cause;
       }
       waitForInputToContinue(scanner);

       logger.info(DASHES);
       logger.info("14. Tag your symmetric KMS Key\n");
       logger.info("""
           By using tags, you can improve the overall management, security, and
governance of your
           KMS keys, making it easier to organize, track, and control access to
your encrypted data within
           your AWS environment
           """);
       waitForInputToContinue(scanner);
       try {
           CompletableFuture<Void> future = kmsActions.tagKMSKeyAsync(targetKeyId);
           future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }

    // Deletes KMS resources with user input.
    private static void requestDeleteResources(String aliasName, String targetKeyId)
{
        logger.info(DASHES);
        logger.info("15. Schedule the deletion of the KMS key\n");
        logger.info("""
            By default, KMS applies a waiting period of 30 days,
            but you can specify a waiting period of 7-30 days. When this operation
is successful,
            the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
            cryptographic operations. It remains in this state for the duration of
the waiting period.

            Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
            all data that was encrypted under the KMS key is unrecoverable.
            """);
        logger.info("Would you like to delete the Key Management resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS KMS resources.");
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
                future.join();
```

```
            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }
```

```
        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteKey(String targetKeyId) {
        try {
            CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }

        try {
            CompletableFuture<Void> future = kmsActions.deleteKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteAliasName(String aliasName) {
        try {
            CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
 kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Defina uma classe que encapsule as ações do KMS.

```
public class KMSActions {
    private static final Logger logger = LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async client,
 with the following configurations:
     * <ul>
```

```
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
```

```
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}", keyState);
                }
```

```
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and key
ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
```

```
    * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
    */
   public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
       SdkBytes myBytes = SdkBytes.fromUtf8String(text);
       EncryptRequest encryptRequest = EncryptRequest.builder()
           .keyId(keyId)
           .plaintext(myBytes)
           .build();

       CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
       return responseFuture.whenComplete((response, ex) -> {
           if (response != null) {
               String algorithm = response.encryptionAlgorithm().toString();
               logger.info("The string was encrypted with algorithm {}.",
algorithm);
           } else {
               throw new RuntimeException(ex);
           }
       }).thenApply(EncryptResponse::ciphertextBlob);
   }

   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
```

```
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
```

```
            });
    }

    /**
     * Enables key rotation asynchronously for the specified key ID.
     *
     * @param keyId the ID of the key for which to enable key rotation
     * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
     * @throws RuntimeException if there was an error enabling key rotation, either
due to a KMS exception or an unexpected error
     */
    public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
        EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to enable key rotation: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }

    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
```

```
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
    */
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
       List<GrantOperation> grantPermissions = List.of(
           GrantOperation.ENCRYPT,
           GrantOperation.DECRYPT,
           GrantOperation.DESCRIBE_KEY
       );

       CreateGrantRequest grantRequest = CreateGrantRequest.builder()
           .keyId(keyId)
           .name("grant1")
           .granteePrincipal(granteePrincipal)
           .operations(grantPermissions)
           .build();

       CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex == null) {
               logger.info("Grant created successfully with ID: " +
response.grantId());
           } else {
               if (ex instanceof KmsException kmsEx) {
                   throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
               }
           }
       });

       return responseFuture.thenApply(CreateGrantResponse::grantId);
   }
```

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
     * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }

    /**
     * Revokes a grant for the specified AWS KMS key asynchronously.
     *
     * @param keyId   The ID or key ARN of the AWS KMS key.
     * @param grantId The identifier of the grant to be revoked.
     * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
```

```
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
           } else {
               if (exception instanceof KmsException kmsEx) {
                   if (kmsEx.getMessage().contains("Grant does not exist")) {
                       logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                   } else {
                       throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                   }
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
               }
           }
       });

       return responseFuture;
   }


   /**
    * Asynchronously decrypts the given encrypted data using the specified key ID.
    *
    * @param encryptedData The encrypted data to be decrypted.
    * @param keyId The ID of the key to be used for decryption.
```

```
    * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
    *         If an error occurs during the decryption process, the
CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
String.
    */
   public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
       DecryptRequest decryptRequest = DecryptRequest.builder()
           .ciphertextBlob(encryptedData)
           .keyId(keyId)
           .build();

       CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
       responseFuture.whenComplete((decryptResponse, exception) -> {
           if (exception == null) {
               logger.info("Data decrypted successfully for key ID: " + keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
               }
           }
       });

       return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
   }

   /**
    * Asynchronously replaces the policy for the specified KMS key.
    *
    * @param keyId       the ID of the KMS key to replace the policy for
    * @param policyName  the name of the policy to be replaced
    * @param accountId   the AWS account ID to be used in the policy
    * @return a {@link CompletableFuture} that completes with a boolean indicating
    *         whether the policy replacement was successful or not
    */
```

```
   public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
       String policy = """
   {
     "Version":"2012-10-17",
     "Statement": [{
       "Effect": "Allow",
       "Principal": {"AWS": "arn:aws:iam::%s:root"},
       "Action": "kms:*",
       "Resource": "*"
     }]
   }
   """.formatted(accountId);

       PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
           .keyId(keyId)
           .policyName(policyName)
           .policy(policy)
           .build();

       // First, get the current policy to check if it exists
       return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
           .thenCompose(response -> {
               logger.info("Current policy exists. Replacing it...");
               return getAsyncClient().putKeyPolicy(keyPolicyRequest);
           })
           .thenApply(putPolicyResponse -> {
               logger.info("The key policy has been replaced.");
               return true;
           })
           .exceptionally(throwable -> {
               if (throwable.getCause() instanceof LimitExceededException) {
                   logger.error("Cannot replace policy, as only one policy is
allowed per key.");
                   return false;
               }
               throw new RuntimeException("Error replacing policy", throwable);
           });
   }


   /**
```

```
     * Asynchronously retrieves the key policy for the specified key ID and policy
 name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
 policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
 policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
```

```
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
```

```
                throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();
```

```
        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS) key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
    }

    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
     * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
```

```
     *
     * @param keyId the ID of the KMS key to delete
     * @return a {@link CompletableFuture} that completes when the key deletion is
 scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
 ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for key
 ID: " + keyId, throwable);
            });
    }


    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey

- EnableKey
- Encrypt
- GetKeyPolicy
- ListAliases
- ListGrants
- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

Ações

`CreateAlias`

O código de exemplo a seguir mostra como usar `CreateAlias`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
```

```
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte CreateAlias em Referência AWS SDK for Java 2.x da API.

`CreateGrant`

O código de exemplo a seguir mostra como usar `CreateGrant`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
    *
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
    */
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
       List<GrantOperation> grantPermissions = List.of(
           GrantOperation.ENCRYPT,
           GrantOperation.DECRYPT,
           GrantOperation.DESCRIBE_KEY
       );

       CreateGrantRequest grantRequest = CreateGrantRequest.builder()
           .keyId(keyId)
           .name("grant1")
           .granteePrincipal(granteePrincipal)
           .operations(grantPermissions)
           .build();

       CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex == null) {
               logger.info("Grant created successfully with ID: " +
response.grantId());
           } else {
               if (ex instanceof KmsException kmsEx) {
                   throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
               }
           }
       });
```

```
        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }
```

- Para obter detalhes da API, consulte CreateGrant em Referência AWS SDK for Java 2.x da API.

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }
```

- Para obter detalhes da API, consulte CreateKey em Referência AWS SDK for Java 2.x da API.

`Decrypt`

O código de exemplo a seguir mostra como usar `Decrypt`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously decrypts the given encrypted data using the specified key ID.
    *
    * @param encryptedData The encrypted data to be decrypted.
    * @param keyId The ID of the key to be used for decryption.
    * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
    *         If an error occurs during the decryption process, the
CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
String.
    */
   public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
       DecryptRequest decryptRequest = DecryptRequest.builder()
           .ciphertextBlob(encryptedData)
           .keyId(keyId)
           .build();

       CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
       responseFuture.whenComplete((decryptResponse, exception) -> {
           if (exception == null) {
               logger.info("Data decrypted successfully for key ID: " + keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
```

```
                    throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK for Java 2.x .

`DeleteAlias`

O código de exemplo a seguir mostra como usar `DeleteAlias`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
```

```
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }
```

- Para obter detalhes da API, consulte DeleteAlias em Referência AWS SDK for Java 2.x da API.

`DescribeKey`

O código de exemplo a seguir mostra como usar `DescribeKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
```

```
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}", keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- Para obter detalhes da API, consulte DescribeKey em Referência AWS SDK for Java 2.x da API.

`DisableKey`

O código de exemplo a seguir mostra como usar `DisableKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS) key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();
```

```
        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
    }
```

- Para obter detalhes da API, consulte DisableKey em Referência AWS SDK for Java 2.x da API.

`EnableKey`

O código de exemplo a seguir mostra como usar `EnableKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
```

```
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte EnableKey em Referência AWS SDK for Java 2.x da API.

`Encrypt`

O código de exemplo a seguir mostra como usar `Encrypt`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Encrypts the given text asynchronously using the specified KMS client and key
ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
```

```
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK for Java 2.x .

`ListAliases`

O código de exemplo a seguir mostra como usar `ListAliases`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
```

```
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }
```

- Para obter detalhes da API, consulte ListAliases em Referência AWS SDK for Java 2.x da API.

`ListGrants`

O código de exemplo a seguir mostra como usar `ListGrants`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
     * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }
```

- Para obter detalhes da API, consulte ListGrants em Referência AWS SDK for Java 2.x da API.

`ListKeyPolicies`

O código de exemplo a seguir mostra como usar `ListKeyPolicies`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously retrieves the key policy for the specified key ID and policy
name.
    *
    * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
    * @param policyName the name of the key policy to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
    */
   public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
       GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
           .keyId(keyId)
           .policyName(policyName)
           .build();

       return getAsyncClient().getKeyPolicy(policyRequest)
           .thenApply(response -> {
               String policy = response.policy();
               logger.info("The response is: " + policy);
               return policy;
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Failed to get key policy", ex);
           });
   }
```

- Para obter detalhes da API, consulte ListKeyPolicies em Referência AWS SDK for Java 2.x da API.

`ListKeys`

O código de exemplo a seguir mostra como usar `ListKeys`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
```

```
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
 is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Para obter detalhes da API, consulte ListKeys em Referência AWS SDK for Java 2.x da API.

`RevokeGrant`

O código de exemplo a seguir mostra como usar `RevokeGrant`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
           } else {
               if (exception instanceof KmsException kmsEx) {
                   if (kmsEx.getMessage().contains("Grant does not exist")) {
                       logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                   } else {
                       throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                   }
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
               }
           }
       });

       return responseFuture;
   }
```

- Para obter detalhes da API, consulte RevokeGrant em Referência AWS SDK for Java 2.x da API.

`ScheduleKeyDeletion`

O código de exemplo a seguir mostra como usar `ScheduleKeyDeletion`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
    * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
           .build();
```

```
        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for key
 ID: " + keyId, throwable);
            });
    }
```

- Para obter detalhes da API, consulte ScheduleKeyDeletion em Referência AWS SDK for Java 2.x da API.

`Sign`

O código de exemplo a seguir mostra como usar `Sign`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
```

```
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
```

```
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }
```

- Consulte detalhes da API em Sign na Referência da API do AWS SDK for Java 2.x .

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
```

```
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }
```

- Para obter detalhes da API, consulte TagResource em Referência AWS SDK for Java 2.x da API.

Exemplos do Lambda usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes em todo o mundo AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

- Exemplos sem servidor
- AWS contribuições da comunidade

Conceitos básicos

Olá, Lambda

O exemplo de código a seguir mostra como começar a usar o Lambda.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/*
 *  Lambda function names appear as:
 *
 *  arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 *  To find this value, look at the function in the AWS Management Console.
 *
 *  Before running this Java code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see this documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
```

```
 *  This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

            Usage:
                <functionName> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\s
                role - The AWS Identity and Access Management (IAM) service role
 that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
 example.Handler::handleRequest).\s
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
 name that contains the .zip or .jar used to update the Lambda function's code.\s
                key - The Amazon S3 key name that represents the .zip or .jar (for
 example, LambdaHello-1.0-SNAPSHOT.jar).
                """;

        if (args.length != 5) {
              System.out.println(usage);
              return;
        }

        String functionName = args[0];
        String role = args[1];
        String handler = args[2];
        String bucketName = args[3];
        String key = args[4];
        LambdaClient awsLambda = LambdaClient.builder()
            .build();
```

```
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Lambda Basics scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS Lambda function.");
        String funArn = createLambdaFunction(awsLambda, functionName, key,
bucketName, role, handler);
        System.out.println("The AWS Lambda ARN is " + funArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get the " + functionName + " AWS Lambda function.");
        getFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List all AWS Lambda functions.");
        listFunctions(awsLambda);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Invoke the Lambda function.");
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it
again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
 Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();
```

```
            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response = awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
```

```
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {
        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
```

```
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the code for an AWS Lambda function.
     *
     * @param awsLambda  the AWS Lambda client
     * @param functionName the name of the Lambda function to update
     * @param bucketName the name of the S3 bucket where the function code is
located
     * @param key the key (file name) of the function code in the S3 bucket
     * @throws LambdaException if there is an error updating the function code
     */
    public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
                .functionName(functionName)
                .publish(true)
                .s3Bucket(bucketName)
                .s3Key(key)
                .build();

            UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
            GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
                .waitUntilFunctionUpdated(getFunctionConfigRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The last modified value is " +
response.lastModified());
```

```
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the configuration of an AWS Lambda function.
     *
     * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
     * @param functionName  the name of the AWS Lambda function to update
     * @param handler       the new handler for the AWS Lambda function
     *
     * @throws LambdaException if there is an error while updating the function
configuration
     */
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName  the name of the Lambda function to be deleted
     *
     * @throws LambdaException if an error occurs while deleting the Lambda function
     */
```

```
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();
```

```
            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateFunction em Referência AWS SDK for Java 2.x da API.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName  the name of the Lambda function to be deleted
```

```
     *
     * @throws LambdaException if an error occurs while deleting the Lambda function
     */
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteFunction em Referência AWS SDK for Java 2.x da API.

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
```

```
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response = awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte GetFunction em Referência AWS SDK for Java 2.x da API.

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
```

```
     */
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {
        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK for Java 2.x .

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Updates the code for an AWS Lambda function.
    *
    * @param awsLambda  the AWS Lambda client
    * @param functionName the name of the Lambda function to update
    * @param bucketName the name of the S3 bucket where the function code is
located
    * @param key the key (file name) of the function code in the S3 bucket
    * @throws LambdaException if there is an error updating the function code
    */
   public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
       try {
           LambdaWaiter waiter = awsLambda.waiter();
           UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
               .functionName(functionName)
               .publish(true)
               .s3Bucket(bucketName)
               .s3Key(key)
               .build();

           UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
           GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .build();

           WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
               .waitUntilFunctionUpdated(getFunctionConfigRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("The last modified value is " +
response.lastModified());

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte UpdateFunctionCode em Referência AWS SDK for Java 2.x da API.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates the configuration of an AWS Lambda function.
    *
    * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
    * @param functionName  the name of the AWS Lambda function to update
    * @param handler       the new handler for the AWS Lambda function
    *
    * @throws LambdaException if there is an error while updating the function
configuration
    */
   public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
       try {
           UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .handler(handler)
               .runtime(Runtime.JAVA17)
               .build();

           awsLambda.updateFunctionConfiguration(configurationRequest);

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte UpdateFunctionConfiguration em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback

dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon SNS

Usar Step Functions para invocar funções do Lambda

O exemplo de código a seguir mostra como criar uma máquina de AWS Step Functions estados que invoca AWS Lambda funções em sequência.

SDK para Java 2.x

Mostra como criar um fluxo de trabalho AWS sem servidor usando AWS Step Functions e o. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Funções de etapa

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Java 2.x

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Use a API do Neptune para consultar dados de gráficos

O exemplo de código a seguir mostra como usar a API do Neptune para consultar dados de gráficos.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Neptune para criar uma função do Lambda que consulta dados de gráficos na VPC.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Lambda
- Neptune

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
 APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
```

```
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                 PreparedStatement statement = connection.prepareStatement("SELECT ?
+ ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
```

```
        ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
```

```
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }

}
```

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do DynamoDB com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
```

```
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();
```

```
            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
```

```
        }

    }
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";
```

```
        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
 {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
             }
         }
```

```
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Java 2.x

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Java.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos do Amazon Lex usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Lex.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar um chatbot Amazon Lex

O exemplo de código a seguir mostra como criar um chatbot para engajar os visitantes do seu site.

SDK para Java 2.x

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Exemplos do Amazon Location usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Location.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Location

O exemplo de código a seguir mostra como começar a usar o Amazon Location Service.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you need to create a collection using the AWS Management
 * console. For information, see the following documentation.
 *
 * https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
public class HelloLocation {

    private static LocationAsyncClient locationAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(HelloLocation.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance.
    private static LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
```

```
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    public static void main(String[] args) {
        final String usage = """

            Usage:
                <collectionName>

            Where:
                collectionName - The Amazon location collection name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        listGeofences(collectionName);
    }

    /**
     * Lists geofences from a specified geofence collection asynchronously.
     *
     * @param collectionName The name of the geofence collection to list geofences
from.
     * @return A {@link CompletableFuture} representing the result of the
asynchronous operation.
```

```
     *         The future completes when all geofences have been processed and
 logged.
     */
    public static CompletableFuture<Void> listGeofences(String collectionName) {
        ListGeofencesRequest geofencesRequest = ListGeofencesRequest.builder()
                .collectionName(collectionName)
                .build();

        ListGeofencesPublisher paginator =
 getClient().listGeofencesPaginator(geofencesRequest);
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            if (response.entries().isEmpty()) {
                logger.info("No Geofences were found in the collection.");
            } else {
                response.entries().forEach(geofence ->
                        logger.info("Geofence ID: " + geofence.geofenceId())
                );
            }
        });
        return future;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - ListGeofenceCollections
  - ListGeofences

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um mapa do Amazon Location.
- Criar uma chave de API do Amazon Location.
- Exibir o URL do mapa.
- Criar uma coleção de geocercas
- Armazenar uma geometria de geocerca.

- Criar um recurso de rastreador.
- Atualizar a posição de um dispositivo.
- Recuperar a atualização de posição mais recente de um dispositivo específico.
- Criar uma calculadora de rotas.
- Determinar a distância entre Seattle e Vancouver.
- Usar as APIs de nível superior do Amazon Location.
- Excluir os ativos do Amazon Location.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon Location Service.

```
/*
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class LocationScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(LocationScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static LocationActions locationActions = new LocationActions();

    public static void main(String[] args) {
        final String usage = """
```

```
            Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
<calculatorName> <deviceId>

            Where:
              mapName - The name of the map to be create (e.g., "AWSMap").
              keyName - The name of the API key to create (e.g., "AWSApiKey").
              collectionName - The name of the geofence collection (e.g.,
"AWSLocationCollection").
              geoId - The geographic identifier used for the geofence or map (e.g.,
"geoId").
              trackerName - The name of the tracker (e.g., "geoTracker").
              calculatorName - The name of the route calculator (e.g.,
"AWSRouteCalc").
              deviceId - The ID of the device (e.g., "iPhone-112356").
            """;

        if (args.length != 7) {
            logger.info(usage);
            return;
        }

        String mapName = args[0];
        String keyName = args[1];
        String collectionName = args[2];
        String geoId = args[3];
        String trackerName = args[4];
        String calculatorName = args[5];
        String deviceId = args[6];

        logger.info("""
            AWS Location Service is a fully managed service offered by Amazon Web
Services (AWS) that
            provides location-based services for developers. This service simplifies
            the integration of location-based features into applications, making it
            easier to build and deploy location-aware applications.

            The AWS Location Service offers a range of location-based services,
            including:

            Maps: The service provides access to high-quality maps, satellite
imagery,\s
            and geospatial data from various providers, allowing developers to\s
            easily embed maps into their applications.
```

```
            Tracking: The Location Service enables real-time tracking of mobile
devices,\s
            assets, or other entities, allowing developers to build applications\s
            that can monitor the location of people, vehicles, or other objects.

            Geocoding: The service provides the ability to convert addresses or\s
            location names into geographic coordinates (latitude and longitude),\s
            and vice versa, enabling developers to integrate location-based search\s
            and routing functionality into their applications.
            """);
        waitForInputToContinue(scanner);
        try {
            runScenario(mapName, keyName, collectionName, geoId, trackerName,
calculatorName, deviceId);
        } catch (RuntimeException e) {
            // Clean up AWS Resources.
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
            logger.info(e.getMessage());
        }
    }

    public static void runScenario(String mapName, String keyName, String
collectionName, String geoId, String trackerName, String calculatorName, String
deviceId) {
        logger.info(DASHES);
        logger.info("1. Create a map");
        logger.info("""
             An AWS Location map can enhance the user experience of your
             application by providing accurate and personalized location-based
             features. For example, you could use the geocoding capabilities to
             allow users to search for and locate businesses, landmarks, or
             other points of interest within a specific region.
            """);

        waitForInputToContinue(scanner);
        String mapArn;
        try {
            mapArn = locationActions.createMap(mapName).join();
            logger.info("The Map ARN is: {}", mapArn);  // Log success in calling
code
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
```

```
                logger.error("The request exceeded the maximum quota: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the map.",
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Location API key");
        logger.info("""
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.
            """);

        try {
            String keyArn = locationActions.createKey(keyName, mapArn).join();
            logger.info("The API key was successfully created: {}", keyArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof AccessDeniedException) {
                logger.error("Request was denied: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the API
key.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Display Map URL");
        logger.info("""
            In order to get the MAP URL, you need to get the API Key value.
            You can get the key value using the AWS Management Console under
            Location Services. This operation cannot be completed using the
            AWS SDK. For more information about getting the key value, see
```

```
            the AWS Location Documentation.
            """);
        String mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/"+mapName+"/
tiles/{z}/{x}/{y}?key={KeyValue}";
        logger.info("Embed this URL in your Web app: " + mapUrl);
        logger.info("");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a geofence collection, which manages and stores
 geofences.");
        waitForInputToContinue(scanner);
        try {
            String collectionArn =
 locationActions.createGeofenceCollection(collectionName).join();
            logger.info("The geofence collection was successfully created: {}",
 collectionArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the
 geofence collection.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Store a geofence geometry in a given geofence collection.");
        logger.info("""
            An AWS Location geofence is a virtual boundary that defines a geographic
 area
            on a map. It is a useful feature for tracking the location of
            assets or monitoring the movement of objects within a specific region.

            To define a geofence, you need to specify the coordinates of a
            polygon that represents the area of interest. The polygon must be
            defined in a counter-clockwise direction, meaning that the points of
            the polygon must be listed in a counter-clockwise order.
```

```
            This is a requirement for the AWS Location service to correctly
            interpret the geofence and ensure that the location data is
            accurately processed within the defined area.
            """);

        waitForInputToContinue(scanner);
        try {
            locationActions.putGeofence(collectionName, geoId).join();
            logger.info("Successfully created geofence: {}", geoId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("A validation error occurred while creating geofence:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Create a tracker resource which lets you retrieve current
and historical location of devices..");
        waitForInputToContinue(scanner);
        try {
            String trackerArn = locationActions.createTracker(trackerName).join();
            logger.info("Successfully created tracker. ARN: {}", trackerArn);  //
Log success
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred while creating the tracker: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("7. Update the position of a device in the location tracking
system.");
        logger.info("""
            The AWS location service does not enforce a strict format for deviceId,
but it must:
              - Be a string (case-sensitive).
              - Be 1-100 characters long.
              - Contain only:
                - Alphanumeric characters (A-Z, a-z, 0-9)
                - Underscores (_)
                - Hyphens (-)
                - Be the same ID used when sending and retrieving positions.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<BatchUpdateDevicePositionResponse> future =
locationActions.updateDevicePosition(trackerName, deviceId);
            future.join();
            logger.info(deviceId + " was successfully updated in the location
tracking system.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. Retrieve the most recent position update for a specified
device..");
        waitForInputToContinue(scanner);
        try {
            GetDevicePositionResponse response =
locationActions.getDevicePosition(trackerName, deviceId).join();
```

```
            logger.info("Successfully fetched device position: {}",
response.position());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Create a route calculator.");
        waitForInputToContinue(scanner);
        try {
            CreateRouteCalculatorResponse response =
locationActions.createRouteCalculator(calculatorName).join();
            logger.info("Route calculator created successfully: {}",
response.calculatorArn());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.info("A conflict occurred: {}", cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("10. Determine the distance between Seattle and Vancouver using
the route calculator.");
        waitForInputToContinue(scanner);
        try {
            CalculateRouteResponse response =
locationActions.calcDistanceAsync(calculatorName).join();
```

```
            logger.info("Successfully calculated route. The distance in kilometers
is {}", response.summary().distance());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("11. Use the GeoPlacesAsyncClient to perform additional
operations.");
        logger.info("""
            This scenario will show use of the GeoPlacesClient that enables
            location search and geocoding capabilities for your applications.\s

            We are going to use this client to perform these AWS Location tasks:
             - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
             - Place Search (searchText): Finds places based on search queries.
             - Nearby Search (searchNearby): Finds places near a specific location.
            """);

        logger.info("First we will perform a Reverse Geocoding operation");
        waitForInputToContinue(scanner);
        try {
            locationActions.reverseGeocode().join();
            logger.info("Now we are going to perform a text search using coffee
shop.");
            waitForInputToContinue(scanner);
            locationActions.searchText("coffee shop").join();
            waitForInputToContinue(scanner);

            logger.info("Now we are going to perform a nearby Search.");
            //waitForInputToContinue(scanner);
            locationActions.searchNearBy().join();
            waitForInputToContinue(scanner);
        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                logger.error("A validation error occurred: {}", cause.getMessage(),
cause);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info("12. Delete the AWS Location Services resources.");
        logger.info("Would you like to delete the AWS Location Services resources?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(" This concludes the AWS Location Service scenario.");
        logger.info(DASHES);
    }

    /**
     * Cleans up resources by deleting the specified map, key, geofence collection,
tracker, and route calculator.
     *
     * @param mapName The name of the map to delete.
     * @param keyName The name of the key to delete.
     * @param collectionName The name of the geofence collection to delete.
     * @param trackerName The name of the tracker to delete.
     * @param calculatorName The name of the route calculator to delete.
     */
    private static void cleanUp(String mapName, String keyName, String
collectionName, String trackerName, String calculatorName) {
        try {
            locationActions.deleteMap(mapName).join();
```

```
            locationActions.deleteKey(keyName).join();
            locationActions.deleteGeofenceCollectionAsync(collectionName).join();
            locationActions.deleteTracker(trackerName).join();
            locationActions.deleteRouteCalculator(calculatorName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
 cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
 cause);
            }
            return;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper para os métodos de SDK do Amazon Location Service.

```
public class LocationActions {

    private static LocationAsyncClient locationAsyncClient;

    private static GeoPlacesAsyncClient geoPlacesAsyncClient;
```

```
    private static final Logger logger =
LoggerFactory.getLogger(LocationActions.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance is used throughout the application.
    private LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    private static GeoPlacesAsyncClient getGeoPlacesClient() {
        if (geoPlacesAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();
```

```
            geoPlacesAsyncClient = GeoPlacesAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return geoPlacesAsyncClient;
    }


    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
```

```
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }


    /**
     * Searches for a place using the provided search query and prints the detailed
information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
```

```
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                }
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                });
    }



    /**
     * Performs reverse geocoding using the AWS Geo Places API.
     * Reverse geocoding is the process of converting geographic coordinates
 (latitude and longitude) to a human-readable address.
     * This method uses the latitude and longitude of San Francisco as the input,
 and prints the resulting address.
     */
    public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        logger.info("Use latitude 37.7749 and longitude -122.4194");

        // AWS expects [longitude, latitude].
```

```
        List<Double> queryPosition = List.of(longitude, latitude);
        ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
            .queryPosition(queryPosition)
            .build();
        CompletableFuture<ReverseGeocodeResponse> futureResponse =
            getGeoPlacesClient().reverseGeocode(request);

        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: " +
cause.getMessage(), cause);
                }
                throw new CompletionException("Error performing reverse geocoding",
exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }


    /**
     * Calculates the distance between two locations asynchronously.
     *
     * @param routeCalcName the name of the route calculator to use
     * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
     */
    public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
        // Define coordinates for Seattle, WA and Vancouver, BC.
        List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
        List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

        CalculateRouteRequest request = CalculateRouteRequest.builder()
            .calculatorName(routeCalcName)
            .departurePosition(departurePosition)
            .destinationPosition(arrivePosition)
            .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
```

```
            .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
            .build();

        return getClient().calculateRoute(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new route calculator with the specified name and data source.
     *
     * @param routeCalcName the name of the route calculator to be created
     */
    public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
        String dataSource = "Esri"; // or "Here"
        CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
            .calculatorName(routeCalcName)
            .dataSource(dataSource)
            .build();

        return getClient().createRouteCalculator(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
```

```
    }


    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
                }
            });
    }


    /**
     * Updates the position of a device in the location tracking system.
     *
     * @param trackerName the name of the tracker associated with the device
     * @param deviceId    the unique identifier of the device
     * @throws RuntimeException if an error occurs while updating the device
position
     */
    public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
        double latitude = 37.7749;  // Example: San Francisco
        double longitude = -122.4194;
```

```
        DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
            .deviceId(deviceId)
            .sampleTime(Instant.now()) // Timestamp of position update.
            .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
            .build();

        BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
            .trackerName(trackerName)
            .updates(positionUpdate)
            .build();

        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
                }
            }
        });
    }


    /**
     * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
     *
     * @param trackerName the name of the tracker to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
     */
    public CompletableFuture<String> createTracker(String trackerName) {
        CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
            .description("Created using the Java V2 SDK")
            .trackerName(trackerName)
```

```
            .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
            .build();

        return getClient().createTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
    }



    /**
     * Adds a new geofence to the specified collection.
     *
     * @param collectionName the name of the geofence collection to add the geofence
to
     * @param geoId          the unique identifier for the geofence
     */
    public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
        // Define the geofence geometry (polygon).
        GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
            .polygon(List.of(
                List.of(
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();
```

```
        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new geofence collection.
     *
     * @param collectionName the name of the geofence collection to be created
     */
    public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
```

```
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the ARN
    }




    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
```

```
    }



    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }



    /**
     * Deletes a geofence collection asynchronously.
     *
```

```
    * @param collectionName the name of the geofence collection to be deleted
    * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
    */
   public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
       DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .build();

       return getClient().deleteGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested geofence
collection was not found.", cause);
                   }
                   throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
               }
               logger.info("The geofence collection {} was deleted.",
collectionName);
           })
           .thenApply(response -> null);
   }



   /**
    * Deletes the specified key from the key-value store.
    *
    * @param keyName the name of the key to be deleted
    * @return a {@link CompletableFuture} that completes when the key has been
deleted
    * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
    */
   public CompletableFuture<Void> deleteKey(String keyName) {
       DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
           .keyName(keyName)
           .build();
```

```
        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
            })
            .thenApply(response -> null);
    }


    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();

        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }
```

```
    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();

        return getClient().deleteTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }


    /**
     * Deletes a route calculator from the system.
     *
     * @param calcName the name of the route calculator to delete
     * @return a {@link CompletableFuture} that completes when the route calculator
has been deleted
     * @throws CompletionException if an error occurs while deleting the route
calculator
```

```
     *                              - If the route calculator was not found, a {@link
 ResourceNotFoundException} will be thrown
     *                              - If any other error occurs, a generic {@link
 CompletionException} will be thrown
     */
    public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
        DeleteRouteCalculatorRequest calculatorRequest =
 DeleteRouteCalculatorRequest.builder()
            .calculatorName(calcName)
            .build();

        return getClient().deleteRouteCalculator(calculatorRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was not
 found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
 calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection

- DeleteKey
- DeleteMap
- DeleteRouteCalculator
- DeleteTracker
- GetDevicePosition
- PutGeofence

Ações

`BatchUpdateDevicePosition`

O código de exemplo a seguir mostra como usar `BatchUpdateDevicePosition`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates the position of a device in the location tracking system.
    *
    * @param trackerName the name of the tracker associated with the device
    * @param deviceId    the unique identifier of the device
    * @throws RuntimeException if an error occurs while updating the device
position
    */
   public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
       double latitude = 37.7749;  // Example: San Francisco
       double longitude = -122.4194;

       DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
           .deviceId(deviceId)
           .sampleTime(Instant.now()) // Timestamp of position update.
```

```
            .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
            .build();

        BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
            .trackerName(trackerName)
            .updates(positionUpdate)
            .build();

        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
                }
            }
        });
    }
```

- Para obter detalhes da API, consulte BatchUpdateDevicePosition em Referência AWS SDK for Java 2.x da API.

`CalculateRoute`

O código de exemplo a seguir mostra como usar `CalculateRoute`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Calculates the distance between two locations asynchronously.
    *
    * @param routeCalcName the name of the route calculator to use
    * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
    */
   public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
       // Define coordinates for Seattle, WA and Vancouver, BC.
       List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
       List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

       CalculateRouteRequest request = CalculateRouteRequest.builder()
           .calculatorName(routeCalcName)
           .departurePosition(departurePosition)
           .destinationPosition(arrivePosition)
           .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
           .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
           .build();

       return getClient().calculateRoute(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
               }
           });
   }
```

- Para obter detalhes da API, consulte CalculateRoute em Referência AWS SDK for Java 2.x da API.

`CreateGeofenceCollection`

O código de exemplo a seguir mostra como usar `CreateGeofenceCollection`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new geofence collection.
    *
    * @param collectionName the name of the geofence collection to be created
    */
   public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
       CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .description("Created by using the AWS SDK for Java")
           .build();

       return getClient().createGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                   }
                   throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
               }
           })
           .thenApply(response -> response.collectionArn()); // Return only the ARN
   }
```

- Para obter detalhes da API, consulte CreateGeofenceCollection em Referência AWS SDK for Java 2.x da API.

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new API key with the specified name and restrictions.
    *
    * @param keyName the name of the API key to be created
    * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
    * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
    * or {@code null} if the operation failed
    */
   public CompletableFuture<String> createKey(String keyName, String mapArn) {
       ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
           .allowActions("geo:GetMap*")
           .allowResources(mapArn)
           .build();

       CreateKeyRequest request = CreateKeyRequest.builder()
           .keyName(keyName)
           .restrictions(keyRestrictions)
           .noExpiry(true)
           .build();

       return getClient().createKey(request)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }
```

- Para obter detalhes da API, consulte CreateKey em Referência AWS SDK for Java 2.x da API.

`CreateMap`

O código de exemplo a seguir mostra como usar `CreateMap`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
```

```
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }
```

- Para obter detalhes da API, consulte CreateMap em Referência AWS SDK for Java 2.x da API.

`CreateRouteCalculator`

O código de exemplo a seguir mostra como usar `CreateRouteCalculator`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new route calculator with the specified name and data source.
    *
    * @param routeCalcName the name of the route calculator to be created
    */
   public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
       String dataSource = "Esri"; // or "Here"
       CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
           .calculatorName(routeCalcName)
           .dataSource(dataSource)
           .build();

       return getClient().createRouteCalculator(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                   }
                   throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
               }
           });
   }
```

- Para obter detalhes da API, consulte CreateRouteCalculator em Referência AWS SDK for Java 2.x da API.

`CreateTracker`

O código de exemplo a seguir mostra como usar `CreateTracker`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
    *
    * @param trackerName the name of the tracker to be created
    * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
    */
   public CompletableFuture<String> createTracker(String trackerName) {
       CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
           .description("Created using the Java V2 SDK")
           .trackerName(trackerName)
           .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
           .build();

       return getClient().createTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
   }
```

- Para obter detalhes da API, consulte CreateTracker em Referência AWS SDK for Java 2.x da API.

DeleteGeofenceCollection

O código de exemplo a seguir mostra como usar `DeleteGeofenceCollection`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a geofence collection asynchronously.
    *
    * @param collectionName the name of the geofence collection to be deleted
    * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
    */
   public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
       DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .build();

       return getClient().deleteGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested geofence
collection was not found.", cause);
                   }
                   throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
               }
```

```
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte DeleteGeofenceCollection em Referência AWS SDK for Java 2.x da API.

`DeleteKey`

O código de exemplo a seguir mostra como usar `DeleteKey`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes the specified key from the key-value store.
     *
     * @param keyName the name of the key to be deleted
     * @return a {@link CompletableFuture} that completes when the key has been
deleted
     * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
     */
    public CompletableFuture<Void> deleteKey(String keyName) {
        DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
            .keyName(keyName)
            .build();

        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
            })
            .thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte DeleteKey em Referência AWS SDK for Java 2.x da API.

DeleteMap

O código de exemplo a seguir mostra como usar `DeleteMap`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();
```

```
        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }
```

- Para obter detalhes da API, consulte DeleteMap em Referência AWS SDK for Java 2.x da API.

`DeleteRouteCalculator`

O código de exemplo a seguir mostra como usar `DeleteRouteCalculator`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a route calculator from the system.
     *
     * @param calcName the name of the route calculator to delete
     * @return a {@link CompletableFuture} that completes when the route calculator
has been deleted
     * @throws CompletionException if an error occurs while deleting the route
calculator
```

```
    *                               - If the route calculator was not found, a {@link
ResourceNotFoundException} will be thrown
    *                               - If any other error occurs, a generic {@link
CompletionException} will be thrown
    */
   public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
       DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
           .calculatorName(calcName)
           .build();

       return getClient().deleteRouteCalculator(calculatorRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The route calculator was not
found.", cause);
                   }
                   throw new CompletionException("Failed to delete the route
calculator: " + exception.getMessage(), exception);
               }
               logger.info("The route calculator {} was deleted.", calcName);
           })
           .thenApply(response -> null);
   }
```

- Para obter detalhes da API, consulte DeleteRouteCalculator em Referência AWS SDK for Java 2.x da API.

`DeleteTracker`

O código de exemplo a seguir mostra como usar `DeleteTracker`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a tracker with the specified name.
    *
    * @param trackerName the name of the tracker to be deleted
    * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
    * @throws CompletionException if an error occurs while deleting the tracker
    *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
    *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
    */
   public CompletableFuture<Void> deleteTracker(String trackerName) {
       DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
           .trackerName(trackerName)
           .build();

       return getClient().deleteTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The tracker was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
               }
               logger.info("The tracker {} was deleted.", trackerName);
           })
           .thenApply(response -> null); // Ensures CompletableFuture<Void>
   }
```

- Para obter detalhes da API, consulte DeleteTracker em Referência AWS SDK for Java 2.x da API.

`GetDevicePosition`

O código de exemplo a seguir mostra como usar `GetDevicePosition`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the position of a device using the provided LocationClient.
    *
    * @param trackerName The name of the tracker associated with the device.
    * @param deviceId    The ID of the device to retrieve the position for.
    * @throws RuntimeException If there is an error fetching the device position.
    */
   public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
       GetDevicePositionRequest request = GetDevicePositionRequest.builder()
           .trackerName(trackerName)
           .deviceId(deviceId)
           .build();

       return getClient().getDevicePosition(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
               }
           });
   }
```

- Para obter detalhes da API, consulte GetDevicePosition em Referência AWS SDK for Java 2.x da API.

`PutGeofence`

O código de exemplo a seguir mostra como usar `PutGeofence`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Adds a new geofence to the specified collection.
    *
    * @param collectionName the name of the geofence collection to add the geofence
to
    * @param geoId          the unique identifier for the geofence
    */
   public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
       // Define the geofence geometry (polygon).
       GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
           .polygon(List.of(
               List.of(
                   List.of(-122.3381, 47.6101), // First point
                   List.of(-122.3281, 47.6101),
                   List.of(-122.3281, 47.6201),
                   List.of(-122.3381, 47.6201),
                   List.of(-122.3381, 47.6101) // Closing the polygon
               )
           ))
           .build();

       PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
           .collectionName(collectionName) // Specify the collection.
           .geofenceId(geoId) // Unique ID for the geofence.
           .geometry(geofenceGeometry)
           .build();

       return getClient().putGeofence(geofenceRequest)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte PutGeofence em Referência AWS SDK for Java 2.x da API.

Exemplos do Location Service Places usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com Location Service Places.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ReverseGeocode`

O código de exemplo a seguir mostra como usar `ReverseGeocode`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Performs reverse geocoding using the AWS Geo Places API.
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       logger.info("Use latitude 37.7749 and longitude -122.4194");

       // AWS expects [longitude, latitude].
       List<Double> queryPosition = List.of(longitude, latitude);
       ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
           .queryPosition(queryPosition)
           .build();
       CompletableFuture<ReverseGeocodeResponse> futureResponse =
           getGeoPlacesClient().reverseGeocode(request);

       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                   throw new CompletionException("A validation error occurred: " +
cause.getMessage(), cause);
               }
               throw new CompletionException("Error performing reverse geocoding",
exception);
           }

           response.resultItems().forEach(result ->
```

```
                logger.info("The address is: " + result.address().label())
            );
        });
    }
```

- Para obter detalhes da API, consulte ReverseGeocode em Referência AWS SDK for Java 2.x da API.

`SearchNearby`

O código de exemplo a seguir mostra como usar `SearchNearby`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }
```

- Para obter detalhes da API, consulte SearchNearby em Referência AWS SDK for Java 2.x da API.

`SearchText`

O código de exemplo a seguir mostra como usar `SearchText`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
     * Searches for a place using the provided search query and prints the detailed
 information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
 coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
```

```
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                }
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                });
    }
```

- Para obter detalhes da API, consulte SearchText em Referência AWS SDK for Java 2.x da API.

AWS Marketplace Exemplos de API de catálogo usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS SDK for Java 2.x with AWS Marketplace Catalog.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Produtos de AMI
- Ofertas de parceiros de canal
- Produtos de contêiner
- Entidades

- Ofertas
- Produtos
- Autorização de revenda
- Produtos de SaaS do
- Serviços públicos

Produtos de AMI

Adicionar uma dimensão a um produto existente da AMI e atualize os termos de preços da oferta

O exemplo de código a seguir mostra como adicionar uma dimensão a um produto de AMI existente e atualizar os termos de preços da oferta.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
```

```
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
```

```
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Adicionar uma região onde um produto de AMI é implantado

O exemplo de código a seguir mostra como adicionar uma região em que um produto de AMI é implantado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar um produto de AMI público ou limitado e uma oferta pública com preços anuais calculados por hora

O exemplo de código a seguir mostra como criar um produto de AMI público ou limitado e uma oferta pública com preços anuais calculados por hora. Este exemplo cria um EULA padrão ou personalizado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar um produto de AMI público ou limitado e uma oferta pública com preços mensais calculados por hora

O exemplo de código a seguir mostra como criar um produto de AMI público ou limitado e uma oferta pública com preços mensais calculados por hora. Este exemplo cria um EULA padrão ou personalizado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
```

```
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
```

```
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
```

```
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "RecurringPaymentTerm",
                        "CurrencyCode": "USD",
                        "BillingPeriod": "Monthly",
                        "Price": "15.0"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar um produto de AMI público ou limitado e uma oferta pública com preços calculados por hora

O exemplo de código a seguir mostra como criar um produto de AMI público ou limitado e uma oferta pública com preços por hora. Este exemplo cria um EULA padrão ou personalizado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
```

```
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
```

```
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
```

```
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly pricing for AmiProduct
using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar o rascunho de um produto de AMI com um rascunho de oferta pública

O exemplo de código a seguir mostra como criar um rascunho de produto de AMI com um rascunho de oferta pública.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
```

```
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Restringir uma região onde um produto de AMI é implantado

O exemplo de código a seguir mostra como restringir uma região em que um produto de AMI é implantado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Limitar a visibilidade do produto

O exemplo de código a seguir mostra como restringir a visibilidade do produto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Especificar se os ativos da AMI são implantados em novas regiões

O exemplo de código a seguir mostra como especificar se os ativos da AMI serão implantados em novas regiões criadas AWS para oferecer suporte a futuras regiões.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Ofertas de parceiros de canal

Criar um rascunho de CPPO para qualquer tipo de produto

O exemplo de código a seguir mostra como criar um rascunho de CPPO para qualquer tipo de produto a fim de analisá-lo internamente antes de publicar para os compradores.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer",
                "Description": "Test product"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada de substituição de autorização de revenda com preços contratuais

O exemplo de código a seguir mostra como criar uma oferta privada de substituição de autorização de revenda com base em um contrato existente com preços contratuais.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-1111111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private resale replacement offer with contract
 pricing for SaaSProduct"
```

```
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Listar todas as CPPOs criadas por um parceiro de canal

O exemplo de código a seguir mostra como listar todos os CPPOs criados por um parceiro de canal.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllCppoOffers {

 /*
  * List all CPPOs created by a channel partner
  */
 public static void main(String[] args) {

  List<String> cppoOfferIds = getAllCppoOfferIds();

  ReferenceCodesUtils.formatOutput(cppoOfferIds);
 }

 public static List<String> getAllCppoOfferIds() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all offer entity ids
  List<String> entityIdList = new ArrayList<String>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
```

```
 for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
  entityIdList.add(entitySummary.entityId());
 }

 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

  for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
   entityIdList.add(entitySummary.entityId());
  }
 }

 // filter for CPPO offers: ResaleAuthorizationId exists in Details

 List<String> cppoOfferIds = new ArrayList<String>();

 for (String entityId : entityIdList) {
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(entityId)
    .build();
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);

  Document resaleAuthorizationDocument =
describeEntityResponse.detailsDocument().asMap().get(ATTRIBUTE_RESALE_AUTHORIZATION_ID);
  String resaleAuthorizationId = resaleAuthorizationDocument != null ?
resaleAuthorizationDocument.asString() : "";

  if (!resaleAuthorizationId.isEmpty()) {
      cppoOfferIds.add(resaleAuthorizationId);
  }
 }
 return cppoOfferIds;
}
```

```
}
```

- Para obter detalhes da API, consulte ListEntities em Referência AWS SDK for Java 2.x da API.

Listar todas as autorizações de revenda compartilhadas disponíveis para um parceiro de canal

O exemplo de código a seguir mostra como listar todas as autorizações de revenda compartilhadas disponíveis para um parceiro de canal.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllSharedResaleAuthorizations {

 /*
  * list all resale authorizations shared to an account
  */
```

```
public static void main(String[] args) {

 List<ListEntitiesResponse> responseList = getListEntityResponseList();
 ReferenceCodesUtils.formatOutput(responseList);
}

public static List<ListEntitiesResponse> getListEntityResponseList() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 List<ListEntitiesResponse> responseList = new ArrayList<ListEntitiesResponse>();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
   .maxResults(10)
   .ownershipType(OWNERSHIP_TYPE_SHARED)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 responseList.add(listEntitiesResponse);

 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest = ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
    .maxResults(10)
    .ownershipType(OWNERSHIP_TYPE_SHARED)
    .nextToken(listEntitiesResponse.nextToken())
    .build();

  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

  responseList.add(listEntitiesResponse);
 }
 return responseList;
```

```
 }

}
```

- Para obter detalhes da API, consulte ListEntities em Referência AWS SDK for Java 2.x da API.

Publicar uma CPPO e anexar um EULA do comprador

O exemplo de código a seguir mostra como publicar uma CPPO e acrescentar um EULA do comprador.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
                {
                 "Type": "CustomEula",
                 "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                }
               ]
           }
        ]
          }
         },
         {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
         },
         {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
         },
         {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar uma CPPO usando autorização única de revenda e atualizar a margem de preço

O exemplo de código a seguir mostra como publicar uma CPPO usando autorização única de revenda em produtos de AMI, de SaaS ou de contêiner e atualizar a margem de preço.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar um rascunho de CPPO e atualizar a margem de preço

O exemplo de código a seguir mostra como publicar um rascunho de CPPO e atualizar a margem de preço.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar a data de validade de uma CPPO

O exemplo de código a seguir mostra como atualizar a data de validade de uma CPPO para dar aos compradores mais tempo para avaliar e aceitar a oferta.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Produtos de contêiner

Criar o rascunho de um produto de contêiner com um rascunho de oferta pública

O exemplo de código a seguir mostra como criar um rascunho de produto de contêiner com um rascunho de oferta pública.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog":"AWSMarketplace",
    "changeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
```

```
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar um produto de contêiner limitado com uma oferta pública e preços calculados por contrato

O exemplo de código a seguir mostra como criar um produto de contêiner limitado com uma oferta pública, preços contratuais e EULA padrão.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
                    "Name": "Cores",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
```

```
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
                "Description": "Some interesting container offer description"
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Entidades

Descrever todas as entidades em uma única chamada

O exemplo de código a seguir mostra como descrever todas as entidades em uma única chamada.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesRequest;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityDetail;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeErrorDetail;

import java.util.Arrays;
import java.util.Map;

public class BatchDescribeEntities {

    /*
     * BatchDescribe my entities in a single call and
     *  check if it contains all the information I need to know about the entities.
     */
    public static void main(String[] args) {

        MarketplaceCatalogClient marketplaceCatalogClient =
                MarketplaceCatalogClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        BatchDescribeEntitiesRequest batchDescribeEntitiesRequest =
                BatchDescribeEntitiesRequest.builder()
                        .entityRequestList(Arrays.asList(
                                EntityRequest.builder()
                                        .catalog(AWS_MP_CATALOG).entityId(OFFER_ID)
                                        .build(),
                                EntityRequest.builder()

 .catalog(AWS_MP_CATALOG).entityId(PRODUCT_ID)
                                        .build()))
                        .build();

        BatchDescribeEntitiesResponse batchDescribeEntitiesResponse =
 marketplaceCatalogClient.batchDescribeEntities(batchDescribeEntitiesRequest);

        // Reading the successful entities response
        Map<String, EntityDetail> entityDetailsMap =
 batchDescribeEntitiesResponse.entityDetails();
```

```
        for (Map.Entry<String, EntityDetail> entry : entityDetailsMap.entrySet()) {
            System.out.println("EntityId: " + entry.getKey());
            ReferenceCodesUtils.formatOutput(entry.getValue());
        }

        // Logging the failed entities error details
        Map<String, BatchDescribeErrorDetail> entityErrorsMap =
 batchDescribeEntitiesResponse.errors();
        for (Map.Entry<String, BatchDescribeErrorDetail> entry :
 entityErrorsMap.entrySet()) {
            System.out.println(String.format("EntityId: %s, ErrorCode: %s,
 ErrorMessage: %s", entry.getKey(),
                    entry.getValue().errorCode(), entry.getValue().errorMessage()));
        }
    }
}
```

- Para obter detalhes da API, consulte BatchDescribeEntities em Referência AWS SDK for Java 2.x da API.

Listar e descrever todas as ofertas associadas a um produto

O exemplo de código a seguir mostra como listar e descrever todas as ofertas associadas a um produto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
```

```
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPrivateOffers {

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();
 /*
  * retrieve all private offer information related to a single product
  */
 public static void main(String[] args) {

  List<EntitySummary> entitySummaryList = getEntitySummaryList();

  // for each offer id, output the offer detail using DescribeEntity API

  for (EntitySummary entitySummary : entitySummaryList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entitySummary.entityId())
     .build();
   DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
```

```
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}
public static List<EntitySummary> getEntitySummaryList() {
 // define list entities filters

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .productId(OfferProductIdFilter.builder()
       .valueList(PRODUCT_ID)
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER).maxResults(50)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 // save all entitySummary of the results into entitySummaryList

 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

 while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER).maxResults(50)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
```

```
    listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);
    entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
   }
   return entitySummaryList;
  }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DescribeEntity
  - ListEntities

Ofertas

Criar uma dimensão personalizada para um produto de SaaS e criar uma oferta privada

O exemplo de código a seguir mostra como criar uma dimensão personalizada para um produto de SaaS e criar uma oferta privada.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
```

```
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar um rascunho de oferta privada para um produto de AMI ou SaaS

O exemplo de código a seguir mostra como criar um rascunho de oferta privada para um produto de AMI ou de SaaS a fim de analisá-lo internamente antes de publicá-lo para os compradores.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Crie uma oferta privada com contrato e Pay-As-You-Go preços para um produto SaaS

O exemplo de código a seguir mostra como criar uma oferta privada com contrato e Pay-As-You-Go preços para um produto SaaS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
```

```
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada com preços contratuais e uma programação de pagamento flexível para um produto de SaaS

O exemplo de código a seguir mostra como criar uma oferta privada com preços contratuais e uma programação de pagamento flexível para um produto de SaaS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            },
                            {
                                "DimensionKey": "PremiumService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada com preços contratuais para um produto de contêiner

O exemplo de código a seguir mostra como criar uma oferta privada com preços contratuais para um produto de contêiner.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
 Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
 contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada com preços contratuais para um produto de AMI

O exemplo de código a seguir mostra como criar uma oferta privada com preços contratuais para um produto de AMI.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
```

```
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada com preços anuais calculados por hora e uma programação de pagamento flexível para um produto de AMI

O exemplo de código a seguir mostra como criar uma oferta privada com preços anuais calculados por hora e uma programação de pagamento flexível para um produto de AMI.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P365D",
                        "Grants": [
                            {
                                "DimensionKey": "t2.micro",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada com preços anuais calculados por hora para um produto de AMI

O exemplo de código a seguir mostra como criar uma oferta privada com preços anuais calculados por hora para um produto de AMI.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "220.00"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada com preços por hora para um produto de AMI

O exemplo de código a seguir mostra como criar uma oferta privada com preços calculados por hora para um produto de AMI.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
```

```
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-01-01"
```

```
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada com preços de assinatura para um produto de SaaS

O exemplo de código a seguir mostra como criar uma oferta privada com preços de assinatura para um produto de SaaS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
```

```
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
```

```
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada com preços contratuais por níveis para um produto de SaaS

O exemplo de código a seguir mostra como criar uma oferta privada com preços contratuais por níveis para um produto de SaaS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
```

```
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta pública de teste gratuito com preços de assinatura para um produto de SaaS

O exemplo de código a seguir mostra como criar uma oferta pública de teste gratuito com preços de assinatura de um produto de SaaS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar uma oferta privada de substituição com preços contratuais

O exemplo de código a seguir mostra como criar uma oferta privada de substituição com base em um contrato existente com preços contratuais.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract pricing
for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Descrever uma oferta pública

O exemplo de código a seguir mostra como descrever uma oferta pública.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
```

```
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Para obter detalhes da API, consulte DescribeEntity em Referência AWS SDK for Java 2.x da API.

Expirar um rascunho de oferta privada

O exemplo de código a seguir mostra como definir a data de validade de uma oferta privada como uma data anterior para que os compradores não vejam mais a oferta.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-01-01"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Listar todas as ofertas privadas

O exemplo de código a seguir mostra como listar todas as ofertas privadas.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilterDate
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferBuyerAccountsFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilterDateRange;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListAllPrivateOffers {

 /*
  * List all my private offers and sort or filter them by Offer Publish Date, Offer
 Expiry Date and Buyer IDs
  *
  * OfferTargetingFilter = BuyerAccounts (private offer);
  * OfferBuyerAccountsFilter: Buyer IDs filter
  * OfferAvailabilityEndDateFilter : Offer Expiry Date filter
  * OfferReleaseDateFilter : Offer Publish Date filter
  */

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 public static void main(String[] args) {

  String offerReleaseDateAfterValue = "2023-01-01T23:59:59Z";
  String offerAvailableEndDateAfterValue = "2040-12-24T23:59:59Z";
```

```
 List<EntitySummary> entitySummaryList =
getEntitySummaryList(offerReleaseDateAfterValue, offerAvailableEndDateAfterValue);

 // for each offer id, output the offer detail using DescribeEntity API


 for (EntitySummary entitySummary : entitySummaryList) {
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(entitySummary.entityId())
    .build();
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}

public static List<EntitySummary> getEntitySummaryList (String
offerReleaseDateAfterValue, String offerAvailableEndDateAfterValue) {

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .buyerAccounts(OfferBuyerAccountsFilter.builder()
       .wildCardValue(BUYER_ACCOUNT_ID)
       .build())
     .availabilityEndDate(OfferAvailabilityEndDateFilter.builder()
       .dateRange(OfferAvailabilityEndDateFilterDateRange.builder()
         .afterValue(offerAvailableEndDateAfterValue).build())
       .build())
     .releaseDate(OfferReleaseDateFilter.builder()
       .dateRange(OfferReleaseDateFilterDateRange.builder()
         .afterValue(offerReleaseDateAfterValue)
         .build())
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
```

```
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER).maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }

  return entitySummaryList;
 }

}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Listar ofertas públicas e privadas lançadas para um ID de produto específico

O exemplo de código a seguir mostra como listar ofertas públicas e privadas lançadas para um ID de produto específico.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPublicOrPrivateReleasedOffers {

 /*
  * List released Public/Private offers for a specific product id.
  * Example below is to list released public offers.
  * To change to released private offers, change OFFER_TARGETING_NONE (None) to
 OFFER_TARGETING_BUYERACCOUNTS(BuyerAccounts)
  */
 public static void main(String[] args) {
```

```
 List<EntitySummary> entitySummaryList = getEntitySummaryLIst();
 ReferenceCodesUtils.formatOutput(entitySummaryList);
}

public static List<EntitySummary> getEntitySummaryLIst() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // define list entities filters

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_NONE)
       .build())
     .state(OfferStateFilter.builder()
       .valueListWithStrings(OFFER_STATE_RELEASED)
       .build())
     .productId(OfferProductIdFilter.builder()
       .valueList(PRODUCT_ID)
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER)
   .maxResults(10)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 // save all entitySummary of the results into entitySummaryList

 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();
```

```
  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }
  return entitySummaryList;
 }

}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar uma oferta para aplicar um contrato com Pay-As-You-Go preços

O exemplo de código a seguir mostra como atualizar uma oferta para aplicar um contrato com Pay-As-You-Go preços.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
```

```
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar uma oferta para aplicar preços anuais calculados por hora

O exemplo de código a seguir mostra como atualizar uma oferta para aplicar preços anuais calculados por hora.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
```

```
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar uma oferta para aplicar a segmentação a regiões geográficas específicas

O exemplo de código a seguir mostra como atualizar uma oferta para aplicar segmentação a regiões geográficas específicas.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
```

```
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar nome e descrição de uma oferta pública

O exemplo de código a seguir mostra como atualizar o nome e a descrição de uma oferta pública.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar o EULA de uma oferta

O exemplo de código a seguir mostra como atualizar o EULA de uma oferta.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar a data de validade de uma oferta privada para uma data futura

O exemplo de código a seguir mostra como atualizar a data de validade de uma oferta privada para uma data futura a fim de dar aos compradores mais tempo para avaliar e aceitar a oferta.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
```

```
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar a duração do teste gratuito de uma oferta pública de teste gratuito de um produto de SaaS

O exemplo de código a seguir mostra como atualizar a duração do teste gratuito de uma oferta pública de teste gratuito de um produto de SaaS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
```

```
                                {
                                    "DimensionKey": "WorkloadSmall"
                                },
                                {
                                    "DimensionKey": "WorkloadMedium"
                                }
                            ]
                        }
                    ]
                }
            }
        ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar a política de reembolso de uma oferta

O exemplo de código a seguir mostra como atualizar a política de reembolso de uma oferta.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Produtos

Descrever um produto de AMI, SaaS ou contêiner

O exemplo de código a seguir mostra como descrever um produto de AMI, de SaaS ou de contêiner e verificar se ele contém todas as informações que você deseja saber sobre o produto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Para obter detalhes da API, consulte DescribeEntity em Referência AWS SDK for Java 2.x da API.

Listar todos os produtos de AMI, SaaS ou Container e ofertas públicas associadas

O exemplo de código a seguir mostra como listar todos os produtos de AMI, de SaaS ou de contêiner e ofertas públicas associadas.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListEntities {

 /*
```

```
 * List all my AMI or SaaS or Container products and associated public offers
 */
public static void main(String[] args) {

 Map<String, List<EntitySummary>> allProductsWithOffers =
getAllProductsWithOffers();

 ReferenceCodesUtils.formatOutput(allProductsWithOffers);
}

public static Map<String, List<EntitySummary>> getAllProductsWithOffers() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 Map<String, List<EntitySummary>> allProductsWithOffers = new HashMap<String,
List<EntitySummary>> ();

 // get all product entities
 List<EntitySummary> productEntityList = new ArrayList<EntitySummary>();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(PRODUCT_TYPE_AMI)
   .maxResults(10)
   .nextToken(null)
   .build();


 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 productEntityList.addAll(listEntitiesResponse.entitySummaryList());


 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
```

```
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
  productEntityList.addAll(listEntitiesResponse.entitySummaryList());
 }

 // loop through each product entity and get the public released offers associated
using product id filter

 for ( EntitySummary productEntitySummary : productEntityList) {
  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(productEntitySummary.entityId())
        .build())
      .build())
    .build();

  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> offerEntitySummaryList = new ArrayList<EntitySummary>();

  offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
```

```
    while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
     listEntitiesRequest =
       ListEntitiesRequest.builder()
       .catalog(AWS_MP_CATALOG)
       .entityType(ENTITY_TYPE_OFFER)
       .maxResults(10)
       .entityTypeFilters(entityTypeFilters)
       .nextToken(listEntitiesResponse.nextToken())
       .build();
     listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
     offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
    }

    // save final results into map; key = product id; value = offer entity summary
 list

    allProductsWithOffers.put(productEntitySummary.entityId(),
 offerEntitySummaryList);
   }
   return allProductsWithOffers;
 }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DescribeEntity
  - ListEntities

Autorização de revenda

Criar um rascunho de autorização de revenda

O exemplo de código a seguir mostra como criar um rascunho de autorização de revenda para qualquer tipo de produto a fim de analisá-lo internamente antes de publicar para um parceiro de canal.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Descrever uma autorização de revenda

O exemplo de código a seguir mostra como descrever uma autorização de revenda.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
```

```
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Para obter detalhes da API, consulte DescribeEntity em Referência AWS SDK for Java 2.x da API.

Publicar uma autorização única de revenda com uma oferta privada

O exemplo de código a seguir mostra como publicar uma autorização de revenda única com uma oferta privada para que um parceiro de canal possa usá-la a fim de criar uma oferta privada de parceiro de canal (CPPO).

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar a autorização de revenda multiuso com data de validade

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso com uma data de validade para um produto de AMI com preços anuais calculados por hora a fim de que um parceiro de canal possa usá-la para criar uma CPPO.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
```

```
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar a autorização de revenda multiuso com data de validade e um EULA

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar a autorização de revenda multiuso com data de validade e documentação do contrato de revendedor

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
```

```
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
```

```
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar a autorização de revenda multiuso com data de validade e adicionar uma conta de comprador específica

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
```

```
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar a autorização de revenda multiuso sem data de validade

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso sem data de validade para um produto de AMI com preços anuais calculados por hora para que um CP possa usá-la para criar uma CPPO.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar a autorização de revenda multiuso sem data de validade e um EULA

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
```

```
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar a autorização de revenda multiuso sem data de validade e documentação do contrato de revendedor

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
```

```
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar a autorização de revenda multiuso sem data de validade e adicionar uma conta de comprador específica

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
```

```
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar uma autorização de revenda única e adicionar uma programação de pagamento flexível

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar uma programação de pagamento flexível.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
                            "DimensionKey": "Users",
                            "MaxQuantity": 10
                          }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ResalePaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2023-09-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2023-12-01",
                                "ChargeAmount": "250.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-06-30",
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar uma autorização de revenda única e adicionar um EULA

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar uma autorização de revenda única e adicionar uma conta de comprador específica

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": "1"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar uma autorização de revenda única e adicionar a documentação do contrato de revendedor

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar uma autorização de revenda única e adicionar se é uma renovação

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar se é uma renovação.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
```

```
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
```

```
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Restringir autorização de revenda

O exemplo de código a seguir mostra como restringir a autorização de revenda.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar o nome e a descrição da autorização de revenda única ou multiuso

O exemplo de código a seguir mostra como atualizar o nome e a descrição da autorização de revenda única ou multiuso antes da publicação de qualquer tipo de produto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Produtos de SaaS do

Criar um rascunho de um produto de SaaS com um rascunho de oferta pública

O exemplo de código a seguir mostra como criar um rascunho de produto de SaaS com um rascunho de oferta pública.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {
              "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
```

```
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar um produto de SaaS público ou limitado e uma oferta pública com preços calculados por contrato

O exemplo de código a seguir mostra como criar um produto de SaaS público ou limitado e uma oferta pública com preços contratuais. Este exemplo cria um EULA padrão ou personalizado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
```

```
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Crie um produto SaaS público ou limitado e uma oferta pública com Pay-As-You-Go contrato e preços

O exemplo de código a seguir mostra como criar um produto SaaS público ou limitado e uma oferta pública com um contrato com Pay-As-You-Go preços. Este exemplo cria um EULA padrão ou personalizado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
```

```
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
```

```
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
```

```
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Criar um produto de SaaS público ou limitado e uma oferta pública com preços de assinatura

O exemplo de código a seguir mostra como criar um produto de SaaS público ou limitado e uma oferta pública com preços de assinatura. Este exemplo cria um EULA padrão ou personalizado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
```

```
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
```

```
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar um produto de SaaS e uma oferta pública associada

O exemplo de código a seguir mostra como publicar um produto de SaaS e a oferta pública associada. Por padrão, o produto estará em um estado limitado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
```

```
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
```

```
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
```

```
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Publicar um produto de SaaS e uma oferta pública associada com base em um rascunho existente

O exemplo de código a seguir mostra como publicar um produto de SaaS e uma oferta pública associada com base em um rascunho existente. Por padrão, o produto estará em um estado limitado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Public"
            }
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Atualizar dimensões em um produto de AMI ou de SaaS

O exemplo de código a seguir mostra como atualizar dimensões em um produto de AMI ou de SaaS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Para executar esse exemplo, passe o conjunto de alterações JSON a seguir para `RunChangesets` em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

Serviços públicos

Utilitários para iniciar um conjunto de alterações

O exemplo de código a seguir mostra como definir utilitários para iniciar um conjunto de alterações.

SDK para Java 2.x

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Utilitário para carregar um conjunto de alterações de um arquivo JSON e começar a processá-lo.

```
package com.example.awsmarketplace.catalogapi;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.protocols.json.internal.unmarshall.document.DocumentUnmarshaller;
import software.amazon.awssdk.protocols.jsoncore.JsonNodeParser;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.Change;
import software.amazon.awssdk.services.marketplacecatalog.model.Entity;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetResponse;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSet;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSetEntity;
import com.example.awsmarketplace.catalogapi.Entity.Root;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
import com.example.awsmarketplace.utils.StringSerializer;

/**
 * Before running this Java V2 code example, convert all Details attribute to
 DetailsDocument if any
 */

public class RunChangesets {

 private static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 public static void main(String[] args) {

  // input json can be specified here or passed from input parameter
  String inputChangeSetFile = "changeSets/offers/
CreateReplacementOfferFromAGWithContractPricingDetailDocument.json";

  if (args.length > 0)
   inputChangeSetFile = args[0];

  // parse the input changeset file to string for process
  String changeSetsInput = readChangeSetToString(inputChangeSetFile);

  // process the changeset request
  try {
   StartChangeSetResponse result = getChangeSetRequestResult(changeSetsInput);
   ReferenceCodesUtils.formatOutput(result);
  } catch (Exception e) {
   e.printStackTrace();
  }
```

```
}

public static StartChangeSetResponse getChangeSetRequestResult(String
changeSetsInput) throws IOException {

 //set up AWS credentials
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 //changeset list to save all the changesets in the changesets file
 List<Change> changeSetLists = new ArrayList<Change>();

 // read all changesets into object
 Root root = GSON.fromJson(changeSetsInput, Root.class);

 // process each changeset and add each changeset request to changesets list
 for (ChangeSet cs : root.changeSet) {

  ChangeSetEntity entity = cs.Entity;
  String entityType = entity.Type;
  String entityIdentifier = StringUtils.defaultIfBlank(entity.Identifier, null);
  Document detailsDocument = getDocumentFromObject(cs.DetailsDocument);

  Entity awsEntity =
    Entity.builder()
    .type(entityType)
    .identifier(entityIdentifier)
    .build();

  Change inputChangeRequest =
    Change.builder()
    .changeType(cs.ChangeType)
    .changeName(cs.ChangeName)
    .entity(awsEntity)
    .detailsDocument(detailsDocument)
    .build();

  changeSetLists.add(inputChangeRequest);
 }

 // process all changeset requests
```

```
 StartChangeSetRequest startChangeSetRequest =
   StartChangeSetRequest.builder()
   .catalog(root.catalog)
   .changeSet(changeSetLists)
   .build();

 StartChangeSetResponse result =
marketplaceCatalogClient.startChangeSet(startChangeSetRequest);

 return result;
}

public static Document getDocumentFromObject(Object detailsObject) {

 String detailsString = "{}";
 try {
  detailsString = IOUtils.toString(new
ByteArrayInputStream(GSON.toJson(detailsObject).getBytes()), "UTF-8");
 } catch (IOException e) {
  e.printStackTrace();
 }

 JsonNodeParser jsonNodeParser = JsonNodeParser.create();
 Document doc = jsonNodeParser.parse(detailsString).visit(new
DocumentUnmarshaller());
 return doc;
}


public static String readChangeSetToString (String inputChangeSetFile) {

 InputStream changesetInputStream =
RunChangesets.class.getClassLoader().getResourceAsStream(inputChangeSetFile);

 String changeSetsInput = null;

 try {
  changeSetsInput = IOUtils.toString(changesetInputStream, "UTF-8");
 } catch (IOException e) {
  e.printStackTrace();
 }

 return changeSetsInput;
```

```
 }
}
```

- Para obter detalhes da API, consulte StartChangeSet em Referência AWS SDK for Java 2.x da API.

AWS Marketplace Exemplos de API de contrato usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS SDK for Java 2.x with AWS Marketplace Agreement.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Contratos

Contratos

Aceitar uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como aceitar uma solicitação de cancelamento de contrato iniciada pelo vendedor.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementCancellationReque
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementCancellationReque

public class AcceptAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";

    public static void main(String[] args) {
        acceptAgreementCancellationRequest();
    }

    private static void acceptAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        AcceptAgreementCancellationRequestRequest request =
                AcceptAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)

 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .build();

        AcceptAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.acceptAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Description: " + response.description());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
```

```
}
```

- Para obter detalhes da API, consulte AcceptAgreementCancellationRequest em Referência AWS SDK for Java 2.x da API.

Aceitar uma solicitação de pagamento por contrato

O exemplo de código a seguir mostra como aceitar uma solicitação de pagamento por contrato iniciada pelo vendedor.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementPaymentRequestReq
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementPaymentRequestRes

public class AcceptAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
    private static final String PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER
 REFERENCE HERE>";

    public static void main(String[] args) {
        acceptAgreementPaymentRequest();
    }
```

```
    private static void acceptAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        AcceptAgreementPaymentRequestRequest request =
                AcceptAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                        .build();

        AcceptAgreementPaymentRequestResponse response =
                marketplaceAgreementClient.acceptAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Name: " + response.name());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte AcceptAgreementPaymentRequest em Referência AWS SDK for Java 2.x da API.

Alterar um termo de renovação do contrato de SaaS

O exemplo de código a seguir mostra como alterar um contrato de SaaS para adicionar ou atualizar seu prazo de renovação.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RenewalTermConfiguration;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model and then
 turn on
 * the auto-renewal setting using the AWS Marketplace Agreement Service APIs.
 *
```

```
 * <p>Scenario: A buyer subscribes to a SaaS product using a public offer that
 supports
 * auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
 * auto-renewal via the RenewalTerm configuration.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code P1M} for
 1 month).</li>
 *   <li>{@code DIMENSION_1_KEY} — the dimension key defined in the offer.</li>
 * </ul>
 */
public class AmendSaaSContractRenewalTerm {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // The dimension key defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the RenewalTerm in your offer.
    private static final String RENEWAL_TERM_ID = "<your-renewal-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";
```

```
    // Term ID for the SupportTerm in your offer.
    private static final String SUPPORT_TERM_ID = "<your-support-term-id>";

    public static void main(String[] args) {
        amendSaaSContractAgreementRenewalTerm();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
disabled.
     * 2. Wait for entitlements to become active.
     * 3. Amend the agreement to enable auto-renewal.
     */
    private static void amendSaaSContractAgreementRenewalTerm() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build())
                                .build()))
                .build();

        // Initial agreement: auto-renewal disabled.
        RequestedTerm renewalTerm = RequestedTerm.builder()
                .id(RENEWAL_TERM_ID)

.configuration(RequestedTermConfiguration.fromRenewalTermConfiguration(
                        RenewalTermConfiguration.builder()
                                .enableAutoRenew(false)
                                .build()))
                .build();
```

```
        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm supportTerm =
RequestedTerm.builder().id(SUPPORT_TERM_ID).build();

        // --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, renewalTerm,
legalTerm, supportTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before amending.
        System.out.println("Waiting for entitlements to become active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("Entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Amend: enable auto-renewal ---
        RequestedTerm renewalTermAmended = RequestedTerm.builder()
                .id(RENEWAL_TERM_ID)
```

```
 .configuration(RequestedTermConfiguration.fromRenewalTermConfiguration(
                        RenewalTermConfiguration.builder()
                                .enableAutoRenew(true)
                                .build()))
                .build();

        // Use Intent.AMEND and sourceAgreementIdentifier to target the existing
 agreement.
        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.AMEND)
                        .requestedTerms(configurableUpfrontPricingTerm,
 renewalTermAmended, legalTerm, supportTerm)

 .sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Amend agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        System.out.println("Amendment accepted. Auto-renewal enabled. New
 AgreementId: " + aarResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK for Java 2.x da API.

Alterar uma AMI ConfigurableUpfrontPricingTerm para o modelo de preços de uso

O exemplo de código a seguir mostra como alterar um contrato de AMI ConfigurableUpfrontPricingTerm atualizando a quantidade da dimensão para o modelo de preços de uso.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
```

```
 * Demonstrates how to create an AMI agreement with ConfigurableUpfrontPricingTerm
 and then amend the dimension quantity
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: An AMI product with USAGE pricing requires two agreements:
 * <ol>
 *   <li>An agreement with <b>usageBasedPricingTerm (UBPT)</b> — accepted first to
 establish the base agreement.</li>
 *   <li>An agreement with <b>configurableUpfrontPricingTerm (CUPT)</b> — accepted
 after the UBPT agreement entitlements are active.</li>
 * </ol>
 * Once both agreement entitlements are available, this sample shows how to
 <b>amend</b> the agreement
 * with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} —  duration for the agreement
 *       (e.g., {@code P365D} for one year).</li>
 *   <li>{@code DIMENSION_1_KEY} — the dimension key defined in the offer (e.g.,
 instance type).</li>
 *   <li>{@code DIMENSION_1_VALUE} — initial quantity; {@code NEW_DIMENSION_1_VALUE}
 — amended quantity.</li>
 * </ul>
 */
public class AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P365D" for 365 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";
```

```
    // The dimension key defined in your offer (e.g., an EC2 instance type like
"c6gn.medium").
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Initial quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the UsageBasedPricingTerm in your offer.
    private static final String USAGE_TERM_ID = "<your-usage-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in your offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // New quantity to use when amending the dimension of CUPT.
    private static final int NEW_DIMENSION_1_VALUE = 5;

    public static void main(String[] args) {
        amendAmiCUPTAgreement();
    }

    /**
     * Full end-to-end flow:
     * 1. Create and accept an agreement request with usageBasedPricingTerm.
     * 2. Wait for entitlements to become active, then create and accept an
agreement request with configurableUpfrontPricingTerm (CUPT).
     * 3. Wait for CUPT entitlements to become active, then amend the dimension
quantity.
     */
    private static void amendAmiCUPTAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageTerm = RequestedTerm.builder()
                .id(USAGE_TERM_ID)
                .build();
        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
                .build();
```

```
        RequestedTerm validityTerm = RequestedTerm.builder()
                .id(VALIDITY_TERM_ID)
                .build();

        // --- Agreement with UBPT ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageTerm, legalTerm, validityTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created. AgreementRequestId:
" + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with UBPT accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before creating the agreement with
CUPT.
        System.out.println("Waiting for UBPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("UBPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)
```

```
.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, legalTerm,
validityTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Agreement request with CUPT created. AgreementRequestId:
" + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String cuptAgreementId = aarResponse.agreementId();
        System.out.println("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

        // Wait for entitlements to become active before amending.
        System.out.println("Waiting for CUPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse cuptEntitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, cuptAgreementId);
        System.out.println("CUPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(cuptEntitlementsResponse);
```

```
        // --- Amend Agreement with CUPT ---
        // Increase the dimension quantity using Intent.AMEND and
 sourceAgreementIdentifier.
        RequestedTerm newConfig = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

 .configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

 .dimensionValue(NEW_DIMENSION_1_VALUE) // Increase quantity for this dimension key
                                                    .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carAmendRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.AMEND)
                        .requestedTerms(newConfig, legalTerm, validityTerm)
                        .sourceAgreementIdentifier(cuptAgreementId)
                        .build();
        CreateAgreementRequestResponse carAmendResponse =
                marketplaceAgreementClient.createAgreementRequest(carAmendRequest);
        System.out.println("Amendment of CUPT agreement request created.
 AgreementRequestId: " + carAmendResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarAmendRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carAmendResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarAmendResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarAmendRequest);
        System.out.println("Amendment accepted. New AgreementId: " +
 aarAmendResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK for Java 2.x da API.

Crie um novo contrato de teste gratuito da AMI

O exemplo de código a seguir mostra como criar um novo contrato de teste gratuito da AMI.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create an AMI Free Trial agreement
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to an AMI product that offers a free trial
 period.
 * The free trial includes a FreeTrialPricingTerm alongside a UsageBasedPricingTerm.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
```

```
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 * </ul>
 */
public class NewAmiFreeTrial {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the FreeTrialPricingTerm in your offer.
    private static final String FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-
pricing-term-id>";

    // Term ID for the UsageBasedPricingTerm in your offer (applies after the trial
 ends).
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the SupportTerm in your offer.
    private static final String SUPPORT_TERM_ID = "<your-support-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    public static void main(String[] args) {
        createAndAcceptAmiFreeTrialAgreementRequest();
    }

    /**
     * Creates an AMI Free Trial agreement.
     * The FreeTrialPricingTerm grants access at no cost for the trial period.
     * The UsageBasedPricingTerm defines the charges that apply once the trial ends.
     */
    private static void createAndAcceptAmiFreeTrialAgreementRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();
```

```
        RequestedTerm freeTrialPricingTerm = RequestedTerm.builder()
                .id(FREE_TRIAL_PRICING_TERM_ID)
                .build();
        RequestedTerm usageBasedPricingTerm = RequestedTerm.builder()
                .id(USAGE_BASED_PRICING_TERM_ID)
                .build();
        RequestedTerm supportTerm = RequestedTerm.builder()
                .id(SUPPORT_TERM_ID)
                .build();
        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
                .build();

        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(freeTrialPricingTerm, usageBasedPricingTerm,
 supportTerm, legalTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with freeTrialPricingTerm accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK for Java 2.x da API.

Crie um novo contrato de SaaS com pagamento adiantado

O exemplo de código a seguir mostra como criar um novo contrato de SaaS com pagamento adiantado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import software.amazon.awssdk.services.marketplaceagreement.model.TaxConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
 * using the AWS Marketplace Agreement Service APIs.
```

```
 *
 * <p>Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
 * selecting an agreement duration and specifying quantities for multiple
 dimensions.
 * Tax estimation is enabled at the time of agreement creation.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code P12M} for
 12 months).</li>
 *   <li>{@code DIMENSION_1_KEY}, {@code DIMENSION_2_KEY} — dimension keys defined
 in the offer.</li>
 *   <li>{@code DIMENSION_1_VALUE}, {@code DIMENSION_2_VALUE} — quantities for each
 dimension.</li>
 * </ul>
 */
public class NewSaaSContractWithUpfrontPayment {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P12M" for 12 months).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // First dimension key and quantity defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-1-key>";
    private static final int DIMENSION_1_VALUE = 10;

    // Second dimension key and quantity defined in your offer.
    private static final String DIMENSION_2_KEY = "<your-dimension-2-key>";
    private static final int DIMENSION_2_VALUE = 20;
```

```
    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Tax estimation setting: "ENABLED" to include estimated taxes in the
agreement.
    private static final String TAX_ESTIMATION = "ENABLED";

    public static void main(String[] args) {
        createSaaSContractAgreement();
    }

    /**
     * Creates a SaaS agreement with CONTRACT pricing model with configurable
upfront pricing
     * for multiple dimensions and tax estimation.
     */
    private static void createSaaSContractAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build(),
                                            Dimension.builder()
                                                    .dimensionKey(DIMENSION_2_KEY)

.dimensionValue(DIMENSION_2_VALUE)
                                                    .build())
                                .build()))
                .build();

        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
```

```
                .build();

        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, legalTerm)

 .taxConfiguration(TaxConfiguration.builder().taxEstimation(TAX_ESTIMATION).build())
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("SaaS agreement request with CONTRACT pricing model
 accepted. AgreementId: " + acceptAgreementRequestResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK for Java 2.x da API.

Obter todos os IDs de contratos

O exemplo de código a seguir mostra como obter todos os IDs de contratos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreementsIds {

 /*
  * Get all purchase agreements ids for your PartyType (Proposer, or Acceptor)
  * Depend on the number of agreements in your account, this code may take some time
 to finish.
  */
 public static void main(String[] args) {

  List<String> agreementIds = getAllAgreementIds();

  ReferenceCodesUtils.formatOutput(agreementIds);
```

```
}

public static List<String> getAllAgreementIds() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
the acceptor.
 Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 List<Filter> searchFilters = new ArrayList<Filter>();

 searchFilters.addAll(Arrays.asList(partyType, agreementType));

 // Save all results in a list array
 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(searchFilters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
```

```
      .nextToken(searchAgreementsResponse.nextToken())
      .filters(searchFilters)
      .build();
    searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
    agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
   }

   List<String> agreementIds = new ArrayList<String>();
   for (AgreementViewSummary summary : agreementSummaryList) {
    agreementIds.add(summary.agreementId());
   }
   return agreementIds;
  }

}
```

- Para obter detalhes da API, consulte SearchAgreements em Referência AWS SDK for Java 2.x da API.

Obter todos os contratos

O exemplo de código a seguir mostra como obter todos os contratos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreements {

 /*
  * Get all purchase agreements for your PartyType (Proposer, or Acceptor)
  * Depend on the number of agreements in your account, this code may take some time
 to finish.
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAllAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAllAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

        // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return
 agreements where you are the acceptor.
        // Change to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements where you
 are the proposer.
  Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_ACCEPTOR).build();
```

```
  Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  List<Filter> searchFilters = new ArrayList<Filter>();

  searchFilters.addAll(Arrays.asList(partyType, agreementType));

  // Save all results in a list array

  List<AgreementViewSummary> agreementSummaryList = new
 ArrayList<AgreementViewSummary>();

  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(searchFilters)
    .build();

  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .nextToken(searchAgreementsResponse.nextToken())
     .filters(searchFilters).build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
   agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }
  return agreementSummaryList;
 }

}
```

- Para obter detalhes da API, consulte SearchAgreements em Referência AWS SDK for Java 2.x da API.

Obter o ID do cliente de um contrato

O exemplo de código a seguir mostra como obter um ID de cliente de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementCustomerInfo {

 /*
  * Obtain metadata about the customer who created the agreement, such as the
 customer's AWS Account ID
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Customer's AWS Account ID is " +
 describeAgreementResponse.acceptor().accountId());
```

```
 }

 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Para obter detalhes da API, consulte DescribeAgreement em Referência AWS SDK for Java 2.x da API.

Obtenha detalhes de uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como obter detalhes de uma solicitação específica de cancelamento de contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementCancellationRequestR
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementCancellationRequestR

public class GetAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";

    public static void main(String[] args) {
        getAgreementCancellationRequest();
    }

    private static void getAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        GetAgreementCancellationRequestRequest request =
                GetAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)

 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .build();

        GetAgreementCancellationRequestResponse response =
                marketplaceAgreementClient.getAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
```

```
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte GetAgreementCancellationRequest em Referência AWS SDK for Java 2.x da API.

Obtenha detalhes de uma solicitação de pagamento por contrato

O exemplo de código a seguir mostra como obter detalhes de uma solicitação de pagamento de um contrato específico.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementPaymentRequestReques
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementPaymentRequestRespon

public class GetAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";

    public static void main(String[] args) {
        getAgreementPaymentRequest();
    }
```

```
    private static void getAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        GetAgreementPaymentRequestRequest request =
                GetAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .build();

        GetAgreementPaymentRequestResponse response =
                marketplaceAgreementClient.getAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Name: " + response.name());
        System.out.println("Charge ID: " + response.chargeId());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte GetAgreementPaymentRequest em Referência AWS SDK for Java 2.x da API.

Obter detalhes financeiros de um contrato

O exemplo de código a seguir mostra como obter os detalhes financeiros de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementFinancialDetails {

 /*
  * Obtain financial details, such as Total Contract Value of the agreement from a
 given agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String totalContractValue = getTotalContractValue(agreementId);

  System.out.println("Total Contract Value is " + totalContractValue);

 }

 public static String getTotalContractValue(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
```

```
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  String totalContractValue = "N/A";

  if ( describeAgreementResponse.estimatedCharges() != null ) {
   totalContractValue =
 describeAgreementResponse.estimatedCharges().agreementValue()
     + " "
     + describeAgreementResponse.estimatedCharges().currencyCode();
  }
  return totalContractValue;
 }
}
```

- Para obter detalhes da API, consulte DescribeAgreement em Referência AWS SDK for Java 2.x da API.

Obter detalhes do teste gratuito de um contrato

O exemplo de código a seguir mostra como obter os detalhes do teste gratuito de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.FreeTrialPricingTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsFreeTrialDetails {

 /*
  * Obtain the details from an agreement of a free trial I have provided to the
 customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<FreeTrialPricingTerm> freeTrialPricingTerms =
 getFreeTrialPricingTerms(agreementId);

  ReferenceCodesUtils.formatOutput(freeTrialPricingTerms);
 }

 public static List<FreeTrialPricingTerm> getFreeTrialPricingTerms(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();
```

```
  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
     .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<FreeTrialPricingTerm> freeTrialPricingTerms = new
 ArrayList<FreeTrialPricingTerm>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.freeTrialPricingTerm() != null) {
    freeTrialPricingTerms.add(acceptedTerm.freeTrialPricingTerm());
   }
  }
  return freeTrialPricingTerms;
 }
}
```

- Para obter detalhes da API, consulte DescribeAgreement em Referência AWS SDK for Java 2.x da API.

Obter informações sobre um contrato

O exemplo de código a seguir mostra como obter informações sobre um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class DescribeAgreement {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse = getResponse(agreementId);

  ReferenceCodesUtils.formatOutput(describeAgreementResponse);

 }

 public static DescribeAgreementResponse getResponse(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Para obter detalhes da API, consulte DescribeAgreement em Referência AWS SDK for Java 2.x da API.

Obter detalhes do produto e da oferta de um contrato

O exemplo de código a seguir mostra como obter os detalhes de produtos e ofertas com base em um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class GetProductAndOfferDetailFromAgreement {

 public static void main(String[] args) {
```

```
 // call Agreement API to get offer and product information for the agreement

 String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

 List<DescribeEntityResponse> entityResponseList = getEntities(agreementId);

 for (DescribeEntityResponse response : entityResponseList) {
  ReferenceCodesUtils.formatOutput(response);
 }
}

public static List<DescribeEntityResponse> getEntities(String agreementId) {
 List<DescribeEntityResponse> entityResponseList = new
ArrayList<DescribeEntityResponse> ();

 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 DescribeAgreementRequest describeAgreementRequest =
   DescribeAgreementRequest.builder()
   .agreementId(agreementId)
   .build();

 DescribeAgreementResponse describeAgreementResponse =
marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

 // get offer id for the given agreement

 String offerId = describeAgreementResponse.proposalSummary().offerId();

 // get all the product ids for this agreement

 List<String> productIds = new ArrayList<String>();
 for (Resource resource : describeAgreementResponse.proposalSummary().resources())
{
  productIds.add(resource.id());
 }

 // call Catalog API to get the details of the offer and products

 MarketplaceCatalogClient marketplaceCatalogClient =
```

```
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId).build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);

  entityResponseList.add(describeEntityResponse);

  for (String productId : productIds) {
   describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(productId).build();
   describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   System.out.println("Print details for product " + productId);
   entityResponseList.add(describeEntityResponse);
  }
  return entityResponseList;
 }
}
```

- Para obter detalhes da API, consulte DescribeAgreement em Referência AWS SDK for Java 2.x da API.

Obter o EULA de um contrato

O exemplo de código a seguir mostra como obter o EULA de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.DocumentItem;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsEula {

 /*
  * Obtain the EULA I have entered into with my customer via the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DocumentItem> legalEulaArray = getLegalEula(agreementId);

  ReferenceCodesUtils.formatOutput(legalEulaArray);
 }

 public static List<DocumentItem> getLegalEula(String agreementId) {
```

```
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<DocumentItem> legalEulaArray = new ArrayList<>();

  getAgreementTermsResponse.acceptedTerms().stream()
      .filter(acceptedTerm -> acceptedTerm.legalTerm() != null &&
 acceptedTerm.legalTerm().hasDocuments())
      .flatMap(acceptedTerm -> acceptedTerm.legalTerm().documents().stream())
      .filter(docItem -> docItem.type() != null)
      .forEach(legalEulaArray::add);
  return legalEulaArray;
 }

}
```

- Para obter detalhes da API, consulte GetAgreementTerms em Referência AWS SDK for Java 2.x da API.

Obter os termos de renovação automática de um contrato

O exemplo de código a seguir mostra como obter os termos de renovação automática de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementAutoRenewal {

 /*
  * Obtain the auto-renewal status of the agreement
  */

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String autoRenewal = getAutoRenewal(agreementId);

  System.out.println("Auto-Renewal status is " + autoRenewal);
 }

 public static String getAutoRenewal(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
```

```
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  String autoRenewal = "No Auto Renewal";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.renewalTerm() != null &&
 acceptedTerm.renewalTerm().configuration() != null
     && acceptedTerm.renewalTerm().configuration().enableAutoRenew() != null) {
    autoRenewal =
 String.valueOf(acceptedTerm.renewalTerm().configuration().enableAutoRenew().booleanValue())
    break;
   }
  }
  return autoRenewal;
 }

}
```

- Para obter detalhes da API, consulte GetAgreementTerms em Referência AWS SDK for Java 2.x da API.

Obter as dimensões adquiridas em um contrato

O exemplo de código a seguir mostra como obter as dimensões compradas em um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionPurchased {

 /*
  * Obtain the dimensions the buyer has purchased from me via the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> dimensionKeys = getDimensionKeys(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionKeys);
 }
```

```
 public static List<String> getDimensionKeys(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<String> dimensionKeys = new ArrayList<String>();
  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !=
 null) {
     List<Dimension> dimensions =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     for (Dimension dimension : dimensions) {
      dimensionKeys.add(dimension.dimensionKey());
     }
    }

   }
  }
  return dimensionKeys;
 }
}
```

- Para obter detalhes da API, consulte GetAgreementTerms em Referência AWS SDK for Java 2.x da API.

Obter as instâncias de cada dimensão adquirida em um contrato

O exemplo de código a seguir mostra como obter as instâncias de cada dimensão compradas em um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionInstances {

 /*
  * get instances of each dimension that buyer has purchased in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  Map<String, List<Dimension>> dimensionMap = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionMap);
```

```
 }

 public static Map<String, List<Dimension>> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  Map<String, List<Dimension>> dimensionMap = new HashMap<String,
 List<Dimension>>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Dimension> dimensionsList = new ArrayList<Dimension>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    String selectorValue = "";
    if (acceptedTerm.configurableUpfrontPricingTerm().configuration() != null) {
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !=
 null) {
      selectorValue =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue();
     }
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().hasDimensions()) {
      dimensionsList =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     }
    }
    if (selectorValue.length() > 0) {
     dimensionMap.put(selectorValue, dimensionsList);
    }
   }
  }
  return dimensionMap;
 }
}
```

- Para obter detalhes da API, consulte GetAgreementTerms em Referência AWS SDK for Java 2.x da API.

Obter a programação de pagamento de um contrato

O exemplo de código a seguir mostra como obter a programação de pagamento de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentScheduleTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.ScheduleItem;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
public class GetAgreementTermsPaymentSchedule {

 /*
  * Obtain the payment schedule I have agreed to with the agreement, including the
 invoice date and invoice amount
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Map<String, Object>> paymentScheduleArray = getPaymentSchedules(agreementId);

  ReferenceCodesUtils.formatOutput(paymentScheduleArray);
 }

 public static List<Map<String, Object>> getPaymentSchedules(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  List<Map<String, Object>> paymentScheduleArray = new ArrayList<>();

  String currencyCode = "";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.paymentScheduleTerm() != null) {
    PaymentScheduleTerm paymentScheduleTerm = acceptedTerm.paymentScheduleTerm();
    if (paymentScheduleTerm.currencyCode() != null) {
     currencyCode = paymentScheduleTerm.currencyCode();
    }
    if (paymentScheduleTerm.hasSchedule()) {
     for (ScheduleItem schedule : paymentScheduleTerm.schedule()) {
      if (schedule.chargeDate() != null) {
       String chargeDate = schedule.chargeDate().toString();
       String chargeAmount = schedule.chargeAmount();
       Map<String, Object> scheduleMap = new HashMap<>();
```

```
        scheduleMap.put(ATTRIBUTE_CURRENCY_CODE, currencyCode);
        scheduleMap.put(ATTRIBUTE_CHARGE_DATE, chargeDate);
        scheduleMap.put(ATTRIBUTE_CHARGE_AMOUNT, chargeAmount);
        paymentScheduleArray.add(scheduleMap);
       }
      }
     }
    }
   }
   return paymentScheduleArray;
  }
}
```

- Para obter detalhes da API, consulte GetAgreementTerms em Referência AWS SDK for Java 2.x da API.

Obter o preço por dimensão em um contrato

O exemplo de código a seguir mostra como obter os preços por dimensão em um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
```

```
import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPricingEachDimension {

 /*
  * Obtain pricing per each dimension in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Object> dimensions = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensions);
 }

 public static List<Object> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<Object> dimensions = new ArrayList<Object>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Object> rateInfo = new ArrayList<Object>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if (acceptedTerm.configurableUpfrontPricingTerm().type() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().type());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().currencyCode() != null) {
```

```
      rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().currencyCode());
     }
     if (acceptedTerm.configurableUpfrontPricingTerm().hasRateCards()) {
      rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().rateCards());
     }
     dimensions.add(rateInfo);
    }
   }
   return dimensions;
  }
}
```

- Para obter detalhes da API, consulte GetAgreementTerms em Referência AWS SDK for Java 2.x da API.

Obter o tipo de preço de um contrato

O exemplo de código a seguir mostra como obter o tipo de preço de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import com.fasterxml.jackson.annotation.JsonAutoDetect.Visibility;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.Set;

import org.apache.commons.lang3.tuple.Triple;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

/*
 * Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
 */
public class GetAgreementPricingType {

 private static final String FILTER_NAME = "OfferId";

 private static final String FILTER_VALUE = OFFER_ID;
```

```
// Product types
private static final String SAAS_PRODUCT = "SaaSProduct";
private static final String AMI_PRODUCT = "AmiProduct";
private static final String ML_PRODUCT = "MachineLearningProduct";
private static final String CONTAINER_PRODUCT = "ContainerProduct";
private static final String DATA_PRODUCT = "DataProduct";
private static final String PROSERVICE_PRODUCT = "ProfessionalServicesProduct";
private static final String AIQ_PRODUCT = "AiqProduct";

// Pricing types
private static final String CCP = "CCP";
private static final String ANNUAL = "Annual";
private static final String CONTRACT = "Contract";
private static final String SFT = "SaaS Free Trial";
private static final String HMA = "Hourly and Monthly Agreements";
private static final String HOURLY = "Hourly";
private static final String MONTHLY = "Monthly";
private static final String AFPS = "Annual FPS";
private static final String CFPS = "Contract FPS";
private static final String CCPFPS = "CCP with FPS";
private static final String BYOL = "BYOL";
private static final String FREE = "Free";
private static final String FTH = "Free Trials and Hourly";

// Agreement term pricing types
private static final Set<String> LEGAL = Set.of("LegalTerm");
private static final Set<String> CONFIGURABLE_UPFRONT =
Set.of("ConfigurableUpfrontPricingTerm");
private static final Set<String> USAGE_BASED = Set.of("UsageBasedPricingTerm");
private static final Set<String> CONFIGURABLE_UPFRONT_AND_USAGE_BASED =
Set.of("ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm");
private static final Set<String> FREE_TRIAL = Set.of("FreeTrialPricingTerm");
private static final Set<String> RECURRING_PAYMENT =
Set.of("RecurringPaymentTerm");
private static final Set<String> USAGE_BASED_AND_RECURRING_PAYMENT =
Set.of("UsageBasedPricingTerm", "RecurringPaymentTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE =
Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED
= Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm",
"UsageBasedPricingTerm");
private static final Set<String> BYOL_PRICING = Set.of("ByolPricingTerm");
private static final Set<String> FREE_TRIAL_AND_USAGE_BASED =
Set.of("FreeTrialPricingTerm", "UsageBasedPricingTerm");
```

```
private static final List<Set<String>> ALL_AGREEMENT_TERM_TYPES_COMBINATION
= Arrays.asList(LEGAL, CONFIGURABLE_UPFRONT, USAGE_BASED,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
  FREE_TRIAL, RECURRING_PAYMENT, USAGE_BASED_AND_RECURRING_PAYMENT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, BYOL_PRICING,
FREE_TRIAL_AND_USAGE_BASED);

private static  MarketplaceAgreementClient marketplaceAgreementClient =
  MarketplaceAgreementClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

private static MarketplaceCatalogClient marketplaceCatalogClient =
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

   /*
    * Get agreement Pricing Type given product type, agreement term types and offer
types if needed
    */
public static String getPricingType(String productType, Set<String>
agreementTermType, Set<String> offerType) {
 Map<Triple<String, Set<String>, Set<String>>, String> pricingTypes = new
HashMap<>();

 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(ML_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
```

```
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FREE_TRIAL, new HashSet<>()), SFT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED_AND_RECURRING_PAYMENT, new
HashSet<>()), HMA);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(ML_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(ML_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), AFPS);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
```

```
 pricingTypes.put(Triple.of(AMI_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(ML_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(DATA_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, LEGAL, new HashSet<>()), FREE);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(ML_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);

 Triple<String, Set<String>, Set<String>> key = Triple.of(productType,
agreementTermType, offerType);

 if (pricingTypes.containsKey(key)) {
  return pricingTypes.get(key);
 } else {
  return "Unknown";
 }
}

/*
 * Given product type and agreement term types, some combinations need to check
offer term types as well.
 */
public static String needToCheckOfferTermsType(String productType, Set<String>
agreementTermTypes) {
 Map<KeyPair, String> offerTermTypes = new HashMap<>();
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE),
"Y");

 KeyPair key = new KeyPair(productType, agreementTermTypes);
 if (offerTermTypes.containsKey(key)) {
  return offerTermTypes.get(key);
```

```
 } else {
  return null;
 }
}

public static List<AgreementViewSummary> getAgreementsById() {

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
the acceptor.
 Filter partyType =
Filter.builder().name(PARTY_TYPE_FILTER_NAME).values(PARTY_TYPE_FILTER_VALUE_PROPOSER).buil

 Filter agreementType =
Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME).values(AGREEMENT_TYPE_FILTER_VALUE_PURCHA

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(partyType, agreementType, customizeFilter).build();

 SearchAgreementsResponse searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());

 while (searchResultResponse.nextToken() != null &&
searchResultResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
SearchAgreementsRequest.builder().catalog(AWS_MP_CATALOG)
    .filters(partyType,
agreementType).nextToken(searchResultResponse.nextToken()).build();
  searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
```

```
}

static class KeyPair {
 private final String first;
 private final Set<String> second;

 public KeyPair(String productType, Set<String> second) {
  this.first = productType;
  this.second = second;
 }

 @Override
 public int hashCode() {
  return Objects.hash(first, second);
 }

 @Override
 public boolean equals(Object obj) {
  if (this == obj)
   return true;
  if (obj == null || getClass() != obj.getClass())
   return false;
  KeyPair other = (KeyPair) obj;
  return Objects.equals(first, other.first) && Objects.equals(second,
other.second);
 }
}

/*
 * Get all the term types for the offer
 */
public static Set<String> getOfferTermTypes(String offerId) {

 Set<String> offerTermTypes = new HashSet<String>();

 DescribeEntityRequest request =
   DescribeEntityRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityId(offerId)
   .build();

 DescribeEntityResponse result = marketplaceCatalogClient.describeEntity(request);
```

```
 String details = result.details();

 try {
  ObjectMapper objectMapper = new ObjectMapper();
  JsonNode rootNode = objectMapper.readTree(details);
  JsonNode termsNode = rootNode.get(ATTRIBUTE_TERMS);

  for (JsonNode termNode : termsNode) {
   if (termNode.get(ATTRIBUTE_TYPE_ENTITY) != null ) {
    offerTermTypes.add(termNode.get(ATTRIBUTE_TYPE_ENTITY).asText());
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }

 return offerTermTypes;

}

/*
 * Get all the agreement term types
 */
public static Set<String> getAgreementTermTypes(GetAgreementTermsResponse
agreementTerm) {
 Set<String> agreementTermTypes = new HashSet<String>();
 try {
  for (AcceptedTerm term : agreementTerm.acceptedTerms()) {
   ObjectMapper objectMapper = new ObjectMapper();
   JsonNode termNode = objectMapper.readTree(getJson(term));
   Iterator<Map.Entry<String, JsonNode>> fieldsIterator = termNode.fields();
   while (fieldsIterator.hasNext()) {
    Map.Entry<String, JsonNode> entry = fieldsIterator.next();
    JsonNode value = entry.getValue();
    if (value.isObject() && value.has(ATTRIBUTE_TYPE_AGREEMENT)) {
     agreementTermTypes.add(value.get(ATTRIBUTE_TYPE_AGREEMENT).asText());
    }
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }
 return agreementTermTypes;
```

```
}

/*
 * make sure all elements in array2 exist in array1
 */
public static boolean allElementsExist(Set<String> array1, Set<String> array2) {
 for (String element : array2) {
  boolean found = false;
  for (String str : array1) {
   if (element.equals(str)) {
    found = true;
    break;
   }
  }
  if (!found) {
   return false;
  }
 }
 return true;
}

/*
 * Find the combinations of the agreement term types for the agreement
 */
public static Set<String> getMatchedTermTypesCombination(Set<String>
agreementTermTypes) {
 Set<String> matchedCombination = new HashSet<String>();
 for (Set<String> element : ALL_AGREEMENT_TERM_TYPES_COMBINATION) {
  if (allElementsExist(agreementTermTypes, element)) {
   matchedCombination = element;
  }
 }
 return matchedCombination;
}

public static void main(String[] args) {

 List<AgreementViewSummary> agreements = getAgreementsById();

 for (AgreementViewSummary summary : agreements) {
  String pricingType = "";
  String agreementId = summary.agreementId();
  System.out.println(agreementId);
  String offerId = summary.proposalSummary().offerId();
```

```
  //get all pricing term types for the offer in the agreement
  Set<String> offerTermTypes = getOfferTermTypes(offerId);
  String productType = summary.proposalSummary().resources().get(0).type();

  //get all pricing term types for the agreement
  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();
  GetAgreementTermsResponse getAgreementTermsResponse =
marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  Set<String> agreementTermTypes =
getAgreementTermTypes(getAgreementTermsResponse);

  //get matched pricing term type combination set
  Set<String> agreementMatchedTermType =
getMatchedTermTypesCombination(agreementTermTypes);

  //check to see if this agreement pricing term combination needs additional check
on offer pricing terms
  String needToCheckOfferType = needToCheckOfferTermsType(productType,
agreementMatchedTermType);

  // get the pricing type for the agreement based on the product type, agreement
term types and offer term types if needed
  if (needToCheckOfferType != null) {
   Set<String> offerMatchedTermType =
getMatchedTermTypesCombination(offerTermTypes);
   pricingType = getPricingType(productType, agreementMatchedTermType,
offerMatchedTermType);
  } else if (agreementMatchedTermType == LEGAL) {
   pricingType = FREE;
  } else {
   pricingType = getPricingType(productType, agreementMatchedTermType, new
HashSet());
  }
  System.out.println("Pricing type is " + pricingType);
 }
}

private static String getJson(Object result) {
 String json = "";

 try {
```

```
    ObjectMapper om = new ObjectMapper();
    om.setVisibility(PropertyAccessor.FIELD, Visibility.ANY);
    om.registerModule(new JavaTimeModule());
    ObjectWriter ow = om.writer().withDefaultPrettyPrinter();

    json = ow.writeValueAsString(result);
   } catch (JsonProcessingException e) {
    e.printStackTrace();
   }
   return json;
  }

}
```

- Para obter detalhes da API, consulte DescribeAgreement em Referência AWS SDK for Java 2.x da API.

Obter o tipo de produto de um contrato

O exemplo de código a seguir mostra como obter o tipo de produto de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
```

```
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementProductType {

 /*
  * Obtain the Product Type of the product the agreement was created on
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> productIds = getProducts(agreementId);

  ReferenceCodesUtils.formatOutput(productIds);
 }

 public static List<String> getProducts(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  List<String> productIds = new ArrayList<String>();
  for (Resource resource : describeAgreementResponse.proposalSummary().resources())
 {
   productIds.add(resource.id() + ":" + resource.type());
  }
  return productIds;
```

```
 }
}
```

- Para obter detalhes da API, consulte DescribeAgreement em Referência AWS SDK for Java 2.x da API.

Obter o status de um contrato

O exemplo de código a seguir mostra como obter o status de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementStatus {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);
```

```
  System.out.println("Agreement status is " + describeAgreementResponse.status());

 }

 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Para obter detalhes da API, consulte DescribeAgreement em Referência AWS SDK for Java 2.x da API.

Obter os termos de suporte de um contrato

O exemplo de código a seguir mostra como obter os termos de suporte de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.SupportTerm;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsSupportTerm {

 /*
  * Obtain the support and refund policy I have provided to the customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<SupportTerm> supportTerms = getSupportTerms(agreementId);

  ReferenceCodesUtils.formatOutput(supportTerms);
 }

 public static List<SupportTerm> getSupportTerms(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();
```

```
  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<SupportTerm> supportTerms = new ArrayList<>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.supportTerm() != null) {
    supportTerms.add(acceptedTerm.supportTerm());
   }
  }
  return supportTerms;
 }

}
```

- Para obter detalhes da API, consulte GetAgreementTerms em Referência AWS SDK for Java 2.x da API.

Obter os termos de um contrato

O exemplo de código a seguir mostra como obter os termos de um contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementTerms {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  GetAgreementTermsResponse getAgreementTermsResponse =
 getAgreementTermsResponse(agreementId);

  ReferenceCodesUtils.formatOutput(getAgreementTermsResponse);

 }

 public static GetAgreementTermsResponse getAgreementTermsResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  return getAgreementTermsResponse;
 }

}
```

- Para obter detalhes da API, consulte GetAgreementTerms em Referência AWS SDK for Java 2.x da API.

Listar solicitações de cancelamento de contratos

O exemplo de código a seguir mostra como listar solicitações de cancelamento de contratos dos quais participo como aceitante.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementCancellationRequestSumm
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementCancellationRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementCancellationRequest

public class ListAgreementCancellationRequests {

    private static final String PARTY_TYPE = "Proposer";

    public static void main(String[] args) {
        listAgreementCancellationRequests();
    }

    private static void listAgreementCancellationRequests() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();
```

```
        String nextToken = null;

        do {
            ListAgreementCancellationRequestsRequest request =
                    ListAgreementCancellationRequestsRequest.builder()
                            .partyType(PARTY_TYPE)
                            .nextToken(nextToken)
                            .build();

            ListAgreementCancellationRequestsResponse response =

 marketplaceAgreementClient.listAgreementCancellationRequests(request);

            for (AgreementCancellationRequestSummary summary : response.items()) {
                System.out.println("Cancellation Request ID: " +
 summary.agreementCancellationRequestId());
                System.out.println("Agreement ID: " + summary.agreementId());
                System.out.println("Status: " + summary.statusAsString());
                System.out.println("Reason Code: " + summary.reasonCodeAsString());
                System.out.println("Agreement Type: " + summary.agreementType());
                System.out.println("Catalog: " + summary.catalog());
                System.out.println("Created At: " + summary.createdAt());
                System.out.println("Updated At: " + summary.updatedAt());
                System.out.println("---");
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    }
}
```

- Para obter detalhes da API, consulte ListAgreementCancellationRequests em Referência AWS SDK for Java 2.x da API.

Listar solicitações de pagamento de contratos

O exemplo de código a seguir mostra como listar solicitações de pagamento de contratos dos quais participo como aceitante.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementPaymentRequestsRequ
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementPaymentRequestsResp
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentRequestSummary;

public class ListAgreementPaymentRequests {

    private static final String PARTY_TYPE = "Proposer";

    public static void main(String[] args) {
        listAgreementPaymentRequests();
    }

    private static void listAgreementPaymentRequests() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        String nextToken = null;

        do {
            ListAgreementPaymentRequestsRequest request =
                    ListAgreementPaymentRequestsRequest.builder()
```

```
                            .partyType(PARTY_TYPE)
                            .nextToken(nextToken)
                            .build();

            ListAgreementPaymentRequestsResponse response =

 marketplaceAgreementClient.listAgreementPaymentRequests(request);

            for (PaymentRequestSummary summary : response.items()) {
                System.out.println("Payment Request ID: " +
 summary.paymentRequestId());
                System.out.println("Agreement ID: " + summary.agreementId());
                System.out.println("Status: " + summary.statusAsString());
                System.out.println("Name: " + summary.name());
                System.out.println("Charge ID: " + summary.chargeId());
                System.out.println("Charge Amount: " + summary.chargeAmount());
                System.out.println("Currency Code: " + summary.currencyCode());
                System.out.println("Created At: " + summary.createdAt());
                System.out.println("Updated At: " + summary.updatedAt());
                System.out.println("---");
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    }
}
```

- Para obter detalhes da API, consulte ListAgreementPaymentRequests em Referência AWS SDK for Java 2.x da API.

Rejeitar uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como rejeitar uma solicitação de cancelamento de contrato iniciada pelo vendedor.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementCancellationReque
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementCancellationReque

public class RejectAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";
    private static final String REJECTION_REASON = "<REJECTION REASON HERE>";

    public static void main(String[] args) {
        rejectAgreementCancellationRequest();
    }

    private static void rejectAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RejectAgreementCancellationRequestRequest request =
                RejectAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
```

```
 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .rejectionReason(REJECTION_REASON)
                        .build();

        RejectAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.rejectAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Para obter detalhes da API, consulte RejectAgreementCancellationRequest em Referência AWS SDK for Java 2.x da API.

Rejeitar uma solicitação de pagamento de contrato

O exemplo de código a seguir mostra como rejeitar uma solicitação de pagamento de contrato iniciada pelo vendedor.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementPaymentRequestReq
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementPaymentRequestRes

public class RejectAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
    private static final String REJECTION_REASON = "<REJECTION REASON HERE>";

    public static void main(String[] args) {
        rejectAgreementPaymentRequest();
    }

    private static void rejectAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RejectAgreementPaymentRequestRequest request =
                RejectAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .rejectionReason(REJECTION_REASON)
                        .build();

        RejectAgreementPaymentRequestResponse response =
                marketplaceAgreementClient.rejectAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Name: " + response.name());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
```

```
    }
}
```

- Para obter detalhes da API, consulte RejectAgreementPaymentRequest em Referência AWS SDK for Java 2.x da API.

Substitua um contrato de SaaS por uma oferta baseada em contrato

O exemplo de código a seguir mostra como substituir um contrato de SaaS por uma nova oferta baseada em contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
```

```
 * Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
 * Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
 * Agreement-Based Offer from the seller. The buyer replaces the existing agreement
 * with the new ABO offer, which may include updated pricing terms and payment
 schedule.
 *
 * <p>Note: Unlike other Replace samples, this example starts from an already-active
 * {@code EXISTING_AGREEMENT_ID} rather than creating a new agreement first. Use
 this
 * pattern when you already have an agreement ID and want to replace it directly.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code EXISTING_AGREEMENT_ID} — the agreement ID of the active agreement to
 replace (starts with {@code agmt-}).</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new ABO offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the new offer's term
 list.</li>
 * </ul>
 */
public class ReplaceSaaSContractWithAgreementBasedOffer {

    // The agreement ID of the active SaaS agreement with CONTRACT pricing model to
 replace (starts with "agmt-").
    private static final String EXISTING_AGREEMENT_ID = "<your-existing-agreement-
id>";

    // The agreementProposalId from the new Agreement-Based Offer.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the LegalTerm in the new offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";
```

```
    // Term ID for the FixedUpfrontPricingTerm in the new offer.
    private static final String FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-
upfront-pricing-term-id>";

    // Term ID for the PaymentScheduleTerm in the new offer.
    private static final String PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-
term-id>";

    public static void main(String[] args) {
        replaceExistingAgreement();
    }

    /**
     * Replaces an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.
     * Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
     */
    private static void replaceExistingAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm validityTerm =
 RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm fixedUpfrontPricingTerm =
 RequestedTerm.builder().id(FIXED_UPFRONT_PRICING_TERM_ID).build();
        RequestedTerm paymentScheduleTerm =
 RequestedTerm.builder().id(PAYMENT_SCHEDULE_TERM_ID).build();

        // Replace the agreement with the new offer
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(fixedUpfrontPricingTerm,
 paymentScheduleTerm, legalTerm, validityTerm)

 .agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)
                        .sourceAgreementIdentifier(EXISTING_AGREEMENT_ID)
                        .build();
```

```
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Replace agreement request created. AgreementRequestId: "
 + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement replaced with ABO. New AgreementId: " +
 acceptAgreementRequestResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK for Java 2.x da API.

Substitua um teste gratuito de SaaS por um contrato com preços de consumo

O exemplo de código a seguir mostra como substituir um teste gratuito de SaaS por um Contrato com Preços de Consumo (CCP).

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS free trial agreement and then replace it with a
 * paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.
 *
 * <p>Scenario: A buyer first starts a free trial on a SaaS product. Once the trial
 is active,
 * they decide to convert by replacing it with a paid offer that includes a
 * FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.
 *
 * <p>Flow:
 * <ol>
 *   <li>Create a SaaS free trial agreement with freeTrialPricingTerm.</li>
 *   <li>Wait for freeTrialPricingTerm agreement entitlements to become active.</li>
 *   <li>Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 free trial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the paid CCP offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term list.</
li>
```

```
 * </ul>
 */
public class ReplaceSaaSFreeTrialWithCCP {

    // The agreementProposalId from the free trial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-
agreement-proposal-identifier>";

    // Term ID for the FreeTrialPricingTerm in the free trial offer.
    private static final String FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-
pricing-term-id>";

    // Term ID for the LegalTerm in the free trial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // The agreementProposalId from the paid CCP offer to convert to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-
agreement-proposal-identifier>";

    // Term ID for the FixedUpfrontPricingTerm in the CCP offer.
    private static final String FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-
upfront-pricing-term-id>";

    // Term ID for the PaymentScheduleTerm in the CCP offer.
    private static final String PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-
term-id>";

    // Term ID for the UsageBasedPricingTerm in the CCP offer.
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the ValidityTerm in the CCP offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the LegalTerm in the CCP offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    public static void main(String[] args) {
        createSaaSFreeTrialAndReplaceWithCCP();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS free trial agreement with freeTrialPricingTerm.
```

```
     * 2. Wait for freeTrialPricingTerm agreement entitlements to become active.
     * 3. Replace the free trial agreement with the paid CCP offer.
     */
    private static void createSaaSFreeTrialAndReplaceWithCCP() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm freeTrialPricingTerm =
RequestedTerm.builder().id(FREE_TRIAL_PRICING_TERM_ID).build();

        // --- Step 1: Agreement with freeTrialPricingTerm ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(freeTrialPricingTerm, legalTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with freeTrialPricingTerm created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with freeTrialPricingTerm accepted.
AgreementId: " + acceptAgreementRequestResponse.agreementId());

        // Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
        System.out.println("Waiting for freeTrialPricingTerm agreement entitlements
to become active...");
```

```
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("freeTrialPricingTerm agreement entitlements are now
active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
        // Use Intent.REPLACE and sourceAgreementIdentifier to replace the free
trial agreement.
        RequestedTerm usageBasedPricingTerm =
RequestedTerm.builder().id(USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm fixedUpfrontPricingTerm =
RequestedTerm.builder().id(FIXED_UPFRONT_PRICING_TERM_ID).build();
        RequestedTerm paymentScheduleTerm =
RequestedTerm.builder().id(PAYMENT_SCHEDULE_TERM_ID).build();
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm newLegalTerm =
RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(usageBasedPricingTerm,
fixedUpfrontPricingTerm, paymentScheduleTerm, validityTerm, newLegalTerm)

.agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)

.sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Replace agreement request created. AgreementRequestId: "
+ carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
```

```
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        System.out.println("Agreement with freeTrialPricingTerm replaced with paid
 CCP offer. New AgreementId: " + aarResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK for Java 2.x da API.

Substitua um termo de preço baseado no uso do SaaS e cancele o contrato anterior

O exemplo de código a seguir mostra como substituir um termo de preço baseado no uso do SaaS e cancelar o contrato anterior.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CancelAgreementRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT) and
 then replace it
 * with a new offer using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
 * The buyer then converts to a different offer by replacing the existing agreement.
 *
 * <p>Flow:
 * <ol>
 *   <li>Create and accept the initial agreement request with UBPT.</li>
 *   <li>Wait for entitlements to become active.</li>
 *   <li>Replace the agreement with a new offer using Intent.REPLACE.</li>
 *   <li>Cancel the new agreement using CancelAgreement.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 initial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new offer to replace to.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term list.</
li>
 * </ul>
 */
public class ReplaceSaaSUsageBasedPricingTermAndCancel {

    // The agreementProposalId from the initial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the initial offer.
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";
```

```
    // Term ID for the ValidityTerm in the initial offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the LegalTerm in the initial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // The agreementProposalId from the new offer to replace to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the new offer.
    private static final String NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-
based-pricing-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String NEW_VALIDITY_TERM_ID = "<your-new-validity-term-
id>";

    // Term ID for the LegalTerm in the new offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    public static void main(String[] args) {
        replaceSaaSUbptAndCancel();
    }

    /**
     * Full end-to-end flow:
     * 1. Create and accept the initial agreement request with
 UsageBasedPricingTerm.
     * 2. Wait for entitlements to become active.
     * 3. Replace the agreement with a new offer.
     * 4. Cancel the new agreement.
     */
    private static void replaceSaaSUbptAndCancel() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageBasedPricingTerm =
 RequestedTerm.builder().id(USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm validityTerm =
 RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
```

```
        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();

        // --- Step 1: Create and accept the initial UBPT agreement request ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageBasedPricingTerm, validityTerm,
legalTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created. AgreementRequestId:
" + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with UBPT accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before replacing.
        System.out.println("Waiting for entitlements to become active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("Entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Replace the UBPT agreement with a new offer ---
        // Use Intent.REPLACE and sourceAgreementIdentifier to replace the existing
agreement.
        RequestedTerm newUsageBasedPricingTerm =
RequestedTerm.builder().id(NEW_USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm newValidityTerm =
RequestedTerm.builder().id(NEW_VALIDITY_TERM_ID).build();
```

```
        RequestedTerm newLegalTerm =
 RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(newUsageBasedPricingTerm, newValidityTerm,
 newLegalTerm)

 .agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)

 .sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Replace agreement request created. AgreementRequestId: "
 + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String agreementIdWithTheNewOffer = aarResponse.agreementId();
        System.out.println("UBPT agreement replaced. New AgreementId: " +
 agreementIdWithTheNewOffer);

        // --- Step 3: Cancel the new agreement ---
        CancelAgreementRequest cancelAgreementRequest =
                CancelAgreementRequest.builder()
                        .agreementId(agreementIdWithTheNewOffer)
                        .build();
        marketplaceAgreementClient.cancelAgreement(cancelAgreementRequest);
        System.out.println("The new agreement has been cancelled. AgreementId: " +
 agreementIdWithTheNewOffer);
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK for Java 2.x da API.

Substitua um termo de preço baseado no uso da AMI, mas mantenha o ConfigurableUpfrontPricingTerm

O exemplo de código a seguir mostra como substituir um termo de preço baseado no uso da AMI e, ao mesmo tempo, manter o existente. ConfigurableUpfrontPricingTerm

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
```

```
 * Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
 * the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active, using
 the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: An AMI product with USAGE pricing requires two agreements:
 * <ol>
 *   <li>An agreement with <b>usageBasedPricingTerm</b> — accepted first to
 establish the base agreement.</li>
 *   <li>An agreement with <b>configurableUpfrontPricingTerm (CUPT)</b> — accepted
 after the usageBasedPricingTerm agreement entitlements are active.</li>
 * </ol>
 * <p> This sample shows how to replace only the agreement with
 usageBasedPricingTerm with a
 * new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
 *
 * <p>Flow:
 * <ol>
 *   <li>Create and accept the initial agreement request with
 usageBasedPricingTerm.</li>
 *   <li>Create and accept the agreement request with configurableUpfrontPricingTerm
 (CUPT) (after usageBasedPricingTerm agreement entitlements are active).</li>
 *   <li>Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 initial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new offer to replace to.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code
 P365D}).</li>
 *   <li>{@code DIMENSION_1_KEY} — dimension key defined in the CUPT term.</li>
 * </ul>
 */
public class ReplaceAmiUsageBasedPricingTermButNotCupt {
```

```
    // The agreementProposalId from the initial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P365D" for 365 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // Dimension key defined in the CUPT term.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the UsageBasedPricingTerm in the initial offer.
    private static final String USAGE_TERM_ID = "<your-usage-term-id>";

    // Term ID for the LegalTerm in the initial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in the initial offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // The agreementProposalId from the new offer to replace to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the new offer.
    private static final String NEW_USAGE_TERM_ID = "<your-new-usage-term-id>";

    // Term ID for the LegalTerm in the new offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String NEW_VALIDITY_TERM_ID = "<your-new-validity-term-
id>";

    public static void main(String[] args) {
        replaceAmiUsageWhenCuptExists();
    }
```

```
   /**
    * Full end-to-end flow:
    * 1. Create and accept an agreement request with usageBasedPricingTerm, then
wait for entitlements.
    * 2. Create and accept an agreement request with CUPT, then wait for
entitlements.
    * 3. Replace the agreement with usageBasedPricingTerm with a new offer
(agreement with CUPT is unaffected).
    */
   private static void replaceAmiUsageWhenCuptExists() {
       MarketplaceAgreementClient marketplaceAgreementClient =
               MarketplaceAgreementClient.builder()
                       .httpClient(ApacheHttpClient.builder().build())
                       .credentialsProvider(ProfileCredentialsProvider.create())
                       .build();

       RequestedTerm usageTerm = RequestedTerm.builder().id(USAGE_TERM_ID).build();
       RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
       RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();

       // --- Step 1: Agreement with UBPT ---
       CreateAgreementRequestRequest createAgreementRequestRequest =
               CreateAgreementRequestRequest.builder()
                       .clientToken(AgreementApiUtils.generateClientToken())
                       .intent(Intent.NEW)
                       .requestedTerms(usageTerm, legalTerm, validityTerm)
                       .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                       .build();
       CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
       System.out.println("Agreement request with UBPT created. AgreementRequestId:
" + createAgreementRequestResponse.agreementRequestId());

       AcceptAgreementRequestRequest acceptAgreementRequestRequest =
               AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                       .build();
       AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
```

```
        final String usageAgreementId =
acceptAgreementRequestResponse.agreementId();
        System.out.println("Agreement request with UBPT accepted. AgreementId: " +
usageAgreementId);

        // Wait for UBPT agreement entitlements to become active before creating the
agreement with CUPT.
        System.out.println("Waiting for UBPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, usageAgreementId);
        System.out.println("UBPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, legalTerm,
validityTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Agreement request with CUPT created. AgreementRequestId:
" + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
```

```
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String cuptAgreementId = aarResponse.agreementId();
        System.out.println("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

        // Wait for CUPT agreement entitlements to become active before replacing
usage.
        System.out.println("Waiting for CUPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse cuptEntitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, cuptAgreementId);
        System.out.println("CUPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(cuptEntitlementsResponse);

        // --- Step 3: Replace Agreement with usageBasedPricingTerm with a new offer
---
        RequestedTerm newUsageTerm =
RequestedTerm.builder().id(NEW_USAGE_TERM_ID).build();
        RequestedTerm newLegalTerm =
RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();
        RequestedTerm newValidityTerm =
RequestedTerm.builder().id(NEW_VALIDITY_TERM_ID).build();

        // Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
usageBasedPricingTerm agreement only.
        // The agreement with configurableUpfrontPricingTerm (CUPT) is NOT affected
by this replacement.
        CreateAgreementRequestRequest carReplaceRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(newUsageTerm, newLegalTerm, newValidityTerm)

.agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)
                        .sourceAgreementIdentifier(usageAgreementId)
                        .build();
        CreateAgreementRequestResponse carReplaceResponse =

marketplaceAgreementClient.createAgreementRequest(carReplaceRequest);
```

```
        System.out.println("Replace agreement request created. AgreementRequestId: "
 + carReplaceResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarReplaceRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carReplaceResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarReplaceResponse =

 marketplaceAgreementClient.acceptAgreementRequest(aarReplaceRequest);
        System.out.println("Agreement with UBPT replaced. New AgreementId: " +
 aarReplaceResponse.agreementId());
    }
}
```

- Para obter detalhes da API, consulte CreateAgreementRequest em Referência AWS SDK for Java 2.x da API.

Pesquisar contratos por data de término

O exemplo de código a seguir mostra como pesquisar contratos por data de término.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class SearchAgreementsByEndDate {

 static String beforeOrAfterEndtimeFilterName =
 BeforeOrAfterEndTimeFilterName.BeforeEndTime.name();

 static String cutoffDate = "2050-11-18T00:00:00Z";

 static String partyTypeFilterValue = PARTY_TYPE_FILTER_VALUE_PROPOSER;

 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
 where you are the proposer.
  // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
 the acceptor.
  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();
```

```
 Filter customizeFilter =
Filter.builder().name(beforeOrAfterEndtimeFilterName).values(cutoffDate).build();

 List<Filter> filters = new ArrayList<Filter>();

 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter));

 // search agreement with filters

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementResponse=
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());

 while (searchAgreementResponse.nextToken() != null &&
searchAgreementResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementResponse.nextToken())
    .build();
  searchAgreementResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Para obter detalhes da API, consulte SearchAgreements em Referência AWS SDK for Java 2.x da API.

Pesquisar contratos com um filtro personalizado

O exemplo de código a seguir mostra como pesquisar contratos com um filtro personalizado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * All filter combinations we support for Proposer and Acceptor:
```

```
 * https://docs.aws.amazon.com/marketplace/latest/APIReference/API_marketplace-
agreements_SearchAgreements.html
 */

public class SearchAgreementsByOneFilter {

 private static final String FILTER_NAME = "ResourceType";

 private static final String FILTER_VALUE = "SaaSProduct";

 /*
  * search agreements by one customize filter
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
 where you are the proposer.
  // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
 the acceptor.
  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  Filter customizeFilter =
 Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

  List<Filter> filters = new ArrayList<Filter>();

  filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
 customizeFilter));
```

```
 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();
 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Para obter detalhes da API, consulte SearchAgreements em Referência AWS SDK for Java 2.x da API.

Pesquisar contratos com dois filtros personalizados

O exemplo de código a seguir mostra como pesquisar contratos com dois filtros personalizados.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * All filter combinations we support for Proposer and Acceptor:
 * https://docs.aws.amazon.com/marketplace/latest/APIReference/API_marketplace-
agreements_SearchAgreements.html
 */

public class SearchAgreementsByTwoFilters {

 public static final String FILTER_1_NAME = "ResourceType";
```

```
public static final String FILTER_1_VALUE = "SaaSProduct";

public static final String FILTER_2_NAME = "Status";

public static final String FILTER_2_VALUE = "ACTIVE";

/*
 * search agreements by two customize filter
 */
public static void main(String[] args) {

 List<AgreementViewSummary> agreementSummaryList = getAgreements();

 ReferenceCodesUtils.formatOutput(agreementSummaryList);

}

public static List<AgreementViewSummary> getAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
the acceptor.
 Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 Filter customizeFilter1 =
Filter.builder().name(FILTER_1_NAME).values(FILTER_1_VALUE).build();

 Filter customizeFilter2 =
Filter.builder().name(FILTER_2_NAME).values(FILTER_2_VALUE).build();


 List<Filter> filters = new ArrayList<Filter>();
```

```
 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter1, customizeFilter2));

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Para obter detalhes da API, consulte SearchAgreements em Referência AWS SDK for Java 2.x da API.

Atualizar pedidos de compra após a aceitação do contrato

O exemplo de código a seguir mostra como atualizar pedidos de compra após a aceitação do contrato.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Charge;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementChargesRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementChargesResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.PurchaseOrder;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import
 software.amazon.awssdk.services.marketplaceagreement.model.UpdatePurchaseOrdersRequest;
import utils.AgreementApiUtils;

/**
```

```
 * Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT pricing model
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model
 and provides a purchase order
 * reference in AcceptAgreementRequest. After acceptance, the buyer lists the
 resulting
 * charges via ListAgreementCharges and associates the purchase order reference with
 a
 * specific charge via UpdatePurchaseOrders.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement.</li>
 *   <li>{@code DIMENSION_1_KEY} — dimension key defined in the offer.</li>
 *   <li>{@code PURCHASE_ORDER_REFERENCE} — your internal purchase order number
 (e.g., {@code po-123456}).</li>
 * </ul>
 */
public class UpdatePurchaseOrdersAfterAgreementAcceptance {

    // Your internal purchase order reference number (e.g., "po-123456").
    private static final String PURCHASE_ORDER_REFERENCE = "po-123456";

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P366D" for 366 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // Dimension key and quantity defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";
```

```
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in your offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    public static void main(String[] args) {
        listAgreementChargesAndUpdatePurchaseOrders();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS agreement with CONTRACT pricing model with a purchase order
reference.
     * 2. List charges to retrieve charge IDs and revisions.
     * 3. Associate the purchase order reference with a specific charge via
UpdatePurchaseOrders.
     * 4. List charges again to confirm the update.
     */
    private static void listAgreementChargesAndUpdatePurchaseOrders() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build())
                                .build()))
                .build();

        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
```

```
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();

        // --- Create Agreement ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, legalTerm,
validityTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        // --- Accept Agreement Request with Purchase Order ---
        // The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
        String chargeId =
createAgreementRequestResponse.chargeSummary().expectedCharges().get(0).id();
        PurchaseOrder purchaseOrderAtAcceptance = PurchaseOrder.builder()
                .chargeId(chargeId)
                .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                .build();
        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .purchaseOrders(purchaseOrderAtAcceptance)
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        final String agreementId = acceptAgreementRequestResponse.agreementId();
        System.out.println("Agreement request accepted with purchase order reference
'" + PURCHASE_ORDER_REFERENCE
                + "'. AgreementId: " + agreementId);

        // --- List Agreement Charges ---
        ListAgreementChargesRequest listAgreementChargesRequest =
                ListAgreementChargesRequest.builder()
```

```
                        .agreementId(agreementId)
                        .build();
        ListAgreementChargesResponse listAgreementChargesResponse =

 marketplaceAgreementClient.listAgreementCharges(listAgreementChargesRequest);

        System.out.println("All charges for agreement " + agreementId + ":");
        AgreementApiUtils.formatOutput(listAgreementChargesResponse);

        // --- Update Purchase Order ---
        Charge firstCharge = listAgreementChargesResponse.items().get(0);
        PurchaseOrder purchaseOrder = PurchaseOrder.builder()
                .agreementId(agreementId)
                .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                .chargeRevision(firstCharge.revision())
                .chargeId(firstCharge.id())
                .build();
        UpdatePurchaseOrdersRequest updatePurchaseOrdersRequest =
                UpdatePurchaseOrdersRequest.builder()
                        .purchaseOrders(purchaseOrder)
                        .build();

 marketplaceAgreementClient.updatePurchaseOrders(updatePurchaseOrdersRequest);
        System.out.println("Purchase order reference '" + PURCHASE_ORDER_REFERENCE
                + "' updated for ChargeId: " + firstCharge.id());

        // --- Verify Update ---
        ListAgreementChargesRequest lacRequest =
                ListAgreementChargesRequest.builder()
                        .agreementId(agreementId)
                        .build();
        ListAgreementChargesResponse lacResponse =
                marketplaceAgreementClient.listAgreementCharges(lacRequest);
        System.out.println("Verified updated charge:");
        AgreementApiUtils.formatOutput(lacResponse.items().get(0));
    }
}
```

- Para obter detalhes da API, consulte UpdatePurchaseOrders em Referência AWS SDK for Java 2.x da API.

MediaConvert exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with MediaConvert.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
```

```
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
 software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
```

```
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
 software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
 software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
```

```
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class CreateJob {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                    <mcRoleARN> <fileInput>\s

                                Where:
                                    mcRoleARN - The MediaConvert Role ARN.\s
                                    fileInput -  The URL of an Amazon S3 bucket
 where the input file is located.\s
                                """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String mcRoleARN = args[0];
                String fileInput = args[1];
                Region region = Region.US_WEST_2;
                MediaConvertClient mc = MediaConvertClient.builder()
                                .region(region)
                                .build();

                String id = createMediaJob(mc, mcRoleARN, fileInput);
```

```
                System.out.println("MediaConvert job created. Job Id = " + id);
                mc.close();
        }

        public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
 String fileInput) {

                String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
 1) + "javasdk/out/";
                String fileOutput = s3path + "index";
                String thumbsOutput = s3path + "thumbs/";
                String mp4Output = s3path + "mp4/";

                try {
                        System.out.println("MediaConvert role arn: " + mcRoleARN);
                        System.out.println("MediaConvert input file: " + fileInput);
                        System.out.println("MediaConvert output path: " + s3path);

                        // output group Preset HLS low profile
                        Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
 750000, 7, 1920, 1080, 640);
                        // output group Preset HLS media profile
                        Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
                        // output group Preset HLS high profole
                        Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
 3500000, 8, 1920, 1080, 1920);

                        OutputGroup appleHLS = OutputGroup.builder().name("Apple
 HLS").customName("Example")

 .outputGroupSettings(OutputGroupSettings.builder()

 .type(OutputGroupType.HLS_GROUP_SETTINGS)

 .hlsGroupSettings(HlsGroupSettings.builder()

 .directoryStructure(

    HlsDirectoryStructure.SINGLE_DIRECTORY)

 .manifestDurationFormat(

    HlsManifestDurationFormat.INTEGER)
```

```
.streamInfResolution(
   HlsStreamInfResolution.INCLUDE)
.clientCache(HlsClientCache.ENABLED)
.captionLanguageSetting(
   HlsCaptionLanguageSetting.OMIT)
.manifestCompression(
   HlsManifestCompression.NONE)
.codecSpecification(
   HlsCodecSpecification.RFC_4281)
.outputSelection(
   HlsOutputSelection.MANIFESTS_AND_SEGMENTS)
.programDateTime(HlsProgramDateTime.EXCLUDE)
.programDateTimePeriod(600)
.timedMetadataId3Frame(
   HlsTimedMetadataId3Frame.PRIV)
.timedMetadataId3Period(10)
.destination(fileOutput)
.segmentControl(HlsSegmentControl.SEGMENTED_FILES)
.minFinalSegmentLength((double) 0)
.segmentLength(4).minSegmentLength(0).build())
                                                        .build())
                                        .outputs(hlsLow, hlsMedium,
hlsHigh).build();
```

```
                        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(mp4Output).build())
                                                        .build())
                                        .outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.MP4).build())

.videoDescription(VideoDescription.builder().width(1280)
                                                                        .height(720)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.H_264)

   .h264Settings(H264Settings

                    .builder()
```

```
                .rateControlMode(
                                H264RateControlMode.QVBR)
                .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                .qualityTuningLevel(
                                H264QualityTuningLevel.SINGLE_PASS)
                .qvbrSettings(
                                H264QvbrSettings.builder()
                                                .qvbrQualityLevel(
                                                                8)
                                                .build())
                .codecLevel(H264CodecLevel.AUTO)
                .codecProfile(H264CodecProfile.MAIN)
                .maxBitrate(2400000)
                .framerateControl(
                                H264FramerateControl.INITIALIZE_FROM_SOURCE)
                .gopSize(2.0)
                .gopSizeUnits(H264GopSizeUnits.SECONDS)
                .numberBFramesBetweenReferenceFrames(
                                2)
                .gopClosedCadence(
                                1)
                .gopBReference(H264GopBReference.DISABLED)
```

```
                .slowPal(H264SlowPal.DISABLED)
                .syntax(H264Syntax.DEFAULT)
                .numberReferenceFrames(
                        3)
                .dynamicSubGop(H264DynamicSubGop.STATIC)
                .fieldEncoding(H264FieldEncoding.PAFF)
                .sceneChangeDetect(
                        H264SceneChangeDetect.ENABLED)
                .minIInterval(0)
                .telecine(H264Telecine.NONE)
                .framerateConversionAlgorithm(
                        H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                .entropyEncoding(
                        H264EntropyEncoding.CABAC)
                .slices(1)
                .unregisteredSeiTimecode(
                        H264UnregisteredSeiTimecode.DISABLED)
                .repeatPps(H264RepeatPps.DISABLED)
                .adaptiveQuantization(
                        H264AdaptiveQuantization.HIGH)
                .spatialAdaptiveQuantization(
                        H264SpatialAdaptiveQuantization.ENABLED)
```

```
                    .temporalAdaptiveQuantization(
                                    H264TemporalAdaptiveQuantization.ENABLED)
                    .flickerAdaptiveQuantization(
                                    H264FlickerAdaptiveQuantization.DISABLED)
                    .softness(0)
                    .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                    .build())
   .build())
                                                                    .build())
.audioDescriptions(AudioDescription.builder()
.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
.languageCodeControl(
   AudioLanguageCodeControl.FOLLOW_INPUT)
.codecSettings(AudioCodecSettings.builder()
   .codec(AudioCodec.AAC)
   .aacSettings(AacSettings
                    .builder()
                    .codecProfile(AacCodecProfile.LC)
                    .rateControlMode(
                                    AacRateControlMode.CBR)
                    .codingMode(AacCodingMode.CODING_MODE_2_0)
                    .sampleRate(44100)
```

```
                    .bitrate(160000)

                    .rawFormat(AacRawFormat.NONE)

                    .specification(AacSpecification.MPEG4)

                    .audioDescriptionBroadcasterMix(

                                    AacAudioDescriptionBroadcasterMix.NORMAL)

                    .build())

   .build())
                                                                        .build())
                                                        .build())
                                        .build();
                        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(thumbsOutput).build())
                                                        .build())
                                        .outputs(Output.builder().extension("jpg")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder()

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)
```

```
.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.FRAME_CAPTURE)

   .frameCaptureSettings(

                    FrameCaptureSettings

                                    .builder()

                                    .framerateNumerator(

                                                    1)

                                    .framerateDenominator(

                                                    1)

                                    .maxCaptures(10000000)

                                    .quality(80)

                                    .build())

   .build())
                                                                                .build())
                                                                .build())
                                        .build();

                       Map<String, AudioSelector> audioSelectors = new HashMap<>();
                       audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
                                                                .offset(0).build());

                       JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                        .audioSelectors(audioSelectors)
                                        .videoSelector(
```

```
VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

.rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0)
                                        .deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                                        .outputGroups(appleHLS, thumbs,
fileMp4).build();

                        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
                                        .settings(jobSettings)
                                        .build();

                        CreateJobResponse createJobResponse =
mc.createJob(createJobRequest);
                        return createJobResponse.job().id();

                } catch (MediaConvertException e) {
                        System.out.println(e.toString());
                        System.exit(0);
                }
                return "";
        }

        private final static Output createOutput(String customName,
                        String nameModifier,
                        String segmentModifier,
                        int qvbrMaxBitrate,
                        int qvbrQualityLevel,
                        int originWidth,
                        int originHeight,
                        int targetWidth) {

                int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
                Output output = null;
```

```
                try {
                        output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                                        .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,

   489, 490, 491, 492)
                                                                                .build())
                                                                .build())
                                        .videoDescription(

VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)
```

```
.timecodeInsertion(
   VideoTimecodeInsertion.DISABLED)
.colorMetadata(ColorMetadata.INSERT)
.respondToAfd(RespondToAfd.NONE)
.afdSignaling(AfdSignaling.NONE)
.dropFrameTimecode(DropFrameTimecode.ENABLED)
.codecSettings(VideoCodecSettings.builder()
   .codec(VideoCodec.H_264)
   .h264Settings(H264Settings
                   .builder()
                   .rateControlMode(
                                   H264RateControlMode.QVBR)
                   .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                   .qualityTuningLevel(
                                   H264QualityTuningLevel.SINGLE_PASS)
                   .qvbrSettings(H264QvbrSettings
                                   .builder()
                                   .qvbrQualityLevel(
                                                   qvbrQualityLevel)
                                   .build())
                   .codecLevel(H264CodecLevel.AUTO)
                   .codecProfile((targetHeight > 720
```

```
                                && targetWidth > 1280)
                                                ? H264CodecProfile.HIGH
                                                : H264CodecProfile.MAIN)
                        .maxBitrate(qvbrMaxBitrate)
                        .framerateControl(
                                H264FramerateControl.INITIALIZE_FROM_SOURCE)
                        .gopSize(2.0)
                        .gopSizeUnits(H264GopSizeUnits.SECONDS)
                        .numberBFramesBetweenReferenceFrames(
                                2)
                        .gopClosedCadence(
                                1)
                        .gopBReference(H264GopBReference.DISABLED)
                        .slowPal(H264SlowPal.DISABLED)
                        .syntax(H264Syntax.DEFAULT)
                        .numberReferenceFrames(
                                3)
                        .dynamicSubGop(H264DynamicSubGop.STATIC)
                        .fieldEncoding(H264FieldEncoding.PAFF)
                        .sceneChangeDetect(
                                H264SceneChangeDetect.ENABLED)
                        .minIInterval(0)
```

```
                    .telecine(H264Telecine.NONE)
                    .framerateConversionAlgorithm(
                                    H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                    .entropyEncoding(
                                    H264EntropyEncoding.CABAC)
                    .slices(1)
                    .unregisteredSeiTimecode(
                                    H264UnregisteredSeiTimecode.DISABLED)
                    .repeatPps(H264RepeatPps.DISABLED)
                    .adaptiveQuantization(
                                    H264AdaptiveQuantization.HIGH)
                    .spatialAdaptiveQuantization(
                                    H264SpatialAdaptiveQuantization.ENABLED)
                    .temporalAdaptiveQuantization(
                                    H264TemporalAdaptiveQuantization.ENABLED)
                    .flickerAdaptiveQuantization(
                                    H264FlickerAdaptiveQuantization.DISABLED)
                    .softness(0)
                    .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                    .build())
    .build())
                                                                    .build())
```

```
 .audioDescriptions(AudioDescription.builder()

 .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

 .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

 .codecSettings(AudioCodecSettings.builder()

 .codec(AudioCodec.AAC).aacSettings(AacSettings

    .builder()

    .codecProfile(AacCodecProfile.LC)

    .rateControlMode(

                    AacRateControlMode.CBR)

    .codingMode(AacCodingMode.CODING_MODE_2_0)

    .sampleRate(44100)

    .bitrate(96000)

    .rawFormat(AacRawFormat.NONE)

    .specification(AacSpecification.MPEG4)

    .audioDescriptionBroadcasterMix(

                    AacAudioDescriptionBroadcasterMix.NORMAL)

    .build())
                                                                    .build())
                                                        .build())
                                        .build();
                } catch (MediaConvertException e) {
                        e.printStackTrace();
                        System.exit(0);
                }
                return output;
        }
}
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK for Java 2.x da API.

GetJob

O código de exemplo a seguir mostra como usar `GetJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
                "  <jobId> \n\n" +
                "Where:\n" +
                "  jobId - The job id value.\n\n";
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
            GetJobRequest jobRequest = GetJobRequest.builder()
                    .id(jobId)
                    .build();

            GetJobResponse response = mc.getJob(jobRequest);
            System.out.println("The ARN of the job is " + response.job().arn());

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- Para obter detalhes da API, consulte GetJob em Referência AWS SDK for Java 2.x da API.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            // Create the ListJobsRequest
            ListJobsRequest jobsRequest = ListJobsRequest.builder()
```

```
                    .maxResults(10)
                    .status("COMPLETE")
                    .build();

            // Call the listJobs operation
            ListJobsResponse jobsResponse = mc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK for Java 2.x da API.

Exemplos do Migration Hub usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Migration Hub.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteProgressUpdateStream`

O código de exemplo a seguir mostra como usar `DeleteProgressUpdateStream`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <progressStream>\s

                Where:
                    progressStream - the name of a progress stream to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
```

```
                .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    public static void deleteStream(MigrationHubClient migrationClient, String
 streamName) {
        try {
            DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
 DeleteProgressUpdateStreamRequest
                    .builder()
                    .progressUpdateStreamName(streamName)
                    .build();


 migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
            System.out.println(streamName + " is deleted");

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteProgressUpdateStream em Referência AWS SDK for Java 2.x da API.

`DescribeApplicationState`

O código de exemplo a seguir mostra como usar `DescribeApplicationState`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeAppState <appId>\s

                Where:
                    appId -  the application id value.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeApplicationState(migrationClient, appId);
        migrationClient.close();
    }
```

```
    public static void describeApplicationState(MigrationHubClient migrationClient,
 String appId) {
        try {
            DescribeApplicationStateRequest applicationStateRequest =
 DescribeApplicationStateRequest.builder()
                    .applicationId(appId)
                    .build();

            DescribeApplicationStateResponse applicationStateResponse =
 migrationClient
                    .describeApplicationState(applicationStateRequest);
            System.out.println("The application status is " +
 applicationStateResponse.applicationStatusAsString());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeApplicationState em Referência AWS SDK for Java 2.x da API.

`DescribeMigrationTask`

O código de exemplo a seguir mostra como usar `DescribeMigrationTask`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
```

```
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeMigrationTask <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeMigTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void describeMigTask(MigrationHubClient migrationClient, String
 migrationTask,
            String progressStream) {
```

```
        try {
            DescribeMigrationTaskRequest migrationTaskRequestRequest =
 DescribeMigrationTaskRequest.builder()
                    .progressUpdateStream(progressStream)
                    .migrationTaskName(migrationTask)
                    .build();

            DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
                    .describeMigrationTask(migrationTaskRequestRequest);
            System.out.println("The name is " +
 migrationTaskResponse.migrationTask().migrationTaskName());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeMigrationTask em Referência AWS SDK for Java 2.x da API.

`ImportMigrationTask`

O código de exemplo a seguir mostra como usar `ImportMigrationTask`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
```

```
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        importMigrTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void importMigrTask(MigrationHubClient migrationClient, String
 migrationTask, String progressStream) {
        try {
            CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
 CreateProgressUpdateStreamRequest.builder()
                    .progressUpdateStreamName(progressStream)
```

```
                    .dryRun(false)
                    .build();

            migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
            ImportMigrationTaskRequest migrationTaskRequest =
 ImportMigrationTaskRequest.builder()
                    .migrationTaskName(migrationTask)
                    .progressUpdateStream(progressStream)
                    .dryRun(false)
                    .build();

            migrationClient.importMigrationTask(migrationTaskRequest);

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ImportMigrationTask em Referência AWS SDK for Java 2.x da API.

`ListApplications`

O código de exemplo a seguir mostra como usar `ListApplications`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
```

```
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listApps(migrationClient);
        migrationClient.close();
    }

    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
 ListApplicationStatesRequest.builder()
                    .maxResults(10)
                    .build();

            ListApplicationStatesResponse response =
 migrationClient.listApplicationStates(applicationStatesRequest);
            List<ApplicationState> apps = response.applicationStateList();
            for (ApplicationState appState : apps) {
                System.out.println("App Id is " + appState.applicationId());
                System.out.println("The status is " +
 appState.applicationStatus().toString());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte ListApplications em Referência AWS SDK for Java 2.x da API.

`ListCreatedArtifacts`

O código de exemplo a seguir mostra como usar `ListCreatedArtifacts`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
```

```
                .region(region)
                .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
 ListCreatedArtifactsRequest.builder()
                    .maxResults(10)
                    .migrationTaskName("SampleApp5")
                    .progressUpdateStream("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
 migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("APp Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListCreatedArtifacts em Referência AWS SDK for Java 2.x da API.

`ListMigrationTasks`

O código de exemplo a seguir mostra como usar `ListMigrationTasks`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listMigrTasks(migrationClient);
        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
 ListMigrationTasksRequest.builder()
                    .maxResults(10)
                    .build();
```

```
            ListMigrationTasksResponse response =
 migrationClient.listMigrationTasks(listMigrationTasksRequest);
            List<MigrationTaskSummary> migrationList =
 response.migrationTaskSummaryList();
            for (MigrationTaskSummary migration : migrationList) {
                System.out.println("Migration task name is " +
 migration.migrationTaskName());
                System.out.println("The Progress update stream is " +
 migration.progressUpdateStream());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListMigrationTasks em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon MSK usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon MSK.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

Exemplos do Neptune usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Neptune.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Neptune

O exemplo de código a seguir mostra como começar a usar o Neptune.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloNeptune {
    public static void main(String[] args) {
        NeptuneAsyncClient neptuneClient = NeptuneAsyncClient.create();
        describeDbCluster(neptuneClient).join(); // This ensures the async code runs
 to completion
    }

    /**
     * Describes the Amazon Neptune DB clusters.
     *
     * @param neptuneClient the Neptune asynchronous client used to make the request
     * @return a {@link CompletableFuture} that completes when the operation is
 finished
     */
    public static CompletableFuture<Void> describeDbCluster(NeptuneAsyncClient
 neptuneClient) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .maxRecords(20)
                .build();

        SdkPublisher<DescribeDbClustersResponse> paginator =
 neptuneClient.describeDBClustersPaginator(request);
        CompletableFuture<Void> future = new CompletableFuture<>();

        paginator.subscribe(new Subscriber<DescribeDbClustersResponse>() {
            private Subscription subscription;

            @Override
            public void onSubscribe(Subscription s) {
                this.subscription = s;
                s.request(Long.MAX_VALUE); // request all items
            }

            @Override
            public void onNext(DescribeDbClustersResponse response) {
                response.dbClusters().forEach(cluster -> {
```

```
                    System.out.println("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                    System.out.println("Status: " + cluster.status());
                });
            }

            @Override
            public void onError(Throwable t) {
                future.completeExceptionally(t);
            }

            @Override
            public void onComplete() {
                future.complete(null);
            }
        });

        return future.whenComplete((result, throwable) -> {
            neptuneClient.close();
            if (throwable != null) {
                System.err.println("Error describing DB clusters: " +
throwable.getMessage());
            }
        });
    }
```

- Para obter detalhes da API, consulte DescribeDBClustersPaginator em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de sub-redes do Amazon Neptune.
- Criar um cluster do Neptune.
- Criar uma instância do Neptune.
- Conferir o status da instância do Neptune.
- Mostrar detalhes do cluster do Neptune.

- Interromper o cluster do Neptune.
- Iniciar o cluster do Neptune.
- Excluir os ativos do Neptune.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos do Neptune.

```
public class NeptuneScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneScenario.class);
    static Scanner scanner = new Scanner(System.in);
    static NeptuneActions neptuneActions = new NeptuneActions();

    public static void main(String[] args) {
        final String usage =
                """
                Usage:
                    <subnetGroupName> <clusterName> <dbInstanceId>

                Where:
                    subnetGroupName - The name of an existing Neptune DB subnet
 group that includes subnets in at least two Availability Zones.
                    clusterName     - The unique identifier for the Neptune DB
 cluster.
                    dbInstanceId    - The identifier for a specific Neptune DB
 instance within the cluster.
                """;
        String subnetGroupName = "neptuneSubnetGroup65";
        String clusterName = "neptuneCluster65";
        String dbInstanceId = "neptuneDB65";

        logger.info("""
                   Amazon Neptune is a fully managed graph
```

```
                    database service by AWS, designed specifically
                    for handling complex relationships and connected
                    datasets at scale. It supports two popular graph models:
                    property graphs (via openCypher and Gremlin) and RDF
                    graphs (via SPARQL). This makes Neptune ideal for
                    use cases such as knowledge graphs, fraud detection,
                    social networking, recommendation engines, and
                    network management, where relationships between
                    entities are central to the data.

                    Being fully managed, Neptune handles database
                    provisioning, patching, backups, and replication,
                    while also offering high availability and durability
                    within AWS's infrastructure.

                    For developers, programming with Neptune allows
                    for building intelligent, relationship-aware
                    applications that go beyond traditional tabular
                    databases. Developers can use the AWS SDK for Java
                    to automate infrastructure operations (via NeptuneClient).

                     Let's get started...
                     """);
        waitForInputToContinue(scanner);
        runScenario(subnetGroupName, dbInstanceId, clusterName);
    }

    public static void runScenario(String subnetGroupName, String dbInstanceId,
 String clusterName) {
        logger.info(DASHES);
        logger.info("1. Create a Neptune DB Subnet Group");
        logger.info("The Neptune DB subnet group is used when launching a Neptune
 cluster");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.createSubnetGroupAsync(subnetGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
 cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
```

```
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create a Neptune Cluster");
        logger.info("A Neptune Cluster allows you to store and query highly
connected datasets with low latency.");
        waitForInputToContinue(scanner);
        String dbClusterId;
        try {
            dbClusterId = neptuneActions.createDBClusterAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a Neptune DB Instance");
        logger.info("In this step, we add a new database instance to the Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
        neptuneActions.createDBInstanceAsync(dbInstanceId, dbClusterId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Check the status of the Neptune DB Instance");
        logger.info("""
                    In this step, we will wait until the DB instance
                    becomes available. This may take around 10 minutes.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5.Show Neptune Cluster details");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.describeDBClustersAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Stop the Amazon Neptune cluster");
        logger.info("""
                    Once stopped, this step polls the status
                    until the cluster is in a stopped state.
```

```
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.stopDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "stopped");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Start the Amazon Neptune cluster");
        logger.info("""
                    Once started, this step polls the clusters
                    status until it's in an available state.
                    We will also poll the instance status.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.startDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "available");
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the Neptune Assets");
```

```
        logger.info("Would you like to delete the Neptune Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the Neptune assets.");
            try {
                neptuneActions.deleteNeptuneResourcesAsync(dbInstanceId,
 clusterName, subnetGroupName);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The request failed due to the resource not found:
 {}", cause.getMessage());
                } else {
                    logger.error("An unexpected error occurred.", cause);
                }
                return;
            }
        } else {
            logger.info("You selected not to delete Neptune assets.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(
                """
                Thank you for checking out the Amazon Neptune Service Use demo. We
 hope you
                learned something new, or got some inspiration for your own apps
 today.
                For more AWS code examples, have a look at:
                https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                """);
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
```

```
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper para métodos de SDK do Neptune.

```
public class NeptuneActions {
    private CompletableFuture<Void> instanceCheckFuture;
    private static NeptuneAsyncClient neptuneAsyncClient;
    private final Region region = Region.US_EAST_1;
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneActions.class);
    private final NeptuneClient neptuneClient =
 NeptuneClient.builder().region(region).build();

    /**
     * Retrieves an instance of the NeptuneAsyncClient.
     * <p>
     * This method initializes and returns a singleton instance of the
 NeptuneAsyncClient. The client
     * is configured with the following settings:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
     * The client is built using the NettyNioAsyncHttpClient.
     *
     * @return the singleton instance of the NeptuneAsyncClient
     */
    private static NeptuneAsyncClient getAsyncClient() {
        if (neptuneAsyncClient == null) {
```

```
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            neptuneAsyncClient = NeptuneAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return neptuneAsyncClient;
    }

    /**
     * Asynchronously deletes a set of Amazon Neptune resources in a defined order.
     * <p>
     * The method performs the following operations in sequence:
     * <ol>
     *     <li>Deletes the Neptune DB instance identified by {@code dbInstanceId}.</
li>
     *     <li>Waits until the DB instance is fully deleted.</li>
     *     <li>Deletes the Neptune DB cluster identified by {@code dbClusterId}.</
li>
     *     <li>Deletes the Neptune DB subnet group identified by {@code
 subnetGroupName}.</li>
     * </ol>
     * <p>
     * If any step fails, the subsequent operations are not performed, and the
 exception
     * is logged. This method blocks the calling thread until all operations
 complete.
     *
     * @param dbInstanceId      the ID of the Neptune DB instance to delete
     * @param dbClusterId       the ID of the Neptune DB cluster to delete
     * @param subnetGroupName   the name of the Neptune DB subnet group to delete
```

```
    */
   public void deleteNeptuneResourcesAsync(String dbInstanceId, String dbClusterId,
String subnetGroupName) {
       deleteDBInstanceAsync(dbInstanceId)
               .thenCompose(v -> waitUntilInstanceDeletedAsync(dbInstanceId))
               .thenCompose(v -> deleteDBClusterAsync(dbClusterId))
               .thenCompose(v -> deleteDBSubnetGroupAsync(subnetGroupName))
               .whenComplete((v, ex) -> {
                   if (ex != null) {
                       logger.info("Failed to delete Neptune resources: " +
ex.getMessage());
                   } else {
                       logger.info("Neptune resources deleted successfully.");
                   }
               })
               .join(); // Waits for the entire async chain to complete
   }

   /**
    * Deletes a subnet group.
    *
    * @param subnetGroupName the identifier of the subnet group to delete
    * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
    */
   public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
       DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(subnetGroupName)
               .build();

       return getAsyncClient().deleteDBSubnetGroup(request)
               .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
   }

   /**
    * Deletes a DB instance asynchronously.
    *
    * @param clusterId the identifier of the cluster to delete
    * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
    */
   public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
```

```
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
    }

    public CompletableFuture<Void> waitUntilInstanceDeletedAsync(String instanceId)
{
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkInstanceDeletedRecursive(instanceId, startTime, future);
        return future;
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
    }


    private void checkInstanceDeletedRecursive(String instanceId, long startTime,
CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
```

```
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof NeptuneException &&
                                ((NeptuneException)
cause).awsErrorDetails().errorCode().equals("DBInstanceNotFound")) {
                            long elapsed = (System.currentTimeMillis() -
startTime) / 1000;
                            logger.info("\r Instance %s deleted after %ds%n",
instanceId, elapsed);
                            future.complete(null);
                            return;
                        }
                        future.completeExceptionally(new CompletionException("Error
polling DB instance", cause));
                        return;
                    }

                    String status =
response.dbInstances().get(0).dbInstanceStatus();
                    long elapsed = (System.currentTimeMillis() - startTime) / 1000;
                    System.out.printf("\r  Waiting: Instance %s status: %-10s (%ds
elapsed)", instanceId, status, elapsed);
                    System.out.flush();

                    CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                            .execute(() -> checkInstanceDeletedRecursive(instanceId,
startTime, future));
                });
    }


    public void waitForClusterStatus(String clusterId, String desiredStatus) {
        System.out.printf("Waiting for cluster '%s' to reach status '%s'...\n",
clusterId, desiredStatus);
        CompletableFuture<Void> future = new CompletableFuture<>();
        checkClusterStatusRecursive(clusterId, desiredStatus,
System.currentTimeMillis(), future);
        future.join();
    }

    private void checkClusterStatusRecursive(String clusterId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
```

```
                .dbClusterIdentifier(clusterId)
                .build();

        getAsyncClient().describeDBClusters(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
cluster status", cause)
                        );
                        return;
                    }

                    List<DBCluster> clusters = response.dbClusters();
                    if (clusters.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Cluster
not found: " + clusterId));
                        return;
                    }

                    String currentStatus = clusters.get(0).status();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Cluster status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune cluster reached desired status
'%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() ->
checkClusterStatusRecursive(clusterId, desiredStatus, startTime, future));
                    }
                });
    }


    /**
     * Starts an Amazon Neptune DB cluster.
     *
```

```
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
       StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().startDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }

                       throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                   } else {
                       logger.info("DB Cluster starting: " + clusterIdentifier);
                   }
               });
   }

   /**
    * Stops an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
       StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().stopDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;
```

```
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }



    /**
     * Asynchronously describes the specified Amazon RDS DB cluster.
     *
     * @param clusterId the identifier of the DB cluster to describe
     * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
     * if an error occurs
     */
    public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        return getAsyncClient().describeDBClusters(request)
                .thenAccept(response -> {
                    for (DBCluster cluster : response.dbClusters()) {
                        logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                        logger.info("Status: " + cluster.status());
                        logger.info("Engine: " + cluster.engine());
                        logger.info("Engine Version: " + cluster.engineVersion());
                        logger.info("Endpoint: " + cluster.endpoint());
                        logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                        logger.info("Availability Zones: " +
cluster.availabilityZones());
                        logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                        logger.info("VPC Security Groups:");
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
```

```
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }


    public CompletableFuture<Void> checkInstanceStatus(String instanceId, String
desiredStatus) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkStatusRecursive(instanceId, desiredStatus.toLowerCase(), startTime,
future);
        return future;
    }

    /**
     * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
```

```
    * @param startTime      the start time of the operation, used to calculate the
elapsed time
    * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
    */
   private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       future.completeExceptionally(
                               new CompletionException("Error checking Neptune
instance status", cause)
                       );
                       return;
                   }

                   List<DBInstance> instances = response.dbInstances();
                   if (instances.isEmpty()) {
                       future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                       return;
                   }

                   String currentStatus = instances.get(0).dbInstanceStatus();
                   long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                   System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                   System.out.flush();

                   if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                       System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                       future.complete(null);
                   } else {
                       CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                               .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
```

```
                    }
                });
    }


    private String formatElapsedTime(int seconds) {
        int minutes = seconds / 60;
        int remainingSeconds = seconds % 60;

        if (minutes > 0) {
            return minutes + (minutes == 1 ? " min" : " mins") + ", " +
                    remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        } else {
            return remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        }
    }


    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
     *
     * @param dbInstanceId the identifier for the new DB instance
     * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
     * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
     * @throws CompletionException if the operation fails, with a cause of either:
     *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
     *                             - a general exception with the failure message
     */
    public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
        CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceId)
                .dbInstanceClass("db.r5.large")
                .engine("neptune")
                .dbClusterIdentifier(dbClusterId)
                .build();

        return getAsyncClient().createDBInstance(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
```

```
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String instanceId =
response.dbInstance().dbInstanceIdentifier();
                    logger.info("Created Neptune DB Instance: " + instanceId);
                    return instanceId;
                });
    }


    /**
     * Creates a new Amazon Neptune DB cluster asynchronously.
     *
     * @param dbName the name of the DB cluster to be created
     * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
     * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createDBClusterAsync(String dbName) {
        CreateDbClusterRequest request = CreateDbClusterRequest.builder()
                .dbClusterIdentifier(dbName)
                .engine("neptune")
                .deletionProtection(false)
                .backupRetentionPeriod(1)
                .build();

        return getAsyncClient().createDBCluster(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
```

```
                    }
                })
                .thenApply(response -> {
                    String clusterId = response.dbCluster().dbClusterIdentifier();
                    logger.info("DB Cluster created: " + clusterId);
                    return clusterId;
                });
    }


    /**
     * Creates a new DB subnet group asynchronously.
     *
     * @param groupName the name of the subnet group to create
     * @return a CompletableFuture that, when completed, returns the Amazon Resource
 Name (ARN) of the created subnet group
     * @throws CompletionException if the operation fails, with a cause that may be
 a ServiceQuotaExceededException if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

        // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
 resources will be created
        String vpcId = getDefaultVpcId();
        logger.info("VPC is : " + vpcId);

        List<String> subnetList = getSubnetIds(vpcId);
        for (String subnetId : subnetList) {
            System.out.println("Subnet group:" +subnetId);
        }

        CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(groupName)
                .dbSubnetGroupDescription("Subnet group for Neptune cluster")
                .subnetIds(subnetList)
                .build();

        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
 because the request would exceed the maximum quota.", cause);
```

```
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }

    private List<String> getSubnetIds(String vpcId) {
        try (Ec2Client ec2 = Ec2Client.builder().region(region).build()) {
            DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                    .filters(builder -> builder.name("vpc-id").values(vpcId))
                    .build();

            DescribeSubnetsResponse response = ec2.describeSubnets(request);
            return response.subnets().stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());
        }
    }

    public static String getDefaultVpcId() {
        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_EAST_1)
                .build();

        Filter myFilter = Filter.builder()
                .name("isDefault")
                .values("true")
                .build();

        List<Filter> filterList = new ArrayList<>();
        filterList.add(myFilter);

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .filters(filterList)
                .build();
```

```
        DescribeVpcsResponse response = ec2.describeVpcs(request);
        if (!response.vpcs().isEmpty()) {
            Vpc defaultVpc = response.vpcs().get(0);
            return defaultVpc.vpcId();
        } else {
            throw new RuntimeException("No default VPC found in this region.");
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDBCluster
  - CreateDBInstance
  - CreateDBSubnetGroup
  - CreateGraph
  - DeleteDBCluster
  - DeleteDBInstance
  - DeleteDBSubnetGroup
  - DescribeDBClusters
  - DescribeDBInstances
  - ExecuteGremlinProfileQuery
  - ExecuteGremlinQuery
  - ExecuteOpenCypherExplainQuery
  - ExecuteQuery
  - Iniciar cluster de banco de dados
  - Interrompa o cluster de banco

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String clusterId = response.dbCluster().dbClusterIdentifier();
                   logger.info("DB Cluster created: " + clusterId);
                   return clusterId;
```

```
                });
    }
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK for Java 2.x .

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
     *
     * @param dbInstanceId the identifier for the new DB instance
     * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
     * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
     * @throws CompletionException if the operation fails, with a cause of either:
     *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
     *                             - a general exception with the failure message
     */
    public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
        CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceId)
                .dbInstanceClass("db.r5.large")
                .engine("neptune")
                .dbClusterIdentifier(dbClusterId)
                .build();

        return getAsyncClient().createDBInstance(request)
```

```
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String instanceId =
response.dbInstance().dbInstanceIdentifier();
                    logger.info("Created Neptune DB Instance: " + instanceId);
                    return instanceId;
                });
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK for Java 2.x .

`CreateDBSubnetGroup`

O código de exemplo a seguir mostra como usar `CreateDBSubnetGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates a new DB subnet group asynchronously.
     *
     * @param groupName the name of the subnet group to create
     * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
```

```
    * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

       // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
       String vpcId = getDefaultVpcId();
       logger.info("VPC is : " + vpcId);

       List<String> subnetList = getSubnetIds(vpcId);
       for (String subnetId : subnetList) {
           System.out.println("Subnet group:" +subnetId);
       }

       CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(groupName)
               .dbSubnetGroupDescription("Subnet group for Neptune cluster")
               .subnetIds(subnetList)
               .build();

       return getAsyncClient().createDBSubnetGroup(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String name = response.dbSubnetGroup().dbSubnetGroupName();
                   String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                   logger.info("Subnet group created: " + name);
                   return arn;
               });
   }
```

- Para obter detalhes da API, consulte CreateDBSubnetGroup em Referência AWS SDK for Java 2.x da API.

`CreateGraph`

O código de exemplo a seguir mostra como usar `CreateGraph`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Executes the process of creating a new Neptune graph.
    *
    * @param client        the Neptune graph client used to interact with the
Neptune service
    * @param graphName     the name of the graph to be created
    * @throws NeptuneGraphException if an error occurs while creating the graph
    */
   public static void executeCreateGraph(NeptuneGraphClient client, String
graphName) {
       try {
           // Create the graph request
           CreateGraphRequest request = CreateGraphRequest.builder()
                   .graphName(graphName)
                   .provisionedMemory(16)
                   .build();

           // Create the graph
           CreateGraphResponse response = client.createGraph(request);

           // Extract the graph name and ARN
           String createdGraphName = response.name();
           String graphArn = response.arn();
           String graphEndpoint = response.endpoint();

           System.out.println("Graph created successfully!");
           System.out.println("Graph Name: " + createdGraphName);
```

```
            System.out.println("Graph ARN: " + graphArn);
            System.out.println("Graph Endpoint: " +graphEndpoint );

        } catch (NeptuneGraphException e) {
            System.err.println("Failed to create graph: " +
e.awsErrorDetails().errorMessage());
        } finally {
            client.close();
        }
    }
```

- Para obter detalhes da API, consulte CreateGraph em Referência AWS SDK for Java 2.x da API.

`DeleteDBCluster`

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
```

```
                .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
    }
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK for Java 2.x .

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
    }
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK for Java 2.x .

`DeleteDBSubnetGroup`

O código de exemplo a seguir mostra como usar `DeleteDBSubnetGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
    }
```

- Para obter detalhes da API, consulte DeleteDBSubnetGroup em Referência AWS SDK for Java 2.x da API.

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously describes the specified Amazon RDS DB cluster.
    *
    * @param clusterId the identifier of the DB cluster to describe
    * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
    * if an error occurs
    */
   public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .dbClusterIdentifier(clusterId)
               .build();

       return getAsyncClient().describeDBClusters(request)
               .thenAccept(response -> {
                   for (DBCluster cluster : response.dbClusters()) {
                       logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                       logger.info("Status: " + cluster.status());
                       logger.info("Engine: " + cluster.engine());
                       logger.info("Engine Version: " + cluster.engineVersion());
                       logger.info("Endpoint: " + cluster.endpoint());
                       logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                       logger.info("Availability Zones: " +
cluster.availabilityZones());
                       logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                       logger.info("VPC Security Groups:");
                       cluster.vpcSecurityGroups().forEach(vpcGroup ->
                               logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                       logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
```

```
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK for Java 2.x .

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
    *
    * @param instanceId     the ID of the Neptune instance to check
    * @param desiredStatus  the desired status of the Neptune instance
    * @param startTime      the start time of the operation, used to calculate the
elapsed time
    * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
    */
   private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       future.completeExceptionally(
                               new CompletionException("Error checking Neptune
instance status", cause)
                       );
                       return;
                   }

                   List<DBInstance> instances = response.dbInstances();
                   if (instances.isEmpty()) {
                       future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                       return;
                   }

                   String currentStatus = instances.get(0).dbInstanceStatus();
                   long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                   System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                   System.out.flush();

                   if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                       System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
```

```
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK for Java 2.x .

`ExecuteGremlinProfileQuery`

O código de exemplo a seguir mostra como usar `ExecuteGremlinProfileQuery`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Executes a Gremlin query against an Amazon Neptune database using the
provided {@link NeptunedataClient}.
     *
     * @param client the {@link NeptunedataClient} instance to use for executing the
Gremlin query
     */
    public static void executeGremlinQuery(NeptunedataClient client) {
        try {
            System.out.println("Querying Neptune...");
            ExecuteGremlinQueryRequest request =
ExecuteGremlinQueryRequest.builder()
                    .gremlinQuery("g.V().has('code', 'ANC')")
                    .build();
```

```
            ExecuteGremlinQueryResponse response =
client.executeGremlinQuery(request);

            System.out.println("Full Response:");
            System.out.println(response);

            // Retrieve and print the result
            if (response.result() != null) {
                System.out.println("Query Result:");
                System.out.println(response.result().toString());
            } else {
                System.out.println("No result returned from the query.");
            }
        } catch (NeptunedataException e) {
            System.err.println("Error calling Neptune: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Para obter detalhes da API, consulte ExecuteGremlinProfileQuery em Referência AWS SDK for Java 2.x da API.

`ExecuteGremlinQuery`

O código de exemplo a seguir mostra como usar `ExecuteGremlinQuery`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Executes a Gremlin PROFILE query using the provided NeptunedataClient.
```

```
     *
     * @param client The NeptunedataClient instance to be used for executing the
Gremlin PROFILE query.
     */
    private static void executeGremlinProfileQuery(NeptunedataClient client) {
        System.out.println("Executing Gremlin PROFILE query...");

        ExecuteGremlinProfileQueryRequest request =
ExecuteGremlinProfileQueryRequest.builder()
                .gremlinQuery("g.V().has('code', 'ANC')")
                .build();

        ExecuteGremlinProfileQueryResponse response =
client.executeGremlinProfileQuery(request);
        if (response.output() != null) {
            System.out.println("Query Profile Output:");
            System.out.println(response.output());
        } else {
            System.out.println("No output returned from the profile query.");
        }
    }
```

- Para obter detalhes da API, consulte ExecuteGremlinQuery em Referência AWS SDK for Java 2.x da API.

`ExecuteOpenCypherExplainQuery`

O código de exemplo a seguir mostra como usar `ExecuteOpenCypherExplainQuery`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Executes an OpenCypher EXPLAIN query using the provided Neptune data client.
     *
```

```
     * @param client The Neptune data client to use for the query execution.
     */
    public static void executeGremlinQuery(NeptunedataClient client) {
        try {
            System.out.println("Executing OpenCypher EXPLAIN query...");
            ExecuteOpenCypherExplainQueryRequest request =
ExecuteOpenCypherExplainQueryRequest.builder()
                    .openCypherQuery("MATCH (n {code: 'ANC'}) RETURN n")
                    .explainMode("debug")
                    .build();

            ExecuteOpenCypherExplainQueryResponse response =
client.executeOpenCypherExplainQuery(request);

            if (response.results() != null) {
                System.out.println("Explain Results:");
                System.out.println(response.results().asUtf8String());
            } else {
                System.out.println("No explain results returned.");
            }

        } catch (NeptunedataException e) {
            System.err.println("Neptune error: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Para obter detalhes da API, consulte ExecuteOpenCypherExplainQuery em Referência AWS SDK for Java 2.x da API.

`ExecuteQuery`

O código de exemplo a seguir mostra como usar `ExecuteQuery`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Executes a Gremlin profile query on the Neptune Analytics graph.
    *
    * @param client       the {@link NeptuneGraphClient} instance to use for the
query
    * @param graphId      the identifier of the graph to execute the query on
    *
    * @throws NeptuneGraphException if an error occurs while executing the query on
the Neptune Graph
    * @throws Exception if an unexpected error occurs
    */
   public static void executeGremlinProfileQuery(NeptuneGraphClient client, String
graphId) {

       try {
           System.out.println("Running openCypher query on Neptune Analytics...");

           ExecuteQueryRequest request = ExecuteQueryRequest.builder()
                   .graphIdentifier(graphId)
                   .queryString("MATCH (n {code: 'ANC'}) RETURN n")
                   .language("OPEN_CYPHER")
                   .build();

           ResponseInputStream<ExecuteQueryResponse> response =
client.executeQuery(request);
           try (BufferedReader reader = new BufferedReader(new
InputStreamReader(response, StandardCharsets.UTF_8))) {
               String result = reader.lines().collect(Collectors.joining("\n"));
               System.out.println("Query Result:");
               System.out.println(result);
           } catch (Exception e) {
               System.err.println("Error reading response: " + e.getMessage());
           }
```

```
        } catch (NeptuneGraphException e) {
            System.err.println("NeptuneGraph error: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Para obter detalhes da API, consulte ExecuteQuery em Referência AWS SDK for Java 2.x da API.

StartDBCluster

O código de exemplo a seguir mostra como usar `StartDBCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Starts an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
        StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().startDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
```

```
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }
```

- Consulte detalhes da API em StartDBCluster na Referência de API do AWS SDK for Java 2.x .

StopDBCluster

O código de exemplo a seguir mostra como usar `StopDBCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
        StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();
```

```
        return getAsyncClient().stopDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }
```

- Consulte os detalhes da API em StopDBCluster na Referência de API do AWS SDK for Java 2.x .

Cenários

Use a API do Neptune para consultar dados de gráficos

O exemplo de código a seguir mostra como usar a API do Neptune para consultar dados de gráficos.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Neptune para criar uma função do Lambda que consulta dados de gráficos na VPC.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Lambda
- Neptune

Exemplos da Central de Parceiros usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Partner Central.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AssignOpportunity`

O código de exemplo a seguir mostra como usar `AssignOpportunity`.

SDK para Java 2.x

Reatribua uma oportunidade existente a outro usuário.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.AssigneeContact;

/*
Purpose
PC-API-07 Assigning a new owner
*/

public class AssignOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String assigneeFirstName = "John";

     String assigneeLastName = "Doe";

     String assigneeEmail = "test@test.com";

     String businessTitle = "PartnerAccountManager";

     AssignOpportunityResponse response = getResponse(opportunityId,
 assigneeFirstName, assigneeLastName, assigneeEmail, businessTitle);

     ReferenceCodesUtils.formatOutput(response);
    }

 static AssignOpportunityResponse getResponse(String opportunityId, String
 assigneeFirstName, String assigneeLastName, String assigneeEmail, String
 businessTitle) {

  AssignOpportunityRequest assignOpportunityRequest =
 AssignOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
```

```
            .identifier(opportunityId)
            .assignee(AssigneeContact.builder()
              .firstName(assigneeFirstName)
              .lastName(assigneeLastName)
              .email(assigneeEmail)
              .businessTitle(businessTitle)
              .build())
            .build();

        AssignOpportunityResponse response =
 client.assignOpportunity(assignOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte AssignOpportunity em Referência AWS SDK for Java 2.x da API.

`AssociateOpportunity`

O código de exemplo a seguir mostra como usar `AssociateOpportunity`.

SDK para Java 2.x

Crie uma associação formal entre uma oportunidade e várias entidades relacionadas.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;

/*
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class AssociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     AssociateOpportunityResponse response = getResponse(opportunityId, entityType,
 entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

static AssociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();
```

```
        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte AssociateOpportunity em Referência AWS SDK for Java 2.x da API.

`CreateOpportunity`

O código de exemplo a seguir mostra como usar `CreateOpportunity`.

SDK para Java 2.x

Crie uma oportunidade.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityResponse;
```

```
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.MonetaryValue;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.SoftwareRevenue;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

public class CreateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 public static void main(String[] args) {

  String inputFile = "CreateOpportunity2.json";

  if (args.length > 0)
   inputFile = args[0];

  CreateOpportunityResponse response = createOpportunity(inputFile);

  client.close();
 }

 static CreateOpportunityResponse createOpportunity(String inputFile) {

  String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

  Root root = GSON.fromJson(inputString, Root.class);
```

```
 List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
 if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
  for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
   NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
     .time(Instant.parse(nextStepsHistoryJson.time))
     .value(nextStepsHistoryJson.value)
                 .build();
   nextStepsHistories.add(nextStepsHistory);
  }
 }

 LifeCycle lifeCycle = null;
 if ( root.lifeCycle != null ) {
  lifeCycle = LifeCycle.builder()
   .closedLostReason(root.lifeCycle.closedLostReason)
   .nextSteps(root.lifeCycle.nextSteps)
   .nextStepsHistory(nextStepsHistories)
   .reviewComments(root.lifeCycle.reviewComments)
   .reviewStatus(root.lifeCycle.reviewStatus)
   .reviewStatusReason(root.lifeCycle.reviewStatusReason)
   .stage(root.lifeCycle.stage)
   .targetCloseDate(root.lifeCycle.targetCloseDate)
   .build();
 }

 Marketing marketing = null;
 if ( root.marketing != null ) {
  marketing = Marketing.builder()
    .awsFundingUsed(root.marketing.awsFundingUsed)
    .campaignName(root.marketing.campaignName)
    .channels(root.marketing.channels)
    .source(root.marketing.source)
    .useCases(root.marketing.useCases)
    .build();

 }

 Address address = null;
 if ( root.customer != null && root.customer.account != null &&
root.customer.account.address != null ) {
  address = Address.builder()
   .city(root.customer.account.address.city)
```

```
                .postalCode(root.customer.account.address.postalCode)
                .stateOrRegion(root.customer.account.address.stateOrRegion)
                .countryCode(root.customer.account.address.countryCode)
                .streetAddress(root.customer.account.address.streetAddress)
                .build();
}

Account account = null;
if ( root.customer != null && root.customer.account!= null) {
 account = Account.builder()
           .address(address)
           .awsAccountId(root.customer.account.awsAccountId)
              .duns(root.customer.account.duns)
              .industry(root.customer.account.industry)
              .otherIndustry(root.customer.account.otherIndustry)
              .companyName(root.customer.account.companyName)
              .websiteUrl(root.customer.account.websiteUrl)
              .build();
}

List<Contact> contacts = new ArrayList<Contact>();
if ( root.customer != null && root.customer.contacts != null) {
 for (org.example.entity.Contact jsonContact : root.customer.contacts) {
  Contact contact = Contact.builder()
                .email(jsonContact.email)
                .firstName(jsonContact.firstName)
                .lastName(jsonContact.lastName)
                .phone(jsonContact.phone)
                .businessTitle(jsonContact.businessTitle)
                .build();
  contacts.add(contact);
 }
}

Customer customer = Customer.builder()
  .account(account)
  .contacts(contacts)
  .build();

Contact oportunityTeamContact = null;
if (root.opportunityTeam != null && root.opportunityTeam.get(0) != null ) {
 oportunityTeamContact = Contact.builder()
              .firstName(root.opportunityTeam.get(0).firstName)
              .lastName(root.opportunityTeam.get(0).lastName)
```

```
                .email(root.opportunityTeam.get(0).email)
                .phone(root.opportunityTeam.get(0).phone)
                .businessTitle(root.opportunityTeam.get(0).businessTitle)
                .build();
 }

 List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
 if ( root.project != null && root.project.expectedCustomerSpend != null) {
  for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
   ExpectedCustomerSpend expectedCustomerSpend = null;
   expectedCustomerSpend = ExpectedCustomerSpend.builder()
     .amount(expectedCustomerSpendJson.amount)
     .currencyCode(expectedCustomerSpendJson.currencyCode)
     .frequency(expectedCustomerSpendJson.frequency)
     .targetCompany(expectedCustomerSpendJson.targetCompany)
     .build();
   expectedCustomerSpends.add(expectedCustomerSpend);
  }
      }

      Project project = null;
      if ( root.project != null) {
       project = Project.builder()
               .title(root.project.title)
               .customerBusinessProblem(root.project.customerBusinessProblem)
               .customerUseCase(root.project.customerUseCase)
               .deliveryModels(root.project.deliveryModels)
               .expectedCustomerSpend(expectedCustomerSpends)
               .salesActivities(root.project.salesActivities)
               .competitorName(root.project.competitorName)
               .otherSolutionDescription(root.project.otherSolutionDescription)
               .build();
      }

      SoftwareRevenue softwareRevenue = null;
      if ( root.softwareRevenue != null) {
       MonetaryValue monetaryValue = null;
       if ( root.softwareRevenue.value != null) {
        monetaryValue = MonetaryValue.builder()
          .amount(root.softwareRevenue.value.amount)
          .currencyCode(root.softwareRevenue.value.currencyCode)
          .build();
```

```
          }
          softwareRevenue = SoftwareRevenue.builder()
            .deliveryModel(root.softwareRevenue.deliveryModel)
            .effectiveDate(root.softwareRevenue.effectiveDate)
            .expirationDate(root.softwareRevenue.expirationDate)
            .value(monetaryValue)
            .build();
        }

  // Building the Actual CreateOpportunity Request
  CreateOpportunityRequest createOpportunityRequest =
 CreateOpportunityRequest.builder()
    .catalog(CATALOG_TO_USE)
    .clientToken(root.clientToken)
    .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws)
    .opportunityType(root.opportunityType)
    .lifeCycle(lifeCycle)
    .marketing(marketing)
    .nationalSecurity(root.nationalSecurity)
    .origin(root.origin)
    .customer(customer)
    .project(project)
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
    .opportunityTeam(oportunityTeamContact)
    .softwareRevenue(softwareRevenue)
    .build();

  CreateOpportunityResponse response =
 client.createOpportunity(createOpportunityRequest);
  System.out.println("Successfully created: " + response);

  return response;
    }

}
```

- Para obter detalhes da API, consulte CreateOpportunity em Referência AWS SDK for Java 2.x da API.

`DisassociateOpportunity`

O código de exemplo a seguir mostra como usar `DisassociateOpportunity`.

SDK para Java 2.x

Remova uma associação existente entre uma oportunidade e entidades relacionadas.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class DisassociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";
```

```
    DisassociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

    ReferenceCodesUtils.formatOutput(response);
   }

 static DisassociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

  DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte DisassociateOpportunity em Referência AWS SDK for Java 2.x da API.

`GetAwsOpportunitySummary`

O código de exemplo a seguir mostra como usar `GetAwsOpportunitySummary`.

SDK para Java 2.x

Recupera um resumo de uma AWS oportunidade.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRequest
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRespons

/*
 * Purpose
 * PC-API-25 Retrieves a summary of an AWS Opportunity.
 */

public class GetAwsOpportunitySummary {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetAwsOpportunitySummaryResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetAwsOpportunitySummaryResponse getResponse(String opportunityId) {

  GetAwsOpportunitySummaryRequest getOpportunityRequest =
 GetAwsOpportunitySummaryRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .relatedOpportunityIdentifier(opportunityId)
          .build();

  GetAwsOpportunitySummaryResponse response =
 client.getAwsOpportunitySummary(getOpportunityRequest);
```

```
        return response;
 }
}
```

- Para obter detalhes da API, consulte GetAwsOpportunitySummary em Referência AWS SDK for Java 2.x da API.

`GetEngagementInvitation`

O código de exemplo a seguir mostra como usar `GetEngagementInvitation`.

SDK para Java 2.x

Recupera os detalhes de um convite de noivado compartilhado AWS com um parceiro.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse

/*
 * Purpose
 * PC-API-22 Get engagement invitation opportunity
 */

public class GetEngagementInvitation {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
```

```
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static GetEngagementInvitationResponse getResponse(String opportunityId) {

  GetEngagementInvitationRequest getOpportunityRequest =
 GetEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte GetEngagementInvitation em Referência AWS SDK for Java 2.x da API.

`GetOpportunity`

O código de exemplo a seguir mostra como usar `GetOpportunity`.

SDK para Java 2.x

Tenha uma oportunidade.

```
package org.example;

import static org.example.utils.Constants.*;
```

```
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;

/*
 * Purpose
 * PC-API-08 Get updated Opportunity
 */

public class GetOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetOpportunityResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();
```

```
        GetOpportunityResponse response =
 client.getOpportunity(getOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte GetOpportunity em Referência AWS SDK for Java 2.x da API.

`ListEngagementInvitations`

O código de exemplo a seguir mostra como usar `ListEngagementInvitations`.

SDK para Java 2.x

Recupera uma lista de convites de engajamento enviados ao parceiro.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsReques
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsRespon
import software.amazon.awssdk.services.partnercentralselling.model.ParticipantType;
import
 software.amazon.awssdk.services.partnercentralselling.model.EngagementInvitationSummary;

public class ListEngagementInvitations {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
```

```
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     List<EngagementInvitationSummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    static List<EngagementInvitationSummary> getResponse() {

  List<EngagementInvitationSummary> opportunitySummaries = new
 ArrayList<EngagementInvitationSummary>();

  ListEngagementInvitationsRequest listOpportunityRequest =
 ListEngagementInvitationsRequest.builder()
               .catalog(CATALOG_TO_USE)
               .participantType(ParticipantType.RECEIVER)
         .maxResults(5)
         .build();

  ListEngagementInvitationsResponse response =
 client.listEngagementInvitations(listOpportunityRequest);

     opportunitySummaries.addAll(response.engagementInvitationSummaries());

     client.close();

        return opportunitySummaries;
  }
 }
```

- Para obter detalhes da API, consulte ListEngagementInvitations em Referência AWS SDK for Java 2.x da API.

`ListOpportunities`

O código de exemplo a seguir mostra como usar `ListOpportunities`.

SDK para Java 2.x

Liste oportunidades.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.OpportunitySummary;

/*
 * Purpose
 * PC-API-18 Getting list of Opportunities
 */

public class ListOpportunititesPaging {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<OpportunitySummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    private static List<OpportunitySummary> getResponse() {
```

```
    List<OpportunitySummary> opportunitySummaries = new
ArrayList<OpportunitySummary>();

 ListOpportunitiesRequest listOpportunityRequest =
ListOpportunitiesRequest.builder()
               .catalog(CATALOG_TO_USE)
         .maxResults(5)
         .build();

    ListOpportunitiesResponse response =
client.listOpportunities(listOpportunityRequest);

    opportunitySummaries.addAll(response.opportunitySummaries());

    while (response.nextToken() != null && response.nextToken().length() > 0) {
     listOpportunityRequest = ListOpportunitiesRequest.builder()
                   .catalog(CATALOG_TO_USE)
             .maxResults(5)
             .nextToken(response.nextToken())
             .build();
     response = client.listOpportunities(listOpportunityRequest);
     opportunitySummaries.addAll(response.opportunitySummaries());
    }

    client.close();

       return opportunitySummaries;
 }
}
```

- Para obter detalhes da API, consulte ListOpportunities em Referência AWS SDK for Java 2.x da API.

`ListSolutions`

O código de exemplo a seguir mostra como usar `ListSolutions`.

SDK para Java 2.x

Recupera uma lista de soluções de parceiros que o parceiro registrou na Central de Parceiros.

```
package org.example;
```

```
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsResponse;
import software.amazon.awssdk.services.partnercentralselling.model.SolutionBase;

/*
 * Purpose
 * PC-API-10 Getting list of solutions
 */

public class ListSolutions {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<SolutionBase> solutionSummaries = getResponse();
        ReferenceCodesUtils.formatOutput(solutionSummaries);
    }

    static List<SolutionBase> getResponse() {
  List<SolutionBase> solutionSummaries = new ArrayList<SolutionBase>();

  ListSolutionsRequest listSolutionsRequest = ListSolutionsRequest.builder()
    .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();
```

```
    ListSolutionsResponse response = client.listSolutions(listSolutionsRequest);

    solutionSummaries.addAll(response.solutionSummaries());

       return solutionSummaries;
 }
}
```

- Para obter detalhes da API, consulte ListSolutions em Referência AWS SDK for Java 2.x da API.

`RejectEngagementInvitation`

O código de exemplo a seguir mostra como usar `RejectEngagementInvitation`.

SDK para Java 2.x

Rejeita e EngagementInvitation isso é AWS compartilhado.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRespo

/*
 * Purpose
 * PC-API-05 AWS Originated(AO) rejection
 */

public class RejectEngagementInvitation {
```

```
 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

  RejectEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static RejectEngagementInvitationResponse getResponse(String invitationId) {

        RejectEngagementInvitationRequest rejectOpportunityRequest =
 RejectEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .rejectionReason("Unable to support")
          .build();

  RejectEngagementInvitationResponse response =
 client.rejectEngagementInvitation(rejectOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte RejectEngagementInvitation em Referência AWS SDK for Java 2.x da API.

StartEngagementByAcceptingInvitationTask

O código de exemplo a seguir mostra como usar `StartEngagementByAcceptingInvitationTask`.

SDK para Java 2.x

Inicia o noivado aceitando um EngagementInvitation.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse
import software.amazon.awssdk.services.partnercentralselling.model.InvitationStatus;

/*
Purpose
PC-API-04: Start Engagement By Accepting InvitationTask for AWS Originated(AO)
 opportunity
*/

public class StartEngagementByAcceptingInvitationTask {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String clientToken = "test-a30d161";

    public static void main(String[] args) {
```

```
    String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

    StartEngagementByAcceptingInvitationTaskResponse response =
getResponse(opportunityId);

    if ( response == null) {
     System.out.println("Opportunity is not AWS Originated.");
    } else {
     ReferenceCodesUtils.formatOutput(response);
    }
   }

   private static GetEngagementInvitationResponse getInvitation(String
invitationId) {

    GetEngagementInvitationRequest getRequest =
GetEngagementInvitationRequest.builder()
         .catalog(Constants.CATALOG_TO_USE)
         .identifier(invitationId)
         .build();

 GetEngagementInvitationResponse response =
client.getEngagementInvitation(getRequest);

       return response;
}

static StartEngagementByAcceptingInvitationTaskResponse getResponse(String
invitationId) {

 if ( getInvitation(invitationId).status().equals(InvitationStatus.PENDING)) {
  StartEngagementByAcceptingInvitationTaskRequest acceptOpportunityRequest =
    StartEngagementByAcceptingInvitationTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .clientToken(clientToken)
          .build();

  StartEngagementByAcceptingInvitationTaskResponse response =
client.startEngagementByAcceptingInvitationTask(acceptOpportunityRequest);
        return response;
 }
 return null;
}
```

```
}
```

- Para obter detalhes da API, consulte StartEngagementByAcceptingInvitationTask em Referência AWS SDK for Java 2.x da API.

StartEngagementFromOpportunityTask

O código de exemplo a seguir mostra como usar `StartEngagementFromOpportunityTask`.

SDK para Java 2.x

Inicia o processo de engajamento com base em uma oportunidade existente aceitando o convite de engajamento e criando uma oportunidade correspondente no sistema do parceiro.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.AwsSubmission;
import
 software.amazon.awssdk.services.partnercentralselling.model.SalesInvolvementType;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import software.amazon.awssdk.services.partnercentralselling.model.Visibility;

/*
 * Purpose
 * PC-API-01 Partner Originated (PO) opp submission(Start Engagement From
 Opportunity Task for AO Originated Opportunity)
 */

public class StartEngagementFromOpportunityTask {
```

```
 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementFromOpportunityTaskResponse response =
 getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static StartEngagementFromOpportunityTaskResponse getResponse(String opportunityId)
 {

  StartEngagementFromOpportunityTaskRequest submitOpportunityRequest =
 StartEngagementFromOpportunityTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .clientToken("test-annjqwesdsd99")

 .awsSubmission(AwsSubmission.builder().involvementType(SalesInvolvementType.CO_SELL).visibi
          .build();

  StartEngagementFromOpportunityTaskResponse response =
 client.startEngagementFromOpportunityTask(submitOpportunityRequest);

        return response;
 }
}
```

- Para obter detalhes da API, consulte StartEngagementFromOpportunityTask em Referência AWS SDK for Java 2.x da API.

`UpdateOpportunity`

O código de exemplo a seguir mostra como usar `UpdateOpportunity`.

SDK para Java 2.x

Atualize uma oportunidade.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.ReviewStatus;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
```

```
import com.google.gson.ToNumberPolicy;

/*
 * Purpose
 * PC-API-02/06 Update opportunity when LifeCycle.ReviewStatus is not Submitted or
 In-Review
 */

public class UpdateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String OPPORTUNITY_ORIGIN = ORIGIN_PARTNER_ORIGINATED;

 public static void main(String[] args) {

  String inputFile = "updateOpportunity.json";

  if (args.length > 0)
   inputFile = args[0];

  UpdateOpportunityResponse response = updateOpportunity(inputFile);

  client.close();
 }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();
```

```
        GetOpportunityResponse response =
client.getOpportunity(getOpportunityRequest);
        System.out.println(opportunityId + ":" + response);
        return response;
}

public static UpdateOpportunityResponse updateOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);
 GetOpportunityResponse response = getResponse(root.identifier);

 if (response != null
   && response.lifeCycle() != null
   && response.lifeCycle().reviewStatus() != null
   && response.lifeCycle().reviewStatus() != ReviewStatus.SUBMITTED
   && response.lifeCycle().reviewStatus() != ReviewStatus.IN_REVIEW) {

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
      .time(Instant.parse(nextStepsHistoryJson.time))
      .value(nextStepsHistoryJson.value)
                  .build();
    nextStepsHistories.add(nextStepsHistory);
   }
  }

  LifeCycle lifeCycle = null;
  if ( root.lifeCycle != null ) {
   lifeCycle = LifeCycle.builder()
    .closedLostReason(root.lifeCycle.closedLostReason)
    .nextSteps(root.lifeCycle.nextSteps)
    .nextStepsHistory(nextStepsHistories)
    .reviewComments(root.lifeCycle.reviewComments)
    .reviewStatus(root.lifeCycle.reviewStatus)
    .reviewStatusReason(root.lifeCycle.reviewStatusReason)
    .stage(root.lifeCycle.stage)
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
  }
```

```
  Marketing marketing = null;
  if ( root.marketing != null ) {
   marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();

  }

  Address address = null;
  if (root.customer != null && root.customer.account != null &&
root.customer.account.address != null) {
   address = Address.builder().postalCode(root.customer.account.address.postalCode)
     .stateOrRegion(root.customer.account.address.stateOrRegion)
     .countryCode(root.customer.account.address.countryCode).build();
  }

  Account account = null;
  if (root.customer != null && root.customer.account != null) {
   account = Account.builder().address(address).duns(root.customer.account.duns)

.industry(root.customer.account.industry).companyName(root.customer.account.companyName)
     .websiteUrl(root.customer.account.websiteUrl).build();
  }

  List<Contact> contacts = new ArrayList<Contact>();
  if ( root.customer != null && root.customer.contacts != null) {
   for (org.example.entity.Contact jsonContact : root.customer.contacts) {
    Contact contact = Contact.builder()
                   .email(jsonContact.email)
                   .firstName(jsonContact.firstName)
                   .lastName(jsonContact.lastName)
                   .phone(jsonContact.phone)
                   .businessTitle(jsonContact.businessTitle)
                   .build();
    contacts.add(contact);
   }
  }
```

```
  Customer customer =
Customer.builder().account(account).contacts(contacts).build();

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

  Project project = null;
  if (root.project != null) {
   project = Project.builder().title(root.project.title)
     .customerBusinessProblem(root.project.customerBusinessProblem)

.customerUseCase(root.project.customerUseCase).deliveryModels(root.project.deliveryModels)
     .expectedCustomerSpend(expectedCustomerSpends)

.salesActivities(root.project.salesActivities).competitorName(root.project.competitorName)
     .otherSolutionDescription(root.project.otherSolutionDescription).build();
  }

  // Building the Actual CreateOpportunity Request
  UpdateOpportunityRequest updateOpportunityRequest =
UpdateOpportunityRequest.builder().catalog(root.catalog)

.identifier(root.identifier).lastModifiedDate(Instant.parse(root.lastModifiedDate))

.primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws).opportunityType(root.opportunityT
    .lifeCycle(lifeCycle)
    .customer(customer)
    .project(project)
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
    .marketing(marketing)
    .nationalSecurity(root.nationalSecurity)
```

```
      .opportunityType(root.opportunityType)
      .build();

    UpdateOpportunityResponse updateResponse =
 client.updateOpportunity(updateOpportunityRequest);
    System.out.println("Successfully updated opportunity: " + updateResponse);

    return updateResponse;
  } else {
    System.out.println("Opportunity cannot be updated.");
    return null;
  }
    }
}
```

- Para obter detalhes da API, consulte UpdateOpportunity em Referência AWS SDK for Java 2.x da API.

Cenários

Atualizar a entidade associada de uma oportunidade

O exemplo de código a seguir mostra como:

- Desassociar uma entidade antiga.
- Associar uma entidade nova.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório Cenários.

Atualizar a entidade associada de uma oportunidade

```
package org.example;
```

```
import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -17 Replacing a solution
*/

public class ReplaceSolution {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";
     String originalEntityIdentifier = "S-0000000";
     String newEntityIdentifier = "S-0011111";

     disassociateOppornititityResponse(opportunityId, entityType,
 originalEntityIdentifier );
     AssociateOpportunityResponse associateOpportunityResponse =
 associateOpportunityResponse(opportunityId, entityType, newEntityIdentifier );
```

```
    ReferenceCodesUtils.formatOutput(associateOpportunityResponse);
   }

private static AssociateOpportunityResponse associateOpportunityResponse(String
opportunityId, String entityType, String entityIdentifier) {

       AssociateOpportunityRequest associateOpportunityRequest =
AssociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();

       AssociateOpportunityResponse response =
client.associateOpportunity(associateOpportunityRequest);

       return response;
}

private static DisassociateOpportunityResponse
disassociateOppornitityResponse(String opportunityId, String entityType, String
entityIdentifier) {
 PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

       DisassociateOpportunityRequest disassociateOpportunityRequest =
DisassociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();

       DisassociateOpportunityResponse response =
client.disassociateOpportunity(disassociateOpportunityRequest);

       return response;
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AssociateOpportunity
  - DisassociateOpportunity

Exemplos do Amazon Personalize usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon Personalize.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateBatchInferenceJob`

O código de exemplo a seguir mostra como usar `CreateBatchInferenceJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                        String solutionVersionArn,
```

```
                        String jobName,
                        String s3InputDataSourcePath,
                        String s3DataDestinationPath,
                        String roleArn,
                        String explorationWeight,
                        String explorationItemAgeCutOff) {

                long waitInMilliseconds = 60 * 1000;
                String status;
                String batchInferenceJobArn;

                try {

                        // Set up data input and output parameters.
                        S3DataConfig inputSource = S3DataConfig.builder()
                                        .path(s3InputDataSourcePath)
                                        .build();

                        S3DataConfig outputDestination = S3DataConfig.builder()
                                        .path(s3DataDestinationPath)
                                        .build();

                        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
                                        .s3DataSource(inputSource)
                                        .build();

                        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
                                        .s3DataDestination(outputDestination)
                                        .build();

                        // Optional code to build the User-Personalization specific
item exploration
                        // config.
                        HashMap<String, String> explorationConfig = new HashMap<>();

                        explorationConfig.put("explorationWeight",
explorationWeight);
                        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

                        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
```

```
                                        .itemExplorationConfig(explorationConfig)
                                        .build();

                        // End optional User-Personalization recipe specific code.

                        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
                                        .builder()
                                        .solutionVersionArn(solutionVersionArn)
                                        .jobInput(jobInput)
                                        .jobOutput(jobOutputLocation)
                                        .jobName(jobName)
                                        .roleArn(roleArn)
                                        .batchInferenceJobConfig(jobConfig) //
Optional
                                        .build();

                        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                                        .batchInferenceJobArn();

                        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
                                        .builder()
                                        .batchInferenceJobArn(batchInferenceJobArn)
                                        .build();

                        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
                        while (Instant.now().getEpochSecond() < maxTime) {

                                BatchInferenceJob batchInferenceJob =
personalizeClient

.describeBatchInferenceJob(describeBatchInferenceJobRequest)
                                                .batchInferenceJob();

                                status = batchInferenceJob.status();
                                System.out.println("Batch inference job status: " +
status);

                                if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {
                                        break;
                                }
```

```
                                try {
                                        Thread.sleep(waitInMilliseconds);
                                } catch (InterruptedException e) {
                                        System.out.println(e.getMessage());
                                }
                        }
                        return batchInferenceJobArn;

                } catch (PersonalizeException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return "";
        }
```

- Para obter detalhes da API, consulte CreateBatchInferenceJob em Referência AWS SDK for Java 2.x da API.

`CreateCampaign`

O código de exemplo a seguir mostra como usar `CreateCampaign`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createPersonalCompaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
           String name) {

       try {
           CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
                   .minProvisionedTPS(1)
                   .solutionVersionArn(solutionVersionArn)
                   .name(name)
                   .build();
```

```
            CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
            System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());
            return campaignResponse.campaignArn();
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte CreateCampaign em Referência AWS SDK for Java 2.x da API.

`CreateDataset`

O código de exemplo a seguir mostra como usar `CreateDataset`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String createDataset(PersonalizeClient personalizeClient,
            String datasetName,
            String datasetGroupArn,
            String datasetType,
            String schemaArn) {
        try {
            CreateDatasetRequest request = CreateDatasetRequest.builder()
                    .name(datasetName)
                    .datasetGroupArn(datasetGroupArn)
                    .datasetType(datasetType)
                    .schemaArn(schemaArn)
                    .build();
```

```
            String datasetArn = personalizeClient.createDataset(request)
                    .datasetArn();
            System.out.println("Dataset " + datasetName + " created.");
            return datasetArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateDataset em Referência AWS SDK for Java 2.x da API.

`CreateDatasetExportJob`

O código de exemplo a seguir mostra como usar `CreateDatasetExportJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String createDatasetExportJob(PersonalizeClient personalizeClient,
            String jobName,
            String datasetArn,
            IngestionMode ingestionMode,
            String roleArn,
            String s3BucketPath,
            String kmsKeyArn) {

        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String status = null;

        try {
```

```
            S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
            DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
                    .build();

            CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
                    .jobName(jobName)
                    .datasetArn(datasetArn)
                    .ingestionMode(ingestionMode)
                    .jobOutput(jobOutput)
                    .roleArn(roleArn)
                    .build();

            String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

            DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
                    .datasetExportJobArn(datasetExportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetExportJob datasetExportJob = personalizeClient
                        .describeDatasetExportJob(describeDatasetExportJobRequest)
                        .datasetExportJob();

                status = datasetExportJob.status();
                System.out.println("Export job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    return status;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
```

```
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateDatasetExportJob em Referência AWS SDK for Java 2.x da API.

`CreateDatasetGroup`

O código de exemplo a seguir mostra como usar `CreateDatasetGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

       try {
           CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                   .name(datasetGroupName)
                   .build();
           return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
       } catch (PersonalizeException e) {
           System.out.println(e.awsErrorDetails().errorMessage());
       }
       return "";
   }
```

Criar um grupo de conjunto de dados de domínio.

```
   public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
           String datasetGroupName,
           String domain) {

       try {
           CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                   .name(datasetGroupName)
                   .domain(domain)
                   .build();
           return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
       } catch (PersonalizeException e) {
           System.out.println(e.awsErrorDetails().errorMessage());
       }
       return "";
   }
```

- Para obter detalhes da API, consulte CreateDatasetGroup em Referência AWS SDK for Java 2.x da API.

`CreateDatasetImportJob`

O código de exemplo a seguir mostra como usar `CreateDatasetImportJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
           String jobName,
           String datasetArn,
           String s3BucketPath,
           String roleArn) {
```

```
        long waitInMilliseconds = 60 * 1000;
        String status;
        String datasetImportJobArn;

        try {
            DataSource importDataSource = DataSource.builder()
                    .dataLocation(s3BucketPath)
                    .build();

            CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
                    .datasetArn(datasetArn)
                    .dataSource(importDataSource)
                    .jobName(jobName)
                    .roleArn(roleArn)
                    .build();

            datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                    .datasetImportJobArn();
            DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
                    .datasetImportJobArn(datasetImportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetImportJob datasetImportJob = personalizeClient
                        .describeDatasetImportJob(describeDatasetImportJobRequest)
                        .datasetImportJob();

                status = datasetImportJob.status();
                System.out.println("Dataset import job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
```

```
                }
            }
            return datasetImportJobArn;

        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateDatasetImportJob em Referência AWS SDK for Java 2.x da API.

`CreateEventTracker`

O código de exemplo a seguir mostra como usar `CreateEventTracker`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
            String datasetGroupArn) {

        String eventTrackerId = "";
        String eventTrackerArn;
        long maxTime = 3 * 60 * 60; // 3 hours
        long waitInMilliseconds = 20 * 1000; // 20 seconds
        String status;

        try {

            CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
                    .name(eventTrackerName)
```

```
                    .datasetGroupArn(datasetGroupArn)
                    .build();

            CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
                    .createEventTracker(createEventTrackerRequest);

            eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
            eventTrackerId = createEventTrackerResponse.trackingId();
            System.out.println("Event tracker ARN: " + eventTrackerArn);
            System.out.println("Event tracker ID: " + eventTrackerId);

            maxTime = Instant.now().getEpochSecond() + maxTime;

            DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
                    .eventTrackerArn(eventTrackerArn)
                    .build();

            while (Instant.now().getEpochSecond() < maxTime) {

                status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
                System.out.println("EventTracker status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return eventTrackerId;
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return eventTrackerId;
    }
```

- Para obter detalhes da API, consulte CreateEventTracker em Referência AWS SDK for Java 2.x da API.

`CreateFilter`

O código de exemplo a seguir mostra como usar `CreateFilter`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static String createFilter(PersonalizeClient personalizeClient,
        String filterName,
        String datasetGroupArn,
        String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .datasetGroupArn(datasetGroupArn)
                .filterExpression(filterExpression)
                .build();

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Para obter detalhes da API, consulte CreateFilter em Referência AWS SDK for Java 2.x da API.

`CreateRecommender`

O código de exemplo a seguir mostra como usar `CreateRecommender`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);

            DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {
```

```
                recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
                System.out.println("Recommender status: " + recommenderStatus);

                if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return recommenderArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateRecommender em Referência AWS SDK for Java 2.x da API.

`CreateSchema`

O código de exemplo a seguir mostra como usar `CreateSchema`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

       String schema = null;
       try {
           schema = new String(Files.readAllBytes(Paths.get(filePath)));
       } catch (IOException e) {
           System.out.println(e.getMessage());
       }

       try {
           CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                   .name(schemaName)
                   .schema(schema)
                   .build();

           String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

           System.out.println("Schema arn: " + schemaArn);

           return schemaArn;

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

Crie um esquema com um domínio.

```
   public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
           String filePath) {

       String schema = null;
       try {
           schema = new String(Files.readAllBytes(Paths.get(filePath)));
       } catch (IOException e) {
           System.out.println(e.getMessage());
       }
```

```
        try {
            CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                    .name(schemaName)
                    .domain(domain)
                    .schema(schema)
                    .build();

            String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

            System.out.println("Schema arn: " + schemaArn);

            return schemaArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateSchema em Referência AWS SDK for Java 2.x da API.

`CreateSolution`

O código de exemplo a seguir mostra como usar `CreateSolution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
            String datasetGroupArn,
```

```
            String solutionName,
            String recipeArn) {

        try {
            CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
                    .name(solutionName)
                    .datasetGroupArn(datasetGroupArn)
                    .recipeArn(recipeArn)
                    .build();

            CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
            return solutionResponse.solutionArn();

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateSolution em Referência AWS SDK for Java 2.x da API.

`CreateSolutionVersion`

O código de exemplo a seguir mostra como usar `CreateSolutionVersion`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
```

```
        String solutionStatus = "";
        String solutionVersionStatus = "";
        String solutionVersionArn = "";

        try {
            DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
                    .solutionArn(solutionArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            // Wait until solution is active.
            while (Instant.now().getEpochSecond() < maxTime) {

                solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
                System.out.println("Solution status: " + solutionStatus);

                if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }

            if (solutionStatus.equals("ACTIVE")) {

                CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                        .solutionArn(solutionArn)
                        .build();

                CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                        .createSolutionVersion(createSolutionVersionRequest);
                solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

                System.out.println("Solution version ARN: " + solutionVersionArn);
```

```
                DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                        .solutionVersionArn(solutionVersionArn)
                        .build();

                while (Instant.now().getEpochSecond() < maxTime) {

                    solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                            .solutionVersion().status();
                    System.out.println("Solution version status: " +
solutionVersionStatus);

                    if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                        break;
                    }
                    try {
                        Thread.sleep(waitInMilliseconds);
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                return solutionVersionArn;
            }
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateSolutionVersion em Referência AWS SDK for Java 2.x da API.

DeleteCampaign

O código de exemplo a seguir mostra como usar `DeleteCampaign`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
 String campaignArn) {
        try {
            DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            personalizeClient.deleteCampaign(campaignRequest);
            System.out.println("Delete request sent successfully.");
        } catch (PersonalizeException e) {
            System.err.println("Error deleting campaign: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteCampaign em Referência AWS SDK for Java 2.x da API.

`DeleteEventTracker`

O código de exemplo a seguir mostra como usar `DeleteEventTracker`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
       try {
           DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
                   .eventTrackerArn(eventTrackerArn)
                   .build();

           int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCod

           System.out.println("Status code:" + status);

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DeleteEventTracker em Referência AWS SDK for Java 2.x da API.

`DeleteSolution`

O código de exemplo a seguir mostra como usar `DeleteSolution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

       try {
           DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
                   .solutionArn(solutionArn)
```

```
                    .build();

            personalizeClient.deleteSolution(solutionRequest);
            System.out.println("Done");

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteSolution em Referência AWS SDK for Java 2.x da API.

`DescribeCampaign`

O código de exemplo a seguir mostra como usar `DescribeCampaign`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

        try {
            DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
            Campaign myCampaign = campaignResponse.campaign();
            System.out.println("The Campaign name is " + myCampaign.name());
            System.out.println("The Campaign status is " + myCampaign.status());
```

```
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeCampaign em Referência AWS SDK for Java 2.x da API.

`DescribeRecipe`

O código de exemplo a seguir mostra como usar `DescribeRecipe`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

        try {
            DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
                    .recipeArn(recipeArn)
                    .build();

            DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
            System.out.println("The recipe name is " +
recipeResponse.recipe().name());

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeRecipe em Referência AWS SDK for Java 2.x da API.

`DescribeSolution`

O código de exemplo a seguir mostra como usar `DescribeSolution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

       try {
           DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
           System.out.println("The Solution name is " +
response.solution().name());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeSolution em Referência AWS SDK for Java 2.x da API.

`ListCampaigns`

O código de exemplo a seguir mostra como usar `ListCampaigns`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

       try {
           ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
                   .maxResults(10)
                   .solutionArn(solutionArn)
                   .build();

           ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
           List<CampaignSummary> campaigns = response.campaigns();
           for (CampaignSummary campaign : campaigns) {
               System.out.println("Campaign name is : " + campaign.name());
               System.out.println("Campaign ARN is : " + campaign.campaignArn());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ListCampaigns em Referência AWS SDK for Java 2.x da API.

`ListDatasetGroups`

O código de exemplo a seguir mostra como usar `ListDatasetGroups`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listDSGroups(PersonalizeClient personalizeClient) {

       try {
           ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
                   .maxResults(15)
                   .build();

           ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
           List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
           for (DatasetGroupSummary group : groups) {
               System.out.println("The DataSet name is : " + group.name());
               System.out.println("The DataSet ARN is : " +
group.datasetGroupArn());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ListDatasetGroups em Referência AWS SDK for Java 2.x da API.

`ListRecipes`

O código de exemplo a seguir mostra como usar `ListRecipes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listAllRecipes(PersonalizeClient personalizeClient) {

        try {
            ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
                    .maxResults(15)
                    .build();

            ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
            List<RecipeSummary> recipes = response.recipes();
            for (RecipeSummary recipe : recipes) {
                System.out.println("The recipe ARN is: " + recipe.recipeArn());
                System.out.println("The recipe name is: " + recipe.name());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListRecipes em Referência AWS SDK for Java 2.x da API.

`ListSolutions`

O código de exemplo a seguir mostra como usar `ListSolutions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {

       try {
           ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
                   .maxResults(10)
                   .datasetGroupArn(datasetGroupArn)
                   .build();

           ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
           List<SolutionSummary> solutions = response.solutions();
           for (SolutionSummary solution : solutions) {
               System.out.println("The solution ARN is: " +
solution.solutionArn());
               System.out.println("The solution name is: " + solution.name());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ListSolutions em Referência AWS SDK for Java 2.x da API.

`UpdateCampaign`

O código de exemplo a seguir mostra como usar `UpdateCampaign`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String updateCampaign(PersonalizeClient personalizeClient,
           String campaignArn,
           String solutionVersionArn,
           Integer minProvisionedTPS) {

       try {
           // build the updateCampaignRequest
           UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
                   .campaignArn(campaignArn)
                   .solutionVersionArn(solutionVersionArn)
                   .minProvisionedTPS(minProvisionedTPS)
                   .build();

           // update the campaign
           personalizeClient.updateCampaign(updateCampaignRequest);

           DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                   .campaignArn(campaignArn)
                   .build();

           DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
           Campaign updatedCampaign = campaignResponse.campaign();

           System.out.println("The Campaign status is " +
updatedCampaign.status());
           return updatedCampaign.status();

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
        return "";
    }
```

- Para obter detalhes da API, consulte UpdateCampaign em Referência AWS SDK for Java 2.x da API.

Exemplos de eventos do Amazon Personalize usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Personalize Events.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                        String datasetArn,
                        String item1Id,
```

```
                        String item1PropertyName,
                        String item1PropertyValue,
                        String item2Id,
                        String item2PropertyName,
                        String item2PropertyValue) {

                int responseCode = 0;
                ArrayList<Item> items = new ArrayList<>();

                try {
                        Item item1 = Item.builder()
                                        .itemId(item1Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        item1PropertyName,
 item1PropertyValue))
                                        .build();

                        items.add(item1);

                        Item item2 = Item.builder()
                                        .itemId(item2Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        item2PropertyName,
 item2PropertyValue))
                                        .build();

                        items.add(item2);

                        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
                                        .datasetArn(datasetArn)
                                        .items(items)
                                        .build();

                        responseCode =
 personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
```

```
        }
```

- Para obter detalhes da API, consulte PutEvents em Referência AWS SDK for Java 2.x da API.

PutUsers

O código de exemplo a seguir mostra como usar `PutUsers`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
                        String datasetArn,
                        String user1Id,
                        String user1PropertyName,
                        String user1PropertyValue,
                        String user2Id,
                        String user2PropertyName,
                        String user2PropertyValue) {

                int responseCode = 0;
                ArrayList<User> users = new ArrayList<>();

                try {
                        User user1 = User.builder()
                                        .userId(user1Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        user1PropertyName,
 user1PropertyValue))
                                        .build();

                        users.add(user1);

                        User user2 = User.builder()
                                        .userId(user2Id)
```

```
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        user2PropertyName,
 user2PropertyValue))
                                        .build();

                        users.add(user2);

                        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
                                        .datasetArn(datasetArn)
                                        .users(users)
                                        .build();

                        responseCode =
 personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
        }
```

- Para obter detalhes da API, consulte PutUsers em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Personalize Runtime usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon Personalize Runtime.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetPersonalizedRanking`

O código de exemplo a seguir mostra como usar `GetPersonalizedRanking`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
           String campaignArn,
           String userId,
           ArrayList<String> items) {

       try {
           GetPersonalizedRankingRequest rankingRecommendationsRequest =
GetPersonalizedRankingRequest.builder()
                   .campaignArn(campaignArn)
                   .userId(userId)
                   .inputList(items)
                   .build();

           GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient
                   .getPersonalizedRanking(rankingRecommendationsRequest);
           List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
           int rank = 1;
           for (PredictedItem item : rankedItems) {
               System.out.println("Item ranked at position " + rank + " details");
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
               System.out.println("----------------------------------------------");
               rank++;
           }
           return rankedItems;
```

```
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte GetPersonalizedRanking em Referência AWS SDK for Java 2.x da API.

`GetRecommendations`

O código de exemplo a seguir mostra como usar `GetRecommendations`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obter uma lista de itens recomendados.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .campaignArn(campaignArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();
           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
```

```
                System.out.println("Item score is : " + item.score());
            }

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Obtenha uma lista de itens recomendados de um recomendador criado em um grupo de conjunto de dados de domínio.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
           String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .recommenderArn(recommenderArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();

           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
           }
       } catch (AwsServiceException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Use um filtro ao solicitar recomendações.

```
   public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
           String campaignArn,
           String userId,
           String filterArn,
           String parameter1Name,
           String parameter1Value1,
           String parameter1Value2,
           String parameter2Name,
           String parameter2Value) {

       try {

           Map<String, String> filterValues = new HashMap<>();

           filterValues.put(parameter1Name, String.format("\"%1$s\",\"%2$s\"",
                   parameter1Value1, parameter1Value2));
           filterValues.put(parameter2Name, String.format("\"%1$s\"",
                   parameter2Value));

           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .campaignArn(campaignArn)
                   .numResults(20)
                   .userId(userId)
                   .filterArn(filterArn)
                   .filterValues(filterValues)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();

           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
           }
       } catch (PersonalizeRuntimeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte GetRecommendations em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Pinpoint usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateApp`

O código de exemplo a seguir mostra como usar `CreateApp`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = """

                 Usage:  <appName>

                 Where:
                  appName - The name of the application to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String appName = args[0];
        System.out.println("Creating an application with name: " + appName);

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String appID = createApplication(pinpoint, appName);
        System.out.println("App ID is: " + appID);
        pinpoint.close();
    }

    public static String createApplication(PinpointClient pinpoint, String appName)
 {
        try {
            CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
                    .name(appName)
                    .build();
```

```
            CreateAppRequest request = CreateAppRequest.builder()
                    .createApplicationRequest(appRequest)
                    .build();

            CreateAppResponse result = pinpoint.createApp(request);
            return result.applicationResponse().id();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateApp em Referência AWS SDK for Java 2.x da API.

`CreateCampaign`

O código de exemplo a seguir mostra como usar `CreateCampaign`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma campanha.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
```

```
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCampaign {
    public static void main(String[] args) {

        final String usage = """

                Usage:   <appId> <segmentId>

                Where:
                  appId - The ID of the application to create the campaign in.
                  segmentId - The ID of the segment to create the campaign from.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String segmentId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPinCampaign(pinpoint, appId, segmentId);
        pinpoint.close();
    }

    public static void createPinCampaign(PinpointClient pinpoint, String appId,
 String segmentId) {
        CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
        System.out.println("Campaign " + result.name() + " created.");
        System.out.println(result.description());
    }
```

```
    public static CampaignResponse createCampaign(PinpointClient client, String
 appID, String segmentID) {

        try {
            Schedule schedule = Schedule.builder()
                    .startTime("IMMEDIATE")
                    .build();

            Message defaultMessage = Message.builder()
                    .action(Action.OPEN_APP)
                    .body("My message body.")
                    .title("My message title.")
                    .build();

            MessageConfiguration messageConfiguration =
 MessageConfiguration.builder()
                    .defaultMessage(defaultMessage)
                    .build();

            WriteCampaignRequest request = WriteCampaignRequest.builder()
                    .description("My description")
                    .schedule(schedule)
                    .name("MyCampaign")
                    .segmentId(segmentID)
                    .messageConfiguration(messageConfiguration)
                    .build();

            CreateCampaignResponse result =
 client.createCampaign(CreateCampaignRequest.builder()
                    .applicationId(appID)
                    .writeCampaignRequest(request).build());

            System.out.println("Campaign ID: " + result.campaignResponse().id());
            return result.campaignResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
}
```

- Para obter detalhes da API, consulte CreateCampaign em Referência AWS SDK for Java 2.x da API.

`CreateExportJob`

O código de exemplo a seguir mostra como usar `CreateExportJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exportar um endpoint.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
```

```
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-
export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = """

                This program performs the following steps:

                1. Exports the endpoints to an Amazon S3 bucket.
                2. Downloads the exported endpoints files from Amazon S3.
                3. Parses the endpoints files to obtain the endpoint IDs and prints
 them.
                Usage: ExportEndpoints <applicationId> <s3BucketName>
 <iamExportRoleArn> <path>

                Where:
                  applicationId - The ID of the Amazon Pinpoint application that has
 the endpoint.
                  s3BucketName - The name of the Amazon S3 bucket to export the JSON
 file to.\s
                  iamExportRoleArn - The ARN of an IAM role that grants Amazon
 Pinpoint write permissions to the S3 bucket.  path - The path where the files
 downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String applicationId = args[0];
        String s3BucketName = args[1];
        String iamExportRoleArn = args[2];
        String path = args[3];
        System.out.println("Deleting an application with ID: " + applicationId);

        Region region = Region.US_EAST_1;
        PinpointClient pinpoint = PinpointClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
        pinpoint.close();
        s3Client.close();
    }

    public static void exportAllEndpoints(PinpointClient pinpoint,
            S3Client s3Client,
            String applicationId,
            String s3BucketName,
            String path,
            String iamExportRoleArn) {

        try {
            List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
s3BucketName, iamExportRoleArn,
                    applicationId);
            List<String> endpointFileKeys = objectKeys.stream().filter(o ->
o.endsWith(".gz"))
                    .collect(Collectors.toList());
            downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
 s3Client, String s3BucketName,
            String iamExportRoleArn, String applicationId) {

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
        String endpointsKeyPrefix = "exports/" + applicationId + "_" +
 dateFormat.format(new Date());
        String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
 "/";
        List<String> objectKeys = new ArrayList<>();
        String key;

        try {
            // Defines the export job that Amazon Pinpoint runs.
            ExportJobRequest jobRequest = ExportJobRequest.builder()
                    .roleArn(iamExportRoleArn)
                    .s3UrlPrefix(s3UrlPrefix)
                    .build();

            CreateExportJobRequest exportJobRequest =
 CreateExportJobRequest.builder()
                    .applicationId(applicationId)
                    .exportJobRequest(jobRequest)
                    .build();

            System.out.format("Exporting endpoints from Amazon Pinpoint application
 %s to Amazon S3 " +
                    "bucket %s . . .\n", applicationId, s3BucketName);

            CreateExportJobResponse exportResult =
 pinpoint.createExportJob(exportJobRequest);
            String jobId = exportResult.exportJobResponse().id();
            System.out.println(jobId);
            printExportJobStatus(pinpoint, applicationId, jobId);

            ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
                    .bucket(s3BucketName)
                    .prefix(endpointsKeyPrefix)
                    .build();

            // Create a list of object keys.
            ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
```

```
            List<S3Object> objects = v2Response.contents();
            for (S3Object object : objects) {
                key = object.key();
                objectKeys.add(key);
            }

            return objectKeys;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void printExportJobStatus(PinpointClient pinpointClient,
            String applicationId,
            String jobId) {

        GetExportJobResponse getExportJobResult;
        String status;

        try {
            // Checks the job status until the job completes or fails.
            GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
                    .jobId(jobId)
                    .applicationId(applicationId)
                    .build();

            do {
                getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
                status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
                System.out.format("Export job %s . . .\n", status);
                TimeUnit.SECONDS.sleep(3);

            } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

            if (status.equals("COMPLETED")) {
                System.out.println("Finished exporting endpoints.");
            } else {
                System.err.println("Failed to export endpoints.");
                System.exit(1);
            }
```

```
        } catch (PinpointException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Download files from an Amazon S3 bucket and write them to the path location.
    public static void downloadFromS3(S3Client s3Client, String path, String
 s3BucketName, List<String> objectKeys) {

        String newPath;
        try {
            for (String key : objectKeys) {
                GetObjectRequest objectRequest = GetObjectRequest.builder()
                        .bucket(s3BucketName)
                        .key(key)
                        .build();

                ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
                byte[] data = objectBytes.asByteArray();

                // Write the data to a local file.
                String fileSuffix = new
 SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
                newPath = path + fileSuffix + ".gz";
                File myFile = new File(newPath);
                OutputStream os = new FileOutputStream(myFile);
                os.write(data);
            }
            System.out.println("Download finished.");

        } catch (S3Exception | NullPointerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateExportJob em Referência AWS SDK for Java 2.x da API.

`CreateImportJob`

O código de exemplo a seguir mostra como usar `CreateImportJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Importar um segmento.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <bucket> <key> <roleArn>\s

                Where:
                  appId - The application ID to create a segment for.
                  bucket - The name of the Amazon S3 bucket that contains the
 segment definitons.
                  key - The key of the S3 object.
```

```
                  roleArn - ARN of the role that allows Amazon Pinpoint to
 access S3. You need to set trust management for this to work. See https://
docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html
                  """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
 key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
 import job status.");
        System.out.println("See application " + response.jobStatus() + " for import
 job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
            String appId,
            String bucket,
            String key,
            String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                    .defineSegment(true)
                    .registerEndpoints(true)
                    .roleArn(roleArn)
                    .format(Format.JSON)
                    .s3Url("s3://" + bucket + "/" + key)
                    .build();
```

```
            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                    .importJobRequest(importRequest)
                    .applicationId(appId)
                    .build();

            CreateImportJobResponse jobResponse =
 client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Para obter detalhes da API, consulte CreateImportJob em Referência AWS SDK for Java 2.x da API.

`CreateSegment`

O código de exemplo a seguir mostra como usar `CreateSegment`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
```

```
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSegment {
        public static void main(String[] args) {
                final String usage = """

                                Usage:   <appId>

                                Where:
                                  appId - The application ID to create a segment
 for.

                                """;

                if (args.length != 1) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String appId = args[0];
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                SegmentResponse result = createSegment(pinpoint, appId);
                System.out.println("Segment " + result.name() + " created.");
                System.out.println(result.segmentType());
                pinpoint.close();
        }
```

```
        public static SegmentResponse createSegment(PinpointClient client, String
appId) {
                try {
                        Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();
                        segmentAttributes.put("Team", AttributeDimension.builder()
                                        .attributeType(AttributeType.INCLUSIVE)
                                        .values("Lakers")
                                        .build());

                        RecencyDimension recencyDimension =
RecencyDimension.builder()
                                        .duration("DAY_30")
                                        .recencyType("ACTIVE")
                                        .build();

                        SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
                                        .recency(recencyDimension)
                                        .build();

                        SegmentDemographics segmentDemographics =
SegmentDemographics
                                        .builder()
                                        .build();

                        SegmentLocation segmentLocation = SegmentLocation
                                        .builder()
                                        .build();

                        SegmentDimensions dimensions = SegmentDimensions
                                        .builder()
                                        .attributes(segmentAttributes)
                                        .behavior(segmentBehaviors)
                                        .demographic(segmentDemographics)
                                        .location(segmentLocation)
                                        .build();

                        WriteSegmentRequest writeSegmentRequest =
WriteSegmentRequest.builder()
                                        .name("MySegment")
                                        .dimensions(dimensions)
                                        .build();
```

```
                        CreateSegmentRequest createSegmentRequest =
 CreateSegmentRequest.builder()
                                        .applicationId(appId)
                                        .writeSegmentRequest(writeSegmentRequest)
                                        .build();

                        CreateSegmentResponse createSegmentResult =
 client.createSegment(createSegmentRequest);
                        System.out.println("Segment ID: " +
 createSegmentResult.segmentResponse().id());
                        System.out.println("Done");
                        return createSegmentResult.segmentResponse();

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
                return null;
        }
}
```

- Para obter detalhes da API, consulte CreateSegment em Referência AWS SDK for Java 2.x da API.

`DeleteApp`

O código de exemplo a seguir mostra como usar `DeleteApp`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir um aplicativo.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
```

```
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                 appId - The ID of the application to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        System.out.println("Deleting an application with ID: " + appId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinApp(pinpoint, appId);
        System.out.println("Done");
        pinpoint.close();
    }

    public static void deletePinApp(PinpointClient pinpoint, String appId) {
        try {
            DeleteAppRequest appRequest = DeleteAppRequest.builder()
                    .applicationId(appId)
                    .build();
```

```
            DeleteAppResponse result = pinpoint.deleteApp(appRequest);
            String appName = result.applicationResponse().name();
            System.out.println("Application " + appName + " has been deleted.");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteApp em Referência AWS SDK for Java 2.x da API.

`DeleteEndpoint`

O código de exemplo a seguir mostra como usar `DeleteEndpoint`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir um endpoint

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appName> <endpointId >

                Where:
                  appId - The id of the application to delete.
                  endpointId - The id of the endpoint to delete.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinEncpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    public static void deletePinEncpoint(PinpointClient pinpoint, String appId,
 String endpointId) {
        try {
            DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
            String id = result.endpointResponse().id();
            System.out.println("The deleted endpoint id  " + id);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte DeleteEndpoint em Referência AWS SDK for Java 2.x da API.

GetEndpoint

O código de exemplo a seguir mostra como usar `GetEndpoint`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:   <appId> <endpoint>

                Where:
                  appId - The ID of the application to delete.
                  endpoint - The ID of the endpoint.\s
                  """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }

    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
String endpoint) {
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpoint)
                    .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();

            // Uses the Google Gson library to pretty print the endpoint JSON.
            Gson gson = new GsonBuilder()
                    .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                    .setPrettyPrinting()
                    .create();

            String endpointJson = gson.toJson(endResponse);
            System.out.println(endpointJson);
```

```
        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Para obter detalhes da API, consulte GetEndpoint em Referência AWS SDK for Java 2.x da API.

`GetSegments`

O código de exemplo a seguir mostra como usar `GetSegments`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar segmentos.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListSegments {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId>

                Where:
                  appId - The ID of the application that contains a segment.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSegs(pinpoint, appId);
        pinpoint.close();
    }

    public static void listSegs(PinpointClient pinpoint, String appId) {
        try {
            GetSegmentsRequest request = GetSegmentsRequest.builder()
                    .applicationId(appId)
                    .build();

            GetSegmentsResponse response = pinpoint.getSegments(request);
            List<SegmentResponse> segments = response.segmentsResponse().item();
            for (SegmentResponse segment : segments) {
                System.out
                        .println("Segement " + segment.id() + " " + segment.name() +
 " " + segment.lastModifiedDate());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte GetSegments em Referência AWS SDK for Java 2.x da API.

`GetSmsChannel`

O código de exemplo a seguir mostra como usar `GetSmsChannel`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = """

                Usage: CreateChannel <appId>
```

```
                Where:
                  appId - The name of the application whose channel is updated.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        SMSChannelResponse getResponse = getSMSChannel(pinpoint, appId);
        toggleSmsChannel(pinpoint, appId, getResponse);
        pinpoint.close();
    }

    private static SMSChannelResponse getSMSChannel(PinpointClient client, String
appId) {
        try {
            GetSmsChannelRequest request = GetSmsChannelRequest.builder()
                    .applicationId(appId)
                    .build();

            SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
            System.out.println("Channel state is " + response.enabled());
            return response;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
        boolean enabled = !getResponse.enabled();
        try {
            SMSChannelRequest request = SMSChannelRequest.builder()
```

```
                    .enabled(enabled)
                    .build();

            UpdateSmsChannelRequest updateRequest =
 UpdateSmsChannelRequest.builder()
                    .smsChannelRequest(request)
                    .applicationId(appId)
                    .build();

            UpdateSmsChannelResponse result =
 client.updateSmsChannel(updateRequest);
            System.out.println("Channel state: " +
 result.smsChannelResponse().enabled());

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetSmsChannel em Referência AWS SDK for Java 2.x da API.

`GetUserEndpoints`

O código de exemplo a seguir mostra como usar `GetUserEndpoints`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <applicationId> <userId>

                Where:
                   applicationId - The ID of the Amazon Pinpoint application that
 has the endpoint.
                   userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String applicationId = args[0];
        String userId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllEndpoints(pinpoint, applicationId, userId);
        pinpoint.close();
    }

    public static void listAllEndpoints(PinpointClient pinpoint,
            String applicationId,
            String userId) {

        try {
```

```
            GetUserEndpointsRequest endpointsRequest =
 GetUserEndpointsRequest.builder()
                    .userId(userId)
                    .applicationId(applicationId)
                    .build();

            GetUserEndpointsResponse response =
 pinpoint.getUserEndpoints(endpointsRequest);
            List<EndpointResponse> endpoints = response.endpointsResponse().item();

            // Display the results.
            for (EndpointResponse endpoint : endpoints) {
                System.out.println("The channel type is: " +
 endpoint.channelType());
                System.out.println("The address is  " + endpoint.address());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetUserEndpoints em Referência AWS SDK for Java 2.x da API.

`SendMessages`

O código de exemplo a seguir mostra como usar `SendMessages`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessage {

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
 content.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
 for Java 2.x
```

```
        """;

        public static void main(String[] args) {
                final String usage = """

                                Usage:    <subject> <appId> <senderAddress>
<toAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
               toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendEmail(pinpoint, subject, senderAddress, toAddress);
        System.out.println("Email was sent");
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
```

```
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Envie uma mensagem de e-mail com cópia para outros destinatários.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import  software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
```

```
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
 for Java 2.x

        """;
    public static void main(String[] args) {
        final String usage = """

            Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified in
 Amazon Pinpoint in the region you're using to send email\s
               toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email\s
               ccAddress - The CC address.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        String ccAddress = args[3];
```

```
        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ArrayList<String> ccList = new ArrayList<>();
        ccList.add(ccAddress);
        sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .ccAddresses(ccAddresses)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");
```

```
        } catch (PinpointException e) {
            // Handle exception
            e.printStackTrace();
        }
    }
}
```

Envie uma mensagem SMS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessage {

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        public static String messageType = "TRANSACTIONAL";

        // The registered keyword associated with the originating short code.
        public static String registeredKeyword = "myKeyword";
```

```
        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        public static String senderId = "MySenderID";

        public static void main(String[] args) {
                final String usage = """

                                Usage:   <message> <appId> <originationNumber>
 <destinationNumber>\s

                                Where:
                                  message - The body of the message to send.
                                  appId - The Amazon Pinpoint project/application ID
 to use when you send this message.
                                  originationNumber - The phone number or short code
 that you specify has to be associated with your Amazon Pinpoint account. For best
 results, specify long codes in E.164 format (for example, +1-555-555-5654).
                                  destinationNumber - The recipient's phone number.
 For best results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).\s
                                  """;

                if (args.length != 4) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String message = args[0];
                String appId = args[1];
                String originationNumber = args[2];
                String destinationNumber = args[3];
                System.out.println("Sending a message");
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                sendSMSMessage(pinpoint, message, appId, originationNumber,
 destinationNumber);
                pinpoint.close();
        }
```

```
        public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                        String originationNumber,
                        String destinationNumber) {
                try {
                        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
                        AddressConfiguration addConfig =
AddressConfiguration.builder()
                                        .channelType(ChannelType.SMS)
                                        .build();

                        addressMap.put(destinationNumber, addConfig);
                        SMSMessage smsMessage = SMSMessage.builder()
                                        .body(message)
                                        .messageType(messageType)
                                        .originationNumber(originationNumber)
                                        .senderId(senderId)
                                        .keyword(registeredKeyword)
                                        .build();

                        // Create a DirectMessageConfiguration object.
                        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
                                        .smsMessage(smsMessage)
                                        .build();

                        MessageRequest msgReq = MessageRequest.builder()
                                        .addresses(addressMap)
                                        .messageConfiguration(direct)
                                        .build();

                        // create a SendMessagesRequest object
                        SendMessagesRequest request = SendMessagesRequest.builder()
                                        .applicationId(appId)
                                        .messageRequest(msgReq)
                                        .build();

                        SendMessagesResponse response =
pinpoint.sendMessages(request);
                        MessageResponse msg1 = response.messageResponse();
                        Map map1 = msg1.result();

                        // Write out the result of sendMessage.
```

```
                        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

Envie mensagens SMS em lote.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";
```

```
    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
    public static String senderId = "MySenderID";

    public static void main(String[] args) {
        final String usage = """

                Usage:   <message> <appId> <originationNumber> <destinationNumber>
 <destinationNumber1>\s

                Where:
                  message - The body of the message to send.
                  appId - The Amazon Pinpoint project/application ID to use when you
 send this message.
                  originationNumber - The phone number or short code that you
 specify has to be associated with your Amazon Pinpoint account. For best results,
 specify long codes in E.164 format (for example, +1-555-555-5654).
                  destinationNumber - The recipient's phone number.  For best
 results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).
                  destinationNumber1 - The second recipient's phone number.  For
 best results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String appId = args[1];
        String originationNumber = args[2];
        String destinationNumber = args[3];
        String destinationNumber1 = args[4];
        System.out.println("Sending a message");
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
        pinpoint.close();
    }

    public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                                      String originationNumber,
                                      String destinationNumber, String
destinationNumber1) {
        try {
            Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
            AddressConfiguration addConfig = AddressConfiguration.builder()
                    .channelType(ChannelType.SMS)
                    .build();

            // Add an entry to the Map object for each number to whom you want to
send a
            // message.
            addressMap.put(destinationNumber, addConfig);
            addressMap.put(destinationNumber1, addConfig);
            SMSMessage smsMessage = SMSMessage.builder()
                    .body(message)
                    .messageType(messageType)
                    .originationNumber(originationNumber)
                    .senderId(senderId)
                    .keyword(registeredKeyword)
                    .build();

            // Create a DirectMessageConfiguration object.
            DirectMessageConfiguration direct = DirectMessageConfiguration.builder()
                    .smsMessage(smsMessage)
                    .build();

            MessageRequest msgReq = MessageRequest.builder()
                    .addresses(addressMap)
                    .messageConfiguration(direct)
                    .build();

            // Create a SendMessagesRequest object.
            SendMessagesRequest request = SendMessagesRequest.builder()
                    .applicationId(appId)
```

```
                    .messageRequest(msgReq)
                    .build();

            SendMessagesResponse response = pinpoint.sendMessages(request);
            MessageResponse msg1 = response.messageResponse();
            Map map1 = msg1.result();

            // Write out the result of sendMessage.
            map1.forEach((k, v) -> System.out.println((k + ":" + v)));

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SendMessages em Referência AWS SDK for Java 2.x da API.

`UpdateEndpoint`

O código de exemplo a seguir mostra como usar `UpdateEndpoint`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
```

```
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                  appId - The ID of the application to create an endpoint for.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        EndpointResponse response = createEndpoint(pinpoint, appId);
        System.out.println("Got Endpoint: " + response.id());
```

```
        pinpoint.close();
    }

    public static EndpointResponse createEndpoint(PinpointClient client, String
appId) {
        String endpointId = UUID.randomUUID().toString();
        System.out.println("Endpoint ID: " + endpointId);

        try {
            EndpointRequest endpointRequest = createEndpointRequestData();
            UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .endpointRequest(endpointRequest)
                    .build();

            UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
            System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

            GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
            System.out.println(getEndpointResponse.endpointResponse().address());

System.out.println(getEndpointResponse.endpointResponse().channelType());

System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
            System.out.println(getEndpointResponse.endpointResponse().requestId());
            System.out.println(getEndpointResponse.endpointResponse().user());

            return getEndpointResponse.endpointResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return null;
    }

    private static EndpointRequest createEndpointRequestData() {
        try {
            List<String> favoriteTeams = new ArrayList<>();
            favoriteTeams.add("Lakers");
            favoriteTeams.add("Warriors");
            HashMap<String, List<String>> customAttributes = new HashMap<>();
            customAttributes.put("team", favoriteTeams);

            EndpointDemographic demographic = EndpointDemographic.builder()
                    .appVersion("1.0")
                    .make("apple")
                    .model("iPhone")
                    .modelVersion("7")
                    .platform("ios")
                    .platformVersion("10.1.1")
                    .timezone("America/Los_Angeles")
                    .build();

            EndpointLocation location = EndpointLocation.builder()
                    .city("Los Angeles")
                    .country("US")
                    .latitude(34.0)
                    .longitude(-118.2)
                    .postalCode("90068")
                    .region("CA")
                    .build();

            Map<String, Double> metrics = new HashMap<>();
            metrics.put("health", 100.00);
            metrics.put("luck", 75.00);

            EndpointUser user = EndpointUser.builder()
                    .userId(UUID.randomUUID().toString())
                    .build();

            DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
"Z" to indicate UTC, no timezone
                                                                            // offset
            String nowAsISO = df.format(new Date());
```

```
            return EndpointRequest.builder()
                    .address(UUID.randomUUID().toString())
                    .attributes(customAttributes)
                    .channelType("APNS")
                    .demographic(demographic)
                    .effectiveDate(nowAsISO)
                    .location(location)
                    .metrics(metrics)
                    .optOut("NONE")
                    .requestId(UUID.randomUUID().toString())
                    .user(user)
                    .build();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Para obter detalhes da API, consulte UpdateEndpoint em Referência AWS SDK for Java 2.x da API.

Exemplos da API SMS and Voice do Amazon Pinpoint usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS SDK for Java 2.x de SMS e voz do Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendVoiceMessage`

O código de exemplo a seguir mostra como usar `SendVoiceMessage`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";
```

```
    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";

    // The content of the message. This example uses SSML to customize and control
    // certain aspects of the message, such as by adding pauses and changing
    // phonation. The message can't contain any line breaks.
    static final String ssmlMessage = "<speak>This is a test message sent from "
            + "<emphasis>Amazon Pinpoint</emphasis> "
            + "using the <break strength='weak'/>AWS "
            + "SDK for Java. "
            + "<amazon:effect phonation='soft'>Thank "
            + "you for listening.</amazon:effect></speak>";

    public static void main(String[] args) {

        final String usage = """
                Usage:   <originationNumber> <destinationNumber>\s

                Where:
                  originationNumber - The phone number or short code that you
specify has to be associated with your Amazon Pinpoint account. For best results,
specify long codes in E.164 format (for example, +1-555-555-5654).
                  destinationNumber - The recipient's phone number.  For best
results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
        String originationNumber = args[0];
        String destinationNumber = args[1];
        System.out.println("Sending a voice message");

        // Set the content type to application/json.
        List<String> listVal = new ArrayList<>();
        listVal.add("application/json");
        Map<String, List<String>> values = new HashMap<>();
        values.put("Content-Type", listVal);
```

```
        ClientOverrideConfiguration config2 = ClientOverrideConfiguration.builder()
                .headers(values)
                .build();

        PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
                .overrideConfiguration(config2)
                .region(Region.US_EAST_1)
                .build();

        sendVoiceMsg(client, originationNumber, destinationNumber);
        client.close();
    }

    public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
 originationNumber,
                                    String destinationNumber) {
        try {
            SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                    .languageCode(languageCode)
                    .text(ssmlMessage)
                    .voiceId(voiceName)
                    .build();

            VoiceMessageContent content = VoiceMessageContent.builder()
                    .ssmlMessage(ssmlMessageType)
                    .build();

            SendVoiceMessageRequest voiceMessageRequest =
 SendVoiceMessageRequest.builder()
                    .destinationPhoneNumber(destinationNumber)
                    .originationPhoneNumber(originationNumber)
                    .content(content)
                    .build();

            client.sendVoiceMessage(voiceMessageRequest);
            System.out.println("The message was sent successfully.");

        } catch (PinpointSmsVoiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SendVoiceMessage em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Polly usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Polly.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DescribeVoices`

O código de exemplo a seguir mostra como usar `DescribeVoices`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                    .languageCode("en-US")
                    .build();

            DescribeVoicesResponse enUsVoicesResult =
 polly.describeVoices(voicesRequest);
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
 myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte DescribeVoices em Referência AWS SDK for Java 2.x da API.

`ListLexicons`

O código de exemplo a seguir mostra como usar `ListLexicons`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();
```

```
        listLexicons(polly);
        polly.close();
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest = ListLexiconsRequest.builder()
                    .build();

            ListLexiconsResponse listLexiconsResult =
 client.listLexicons(listLexiconsRequest);
            List<LexiconDescription> lexiconDescription =
 listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
 lexDescription.name());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListLexicons em Referência AWS SDK for Java 2.x da API.

`SynthesizeSpeech`

O código de exemplo a seguir mostra como usar `SynthesizeSpeech`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
 built this working demo " +
            " of Amazon Polly in Java Version 2. Have fun building voice enabled
 apps with Amazon Polly (that's me!), and always "
            +
            " look at the AWS website for tips and tricks on using Amazon Polly and
 other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        talkPolly(polly);
        polly.close();
    }
```

```
   public static void talkPolly(PollyClient polly) {
       try {
           DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
                   .engine("standard")
                   .build();

           DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
           Voice voice = describeVoicesResult.voices().stream()
                   .filter(v -> v.name().equals("Joanna"))
                   .findFirst()
                   .orElseThrow(() -> new RuntimeException("Voice not found"));
           InputStream stream = synthesize(polly, SAMPLE, voice, OutputFormat.MP3);
           AdvancedPlayer player = new AdvancedPlayer(stream,

javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
           player.setPlayBackListener(new PlaybackListener() {
               public void playbackStarted(PlaybackEvent evt) {
                   System.out.println("Playback started");
                   System.out.println(SAMPLE);
               }

               public void playbackFinished(PlaybackEvent evt) {
                   System.out.println("Playback finished");
               }
           });

           // play it!
           player.play();

       } catch (PollyException | JavaLayerException | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static InputStream synthesize(PollyClient polly, String text, Voice
voice, OutputFormat format)
           throws IOException {
       SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
               .text(text)
               .voiceId(voice.id())
               .outputFormat(format)
```

```
                .build();

        ResponseInputStream<SynthesizeSpeechResponse> synthRes =
 polly.synthesizeSpeech(synthReq);
        return synthRes;
    }
}
```

- Para obter detalhes da API, consulte SynthesizeSpeech em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda

- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon RDS usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon RDS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon RDS

O exemplo de código a seguir mostra como começar a usar o Amazon RDS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
```

```
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK for Java 2.x .

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de banco de dados e definir os valores dos parâmetros.
- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.
- Exclua a instância e o grupo de parâmetros.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute várias operações.

```
import com.google.gson.Gson;
```

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySql database and uses
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
 <dbName> <dbSnapshotIdentifier> <secretName>

                Where:
```

```
                    dbGroupName - The database group name.\s
                    dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
                    dbInstanceIdentifier - The database instance identifier\s
                    dbName - The database name.\s
                    dbSnapshotIdentifier - The snapshot identifier.\s
                    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"
                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceIdentifier = args[2];
        String dbName = args[3];
        String dbSnapshotIdentifier = args[4];
        String secretName = args[5];

        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String masterUsername = user.getUsername();
        String masterUserPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon RDS example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbParameterGroups(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbParameters(rdsClient, dbGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBParas(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated value");
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get a list of micro instance classes available for
the selected engine");
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "9. Create an RDS database instance that contains a MySql database
and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
                masterUserPassword);
        System.out.println("The ARN of the new database is " + dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("10. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();
```

```
        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
```

```
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Delete the DB instance.
    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the snapshot instance is available.
    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
            String dbSnapshotIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
```

```
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
                List<DBSnapshot> snapshotList = response.dbSnapshots();
                for (DBSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
```

```
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Create a database instance and return the ARN of the database.
    public static String createDatabaseInstance(RdsClient rdsClient,
            String dbGroupName,
            String dbInstanceIdentifier,
```

```
            String dbName,
            String userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }

        return "";
    }

    // Get a list of micro instances.
    public static void getMicroInstances(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("mysql")
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
```

```
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
{
                System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
                System.out.println("The engine description is " +
dbInstanceOption.engine());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
```

```
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
```

```
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
```

```
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Ações

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 *
 */
```

```
public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbName> <secretName>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    dbName - The database name.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials."
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
 user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)
```

```
.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
               .build();
   }

   private static String getSecretValues(String secretName) {
       SecretsManagerClient secretClient = getSecretClient();
       GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
               .secretId(secretName)
               .build();

       GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
       return valueResponse.secretString();
   }

   public static void createDatabaseInstance(RdsClient rdsClient,
           String dbInstanceIdentifier,
           String dbName,
           String userName,
           String userPassword) {

       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .allocatedStorage(100)
               .dbName(dbName)
               .engine("mysql")
               .dbInstanceClass("db.t3.medium") // Updated to a supported class
               .engineVersion("8.0.32")         // Updated to a supported version
               .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
               .masterUsername(userName)
               .masterUserPassword(userPassword)
               .build();

           CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
           System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            // Loop until the cluster is ready.
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true;
                    else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK for Java 2.x .

`CreateDBParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBParameterGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
       try {
           CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CreateDBParameterGroup em Referência AWS SDK for Java 2.x da API.

`CreateDBSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBSnapshot`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Create an Amazon RDS snapshot.
   public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
       try {
           CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .dbSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
           System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK for Java 2.x .

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
```

```
                .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK for Java 2.x .

`DeleteDBParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBParameterGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Delete the parameter group after database has been deleted.
   // An exception is thrown if you attempt to delete the para group while database
   // exists.
   public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(dbARN) == 0) {
                       System.out.println(dbARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
                   index++;
               }
           }

           // Delete the para group.
           DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .build();

           rdsClient.deleteDBParameterGroup(parameterGroupRequest);
           System.out.println(dbGroupName + " was deleted.");
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DeleteDBParameterGroup em Referência AWS SDK for Java 2.x da API.

`DescribeAccountAttributes`

O código de exemplo a seguir mostra como usar `DescribeAccountAttributes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
```

```
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
 rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeAccountAttributes em Referência AWS SDK for Java 2.x da API.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .defaultOnly(true)
                   .engine("mysql")
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions em Referência AWS SDK for Java 2.x da API.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
```

```
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK for Java 2.x .

`DescribeDBParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBParameterGroups`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeDbParameterGroups(RdsClient rdsClient, String
 dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
 DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
 rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
```

```
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBParameterGroups em Referência AWS SDK for Java 2.x da API.

`DescribeDBParameters`

O código de exemplo a seguir mostra como usar `DescribeDBParameters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
```

```
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK for Java 2.x .

`GenerateRDSAuthToken`

O código de exemplo a seguir mostra como usar `GenerateRDSAuthToken`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a RdsUtilities classe para gerar um token de autenticação.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <masterUsername>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUsername - The master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
 masterUsername);
        System.out.println("The token response is " + token);
    }

    public static String getAuthToken(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
```

```
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
 GenerateAuthenticationTokenRequest.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .username(masterUsername)
                    .port(3306)
                    .hostname(dbInstanceIdentifier)
                    .build();

            return utilities.generateAuthenticationToken(tokenRequest);

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte GenerateRDSAuthToken em Referência AWS SDK for Java 2.x da API.

`ModifyDBInstance`

O código de exemplo a seguir mostra como usar `ModifyDBInstance`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUserPassword - The updated password that corresponds to
 the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
 ModifyDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .publiclyAccessible(true)
```

```
                    .masterUserPassword(masterUserPassword)
                    .build();

            ModifyDbInstanceResponse instanceResponse =
 rdsClient.modifyDBInstance(modifyDbInstanceRequest);
            System.out.print("The ARN of the modified database is: " +
 instanceResponse.dbInstance().dbInstanceArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em ModifyDBInstance na Referência da API AWS SDK for Java 2.x .

`ModifyDBParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBParameterGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
```

```
            ModifyDbParameterGroupRequest groupRequest =
 ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
 rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
 response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ModifyDBParameterGroup em Referência AWS SDK for Java 2.x da API.

`RebootDBInstance`

O código de exemplo a seguir mostra como usar `RebootDBInstance`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
 RebootDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            RebootDbInstanceResponse instanceResponse =
 rdsClient.rebootDBInstance(rebootDbInstanceRequest);
            System.out.print("The database " +
 instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em RebootDBInstance na Referência da API AWS SDK for Java 2.x .

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
 APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();
```

```
        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                 PreparedStatement statement = connection.prepareStatement("SELECT ?
 + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();
```

```
        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

Exemplos do Amazon RDS Data Service usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon RDS Data Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon Redshift usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Redshift.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon Redshift

O exemplo de código a seguir mostra como começar a usar o Amazon Redshift.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloRedshift {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        listClustersPaginator(redshiftClient);
    }

    public static void listClustersPaginator(RedshiftClient redshiftClient) {
```

```
        DescribeClustersIterable clustersIterable =
 redshiftClient.describeClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.clusters().stream())
            .forEach(cluster -> System.out
                .println(" Cluster identifier: " + cluster.clusterIdentifier() + "
 status = " + cluster.clusterStatus()));
    }
}
```

- Para obter detalhes da API, consulte DescribeClusters em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um cluster do Redshift.
- Listar bancos de dados no cluster.
- Criar uma tabela chamada Filmes.
- Preencher a tabela Filmes.
- Consultar a tabela Filmes por ano.
- Modificar o cluster do Redshift.
- Excluir o cluster do Amazon Redshift.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstra os recursos do Amazon Redshift.

```
import com.example.redshift.User;
import com.google.gson.Gson;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.model.ClusterAlreadyExistsException;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *
 *  This example requires an AWS Secrets Manager secret that contains the
 *  database credentials. If you do not create a
 *  secret that specifies user name and password, this example will not work. For
 details, see:
 *
 *  https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
```

```
 * 6. Inserts a specified number of records into the "Movies" table by reading the
 Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
 * 11. If confirmed, deletes the specified Redshift cluster.
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftScenario.class);

    static RedshiftActions redshiftActions = new RedshiftActions();
    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <jsonFilePath> <secretName>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
 file in ../../../resources/sample_files/movies.json)
                secretName - The name of the secret that belongs to Secret Manager
 that stores the user name and password used in this scenario.
            """;

        if (args.length != 2) {
            logger.info(usage);
            return;
        }

        String jsonFilePath = args[0];
        String secretName = args[1];
        Scanner scanner = new Scanner(System.in);
        logger.info(DASHES);
        logger.info("Welcome to the Amazon Redshift SDK Basics scenario.");
        logger.info("""
            This Java program demonstrates how to interact with Amazon Redshift by
 using the AWS SDK for Java (v2).\s
            Amazon Redshift is a fully managed, petabyte-scale data warehouse
 service hosted in the cloud.
```

```
            The program's primary functionalities include cluster creation,
 verification of cluster readiness,\s
            list databases, table creation, data population within the table, and
 execution of SQL statements.
            Furthermore, it demonstrates the process of querying data from the Movie
 table.\s

            Upon completion of the program, all AWS resources are cleaned up.
            """);

        logger.info("Lets get started...");
        logger.info("""
            First, we will retrieve the user name and password from Secrets Manager.

            Using Amazon Secrets Manager to store Redshift credentials provides
 several security benefits.
            It allows you to securely store and manage sensitive information, such
 as passwords, API keys, and
            database credentials, without embedding them directly in your
 application code.

            More information can be found here:

            https://docs.aws.amazon.com/secretsmanager/latest/userguide/
integrating_how-services-use-secrets_RS.html
            """);
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(user, scanner, jsonFilePath);
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }
    }

    private static void runScenario(User user, Scanner scanner,  String
 jsonFilePath) throws Throwable {
        String databaseName = "dev";
```

```
        System.out.println(DASHES);
        logger.info("Create a Redshift Cluster");
        logger.info("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
        logger.info("Enter a cluster id value or accept the default by hitting Enter
(default is redshift-cluster-movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
        try {
            CompletableFuture<CreateClusterResponse> future =
redshiftActions.createClusterAsync(clusterId, user.getUserName(),
user.getUserPassword());
            CreateClusterResponse response = future.join();
            logger.info("Cluster successfully created. Cluster Identifier {} ",
response.cluster().clusterIdentifier());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof ClusterAlreadyExistsException) {
                logger.info("The Cluster {} already exists. Moving on...",
clusterId);
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Wait until {} is available.", clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.waitForClusterReadyAsync(clusterId);
            future.join();
            logger.info("Cluster is ready!");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftException redshiftEx) {
                logger.info("Redshift error occurred: Error message: {}, Error code
{}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        String databaseInfo = """
            When you created $clusteridD, the dev database is created by default and
 used in this scenario.\s

            To create a custom database, you need to have a CREATEDB privilege.\s
            For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
           """.replace("$clusteridD", clusterId);

        logger.info(databaseInfo);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("List databases in {} ",clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.listAllDatabasesAsync(clusterId, user.getUserName(), "dev");
            future.join();
            logger.info("Databases listed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.error("Redshift Data error occurred: {} Error code: {}",
 redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.error("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will create a table named Movies.");
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<ExecuteStatementResponse> future =
redshiftActions.createTableAsync(clusterId, databaseName, user.getUserName());
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Populate the Movies table using the Movies.json file.");
        logger.info("Specify the number of records you would like to add to the
Movies Table.");
        logger.info("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            logger.info("Enter a value: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a value between 50 and
200.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        try {
            redshiftActions.popTableAsync(clusterId, databaseName,
user.getUserName(), jsonFilePath, numRecords).join();  // Wait for the operation to
complete
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
```

```
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            logger.info("Enter a year: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a valid year between 2012
and 2014.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id;
        try {
            CompletableFuture<String> future =
redshiftActions.queryMoviesByYearAsync(databaseName, user.getUserName(), movieYear,
clusterId);
            id = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        logger.info("The identifier of the statement is " + id);
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Void> future =
redshiftActions.checkStatementAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = redshiftActions.getResultsAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will modify the Redshift cluster.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ModifyClusterResponse> future =
redshiftActions.modifyClusterAsync(clusterId);;
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
```

```
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Would you like to delete the Amazon Redshift cluster? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete {} ", clusterId);
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<DeleteClusterResponse> future =
redshiftActions.deleteRedshiftClusterAsync(clusterId);;
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof RedshiftDataException redshiftEx) {
                    logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: {}",
rt.getMessage());
                }
                throw cause;
            }
        } else {
            logger.info("The {}  was not deleted", clusterId);
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the Amazon Redshift SDK Basics scenario.");
        logger.info(DASHES);
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_EAST_1;
```

```
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    // Get the Amazon Redshift credentials from AWS Secrets Manager.
    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }
}
```

Uma classe de wrapper para métodos do SDK do Amazon Redshift.

```
public class RedshiftActions {

    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftActions.class);
    private static RedshiftDataAsyncClient redshiftDataAsyncClient;
```

```
    private static RedshiftAsyncClient redshiftAsyncClient;

    private static RedshiftAsyncClient getAsyncClient() {
        if (redshiftAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftAsyncClient = RedshiftAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftAsyncClient;
    }

    private static RedshiftDataAsyncClient getAsyncDataClient() {
        if (redshiftDataAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftDataAsyncClient = RedshiftDataAsyncClient.builder()
```

```
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftDataAsyncClient;
    }

    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
```

```
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
```

```
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }

    /**
```

```
    * Creates an asynchronous task to execute a SQL statement for creating a new
table.
    *
    * @param clusterId    the identifier of the Amazon Redshift cluster
    * @param databaseName the name of the database to create the table in
    * @param userName     the username to use for the database connection
    * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
    * @throws RuntimeException if there is an error creating the table
    */
   public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
       ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(userName)
           .database(databaseName)
           .sql("CREATE TABLE Movies (" +
               "id INT PRIMARY KEY, " +
               "title VARCHAR(100), " +
               "year INT)")
           .build();

       return getAsyncDataClient().executeStatement(createTableRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
               } else {
                   logger.info("Table created: Movies");
               }
           });
   }

   /**
    * Asynchronously pops a table from a JSON file.
    *
    * @param clusterId   the ID of the cluster
    * @param databaseName the name of the database
    * @param userName    the username
    * @param fileName    the name of the JSON file
    * @param number      the number of records to process
    * @return a CompletableFuture that completes with the number of records added
to the Movies table
```

```
    */
   public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
       return CompletableFuture.supplyAsync(() -> {
               try {
                   JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                   JsonNode rootNode = new ObjectMapper().readTree(parser);
                   Iterator<JsonNode> iter = rootNode.iterator();
                   return iter;
               } catch (IOException e) {
                   throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
               }
           }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   logger.info("Error {} ", exception.getMessage());
               } else {
                   logger.info("{} records were added to the Movies table." ,
result);
               }
           });
   }

   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
                       break;
                   JsonNode currentNode = iter.next();
                   int year = currentNode.get("year").asInt();
                   String title = currentNode.get("title").asText();

                   // Use SqlParameter to avoid SQL injection.
                   List<SqlParameter> parameterList = new ArrayList<>();
                   String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                   SqlParameter idParam = SqlParameter.builder()
                       .name("id")
```

```
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();

                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();

                    getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }

    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
```

```
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }

    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the results
     * @return a {@link CompletableFuture} that completes when the statement result
has been processed
```

```
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
               }

               // Extract and print the field values using streams if the response
is valid.
               response.records().stream()
                   .flatMap(List::stream)
                   .map(Field::stringValue)
                   .filter(value -> value != null)
                   .forEach(value -> System.out.println("The Movie title field is "
+ value));

               return response;
           }).thenAccept(response -> {
               // Optionally add more logic here if needed after handling the
response
           });
   }


   /**
    * Asynchronously queries movies by a given year from a Redshift database.
    *
    * @param database    the name of the database to query
    * @param dbUser      the user to connect to the database with
    * @param year        the year to filter the movies by
    * @param clusterId   the identifier of the Redshift cluster to connect to
    * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
    */
   public CompletableFuture<String> queryMoviesByYearAsync(String database,
```

```
                                                                    String dbUser,
                                                                    int year,
                                                                    String clusterId)
{

       String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
       SqlParameter yearParam = SqlParameter.builder()
           .name("year")
           .value(String.valueOf(year))
           .build();

       ExecuteStatementRequest statementRequest = ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .database(database)
           .dbUser(dbUser)
           .parameters(yearParam)
           .sql(sqlStatement)
           .build();

       return CompletableFuture.supplyAsync(() -> {
           try {
               ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
               return response.id();
           } catch (RedshiftDataException e) {
               throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
           }
       }).exceptionally(exception -> {
           logger.info("Error: {}", exception.getMessage());
           return "";
       });
   }

   /**
    * Modifies an Amazon Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the cluster to be modified
    * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
    */
   public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
```

```
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified and
has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
                }
            });
    }

    /**
     * Deletes a Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
     */
    public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
        DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
```

```
                        logger.info("Unexpected error: {}", exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cluster.

```
    /**
```

```
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }
```

- Para obter detalhes da API, consulte CreateCluster em Referência AWS SDK for Java 2.x da API.

`DeleteCluster`

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir o cluster.

```
   /**
    * Deletes a Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the Redshift cluster to be deleted
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
    */
   public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
       DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .skipFinalClusterSnapshot(true)
           .build();

       return getAsyncClient().deleteCluster(deleteClusterRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   // Handle exceptions
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {}", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {}", exception.getMessage());
                   }
               } else {
                   // Handle successful response
                   logger.info("The status is {}",
response.cluster().clusterStatus());
               }
           });
   }
```

- Para obter detalhes da API, consulte DeleteCluster em Referência AWS SDK for Java 2.x da API.

`DescribeClusters`

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Descrever o cluster.

```
    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
```

```
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }
```

- Para obter detalhes da API, consulte DescribeClusters em Referência AWS SDK for Java 2.x da API.

`DescribeStatement`

O código de exemplo a seguir mostra como usar `DescribeStatement`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
    *
    * @param sqlId the ID of the SQL statement to check
    * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
    */
   public CompletableFuture<Void> checkStatementAsync(String sqlId) {
       DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
           .id(sqlId)
           .build();

       return getAsyncDataClient().describeStatement(statementRequest)
           .thenCompose(response -> {
               String status = response.statusAsString();
               logger.info("... Status: {} ", status);

               if ("FAILED".equals(status)) {
                   throw new RuntimeException("The Query Failed. Ending program");
               } else if ("FINISHED".equals(status)) {
                   return CompletableFuture.completedFuture(null);
               } else {
                   // Sleep for 1 second and recheck status
                   return CompletableFuture.runAsync(() -> {
                       try {
                           TimeUnit.SECONDS.sleep(1);
                       } catch (InterruptedException e) {
                           throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                       }
                   }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
```

```
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }
```

- Para obter detalhes da API, consulte DescribeStatement em Referência AWS SDK for Java 2.x da API.

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Executa uma instrução SQL para criar uma tabela de banco de dados.

```
    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
     * @throws RuntimeException if there is an error creating the table
     */
    public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
```

```
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies (" +
                "id INT PRIMARY KEY, " +
                "title VARCHAR(100), " +
                "year INT)")
            .build();

        return getAsyncDataClient().executeStatement(createTableRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }
```

Executa uma instrução SQL para inserir dados em uma tabela de banco de dados.

```
    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records added
to the Movies table
     */
    public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
        return CompletableFuture.supplyAsync(() -> {
                try {
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
```

```
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
                }
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
    }

    private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
        return CompletableFuture.supplyAsync(() -> {
            int t = 0;
            try {
                while (iter.hasNext()) {
                    if (t == number)
                        break;
                    JsonNode currentNode = iter.next();
                    int year = currentNode.get("year").asInt();
                    String title = currentNode.get("title").asText();

                    // Use SqlParameter to avoid SQL injection.
                    List<SqlParameter> parameterList = new ArrayList<>();
                    String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                    SqlParameter idParam = SqlParameter.builder()
                        .name("id")
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();
```

```
                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();

                    getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }
```

Executa uma instrução SQL para consultar uma tabela de banco de dados.

```
    /**
     * Asynchronously queries movies by a given year from a Redshift database.
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
```

```
                                                                    String dbUser,
                                                                    int year,
                                                                    String clusterId)
{

        String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest = ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK for Java 2.x da API.

`GetStatementResult`

O código de exemplo a seguir mostra como usar `GetStatementResult`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Verifique o resultado da instrução

```
   /**
    * Asynchronously retrieves the results of a statement execution.
    *
    * @param statementId the ID of the statement for which to retrieve the results
    * @return a {@link CompletableFuture} that completes when the statement result
has been processed
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
               }

               // Extract and print the field values using streams if the response
is valid.
               response.records().stream()
                   .flatMap(List::stream)
                   .map(Field::stringValue)
                   .filter(value -> value != null)
                   .forEach(value -> System.out.println("The Movie title field is "
+ value));

               return response;
           }).thenAccept(response -> {
```

```
                // Optionally add more logic here if needed after handling the
response
            });
    }
```

- Para obter detalhes da API, consulte GetStatementResult em Referência AWS SDK for Java 2.x da API.

`ListDatabases`

O código de exemplo a seguir mostra como usar `ListDatabases`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
```

```
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }
```

- Para obter detalhes da API, consulte ListDatabases em Referência AWS SDK for Java 2.x da API.

`ModifyCluster`

O código de exemplo a seguir mostra como usar `ModifyCluster`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Modificar um cluster.

```
    /**
     * Modifies an Amazon Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the cluster to be modified
     * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
     */
    public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
```

```
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified and
has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
                }
            });
    }
```

- Para obter detalhes da API, consulte ModifyCluster em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação Web para rastrear dados do Amazon Redshift

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho usando um banco de dados do Amazon Redshift.

SDK para Java 2.x

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- banco de dados de origem
- Amazon SES

Exemplos do Amazon Rekognition usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Rekognition.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CompareFaces`

O código de exemplo a seguir mostra como usar `CompareFaces`.

Para obter mais informações, consulte Comparação de faces em imagens.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import software.amazon.awssdk.core.SdkBytes;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```

```
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

   /**
    * Compares two faces from images stored in an Amazon S3 bucket using AWS
Rekognition.
    *
    * <p>This method takes two image keys from an S3 bucket and compares the faces
within them.
    * It prints out the confidence level of matched faces and reports the number of
unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the S3
bucket.
    * @param targetKey   The object key (file path) for the target image in the S3
bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
        try {
            Float similarityThreshold = 70F;
            S3Object s3ObjectSource = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceKey)
                    .build();

            Image sourceImage = Image.builder()
                    .s3Object(s3ObjectSource)
                    .build();

            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(targetKey)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            CompareFacesRequest facesRequest = CompareFacesRequest.builder()
```

```
                    .sourceImage(sourceImage)
                    .targetImage(targetImage)
                    .similarityThreshold(similarityThreshold)
                    .build();

            // Compare the two images.
            CompareFacesResponse compareFacesResult =
 rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();

            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");
            }

            List<ComparedFace> unmatchedFaces = compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte CompareFaces em Referência AWS SDK for Java 2.x da API.

`CreateCollection`

O código de exemplo a seguir mostra como usar `CreateCollection`.

Para obter mais informações, consulte Criar uma coleção.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = """

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
```

```
        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
     * @param rekClient    the Amazon Rekognition client used to interact with the
 Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateCollection em Referência AWS SDK for Java 2.x da API.

`DeleteCollection`

O código de exemplo a seguir mostra como usar `DeleteCollection`.

Para obter mais informações, consulte Excluir uma coleção.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId>\s

            Where:
                collectionId - The id of the collection to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
```

```
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteCollection em Referência AWS SDK for Java 2.x da API.

`DeleteFaces`

O código de exemplo a seguir mostra como usar `DeleteFaces`.

Para obter mais informações, consulte Excluir faces de uma coleção.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <faceId>\s

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
```

```
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should be
 deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteFaces em Referência AWS SDK for Java 2.x da API.

`DescribeCollection`

O código de exemplo a seguir mostra como usar `DescribeCollection`.

Para obter mais informações, consulte Descrever uma coleção.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
```

```
                .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
 request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
 collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
 collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
 DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
 describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
 describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeCollection em Referência AWS SDK for Java 2.x da API.

`DetectFaces`

O código de exemplo a seguir mostra como usar `DetectFaces`.

Para obter mais informações, consulte Detectar faces em uma imagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = """

            Usage:   <bucketName> <sourceImage>

            Where:
                bucketName = The name of the Amazon S3 bucket where the source image
 is stored.
                sourceImage - The name of the source image file in the Amazon S3
 bucket. (for example, pic1.png).\s
            """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with the
Rekognition service.
     * @param bucketName   The name of the Amazon S3 bucket where the source image
is stored.
     * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
     */
    public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image targetImage = Image.builder()
                .s3Object(s3ObjectTarget)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(targetImage)
                .build();

            DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
```

```
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between "
                        + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                        + " years old.");

                System.out.println("There is a smile : " +
 face.smile().value().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectFaces em Referência AWS SDK for Java 2.x da API.

`DetectLabels`

O código de exemplo a seguir mostra como usar `DetectLabels`.

Para obter mais informações, consulte Detectar rótulos em uma imagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
 stored
                sourceImage - The name of the image file (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the Amazon
 Rekognition service.
     *
```

```
     * @param rekClient     the Amazon Rekognition client used to make the detection
 request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                    .image(souImage)
                    .maxLabels(10)
                    .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label : labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectLabels em Referência AWS SDK for Java 2.x da API.

`DetectModerationLabels`

O código de exemplo a seguir mostra como usar `DetectModerationLabels`.

Para obter mais informações, consulte Detectar imagens impróprias.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = """
            Usage:  <bucketName>  <sourceImage>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
                sourceImage - The name of the image (for example, pic1.png).\s
            """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image detection
process
     */
    public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                    .image(targetImage)
                    .minConfidence(60F)
                    .build();
```

```
            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                    .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
 moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                        + "\n Confidence: " + label.confidence().toString() + "%"
                        + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectModerationLabels em Referência AWS SDK for Java 2.x da API.

`DetectText`

O código de exemplo a seguir mostra como usar `DetectText`.

Para obter mais informações, consulte Detectar texto em uma imagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored\n"
 +
            "   sourceImage - The path to the image that contains text (for example,
 pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
 Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
```

```
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
 Rekognition API
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " + text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DetectText em Referência AWS SDK for Java 2.x da API.

`IndexFaces`

O código de exemplo a seguir mostra como usar `IndexFaces`.

Para obter mais informações, consulte Adicionar faces a uma coleção.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <sourceImage> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
     * @param bucketName    the name of the Amazon S3 bucket containing the image
     * @param sourceImage   the name of the image file to add to the collection
     * @throws RekognitionException if there is an error while interacting with the
Amazon Rekognition service
     */
    public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                    .collectionId(collectionId)
                    .image(targetImage)
                    .maxFaces(1)
                    .qualityFilter(QualityFilter.AUTO)
                    .detectionAttributes(Attribute.DEFAULT)
                    .build();

            IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
            System.out.println("Results for the image");
```

```
            System.out.println("\n Faces indexed:");
            List<FaceRecord> faceRecords = facesResponse.faceRecords();
            for (FaceRecord faceRecord : faceRecords) {
                System.out.println("  Face ID: " + faceRecord.face().faceId());
                System.out.println("  Location:" +
 faceRecord.faceDetail().boundingBox().toString());
            }

            List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
            System.out.println("Faces not indexed:");
            for (UnindexedFace unindexedFace : unindexedFaces) {
                System.out.println("  Location:" +
 unindexedFace.faceDetail().boundingBox().toString());
                System.out.println("  Reasons:");
                for (Reason reason : unindexedFace.reasons()) {
                    System.out.println("Reason:  " + reason);
                }
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte IndexFaces em Referência AWS SDK for Java 2.x da API.

`ListCollections`

O código de exemplo a seguir mostra como usar `ListCollections`.

Para obter mais informações, consulte Listar coleções.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
                    .maxResults(10)
                    .build();

            ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListCollections em Referência AWS SDK for Java 2.x da API.

ListFaces

O código de exemplo a seguir mostra como usar `ListFaces`.

Para obter mais informações, consulte Listar faces em uma coleção.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <collectionId>

                Where:
                   collectionId - The name of the collection.\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                    .collectionId(collectionId)
                    .maxResults(10)
                    .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
face.confidence());
                System.out.println("The face Id value is " + face.faceId());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte ListFaces em Referência AWS SDK for Java 2.x da API.

`RecognizeCelebrities`

O código de exemplo a seguir mostra como usar `RecognizeCelebrities`.

Para obter mais informações, consulte Reconhecer celebridades em uma imagem.

SDK para Java 2.x

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = """
                Usage:   <bucketName> <sourceImage>
```

```
                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
       }

        String bucketName = args[0];;
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source image
 is stored
     * @param sourceImage  the name of the source image file stored in the Amazon S3
 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
```

```
                    .build();

            RecognizeCelebritiesRequest request =
 RecognizeCelebritiesRequest.builder()
                    .image(souImage)
                    .build();

            RecognizeCelebritiesResponse result =
 rekClient.recognizeCelebrities(request);
            List<Celebrity> celebs = result.celebrityFaces();
            System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
            for (Celebrity celebrity : celebs) {
                System.out.println("Celebrity recognized: " + celebrity.name());
                System.out.println("Celebrity ID: " + celebrity.id());

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte RecognizeCelebrities em Referência AWS SDK for Java 2.x da API.

`SearchFaces`

O código de exemplo a seguir mostra como usar `SearchFaces`.

Para obter mais informações, consulte Pesquisar uma face (face ID).

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s

                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " + face.similarity());
                System.out.println();
            }
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SearchFaces em Referência AWS SDK for Java 2.x da API.

`SearchFacesByImage`

O código de exemplo a seguir mostra como usar `SearchFacesByImage`.

Para obter mais informações, consulte Pesquisar uma face (imagem).

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " + face.similarity());
```

```
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SearchFacesByImage em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

Detectar EPI em imagens

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar informações em vídeos

O exemplo de código a seguir mostra como:

- Iniciar trabalhos do Amazon Rekognition para detectar elementos como pessoas, objetos e texto em vídeos.
- Verificar o status do trabalho até que os trabalhos terminem.
- Visualizar a lista de elementos detectados por cada trabalho.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha resultados de celebridades a partir de um vídeo localizado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>
```

```
                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
                """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucket = args[0];
       String video = args[1];
       String topicArn = args[2];
       String roleArn = args[3];
       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

       NotificationChannel channel = NotificationChannel.builder()
               .snsTopicArn(topicArn)
               .roleArn(roleArn)
               .build();

       startCelebrityDetection(rekClient, channel, bucket, video);
       getCelebrityDetectionResults(rekClient);
       System.out.println("This example is done!");
       rekClient.close();
   }

   public static void startCelebrityDetection(RekognitionClient rekClient,
           NotificationChannel channel,
           String bucket,
           String video) {
       try {
           S3Object s3Obj = S3Object.builder()
                   .bucket(bucket)
                   .name(video)
```

```
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
                    .startCelebrityRecognition(recognitionRequest);
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCelebrityDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (recognitionResponse != null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                        .maxResults(10)
                        .build();
```

```
                // Wait until the job succeeds
                while (!finished) {
                    recognitionResponse =
 rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs =
 recognitionResponse.celebrities();
                for (CelebrityRecognition celeb : celebs) {
                    long seconds = celeb.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details = celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte rótulos em um vídeo por meio de uma operação de detecção de rótulos.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
```

```
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
```

```
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
```

```
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();
```

```
                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels = labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
```

```
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Detecte faces em um vídeo armazenado em um bucket do Amazon S3.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
```

```
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();
```

```
            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
```

```
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
```

```
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels = labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
```

```
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
 instance.confidence().toString());
                            System.out.println("        Bounding box: " +
 instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
 ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Detecte conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
```

```
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startModerationDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {

        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
```

```
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();

                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;
```

```
                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
 modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse != null &&
 modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte segmentos de sinal técnico e segmentos de detecção de tomada em um vídeo armazenado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
```

```
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startSegmentDetection(rekClient, channel, bucket, video);
        getSegmentResults(rekClient);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startSegmentDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();
```

```
            StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
                    .shotFilter(cueDetectionFilter)
                    .technicalCueFilter(technicalCueDetectionFilter)
                    .build();

            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                    .video(vidOb)
                    .filters(filters)
                    .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getSegmentResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
```

```
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }

                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
```

```
                        TechnicalCueSegment segmentCue =
 detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
 segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
 segmentShot.confidence().toString());
                    }

                    long seconds = detectedSegment.durationMillis();
                    System.out.println("\tDuration : " + seconds + " milliseconds");
                    System.out.println("\tStart time code: " +
 detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
 detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
 detectedSegment.durationSMPTE());
                    System.out.println();
                }

            } while (segDetectionResponse != null &&
 segDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte texto em um vídeo armazenado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
```

```
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
```

```
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
```

```
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
```

```
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels =
 textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
                    System.out.println("Confidence: " +
 detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " + detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
 detectedText.textDetection().parentId());
                    System.out.println("Type: " +
 detectedText.textDetection().type());
                    System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                    System.out.println();
                }

            } while (textDetectionResponse != null &&
 textDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte pessoas em um vídeo armazenado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
```

```
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
```

```
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
 personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
 detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
 personTrackingResult.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3

- Amazon SES
- Amazon SNS
- Amazon SQS

Exemplos para registro de domínios do Route 53 usando SDKs para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o registro de domínio AWS SDK for Java 2.x com o Route 53.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Registro de domínios do Olá, Route 53

O exemplo de código a seguir mostra como começar a usar o registro de domínio do Route 53.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code examples performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the “com” type
 * and displays the prices for Registration and Renewal.
 *
 */
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
                "    <hostedZoneId> \n\n" +
                "Where:\n" +
                "    hostedZoneId - The id value of an existing hosted zone. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
                .region(region)
                .build();
```

```
        System.out.println(DASHES);
        System.out.println("Invokes ListPrices for at least one domain type.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
 domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .maxItems(10)
                    .tld(domainType)
                    .build();

            ListPricesResponse response =
 route53DomainsClient.listPrices(pricesRequest);
            List<DomainPrice> prices = response.prices();
            for (DomainPrice pr : prices) {
                System.out.println("Name: " + pr.name());
                System.out.println(
                        "Registration: " + pr.registrationPrice().price() + " " +
 pr.registrationPrice().currency());
                System.out.println("Renewal: " + pr.renewalPrice().price() + " " +
 pr.renewalPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + " " +
 pr.transferPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + " " +
 pr.transferPrice().currency());
                System.out.println("Change Ownership: " +
 pr.changeOwnershipPrice().price() + " "
                        + pr.changeOwnershipPrice().currency());
                System.out.println(
                        "Restoration: " + pr.restorationPrice().price() + " " +
 pr.restorationPrice().currency());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListPrices em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Listar os domínios atuais e as operações do ano passado.
- Ver o faturamento do ano passado e os preços dos tipos de domínio.
- Receber sugestões de domínio.
- Verificar a disponibilidade e a transferibilidade de um domínio.
- Opcionalmente, solicitar o registro de um domínio.
- Obter os detalhes de uma operação.
- Opcionalmente, obtenha os detalhes de um domínio.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <domainType> <phoneNumber> <email> <domainSuggestion>
 <firstName> <lastName> <city>

                Where:
                    domainType - The domain type (for example, com).\s
                    phoneNumber - The phone number to use (for example,
 +91.9966564xxx)      email - The email address to use.      domainSuggestion - The
 domain suggestion (for example, findmy.accountants).\s
                    firstName - The first name to use to register a domain.\s
                    lastName -  The last name to use to register a domain.\s
                    city - the city to use to register a domain.\s
                    """;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        String phoneNumber = args[1];
```

```
        String email = args[2];
        String domainSuggestion = args[3];
        String firstName = args[4];
        String lastName = args[5];
        String city = args[6];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. List current domains.");
        listDomains(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List operations in the past year.");
        listOperations(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. View billing for the account in the past year.");
        listBillingRecords(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. View prices for domain types.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get domain suggestions.");
        listDomainSuggestions(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Check domain availability.");
        checkDomainAvailability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Check domain transferability.");
        checkDomainTransferability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Request a domain registration.");
        String opId = requestDomainRegistration(route53DomainsClient,
domainSuggestion, phoneNumber, email, firstName,
                lastName, city);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get operation details.");
        getOperationalDetail(route53DomainsClient, opId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get domain details.");
        System.out.println("Note: You must have a registered domain to get
details.");
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxx not found in xxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());
```

```
        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                    .operationId(operationId)
                    .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
            String domainSuggestion,
            String phoneNumber,
            String email,
            String firstName,
            String lastName,
            String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                    .contactType(ContactType.COMPANY)
                    .state("LA")
                    .countryCode(CountryCode.IN)
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
```

```
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();

            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainTransferabilityResponse response = route53DomainsClient
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           CheckDomainAvailabilityResponse response = route53DomainsClient
                   .checkDomainAvailability(availabilityRequest);
           System.out.println(domainSuggestion + " is " +
response.availability().toString());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                   .domainName(domainSuggestion)
                   .suggestionCount(5)
                   .onlyAvailable(true)
                   .build();

           GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
           List<DomainSuggestion> suggestions = response.suggestionsList();
           for (DomainSuggestion suggestion : suggestions) {
               System.out.println("Suggestion Name: " + suggestion.domainName());
               System.out.println("Availability: " + suggestion.availability());
               System.out.println(" ");
           }

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

```
   public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
       try {
           ListPricesRequest pricesRequest = ListPricesRequest.builder()
                   .tld(domainType)
                   .build();

           ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
           listRes.stream()
                   .flatMap(r -> r.prices().stream())
                   .forEach(content -> System.out.println(" Name: " +
content.name() +
                           " Registration: " + content.registrationPrice().price()
+ " "
                           + content.registrationPrice().currency() +
                           " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
       try {
           Date currentDate = new Date();
           LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
           ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
           LocalDateTime localDateTime2 = localDateTime.minusYears(1);
           Instant myStartTime = localDateTime2.toInstant(zoneOffset);
           Instant myEndTime = localDateTime.toInstant(zoneOffset);

           ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                   .start(myStartTime)
                   .end(myEndTime)
                   .build();

           ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
```

```
            listRes.stream()
                    .flatMap(r -> r.billingRecords().stream())
                    .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                            " Operation: " + content.operationAsString() +
                            " Price: " + content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listOperations(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
```

```
            ListDomainsIterable listRes =
 route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is " +
 content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Ações

`CheckDomainAvailability`

O código de exemplo a seguir mostra como usar `CheckDomainAvailability`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           CheckDomainAvailabilityResponse response = route53DomainsClient
                   .checkDomainAvailability(availabilityRequest);
           System.out.println(domainSuggestion + " is " +
response.availability().toString());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CheckDomainAvailability em Referência AWS SDK for Java 2.x da API.

`CheckDomainTransferability`

O código de exemplo a seguir mostra como usar `CheckDomainTransferability`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           CheckDomainTransferabilityResponse response = route53DomainsClient
                   .checkDomainTransferability(transferabilityRequest);
           System.out.println("Transferability: " +
response.transferability().transferable().toString());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte CheckDomainTransferability em Referência AWS SDK for Java 2.x da API.

`GetDomainDetail`

O código de exemplo a seguir mostra como usar `GetDomainDetail`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
       try {
           GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
           System.out.println("The contact first name is " +
response.registrantContact().firstName());
           System.out.println("The contact last name is " +
response.registrantContact().lastName());
           System.out.println("The contact org name is " +
response.registrantContact().organizationName());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte GetDomainDetail em Referência AWS SDK for Java 2.x da API.

`GetDomainSuggestions`

O código de exemplo a seguir mostra como usar `GetDomainSuggestions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                   .domainName(domainSuggestion)
                   .suggestionCount(5)
                   .onlyAvailable(true)
                   .build();

           GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
           List<DomainSuggestion> suggestions = response.suggestionsList();
           for (DomainSuggestion suggestion : suggestions) {
               System.out.println("Suggestion Name: " + suggestion.domainName());
               System.out.println("Availability: " + suggestion.availability());
               System.out.println(" ");
           }

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte GetDomainSuggestions em Referência AWS SDK for Java 2.x da API.

`GetOperationDetail`

O código de exemplo a seguir mostra como usar `GetOperationDetail`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
       try {
           GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                   .operationId(operationId)
                   .build();

           GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
           System.out.println("Operation detail message is " + response.message());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte GetOperationDetail em Referência AWS SDK for Java 2.x da API.

`ListDomains`

O código de exemplo a seguir mostra como usar `ListDomains`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListDomains em Referência AWS SDK for Java 2.x da API.

`ListOperations`

O código de exemplo a seguir mostra como usar `ListOperations`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listOperations(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);
```

```
            ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListOperations em Referência AWS SDK for Java 2.x da API.

`ListPrices`

O código de exemplo a seguir mostra como usar `ListPrices`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .tld(domainType)
```

```
                    .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                    .flatMap(r -> r.prices().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.name() +
                            " Registration: " + content.registrationPrice().price()
+ " "
                            + content.registrationPrice().currency() +
                            " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte ListPrices em Referência AWS SDK for Java 2.x da API.

`RegisterDomain`

O código de exemplo a seguir mostra como usar `RegisterDomain`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
            String domainSuggestion,
            String phoneNumber,
            String email,
            String firstName,
```

```
            String lastName,
            String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                    .contactType(ContactType.COMPANY)
                    .state("LA")
                    .countryCode(CountryCode.IN)
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();

            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte RegisterDomain em Referência AWS SDK for Java 2.x da API.

ViewBilling

O código de exemplo a seguir mostra como usar `ViewBilling`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
       try {
           Date currentDate = new Date();
           LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
           ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
           LocalDateTime localDateTime2 = localDateTime.minusYears(1);
           Instant myStartTime = localDateTime2.toInstant(zoneOffset);
           Instant myEndTime = localDateTime.toInstant(zoneOffset);

           ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                   .start(myStartTime)
                   .end(myEndTime)
                   .build();

           ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
           listRes.stream()
                   .flatMap(r -> r.billingRecords().stream())
                   .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                           " Operation: " + content.operationAsString() +
                           " Price: " + content.price()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ViewBilling em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon S3 usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
```

```
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Uma situação de exemplo.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger = LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = """
            Usage:
               <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The name of the  S3 bucket.
                key - The unique identifier for the object stored in the S3 bucket.
```

```
                objectPath - The full file path of the object within the S3 bucket
(e.g., "documents/reports/annual_report.pdf").
                savePath - The local file path where the object will be downloaded
and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
                toBucket - The name of the S3 bucket to which the object will be
copied.
            """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info("""
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to store
and retrieve
            any amount of data, from anywhere on the web, at any time.

            The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set
of methods to
            programmatically interact with the Amazon S3 (Simple Storage Service)
service. This allows
            developers to automate the management and manipulation of S3 buckets and
objects as
            part of their application deployment pipelines. With S3, teams can focus
on building
            and deploying their applications without having to worry about the
underlying storage
            infrastructure required to host and manage large amounts of data.

            This scenario walks you through how to perform key operations for this
service.
            Let's get started...
            """);
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            // Run the methods that belong to this scenario.
            runScenario(bucketName, key, objectPath, savePath, toBucket);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create an Amazon S3 bucket.");
        try {
            CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
            future.join();
            waitForInputToContinue(scanner);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Upload a local file to the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
            future.join();
            logger.info("File uploaded successfully to {}/{}", bucketName, key);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("3. Download the object to another local file.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future = s3Actions.multipartUpload(bucketName,
multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{}' and
key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. List all objects located in the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future = s3Actions.performMultiCopy(toBucket,
bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Delete the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Void> future =
 s3Actions.deleteBucketAsync(bucketName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
 s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon S3 scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper que contém as operações.

```
public class S3Actions {

    private static final Logger logger = LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }
```

```
    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
     * @throws RuntimeException if there is a failure while creating the bucket
     */
    public CompletableFuture<Void> createBucketAsync(String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
        return response.thenCompose(resp -> {
            S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                    logger.info(bucketName + " is ready");
                });
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }


    /**
     * Uploads a local file to an AWS S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
```

```
    * @param key        the key (object name) to use for the uploaded file
    * @param objectPath the local file path of the file to be uploaded
    * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
    */
   public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to upload file", ex);
           }
       });
   }


   /**
    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
```

```
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }


    /**
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
```

```
   /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param fromBucket the name of the source S3 bucket
    * @param objectKey  the key (name) of the object to be copied
    * @param toBucket   the name of the destination S3 bucket
    * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
    * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
    */
   public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
       CopyObjectRequest copyReq = CopyObjectRequest.builder()
           .sourceBucket(fromBucket)
           .sourceKey(objectKey)
           .destinationBucket(toBucket)
           .destinationKey(objectKey)
           .build();

       CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
       response.whenComplete((copyRes, ex) -> {
           if (copyRes != null) {
               logger.info("The " + objectKey + " was copied to " + toBucket);
           } else {
               throw new RuntimeException("An S3 exception occurred during copy",
ex);
           }
       });

       return response.thenApply(CopyObjectResponse::copyObjectResult)
           .thenApply(Object::toString);
   }

   /**
    * Performs a multipart upload to an Amazon S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
    * @param key        the key (name) of the file to be uploaded
    * @return a {@link CompletableFuture} that completes when the multipart upload
is successful
```

```
     */
    public CompletableFuture<Void> multipartUpload(String bucketName, String key) {
        int mB = 1024 * 1024;

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        return getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenCompose(createResponse -> {
                String uploadId = createResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(1)
                    .contentLength((long) (5 * mB)) // Specify the content length
                    .build();

                CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(1)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Upload part 2.
                UploadPartRequest uploadPartRequest2 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(2)
                    .contentLength((long) (3 * mB))
                    .build();

                CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
```

```
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(2)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Combine the results of both parts.
                return CompletableFuture.allOf(part1Future, part2Future)
                    .thenCompose(v -> {
                        CompletedPart part1 = part1Future.join();
                        CompletedPart part2 = part2Future.join();

                        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                            .parts(part1, part2)
                            .build();

                        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                            .bucket(bucketName)
                            .key(key)
                            .uploadId(uploadId)
                            .multipartUpload(completedMultipartUpload)
                            .build();

                        // Complete the multipart upload
                        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
                    });
            })
            .thenAccept(response -> System.out.println("Multipart upload completed
successfully"))
            .exceptionally(ex -> {
                System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }


    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
```

```
    * @param bucketName the name of the S3 bucket
    * @param key        the key (file name) of the object to be deleted
    * @return a {@link CompletableFuture} that completes when the object has been
deleted
    */
   public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
       DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(key + " was deleted");
           } else {
               throw new RuntimeException("An S3 exception occurred during delete",
ex);
           }
       });

       return response.thenApply(r -> null);
   }


   /**
    * Deletes an S3 bucket asynchronously.
    *
    * @param bucket the name of the bucket to be deleted
    * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
    * if an error occurs during the deletion process
    */
   public CompletableFuture<Void> deleteBucketAsync(String bucket) {
       DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
           .bucket(bucket)
           .build();

       CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
```

```
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }

    public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
```

```
                    System.out.println("Upload Part Copy completed successfully.
 ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        ByteBuffer buffer = ByteBuffer.allocate(size);
        for (int i = 0; i < size; i++) {
            buffer.put((byte) (Math.random() * 256));
        }
        buffer.flip();
        return buffer;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações

`AbortMultipartUpload`

O código de exemplo a seguir mostra como usar `AbortMultipartUpload`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
```

```
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
 LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        String key = UUID.randomUUID().toString();

        createBucket(bucketName);
        try {
            initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
            abortIncompleteMultipartUploadsFromList(bucketName);
            abortMultipartUploadUsingUploadId(bucketName, key);
            abortMultipartUploadsUsingLifecycleConfig(bucketName);
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources(bucketName, key);
        }
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
```

```
    */
   public static void abortIncompleteMultipartUploadsFromList(String bucketName) {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
           .bucket(bucketName)
           .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(upload.key())
               .expectedBucketOwner(accountId)
               .uploadId(upload.uploadId())
               .build();

           AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
           if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
               logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
           }
       }
   }

   /**
    * Aborts incomplete multipart uploads older than the specified point in time.
    *
    * @param bucketName  the name of the S3 bucket
    * @param pointInTime the cutoff time; uploads initiated before this are aborted
    */
   static void abortIncompleteMultipartUploadsOlderThan(String bucketName, Instant
pointInTime) {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
           .bucket(bucketName)
           .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
```

```
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }

    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key) {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
        }
    }

    /**
```

```
    * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
    *
    * @param bucketName the name of the S3 bucket
    */
   static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
       Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
           .abortIncompleteMultipartUpload(b -> b.
               daysAfterInitiation(7))
           .status("Enabled")
           .filter(SdkBuilder::build) // Filter element is required.
           .build());

       PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
           .bucket(bucketName)
           .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

       if (response.sdkHttpResponse().isSuccessful()) {
           logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
       } else {
           logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
       }
   }

   /************************
    Multipart upload methods
    ***********************/

   static void initiateAndInterruptMultiPartUpload(String bucketName, String key,
String threadName) {
       Runnable upload = () -> {
           try {
               doMultipartUpload(bucketName, key);
           } catch (SdkException e) {
               logger.error(e.getMessage());
           }
       };
       Thread uploadThread = new Thread(upload, threadName);
       uploadThread.start();
       try {
           Thread.sleep(Duration.ofSeconds(1).toMillis());
```

```
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key) {
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
        return secondUploadInstant;
    }

    static void doMultipartUpload(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
        step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
    }

    static String step1CreateMultipartUpload(String bucketName, String key) {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
```

```
                long read = file.getChannel().read(bb);

                bb.flip();
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);
                logger.info("Part {} upload", partNumber);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }

    static void step3CompleteMultipartUpload(String bucketName, String key, String
uploadId, List<CompletedPart> completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }

    static String startUploadReturningUploadId(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
```

```
        doMultipartUploadWithUploadId(bucketName, key, uploadId);
        return uploadId;
    }

    static void doMultipartUploadWithUploadId(String bucketName, String key, String
uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
                step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }

    /************************
     Resource handling methods
     ************************/

    static void createBucket(String bucketName) {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources(String bucketName, String key) {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
```

```
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL = PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
 Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
    }
}
```

- Para obter detalhes da API, consulte AbortMultipartUpload em Referência AWS SDK for Java 2.x da API.

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie um objeto usando um S3Client.

```
    /**
```

```
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during copy",
ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }
```

Use an S3TransferManager para copiar um objeto de um bucket para outro. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
```

```
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK for Java 2.x da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket.

```
   /**
    * Creates an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to create
    * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
    * @throws RuntimeException if there is a failure while creating the bucket
    */
   public CompletableFuture<Void> createBucketAsync(String bucketName) {
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .build();

       CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
       return response.thenCompose(resp -> {
           S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
           HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
               .bucket(bucketName)
               .build();

           CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
               s3Waiter.waitUntilBucketExists(bucketRequestWait);
           return waiterResponseFuture.thenAccept(waiterResponse -> {
               waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                   logger.info(bucketName + " is ready");
               });
           });
       }).whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to create bucket", ex);
           }
       });
   }
```

Crie um bucket com o bloqueio de objetos habilitado.

```
   // Create a new Amazon S3 bucket with object lock options.
```

```
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }
```

- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK for Java 2.x da API.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an S3 bucket asynchronously.
     *
     * @param bucket the name of the bucket to be deleted
     * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
```

```
     * if an error occurs during the deletion process
     */
    public CompletableFuture<Void> deleteBucketAsync(String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }
```

- Para obter detalhes da API, consulte DeleteBucket em Referência AWS SDK for Java 2.x da API.

`DeleteBucketPolicy`

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from (for
 example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.
     *
     * @param s3 the {@link S3Client} instance to use for the operation
     * @param bucketName the name of the S3 bucket for which the policy should be
 deleted
     *
```

```
     * @throws S3Exception if there is an error deleting the bucket policy
     */
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteBucketPolicy em Referência AWS SDK for Java 2.x da API.

`DeleteBucketWebsite`

O código de exemplo a seguir mostra como usar `DeleteBucketWebsite`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the website
 configuration from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s
\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     * @param bucketName The name of the S3 bucket for which the website
 configuration should be deleted.
```

```
     * @throws S3Exception If an error occurs while deleting the website
 configuration.
     */
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteBucketWebsite em Referência AWS SDK for Java 2.x da API.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the key (file name) of the object to be deleted
```

```
     * @return a {@link CompletableFuture} that completes when the object has been
deleted
     */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during delete",
ex);
            }
        });

        return response.thenApply(r -> null);
    }
```

- Para obter detalhes da API, consulte DeleteObject em Referência AWS SDK for Java 2.x da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName>

            Where:
               bucketName - the Amazon S3 bucket name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }
```

```
    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                .key(keyName)
                .build();

            putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
            .objects(keys)
            .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(del)
                .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK for Java 2.x da API.

`GetBucketAcl`

O código de exemplo a seguir mostra como usar `GetBucketAcl`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <objectKey>

            Where:
              bucketName - The Amazon S3 bucket to get the access control list (ACL)
 for.
              objectKey - The object to get the ACL for.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an empty
 string if an error occurs.
     */
```

```
    public static String getBucketACL(S3Client s3, String objectKey, String
 bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
 grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- Para obter detalhes da API, consulte GetBucketAcl em Referência AWS SDK for Java 2.x da API.

`GetBucketPolicy`

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }
```

```
    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     * @param bucketName the name of the S3 bucket for which to retrieve the policy
     * @return the policy text for the specified bucket, or an empty string if an
 error occurs
     */
    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- Para obter detalhes da API, consulte GetBucketPolicy em Referência AWS SDK for Java 2.x da API.

`GetBucketReplication`

O código de exemplo a seguir mostra como usar `GetBucketReplication`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Retrieves the replication details for the specified S3 bucket.
    *
    * @param s3Client           the S3 client used to interact with the S3 service
    * @param sourceBucketName   the name of the S3 bucket to retrieve the
replication details for
    *
    * @throws S3Exception if there is an error retrieving the replication details
    */
   public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
       GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
           .bucket(sourceBucketName)
           .build();

       try {
           ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
           ReplicationRule rule = replicationConfig.rules().get(0);
           System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
           System.out.println("Retrieved priority: " + rule.priority());
           System.out.println("Retrieved source-bucket replication rule status: " +
rule.status());

       } catch (S3Exception e) {
           System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
       }
   }
```

- Para obter detalhes da API, consulte GetBucketReplication em Referência AWS SDK for Java 2.x da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Leia dados como uma matriz de bytes usando um S3Client.

```
   /**
    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
       return response.thenAccept(objectBytes -> {
           try {
               byte[] data = objectBytes.asByteArray();
               Path filePath = Paths.get(path);
```

```
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
 ex);
            }
        });
    }
```

Use an S3TransferManager para baixar um objeto em um bucket do S3 para um arquivo local. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();
```

```
        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Leia etiquetas que pertencem a um objeto usando um S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
```

```
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Lists the tags associated with an Amazon S3 object.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket that contains the object
     * @param keyName the key (name) of the S3 object
     */
    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

Obtenha um URL para um objeto usando um S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
```

```
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be retrieved
     * @throws S3Exception if there is an error retrieving the URL for the specified
 object
     */
    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Obtenha um objeto usando o objeto de cliente S3Presigner usando um S3Client.

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }
```

```
   /**
    * Generates a pre-signed URL for an Amazon S3 object.
    *
    * @param presigner The {@link S3Presigner} instance to use for generating the
pre-signed URL.
    * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
    * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
    *
    * @throws S3Exception If there is an error interacting with the Amazon S3
service.
    * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
    */
   public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
       try {
           GetObjectRequest getObjectRequest = GetObjectRequest.builder()
               .bucket(bucketName)
               .key(keyName)
               .build();

           GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
               .signatureDuration(Duration.ofMinutes(60))
               .getObjectRequest(getObjectRequest)
               .build();

           PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
           String theUrl = presignedGetObjectRequest.url().toString();
           System.out.println("Presigned URL: " + theUrl);
           HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
           presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
               values.forEach(value -> {
                   connection.addRequestProperty(header, value);
               });
           });

           // Send any request payload that the service needs (not needed when
```

```
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                     OutputStream httpOutputStream = connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.printStackTrace();
        }
    }
}
```

Obtenha um objeto usando um ResponseTransformer objeto e o https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/s3/S3Client.html S3Client.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves them
 to a local file.
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
```

```
     * @param path The local file path where the object's bytes will be saved.
     * @throws IOException If an I/O error occurs while writing the bytes to the
 local file.
     * @throws S3Exception If an error occurs while retrieving the object from the
 S3 bucket.
     */
    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK for Java 2.x da API.

GetObjectLegalHold

O código de exemplo a seguir mostra como usar `GetObjectLegalHold`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get the legal hold details for an S3 object.
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
           System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
               ":\n\tStatus: " + response.legalHold().status());
           return response.legalHold();

       } catch (S3Exception ex) {
           System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
"'");
       }

       return null;
   }
```

- Para obter detalhes da API, consulte GetObjectLegalHold em Referência AWS SDK for Java 2.x da API.

`GetObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `GetObjectLockConfiguration`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get the object lock configuration details for an S3 bucket.
   public void getBucketObjectLockConfiguration(String bucketName) {
       GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
           .bucket(bucketName)
           .build();

       GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
       System.out.println("Bucket object lock config for "+bucketName +":  ");
       System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
       System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
   }
```

- Para obter detalhes da API, consulte GetObjectLockConfiguration em Referência AWS SDK for Java 2.x da API.

`GetObjectRetention`

O código de exemplo a seguir mostra como usar `GetObjectRetention`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }
```

- Para obter detalhes da API, consulte GetObjectRetention em Referência AWS SDK for Java 2.x da API.

`HeadObject`

O código de exemplo a seguir mostra como usar `HeadObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Determine o tipo de conteúdo de um objeto.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the content type of an object stored in an Amazon S3 bucket.
```

```
     *
     * @param s3 an instance of the {@link S3Client} class, which is used to
 interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the key (file name) of the object in the S3 bucket
     */
    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Obtenha o status de restauração de um objeto.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
```

```
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3         an instance of the {@link S3Client} class used to interact
with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param keyName    the name of the Amazon S3 object to be checked
     * @throws S3Exception if an error occurs while interacting with the Amazon S3
service
     */
    public static void checkStatus(S3Client s3, String bucketName, String keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte HeadObject em Referência AWS SDK for Java 2.x da API.

ListBuckets

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }
```

```
    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
 Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK for Java 2.x da API.

`ListMultipartUploads`

O código de exemplo a seguir mostra como usar `ListMultipartUploads`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon S3
 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
 uploads for
     */
    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
```

```
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key() +
 "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListMultipartUploads em Referência AWS SDK for Java 2.x da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
 listed
     */
```

```
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
 getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
 bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
```

Liste objetos usando paginação.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the S3 bucket to list the objects from
     */
    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() + "
 size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK for Java 2.x API.

PutBucketAcl

O código de exemplo a seguir mostra como usar `PutBucketAcl`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <id>\s

            Where:
              bucketName - The Amazon S3 bucket to grant permissions on.\s
```

```
             id - The ID of the owner of this bucket (you can get this value from
the AWS Management Console).
           """;

       if (args.length != 2) {
           System.out.println(usage);
           return;
       }

       String bucketName = args[0];
       String id = args[1];
       System.out.format("Setting access \n");
       System.out.println(" in bucket: " + bucketName);
       Region region = Region.US_EAST_1;
       S3Client s3 = S3Client.builder()
           .region(region)
           .build();

       setBucketAcl(s3, bucketName, id);
       System.out.println("Done!");
       s3.close();
   }

   /**
    * Sets the Access Control List (ACL) for an Amazon S3 bucket.
    *
    * @param s3 the S3Client instance to be used for the operation
    * @param bucketName the name of the S3 bucket to set the ACL for
    * @param id the ID of the AWS user or account that will be granted full control
of the bucket
    * @throws S3Exception if an error occurs while setting the bucket ACL
    */
   public static void setBucketAcl(S3Client s3, String bucketName, String id) {
       try {
           Grant ownerGrant = Grant.builder()
               .grantee(builder -> builder.id(id)
                   .type(Type.CANONICAL_USER))
               .permission(Permission.FULL_CONTROL)
               .build();

           List<Grant> grantList2 = new ArrayList<>();
           grantList2.add(ownerGrant);

           AccessControlPolicy acl = AccessControlPolicy.builder()
```

```
                .owner(builder -> builder.id(id))
                .grants(grantList2)
                .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutBucketAcl em Referência AWS SDK for Java 2.x da API.

`PutBucketCors`

O código de exemplo a seguir mostra como usar `PutBucketCors`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
import java.util.List;
```

```
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
```

```
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an Amazon
S3 bucket.
     *
     * @param s3            the {@link S3Client} instance used to interact with the
Amazon S3 service
     * @param bucketName    the name of the Amazon S3 bucket for which the CORS
configuration should be deleted
     * @param accountId     the expected AWS account ID of the bucket owner
     *
     * @throws S3Exception if an error occurs while deleting the CORS configuration
for the bucket
     */
    public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
specified S3 bucket.
     *
     * @param s3 the S3Client instance to use for the operation
     * @param bucketName the name of the S3 bucket to retrieve the CORS
configuration for
     * @param accountId the expected bucket owner's account ID
     *
     * @throws S3Exception if there is an error retrieving the CORS configuration
     */
```

```
   public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
       try {
           GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
               .bucket(bucketName)
               .expectedBucketOwner(accountId)
               .build();

           GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
           List<CORSRule> corsRules = corsResponse.corsRules();
           for (CORSRule rule : corsRules) {
               System.out.println("allowOrigins: " + rule.allowedOrigins());
               System.out.println("AllowedMethod: " + rule.allowedMethods());
           }

       } catch (S3Exception e) {

           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   /**
    * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3 bucket.
    *
    * @param s3 The S3Client object used to interact with the Amazon S3 service.
    * @param bucketName The name of the S3 bucket to set the CORS rules for.
    * @param accountId The AWS account ID of the bucket owner.
    */
   public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
       List<String> allowMethods = new ArrayList<>();
       allowMethods.add("PUT");
       allowMethods.add("POST");
       allowMethods.add("DELETE");

       List<String> allowOrigins = new ArrayList<>();
       allowOrigins.add("http://example.com");
       try {
           // Define CORS rules.
           CORSRule corsRule = CORSRule.builder()
               .allowedMethods(allowMethods)
               .allowedOrigins(allowOrigins)
```

```
                .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                .corsRules(corsRules)
                .build();

            PutBucketCorsRequest putBucketCorsRequest =
 PutBucketCorsRequest.builder()
                .bucket(bucketName)
                .corsConfiguration(configuration)
                .expectedBucketOwner(accountId)
                .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutBucketCors em Referência AWS SDK for Java 2.x da API.

`PutBucketLifecycleConfiguration`

O código de exemplo a seguir mostra como usar `PutBucketLifecycleConfiguration`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <accountId>\s

            Where:
              bucketName - The Amazon Simple Storage Service (Amazon S3) bucket to
 upload an object into.
              accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3           The Amazon S3 client to use for the operation.
     * @param bucketName   The name of the Amazon S3 bucket.
     * @param accountId    The expected owner of the Amazon S3 bucket.
     *
     * @throws S3Exception if there is an error setting the lifecycle configuration.
     */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix to
the
            // S3 Glacier Flexible Retrieval storage class immediately.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
```

```
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Create a second rule.
            Transition transition2 = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            List<Transition> transitionList = new ArrayList<>();
            transitionList.add(transition2);

            LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            LifecycleRule rule2 = LifecycleRule.builder()
                .id("Archive and then delete rule")
                .filter(ruleFilter2)
                .transitions(transitionList)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the LifecycleRule objects to an ArrayList.
            ArrayList<LifecycleRule> ruleList = new ArrayList<>();
            ruleList.add(rule1);
            ruleList.add(rule2);

            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(ruleList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a new
lifecycle rule to it.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket
     * @param accountId the expected owner of the Amazon S3 bucket
     */
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag predicate.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();
```

```
            LifecycleRule rule1 = LifecycleRule.builder()
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the new rule to the list.
            newList.add(rule1);
            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} to use for the operation
     * @param bucketName the name of the S3 bucket
     * @param accountId the expected account owner of the S3 bucket
     *
     * @throws S3Exception if an error occurs while deleting the lifecycle
configuration
     */
    public static void deleteLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
```

```
            DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
 DeleteBucketLifecycleRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutBucketLifecycleConfiguration em Referência AWS SDK for Java 2.x da API.

`PutBucketPolicy`

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
```

```
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <polFile>

            Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
 Readme for an example).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }
```

```
    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3         the {@link S3Client} object used to interact with the
Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
     */
    public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)
                .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    /**
     * Retrieves the bucket policy from a specified file.
     *
     * @param policyFile the path to the file containing the bucket policy
     * @return the content of the bucket policy file as a string
     */
    public static String getBucketPolicyFromFile(String policyFile) {
        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
```

```
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- Para obter detalhes da API, consulte PutBucketPolicy em Referência AWS SDK for Java 2.x da API.

`PutBucketReplication`

O código de exemplo a seguir mostra como usar `PutBucketReplication`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Sets the replication configuration for an Amazon S3 bucket.
     *
```

```
    * @param s3Client             the S3Client instance to use for the operation
    * @param sourceBucketName     the name of the source bucket
    * @param destBucketName       the name of the destination bucket
    * @param destinationBucketARN the Amazon Resource Name (ARN) of the destination
bucket
    * @param roleARN              the ARN of the IAM role to use for the
replication configuration
    */
   public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
       try {
           Destination destination = Destination.builder()
               .bucket(destinationBucketARN)
               .storageClass(StorageClass.STANDARD)
               .build();

           // Define a prefix filter for replication.
           ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
               .prefix("documents/")
               .build();

           // Define delete marker replication setting.
           DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
               .status(DeleteMarkerReplicationStatus.DISABLED)
               .build();

           // Create the replication rule.
           ReplicationRule replicationRule = ReplicationRule.builder()
               .priority(1)
               .filter(ruleFilter)
               .status(ReplicationRuleStatus.ENABLED)
               .deleteMarkerReplication(deleteMarkerReplication)
               .destination(destination)
               .build();

           List<ReplicationRule> replicationRuleList = new ArrayList<>();
           replicationRuleList.add(replicationRule);

           // Define the replication configuration with IAM role.
           ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
               .role(roleARN)
               .rules(replicationRuleList)
```

```
                .build();

            // Apply the replication configuration to the source bucket.
            PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
                .bucket(sourceBucketName)
                .replicationConfiguration(configuration)
                .build();

            s3Client.putBucketReplication(replicationRequest);
            System.out.println("Replication configuration set successfully.");

        } catch (IllegalArgumentException e) {
            System.err.println("Configuration error: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
            System.err.println("Status Code: " + e.statusCode());
            System.err.println("Error Code: " + e.awsErrorDetails().errorCode());


        } catch (SdkException e) {
            System.err.println("SDK Exception: " + e.getMessage());
        }
    }
```

- Para obter detalhes da API, consulte PutBucketReplication em Referência AWS SDK for Java 2.x da API.

`PutBucketVersioning`

O código de exemplo a seguir mostra como usar `PutBucketVersioning`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Enables bucket versioning for the specified S3 bucket.
     *
     * @param s3Client the S3 client to use for the operation
     * @param bucketName the name of the S3 bucket to enable versioning for
     */
    public static void enableBucketVersioning(S3Client s3Client, String bucketName){
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        s3Client.putBucketVersioning(versioningRequest);
        System.out.println("Bucket versioning has been enabled for "+bucketName);
    }
```

- Para obter detalhes da API, consulte PutBucketVersioning em Referência AWS SDK for Java 2.x da API.

`PutBucketWebsite`

O código de exemplo a seguir mostra como usar `PutBucketWebsite`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
```

```
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName> [indexdoc]\s

            Where:
               bucketName   - The Amazon S3 bucket to set the website configuration
 on.\s
               indexdoc - The index document, ex. 'index.html'
                          If not specified, 'index.html' will be set.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    /**
```

```
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
     * @param indexDoc The name of the index document to use for the website
 configuration.
     */
    public static void setWebsiteConfig(S3Client s3, String bucketName, String
 indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                .build();

            PutBucketWebsiteRequest pubWebsiteReq =
 PutBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .websiteConfiguration(websiteConfig)
                .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutBucketWebsite em Referência AWS SDK for Java 2.x da API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de um arquivo em um bucket usando um S3Client.

```
   /**
    * Uploads a local file to an AWS S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
    * @param key        the key (object name) to use for the uploaded file
    * @param objectPath the local file path of the file to be uploaded
    * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
    */
   public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to upload file", ex);
           }
       });
   }
```

Use an S3TransferManager para fazer upload de um arquivo para um bucket. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Faça upload de um objeto em um bucket e defina etiquetas usando um S3Client.

```
    /**
     * Puts tags on an Amazon S3 object.
     *
     * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
     * @param bucketName The name of the Amazon S3 bucket.
     * @param objectKey The key of the Amazon S3 object.
     * @param objectPath The file path of the object to be uploaded.
     */
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                .key("Tag 1")
                .value("This is tag 1")
```

```
                .build();

            Tag tag2 = Tag.builder()
                .key("Tag 2")
                .value("This is tag 2")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);

            Tagging allTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(allTags)
                .build();

            s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the tags associated with an object in an Amazon S3 bucket.
     *
     * @param s3 an instance of the S3Client class, which is used to interact with
the Amazon S3 service
     * @param bucketName the name of the S3 bucket containing the object
     * @param objectKey the key (or name) of the object in the S3 bucket
     * @throws S3Exception if there is an error updating the object's tags
     */
    public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
        try {
            GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
                .bucket(bucketName)
```

```
                .key(objectKey)
                .build();

            GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.tagSet();
            for (Tag sinTag : obTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

            // Replace the object's tags with two new tags.
            Tag tag3 = Tag.builder()
                .key("Tag 3")
                .value("This is tag 3")
                .build();

            Tag tag4 = Tag.builder()
                .key("Tag 4")
                .value("This is tag 4")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag3);
            tags.add(tag4);

            Tagging updatedTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(updatedTags)
                .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
```

```
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the contents of a file as a byte array.
     *
     * @param filePath the path of the file to be read
     * @return a byte array containing the contents of the file, or null if an error
 occurs
     */
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

Faça upload de um objeto em um bucket e defina metadados usando um S3Client.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
              <bucketName> <objectKey> <objectPath>\s

            Where:
              bucketName - The Amazon S3 bucket to upload an object into.
              objectKey - The object to upload (for example, book.pdf).
              objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).\s
            """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
```

```
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    /**
     * Uploads an object to an Amazon S3 bucket with metadata.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
 String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
 + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Faça upload de um objeto em um bucket e defina um valor de retenção de objetos usando um S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object (for
 example, bucket1).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
```

```
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
     *
     * @param s3     the S3Client object used to interact with the Amazon S3 service
     * @param key    the key (name) of the object in the S3 bucket
     * @param bucket the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
period
     */
    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                .mode("COMPLIANCE")
                .retainUntilDate(instant)
                .build();

            PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
                .bucket(bucket)
                .key(key)
                .bypassGovernanceRetention(true)
                .retention(lockRetention)
                .build();

            // To set Retention on an object, the Amazon S3 bucket must support
object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
```

```
            System.out.print("An object retention configuration was successfully
 placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK for Java 2.x da API.

`PutObjectLegalHold`

O código de exemplo a seguir mostra como usar `PutObjectLegalHold`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
 legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
 PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
```

```
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
    }
```

- Para obter detalhes da API, consulte PutObjectLegalHold em Referência AWS SDK for Java 2.x da API.

`PutObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `PutObjectLockConfiguration`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de Bloqueio de Objetos de um bucket.

```
    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
```

```
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on "+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage() +
"'");
        }
    }
```

Defina o período de retenção padrão de um bucket.

```
    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
```

```
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();

        getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName +".");
    }
```

- Para obter detalhes da API, consulte PutObjectLockConfiguration em Referência AWS SDK for Java 2.x da API.

`PutObjectRetention`

O código de exemplo a seguir mostra como usar `PutObjectRetention`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time zone.
        ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());
```

```
        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }
```

- Para obter detalhes da API, consulte PutObjectRetention em Referência AWS SDK for Java 2.x da API.

`RestoreObject`

O código de exemplo a seguir mostra como usar `RestoreObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 *  Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
 Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
 obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
```

```
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *
     * @param s3                   an instance of the {@link S3Client} to be used
for interacting with Amazon S3
     * @param bucketName           the name of the S3 bucket where the object is
stored
     * @param keyName              the key (object name) of the S3 object to be
restored
     * @param expectedBucketOwner  the AWS account ID of the expected bucket owner
     */
    public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                .expectedBucketOwner(expectedBucketOwner)
                .bucket(bucketName)
                .key(keyName)
                .restoreRequest(restoreRequest)
                .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte RestoreObject em Referência AWS SDK for Java 2.x da API.

`SelectObjectContent`

O código de exemplo a seguir mostra como usar `SelectObjectContent`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo a seguir mostra uma consulta usando um objeto JSON. O exemplo completo também mostra o uso de um objeto CSV.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
```

```
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where c.area
 < 350000";
        final String fileType = FILE_JSON;
```

```
        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
                .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times", fileType,
eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
```

```
        // Use a Visitor to process the response stream. This visitor logs
 information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
 SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
```

```
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- Para obter detalhes da API, consulte SelectObjectContent em Referência AWS SDK for Java 2.x da API.

UploadPartCopy

O código de exemplo a seguir mostra como usar `UploadPartCopy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
           .bucket(toBucket)
           .key(key)
           .build();

       getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
           .thenApply(createMultipartUploadResponse -> {
               String uploadId = createMultipartUploadResponse.uploadId();
               System.out.println("Upload ID: " + uploadId);

               UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                   .sourceBucket(bucketName)
                   .destinationBucket(toBucket)
                   .sourceKey(key)
                   .destinationKey(key)
                   .uploadId(uploadId)  // Use the valid uploadId.
                   .partNumber(1)  // Ensure the part number is correct.
                   .copySourceRange("bytes=0-1023")  // Adjust range as needed
                   .build();

               return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
           })
           .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
           .whenComplete((uploadPartCopyResponse, exception) -> {
               if (exception != null) {
                   // Handle any exceptions.
                   logger.error("Error during upload part copy: " +
exception.getMessage());
               } else {
                   // Successfully completed the upload part copy.
```

```
                    System.out.println("Upload Part Copy completed successfully.
 ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }
```

- Para obter detalhes da API, consulte UploadPartCopy em Referência AWS SDK for Java 2.x da API.

Cenários

Verificar se um bucket existe

O exemplo de código a seguir mostra como verificar se um bucket existe.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Você pode usar o `doesBucketExists` método a seguir como substituto do SDK para Java V1 AmazonS3Client #does BucketExist V2 (String).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

public class DoesBucketExist {
    private static final Logger logger =
 LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();
```

```
        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the bucket
 name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName, s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
 global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't be
 used to create a new bucket.
     * <p>
     * Internally this method uses the <a
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
     *
     * @param bucketName   The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks for
 the bucket in the AWS Region
     *                     configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the value
 false if there is no bucket in
     *         Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket exists
 but it's not in this region
```

```
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
 "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
    }
}
```

- Para obter detalhes da API, consulte GetBucketAcl em Referência AWS SDK for Java 2.x da API.

Verificar se um objeto existe

O exemplo de código a seguir mostra como verificar se um objeto existe.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Você pode usar o `doesObjectExist` método a seguir como substituto do SDK para Java V1 AmazonS3Client #does ObjectExist (String, String).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;
import software.amazon.awssdk.utils.Validate;

public class DoesObjectExist {
```

```
    private static final Logger logger =
 LoggerFactory.getLogger(DoesObjectExist.class);

    public static void main(String[] args) {
        DoesObjectExist doesObjectExist = new DoesObjectExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
        final String key = "my-key"; // Replace with your object key.

        boolean exists = doesObjectExist.doesObjectExist(bucketName, key,
 s3SyncClient);
        logger.info("Object exists: {}", exists);
    }

    /**
     * Checks if the specified object exists in the specified bucket.
     * <p>
     * Internally this method uses the
     * <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#headObject(java.util.function.Consumer)">S3Client.headObject</a>
     * operation to determine whether the object exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's
     * <a href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/
amazonaws/services/s3/AmazonS3Client.html#doesObjectExist-java.lang.String-
java.lang.String-">AmazonS3Client#doesObjectExist(String, String)</a>.
     * <p>
     * <b>Note:</b> This method returns {@code false} only when S3 responds with 404
 (NoSuchKey). If the caller
     * does not have {@code s3:ListBucket} permission on the bucket, S3 may return
 403 instead of 404 for
     * non-existent objects, which will be thrown as an exception.
     *
     * @param bucketName   The name of the bucket containing the object.
     * @param key          The key of the object to check.
     * @param s3SyncClient An {@code S3Client} instance.
     * @return {@code true} if the object exists; {@code false} if it does not
 exist.
     */
    public boolean doesObjectExist(String bucketName, String key, S3Client
 s3SyncClient) {
```

```
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            Validate.notEmpty(key, "The object key must not be null or an empty
 string.", "");
            s3SyncClient.headObject(HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .build());
            return true;
        } catch (NoSuchKeyException e) {
            return false;
        }
    }
}
```

- Para obter detalhes da API, consulte HeadObject em Referência AWS SDK for Java 2.x da API.

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Veja a seguir três exemplos de como criar URLs pré-assinados e usar os URLs com bibliotecas de cliente HTTP:

- Uma solicitação HTTP GET que usa o URL com três bibliotecas de cliente HTTP
- Uma solicitação HTTP PUT com metadados nos cabeçalhos que usa o URL com três bibliotecas de cliente HTTP
- Uma solicitação HTTP PUT com parâmetros de consulta que usa o URL com uma biblioteca cliente HTTP

Gere um URL pré-assinado para um objeto e faça download dele (solicitação GET).

Importações.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Gere o URL.

```
    /* Create a pre-signed URL to download an object in a subsequent GET request. */
    public String createPresignedGetUrl(String bucketName, String keyName) {
        try (S3Presigner presigner = S3Presigner.create()) {

            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
```

```
                    .key(keyName)
                    .build();

            GetObjectPresignRequest presignRequest =
GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL will
expire in 10 minutes.
                    .getObjectRequest(objectRequest)
                    .build();

            PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
            logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
            logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Faça o download do objeto usando uma das três abordagens a seguir.

Use a classe `HttpURLConnection` do JDK (desde a v1.1) para fazer o download.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
    public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setRequestMethod("GET");
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                IoUtils.copy(content, byteArrayOutputStream);
            }
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
```

```
        return byteArrayOutputStream.toByteArray();
    }
```

Use a classe `HttpClient` do JDK (desde a v11) para fazer o download.

```
    /* Use the JDK HttpClient (since v11) class to do the download. */
    public byte[] useHttpClientToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpResponse<InputStream> response = httpClient.send(requestBuilder
                            .uri(presignedUrl.toURI())
                            .GET()
                            .build(),
                    HttpResponse.BodyHandlers.ofInputStream());

            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Use a `SdkHttpClient` classe AWS SDK para Java para fazer o download.

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToGet(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
```

```
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
 byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Gere um URL pré-assinado com metadados em cabeçalhos para um upload e faça upload de um arquivo (solicitação PUT).

Importações.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
```

```
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Gere o URL.

```
    /* Create a presigned URL to use in a subsequent PUT request */
    public String createPresignedUrl(String bucketName, String keyName, Map<String,
 String> metadata) {
        try (S3Presigner presigner = S3Presigner.create()) {

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .metadata(metadata)
                    .build();

            PutObjectPresignRequest presignRequest =
 PutObjectPresignRequest.builder()
```

```
                    .signatureDuration(Duration.ofMinutes(10))  // The URL expires
in 10 minutes.
                    .putObjectRequest(objectRequest)
                    .build();


           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

Faça upload de um objeto de arquivo usando uma das três abordagens a seguir.

Use a classe `HttpURLConnection` do JDK (desde a v1.1) para fazer o upload.

```
   /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
   public void useHttpUrlConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());
       try {
           URL presignedUrl = new URL(presignedUrlString);
           HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
           connection.setDoOutput(true);
           metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));
           connection.setRequestMethod("PUT");
           OutputStream out = connection.getOutputStream();

           try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                FileChannel inChannel = file.getChannel()) {
               ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

               while (inChannel.read(buffer) > 0) {
                   buffer.flip();
                   for (int i = 0; i < buffer.limit(); i++) {
                       out.write(buffer.get());
```

```
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Use a classe `HttpClient` do JDK (desde a v11) para fazer o upload.

```
    /* Use the JDK HttpClient (since v11) class to do the upload. */
    public void useHttpClientToPut(String presignedUrlString, File fileToPut,
 Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            final HttpResponse<Void> response = httpClient.send(requestBuilder
                            .uri(new URL(presignedUrlString).toURI())

 .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                            .build(),
                    HttpResponse.BodyHandlers.discarding());

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Use a `SdkHttpClient` classe AWS for Java V2 para fazer o upload.

```
   /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
   public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       try {
           URL presignedUrl = new URL(presignedUrlString);

           SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                   .method(SdkHttpMethod.PUT)
                   .uri(presignedUrl.toURI());
           // Add headers
           metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k,
v));
           // Finish building the request.
           SdkHttpRequest request = requestBuilder.build();

           HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                   .request(request)
                   .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                   .build();

           try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
               HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
               logger.info("Response code: {}",
response.httpResponse().statusCode());
           }
       } catch (URISyntaxException | IOException e) {
           logger.error(e.getMessage(), e);
       }
   }
```

Gere um URL pré-assinado com parâmetros de consulta para um upload e faça upload de um arquivo (solicitação PUT).

Importações.

```
import com.example.s3.util.PresignUrlUtils;
```

```
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Gere o URL.

```
   /**
    *  Creates a presigned URL to use in a subsequent HTTP PUT request. The code
 adds query parameters
    *  to the request instead of using headers. By using query parameters, you do
 not need to add the
    *  the parameters as headers when the PUT request is eventually sent.
    *
    * @param bucketName Bucket name where the object will be uploaded.
    * @param keyName Key name of the object that will be uploaded.
    * @param queryParams Query string parameters to be added to the presigned URL.
    * @return
    */
   public String createPresignedUrl(String bucketName, String keyName, Map<String,
 String> queryParams) {
       try (S3Presigner presigner = S3Presigner.create()) {
           // Create an override configuration to store the query parameters.
```

```
            AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder =
AwsRequestOverrideConfiguration.builder();

            queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .overrideConfiguration(overrideConfigurationBuilder.build()) //
Add the override configuration.
                    .build();

            PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL expires
in 10 minutes.
                    .putObjectRequest(objectRequest)
                    .build();


            PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
            String myURL = presignedRequest.url().toString();
            logger.info("Presigned URL to upload a file to: [{}]", myURL);
            logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Use a `SdkHttpClient` classe AWS for Java V2 para fazer o upload.

```
    /**
     * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT request.
Since the
     * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
     *
     * @param presignedUrlString The URL for the PUT request.
     * @param fileToPut File to uplaod
     */
```

```
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());

            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Excluir uploads multiparte incompletos

O exemplo de código a seguir mostra como excluir ou interromper uploads multiparte incompletos do Amazon S3.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Para interromper uploads multiparte que estão em andamento ou incompletos por qualquer motivo, você pode obter uma lista de uploads para excluí-los, conforme mostrado no exemplo a seguir.

```
    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     */
    public static void abortIncompleteMultipartUploadsFromList(String bucketName) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();
```

```
        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
            }
        }
    }
```

Para excluir uploads multiparte incompletos que foram iniciados antes ou depois de uma data, você pode excluir seletivamente os uploads multiparte com base em um momento específico, conforme mostrado no exemplo a seguir.

```
    /**
     * Aborts incomplete multipart uploads older than the specified point in time.
     *
     * @param bucketName  the name of the S3 bucket
     * @param pointInTime the cutoff time; uploads initiated before this are aborted
     */
    static void abortIncompleteMultipartUploadsOlderThan(String bucketName, Instant
pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
```

```
        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
```

Se você tiver acesso ao ID de upload depois de iniciar um upload multiparte, poderá excluir o upload em andamento usando esse ID.

```
    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key) {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
```

```
        }
    }
```

Para excluir uploads multiparte incompletos após determinado número de dias de maneira consistente, defina uma configuração de ciclo de vida para o bucket. O exemplo a seguir mostra como criar uma regra para excluir uploads multiparte existentes há mais de 7 dias.

```
    /**
     * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
     *
     * @param bucketName the name of the S3 bucket
     */
    static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
        Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
            .abortIncompleteMultipartUpload(b -> b.
                daysAfterInitiation(7))
            .status("Enabled")
            .filter(SdkBuilder::build) // Filter element is required.
            .build());

        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
            .bucket(bucketName)
            .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AbortMultipartUpload
  - ListMultipartUploads

- PutBucketLifecycleConfiguration

Detectar EPI em imagens

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition

- Amazon S3
- Amazon SES

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Baixar “diretórios” do S3

O exemplo de código a seguir mostra como baixar e filtrar o conteúdo de “diretórios” de bucket do Amazon S3.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo mostra como usar o S3TransferManager in AWS SDK for Java 2.x para baixar “diretórios” de um bucket do Amazon S3. Também demonstra como usar DownloadFilters na solicitação.

```
   /**
    * For standard buckets, S3 provides the illusion of a directory structure
through the use of keys. When you upload
    * an object to an S3 bucket, you specify a key, which is essentially the "path"
to the object. The key can contain
    * forward slashes ("/") to make it appear as if the object is stored in a
directory structure, but this is just a
    * logical representation, not an actual directory.
    * <p><pre>
    * In this example, our S3 bucket contains the following objects:
    *
    * folder1/file1.txt
    * folder1/file2.txt
    * folder1/file3.txt
    * folder2/file1.txt
    * folder2/file2.txt
    * folder2/file3.txt
    * folder3/file1.txt
    * folder3/file2.txt
    * folder3/file3.txt
    *
    * When method `downloadS3Directories` is invoked with
    * `destinationPathURI` set to `/test`, the downloaded
    * directory looks like:
    *
    * |- test
    *    |- folder1
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    *    |- folder3
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    * </pre>
    *
    * @param transferManager    An S3TransferManager instance.
    * @param destinationPathURI local directory to hold the downloaded S3
'directories' and files.
```

```
     * @param bucketName         The S3 bucket that contains the 'directories' to
 download.
     * @return The number of objects (files, in this case) that were downloaded.
     */
    public Integer downloadS3Directories(S3TransferManager transferManager,
                                         URI destinationPathURI, String bucketName)
 {

        // Define the filters for which 'directories' we want to download.
        DownloadFilter folder1Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder1/");
        DownloadFilter folder3Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder3/");
        DownloadFilter folderFilter = s3Object ->
 folder1Filter.or(folder3Filter).test(s3Object);

        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .filter(folderFilter)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
 Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
 Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2` was
 not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object failed to transfer  [{}]",
 fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
```

```
    }
```

Fazer download de objetos em um diretório local

O exemplo de código a seguir mostra como fazer download de todos os objetos de um bucket do Amazon Simple Storage Service (Amazon S3) em um diretório local.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use an S3TransferManager para baixar todos os objetos do S3 no mesmo bucket do S3. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
```

```
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- Para obter detalhes da API, consulte DownloadDirectory em Referência AWS SDK for Java 2.x da API.

Bloquear objetos do Amazon S3

O exemplo de código a seguir mostra como trabalhar com os recursos de bloqueio de objetos do S3.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do Amazon S3.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
```

```
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        final String usage = """
            Usage:
                <bucketName> \s

            Where:
                bucketName - The Amazon S3 bucket name.
           """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
 Locking Feature Scenario.");
        System.out.println("Press Enter to continue...");
```

```
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
```

```
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by selecting
one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
                }
```

```
                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to view:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
```

```
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }

    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key, false);
                    }
                }
                // Check for a retention period.
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
```

```
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking features.
                """);

        System.out.println("S3 buckets can be created either with or without object
lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();
```

```
        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on individual
files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){
```

```
                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put a
legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance retention
period to %s in %s (y/n)?
                                Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
    }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
```

```
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

Uma classe de wrapper para funções do S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
```

```
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time zone.
        ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
 HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
 until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
```

```
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
           System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
               ":\n\tStatus: " + response.legalHold().status());
           return response.legalHold();

       } catch (S3Exception ex) {
           System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
"'");
       }

       return null;
   }

   // Create a new Amazon S3 bucket with object lock options.
   public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
       S3Waiter s3Waiter = getClient().waiter();
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .objectLockEnabledForBucket(enableObjectLock)
           .build();

       getClient().createBucket(bucketRequest);
       HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
           .bucket(bucketName)
           .build();

       // Wait until the bucket is created and print out the response.
       s3Waiter.waitUntilBucketExists(bucketRequestWait);
       System.out.println(bucketName + " is ready");
   }
```

```
   public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
       AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
       return bucketNames.stream()
           .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
               .map(version -> {
                   S3InfoObject s3InfoObject = new S3InfoObject();
                   s3InfoObject.setBucketName(bucketName);
                   s3InfoObject.setVersion(version.versionId());
                   s3InfoObject.setKeyName(version.key());
                   return s3InfoObject;
               }))
           .peek(s3InfoObject -> {
               int i = counter.incrementAndGet(); // Increment and get the updated
value.
               if (interactive) {
                   System.out.println(i + ": "+ s3InfoObject.getKeyName());
                   System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                   System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
               }
           })
           .collect(Collectors.toList());
   }

   public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
       ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
           .bucket(bucketName)
           .build();

       return getClient().listObjectVersions(versionsRequest);
   }

   // Set or modify a retention period on an S3 bucket.
   public void modifyBucketDefaultRetention(String bucketName) {
       VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
           .mfaDelete(MFADelete.DISABLED)
           .status(BucketVersioningStatus.ENABLED)
           .build();
```

```
        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();

        getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName +".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
```

```
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on "+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage() +
"'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath) {
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
        ObjectLockLegalHold legalHold ;
```

```
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey, boolean
hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }

    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
```

```
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Gerenciar mensagens grandes usando o S3

O exemplo de código a seguir mostra como usar a biblioteca do cliente em versão ampliada do Amazon SQS para trabalhar com mensagens grandes do SQS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
```

```
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
```

```
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
```

```
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
```

```
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK for Java 2.x.
- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

Analisar URIs

O exemplo de código a seguir mostra como usar analisar URIs do Amazon S3 para extrair componentes importantes como o nome do bucket e a chave do objeto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Analise um URI do Amazon S3 usando a classe S3Uri.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
```

```
import java.util.Map;

    /**
     *
     * @param s3Client    - An S3Client through which you acquire an S3Uri instance.
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate S3Uri
     *                      capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri() method.
        URI uri = URI.create(s3ObjectUrl);
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
 returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
```

```
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123], partNumber=[77,
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
 s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

        String partNumber =
 s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
 s3Uri.firstMatchingRawQueryParameters("partNumber");
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters, must
 be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".", the
 dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
```

```
            logger.info("{}: {}", s3UriElement, element);
        }
    }
```

Processar notificações de eventos do S3

O exemplo de código a seguir mostra como trabalhar com notificações de eventos do S3 de uma forma orientada a objetos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo mostra como processar o evento de notificação do S3 usando o Amazon SQS.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload and
 logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification API</
a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
     * <groupId>software.amazon.awssdk</groupId>
     * <artifactId>s3-event-notifications</artifactId>
     * <version><LATEST></version>
     * </dependency>
```

```
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String queueArn)
{
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                       ).thenAccept(attributeResponse ->
                               logger.info("Approximate number of messages in the
queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                       .join();

               // Receive the messages.
```

```
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}", eventName,
key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {
                    sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
```

```
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Enviar notificações de eventos para EventBridge

O exemplo de código a seguir mostra como habilitar um bucket para enviar notificações de eventos do S3 EventBridge e encaminhá-las para um tópico do Amazon SNS e uma fila do Amazon SQS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
```

```
        try {
            // Enable bucket to emit S3 Event notifications to EventBridge.
            s3Client.putBucketNotificationConfiguration(b -> b
                    .bucket(bucketName)
                    .notificationConfiguration(b1 -> b1
                            .eventBridgeConfiguration(
                                    SdkBuilder::build)
                    ).build()).join();

            // Create an EventBridge rule to route Object Created notifications.
            PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                    .name(RULE_NAME)
                    .eventPattern("""
                            {
                              "source": ["aws.s3"],
                              "detail-type": ["Object Created"],
                              "detail": {
                                "bucket": {
                                  "name": ["%s"]
                                }
                              }
                            }
                            """.formatted(bucketName))
                    .build();

            // Add the rule to the default event bus.
            PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                    .whenComplete((r, t) -> {
                        if (t != null) {
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
```

```
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Monitorar uploads e downloads

O exemplo de código a seguir mostra como usar o Amazon S3 para carregar ou baixar arquivos grandes.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Acompanhe o andamento do carregamento de um arquivo.

```
   public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                            String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
               .putObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .source(Paths.get(filePathURI))
               .build();

       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

       fileUpload.completionFuture().join();
       /*
           The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
           You can also implement the interface to provide your own logic.

           Configure log4J2 with settings such as the following.
               <Configuration status="WARN">
                   <Appenders>
                       <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
                           <PatternLayout pattern="%m%n"/>
                       </Console>
                   </Appenders>

                   <Loggers>
                       <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                           <AppenderRef ref="AlignedConsoleAppender"/>
                       </logger>
                   </Loggers>
               </Configuration>

           Log4J2 logs the progress. The following is example output for a 21.3 MB
file upload.
               Transfer initiated...
               |                    | 0.0%
               |====                | 21.1%
               |============        | 60.5%
               |====================| 100.0%
               Transfer complete!
```

```
        */
    }
```

Acompanhe o andamento do download de um arquivo.

```
   public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                            String key, String downloadedFileWithPath) {
       DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
               .getObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .destination(Paths.get(downloadedFileWithPath))
               .build();

       FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

       CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
       /*
           The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
           You can also implement the interface to provide your own logic.

           Configure log4J2 with settings such as the following.
               <Configuration status="WARN">
                   <Appenders>
                       <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
                           <PatternLayout pattern="%m%n"/>
                       </Console>
                   </Appenders>

                   <Loggers>
                       <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                           <AppenderRef ref="AlignedConsoleAppender"/>
                       </logger>
                   </Loggers>
               </Configuration>
```

```
            Log4J2 logs the progress. The following is example output for a 21.3 MB
file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetObject
  - PutObject

Fazer upload em um bucket

O exemplo de código a seguir mostra como fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service (Amazon S3).

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use an S3TransferManager para fazer upload de um diretório local. Veja o arquivo completo e teste.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;
```

```
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;

    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
 transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
 directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
 fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- Para obter detalhes da API, consulte UploadDirectory em Referência AWS SDK for Java 2.x da API.

Fazer upload ou download de arquivos grandes

O exemplo de código a seguir mostra como fazer upload ou download de arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte Carregar um objeto usando carregamento fracionado.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Chame funções que transferem arquivos de e para um bucket do S3 usando o. S3TransferManager

```
   public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
           URI destinationPathURI, String bucketName) {
       DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
               .destination(Paths.get(destinationPathURI))
               .bucket(bucketName)
               .build());
       CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

       completedDirectoryDownload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryDownload.failedTransfers().size();
   }
```

Carregue um diretório local inteiro.

```
   public Integer uploadDirectory(S3TransferManager transferManager,
           URI sourceDirectory, String bucketName) {
       DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
               .source(Paths.get(sourceDirectory))
               .bucket(bucketName)
               .build());

       CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
       completedDirectoryUpload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryUpload.failedTransfers().size();
   }
```

Carregue um único arquivo.

```
   public String uploadFile(S3TransferManager transferManager, String bucketName,
                            String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
           .putObjectRequest(b -> b.bucket(bucketName).key(key))
```

```
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Os exemplos de código usam as importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Use o S3 Transfer Manager sobre o cliente AWS CRT-based S3 para realizar de forma transparente um upload de várias partes quando o tamanho do conteúdo exceder um limite. O limite padrão é 8 MB.

```
    /**
     * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the file path of the file to be uploaded
     */
    public void multipartUploadWithTransferManager(String bucketName, String key,
String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b
                .bucket(bucketName)
                .key(key))
            .source(Paths.get(filePath))
            .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

Use a API S3Client para realizar um carregamento fracionado.

```
    /**
     * Performs a multipart upload to Amazon S3 using the provided S3 client.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the path to the file to be uploaded
     */
    public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
```

```
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
```

```
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
   }
```

Use a S3AsyncClient API com suporte a várias partes ativado para realizar um upload de várias partes.

```
   /**
    * Uploads a file to an S3 bucket using the S3AsyncClient and enabling multipart
support.
    *
    * @param bucketName the name of the S3 bucket
    * @param key        the object key
    * @param filePath   the local file path of the file to be uploaded
    */
   public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
       // Enable multipart support.
       S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
           .multipartEnabled(true)
           .build();

       CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b ->
b
               .bucket(bucketName)
               .key(key),
           Paths.get(filePath));

       response.join();
       logger.info("File uploaded in multiple 8 MiB parts using S3AsyncClient.");
   }
```

Upload de fluxo de tamanho desconhecido

O exemplo de código a seguir mostra como fazer upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o AWS CRT-based S3 Client.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsync {
    private static final Logger logger =
 LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
            S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
            PutObjectResponse putObjectResponse =
 example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
```

```
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown size,
 use can the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient s33CrtAsyncClient,
 String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Use o cliente do S3 assíncrono padrão com o upload de várias partes ativado.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {
    private static final Logger logger =
 LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsyncMp example = new PutObjectFromStreamAsyncMp();
            S3AsyncClient s3AsyncClientMp =
 S3AsyncClient.builder().multipartEnabled(true).build();
            PutObjectResponse putObjectResponse =
 example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
```

```
    }

    /**
     * @param s3AsyncClientMp - To upload content from a stream of unknown size, use
 can the S3 asynchronous client with multipart enabled.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
 String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key), body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Use o Gerenciador de transferências do Amazon S3.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
 LoggerFactory.getLogger(UploadStream.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
 example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
 completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param transferManager - To upload content from a stream of unknown size, you
 can use the S3TransferManager based on the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
```

```
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
executor);

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        CompletedUpload completedUpload = upload.completionFuture().join();
        executor.shutdown();
        return completedUpload;
    }
}
```

Usar somas de verificação

O exemplo de código a seguir mostra como usar somas de verificação para trabalhar com um objeto do Amazon S3.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Os exemplos de código usam um subconjunto das importações a seguir.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
```

```
import java.util.UUID;
```

Especifique um algoritmo de soma de verificação para o método `putObject` ao criar `PutObjectRequest`.

```
    public void putObjectWithChecksum(String bucketName, String key) {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
            RequestBody.fromString("This is a test"));
    }
```

Verifique a soma de verificação do `getObject` método ao criar o. GetObjectRequest

```
    public GetObjectResponse getObjectWithChecksum(String bucketName, String key) {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
            .response();
    }
```

Pre-calculate uma soma de verificação para o `putObject` método quando você cria o. `PutObjectRequest`

```
    public void putObjectWithPrecalculatedChecksum(String bucketName, String key,
 String filePath) {
        String checksum = calculateChecksum(filePath, "SHA-256");

        s3Client.putObject((b -> b
                .bucket(bucketName)
                .key(key)
                .checksumSHA256(checksum)),
            RequestBody.fromFile(Paths.get(filePath)));
    }
```

Use o S3 Transfer Manager sobre o cliente AWS CRT-based S3 para realizar de forma transparente um upload de várias partes quando o tamanho do conteúdo exceder um limite. O limite padrão é 8 MB.

Você pode especificar um algoritmo de soma de verificação para o SDK usar. Por padrão, o SDK usa o algoritmo CRC32.

```
   public void multipartUploadWithChecksumTm(String bucketName, String key, String
filePath) {
       S3TransferManager transferManager = S3TransferManager.create();
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
           .putObjectRequest(b -> b
               .bucket(bucketName)
               .key(key)
               .checksumAlgorithm(ChecksumAlgorithm.SHA1))
           .source(Paths.get(filePath))
           .build();
       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
       fileUpload.completionFuture().join();
       transferManager.close();
   }
```

Use a API ou (S3AsyncClient API) do S3Client para realizar um upload de várias partes. Se você especificar uma soma de verificação adicional, deverá indicar o algoritmo a ser usado no início do carregamento. Você também deve especificar o algoritmo para a solicitação de cada parte e fornecer a soma de verificação calculada para cada parte após o carregamento.

```
   public void multipartUploadWithChecksumS3Client(String bucketName, String key,
String filePath) {
       ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

       // Initiate the multipart upload.
       CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
           .bucket(bucketName)
           .key(key)
           .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
       String uploadId = createMultipartUploadResponse.uploadId();

       // Upload the parts of the file.
       int partNumber = 1;
```

```
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .checksumAlgorithm(algorithm) // Checksum specified on each
part.
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
```

```
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
   }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

Exemplos do Amazon S3 Control usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon S3 Control.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon S3 Control

O exemplo de código a seguir mostra como começar a usar o Amazon S3 Control.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
```

```
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have not
 configured
 * authentication for SDKs and tools,see https://docs.aws.amazon.com/sdkref/latest/
guide/access.html in the AWS SDKs and Tools Reference Guide.
 * <p/>
 * You must have a runtime environment configured with the Java SDK.
 * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 the Developer Guide if this is not set up.
 */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
 ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }
```

```
    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
```

```
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation is
finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)
            .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como aprender as principais operações do Amazon S3 Control.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Conheça as principais operações.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;

public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing of
 large-scale
            data stored in Amazon S3. It automatically scales resources to handle
 varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
 operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove tags
 on thousands or millions of
```

```
            objects in a single operation, streamlining the management and
 organization of their data.

            This can be particularly useful for tasks such as cost allocation,
 lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
 essential.
            S3 Batch's scalability and serverless nature make it an ideal solution
 for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this scenario.
        System.out.println("Use CloudFormation to stand up the resource required for
 this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
 CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
```

```
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
             In this step, we modify the job priority value. The higher the number,
the higher the priority.
             So, a job with a priority of `30` would have a higher priority than a
job with
             a priority of `20`. This is a common way to represent the priority of a
task
             or job, with higher numbers indicating a higher priority.

             Ensure that the job status allows for priority updates. Jobs in
certain
             states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
             updated. Only jobs in the Active or Suspended state typically allow
priority
             updates.
             """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
```

```
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " + ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.cancelJobAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Cancel job failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to cancel job: " + ex.getMessage());
            }
        } else {
            System.out.println("Job " +jobId +" was not canceled.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Describe the job that was just created");
        waitForInputToContinue(scanner);
        try {
            actions.describeJobAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Describe job failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to describe job: " + ex.getMessage());
        }
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the tags associated with the job");
        waitForInputToContinue(scanner);
        try {
            actions.getJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Get job tags failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to get job tags: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update Batch Job Tags");
        waitForInputToContinue(scanner);
        try {
            actions.putJobTaggingAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Put job tagging failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to put job tagging: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the Amazon S3 Batch job tagging.");
        System.out.print("Do you want to delete Batch job tagging? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.deleteBatchJobTagsAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                        return null;
```

```
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
            }
        } else {
            System.out.println("Tagging was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println();
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println();
                break;
            } else {
```

```
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }


}
```

Uma classe de ação que envolve operações.

```
public class S3BatchActions {

    private static S3ControlAsyncClient asyncClient;

    private static S3AsyncClient s3AsyncClient ;
    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
```

```
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    private static S3AsyncClient getS3AsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
```

```
        }
        return s3AsyncClient;
    }


    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
     */
```

```
   public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
       UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .priority(60)
           .build();

       CompletableFuture<Void> future = new CompletableFuture<>();
       getAsyncClient().updateJobPriority(priorityRequest)
           .thenAccept(response -> {
               System.out.println("The job priority was updated");
               future.complete(null); // Complete the CompletableFuture on
successful execution
           })
           .exceptionally(ex -> {
               System.err.println("Failed to update job priority: " +
ex.getMessage());
               future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
               return null; // Return null to handle the exception
           });

       return future;
   }

   /**
    * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
    *
    * @param jobId     the ID of the job for which to retrieve the tags
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
    * @throws RuntimeException if an error occurs while retrieving the job tags
    */
   public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
       GetJobTaggingRequest request = GetJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return asyncClient.getJobTagging(request)
```

```
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Asynchronously deletes the tags associated with a specific batch job.
     *
     * @param jobId     The ID of the batch job whose tags should be deleted.
     * @param accountId The ID of the account associated with the batch job.
     * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
     */
    public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
        DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .build();

        return asyncClient.deleteJobTagging(jobTaggingRequest)
            .thenAccept(response -> {
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

```
    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job description
is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " + response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " + manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
```

```
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId         the AWS account ID associated with the job
     * @param iamRoleArn        the ARN of the IAM role to be used for the job
     * @param manifestLocation  the location of the job manifest file in S3
     * @param reportBucketName  the name of the S3 bucket to store the job report
     * @param uuid              a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of the
 S3 job.
     *         The CompletableFuture will return the job ID if the job is created
 successfully,
     *         or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
 iamRoleArn,
                                                       String manifestLocation,
 String reportBucketName, String uuid) {

        String[] bucketName = new String[]{""};
        String[] parts = reportBucketName.split(":::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
```

```
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
```

```
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }

    /**
     * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3 bucket.
     *
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param key the key (file name) of the object in the Amazon S3 bucket
     * @return the ETag of the object
     */
    public String getETag(String bucketName, String key) {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```

```
        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        return headObjectResponse.eTag();
    }

    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId     the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    // Setup the S3 bucket required for this scenario.
    /**
```

```
     * Creates an Amazon S3 bucket with the specified name.
     *
     * @param bucketName the name of the S3 bucket to create
     * @throws S3Exception if there is an error creating the bucket
     */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();


            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
```

```
        Path filePath = Paths.get("src/main/resources/batch/",
 fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        CompletableFuture<PutObjectResponse> future =
 getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into bucket
 " + bucketName);
            }
        }).join();
    }


    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
   /**
    * Deletes an object from an Amazon S3 bucket asynchronously.
    *
    * @param bucketName The name of the S3 bucket where the object is stored.
    * @param objectName The name of the object to be deleted.
    * @return A {@link CompletableFuture} that completes when the object has been
deleted,
    *         or throws a {@link RuntimeException} if an error occurs during the
deletion.
    */
   public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
objectName) {
       ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
       toDelete.add(ObjectIdentifier.builder()
           .key(objectName)
           .build());

       DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
           .bucket(bucketName)
           .delete(Delete.builder()
               .objects(toDelete).build())
           .build();

       return getS3AsyncClient().deleteObjects(dor)
           .thenAccept(result -> {
               System.out.println("The object was deleted!");
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
           });
   }

   /**
    * Deletes a folder and all its contents asynchronously from an Amazon S3
bucket.
    *
    * @param bucketName the name of the S3 bucket containing the folder to be
deleted
    * @return a {@link CompletableFuture} that completes when the folder and its
contents have been deleted
    * @throws RuntimeException if any error occurs during the deletion process
    */
```

```
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(folderName)
            .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(obj.key())
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
                .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
                .thenCompose(v -> {
                    // Delete the folder.
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(folderName)
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted folder: " + folderName);
                            return deleteResponse;
                        });
                });
        }).join();
    }
```

```
    /**
     * Deletes an Amazon S3 bucket.
     *
     * @param bucketName the name of the bucket to delete
     * @return a {@link CompletableFuture} that completes when the bucket has been
deleted, or exceptionally if there is an error
     * @throws RuntimeException if there is an error deleting the bucket
     */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " + bucketName,
ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or uploading
the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " + bucketName);
    }

    /**
```

```
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     * @param fileNames an array of file names to be deleted from the bucket
     * @param actions the S3BatchActions instance to be used for the file deletion
     * @throws IOException if an I/O error occurs during the file deletion
     */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
                   actions.deleteBucketObjects(bucketName, fileName)
                  .thenRun(() -> System.out.println("Object deletion completed"))
                  .exceptionally(ex -> {
                      System.err.println("Error occurred: " + ex.getMessage());
                      return null;
                  });
        }
        System.out.println("All files have been deleted from the bucket " +
bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\\batch";
        Path path = Paths.get(directoryPath);

        try {
            // Create the directory if it doesn't exist.
            if (Files.notExists(path)) {
                Files.createDirectories(path);
                System.out.println("Created directory: " + path.toString());
            } else {
                System.out.println("Directory already exists: " + path.toString());
            }

            for (String fileName : fileNames) {
                // Check if the file is a .txt file.
                if (fileName.endsWith(".txt")) {
                    // Define the path for the new file.
                    Path filePath = path.resolve(fileName);
                    System.out.println("Attempting to create file: " +
filePath.toString());

                    // Create and write content to the new file.
```

```
                    Files.write(filePath, "This is a test".getBytes());

                    // Verify the file was created.
                    if (Files.exists(filePath)) {
                        System.out.println("Successfully created file: " +
 filePath.toString());
                    } else {
                        System.out.println("Failed to create file: " +
 filePath.toString());
                    }
                }
            }

        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public String getAccountId() {
        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetCallerIdentityResponse callerIdentityResponse =
 stsClient.getCallerIdentity();
        return callerIdentityResponse.account();
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority

- UpdateJobStatus

Ações

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um trabalho assíncrono do S3.

```
   /**
    * Creates an asynchronous S3 job using the AWS Java SDK.
    *
    * @param accountId         the AWS account ID associated with the job
    * @param iamRoleArn        the ARN of the IAM role to be used for the job
    * @param manifestLocation  the location of the job manifest file in S3
    * @param reportBucketName  the name of the S3 bucket to store the job report
    * @param uuid              a unique identifier for the job
    * @return a CompletableFuture that represents the asynchronous creation of the
S3 job.
    *         The CompletableFuture will return the job ID if the job is created
successfully,
    *         or throw an exception if there is an error.
    */
   public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                     String manifestLocation,
String reportBucketName, String uuid) {

       String[] bucketName = new String[]{""};
       String[] parts = reportBucketName.split(":::");
       if (parts.length > 1) {
           bucketName[0] = parts[1];
       } else {
```

```
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
 S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
 JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();
```

```
                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }
```

Crie um trabalho de retenção de conformidade.

```
    /**
```

```
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
 to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();
```

```
        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Crie uma suspensão legal do trabalho.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
with
```

```
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
```

```
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Crie um trabalho de retenção de governança.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = """

            Usage:
                <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
 <manifestObjectVersionId>
```

```
            Where:
                manifestObjectArn - The Amazon Resource Name (ARN) of the S3 object
that contains the manifest file for the governance objects.\s
                bucketName - The ARN of the S3 bucket where the job report will be
stored.
                roleArn - The ARN of the IAM role that will be used to perform the
governance retention operation.
                accountId - Your AWS account Id.
                manifestObjectVersionId =  A unique value that is used as the `eTag`
property of the `JobManifestLocation` object.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
roleArn, String accountId, String manifestObjectVersionId) throws ParseException {
        final JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        final JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
            .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
            .build();

        final JobManifest manifestToPublicApi = JobManifest.builder()
```

```
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        final String jobReportPrefix = "reports/governance-objects";
        final JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        final Date jan30th = format.parse("30/01/2025");

        final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.GOVERNANCE)
                .retainUntilDate(jan30th.toInstant())
                .build())
            .build();

        final JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(s3SetObjectRetentionOperation)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
```

```
    }
}
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK for Java 2.x da API.

DeleteJobTagging

O código de exemplo a seguir mostra como usar `DeleteJobTagging`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously deletes the tags associated with a specific batch job.
    *
    * @param jobId     The ID of the batch job whose tags should be deleted.
    * @param accountId The ID of the account associated with the batch job.
    * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
    */
   public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
       DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .build();

       return asyncClient.deleteJobTagging(jobTaggingRequest)
           .thenAccept(response -> {
               System.out.println("You have successfully deleted " + jobId + "
tagging.");
           })
           .exceptionally(ex -> {
               System.err.println("Failed to delete job tags: " + ex.getMessage());
               throw new RuntimeException(ex);
```

```
            });
    }
```

- Para obter detalhes da API, consulte DeleteJobTagging em Referência AWS SDK for Java 2.x da API.

DescribeJob

O código de exemplo a seguir mostra como usar `DescribeJob`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job description
is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " + response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
```

```
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " + manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- Para obter detalhes da API, consulte DescribeJob em Referência AWS SDK for Java 2.x da API.

`GetJobTagging`

O código de exemplo a seguir mostra como usar `GetJobTagging`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
     *
     * @param jobId     the ID of the job for which to retrieve the tags
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
     * @throws RuntimeException if an error occurs while retrieving the job tags
     */
    public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
        GetJobTaggingRequest request = GetJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Para obter detalhes da API, consulte GetJobTagging em Referência AWS SDK for Java 2.x da API.

`PutJobTagging`

O código de exemplo a seguir mostra como usar `PutJobTagging`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Asynchronously adds tags to a job in the system.
    *
    * @param jobId     the ID of the job to add tags to
    * @param accountId the account ID associated with the job
    * @return a CompletableFuture that completes when the tagging operation is
finished
    */
   public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
       S3Tag departmentTag = S3Tag.builder()
           .key("department")
           .value("Marketing")
           .build();

       S3Tag fiscalYearTag = S3Tag.builder()
           .key("FiscalYear")
           .value("2020")
           .build();

       PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .tags(departmentTag, fiscalYearTag)
           .build();

       return asyncClient.putJobTagging(putJobTaggingRequest)
```

```
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Para obter detalhes da API, consulte PutJobTagging em Referência AWS SDK for Java 2.x da API.

`UpdateJobPriority`

O código de exemplo a seguir mostra como usar `UpdateJobPriority`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates the priority of a job asynchronously.
    *
    * @param jobId     the ID of the job to update
    * @param accountId the ID of the account associated with the job
    * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
    */
   public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
       UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
```

```
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }
```

- Para obter detalhes da API, consulte UpdateJobPriority em Referência AWS SDK for Java 2.x da API.

`UpdateJobStatus`

O código de exemplo a seguir mostra como usar `UpdateJobStatus`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
```

```
    * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
    *         If an error occurs during the update, the returned future will
complete exceptionally.
    */
   public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
       UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
           .build();

       return asyncClient.updateJobStatus(updateJobStatusRequest)
           .thenAccept(updateJobStatusResponse -> {
               System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
           })
           .exceptionally(ex -> {
               System.err.println("Failed to cancel job: " + ex.getMessage());
               throw new RuntimeException(ex); // Propagate the exception
           });
   }
```

- Para obter detalhes da API, consulte UpdateJobStatus em Referência AWS SDK for Java 2.x da API.

Exemplos de buckets de diretório do S3 usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com S3 Directory Buckets.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon S3 directory buckets

O exemplo de código a seguir mostra como começar a usar buckets de diretório do Amazon S3.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.s3.directorybucket;


import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
```

```
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        Region region = Region.US_WEST_2;
```

```
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected error:
{}", e.getMessage(), e);
            } finally {
                s3Client.close();
            }
        }
    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
```

```
   public static void createDirectoryBucket(S3Client s3Client, String bucketName,
String zone) throws S3Exception {
       logger.info("Creating bucket: {}", bucketName);

       CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
               .location(LocationInfo.builder()
                       .type(LocationType.AVAILABILITY_ZONE)
                       .name(zone).build())
               .bucket(BucketInfo.builder()
                       .type(BucketType.DIRECTORY)
                       .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                       .build())
               .build();
       try {
           CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                   .bucket(bucketName)
                   .createBucketConfiguration(bucketConfiguration).build();
           CreateBucketResponse response = s3Client.createBucket(bucketRequest);
           logger.info("Bucket created successfully with location: {}",
response.location());
       } catch (S3Exception e) {
           logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode(), e);
           throw e;
       }
   }

   /**
    * Lists all S3 directory buckets.
    *
    * @param s3Client The S3 client used to interact with S3
    * @return A list of bucket names
    */
   public static List<String> listDirectoryBuckets(S3Client s3Client) {
       logger.info("Listing all directory buckets");

       try {
           // Create a ListBucketsRequest
           ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

           // Retrieve the list of buckets
```

```
            ListDirectoryBucketsResponse response =
 s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        try {
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            s3Client.deleteBucket(deleteBucketRequest);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: " + bucketName + " - Error code:
 " + e.awsErrorDetails().errorCode(),
                    e);
            throw e;
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateBucket

- ListDirectoryBuckets

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Configure uma VPC e um endpoint da VPC.
- Configure as políticas, os perfis e o usuário para trabalhar com os buckets de diretório do S3 e a classe de armazenamento S3 Express One Zone.
- Crie dois clientes do S3.
- Crie dois buckets.
- Crie um objeto e copie-o.
- Demonstre a diferença de desempenho.
- Preencha os buckets para mostrar a diferença lexicográfica.
- Confirme se o usuário deseja limpar os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon S3.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;
```

```
   private static String mdirectoryBucketName;
   private static String mregularBucketName;

   private static String stackName = "cfn-stack-s3-express-basics--" +
UUID.randomUUID();

   private static String regularUser = "";
   private static String vpcId = "";
   private static String expressUser = "";

   private static String vpcEndpointId = "";

   private static final S3DirectoriesActions s3DirectoriesActions = new
S3DirectoriesActions();

   public static void main(String[] args) {
       try {
           s3ExpressScenario();
       } catch (RuntimeException e) {
           logger.info(e.getMessage());
       }
   }

   // Runs the scenario.
   private static void s3ExpressScenario() {
       logger.info(DASHES);
       logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK for
Java V2.");
       logger.info("""
           Let's get started! First, please note that S3 Express One Zone works
best when working within the AWS infrastructure,
           specifically when working in the same Availability Zone (AZ). To see the
best results in this example and when you implement
           directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
           bucket.
           """);
       waitForInputToContinue(scanner);
       logger.info(DASHES);

       // Create an optional VPC and create 2 IAM users.
       UserNames userNames = createVpcUsers();
       String expressUserName = userNames.getExpressUserName();
```

```
        String regularUserName = userNames.getRegularUserName();

        //  Set up two S3 clients, one regular and one express,
        //  and two buckets, one regular and one directory.
        setupClientsAndBuckets(expressUserName, regularUserName);

        // Create an S3 session for the express S3 client and add objects to the
buckets.
        logger.info("Now let's add some objects to our buckets and demonstrate how
to work with S3 Sessions.");
        waitForInputToContinue(scanner);
        String bucketObject = createSessionAddObjects();

        // Demonstrate performance differences between regular and directory
buckets.
        demonstratePerformance(bucketObject);

        // Populate the buckets to show the lexicographical difference between
        // regular and express buckets.
        showLexicographicalDifferences(bucketObject);

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3 Express
One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /*
      Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {
                s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            }
            logger.info("Deleted directory bucket " + mdirectoryBucketName);
            mdirectoryBucketName = null;
            if (mregularBucketName != null) {
```

```
                s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
            }
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {
        logger.info(DASHES);
        logger.info("""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
            when object names are listed. Now let's explore how directory buckets
store
            objects in a different manner to regular buckets. The key is in the
name
            "Directory". Where regular buckets store their key/value pairs in a
            flat manner, directory buckets use actual directories/folders.
            This allows for more rapid indexing, traversing, and therefore
            retrieval times!

            The more segmented your bucket is, with lots of
            directories, sub-directories, and objects, the more efficient it
becomes.
            This structural difference also causes `ListObject` operations to
behave
            differently, which can cause unexpected results. Let's add a few more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);
```

```
        //  Populate a few more files in each bucket so that we can use
        //  ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof NoSuchBucketException) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        try {
            // List objects in both S3 buckets.
            List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient, mregularBucketName).join();

            logger.info("Directory bucket content");
            for (String obj : dirBucketObjects) {
                logger.info(obj);
            }
```

```
            logger.info("Regular bucket content");
            for (String obj : regBucketObjects) {
                logger.info(obj);
            }
        } catch (CompletionException e) {
            logger.error("Async operation failed: {} ", e.getCause().getMessage());
            return;
        }

        logger.info("""
            Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
            because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
            creates directories and uses the object "key" as a path to the object.
            """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from a
directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and measures
the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this example
is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("""
```

```
            Now, let's do a performance test. We'll download the same object from
each
            bucket repeatedly and compare the total time needed.

            Note: the performance difference will be much more pronounced if this
            example is run in an EC2 instance in the same Availability Zone as
            the bucket.
            """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

        // Ask if the user wants to download a different number.
        logger.info("Would you like to download the file a different number of
times? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            int maxDownloads = 1_000_000;

            // Ask for a valid number of downloads.
            while (true) {
                logger.info("Enter a number between 1 and " + maxDownloads + " for
the number of downloads: ");
                if (scanner.hasNextInt()) {
                    downloads = scanner.nextInt();
                    if (downloads >= 1 && downloads <= maxDownloads) {
                        break;
                    } else {
                        logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                    }
                } else {
                    logger.info("Invalid input. Please enter a valid integer.");
                    scanner.next();
                }
            }

            logger.info("You have chosen to download {}  items.", downloads);
        } else {
            logger.info("No changes made. Using default downloads: {}", downloads);
        }
        // Simulating the download process for the directory bucket.
```

```
        logger.info("Downloading from the directory bucket.");
        long directoryTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                // Get the object from the directory bucket.
                s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
        long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

        // Download from the regular bucket.
        logger.info("Downloading from the regular bucket.");
        long normalTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
```

```
                }
                return;
            }
        }

        long normalTimeDifference = System.nanoTime() - normalTimeStart;
        logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
        long difference = normalTimeDifference - directoryTimeDifference;
        logger.info("That's a difference of " + difference + " nanoseconds, or");
        logger.info(difference / 1_000_000_000.0 + " seconds.");

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if you
are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAddObjects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
```

```
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with permission
to
            work with directory buckets.

            It's important to remember the user permissions when interacting with
            directory buckets. Instead of validating permissions on every call as
            regular buckets do, directory buckets utilize the user credentials and
session
            token to validate. This allows for much faster connection speeds on
every call.
            For single calls, this is low, but for many concurrent calls
            this adds up to a lot of time saved.
            """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
     *     <li>Creates two IAM users: one with S3 Express One Zone permissions and
one without.</li>
     * </ol>
     *
     * @return a {@link UserNames} object containing the names of the created IAM
users
     */
    public static UserNames createVpcUsers() {
        /*
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
without.
```

```
        */
        logger.info(DASHES);
        logger.info("""
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("""
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                """);
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("""
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            """);
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
```

```
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");

        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the stack
outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express permissions
     * @param regularUserName the username for the user with regular S3 permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
```

```
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and one
uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
```

```
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info("""
            We can now use the ExpressUser client to make calls to S3 Express
operations.
            """);
        waitForInputToContinue(locscanner);
        logger.info(DASHES);
        logger.info("""
            4. Create two buckets.
            Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
            behave differently from regular S3 buckets which we will explore here.
We'll also create a regular bucket, put
            an object into the regular bucket, and copy it to the directory bucket.
            """);

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory bucket.
            We'll choose one that is supported.
            """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
            regularBucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
```

```
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
bucket."
             """);

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-s3"; //
Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }
```

```
   /**
    * Creates an asynchronous S3 client with the specified access key and secret
access key.
    *
    * @param accessKeyId     the AWS access key ID
    * @param secretAccessKey the AWS secret access key
    * @return a {@link CompletableFuture} that asynchronously creates the S3 client
    * @throws IllegalArgumentException if the access key ID or secret access key is
null
    */
   public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
       return CompletableFuture.supplyAsync(() -> {
           // Validate input parameters
           if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey ==
null || secretAccessKey.isBlank()) {
               throw new IllegalArgumentException("Access Key ID and Secret Access
Key must not be null or empty");
           }

           AwsBasicCredentials awsCredentials =
AwsBasicCredentials.create(accessKeyId, secretAccessKey);
           return S3AsyncClient.builder()

.credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
               .region(Region.US_WEST_2)
               .build();
       });
   }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           logger.info("");
           logger.info("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               logger.info("Continuing with the program...");
               logger.info("");
               break;
           } else {
               logger.info("Invalid input. Please try again.");
           }
```

```
        }
    }
}
```

Uma classe de wrapper para os métodos de SDK do Amazon S3.

```
public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            iamAsyncClient = IamAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iamAsyncClient;
    }

    private static Ec2AsyncClient getEc2AsyncClient() {
        if (ec2AsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
```

```
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .httpClient(httpClient)
                .region(Region.US_WEST_2)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes the specified S3 bucket and all the objects within it asynchronously.
     *
     * @param s3AsyncClient the S3 asynchronous client to use for the operations
     * @param bucketName the name of the S3 bucket to be deleted
     * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
     *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
     * @throws CompletionException if there was an error deleting the bucket or its
objects
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
        ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3AsyncClient.listObjectsV2(listRequest)
            .thenCompose(listResponse -> {
                if (!listResponse.contents().isEmpty()) {
                    List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
```

```
                        .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                        .collect(Collectors.toList());

                    DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                        .bucket(bucketName)
                        .delete(Delete.builder().objects(objectIdentifiers).build())
                        .build();

                    return s3AsyncClient.deleteObjects(deleteRequest)
                        .thenAccept(deleteResponse -> {
                            if (!deleteResponse.errors().isEmpty()) {
                                deleteResponse.errors().forEach(error ->
                                    logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                            }
                        });
                }
                return CompletableFuture.completedFuture(null);
            })
            .thenCompose(ignored -> {
                DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3AsyncClient.deleteBucket(deleteBucketRequest);
            })
            .thenCompose(ignored -> {
                S3AsyncWaiter waiter = s3AsyncClient.waiter();
                HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
                return waiter.waitUntilBucketNotExists(headBucketRequest);
            })
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
                    }
                    throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
                }
```

```
                logger.info("Bucket deleted successfully: " + bucketName);
            });
    }

    /**
     *  Lists the objects in an S3 bucket asynchronously.
     *
     * @param s3Client the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the objects to list
     * @return a {@link CompletableFuture} that contains the list of object keys in
the specified bucket
     */
    public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient s3Client,
String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3Client.listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .toList())
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Couldn't list objects in bucket:
" + bucketName, exception);
                }
            });
    }

    /**
     * Retrieves an object from an Amazon S3 bucket asynchronously.
     *
     * @param s3Client   the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName    the unique identifier (key) of the object to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
     */
    public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
```

```
            .build();

        // Get the object asynchronously and transform it into a byte array
        return s3Client.getObject(objectRequest, AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object. Reason:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client           the S3 async client to use for the copy operation
     * @param sourceBucket       the name of the source bucket
     * @param sourceKey          the key of the object to be copied in the source
bucket
     * @param destinationBucket  the name of the destination bucket
     * @param destinationKey     the key of the copied object in the destination
bucket
     * @return a {@link CompletableFuture} that completes when the copy operation is
finished
     */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyRequest = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
```

```
                        throw new CompletionException("Couldn't copy object '" +
sourceKey + "' from bucket '" + sourceBucket + "' to bucket '" + destinationBucket
+ "'. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Failed to copy object",
exception);
                }
            });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client   the S3 asynchronous client to use for creating the session
     * @param bucketName the name of the S3 bucket for which to create the session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse> createSessionAsync(S3AsyncClient
s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the session.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });

    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
```

```
     * @param s3Client   The asynchronous S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The Availability Zone where the bucket will be created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String zone)
{
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
            .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The bucket already exists: "
+ ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
```

```
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
     * @param bucketName  the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
     *         when the bucket is successfully created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createBucket(bucketRequest)
            .thenCompose(response -> {
                S3AsyncWaiter s3Waiter = s3Client.waiter();
                HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3Waiter.waitUntilBucketExists(bucketRequestWait);
            })
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                    }
                }
                logger.info(bucketName + " is ready");
            });
    }

    /**
     * Uploads an object to an Amazon S3 bucket asynchronously.
     *
     * @param s3Client     the S3 async client to use for the upload
     * @param bucketName   the destination S3 bucket name
     * @param bucketObject the name of the object to be uploaded
```

```
     * @param text         the content to be uploaded as the object
     */
    public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(bucketObject)
            .build();

        return s3Client.putObject(objectRequest, AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof NoSuchBucketException) {
                        throw new CompletionException("The S3 bucket does not exist:
" + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Creates an AWS IAM access key asynchronously for the specified user name.
     *
     * @param userName the name of the IAM user for whom to create the access key
     * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
     */
    public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(userName)
            .build();

        return getIAMAsyncClient().createAccessKey(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Access Key Created.");
                } else {
                    if (exception == null) {
                        Throwable cause = exception.getCause();
```

```
                        if (cause instanceof IamException) {
                            throw new CompletionException("IAM error while creating
 access key: " + cause.getMessage(), cause);
                        } else {
                            throw new CompletionException("Failed to create access
 key: " + exception.getMessage(), exception);
                        }
                    }
                }
            });
    }

    /**
     * Asynchronously selects an Availability Zone ID from the available EC2 zones.
     *
     * @return A {@link CompletableFuture} that resolves to the selected
 Availability Zone ID.
     * @throws CompletionException if an error occurs during the request or
 processing.
     */
    public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
        DescribeAvailabilityZonesRequest zonesRequest =
 DescribeAvailabilityZonesRequest.builder()
            .build();

        return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
            .thenCompose(response -> {
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return null
 if no zones are found
                }

                List<String> zoneIds = zonesList.stream()
                    .map(AvailabilityZone::zoneId) // Get the zoneId (e.g., "usw2-
az1")
                    .toList();

                return CompletableFuture.supplyAsync(() ->
 promptUserForZoneSelection(zonesList, zoneIds))
                    .thenApply(selectedZone -> {
                        // Return only the selected Zone ID (e.g., "usw2-az1").
                        return selectedZone.zoneId();
```

```
                    });
            })
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Selected Availability Zone ID: " + result);
                    } else {
                        logger.info("No availability zone selected.");
                    }
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof Ec2Exception) {
                        throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prompts the user to select an Availability Zone from the given list.
     *
     * @param zonesList the list of Availability Zones
     * @param zoneIds the list of zone IDs
     * @return the selected Availability Zone
     */
    private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String> zoneIds)
{
        Scanner scanner = new Scanner(System.in);
        int index = -1;

        while (index < 0 || index >= zoneIds.size()) {
            logger.info("Select an availability zone:");
            IntStream.range(0, zoneIds.size()).forEach(i ->
                logger.info(i + ": " + zoneIds.get(i))
            );

            logger.info("Enter the number corresponding to your choice: ");
            if (scanner.hasNextInt()) {
                index = scanner.nextInt();
            } else {
```

```
                scanner.next();
            }
        }

        AvailabilityZone selectedZone = zonesList.get(index);
        logger.info("You selected: " + selectedZone.zoneId());
        return selectedZone;
    }

    /**
     * Asynchronously sets up a new VPC, including creating the VPC, finding the
associated route table, and
     * creating a VPC endpoint for the S3 service.
     *
     * @return a {@link CompletableFuture} that, when completed, contains a
AbstractMap with the
     *         VPC ID and VPC endpoint ID.
     */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();
```

```
                DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found for
VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId, routeTableId);
                    });
            })
            .thenCompose(vpcAndRouteTable -> {
                String vpcId = vpcAndRouteTable.getKey();
                String routeTableId = vpcAndRouteTable.getValue();
                Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
                String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

                CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
                    .vpcId(vpcId)
                    .routeTableIds(routeTableId)
                    .serviceName(serviceName)
                    .build();

                return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
                    .thenApply(vpcEndpointResponse -> {
                        String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                        logger.info("VPC Endpoint created: {}", vpcEndpointId);
                        return new AbstractMap.SimpleEntry<>(vpcId, vpcEndpointId);
                    });
            })
            .exceptionally(exception -> {
                Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
```

```
                if (cause instanceof Ec2Exception) {
                    logger.error("EC2 error during VPC setup: {}",
 cause.getMessage(), cause);
                    throw new CompletionException("EC2 error during VPC setup: " +
 cause.getMessage(), cause);
                }

                logger.error("VPC setup failed: {}", cause.getMessage(), cause);
                throw new CompletionException("VPC setup failed: " +
 cause.getMessage(), cause);
            });
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Ações

`AbortMultipartUpload`

O código de exemplo a seguir mostra como usar `AbortMultipartUpload`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Anular um upload fracionado em um bucket de diretórios.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;


    /**
     * Aborts a specific multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID of the multipart upload to abort
     * @return True if the multipart upload is successfully aborted, false otherwise
     */
    public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName,
            String objectKey, String uploadId) {
        logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
 bucketName);
        try {
            // Abort the multipart upload
            AbortMultipartUploadRequest abortMultipartUploadRequest =
 AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .build();

            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            logger.info("Aborted multipart upload: {} for object: {}", uploadId,
 objectKey);
            return true;
        } catch (S3Exception e) {
```

```
            logger.error("Failed to abort multipart upload: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Para obter detalhes da API, consulte AbortMultipartUpload em Referência AWS SDK for Java 2.x da API.

`CompleteMultipartUpload`

O código de exemplo a seguir mostra como usar `CompleteMultipartUpload`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Concluir um upload fracionado em um bucket de diretórios.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * This method completes the multipart upload request by collating all the
     * upload parts.
     *
     * @param s3Client    The S3 client used to interact with S3
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to be uploaded
     * @param uploadId    The upload ID used to track the multipart upload
     * @param uploadParts The list of completed parts
     * @return True if the multipart upload is successfully completed, false
     *         otherwise
     */
    public static boolean completeDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey,
            String uploadId, List<CompletedPart> uploadParts) {
        try {
            CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                    .parts(uploadParts)
                    .build();
            CompleteMultipartUploadRequest completeMultipartUploadRequest =
 CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse response =
 s3Client.completeMultipartUpload(completeMultipartUploadRequest);
            logger.info("Multipart upload completed. ETag: {}", response.eTag());
            return true;
        } catch (S3Exception e) {
```

```
            logger.error("Failed to complete multipart upload: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CompleteMultipartUpload em Referência AWS SDK for Java 2.x da API.

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie um objeto de um bucket de diretório para um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Copies an object from one S3 general purpose bucket to one S3 directory
     * bucket.
     *
     * @param s3Client     The S3 client used to interact with S3
     * @param sourceBucket The name of the source bucket
     * @param objectKey    The key (name) of the object to be copied
     * @param targetBucket The name of the target bucket
     */
    public static void copyDirectoryBucketObject(S3Client s3Client, String
 sourceBucket, String objectKey,
            String targetBucket) {
        logger.info("Copying object: {} from bucket: {} to bucket: {}", objectKey,
 sourceBucket, targetBucket);

        try {
            // Create a CopyObjectRequest
            CopyObjectRequest copyReq = CopyObjectRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(objectKey)
                    .destinationBucket(targetBucket)
                    .destinationKey(objectKey)
                    .build();

            // Copy the object
            CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
            logger.info("Successfully copied {} from bucket {} into bucket {}.
 CopyObjectResponse: {}",
                    objectKey, sourceBucket, targetBucket,
 copyRes.copyObjectResult().toString());

        } catch (S3Exception e) {
            logger.error("Failed to copy object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK for Java 2.x da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para Java 2.x

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket de diretório do S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
```

```
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK for Java 2.x da API.

`CreateMultipartUpload`

O código de exemplo a seguir mostra como usar `CreateMultipartUpload`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um upload fracionado em um bucket de diretório.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * This method creates a multipart upload request that generates a unique upload
     * ID used to track
     * all the upload parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @return The upload ID used to track the multipart upload
     */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                    .bucket(bucketName)
```

```
                    .key(objectKey)
                    .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;

        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte CreateMultipartUpload em Referência AWS SDK for Java 2.x da API.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um bucket de diretório do S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        logger.info("Deleting bucket: {}", bucketName);

        try {
            // Create a DeleteBucketRequest
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket
            s3Client.deleteBucket(deleteBucketRequest);
            logger.info("Successfully deleted bucket: {}", bucketName);

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteBucket em Referência AWS SDK for Java 2.x da API.

`DeleteBucketEncryption`

O código de exemplo a seguir mostra como usar `DeleteBucketEncryption`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua a configuração de criptografia de um bucket de diretório.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Deletes the encryption configuration from an S3 bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        DeleteBucketEncryptionRequest deleteRequest =
 DeleteBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3Client.deleteBucketEncryption(deleteRequest);
            logger.info("Bucket encryption deleted for bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket encryption: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
```

```
    }
```

- Para obter detalhes da API, consulte DeleteBucketEncryption em Referência AWS SDK for Java 2.x da API.

`DeleteBucketPolicy`

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua uma política de um bucket de diretório.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


    /**
     * Deletes the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
```

```
    public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
        logger.info("Deleting policy for bucket: {}", bucketName);

        try {
            // Create a DeleteBucketPolicyRequest
            DeleteBucketPolicyRequest deletePolicyReq =
DeleteBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket policy
            s3Client.deleteBucketPolicy(deletePolicyReq);
            logger.info("Successfully deleted bucket policy");

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteBucketPolicy em Referência AWS SDK for Java 2.x da API.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um objeto em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;




    /**
     * Deletes an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be deleted
     */
    public static void deleteDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Deleting object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a DeleteObjectRequest
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Delete the object
            s3Client.deleteObject(deleteObjectRequest);
            logger.info("Object {} has been deleted", objectKey);

        } catch (S3Exception e) {
            logger.error("Failed to delete object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
```

```
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObject em Referência AWS SDK for Java 2.x da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua vários objetos em um bucket de diretório.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Deletes multiple objects from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKeys The list of keys (names) of the objects to be deleted
     */
    public static void deleteDirectoryBucketObjects(S3Client s3Client, String
 bucketName, List<String> objectKeys) {
        logger.info("Deleting objects from bucket: {}", bucketName);

        try {
            // Create a list of ObjectIdentifier.
            List<ObjectIdentifier> identifiers = objectKeys.stream()
                    .map(key -> ObjectIdentifier.builder().key(key).build())
                    .toList();

            Delete delete = Delete.builder()
                    .objects(identifiers)
                    .build();

            DeleteObjectsRequest deleteObjectsRequest =
 DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(delete)
                    .build();

            DeleteObjectsResponse deleteObjectsResponse =
 s3Client.deleteObjects(deleteObjectsRequest);
            deleteObjectsResponse.deleted().forEach(deleted -> logger.info("Deleted
 object: {}", deleted.key()));

        } catch (S3Exception e) {
            logger.error("Failed to delete objects: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK for Java 2.x da API.

`GetBucketEncryption`

O código de exemplo a seguir mostra como usar `GetBucketEncryption`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a configuração de criptografia de um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Retrieves the encryption configuration for an S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The type of server-side encryption applied to the bucket (e.g.,
     *         AES256, aws:kms)
     */
```

```
   public static String getDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
       try {
           // Create a GetBucketEncryptionRequest
           GetBucketEncryptionRequest getRequest =
GetBucketEncryptionRequest.builder()
                   .bucket(bucketName)
                   .build();

           // Retrieve the bucket encryption configuration
           GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
           ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

           String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
           logger.info("Bucket encryption algorithm: {}", encryptionType);
           logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
           logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

           return encryptionType;
       } catch (S3Exception e) {
           logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode(), e);
           throw e;
       }
   }
```

- Para obter detalhes da API, consulte GetBucketEncryption em Referência AWS SDK for Java 2.x da API.

GetBucketPolicy

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha a política de um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

    /**
     * Retrieves the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The bucket policy text
     */
    public static String getDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Getting policy for bucket: {}", bucketName);

        try {
            // Create a GetBucketPolicyRequest
            GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket policy
```

```
            GetBucketPolicyResponse response = s3Client.getBucketPolicy(policyReq);

            // Print and return the policy text
            String policyText = response.policy();
            logger.info("Bucket policy: {}", policyText);
            return policyText;

        } catch (S3Exception e) {
            logger.error("Failed to get bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte GetBucketPolicy em Referência AWS SDK for Java 2.x da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um objeto de um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be retrieved
     * @return The retrieved object as a ResponseInputStream
     */
    public static boolean getDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Retrieving object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .key(objectKey)
                    .bucket(bucketName)
                    .build();

            // Retrieve the object as bytes
            ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Print object contents to console
            String objectContent = new String(data, StandardCharsets.UTF_8);
            logger.info("Object contents: \n{}", objectContent);

            return true;

        } catch (S3Exception e) {
```

```
            logger.error("Failed to retrieve object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK for Java 2.x da API.

`GetObjectAttributes`

O código de exemplo a seguir mostra como usar `GetObjectAttributes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha atributos de um objeto de um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves attributes for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve attributes for
     * @return True if the object attributes are successfully retrieved, false
     *         otherwise
     */
    public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Retrieving attributes for object: {} from bucket: {}",
 objectKey, bucketName);

        try {
            // Create a GetObjectAttributesRequest
            GetObjectAttributesRequest getObjectAttributesRequest =
 GetObjectAttributesRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .objectAttributes(ObjectAttributes.E_TAG,
 ObjectAttributes.STORAGE_CLASS,
                            ObjectAttributes.OBJECT_SIZE)
                    .build();

            // Retrieve the object attributes
            GetObjectAttributesResponse response =
 s3Client.getObjectAttributes(getObjectAttributesRequest);
            logger.info("Attributes for object {}:", objectKey);
            logger.info("ETag: {}", response.eTag());
            logger.info("Storage Class: {}", response.storageClass());
            logger.info("Object Size: {}", response.objectSize());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object attributes: {} - Error code:
 {}",
                    e.awsErrorDetails().errorMessage(),
 e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Para obter detalhes da API, consulte GetObjectAttributes em Referência AWS SDK for Java 2.x da API.

`HeadBucket`

O código de exemplo a seguir mostra como usar `HeadBucket`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Verifica se o bucket de diretório do S3 especificado existe e está acessível.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Checks if the specified S3 directory bucket exists and is accessible.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to check
     * @return True if the bucket exists and is accessible, false otherwise
     */
    public static boolean headDirectoryBucket(S3Client s3Client, String bucketName)
 {
        logger.info("Checking if bucket exists: {}", bucketName);
```

```
        try {
            // Create a HeadBucketRequest
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            // If the bucket doesn't exist, the following statement throws
NoSuchBucketException,
            // which is a subclass of S3Exception.
            s3Client.headBucket(headBucketRequest);
            logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to access bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte HeadBucket em Referência AWS SDK for Java 2.x da API.

`HeadObject`

O código de exemplo a seguir mostra como usar `HeadObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha metadados de um objeto em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves metadata for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve metadata for
     * @return True if the object exists, false otherwise
     */
    public static boolean headDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Retrieving metadata for object: {} from bucket: {}", objectKey,
 bucketName);

        try {
            // Create a HeadObjectRequest
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Retrieve the object metadata
            HeadObjectResponse response = s3Client.headObject(headObjectRequest);
            logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with ETag:
 \"{}\"", objectKey, bucketName,
                    response.eTag());
            logger.info("Content-Type: {}", response.contentType());
            logger.info("Content-Length: {}", response.contentLength());
            logger.info("Last Modified: {}", response.lastModified());
            return true;
```

```
        } catch (S3Exception e) {
            logger.error("Failed to retrieve object metadata: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Para obter detalhes da API, consulte HeadObject em Referência AWS SDK for Java 2.x da API.

`ListDirectoryBuckets`

O código de exemplo a seguir mostra como usar `ListDirectoryBuckets`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste todos os buckets de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

     /**
     * Lists all S3 directory buckets and no general purpose buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listDirectoryBucketsRequest =
 ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
 s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte ListDirectoryBuckets em Referência AWS SDK for Java 2.x da API.

`ListMultipartUploads`

O código de exemplo a seguir mostra como usar `ListMultipartUploads`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os uploads fracionados em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * Lists multipart uploads for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of MultipartUpload objects representing the multipart uploads
     */
```

```
   public static List<MultipartUpload> listDirectoryBucketMultipartUploads(S3Client
s3Client, String bucketName) {
       logger.info("Listing in-progress multipart uploads for bucket: {}",
bucketName);

       try {
           // Create a ListMultipartUploadsRequest
           ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                   .bucket(bucketName)
                   .build();

           // List the multipart uploads
           ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
           List<MultipartUpload> uploads = response.uploads();
           for (MultipartUpload upload : uploads) {
               logger.info("In-progress multipart upload: Upload ID: {}, Key: {},
Initiated: {}", upload.uploadId(),
                       upload.key(), upload.initiated());
           }
           return uploads;

       } catch (S3Exception e) {
           logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode());
           return List.of(); // Return an empty list if an exception is thrown
       }
   }
```

- Para obter detalhes da API, consulte ListMultipartUploads em Referência AWS SDK for Java 2.x da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os objetos em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Lists objects in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of object keys in the bucket
     */
    public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
 String bucketName) {
        logger.info("Listing objects in bucket: {}", bucketName);
```

```
        try {
            // Create a ListObjectsV2Request
            ListObjectsV2Request listObjectsV2Request =
ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the list of objects
            ListObjectsV2Response response =
s3Client.listObjectsV2(listObjectsV2Request);

            // Extract and return the object keys
            return response.contents().stream()
                    .map(S3Object::key)
                    .collect(Collectors.toList());

        } catch (S3Exception e) {
            logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK for Java 2.x API.

`ListParts`

O código de exemplo a seguir mostra como usar `ListParts`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as partes de um upload fracionado em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Lists the parts of a multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object being uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @return A list of Part representing the parts of the multipart upload
     */
    public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
 s3Client, String bucketName,
            String objectKey, String uploadId) {
        logger.info("Listing parts for object: {} in bucket: {}", objectKey,
 bucketName);

        try {
            // Create a ListPartsRequest
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
```

```
                    .bucket(bucketName)
                    .uploadId(uploadId)
                    .key(objectKey)
                    .build();

            // List the parts of the multipart upload
            ListPartsResponse response = s3Client.listParts(listPartsRequest);
            List<Part> parts = response.parts();
            for (Part part : parts) {
                logger.info("Uploaded part: Part number = \"{}\", etag = {}",
part.partNumber(), part.eTag());
            }
            return parts;

        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- Para obter detalhes da API, consulte ListParts em Referência AWS SDK for Java 2.x da API.

`PutBucketEncryption`

O código de exemplo a seguir mostra como usar `PutBucketEncryption`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a criptografia em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

    /**
     * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param kmsKeyId   The ID of the customer-managed KMS key
     */
    public static void putDirectoryBucketEncryption(S3Client s3Client, String
 bucketName, String kmsKeyId) {
        // Define the default encryption configuration to use SSE-KMS. For directory
        // buckets, AWS managed KMS keys aren't supported. Only customer-managed
 keys
        // are supported.
        ServerSideEncryptionByDefault encryptionByDefault =
 ServerSideEncryptionByDefault.builder()
                .sseAlgorithm(ServerSideEncryption.AWS_KMS)
                .kmsMasterKeyID(kmsKeyId)
                .build();

        // Create a server-side encryption rule to apply the default encryption
        // configuration. For directory buckets, the bucketKeyEnabled field is
 enforced
        // to be true.
        ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .applyServerSideEncryptionByDefault(encryptionByDefault)
                .build();
```

```
        // Create the server-side encryption configuration for the bucket
        ServerSideEncryptionConfiguration encryptionConfiguration =
ServerSideEncryptionConfiguration.builder()
                .rules(rule)
                .build();

        // Create the PutBucketEncryption request
        PutBucketEncryptionRequest putRequest = PutBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .serverSideEncryptionConfiguration(encryptionConfiguration)
                .build();

        // Set the bucket encryption
        try {
            s3Client.putBucketEncryption(putRequest);
            logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte PutBucketEncryption em Referência AWS SDK for Java 2.x da API.

`PutBucketPolicy`

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Aplique uma política a um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

    /**
     * Sets the following bucket policy for the specified S3 directory bucket.
     *<pre>
     * {
     *     "Version":"2012-10-17",
     *     "Statement": [
     *         {
     *             "Sid": "AdminPolicy",
     *             "Effect": "Allow",
     *             "Principal": {
     *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
     *             },
     *             "Action": "s3express:*",
     *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
     *         }
     *     ]
     * }
     * </pre>
     * This policy grants all S3 directory bucket actions to identities in the same
 account as the bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param policyText The policy text to be applied
     */
    public static void putDirectoryBucketPolicy(S3Client s3Client, String
 bucketName, String policyText) {
        logger.info("Setting policy on bucket: {}", bucketName);
        logger.info("Policy: {}", policyText);
```

```
        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3Client.putBucketPolicy(policyReq);
            logger.info("Bucket policy set successfully!");

        } catch (S3Exception e) {
            logger.error("Failed to set bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte PutBucketPolicy em Referência AWS SDK for Java 2.x da API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Coloque um objeto em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * Puts an object into the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be placed in the bucket
     * @param filePath   The path of the file to be uploaded
     */
    public static void putDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey, Path filePath) {
        logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

        try {
            // Create a PutObjectRequest
            PutObjectRequest putObj = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Upload the object
            s3Client.putObject(putObj, filePath);
            logger.info("Successfully placed {} into bucket {}", objectKey,
 bucketName);

        } catch (UncheckedIOException e) {
            throw S3Exception.builder().message("Failed to read the file: " +
 e.getMessage()).cause(e)
                    .awsErrorDetails(AwsErrorDetails.builder()
                            .errorCode("ClientSideException:FailedToReadFile")
                            .errorMessage(e.getMessage())
                            .build())
```

```
                    .build();
        } catch (S3Exception e) {
            logger.error("Failed to put object: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK for Java 2.x da API.

`UploadPart`

O código de exemplo a seguir mostra como usar `UploadPart`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de uma parte de um upload fracionado em um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * This method creates part requests and uploads individual parts to S3.
     * While it uses the UploadPart API to upload a single part, it does so
     * sequentially to handle multiple parts of a file, returning all the completed
     * parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @param filePath   The path to the file to be uploaded
     * @return A list of uploaded parts
     * @throws IOException if an I/O error occurs
     */
    public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
 s3Client, String bucketName,
            String objectKey, String uploadId, Path filePath) throws IOException {
        logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
 bucketName);

        int partNumber = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // Read the local file, break down into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(), "r")) {
            long fileSize = file.length();
            int position = 0;

            // Sequentially upload parts of the file
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);
```

```
                bb.flip(); // Swap position and limit before reading from the buffer
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(objectKey)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                // Build the uploaded part
                CompletedPart uploadedPart = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();

                // Add the uploaded part to the list
                uploadedParts.add(uploadedPart);

                // Log to indicate the part upload is done
                logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
        return uploadedParts;
    }
```

- Para obter detalhes da API, consulte UploadPart em Referência AWS SDK for Java 2.x da API.

UploadPartCopy

O código de exemplo a seguir mostra como usar `UploadPartCopy`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie partes de cópia com base no tamanho do objeto de origem e copie partes individuais para um bucket de diretório.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
 com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Creates copy parts based on source object size and copies over individual
     * parts.
     *
     * @param s3Client          The S3 client used to interact with S3
     * @param sourceBucket      The name of the source bucket
     * @param sourceKey         The key (name) of the source object
     * @param destinationBucket The name of the destination bucket
     * @param destinationKey    The key (name) of the destination object
     * @param uploadId          The upload ID used to track the multipart upload
     * @return A list of completed parts
     */
    public static List<CompletedPart> multipartUploadCopyForDirectoryBucket(S3Client
 s3Client, String sourceBucket,
            String sourceKey, String destinationBucket, String destinationKey,
 String uploadId) {
        // Get the object size to track the end of the copy operation
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(sourceBucket)
                .key(sourceKey)
                .build();
        HeadObjectResponse headObjectResponse =
 s3Client.headObject(headObjectRequest);
        long objectSize = headObjectResponse.contentLength();

        logger.info("Source Object size: {}", objectSize);

        // Copy the object using 20 MB parts
        long partSize = 20 * 1024 * 1024; // 20 MB
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);
            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
 partNum, bytePosition, lastByte);

            try {
```

```
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                        .sourceBucket(sourceBucket)
                        .sourceKey(sourceKey)
                        .destinationBucket(destinationBucket)
                        .destinationKey(destinationKey)
                        .uploadId(uploadId)
                        .copySourceRange("bytes=" + bytePosition + "-" + lastByte)
                        .partNumber(partNum)
                        .build();
                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNum)
                        .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                        .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code: {}",
partNum,
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
                throw e;
            }
        }

        return uploadedParts;
    }
```

- Para obter detalhes da API, consulte UploadPartCopy em Referência AWS SDK for Java 2.x da API.

Cenários

Criar um URL pré-assinado para obter um objeto

O exemplo de código a seguir mostra como criar um URL pré-assinado para buckets de diretório do S3 e obter um objeto.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere um URL GET pré-assinado para acessar um objeto em um bucket de diretório do S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Generates a presigned URL for accessing an object in the specified S3
```

```
     * directory bucket.
     *
     * @param s3Presigner The S3 presigner client used to generate the presigned URL
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to access
     * @return A presigned URL for accessing the specified object
     */
    public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
s3Presigner, String bucketName,
            String objectKey) {
        logger.info("Generating presigned URL for object: {} in bucket: {}",
objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Create a GetObjectPresignRequest
            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
                    .getObjectRequest(getObjectRequest)
                    .build();

            // Generate the presigned URL
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK for Java 2.x da API.

SageMaker Exemplos de IA usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com SageMaker IA.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá SageMaker AI

O exemplo de código a seguir mostra como começar a usar a SageMaker IA.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
 sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
 notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
 item.notebookInstanceName());
            }

        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListNotebookInstances em Referência AWS SDK for Java 2.x da API.

Ações

`CreatePipeline`

O código de exemplo a seguir mostra como usar `CreatePipeline`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Create a pipeline from the example pipeline JSON.
   public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
           String functionArn, String pipelineName) {
       System.out.println("Setting up the pipeline.");
       JSONParser parser = new JSONParser();

       // Read JSON and get pipeline definition.
       try (FileReader reader = new FileReader(filePath)) {
           Object obj = parser.parse(reader);
           JSONObject jsonObject = (JSONObject) obj;
           JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
           for (Object stepObj : stepsArray) {
               JSONObject step = (JSONObject) stepObj;
               if (step.containsKey("FunctionArn")) {
                   step.put("FunctionArn", functionArn);
               }
           }
           System.out.println(jsonObject);

           // Create the pipeline.
           CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
                   .pipelineDescription("Java SDK example pipeline")
                   .roleArn(roleArn)
                   .pipelineName(pipelineName)
                   .pipelineDefinition(jsonObject.toString())
                   .build();
```

```
            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte CreatePipeline em Referência AWS SDK for Java 2.x da API.

`DeletePipeline`

O código de exemplo a seguir mostra como usar `DeletePipeline`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }
```

- Para obter detalhes da API, consulte DeletePipeline em Referência AWS SDK for Java 2.x da API.

`DescribePipelineExecution`

O código de exemplo a seguir mostra como usar `DescribePipelineExecution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Check the status of a pipeline execution.
   public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
           throws InterruptedException {
       String status;
       int index = 0;
       do {
           DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                   .pipelineExecutionArn(executionArn)
                   .build();

           DescribePipelineExecutionResponse response = sageMakerClient
                   .describePipelineExecution(pipelineExecutionRequest);
           status = response.pipelineExecutionStatusAsString();
           System.out.println(index + ". The Status of the pipeline is " + status);
           TimeUnit.SECONDS.sleep(4);
           index++;
       } while ("Executing".equals(status));
       System.out.println("Pipeline finished with status " + status);
   }
```

- Para obter detalhes da API, consulte DescribePipelineExecution em Referência AWS SDK for Java 2.x da API.

`StartPipelineExecution`

O código de exemplo a seguir mostra como usar `StartPipelineExecution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";
```

```
        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();

        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();

        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();
```

```
        System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);

        StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }
```

- Para obter detalhes da API, consulte StartPipelineExecution em Referência AWS SDK for Java 2.x da API.

Cenários

Conceitos básicos de trabalhos geoespaciais e pipelines

O exemplo de código a seguir mostra como:

- Configurar recursos para um pipeline.
- Configurar um pipeline que executa um trabalho geoespacial.
- Iniciar a execução de um pipeline.
- Monitorar o status da execução.
- Ver a saída do pipeline.
- Limpar recursos.

Para obter mais informações, consulte Criar e executar SageMaker pipelines usando AWS SDKs em. Community.aws

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
 <functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
 <pipelineName>\n\n"
                +
                "Where:\n" +
                "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
 +
                "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
                "    functionFileLocation - The file location where the JAR file
 that represents the AWS Lambda function is located.\n\n"
                +
                "    functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).\n\n" +
                "    queueName - The name of the Amazon Simple Queue Service (Amazon
 SQS) queue.\n\n" +
                "    bucketName - The name of the Amazon Simple Storage Service
 (Amazon S3) bucket.\n\n" +
                "    lnglatData - The file location of the latlongtest.csv file
 required for this use case.\n\n" +
                "    spatialPipelinePath - The file location of the
 GeoSpatialPipeline.json file required for this use case.\n\n"
                +
                "    pipelineName - The name of the pipeline to create (for example,
 sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String sageMakerRoleName = args[0];
        String lambdaRoleName = args[1];
        String functionFileLocation = args[2];
        String functionName = args[3];
        String queueName = args[4];
        String bucketName = args[5];
        String lnglatData = args[6];
        String spatialPipelinePath = args[7];
        String pipelineName = args[8];
        String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        LambdaClient lambdaClient = LambdaClient.builder()
                .region(region)
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
        System.out.println(
                "\nThis example workflow will guide you through setting up and
running an" +
                        "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
```

```
                        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
                        "\nreverse geocode addresses in an input file and store the
results in an export file.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
       String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
       String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

       String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
               handlerName);
       String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
       System.out.println("The queue URL is " + queueUrl);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Setting up bucket " + bucketName);
       if (!checkBucket(s3Client, bucketName)) {
           setupBucket(s3Client, bucketName);
           System.out.println("Put " + lnglatData + " into " + bucketName);
           putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
       }
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Now we can create and run our pipeline.");
       setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
       String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
               pipelineName);
       System.out.println("The pipeline execution ARN value is " +
pipelineExecutionARN);
       waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
       System.out.println("Getting output results " + bucketName);
       getOutputResults(s3Client, bucketName);
       System.out.println(DASHES);

       System.out.println(DASHES);
```

```
        System.out.println("The pipeline has completed. To view the pipeline and
 runs " +
                "in SageMaker Studio, follow these instructions:" +
                "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
 Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }

    private static void readObject(S3Client s3Client, String bucketName, String key)
 {
        System.out.println("Output file contents: \n");
        GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
        byte[] byteArray = objectBytes.asByteArray();
```

```
        String text = new String(byteArray, StandardCharsets.UTF_8);
        System.out.println("Text output: " + text);
    }

    // Display some results from the output directory.
    public static void getOutputResults(S3Client s3Client, String bucketName) {
        System.out.println("Getting output results {bucketName}.");
        ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
                .bucket(bucketName)
                .prefix("outputfiles/")
                .build();

        ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
        List<S3Object> s3Objects = response.contents();
        for (S3Object object : s3Objects) {
            readObject(s3Client, bucketName, object.key());
        }
    }

    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
            throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                    .pipelineExecutionArn(executionArn)
                    .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " + status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }

    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
```

```
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }

    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
```

```
    }

    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";

        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();
```

```
        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();

        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();

        System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);
```

```
        StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
                .uuid(eventSourceMapping)
                .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        try {
            for (String policy : lambdaRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific AWS Lambda function.
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("*** " + functionName + " was deleted");
```

```
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Delete the specific S3 bucket.
    public static void deleteBucket(S3Client s3Client, String bucketName) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("*** " + bucketName + " was deleted.");
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                deleteBucketObjects(s3, bucketName, myValue.key());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
                .key(objectName)
                .build());
        try {
            DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
```

```
                    .bucket(bucketName)
                    .delete(Delete.builder()
                            .objects(toDelete).build())
                    .build();

            s3.deleteObjects(dor);
            System.out.println("*** " + bucketName + " objects were deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific Amazon SQS queue.
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
 String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key("samplefiles/" + objectKey)
                    .metadata(metadata)
                    .build();
```

```
            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void setupBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Set up the SQS queue to use with the pipeline.
    public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
            String lambdaName) {
        System.out.println("Setting up queue named " + queueName);
        try {
            Map<QueueAttributeName, String> queueAtt = new HashMap<>();
            queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
            queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
```

```
            queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(queueAtt)
                    .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            TimeUnit.SECONDS.sleep(15);

            connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
            System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    // Connect the queue to the Lambda function as an event source.
    public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
            String lambdaName) {
        System.out.println("Connecting the Lambda function and queue for the
pipeline.");
        String queueArn = "";

        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);
        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();
```

```
        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
            System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
            queueArn = queueAtt.getValue();
        }

        CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
                .eventSourceArn(queueArn)
                .functionName(lambdaName)
                .build();

        CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
        eventSourceMapping = response1.uuid();
        System.out.println("The mapping between the event source and Lambda function
was successful");
    }

    // Create an AWS Lambda function.
    public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
            String handler) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .functionName(functionName)
                    .description("SageMaker example function.")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA11)
                    .timeout(200)
                    .memorySize(1024)
                    .role(role)
```

```
                    .build();

            // Create a Lambda function using a waiter.
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The function ARN is " +
functionResponse.functionArn());
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSageMakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        System.out.println("Creating a role to use with SageMaker.");
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
```

```
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : sageMakerRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    private static String createLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
```

```
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : lambdaRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());

        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
            String handler) {
        System.out.println("Create an AWS Lambda function used in this workflow.");
        String functionArn;
```

```
        try {
            // Does this function already exist.
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
            functionArn = response.configuration().functionArn();

        } catch (LambdaException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
        }
        return functionArn;
    }

    // Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
    // method returns true.
    public static boolean checkBucket(S3Client s3, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3.headBucket(headBucketRequest);
            System.out.println(bucketName + " exists");
            return true;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Checks to see if the Amazon SQS queue exists. If not, this method creates a
    // new queue
    // and returns the ARN value.
    public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
            String lambdaName) {
        System.out.println("Creating a queue for this use case.");
        String queueUrl;
```

```
        try {
            GetQueueUrlRequest request = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
            queueUrl = response.queueUrl();
            System.out.println(queueUrl);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
        }
        return queueUrl;
    }

    // Checks to see if the Lambda role exists. If not, this method creates it.
    public static String checkLambdaRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS Lambda to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createLambdaRole(iam, roleName);
        }
        return roleArn;
    }

    // Checks to see if the SageMaker role exists. If not, this method creates it.
    public static String checkSageMakerRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS SageMaker to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();
```

```
            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createSageMakerRole(iam, roleName);
        }
        return roleArn;
    }

    private static String[] getSageMakerRolePolicies() {
        String[] sageMakerRolePolicies = new String[3];
        sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
        sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
 "AmazonSageMakerGeospatialFullAccess";
        sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        return sageMakerRolePolicies;
    }

    private static String[] getLambdaRolePolicies() {
        String[] lambdaRolePolicies = new String[5];
        lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
        lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess";
        lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
                + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
        lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole";
        return lambdaRolePolicies;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution

- StartPipelineExecution
- UpdatePipeline

Exemplos do Secrets Manager usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetSecretValue`

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-
secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <secretName>\s

                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
                .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }

    public static void getValue(SecretsManagerClient secretsClient, String
 secretName) {
```

```
        try {
            GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                    .secretId(secretName)
                    .build();

            GetSecretValueResponse valueResponse =
 secretsClient.getSecretValue(valueRequest);
            String secret = valueResponse.secretString();
            System.out.println(secret);

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetSecretValue em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon SES usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`ListIdentities`

O código de exemplo a seguir mostra como usar `ListIdentities`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        listSESIdentities(client);
    }

    public static void listSESIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
```

```
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListIdentities em Referência AWS SDK for Java 2.x da API.

`ListTemplates`

O código de exemplo a seguir mostra como usar `ListTemplates`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();
```

```
        listAllTemplates(sesv2Client);
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
 ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();

            ListEmailTemplatesResponse response =
 sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                    .forEach(template -> System.out.println("Template name: " +
 template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListTemplates em Referência AWS SDK for Java 2.x da API.

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.\s
                    subject - The  subject line.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
                .region(region)
                .build();
```

```
        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        try {
            send(client, sender, recipient, subject, bodyHTML);
            client.close();
            System.out.println("Done");

        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void send(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyHTML) throws MessagingException {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
```

```
                .message(msg)
                .source(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject> <fileLocation>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.\s
                    subject - The  subject line.\s
                    fileLocation - The location of a Microsoft Excel file to use as
 an attachment (C:/AWS/customers.xls).\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
                + "of customers to contact.";

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p>Please see the attached file for a " + "list of customers to
 contact.</p>" + "</body>"
                + "</html>";

        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();
```

```
        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
            client.close();
            System.out.println("Done");

        } catch (IOException | MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void sendemailAttachment(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyText,
            String bodyHTML,
            String fileLocation) throws AddressException, MessagingException,
IOException {

        java.io.File theFile = new java.io.File(fileLocation);
        byte[] fileContent = Files.readAllBytes(theFile.toPath());

        Session session = Session.getDefaultInstance(new Properties());

        // Create a new MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Add subject, from and to lines.
        message.setSubject(subject, "UTF-8");
        message.setFrom(new InternetAddress(sender));
        message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

        // Create a multipart/alternative child container.
        MimeMultipart msgBody = new MimeMultipart("alternative");

        // Create a wrapper for the HTML and text parts.
        MimeBodyPart wrap = new MimeBodyPart();

        // Define the text part.
        MimeBodyPart textPart = new MimeBodyPart();
        textPart.setContent(bodyText, "text/plain; charset=UTF-8");
```

```
        // Define the HTML part.
        MimeBodyPart htmlPart = new MimeBodyPart();
        htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

        // Add the text and HTML parts to the child container.
        msgBody.addBodyPart(textPart);
        msgBody.addBodyPart(htmlPart);

        // Add the child container to the wrapper object.
        wrap.setContent(msgBody);

        // Create a multipart/mixed parent container.
        MimeMultipart msg = new MimeMultipart("mixed");

        // Add the parent container to the message.
        message.setContent(msg);
        msg.addBodyPart(wrap);

        // Define the attachment.
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new ByteArrayDataSource(fileContent,
                "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
        att.setDataHandler(new DataHandler(fds));

        String reportName = "WorkReport.xls";
        att.setFileName(reportName);

        // Add the attachment to the message.
        msg.addBodyPart(att);

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);

            ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

            byte[] arr = new byte[buf.remaining()];
            buf.get(arr);

            SdkBytes data = SdkBytes.fromByteArray(arr);
```

```
            RawMessage rawMessage = RawMessage.builder()
                    .data(data)
                    .build();

            SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
                    .rawMessage(rawMessage)
                    .build();

            client.sendRawEmail(rawEmailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Email sent using SesClient with attachment");
    }
}
```

- Para obter detalhes da API, consulte SendEmail em Referência AWS SDK for Java 2.x da API.

`SendTemplatedEmail`

O código de exemplo a seguir mostra como usar `SendTemplatedEmail`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;
```

```
/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <template> <sender> <recipient>\s

                Where:
                    template - The name of the email template.
                    sender - An email address that represents the sender.\s
                    recipient - An email address that represents the recipient.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String templateName = args[0];
        String sender = args[1];
        String recipient = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        send(sesv2Client, sender, recipient, templateName);
    }
```

```
    public static void send(SesV2Client client, String sender, String recipient,
 String templateName) {
        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        /*
         * Specify both name and favorite animal (favoriteanimal) in your code when
         * defining the Template object.
         * If you don't specify all the variables in the template, Amazon SES
 doesn't
         * send the email.
         */
        Template myTemplate = Template.builder()
                .templateName(templateName)
                .templateData("{\n" +
                        "  \"name\": \"Jason\"\n," +
                        "  \"favoriteanimal\": \"Cat\"\n" +
                        "}")
                .build();

        EmailContent emailContent = EmailContent.builder()
                .template(myTemplate)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email based on a template
 using the AWS SDK for Java (v2)...");
            client.sendEmail(emailRequest);
            System.out.println("email based on a template was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SendTemplatedEmail em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Java 2.x

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Criar uma aplicação Web para rastrear dados do Amazon Redshift

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho usando um banco de dados do Amazon Redshift.

SDK para Java 2.x

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- banco de dados de origem
- Amazon SES

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Java 2.x

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Para obter o código-fonte completo e instruções sobre como configurar e executar um exemplo que usa a API JDBC, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Detectar EPI em imagens

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

SDK para Java 2.x

Mostra como criar uma AWS Lambda função que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service

(Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Usar Step Functions para invocar funções do Lambda

O exemplo de código a seguir mostra como criar uma máquina de AWS Step Functions estados que invoca AWS Lambda funções em sequência.

SDK para Java 2.x

Mostra como criar um fluxo de trabalho AWS sem servidor usando AWS Step Functions e o. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Funções de etapa

Exemplos do API v2 do Amazon SES usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API v2 AWS SDK for Java 2.x com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateContact`

O código de exemplo a seguir mostra como usar `CreateContact`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
try {
  // Create a new contact with the provided email address in the
  CreateContactRequest contactRequest = CreateContactRequest.builder()
      .contactListName(CONTACT_LIST_NAME)
      .emailAddress(emailAddress)
      .build();

  sesClient.createContact(contactRequest);
  contacts.add(emailAddress);

  System.out.println("Contact created: " + emailAddress);
```

```
        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }
```

- Para obter detalhes da API, consulte CreateContact em Referência AWS SDK for Java 2.x da API.

`CreateContactList`

O código de exemplo a seguir mostra como usar `CreateContactList`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
 CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }
```

- Para obter detalhes da API, consulte CreateContactList em Referência AWS SDK for Java 2.x da API.

`CreateEmailIdentity`

O código de exemplo a seguir mostra como usar `CreateEmailIdentity`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   try {
     CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
         .emailIdentity(verifiedEmail)
         .build();
     sesClient.createEmailIdentity(createEmailIdentityRequest);
     System.out.println("Email identity created: " + verifiedEmail);
   } catch (AlreadyExistsException e) {
     System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
   } catch (NotFoundException e) {
     System.err.println("The provided email address is not verified: " +
verifiedEmail);
     throw e;
   } catch (LimitExceededException e) {
     System.err
         .println("You have reached the limit for email identities. Please remove
some identities and try again.");
     throw e;
   } catch (SesV2Exception e) {
     System.err.println("Error creating email identity: " + e.getMessage());
     throw e;
   }
```

- Para obter detalhes da API, consulte CreateEmailIdentity em Referência AWS SDK for Java 2.x da API.

`CreateEmailTemplate`

O código de exemplo a seguir mostra como usar `CreateEmailTemplate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
              .html(newsletterHtml)
              .text(newsletterText)
              .build())
          .build();

      sesClient.createEmailTemplate(templateRequest);

      System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
      // If the template already exists, skip this step and proceed with the next
      // operation
      System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
      // If the limit for email templates is exceeded, fail the workflow and inform
      // the user
      System.err.println("You have reached the limit for email templates. Please
 remove some templates and try again.");
      throw e;
    } catch (Exception e) {
      System.err.println("Error occurred while creating email template: " +
 e.getMessage());
      throw e;
```

```
    }
```

- Para obter detalhes da API, consulte CreateEmailTemplate em Referência AWS SDK for Java 2.x da API.

`DeleteContactList`

O código de exemplo a seguir mostra como usar `DeleteContactList`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // Delete the contact list
      DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      sesClient.deleteContactList(deleteContactListRequest);

      System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
    } catch (NotFoundException e) {
      // If the contact list does not exist, log the error and proceed
      System.out.println("Contact list not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }
```

- Para obter detalhes da API, consulte DeleteContactList em Referência AWS SDK for Java 2.x da API.

`DeleteEmailIdentity`

O código de exemplo a seguir mostra como usar `DeleteEmailIdentity`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // Delete the email identity
      DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
          .emailIdentity(this.verifiedEmail)
          .build();

      sesClient.deleteEmailIdentity(deleteIdentityRequest);

      System.out.println("Email identity deleted: " + this.verifiedEmail);
    } catch (NotFoundException e) {
      // If the email identity does not exist, log the error and proceed
      System.out.println("Email identity not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
      e.printStackTrace();
    }
  } else {
    System.out.println("Skipping email identity deletion.");
  }
```

- Para obter detalhes da API, consulte DeleteEmailIdentity em Referência AWS SDK for Java 2.x da API.

`DeleteEmailTemplate`

O código de exemplo a seguir mostra como usar `DeleteEmailTemplate`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
      e.printStackTrace();
    }
```

- Para obter detalhes da API, consulte DeleteEmailTemplate em Referência AWS SDK for Java 2.x da API.

`ListContacts`

O código de exemplo a seguir mostra como usar `ListContacts`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    ListContactsRequest contactListRequest = ListContactsRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .build();

    List<String> contactEmails;
    try {
      ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);

      contactEmails = contactListResponse.contacts().stream()
          .map(Contact::emailAddress)
          .toList();
    } catch (Exception e) {
      // TODO: Remove when listContacts's GET body issue is resolved.
      contactEmails = this.contacts;
    }
```

- Para obter detalhes da API, consulte ListContacts em Referência AWS SDK for Java 2.x da API.

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = """

                                Usage:
                                    <sender> <recipient> <subject>\s

                                Where:
                                    sender - An email address that represents the
 sender.\s
                                    recipient - An email address that represents the
 recipient.\s
                                    subject - The subject line.\s
                                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
```

```
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        send(sesv2Client, sender, recipient, subject, bodyHTML);
    }

    public static void send(SesV2Client client,
                            String sender,
                            String recipient,
                            String subject,
                            String bodyHTML) {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        EmailContent emailContent = EmailContent.builder()
                .simple(msg)
                .build();
```

```
        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES "
                    + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Envia uma mensagem usando um modelo.

```
      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
```

```
    }
```

Envia uma mensagem com informações do cabeçalho.

```
public class SendwithHeader {

    public static void main(String[] args) {
        final String usage = """

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        String bodyHTML = """
                <html>
                    <head></head>
                    <body>
                        <h1>Hello!</h1>
                        <p>See the list of customers.</p>
                    </body>
                </html>
                """;

        sendWithHeader(sesv2Client, sender, recipient, subject, bodyHTML);
```

```
        sesv2Client.close();
    }

    /**
     * Sends an email using the AWS SES V2 client.
     *
     * @param sesv2Client the SES V2 client to use for sending the email
     * @param sender the email address of the sender
     * @param recipient the email address of the recipient
     * @param subject the subject of the email
     * @param bodyHTML the HTML content of the email body
     */
    public static void sendWithHeader(SesV2Client sesv2Client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyHTML) {
        EmailContent emailContent = EmailContent.builder()
                .simple(Message.builder()
                        .body(b -> b.html(c ->
 c.charset(UTF_8.name()).data(bodyHTML))
                                .text(c -> c.charset(UTF_8.name()).data(bodyHTML)))
                        .subject(c -> c.charset(UTF_8.name()).data(subject))
                        .headers(List.of(
                                MessageHeader.builder()
                                        .name("List-Unsubscribe")
                                        .value("<https://nutrition.co/?
address=x&topic=x>, <mailto:unsubscribe@nutrition.co?subject=TopicUnsubscribe>")
                                        .build(),
                                MessageHeader.builder()
                                        .name("List-Unsubscribe-Post")
                                        .value("List-Unsubscribe=One-Click")
                                        .build()))
                        .build())
                .build();

        SendEmailRequest request = SendEmailRequest.builder()
                .fromEmailAddress(sender)
                .destination(d -> d.toAddresses(recipient))
                .content(emailContent)
                .build();

        try {
            SendEmailResponse response = sesv2Client.sendEmail(request);
```

```
            System.out.println("Email sent! Message ID: " + response.messageId());
        } catch (SesV2Exception e) {
            System.err.println("Failed to send email: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte SendEmail em Referência AWS SDK for Java 2.x da API.

Cenários

Cenário de boletim informativo

O exemplo de código a seguir mostra como executar o cenário do boletim informativo da API v2 do Amazon SES.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
 CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
```

```
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }

      try {
        // Create a new contact with the provided email address in the
        CreateContactRequest contactRequest = CreateContactRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .emailAddress(emailAddress)
            .build();

        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);

        System.out.println("Contact created: " + emailAddress);

        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
```

```
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }

      ListContactsRequest contactListRequest = ListContactsRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      List<String> contactEmails;
      try {
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }


      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
```

```
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
      }

    try {
      CreateEmailIdentityRequest createEmailIdentityRequest =
 CreateEmailIdentityRequest.builder()
          .emailIdentity(verifiedEmail)
          .build();
      sesClient.createEmailIdentity(createEmailIdentityRequest);
      System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
      System.out.println("Email identity already exists, skipping creation: " +
 verifiedEmail);
    } catch (NotFoundException e) {
      System.err.println("The provided email address is not verified: " +
 verifiedEmail);
      throw e;
    } catch (LimitExceededException e) {
      System.err
          .println("You have reached the limit for email identities. Please remove
 some identities and try again.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating email identity: " + e.getMessage());
      throw e;
    }

    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
```

```
            .html(newsletterHtml)
            .text(newsletterText)
            .build())
        .build();

    sesClient.createEmailTemplate(templateRequest);

    System.out.println("Email template created: " + TEMPLATE_NAME);
  } catch (AlreadyExistsException e) {
    // If the template already exists, skip this step and proceed with the next
    // operation
    System.out.println("Email template already exists, skipping creation...");
  } catch (LimitExceededException e) {
    // If the limit for email templates is exceeded, fail the workflow and inform
    // the user
    System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
    throw e;
  } catch (Exception e) {
    System.err.println("Error occurred while creating email template: " +
e.getMessage());
    throw e;
  }

  try {
    // Delete the contact list
    DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .build();

    sesClient.deleteContactList(deleteContactListRequest);

    System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
  } catch (NotFoundException e) {
    // If the contact list does not exist, log the error and proceed
    System.out.println("Contact list not found. Skipping deletion...");
  } catch (Exception e) {
    System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
    e.printStackTrace();
  }

    try {
```

```
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }

    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
      e.printStackTrace();
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .

- CreateContact
- CreateContactList
- CreateEmailIdentity
- CreateEmailTemplate
- DeleteContactList
- DeleteEmailIdentity
- DeleteEmailTemplate
- ListContacts
- SendEmail.simple
- SendEmail.template

Exemplos do Amazon SNS usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon SNS

O exemplo de código a seguir mostra como começar a usar o Amazon SNS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK for Java 2.x da API.

Ações

`CheckIfPhoneNumberIsOptedOut`

O código de exemplo a seguir mostra como usar `CheckIfPhoneNumberIsOptedOut`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>
```

```
                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOut em Referência AWS SDK for Java 2.x da API.

`ConfirmSubscription`

O código de exemplo a seguir mostra como usar `ConfirmSubscription`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
                    .topicArn(topicArn)
                    .build();

            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ConfirmSubscription em Referência AWS SDK for Java 2.x da API.

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte CreateTopic em Referência AWS SDK for Java 2.x da API.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
```

```
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteTopic em Referência AWS SDK for Java 2.x da API.

`GetSMSAttributes`

O código de exemplo a seguir mostra como usar `GetSMSAttributes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
```

```
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- Consulte detalhes da API em GetSMSAttributes na Referência da API AWS SDK for Java 2.x .

`GetTopicAttributes`

O código de exemplo a seguir mostra como usar `GetTopicAttributes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Getting attributes for a topic with name: " + topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
 snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
 result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetTopicAttributes em Referência AWS SDK for Java 2.x da API.

`ListPhoneNumbersOptedOut`

O código de exemplo a seguir mostra como usar `ListPhoneNumbersOptedOut`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }
```

```
    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " + result.sdkHttpResponse().statusCode()
 + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListPhoneNumbersOptedOut em Referência AWS SDK for Java 2.x da API.

`ListSubscriptions`

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListSubscriptions em Referência AWS SDK for Java 2.x da API.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK for Java 2.x da API.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>
```

```
                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Java 2.x .

`SetSMSAttributes`

O código de exemplo a seguir mostra como usar `SetSMSAttributes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
 attributes) {
        try {
```

```
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
 was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para receber detalhes da API, consulte SetSMSAttributes na Referência da API AWS SDK for Java 2.x .

`SetSubscriptionAttributes`

O código de exemplo a seguir mostra como usar `SetSubscriptionAttributes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
```

```
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SetSubscriptionAttributes em Referência AWS SDK for Java 2.x da API.

`SetTopicAttributes`

O código de exemplo a seguir mostra como usar `SetTopicAttributes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
```

```
            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() + "\n
\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte SetTopicAttributes em Referência AWS SDK for Java 2.x da API.

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
```

```
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
```

```
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Inscrever um endpoint HTTP em um tópico.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive notifications.
                """;

        if (args.length < 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
```

```
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK for Java 2.x .

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);
```

```
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte TagResource em Referência AWS SDK for Java 2.x da API.

Cancelar assinatura

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK for Java 2.x .

Cenários

Criar uma aplicação para enviar dados para uma tabela do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação que envia dados para uma tabela do Amazon DynamoDB e notifica você quando um usuário atualiza a tabela.

SDK para Java 2.x

Mostra como criar uma aplicação Web dinâmica que envia dados usando a API Java do Amazon DynamoDB e envia uma mensagem de texto usando a API Java do Amazon Simple Notification Service.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SNS

Criação de uma aplicação do Amazon SNS

O exemplo de código a seguir mostra como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo que usa a API Java Async, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Criar um endpoint de plataforma para notificações por push

O exemplo de código a seguir mostra como criar um endpoint de plataforma para notificações por push do Amazon SNS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
```

```
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
               token - The device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               platformApplicationArn - The ARN value of platform application. You
 can get this value from the AWS Management Console.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();
```

```
            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar e publicar em um tópico FIFO

O exemplo de código a seguir mostra como criar e publicar em um tópico FIFO do Amazon SNS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O teste verifica o recebimento da mensagem em cada fila. O exemplo completo também mostra a adição de políticas de acesso e exclui os recursos no final.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
            "Where:\n" +
            "   fifoTopicName - The name of the FIFO topic that you want to create.
 \n\n" +
            "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
            +
            "   retailQueueARN - The name of a SQS FIFO queue that will created for
 the retail consumer. \n\n" +
            "   analyticsQueueARN - The name of a SQS standard queue that will be
 created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
```

```
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
```

```
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";
```

```
            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTopic
  - Publicar
  - Assinar

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Publicar mensagens SMS em um tópico

O exemplo de código a seguir mostra como:

- Criar um tópico do Amazon SNS.
- Inscrever números de telefone no tópico.
- Publicar mensagens SMS no tópico para que todos os números de telefone inscritos recebam a mensagem de uma só vez.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar um tópico e retorne seu ARN.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
```

```
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

Inscreva um endpoint em um tópico.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives notifications
 (for example, +1XXX5550100).
                """;
```

```
        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Defina atributos na mensagem, como o ID do remetente, o preço máximo e seu tipo. Os atributos de mensagem são opcionais.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
 attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
 was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Publique uma mensagem em um tópico. A mensagem é enviada para todos os assinantes.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is sent
 (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }
```

```
    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is sent
 (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
```

```
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Java 2.x .

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
```

```
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
```

```
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
```

```
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and the
 subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                +
                "        within the five-minute deduplication interval, is accepted
 but not delivered.\n" +
```

```
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
 be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
```

```
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
```

```
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
```

```
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
```

```
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();
```

```
            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
```

```
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
```

```
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }
```

```
            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
```

```
                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
```

```
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient
```

```
.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
```

```
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para Java 2.x

Mostra como criar uma AWS Lambda função usando a API de tempo de execução Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma função do Lambda invocada pelo Amazon API Gateway que verifica uma tabela do Amazon DynamoDB em busca de aniversários de trabalho e usa o Amazon Simple Notification Service (Amazon SNS) para enviar uma mensagem de texto aos seus funcionários que os parabeniza em sua data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Java 2.x

Mostra como criar um evento EventBridge agendado pela Amazon que invoca uma AWS Lambda função. Configure EventBridge para usar uma expressão cron para agendar quando a função Lambda for invocada. Neste exemplo, você cria uma função do Lambda usando a API de runtime de Java do Lambda. Este exemplo invoca diferentes AWS serviços para realizar um caso de uso específico. Este exemplo mostra como criar uma aplicação que envia uma mensagem de texto móvel para seus funcionários que os parabeniza na data de aniversário de um ano.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;
```

```
public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

Exemplos do Amazon SQS usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon SQS

O exemplo de código a seguir mostra como começar a usar o Amazon SQS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
```

```
                .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                    .flatMap(r -> r.queueUrls().stream())
                    .forEach(content -> System.out.println(" Queue URL: " +
 content.toLowerCase()));

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK for Java 2.x da API.

Ações

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");
```

```
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void listQueues(SqsClient sqsClient) {

        System.out.println("\nList Queues");
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
        // List queues with filters
        String namePrefix = "queue";
```

```
        ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
                .queueNamePrefix(namePrefix)
                .build();

        ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
        System.out.println("Queue URLs with prefix: " + namePrefix);
        for (String url : listQueuesFilteredResponse.queueUrls()) {
            System.out.println(url);
        }

        System.out.println("\nSend message");
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody("Hello world!")
                    .delaySeconds(10)
                    .build());

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

        System.out.println("\nSend multiple messages");
        try {
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                    .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

        System.out.println("\nReceive messages");
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

        System.out.println("\nChange Message Visibility");
        try {

            for (Message message : messages) {
                ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .visibilityTimeout(100)
                        .build();
                sqsClient.changeMessageVisibility(req);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
 List<Message> messages) {
        System.out.println("\nDelete Messages");

        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
 DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateQueue em Referência AWS SDK for Java 2.x da API.

`DeleteMessage`

O código de exemplo a seguir mostra como usar `DeleteMessage`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
 DeleteMessageRequest.builder()
```

```
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- Para obter detalhes da API, consulte DeleteMessage em Referência AWS SDK for Java 2.x da API.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteQueue {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName>

                Where:
                   queueName - The name of the Amazon SQS queue to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        SqsClient sqs = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        deleteSQSQueue(sqs, queueName);
        sqs.close();
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteQueue em Referência AWS SDK for Java 2.x da API.

GetQueueUrl

O código de exemplo a seguir mostra como usar `GetQueueUrl`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
```

- Para obter detalhes da API, consulte GetQueueUrl em Referência AWS SDK for Java 2.x da API.

ListQueues

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        String prefix = "que";
```

```
        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK for Java 2.x da API.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return null;
```

- Para obter detalhes da API, consulte ReceiveMessage em Referência AWS SDK for Java 2.x da API.

SendMessage

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Seguem dois exemplos da operação `SendMessage`:

- Enviar uma mensagem com um corpo e um atraso
- Enviar uma mensagem com um corpo e atributos de mensagem

Enviar uma mensagem com um corpo e um atraso.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName> <message>

                Where:
                   queueName - The name of the queue.
                   message - The message to send.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }

    public static void sendMessage(SqsClient sqsClient, String queueName, String
 message) {
        try {
            CreateQueueRequest request = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
            sqsClient.createQueue(request);

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(message)
                    .delaySeconds(5)
                    .build();
```

```
            sqsClient.sendMessage(sendMsgRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Enviar uma mensagem com um corpo e atributos de mensagem.

```
    /**
     * <p>This method demonstrates how to add message attributes to a message.
     * Each attribute must specify a name, value, and data type. You use a Java Map
 to supply the attributes. The map's
     * key is the attribute name, and you specify the map's entry value using a
 builder that includes the attribute
     * value and data type.</p>
     *
     * <p>The data type must start with one of "String", "Number" or "Binary". You
 can optionally
     * define a custom extension by using a "." and your extension.</p>
     *
     * <p>The SQS Developer Guide provides more information on @see <a
     * href="https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-message-metadata.html#sqs-message-attributes">message
     * attributes</a>.</p>
     *
     * @param thumbailPath Filesystem path of the image.
     * @param queueUrl     URL of the SQS queue.
     */
    static void sendMessageWithAttributes(Path thumbailPath, String queueUrl) {
        Map<String, MessageAttributeValue> messageAttributeMap;
        try {
            messageAttributeMap = Map.of(
                    "Name", MessageAttributeValue.builder()
                            .stringValue("Jane Doe")
                            .dataType("String").build(),
                    "Age", MessageAttributeValue.builder()
                            .stringValue("42")
                            .dataType("Number.int").build(),
```

```
                    "Image", MessageAttributeValue.builder()

.binaryValue(SdkBytes.fromByteArray(Files.readAllBytes(thumbailPath)))
                            .dataType("Binary.jpg").build()
            );
        } catch (IOException e) {
            LOGGER.error("An I/O exception occurred reading thumbnail image: {}",
e.getMessage(), e);
            throw new RuntimeException(e);
        }

        SendMessageRequest request = SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello SQS")
                .messageAttributes(messageAttributeMap)
                .build();
        try {
            SendMessageResponse sendMessageResponse =
SQS_CLIENT.sendMessage(request);
            LOGGER.info("Message ID: {}", sendMessageResponse.messageId());
        } catch (SqsException e) {
            LOGGER.error("Exception occurred sending message: {}", e.getMessage(),
e);
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte SendMessage em Referência AWS SDK for Java 2.x da API.

`SendMessageBatch`

O código de exemplo a seguir mostra como usar `SendMessageBatch`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                    .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Para obter detalhes da API, consulte SendMessageBatch em Referência AWS SDK for Java 2.x da API.

`SetQueueAttributes`

O código de exemplo a seguir mostra como usar `SetQueueAttributes`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure um Amazon SQS para usar a criptografia do lado do servidor (SSE) usando uma chave do KMS personalizada.

```
    public static void addEncryption(String queueName, String kmsMasterKeyAlias) {
        SqsClient sqsClient = SqsClient.create();

        GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

        GetQueueUrlResponse getQueueUrlResponse;
        try {
```

```
            getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
        } catch (QueueDoesNotExistException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        }
        String queueUrl = getQueueUrlResponse.queueUrl();


        Map<QueueAttributeName, String> attributes = Map.of(
                QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
                QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" // Set
the data key reuse period to 140 seconds.
        );                                                                  // This
is how long SQS can reuse the data key before requesting a new one from KMS.

        SetQueueAttributesRequest attRequest = SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attributes)
                .build();
        try {
            sqsClient.setQueueAttributes(attRequest);
            LOGGER.info("The attributes have been applied to {}", queueName);
        } catch (InvalidAttributeNameException | InvalidAttributeValueException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        } finally {
            sqsClient.close();
        }
    }
```

- Para obter detalhes da API, consulte SetQueueAttributes em Referência AWS SDK for Java 2.x da API.

Cenários

Crie um aplicativo de mensagem

O exemplo de código a seguir mostra como criar uma aplicação de mensagens usando o Amazon SQS.

SDK para Java 2.x

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Criar e publicar em um tópico FIFO

O exemplo de código a seguir mostra como criar e publicar em um tópico FIFO do Amazon SNS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo

- cria um tópico FIFO do Amazon SNS, duas filas FIFO do Amazon SQS e uma fila padrão.
- inscreve as filas no tópico e publica a mensagem no tópico.

O teste verifica o recebimento da mensagem em cada fila. O exemplo completo também mostra a adição de políticas de acesso e exclui os recursos no final.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
```

```
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "   fifoTopicName - The name of the FIFO topic that you want to create.
\n\n" +
            "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "   retailQueueARN - The name of a SQS FIFO queue that will created for
the retail consumer. \n\n" +
            "   analyticsQueueARN - The name of a SQS standard queue that will be
created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
```

```
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
```

```
            // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateTopic
  - Publicar
  - Assinar

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Gerenciar mensagens grandes usando o S3

O exemplo de código a seguir mostra como usar a biblioteca do cliente em versão ampliada do Amazon SQS para trabalhar com mensagens grandes do SQS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;
```

```
/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
 ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);
```

```
        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
```

```
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK for Java 2.x.
- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

Processar notificações de eventos do S3

O exemplo de código a seguir mostra como trabalhar com notificações de eventos do S3 de uma forma orientada a objetos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo mostra como processar o evento de notificação do S3 usando o Amazon SQS.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload and
 logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification API</
a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
     * <groupId>software.amazon.awssdk</groupId>
     * <artifactId>s3-event-notifications</artifactId>
     * <version><LATEST></version>
     * </dependency>
     * <p>
     * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
     * <p>
     * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
     * in AWS SNS or AWS Lambda as well.
     * <p>
     * Note: The S3EventNotification class does not work with messages routed
 through AWS EventBridge.
     */
    static void processS3Events(String bucketName, String queueUrl, String queueArn)
 {
        try {
            // Configure the bucket to send Object Created and Object Tagging
 notifications to an existing SQS queue.
            s3Client.putBucketNotificationConfiguration(b -> b
                    .notificationConfiguration(ncb -> ncb
                            .queueConfigurations(qcb -> qcb
```

```
                                    .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                    .queueArn(queueArn)))
                            .bucket(bucketName)
            ).join();

            triggerS3EventNotifications(bucketName);
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {
                // Display the number of messages that are available in the queue.
                sqsClient.getQueueAttributes(b -> b
                                .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                        ).thenAccept(attributeResponse ->
                                logger.info("Approximate number of messages in the
queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());
```

```
                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}", eventName,
key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {
                    sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Java 2.x

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
```

```
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
```

```
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";
```

```
        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and the
 subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                +
                "        within the five-minute deduplication interval, is accepted
 but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
```

```
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
 be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");
```

```
        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
```

```
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
```

```
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
```

```
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }
```

```
            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

```
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
```

```
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
```

```
   public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
       try {
           SubscribeRequest request;
           if (filterList.isEmpty()) {
               // No filter subscription is added.
               request = SubscribeRequest.builder()
                   .protocol("sqs")
                   .endpoint(queueArn)
                   .returnSubscriptionArn(true)
                   .topicArn(topicArn)
                   .build();

               SubscribeResponse result = snsClient.subscribe(request);
               System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                   "with the subscription ARN " + result.subscriptionArn());
               return result.subscriptionArn();
           } else {
               request = SubscribeRequest.builder()
                   .protocol("sqs")
                   .endpoint(queueArn)
                   .returnSubscriptionArn(true)
                   .topicArn(topicArn)
                   .build();

               SubscribeResponse result = snsClient.subscribe(request);
               System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                   "with the subscription ARN " + result.subscriptionArn());

               String attributeName = "FilterPolicy";
               Gson gson = new Gson();
               String jsonString = "{\"tone\": []}";
               JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
               JsonArray toneArray = jsonObject.getAsJsonArray("tone");
               for (String value : filterList) {
                   toneArray.add(new JsonPrimitive(value));
               }

               String updatedJsonString = gson.toJson(jsonObject);
               System.out.println(updatedJsonString);
               SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
```

```
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);
```

```
        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
```

```
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();
```

```
            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Enviar e receber lotes de mensagens

O exemplo de código a seguir mostra como:

- Criar uma fila do Amazon SQS.
- Enviar lotes de mensagens para a fila.
- Receber lotes de mensagens de uma fila.
- Excluir lotes de mensagens de uma fila.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Conforme mostrado nos exemplos a seguir, você pode gerenciar operações de mensagens em lote com o Amazon SQS usando duas abordagens diferentes com o AWS SDK for Java 2.x:

SendRecvBatch.javausa operações explícitas em lote. Você cria manualmente lotes de mensagens e chama `sendMessageBatch()` e `deleteMessageBatch()` diretamente. Você também gerencia respostas em lote, incluindo mensagens com falha. Essa abordagem oferece controle total sobre o dimensionamento do lote e o gerenciamento de erros. No entanto, ela requer mais código para gerenciar a lógica de agrupamento em lote.

SimpleProducerConsumer.javausa a `SqsAsyncBatchManager` biblioteca de alto nível para o agrupamento automático de solicitações. Você faz chamadas individuais para `sendMessage()` e `deleteMessage()` com as mesmas assinaturas de método do cliente padrão. O SDK armazena automaticamente essas chamadas em buffer e as envia como operações em lote. Essa abordagem requer alterações mínimas no código e, ao mesmo tempo, fornece benefícios de desempenho em lote.

Use lotes explícitos quando precisar de um controle refinado sobre a composição do lote e o gerenciamento de erros. Use o agrupamento automático em lote quando quiser otimizar o desempenho com o mínimo de alterações no código.

SendRecvBatch.java - Usa operações explícitas em lote com mensagens.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.BatchResultErrorEntry;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
```

```
import software.amazon.awssdk.services.sqs.model.MessageAttributeValue;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

/**
 * This code demonstrates basic message operations in Amazon Simple Queue Service
 (Amazon SQS).
 */

public class SendRecvBatch {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SendRecvBatch.class);
    private static final SqsClient sqsClient = SqsClient.create();


    public static void main(String[] args) {
        usageDemo();
    }
    /**
     * Send a batch of messages in a single request to an SQS queue.
```

```
    * This request may return overall success even when some messages were not
sent.
    * The caller must inspect the Successful and Failed lists in the response and
    * resend any failed messages.
    *
    * @param queueUrl  The URL of the queue to receive the messages.
    * @param messages  The messages to send to the queue. Each message contains a
body and attributes.
    * @return The response from SQS that contains the list of successful and failed
messages.
    */
   public static SendMessageBatchResponse sendMessages(
           String queueUrl, List<MessageEntry> messages) {

       try {
           List<SendMessageBatchRequestEntry> entries = new ArrayList<>();

           for (int i = 0; i < messages.size(); i++) {
               MessageEntry message = messages.get(i);
               entries.add(SendMessageBatchRequestEntry.builder()
                       .id(String.valueOf(i))
                       .messageBody(message.getBody())
                       .messageAttributes(message.getAttributes())
                       .build());
           }

           SendMessageBatchRequest sendBatchRequest =
SendMessageBatchRequest.builder()
                   .queueUrl(queueUrl)
                   .entries(entries)
                   .build();

           SendMessageBatchResponse response =
sqsClient.sendMessageBatch(sendBatchRequest);

           if (!response.successful().isEmpty()) {
               for (SendMessageBatchResultEntry resultEntry :
response.successful()) {
                   LOGGER.info("Message sent: {}: {}", resultEntry.messageId(),

messages.get(Integer.parseInt(resultEntry.id())).getBody());
               }
           }
```

```
            if (!response.failed().isEmpty()) {
                for (BatchResultErrorEntry errorEntry : response.failed()) {
                    LOGGER.warn("Failed to send: {}: {}", errorEntry.id(),

messages.get(Integer.parseInt(errorEntry.id())).getBody());
                }
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Send messages failed to queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Receive a batch of messages in a single request from an SQS queue.
     *
     * @param queueUrl   The URL of the queue from which to receive messages.
     * @param maxNumber  The maximum number of messages to receive (capped at 10 by
SQS).
     *                   The actual number of messages received might be less.
     * @param waitTime   The maximum time to wait (in seconds) before returning.
When
     *                   this number is greater than zero, long polling is used.
This
     *                   can result in reduced costs and fewer false empty
responses.
     * @return The list of Message objects received. These each contain the body
     *         of the message and metadata and custom attributes.
     */
    public static List<Message> receiveMessages(String queueUrl, int maxNumber, int
waitTime) {
        try {
            ReceiveMessageRequest receiveRequest = ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(maxNumber)
                    .waitTimeSeconds(waitTime)
                    .messageAttributeNames("All")
                    .build();

            List<Message> messages =
sqsClient.receiveMessage(receiveRequest).messages();
```

```
            for (Message message : messages) {
                LOGGER.info("Received message: {}: {}", message.messageId(),
message.body());
            }

            return messages;

        } catch (SqsException e) {
            LOGGER.error("Couldn't receive messages from queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Delete a batch of messages from a queue in a single request.
     *
     * @param queueUrl  The URL of the queue from which to delete the messages.
     * @param messages  The list of messages to delete.
     * @return The response from SQS that contains the list of successful and failed
     *         message deletions.
     */
    public static DeleteMessageBatchResponse deleteMessages(String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();

            for (int i = 0; i < messages.size(); i++) {
                entries.add(DeleteMessageBatchRequestEntry.builder()
                        .id(String.valueOf(i))
                        .receiptHandle(messages.get(i).receiptHandle())
                        .build());
            }

            DeleteMessageBatchRequest deleteRequest =
DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            DeleteMessageBatchResponse response =
sqsClient.deleteMessageBatch(deleteRequest);

            if (!response.successful().isEmpty()) {
```

```
                for (DeleteMessageBatchResultEntry resultEntry :
response.successful()) {
                    LOGGER.info("Deleted {}",
messages.get(Integer.parseInt(resultEntry.id())).receiptHandle());
                }
            }

            if (!response.failed().isEmpty()) {
                for (BatchResultErrorEntry errorEntry : response.failed()) {
                    LOGGER.warn("Could not delete {}",
messages.get(Integer.parseInt(errorEntry.id())).receiptHandle());
                }
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Couldn't delete messages from queue {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Helper class to represent a message with body and attributes.
     */
    public static class MessageEntry {
        private final String body;
        private final Map<String, MessageAttributeValue> attributes;

        public MessageEntry(String body, Map<String, MessageAttributeValue>
attributes) {
            this.body = body;
            this.attributes = attributes != null ? attributes : new HashMap<>();
        }

        public String getBody() {
            return body;
        }

        public Map<String, MessageAttributeValue> getAttributes() {
            return attributes;
        }
    }
```

```
    /**
     * Shows how to:
     * * Read the lines from a file and send the lines in
     *   batches of 10 as messages to a queue.
     * * Receive the messages in batches until the queue is empty.
     * * Reassemble the lines of the file and verify they match the original file.
     */
    public static void usageDemo() {
        LOGGER.info("-".repeat(88));
        LOGGER.info("Welcome to the Amazon Simple Queue Service (Amazon SQS)
demo!");
        LOGGER.info("-".repeat(88));

        String queueUrl = null;
        try {
            // Create a queue for the demo.
            String queueName = "sqs-usage-demo-message-wrapper-" +
System.currentTimeMillis();
            CreateQueueRequest createRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
            queueUrl = sqsClient.createQueue(createRequest).queueUrl();
            LOGGER.info("Created queue: {}", queueUrl);

            try (InputStream inputStream =
SendRecvBatch.class.getResourceAsStream("/log4j2.xml");
                 BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream))) {

                List<String> lines = reader.lines().toList();

                // Send file lines in batches.
                int batchSize = 10;
                LOGGER.info("Sending file lines in batches of {} as messages.",
batchSize);

                for (int i = 0; i < lines.size(); i += batchSize) {
                    List<MessageEntry> messageBatch = new ArrayList<>();

                    for (int j = i; j < Math.min(i + batchSize, lines.size()); j++)
{
                        String line = lines.get(j);
                        if (line == null || line.trim().isEmpty()) {
                            continue; // Skip empty lines.
```

```
                        }

                        Map<String, MessageAttributeValue> attributes = new
HashMap<>();
                        attributes.put("line", MessageAttributeValue.builder()
                                .dataType("String")
                                .stringValue(String.valueOf(j))
                                .build());

                        messageBatch.add(new MessageEntry(lines.get(j),
attributes));
                    }

                    sendMessages(queueUrl, messageBatch);
                    System.out.print(".");
                    System.out.flush();
                }

                LOGGER.info("\nDone. Sent {} messages.", lines.size());

                // Receive and process messages.
                LOGGER.info("Receiving, handling, and deleting messages in batches
of {}.", batchSize);
                String[] receivedLines = new String[lines.size()];
                boolean moreMessages = true;

                while (moreMessages) {
                    List<Message> receivedMessages = receiveMessages(queueUrl,
batchSize, 5);

                    for (Message message : receivedMessages) {
                        int lineNumber =
Integer.parseInt(message.messageAttributes().get("line").stringValue());
                        receivedLines[lineNumber] = message.body();
                    }

                    if (!receivedMessages.isEmpty()) {
                        deleteMessages(queueUrl, receivedMessages);
                    } else {
                        moreMessages = false;
                    }
                }

                LOGGER.info("\nDone.");
```

```
                // Verify that all lines were received correctly.
                boolean allLinesMatch = true;
                for (int i = 0; i < lines.size(); i++) {
                    String originalLine = lines.get(i);
                    String receivedLine = receivedLines[i] == null ? "" :
receivedLines[i];

                    if (!originalLine.equals(receivedLine)) {
                        allLinesMatch = false;
                        break;
                    }
                }

                if (allLinesMatch) {
                    LOGGER.info("Successfully reassembled all file lines!");
                } else {
                    LOGGER.info("Uh oh, some lines were missed!");
                }
            }
        } catch (SqsException e) {
            LOGGER.error("SQS operation failed", e);
        } catch (RuntimeException | IOException e) {
            LOGGER.error("Unexpected runtime error during demo", e);
        } finally {
            // Clean up by deleting the queue if it was created.
            if (queueUrl != null) {
                try {
                    DeleteQueueRequest deleteQueueRequest =
DeleteQueueRequest.builder()
                            .queueUrl(queueUrl)
                            .build();
                    sqsClient.deleteQueue(deleteQueueRequest);
                    LOGGER.info("Deleted queue: {}", queueUrl);
                } catch (SqsException e) {
                    LOGGER.error("Failed to delete queue: {}", queueUrl, e);
                }
            }
        }

        LOGGER.info("Thanks for watching!");
        LOGGER.info("-".repeat(88));
    }
}
```

SimpleProducerConsumer.java - Usa o agrupamento automático de mensagens.

```
package com.example.sqs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsAsyncClient;
import software.amazon.awssdk.services.sqs.batchmanager.SqsAsyncBatchManager;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageResponse;
import software.amazon.awssdk.core.exception.SdkException;

import java.math.BigInteger;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Demonstrates the AWS SDK for Java 2.x Automatic Request Batching API for Amazon
 SQS.
 *
 * This example showcases the high-level SqsAsyncBatchManager library that provides
 * efficient batching and buffering for SQS operations. The batch manager offers
 * methods that directly mirror SqsAsyncClient methods—sendMessage,
 changeMessageVisibility,
 * deleteMessage, and receiveMessage—making it a drop-in replacement with minimal
 code changes.
 *
 * Key features of the SqsAsyncBatchManager:
 * - Automatic batching: The SDK automatically buffers individual requests and sends
 them
```

```
 *   as batches when maxBatchSize (default: 10) or sendRequestFrequency (default:
 200ms)
 *   thresholds are reached
 * - Familiar API: Method signatures match SqsAsyncClient exactly, requiring no
 learning curve
 * - Background optimization: The batch manager maintains internal buffers and
 handles
 *   batching logic transparently
 * - Asynchronous operations: All methods return CompletableFuture for non-blocking
 execution
 *
 * Performance benefits demonstrated:
 * - Reduced API calls: Multiple individual requests are consolidated into single
 batch operations
 * - Lower costs: Fewer API calls result in reduced SQS charges
 * - Higher throughput: Batch operations process more messages per second
 * - Efficient resource utilization: Fewer network round trips and better connection
 reuse
 *
 * This example compares:
 * 1. Single-message operations using SqsAsyncClient directly
 * 2. Batch operations using SqsAsyncBatchManager with identical method calls
 *
 * Usage patterns:
 * - Set batch size to 1 to use SqsAsyncClient for baseline performance measurement
 * - Set batch size > 1 to use SqsAsyncBatchManager for optimized batch processing
 * - Monitor real-time throughput metrics to observe performance improvements
 *
 * Prerequisites:
 * - AWS SDK for Java 2.x version 2.28.0 or later
 * - An existing SQS queue
 * - Valid AWS credentials configured
 *
 * The program displays real-time metrics showing the dramatic performance
 difference
 * between individual operations and automatic batching.
 */
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Logger log =
 LoggerFactory.getLogger(SimpleProducerConsumer.class);
```

```
    /**
     * Runs the SQS batching demonstration with user-configured parameters.
     *
     * Prompts for queue name, thread counts, batch size, message size, and runtime.
     * Creates producer and consumer threads to demonstrate batching performance.
     *
     * @param args command line arguments (not used)
     * @throws InterruptedException if thread operations are interrupted
     */
    public static void main(String[] args) throws InterruptedException {

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
        final int producerCount = input.nextInt();

        System.out.print("Enter the number of consumers: ");
        final int consumerCount = input.nextInt();

        System.out.print("Enter the number of messages per batch: ");
        final int batchSize = input.nextInt();

        System.out.print("Enter the message size in bytes: ");
        final int messageSizeByte = input.nextInt();

        System.out.print("Enter the run time in minutes: ");
        final int runTimeMinutes = input.nextInt();

        // Create SQS async client and batch manager for all operations.
        // The SqsAsyncBatchManager is created from the SqsAsyncClient using the
        // batchManager() factory method, which provides default batching
configuration.
        // This high-level library automatically handles request buffering and
batching
        // while maintaining the same method signatures as SqsAsyncClient.
        final SqsAsyncClient sqsAsyncClient = SqsAsyncClient.create();
        final SqsAsyncBatchManager batchManager = sqsAsyncClient.batchManager();

        final String queueUrl =
sqsAsyncClient.getQueueUrl(GetQueueUrlRequest.builder()
                .queueName(queueName)
```

```
                .build()).join().queueUrl();

        // The flag used to stop producer, consumer, and monitor threads.
        final AtomicBoolean stop = new AtomicBoolean(false);

        // Start the producers.
        final AtomicInteger producedCount = new AtomicInteger();
        final Thread[] producers = new Thread[producerCount];
        for (int i = 0; i < producerCount; i++) {
            if (batchSize == 1) {
                producers[i] = new Producer(sqsAsyncClient, queueUrl,
messageSizeByte,
                        producedCount, stop);
            } else {
                producers[i] = new BatchProducer(batchManager, queueUrl, batchSize,
                        messageSizeByte, producedCount, stop);
            }
            producers[i].start();
        }

        // Start the consumers.
        final AtomicInteger consumedCount = new AtomicInteger();
        final Thread[] consumers = new Thread[consumerCount];
        for (int i = 0; i < consumerCount; i++) {
            if (batchSize == 1) {
                consumers[i] = new Consumer(sqsAsyncClient, queueUrl, consumedCount,
stop);
            } else {
                consumers[i] = new BatchConsumer(batchManager, queueUrl, batchSize,
                        consumedCount, stop);
            }
            consumers[i].start();
        }

        // Start the monitor thread.
        final Thread monitor = new Monitor(producedCount, consumedCount, stop);
        monitor.start();

        // Wait for the specified amount of time then stop.
        Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runTimeMinutes,
                MAX_RUNTIME_MINUTES)));
        stop.set(true);

        // Join all threads.
```

```
        for (int i = 0; i < producerCount; i++) {
            producers[i].join();
        }

        for (int i = 0; i < consumerCount; i++) {
            consumers[i].join();
        }

        monitor.interrupt();
        monitor.join();

        // Close resources
        batchManager.close();
        sqsAsyncClient.close();
    }

    /**
     * Creates a random string of approximately the specified size in bytes.
     *
     * @param sizeByte the target size in bytes for the generated string
     * @return a random string encoded in base-32
     */
    private static String makeRandomString(int sizeByte) {
        final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
        new Random().nextBytes(bs);
        bs[0] = (byte) ((bs[0] | 64) & 127);
        return new BigInteger(bs).toString(32);
    }

    /**
     * Sends messages individually using SqsAsyncClient for baseline performance
measurement.
     *
     * This producer demonstrates traditional single-message operations without
batching.
     * Each sendMessage() call results in a separate API request to SQS, providing
     * a performance baseline for comparison with the batch operations.
     *
     * The sendMessage() method signature is identical to
SqsAsyncBatchManager.sendMessage(),
     * showing how the high-level batching library maintains API compatibility while
     * adding automatic optimization behind the scenes.
     */
    private static class Producer extends Thread {
```

```
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for sending messages
         * @param queueUrl the URL of the target queue
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        Producer(SqsAsyncClient sqsAsyncClient, String queueUrl, int
messageSizeByte,
                 AtomicInteger producedCount, AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends messages until the stop flag is set.
         *
         * Uses SqsAsyncClient.sendMessage() directly, resulting in one API call per
message.
         * This approach provides baseline performance metrics for comparison with
batching.
         * Each call blocks until the individual message is sent, demonstrating
traditional
         * one-request-per-operation behavior.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    sqsAsyncClient.sendMessage(SendMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .messageBody(theMessage)
                            .build()).join();
                    producedCount.incrementAndGet();
```

```
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Producer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Sends messages using SqsAsyncBatchManager for automatic request batching and
optimization.
     *
     * This producer demonstrates the AWS SDK for Java 2.x high-level batching
library.
     * The SqsAsyncBatchManager automatically buffers individual sendMessage() calls
and
     * sends them as batches when thresholds are reached:
     * - maxBatchSize: Maximum 10 messages per batch (default)
     * - sendRequestFrequency: 200ms timeout before sending partial batches
(default)
     *
     * Key advantages of the batching approach:
     * - Identical API: batchManager.sendMessage() has the same signature as
sqsAsyncClient.sendMessage()
     * - Automatic optimization: No code changes needed to benefit from batching
     * - Transparent buffering: The SDK handles batching logic internally
     * - Reduced API calls: Multiple messages sent in single batch requests
     * - Lower costs: Fewer API calls result in reduced SQS charges
     * - Higher throughput: Batch operations process significantly more messages per
second
     */
    private static class BatchProducer extends Thread {
        final SqsAsyncBatchManager batchManager;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for batch operations.
```

```
         *
         * @param batchManager the batch manager for efficient message sending
         * @param queueUrl the URL of the target queue
         * @param batchSize the number of messages to send per batch
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        BatchProducer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      int messageSizeByte, AtomicInteger producedCount,
                      AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends batches of messages using the high-level batching
library.
         *
         * Notice how batchManager.sendMessage() uses the exact same method
signature
         * and request builder pattern as SqsAsyncClient.sendMessage(). This
demonstrates
         * the drop-in replacement capability of the SqsAsyncBatchManager.
         *
         * The SDK automatically:
         * - Buffers individual sendMessage() calls internally
         * - Groups them into batch requests when thresholds are met
         * - Sends SendMessageBatchRequest operations to SQS
         * - Returns individual CompletableFuture responses for each message
         *
         * This transparent batching provides significant performance improvements
         * without requiring changes to application logic or error handling
patterns.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    // Send multiple messages using the high-level batch manager.
```

```
                    // Each batchManager.sendMessage() call uses identical syntax to
                    // sqsAsyncClient.sendMessage(), demonstrating API
compatibility.
                    // The SDK automatically buffers these calls and sends them as
                    // batch operations when maxBatchSize (10) or
sendRequestFrequency (200ms)
                    // thresholds are reached, significantly improving throughput.
                    for (int i = 0; i < batchSize; i++) {
                        CompletableFuture<SendMessageResponse> future =
batchManager.sendMessage(
                                SendMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .messageBody(theMessage)
                                        .build());

                        // Handle the response asynchronously
                        future.whenComplete((response, throwable) -> {
                            if (throwable == null) {
                                producedCount.incrementAndGet();
                            } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                        !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                log.error("BatchProducer: Failed to send message",
throwable);
                            }
                            // Ignore CancellationException and executor shutdown
errors - expected during shutdown
                        });
                    }

                    // Small delay to allow batching to occur
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchProducer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                log.error("BatchProducer: " + e.getMessage());
                System.exit(1);
            }
        }
    }
```

```
    /**
     * Receives and deletes messages individually using SqsAsyncClient for baseline
 measurement.
     *
     * This consumer demonstrates traditional single-message operations without
 batching.
     * Each receiveMessage() and deleteMessage() call results in separate API
 requests,
     * providing a performance baseline for comparison with batch operations.
     *
     * The method signatures are identical to SqsAsyncBatchManager methods:
     * - receiveMessage() matches batchManager.receiveMessage()
     * - deleteMessage() matches batchManager.deleteMessage()
     *
     * This API consistency allows easy migration to the high-level batching
 library.
     */
    private static class Consumer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger consumedCount;
        final AtomicBoolean stop;

        /**
         * Creates a consumer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for receiving messages
         * @param queueUrl the URL of the source queue
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
        Consumer(SqsAsyncClient sqsAsyncClient, String queueUrl, AtomicInteger
 consumedCount,
                 AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using traditional single-
request operations.
         *
```

```
        * Uses SqsAsyncClient methods directly:
        * - receiveMessage(): One API call per receive operation
        * - deleteMessage(): One API call per delete operation
        *
        * This approach demonstrates the baseline performance without batching
optimization.
        * Compare these method calls with the identical signatures used in
BatchConsumer
        * to see how the high-level batching library maintains API compatibility.
        */
       public void run() {
           try {
               while (!stop.get()) {
                   try {
                       final ReceiveMessageResponse result =
sqsAsyncClient.receiveMessage(
                               ReceiveMessageRequest.builder()
                                       .queueUrl(queueUrl)
                                       .build()).join();

                       if (!result.messages().isEmpty()) {
                           final Message m = result.messages().get(0);
                           // Note: deleteMessage() signature identical to
batchManager.deleteMessage()

sqsAsyncClient.deleteMessage(DeleteMessageRequest.builder()
                                   .queueUrl(queueUrl)
                                   .receiptHandle(m.receiptHandle())
                                   .build()).join();
                           consumedCount.incrementAndGet();
                       }
                   } catch (SdkException | java.util.concurrent.CompletionException
e) {
                       log.error(e.getMessage());
                   }
               }
           } catch (SdkException | java.util.concurrent.CompletionException e) {
               // Handle both SdkException and CompletionException from async
operations.
               // If this unlikely condition occurs, stop.
               log.error("Consumer: " + e.getMessage());
               System.exit(1);
           }
       }
```

```
    }

    /**
     * Receives and deletes messages using SqsAsyncBatchManager for automatic
optimization.
     *
     * This consumer demonstrates the AWS SDK for Java 2.x high-level batching
library
     * for message consumption. The SqsAsyncBatchManager provides two key
optimizations:
     *
     * 1. Receive optimization: Maintains an internal buffer of messages fetched in
the
     *    background, so receiveMessage() calls return immediately from the buffer
     * 2. Delete batching: Automatically buffers deleteMessage() calls and sends
them
     *    as DeleteMessageBatchRequest operations when thresholds are reached
     *
     * Key features:
     * - Identical API: receiveMessage() and deleteMessage() have the same
signatures
     *   as SqsAsyncClient methods, making this a true drop-in replacement
     * - Background fetching: The batch manager continuously fetches messages to
keep
     *   the internal buffer populated, reducing receive latency
     * - Automatic delete batching: Individual deleteMessage() calls are buffered
and
     *   sent as batch operations (up to 10 per batch, 200ms frequency)
     * - Transparent optimization: No application logic changes needed to benefit
     *
     * Performance benefits:
     * - Reduced API calls through automatic batching of delete operations
     * - Lower latency for receives due to background message buffering
     * - Higher overall throughput with fewer network round trips
     */
    private static class BatchConsumer extends Thread {
        final SqsAsyncBatchManager batchManager;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger consumedCount;
        final AtomicBoolean stop;

        /**
         * Creates a consumer thread for batch operations.
```

```
         *
         * @param batchManager the batch manager for efficient message processing
         * @param queueUrl the URL of the source queue
         * @param batchSize the maximum number of messages to receive per batch
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
        BatchConsumer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      AtomicInteger consumedCount, AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using the high-level batching
library.
         *
         * Demonstrates the key advantage of SqsAsyncBatchManager: identical method
signatures
         * with automatic optimization. Notice how:
         *
         * - batchManager.receiveMessage() uses the same syntax as
sqsAsyncClient.receiveMessage()
         * - batchManager.deleteMessage() uses the same syntax as
sqsAsyncClient.deleteMessage()
         *
         * Behind the scenes, the batch manager:
         * 1. Maintains an internal message buffer populated by background fetching
         * 2. Returns messages immediately from the buffer (reduced latency)
         * 3. Automatically batches deleteMessage() calls into
DeleteMessageBatchRequest operations
         * 4. Sends batch deletes when maxBatchSize (10) or sendRequestFrequency
(200ms) is reached
         *
         * This provides significant performance improvements with zero code changes
         * compared to traditional SqsAsyncClient usage patterns.
         */
        public void run() {
            try {
                while (!stop.get()) {
```

```
                    // Receive messages using the high-level batch manager.
                    // This call uses identical syntax to
sqsAsyncClient.receiveMessage()
                    // but benefits from internal message buffering for improved
performance.
                    final ReceiveMessageResponse result =
batchManager.receiveMessage(
                            ReceiveMessageRequest.builder()
                                    .queueUrl(queueUrl)
                                    .maxNumberOfMessages(Math.min(batchSize, 10))
                                    .build()).join();

                    if (!result.messages().isEmpty()) {
                        final List<Message> messages = result.messages();

                        // Delete messages using the batch manager.
                        // Each deleteMessage() call uses identical syntax to
SqsAsyncClient
                        // but the SDK automatically buffers these calls and sends
them
                        // as DeleteMessageBatchRequest operations for optimal
performance.
                        for (Message message : messages) {
                            CompletableFuture<DeleteMessageResponse> future =
batchManager.deleteMessage(
                                    DeleteMessageRequest.builder()
                                            .queueUrl(queueUrl)
                                            .receiptHandle(message.receiptHandle())
                                            .build());

                            future.whenComplete((response, throwable) -> {
                                if (throwable == null) {
                                    consumedCount.incrementAndGet();
                                } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                            !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                    log.error("BatchConsumer: Failed to delete
message", throwable);
                                }
                                // Ignore CancellationException and executor
shutdown errors - expected during shutdown
                            });
                        }
```

```
                    }

                    // Small delay to prevent tight polling
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchConsumer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("BatchConsumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Displays real-time throughput statistics every second.
     *
     * This thread logs the current count of produced and consumed messages
     * to help you monitor the performance comparison.
     */
    private static class Monitor extends Thread {
        private final AtomicInteger producedCount;
        private final AtomicInteger consumedCount;
        private final AtomicBoolean stop;

        /**
         * Creates a monitoring thread that displays throughput statistics.
         *
         * @param producedCount shared counter for messages sent
         * @param consumedCount shared counter for messages processed
         * @param stop shared flag to signal thread termination
         */
        Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
                AtomicBoolean stop) {
            this.producedCount = producedCount;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
```

```
         * Logs throughput statistics every second until stopped.
         *
         * Displays the current count of produced and consumed messages
         * to help monitor the performance comparison between batching strategies.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    Thread.sleep(1000);
                    log.info("produced messages = " + producedCount.get()
                            + ", consumed messages = " + consumedCount.get());
                }
            } catch (InterruptedException e) {
                // Allow the thread to exit.
            }
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage
  - SendMessage
  - SendMessageBatch

Usar a Biblioteca de Mensagens Java do Amazon SQS para trabalhar com a interface do JMS

O exemplo de código a seguir mostra como usar a Biblioteca de Mensagens Java do Amazon SQS para trabalhar com a interface do JMS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Os exemplos a seguir funcionam com filas padrão do Amazon SQS e incluem:

- Envio de uma mensagem de texto.
- Recebimento de mensagens de forma síncrona.
- Recebimento de mensagens de forma assíncrona.
- Recebimento de mensagens usando o modo CLIENT_ACKNOWLEDGE.
- Recebimento de mensagens usando o modo UNORDERED_ACKNOWLEDGE.
- Uso do Spring para injetar dependências.
- Uma classe de utilitário que fornece métodos comuns usados pelos outros exemplos.

Consulte mais informações sobre como usar o JMS com o Amazon SQS no Guia do desenvolvedor do Amazon SQS.

Envio de uma mensagem de texto.

```
   /**
    * This method establishes a connection to a standard Amazon SQS queue using the
Amazon SQS
    * Java Messaging Library and sends text messages to it. It uses JMS (Java
Message Service) API
    * with automatic acknowledgment mode to ensure reliable message delivery, and
automatically
    * manages all messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or sending messages
to the queue
    */
   public static void doSendTextMessage() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );
```

```
        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try (SQSConnection connection = connectionFactory.createConnection()) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session that uses the JMS auto-acknowledge mode.
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));

            createAndSendMessages(session, producer);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
    }

    /**
     * This method reads text input from the keyboard and sends each line as a
separate message
     * to a standard Amazon SQS queue using the Amazon SQS Java Messaging Library.
It continues
     * to accept input until the user enters an empty line, using JMS (Java Message
Service) API to
     * handle the message delivery.
     *
     * @param session The JMS session used to create messages
     * @param producer The JMS message producer used to send messages to the queue
     */
    private static void createAndSendMessages(Session session, MessageProducer
producer) {
        BufferedReader inputReader = new BufferedReader(
                new InputStreamReader(System.in, Charset.defaultCharset()));

        try {
            String input;
            while (true) {
                LOGGER.info("Enter message to send (leave empty to exit): ");
                input = inputReader.readLine();
                if (input == null || input.isEmpty()) break;
```

```
                TextMessage message = session.createTextMessage(input);
                producer.send(message);
                LOGGER.info("Send message {}", message.getJMSMessageID());
            }
        } catch (EOFException e) {
            // Just return on EOF
        } catch (IOException e) {
            LOGGER.error("Failed reading input: {}", e.getMessage(), e);
        } catch (JMSException e) {
            LOGGER.error("Failed sending message: {}", e.getMessage(), e);
        }
    }
```

Recebimento de mensagens de forma síncrona.

```
    /**
     * This method receives messages from a standard Amazon SQS queue using the
Amazon SQS Java
     * Messaging Library. It creates a connection to the queue using JMS (Java
Message Service),
     * waits for messages to arrive, and processes them one at a time. The method
handles all
     * necessary setup and cleanup of messaging resources.
     *
     * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
     */
    public static void doReceiveMessageSync() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session.
```

```
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            connection.start();

            receiveMessages(consumer);
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
    }

    /**
     * This method continuously checks for new messages from a standard Amazon SQS
queue using
     * the Amazon SQS Java Messaging Library. It waits up to 20 seconds for each
message, processes
     * it using JMS (Java Message Service), and confirms receipt. The method stops
checking for
     * messages after 20 seconds of no activity.
     *
     * @param consumer The JMS message consumer that receives messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 1 minute for a message
                Message message =
consumer.receive(Duration.ofSeconds(20).toMillis());
                if (message == null) {
                    LOGGER.info("Shutting down after 20 seconds of silence.");
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Acknowledged message {}", message.getJMSMessageID());
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS: {}", e.getMessage(), e);
        }
    }
```

Recebimento de mensagens de forma assíncrona.

```
   /**
    * This method sets up automatic message handling for a standard Amazon SQS
queue using the
    * Amazon SQS Java Messaging Library. It creates a listener that processes
messages as soon
    * as they arrive using JMS (Java Message Service), runs for 5 seconds, then
cleans up all
    * messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
    */
   public static void doReceiveMessageAsync() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create a connection.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

           try {
               // Create a consumer for the queue.
               MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
               // Provide an implementation of the MessageListener interface, which
has a single 'onMessage' method.
               // We use a lambda expression for the implementation.
               consumer.setMessageListener(message -> {
                   try {
```

```
                        SqsJmsExampleUtils.handleMessage(message);
                        message.acknowledge();
                    } catch (JMSException e) {
                        LOGGER.error("Error processing message: {}",
e.getMessage());
                    }
                });
                // Start receiving incoming messages.
                connection.start();
                LOGGER.info("Waiting for messages...");
            } catch (JMSException e) {
                throw new RuntimeException(e);
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info( "Connection closed" );
    }
```

Recebimento de mensagens usando o modo CLIENT_ACKNOWLEDGE.

```
    /**
     * This method demonstrates how message acknowledgment affects message
processing in a standard
     * Amazon SQS queue using the Amazon SQS Java Messaging Library. It sends
messages to the queue,
     * then shows how JMS (Java Message Service) client acknowledgment mode handles
both explicit
     * and implicit message confirmations, including how acknowledging one message
can automatically
     * acknowledge previous messages.
     *
     * @throws JMSException If there is a problem with the messaging operations
     */
    public static void doReceiveMessagesSyncClientAcknowledge() throws JMSException
{
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
```

```
                SqsClient.create()
        );

        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

            // Create a session with client acknowledge mode.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

            // Create a producer and consumer.
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            // Open the connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue,
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }
```

```
            /*  We will attempt to receive another message, but none will be
available. This is because in
                CLIENT_ACKNOWLEDGE mode, when we acknowledged the second message,
all previous messages were
                automatically acknowledged as well. Therefore, although we never
directly acknowledged the first
                message, it was implicitly acknowledged when we confirmed the second
one. */
            receiveMessage(consumer, true);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed.");

    }


    /**
     * Sends a text message using the specified JMS MessageProducer and Session.
     *
     * @param producer    The JMS MessageProducer used to send the message
     * @param session     The JMS Session used to create the text message
     * @param messageText The text content to be sent in the message
     * @throws JMSException If there is an error creating or sending the message
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }

    /**
     * Receives and processes a message from a JMS queue using the specified
consumer.
     * The method waits for a message until the configured timeout period is
reached.
     * If a message is received, it is logged and optionally acknowledged based on
the
     * acknowledge parameter.
     *
     * @param consumer    The JMS MessageConsumer used to receive messages from the
queue
     * @param acknowledge Boolean flag indicating whether to acknowledge the
message.
     *                    If true, the message will be acknowledged after processing
```

```
    * @throws JMSException If there is an error receiving, processing, or
acknowledging the message
    */
   private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
       // Receive a message.
       Message message = consumer.receive(TIME_OUT_MILLIS);

       if (message == null) {
           LOGGER.info("Queue is empty!");
       } else {
           // Since this queue has only text messages, cast the message object and
print the text.
           LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

           // Acknowledge the message if asked.
           if (acknowledge) message.acknowledge();
       }
   }
```

Recebimento de mensagens usando o modo UNORDERED_ACKNOWLEDGE.

```
   /**
    * Demonstrates message acknowledgment behavior in UNORDERED_ACKNOWLEDGE mode
with Amazon SQS JMS.
    * In this mode, each message must be explicitly acknowledged regardless of
receive order.
    * Unacknowledged messages return to the queue after the visibility timeout
expires,
    * unlike CLIENT_ACKNOWLEDGE mode where acknowledging one message acknowledges
all previous messages.
    *
    * @throws JMSException         If a JMS-related error occurs during message
operations
    */
   public static void doReceiveMessagesUnorderedAcknowledge() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );
```

```
        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try( SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

            // Create a session with unordered acknowledge mode.
            Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

            // Create the producer and consumer.
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            // Open a connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue.
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }
```

```
            /*  We will attempt to receive another message, and we'll get the first
message again. This occurs
                because in UNORDERED_ACKNOWLEDGE mode, each message requires its own
separate acknowledgment.
                Since we only acknowledged the second message, the first message
remains in the queue for
                redelivery. */
            receiveMessage(consumer, true);

            LOGGER.info("Connection closed.");
        } // The connection closes automatically. This also closes the session.
    }

    /**
     * Sends a text message to an Amazon SQS queue using JMS.
     *
     * @param producer    The JMS MessageProducer for the queue
     * @param session     The JMS Session for message creation
     * @param messageText The message content
     * @throws JMSException If message creation or sending fails
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }
    /**
     * Synchronously receives a message from an Amazon SQS queue using the JMS API
     * with an acknowledgment parameter.
     *
     * @param consumer    The JMS MessageConsumer for the queue
     * @param acknowledge If true, acknowledges the message after receipt
     * @throws JMSException If message reception or acknowledgment fails
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object and
print the text.
```

```
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
 message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

Uso do Spring para injetar dependências.

```
package com.example.sqs.jms.spring;

import com.amazon.sqs.javamessaging.SQSConnection;
import com.example.sqs.jms.SqsJmsExampleUtils;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.NoSuchBeanDefinitionException;
import org.springframework.context.support.FileSystemXmlApplicationContext;

import java.io.File;
import java.net.URL;
import java.util.concurrent.TimeUnit;

/**
 * Demonstrates how to send and receive messages using the Amazon SQS Java Messaging
 Library
 * with Spring Framework integration. This example connects to a standard Amazon SQS
 message
 * queue using Spring's dependency injection to configure the connection and
 messaging components.
 * The application uses the JMS (Java Message Service) API to handle message
 operations.
 */
public class SpringExample {
    private static final Integer POLLING_SECONDS = 15;
    private static final String SPRING_XML_CONFIG_FILE =
 "SpringExampleConfiguration.xml.txt";
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SpringExample.class);

    /**
```

```
    * Demonstrates sending and receiving messages through a standard Amazon SQS
message queue
    * using Spring Framework configuration. This method loads connection settings
from an XML file,
    * establishes a messaging session using the Amazon SQS Java Messaging Library,
and processes
    * messages using JMS (Java Message Service) operations. If the queue doesn't
exist, it will
    * be created automatically.
    *
    * @param args Command line arguments (not used)
    */
   public static void main(String[] args) {

       URL resource =
SpringExample.class.getClassLoader().getResource(SPRING_XML_CONFIG_FILE);
       File springFile = new File(resource.getFile());
       if (!springFile.exists() || !springFile.canRead()) {
           LOGGER.error("File " + SPRING_XML_CONFIG_FILE + " doesn't exist or isn't
readable.");
           System.exit(1);
       }

       try (FileSystemXmlApplicationContext context =
                   new FileSystemXmlApplicationContext("file://" +
springFile.getAbsolutePath())) {

           Connection connection;
           try {
               connection = context.getBean(Connection.class);
           } catch (NoSuchBeanDefinitionException e) {
               LOGGER.error("Can't find the JMS connection to use: " +
e.getMessage(), e);
               System.exit(2);
               return;
           }

           String queueName;
           try {
               queueName = context.getBean("queueName", String.class);
           } catch (NoSuchBeanDefinitionException e) {
               LOGGER.error("Can't find the name of the queue to use: " +
e.getMessage(), e);
               System.exit(3);
```

```
                return;
            }
            try {
                if (connection instanceof SQSConnection) {
                    SqsJmsExampleUtils.ensureQueueExists((SQSConnection) connection,
queueName, SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
                }
                // Create the JMS session.
                Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

                SqsJmsExampleUtils.sendTextMessage(session, queueName);
                MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));

                receiveMessages(consumer);
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        }   // Spring context autocloses. Managed Spring beans that implement
AutoClosable, such as the
        // 'connection' bean, are also closed.
        LOGGER.info("Context closed");
    }

    /**
     * Continuously checks for and processes messages from a standard Amazon SQS
message queue
     * using the Amazon SQS Java Messaging Library underlying the JMS API. This
method waits for incoming messages,
     * processes them when they arrive, and acknowledges their receipt using JMS
(Java Message
     * Service) operations. The method will stop checking for messages after 15
seconds of
     * inactivity.
     *
     * @param consumer The JMS message consumer used to receive messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
```

```
                // Wait 15 seconds for a message.
                Message message =
 consumer.receive(TimeUnit.SECONDS.toMillis(POLLING_SECONDS));
                if (message == null) {
                    LOGGER.info("Shutting down after {} seconds of silence.",
 POLLING_SECONDS);
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Message acknowledged.");
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS.", e);
        }
    }
}
```

Definições de bean do Spring.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/
beans/spring-beans-3.0.xsd
        ">
    <!-- Define the AWS Region -->
    <bean id="region" class="software.amazon.awssdk.regions.Region" factory-
method="of">
        <constructor-arg value="us-east-1"/>
    </bean>

    <bean id="credentialsProviderBean"
 class="software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider"
          factory-method="create"/>

    <bean id="clientBuilder" class="software.amazon.awssdk.services.sqs.SqsClient"
 factory-method="builder"/>
```

```
    <bean id="regionSetClientBuilder" factory-bean="clientBuilder" factory-
method="region">
        <constructor-arg ref="region"/>
    </bean>

    <!-- Configure the Builder with Credentials Provider -->
    <bean id="sqsClient" factory-bean="regionSetClientBuilder" factory-
method="credentialsProvider">
        <constructor-arg ref="credentialsProviderBean"/>
    </bean>

    <bean id="providerConfiguration"
 class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="connectionFactory"
 class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="providerConfiguration"/>
        <constructor-arg ref="clientBuilder"/>
    </bean>

    <bean id="connection"
          factory-bean="connectionFactory"
          factory-method="createConnection"
          init-method="start"
          destroy-method="close"/>

    <bean id="queueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
    </bean>
</beans>
```

Uma classe de utilitário que fornece métodos comuns usados pelos outros exemplos.

```
package com.example.sqs.jms;

import com.amazon.sqs.javamessaging.AmazonSQSMessagingClientWrapper;
import com.amazon.sqs.javamessaging.ProviderConfiguration;
import com.amazon.sqs.javamessaging.SQSConnection;
import com.amazon.sqs.javamessaging.SQSConnectionFactory;
```

```
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;

import java.time.Duration;
import java.util.Base64;
import java.util.Map;

/**
 * This utility class provides helper methods for working with Amazon Simple Queue
 Service (Amazon SQS)
 * through the Java Message Service (JMS) interface. It contains common operations
 for managing message
 * queues and handling message delivery.
 */
public class SqsJmsExampleUtils {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SqsJmsExampleUtils.class);
    public static final Long QUEUE_VISIBILITY_TIMEOUT = 5L;

    /**
     * This method verifies that a message queue exists and creates it if necessary.
 The method checks for
     * an existing queue first to optimize performance.
     *
     * @param connection The active connection to the messaging service
     * @param queueName The name of the queue to verify or create
     * @param visibilityTimeout The duration in seconds that messages will be hidden
 after being received
     * @throws JMSException If there is an error accessing or creating the queue
     */
    public static void ensureQueueExists(SQSConnection connection, String queueName,
 Long visibilityTimeout) throws JMSException {
        AmazonSQSMessagingClientWrapper client =
 connection.getWrappedAmazonSQSClient();

      /* In most cases, you can do this with just a 'createQueue' call, but
 'getQueueUrl'
      (called by 'queueExists') is a faster operation for the common case where the
 queue
```

```
      already exists. Also, many users and roles have permission to call
'getQueueUrl'
      but don't have permission to call 'createQueue'.
      */
       if( !client.queueExists(queueName) ) {
           CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .attributes(Map.of(QueueAttributeName.VISIBILITY_TIMEOUT,
String.valueOf(visibilityTimeout)))
                   .build();
           client.createQueue( createQueueRequest );
       }
   }

   /**
    * This method sends a simple text message to a specified message queue. It
handles all necessary
    * setup for the message delivery process.
    *
    * @param session The active messaging session used to create and send the
message
    * @param queueName The name of the queue where the message will be sent
    */
   public static void sendTextMessage(Session session, String queueName) {
       // Rest of implementation...

       try {
           MessageProducer producer =
session.createProducer( session.createQueue( queueName) );
           Message message = session.createTextMessage("Hello world!");
           producer.send(message);
       } catch (JMSException e) {
           LOGGER.error( "Error receiving from SQS", e );
       }
   }

   /**
    * This method processes incoming messages and logs their content based on the
message type.
    * It supports text messages, binary data, and Java objects.
    *
    * @param message The message to be processed and logged
    * @throws JMSException If there is an error reading the message content
    */
```

```
    public static void handleMessage(Message message) throws JMSException {
        // Rest of implementation...
        LOGGER.info( "Got message {}", message.getJMSMessageID() );
        LOGGER.info( "Content: ");
        if(message instanceof TextMessage txtMessage) {
            LOGGER.info( "\t{}", txtMessage.getText() );
        } else if(message instanceof BytesMessage byteMessage){
            // Assume the length fits in an int - SQS only supports sizes up to 256k
so that
            // should be true
            byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
            byteMessage.readBytes(bytes);
            LOGGER.info( "\t{}", Base64.getEncoder().encodeToString( bytes ) );
        } else if( message instanceof ObjectMessage) {
            ObjectMessage objMessage = (ObjectMessage) message;
            LOGGER.info( "\t{}", objMessage.getObject() );
        }
    }

    /**
     * This method sets up automatic message processing for a specified queue. It
creates a listener
     * that will receive and handle incoming messages without blocking the main
program.
     *
     * @param session The active messaging session
     * @param queueName The name of the queue to monitor
     * @param connection The active connection to the messaging service
     */
    public static void receiveMessagesAsync(Session session, String queueName,
Connection connection) {
        // Rest of implementation...
        try {
            // Create a consumer for the queue.
            MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));
            // Provide an implementation of the MessageListener interface, which has
a single 'onMessage' method.
            // We use a lambda expression for the implementation.
            consumer.setMessageListener(message -> {
                try {
                    SqsJmsExampleUtils.handleMessage(message);
                    message.acknowledge();
                } catch (JMSException e) {
```

```
                    LOGGER.error("Error processing message: {}", e.getMessage());
                }
            });
            // Start receiving incoming messages.
            connection.start();
        } catch (JMSException e) {
            throw new RuntimeException(e);
        }
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * This method performs cleanup operations after message processing is complete.
It receives
     * any messages in the specified queue, removes the message queue and closes all
     * active connections to prevent resource leaks.
     *
     * @param queueName The name of the queue to be removed
     * @param visibilityTimeout The duration in seconds that messages are hidden
after being received
     * @throws JMSException If there is an error during the cleanup process
     */
    public static void cleanUpExample(String queueName, Long visibilityTimeout)
throws JMSException {
        LOGGER.info("Performing cleanup.");

        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        try (SQSConnection connection = connectionFactory.createConnection() ) {
            ensureQueueExists(connection, queueName, visibilityTimeout);
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

            receiveMessagesAsync(session, queueName, connection);
```

```
            SqsClient sqsClient =
connection.getWrappedAmazonSQSClient().getAmazonSQSClient();
            try {
                String queueUrl = sqsClient.getQueueUrl(b ->
b.queueName(queueName)).queueUrl();
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue deleted: {}", queueUrl);
            } catch (SdkException e) {
                LOGGER.error("Error during SQS operations: ", e);
            }
        }
        LOGGER.info("Clean up: Connection closed");
    }

    /**
     * This method creates a background task that sends multiple messages to a
specified queue
     * after waiting for a set time period. The task operates independently to
ensure efficient
     * message processing without interrupting other operations.
     *
     * @param queueName The name of the queue where messages will be sent
     * @param secondsToWait The number of seconds to wait before sending messages
     * @param numMessages The number of messages to send
     * @param visibilityTimeout The duration in seconds that messages remain hidden
after being received
     * @return A task that can be executed to send the messages
     */
    public static Runnable sendAMessageAsync(String queueName, Long secondsToWait,
Integer numMessages, Long visibilityTimeout) {
        return () -> {
            try {
                Thread.sleep(Duration.ofSeconds(secondsToWait).toMillis());
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException(e);
            }
            try {
                SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                        new ProviderConfiguration(),
                        SqsClient.create()
                );
                try (SQSConnection connection =
connectionFactory.createConnection()) {
```

```
                    ensureQueueExists(connection, queueName, visibilityTimeout);
                    Session session = connection.createSession(false,
 Session.CLIENT_ACKNOWLEDGE);
                    for (int i = 1; i <= numMessages; i++) {
                        MessageProducer producer =
 session.createProducer(session.createQueue(queueName));
                        producer.send(session.createTextMessage("Hello World " + i +
 "!"));
                    }
                }
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        };
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateQueue
  - DeleteQueue

Trabalhar com tags de fila

O exemplo de código a seguir mostra como realizar uma operação de marcação com o Amazon SQS.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O exemplo a seguir cria tags para uma fila, lista tags e remove uma tag.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ListQueueTagsResponse;
import software.amazon.awssdk.services.sqs.model.QueueDoesNotExistException;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.Map;
import java.util.UUID;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials. For more
 * information, see the <a href="https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/get-started.html">AWS
 * SDK for Java Developer Guide</a>.
 */
public class TagExamples {
    static final SqsClient sqsClient = SqsClient.create();
    static final String queueName = "TagExamples-queue-" +
 UUID.randomUUID().toString().replace("-", "").substring(0, 20);
    private static final Logger LOGGER = LoggerFactory.getLogger(TagExamples.class);

    public static void main(String[] args) {
        final String queueUrl;
        try {
            queueUrl = sqsClient.createQueue(b ->
 b.queueName(queueName)).queueUrl();
            LOGGER.info("Queue created. The URL is: {}", queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because queue was not created.");
            throw new RuntimeException(e);
        }
        try {
            addTags(queueUrl);
            listTags(queueUrl);
            removeTags(queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because of an error in a method.");
        } finally {
            try {
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue successfully deleted. Program ending.");
                sqsClient.close();
```

```
            } catch (RuntimeException e) {
                LOGGER.error("Program ending.");
            } finally {
                sqsClient.close();
            }
        }
    }

    /** This method demonstrates how to use a Java Map to a tag a aueue.
     * @param queueUrl The URL of the queue to tag.
     */
    public static void addTags(String queueUrl) {
        // Build a map of the tags.
        final Map<String, String> tagsToAdd = Map.of(
                "Team", "Development",
                "Priority", "Beta",
                "Accounting ID", "456def");

        try {
            // Add tags to the queue using a Consumer<TagQueueRequest.Builder>
parameter.
            sqsClient.tagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tags(tagsToAdd)
            );
        } catch (QueueDoesNotExistException e) {
            LOGGER.error("Queue does not exist: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }

    /** This method demonstrates how to view the tags for a queue.
     * @param queueUrl The URL of the queue whose tags you want to list.
     */
    public static void listTags(String queueUrl) {
        ListQueueTagsResponse response;
        try {
            // Call the listQueueTags method with a
Consumer<ListQueueTagsRequest.Builder> parameter that creates a
ListQueueTagsRequest.
            response = sqsClient.listQueueTags(b -> b
                    .queueUrl(queueUrl));
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
```

```
            throw new RuntimeException(e);
        }

        // Log the tags.
        response.tags()
                .forEach((k, v) ->
                        LOGGER.info("Key: {} -> Value: {}", k, v));
    }

    /**
     * This method demonstrates how to remove tags from a queue.
     * @param queueUrl The URL of the queue whose tags you want to remove.
     */
    public static void removeTags(String queueUrl) {
        try {
            // Call the untagQueue method with a Consumer<UntagQueueRequest.Builder>
 parameter.
            sqsClient.untagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tagKeys("Accounting ID") // Remove a single tag.
            );
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - ListQueueTags
  - TagQueue
  - UntagQueue

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
```

```
            context.getLogger().log("An error occurred");
            throw e;
        }

    }
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
```

```
            try {
                //process your message
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

Exemplos do Step Functions usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Step Functions.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Step Functions

O exemplo de código a seguir mostra como começar a usar o Step Functions.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Versão Java do Olá.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
```

```
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListStateMachines em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma atividade.
- Criar uma máquina de estado a partir de uma definição da Amazon States Language que contenha a atividade criada anteriormente como uma etapa.
- Executar a máquina de estado e responder à atividade com entrada do usuário.
- Obter o status e a saída finais após a conclusão da execução e, em seguida, limpar os recursos.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 * <p>
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 * <p>
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 * <p>
 * Also, set up your development environment, including your credentials.
 * <p>
 * For information, see this documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java code example performs the following tasks:
 * <p>
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <roleARN> <activityName> <stateMachineName>

            Where:
                roleName - The name of the IAM role to create for this state
 machine.
                activityName - The name of an activity to create.
                stateMachineName - The name of the state machine to create.
                jsonFile - The location of the chat_sfn_state_machine.json file. You
 can located it in resources/sample_files.
            """;
```

```
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String jsonFile = args[3];
        String polJSON = """
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "states.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
            """;


        Scanner sc = new Scanner(System.in);
        boolean action = false;

        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(regionGl)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Step Functions example scenario.");
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("1. Create an activity.");
        String activityArn = createActivity(sfnClient, activityName);
        System.out.println("The ARN of the activity is " + activityArn);
        System.out.println(DASHES);

        // Read the file using FileInputStream
        FileInputStream inputStream = new FileInputStream(jsonFile);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(inputStream, JsonNode.class);
        String jsonString = mapper.writeValueAsString(jsonNode);

        // Modify the Resource node.
        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode root = objectMapper.readTree(jsonString);
        ((ObjectNode) root.path("States").path("GetInput")).put("Resource",
 activityArn);

        // Convert the modified Java object back to a JSON string.
        String stateDefinition = objectMapper.writeValueAsString(root);
        System.out.println(stateDefinition);

        System.out.println(DASHES);
        System.out.println("2. Create a state machine.");
        String roleARN = createIAMRole(iam, roleName, polJSON);
        String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
 stateDefinition);
        System.out.println("The ARN of the state machine is " + stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe the state machine.");
        describeStateMachine(sfnClient, stateMachineArn);
        System.out.println("What should ChatSFN call you?");
        String userName = sc.nextLine();
        System.out.println("Hello " + userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        // The JSON to pass to the StartExecution call.
        String executionJson = "{ \"name\" : \"" + userName + "\" }";
        System.out.println(executionJson);
        System.out.println("4. Start execution of the state machine and interact
 with it.");
```

```
        String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
        System.out.println("The ARN of the state machine execution is " + runArn);
        List<String> myList;
        while (!action) {
            myList = getActivityTask(sfnClient, activityArn);
            System.out.println("ChatSFN: " + myList.get(1));
            System.out.println(userName + " please specify a value.");
            String myAction = sc.nextLine();
            if (myAction.compareTo("done") == 0)
                action = true;

            System.out.println("You have selected " + myAction);
            String taskJson = "{ \"action\" : \"" + myAction + "\" }";
            System.out.println(taskJson);
            sendTaskSuccess(sfnClient, myList.get(0), taskJson);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the execution.");
        describeExe(sfnClient, runArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Delete the activity.");
        deleteActivity(sfnClient, activityArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the state machines.");
        deleteMachine(sfnClient, stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Step Functions example scenario is complete.");
        System.out.println(DASHES);
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(polJSON)
```

```
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }

    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
 json) {
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
                .taskToken(token)
                .output(json)
                .build();

            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
        List<String> myList = new ArrayList<>();
        GetActivityTaskRequest getActivityTaskRequest =
 GetActivityTaskRequest.builder()
            .activityArn(actArn)
            .build();

        GetActivityTaskResponse response =
 sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }

    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
response.name());
            System.out.println("The status of the State machine is " +
response.status());
            System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }

    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status
is " + response.status());
```

```
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }

    public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                .input(jsonEx)
                .stateMachineArn(stateMachineArn)
                .name(uuidValue)
                .build();

            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
                .definition(json)
                .name(stateMachineName)
                .roleArn(roleARN)
                .type(StateMachineType.STANDARD)
                .build();

            CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
```

```
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
            CreateActivityRequest activityRequest = CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
 sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines

- SendTaskSuccess
- StartExecution
- StopExecution

Ações

`CreateActivity`

O código de exemplo a seguir mostra como usar `CreateActivity`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
            CreateActivityRequest activityRequest = CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte CreateActivity em Referência AWS SDK for Java 2.x da API.

`CreateStateMachine`

O código de exemplo a seguir mostra como usar `CreateStateMachine`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
       try {
           CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
               .definition(json)
               .name(stateMachineName)
               .roleArn(roleARN)
               .type(StateMachineType.STANDARD)
               .build();

           CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
           return response.stateMachineArn();

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte CreateStateMachine em Referência AWS SDK for Java 2.x da API.

`DeleteActivity`

O código de exemplo a seguir mostra como usar `DeleteActivity`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();

        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Para obter detalhes da API, consulte DeleteActivity em Referência AWS SDK for Java 2.x da API.

`DeleteStateMachine`

O código de exemplo a seguir mostra como usar `DeleteStateMachine`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
```

```
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status
is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }
```

- Para obter detalhes da API, consulte DeleteStateMachine em Referência AWS SDK for Java 2.x da API.

`DescribeExecution`

O código de exemplo a seguir mostra como usar `DescribeExecution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte DescribeExecution em Referência AWS SDK for Java 2.x da API.

`DescribeStateMachine`

O código de exemplo a seguir mostra como usar `DescribeStateMachine`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
       try {
           DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
               .stateMachineArn(stateMachineArn)
               .build();

           DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
           System.out.println("The name of the State machine is " +
response.name());
           System.out.println("The status of the State machine is " +
response.status());
           System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
           System.out.println("The role ARN value is " + response.roleArn());

       } catch (SfnException e) {
           System.err.println(e.getMessage());
       }
   }
```

- Para obter detalhes da API, consulte DescribeStateMachine em Referência AWS SDK for Java 2.x da API.

`GetActivityTask`

O código de exemplo a seguir mostra como usar `GetActivityTask`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
       List<String> myList = new ArrayList<>();
       GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
           .activityArn(actArn)
           .build();

       GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
       myList.add(response.taskToken());
       myList.add(response.input());
       return myList;
   }
```

- Para obter detalhes da API, consulte GetActivityTask em Referência AWS SDK for Java 2.x da API.

`ListActivities`

O código de exemplo a seguir mostra como usar `ListActivities`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
```

```
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
 ListActivitiesRequest.builder()
                    .maxResults(10)
                    .build();

            ListActivitiesResponse response =
 sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte ListActivities em Referência AWS SDK for Java 2.x da API.

ListExecutions

O código de exemplo a seguir mostra como usar `ListExecutions`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void getExeHistory(SfnClient sfnClient, String exeARN) {
       try {
           GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
                   .executionArn(exeARN)
                   .maxResults(10)
                   .build();

           GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
           List<HistoryEvent> events = historyResponse.events();
           for (HistoryEvent event : events) {
               System.out.println("The event type is " + event.type().toString());
           }

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ListExecutions em Referência AWS SDK for Java 2.x da API.

`ListStateMachines`

O código de exemplo a seguir mostra como usar `ListStateMachines`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
```

```
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte ListStateMachines em Referência AWS SDK for Java 2.x da API.

`SendTaskSuccess`

O código de exemplo a seguir mostra como usar `SendTaskSuccess`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
 json) {
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
                .taskToken(token)
                .output(json)
                .build();
```

```
            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Para obter detalhes da API, consulte SendTaskSuccess em Referência AWS SDK for Java 2.x da API.

StartExecution

O código de exemplo a seguir mostra como usar `StartExecution`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                .input(jsonEx)
                .stateMachineArn(stateMachineArn)
                .name(uuidValue)
                .build();

            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte StartExecution em Referência AWS SDK for Java 2.x da API.

Cenários

Usar Step Functions para invocar funções do Lambda

O exemplo de código a seguir mostra como criar uma máquina de AWS Step Functions estados que invoca AWS Lambda funções em sequência.

SDK para Java 2.x

Mostra como criar um fluxo de trabalho AWS sem servidor usando AWS Step Functions e o. AWS SDK for Java 2.x Cada etapa do fluxo de trabalho é implementada usando uma AWS Lambda função.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SES
- Funções de etapa

AWS STS exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version":"2012-10-17",
 * "Statement": [
 * {
```

```
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to assume
 (for example, arn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();
```

```
        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn, String
 roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " + exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte AssumeRole em Referência AWS SDK for Java 2.x da API.

Suporte exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with Suporte.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Suporte

O exemplo de código a seguir mostra como começar a usar o Suporte.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
```

```
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
                .region(region)
                .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
 DescribeServicesRequest.builder()
                    .language("en")
                    .build();
```

```
            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
                index++;
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeServices em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Obter e exibir os serviços disponíveis e os níveis de gravidade dos casos.
- Criar um caso de suporte usando um serviço, uma categoria e um nível de gravidade selecionados.
- Obter e exibir uma lista de casos em aberto para o dia atual.

- Adicionar um conjunto de anexos e uma comunicação ao novo caso.
- Descrever o novo anexo e a comunicação para o caso.
- Resolver o caso.
- Obtenha e exiba uma lista de casos resolvidos para o dia atual.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute várias Suporte operações.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
```

```
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <fileAttachment>Where:
                    fileAttachment - The file can be a simple saved .txt file to use
as an email attachment.\s
                """;

     //  if (args.length != 1) {
     //      System.out.println(usage);
     //      System.exit(1);
     //  }

        String fileAttachment = "C:\\AWS\\test.txt" ; //args[0];
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("***** Welcome to the AWS Support case example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Get and display available services.");
        List<String> sevCatList = displayServices(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get and display Support severity levels.");
        String sevLevel = displaySevLevels(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a support case using the selected service,
category, and severity level.");
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
        if (caseId.compareTo("") == 0) {
            System.out.println("A support case was not successfully created!");
            System.exit(1);
        } else
            System.out.println("Support case " + caseId + " was successfully
created!");
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
```

```
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(30)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .includeResolvedCases(true)
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                if (sinCase.status().compareTo("resolved") == 0)
                    System.out.println("The case status is " + sinCase.status());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();

            ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

        } catch (SupportException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeAttachment(SupportClient supportClient, String
attachId) {
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                    .attachmentId(attachId)
                    .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                    .caseId(caseId)
                    .maxResults(10)
                    .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
```

```
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
        try {
            AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                    .caseId(caseId)
                    .attachmentSetId(attachmentSetId)
                    .communicationBody("Please refer to attachment for details.")
                    .build();

            AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
            if (response.result())
                System.out.println("You have successfully added a communication to
an AWS Support case");
            else
                System.out.println("There was an error adding the communication to
an AWS Support case");

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
        try {
            File myFile = new File(fileAttachment);
            InputStream sourceStream = new FileInputStream(myFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            Attachment attachment = Attachment.builder()
```

```
                    .fileName(myFile.getName())
                    .data(sourceBytes)
                    .build();

            AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                    .attachments(attachment)
                    .build();

            AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
            return response.attachmentSetId();

        } catch (SupportException | FileNotFoundException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
        try {
            String serviceCode = sevCatList.get(0);
            String caseCat = sevCatList.get(1);
            CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                    .categoryCode(caseCat.toLowerCase())
                    .serviceCode(serviceCode.toLowerCase())
                    .severityCode(sevLevel.toLowerCase())
                    .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                    .subject("Test case, please ignore")
                    .language("en")
                    .issueType("technical")
                    .build();

            CreateCaseResponse response = supportClient.createCase(caseRequest);
            return response.caseId();

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static String displaySevLevels(SupportClient supportClient) {
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
```

```
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
```

```
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Ações

`AddAttachmentsToSet`

O código de exemplo a seguir mostra como usar `AddAttachmentsToSet`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
       try {
           File myFile = new File(fileAttachment);
           InputStream sourceStream = new FileInputStream(myFile);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           Attachment attachment = Attachment.builder()
                   .fileName(myFile.getName())
                   .data(sourceBytes)
                   .build();

           AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                   .attachments(attachment)
                   .build();

           AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
           return response.attachmentSetId();

       } catch (SupportException | FileNotFoundException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte AddAttachmentsToSet em Referência AWS SDK for Java 2.x da API.

`AddCommunicationToCase`

O código de exemplo a seguir mostra como usar `AddCommunicationToCase`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
       try {
           AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                   .caseId(caseId)
                   .attachmentSetId(attachmentSetId)
                   .communicationBody("Please refer to attachment for details.")
                   .build();

           AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
           if (response.result())
               System.out.println("You have successfully added a communication to
an AWS Support case");
           else
               System.out.println("There was an error adding the communication to
an AWS Support case");

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte AddCommunicationToCase em Referência AWS SDK for Java 2.x da API.

`CreateCase`

O código de exemplo a seguir mostra como usar `CreateCase`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
       try {
           String serviceCode = sevCatList.get(0);
           String caseCat = sevCatList.get(1);
           CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                   .categoryCode(caseCat.toLowerCase())
                   .serviceCode(serviceCode.toLowerCase())
                   .severityCode(sevLevel.toLowerCase())
                   .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                   .subject("Test case, please ignore")
                   .language("en")
                   .issueType("technical")
                   .build();

           CreateCaseResponse response = supportClient.createCase(caseRequest);
           return response.caseId();

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte CreateCase em Referência AWS SDK for Java 2.x da API.

`DescribeAttachment`

O código de exemplo a seguir mostra como usar `DescribeAttachment`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void describeAttachment(SupportClient supportClient, String
attachId) {
       try {
           DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                   .attachmentId(attachId)
                   .build();

           DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
           System.out.println("The name of the file is " +
response.attachment().fileName());

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeAttachment em Referência AWS SDK for Java 2.x da API.

`DescribeCases`

O código de exemplo a seguir mostra como usar `DescribeCases`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void getOpenCase(SupportClient supportClient) {
       try {
           // Specify the start and end time.
           Instant now = Instant.now();
           java.time.LocalDate.now();
           Instant yesterday = now.minus(1, ChronoUnit.DAYS);

           DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                   .maxResults(20)
                   .afterTime(yesterday.toString())
                   .beforeTime(now.toString())
                   .build();

           DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
           List<CaseDetails> cases = response.cases();
           for (CaseDetails sinCase : cases) {
               System.out.println("The case status is " + sinCase.status());
               System.out.println("The case Id is " + sinCase.caseId());
               System.out.println("The case subject is " + sinCase.subject());
           }

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte DescribeCases em Referência AWS SDK for Java 2.x da API.

`DescribeCommunications`

O código de exemplo a seguir mostra como usar `DescribeCommunications`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static String listCommunications(SupportClient supportClient, String
caseId) {
       try {
           String attachId = null;
           DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                   .caseId(caseId)
                   .maxResults(10)
                   .build();

           DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
           List<Communication> communications = response.communications();
           for (Communication comm : communications) {
               System.out.println("the body is: " + comm.body());

               // Get the attachment id value.
               List<AttachmentDetails> attachments = comm.attachmentSet();
               for (AttachmentDetails detail : attachments) {
                   attachId = detail.attachmentId();
               }
           }
           return attachId;

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Para obter detalhes da API, consulte DescribeCommunications em Referência AWS SDK for Java 2.x da API.

`DescribeServices`

O código de exemplo a seguir mostra como usar `DescribeServices`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();
```

```
                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
```

- Para obter detalhes da API, consulte DescribeServices em Referência AWS SDK for Java 2.x da API.

`DescribeSeverityLevels`

O código de exemplo a seguir mostra como usar `DescribeSeverityLevels`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public static String displaySevLevels(SupportClient supportClient) {
        try {
```

```
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- Para obter detalhes da API, consulte DescribeSeverityLevels em Referência AWS SDK for Java 2.x da API.

`ResolveCase`

O código de exemplo a seguir mostra como usar `ResolveCase`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
       try {
           ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                   .caseId(caseId)
                   .build();

           ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
           System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Para obter detalhes da API, consulte ResolveCase em Referência AWS SDK for Java 2.x da API.

Exemplos do Systems Manager usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Systems Manager.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Systems Manager

O exemplo de código a seguir mostra como começar a usar o Systems Manager.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <awsAccount>

                Where:
                    awsAccount - Your AWS Account number.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();
```

```
        listDocuments(ssmClient, awsAccount);
    }

    /*
    This code automatically fetches the next set of results using the `nextToken`
 and
    stops once the desired maxResults (20 in this case) have been reached.
    */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                    )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
 System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- Para obter detalhes da API, consulte ListDocuments em Referência AWS SDK for Java 2.x da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma janela de manutenção.
- Modificar a programação da janela de manutenção.
- Criar um documento.
- Enviar um comando para uma instância do EC2 especificada.
- Crie um OpsItem.
- Atualize e resolva OpsItem o.
- Exclua a janela de manutenção OpsItem e o documento.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Map;
import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String ROLES_STACK = "SsmStack3`1";

    public static void main(String[] args) {
        String usage = """
            Usage:
              <title> <source> <category> <severity>

            Where:
                title - The title of the parameter (default is Disk Space Alert).
                source - The source of the parameter (default is EC2).
                category - The category of the parameter. Valid values are
 'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
 Performance).
                severity - The severity of the parameter. Severity should be a
 number from 1 to 4 (default is 2).
        """;
```

```
        Scanner scanner = new Scanner(System.in);
        SSMActions actions = new SSMActions();
        String documentName;
        String windowName;

        System.out.println("Use AWS CloudFormation to create the EC2 instance that
is required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String instanceId = stackOutputs.get("InstanceId");
        System.out.println("The Instance ID: " + instanceId +" was created.");
        String title = "Disk Space Alert" ;
        String source = "EC2" ;
        String category = "Availability" ;
        String severity = "2" ;

        System.out.println(DASHES);
        System.out.println("""
                Welcome to the AWS Systems Manager SDK Basics scenario.
                This Java program demonstrates how to interact with AWS Systems
Manager using the AWS SDK for Java (v2).
                AWS Systems Manager is the operations hub for your AWS applications
and resources and a secure end-to-end management solution.
                The program's primary functionalities include creating a maintenance
window, creating a document, sending a command to a document,
                listing documents, listing commands, creating an OpsItem, modifying
an OpsItem, and deleting AWS SSM resources.
                Upon completion of the program, all AWS resources are cleaned up.
                Let's get started...

                """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("1. Create an SSM maintenance window.");
        System.out.println("Please enter the maintenance window name (default is
ssm-maintenance-window):");
        String win = scanner.nextLine();
        windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
        String winId = null;
        try {
            winId = actions.createMaintenanceWindow(windowName);
```

```
            waitForInputToContinue(scanner);
            System.out.println("The maintenance window ID is: " + winId);
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM maintenance window already exists.
Retrieving existing window ID...");
            String existingWinId = actions.createMaintenanceWindow(windowName);
            System.out.println("Existing window ID: " + existingWinId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("2. Modify the maintenance window by changing the
schedule");
        waitForInputToContinue(scanner);
        try {
            actions.updateSSMMaintenanceWindow(winId, windowName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM maintenance window was successfully
updated");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("3. Create an SSM document that defines the actions that
Systems Manager performs on your managed nodes.");
        System.out.println("Please enter the document name (default is
ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        try {
            actions.createSSMDoc(documentName);
            waitForInputToContinue(scanner);
```

```
            System.out.println("The SSM document was successfully created");
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM document already exists. Moving on");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("4. Now we are going to run a command on an EC2
instance");
        waitForInputToContinue(scanner);
        String commandId="";
        try {
            commandId = actions.sendSSMCommand(documentName, instanceId);
            waitForInputToContinue(scanner);
            System.out.println("The command was successfully sent. Command ID: " +
commandId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
        } catch (InterruptedException e) {
            System.err.println("Thread was interrupted: " + e.getMessage());
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("5. Lets get the time when the specific command was sent
to the specific managed node");
        waitForInputToContinue(scanner);
        try {
            actions.displayCommands(commandId);
            System.out.println("The command invocations were successfully
displayed.");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
```

```
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
             6. Now we will create an SSM OpsItem.
             A SSM OpsItem is a feature provided by Amazon's Systems Manager (SSM)
service.
             It is a type of operational data item that allows you to manage and
track various operational issues,
             events, or tasks within your AWS environment.

             You can create OpsItems to track and manage operational issues as they
arise.
             For example, you could create an OpsItem whenever your application
detects a critical error
             or an anomaly in your infrastructure.
            """);

        waitForInputToContinue(scanner);
        String opsItemId;
        try {
            opsItemId = actions.createSSMOpsItem(title, source, category, severity);
            System.out.println(opsItemId + " was created");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Now we will update the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        String description = "An update to "+opsItemId ;
        try {
            actions.updateOpsItem(opsItemId, title, description);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
```

```
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("8. Now we will get the status of the SSM OpsItem
"+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.describeOpsItems(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("9. Now we will resolve the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.resolveOpsItem(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("10. Would you like to delete the AWS Systems Manager
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            waitForInputToContinue(scanner);
            try {
                actions.deleteMaintenanceWindow(winId);
                actions.deleteDoc(documentName);
            } catch (SsmException e) {
                System.err.println("SSM error: " + e.getMessage());
```

```
                return;
            } catch (RuntimeException e) {
                System.err.println("Unexpected error: " + e.getMessage());
                return;
            }
        } else {
            System.out.println("The AWS Systems Manager resources will not be
 deleted");
        }
        System.out.println(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        System.out.println("This concludes the AWS Systems Manager SDK Basics
 scenario.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de wrapper para métodos do SDK do Systems Manager.

```
public class SSMActions {

    private static SsmAsyncClient ssmAsyncClient;

    private static SsmAsyncClient getAsyncClient() {
        if (ssmAsyncClient == null) {
```

```
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryPolicy(RetryPolicy.builder()
                            .numRetries(3)
                            .build())
                    .build();

            ssmAsyncClient = SsmAsyncClient.builder()
                    .region(Region.US_WEST_2)
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ssmAsyncClient;
    }

    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
```

```
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteMaintenanceWindow(String winId) {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteMaintenanceWindow(windowRequest)
                    .thenAccept(response -> {
                        System.out.println("The maintenance window was successfully
deleted.");
                    })
```

```
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
```

```
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
```

```
                    .thenAccept(itemsResponse -> {
                        List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                        for (OpsItemSummary item : items) {
                            System.out.println("The item title is " + item.title() +
" and the status is " + item.status().toString());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Updates the AWS SSM OpsItem asynchronously.
     *
     * @param opsItemId The ID of the OpsItem to update.
     * @param title The new title of the OpsItem.
     * @param description The new description of the OpsItem.
     * <p>
     * This method initiates an asynchronous request to update an SSM OpsItem.
     * If the request is successful, it completes without returning a value.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateOpsItem(String opsItemId, String title, String description) {
        Map<String, OpsItemDataValue> operationalData = new HashMap<>();
```

```
        operationalData.put("key1",
 OpsItemDataValue.builder().value("value1").build());
        operationalData.put("key2",
 OpsItemDataValue.builder().value("value2").build());

        CompletableFuture<Void> future = getOpsItem(opsItemId).thenCompose(opsItem -
> {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                    .opsItemId(opsItemId)
                    .title(title)
                    .operationalData(operationalData)
                    .status(opsItem.statusAsString())
                    .description(description)
                    .build();

            return getAsyncClient().updateOpsItem(request).thenAccept(response -> {
                System.out.println(opsItemId + " updated successfully.");
            }).exceptionally(ex -> {
                throw new CompletionException(ex);
            });
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
 ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }


    private static CompletableFuture<OpsItem> getOpsItem(String opsItemId) {
        GetOpsItemRequest request =
 GetOpsItemRequest.builder().opsItemId(opsItemId).build();
```

```
        return
getAsyncClient().getOpsItem(request).thenApply(GetOpsItemResponse::opsItem);
    }

    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }

    /**
```

```
    * Displays the date and time when the specific command was invoked.
    *
    * @param commandId The ID of the command to describe.
    * <p>
    * This method initiates an asynchronous request to list command invocations and
prints the date and time of each command invocation.
    * If an exception occurs, it handles the error appropriately.
    */
   public void displayCommands(String commandId) {
       ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
               .commandId(commandId)
               .build();

       CompletableFuture<ListCommandInvocationsResponse> future =
getAsyncClient().listCommandInvocations(commandInvocationsRequest);
       future.thenAccept(response -> {
           List<CommandInvocation> commandList = response.commandInvocations();
           DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
           for (CommandInvocation invocation : commandList) {
               System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
           }
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
           if (cause instanceof SsmException) {
               throw (SsmException) cause;
           } else {
               throw new RuntimeException(cause);
           }
       }).join();
   }

   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
```

```
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       throw new RuntimeException(e);
                   }
               }
           }
       });

       documentActiveFuture.join();

       // Create the SendCommandRequest.
       SendCommandRequest commandRequest = SendCommandRequest.builder()
               .documentName(documentName)
               .instanceIds(instanceId)
               .build();

       // Send the command.
       CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
```

```
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
```

```
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }

    /**
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
```

```
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }

    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();
```

```
        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }

    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
window.
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
                .name(winName)
                .description("This is my maintenance window")
                .allowUnassociatedTargets(true)
                .duration(2)
                .cutoff(1)
                .schedule("cron(0 10 ? * MON-FRI *)")
                .build();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
```

```
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
```

```
                    Throwable describeCause = (describeEx instanceof
 CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
 windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Ações

`CreateDocument`

O código de exemplo a seguir mostra como usar `CreateDocument`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
```

```
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

- Para obter detalhes da API, consulte CreateDocument em Referência AWS SDK for Java 2.x da API.

`CreateMaintenanceWindow`

O código de exemplo a seguir mostra como usar `CreateMaintenanceWindow`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
window.
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
                .name(winName)
                .description("This is my maintenance window")
                .allowUnassociatedTargets(true)
```

```
                .duration(2)
                .cutoff(1)
                .schedule("cron(0 10 ? * MON-FRI *)")
                .build();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
```

```
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
```

- Para obter detalhes da API, consulte CreateMaintenanceWindow em Referência AWS SDK for Java 2.x da API.

`CreateOpsItem`

O código de exemplo a seguir mostra como usar `CreateOpsItem`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
```

```
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }
```

- Para obter detalhes da API, consulte CreateOpsItem em Referência AWS SDK for Java 2.x da API.

`DeleteDocument`

O código de exemplo a seguir mostra como usar `DeleteDocument`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });
```

```
        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Para obter detalhes da API, consulte DeleteDocument em Referência AWS SDK for Java 2.x da API.

`DeleteMaintenanceWindow`

O código de exemplo a seguir mostra como usar `DeleteMaintenanceWindow`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteMaintenanceWindow(String winId) {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteMaintenanceWindow(windowRequest)
```

```
                    .thenAccept(response -> {
                        System.out.println("The maintenance window was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Para obter detalhes da API, consulte DeleteMaintenanceWindow em Referência AWS SDK for Java 2.x da API.

`DescribeOpsItems`

O código de exemplo a seguir mostra como usar `DescribeOpsItems`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
                    .thenAccept(itemsResponse -> {
                        List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                        for (OpsItemSummary item : items) {
                            System.out.println("The item title is " + item.title() +
" and the status is " + item.status().toString());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
```

```
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }
```

- Para obter detalhes da API, consulte DescribeOpsItems em Referência AWS SDK for Java 2.x da API.

`DescribeParameters`

O código de exemplo a seguir mostra como usar `DescribeParameters`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class GetParameter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                    .name(paraName)
                    .build();

            GetParameterResponse parameterResponse =
 ssmClient.getParameter(parameterRequest);
            System.out.println("The parameter value is " +
 parameterResponse.parameter().value());

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeParameters em Referência AWS SDK for Java 2.x da API.

PutParameter

O código de exemplo a seguir mostra como usar `PutParameter`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                    paraValue - The value of the parameter.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String paraName = args[0];
        String paraValue = args[1];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        putParaValue(ssmClient, paraName, paraValue);
        ssmClient.close();
    }

    public static void putParaValue(SsmClient ssmClient, String paraName, String
 value) {
        try {
            PutParameterRequest parameterRequest = PutParameterRequest.builder()
                    .name(paraName)
                    .type(ParameterType.STRING)
                    .value(value)
                    .build();

            ssmClient.putParameter(parameterRequest);
            System.out.println("The parameter was successfully added.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte PutParameter em Referência AWS SDK for Java 2.x da API.

SendCommand

O código de exemplo a seguir mostra como usar `SendCommand`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
```

```
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        documentActiveFuture.join();

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
```

```
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }
```

- Para obter detalhes da API, consulte SendCommand em Referência AWS SDK for Java 2.x da API.

UpdateMaintenanceWindow

O código de exemplo a seguir mostra como usar `UpdateMaintenanceWindow`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Updates an SSM maintenance window asynchronously.
    *
    * @param id The ID of the maintenance window to update.
    * @param name The new name for the maintenance window.
    * <p>
    * This method initiates an asynchronous request to update an SSM maintenance
window.
    * If the request is successful, it prints a success message.
    * If an exception occurs, it handles the error appropriately.
    */
   public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
       UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
               .windowId(id)
               .allowUnassociatedTargets(true)
               .duration(24)
               .enabled(true)
               .name(name)
               .schedule("cron(0 0 ? * MON *)")
               .build();

       CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
       future.whenComplete((response, ex) -> {
           if (response != null) {
               System.out.println("The SSM maintenance window was successfully
updated");
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof SsmException) {
                   throw new CompletionException(cause);
```

```
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }
```

- Para obter detalhes da API, consulte UpdateMaintenanceWindow em Referência AWS SDK for Java 2.x da API.

`UpdateOpsItem`

O código de exemplo a seguir mostra como usar `UpdateOpsItem`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
```

```
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Para obter detalhes da API, consulte UpdateOpsItem em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Textract usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Textract.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AnalyzeDocument`

O código de exemplo a seguir mostra como usar `AnalyzeDocument`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
                    sourceDoc - The path where the document is located (must be an
 image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        analyzeDoc(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            List<FeatureType> featureTypes = new ArrayList<FeatureType>();
            featureTypes.add(FeatureType.FORMS);
            featureTypes.add(FeatureType.TABLES);
```

```
            AnalyzeDocumentRequest analyzeDocumentRequest =
 AnalyzeDocumentRequest.builder()
                    .featureTypes(featureTypes)
                    .document(myDoc)
                    .build();

            AnalyzeDocumentResponse analyzeDocument =
 textractClient.analyzeDocument(analyzeDocumentRequest);
            List<Block> docInfo = analyzeDocument.blocks();
            Iterator<Block> blockIterator = docInfo.iterator();

            while (blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is " +
 block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte AnalyzeDocument em Referência AWS SDK for Java 2.x da API.

`DetectDocumentText`

O código de exemplo a seguir mostra como usar `DetectDocumentText`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Detecte texto de um documento de entrada.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
                    sourceDoc - The path where the document is located (must be an
 image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
```

```
                .region(region)
                .build();

        detectDocText(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void detectDocText(TextractClient textractClient, String
 sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes.
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the Detect operation.
            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            List<Block> docInfo = textResponse.blocks();
            for (Block block : docInfo) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte texto de um documento localizado em um bucket do Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains the
 document.\s

                    docName - The document name (must be an image, i.e., book.png).
\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }

    public static void detectDocTextS3(TextractClient textractClient, String
bucketName, String docName) {
        try {
            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            // Create a Document object and reference the s3Object instance.
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            DetectDocumentTextResponse textResponse =
textractClient.detectDocumentText(detectDocumentTextRequest);
            for (Block block : textResponse.blocks()) {
                System.out.println("The block type is " +
block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
documentMetadata.pages());

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte DetectDocumentText em Referência AWS SDK for Java 2.x da API.

StartDocumentAnalysis

O código de exemplo a seguir mostra como usar `StartDocumentAnalysis`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains the
 document.\s
                    docName - The document name (must be an image, for example,
 book.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
        String status = getJobResults(textractClient, jobId);
        System.out.println("The job status is " + status);
        textractClient.close();
    }

    public static String startDocAnalysisS3(TextractClient textractClient, String
 bucketName, String docName) {
        try {
            List<FeatureType> myList = new ArrayList<>();
            myList.add(FeatureType.TABLES);
            myList.add(FeatureType.FORMS);

            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();
```

```
            DocumentLocation location = DocumentLocation.builder()
                    .s3Object(s3Object)
                    .build();

            StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                    .documentLocation(location)
                    .featureTypes(myList)
                    .build();

            StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

            // Get the job ID
            String jobId = response.jobId();
            return jobId;

        } catch (TextractException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    private static String getJobResults(TextractClient textractClient, String jobId)
{
        boolean finished = false;
        int index = 0;
        String status = "";

        try {
            while (!finished) {
                GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                        .jobId(jobId)
                        .maxResults(1000)
                        .build();

                GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
                status = response.jobStatus().toString();

                if (status.compareTo("SUCCEEDED") == 0)
```

```
                    finished = true;
                else {
                    System.out.println(index + " status is: " + status);
                    Thread.sleep(1000);
                }
                index++;
            }

            return status;

        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Para obter detalhes da API, consulte StartDocumentAnalysis em Referência AWS SDK for Java 2.x da API.

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.

- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon Transcribe usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Transcribe.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`ListTranscriptionJobs`

O código de exemplo a seguir mostra como usar `ListTranscriptionJobs`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

            listTranscriptionJobs(transcribeClient);
        }

        public static void listTranscriptionJobs(TranscribeClient transcribeClient)
 {
            ListTranscriptionJobsRequest listJobsRequest =
 ListTranscriptionJobsRequest.builder()
                .build();


 transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
                .flatMap(response -> response.transcriptionJobSummaries().stream())
                .forEach(jobSummary -> {
                    System.out.println("Job Name: " +
 jobSummary.transcriptionJobName());
                    System.out.println("Job Status: " +
 jobSummary.transcriptionJobStatus());
                    System.out.println("Output Location: " +
 jobSummary.outputLocationType());
                    // Add more information as needed

                    // Retrieve additional details for the job if necessary
                    GetTranscriptionJobResponse jobDetails =
 transcribeClient.getTranscriptionJob(
                        GetTranscriptionJobRequest.builder()
                            .transcriptionJobName(jobSummary.transcriptionJobName())
                            .build());
```

```
                    // Display additional details
                    System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
                    System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
                    // Add more details as needed

                    System.out.println("--------------");
                });
        }
    }
```

- Para obter detalhes da API, consulte ListTranscriptionJobs em Referência AWS SDK for Java 2.x da API.

Cenários

Transcrever áudio e obter dados do trabalho

O exemplo de código a seguir mostra como:

- Iniciar um trabalho de transcrição com o Amazon Transcribe.
- Aguardar a conclusão do trabalho.
- Obter o URI em que a transcrição está armazenada.

Para obter mais informações, consulte Começar a usar o Amazon Transcribe.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Transcreve um arquivo PCM.

```
/**
 * To run this AWS code example, ensure that you have set up your development
```

```
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
```

```
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }
```

```
    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
```

```
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
```

```
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

Transcreve o streaming de áudio do microfone do computador.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }
```

```
        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
```

```
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
```

```
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
```

```
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x .
  - GetTranscriptionJob
  - StartTranscriptionJob

Exemplos do Amazon Transcribe Streaming usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x Amazon Transcribe Streaming.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`StartMedicalStreamTranscription`

O código de exemplo a seguir mostra como usar `StartMedicalStreamTranscription`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/*
To run this AWS code example, ensure that you have set up your development
 environment, including your AWS credentials.

 For information, see this documentation topic:

 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

This code demonstrates the process of starting a medical transcription job using the
 AWS Transcribe
Streaming service, including setting up the audio input stream, configuring the
 transcription request,
and handling the transcription response.
 */

public class TranscribeMedicalStreamingDemoApp {
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws ExecutionException, InterruptedException, LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .build();

        CompletableFuture<Void> result =
 client.startMedicalStreamTranscription(getMedicalRequest(16_000),
            new AudioStreamPublisher(getStreamFromMic()),
            getMedicalResponseHandler());
```

```
        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            throw new LineUnavailableException("The audio system microphone line is
not supported.");
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartMedicalStreamTranscriptionRequest getMedicalRequest(Integer
mediaSampleRateHertz) {
        return StartMedicalStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString()) // For medical
transcription, EN_US is typically used.
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .specialty(Specialty.PRIMARYCARE) // Specify the medical specialty.
            .type(Type.CONVERSATION) // Set the type as CONVERSATION or DICTATION.
            .build();
    }

    private static StartMedicalStreamTranscriptionResponseHandler
getMedicalResponseHandler() {
```

```
        return StartMedicalStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records streamed successfully ===");
            })
            .subscriber(event -> {
                List<MedicalResult> results = ((MedicalTranscriptEvent)
 event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
```

```
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);
            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }
```

```
        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                .audioChunk(SdkBytes.fromByteBuffer(bb))
                .build();
        }
    }
}
```

- Para obter detalhes da API, consulte StartMedicalStreamTranscription em Referência AWS SDK for Java 2.x da API.

`StartStreamTranscription`

O código de exemplo a seguir mostra como usar `StartStreamTranscription`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }
```

```
        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
```

```
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
```

```
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
```

```
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Para obter detalhes da API, consulte StartStreamTranscription em Referência AWS SDK for Java 2.x da API.

Cenários

Transcrever um arquivo de áudio

O exemplo de código a seguir mostra como gerar uma transcrição de um arquivo de áudio de origem usando o streaming do Amazon Transcribe.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }
```

```
    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
```

```
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
```

```
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Para obter detalhes da API, consulte StartStreamTranscription em Referência AWS SDK for Java 2.x da API.

Transcrever o áudio de um microfone

O exemplo de código a seguir mostra como gerar uma transcrição por um microfone usando o streaming do Amazon Transcribe.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {
```

```
        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
```

```
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);
```

```
        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];
```

```
            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Para obter detalhes da API, consulte StartStreamTranscription em Referência AWS SDK for Java 2.x da API.

Exemplos do Amazon Translate usando o SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x com o Amazon Translate.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar um chatbot Amazon Lex

O exemplo de código a seguir mostra como criar um chatbot para engajar os visitantes do seu site.

SDK para Java 2.x

Mostra como usar a API do Amazon Lex para criar um Chatbot em uma aplicação da web para envolver os visitantes do seu site.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Criação de uma aplicação do Amazon SNS

O exemplo de código a seguir mostra como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

SDK para Java 2.x

Mostra como usar a API Java do Amazon Simple Notification Service para criar uma aplicação Web com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo que usa a API Java Async, consulte o exemplo completo em. GitHub

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Java 2.x

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

X-Ray exemplos de uso do SDK para Java 2.x

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Java 2.x with X-Ray.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateGroup`

O código de exemplo a seguir mostra como usar `CreateGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.CreateGroupRequest;
import software.amazon.awssdk.services.xray.model.CreateGroupResponse;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
```

```
                    groupName - The name of the group to create\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        createNewGroup(xRayClient, groupName);
    }

    public static void createNewGroup(XRayClient xRayClient, String groupName) {
        try {
            CreateGroupRequest groupRequest = CreateGroupRequest.builder()
                    .filterExpression("fault = true AND http.url CONTAINS \"example/
game\" AND responsetime >= 5")
                    .groupName(groupName)
                    .build();

            CreateGroupResponse groupResponse =
 xRayClient.createGroup(groupRequest);
            System.out.println("The Group ARN is " +
 groupResponse.group().groupARN());

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateGroup em Referência AWS SDK for Java 2.x da API.

`CreateSamplingRule`

O código de exemplo a seguir mostra como usar `CreateSamplingRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.CreateSamplingRuleRequest;
import software.amazon.awssdk.services.xray.model.SamplingRule;
import software.amazon.awssdk.services.xray.model.XRayException;
import software.amazon.awssdk.services.xray.model.CreateSamplingRuleResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSamplingRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <ruleName>

                Where:
                   ruleName - The name of the rule\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String ruleName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        createRule(xRayClient, ruleName);
    }

    public static void createRule(XRayClient xRayClient, String ruleName) {
        try {

            SamplingRule rule = SamplingRule.builder()
                    .ruleName(ruleName)
                    .priority(1)
                    .httpMethod("*")
                    .serviceType("*")
                    .serviceName("*")
                    .urlPath("*")
                    .version(1)
                    .host("*")
                    .resourceARN("*")
                    .build();

            CreateSamplingRuleRequest ruleRequest =
 CreateSamplingRuleRequest.builder()
                    .samplingRule(rule)
                    .build();

            CreateSamplingRuleResponse ruleResponse =
 xRayClient.createSamplingRule(ruleRequest);
            System.out.println(
                    "The ARN of the new rule is " +
 ruleResponse.samplingRuleRecord().samplingRule().ruleARN());

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte CreateSamplingRule em Referência AWS SDK for Java 2.x da API.

DeleteGroup

O código de exemplo a seguir mostra como usar `DeleteGroup`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.DeleteGroupRequest;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of the group to delete\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        deleteSpecificGroup(xRayClient, groupName);
    }

    public static void deleteSpecificGroup(XRayClient xRayClient, String groupName)
 {
        try {
            DeleteGroupRequest groupRequest = DeleteGroupRequest.builder()
                    .groupName(groupName)
                    .build();

            xRayClient.deleteGroup(groupRequest);
            System.out.println(groupName + " was deleted!");

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteGroup em Referência AWS SDK for Java 2.x da API.

`DeleteSamplingRule`

O código de exemplo a seguir mostra como usar `DeleteSamplingRule`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.DeleteSamplingRuleRequest;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSamplingRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <ruleName>

                Where:
                   ruleName - The name of the rule to delete\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();
```

```
        deleteRule(xRayClient, ruleName);
    }

    public static void deleteRule(XRayClient xRayClient, String ruleName) {
        try {
            DeleteSamplingRuleRequest ruleRequest =
 DeleteSamplingRuleRequest.builder()
                    .ruleName(ruleName)
                    .build();

            xRayClient.deleteSamplingRule(ruleRequest);
        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte DeleteSamplingRule em Referência AWS SDK for Java 2.x da API.

`GetGroups`

O código de exemplo a seguir mostra como usar `GetGroups`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetGroupsResponse;
import software.amazon.awssdk.services.xray.model.GroupSummary;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetGroups {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        getAllGroups(xRayClient);
    }

    public static void getAllGroups(XRayClient xRayClient) {
        try {
            GetGroupsResponse groupsResponse = xRayClient.getGroups();
            List<GroupSummary> groups = groupsResponse.groups();
            for (GroupSummary group : groups) {
                System.out.println("The AWS XRay group name is " +
 group.groupName());
            }

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetGroups em Referência AWS SDK for Java 2.x da API.

GetSamplingRules

O código de exemplo a seguir mostra como usar `GetSamplingRules`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetSamplingRulesResponse;
import software.amazon.awssdk.services.xray.model.SamplingRuleRecord;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSamplingRules {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        getRules(xRayClient);
    }

    public static void getRules(XRayClient xRayClient) {
        try {
            GetSamplingRulesResponse response = xRayClient.getSamplingRules(r ->
 r.build());
            List<SamplingRuleRecord> records = response.samplingRuleRecords();

            for (SamplingRuleRecord record : records) {
```

```
                System.out.println("The rule name is: " +
 record.samplingRule().ruleName());
                System.out.println("The related service is: " +
 record.samplingRule().serviceName());
            }
        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetSamplingRules em Referência AWS SDK for Java 2.x da API.

`GetServiceGraph`

O código de exemplo a seguir mostra como usar `GetServiceGraph`.

SDK para Java 2.x

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetServiceGraphRequest;
import software.amazon.awssdk.services.xray.model.GetServiceGraphResponse;
import software.amazon.awssdk.services.xray.model.Service;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.time.LocalDateTime;
import java.time.Instant;
import java.time.ZoneId;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetServiceGraph {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of a group based on which you want to
 generate a graph.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();
        getGraph(xRayClient, groupName);
    }

    public static void getGraph(XRayClient xRayClient, String groupName) {
        try {
            // The Instant values have to be 6 hours apart.
            LocalDateTime localDateTime =
 LocalDateTime.parse("2022-03-09T06:00:00");
            Instant start = localDateTime.atZone(ZoneId.of("America/
New_York")).toInstant();

            LocalDateTime localDateTime2 =
 LocalDateTime.parse("2022-03-09T12:00:00");
            Instant end = localDateTime2.atZone(ZoneId.of("America/
New_York")).toInstant();
```

```
            GetServiceGraphRequest getServiceGraphRequest =
 GetServiceGraphRequest.builder()
                    .groupName(groupName)
                    .startTime(start)
                    .endTime(end)
                    .build();

            GetServiceGraphResponse graphResponse =
 xRayClient.getServiceGraph(getServiceGraphRequest);
            List<Service> services = graphResponse.services();
            for (Service service : services) {
                System.out.println("The name of the service is  " + service.name());
            }

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obter detalhes da API, consulte GetServiceGraph em Referência AWS SDK for Java 2.x da API.

Exemplos de código para SDK para Kotlin

Os exemplos de código a seguir mostram como usar o AWS SDK para Kotlin com. AWS

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para Kotlin — Saiba mais sobre como usar o Kotlin com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos do API Gateway usando o SDK para Kotlin
- Exemplos de Aurora usando o SDK para Kotlin
- Exemplos do Auto Scaling usando o SDK para Kotlin
- Exemplos do Amazon Bedrock usando o SDK para Kotlin
- Exemplos do Amazon Bedrock Runtime usando o SDK para Kotlin
- CloudWatch exemplos de uso do SDK para Kotlin
- CloudWatch Exemplos de registros usando o SDK para Kotlin
- Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Kotlin
- Exemplos do Amazon Comprehend usando o SDK para Kotlin
- Exemplos do DynamoDB usando o SDK para Kotlin
- Exemplos do Amazon EC2 usando o SDK para Kotlin
- Exemplos do Amazon ECR usando o SDK para Kotlin
- OpenSearch Exemplos de serviços usando o SDK para Kotlin
- EventBridge exemplos de uso do SDK para Kotlin
- AWS Glue exemplos de uso do SDK para Kotlin
- Exemplos de IAM usando o SDK para Kotlin
- AWS IoT exemplos de uso do SDK para Kotlin
- AWS IoT data exemplos de uso do SDK para Kotlin
- Exemplos do Amazon Keyspaces usando o SDK para Kotlin
- AWS KMS exemplos de uso do SDK para Kotlin
- Exemplos de Lambda usando o SDK para Kotlin

- Exemplos do Amazon Location usando o SDK para Kotlin
- MediaConvert exemplos de uso do SDK para Kotlin
- Exemplos do Amazon Pinpoint usando o SDK para Kotlin
- Exemplos do Amazon RDS usando o SDK para Kotlin
- Exemplos do Amazon RDS Data Service usando o SDK para Kotlin
- Exemplos do Amazon Redshift usando o SDK para Kotlin
- Exemplos do Amazon Rekognition usando o SDK para Kotlin
- Exemplos de registro de domínios do Route 53 usando SDKs para Kotlin
- Exemplos do Amazon S3 usando o SDK para Kotlin
- SageMaker Exemplos de IA usando SDK para Kotlin
- Exemplos de Secrets Manager usando o SDK para Kotlin
- Exemplos do Amazon SES usando o SDK para Kotlin
- Exemplos do Amazon SNS usando o SDK para Kotlin
- Exemplos do Amazon SQS usando o SDK para Kotlin
- Exemplos do Step Functions usando o SDK para Kotlin
- Suporte exemplos de uso do SDK para Kotlin
- Exemplos do Amazon Translate usando o SDK para Kotlin
- X-Ray exemplos de uso do SDK para Kotlin

Exemplos do API Gateway usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o API Gateway.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos de Aurora usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Aurora.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de cluster do banco de dados do Aurora e definir os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Crie um snapshot do cluster do banco de dados e limpe os recursos.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:
```

```
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
```

```
            secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()

    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)
```

```
    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
 dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)

    println("10. Get a list of instance classes available for the selected engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
 dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)

    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
```

```
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
```

```
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
```

```
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
```

```
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
```

```
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
```

```
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
```

```
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}

suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
```

```
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Kotlin.

`CreateDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Para obter detalhes da API, consulte a CreateDBClusterParameterGroup referência da API AWS SDK para Kotlin.

`CreateDBClusterSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBClusterSnapshot`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Para obter detalhes da API, consulte a CreateDBClusterSnapshot referência da API AWS SDK para Kotlin.

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Kotlin.

`DeleteDBCluster`

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Kotlin.

`DeleteDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBClusterParameterGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
```

```
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteDBClusterParameterGroup referência da API AWS SDK para Kotlin.

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Kotlin.

`DescribeDBClusterParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameterGroups`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
```

```
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameterGroups referência da API AWS SDK para Kotlin.

`DescribeDBClusterParameters`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameters`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
```

```
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameters referência da API AWS SDK para Kotlin.

`DescribeDBClusterSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBClusterSnapshots`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
```

```
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Para obter detalhes da API, consulte a DescribeDBClusterSnapshots referência da API AWS SDK para Kotlin.

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
```

```
            }
        }
    }
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Kotlin.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersions referência da API AWS SDK para Kotlin.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Kotlin.

`ModifyDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Para obter detalhes da API, consulte a ModifyDBClusterParameterGroup referência da API AWS SDK para Kotlin.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Auto Scaling usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com Auto Scaling.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo do Amazon EC2 Auto Scaling com um modelo de inicialização e zonas de disponibilidade e obter informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas para CloudWatch métricas e, em seguida, limpe os recursos.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
```

```
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-linked
role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances in
the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of the
groups")
```

```
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
```

```
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
```

```
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
```

```
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
```

```
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateAutoScalingGroup

- DeleteAutoScalingGroup
- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Ações

`CreateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `CreateAutoScalingGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
```

```
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

- Para obter detalhes da API, consulte a CreateAutoScalingGroup referência da API AWS SDK para Kotlin.

`DeleteAutoScalingGroup`

O código de exemplo a seguir mostra como usar `DeleteAutoScalingGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
```

```
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAutoScalingGroup referência da API AWS SDK para Kotlin.

`DescribeAutoScalingGroups`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingGroups`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroups referência da API AWS SDK para Kotlin.

`DescribeAutoScalingInstances`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAutoScalingInstances referência da API AWS SDK para Kotlin.

`DescribeScalingActivities`

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeScalingActivities referência da API AWS SDK para Kotlin.

`DisableMetricsCollection`

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- Para obter detalhes da API, consulte a DisableMetricsCollection referência da API AWS SDK para Kotlin.

`EnableMetricsCollection`

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
```

```
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- Para obter detalhes da API, consulte a EnableMetricsCollection referência da API AWS SDK para Kotlin.

`SetDesiredCapacity`

O código de exemplo a seguir mostra como usar `SetDesiredCapacity`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- Para obter detalhes da API, consulte a SetDesiredCapacity referência da API AWS SDK para Kotlin.

`TerminateInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- Para obter detalhes da API, consulte a TerminateInstanceInAutoScalingGroup referência da API AWS SDK para Kotlin.

`UpdateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `UpdateAutoScalingGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}
```

- Para obter detalhes da API, consulte a UpdateAutoScalingGroup referência da API AWS SDK para Kotlin.

Exemplos do Amazon Bedrock usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListFoundationModels`

O código de exemplo a seguir mostra como usar `ListFoundationModels`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Amazon Bedrock disponíveis.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient.fromEnvironment { region = "us-east-1" }.use { bedrockClient ->
        val response =
 bedrockClient.listFoundationModels(ListFoundationModelsRequest {})
        response.modelSummaries?.forEach { model ->
            println("==========================================")
            println(" Model ID: ${model.modelId}")
            println("------------------------------------------")
            println(" Name: ${model.modelName}")
```

```
            println(" Provider: ${model.providerName}")
            println(" Input modalities: ${model.inputModalities}")
            println(" Output modalities: ${model.outputModalities}")
            println(" Supported customizations: ${model.customizationsSupported}")
            println(" Supported inference types: ${model.inferenceTypesSupported}")
            println("----------------------------------------\n")
        }
        return response.modelSummaries
    }
}
```

- Para obter detalhes da API, consulte a ListFoundationModels referência da API AWS SDK para Kotlin.

Exemplos do Amazon Bedrock Runtime usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Bedrock Runtime.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Amazônia Nova

Amazônia Nova

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
suspend fun main() {
    converse().also { println(it) }
}

suspend fun converse(): String {
    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in one line."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
```

```
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Send the request and process the model's response
        runCatching {
            val response = client.converse(request)
            return response.output!!.asMessage().content.first().asText()
        }.getOrElse { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId",
 error)
        }
    }
}
```

- Consulte detalhes da API em Converse na Referência da API do AWS SDK para Kotlin.

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processar o fluxo de respostas em tempo real.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamOutput
```

```
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message with a prompt
 * - Configure a streaming request with parameters
 * - Process the response stream in real time
 */
suspend fun main() {
    converseStream()
}

suspend fun converseStream(): String {
    // A buffer to collect the complete response
    val completeResponseBuffer = StringBuilder()

    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in a
 paragraph."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseStreamRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
```

```
            }
        }

        // Process the streaming response
        runCatching {
            client.converseStream(request) { response ->
                response.stream?.collect { chunk ->
                    when (chunk) {
                        is ConverseStreamOutput.ContentBlockDelta -> {
                            // Process each text chunk as it arrives
                            chunk.value.delta?.asText()?.let { text ->
                                print(text)
                                System.out.flush() // Ensure immediate output
                                completeResponseBuffer.append(text)
                            }
                        }
                        else -> {} // Other output block types can be handled as
 needed
                    }
                }
            }
        }.onFailure { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId:
 $error", error)
        }
    }

    return completeResponseBuffer.toString()
}
```

- Para obter detalhes da API, consulte a ConverseStream referência da API AWS SDK para Kotlin.

CloudWatch exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. CloudWatch

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá CloudWatch

O exemplo de código a seguir mostra como começar a usar o CloudWatch.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
```

```
            <namespace>
        Where:
            namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListMetrics referência da API AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Liste CloudWatch namespaces e métricas.
- Obter estatísticas para uma métrica e para faturamento estimado.

- Criar e atualizar um painel.
- Criar e adicionar dados a uma métrica.
- Criar e acionar um alarme e, em seguida, visualizar o histórico de alarmes.
- Criar um detector de anomalias.
- Obter uma imagem de métrica e, em seguida, limpar os recursos.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os CloudWatch recursos.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 To enable billing metrics and statistics for this example, make sure billing alerts
 are enabled for your account:
 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

 This Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 2. List available metrics within the selected namespace.
 3. Get statistics for the selected metric over the last day.
 4. Get CloudWatch estimated billing for the last week.
 5. Create a new CloudWatch dashboard with metrics.
 6. List dashboards using a paginator.
 7. Create a new custom metric by adding data for it.
 8. Add the custom metric to the dashboard.
```

```
 9. Create an alarm for the custom metric.
 10. Describe current alarms.
 11. Get current data for the new custom metric.
 12. Push data into the custom metric to trigger the alarm.
 13. Check the alarm state using the action DescribeAlarmsForMetric.
 14. Get alarm history for the new alarm.
 15. Add an anomaly detector for the custom metric.
 16. Describe current anomaly detectors.
 17. Get a metric image for the custom metric.
 18. Clean up the Amazon CloudWatch resources.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
 <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For example,
 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a dashboard.
 (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
 graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
```

```
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select one
from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList!![num - 1]
    } else {
```

```
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics, metricOption,
myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
```

```
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)

    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)
```

```
    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
```

```
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
```

```
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
```

```
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}

suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
```

```
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
```

```
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }
```

```
    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
```

```
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
```

```
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}

suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest =
        PutMetricAlarmRequest {
            alarmActions = alarmActionObs
            alarmDescription = "Example metric alarm"
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
            threshold = 100.00
            metricName = customMetricName
            namespace = customMetricNamespace
            evaluationPeriods = 1
            period = 10
            statistic = Statistic.Maximum
            datapointsToAlarm = 1
            treatMissingData = "ignore"
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}
```

```
suspend fun addMetricToDashboard(
    fileNameVal: String,
    dashboardNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension =
        Dimension {
            name = "UNIQUE_PAGES"
            value = "URLS"
        }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = "PAGES_VISITED"
            unit = StandardUnit.None
            value = dataPoint
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
            dimensions = listOf(dimension)
        }

    val request =
        PutMetricDataRequest {
            namespace = "SITE/TRAFFIC"
            metricData = listOf(datum)
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

```
fun readFileAsString(file: String): String =
 String(Files.readAllBytes(Paths.get(file)))

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension =
        Dimension {
            name = "Currency"
            value = "USD"
        }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest =
        GetMetricStatisticsRequest {
            metricName = "EstimatedCharges"
            namespace = "AWS/Billing"
            dimensions = dimensionList
            statistics = listOf(Statistic.Maximum)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            period = 86400
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
```

```
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

Ações

`DeleteAlarms`

O código de exemplo a seguir mostra como usar `DeleteAlarms`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
```

```
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAlarms referência da API AWS SDK para Kotlin.

`DeleteAnomalyDetector`

O código de exemplo a seguir mostra como usar `DeleteAnomalyDetector`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
```

```
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAnomalyDetector referência da API AWS SDK para Kotlin.

`DeleteDashboards`

O código de exemplo a seguir mostra como usar `DeleteDashboards`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteDashboards referência da API AWS SDK para Kotlin.

`DescribeAlarmHistory`

O código de exemplo a seguir mostra como usar `DescribeAlarmHistory`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
```

```
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAlarmHistory referência da API AWS SDK para Kotlin.

`DescribeAlarms`

O código de exemplo a seguir mostra como usar `DescribeAlarms`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAlarms referência da API AWS SDK para Kotlin.

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
```

```
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAlarmsForMetric referência da API AWS SDK para Kotlin.

`DescribeAnomalyDetectors`

O código de exemplo a seguir mostra como usar `DescribeAnomalyDetectors`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
```

```
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAnomalyDetectors referência da API AWS SDK para Kotlin.

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- Para obter detalhes da API, consulte a DisableAlarmActions referência da API AWS SDK para Kotlin.

`EnableAlarmActions`

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- Para obter detalhes da API, consulte a EnableAlarmActions referência da API AWS SDK para Kotlin.

`GetMetricData`

O código de exemplo a seguir mostra como usar `GetMetricData`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
```

```
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetMetricData referência da API AWS SDK para Kotlin.

`GetMetricStatistics`

O código de exemplo a seguir mostra como usar `GetMetricStatistics`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
```

```
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a GetMetricStatistics referência da API AWS SDK para Kotlin.

GetMetricWidgetImage

O código de exemplo a seguir mostra como usar `GetMetricWidgetImage`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- Para obter detalhes da API, consulte a GetMetricWidgetImage referência da API AWS SDK para Kotlin.

`ListDashboards`

O código de exemplo a seguir mostra como usar `ListDashboards`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListDashboards referência da API AWS SDK para Kotlin.

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
```

- Para obter detalhes da API, consulte a ListMetrics referência da API AWS SDK para Kotlin.

`PutAnomalyDetector`

O código de exemplo a seguir mostra como usar `PutAnomalyDetector`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- Para obter detalhes da API, consulte a PutAnomalyDetector referência da API AWS SDK para Kotlin.

`PutDashboard`

O código de exemplo a seguir mostra como usar `PutDashboard`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a PutDashboard referência da API AWS SDK para Kotlin.

`PutMetricAlarm`

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun putMetricAlarm(
    alarmNameVal: String,
    instanceIdVal: String,
) {
    val dimensionOb =
        Dimension {
            name = "InstanceId"
            value = instanceIdVal
        }

    val request =
        PutMetricAlarmRequest {
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanThreshold
            evaluationPeriods = 1
            metricName = "CPUUtilization"
            namespace = "AWS/EC2"
            period = 60
            statistic = Statistic.fromValue("Average")
            threshold = 70.0
            actionsEnabled = false
            alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
            unit = StandardUnit.fromValue("Seconds")
            dimensions = listOf(dimensionOb)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- Para obter detalhes da API, consulte a PutMetricAlarm referência da API AWS SDK para Kotlin.

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)
```

```
    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- Para obter detalhes da API, consulte a PutMetricData referência da API AWS SDK para Kotlin.

CloudWatch Exemplos de registros usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com registros. CloudWatch

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteSubscriptionFilter`

O código de exemplo a seguir mostra como usar `DeleteSubscriptionFilter`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- Para obter detalhes da API, consulte a DeleteSubscriptionFilter referência da API AWS SDK para Kotlin.

`DescribeSubscriptionFilters`

O código de exemplo a seguir mostra como usar `DescribeSubscriptionFilters`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSubscriptionFilters referência da API AWS SDK para Kotlin.

StartLiveTail

O código de exemplo a seguir mostra como usar `StartLiveTail`.

SDK para Kotlin

Inclua os arquivos necessários.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Inicie a sessão do Live Tail.

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
```

```
        logEventFilterPattern = logEventFilterPatternVal
    }

    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- Para obter detalhes da API, consulte a StartLiveTail referência da API AWS SDK para Kotlin.

Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AdminGetUser`

O código de exemplo a seguir mostra como usar `AdminGetUser`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- Para obter detalhes da API, consulte a AdminGetUser referência da API AWS SDK para Kotlin.

`AdminInitiateAuth`

O código de exemplo a seguir mostra como usar `AdminInitiateAuth`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- Para obter detalhes da API, consulte a AdminInitiateAuth referência da API AWS SDK para Kotlin.

`AdminRespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `AdminRespondToAuthChallenge`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }
```

```
    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- Para obter detalhes da API, consulte a AdminRespondToAuthChallenge referência da API AWS SDK para Kotlin.

`AssociateSoftwareToken`

O código de exemplo a seguir mostra como usar `AssociateSoftwareToken`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
```

```
}
```

- Para obter detalhes da API, consulte a AssociateSoftwareToken referência da API AWS SDK para Kotlin.

`ConfirmSignUp`

O código de exemplo a seguir mostra como usar `ConfirmSignUp`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- Para obter detalhes da API, consulte a ConfirmSignUp referência da API AWS SDK para Kotlin.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListUsers referência da API AWS SDK para Kotlin.

`ResendConfirmationCode`

O código de exemplo a seguir mostra como usar `ResendConfirmationCode`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- Para obter detalhes da API, consulte a ResendConfirmationCode referência da API AWS SDK para Kotlin.

`SignUp`

O código de exemplo a seguir mostra como usar `SignUp`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
```

```
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Para obter detalhes da API, consulte a SignUp referência da API AWS SDK para Kotlin.

`VerifySoftwareToken`

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
```

```
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- Para obter detalhes da API, consulte a VerifySoftwareToken referência da API AWS SDK para Kotlin.

Cenários

Inscrever um usuário em um grupo de usuários que exija MFA

O exemplo de código a seguir mostra como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.
```

```
 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being prompted
 to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
 script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
```

```
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a new
code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
```

```
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
```

```
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }
```

```
    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
```

```
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Exemplos do Amazon Comprehend usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Comprehend.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um aplicativo de mensagem

O exemplo de código a seguir mostra como criar uma aplicação de mensagens usando o Amazon SQS.

SDK para Kotlin

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Exemplos do DynamoDB usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma tabela do DynamoDB.

```
suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
```

```
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
```

```
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

Obtenha um item de uma tabela.

```
suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

Exemplo completo.

```
suspend fun main() {
    val tableName = "Movies"
    val fileName = "../../../resources/sample_files/movies.json"
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id and
 a sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
```

```
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deletIssuesTable(tableName)
}

suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
```

```
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
```

```
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }
```

```
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
```

```
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewTable(
    tableNameVal: String,
    key: String,
```

```
): String? {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

- Para obter detalhes da API, consulte a CreateTable referência da API AWS SDK para Kotlin.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDynamoDBItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        DeleteItemRequest {
            tableName = tableNameVal
            key = keyToGet
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- Para obter detalhes da API, consulte a DeleteItem referência da API AWS SDK para Kotlin.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTable referência da API AWS SDK para Kotlin.

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSpecificItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
```

```
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- Para obter detalhes da API, consulte a GetItem referência da API AWS SDK para Kotlin.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllTables() {
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte a ListTables referência da API AWS SDK para Kotlin.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
    songTitle: String,
    songTitleVal: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }
```

```
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- Para obter detalhes da API, consulte a PutItem referência da API AWS SDK para Kotlin.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }
```

```
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Kotlin.

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Kotlin.

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
    name: String,
    updateVal: String,
) {
    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] =
        AttributeValueUpdate {
            value = AttributeValue.S(updateVal)
            action = AttributeAction.Put
        }

    val request =
        UpdateItemRequest {
            tableName = tableNameVal
            key = itemKey
            attributeUpdates = updatedValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- Para obter detalhes da API, consulte a UpdateItem referência da API AWS SDK para Kotlin.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named id
 and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(
    ddb: DynamoDbClient,
    tableNameVal: String,
```

```
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {
```

```
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,
 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)
```

```
    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
```

```
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
```

```
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- Para obter detalhes da API, consulte a BatchExecuteStatement referência da API AWS SDK para Kotlin.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.

- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQ"
    val fileName = "../../../resources/sample_files/movies.json"
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key named
 id and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("******* Getting data from the MoviesPartiQ table.")
    getMoviePartiQL(ddb)

    println("******* Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("******* Updating a record.")
    updateTableItemPartiQL(ddb)

    println("******* Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("******* Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}

suspend fun createTablePartiQL(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
```

```
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(
    ddb: DynamoDbClient,
    fileName: String,
```

```
) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {
        if (t == 200) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
```

```
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
 Cooper\",\"Ernest B. Schoedsack\' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>,
): ExecuteStatementResponse {
    val request =
```

```
        ExecuteStatementRequest {
            statement = statementVal
            parameters = parametersVal
        }

    return ddb.executeStatement(request)
}
```

- Para obter detalhes da API, consulte a ExecuteStatement referência da API AWS SDK para Kotlin.

Exemplos do Amazon EC2 usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon EC2.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSecurityGroups referência da API AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks:

 1. Creates an RSA key pair and saves the private key data as a .pem file.
 2. Lists key pairs.
 3. Creates a security group for the default VPC.
 4. Displays security group information.
 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 6. Gets more information about the image.
 7. Gets a list of instance types that are compatible with the selected AMI’s
 architecture.
 8. Creates an instance with the key pair, security group, AMI, and an instance
 type.
 9. Displays information about the instance.
 10. Stops the instance and waits for it to stop.
 11. Starts the instance and waits for it to start.
 12. Allocates an Elastic IP address and associates it with the instance.
 13. Displays SSH connection info for the instance.
 14. Disassociates and deletes the Elastic IP address.
 15. Terminates the instance.
 16. Deletes the security group.
 17. Deletes the key pair.
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
```

```
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
 Console.
            myIpAddress - The IP address of your development machine.

"""

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }

    val keyName = args[0]
    val fileName = args[1]
    val groupName = args[2]
    val groupDesc = args[3]
    val vpcId = args[4]
    val myIpAddress = args[5]
    var newInstanceId: String? = ""

    println(DASHES)
    println("Welcome to the Amazon EC2 example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an RSA key pair and save the private key material as a .pem
 file.")
    createKeyPairSc(keyName, fileName)
    println(DASHES)

    println(DASHES)
    println("2. List key pairs.")
    describeEC2KeysSc()
    println(DASHES)

    println(DASHES)
```

```
    println("3. Create a security group.")
    val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId, myIpAddress)
    println(DASHES)

    println(DASHES)
    println("4. Display security group info for the newly created security group.")
    describeSecurityGroupsSc(groupId.toString())
    println(DASHES)

    println(DASHES)
    println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in the
name.")
    val instanceId = getParaValuesSc()
    if (instanceId == "") {
        println("The instance Id value isn't valid.")
        exitProcess(0)
    }
    println("The instance Id is $instanceId.")
    println(DASHES)

    println(DASHES)
    println("6. Get more information about an amzn2 image and return the AMI
value.")
    val amiValue = instanceId?.let { describeImageSc(it) }
    if (instanceId == "") {
        println("The instance Id value is invalid.")
        exitProcess(0)
    }
    println("The AMI value is $amiValue.")
    println(DASHES)

    println(DASHES)
    println("7. Get a list of instance types.")
    val instanceType = getInstanceTypesSc()
    println(DASHES)

    println(DASHES)
    println("8. Create an instance.")
    if (amiValue != null) {
        newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
        println("The instance Id is $newInstanceId")
    }
    println(DASHES)
```

```
    println(DASHES)
    println("9. Display information about the running instance. ")
    var ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("10.  Stop the instance.")
    if (newInstanceId != null) {
        stopInstanceSc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("11.  Start the instance.")
    if (newInstanceId != null) {
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
 instance.")
    val allocationId = allocateAddressSc()
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
    println("The associate Id value is $associationId")
    println(DASHES)

    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("14. Disassociate and release the Elastic IP address.")
    disassociateAddressSc(associationId)
    releaseEC2AddressSc(allocationId)
```

```
    println(DASHES)

    println(DASHES)
    println("15. Terminate the instance and use a waiter.")
    if (newInstanceId != null) {
        terminateEC2Sc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("16. Delete the security group.")
    if (groupId != null) {
        deleteEC2SecGroupSc(groupId)
    }
    println(DASHES)

    println(DASHES)
    println("17. Delete the key pair.")
    deleteKeysSc(keyName)
    println(DASHES)

    println(DASHES)
    println("You successfully completed the Amazon EC2 scenario.")
    println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
```

```
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitUntilInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
```

```
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
                    ?.get(0)
                    ?.instances
                    ?.get(0)
                    ?.state
                    ?.name
                    ?. value
            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
 ${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
 ${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
```

```
                    println("Instance state is
 ${response.reservations!!.get(0).instances?.get(0)?.state}")
                    pubAddress =
                        response.reservations!!
                            .get(0)
                            .instances
                            ?.get(0)
                            ?.publicIpAddress
                            .toString()
                    println("Instance address is $pubAddress")
                    isRunning = true
                }
            }
        }
    }
    return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
            imageId = amiIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
```

```
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
 ${response.images?.get(0)?.description}")
        println("The name of the first image is  ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
```

```
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient.fromEnvironment { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() + "
 and group VPC " + group.vpcId)
        }
    }
}
```

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
```

```
    }
}

suspend fun describeEC2KeysSc() {
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair

- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
```

```
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

- Para obter detalhes da API, consulte a AllocateAddress referência da API AWS SDK para Kotlin.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
```

```
        return associateResponse.associationId
    }
}
```

- Para obter detalhes da API, consulte a AssociateAddress referência da API AWS SDK para Kotlin.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
```

```
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

- Para obter detalhes da API, consulte a AuthorizeSecurityGroupIngress referência da API AWS SDK para Kotlin.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- Para obter detalhes da API, consulte a CreateKeyPair referência da API AWS SDK para Kotlin.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
```

```
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "0.0.0.0/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

- Para obter detalhes da API, consulte a CreateSecurityGroup referência da API AWS SDK para Kotlin.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- Para obter detalhes da API, consulte a DeleteKeyPair referência da API AWS SDK para Kotlin.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
```

```
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteSecurityGroup referência da API AWS SDK para Kotlin.

`DescribeInstanceTypes`

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
```

```
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}
```

- Para obter detalhes da API, consulte a DescribeInstanceTypes referência da API AWS SDK para Kotlin.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstances(request)
        response.reservations?.forEach { reservation ->
            reservation.instances?.forEach { instance ->
                println("Instance Id is ${instance.instanceId}")
                println("Image id is ${instance.imageId}")
                println("Instance type is ${instance.instanceType}")
```

```
                println("Instance state name is ${instance.state?.name}")
                println("monitoring information is ${instance.monitoring?.state}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeInstances referência da API AWS SDK para Kotlin.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEC2Keys() {
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeKeyPairs referência da API AWS SDK para Kotlin.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSecurityGroups referência da API AWS SDK para Kotlin.

`DisassociateAddress`

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
```

```
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- Para obter detalhes da API, consulte a DisassociateAddress referência da API AWS SDK para Kotlin.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- Para obter detalhes da API, consulte a ReleaseAddress referência da API AWS SDK para Kotlin.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
                value = name
            }

        val requestTags =
            CreateTagsRequest {
                resources = listOf(instanceId.toString())
                tags = listOf(tag)
            }
        ec2.createTags(requestTags)
        println("Successfully started EC2 Instance $instanceId based on AMI $amiId")
        return instanceId
    }
}
```

- Para obter detalhes da API, consulte a RunInstances referência da API AWS SDK para Kotlin.

StartInstances

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- Para obter detalhes da API, consulte a StartInstances referência da API AWS SDK para Kotlin.

StopInstances

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

- Para obter detalhes da API, consulte a StopInstances referência da API AWS SDK para Kotlin.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is ${instance.instanceId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a TerminateInstances referência da API AWS SDK para Kotlin.

Exemplos do Amazon ECR usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon ECR.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Amazon ECR

O exemplo de código a seguir mostra como começar a usar o Amazon ECR.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
```

```
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- Consulte detalhes da API em listImages na Referência de APIs do AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um repositório do Amazon ECR.
- Defina políticas de repositório.
- Recupere URIs de repositórios.
- Obtenha tokens de autorização do Amazon ECR.
- Defina políticas de ciclo de vida para repositórios do Amazon ECR.
- Envie por push uma imagem do Docker para um repositório do Amazon ECR.
- Verifique a existência de uma imagem em um repositório do Amazon ECR.
- Liste os repositórios do Amazon ECR da conta e verifique os detalhes sobre eles.
- Exclua repositórios do Amazon ECR.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre os recursos do Amazon ECR.

```
import java.util.Scanner
```

```
/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with the
 Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a local
 image,
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage: <iamRoleARN> <accountId>

        Where:
           iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
           accountId - Your AWS account number.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        return
    }
```

```
    var iamRole = args[0]
    var localImageName: String
    var accountId = args[1]
    val ecrActions = ECRActions()
    val scanner = Scanner(System.`in`)

    println(
        """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
        service provided by AWS. It allows developers and organizations to securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images throughout
their lifecycle,
        from building and testing to production deployment.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
developers to
        automate the storage, retrieval, and management of container images as part
of their application
        deployment pipelines. With ECR, teams can focus on building and deploying
their
        applications without having to worry about the underlying infrastructure
required to
        host and manage a container registry.

        This scenario walks you through how to perform key operations for this
service.
        Let's get started...

         You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
            this program that did not complete).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
```

```
            println("Continuing with the program...")
            println("")
            break
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }

    waitForInputToContinue(scanner)
    println(DASHES)
    println(
        """
        1. Create an ECR repository.

        The first task is to ensure we have a local Docker image named echo-text.
        If this image exists, then an Amazon ECR repository is created.

        An ECR repository is a private Docker container repository provided
        by Amazon Web Services (AWS). It is a managed service that makes it easy
        to store, manage, and deploy Docker container images.

        """.trimIndent(),
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)
```

```
    println(DASHES)
    println(
        """
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function is
crucial for maintaining
        the security and integrity of your container images. The repository policy
allows you to
        define specific rules and restrictions for accessing and managing the images
stored within your ECR
        repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
```

```
        valid authorization token, which is required to authenticate your requests
to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.


        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.getAuthToken()
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to deploy a
container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime to
pull the
        correct container image from the ECR repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
    println("The repository URI is $repositoryURI")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        6. Set an ECR Lifecycle Policy.

        An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
```

```
        These policies allow you to automatically remove old or unused Docker images
from your repositories,
        freeing up storage space and reducing costs.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val pol = ecrActions.setLifeCyclePolicy(repoName)
    println(pol)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by making a
call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message indicating
that the image was pushed to ECR.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)
```

```
    println(DASHES)
    println("9. As an optional step, you can interact with the image in Amazon ECR
 by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
 image? (y/n)")
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
        1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
 you need to authenticate with the registry. You can do this using the AWS CLI:

            aws ecr get-login-password --region us-east-1 | docker login --username
 AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

        2. Describe the image using this command:

           aws ecr describe-images --repository-name $repoName --image-ids imageTag=
$localImageName

        3. Run the Docker container and view the output using this command:

           docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository and
 have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
```

```
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }
}
```

Uma classe de wrapper para os métodos do SDK do Amazon ECR.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
```

```
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
```

```
                        "countNumber": 14
                    },
                    "action": {
                        "type": "expire"
                    }
                }
            ]
            }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
```

```
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
```

```
            val responseText = response.policyText
            return responseText
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeResponse = ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
        }
    }
```

```
    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
```

```
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
```

```
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)
```

```
            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
            val registryURL: String = repoData?.repositoryUri?.split("/")?.get(0) ?:
 ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Ações

`CreateRepository`

O código de exemplo a seguir mostra como usar `CreateRepository`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
    * @throws RepositoryAlreadyExistsException if the repository exists.
    * @throws EcrException         if an error occurs while creating the
repository.
    */
   suspend fun createECRRepository(repoName: String?): String? {
       val request =
           CreateRepositoryRequest {
               repositoryName = repoName
           }

       return try {
           EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
               val response = ecrClient.createRepository(request)
               response.repository?.repositoryArn
           }
       } catch (e: RepositoryAlreadyExistsException) {
           println("Repository already exists: $repoName")
           repoName?.let { getRepoARN(it) }
       } catch (e: EcrException) {
           println("An error occurred: ${e.message}")
           null
       }
   }
```

- Para obter detalhes da API, consulte a CreateRepository referência da API AWS SDK para Kotlin.

`DeleteRepository`

O código de exemplo a seguir mostra como usar `DeleteRepository`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- Para obter detalhes da API, consulte a DeleteRepository referência da API AWS SDK para Kotlin.

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
```

```
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a DescribeImages referência da API AWS SDK para Kotlin.

`DescribeRepositories`

O código de exemplo a seguir mostra como usar `DescribeRepositories`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
```

```
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- Para obter detalhes da API, consulte a DescribeRepositories referência da API AWS SDK para Kotlin.

`GetAuthorizationToken`

O código de exemplo a seguir mostra como usar `GetAuthorizationToken`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
```

```
        }
    }
```

- Para obter detalhes da API, consulte a GetAuthorizationToken referência da API AWS SDK para Kotlin.

`GetRepositoryPolicy`

O código de exemplo a seguir mostra como usar `GetRepositoryPolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- Para obter detalhes da API, consulte a GetRepositoryPolicy referência da API AWS SDK para Kotlin.

PushImageCmd

O código de exemplo a seguir mostra como usar PushImageCmd.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   suspend fun pushDockerImage(
       repoName: String,
       imageName: String,
   ) {
       println("Pushing $imageName to $repoName will take a few seconds")
       val authConfig = getAuthConfig(repoName)

       EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
           val desRequest =
               DescribeRepositoriesRequest {
                   repositoryNames = listOf(repoName)
               }

           val describeRepoResponse = ecrClient.describeRepositories(desRequest)
           val repoData =
               describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                   ?: throw RuntimeException("Repository not found: $repoName")
```

```
            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

- Para obter detalhes da API, consulte a PushImageCmd referência da API AWS SDK para Kotlin.

`SetRepositoryPolicy`

O código de exemplo a seguir mostra como usar `SetRepositoryPolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a SetRepositoryPolicy referência da API AWS SDK para Kotlin.

StartLifecyclePolicyPreview

O código de exemplo a seguir mostra como usar `StartLifecyclePolicyPreview`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
```

```
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Para obter detalhes da API, consulte a StartLifecyclePolicyPreview referência da API AWS SDK para Kotlin.

OpenSearch Exemplos de serviços usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin with Service. OpenSearch

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateDomain`

O código de exemplo a seguir mostra como usar `CreateDomain`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }

    val ebsOptionsOb =
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptionsOb =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
            clusterConfig = clusterConfigOb
            ebsOptions = ebsOptionsOb
            nodeToNodeEncryptionOptions = encryptionOptionsOb
        }

    println("Sending domain creation request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- Para obter detalhes da API, consulte a CreateDomain referência da API AWS SDK para Kotlin.

DeleteDomain

O código de exemplo a seguir mostra como usar `DeleteDomain`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteDomain referência da API AWS SDK para Kotlin.

ListDomainNames

O código de exemplo a seguir mostra como usar `ListDomainNames`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllDomains() {
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- Para obter detalhes da API, consulte a ListDomainNames referência da API AWS SDK para Kotlin.

`UpdateDomainConfig`

O código de exemplo a seguir mostra como usar `UpdateDomainConfig`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
```

```
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
}
```

- Para obter detalhes da API, consulte a UpdateDomainConfig referência da API AWS SDK para Kotlin.

EventBridge exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. EventBridge

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá EventBridge

O exemplo de código a seguir mostra como começar a usar o EventBridge.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request =
        ListEventBusesRequest {
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-west-2" }.use { eventBrClient -
>
        val response: ListEventBusesResponse = eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListEventBuses referência da API AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma regra e adicionar um destino a ela.
- Habilitar e desabilitar regras.
- Listar e atualizar regras e destinos.
- Enviar eventos e, em seguida, limpar os recursos.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/*
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
 2. Creates an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
 events enabled.
 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 4. Lists rules on the event bus.
 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and lets the
 user subscribe to it.
 6. Adds a target to the rule that sends an email to the specified topic.
 7. Creates an EventBridge event that sends an email when an Amazon S3 object is
 created.
 8. Lists targets.
 9. Lists the rules for the same target.
 10. Triggers the rule by uploading a file to the S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints the state of the rule.
 13. Adds a transform to the rule to change the text of the email.
 14. Enables a specific rule.
```

```
 15. Triggers the updated rule by uploading a file to the S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleName> <bucketName> <topicName> <eventRuleName>

    Where:
        roleName - The name of the role to create.
        bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name to
 create.
        topicName - The name of the Amazon Simple Notification Service (Amazon SNS)
 topic to create.
        eventRuleName - The Amazon EventBridge rule name to create.
    """
    val polJSON =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
```

```
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS Identity and Access Management (IAM) role to use with
Amazon EventBridge.")
    val roleArn = createIAMRole(roleName, polJSON)
    println(DASHES)

    println(DASHES)
    println("2. Create an S3 bucket with EventBridge events enabled.")
    if (checkBucket(bucketName)) {
        println("$bucketName already exists. Ending this scenario.")
        exitProcess(1)
    }

    createBucket(bucketName)
    delay(3000)
    setBucketNotification(bucketName)
    println(DASHES)

    println(DASHES)
    println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
    delay(10000)
    addEventRule(roleArn, bucketName, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("4. List rules on the event bus.")
    listRules()
    println(DASHES)

    println(DASHES)
    println("5. Create a new SNS topic for testing and let the user subscribe to the
topic.")
    val topicArn = createSnsTopic(topicName)
    println(DASHES)

    println(DASHES)
    println("6. Add a target to the rule that sends an email to the specified
topic.")
    println("Enter your email to subscribe to the Amazon SNS topic:")
    val email = sc.nextLine()
```

```
    subEmail(topicArn, email)
    println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("7. Create an EventBridge event that sends an email when an Amazon S3
object is created.")
    addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName, bucketName)
    println(DASHES)

    println(DASHES)
    println("8. List targets.")
    listTargets(eventRuleName)
    println(DASHES)

    println(DASHES)
    println(" 9. List the rules for the same target.")
    listTargetRules(topicArn)
    println(DASHES)

    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)
```

```
    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
    println(DASHES)

    println(DASHES)
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
    println("17. Send an event to trigger the rule. This will trigger a subscription
 email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("18. Clean up resources.")
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully completed.")
    println(DASHES)
}
```

```
suspend fun cleanupResources(
    topicArn: String?,
    eventRuleName: String?,
    bucketName: String?,
    roleName: String?,
) {
    println("Removing all targets from the event rule.")
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest =
        DetachRolePolicyRequest {
            policyArn = policyArnVal
            roleName = roleNameVal
        }
    IamClient.fromEnvironment { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")

        // Delete the role.
        val roleRequest =
            DeleteRoleRequest {
                roleName = roleNameVal
            }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
```

```
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
```

```
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
```

```
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putEvents(eventsRequest)
    }
}

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
```

```
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern =
        "{" +
            "\"source\": [\"ExampleSource\"]," +
            "\"detail-type\": [\"ExampleType\"]" +
            "}"
    val request =
        PutRuleRequest {
            name = ruleName
            description = "Custom test rule"
            eventPattern = customEventsPattern
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb =
        InputTransformer {
```

```
            inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
            inputPathsMap = myMap
        }
    val targetOb =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(targetOb)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
```

```
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb =
        PutObjectRequest {
            bucket = bucketName
            key = fileName
            body = myFile.asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
```

```
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
```

```
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}

suspend fun subEmail(
    topicArnVal: String?,
    email: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Sid": "EventBridgePublishTopic",
                "Effect": "Allow",
                "Principal": {
```

```
                    "Service": "events.amazonaws.com"
                },
                "Resource": "*",
                "Action": "sns:Publish"
            }
        ]
    }
    """.trimIndent()

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
    }
}

suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
```

```
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """
    {
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
    val eventBridgeConfig =
        EventBridgeConfiguration {
        }

    val configuration =
        NotificationConfiguration {
            eventBridgeConfiguration = eventBridgeConfig
        }

    val configurationRequest =
```

```
        PutBucketNotificationConfigurationRequest {
            bucket = bucketName
            notificationConfiguration = configuration
            skipDestinationValidation = true
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest =
            HeadBucketRequest {
                bucket = bucketName
            }

        S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
```

```
}

suspend fun createIAMRole(
    rolenameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    val rolePolicyRequest =
        AttachRolePolicyRequest {
            roleName = rolenameVal
            policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
        }

    IamClient.fromEnvironment { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Ações

`DeleteRule`

O código de exemplo a seguir mostra como usar `DeleteRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- Para obter detalhes da API, consulte a DeleteRule referência da API AWS SDK para Kotlin.

`DescribeRule`

O código de exemplo a seguir mostra como usar `DescribeRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- Para obter detalhes da API, consulte a DescribeRule referência da API AWS SDK para Kotlin.

`DisableRule`

O código de exemplo a seguir mostra como usar `DisableRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
```

```
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- Para obter detalhes da API, consulte a DisableRule referência da API AWS SDK para Kotlin.

`EnableRule`

O código de exemplo a seguir mostra como usar `EnableRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
```

```
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- Para obter detalhes da API, consulte a EnableRule referência da API AWS SDK para Kotlin.

`ListRuleNamesByTarget`

O código de exemplo a seguir mostra como usar `ListRuleNamesByTarget`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListRuleNamesByTarget referência da API AWS SDK para Kotlin.

`ListRules`

O código de exemplo a seguir mostra como usar `ListRules`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListRules referência da API AWS SDK para Kotlin.

`ListTargetsByRule`

O código de exemplo a seguir mostra como usar `ListTargetsByRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListTargetsByRule referência da API AWS SDK para Kotlin.

`PutEvents`

O código de exemplo a seguir mostra como usar `PutEvents`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun triggerCustomRule(email: String) {
```

```
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- Para obter detalhes da API, consulte a PutEvents referência da API AWS SDK para Kotlin.

`PutRule`

O código de exemplo a seguir mostra como usar `PutRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma regra agendada

```
suspend fun createScRule(
    ruleName: String?,
    cronExpression: String?,
) {
    val ruleRequest =
        PutRuleRequest {
            name = ruleName
            eventBusName = "default"
            scheduleExpression = cronExpression
            state = RuleState.Enabled
            description = "A test rule that runs on a schedule created by the Kotlin
 API"
        }

    EventBridgeClient.fromEnvironment { region = "us-west-2" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

Crie uma regra que seja acionada quando um objeto é adicionado a um bucket do Amazon Simple Storage Service.

```
// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """
    {
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }
    """.trimIndent()
```

```
    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- Para obter detalhes da API, consulte a PutRule referência da API AWS SDK para Kotlin.

`PutTargets`

O código de exemplo a seguir mostra como usar `PutTargets`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
```

```
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}
```

Adicione um transformador de entrada a um destino para uma regra.

```
suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }
```

```
    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- Para obter detalhes da API, consulte a PutTargets referência da API AWS SDK para Kotlin.

`RemoveTargets`

O código de exemplo a seguir mostra como usar `RemoveTargets`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
```

```
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a RemoveTargets referência da API AWS SDK para Kotlin.

AWS Glue exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. AWS Glue

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
 <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon
 S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
```

```
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
```

```
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }
```

```
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

```
suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {
    val runRequest =
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
```

```
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
            maxResults = 10
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
```

```
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable

- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
```

```
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- Para obter detalhes da API, consulte a CreateCrawler referência da API AWS SDK para Kotlin.

`GetCrawler`

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
```

```
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- Para obter detalhes da API, consulte a GetCrawler referência da API AWS SDK para Kotlin.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- Para obter detalhes da API, consulte a GetDatabase referência da API AWS SDK para Kotlin.

`StartCrawler`

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- Para obter detalhes da API, consulte a StartCrawler referência da API AWS SDK para Kotlin.

Exemplos de IAM usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações do usuário do IAM.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>
```

```
    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
```

```
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:*"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject
```

```
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
```

```
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient = StsClient.fromEnvironment { region = "us-east-1" }
    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials = StaticCredentialsProvider {
        accessKeyId = key
        secretAccessKey = secKey
        sessionToken = secToken
    }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 = S3Client.fromEnvironment {
        region = "us-east-1"
```

```
        credentialsProvider = staticCredentials
    }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
```

```
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy

- DetachRolePolicy
- PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
```

```
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- Para obter detalhes da API, consulte a AttachRolePolicy referência da API AWS SDK para Kotlin.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
```

```
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- Para obter detalhes da API, consulte a CreateAccessKey referência da API AWS SDK para Kotlin.

`CreateAccountAlias`

O código de exemplo a seguir mostra como usar `CreateAccountAlias`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- Para obter detalhes da API, consulte a CreateAccountAlias referência da API AWS SDK para Kotlin.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "dynamodb:DeleteItem",
                    "dynamodb:GetItem",
                    "dynamodb:PutItem",
                    "dynamodb:Scan",
                    "dynamodb:UpdateItem"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- Para obter detalhes da API, consulte a CreatePolicy referência da API AWS SDK para Kotlin.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- Para obter detalhes da API, consulte a CreateUser referência da API AWS SDK para Kotlin.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAccessKey referência da API AWS SDK para Kotlin.

`DeleteAccountAlias`

O código de exemplo a seguir mostra como usar `DeleteAccountAlias`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- Para obter detalhes da API, consulte a DeleteAccountAlias referência da API AWS SDK para Kotlin.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- Para obter detalhes da API, consulte a DeletePolicy referência da API AWS SDK para Kotlin.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteUser referência da API AWS SDK para Kotlin.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DetachRolePolicy referência da API AWS SDK para Kotlin.

`GetPolicy`

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
```

```
    }
}
```

- Para obter detalhes da API, consulte a GetPolicy referência da API AWS SDK para Kotlin.

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListAccessKeys referência da API AWS SDK para Kotlin.

`ListAccountAliases`

O código de exemplo a seguir mostra como usar `ListAccountAliases`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAliases() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListAccountAliases referência da API AWS SDK para Kotlin.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllUsers() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
```

```
            if (permissionsBoundary != null) {
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a ListUsers referência da API AWS SDK para Kotlin.

`UpdateUser`

O código de exemplo a seguir mostra como usar `UpdateUser`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- Para obter detalhes da API, consulte a UpdateUser referência da API AWS SDK para Kotlin.

AWS IoT exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. AWS IoT

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
```

```
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Para detalhes da API, consulte listThings em Referência de API AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie uma AWS IoT coisa.
- Gerar um certificado de dispositivo.
- Atualize AWS IoT qualquer coisa com atributos.
- Exibir um endpoint exclusivo.
- Liste seus AWS IoT certificados.
- Atualize uma AWS IoT sombra.
- Gravar informações do estado.
- Cria uma regra.
- Listar suas regras.
- Pesquisar coisas usando o nome da coisa.

- Exclua qualquer AWS IoT coisa.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
```

```
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-
started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS Internet
 of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an IoT
 thing, generating a device certificate,
```

```
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that can
be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
        A device certificate performs a role in securing the communication between
devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
```

```
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
```

```
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example, you
can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
```

```
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName? (y/
n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
 delete.")
    }
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase = true))
 {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
```

```
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
```

```
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse = iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}
```

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
```

```
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
```

```
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }
```

```
    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn
```

```
        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration

- UpdateThing

Ações

`AttachThingPrincipal`

O código de exemplo a seguir mostra como usar `AttachThingPrincipal`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- Para obter detalhes da API, consulte a AttachThingPrincipal referência da API AWS SDK para Kotlin.

`CreateKeysAndCertificate`

O código de exemplo a seguir mostra como usar `CreateKeysAndCertificate`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- Para obter detalhes da API, consulte a CreateKeysAndCertificate referência da API AWS SDK para Kotlin.

`CreateThing`

O código de exemplo a seguir mostra como usar `CreateThing`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIoTThing(thingNameVal: String) {
```

```
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Para obter detalhes da API, consulte a CreateThing referência da API AWS SDK para Kotlin.

`CreateTopicRule`

O código de exemplo a seguir mostra como usar `CreateTopicRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
```

```
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- Para obter detalhes da API, consulte a CreateTopicRule referência da API AWS SDK para Kotlin.

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
```

```
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteCertificate referência da API AWS SDK para Kotlin.

`DeleteThing`

O código de exemplo a seguir mostra como usar `DeleteThing`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteThing referência da API AWS SDK para Kotlin.

`DescribeEndpoint`

O código de exemplo a seguir mostra como usar `DescribeEndpoint`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- Para obter detalhes da API, consulte a DescribeEndpoint referência da API AWS SDK para Kotlin.

`DescribeThing`

O código de exemplo a seguir mostra como usar `DescribeThing`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
```

```
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeThing referência da API AWS SDK para Kotlin.

`DetachThingPrincipal`

O código de exemplo a seguir mostra como usar `DetachThingPrincipal`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
```

```
}
```

- Para obter detalhes da API, consulte a DetachThingPrincipal referência da API AWS SDK para Kotlin.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListCertificates referência da API AWS SDK para Kotlin.

`SearchIndex`

O código de exemplo a seguir mostra como usar `SearchIndex`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- Para obter detalhes da API, consulte a SearchIndex referência da API AWS SDK para Kotlin.

`UpdateThing`

O código de exemplo a seguir mostra como usar `UpdateThing`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- Para obter detalhes da API, consulte a UpdateThing referência da API AWS SDK para Kotlin.

AWS IoT data exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. AWS IoT data

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetThingShadow`

O código de exemplo a seguir mostra como usar `GetThingShadow`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}
```

- Para obter detalhes da API, consulte a GetThingShadow referência da API AWS SDK para Kotlin.

`UpdateThingShadow`

O código de exemplo a seguir mostra como usar `UpdateThingShadow`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- Para obter detalhes da API, consulte a UpdateThingShadow referência da API AWS SDK para Kotlin.

Exemplos do Amazon Keyspaces usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Keyspaces.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Keyspaces

O exemplo de código a seguir mostra como começar a usar o Amazon Keyspaces.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/

suspend fun main() {
    listKeyspaces()
}
```

```
suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListKeyspaces referência da API AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um keyspace e uma tabela. O esquema da tabela contém dados do filme e tem a recuperação para um ponto no tempo.
- Conectar-se ao keyspace usando uma conexão TLS segura com autenticação SigV4.
- Consultar a tabela. Adicionar, recuperar e atualizar dados do filme.
- Atualizar a tabela. Adicionar uma coluna para rastrear os filmes assistidos.
- Restaurar a tabela ao estado anterior e limpar os recursos.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This example uses a secure file format to hold certificate information for
 Kotlin applications. This is required to make a connection to Amazon Keyspaces.
 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html

 This Kotlin example performs the following tasks:

 1. Create a keyspace.
 2. Check for keyspace existence.
 3. List keyspaces using a paginator.
 4. Create a table with a simple movie data schema and enable point-in-time
 recovery.
 5. Check for the table to be in an Active state.
 6. List all tables in the keyspace.
 7. Use a Cassandra driver to insert some records into the Movie table.
 8. Get all records from the Movie table.
 9. Get a specific Movie.
 10. Get a UTC timestamp for the current time.
 11. Update the table schema to add a ‘watched’ Boolean column.
 12. Update an item as watched.
 13. Query for items with watched = True.
 14. Restore the table back to the previous state using the timestamp.
 15. Check for completion of the restore action.
 16. Delete the table.
 17. Confirm that both tables are deleted.
 18. Delete the keyspace.
 */

/*
   Usage:
     fileName - The name of the JSON file that contains movie data. (Get this file
 from the GitHub repo at resources/sample_file.)
     keyspaceName - The name of the keyspace to create.
```

```
 */
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
    println(DASHES)

    println(DASHES)
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("3. List keyspaces using a paginator.")
    listKeyspacesPaginator()
    println(DASHES)

    println(DASHES)
    println("4. Create a table with a simple movie data schema and enable point-in-
time recovery.")
    createTable(keyspaceName, tableName)
    println(DASHES)
```

```
    println(DASHES)
    println("5. Check for the table to be in an Active state.")
    delay(6000)
    checkTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("6. List all tables in the keyspace.")
    listTables(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("7. Use a Cassandra driver to insert some records into the Movie
 table.")
    delay(6000)
    loadData(session, fileName, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("8. Get all records from the Movie table.")
    getMovieData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("9. Get a specific Movie.")
    getSpecificMovie(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("10. Get a UTC timestamp for the current time.")
    val utc = ZonedDateTime.now(ZoneOffset.UTC)
    println("DATETIME = ${Date.from(utc.toInstant())}")
    println(DASHES)

    println(DASHES)
    println("11. Update the table schema to add a watched Boolean column.")
    updateTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("12. Update an item as watched.")
    delay(10000) // Wait 10 seconds for the update.
    updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
    println(DASHES)
```

```
    println(DASHES)
    println("13. Query for items with watched = True.")
    getWatchedData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("14. Restore the table back to the previous state using the timestamp.")
    println("Note that the restore operation can take up to 20 minutes.")
    restoreTable(keyspaceName, utc)
    println(DASHES)

    println(DASHES)
    println("15. Check for completion of the restore action.")
    delay(5000)
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("17. Confirm that both tables are deleted.")
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
```

```
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
            // Keep looping until the table cannot be found and a
 ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
```

```
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }

        val cols = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun restoreTable(
```

```
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\"
 WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0 AND
 year = :k1;"
```

```
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\" WHERE title
 = 'The Family' ALLOW FILTERING ;")
```

```
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin
\";")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
 (:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
```

```
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )

        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null
```

```
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
```

```
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
```

```
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
```

```
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Ações

`CreateKeyspace`

O código de exemplo a seguir mostra como usar `CreateKeyspace`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
```

```
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Para obter detalhes da API, consulte a CreateKeyspace referência da API AWS SDK para Kotlin.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
```

```
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
```

```
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- Para obter detalhes da API, consulte a CreateTable referência da API AWS SDK para Kotlin.

`DeleteKeyspace`

O código de exemplo a seguir mostra como usar `DeleteKeyspace`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- Para obter detalhes da API, consulte a DeleteKeyspace referência da API AWS SDK para Kotlin.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- Para obter detalhes da API, consulte a DeleteTable referência da API AWS SDK para Kotlin.

`GetKeyspace`

O código de exemplo a seguir mostra como usar `GetKeyspace`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- Para obter detalhes da API, consulte a GetKeyspace referência da API AWS SDK para Kotlin.

`GetTable`

O código de exemplo a seguir mostra como usar `GetTable`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
```

```
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a GetTable referência da API AWS SDK para Kotlin.

`ListKeyspaces`

O código de exemplo a seguir mostra como usar `ListKeyspaces`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
```

```
    }
}
```

- Para obter detalhes da API, consulte a ListKeyspaces referência da API AWS SDK para Kotlin.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListTables referência da API AWS SDK para Kotlin.

`RestoreTable`

O código de exemplo a seguir mostra como usar `RestoreTable`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- Para obter detalhes da API, consulte a RestoreTable referência da API AWS SDK para Kotlin.

`UpdateTable`

O código de exemplo a seguir mostra como usar `UpdateTable`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- Para obter detalhes da API, consulte a UpdateTable referência da API AWS SDK para Kotlin.

AWS KMS exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. AWS KMS

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateAlias`

O código de exemplo a seguir mostra como usar `CreateAlias`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- Para obter detalhes da API, consulte a CreateAlias referência da API AWS SDK para Kotlin.

`CreateGrant`

O código de exemplo a seguir mostra como usar `CreateGrant`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- Para obter detalhes da API, consulte a CreateGrant referência da API AWS SDK para Kotlin.

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- Para obter detalhes da API, consulte a CreateKey referência da API AWS SDK para Kotlin.

`Decrypt`

O código de exemplo a seguir mostra como usar `Decrypt`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
```

```
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK para Kotlin.

`DescribeKey`

O código de exemplo a seguir mostra como usar `DescribeKey`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeKey referência da API AWS SDK para Kotlin.

`DisableKey`

O código de exemplo a seguir mostra como usar `DisableKey`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
```

```
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- Para obter detalhes da API, consulte a DisableKey referência da API AWS SDK para Kotlin.

`EnableKey`

O código de exemplo a seguir mostra como usar `EnableKey`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- Para obter detalhes da API, consulte a EnableKey referência da API AWS SDK para Kotlin.

`Encrypt`

O código de exemplo a seguir mostra como usar `Encrypt`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
```

```
        print(myVal)
    }
}
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK para Kotlin.

`ListAliases`

O código de exemplo a seguir mostra como usar `ListAliases`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListAliases referência da API AWS SDK para Kotlin.

`ListGrants`

O código de exemplo a seguir mostra como usar `ListGrants`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListGrants referência da API AWS SDK para Kotlin.

`ListKeys`

O código de exemplo a seguir mostra como usar `ListKeys`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllKeys() {
```

```
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListKeys referência da API AWS SDK para Kotlin.

Exemplos de Lambda usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>

        Where:
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that
 has AWS Lambda permissions.
            handler - The fully qualified method name (for example,
 example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
 that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
 or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file (for
 example, LambdaHello-1.0-SNAPSHOT.jar).
            """
```

```
    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val functionName = args[0]
    val role = args[1]
    val handler = args[2]
    val bucketName = args[3]
    val updatedBucketName = args[4]
    val key = args[5]

    println("Creating a Lambda function named $functionName.")
    val funArn = createScFunction(functionName, bucketName, key, handler, role)
    println("The AWS Lambda ARN is $funArn")

    // Get a specific Lambda function.
    println("Getting the $functionName AWS Lambda function.")
    getFunction(functionName)

    // List the Lambda functions.
    println("Listing all AWS Lambda functions.")
    listFunctionsSc()

    // Invoke the Lambda function.
    println("*** Invoke the Lambda function.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function code.
    println("*** Update the Lambda function code.")
    updateFunctionCode(functionName, updatedBucketName, key)

    // println("*** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    updateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}
```

```
suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
}

suspend fun getFunction(functionNameVal: String) {
    val functionRequest =
        GetFunctionRequest {
            functionName = functionNameVal
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
 ${response.configuration?.runtime}")
```

```
    }
}

suspend fun listFunctionsSc() {
    val request =
        ListFunctionsRequest {
            maxItems = 10
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            payload = byteArray
            logType = LogType.Tail
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(
    functionNameVal: String?,
    bucketName: String?,
    key: String?,
) {
    val functionCodeRequest =
        UpdateFunctionCodeRequest {
            functionName = functionNameVal
            publish = true
            s3Bucket = bucketName
            s3Key = key
```

```
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun updateFunctionConfiguration(
    functionNameVal: String?,
    handlerVal: String?,
) {
    val configurationRequest =
        UpdateFunctionConfigurationRequest {
            functionName = functionNameVal
            handler = handlerVal
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateFunction
  - DeleteFunction

- GetFunction
- Invoke
- ListFunctions
- UpdateFunctionCode
- UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String? {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
```

```
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

- Para obter detalhes da API, consulte a CreateFunction referência da API AWS SDK para Kotlin.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Para obter detalhes da API, consulte a DeleteFunction referência da API AWS SDK para Kotlin.

`Invocar`

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun invokeFunction(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            logType = LogType.Tail
            payload = byteArray
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
        println("The log result is ${res.logResult}")
    }
}
```

- Consulte detalhes da API em Invoke na Referência da API AWS SDK para Kotlin.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos do Amazon Location usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Location.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Location

O exemplo de código a seguir mostra como começar a usar o Amazon Location Service.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you need to create a collection using the AWS Management
console. For information, see the following documentation.

https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:
            <colletionName>
```

```
        Where:
            colletionName - The Amazon location collection name.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }
    val colletionName = args[0]
    listGeofences(colletionName)
}

/**
 * Lists the geofences for the specified collection name.
 *
 * @param collectionName the name of the geofence collection
 */
suspend fun listGeofences(collectionName: String) {
    val request = ListGeofencesRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.listGeofences(request)
        val geofences = response.entries
        if (geofences.isNullOrEmpty()) {
            println("No Geofences found")
        } else {
            geofences.forEach { geofence ->
                println("Geofence ID: ${geofence.geofenceId}")
            }
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - ListGeofenceCollections
  - ListGeofences

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um mapa do Amazon Location.
- Criar uma chave de API do Amazon Location.
- Exibir o URL do mapa.
- Criar uma coleção de geocercas
- Armazenar uma geometria de geocerca.
- Criar um recurso de rastreador.
- Atualizar a posição de um dispositivo.
- Recuperar a atualização de posição mais recente de um dispositivo específico.
- Criar uma calculadora de rotas.
- Determinar a distância entre Seattle e Vancouver.
- Usar as APIs de nível superior do Amazon Location.
- Excluir os ativos de localização da Amazon.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

        Where:
            mapName - The name of the map to create (e.g., "AWSMap").
            keyName - The name of the API key to create (e.g., "AWSApiKey").
            collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
            geoId - The geographic identifier used for the geofence or map (e.g.,
 "geoId").
            trackerName - The name of the tracker (e.g., "geoTracker").
            calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
            deviceId - The ID of the device (e.g., "iPhone-112356").
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(0)
    }

    val mapName = args[0]
    val keyName = args[1]
    val collectionName = args[2]
    val geoId = args[3]
    val trackerName = args[4]
    val calculatorName = args[5]
    val deviceId = args[6]

    println(
        """
    AWS Location Service is a fully managed service offered by Amazon Web Services
 (AWS) that
    provides location-based services for developers. This service simplifies
    the integration of location-based features into applications, making it
    easier to build and deploy location-aware applications.

    The AWS Location Service offers a range of location-based services,
    including:

    - Maps: The service provides access to high-quality maps, satellite imagery,
```

```
      and geospatial data from various providers, allowing developers to
      easily embed maps into their applications.

    - Tracking: The Location Service enables real-time tracking of mobile devices,
      assets, or other entities, allowing developers to build applications
      that can monitor the location of people, vehicles, or other objects.

    - Geocoding: The service provides the ability to convert addresses or
      location names into geographic coordinates (latitude and longitude),
      and vice versa, enabling developers to integrate location-based search
      and routing functionality into their applications.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println(DASHES)
    println("1. Create an AWS Location Service map")
    println(
        """
        An AWS Location map can enhance the user experience of your
        application by providing accurate and personalized location-based
        features. For example, you could use the geocoding capabilities to
        allow users to search for and locate businesses, landmarks, or
        other points of interest within a specific region.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    val mapArn = createMap(mapName)
    println("The Map ARN is: $mapArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    waitForInputToContinue(scanner)
    println("2. Create an AWS Location API key")
    println(
        """
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.
```

```
        """.trimIndent(),
    )
    val keyArn = createKey(keyName, mapArn)
    println("The Key ARN is: $keyArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("3. Display Map URL")
    println(
        """
        In order to get the MAP URL, you need to get the API Key value.
        You can get the key value using the AWS Management Console under
        Location Services. This operation cannot be completed using the
        AWS SDK. For more information about getting the key value, see
        the AWS Location Documentation.
        """.trimIndent(),
    )
    val mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/$mapName/tiles/{z}/
{x}/{y}?key={KeyValue}"
    println("Embed this URL in your Web app: $mapUrl")
    println("")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("4. Create a geofence collection, which manages and stores geofences.")
    waitForInputToContinue(scanner)
    val collectionArn: String =
        createGeofenceCollection(collectionName)
    println("The geofence collection was successfully created: $collectionArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println("5. Store a geofence geometry in a given geofence collection.")
    println(
        """
        An AWS Location geofence is a virtual boundary that defines a geographic
 area
        on a map. It is a useful feature for tracking the location of
        assets or monitoring the movement of objects within a specific region.

        To define a geofence, you need to specify the coordinates of a
        polygon that represents the area of interest. The polygon must be
```

```
        defined in a counter-clockwise direction, meaning that the points of
        the polygon must be listed in a counter-clockwise order.

        This is a requirement for the AWS Location service to correctly
        interpret the geofence and ensure that the location data is
        accurately processed within the defined area.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    putGeofence(collectionName, geoId)
    println("Successfully created geofence: $geoId")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Create a tracker resource which lets you retrieve current and
historical location of devices.")
    waitForInputToContinue(scanner)
    val trackerArn: String = createTracker(trackerName)
    println("Successfully created tracker. ARN: $trackerArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Update the position of a device in the location tracking system.")
    println(
        """
        The AWS location service does not enforce a strict format for deviceId, but
it must:
            - Be a string (case-sensitive).
            - Be 1–100 characters long.
            - Contain only:
            - Alphanumeric characters (A-Z, a-z, 0-9)
            - Underscores (_)
            - Hyphens (-)
            - Be the same ID used when sending and retrieving positions.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    updateDevicePosition(trackerName, deviceId)
    println("$deviceId was successfully updated in the location tracking system.")
```

```
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("8. Retrieve the most recent position update for a specified device.")
    waitForInputToContinue(scanner)
    val response = getDevicePosition(trackerName, deviceId)
    println("Successfully fetched device position: ${response.position}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("9. Create a route calculator.")
    waitForInputToContinue(scanner)
    val routeResponse = createRouteCalculator(calculatorName)
    println("Route calculator created successfully: ${routeResponse.calculatorArn}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("10. Determine the distance in kilometers between Seattle and Vancouver
using the route calculator.")
    waitForInputToContinue(scanner)
    val responseDis = calcDistance(calculatorName)
    println("Successfully calculated route. The distance in kilometers is
${responseDis.summary?.distance}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("11. Use the GeoPlacesClient to perform additional operations.")
    println(
        """
        This scenario will show use of the GeoPlacesClient that enables
        location search and geocoding capabilities for your applications.

        We are going to use this client to perform these AWS Location tasks:
            - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
            - Place Search (searchText): Finds places based on search queries.
            - Nearby Search (searchNearby): Finds places near a specific location.

        """.trimIndent(),
    )
```

```
    waitForInputToContinue(scanner)
    println("First we will perform a Reverse Geocoding operation")
    waitForInputToContinue(scanner)
    reverseGeocode()

    println("Now we are going to perform a text search using coffee shop.")
    waitForInputToContinue(scanner)
    searchText("coffee shop")
    waitForInputToContinue(scanner)

    println("Now we are going to perform a nearby Search.")
    waitForInputToContinue(scanner)
    searchNearby()
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS Location Services resources.")
    println("Would you like to delete the AWS Location Services resources? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        deleteMap(mapName)
        deleteKey(keyName)
        deleteGeofenceCollection(collectionName)
        deleteTracker(trackerName)
        deleteRouteCalculator(calculatorName)
    } else {
        println("The AWS resources will not be deleted.")
    }
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println(" This concludes the AWS Location Service scenario.")
    println(DASHES)
}

/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
```

```
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}


/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}


/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}


/**
 * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
 * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
 * The results are processed and printed once the search completes successfully.
 */
suspend fun searchNearby() {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)
```

```
    // Set up the request for searching nearby places.
    val request = SearchNearbyRequest {
        this.queryPosition = queryPosition
        this.queryRadius = 1000L
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchNearby(request)

        // Process the response and print the results.
        response.resultItems?.forEach { result ->
            println("Title: ${result.title}")
            println("Address: ${result.address?.label}")
            println("Distance: ${result.distance} meters")
            println("-------------------------")
        }
    }
}


/**
 * Searches for a place using the provided search query and prints the detailed
 information of the first result.
 *
 * @param searchQuery the search query to be used for the place search (ex, coffee
 shop)
 */
suspend fun searchText(searchQuery: String) {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    val request = SearchTextRequest {
        this.queryText = searchQuery
        this.biasPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchText(request)

        response.resultItems?.firstOrNull()?.let { result ->
            val placeId = result.placeId // Get Place ID
            println("Found Place with id: $placeId")
```

```
            // Fetch detailed info using getPlace.
            val getPlaceRequest = GetPlaceRequest {
                this.placeId = placeId
            }

            val placeResponse = client.getPlace(getPlaceRequest)

            // Print detailed place information.
            println("Detailed Place Information:")
            println("Title: ${placeResponse.title}")
            println("Address: ${placeResponse.address?.label}")

            // Print each food type (if any).
            placeResponse.foodTypes?.takeIf { it.isNotEmpty() }?.let {
                println("Food Types:")
                it.forEach { foodType ->
                    println("  - $foodType")
                }
            } ?: run {
                println("No food types available.")
            }

            println("-------------------------")
        }
    }
}

/**
 * Performs reverse geocoding using the AWS Geo Places API.
 * Reverse geocoding is the process of converting geographic coordinates (latitude
 and longitude) to a human-readable address.
 * This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.
 */
suspend fun reverseGeocode() {
    val latitude = 37.7749
    val longitude = -122.4194
    println("Use latitude 37.7749 and longitude -122.4194")

    // AWS expects [longitude, latitude].
    val queryPosition = listOf(longitude, latitude)
    val request = ReverseGeocodeRequest {
        this.queryPosition = queryPosition
```

```
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.reverseGeocode(request)
        response.resultItems?.forEach { result ->
            println("The address is: ${result.address?.label}")
        }
    }
}


/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}

/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
```

```
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}

/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}

/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
```

```
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}

/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}

/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
```

```
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}

/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}

/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
```

```
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}

/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}

fun waitForInputToContinue(scanner: Scanner) {
```

```
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            println("Invalid input. Please try again.")
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

Ações

`BatchUpdateDevicePosition`

O código de exemplo a seguir mostra como usar `BatchUpdateDevicePosition`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

- Para obter detalhes da API, consulte a BatchUpdateDevicePosition referência da API AWS SDK para Kotlin.

`CalculateRoute`

O código de exemplo a seguir mostra como usar `CalculateRoute`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}
```

- Para obter detalhes da API, consulte a CalculateRoute referência da API AWS SDK para Kotlin.

`CreateGeofenceCollection`

O código de exemplo a seguir mostra como usar `CreateGeofenceCollection`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}
```

- Para obter detalhes da API, consulte a CreateGeofenceCollection referência da API AWS SDK para Kotlin.

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}
```

- Para obter detalhes da API, consulte a CreateKey referência da API AWS SDK para Kotlin.

`CreateMap`

O código de exemplo a seguir mostra como usar `CreateMap`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}
```

- Para obter detalhes da API, consulte a CreateMap referência da API AWS SDK para Kotlin.

`CreateRouteCalculator`

O código de exemplo a seguir mostra como usar `CreateRouteCalculator`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}
```

- Para obter detalhes da API, consulte a CreateRouteCalculator referência da API AWS SDK para Kotlin.

`CreateTracker`

O código de exemplo a seguir mostra como usar `CreateTracker`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}
```

- Para obter detalhes da API, consulte a CreateTracker referência da API AWS SDK para Kotlin.

`DeleteGeofenceCollection`

O código de exemplo a seguir mostra como usar `DeleteGeofenceCollection`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteGeofenceCollection referência da API AWS SDK para Kotlin.

`DeleteKey`

O código de exemplo a seguir mostra como usar `DeleteKey`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteKey referência da API AWS SDK para Kotlin.

`DeleteMap`

O código de exemplo a seguir mostra como usar `DeleteMap`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
```

```
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteMap referência da API AWS SDK para Kotlin.

`DeleteRouteCalculator`

O código de exemplo a seguir mostra como usar `DeleteRouteCalculator`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
```

```
    }
}
```

- Para obter detalhes da API, consulte a DeleteRouteCalculator referência da API AWS SDK para Kotlin.

`DeleteTracker`

O código de exemplo a seguir mostra como usar `DeleteTracker`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTracker referência da API AWS SDK para Kotlin.

`GetDevicePosition`

O código de exemplo a seguir mostra como usar `GetDevicePosition`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}
```

- Para obter detalhes da API, consulte a GetDevicePosition referência da API AWS SDK para Kotlin.

`PutGeofence`

O código de exemplo a seguir mostra como usar `PutGeofence`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}
```

- Para obter detalhes da API, consulte a PutGeofence referência da API AWS SDK para Kotlin.

MediaConvert exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. MediaConvert

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInput1: String,
): String? {
    // Step 1: Describe endpoints to get the MediaConvert endpoint URL
    val describeResponse = mcClient.describeEndpoints(
        DescribeEndpointsRequest {
            maxResults = 1
        },
    )

    val endpointUrl = describeResponse.endpoints?.firstOrNull()?.url
        ?: error("No MediaConvert endpoint found")

    // Step 2: Create MediaConvert client with resolved endpoint
    val mediaConvert = MediaConvertClient.fromEnvironment {
        region = "us-west-2"
        endpointProvider = MediaConvertEndpointProvider {
            Endpoint(endpointUrl)
        }
```

```
    }

    // Output destination folder in S3 - put in 'output/' folder beside input
    val outputDestination = fileInput1.substringBeforeLast('/') + "/output/"

    // Step 3: Create the job request with minimal valid video codec settings
    val jobRequest = CreateJobRequest {
        role = mcRoleARN
        settings = JobSettings {
            inputs = listOf(
                Input {
                    fileInput = fileInput1
                },
            )
            outputGroups = listOf(
                OutputGroup {
                    outputGroupSettings = OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings = FileGroupSettings {
                            destination = outputDestination
                        }
                    }
                    outputs = listOf(
                        Output {
                            containerSettings = ContainerSettings {
                                container = ContainerType.Mp4
                            }
                            videoDescription = VideoDescription {
                                width = 1280
                                height = 720
                                codecSettings = VideoCodecSettings {
                                    codec = VideoCodec.H264
                                    h264Settings = H264Settings {
                                        rateControlMode = H264RateControlMode.Qvbr
                                        qvbrSettings = H264QvbrSettings {
                                            qvbrQualityLevel = 7
                                        }
                                        maxBitrate = 5_000_000
                                        codecLevel = H264CodecLevel.Auto
                                        codecProfile = H264CodecProfile.Main
                                        framerateControl =
H264FramerateControl.InitializeFromSource
                                    }
                                }
```

```
                            }
                        },
                    )
                },
            )
        }
    }

    // Step 4: Call MediaConvert to create the job
    val response = mediaConvert.createJob(jobRequest)

    // Return the job ID or null if not found
    return response.job?.id
}
```

- Para obter detalhes da API, consulte a CreateJob referência da API AWS SDK para Kotlin.

`GetJob`

O código de exemplo a seguir mostra como usar `GetJob`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSpecificJob(mcClient: MediaConvertClient, jobId: String) {
    // 1. Discover the correct endpoint
    val res = mcClient.describeEndpoints(DescribeEndpointsRequest { maxResults =
 1 })
    var endpointUrl = res.endpoints?.firstOrNull()?.url
        ?: error(" No MediaConvert endpoint found")

    // 2. Create a new client using the endpoint
    val clientWithEndpoint = MediaConvertClient {
        region = "us-west-2"
        endpointUrl = endpointUrl
```

```
    }

    // 3. Get the job details
    val jobResponse = clientWithEndpoint.getJob(GetJobRequest { id = jobId })
    val job = jobResponse.job

    println("Job status: ${job?.status}")
    println("Job ARN: ${job?.arn}")
    println("Output group count: ${job?.settings?.outputGroups?.size}")
}
```

- Para obter detalhes da API, consulte a GetJob referência da API AWS SDK para Kotlin.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listCompleteJobs(mcClient: MediaConvertClient) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
    val endpointURL = res.endpoints!![0].url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
```

```
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobsRequest =
        ListJobsRequest {
            maxResults = 10
            status = JobStatus.fromValue("COMPLETE")
        }

    val jobsResponse = mediaConvert.listJobs(jobsRequest)
    val jobs = jobsResponse.jobs
    if (jobs != null) {
        for (job in jobs) {
            println("The JOB ARN is ${job.arn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListJobs referência da API AWS SDK para Kotlin.

Exemplos do Amazon Pinpoint usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateApp`

O código de exemplo a seguir mostra como usar `CreateApp`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- Para obter detalhes da API, consulte a CreateApp referência da API AWS SDK para Kotlin.

`CreateCampaign`

O código de exemplo a seguir mostra como usar `CreateCampaign`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val scheduleOb =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessageOb =
        Message {
            action = Action.OpenApp
            body = "My message body"
            title = "My message title"
        }

    val messageConfigurationOb =
        MessageConfiguration {
            defaultMessage = defaultMessageOb
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = scheduleOb
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfigurationOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
```

```
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- Para obter detalhes da API, consulte a CreateCampaign referência da API AWS SDK para Kotlin.

`CreateSegment`

O código de exemplo a seguir mostra como usar `CreateSegment`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {
    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts =
        AttributeDimension {
            attributeType = AttributeType.Inclusive
            values = myList
        }

    segmentAttributes["Team"] = atts
    val recencyDimension =
        RecencyDimension {
            duration = Duration.fromValue("DAY_30")
            recencyType = RecencyType.fromValue("ACTIVE")
        }
```

```
    val segmentBehaviors =
        SegmentBehaviors {
            recency = recencyDimension
        }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb =
        SegmentDimensions {
            attributes = segmentAttributes
            behavior = segmentBehaviors
            demographic = SegmentDemographics {}
            location = segmentLocation
        }

    val writeSegmentRequestOb =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensionsOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequestOb
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- Para obter detalhes da API, consulte a CreateSegment referência da API AWS SDK para Kotlin.

`DeleteApp`

O código de exemplo a seguir mostra como usar `DeleteApp`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteApp referência da API AWS SDK para Kotlin.

`DeleteEndpoint`

O código de exemplo a seguir mostra como usar `DeleteEndpoint`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
```

```
) {
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- Para obter detalhes da API, consulte a DeleteEndpoint referência da API AWS SDK para Kotlin.

`GetEndpoint`

O código de exemplo a seguir mostra como usar `GetEndpoint`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.getEndpoint(
                GetEndpointRequest {
                    applicationId = appId
                    endpointId = endpoint
                },
```

```
            )
        val endResponse = result.endpointResponse

        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson =
            GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create()

        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

- Para obter detalhes da API, consulte a GetEndpoint referência da API AWS SDK para Kotlin.

`GetSegments`

O código de exemplo a seguir mostra como usar `GetSegments`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listSegs(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte a GetSegments referência da API AWS SDK para Kotlin.

SendMessages

O código de exemplo a seguir mostra como usar `SendMessages`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val body: String =
    """
    Amazon Pinpoint test (AWS SDK for Kotlin)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK for
 Kotlin.

    """.trimIndent()

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <subject> <appId> <senderAddress> <toAddress>

    Where:
        subject - The email subject to use.
```

```
        senderAddress - The from address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
        toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
    """

    if (args.size != 3) {
        println(usage)
        exitProcess(0)
    }

    val subject = args[0]
    val senderAddress = args[1]
    val toAddress = args[2]
    sendEmail(subject, senderAddress, toAddress)
}

suspend fun sendEmail(
    subjectVal: String?,
    senderAddress: String,
    toAddressVal: String,
) {
    var content =
        Content {
            data = body
        }

    val messageBody =
        Body {
            text = content
        }

    val subContent =
        Content {
            data = subjectVal
        }

    val message =
        Message {
            body = messageBody
            subject = subContent
        }

    val destinationOb =
```

```
        Destination {
            toAddresses = listOf(toAddressVal)
        }

    val emailContent =
        EmailContent {
            simple = message
        }

    val sendEmailRequest =
        SendEmailRequest {
            fromEmailAddress = senderAddress
            destination = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient.fromEnvironment { region = "us-east-1" }.use { pinpointemail
 ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- Para obter detalhes da API, consulte a SendMessages referência da API AWS SDK para Kotlin.

Exemplos do Amazon RDS usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon RDS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de banco de dados e definir os valores dos parâmetros.
- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.
- Exclua a instância e o grupo de parâmetros.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this code example, set up your development environment, including
 your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

```
This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking the
 createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment parameters
 by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions
 method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance that
 contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>

        Where:
            dbGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
```

```
            secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val dbSnapshotIdentifier = args[4]
    val secretName = args[5]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeDBEngines()

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)
```

```
    println("8. Get a list of micro instance classes available for the selected
 engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database and
 uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
 username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
```

```
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

```
// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
```

```
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
    dbGroupNameVal: String?,
```

```
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}
```

```
// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
    }
}
```

```
// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset or
 auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    System.out.println("*** The parameter value is
 ${para.parameterValue}")
                    System.out.println("*** The parameter data type is
 ${para.dataType}")
                    System.out.println("*** The parameter description is
 ${para.description}")
                    System.out.println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
```

```
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-west-2" }.use
 { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions

- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Ações

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Kotlin.

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Kotlin.

`DescribeAccountAttributes`

O código de exemplo a seguir mostra como usar `DescribeAccountAttributes`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAccountAttributes() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeAccountAttributes referência da API AWS SDK para Kotlin.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeInstances() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest {})
        response.dbInstances?.forEach { instance ->
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Kotlin.

`ModifyDBInstance`

O código de exemplo a seguir mostra como usar `ModifyDBInstance`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- Consulte detalhes da API em ModifyDBInstance na Referência da API AWS SDK para Kotlin.

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon RDS Data Service usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon RDS Data Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon Redshift usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Redshift.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cluster.

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient.fromEnvironment { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
```

```
}
```

- Para obter detalhes da API, consulte a CreateCluster referência da API AWS SDK para Kotlin.

DeleteCluster

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir o cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- Para obter detalhes da API, consulte a DeleteCluster referência da API AWS SDK para Kotlin.

DescribeClusters

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Descrever o cluster.

```
suspend fun describeRedshiftClusters() {
    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse =
 redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeClusters referência da API AWS SDK para Kotlin.

`ModifyCluster`

O código de exemplo a seguir mostra como usar `ModifyCluster`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Modificar um cluster.

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
            "The modified cluster was successfully modified and has
 ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance window",
        )
    }
}
```

- Para obter detalhes da API, consulte a ModifyCluster referência da API AWS SDK para Kotlin.

Cenários

Criar uma aplicação Web para rastrear dados do Amazon Redshift

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho usando um banco de dados do Amazon Redshift.

SDK para Kotlin

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- banco de dados de origem
- Amazon SES

Exemplos do Amazon Rekognition usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Rekognition.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CompareFaces`

O código de exemplo a seguir mostra como usar `CompareFaces`.

Para obter mais informações, consulte Comparação de faces em imagens.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
```

```
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches with
${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
${compareFacesResult.sourceImageOrientationCorrection}")
```

```
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- Para obter detalhes da API, consulte a CompareFaces referência da API AWS SDK para Kotlin.

`CreateCollection`

O código de exemplo a seguir mostra como usar `CreateCollection`.

Para obter mais informações, consulte Criar uma coleção.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- Para obter detalhes da API, consulte a CreateCollection referência da API AWS SDK para Kotlin.

`DeleteCollection`

O código de exemplo a seguir mostra como usar `DeleteCollection`.

Para obter mais informações, consulte Excluir uma coleção.

SDK para Kotlin

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Para obter detalhes da API, consulte a DeleteCollection referência da API AWS SDK para Kotlin.

`DeleteFaces`

O código de exemplo a seguir mostra como usar `DeleteFaces`.

Para obter mais informações, consulte Excluir faces de uma coleção.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Para obter detalhes da API, consulte a DeleteFaces referência da API AWS SDK para Kotlin.

`DescribeCollection`

O código de exemplo a seguir mostra como usar `DescribeCollection`.

Para obter mais informações, consulte Descrever uma coleção.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeCollection referência da API AWS SDK para Kotlin.

`DetectFaces`

O código de exemplo a seguir mostra como usar `DetectFaces`.

Para obter mais informações, consulte Detectar faces em uma imagem.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
```

```
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between ${ageRange?.low}
 and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectFaces referência da API AWS SDK para Kotlin.

`DetectLabels`

O código de exemplo a seguir mostra como usar `DetectLabels`.

Para obter mais informações, consulte Detectar rótulos em uma imagem.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
```

```
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DetectLabels referência da API AWS SDK para Kotlin.

`DetectModerationLabels`

O código de exemplo a seguir mostra como usar `DetectModerationLabels`.

Para obter mais informações, consulte Detectar imagens impróprias.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
```

```
    }
}
```

- Para obter detalhes da API, consulte a DetectModerationLabels referência da API AWS SDK para Kotlin.

`DetectText`

O código de exemplo a seguir mostra como usar `DetectText`.

Para obter mais informações, consulte Detectar texto em uma imagem.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
```

```
}
```

- Para obter detalhes da API, consulte a DetectText referência da API AWS SDK para Kotlin.

IndexFaces

O código de exemplo a seguir mostra como usar `IndexFaces`.

Para obter mais informações, consulte Adicionar faces a uma coleção.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
```

```
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a IndexFaces referência da API AWS SDK para Kotlin.

`ListCollections`

O código de exemplo a seguir mostra como usar `ListCollections`.

Para obter mais informações, consulte Listar coleções.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
```

```
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- Para obter detalhes da API, consulte a ListCollections referência da API AWS SDK para Kotlin.

`ListFaces`

O código de exemplo a seguir mostra como usar `ListFaces`.

Para obter mais informações, consulte Listar faces em uma coleção.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListFaces referência da API AWS SDK para Kotlin.

`RecognizeCelebrities`

O código de exemplo a seguir mostra como usar `RecognizeCelebrities`.

Para obter mais informações, consulte Reconhecer celebridades em uma imagem.

SDK para Kotlin

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- Para obter detalhes da API, consulte a RecognizeCelebrities referência da API AWS SDK para Kotlin.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar informações em vídeos

O exemplo de código a seguir mostra como:

- Iniciar trabalhos do Amazon Rekognition para detectar elementos como pessoas, objetos e texto em vídeos.
- Verificar o status do trabalho até que os trabalhos terminem.
- Visualizar a lista de elementos detectados por cada trabalho.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Detecte faces em um vídeo armazenado em um bucket do Amazon S3.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
```

```
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Detecte conteúdo impróprio ou ofensivo em um vídeo armazenado em um bucket do Amazon S3.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
```

```
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection

- StartTextDetection

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Exemplos de registro de domínios do Route 53 usando SDKs para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o registro de domínio do Route 53.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Conceitos básicos
- Ações

Conceitos básicos

Registro de domínios do Olá, Route 53

O exemplo de código a seguir mostra como começar a usar o registro de domínio do Route 53.

SDK para Kotlin

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <domainType>

       Where:
           domainType - The domain type (for example, com).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val domainType = args[0]
    println("Invokes ListPrices using a Paginated method.")
```

```
    listPricesPaginated(domainType)
}

suspend fun listPricesPaginated(domainType: String) {
    val pricesRequest =
        ListPricesRequest {
            maxItems = 10
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListPrices referência da API AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Listar os domínios atuais e as operações do ano passado.
- Ver o faturamento do ano passado e os preços dos tipos de domínio.
- Receber sugestões de domínio.
- Verificar a disponibilidade e a transferibilidade de um domínio.

- Opcionalmente, solicitar o registro de um domínio.
- Obter os detalhes de uma operação.
- Opcionalmente, obtenha os detalhes de um domínio.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin code example performs the following operations:

1. List current domains.
2. List operations in the past year.
3. View billing for the account in the past year.
4. View prices for domain types.
5. Get domain suggestions.
6. Check domain availability.
7. Check domain transferability.
8. Request a domain registration.
9. Get operation details.
10. Optionally, get domain details.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
 <lastName> <city>
```

```
        Where:
           domainType - The domain type (for example, com).
           phoneNumber - The phone number to use (for example, +1.2065550100)
           email - The email address to use.
           domainSuggestion - The domain suggestion (for example, findmy.example).
           firstName - The first name to use to register a domain.
           lastName -  The last name to use to register a domain.
           city - The city to use to register a domain.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val domainType = args[0]
    val phoneNumber = args[1]
    val email = args[2]
    val domainSuggestion = args[3]
    val firstName = args[4]
    val lastName = args[5]
    val city = args[6]

    println(DASHES)
    println("Welcome to the Amazon Route 53 domains example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List current domains.")
    listDomains()
    println(DASHES)

    println(DASHES)
    println("2. List operations in the past year.")
    listOperations()
    println(DASHES)

    println(DASHES)
    println("3. View billing for the account in the past year.")
    listBillingRecords()
    println(DASHES)

    println(DASHES)
    println("4. View prices for domain types.")
```

```
    listAllPrices(domainType)
    println(DASHES)

    println(DASHES)
    println("5. Get domain suggestions.")
    listDomainSuggestions(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("6. Check domain availability.")
    checkDomainAvailability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("7. Check domain transferability.")
    checkDomainTransferability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("8. Request a domain registration.")
    val opId = requestDomainRegistration(domainSuggestion, phoneNumber, email,
 firstName, lastName, city)
    println(DASHES)

    println(DASHES)
    println("9. Get operation details.")
    getOperationalDetail(opId)
    println(DASHES)

    println(DASHES)
    println("10. Get domain details.")
    println("Note: You must have a registered domain to get details.")
    println("Otherwise an exception is thrown that states ")
    println("Domain xxxxxxx not found in xxxxxxx account.")
    getDomainDetails(domainSuggestion)
    println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
```

```
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}

suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}

suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
```

```
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}

suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}

suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
```

```
}

suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}

suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

```
suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}

suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }
```

```
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}

suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Ações

`CheckDomainAvailability`

O código de exemplo a seguir mostra como usar `CheckDomainAvailability`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- Para obter detalhes da API, consulte a CheckDomainAvailability referência da API AWS SDK para Kotlin.

`CheckDomainTransferability`

O código de exemplo a seguir mostra como usar `CheckDomainTransferability`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- Para obter detalhes da API, consulte a CheckDomainTransferability referência da API AWS SDK para Kotlin.

`GetDomainDetail`

O código de exemplo a seguir mostra como usar `GetDomainDetail`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
```

```
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}
```

- Para obter detalhes da API, consulte a GetDomainDetail referência da API AWS SDK para Kotlin.

`GetDomainSuggestions`

O código de exemplo a seguir mostra como usar `GetDomainSuggestions`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
```

```
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetDomainSuggestions referência da API AWS SDK para Kotlin.

`GetOperationDetail`

O código de exemplo a seguir mostra como usar `GetOperationDetail`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- Para obter detalhes da API, consulte a GetOperationDetail referência da API AWS SDK para Kotlin.

`ListDomains`

O código de exemplo a seguir mostra como usar `ListDomains`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListDomains referência da API AWS SDK para Kotlin.

`ListOperations`

O código de exemplo a seguir mostra como usar `ListOperations`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listOperations() {
```

```
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListOperations referência da API AWS SDK para Kotlin.

`ListPrices`

O código de exemplo a seguir mostra como usar `ListPrices`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListPrices referência da API AWS SDK para Kotlin.

`RegisterDomain`

O código de exemplo a seguir mostra como usar `RegisterDomain`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
```

```
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- Para obter detalhes da API, consulte a RegisterDomain referência da API AWS SDK para Kotlin.

`ViewBilling`

O código de exemplo a seguir mostra como usar `ViewBilling`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ViewBilling referência da API AWS SDK para Kotlin.

Exemplos do Amazon S3 usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for example,
 C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
```

```
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }
```

```
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
```

```
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
```

```
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- Para obter detalhes da API, consulte a CopyObject referência da API AWS SDK para Kotlin.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
```

```
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- Para obter detalhes da API, consulte a CreateBucket referência da API AWS SDK para Kotlin.

`CreateMultiRegionAccessPoint`

O código de exemplo a seguir mostra como usar `CreateMultiRegionAccessPoint`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o cliente de controle do S3 para enviar a solicitação à região us-west-2.

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West (Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

Crie o ponto de Multi-Region acesso.

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
```

```
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

Aguarde até que o ponto de Multi-Region acesso fique disponível.

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )

            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Kotlin.
- Para obter detalhes da API, consulte a CreateMultiRegionAccessPoint referência da API AWS SDK para Kotlin.

`DeleteBucketPolicy`

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- Para obter detalhes da API, consulte a DeleteBucketPolicy referência da API AWS SDK para Kotlin.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
```

```
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- Para obter detalhes da API, consulte a DeleteObjects referência da API AWS SDK para Kotlin.

`GetBucketPolicy`

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")
```

```
    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- Para obter detalhes da API, consulte a GetBucketPolicy referência da API AWS SDK para Kotlin.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
```

```
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetObject referência da API AWS SDK para Kotlin.

`GetObjectAcl`

O código de exemplo a seguir mostra como usar `GetObjectAcl`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetObjectAcl referência da API AWS SDK para Kotlin.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- Para obter detalhes da API, consulte ListObjects V2 no AWS SDK para referência da API Kotlin.

`PutBucketAcl`

O código de exemplo a seguir mostra como usar `PutBucketAcl`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
```

```
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- Para obter detalhes da API, consulte a PutBucketAcl referência da API AWS SDK para Kotlin.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
```

```
    }
}
```

- Para obter detalhes da API, consulte a PutObject referência da API AWS SDK para Kotlin.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma solicitação pré-assinada de `GetObject` e use o URL para fazer download de um objeto.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET request
 to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()
```

```
    return objectContents
}
```

Crie uma solicitação `GetObject` atribuída previamente com opções avançadas.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be used
 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Crie uma solicitação pré-assinada de `PutObject` e use-a para fazer upload de um objeto.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
```

```
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Kotlin.

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Obter um objeto de um ponto de Multi-Region acesso

O exemplo de código a seguir mostra como obter um objeto de um ponto de Multi-Region acesso.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure o cliente do S3 para usar o algoritmo de assinatura Asymmetric Sigv4 (Sigv4a).

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a) signing
algorithm.
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

Use o ARN do ponto de Multi-Region acesso em vez de um nome de bucket para recuperar o objeto.

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
```

```
        return stringObj
    }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Kotlin.
- Para obter detalhes da API, consulte a GetObject referência da API AWS SDK para Kotlin.

SageMaker Exemplos de IA usando SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com IA. SageMaker

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá SageMaker AI

O exemplo de código a seguir mostra como começar a usar a SageMaker IA.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listBooks() {
    SageMakerClient.fromEnvironment { region = "us-west-2" }.use { sageMakerClient -
>
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListNotebookInstances referência da API AWS SDK para Kotlin.

Ações

`CreatePipeline`

O código de exemplo a seguir mostra como usar `CreatePipeline`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
```

```
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}
```

- Para obter detalhes da API, consulte a CreatePipeline referência da API AWS SDK para Kotlin.

`DeletePipeline`

O código de exemplo a seguir mostra como usar `DeletePipeline`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeletePipeline referência da API AWS SDK para Kotlin.

`DescribePipelineExecution`

O código de exemplo a seguir mostra como usar `DescribePipelineExecution`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
```

```
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}
```

- Para obter detalhes da API, consulte a DescribePipelineExecution referência da API AWS SDK para Kotlin.

`StartPipelineExecution`

O código de exemplo a seguir mostra como usar `StartPipelineExecution`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
```

```
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3:///$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
```

```
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}
```

- Para obter detalhes da API, consulte a StartPipelineExecution referência da API AWS SDK para Kotlin.

Cenários

Conceitos básicos de trabalhos geoespaciais e pipelines

O exemplo de código a seguir mostra como:

- Configurar recursos para um pipeline.

- Configurar um pipeline que executa um trabalho geoespacial.
- Iniciar a execução de um pipeline.
- Monitorar o status da execução.
- Ver a saída do pipeline.
- Limpar recursos.

Para obter mais informações, consulte Criar e executar SageMaker pipelines usando AWS SDKs em. Community.aws

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
          <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
 <queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
 <pipelineName>

    Where:
        sageMakerRoleName - The name of the Amazon SageMaker role.
        lambdaRoleName - The name of the AWS Lambda role.
        functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).
        functionKey - The name of the Amazon S3 key name that represents the Lambda
 function (for example, SageMakerLambda.zip).
        queueName - The name of the Amazon Simple Queue Service (Amazon SQS) queue.
        bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
 bucket.
        bucketFunction - The name of the Amazon S3 bucket that contains the Lambda
 ZIP file.
```

```
        lnglatData - The file location of the latlongtest.csv file required for this
 use case.
        spatialPipelinePath - The file location of the GeoSpatialPipeline.json file
 required for this use case.
        pipelineName - The name of the pipeline to create (for example, sagemaker-
sdk-example-pipeline).
    """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]
    val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

    println(DASHES)
    println("Welcome to the Amazon SageMaker pipeline example scenario.")
    println(
        """
         This example workflow will guide you through setting up and running an
         Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and an
         Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
         reverse geocode addresses in an input file and store the results in an
 export file.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("First, we will set up the roles, functions, and queue needed by the
 SageMaker pipeline.")
    val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
    val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
```

```
    val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
    val queueUrl = checkQueue(queueName, functionName)
    println(DASHES)

    println(DASHES)
    println("Setting up bucket $bucketName")
    if (!checkBucket(bucketName)) {
        setupBucket(bucketName)
        println("Put $lnglatData into $bucketName")
        val objectKey = "samplefiles/latlongtest.csv"
        putS3Object(bucketName, objectKey, lnglatData)
    }
    println(DASHES)

    println(DASHES)
    println("Now we can create and run our pipeline.")
    setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn, pipelineName)
    val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
    println("The pipeline execution ARN value is $pipelineExecutionARN")
    waitForPipelineExecution(pipelineExecutionARN)
    println("Wait 30 secs to get output results $bucketName")
    TimeUnit.SECONDS.sleep(30)
    getOutputResults(bucketName)
    println(DASHES)

    println(DASHES)
    println(
        """
            The pipeline has completed. To view the pipeline and runs in SageMaker
Studio, follow these instructions:
            https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-studio.html
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("Do you want to delete the AWS resources used in this Workflow? (y/n)")
    val `in` = Scanner(System.`in`)
    val delResources = `in`.nextLine()
    if (delResources.compareTo("y") == 0) {
        println("Lets clean up the AWS resources. Wait 30 seconds")
        TimeUnit.SECONDS.sleep(30)
```

```
        deleteEventSourceMapping(functionName)
        deleteSQSQueue(queueName)
        listBucketObjects(bucketName)
        deleteBucket(bucketName)
        delLambdaFunction(functionName)
        deleteLambdaRole(lambdaRoleName)
        deleteSagemakerRole(sageMakerRoleName)
        deletePipeline(pipelineName)
    } else {
        println("The AWS Resources were not deleted!")
    }
    println(DASHES)

    println(DASHES)
    println("SageMaker pipeline scenario is complete.")
    println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}

suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
```

```
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun delLambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}

suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
        bucket = bucketNameVal
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
```

```
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
                    eventSourceMapping = event.uuid.toString()
                }
            }
        }
    }

    val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
        uuid = eventSourceMapping
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
        println("The event mapping is deleted!")
    }
}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
```

```
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3Objects: List<Object>? = response.contents
        if (s3Objects != null) {
            for (`object` in s3Objects) {
                if (bucketName != null) {
                    readObject(bucketName, (`object`.key))
                }
            }
        }
    }
}

suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
```

```
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
```

```
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
```

```
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            println("$bucketName exists")
            return true
        }
    } catch (e: S3Exception) {
        println("Bucket does not exist")
    }
    return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
```

```
    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }
    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
 lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
 successful")
    }
}

// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
    val queueAtt: MutableMap<String, String> = HashMap()
    queueAtt.put("DelaySeconds", "5")
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
 { queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
```

```
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a new
 queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
 created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
}

suspend fun createNewFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
```

```
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
 function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val response = lambdaClient.getFunction(functionRequest)
            functionArn = response.configuration?.functionArn.toString()
            println("$functionArn exists")
        }
    } catch (e: LambdaException) {
        println(e.message + " A new function will be created")
        functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
 myHandler, myRole)
    }
    return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
```

```
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in sageMakerRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
```

```
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        System.out.println("Role ready with ARN ${roleResult.role?.arn}")
        return roleResult.role?.arn.toString()
    }
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createLambdaRole(roleNameVal)
    }

    return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
```

```
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        println("Role ready with ARN " + roleResult.role?.arn)
        return roleResult.role?.arn.toString()
    }
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<String>(5)
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy"
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
```

```
    return lambdaRolePolicies
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Exemplos de Secrets Manager usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetSecretValue`

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use
 { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

- Para obter detalhes da API, consulte a GetSecretValue referência da API AWS SDK para Kotlin.

Exemplos do Amazon SES usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon SES.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como usar a API do Amazon DynamoDB para construir uma aplicação Web dinâmica que monitora os dados de trabalho do DynamoDB.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Criar uma aplicação Web para rastrear dados do Amazon Redshift

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho usando um banco de dados do Amazon Redshift.

SDK para Kotlin

Mostra como criar uma aplicação Web que rastreia e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon Redshift.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Redshift e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- banco de dados de origem
- Amazon SES

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Kotlin

Mostra como construir uma aplicação Web que monitora e gera relatórios sobre itens de trabalho armazenados em um banco de dados do Amazon RDS.

Para obter o código-fonte completo e instruções sobre como configurar uma API Spring REST que consulta dados do Amazon Aurora Serverless e para uso por um aplicativo React, veja o exemplo completo em. GitHub

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Exemplos do Amazon SNS usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon SNS

O exemplo de código a seguir mostra como começar a usar o Amazon SNS.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListTopics referência da API AWS SDK para Kotlin.

Ações

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- Para obter detalhes da API, consulte a CreateTopic referência da API AWS SDK para Kotlin.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteTopic referência da API AWS SDK para Kotlin.

`GetTopicAttributes`

O código de exemplo a seguir mostra como usar `GetTopicAttributes`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Para obter detalhes da API, consulte a GetTopicAttributes referência da API AWS SDK para Kotlin.

`ListSubscriptions`

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listSNSSubscriptions() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListSubscriptions referência da API AWS SDK para Kotlin.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listSNSTopics() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListTopics referência da API AWS SDK para Kotlin.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para Kotlin.

`SetTopicAttributes`

O código de exemplo a seguir mostra como usar `SetTopicAttributes`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Para obter detalhes da API, consulte a SetTopicAttributes referência da API AWS SDK para Kotlin.

`Assinar`

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Inscrever uma função do Lambda em um tópico.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }
```

```
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Consulte detalhes da API em Subscribe na Referência da API AWS SDK para Kotlin.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
        Tag {
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request =
        TagResourceRequest {
```

```
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- Para obter detalhes da API, consulte a TagResource referência da API AWS SDK para Kotlin.

Cancelar assinatura

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Consulte detalhes da API em Cancelar assinatura na Referência da API AWS SDK para Kotlin.

Cenários

Criação de uma aplicação do Amazon SNS

O exemplo de código a seguir mostra como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon SNS para criar uma aplicação com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como criar um aplicativo web, veja o exemplo completo em GitHub.

Para obter o código-fonte completo e instruções sobre como criar um aplicativo Android nativo, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK para Kotlin.

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
```

```
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
```

```
                In this scenario, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
```

```
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
 the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
```

```
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
```

```
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)
```

```
    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
```

```
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
```

```
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
```

```
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
```

```
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
```

```
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }
```

```
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
```

```
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Exemplos do Amazon SQS usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon SQS

O exemplo de código a seguir mostra como começar a usar o Amazon SQS.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
```

```
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- Para obter detalhes da API, consulte a ListQueues referência da API AWS SDK para Kotlin.

Ações

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val getQueueUrlRequest =
            GetQueueUrlRequest {
                queueName = queueNameVal
```

```
            }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

- Para obter detalhes da API, consulte a CreateQueue referência da API AWS SDK para Kotlin.

`DeleteMessage`

O código de exemplo a seguir mostra como usar `DeleteMessage`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }
```

```
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Para obter detalhes da API, consulte a DeleteMessage referência da API AWS SDK para Kotlin.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }
```

```
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Para obter detalhes da API, consulte a DeleteQueue referência da API AWS SDK para Kotlin.

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- Para obter detalhes da API, consulte a ListQueues referência da API AWS SDK para Kotlin.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- Para obter detalhes da API, consulte a ReceiveMessage referência da API AWS SDK para Kotlin.

`SendMessage`

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
            messageBody = message
            delaySeconds = 10
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }

    val msg2 =
        SendMessageBatchRequestEntry {
            id = "id2"
            messageBody = "Hello from msg 2"
        }
```

```
    val sendMessageBatchRequest =
        SendMessageBatchRequest {
            queueUrl = queueUrlVal
            entries = listOf(msg1, msg2)
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- Para obter detalhes da API, consulte a SendMessage referência da API AWS SDK para Kotlin.

Cenários

Crie um aplicativo de mensagem

O exemplo de código a seguir mostra como criar uma aplicação de mensagens usando o Amazon SQS.

SDK para Kotlin

Mostra como usar a API do Amazon SQS para desenvolver uma API REST que envia e recupera mensagens.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon SQS

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.

- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
```

```
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
```

```
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)
```

```
    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
 the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
```

```
            "Service": "sns.amazonaws.com"
        },
        "Action": "sqs:SendMessage",
                "Resource": "$sqsQueueArn",
                "Condition": {
            "ArnEquals": {
                "aws:SourceArn": "$topicArn"
            }
        }
    }
    ]
    }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
```

```
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
```

```
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
```

```
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
```

```
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
```

```
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }
```

```
            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
```

```
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
```

```
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")
```

```
            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
```

```
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Exemplos do Step Functions usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com Step Functions.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Step Functions

O exemplo de código a seguir mostra como começar a usar o Step Functions.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
```

```
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListStateMachines referência da API AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma atividade.
- Criar uma máquina de estado a partir de uma definição da Amazon States Language que contenha a atividade criada anteriormente como uma etapa.
- Executar a máquina de estado e responder à atividade com entrada do usuário.
- Obter o status e a saída finais após a conclusão da execução e, em seguida, limpar os recursos.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.

 You can obtain the JSON file to create a state machine in the following GitHub
 location:

 https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files

 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin code example performs the following tasks:
```

```
 1. List activities using a paginator.
 2. List state machines using a paginator.
 3. Creates an activity.
 4. Creates a state machine.
 5. Describes the state machine.
 6. Starts execution of the state machine and interacts with it.
 7. Describes the execution.
 8. Deletes the activity.
 9. Deletes the state machine.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleARN> <activityName> <stateMachineName>

    Where:
        roleName - The name of the IAM role to create for this state machine.
        activityName - The name of an activity to create.
        stateMachineName - The name of the state machine to create.
        jsonFile - The location of the chat_sfn_state_machine.json file. You can
 located it in resources/sample_files.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
    val stateMachineName = args[2]
    val jsonFile = args[3]
    val sc = Scanner(System.`in`)
    var action = false

    val polJSON = """{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }"""

    println(DASHES)
    println("Welcome to the AWS Step Functions example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List activities using a Paginator.")
    listActivitesPagnator()
    println(DASHES)

    println(DASHES)
    println("2. List state machines using a paginator.")
    listStatemachinesPagnator()
    println(DASHES)

    println(DASHES)
    println("3. Create a new activity.")
    val activityArn = createActivity(activityName)
    println("The ARN of the Activity is $activityArn")
    println(DASHES)

    // Get JSON to use for the state machine and place the activityArn value into
it.
    val stream = GetStream()
    val jsonString = stream.getStream(jsonFile)

    // Modify the Resource node.
    val objectMapper = ObjectMapper()
    val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
activityArn)

    // Convert the modified Java object back to a JSON string.
    val stateDefinition = objectMapper.writeValueAsString(root)
    println(stateDefinition)

    println(DASHES)
```

```
    println("4. Create a state machine.")
    val roleARN = createIAMRole(roleName, polJSON)
    val stateMachineArn = createMachine(roleARN, stateMachineName, stateDefinition)
    println("The ARN of the state machine is $stateMachineArn")
    println(DASHES)

    println(DASHES)
    println("5. Describe the state machine.")
    describeStateMachine(stateMachineArn)
    println("What should ChatSFN call you?")
    val userName = sc.nextLine()
    println("Hello $userName")
    println(DASHES)

    println(DASHES)
    // The JSON to pass to the StartExecution call.
    val executionJson = "{ \"name\" : \"$userName\" }"
    println(executionJson)
    println("6. Start execution of the state machine and interact with it.")
    val runArn = startWorkflow(stateMachineArn, executionJson)
    println("The ARN of the state machine execution is $runArn")
    var myList: List<String>
    while (!action) {
        myList = getActivityTask(activityArn)
        println("ChatSFN: " + myList[1])
        println("$userName please specify a value.")
        val myAction = sc.nextLine()
        if (myAction.compareTo("done") == 0) {
            action = true
        }
        println("You have selected $myAction")
        val taskJson = "{ \"action\" : \"$myAction\" }"
        println(taskJson)
        sendTaskSuccess(myList[0], taskJson)
    }
    println(DASHES)

    println(DASHES)
    println("7. Describe the execution.")
    describeExe(runArn)
    println(DASHES)

    println(DASHES)
    println("8. Delete the activity.")
```

```
    deleteActivity(activityArn)
    println(DASHES)

    println(DASHES)
    println("9. Delete the state machines.")
    deleteMachine(stateMachineArn)
    println(DASHES)

    println(DASHES)
    println("The AWS Step Functions example scenario is complete.")
    println(DASHES)
}

suspend fun listStatemachinesPagnator() {
    val machineRequest =
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listStateMachinesPaginated(machineRequest)
            .transform { it.stateMachines?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The state machine ARN is ${obj.stateMachineArn}")
            }
    }
}

suspend fun listActivitesPagnator() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listActivitiesPaginated(activitiesRequest)
            .transform { it.activities?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The activity ARN is ${obj.activityArn}")
            }
    }
}
```

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}

suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}

suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
```

```
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}

suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}

suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
```

```
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}

suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}

suspend fun createIAMRole(
```

```
    roleNameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = roleNameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines

- SendTaskSuccess
- StartExecution
- StopExecution

Ações

`CreateActivity`

O código de exemplo a seguir mostra como usar `CreateActivity`.

SDK para Kotlin

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Para obter detalhes da API, consulte a CreateActivity referência da API AWS SDK para Kotlin.

`CreateStateMachine`

O código de exemplo a seguir mostra como usar `CreateStateMachine`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- Para obter detalhes da API, consulte a CreateStateMachine referência da API AWS SDK para Kotlin.

`DeleteActivity`

O código de exemplo a seguir mostra como usar `DeleteActivity`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- Para obter detalhes da API, consulte a DeleteActivity referência da API AWS SDK para Kotlin.

`DeleteStateMachine`

O código de exemplo a seguir mostra como usar `DeleteStateMachine`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
```

```
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteStateMachine referência da API AWS SDK para Kotlin.

`DescribeExecution`

O código de exemplo a seguir mostra como usar `DescribeExecution`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
```

```
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}
```

- Para obter detalhes da API, consulte a DescribeExecution referência da API AWS SDK para Kotlin.

`DescribeStateMachine`

O código de exemplo a seguir mostra como usar `DescribeStateMachine`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeStateMachine referência da API AWS SDK para Kotlin.

GetActivityTask

O código de exemplo a seguir mostra como usar `GetActivityTask`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

- Para obter detalhes da API, consulte a GetActivityTask referência da API AWS SDK para Kotlin.

ListActivities

O código de exemplo a seguir mostra como usar `ListActivities`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listActivities(activitiesRequest)
        response.activities?.forEach { item ->
            println("The activity ARN is ${item.activityArn}")
            println("The activity name is ${item.name}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListActivities referência da API AWS SDK para Kotlin.

`ListExecutions`

O código de exemplo a seguir mostra como usar `ListExecutions`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getExeHistory(exeARN: String?) {
    val historyRequest =
```

```
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListExecutions referência da API AWS SDK para Kotlin.

`ListStateMachines`

O código de exemplo a seguir mostra como usar `ListStateMachines`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
```

```
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListStateMachines referência da API AWS SDK para Kotlin.

`SendTaskSuccess`

O código de exemplo a seguir mostra como usar `SendTaskSuccess`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
```

```
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- Para obter detalhes da API, consulte a SendTaskSuccess referência da API AWS SDK para Kotlin.

`StartExecution`

O código de exemplo a seguir mostra como usar `StartExecution`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- Para obter detalhes da API, consulte a StartExecution referência da API AWS SDK para Kotlin.

Suporte exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. Suporte

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Suporte

O exemplo de código a seguir mostra como começar a usar o Suporte.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
```

```
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you must have the AWS Business Support Plan to use the AWS Support Java
 API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following task:

1. Gets and displays available services.
 */

suspend fun main() {
    displaySomeServices()
}

// Return a List that contains a Service name and Category name.
suspend fun displaySomeServices() {
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is: " + service.name)

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                index++
            }
```

```
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeServices referência da API AWS SDK para Kotlin.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Obter e exibir os serviços disponíveis e os níveis de gravidade dos casos.
- Criar um caso de suporte usando um serviço, uma categoria e um nível de gravidade selecionados.
- Obter e exibir uma lista de casos em aberto para o dia atual.
- Adicionar um conjunto de anexos e uma comunicação ao novo caso.
- Descrever o novo anexo e a comunicação para o caso.
- Resolver o caso.
- Obtenha e exiba uma lista de casos resolvidos para o dia atual.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
```

```
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
In addition, you must have the AWS Business Support Plan to use the AWS Support Java
 API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following tasks:
1. Gets and displays available services.
2. Gets and displays severity levels.
3. Creates a support case by using the selected service, category, and severity
 level.
4. Gets a list of open cases for the current day.
5. Creates an attachment set with a generated file.
6. Adds a communication with the attachment to the support case.
7. Lists the communications of the support case.
8. Describes the attachment set included with the communication.
9. Resolves the support case.
10. Gets a list of resolved cases for the current day.
*/

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <fileAttachment>
    Where:
         fileAttachment - The file can be a simple saved .txt file to use as an
 email attachment.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val fileAttachment = args[0]
    println("***** Welcome to the AWS Support case example scenario.")
    println("***** Step 1. Get and display available services.")
    val sevCatList = displayServices()

    println("***** Step 2. Get and display Support severity levels.")
    val sevLevel = displaySevLevels()
```

```
    println("***** Step 3. Create a support case using the selected service,
 category, and severity level.")
    val caseIdVal = createSupportCase(sevCatList, sevLevel)
    if (caseIdVal != null) {
        println("Support case $caseIdVal was successfully created!")
    } else {
        println("A support case was not successfully created!")
        exitProcess(1)
    }

    println("***** Step 4. Get open support cases.")
    getOpenCase()

    println("***** Step 5. Create an attachment set with a generated file to add to
 the case.")
    val attachmentSetId = addAttachment(fileAttachment)
    println("The Attachment Set id value is $attachmentSetId")

    println("***** Step 6. Add communication with the attachment to the support
 case.")
    addAttachSupportCase(caseIdVal, attachmentSetId)

    println("***** Step 7. List the communications of the support case.")
    val attachId = listCommunications(caseIdVal)
    println("The Attachment id value is $attachId")

    println("***** Step 8. Describe the attachment set included with the
 communication.")
    describeAttachment(attachId)

    println("***** Step 9. Resolve the support case.")
    resolveSupportCase(caseIdVal)

    println("***** Step 10. Get a list of resolved cases for the current day.")
    getResolvedCase()
    println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
```

```
        DescribeCasesRequest {
            maxResults = 30
            afterTime = yesterday.toString()
            beforeTime = now.toString()
            includeResolvedCases = true
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
```

```
            maxResults = 10
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
```

```
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
```

```
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }
```

```
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Kotlin.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications

- DescribeServices
- DescribeSeverityLevels
- ResolveCase

Ações

`AddAttachmentsToSet`

O código de exemplo a seguir mostra como usar `AddAttachmentsToSet`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- Para obter detalhes da API, consulte a AddAttachmentsToSet referência da API AWS SDK para Kotlin.

`AddCommunicationToCase`

O código de exemplo a seguir mostra como usar `AddCommunicationToCase`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}
```

- Para obter detalhes da API, consulte a AddCommunicationToCase referência da API AWS SDK para Kotlin.

`CreateCase`

O código de exemplo a seguir mostra como usar `CreateCase`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- Para obter detalhes da API, consulte a CreateCase referência da API AWS SDK para Kotlin.

`DescribeAttachment`

O código de exemplo a seguir mostra como usar `DescribeAttachment`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeAttachment referência da API AWS SDK para Kotlin.

`DescribeCases`

O código de exemplo a seguir mostra como usar `DescribeCases`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
```

```
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeCases referência da API AWS SDK para Kotlin.

`DescribeCommunications`

O código de exemplo a seguir mostra como usar `DescribeCommunications`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }
```

```
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}
```

- Para obter detalhes da API, consulte a DescribeCommunications referência da API AWS SDK para Kotlin.

`DescribeServices`

O código de exemplo a seguir mostra como usar `DescribeServices`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
```

```
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Para obter detalhes da API, consulte a DescribeServices referência da API AWS SDK para Kotlin.

`DescribeSeverityLevels`

O código de exemplo a seguir mostra como usar `DescribeSeverityLevels`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}
```

- Para obter detalhes da API, consulte a DescribeSeverityLevels referência da API AWS SDK para Kotlin.

`ResolveCase`

O código de exemplo a seguir mostra como usar `ResolveCase`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- Para obter detalhes da API, consulte a ResolveCase referência da API AWS SDK para Kotlin.

Exemplos do Amazon Translate usando o SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com o Amazon Translate.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criação de uma aplicação do Amazon SNS

O exemplo de código a seguir mostra como criar uma aplicação que oferece funcionalidade de assinatura e publicação e tradução de mensagens.

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon SNS para criar uma aplicação com funcionalidade de assinatura e publicação. Além disso, essa aplicação de exemplo também traduz mensagens.

Para obter o código-fonte completo e instruções sobre como criar um aplicativo web, veja o exemplo completo em GitHub.

Para obter o código-fonte completo e instruções sobre como criar um aplicativo Android nativo, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon SNS
- Amazon Translate

X-Ray exemplos de uso do SDK para Kotlin

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Kotlin com. X-Ray

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateGroup`

O código de exemplo a seguir mostra como usar `CreateGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createNewGroup(groupNameVal: String?) {
    val groupRequest =
        CreateGroupRequest {
            filterExpression = "fault = true AND http.url CONTAINS \"example/game\"
 AND responsetime >= 5"
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val groupResponse = xRayClient.createGroup(groupRequest)
        println("The Group ARN is " + (groupResponse.group?.groupArn))
    }
}
```

- Para obter detalhes da API, consulte a CreateGroup referência da API AWS SDK para Kotlin.

`CreateSamplingRule`

O código de exemplo a seguir mostra como usar `CreateSamplingRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createRule(ruleNameVal: String?) {
    val rule =
        SamplingRule {
            ruleName = ruleNameVal
            priority = 1
            httpMethod = "*"
            serviceType = "*"
            serviceName = "*"
            urlPath = "*"
            version = 1
            host = "*"
            resourceArn = "*"
        }

    val ruleRequest =
        CreateSamplingRuleRequest {
            samplingRule = rule
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val ruleResponse: CreateSamplingRuleResponse =
 xRayClient.createSamplingRule(ruleRequest)
        println("The ARN of the new rule is
 ${ruleResponse.samplingRuleRecord?.samplingRule?.ruleArn}")
    }
}
```

- Para obter detalhes da API, consulte a CreateSamplingRule referência da API AWS SDK para Kotlin.

`DeleteGroup`

O código de exemplo a seguir mostra como usar `DeleteGroup`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteSpecificGroup(groupNameVal: String) {
    val groupRequest =
        DeleteGroupRequest {
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteGroup(groupRequest)
        println("$groupNameVal was deleted!")
    }
}
```

- Para obter detalhes da API, consulte a DeleteGroup referência da API AWS SDK para Kotlin.

`DeleteSamplingRule`

O código de exemplo a seguir mostra como usar `DeleteSamplingRule`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteRule(ruleNameVal: String?) {
```

```
    val ruleRequest =
        DeleteSamplingRuleRequest {
            ruleName = ruleNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteSamplingRule(ruleRequest)
        println("$ruleNameVal was deleted")
    }
}
```

- Para obter detalhes da API, consulte a DeleteSamplingRule referência da API AWS SDK para Kotlin.

`GetGroups`

O código de exemplo a seguir mostra como usar `GetGroups`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getAllGroups() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getGroups(GetGroupsRequest {})
        response.groups?.forEach { group ->
            println("The AWS X-Ray group name is ${group.groupName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetGroups referência da API AWS SDK para Kotlin.

GetSamplingRules

O código de exemplo a seguir mostra como usar `GetSamplingRules`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getRules() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getSamplingRules(GetSamplingRulesRequest {})
        response.samplingRuleRecords?.forEach { record ->
            println("The rule name is ${record.samplingRule?.ruleName}")
            println("The related service is: ${record.samplingRule?.serviceName}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetSamplingRules referência da API AWS SDK para Kotlin.

GetServiceGraph

O código de exemplo a seguir mostra como usar `GetServiceGraph`.

SDK para Kotlin

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun getGraph(groupNameVal: String?) {
```

```
    val time = aws.smithy.kotlin.runtime.time.Instant
    val getServiceGraphRequest =
        GetServiceGraphRequest {
            groupName = groupNameVal
            this.startTime = time.now()
            endTime = time.now()
        }
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getServiceGraph(getServiceGraphRequest)
        response.services?.forEach { service ->
            println("The name of the service is  ${service.name}")
        }
    }
}
```

- Para obter detalhes da API, consulte a GetServiceGraph referência da API AWS SDK para Kotlin.

Exemplos de código para o SDK para PHP.

Os exemplos de código a seguir mostram como usar o AWS SDK para PHP with AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para PHP — Saiba mais sobre como usar o PHP com AWS.
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.

- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos da API Gateway usando o SDK para PHP
- Exemplos do Aurora usando o SDK para PHP
- Exemplos do Auto Scaling usando o SDK para PHP
- Exemplos do Amazon Bedrock usando o SDK para PHP
- Exemplos do Amazon Bedrock Runtime usando o SDK para PHP
- Exemplos do Amazon DocumentDB usando o SDK para PHP
- Exemplos de código do DynamoDB usando o SDK para PHP
- Exemplos do Amazon EC2 usando o SDK para PHP
- AWS Glue exemplos de uso do SDK para PHP
- Exemplos de IAM usando o SDK para PHP
- Exemplos do Kinesis usando o SDK para PHP
- AWS KMS exemplos de uso do SDK para PHP
- Exemplos de Lambda usando SDKs para PHP
- Exemplos do Amazon MSK usando o SDK para PHP
- Exemplos do Amazon RDS usando o SDK para PHP
- Exemplos do Amazon RDS Data Service usando o SDK para PHP
- Exemplos do Amazon Rekognition usando o SDK para PHP
- Exemplos de código do Amazon S3 usando o SDK para PHP
- Exemplos de buckets de diretório do S3 usando o SDK para PHP
- Exemplos do Amazon SES usando o SDK para PHP
- Exemplos de código para o Amazon SNS usando o SDK para PHP
- Exemplos do Amazon SQS usando o SDK para PHP

Exemplos da API Gateway usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com API Gateway.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`GetBasePathMapping`

O código de exemplo a seguir mostra como usar `GetBasePathMapping`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
```

```
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////////// */

function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
        ]);
        return 'The base path mapping\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function getsTheBasePathMapping()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- Para obter detalhes da API, consulte GetBasePathMapping em Referência AWS SDK para PHP da API.

`ListBasePathMappings`

O código de exemplo a seguir mostra como usar `ListBasePathMappings`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 * /////////////////////////////////////////////////////////////////////////// */

function listBasePathMappings($apiGatewayClient, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMappings([
            'domainName' => $domainName
        ]);
        return 'The base path mapping(s) effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
```

```
        return 'Error: ' . $e['message'];
    }
}

function listTheBasePathMappings()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listTheBasePathMappings();
```

- Para obter detalhes da API, consulte ListBasePathMappings em Referência AWS SDK para PHP da API.

`UpdateBasePathMapping`

O código de exemplo a seguir mostra como usar `UpdateBasePathMapping`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ////////////////////////////////////////////////////////////////////////////
```

```
 *
 * Purpose: Updates the base path mapping for a custom domain name
 * in Amazon API Gateway.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 * - $patchOperations: The base path update operations to apply.
 *
 * Returns: Information about the updated base path mapping, if available;
 * otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////// */

function updateBasePathMapping(
    $apiGatewayClient,
    $basePath,
    $domainName,
    $patchOperations
) {
    try {
        $result = $apiGatewayClient->updateBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
            'patchOperations' => $patchOperations
        ]);
        return 'The updated base path\'s URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function updateTheBasePathMapping()
{
    $patchOperations = array([
        'op' => 'replace',
        'path' => '/stage',
        'value' => 'stage2'
    ]);

    $apiGatewayClient = new ApiGatewayClient([
```

```
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo updateBasePathMapping(
        $apiGatewayClient,
        '(none)',
        'example.com',
        $patchOperations
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateTheBasePathMapping();
```

- Para obter detalhes da API, consulte UpdateBasePathMapping em Referência AWS SDK para PHP da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos do Aurora usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Aurora.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para PHP

Mostra como usar o AWS SDK para PHP para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon RDS e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful PHP.

- Integre um aplicativo React.js web com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.

- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Auto Scaling usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com Auto Scaling.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Auto Scaling

O exemplo de código a seguir mostra como começar a usar o Auto Scaling.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function helloService()
{
    $autoScalingClient = new AutoScalingClient([
        'region' => 'us-west-2',
        'version' => 'latest',
        'profile' => 'default',
    ]);

    $groups = $autoScalingClient->describeAutoScalingGroups([]);
    var_dump($groups);
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK para PHP da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo do Amazon EC2 Auto Scaling com um modelo de inicialização e zonas de disponibilidade e obter informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas para CloudWatch métricas e, em seguida, limpe os recursos.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
```

```
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an Auto
 Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you created
 in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );
```

```
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
```

```
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups: {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this->autoScalingGroupName,
 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
```

```
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
         */
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
```

```
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']} recorded:
\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
```

```
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .

- CreateAutoScalingGroup
- DeleteAutoScalingGroup
- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Ações

`CreateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `CreateAutoScalingGroup`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function createAutoScalingGroup(
        $autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    ) {
        return $this->autoScalingClient->createAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'AvailabilityZones' => $availabilityZones,
            'MinSize' => $minSize,
            'MaxSize' => $maxSize,
```

```
            'LaunchTemplate' => [
                'LaunchTemplateId' => $launchTemplateId,
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte CreateAutoScalingGroup em Referência AWS SDK para PHP da API.

`DeleteAutoScalingGroup`

O código de exemplo a seguir mostra como usar `DeleteAutoScalingGroup`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteAutoScalingGroup em Referência AWS SDK para PHP da API.

`DescribeAutoScalingGroups`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingGroups`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function describeAutoScalingGroups($autoScalingGroupNames)
{
    return $this->autoScalingClient->describeAutoScalingGroups([
        'AutoScalingGroupNames' => $autoScalingGroupNames
    ]);
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK para PHP da API.

`DescribeAutoScalingInstances`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function describeAutoScalingInstances($instanceIds)
{
    return $this->autoScalingClient->describeAutoScalingInstances([
        'InstanceIds' => $instanceIds
    ]);
}
```

- Para obter detalhes da API, consulte DescribeAutoScalingInstances em Referência AWS SDK para PHP da API.

`DescribeScalingActivities`

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- Para obter detalhes da API, consulte DescribeScalingActivities em Referência AWS SDK para PHP da API.

`DisableMetricsCollection`

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function disableMetricsCollection($autoScalingGroupName)
```

```
    {
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- Para obter detalhes da API, consulte DisableMetricsCollection em Referência AWS SDK para PHP da API.

`EnableMetricsCollection`

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
    {
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- Para obter detalhes da API, consulte EnableMetricsCollection em Referência AWS SDK para PHP da API.

`SetDesiredCapacity`

O código de exemplo a seguir mostra como usar `SetDesiredCapacity`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
{
    return $this->autoScalingClient->setDesiredCapacity([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'DesiredCapacity' => $desiredCapacity,
    ]);
}
```

- Para obter detalhes da API, consulte SetDesiredCapacity em Referência AWS SDK para PHP da API.

`TerminateInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function terminateInstanceInAutoScalingGroup(
    $instanceId,
    $shouldDecrementDesiredCapacity = true,
    $attempts = 0
) {
    try {
        return $this->autoScalingClient->terminateInstanceInAutoScalingGroup([
            'InstanceId' => $instanceId,
```

```
                'ShouldDecrementDesiredCapacity' => $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
$attempts < 5) {
                error_log("Cannot terminate an instance while it is still pending.
Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- Para obter detalhes da API, consulte TerminateInstanceInAutoScalingGroup em Referência AWS SDK para PHP da API.

`UpdateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `UpdateAutoScalingGroup`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function updateAutoScalingGroup($autoScalingGroupName, $args)
    {
        if (array_key_exists('MaxSize', $args)) {
            $maxSize = ['MaxSize' => $args['MaxSize']];
        } else {
            $maxSize = [];
```

```
        }
        if (array_key_exists('MinSize', $args)) {
            $minSize = ['MinSize' => $args['MinSize']];
        } else {
            $minSize = [];
        }
        $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
        $parameters = array_merge($parameters, $minSize, $maxSize);
        return $this->autoScalingClient->updateAutoScalingGroup($parameters);
    }
```

- Para obter detalhes da API, consulte UpdateAutoScalingGroup em Referência AWS SDK para PHP da API.

Exemplos do Amazon Bedrock usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListFoundationModels`

O código de exemplo a seguir mostra como usar `ListFoundationModels`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Amazon Bedrock disponíveis.

```
public function listFoundationModels()
{
    $bedrockClient = new BedrockClient([
        'region' => 'us-west-2',
        'profile' => 'default'
    ]);
    $response = $bedrockClient->listFoundationModels();
    return $response['modelSummaries'];
}
```

- Para obter detalhes da API, consulte ListFoundationModels em Referência AWS SDK para PHP da API.

Exemplos do Amazon Bedrock Runtime usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP Amazon Bedrock Runtime.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Amazônia Nova
- Stable Image Core

Amazônia Nova

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
// Use the Conversation API to send a text message to Amazon Nova.

use Aws\BedrockRuntime\BedrockRuntimeClient;
use Aws\Exception\AwsException;
use RuntimeException;

class Converse
{
    public function converse(): string
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        $client = new BedrockRuntimeClient([
            'region' => 'us-east-1',
            'profile' => 'default'
        ]);

        // Set the model ID, e.g., Amazon Nova Lite.
        $modelId = 'amazon.nova-lite-v1:0';

        // Start a conversation with the user message.
        $userMessage = "Describe the purpose of a 'hello world' program in one
 line.";
        $conversation = [
            [
                "role" => "user",
```

```
                "content" => [["text" => $userMessage]]
            ]
        ];

        try {
            // Send the message to the model, using a basic inference configuration.
            $response = $client->converse([
                'modelId' => $modelId,
                'messages' => $conversation,
                'inferenceConfig' => [
                    'maxTokens' => 512,
                    'temperature' => 0.5
                ]
            ]);

            // Extract and return the response text.
            $responseText = $response['output']['message']['content'][0]['text'];
            return $responseText;
        } catch (AwsException $e) {
            echo "ERROR: Can't invoke {$modelId}. Reason: {$e-
>getAwsErrorMessage()}";
            throw new RuntimeException("Failed to invoke model: " . $e-
>getAwsErrorMessage(), 0, $e);
        }
    }
}

$demo = new Converse();
echo $demo->converse();
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para PHP .

Stable Image Core

InvokeModel

O exemplo de código a seguir mostra como invocar o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Diffusion.

```
    public function invokeStableDiffusion(string $prompt, int $seed = 0, string
 $aspect_ratio = '1:1')
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and available parameters of Stable Diffusion
 models refer to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
stability-diffusion.html

        $base64_image_data = "";
        try {
            $modelId = 'stability.stable-image-core-v1:1';
            $body = [
                'prompt' => $prompt,
                'aspect_ratio' => $aspect_ratio,
                'seed' => $seed,
                'output_format' => 'png',
            ];

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->images[0];
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- Para obter detalhes da API, consulte InvokeModel em Referência AWS SDK para PHP da API.

Exemplos do Amazon DocumentDB usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando PHP.

```
<?php

require __DIR__.'/vendor/autoload.php';
```

```
use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Exemplos de código do DynamoDB usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Como esse exemplo usa arquivos de suporte, não deixe de ler as orientações no README.md arquivo de exemplos de PHP.

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
```

```
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];
        $attributes = ["rating" =>
            [
                'AttributeName' => 'rating',
                'AttributeType' => 'N',
                'Value' => $rating,
            ],
```

```
            'plot' => [
                'AttributeName' => 'plot',
                'AttributeType' => 'S',
                'Value' => $plot,
            ]
        ];
        $service->updateItemAttributesByKey($tableName, $key, $attributes);
        echo "Movie added and updated.";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);


        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
 {$movie['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
 $rating);

        $movie = $service->getItemByKey($tableName, $key);
        echo "Ok, you have rated {$movie['Item']['title']['S']} as a {$movie['Item']
['rating']['N']}\n";

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
```

```
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - BatchWriteItem

- CreateTable
- DeleteItem
- DeleteTable
- DescribeTable
- GetItem
- PutItem
- Query
- Scan
- UpdateItem

Ações

`BatchExecuteStatement`

O código de exemplo a seguir mostra como usar `BatchExecuteStatement`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
```

```
        ]);
    }

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para PHP da API.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public function writeBatch(string $TableName, array $Batch, int $depth = 2)
   {
       if (--$depth <= 0) {
           throw new Exception("Max depth exceeded. Please try with fewer batch
items or increase depth.");
       }

       $marshal = new Marshaler();
       $total = 0;
       foreach (array_chunk($Batch, 25) as $Items) {
           foreach ($Items as $Item) {
               $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' => ['Item'
=> $marshal->marshalItem($Item)]];
           }
           try {
               echo "Batching another " . count($Items) . " for a total of " .
($total += count($Items)) . " items!\n";
               $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
               $BatchWrite = [];
           } catch (Exception $e) {
               echo "uh oh...";
               echo $e->getMessage();
               die();
           }
           if ($total >= 250) {
               echo "250 movies is probably enough. Right? We can stop there.\n";
               break;
           }
       }
   }
```

- Para obter detalhes da API, consulte BatchWriteItem em Referência AWS SDK para PHP da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Crie uma tabela.

```
        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

    public function createTable(string $tableName, array $attributes)
    {
        $keySchema = [];
        $attributeDefinitions = [];
        foreach ($attributes as $attribute) {
            if (is_a($attribute, DynamoDBAttribute::class)) {
                $keySchema[] = ['AttributeName' => $attribute->AttributeName,
'KeyType' => $attribute->KeyType];
                $attributeDefinitions[] =
                    ['AttributeName' => $attribute->AttributeName, 'AttributeType'
=> $attribute->AttributeType];
            }
        }
```

```
        $this->dynamoDbClient->createTable([
            'TableName' => $tableName,
            'KeySchema' => $keySchema,
            'AttributeDefinitions' => $attributeDefinitions,
            'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,
'WriteCapacityUnits' => 10],
        ]);
    }
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para PHP da API.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

    public function deleteItemByKey(string $tableName, array $key)
```

```
    {
        $this->dynamoDbClient->deleteItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteItem em Referência AWS SDK para PHP da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function deleteTable(string $TableName)
    {
        $this->customWaiter(function () use ($TableName) {
            return $this->dynamoDbClient->deleteTable([
                'TableName' => $TableName,
            ]);
        });
    }
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para PHP da API.

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
$tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para PHP da API.

GetItem

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

    public function getItemByKey(string $tableName, array $key)
    {
        return $this->dynamoDbClient->getItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- Para obter detalhes da API, consulte GetItem em Referência AWS SDK para PHP da API.

ListTables

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function listTables($exclusiveStartTableName = "", $limit = 100)
{
    $this->dynamoDbClient->listTables([
        'ExclusiveStartTableName' => $exclusiveStartTableName,
        'Limit' => $limit,
    ]);
}
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para PHP da API.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    echo "What's the name of the last movie you watched?\n";
    while (empty($movieName)) {
        $movieName = testable_readline("Movie name: ");
    }
    echo "And what year was it released?\n";
    $movieYear = "year";
    while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
        $movieYear = testable_readline("Year released: ");
    }
```

```
        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

    public function putItem(array $array)
    {
        $this->dynamoDbClient->putItem($array);
    }
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK para PHP da API.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
```

```
    public function query(string $tableName, $key)
    {
        $expressionAttributeValues = [];
        $expressionAttributeNames = [];
        $keyConditionExpression = "";
        $index = 1;
        foreach ($key as $name => $value) {
            $keyConditionExpression .= "#" . array_key_first($value) . " = :v
$index,";
            $expressionAttributeNames["#" . array_key_first($value)] =
 array_key_first($value);
            $hold = array_pop($value);
            $expressionAttributeValues[":v$index"] = [
                array_key_first($hold) => array_pop($hold),
            ];
        }
        $keyConditionExpression = substr($keyConditionExpression, 0, -1);
        $query = [
            'ExpressionAttributeValues' => $expressionAttributeValues,
            'ExpressionAttributeNames' => $expressionAttributeNames,
            'KeyConditionExpression' => $keyConditionExpression,
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->query($query);
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para PHP .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $yearsKey = [
```

```
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

    public function scan(string $tableName, array $key, string $filters)
    {
        $query = [
            'ExpressionAttributeNames' => ['#year' => 'year'],
            'ExpressionAttributeValues' => [
                ":min" => ['N' => '1990'],
                ":max" => ['N' => '1999'],
            ],
            'FilterExpression' => "#year between :min and :max",
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->scan($query);
    }
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para PHP .

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

    public function updateItemAttributeByKey(
        string $tableName,
        array $key,
        string $attributeName,
        string $attributeType,
        string $newValue
    ) {
        $this->dynamoDbClient->updateItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
            'UpdateExpression' => "set #NV=:NV",
            'ExpressionAttributeNames' => [
                '#NV' => $attributeName,
            ],
            'ExpressionAttributeValues' => [
                ':NV' => [
                    $attributeType => $newValue
                ]
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para PHP da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
```

```
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQLBatch($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);
        echo "Movie added and updated.\n";
```

```
        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']}
        was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']['S']}
        as a {$movie['Responses'][0]['Item']['rating']['N']}\n";

        $service->deleteItemByPartiQLBatch($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
```

```
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
```

```
            ],
        ]);
    }

    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para PHP da API.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\testable_readline;
use function AwsUtilities\loadMovieData;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
```

```
        echo("-------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQL($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
```

```
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
        echo "Movie added and updated.\n";



        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
 {$movie['Items'][0]['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Items'][0]['title']['S']}?
\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);

        $movie = $service->getItemByPartiQL($tableName, $key);
```

```
        echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
 {$movie['Items'][0]['rating']['N']}\n";

        $service->deleteItemByPartiQL($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
```

```
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
 $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
```

```
        ]);
    }
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para PHP da API.

Exemplos sem servidor

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do DynamoDB com o Lambda usando PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
```

```
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do DynamoDB com o Lambda usando PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
```

```
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Exemplos do Amazon EC2 usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon EC2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateVpc`

O código de exemplo a seguir mostra como usar `CreateVpc`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param string $cidr
     * @return array
     */
    public function createVpc(string $cidr): array
    {
        try {
            $result = $this->ec2Client->createVpc([
                "CidrBlock" => $cidr,
            ]);
            return $result['Vpc'];
        }catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte CreateVpc em Referência AWS SDK para PHP da API.

`CreateVpcEndpoint`

O código de exemplo a seguir mostra como usar `CreateVpcEndpoint`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param string $serviceName
     * @param string $vpcId
     * @param array $routeTableIds
     * @return array
     */
    public function createVpcEndpoint(string $serviceName, string $vpcId, array
 $routeTableIds): array
    {
        try {
            $result = $this->ec2Client->createVpcEndpoint([
                'ServiceName' => $serviceName,
                'VpcId' => $vpcId,
                'RouteTableIds' => $routeTableIds,
            ]);

            return $result["VpcEndpoint"];
        } catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte CreateVpcEndpoint em Referência AWS SDK para PHP da API.

`DeleteVpc`

O código de exemplo a seguir mostra como usar `DeleteVpc`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param string $vpcId
     * @return void
     */
    public function deleteVpc(string $vpcId)
    {
        try {
            $this->ec2Client->deleteVpc([
                "VpcId" => $vpcId,
            ]);
        }catch(Ec2Exception $caught){
            echo "There was a problem deleting the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DeleteVpc em Referência AWS SDK para PHP da API.

`DeleteVpcEndpoints`

O código de exemplo a seguir mostra como usar `DeleteVpcEndpoints`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * @param string $vpcEndpointId
     * @return void
     */
    public function deleteVpcEndpoint(string $vpcEndpointId)
    {
        try {
            $this->ec2Client->deleteVpcEndpoints([
                "VpcEndpointIds" => [$vpcEndpointId],
            ]);
        }catch (Ec2Exception $caught){
            echo "There was a problem deleting the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DeleteVpcEndpoints em Referência AWS SDK para PHP da API.

`DescribeRouteTables`

O código de exemplo a seguir mostra como usar `DescribeRouteTables`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * @param array $routeTableIds
    * @param array $filters
    * @return array
    */
   public function describeRouteTables(array $routeTableIds = [], array $filters =
[]): array
   {
       $parameters = [];
       if($routeTableIds){
           $parameters['RouteTableIds'] = $routeTableIds;
       }
       if($filters){
           $parameters['Filters'] = $filters;
       }
       try {
           $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
$parameters);
           $contents = [];
           foreach ($paginator as $result) {
               foreach ($result['RouteTables'] as $object) {
                   $contents[] = $object['RouteTableId'];
               }
           }
       }catch (Ec2Exception $caught){
           echo "There was a problem paginating the results of DescribeRouteTables:
{$caught->getAwsErrorMessage()}\n";
           throw $caught;
       }
       return $contents;
   }
```

- Para obter detalhes da API, consulte DescribeRouteTables em Referência AWS SDK para PHP da API.

AWS Glue exemplos de uso do SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP with AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");
```

```
        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
```

```
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
```

```
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
```

```
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

```
    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
```

```
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
```

```
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler

- StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- Para obter detalhes da API, consulte CreateCrawler em Referência AWS SDK para PHP da API.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

   public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
   {
       return $this->glueClient->createJob([
           'Name' => $jobName,
           'Role' => $role,
           'Command' => [
               'Name' => 'glueetl',
               'ScriptLocation' => $scriptLocation,
               'PythonVersion' => $pythonVersion,
           ],
           'GlueVersion' => $glueVersion,
       ]);
   }
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK para PHP da API.

DeleteCrawler

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteCrawler em Referência AWS SDK para PHP da API.

DeleteDatabase

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteDatabase em Referência AWS SDK para PHP da API.

`DeleteJob`

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
```

```
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteJob em Referência AWS SDK para PHP da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para PHP da API.

`GetCrawler`

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- Para obter detalhes da API, consulte GetCrawler em Referência AWS SDK para PHP da API.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    $databaseName = "doc-example-database-$uniqid";

    $database = $glueService->getDatabase($databaseName);
    echo "Found a database named " . $database['Database']['Name'] . "\n";

public function getDatabase(string $databaseName): Result
{
    return $this->customWaiter(function () use ($databaseName) {
        return $this->glueClient->getDatabase([
            'Name' => $databaseName,
        ]);
    });
}
```

- Para obter detalhes da API, consulte GetDatabase em Referência AWS SDK para PHP da API.

`GetJobRun`

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    $jobName = 'test-job-' . $uniqid;
```

```
        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- Para obter detalhes da API, consulte GetJobRun em Referência AWS SDK para PHP da API.

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);
```

```
    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- Para obter detalhes da API, consulte GetJobRuns em Referência AWS SDK para PHP da API.

`GetTables`

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Para obter detalhes da API, consulte GetTables em Referência AWS SDK para PHP da API.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para PHP da API.

`StartCrawler`

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Para obter detalhes da API, consulte StartCrawler em Referência AWS SDK para PHP da API.

`StartJobRun`

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";
```

```
        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- Para obter detalhes da API, consulte StartJobRun em Referência AWS SDK para PHP da API.

Exemplos de IAM usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
```

```
echo("--------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("--------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
```

```
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
```

```
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- Para obter detalhes da API, consulte AttachRolePolicy em Referência AWS SDK para PHP da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    /**
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    public function createPolicy(string $policyName, string $policyDocument)
    {
        $result = $this->customWaiter(function () use ($policyName, $policyDocument)
 {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK para PHP da API.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
```

```
    }
```

- Para obter detalhes da API, consulte CreateRole em Referência AWS SDK para PHP da API.

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- Para obter detalhes da API, consulte CreateServiceLinkedRole em Referência AWS SDK para PHP da API.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);

        return $result['User'];
    }
```

- Para obter detalhes da API, consulte CreateUser em Referência AWS SDK para PHP da API.

`GetAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicy em Referência AWS SDK para PHP da API.

`GetPolicy`

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
```

```
    }
```

- Para obter detalhes da API, consulte GetPolicy em Referência AWS SDK para PHP da API.

GetRole

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- Para obter detalhes da API, consulte GetRole em Referência AWS SDK para PHP da API.

ListAttachedRolePolicies

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker =
 "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- Para obter detalhes da API, consulte ListAttachedRolePolicies em Referência AWS SDK para PHP da API.

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
```

```
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- Para obter detalhes da API, consulte ListGroups em Referência AWS SDK para PHP da API.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
```

```
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para PHP da API.

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- Para obter detalhes da API, consulte ListRolePolicies em Referência AWS SDK para PHP da API.

`ListRoles`

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- Para obter detalhes da API, consulte ListRoles em Referência AWS SDK para PHP da API.

`Listar provedores de SAML`

O código de exemplo a seguir mostra como usar `ListSAMLProviders`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK para PHP .

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
```

```
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para PHP da API.

Exemplos do Kinesis usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Kinesis.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data
```

```
            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';
```

```
class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
```

```
}

$logger = new StderrLogger();
return new Handler($logger);
```

AWS KMS exemplos de uso do SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP with AWS KMS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS KMS

O exemplo de código a seguir mostra como começar a usar o AWS Key Management Service.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get the
 keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or to
 break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
}
```

- Para obter detalhes da API, consulte ListKeys em Referência AWS SDK para PHP da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma chave do KMS.
- Listar chaves KMS para sua conta e obter detalhes sobre elas.
- Habilitar e desabilitar chaves do KMS.
- Gerar uma chave de dados simétrica que possa ser usada para criptografia do lado do cliente.
- Gere uma chave assimétrica usada para assinar dados digitalmente.
- Marcar chaves com tags.
- Excluir chaves do KMS.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo "\n";
        echo "-------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management Service using the
 AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service that
 allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys, making
 it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;
        echo "-------------------------------------\n";
        $this->pressEnter();

        $this->kmsClient = new KmsClient([]);
        // Initialize the KmsService class with the client. This allows you to
 override any defaults in the client before giving it to the service class.
        $this->kmsService = new KmsService($this->kmsClient);

        // 1. Create a symmetric KMS key.
        echo "\n";
        echo "1. Create a symmetric KMS key.\n";
```

```
        echo "First, we will create a symmetric KMS key that is used to encrypt and
 decrypt data by invoking createKey().\n";
        $this->pressEnter();

        $key = $this->kmsService->createKey();
        $this->resources['symmetricKey'] = $key['KeyId'];
        echo "Created a customer key with ARN {$key['Arn']}.\n";
        $this->pressEnter();

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code checks
 to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

        $keyInfo = $this->kmsService->describeKey($key['KeyId']);
        if(!$keyInfo['Enabled']){
            echo "The key was not enabled, so we will enable it.\n";
            $this->pressEnter();
            $this->kmsService->enableKey($key['KeyId']);
            echo "The key was successfully enabled.\n";
        }else{
            echo "The key was already enabled, so there was no need to enable it.
\n";
        }
        $this->pressEnter();

        // 3. Encrypt data using the symmetric KMS key.
        echo "\n";
        echo "3. Encrypt data using the symmetric KMS key.\n";
        echo "One of the main uses of symmetric keys is to encrypt and decrypt data.
\n";
        echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
 SYMMETRIC_DEFAULT encryption algorithm.\n";
        $this->pressEnter();
        $text = "Hello, AWS KMS!";
        $encryption = $this->kmsService->encrypt($key['KeyId'], $text);
        echo "The plaintext data was successfully encrypted with the algorithm:
 {$encryption['EncryptionAlgorithm']}.\n";
        $this->pressEnter();

        // 4. Create an alias.
```

```
        echo "\n";
        echo "4. Create an alias.\n";
        $aliasInput = testable_readline("Please enter an alias prefixed with
 \"alias/\" or press enter to use a default value: ");
        if($aliasInput == ""){
            $aliasInput = "alias/dev-encryption-key";
        }
        $this->kmsService->createAlias($key['KeyId'], $aliasInput);
        $this->resources['alias'] = $aliasInput;
        echo "The alias \"$aliasInput\" was successfully created.\n";
        $this->pressEnter();

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
        echo "6. Enable automatic rotation of the KMS key.\n";
        echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days) from
 the enable date and every year
thereafter.";
        $this->pressEnter();
        $this->kmsService->enableKeyRotation($key['KeyId']);
        echo "The key's rotation was successfully set for key: {$key['KeyId']}\n";
        $this->pressEnter();

        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage grants.
```

```
When authorizing access to a KMS key, grants are considered along with key policies
 and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon Web
 Services accounts, IAM users, IAM roles, federated users, and assumed role users.
 For help with the ARN syntax for a principal, see IAM ARNs in the Identity and
 Access Management User Guide. \nTo skip this step, press enter without any other
 values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
 $operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

        // 8. List grants for the KMS key.
        if($granteeARN){
            echo "8. List grants for the KMS key.\n\n";
            $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
            foreach($grantsPaginator as $page){
                foreach($page['Grants'] as $grant){
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

        // 9. Revoke the grant.
        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
```

```
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted earlier
 in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
 $encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are the
 primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in the
 key policy determine who has permission to use the KMS key and how they can use it.
\n";
        echo " You can also use IAM policies and grants to control access to the KMS
 key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
```

```
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

        // 12. Retrieve the key policy.
        echo "\n";
        echo "12. Retrieve the key policy.\n";
        echo "Let's get some information about the new policy and print it to the
screen.\n";
        $this->pressEnter();
        $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
        echo "We got the info! Here is the policy: \n";
        echo $policyInfo['Policy'] . "\n";
        $this->pressEnter();

        // 13. Create an asymmetric KMS key and sign data.
        echo "\n";
        echo "13. Create an asymmetric KMS key and sign data.\n";
        echo "Signing your data with an AWS key can provide several benefits that
make it an attractive option for your data signing needs.\n";
        echo "By using an AWS KMS key, you can leverage the security controls and
compliance features provided by AWS, which can help you meet various regulatory
requirements and enhance the overall security posture of your organization.\n";
        echo "First we'll create the asymmetric key.\n";
        $this->pressEnter();
        $keySpec = "RSA_2048";
        $keyUsage = "SIGN_VERIFY";
        $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
        $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text, $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
$sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
        $this->pressEnter();

        // 14. Tag the symmetric KMS key.
        echo "\n";
        echo "14. Tag the symmetric KMS key.\n";
        echo "By using tags, you can improve the overall management, security,
and governance of your KMS keys, making it easier to organize, track, and control
access to your encrypted data within your AWS environment.\n";
```

```
        echo "Let's tag our symmetric key as Environment->Production\n";
        $this->pressEnter();
        $this->kmsService->tagResource($key['KeyId'], [
            [
                'TagKey' => "Environment",
                'TagValue' => "Production",
            ],
        ]);
        echo "The key was successfully tagged!\n";
        $this->pressEnter();

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
 specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
 changes to PendingDeletion and the key can't be used in any cryptographic
 operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n\n";

        echo "Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted, all data that was encrypted under the KMS key
 is unrecoverable.\n\n";

        $cleanUp = testable_readline("Would you like to delete the resources created
 during this scenario, including the keys? (y/n): ");
        if($cleanUp == "Y" || $cleanUp == "y"){
            $this->cleanUp();
        }

        echo
 "------------------------------------------------------------------------------
\n";
        echo "This concludes the AWS Key Management SDK Basics scenario\n";
        echo
 "------------------------------------------------------------------------------
\n";



namespace Kms;

use Aws\Kms\Exception\KmsException;
```

```
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

    /***
     * @param KmsClient|null $client
     * @param bool $verbose
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }


    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
 $description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
```

```
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
 more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
 "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
```

```
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is not
enabled.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $alias
```

```
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name is
not valid.";
            }
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token is
not valid.\n";
```

```
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
```

```
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
```

```
    }



    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }


    /***
     * @param string $keyId
     * @return Result
     */
    public function getKeyPolicy(string $keyId)
    {
        try {
            return $this->client->getKeyPolicy([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


    /***
     * @param string $grantId
```

```
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param array $tags
     * @return void
```

```
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $rotationPeriodInDays
     * @return void
```

```
     */
    public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
 365)
    {
        try{
            $this->client->enableKeyRotation([
                'KeyId' => $keyId,
                'RotationPeriodInDays' => $rotationPeriodInDays,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $aliasName
     * @return void
```

```
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature, string
 $signingAlgorithm)
    {
        try {
            $result = $this->client->verify([
                'KeyId' => $keyId,
                'Message' => $message,
                'Signature' => $signature,
                'SigningAlgorithm' => $signingAlgorithm,
            ]);
            return $result['SignatureValid'];
        }catch(KmsException $caught){
            echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

Ações

`CreateAlias`

O código de exemplo a seguir mostra como usar `CreateAlias`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name is
not valid.";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte CreateAlias em Referência AWS SDK para PHP da API.

`CreateGrant`

O código de exemplo a seguir mostra como usar `CreateGrant`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
```

```
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token is
not valid.\n";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte CreateGrant em Referência AWS SDK para PHP da API.

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /***
    * @param string $keySpec
    * @param string $keyUsage
    * @param string $description
    * @return array
    */
   public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
   {
       $parameters = ['Description' => $description];
       if($keySpec && $keyUsage){
           $parameters['KeySpec'] = $keySpec;
           $parameters['KeyUsage'] = $keyUsage;
       }
       try {
           $result = $this->client->createKey($parameters);
           return $result['KeyMetadata'];
       }catch(KmsException $caught){
           // Check for error specific to createKey operations
           if ($caught->getAwsErrorMessage() == "LimitExceededException"){
               echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
           }
           throw $caught;
       }
   }
```

- Para obter detalhes da API, consulte CreateKey em Referência AWS SDK para PHP da API.

Decrypt

O código de exemplo a seguir mostra como usar `Decrypt`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
 "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK para PHP .

`DeleteAlias`

O código de exemplo a seguir mostra como usar `DeleteAlias`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DeleteAlias em Referência AWS SDK para PHP da API.

`DescribeKey`

O código de exemplo a seguir mostra como usar `DescribeKey`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DescribeKey em Referência AWS SDK para PHP da API.

`DisableKey`

O código de exemplo a seguir mostra como usar `DisableKey`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
```

```
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte DisableKey em Referência AWS SDK para PHP da API.

`EnableKey`

O código de exemplo a seguir mostra como usar `EnableKey`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
```

```
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte EnableKey em Referência AWS SDK para PHP da API.

`Encrypt`

O código de exemplo a seguir mostra como usar `Encrypt`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is not
enabled.\n";
```

```
            }
            throw $caught;
        }
    }
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK para PHP .

`ListAliases`

O código de exemplo a seguir mostra como usar `ListAliases`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
```

```
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte ListAliases em Referência AWS SDK para PHP da API.

`ListGrants`

O código de exemplo a seguir mostra como usar `ListGrants`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte ListGrants em Referência AWS SDK para PHP da API.

`ListKeys`

O código de exemplo a seguir mostra como usar `ListKeys`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte ListKeys em Referência AWS SDK para PHP da API.

`PutKeyPolicy`

O código de exemplo a seguir mostra como usar `PutKeyPolicy`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte PutKeyPolicy em Referência AWS SDK para PHP da API.

`RevokeGrant`

O código de exemplo a seguir mostra como usar `RevokeGrant`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte RevokeGrant em Referência AWS SDK para PHP da API.

`ScheduleKeyDeletion`

O código de exemplo a seguir mostra como usar `ScheduleKeyDeletion`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte ScheduleKeyDeletion em Referência AWS SDK para PHP da API.

`Sign`

O código de exemplo a seguir mostra como usar `Sign`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Consulte detalhes da API em Sign na Referência da API do AWS SDK para PHP .

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Para obter detalhes da API, consulte TagResource em Referência AWS SDK para PHP da API.

Exemplos de Lambda usando SDKs para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithLambda
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $iamService = new IAMService();
        $s3client = new S3Client($clientArgs);
        $lambdaService = new LambdaService();

        echo "First, let's create a role to run our Lambda code.\n";
        $roleName = "test-lambda-role-$uniqid";
        $rolePolicyDocument = "{
            \"Version\": \"2012-10-17\",
            \"Statement\": [
                {
                    \"Effect\": \"Allow\",
                    \"Principal\": {
                        \"Service\": \"lambda.amazonaws.com\"
                    },
                    \"Action\": \"sts:AssumeRole\"
                }
            ]
        }";
        $role = $iamService->createRole($roleName, $rolePolicyDocument);
        echo "Created role {$role['RoleName']}.\n";
```

```
        $iamService->attachRolePolicy(
            $role['RoleName'],
            "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        );
        echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.\n";

        echo "\nNow let's create an S3 bucket and upload our Lambda code there.\n";
        $bucketName = "amzn-s3-demo-bucket-$uniqid";
        $s3client->createBucket([
            'Bucket' => $bucketName,
        ]);
        echo "Created bucket $bucketName.\n";

        $functionName = "doc_example_lambda_$uniqid";
        $codeBasic = __DIR__ . "/lambda_handler_basic.zip";
        $handler = "lambda_handler_basic";
        $file = file_get_contents($codeBasic);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "Uploaded the Lambda code.\n";

        $createLambdaFunction = $lambdaService->createFunction($functionName, $role,
 $bucketName, $handler);
        // Wait until the function has finished being created.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['State'] == "Pending");
        echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\n";

        sleep(1);

        echo "\nOk, let's invoke that Lambda code.\n";
        $basicParams = [
            'action' => 'increment',
            'number' => 3,
        ];
        /** @var Stream $invokeFunction */
        $invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
```

```
        $result = json_decode($invokeFunction->getContents())->result;
        echo "After invoking the Lambda code with the input of
 {$basicParams['number']} we received $result.\n";

        echo "\nSince that's working, let's update the Lambda code.\n";
        $codeCalculator = "lambda_handler_calculator.zip";
        $handlerCalculator = "lambda_handler_calculator";
        echo "First, put the new code into the S3 bucket.\n";
        $file = file_get_contents($codeCalculator);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "New code uploaded.\n";

        $lambdaService->updateFunctionCode($functionName, $bucketName,
 $functionName);
        // Wait for the Lambda code to finish updating.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "New Lambda code uploaded.\n";

        $environment = [
            'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
        ];
        $lambdaService->updateFunctionConfiguration($functionName,
 $handlerCalculator, $environment);
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for more
 information.\n";

        echo "Invoke the new code with some new data.\n";
        $calculatorParams = [
            'action' => 'plus',
            'x' => 5,
            'y' => 4,
```

```
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $calculatorParams,
"Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does equal
$result.\n";
        echo "Here's the extra debug info: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nBut what happens if you try to divide by zero?\n";
        $divZeroParams = [
            'action' => 'divide',
            'x' => 5,
            'y' => 0,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
"Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "You get a |$result| result.\n";
        echo "And an error message: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nHere's all the Lambda functions you have in this Region:\n";
        $listLambdaFunctions = $lambdaService->listFunctions(5);
        $allLambdaFunctions = $listLambdaFunctions['Functions'];
        $next = $listLambdaFunctions->get('NextMarker');
        while ($next != false) {
            $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
            $next = $listLambdaFunctions->get('NextMarker');
            $allLambdaFunctions = array_merge($allLambdaFunctions,
$listLambdaFunctions['Functions']);
        }
        foreach ($allLambdaFunctions as $function) {
            echo "{$function['FunctionName']}\n";
        }

        echo "\n\nAnd don't forget to clean up your data!\n";

        $lambdaService->deleteFunction($functionName);
        echo "Deleted Lambda function.\n";
        $iamService->deleteRole($role['RoleName']);
        echo "Deleted Role.\n";
        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
```

```
        ]);
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Deleted all objects from the S3 bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);
        echo "Deleted the bucket.\n";
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function createFunction($functionName, $role, $bucketName, $handler)
    {
        //This assumes the Lambda function is in an S3 bucket.
        return $this->customWaiter(function () use ($functionName, $role,
$bucketName, $handler) {
            return $this->lambdaClient->createFunction([
                'Code' => [
                    'S3Bucket' => $bucketName,
                    'S3Key' => $functionName,
                ],
                'FunctionName' => $functionName,
                'Role' => $role['Arn'],
                'Runtime' => 'python3.9',
                'Handler' => "$handler.lambda_handler",
            ]);
        });
    }
```

- Para obter detalhes da API, consulte CreateFunction em Referência AWS SDK para PHP da API.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Para obter detalhes da API, consulte DeleteFunction em Referência AWS SDK para PHP da API.

GetFunction

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para PHP

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
public function getFunction($functionName)
{
    return $this->lambdaClient->getFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- Para obter detalhes da API, consulte GetFunction em Referência AWS SDK para PHP da API.

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para PHP

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function invoke($functionName, $params, $logType = 'None')
    {
        return $this->lambdaClient->invoke([
            'FunctionName' => $functionName,
            'Payload' => json_encode($params),
            'LogType' => $logType,
        ]);
    }
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para PHP .

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function listFunctions($maxItems = 50, $marker = null)
    {
        if (is_null($marker)) {
            return $this->lambdaClient->listFunctions([
                'MaxItems' => $maxItems,
            ]);
        }

        return $this->lambdaClient->listFunctions([
            'Marker' => $marker,
            'MaxItems' => $maxItems,
        ]);
    }
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para PHP da API.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- Para obter detalhes da API, consulte UpdateFunctionCode em Referência AWS SDK para PHP da API.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
```

```
    {
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
            'Environment' => $environment,
        ]);
    }
```

- Para obter detalhes da API, consulte UpdateFunctionConfiguration em Referência AWS SDK para PHP da API.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
```

```
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
```

```
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}
```

```
$logger = new StderrLogger();
return new Handler($logger);
```

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do DynamoDB com o Lambda usando PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
```

```
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando PHP.

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
```

```
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do

bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do S3 com o Lambda usando PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());
```

```
            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
```

```
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}
```

```
$logger = new StderrLogger();
return new Handler($logger);
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do DynamoDB com o Lambda usando PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
```

```
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
```

```
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Exemplos do Amazon MSK usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon MSK.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
```

```
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Exemplos do Amazon RDS usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon RDS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

- Exemplos sem servidor

Ações

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para PHP .

`CreateDBSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBSnapshot`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
```

```
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para PHP .

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
```

```
    echo "\n";
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para PHP .

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
} catch (AwsException $e) {
```

```
    echo $e->getMessage();
    echo "\n";
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para PHP .

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para PHP

Mostra como usar o AWS SDK para PHP para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon RDS e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful PHP.

- Integre um aplicativo React.js web com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
```

```
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
```

```
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Exemplos do Amazon RDS Data Service usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP Amazon RDS Data Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para PHP

Mostra como usar o AWS SDK para PHP para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon RDS e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful PHP.

- Integre um aplicativo React.js web com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon Rekognition usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon Rekognition.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos de código do Amazon S3 usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para PHP da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
```

```
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
```

```
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

```
        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }

        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cópia simples de um objeto.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception->getMessage();
    exit("Please fix error with object copying before continuing.");
}
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK para PHP da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
```

```
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK para PHP da API.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um bucket vazio.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- Para obter detalhes da API, consulte DeleteBucket em Referência AWS SDK para PHP da API.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   public function deleteObject(string $bucketName, string $fileName, array $args =
[])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
       try {
           $this->client->deleteObject($parameters);
           if ($this->verbose) {
               echo "Deleted the object named: $fileName from $bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object deletion before continuing.";
           }
           throw $exception;
       }
   }
```

- Para obter detalhes da API, consulte DeleteObject em Referência AWS SDK para PHP da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um conjunto de objetos de uma lista de chaves.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK para PHP da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um objeto.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK para PHP da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste objetos em um bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK para PHP API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Carregue um objeto em um bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK para PHP da API.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;
```

```
require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the S3
 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in the
 given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();
```

```
namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
```

```
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
```

```
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
```

```
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
 $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
 "MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
```

```
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
=> $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
```

```
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
 error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array $args =
 [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
```

```
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
```

```
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }

}
```

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do S3 com o Lambda usando PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
```

```
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Exemplos de buckets de diretório do S3 usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com S3 Directory Buckets.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Configure uma VPC e um endpoint da VPC.
- Configure as políticas, os perfis e o usuário para trabalhar com os buckets de diretório do S3 e a classe de armazenamento S3 Express One Zone.
- Crie dois clientes do S3.
- Crie dois buckets.
- Crie um objeto e copie-o.
- Demonstre a diferença de desempenho.
- Preencha os buckets para mostrar a diferença lexicográfica.
- Confirme se o usuário deseja limpar os recursos.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário demonstrando os conceitos básicos sobre os buckets de diretório do Amazon S3 e a classe S3 Express One Zone.

```
        echo "\n";
```

```
        echo "--------------------------------------\n";
        echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
        echo "--------------------------------------\n";

        // Change these both of these values to use a different region/availability
 zone.
        $region = "us-west-2";
        $az = "usw2-az1";

        $this->s3Service = new S3Service(new S3Client(['region' => $region]));
        $this->iamService = new IAMService(new IamClient(['region' => $region]));

        $uuid = uniqid();

        echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results in
 this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute resources
 in the same AZ as your Directory
bucket.\n
INTRO;
        pressEnter();
        // 1. Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
        // the need to pass through an internet gateway or NAT device.
        echo "\n";
        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program is
 running in an EC2 instance in the same AZ as your Directory buckets will be.\n";
        $ec2Choice = testable_readline("Are you running this in an EC2 instance
 located in the same AZ as your intended Directory buckets? Enter Y/y to setup a VPC
 Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
 $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);
```

```
            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
 ".s3express";
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
 $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
        echo "Now, we'll set up some policies, roles, and a user. This user will
 only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();
        $file = file_get_contents(__DIR__ . "/../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
        $result = $this->cloudFormationClient->createStack([
            'StackName' => $stackName,
            'TemplateBody' => $file,
            'Capabilities' => ['CAPABILITY_IAM'],
        ]);
        $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
 ['StackName' => $stackName]);
        try {
            $waiter->promise()->wait();
        }catch(CloudFormationException $caught){
            echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
        $this->resources['stackName'] = $stackName;
```

```
        $stackInfo = $this->cloudFormationClient->describeStacks([
            'StackName' => $result['StackId'],
        ]);

        $expressUserName = "";
        $regularUserName = "";
        foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
            if ($output['OutputKey'] == "RegularUser") {
                $regularUserName = $output['OutputValue'];
            }
            if ($output['OutputKey'] == "ExpressUser") {
                $expressUserName = $output['OutputValue'];
            }
        }
        $regularKey = $this->iamService->createAccessKey($regularUserName);
        $regularCredentials = new Credentials($regularKey['AccessKeyId'],
$regularKey['SecretAccessKey']);
        $expressKey = $this->iamService->createAccessKey($expressUserName);
        $expressCredentials = new Credentials($expressKey['AccessKeyId'],
$expressKey['SecretAccessKey']);

        // 3. Create an additional client using the credentials with S3 Express
permissions.
        echo "\n";
        echo "3. Create an additional client using the credentials with S3 Express
permissions.\n";
        echo "This client is created with the credentials associated with the
user account with the S3 Express policy attached, so it can perform S3 Express
operations.\n";
        pressEnter();
        $s3RegularClient = new S3Client([
            'Region' => $region,
            'Credentials' => $regularCredentials,
        ]);
        $s3RegularService = new S3Service($s3RegularClient);
        $s3ExpressClient = new S3Client([
            'Region' => $region,
            'Credentials' => $expressCredentials,
        ]);
        $s3ExpressService = new S3Service($s3ExpressClient);
        echo "All the roles and policies were created an attached to the user. Then,
a new S3 Client and Service were created using that user's credentials.\n";
```

```
        echo "We can now use this client to make calls to S3 Express operations.
Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
Express.\n";
        pressEnter();

        // 4. Create two buckets.
        echo "\n";
        echo "3. Create two buckets.\n";
        echo "Now we will create a Directory bucket, which is the linchpin of the S3
Express One Zone service.\n";
        echo "Directory buckets behave in different ways from regular S3 buckets,
which we will explore here.\n";
        echo "We'll also create a normal bucket, put an object into the normal
bucket, and copy it over to the Directory bucket.\n";
        pressEnter();

        // Create a directory bucket. These are different from normal S3 buckets in
subtle ways.
        $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-s3";
        echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
        pressEnter();
        $s3ExpressService->createBucket($directoryBucketName, [
            'CreateBucketConfiguration' => [
                'Bucket' => [
                    'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
                    'DataRedundancy' => "SingleAvailabilityZone",
                ],
                'Location' => [
                    'Name' => $az,
                    'Type' => "AvailabilityZone",
                ],
            ],
        ]);
        $this->resources['directoryBucketName'] = $directoryBucketName;

        // Create a normal bucket.
        $normalBucketName = "normal-bucket-$uuid";
        $s3RegularService->createBucket($normalBucketName);
        $this->resources['normalBucketName'] = $normalBucketName;
        echo "Great! Both buckets were created.\n";
        pressEnter();
```

```
        // 5. Create an object and copy it over.
        echo "\n";
        echo "5. Create an object and copy it over.\n";
        echo "We'll create a basic object consisting of some text and upload it to
the normal bucket.\n";
        echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
        echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
        pressEnter();

        $objectKey = "basic-text-object";
        $s3RegularService->putObject($normalBucketName, $objectKey, $args = ['Body'
=> "Look Ma, I'm a bucket!"]);
        $this->resources['objectKey'] = $objectKey;

        // Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
        $s3ExpressService->createSession($directoryBucketName);
        $s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

        echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
        echo "Instead of validating permissions on every call as normal buckets do,
Directory buckets utilize the user credentials and session token to validate.\n";
        echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot of
time saved.\n";
        pressEnter();

        // 6. Demonstrate performance difference.
        echo "\n";
        echo "6. Demonstrate performance difference.\n";
        $downloads = 1000;
        echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
        $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
        if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
```

```
            $downloads = $downloadChoice;
        }

        // Download the object $downloads times from each bucket and time it to
 demonstrate the speed difference.
        $directoryStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3ExpressService->getObject($directoryBucketName, $objectKey);
        }
        $directoryEndTime = hrtime(true);
        $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

        $normalStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3RegularService->getObject($normalBucketName, $objectKey);
        }
        $normalEndTime = hrtime(true);
        $normalTimeDiff = $normalEndTime - $normalStartTime;

        echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
 normal bucket took $normalTimeDiff.\n";
        echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
 " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
 seconds.\n";
        pressEnter();

        // 7. Populate the buckets to show the lexicographical difference.
        echo "\n";
        echo "7. Populate the buckets to show the lexicographical difference.\n";
        echo "Now let's explore how Directory buckets store objects in a different
 manner to regular buckets.\n";
        echo "The key is in the name \"Directory!\"\n";
        echo "Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual directories/folders.\n";
        echo "This allows for more rapid indexing, traversing, and therefore
 retrieval times!\n";
        echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
        echo "This structural difference also causes ListObjects to behave
 differently, which can cause unexpected results.\n";
        echo "Let's add a few more objects with layered directories as see how the
 output of ListObjects changes.\n";
        pressEnter();
```

```
        // Populate a few more files in each bucket so that we can use ListObjects
and show the difference.
        $otherObject = "other/$objectKey";
        $altObject = "alt/$objectKey";
        $otherAltObject = "other/alt/$objectKey";
        $s3ExpressService->putObject($directoryBucketName, $otherObject);
        $s3RegularService->putObject($normalBucketName, $otherObject);
        $this->resources['otherObject'] = $otherObject;
        $s3ExpressService->putObject($directoryBucketName, $altObject);
        $s3RegularService->putObject($normalBucketName, $altObject);
        $this->resources['altObject'] = $altObject;
        $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
        $s3RegularService->putObject($normalBucketName, $otherAltObject);
        $this->resources['otherAltObject'] = $otherAltObject;

        $listDirectoryBucket = $s3ExpressService->listObjects($directoryBucketName);
        $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

        // Directory bucket content
        echo "Directory bucket content\n";
        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical order,
while the directory bucket does not. This is because the normal bucket considers
the whole \"key\" to be the object identifies, while the directory bucket actually
creates directories and uses the object \"key\" as a path to the object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }
```

```
namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
```

```
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
```

```
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
```

```
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
 $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
 "MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
```

```
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
=> $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
```

```
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
 error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array $args =
 [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
```

```
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
```

```
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }

}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Exemplos do Amazon SES usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon SES.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para PHP

Mostra como usar o AWS SDK para PHP para criar um aplicativo web que rastreia itens de trabalho em um banco de dados do Amazon RDS e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa um front-end criado com React.js para interagir com um back-end RESTful PHP.

- Integre um aplicativo React.js web com AWS serviços.
- Liste, adicione, atualize e exclua itens em uma tabela do Amazon RDS.
- Envie um relatório por e-mail dos itens de trabalho filtrados usando o Amazon SES.
- Implante e gerencie recursos de exemplo com o AWS CloudFormation script incluído.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora

- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos de código para o Amazon SNS usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários
- Exemplos sem servidor

Ações

`CheckIfPhoneNumberIsOptedOut`

O código de exemplo a seguir mostra como usar `CheckIfPhoneNumberIsOptedOut`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte CheckIfPhoneNumberIsOptedOut em Referência AWS SDK para PHP da API.

`ConfirmSubscription`

O código de exemplo a seguir mostra como usar `ConfirmSubscription`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte ConfirmSubscription em Referência AWS SDK para PHP da API.

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the requested
 region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
```

```
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte CreateTopic em Referência AWS SDK para PHP da API.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
```

```
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte DeleteTopic em Referência AWS SDK para PHP da API.

`GetSMSAttributes`

O código de exemplo a seguir mostra como usar `GetSMSAttributes`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Consulte detalhes da API em GetSMSAttributes na Referência da API AWS SDK para PHP .

`GetTopicAttributes`

O código de exemplo a seguir mostra como usar `GetTopicAttributes`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte GetTopicAttributes em Referência AWS SDK para PHP da API.

`ListPhoneNumbersOptedOut`

O código de exemplo a seguir mostra como usar `ListPhoneNumbersOptedOut`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages from
 your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para obter detalhes da API, consulte ListPhoneNumbersOptedOut em Referência AWS SDK para PHP da API.

`ListSubscriptions`

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte ListSubscriptions em Referência AWS SDK para PHP da API.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the region
 specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para PHP da API.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Consulte detalhes da API em Publish na Referência da API AWS SDK para PHP .

`SetSMSAttributes`

O código de exemplo a seguir mostra como usar `SetSMSAttributes`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Para receber detalhes da API, consulte SetSMSAttributes na Referência da API AWS SDK para PHP .

`SetTopicAttributes`

O código de exemplo a seguir mostra como usar `SetTopicAttributes`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte SetTopicAttributes em Referência AWS SDK para PHP da API.

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';
```

```
try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Inscrever um endpoint HTTP em um tópico.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
```

```
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter detalhes da API, consulte Subscribe na Referência da API AWS SDK para PHP .

Cancelar assinatura

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
```

```
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Consulte detalhes da API em Unsubscribe na Referência da API do AWS SDK para PHP .

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para PHP

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
```

```
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para PHP.
- Consulte detalhes da API em Publish na Referência da API AWS SDK para PHP .

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
```

```
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Exemplos do Amazon SQS usando o SDK para PHP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para PHP com o Amazon SQS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para PHP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
```

```
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Exemplos de código para Tools for PowerShell V5

Os exemplos de código a seguir mostram como usar o Ferramentas da AWS para PowerShell V5 com AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor de ferramentas para PowerShell V5 — Saiba mais sobre como usar PowerShell com AWS.
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos de ACM usando ferramentas para PowerShell V5
- Exemplos de escalonamento automático de aplicativos usando ferramentas para PowerShell V5
- WorkSpaces Exemplos de aplicativos usando Tools for PowerShell V5
- Exemplos do Aurora usando ferramentas para PowerShell V5
- Exemplos de Auto Scaling usando Tools for PowerShell V5
- AWS Budgets exemplos usando ferramentas para PowerShell V5
- AWS Cloud9 exemplos usando ferramentas para PowerShell V5
- CloudFormation exemplos de uso de ferramentas para PowerShell V5

- CloudFront exemplos de uso de ferramentas para PowerShell V5
- CloudTrail exemplos de uso de ferramentas para PowerShell V5
- CloudWatch exemplos de uso de ferramentas para PowerShell V5
- CodeCommit exemplos de uso de ferramentas para PowerShell V5
- CodeDeploy exemplos de uso de ferramentas para PowerShell V5
- CodePipeline exemplos de uso de ferramentas para PowerShell V5
- Exemplos de identidade do Amazon Cognito usando ferramentas para PowerShell V5
- AWS Config exemplos de uso de ferramentas para PowerShell V5
- Exemplos de Device Farm usando Tools for PowerShell V5
- Directory Service exemplos usando ferramentas para PowerShell V5
- AWS DMS exemplos usando ferramentas para PowerShell V5
- Exemplos do DynamoDB usando ferramentas para V5 PowerShell
- Exemplos do Amazon EC2 usando ferramentas para PowerShell V5
- Exemplos do Amazon ECR usando ferramentas para PowerShell V5
- Exemplos do Amazon ECS usando ferramentas para PowerShell V5
- Exemplos do Amazon EFS usando ferramentas para PowerShell V5
- Exemplos do Amazon EKS usando ferramentas para PowerShell V5
- Elastic Load Balancing - exemplos da versão 1 usando ferramentas para PowerShell V5
- Elastic Load Balancing - exemplos da versão 2 usando ferramentas para PowerShell V5
- Exemplos do Amazon FSx usando ferramentas para PowerShell V5
- Exemplos do Amazon Glacier usando ferramentas para PowerShell V5
- AWS Glue exemplos usando ferramentas para PowerShell V5
- AWS Health exemplos usando ferramentas para PowerShell V5
- Exemplos de IAM usando ferramentas para PowerShell V5
- Exemplos do Kinesis usando ferramentas para PowerShell V5
- Exemplos do Lambda usando ferramentas para PowerShell V5
- Exemplos de Amazon ML usando ferramentas para PowerShell V5
- Exemplos de Macie usando Ferramentas para PowerShell V5
- AWS Price List exemplos usando ferramentas para PowerShell V5
- Exemplos de grupos de recursos usando ferramentas para PowerShell V5

- Exemplos de API de marcação de grupos de recursos usando ferramentas para PowerShell V5
- Exemplos do Route 53 usando ferramentas para PowerShell V5
- Exemplos do Amazon S3 usando ferramentas para PowerShell V5
- Exemplos de CSPM do Security Hub usando ferramentas para V5 PowerShell
- Exemplos do Amazon SES usando ferramentas para PowerShell V5
- Exemplos da API Amazon SES v2 usando ferramentas para PowerShell V5
- Exemplos do Amazon SNS usando ferramentas para PowerShell V5
- Exemplos do Amazon SQS usando ferramentas para PowerShell V5
- AWS STS exemplos usando ferramentas para PowerShell V5
- Suporte exemplos usando ferramentas para PowerShell V5
- Exemplos do Systems Manager usando ferramentas para PowerShell V5
- Exemplos do Amazon Translate usando ferramentas para PowerShell V5
- AWS WAFV2 exemplos usando ferramentas para PowerShell V5
- WorkSpaces exemplos de uso de ferramentas para PowerShell V5

Exemplos de ACM usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o ACM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-ACMCertificate`

O código de exemplo a seguir mostra como usar `Get-ACMCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como retornar um certificado e sua cadeia usando o ARN do certificado.

```
Get-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Para obter detalhes sobre a API, consulte GetCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ACMCertificateDetail`

O código de exemplo a seguir mostra como usar `Get-ACMCertificateDetail`.

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes do certificado especificado.

```
Get-ACMCertificateDetail -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

Saída:

```
CertificateArn          : arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
CreatedAt               : 1/21/2016 5:55:59 PM
DomainName              : www.example.com
DomainValidationOptions : {www.example.com}
InUseBy                 : {}
IssuedAt                : 1/1/0001 12:00:00 AM
Issuer                  :
KeyAlgorithm            : RSA-2048
NotAfter                : 1/1/0001 12:00:00 AM
NotBefore               : 1/1/0001 12:00:00 AM
RevocationReason        :
RevokedAt               : 1/1/0001 12:00:00 AM
Serial                  :
SignatureAlgorithm      : SHA256WITHRSA
Status                  : PENDING_VALIDATION
Subject                 : CN=www.example.com
```

```
SubjectAlternativeNames : {www.example.net}
```

- Para obter detalhes sobre a API, consulte DescribeCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ACMCertificateList`

O código de exemplo a seguir mostra como usar `Get-ACMCertificateList`.

Ferramentas para PowerShell V5

Exemplo 1: recupera uma lista todos os ARNs de certificado e o nome de domínio de cada. O cmdlet fará a paginação automaticamente para recuperar todos os ARNs. Para controlar manualmente a paginação, use o MaxItem parâmetro - para controlar quantos ARNs certificados são retornados para cada chamada de serviço e o NextToken parâmetro - para indicar o ponto inicial de cada chamada.

```
Get-ACMCertificateList
```

Saída:

```
CertificateArn
 DomainName
--------------
 ----------
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
 www.example.com
```

Exemplo 2: recupera uma lista de todos os ARNs do certificado em que o status corresponde aos estados fornecidos.

```
Get-ACMCertificateList -CertificateStatus "VALIDATION_TIMED_OUT","FAILED"
```

Exemplo 3: esse exemplo retorna uma lista de todos os certificados na região us-east-1 que têm um tipo de chave RSA_2048 e um uso estendido da chave, ou uma finalidade, de CODE_SIGNING. Você pode encontrar os valores desses parâmetros de filtragem no tópico de referência da API ListCertificates Filters: https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html.

```
Get-ACMCertificateList -Region us-east-1 -Includes_KeyType RSA_2048 -
Includes_ExtendedKeyUsage CODE_SIGNING
```

Saída:

```
CertificateArn
 DomainName
--------------
 ----------
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-d7c0-48c1-af8d-2133d8f30zzz
 *.route53docs.com
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-98a5-443d-a734-800430c80zzz
 nerdzizm.net
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-2be6-4376-8fa7-bad559525zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-e7ca-44c5-803e-24d9f2f36zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-1241-4b71-80b1-090305a62zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-8709-4568-8c64-f94617c99zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-a8fa-4a61-98cf-e08ccc0eezzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-fa47-40fe-a714-2d277d3eezzz
 *.route53docs.com
```

- Para obter detalhes sobre a API, consulte ListCertificates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ACMCertificate`

O código de exemplo a seguir mostra como usar `New-ACMCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: cria um certificado. O serviço retorna o ARN do novo certificado.

```
New-ACMCertificate -DomainName "www.example.com"
```

Saída:

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

Exemplo 2: cria um certificado. O serviço retorna o ARN do novo certificado.

```
New-ACMCertificate -DomainName "www.example.com" -SubjectAlternativeName
 "example.com","www.example.net"
```

Saída:

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

- Para obter detalhes sobre a API, consulte RequestCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ACMCertificate`

O código de exemplo a seguir mostra como usar `Remove-ACMCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: exclui o certificado identificado pelo ARN fornecido e pela chave privada associada. O cmdlet solicitará confirmação antes de prosseguir. Adicione a opção -Force para ignorar a confirmação.

```
Remove-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Para obter detalhes sobre a API, consulte DeleteCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Send-ACMValidationEmail`

O código de exemplo a seguir mostra como usar `Send-ACMValidationEmail`.

Ferramentas para PowerShell V5

Exemplo 1: solicita que o e-mail para validar a propriedade do domínio para “www.example.com” seja enviado. Se o $ do seu shell ConfirmPreference estiver definido como 'Médio' ou inferior,

o cmdlet solicitará confirmação antes de continuar. Adicione a opção -Force para ignorar as solicitações de confirmação.

```
$params = @{
    CertificateArn="arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
    Domain="www.example.com"
    ValidationDomain="example.com"
}
Send-ACMValidationEmail @params
```

- Para obter detalhes sobre a API, consulte ResendValidationEmail Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de escalonamento automático de aplicativos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Application Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-AASScalableTarget`

O código de exemplo a seguir mostra como usar `Add-AASScalableTarget`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet registra ou atualiza um destino escalável. Um destino escalável é um recurso cuja escala pode ser aumentada ou reduzida horizontalmente pelo Application Auto Scaling.

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- Para obter detalhes sobre a API, consulte RegisterScalableTarget Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-AASScalableTarget`

O código de exemplo a seguir mostra como usar `Get-AASScalableTarget`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo fornecerá informações sobre os destinos escaláveis do Application Auto Scaling no namespace especificado.

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

Saída:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- Para obter detalhes sobre a API, consulte DescribeScalableTargets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-AASScalingActivity`

O código de exemplo a seguir mostra como usar `Get-AASScalingActivity`.

Ferramentas para PowerShell V5

Exemplo 1: fornece informações descritivas sobre as atividades de escalabilidade no namespace especificado das últimas seis semanas.

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

Saída:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in state
 ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- Para obter detalhes sobre a API, consulte DescribeScalingActivities Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-AASScalingPolicy`

O código de exemplo a seguir mostra como usar `Get-AASScalingPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet descreve as políticas de escalabilidade do Application Auto Scaling para o namespace de serviço especificado.

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

Saída:

```
Alarms                                  : {Appstream2-LabFleet-default-scale-out-
Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-out
PolicyName                              : default-scale-out
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                  : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-in
PolicyName                              : default-scale-in
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- Para obter detalhes sobre a API, consulte DescribeScalingPolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-AASScheduledAction`

O código de exemplo a seguir mostra como usar `Get-AASScheduledAction`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet lista as ações programadas para o grupo do Auto Scaling que ainda não foram executadas ou que ainda não atingiram o horário de término.

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

Saída:

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- Para obter detalhes sobre a API, consulte DescribeScheduledActions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-AASScalableTarget`

O código de exemplo a seguir mostra como usar `Remove-AASScalableTarget`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cancela o registro de um destino escalável do Application Auto Scaling. O cancelamento do registro de um destino escalável exclui as políticas de escalabilidade associadas a ele.

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeregisterScalableTarget Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-AASScalingPolicy`

O código de exemplo a seguir mostra como usar `Remove-AASScalingPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui a política de escalabilidade especificada para um destino escalável do Application Auto Scaling.

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-out"
 -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- Para obter detalhes sobre a API, consulte DeleteScalingPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-AASScheduledAction`

O código de exemplo a seguir mostra como usar `Remove-AASScheduledAction`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui a ação programada especificada para um destino escalável do Application Auto Scaling.

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteScheduledAction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-AASScalingPolicy`

O código de exemplo a seguir mostra como usar `Set-AASScalingPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cria ou atualiza uma política para um destino escalável do Application Auto Scaling. Cada destino escalável é identificado por um namespace de serviço, ID de recurso e dimensão escalável.

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

Saída:

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- Para obter detalhes sobre a API, consulte PutScalingPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-AASScheduledAction`

O código de exemplo a seguir mostra como usar `Set-AASScheduledAction`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cria ou atualiza uma ação programada para um destino escalável do Application Auto Scaling. Cada destino escalável é identificado por um namespace de serviço, ID de recurso e dimensão escalável.

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- Para obter detalhes sobre a API, consulte PutScheduledAction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

WorkSpaces Exemplos de aplicativos usando Tools for PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com WorkSpaces aplicativos.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-APSResourceTag`

O código de exemplo a seguir mostra como usar `Add-APSResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo adiciona uma tag de recurso ao AppStream recurso

```
Add-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest -Tag @{StackState='Test'} -Select ^Tag
```

Saída:

```
Name                           Value
----                           -----
StackState                     Test
```

- Para obter detalhes sobre a API, consulte TagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Copy-APSImage`

O código de exemplo a seguir mostra como usar `Copy-APSImage`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra copia uma imagem para outra região

```
Copy-APSImage -DestinationImageName TestImageCopy -DestinationRegion us-west-2 -
SourceImageName Powershell
```

Saída:

```
TestImageCopy
```

- Para obter detalhes sobre a API, consulte CopyImage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-APSUSer`

O código de exemplo a seguir mostra como usar `Disable-APSUSer`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra desativa um usuário em USERPOOL

```
Disable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- Para obter detalhes sobre a API, consulte DisableUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-APSUser`

O código de exemplo a seguir mostra como usar `Enable-APSUser`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra ativa um usuário desativado em USERPOOL

```
Enable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- Para obter detalhes sobre a API, consulte EnableUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSAssociatedFleetList`

O código de exemplo a seguir mostra como usar `Get-APSAssociatedFleetList`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exibe a frota associada a uma pilha

```
Get-APSAssociatedFleetList -StackName PowershellStack
```

Saída:

```
PowershellFleet
```

- Para obter detalhes sobre a API, consulte ListAssociatedFleets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSAssociatedStackList`

O código de exemplo a seguir mostra como usar `Get-APSAssociatedStackList`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exibe a frota associada a uma frota

```
Get-APSAssociatedStackList -FleetName PowershellFleet
```

Saída:

```
PowershellStack
```

- Para obter detalhes sobre a API, consulte ListAssociatedStacks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSDirectoryConfigList`

O código de exemplo a seguir mostra como usar `Get-APSDirectoryConfigList`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exibe as configurações de diretório criadas no AppStream

```
Get-APSDirectoryConfigList | Select DirectoryName,
 OrganizationalUnitDistinguishedNames, CreatedTime
```

Saída:

```
DirectoryName OrganizationalUnitDistinguishedNames CreatedTime
------------- ------------------------------------ -----------
Test.com      {OU=AppStream,DC=Test,DC=com}    9/6/2019 10:56:40 AM
contoso.com   {OU=AppStream,OU=contoso,DC=contoso,DC=com}  8/9/2019 9:08:50 AM
```

- Para obter detalhes sobre a API, consulte DescribeDirectoryConfigs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSFleetList`

O código de exemplo a seguir mostra como usar `Get-APSFleetList`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exibe detalhes de uma frota

```
Get-APSFleetList -Name Test
```

Saída:

```
Arn                            : arn:aws:appstream:us-east-1:1234567890:fleet/Test
ComputeCapacityStatus          : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                    : 9/12/2019 5:00:45 PM
Description                    : Test
DisconnectTimeoutInSeconds     : 900
DisplayName                    : Test
DomainJoinInfo                 :
EnableDefaultInternetAccess    : False
FleetErrors                    : {}
FleetType                      : ON_DEMAND
IamRoleArn                     :
IdleDisconnectTimeoutInSeconds : 900
ImageArn                       : arn:aws:appstream:us-east-1:1234567890:image/Test
ImageName                      : Test
InstanceType                   : stream.standard.medium
MaxUserDurationInSeconds       : 57600
Name                           : Test
State                          : STOPPED
VpcConfig                      : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes sobre a API, consulte DescribeFleets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSImageBuilderList`

O código de exemplo a seguir mostra como usar `Get-APSImageBuilderList`.

Ferramentas para PowerShell V5

Exemplo 1: Esta amostra exibe detalhes de um ImageBuilder

```
Get-APSImageBuilderList -Name TestImage
```

Saída:

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
```

```
Arn                          : arn:aws:appstream:us-east-1:1234567890:image-builder/
TestImage
CreatedTime                  : 1/14/2019 4:33:05 AM
Description                  :
DisplayName                  : TestImage
DomainJoinInfo               :
EnableDefaultInternetAccess  : False
IamRoleArn                   :
ImageArn                     : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors           : {}
InstanceType                 : stream.standard.large
Name                         : TestImage
NetworkAccessConfiguration   : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                     : WINDOWS
State                        : STOPPED
StateChangeReason            :
VpcConfig                    : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes sobre a API, consulte DescribeImageBuilders Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSImageList`

O código de exemplo a seguir mostra como usar `Get-APSImageList`.

Ferramentas para PowerShell V5

Exemplo 1: Esta amostra exibe AppStream imagens privadas

```
Get-APSImageList -Type PRIVATE | select DisplayName, ImageBuilderName, Visibility,
 arn
```

Saída:

```
DisplayName          ImageBuilderName      Visibility Arn
-----------          ----------------      ---------- ---
OfficeApps           OfficeApps            PRIVATE    arn:aws:appstream:us-
east-1:123456789012:image/OfficeApps
SessionScriptV2      SessionScriptTest     PRIVATE    arn:aws:appstream:us-
east-1:123456789012:image/SessionScriptV2
```

- Para obter detalhes sobre a API, consulte DescribeImages Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSImagePermission`

O código de exemplo a seguir mostra como usar `Get-APSImagePermission`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exibe permissões de imagem em uma AppStream imagem compartilhada

```
Get-APSImagePermission -Name Powershell  | select SharedAccountId,
 @{n="AllowFleet";e={$_.ImagePermissions.AllowFleet}},
 @{n="AllowImageBuilder";e={$_.ImagePermissions.AllowImageBuilder}}
```

Saída:

```
SharedAccountId AllowFleet AllowImageBuilder
--------------- ---------- -----------------
123456789012          True              True
```

- Para obter detalhes sobre a API, consulte DescribeImagePermissions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSSessionList`

O código de exemplo a seguir mostra como usar `Get-APSSessionList`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exibe a lista de sessões de uma frota

```
Get-APSSessionList -FleetName PowershellFleet -StackName PowershellStack
```

Saída:

```
AuthenticationType         : API
ConnectionState            : CONNECTED
FleetName                  : PowershellFleet
Id                         : d8987c70-4394-4324-a396-2d485c26f2a2
```

```
MaxExpirationTime          : 12/27/2019 4:54:07 AM
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
StackName                  : PowershellStack
StartTime                  : 12/26/2019 12:54:12 PM
State                      : ACTIVE
UserId                     : Test
```

- Para obter detalhes sobre a API, consulte DescribeSessions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSStackList`

O código de exemplo a seguir mostra como usar `Get-APSStackList`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exibe a lista de AppStream Stack

```
Get-APSStackList | Select DisplayName, Arn, CreatedTime
```

Saída:

```
DisplayName                   Arn
                        CreatedTime
-----------                   ---
                        -----------
PowershellStack               arn:aws:appstream:us-east-1:123456789012:stack/
PowershellStack               4/24/2019 8:49:29 AM
SessionScriptTest             arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest             9/12/2019 3:23:12 PM
```

- Para obter detalhes sobre a API, consulte DescribeStacks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSTagsForResourceList`

O código de exemplo a seguir mostra como usar `Get-APSTagsForResourceList`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exibe tags em um AppStream recurso

```
Get-APSTagsForResourceList -ResourceArn arn:aws:appstream:us-
east-1:123456789012:stack/SessionScriptTest
```

Saída:

```
Key        Value
---        -----
StackState Test
```

- Para obter detalhes sobre a API, consulte ListTagsForResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSUsageReportSubscription`

O código de exemplo a seguir mostra como usar `Get-APSUsageReportSubscription`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exibe detalhes AppStreamUsageReport da configuração

```
Get-APSUsageReportSubscription
```

Saída:

```
LastGeneratedReportDate S3BucketName                                    Schedule
 SubscriptionErrors
----------------------- ------------                                    --------
 ------------------
1/1/0001 12:00:00 AM    appstream-logs-us-east-1-123456789012-sik1hnxe DAILY    {}
```

- Para obter detalhes sobre a API, consulte DescribeUsageReportSubscriptions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSUser`

O código de exemplo a seguir mostra como usar `Get-APSUser`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exibe a lista de usuários com status ativado

```
Get-APSUser -AuthenticationType USERPOOL | Select-Object UserName,
 AuthenticationType, Enabled
```

Saída:

```
UserName                   AuthenticationType Enabled
--------                   ------------------ -------
foo1@contoso.com USERPOOL              True
foo2@contoso.com        USERPOOL              True
foo3@contoso.com    USERPOOL              True
foo4@contoso.com   USERPOOL              True
foo5@contoso.com         USERPOOL              True
```

- Para obter detalhes sobre a API, consulte DescribeUsers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-APSUserStackAssociation`

O código de exemplo a seguir mostra como usar `Get-APSUserStackAssociation`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exibe a lista de usuários atribuídos a uma pilha

```
Get-APSUserStackAssociation -StackName PowershellStack
```

Saída:

```
AuthenticationType SendEmailNotification StackName       UserName
------------------ --------------------- ---------       --------
USERPOOL           False                 PowershellStack TestUser1@lab.com
USERPOOL           False                 PowershellStack TestUser2@lab.com
```

- Para obter detalhes sobre a API, consulte DescribeUserStackAssociations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-APSDirectoryConfig`

O código de exemplo a seguir mostra como usar `New-APSDirectoryConfig`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma configuração de diretório no AppStream

```
New-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso\ServiceAccount
 -ServiceAccountCredentials_AccountPassword MyPass -DirectoryName contoso.com -
OrganizationalUnitDistinguishedName "OU=AppStream,OU=Contoso,DC=Contoso,DC=com"
```

Saída:

```
CreatedTime            DirectoryName OrganizationalUnitDistinguishedNames
 ServiceAccountCredentials
-----------            ------------- -----------------------------------
 -------------------------
12/27/2019 11:00:30 AM contoso.com   {OU=AppStream,OU=Contoso,DC=Contoso,DC=com}
 Amazon.AppStream.Model.ServiceAccountCredentials
```

- Para obter detalhes sobre a API, consulte CreateDirectoryConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-APSFleet`

O código de exemplo a seguir mostra como usar `New-APSFleet`.

Ferramentas para PowerShell V5

Exemplo 1: Esta amostra cria uma nova AppStream frota

```
New-APSFleet -ComputeCapacity_DesiredInstance 1 -InstanceType stream.standard.medium
 -Name TestFleet -DisplayName TestFleet -FleetType ON_DEMAND -
EnableDefaultInternetAccess $True -VpcConfig_SubnetIds "subnet-123ce32","subnet-
a1234cfd" -VpcConfig_SecurityGroupIds sg-4d012a34 -ImageName SessionScriptTest -
Region us-west-2
```

Saída:

```
Arn                           : arn:aws:appstream:us-west-2:123456789012:fleet/
TestFleet
ComputeCapacityStatus         : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                   : 12/27/2019 11:24:42 AM
Description                   :
```

```
DisconnectTimeoutInSeconds     : 900
DisplayName                    : TestFleet
DomainJoinInfo                 :
EnableDefaultInternetAccess    : True
FleetErrors                    : {}
FleetType                      : ON_DEMAND
IamRoleArn                     :
IdleDisconnectTimeoutInSeconds : 0
ImageArn                       : arn:aws:appstream:us-west-2:123456789012:image/
SessionScriptTest
ImageName                      : SessionScriptTest
InstanceType                   : stream.standard.medium
MaxUserDurationInSeconds       : 57600
Name                           : TestFleet
State                          : STOPPED
VpcConfig                      : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes sobre a API, consulte CreateFleet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-APSImageBuilder

O código de exemplo a seguir mostra como usar `New-APSImageBuilder`.

Ferramentas para PowerShell V5

Exemplo 1: Esta amostra cria um construtor de imagens no AppStream

```
New-APSImageBuilder -InstanceType stream.standard.medium -Name TestIB -DisplayName
 TestIB -ImageName AppStream-WinServer2012R2-12-12-2019 -EnableDefaultInternetAccess
 $True -VpcConfig_SubnetId subnet-a1234cfd -VpcConfig_SecurityGroupIds sg-2d012a34 -
Region us-west-2
```

Saída:

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 12-16-2019
Arn                         : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime                 : 12/27/2019 11:39:24 AM
Description                 :
DisplayName                 : TestIB
```

```
DomainJoinInfo              :
EnableDefaultInternetAccess : True
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors          : {}
InstanceType                : stream.standard.medium
Name                        : TestIB
NetworkAccessConfiguration  :
Platform                    : WINDOWS
State                       : PENDING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes sobre a API, consulte CreateImageBuilder Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-APSImageBuilderStreamingURL

O código de exemplo a seguir mostra como usar `New-APSImageBuilderStreamingURL`.

Ferramentas para PowerShell V5

Exemplo 1: Essa amostra cria um URL ImageBuilder de streaming com validade de 2 horas

```
New-APSImageBuilderStreamingURL -Name TestIB -Validity 7200 -Region us-west-2
```

Saída:

```
Expires               StreamingURL
-------               ------------
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRJZCI6IjM5MzQwM
```

- Para obter detalhes sobre a API, consulte CreateImageBuilderStreamingURL Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-APSStack

O código de exemplo a seguir mostra como usar `New-APSStack`.

Ferramentas para PowerShell V5

Exemplo 1: Essa amostra cria uma nova AppStream pilha

```
New-APSStack -Name TestStack -DisplayName TestStack -ApplicationSettings_Enabled
 $True -ApplicationSettings_SettingsGroup TestStack -Region us-west-2
```

Saída:

```
AccessEndpoints     : {}
ApplicationSettings : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                 : arn:aws:appstream:us-west-2:123456789012:stack/TestStack
CreatedTime         : 12/27/2019 12:34:19 PM
Description         :
DisplayName         : TestStack
EmbedHostDomains    : {}
FeedbackURL         :
Name                : TestStack
RedirectURL         :
StackErrors         : {}
StorageConnectors   : {}
UserSettings        : {Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting}
```

- Para obter detalhes sobre a API, consulte CreateStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-APSStreamingURL`

O código de exemplo a seguir mostra como usar `New-APSStreamingURL`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra cria um URL de streaming da pilha

```
New-APSStreamingURL -StackName SessionScriptTest -FleetName SessionScriptNew -UserId
 TestUser
```

Saída:

```
Expires                 StreamingURL
-------                 ------------
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M
```

- Para obter detalhes da API, consulte CreateStreaming URL em Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-APSUsageReportSubscription`

O código de exemplo a seguir mostra como usar `New-APSUsageReportSubscription`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo ativa relatórios AppStream de uso

```
New-APSUsageReportSubscription
```

Saída:

```
S3BucketName                                    Schedule
------------                                    --------
appstream-logs-us-east-1-123456789012-sik2hnxe DAILY
```

- Para obter detalhes sobre a API, consulte CreateUsageReportSubscription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-APSUser`

O código de exemplo a seguir mostra como usar `New-APSUser`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra cria um usuário em USERPOOL

```
New-APSUser -UserName Test@lab.com -AuthenticationType USERPOOL -FirstName 'kt' -
LastName 'aws' -Select ^UserName
```

Saída:

```
Test@lab.com
```

- Para obter detalhes sobre a API, consulte CreateUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-APSFleet`

O código de exemplo a seguir mostra como usar `Register-APSFleet`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra registra a frota com uma pilha

```
Register-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- Para obter detalhes sobre a API, consulte AssociateFleet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-APSUserStackBatch`

O código de exemplo a seguir mostra como usar `Register-APSUserStackBatch`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra atribui a pilha a um usuário em USERPOOL

```
Register-APSUserStackBatch -UserStackAssociation
 @{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- Para obter detalhes sobre a API, consulte BatchAssociateUserStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSDirectoryConfig`

O código de exemplo a seguir mostra como usar `Remove-APSDirectoryConfig`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove a configuração AppStream do diretório

```
Remove-APSDirectoryConfig -DirectoryName contoso.com
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSDirectoryConfig (DeleteDirectoryConfig)" on
 target "contoso.com".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Para obter detalhes sobre a API, consulte DeleteDirectoryConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSFleet`

O código de exemplo a seguir mostra como usar `Remove-APSFleet`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove, exclui uma frota AppStream

```
Remove-APSFleet -Name TestFleet -Region us-west-2
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSFleet (DeleteFleet)" on target "TestFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Para obter detalhes sobre a API, consulte DeleteFleet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSImage`

O código de exemplo a seguir mostra como usar `Remove-APSImage`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exclui uma imagem

```
Remove-APSImage -Name TestImage -Region us-west-2
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImage (DeleteImage)" on target "TestImage".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A

Applications                : {}
AppstreamAgentVersion       : LATEST
Arn                         : arn:aws:appstream:us-west-2:123456789012:image/
TestImage
BaseImageArn                :
CreatedTime                 : 12/27/2019 1:34:10 PM
Description                 :
DisplayName                 : TestImage
ImageBuilderName            :
ImageBuilderSupported       : True
ImagePermissions            :
Name                        : TestImage
Platform                    : WINDOWS
PublicBaseImageReleasedDate : 6/12/2018 12:00:00 AM
State                       : AVAILABLE
StateChangeReason           :
Visibility                  : PRIVATE
```

- Para obter detalhes sobre a API, consulte DeleteImage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSImageBuilder`

O código de exemplo a seguir mostra como usar `Remove-APSImageBuilder`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo exclui um ImageBuilder

```
Remove-APSImageBuilder -Name TestIB -Region us-west-2
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImageBuilder (DeleteImageBuilder)" on target
 "TestIB".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A

AccessEndpoints             : {}
AppstreamAgentVersion       : 12-16-2019
Arn                         : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime                 : 12/27/2019 11:39:24 AM
Description                 :
DisplayName                 : TestIB
DomainJoinInfo              :
EnableDefaultInternetAccess : True
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors          : {}
InstanceType                : stream.standard.medium
Name                        : TestIB
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : DELETING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes sobre a API, consulte DeleteImageBuilder Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSImagePermission`

O código de exemplo a seguir mostra como usar `Remove-APSImagePermission`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra remove as permissões de uma imagem

```
Remove-APSImagePermission -Name Powershell -SharedAccountId 123456789012
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImagePermission (DeleteImagePermissions)" on
 target "Powershell".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Para obter detalhes sobre a API, consulte DeleteImagePermissions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSResourceTag`

O código de exemplo a seguir mostra como usar `Remove-APSResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove uma tag de recurso do AppStream recurso

```
Remove-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest -TagKey StackState
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSResourceTag (UntagResource)" on target
 "arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Para obter detalhes sobre a API, consulte UntagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSStack`

O código de exemplo a seguir mostra como usar `Remove-APSStack`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exclui uma pilha

```
Remove-APSStack -Name TestStack -Region us-west-2
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSStack (DeleteStack)" on target "TestStack".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Para obter detalhes sobre a API, consulte DeleteStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSUsageReportSubscription`

O código de exemplo a seguir mostra como usar `Remove-APSUsageReportSubscription`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo desativa a assinatura do Relatório AppStream de Uso

```
Remove-APSUsageReportSubscription
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSUsageReportSubscription
 (DeleteUsageReportSubscription)" on target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Para obter detalhes sobre a API, consulte DeleteUsageReportSubscription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-APSUser`

O código de exemplo a seguir mostra como usar `Remove-APSUser`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exclui um usuário de USERPOOL

```
Remove-APSUser -UserName TestUser@lab.com -AuthenticationType USERPOOL
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSUser (DeleteUser)" on target "TestUser@lab.com".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- Para obter detalhes sobre a API, consulte DeleteUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Revoke-APSSession`

O código de exemplo a seguir mostra como usar `Revoke-APSSession`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo revoga uma sessão para a frota AppStream

```
Revoke-APSSession -SessionId 6cd2f9a3-f948-4aa1-8014-8a7dcde14877
```

- Para obter detalhes sobre a API, consulte ExpireSession Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-APSFleet`

O código de exemplo a seguir mostra como usar `Start-APSFleet`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra inicia uma frota

```
Start-APSFleet -Name PowershellFleet
```

- Para obter detalhes sobre a API, consulte StartFleet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-APSImageBuilder`

O código de exemplo a seguir mostra como usar `Start-APSImageBuilder`.

Ferramentas para PowerShell V5

Exemplo 1: Essa amostra inicia um ImageBuilder

```
Start-APSImageBuilder -Name TestImage
```

Saída:

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
Arn                         : arn:aws:appstream:us-east-1:123456789012:image-
builder/TestImage
CreatedTime                 : 1/14/2019 4:33:05 AM
Description                 :
DisplayName                 : TestImage
DomainJoinInfo              :
EnableDefaultInternetAccess : False
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors          : {}
InstanceType                : stream.standard.large
Name                        : TestImage
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : PENDING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes sobre a API, consulte StartImageBuilder Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-APSFleet`

O código de exemplo a seguir mostra como usar `Stop-APSFleet`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra interrompe uma frota

```
Stop-APSFleet -Name PowershellFleet
```

- Para obter detalhes sobre a API, consulte StopFleet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-APSImageBuilder`

O código de exemplo a seguir mostra como usar `Stop-APSImageBuilder`.

Ferramentas para PowerShell V5

Exemplo 1: Essa amostra interrompe um ImageBuilder

```
Stop-APSImageBuilder -Name TestImage
```

Saída:

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
Arn                         : arn:aws:appstream:us-east-1:123456789012:image-
builder/TestImage
CreatedTime                 : 1/14/2019 4:33:05 AM
Description                 :
DisplayName                 : TestImage
DomainJoinInfo              :
EnableDefaultInternetAccess : False
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors          : {}
InstanceType                : stream.standard.large
Name                        : TestImage
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
```

```
Platform                    : WINDOWS
State                       : STOPPING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes sobre a API, consulte StopImageBuilder Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-APSFleet`

O código de exemplo a seguir mostra como usar `Unregister-APSFleet`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra cancela o registro de uma frota da pilha

```
Unregister-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- Para obter detalhes sobre a API, consulte DisassociateFleet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-APSUserStackBatch`

O código de exemplo a seguir mostra como usar `Unregister-APSUserStackBatch`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra remove um usuário de uma pilha atribuída

```
Unregister-APSUserStackBatch -UserStackAssociation
 @{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- Para obter detalhes sobre a API, consulte BatchDisassociateUserStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-APSDirectoryConfig`

O código de exemplo a seguir mostra como usar `Update-APSDirectoryConfig`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo atualiza a configuração do diretório criada no AppStream

```
Update-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso
\ServiceAccount -ServiceAccountCredentials_AccountPassword MyPass@1$@#
 -DirectoryName contoso.com -OrganizationalUnitDistinguishedName
 "OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com"
```

Saída:

```
CreatedTime           DirectoryName OrganizationalUnitDistinguishedNames
 ServiceAccountCredentials
-----------           ------------- ------------------------------------
 -------------------------
12/27/2019 3:50:02 PM contoso.com   {OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com}
 Amazon.AppStream.Model.ServiceAccountCredentials
```

- Para obter detalhes sobre a API, consulte UpdateDirectoryConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-APSFleet`

O código de exemplo a seguir mostra como usar `Update-APSFleet`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra atualiza as propriedades de uma frota

```
Update-APSFleet -Name PowershellFleet -EnableDefaultInternetAccess $True -
DisconnectTimeoutInSecond 950
```

Saída:

```
Arn                            : arn:aws:appstream:us-east-1:123456789012:fleet/
PowershellFleet
ComputeCapacityStatus          : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                    : 4/24/2019 8:39:41 AM
Description                    : PowershellFleet
DisconnectTimeoutInSeconds     : 950
```

```
DisplayName                    : PowershellFleet
DomainJoinInfo                 :
EnableDefaultInternetAccess    : True
FleetErrors                    : {}
FleetType                      : ON_DEMAND
IamRoleArn                     :
IdleDisconnectTimeoutInSeconds : 900
ImageArn                       : arn:aws:appstream:us-east-1:123456789012:image/
Powershell
ImageName                      : Powershell
InstanceType                   : stream.standard.medium
MaxUserDurationInSeconds       : 57600
Name                           : PowershellFleet
State                          : STOPPED
VpcConfig                      : Amazon.AppStream.Model.VpcConfig
```

- Para obter detalhes sobre a API, consulte UpdateFleet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-APSImagePermission`

O código de exemplo a seguir mostra como usar `Update-APSImagePermission`.

Ferramentas para PowerShell V5

Exemplo 1: Esta amostra compartilha uma AppStream imagem com outra conta

```
Update-APSImagePermission -Name Powershell -SharedAccountId 123456789012 -
ImagePermissions_AllowFleet $True -ImagePermissions_AllowImageBuilder $True
```

- Para obter detalhes sobre a API, consulte UpdateImagePermissions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-APSStack`

O código de exemplo a seguir mostra como usar `Update-APSStack`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra atualiza (ativa) a persistência das configurações do aplicativo e as pastas pessoais em uma pilha

```
Update-APSStack -Name PowershellStack -ApplicationSettings_Enabled $True
 -ApplicationSettings_SettingsGroup PowershellStack -StorageConnector
 @{ConnectorType="HOMEFOLDERS"}
```

Saída:

```
AccessEndpoints     : {}
ApplicationSettings : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                 : arn:aws:appstream:us-east-1:123456789012:stack/PowershellStack
CreatedTime         : 4/24/2019 8:49:29 AM
Description         : PowershellStack
DisplayName         : PowershellStack
EmbedHostDomains    : {}
FeedbackURL         :
Name                : PowershellStack
RedirectURL         :
StackErrors         : {}
StorageConnectors   : {Amazon.AppStream.Model.StorageConnector,
 Amazon.AppStream.Model.StorageConnector}
UserSettings        : {Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting}
```

- Para obter detalhes sobre a API, consulte UpdateStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Aurora usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Aurora.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-RDSOrderableDBInstanceOption`

O código de exemplo a seguir mostra como usar `Get-RDSOrderableDBInstanceOption`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo lista as versões do mecanismo de banco de dados compatíveis com uma classe de instância de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Exemplo 2: este exemplo lista as classes de instância de bancos de dados compatíveis com uma versão específica do mecanismo de banco de dados em uma Região da AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Para obter detalhes sobre a API, consulte DescribeOrderableDBInstanceOptions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de Auto Scaling usando Tools for PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-ASLoadBalancer`

O código de exemplo a seguir mostra como usar `Add-ASLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo anexa o Balanceador de Carga especificado ao grupo do Auto Scaling especificado.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Para obter detalhes sobre a API, consulte AttachLoadBalancers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Complete-ASLifecycleAction`

O código de exemplo a seguir mostra como usar `Complete-ASLifecycleAction`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo conclui a ação de ciclo de vida especificada.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Para obter detalhes sobre a API, consulte CompleteLifecycleAction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-ASMetricsCollection`

O código de exemplo a seguir mostra como usar `Disable-ASMetricsCollection`.

Ferramentas para PowerShell V5

Example 1: Desativa o monitoramento de métricas especificadas para o grupo de Auto Scaling.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric @("GroupMinSize",
 "GroupMaxSize")
```

Exemplo 2: Esse exemplo desativa o monitoramento de todas as métricas do grupo do Auto Scaling especificado.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Para obter detalhes sobre a API, consulte DisableMetricsCollection Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Dismount-ASInstance`

O código de exemplo a seguir mostra como usar `Dismount-ASInstance`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo desvincula a instância especificada do grupo do Auto Scaling especificado e diminui a capacidade desejada para que o Auto Scaling não inicie uma instância substituta.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached in
 response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
```

```
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo desvincula a instância especificada do grupo de Auto Scaling especificado sem diminuir a capacidade desejada. O Auto Scaling inicia uma nova instância de substituição.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached in
 response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes sobre a API, consulte DetachInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Dismount-ASLoadBalancer`

O código de exemplo a seguir mostra como usar `Dismount-ASLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo desvincula o balanceador de carga especificado do grupo de Auto Scaling especificado.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Para obter detalhes sobre a API, consulte DetachLoadBalancers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-ASMetricsCollection`

O código de exemplo a seguir mostra como usar `Enable-ASMetricsCollection`.

Ferramentas para PowerShell V5

Exemplo 1: Ativa o monitoramento de métricas especificadas para o grupo de Auto Scaling.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Exemplo 2: Este exemplo permite o monitoramento de todas as métricas do grupo do Auto Scaling especificado.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Para obter detalhes sobre a API, consulte EnableMetricsCollection Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enter-ASStandby`

O código de exemplo a seguir mostra como usar `Enter-ASStandby`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo coloca a instância especificada em modo de espera e diminui a capacidade desejada para que o Auto Scaling não inicie uma instância substituta.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
```

```
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo coloca a instância especificada em modo de espera sem diminuir a capacidade desejada. O Auto Scaling inicia uma nova instância de substituição.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes sobre a API, consulte EnterStandby Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Exit-ASStandby`

O código de exemplo a seguir mostra como usar `Exit-ASStandby`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo tira as instâncias especificadas do modo de espera.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Saída:

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out of
 standby in response to a user
                       request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- Para obter detalhes sobre a API, consulte ExitStandby Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASAccountLimit`

O código de exemplo a seguir mostra como usar `Get-ASAccountLimit`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo descreve os limites de recursos do Auto Scaling para sua AWS conta.

```
Get-ASAccountLimit
```

Saída:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- Para obter detalhes sobre a API, consulte DescribeAccountLimits Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASAdjustmentType`

O código de exemplo a seguir mostra como usar `Get-ASAdjustmentType`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve os tipos de ajuste que são compatíveis com o Auto Scaling.

```
Get-ASAdjustmentType
```

Saída:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Para obter detalhes sobre a API, consulte DescribeAdjustmentTypes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASAutoScalingGroup`

O código de exemplo a seguir mostra como usar `Get-ASAutoScalingGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os nomes de seus grupos do Auto Scaling.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Saída:

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Exemplo 2: Este exemplo descreve o grupo do Auto Scaling especificado.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Saída:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

Exemplo 3: Este exemplo descreve os grupos do Auto Scaling especificados.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

Exemplo 4: Este exemplo descreve as instâncias do Auto Scaling do grupo do Auto Scaling especificado.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Exemplo 5: Este exemplo descreve todos os seus grupos do Auto Scaling.

```
Get-ASAutoScalingGroup
```

Exemplo 6: Este LaunchTemplate exemplo descreve o grupo de Auto Scaling especificado. Este exemplo pressupõe que as “Opções de compra de instância” estejam definidas como “Aderir ao modelo de lançamento”. Caso essa opção esteja definida como “Combinar opções

de compra e tipos de instância", LaunchTemplate poderá ser acessada usando a propriedade MixedInstancesPolicy.LaunchTemplate "".

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Saída:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- Para obter detalhes sobre a API, consulte DescribeAutoScalingGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASAutoScalingInstance`

O código de exemplo a seguir mostra como usar `Get-ASAutoScalingInstance`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os IDs de suas instâncias do Auto Scaling.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Saída:

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

Exemplo 2: Este exemplo descreve a instância do Auto Scaling especificada.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Saída:

```
AutoScalingGroupName    : my-asg
```

```
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Exemplo 3: Este exemplo descreve duas instâncias do Auto Scaling especificadas.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

Exemplo 4: Este exemplo descreve as instâncias do Auto Scaling do grupo do Auto Scaling especificado.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

Exemplo 5: Este exemplo descreve todas as instâncias do Auto Scaling.

```
Get-ASAutoScalingInstance
```

- Para obter detalhes sobre a API, consulte DescribeAutoScalingInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASAutoScalingNotificationType`

O código de exemplo a seguir mostra como usar `Get-ASAutoScalingNotificationType`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os tipos de notificação compatíveis com o Auto Scaling.

```
Get-ASAutoScalingNotificationType
```

Saída:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
```

```
autoscaling:TEST_NOTIFICATION
```

- Para obter detalhes sobre a API, consulte DescribeAutoScalingNotificationTypes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASLaunchConfiguration`

O código de exemplo a seguir mostra como usar `Get-ASLaunchConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os nomes de suas configurações de execução.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Saída:

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Exemplo 2: Este exemplo descreve a configuração de execução especificada.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Saída:

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
```

```
InstanceType                 : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN       : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                               e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName      : my-lc-1
PlacementTenancy             :
RamdiskId                    :
SecurityGroups               : {sg-67ef0308}
SpotPrice                    :
UserData                     :
```

Exemplo 3: Este exemplo descreve as duas configurações de execução especificadas.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Exemplo 4: Este exemplo descreve todas as suas configurações de execução.

```
Get-ASLaunchConfiguration
```

- Para obter detalhes sobre a API, consulte DescribeLaunchConfigurations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASLifecycleHook`

O código de exemplo a seguir mostra como usar `Get-ASLifecycleHook`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o gancho do ciclo de vida especificado.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook
```

Saída:

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
```

```
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

Exemplo 2: Descreve os ganchos do ciclo de vida do grupo de Auto Scaling especificado.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Exemplo 3: Este exemplo descreve todos ganchos do ciclo de vida os todos seus grupos do Auto Scaling.

```
Get-ASLifecycleHook
```

- Para obter detalhes sobre a API, consulte DescribeLifecycleHooks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASLifecycleHookType`

O código de exemplo a seguir mostra como usar `Get-ASLifecycleHookType`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os tipos de gancho do ciclo de vida compatíveis com o Auto Scaling.

```
Get-ASLifecycleHookType
```

Saída:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Para obter detalhes sobre a API, consulte DescribeLifecycleHookTypes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASLoadBalancer`

O código de exemplo a seguir mostra como usar `Get-ASLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: Descreve os load balancers do grupo de Auto Scaling especificado.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Saída:

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- Para obter detalhes sobre a API, consulte DescribeLoadBalancers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASMetricCollectionType`

O código de exemplo a seguir mostra como usar `Get-ASMetricCollectionType`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os tipos de coleta de métricas compatíveis com o Auto Scaling.

```
(Get-ASMetricCollectionType).Metrics
```

Saída:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Exemplo 2: Este exemplo lista as granularidades correspondentes.

```
(Get-ASMetricCollectionType).Granularities
```

Saída:

```
Granularity
-----------
1Minute
```

- Para obter detalhes sobre a API, consulte DescribeMetricCollectionTypes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASNotificationConfiguration`

O código de exemplo a seguir mostra como usar `Get-ASNotificationConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo descreve as ações de notificação associadas ao grupo de Auto Scaling especificado.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Saída:

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

Exemplo 2: Este exemplo descreve as ações de notificação associadas a todos os seus grupos do Auto Scaling.

```
Get-ASNotificationConfiguration
```

- Para obter detalhes sobre a API, consulte DescribeNotificationConfigurations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASPolicy`

O código de exemplo a seguir mostra como usar `Get-ASPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo descreve o grupo do Auto Scaling especificado.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Saída:

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

Exemplo 2: Este exemplo descreve duas políticas especificadas para o grupo do Auto Scaling especificado.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

Exemplo 3: Este exemplo descreve todas as políticas para todos os seus grupos do Auto Scaling.

```
Get-ASPolicy
```

- Para obter detalhes sobre a API, consulte DescribePolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASScalingActivity`

O código de exemplo a seguir mostra como usar `Get-ASScalingActivity`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve as ações de escalabilidade das últimas seis semanas do grupo do Auto Scaling especificado.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Saída:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set group
 desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the capacity
 from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity from
 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
```

```
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

Exemplo 2: Este exemplo descreve a atividade de escala especificada.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Exemplo 3: Este exemplo descreve as atividades de escala das últimas seis semanas para todos os seus grupos do Auto Scaling.

```
Get-ASScalingActivity
```

- Para obter detalhes sobre a API, consulte DescribeScalingActivities Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASScalingProcessType`

O código de exemplo a seguir mostra como usar `Get-ASScalingProcessType`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista os tipos de processo compatíveis com o Auto Scaling.

```
Get-ASScalingProcessType
```

Saída:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Para obter detalhes sobre a API, consulte DescribeScalingProcessTypes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASScheduledAction`

O código de exemplo a seguir mostra como usar `Get-ASScheduledAction`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve as ações de escalabilidade agendadas do grupo do Auto Scaling especificado.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Saída:

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-asg:scheduledActionName/
myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

Exemplo 2: Este exemplo descreve as ações de escala programada especificadas.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

Exemplo 3: Este exemplo descreve as ações de escala programadas que começam no horário especificado.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Exemplo 4: Este exemplo descreve as ações de escala programadas que terminam no horário especificado.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Exemplo 5: Este exemplo descreve as ações de escala programadas para todos os seus grupos do Auto Scaling.

```
Get-ASScheduledAction
```

- Para obter detalhes sobre a API, consulte DescribeScheduledActions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASTag`

O código de exemplo a seguir mostra como usar `Get-ASTag`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve as tags com um valor-chave de “myTag” ou “myTag2”. Os valores possíveis para o nome do filtro são “grupo do auto scaling”, “chave”, “valor” e “propagar na execução”. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Saída:

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro para o parâmetro Filter.

```
$keys = New-Object string[] 2
```

```
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Exemplo 3: Este exemplo descreve todas as tags para tudos seus grupos do Auto Scaling.

```
Get-ASTag
```

- Para obter detalhes sobre a API, consulte DescribeTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASTerminationPolicyType`

O código de exemplo a seguir mostra como usar `Get-ASTerminationPolicyType`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as políticas de terminação que são compatíveis com o Auto Scaling.

```
Get-ASTerminationPolicyType
```

Saída:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Para obter detalhes sobre a API, consulte DescribeTerminationPolicyTypes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Mount-ASInstance`

O código de exemplo a seguir mostra como usar `Mount-ASInstance`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo anexa o grupo de destino especificado ao grupo do Auto Scaling especificado. O Auto Scaling aumenta automaticamente a capacidade desejada do grupo do Auto Scaling.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Para obter detalhes sobre a API, consulte AttachInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ASAutoScalingGroup`

O código de exemplo a seguir mostra como usar `New-ASAutoScalingGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um grupo do Auto Scaling com o nome e os atributos especificados. A capacidade desejada padrão é o tamanho mínimo. Portanto, esse grupo do Auto Scaling inicia duas instâncias, uma em cada uma das duas zonas de disponibilidade especificadas.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -
MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Para obter detalhes sobre a API, consulte CreateAutoScalingGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ASLaunchConfiguration`

O código de exemplo a seguir mostra como usar `New-ASLaunchConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma configuração de inicialização chamada “my-lc”. As instâncias do EC2 iniciadas por grupos do Auto Scaling que usam essa configuração de inicialização usam o tipo de instância, a AMI, o grupo de segurança e o perfil do IAM especificados.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType "m3.medium" -
ImageId "ami-12345678" -SecurityGroup "sg-12345678" -IamInstanceProfile "myIamRole"
```

- Para obter detalhes sobre a API, consulte CreateLaunchConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ASAutoScalingGroup`

O código de exemplo a seguir mostra como usar `Remove-ASAutoScalingGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui o grupo do Auto Scaling especificado se ele não tiver instâncias em execução. A confirmação será solicitada antes que a operação continue.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on Target
 "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Exemplo 3: Este exemplo exclui o grupo do Auto Scaling especificado e encerra todas as instâncias em execução que ele contém.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Para obter detalhes sobre a API, consulte DeleteAutoScalingGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ASLaunchConfiguration`

O código de exemplo a seguir mostra como usar `Remove-ASLaunchConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a configuração de inicialização especificada se ela não estiver anexada a um grupo do Auto Scaling. A confirmação será solicitada antes que a operação continue.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)" on
 Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Para obter detalhes sobre a API, consulte DeleteLaunchConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ASLifecycleHook`

O código de exemplo a seguir mostra como usar `Remove-ASLifecycleHook`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o gancho do ciclo de vida especificado para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Saída:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- Para obter detalhes sobre a API, consulte DeleteLifecycleHook Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ASNotificationConfiguration`

O código de exemplo a seguir mostra como usar `Remove-ASNotificationConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a ação de notificação especificada. A confirmação será solicitada antes que a operação continue.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Para obter detalhes sobre a API, consulte DeleteNotificationConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ASPolicy`

O código de exemplo a seguir mostra como usar `Remove-ASPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a política especificada para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Para obter detalhes sobre a API, consulte DeletePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ASScheduledAction`

O código de exemplo a seguir mostra como usar `Remove-ASScheduledAction`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a política especificada para o grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- Para obter detalhes sobre a API, consulte DeleteScheduledAction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ASTag`

O código de exemplo a seguir mostra como usar `Remove-ASTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag especificada do grupo do Auto Scaling especificado. A confirmação será solicitada antes que a operação continue. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Se você especificar o parâmetro Force, não será solicitada a confirmação antes de prosseguir com a operação.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Exemplo 3: Com o Powershell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Para obter detalhes sobre a API, consulte DeleteTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Resume-ASProcess`

O código de exemplo a seguir mostra como usar `Resume-ASProcess`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo reinicia o processo do Auto Scaling especificado para o grupo de Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Exemplo 2: Este exemplo retoma todos os processos suspensos do Auto Scaling para o grupo do Auto Scaling especificado.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Para obter detalhes sobre a API, consulte ResumeProcesses Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ASDesiredCapacity`

O código de exemplo a seguir mostra como usar `Set-ASDesiredCapacity`.

Ferramentas para PowerShell V5

Exemplo 1: Define o tamanho do grupo de Auto Scaling especificado.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Exemplo 2: Este exemplo define o tamanho do grupo do Auto Scaling especificado e aguarda a conclusão do período de espera antes de fazer o escalonamento para o novo tamanho.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown
 $true
```

- Para obter detalhes sobre a API, consulte SetDesiredCapacity Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ASInstanceHealth`

O código de exemplo a seguir mostra como usar `Set-ASInstanceHealth`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo define o status da instância especificada como “Não saudável”, tirando-a de serviço. O Auto Scaling termina e substitui a instância.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Exemplo 2: Este exemplo define o status da instância especificada como “Saudável”, mantendo-a em serviço. Definir um período de carência da verificação de integridade para um grupo do Auto Scaling

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- Para obter detalhes sobre a API, consulte SetInstanceHealth Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ASInstanceProtection`

O código de exemplo a seguir mostra como usar `Set-ASInstanceProtection`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo habilita a proteção de instância na a instância especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Exemplo 2: Este exemplo desabilita a proteção de instância na instância especificada.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- Para obter detalhes sobre a API, consulte SetInstanceProtection Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ASTag`

O código de exemplo a seguir mostra como usar `Set-ASTag`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo adiciona uma única tag ao grupo do Auto Scaling especificado. A chave da tag é 'myTag' e o valor da tag é 'my'TagValue. O Auto Scaling propaga essa tag para as instâncias do EC2 subsequentes lançadas pelo grupo do Auto Scaling. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- Para obter detalhes sobre a API, consulte CreateOrUpdateTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-ASPolicy`

O código de exemplo a seguir mostra como usar `Start-ASPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo executa a política especificada para o grupo do Auto Scaling especificado.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Exemplo 2: Este exemplo executa a política especificada para o grupo do Auto Scaling especificado, depois de aguardar a conclusão do período de espera.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Para obter detalhes sobre a API, consulte ExecutePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-ASInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `Stop-ASInstanceInAutoScalingGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo encerra a instância especificada e diminui a capacidade desejada de seu grupo do Auto Scaling para que o Auto Scaling não inicie uma instância substituta.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Saída:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of
 service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

Exemplo 2: Este exemplo encerra a instância especificada sem diminuir a capacidade desejada de seu grupo do Auto Scaling. Auto Scaling inicia uma nova instância de substituição.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Saída:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of
 service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
```

```
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- Para obter detalhes sobre a API, consulte TerminateInstanceInAutoScalingGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Suspend-ASProcess`

O código de exemplo a seguir mostra como usar `Suspend-ASProcess`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo suspende o processo de escalabilidade especificado para o grupo do Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Exemplo 2: Este exemplo suspende todos os processos do Auto Scaling para o grupo do Auto Scaling especificado.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Para obter detalhes sobre a API, consulte SuspendProcesses Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-ASAutoScalingGroup`

O código de exemplo a seguir mostra como usar `Update-ASAutoScalingGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo atualiza o tamanho mínimo e máximo do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Exemplo 2: Este exemplo atualiza o período de espera padrão do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Exemplo 3: Este exemplo atualiza as zonas de disponibilidade do grupo do Auto Scaling especificado.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Exemplo 4: Este exemplo atualiza o grupo do Auto Scaling especificado para usar verificações de integridade do Elastic Load Balancing.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Para obter detalhes sobre a API, consulte UpdateAutoScalingGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-ASLifecycleActionHeartbeat`

O código de exemplo a seguir mostra como usar `Write-ASLifecycleActionHeartbeat`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo registra uma pulsação para a ação de ciclo de vida especificada. Isso mantém a instância em um estado pendente até que você conclua a ação personalizada.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Para obter detalhes sobre a API, consulte RecordLifecycleActionHeartbeat Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-ASLifecycleHook`

O código de exemplo a seguir mostra como usar `Write-ASLifecycleHook`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo adiciona o gancho do ciclo de vida especificado ao grupo do Auto Scaling especificado.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- Para obter detalhes sobre a API, consulte PutLifecycleHook Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-ASNotificationConfiguration`

O código de exemplo a seguir mostra como usar `Write-ASNotificationConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo configura o grupo do Auto Scaling especificado para enviar uma notificação ao tópico SNS especificado quando ele iniciar instâncias do EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-
topic"
```

Exemplo 2: Este exemplo configura o grupo do Auto Scaling especificado para enviar uma notificação ao tópico SNS especificado quando ele iniciar ou encerrar instâncias do EC2.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Para obter detalhes sobre a API, consulte PutNotificationConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-ASScalingPolicy`

O código de exemplo a seguir mostra como usar `Write-ASScalingPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo adiciona a política especificada ao grupo do Auto Scaling especificado. O tipo de ajuste especificado determina como interpretar o ScalingAdjustment parâmetro. Com

'ChangeInCapacity', um valor positivo aumenta a capacidade pelo número especificado de instâncias e um valor negativo diminui a capacidade pelo número especificado de instâncias.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Saída:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Para obter detalhes sobre a API, consulte PutScalingPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-ASScheduledUpdateGroupAction`

O código de exemplo a seguir mostra como usar `Write-ASScheduledUpdateGroupAction`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria ou atualiza uma ação programada única para alterar a capacidade desejada na hora de início especificada.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName
 "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- Para obter detalhes sobre a API, consulte PutScheduledUpdateGroupAction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS Budgets exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS Budgets.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`New-BGTBudget`

O código de exemplo a seguir mostra como usar `New-BGTBudget`.

Ferramentas para PowerShell V5

Exemplo 1: cria um orçamento com as restrições orçamentárias e de tempo especificadas com notificações por e-mail.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @("user@domain.com")
    SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- Para obter detalhes sobre a API, consulte CreateBudget Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS Cloud9 exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS Cloud9.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-C9EnvironmentData`

O código de exemplo a seguir mostra como usar `Get-C9EnvironmentData`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém informações sobre os ambientes de desenvolvimento do AWS Cloud9 especificados.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Saída:

```
Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
Description : Created from CodeStar.
Id          : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ec2-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ec2
```

```
Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description :
Id          : 1980b80e5f584920801c09086667f0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ssh-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ssh
```

Exemplo 2: Este exemplo obtém informações sobre o status do ciclo de vida do ambiente de desenvolvimento AWS Cloud9 especificado.

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Saída:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- Para obter detalhes sobre a API, consulte DescribeEnvironments Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-C9EnvironmentList`

O código de exemplo a seguir mostra como usar `Get-C9EnvironmentList`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém uma lista dos identificadores do ambiente de desenvolvimento do AWS Cloud9 disponíveis.

```
Get-C9EnvironmentList
```

Saída:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- Para obter detalhes sobre a API, consulte ListEnvironments Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-C9EnvironmentMembershipList`

O código de exemplo a seguir mostra como usar `Get-C9EnvironmentMembershipList`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém informações sobre os membros do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Exemplo 2: Este exemplo obtém informações sobre o proprietário do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
```

```
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Exemplo 3: Este exemplo obtém informações sobre o membro do ambiente especificado para vários ambientes de desenvolvimento do AWS Cloud9.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

- Para obter detalhes sobre a API, consulte DescribeEnvironmentMemberships Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-C9EnvironmentStatus`

O código de exemplo a seguir mostra como usar `Get-C9EnvironmentStatus`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém informações de status para o ambiente de desenvolvimento AWS Cloud9 especificado.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Saída:

```
Message                     Status
-------                     ------
Environment is ready to use ready
```

- Para obter detalhes sobre a API, consulte DescribeEnvironmentStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-C9EnvironmentEC2

O código de exemplo a seguir mostra como usar `New-C9EnvironmentEC2`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um ambiente de desenvolvimento do AWS Cloud9 com as configurações especificadas, inicia uma instância do Amazon Elastic Compute Cloud (Amazon EC2) e, em seguida, se conecta da instância ao ambiente.

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
 "My demonstration development environment." -InstanceType t2.micro -OwnerArn
 arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

Saída:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes sobre a API, consulte CreateEnvironmentEc2 Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-C9EnvironmentMembership

O código de exemplo a seguir mostra como usar `New-C9EnvironmentMembership`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo adiciona o membro do ambiente especificado ao ambiente de desenvolvimento do AWS Cloud9 especificado.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
 -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-write
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
```

```
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Para obter detalhes sobre a API, consulte CreateEnvironmentMembership Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-C9Environment`

O código de exemplo a seguir mostra como usar `Remove-C9Environment`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo exclui o ambiente de desenvolvimento AWS Cloud9 especificado. Se uma instância do Amazon EC2 estiver conectada ao ambiente, também encerrará a instância.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes sobre a API, consulte DeleteEnvironment Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-C9EnvironmentMembership`

O código de exemplo a seguir mostra como usar `Remove-C9EnvironmentMembership`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo exclui o membro do ambiente especificado do ambiente de desenvolvimento AWS Cloud9 especificado.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Para obter detalhes sobre a API, consulte DeleteEnvironmentMembership Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-C9Environment`

O código de exemplo a seguir mostra como usar `Update-C9Environment`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo altera as configurações especificadas do ambiente de desenvolvimento existente do AWS Cloud9.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- Para obter detalhes sobre a API, consulte UpdateEnvironment Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-C9EnvironmentMembership`

O código de exemplo a seguir mostra como usar `Update-C9EnvironmentMembership`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo altera as configurações do membro do ambiente existente especificado para o ambiente de desenvolvimento AWS Cloud9 especificado.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Saída:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Para obter detalhes sobre a API, consulte UpdateEnvironmentMembership Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

CloudFormation exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com CloudFormation.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-CFNStack`

O código de exemplo a seguir mostra como usar `Get-CFNStack`.

Ferramentas para PowerShell V5

Exemplo 1: retorna um conjunto de instâncias de pilha descrevendo todas as pilhas do usuário.

```
Get-CFNStack
```

Exemplo 2: retorna uma instância de pilha descrevendo a pilha especificada

```
Get-CFNStack -StackName "myStack"
```

- Para obter detalhes sobre a API, consulte DescribeStacks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFNStackEvent`

O código de exemplo a seguir mostra como usar `Get-CFNStackEvent`.

Ferramentas para PowerShell V5

Exemplo 1: retorna todos os eventos relacionados à pilha especificada.

```
Get-CFNStackEvent -StackName "myStack"
```

- Para obter detalhes sobre a API, consulte DescribeStackEvents Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFNStackResource`

O código de exemplo a seguir mostra como usar `Get-CFNStackResource`.

Ferramentas para PowerShell V5

Exemplo 1: retorna a descrição de um recurso identificado no modelo associado à pilha especificada pelo ID lógico "MyDBInstance".

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Para obter detalhes sobre a API, consulte DescribeStackResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFNStackResourceList`

O código de exemplo a seguir mostra como usar `Get-CFNStackResourceList`.

Ferramentas para PowerShell V5

Exemplo 1: retorna as descrições dos AWS recursos de até 100 recursos associados à pilha especificada. Para obter detalhes de todos os recursos associados a uma pilha, use o Get-CFNStackResourceSummary, que também oferece suporte à paginação manual dos resultados.

```
Get-CFNStackResourceList -StackName "myStack"
```

Exemplo 2: retorna a descrição da instância do Amazon EC2 identificada no modelo associado à pilha especificada pelo ID lógico "Ec2Instance".

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Exemplo 3: retorna a descrição de até 100 recursos associados à pilha contendo uma instância do Amazon EC2 identificada pelo ID de instância "i-123456". Para obter detalhes de todos os recursos associados a uma pilha, use o Get-CFNStackResourceSummary, que também oferece suporte à paginação manual dos resultados.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Exemplo 4: retorna a descrição da instância do Amazon EC2 identificada pelo ID lógico "Ec2Instance" no modelo de uma pilha. A pilha é identificada usando o ID de recurso físico de um recurso que ela contém; nesse caso, também uma instância do Amazon EC2 com o ID de instância "i-123456". Um outro recurso físico também poderia ser usado para identificar a pilha, dependendo do conteúdo do modelo, por exemplo, um bucket do Amazon S3.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- Para obter detalhes sobre a API, consulte DescribeStackResources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFNStackResourceSummary`

O código de exemplo a seguir mostra como usar `Get-CFNStackResourceSummary`.

Ferramentas para PowerShell V5

Exemplo 1: retorna as descrições de todos os recursos associados à pilha especificada.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- Para obter detalhes sobre a API, consulte ListStackResources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFNStackSummary`

O código de exemplo a seguir mostra como usar `Get-CFNStackSummary`.

Ferramentas para PowerShell V5

Exemplo 1: retorna informações resumidas de todas as pilhas.

```
Get-CFNStackSummary
```

Exemplo 2: retorna informações resumidas de todas as pilhas que estão sendo criadas no momento.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Exemplo 3: retorna informações resumidas de todas as pilhas que estão sendo criadas ou atualizadas no momento.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

- Para obter detalhes sobre a API, consulte ListStacks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFNTemplate`

O código de exemplo a seguir mostra como usar `Get-CFNTemplate`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o modelo associado à pilha especificada.

```
Get-CFNTemplate -StackName "myStack"
```

- Para obter detalhes sobre a API, consulte GetTemplate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Measure-CFNTemplateCost`

O código de exemplo a seguir mostra como usar `Measure-CFNTemplateCost`.

Ferramentas para PowerShell V5

Exemplo 1: retorna uma URL de calculadora mensal AWS simples com uma string de consulta que descreve os recursos necessários para executar o modelo. O modelo é obtido da URL do Amazon S3 especificada e do único parâmetro de personalização aplicado. O parâmetro também pode ser especificado usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Exemplo 2: retorna uma URL de calculadora mensal AWS simples com uma string de consulta que descreve os recursos necessários para executar o modelo. O modelo é analisado a partir do conteúdo fornecido e dos parâmetros de personalização aplicados (este exemplo pressupõe que o conteúdo do modelo teria declarado dois parâmetros, '' e 'KeyName')InstanceType. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                      @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Exemplo 3: Usa New-Object para criar o conjunto de parâmetros do modelo e retorna uma URL da Calculadora Mensal AWS Simples com uma string de consulta que descreve os recursos necessários para executar o modelo. O modelo é analisado a partir do conteúdo fornecido, com parâmetros de personalização (este exemplo pressupõe que o conteúdo do modelo teria declarado dois parâmetros, '' e KeyName ')InstanceType.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter @( $p1,
 $p2 )
```

- Para obter detalhes sobre a API, consulte EstimateTemplateCost Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-CFNStack`

O código de exemplo a seguir mostra como usar `New-CFNStack`.

Ferramentas para PowerShell V5

Exemplo 1: cria uma nova pilha com o nome especificado. O modelo é analisado a partir do conteúdo fornecido com parâmetros de personalização ("PK1" e "PK2" representam os nomes dos parâmetros declarados no conteúdo do modelo, "PV1" e "PV2" representam os valores desses parâmetros. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se houver falha na criação da pilha, ela não será revertida.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Exemplo 2: cria uma nova pilha com o nome especificado. O modelo é analisado a partir do conteúdo fornecido com parâmetros de personalização ("PK1" e "PK2" representam os nomes dos parâmetros declarados no conteúdo do modelo, "PV1" e "PV2" representam os valores desses parâmetros. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se houver falha na criação da pilha, ela será revertida.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

Exemplo 3: cria uma nova pilha com o nome especificado. O modelo é obtido da URL do Amazon S3 com parâmetros de personalização ('PK1' representa o nome de um parâmetro declarado no conteúdo do modelo, 'PV1' representa o valor do parâmetro. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e

'ParameterValue'. Se a criação da pilha falhar, ela será revertida (o mesmo que especificar -DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 4: cria uma nova pilha com o nome especificado. O modelo é obtido da URL do Amazon S3 com parâmetros de personalização ('PK1' representa o nome de um parâmetro declarado no conteúdo do modelo, 'PV1' representa o valor do parâmetro. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. Se a criação da pilha falhar, ela será revertida (o mesmo que especificar - DisableRollback $false). Os AENs de notificação especificados receberão os eventos relacionados à pilha publicados.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- Para obter detalhes sobre a API, consulte CreateStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CFNStack`

O código de exemplo a seguir mostra como usar `Remove-CFNStack`.

Ferramentas para PowerShell V5

Exemplo 1: exclui a pilha especificada.

```
Remove-CFNStack -StackName "myStack"
```

- Para obter detalhes sobre a API, consulte DeleteStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Resume-CFNUpdateRollback`

O código de exemplo a seguir mostra como usar `Resume-CFNUpdateRollback`.

Ferramentas para PowerShell V5

Exemplo 1: continua a reversão da pilha nomeada, que deve estar no estado "UPDATE_ROLLBACK_FAILED". Se a reversão contínua for bem-sucedida, a pilha entrará no estado "UPDATE_ROLLBACK_COMPLETE".

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Para obter detalhes sobre a API, consulte ContinueUpdateRollback Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-CFNUpdateStack`

O código de exemplo a seguir mostra como usar `Stop-CFNUpdateStack`.

Ferramentas para PowerShell V5

Exemplo 1: cancela uma atualização na pilha especificada.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Para obter detalhes sobre a API, consulte CancelUpdateStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Test-CFNStack`

O código de exemplo a seguir mostra como usar `Test-CFNStack`.

Ferramentas para PowerShell V5

Exemplo 1: testa se a pilha atingiu um dos estados UPDATE_ROLLBACK_COMPLETE, CREATE_COMPLETE, ROLLBACK_COMPLETE ou UPDATE_COMPLETE.

```
Test-CFNStack -StackName MyStack
```

Saída:

```
False
```

Exemplo 2: testa se a pilha atingiu o status UPDATE_COMPLETE ou UPDATE_ROLLBACK_COMPLETE.

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

Saída:

```
True
```

- Para obter detalhes sobre a API, consulte Test-CFNStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Test-CFNTemplate`

O código de exemplo a seguir mostra como usar `Test-CFNTemplate`.

Ferramentas para PowerShell V5

Exemplo 1: valida o conteúdo do modelo especificado. A saída detalha os recursos, a descrição e os parâmetros do modelo.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Exemplo 2: valida o modelo especificado acessado por meio de uma URL do Amazon S3. A saída detalha os recursos, a descrição e os parâmetros do modelo.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- Para obter detalhes sobre a API, consulte ValidateTemplate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CFNStack`

O código de exemplo a seguir mostra como usar `Update-CFNStack`.

Ferramentas para PowerShell V5

Exemplo 1: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "PK1" representa o nome de um parâmetro declarado no modelo e "PV1" representa seu valor. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 2: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemplo 3: atualiza a pilha "myStack" com o modelo e os parâmetros de personalização especificados. "PK1" representa o nome de um parâmetro declarado no modelo e "PV2" representa seu valor. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Exemplo 4: atualiza a pilha "myStack" com o modelo obtido no Amazon S3 e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
```

```
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Exemplo 5: atualiza a pilha "myStack" que, este exemplo pressupõe conter recursos do IAM, com o modelo obtido do Amazon S3 e os parâmetros de personalização especificados. "K1" e "PK2" representam os nomes dos parâmetros declarados no modelo, "PV1" e "PV2" representam os valores solicitados. Os parâmetros de personalização também podem ser especificados usando 'Chave' e 'Valor' em vez de 'ParameterKey' e 'ParameterValue'. As pilhas contendo recursos do IAM exigem que você especifique o parâmetro -Capabilities “CAPABILITY_IAM”, caso contrário, a atualização falhará com um erro ". InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- Para obter detalhes sobre a API, consulte UpdateStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Wait-CFNStack`

O código de exemplo a seguir mostra como usar `Wait-CFNStack`.

Ferramentas para PowerShell V5

Exemplo 1: testa se a pilha atingiu um dos estados UPDATE_ROLLBACK_COMPLETE, CREATE_COMPLETE, ROLLBACK_COMPLETE ou UPDATE_COMPLETE. Se a pilha não estiver em um dos estados, o comando aguarda dois segundos antes de testar o status novamente. Isso é repetido até que a pilha alcance um dos estados solicitados ou até que o tempo limite padrão de 60 segundos termine. Se o tempo limite for excedido, será lançada uma exceção. Se a pilha atingir um dos estados solicitados dentro do tempo limite, ela será retornada ao pipeline.

```
$stack = Wait-CFNStack -StackName MyStack
```

Exemplo 2: esse exemplo aguarda por um total de 5 minutos (300 segundos) para que a pilha alcance qualquer um dos estados especificados. Nesse exemplo, o estado é atingido antes do tempo limite e, portanto, o objeto da pilha é retornado ao pipeline.

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status
 CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Saída:

```
Capabilities      : {CAPABILITY_IAM}
ChangeSetId       :
CreationTime      : 6/1/2017 9:29:33 AM
Description       : AWS CloudFormation Sample Template
 ec2_instance_with_instance_profile: Create an EC2 instance with an associated
 instance profile. **WARNING** This template creates one or more Amazon EC2
                    instances and an Amazon SQS queue. You will be billed for the
 AWS resources used if you create a stack from this template.
DisableRollback   : False
LastUpdatedTime   : 1/1/0001 12:00:00 AM
NotificationARNs  : {}
Outputs           : {}
Parameters        : {}
RoleARN           :
StackId           : arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/7ea87b50-46e7-11e7-9c9b-503a90a9c4d1
StackName         : MyStack
StackStatus       : CREATE_COMPLETE
StackStatusReason :
Tags              : {}
TimeoutInMinutes  : 0
```

Exemplo 3: esse exemplo mostra a saída de erro quando uma pilha não atinge um dos estados solicitados dentro do tempo limite (nesse caso, o tempo padrão de 60 segundos).

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Saída:

```
Wait-CFNStack : Timed out after 60 seconds waiting for CloudFormation
 stack MyStack in region us-west-2 to reach one of state(s):
 UPDATE_ROLLBACK_COMPLETE,CREATE_COMPLETE,ROLLBACK_COMPLETE,UPDATE_COMPLETE
```

```
At line:1 char:1
+ Wait-CFNStack -StackName MyStack -State CREATE_COMPLETE,ROLLBACK_COMPLETE
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation:
 (Amazon.PowerShe...tCFNStackCmdlet:WaitCFNStackCmdlet) [Wait-CFNStack],
 InvalidOperationException
    + FullyQualifiedErrorId :
 InvalidOperationException,Amazon.PowerShell.Cmdlets.CFN.WaitCFNStackCmdlet
```

- Para obter detalhes sobre a API, consulte Wait-CFNStack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

CloudFront exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com CloudFront.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-CFCloudFrontOriginAccessIdentity`

O código de exemplo a seguir mostra como usar `Get-CFCloudFrontOriginAccessIdentity`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna uma identidade de acesso de CloudFront origem específica da Amazon, especificada pelo parâmetro -Id. Embora o parâmetro -Id não seja obrigatório, se você não o especificar, nenhum resultado será retornado.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Saída:

```
    CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
    -----------------------------------    --
-----------------
    Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- Para obter detalhes sobre a API, consulte GetCloudFrontOriginAccessIdentity Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFCloudFrontOriginAccessIdentityConfig`

O código de exemplo a seguir mostra como usar `Get-CFCloudFrontOriginAccessIdentityConfig`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna informações de configuração sobre uma única identidade de acesso de CloudFront origem da Amazon, especificada pelo parâmetro -Id. Ocorrem erros se nenhum parâmetro -Id for especificado.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Saída:

```
    CallerReference                                          Comment
    ---------------                                          -------
    mycallerreference: 2/1/2011 1:16:32 PM                   Caller reference:
2/1/2011 1:16:32 PM
```

- Para obter detalhes sobre a API, consulte GetCloudFrontOriginAccessIdentityConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFCloudFrontOriginAccessIdentityList`

O código de exemplo a seguir mostra como usar `Get-CFCloudFrontOriginAccessIdentityList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna uma lista de identidades de acesso de CloudFront origem da Amazon. Como o MaxItem parâmetro - especifica um valor de 2, os resultados incluem duas identidades.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Saída:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- Para obter detalhes sobre a API, consulte ListCloudFrontOriginAccessIdentities Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFDistribution`

O código de exemplo a seguir mostra como usar `Get-CFDistribution`.

Ferramentas para PowerShell V5

Exemplo 1: recupera as informações de uma distribuição específica.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Para obter detalhes sobre a API, consulte GetDistribution Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFDistributionConfig`

O código de exemplo a seguir mostra como usar `Get-CFDistributionConfig`.

Ferramentas para PowerShell V5

Exemplo 1: recupera a configuração de uma distribuição específica.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Para obter detalhes sobre a API, consulte GetDistributionConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFDistributionList`

O código de exemplo a seguir mostra como usar `Get-CFDistributionList`.

Ferramentas para PowerShell V5

Exemplo 1: retorno de distribuições.

```
Get-CFDistributionList
```

- Para obter detalhes sobre a API, consulte ListDistributions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-CFDistribution`

O código de exemplo a seguir mostra como usar `New-CFDistribution`.

Ferramentas para PowerShell V5

Exemplo 1: Cria uma CloudFront distribuição básica, configurada com registro e armazenamento em cache.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
      -Logging_Prefix "help/" `
```

```
      -DistributionConfig_CallerReference Client1 `
      -DistributionConfig_DefaultRootObject index.html `
      -DefaultCacheBehavior_TargetOriginId $origin.Id `
      -ForwardedValues_QueryString $true `
      -Cookies_Forward all `
      -WhitelistedNames_Quantity 0 `
      -TrustedSigners_Enabled $false `
      -TrustedSigners_Quantity 0 `
      -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
      -DefaultCacheBehavior_MinTTL 1000 `
      -DistributionConfig_PriceClass "PriceClass_All" `
      -CacheBehaviors_Quantity 0 `
      -Aliases_Quantity 0
```

- Para obter detalhes sobre a API, consulte CreateDistribution Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-CFInvalidation

O código de exemplo a seguir mostra como usar `New-CFInvalidation`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma nova invalidação em uma distribuição com um ID de EXAMPLENSTXAXE. CallerReference É um ID exclusivo escolhido pelo usuário; nesse caso, é usado um carimbo de data e hora representando 15 de maio de 2019 às 9h. A variável $Paths armazena três caminhos para arquivos de imagem e mídia que o usuário não deseja como parte do cache distribuído. O valor do Paths_Quantity parâmetro - é o número total de caminhos especificados no Paths_Item parâmetro -.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
 3
```

Saída:

```
Invalidation                        Location

------------                        --------
```

```
Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- Para obter detalhes sobre a API, consulte CreateInvalidation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-CFSignedCookie

O código de exemplo a seguir mostra como usar `New-CFSignedCookie`.

Ferramentas para PowerShell V5

Exemplo 1: cria um cookie assinado para o recurso especificado usando uma política predefinida. O cookie tem validade de um ano.

```
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params
```

Saída:

```
Expires
-------
[CloudFront-Expires, 1472227284]
```

Exemplo 2: cria um cookie assinado para os recursos especificados usando uma política personalizada. O cookie tem validade de 24 horas e expira uma semana depois.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=$start.AddDays(7)
    "ActiveFrom"=$start
}
```

```
New-CFSignedCookie @params
```

Saída:

```
Policy
------
[CloudFront-Policy, eyJTd...wIjo...
```

Exemplo 3: cria um cookie assinado para os recursos especificados usando uma política personalizada. O cookie tem validade de 24 horas e expira uma semana depois. O acesso aos recursos é restrito ao intervalo de IP especificado.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=$start.AddDays(7)
    "ActiveFrom"=$start
 "IpRange"="192.0.2.0/24"
}

New-CFSignedCookie @params
```

Saída:

```
Policy
                                                                  ------

                                     [CloudFront-Policy, eyJTd...wIjo...
```

- Para obter detalhes sobre a API, consulte New-CFSignedCookie Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-CFSignedUrl`

O código de exemplo a seguir mostra como usar `New-CFSignedUrl`.

Ferramentas para PowerShell V5

Exemplo 1: cria um URL assinado para o recurso especificado usando uma política predefinida. O URL tem validade de uma hora. Um System.Uri objeto contendo o URL assinado é emitido para o pipeline.

```
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

Exemplo 2: cria um URL assinado para o recurso especificado usando uma política personalizada. O URL tem validade a partir de 24 horas e expira uma semana depois.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddDays(7)
    "ActiveFrom"=$start
}
New-CFSignedUrl @params
```

Exemplo 3: cria um URL assinado para o recurso especificado usando uma política personalizada. O URL tem validade a partir de 24 horas e expira uma semana depois. O acesso ao recurso é restrito ao intervalo de IP especificado.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddDays(7)
    "ActiveFrom"=$start
    "IpRange"="192.0.2.0/24"
}
New-CFSignedUrl @params
```

- Para obter detalhes sobre a API, consulte New-CFSignedUrl Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

CloudTrail exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com CloudTrail.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Find-CTEvent`

O código de exemplo a seguir mostra como usar `Find-CTEvent`.

Ferramentas para PowerShell V5

Exemplo 1: retorna todos os eventos que ocorreram nos últimos sete dias. Por padrão, o cmdlet faz automaticamente várias chamadas para entregar todos os eventos, saindo quando o serviço indica que não há mais dados disponíveis.

```
Find-CTEvent
```

Exemplo 2: retorna todos os eventos que ocorreram nos últimos sete dias, especificando uma região que não é o padrão atual do shell.

```
Find-CTEvent -Region eu-central-1
```

Exemplo 3: retorna todos os eventos associados à chamada da RunInstances API.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Exemplo 4: retorna os primeiros cinco eventos disponíveis.

```
Find-CTEvent -MaxResult 5
```

- Para obter detalhes sobre a API, consulte LookupEvents Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CTTrail`

O código de exemplo a seguir mostra como usar `Get-CTTrail`.

Ferramentas para PowerShell V5

Exemplo 1: recupera as configurações de todas as trilhas associadas à região atual de sua conta.

```
Get-CTTrail
```

Exemplo 2: retorna as configurações das trilhas especificadas.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Exemplo 3: retorna as configurações das trilhas especificadas que foram criadas em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt, eu-central-1).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- Para obter detalhes sobre a API, consulte DescribeTrails Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CTTrailStatus`

O código de exemplo a seguir mostra como usar `Get-CTTrailStatus`.

Ferramentas para PowerShell V5

Exemplo 1: retorna as informações de status da trilha com o nome ExampleTrail 'my'. Os dados retornados incluem informações sobre erros de envio, erros do Amazon SNS e do Amazon S3, além dos horários de início e término do registro em log da trilha. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Get-CTTrailStatus -Name myExampleTrail
```

Exemplo 2: retorna as informações de status de uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt, eu-central-1).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes sobre a API, consulte GetTrailStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-CTTrail`

O código de exemplo a seguir mostra como usar `New-CTTrail`.

Ferramentas para PowerShell V5

Exemplo 1: cria uma trilha que usará o bucket “amzn-s3-demo-bucket” para armazenamento de arquivos de log.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

Exemplo 2: cria uma trilha que usará o bucket “amzn-s3-demo-bucket” para armazenamento de arquivos de log. Os objetos do S3 que representam os logs terão um prefixo de chave comum de “mylogs”. Quando novos registros forem entregues ao bucket, uma notificação será enviada para o tópico do SNS “mlog-deliverytopic”. Este exemplo usa o nivelamento para fornecer os valores dos parâmetros ao cmdlet.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
```

```
New-CTTrail @params
```

- Para obter detalhes sobre a API, consulte CreateTrail Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CTTrail`

O código de exemplo a seguir mostra como usar `Remove-CTTrail`.

Ferramentas para PowerShell V5

Exemplo 1: exclui a trilha especificada. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a confirmação, adicione o parâmetro -Force.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Para obter detalhes sobre a API, consulte DeleteTrail Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-CTLogging`

O código de exemplo a seguir mostra como usar `Start-CTLogging`.

Ferramentas para PowerShell V5

Exemplo 1: inicia a gravação das chamadas de AWS API e a entrega do arquivo de log para a trilha chamada ExampleTrail 'my'. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Start-CTLogging -Name myExampleTrail
```

Exemplo 2: inicia a gravação de chamadas de AWS API e entrega de arquivos de log para uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes sobre a API, consulte StartLogging Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-CTLogging`

O código de exemplo a seguir mostra como usar `Stop-CTLogging`.

Ferramentas para PowerShell V5

Exemplo 1: suspende a gravação de chamadas de AWS API e a entrega do arquivo de log para a trilha chamada 'my'ExampleTrail. Este exemplo pressupõe que a trilha foi criada na mesma região do shell padrão atual.

```
Stop-CTLogging -Name myExampleTrail
```

Exemplo 2: suspende a gravação de chamadas de AWS API e a entrega de arquivos de log para uma trilha que foi criada em uma região diferente do padrão atual do shell (nesse caso, a região de Frankfurt (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Para obter detalhes sobre a API, consulte StopLogging Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CTTrail`

O código de exemplo a seguir mostra como usar `Update-CTTrail`.

Ferramentas para PowerShell V5

Exemplo 1: atualiza a trilha especificada para que os eventos globais do serviço (como os do IAM) sejam registrados e altera o prefixo de chave comum dos arquivos de log futuros para “globallogs”.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Exemplo 2: atualiza a trilha especificada para que as notificações sobre novas entregas de logs sejam enviadas ao tópico do SNS especificado.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Exemplo 3: atualiza a trilha especificada para que os logs sejam entregues em um bucket diferente.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Para obter detalhes sobre a API, consulte UpdateTrail Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

CloudWatch exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-CWDashboard`

O código de exemplo a seguir mostra como usar `Get-CWDashboard`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o ARN do corpo do painel especificado.

```
Get-CWDashboard -DashboardName Dashboard1
```

Saída:

```
DashboardArn                                        DashboardBody
------------                                        -------------
```

```
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Para obter detalhes sobre a API, consulte GetDashboard Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CWDashboardList`

O código de exemplo a seguir mostra como usar `Get-CWDashboardList`.

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de painéis para sua conta.

```
Get-CWDashboardList
```

Saída:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Exemplo 2: retorna a coleção de painéis para sua conta cujos nomes começam com o prefixo “dev”.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Para obter detalhes sobre a API, consulte ListDashboards Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CWDashboard`

O código de exemplo a seguir mostra como usar `Remove-CWDashboard`.

Ferramentas para PowerShell V5

Exemplo 1: exclui o painel especificado, solicitando uma confirmação antes de continuar. Para ignorar a confirmação, adicione a opção -Force para o comando.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Para obter detalhes sobre a API, consulte DeleteDashboards Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-CWDashboard`

O código de exemplo a seguir mostra como usar `Write-CWDashboard`.

Ferramentas para PowerShell V5

Exemplo 1: cria ou atualiza o painel denominado “Dashboard1” para incluir dois widgets de métricas lado a lado.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
                      "CPUUtilization",
                      "InstanceId",
                      "i-012345"
                   ]
                ],
                "period":300,
                "stat":"Average",
                "region":"us-east-1",
                "title":"EC2 Instance CPU"
             }
        },
        {
             "type":"metric",
             "x":12,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
```

```
                "metrics":[
                   [
                      "AWS/S3",
                      "BucketSizeBytes",
                      "BucketName",
                      "amzn-s3-demo-bucket"
                   ]
                ],
                "period":86400,
                "stat":"Maximum",
                "region":"us-east-1",
                "title":"amzn-s3-demo-bucket bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Exemplo 2: cria ou atualiza o painel, redirecionando o conteúdo que descreve o painel para o cmdlet.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Para obter detalhes sobre a API, consulte PutDashboard Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-CWMetricData`

O código de exemplo a seguir mostra como usar `Write-CWMetricData`.

Ferramentas para PowerShell V5

Exemplo 1: Cria um novo MetricDatum objeto e o grava nas CloudWatch métricas da Amazon Web Services.

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- Para obter detalhes sobre a API, consulte PutMetricData Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

CodeCommit exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com CodeCommit.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-CCBranch`

O código de exemplo a seguir mostra como usar `Get-CCBranch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a ramificação especificada para o repositório especificado.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Saída:

```
BranchName                                CommitId
----------                                --------
MyNewBranch                               7763222d...561fc9c9
```

- Para obter detalhes sobre a API, consulte GetBranch Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CCBranchList`

O código de exemplo a seguir mostra como usar `Get-CCBranchList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de nomes de ramificações para o repositório especificado.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Saída:

```
master
MyNewBranch
```

- Para obter detalhes sobre a API, consulte ListBranches Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CCRepository`

O código de exemplo a seguir mostra como usar `Get-CCRepository`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações para o repositório especificado.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Saída:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/8/2015 3:21:33 PM
DefaultBranch         :
LastModifiedDate      : 9/8/2015 3:21:33 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName        : MyDemoRepo
```

- Para obter detalhes sobre a API, consulte GetRepository Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CCRepositoryBatch`

O código de exemplo a seguir mostra como usar `Get-CCRepositoryBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo confirma quais dos repositórios especificados foram e não foram encontrados.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Saída:

```
Repositories                            RepositoriesNotFound
------------                            --------------------
{MyDemoRepo, MyNewRepo}                {AMissingRepo}
```

- Para obter detalhes sobre a API, consulte BatchGetRepositories Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CCRepositoryList`

O código de exemplo a seguir mostra como usar `Get-CCRepositoryList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os repositórios em ordem crescente por nome do repositório.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Saída:

```
RepositoryId                            RepositoryName
------------                            --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- Para obter detalhes sobre a API, consulte ListRepositories Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-CCBranch`

O código de exemplo a seguir mostra como usar `New-CCBranch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma ramificação com o nome especificado para o repositório especificado e o ID do commit especificado.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- Para obter detalhes sobre a API, consulte CreateBranch Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-CCRepository`

O código de exemplo a seguir mostra como usar `New-CCRepository`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um repositório com o nome especificado e a descrição especificada.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Saída:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/18/2015 4:13:25 PM
DefaultBranch         :
LastModifiedDate      : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName        : MyDemoRepo
```

- Para obter detalhes sobre a API, consulte CreateRepository Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CCRepository`

O código de exemplo a seguir mostra como usar `Remove-CCRepository`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui forçosamente o repositório especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o repositório sem um aviso.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Saída:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Para obter detalhes sobre a API, consulte DeleteRepository Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CCDefaultBranch`

O código de exemplo a seguir mostra como usar `Update-CCDefaultBranch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera a ramificação padrão do repositório especificado para a ramificação especificada.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Para obter detalhes sobre a API, consulte UpdateDefaultBranch Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CCRepositoryDescription`

O código de exemplo a seguir mostra como usar `Update-CCRepositoryDescription`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera a descrição do repositório especificado.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- Para obter detalhes sobre a API, consulte UpdateRepositoryDescription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CCRepositoryName`

O código de exemplo a seguir mostra como usar `Update-CCRepositoryName`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o nome do repositório especificado.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Para obter detalhes sobre a API, consulte UpdateRepositoryName Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

CodeDeploy exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com CodeDeploy.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-CDOnPremiseInstanceTag`

O código de exemplo a seguir mostra como usar `Add-CDOnPremiseInstanceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona uma tag de instância on-premises com a chave e o valor especificados para a instância local especificada.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
 "Value" = "CodeDeployDemo-OnPrem"}
```

- Para obter detalhes sobre a API, consulte AddTagsToOnPremisesInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDApplication`

O código de exemplo a seguir mostra como usar `Get-CDApplication`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a aplicação especificada.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Saída:

```
ApplicationId                           ApplicationName              CreateTime
         LinkedToGitHub
-------------                           ---------------              ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
```

- Para obter detalhes sobre a API, consulte GetApplication Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDApplicationBatch`

O código de exemplo a seguir mostra como usar `Get-CDApplicationBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as aplicações especificadas.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Saída:

```
ApplicationId                           ApplicationName              CreateTime
         LinkedToGitHub
-------------                           ---------------              ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication  8/13/2015
 5:53:26 PM    False
```

- Para obter detalhes sobre a API, consulte BatchGetApplications Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDApplicationList`

O código de exemplo a seguir mostra como usar `Get-CDApplicationList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista das aplicações disponíveis.

```
Get-CDApplicationList
```

Saída:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Para obter detalhes sobre a API, consulte ListApplications Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDApplicationRevision`

O código de exemplo a seguir mostra como usar `Get-CDApplicationRevision`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a revisão da aplicação especificada.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
 -S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Saída:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- Para obter detalhes sobre a API, consulte GetApplicationRevision Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDApplicationRevisionList`

O código de exemplo a seguir mostra como usar `Get-CDApplicationRevisionList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as revisões disponíveis para a aplicação especificada.

```
ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
 CodeDeployDemoApplication -Deployed Ignore)) {
>>   If ($revision.RevisionType -Eq "S3") {
>>     Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
 + ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
 $revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>>   }
>>   If ($revision.RevisionType -Eq "GitHub") {
>>     Write-Output ("Type = GitHub, CommitId = " +
 $revision.GitHubLocation.CommitId + ", Repository = " +
 $revision.GitHubLocation.Repository)
>>   }
>> }
>>
```

Saída:

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Para obter detalhes sobre a API, consulte ListApplicationRevisions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeployment`

O código de exemplo a seguir mostra como usar `Get-CDDeployment`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações de resumo sobre a implantação especificada.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Saída:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

Exemplo 2: esse exemplo obtém informações sobre o status das instâncias que estão participando da implantação especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Saída:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Exemplo 3: esse exemplo obtém informações sobre a revisão da aplicação para a implantação especificada.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Saída:

```
Bucket     : amzn-s3-demo-bucket
```

```
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- Para obter detalhes sobre a API, consulte GetDeployment Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeploymentBatch`

O código de exemplo a seguir mostra como usar `Get-CDDeploymentBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as implantações especificadas.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Saída:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded

ApplicationName               : CodePipelineDemoApplication
CompleteTime                  : 7/23/2015 6:07:30 PM
CreateTime                    : 7/23/2015 6:06:29 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodePipelineDemoFleet
DeploymentId                  : d-RR0T5KTEX
```

```
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

- Para obter detalhes sobre a API, consulte BatchGetDeployments Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeploymentConfig`

O código de exemplo a seguir mostra como usar `Get-CDDeploymentConfig`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações de resumo sobre a configuração da implantação especificada.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Saída:

```
CreateTime              DeploymentConfigId                      DeploymentConfigName
    MinimumHealthyHosts
----------              ------------------                      --------------------
    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE    ThreeQuartersHealthy
    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Exemplo 2: esse exemplo obtém informações sobre a configuração da implantação especificada.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Saída:

```
Type             Value
----             -----
```

```
FLEET_PERCENT    75
```

- Para obter detalhes sobre a API, consulte GetDeploymentConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeploymentConfigList`

O código de exemplo a seguir mostra como usar `Get-CDDeploymentConfigList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista das configurações de implantação disponíveis.

```
Get-CDDeploymentConfigList
```

Saída:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Para obter detalhes sobre a API, consulte ListDeploymentConfigs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeploymentGroup`

O código de exemplo a seguir mostra como usar `Get-CDDeploymentGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre o grupo de implantação especificado.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Saída:

```
ApplicationName              : CodeDeployDemoApplication
AutoScalingGroups            : {}
DeploymentConfigName         : CodeDeployDefault.OneAtATime
```

```
DeploymentGroupId            : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName          : CodeDeployDemoFleet
Ec2TagFilters                : {Name}
OnPremisesInstanceTagFilters : {}
ServiceRoleArn               : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- Para obter detalhes sobre a API, consulte GetDeploymentGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeploymentGroupList`

O código de exemplo a seguir mostra como usar `Get-CDDeploymentGroupList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de grupos de implantação para a aplicação especificada.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Saída:

```
ApplicationName              DeploymentGroups
 NextToken
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Para obter detalhes sobre a API, consulte ListDeploymentGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeploymentInstance`

O código de exemplo a seguir mostra como usar `Get-CDDeploymentInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a instância especificada para a implantação especificada.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Saída:

```
DeploymentId    : d-QZMRGSTEX
InstanceId      : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt   : 7/23/2015 11:25:24 PM
LifecycleEvents : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status          : Succeeded
```

- Para obter detalhes sobre a API, consulte GetDeploymentInstance Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeploymentInstanceList`

O código de exemplo a seguir mostra como usar `Get-CDDeploymentInstanceList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de IDs de instância para a implantação especificada.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Saída:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Para obter detalhes sobre a API, consulte ListDeploymentInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDDeploymentList`

O código de exemplo a seguir mostra como usar `Get-CDDeploymentList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de IDs de implantação para a aplicação e o grupo de implantação especificados.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName
 CodeDeployDemoFleet
```

Saída:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- Para obter detalhes sobre a API, consulte ListDeployments Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDOnPremiseInstance`

O código de exemplo a seguir mostra como usar `Get-CDOnPremiseInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre a instância on-premises especificada.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Saída:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Para obter detalhes sobre a API, consulte GetOnPremisesInstance Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDOnPremiseInstanceBatch`

O código de exemplo a seguir mostra como usar `Get-CDOnPremiseInstanceBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as instâncias on-premises especificadas.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

Saída:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName   : AssetTag12010298EX-2
RegisterTime   : 4/3/2015 6:38:52 PM
Tags           : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Para obter detalhes sobre a API, consulte BatchGetOnPremisesInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CDOnPremiseInstanceList`

O código de exemplo a seguir mostra como usar `Get-CDOnPremiseInstanceList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista de nomes de instâncias on-premises disponíveis.

```
Get-CDOnPremiseInstanceList
```

Saída:

```
AssetTag12010298EX
```

```
AssetTag12010298EX-2
```

- Para obter detalhes sobre a API, consulte ListOnPremisesInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-CDApplication

O código de exemplo a seguir mostra como usar `New-CDApplication`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma aplicação com o nome especificado.

```
New-CDApplication -ApplicationName MyNewApplication
```

Saída:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Para obter detalhes sobre a API, consulte CreateApplication Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-CDDeployment

O código de exemplo a seguir mostra como usar `New-CDDeployment`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma implantação para a aplicação e o grupo de implantação especificados com a configuração de implantação e a revisão da aplicação especificadas.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

Saída:

```
d-ZHROG7UEX
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True
 -S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

Saída:

```
d-ZHROG7UEX
```

- Para obter detalhes sobre a API, consulte CreateDeployment Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-CDDeploymentConfig

O código de exemplo a seguir mostra como usar `New-CDDeploymentConfig`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma configuração de implantação com o nome e o comportamento especificados.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Saída:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Para obter detalhes sobre a API, consulte CreateDeploymentConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-CDDeploymentGroup

O código de exemplo a seguir mostra como usar `New-CDDeploymentGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um grupo de implantação com o nome, o grupo do Auto Scaling, a configuração de implantação, a tag e o perfil de serviço especificados para a aplicação especificada.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

Saída:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

Saída:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Para obter detalhes sobre a API, consulte CreateDeploymentGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-CDApplicationRevision`

O código de exemplo a seguir mostra como usar `Register-CDApplicationRevision`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma revisão da aplicação com o local do Amazon S3 especificado para a aplicação especificada.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key aws-
codedeploy_linux-master.zip -Revision_RevisionType S3
```

- Para obter detalhes sobre a API, consulte RegisterApplicationRevision Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-CDOnPremiseInstance`

O código de exemplo a seguir mostra como usar `Register-CDOnPremiseInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma instância on-premises com o nome e o usuário do IAM especificados.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Para obter detalhes sobre a API, consulte RegisterOnPremisesInstance Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CDApplication`

O código de exemplo a seguir mostra como usar `Remove-CDApplication`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a aplicação com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a aplicação sem um aviso.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Para obter detalhes sobre a API, consulte DeleteApplication Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CDDeploymentConfig`

O código de exemplo a seguir mostra como usar `Remove-CDDeploymentConfig`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a configuração de implantação com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a configuração de implantação sem um aviso.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Para obter detalhes sobre a API, consulte DeleteDeploymentConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CDDeploymentGroup`

O código de exemplo a seguir mostra como usar `Remove-CDDeploymentGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o grupo de implantação com o nome especificado para a aplicação especificada. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o grupo de implantação sem um aviso.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- Para obter detalhes sobre a API, consulte DeleteDeploymentGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CDOnPremiseInstanceTag`

O código de exemplo a seguir mostra como usar `Remove-CDOnPremiseInstanceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a tag especificada para a instância on-premises com o nome especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a tag sem um aviso.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Para obter detalhes sobre a API, consulte RemoveTagsFromOnPremisesInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-CDDeployment`

O código de exemplo a seguir mostra como usar `Stop-CDDeployment`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo tenta interromper a implantação com o ID de implantação especificado.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Saída:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- Para obter detalhes sobre a API, consulte StopDeployment Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-CDOnPremiseInstance`

O código de exemplo a seguir mostra como usar `Unregister-CDOnPremiseInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro de uma instância on-premises com o nome especificado.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Para obter detalhes sobre a API, consulte DeregisterOnPremisesInstance Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CDApplication`

O código de exemplo a seguir mostra como usar `Update-CDApplication`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o nome da aplicação especificada.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- Para obter detalhes sobre a API, consulte UpdateApplication Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CDDeploymentGroup`

O código de exemplo a seguir mostra como usar `Update-CDDeploymentGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o nome do grupo de implantação especificado para a aplicação especificada.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Exemplo 2: Este exemplo mostra como especificar grupos de tags de instância do EC2 pelos quais uma instância deve ser identificada para que seja incluída no ambiente substituto de uma blue/green implantação.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- Para obter detalhes sobre a API, consulte UpdateDeploymentGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

CodePipeline exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com CodePipeline.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Confirm-CPJob`

O código de exemplo a seguir mostra como usar `Confirm-CPJob`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém o status do trabalho especificado.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Saída:

```
Value
-----
InProgress
```

- Para obter detalhes sobre a API, consulte AcknowledgeJob Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-CPStageTransition`

O código de exemplo a seguir mostra como usar `Disable-CPStageTransition`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desativa a transição de entrada para o estágio especificado no pipeline especificado.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- Para obter detalhes sobre a API, consulte DisableStageTransition Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-CPStageTransition`

O código de exemplo a seguir mostra como usar `Enable-CPStageTransition`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo ativa a transição de entrada para o estágio especificado no pipeline especificado.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- Para obter detalhes sobre a API, consulte EnableStageTransition Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CPActionType`

O código de exemplo a seguir mostra como usar `Get-CPActionType`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre as ações disponíveis para o proprietário especificado.

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {
```

```
  Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +
 $actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version = " +
 $actionType.Id.Version + ":")
  Write-Output ("  ActionConfigurationProperties:")
  ForEach ($acp in $actionType.ActionConfigurationProperties) {
    Write-Output ("    For " + $acp.Name + ":")
    Write-Output ("      Description = " + $acp.Description)
    Write-Output ("      Key = " + $acp.Key)
    Write-Output ("      Queryable = " + $acp.Queryable)
    Write-Output ("      Required = " + $acp.Required)
    Write-Output ("      Secret = " + $acp.Secret)
  }
  Write-Output ("  InputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.InputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.InputArtifactDetails.MinimumCount)
  Write-Output ("  OutputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.OutputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.OutputArtifactDetails.MinimumCount)
  Write-Output ("  Settings:")
  Write-Output ("    EntityUrlTemplate = " + $actionType.Settings.EntityUrlTemplate)
  Write-Output ("    ExecutionUrlTemplate = " +
 $actionType.Settings.ExecutionUrlTemplate)
}
```

Saída:

```
For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS Elastic Beanstalk Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For EnvironmentName:
      Description = The AWS Elastic Beanstalk Environment name
      Key = True
      Queryable = False
      Required = True
```

```
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS CodeDeploy Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For DeploymentGroupName:
      Description = The AWS CodeDeploy Deployment Group name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
applications/{Config:ApplicationName}/deployment-groups/{Config:DeploymentGroupName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}
```

- Para obter detalhes sobre a API, consulte ListActionTypes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CPActionableJobList`

O código de exemplo a seguir mostra como usar `Get-CPActionableJobList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações sobre todos os trabalhos acionáveis para a categoria de ação, o proprietário, o provedor, a versão, os parâmetros de consulta especificados.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Saída:

```
AccountId       Data                                Id
         Nonce
---------       ----                                --
         -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData   0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- Para obter detalhes sobre a API, consulte PollForJobs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CPJobDetail`

O código de exemplo a seguir mostra como usar `Get-CPJobDetail`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações gerais sobre o trabalho especificado.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Saída:

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o trabalho especificado.

```
$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output ("  AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output ("  Configuration:")
ForEach ($key in $jobData.ActionConfiguration.Keys) {
  $value = $jobData.ActionConfiguration.$key
  Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
 $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
  Write-Output ("    " + $ia.Name)
}
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
  Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)
```

Saída:

```
For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
```

```
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
    MyApp
  OutputArtifacts:
    MyAppBuild
  PipelineContext:
    Name = Build
    PipelineName = CodePipelineDemo
    Stage = Build
```

- Para obter detalhes sobre a API, consulte GetJobDetails Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CPPipeline`

O código de exemplo a seguir mostra como usar `Get-CPPipeline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações gerais sobre o pipeline especificado.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o pipeline especificado.

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
```

```
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
Write-Output ("  Location = " + $pipeline.ArtifactStore.Location)
Write-Output ("  Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
  Write-Output ("  Name = " + $stage.Name)
  Write-Output ("    Actions:")
  ForEach ($action in $stage.Actions) {
    Write-Output ("      Name = " + $action.Name)
 Write-Output ("        Category = " + $action.ActionTypeId.Category)
 Write-Output ("        Owner = " + $action.ActionTypeId.Owner)
 Write-Output ("        Provider = " + $action.ActionTypeId.Provider)
 Write-Output ("        Version = " + $action.ActionTypeId.Version)
 Write-Output ("        Configuration:")
 ForEach ($key in $action.Configuration.Keys) {
   $value = $action.Configuration.$key
   Write-Output ("          " + $key + " = " + $value)
 }
 Write-Output ("        InputArtifacts:")
 ForEach ($ia in $action.InputArtifacts) {
   Write-Output ("          " + $ia.Name)
 }
 ForEach ($oa in $action.OutputArtifacts) {
   Write-Output ("          " + $oa.Name)
 }
 Write-Output ("        RunOrder = " + $action.RunOrder)
  }
}
```

Saída:

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = amzn-s3-demo-bucket
  Type = S3
Stages:
  Name = Source
    Actions:
      Name = Source
        Category = Source
```

```
        Owner = ThirdParty
        Provider = GitHub
        Version = 1
        Configuration:
          Branch = master
          OAuthToken = ****
          Owner = my-user-name
          Repo = MyRepoName
        InputArtifacts:
          MyApp
        RunOrder = 1
  Name = Build
    Actions:
      Name = Build
        Category = Build
        Owner = Custom
        Provider = MyCustomProviderName
        Version = 1
        Configuration:
          ProjectName = MyProjectName
        InputArtifacts:
          MyApp
          MyAppBuild
        RunOrder = 1
  Name = Beta
    Actions:
      Name = CodePipelineDemoFleet
        Category = Deploy
        Owner = AWS
        Provider = CodeDeploy
        Version = 1
        Configuration:
          ApplicationName = CodePipelineDemoApplication
          DeploymentGroupName = CodePipelineDemoFleet
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
  Name = TestStage
    Actions:
      Name = MyJenkinsTestAction
        Category = Test
        Owner = Custom
        Provider = MyCustomTestProvider
        Version = 1
```

```
        Configuration:
          ProjectName = MyJenkinsProjectName
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
```

- Para obter detalhes sobre a API, consulte GetPipeline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CPPipelineList`

O código de exemplo a seguir mostra como usar `Get-CPPipelineList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma lista dos pipelines disponíveis.

```
Get-CPPipelineList
```

Saída:

```
Created                  Name                Updated                  Version
-------                  ----                -------                  -------
8/13/2015 10:17:54 PM    CodePipelineDemo    8/13/2015 10:17:54 PM    3
7/8/2015 2:41:53 AM      MyFirstPipeline     7/22/2015 9:06:37 PM     7
```

- Para obter detalhes sobre a API, consulte ListPipelines Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CPPipelineState`

O código de exemplo a seguir mostra como usar `Get-CPPipelineState`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém informações gerais sobre os estágios do pipeline especificado.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Saída:

```
Created         : 8/13/2015 10:17:54 PM
PipelineName    : CodePipelineDemo
PipelineVersion : 1
StageStates     : {Source, Build, Beta, TestStage}
Updated         : 8/13/2015 10:17:54 PM
```

Exemplo 2: esse exemplo obtém informações detalhadas sobre o estado do pipeline especificado.

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
  Write-Output ("For " + $stageState.StageName + ":")
  Write-Output ("  InboundTransitionState:")
  Write-Output ("    DisabledReason = " +
 $stageState.InboundTransitionState.DisabledReason)
  Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
  Write-Output ("    LastChangedAt = " +
 $stageState.InboundTransitionState.LastChangedAt)
  Write-Output ("    LastChangedBy = " +
 $stageState.InboundTransitionState.LastChangedBy)
  Write-Output ("  ActionStates:")
  ForEach ($actionState in $stageState.ActionStates) {
    Write-Output ("    For " + $actionState.ActionName + ":")
 Write-Output ("      CurrentRevision:")
    Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
 Write-Output ("        RevisionChangeId = " +
 $actionState.CurrentRevision.RevisionChangeId)
 Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
 Write-Output ("      EntityUrl = " + $actionState.EntityUrl)
 Write-Output ("      LatestExecution:")
    Write-Output ("        ErrorDetails:")
    Write-Output ("          Code = " +
 $actionState.LatestExecution.ErrorDetails.Code)
 Write-Output ("          Message = " +
 $actionState.LatestExecution.ErrorDetails.Message)
 Write-Output ("        ExternalExecutionId = " +
 $actionState.LatestExecution.ExternalExecutionId)
 Write-Output ("        ExternalExecutionUrl = " +
 $actionState.LatestExecution.ExternalExecutionUrl)
 Write-Output ("        LastStatusChange = " +
 $actionState.LatestExecution.LastStatusChange)
 Write-Output ("        PercentComplete = " +
 $actionState.LatestExecution.PercentComplete)
 Write-Output ("        Status = " + $actionState.LatestExecution.Status)
```

```
 Write-Output ("        Summary = " + $actionState.LatestExecution.Summary)
 Write-Output ("      RevisionUrl = " + $actionState.RevisionUrl)
  }
}
```

Saída:

```
For Source:
  InboundTransitionState:
    DisabledReason =
    Enabled =
    LastChangedAt =
    LastChangedBy =
  ActionStates:
    For Source:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/20/2015 23:28:45
        PercentComplete = 0
        Status = Succeeded
        Summary =
      RevisionUrl =
For Build:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For Build:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
```

```
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code = TimeoutError
          Message = The action failed because a job worker exceeded its time limit.
 If this is a custom action, make sure that the job worker is configured correctly.
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/21/2015 00:29:29
        PercentComplete = 0
        Status = Failed
        Summary =
      RevisionUrl =
For Beta:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For CodePipelineDemoFleet:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = d-D5LTCZXEX
        ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
        LastStatusChange = 07/08/2015 22:07:42
        PercentComplete = 0
        Status = Succeeded
        Summary = Deployment Succeeded
      RevisionUrl =
For TestStage:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
```

```
    LastChangedBy =
  ActionStates:
    For MyJenkinsTestAction25:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = 5
        ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
        LastStatusChange = 07/08/2015 22:09:03
        PercentComplete = 0
        Status = Succeeded
        Summary = Finished
      RevisionUrl =
```

- Para obter detalhes sobre a API, consulte GetPipelineState Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-CPCustomActionType`

O código de exemplo a seguir mostra como usar `New-CPCustomActionType`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma ação personalizada com as propriedades especificadas.

```
New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
 = "The name of the build project must be provided when this action is added
 to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
 = $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
 -Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
 1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1
```

Saída:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Para obter detalhes sobre a API, consulte CreateCustomActionType Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-CPPipeline

O código de exemplo a seguir mostra como usar `New-CPPipeline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um pipeline com as configurações especificadas.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
```

```
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Para obter detalhes sobre a API, consulte CreatePipeline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CPCustomActionType`

O código de exemplo a seguir mostra como usar `Remove-CPCustomActionType`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a ação personalizada especificada. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir a ação personalizada sem um aviso.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- Para obter detalhes sobre a API, consulte DeleteCustomActionType Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CPPipeline`

O código de exemplo a seguir mostra como usar `Remove-CPPipeline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o pipeline especificado. O comando solicitará a confirmação antes de continuar. Adicione o parâmetro -Force para excluir o pipeline sem um aviso.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Para obter detalhes sobre a API, consulte DeletePipeline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-CPPipelineExecution`

O código de exemplo a seguir mostra como usar `Start-CPPipelineExecution`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo começa a executar o pipeline especificado.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Para obter detalhes sobre a API, consulte StartPipelineExecution Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CPPipeline`

O código de exemplo a seguir mostra como usar `Update-CPPipeline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza o pipeline existente especificado com as configurações especificadas.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1
```

```
Update-CPPipeline -Pipeline $pipeline
```

Saída:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Para obter detalhes sobre a API, consulte UpdatePipeline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de identidade do Amazon Cognito usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon Cognito Identity.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-CGIIdentityPool`

O código de exemplo a seguir mostra como usar `Get-CGIIdentityPool`.

Ferramentas para PowerShell V5

Exemplo 1: Recupera informações sobre um banco de identidades específico por meio de seu ID.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Saída:

```
LoggedAt                       : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : CommonTests1
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 142
HttpStatusCode                 : OK
```

- Para obter detalhes sobre a API, consulte DescribeIdentityPool Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CGIIdentityPoolList`

O código de exemplo a seguir mostra como usar `Get-CGIIdentityPoolList`.

Ferramentas para PowerShell V5

Exemplo 1: Recupera uma lista de bancos de identidades existentes.

```
Get-CGIIdentityPoolList
```

Saída:

```
IdentityPoolId                                                  IdentityPoolName
--------------                                                  ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                  CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                  Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                  CommonTests13
```

- Para obter detalhes sobre a API, consulte ListIdentityPools Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CGIIdentityPoolRole`

O código de exemplo a seguir mostra como usar `Get-CGIIdentityPoolRole`.

Ferramentas para PowerShell V5

Exemplo 1: Obtém as informações sobre as funções de um banco de identidades específico.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Saída:

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- Para obter detalhes sobre a API, consulte GetIdentityPoolRoles Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-CGIIdentityPool`

O código de exemplo a seguir mostra como usar `New-CGIIdentityPool`.

Ferramentas para PowerShell V5

Exemplo 1: Cria um novo banco de identidades que permite identidades não autenticadas.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Saída:

```
LoggedAt                       : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
```

```
IdentityPoolId                  : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName                : CommonTests13
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 136
HttpStatusCode                  : OK
```

- Para obter detalhes sobre a API, consulte CreateIdentityPool Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CGIIdentityPool`

O código de exemplo a seguir mostra como usar `Remove-CGIIdentityPool`.

Ferramentas para PowerShell V5

Exemplo 1: Exclui um banco de identidades específico.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Para obter detalhes sobre a API, consulte DeleteIdentityPool Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-CGIIdentityPoolRole`

O código de exemplo a seguir mostra como usar `Set-CGIIdentityPoolRole`.

Ferramentas para PowerShell V5

Exemplo 1: Configura o banco de identidades específico para ter um perfil do IAM não autenticado.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Para obter detalhes sobre a API, consulte SetIdentityPoolRoles Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-CGIIdentityPool`

O código de exemplo a seguir mostra como usar `Update-CGIIdentityPool`.

Ferramentas para PowerShell V5

Exemplo 1: Atualiza algumas das propriedades do banco de identidades, neste caso, o nome do banco de identidades.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Saída:

```
LoggedAt                       : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : NewPoolName
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 135
HttpStatusCode                 : OK
```

- Para obter detalhes sobre a API, consulte UpdateIdentityPool Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS Config exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS Config.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-CFGResourceTag`

O código de exemplo a seguir mostra como usar `Add-CFGResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo associa a tag especificada ao recurso ARN, que é config- rule/config - rule-16iyn0 nesse caso.

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- Para obter detalhes sobre a API, consulte TagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGAggregateComplianceByConfigRuleList`

O código de exemplo a seguir mostra como usar `Get-CFGAggregateComplianceByConfigRuleList`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém os detalhes da filtragem ConfigurationAggregator 'kaju' para a regra de configuração fornecida e a 'Conformidade' expands/returns da regra.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
 -Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

Saída:

```
ComplianceContributorCount                          ComplianceType
--------------------------                          --------------
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

Exemplo 2: Este exemplo obtém detalhes do dado ConfigurationAggregator, os filtra para a conta fornecida para todas as regiões cobertas no agregador e retorna ainda mais a conformidade de todas as regras.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
 kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
 @{N="Compliance";E={$_.Compliance.ComplianceType}}
```

Saída:

```
ConfigRuleName                      Compliance
--------------                      ----------
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK NON_COMPLIANT
ec2-instance-no-public-ip           NON_COMPLIANT
desired-instance-type               NON_COMPLIANT
```

- Para obter detalhes sobre a API, consulte DescribeAggregateComplianceByConfigRules Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGAggregateComplianceDetailsByConfigRule`

O código de exemplo a seguir mostra como usar `Get-CFGAggregateComplianceDetailsByConfigRule`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo retorna os resultados da avaliação selecionando a saída com resource-id e resource-type para a regra de AWS configuração 'desired-instance-type' que estão no estado 'COMPATÍVEL' para determinada conta, agregador, região e regra de configuração

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
 EvaluationResultQualifier
```

Saída:

```
ConfigRuleName        ResourceId          ResourceType
--------------        ----------          ------------
```

```
desired-instance-type i-0f1bf2f34c5678d12 AWS::EC2::Instance
desired-instance-type i-0fd12dd3456789123 AWS::EC2::Instance
```

- Para obter detalhes sobre a API, consulte GetAggregateComplianceDetailsByConfigRule Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGAggregateConfigRuleComplianceSummary`

O código de exemplo a seguir mostra como usar `Get-CFGAggregateConfigRuleComplianceSummary`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o número de regras não compatíveis para o agregador em questão.

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
 raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

Saída:

```
CapExceeded CappedCount
----------- -----------
False       5
```

- Para obter detalhes sobre a API, consulte GetAggregateConfigRuleComplianceSummary Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGAggregateDiscoveredResourceCount`

O código de exemplo a seguir mostra como usar `Get-CFGAggregateDiscoveredResourceCount`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna a contagem de recursos do agregador especificado filtrado para a região us-east-1.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

Saída:

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
---------- -------------------- --------- -----------------------
           {}                              455
```

Exemplo 2: esse exemplo retorna a contagem de recursos agrupada por RESOURCE_TYPE para a região filtrada de determinado agregador.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
   Select-Object - expandProperty GroupedResourceCounts
```

Saída:

```
GroupName                          ResourceCount
---------                          -------------
AWS::CloudFormation::Stack         12
AWS::CloudFront::Distribution      1
AWS::CloudTrail::Trail             1
AWS::DynamoDB::Table               1
AWS::EC2::EIP                      2
AWS::EC2::FlowLog                  2
AWS::EC2::InternetGateway          4
AWS::EC2::NatGateway               2
AWS::EC2::NetworkAcl               4
AWS::EC2::NetworkInterface         12
AWS::EC2::RouteTable               13
AWS::EC2::SecurityGroup            18
AWS::EC2::Subnet                   16
AWS::EC2::VPC                      4
AWS::EC2::VPCEndpoint              2
AWS::EC2::VPCPeeringConnection     1
AWS::IAM::Group                    2
AWS::IAM::Policy                   51
AWS::IAM::Role                     78
AWS::IAM::User                     7
AWS::Lambda::Function              3
AWS::RDS::DBSecurityGroup          1
AWS::S3::Bucket                    3
AWS::SSM::AssociationCompliance    107
AWS::SSM::ManagedInstanceInventory 108
```

- Para obter detalhes sobre a API, consulte GetAggregateDiscoveredResourceCounts Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGAggregateDiscoveredResourceList`

O código de exemplo a seguir mostra como usar `Get-CFGAggregateDiscoveredResourceList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna os identificadores de recursos para determinado tipo de recurso agregado no agregador “Irlanda”. Para ver a lista de tipos de recursos, consultehttps://docs.aws.amazon.com/sdkfornet/v4/apidocs/index.html? página = &tocid=ConfigService/TConfigServiceResourceType.html. Amazon_ConfigService_ResourceType

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)
```

Saída:

```
ResourceId      : arn:aws:autoscaling:eu-
west-1:123456789012:autoScalingGroup:12e3b4fc-1234-1234-
a123-1d2ba3c45678:autoScalingGroupName/asg-1
ResourceName    : asg-1
ResourceType    : AWS::AutoScaling::AutoScalingGroup
SourceAccountId : 123456789012
SourceRegion    : eu-west-1
```

Exemplo 2: esse exemplo retorna o tipo de recurso **`AwsEC2SecurityGroup`** chamado “padrão” para o agregador especificado filtrado com a região us-east-1.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

Saída:

```
ResourceId      : sg-01234bd5dbfa67c89
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
```

```
SourceRegion    : us-east-1

ResourceId      : sg-0123a4ebbf56789be
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1

ResourceId      : sg-4fc1d234
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1
```

- Para obter detalhes sobre a API, consulte ListAggregateDiscoveredResources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGAggregateResourceConfig`

O código de exemplo a seguir mostra como usar `Get-CFGAggregateResourceConfig`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o item de configuração para determinado recurso agregado e expande a configuração.

```
(Get-CFGAggregateResourceConfig -ResourceIdentifier_SourceRegion
 us-east-1 -ResourceIdentifier_SourceAccountId 123456789012 -
ResourceIdentifier_ResourceId sg-4fc1d234 -ResourceIdentifier_ResourceType
 ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
ConfigurationAggregatorName raju).Configuration | ConvertFrom-Json
```

Saída:


```
{"description":"default VPC security group","groupName":"default","ipPermissions":
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[{"groupId":"sg-4fc1d234","userId":"123456789012"}],"ipv4Ranges":
[],"ipRanges":[]},{"fromPort":3389,"ipProtocol":"tcp","ipv6Ranges":
[],"prefixListIds":[],"toPort":3389,"userIdGroupPairs":[],"ipv4Ranges":
[{"cidrIp":"54.240.197.224/29","description":"office subnet"},
{"cidrIp":"72.21.198.65/32","description":"home pc"}],"ipRanges":
["54.240.197.224/29","72.21.198.65/32"]}],"ownerId":"123456789012","groupId":"sg-4fc1d234","
```

```
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[],"ipv4Ranges":[{"cidrIp":"0.0.0.0/0"}],"ipRanges":["0.0.0.0/0"]}],"tags":
[],"vpcId":"vpc-2d1c2e34"}
```

- Para obter detalhes sobre a API, consulte GetAggregateResourceconfig-service Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGAggregateResourceConfigBatch`

O código de exemplo a seguir mostra como usar `Get-CFGAggregateResourceConfigBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo busca o item de configuração atual do recurso (identificado) presente no agregador fornecido.

```
$resIdentifier=[Amazon.ConfigService.Model.AggregateResourceIdentifier]@{
  ResourceId= "i-012e3cb4df567e8aa"
  ResourceName = "arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa"
  ResourceType = [Amazon.ConfigService.ResourceType]::AWSEC2Instance
  SourceAccountId = "123456789012"
  SourceRegion = "eu-west-1"
 }

 Get-CFGAggregateResourceConfigBatch -ResourceIdentifier $resIdentifier -
ConfigurationAggregatorName raju
```

Saída:

```
BaseConfigurationItems UnprocessedResourceIdentifiers
---------------------- -----------------------------
{}                     {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- Para obter detalhes sobre a API, consulte BatchGetAggregateResourceconfig-service Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGAggregationAuthorizationList`

O código de exemplo a seguir mostra como usar `Get-CFGAggregationAuthorizationList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera as autorizações concedidas aos agregadores.

```
Get-CFGAggregationAuthorizationList
```

Saída:

```
AggregationAuthorizationArn
   AuthorizedAccountId AuthorizedAwsRegion CreationTime
---------------------------
   ------------------- ------------------- ------------
arn:aws:config-service:eu-west-1:123456789012:aggregation-
authorization/123456789012/eu-west-1 123456789012        eu-west-1
 8/26/2019 12:55:27 AM
```

- Para obter detalhes sobre a API, consulte DescribeAggregationAuthorizations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGComplianceByConfigRule`

O código de exemplo a seguir mostra como usar `Get-CFGComplianceByConfigRule`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera detalhes de conformidade da regra ebs-optimized-instance, para a qual não há resultados de avaliação atuais para a regra e, portanto, retorna INSUFFICIENT_DATA

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Saída:

```
ComplianceContributorCount ComplianceType
------------------------- --------------
                           INSUFFICIENT_DATA
```

Exemplo 2: este exemplo retorna o número de recursos que não estão em conformidade com a regra ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Saída:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- Para obter detalhes sobre a API, consulte DescribeComplianceByConfigRule Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGComplianceByResource`

O código de exemplo a seguir mostra como usar `Get-CFGComplianceByResource`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo verifica o tipo de recurso **`AWS::SSM::ManagedInstanceInventory`** quanto ao tipo de conformidade “COMPLIANT”.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Saída:

```
Compliance                            ResourceId          ResourceType
----------                            ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- Para obter detalhes sobre a API, consulte DescribeComplianceByResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGComplianceDetailsByConfigRule`

O código de exemplo a seguir mostra como usar `Get-CFGComplianceDetailsByConfigRule`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo obtém os resultados da avaliação para a regra access-keys-rotated e retorna a saída agrupada por compliance-type

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Saída:

```
Count Name                      Group
----- ----                      -----
    2 COMPLIANT                 {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult}
    5 NON_COMPLIANT             {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationRes...
```

Exemplo 2: este exemplo consulta os detalhes de conformidade da regra access-keys-rotated para recursos COMPLIANT.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Saída:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Para obter detalhes sobre a API, consulte GetComplianceDetailsByConfigRule Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGComplianceDetailsByResource`

O código de exemplo a seguir mostra como usar `Get-CFGComplianceDetailsByResource`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo avalia resultados para um determinado recurso.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
 'AWS::IAM::User'
```

Saída:

```
Annotation                 :
ComplianceType             : COMPLIANT
ConfigRuleInvokedTime      : 8/25/2019 11:34:56 PM
EvaluationResultIdentifier : Amazon.ConfigService.Model.EvaluationResultIdentifier
ResultRecordedTime         : 8/25/2019 11:34:56 PM
ResultToken                :
```

- Para obter detalhes sobre a API, consulte GetComplianceDetailsByResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGComplianceSummaryByConfigRule`

O código de exemplo a seguir mostra como usar `Get-CFGComplianceSummaryByConfigRule`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o número de regras do Config que não estão em conformidade.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Saída:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- Para obter detalhes sobre a API, consulte GetComplianceSummaryByConfigRule Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGComplianceSummaryByResourceType`

O código de exemplo a seguir mostra como usar `Get-CFGComplianceSummaryByResourceType`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o número de recursos que estão em conformidade ou não e converte a saída em json.

```
Get-CFGComplianceSummaryByResourceType -Select
 ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- Para obter detalhes sobre a API, consulte GetComplianceSummaryByResourceType Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGConfigRule`

O código de exemplo a seguir mostra como usar `Get-CFGConfigRule`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo lista as regras de configuração da conta, com propriedades selecionadas.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Saída:

```
ConfigRuleName                                      ConfigRuleId       ConfigRuleArn
                                                        ConfigRuleState
```

```
--------------                                      ------------       -------------
                                                        ---------------
ALB_REDIRECTION_CHECK                               config-rule-12iyn3 arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-12iyn3 ACTIVE
access-keys-rotated                                 config-rule-aospfr arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-aospfr ACTIVE
autoscaling-group-elb-healthcheck-required          config-rule-cn1f2x arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x ACTIVE
```

- Para obter detalhes sobre a API, consulte DescribeConfigRules Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGConfigRuleEvaluationStatus`

O código de exemplo a seguir mostra como usar `Get-CFGConfigRuleEvaluationStatus`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna as informações de status das regras de configuração fornecidas.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Saída:

```
ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk
ConfigRuleId                 : config-rule-kvq1wk
ConfigRuleName               : root-account-mfa-enabled
FirstActivatedTime           : 8/27/2019 8:05:17 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime : 12/13/2019 8:12:03 AM

ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
ConfigRuleId                 : config-rule-z1s23b
```

```
ConfigRuleName               : vpc-flow-logs-enabled
FirstActivatedTime           : 8/14/2019 6:23:44 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime : 12/13/2019 7:12:01 AM
```

- Para obter detalhes sobre a API, consulte DescribeConfigRuleEvaluationStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGConfigurationAggregatorList`

O código de exemplo a seguir mostra como usar `Get-CFGConfigurationAggregatorList`.

Ferramentas para PowerShell V5

Exemplo 1: Essa amostra retorna todos os agregadores do region/account.

```
Get-CFGConfigurationAggregatorList
```

Saída:

```
AccountAggregationSources     :
 {Amazon.ConfigService.Model.AccountAggregationSource}
ConfigurationAggregatorArn    : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-xabca1me
ConfigurationAggregatorName   : IrelandMaster
CreationTime                  : 8/25/2019 11:42:39 PM
LastUpdatedTime               : 8/25/2019 11:42:39 PM
OrganizationAggregationSource :

AccountAggregationSources     : {}
ConfigurationAggregatorArn    : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-qubqabcd
ConfigurationAggregatorName   : raju
CreationTime                  : 8/11/2019 8:39:25 AM
LastUpdatedTime               : 8/11/2019 8:39:25 AM
OrganizationAggregationSource :
 Amazon.ConfigService.Model.OrganizationAggregationSource
```

- Para obter detalhes sobre a API, consulte DescribeConfigurationAggregators Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-CFGConfigurationAggregatorSourcesStatus

O código de exemplo a seguir mostra como usar `Get-CFGConfigurationAggregatorSourcesStatus`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra exibe os campos solicitados para as origens em determinado agregador.

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
 select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

Saída:

```
SourceType   LastUpdateStatus LastUpdateTime        SourceId
----------   ---------------- --------------        --------
ORGANIZATION SUCCEEDED        12/31/2019 7:45:06 AM Organization
ACCOUNT      SUCCEEDED        12/31/2019 7:09:38 AM 612641234567
ACCOUNT      SUCCEEDED        12/31/2019 7:12:53 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:18:10 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:25:17 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:25:49 AM 612641234567
ACCOUNT      SUCCEEDED        12/31/2019 7:26:11 AM 612641234567
```

- Para obter detalhes sobre a API, consulte DescribeConfigurationAggregatorSourcesStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-CFGConfigurationRecorder

O código de exemplo a seguir mostra como usar `Get-CFGConfigurationRecorder`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna os detalhes dos gravadores de configuração.

```
Get-CFGConfigurationRecorder | Format-List
```

Saída:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Para obter detalhes sobre a API, consulte DescribeConfigurationRecorders Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGConfigurationRecorderStatus`

O código de exemplo a seguir mostra como usar `Get-CFGConfigurationRecorderStatus`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o status dos gravadores de configuração.

```
Get-CFGConfigurationRecorderStatus
```

Saída:

```
LastErrorCode        :
LastErrorMessage     :
LastStartTime        : 10/11/2019 10:13:51 AM
LastStatus           : Success
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime         : 10/11/2019 10:13:46 AM
Name                 : default
Recording            : True
```

- Para obter detalhes sobre a API, consulte DescribeConfigurationRecorderStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGConformancePack`

O código de exemplo a seguir mostra como usar `Get-CFGConformancePack`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os pacotes de conformidade.

```
Get-CFGConformancePack
```

Saída:

```
ConformancePackArn             : arn:aws:config:eu-west-1:123456789012:conformance-
pack/dono/conformance-pack-p0acq8bpz
ConformancePackId              : conformance-pack-p0acabcde
ConformancePackInputParameters : {}
ConformancePackName            : dono
CreatedBy                      :
DeliveryS3Bucket               : kt-ps-examples
DeliveryS3KeyPrefix            :
LastUpdateRequestedTime        : 12/31/2019 8:45:31 AM
```

- Para obter detalhes sobre a API, consulte DescribeConformancePacks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGDeliveryChannel`

O código de exemplo a seguir mostra como usar `Get-CFGDeliveryChannel`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera o canal de entrega da região e exibe detalhes.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Saída:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- Para obter detalhes sobre a API, consulte DescribeDeliveryChannels Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-CFGResourceTag`

O código de exemplo a seguir mostra como usar `Get-CFGResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as tags associadas a determinado recurso

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

Saída:

```
Key     Value
---     -----
Version 1.3
```

- Para obter detalhes sobre a API, consulte ListTagsForResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-CFGConformancePack`

O código de exemplo a seguir mostra como usar `Remove-CFGConformancePack`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra remove o pacote de conformidade fornecido, junto com todas as regras, as ações de remediação e os resultados de avaliação do pacote.

```
Remove-CFGConformancePack -ConformancePackName dono
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-CFGConformancePack (DeleteConformancePack)" on
 target "dono".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteConformancePack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-CFGConformancePack`

O código de exemplo a seguir mostra como usar `Write-CFGConformancePack`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um pacote de conformidade, buscando o modelo do arquivo yaml fornecido.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket amzn-s3-demo-
bucket -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- Para obter detalhes sobre a API, consulte PutConformancePack Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-CFGDeliveryChannel`

O código de exemplo a seguir mostra como usar `Write-CFGDeliveryChannel`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo altera a propriedade deliveryFrequency de um canal de entrega existente.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName amzn-
s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- Para obter detalhes sobre a API, consulte PutDeliveryChannel Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de Device Farm usando Tools for PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Device Farm.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`New-DFUpload`

O código de exemplo a seguir mostra como usar `New-DFUpload`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo cria um upload AWS do Device Farm para um aplicativo Android. Você pode obter o ARN do projeto a partir da saída de New-DFProject ou Get-DFProjectList. Use o URL assinado na New-DFUpload saída para fazer upload de um arquivo para o Device Farm.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn
 "arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- Para obter detalhes sobre a API, consulte CreateUpload Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Directory Service exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com Directory Service.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-DSIpRoute`

O código de exemplo a seguir mostra como usar `Add-DSIpRoute`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando remove a etiqueta de recurso atribuída ao especificado Directory-id

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- Para obter detalhes sobre a API, consulte AddIpRoutes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-DSResourceTag`

O código de exemplo a seguir mostra como usar `Add-DSResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando adiciona a tag de recurso ao especificado Directory-id

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- Para obter detalhes sobre a API, consulte AddTagsToResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Approve-DSTrust`

O código de exemplo a seguir mostra como usar `Approve-DSTrust`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo chama a operação da VerifyTrust API AWS Directory Service para o Trustid especificado.

```
Approve-DSTrust -TrustId t-9067157123
```

- Para obter detalhes sobre a API, consulte VerifyTrust Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Confirm-DSSharedDirectory`

O código de exemplo a seguir mostra como usar `Confirm-DSSharedDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo aceita uma solicitação de compartilhamento de diretórios enviada pelo proprietário do diretório Conta da AWS.

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Saída:

```
CreatedDateTime     : 12/30/2019 4:20:27 AM
LastUpdatedDateTime : 12/30/2019 4:21:40 AM
OwnerAccountId      : 123456781234
OwnerDirectoryId    : d-123456ijkl
SharedAccountId     : 123456784321
SharedDirectoryId   : d-9067012345
ShareMethod         :
ShareNotes          : This is test sharing
ShareStatus         : Sharing
```

- Para obter detalhes sobre a API, consulte AcceptSharedDirectory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Connect-DSDirectory`

O código de exemplo a seguir mostra como usar `Connect-DSDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um AD Connector para conexão com um diretório on-premises.

```
Connect-DSDirectory -Name contoso.com -ConnectSettings_CustomerUserName
 Administrator -Password $Password -ConnectSettings_CustomerDnsIp 172.31.36.96
 -ShortName CONTOSO -Size Small -ConnectSettings_VpcId vpc-123459da -
ConnectSettings_SubnetId subnet-1234ccaa, subnet-5678ffbb
```

- Para obter detalhes sobre a API, consulte ConnectDirectory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Deny-DSSharedDirectory`

O código de exemplo a seguir mostra como usar `Deny-DSSharedDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo rejeita uma solicitação de compartilhamento de diretório que foi enviada da conta do proprietário do diretório.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Saída:

```
d-9067012345
```

- Para obter detalhes sobre a API, consulte RejectSharedDirectory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-DSDirectoryShare`

O código de exemplo a seguir mostra como usar `Disable-DSDirectoryShare`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo interrompe o compartilhamento de diretório entre o proprietário do diretório e a conta de consumidor.

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -
UnshareTarget_Type ACCOUNT
```

Saída:

```
d-9067012345
```

- Para obter detalhes sobre a API, consulte UnshareDirectory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-DSLDAPS`

O código de exemplo a seguir mostra como usar `Disable-DSLDAPS`.

Ferramentas para PowerShell V5

Exemplo 1:esse exemplo desativa as chamadas de segurança de LDAP para o diretório especificado.

```
Disable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Consulte detalhes da API em DisableLDAPS na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Disable-DSRadius`

O código de exemplo a seguir mostra como usar `Disable-DSRadius`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desativa o servidor RADIUS configurado para um diretório do AD Connector ou do Microsoft AD.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- Para obter detalhes sobre a API, consulte DisableRadius Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-DSSso`

O código de exemplo a seguir mostra como usar `Disable-DSSso`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desativa o login único para um diretório.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- Para obter detalhes sobre a API, consulte DisableSso Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-DSDirectoryShare`

O código de exemplo a seguir mostra como usar `Enable-DSDirectoryShare`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo compartilha um diretório especificado em sua AWS conta com outra AWS conta usando o método Handshake.

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

Saída:

```
d-9067012345
```

- Para obter detalhes sobre a API, consulte ShareDirectory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-DSLDAPS`

O código de exemplo a seguir mostra como usar `Enable-DSLDAPS`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo ativa a opção para o diretório especificado para sempre usar chamadas seguras LDAP.

```
Enable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Consulte detalhes da API em EnableLDAPS na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Enable-DSRadius`

O código de exemplo a seguir mostra como usar `Enable-DSRadius`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita a autenticação multifator (MFA) com a configuração do servidor RADIUS fornecida para um diretório do AD Connector ou do Microsoft AD.

```
Enable-DSRadius -DirectoryId d-123456ijkl
-RadiusSettings_AuthenticationProtocol PAP
-RadiusSettings_DisplayLabel Radius
-RadiusSettings_RadiusPort 1812
-RadiusSettings_RadiusRetry 4
-RadiusSettings_RadiusServer 10.4.185.113
-RadiusSettings_RadiusTimeout 50
-RadiusSettings_SharedSecret wJalrXUtnFEMI
```

- Para obter detalhes sobre a API, consulte EnableRadius Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-DSSso`

O código de exemplo a seguir mostra como usar `Enable-DSSso`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo ativa o login único para um diretório.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- Para obter detalhes sobre a API, consulte EnableSso Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSCertificate`

O código de exemplo a seguir mostra como usar `Get-DSCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe informações sobre o certificado registrado para uma conexão LDAP segura.

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

Saída:

```
CertificateId      : c-906731e34f
CommonName         : contoso-EC2AMAZ-CTGG2NM-CA
```

```
ExpiryDateTime     : 4/15/2025 6:34:15 PM
RegisteredDateTime : 4/15/2020 6:38:56 PM
State              : Registered
StateReason        : Certificate registered successfully.
```

- Para obter detalhes sobre a API, consulte DescribeCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSCertificateList`

O código de exemplo a seguir mostra como usar `Get-DSCertificateList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os certificados registrados para uma conexão LDAP segura para o diretório especificado.

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

Saída:

```
CertificateId CommonName                 ExpiryDateTime       State
------------- ----------                 --------------       -----
c-906731e34f  contoso-EC2AMAZ-CTGG2NM-CA 4/15/2025 6:34:15 PM Registered
```

- Para obter detalhes sobre a API, consulte ListCertificates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSConditionalForwarder`

O código de exemplo a seguir mostra como usar `Get-DSConditionalForwarder`.

Ferramentas para PowerShell V5

Exemplo 1: Este comando obtém todos os encaminhadores condicionais configurados de um dado. Directory-id

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

Saída:

```
DnsIpAddrs      RemoteDomainName ReplicationScope
----------      ---------------- ----------------
{172.31.77.239} contoso.com      Domain
```

- Para obter detalhes sobre a API, consulte DescribeConditionalForwarders Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSDirectory`

O código de exemplo a seguir mostra como usar `Get-DSDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém informações sobre os diretórios que pertencem a essa conta.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Saída:

```
DirectoryId  Name           DnsIpAddrs                     Type
-----------  ----           ----------                     ----
d-123456abcd abcd.example.com {172.31.74.189, 172.31.13.145} SimpleAD
d-123456efgh wifi.example.com {172.31.16.108, 172.31.10.56}  ADConnector
d-123456ijkl lan2.example.com {172.31.10.56, 172.31.16.108}  MicrosoftAD
```

- Para obter detalhes sobre a API, consulte DescribeDirectories Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSDirectoryLimit`

O código de exemplo a seguir mostra como usar `Get-DSDirectoryLimit`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra as informações de limite de diretórios da região us-east-1.

```
Get-DSDirectoryLimit -Region us-east-1
```

Saída:

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit        : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- Para obter detalhes sobre a API, consulte GetDirectoryLimits Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSDomainControllerList`

O código de exemplo a seguir mostra como usar `Get-DSDomainControllerList`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém a lista detalhada dos controladores de domínio lançados para o ID de diretório mencionado.

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

Saída:

```
AvailabilityZone          : us-east-1b
DirectoryId               : d-123456ijkl
DnsIpAddr                 : 172.31.16.108
DomainControllerId        : dc-1234567aa6
LaunchTime                : 4/4/2019 4:53:43 AM
Status                    : Active
StatusLastUpdatedDateTime : 4/24/2019 1:37:54 PM
StatusReason              :
SubnetId                  : subnet-1234kkaa
VpcId                     : vpc-123459d

AvailabilityZone          : us-east-1d
DirectoryId               : d-123456ijkl
DnsIpAddr                 : 172.31.10.56
DomainControllerId        : dc-1234567aa7
LaunchTime                : 4/4/2019 4:53:43 AM
Status                    : Active
```

```
StatusLastUpdatedDateTime : 4/4/2019 5:14:31 AM
StatusReason              :
SubnetId                  : subnet-5678ffbb
VpcId                     : vpc-123459d
```

- Para obter detalhes sobre a API, consulte DescribeDomainControllers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSEventTopic`

O código de exemplo a seguir mostra como usar `Get-DSEventTopic`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando mostra informações do tópico do SNS configurado para notificação enquanto o status do diretório é alterado.

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

Saída:

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status          : Registered
TopicArn        : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName       : snstopicname
```

- Para obter detalhes sobre a API, consulte DescribeEventTopics Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSIpRouteList`

O código de exemplo a seguir mostra como usar `Get-DSIpRouteList`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém os blocos de endereços IP públicos configurados no roteamento IP do diretório.

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

Saída:

```
AddedDateTime       : 12/13/2019 12:27:22 PM
CidrIp              : 203.0.113.5/32
Description         : Public IP of On-Prem DNS Server
DirectoryId         : d-123456ijkl
IpRouteStatusMsg    : Added
IpRouteStatusReason :
```

- Para obter detalhes sobre a API, consulte ListIpRoutes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSLDAPSSetting`

O código de exemplo a seguir mostra como usar `Get-DSLDAPSSetting`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o status da segurança LDAP do diretório especificado.

```
Get-DSLDAPSSetting -DirectoryId d-123456ijkl
```

Saída:

```
LastUpdatedDateTime  LDAPSStatus LDAPSStatusReason
-------------------  ----------- -----------------
4/15/2020 6:51:03 PM Enabled     LDAPS is enabled successfully.
```

- Consulte detalhes da API em DescribeLDAPSSettings na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Get-DSLogSubscriptionList`

O código de exemplo a seguir mostra como usar `Get-DSLogSubscriptionList`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém as informações de assinaturas de log do ID do diretório especificado

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

Saída:

```
DirectoryId  LogGroupName
 SubscriptionCreatedDateTime
-----------  ------------
 ---------------------------
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
 AM
```

- Para obter detalhes sobre a API, consulte ListLogSubscriptions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSResourceTag`

O código de exemplo a seguir mostra como usar `Get-DSResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém todas as tags do diretório especificado.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

Saída:

```
Key   Value
---   -----
myTag myTagValue
```

- Para obter detalhes sobre a API, consulte ListTagsForResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSSchemaExtension`

O código de exemplo a seguir mostra como usar `Get-DSSchemaExtension`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as extensões de esquema aplicadas a um Microsoft AD Directory.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

Saída:

```
Description                 : ManagedADSchemaExtension
DirectoryId                 : d-123456ijkl
EndDateTime                 : 4/12/2020 10:30:49 AM
SchemaExtensionId           : e-9067306643
SchemaExtensionStatus       : Completed
SchemaExtensionStatusReason : Schema updates are complete.
StartDateTime               : 4/12/2020 10:28:42 AM
```

- Para obter detalhes sobre a API, consulte ListSchemaExtensions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSSharedDirectory`

O código de exemplo a seguir mostra como usar `Get-DSSharedDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo obtém os diretórios compartilhados da sua conta AWS

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

Saída:

```
CreatedDateTime     : 12/30/2019 4:34:37 AM
LastUpdatedDateTime : 12/30/2019 4:35:22 AM
OwnerAccountId      : 123456781234
OwnerDirectoryId    : d-123456ijkl
SharedAccountId     : 123456784321
SharedDirectoryId   : d-9067012345
ShareMethod         : HANDSHAKE
ShareNotes          : This is a test Sharing
ShareStatus         : Shared
```

- Para obter detalhes sobre a API, consulte DescribeSharedDirectories Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSSnapshot`

O código de exemplo a seguir mostra como usar `Get-DSSnapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém informações sobre os snapshots do diretório especificado que pertencem a essa conta.

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

Saída:

```
DirectoryId : d-123456ijkl
Name        :
SnapshotId  : s-9064bd1234
StartTime   : 12/13/2019 6:33:01 PM
Status      : Completed
Type        : Auto

DirectoryId : d-123456ijkl
Name        :
SnapshotId  : s-9064bb4321
StartTime   : 12/9/2019 9:48:11 PM
Status      : Completed
Type        : Auto
```

- Para obter detalhes sobre a API, consulte DescribeSnapshots Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSSnapshotLimit`

O código de exemplo a seguir mostra como usar `Get-DSSnapshotLimit`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém os limites manuais de snapshot para um diretório especificado.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

Saída:

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
--------------------------- -------------------- --------------------------
0                           5                    False
```

- Para obter detalhes sobre a API, consulte GetSnapshotLimits Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DSTrust`

O código de exemplo a seguir mostra como usar `Get-DSTrust`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando obtém as informações das relações de confiança criadas para o ID de diretório especificado.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Saída:

```
CreatedDateTime          : 7/5/2019 4:55:42 AM
DirectoryId              : d-123456abcd
LastUpdatedDateTime      : 7/5/2019 4:56:04 AM
RemoteDomainName         : contoso.com
SelectiveAuth            : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
TrustDirection           : One-Way: Incoming
TrustId                  : t-9067157123
TrustState               : Created
TrustStateReason         :
TrustType                : Forest
```

- Para obter detalhes sobre a API, consulte DescribeTrusts Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DSAlias

O código de exemplo a seguir mostra como usar `New-DSAlias`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando cria um alias para um diretório e atribui esse alias ao directory-id especificado.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

Saída:

```
Alias     DirectoryId
-----     -----------
myorgname d-123456ijkl
```

- Para obter detalhes sobre a API, consulte CreateAlias Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DSComputer

O código de exemplo a seguir mostra como usar `New-DSComputer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um objeto de computador do Active Directory.

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password
 $Password
```

Saída:

```
ComputerAttributes                  ComputerId
 ComputerName
------------------                  ----------
 ------------
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
 ADMemberServer
```

- Para obter detalhes sobre a API, consulte CreateComputer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DSConditionalForwarder

O código de exemplo a seguir mostra como usar `New-DSConditionalForwarder`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um encaminhador condicional conforme especificado. AWS Directory-id

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- Para obter detalhes sobre a API, consulte CreateConditionalForwarder Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DSDirectory

O código de exemplo a seguir mostra como usar `New-DSDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um diretório do Simple AD.

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Para obter detalhes sobre a API, consulte CreateDirectory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DSLogSubscription

O código de exemplo a seguir mostra como usar `New-DSLogSubscription`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma assinatura para encaminhar registros de segurança do controlador de domínio do Directory Service em tempo real para o grupo de CloudWatch registros da Amazon especificado em seu Conta da AWS.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/
d-123456ijkl-lan2.example.com
```

- Para obter detalhes sobre a API, consulte CreateLogSubscription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DSMicrosoftAD

O código de exemplo a seguir mostra como usar `New-DSMicrosoftAD`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um novo Microsoft AD Directory em Nuvem AWS.

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- Para obter detalhes da API, consulte CreateMicrosoft AD in Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DSSnapshot

O código de exemplo a seguir mostra como usar `New-DSSnapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um snapshot do diretório

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- Para obter detalhes sobre a API, consulte CreateSnapshot Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DSTrust

O código de exemplo a seguir mostra como usar `New-DSTrust`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria Two-Way uma confiança em toda a floresta entre seu diretório AWS gerenciado do Microsoft AD e o Microsoft Active Directory local existente.

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection
 Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr
 172.31.36.96
```

Saída:

```
t-9067157123
```

- Para obter detalhes sobre a API, consulte CreateTrust Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Register-DSCertificate

O código de exemplo a seguir mostra como usar `Register-DSCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra um certificado para conexão LDAP segura.

```
$Certificate = Get-Content contoso.cer -Raw
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

Saída:

```
c-906731e350
```

- Para obter detalhes sobre a API, consulte RegisterCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Register-DSEventTopic

O código de exemplo a seguir mostra como usar `Register-DSEventTopic`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa um diretório como publicador a um tópico do SNS.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Para obter detalhes sobre a API, consulte RegisterEventTopic Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-DSConditionalForwarder`

O código de exemplo a seguir mostra como usar `Remove-DSConditionalForwarder`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove o encaminhador condicional que foi configurado para seu AWS diretório.

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName
 contoso.com
```

- Para obter detalhes sobre a API, consulte DeleteConditionalForwarder Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-DSDirectory`

O código de exemplo a seguir mostra como usar `Remove-DSDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui um AWS diretório de serviço de diretório (Simple AD/Microsoft AD/AD Connector)

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- Para obter detalhes sobre a API, consulte DeleteDirectory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-DSIpRoute`

O código de exemplo a seguir mostra como usar `Remove-DSIpRoute`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando remove o IP especificado das rotas IP configuradas do Directory-id.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- Para obter detalhes sobre a API, consulte RemoveIpRoutes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-DSLogSubscription`

O código de exemplo a seguir mostra como usar `Remove-DSLogSubscription`.

Ferramentas para PowerShell V5

Exemplo 1: Este comando remove a assinatura de log do especificado Directory-id

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- Para obter detalhes sobre a API, consulte DeleteLogSubscription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-DSResourceTag`

O código de exemplo a seguir mostra como usar `Remove-DSResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando remove a etiqueta de recurso atribuída ao especificado Directory-id

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- Para obter detalhes sobre a API, consulte RemoveTagsFromResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-DSSnapshot`

O código de exemplo a seguir mostra como usar `Remove-DSSnapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o snapshot criado manualmente.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- Para obter detalhes sobre a API, consulte DeleteSnapshot Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-DSTrust`

O código de exemplo a seguir mostra como usar `Remove-DSTrust`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove a relação de confiança existente entre seu Diretório AD AWS gerenciado e um domínio externo.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

Saída:

```
t-9067157123
```

- Para obter detalhes sobre a API, consulte DeleteTrust Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Reset-DSUserPassword`

O código de exemplo a seguir mostra como usar `Reset-DSUserPassword`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo redefine a senha do usuário do Active Directory chamado ADUser no AWS Managed microsoft AD ou no Simple AD Directory

```
Reset-DSUserPassword -UserName ADuser -DirectoryId d-123456ijkl -NewPassword
 $Password
```

- Para obter detalhes sobre a API, consulte ResetUserPassword Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Restore-DSFromSnapshot`

O código de exemplo a seguir mostra como usar `Restore-DSFromSnapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo restaura um diretório usando um snapshot de diretório existente.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- Para obter detalhes sobre a API, consulte RestoreFromSnapshot Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-DSDomainControllerCount`

O código de exemplo a seguir mostra como usar `Set-DSDomainControllerCount`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo define o número do controlador de domínio como 3 para o ID de diretório especificado.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- Para obter detalhes sobre a API, consulte UpdateNumberOfDomainControllers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-DSSchemaExtension`

O código de exemplo a seguir mostra como usar `Start-DSSchemaExtension`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo aplica uma extensão de esquema a um diretório do Microsoft AD.

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

Saída:

```
e-9067306643
```

- Para obter detalhes sobre a API, consulte StartSchemaExtension Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-DSSchemaExtension`

O código de exemplo a seguir mostra como usar `Stop-DSSchemaExtension`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela uma extensão de esquema em andamento para um diretório do Microsoft AD.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- Para obter detalhes sobre a API, consulte CancelSchemaExtension Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-DSCertificate`

O código de exemplo a seguir mostra como usar `Unregister-DSCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui do sistema o certificado que foi registrado para uma conexão LDAP segura.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- Para obter detalhes sobre a API, consulte DeregisterCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-DSEventTopic`

O código de exemplo a seguir mostra como usar `Unregister-DSEventTopic`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o diretório especificado como um publicador para o tópico do SNS especificado.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- Para obter detalhes sobre a API, consulte DeregisterEventTopic Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-DSConditionalForwarder`

O código de exemplo a seguir mostra como usar `Update-DSConditionalForwarder`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo atualiza um encaminhador condicional que foi configurado para seu AWS diretório.

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- Para obter detalhes sobre a API, consulte UpdateConditionalForwarder Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-DSRadius`

O código de exemplo a seguir mostra como usar `Update-DSRadius`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza as informações do servidor RADIUS para um diretório do AD Connector ou do Microsoft AD.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- Para obter detalhes sobre a API, consulte UpdateRadius Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-DSTrust`

O código de exemplo a seguir mostra como usar `Update-DSTrust`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo atualiza o SelectiveAuth parâmetro do trust-id especificado de Desativado para Ativado.

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

Saída:

```
RequestId                            TrustId
---------                            -------
138864a7-c9a8-4ad1-a828-eae479e85b45 t-9067157123
```

- Para obter detalhes sobre a API, consulte UpdateTrust Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS DMS exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS DMS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`New-DMSReplicationTask`

O código de exemplo a seguir mostra como usar `New-DMSReplicationTask`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma nova tarefa de replicação do AWS Database Migration Service que usa CdcStartTime em vez de CdcStartPosition. O MigrationType está definido como “full-load-and-cdc”, o que significa que a tabela de destino deve estar vazia. A nova tarefa é marcada com uma tag que tem uma chave de Estágio e um valor de chave de Teste. Para obter mais informações sobre os valores usados por esse cmdlet, consulte Criando uma tarefa (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) no Guia do usuário do AWS Database Migration Service.

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-
east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE"`
  -CdcStartTime "2019-08-08T12:12:12"`
  -CdcStopPosition "server_time:2019-08-09T12:12:12"`
  -MigrationType "full-load-and-cdc"`
  -ReplicationTaskIdentifier "task1"`
  -ReplicationTaskSetting ""`
  -SourceEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE"`
  -TableMapping "file:////home/testuser/table-mappings.json"`
  -Tag @{"Key"="Stage";"Value"="Test"}`
  -TargetEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- Para obter detalhes sobre a API, consulte CreateReplicationTask Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do DynamoDB usando ferramentas para V5 PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o DynamoDB.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-DDBIndexSchema`

O código de exemplo a seguir mostra como usar `Add-DDBIndexSchema`.

Ferramentas para PowerShell V5

Exemplo 1: cria um TableSchema objeto vazio e adiciona uma nova definição de índice secundário local a ele antes de gravar o TableSchema objeto no pipeline.

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

Saída:

```
AttributeSchema                              KeySchema
    LocalSecondaryIndexSchema
---------------                              ---------
    -------------------------
{LastPostDateTime}                           {}
    {LastPostIndex}
```

Exemplo 2: adiciona uma nova definição de índice secundário local ao TableSchema objeto fornecido antes de TableSchema gravá-lo novamente no pipeline. O TableSchema objeto também pode ser fornecido usando o parâmetro -Schema.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

Saída:

```
AttributeSchema                                KeySchema
    LocalSecondaryIndexSchema
---------------                                ---------
    -------------------------
{LastPostDateTime}                             {}
    {LastPostIndex}
```

- Para obter detalhes sobre a API, consulte Add-DDBIndexSchema Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Add-DDBKeySchema

O código de exemplo a seguir mostra como usar `Add-DDBKeySchema`.

Ferramentas para PowerShell V5

Exemplo 1: cria um TableSchema objeto vazio e adiciona entradas de definição de chave e atributo a ele usando os dados-chave especificados antes de gravar o TableSchema objeto no pipeline. O tipo de chave é declarado como 'HASH' por padrão; use o KeyType parâmetro - com um valor de 'RANGE' para declarar uma chave de intervalo.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Saída:

```
AttributeSchema                                KeySchema
    LocalSecondaryIndexSchema
---------------                                ---------
    -------------------------
{ForumName}                                    {ForumName}
    {}
```

Exemplo 2: adiciona novas entradas de definição de chave e atributo ao TableSchema objeto fornecido antes de gravar o TableSchema objeto no pipeline. O tipo de chave é declarado como 'HASH' por padrão; use o KeyType parâmetro - com um valor de 'RANGE' para declarar uma chave de intervalo. O TableSchema objeto também pode ser fornecido usando o parâmetro -Schema.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Saída:

```
AttributeSchema                              KeySchema
    LocalSecondaryIndexSchema
---------------                              ---------
    -------------------------
{ForumName}                                  {ForumName}
    {}
```

- Para obter detalhes sobre a API, consulte Add-DDBKeySchema Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

ConvertFrom-DDBItem

O código de exemplo a seguir mostra como usar `ConvertFrom-DDBItem`.

Ferramentas para PowerShell V5

Exemplo 1: ConvertFrom-DDBItem é usado para converter o resultado de uma tabela de hash Get-DDBItem do DynamoDB AttributeValues em uma tabela de hash de tipos comuns, como string e double.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
```

```
AlbumTitle                      Somewhat Famous
```

- Para obter detalhes sobre a API, consulte ConvertFrom-DDBItem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`ConvertTo-DDBItem`

O código de exemplo a seguir mostra como usar `ConvertTo-DDBItem`.

Ferramentas para PowerShell V5

Exemplo 1: um exemplo de conversão de uma tabela de hash em um dicionário de valores de atributos do DynamoDB.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Key       Value
---       -----
SongTitle Amazon.DynamoDBv2.Model.AttributeValue
Artist    Amazon.DynamoDBv2.Model.AttributeValue
```

Exemplo 2: um exemplo de conversão de uma tabela de hash em um dicionário de valores de atributos do DynamoDB.

```
@{
    MyMap        = @{
        MyString = 'my string'
    }
    MyStringSet  = [System.Collections.Generic.HashSet[String]]@('my', 'string')
    MyNumericSet = [System.Collections.Generic.HashSet[Int]]@(1, 2, 3)
    MyBinarySet  = [System.Collections.Generic.HashSet[System.IO.MemoryStream]]@(
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('my'))),
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('string')))
    )
    MyList1      = @('my', 'string')
    MyList2      = [System.Collections.Generic.List[Int]]@(1, 2)
    MyList3      = [System.Collections.ArrayList]@('one', 2, $true)
} | ConvertTo-DDBItem
```

Saída:

```
Key          Value
---          -----
MyStringSet  Amazon.DynamoDBv2.Model.AttributeValue
MyList1      Amazon.DynamoDBv2.Model.AttributeValue
MyNumericSet Amazon.DynamoDBv2.Model.AttributeValue
MyList2      Amazon.DynamoDBv2.Model.AttributeValue
MyBinarySet  Amazon.DynamoDBv2.Model.AttributeValue
MyMap        Amazon.DynamoDBv2.Model.AttributeValue
MyList3      Amazon.DynamoDBv2.Model.AttributeValue
```

- Para obter detalhes sobre a API, consulte ConvertTo-DDBItem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DDBBatchItem`

O código de exemplo a seguir mostra como usar `Get-DDBBatchItem`.

Ferramentas para PowerShell V5

Exemplo 1: Obtém o item com SongTitle “Somewhere Down The Road” das tabelas 'Music' e 'Songs' do DynamoDB.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
 'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
 Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
```

```
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-DDBItem
```

Saída:

```
Name                           Value
----                           -----
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
CriticRating                   10
Genre                          Country
Price                          1.94
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
CriticRating                   10
Genre                          Country
Price                          1.94
```

- Para obter detalhes sobre a API, consulte BatchGetItem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DDBItem`

O código de exemplo a seguir mostra como usar `Get-DDBItem`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o item do DynamoDB com a chave de partição SongTitle e a chave de classificação Artist.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Saída:

```
Name                           Value
----                           -----
```

```
Genre                          Country
SongTitle                      Somewhere Down The Road
Price                          1.94
Artist                         No One You Know
CriticRating                   9
AlbumTitle                     Somewhat Famous
```

- Para obter detalhes sobre a API, consulte GetItem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DDBTable`

O código de exemplo a seguir mostra como usar `Get-DDBTable`.

Ferramentas para PowerShell V5

Exemplo 1: exibe detalhes da tabela especificada.

```
Get-DDBTable -TableName "myTable"
```

- Para obter detalhes sobre a API, consulte DescribeTable Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-DDBTableList`

O código de exemplo a seguir mostra como usar `Get-DDBTableList`.

Ferramentas para PowerShell V5

Exemplo 1: exibe detalhes de todas as tabelas, iterando automaticamente até que o serviço indique que não existem mais tabelas.

```
Get-DDBTableList
```

- Para obter detalhes sobre a API, consulte ListTables Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Invoke-DDBQuery`

O código de exemplo a seguir mostra como usar `Invoke-DDBQuery`.

Ferramentas para PowerShell V5

Exemplo 1: invoca uma consulta que retorna itens do DynamoDB com o especificado e o Artista. SongTitle

```
$invokeDDBQuery = @{
    TableName = 'Music'
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Para ver detalhes da API, consulte Query na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Invoke-DDBScan`

O código de exemplo a seguir mostra como usar `Invoke-DDBScan`.

Ferramentas para PowerShell V5

Exemplo 1: exibe todos os itens da tabela Music.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Saída:

```
Name                           Value
```

```
----                            -----
Genre                           Country
Artist                          No One You Know
Price                           1.94
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
Genre                           Country
Artist                          No One You Know
Price                           1.98
CriticRating                    8.4
SongTitle                       My Dog Spot
AlbumTitle                      Hey Now
```

Exemplo 2: Retorna itens na tabela Música com um valor CriticRating maior ou igual a nove.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

Saída:

```
Name                            Value
----                            -----
Genre                           Country
Artist                          No One You Know
Price                           1.94
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
```

- Para ver detalhes da API, consulte Scan na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-DDBTable`

O código de exemplo a seguir mostra como usar `New-DDBTable`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo cria uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Assunto' (intervalo de tipos de chave). O esquema usado para construir a tabela pode ser canalizado para cada cmdlet conforme mostrado ou especificado usando o parâmetro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {}
```

Exemplo 2: Esse exemplo cria uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Assunto' (intervalo de tipos de chave). Um índice secundário local também é definido. A chave do índice secundário local será definida automaticamente a partir da chave de hash primária na tabela (ForumName). O esquema usado para construir a tabela pode ser canalizado para cada cmdlet conforme mostrado ou especificado usando o parâmetro -Schema.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
```

```
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

Exemplo 3: Esse exemplo mostra como usar um único pipeline para criar uma tabela chamada Thread que tem uma chave primária que consiste em 'ForumName' (hash do tipo de chave) e 'Assunto' (intervalo de tipos de chave) e um índice secundário local. Em Add-DDBKeySchema e Add-DDBIndexSchema crie um novo TableSchema objeto para você se um não for fornecido pelo pipeline ou pelo parâmetro -Schema.

```
New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
                     -RangeKeyName "LastPostDateTime" `
                     -RangeKeyDataType "S" `
                     -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Saída:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

- Para obter detalhes sobre a API, consulte CreateTable Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-DDBTableSchema

O código de exemplo a seguir mostra como usar `New-DDBTableSchema`.

Ferramentas para PowerShell V5

Exemplo 1: Cria um TableSchema objeto vazio pronto para aceitar definições de chave e índice para uso na criação de uma nova tabela do Amazon DynamoDB. O objeto retornado pode ser canalizado para os Add-DDBKeySchema New-DDBTable cmdlets Add-DDBIndexSchema e ou passado para eles usando o parâmetro -Schema em cada cmdlet.

```
New-DDBTableSchema
```

Saída:

```
AttributeSchema                              KeySchema
    LocalSecondaryIndexSchema
---------------                              ---------
    -------------------------
{}                                           {}
    {}
```

- Para obter detalhes sobre a API, consulte New-DDBTableSchema Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Remove-DDBItem

O código de exemplo a seguir mostra como usar `Remove-DDBItem`.

Ferramentas para PowerShell V5

Exemplo 1: remove o item do DynamoDB que corresponde à chave fornecida.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Para obter detalhes sobre a API, consulte DeleteItem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-DDBTable`

O código de exemplo a seguir mostra como usar `Remove-DDBTable`.

Ferramentas para PowerShell V5

Exemplo 1: exclui a tabela especificada. A confirmação será solicitada antes que a operação continue.

```
Remove-DDBTable -TableName "myTable"
```

Exemplo 2: exclui a tabela especificada. A confirmação não será solicitada antes que a operação continue.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Para obter detalhes sobre a API, consulte DeleteTable Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-DDBBatchItem`

O código de exemplo a seguir mostra como usar `Set-DDBBatchItem`.

Ferramentas para PowerShell V5

Exemplo 1: cria um item ou substitui um item por um novo item nas tabelas Music e Songs do DynamoDB.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
    Genre = 'Country'
    CriticRating = 10.0
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item
```

```
$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Para obter detalhes sobre a API, consulte BatchWriteItem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-DDBItem`

O código de exemplo a seguir mostra como usar `Set-DDBItem`.

Ferramentas para PowerShell V5

Exemplo 1: cria um item ou substitui um item por um novo item.

```
$item = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
        AlbumTitle = 'Somewhat Famous'
        Price = 1.94
        Genre = 'Country'
        CriticRating = 9.0
} | ConvertTo-DDBItem
Set-DDBItem -TableName 'Music' -Item $item
```

- Para obter detalhes sobre a API, consulte PutItem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-DDBItem`

O código de exemplo a seguir mostra como usar `Update-DDBItem`.

Ferramentas para PowerShell V5

Exemplo 1: define o atributo de gênero como 'Rap' no item do DynamoDB com a chave de partição SongTitle e a chave de classificação Artist.

```
$key = @{
```

```
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$updateDdbItem = @{
    TableName = 'Music'
    Key = $key
    UpdateExpression = 'set Genre = :val1'
    ExpressionAttributeValue = (@{
        ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
    })
}
Update-DDBItem @updateDdbItem
```

Saída:

```
Name                           Value
----                           -----
Genre                          Rap
```

- Para obter detalhes sobre a API, consulte UpdateItem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-DDBTable`

O código de exemplo a seguir mostra como usar `Update-DDBTable`.

Ferramentas para PowerShell V5

Exemplo 1: atualiza os valores de throughput provisionado da tabela especificada.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Para obter detalhes sobre a API, consulte UpdateTable Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon EC2 usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon EC2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-EC2CapacityReservation`

O código de exemplo a seguir mostra como usar `Add-EC2CapacityReservation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma reserva de capacidade com os atributos especificados.

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

Saída:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
State                  : active
Tags                   : {}
Tenancy                : default
TotalInstanceCount     : 2
```

- Para obter detalhes sobre a API, consulte CreateCapacityReservation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-EC2InternetGateway`

O código de exemplo a seguir mostra como usar `Add-EC2InternetGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo anexa o gateway da internet especificado à VPC especificada.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Exemplo 2: esse exemplo cria uma VPC e um gateway da internet e, depois, conecta o gateway da internet à VPC.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- Para obter detalhes sobre a API, consulte AttachInternetGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-EC2NetworkInterface`

O código de exemplo a seguir mostra como usar `Add-EC2NetworkInterface`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo anexa a interface de rede especificada à instância especificada.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

Saída:

```
eni-attach-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte AttachNetworkInterface Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-EC2Volume`

O código de exemplo a seguir mostra como usar `Add-EC2Volume`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo anexa o volume especificado à instância especificada e o expõe com o nome do dispositivo especificado.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Saída:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- Para obter detalhes sobre a API, consulte AttachVolume Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-EC2VpnGateway`

O código de exemplo a seguir mostra como usar `Add-EC2VpnGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo anexa o gateway privado virtual especificado à VPC especificada.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Saída:

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- Para obter detalhes sobre a API, consulte AttachVpnGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Approve-EC2VpcPeeringConnection`

O código de exemplo a seguir mostra como usar `Approve-EC2VpcPeeringConnection`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo aprova o pcx-1dfad234b56ff78be solicitado VpcPeeringConnectionId

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

Saída:

```
AccepterVpcInfo        : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime         : 1/1/0001 12:00:00 AM
RequesterVpcInfo       : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status                 : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                   : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- Para obter detalhes sobre a API, consulte AcceptVpcPeeringConnection Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Confirm-EC2ProductInstance`

O código de exemplo a seguir mostra como usar `Confirm-EC2ProductInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo determina se o código do produto especificado está associado à instância especificada.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- Para obter detalhes sobre a API, consulte ConfirmProductInstance Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Copy-EC2Image`

O código de exemplo a seguir mostra como usar `Copy-EC2Image`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo copia a AMI especificada na região “UE (Irlanda)” para a região “Oeste dos EUA (Oregon)”. Se -Region não for especificada, a região padrão atual será usada como a região de destino.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-west-2
 -Name "Copy of ami-12345678"
```

Saída:

```
ami-87654321
```

- Para obter detalhes sobre a API, consulte CopyImage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Copy-EC2Snapshot`

O código de exemplo a seguir mostra como usar `Copy-EC2Snapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo copia a AMI especificada na região “UE (Irlanda)” para a região “Oeste dos EUA (Oregon)”.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region us-
west-2
```

Exemplo 2: se você definir uma região padrão e omitir o parâmetro Região, a região de destino padrão será a região padrão.

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- Para obter detalhes sobre a API, consulte CopySnapshot Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Deny-EC2VpcPeeringConnection`

O código de exemplo a seguir mostra como usar `Deny-EC2VpcPeeringConnection`.

Ferramentas para PowerShell V5

Exemplo 1: O exemplo acima nega a solicitação de ID de solicitação VpcPeering pcx-01a2b3ce45fe67eb8

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Para obter detalhes sobre a API, consulte RejectVpcPeeringConnection Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-EC2VgwRoutePropagation`

O código de exemplo a seguir mostra como usar `Disable-EC2VgwRoutePropagation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo impede que o VGW propague automaticamente as rotas para a tabela de rotas especificada.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte DisableVgwRoutePropagation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-EC2VpcClassicLink`

O código de exemplo a seguir mostra como usar `Disable-EC2VpcClassicLink`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo desativa o EC2VpcClassicLink vpc-01e23c4a5d6db78e9. Ele retorna Verdadeiro ou Falso

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Para obter detalhes sobre a API, consulte DisableVpcClassicLink Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-EC2VpcClassicLinkDnsSupport`

O código de exemplo a seguir mostra como usar `Disable-EC2VpcClassicLinkDnsSupport`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo desativa o suporte de ClassicLink DNS para o vpc-0b12d3456a7e8910d

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Para obter detalhes sobre a API, consulte DisableVpcClassicLinkDnsSupport Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Dismount-EC2InternetGateway`

O código de exemplo a seguir mostra como usar `Dismount-EC2InternetGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desanexa o gateway da internet especificado da VPC especificada.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- Para obter detalhes sobre a API, consulte DetachInternetGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Dismount-EC2NetworkInterface`

O código de exemplo a seguir mostra como usar `Dismount-EC2NetworkInterface`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o anexo especificado entre uma interface de rede e uma instância.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Para obter detalhes sobre a API, consulte DetachNetworkInterface Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Dismount-EC2Volume`

O código de exemplo a seguir mostra como usar `Dismount-EC2Volume`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desvincula o volume especificado.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Saída:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

Exemplo 2: você também pode especificar o ID da instância e o nome do dispositivo para garantir que esteja desanexando o volume correto.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- Para obter detalhes sobre a API, consulte DetachVolume Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Dismount-EC2VpnGateway`

O código de exemplo a seguir mostra como usar `Dismount-EC2VpnGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desvincula o gateway privado virtual especificado da VPC especificada.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- Para obter detalhes sobre a API, consulte DetachVpnGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2CapacityReservation`

O código de exemplo a seguir mostra como usar `Edit-EC2CapacityReservation`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo modifica o CapacityReservationId cr-0c1f2345db6f7cdba alterando a contagem de instâncias para 1

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Saída:

```
True
```

- Para obter detalhes sobre a API, consulte ModifyCapacityReservation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2Host`

O código de exemplo a seguir mostra como usar `Edit-EC2Host`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo modifica as AutoPlacement configurações para desativadas para o host dedicado h-01e23f4cd567890f3

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Saída:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- Para obter detalhes sobre a API, consulte ModifyHosts Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2IdFormat`

O código de exemplo a seguir mostra como usar `Edit-EC2IdFormat`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o formato de ID mais longo para o tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Exemplo 2: esse exemplo desativa o formato de ID mais longo para o tipo de recurso especificado.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Para obter detalhes sobre a API, consulte ModifyIdFormat Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2ImageAttribute`

O código de exemplo a seguir mostra como usar `Edit-EC2ImageAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza a descrição da AMI especificada.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Exemplo 2: Esse exemplo torna a AMI pública (por exemplo, para que qualquer Conta da AWS pessoa possa usá-la).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

Exemplo 3: esse exemplo torna a AMI privada (por exemplo, para que somente você, como proprietário, possa usá-la).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

Exemplo 4: Este exemplo concede permissão de lançamento ao especificado Conta da AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

Exemplo 5: Este exemplo remove a permissão de lançamento do especificado Conta da AWS.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- Para obter detalhes sobre a API, consulte ModifyImageAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2InstanceAttribute`

O código de exemplo a seguir mostra como usar `Edit-EC2InstanceAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica o tipo de instância da instância especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Exemplo 2: Este exemplo habilita redes aprimoradas para a instância especificada, especificando “simple” como o valor do parâmetro de suporte de rede de I/O virtualização raiz única (SR-IOV), -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Exemplo 3: esse exemplo modifica os grupos de segurança da instância especificada. A instância deve estar em uma VPC. Você deve especificar o ID de cada grupo de segurança, não o nome.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

Exemplo 4: Esse exemplo permite a I/O otimização do EBS para a instância especificada. Esse recurso não está disponível com todos os tipos de instâncias. Cobranças adicionais de uso são aplicadas ao usar uma EBS-optimized instância.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Exemplo 5: Esse exemplo permite a source/destination verificação da instância especificada. Para que a instância execute a NAT, o valor deve ser “false”.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Exemplo 6: esse exemplo desabilita o encerramento da instância especificada.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Exemplo 7: esse exemplo altera a instância especificada para que ela seja encerrada quando o desligamento for iniciado por meio da instância.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior
 terminate
```

- Para obter detalhes sobre a API, consulte ModifyInstanceAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2InstanceCreditSpecification`

O código de exemplo a seguir mostra como usar `Edit-EC2InstanceCreditSpecification`.

Ferramentas para PowerShell V5

Exemplo 1: isso habilita créditos ilimitados de T2, por exemplo, i-01234567890abcdef.

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Para obter detalhes sobre a API, consulte ModifyInstanceCreditSpecification Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2NetworkInterfaceAttribute`

O código de exemplo a seguir mostra como usar `Edit-EC2NetworkInterfaceAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica a interface de rede especificada para que o anexo especificado seja excluído no encerramento.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Exemplo 2: esse exemplo modifica a descrição da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
 description"
```

Exemplo 3: esse exemplo modifica o grupo de segurança da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

Exemplo 4: Esse exemplo desativa a source/destination verificação da interface de rede especificada.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
 $false
```

- Para obter detalhes sobre a API, consulte ModifyNetworkInterfaceAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2ReservedInstance`

O código de exemplo a seguir mostra como usar `Edit-EC2ReservedInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica a zona de disponibilidade, a contagem de instâncias e a plataforma das instâncias reservadas especificadas.

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"
```

```
Edit-EC2ReservedInstance `
-ReservedInstancesId @("FE32132D-70D5-4795-B400-AE435EXAMPLE", "0CC556F3-7AB8-4C00-
B0E5-98666EXAMPLE") `
-TargetConfiguration $config
```

- Para obter detalhes sobre a API, consulte ModifyReservedInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2SnapshotAttribute`

O código de exemplo a seguir mostra como usar `Edit-EC2SnapshotAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo torna público o instantâneo especificado definindo seu CreateVolumePermission atributo.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- Para obter detalhes sobre a API, consulte ModifySnapshotAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2SpotFleetRequest`

O código de exemplo a seguir mostra como usar `Edit-EC2SpotFleetRequest`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza a capacidade de destino da solicitação de frota spot especificada.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Saída:

```
True
```

- Para obter detalhes sobre a API, consulte ModifySpotFleetRequest Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2SubnetAttribute`

O código de exemplo a seguir mostra como usar `Edit-EC2SubnetAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o endereçamento IP público para a sub-rede especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Exemplo 2: esse exemplo desabilita o endereçamento IP público para a sub-rede especificada.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Para obter detalhes sobre a API, consulte ModifySubnetAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2VolumeAttribute`

O código de exemplo a seguir mostra como usar `Edit-EC2VolumeAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo modifica o atributo especificado do volume especificado. I/O as operações do volume são retomadas automaticamente após serem suspensas devido a dados potencialmente inconsistentes.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Para obter detalhes sobre a API, consulte ModifyVolumeAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-EC2VpcAttribute`

O código de exemplo a seguir mostra como usar `Edit-EC2VpcAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o suporte a nomes de host DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Exemplo 2: esse exemplo desabilita o suporte a nomes de host DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Exemplo 3: esse exemplo habilita o suporte à resolução DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Exemplo 4: esse exemplo desabilita o suporte à resolução DNS para a VPC especificada.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Para obter detalhes sobre a API, consulte ModifyVpcAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-EC2VgwRoutePropagation`

O código de exemplo a seguir mostra como usar `Enable-EC2VgwRoutePropagation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o VGW especificado a propagar automaticamente as rotas para a tabela de rotas especificada.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte EnableVgwRoutePropagation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-EC2VolumeIO`

O código de exemplo a seguir mostra como usar `Enable-EC2VolumeIO`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo habilita I/O operações para o volume especificado, se I/O as operações estiverem desativadas.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Para obter detalhes sobre a API, consulte EnableVolumeIo Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-EC2VpcClassicLink`

O código de exemplo a seguir mostra como usar `Enable-EC2VpcClassicLink`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo habilita a VPC vpc-0123456b789b0d12f para ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Saída:

```
True
```

- Para obter detalhes sobre a API, consulte EnableVpcClassicLink Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-EC2VpcClassicLinkDnsSupport`

O código de exemplo a seguir mostra como usar `Enable-EC2VpcClassicLinkDnsSupport`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo permite que vpc-0b12d3456a7e8910d ofereça suporte à resolução de nomes de host DNS para ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Para obter detalhes sobre a API, consulte EnableVpcClassicLinkDnsSupport Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2AccountAttribute`

O código de exemplo a seguir mostra como usar `Get-EC2AccountAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve se você pode iniciar instâncias EC2-VPC na EC2-Classic e na região, ou somente em EC2-VPC.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Saída:

```
AttributeValue
--------------
EC2
VPC
```

Exemplo 2: esse exemplo descreve a VPC padrão ou indica “nenhuma” caso você não tenha uma VPC padrão na região.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Saída:

```
AttributeValue
--------------
vpc-12345678
```

Exemplo 3: Esse exemplo descreve o número máximo de On-Demand instâncias que você pode executar.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Saída:

```
AttributeValue
--------------
20
```

- Para obter detalhes sobre a API, consulte DescribeAccountAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Address`

O código de exemplo a seguir mostra como usar `Get-EC2Address`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Saída:

```
AllocationId            : eipalloc-12345678
AssociationId           : eipassoc-12345678
Domain                  : vpc
InstanceId              : i-87654321
NetworkInterfaceId      : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress        : 10.0.2.172
PublicIp                : 198.51.100.2
```

Exemplo 2: esse exemplo descreve os endereços IP elásticos para instâncias em uma VPC. Essa sintaxe requer a PowerShell versão 3 ou posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Exemplo 3: Este exemplo descreve o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Saída:

```
AllocationId            :
```

```
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

Exemplo 4: Este exemplo descreve seus endereços IP elásticos para instâncias em EC2-Classic. Essa sintaxe requer a PowerShell versão 3 ou posterior.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Exemplo 5: esse exemplo descreve todos os endereços IP elásticos.

```
Get-EC2Address
```

Exemplo 6: esse exemplo retorna o IP público e privado para o ID da instância fornecido no filtro

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Saída:

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

Exemplo 7: esse exemplo recupera todos os IPs elásticos com seu ID de alocação, ID de associação e IDs de instância

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Saída:

```
InstanceId          AssociationId              AllocationId               PublicIp
----------          -------------              ------------               --------
```

```
                                                    eipalloc-012e3b456789e1fad
 17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
 17.212.124.77
                                                    eipalloc-012345678eeabcfad
 17.212.225.7
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
 37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
 37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
 37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
 37.215.222.270
```

Exemplo 8: esse exemplo busca uma lista de endereços IP do EC2 que correspondem à chave de tag “Category” com o valor “Prod”

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Saída:

```
AllocationId            : eipalloc-0123f456f81a01b58
AssociationId           : eipassoc-0d1b23a456d103810
CustomerOwnedIp         :
CustomerOwnedIpv4Pool   :
Domain                  : vpc
InstanceId              : i-012e3cb4df567e1aa
NetworkBorderGroup      : eu-west-1
NetworkInterfaceId      : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress        : 192.168.1.84
PublicIp                : 34.250.81.29
PublicIpv4Pool          : amazon
Tags                    : {Category, Name}
```

- Para obter detalhes sobre a API, consulte DescribeAddresses Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2AvailabilityZone`

O código de exemplo a seguir mostra como usar `Get-EC2AvailabilityZone`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as zonas de disponibilidade da região atual que estão disponíveis para você.

```
Get-EC2AvailabilityZone
```

Saída:

```
Messages    RegionName    State        ZoneName
--------    ----------    -----        --------
{}          us-west-2     available    us-west-2a
{}          us-west-2     available    us-west-2b
{}          us-west-2     available    us-west-2c
```

Exemplo 2: esse exemplo descreve todas as zonas de disponibilidade que estão em estado de comprometimento. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- Para obter detalhes sobre a API, consulte DescribeAvailabilityZones Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2BundleTask`

O código de exemplo a seguir mostra como usar `Get-EC2BundleTask`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a tarefa de empacotamento especificada.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Exemplo 2: esse exemplo descreve as tarefas do pacote cujo estado é “concluído” ou “com falha”.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- Para obter detalhes sobre a API, consulte DescribeBundleTasks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2CapacityReservation`

O código de exemplo a seguir mostra como usar `Get-EC2CapacityReservation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descrever uma ou mais de suas reservas de capacidade para a região

```
Get-EC2CapacityReservation -Region eu-west-1
```

Saída:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
State                  : active
Tags                   : {}
Tenancy                : default
TotalInstanceCount     : 2
```

- Para obter detalhes sobre a API, consulte DescribeCapacityReservations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2ConsoleOutput`

O código de exemplo a seguir mostra como usar `Get-EC2ConsoleOutput`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém a saída do console para a instância Linux especificada. A saída do console é codificada.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Saída:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Exemplo 2: esse exemplo armazena a saída codificada do console em uma variável e, depois, a decodifica.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- Para obter detalhes sobre a API, consulte GetConsoleOutput Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2CustomerGateway`

O código de exemplo a seguir mostra como usar `Get-EC2CustomerGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o gateway do cliente especificado.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Saída:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
```

```
State             : available
Tags              : {}
Type              : ipsec.1
```

Exemplo 2: esse exemplo descreve qualquer gateway do cliente cujo estado seja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Exemplo 3: esse exemplo descreve todos os gateways do cliente.

```
Get-EC2CustomerGateway
```

- Para obter detalhes sobre a API, consulte DescribeCustomerGateways Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2DhcpOption`

O código de exemplo a seguir mostra como usar `Get-EC2DhcpOption`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista seus conjuntos de opções DHCP.

```
Get-EC2DhcpOption
```

Saída:

```
DhcpConfigurations                  DhcpOptionsId    Tag
------------------                  -------------    ---
{domain-name, domain-name-servers}  dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}  dopt-2a3b4c5d    {}
{domain-name-servers}               dopt-3a4b5c6d    {}
```

Exemplo 2: esse exemplo obtém detalhes de configuração para o conjunto de opções DHCP especificado.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Saída:

```
Key                    Values
---                    ------
domain-name            {abc.local}
domain-name-servers    {10.0.0.101, 10.0.0.102}
```

- Para obter detalhes sobre a API, consulte DescribeDhcpOptions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2FlowLog`

O código de exemplo a seguir mostra como usar `Get-EC2FlowLog`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve um ou mais logs de fluxo com o tipo de destino de log “s3”

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

Saída:

```
CreationTime             : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage  :
DeliverLogsPermissionArn :
DeliverLogsStatus        : SUCCESS
FlowLogId                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType       : s3
LogGroupName             :
ResourceId               : eni-01d2dda3456b7e890
TrafficType              : ALL
```

- Para obter detalhes sobre a API, consulte DescribeFlowLogs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-EC2Host

O código de exemplo a seguir mostra como usar `Get-EC2Host`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna os detalhes do host do EC2.

```
Get-EC2Host
```

Saída:

```
AllocationTime    : 3/23/2019 4:55:22 PM
AutoPlacement     : off
AvailabilityZone  : eu-west-1b
AvailableCapacity : Amazon.EC2.Model.AvailableCapacity
ClientToken       :
HostId            : h-01e23f4cd567890f1
HostProperties    : Amazon.EC2.Model.HostProperties
HostReservationId :
Instances         : {}
ReleaseTime       : 1/1/0001 12:00:00 AM
State             : available
Tags              : {}
```

Exemplo 2: Este exemplo consulta o AvailableInstanceCapacity para o host h-01e23f4cd567899f1

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Saída:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- Para obter detalhes sobre a API, consulte DescribeHosts Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2HostReservationOffering`

O código de exemplo a seguir mostra como usar `Get-EC2HostReservationOffering`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve as reservas de host dedicado que estão disponíveis para compra para o filtro específico 'família de instâncias', onde está PaymentOption " NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

Saída:

```
CurrencyCode   :
Duration       : 94608000
HourlyPrice    : 1.307
InstanceFamily : m4
OfferingId     : hro-0c1f234567890d9ab
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000

CurrencyCode   :
Duration       : 31536000
HourlyPrice    : 1.830
InstanceFamily : m4
OfferingId     : hro-04ad12aaaf34b5a67
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000
```

- Para obter detalhes sobre a API, consulte DescribeHostReservationOfferings Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2HostReservationPurchasePreview`

O código de exemplo a seguir mostra como usar `Get-EC2HostReservationPurchasePreview`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo permite visualizar uma compra de reserva com configurações que correspondem às do seu host dedicado h-01e23f4cd567890f1

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

Saída:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- Para obter detalhes sobre a API, consulte GetHostReservationPurchasePreview Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2IdFormat`

O código de exemplo a seguir mostra como usar `Get-EC2IdFormat`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o formato de ID para o tipo de recurso especificado.

```
Get-EC2IdFormat -Resource instance
```

Saída:

```
Resource       UseLongIds
--------       ----------
instance       False
```

Exemplo 2: esse exemplo descreve os formatos de ID para todos os tipos de recursos que oferecem suporte a IDs mais longos.

```
Get-EC2IdFormat
```

Saída:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- Para obter detalhes sobre a API, consulte DescribeIdFormat Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2IdentityIdFormat`

O código de exemplo a seguir mostra como usar `Get-EC2IdentityIdFormat`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o formato de ID do recurso “image” para a função especificada

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
 image
```

Saída:

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- Para obter detalhes sobre a API, consulte DescribeIdentityIdFormat Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Image`

O código de exemplo a seguir mostra como usar `Get-EC2Image`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a AMI especificada.

```
Get-EC2Image -ImageId ami-12345678
```

Saída:

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/xvda}
CreationDate        : 2014-10-20T00:56:28.000Z
Description         : My image
Hypervisor          : xen
```

```
ImageId             : ami-12345678
ImageLocation       : 123456789012/my-image
ImageOwnerAlias     :
ImageType           : machine
KernelId            :
Name                : my-image
OwnerId             : 123456789012
Platform            :
ProductCodes        : {}
Public              : False
RamdiskId           :
RootDeviceName      : /dev/xvda
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {Name}
VirtualizationType  : hvm
```

Exemplo 2: esse exemplo descreve as AMIs que você possui.

```
Get-EC2Image -owner self
```

Exemplo 3: esse exemplo descreve as AMIs públicas que executam o Microsoft Windows Server.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Exemplo 4: esse exemplo descreve todas as AMIs públicas na região “us-west-2”.

```
Get-EC2Image -Region us-west-2
```

- Para obter detalhes sobre a API, consulte DescribeImages Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2ImageAttribute`

O código de exemplo a seguir mostra como usar `Get-EC2ImageAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém a descrição da AMI especificada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Saída:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Exemplo 2: esse exemplo obtém as permissões de lançamento da AMI especificada.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Saída:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Exemplo 3: esse exemplo testa se as redes aprimoradas estão habilitadas.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Saída:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
```

```
RamdiskId           :
SriovNetSupport     : simple
```

- Para obter detalhes sobre a API, consulte DescribeImageAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2ImportImageTask`

O código de exemplo a seguir mostra como usar `Get-EC2ImportImageTask`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a tarefa de importação de imagem especificada.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Saída:

```
Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

Exemplo 2: esse exemplo descreve todas as tarefas de importação de imagens.

```
Get-EC2ImportImageTask
```

Saída:

```
Architecture    :
Description     : Windows Image 1
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
```

```
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {}
Status          : deleted
StatusMessage   : User initiated task cancelation

Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

- Para obter detalhes sobre a API, consulte DescribeImportImageTasks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2ImportSnapshotTask`

O código de exemplo a seguir mostra como usar `Get-EC2ImportSnapshotTask`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a tarefa de importação de snapshot especificada.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Saída:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

Exemplo 2: esse exemplo descreve todas as tarefas de importação de snapshots.

```
Get-EC2ImportSnapshotTask
```

Saída:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2     import-snap-hgfedcba
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Para obter detalhes sobre a API, consulte DescribeImportSnapshotTasks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Instance`

O código de exemplo a seguir mostra como usar `Get-EC2Instance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a instância especificada.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Saída:

```
AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           : TleEy1448154045270
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    : Amazon.EC2.Model.IamInstanceProfile
ImageId               : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle     :
InstanceType          : t2.micro
KernelId              :
```

```
KeyName               : my-key-pair
LaunchTime            : 12/4/2015 4:44:40 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {ip-10-0-2-172.us-west-2.compute.internal}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress      : 10.0.2.172
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       :
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {default}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-12345678
Tags                  : {Name}
VirtualizationType    : hvm
VpcId                 : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as instâncias na região atual, agrupadas por reserva. Para ver os detalhes da instância, expanda a coleção de instâncias em cada objeto de reserva.

```
Get-EC2Instance
```

Saída:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
Instances     : {}
```

```
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Exemplo 3: esse exemplo ilustra o uso de um filtro para consultar instâncias do EC2 em uma sub-rede específica de uma VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-01af...82cf180e19 t2.medium    Windows  10.0.0.98                       ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0374...7e9d5b0c45 t2.xlarge    Windows  10.0.0.53                       ...
      subnet-1a2b3c4d vpc-1a2b3c4d
```

Exemplo 4: esse exemplo ilustra o uso de um filtro com vários valores para consultar instâncias do EC2 que estão em execução e que foram interrompidas

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
```

```
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10                       ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11                       ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
```

Exemplo 5: Este exemplo ilustra o uso de um filtro com vários valores para consultar instâncias do EC2 que estão em execução e interrompidas e o uso do Select-Object cmdlet para escolher valores específicos para saída.

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

$SelectParams = @{
    Property = @(
        "InstanceID", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_.Tags.Key.IndexOf("Name")].Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

Saída:

```
InstanceId          InstanceType Platform PrivateIpAddress Name         State
----------          ------------ -------- ---------------- ----         -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7         ec2-name-01  running
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8         ec2-name-02  stopped
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10        ec2-name-03  running
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11        ec2-name-04  stopped
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5         ec2-name-05  running
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6         ec2-name-06  stopped
```

Exemplo 6: Esse exemplo valida as permissões para obter instâncias do EC2 usando o DryRun parâmetro sem realmente buscá-las. Nota: se isso for feito com sucesso, será gerada uma exceção, que é o comportamento esperado.

```
Get-EC2Tag -DryRun $true
```

Saída:

```
Get-EC2Instance: Request would have succeeded, but DryRun flag is set.
```

- Para obter detalhes sobre a API, consulte DescribeInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2InstanceAttribute`

O código de exemplo a seguir mostra como usar `Get-EC2InstanceAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o tipo de instância da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Saída:

```
InstanceType                     : t2.micro
```

Exemplo 2: esse exemplo descreve se a rede avançada está habilitada para a instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Saída:

```
SriovNetSupport                  : simple
```

Exemplo 3: esse exemplo descreve os grupos de segurança para a instância especificada.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Saída:

```
GroupId
-------
sg-12345678
sg-45678901
```

Exemplo 4: esse exemplo descreve se a otimização do EBS está habilitada para a instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Saída:

```
EbsOptimized                     : False
```

Exemplo 5: Esse exemplo descreve o atributo ApiTermination 'disable' da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Saída:

```
DisableApiTermination            : False
```

Exemplo 6: Esse exemplo descreve o atributo InitiatedShutdownBehavior 'instance' da instância especificada.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

Saída:

```
InstanceInitiatedShutdownBehavior : stop
```

- Para obter detalhes sobre a API, consulte DescribeInstanceAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2InstanceMetadata`

O código de exemplo a seguir mostra como usar `Get-EC2InstanceMetadata`.

Ferramentas para PowerShell V5

Exemplo 1: lista as categorias disponíveis de metadados de instância que podem ser consultados.

```
Get-EC2InstanceMetadata -ListCategory
```

Saída:

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

Exemplo 2: retorna o ID da imagem de máquina da Amazon (AMI) usada para executar a instância.

```
Get-EC2InstanceMetadata -Category AmiId
```

Saída:

```
ami-b2e756ca
```

Exemplo 3: Esse exemplo consulta o documento de JSON-formatted identidade da instância.

```
Get-EC2InstanceMetadata -Category IdentityDocument
{
  "availabilityZone" : "us-west-2a",
  "devpayProductCodes" : null,
  "marketplaceProductCodes" : null,
  "version" : "2017-09-30",
  "instanceId" : "i-01ed50f7e2607f09e",
  "billingProducts" : [ "bp-6ba54002" ],
  "instanceType" : "t2.small",
  "pendingTime" : "2018-03-07T16:26:04Z",
  "imageId" : "ami-b2e756ca",
  "privateIp" : "10.0.0.171",
  "accountId" : "111122223333",
  "architecture" : "x86_64",
  "kernelId" : null,
  "ramdiskId" : null,
  "region" : "us-west-2"
}
```

Exemplo 4: esse exemplo usa uma consulta de caminho para obter os macs da interface de rede para a instância.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

Saída:

```
02:80:7f:ef:4c:e0/
```

Exemplo 5: Se houver uma função do IAM associada à instância, retornará informações sobre a última vez em que o perfil da instância foi atualizado, incluindo a LastUpdated data da instância InstanceProfileArn, InstanceProfileId e.

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

Saída:

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- Para obter detalhes sobre a API, consulte Get-EC2InstanceMetadata Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2InstanceStatus`

O código de exemplo a seguir mostra como usar `Get-EC2InstanceStatus`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o status da instância especificada.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Saída:

```
AvailabilityZone : us-west-2a
Events           : {}
InstanceId       : i-12345678
InstanceState    : Amazon.EC2.Model.InstanceState
Status           : Amazon.EC2.Model.InstanceStatusSummary
SystemStatus     : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

Saída:

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

Saída:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

Saída:

```
Details           Status
-------           ------
{reachability}    ok
```

- Para obter detalhes sobre a API, consulte DescribeInstanceStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2InternetGateway`

O código de exemplo a seguir mostra como usar `Get-EC2InternetGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o gateway da internet especificado.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Saída:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

Exemplo 2: esse exemplo descreve todos os gateways da internet.

```
Get-EC2InternetGateway
```

Saída:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- Para obter detalhes sobre a API, consulte DescribeInternetGateways Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2KeyPair`

O código de exemplo a seguir mostra como usar `Get-EC2KeyPair`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o par de chaves especificado.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Saída:

```
KeyFingerprint                                                  KeyName
--------------                                                  -------
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f my-key-pair
```

Exemplo 2: esse exemplo descreve todos os pares de chaves.

```
Get-EC2KeyPair
```

- Para obter detalhes sobre a API, consulte DescribeKeyPairs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2NetworkAcl`

O código de exemplo a seguir mostra como usar `Get-EC2NetworkAcl`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a ACL de rede especificada.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Saída:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Exemplo 2: esse exemplo descreve duas regras para a ACL de rede especificada.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Saída:

```
CidrBlock    : 0.0.0.0/0
Egress       : True
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767
```

Exemplo 3: esse exemplo descreve todas as ACLs de rede.

```
Get-EC2NetworkAcl
```

- Para obter detalhes sobre a API, consulte DescribeNetworkAcls Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2NetworkInterface`

O código de exemplo a seguir mostra como usar `Get-EC2NetworkInterface`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Saída:

```
Association        :
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone   : us-west-2c
Description        :
Groups             : {my-security-group}
MacAddress         : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : in-use
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as interfaces de rede.

```
Get-EC2NetworkInterface
```

- Para obter detalhes sobre a API, consulte DescribeNetworkInterfaces Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2NetworkInterfaceAttribute`

O código de exemplo a seguir mostra como usar `Get-EC2NetworkInterfaceAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

Saída:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Exemplo 2: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

Saída:

```
Description        : My description
```

Exemplo 3: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

Saída:

```
Groups             : {my-security-group}
```

Exemplo 4: esse exemplo descreve a interface de rede especificada.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

Saída:

```
SourceDestCheck    : True
```

- Para obter detalhes sobre a API, consulte DescribeNetworkInterfaceAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2PasswordData`

O código de exemplo a seguir mostra como usar `Get-EC2PasswordData`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descriptografa a senha que o Amazon EC2 atribuiu à conta do administrador para a instância especificada do Windows. Quando um arquivo pem foi especificado, a configuração da opção -Decrypt é automaticamente assumida.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Saída:

```
mYZ(PA9?C)Q
```

Exemplo 2: (PowerShell somente Windows) Inspeciona a instância para determinar o nome do par de chaves usado para iniciar a instância e, em seguida, tenta encontrar os dados do par de chaves correspondentes no armazenamento de configuração do AWS Toolkit for Visual Studio. Se os dados do par de chaves forem encontrados, a senha será descriptografada.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Saída:

```
mYZ(PA9?C)Q
```

Exemplo 3: retorna os dados da senha criptografada para a instância.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Saída:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Para obter detalhes sobre a API, consulte GetPasswordData Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2PlacementGroup`

O código de exemplo a seguir mostra como usar `Get-EC2PlacementGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o grupo de posicionamento especificado.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Saída:

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- Para obter detalhes sobre a API, consulte DescribePlacementGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2PrefixList`

O código de exemplo a seguir mostra como usar `Get-EC2PrefixList`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo busca o disponível Serviços da AWS em um formato de lista de prefixos para a região

```
Get-EC2PrefixList
```

Saída:

```
Cidrs                                          PrefixListId PrefixListName
-----                                          ------------ --------------
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23} pl-6fa54006  com.amazonaws.eu-
west-1.dynamodb
```

```
{52.218.0.0/17, 54.231.128.0/19}              pl-6da54004  com.amazonaws.eu-
west-1.s3
```

- Para obter detalhes sobre a API, consulte DescribePrefixLists Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Region`

O código de exemplo a seguir mostra como usar `Get-EC2Region`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as regiões que estão disponíveis para você.

```
Get-EC2Region
```

Saída:

```
Endpoint                            RegionName
--------                            ----------
ec2.eu-west-1.amazonaws.com         eu-west-1
ec2.ap-southeast-1.amazonaws.com    ap-southeast-1
ec2.ap-southeast-2.amazonaws.com    ap-southeast-2
ec2.eu-central-1.amazonaws.com      eu-central-1
ec2.ap-northeast-1.amazonaws.com    ap-northeast-1
ec2.us-east-1.amazonaws.com         us-east-1
ec2.sa-east-1.amazonaws.com         sa-east-1
ec2.us-west-1.amazonaws.com         us-west-1
ec2.us-west-2.amazonaws.com         us-west-2
```

- Para obter detalhes sobre a API, consulte DescribeRegions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2RouteTable`

O código de exemplo a seguir mostra como usar `Get-EC2RouteTable`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve todas as tabelas de rotas.

```
Get-EC2RouteTable
```

Saída:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId  :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId  :
```

Exemplo 2: esse exemplo retorna os detalhes da tabela de rotas especificada.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Exemplo 3: esse exemplo descreve as tabelas de rotas da VPC especificada.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Saída:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte DescribeRouteTables Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2ScheduledInstance`

O código de exemplo a seguir mostra como usar `Get-EC2ScheduledInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a Instância Programada especificada.

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

Saída:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

Exemplo 2: esse exemplo descreve todas as suas Instâncias Programadas.

```
Get-EC2ScheduledInstance
```

- Para obter detalhes sobre a API, consulte DescribeScheduledInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2ScheduledInstanceAvailability`

O código de exemplo a seguir mostra como usar `Get-EC2ScheduledInstanceAvailability`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve uma programação que ocorre toda semana no domingo, começando na data especificada.

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency
 Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

Saída:

```
AvailabilityZone            : us-west-2b
AvailableInstanceCount      : 20
FirstSlotStartTime          : 1/31/2016 8:00:00 AM
HourlyPrice                 : 0.095
InstanceType                : c4.large
MaxTermDurationInDays       : 366
MinTermDurationInDays       : 366
NetworkPlatform             : EC2-VPC
Platform                    : Linux/UNIX
PurchaseToken               : eyJ2IjoiMSIsInMiOjEsImMiOi...
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
SlotDurationInHours         : 23
TotalScheduledInstanceHours : 1219

...
```

Exemplo 2: para restringir os resultados, você pode adicionar filtros para critérios como sistema operacional, rede e tipo de instância.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Para obter detalhes sobre a API, consulte DescribeScheduledInstanceAvailability Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2SecurityGroup`

O código de exemplo a seguir mostra como usar `Get-EC2SecurityGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o grupo de segurança especificado para uma VPC. Ao trabalhar com grupos de segurança pertencentes a uma VPC, você deve usar o ID do grupo de segurança (GroupId parâmetro -), não o nome (GroupName parâmetro -), para fazer referência ao grupo.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Saída:

```
Description         : default VPC security group
GroupId             : sg-12345678
GroupName           : default
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId             : 123456789012
Tags                : {}
VpcId               : vpc-12345678
```

Exemplo 2: Este exemplo descreve o grupo de segurança especificado para EC2-Classic. Ao trabalhar com grupos de segurança, EC2-Classic você pode usar o nome do grupo (GroupName parâmetro -) ou a ID do grupo (GroupId parâmetro -) para fazer referência ao grupo de segurança.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Saída:

```
Description         : my security group
GroupId             : sg-45678901
GroupName           : my-security-group
IpPermissions       : {Amazon.EC2.Model.IpPermission, Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
OwnerId             : 123456789012
Tags                : {}
VpcId               :
```

Exemplo 3: esse exemplo recupera todos os grupos de segurança da vpc-0fc1ff23456b789eb

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Para obter detalhes sobre a API, consulte DescribeSecurityGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Snapshot`

O código de exemplo a seguir mostra como usar `Get-EC2Snapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o snapshot especificado.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Saída:

```
DataEncryptionKeyId :
Description         : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from
 vol-12345678
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            : 100%
SnapshotId          : snap-12345678
StartTime           : 10/23/2014 6:01:28 AM
State               : completed
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 8
```

Exemplo 2: esse exemplo descreve os snapshots que têm uma tag “Name”.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Exemplo 3: Esse exemplo descreve os instantâneos que têm uma tag 'Nome' com o valor 'TestValue'.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

Exemplo 4: esse exemplo descreve todos os snapshots.

```
Get-EC2Snapshot -Owner self
```

- Para obter detalhes sobre a API, consulte DescribeSnapshots Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2SnapshotAttribute`

O código de exemplo a seguir mostra como usar `Get-EC2SnapshotAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o atributo especificado do snapshot especificado.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Saída:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

Exemplo 2: esse exemplo descreve o atributo especificado do snapshot especificado.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

Saída:

```
Group    UserId
-----    ------
all
```

- Para obter detalhes sobre a API, consulte DescribeSnapshotAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2SpotDatafeedSubscription`

O código de exemplo a seguir mostra como usar `Get-EC2SpotDatafeedSubscription`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o feed de dados da instância Spot.

```
Get-EC2SpotDatafeedSubscription
```

Saída:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Para obter detalhes sobre a API, consulte DescribeSpotDatafeedSubscription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2SpotFleetInstance`

O código de exemplo a seguir mostra como usar `Get-EC2SpotFleetInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as instâncias associadas à solicitação de frota spot especificada.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Saída:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- Para obter detalhes sobre a API, consulte DescribeSpotFleetInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2SpotFleetRequest`

O código de exemplo a seguir mostra como usar `Get-EC2SpotFleetRequest`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a solicitação de frota spot especificada.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
 | format-list
```

Saída:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Exemplo 2: esse exemplo descreve todas as suas solicitações de frota spot.

```
Get-EC2SpotFleetRequest
```

- Para obter detalhes sobre a API, consulte DescribeSpotFleetRequests Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2SpotFleetRequestHistory`

O código de exemplo a seguir mostra como usar `Get-EC2SpotFleetRequestHistory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o histórico de solicitação de frota spot especificada.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Saída:

```
HistoryRecords     : {Amazon.EC2.Model.HistoryRecord,
 Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime  : 12/26/2015 8:29:11 AM
NextToken          :
SpotFleetRequestId : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime          : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

Saída:

```
EventInformation                    EventType             Timestamp
----------------                    ---------             ---------
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:05 AM
```

- Para obter detalhes sobre a API, consulte DescribeSpotFleetRequestHistory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2SpotInstanceRequest`

O código de exemplo a seguir mostra como usar `Get-EC2SpotInstanceRequest`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a solicitação de instância spot especificada.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Saída:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 4/8/2015 2:51:33 PM
Fault                    :
InstanceId               : i-12345678
```

```
LaunchedAvailabilityZone : us-west-2b
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.020000
State                    : active
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {Name}
Type                     : one-time
```

Exemplo 2: esse exemplo descreve todas as solicitações de instância spot.

```
Get-EC2SpotInstanceRequest
```

- Para obter detalhes sobre a API, consulte DescribeSpotInstanceRequests Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2SpotPriceHistory`

O código de exemplo a seguir mostra como usar `Get-EC2SpotPriceHistory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém as últimas 10 entradas no histórico de preços spot do tipo de instância especificado e da zona de disponibilidade. Observe que o valor especificado para o AvailabilityZone parâmetro - deve ser válido para o valor da região fornecido ao parâmetro -Region do cmdlet (não mostrado no exemplo) ou definido como padrão no shell. Este exemplo de comando pressupõe que uma região padrão de “us-west-2” tenha sido definida no ambiente.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Saída:

```
AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:39:49 AM
```

```
AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:38:29 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 6:57:13 AM
...
```

- Para obter detalhes sobre a API, consulte DescribeSpotPriceHistory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Subnet`

O código de exemplo a seguir mostra como usar `Get-EC2Subnet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a sub-rede especificada.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Saída:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : available
SubnetId                : subnet-1a2b3c4d
Tags                    : {}
VpcId                   : vpc-12345678
```

Exemplo 2: esse exemplo descreve todas as sub-redes.

```
Get-EC2Subnet
```

- Para obter detalhes sobre a API, consulte DescribeSubnets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Tag`

O código de exemplo a seguir mostra como usar `Get-EC2Tag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo busca as tags do tipo de recurso “image”

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Saída:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

Exemplo 2: esse exemplo busca todas as tags de todos os recursos e as agrupa por tipo de recurso

```
Get-EC2Tag | Group-Object resourcetype
```

Saída:

```
Count Name                      Group
----- ----                      -----
    9 subnet                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   53 instance                  {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    3 route-table               {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    5 security-group            {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
```

```
   30 volume                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    1 internet-gateway          {Amazon.EC2.Model.TagDescription}
    3 network-interface         {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    4 elastic-ip                {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    1 dhcp-options              {Amazon.EC2.Model.TagDescription}
    2 image                     {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    3 vpc                       {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
```

Exemplo 3: esse exemplo exibe todos os recursos com a tag “auto-delete” com o valor “no” para a região em questão

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Saída:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
auto-delete i-0f1bce234d5dd678b   instance     no
auto-delete vol-01d234aa5678901a2 volume       no
auto-delete vol-01234bfb5def6f7b8 volume       no
auto-delete vol-01ccb23f4c5e67890 volume       no
```

Exemplo 4: Este exemplo obtém todos os recursos com a tag 'exclusão automática' com o valor 'nenhum' e outros filtros no próximo canal para analisar apenas os tipos de recursos de 'instância' e, eventualmente, cria a tag 'ThisInstance' para cada recurso de instância, com o valor sendo o próprio ID da instância

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"} |
 Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -ResourceId
 $_.ResourceId -Tag @{Key="ThisInstance";Value=$_.ResourceId}}
```

Exemplo 5: esse exemplo busca tags para todos os recursos da instância, bem como para as chaves “Name”, e as exibe em formato de tabela

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Saída:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

Exemplo 6: Este exemplo valida as permissões para obter tags do EC2 usando o DryRun parâmetro sem realmente buscá-las. Nota: se isso for feito com sucesso, será gerada uma exceção, que é o comportamento esperado.

```
Get-EC2Tag -DryRun $true
```

Saída:

```
Get-EC2Tag: Request would have succeeded, but DryRun flag is set.
```

- Para obter detalhes sobre a API, consulte DescribeTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Volume`

O código de exemplo a seguir mostra como usar `Get-EC2Volume`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve volume do EBS especificado.

```
Get-EC2Volume -VolumeId vol-12345678
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2c
```

```
CreateTime       : 7/17/2015 4:35:19 PM
Encrypted        : False
Iops             : 90
KmsKeyId         :
Size             : 30
SnapshotId       : snap-12345678
State            : in-use
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : standard
```

Exemplo 2: esse exemplo descreve os volumes do EBS que têm o status “available”.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 12/21/2015 2:31:29 PM
Encrypted        : False
Iops             : 60
KmsKeyId         :
Size             : 20
SnapshotId       : snap-12345678
State            : available
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
...
```

Exemplo 3: esse exemplo descreve todos os volumes do EBS.

```
Get-EC2Volume
```

- Para obter detalhes sobre a API, consulte DescribeVolumes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VolumeAttribute`

O código de exemplo a seguir mostra como usar `Get-EC2VolumeAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o atributo especificado do volume especificado.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Saída:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- Para obter detalhes sobre a API, consulte DescribeVolumeAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VolumeStatus`

O código de exemplo a seguir mostra como usar `Get-EC2VolumeStatus`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o status do volume especificado.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Saída:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Saída:

```
Details                         Status
-------                         ------
```

```
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Saída:

```
Name                            Status
----                            ------
io-enabled                      passed
io-performance                  not-applicable
```

- Para obter detalhes sobre a API, consulte DescribeVolumeStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2Vpc`

O código de exemplo a seguir mostra como usar `Get-EC2Vpc`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a VPC especificada.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Saída:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

Exemplo 2: esse exemplo descreve a VPC padrão (só pode haver uma por região). Se sua conta oferece suporte EC2-Classic nessa região, não há VPC padrão.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

Saída:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

Exemplo 3: Este exemplo descreve as VPCs que correspondem ao filtro especificado (ou seja, têm um CIDR que corresponde ao valor '10.0.0. 0/16'e estão no estado' disponível ').

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Exemplo 4: esse exemplo descreve todas as VPCs.

```
Get-EC2Vpc
```

- Para obter detalhes sobre a API, consulte DescribeVpcs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VpcAttribute`

O código de exemplo a seguir mostra como usar `Get-EC2VpcAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o atributo DnsSupport 'habilitar'.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Saída:

```
EnableDnsSupport
----------------
True
```

Exemplo 2: Este exemplo descreve o atributo DnsHostnames 'habilitar'.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Saída:

```
EnableDnsHostnames
------------------
True
```

- Para obter detalhes sobre a API, consulte DescribeVpcAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VpcClassicLink`

O código de exemplo a seguir mostra como usar `Get-EC2VpcClassicLink`.

Ferramentas para PowerShell V5

Exemplo 1: O exemplo acima retorna todas as VPCs com seus ClassicLinkEnabled estados para a região

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Saída:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Para obter detalhes sobre a API, consulte DescribeVpcClassicLink Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VpcClassicLinkDnsSupport`

O código de exemplo a seguir mostra como usar `Get-EC2VpcClassicLinkDnsSupport`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve o status de suporte de ClassicLink DNS das VPCs para a região eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Saída:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Para obter detalhes sobre a API, consulte DescribeVpcClassicLinkDnsSupport Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VpcEndpoint`

O código de exemplo a seguir mostra como usar `Get-EC2VpcEndpoint`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve um ou mais dos endpoints da VPC para a região eu-west-1. Em seguida, ele canaliza a saída para o próximo comando, que seleciona a VpcEndpointId propriedade e retorna o ID VPC da matriz como matriz de strings.

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

Saída:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Exemplo 2: Este exemplo descreve todos os endpoints vpc para a região eu-west-1 e seleciona VpcEndpointId VpcId, ServiceName e as PrivateDnsEnabled propriedades para apresentá-la em formato tabular

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Saída:

```
VpcEndpointId          VpcId                 ServiceName
 PrivateDnsEnabled
-------------          -----                 -----------
 -----------------
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
          True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
          True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
          True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
          True
```

Exemplo 3: esse exemplo exporta o documento de política do endpoint da VPC vpce-01a2ab3f4f5cc6f7d em um arquivo json

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-
Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Para obter detalhes sobre a API, consulte DescribeVpcEndpoints Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VpcEndpointService`

O código de exemplo a seguir mostra como usar `Get-EC2VpcEndpointService`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o serviço de endpoint da VPC do EC2 com o filtro fornecido, nesse caso com.amazonaws.eu-west-1.ecs. Além disso, ele também expande a ServiceDetails propriedade e exibe os detalhes

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

Saída:

```
AcceptanceRequired         : False
AvailabilityZones          : {eu-west-1a, eu-west-1b, eu-west-1c}
BaseEndpointDnsNames       : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                      : amazon
PrivateDnsName             : ecs.eu-west-1.amazonaws.com
ServiceName                : com.amazonaws.eu-west-1.ecs
ServiceType                : {Amazon.EC2.Model.ServiceTypeDetail}
VpcEndpointPolicySupported : False
```

Exemplo 2: Este exemplo recupera todos os serviços do EC2 VPC Endpoint e retorna o ServiceNames “ssm” correspondente

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

Saída:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- Para obter detalhes sobre a API, consulte DescribeVpcEndpointServices Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VpnConnection`

O código de exemplo a seguir mostra como usar `Get-EC2VpnConnection`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve conexão VPN especificada.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Saída:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      : Amazon.EC2.Model.VpnConnectionOptions
```

```
Routes                        : {Amazon.EC2.Model.VpnStaticRoute}
State                         : available
Tags                          : {}
Type                          : ipsec.1
VgwTelemetry                  : {Amazon.EC2.Model.VgwTelemetry,
 Amazon.EC2.Model.VgwTelemetry}
VpnConnectionId               : vpn-12345678
VpnGatewayId                  : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo descreve qualquer conexão VPN cujo estado esteja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Exemplo 3: esse exemplo descreve todas as conexões VPN.

```
Get-EC2VpnConnection
```

- Para obter detalhes sobre a API, consulte DescribeVpnConnections Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EC2VpnGateway`

O código de exemplo a seguir mostra como usar `Get-EC2VpnGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o gateway privado virtual especificado.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Saída:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
```

```
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo descreve qualquer gateway privado virtual cujo estado seja pendente ou disponível.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Exemplo 3: esse exemplo descreve todos os seus gateways privados virtuais.

```
Get-EC2VpnGateway
```

- Para obter detalhes sobre a API, consulte DescribeVpnGateways Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Grant-EC2SecurityGroupEgress`

O código de exemplo a seguir mostra como usar `Grant-EC2SecurityGroupEgress`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo define uma regra de saída para o grupo de segurança especificado para EC2-VPC. A regra concede acesso ao intervalo de endereços IP especificados na porta TCP 80. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
```

```
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: esse exemplo concede acesso ao grupo de segurança de origem especificado na porta TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Para obter detalhes sobre a API, consulte AuthorizeSecurityGroupEgress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Grant-EC2SecurityGroupIngress`

O código de exemplo a seguir mostra como usar `Grant-EC2SecurityGroupIngress`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo define regras de entrada para um grupo de segurança para EC2-VPC. Essas regras concedem acesso a um endereço IP específico para SSH (porta 22) e RDC (porta 3389). Observe que você deve identificar grupos de segurança para EC2-VPC usar a ID do grupo de segurança e não o nome do grupo de segurança. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar os IpPermission objetos.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
```

```
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Exemplo 3: Este exemplo define regras de entrada para um grupo de segurança para EC2-Classic. Essas regras concedem acesso a um endereço IP específico para SSH (porta 22) e RDC (porta 3389). A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission @( $ip1,
 $ip2 )
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar os IpPermission objetos.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission @( $ip1,
 $ip2 )
```

Exemplo 5: esse exemplo concede acesso à porta TCP 8081 do grupo de segurança de origem especificado (sg-1a2b3c4d) ao grupo de segurança especificado (sg-12345678).

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

Exemplo 6: Este exemplo adiciona o CIDR 5.5.5. 5/32 às regras de entrada do grupo de segurança sg-1234abcd para tráfego da porta TCP 22 com uma descrição.

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- Para obter detalhes sobre a API, consulte AuthorizeSecurityGroupIngress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Import-EC2Image`

O código de exemplo a seguir mostra como usar `Import-EC2Image`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo importa uma imagem de máquina virtual de disco único do bucket do Amazon S3 especificado para o Amazon EC2 com um token de idempotência. O exemplo exige que exista um perfil de serviço de importação de VM com o nome padrão “vmimport”, com uma política que permita que o Amazon EC2 acesse o bucket especificado, conforme explicado no tópico Pré-requisitos de importação da VM. Para usar um perfil personalizado, especifique o nome do perfil usando o parâmetro **`-RoleName`**.

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"
```

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

Saída:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- Para obter detalhes sobre a API, consulte ImportImage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Import-EC2KeyPair`

O código de exemplo a seguir mostra como usar `Import-EC2KeyPair`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo importa uma chave pública para o EC2. A primeira linha armazena o conteúdo do arquivo de chave pública (*.pub) na variável **`$publickey`**. Em seguida, o exemplo converte o formato UTF8 do arquivo de chave pública em uma Base64-encoded string e armazena a string convertida na variável. **`$pkbase64`** Na última linha, a chave pública convertida é importada para o EC2. O cmdlet retorna a impressão digital e o nome da chave como resultados.

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
```

```
$pkbase64 =
 [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

Saída:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- Para obter detalhes sobre a API, consulte ImportKeyPair Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Import-EC2Snapshot`

O código de exemplo a seguir mostra como usar `Import-EC2Snapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo importa uma imagem de disco de VM no formato “VMDK” para um snapshot do Amazon EBS. O exemplo exige uma função de serviço de importação de VM com o nome padrão 'vmimport', com uma política que permite o acesso do Amazon EC2 ao bucket especificado, conforme explicado no **`VM Import Prequisites`** tópico em http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html. Para usar uma função personalizada, especifique o nome da função usando o parâmetro. **`-RoleName`**

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

Saída:

```
Description            ImportTaskId               SnapshotTaskDetail
```

```
-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Para obter detalhes sobre a API, consulte ImportSnapshot Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Move-EC2AddressToVpc`

O código de exemplo a seguir mostra como usar `Move-EC2AddressToVpc`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo move uma instância do EC2 com um endereço IP público de 12.345.67.89 para a EC2-VPC plataforma na região Leste dos EUA (Norte da Virgínia).

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Exemplo 2: Esse exemplo envia os resultados de um Get-EC2Instance comando para o Move-EC2AddressToVpc cmdlet. O Get-EC2Instance comando obtém uma instância especificada pelo ID da instância e, em seguida, retorna a propriedade de endereço IP público da instância.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- Para obter detalhes sobre a API, consulte MoveAddressToVpc Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2Address`

O código de exemplo a seguir mostra como usar `New-EC2Address`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo aloca um endereço IP elástico para usar com uma instância em uma VPC.

```
New-EC2Address -Domain Vpc
```

Saída:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

Exemplo 2: Este exemplo aloca um endereço IP elástico para usar com uma instância em. EC2-Classic

```
New-EC2Address
```

Saída:

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- Para obter detalhes sobre a API, consulte AllocateAddress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2CustomerGateway`

O código de exemplo a seguir mostra como usar `New-EC2CustomerGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria o gateway do cliente especificado.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

Saída:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- Para obter detalhes sobre a API, consulte CreateCustomerGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2DhcpOption

O código de exemplo a seguir mostra como usar `New-EC2DhcpOption`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria o conjunto especificado de opções DHCP. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

Saída:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada opção de DHCP.

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @("10.0.0.101","10.0.0.102")

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

Saída:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- Para obter detalhes sobre a API, consulte CreateDhcpOptions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2FlowLog

O código de exemplo a seguir mostra como usar `New-EC2FlowLog`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um log de fluxo do EC2 para a sub-rede subnet-1d234567 para o cloud-watch-log chamado “subnet1-log” para todo o tráfego “REJECT” usando as permissões do perfil “Admin”

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Saída:

```
ClientToken                                  FlowLogIds             Unsuccessful
-----------                                  ----------             ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- Para obter detalhes sobre a API, consulte CreateFlowLogs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2Host

O código de exemplo a seguir mostra como usar `New-EC2Host`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo aloca um host dedicado à sua conta para o tipo de instância e zona de disponibilidade específicos

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType m4.xlarge -
Quantity 1
```

Saída:

```
h-01e23f4cd567890f3
```

- Para obter detalhes sobre a API, consulte AllocateHosts Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2HostReservation

O código de exemplo a seguir mostra como usar `New-EC2HostReservation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo compra a oferta de reserva hro-0c1f23456789d0ab com configurações que correspondem às do seu host dedicado h-01e23f4cd567890f1

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

Saída:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- Para obter detalhes sobre a API, consulte PurchaseHostReservation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2Image

O código de exemplo a seguir mostra como usar `New-EC2Image`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma AMI com o nome e a descrição especificados com base na instância especificada. O Amazon EC2 tenta desligar a instância de forma limpa antes de criar a imagem e reinicia a instância após a conclusão.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

Exemplo 2: esse exemplo cria uma AMI com o nome e a descrição especificados com base na instância especificada. O Amazon EC2 cria a imagem sem desligar e reiniciar a instância; portanto, a integridade do sistema de arquivos na imagem criada não pode ser garantida.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

Exemplo 3: esse exemplo cria uma AMI com três volumes. O primeiro volume é baseado em um snapshot do Amazon EBS. O segundo volume é um volume vazio de 100 GiB do Amazon EBS. O terceiro volume é um volume de armazenamento de instância. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
 "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
$ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
sdc";VirtualName="ephemeral0"})
```

- Para obter detalhes sobre a API, consulte CreateImage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2Instance`

O código de exemplo a seguir mostra como usar `New-EC2Instance`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo inicia uma única instância da AMI especificada em EC2-Classic ou em uma VPC padrão.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Exemplo 2: esse exemplo executa uma única instância da AMI especificada em uma VPC.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

Exemplo 3: para adicionar um volume do EBS ou um volume de armazenamento de instância, defina um mapeamento de dispositivos de blocos e adicione-o ao comando. Esse exemplo adiciona um volume de armazenamento de instância.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Exemplo 4: Para especificar uma das AMIs atuais do Windows, obtenha seu ID de AMI usando Get-SSMLatestEC2Image. Esse exemplo executa uma instância da AMI base atual para Windows Server 2016.

```
$ami = (Get-SSMLatestEC2Image -Path 'ami-windows-latest' -ImageName
 'WINDOWS*2016*English*Core*BASE').Value

New-EC2Instance -ImageId $ami ...
```

Exemplo 5: executa uma instância no ambiente de host dedicado especificado.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

Exemplo 6: essa solicitação executa duas instâncias e aplica uma tag com uma chave de servidor da web e um valor de produção às instâncias. A solicitação também aplica uma tag com uma chave de centro de custos e um valor de cc123 aos volumes criados (nesse caso, o volume do dispositivo raiz de cada instância).

```
$tag1 = @{ Key="webserver"; Value="production" }
$tag2 = @{ Key="cost-center"; Value="cc123" }

$tagspec1 = new-object Amazon.EC2.Model.TagSpecification
$tagspec1.ResourceType = "instance"
$tagspec1.Tags.Add($tag1)

$tagspec2 = new-object Amazon.EC2.Model.TagSpecification
$tagspec2.ResourceType = "volume"
$tagspec2.Tags.Add($tag2)
```

```
New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,
$tagspec2
```

Exemplo 7: Esse exemplo valida as permissões para iniciar uma instância do EC2 usando o DryRun parâmetro sem realmente criar a instância. Nota: se isso for feito com sucesso, será gerada uma exceção, que é o comportamento esperado.

```
New-EC2Instance -ImageId 'ami-12345678' -InstanceType 't2.micro' -KeyName 'my-key-
pair' -Region 'us-west-2' -DryRun $true
```

- Para obter detalhes sobre a API, consulte RunInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2InstanceExportTask`

O código de exemplo a seguir mostra como usar `New-EC2InstanceExportTask`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exporta uma instância interrompida, **`i-0800b00a00EXAMPLE`**, como um disco rígido virtual (VHD) para o bucket do S3 **`testbucket-export-instances-2019`**. O ambiente de destino é**`Microsoft`**, e o parâmetro region é adicionado porque a instância está na **`us-east-1`** região, enquanto a AWS região padrão do usuário não é us-east-1. Para obter o status da tarefa de exportação, copie o valor **`ExportTaskId`** dos resultados desse comando e execute **`Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.`**

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket" -
TargetEnvironment Microsoft -Region us-east-1
```

Saída:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
```

```
StatusMessage         :
```

- Para obter detalhes sobre a API, consulte CreateInstanceExportTask Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2InternetGateway

O código de exemplo a seguir mostra como usar `New-EC2InternetGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um gateway da internet.

```
New-EC2InternetGateway
```

Saída:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- Para obter detalhes sobre a API, consulte CreateInternetGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2KeyPair

O código de exemplo a seguir mostra como usar `New-EC2KeyPair`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo cria um par de chaves e captura a chave privada PEM-encoded RSA em um arquivo com o nome especificado. Quando você estiver usando PowerShell, a codificação deve ser definida como ascii para gerar uma chave válida. Para obter mais informações, consulte Criar, exibir e excluir pares de chaves do Amazon EC2 (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) no Guia do usuário da interface de linha de AWS comando.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- Para obter detalhes sobre a API, consulte CreateKeyPair Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2NetworkAcl

O código de exemplo a seguir mostra como usar `New-EC2NetworkAcl`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma ACL de rede para a VPC especificada.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Saída:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- Para obter detalhes sobre a API, consulte CreateNetworkAcl Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2NetworkAclEntry

O código de exemplo a seguir mostra como usar `New-EC2NetworkAclEntry`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma entrada para a ACL de rede especificada. A regra permite tráfego de entrada de qualquer lugar (0.0.0. 0/0) na porta UDP 53 (DNS) em qualquer sub-rede associada.

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -RuleAction
 allow
```

- Para obter detalhes sobre a API, consulte CreateNetworkAclEntry Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2NetworkInterface

O código de exemplo a seguir mostra como usar `New-EC2NetworkInterface`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria a interface de rede especificada.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Saída:

```
Association        :
Attachment         :
AvailabilityZone   : us-west-2c
Description        : my network interface
Groups             : {my-security-group}
MacAddress         : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.17
PrivateIpAddresses : {}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : pending
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

- Para obter detalhes sobre a API, consulte CreateNetworkInterface Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2PlacementGroup

O código de exemplo a seguir mostra como usar `New-EC2PlacementGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um grupo de posicionamento com o nome especificado.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- Para obter detalhes sobre a API, consulte CreatePlacementGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2Route

O código de exemplo a seguir mostra como usar `New-EC2Route`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria a rota especificada para a tabela de rotas especificada. A rota faz a correspondência de todo o tráfego e o encaminha para o gateway da internet especificado.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId
 igw-1a2b3c4d
```

Saída:

```
True
```

- Para obter detalhes sobre a API, consulte CreateRoute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2RouteTable

O código de exemplo a seguir mostra como usar `New-EC2RouteTable`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma tabela de rotas para a VPC especificada.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Saída:

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- Para obter detalhes sobre a API, consulte CreateRouteTable Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2ScheduledInstance

O código de exemplo a seguir mostra como usar `New-EC2ScheduledInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo inicia a instância programada especificada.

```
New-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012 -
InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678`
-LaunchSpecification_SecurityGroupId sg-12345678
```

- Para obter detalhes sobre a API, consulte RunScheduledInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2ScheduledInstancePurchase

O código de exemplo a seguir mostra como usar `New-EC2ScheduledInstancePurchase`.

Ferramentas para PowerShell V5

Exemplo: esse exemplo compra uma Instância Programada.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
```

```
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Saída:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

- Para obter detalhes sobre a API, consulte PurchaseScheduledInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2SecurityGroup`

O código de exemplo a seguir mostra como usar `New-EC2SecurityGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um grupo de segurança para a VPC especificada.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -
VpcId vpc-12345678
```

Saída:

```
sg-12345678
```

Exemplo 2: Este exemplo cria um grupo de segurança para EC2-Classic.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

Saída:

```
sg-45678901
```

- Para obter detalhes sobre a API, consulte CreateSecurityGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2Snapshot

O código de exemplo a seguir mostra como usar `New-EC2Snapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um snapshot do volume especificado.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Saída:

```
DataEncryptionKeyId :
Description         : This is a test
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            :
SnapshotId          : snap-12345678
StartTime           : 12/22/2015 1:28:42 AM
State               : pending
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 20
```

- Para obter detalhes sobre a API, consulte CreateSnapshot Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2SpotDatafeedSubscription

O código de exemplo a seguir mostra como usar `New-EC2SpotDatafeedSubscription`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um feed de dados da instância Spot.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Saída:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Para obter detalhes sobre a API, consulte CreateSpotDatafeedSubscription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2Subnet`

O código de exemplo a seguir mostra como usar `New-EC2Subnet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma sub-rede com o CIDR especificado.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Saída:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : pending
SubnetId                : subnet-1a2b3c4d
Tag                     : {}
VpcId                   : vpc-12345678
```

- Para obter detalhes sobre a API, consulte CreateSubnet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2Tag

O código de exemplo a seguir mostra como usar `New-EC2Tag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona uma única tag ao recurso especificado. A chave da tag é 'myTag' e o valor da tag é 'my'TagValue. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Exemplo 2: esse exemplo atualiza ou adiciona as tags especificadas ao recurso especificado. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Para obter detalhes sobre a API, consulte CreateTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2Volume

O código de exemplo a seguir mostra como usar `New-EC2Volume`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria o volume especificado.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Saída:

```
Attachments      : {}
AvailabilityZone : us-west-2a
CreateTime       : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size             : 50
SnapshotId       :
State            : creating
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
```

Exemplo 2: esse exemplo de solicitação cria um volume e aplica uma tag com uma chave de pilha e um valor de produção.

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- Para obter detalhes sobre a API, consulte CreateVolume Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2Vpc`

O código de exemplo a seguir mostra como usar `New-EC2Vpc`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma VPC com o CIDR especificado. A Amazon VPC também cria o seguinte para a VPC: um conjunto de opções DHCP padrão, uma tabela de rotas principal e uma ACL de rede padrão.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Saída:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- Para obter detalhes sobre a API, consulte CreateVpc Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2VpcEndpoint

O código de exemplo a seguir mostra como usar `New-EC2VpcEndpoint`.

Ferramentas para PowerShell V5

Exemplo 1: nesse exemplo, crie um endpoint da VPC para o serviço com.amazonaws.eu-west-1.s3 na VPC vpc-0fc1ff23f45b678eb

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Saída:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- Para obter detalhes sobre a API, consulte CreateVpcEndpoint Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EC2VpnConnection

O código de exemplo a seguir mostra como usar `New-EC2VpnConnection`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma conexão VPN entre o gateway privado virtual especificado e o gateway do cliente especificado. A saída inclui as informações de configuração de que o administrador de rede precisa no formato XML.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

Saída:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Exemplo 2: esse exemplo cria a conexão VPN e captura a configuração em um arquivo com o nome especificado.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

Exemplo 3: esse exemplo cria uma conexão VPN, com roteamento estático, entre o gateway privado virtual especificado e o gateway do cliente especificado.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Para obter detalhes sobre a API, consulte CreateVpnConnection Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2VpnConnectionRoute`

O código de exemplo a seguir mostra como usar `New-EC2VpnConnectionRoute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria a rota estática especificada para a conexão VPN especificada.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- Para obter detalhes sobre a API, consulte CreateVpnConnectionRoute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EC2VpnGateway`

O código de exemplo a seguir mostra como usar `New-EC2VpnGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria o gateway privado virtual especificado.

```
New-EC2VpnGateway -Type ipsec.1
```

Saída:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte CreateVpnGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-EC2Address`

O código de exemplo a seguir mostra como usar `Register-EC2Address`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa o endereço IP elástico especificado à instância especificada em uma VPC.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Saída:

```
eipassoc-12345678
```

Exemplo 2: Esse exemplo associa o endereço IP elástico especificado à instância especificada em EC2-Classic.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- Para obter detalhes sobre a API, consulte AssociateAddress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-EC2DhcpOption`

O código de exemplo a seguir mostra como usar `Register-EC2DhcpOption`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa o conjunto de opções de DHCP especificado à VPC especificada.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Exemplo 2: esse exemplo associa o conjunto padrão de opções de DHCP à VPC especificada.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- Para obter detalhes sobre a API, consulte AssociateDhcpOptions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-EC2Image`

O código de exemplo a seguir mostra como usar `Register-EC2Image`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma AMI usando o arquivo de manifesto especificado no Amazon S3.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- Para obter detalhes sobre a API, consulte RegisterImage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-EC2PrivateIpAddress`

O código de exemplo a seguir mostra como usar `Register-EC2PrivateIpAddress`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atribui os endereços IP privados secundários especificados à interface de rede especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

Exemplo 2: esse exemplo cria dois endereços IP privados secundários e os atribui à interface de rede especificada.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- Para obter detalhes sobre a API, consulte AssignPrivateIpAddresses Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-EC2RouteTable`

O código de exemplo a seguir mostra como usar `Register-EC2RouteTable`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa a tabela de rotas especificada à sub-rede especificada.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Saída:

```
rtbassoc-12345678
```

- Para obter detalhes sobre a API, consulte AssociateRouteTable Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Address`

O código de exemplo a seguir mostra como usar `Remove-EC2Address`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo libera o endereço IP elástico especificado para instâncias em uma VPC.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Exemplo 2: Este exemplo libera o endereço IP elástico especificado para instâncias em EC2-Classic.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Para obter detalhes sobre a API, consulte ReleaseAddress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2CapacityReservation`

O código de exemplo a seguir mostra como usar `Remove-EC2CapacityReservation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a reserva de capacidade cr-0c1f2345db6f7cdba

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation (CancelCapacityReservation)"
 on target "cr-0c1f2345db6f7cdba".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
True
```

- Para obter detalhes sobre a API, consulte CancelCapacityReservation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2CustomerGateway`

O código de exemplo a seguir mostra como usar `Remove-EC2CustomerGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o gateway do cliente especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on Target
 "cgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteCustomerGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2DhcpOption`

O código de exemplo a seguir mostra como usar `Remove-EC2DhcpOption`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o conjunto de opções DHCP especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
 "dopt-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteDhcpOptions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2FlowLog`

O código de exemplo a seguir mostra como usar `Remove-EC2FlowLog`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove o FlowLogId fl-01a2b3456a789c01 fornecido

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
 "fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteFlowLogs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Host`

O código de exemplo a seguir mostra como usar `Remove-EC2Host`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo libera o ID de host h-0badafd1dcb2f3456 fornecido

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
 "h-0badafd1dcb2f3456".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Successful            Unsuccessful
----------            ------------
{h-0badafd1dcb2f3456} {}
```

- Para obter detalhes sobre a API, consulte ReleaseHosts Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Instance`

O código de exemplo a seguir mostra como usar `Remove-EC2Instance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo encerra a instância especificada (a instância pode estar em execução ou no estado “interrompido”). O cmdlet solicitará uma confirmação antes de continuar. Use a opção -Force para suprimir a solicitação.

```
Remove-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
```

```
------------                        ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para obter detalhes sobre a API, consulte TerminateInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2InternetGateway`

O código de exemplo a seguir mostra como usar `Remove-EC2InternetGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o gateway da internet especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on Target
 "igw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteInternetGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2KeyPair`

O código de exemplo a seguir mostra como usar `Remove-EC2KeyPair`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o par de chaves especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteKeyPair Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2NetworkAcl`

O código de exemplo a seguir mostra como usar `Remove-EC2NetworkAcl`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a ACL de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
 "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteNetworkAcl Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2NetworkAclEntry`

O código de exemplo a seguir mostra como usar `Remove-EC2NetworkAclEntry`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a regra especificada da ACL de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on Target
 "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteNetworkAclEntry Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2NetworkInterface`

O código de exemplo a seguir mostra como usar `Remove-EC2NetworkInterface`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a interface de rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on Target
 "eni-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteNetworkInterface Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2PlacementGroup`

O código de exemplo a seguir mostra como usar `Remove-EC2PlacementGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o grupo de posicionamento especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
 "my-placement-group".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeletePlacementGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Route`

O código de exemplo a seguir mostra como usar `Remove-EC2Route`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a rota especificada da tabela de rotas especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteRoute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2RouteTable`

O código de exemplo a seguir mostra como usar `Remove-EC2RouteTable`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a tabela de rotas especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
 "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteRouteTable Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2SecurityGroup`

O código de exemplo a seguir mostra como usar `Remove-EC2SecurityGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui o grupo de segurança especificado para EC2-VPC. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
 "sg-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Este exemplo exclui o grupo de segurança especificado para EC2-Classic.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Para obter detalhes sobre a API, consulte DeleteSecurityGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Snapshot`

O código de exemplo a seguir mostra como usar `Remove-EC2Snapshot`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o snapshot especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
 "snap-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteSnapshot Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2SpotDatafeedSubscription`

O código de exemplo a seguir mostra como usar `Remove-EC2SpotDatafeedSubscription`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o feed de dados da instância spot. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2SpotDatafeedSubscription
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
 (DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteSpotDatafeedSubscription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Subnet`

O código de exemplo a seguir mostra como usar `Remove-EC2Subnet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a sub-rede especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target "subnet-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteSubnet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Tag`

O código de exemplo a seguir mostra como usar `Remove-EC2Tag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a tag especificada do recurso especificado, independentemente do valor da tag. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Exemplo 2: esse exemplo exclui a tag especificada do recurso especificado, mas somente se o valor da tag corresponder. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

Exemplo 3: esse exemplo exclui a tag especificada do recurso especificado, independentemente do valor da tag.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Exemplo 4: esse exemplo exclui a tag especificada do recurso especificado, mas somente se o valor da tag corresponder.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Para obter detalhes sobre a API, consulte DeleteTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Volume`

O código de exemplo a seguir mostra como usar `Remove-EC2Volume`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desvincula o volume especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target "vol-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteVolume Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2Vpc`

O código de exemplo a seguir mostra como usar `Remove-EC2Vpc`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a VPC especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteVpc Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2VpnConnection`

O código de exemplo a seguir mostra como usar `Remove-EC2VpnConnection`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui conexão VPN especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
 "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteVpnConnection Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2VpnConnectionRoute`

O código de exemplo a seguir mostra como usar `Remove-EC2VpnConnectionRoute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a rota estática especificada da conexão VPN especificada. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on
 Target "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteVpnConnectionRoute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EC2VpnGateway`

O código de exemplo a seguir mostra como usar `Remove-EC2VpnGateway`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o gateway privado virtual especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
 "vgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteVpnGateway Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Request-EC2SpotFleet`

O código de exemplo a seguir mostra como usar `Request-EC2SpotFleet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma solicitação de frota spot na zona de disponibilidade com o menor preço para o tipo de instância especificado. Se sua conta oferecer suporte EC2-VPC somente, a frota spot iniciará as instâncias na zona de disponibilidade de menor preço que tenha uma sub-rede padrão. Se sua conta for compatível EC2-Classic, a frota spot iniciará as instâncias EC2-Classic na zona de disponibilidade de menor preço. Observe que o preço pago não excederá o preço spot especificado para a solicitação.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-fleet-
role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- Para obter detalhes sobre a API, consulte RequestSpotFleet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Request-EC2SpotInstance`

O código de exemplo a seguir mostra como usar `Request-EC2SpotInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo solicita uma instância spot única na sub-rede especificada. Observe que o grupo de segurança deve ser criado para a VPC que contém a sub-rede especificada e deve ser especificado por ID usando a interface de rede. Ao especificar uma interface de rede, você deve incluir o ID da sub-rede usando a interface de rede.

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
-LaunchSpecification_NetworkInterface $n
```

Saída:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 12/26/2015 7:44:10 AM
Fault                    :
InstanceId               :
LaunchedAvailabilityZone :
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.050000
State                    : open
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {}
Type                     : one-time
```

- Para obter detalhes sobre a API, consulte RequestSpotInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Reset-EC2ImageAttribute`

O código de exemplo a seguir mostra como usar `Reset-EC2ImageAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo redefine o atributo “launchPermission” para seu valor padrão. Por padrão, as AMIs são privadas.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Para obter detalhes sobre a API, consulte ResetImageAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Reset-EC2InstanceAttribute`

O código de exemplo a seguir mostra como usar `Reset-EC2InstanceAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo redefine o atributo NetSupport 'sriov para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Exemplo 2: esse exemplo redefine o atributo “ebsOptimized” para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Exemplo 3: Esse exemplo redefine o atributo 'sourceDestCheck' para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Exemplo 4: Esse exemplo redefine o atributo 'disableApiTermination' para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Exemplo 5: Esse exemplo redefine o atributo 'instanceInitiatedShutdownBehavior' para a instância especificada.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- Para obter detalhes sobre a API, consulte ResetInstanceAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Reset-EC2NetworkInterfaceAttribute`

O código de exemplo a seguir mostra como usar `Reset-EC2NetworkInterfaceAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo redefine a source/destination verificação da interface de rede especificada.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- Para obter detalhes sobre a API, consulte ResetNetworkInterfaceAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Reset-EC2SnapshotAttribute`

O código de exemplo a seguir mostra como usar `Reset-EC2SnapshotAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo redefine o atributo especificado do snapshot especificado.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- Para obter detalhes sobre a API, consulte ResetSnapshotAttribute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Restart-EC2Instance`

O código de exemplo a seguir mostra como usar `Restart-EC2Instance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo reinicia a instância especificada.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Para obter detalhes sobre a API, consulte RebootInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Revoke-EC2SecurityGroupEgress`

O código de exemplo a seguir mostra como usar `Revoke-EC2SecurityGroupEgress`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo remove a regra do grupo de segurança especificado para EC2-VPC. Isso revoga o acesso ao intervalo de endereços IP especificado na porta TCP 80. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: esse exemplo revoga o acesso ao grupo de segurança de origem especificado na porta TCP 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Para obter detalhes sobre a API, consulte RevokeSecurityGroupEgress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Revoke-EC2SecurityGroupIngress`

O código de exemplo a seguir mostra como usar `Revoke-EC2SecurityGroupIngress`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo revoga o acesso à porta TCP 22 do intervalo de endereços especificado para o grupo de segurança especificado para. EC2-VPC Observe que você deve identificar grupos de segurança para EC2-VPC usar a ID do grupo de segurança e não o nome do grupo de segurança. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Exemplo 3: Este exemplo revoga o acesso à porta TCP 22 do intervalo de endereços especificado para o grupo de segurança especificado para. EC2-Classic A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar o IpPermission objeto.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
```

```
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- Para obter detalhes sobre a API, consulte RevokeSecurityGroupIngress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Send-EC2InstanceStatus`

O código de exemplo a seguir mostra como usar `Send-EC2InstanceStatus`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo informa o feedback de status para a instância especificada.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- Para obter detalhes sobre a API, consulte ReportInstanceStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-EC2NetworkAclAssociation`

O código de exemplo a seguir mostra como usar `Set-EC2NetworkAclAssociation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa a ACL de rede especificada à sub-rede da associação de ACL de rede especificada.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Saída:

```
aclassoc-87654321
```

- Para obter detalhes sobre a API, consulte ReplaceNetworkAclAssociation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-EC2NetworkAclEntry`

O código de exemplo a seguir mostra como usar `Set-EC2NetworkAclEntry`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui a entrada especificada da ACL de rede especificada. A nova regra permite tráfego de entrada do endereço especificado para qualquer sub-rede associada.

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -
RuleAction allow
```

- Para obter detalhes sobre a API, consulte ReplaceNetworkAclEntry Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-EC2Route`

O código de exemplo a seguir mostra como usar `Set-EC2Route`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui a rota especificada para a tabela de rotas especificada. A nova rota envia o tráfego especificado ao gateway privado virtual especificado.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId
 vgw-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte ReplaceRoute Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-EC2RouteTableAssociation`

O código de exemplo a seguir mostra como usar `Set-EC2RouteTableAssociation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa a tabela de rotas especificada à sub-rede para a associação da tabela de rotas especificada.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

Saída:

```
rtbassoc-87654321
```

- Para obter detalhes sobre a API, consulte ReplaceRouteTableAssociation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-EC2Instance`

O código de exemplo a seguir mostra como usar `Start-EC2Instance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo inicia a instância especificada.

```
Start-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

Exemplo 2: esse exemplo inicia as instâncias especificadas.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Exemplo 3: esse exemplo inicia o conjunto de instâncias que estão atualmente paradas. Os objetos de instância retornados por Get-EC2Instance são canalizados para Start-EC2Instance. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou superior.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances
 | Start-EC2Instance
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar o filtro para o parâmetro Filter.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

Exemplo 5: Esse exemplo valida as permissões para iniciar uma instância do EC2 usando o DryRun parâmetro sem realmente iniciar a instância. Nota: se isso for feito com sucesso, será gerada uma exceção, que é o comportamento esperado.

```
Start-EC2Instance -InstanceId 'i-0abcdef123456' -Region 'us-west-1' -DryRun $true
```

- Para obter detalhes sobre a API, consulte StartInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-EC2InstanceMonitoring`

O código de exemplo a seguir mostra como usar `Start-EC2InstanceMonitoring`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo de comando habilita o monitoramento detalhado da instância especificada.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Saída:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Para obter detalhes sobre a API, consulte MonitorInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-EC2ImportTask`

O código de exemplo a seguir mostra como usar `Stop-EC2ImportTask`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a tarefa de importação especificada (importação de snapshot ou de imagem). Se necessário, um motivo pode ser fornecido usando o parâmetro **`-CancelReason`**.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- Para obter detalhes sobre a API, consulte CancelImportTask Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-EC2Instance`

O código de exemplo a seguir mostra como usar `Stop-EC2Instance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo interrompe a instância especificada.

```
Stop-EC2Instance -InstanceId i-12345678
```

Saída:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Para obter detalhes sobre a API, consulte StopInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-EC2InstanceMonitoring`

O código de exemplo a seguir mostra como usar `Stop-EC2InstanceMonitoring`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo de comando desabilita o monitoramento detalhado da instância especificada.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Saída:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Para obter detalhes sobre a API, consulte UnmonitorInstances Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-EC2SpotFleetRequest`

O código de exemplo a seguir mostra como usar `Stop-EC2SpotFleetRequest`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a solicitação de frota spot especificada e encerra as instâncias spot associadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Exemplo 2: esse exemplo cancela a solicitação de frota spot especificada sem encerrar as instâncias spot associadas.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- Para obter detalhes sobre a API, consulte CancelSpotFleetRequests Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-EC2SpotInstanceRequest`

O código de exemplo a seguir mostra como usar `Stop-EC2SpotInstanceRequest`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a solicitação de instância spot especificada.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Saída:

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- Para obter detalhes sobre a API, consulte CancelSpotInstanceRequests Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-EC2Address`

O código de exemplo a seguir mostra como usar `Unregister-EC2Address`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo desassocia o endereço IP elástico especificado da instância especificada em uma VPC.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Exemplo 2: Esse exemplo dissocia o endereço IP elástico especificado da instância especificada em. EC2-Classic

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Para obter detalhes sobre a API, consulte DisassociateAddress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-EC2Image`

O código de exemplo a seguir mostra como usar `Unregister-EC2Image`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro da AMI especificada.

```
Unregister-EC2Image -ImageId ami-12345678
```

- Para obter detalhes sobre a API, consulte DeregisterImage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-EC2PrivateIpAddress`

O código de exemplo a seguir mostra como usar `Unregister-EC2PrivateIpAddress`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela a atribuição do endereço IP privado especificado da interface de rede especificada.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- Para obter detalhes sobre a API, consulte UnassignPrivateIpAddresses Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-EC2RouteTable`

O código de exemplo a seguir mostra como usar `Unregister-EC2RouteTable`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a associação especificada entre uma tabela de rotas e uma sub-rede.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte DisassociateRouteTable Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-EC2SecurityGroupRuleIngressDescription`

O código de exemplo a seguir mostra como usar `Update-EC2SecurityGroupRuleIngressDescription`.

Ferramentas para PowerShell V5

Exemplo 1: atualiza a descrição de uma regra de grupo de segurança de entrada.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithUpdatedDescription
```

Exemplo 2: remove a descrição de uma regra de grupo de segurança de entrada (entrada) existente (omitindo o parâmetro na solicitação).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithoutDescription
```

- Para obter detalhes sobre a API, consulte UpdateSecurityGroupRuleDescriptionsIngress Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon ECR usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon ECR.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-ECRLoginCommand`

O código de exemplo a seguir mostra como usar `Get-ECRLoginCommand`.

Ferramentas para PowerShell V5

Exemplo 1: retorna um PSObject contendo informações de login que podem ser usadas para se autenticar em qualquer registro do Amazon ECR ao qual a entidade principal do IAM tenha acesso. As credenciais e o endpoint da região necessários para que a chamada obtenha o token de autorização são obtidos dos padrões do shell (configurados pelos cmdlets **`Set-AWSCredential/Set-DefaultAWSRegion`** ou **`Initialize-AWSDefaultConfiguration`**). Você pode usar a propriedade Command com Invoke-Expression para fazer login no registro especificado ou usar as credenciais retornadas em outras ferramentas que exigem login.

```
Get-ECRLoginCommand
```

Saída:

```
Username      : AWS
Password      : eyJwYXlsb2Fk...kRBVEFfS0VZIn0=
ProxyEndpoint : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
Endpoint      : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
ExpiresAt     : 9/26/2017 6:08:23 AM
Command       : docker login --username AWS --password
 eyJwYXlsb2Fk...kRBVEFfS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

Exemplo 2: recupera um PSObject com informações de login que você usa como entrada para um comando de login do Docker. Você pode especificar qualquer URI de registro do Amazon ECR para se autenticar, desde que a entidade principal do IAM tenha acesso a esse registro.

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
 012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- Para obter detalhes sobre a API, consulte Get-ECRLoginCommand Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon ECS usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon ECS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-ECSClusterDetail`

O código de exemplo a seguir mostra como usar `Get-ECSClusterDetail`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet descreve um ou mais de seus clusters do ECS.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Saída:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- Para obter detalhes sobre a API, consulte DescribeClusters Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ECSClusterList`

O código de exemplo a seguir mostra como usar `Get-ECSClusterList`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet retorna uma lista de clusters do ECS existentes.

```
Get-ECSClusterList
```

Saída:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Para obter detalhes sobre a API, consulte ListClusters Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ECSClusterService`

O código de exemplo a seguir mostra como usar `Get-ECSClusterService`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os serviços em execução no cluster padrão.

```
Get-ECSClusterService
```

Exemplo 2: esse exemplo lista todos os serviços em execução no cluster especificado.

```
Get-ECSClusterService -Cluster myCluster
```

- Para obter detalhes sobre a API, consulte ListServices Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ECSService`

O código de exemplo a seguir mostra como usar `Get-ECSService`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como recuperar detalhes de um serviço específico do cluster padrão.

```
Get-ECSService -Service my-hhtp-service
```

Exemplo 2: esse exemplo mostra como recuperar detalhes de um serviço específico em execução no cluster nomeado.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- Para obter detalhes sobre a API, consulte DescribeServices Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ECSCluster`

O código de exemplo a seguir mostra como usar `New-ECSCluster`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cria um cluster do Amazon ECS.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Saída:

```
ActiveServicesCount               : 0
```

```
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-
ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Para obter detalhes sobre a API, consulte CreateCluster Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ECSService`

O código de exemplo a seguir mostra como usar `New-ECSService`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo de comando cria um serviço no cluster padrão chamado “ecs-simple-service”. O serviço usa a definição de tarefa “ecs-demo” e mantém 10 instanciações dessa tarefa.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Exemplo 2: esse exemplo de comando cria um serviço por trás de um balanceador de carga no cluster padrão chamado “ecs-simple-service”. O serviço usa a definição de tarefa “ecs-demo” e mantém 10 instanciações dessa tarefa.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- Para obter detalhes sobre a API, consulte CreateService Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ECSCluster`

O código de exemplo a seguir mostra como usar `Remove-ECSCluster`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui o cluster do ECS especificado. Cancele o registro de todas as instâncias de contêiner desse cluster antes de excluí-las.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ECSCluster (DeleteCluster)" on target "LAB-ECS".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteCluster Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ECSService`

O código de exemplo a seguir mostra como usar `Remove-ECSService`.

Ferramentas para PowerShell V5

Exemplo 1: exclui o serviço chamado “my-http-service” no cluster padrão. O serviço deve ter uma contagem desejada e uma contagem contínua de 0 antes que você possa excluí-lo. A confirmação será solicitada antes que o comando continue. Para ignorar a solicitação confirmação, adicione a opção -Force.

```
Remove-ECSService -Service my-http-service
```

Exemplo 2: exclui o serviço chamado “my-http-service” no cluster nomeado.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Para obter detalhes sobre a API, consulte DeleteService Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-ECSClusterSetting`

O código de exemplo a seguir mostra como usar `Update-ECSClusterSetting`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet modifica as configurações a serem usadas em um cluster do ECS.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="disabled"}
```

Saída:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-
ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Para obter detalhes sobre a API, consulte UpdateClusterSettings Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-ECSService`

O código de exemplo a seguir mostra como usar `Update-ECSService`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo de comando atualiza o serviço “my-http-service” para usar a definição de tarefa “amazon-ecs-sample”.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Exemplo 2: esse exemplo de comando atualiza a contagem desejada do serviço “my-http-service” para 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Para obter detalhes sobre a API, consulte UpdateService Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon EFS usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon EFS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Edit-EFSMountTargetSecurityGroup`

O código de exemplo a seguir mostra como usar `Edit-EFSMountTargetSecurityGroup`.

Ferramentas para PowerShell V5

Exemplo 1: atualiza os grupos de segurança em vigor para o destino de montagem especificado. Até cinco podem ser especificados, no formato “sg-xxxxxxxx”.

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-
group1,sg-group3
```

- Para obter detalhes sobre a API, consulte ModifyMountTargetSecurityGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EFSFileSystem`

O código de exemplo a seguir mostra como usar `Get-EFSFileSystem`.

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de todos os sistemas de arquivos de propriedade da conta do chamador na região.

```
Get-EFSFileSystem
```

Saída:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : available
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime         : 5/26/2015 4:06:23 PM
CreationToken        : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-4d3c2b1a
...
```

Exemplo 2: retorna os detalhes do sistema de arquivos especificado.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Exemplo 3: retorna os detalhes de um sistema de arquivos usando o token de criação de idempotência que foi especificado no momento em que o sistema de arquivos foi criado.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Para obter detalhes sobre a API, consulte DescribeFileSystems Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EFSMountTarget`

O código de exemplo a seguir mostra como usar `Get-EFSMountTarget`.

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de destinos de montagem associados ao sistema de arquivos especificado.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Saída:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte DescribeMountTargets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EFSMountTargetSecurityGroup`

O código de exemplo a seguir mostra como usar `Get-EFSMountTargetSecurityGroup`.

Ferramentas para PowerShell V5

Exemplo 1: retorna os IDs dos grupos de segurança atualmente atribuídos à interface de rede associada ao destino de montagem.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

Saída:

```
sg-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte DescribeMountTargetSecurityGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EFSTag`

O código de exemplo a seguir mostra como usar `Get-EFSTag`.

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de tags atualmente associadas ao sistema de arquivos especificado.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Saída:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- Para obter detalhes sobre a API, consulte DescribeTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EFSFileSystem`

O código de exemplo a seguir mostra como usar `New-EFSFileSystem`.

Ferramentas para PowerShell V5

Exemplo 1: cria um sistema de arquivos vazio. O token usado para garantir a criação idempotente será gerado automaticamente e poderá ser acessado pelo membro **`CreationToken`** do objeto retornado.

```
New-EFSFileSystem
```

Saída:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : creating
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize
```

Exemplo 2: cria um sistema de arquivos vazio usando um token personalizado para garantir uma criação idempotente.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Para obter detalhes sobre a API, consulte CreateFileSystem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-EFSMountTarget`

O código de exemplo a seguir mostra como usar `New-EFSMountTarget`.

Ferramentas para PowerShell V5

Exemplo 1: cria um destino de montagem para um sistema de arquivos. A sub-rede especificada será usada para determinar a nuvem privada virtual (VPC) na qual o destino de montagem será criado e o endereço IP que será atribuído automaticamente (do intervalo de endereços da sub-rede). O endereço IP atribuído pode ser usado para, em seguida, montar esse sistema de arquivos em uma instância do Amazon EC2. Como nenhum grupo de segurança foi especificado, a interface de rede criada para o destino está associada ao grupo de segurança padrão para a VPC da sub-rede.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Saída:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
```

```
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Exemplo 2: cria um destino de montagem para o sistema de arquivos especificado com endereço IP atribuído automaticamente. A interface de rede criada para o destino de montagem está associada aos grupos de segurança especificados (até cinco, no formato “sg-xxxxxxxx”, podem ser especificados).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Exemplo 3: cria um destino de montagem para o sistema de arquivos especificado com o endereço IP especificado.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- Para obter detalhes sobre a API, consulte CreateMountTarget Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EFSTag

O código de exemplo a seguir mostra como usar `New-EFSTag`.

Ferramentas para PowerShell V5

Exemplo 1: aplica a coleção de tags ao sistema de arquivos especificado. Se uma tag com a chave especificada já existir no sistema de arquivos, o valor da tag será atualizado.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
 @{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

Exemplo 2: define a tag de nome para o sistema de arquivos especificado. Esse valor é retornado junto com outros detalhes do sistema de arquivos quando o Get-EFSFileSystem cmdlet é usado.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- Para obter detalhes sobre a API, consulte CreateTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EFSFileSystem`

O código de exemplo a seguir mostra como usar `Remove-EFSFileSystem`.

Ferramentas para PowerShell V5

Exemplo 1: exclui o sistema de arquivos especificado que não está mais em uso (se o sistema de arquivos tiver destinos de montagem, eles deverão ser removidos primeiro). É solicitada uma confirmação antes que o cmdlet continue. Para ignorar a confirmação, use a opção **`-Force`**.

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte DeleteFileSystem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EFSMountTarget`

O código de exemplo a seguir mostra como usar `Remove-EFSMountTarget`.

Ferramentas para PowerShell V5

Exemplo 1: exclui o destino de montagem especificado. A confirmação será solicitada antes que a operação continue. Para suprimir o prompt, use a opção **`-Force`**. Observe que essa operação interrompe à força qualquer montagem do sistema de arquivos por meio do destino. Convém considerar a desmontagem do sistema de arquivos antes de executar esse comando, se possível.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Para obter detalhes sobre a API, consulte DeleteMountTarget Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EFSTag`

O código de exemplo a seguir mostra como usar `Remove-EFSTag`.

Ferramentas para PowerShell V5

Exemplo 1: exclui a coleção de uma ou mais tags de um sistema de arquivos. É solicitada uma confirmação antes que o cmdlet continue. Para ignorar a confirmação, use a opção **`-Force`**.

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- Para obter detalhes sobre a API, consulte DeleteTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon EKS usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon EKS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-EKSResourceTag`

O código de exemplo a seguir mostra como usar `Add-EKSResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet associa as tags especificadas a um recurso com o resourceArn especificado.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -
Tag @{Name = "EKSPRODCLUSTER"}
```

- Para obter detalhes sobre a API, consulte TagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EKSCluster`

O código de exemplo a seguir mostra como usar `Get-EKSCluster`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet retorna informações descritivas sobre um cluster do Amazon EKS.

```
Get-EKSCluster -Name "PROD"
```

Saída:

```
Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/PROD
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 6:46:17 AM
Endpoint             : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : PROD
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : ACTIVE
Tags                 : {}
Version              : 1.14
```

- Para obter detalhes sobre a API, consulte DescribeCluster Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EKSClusterList`

O código de exemplo a seguir mostra como usar `Get-EKSClusterList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet lista os clusters do Amazon EKS Conta da AWS em sua região especificada.

```
Get-EKSClusterList
```

Saída:

```
 PROD
```

- Para obter detalhes sobre a API, consulte ListClusters Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EKSFargateProfile`

O código de exemplo a seguir mostra como usar `Get-EKSFargateProfile`.

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet retorna informações descritivas sobre um AWS perfil do Fargate.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Saída:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : ACTIVE
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes sobre a API, consulte DescribeFargateProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-EKSFargateProfileList

O código de exemplo a seguir mostra como usar `Get-EKSFargateProfileList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet lista os perfis do AWS Fargate associados ao cluster especificado Conta da AWS em sua região especificada.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Saída:

```
EKSFargate
EKSFargateProfile
```

- Para obter detalhes sobre a API, consulte ListFargateProfiles Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-EKSNodegroup

O código de exemplo a seguir mostra como usar `Get-EKSNodegroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet retorna informações descritivas sobre um grupo de nós do Amazon EKS.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Saída:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
```

```
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes sobre a API, consulte DescribeNodegroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EKSNodegroupList`

O código de exemplo a seguir mostra como usar `Get-EKSNodegroupList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet lista os grupos de nós do Amazon EKS associados ao cluster especificado Conta da AWS em sua região especificada.

```
Get-EKSNodegroupList -ClusterName PROD
```

Saída:

```
ProdEKSNodeGroup
```

- Para obter detalhes sobre a API, consulte ListNodegroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EKSResourceTag`

O código de exemplo a seguir mostra como usar `Get-EKSResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet lista as tags para um recurso do Amazon EKS.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Saída:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- Para obter detalhes sobre a API, consulte ListTagsForResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EKSUpdate`

O código de exemplo a seguir mostra como usar `Get-EKSUpdate`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet retorna informações descritivas sobre uma atualização no cluster do Amazon EKS ou grupo de nós gerenciados associado.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Saída:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Para obter detalhes sobre a API, consulte DescribeUpdate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-EKSUpdateList`

O código de exemplo a seguir mostra como usar `Get-EKSUpdateList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet lista as atualizações associadas a um cluster ou grupo de nós gerenciados do Amazon EKS em sua Conta da AWS região especificada.

```
Get-EKSUpdateList -Name "PROD"
```

Saída:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Para obter detalhes sobre a API, consulte ListUpdates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EKSCluster

O código de exemplo a seguir mostra como usar `New-EKSCluster`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um cluster chamado “prod”.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Saída:

```
Arn                  : arn:aws:eks:us-west-2:012345678901:cluster/prod
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/10/2018 9:25:31 PM
Endpoint             :
Name                 : prod
PlatformVersion      : eks.3
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678901:role/eks-service-role
Status               : CREATING
Version              : 1.10
```

- Para obter detalhes sobre a API, consulte CreateCluster Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EKSFargateProfile

O código de exemplo a seguir mostra como usar `New-EKSFargateProfile`.

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet cria um perfil AWS Fargate para seu cluster Amazon EKS. Você deve ter pelo menos um perfil do Fargate em um cluster para poder programar pods na infraestrutura do Fargate.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
 arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
 @{Namespace="default"}
```

Saída:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:38:21 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName  : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : CREATING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes sobre a API, consulte CreateFargateProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-EKSNodeGroup

O código de exemplo a seguir mostra como usar `New-EKSNodeGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet cria um grupo gerenciado de nós de processamento para um cluster do Amazon EKS. Você pode criar um grupo de nós para o cluster somente se for igual à versão atual

de Kubernetes para o cluster. Todos os grupos de nós são criados com a versão de lançamento mais recente da AMI para a respectiva versão secundária de Kubernetes do cluster.

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
 -DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
 -NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
 "subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"
```

Saída:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes sobre a API, consulte CreateNodegroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EKSCluster`

O código de exemplo a seguir mostra como usar `Remove-EKSCluster`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui um ambiente de gerenciamento de cluster do Amazon EKS.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSCluster (DeleteCluster)" on target "DEV-KUBE-
CL".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 9:33:25 AM
Endpoint             : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : DEV-KUBE-CL
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : DELETING
Tags                 : {}
Version              : 1.14
```

- Para obter detalhes sobre a API, consulte DeleteCluster Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EKSFargateProfile`

O código de exemplo a seguir mostra como usar `Remove-EKSFargateProfile`.

Ferramentas para PowerShell V5

Exemplo 1: Esse cmdlet exclui um AWS perfil do Fargate. Quando você exclui um perfil do Fargate, qualquer pod em execução no Fargate que foi criado com o perfil é excluído.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSFargateProfile (DeleteFargateProfile)" on target
 "EKSFargate".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : DELETING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Para obter detalhes sobre a API, consulte DeleteFargateProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EKSNodegroup`

O código de exemplo a seguir mostra como usar `Remove-EKSNodegroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui um grupo de nós do Amazon EKS para um cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSNodegroup (DeleteNodegroup)" on target
 "ProdEKSNodeGroup".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 11:01:16 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      : Amazon.EKS.Model.NodegroupResources
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : DELETING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Para obter detalhes sobre a API, consulte DeleteNodegroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-EKSResourceTag`

O código de exemplo a seguir mostra como usar `Remove-EKSResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet exclui tags especificadas de um recurso do EKS.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -TagKey "Name"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
 "arn:aws:eks:us-west-2:012345678912:cluster/PROD".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte UntagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-EKSClusterConfig`

O código de exemplo a seguir mostra como usar `Update-EKSClusterConfig`.

Ferramentas para PowerShell V5

Exemplo 1: atualiza uma configuração de cluster do Amazon EKS. O cluster continua funcionando durante a atualização.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Saída:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Para obter detalhes sobre a API, consulte UpdateClusterConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-EKSClusterVersion`

O código de exemplo a seguir mostra como usar `Update-EKSClusterVersion`.

Ferramentas para PowerShell V5

Exemplo 1: esse cmdlet atualiza um cluster do Amazon EKS para a versão especificada do Kubernetes. O cluster continua funcionando durante a atualização.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Saída:

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- Para obter detalhes sobre a API, consulte UpdateClusterVersion Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Elastic Load Balancing - exemplos da versão 1 usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Elastic Load Balancing - Versão 1.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-ELBLoadBalancerToSubnet`

O código de exemplo a seguir mostra como usar `Add-ELBLoadBalancerToSubnet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a sub-rede especificada ao conjunto de sub-redes configuradas para o balanceador de carga especificado. A saída inclui a lista completa de sub-redes.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Saída:

```
subnet-12345678
subnet-87654321
```

- Para obter detalhes sobre a API, consulte AttachLoadBalancerToSubnets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-ELBResourceTag`

O código de exemplo a seguir mostra como usar `Add-ELBResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona as tags especificadas ao balanceador de carga especificado. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar uma tag para o parâmetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Para obter detalhes sobre a API, consulte AddTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-ELBAvailabilityZoneForLoadBalancer`

O código de exemplo a seguir mostra como usar `Disable-ELBAvailabilityZoneForLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a Zona de Disponibilidade especificada ao balanceador de carga especificado. A saída inclui as zonas de disponibilidade restantes.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Saída:

```
us-west-2b
```

- Para obter detalhes sobre a API, consulte DisableAvailabilityZonesForLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Dismount-ELBLoadBalancerFromSubnet`

O código de exemplo a seguir mostra como usar `Dismount-ELBLoadBalancerFromSubnet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a sub-rede especificada ao conjunto de sub-redes configuradas para o balanceador de carga especificado. A saída inclui as sub-redes restantes.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Saída:

```
subnet-87654321
```

- Para obter detalhes sobre a API, consulte DetachLoadBalancerFromSubnets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-ELBLoadBalancerAttribute`

O código de exemplo a seguir mostra como usar `Edit-ELBLoadBalancerAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo habilita o balanceamento de carga entre zonas para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Exemplo 2: esse exemplo desativa a drenagem da conexão para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Exemplo 3: esse exemplo habilita os logs de acesso para o balanceador de carga especificado.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Para obter detalhes sobre a API, consulte ModifyLoadBalancerAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Enable-ELBAvailabilityZoneForLoadBalancer`

O código de exemplo a seguir mostra como usar `Enable-ELBAvailabilityZoneForLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a Zona de Disponibilidade especificada ao balanceador de carga especificado. O resultado inclui a lista completa de zonas de disponibilidade.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Saída:

```
us-west-2a
us-west-2b
```

- Para obter detalhes sobre a API, consulte EnableAvailabilityZonesForLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELBInstanceHealth`

O código de exemplo a seguir mostra como usar `Get-ELBInstanceHealth`.

Ferramentas para PowerShell V5

Exemplo 1: descreve o estado das instâncias registradas com o balanceador de carga especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Saída:

```
Description                   InstanceId                    ReasonCode
      State
-----------                   ----------                    ----------
      -----
N/A                           i-87654321                    N/A
      InService
Instance has failed at lea... i-12345678                    Instance
      OutOfService
```

Exemplo 2: descreve o estado da instância especificada registrada com o balanceador de carga especificado.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Exemplo 3: esse exemplo exibe a descrição completa do estado da instância especificada.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Saída:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- Para obter detalhes sobre a API, consulte DescribeInstanceHealth Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELBLoadBalancer`

O código de exemplo a seguir mostra como usar `Get-ELBLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os nomes de seus balanceadores de carga.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Saída:

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Exemplo 2: esse exemplo descreve o balanceador de carga especificado.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Saída:

```
AvailabilityZones         : {us-west-2a, us-west-2b}
BackendServerDescriptions :
 {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName   : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime               : 4/11/2015 12:12:45 PM
DNSName                   : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
HealthCheck               : Amazon.ElasticLoadBalancing.Model.HealthCheck
Instances                 : {i-207d9717, i-afefb49b}
ListenerDescriptions      : {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
```

```
LoadBalancerName          : my-load-balancer
Policies                  : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                    : internet-facing
SecurityGroups            : {sg-a61988c3}
SourceSecurityGroup       : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                   : {subnet-15aaab61}
VPCId                     : vpc-a01106c2
```

Exemplo 3: Esse exemplo descreve todos os seus balanceadores de carga na AWS região atual.

```
Get-ELBLoadBalancer
```

Exemplo 4: esse exemplo descreve todos os balanceadores de carga em todas as Regiões da AWS disponíveis.

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Para obter detalhes sobre a API, consulte DescribeLoadBalancers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELBLoadBalancerAttribute`

O código de exemplo a seguir mostra como usar `Get-ELBLoadBalancerAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve os atributos para o balanceador de carga especificado.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Saída:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Para obter detalhes sobre a API, consulte DescribeLoadBalancerAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELBLoadBalancerPolicy`

O código de exemplo a seguir mostra como usar `Get-ELBLoadBalancerPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as políticas associadas ao balanceador de carga especificado.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Saída:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Exemplo 2: esse exemplo descreve os atributos da política especificada.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Saída:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Exemplo 3: esse exemplo descreve as políticas predefinidas, incluindo os exemplos de políticas. Os nomes das políticas de amostra têm o prefixo ELBSample-.

```
Get-ELBLoadBalancerPolicy
```

Saída:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-05
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
 SSLNegotiationPolicyType
```

- Para obter detalhes sobre a API, consulte DescribeLoadBalancerPolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELBLoadBalancerPolicyType`

O código de exemplo a seguir mostra como usar `Get-ELBLoadBalancerPolicyType`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os tipos de políticas compatíveis com o Elastic Load Balancing.

```
Get-ELBLoadBalancerPolicyType
```

Saída:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Stickiness policy with session lifet... {CookieExpirationPeriod}
 LBCookieStickinessPolicyType
```

```
Policy that controls authentication ... {PublicKeyPolicyName}
 BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
 SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
 PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
 AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Exemplo 2: esse exemplo descreve o tipo de política especificado.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Saída:

```
Description                            PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                            -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Exemplo 3: esse exemplo exibe a descrição completa do tipo de política especificado.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Saída:

```
Policy that controls whether to include the IP address and port of the originating
 request for TCP messages.
This policy operates on TCP/SSL listeners only
```

- Para obter detalhes sobre a API, consulte DescribeLoadBalancerPolicyTypes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELBResourceTag`

O código de exemplo a seguir mostra como usar `Get-ELBResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as tags para os balanceadores de carga especificados.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Saída:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

Exemplo 2: esse exemplo descreve as tags para o balanceador de carga especificado.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Saída:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- Para obter detalhes sobre a API, consulte DescribeTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Join-ELBSecurityGroupToLoadBalancer`

O código de exemplo a seguir mostra como usar `Join-ELBSecurityGroupToLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui o grupo de segurança atual do balanceador de carga especificado pelo grupo de segurança especificado.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Saída:

```
sg-87654321
```

Exemplo 2: para manter o grupo de segurança atual e especificar um grupo de segurança adicional, especifique os grupos de segurança existentes e os novos.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Saída:

```
sg-12345678
sg-87654321
```

- Para obter detalhes sobre a API, consulte ApplySecurityGroupsToLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELBAppCookieStickinessPolicy`

O código de exemplo a seguir mostra como usar `New-ELBAppCookieStickinessPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma política de permanência com os tempos de vida de sessão persistente do cookie gerado pela aplicação.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
app-cookie-policy -CookieName my-app-cookie
```

- Para obter detalhes sobre a API, consulte CreateAppCookieStickinessPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELBLBCookieStickinessPolicy`

O código de exemplo a seguir mostra como usar `New-ELBLBCookieStickinessPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo gera uma política de persistência com tempos de duração de sessão persistente controlados por um período de validade especificado (em segundos).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy -CookieExpirationPeriod 60
```

Exemplo 2: esse exemplo cria uma política de persistência em que o tempo de vida da sessão persistente é controlado pelo tempo de vida do navegador (usuário-agente).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

- Para obter detalhes sobre a API, consulte CreateLbCookieStickinessPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELBLoadBalancer`

O código de exemplo a seguir mostra como usar `New-ELBLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um balanceador de carga com um receptor HTTP em uma VPC.

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Exemplo 2: Esse exemplo cria um balanceador de carga com um ouvinte HTTP. EC2-Classic

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a` -Listener $httpListener
```

Saída:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Exemplo 3: esse exemplo cria um balanceador de carga com um receptor HTTPS.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-west-2a
 -Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- Para obter detalhes sobre a API, consulte CreateLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELBLoadBalancerListener`

O código de exemplo a seguir mostra como usar `New-ELBLoadBalancerListener`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona um receptor HTTPS ao balanceador de carga especificado.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
 $httpsListener
```

- Para obter detalhes sobre a API, consulte CreateLoadBalancerListeners Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELBLoadBalancerPolicy`

O código de exemplo a seguir mostra como usar `New-ELBLoadBalancerPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma política de protocolo de proxy para um balanceador de carga especificado.

```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -Property
 @{
         AttributeName="ProxyProtocol"
         AttributeValue="True"
    }
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
 $attribute
```

- Para obter detalhes sobre a API, consulte CreateLoadBalancerPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-ELBInstanceWithLoadBalancer`

O código de exemplo a seguir mostra como usar `Register-ELBInstanceWithLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra a instância do EC2 especificada com o balanceador de carga especificado.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Saída:

```
InstanceId
----------
i-12345678
i-87654321
```

- Para obter detalhes sobre a API, consulte RegisterInstancesWithLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELBInstanceFromLoadBalancer`

O código de exemplo a seguir mostra como usar `Remove-ELBInstanceFromLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a instância do EC2 especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBInstanceFromLoadBalancer
 (DeregisterInstancesFromLoadBalancer)" on Target
"Amazon.ElasticLoadBalancing.Model.Instance".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

InstanceId
----------
i-87654321
```

- Para obter detalhes sobre a API, consulte DeregisterInstancesFromLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELBLoadBalancer`

O código de exemplo a seguir mostra como usar `Remove-ELBLoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancer (DeleteLoadBalancer)" on Target "my-
load-balancer".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELBLoadBalancerListener`

O código de exemplo a seguir mostra como usar `Remove-ELBLoadBalancerListener`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o receptor da porta 80 do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort
 80
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerListener (DeleteLoadBalancerListeners)"
 on Target "80".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteLoadBalancerListeners Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELBLoadBalancerPolicy`

O código de exemplo a seguir mostra como usar `Remove-ELBLoadBalancerPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a política especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerPolicy (DeleteLoadBalancerPolicy)" on
 Target "my-duration-cookie-policy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Para obter detalhes sobre a API, consulte DeleteLoadBalancerPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELBResourceTag`

O código de exemplo a seguir mostra como usar `Remove-ELBResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag especificada do balanceador de carga especificado. Você recebe uma solicitação de confirmação antes de prosseguir com a operação, a menos que também especifique o parâmetro Force. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELBResourceTag (RemoveTags)" on target
 "Amazon.ElasticLoadBalancing.Model.TagKeyOnly".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Exemplo 2: Com a versão 2 do Powershell, você deve usar New-Object para criar a tag para o parâmetro Tag.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Para obter detalhes sobre a API, consulte RemoveTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ELBHealthCheck`

O código de exemplo a seguir mostra como usar `Set-ELBHealthCheck`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo define as configurações de verificação de integridade do balanceador de carga especificado.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Saída:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- Para obter detalhes sobre a API, consulte ConfigureHealthCheck Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ELBLoadBalancerListenerSSLCertificate`

O código de exemplo a seguir mostra como usar `Set-ELBLoadBalancerListenerSSLCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui o certificado que encerra as conexões SSL para o receptor especificado.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- Para obter detalhes sobre a API, consulte SetLoadBalancerListenerSslCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ELBLoadBalancerPolicyForBackendServer`

O código de exemplo a seguir mostra como usar `Set-ELBLoadBalancerPolicyForBackendServer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui as políticas da porta especificada pela política especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Exemplo 2: esse exemplo remove todas as políticas associadas à porta especificada.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- Para obter detalhes sobre a API, consulte SetLoadBalancerPoliciesForBackendServer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ELBLoadBalancerPolicyOfListener`

O código de exemplo a seguir mostra como usar `Set-ELBLoadBalancerPolicyOfListener`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo substitui as políticas do receptor especificado pela política especificada.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Exemplo 2: esse exemplo remove todas as políticas associadas ao receptor especificado.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- Para obter detalhes sobre a API, consulte SetLoadBalancerPoliciesOfListener Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Elastic Load Balancing - exemplos da versão 2 usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Elastic Load Balancing - Versão 2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-ELB2ListenerCertificate`

O código de exemplo a seguir mostra como usar `Add-ELB2ListenerCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona um certificado adicional ao receptor especificado.

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618' -
Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'}
```

Saída:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
 False
```

- Para obter detalhes sobre a API, consulte AddListenerCertificates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-ELB2Tag`

O código de exemplo a seguir mostra como usar `Add-ELB2Tag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona uma nova tag ao recurso **`AWS.Tools.ElasticLoadBalancingV2`** especificado.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- Para obter detalhes sobre a API, consulte AddTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-ELB2Listener`

O código de exemplo a seguir mostra como usar `Edit-ELB2Listener`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica a ação padrão do receptor especificado para resposta fixa.

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port
 8080 -DefaultAction $newDefaultAction
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port            : 8080
Protocol        : HTTP
SslPolicy       :
```

- Para obter detalhes sobre a API, consulte ModifyListener Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-ELB2LoadBalancerAttribute`

O código de exemplo a seguir mostra como usar `Edit-ELB2LoadBalancerAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica os atributos do balanceador de carga especificado.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
 'deletion_protection.enabled'; Value = 'true'}
```

Saída:

```
Key                                           Value
---                                           -----
deletion_protection.enabled                   true
access_logs.s3.enabled                        false
access_logs.s3.bucket
access_logs.s3.prefix
idle_timeout.timeout_seconds                  60
routing.http2.enabled                         true
routing.http.drop_invalid_header_fields.enabled false
```

- Para obter detalhes sobre a API, consulte ModifyLoadBalancerAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-ELB2Rule`

O código de exemplo a seguir mostra como usar `Edit-ELB2Rule`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica as configurações especificadas da regra do receptor.

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
    "Values" = "/login1","/login2","/login3"
  }
  "Field" = "path-pattern"
}

Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/
f4f51dfaa033a8cc' -Condition $newRuleCondition
```

Saída:

```
Actions     : {Amazon.ElasticLoadBalancingV2.Model.Action}
```

```
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Para obter detalhes sobre a API, consulte ModifyRule Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-ELB2TargetGroup`

O código de exemplo a seguir mostra como usar `Edit-ELB2TargetGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica as propriedades do grupo de destino especificado.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond
 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Saída:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 60
HealthCheckPath            : /index.html
HealthCheckPort            : 8080
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- Para obter detalhes sobre a API, consulte ModifyTargetGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-ELB2TargetGroupAttribute`

O código de exemplo a seguir mostra como usar `Edit-ELB2TargetGroupAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica o atributo deregistration_delay do grupo de destino especificado.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Saída:

```
Key                                 Value
---                                 -----
stickiness.enabled                  false
deregistration_delay.timeout_seconds  600
stickiness.type                     lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds         0
load_balancing.algorithm.type       round_robin
```

- Para obter detalhes sobre a API, consulte ModifyTargetGroupAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2AccountLimit`

O código de exemplo a seguir mostra como usar `Get-ELB2AccountLimit`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando lista os limites da conta ELB2 para determinada região.

```
Get-ELB2AccountLimit
```

Saída:

```
Max  Name
---  ----
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- Para obter detalhes sobre a API, consulte DescribeAccountLimits Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2Listener`

O código de exemplo a seguir mostra como usar `Get-ELB2Listener`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo descreve os ouvintes do especificado ALB/NLB.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/test-alb/3651b4394dd9a24f
Port            : 80
Protocol        : HTTP
```

```
SslPolicy       :

Certificates    : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/test-alb/3651b4394dd9a24f
Port            : 443
Protocol        : HTTPS
SslPolicy       : ELBSecurityPolicy-2016-08
```

- Para obter detalhes sobre a API, consulte DescribeListeners Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2ListenerCertificate`

O código de exemplo a seguir mostra como usar `Get-ELB2ListenerCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o certificado do receptor especificado.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
 True
```

- Para obter detalhes sobre a API, consulte DescribeListenerCertificates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2LoadBalancer`

O código de exemplo a seguir mostra como usar `Get-ELB2LoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe todos os balanceadores de carga de determinada região.

```
Get-ELB2LoadBalancer
```

Saída:

```
AvailabilityZones     : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
CreatedTime           : 6/22/18 11:21:50 AM
DNSName               : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName      : test-elb1234567890
Scheme                : internet-facing
SecurityGroups        : {}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : network
VpcId                 : vpc-2cf00000
```

- Para obter detalhes sobre a API, consulte DescribeLoadBalancers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2LoadBalancerAttribute`

O código de exemplo a seguir mostra como usar `Get-ELB2LoadBalancerAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando descreve os atributos de determinado balanceador de carga.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Saída:

```
Key                                Value
---                                -----
```

```
access_logs.s3.enabled            false
load_balancing.cross_zone.enabled true
access_logs.s3.prefix
deletion_protection.enabled       false
access_logs.s3.bucket
```

- Para obter detalhes sobre a API, consulte DescribeLoadBalancerAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2Rule`

O código de exemplo a seguir mostra como usar `Get-ELB2Rule`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as regras do receptor para o ARN do receptor especificado.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 1
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 2
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {}
IsDefault  : True
Priority   : default
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf
```

- Para obter detalhes sobre a API, consulte DescribeRules Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2SSLPolicy`

O código de exemplo a seguir mostra como usar `Get-ELB2SSLPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo lista todas as políticas de ouvinte disponíveis para ElasticLoadBalancingV2.

```
Get-ELB2SSLPolicy
```

Saída:

```
Ciphers
                              Name                                  SslProtocols
-------
                              ----                                  ------------
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2016-08            {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-2017-01     {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01     {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06         {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05            {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04     {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08  {TLSv1.2}
```

```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08      {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08      {TLSv1.2}
```

- Para obter detalhes sobre a API, consulte DescribeSslPolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2Tag`

O código de exemplo a seguir mostra como usar `Get-ELB2Tag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as tags para o recurso especificado.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
ResourceArn
           Tags
-----------
           ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Para obter detalhes sobre a API, consulte DescribeTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2TargetGroup`

O código de exemplo a seguir mostra como usar `Get-ELB2TargetGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve o grupo de destino especificado.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /
HealthCheckPort            : traffic-port
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- Para obter detalhes sobre a API, consulte DescribeTargetGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2TargetGroupAttribute`

O código de exemplo a seguir mostra como usar `Get-ELB2TargetGroupAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve os atributos do grupo de destino especificado.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
Key                                     Value
```

```
---                                     -----
stickiness.enabled                      false
deregistration_delay.timeout_seconds    300
stickiness.type                         lb_cookie
stickiness.lb_cookie.duration_seconds   86400
slow_start.duration_seconds             0
load_balancing.algorithm.type           round_robin
```

- Para obter detalhes sobre a API, consulte DescribeTargetGroupAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ELB2TargetHealth`

O código de exemplo a seguir mostra como usar `Get-ELB2TargetHealth`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o status de integridade dos destinos presentes no grupo de destino especificado.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Saída:

```
HealthCheckPort Target                                                  TargetHealth
--------------- ------                                                  ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- Para obter detalhes sobre a API, consulte DescribeTargetHealth Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELB2Listener`

O código de exemplo a seguir mostra como usar `New-ELB2Listener`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um receptor do ALB com a ação padrão “Encaminhar” para enviar tráfego ao grupo de destino especificado.

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  ForwardConfig = @{
    TargetGroups = @(
      @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
    )
    TargetGroupStickinessConfig = @{
      DurationSeconds = 900
      Enabled = $true
    }
  }
  Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -Protocol
 "HTTP" -DefaultAction $defaultAction
```

Saída:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port            : 8001
Protocol        : HTTP
SslPolicy       :
```

- Para obter detalhes sobre a API, consulte CreateListener Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELB2LoadBalancer`

O código de exemplo a seguir mostra como usar `New-ELB2LoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um Application Load Balancer voltado para a internet com duas sub-redes.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Saída:

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
Scheme                : internet-facing
SecurityGroups        : {sg-07c3414abb8811cbd}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : application
VpcId                 : vpc-2cfd7000
```

- Para obter detalhes sobre a API, consulte CreateLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELB2Rule`

O código de exemplo a seguir mostra como usar `New-ELB2Rule`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma regra de receptor com ação de resposta fixa com base no valor do cabeçalho do cliente para o receptor especificado.

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}
```

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
  "httpHeaderConfig" = @{
    "HttpHeaderName" = "customHeader"
    "Values" = "header2","header1"
  }
  "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -Action
 $newRuleAction -Condition $newRuleCondition -Priority 10
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Para obter detalhes sobre a API, consulte CreateRule Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ELB2TargetGroup`

O código de exemplo a seguir mostra como usar `New-ELB2TargetGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um grupo de destino com os parâmetros fornecidos.

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

Saída:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /index.html
```

```
HealthCheckPort            : 80
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 2
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
TargetGroupName            : NewTargetGroup
TargetType                 : instance
UnhealthyThresholdCount    : 5
VpcId                      : vpc-2cfd7000
```

- Para obter detalhes sobre a API, consulte CreateTargetGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-ELB2Target`

O código de exemplo a seguir mostra como usar `Register-ELB2Target`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra a instância “i-0672a4c4cdeae3111” com o grupo de destino especificado.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
 'i-0672a4c4cdeae3111'}
```

- Para obter detalhes sobre a API, consulte RegisterTargets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELB2Listener`

O código de exemplo a seguir mostra como usar `Remove-ELB2Listener`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o receptor especificado.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

Exemplo 2: esse exemplo remove o receptor especificado do balanceador de carga.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes sobre a API, consulte DeleteListener Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELB2ListenerCertificate`

O código de exemplo a seguir mostra como usar `Remove-ELB2ListenerCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o certificado especificado do grupo de destino especificado.

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -ListenerArn
```

```
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
 (RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes sobre a API, consulte RemoveListenerCertificates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELB2LoadBalancer`

O código de exemplo a seguir mostra como usar `Remove-ELB2LoadBalancer`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o balanceador de carga especificado.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes sobre a API, consulte DeleteLoadBalancer Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELB2Rule`

O código de exemplo a seguir mostra como usar `Remove-ELB2Rule`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a regra especificada do receptor

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes sobre a API, consulte DeleteRule Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELB2Tag`

O código de exemplo a seguir mostra como usar `Remove-ELB2Tag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag para a chave especificada.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes sobre a API, consulte RemoveTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-ELB2TargetGroup`

O código de exemplo a seguir mostra como usar `Remove-ELB2TargetGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o grupo de destino especificado.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
 target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testsssss/4e0b6076bc6483a7".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Para obter detalhes sobre a API, consulte DeleteTargetGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ELB2IpAddressType`

O código de exemplo a seguir mostra como usar `Set-ELB2IpAddressType`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo altera o tipo de endereço IP do balanceador de carga de 'IPv4' para ". DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Saída:

```
Value
-----
dualstack
```

- Para obter detalhes sobre a API, consulte SetIpAddressType Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ELB2RulePriority`

O código de exemplo a seguir mostra como usar `Set-ELB2RulePriority`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera a prioridade da regra de receptor especificada.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Saída:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 11
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- Para obter detalhes sobre a API, consulte SetRulePriorities Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ELB2SecurityGroup`

O código de exemplo a seguir mostra como usar `Set-ELB2SecurityGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona o grupo de segurança “sg-07c3414abb8811cbd” ao balanceador de carga especificado.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Saída:

```
sg-07c3414abb8811cbd
```

- Para obter detalhes sobre a API, consulte SetSecurityGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-ELB2Subnet`

O código de exemplo a seguir mostra como usar `Set-ELB2Subnet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo modifica as sub-redes do balanceador de carga especificado.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Saída:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- Para obter detalhes sobre a API, consulte SetSubnets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-ELB2Target`

O código de exemplo a seguir mostra como usar `Unregister-ELB2Target`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro da instância “i-0672a4c4cdeae3111” do grupo de destino especificado.

```
$targetDescription = New-Object
 Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- Para obter detalhes sobre a API, consulte DeregisterTargets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon FSx usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon FSx.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-FSXResourceTag`

O código de exemplo a seguir mostra como usar `Add-FSXResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona tags ao recurso em questão.

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"}
```

Saída:

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- Para obter detalhes sobre a API, consulte TagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-FSXBackup`

O código de exemplo a seguir mostra como usar `Get-FSXBackup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo busca backups criados desde ontem para o ID do sistema de arquivos fornecido.

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-
Object CreationTime -gt (Get-Date).AddDays(-1)
```

Saída:

```
BackupId        : backup-01dac234e56782bcc
CreationTime    : 6/14/2019 3:35:14 AM
FailureDetails  :
FileSystem      : Amazon.FSx.Model.FileSystem
KmsKeyId        : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f1-
e1234c5af123
Lifecycle       : AVAILABLE
ProgressPercent : 100
ResourceARN     : arn:aws:fsx:eu-west-1:123456789012:backup/backup-01dac234e56782bcc
Tags            : {}
Type            : AUTOMATIC
```

- Para obter detalhes sobre a API, consulte DescribeBackups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-FSXFileSystem

O código de exemplo a seguir mostra como usar `Get-FSXFileSystem`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna a descrição de determinado filesystemId.

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

Saída:

```
CreationTime         : 1/17/2019 9:55:30 AM
DNSName              : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails       :
FileSystemId         : fs-01cd23bc4bdf5678a
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-8bde-
a9f0-e1234c5af678
Lifecycle            : AVAILABLE
LustreConfiguration  :
NetworkInterfaceIds  : {eni-07d1dda1322b7e209}
OwnerId              : 123456789012
ResourceARN          : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a
StorageCapacity      : 300
SubnetIds            : {subnet-7d123456}
Tags                 : {FSx-Service}
VpcId                : vpc-41cf2b3f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Para obter detalhes sobre a API, consulte DescribeFileSystems Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-FSXResourceTagList

O código de exemplo a seguir mostra como usar `Get-FSXResourceTagList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as tags para o ARN de recurso fornecido.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

Saída:

```
Key         Value
---         -----
FSx-Service Windows
Users       Dev
```

- Para obter detalhes sobre a API, consulte ListTagsForResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-FSXBackup`

O código de exemplo a seguir mostra como usar `New-FSXBackup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um backup do sistema de arquivos fornecido.

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

Saída:

```
BackupId        : backup-0b1fac2345623456ba
CreationTime    : 6/14/2019 5:37:17 PM
FailureDetails  :
FileSystem      : Amazon.FSx.Model.FileSystem
KmsKeyId        : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f3-
e1234c5af678
Lifecycle       : CREATING
ProgressPercent : 0
ResourceARN     : arn:aws:fsx:eu-west-1:123456789012:backup/
backup-0b1fac2345623456ba
Tags            : {}
Type            : USER_INITIATED
```

- Para obter detalhes sobre a API, consulte CreateBackup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-FSXFileSystem

O código de exemplo a seguir mostra como usar `New-FSXFileSystem`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um sistema de arquivos do Windows de 300 GB, permitindo acesso pela sub-rede especificada, que permite throughput de até 8 megabytes por segundo. O novo sistema de arquivos é automaticamente associado ao Microsoft Active Directory especificado.

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
 300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

Saída:

```
CreationTime         : 12/10/2018 6:06:59 PM
DNSName              : fs-abcdef01234567890.example.com
FailureDetails       :
FileSystemId         : fs-abcdef01234567890
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:us-west-2:123456789012:key/a1234567-252c-45e9-
afaa-123456789abc
Lifecycle            : CREATING
LustreConfiguration  :
NetworkInterfaceIds  : {}
OwnerId              : 123456789012
ResourceARN          : arn:aws:fsx:us-west-2:123456789012:file-system/fs-
abcdef01234567890
StorageCapacity      : 300
SubnetIds            : {subnet-1a2b3c4d5e6f}
Tags                 : {}
VpcId                : vpc-1a2b3c4d5e6f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Para obter detalhes sobre a API, consulte CreateFileSystem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-FSXFileSystemFromBackup

O código de exemplo a seguir mostra como usar `New-FSXFileSystemFromBackup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um sistema de arquivos do Amazon FSx por meio de um backup do Amazon FSx para Windows File Server existente.

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

Saída:

```
CreationTime         : 8/8/2019 12:59:58 PM
DNSName              : fs-012ff34e56789120.ktmsad.local
FailureDetails       :
FileSystemId         : fs-012ff34e56789120
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-1bde-
a2f3-e4567c8a9321
Lifecycle            : CREATING
LustreConfiguration  :
NetworkInterfaceIds  : {}
OwnerId              : 933303704102
ResourceARN          : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-012ff34e56789120
StorageCapacity      : 300
SubnetIds            : {subnet-fa1ae23c}
Tags                 : {tag:Name}
VpcId                : vpc-12cf3b4f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Para obter detalhes sobre a API, consulte CreateFileSystemFromBackup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Remove-FSXBackup

O código de exemplo a seguir mostra como usar `Remove-FSXBackup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o ID de backup fornecido.

```
Remove-FSXBackup -BackupId $backupID
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXBackup (DeleteBackup)" on target
 "backup-0bbca1e2345678e12".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

BackupId                Lifecycle
--------                ---------
backup-0bbca1e2345678e12 DELETED
```

- Para obter detalhes sobre a API, consulte DeleteBackup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-FSXFileSystem`

O código de exemplo a seguir mostra como usar `Remove-FSXFileSystem`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o ID do sistema de arquivos do FSX fornecido.

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXFileSystem (DeleteFileSystem)" on target
 "fs-012ff34e567890120".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

```
FileSystemId         Lifecycle WindowsResponse
------------         --------- ---------------
fs-012ff34e567890120 DELETING  Amazon.FSx.Model.DeleteFileSystemWindowsResponse
```

- Para obter detalhes sobre a API, consulte DeleteFileSystem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-FSXResourceTag`

O código de exemplo a seguir mostra como usar `Remove-FSXResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag de recurso para o ARN de recurso do sistema de arquivos do FSX fornecido.

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXResourceTag (UntagResource)" on target
 "arn:aws:fsx:eu-west-1:933303704102:file-system/fs-07cd45bc6bdf2674a".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte UntagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-FSXFileSystem`

O código de exemplo a seguir mostra como usar `Update-FSXFileSystem`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo atualiza os dias de retenção automática de backup do sistema de arquivos FSX via UpdateFileSystemWindowsConfiguration.

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
```

```
$UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
$UpdateFSXWinConfig
```

Saída:

```
CreationTime         : 1/17/2019 9:55:30 AM
DNSName              : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails       :
FileSystemId         : fs-01cd23bc4bdf5678a
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-
a1f2-e1234c5af678
Lifecycle            : AVAILABLE
LustreConfiguration  :
NetworkInterfaceIds  : {eni-01cd23bc4bdf5678a}
OwnerId              : 933303704102
ResourceARN          : arn:aws:fsx:eu-west-1:933303704102:file-system/
fs-07cd45bc6bdf2674a
StorageCapacity      : 300
SubnetIds            : {subnet-1d234567}
Tags                 : {FSx-Service}
VpcId                : vpc-23cf4b5f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- Para obter detalhes sobre a API, consulte UpdateFileSystem Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon Glacier usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Amazon Glacier.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-GLCJob`

O código de exemplo a seguir mostra como usar `Get-GLCJob`.

Ferramentas para PowerShell V5

Exemplo 1: exibir detalhes do trabalho especificado. Quando o trabalho for concluído com êxito, o Read-GCJobOutput cmdlet poderá ser usado para recuperar o conteúdo do trabalho (um arquivo ou uma lista de inventário) no sistema de arquivos local.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Saída:

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
JobDescription               :
JobId                        : op1x...JSbthM
JobOutputPath                :
OutputLocation               :
RetrievalByteRange           : 0-38034479
SelectParameters             :
SHA256TreeHash               : 79f3ea754c02f58...dc57bf4395b
SNSTopic                     :
StatusCode                   : InProgress
StatusMessage                :
Tier                         : Standard
VaultARN                     : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- Para obter detalhes sobre a API, consulte DescribeJob Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-GLCVault`

O código de exemplo a seguir mostra como usar `New-GLCVault`.

Ferramentas para PowerShell V5

Exemplo 1: criar novo cofre para a conta do usuário. Como nenhum valor foi fornecido ao AccountId parâmetro -, os cmdlets usam um padrão de "-" indicando a conta atual.

```
New-GLCVault -VaultName myvault
```

Saída:

```
/01234567812/vaults/myvault
```

- Para obter detalhes sobre a API, consulte CreateVault Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Read-GLCJobOutput`

O código de exemplo a seguir mostra como usar `Read-GLCJobOutput`.

Ferramentas para PowerShell V5

Exemplo 1: baixar o conteúdo do arquivo que foi agendado para recuperação na tarefa especificada e armazená-lo em um arquivo no disco. O download valida a soma de verificação para você, se houver uma disponível. Se desejado, toda a resposta, incluindo a soma de verificação, pode ser exibida especificando **`-Select '*'`**.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp
\blue.bin"
```

- Para obter detalhes sobre a API, consulte GetJobOutput Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-GLCJob`

O código de exemplo a seguir mostra como usar `Start-GLCJob`.

Ferramentas para PowerShell V5

Exemplo 1: iniciar um trabalho para recuperar um arquivo do cofre especificado de propriedade do usuário. O status do trabalho pode ser verificado usando o Get-GLCJob cmdlet. Quando o trabalho for concluído com êxito, o Read-GCJobOutput cmdlet poderá ser usado para recuperar o conteúdo do arquivo no sistema de arquivos local.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Saída:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM
```

- Para obter detalhes sobre a API, consulte InitiateJob Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-GLCArchive`

O código de exemplo a seguir mostra como usar `Write-GLCArchive`.

Ferramentas para PowerShell V5

Exemplo 1: carregar um único arquivo no cofre especificado, exibindo o ID do arquivo e a soma de verificação computada.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Saída:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Exemplo 2: carregar o conteúdo de uma hierarquia de pastas no cofre especificado na conta do usuário. Para cada arquivo carregado, o cmdlet emite o nome do arquivo, o ID do arquivo correspondente e a soma de verificação computada do arquivo.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Saída:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- Para obter detalhes sobre a API, consulte UploadArchive Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS Glue exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS Glue.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`New-GLUEJob`

O código de exemplo a seguir mostra como usar `New-GLUEJob`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma nova tarefa no AWS Glue. O valor do nome do comando é sempre**`glueetl`**. AWS O Glue suporta a execução de scripts de trabalho escritos em Python ou Scala. Neste exemplo, o script de trabalho (MyTestGlueJob.py) é escrito em Python. Os parâmetros do Python são especificados na **`$DefArgs`** variável e depois passados para o PowerShell comando no **`DefaultArguments`** parâmetro, que aceita uma tabela de hash. Os parâmetros na **`$JobParams`** variável vêm da CreateJob API, documentados no tópico Jobs (https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) da referência da API AWS Glue.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
```

```
    }

New-GlueJob @JobParams
```

- Para obter detalhes sobre a API, consulte CreateJob Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS Health exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS Health.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-HLTHEvent`

O código de exemplo a seguir mostra como usar `Get-HLTHEvent`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna eventos do AWS Personal Health Dashboard. O usuário adiciona o parâmetro -Region para ver os eventos disponíveis para o serviço na região Leste dos EUA (Norte da Virgínia), mas o Filter_Region parâmetro - filtra os eventos registrados nas regiões UE (Londres) e Oeste dos EUA (Oregon) (eu-west-2 e us-west-2). O Filter_StartTime parâmetro - filtra por um intervalo de vezes em que os eventos podem começar, enquanto o Filter_EndTime parâmetro - filtra por um intervalo de vezes em que os eventos podem terminar. O resultado é um evento de manutenção programado para o RDS que começa dentro do Filter_StartTime intervalo especificado e termina dentro do Filter_EndTime intervalo programado.

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

Saída:

```
Arn               : arn:aws:health:us-west-2::event/RDS/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321
AvailabilityZone  :
EndTime           : 3/21/2019 2:00:00 PM
EventTypeCategory : scheduledChange
EventTypeCode     : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime   : 2/28/2019 2:26:07 PM
Region            : us-west-2
Service           : RDS
StartTime         : 3/14/2019 2:00:00 PM
StatusCode        : open
```

- Para obter detalhes sobre a API, consulte DescribeEvents Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de IAM usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o IAM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-IAMClientIDToOpenIDConnectProvider`

O código de exemplo a seguir mostra como usar `Add-IAMClientIDToOpenIDConnectProvider`.

Ferramentas para PowerShell V5

Exemplo 1: este comando adiciona o ID do cliente (ou público) **`my-application-ID`** ao provedor OIDC existente denominado **`server.example.com`**.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Para obter detalhes sobre a API, consulte AddClientIdToOpenIdConnectProvider Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-IAMRoleTag`

O código de exemplo a seguir mostra como usar `Add-IAMRoleTag`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo adiciona uma tag ao perfil no Identity Management Service

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para obter detalhes sobre a API, consulte TagRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-IAMRoleToInstanceProfile`

O código de exemplo a seguir mostra como usar `Add-IAMRoleToInstanceProfile`.

Ferramentas para PowerShell V5

Exemplo 1: este comando adiciona o perfil denominado **`S3Access`** a um perfil de instância existente denominado **`webserver`**. Para criar o perfil de instância, use o comando **`New-IAMInstanceProfile`**. Depois de criar o perfil de instância e associá-lo a um perfil usando

esse comando, você pode anexá-lo a uma instância do EC2. Para isso, use o cmdlet **`New-EC2Instance`** com o parâmetro **`InstanceProfile_Arn`** ou **`InstanceProfile-Name`** para executar a nova instância.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- Para obter detalhes sobre a API, consulte AddRoleToInstanceProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-IAMUserTag`

O código de exemplo a seguir mostra como usar `Add-IAMUserTag`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo adiciona uma tag ao usuário no Identity Management Service

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Para obter detalhes sobre a API, consulte TagUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Add-IAMUserToGroup`

O código de exemplo a seguir mostra como usar `Add-IAMUserToGroup`.

Ferramentas para PowerShell V5

Exemplo 1: este comando adiciona o usuário chamado **`Bob`** ao grupo denominado **`Admins`**.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Para obter detalhes sobre a API, consulte AddUserToGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Disable-IAMMFADevice`

O código de exemplo a seguir mostra como usar `Disable-IAMMFADevice`.

Ferramentas para PowerShell V5

Exemplo 1: este comando desabilita o dispositivo de MFA de hardware associado ao usuário **Bob** que tem o número de série **123456789012**.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Exemplo 2: este comando desativa o dispositivo de MFA virtual associado ao usuário **David** que tem o ARN **arn:aws:iam::210987654321:mfa/David**. Observe que o dispositivo de MFA virtual não é excluído da conta. O dispositivo virtual ainda está presente e aparece na saída do comando **Get-IAMVirtualMFADevice**. Antes de criar um dispositivo de MFA virtual para o mesmo usuário, você deve excluir o antigo usando o comando **Remove-IAMVirtualMFADevice**.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/
David"
```

- Para obter detalhes sobre a API, consulte DeactivateMfaDevice Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Edit-IAMPassword

O código de exemplo a seguir mostra como usar `Edit-IAMPassword`.

Ferramentas para PowerShell V5

Exemplo 1: este comando altera a senha do usuário que está executando o comando. Esse comando pode ser chamado somente por usuários do IAM. Se esse comando for chamado quando você estiver conectado com as credenciais da AWS conta (raiz), o comando retornará um erro. **InvalidUserType**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Para obter detalhes sobre a API, consulte ChangePassword Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Enable-IAMMFADevice

O código de exemplo a seguir mostra como usar `Enable-IAMMFADevice`.

Ferramentas para PowerShell V5

Exemplo 1: este comando habilita o dispositivo de MFA de hardware com o número de série **987654321098** e associa o dispositivo ao usuário **Bob**. Ele inclui os dois primeiros códigos em sequência do dispositivo.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Exemplo 2: este exemplo cria e habilita um dispositivo de MFA virtual. O primeiro comando cria o dispositivo virtual e retorna a representação de objeto do dispositivo na variável **$MFADevice**. Você pode usar as propriedades **.Base32StringSeed** ou **QRCodePng** para configurar a aplicação de software do usuário. O comando final atribui o dispositivo ao usuário **David**, identificando o dispositivo pelo número de série. O comando também sincroniza o dispositivo com AWS a inclusão dos dois primeiros códigos em sequência do dispositivo MFA virtual.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- Para obter detalhes sobre a API, consulte EnableMfaDevice Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-IAMAccessKey

O código de exemplo a seguir mostra como usar `Get-IAMAccessKey`.

Ferramentas para PowerShell V5

Exemplo 1: este comando lista as chaves de acesso do usuário do IAM chamado **Bob**. Observe que não é possível listar as chaves de acesso secretas dos usuários do IAM. Se as chaves de acesso secretas forem perdidas, você deverá criar novas chaves de acesso com o cmdlet **New-IAMAccessKey**.

```
Get-IAMAccessKey -UserName "Bob"
```

Saída:

```
AccessKeyId                CreateDate                  Status              UserName
-----------                ----------                  ------              --------
AKIAIOSFODNN7EXAMPLE       12/3/2014 10:53:41 AM       Active              Bob
AKIAI44QH8DHBEXAMPLE       6/6/2013 8:42:26 PM         Inactive            Bob
```

- Para obter detalhes sobre a API, consulte ListAccessKeys Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMAccessKeyLastUsed`

O código de exemplo a seguir mostra como usar `Get-IAMAccessKeyLastUsed`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o nome de usuário proprietário e as informações do último uso da chave de acesso fornecida.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Para obter detalhes sobre a API, consulte GetAccessKeyLastUsed Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMAccountAlias`

O código de exemplo a seguir mostra como usar `Get-IAMAccountAlias`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o alias da conta da Conta da AWS.

```
Get-IAMAccountAlias
```

Saída:

```
ExampleCo
```

- Para obter detalhes sobre a API, consulte ListAccountAliases Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMAccountAuthorizationDetail`

O código de exemplo a seguir mostra como usar `Get-IAMAccountAuthorizationDetail`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo obtém detalhes de autorização sobre as identidades na AWS conta e exibe a lista de elementos do objeto retornado, incluindo usuários, grupos e funções. Por exemplo, a propriedade **`UserDetailList`** exibe detalhes sobre os usuários. Informações semelhantes estão disponíveis nas propriedades **`RoleDetailList`** e **`GroupDetailList`**.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Saída:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Saída:

```
Arn            : arn:aws:iam::123456789012:user/Administrator
CreateDate     : 10/16/2014 9:03:09 AM
GroupList      : {Administrators}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE1
UserName       : Administrator
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/Bob
CreateDate     : 4/6/2015 12:54:42 PM
GroupList      : {Developers}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE2
UserName       : bab
UserPolicyList : {}
```

```
Arn             : arn:aws:iam::123456789012:user/BackupToS3
CreateDate      : 1/27/2015 10:15:08 AM
GroupList       : {Backup}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE3
UserName        : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- Para obter detalhes sobre a API, consulte GetAccountAuthorizationDetails Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `Get-IAMAccountPasswordPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre a política de senha da conta atual. Se nenhuma política de senha estiver definida na conta, o comando retorna um erro **`NoSuchEntity`**.

```
Get-IAMAccountPasswordPolicy
```

Saída:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- Para obter detalhes sobre a API, consulte GetAccountPasswordPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMAccountSummary`

O código de exemplo a seguir mostra como usar `Get-IAMAccountSummary`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna informações sobre o uso atual da entidade do IAM e das cotas atuais da entidade do IAM na Conta da AWS.

```
Get-IAMAccountSummary
```

Saída:

```
Key                                     Value
Users                                   7
GroupPolicySizeQuota                    5120
PolicyVersionsInUseQuota                10000
ServerCertificatesQuota                 20
AccountSigningCertificatesPresent       0
AccountAccessKeysPresent                0
Groups                                  3
UsersQuota                              5000
RolePolicySizeQuota                     10240
UserPolicySizeQuota                     2048
GroupsPerUserQuota                      10
AssumeRolePolicySizeQuota               2048
AttachedPoliciesPerGroupQuota           2
Roles                                   9
VersionsPerPolicyQuota                  5
GroupsQuota                             100
PolicySizeQuota                         5120
Policies                                5
RolesQuota                              250
ServerCertificates                      0
AttachedPoliciesPerRoleQuota            2
MFADevicesInUse                         2
PoliciesQuota                           1000
AccountMFAEnabled                       1
Providers                               2
InstanceProfilesQuota                   100
MFADevices                              4
AccessKeysPerUserQuota                  2
AttachedPoliciesPerUserQuota            2
SigningCertificatesPerUserQuota         2
PolicyVersionsInUse                     4
InstanceProfiles                        1
```

```
...
```

- Para obter detalhes sobre a API, consulte GetAccountSummary Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMAttachedGroupPolicyList`

O código de exemplo a seguir mostra como usar `Get-IAMAttachedGroupPolicyList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna os nomes e ARNs das políticas gerenciadas que estão anexadas ao grupo do IAM nomeado **`Admins`** na AWS conta. Para ver a lista de políticas em linha incorporadas no grupo, use o comando **`Get-IAMGroupPolicyList`**.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Saída:

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/SecurityAudit                      SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess                AdministratorAccess
```

- Para obter detalhes sobre a API, consulte ListAttachedGroupPolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMAttachedRolePolicyList`

O código de exemplo a seguir mostra como usar `Get-IAMAttachedRolePolicyList`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os nomes e os ARNs das políticas gerenciadas anexadas ao perfil do IAM denominado **`SecurityAuditRole`** na conta da AWS . Para ver a lista de políticas em linha incorporadas no perfil, use o comando **`Get-IAMRolePolicyList`**.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Saída:

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/SecurityAudit                      SecurityAudit
```

- Para obter detalhes sobre a API, consulte ListAttachedRolePolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMAttachedUserPolicyList`

O código de exemplo a seguir mostra como usar `Get-IAMAttachedUserPolicyList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna os nomes e ARNs das políticas gerenciadas para o usuário do IAM nomeado **`Bob`** na AWS conta. Para ver a lista de políticas em linha incorporadas no usuário do IAM, use o comando **`Get-IAMUserPolicyList`**.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Saída:

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/TesterPolicy                       TesterPolicy
```

- Para obter detalhes sobre a API, consulte ListAttachedUserPolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMContextKeysForCustomPolicy`

O código de exemplo a seguir mostra como usar `Get-IAMContextKeysForCustomPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo busca todas as chaves de contexto presentes no JSON da política fornecida. A fim de produzir várias políticas, você pode fornecer uma lista de valores separados por vírgula.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
```

```
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- Para obter detalhes sobre a API, consulte GetContextKeysForCustomPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMContextKeysForPrincipalPolicy`

O código de exemplo a seguir mostra como usar `Get-IAMContextKeysForPrincipalPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo busca todas as chaves de contexto presentes na política json fornecida e as políticas anexadas à entidade IAM (user/role etc.). Para - PolicyInputList você pode fornecer uma lista de vários valores como valores separados por vírgula.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- Para obter detalhes sobre a API, consulte GetContextKeysForPrincipalPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMCredentialReport`

O código de exemplo a seguir mostra como usar `Get-IAMCredentialReport`.

Ferramentas para PowerShell V5

Exemplo1: este exemplo abre o relatório retornado e o envia ao pipeline como uma matriz de linhas de texto. A primeira linha é o cabeçalho com nomes de colunas separados por

vírgula. Cada linha sucessiva é a linha de detalhes de um usuário, com cada campo separado por vírgulas. Antes de visualizar o relatório, você deve gerá-lo com o cmdlet **`Request-IAMCredentialReport`**. Para recuperar o relatório como uma única string, use **`-Raw`** em vez de **`-AsTextArray`**. O alias **`-SplitLines`** também é aceito no switch **`-AsTextArray`**. Para obter a lista completa de colunas na saída, consulte a referência de API do serviço. Observe que, se não usar **`-AsTextArray`** ou **`-SplitLines`**, você deve extrair o texto da propriedade **`.Content`** usando a classe **`StreamReader`** .NET.

```
Request-IAMCredentialReport
```

Saída:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Saída:

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,passw
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00+0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/
A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- Para obter detalhes sobre a API, consulte GetCredentialReport Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMEntitiesForPolicy`

O código de exemplo a seguir mostra como usar `Get-IAMEntitiesForPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna uma lista de grupos, perfis e usuários do IAM que têm a política **`arn:aws:iam::123456789012:policy/TestPolicy`** anexada.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Saída:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- Para obter detalhes sobre a API, consulte ListEntitiesForPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMGroup`

O código de exemplo a seguir mostra como usar `Get-IAMGroup`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre o grupo do IAM **`Testers`**, incluindo uma compilação de todos os usuários do IAM que pertencem ao grupo.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Saída:

```
Group                                     IsTruncated           Marker
  Users
-----                                     -----------           ------
  -----
Amazon.IdentityManagement.Model.Group     False
  {Theresa, David}
```

```
$results.Group
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- Para obter detalhes sobre a API, consulte GetGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMGroupForUser`

O código de exemplo a seguir mostra como usar `Get-IAMGroupForUser`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a lista de grupos do IAM aos quais o usuário do IAM **`David`** pertence.

```
Get-IAMGroupForUser -UserName David
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- Para obter detalhes sobre a API, consulte ListGroupsForUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMGroupList`

O código de exemplo a seguir mostra como usar `Get-IAMGroupList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna uma coleção de todos os grupos do IAM definidos no atual Conta da AWS.

```
Get-IAMGroupList
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
```

```
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- Para obter detalhes sobre a API, consulte ListGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMGroupPolicy`

O código de exemplo a seguir mostra como usar `Get-IAMGroupPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre a política em linha incorporada denominada **`PowerUserAccess-Testers`** do grupo **`Testers`**. A propriedade **`PolicyDocument`** é codificada em URL. Ela é decodificada neste exemplo com o método .NET **`UrlDecode`**.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

Saída:

```
GroupName     PolicyDocument                                                PolicyName
---------     --------------                                                ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- Para obter detalhes sobre a API, consulte GetGroupPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMGroupPolicyList`

O código de exemplo a seguir mostra como usar `Get-IAMGroupPolicyList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna uma lista das políticas em linha incorporadas no grupo **`Testers`**. Para obter as políticas gerenciadas anexadas ao grupo, use o comando **`Get-IAMAttachedGroupPolicyList`**.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Saída:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Para obter detalhes sobre a API, consulte ListGroupPolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMInstanceProfile`

O código de exemplo a seguir mostra como usar `Get-IAMInstanceProfile`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna detalhes do perfil da instância nomeado **`ec2instancerole`** que está definido na AWS conta atual.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Para obter detalhes sobre a API, consulte GetInstanceProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMInstanceProfileForRole`

O código de exemplo a seguir mostra como usar `Get-IAMInstanceProfileForRole`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes do perfil de instância associado ao perfil **`ec2instancerole`**.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Saída:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- Para obter detalhes sobre a API, consulte ListInstanceProfilesForRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMInstanceProfileList`

O código de exemplo a seguir mostra como usar `Get-IAMInstanceProfileList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna uma coleção dos perfis de instância definidos no atual Conta da AWS.

```
Get-IAMInstanceProfileList
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Para obter detalhes sobre a API, consulte ListInstanceProfiles Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMLoginProfile`

O código de exemplo a seguir mostra como usar `Get-IAMLoginProfile`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a data de criação da senha e se uma redefinição de senha é necessária para o usuário do IAM **`David`**.

```
Get-IAMLoginProfile -UserName David
```

Saída:

```
CreateDate                  PasswordResetRequired                UserName
----------                  ---------------------                --------
12/10/2014 3:39:44 PM       False                                David
```

- Para obter detalhes sobre a API, consulte GetLoginProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMMFADevice`

O código de exemplo a seguir mostra como usar `Get-IAMMFADevice`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre o dispositivo de MFA atribuído ao usuário do IAM **`David`**. Neste exemplo, você percebe que é um dispositivo virtual porque o **`SerialNumber`** é um ARN em vez do número de série real de um dispositivo físico.

```
Get-IAMMFADevice -UserName David
```

Saída:

```
EnableDate                  SerialNumber                          UserName
----------                  ------------                          --------
4/8/2015 9:41:10 AM         arn:aws:iam::123456789012:mfa/David    David
```

- Para obter detalhes sobre a API, consulte ListMfaDevices Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMOpenIDConnectProvider`

O código de exemplo a seguir mostra como usar `Get-IAMOpenIDConnectProvider`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre o provedor OpenID Connect cujo ARN é **`arn:aws:iam::123456789012:oidc-provider/accounts.google.com`**. A propriedade **`ClientIDList`** é uma compilação que contém todos os IDs de cliente definidos para esse provedor.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Saída:

```
ClientIDList         CreateDate                ThumbprintList
        Url
------------         ----------                --------------
        ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- Para obter detalhes sobre a API, consulte GetOpenIdConnectProvider Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMOpenIDConnectProviderList`

O código de exemplo a seguir mostra como usar `Get-IAMOpenIDConnectProviderList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna uma lista de ARNS de todos os provedores OpenID Connect definidos na Conta da AWS atual.

```
Get-IAMOpenIDConnectProviderList
```

Saída:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Para obter detalhes sobre a API, consulte ListOpenIdConnectProviders Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMPolicy`

O código de exemplo a seguir mostra como usar `Get-IAMPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna detalhes sobre a política gerenciada cujo ARN é **`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Saída:

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- Para obter detalhes sobre a API, consulte GetPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMPolicyList`

O código de exemplo a seguir mostra como usar `Get-IAMPolicyList`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo retorna uma coleção das três primeiras políticas gerenciadas disponíveis na AWS conta corrente. Como não **`-scope`** está especificado, ele usa como padrão **`all`** e inclui políticas AWS gerenciadas e gerenciadas pelo cliente.

```
Get-IAMPolicyList -MaxItem 3
```

Saída:

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
```

```
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

Exemplo 2: Esse exemplo retorna uma coleção das duas primeiras políticas gerenciadas pelo cliente disponíveis na AWS conta corrente. Ele usa **`-Scope local`** para limitar a saída somente às políticas gerenciadas pelo cliente.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Saída:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM
```

```
Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- Para obter detalhes sobre a API, consulte ListPolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMPolicyVersion`

O código de exemplo a seguir mostra como usar `Get-IAMPolicyVersion`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o documento da política na versão **`v2`** da política cujo ARN é **`arn:aws:iam::123456789012:policy/MyManagedPolicy`**. O documento da política na propriedade **`Document`** é codificado em URL, sendo decodificado neste exemplo com o método .NET **`UrlDecode`**.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Saída:

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
        v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
```

```
$policy
{
  "Version":"2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- Para obter detalhes sobre a API, consulte GetPolicyVersion Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMPolicyVersionList`

O código de exemplo a seguir mostra como usar `Get-IAMPolicyVersionList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a lista de versões disponíveis da política cujo ARN é **`arn:aws:iam::123456789012:policy/MyManagedPolicy`**. Para obter o documento de política de uma versão específica, use o comando **`Get-IAMPolicyVersion`** e especifique o **`VersionId`** do que você deseja.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

Saída:

```
CreateDate                  Document                IsDefaultVersion
    VersionId
----------                  --------                ----------------
    ---------
2/12/2015 9:39:53 AM                                True
    v2
2/12/2015 9:39:09 AM                                False
    v1
```

- Para obter detalhes sobre a API, consulte ListPolicyVersions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-IAMRole

O código de exemplo a seguir mostra como usar `Get-IAMRole`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna os detalhes do **lamda_exec_role**. Ele inclui o documento da política de confiança que especifica quem pode assumir esse perfil. O documento da política é codificado em URL e pode ser decodificado usando o método .NET **UrlDecode**. Neste exemplo, todos os espaços em branco da política original foram removidos antes de ela ser carregada na política. Para ver os documentos da política de permissões que determinam o que alguém que assume o perfil pode fazer, use **Get-IAMRolePolicy** para políticas em linha e **Get-IAMPolicyVersion** para políticas gerenciadas anexadas.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Saída:

```
Arn                      : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A
%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate               : 4/2/2015 9:16:11 AM
Path                     : /
RoleId                   : 2YBIKAIBHNKB4EXAMPLE1
RoleName                 : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Saída:

```
{"Version":"2012-10-17",       "Statement":[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Para obter detalhes sobre a API, consulte GetRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMRoleList`

O código de exemplo a seguir mostra como usar `Get-IAMRoleList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera uma lista de todos os perfis do IAM na Conta da AWS.

```
Get-IAMRoleList
```

- Para obter detalhes sobre a API, consulte ListRoles Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMRolePolicy`

O código de exemplo a seguir mostra como usar `Get-IAMRolePolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna o documento da política de permissões da política denominada **`oneClick_lambda_exec_role_policy`** incorporada no perfil do IAM **`lamda_exec_role`**. O documento resultante da política é codificado em URL. Ela é decodificada neste exemplo com o método .NET **`UrlDecode`**.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Saída:

```
PolicyDocument                                          PolicyName
           UserName
```

```
--------------                                                  ----------
           --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

Saída:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- Para obter detalhes sobre a API, consulte GetRolePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMRolePolicyList`

O código de exemplo a seguir mostra como usar `Get-IAMRolePolicyList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a lista de nomes de políticas em linha incorporadas no perfil do IAM **`lamda_exec_role`**. Para ver os detalhes de uma política em linha, use o comando **`Get-IAMRolePolicy`**.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Saída:

```
oneClick_lambda_exec_role_policy
```

- Para obter detalhes sobre a API, consulte ListRolePolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMRoleTagList`

O código de exemplo a seguir mostra como usar `Get-IAMRoleTagList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo busca a tag associada ao perfil.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Para obter detalhes sobre a API, consulte ListRoleTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMSAMLProvider`

O código de exemplo a seguir mostra como usar `Get-IAMSAMLProvider`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo recupera os detalhes sobre o provedor SAML 2.0 cujo ARM é arn:aws:iam: :123456789012:saml-. provider/SAMLADFS A resposta inclui o documento de metadados que você obteve do provedor de identidade para criar a entidade do provedor AWS SAML, bem como as datas de criação e expiração.

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Saída:

```
CreateDate                 SAMLMetadataDocument
     ValidUntil
----------                 --------------------
     ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-example1...
    12/23/2114 12:16:54 PM
```

- Para obter detalhes sobre a API, consulte GetSamlProvider Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMSAMLProviderList`

O código de exemplo a seguir mostra como usar `Get-IAMSAMLProviderList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a lista de provedores SAML 2.0 criados na Conta da AWS atual. Ele retorna o ARN, a data de criação e a data de expiração de cada provedor SAML.

```
Get-IAMSAMLProviderList
```

Saída:

```
Arn                                                CreateDate
 ValidUntil
---                                                ----------
 ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
 12/23/2114 12:16:54 PM
```

- Para obter detalhes sobre a API, consulte ListSAMLProviders na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMServerCertificate`

O código de exemplo a seguir mostra como usar `Get-IAMServerCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera detalhes sobre o certificado do servidor denominado **`MyServerCertificate`**. Você pode encontrar os detalhes do certificado nas propriedades **`CertificateBody`** e **`ServerCertificateMetadata`**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Saída:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                            YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                            21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
```

```
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes sobre a API, consulte GetServerCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMServerCertificateList`

O código de exemplo a seguir mostra como usar `Get-IAMServerCertificateList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a lista de certificados de servidor enviados à Conta da AWS atual.

```
Get-IAMServerCertificateList
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
```

```
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes sobre a API, consulte ListServerCertificates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMServiceLastAccessedDetail`

O código de exemplo a seguir mostra como usar `Get-IAMServiceLastAccessedDetail`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo fornece detalhes do último serviço acessado pela entidade do IAM (usuário, grupo, perfil ou política) associada na chamada de solicitação.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Saída:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Para obter detalhes sobre a API, consulte GetServiceLastAccessedDetails Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMServiceLastAccessedDetailWithEntity`

O código de exemplo a seguir mostra como usar `Get-IAMServiceLastAccessedDetailWithEntity`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo fornece o carimbo de data e hora do último acesso do serviço na solicitação pela respectiva entidade do IAM.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Saída:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

Saída:

```
EntityInfo                                LastAuthenticated
----------                                -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Saída:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Para obter detalhes sobre a API, consulte GetServiceLastAccessedDetailsWithEntities Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMSigningCertificate`

O código de exemplo a seguir mostra como usar `Get-IAMSigningCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera detalhes sobre o certificado de assinatura associado ao usuário chamado **`Bob`**.

```
Get-IAMSigningCertificate -UserName Bob
```

Saída:

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                  FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Para obter detalhes sobre a API, consulte ListSigningCertificates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMUser`

O código de exemplo a seguir mostra como usar `Get-IAMUser`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera detalhes sobre o usuário chamado **`David`**.

```
Get-IAMUser -UserName David
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/David
```

```
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

Exemplo 2: este exemplo recupera detalhes sobre o usuário do IAM atualmente conectado.

```
Get-IAMUser
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- Para obter detalhes sobre a API, consulte GetUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMUserList`

O código de exemplo a seguir mostra como usar `Get-IAMUserList`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo recupera uma coleção de usuários no atual Conta da AWS.

```
Get-IAMUserList
```

Saída:

```
      Arn              : arn:aws:iam::123456789012:user/Administrator
      CreateDate       : 10/16/2014 9:03:09 AM
      PasswordLastUsed : 3/4/2015 12:12:33 PM
      Path             : /
      UserId           : 7K3GJEANSKZF2EXAMPLE1
      UserName         : Administrator
```

```
    Arn              : arn:aws:iam::123456789012:user/Bob
    CreateDate       : 4/6/2015 12:54:42 PM
    PasswordLastUsed : 1/1/0001 12:00:00 AM
    Path             : /
    UserId           : L3EWNONDOM3YUEXAMPLE2
    UserName         : bab

    Arn              : arn:aws:iam::123456789012:user/David
    CreateDate       : 12/10/2014 3:39:27 PM
    PasswordLastUsed : 3/19/2015 8:44:04 AM
    Path             : /
    UserId           : Y4FKWQCXTA52QEXAMPLE3
    UserName         : David
```

- Para obter detalhes sobre a API, consulte ListUsers Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMUserPolicy`

O código de exemplo a seguir mostra como usar `Get-IAMUserPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera os detalhes da política em linha denominada **`Davids_IAM_Admin_Policy`** incorporada no usuário do IAM chamado **`David`**. O documento de política é codificado em URL.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Saída:

```
PolicyDocument                                              PolicyName
    UserName
--------------                                              ----------
    --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
    David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- Para obter detalhes sobre a API, consulte GetUserPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMUserPolicyList`

O código de exemplo a seguir mostra como usar `Get-IAMUserPolicyList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a lista de nomes das políticas em linha incorporadas no usuário do IAM chamado **`David`**.

```
Get-IAMUserPolicyList -UserName David
```

Saída:

```
Davids_IAM_Admin_Policy
```

- Para obter detalhes sobre a API, consulte ListUserPolicies Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMUserTagList`

O código de exemplo a seguir mostra como usar `Get-IAMUserTagList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo busca a tag associada ao usuário.

```
Get-IAMUserTagList -UserName joe
```

- Para obter detalhes sobre a API, consulte ListUserTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-IAMVirtualMFADevice`

O código de exemplo a seguir mostra como usar `Get-IAMVirtualMFADevice`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo recupera uma coleção de dispositivos virtuais de MFA atribuídos aos usuários na AWS conta. A propriedade do **`User`** de cada um é um objeto com detalhes do usuário do IAM ao qual o dispositivo está atribuído.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Saída:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- Para obter detalhes sobre a API, consulte ListVirtualMfaDevices Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-IAMAccessKey`

O código de exemplo a seguir mostra como usar `New-IAMAccessKey`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma chave de acesso e um par de chaves de acesso secreto e os atribui ao usuário **`David`**. Certifique-se de salvar os valores **`AccessKeyId`** e **`SecretAccessKey`** em um arquivo, pois este é o único momento em que você pode obter a **`SecretAccessKey`**. Não será possível recuperá-la depois. Caso perca a chave secreta, você deve criar um par de chaves de acesso.

```
New-IAMAccessKey -UserName David
```

Saída:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- Para obter detalhes sobre a API, consulte CreateAccessKey Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-IAMAccountAlias`

O código de exemplo a seguir mostra como usar `New-IAMAccountAlias`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo altera o alias da AWS conta para**`mycompanyaws`**. <accountidnumber>O endereço da página de login do usuário muda para o URL original usando https://mycompanyaws.signin.aws.amazon.com/console. o número de ID da sua conta em vez do alias (https://.signin.aws.amazon). com/console) continua funcionando. No entanto, todos os URLs baseados em alias definidos anteriormente param de funcionar.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obter detalhes sobre a API, consulte CreateAccountAlias Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMGroup

O código de exemplo a seguir mostra como usar `New-IAMGroup`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um grupo do IAM denominado **`Developers`**.

```
New-IAMGroup -GroupName Developers
```

Saída:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- Para obter detalhes sobre a API, consulte CreateGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMInstanceProfile

O código de exemplo a seguir mostra como usar `New-IAMInstanceProfile`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um perfil de instância do IAM denominado **`ProfileForDevEC2Instance`**. Você deve executar o comando **`Add-IAMRoleToInstanceProfile`** separadamente para associar o perfil de instância a um perfil do IAM existente que fornece permissões à instância. Por fim, anexe o perfil de instância a uma instância do EC2 ao executá-la. Para isso, use o cmdlet **`New-EC2Instance`** com o parâmetro **`InstanceProfile_Arn`** ou **`InstanceProfile_Name`**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Saída:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
```

```
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- Para obter detalhes sobre a API, consulte CreateInstanceProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMLoginProfile

O código de exemplo a seguir mostra como usar `New-IAMLoginProfile`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma senha (temporária) para o usuário do IAM chamado Bob e define a sinalização que exige que o usuário altere a senha na próxima vez que **Bob** fizer login.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Saída:

```
CreateDate                    PasswordResetRequired                UserName
----------                    ---------------------                --------
4/14/2015 12:26:30 PM         True                                 Bob
```

- Para obter detalhes sobre a API, consulte CreateLoginProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMOpenIDConnectProvider

O código de exemplo a seguir mostra como usar `New-IAMOpenIDConnectProvider`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um provedor OIDC do IAM associado ao serviço do provedor compatível com OIDC encontrado no URL **`https://example.oidcprovider.com`** e no ID do cliente **`my-testapp-1`**. O provedor OIDC fornece a impressão digital. Para autenticar a impressão digital, siga as etapas em http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-
testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Saída:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para obter detalhes sobre a API, consulte CreateOpenIdConnectProvider Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-IAMPolicy`

O código de exemplo a seguir mostra como usar `New-IAMPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo cria uma nova política do IAM na AWS conta atual chamada **`MySamplePolicy`** O arquivo **`MySamplePolicy.json`** fornece o conteúdo da política. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o arquivo de política JSON.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Saída:

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- Para obter detalhes sobre a API, consulte CreatePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMPolicyVersion

O código de exemplo a seguir mostra como usar `New-IAMPolicyVersion`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma versão "v2" da política do IAM cujo ARN é **`arn:aws:iam::123456789012:policy/MyPolicy`** e a torna a versão padrão. O arquivo **`NewPolicyVersion.json`** fornece o conteúdo da política. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o arquivo de política JSON.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Saída:

```
CreateDate                          Document                  IsDefaultVersion
        VersionId
----------                          --------                  ----------------
        ---------
4/15/2015 10:54:54 AM                                         True
        v2
```

- Para obter detalhes sobre a API, consulte CreatePolicyVersion Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMRole

O código de exemplo a seguir mostra como usar `New-IAMRole`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um perfil denominado **`MyNewRole`** e anexa a ele a política encontrada no arquivo **`NewRoleTrustPolicy.json`**. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o arquivo de política JSON. O documento de política exibido na saída é codificado em URL. Ele é decodificado nesse exemplo com o método .NET **`UrlDecode`**.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
```

```
$results
```

Saída:

```
Arn                      : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0D
%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C
%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                           %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D%0A
%20%20%20%20%7D%0D%0A%20
                           %20%5D%0D%0A%7D
CreateDate               : 4/15/2015 11:04:23 AM
Path                     : /
RoleId                   : V5PAJI2KPN4EAEXAMPLE1
RoleName                 : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Para obter detalhes sobre a API, consulte CreateRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMSAMLProvider

O código de exemplo a seguir mostra como usar `New-IAMSAMLProvider`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma entidade provedora SAML no IAM. Ele é denominado **`MySAMLProvider`** e descrito pelo documento de metadados SAML encontrado no arquivo **`SAMLMetaData.xml`**, que foi baixado separadamente do site do provedor de serviços SAML.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Saída:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Para obter detalhes sobre a API, consulte CreateSAMLProvider na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMServiceLinkedRole

O código de exemplo a seguir mostra como usar `New-IAMServiceLinkedRole`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um perfil vinculado ao serviço para o serviço de ajuste de escala automático.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- Para obter detalhes sobre a API, consulte CreateServiceLinkedRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMUser

O código de exemplo a seguir mostra como usar `New-IAMUser`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um usuário do IAM chamado **Bob**. Se Bob precisar entrar no AWS console, você deverá executar o comando separadamente **New-IAMLoginProfile** para criar um perfil de login com uma senha. Se Bob precisar executar AWS PowerShell comandos CLI multiplataforma ou fazer chamadas de AWS API, você deverá executar o **New-IAMAccessKey** comando separadamente para criar chaves de acesso.

```
New-IAMUser -UserName Bob
```

Saída:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- Para obter detalhes sobre a API, consulte CreateUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-IAMVirtualMFADevice

O código de exemplo a seguir mostra como usar `New-IAMVirtualMFADevice`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um dispositivo de MFA virtual. As linhas 2 e 3 extraem o valor de **Base32StringSeed** de que o programa de software de MFA virtual precisa para criar uma conta (como alternativa ao código QR). Depois de configurar o programa com o valor, obtenha dois códigos de autenticação sequencial do programa. Por fim, use o último comando para vincular o dispositivo de MFA virtual ao usuário do IAM **Bob** e sincronizar a conta com os dois códigos de autenticação.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Saída:

```
-- Pause here to enter base-32 string seed code into virtual MFA program to register
 account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

Exemplo 2: este exemplo cria um dispositivo de MFA virtual. As linhas 2 e 3 extraem o valor de **`QRCodePNG`** e o gravam em um arquivo. Essa imagem pode ser digitalizada pelo programa de software virtual MFA para criar uma conta (como alternativa à inserção manual do Base32StringSeed valor). Depois de criar a conta no programa de MFA virtual, obtenha dois códigos de autenticação sequencial e insira-os nos últimos comandos para vincular o dispositivo MFA virtual ao usuário do IAM **`Bob`** e sincronizar a conta.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path QRCode.png
```

Saída:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Para obter detalhes sobre a API, consulte CreateVirtualMfaDevice Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Publish-IAMServerCertificate`

O código de exemplo a seguir mostra como usar `Publish-IAMServerCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo faz upload de um novo certificado de servidor na conta do IAM. Os arquivos contendo o corpo do certificado, a chave privada e (opcionalmente) a cadeia de certificação devem ser codificados em PEM. Observe que os parâmetros exigem o conteúdo real dos arquivos em vez dos nomes deles. Você deve usar o parâmetro switch **`-Raw`** para processar com êxito o conteúdo do arquivo.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Saída:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- Para obter detalhes sobre a API, consulte UploadServerCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Publish-IAMSigningCertificate`

O código de exemplo a seguir mostra como usar `Publish-IAMSigningCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo carrega um novo certificado de X.509 assinatura e o associa ao usuário IAM chamado. **`Bob`** O arquivo que contém o corpo do certificado é codificado em PEM. O parâmetro **`CertificateBody`** exige o conteúdo real do arquivo de certificado em vez do nome do arquivo. Você deve usar o parâmetro switch **`-Raw`** para processar o arquivo com êxito.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Saída:

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
```

```
                    rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                    Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                    nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                    FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                    NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                    -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Para obter detalhes sobre a API, consulte UploadSigningCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-IAMGroupPolicy`

O código de exemplo a seguir mostra como usar `Register-IAMGroupPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo anexa a política gerenciada pelo cliente denominada **`TesterPolicy`** ao grupo do IAM **`Testers`**. Os usuários desse grupo são imediatamente afetados pelas permissões definidas na versão padrão dessa política.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Exemplo 2: Esse exemplo anexa a política AWS gerenciada nomeada **`AdministratorAccess`** ao grupo **`Admins`** IAM. Os usuários desse grupo são imediatamente afetados pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Para obter detalhes sobre a API, consulte AttachGroupPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-IAMRolePolicy`

O código de exemplo a seguir mostra como usar `Register-IAMRolePolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo anexa a política AWS gerenciada nomeada **`SecurityAudit`** à função **`CoSecurityAuditors`** do IAM. Os usuários que assumem esse perfil são imediatamente afetados pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- Para obter detalhes sobre a API, consulte AttachRolePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-IAMUserPolicy`

O código de exemplo a seguir mostra como usar `Register-IAMUserPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo anexa a política AWS gerenciada nomeada **`AmazonCognitoPowerUser`** ao usuário **`Bob`** do IAM. O usuário é imediatamente afetado pelas permissões definidas na versão mais recente dessa política.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- Para obter detalhes sobre a API, consulte AttachUserPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMAccessKey`

O código de exemplo a seguir mostra como usar `Remove-IAMAccessKey`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo exclui o par de chaves de AWS acesso com o ID **`AKIAIOSFODNN7EXAMPLE`** da chave do usuário nomeado**`Bob`**.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Para obter detalhes sobre a API, consulte DeleteAccessKey Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMAccountAlias`

O código de exemplo a seguir mostra como usar `Remove-IAMAccountAlias`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove o alias da conta do seu Conta da AWS. A página de login do usuário com o alias em https://mycompanyaws.signin.aws.amazon.com/console não funciona mais. Em vez disso, você deve usar o URL original com seu número de Conta da AWS identificação em https://<accountidnumber>.signin.aws.amazon. com/console.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Para obter detalhes sobre a API, consulte DeleteAccountAlias Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `Remove-IAMAccountPasswordPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo exclui a política de senha do Conta da AWS e redefine todos os valores para seus padrões originais. Se uma política de senha não existir atualmente, a seguinte mensagem de erro será exibida: A política de conta com nome PasswordPolicy não pode ser encontrada.

```
Remove-IAMAccountPasswordPolicy
```

- Para obter detalhes sobre a API, consulte DeleteAccountPasswordPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMClientIDFromOpenIDConnectProvider`

O código de exemplo a seguir mostra como usar `Remove-IAMClientIDFromOpenIDConnectProvider`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove o ID do cliente **My-TestApp-3** da lista de IDs de cliente associados ao provedor OIDC do IAM cujo ARN é **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- Para obter detalhes sobre a API, consulte RemoveClientIdFromOpenIdConnectProvider Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Remove-IAMGroup

O código de exemplo a seguir mostra como usar `Remove-IAMGroup`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o grupo do IAM denominado **MyTestGroup**. O primeiro comando remove todos os usuários do IAM que são membros do grupo, e o segundo exclui o grupo do IAM. Ambos os comandos funcionam sem nenhuma solicitação de confirmação.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Para obter detalhes sobre a API, consulte DeleteGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Remove-IAMGroupPolicy

O código de exemplo a seguir mostra como usar `Remove-IAMGroupPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove a política em linha denominada **TesterPolicy** do grupo do IAM **Testers**. Os usuários desse grupo perdem imediatamente as permissões definidas nessa política.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Para obter detalhes sobre a API, consulte DeleteGroupPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMInstanceProfile`

O código de exemplo a seguir mostra como usar `Remove-IAMInstanceProfile`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o perfil de instância do EC2 denominado **`MyAppInstanceProfile`**. O primeiro comando desassocia todos os perfis do perfil de instância e, em seguida, o segundo comando exclui o perfil de instância.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-
IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Para obter detalhes sobre a API, consulte DeleteInstanceProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMLoginProfile`

O código de exemplo a seguir mostra como usar `Remove-IAMLoginProfile`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o perfil de login do usuário do IAM denominado **`Bob`**. Isso impede que o usuário faça login no AWS console. Isso não impede que o usuário execute qualquer AWS chamada de CLI ou API usando chaves de AWS acesso que ainda possam estar anexadas à conta do usuário. PowerShell

```
Remove-IAMLoginProfile -UserName Bob
```

- Para obter detalhes sobre a API, consulte DeleteLoginProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMOpenIDConnectProvider`

O código de exemplo a seguir mostra como usar `Remove-IAMOpenIDConnectProvider`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o provedor OIDC do IAM que se conecta ao provedor **`example.oidcprovider.com`**. Certifique-se de atualizar ou excluir quaisquer perfis que façam referência a esse provedor no elemento **`Principal`** da política de confiança do perfil.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Para obter detalhes sobre a API, consulte DeleteOpenIdConnectProvider Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMPolicy`

O código de exemplo a seguir mostra como usar `Remove-IAMPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui a política cujo ARN é **`arn:aws:iam::123456789012:policy/MySamplePolicy`**. Antes de excluir a política, exclua primeiro todas as versões, exceto a padrão, executando **`Remove-IAMPolicyVersion`**. Você também deve desassociar a política de qualquer usuário, grupo ou perfil do IAM.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Exemplo 2: este exemplo exclui uma política excluindo primeiro todas as versões não padrão da política, desassociando-a de todas as entidades do IAM anexadas e, por fim, excluindo a própria política. A primeira linha recupera o objeto da política. A segunda linha recupera todas as versões da política que não estão marcadas como a versão padrão em uma compilação e depois exclui cada política na compilação. A terceira linha recupera todos os usuários, grupos e perfis do IAM aos quais a política está anexada. As linhas de quatro a seis desassociam a política de cada entidade anexada. A última linha usa esse comando para remover a política gerenciada e a versão padrão restante. O exemplo inclui o parâmetro switch **`-Force`** em qualquer linha que precise dele para suprimir solicitações de confirmação.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

```
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- Para obter detalhes sobre a API, consulte DeletePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMPolicyVersion`

O código de exemplo a seguir mostra como usar `Remove-IAMPolicyVersion`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui a versão identificada como **`v2`** da política cujo ARN é **`arn:aws:iam::123456789012:policy/MySamplePolicy`**.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

Exemplo 2: este exemplo exclui uma política excluindo primeiro todas as versões não padrão da política e depois excluindo a própria política. A primeira linha recupera o objeto da política. A segunda linha recupera todas as versões da política que não estão marcadas como padrão em uma compilação e, em seguida, usa esse comando para excluir cada política na compilação. A última linha remove a política em si, bem como a versão padrão restante. Observe que, para excluir com êxito uma política gerenciada, você também deve desassociar a política de qualquer usuário, grupo ou perfis usando os comandos **`Unregister-IAMUserPolicy`**, **`Unregister-IAMGroupPolicy`** e **`Unregister-IAMRolePolicy`**. Veja o exemplo do cmdlet **`Remove-IAMPolicy`**.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- Para obter detalhes sobre a API, consulte DeletePolicyVersion Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMRole`

O código de exemplo a seguir mostra como usar `Remove-IAMRole`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o perfil denominado **`MyNewRole`** da conta atual do IAM. Antes de excluir o perfil, use primeiro o comando **`Unregister-IAMRolePolicy`** para desassociar todas as políticas gerenciadas. As políticas em linha são excluídas com o perfil.

```
Remove-IAMRole -RoleName MyNewRole
```

Exemplo 2: este exemplo desassocia todas as políticas gerenciadas do perfil denominado **`MyNewRole`** e depois o exclui. A primeira linha recupera todas as políticas gerenciadas anexadas ao perfil como uma compilação e, em seguida, desassocia cada política da compilação do perfil. A segunda linha exclui o perfil em si. As políticas em linha são excluídas com o perfil.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- Para obter detalhes sobre a API, consulte DeleteRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMRoleFromInstanceProfile`

O código de exemplo a seguir mostra como usar `Remove-IAMRoleFromInstanceProfile`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o perfil denominado **`MyNewRole`** do perfil de instância do EC2 denominado **`MyNewRole`**. Um perfil de instância criado no console do IAM sempre tem o mesmo nome do perfil, como neste exemplo. Se você os criar na API ou na CLI, eles poderão ter nomes diferentes.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole
 -Force
```

- Para obter detalhes sobre a API, consulte RemoveRoleFromInstanceProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMRolePermissionsBoundary`

O código de exemplo a seguir mostra como usar `Remove-IAMRolePermissionsBoundary`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra como remover o limite de permissões anexado a um perfil do IAM.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Para obter detalhes sobre a API, consulte DeleteRolePermissionsBoundary Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMRolePolicy`

O código de exemplo a seguir mostra como usar `Remove-IAMRolePolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui a política em linha **`S3AccessPolicy`** incorporada no perfil do IAM **`S3BackupRole`**.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Para obter detalhes sobre a API, consulte DeleteRolePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMRoleTag`

O código de exemplo a seguir mostra como usar `Remove-IAMRoleTag`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove a tag da função chamada "MyRoleName" com a chave de tag como “abac”. Para remover várias tags, forneça uma lista de chaves de tag separadas por vírgulas.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Para obter detalhes sobre a API, consulte UntagRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMSAMLProvider`

O código de exemplo a seguir mostra como usar `Remove-IAMSAMLProvider`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o provedor SAML 2.0 do IAM cujo ARN é **`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`**.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- Para obter detalhes sobre a API, consulte DeleteSAMLProvider na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMServerCertificate`

O código de exemplo a seguir mostra como usar `Remove-IAMServerCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o certificado do servidor denominado **`MyServerCert`**.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Para obter detalhes sobre a API, consulte DeleteServerCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMServiceLinkedRole`

O código de exemplo a seguir mostra como usar `Remove-IAMServiceLinkedRole`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo excluiu o perfil vinculado ao serviço. Observe que, se o serviço ainda estiver usando esse perfil, esse comando resultará em uma falha.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Para obter detalhes sobre a API, consulte DeleteServiceLinkedRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMSigningCertificate`

O código de exemplo a seguir mostra como usar `Remove-IAMSigningCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o certificado de assinatura com o ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** do usuário do IAM chamado **`Bob`**.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Para obter detalhes sobre a API, consulte DeleteSigningCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMUser`

O código de exemplo a seguir mostra como usar `Remove-IAMUser`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o usuário do IAM chamado **`Bob`**.

```
Remove-IAMUser -UserName Bob
```

Exemplo 2: este exemplo exclui a usuária do IAM chamada **`Theresa`** com todos os elementos que devem ser excluídos primeiro.

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName $name -Force }
```

```
# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn $pol.PolicyArn
 -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- Para obter detalhes sobre a API, consulte DeleteUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMUserFromGroup`

O código de exemplo a seguir mostra como usar `Remove-IAMUserFromGroup`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove o usuário do IAM **Bob** do grupo **Testers**.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Exemplo 2: este exemplo encontra todos os grupos dos quais a usuária do IAM **Theresa** é membro e, em seguida, remove a **Theresa** desses grupos.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

Exemplo 3: este exemplo mostra uma forma alternativa de remover o usuário do IAM **Bob** do grupo **Testers**.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
 Testers -Force
```

- Para obter detalhes sobre a API, consulte RemoveUserFromGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMUserPermissionsBoundary`

O código de exemplo a seguir mostra como usar `Remove-IAMUserPermissionsBoundary`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra como remover o limite de permissões anexado a um usuário do IAM.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Para obter detalhes sobre a API, consulte DeleteUserPermissionsBoundary Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMUserPolicy`

O código de exemplo a seguir mostra como usar `Remove-IAMUserPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui a política em linha denominada **`AccessToEC2Policy`** incorporada no usuário do IAM chamado **`Bob`**.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Exemplo 2: este exemplo encontra todas as políticas em linha incorporadas na usuária do IAM chamada **`Theresa`** e depois as exclui.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- Para obter detalhes sobre a API, consulte DeleteUserPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMUserTag`

O código de exemplo a seguir mostra como usar `Remove-IAMUserTag`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove a tag do usuário denominado “joe” com a chave de tag como “abac” e “xyzw”. Para remover várias tags, forneça uma lista de chaves de tag separadas por vírgulas.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Para obter detalhes sobre a API, consulte UntagUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-IAMVirtualMFADevice`

O código de exemplo a seguir mostra como usar `Remove-IAMVirtualMFADevice`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o dispositivo de MFA virtual do IAM cujo ARN é **`arn:aws:iam::123456789012:mfa/bob`**.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Exemplo 2: este exemplo verifica se a usuária do IAM Theresa tem um dispositivo de MFA atribuído. Se for encontrado, o dispositivo é desabilitado para a usuária do IAM. Se o dispositivo for virtual, ele também é excluído.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

- Para obter detalhes sobre a API, consulte DeleteVirtualMfaDevice Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Request-IAMCredentialReport`

O código de exemplo a seguir mostra como usar `Request-IAMCredentialReport`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo solicita a geração de um novo relatório, que pode ser feito a cada quatro horas. Se o último relatório ainda for recente, o campo Estado será **`COMPLETE`**. Use **`Get-IAMCredentialReport`** para visualizar o relatório completo.

```
Request-IAMCredentialReport
```

Saída:

```
Description                                                  State
-----------                                                  -----
No report exists. Starting a new report generation task      STARTED
```

- Para obter detalhes sobre a API, consulte GenerateCredentialReport Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Request-IAMServiceLastAccessedDetail`

O código de exemplo a seguir mostra como usar `Request-IAMServiceLastAccessedDetail`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo é um cmdlet equivalente da GenerateServiceLastAccessedDetails API. Isso fornece um ID de trabalho que pode ser usado em Get-IAMServiceLastAccessedDetail e Get-IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Para obter detalhes sobre a API, consulte GenerateServiceLastAccessedDetails Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-IAMDefaultPolicyVersion`

O código de exemplo a seguir mostra como usar `Set-IAMDefaultPolicyVersion`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo define a versão **`v2`** da política cujo ARN é **`arn:aws:iam::123456789012:policy/MyPolicy`** como versão ativa padrão.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
VersionId v2
```

- Para obter detalhes sobre a API, consulte SetDefaultPolicyVersion Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-IAMRolePermissionsBoundary`

O código de exemplo a seguir mostra como usar `Set-IAMRolePermissionsBoundary`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra como definir o limite de permissões de um perfil do IAM. Você pode definir políticas AWS gerenciadas ou políticas personalizadas como limite de permissão.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Para obter detalhes sobre a API, consulte PutRolePermissionsBoundary Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-IAMUserPermissionsBoundary`

O código de exemplo a seguir mostra como usar `Set-IAMUserPermissionsBoundary`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra como definir o limite de permissões do usuário. Você pode definir políticas AWS gerenciadas ou políticas personalizadas como limite de permissão.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Para obter detalhes sobre a API, consulte PutUserPermissionsBoundary Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Sync-IAMMFADevice`

O código de exemplo a seguir mostra como usar `Sync-IAMMFADevice`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo sincroniza o dispositivo de MFA associado ao usuário do IAM **`Bob`** e cujo ARN é **`arn:aws:iam::123456789012:mfa/bob`** com um programa autenticador que forneceu os dois códigos de autenticação.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Exemplo 2: este exemplo sincroniza o dispositivo de MFA do IAM associado à usuária do IAM **`Theresa`** com um dispositivo físico que tem o número de série **`ABCD12345678`** e que forneceu os dois códigos de autenticação.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- Para obter detalhes sobre a API, consulte ResyncMfaDevice Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-IAMGroupPolicy`

O código de exemplo a seguir mostra como usar `Unregister-IAMGroupPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo desassocia a política de grupo gerenciado cujo ARN é **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** do grupo denominado **`Testers`**.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao grupo denominado **`Testers`** e as desassocia do grupo.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- Para obter detalhes sobre a API, consulte DetachGroupPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-IAMRolePolicy`

O código de exemplo a seguir mostra como usar `Unregister-IAMRolePolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo desassocia a política de grupo gerenciado cujo ARN é **`arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy`** do perfil denominado **`FedTesterRole`**.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao perfil denominado **`FedTesterRole`** e as desassocia dele.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -
Rolename FedTesterRole
```

- Para obter detalhes sobre a API, consulte DetachRolePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-IAMUserPolicy`

O código de exemplo a seguir mostra como usar `Unregister-IAMUserPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo desassocia a política gerenciada cujo ARN é **`arn:aws:iam::123456789012:policy/TesterPolicy`** do usuário do IAM chamado **`Bob`**.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/
TesterPolicy
```

Exemplo 2: este exemplo encontra todas as políticas gerenciadas que estão anexadas ao usuário do IAM chamado **`Theresa`** e as desassocia dele.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username
 Theresa
```

- Para obter detalhes sobre a API, consulte DetachUserPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMAccessKey`

O código de exemplo a seguir mostra como usar `Update-IAMAccessKey`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo altera para **`Inactive`** o status da chave de acesso **`AKIAIOSFODNN7EXAMPLE`** do usuário do IAM denominado **`Bob`**.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- Para obter detalhes sobre a API, consulte UpdateAccessKey Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `Update-IAMAccountPasswordPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a política de senha da conta com as configurações especificadas. Observe que quaisquer parâmetros que não estejam incluídos no comando não são modificados. Em vez disso, eles são redefinidos para os valores padrão.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- Para obter detalhes sobre a API, consulte UpdateAccountPasswordPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMAssumeRolePolicy`

O código de exemplo a seguir mostra como usar `Update-IAMAssumeRolePolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza o perfil do IAM denominado **`ClientRole`** com uma nova política de confiança, cujo conteúdo vem do arquivo **`ClientRolePolicy.json`**. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- Para obter detalhes sobre a API, consulte UpdateAssumeRolePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMGroup`

O código de exemplo a seguir mostra como usar `Update-IAMGroup`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo renomeia o grupo do IAM **`Testers`** para **`AppTesters`**.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Exemplo 2: este exemplo altera o caminho do grupo do IAM **`AppTesters`** para **`/Org1/Org2/`**. Isso altera o ARN do grupo para **`arn:aws:iam::123456789012:group/Org1/Org2/AppTesters`**.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Para obter detalhes sobre a API, consulte UpdateGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMLoginProfile`

O código de exemplo a seguir mostra como usar `Update-IAMLoginProfile`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo define uma nova senha temporária para o usuário **`Bob`** do IAM e exige que a pessoa altere a senha na próxima vez que fizer login.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired
 $true
```

- Para obter detalhes sobre a API, consulte UpdateLoginProfile Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMOpenIDConnectProviderThumbprint`

O código de exemplo a seguir mostra como usar `Update-IAMOpenIDConnectProviderThumbprint`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a lista de impressões digitais do certificado do provedor OIDC cujo ARN é **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`**, a fim de usar uma nova impressão digital. O provedor OIDC compartilha o novo valor quando o certificado associado ao provedor é alterado.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Para obter detalhes sobre a API, consulte UpdateOpenIdConnectProviderThumbprint Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMRole`

O código de exemplo a seguir mostra como usar `Update-IAMRole`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a descrição do perfil e o valor máximo da duração da sessão (em segundos) para o qual a sessão de um perfil pode ser solicitada.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Para obter detalhes sobre a API, consulte UpdateRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMRoleDescription`

O código de exemplo a seguir mostra como usar `Update-IAMRoleDescription`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a descrição de um perfil do IAM na sua conta.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Para obter detalhes sobre a API, consulte UpdateRoleDescription Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMSAMLProvider`

O código de exemplo a seguir mostra como usar `Update-IAMSAMLProvider`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza o provedor SAML no IAM cujo ARN é **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** com um novo documento de metadados SAML do arquivo **`SAMLMetaData.xml`**. Observe que você deve usar o parâmetro switch **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Para obter detalhes sobre a API, consulte UpdateSamlProvider Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMServerCertificate`

O código de exemplo a seguir mostra como usar `Update-IAMServerCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo renomeia o certificado denominado **`MyServerCertificate`** para **`MyRenamedServerCertificate`**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Exemplo 2: Esse exemplo move o certificado nomeado **`MyServerCertificate`** para o caminho/Org1/Org2/. Isso altera o ARN do recurso para **`arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate`**.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- Para obter detalhes sobre a API, consulte UpdateServerCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMSigningCertificate`

O código de exemplo a seguir mostra como usar `Update-IAMSigningCertificate`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza o certificado associado ao usuário do IAM chamado **`Bob`** e cujo ID do certificado é **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** para marcá-lo como inativo.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- Para obter detalhes sobre a API, consulte UpdateSigningCertificate Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-IAMUser`

O código de exemplo a seguir mostra como usar `Update-IAMUser`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo renomeia o usuário do IAM **`Bob`** para **`Robert`**.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Exemplo 2: este exemplo altera o caminho do usuário do IAM **`Bob`** para **`/Org1/Org2/`**, o que efetivamente altera o ARN do usuário para **`arn:aws:iam::123456789012:user/Org1/Org2/bob`**.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Para obter detalhes sobre a API, consulte UpdateUser Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-IAMGroupPolicy`

O código de exemplo a seguir mostra como usar `Write-IAMGroupPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma política em linha denominada **`AppTesterPolicy`** e a incorpora no grupo do IAM **`AppTesters`**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem no arquivo **`apptesterpolicy.json`**. Observe que você deve usar o parâmetro **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Para obter detalhes sobre a API, consulte PutGroupPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-IAMRolePolicy`

O código de exemplo a seguir mostra como usar `Write-IAMRolePolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma política em linha denominada **`FedTesterRolePolicy`** e a incorpora no perfil do IAM **`FedTesterRole`**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem do arquivo **`FedTesterPolicy.json`**. Observe que você deve usar o parâmetro **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Para obter detalhes sobre a API, consulte PutRolePolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-IAMUserPolicy`

O código de exemplo a seguir mostra como usar `Write-IAMUserPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma política em linha denominada **`EC2AccessPolicy`** e a incorpora no usuário do IAM **`Bob`**. Se já existir uma política em linha com o mesmo nome, ela será substituída. O conteúdo da política JSON vem do arquivo **`EC2AccessPolicy.json`**. Observe que você deve usar o parâmetro **`-Raw`** para processar com êxito o conteúdo do arquivo JSON.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-
Content -Raw EC2AccessPolicy.json)
```

- Para obter detalhes sobre a API, consulte PutUserPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Kinesis usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Kinesis.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-KINRecord`

O código de exemplo a seguir mostra como usar `Get-KINRecord`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como retornar e extrair dados de uma série de um ou mais registros. O iterador fornecido Get-KINRecord determina a posição inicial dos registros a serem retornados, que neste exemplo são capturados em uma variável, $records. Cada registro

individual pode então ser acessado indexando a coleção $records. Supondo que os dados no registro sejam texto UTF-8 codificado, o comando final mostra como você pode extrair os dados do MemoryStream objeto e devolvê-los como texto ao console.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Saída:

```
MillisBehindLatest NextShardIterator            Records
------------------ -----------------            -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Saída:

```
ApproximateArrivalTimestamp Data                   PartitionKey SequenceNumber
--------------------------- ----                   ------------ --------------
3/7/2016 5:14:33 PM         System.IO.MemoryStream Key1
 4955986459776...931586
```

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

Saída:

```
test data from string
```

- Para obter detalhes sobre a API, consulte GetRecords Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-KINShardIterator`

O código de exemplo a seguir mostra como usar `Get-KINShardIterator`.

Ferramentas para PowerShell V5

Exemplo 1: retorna um iterador de fragmentos para o fragmento e a posição inicial especificados. Detalhes dos identificadores de fragmentos e números de sequência podem ser obtidos na saída

do Get-KINStream cmdlet, fazendo referência à coleção Shards do objeto de fluxo retornado. O iterador retornado pode ser usado com o Get-KINRecord cmdlet para extrair registros de dados no fragmento.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Saída:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Para obter detalhes sobre a API, consulte GetShardIterator Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-KINStream`

O código de exemplo a seguir mostra como usar `Get-KINStream`.

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes do fluxo especificado.

```
Get-KINStream -StreamName "mystream"
```

Saída:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- Para obter detalhes sobre a API, consulte DescribeStream Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-KINStream`

O código de exemplo a seguir mostra como usar `New-KINStream`.

Ferramentas para PowerShell V5

Exemplo 1: cria um fluxo.

```
New-KINStream -StreamName "mystream" -ShardCount 1
```

- Para obter detalhes sobre a API, consulte CreateStream Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-KINStream`

O código de exemplo a seguir mostra como usar `Remove-KINStream`.

Ferramentas para PowerShell V5

Exemplo 1: exclui o fluxo especificado. A confirmação será solicitada antes que a operação continue. Para ignorar a solicitação de confirmação, use a opção -Force.

```
Remove-KINStream -StreamName "mystream"
```

- Para obter detalhes sobre a API, consulte DeleteStream Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-KINRecord`

O código de exemplo a seguir mostra como usar `Write-KINRecord`.

Ferramentas para PowerShell V5

Exemplo 1: grava um registro contendo a string fornecida ao parâmetro -Text.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey
 "Key1"
```

Exemplo 2: grava um registro contendo os dados contidos no arquivo especificado. O arquivo é tratado como uma sequência de bytes, portanto, se contiver texto, ele deverá ser gravado com qualquer codificação necessária antes de ser usado com esse cmdlet.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- Para obter detalhes sobre a API, consulte PutRecord Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Lambda usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Lambda.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-LMResourceTag`

O código de exemplo a seguir mostra como usar `Add-LMResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: adiciona as três tags (Washington, Oregon e Califórnia) e seus valores associados à função especificada identificada por seu ARN.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon"
 = "Salem"; "California" = "Sacramento" }
```

- Para obter detalhes sobre a API, consulte TagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMAccountSetting`

O código de exemplo a seguir mostra como usar `Get-LMAccountSetting`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo é apresentado para comparar o limite da conta e o uso da conta

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Saída:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- Para obter detalhes sobre a API, consulte GetAccountSettings Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMAlias`

O código de exemplo a seguir mostra como usar `Get-LMAlias`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera os pesos da configuração de roteamento para o alias de uma função do Lambda específica.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Saída:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- Para obter detalhes sobre a API, consulte GetAlias Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMFunctionConcurrency`

O código de exemplo a seguir mostra como usar `Get-LMFunctionConcurrency`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo obtém a simultaneidade reservada para a função do Lambda

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Saída:

```
ReservedConcurrentExecutions
----------------------------
100
```

- Para obter detalhes sobre a API, consulte GetFunctionConcurrency Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMFunctionConfiguration`

O código de exemplo a seguir mostra como usar `Get-LMFunctionConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a configuração específica de versão de uma função do Lambda.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Saída:

```
CodeSha256                 : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                   : 1426
DeadLetterConfig           : Amazon.Lambda.Model.DeadLetterConfig
Description                : Verson 3 to test Aliases
Environment                : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn                : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
                             :PowershellAlias
FunctionName               : MylambdaFunction123
Handler                    : lambda_function.launch_instance
```

```
KMSKeyArn                   :
LastModified                : 2019-12-25T09:52:59.872+0000
LastUpdateStatus            : Successful
LastUpdateStatusReason      :
LastUpdateStatusReasonCode  :
Layers                      : {}
MasterArn                   :
MemorySize                  : 128
RevisionId                  : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                        : arn:aws:iam::123456789012:role/service-role/lambda
Runtime                     : python3.8
State                       : Active
StateReason                 :
StateReasonCode             :
Timeout                     : 600
TracingConfig               : Amazon.Lambda.Model.TracingConfigResponse
Version                     : 4
VpcConfig                   : Amazon.Lambda.Model.VpcConfigDetail
```

- Para obter detalhes sobre a API, consulte GetFunctionConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMFunctionList`

O código de exemplo a seguir mostra como usar `Get-LMFunctionList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra todas as funções do Lambda com tamanho de código classificado

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Saída:

```
FunctionName                                                    Runtime   Timeout
 CodeSize
------------                                                    -------   -------
 --------
test                                                            python2.7       3
 243
```

```
MylambdaFunction123                                          python3.8     600
 659
myfuncpython1                                                python3.8     303
 675
```

- Para obter detalhes sobre a API, consulte ListFunctions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMPolicy`

O código de exemplo a seguir mostra como usar `Get-LMPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo mostra a política de função da função do Lambda

```
Get-LMPolicy -FunctionName test -Select Policy
```

Saída:

```
{"Version":"2012-10-17",       "Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lambda:
east-1:123456789102:function:test"}]}
```

- Para obter detalhes sobre a API, consulte GetPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMProvisionedConcurrencyConfig`

O código de exemplo a seguir mostra como usar `Get-LMProvisionedConcurrencyConfig`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo obtém a configuração de simultaneidade provisionada para o alias especificado da função do Lambda.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Saída:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- Para obter detalhes sobre a API, consulte GetProvisionedConcurrencyConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMProvisionedConcurrencyConfigList`

O código de exemplo a seguir mostra como usar `Get-LMProvisionedConcurrencyConfigList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a lista de configurações de simultaneidade provisionada para uma função do Lambda.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- Para obter detalhes sobre a API, consulte ListProvisionedConcurrencyConfigs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMResourceTag`

O código de exemplo a seguir mostra como usar `Get-LMResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: recupera as tags e seus valores atualmente definidos na função especificada.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Saída:

```
Key        Value
```

```
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- Para obter detalhes sobre a API, consulte ListTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-LMVersionsByFunction`

O código de exemplo a seguir mostra como usar `Get-LMVersionsByFunction`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna a lista de configurações específicas de versão para cada versão da função do Lambda.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Saída:

```
FunctionName        Runtime   MemorySize Timeout CodeSize LastModified
   RoleName
------------        -------   ---------- ------- -------- ------------
   --------
MylambdaFunction123 python3.8        128     600      659
 2020-01-10T03:20:56.390+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:19:02.238+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:39:36.779+0000 lambda
MylambdaFunction123 python3.8        128     600     1426
 2019-12-25T09:52:59.872+0000 lambda
```

- Para obter detalhes sobre a API, consulte ListVersionsByFunction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-LMAlias`

O código de exemplo a seguir mostra como usar `New-LMAlias`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um novo alias do Lambda para versão e configuração de roteamento especificadas a fim de indicar o percentual de solicitações de invocação que ele receberá.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias for
 version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Para obter detalhes sobre a API, consulte CreateAlias Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Publish-LMFunction`

O código de exemplo a seguir mostra como usar `Publish-LMFunction`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria uma nova função C# (dotnetcore1.0 runtime) chamada MyFunction no AWS Lambda, fornecendo os binários compilados para a função a partir de um arquivo zip no sistema de arquivos local (caminhos relativos ou absolutos podem ser usados). As funções C# Lambda especificam o manipulador da função usando a designação AssemblyName::::. Namespace.ClassName MethodName É necessário substituir adequadamente o nome da montagem (sem o sufixo .dll), o namespace, o nome da classe e o nome do método da especificação do manipulador. A nova função terá as variáveis de ambiente “envvar1” e “envvar2” configuradas com base nos valores fornecidos.

```
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -ZipFilename .\MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

Saída:

```
CodeSha256       : /NgBMd...gq71I=
```

```
CodeSize         : 214784
DeadLetterConfig :
Description      : My C# Lambda Function
Environment      : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn      : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName     : MyFunction
Handler          : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn        :
LastModified     : 2016-12-29T23:50:14.207+0000
MemorySize       : 128
Role             : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime          : dotnetcore1.0
Timeout          : 3
Version          : $LATEST
VpcConfig        :
```

Exemplo 2: este exemplo é semelhante ao anterior, com a exceção de que os binários da função são carregados primeiramente em um bucket do Amazon S3 (que deve estar na mesma região da função do Lambda desejada) e o objeto resultante do S3 será referenciado ao criar a função.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName amzn-s3-demo-bucket `
        -Key MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- Para obter detalhes sobre a API, consulte CreateFunction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Publish-LMVersion`

O código de exemplo a seguir mostra como usar `Publish-LMVersion`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria uma versão para o snapshot existente do código da função do Lambda

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- Para obter detalhes sobre a API, consulte PublishVersion Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-LMAlias`

O código de exemplo a seguir mostra como usar `Remove-LMAlias`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui o alias da função do Lambda mencionado no comando.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Para obter detalhes sobre a API, consulte DeleteAlias Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-LMFunction`

O código de exemplo a seguir mostra como usar `Remove-LMFunction`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo exclui uma versão específica de uma função do Lambda

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Para obter detalhes sobre a API, consulte DeleteFunction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-LMFunctionConcurrency`

O código de exemplo a seguir mostra como usar `Remove-LMFunctionConcurrency`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove a simultaneidade de função da função do Lambda.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Para obter detalhes sobre a API, consulte DeleteFunctionConcurrency Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-LMPermission`

O código de exemplo a seguir mostra como usar `Remove-LMPermission`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo remove a política de função para a especificação StatementId de uma função do Lambda.

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- Para obter detalhes sobre a API, consulte RemovePermission Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-LMProvisionedConcurrencyConfig`

O código de exemplo a seguir mostra como usar `Remove-LMProvisionedConcurrencyConfig`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo remove a configuração de simultaneidade provisionada para um alias específico.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier
 "NewAlias1"
```

- Para obter detalhes sobre a API, consulte DeleteProvisionedConcurrencyConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-LMResourceTag`

O código de exemplo a seguir mostra como usar `Remove-LMResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: remove as tags fornecidas de uma função. A menos que a opção -Force esteja especificada, o cmdlet solicitará a confirmação antes de continuar. Uma única chamada será feita para o serviço a fim de remover as tags.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

Exemplo 2: remove as tags fornecidas de uma função. A menos que a opção -Force esteja especificada, o cmdlet solicitará a confirmação antes de continuar. Isso acontece porque a chamada para o serviço é feita pela tag fornecida.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Para obter detalhes sobre a API, consulte UntagResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-LMAlias`

O código de exemplo a seguir mostra como usar `Update-LMAlias`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a configuração de um alias de função do Lambda existente. Ele atualiza o RoutingConfiguration valor para transferir 60% (0,6) do tráfego para a versão 1

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version
 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight
 @{Name="1";Value="0.6}
```

- Para obter detalhes sobre a API, consulte UpdateAlias Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-LMFunctionCode`

O código de exemplo a seguir mostra como usar `Update-LMFunctionCode`.

Ferramentas para PowerShell V5

Exemplo 1: atualiza a função chamada 'MyFunction' com o novo conteúdo contido no arquivo zip especificado. Para uma função do Lambda em C# .NET Core, o arquivo zip deve conter a montagem compilada.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Exemplo 2: este exemplo é semelhante ao anterior, mas usa um objeto do Amazon S3 contendo o código atualizado para atualizar a função.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -Key
 UpdatedCode.zip
```

- Para obter detalhes sobre a API, consulte UpdateFunctionCode Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-LMFunctionConfiguration`

O código de exemplo a seguir mostra como usar `Update-LMFunctionConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo atualiza a configuração da função do Lambda existente

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler
 "lambda_function.launch_instance" -Timeout 600 -Environment_Variable
 @{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:
 123456789101:MyfirstTopic
```

- Para obter detalhes sobre a API, consulte UpdateFunctionConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-LMFunctionConcurrency`

O código de exemplo a seguir mostra como usar `Write-LMFunctionConcurrency`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo aplica as configurações de simultaneidade para a função de maneira geral.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- Para obter detalhes sobre a API, consulte PutFunctionConcurrency Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-LMProvisionedConcurrencyConfig`

O código de exemplo a seguir mostra como usar `Write-LMProvisionedConcurrencyConfig`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo adiciona uma configuração de simultaneidade provisionada ao alias de uma função

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Para obter detalhes sobre a API, consulte PutProvisionedConcurrencyConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de Amazon ML usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon ML.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-MLBatchPrediction`

O código de exemplo a seguir mostra como usar `Get-MLBatchPrediction`.

Ferramentas para PowerShell V5

Exemplo 1: retorna os metadados detalhados para uma previsão em lote com ID de identificação.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- Para obter detalhes sobre a API, consulte GetBatchPrediction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-MLBatchPredictionList`

O código de exemplo a seguir mostra como usar `Get-MLBatchPredictionList`.

Ferramentas para PowerShell V5

Exemplo 1: retorna uma lista de todos BatchPredictions e seus registros de dados associados que correspondem ao critério de pesquisa fornecido na solicitação.

```
Get-MLBatchPredictionList
```

Exemplo 2: Retorna uma lista de todos BatchPredictions com o status CONCLUÍDO.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- Para obter detalhes sobre a API, consulte DescribeBatchPredictions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-MLDataSource`

O código de exemplo a seguir mostra como usar `Get-MLDataSource`.

Ferramentas para PowerShell V5

Exemplo 1: retorna os metadados, o status e as informações do arquivo de dados de a DataSource com o ID de id

```
Get-MLDataSource -DataSourceId ID
```

- Para obter detalhes sobre a API, consulte GetDataSource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-MLDataSourceList`

O código de exemplo a seguir mostra como usar `Get-MLDataSourceList`.

Ferramentas para PowerShell V5

Exemplo 1: retorna uma lista de todos DataSources e seus registros de dados associados.

```
Get-MLDataSourceList
```

Exemplo 2: Retorna uma lista de todos DataSources com o status CONCLUÍDO.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- Para obter detalhes sobre a API, consulte DescribeDataSources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-MLEvaluation`

O código de exemplo a seguir mostra como usar `Get-MLEvaluation`.

Ferramentas para PowerShell V5

Exemplo 1: retorna metadados e status para uma avaliação com ID de id.

```
Get-MLEvaluation -EvaluationId ID
```

- Para obter detalhes sobre a API, consulte GetEvaluation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-MLEvaluationList`

O código de exemplo a seguir mostra como usar `Get-MLEvaluationList`.

Ferramentas para PowerShell V5

Exemplo 1: retorna uma lista de todos os recursos de avaliação

```
Get-MLEvaluationList
```

Exemplo 2: retorna uma lista de todas as avaliações com o status COMPLETED.

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- Para obter detalhes sobre a API, consulte DescribeEvaluations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-MLModel`

O código de exemplo a seguir mostra como usar `Get-MLModel`.

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes de metadados, status, esquema e informações do arquivo de dados de um MLModel com ID de id.

```
Get-MLModel -ModelId ID
```

- Consulte detalhes da API em GetMLModel na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Get-MLModelList`

O código de exemplo a seguir mostra como usar `Get-MLModelList`.

Ferramentas para PowerShell V5

Exemplo 1: retorna uma lista de todos os modelos e seus registros de dados associados.

```
Get-MLModelList
```

Exemplo 2: retorna uma lista de todos os modelos com o status COMPLETED.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- Consulte detalhes da API em DescribeMLModels na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Get-MLPrediction`

O código de exemplo a seguir mostra como usar `Get-MLPrediction`.

Ferramentas para PowerShell V5

Exemplo 1: envie um registro para o URL do endpoint de previsão em tempo real para o modelo com ID de id.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- Consulte detalhes da API em Predict na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`New-MLBatchPrediction`

O código de exemplo a seguir mostra como usar `New-MLBatchPrediction`.

Ferramentas para PowerShell V5

Exemplo 1: crie uma solicitação de previsão de lote para o modelo com ID de id e coloca a saída no local especificado do S3.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- Para obter detalhes sobre a API, consulte CreateBatchPrediction Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-MLDataSourceFromS3`

O código de exemplo a seguir mostra como usar `New-MLDataSourceFromS3`.

Ferramentas para PowerShell V5

Exemplo 1: crie uma fonte de dados com dados de um local do S3, com um nome de NAME e um esquema de SCHEMA.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3
 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- Para obter detalhes sobre a API, consulte CreateDataSourceFromS3 Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-MLEvaluation

O código de exemplo a seguir mostra como usar `New-MLEvaluation`.

Ferramentas para PowerShell V5

Exemplo 1: crie uma avaliação para um determinado ID de fonte de dados e ID de modelo

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- Para obter detalhes sobre a API, consulte CreateEvaluation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-MLModel

O código de exemplo a seguir mostra como usar `New-MLModel`.

Ferramentas para PowerShell V5

Exemplo 1: crie um modelo com dados de treinamento.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- Consulte detalhes da API em CreateMLModel na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

New-MLRealtimeEndpoint

O código de exemplo a seguir mostra como usar `New-MLRealtimeEndpoint`.

Ferramentas para PowerShell V5

Exemplo 1: crie um endpoint de previsão em tempo real para o ID do modelo fornecido.

```
New-MLRealtimeEndpoint -ModelId ID
```

- Para obter detalhes sobre a API, consulte CreateRealtimeEndpoint Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de Macie usando Ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Macie.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-MAC2FindingList`

O código de exemplo a seguir mostra como usar `Get-MAC2FindingList`.

Ferramentas para PowerShell V5

Exemplo 1: retorna uma lista de quatro descobertas contendo uma detecção FindingIds de dados confidenciais com o tipo “CREDIT_CARD_NUMBER” ou “US_SOCIAL_SECURITY_NUMBER”

```
$criterionAddProperties = New-Object
 Amazon.Macie2.Model.CriterionAdditionalProperties

$criterionAddProperties.Eq = @(
"CREDIT_CARD_NUMBER"
"US_SOCIAL_SECURITY_NUMBER"
)
```

```
$FindingCriterion = @{
'classificationDetails.result.sensitiveData.detections.type' =
 [Amazon.Macie2.Model.CriterionAdditionalProperties]$criterionAddProperties
}

Get-MAC2FindingList -FindingCriteria_Criterion $FindingCriterion -MaxResult 5
```

- Para obter detalhes sobre a API, consulte ListFindings Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS Price List exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS Price List.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-PLSAttributeValue`

O código de exemplo a seguir mostra como usar `Get-PLSAttributeValue`.

Ferramentas para PowerShell V5

Exemplo 1: retorna os valores do atributo “volumeType” para o Amazon EC2 na região us-east-1.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-
east-1
```

Saída:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Para obter detalhes sobre a API, consulte GetAttributeValues Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-PLSProduct`

O código de exemplo a seguir mostra como usar `Get-PLSProduct`.

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes de todos os produtos para o Amazon EC2.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Saída:

```
{"product":{"productFamily":"Compute Instance","attributes":
{"enhancedNetworkingSupported":"Yes","memory":"30.5
 GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS
 Region","storage":"EBS only","instanceFamily":"Memory
 optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4
 (Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up
 to 10 Gigabit","servicename":"Amazon Elastic Compute
 Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-
BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX,
 Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License
 required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West
 (Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...
```

Exemplo 2: Retorna dados do Amazon EC2 na região us-east-1 filtrados por tipos de volume de “Uso geral” que são. SSD-backed

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
 @{Type="TERM_MATCH";Field="volumeType";Value="General
```

```
 Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region us-
east-1
```

Saída:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- Para obter detalhes sobre a API, consulte GetProducts Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-PLSService`

O código de exemplo a seguir mostra como usar `Get-PLSService`.

Ferramentas para PowerShell V5

Exemplo 1: retorna os metadados de todos os códigos de serviço disponíveis na região us-east-1.

```
Get-PLSService -Region us-east-1
```

Saída:

```
AttributeNames                                                  ServiceCode
--------------                                                  -----------
{productFamily, servicecode, groupDescription, termType...}     AWSBudgets
{productFamily, servicecode, termType, usagetype...}            AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}            AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}            AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}            AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}            AWSConfig
...
```

Exemplo 2: retorna os metadados para o serviço do Amazon EC2 na região us-east-1.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Saída:

```
AttributeNames                                                      ServiceCode
--------------                                                      -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Para obter detalhes sobre a API, consulte DescribeServices Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de grupos de recursos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com grupos de recursos.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-RGResourceTag`

O código de exemplo a seguir mostra como usar `Add-RGResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona a chave de tag “Instances” com o valor “workboxes” ao ARN do grupo de recursos fornecido

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Saída:

```
Arn                                                         Tags
```

```
---                                                                 ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Consulte detalhes da API em Tag na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Find-RGResource`

O código de exemplo a seguir mostra como usar `Find-RGResource`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um tipo de recurso ResourceQuery para instâncias com filtros de tags e localiza recursos.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
    Key = 'auto'
    Values = @('no')
  })
 }

Find-RGResource -ResourceQuery $query | Select-Object - expandProperty
 ResourceIdentifiers
```

Saída:

```
ResourceArn                                                         ResourceType
-----------                                                         ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- Para obter detalhes sobre a API, consulte SearchResources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-RGGroup`

O código de exemplo a seguir mostra como usar `Get-RGGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera o grupo de recursos de acordo com o nome do grupo

```
Get-RGGroup -GroupName auto-no
```

Saída:

```
Description GroupArn                                                    Name
----------- --------                                                    ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Para obter detalhes sobre a API, consulte GetGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-RGGroupList`

O código de exemplo a seguir mostra como usar `Get-RGGroupList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista o grupo de recursos já criado.

```
Get-RGGroupList
```

Saída:

```
GroupArn                                                                GroupName
--------                                                                ---------
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no            auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes           auto-yes
arn:aws:resource-groups:eu-west-1:123456789012:group/build600           build600
```

- Para obter detalhes sobre a API, consulte ListGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-RGGroupQuery`

O código de exemplo a seguir mostra como usar `Get-RGGroupQuery`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo busca a consulta de recursos para determinado grupo de recursos

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Saída:

```
Query
         Type
-----
         ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":
["no"]}]} TAG_FILTERS_1_0
```

- Para obter detalhes sobre a API, consulte GetGroupQuery Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-RGGroupResourceList`

O código de exemplo a seguir mostra como usar `Get-RGGroupResourceList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os recursos do grupo com base no filtro por tipo de recurso

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
 -GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

Saída:

```
ResourceArn                                                      ResourceType
-----------                                                      ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012 AWS::EC2::Instance
```

- Para obter detalhes sobre a API, consulte ListGroupResources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-RGResourceTag`

O código de exemplo a seguir mostra como usar `Get-RGResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista tags para o ARN do grupo de recursos fornecido

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Saída:

```
Key       Value
---       -----
Instances workboxes
```

- Para obter detalhes sobre a API, consulte GetTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-RGGroup`

O código de exemplo a seguir mostra como usar `New-RGGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo cria um novo grupo de recursos de grupos AWS de recursos baseado em tags chamado TestPowerShellGroup. O grupo inclui instâncias do Amazon EC2 na região atual que são marcadas com a chave de tag “Name” e o valor da tag “test2". O comando retorna a consulta, o tipo de grupo e os resultados da operação.

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
$ResourceQuery.Type = "TAG_FILTERS_1_0"
$ResourceQuery.Query = '{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"Name","Values":["test2"]}]}'
$ResourceQuery

New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
 "Test resource group."
```

Saída:

```
Query
            Type
-----
            ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":
["test2"]}]} TAG_FILTERS_1_0

LoggedAt         : 11/20/2018 2:40:59 PM
Group            : Amazon.ResourceGroups.Model.Group
ResourceQuery    : Amazon.ResourceGroups.Model.ResourceQuery
Tags             : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 338
HttpStatusCode   : OK
```

- Para obter detalhes sobre a API, consulte CreateGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-RGGroup`

O código de exemplo a seguir mostra como usar `Remove-RGGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove o grupo de recursos nomeado

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Description GroupArn
 Name
----------- --------
 ----
```

```
            arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-elbv2
 non-tag-cfn-elbv2
```

- Para obter detalhes sobre a API, consulte DeleteGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-RGResourceTag`

O código de exemplo a seguir mostra como usar `Remove-RGResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag mencionada do grupo de recursos

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGResourceTag (Untag)" on target "arn:aws:resource-
groups:eu-west-1:933303704102:group/workboxes".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                                                          Keys
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- Consulte detalhes da API em Untag na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Update-RGGroup`

O código de exemplo a seguir mostra como usar `Update-RGGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza a descrição do grupo

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Saída:

```
Description             GroupArn
  Name
-----------             --------
  ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-
yes auto-yes
```

- Para obter detalhes sobre a API, consulte UpdateGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-RGGroupQuery`

O código de exemplo a seguir mostra como usar `Update-RGGroupQuery`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um objeto de consulta e atualiza a consulta para o grupo.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
  Key='Environment'
  Values='Build600.11'
  })
} | ConvertTo-Json -Compress -Depth 4

Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

Saída:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Para obter detalhes sobre a API, consulte UpdateGroupQuery Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de API de marcação de grupos de recursos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com a API Resource Groups Tagging.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-RGTResourceTag`

O código de exemplo a seguir mostra como usar `Add-RGTResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona as chaves de tag “stage” e “version” com os valores “beta” e “preprod_test” a um bucket do Amazon S3 e a uma tabela do Amazon DynamoDB. Uma única chamada será feita para o serviço a fim de aplicar as tags.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

Exemplo 2: esse exemplo adiciona as tags e os valores especificados a um bucket do Amazon S3 e a uma tabela do Amazon DynamoDB. Duas chamadas são feitas para o serviço, uma para cada ARN de recurso canalizado para o cmdlet.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- Para obter detalhes sobre a API, consulte TagResources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-RGTResource`

O código de exemplo a seguir mostra como usar `Get-RGTResource`.

Ferramentas para PowerShell V5

Exemplo 1: retorna todos os recursos marcados em uma região e as chaves de tag associadas ao recurso. Se nenhum parâmetro -Region for fornecido ao cmdlet, ele tentará inferir a região pelo shell ou pelos metadados da instância do EC2.

```
Get-RGTResource
```

Saída:

```
ResourceARN                                                       Tags
-----------                                                       ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable             {stage, version}
arn:aws:s3:::amzn-s3-demo-bucket                                            {stage,
 version, othertag}
```

Exemplo 2: retorna todos os recursos marcados do tipo especificado em uma região. A string para cada nome de serviço e tipo de recurso é a mesma incorporada ao nome do recurso da Amazon (ARN) de um recurso.

```
Get-RGTResource -ResourceType "s3"
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

Exemplo 3: retorna todos os recursos marcados do tipo especificado em uma região. Observe que quando os tipos de recursos são canalizados para o cmdlet, uma chamada para o serviço é feita para cada tipo de recurso fornecido.

```
"dynamodb","s3" | Get-RGTResource
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

Exemplo 4: retorna todos os recursos marcados que correspondem ao filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

Exemplo 5: retorna todos os recursos marcados que correspondem ao filtro e ao tipo de recurso especificados.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
```

```
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

Exemplo 6: retorna todos os recursos marcados que correspondem ao filtro especificado.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Saída:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

- Para obter detalhes sobre a API, consulte GetResources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-RGTTagKey`

O código de exemplo a seguir mostra como usar `Get-RGTTagKey`.

Ferramentas para PowerShell V5

Exemplo 1: retorna todas as chaves de tag na região especificada. Se o parâmetro -Region não for especificado, o cmdlet tentará inferir a região por meio da região padrão do shell ou dos metadados da instância do EC2. Observe que as chaves da tag não são retornadas em nenhuma ordem específica.

```
Get-RGTTagKey -region us-west-2
```

Saída:

```
version
stage
```

- Para obter detalhes sobre a API, consulte GetTagKeys Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-RGTTagValue`

O código de exemplo a seguir mostra como usar `Get-RGTTagValue`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o valor da tag especificada em uma região. Se o parâmetro -Region não for especificado, o cmdlet tentará inferir a região por meio da região padrão do shell ou dos metadados da instância do EC2.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Saída:

```
beta
```

- Para obter detalhes sobre a API, consulte GetTagValues Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-RGTResourceTag`

O código de exemplo a seguir mostra como usar `Remove-RGTResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: remove as chaves de tag “stage” e “version” e os valores associados de um bucket do Amazon S3 e de uma tabela do Amazon DynamoDB. Uma única chamada será feita para o serviço a fim de remover as tags. Antes que as tags sejam removidas, o cmdlet solicitará a confirmação. Para ignorar a confirmação, adicione o parâmetro -Force.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Exemplo 2: remove as chaves de tag “stage” e “version” e os valores associados de um bucket do Amazon S3 e de uma tabela do Amazon DynamoDB. Duas chamadas são feitas para o serviço, uma para cada ARN de recurso canalizado para o cmdlet. Antes de cada chamada, o cmdlet solicitará a confirmação. Para ignorar a confirmação, adicione o parâmetro -Force.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"
```

```
$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Para obter detalhes sobre a API, consulte UntagResources Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Route 53 usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Route 53.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Edit-R53ResourceRecordSet`

O código de exemplo a seguir mostra como usar `Edit-R53ResourceRecordSet`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo cria um registro A para www.example.com e altera o registro A para test.example.com de 192.0.2.3 a 192.0.2.1. Observe que os valores dos TXT-type registros de alterações devem estar entre aspas duplas. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar e determinar quando as alterações serão concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
```

```
$change1.ResourceRecordSet.ResourceRecords = @()
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords = @()
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
$change3.ResourceRecordSet.ResourceRecords = @()
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
 ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
 and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
 ChangeBatch_Change=$change1,$change2,$change3
}

Edit-R53ResourceRecordSet @params
```

Exemplo 2: este exemplo mostra como criar conjuntos de registros de recurso do alias. 'Z222222222' é o ID da zona hospedada do Amazon Route 53 na qual você está criando o conjunto de registros de recurso do alias. 'exemplo.com' é o ápex da zona para o qual você deseja criar um alias e 'www.exemplo.com' é um subdomínio para o qual você também deseja criar um alias. 'Z1111111111111' é um exemplo de ID de zona hospedada para o balanceador de carga e 'example-load-balancer-1111111111.us-east-1.elb.amazonaws.com' é um exemplo de nome de domínio do balanceador de carga com o qual o Amazon Route 53 responde às consultas para example.com e www.example.com. Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar e determinar quando as alterações serão concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
```

```
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets, one
 for the zone apex, example.com, and one for www.example.com, that both point to
 example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemplo 3: Esse exemplo cria dois registros A para www.example.com. One-fourth Na maioria das vezes (1/ (1+3)), o Amazon Route 53 responde às consultas de www.example.com com os dois valores do primeiro conjunto de registros de recursos (192.0.2.9 e 192.0.2.10). Three-fourths da época (3/ (1+3)) O Amazon Route 53 responde às consultas de www.example.com com os dois valores para o segundo conjunto de registros de recursos (192.0.2.11 e 192.0.2.12). Consulte a documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar e determinar quando as alterações serão concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
```

```
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords = @()
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords = @()
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets, each
 of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemplo 4: Esse exemplo mostra como criar conjuntos de registros de recursos de aliases ponderados presumindo que example.com seja o domínio para o qual você deseja criar conjuntos de registros de recursos de aliases ponderados. SetIdentifier diferencia os dois conjuntos de registros de recursos de aliases ponderados um do outro. Esse elemento é necessário porque os elementos Nome e Tipo têm os mesmos valores para os dois conjuntos de registros de recurso. Z1111111111111 e Z3333333333333 são exemplos de IDs de zona hospedada para o balanceador de carga do ELB especificado pelo valor de DNSName. example-load-balancer-2222222222.us-east-1.elb.amazonaws.com e example-load-balancer-4444444444.us-east-1.amazonaws.com são exemplos de domínios do Elastic Load Balancing dos quais o Amazon Route 53 responde às consultas para example.com. Consulte a documentação do

Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar e determinar quando as alterações serão concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
 record sets. Amazon Route 53 responds to queries for example.com with the first ELB
 domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Exemplo 5: este exemplo cria dois conjuntos de registros de recurso do alias de latência, um para um balanceador de carga do ELB na região Oeste dos EUA (Oregon) (us-west-2) e outro para um balanceador de carga na região Ásia-Pacífico (Singapura) (ap-southeast-1). Consulte a

documentação do Amazon Route 53 para obter mais detalhes. Você pode usar o Get-R53Change cmdlet para pesquisar e determinar quando as alterações serão concluídas.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
 sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
 region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- Para obter detalhes sobre a API, consulte ChangeResourceRecordSets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-R53AccountLimit`

O código de exemplo a seguir mostra como usar `Get-R53AccountLimit`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o número máximo de zonas hospedadas que podem ser criadas usando a conta atual.

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

Saída:

```
15
```

- Para obter detalhes sobre a API, consulte GetAccountLimit Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-R53CheckerIpRanges`

O código de exemplo a seguir mostra como usar `Get-R53CheckerIpRanges`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna os CIDRs para os verificadores de integridade do Route53

```
Get-R53CheckerIpRanges
```

Saída:

```
15.177.2.0/23
15.177.6.0/23
15.177.10.0/23
15.177.14.0/23
15.177.18.0/23
15.177.22.0/23
15.177.26.0/23
15.177.30.0/23
15.177.34.0/23
15.177.38.0/23
15.177.42.0/23
```

```
15.177.46.0/23
15.177.50.0/23
15.177.54.0/23
15.177.58.0/23
15.177.62.0/23
54.183.255.128/26
54.228.16.0/26
54.232.40.64/26
54.241.32.64/26
54.243.31.192/26
54.244.52.192/26
54.245.168.0/26
54.248.220.0/26
54.250.253.192/26
54.251.31.128/26
54.252.79.128/26
54.252.254.192/26
54.255.254.192/26
107.23.255.0/26
176.34.159.192/26
177.71.207.128/26
```

- Para obter detalhes sobre a API, consulte GetCheckerIpRanges Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-R53HostedZone`

O código de exemplo a seguir mostra como usar `Get-R53HostedZone`.

Ferramentas para PowerShell V5

Exemplo 1: retorna detalhes da zona hospedada com o ID Z1D633PJN98FT9.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Para obter detalhes sobre a API, consulte GetHostedZone Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-R53HostedZoneCount`

O código de exemplo a seguir mostra como usar `Get-R53HostedZoneCount`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o número total de zonas hospedadas públicas e privadas da atual Conta da AWS.

```
Get-R53HostedZoneCount
```

- Para obter detalhes sobre a API, consulte GetHostedZoneCount Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-R53HostedZoneLimit`

O código de exemplo a seguir mostra como usar `Get-R53HostedZoneLimit`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o limite do número máximo de registros que podem ser criados na zona hospedada especificada.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

Saída:

```
5
```

- Para obter detalhes sobre a API, consulte GetHostedZoneLimit Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-R53HostedZoneList`

O código de exemplo a seguir mostra como usar `Get-R53HostedZoneList`.

Ferramentas para PowerShell V5

Exemplo 1: apresenta todas as zonas hospedadas públicas e privadas.

```
Get-R53HostedZoneList
```

Exemplo 2: apresenta todas as zonas hospedadas associadas ao conjunto de delegações reutilizáveis que tem o ID NZ8X2CISAMPLE

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Para obter detalhes sobre a API, consulte ListHostedZones Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-R53HostedZonesByName

O código de exemplo a seguir mostra como usar `Get-R53HostedZonesByName`.

Ferramentas para PowerShell V5

Exemplo 1: retorna todas as zonas hospedadas públicas e privadas em ordem ASCII por nome de domínio.

```
Get-R53HostedZonesByName
```

Exemplo 2: retorna as zonas hospedadas públicas e privadas em ordem ASCII por nome de domínio, a partir do nome DNS especificado.

```
Get-R53HostedZonesByName -DnsName example2.com
```

- Para obter detalhes sobre a API, consulte ListHostedZonesByName Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Get-R53QueryLoggingConfigList

O código de exemplo a seguir mostra como usar `Get-R53QueryLoggingConfigList`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo retorna todas as configurações de registro em log de consultas ao DNS associadas à Conta da AWS atual.

```
Get-R53QueryLoggingConfigList
```

Saída:

```
Id                                   HostedZoneId   CloudWatchLogsLogGroupArn
```

```
--                                   ------------   -------------------------
59b0fa33-4fea-4471-a88c-926476aaa40d Z385PDS6EAAAZR arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example1.com:*
ee528e95-4e03-4fdc-9d28-9e24ddaaa063 Z94SJHBV1AAAAZ arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b Z3MEQ8T7AAA1BF arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example3.com:*
```

- Para obter detalhes sobre a API, consulte ListQueryLoggingConfigs Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-R53ReusableDelegationSet`

O código de exemplo a seguir mostra como usar `Get-R53ReusableDelegationSet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera informações sobre o conjunto de delegações especificado, incluindo os quatro servidores atribuídos ao conjunto de delegações.

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

Saída:

```
Id                            CallerReference NameServers
--                            --------------- -----------
/delegationset/N23DS9X4AYEAAA testcaller      {ns-545.awsdns-04.net,
 ns-1264.awsdns-30.org, ns-2004.awsdns-58.co.uk, ns-240.awsdns-30.com}
```

- Para obter detalhes sobre a API, consulte GetReusableDelegationSet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-R53HostedZone`

O código de exemplo a seguir mostra como usar `New-R53HostedZone`.

Ferramentas para PowerShell V5

Exemplo 1: cria uma nova zona hospedada denominada 'example.com', associada a um conjunto de delegações reutilizável. Observe que você deve fornecer um valor para o CallerReference

parâmetro para que as solicitações precisem ser repetidas, se necessário, sem o risco de executar a operação duas vezes. Como a zona hospedada está sendo criada em uma VPC, ela é automaticamente privada e você não deve definir o PrivateZone parâmetro - HostedZoneConfig _.

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
    VPC_VPCRegion="us-east-1"
}

New-R53HostedZone @params
```

- Para obter detalhes sobre a API, consulte CreateHostedZone Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-R53QueryLoggingConfig`

O código de exemplo a seguir mostra como usar `New-R53QueryLoggingConfig`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma nova configuração de registro de consultas ao DNS do Route53 para a zona hospedada especificada. O Amazon Route53 publicará logs de consulta ao DNS no grupo de logs do Cloudwatch especificado.

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
 arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

Saída:

```
QueryLoggingConfig                      Location
------------------                      --------
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- Para obter detalhes sobre a API, consulte CreateQueryLoggingConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-R53ReusableDelegationSet

O código de exemplo a seguir mostra como usar `New-R53ReusableDelegationSet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um conjunto de delegação reutilizável de quatro servidores de nomes que podem ser utilizados novamente por várias zonas hospedadas.

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

Saída:

```
DelegationSet                      Location
-------------                      --------
Amazon.Route53.Model.DelegationSet https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM
```

- Para obter detalhes sobre a API, consulte CreateReusableDelegationSet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Register-R53VPCWithHostedZone

O código de exemplo a seguir mostra como usar `Register-R53VPCWithHostedZone`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa a VPC especificada à zona hospedada privada.

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

Saída:

```
Id                     Status  SubmittedAt         Comment
--                     ------  -----------         -------
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- Para obter detalhes sobre a API, consulte AssociateVPCWithHostedZone Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-R53HostedZone`

O código de exemplo a seguir mostra como usar `Remove-R53HostedZone`.

Ferramentas para PowerShell V5

Exemplo 1: exclui a zona hospedada com o ID especificado. Será solicitada uma confirmação antes que o comando continue, a menos que você adicione o parâmetro de opção -Force.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Para obter detalhes sobre a API, consulte DeleteHostedZone Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-R53QueryLoggingConfig`

O código de exemplo a seguir mostra como usar `Remove-R53QueryLoggingConfig`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a configuração especificada para o registro de consultas ao DNS.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- Para obter detalhes sobre a API, consulte DeleteQueryLoggingConfig Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-R53ReusableDelegationSet`

O código de exemplo a seguir mostra como usar `Remove-R53ReusableDelegationSet`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o conjunto de delegações reutilizáveis especificado.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- Para obter detalhes sobre a API, consulte DeleteReusableDelegationSet Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-R53VPCFromHostedZone`

O código de exemplo a seguir mostra como usar `Unregister-R53VPCFromHostedZone`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo dissocia a VPC especificada da zona hospedada privada.

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa
 -VPC_VPCRegion us-east-1
```

Saída:

```
Id                      Status  SubmittedAt         Comment
--                      ------  -----------         -------
/change/C2XFCAAAA9HKZG PENDING 01/28/2020 10:35:55
```

- Para obter detalhes sobre a API, consulte DisassociateVPCFromHostedZone Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-R53HostedZoneComment`

O código de exemplo a seguir mostra como usar `Update-R53HostedZoneComment`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando atualiza o comentário para a zona hospedada especificada.

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted
 zone"
```

Saída:

```
Id                     : /hostedzone/Z385PDS6AAAAAR
Name                   : example.com.
CallerReference        : C5B55555-7147-EF04-8341-69131E805C89
Config                 : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService          :
```

- Para obter detalhes sobre a API, consulte UpdateHostedZoneComment Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon S3 usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon S3.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Copy-S3Object`

O código de exemplo a seguir mostra como usar `Copy-S3Object`.

Ferramentas para PowerShell V5

Exemplo 1: este comando copia o objeto “sample.txt” do bucket “test-files” para o mesmo bucket, mas com uma nova chave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

Exemplo 2: este comando copia o objeto “sample.txt” do bucket “test-files” para o bucket “backup-files” com uma nova chave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey
 sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Exemplo 3: este comando baixa o objeto “sample.txt” do bucket “test-files” em um arquivo local com o nome “local-sample.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Exemplo 4: um único objeto é baixado no arquivo especificado. O arquivo baixado encontra-se em c:\downloads\data\archive.zip.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:
\downloads
```

Exemplo 5: todos os objetos que correspondem ao prefixo de chave especificado são baixados na pasta local. A hierarquia relativa de chaves será preservada como subpastas no local geral do download.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

Exemplo 6: baixa um único objeto em um arquivo local usando o download paralelo de várias partes para melhorar o desempenho em arquivos grandes.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt -UseMultipartDownload
```

Exemplo 7: baixa um objeto grande usando o modo RANGE com um tamanho de parte personalizado de 64 MB. O modo RANGE usa solicitações HTTP de intervalo de bytes e funciona para todos os objetos, independentemente de como eles foram carregados.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-
large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Exemplo 8: baixa todos os objetos que correspondem ao prefixo de chave em uma pasta local usando o download de várias partes com downloads simultâneos de arquivos.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:
\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Exemplo 9: baixa um único objeto usando o download de várias partes com 20 conexões simultâneas para obter o throughput máximo em redes de alta largura de banda.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-
large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Exemplo 10: baixa todos os objetos que correspondem ao prefixo de chave para uma pasta local usando o download de várias partes, continuando com as falhas de arquivos individuais em vez de interrompê-las.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:
\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Para obter detalhes sobre a API, consulte CopyObject Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3Bucket`

O código de exemplo a seguir mostra como usar `Get-S3Bucket`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todos os buckets do S3.

```
Get-S3Bucket
```

Exemplo 2: este comando retorna um bucket denominado “test-files”.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes sobre a API, consulte ListBuckets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketACL`

O código de exemplo a seguir mostra como usar `Get-S3BucketACL`.

Ferramentas para PowerShell V5

Exemplo 1: o comando obtém os detalhes do proprietário do objeto do S3.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Saída:

```
DisplayName Id
----------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Para obter detalhes sobre a API, consulte GetBucketAcl Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketAccelerateConfiguration`

O código de exemplo a seguir mostra como usar `Get-S3BucketAccelerateConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: se as configurações de aceleração de transferência estiverem habilitadas para o bucket especificado, este comando retornará o valor Habilitado.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
-----
Enabled
```

- Para obter detalhes sobre a API, consulte GetBucketAccelerateConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketAnalyticsConfiguration`

O código de exemplo a seguir mostra como usar `Get-S3BucketAnalyticsConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes do filtro de análise com o nome “testfilter” no bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Para obter detalhes sobre a API, consulte GetBucketAnalyticsConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketAnalyticsConfigurationList`

O código de exemplo a seguir mostra como usar `Get-S3BucketAnalyticsConfigurationList`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna as cem primeiras análises do bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte ListBucketAnalyticsConfigurations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketEncryption`

O código de exemplo a seguir mostra como usar `Get-S3BucketEncryption`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todas as regras de criptografia do lado do servidor associadas ao bucket em questão.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte GetBucketEncryption Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketInventoryConfiguration`

O código de exemplo a seguir mostra como usar `Get-S3BucketInventoryConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes do inventário denominado “testinventory” para o bucket do S3 em questão.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- Para obter detalhes sobre a API, consulte GetBucketInventoryConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketInventoryConfigurationList`

O código de exemplo a seguir mostra como usar `Get-S3BucketInventoryConfigurationList`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna as cem primeiras configurações de inventário do bucket do S3 em questão.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte ListBucketInventoryConfigurations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketLocation`

O código de exemplo a seguir mostra como usar `Get-S3BucketLocation`.

Ferramentas para PowerShell V5

Exemplo 1: se houver uma restrição, este comando retornará a restrição de localização do bucket “amzn-s3-demo-bucket”.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
-----
ap-south-1
```

- Para obter detalhes sobre a API, consulte GetBucketLocation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketLogging`

O código de exemplo a seguir mostra como usar `Get-S3BucketLogging`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status de registro em log do bucket especificado.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- Para obter detalhes sobre a API, consulte GetBucketLogging Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketMetricsConfiguration`

O código de exemplo a seguir mostra como usar `Get-S3BucketMetricsConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna detalhes sobre o filtro de métricas denominado “testfilter” para o bucket do S3 em questão.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- Para obter detalhes sobre a API, consulte GetBucketMetricsConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketNotification`

O código de exemplo a seguir mostra como usar `Get-S3BucketNotification`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a configuração de notificação do bucket em questão.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

Saída:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Para obter detalhes sobre a API, consulte GetBucketNotification Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketPolicy`

O código de exemplo a seguir mostra como usar `Get-S3BucketPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: este comando gera a política de bucket associada ao bucket do S3 em questão.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte GetBucketPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketPolicyStatus`

O código de exemplo a seguir mostra como usar `Get-S3BucketPolicyStatus`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status da política do bucket do S3 em questão, indicando se o bucket é público.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte GetBucketPolicyStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketReplication`

O código de exemplo a seguir mostra como usar `Get-S3BucketReplication`.

Ferramentas para PowerShell V5

Exemplo 1: retorna as informações da configuração de replicação definida no bucket denominado “amzn-s3-demo-bucket”.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes sobre a API, consulte GetBucketReplication Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketRequestPayment`

O código de exemplo a seguir mostra como usar `Get-S3BucketRequestPayment`.

Ferramentas para PowerShell V5

Exemplo 1: retorna as informações da configuração de pagamento de solicitação do bucket denominado “amzn-s3-demo-bucket”. Por padrão, o proprietário do bucket paga pelos downloads feitos no bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes sobre a API, consulte GetBucketRequestPayment Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketTagging`

O código de exemplo a seguir mostra como usar `Get-S3BucketTagging`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todas as tags associadas ao bucket em questão.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte GetBucketTagging Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketVersioning`

O código de exemplo a seguir mostra como usar `Get-S3BucketVersioning`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status de versionamento referente ao bucket em questão.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte GetBucketVersioning Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3BucketWebsite`

O código de exemplo a seguir mostra como usar `Get-S3BucketWebsite`.

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes das configurações do site estático do bucket do S3 em questão.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte GetBucketWebsite Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3CORSConfiguration`

O código de exemplo a seguir mostra como usar `Get-S3CORSConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando retorna um objeto que contém todas as regras de configuração do CORS correspondentes ao bucket do S3 em questão.

```
Get-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 Configuration.Rules
```

Saída:

```
AllowedMethods : {PUT, POST, DELETE}
AllowedOrigins : {http://www.example1.com}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {*}

AllowedMethods : {PUT, POST, DELETE}
AllowedOrigins : {http://www.example2.com}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {*}

AllowedMethods : {GET}
AllowedOrigins : {*}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {}
```

- Consulte detalhes da API em GetCORSConfiguration na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Get-S3HeadBucket`

O código de exemplo a seguir mostra como usar `Get-S3HeadBucket`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna a saída com o código de status HTTP 200 OK para o bucket existente quando o usuário tem permissão para acessá-lo. BucketArn O parâmetro só é suportado para buckets de diretório do S3.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Saída:

```
AccessPointAlias   : False
BucketArn          :
BucketLocationName :
BucketLocationType :
BucketRegion       : us-east-2
ResponseMetadata   : Amazon.Runtime.ResponseMetadata
ContentLength      : 0
HttpStatusCode     : OK
```

Exemplo 2: Esse comando gera um erro com o código de status HTTP NotFound para um bucket inexistente.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Saída:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status Code
 NotFound. No further error information was returned by the service.
```

Exemplo 3: esse comando gera um erro com o código de status HTTP Proibido para um bucket existente em que o usuário não tem permissão para acessá-lo.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Saída:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status
 Code Forbidden. No further error information was returned by the service.
```

- Para obter detalhes sobre a API, consulte HeadBucket Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3LifecycleConfiguration`

O código de exemplo a seguir mostra como usar `Get-S3LifecycleConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera a configuração do ciclo de vida do bucket.

```
Get-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket
```

Saída:

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- Para obter detalhes sobre a API, consulte GetLifecycleConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3Object`

O código de exemplo a seguir mostra como usar `Get-S3Object`.

Ferramentas para PowerShell V5

Exemplo 1: este comando recupera as informações sobre todos os itens no bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Exemplo 2: este comando recupera as informações sobre o item “sample.txt” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 3: este comando recupera as informações sobre todos os itens com prefixo “sample” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Para obter detalhes sobre a API, consulte ListObjects Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3ObjectACL`

O código de exemplo a seguir mostra como usar `Get-S3ObjectACL`.

Ferramentas para PowerShell V5

Exemplo 1: o comando obtém os detalhes do proprietário do objeto do S3.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
 *).Owner
```

Saída:

```
DisplayName Id
----------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Para obter detalhes sobre a API, consulte GetObjectAcl Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3ObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `Get-S3ObjectLockConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: se a configuração do Bloqueio de Objetos estiver habilitada para o bucket do S3 em questão, este comando retornará o valor “Habilitado”.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Saída:

```
Value
-----
Enabled
```

- Para obter detalhes sobre a API, consulte GetObjectLockConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3ObjectMetadata`

O código de exemplo a seguir mostra como usar `Get-S3ObjectMetadata`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna os metadados do objeto com a chave 'ListTrusts.txt' no bucket do S3 fornecido.

```
Get-S3ObjectMetadata -BucketName 'amzn-s3-demo-bucket' -Key 'ListTrusts.txt'
```

Saída:

```
Headers                                       : Amazon.S3.Model.HeadersCollection
Metadata                                      : Amazon.S3.Model.MetadataCollection
DeleteMarker                                  :
AcceptRanges                                  : bytes
ContentRange                                  :
Expiration                                    :
RestoreExpiration                             :
RestoreInProgress                             : False
LastModified                                  : 01/01/2020 08:02:05
ETag                                          : "d000011112a222e333e3bb4ee5d43d21"
MissingMeta                                   : 0
VersionId                                     : null
Expires                                       : 01/01/0001 00:00:00
WebsiteRedirectLocation                       :
ServerSideEncryptionMethod                    : AES256
ServerSideEncryptionCustomerMethod            :
ServerSideEncryptionKeyManagementServiceKeyId :
ReplicationStatus                             :
PartsCount                                    :
ObjectLockLegalHoldStatus                     :
ObjectLockMode                                :
```

```
ObjectLockRetainUntilDate                       : 01/01/0001 00:00:00
StorageClass                                    :
RequestCharged                                  :
```

- Para obter detalhes sobre a API, consulte GetObjectMetadata Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3ObjectRetention`

O código de exemplo a seguir mostra como usar `Get-S3ObjectRetention`.

Ferramentas para PowerShell V5

Exemplo 1: o comando retorna o modo e a data até a qual o objeto ficará retido.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Para obter detalhes sobre a API, consulte GetObjectRetention Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3ObjectTagSet`

O código de exemplo a seguir mostra como usar `Get-S3ObjectTagSet`.

Ferramentas para PowerShell V5

Exemplo 1: o exemplo retorna as tags associadas ao objeto presente no bucket do S3 em questão.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Key  Value
---  -----
test value
```

- Para obter detalhes sobre a API, consulte GetObjectTagging Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3PreSignedURL`

O código de exemplo a seguir mostra como usar `Get-S3PreSignedURL`.

Ferramentas para PowerShell V5

Exemplo 1: o comando retorna um URL pré-assinado para uma chave especificada e uma data de validade.

```
Get-S3PreSignedURL -BucketName 'amzn-s3-demo-bucket' -Key 'testkey' -Expires
 '2023-11-16'
```

Exemplo 2: o comando retorna um URL pré-assinado para um bucket de diretório com a chave especificada e uma data de validade.

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
      Get-S3PreSignedURL -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -Key
 'testkey' -Expire '2023-11-17'
```

- Para obter detalhes sobre a API, consulte GetPreSignedURL Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3PublicAccessBlock`

O código de exemplo a seguir mostra como usar `Get-S3PublicAccessBlock`.

Ferramentas para PowerShell V5

Exemplo 1: o comando retorna a configuração do Bloqueio de Acesso Público do bucket do S3 em questão.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte GetPublicAccessBlock Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-S3Version`

O código de exemplo a seguir mostra como usar `Get-S3Version`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando retorna os metadados sobre todas as versões dos objetos no bucket do S3 em questão.

```
Get-S3Version -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
IsTruncated         : False
KeyMarker           :
VersionIdMarker     :
NextKeyMarker       :
NextVersionIdMarker :
Versions            : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
 ListTrusts.json}
Name                : amzn-s3-demo-bucket
Prefix              :
MaxKeys             : 1000
CommonPrefixes      : {}
Delimiter           :
```

- Para obter detalhes sobre a API, consulte ListVersions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-S3Bucket`

O código de exemplo a seguir mostra como usar `New-S3Bucket`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando cria um bucket privado chamado “sample-bucket”.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket
```

Exemplo 2: esse comando cria um bucket chamado “sample-bucket” com permissões de leitura e gravação.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadWrite
```

Exemplo 3: esse comando cria um novo bucket chamado “sample-bucket” com permissões somente de leitura.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadOnly
```

Exemplo 4: Esse comando cria um novo bucket de diretório chamado “amzn-s3-demo-bucket--use1-az5--x-s3" com. PutBucketConfiguration

```
$bucketConfiguration = @{
      BucketInfo = @{
          DataRedundancy = 'SingleAvailabilityZone'
          Type = 'Directory'
      }
      Location = @{
          Name = 'usw2-az1'
          Type = 'AvailabilityZone'
      }
    }
New-S3Bucket -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -BucketConfiguration
 $bucketConfiguration -Region us-west-2
```

- Para obter detalhes sobre a API, consulte PutBucket Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Read-S3Object`

O código de exemplo a seguir mostra como usar `Read-S3Object`.

Ferramentas para PowerShell V5

Exemplo 1: este comando recupera o item “sample.txt” do bucket “amzn-s3-demo-bucket” e o salva em um arquivo chamado “local-sample.txt” no local atual. O arquivo “local-sample.txt” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Exemplo 2: Esse comando recupera o diretório virtual “DIR” do bucket “amzn-s3-demo-bucket” e o salva em uma pasta chamada "" no local atual. Local-DIR A pasta "Local-DIR" não precisa existir para que esse comando seja chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Exemplo 3: baixa todos os objetos com chaves terminadas em ".json" de buckets com "config" no respectivo nome em arquivos na pasta especificada. As chaves do objeto são usadas para definir o nome dos arquivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

Exemplo 4: baixa um único objeto usando o download paralelo de várias partes para melhorar o desempenho em arquivos grandes.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
 -UseMultipartDownload
```

Exemplo 5: baixa um objeto grande usando o modo RANGE com um tamanho de parte personalizado de 64 MB. O modo RANGE usa solicitações HTTP de intervalo de bytes e funciona para todos os objetos, independentemente de como eles foram carregados.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-
file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Exemplo 6: baixa todos os objetos com o prefixo de chave "data/" para uma pasta local usando o download de várias partes com downloads simultâneos de arquivos.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data
 -UseMultipartDownload -DownloadFilesConcurrently
```

Exemplo 7: baixa um único objeto usando o download de várias partes com 20 conexões simultâneas para obter o throughput máximo em redes de alta largura de banda.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-
file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Exemplo 8: baixa todos os objetos com o prefixo de chave "data/" para uma pasta local usando o download de várias partes, continuando com as falhas de arquivos individuais em vez de interrompê-las.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data
 -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Para obter detalhes sobre a API, consulte GetObject Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3Bucket`

O código de exemplo a seguir mostra como usar `Remove-S3Bucket`.

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todos os objetos e versões de objetos do bucket “test-files” e, em seguida, exclui o bucket. O comando solicitará a confirmação antes de continuar. Adicione a opção -Force para ignorar a confirmação. Observe que os buckets que não estão vazios não podem ser excluídos.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Para obter detalhes sobre a API, consulte DeleteBucket Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3BucketAnalyticsConfiguration`

O código de exemplo a seguir mostra como usar `Remove-S3BucketAnalyticsConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: o comando remove o filtro de análise com o nome “testfilter” no bucket do S3 em questão.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Para obter detalhes sobre a API, consulte DeleteBucketAnalyticsConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3BucketEncryption`

O código de exemplo a seguir mostra como usar `Remove-S3BucketEncryption`.

Ferramentas para PowerShell V5

Exemplo 1: isso desabilita a criptografia habilitada para o bucket do S3 fornecido.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteBucketEncryption Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3BucketInventoryConfiguration`

O código de exemplo a seguir mostra como usar `Remove-S3BucketInventoryConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando remove o inventário chamado InventoryName 'test' correspondente ao bucket do S3 fornecido.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testInventoryName'
```

Saída:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteBucketInventoryConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3BucketMetricsConfiguration`

O código de exemplo a seguir mostra como usar `Remove-S3BucketMetricsConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: o comando remove o filtro de métricas com o nome “testmetrics” no bucket do S3 em questão.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- Para obter detalhes sobre a API, consulte DeleteBucketMetricsConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3BucketPolicy`

O código de exemplo a seguir mostra como usar `Remove-S3BucketPolicy`.

Ferramentas para PowerShell V5

Exemplo 1: o comando remove a política de bucket associada ao bucket do S3 em questão.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte DeleteBucketPolicy Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3BucketReplication`

O código de exemplo a seguir mostra como usar `Remove-S3BucketReplication`.

Ferramentas para PowerShell V5

Exemplo 1: exclui a configuração de replicação associada ao bucket denominado “amzn-s3-demo-bucket”. Observe que essa operação requer permissão para a DeleteReplicationConfiguration ação s3:. Será solicitada uma confirmação antes que a operação continue. Para ignorar a confirmação, use a opção -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes sobre a API, consulte DeleteBucketReplication Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3BucketTagging`

O código de exemplo a seguir mostra como usar `Remove-S3BucketTagging`.

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todas as tags associadas ao bucket do S3 em questão.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteBucketTagging Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3BucketWebsite`

O código de exemplo a seguir mostra como usar `Remove-S3BucketWebsite`.

Ferramentas para PowerShell V5

Exemplo 1: este comando desabilita a propriedade de hospedagem de site estático do bucket do S3 em questão.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteBucketWebsite Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3CORSConfiguration`

O código de exemplo a seguir mostra como usar `Remove-S3CORSConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando remove a configuração CORS do bucket do S3 em questão.

```
Remove-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
 target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Consulte detalhes da API em DeleteCORSConfiguration na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

`Remove-S3LifecycleConfiguration`

O código de exemplo a seguir mostra como usar `Remove-S3LifecycleConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: o comando remove todas as regras do ciclo de vida do bucket do S3 em questão.

```
Remove-S3LifecycleConfiguration -BucketName 'amzn-s3-demo-bucket'
```

- Para obter detalhes sobre a API, consulte DeleteLifecycleConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3MultipartUpload`

O código de exemplo a seguir mostra como usar `Remove-S3MultipartUpload`.

Ferramentas para PowerShell V5

Exemplo 1: este comando interrompe os carregamentos fracionados criados há mais de cinco dias.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Exemplo 2: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "Thursday,
 January 02, 2014"
```

Exemplo 3: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014, às 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "2014/01/02
 10:45:37"
```

- Para obter detalhes sobre a API, consulte AbortMultipartUpload Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3Object`

O código de exemplo a seguir mostra como usar `Remove-S3Object`.

Ferramentas para PowerShell V5

Exemplo 1: este comando remove o objeto “sample.txt” do bucket “test-files”. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a solicitação, use a opção -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 2: este comando remove a versão especificada do objeto “sample.txt” do bucket “test-files”, supondo que o bucket tenha sido configurado para habilitar versões de objetos.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemplo 3: este comando remove objetos “sample1.txt”, “sample2.txt” e “sample3.txt” do bucket “test-files” como uma operação em lote única. A resposta do serviço listará todas as chaves processadas, independentemente do status de êxito ou erro da exclusão. Para obter somente erros para chaves que não puderam ser processadas pelo serviço, adicione o ReportErrorsOnly parâmetro - (esse parâmetro também pode ser especificado com o alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Exemplo 4: Esse exemplo usa uma expressão embutida com o KeyCollection parâmetro - para obter as chaves dos objetos a serem excluídos. Get-S3Object retorna uma coleção de Amazon.S3.Model.S3Object instâncias, cada uma com um membro Key do tipo string identificando o objeto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Exemplo 5: Esse exemplo obtém todos os objetos que têm um prefixo de chave "prefix/subprefix" no bucket e os exclui. Observe que os objetos de entrada são processados um de cada vez. Para coleções grandes, considere passar a coleção para o parâmetro - InputObject (alias -

S3ObjectCollection) do cmdlet para permitir que a exclusão ocorra como um lote com uma única chamada para o serviço.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

Exemplo 6: Esse exemplo envia uma coleção de Amazon.S3.Model.S3ObjectVersion instâncias que representam marcadores de exclusão para o cmdlet para exclusão. Observe que os objetos de entrada são processados um de cada vez. Para coleções grandes, considere passar a coleção para o parâmetro - InputObject (alias -S3ObjectCollection) do cmdlet para permitir que a exclusão ocorra como um lote com uma única chamada para o serviço.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemplo 7: Esse script mostra como realizar uma exclusão em lote de um conjunto de objetos (nesse caso, marcadores de exclusão) construindo uma matriz de objetos a serem usados com o parâmetro -KeyAndVersionCollection .

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -Force
```

- Para obter detalhes sobre a API, consulte DeleteObjects Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3ObjectTagSet`

O código de exemplo a seguir mostra como usar `Remove-S3ObjectTagSet`.

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todas as tags associadas ao objeto com a chave “testfile.txt” no bucket do S3 em questão.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -Select
 '^Key'
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- Para obter detalhes sobre a API, consulte DeleteObjectTagging Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-S3PublicAccessBlock`

O código de exemplo a seguir mostra como usar `Remove-S3PublicAccessBlock`.

Ferramentas para PowerShell V5

Exemplo 1: este comando desativa a configuração do Bloqueio de Acesso Público do bucket em questão.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

Saída:

```
amzn-s3-demo-bucket
```

- Para obter detalhes sobre a API, consulte DeletePublicAccessBlock Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-S3BucketEncryption`

O código de exemplo a seguir mostra como usar `Set-S3BucketEncryption`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando ativa a criptografia AES256 padrão do lado do servidor com as chaves gerenciadas do Amazon S3 (SSE-S3) no bucket especificado.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Para obter detalhes sobre a API, consulte PutBucketEncryption Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Test-S3Bucket`

O código de exemplo a seguir mostra como usar `Test-S3Bucket`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando retornará True se o bucket existir, caso contrário, False. O comando retornará True mesmo que o bucket não pertença ao usuário.

```
Test-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Para obter detalhes sobre a API, consulte Test-S3Bucket Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3BucketAccelerateConfiguration`

O código de exemplo a seguir mostra como usar `Write-S3BucketAccelerateConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: este comando habilita a aceleração de transferência do bucket do S3 em questão.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- Para obter detalhes sobre a API, consulte PutBucketAccelerateConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3BucketNotification`

O código de exemplo a seguir mostra como usar `Write-S3BucketNotification`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo configura a configuração do tópico SNS para o evento S3 ObjectRemovedDelete e habilita a notificação para o bucket s3 fornecido

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
 $topic
```

Exemplo 2: Esse exemplo permite que as notificações de ObjectCreatedAll um determinado bucket sejam enviadas para a função Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

Exemplo 3: este exemplo cria duas configurações diferentes do Lambda com base em diferentes sufixos de chave e configura ambas em um único comando.

```
#Lambda Config 1
```

```
$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- Para obter detalhes sobre a API, consulte PutBucketNotification Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3BucketReplication`

O código de exemplo a seguir mostra como usar `Write-S3BucketReplication`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo define uma configuração de replicação com uma única regra que permite replicar para o bucket 'amzn-s3-demo-bucket' quaisquer novos objetos criados com o prefixo de nome de chave "" no bucket 'amzn-s3-demo-bucket'. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Exemplo 2: Este exemplo define uma configuração de replicação com várias regras que permitem a replicação para o bucket 'amzn-s3-demo-bucket' de qualquer objeto novo criado com o prefixo de nome de chave "" ou "”. TaxDocs OtherDocs Os prefixos de chave não devem se sobrepor.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
```

```
}

Write-S3BucketReplication @params
```

Exemplo 3: Este exemplo atualiza a configuração de replicação no bucket especificado para desativar a regra que controla a replicação de objetos com o prefixo de nome de chave "TaxDocs" no bucket 'amzn-s3-demo-bucket'.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Para obter detalhes sobre a API, consulte PutBucketReplication Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3BucketRequestPayment`

O código de exemplo a seguir mostra como usar `Write-S3BucketRequestPayment`.

Ferramentas para PowerShell V5

Exemplo 1: atualiza a configuração de pagamento de solicitação do bucket denominado “amzn-s3-demo-bucket” para que a pessoa que está solicitando downloads pelo bucket seja cobrada pelo download. Por padrão, o proprietário do bucket paga pelos downloads. Para redefinir o pagamento solicitado como padrão, use 'BucketOwner' para o parâmetro RequestPaymentConfiguration _Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Para obter detalhes sobre a API, consulte PutBucketRequestPayment Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3BucketTagging`

O código de exemplo a seguir mostra como usar `Write-S3BucketTagging`.

Ferramentas para PowerShell V5

Exemplo 1: este comando aplica duas tags a um bucket denominado **`cloudtrail-test-2018`** (uma tag com uma chave de Stage e um valor de Test e uma tag com uma chave de Environment e um valor de Alpha). Para verificar se as tags foram adicionadas ao bucket, execute **`Get-S3BucketTagging -BucketName bucket_name`**. Os resultados devem mostrar as tags que você aplicou ao bucket no primeiro comando. Observe que **`Write-S3BucketTagging`** substitui todo o conjunto de tags existente em um bucket. Para adicionar ou excluir tags individuais, execute os cmdlets da API de grupos de recursos e marcação: **`Add-RGTResourceTag`** e **`Remove-RGTResourceTag`**. Como alternativa, use o Tag Editor no AWS Management Console para gerenciar tags de bucket do S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemplo 2: este comando envia um bucket denominado **`cloudtrail-test-2018`** para o cmdlet **`Write-S3BucketTagging`**. Ele aplica etiquetas Stage:Production e Department:Finance ao balde. Observe que **`Write-S3BucketTagging`** substitui todo o conjunto de tags existente em um bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet
 @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Para obter detalhes sobre a API, consulte PutBucketTagging Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3BucketVersioning`

O código de exemplo a seguir mostra como usar `Write-S3BucketVersioning`.

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita o versionamento do bucket do S3 em questão.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -VersioningConfig_Status
 Enabled
```

- Para obter detalhes sobre a API, consulte PutBucketVersioning Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3BucketWebsite`

O código de exemplo a seguir mostra como usar `Write-S3BucketWebsite`.

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita a hospedagem de sites para o bucket em questão com o documento de índice como “index.html” e o documento de erro como “error.html”.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Para obter detalhes sobre a API, consulte PutBucketWebsite Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3LifecycleConfiguration`

O código de exemplo a seguir mostra como usar `Write-S3LifecycleConfiguration`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo grava/substitui a configuração fornecida no $NewRule. Essa configuração garante limitar os objetos do escopo com determinados valores de prefixo e tag.

```
$NewRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  @{
   Days=  50
  }
  Id =  "Test-From-Write-cmdlet-1"
  Filter=  @{
```

```
    LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
     Operands=  @(
      [Amazon.S3.Model.LifecyclePrefixPredicate] @{
       "Prefix" =  "py"
      },
      [Amazon.S3.Model.LifecycleTagPredicate] @{
       "Tag"=  @{
        "Key" =  "non-use"
        "Value" = "yes"
       }
      }
     )
    }
   }
   "Status"= 'Enabled'
   NoncurrentVersionExpiration = @{
    NoncurrentDays = 75
   }
  }

  Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
  $NewRule
```

Exemplo 2: Esse exemplo define várias regras com filtragem. $ ArchiveRule define os objetos para arquivar em 30 dias no Glacier e 120 no. DeepArchive $ ExpireRule expira as versões atual e anterior em 150 dias para objetos com o prefixo 'py' e tag:key 'archieved' definido como 'yes'.

```
$ExpireRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  @{
   Days=  150
  }
  Id =  "Remove-in-150-days"
  Filter=  @{
   LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
    Operands=  @(
     [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" =  "py"
     },
     [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"=  @{
       "Key" =  "archived"
       "Value" = "yes"
      }
```

```
     }
    )
   }
  }
  Status= 'Enabled'
  NoncurrentVersionExpiration = @{
   NoncurrentDays = 150
  }
 }

 $ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  $null
  Id =  "Archive-to-Glacier-in-30-days"
  Filter=  @{
   LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
    Operands= @(
     [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" =  "py"
     },
     [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"=  @{
       "Key" =  "reviewed"
       "Value" = "yes"
      }
     }
    )
   }
  }
  Status = 'Enabled'
  NoncurrentVersionExpiration = @{
   NoncurrentDays = 75
  }
  Transitions = @(
   @{
    Days = 30
    "StorageClass"= 'Glacier'
   },
   @{
    Days = 120
    "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
   }
  )
 }
```

```
Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
 $ExpireRule,$ArchiveRule
```

- Para obter detalhes sobre a API, consulte PutLifecycleConfiguration Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3Object`

O código de exemplo a seguir mostra como usar `Write-S3Object`.

Ferramentas para PowerShell V5

Exemplo 1: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Exemplo 2: este comando carrega o arquivo único “sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”. Quando o parâmetro -Key não é fornecido, usa-se o nome do arquivo como chave do objeto do S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Exemplo 3: Esse comando carrega o arquivo único "local-sample.txt" para o Amazon S3, criando um objeto com a chave "prefix/to/sample.txt" no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .
\local-sample.txt
```

Exemplo 4: Esse comando carrega todos os arquivos no subdiretório “Scripts” para o bucket “test-files” e aplica o prefixo de chave comum "" SampleScripts a cada objeto. Cada arquivo enviado terá uma chave de "SampleScripts/filename", em que o 'nome do arquivo' varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

Exemplo 5: Esse comando carrega todos os arquivos*.ps1 no diretório local “Scripts” para o bucket “test-files” e aplica o prefixo de chave comum "" a cada objeto. SampleScripts Cada

arquivo enviado terá uma chave de "SampleScripts/filename.ps1", em que o 'nome do arquivo' varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

Exemplo 6: este comando cria um objeto do S3 contendo a string de conteúdo especificada com a chave “sample.txt”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

Exemplo 7: este comando carrega o arquivo especificado (o nome do arquivo é usado como chave) e aplica as tags especificadas ao novo objeto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Exemplo 8: este comando carrega recursivamente a pasta especificada e aplica as tags especificadas a todos os novos objetos.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles" -
Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Para obter detalhes sobre a API, consulte PutObject Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-S3ObjectRetention`

O código de exemplo a seguir mostra como usar `Write-S3ObjectRetention`.

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita o modo de retenção de governança até a data “31st Dec 2019 00:00:00” para o objeto “testfile.txt” no bucket do S3 em questão.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Para obter detalhes sobre a API, consulte PutObjectRetention Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de CSPM do Security Hub usando ferramentas para V5 PowerShell

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com Security Hub CSPM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-SHUBFinding`

O código de exemplo a seguir mostra como usar `Get-SHUBFinding`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando recupera as descobertas do Security Hub do serviço do Amazon EC2.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.ResourceType = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'PREFIX'
    Value = 'AwsEc2'
}
Get-SHUBFinding -Filter $filter
```

Exemplo 2: Esse comando recupera as descobertas do Security Hub da ID da AWS conta 123456789012.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.AwsAccountId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'EQUALS'
    Value = '123456789012'
}
Get-SHUBFinding -Filter $filter
```

Exemplo 3: esse comando recupera as descobertas do Security Hub geradas para o padrão “pci-dss”.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.GeneratorId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'PREFIX'
    Value = 'pci-dss'
}
Get-SHUBFinding -Filter $filter
```

Exemplo 4: esse comando recupera descobertas com gravidade crítica do Security Hub que têm um status de fluxo de trabalho de NOTIFIED.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.SeverityLabel = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'EQUALS'
    Value = 'CRITICAL'
}
$filter.WorkflowStatus = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter
 -Property @{
    Comparison = 'EQUALS'
    Value = 'NOTIFIED'
}
Get-SHUBFinding -Filter $filter
```

- Para obter detalhes sobre a API, consulte GetFindings Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon SES usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Get-SESIdentity`

O código de exemplo a seguir mostra como usar `Get-SESIdentity`.

Ferramentas para PowerShell V5

Exemplo 1: Esse comando retorna uma lista contendo todas as identidades (endereços de e-mail e domínios) de uma AWS conta específica, independentemente do status da verificação.

```
Get-SESIdentity
```

- Para obter detalhes sobre a API, consulte ListIdentities Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SESSendQuota`

O código de exemplo a seguir mostra como usar `Get-SESSendQuota`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando retorna os limites de envio atuais do usuário.

```
Get-SESSendQuota
```

- Para obter detalhes sobre a API, consulte GetSendQuota Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SESSendStatistic`

O código de exemplo a seguir mostra como usar `Get-SESSendStatistic`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando retorna as estatísticas de envio do usuário. O resultado é uma lista de pontos de dados que representa as duas últimas semanas de atividades de envio. Cada ponto de dados na lista contém estatísticas para um intervalo de 15 minutos.

```
Get-SESSendStatistic
```

- Para obter detalhes sobre a API, consulte GetSendStatistics Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos da API Amazon SES v2 usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com a API Amazon SES v2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Send-SES2Email`

O código de exemplo a seguir mostra como usar `Send-SES2Email`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como enviar uma mensagem de e-mail padrão.

```
Send-SES2Email -FromEmailAddress "sender@example.com" -Destination_ToAddress
 "recipient@example.com" -Subject_Data "Email Subject" -Text_Data "Email Body"
```

- Para obter detalhes sobre a API, consulte SendEmail Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon SNS usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Publish-SNSMessage`

O código de exemplo a seguir mostra como usar `Publish-SNSMessage`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo mostra a publicação de uma mensagem com uma única mensagem em linha MessageAttribute declarada.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute
```

```
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
 StringValue ='AnyCity'}}
```

Exemplo 2: Este exemplo mostra a publicação de uma mensagem com várias MessageAttributes declaradas com antecedência.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute $messageAttributes
```

- Consulte detalhes da API em Publish na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Exemplos do Amazon SQS usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-SQSPermission`

O código de exemplo a seguir mostra como usar `Add-SQSPermission`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo permite que Conta da AWS o especificado envie mensagens da fila especificada.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label
 SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Para obter detalhes sobre a API, consulte AddPermission Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Clear-SQSQueue`

O código de exemplo a seguir mostra como usar `Clear-SQSQueue`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui todas as mensagens da fila especificada.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes sobre a API, consulte PurgeQueue Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-SQSMessageVisibility`

O código de exemplo a seguir mostra como usar `Edit-SQSMessageVisibility`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o tempo limite de visibilidade da mensagem com o identificador de recebimento especificado na fila especificada para 10 horas (1 hora x 60 minutos x 60 segundos = 36.000 segundos).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/
MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- Para obter detalhes sobre a API, consulte ChangeMessageVisibility Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-SQSMessageVisibilityBatch`

O código de exemplo a seguir mostra como usar `Edit-SQSMessageVisibilityBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o tempo limite de visibilidade para duas mensagens com os identificadores de recebimento especificados na fila especificada. O tempo limite de visibilidade da primeira mensagem é alterado para 10 horas (10 horas x 60 minutos x 60 segundos = 36.000 segundos). O tempo limite de visibilidade da segunda mensagem é alterado para 5 horas (5 horas x 60 minutos x 60 segundos = 18.000 segundos).

```
$changeVisibilityRequest1 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
 $changeVisibilityRequest2
```

Saída:

```
Failed    Successful

------    ----------
```

```
{}        {Request2, Request1}
```

- Para obter detalhes sobre a API, consulte ChangeMessageVisibilityBatch Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SQSDeadLetterSourceQueue`

O código de exemplo a seguir mostra como usar `Get-SQSDeadLetterSourceQueue`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os URLs de todas as filas que dependem da fila especificada como a fila de mensagens não entregues.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Para obter detalhes sobre a API, consulte ListDeadLetterSourceQueues Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SQSQueue`

O código de exemplo a seguir mostra como usar `Get-SQSQueue`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as filas.

```
Get-SQSQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Exemplo 2: esse exemplo lista qualquer fila que comece com o nome especificado.

```
Get-SQSQueue -QueueNamePrefix My
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Para obter detalhes sobre a API, consulte ListQueues Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SQSQueueAttribute`

O código de exemplo a seguir mostra como usar `Get-SQSQueueAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os atributos da fila especificada.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
```

```
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14
                                        495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                        398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                        arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                        [ApproximateNumberOfMessagesNotVisible, 0],
 [ApproximateNumberOfMessagesDelayed, 0]...}
```

Exemplo 2: esse exemplo lista separadamente somente os atributos especificados da fila especificada.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14
                                        495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
```

```
                                        398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                        arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- Para obter detalhes sobre a API, consulte GetQueueAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SQSQueueUrl`

O código de exemplo a seguir mostra como usar `Get-SQSQueueUrl`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista o URL da fila com o nome especificado.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes sobre a API, consulte GetQueueUrl Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-SQSQueue`

O código de exemplo a seguir mostra como usar `New-SQSQueue`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma fila com o nome especificado.

```
New-SQSQueue -QueueName MyQueue
```

Saída:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes sobre a API, consulte CreateQueue Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Receive-SQSMessage`

O código de exemplo a seguir mostra como usar `Receive-SQSMessage`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as informações de até as próximas 10 mensagens a serem recebidas na fila especificada. As informações conterão valores para os atributos de mensagem especificados, se existirem.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
Attributes             : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
 1451495923744]}
Body                   : Information about John Doe's grade.
MD5OfBody              : ea572796e3c231f974fe75d89EXAMPLE
MD5OfMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==
```

- Para obter detalhes sobre a API, consulte ReceiveMessage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SQSMessage`

O código de exemplo a seguir mostra como usar `Remove-SQSMessage`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a mensagem com os identificadores de recebimento especificados na fila especificada.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
 -ReceiptHandle AQEBd329...v6gl8Q==
```

- Para obter detalhes sobre a API, consulte DeleteMessage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SQSMessageBatch`

O código de exemplo a seguir mostra como usar `Remove-SQSMessageBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui duas mensagens com os identificadores de recebimento especificados na fila especificada.

```
$deleteMessageRequest1 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBqOVY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $deleteMessageRequest1, $deleteMessageRequest2
```

Saída:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- Para obter detalhes sobre a API, consulte DeleteMessageBatch Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SQSPermission`

O código de exemplo a seguir mostra como usar `Remove-SQSPermission`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove as configurações permissão com o rótulo especificado da fila especificada.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes sobre a API, consulte RemovePermission Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SQSQueue`

O código de exemplo a seguir mostra como usar `Remove-SQSQueue`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a fila especificada.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes sobre a API, consulte DeleteQueue Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Send-SQSMessage`

O código de exemplo a seguir mostra como usar `Send-SQSMessage`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo envia uma mensagem com os atributos e o corpo da mensagem especificados para a fila especificada com a entrega da mensagem atrasada por 10 segundos.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
```

```
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl https://
sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Saída:

```
MD5OfMessageAttributes              MD5OfMessageBody                    MessageId

----------------------              ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE    51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- Para obter detalhes sobre a API, consulte SendMessage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Send-SQSMessageBatch`

O código de exemplo a seguir mostra como usar `Send-SQSMessageBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo envia duas mensagens com os atributos especificados e corpos de mensagem para a fila especificada. A entrega é adiada por 15 segundos para a primeira mensagem e 10 segundos para a segunda mensagem.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
```

```
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"

$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

Saída:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- Para obter detalhes sobre a API, consulte SendMessageBatch Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Set-SQSQueueAttribute`

O código de exemplo a seguir mostra como usar `Set-SQSQueueAttribute`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra como definir uma política para assinar um tópico do SNS com uma fila. Quando uma mensagem é publicada no tópico, ela é enviada à fila assinada.

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
      {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
          }
      }
    }
  ]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

Exemplo 2: esse exemplo define os atributos especificados da fila especificada.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Para obter detalhes sobre a API, consulte SetQueueAttributes Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS STS exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Convert-STSAuthorizationMessage`

O código de exemplo a seguir mostra como usar `Convert-STSAuthorizationMessage`.

Ferramentas para PowerShell V5

Exemplo 1: decodifica as informações adicionais contidas no conteúdo da mensagem codificada fornecida que foi retornada em resposta a uma solicitação. As informações adicionais são codificadas porque os detalhes do status da autorização podem constituir informações privilegiadas que o usuário responsável por solicitar a ação não deve ver.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Para obter detalhes sobre a API, consulte DecodeAuthorizationMessage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-STSFederationToken`

O código de exemplo a seguir mostra como usar `Get-STSFederationToken`.

Ferramentas para PowerShell V5

Exemplo 1: solicita um token federado válido por uma hora usando "Bob" como nome do usuário federado. Esse nome pode ser usado para referenciar o nome do usuário federado em uma política baseada em recursos (como uma política de bucket do Amazon S3). A política do IAM fornecida, no formato JSON, é usada para definir o escopo das permissões que estão disponíveis para o usuário do IAM. A política fornecida não pode conceder mais permissões do que as concedidas ao usuário solicitante, com as permissões finais do usuário federado sendo o conjunto mais restritivo com base na interseção da política aprovada com a política de usuário do IAM.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- Para obter detalhes sobre a API, consulte GetFederationToken Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-STSSessionToken`

O código de exemplo a seguir mostra como usar `Get-STSSessionToken`.

Ferramentas para PowerShell V5

Exemplo 1: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por um determinado período. As credenciais usadas para solicitar credenciais temporárias são inferidas dos padrões atuais do shell. Para especificar outras credenciais, use os parâmetros - ProfileName ou - AccessKey /-SecretKey .

```
Get-STSSessionToken
```

Saída:

```
AccessKeyId                             Expiration
 SecretAccessKey                        SessionToken
-----------                             ----------
 ---------------                        ------------
EXAMPLEACCESSKEYID                      2/16/2015 9:12:28 PM
 examplesecretaccesskey...              SamPleTokeN.....
```

Exemplo 2: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por uma hora. As credenciais usadas para fazer a solicitação são obtidas do perfil especificado.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Saída:

```
AccessKeyId                             Expiration
 SecretAccessKey                        SessionToken
-----------                             ----------
 ---------------                        ------------
EXAMPLEACCESSKEYID                      2/16/2015 9:12:28 PM
 examplesecretaccesskey...              SamPleTokeN.....
```

Exemplo 3: retorna uma instância **`Amazon.RuntimeAWSCredentials`** contendo credenciais temporárias válidas por uma hora usando o número de identificação do dispositivo de MFA associado à conta cujas credenciais estão especificadas no perfil 'myprofilename' e o valor fornecido pelo dispositivo.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Saída:

```
AccessKeyId                             Expiration
 SecretAccessKey                        SessionToken
-----------                             ----------
 ---------------                        ------------
EXAMPLEACCESSKEYID                      2/16/2015 9:12:28 PM
 examplesecretaccesskey...              SamPleTokeN.....
```

- Para obter detalhes sobre a API, consulte GetSessionToken Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Use-STSRole`

O código de exemplo a seguir mostra como usar `Use-STSRole`.

Ferramentas para PowerShell V5

Exemplo 1: retorna um conjunto de credenciais temporárias (chave de acesso, chave secreta e token de sessão) que podem ser usadas por uma hora para acessar AWS recursos aos quais o usuário solicitante normalmente não teria acesso. As credenciais retornadas têm as permissões permitidas pela política de acesso do perfil assumido e pela política fornecida (não é possível usar a política fornecida para conceder permissões além das definidas pela política de acesso do perfil que está sendo assumido).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

Exemplo 2: retorna um conjunto de credenciais temporárias, válidas por uma hora, que têm as mesmas permissões definidas na política de acesso do perfil que está sendo assumido.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

Exemplo 3: retorna um conjunto de credenciais temporárias que fornecem o número de série e o token gerado de uma MFA associada às credenciais do usuário usadas para executar o cmdlet.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Exemplo 4: retorna um conjunto de credenciais temporárias que assumiram um perfil definido em uma conta de cliente. Para cada função que o terceiro pode assumir, a conta do cliente deve criar uma função usando um identificador que deve ser passado no ExternalId parâmetro - sempre que a função for assumida.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- Para obter detalhes sobre a API, consulte AssumeRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Use-STSWebIdentityRole`

O código de exemplo a seguir mostra como usar `Use-STSWebIdentityRole`.

Ferramentas para PowerShell V5

Exemplo 1: retorna um conjunto temporário de credenciais, válido por uma hora, para um usuário que foi autenticado com o provedor de identidade Login with Amazon. As credenciais assumem a política de acesso associada ao perfil identificado pelo ARN do perfil. Opcionalmente, você pode transmitir uma política JSON ao parâmetro -Policy que refina ainda mais as permissões de acesso (não é possível conceder mais permissões do que as disponíveis nas permissões associadas ao perfil). O valor fornecido para o - WebIdentityToken é o identificador de usuário exclusivo que foi retornado pelo provedor de identidade.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Para obter detalhes sobre a API, consulte AssumeRoleWithWebIdentity Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Suporte exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com Suporte.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-ASACommunicationToCase`

O código de exemplo a seguir mostra como usar `Add-ASACommunicationToCase`.

Ferramentas para PowerShell V5

Exemplo 1: adiciona o corpo de uma comunicação por e-mail ao caso especificado.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Exemplo 2: adiciona o corpo de uma comunicação por e-mail ao caso especificado mais um ou mais endereços de e-mail contidos na linha CC do e-mail.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- Para obter detalhes sobre a API, consulte AddCommunicationToCase Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASACase`

O código de exemplo a seguir mostra como usar `Get-ASACase`.

Ferramentas para PowerShell V5

Exemplo 1: retorna os detalhes de todos os casos de suporte.

```
Get-ASACase
```

Exemplo 2: retorna os detalhes de todos os casos de suporte desde a data e a hora especificadas.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Exemplo 3: retorna os detalhes dos primeiros 10 casos de suporte, incluindo aqueles que foram resolvidos.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Exemplo 4: retorna os detalhes do único caso de suporte especificado.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemplo 5: retorna os detalhes dos casos de suporte especificados.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

- Para obter detalhes sobre a API, consulte DescribeCases Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASACommunication`

O código de exemplo a seguir mostra como usar `Get-ASACommunication`.

Ferramentas para PowerShell V5

Exemplo 1: retorna todas as comunicações do caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Exemplo 2: retorna todas as comunicações desde a meia-noite UTC de 1.º de janeiro de 2012 para o caso especificado.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

- Para obter detalhes sobre a API, consulte DescribeCommunications Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASAService`

O código de exemplo a seguir mostra como usar `Get-ASAService`.

Ferramentas para PowerShell V5

Exemplo 1: retorna todos os códigos de serviço, nomes e categorias disponíveis.

```
Get-ASAService
```

Exemplo 2: retorna o nome e as categorias do serviço com o código especificado.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Exemplo 3: retorna o nome e as categorias dos códigos de serviço especificados.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Exemplo 4: retorna o nome e as categorias (em japonês) dos códigos de serviço especificados. Atualmente, os códigos de idioma inglês (“en”) e japonês (“ja”) são compatíveis.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- Para obter detalhes sobre a API, consulte DescribeServices Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASASeverityLevel`

O código de exemplo a seguir mostra como usar `Get-ASASeverityLevel`.

Ferramentas para PowerShell V5

Exemplo 1: retorna a lista de níveis de gravidade que podem ser atribuídos a um caso de AWS suporte.

```
Get-ASASeverityLevel
```

Exemplo 2: retorna a lista de níveis de gravidade que podem ser atribuídos a um caso de AWS suporte. Os nomes dos níveis são retornados em japonês.

```
Get-ASASeverityLevel -Language "ja"
```

- Para obter detalhes sobre a API, consulte DescribeSeverityLevels Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASATrustedAdvisorCheck`

O código de exemplo a seguir mostra como usar `Get-ASATrustedAdvisorCheck`.

Ferramentas para PowerShell V5

Exemplo 1: retorna a coleção de verificações do Trusted Advisor. Você deve especificar o parâmetro Language, que pode aceitar “en” para saída em inglês ou “ja” para saída em japonês.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Para obter detalhes sobre a API, consulte DescribeTrustedAdvisorChecks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASATrustedAdvisorCheckRefreshStatus`

O código de exemplo a seguir mostra como usar `Get-ASATrustedAdvisorCheckRefreshStatus`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o status atual das solicitações de atualização para as verificações especificadas. Request-ASATrustedAdvisorCheckRefresh pode ser usado para solicitar que as informações de status das verificações sejam atualizadas.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Para obter detalhes sobre a API, consulte DescribeTrustedAdvisorCheckRefreshStatuses Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASATrustedAdvisorCheckResult`

O código de exemplo a seguir mostra como usar `Get-ASATrustedAdvisorCheckResult`.

Ferramentas para PowerShell V5

Exemplo 1: retorna os resultados de uma verificação do Trusted Advisor. A lista de verificações disponíveis do Trusted Advisor pode ser obtida usando Get-ASATrustedAdvisorChecks. A saída é o status geral da verificação, a data e hora em que a verificação foi executada pela última vez e o ID de verificação exclusivo da verificação específica. Para que os resultados sejam exibidos em japonês, adicione o parâmetro -Language “ja”.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Para obter detalhes sobre a API, consulte DescribeTrustedAdvisorCheckResult Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-ASATrustedAdvisorCheckSummary`

O código de exemplo a seguir mostra como usar `Get-ASATrustedAdvisorCheckSummary`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o resumo mais recente da verificação especificada do Trusted Advisor.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Exemplo 2: retorna os resumos mais recentes das verificações especificadas do Trusted Advisor.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Para obter detalhes sobre a API, consulte DescribeTrustedAdvisorCheckSummaries Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-ASACase`

O código de exemplo a seguir mostra como usar `New-ASACase`.

Ferramentas para PowerShell V5

Exemplo 1: Cria um novo caso no Centro de AWS Suporte. Os valores dos CategoryCode parâmetros - ServiceCode e - podem ser obtidos usando o Get-ASAService cmdlet. O valor do SeverityCode parâmetro - pode ser obtido usando o Get-ASASeverityLevel cmdlet. O valor do IssueType parâmetro - pode ser “atendimento ao cliente” ou “técnico”. Se for bem-sucedido, o número do caso de AWS suporte será exibido. Por padrão, o caso será tratado em inglês. Para usar o japonês, adicione o parâmetro -Language “ja”. Os CommunicationBody parâmetros - ServiceCode, -CategoryCode, -Assunto e - são obrigatórios.

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode
 "low" -Subject "subject text" -CommunicationBody "description of the case" -
CcEmailAddress @("email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- Para obter detalhes sobre a API, consulte CreateCase Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Request-ASATrustedAdvisorCheckRefresh`

O código de exemplo a seguir mostra como usar `Request-ASATrustedAdvisorCheckRefresh`.

Ferramentas para PowerShell V5

Exemplo 1: solicita uma atualização para a verificação especificada do Trusted Advisor.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Para obter detalhes sobre a API, consulte RefreshTrustedAdvisorCheck Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Resolve-ASACase`

O código de exemplo a seguir mostra como usar `Resolve-ASACase`.

Ferramentas para PowerShell V5

Exemplo 1: retorna o estado inicial do caso especificado e o estado atual após a conclusão da chamada para resolvê-lo.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Para obter detalhes sobre a API, consulte ResolveCase Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Systems Manager usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a Ferramentas da AWS para PowerShell V5 com o Systems Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Add-SSMResourceTag`

O código de exemplo a seguir mostra como usar `Add-SSMResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza uma janela de manutenção com novas tags. Não haverá saída se o comando for bem-sucedido. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada tag. Não haverá saída se o comando for bem-sucedido.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- Para obter detalhes sobre a API, consulte AddTagsToResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-SSMDocumentPermission`

O código de exemplo a seguir mostra como usar `Edit-SSMDocumentPermission`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona permissões de "compartilhamento" a todas as contas para um documento. Não haverá saída se o comando for bem-sucedido.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Exemplo 2: esse exemplo adiciona permissões de "compartilhamento" a uma conta específica para um documento. Não haverá saída se o comando for bem-sucedido.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Para obter detalhes sobre a API, consulte ModifyDocumentPermission Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMActivation`

O código de exemplo a seguir mostra como usar `Get-SSMActivation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo fornece detalhes sobre as ativações em sua conta.

```
Get-SSMActivation
```

Saída:

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
RegistrationLimit   : 10
RegistrationsCount  : 0
```

- Para obter detalhes sobre a API, consulte DescribeActivations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAssociation`

O código de exemplo a seguir mostra como usar `Get-SSMAssociation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve a associação entre uma instância e um documento.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Para obter detalhes sobre a API, consulte DescribeAssociation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAssociationExecution`

O código de exemplo a seguir mostra como usar `Get-SSMAssociationExecution`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna as execuções para o ID de associação fornecido

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Saída:

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- Para obter detalhes sobre a API, consulte DescribeAssociationExecutions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAssociationExecutionTarget`

O código de exemplo a seguir mostra como usar `Get-SSMAssociationExecutionTarget`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe o ID do recurso e seu status de execução que fazem parte dos destinos de execução da associação

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Saída:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Exemplo 2: esse comando verifica a execução específica de uma automação específica desde ontem, onde documento de comandos está associado. Além disso, ele verifica se a execução da associação falhou e, em caso afirmativo, exibie os detalhes da invocação do comando para a execução junto com o ID da instância

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
```

```
        Get-SSMCommandInvocation -CommandId $execution.OutputSource.OutputSourceId -
Detail:$true | Select-Object -ExpandProperty CommandPlugins
    }
}
```

Saída:

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0 on
 i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
StatusDetails          : Failed
```

- Para obter detalhes sobre a API, consulte DescribeAssociationExecutionTargets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAssociationList`

O código de exemplo a seguir mostra como usar `Get-SSMAssociationList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as associações para uma instância. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Exemplo 2: esse exemplo lista todas as associações para um documento de configuração. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Exemplo 3: Com a PowerShell versão 2, você deve usar New-Object para criar cada filtro.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Saída:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
```

```
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- Para obter detalhes sobre a API, consulte ListAssociations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAssociationVersionList`

O código de exemplo a seguir mostra como usar `Get-SSMAssociationVersionList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera todas as versões da associação fornecida.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Saída:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
```

```
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- Para obter detalhes sobre a API, consulte ListAssociationVersions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAutomationExecution`

O código de exemplo a seguir mostra como usar `Get-SSMAutomationExecution`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe os detalhes de uma execução do Automation.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Saída:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU
                            gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-
q0CK0ezS3xfh_y0ilaUGOAZG-xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
```

```
                              6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                     : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                  : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions              : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Exemplo 2: esse exemplo lista os detalhes da etapa para o ID de execução do Automation fornecido

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Saída:

```
StepName                  Action                  StepStatus ValidNextSteps
--------                  ------                  ---------- --------------
LaunchInstance            aws:runInstances        Success    {OSCompatibilityCheck}
OSCompatibilityCheck      aws:runCommand          Success    {RunPreUpdateScript}
RunPreUpdateScript        aws:runCommand          Success    {UpdateEC2Config}
UpdateEC2Config           aws:runCommand          Cancelled  {}
UpdateSSMAgent            aws:runCommand          Pending    {}
UpdateAWSPVDriver         aws:runCommand          Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand          Pending    {}
UpdateAWSNVMe             aws:runCommand          Pending    {}
InstallWindowsUpdates     aws:runCommand          Pending    {}
RunPostUpdateScript       aws:runCommand          Pending    {}
RunSysprepGeneralize      aws:runCommand          Pending    {}
StopInstance              aws:changeInstanceState Pending    {}
CreateImage               aws:createImage         Pending    {}
TerminateInstance         aws:changeInstanceState Pending    {}
```

- Para obter detalhes sobre a API, consulte GetAutomationExecution Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAutomationExecutionList`

O código de exemplo a seguir mostra como usar `Get-SSMAutomationExecutionList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve todas as execuções do Automation ativas e encerradas associadas à sua conta.

```
Get-SSMAutomationExecutionList
```

Saída:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Exemplo 2: Este exemplo exibe executionID, documento e start/end timestamp de execução para execuções com AutomationExecutionStatus algo diferente de 'Sucesso'

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Saída:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
---------------------                ------------
 ------------------------- ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
```

```
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi                    Failed
                    4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- Para obter detalhes sobre a API, consulte DescribeAutomationExecutions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAutomationStepExecution`

O código de exemplo a seguir mostra como usar `Get-SSMAutomationStepExecution`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe informações sobre todas as execuções de etapas ativas e encerradas em um fluxo de trabalho do Automation.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Saída:

```
StepName                  Action                  StepStatus
--------                  ------                  ----------
LaunchInstance            aws:runInstances        Success
OSCompatibilityCheck      aws:runCommand          Success
RunPreUpdateScript        aws:runCommand          Success
UpdateEC2Config           aws:runCommand          Cancelled
UpdateSSMAgent            aws:runCommand          Pending
UpdateAWSPVDriver         aws:runCommand          Pending
UpdateAWSEnaNetworkDriver aws:runCommand          Pending
UpdateAWSNVMe             aws:runCommand          Pending
InstallWindowsUpdates     aws:runCommand          Pending
RunPostUpdateScript       aws:runCommand          Pending
RunSysprepGeneralize      aws:runCommand          Pending
StopInstance              aws:changeInstanceState Pending
CreateImage               aws:createImage         Pending
TerminateInstance         aws:changeInstanceState Pending
```

- Para obter detalhes sobre a API, consulte DescribeAutomationStepExecutions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMAvailablePatch`

O código de exemplo a seguir mostra como usar `Get-SSMAvailablePatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém todos os patches disponíveis para o Windows Server 2012 que apresentam gravidade MSRC crítica A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Saída:

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
Description    : A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control
                 over it. You can help protect your system by installing this update
 from Microsoft. After you install this update, you may have to
                 restart your system.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

Exemplo 2: Com a PowerShell versão 2, você deve usar New-Object para criar cada filtro.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
```

```
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Exemplo 3: Este exemplo busca todas as atualizações lançadas nos últimos 20 dias e aplicáveis aos produtos correspondentes WindowsServer a 2019

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
 Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
 Title
```

Saída:

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash Player
 for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows Server
 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- Para obter detalhes sobre a API, consulte DescribeAvailablePatches Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMCommand`

O código de exemplo a seguir mostra como usar `Get-SSMCommand`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os comandos solicitados.

```
Get-SSMCommand
```

Saída:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
```

```
CompletedCount     : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Exemplo 2: esse exemplo obtém o status de um comando específico

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Exemplo 3: Este exemplo recupera todos os comandos SSM invocados após 01/04/2019 T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
 RequestedDateTime -Descending
```

Saída:

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled 4/16/2019
 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success   4/6/2019
 9:19:42 AM
```

```
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed    4/2/2019
 4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed    4/1/2019
 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success   4/1/2019
 1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed    4/1/2019
 11:13:09 AM
```

- Para obter detalhes sobre a API, consulte ListCommands Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMCommandInvocation`

O código de exemplo a seguir mostra como usar `Get-SSMCommandInvocation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as invocações de um comando.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
 $true
```

Saída:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :
```

Exemplo 2: Este exemplo lista CommandPlugins para invocação do comando id e1eb2e3c-ed4c-5123-45c1-234f5612345f

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Saída:

```
Name                   : aws:runPowerShellScript
Output                 : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                          kumo available

OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Para obter detalhes sobre a API, consulte ListCommandInvocations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMCommandInvocationDetail`

O código de exemplo a seguir mostra como usar `Get-SSMCommandInvocationDetail`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe os detalhes de um comando executado em uma instância.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Saída:

```
CommandId              : b8eac879-0541-439d-94ec-47a80d554f44
Comment                : IP config
DocumentName           : AWS-RunShellScript
ExecutionElapsedTime   : PT0.004S
```

```
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
StandardErrorUrl       :
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Para obter detalhes sobre a API, consulte GetCommandInvocation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMComplianceItemList`

O código de exemplo a seguir mostra como usar `Get-SSMComplianceItemList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra a lista de itens de conformidade para o ID e tipo de recurso fornecidos, filtrados pelo tipo de conformidade "Associação"

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Saída:

```
ComplianceType   : Association
Details          : {[DocumentName, AWS-GatherSoftwareInventory], [DocumentVersion,
 1]}
ExecutionSummary : Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id               : 123a45a1-c234-1234-1245-67891236db4e
ResourceId       : i-1a2caf345f67d0dc2
ResourceType     : ManagedInstance
Severity         : UNSPECIFIED
Status           : COMPLIANT
Title            :
```

- Para obter detalhes sobre a API, consulte ListComplianceItems Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMComplianceSummaryList`

O código de exemplo a seguir mostra como usar `Get-SSMComplianceSummaryList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo devolve uma contagem resumida de recursos em ou fora de conformidade para todos os tipos de conformidade.

```
Get-SSMComplianceSummaryList
```

Saída:

```
ComplianceType CompliantSummary
 NonCompliantSummary
-------------- ----------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- Para obter detalhes sobre a API, consulte ListComplianceSummaries Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMConnectionStatus`

O código de exemplo a seguir mostra como usar `Get-SSMConnectionStatus`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera o status de conexão do Gerenciador de Sessões de uma instância para determinar se ela está conectada e pronta para receber conexões do Gerenciador de Sessões.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Saída:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- Para obter detalhes sobre a API, consulte GetConnectionStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMDefaultPatchBaseline`

O código de exemplo a seguir mostra como usar `Get-SSMDefaultPatchBaseline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe a lista de referência de patches padrão.

```
Get-SSMDefaultPatchBaseline
```

Saída:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Para obter detalhes sobre a API, consulte GetDefaultPatchBaseline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMDeployablePatchSnapshotForInstance`

O código de exemplo a seguir mostra como usar `Get-SSMDeployablePatchSnapshotForInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe o instantâneo atual da lista de referência de patches usada por uma instância. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que use as credenciais da instância, o exemplo passa um objeto **`Amazon.Runtime.InstanceProfileAWSCredentials`** para o parâmetro Credentials.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
```

```
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Saída:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Exemplo 2: Este exemplo mostra como obter a versão completa SnapshotDownloadUrl. Esse comando deve ser executado da instância usando as credenciais da instância. Para garantir que ela use as credenciais da instância, o exemplo configura a PowerShell sessão para usar um **`Amazon.Runtime.InstanceProfileAWSCredentials`** objeto.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Saída:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Para obter detalhes sobre a API, consulte GetDeployablePatchSnapshotForInstance Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMDocument`

O código de exemplo a seguir mostra como usar `Get-SSMDocument`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna o conteúdo de um documento.

```
Get-SSMDocument -Name "RunShellScript"
```

Saída:

```
Content
-------
{...
```

Exemplo 2: esse exemplo exibe o conteúdo completo de um documento.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      }
   ]
}
```

- Para obter detalhes sobre a API, consulte GetDocument Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMDocumentDescription`

O código de exemplo a seguir mostra como usar `Get-SSMDocumentDescription`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna informações sobre um documento.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Saída:

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            : f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- Para obter detalhes sobre a API, consulte DescribeDocument Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMDocumentList`

O código de exemplo a seguir mostra como usar `Get-SSMDocumentList`.

Ferramentas para PowerShell V5

Exemplo 1: lista todos os documentos de configuração em sua conta.

```
Get-SSMDocumentList
```

Saída:

```
DocumentType    : Command
```

```
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Exemplo 2: esse exemplo recupera todos os documentos de automação com o nome correspondente a "Plataform"

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
 Where-Object Name -Match "Platform"
```

Saída:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- Para obter detalhes sobre a API, consulte ListDocuments Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMDocumentPermission`

O código de exemplo a seguir mostra como usar `Get-SSMDocumentPermission`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as versões de um documento.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Saída:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- Para obter detalhes sobre a API, consulte DescribeDocumentPermission Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMDocumentVersionList`

O código de exemplo a seguir mostra como usar `Get-SSMDocumentVersionList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as versões de um documento.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

Saída:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
Status            : Active
```

- Para obter detalhes sobre a API, consulte ListDocumentVersions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMEffectiveInstanceAssociationList`

O código de exemplo a seguir mostra como usar `Get-SSMEffectiveInstanceAssociationList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo descreve as associações efetivas de uma instância.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
 5
```

Saída:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Exemplo 2: esse exemplo exibe o conteúdo das associações efetivas de uma instância.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Saída:

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or specified
 version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent to
 install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "allowDowngrade": {
            "default": "false",
            "description": "(Optional) Allow the Amazon SSM Agent service to be
 downgraded to an earlier version. If set
```

```
 to false, the service can be upgraded to newer versions only (default). If set to
 true, specify the earlier version.",
            "type": "String",
            "allowedValues": [
                "true",
                "false"
            ]
        }
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}
```

- Para obter detalhes sobre a API, consulte DescribeEffectiveInstanceAssociations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMEffectivePatchesForPatchBaseline`

O código de exemplo a seguir mostra como usar `Get-SSMEffectivePatchesForPatchBaseline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra todas as listas de referência de patches, com uma lista de resultados máxima de 1.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Saída:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Exemplo 2: esse exemplo mostra o status do patch para todas as listas de referência de patches, com uma lista de resultados máxima de 1.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Saída:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- Para obter detalhes sobre a API, consulte DescribeEffectivePatchesForPatchBaseline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInstanceAssociationsStatus`

O código de exemplo a seguir mostra como usar `Get-SSMInstanceAssociationsStatus`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra detalhes das associações de uma instância.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Saída:

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
Name             : AWS-UpdateSSMAgent
```

```
OutputUrl        :
Status           : Pending
```

Exemplo 2: esse exemplo verifica o status da associação da instância para o ID da instância fornecido e exibe o status de execução dessas associações

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
 {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Saída:

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status                : Success
```

- Para obter detalhes sobre a API, consulte DescribeInstanceAssociationsStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInstanceInformation`

O código de exemplo a seguir mostra como usar `Get-SSMInstanceInformation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra detalhes de cada uma de suas instâncias.

```
Get-SSMInstanceInformation
```

Saída:

```
ActivationId                          :
AgentVersion                          : 2.0.672.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : ip-172-31-44-222.us-west-2.compute.internal
```

```
IamRole                                 :
InstanceId                              : i-0cb2b964d3e14fd9f
IPAddress                               : 172.31.44.222
IsLatestVersion                         : True
LastAssociationExecutionDate            : 2/24/2017 3:18:09 AM
LastPingDateTime                        : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                    :
PingStatus                              : ConnectionLost
PlatformName                            : Amazon Linux AMI
PlatformType                            : Linux
PlatformVersion                         : 2016.09
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Exemplo 2: Esse exemplo mostra como usar o parâmetro -Filter para filtrar os resultados somente para as instâncias do AWS Systems Manager na região **`us-east-1`** com um **`AgentVersion`** de**`2.2.800.0`**. Você pode encontrar uma lista de valores válidos da chave -Filter no tópico de referência da InstanceInformation API (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager -Type-InstanceInformation-ActivationId).

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Saída:

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEb0792d98ce
IPAddress                               : 10.0.0.01
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:02:50 AM
LastPingDateTime                        : 8/16/2018 7:40:27 PM
```

```
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance

ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEac7501d023
IPAddress                              : 10.0.0.02
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:00:20 AM
LastPingDateTime                       : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Exemplo 3: Esse exemplo mostra como usar o InstanceInformationFilterList parâmetro - para filtrar os resultados somente para as instâncias do AWS Systems Manager na região **`us-east-1`** com **`PlatformTypes`** of **`Windows`** ou**`Linux`**. Você pode encontrar uma lista de InstanceInformationFilterList valores-chave válidos no tópico de referência da InstanceInformationFilter API (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList $Filters
```

Saída:

```
ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEb0792d98ce
IPAddress                              : 10.0.0.27
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:02:50 AM
LastPingDateTime                       : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Ubuntu Server 18.04 LTS
PlatformType                           : Linux
PlatformVersion                        : 18.04
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance

ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEac7501d023
IPAddress                              : 10.0.0.100
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:00:20 AM
LastPingDateTime                       : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Exemplo 4: Este exemplo lista instâncias e exportações InstanceId gerenciadas por ssm LastPingDateTime e PlatformName em um arquivo csv. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
 PlatformName | Export-Csv Instance-details.csv -NoTypeInformation
```

- Para obter detalhes sobre a API, consulte DescribeInstanceInformation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInstancePatch`

O código de exemplo a seguir mostra como usar `Get-SSMInstancePatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os detalhes de conformidade do patch para uma instância.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Para obter detalhes sobre a API, consulte DescribeInstancePatches Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInstancePatchState`

O código de exemplo a seguir mostra como usar `Get-SSMInstancePatchState`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os estados resumidos de patches para uma instância.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Exemplo 2: esse exemplo obtém os estados resumidos de patches para duas instâncias.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Para obter detalhes sobre a API, consulte DescribeInstancePatchStates Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInstancePatchStatesForPatchGroup`

O código de exemplo a seguir mostra como usar `Get-SSMInstancePatchStatesForPatchGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os estados de resumo de patches por instância de um grupo de patches.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Para obter detalhes sobre a API, consulte DescribeInstancePatchStatesForPatchGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInventory`

O código de exemplo a seguir mostra como usar `Get-SSMInventory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os metadados personalizados do seu inventário.

```
Get-SSMInventory
```

Saída:

```
Data
  Id
----
  --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Para obter detalhes sobre a API, consulte GetInventory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInventoryEntriesList`

O código de exemplo a seguir mostra como usar `Get-SSMInventoryEntriesList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as entradas de inventário personalizadas para uma instância.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Saída:

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Exemplo 2: esse exemplo lista os detalhes.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Saída:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Para obter detalhes sobre a API, consulte ListInventoryEntries Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInventoryEntryList`

O código de exemplo a seguir mostra como usar `Get-SSMInventoryEntryList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo recupera entradas de inventário do tipo **`AWS:Network`** da instância.

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
 Select-Object -ExpandProperty Entries
```

Saída:

```
Key        Value
---        -----
DHCPServer 172.31.11.2
DNSServer  172.31.0.1
Gateway    172.31.11.2
IPV4       172.31.11.222
IPV6       fe12::3456:7da8:901a:12a3
MacAddress 1A:23:4E:5B:FB:67
Name       Amazon Elastic Network Adapter
SubnetMask 255.255.240.0
```

- Para obter detalhes sobre a API, consulte Get-SSMInventoryEntryList Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMInventorySchema`

O código de exemplo a seguir mostra como usar `Get-SSMInventorySchema`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo retorna uma lista de nomes de tipos de inventário para a conta.

```
Get-SSMInventorySchema
```

- Para obter detalhes sobre a API, consulte GetInventorySchema Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMLatestEC2Image`

O código de exemplo a seguir mostra como usar `Get-SSMLatestEC2Image`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as AMIs do Windows mais recentes.

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

Saída:

```
Name                                                                Value
----                                                                -----
Windows_Server-2008-R2_SP1-English-64Bit-SQL_2012_SP4_Express
 ami-0e5ddd288daff4fab
Windows_Server-2012-R2_RTM-Chinese_Simplified-64Bit-Base
 ami-0c5ea64e6bec1cb50
Windows_Server-2012-R2_RTM-Chinese_Traditional-64Bit-Base
 ami-09775eff0bf8c113d
Windows_Server-2012-R2_RTM-Dutch-64Bit-Base
 ami-025064b67e28cf5df
...
```

Exemplo 2: esse exemplo recupera o ID da AMI de uma imagem específica do Amazon Linux para a região us-west-2.

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-
x86_64-ebs -Region us-west-2
```

Saída:

```
ami-09b92cd132204c704
```

Exemplo 3: esse exemplo lista todas as AMIs do Windows mais recentes que correspondem à expressão curinga especificada.

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

Saída:

```
Name                                                    Value
----                                                    -----
Windows_Server-2019-English-Full-SQL_2017_Web           ami-085e9d27da5b73a42
Windows_Server-2019-English-STIG-Core                   ami-0bfd85c29148c7f80
Windows_Server-2019-English-Full-SQL_2019_Web           ami-02099560d7fb11f20
Windows_Server-2019-English-Full-SQL_2016_SP2_Standard  ami-0d7ae2d81c07bd598
...
```

- Para obter detalhes sobre a API, consulte Get-SSMLatestEC2Image Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindow`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindow`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém detalhes sobre uma janela de manutenção.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Saída:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para obter detalhes sobre a API, consulte GetMaintenanceWindow Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindowExecution`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowExecution`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista informações sobre uma tarefa que foi executada como parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes sobre a API, consulte GetMaintenanceWindowExecution Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindowExecutionList`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowExecutionList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as execuções para uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Saída:

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Exemplo 2: esse exemplo lista todas as execuções para uma janela de manutenção antes de uma data especificada.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Exemplo 3: esse exemplo lista todas as execuções para uma janela de manutenção após uma data especificada.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- Para obter detalhes sobre a API, consulte DescribeMaintenanceWindowExecutions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindowExecutionTask`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowExecutionTask`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista informações sobre uma tarefa que fazia parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
 -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsMana
                    meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes sobre a API, consulte GetMaintenanceWindowExecutionTask Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindowExecutionTaskInvocationList`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowExecutionTaskInvocationList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as invocações de uma tarefa executada como parte da execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- Para obter detalhes sobre a API, consulte DescribeMaintenanceWindowExecutionTaskInvocations Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindowExecutionTaskList`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowExecutionTaskList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as tarefas associadas à execução de uma janela de manutenção.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Saída:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Para obter detalhes sobre a API, consulte DescribeMaintenanceWindowExecutionTasks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindowList`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as janelas de manutenção em sua conta.

```
Get-SSMMaintenanceWindowList
```

Saída:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- Para obter detalhes sobre a API, consulte DescribeMaintenanceWindows Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindowTarget`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowTarget`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os destinos para uma janela de manutenção.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- Para obter detalhes sobre a API, consulte DescribeMaintenanceWindowTargets Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMMaintenanceWindowTaskList`

O código de exemplo a seguir mostra como usar `Get-SSMMaintenanceWindowTaskList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todas as tarefas para uma janela de manutenção.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
LoggingInfo    :
MaxConcurrency : 1
```

```
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- Para obter detalhes sobre a API, consulte DescribeMaintenanceWindowTasks Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMParameterHistory`

O código de exemplo a seguir mostra como usar `Get-SSMParameterHistory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista o histórico de valores de um parâmetro.

```
Get-SSMParameterHistory -Name "Welcome"
```

Saída:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- Para obter detalhes sobre a API, consulte GetParameterHistory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMParameterList`

O código de exemplo a seguir mostra como usar `Get-SSMParameterList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista todos os parâmetros.

```
Get-SSMParameterList
```

Saída:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- Para obter detalhes sobre a API, consulte DescribeParameters Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMParameterValue`

O código de exemplo a seguir mostra como usar `Get-SSMParameterValue`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os valores de um parâmetro.

```
Get-SSMParameterValue -Name "Welcome"
```

Saída:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Exemplo 2: esse exemplo retorna os detalhes do valor.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Saída:

```
Name    Type   Value
```

```
----    ----   -----
Welcome String Good day, Sunshine!
```

- Para obter detalhes sobre a API, consulte GetParameters Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMPatchBaseline`

O código de exemplo a seguir mostra como usar `Get-SSMPatchBaseline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo mostra todas as listas de referência de patches.

```
Get-SSMPatchBaseline
```

Saída:

```
BaselineDescription                                                BaselineId
                                                       BaselineName
-------------------                                                ----------
                                                       ------------
Default Patch Baseline Provided by AWS.                            arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems pb-045f10b4f382baeda
                                                       Production-B...
Baseline containing all updates approved for production systems pb-0a2f1059b670ebd31
                                                       Production-B...
```

Exemplo 2: Este exemplo lista todas as linhas de base do patch fornecidas pelo. AWS A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Saída:

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemplo 3: esse exemplo mostra todas as listas de referência de patches pertencentes a você. A sintaxe usada neste exemplo requer a PowerShell versão 3 ou posterior.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Saída:

```
Get-SSMPatchBaseline -Filter $filter1
```

Exemplo 4: Com a PowerShell versão 2, você deve usar New-Object para criar cada tag.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Saída:

```
BaselineDescription                     BaselineId
                          BaselineName             DefaultBaselin

                                                   e
-------------------                     ----------
                          ------------             --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline True
```

- Para obter detalhes sobre a API, consulte DescribePatchBaselines Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMPatchBaselineDetail`

O código de exemplo a seguir mostra como usar `Get-SSMPatchBaselineDetail`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe os detalhes de uma lista de referência de patches.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Saída:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- Para obter detalhes sobre a API, consulte GetPatchBaseline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMPatchBaselineForPatchGroup`

O código de exemplo a seguir mostra como usar `Get-SSMPatchBaselineForPatchGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exibe a lista de referência de patches para um grupo de patches.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Para obter detalhes sobre a API, consulte GetPatchBaselineForPatchGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMPatchGroup`

O código de exemplo a seguir mostra como usar `Get-SSMPatchGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista os registros do grupo de patches.

```
Get-SSMPatchGroup
```

Saída:

```
BaselineIdentity                                          PatchGroup
----------------                                          ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Para obter detalhes sobre a API, consulte DescribePatchGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMPatchGroupState`

O código de exemplo a seguir mostra como usar `Get-SSMPatchGroupState`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém o resumo da conformidade de patches de alto nível para um grupo de patches.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Saída:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- Para obter detalhes sobre a API, consulte DescribePatchGroupState Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMResourceComplianceSummaryList`

O código de exemplo a seguir mostra como usar `Get-SSMResourceComplianceSummaryList`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém uma contagem resumida em nível de recurso. O resumo inclui informações sobre status de conformidade e não conformidade e contagens detalhadas de gravidade de itens de conformidade para produtos que correspondem a "Windows10". Como o MaxResult padrão é 100 se o parâmetro não for especificado e esse valor não for válido, o MaxResult parâmetro será adicionado e o valor será definido como 50.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Para obter detalhes sobre a API, consulte ListResourceComplianceSummaries Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-SSMResourceTag`

O código de exemplo a seguir mostra como usar `Get-SSMResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo lista as tags para uma janela de manutenção.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Saída:

```
Key   Value
---   -----
Stack Production
```

- Para obter detalhes sobre a API, consulte ListTagsForResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-SSMActivation

O código de exemplo a seguir mostra como usar `New-SSMActivation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma instância gerenciada.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

Saída:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Para obter detalhes sobre a API, consulte CreateActivation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-SSMAssociation

O código de exemplo a seguir mostra como usar `New-SSMAssociation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa um documento de configuração a uma instância usando IDs de instância.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Associated
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
```

```
Status.AdditionalInfo :
```

Exemplo 2: esse exemplo associa um documento de configuração a uma instância usando destinos.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Exemplo 3: esse exemplo associa um documento de configuração a uma instância usando destinos e parâmetros.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName "CWConfiguration" -
Target $target -Parameter $params
```

Saída:

```
Name                  : Configure-CloudWatch
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Exemplo 4: esse exemplo cria uma associação com todas as instâncias na região, com **`AWS-GatherSoftwareInventory`**. Ele também fornece arquivos personalizados e locais de registro nos parâmetros a serem coletados

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Saída:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Para obter detalhes sobre a API, consulte CreateAssociation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-SSMAssociationFromBatch`

O código de exemplo a seguir mostra como usar `New-SSMAssociationFromBatch`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo associa um documento de configuração a várias instâncias. A saída retorna uma lista de operações bem e malsucedidas, se aplicável.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Saída:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Exemplo 2: esse exemplo mostrará os detalhes completos de uma operação bem-sucedida.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Para obter detalhes sobre a API, consulte CreateAssociationBatch Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-SSMDocument`

O código de exemplo a seguir mostra como usar `New-SSMDocument`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um documento na sua conta. O documento deve estar em formato JSON. Para obter mais informações sobre como escrever um documento de configuração, consulte Documento de configuração na Referência da API do SSM.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Saída:

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
```

```
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Creating
```

- Para obter detalhes sobre a API, consulte CreateDocument Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-SSMMaintenanceWindow

O código de exemplo a seguir mostra como usar `New-SSMMaintenanceWindow`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma nova janela de manutenção com o nome especificado que é executada às 16h toda terça-feiras por 4 horas, com um limite de 1 hora, e que permite destinos não associados.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Saída:

```
mw-03eb53e1ea7383998
```

- Para obter detalhes sobre a API, consulte CreateMaintenanceWindow Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

New-SSMPatchBaseline

O código de exemplo a seguir mostra como usar `New-SSMPatchBaseline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria uma lista de referência de patches que aprova patches, sete dias após serem lançados pela Microsoft, para instâncias gerenciadas que executam o Windows Server 2019 em um ambiente de produção.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description "Baseline
 containing all updates approved for production systems" -ApprovalRules_PatchRule
 $rule
```

Saída:

```
pb-0z4z6221c4296b23z
```

- Para obter detalhes sobre a API, consulte CreatePatchBaseline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-SSMDefaultPatchBaseline`

O código de exemplo a seguir mostra como usar `Register-SSMDefaultPatchBaseline`.

Ferramentas para PowerShell V5

Exemplo 1: este exemplo registra uma lista de referência de patches como a lista de referência de patches padrão.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Saída:

```
pb-03da896ca3b68b639
```

- Para obter detalhes sobre a API, consulte RegisterDefaultPatchBaseline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-SSMPatchBaselineForPatchGroup`

O código de exemplo a seguir mostra como usar `Register-SSMPatchBaselineForPatchGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma lista de referência de patches para um grupo de patches.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- Para obter detalhes sobre a API, consulte RegisterPatchBaselineForPatchGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-SSMTargetWithMaintenanceWindow`

O código de exemplo a seguir mostra como usar `Register-SSMTargetWithMaintenanceWindow`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma instância com uma janela de manutenção.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Saída:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Exemplo 2: esse exemplo registra várias instâncias com uma janela de manutenção.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Saída:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Exemplo 3: esse exemplo registra uma instância com uma janela de manutenção usando tags do EC2.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Saída:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Para obter detalhes sobre a API, consulte RegisterTargetWithMaintenanceWindow Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-SSMTaskWithMaintenanceWindow`

O código de exemplo a seguir mostra como usar `Register-SSMTaskWithMaintenanceWindow`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra uma tarefa com uma janela de manutenção usando um ID de instância. A saída é o ID da tarefa.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Saída:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemplo 2: esse exemplo registra uma tarefa com uma janela de manutenção usando um ID de destino. A saída é o ID da tarefa.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -TaskType
 "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Saída:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Exemplo 3: esse exemplo cria um objeto de parâmetro para o documento de comandos de execução **`AWS-RunPowerShellScript`** e cria uma tarefa com uma janela de manutenção determinada usando o ID de destino. A saída devolvida é o ID da tarefa.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target = @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Saída:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Exemplo 4: Este exemplo registra uma tarefa de automação do AWS Systems Manager usando um documento chamado**`Create-Snapshots`**.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
```

```
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- Para obter detalhes sobre a API, consulte RegisterTaskWithMaintenanceWindow Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SSMActivation`

O código de exemplo a seguir mostra como usar `Remove-SSMActivation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui uma ativação. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Para obter detalhes sobre a API, consulte DeleteActivation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SSMAssociation`

O código de exemplo a seguir mostra como usar `Remove-SSMAssociation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui a associação entre uma instância e um documento. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- Para obter detalhes sobre a API, consulte DeleteAssociation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SSMDocument`

O código de exemplo a seguir mostra como usar `Remove-SSMDocument`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui um documento. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Para obter detalhes sobre a API, consulte DeleteDocument Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SSMMaintenanceWindow`

O código de exemplo a seguir mostra como usar `Remove-SSMMaintenanceWindow`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove uma janela de manutenção.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Saída:

```
mw-06d59c1a07c022145
```

- Para obter detalhes sobre a API, consulte DeleteMaintenanceWindow Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SSMParameter`

O código de exemplo a seguir mostra como usar `Remove-SSMParameter`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMParameter -Name "helloWorld"
```

- Para obter detalhes sobre a API, consulte DeleteParameter Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SSMPatchBaseline`

O código de exemplo a seguir mostra como usar `Remove-SSMPatchBaseline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui uma lista de referência de patches.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Saída:

```
pb-045f10b4f382baeda
```

- Para obter detalhes sobre a API, consulte DeletePatchBaseline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-SSMResourceTag`

O código de exemplo a seguir mostra como usar `Remove-SSMResourceTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove uma tag de uma janela de manutenção. Não haverá saída se o comando for bem-sucedido.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- Para obter detalhes sobre a API, consulte RemoveTagsFromResource Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Send-SSMCommand`

O código de exemplo a seguir mostra como usar `Send-SSMCommand`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo executa um comando echo em uma instância de destino.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Saída:

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Exemplo 2: esse exemplo mostra como executar um comando que aceita parâmetros aninhados.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- Para obter detalhes sobre a API, consulte SendCommand Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-SSMAutomationExecution`

O código de exemplo a seguir mostra como usar `Start-SSMAutomationExecution`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo executa um documento especificando um perfil do Automation, um ID de origem da AMI e um perfil de instância do Amazon EC2.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Saída:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Para obter detalhes sobre a API, consulte StartAutomationExecution Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Start-SSMSession`

O código de exemplo a seguir mostra como usar `Start-SSMSession`.

Ferramentas para PowerShell V5

Exemplo 1: Este exemplo inicia uma conexão com um destino para uma sessão do Session Manager, habilitando o encaminhamento de portas. Observação: você deverá adicionar o parâmetro Region, se ainda não estiver preparado para usar o AWSCredentials.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' } -Region 'us-west-1'
```

Saída:

```
Starting session with SessionId: testUser-xi4glew849asyeryde34u4dfsdfy
Port 8080 opened for sessionId testUser-xi4glew849asyeryde34u4dfsdfy.
Waiting for connections...
```

Exemplo 2: esse exemplo cria uma sessão interativa com uma instância especificada para uma sessão do Session Manager.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1'
```

Saída:

```
Starting session with SessionId : testUser-xi4glew849asyeryde34u4dfsdfy
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.



Install the latest PowerShell for new features and improvements!


PS C:\Windows\system32> whoami
ec2amaz-fnsdrwv\ec2-test-user
PS C:\Windows\system32>
```

Exemplo 3: Este exemplo cria uma sessão sem se conectar a ela e retorna as TokenValue propriedades SessionId StreamUrl,, e necessárias para se conectar à sessão.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1' -
DisablePluginInvocation
```

Saída:

```
SessionId        : testUser-xi4glew849asyeryde34u4dfsdfy
StreamUrl        : {StreamUrl value redacted}
TokenValue       : {Token value redacted}
ContentLength    : 1207
HttpStatusCode   : OK
```

- Para obter detalhes sobre a API, consulte StartSession Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-SSMAutomationExecution`

O código de exemplo a seguir mostra como usar `Stop-SSMAutomationExecution`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo interrompe uma execução do Automation. Não haverá saída se o comando for bem-sucedido.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- Para obter detalhes sobre a API, consulte StopAutomationExecution Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-SSMCommand`

O código de exemplo a seguir mostra como usar `Stop-SSMCommand`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo tenta cancelar um comando. Não haverá saída se a operação for bem-sucedida.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Para obter detalhes sobre a API, consulte CancelCommand Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-SSMManagedInstance`

O código de exemplo a seguir mostra como usar `Unregister-SSMManagedInstance`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro de uma instância gerenciada. Não haverá saída se o comando for bem-sucedido.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Para obter detalhes sobre a API, consulte DeregisterManagedInstance Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-SSMPatchBaselineForPatchGroup`

O código de exemplo a seguir mostra como usar `Unregister-SSMPatchBaselineForPatchGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cancela o registro de um grupo de patches de uma lista de referência de patches.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Saída:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Para obter detalhes sobre a API, consulte DeregisterPatchBaselineForPatchGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-SSMTargetFromMaintenanceWindow`

O código de exemplo a seguir mostra como usar `Unregister-SSMTargetFromMaintenanceWindow`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove um destino de uma janela de manutenção.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Saída:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Para obter detalhes sobre a API, consulte DeregisterTargetFromMaintenanceWindow Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-SSMTaskFromMaintenanceWindow`

O código de exemplo a seguir mostra como usar `Unregister-SSMTaskFromMaintenanceWindow`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove uma tarefa de uma janela de manutenção.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Saída:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Para obter detalhes sobre a API, consulte DeregisterTaskFromMaintenanceWindow Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-SSMAssociation`

O código de exemplo a seguir mostra como usar `Update-SSMAssociation`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza uma associação com uma nova versão de documento.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Saída:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Para obter detalhes sobre a API, consulte UpdateAssociation Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-SSMAssociationStatus`

O código de exemplo a seguir mostra como usar `Update-SSMAssociationStatus`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza o status da associação entre uma instância e um documento de configuração.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Saída:

```
Name                   : AWS-UpdateSSMAgent
InstanceId             : i-0000293ffd8c57862
Date                   : 2/23/2017 6:55:22 PM
Status.Name            : Pending
Status.Date            : 2/20/2015 8:31:11 AM
Status.Message         : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Para obter detalhes sobre a API, consulte UpdateAssociationStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-SSMDocument`

O código de exemplo a seguir mostra como usar `Update-SSMDocument`.

Ferramentas para PowerShell V5

Exemplo 1: isso cria uma nova versão de um documento com o conteúdo atualizado do arquivo json que você especificar. O documento deve estar em formato JSON. Você pode obter a versão do documento com o cmdlet Get-SSMDocumentVersionList "”.

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -
Raw "c:\temp\RunShellScript.json")
```

Saída:

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- Para obter detalhes sobre a API, consulte UpdateDocument Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-SSMDocumentDefaultVersion`

O código de exemplo a seguir mostra como usar `Update-SSMDocumentDefaultVersion`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza a versão padrão de um documento. Você pode obter as versões de documentos disponíveis com o cmdlet Get-SSMDocumentVersionList "".

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Saída:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- Para obter detalhes sobre a API, consulte UpdateDocumentDefaultVersion Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-SSMMaintenanceWindow`

O código de exemplo a seguir mostra como usar `Update-SSMMaintenanceWindow`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza o nome de uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Exemplo 2: esse exemplo habilita uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Exemplo 3: esse exemplo desabilita uma janela de manutenção.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Saída:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Para obter detalhes sobre a API, consulte UpdateMaintenanceWindow Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-SSMManagedInstanceRole`

O código de exemplo a seguir mostra como usar `Update-SSMManagedInstanceRole`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atualiza o perfil de uma instância gerenciada. Não haverá saída se o comando for bem-sucedido.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- Para obter detalhes sobre a API, consulte UpdateManagedInstanceRole Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Update-SSMPatchBaseline`

O código de exemplo a seguir mostra como usar `Update-SSMPatchBaseline`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona dois patches como rejeitados e um patch como aprovado a uma lista de referência de patches existente.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Saída:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- Para obter detalhes sobre a API, consulte UpdatePatchBaseline Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-SSMComplianceItem`

O código de exemplo a seguir mostra como usar `Write-SSMComplianceItem`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo grava um item de conformidade personalizado para a instância gerenciada especificada

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- Para obter detalhes sobre a API, consulte PutComplianceItems Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-SSMInventory`

O código de exemplo a seguir mostra como usar `Write-SSMInventory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo atribui informações de localização de rack a uma instância. Não haverá saída se o comando for bem-sucedido.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
$items.Add($data)

$customInventoryItem = New-Object Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- Para obter detalhes sobre a API, consulte PutInventory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Write-SSMParameter`

O código de exemplo a seguir mostra como usar `Write-SSMParameter`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo cria um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Exemplo 2: esse exemplo altera um parâmetro. Não haverá saída se o comando for bem-sucedido.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- Para obter detalhes sobre a API, consulte PutParameter Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos do Amazon Translate usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com o Amazon Translate.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ConvertTo-TRNTargetLanguage`

O código de exemplo a seguir mostra como usar `ConvertTo-TRNTargetLanguage`.

Ferramentas para PowerShell V5

Exemplo 1: converte o texto em inglês especificado em francês. O texto a ser convertido também pode ser passado como o parâmetro -Text.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- Para obter detalhes sobre a API, consulte TranslateText Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

AWS WAFV2 exemplos usando ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com AWS WAFV2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`New-WAF2WebACL`

O código de exemplo a seguir mostra como usar `New-WAF2WebACL`.

Ferramentas para PowerShell V5

Exemplo 1: esse comando cria uma ACL da web chamada “waf-test”. Observe que, de acordo com a documentação da API de serviço, 'DefaultAction' é uma propriedade obrigatória. Portanto, o valor de '- DefaultAction _Allow' '- and/or DefaultAction _Block' deve ser especificado. Como '- DefaultAction _Allow' e '- DefaultAction _Block' não são as propriedades necessárias, o valor '@ {} 'pode ser usado como espaço reservado, conforme mostrado no exemplo acima.

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-
west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -
VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-
test" -Description "Test" -DefaultAction_Allow @{}
```

Saída:

```
ARN         : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-
test/19460b3f-db14-4b9a-8e23-a417e1eb007f
Description : Test
Id          : 19460b3f-db14-4b9a-8e23-a417e1eb007f
```

```
LockToken   : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab
Name        : waf-test
```

- Para obter detalhes sobre a API, consulte CreateWebAcl Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

WorkSpaces exemplos de uso de ferramentas para PowerShell V5

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o Ferramentas da AWS para PowerShell V5 com WorkSpaces.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Approve-WKSIpRule`

O código de exemplo a seguir mostra como usar `Approve-WKSIpRule`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona regras a um grupo de IP existente

```
$Rule = @(
@{IPRule = "10.1.0.0/0"; RuleDesc = "First Rule Added"},
@{IPRule = "10.2.0.0/0"; RuleDesc = "Second Rule Added"}
)

Approve-WKSIpRule -GroupId wsipg-abcnx2fcw -UserRule $Rule
```

- Para obter detalhes sobre a API, consulte AuthorizeIpRules Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Copy-WKSWorkspaceImage`

O código de exemplo a seguir mostra como usar `Copy-WKSWorkspaceImage`.

Ferramentas para PowerShell V5

Exemplo 1: Esta amostra copia a imagem do espaço de trabalho com a ID especificada de us-west-2 para a região atual com o nome "" CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId
 wsi-djfoedhw6
```

Saída:

```
wsi-456abaqfe
```

- Para obter detalhes sobre a API, consulte CopyWorkspaceImage Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-WKSClientProperty`

O código de exemplo a seguir mostra como usar `Edit-WKSClientProperty`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo permite a reconexão para o cliente do Workspaces

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled
 "ENABLED" -ResourceId d-123414a369
```

- Para obter detalhes sobre a API, consulte ModifyClientProperties Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-WKSSelfServicePermission`

O código de exemplo a seguir mostra como usar `Edit-WKSSelfServicePermission`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo concede permissões de autoatendimento para alterar o tipo de computação e aumentar o tamanho do volume para o diretório especificado

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId
 d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- Para obter detalhes sobre a API, consulte ModifySelfservicePermissions Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-WKSWorkspaceAccessProperty`

O código de exemplo a seguir mostra como usar `Edit-WKSWorkspaceAccessProperty`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo permite o acesso ao Workspace no Android e no Chrome OS para o diretório especificado

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- Para obter detalhes sobre a API, consulte ModifyWorkspaceAccessProperties Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-WKSWorkspaceCreationProperty`

O código de exemplo a seguir mostra como usar `Edit-WKSWorkspaceCreationProperty`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo permite que o Modo de Manutenção e Acesso à Internet seja verdadeiro como valores padrão ao criar um espaço de trabalho

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Para obter detalhes sobre a API, consulte ModifyWorkspaceCreationProperties Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-WKSWorkspaceProperty`

O código de exemplo a seguir mostra como usar `Edit-WKSWorkspaceProperty`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera a propriedade do modo de execução do espaço de trabalho para interrupção automática referente ao espaço de trabalho especificado

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- Para obter detalhes sobre a API, consulte ModifyWorkspaceProperties Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Edit-WKSWorkspaceState`

O código de exemplo a seguir mostra como usar `Edit-WKSWorkspaceState`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo altera o estado do espaço de trabalho especificado para Disponível

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState
 AVAILABLE
```

- Para obter detalhes sobre a API, consulte ModifyWorkspaceState Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSClientProperty`

O código de exemplo a seguir mostra como usar `Get-WKSClientProperty`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém as propriedades do cliente do Workspace Client para o diretório especificado

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- Para obter detalhes sobre a API, consulte DescribeClientProperties Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSIpGroup`

O código de exemplo a seguir mostra como usar `Get-WKSIpGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo obtém os detalhes do grupo de IP especificado na região especificada

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

Saída:

```
GroupDesc GroupId         GroupName UserRules
--------- -------         --------- ---------
          wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
 Amazon.WorkSpaces.Model.IpRuleItem}
```

- Para obter detalhes sobre a API, consulte DescribeIpGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSTag`

O código de exemplo a seguir mostra como usar `Get-WKSTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo busca a tag para o espaço de trabalho fornecido

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Saída:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- Para obter detalhes sobre a API, consulte DescribeTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSWorkspace`

O código de exemplo a seguir mostra como usar `Get-WKSWorkspace`.

Ferramentas para PowerShell V5

Exemplo 1: recupera detalhes de todos os seus WorkSpaces para o funil.

```
Get-WKSWorkspace
```

Saída:

```
BundleId                    : wsb-1a2b3c4d
ComputerName                :
DirectoryId                 : d-1a2b3c4d
ErrorCode                   :
ErrorMessage                :
IpAddress                   :
RootVolumeEncryptionEnabled : False
State                       : PENDING
SubnetId                    :
UserName                    : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey         :
WorkspaceId                 : ws-1a2b3c4d
WorkspaceProperties         : Amazon.WorkSpaces.Model.WorkspaceProperties
```

Exemplo 2: esse comando mostra os valores das propriedades secundárias de **`WorkSpaceProperties`** para um espaço de trabalho na região **`us-west-2`**. Para obter mais informações sobre as propriedades secundárias de**`WorkSpaceProperties`**, consulte https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-xdaf7hc9s).WorkspaceProperties
```

Saída:

```
        ComputeTypeName                     : STANDARD
```

```
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

Exemplo 3: esse comando mostra o valor da propriedade secundária **`RootVolumeSizeGib`** de **`WorkSpaceProperties`** para um espaço de trabalho na região **`us-west-2`**. O tamanho do volume raiz, em GiB, é 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Saída:

```
        80
```

- Para obter detalhes sobre a API, consulte DescribeWorkspaces Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSWorkspaceBundle`

O código de exemplo a seguir mostra como usar `Get-WKSWorkspaceBundle`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra busca detalhes de todos os pacotes do Workspace na região atual

```
Get-WKSWorkspaceBundle
```

Saída:

```
BundleId        : wsb-sfhdgv342
ComputeType     : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId         : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name            : CustomBundleTest
Owner           : 233816212345
RootStorage     : Amazon.WorkSpaces.Model.RootStorage
```

```
UserStorage     : Amazon.WorkSpaces.Model.UserStorage
```

- Para obter detalhes sobre a API, consulte DescribeWorkspaceBundles Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSWorkspaceDirectory`

O código de exemplo a seguir mostra como usar `Get-WKSWorkspaceDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra lista os detalhes do diretório para diretórios registrados

```
Get-WKSWorkspaceDirectory
```

Saída:

```
Alias                       : TestWorkspace
CustomerUserName            : Administrator
DirectoryId                 : d-123414a369
DirectoryName               : TestDirectory.com
DirectoryType               : MicrosoftAD
DnsIpAddresses              : {172.31.43.45, 172.31.2.97}
IamRoleId                   : arn:aws:iam::761234567801:role/workspaces_RoleDefault
IpGroupIds                  : {}
RegistrationCode            : WSpdx+4RRT43
SelfservicePermissions      : Amazon.WorkSpaces.Model.SelfservicePermissions
State                       : REGISTERED
SubnetIds                   : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy                     : SHARED
WorkspaceAccessProperties   : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
 Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
WorkspaceSecurityGroupId    : sg-0ed2441234a123c43
```

- Para obter detalhes sobre a API, consulte DescribeWorkspaceDirectories Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSWorkspaceImage`

O código de exemplo a seguir mostra como usar `Get-WKSWorkspaceImage`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra busca todos os detalhes de todas as imagens na região

```
Get-WKSWorkspaceImage
```

Saída:

```
Description     :This image is copied from another image
ErrorCode       :
ErrorMessage    :
ImageId         : wsi-345ahdjgo
Name            : CopiedImageTest
OperatingSystem : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy : DEFAULT
State           : AVAILABLE
```

- Para obter detalhes sobre a API, consulte DescribeWorkspaceImages Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSWorkspaceSnapshot`

O código de exemplo a seguir mostra como usar `Get-WKSWorkspaceSnapshot`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra mostra o carimbo de data/hora do snapshot mais recente criado para o espaço de trabalho especificado

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

Saída:

```
RebuildSnapshots                   RestoreSnapshots
----------------                   ----------------
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- Para obter detalhes sobre a API, consulte DescribeWorkspaceSnapshots Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Get-WKSWorkspacesConnectionStatus`

O código de exemplo a seguir mostra como usar `Get-WKSWorkspacesConnectionStatus`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra busca o status da conexão para o espaço de trabalho especificado

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Para obter detalhes sobre a API, consulte DescribeWorkspacesConnectionStatus Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-WKSIpGroup`

O código de exemplo a seguir mostra como usar `New-WKSIpGroup`.

Ferramentas para PowerShell V5

Exemplo 1: Esse exemplo cria um grupo de IP vazio chamado FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

Saída:

```
wsipg-w45rty4ty
```

- Para obter detalhes sobre a API, consulte CreateIpGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-WKSTag`

O código de exemplo a seguir mostra como usar `New-WKSTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo adiciona uma nova tag a um espaço de trabalho chamado **`ws-wsname`**. A tag tem uma chave de “Name” e um valor-chave de **`AWS_Workspace`**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
```

```
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

Exemplo 2: esse exemplo adiciona várias tags a um espaço de trabalho chamado **`ws-wsname`**. Uma tag tem uma chave de “Name” e um valor-chave de **`AWS_Workspace`**; a outra tag tem uma chave de tag de “Stage” e um valor-chave de “Test”.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- Para obter detalhes sobre a API, consulte CreateTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`New-WKSWorkspace`

O código de exemplo a seguir mostra como usar `New-WKSWorkspace`.

Ferramentas para PowerShell V5

Exemplo 1: Crie um WorkSpace para o pacote, diretório e usuário fornecidos.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Exemplo 2: Este exemplo cria vários WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Para obter detalhes sobre a API, consulte CreateWorkspaces Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-WKSIpGroup`

O código de exemplo a seguir mostra como usar `Register-WKSIpGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra o grupo de IP especificado com o diretório especificado

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- Para obter detalhes sobre a API, consulte AssociateIpGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Register-WKSWorkspaceDirectory`

O código de exemplo a seguir mostra como usar `Register-WKSWorkspaceDirectory`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo registra o diretório especificado para o Workspaces Service

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Para obter detalhes sobre a API, consulte RegisterWorkspaceDirectory Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-WKSIpGroup`

O código de exemplo a seguir mostra como usar `Remove-WKSIpGroup`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo exclui o grupo de IP especificado

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

Saída:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-WKSIpGroup (DeleteIpGroup)" on target
 "wsipg-32fhgtred".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteIpGroup Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-WKSTag`

O código de exemplo a seguir mostra como usar `Remove-WKSTag`.

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo remove a tag associada ao espaço de trabalho

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Saída:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Para obter detalhes sobre a API, consulte DeleteTags Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Remove-WKSWorkspace`

O código de exemplo a seguir mostra como usar `Remove-WKSWorkspace`.

Ferramentas para PowerShell V5

Exemplo 1: Encerra vários WorkSpaces. O uso da opção -Force impede que o cmdlet solicite confirmação.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Exemplo 2: Recupera a coleção de todos os seus WorkSpaces e envia os IDs para o WorkSpaceId parâmetro - de Remove-WKSWorkspace, encerrando todos os. WorkSpaces O cmdlet solicitará antes que cada um WorkSpace seja encerrado. Para ignorar a solicitação de confirmação, adicione a opção -Force.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Exemplo 3: Este exemplo mostra como passar TerminateRequest objetos definindo o WorkSpaces que será encerrado. O cmdlet solicitará a confirmação antes de continuar, a menos que o parâmetro da opção -Force também seja especificado.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Remove-WKSWorkspace -Request $arrRequest
```

- Para obter detalhes sobre a API, consulte TerminateWorkspaces Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Reset-WKSWorkspace`

O código de exemplo a seguir mostra como usar `Reset-WKSWorkspace`.

Ferramentas para PowerShell V5

Exemplo 1: Reconstrói o especificado. WorkSpace

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemplo 2: Recupera a coleção de todas as suas WorkSpaces e envia as IDs para o WorkSpaceId parâmetro - de Reset-WKSWorkspace, fazendo com que WorkSpaces a seja reconstruída.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Para obter detalhes sobre a API, consulte RebuildWorkspaces Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Restart-WKSWorkspace`

O código de exemplo a seguir mostra como usar `Restart-WKSWorkspace`.

Ferramentas para PowerShell V5

Exemplo 1: Reinicializa o especificado WorkSpace.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Exemplo 2: Reinicializa várias WorkSpaces.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Exemplo 3: Recupera a coleção de todos os seus WorkSpaces e envia os IDs para o WorkSpaceId parâmetro - de Restart-WKSWorkspace, fazendo com que o WorkSpaces seja reiniciado.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Para obter detalhes sobre a API, consulte RebootWorkspaces Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Stop-WKSWorkspace`

O código de exemplo a seguir mostra como usar `Stop-WKSWorkspace`.

Ferramentas para PowerShell V5

Exemplo 1: interrompe várias WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Exemplo 2: Recupera a coleção de todas as suas WorkSpaces e envia as IDs para o WorkSpaceId parâmetro - de fazer Stop-WKSWorkspace com que WorkSpaces a seja interrompida.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Exemplo 3: Este exemplo mostra como passar StopRequest objetos definindo o WorkSpaces que deve ser interrompido.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Stop-WKSWorkspace -Request $arrRequest
```

- Para obter detalhes sobre a API, consulte StopWorkspaces Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

`Unregister-WKSIpGroup`

O código de exemplo a seguir mostra como usar `Unregister-WKSIpGroup`.

Ferramentas para PowerShell V5

Exemplo 1: essa amostra cancela o registro do grupo de IP especificado do diretório especificado

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- Para obter detalhes sobre a API, consulte DisassociateIpGroups Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Exemplos de código para SDK para Python

Os exemplos de código a seguir mostram como usar o AWS SDK para Python with AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para Python — Saiba mais sobre como usar o Python com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos do DynamoDB usando o SDK para Python

Exemplos do DynamoDB usando o SDK para Python

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python com o DynamoDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

SDK for Python

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
This is a test output. Examples for AWS SDK for Python
for AWS DynamoDB will be available here soon.
```

- Para obter detalhes da API, consulte ListTables em AWS SDK for Python API Reference.

Exemplos de código para o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como usar o AWS SDK para Python (Boto3) with AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para Python (Boto3) — Saiba mais sobre como usar Python com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.

- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos de ACM usando o SDK para Python (Boto3)
- Exemplos da API Gateway usando o SDK para Python (Boto3)
- Exemplos do Application Recovery Controller usando o SDK para Python (Boto3)
- Exemplos do Audit Manager usando o SDK para Python (Boto3)
- Exemplos de Aurora usando o SDK para Python (Boto3)
- Exemplos do Auto Scaling usando o SDK para Python (Boto3)
- Exemplos do Amazon Bedrock usando o SDK para Python (Boto3)
- Exemplos do Amazon Bedrock Runtime usando o SDK para Python (Boto3)
- Exemplos de agente dos Amazon Bedrock usando o SDK para Python (Boto3)
- Exemplos do Amazon Bedrock Agents Runtime usando o SDK para Python (Boto3)
- CloudFormation exemplos de uso do SDK para Python (Boto3)
- CloudFront exemplos de uso do SDK para Python (Boto3)
- CloudWatch exemplos de uso do SDK para Python (Boto3)
- CloudWatch Exemplos de registros usando o SDK para Python (Boto3)
- Exemplos de código do Amazon Cognito Identity usando o SDK para Python (Boto3)
- Exemplos de código do Provedor de Identidades Amazon Cognito usando o SDK para Python (Boto3)
- Exemplos do Amazon Comprehend usando o SDK para Python (Boto3)
- AWS Config exemplos de uso do SDK para Python (Boto3)
- AWS Control Tower exemplos de uso do SDK para Python (Boto3)
- Exemplos do Firehose usando o SDK para Python (Boto3)
- Exemplos de Device Farm usando o SDK para Python (Boto3)
- Exemplos do Amazon DocumentDB usando o SDK para Python (Boto3)
- Exemplos de DynamoDB usando o SDK para Python (Boto3)
- Exemplos do Amazon EC2 usando o SDK para Python (Boto3)
- Exemplos do Amazon ECR usando o SDK para Python (Boto3)

- Exemplos do Elastic Load Balancing Versão 2 usando o SDK para Python (Boto3)
- Exemplos do Amazon EMR usando o SDK para Python (Boto3)
- EventBridge exemplos de uso do SDK para Python (Boto3)
- EventBridge Exemplos de agendador usando SDK para Python (Boto3)
- Exemplos do Amazon Glacier usando o SDK para Python (Boto3)
- AWS Glue exemplos de uso do SDK para Python (Boto3)
- HealthImaging exemplos de uso do SDK para Python (Boto3)
- HealthLake exemplos de uso do SDK para Python (Boto3)
- Exemplos de IAM usando o SDK para Python (Boto3)
- AWS IoT exemplos de uso do SDK para Python (Boto3)
- AWS IoT data exemplos de uso do SDK para Python (Boto3)
- AWS IoT SiteWise exemplos de uso do SDK para Python (Boto3)
- Exemplos do Amazon Keyspaces usando o SDK para Python (Boto3)
- Exemplos do Kinesis usando o SDK para Python (Boto3)
- AWS KMS exemplos de uso do SDK para Python (Boto3)
- Exemplos de Lambda usando o SDK para Python (Boto3)
- Exemplos de Serviço gerenciado para Apache Flink usando o SDK para Python (Boto3)
- AWS Marketplace Exemplos de API de catálogo usando SDK para Python (Boto3)
- AWS Marketplace Exemplos de API de contrato usando SDK para Python (Boto3)
- Exemplos do Amazon MSK usando o SDK para Python (Boto3)
- Exemplos do Neptune usando o SDK para Python (Boto3)
- Exemplos do Organizations usando o SDK para Python (Boto3)
- Exemplos da Central de Parceiros usando o SDK para Python (Boto3)
- Exemplos do Amazon Pinpoint usando o SDK para Python (Boto3)
- Exemplos da API SMS and Voice do Amazon Pinpoint usando o SDK para Python (Boto3)
- Exemplos do Amazon Polly usando o SDK para Python (Boto3)
- Exemplos do Amazon RDS usando o SDK para Python (Boto3)
- Exemplos do Amazon RDS Data Service usando o SDK para Python (Boto3)
- Exemplos do Amazon Redshift usando o SDK para Python (Boto3)

- Exemplos do Amazon Rekognition usando o SDK para Python (Boto3)
- Exemplos do Amazon S3 usando o SDK para Python (Boto3)
- Exemplos de controle do Amazon S3 usando o SDK para Python (Boto3)
- Exemplos de buckets de diretório do S3 usando o SDK para Python (Boto3)
- Exemplos do Secrets Manager usando o SDK para Python (Boto3)
- Exemplos do Amazon SES usando o SDK para Python (Boto3)
- Exemplos da API v2 do Amazon SES usando o SDK para Python (Boto3)
- Exemplos do Amazon SNS usando o SDK para Python (Boto3)
- Exemplos do Amazon SQS usando o SDK para Python (Boto3)
- Exemplos do Step Functions usando o SDK para Python (Boto3)
- AWS STS exemplos de uso do SDK para Python (Boto3)
- Suporte exemplos de uso do SDK para Python (Boto3)
- Exemplos do Systems Manager usando o SDK para Python (Boto3)
- Exemplos do Amazon Textract usando o SDK para Python (Boto3)
- Exemplos do Amazon Transcribe usando o SDK para Python (Boto3)

Exemplos de ACM usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o ACM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Solicitar um certificado do ACM.
- Importar um certificado autoassinado.
- Listar e descrever certificados.
- Remover certificados.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva operações do ACM.

```
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client
```

```
    def request_validation(
        self, domain, alternate_domains, method, validation_domains=None
    ):
        """
        Starts a validation request that results in a new certificate being issued
        by ACM. DNS validation requires that you add CNAME records to your DNS
        provider. Email validation sends email to a list of email addresses that
        are associated with the domain.

        For more information, see _Issuing and managing certificates_ in the ACM
        user guide.
            https://docs.aws.amazon.com/acm/latest/userguide/gs.html

        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternate Names (SANs) for the
certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation,
when
                                   the email domain differs from the primary domain
of
                                   the certificate.
        :return: The ARN of the requested certificate.
        """
        try:
            kwargs = {
                "DomainName": domain,
                "ValidationMethod": method,
                "SubjectAlternativeNames": alternate_domains,
            }
            if validation_domains is not None:
                kwargs["DomainValidationOptions"] = [
                    {"DomainName": key, "ValidationDomain": value}
                    for key, value in validation_domains.items()
                ]
            response = self.acm_client.request_certificate(**kwargs)
            certificate_arn = response["CertificateArn"]
            logger.info(
                "Requested %s validation for domain %s. Certificate ARN is %s.",
                method,
                domain,
                certificate_arn,
            )
```

```
        except ClientError:
            logger.exception(
                "Request for %s validation of domain %s failed.", method, domain
            )
            raise
        else:
            return certificate_arn


    def import_certificate(self, certificate_body, private_key):
        """
        Imports a self-signed certificate to ACM.

        :param certificate_body: The body of the certificate, in PEM format.
        :param private_key: The unencrypted private key of the certificate, in PEM
                            format.
        :return: The ARN of the imported certificate.
        """
        try:
            response = self.acm_client.import_certificate(
                Certificate=certificate_body, PrivateKey=private_key
            )
            certificate_arn = response["CertificateArn"]
            logger.info("Imported certificate.")
        except ClientError:
            logger.exception("Couldn't import certificate.")
            raise
        else:
            return certificate_arn


    def list(
        self,
        max_items,
        statuses=None,
        key_usage=None,
        extended_key_usage=None,
        key_types=None,
    ):
        """
        Lists the certificates for the current account.

        :param max_items: The maximum number of certificates to list.
        :param statuses: Filters the results to the specified statuses. If None, all
```

```
                         certificates are included.
        :param key_usage: Filters the results to the specified key usages. If None,
                          all key usages are included.
        :param extended_key_usage: Filters the results to the specified extended key
                                   usages. If None, all extended key usages are
                                   included.
        :param key_types: Filters the results to the specified key types. If None,
all
                          key types are included.
        :return: The list of certificates.
        """
        try:
            kwargs = {"MaxItems": max_items}
            if statuses is not None:
                kwargs["CertificateStatuses"] = statuses
            includes = {}
            if key_usage is not None:
                includes["keyUsage"] = key_usage
            if extended_key_usage is not None:
                includes["extendedKeyUsage"] = extended_key_usage
            if key_types is not None:
                includes["keyTypes"] = key_types
            if includes:
                kwargs["Includes"] = includes
            response = self.acm_client.list_certificates(**kwargs)
            certificates = response["CertificateSummaryList"]
            logger.info("Got %s certificates.", len(certificates))
        except ClientError:
            logger.exception("Couldn't get certificates.")
            raise
        else:
            return certificates


    def describe(self, certificate_arn):
        """
        Gets certificate metadata.

        :param certificate_arn: The Amazon Resource Name (ARN) of the certificate.
        :return: Metadata about the certificate.
        """
        try:
            response = self.acm_client.describe_certificate(
                CertificateArn=certificate_arn
```

```
            )
            certificate = response["Certificate"]
            logger.info(
                "Got metadata for certificate for domain %s.",
certificate["DomainName"]
            )
        except ClientError:
            logger.exception("Couldn't get data for certificate %s.",
certificate_arn)
            raise
        else:
            return certificate


    def get(self, certificate_arn):
        """
        Gets the body and certificate chain of a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The body and chain of a certificate.
        """
        try:
            response =
self.acm_client.get_certificate(CertificateArn=certificate_arn)
            logger.info("Got certificate %s and its chain.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't get certificate %s.", certificate_arn)
            raise
        else:
            return response


    def add_tags(self, certificate_arn, tags):
        """
        Adds tags to a certificate. Tags are key-value pairs that contain custom
        metadata.

        :param certificate_arn: The ARN of the certificate.
        :param tags: A dictionary of key-value tags to add to the certificate.
        """
        try:
            self.acm_client.add_tags_to_certificate(
                CertificateArn=certificate_arn,
                Tags=[{"Key": key, "Value": value} for key, value in tags.items()],
```

```
            )
            logger.info("Added %s tags to certificate %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't add tags to certificate %s.",
certificate_arn)
            raise


    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The dictionary of certificate tags.
        """
        try:
            response = self.acm_client.list_tags_for_certificate(
                CertificateArn=certificate_arn
            )
            tags = {tag["Key"]: tag["Value"] for tag in response["Tags"]}
            logger.info("Got %s tags for certificates %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't get tags for certificate %s.",
certificate_arn)
            raise
        else:
            return tags


    def remove_tags(self, certificate_arn, tags):
        """
        Removes tags from a certificate. If the value of a tag is specified, the tag
is
        removed only when the value matches the value of the certificate's tag.
        Otherwise, the tag is removed regardless of its value.

        :param certificate_arn: The ARN of the certificate.
        :param tags: The dictionary of tags to remove.
        """
        try:
            cert_tags = []
            for key, value in tags.items():
```

```
                tag = {"Key": key}
                if value is not None:
                    tag["Value"] = value
                cert_tags.append(tag)
            self.acm_client.remove_tags_from_certificate(
                CertificateArn=certificate_arn, Tags=cert_tags
            )
            logger.info(
                "Removed %s tags from certificate %s.", len(tags), certificate_arn
            )
        except ClientError:
            logger.exception(
                "Couldn't remove tags from certificate %s.", certificate_arn
            )
            raise


    def remove(self, certificate_arn):
        """
        Removes a certificate.

        :param certificate_arn: The ARN of the certificate to remove.
        """
        try:
            self.acm_client.delete_certificate(CertificateArn=certificate_arn)
            logger.info("Removed certificate %s.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't remove certificate %s.", certificate_arn)
            raise
```

Use a classe wrapper para gerenciar certificados para sua conta.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the AWS Certificate Manager (ACM) demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    acm_certificate = AcmCertificate(boto3.client("acm"))
    domain = "example.com"
```

```
    sub_domains = [f"{sub}.{domain}" for sub in ["test", "dev"]]
    print(f"Request a certificate for {domain}.")
    certificate_arn = acm_certificate.request_validation(domain, sub_domains, "DNS")
    print(f"Started validation, got certificate ARN: {certificate_arn}.")

    import_cert_arn = None
    cert_file_name = input(
        "Enter the file name for a self-signed certificate in PEM format. "
        "This certificate will be imported to ACM. Press Enter to skip: "
    )
    if cert_file_name:
        pk_file_name = input(
            "Enter the file name for the unencrypted private key of the certificate.
"
            "This file must also be in PEM format: "
        )
        if pk_file_name:
            with open(cert_file_name, "rb") as cert_file:
                import_cert = cert_file.read()
            with open(pk_file_name, "rb") as pk_file:
                import_pk = pk_file.read()
            import_cert_arn = acm_certificate.import_certificate(import_cert,
import_pk)
            print(f"Certificate imported, got ARN: {import_cert_arn}")
        else:
            print("No private key file entered. Skipping certificate import.")
    else:
        print("Skipping self-signed certificate import.")

    print("Getting the first 10 issued certificates.")
    certificates = acm_certificate.list(10, statuses=["ISSUED"])
    print(f"Found {len(certificates)} issued certificates.")

    print(f"Getting metadata for certificate {certificate_arn}")
    cert_metadata = acm_certificate.describe(certificate_arn)
    pprint(cert_metadata)

    if import_cert_arn is not None:
        print(f"Getting certificate for imported certificate {import_cert_arn}")
        import_cert_data = acm_certificate.get(import_cert_arn)
        pprint(import_cert_data)

    print(f"Adding tags to certificate {certificate_arn}.")
```

```
    acm_certificate.add_tags(certificate_arn, {"purpose": "acm demo", "color":
"green"})
    tags = acm_certificate.list_tags(certificate_arn)
    print(f"Found tags: {tags}")
    acm_certificate.remove_tags(certificate_arn, {key: None for key in tags})
    print("Removed tags.")

    print("Removing certificates added during the demo.")
    acm_certificate.remove(certificate_arn)
    if import_cert_arn is not None:
        acm_certificate.remove(import_cert_arn)

    print("Thanks for watching!")
    print("-" * 88)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AddTagsToCertificate
  - DeleteCertificate
  - DescribeCertificate
  - GetCertificate
  - ImportCertificate
  - ListCertificates
  - ListTagsForCertificate
  - RemoveTagsFromCertificate
  - RequestCertificate
  - ResendValidationEmail

Ações

`AddTagsToCertificate`

O código de exemplo a seguir mostra como usar `AddTagsToCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def add_tags(self, certificate_arn, tags):
        """
        Adds tags to a certificate. Tags are key-value pairs that contain custom
        metadata.

        :param certificate_arn: The ARN of the certificate.
        :param tags: A dictionary of key-value tags to add to the certificate.
        """
        try:
            self.acm_client.add_tags_to_certificate(
                CertificateArn=certificate_arn,
                Tags=[{"Key": key, "Value": value} for key, value in tags.items()],
            )
            logger.info("Added %s tags to certificate %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't add tags to certificate %s.",
certificate_arn)
            raise
```

- Para obter detalhes da API, consulte a AddTagsToCertificate Referência da API AWS SDK for Python (Boto3).

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def remove(self, certificate_arn):
        """
        Removes a certificate.

        :param certificate_arn: The ARN of the certificate to remove.
        """
        try:
            self.acm_client.delete_certificate(CertificateArn=certificate_arn)
            logger.info("Removed certificate %s.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't remove certificate %s.", certificate_arn)
            raise
```

- Para obter detalhes da API, consulte a DeleteCertificate Referência da API AWS SDK for Python (Boto3).

`DescribeCertificate`

O código de exemplo a seguir mostra como usar `DescribeCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def describe(self, certificate_arn):
        """
        Gets certificate metadata.

        :param certificate_arn: The Amazon Resource Name (ARN) of the certificate.
        :return: Metadata about the certificate.
        """
        try:
            response = self.acm_client.describe_certificate(
                CertificateArn=certificate_arn
            )
            certificate = response["Certificate"]
            logger.info(
                "Got metadata for certificate for domain %s.",
 certificate["DomainName"]
```

```
            )
        except ClientError:
            logger.exception("Couldn't get data for certificate %s.",
 certificate_arn)
            raise
        else:
            return certificate
```

- Para obter detalhes da API, consulte a DescribeCertificate Referência da API AWS SDK for Python (Boto3).

`GetCertificate`

O código de exemplo a seguir mostra como usar `GetCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def get(self, certificate_arn):
        """
        Gets the body and certificate chain of a certificate.
```

```
        :param certificate_arn: The ARN of the certificate.
        :return: The body and chain of a certificate.
        """
        try:
            response =
self.acm_client.get_certificate(CertificateArn=certificate_arn)
            logger.info("Got certificate %s and its chain.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't get certificate %s.", certificate_arn)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a GetCertificate Referência da API AWS SDK for Python (Boto3).

`ImportCertificate`

O código de exemplo a seguir mostra como usar `ImportCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client
```

```
    def import_certificate(self, certificate_body, private_key):
        """
        Imports a self-signed certificate to ACM.

        :param certificate_body: The body of the certificate, in PEM format.
        :param private_key: The unencrypted private key of the certificate, in PEM
                            format.
        :return: The ARN of the imported certificate.
        """
        try:
            response = self.acm_client.import_certificate(
                Certificate=certificate_body, PrivateKey=private_key
            )
            certificate_arn = response["CertificateArn"]
            logger.info("Imported certificate.")
        except ClientError:
            logger.exception("Couldn't import certificate.")
            raise
        else:
            return certificate_arn
```

- Para obter detalhes da API, consulte a ImportCertificate Referência da API AWS SDK for Python (Boto3).

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
```

```
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def list(
        self,
        max_items,
        statuses=None,
        key_usage=None,
        extended_key_usage=None,
        key_types=None,
    ):
        """
        Lists the certificates for the current account.

        :param max_items: The maximum number of certificates to list.
        :param statuses: Filters the results to the specified statuses. If None, all
                         certificates are included.
        :param key_usage: Filters the results to the specified key usages. If None,
                          all key usages are included.
        :param extended_key_usage: Filters the results to the specified extended key
                                   usages. If None, all extended key usages are
                                   included.
        :param key_types: Filters the results to the specified key types. If None,
all
                          key types are included.
        :return: The list of certificates.
        """
        try:
            kwargs = {"MaxItems": max_items}
            if statuses is not None:
                kwargs["CertificateStatuses"] = statuses
            includes = {}
            if key_usage is not None:
                includes["keyUsage"] = key_usage
            if extended_key_usage is not None:
                includes["extendedKeyUsage"] = extended_key_usage
            if key_types is not None:
```

```
                includes["keyTypes"] = key_types
            if includes:
                kwargs["Includes"] = includes
            response = self.acm_client.list_certificates(**kwargs)
            certificates = response["CertificateSummaryList"]
            logger.info("Got %s certificates.", len(certificates))
        except ClientError:
            logger.exception("Couldn't get certificates.")
            raise
        else:
            return certificates
```

- Para obter detalhes da API, consulte a ListCertificates Referência da API AWS SDK for Python (Boto3).

`ListTagsForCertificate`

O código de exemplo a seguir mostra como usar `ListTagsForCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client
```

```
    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The dictionary of certificate tags.
        """
        try:
            response = self.acm_client.list_tags_for_certificate(
                CertificateArn=certificate_arn
            )
            tags = {tag["Key"]: tag["Value"] for tag in response["Tags"]}
            logger.info("Got %s tags for certificates %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't get tags for certificate %s.",
certificate_arn)
            raise
        else:
            return tags
```

- Para obter detalhes da API, consulte a ListTagsForCertificate Referência da API AWS SDK for Python (Boto3).

`RemoveTagsFromCertificate`

O código de exemplo a seguir mostra como usar `RemoveTagsFromCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
```

```
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def remove_tags(self, certificate_arn, tags):
        """
        Removes tags from a certificate. If the value of a tag is specified, the tag
is
        removed only when the value matches the value of the certificate's tag.
        Otherwise, the tag is removed regardless of its value.

        :param certificate_arn: The ARN of the certificate.
        :param tags: The dictionary of tags to remove.
        """
        try:
            cert_tags = []
            for key, value in tags.items():
                tag = {"Key": key}
                if value is not None:
                    tag["Value"] = value
                cert_tags.append(tag)
            self.acm_client.remove_tags_from_certificate(
                CertificateArn=certificate_arn, Tags=cert_tags
            )
            logger.info(
                "Removed %s tags from certificate %s.", len(tags), certificate_arn
            )
        except ClientError:
            logger.exception(
                "Couldn't remove tags from certificate %s.", certificate_arn
            )
            raise
```

- Para obter detalhes da API, consulte a RemoveTagsFromCertificate Referência da API AWS SDK for Python (Boto3).

`RequestCertificate`

O código de exemplo a seguir mostra como usar `RequestCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def request_validation(
        self, domain, alternate_domains, method, validation_domains=None
    ):
        """
        Starts a validation request that results in a new certificate being issued
        by ACM. DNS validation requires that you add CNAME records to your DNS
        provider. Email validation sends email to a list of email addresses that
        are associated with the domain.

        For more information, see _Issuing and managing certificates_ in the ACM
        user guide.
            https://docs.aws.amazon.com/acm/latest/userguide/gs.html

        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternate Names (SANs) for the
 certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation,
 when
```

```
                                    the email domain differs from the primary domain
of
                                    the certificate.
        :return: The ARN of the requested certificate.
        """
        try:
            kwargs = {
                "DomainName": domain,
                "ValidationMethod": method,
                "SubjectAlternativeNames": alternate_domains,
            }
            if validation_domains is not None:
                kwargs["DomainValidationOptions"] = [
                    {"DomainName": key, "ValidationDomain": value}
                    for key, value in validation_domains.items()
                ]
            response = self.acm_client.request_certificate(**kwargs)
            certificate_arn = response["CertificateArn"]
            logger.info(
                "Requested %s validation for domain %s. Certificate ARN is %s.",
                method,
                domain,
                certificate_arn,
            )
        except ClientError:
            logger.exception(
                "Request for %s validation of domain %s failed.", method, domain
            )
            raise
        else:
            return certificate_arn
```

- Para obter detalhes da API, consulte a RequestCertificate Referência da API AWS SDK for Python (Boto3).

`ResendValidationEmail`

O código de exemplo a seguir mostra como usar `ResendValidationEmail`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def resend_validation_email(self, certificate_arn, domain, validation_domain):
        """
        Request that validation email is sent again, for a certificate that was
        previously requested with email validation.

        :param certificate_arn: The ARN of the certificate.
        :param domain: The primary domain of the certificate.
        :param validation_domain: Alternate domain to use for determining email
                                  addresses to use for validation.
        """
        try:
            self.acm_client.resend_validation_email(
                CertificateArn=certificate_arn,
                Domain=domain,
                ValidationDomain=validation_domain,
            )
            logger.info(
                "Validation email resent to validation domain %s.",
 validation_domain
            )
        except ClientError:
            logger.exception(
```

```
            "Couldn't resend validation email to %s.", validation_domain
        )
        raise
```

- Para obter detalhes da API, consulte a ResendValidationEmail Referência da API AWS SDK for Python (Boto3).

Exemplos da API Gateway usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com API Gateway.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateDeployment`

O código de exemplo a seguir mostra como usar `CreateDeployment`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def deploy_api(self, stage_name):
        """
        Deploys a REST API. After a REST API is deployed, it can be called from any
        REST client, such as the Python Requests package or Postman.

        :param stage_name: The stage of the API to deploy, such as 'test'.
        :return: The base URL of the deployed REST API.
        """
        try:
            self.apig_client.create_deployment(
                restApiId=self.api_id, stageName=stage_name
            )
            self.stage = stage_name
            logger.info("Deployed stage %s.", stage_name)
        except ClientError:
            logger.exception("Couldn't deploy stage %s.", stage_name)
            raise
```

```
        else:
            return self.api_url()



    def api_url(self, resource=None):
        """
        Builds the REST API URL from its parts.

        :param resource: The resource path to append to the base URL.
        :return: The REST URL to the specified resource.
        """
        url = (
            f"https://{self.api_id}.execute-api.{self.apig_client.meta.region_name}"
            f".amazonaws.com/{self.stage}"
        )
        if resource is not None:
            url = f"{url}/{resource}"
        return url
```

- Para obter detalhes da API, consulte a CreateDeployment Referência da API AWS SDK for Python (Boto3).

`CreateResource`

O código de exemplo a seguir mostra como usar `CreateResource`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
```

```
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_rest_resource(self, parent_id, resource_path):
        """
        Adds a resource to a REST API.

        :param parent_id: The ID of the parent resource.
        :param resource_path: The path of the new resource, relative to the parent.
        :return: The ID of the new resource.
        """
        try:
            result = self.apig_client.create_resource(
                restApiId=self.api_id, parentId=parent_id, pathPart=resource_path
            )
            resource_id = result["id"]
            logger.info("Created resource %s.", resource_path)
        except ClientError:
            logger.exception("Couldn't create resource %s.", resource_path)
            raise
        else:
            return resource_id
```

- Para obter detalhes da API, consulte a CreateResource Referência da API AWS SDK for Python (Boto3).

`CreateRestApi`

O código de exemplo a seguir mostra como usar `CreateRestApi`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise
```

```
        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err

        return self.api_id
```

- Para obter detalhes da API, consulte a CreateRestApi Referência da API AWS SDK for Python (Boto3).

`DeleteRestApi`

O código de exemplo a seguir mostra como usar `DeleteRestApi`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
```

```
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def delete_rest_api(self):
        """
        Deletes a REST API, including all of its resources and configuration.
        """
        try:
            self.apig_client.delete_rest_api(restApiId=self.api_id)
            logger.info("Deleted REST API %s.", self.api_id)
            self.api_id = None
        except ClientError:
            logger.exception("Couldn't delete REST API %s.", self.api_id)
            raise
```

- Para obter detalhes da API, consulte a DeleteRestApi Referência da API AWS SDK for Python (Boto3).

`GetResources`

O código de exemplo a seguir mostra como usar `GetResources`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
```

```
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise

        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err

        return self.api_id
```

- Para obter detalhes da API, consulte a GetResources Referência da API AWS SDK for Python (Boto3).

GetRestApis

O código de exemplo a seguir mostra como usar `GetRestApis`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def get_rest_api_id(self, api_name):
        """
        Gets the ID of a REST API from its name by searching the list of REST APIs
        for the current account. Because names need not be unique, this returns only
        the first API with the specified name.

        :param api_name: The name of the API to look up.
```

```
        :return: The ID of the specified API.
        """
        try:
            rest_api = None
            paginator = self.apig_client.get_paginator("get_rest_apis")
            for page in paginator.paginate():
                rest_api = next(
                    (item for item in page["items"] if item["name"] == api_name),
None
                )
                if rest_api is not None:
                    break
            self.api_id = rest_api["id"]
            logger.info("Found ID %s for API %s.", rest_api["id"], api_name)
        except ClientError:
            logger.exception("Couldn't find ID for API %s.", api_name)
            raise
        else:
            return rest_api["id"]
```

- Para obter detalhes da API, consulte a GetRestApis Referência da API AWS SDK for Python (Boto3).

`PutIntegration`

O código de exemplo a seguir mostra como usar `PutIntegration`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
```

```
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
```

```
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
```

```
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Para obter detalhes da API, consulte a PutIntegration Referência da API AWS SDK for Python (Boto3).

`PutIntegrationResponse`

O código de exemplo a seguir mostra como usar `PutIntegrationResponse`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
```

```
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
```

```
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
```

```
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Para obter detalhes da API, consulte a PutIntegrationResponse Referência da API AWS SDK for Python (Boto3).

`PutMethod`

O código de exemplo a seguir mostra como usar `PutMethod`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
```

```
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
```

```
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Para obter detalhes da API, consulte a PutMethod Referência da API AWS SDK for Python (Boto3).

`PutMethodResponse`

O código de exemplo a seguir mostra como usar `PutMethodResponse`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
```

```
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
```

```
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Para obter detalhes da API, consulte a PutMethodResponse Referência da API AWS SDK for Python (Boto3).

Cenários

Crie uma API REST para rastrear COVID-19 dados

O exemplo de código a seguir mostra como criar uma API REST que simula um sistema para rastrear casos diários de COVID-19 nos Estados Unidos usando dados fictícios.

SDK para Python (Boto3)

Mostra como usar o AWS Chalice com o AWS SDK para Python (Boto3) para criar uma API REST sem servidor que usa o Amazon API Gateway e o Amazon AWS Lambda DynamoDB. A API REST simula um sistema que rastreia casos diários de COVID-19 nos Estados Unidos, usando dados fictícios. Aprenda como:

- Use o AWS Chalice para definir rotas em funções do Lambda que são chamadas para lidar com solicitações REST que vêm por meio do API Gateway.
- Usar as funções do Lambda para recuperar e armazenar dados em uma tabela do DynamoDB para atender a solicitações REST.
- Defina a estrutura da tabela e os recursos da função de segurança em um AWS CloudFormation modelo.
- Use o AWS Chalice e CloudFormation empacote e implante todos os recursos necessários.
- Use CloudFormation para limpar todos os recursos criados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice para criar uma API REST apoiada por um banco de dados Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Use AWS Secrets Manager para gerenciar as credenciais do banco de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST sem servidor no Amazon API Gateway e. AWS Lambda
- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Criar uma aplicação de chat websocket

O exemplo de código a seguir mostra como criar uma aplicação de chat que é atendido por uma API de Websocket criada no Amazon API Gateway.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon API Gateway V2 para criar uma API de websocket que se integre com AWS Lambda o Amazon DynamoDB.

- Crie uma API de Websocket atendida pelo API Gateway.
- Defina um manipulador do Lambda que armazena conexões no DynamoDB e publica mensagens para outros participantes do chat.

- Conecte-se à aplicação de chat websocket e envie mensagens com o pacote Websockets.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Criar e implantar uma API REST

O exemplo de código a seguir mostra como:

- Criar uma API REST atendida pelo API Gateway.
- Adicionar recursos à API REST para representar um perfil de usuário.
- Adicionar métodos de integração para que a API REST use uma tabela do DynamoDB para armazenar dados de perfil de usuário.
- Enviar solicitações HTTP para a API REST para adicionar e recuperar perfis de usuário.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva operações do API Gateway.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests
```

```
logger = logging.getLogger(__name__)


class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise

        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
```

```
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err

        return self.api_id


    def add_rest_resource(self, parent_id, resource_path):
        """
        Adds a resource to a REST API.

        :param parent_id: The ID of the parent resource.
        :param resource_path: The path of the new resource, relative to the parent.
        :return: The ID of the new resource.
        """
        try:
            result = self.apig_client.create_resource(
                restApiId=self.api_id, parentId=parent_id, pathPart=resource_path
            )
            resource_id = result["id"]
            logger.info("Created resource %s.", resource_path)
        except ClientError:
            logger.exception("Couldn't create resource %s.", resource_path)
            raise
        else:
            return resource_id


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.
```

```
        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
```

```
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise


    def deploy_api(self, stage_name):
        """
        Deploys a REST API. After a REST API is deployed, it can be called from any
        REST client, such as the Python Requests package or Postman.

        :param stage_name: The stage of the API to deploy, such as 'test'.
        :return: The base URL of the deployed REST API.
        """
        try:
            self.apig_client.create_deployment(
                restApiId=self.api_id, stageName=stage_name
            )
            self.stage = stage_name
            logger.info("Deployed stage %s.", stage_name)
```

```
        except ClientError:
            logger.exception("Couldn't deploy stage %s.", stage_name)
            raise
        else:
            return self.api_url()



    def api_url(self, resource=None):
        """
        Builds the REST API URL from its parts.

        :param resource: The resource path to append to the base URL.
        :return: The REST URL to the specified resource.
        """
        url = (
            f"https://{self.api_id}.execute-api.{self.apig_client.meta.region_name}"
            f".amazonaws.com/{self.stage}"
        )
        if resource is not None:
            url = f"{url}/{resource}"
        return url
```

Implantar uma API REST e chamá-la com o pacote Requests.

```
def usage_demo(table_name, role_name, rest_api_name):
    """
    Demonstrates how to used API Gateway to create and deploy a REST API, and how
    to use the Requests package to call it.

    :param table_name: The name of the demo DynamoDB table.
    :param role_name: The name of the demo role that grants API Gateway permission
 to
                      call DynamoDB.
    :param rest_api_name: The name of the demo REST API created by the demo.
    """
    gateway = ApiGatewayToService(boto3.client("apigateway"))
    role = boto3.resource("iam").Role(role_name)

    print("Creating REST API in API Gateway.")
    gateway.create_rest_api(rest_api_name)
```

```
    print("Adding resources to the REST API.")
    profiles_id = gateway.add_rest_resource(gateway.root_id, "profiles")
    username_id = gateway.add_rest_resource(profiles_id, "{username}")

    # The DynamoDB service requires that all integration requests use POST.
    print("Adding integration methods to read and write profiles in Amazon
DynamoDB.")
    gateway.add_integration_method(
        profiles_id,
        "GET",
        "dynamodb",
        "Scan",
        "POST",
        role.arn,
        {"TableName": table_name},
    )
    gateway.add_integration_method(
        profiles_id,
        "POST",
        "dynamodb",
        "PutItem",
        "POST",
        role.arn,
        {
            "TableName": table_name,
            "Item": {
                "username": {"S": "$input.path('$.username')"},
                "name": {"S": "$input.path('$.name')"},
                "title": {"S": "$input.path('$.title')"},
            },
        },
    )
    gateway.add_integration_method(
        username_id,
        "GET",
        "dynamodb",
        "GetItem",
        "POST",
        role.arn,
        {
            "TableName": table_name,
            "Key": {"username": {"S": "$method.request.path.username"}},
        },
```

```
    )

    stage = "test"
    print(f"Deploying the {stage} stage.")
    gateway.deploy_api(stage)

    profiles_url = gateway.api_url("profiles")
    print(
        f"Using the Requests package to post some people to the profiles REST API at
 "
        f"{profiles_url}."
    )
    requests.post(
        profiles_url,
        json={"username": "will", "name": "William Shakespeare", "title":
 "playwright"},
    )
    requests.post(
        profiles_url,
        json={
            "username": "ludwig",
            "name": "Ludwig van Beethoven",
            "title": "composer",
        },
    )
    requests.post(
        profiles_url,
        json={"username": "jane", "name": "Jane Austen", "title": "author"},
    )
    print("Getting the list of profiles from the REST API.")
    profiles = requests.get(profiles_url).json()
    pprint(profiles)
    print(f"Getting just the profile for username 'jane' (URL: {profiles_url}/
jane).")
    jane = requests.get(f"{profiles_url}/jane").json()
    pprint(jane)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDeployment

- CreateResource
- CreateRestApi
- DeleteRestApi
- GetResources
- GetRestApis
- PutIntegration
- PutIntegrationResponse
- PutMethod
- PutMethodResponse

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para Python (Boto3)

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.
- Use o pacote Requests para enviar solicitações à API REST.
- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

- Amazon SNS

Exemplos do Application Recovery Controller usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Application Recovery Controller.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetRoutingControlState`

O código de exemplo a seguir mostra como usar `GetRoutingControlState`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
```

```
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def get_routing_control_state(routing_control_arn, cluster_endpoints):
    """
    Gets the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to look up.
    :param cluster_endpoints: The list of cluster endpoints to query.
    :return: The routing control state response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get or
 set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.get_routing_control_state(
                RoutingControlArn=routing_control_arn
            )
            return response
        except Exception as error:
            print(error)
            raise error
```

- Para obter detalhes da API, consulte a GetRoutingControlState Referência da API AWS SDK for Python (Boto3).

`UpdateRoutingControlState`

O código de exemplo a seguir mostra como usar `UpdateRoutingControlState`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def update_routing_control_state(
    routing_control_arn, cluster_endpoints, routing_control_state
):
    """
    Updates the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.
```

```
    :param routing_control_arn: The ARN of the routing control to update the state
 for.
    :param cluster_endpoints: The list of cluster endpoints to try.
    :param routing_control_state: The new routing control state.
    :return: The routing control update response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get or
 set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.update_routing_control_state(
                RoutingControlArn=routing_control_arn,
                RoutingControlState=routing_control_state,
            )
            return response
        except Exception as error:
            print(error)
```

- Para obter detalhes da API, consulte a UpdateRoutingControlState Referência da API AWS SDK for Python (Boto3).

Exemplos do Audit Manager usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Audit Manager.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie uma estrutura personalizada a partir de um AWS Config pacote de conformidade

O exemplo de código a seguir mostra como:

- Obtenha uma lista dos pacotes de AWS Config conformidade.
- Criar um controle personalizado do Audit Manager para cada regra gerenciada em um pacote de conformidade.
- Criar uma estrutura personalizada do Audit Manager que contenha os controles.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class ConformancePack:
    def __init__(self, config_client, auditmanager_client):
        self.config_client = config_client
        self.auditmanager_client = auditmanager_client

    def get_conformance_pack(self):
        """
        Return a selected conformance pack from the list of conformance packs.

        :return: selected conformance pack
        """
        try:
            conformance_packs = self.config_client.describe_conformance_packs()
            print(
                "Number of conformance packs fetched: ",
```

```
                len(conformance_packs.get("ConformancePackDetails")),
            )
            print("Fetched the following conformance packs: ")
            all_cpack_names = {
                cp["ConformancePackName"]
                for cp in conformance_packs.get("ConformancePackDetails")
            }
            for pack in all_cpack_names:
                print(f"\t{pack}")
            cpack_name = input(
                "Provide ConformancePackName that you want to create a custom "
                "framework for: "
            )
            if cpack_name not in all_cpack_names:
                print(f"{cpack_name} is not in the list of conformance packs!")
                print(
                    "Provide a conformance pack name from the available list of "
                    "conformance packs."
                )
                raise Exception("Invalid conformance pack")
            print("-" * 88)
        except ClientError:
            logger.exception("Couldn't select conformance pack.")
            raise
        else:
            return cpack_name

    def create_custom_controls(self, cpack_name):
        """
        Create custom controls for all managed AWS Config rules in a conformance
pack.

        :param cpack_name: The name of the conformance pack to create controls for.
        :return: The list of custom control IDs.
        """
        try:
            rules_in_pack = self.config_client.describe_conformance_pack_compliance(
                ConformancePackName=cpack_name
            )
            print(
                "Number of rules in the conformance pack: ",
                len(rules_in_pack.get("ConformancePackRuleComplianceList")),
            )
            for rule in rules_in_pack.get("ConformancePackRuleComplianceList"):
```

```
                print(f"\t{rule.get('ConfigRuleName')}")
            print("-" * 88)
            print(
                "Creating a custom control for each rule and a custom framework "
                "consisting of these rules in Audit Manager."
            )
            am_controls = []
            for rule in rules_in_pack.get("ConformancePackRuleComplianceList"):
                config_rule = self.config_client.describe_config_rules(
                    ConfigRuleNames=[rule.get("ConfigRuleName")]
                )
                source_id = (
                    config_rule.get("ConfigRules")[0]
                    .get("Source", {})
                    .get("SourceIdentifier")
                )
                custom_control = self.auditmanager_client.create_control(
                    name="Config-" + rule.get("ConfigRuleName"),
                    controlMappingSources=[
                        {
                            "sourceName": "ConfigRule",
                            "sourceSetUpOption": "System_Controls_Mapping",
                            "sourceType": "AWS_Config",
                            "sourceKeyword": {
                                "keywordInputType": "SELECT_FROM_LIST",
                                "keywordValue": source_id,
                            },
                        }
                    ],
                ).get("control", {})
                am_controls.append({"id": custom_control.get("id")})
            print("Successfully created a control for each config rule.")
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom controls.")
            raise
        else:
            return am_controls

    def create_custom_framework(self, cpack_name, am_control_ids):
        """
        Create a custom Audit Manager framework from a selected AWS Config
conformance
        pack.
```

```
        :param cpack_name: The name of the conformance pack to create a framework
 from.
        :param am_control_ids: The IDs of the custom controls created from the
                               conformance pack.
        """
        try:
            print("Creating custom framework...")
            custom_framework = self.auditmanager_client.create_assessment_framework(
                name="Config-Conformance-pack-" + cpack_name,
                controlSets=[{"name": cpack_name, "controls": am_control_ids}],
            )
            print(
                f"Successfully created the custom framework: ",
                f"{custom_framework.get('framework').get('name')}: ",
                f"{custom_framework.get('framework').get('id')}",
            )
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom framework.")
            raise


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager custom framework demo!")
    print("-" * 88)
    print(
        "You can use this sample to select a conformance pack from AWS Config and "
        "use AWS Audit Manager to create a custom control for all the managed "
        "rules under the conformance pack. A custom framework is also created "
        "with these controls."
    )
    print("-" * 88)
    conf_pack = ConformancePack(boto3.client("config"),
 boto3.client("auditmanager"))
    cpack_name = conf_pack.get_conformance_pack()
    am_controls = conf_pack.create_custom_controls(cpack_name)
    conf_pack.create_custom_framework(cpack_name, am_controls)


if __name__ == "__main__":
    run_demo()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAssessmentFramework
  - CreateControl

Crie uma estrutura personalizada que contenha controles CSPM do Security Hub

O exemplo de código a seguir mostra como:

- Obtenha uma lista de todos os controles padrão que têm o Security Hub CSPM como fonte de dados.
- Criar uma estrutura personalizada do Audit Manager que contenha os controles.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class SecurityHub:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_sechub_controls(self):
        """
        Gets the list of controls that use Security Hub as their data source.

        :return: The list of Security Hub controls.
```

```
        """
        print("-" * 88)
        next_token = None
        page = 1
        sechub_control_list = []
        while True:
            print("Page [" + str(page) + "]")
            if next_token is None:
                control_list = self.auditmanager_client.list_controls(
                    controlType="Standard", maxResults=100
                )
            else:
                control_list = self.auditmanager_client.list_controls(
                    controlType="Standard", nextToken=next_token, maxResults=100
                )
            print("Total controls found:",
len(control_list.get("controlMetadataList")))
            for control in control_list.get("controlMetadataList"):
                control_details = self.auditmanager_client.get_control(
                    controlId=control.get("id")
                ).get("control", {})
                if "AWS Security Hub" in control_details.get("controlSources"):
                    sechub_control_list.append({"id": control_details.get("id")})
            next_token = control_list.get("nextToken")
            if not next_token:
                break
            page += 1
        print("Number of Security Hub controls found: ", len(sechub_control_list))
        return sechub_control_list

    def create_custom_framework(self, am_controls):
        """
        Create a custom framework with a list of controls.

        :param am_controls: The list of controls to include in the framework.
        """
        try:
            print("Creating custom framework...")
            custom_framework = self.auditmanager_client.create_assessment_framework(
                name="All Security Hub Controls Framework",
                controlSets=[{"name": "Security-Hub", "controls": am_controls}],
            )
            print(
                f"Successfully created the custom framework: "
```

```
                f"{custom_framework.get('framework').get('name')}: "
                f"{custom_framework.get('framework').get('id')}"
            )
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom framework.")
            raise


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager Security Hub demo!")
    print("-" * 88)
    print(" This script creates a custom framework with all Security Hub controls.")
    print("-" * 88)
    sechub = SecurityHub(boto3.client("auditmanager"))
    am_controls = sechub.get_sechub_controls()
    sechub.create_custom_framework(am_controls)


if __name__ == "__main__":
    run_demo()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAssessmentFramework
  - GetControl
  - ListControls

Criar um relatório de avaliação

O exemplo de código a seguir mostra como criar um relatório de avaliação do Audit Manager que contenha um dia de evidência.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import dateutil.parser
import logging
import time
import urllib.request
import uuid
import boto3
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


class AuditReport:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_input(self):
        print("-" * 40)
        try:
            assessment_id = input("Provide assessment id [uuid]: ").lower()
            try:
                assessment_uuid = uuid.UUID(assessment_id)
            except ValueError:
                logger.error("Assessment Id is not a valid UUID: %s", assessment_id)
                raise
            evidence_folder = input("Provide evidence date [yyyy-mm-dd]: ")
            try:
                evidence_date = dateutil.parser.parse(evidence_folder).date()
            except ValueError:
                logger.error("Invalid date : %s", evidence_folder)
                raise
            try:
                self.auditmanager_client.get_assessment(
                    assessmentId=str(assessment_uuid)
```

```
                )
            except ClientError:
                logger.exception("Couldn't get assessment %s.", assessment_uuid)
                raise
        except (ValueError, ClientError):
            return None, None
        else:
            return assessment_uuid, evidence_date

    def clear_staging(self, assessment_uuid, evidence_date):
        """
        Find all the evidence in the report and clear it.
        """
        next_token = None
        page = 1
        interested_folder_id_list = []
        while True:
            print(f"Page [{page}]")
            if next_token is None:
                folder_list = (
                    self.auditmanager_client.get_evidence_folders_by_assessment(
                        assessmentId=str(assessment_uuid), maxResults=1000
                    )
                )
            else:
                folder_list = (
                    self.auditmanager_client.get_evidence_folders_by_assessment(
                        assessmentId=str(assessment_uuid),
                        nextToken=next_token,
                        maxResults=1000,
                    )
                )
            folders = folder_list.get("evidenceFolders")
            print(f"Got {len(folders)} folders.")
            for folder in folders:
                folder_id = folder.get("id")
                if folder.get("name") == str(evidence_date):
                    interested_folder_id_list.append(folder_id)
                if folder.get("assessmentReportSelectionCount") == folder.get(
                    "totalEvidence"
                ):
                    print(
                        f"Removing folder from report selection :
{folder.get('name')} "
```

```
                        f"{folder_id} {folder.get('controlId')}"
                    )

self.auditmanager_client.disassociate_assessment_report_evidence_folder(
                        assessmentId=str(assessment_uuid),
evidenceFolderId=folder_id
                    )
                elif folder.get("assessmentReportSelectionCount") > 0:
                    # Get all evidence in the folder and
                    # add selected evidence in the selected_evidence_list.
                    evidence_list = (
                        self.auditmanager_client.get_evidence_by_evidence_folder(
                            assessmentId=str(assessment_uuid),
                            controlSetId=folder_id,
                            evidenceFolderId=folder_id,
                            maxResults=1000,
                        )
                    )
                    selected_evidence_list = []
                    for evidence in evidence_list.get("evidence"):
                        if evidence.get("assessmentReportSelection") == "Yes":
                            selected_evidence_list.append(evidence.get("id"))
                    print(
                        f"Removing evidence report selection : {folder.get('name')}
"
                        f"{len(selected_evidence_list)}"
                    )

self.auditmanager_client.batch_disassociate_assessment_report_evidence(
                        assessmentId=str(assessment_uuid),
                        evidenceFolderId=folder_id,
                        evidenceIds=selected_evidence_list,
                    )
            next_token = folder_list.get("nextToken")
            if not next_token:
                break
            page += 1
        return interested_folder_id_list

    def add_folder_to_staging(self, assessment_uuid, folder_id_list):
        print(f"Adding folders to report : {folder_id_list}")
        for folder in folder_id_list:
            self.auditmanager_client.associate_assessment_report_evidence_folder(
                assessmentId=str(assessment_uuid), evidenceFolderId=folder
```

```
            )

    def get_report(self, assessment_uuid):
        report = self.auditmanager_client.create_assessment_report(
            name="ReportViaScript",
            description="testing",
            assessmentId=str(assessment_uuid),
        )
        if self._is_report_generated(report.get("assessmentReport").get("id")):
            report_url = self.auditmanager_client.get_assessment_report_url(
                assessmentReportId=report.get("assessmentReport").get("id"),
                assessmentId=str(assessment_uuid),
            )
            print(report_url.get("preSignedUrl"))
            urllib.request.urlretrieve(
                report_url.get("preSignedUrl").get("link"),
                report_url.get("preSignedUrl").get("hyperlinkName"),
            )
            print(
                f"Report saved as
{report_url.get('preSignedUrl').get('hyperlinkName')}."
            )
        else:
            print("Report generation did not finish in 15 minutes.")
            print(
                "Failed to download report. Go to the console and manually download
"
                "the report."
            )

    def _is_report_generated(self, assessment_report_id):
        max_wait_time = 0
        while max_wait_time < 900:
            print(f"Checking status of the report {assessment_report_id}")
            report_list =
self.auditmanager_client.list_assessment_reports(maxResults=1)
            if (
                report_list.get("assessmentReports")[0].get("id")
                == assessment_report_id
                and report_list.get("assessmentReports")[0].get("status") ==
"COMPLETE"
            ):
                return True
            print("Sleeping for 5 seconds...")
```

```
            time.sleep(5)
            max_wait_time += 5


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager samples demo!")
    print("-" * 88)
    print(
        "This script creates an assessment report for an assessment with all the "
        "evidence collected on the provided date."
    )
    print("-" * 88)

    report = AuditReport(boto3.client("auditmanager"))
    assessment_uuid, evidence_date = report.get_input()
    if assessment_uuid is not None and evidence_date is not None:
        folder_id_list = report.clear_staging(assessment_uuid, evidence_date)
        report.add_folder_to_staging(assessment_uuid, folder_id_list)
        report.get_report(assessment_uuid)


if __name__ == "__main__":
    run_demo()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AssociateAssessmentReportEvidenceFolder
  - BatchDisassociateAssessmentReportEvidence
  - CreateAssessmentReport
  - DisassociateAssessmentReportEvidenceFolder
  - GetAssessment
  - GetAssessmentReportUrl
  - GetEvidenceByEvidenceFolder
  - GetEvidenceFoldersByAssessment
  - ListAssessmentReports

Exemplos de Aurora usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Aurora.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Aurora

O exemplo de código a seguir mostra como começar a usar o Aurora.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
```

```
# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de cluster do banco de dados do Aurora e definir os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Crie um snapshot do cluster do banco de dados e limpe os recursos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database engine
 and
        create a DB cluster parameter group that is compatible with a selected
 engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created parameter
 group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
 {parameter_group_name}."
```

```
        )
        parameter_group =
self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ", families)
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group {parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
```

```
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name, update_params)
        print(
            "You can get a list of parameters you've set by specifying a source of
'user'."
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster parameter
group.

        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the DB
cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ", q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters): ",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
```

```
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new DB
cluster
        contains no DB instances, so you must add one. The first DB instance that is
added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
```

```
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"] + ", storage type: " + opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
minutes."
            )
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on how
to
        connect to it.

        :param cluster: The DB cluster to display.
        """
        print(
```

```
            "You can now connect to your database using your favorite MySql client.
\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora.Co
        )
        print("-" * 88)

    def create_snapshot(self, cluster_name):
        """
        Shows how to create a DB cluster snapshot and wait until it's available.

        :param cluster_name: The name of a DB cluster to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB cluster (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.aurora_wrapper.create_cluster_snapshot(
                snapshot_id, cluster_name
            )
            while snapshot.get("Status") != "available":
                wait(30)
                snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)
```

```
    def cleanup(self, db_inst, cluster, parameter_group):
        """
        Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
        Before the DB cluster parameter group can be deleted, all associated DB
instances and
        DB clusters must first be deleted.

        :param db_inst: The DB instance to delete.
        :param cluster: The DB cluster to delete.
        :param parameter_group: The DB cluster parameter group to delete.
        """
        cluster_name = cluster["DBClusterIdentifier"]
        parameter_group_name = parameter_group["DBClusterParameterGroupName"]
        if q.ask(
            "\nDo you want to delete the database instance, DB cluster, and
parameter "
            "group (y/n)? ",
            q.is_yesno,
        ):
            print(f"Deleting database instance {db_inst['DBInstanceIdentifier']}.")
            self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, cluster_name, db_name):
        print("-" * 88)
        print(
```

```
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
 started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
 parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
 parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Aurora.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
```

```
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
```

```
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why: %s:
%s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created, such
as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
```

```
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
```

```
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

```
    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
```

```
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list
of
                                       engine versions to those that are compatible
with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by the
DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
```

```
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
```

```
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters

- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created, such
as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
```

```
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Python (Boto3).

`CreateDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
```

```
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateDBClusterParameterGroup Referência da API AWS SDK for Python (Boto3).

`CreateDBClusterSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBClusterSnapshot`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
 DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Para obter detalhes da API, consulte a CreateDBClusterSnapshot Referência da API AWS SDK for Python (Boto3).

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Python (Boto3).

`DeleteDBCluster`

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
```

```
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Python (Boto3).

`DeleteDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBClusterParameterGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteDBClusterParameterGroup Referência da API AWS SDK for Python (Boto3).

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Python (Boto3).

`DescribeDBClusterParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameterGroups`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameterGroups Referência da API AWS SDK for Python (Boto3).

`DescribeDBClusterParameters`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameters`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
 filtered
                            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
 retrieved.
                       For example, a source of 'user' retrieves only parameters
 that
                       were set by a user.
        :return: The list of requested parameters.
        """
```

```
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameters Referência da API AWS SDK for Python (Boto3).

`DescribeDBClusterSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBClusterSnapshots`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Para obter detalhes da API, consulte a DescribeDBClusterSnapshots Referência da API AWS SDK for Python (Boto3).

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
```

```
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why: %s: 
%s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Python (Boto3).

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
```

```
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list
of
                                       engine versions to those that are compatible
with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersions Referência da API AWS SDK for Python (Boto3).

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
```

```
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by the
DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return inst_opts
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptions Referência da API AWS SDK for Python (Boto3).

`ModifyDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
```

```
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a ModifyDBClusterParameterGroup Referência da API AWS SDK for Python (Boto3).

Cenários

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice para criar uma API REST apoiada por um banco de dados Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Use AWS Secrets Manager para gerenciar as credenciais do banco de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST sem servidor no Amazon API Gateway e. AWS Lambda
- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um AWS Secrets Manager segredo que contenha credenciais do banco de dados e use-o para autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Auto Scaling usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com Auto Scaling.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Auto Scaling

O exemplo de código a seguir mostra como começar a usar o Auto Scaling.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling client
 and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroups Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo do Amazon EC2 Auto Scaling com um modelo de inicialização e zonas de disponibilidade e obter informações sobre instâncias em execução.
- Ative a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas para CloudWatch métricas e, em seguida, limpe os recursos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) -> None:
    """
    Runs the scenario demonstrating the management of Auto Scaling groups and
 instances.

    :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
 Scaling groups.
    :param svc_helper: An instance of the ServiceHelper that interacts with AWS
 services.
    :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")
```

```
    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
instances."
    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
new one: "
    )
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
    print("-" * 88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask(
        "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) + 1)
    )
    group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_autoscaling_group(group_name, group_zones, template_name, 1,
1)
    wait(10)
```

```
    group = as_wrapper.describe_group(group_name)
    logger.info("Created Auto Scaling group %s.", group_name)
    print("Created group:")
    pp(group)
    print("Waiting for instance to start...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    use_metrics = q.ask(
        "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
        q.is_yesno,
    )
    if use_metrics:
        as_wrapper.enable_metrics(
            group_name,
            [
                "GroupMinSize",
                "GroupMaxSize",
                "GroupDesiredCapacity",
                "GroupInServiceInstances",
                "GroupTotalInstances",
            ],
        )
        logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
        print(f"Metrics enabled for {group_name}.")
    print("-" * 88)

    print(f"Let's update the maximum number of instances in {group_name} from 1 to
3.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.update_group(group_name, MaxSize=3)
    group = as_wrapper.describe_group(group_name)
    logger.info("Updated maximum size for group %s to 3.", group_name)
    print("The group still has one running instance, but can have up to three:")
    print_simplified_group(group)
    print("-" * 88)

    print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.set_desired_capacity(group_name, 2)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Set desired capacity for group %s to 2.", group_name)
```

```
    print("Here's the current state of the group:")
    print_simplified_group(group)
    print("-" * 88)
    print("Waiting for the new instance to start...")
    instance_ids = wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's terminate one of the instances in {group_name}.")
    print("Because the desired capacity is 2, another instance will start.")
    print("The currently running instances are:")
    for index, inst_id in enumerate(instance_ids):
        print(f"\t{index+1}. {inst_id}")
    inst_sel = q.ask(
        "Which instance do you want to stop? ",
        q.is_int,
        q.in_range(1, len(instance_ids) + 1),
    )
    print(f"Stopping {instance_ids[inst_sel-1]}...")
    as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info(
        "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
group_name
    )
    print(f"Here's the state of {group_name}:")
    print_simplified_group(group)
    print("Waiting for the scaling activities to complete...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's get a report of scaling activities for {group_name}.")
    q.ask("Press Enter when you're ready.")
    activities = as_wrapper.describe_scaling_activities(group_name)
    logger.info(
        "Retrieved %d scaling activities for group %s.", len(activities), group_name
    )
    print(
        f"Found {len(activities)} activities.\n"
        f"Activities are ordered with the most recent one first:"
    )
    for act in activities:
        pp(act)
    print("-" * 88)
```

```
    if use_metrics:
        print("Let's look at CloudWatch metrics.")
        metric_namespace = "AWS/AutoScaling"
        metric_dimensions = [{"Name": "AutoScalingGroupName", "Value": group_name}]
        print(f"The following metrics are enabled for {group_name}:")
        done = False
        while not done:
            metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
            for index, metric in enumerate(metrics):
                print(f"\t{index+1}. {metric.name}")
            print(f"\t{len(metrics)+1}. None")
            metric_sel = q.ask(
                "Which metric do you want to see? ",
                q.is_int,
                q.in_range(1, len(metrics) + 1),
            )
            if metric_sel < len(metrics) + 1:
                span = 5
                metric = metrics[metric_sel - 1]
                print(f"Over the last {span} minutes, {metric.name} recorded:")
                # CloudWatch metric times are in the UTC+0 time zone.
                now = datetime.now(timezone.utc)
                metric_data = svc_helper.get_metric_statistics(
                    metric_dimensions, metric, now - timedelta(minutes=span), now
                )
                pp(metric_data)
                if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
                    done = True
            else:
                done = True

    print(f"Let's clean up.")
    q.ask("Press Enter when you're ready.")
    if use_metrics:
        print(f"Stopping metrics collection for {group_name}.")
        as_wrapper.disable_metrics(group_name)
        logger.info("Disabled metrics collection for group %s.", group_name)

    print(
        "You must terminate all instances in the group before you can delete the
group."
    )
```

```
    print("Set minimum size to 0.")
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
        logger.info("Terminated instance %s in group %s.", inst_id, group_name)
    print("Waiting for instances to stop...")
    wait_for_instances(instance_ids, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_autoscaling_group(group_name)
    logger.info("Deleted Auto Scaling group %s.", group_name)
    print("-" * 88)

    if template is not None:
        if q.ask(
            f"Do you want to delete launch template {template_name} used in this
 demo (y/n)? "
        ):
            svc_helper.delete_template(template_name)
            logger.info("Deleted launch template %s.", template_name)
            print("Template deleted.")

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")
```

Defina as funções que são chamadas pelo cenário para gerenciar modelos e métricas de lançamento. Essas funções envolvem o Amazon EC2 e CloudWatch as ações.

```
class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""
```

```
    def __init__(self, ec2_client, cloudwatch_resource):
        """
        :param ec2_client: A Boto3 Amazon EC2 client.
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.ec2_client = ec2_client
        self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for instances
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateNames=[template_name]
            )
            template = response["LaunchTemplates"][0]
            logger.info("Launch template %s retrieved successfully.", template_name)
            return template
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                logger.warning("Launch template %s does not exist.", template_name)
            else:
                logger.error(
                    "Couldn't verify launch template %s. Error: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
```

```
        :param template_name: The name to give to the template.
        :param inst_type: The type of the instance, such as t1.micro.
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when creating
                       an instance.
        :return: Information about the newly created template.
        :raises ClientError: If there is an error creating the launch template.
        """
        try:
            response = self.ec2_client.create_launch_template(
                LaunchTemplateName=template_name,
                LaunchTemplateData={"InstanceType": inst_type, "ImageId": ami_id},
            )
            template = response["LaunchTemplate"]
            logger.info(
                "Created launch template %s with instance type %s and AMI ID %s.",
                template_name,
                inst_type,
                ami_id,
            )
            return template
        except ClientError as err:
            logger.error(
                "Couldn't create launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete_template(self, template_name: str) -> None:
        """
        Deletes a launch template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If there is an error deleting the launch template.
        """
        try:
            self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
            logger.info("Deleted launch template %s.", template_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise

    def get_availability_zones(self) -> list:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        :raises ClientError: If there is an error retrieving availability zones.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            logger.info("Retrieved availability zones: %s.", ", ".join(zones))
            return zones
        except ClientError as err:
            logger.error(
                "Couldn't get availability zones. Error: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_metrics(self, namespace: str, dimensions: list) -> list:
        """
        Gets a list of CloudWatch metrics filtered by namespace and dimensions.

        :param namespace: The namespace of the metrics to look up.
        :param dimensions: The dimensions of the metrics to look up.
        :return: The list of metrics.
        :raises ClientError: If there is an error retrieving CloudWatch metrics.
        """
        try:
            metrics = list(
                self.cloudwatch_resource.metrics.filter(
                    Namespace=namespace, Dimensions=dimensions
                )
            )
            logger.info(
                "Retrieved metrics for namespace %s with dimensions %s.",
                namespace,
                dimensions,
```

```
            )
            return metrics
        except ClientError as err:
            logger.error(
                "Couldn't get metrics for %s, %s. Error: %s: %s",
                namespace,
                dimensions,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    @staticmethod
    def get_metric_statistics(
        dimensions: list, metric, start: datetime, end: datetime
    ) -> list:
        """
        Gets statistics for a CloudWatch metric within a specified time span.

        :param dimensions: The dimensions of the metric.
        :param metric: The metric to look up.
        :param start: The start of the time span for retrieved metrics.
        :param end: The end of the time span for retrieved metrics.
        :return: The list of data points found for the specified metric.
        :raises ClientError: If there is an error retrieving metric statistics.
        """
        try:
            response = metric.get_statistics(
                Dimensions=dimensions,
                StartTime=start,
                EndTime=end,
                Period=60,
                Statistics=["Sum"],
            )
            data = response["Datapoints"]
            logger.info("Retrieved statistics for metric %s.", metric.name)
            return data
        except ClientError as err:
            logger.error(
                "Couldn't get statistics for metric %s. Error: %s: %s",
                metric.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise


def print_simplified_group(group: dict) -> None:
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
 {group['DesiredCapacity']}"
    )
    if group["Instances"]:
        print(f"\tInstances:")
        for inst in group["Instances"]:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")


def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)


def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
```

```
    """
    ready = False
    instances = []
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids else
[]
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
            f"Here are the details of the instance{'s' if len(instances) > 1 else
''}:"
        )
        for instance in instances:
            pp(instance)
    return instance_ids
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Ações

`AttachLoadBalancerTargetGroups`

O código de exemplo a seguir mostra como usar `AttachLoadBalancerTargetGroups`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
```

```
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
```

```
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a AttachLoadBalancerTargetGroups Referência da API AWS SDK for Python (Boto3).

`CreateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `CreateAutoScalingGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
```

```
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def create_group(
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
    ) -> None:
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
        :param group_zones: The Availability Zones in which instances can be
created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
template.
                                     The launch template specifies the configuration
of
                                     instances that are created by auto scaling
activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling group.
        """
        try:
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=group_name,
                AvailabilityZones=group_zones,
                LaunchTemplate={
                    "LaunchTemplateName": launch_template_name,
                    "Version": "$Default",
                },
                MinSize=min_size,
                MaxSize=max_size,
            )
```

```
            # Wait for the group to exist.
            waiter = self.autoscaling_client.get_waiter("group_exists")
            waiter.wait(AutoScalingGroupNames=[group_name])

            logger.info(f"Successfully created Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to create Auto Scaling group {group_name}.")
            if error_code == "AlreadyExistsFault":
                logger.error(
                    f"An Auto Scaling group with the name '{group_name}' already
 exists. "
                    "Please use a different name or update the existing group.",
                )
            elif error_code == "LimitExceededFault":
                logger.error(
                    "The request failed because you have reached the limit "
                    "on the number of Auto Scaling groups or launch configurations.
 "
                    "Consider deleting unused resources or request a limit increase.
 "
                    "\nSee Auto Scaling Service Quota documentation here:"
                    "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/ec2-
auto-scaling-quotas.html"
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Para obter detalhes da API, consulte a CreateAutoScalingGroup Referência da API AWS SDK for Python (Boto3).

`DeleteAutoScalingGroup`

O código de exemplo a seguir mostra como usar `DeleteAutoScalingGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize o tamanho mínimo de um grupo do Auto Scaling para zero, encerre todas as instâncias no grupo e exclua o grupo.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
```

```
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
```

```
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DeleteAutoScalingGroup Referência da API AWS SDK for Python (Boto3).

`DescribeAutoScalingGroups`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingGroups`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
```

```
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
        """
        Gets information about an Auto Scaling group.

        :param group_name: The name of the group to look up.
        :return: A dictionary with information about the group if found, otherwise
None.
        :raises ClientError: If there is an error describing the Auto Scaling group.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_groups"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupNames=[group_name])
            groups = []
            for response in response_iterator:
                groups.extend(response.get("AutoScalingGroups", []))

            logger.info(
                f"Successfully retrieved information for Auto Scaling group
{group_name}."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to describe Auto Scaling group {group_name}.")
            if error_code == "ResourceContentionFault":
                logger.error(
                    "There is a conflict with another operation that is modifying
the "
                    f"Auto Scaling group '{group_name}' Please try again later."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

```
        else:
            return groups[0] if len(groups) > 0 else None
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroups Referência da API AWS SDK for Python (Boto3).

`DescribeAutoScalingInstances`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_instances(self, instance_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: A list of dictionaries with information about each instance,
                 or an empty list if none are found.
        :raises ClientError: If there is an error describing the instances.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
```

```
                "describe_auto_scaling_instances"
            )
            response_iterator = paginator.paginate(InstanceIds=instance_ids)

            instances = []
            for response in response_iterator:
                instances.extend(response.get("AutoScalingInstances", []))

            logger.info(f"Successfully described instances: {instance_ids}")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe instances {instance_ids}. Error code:
 {error_code}, Message: {err.response['Error']['Message']}"
            )
            raise
        else:
            return instances
```

- Para obter detalhes da API, consulte a DescribeAutoScalingInstances Referência da API AWS SDK for Python (Boto3).

`DescribeScalingActivities`

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
```

```
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_scaling_activities(self, group_name: str) -> List[Dict[str, Any]]:
        """
        Gets information about scaling activities for the group. Scaling activities
        are things like instances stopping or starting in response to user requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: A list of dictionaries representing the scaling activities for the
                 group, ordered with the most recent activity first.
        :raises ClientError: If there is an error describing the scaling activities.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_scaling_activities"
            )
            response_iterator = paginator.paginate(AutoScalingGroupName=group_name)
            activities = []
            for response in response_iterator:
                activities.extend(response.get("Activities", []))

            logger.info(
                f"Successfully described scaling activities for group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe scaling activities for group '{group_name}'.
Error code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is modifying
the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
```

```
            raise
        else:
            return activities
```

- Para obter detalhes da API, consulte a DescribeScalingActivities Referência da API AWS SDK for Python (Boto3).

`DisableMetricsCollection`

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def disable_metrics(self, group_name: str) -> Dict[str, Any]:
        """
        Stops CloudWatch metric collection for the Auto Scaling group.

        :param group_name: The name of the group.
        :return: A dictionary with the response from disabling the metrics
 collection.
        :raises ClientError: If there is an error disabling metrics collection.
        """
        try:
```

```
            response = self.autoscaling_client.disable_metrics_collection(
                AutoScalingGroupName=group_name
            )
            logger.info(
                f"Successfully disabled metrics collection for group
'{group_name}'."
            )
            return response
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is modifying
the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
```

- Para obter detalhes da API, consulte a DisableMetricsCollection Referência da API AWS SDK for Python (Boto3).

`EnableMetricsCollection`

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
```

```
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str, Any]:
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        :return: A dictionary with the response from enabling the metrics
collection.
        :raises ClientError: If there is an error enabling metrics collection.
        """
        try:
            response = self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
Granularity="1Minute"
            )
            logger.info(
                f"Successfully enabled metrics for Auto Scaling group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is modifying
the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            elif error_code == "InvalidParameterCombination":
                logger.error(
```

```
                    f"The combination of parameters provided for enabling metrics on
'{group_name}' is not valid. "
                    "Please check the parameters and try again."
                )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a EnableMetricsCollection Referência da API AWS SDK for Python (Boto3).

`SetDesiredCapacity`

O código de exemplo a seguir mostra como usar `SetDesiredCapacity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
keep the
        number of running instances equal to the desired capacity.
```

```
        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
                f"Successfully set desired capacity of {capacity} for Auto Scaling
group '{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
            )
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'. "
                    "Please wait for the activity to complete before attempting to
set the desired capacity."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Para obter detalhes da API, consulte a SetDesiredCapacity Referência da API AWS SDK for Python (Boto3).

`TerminateInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
    ) -> Dict[str, Any]:
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired capacity
                                  of the group. When passing True for this
 parameter,
                                  you can stop an instance without having a
 replacement
                                  instance start when the desired capacity threshold
 is
                                  crossed.
        :return: A dictionary containing details of the scaling activity that occurs
                 in response to this action.
        :raises ClientError: If there is an error terminating the instance.
        """
        try:
            response =
 self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
 ShouldDecrementDesiredCapacity=decrease_capacity
```

```
            )
            logger.info(f"Successfully terminated instance {instance_id}.")
            return response["Activity"]

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to terminate instance {instance_id}.")
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    "A scaling activity is currently in progress for the Auto
Scaling group "
                    f"associated with instance '{instance_id}'. "
                    "Please wait for the activity to complete before attempting to
terminate the instance."
                )
            elif error_code == "ResourceInUse":
                logger.error(
                    f"The instance '{instance_id}' or an associated resource is
currently in use "
                    "and cannot be terminated. "
                    "Ensure the instance is not involved in any ongoing processes
and try again."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Para obter detalhes da API, consulte a TerminateInstanceInAutoScalingGroup Referência da API AWS SDK for Python (Boto3).

`UpdateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `UpdateAutoScalingGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def update_group(self, group_name: str, **kwargs: Any) -> None:
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        :return: None
        :raises ClientError: If there is an error updating the Auto Scaling group.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, **kwargs
            )
            logger.info(f"Successfully updated Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to update Auto Scaling group {group_name}.")
            if error_code == "ResourceInUse":
                logger.error(
                    "The Auto Scaling group '%s' is currently in use and cannot be
 modified. Please try again later.",
                    group_name,
                )
            elif error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
 Scaling group '{group_name}'."
                    "Please wait for the activity to complete before attempting to
 update the group."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Para obter detalhes da API, consulte a UpdateAutoScalingGroup Referência da API AWS SDK for Python (Boto3).

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
class Runner:
```

```
    """
    Manages the deployment, demonstration, and destruction of resources for the
resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example, such
as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
```

```
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
```

```
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port is
 open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access from
 this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging from
 your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
```

```
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer endpoint.
Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as sending
requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer endpoint.")
                endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
```

```
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check to
 update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
 failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
 failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a non-
existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
```

```
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance replacement.")
        self.demo_choices()
```

```
        logging.info("Simulating a complete failure of the recommendation service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up manually
to avoid unexpected charges."
```

```
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM policies
 and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
```

```
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
```

```
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
```

```
    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
```

```
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to attach
 to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role {role_name}.")
```

```
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is assumed
by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
```

```
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
```

```
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
```

```
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
```

```
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
```

```
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
```

```
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
```

```
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones
```

```
    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with availability
zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.
```

```
        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) -> None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)
```

```
            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
```

```
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
```

```
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
```

```
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress from
this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
```

```
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to either
this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist. Please
check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
```

```
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}' on
port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
```

```
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
 how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
 and
        lower thresholds. In production, you might want to decrease the sensitivity
 of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
```

```
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
```

```
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
```

```
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
```

```
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener to
the
                             load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
```

```
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.
```

```
        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
```

```
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint, retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
```

```
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
 and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books, movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary parameters.
```

```
        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended, such
as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5, "WriteCapacityUnits":
5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
```

```
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name, data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}: {err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Crie uma classe que envolva as ações do Systems Manager.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters to
 drive
    the demonstration of resilient architecture, such as failure of a dependency or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                            service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
```

```
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False. "
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate

- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Exemplos do Amazon Bedrock usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

- Cenários

Conceitos básicos

Olá, Amazon Bedrock

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Lists the available Amazon Bedrock models.
"""
import logging
import json
import boto3


from botocore.exceptions import ClientError


logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


def list_foundation_models(bedrock_client):
    """
    Gets a list of available Amazon Bedrock foundation models.

    :return: The list of available bedrock foundation models.
    """

    try:
        response = bedrock_client.list_foundation_models()
```

```
        models = response["modelSummaries"]
        logger.info("Got %s foundation models.", len(models))
        return models

    except ClientError:
        logger.error("Couldn't list foundation models.")
        raise


def main():
    """Entry point for the example. Uses the AWS SDK for Python (Boto3)
    to create an Amazon Bedrock client. Then lists the available Bedrock models
    in the region set in the callers profile and credentials.
    """

    bedrock_client = boto3.client(service_name="bedrock")

    fm_models = list_foundation_models(bedrock_client)
    for model in fm_models:
        print(f"Model: {model['modelName']}")
        print(json.dumps(model, indent=2))
        print("---------------------------\n")

    logger.info("Done.")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ListFoundationModels Referência da API AWS SDK for Python (Boto3).

Ações

`GetFoundationModel`

O código de exemplo a seguir mostra como usar `GetFoundationModel`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha detalhes de um modelo de base.

```
    def get_foundation_model(self, model_identifier):
        """
        Get details about an Amazon Bedrock foundation model.

        :return: The foundation model's details.
        """

        try:
            return self.bedrock_client.get_foundation_model(
                modelIdentifier=model_identifier
            )["modelDetails"]
        except ClientError:
            logger.error(
                f"Couldn't get foundation models details for {model_identifier}"
            )
            raise
```

- Para obter detalhes da API, consulte a GetFoundationModel Referência da API AWS SDK for Python (Boto3).

`ListFoundationModels`

O código de exemplo a seguir mostra como usar `ListFoundationModels`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar os modelos de base do Amazon Bedrock disponíveis.

```
    def list_foundation_models(self):
        """
        List the available Amazon Bedrock foundation models.

        :return: The list of available bedrock foundation models.
        """

        try:
            response = self.bedrock_client.list_foundation_models()
            models = response["modelSummaries"]
            logger.info("Got %s foundation models.", len(models))
            return models

        except ClientError:
            logger.error("Couldn't list foundation models.")
            raise
```

- Para obter detalhes da API, consulte a ListFoundationModels Referência da API AWS SDK for Python (Boto3).

Cenários

Orquestrar aplicações de IA generativa com o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Funções de etapa

Exemplos do Amazon Bedrock Runtime usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) Amazon Bedrock Runtime.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários
- Amazônia Nova
- Amazon Nova Canvas
- Amazon Nova Reel
- Gerador de Imagens do Amazon Titan
- Amazon Titan Text
- Incorporações de texto Amazon Titan
- Claude da Anthropic
- Command da Cohere
- DeepSeek
- Llama da Meta
- Mistral AI
- Stable Image Core

Cenários

Criar um aplicativo playground para interagir com os modelos de base do Amazon Bedrock

O exemplo de código a seguir mostra como criar playgrounds para interagir com os modelos de base do Amazon Bedrock por meio de diferentes modalidades.

SDK para Python (Boto3)

O Python Foundation Model (FM) Playground é um aplicativo de Python/FastAPI amostra que mostra como usar o Amazon Bedrock com Python. Este exemplo mostra como os

desenvolvedores de Python podem usar o Amazon Bedrock para criar aplicativos AI-enabled generativos. É possível testar e interagir com os modelos de base do Amazon Bedrock usando os três playgrounds a seguir:

- Um playground de texto.
- Um playground de chat.
- Um playground de imagens.

O exemplo também lista e exibe os modelos de base aos quais você tem acesso e respectivas características. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock Runtime

Criar e invocar um prompt gerenciado

O exemplo de código a seguir mostra como:

- Criar um prompt gerenciado.
- Criar uma versão do prompt.
- Invocar o prompt usando a versão.
- Limpar os recursos (opcional).

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar e invocar um prompt gerenciado.

```
import argparse
import boto3
import logging
import time
```

```
# Now import the modules
from prompt import create_prompt, create_prompt_version, delete_prompt
from run_prompt import invoke_prompt

logging.basicConfig(
    level=logging.INFO,
    format='%(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)



def run_scenario(bedrock_client, bedrock_runtime_client, model_id, cleanup=True):
    """
    Runs the Amazon Bedrock managed prompt scenario.

    Args:
        bedrock_client: The Amazon Bedrock Agent client.
        bedrock_runtime_client: The Amazon Bedrock Runtime client.
        model_id (str): The model ID to use for the prompt.
        cleanup (bool): Whether to clean up resources at the end of the scenario.

    Returns:
        dict: A dictionary containing the created resources.
    """
    prompt_id = None

    try:
        # Step 1: Create a prompt
        print("\n=== Step 1: Creating a prompt ===")
        prompt_name = f"PlaylistGenerator-{int(time.time())}"
        prompt_description = "Playlist generator"
        prompt_template = """
          Make me a {{genre}} playlist consisting of the following number of songs:
 {{number}}."""

        create_response = create_prompt(
            bedrock_client,
            prompt_name,
            prompt_description,
            prompt_template,
            model_id
        )
```

```
        prompt_id = create_response['id']
        print(f"Created prompt: {prompt_name} with ID: {prompt_id}")

        # Create a version of the prompt
        print("\n=== Creating a version of the prompt ===")
        version_response = create_prompt_version(
            bedrock_client,
            prompt_id,
            description="Initial version of the product description generator"
        )

        prompt_version_arn = version_response['arn']
        prompt_version = version_response['version']

        print(f"Created prompt version: {prompt_version}")
        print(f"Prompt version ARN: {prompt_version_arn}")

        # Step 2: Invoke the prompt directly
        print("\n=== Step 2: Invoking the prompt ===")
        input_variables = {
            "genre": "pop",
            "number": "2",
           }

        # Use the ARN from the create_prompt_version response
        result = invoke_prompt(
            bedrock_runtime_client,
            prompt_version_arn,
            input_variables
        )
        # Display the playlist
        print(f"\n{result}")


        # Step 3: Clean up resources (optional)
        if cleanup:
            print("\n=== Step 3: Cleaning up resources ===")

            # Delete the prompt
            print(f"Deleting prompt {prompt_id}...")
            delete_prompt(bedrock_client, prompt_id)

            print("Cleanup complete")
```

```
        else:
            print("\n=== Resources were not cleaned up ===")
            print(f"Prompt ID: {prompt_id}")



    except Exception as e:
        logger.exception("Error in scenario: %s", str(e))

        # Attempt to clean up if an error occurred and cleanup was requested
        if cleanup and prompt_id:
            try:
                print("\nCleaning up resources after error...")

                # Delete the prompt
                try:
                    delete_prompt(bedrock_client, prompt_id)
                    print("Cleanup after error complete")
                except Exception as cleanup_error:
                    logger.error("Error during cleanup: %s", str(cleanup_error))
            except Exception as final_error:
                logger.error("Final error during cleanup: %s", str(final_error))

        # Re-raise the original exception
        raise

def main():
    """
    Entry point for the Amazon Bedrock managed prompt scenario.
    """
    parser = argparse.ArgumentParser(
        description="Run the Amazon Bedrock managed prompt scenario."
    )
    parser.add_argument(
        '--region',
        default='us-east-1',
        help="The AWS Region to use."
    )
    parser.add_argument(
        '--model-id',
        default='anthropic.claude-v2',
        help="The model ID to use for the prompt."
    )
    parser.add_argument(
```

```
        '--cleanup',
        action='store_true',
        default=True,
        help="Clean up resources at the end of the scenario."
    )
    parser.add_argument(
        '--no-cleanup',
        action='store_false',
        dest='cleanup',
        help="Don't clean up resources at the end of the scenario."
    )
    args = parser.parse_args()

    bedrock_client = boto3.client('bedrock-agent', region_name=args.region)
    bedrock_runtime_client = boto3.client('bedrock-runtime',
 region_name=args.region)

    print("=== Amazon Bedrock Managed Prompt Scenario ===")
    print(f"Region: {args.region}")
    print(f"Model ID: {args.model_id}")
    print(f"Cleanup resources: {args.cleanup}")

    try:
        run_scenario(
            bedrock_client,
            bedrock_runtime_client,
            args.model_id,
            args.cleanup
        )

    except Exception as e:
        logger.exception("Error running scenario: %s", str(e))

if __name__ == "__main__":
    main()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - Converse
  - CreatePrompt

- CreatePromptVersion
- DeletePrompt

Orquestrar aplicações de IA generativa com o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock

- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Funções de etapa

Uso de ferramenta com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O script de execução principal da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool
```

```
logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""
```

```
# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
 Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
 conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)
```

```
            # Recursively handle the model's response until the model has returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()

    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message to
send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and return
the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """
```

```
        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool use
handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text, and
finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation is
sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
```

```
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the tool
                tool_response = self._invoke_tool(content_block["toolUse"])

                # Add the tool use ID and the tool's response to the list of results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)
```

```
            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based on
 its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
```

```
    """
    Fetches weather data for the given latitude and longitude using the Open-Meteo
API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather": True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Amazônia Nova

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Amazon Nova.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processar o fluxo de respostas em tempo real.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
# Use the Conversation API to send a text message to Amazon Nova Text
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
```

```
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStream Referência da API AWS SDK for Python (Boto3).

Compreensão do documento

O exemplo de código a seguir mostra como enviar e processar um documento com o Amazon Nova no Amazon Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com o Amazon Nova no Amazon Bedrock.

```
# Send and process a document with Amazon Nova on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Amazon Nova Lite.
```

```
model_id = "amazon.nova-lite-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Amazon Nova Canvas

InvokeModel

O exemplo de código a seguir mostra como invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
# Use the native inference API to create an image with Amazon Nova Canvas

import base64
import json
import os
import random

import boto3

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID.
model_id = "amazon.nova-canvas-v1:0"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed between 0 and 858,993,459
seed = random.randint(0, 858993460)

# Format the request payload using the model's native structure.
```

```
native_request = {
    "taskType": "TEXT_IMAGE",
    "textToImageParams": {"text": prompt},
    "imageGenerationConfig": {
        "seed": seed,
        "quality": "standard",
        "height": 512,
        "width": 512,
        "numberOfImages": 1,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"nova_canvas_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"nova_canvas_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Para obter detalhes da API, consulte a InvokeModel Referência da API AWS SDK for Python (Boto3).

Amazon Nova Reel

Text-to-video

O exemplo de código a seguir mostra como usar o Amazon Nova Reel para gerar um vídeo de um prompt de texto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o Amazon Nova Reel para gerar um vídeo de um prompt de texto.

```
"""
This example demonstrates how to use Amazon Nova Reel to generate a video from a
 text prompt.

It shows how to:
- Set up the Amazon Bedrock runtime client
- Configure a text-to-video request
- Submit an asynchronous job for video generation
- Poll for job completion status
- Access the generated video from S3
"""

import random
import time

import boto3

# Replace with your own S3 bucket to store the generated video
# Format: s3://your-bucket-name
OUTPUT_S3_URI = "s3://REPLACE-WITH-YOUR-S3-BUCKET-NAME"


def start_text_to_video_generation_job(bedrock_runtime, prompt, output_s3_uri):
    """
    Starts an asynchronous text-to-video generation job using Amazon Nova Reel.
```

```
    :param bedrock_runtime: The Bedrock runtime client
    :param prompt: The text description of the video to generate
    :param output_s3_uri: S3 URI where the generated video will be stored

    :return: The invocation ARN of the async job
    """
    # Specify the model ID for text-to-video generation
    model_id = "amazon.nova-reel-v1:0"

    # Generate a random seed between 0 and 2,147,483,646
    # This helps ensure unique video generation results
    seed = random.randint(0, 2147483646)

    # Configure the video generation request with additional parameters
    model_input = {
        "taskType": "TEXT_VIDEO",
        "textToVideoParams": {"text": prompt},
        "videoGenerationConfig": {
            "fps": 24,
            "durationSeconds": 6,
            "dimension": "1280x720",
            "seed": seed,
        },
    }

    # Specify the S3 location for the output video
    output_config = {"s3OutputDataConfig": {"s3Uri": output_s3_uri}}

    # Invoke the model asynchronously
    response = bedrock_runtime.start_async_invoke(
        modelId=model_id, modelInput=model_input, outputDataConfig=output_config
    )

    invocation_arn = response["invocationArn"]

    return invocation_arn


def query_job_status(bedrock_runtime, invocation_arn):
    """
    Queries the status of an asynchronous video generation job.

    :param bedrock_runtime: The Bedrock runtime client
    :param invocation_arn: The ARN of the async invocation to check
```

```
    :return: The runtime response containing the job status and details
    """
    return bedrock_runtime.get_async_invoke(invocationArn=invocation_arn)


def main():
    """
    Main function that demonstrates the complete workflow for generating
    a video from a text prompt using Amazon Nova Reel.
    """
    # Create a Bedrock Runtime client
    # Note: Credentials will be loaded from the environment or AWS CLI config
    bedrock_runtime = boto3.client("bedrock-runtime", region_name="us-east-1")

    # Configure the text prompt and output location
    prompt = "Closeup of a cute old steampunk robot. Camera zoom in."

    # Verify the S3 URI has been set to a valid bucket
    if "REPLACE-WITH-YOUR-S3-BUCKET-NAME" in OUTPUT_S3_URI:
        print("ERROR: You must replace the OUTPUT_S3_URI with your own S3 bucket
 URI")
        return

    print("Submitting video generation job...")
    invocation_arn = start_text_to_video_generation_job(
        bedrock_runtime, prompt, OUTPUT_S3_URI
    )
    print(f"Job started with invocation ARN: {invocation_arn}")

    # Poll for job completion
    while True:
        print("\nPolling job status...")
        job = query_job_status(bedrock_runtime, invocation_arn)
        status = job["status"]

        if status == "Completed":
            bucket_uri = job["outputDataConfig"]["s3OutputDataConfig"]["s3Uri"]
            print(f"\nSuccess! The video is available at: {bucket_uri}/output.mp4")
            break
        elif status == "Failed":
            print(
                f"\nVideo generation failed: {job.get('failureMessage', 'Unknown
 error')}"
```

```
            )
            break
        else:
            print("In progress. Waiting 15 seconds...")
            time.sleep(15)


if __name__ == "__main__":
    main()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - GetAsyncInvoke
  - StartAsyncInvoke

Gerador de Imagens do Amazon Titan

InvokeModel

O exemplo de código a seguir mostra como invocar o Amazon Titan Image no Amazon Bedrock para gerar uma imagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Gerador de Imagens do Amazon Titan.

```
# Use the native inference API to create an image with Amazon Titan Image Generator

import base64
import boto3
import json
import os
import random
```

```
# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Image Generator G1.
model_id = "amazon.titan-image-generator-v2:0"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed.
seed = random.randint(0, 2147483647)

# Format the request payload using the model's native structure.
native_request = {
    "taskType": "TEXT_IMAGE",
    "textToImageParams": {"text": prompt},
    "imageGenerationConfig": {
        "numberOfImages": 1,
        "quality": "standard",
        "cfgScale": 8.0,
        "height": 512,
        "width": 512,
        "seed": seed,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"titan_{i}.png")):
```

```
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"titan_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Para obter detalhes da API, consulte a InvokeModel Referência da API AWS SDK for Python (Boto3).

Amazon Titan Text

InvokeModel

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Amazon Titan Text usando a API InvokeModel.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use a API InvokeModel para enviar uma mensagem de texto.

```
# Use the native inference API to send a text message to Amazon Titan Text.

import boto3
import json

from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")
```

```
# Set the model ID, e.g., Titan Text Premier.
model_id = "amazon.titan-text-premier-v1:0"

# Define the prompt for the model.
prompt = "Describe the purpose of a 'hello world' program in one line."

# Format the request payload using the model's native structure.
native_request = {
    "inputText": prompt,
    "textGenerationConfig": {
        "maxTokenCount": 512,
        "temperature": 0.5,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

try:
    # Invoke the model with the request.
    response = client.invoke_model(modelId=model_id, body=request)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract and print the response text.
response_text = model_response["results"][0]["outputText"]
print(response_text)
```

- Para obter detalhes da API, consulte a InvokeModel Referência da API AWS SDK for Python (Boto3).

Incorporações de texto Amazon Titan

InvokeModel

O exemplo de código a seguir mostra como:

- Começar a criar sua primeira incorporação.
- Criar incorporações configurando o número de dimensões e a normalização (somente v2).

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a primeira incorporação com o Incorporador de Texto do Amazon Titan.

```
# Generate and print an embedding with Amazon Titan Text Embeddings V2.

import boto3
import json

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Text Embeddings V2.
model_id = "amazon.titan-embed-text-v2:0"

# The text to convert to an embedding.
input_text = "Please recommend books with a theme similar to the movie 'Inception'."

# Create the request for the model.
native_request = {"inputText": input_text}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the model's native response body.
model_response = json.loads(response["body"].read())

# Extract and print the generated embedding and the input text token count.
embedding = model_response["embedding"]
```

```
input_token_count = model_response["inputTextTokenCount"]

print("\nYour input:")
print(input_text)
print(f"Number of input tokens: {input_token_count}")
print(f"Size of the generated embedding: {len(embedding)}")
print("Embedding:")
print(embedding)
```

- Para obter detalhes da API, consulte a InvokeModel Referência da API AWS SDK for Python (Boto3).

Claude da Anthropic

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Anthropic Claude.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"
```

```
# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
# Use the Conversation API to send a text message to Anthropic Claude
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStream Referência da API AWS SDK for Python (Boto3).

Compreensão do documento

O exemplo de código a seguir mostra como enviar e processar um documento com o Anthropic Claude no Amazon Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Enviar e processar um documento com o Claude da Anthropic no Amazon Bedrock.

```
# Send and process a document with Anthropic Claude on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
```

```
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Cenário: uso de ferramentas com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O script de execução principal da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"
```

```
# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
```

```
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()

    def _send_conversation_to_bedrock(self, conversation):
        """
```

```
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message to
send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and return
the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
```

```
            # If the stop reason is "tool_use", forward everything to the tool use
handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text, and
finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation is
sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the tool
                tool_response = self._invoke_tool(content_block["toolUse"])

                # Add the tool use ID and the tool's response to the list of results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
```

```
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
```

```
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.
```

```
    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based on
 its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-Meteo
 API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather": True}

    try:
```

```
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Command da Cohere

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Cohere Command usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Cohere Command.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Command R.
model_id = "cohere.command-r-v1:0"
```

```
# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Command da Cohere usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
# Use the Conversation API to send a text message to Cohere Command
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Command R.
model_id = "cohere.command-r-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStream Referência da API AWS SDK for Python (Boto3).

Compreensão do documento

O exemplo de código a seguir mostra como enviar e processar um documento com modelos Cohere Command no Amazon Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com modelos Cohere Command no Amazon Bedrock.

```
# Send and process a document with Cohere Command models on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Command R+.
model_id = "cohere.command-r-plus-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
```

```
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Cenário: uso de ferramentas com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O script de execução principal da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"
```

```
# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
```

```
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()

    def _send_conversation_to_bedrock(self, conversation):
        """
```

```
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message to
send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and return
the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
```

```
            # If the stop reason is "tool_use", forward everything to the tool use
handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text, and
finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation is
sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the tool
                tool_response = self._invoke_tool(content_block["toolUse"])

                # Add the tool use ID and the tool's response to the list of results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
```

```
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
```

```
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.
```

```
    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based on
 its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-Meteo
 API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather": True}

    try:
```

```
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

DeepSeek

Compreensão do documento

O exemplo de código a seguir mostra como enviar e processar um documento com o DeepSeek Amazon Bedrock.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com o DeepSeek Amazon Bedrock.

```
# Send and process a document with DeepSeek on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. DeepSeek-R1
model_id = "us.deepseek.r1-v1:0"

# Load the document
```

```
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 2000, "temperature": 0.3},
    )

    # Extract and print the reasoning and response text.
    reasoning, response_text = "", ""
    for item in response["output"]["message"]["content"]:
        for key, value in item.items():
            if key == "reasoningContent":
                reasoning = value["reasoningText"]["text"]
            elif key == "text":
                response_text = value

    print(f"\nReasoning:\n{reasoning}")
    print(f"\nResponse:\n{response_text}")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
```

```
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Llama da Meta

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Meta Llama.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Llama 3 8b Instruct.
model_id = "meta.llama3-8b-instruct-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
```

```
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
# Use the Conversation API to send a text message to Meta Llama
# and print the response stream.
```

```
import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Llama 3 8b Instruct.
model_id = "meta.llama3-8b-instruct-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStream Referência da API AWS SDK for Python (Boto3).

Compreensão do documento

O exemplo de código a seguir mostra como enviar e processar um documento com o Llama no Amazon Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com o Llama no Amazon Bedrock.

```
# Send and process a document with Llama on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Llama 3.1 8B Instruct.
model_id = "us.meta.llama3-1-8b-instruct-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
```

```
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Mistral AI

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto à Mistral usando a API Converse do Bedrock.

```
# Use the Conversation API to send a text message to Mistral.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto à Mistral usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para a Mistral usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
# Use the Conversation API to send a text message to Mistral
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
```

```
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Para obter detalhes da API, consulte a ConverseStream Referência da API AWS SDK for Python (Boto3).

Compreensão do documento

O exemplo de código a seguir mostra como enviar e processar um documento com modelos Mistral no Amazon Bedrock.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie e processe um documento com modelos Mistral no Amazon Bedrock.

```
# Send and process a document with Mistral models on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
```

```
model_id = "mistral.mistral-large-2402-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Python (Boto3).

Stable Image Core

InvokeModel

O exemplo de código a seguir mostra como invocar o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Diffusion.

```
# Use the native inference API to create an image with Stability AI Stable Image
 Core

import base64
import boto3
import json
import os
import random

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-west-2")

# Set the model ID, e.g., Stable Image Core.
model_id = "stability.stable-image-core-v1:1"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed.
seed = random.randint(0, 4294967295)
```

```
# Format the request payload using the model's native structure.
native_request = {
    "prompt": prompt,
    "aspect_ratio": "1:1",
    "seed": seed,
    "output_format": "png",
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"stability_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"stability_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Para obter detalhes da API, consulte a InvokeModel Referência da API AWS SDK for Python (Boto3).

Exemplos de agente dos Amazon Bedrock usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com os agentes Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateAgent`

O código de exemplo a seguir mostra como usar `CreateAgent`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um agente do .

```
    def create_agent(self, agent_name, foundation_model, role_arn, instruction):
        """
        Creates an agent that orchestrates interactions between foundation models,
        data sources, software applications, user conversations, and APIs to carry
```

```
        out tasks to help customers.

        :param agent_name: A name for the agent.
        :param foundation_model: The foundation model to be used for orchestration
by the agent.
        :param role_arn: The ARN of the IAM role with permissions needed by the
agent.
        :param instruction: Instructions that tell the agent what it should do and
how it should
                            interact with users.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """
        try:
            response = self.client.create_agent(
                agentName=agent_name,
                foundationModel=foundation_model,
                agentResourceRoleArn=role_arn,
                instruction=instruction,
            )
        except ClientError as e:
            logger.error(f"Error: Couldn't create agent. Here's why: {e}")
            raise
        else:
            return response["agent"]
```

- Para obter detalhes da API, consulte a CreateAgent Referência da API AWS SDK for Python (Boto3).

`CreateAgentActionGroup`

O código de exemplo a seguir mostra como usar `CreateAgentActionGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um grupo de ação do agente.

```
   def create_agent_action_group(
           self, name, description, agent_id, agent_version, function_arn,
api_schema
   ):
       """
       Creates an action group for an agent. An action group defines a set of
actions that an
       agent should carry out for the customer.

       :param name: The name to give the action group.
       :param description: The description of the action group.
       :param agent_id: The unique identifier of the agent for which to create the
action group.
       :param agent_version: The version of the agent for which to create the
action group.
       :param function_arn: The ARN of the Lambda function containing the business
logic that is
                            carried out upon invoking the action.
       :param api_schema: Contains the OpenAPI schema for the action group.
       :return: Details about the action group that was created.
       """
       try:
           response = self.client.create_agent_action_group(
               actionGroupName=name,
               description=description,
               agentId=agent_id,
               agentVersion=agent_version,
               actionGroupExecutor={"lambda": function_arn},
               apiSchema={"payload": api_schema},
           )
           agent_action_group = response["agentActionGroup"]
       except ClientError as e:
           logger.error(f"Error: Couldn't create agent action group. Here's why:
{e}")
           raise
       else:
           return agent_action_group
```

- Para obter detalhes da API, consulte a CreateAgentActionGroup Referência da API AWS SDK for Python (Boto3).

`CreateAgentAlias`

O código de exemplo a seguir mostra como usar `CreateAgentAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um alias de agente.

```
    def create_agent_alias(self, name, agent_id):
        """
        Creates an alias of an agent that can be used to deploy the agent.

        :param name: The name of the alias.
        :param agent_id: The unique identifier of the agent.
        :return: Details about the alias that was created.
        """
        try:
            response = self.client.create_agent_alias(
                agentAliasName=name, agentId=agent_id
            )
            agent_alias = response["agentAlias"]
        except ClientError as e:
            logger.error(f"Couldn't create agent alias. {e}")
            raise
        else:
            return agent_alias
```

- Para obter detalhes da API, consulte a CreateAgentAlias Referência da API AWS SDK for Python (Boto3).

`CreateFlow`

O código de exemplo a seguir mostra como usar `CreateFlow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um fluxo do Amazon Bedrock.

```
def create_flow(client, flow_name, flow_description, role_arn, flow_def):
    """
    Creates an Amazon Bedrock flow.

    Args:
    client: Amazon Bedrock agent boto3 client.
    flow_name (str): The name for the new flow.
    role_arn (str):  The ARN for the IAM role that use flow uses.
    flow_def (json): The JSON definition of the flow that you want to create.

    Returns:
        dict: The response from CreateFlow.
    """
    try:

        logger.info("Creating flow: %s.", flow_name)

        response = client.create_flow(
            name=flow_name,
            description=flow_description,
            executionRoleArn=role_arn,
            definition=flow_def
        )

        logger.info("Successfully created flow: %s. ID: %s",
                    flow_name,
                    {response['id']})

        return response
```

```
    except ClientError as e:
        logger.exception("Client error creating flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error creating flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a CreateFlow Referência da API AWS SDK for Python (Boto3).

`CreateFlowAlias`

O código de exemplo a seguir mostra como usar `CreateFlowAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um alias para um fluxo do Amazon Bedrock.

```
def create_flow_alias(client, flow_id, flow_version, name, description):
    """
    Creates an alias for an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        str: The ID for the flow alias.
    """

    try:
        logger.info("Creating flow alias for flow: %s.", flow_id)
```

```
        response = client.create_flow_alias(
            flowIdentifier=flow_id,
            name=name,
            description=description,
            routingConfiguration=[
                {
                    "flowVersion": flow_version
                }
            ]
        )
        logger.info("Successfully created flow alias for %s.", flow_id)

        return response['id']

    except ClientError as e:
        logging.exception("Client error creating alias for flow: %s - %s",
                flow_id, str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error creating alias for flow : %s - %s",
                flow_id, str(e))
        raise
```

- Para obter detalhes da API, consulte a CreateFlowAlias Referência da API AWS SDK for Python (Boto3).

`CreateFlowVersion`

O código de exemplo a seguir mostra como usar `CreateFlowVersion`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma versão de um fluxo do Amazon Bedrock.

```
def create_flow_version(client, flow_id, description):
```

```
    """
    Creates a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
        description (str) : A description for the flow.

    Returns:
        str: The version for the flow.
    """
    try:

        logger.info("Creating flow version for flow: %s.", flow_id)

        # Call CreateFlowVersion operation
        response = client.create_flow_version(
            flowIdentifier=flow_id,
            description=description
        )

        logging.info("Successfully created flow version %s for flow %s.",
            response['version'], flow_id)

        return response['version']

    except ClientError as e:
        logging.exception("Client error creating flow: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error creating flow : %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a CreateFlowVersion Referência da API AWS SDK for Python (Boto3).

`CreateKnowledgeBase`

O código de exemplo a seguir mostra como usar `CreateKnowledgeBase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar uma base de conhecimento do Amazon Bedrock.

```
def create_knowledge_base(bedrock_agent_client, name, role_arn, description=None):
    """
    Creates a new knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        name (str): The name of the knowledge base.
        role_arn (str): The ARN of the IAM role that the knowledge base assumes to
 access resources.
        description (str, optional): A description of the knowledge base.

    Returns:
        dict: The details of the created knowledge base.
    """
    try:
        kwargs = {
            "name": name,
            "roleArn": role_arn,
            "knowledgeBaseConfiguration": {
                "type": "VECTOR",
                "vectorKnowledgeBaseConfiguration": {
                    "embeddingModelArn": "arn:aws:bedrock:us-east-1::foundation-
model/amazon.titan-embed-text-v1"
                }
            },
            "storageConfiguration": {
                "type": "OPENSEARCH_SERVERLESS",
                # Note: You will need to create an OpenSearch Serverless collection
 first and replace this ARN
                # with your actual collection ARN from the OpenSearch console. If
 you use the console instead,
                # you can use the quick-create flow to have Knowledge Bases create
 the collection for you.
```

```
                "opensearchServerlessConfiguration": {
                    "collectionArn": "arn:aws:aoss:us-
east-1::123456789012:collection/abcdefgh12345678defgh",
                        "fieldMapping": {
                        "metadataField": "metadata",
                        "textField": "text",
                        "vectorField": "vector"
                        },
                    "vectorIndexName": "test-uuid"
                    },
                },
            "clientToken": "test-client-token-" + str(uuid.uuid4())
        }

        if description:
            kwargs["description"] = description

        response = bedrock_agent_client.create_knowledge_base(**kwargs)

        logger.info("Created knowledge base with ID: %s", response["knowledgeBase"]
["knowledgeBaseId"])
        return response["knowledgeBase"]

    except ClientError as err:
        logger.error(
            "Couldn't create knowledge base. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a CreateKnowledgeBase Referência da API AWS SDK for Python (Boto3).

`CreatePrompt`

O código de exemplo a seguir mostra como usar `CreatePrompt`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um prompt gerenciado do Amazon Bedrock.

```
def create_prompt(client, prompt_name, prompt_description, prompt_template,
 model_id=None):
    """
    Creates an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_name (str): The name for the new prompt.
    prompt_description (str): The description for the new prompt.
    prompt_template (str): The template for the prompt.
    model_id (str, optional): The model ID to associate with the prompt.

    Returns:
        dict: The response from CreatePrompt.
    """
    try:
        logger.info("Creating prompt: %s.", prompt_name)

        # Create a variant with the template
        variant = {
            "name": "default",
            "templateType": "TEXT",
            "templateConfiguration": {
                "text": {
                    "text": prompt_template,
                    "inputVariables": []
                }
            }
        }

        # Extract input variables from the template
        # Look for patterns like {{variable_name}}
```

```
        variables = re.findall(r'{{(.*?)}}', prompt_template)
        for var in variables:
            variant["templateConfiguration"]["text"]
["inputVariables"].append({"name": var.strip()})

        # Add model ID if provided
        if model_id:
            variant["modelId"] = model_id

        # Create the prompt with the variant
        create_params = {
            'name': prompt_name,
            'description': prompt_description,
            'variants': [variant]
        }

        response = client.create_prompt(**create_params)

        logger.info("Successfully created prompt: %s. ID: %s",
                    prompt_name,
                    response['id'])

        return response

    except ClientError as e:
        logger.exception("Client error creating prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error creating prompt: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a CreatePrompt Referência da API AWS SDK for Python (Boto3).

`CreatePromptVersion`

O código de exemplo a seguir mostra como usar `CreatePromptVersion`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma versão de um prompt gerenciado do Amazon Bedrock.

```
def create_prompt_version(client, prompt_id, description=None):
    """
    Creates a version of an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt to create a version for.
    description (str, optional): A description for the version.

    Returns:
        dict: The response from CreatePromptVersion.
    """
    try:
        logger.info("Creating version for prompt ID: %s.", prompt_id)

        create_params = {
            'promptIdentifier': prompt_id
        }

        if description:
            create_params['description'] = description

        response = client.create_prompt_version(**create_params)

        logger.info("Successfully created prompt version: %s", response['version'])
        logger.info("Prompt version ARN: %s", response['arn'])

        return response


    except ClientError as e:
        logger.exception("Client error creating prompt version: %s", str(e))
        raise
```

```
    except Exception as e:
        logger.exception("Unexpected error creating prompt version: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a CreatePromptVersion Referência da API AWS SDK for Python (Boto3).

DeleteAgent

O código de exemplo a seguir mostra como usar `DeleteAgent`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um agente.

```
    def delete_agent(self, agent_id):
        """
        Deletes an Amazon Bedrock agent.

        :param agent_id: The unique identifier of the agent to delete.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """

        try:
            response = self.client.delete_agent(
                agentId=agent_id, skipResourceInUseCheck=False
            )
        except ClientError as e:
            logger.error(f"Couldn't delete agent. {e}")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteAgent Referência da API AWS SDK for Python (Boto3).

`DeleteAgentAlias`

O código de exemplo a seguir mostra como usar `DeleteAgentAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um alias de agente.

```
    def delete_agent_alias(self, agent_id, agent_alias_id):
        """
        Deletes an alias of an Amazon Bedrock agent.

        :param agent_id: The unique identifier of the agent that the alias belongs
to.
        :param agent_alias_id: The unique identifier of the alias to delete.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """

        try:
            response = self.client.delete_agent_alias(
                agentId=agent_id, agentAliasId=agent_alias_id
            )
        except ClientError as e:
            logger.error(f"Couldn't delete agent alias. {e}")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteAgentAlias Referência da API AWS SDK for Python (Boto3).

`DeleteFlow`

O código de exemplo a seguir mostra como usar `DeleteFlow`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um fluxo do Amazon Bedrock.

```
def delete_flow(client, flow_id):
    """
    Deletes an Amazon Bedrock flow.

    Args:
    client: Amazon Bedrock agent boto3 client.
    flow_id (str): The identifier of the flow that you want to delete.

    Returns:
        dict: The response from the DeleteFLow operation.
    """
    try:

        logger.info("Deleting flow ID: %s.",
                    flow_id)

        # Call DeleteFlow operation
        response = client.delete_flow(
            flowIdentifier=flow_id,
            skipResourceInUseCheck=True
        )

        logger.info("Finished deleting flow ID: %s", flow_id)

        return response
```

```
    except ClientError as e:
        logger.exception("Client error deleting flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error deleting flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a DeleteFlow Referência da API AWS SDK for Python (Boto3).

`DeleteFlowAlias`

O código de exemplo a seguir mostra como usar `DeleteFlowAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um alias de um fluxo do Amazon Bedrock.

```
def delete_flow_alias(client, flow_id, flow_alias_id):
    """
    Deletes an Amazon Bedrock flow alias.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from the call to DetectFLowAlias
    """
    try:

        logger.info("Deleting flow alias %s for flow: %s.", flow_alias_id, flow_id)
```

```
        # Delete the flow alias.
        response = client.delete_flow_alias(
            aliasIdentifier=flow_alias_id,
            flowIdentifier=flow_id
        )

        logging.info("Successfully deleted flow version for %s.", flow_id)
        return response

    except ClientError as e:
        logging.exception("Client error deleting flow version: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected deleting flow version: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a DeleteFlowAlias Referência da API AWS SDK for Python (Boto3).

`DeleteFlowVersion`

O código de exemplo a seguir mostra como usar `DeleteFlowVersion`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua uma versão de um fluxo do Amazon Bedrock.

```
def delete_flow_version(client, flow_id, flow_version):
    """
    Deletes a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
```

```
    Returns:
        dict: The response from DeleteFlowVersion.
    """
    try:

        logger.info("Deleting flow version %s for flow: %s.",flow_version, flow_id)

        # Call DeleteFlowVersion operation
        response = client.delete_flow_version(
            flowIdentifier=flow_id,
            flowVersion=flow_version
        )

        logging.info("Successfully deleted flow version %s for %s.",
                flow_version,
                flow_id)
        return response

    except ClientError as e:
        logging.exception("Client error deleting flow version: %s ", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected deleting flow version: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a DeleteFlowVersion Referência da API AWS SDK for Python (Boto3).

`DeleteKnowledgeBase`

O código de exemplo a seguir mostra como usar `DeleteKnowledgeBase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir uma base de conhecimento do Amazon Bedrock.

```
def delete_knowledge_base(bedrock_agent_client, knowledge_base_id):
    """
    Deletes a knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base to delete.

    Returns:
        bool: True if the deletion was successful.
    """
    try:
        bedrock_agent_client.delete_knowledge_base(
            knowledgeBaseId=knowledge_base_id
        )

        logger.info("Deleted knowledge base: %s", knowledge_base_id)
        return True
    except ClientError as err:
        logger.error(
            "Couldn't delete knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a DeleteKnowledgeBase Referência da API AWS SDK for Python (Boto3).

`DeletePrompt`

O código de exemplo a seguir mostra como usar `DeletePrompt`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um prompt gerenciado do Amazon Bedrock.

```
def delete_prompt(client, prompt_id):
    """
    Deletes an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt that you want to delete.

    Returns:
        dict: The response from the DeletePrompt operation.
    """
    try:
        logger.info("Deleting prompt ID: %s.", prompt_id)

        response = client.delete_prompt(
            promptIdentifier=prompt_id
        )

        logger.info("Finished deleting prompt ID: %s", prompt_id)

        return response

    except ClientError as e:
        logger.exception("Client error deleting prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error deleting prompt: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a DeletePrompt Referência da API AWS SDK for Python (Boto3).

GetAgent

O código de exemplo a seguir mostra como usar `GetAgent`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um agente.

```
    def get_agent(self, agent_id, log_error=True):
        """
        Gets information about an agent.

        :param agent_id: The unique identifier of the agent.
        :param log_error: Whether to log any errors that occur when getting the
agent.
                          If True, errors will be logged to the logger. If False,
errors
                          will still be raised, but not logged.
        :return: The information about the requested agent.
        """

        try:
            response = self.client.get_agent(agentId=agent_id)
            agent = response["agent"]
        except ClientError as e:
            if log_error:
                logger.error(f"Couldn't get agent {agent_id}. {e}")
            raise
        else:
            return agent
```

- Para obter detalhes da API, consulte a GetAgent Referência da API AWS SDK for Python (Boto3).

GetFlow

O código de exemplo a seguir mostra como usar `GetFlow`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um fluxo do Amazon Bedrock.

```
def get_flow(client, flow_id):
    """
    Gets an Amazon Bedrock flow.

    Args:
    client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow that you want to get.

    Returns:
        dict: The response from the GetFlow operation.
    """
    try:

        logger.info("Getting flow ID: %s.",
                    flow_id)

        # Call GetFlow operation.
        response = client.get_flow(
            flowIdentifier=flow_id
        )

        logger.info("Retrieved flow ID: %s. Name: %s", flow_id,
                    response['name'])

        return response
```

```
    except ClientError as e:
        logger.exception("Client error getting flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error getting flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a GetFlow Referência da API AWS SDK for Python (Boto3).

`GetFlowVersion`

O código de exemplo a seguir mostra como usar `GetFlowVersion`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha uma versão de um fluxo do Amazon Bedrock.

```
def get_flow_version(client, flow_id, flow_version):
    """
    Gets information about a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
        flow_version (str): The flow version of the flow.

    Returns:
        dict: The response from the call to GetFlowVersion.
    """
    try:
```

```
        logger.info("Deleting flow version for flow: %s.", flow_id)

        # Call GetFlowVersion operation
        response = client.get_flow_version(
            flowIdentifier=flow_id,
            flowVersion=flow_version
        )

        logging.info("Successfully got flow version %s information for flow %s.",
                     flow_version,
                     flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error getting flow version: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error getting flow version: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a GetFlowVersion Referência da API AWS SDK for Python (Boto3).

`GetKnowledgeBase`

O código de exemplo a seguir mostra como usar `GetKnowledgeBase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha uma base de conhecimento do Amazon Bedrock.

```
def get_knowledge_base(bedrock_agent_client, knowledge_base_id):
    """
    Gets details about a specific knowledge base.
```

```
    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base.

    Returns:
        dict: The details of the knowledge base.
    """
    try:
        response = bedrock_agent_client.get_knowledge_base(
            knowledgeBaseId=knowledge_base_id
        )

        logger.info("Retrieved knowledge base: %s", knowledge_base_id)
        return response["knowledgeBase"]
    except ClientError as err:
        logger.error(
            "Couldn't get knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a GetKnowledgeBase Referência da API AWS SDK for Python (Boto3).

`GetPrompt`

O código de exemplo a seguir mostra como usar `GetPrompt`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um prompt gerenciado do Amazon Bedrock.

```
def get_prompt(client, prompt_id):
    """
    Gets an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt that you want to get.

    Returns:
        dict: The response from the GetPrompt operation.
    """
    try:
        logger.info("Getting prompt ID: %s.", prompt_id)

        response = client.get_prompt(
            promptIdentifier=prompt_id
        )

        logger.info("Retrieved prompt ID: %s. Name: %s",
                    prompt_id,
                    response['name'])

        return response

    except ClientError as e:
        logger.exception("Client error getting prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error getting prompt: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a GetPrompt Referência da API AWS SDK for Python (Boto3).

`ListAgentActionGroups`

O código de exemplo a seguir mostra como usar `ListAgentActionGroups`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os grupos de ação de um agente.

```
    def list_agent_action_groups(self, agent_id, agent_version):
        """
        List the action groups for a version of an Amazon Bedrock Agent.

        :param agent_id: The unique identifier of the agent.
        :param agent_version: The version of the agent.
        :return: The list of action group summaries for the version of the agent.
        """

        try:
            action_groups = []

            paginator = self.client.get_paginator("list_agent_action_groups")
            for page in paginator.paginate(
                    agentId=agent_id,
                    agentVersion=agent_version,
                    PaginationConfig={"PageSize": 10},
            ):
                action_groups.extend(page["actionGroupSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list action groups. {e}")
            raise
        else:
            return action_groups
```

- Para obter detalhes da API, consulte a ListAgentActionGroups Referência da API AWS SDK for Python (Boto3).

`ListAgentKnowledgeBases`

O código de exemplo a seguir mostra como usar `ListAgentKnowledgeBases`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as bases de conhecimento associadas a um agente.

```
    def list_agent_knowledge_bases(self, agent_id, agent_version):
        """
        List the knowledge bases associated with a version of an Amazon Bedrock
Agent.

        :param agent_id: The unique identifier of the agent.
        :param agent_version: The version of the agent.
        :return: The list of knowledge base summaries for the version of the agent.
        """

        try:
            knowledge_bases = []

            paginator = self.client.get_paginator("list_agent_knowledge_bases")
            for page in paginator.paginate(
                    agentId=agent_id,
                    agentVersion=agent_version,
                    PaginationConfig={"PageSize": 10},
            ):
                knowledge_bases.extend(page["agentKnowledgeBaseSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list knowledge bases. {e}")
            raise
        else:
            return knowledge_bases
```

- Para obter detalhes da API, consulte a ListAgentKnowledgeBases Referência da API AWS SDK for Python (Boto3).

`ListAgents`

O código de exemplo a seguir mostra como usar `ListAgents`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os agentes que pertencem a uma conta.

```
    def list_agents(self):
        """
        List the available Amazon Bedrock Agents.

        :return: The list of available bedrock agents.
        """

        try:
            all_agents = []

            paginator = self.client.get_paginator("list_agents")
            for page in paginator.paginate(PaginationConfig={"PageSize": 10}):
                all_agents.extend(page["agentSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list agents. {e}")
            raise
        else:
            return all_agents
```

- Para obter detalhes da API, consulte a ListAgents Referência da API AWS SDK for Python (Boto3).

`ListFlowAliases`

O código de exemplo a seguir mostra como usar `ListFlowAliases`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os aliases de um fluxo do Amazon Bedrock.

```
def list_flow_aliases(client, flow_id):
    """
    Lists the aliases of an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from ListFlowAliases.
    """
    try:

        finished = False

        logger.info("Listing flow aliases for flow: %s.", flow_id)

        print(f"Aliases for flow: {flow_id}")

        response = client.list_flow_aliases(
            flowIdentifier=flow_id,
            maxResults=10)

        while finished is False:

            for alias in response['flowAliasSummaries']:
                print(f"Alias Name: {alias['name']}")
                print(f"ID: {alias['id']}")
```

```
                print(f"Description: {alias.get('description', 'No
 description')}\n")

                if 'nextToken' in response:
                    next_token = response['nextToken']
                    response = client.list_flow_aliases(maxResults=10,
                                                        nextToken=next_token)
                else:
                    finished = True

        logging.info("Successfully listed flow aliases for flow %s.",
                flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error listing flow aliases: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow aliases: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a ListFlowAliases Referência da API AWS SDK for Python (Boto3).

`ListFlowVersions`

O código de exemplo a seguir mostra como usar `ListFlowVersions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as versões de um fluxo do Amazon Bedrock.

```
def list_flow_versions(client, flow_id):
    """
```

```
    Lists the versions of an Amazon Bedrock flow.

    Args:
        client: Amazon bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from ListFlowVersions.
    """
    try:

        finished = False

        logger.info("Listing flow versions for flow: %s.", flow_id)

        response = client.list_flow_versions(
            flowIdentifier=flow_id,
            maxResults=10)

        while finished is False:

            print(f"Versions for flow:{flow_id}")
            for version in response['flowVersionSummaries']:
                print(f"Version: {version['version']}")
                print(f"Status: {version['status']}\n")

                if 'nextToken' in response:
                    next_token = response['nextToken']
                    response = client.list_flow_versions(maxResults=10,
                                                         nextToken=next_token)
                else:
                    finished = True


        logging.info("Successfully listed flow versions for flow %s.",
                flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error listing flow versions: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow versions: %s", str(e))
```

```
        raise
```

- Para obter detalhes da API, consulte a ListFlowVersions Referência da API AWS SDK for Python (Boto3).

ListFlows

O código de exemplo a seguir mostra como usar `ListFlows`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os fluxos do Amazon Bedrock.

```
def list_flows(client):
    """
    Lists versions of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        Nothing.
    """
    try:
        finished = False

        logger.info("Listing flows:")

        response = client.list_flows(maxResults=10)

        while finished is False:

            for flow in response['flowSummaries']:
                print(f"ID: {flow['id']}")
```

```
                print(f"Name: {flow['name']}")
                print(
                    f"Description: {flow.get('description', 'No description')}")
                print(f"Latest version: {flow['version']}")
                print(f"Status: {flow['status']}\n")

            if 'nextToken' in response:
                next_token = response['nextToken']
                response = client.list_flows(maxResults=10,
                                             nextToken=next_token)
            else:
                finished = True

        logging.info("Successfully listed flows.")


    except ClientError as e:
        logging.exception("Client error listing flow versions: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow versions: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a ListFlows Referência da API AWS SDK for Python (Boto3).

`ListKnowledgeBases`

O código de exemplo a seguir mostra como usar `ListKnowledgeBases`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste as bases de conhecimento do Amazon Bedrock.

```
def list_knowledge_bases(bedrock_agent_client, max_results=None):
```

```
    """
    Lists the knowledge bases in your AWS account.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        max_results (int, optional): The maximum number of knowledge bases to
return.

    Returns:
        list: A list of knowledge base details.
    """
    try:
        kwargs = {}
        if max_results is not None:
            kwargs["maxResults"] = max_results

        # Initialize an empty list to store all knowledge bases
        all_knowledge_bases = []

        # Use paginator to handle pagination automatically
        paginator = bedrock_agent_client.get_paginator('list_knowledge_bases')
        page_iterator = paginator.paginate(**kwargs)

        # Iterate through each page of results
        for page in page_iterator:
            all_knowledge_bases.extend(page.get('knowledgeBaseSummaries', []))

        logger.info("Found %s knowledge bases.", len(all_knowledge_bases))
        return all_knowledge_bases
    except ClientError as err:
        logger.error(
            "Couldn't list knowledge bases. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a ListKnowledgeBases Referência da API AWS SDK for Python (Boto3).

`ListPrompts`

O código de exemplo a seguir mostra como usar `ListPrompts`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste os prompts gerenciados do Amazon Bedrock.

```
def list_prompts(client, max_results=10):
    """
    Lists Amazon Bedrock managed prompts.

    Args:
        client: Amazon Bedrock Agent boto3 client.
        max_results (int): Maximum number of results to return per page.

    Returns:
        list: A list of prompt summaries.
    """
    try:
        logger.info("Listing prompts:")

        # Create a paginator for the list_prompts operation
        paginator = client.get_paginator('list_prompts')

        # Create the pagination parameters
        pagination_config = {
            'maxResults': max_results
        }

        # Initialize an empty list to store all prompts
        all_prompts = []

        # Iterate through all pages
        for page in paginator.paginate(**pagination_config):
            all_prompts.extend(page.get('promptSummaries', []))
```

```
        logger.info("Successfully listed %s prompts.", len(all_prompts))
        return all_prompts

    except ClientError as e:
        logger.exception("Client error listing prompts: %s", str(e))
        raise
    except Exception as e:
        logger.exception("Unexpected error listing prompts: %s", str(e))
        raise
```

- Para obter detalhes da API, consulte a ListPrompts Referência da API AWS SDK for Python (Boto3).

`PrepareAgent`

O código de exemplo a seguir mostra como usar `PrepareAgent`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Prepare um agente para testes internos.

```
    def prepare_agent(self, agent_id):
        """
        Creates a DRAFT version of the agent that can be used for internal testing.

        :param agent_id: The unique identifier of the agent to prepare.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """
        try:
            prepared_agent_details = self.client.prepare_agent(agentId=agent_id)
        except ClientError as e:
```

```
            logger.error(f"Couldn't prepare agent. {e}")
            raise
        else:
            return prepared_agent_details
```

- Para obter detalhes da API, consulte a PrepareAgent Referência da API AWS SDK for Python (Boto3).

`PrepareFlow`

O código de exemplo a seguir mostra como usar `PrepareFlow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Prepare um fluxo do Amazon Bedrock.

```
def prepare_flow(client, flow_id):
    """
    Prepares an Amazon Bedrock Flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow that you want to prepare.

    Returns:
        str: The status of the flow preparation
    """
    try:

        # Prepare the flow.
        logger.info("Preparing flow ID: %s",
                    flow_id)
```

```
        response = client.prepare_flow(
            flowIdentifier=flow_id
        )

        status = response.get('status')

        while status == "Preparing":
            logger.info("Preparing flow ID: %s. Status %s",
                        flow_id, status)

            sleep(5)
            response = client.get_flow(
                flowIdentifier=flow_id
            )
            status = response.get('status')
            print(f"Flow Status: {status}")

        if status == "Prepared":
            logger.info("Finished preparing flow ID: %s. Status %s",
                        flow_id, status)
        else:
            logger.warning("flow ID: %s not prepared. Status %s",
                           flow_id, status)

        return status

    except ClientError as e:
        logger.exception("Client error preparing flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error preparing flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a PrepareFlow Referência da API AWS SDK for Python (Boto3).

`UpdateFlow`

O código de exemplo a seguir mostra como usar `UpdateFlow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize um fluxo do Amazon Bedrock.

```
def update_flow(client, flow_id, flow_name, flow_description, role_arn, flow_def):
    """
    Updates an Amazon Bedrock flow.

    Args:
    client: bedrock agent boto3 client.
    flow_id (str): The ID for the flow that you want to update.
    flow_name (str): The name for the flow.
    role_arn (str):  The ARN for the IAM role that use flow uses.
    flow_def (json): The JSON definition of the flow that you want to create.

    Returns:
        dict: Flow information if successful.
    """
    try:

        logger.info("Updating flow: %s.", flow_id)

        response = client.update_flow(
            flowIdentifier=flow_id,
            name=flow_name,
            description=flow_description,
            executionRoleArn=role_arn,
            definition=flow_def
        )

        logger.info("Successfully updated flow: %s. ID: %s",
                    flow_name,
                    {response['id']})

        return response

    except ClientError as e:
```

```
        logger.exception("Client error updating flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error updating flow: %s", {str(e)})
        raise
```

- Para obter detalhes da API, consulte a UpdateFlow Referência da API AWS SDK for Python (Boto3).

`UpdateFlowAlias`

O código de exemplo a seguir mostra como usar `UpdateFlowAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize um alias para um fluxo do Amazon Bedrock.

```
def update_flow_alias(client, flow_id, alias_id, flow_version, name, description):
    """
    Updates an alias for an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        str: The response from UpdateFlowAlias.
    """

    try:
        logger.info("Updating flow alias %s for flow: %s.", alias_id, flow_id)

        response = client.update_flow_alias(
            aliasIdentifier=alias_id,
```

```
            flowIdentifier=flow_id,
            name=name,
            description=description,
            routingConfiguration=[
                {
                    "flowVersion": flow_version
                }
            ]
        )
        logger.info("Successfully updated flow alias %s for %s.", alias_id, flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error updating alias %s for flow: %s - %s",
                alias_id, flow_id, str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error updating alias %s for flow : %s - %s",
                alias_id, flow_id, str(e))
        raise
```

- Para obter detalhes da API, consulte a UpdateFlowAlias Referência da API AWS SDK for Python (Boto3).

`UpdateKnowledgeBase`

O código de exemplo a seguir mostra como usar `UpdateKnowledgeBase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize uma base de conhecimento do Amazon Bedrock.

```
def update_knowledge_base(bedrock_agent_client, knowledge_base_id, name=None,
 description=None, role_arn=None):
```

```
    """
    Updates an existing knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base to update.
        name (str, optional): The new name for the knowledge base.
        description (str, optional): The new description for the knowledge base.
        role_arn (str, optional): The new IAM role ARN for the knowledge base.

    Returns:
        dict: The details of the updated knowledge base.
    """
    try:
        kwargs = {
            "knowledgeBaseId": knowledge_base_id,
            "knowledgeBaseConfiguration": {
                "type": "VECTOR",
                "vectorKnowledgeBaseConfiguration": {
                    "embeddingModelArn": "arn:aws:bedrock:us-east-1::foundation-
model/amazon.titan-embed-text-v1"
                }
            }
        }

        if name:
            kwargs["name"] = name
        if description:
            kwargs["description"] = description
        if role_arn:
            kwargs["roleArn"] = role_arn

        response = bedrock_agent_client.update_knowledge_base(**kwargs)

        logger.info("Updated knowledge base: %s", knowledge_base_id)
        return response["knowledgeBase"]

    except ClientError as err:
        logger.error(
            "Couldn't update knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
```

```
        raise
```

- Para obter detalhes da API, consulte a UpdateKnowledgeBase Referência da API AWS SDK for Python (Boto3).

Cenários

Criar e invocar um fluxo

O exemplo de código a seguir mostra como:

- Criar um perfil de execução para o fluxo.
- Criar o fluxo.
- Implantar o fluxo totalmente configurado.
- Invocar o fluxo com prompts fornecidos pelo usuário.
- Exclua todos os recursos criados.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gera uma playlist de música com base no gênero e no número de músicas especificados pelo usuário.

```
from datetime import datetime
import logging
import boto3

from botocore.exceptions import ClientError

from roles import create_flow_role, delete_flow_role, update_role_policy
from flow import create_flow, prepare_flow, delete_flow
from run_flow import run_playlist_flow
from flow_version import create_flow_version, delete_flow_version
```

```
from flow_alias import create_flow_alias, delete_flow_alias

logging.basicConfig(
    level=logging.INFO
)
logger = logging.getLogger(__name__)

def create_input_node(name):
    """
    Creates an input node configuration for an Amazon Bedrock flow.

    The input node serves as the entry point for the flow and defines
    the initial document structure that will be passed to subsequent nodes.

    Args:
        name (str): The name of the input node.

    Returns:
        dict: The input node configuration.

    """
    return {
        "type": "Input",
        "name": name,
        "outputs": [
            {
                "name": "document",
                "type": "Object"
            }
        ]
    }


def create_prompt_node(name, model_id):
    """
    Creates a prompt node configuration for a Bedrock flow that generates music
 playlists.

    The prompt node defines an inline prompt template that creates a music playlist
 based on
    a specified genre and number of songs. The prompt uses two variables that are
 mapped from
    the input JSON object:
    - {{genre}}: The genre of music to create a playlist for
```

```
    - {{number}}: The number of songs to include in the playlist

    Args:
        name (str): The name of the prompt node.
        model_id (str): The identifier of the foundation model to use for the
prompt.

    Returns:
        dict: The prompt node.

    """

    return {
        "type": "Prompt",
        "name": name,
        "configuration": {
            "prompt": {
                "sourceConfiguration": {
                    "inline": {
                        "modelId": model_id,
                        "templateType": "TEXT",
                        "inferenceConfiguration": {
                            "text": {
                                "temperature": 0.8
                            }
                        },
                        "templateConfiguration": {
                            "text": {
                                "text": "Make me a {{genre}} playlist consisting of
the following number of songs: {{number}}."
                            }
                        }
                    }
                }
            }
        },
        "inputs": [
            {
                "name": "genre",
                "type": "String",
                "expression": "$.data.genre"
            },
            {
                "name": "number",
```

```
                "type": "Number",
                "expression": "$.data.number"
            }
        ],
        "outputs": [
            {
                "name": "modelCompletion",
                "type": "String"
            }
        ]
    }


def create_output_node(name):
    """
    Creates an output node configuration for a Bedrock flow.

    The output node validates that the output from the last node is a string
    and returns it unmodified. The input name must be "document".

    Args:
        name (str): The name of the output node.

    Returns:
        dict: The output node configuration containing the output node:

    """

    return {
        "type": "Output",
        "name": name,
        "inputs": [
            {
                "name": "document",
                "type": "String",
                "expression": "$.data"
            }
        ]
    }
```

```
def create_playlist_flow(client, flow_name, flow_description, role_arn,
 prompt_model_id):
    """
    Creates the playlist generator flow.
    Args:
        client: bedrock agent boto3 client.
        role_arn (str): Name for the new IAM role.
        prompt_model_id (str): The id of the model to use in the prompt node.
    Returns:
        dict: The response from the create_flow operation.
    """

    input_node = create_input_node("FlowInput")
    prompt_node = create_prompt_node("MakePlaylist", prompt_model_id)
    output_node = create_output_node("FlowOutput")

    # Create connections between the nodes
    connections = []

    #  First, create connections between the output of the flow
    # input node and each input of the prompt node.
    for prompt_node_input in prompt_node["inputs"]:
        connections.append(
            {
                "name": "_".join([input_node["name"], prompt_node["name"],
                                  prompt_node_input["name"]]),
                "source": input_node["name"],
                "target": prompt_node["name"],
                "type": "Data",
                "configuration": {
                    "data": {
                        "sourceOutput": input_node["outputs"][0]["name"],
                        "targetInput": prompt_node_input["name"]
                    }
                }
            }
        )

    # Then, create a connection between the output of the prompt node and the input
 of the flow output node
    connections.append(
        {
            "name": "_".join([prompt_node["name"], output_node["name"]]),
            "source": prompt_node["name"],
```

```
            "target": output_node["name"],
            "type": "Data",
            "configuration": {
                "data": {
                    "sourceOutput": prompt_node["outputs"][0]["name"],
                    "targetInput": output_node["inputs"][0]["name"]
                }
            }
        }
    )

    flow_def = {
        "nodes": [input_node, prompt_node, output_node],
        "connections": connections
    }

    # Create the flow.

    response = create_flow(
        client, flow_name, flow_description, role_arn, flow_def)

    return response



def get_model_arn(client, model_id):
    """
    Gets the Amazon Resource Name (ARN) for a model.
    Args:
        client (str): Amazon Bedrock boto3 client.
        model_id (str): The id of the model.
    Returns:
        str: The ARN of the model.
    """

    try:
        # Call GetFoundationModelDetails operation
        response = client.get_foundation_model(modelIdentifier=model_id)

        # Extract model ARN from the response
        model_arn = response['modelDetails']['modelArn']

        return model_arn
```

```
    except ClientError as e:
        logger.exception("Client error getting model ARN: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexpected error getting model ARN: %s", {str(e)})
        raise


def prepare_flow_version_and_alias(bedrock_agent_client,
                                   flow_id):
    """
    Prepares the flow and then creates a flow version and flow alias.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        flowd_id (str): The ID of the flow that you want to prepare.
    Returns: The flow_version and flow_alias.

    """

    status = prepare_flow(bedrock_agent_client, flow_id)

    flow_version = None
    flow_alias = None

    if status == 'Prepared':

        # Create the flow version and alias.
        flow_version = create_flow_version(bedrock_agent_client,
                                           flow_id,
                                           f"flow version for flow {flow_id}.")

        flow_alias = create_flow_alias(bedrock_agent_client,
                                       flow_id,
                                       flow_version,
                                       "latest",
                                       f"Alias for flow {flow_id}, version
 {flow_version}")

    return flow_version, flow_alias



def delete_role_resources(bedrock_agent_client,
```

```
                          iam_client,
                          role_name,
                          flow_id,
                          flow_version,
                          flow_alias):
    """
    Deletes the flow, flow alias, flow version, and IAM roles.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        iam_client: Amazon IAM boto3 client.
        role_name (str): The name of the IAM role.
        flow_id (str): The id of the flow.
        flow_version (str): The version of the flow.
        flow_alias (str): The alias of the flow.
    """

    if flow_id is not None:
        if flow_alias is not None:
            delete_flow_alias(bedrock_agent_client, flow_id, flow_alias)
        if flow_version is not None:
            delete_flow_version(bedrock_agent_client,
                        flow_id, flow_version)
        delete_flow(bedrock_agent_client, flow_id)

    if role_name is not None:
        delete_flow_role(iam_client, role_name)



def main():
    """
    Creates, runs, and optionally deletes a Bedrock flow for generating music
 playlists.

    Note:
        Requires valid AWS credentials in the default profile
    """

    delete_choice = "y"
    try:

        # Get various boto3 clients.
        session = boto3.Session(profile_name='default')
        bedrock_agent_runtime_client = session.client('bedrock-agent-runtime')
```

```
        bedrock_agent_client = session.client('bedrock-agent')
        bedrock_client = session.client('bedrock')
        iam_client = session.client('iam')

        role_name = None
        flow_id = None
        flow_version = None
        flow_alias = None

        #Change the model as needed.
        prompt_model_id = "amazon.nova-pro-v1:0"

        # Base the flow name on the current date and time
        current_time = datetime.now()
        timestamp = current_time.strftime("%Y-%m-%d-%H-%M-%S")
        flow_name = f"FlowPlayList_{timestamp}"
        flow_description = "A flow to generate a music playlist."

        # Create a role for the flow.
        role_name = f"BedrockFlowRole-{flow_name}"
        role = create_flow_role(iam_client, role_name)
        role_arn = role['Arn']

        # Create the flow.
        response = create_playlist_flow(
            bedrock_agent_client, flow_name, flow_description, role_arn,
prompt_model_id)
        flow_id = response.get('id')

        if flow_id:
            # Update accessible resources in the role.
            model_arn = get_model_arn(bedrock_client, prompt_model_id)
            update_role_policy(iam_client, role_name, [
                               response.get('arn'), model_arn])

            # Prepare the flow and flow version.
            flow_version, flow_alias = prepare_flow_version_and_alias(
                bedrock_agent_client, flow_id)

            # Run the flow.
            if flow_version and flow_alias:
                run_playlist_flow(bedrock_agent_runtime_client,
                                  flow_id, flow_alias)
```

```
                delete_choice = input("Delete flow? y or n : ").lower()


            else:
                print("Couldn't run. Deleting flow and role.")
                delete_flow(bedrock_agent_client, flow_id)
                delete_flow_role(iam_client, role_name)
        else:
            print("Couldn't create flow.")


    except Exception as e:
        print(f"Fatal error: {str(e)}")

    finally:
        if delete_choice == 'y':
                delete_role_resources(bedrock_agent_client,
                                      iam_client,
                                      role_name,
                                      flow_id,
                                      flow_version,
                                      flow_alias)
        else:
            print("Flow not deleted. ")
            print(f"\tFlow ID: {flow_id}")
            print(f"\tFlow version: {flow_version}")
            print(f"\tFlow alias: {flow_alias}")
            print(f"\tRole ARN: {role_arn}")

        print("Done!")

if __name__ == "__main__":
    main()


def invoke_flow(client, flow_id, flow_alias_id, input_data):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.
```

```
    Returns:
        Dict containing flow status and flow output.
    """

    response = None
    request_params = None

    request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }


    response = client.invoke_flow(**request_params)

    flow_status = ""
    output= ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Save the model output.
        elif 'flowOutputEvent' in event:
            output = event['flowOutputEvent']['content']['document']
            logger.info("Output : %s", output)

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "output": output

    }
```

```
def run_playlist_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Runs the playlist generator flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """


    print ("Welcome to the playlist generator flow.")
    # Get the initial prompt from the user.
    genre = input("Enter genre: ")
    number_of_songs = int(input("Enter number of songs: "))


    # Use prompt to create input data for the input node.
    flow_input_data = {
        "content": {
            "document": {
                "genre" : genre,
                "number" : number_of_songs
            }
        },
        "nodeName": "FlowInput",
        "nodeOutputName": "document"
    }

    try:

        result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data)

        status = result['flow_status']

        if status == "SUCCESS":
                # The flow completed successfully.
                logger.info("The flow %s successfully completed.", flow_id)
                print(result['output'])
        else:
```

```
            logger.warning("Flow status: %s",status)

    except ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})
        raise

    except Exception as e:
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})
        raise



def create_flow_role(client, role_name):
    """
    Creates an IAM role for Amazon Bedrock with permissions to run a flow.

    Args:
        role_name (str): Name for the new IAM role.
    Returns:
        str: The role Amazon Resource Name.
    """


    # Trust relationship policy - allows Amazon Bedrock service to assume this role.
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "bedrock.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }]
    }

    # Basic inline policy for for running a flow.

    resources = "*"

    bedrock_policy = {
        "Version":"2012-10-17",
        "Statement": [
```

```
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                # Using * as placeholder - Later you update with specific ARNs.
                "Resource": resources
            }
        ]
    }



    try:
        # Create the IAM role with trust policy
        logging.info("Creating role: %s",role_name)
        role = client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy),
            Description="Role for Amazon Bedrock operations"
        )

        # Attach inline policy to the role
        print("Attaching inline policy")
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(bedrock_policy)
        )

        logging.info("Create Role ARN: %s", role['Role']['Arn'])
        return role['Role']

    except ClientError as e:
        logging.warning("Error creating role: %s", str(e))
        raise
    except Exception as e:
        logging.warning("Unexpected error: %s", str(e))
        raise


def update_role_policy(client, role_name, resource_arns):
```

```
    """
    Updates an IAM role's inline policy with specific resource ARNs.

    Args:
        role_name (str): Name of the existing role.
        resource_arns (list): List of resource ARNs to allow access to.
    """


    updated_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:GetFlow",
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                "Resource": resource_arns
            }
        ]
    }

    try:
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(updated_policy)
        )
        logging.info("Updated policy for role: %s",role_name)

    except ClientError as e:
        logging.warning("Error updating role policy: %s", str(e))
        raise


def delete_flow_role(client, role_name):
    """
    Deletes an IAM role.

    Args:
        role_name (str): Name of the role to delete.
```

```
    """



    try:
        # Detach and delete inline policies
        policies = client.list_role_policies(RoleName=role_name)['PolicyNames']
        for policy_name in policies:
            client.delete_role_policy(RoleName=role_name, PolicyName=policy_name)

        # Delete the role
        client.delete_role(RoleName=role_name)
        logging.info("Deleted role: %s", role_name)


    except ClientError as e:
        logging.info("Error Deleting role: %s", str(e))
        raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateFlow
  - CreateFlowAlias
  - CreateFlowVersion
  - DeleteFlow
  - DeleteFlowAlias
  - DeleteFlowVersion
  - GetFlow
  - GetFlowAlias
  - GetFlowVersion
  - InvokeFlow
  - PrepareFlow

Criar e invocar um prompt gerenciado

O exemplo de código a seguir mostra como:

- Criar um prompt gerenciado.
- Criar uma versão do prompt.
- Invocar o prompt usando a versão.
- Limpar os recursos (opcional).

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar e invocar um prompt gerenciado.

```
import argparse
import boto3
import logging
import time

# Now import the modules
from prompt import create_prompt, create_prompt_version, delete_prompt
from run_prompt import invoke_prompt

logging.basicConfig(
    level=logging.INFO,
    format='%(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)



def run_scenario(bedrock_client, bedrock_runtime_client, model_id, cleanup=True):
    """
    Runs the Amazon Bedrock managed prompt scenario.

    Args:
        bedrock_client: The Amazon Bedrock Agent client.
        bedrock_runtime_client: The Amazon Bedrock Runtime client.
        model_id (str): The model ID to use for the prompt.
```

```
        cleanup (bool): Whether to clean up resources at the end of the scenario.

    Returns:
        dict: A dictionary containing the created resources.
    """
    prompt_id = None

    try:
        # Step 1: Create a prompt
        print("\n=== Step 1: Creating a prompt ===")
        prompt_name = f"PlaylistGenerator-{int(time.time())}"
        prompt_description = "Playlist generator"
        prompt_template = """
          Make me a {{genre}} playlist consisting of the following number of songs:
{{number}}."""

        create_response = create_prompt(
            bedrock_client,
            prompt_name,
            prompt_description,
            prompt_template,
            model_id
        )

        prompt_id = create_response['id']
        print(f"Created prompt: {prompt_name} with ID: {prompt_id}")

        # Create a version of the prompt
        print("\n=== Creating a version of the prompt ===")
        version_response = create_prompt_version(
            bedrock_client,
            prompt_id,
            description="Initial version of the product description generator"
        )

        prompt_version_arn = version_response['arn']
        prompt_version = version_response['version']

        print(f"Created prompt version: {prompt_version}")
        print(f"Prompt version ARN: {prompt_version_arn}")

        # Step 2: Invoke the prompt directly
        print("\n=== Step 2: Invoking the prompt ===")
        input_variables = {
```

```
            "genre": "pop",
            "number": "2",
           }

        # Use the ARN from the create_prompt_version response
        result = invoke_prompt(
            bedrock_runtime_client,
            prompt_version_arn,
            input_variables
        )
        # Display the playlist
        print(f"\n{result}")


        # Step 3: Clean up resources (optional)
        if cleanup:
            print("\n=== Step 3: Cleaning up resources ===")

            # Delete the prompt
            print(f"Deleting prompt {prompt_id}...")
            delete_prompt(bedrock_client, prompt_id)

            print("Cleanup complete")
        else:
            print("\n=== Resources were not cleaned up ===")
            print(f"Prompt ID: {prompt_id}")



    except Exception as e:
        logger.exception("Error in scenario: %s", str(e))

        # Attempt to clean up if an error occurred and cleanup was requested
        if cleanup and prompt_id:
            try:
                print("\nCleaning up resources after error...")

                # Delete the prompt
                try:
                    delete_prompt(bedrock_client, prompt_id)
                    print("Cleanup after error complete")
                except Exception as cleanup_error:
                    logger.error("Error during cleanup: %s", str(cleanup_error))
            except Exception as final_error:
```

```
                logger.error("Final error during cleanup: %s", str(final_error))

        # Re-raise the original exception
        raise

def main():
    """
    Entry point for the Amazon Bedrock managed prompt scenario.
    """
    parser = argparse.ArgumentParser(
        description="Run the Amazon Bedrock managed prompt scenario."
    )
    parser.add_argument(
        '--region',
        default='us-east-1',
        help="The AWS Region to use."
    )
    parser.add_argument(
        '--model-id',
        default='anthropic.claude-v2',
        help="The model ID to use for the prompt."
    )
    parser.add_argument(
        '--cleanup',
        action='store_true',
        default=True,
        help="Clean up resources at the end of the scenario."
    )
    parser.add_argument(
        '--no-cleanup',
        action='store_false',
        dest='cleanup',
        help="Don't clean up resources at the end of the scenario."
    )
    args = parser.parse_args()

    bedrock_client = boto3.client('bedrock-agent', region_name=args.region)
    bedrock_runtime_client = boto3.client('bedrock-runtime',
 region_name=args.region)

    print("=== Amazon Bedrock Managed Prompt Scenario ===")
    print(f"Region: {args.region}")
    print(f"Model ID: {args.model_id}")
    print(f"Cleanup resources: {args.cleanup}")
```

```
    try:
        run_scenario(
            bedrock_client,
            bedrock_runtime_client,
            args.model_id,
            args.cleanup
        )

    except Exception as e:
        logger.exception("Error running scenario: %s", str(e))

if __name__ == "__main__":
    main()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - Converse
  - CreatePrompt
  - CreatePromptVersion
  - DeletePrompt

Criar e invocar um agente

O exemplo de código a seguir mostra como:

- Criar um perfil de execução para o agente.
- Criar o agente e implantar uma versão de RASCUNHO.
- Criar uma função do Lambda que implante os recursos do agente.
- Criar um grupo de ação que conecte o agente à função do Lambda.
- Implantar o agente totalmente configurado.
- Invocar o agente com prompts fornecidos pelo usuário.
- Exclua todos os recursos criados.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie e invoque um agente.

```
REGION = "us-east-1"
ROLE_POLICY_NAME = "agent_permissions"


class BedrockAgentScenarioWrapper:
    """Runs a scenario that shows how to get started using Amazon Bedrock Agents."""

    def __init__(
            self, bedrock_agent_client, runtime_client, lambda_client, iam_resource,
 postfix
    ):
        self.iam_resource = iam_resource
        self.lambda_client = lambda_client
        self.bedrock_agent_runtime_client = runtime_client
        self.postfix = postfix

        self.bedrock_wrapper = BedrockAgentWrapper(bedrock_agent_client)

        self.agent = None
        self.agent_alias = None
        self.agent_role = None
        self.prepared_agent_details = None
        self.lambda_role = None
        self.lambda_function = None

    def run_scenario(self):
        print("=" * 88)
        print("Welcome to the Amazon Bedrock Agents demo.")
        print("=" * 88)

        # Query input from user
        print("Let's start with creating an agent:")
        print("-" * 40)
```

```
        name, foundation_model = self._request_name_and_model_from_user()
        print("-" * 40)

        # Create an execution role for the agent
        self.agent_role = self._create_agent_role(foundation_model)

        # Create the agent
        self.agent = self._create_agent(name, foundation_model)

        # Prepare a DRAFT version of the agent
        self.prepared_agent_details = self._prepare_agent()

        # Create the agent's Lambda function
        self.lambda_function = self._create_lambda_function()

        # Configure permissions for the agent to invoke the Lambda function
        self._allow_agent_to_invoke_function()
        self._let_function_accept_invocations_from_agent()

        # Create an action group to connect the agent with the Lambda function
        self._create_agent_action_group()

        # If the agent has been modified or any components have been added, prepare
the agent again
        components = [self._get_agent()]
        components += self._get_agent_action_groups()
        components += self._get_agent_knowledge_bases()

        latest_update = max(component["updatedAt"] for component in components)
        if latest_update > self.prepared_agent_details["preparedAt"]:
            self.prepared_agent_details = self._prepare_agent()

        # Create an agent alias
        self.agent_alias = self._create_agent_alias()

        # Test the agent
        self._chat_with_agent(self.agent_alias)

        print("=" * 88)
        print("Thanks for running the demo!\n")

        if q.ask("Do you want to delete the created resources? [y/N] ", q.is_yesno):
            self._delete_resources()
            print("=" * 88)
```

```
            print(
                "All demo resources have been deleted. Thanks again for running the
demo!"
            )
        else:
            self._list_resources()
            print("=" * 88)
            print("Thanks again for running the demo!")

    def _request_name_and_model_from_user(self):
        existing_agent_names = [
            agent["agentName"] for agent in self.bedrock_wrapper.list_agents()
        ]

        while True:
            name = q.ask("Enter an agent name: ", self.is_valid_agent_name)
            if name.lower() not in [n.lower() for n in existing_agent_names]:
                break
            print(
                f"Agent {name} conflicts with an existing agent. Please use a
different name."
            )

        models = ["anthropic.claude-instant-v1", "anthropic.claude-v2"]
        model_id = models[
            q.choose("Which foundation model would you like to use? ", models)
        ]

        return name, model_id

    def _create_agent_role(self, model_id):
        role_name = f"AmazonBedrockExecutionRoleForAgents_{self.postfix}"
        model_arn = f"arn:aws:bedrock:{REGION}::foundation-model/{model_id}*"

        print("Creating an an execution role for the agent...")

        try:
            role = self.iam_resource.create_role(
                RoleName=role_name,
                AssumeRolePolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
```

```
                                "Effect": "Allow",
                                "Principal": {"Service": "bedrock.amazonaws.com"},
                                "Action": "sts:AssumeRole",
                            }
                        ],
                    }
                ),
            )

            role.Policy(ROLE_POLICY_NAME).put(
                PolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "bedrock:InvokeModel",
                                "Resource": model_arn,
                            }
                        ],
                    }
                )
            )
        except ClientError as e:
            logger.error(f"Couldn't create role {role_name}. Here's why: {e}")
            raise

        return role

    def _create_agent(self, name, model_id):
        print("Creating the agent...")

        instruction = """
            You are a friendly chat bot. You have access to a function called that
returns
            information about the current date and time. When responding with date
or time,
            please make sure to add the timezone UTC.
            """
        agent = self.bedrock_wrapper.create_agent(
            agent_name=name,
            foundation_model=model_id,
            instruction=instruction,
            role_arn=self.agent_role.arn,
```

```
        )
        self._wait_for_agent_status(agent["agentId"], "NOT_PREPARED")

        return agent

    def _prepare_agent(self):
        print("Preparing the agent...")

        agent_id = self.agent["agentId"]
        prepared_agent_details = self.bedrock_wrapper.prepare_agent(agent_id)
        self._wait_for_agent_status(agent_id, "PREPARED")

        return prepared_agent_details

    def _create_lambda_function(self):
        print("Creating the Lambda function...")

        function_name = f"AmazonBedrockExampleFunction_{self.postfix}"

        self.lambda_role = self._create_lambda_role()

        try:
            deployment_package = self._create_deployment_package(function_name)

            lambda_function = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="Lambda function for Amazon Bedrock example",
                Runtime="python3.11",
                Role=self.lambda_role.arn,
                Handler=f"{function_name}.lambda_handler",
                Code={"ZipFile": deployment_package},
                Publish=True,
            )

            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)

        except ClientError as e:
            logger.error(
                f"Couldn't create Lambda function {function_name}. Here's why: {e}"
            )
            raise

        return lambda_function
```

```
    def _create_lambda_role(self):
        print("Creating an execution role for the Lambda function...")

        role_name = f"AmazonBedrockExecutionRoleForLambda_{self.postfix}"

        try:
            role = self.iam_resource.create_role(
                RoleName=role_name,
                AssumeRolePolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {"Service": "lambda.amazonaws.com"},
                                "Action": "sts:AssumeRole",
                            }
                        ],
                    }
                ),
            )
            role.attach_policy(
                PolicyArn="arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"
            )
            print(f"Created role {role_name}")
        except ClientError as e:
            logger.error(f"Couldn't create role {role_name}. Here's why: {e}")
            raise

        print("Waiting for the execution role to be fully propagated...")
        wait(10)

        return role

    def _allow_agent_to_invoke_function(self):
        policy = self.iam_resource.RolePolicy(
            self.agent_role.role_name, ROLE_POLICY_NAME
        )
        doc = policy.policy_document
        doc["Statement"].append(
            {
                "Effect": "Allow",
```

```
                "Action": "lambda:InvokeFunction",
                "Resource": self.lambda_function["FunctionArn"],
            }
        )
        self.agent_role.Policy(ROLE_POLICY_NAME).put(PolicyDocument=json.dumps(doc))

    def _let_function_accept_invocations_from_agent(self):
        try:
            self.lambda_client.add_permission(
                FunctionName=self.lambda_function["FunctionName"],
                SourceArn=self.agent["agentArn"],
                StatementId="BedrockAccess",
                Action="lambda:InvokeFunction",
                Principal="bedrock.amazonaws.com",
            )
        except ClientError as e:
            logger.error(
                f"Couldn't grant Bedrock permission to invoke the Lambda function.
Here's why: {e}"
            )
            raise

    def _create_agent_action_group(self):
        print("Creating an action group for the agent...")

        try:
            with open("./scenario_resources/api_schema.yaml") as file:
                self.bedrock_wrapper.create_agent_action_group(
                    name="current_date_and_time",
                    description="Gets the current date and time.",
                    agent_id=self.agent["agentId"],
                    agent_version=self.prepared_agent_details["agentVersion"],
                    function_arn=self.lambda_function["FunctionArn"],
                    api_schema=json.dumps(yaml.safe_load(file)),
                )
        except ClientError as e:
            logger.error(f"Couldn't create agent action group. Here's why: {e}")
            raise

    def _get_agent(self):
        return self.bedrock_wrapper.get_agent(self.agent["agentId"])

    def _get_agent_action_groups(self):
        return self.bedrock_wrapper.list_agent_action_groups(
```

```
            self.agent["agentId"], self.prepared_agent_details["agentVersion"]
        )

    def _get_agent_knowledge_bases(self):
        return self.bedrock_wrapper.list_agent_knowledge_bases(
            self.agent["agentId"], self.prepared_agent_details["agentVersion"]
        )

    def _create_agent_alias(self):
        print("Creating an agent alias...")

        agent_alias_name = "test_agent_alias"
        agent_alias = self.bedrock_wrapper.create_agent_alias(
            agent_alias_name, self.agent["agentId"]
        )

        self._wait_for_agent_status(self.agent["agentId"], "PREPARED")

        return agent_alias

    def _wait_for_agent_status(self, agent_id, status):
        while self.bedrock_wrapper.get_agent(agent_id)["agentStatus"] != status:
            wait(2)

    def _chat_with_agent(self, agent_alias):
        print("-" * 88)
        print("The agent is ready to chat.")
        print("Try asking for the date or time. Type 'exit' to quit.")

        # Create a unique session ID for the conversation
        session_id = uuid.uuid4().hex

        while True:
            prompt = q.ask("Prompt: ", q.non_empty)

            if prompt == "exit":
                break

            response = asyncio.run(self._invoke_agent(agent_alias, prompt,
session_id))

            print(f"Agent: {response}")

    async def _invoke_agent(self, agent_alias, prompt, session_id):
```

```
        response = self.bedrock_agent_runtime_client.invoke_agent(
            agentId=self.agent["agentId"],
            agentAliasId=agent_alias["agentAliasId"],
            sessionId=session_id,
            inputText=prompt,
        )

        completion = ""

        for event in response.get("completion"):
            chunk = event["chunk"]
            completion += chunk["bytes"].decode()

        return completion

    def _delete_resources(self):
        if self.agent:
            agent_id = self.agent["agentId"]

            if self.agent_alias:
                agent_alias_id = self.agent_alias["agentAliasId"]
                print("Deleting agent alias...")
                self.bedrock_wrapper.delete_agent_alias(agent_id, agent_alias_id)

            print("Deleting agent...")
            agent_status = self.bedrock_wrapper.delete_agent(agent_id)
["agentStatus"]
            while agent_status == "DELETING":
                wait(5)
                try:
                    agent_status = self.bedrock_wrapper.get_agent(
                        agent_id, log_error=False
                    )["agentStatus"]
                except ClientError as err:
                    if err.response["Error"]["Code"] == "ResourceNotFoundException":
                        agent_status = "DELETED"

        if self.lambda_function:
            name = self.lambda_function["FunctionName"]
            print(f"Deleting function '{name}'...")
            self.lambda_client.delete_function(FunctionName=name)

        if self.agent_role:
            print(f"Deleting role '{self.agent_role.role_name}'...")
```

```
            self.agent_role.Policy(ROLE_POLICY_NAME).delete()
            self.agent_role.delete()

        if self.lambda_role:
            print(f"Deleting role '{self.lambda_role.role_name}'...")
            for policy in self.lambda_role.attached_policies.all():
                policy.detach_role(RoleName=self.lambda_role.role_name)
            self.lambda_role.delete()

    def _list_resources(self):
        print("-" * 40)
        print(f"Here is the list of created resources in '{REGION}'.")
        print("Make sure you delete them once you're done to avoid unnecessary
costs.")
        if self.agent:
            print(f"Bedrock Agent:   {self.agent['agentName']}")
        if self.lambda_function:
            print(f"Lambda function: {self.lambda_function['FunctionName']}")
        if self.agent_role:
            print(f"IAM role:        {self.agent_role.role_name}")
        if self.lambda_role:
            print(f"IAM role:        {self.lambda_role.role_name}")

    @staticmethod
    def is_valid_agent_name(answer):
        valid_regex = r"^[a-zA-Z0-9_-]{1,100}$"
        return (
            answer
            if answer and len(answer) <= 100 and re.match(valid_regex, answer)
            else None,
            "I need a name for the agent, please. Valid characters are a-z, A-Z,
0-9, _ (underscore) and - (hyphen).",
        )

    @staticmethod
    def _create_deployment_package(function_name):
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, "w") as zipped:
            zipped.write(
                "./scenario_resources/lambda_function.py", f"{function_name}.py"
            )
        buffer.seek(0)
        return buffer.read()
```

```
if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    postfix = "".join(
        random.choice(string.ascii_lowercase + "0123456789") for _ in range(8)
    )
    scenario = BedrockAgentScenarioWrapper(
        bedrock_agent_client=boto3.client(
            service_name="bedrock-agent", region_name=REGION
        ),
        runtime_client=boto3.client(
            service_name="bedrock-agent-runtime", region_name=REGION
        ),
        lambda_client=boto3.client(service_name="lambda", region_name=REGION),
        iam_resource=boto3.resource("iam"),
        postfix=postfix,
    )
    try:
        scenario.run_scenario()
    except Exception as e:
        logging.exception(f"Something went wrong with the demo. Here's what: {e}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência de API do AWS SDK para Python (Boto3).
  - CreateAgent
  - CreateAgentActionGroup
  - CreateAgentAlias
  - DeleteAgent
  - DeleteAgentAlias
  - GetAgent
  - ListAgentActionGroups
  - ListAgentKnowledgeBases
  - ListAgents
  - PrepareAgent

Orquestrar aplicações de IA generativa com o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime

- Funções de etapa

Exemplos do Amazon Bedrock Agents Runtime usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) Amazon Bedrock Agents Runtime.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como usar InvokeFlow para conversar com um fluxo do Amazon Bedrock que inclui um nó de agente.

Consulte mais informações em Converse with an Amazon Bedrock flow.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Shows how to run an Amazon Bedrock flow with InvokeFlow and handle muli-turn
 interaction
for a single conversation.
For more information, see https://docs.aws.amazon.com/bedrock/latest/userguide/
flows-multi-turn-invocation.html.

"""
import logging
import boto3
import botocore

import botocore.exceptions

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


def invoke_flow(client, flow_id, flow_alias_id, input_data, execution_id):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.
        execution_id: Execution ID for continuing a flow. Use the value None on
 first run.

    Returns:
        Dict containing flow_complete status, input_required info, and execution_id
```

```
    """

    response = None
    request_params = None

    if execution_id is None:
        # Don't pass execution ID for first run.
        request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }
    else:
        request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "executionId": execution_id,
            "inputs": [input_data],
            "enableTrace": True
        }

    response = client.invoke_flow(**request_params)

    if "executionId" not in request_params:
        execution_id = response['executionId']

    input_required = None
    flow_status = ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Check if more input us needed from user.
        elif 'flowMultiTurnInputRequestEvent' in event:
            input_required = event

        # Print the model output.
        elif 'flowOutputEvent' in event:
            print(event['flowOutputEvent']['content']['document'])
```

```
        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "input_required": input_required,
        "execution_id": execution_id
    }


def converse_with_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Run a conversation with the supplied flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """

    flow_execution_id = None
    finished = False

    # Get the intial prompt from the user.
    user_input = input("Enter input: ")

    # Use prompt to create input data.
    flow_input_data = {
        "content": {
            "document": user_input
        },
        "nodeName": "FlowInputNode",
        "nodeOutputName": "document"
    }

    try:
        while not finished:
            # Invoke the flow until successfully finished.

            result = invoke_flow(
```

```
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data,
 flow_execution_id)

            status = result['flow_status']
            flow_execution_id = result['execution_id']
            more_input = result['input_required']
            if status == "INPUT_REQUIRED":
                # The flow needs more information from the user.
                logger.info("The flow %s requires more input", flow_id)
                user_input = input(
                    more_input['flowMultiTurnInputRequestEvent']['content']
['document'] + ": ")
                flow_input_data = {
                    "content": {
                        "document": user_input
                    },
                    "nodeName": more_input['flowMultiTurnInputRequestEvent']
['nodeName'],
                    "nodeInputName": "agentInputText"

                }
            elif status == "SUCCESS":
                # The flow completed successfully.
                finished = True
                logger.info("The flow %s successfully completed.", flow_id)

    except botocore.exceptions.ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})

    except Exception as e:
        print(f"An error occurred: {str(e)}")
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})


def main():
    """
    Main entry point for the script.
    """

    # Replace these with your actual flow ID and flow alias ID.
    FLOW_ID = 'YOUR_FLOW_ID'
    FLOW_ALIAS_ID = 'YOUR_FLOW_ALIAS_ID'
```

```
    logger.info("Starting conversation with FLOW: %s ID: %s",
                FLOW_ID, FLOW_ALIAS_ID)

    # Get the Bedrock agent runtime client.
    session = boto3.Session(profile_name='default')
    bedrock_agent_client = session.client('bedrock-agent-runtime')

    # Start the conversation.
    converse_with_flow(bedrock_agent_client, FLOW_ID, FLOW_ALIAS_ID)

    logger.info("Conversation with FLOW: %s ID: %s finished",
                FLOW_ID, FLOW_ALIAS_ID)


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a InvokeFlow Referência da API AWS SDK for Python (Boto3).

Ações

`InvokeAgent`

O código de exemplo a seguir mostra como usar `InvokeAgent`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Invoque um agente.

```
    def invoke_agent(self, agent_id, agent_alias_id, session_id, prompt):
        """
        Sends a prompt for the agent to process and respond to.
```

```
        :param agent_id: The unique identifier of the agent to use.
        :param agent_alias_id: The alias of the agent to use.
        :param session_id: The unique identifier of the session. Use the same value
across requests
                            to continue the same conversation.
        :param prompt: The prompt that you want Claude to complete.
        :return: Inference response from the model.
        """

        try:
            # Note: The execution time depends on the foundation model, complexity
of the agent,
            # and the length of the prompt. In some cases, it can take up to a
minute or more to
            # generate a response.
            response = self.agents_runtime_client.invoke_agent(
                agentId=agent_id,
                agentAliasId=agent_alias_id,
                sessionId=session_id,
                inputText=prompt,
            )

            completion = ""

            for event in response.get("completion"):
                chunk = event["chunk"]
                completion = completion + chunk["bytes"].decode()

        except ClientError as e:
            logger.error(f"Couldn't invoke agent. {e}")
            raise

        return completion
```

- Para obter detalhes da API, consulte a InvokeAgent Referência da API AWS SDK for Python (Boto3).

`InvokeFlow`

O código de exemplo a seguir mostra como usar `InvokeFlow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Invoque um fluxo.

```
    def invoke_flow(self, flow_id, flow_alias_id, input_data, execution_id):
        """
        Invoke an Amazon Bedrock flow and handle the response stream.

        Args:
            param flow_id: The ID of the flow to invoke.
            param flow_alias_id: The alias ID of the flow.
            param input_data: Input data for the flow.
            param execution_id: Execution ID for continuing a flow. Use the value
None on first run.

        Return: Response from the flow.
        """
        try:

            request_params = None

            if execution_id is None:
                # Don't pass execution ID for first run.
                request_params = {
                    "flowIdentifier": flow_id,
                    "flowAliasIdentifier": flow_alias_id,
                    "inputs": input_data,
                    "enableTrace": True
                }
            else:
                request_params = {
                    "flowIdentifier": flow_id,
                    "flowAliasIdentifier": flow_alias_id,
                    "executionId": execution_id,
                    "inputs": input_data,
                    "enableTrace": True
                }
```

```
        response = self.agents_runtime_client.invoke_flow(**request_params)

        if "executionId" not in request_params:
            execution_id = response['executionId']

        result = ""

        # Get the streaming response
        for event in response['responseStream']:
            result = result + str(event) + '\n'
        print(result)

    except ClientError as e:
        logger.error("Couldn't invoke flow %s.", {e})
        raise

    return result
```

- Para obter detalhes da API, consulte a InvokeFlow Referência da API AWS SDK for Python (Boto3).

Cenários

Criar e invocar um fluxo

O exemplo de código a seguir mostra como:

- Criar um perfil de execução para o fluxo.
- Criar o fluxo.
- Implantar o fluxo totalmente configurado.
- Invocar o fluxo com prompts fornecidos pelo usuário.
- Exclua todos os recursos criados.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gera uma playlist de música com base no gênero e no número de músicas especificados pelo usuário.

```
from datetime import datetime
import logging
import boto3

from botocore.exceptions import ClientError

from roles import create_flow_role, delete_flow_role, update_role_policy
from flow import create_flow, prepare_flow, delete_flow
from run_flow import run_playlist_flow
from flow_version import create_flow_version, delete_flow_version
from flow_alias import create_flow_alias, delete_flow_alias

logging.basicConfig(
    level=logging.INFO
)
logger = logging.getLogger(__name__)

def create_input_node(name):
    """
    Creates an input node configuration for an Amazon Bedrock flow.

    The input node serves as the entry point for the flow and defines
    the initial document structure that will be passed to subsequent nodes.

    Args:
        name (str): The name of the input node.

    Returns:
        dict: The input node configuration.

    """
```

```
    return {
        "type": "Input",
        "name": name,
        "outputs": [
            {
                "name": "document",
                "type": "Object"
            }
        ]
    }


def create_prompt_node(name, model_id):
    """
    Creates a prompt node configuration for a Bedrock flow that generates music
 playlists.

    The prompt node defines an inline prompt template that creates a music playlist
 based on
    a specified genre and number of songs. The prompt uses two variables that are
 mapped from
    the input JSON object:
    - {{genre}}: The genre of music to create a playlist for
    - {{number}}: The number of songs to include in the playlist

    Args:
        name (str): The name of the prompt node.
        model_id (str): The identifier of the foundation model to use for the
 prompt.

    Returns:
        dict: The prompt node.

    """

    return {
        "type": "Prompt",
        "name": name,
        "configuration": {
            "prompt": {
                "sourceConfiguration": {
                    "inline": {
                        "modelId": model_id,
                        "templateType": "TEXT",
```

```
                        "inferenceConfiguration": {
                            "text": {
                                "temperature": 0.8
                            }
                        },
                        "templateConfiguration": {
                            "text": {
                                "text": "Make me a {{genre}} playlist consisting of
 the following number of songs: {{number}}."
                            }
                        }
                    }
                }
            }
        },
        "inputs": [
            {
                "name": "genre",
                "type": "String",
                "expression": "$.data.genre"
            },
            {
                "name": "number",
                "type": "Number",
                "expression": "$.data.number"
            }
        ],
        "outputs": [
            {
                "name": "modelCompletion",
                "type": "String"
            }
        ]
    }


def create_output_node(name):
    """
    Creates an output node configuration for a Bedrock flow.

    The output node validates that the output from the last node is a string
    and returns it unmodified. The input name must be "document".

    Args:
```

```
        name (str): The name of the output node.

    Returns:
        dict: The output node configuration containing the output node:

    """

    return {
        "type": "Output",
        "name": name,
        "inputs": [
            {
                "name": "document",
                "type": "String",
                "expression": "$.data"
            }
        ]
    }




def create_playlist_flow(client, flow_name, flow_description, role_arn,
 prompt_model_id):
    """
    Creates the playlist generator flow.
    Args:
        client: bedrock agent boto3 client.
        role_arn (str): Name for the new IAM role.
        prompt_model_id (str): The id of the model to use in the prompt node.
    Returns:
        dict: The response from the create_flow operation.
    """

    input_node = create_input_node("FlowInput")
    prompt_node = create_prompt_node("MakePlaylist", prompt_model_id)
    output_node = create_output_node("FlowOutput")

    # Create connections between the nodes
    connections = []

    #  First, create connections between the output of the flow
    # input node and each input of the prompt node.
    for prompt_node_input in prompt_node["inputs"]:
```

```
        connections.append(
            {
                "name": "_".join([input_node["name"], prompt_node["name"],
                                  prompt_node_input["name"]]),
                "source": input_node["name"],
                "target": prompt_node["name"],
                "type": "Data",
                "configuration": {
                    "data": {
                        "sourceOutput": input_node["outputs"][0]["name"],
                        "targetInput": prompt_node_input["name"]
                    }
                }
            }
        )

    # Then, create a connection between the output of the prompt node and the input
of the flow output node
    connections.append(
        {
            "name": "_".join([prompt_node["name"], output_node["name"]]),
            "source": prompt_node["name"],
            "target": output_node["name"],
            "type": "Data",
            "configuration": {
                "data": {
                    "sourceOutput": prompt_node["outputs"][0]["name"],
                    "targetInput": output_node["inputs"][0]["name"]
                }
            }
        }
    )

    flow_def = {
        "nodes": [input_node, prompt_node, output_node],
        "connections": connections
    }

    # Create the flow.

    response = create_flow(
        client, flow_name, flow_description, role_arn, flow_def)

    return response
```

```
def get_model_arn(client, model_id):
    """
    Gets the Amazon Resource Name (ARN) for a model.
    Args:
        client (str): Amazon Bedrock boto3 client.
        model_id (str): The id of the model.
    Returns:
        str: The ARN of the model.
    """

    try:
        # Call GetFoundationModelDetails operation
        response = client.get_foundation_model(modelIdentifier=model_id)

        # Extract model ARN from the response
        model_arn = response['modelDetails']['modelArn']

        return model_arn

    except ClientError as e:
        logger.exception("Client error getting model ARN: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexpected error getting model ARN: %s", {str(e)})
        raise


def prepare_flow_version_and_alias(bedrock_agent_client,
                                   flow_id):
    """
    Prepares the flow and then creates a flow version and flow alias.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        flowd_id (str): The ID of the flow that you want to prepare.
    Returns: The flow_version and flow_alias.

    """

    status = prepare_flow(bedrock_agent_client, flow_id)
```

```
    flow_version = None
    flow_alias = None

    if status == 'Prepared':

        # Create the flow version and alias.
        flow_version = create_flow_version(bedrock_agent_client,
                                           flow_id,
                                           f"flow version for flow {flow_id}.")

        flow_alias = create_flow_alias(bedrock_agent_client,
                                       flow_id,
                                       flow_version,
                                       "latest",
                                       f"Alias for flow {flow_id}, version
 {flow_version}")

    return flow_version, flow_alias



def delete_role_resources(bedrock_agent_client,
                          iam_client,
                          role_name,
                          flow_id,
                          flow_version,
                          flow_alias):
    """
    Deletes the flow, flow alias, flow version, and IAM roles.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        iam_client: Amazon IAM boto3 client.
        role_name (str): The name of the IAM role.
        flow_id (str): The id of the flow.
        flow_version (str): The version of the flow.
        flow_alias (str): The alias of the flow.
    """

    if flow_id is not None:
        if flow_alias is not None:
            delete_flow_alias(bedrock_agent_client, flow_id, flow_alias)
        if flow_version is not None:
            delete_flow_version(bedrock_agent_client,
                        flow_id, flow_version)
```

```
        delete_flow(bedrock_agent_client, flow_id)

    if role_name is not None:
        delete_flow_role(iam_client, role_name)



def main():
    """
    Creates, runs, and optionally deletes a Bedrock flow for generating music
 playlists.

    Note:
        Requires valid AWS credentials in the default profile
    """

    delete_choice = "y"
    try:

        # Get various boto3 clients.
        session = boto3.Session(profile_name='default')
        bedrock_agent_runtime_client = session.client('bedrock-agent-runtime')
        bedrock_agent_client = session.client('bedrock-agent')
        bedrock_client = session.client('bedrock')
        iam_client = session.client('iam')

        role_name = None
        flow_id = None
        flow_version = None
        flow_alias = None

        #Change the model as needed.
        prompt_model_id = "amazon.nova-pro-v1:0"

        # Base the flow name on the current date and time
        current_time = datetime.now()
        timestamp = current_time.strftime("%Y-%m-%d-%H-%M-%S")
        flow_name = f"FlowPlayList_{timestamp}"
        flow_description = "A flow to generate a music playlist."

        # Create a role for the flow.
        role_name = f"BedrockFlowRole-{flow_name}"
        role = create_flow_role(iam_client, role_name)
        role_arn = role['Arn']
```

```
        # Create the flow.
        response = create_playlist_flow(
            bedrock_agent_client, flow_name, flow_description, role_arn,
prompt_model_id)
        flow_id = response.get('id')

        if flow_id:
            # Update accessible resources in the role.
            model_arn = get_model_arn(bedrock_client, prompt_model_id)
            update_role_policy(iam_client, role_name, [
                               response.get('arn'), model_arn])

            # Prepare the flow and flow version.
            flow_version, flow_alias = prepare_flow_version_and_alias(
                bedrock_agent_client, flow_id)

            # Run the flow.
            if flow_version and flow_alias:
                run_playlist_flow(bedrock_agent_runtime_client,
                                  flow_id, flow_alias)

                delete_choice = input("Delete flow? y or n : ").lower()


            else:
                print("Couldn't run. Deleting flow and role.")
                delete_flow(bedrock_agent_client, flow_id)
                delete_flow_role(iam_client, role_name)
        else:
            print("Couldn't create flow.")


    except Exception as e:
        print(f"Fatal error: {str(e)}")

    finally:
        if delete_choice == 'y':
                delete_role_resources(bedrock_agent_client,
                                      iam_client,
                                      role_name,
                                      flow_id,
                                      flow_version,
                                      flow_alias)
```

```
        else:
            print("Flow not deleted. ")
            print(f"\tFlow ID: {flow_id}")
            print(f"\tFlow version: {flow_version}")
            print(f"\tFlow alias: {flow_alias}")
            print(f"\tRole ARN: {role_arn}")

        print("Done!")

if __name__ == "__main__":
    main()


def invoke_flow(client, flow_id, flow_alias_id, input_data):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.

    Returns:
        Dict containing flow status and flow output.
    """

    response = None
    request_params = None

    request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }


    response = client.invoke_flow(**request_params)

    flow_status = ""
    output= ""

    # Process the streaming response
```

```
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Save the model output.
        elif 'flowOutputEvent' in event:
            output = event['flowOutputEvent']['content']['document']
            logger.info("Output : %s", output)

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "output": output

    }




def run_playlist_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Runs the playlist generator flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """



    print ("Welcome to the playlist generator flow.")
    # Get the initial prompt from the user.
    genre = input("Enter genre: ")
    number_of_songs = int(input("Enter number of songs: "))


    # Use prompt to create input data for the input node.
    flow_input_data = {
```

```
        "content": {
            "document": {
                "genre" : genre,
                "number" : number_of_songs
            }
        },
        "nodeName": "FlowInput",
        "nodeOutputName": "document"
    }

    try:

        result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data)

        status = result['flow_status']

        if status == "SUCCESS":
                # The flow completed successfully.
                logger.info("The flow %s successfully completed.", flow_id)
                print(result['output'])
        else:
            logger.warning("Flow status: %s",status)

    except ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})
        raise

    except Exception as e:
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})
        raise



def create_flow_role(client, role_name):
    """
    Creates an IAM role for Amazon Bedrock with permissions to run a flow.

    Args:
        role_name (str): Name for the new IAM role.
    Returns:
        str: The role Amazon Resource Name.
```

```
    """


    # Trust relationship policy - allows Amazon Bedrock service to assume this role.
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "bedrock.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }]
    }

    # Basic inline policy for for running a flow.

    resources = "*"

    bedrock_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                # Using * as placeholder - Later you update with specific ARNs.
                "Resource": resources
            }
        ]
    }



    try:
        # Create the IAM role with trust policy
        logging.info("Creating role: %s",role_name)
        role = client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy),
            Description="Role for Amazon Bedrock operations"
```

```
        )

        # Attach inline policy to the role
        print("Attaching inline policy")
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(bedrock_policy)
        )

        logging.info("Create Role ARN: %s", role['Role']['Arn'])
        return role['Role']

    except ClientError as e:
        logging.warning("Error creating role: %s", str(e))
        raise
    except Exception as e:
        logging.warning("Unexpected error: %s", str(e))
        raise


def update_role_policy(client, role_name, resource_arns):
    """
    Updates an IAM role's inline policy with specific resource ARNs.

    Args:
        role_name (str): Name of the existing role.
        resource_arns (list): List of resource ARNs to allow access to.
    """


    updated_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:GetFlow",
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                "Resource": resource_arns
            }
```

```
        ]
    }

    try:
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(updated_policy)
        )
        logging.info("Updated policy for role: %s",role_name)

    except ClientError as e:
        logging.warning("Error updating role policy: %s", str(e))
        raise


def delete_flow_role(client, role_name):
    """
    Deletes an IAM role.

    Args:
        role_name (str): Name of the role to delete.
    """




    try:
        # Detach and delete inline policies
        policies = client.list_role_policies(RoleName=role_name)['PolicyNames']
        for policy_name in policies:
            client.delete_role_policy(RoleName=role_name, PolicyName=policy_name)

        # Delete the role
        client.delete_role(RoleName=role_name)
        logging.info("Deleted role: %s", role_name)


    except ClientError as e:
        logging.info("Error Deleting role: %s", str(e))
        raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateFlow
  - CreateFlowAlias
  - CreateFlowVersion
  - DeleteFlow
  - DeleteFlowAlias
  - DeleteFlowVersion
  - GetFlow
  - GetFlowAlias
  - GetFlowVersion
  - InvokeFlow
  - PrepareFlow

Orquestrar aplicações de IA generativa com o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.
- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência,

como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.

- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Funções de etapa

CloudFormation exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with CloudFormation.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie uma API REST para rastrear COVID-19 dados

O exemplo de código a seguir mostra como criar uma API REST que simula um sistema para rastrear casos diários de COVID-19 nos Estados Unidos usando dados fictícios.

SDK para Python (Boto3)

Mostra como usar o AWS Chalice com o AWS SDK para Python (Boto3) para criar uma API REST sem servidor que usa o Amazon API Gateway e o Amazon AWS Lambda DynamoDB. A API REST simula um sistema que rastreia casos diários de COVID-19 nos Estados Unidos, usando dados fictícios. Aprenda como:

- Use o AWS Chalice para definir rotas em funções do Lambda que são chamadas para lidar com solicitações REST que vêm por meio do API Gateway.
- Usar as funções do Lambda para recuperar e armazenar dados em uma tabela do DynamoDB para atender a solicitações REST.
- Defina a estrutura da tabela e os recursos da função de segurança em um AWS CloudFormation modelo.
- Use o AWS Chalice e CloudFormation empacote e implante todos os recursos necessários.
- Use CloudFormation para limpar todos os recursos criados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

CloudFront exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with CloudFront.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetDistributionConfig`

O código de exemplo a seguir mostra como usar `GetDistributionConfig`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )
```

```
        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- Para obter detalhes da API, consulte a GetDistributionConfig Referência da API AWS SDK for Python (Boto3).

`ListDistributions`

O código de exemplo a seguir mostra como usar `ListDistributions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
```

```
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions["DistributionList"]["Quantity"] > 0:
            for distribution in distributions["DistributionList"]["Items"]:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(
                    f"Certificate Source: "
                    f"{distribution['ViewerCertificate']['CertificateSource']}"
                )
                if distribution["ViewerCertificate"]["CertificateSource"] == "acm":
                    print(
                        f"Certificate: {distribution['ViewerCertificate']
['Certificate']}"
                    )
                print("")
        else:
            print("No CloudFront distributions detected.")
```

- Para obter detalhes da API, consulte a ListDistributions Referência da API AWS SDK for Python (Boto3).

`UpdateDistribution`

O código de exemplo a seguir mostra como usar `UpdateDistribution`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- Para obter detalhes da API, consulte a UpdateDistribution Referência da API AWS SDK for Python (Boto3).

CloudWatch exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DeleteAlarms`

O código de exemplo a seguir mostra como usar `DeleteAlarms`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
```

```
        """
        self.cloudwatch_resource = cloudwatch_resource


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteAlarms Referência da API AWS SDK for Python (Boto3).

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
        return alarm_iter
```

- Para obter detalhes da API, consulte a DescribeAlarmsForMetric Referência da API AWS SDK for Python (Boto3).

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
```

```
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise, they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- Para obter detalhes da API, consulte a DisableAlarmActions Referência da API AWS SDK for Python (Boto3).

`EnableAlarmActions`

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise, they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- Para obter detalhes da API, consulte a EnableAlarmActions Referência da API AWS SDK for Python (Boto3).

`GetMetricStatistics`

O código de exemplo a seguir mostra como usar `GetMetricStatistics`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_statistics(self, namespace, name, start, end, period,
 stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
 grouped
        into the specified period.
```

```
        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average value
                           or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period, Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
            raise
        else:
            return stats
```

- Para obter detalhes da API, consulte a GetMetricStatistics Referência da API AWS SDK for Python (Boto3).

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def list_metrics(self, namespace, name, recent=False):
        """
        Gets the metrics within a namespace that have the specified name.
        If the metric has no dimensions, a single metric is returned.
        Otherwise, metrics for all dimensions are returned.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param recent: When True, only metrics that have been active in the last
                       three hours are returned.
        :return: An iterator that yields the retrieved metrics.
        """
        try:
            kwargs = {"Namespace": namespace, "MetricName": name}
            if recent:
                kwargs["RecentlyActive"] = "PT3H"  # List past 3 hours only
            metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
            logger.info("Got metrics for %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
            raise
        else:
            return metric_iter
```

- Para obter detalhes da API, consulte a ListMetrics Referência da API AWS SDK for Python (Boto3).

`PutMetricAlarm`

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.
```

```
        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over the
                             alarm threshold before the alarm is set into an alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm
```

- Para obter detalhes da API, consulte a PutMetricAlarm Referência da API AWS SDK for Python (Boto3).

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
```

```
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace, name)
            raise
```

Coloque um conjunto de dados em uma CloudWatch métrica.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
```

```
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.", namespace,
name)
            raise
```

- Para obter detalhes da API, consulte a PutMetricData Referência da API AWS SDK for Python (Boto3).

Cenários

Gerencie métricas e alarmes

O exemplo de código a seguir mostra como:

- Crie um alarme para observar uma CloudWatch métrica.
- Inserir dados em uma métrica e acionar o alarme.
- Obter dados do alarme.
- Excluir o alarme.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as CloudWatch operações.

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
import time
```

```
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
```

```
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.", namespace,
name)
            raise


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over the
                             alarm threshold before the alarm is set into an alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
```

```
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace, name)
            raise


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
```

```
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average value
                           or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period, Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
            raise
        else:
            return stats


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
```

```
        logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
        return alarm_iter


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

Use a classe criada para colocar dados em uma métrica, acionar um alarme que observa a métrica e obter dados do alarme.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    cw_wrapper = CloudWatchWrapper(boto3.resource("cloudwatch"))

    minutes = 20
    metric_namespace = "doc-example-metric"
    metric_name = "page_views"
```

```
    start = datetime.utcnow() - timedelta(minutes=minutes)
    print(
        f"Putting data into metric {metric_namespace}.{metric_name} spanning the "
        f"last {minutes} minutes."
    )
    for offset in range(0, minutes):
        stamp = start + timedelta(minutes=offset)
        cw_wrapper.put_metric_data_set(
            metric_namespace,
            metric_name,
            stamp,
            "Count",
            {
                "values": [
                    random.randint(bound, bound * 2)
                    for bound in range(offset + 1, offset + 11)
                ],
                "counts": [random.randint(1, offset + 1) for _ in range(10)],
            },
        )

    alarm_name = "high_page_views"
    period = 60
    eval_periods = 2
    print(f"Creating alarm {alarm_name} for metric {metric_name}.")
    alarm = cw_wrapper.create_metric_alarm(
        metric_namespace,
        metric_name,
        alarm_name,
        "Maximum",
        period,
        eval_periods,
        100,
        "GreaterThanThreshold",
    )
    print(f"Alarm ARN is {alarm.alarm_arn}.")
    print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Sending data to trigger the alarm. This requires data over the threshold "
        f"for {eval_periods} periods of {period} seconds each."
    )
    while alarm.state_value == "INSUFFICIENT_DATA":
        print("Sending data for the metric.")
```

```
        cw_wrapper.put_metric_data(
            metric_namespace, metric_name, random.randint(100, 200), "Count"
        )
        alarm.load()
        print(f"Current alarm state is: {alarm.state_value}.")
        if alarm.state_value == "INSUFFICIENT_DATA":
            print(f"Waiting for {period} seconds...")
            time.sleep(period)
        else:
            print("Wait for a minute for eventual consistency of metric data.")
            time.sleep(period)
            if alarm.state_value == "OK":
                alarm.load()
                print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Getting data for metric {metric_namespace}.{metric_name} during timespan "
        f"of {start} to {datetime.utcnow()} (times are UTC)."
    )
    stats = cw_wrapper.get_metric_statistics(
        metric_namespace,
        metric_name,
        start,
        datetime.utcnow(),
        60,
        ["Average", "Minimum", "Maximum"],
    )
    print(
        f"Got {len(stats['Datapoints'])} data points for metric "
        f"{metric_namespace}.{metric_name}."
    )
    pprint(sorted(stats["Datapoints"], key=lambda x: x["Timestamp"]))

    print(f"Getting alarms for metric {metric_name}.")
    alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
    for alarm in alarms:
        print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

    print(f"Deleting alarms for metric {metric_name}.")
    cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

    print("Thanks for watching!")
    print("-" * 88)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - DeleteAlarms
  - DescribeAlarmsForMetric
  - DisableAlarmActions
  - EnableAlarmActions
  - GetMetricStatistics
  - ListMetrics
  - PutMetricAlarm
  - PutMetricData

CloudWatch Exemplos de registros usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with CloudWatch Logs.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`GetQueryResults`

O código de exemplo a seguir mostra como usar `GetQueryResults`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def _wait_for_query_results(self, client, query_id):
    """
    Waits for the query to complete and retrieves the results.

    :param query_id: The ID of the initiated query.
    :type query_id: str
    :return: A list containing the results of the query.
    :rtype: list
    """
    while True:
        time.sleep(1)
        results = client.get_query_results(queryId=query_id)
        if results["status"] in [
            "Complete",
            "Failed",
            "Cancelled",
            "Timeout",
            "Unknown",
        ]:
            return results.get("results", [])
```

- Para obter detalhes da API, consulte a GetQueryResults Referência da API AWS SDK for Python (Boto3).

StartLiveTail

O código de exemplo a seguir mostra como usar `StartLiveTail`.

SDK para Python (Boto3)

Inclua os arquivos necessários.

```
import boto3
import time
from datetime import datetime
```

Inicie a sessão do Live Tail.

```
    # Initialize the client
    client = boto3.client('logs')

    start_time = time.time()

    try:
        response = client.start_live_tail(
            logGroupIdentifiers=log_group_identifiers,
            logStreamNames=log_streams,
            logEventFilterPattern=filter_pattern
        )
        event_stream = response['responseStream']
        # Handle the events streamed back in the response
        for event in event_stream:
            # Set a timeout to close the stream.
            # This will end the Live Tail session.
            if (time.time() - start_time >= 10):
                event_stream.close()
                break
            # Handle when session is started
            if 'sessionStart' in event:
                session_start_event = event['sessionStart']
                print(session_start_event)
            # Handle when log event is given in a session update
            elif 'sessionUpdate' in event:
                log_events = event['sessionUpdate']['sessionResults']
                for log_event in log_events:
```

```
                    print('[{date}]
{log}'.format(date=datetime.fromtimestamp(log_event['timestamp']/1000),log=log_event['messa
            else:
                # On-stream exceptions are captured here
                raise RuntimeError(str(event))
    except Exception as e:
        print(e)
```

- Para obter detalhes da API, consulte a StartLiveTail Referência da API AWS SDK for Python (Boto3).

`StartQuery`

O código de exemplo a seguir mostra como usar `StartQuery`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for the
query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
```

```
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for the
query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(
                self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
```

```
            end_time = round(
                self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- Para obter detalhes da API, consulte a StartQuery Referência da API AWS SDK for Python (Boto3).

Cenários

Executar uma consulta grande

O exemplo de código a seguir mostra como usar o CloudWatch Logs para consultar mais de 10.000 registros.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse arquivo invoca um módulo de exemplo para gerenciar CloudWatch consultas que excedam 10.000 resultados.

```
import logging
import os
import sys
```

```
import boto3
from botocore.config import Config

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"


class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
        Initializes the CloudWatchLogsQueryRunner class by setting up date utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
        Creates and returns a CloudWatch Logs client with a specified retry
 configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts": 10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and end
 dates.
        Fetches the environment variable for log group, returning the default value
 if it
```

```
        does not exist.

        :return: Tuple of query start date and end date as integers and the log
group.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables are
required."
            )
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)

        try:
            log_group = os.environ["QUERY_LOG_GROUP"]
        except KeyError:
            logging.warning("No QUERY_LOG_GROUP environment variable, using default
value")
            log_group = DEFAULT_QUERY_LOG_GROUP

        return query_start_date, query_end_date, log_group

    def convert_dates_to_iso8601(self, start_date, end_date):
        """
        Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

        :param start_date: The start date in UNIX timestamp.
        :type start_date: int
        :param end_date: The end date in UNIX timestamp.
        :type end_date: int
        :return: Start and end dates in ISO 8601 format.
        :rtype: tuple
        """
        start_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            start_date
        )
```

```
        end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            end_date
        )
        return start_date_iso8601, end_date_iso8601

    def execute_query(
        self,
        start_date_iso8601,
        end_date_iso8601,
        log_group="/workflows/cloudwatch-logs/large-query",
        query="fields @timestamp, @message | sort @timestamp asc"
    ):
        """
        Creates a CloudWatchQuery instance and executes the query with provided date
 range.

        :param start_date_iso8601: The start date in ISO 8601 format.
        :type start_date_iso8601: str
        :param end_date_iso8601: The end date in ISO 8601 format.
        :type end_date_iso8601: str
        :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
        :type log_group: str
        :param query: Query string to pass to the CloudWatchQuery instance
        :type query: str
        """
        cloudwatch_query = CloudWatchQuery(
            log_group=log_group,
            query_string=query
        )
        cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
        logging.info("Query executed successfully.")
        logging.info(
            f"Queries completed in {cloudwatch_query.query_duration} seconds. Total
 logs found: {len(cloudwatch_query.query_results)}"
        )


def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
```

```
    runner = CloudWatchLogsQueryRunner()
    query_start_date, query_end_date, log_group =
 runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
 DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601, log_group=log_group)


if __name__ == "__main__":
    main()
```

Esse módulo processa CloudWatch consultas que excedem 10.000 resultados.

```
import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass


class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    :log_group str: Name of the log group to query
    :query_string str: query
```

```
    """

    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000

    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and calculates
the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
        """
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n       START:     {start_date}"
            f"\n       END:       {end_date}"
            f"\n       LOG GROUP: {self.log_group}"
        )
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
recursively if necessary.

        :param date_range: The date range to fetch logs for, specified as a tuple
(start_timestamp, end_timestamp).
```

```
        :type date_range: tuple
        :return: None if the recursive fetching is continued or stops when the final
batch of logs is processed.
                 Although it doesn't explicitly return the query results, this
method accumulates all fetched logs
                 in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
            new_range = (most_recent_log_timestamp, date_range[1])
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()

            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.
        :return: log
        :type :return List containing log item details
        """
```

```
        most_recent_log = None
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for the
query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
```

```
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for the
query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(
                self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(
                self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
```

```
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")


    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - GetQueryResults
  - StartQuery

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento agendado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função do Lambda o alvo.
- Concede permissão para permitir que a função Lambda seja EventBridge invocada.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações agendadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos de código do Amazon Cognito Identity usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Cognito Identity.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Exemplos de código do Provedor de Identidades Amazon Cognito usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon Cognito

O exemplo de código a seguir mostra como começar a usar o Amazon Cognito.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(MaxResults=10)
```

```
# Initialize variables for pagination
user_pools = []

# Handle pagination
for page in page_iterator:
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- Para obter detalhes da API, consulte a ListUserPools Referência da API AWS SDK for Python (Boto3).

Ações

`AdminGetUser`

O código de exemplo a seguir mostra como usar `AdminGetUser`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
```

```
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code
that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
```

```
                )
                logger.warning(
                    "User %s exists and is %s.", user_name, response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- Para obter detalhes da API, consulte a AdminGetUser Referência da API AWS SDK for Python (Boto3).

`AdminInitiateAuth`

O código de exemplo a seguir mostra como usar `AdminInitiateAuth`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
```

```
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
server.

        If the user pool is configured to require MFA and this is the first sign-in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful, this
                 returns an access token that can be used to get AWS credentials.
Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
```

```
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is not
"
                        "configured for TOTP MFA. This example requires TOTP MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Para obter detalhes da API, consulte a AdminInitiateAuth Referência da API AWS SDK for Python (Boto3).

`AdminRespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `AdminRespondToAuthChallenge`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Responda a um desafio de MFA fornecendo um código gerado por uma aplicação de MFA associada.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step of
        a two-factor sign-in. When sign-in is successful, it returns an access token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
```

```
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You might
have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

- Para obter detalhes da API, consulte a AdminRespondToAuthChallenge Referência da API AWS SDK for Python (Boto3).

`AssociateSoftwareToken`

O código de exemplo a seguir mostra como usar `AssociateSoftwareToken`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
 self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Para obter detalhes da API, consulte a AssociateSoftwareToken Referência da API AWS SDK for Python (Boto3).

`ConfirmDevice`

O código de exemplo a seguir mostra como usar `ConfirmDevice`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
```

```
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False. When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt + device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value, srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
```

```
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- Para obter detalhes da API, consulte a ConfirmDevice Referência da API AWS SDK for Python (Boto3).

`ConfirmSignUp`

O código de exemplo a seguir mostra como usar `ConfirmSignUp`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
```

```
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- Para obter detalhes da API, consulte a ConfirmSignUp Referência da API AWS SDK for Python (Boto3).

`InitiateAuth`

O código de exemplo a seguir mostra como usar `InitiateAuth`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo mostra como iniciar a autenticação com um dispositivo rastreado. Para concluir o login, o cliente deve responder corretamente aos desafios de Secure Remote Password (SRP).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
```

```
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.

        Signing in with a tracked device requires that the client respond to the SRP
        protocol. The scenario associated with this example uses the warrant package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
 {response_init['ChallengeName']}."
                )
```

```
            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- Para obter detalhes da API, consulte a InitiateAuth Referência da API AWS SDK for Python (Boto3).

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
 self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
```

```
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

- Para obter detalhes da API, consulte a ListUsers Referência da API AWS SDK for Python (Boto3).

`ResendConfirmationCode`

O código de exemplo a seguir mostra como usar `ResendConfirmationCode`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- Para obter detalhes da API, consulte a ResendConfirmationCode Referência da API AWS SDK for Python (Boto3).

`RespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `RespondToAuthChallenge`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça login com um dispositivo rastreado. Para concluir o login, o cliente deve responder corretamente aos desafios de Secure Remote Password (SRP).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.

        Signing in with a tracked device requires that the client respond to the SRP
        protocol. The scenario associated with this example uses the warrant package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
```

```
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
```

```
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- Para obter detalhes da API, consulte a RespondToAuthChallenge Referência da API AWS SDK for Python (Boto3).

`SignUp`

O código de exemplo a seguir mostra como usar `SignUp`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code
 that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
 whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
```

```
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name, response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- Para obter detalhes da API, consulte a SignUp Referência da API AWS SDK for Python (Boto3).

`VerifySoftwareToken`

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
```

```
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Para obter detalhes da API, consulte a VerifySoftwareToken Referência da API AWS SDK for Python (Boto3).

Cenários

Inscrever um usuário em um grupo de usuários que exija MFA

O exemplo de código a seguir mostra como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que englobe as funções do Amazon Cognito usadas no cenário.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.

        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code
that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name, response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

```
    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
server.

        If the user pool is configured to require MFA and this is the first sign-in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful, this
                 returns an access token that can be used to get AWS credentials.
Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
```

```
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is not
"
                        "configured for TOTP MFA. This example requires TOTP MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            response.pop("ResponseMetadata", None)
            return response


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step of
        a two-factor sign-in. When sign-in is successful, it returns an access token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
```

```
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You might
have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
```

```
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False. When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt + device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value, srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
```

```
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.

        Signing in with a tracked device requires that the client respond to the SRP
        protocol. The scenario associated with this example uses the warrant package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
```

```
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
```

```
            response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

Crie uma classe que execute o cenário. Este exemplo também registra um dispositivo de MFA a ser rastreado pelo Amazon Cognito e mostra como fazer login usando uma senha e informações do dispositivo rastreado. Isso evita a necessidade de inserir um novo código de MFA.

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)

    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )

    user_name = q.ask("Let's sign up a new user. Enter a user name: ", q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
```

```
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ", q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print("-" * 88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response["ChallengeName"]
        elif response["ChallengeName"] == "MFA_SETUP":
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA "
                "application, such as Google Authenticator."
            )
```

```
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to appear
in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                    user_name, response["Session"], mfa_code
                )
            print(f"You're signed in as {user_name}.")
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
            print("And your device information:")
            pp(auth_tokens["NewDeviceMetadata"])
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge {response['ChallengeName']}")
    print("-" * 88)

    device_group_key = auth_tokens["NewDeviceMetadata"]["DeviceGroupKey"]
    device_key = auth_tokens["NewDeviceMetadata"]["DeviceKey"]
    device_password = base64.standard_b64encode(os.urandom(40)).decode("utf-8")

    print("Let's confirm your MFA device so you don't have re-enter MFA tokens for
it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name,
        device_key,
```

```
        device_group_key,
        device_password,
        auth_tokens["AccessToken"],
        aws_srp,
    )
    print(f"Your device {device_key} is confirmed.")
    print("-" * 88)

    print(
        f"Now let's sign in as {user_name} from your confirmed device {device_key}.
\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code."
    )
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password, aws_srp
    )
    print("You're signed in. Your access token is:")
    pp(auth_tokens["AccessToken"])
    print("-" * 88)

    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
    print("-" * 88)


def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the example."
    )
    args = parser.parse_args()
    try:
        run_scenario(boto3.client("cognito-idp"), args.user_pool_id, args.client_id)
    except Exception:
```

```
        logging.exception("Something went wrong with the demo.")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Usar banco de identidades do Amazon Cognito

O exemplo de código a seguir mostra como criar uma aplicação de demonstração baseada na web que demonstra fluxos de identidades de bancos de identidades.

SDK para Python (Boto3)

Mostra uma aplicação de demonstração baseada na web que demonstra os fluxos de autenticação dos bancos de identidades do Amazon Cognito, permitindo que os usuários explorem interativamente os fluxos de autenticação aprimorada e básica com vários provedores de identidade.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Provedor de identidades do Amazon Cognito

Exemplos do Amazon Comprehend usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Comprehend.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateDocumentClassifier`

O código de exemplo a seguir mostra como usar `CreateDocumentClassifier`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it immediately
        starts training on the data found in the specified Amazon S3 bucket.
Training
        can take 30 minutes or longer. The `describe_document_classifier` function
        can be used to get training status and returns a status of TRAINED when the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
data.
        :param training_key: The prefix used to find training data in the training
                             bucket. If multiple objects have the same prefix, all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
```

```
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn
```

- Para obter detalhes da API, consulte a CreateDocumentClassifier Referência da API AWS SDK for Python (Boto3).

`DeleteDocumentClassifier`

O código de exemplo a seguir mostra como usar `DeleteDocumentClassifier`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def delete(self):
        """
        Deletes the classifier.
```

```
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.", self.classifier_arn)
            raise
```

- Para obter detalhes da API, consulte a DeleteDocumentClassifier Referência da API AWS SDK for Python (Boto3).

`DescribeDocumentClassificationJob`

O código de exemplo a seguir mostra como usar `DescribeDocumentClassificationJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe_job(self, job_id):
        """
```

```
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job
```

- Para obter detalhes da API, consulte a DescribeDocumentClassificationJob Referência da API AWS SDK for Python (Boto3).

`DescribeDocumentClassifier`

O código de exemplo a seguir mostra como usar `DescribeDocumentClassifier`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
```

```
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier
```

- Para obter detalhes da API, consulte a DescribeDocumentClassifier Referência da API AWS SDK for Python (Boto3).

`DescribeTopicsDetectionJob`

O código de exemplo a seguir mostra como usar `DescribeTopicsDetectionJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job
```

- Para obter detalhes da API, consulte a DescribeTopicsDetectionJob Referência da API AWS SDK for Python (Boto3).

`DetectDominantLanguage`

O código de exemplo a seguir mostra como usar `DetectDominantLanguage`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages
```

- Para obter detalhes da API, consulte a DetectDominantLanguage Referência da API AWS SDK for Python (Boto3).

`DetectEntities`

O código de exemplo a seguir mostra como usar `DetectEntities`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
 places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities
```

- Para obter detalhes da API, consulte a DetectEntities Referência da API AWS SDK for Python (Boto3).

`DetectKeyPhrases`

O código de exemplo a seguir mostra como usar `DetectKeyPhrases`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
```

```
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases
```

- Para obter detalhes da API, consulte a DetectKeyPhrases Referência da API AWS SDK for Python (Boto3).

`DetectPiiEntities`

O código de exemplo a seguir mostra como usar `DetectPiiEntities`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
```

```
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities
```

- Para obter detalhes da API, consulte a DetectPiiEntities Referência da API AWS SDK for Python (Boto3).

`DetectSentiment`

O código de exemplo a seguir mostra como usar `DetectSentiment`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
```

```
    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DetectSentiment Referência da API AWS SDK for Python (Boto3).

`DetectSyntax`

O código de exemplo a seguir mostra como usar `DetectSyntax`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

- Para obter detalhes da API, consulte a DetectSyntax Referência da API AWS SDK for Python (Boto3).

`ListDocumentClassificationJobs`

O código de exemplo a seguir mostra como usar `ListDocumentClassificationJobs`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

- Para obter detalhes da API, consulte a ListDocumentClassificationJobs Referência da API AWS SDK for Python (Boto3).

`ListDocumentClassifiers`

O código de exemplo a seguir mostra como usar `ListDocumentClassifiers`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers
```

- Para obter detalhes da API, consulte a ListDocumentClassifiers Referência da API AWS SDK for Python (Boto3).

`ListTopicsDetectionJobs`

O código de exemplo a seguir mostra como usar `ListTopicsDetectionJobs`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

- Para obter detalhes da API, consulte a ListTopicsDetectionJobs Referência da API AWS SDK for Python (Boto3).

StartDocumentClassificationJob

O código de exemplo a seguir mostra como usar `StartDocumentClassificationJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
```

```
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a StartDocumentClassificationJob Referência da API AWS SDK for Python (Boto3).

StartTopicsDetectionJob

O código de exemplo a seguir mostra como usar `StartTopicsDetectionJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
 stored
        in a tar archive compressed in gzip format. The job runs asynchronously, so
 you
        can call `describe_topics_detection_job` to get job status until it
```

```
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                            bucket. If multiple objects have the same prefix, all
                            of them are used.
        :param input_format: The format of the input data, either one document per
                            file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a StartTopicsDetectionJob Referência da API AWS SDK for Python (Boto3).

Cenários

Detectar elementos do documento

O exemplo de código a seguir mostra como:

- Detecte idiomas, entidades e frases-chave em um documento.
- Detecte informações de identificação pessoal (PII) em um documento.
- Detecte o sentimento de um documento.
- Detectar elementos sintáticos em um documento.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva as ações do Amazon Comprehend.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
```

```
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
```

```
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
```

```
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

Chame funções na classe wrapper para detectar entidades, frases e muito mais em um documento.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    comp_detect = ComprehendDetect(boto3.client("comprehend"))
    with open("detect_sample.txt") as sample_file:
        sample_text = sample_file.read()

    demo_size = 3

    print("Sample text used for this demo:")
    print("-" * 88)
    print(sample_text)
    print("-" * 88)

    print("Detecting languages.")
    languages = comp_detect.detect_languages(sample_text)
    pprint(languages)
    lang_code = languages[0]["LanguageCode"]

    print("Detecting entities.")
    entities = comp_detect.detect_entities(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(entities[:demo_size])

    print("Detecting key phrases.")
    phrases = comp_detect.detect_key_phrases(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(phrases[:demo_size])

    print("Detecting personally identifiable information (PII).")
    pii_entities = comp_detect.detect_pii(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(pii_entities[:demo_size])

    print("Detecting sentiment.")
    sentiment = comp_detect.detect_sentiment(sample_text, lang_code)
    print(f"Sentiment: {sentiment['Sentiment']}")
    print("SentimentScore:")
    pprint(sentiment["SentimentScore"])

    print("Detecting syntax elements.")
    syntax_tokens = comp_detect.detect_syntax(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(syntax_tokens[:demo_size])

    print("Thanks for watching!")
```

```
    print("-" * 88)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - DetectDominantLanguage
  - DetectEntities
  - DetectKeyPhrases
  - DetectPiiEntities
  - DetectSentiment
  - DetectSyntax

Detectar entidades em texto extraído de uma imagem

O exemplo de código a seguir mostra como usar o Amazon Comprehend para detectar entidades em texto extraído pelo Amazon Textract de uma imagem armazenada no Amazon S3.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) em um notebook Jupyter para detectar entidades no texto que é extraído de uma imagem. Este exemplo usa o Amazon Textract para extrair texto de uma imagem armazenada no Amazon Simple Storage Service (Amazon S3) e no Amazon Comprehend para detectar entidades no texto extraído.

Este exemplo é um caderno Jupyter e deve ser executado em um ambiente que possa hospedar blocos de anotações. Para obter instruções sobre como executar o exemplo usando o Amazon SageMaker AI, consulte as instruções em TextractAndComprehendNotebook.ipynb.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Execute uma tarefa de modelagem de tópicos em dados de amostra

O exemplo de código a seguir mostra como:

- Execute uma tarefa de modelagem de tópicos do Amazon Comprehend em dados de amostra.
- Informações sobre a tarefa.
- Extrair dados de saída do trabalho no Amazon S3.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe wrapper para chamar as ações de modelagem de tópicos do Amazon Comprehend.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
```

```
        input bucket and written to the specified output bucket. Output data is
stored
        in a tar archive compressed in gzip format. The job runs asynchronously, so
you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                            bucket. If multiple objects have the same prefix, all
                            of them are used.
        :param input_format: The format of the input data, either one document per
                            file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
```

```
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

Use a classe wrapper para executar uma tarefa de modelagem de tópicos e obter dados dela.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend topic modeling demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    input_prefix = "input/"
    output_prefix = "output/"
    demo_resources = ComprehendDemoResources(
        boto3.resource("s3"), boto3.resource("iam")
    )
    topic_modeler = ComprehendTopicModeler(boto3.client("comprehend"))

    print("Setting up storage and security resources needed for the demo.")
    demo_resources.setup("comprehend-topic-modeler-demo")
    print("Copying sample data from public bucket into input bucket.")
    demo_resources.bucket.copy(
        {"Bucket": "public-sample-us-west-2", "Key": "TopicModeling/Sample.txt"},
        f"{input_prefix}sample.txt",
    )

    print("Starting topic modeling job on sample data.")
    job_info = topic_modeler.start_job(
        "demo-topic-modeling-job",
        demo_resources.bucket.name,
        input_prefix,
        JobInputFormat.per_line,
        demo_resources.bucket.name,
        output_prefix,
        demo_resources.data_access_role.arn,
    )

    print(
        f"Waiting for job {job_info['JobId']} to complete. This typically takes "
        f"20 - 30 minutes."
    )
    job_waiter = JobCompleteWaiter(topic_modeler.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = topic_modeler.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from the output Amazon S3 bucket: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = demo_resources.extract_job_output(job)
    lines = 10
    print(f"First {lines} lines of document topics output:")
```

```
    pprint(job_output["doc-topics.csv"]["data"][:lines])
    print(f"First {lines} lines of terms output:")
    pprint(job_output["topic-terms.csv"]["data"][:lines])

    print("Cleaning up resources created for the demo.")
    demo_resources.cleanup()

    print("Thanks for watching!")
    print("-" * 88)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - DescribeTopicsDetectionJob
  - ListTopicsDetectionJobs
  - StartTopicsDetectionJob

Treinar um classificador personalizado e classificar documentos

O exemplo de código a seguir mostra como:

- Crie um classificador de vários rótulos do Amazon Comprehend.
- Treine o classificador em dados de amostra.
- Execute uma tarefa de classificação em um segundo conjunto de dados.
- Extrair dados de saída da tarefa do Amazon S3.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe wrapper para chamar as ações do classificador de documentos Amazon Comprehend.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it immediately
        starts training on the data found in the specified Amazon S3 bucket.
 Training
        can take 30 minutes or longer. The `describe_document_classifier` function
        can be used to get training status and returns a status of TRAINED when the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
 data.
        :param training_key: The prefix used to find training data in the training
                             bucket. If multiple objects have the same prefix, all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
```

```
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
```

```
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.", self.classifier_arn)
            raise


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.
```

```
        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
```

```
        try:
            response = self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

Crie uma classe para ajudar na execução do cenário.

```
class ClassifierDemo:
    """
    Encapsulates functions used to run the demonstration.
    """

    def __init__(self, demo_resources):
        """
```

```
        :param demo_resources: A ComprehendDemoResources class that manages
 resources
                                for the demonstration.
        """
        self.demo_resources = demo_resources
        self.training_prefix = "training/"
        self.input_prefix = "input/"
        self.input_format = JobInputFormat.per_line
        self.output_prefix = "output/"

    def setup(self):
        """Creates AWS resources used by the demo."""
        self.demo_resources.setup("comprehend-classifier-demo")

    def cleanup(self):
        """Deletes AWS resources used by the demo."""
        self.demo_resources.cleanup()

    @staticmethod
    def _sanitize_text(text):
        """Removes characters that cause errors for the document parser."""
        return text.replace("\r", " ").replace("\n", " ").replace(",", ";")

    @staticmethod
    def _get_issues(query, issue_count):
        """
        Gets issues from GitHub using the specified query parameters.

        :param query: The query string used to request issues from the GitHub API.
        :param issue_count: The number of issues to retrieve.
        :return: The list of issues retrieved from GitHub.
        """
        issues = []
        logger.info("Requesting issues from %s?%s.", GITHUB_SEARCH_URL, query)
        response = requests.get(f"{GITHUB_SEARCH_URL}?
{query}&per_page={issue_count}")
        if response.status_code == 200:
            issue_page = response.json()["items"]
            logger.info("Got %s issues.", len(issue_page))
            issues = [
                {
                    "title": ClassifierDemo._sanitize_text(issue["title"]),
                    "body": ClassifierDemo._sanitize_text(issue["body"]),
                    "labels": {label["name"] for label in issue["labels"]},
```

```
                }
                for issue in issue_page
            ]
        else:
            logger.error(
                "GitHub returned error code %s with message %s.",
                response.status_code,
                response.json(),
            )
        logger.info("Found %s issues.", len(issues))
        return issues

    def get_training_issues(self, training_labels):
        """
        Gets issues used for training the custom classifier. Training issues are
        closed issues from the Boto3 repo that have known labels. Comprehend
        requires a minimum of ten training issues per label.

        :param training_labels: The issue labels to use for training.
        :return: The set of issues used for training.
        """
        issues = []
        per_label_count = 15
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:closed+label:{label}",
                per_label_count,
            )
            for issue in issues:
                issue["labels"] = issue["labels"].intersection(training_labels)
        return issues

    def get_input_issues(self, training_labels):
        """
        Gets input issues from GitHub. For demonstration purposes, input issues
        are open issues from the Boto3 repo with known labels, though in practice
        any issue could be submitted to the classifier for labeling.

        :param training_labels: The set of labels to query for.
        :return: The set of issues used for input.
        """
        issues = []
        per_label_count = 5
        for label in training_labels:
```

```
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:open+label:{label}",
                per_label_count,
            )
        return issues

    def upload_issue_data(self, issues, training=False):
        """
        Uploads issue data to an Amazon S3 bucket, either for training or for input.
        The data is first put into the format expected by Comprehend. For training,
        the set of pipe-delimited labels is prepended to each document. For
        input, labels are not sent.

        :param issues: The set of issues to upload to Amazon S3.
        :param training: Indicates whether the issue data is used for training or
                         input.
        """
        try:
            obj_key = (
                self.training_prefix if training else self.input_prefix
            ) + "issues.txt"
            if training:
                issue_strings = [
                    f"{'|'.join(issue['labels'])},{issue['title']} {issue['body']}"
                    for issue in issues
                ]
            else:
                issue_strings = [
                    f"{issue['title']} {issue['body']}" for issue in issues
                ]
            issue_bytes = BytesIO("\n".join(issue_strings).encode("utf-8"))
            self.demo_resources.bucket.upload_fileobj(issue_bytes, obj_key)
            logger.info(
                "Uploaded data as %s to bucket %s.",
                obj_key,
                self.demo_resources.bucket.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload data to bucket %s.",
self.demo_resources.bucket.name
            )
            raise
```

```
    def extract_job_output(self, job):
        """Extracts job output from Amazon S3."""
        return self.demo_resources.extract_job_output(job)

    @staticmethod
    def reconcile_job_output(input_issues, output_dict):
        """
        Reconciles job output with the list of input issues. Because the input
 issues
        have known labels, these can be compared with the labels added by the
        classifier to judge the accuracy of the output.

        :param input_issues: The list of issues used as input.
        :param output_dict: The dictionary of data that is output by the classifier.
        :return: The list of reconciled input and output data.
        """
        reconciled = []
        for archive in output_dict.values():
            for line in archive["data"]:
                in_line = int(line["Line"])
                in_labels = input_issues[in_line]["labels"]
                out_labels = {
                    label["Name"]
                    for label in line["Labels"]
                    if float(label["Score"]) > 0.3
                }
                reconciled.append(
                    f"{line['File']}, line {in_line} has labels {in_labels}.\n"
                    f"\tClassifier assigned {out_labels}."
                )
        logger.info("Reconciled input and output labels.")
        return reconciled
```

Treine um classificador sobre um conjunto de GitHub problemas com rótulos conhecidos e envie um segundo conjunto de GitHub problemas ao classificador para que eles possam ser rotulados.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend custom document classifier demo!")
    print("-" * 88)
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    comp_demo = ClassifierDemo(
        ComprehendDemoResources(boto3.resource("s3"), boto3.resource("iam"))
    )
    comp_classifier = ComprehendClassifier(boto3.client("comprehend"))
    classifier_trained_waiter = ClassifierTrainedWaiter(
        comp_classifier.comprehend_client
    )
    training_labels = {"bug", "feature-request", "dynamodb", "s3"}

    print("Setting up storage and security resources needed for the demo.")
    comp_demo.setup()

    print("Getting training data from GitHub and uploading it to Amazon S3.")
    training_issues = comp_demo.get_training_issues(training_labels)
    comp_demo.upload_issue_data(training_issues, True)

    classifier_name = "doc-example-classifier"
    print(f"Creating document classifier {classifier_name}.")
    comp_classifier.create(
        classifier_name,
        "en",
        comp_demo.demo_resources.bucket.name,
        comp_demo.training_prefix,
        comp_demo.demo_resources.data_access_role.arn,
        ClassifierMode.multi_label,
    )
    print(
        f"Waiting until {classifier_name} is trained. This typically takes "
        f"30-40 minutes."
    )
    classifier_trained_waiter.wait(comp_classifier.classifier_arn)

    print(f"Classifier {classifier_name} is trained:")
    pprint(comp_classifier.describe())

    print("Getting input data from GitHub and uploading it to Amazon S3.")
    input_issues = comp_demo.get_input_issues(training_labels)
    comp_demo.upload_issue_data(input_issues)

    print("Starting classification job on input data.")
    job_info = comp_classifier.start_job(
```

```
        "issue_classification_job",
        comp_demo.demo_resources.bucket.name,
        comp_demo.input_prefix,
        comp_demo.input_format,
        comp_demo.demo_resources.bucket.name,
        comp_demo.output_prefix,
        comp_demo.demo_resources.data_access_role.arn,
    )
    print(f"Waiting for job {job_info['JobId']} to complete.")
    job_waiter = JobCompleteWaiter(comp_classifier.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = comp_classifier.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from Amazon S3: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = comp_demo.extract_job_output(job)
    print("Job output:")
    pprint(job_output)

    print("Reconciling job output with labels from GitHub:")
    reconciled_output = comp_demo.reconcile_job_output(input_issues, job_output)
    print(*reconciled_output, sep="\n")

    answer = input(f"Do you want to delete the classifier {classifier_name} (y/n)?
")
    if answer.lower() == "y":
        print(f"Deleting {classifier_name}.")
        comp_classifier.delete()

    print("Cleaning up resources created for the demo.")
    comp_demo.cleanup()

    print("Thanks for watching!")
    print("-" * 88)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateDocumentClassifier
  - DeleteDocumentClassifier
  - DescribeDocumentClassificationJob
  - DescribeDocumentClassifier
  - ListDocumentClassificationJobs
  - ListDocumentClassifiers
  - StartDocumentClassificationJob

AWS Config exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with AWS Config.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteConfigRule`

O código de exemplo a seguir mostra como usar `DeleteConfigRule`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def delete_config_rule(self, rule_name):
        """
        Delete the specified rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.config_client.delete_config_rule(ConfigRuleName=rule_name)
            logger.info("Deleted rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't delete rule %s.", rule_name)
            raise
```

- Para obter detalhes da API, consulte a DeleteConfigRule Referência da API AWS SDK for Python (Boto3).

`DescribeConfigRules`

O código de exemplo a seguir mostra como usar `DescribeConfigRules`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def describe_config_rule(self, rule_name):
        """
        Gets data for the specified rule.

        :param rule_name: The name of the rule to retrieve.
        :return: The rule data.
        """
        try:
            response = self.config_client.describe_config_rules(
                ConfigRuleNames=[rule_name]
            )
            rule = response["ConfigRules"]
            logger.info("Got data for rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't get data for rule %s.", rule_name)
            raise
        else:
            return rule
```

- Para obter detalhes da API, consulte a DescribeConfigRules Referência da API AWS SDK for Python (Boto3).

`PutConfigRule`

O código de exemplo a seguir mostra como usar `PutConfigRule`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def put_config_rule(self, rule_name):
        """
        Sets a configuration rule that prohibits making Amazon S3 buckets publicly
        readable.

        :param rule_name: The name to give the rule.
        """
        try:
            self.config_client.put_config_rule(
                ConfigRule={
                    "ConfigRuleName": rule_name,
                    "Description": "S3 Public Read Prohibited Bucket Rule",
```

```
                    "Scope": {
                        "ComplianceResourceTypes": [
                            "AWS::S3::Bucket",
                        ],
                    },
                    "Source": {
                        "Owner": "AWS",
                        "SourceIdentifier": "S3_BUCKET_PUBLIC_READ_PROHIBITED",
                    },
                    "InputParameters": "{}",
                    "ConfigRuleState": "ACTIVE",
                }
            )
            logger.info("Created configuration rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't create configuration rule %s.", rule_name)
            raise
```

- Para obter detalhes da API, consulte a PutConfigRule Referência da API AWS SDK for Python (Boto3).

AWS Control Tower exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with AWS Control Tower.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos

- Ações

Conceitos básicos

Hello AWS Control Tower

O exemplo de código a seguir mostra como começar a usar o AWS Control Tower.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
from typing import Any, List


def hello_controltower(controltower_client: Any) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Control Tower client
    and list all available baselines.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param controltower_client: A Boto3 AWS Control Tower Client object. This object
 wraps
                                the low-level AWS Control Tower service API.
    """
    print("Hello, AWS Control Tower! Let's list available baselines:\n")
    paginator = controltower_client.get_paginator("list_baselines")
    page_iterator = paginator.paginate()

    baseline_names: List[str] = []
    try:
        for page in page_iterator:
            for baseline in page["baselines"]:
                baseline_names.append(baseline["name"])

        print(f"{len(baseline_names)} baseline(s) retrieved.")
```

```
        for baseline_name in baseline_names:
            print(f"\t{baseline_name}")

    except controltower_client.exceptions.AccessDeniedException:
        print("Access denied. Please ensure you have the necessary permissions.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")


if __name__ == "__main__":
    hello_controltower(boto3.client("controltower"))
```

- Para obter detalhes da API, consulte a ListBaselines Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Liste as zonas de pouso.
- Liste, ative, obtenha, redefina e desative as linhas de base.
- Liste, ative, obtenha e desative controles.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando os AWS Control Tower recursos.

```
class ControlTowerScenario:
    IDENTITY_CENTER_BASELINE = "baseline/LN25R72TTG6IGPTQ"
    stack_name = ""
```

```
    def __init__(
        self, controltower_wrapper: ControlTowerWrapper, org_client: boto3.client
    ):
        """
        :param controltower_wrapper: An instance of the ControlTowerWrapper class.
        :param org_client: A Boto3 Organization client.
        """
        self.controltower_wrapper = controltower_wrapper
        self.org_client = org_client
        self.stack = None
        self.ou_id = None
        self.ou_arn = None
        self.account_id = None
        self.landing_zone_id = None
        self.use_landing_zone = False

    def run_scenario(self) -> None:
        print("-" * 88)
        print(
            "\tWelcome to the AWS Control Tower with ControlCatalog example
 scenario."
        )
        print("-" * 88)

        print(
            "This demo will walk you through working with AWS Control Tower for
 landing zones,\n"
            "managing baselines, and working with controls."
        )

        self.account_id = boto3.client("sts").get_caller_identity()["Account"]

        print(
            "Some demo operations require the use of a landing zone. "
            "\nYou can use an existing landing zone or opt out of these operations
 in the demo."
            "\nFor instructions on how to set up a landing zone, "
            "\nsee https://docs.aws.amazon.com/controltower/latest/userguide/
getting-started-from-console.html"
        )
        # List available landing zones
        landing_zones = self.controltower_wrapper.list_landing_zones()
        if landing_zones:
            print("\nAvailable Landing Zones:")
```

```
            for i, lz in enumerate(landing_zones, 1):
                print(f"{i} {lz['arn']}})")

            # Ask if user wants to use the first landing zone in the list
            if q.ask(
                f"Do you want to use the first landing zone in the list
({landing_zones[0]['arn']})? (y/n) ",
                q.is_yesno,
            ):
                self.use_landing_zone = True
                self.landing_zone_id = landing_zones[0]["arn"]
                print(f"Using landing zone ID: {self.landing_zone_id})")
                # Set up organization and get Sandbox OU ID.
                sandbox_ou_id = self.setup_organization()
                # Store the OU ID for use in the CloudFormation template.
                self.ou_id = sandbox_ou_id
            elif q.ask(
                f"Do you want to use a different existing Landing Zone for this
demo? (y/n) ",
                q.is_yesno,
            ):
                self.use_landing_zone = True
                self.landing_zone_id = q.ask("Enter landing zone id: ", q.non_empty)
                # Set up organization and get Sandbox OU ID.
                sandbox_ou_id = self.setup_organization()
                # Store the OU ID for use in the CloudFormation template.
                self.ou_id = sandbox_ou_id

        # List and Enable Baseline.
        print("\nManaging Baselines:")
        control_tower_baseline = None
        identity_center_baseline = None
        baselines = self.controltower_wrapper.list_baselines()
        print("\nListing available Baselines:")
        for baseline in baselines:
            if baseline["name"] == "AWSControlTowerBaseline":
                control_tower_baseline = baseline
            print(f"{baseline['name']}")

        if self.use_landing_zone:
            print("\nListing enabled baselines:")
            enabled_baselines = self.controltower_wrapper.list_enabled_baselines()
            for baseline in enabled_baselines:
```

```
                # If the Identity Center baseline is enabled, the identifier must be
used for other baselines.
                if self.IDENTITY_CENTER_BASELINE in baseline["baselineIdentifier"]:
                    identity_center_baseline = baseline
                print(f"{baseline['baselineIdentifier']}")

            if q.ask(
                f"Do you want to enable the Control Tower Baseline? (y/n) ",
                q.is_yesno,
            ):
                print("\nEnabling Control Tower Baseline.")
                ic_baseline_arn = (
                    identity_center_baseline["arn"]
                    if identity_center_baseline
                    else None
                )
                baseline_arn = self.controltower_wrapper.enable_baseline(
                    self.ou_arn, ic_baseline_arn, control_tower_baseline["arn"],
"5.0"
                )
                if baseline_arn:
                    print(f"Enabled baseline ARN: {baseline_arn}")
                else:
                    # Find the enabled baseline so we can reset it.
                    for enabled_baseline in enabled_baselines:
                        if (
                            enabled_baseline["baselineIdentifier"]
                            == control_tower_baseline["arn"]
                        ):
                            baseline_arn = enabled_baseline["arn"]
                    if baseline_arn:
                        print("No change, the selected baseline was already
enabled.")

                if baseline_arn and q.ask(
                    f"Do you want to reset the Control Tower Baseline? (y/n) ",
                    q.is_yesno,
                ):
                    print(f"\nResetting Control Tower Baseline. {baseline_arn}")
                    operation_id = self.controltower_wrapper.reset_enabled_baseline(
                        baseline_arn
                    )
                    print(f"\nReset baseline operation id {operation_id}.")
```

```
            if baseline_arn and q.ask(
                f"Do you want to disable the Control Tower Baseline? (y/n) ",
                q.is_yesno,
            ):
                print(f"Disabling baseline ARN: {baseline_arn}")
                operation_id = self.controltower_wrapper.disable_baseline(
                    baseline_arn
                )
                print(f"\nDisabled baseline operation id {operation_id}.")

                # Re-enable the baseline for the next step.
                print("\nEnabling Control Tower Baseline.")
                self.controltower_wrapper.enable_baseline(
                    self.ou_arn,
                    ic_baseline_arn,
                    control_tower_baseline["arn"],
                    "5.0",
                )

        # List and Enable Controls.
        print("\nManaging Controls:")
        controls = self.controltower_wrapper.list_controls()
        print("\nListing first 5 available Controls:")
        for i, control in enumerate(controls[:5], 1):
            print(f"{i}. {control['Name']} - {control['Arn']}")

        if self.use_landing_zone:
            target_ou = self.ou_arn
            enabled_controls = self.controltower_wrapper.list_enabled_controls(
                target_ou
            )
            print("\nListing enabled controls:")
            for i, control in enumerate(enabled_controls, 1):
                print(f"{i}. {control['controlIdentifier']}")

            # Enable first non-enabled control as an example.
            enabled_control_arns = [control["arn"] for control in enabled_controls]
            control_arn = next(
                control["Arn"]
                for control in controls
                if control["Arn"] not in enabled_control_arns
            )

            if control_arn and q.ask(
```

```
                f"Do you want to enable the control {control_arn}? (y/n) ",
                q.is_yesno,
            ):
                print(f"\nEnabling control: {control_arn}")
                operation_id = self.controltower_wrapper.enable_control(
                    control_arn, target_ou
                )

                if operation_id:
                    print(f"Enabled control with operation id {operation_id}")

            if control_arn and q.ask(
                f"Do you want to disable the control? (y/n) ",
                q.is_yesno,
            ):
                print("\nDisabling the control...")
                operation_id = self.controltower_wrapper.disable_control(
                    control_arn, target_ou
                )
                print(f"Disable operation ID: {operation_id}")

        print("\nThis concludes the example scenario.")

        print("Thanks for watching!")
        print("-" * 88)

    def setup_organization(self):
        """
        Checks if the current account is part of an organization and creates one if
needed.
        Also ensures a Sandbox OU exists and returns its ID.

        :return: The ID of the Sandbox OU
        """
        print("\nChecking organization status...")

        try:
            # Check if account is part of an organization
            org_response = self.org_client.describe_organization()
            org_id = org_response["Organization"]["Id"]
            print(f"Account is part of organization: {org_id}")

        except ClientError as error:
```

```
            if error.response["Error"]["Code"] ==
 "AWSOrganizationsNotInUseException":
                print("No organization found. Creating a new organization...")
                try:
                    create_response = self.org_client.create_organization(
                        FeatureSet="ALL"
                    )
                    org_id = create_response["Organization"]["Id"]
                    print(f"Created new organization: {org_id}")

                    # Wait for organization to be available.
                    waiter = self.org_client.get_waiter("organization_active")
                    waiter.wait(
                        Organization=org_id,
                        WaiterConfig={"Delay": 5, "MaxAttempts": 12},
                    )

                except ClientError as create_error:
                    logger.error(
                        "Couldn't create organization. Here's why: %s: %s",
                        create_error.response["Error"]["Code"],
                        create_error.response["Error"]["Message"],
                    )
                    raise
            else:
                logger.error(
                    "Couldn't describe organization. Here's why: %s: %s",
                    error.response["Error"]["Code"],
                    error.response["Error"]["Message"],
                )
                raise

        # Look for Sandbox OU.
        sandbox_ou_id = None
        paginator = self.org_client.get_paginator(
            "list_organizational_units_for_parent"
        )

        try:
            # Get root ID first.
            roots = self.org_client.list_roots()["Roots"]
            if not roots:
                raise ValueError("No root found in organization")
            root_id = roots[0]["Id"]
```

```
            # Search for existing Sandbox OU.
            print("Checking for Sandbox OU...")
            for page in paginator.paginate(ParentId=root_id):
                for ou in page["OrganizationalUnits"]:
                    if ou["Name"] == "Sandbox":
                        sandbox_ou_id = ou["Id"]
                        self.ou_arn = ou["Arn"]
                        print(f"Found existing Sandbox OU: {sandbox_ou_id}")
                        break
                if sandbox_ou_id:
                    break

            # Create Sandbox OU if it doesn't exist.
            if not sandbox_ou_id:
                print("Creating Sandbox OU...")
                create_ou_response = self.org_client.create_organizational_unit(
                    ParentId=root_id, Name="Sandbox"
                )
                sandbox_ou_id = create_ou_response["OrganizationalUnit"]["Id"]
                print(f"Created new Sandbox OU: {sandbox_ou_id}")

                # Wait for OU to be available.
                waiter = self.org_client.get_waiter("organizational_unit_active")
                waiter.wait(
                    OrganizationalUnitId=sandbox_ou_id,
                    WaiterConfig={"Delay": 5, "MaxAttempts": 12},
                )

        except ClientError as error:
            logger.error(
                "Couldn't set up Sandbox OU. Here's why: %s: %s",
                error.response["Error"]["Code"],
                error.response["Error"]["Message"],
            )
            raise

        return sandbox_ou_id


if __name__ == "__main__":
    try:
        org = boto3.client("organizations")
        control_tower_wrapper = ControlTowerWrapper.from_client()
```

```
        scenario = ControlTowerScenario(control_tower_wrapper, org)
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the scenario.")



class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_baselines(self):
        """
        Lists all baselines.

        :return: List of baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_baselines")
            baselines = []
            for page in paginator.paginate():
                baselines.extend(page["baselines"])
            return baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
```

```
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def enable_baseline(
        self,
        target_identifier: str,
        identity_center_baseline: str,
        baseline_identifier: str,
        baseline_version: str,
    ):
        """
        Enables a baseline for the specified target if it's not already enabled.

        :param target_identifier: The ARN of the target.
        :param baseline_identifier: The identifier of baseline to enable.
        :param identity_center_baseline: The identifier of identity center baseline
if it is enabled.
        :param baseline_version: The version of baseline to enable.
        :return: The enabled baseline ARN or None if already enabled.
        :raises ClientError: If enabling the baseline fails for reasons other than
it being already enabled.
        """
        try:
            # Only include parameters if identity_center_baseline is not empty
            parameters = []
            if identity_center_baseline:
                parameters = [
                    {
                        "key": "IdentityCenterEnabledBaselineArn",
                        "value": identity_center_baseline,
                    }
                ]

            response = self.controltower_client.enable_baseline(
```

```
                baselineIdentifier=baseline_identifier,
                baselineVersion=baseline_version,
                targetIdentifier=target_identifier,
                parameters=parameters,
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["arn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                if "already enabled" in err.response["Error"]["Message"]:
                    print("Baseline is already enabled for this target")
                else:
                    print(
                        "Unable to enable baseline due to validation exception: %s:
%s",
                        err.response["Error"]["Code"],
                        err.response["Error"]["Message"],
                    )
            logger.error(
                "Couldn't enable baseline. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            return None


    def list_controls(self):
        """
        Lists all controls in the Control Tower control catalog.

        :return: List of controls.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controlcatalog_client.get_paginator("list_controls")
            controls = []
```

```
            for page in paginator.paginate():
                controls.extend(page["Controls"])
            return controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def enable_control(self, control_arn: str, target_identifier: str):
        """
        Enables a control for a specified target.

        :param control_arn: The ARN of the control to enable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If enabling the control fails.
        """
        try:
            print(control_arn)
            print(target_identifier)
            response = self.controltower_client.enable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id
```

```
        except ClientError as err:
            if (
                err.response["Error"]["Code"] == "ValidationException"
                and "already enabled" in err.response["Error"]["Message"]
            ):
                logger.info("Control is already enabled for this target")
                return None
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with controls.")
                return None
            logger.error(
                "Couldn't enable control. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_control_operation(self, operation_id: str):
        """
        Gets the status of a control operation.

        :param operation_id: The ID of the control operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_control_operation(
                operationIdentifier=operation_id
            )
            return response["controlOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get control operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
```

```
                )
            raise


    def get_baseline_operation(self, operation_id: str):
        """
        Gets the status of a baseline operation.

        :param operation_id: The ID of the baseline operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_baseline_operation(
                operationIdentifier=operation_id
            )
            return response["baselineOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get baseline operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def disable_control(self, control_arn: str, target_identifier: str):
        """
        Disables a control for a specified target.

        :param control_arn: The ARN of the control to disable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If disabling the control fails.
        """
        try:
            response = self.controltower_client.disable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
```

```
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Control not found.")
            else:
                logger.error(
                    "Couldn't disable control. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def list_landing_zones(self):
        """
        Lists all landing zones.

        :return: List of landing zones.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_landing_zones")
            landing_zones = []
            for page in paginator.paginate():
                landing_zones.extend(page["landingZones"])
            return landing_zones

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list landing zones. Here's why: %s: %s",
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
            raise


    def list_enabled_baselines(self):
        """
        Lists all enabled baselines.

        :return: List of enabled baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_baselines")
            enabled_baselines = []
            for page in paginator.paginate():
                enabled_baselines.extend(page["enabledBaselines"])
            return enabled_baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't list enabled baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def reset_enabled_baseline(self, enabled_baseline_identifier: str):
        """
        Resets an enabled baseline for a specific target.

        :param enabled_baseline_identifier: The identifier of the enabled baseline
to reset.
        :return: The operation ID.
        :raises ClientError: If resetting the baseline fails.
        """
        try:
            response = self.controltower_client.reset_enabled_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
```

```
            )
            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't reset enabled baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def disable_baseline(self, enabled_baseline_identifier: str):
        """
        Disables a baseline for a specific target and waits for the operation to
complete.

        :param enabled_baseline_identifier: The identifier of the baseline to
disable.
        :return: The operation ID.
        :raises ClientError: If disabling the baseline fails.
        """
        try:
            response = self.controltower_client.disable_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
```

```
            return response["operationIdentifier"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                print(
                    f"Conflict disabling baseline: {err.response['Error']
['Message']}. Skipping disable step."
                )
                return None
            else:
                logger.error(
                    "Couldn't disable baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise


    def list_enabled_controls(self, target_identifier: str):
        """
        Lists all enabled controls for a specific target.

        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: List of enabled controls.
        :raises ClientError: If the listing operation fails.
        """
        enabled_controls = []
        try:
            paginator =
 self.controltower_client.get_paginator("list_enabled_controls")

            for page in paginator.paginate(targetIdentifier=target_identifier):
                enabled_controls.extend(page["enabledControls"])
            return enabled_controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
 permissions."
                )
                return enabled_controls
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
```

```
            in err.response["Error"]["Message"]
        ):
            logger.error("Control Tower must be enabled to work with controls.")
            return enabled_controls
        else:
            logger.error(
                "Couldn't list enabled controls. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
        raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Ações

`DisableBaseline`

O código de exemplo a seguir mostra como usar `DisableBaseline`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def disable_baseline(self, enabled_baseline_identifier: str):
        """
        Disables a baseline for a specific target and waits for the operation to
 complete.

        :param enabled_baseline_identifier: The identifier of the baseline to
 disable.
        :return: The operation ID.
        :raises ClientError: If disabling the baseline fails.
        """
        try:
            response = self.controltower_client.disable_baseline(
```

```
                enabledBaselineIdentifier=enabled_baseline_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["operationIdentifier"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                print(
                    f"Conflict disabling baseline: {err.response['Error']
['Message']}. Skipping disable step."
                )
                return None
            else:
                logger.error(
                    "Couldn't disable baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a DisableBaseline Referência da API AWS SDK for Python (Boto3).

`DisableControl`

O código de exemplo a seguir mostra como usar `DisableControl`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def disable_control(self, control_arn: str, target_identifier: str):
        """
        Disables a control for a specified target.

        :param control_arn: The ARN of the control to disable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If disabling the control fails.
        """
        try:
            response = self.controltower_client.disable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
```

```
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Control not found.")
            else:
                logger.error(
                    "Couldn't disable control. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DisableControl Referência da API AWS SDK for Python (Boto3).

`EnableBaseline`

O código de exemplo a seguir mostra como usar `EnableBaseline`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def enable_baseline(
        self,
        target_identifier: str,
        identity_center_baseline: str,
        baseline_identifier: str,
        baseline_version: str,
    ):
        """
        Enables a baseline for the specified target if it's not already enabled.

        :param target_identifier: The ARN of the target.
        :param baseline_identifier: The identifier of baseline to enable.
        :param identity_center_baseline: The identifier of identity center baseline
 if it is enabled.
        :param baseline_version: The version of baseline to enable.
        :return: The enabled baseline ARN or None if already enabled.
        :raises ClientError: If enabling the baseline fails for reasons other than
 it being already enabled.
        """
        try:
            # Only include parameters if identity_center_baseline is not empty
            parameters = []
            if identity_center_baseline:
                parameters = [
```

```
                    {
                        "key": "IdentityCenterEnabledBaselineArn",
                        "value": identity_center_baseline,
                    }
                ]

            response = self.controltower_client.enable_baseline(
                baselineIdentifier=baseline_identifier,
                baselineVersion=baseline_version,
                targetIdentifier=target_identifier,
                parameters=parameters,
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["arn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                if "already enabled" in err.response["Error"]["Message"]:
                    print("Baseline is already enabled for this target")
                else:
                    print(
                        "Unable to enable baseline due to validation exception: %s:
%s",
                        err.response["Error"]["Code"],
                        err.response["Error"]["Message"],
                    )
            logger.error(
                "Couldn't enable baseline. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            return None
```

- Para obter detalhes da API, consulte a EnableBaseline Referência da API AWS SDK for Python (Boto3).

`EnableControl`

O código de exemplo a seguir mostra como usar `EnableControl`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def enable_control(self, control_arn: str, target_identifier: str):
        """
        Enables a control for a specified target.

        :param control_arn: The ARN of the control to enable.
```

```
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If enabling the control fails.
        """
        try:
            print(control_arn)
            print(target_identifier)
            response = self.controltower_client.enable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id

        except ClientError as err:
            if (
                err.response["Error"]["Code"] == "ValidationException"
                and "already enabled" in err.response["Error"]["Message"]
            ):
                logger.info("Control is already enabled for this target")
                return None
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with controls.")
                return None
            logger.error(
                "Couldn't enable control. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a EnableControl Referência da API AWS SDK for Python (Boto3).

GetBaselineOperation

O código de exemplo a seguir mostra como usar `GetBaselineOperation`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def get_baseline_operation(self, operation_id: str):
        """
        Gets the status of a baseline operation.

        :param operation_id: The ID of the baseline operation.
```

```
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_baseline_operation(
                operationIdentifier=operation_id
            )
            return response["baselineOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get baseline operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a GetBaselineOperation Referência da API AWS SDK for Python (Boto3).

`GetControlOperation`

O código de exemplo a seguir mostra como usar `GetControlOperation`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""
```

```
    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def get_control_operation(self, operation_id: str):
        """
        Gets the status of a control operation.

        :param operation_id: The ID of the control operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_control_operation(
                operationIdentifier=operation_id
            )
            return response["controlOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get control operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a GetControlOperation Referência da API AWS SDK for Python (Boto3).

`ListBaselines`

O código de exemplo a seguir mostra como usar `ListBaselines`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_baselines(self):
        """
        Lists all baselines.

        :return: List of baselines.
```

```
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_baselines")
            baselines = []
            for page in paginator.paginate():
                baselines.extend(page["baselines"])
            return baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListBaselines Referência da API AWS SDK for Python (Boto3).

`ListEnabledBaselines`

O código de exemplo a seguir mostra como usar `ListEnabledBaselines`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_enabled_baselines(self):
        """
        Lists all enabled baselines.

        :return: List of enabled baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
 self.controltower_client.get_paginator("list_enabled_baselines")
            enabled_baselines = []
            for page in paginator.paginate():
                enabled_baselines.extend(page["enabledBaselines"])
            return enabled_baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't list enabled baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
```

```
                )
            raise
```

- Para obter detalhes da API, consulte a ListEnabledBaselines Referência da API AWS SDK for Python (Boto3).

`ListEnabledControls`

O código de exemplo a seguir mostra como usar `ListEnabledControls`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)
```

```
    def list_enabled_controls(self, target_identifier: str):
        """
        Lists all enabled controls for a specific target.

        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: List of enabled controls.
        :raises ClientError: If the listing operation fails.
        """
        enabled_controls = []
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_controls")

            for page in paginator.paginate(targetIdentifier=target_identifier):
                enabled_controls.extend(page["enabledControls"])
            return enabled_controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
                return enabled_controls
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with controls.")
                return enabled_controls
            else:
                logger.error(
                    "Couldn't list enabled controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListEnabledControls Referência da API AWS SDK for Python (Boto3).

`ListLandingZones`

O código de exemplo a seguir mostra como usar `ListLandingZones`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_landing_zones(self):
        """
        Lists all landing zones.

        :return: List of landing zones.
```

```
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_landing_zones")
            landing_zones = []
            for page in paginator.paginate():
                landing_zones.extend(page["landingZones"])
            return landing_zones

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list landing zones. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListLandingZones Referência da API AWS SDK for Python (Boto3).

`ResetEnabledBaseline`

O código de exemplo a seguir mostra como usar `ResetEnabledBaseline`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def reset_enabled_baseline(self, enabled_baseline_identifier: str):
        """
        Resets an enabled baseline for a specific target.

        :param enabled_baseline_identifier: The identifier of the enabled baseline
 to reset.
        :return: The operation ID.
        :raises ClientError: If resetting the baseline fails.
        """
        try:
            response = self.controltower_client.reset_enabled_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )
            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't reset enabled baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ResetEnabledBaseline Referência da API AWS SDK for Python (Boto3).

Exemplos do Firehose usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with Firehose.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`PutRecord`

O código de exemplo a seguir mostra como usar `PutRecord`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.
```

```
        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose delivery
 stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise
```

- Para obter detalhes da API, consulte a PutRecord Referência da API AWS SDK for Python (Boto3).

`PutRecordBatch`

O código de exemplo a seguir mostra como usar `PutRecordBatch`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
```

```
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
```

```
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records. Error:
 {e}")
```

- Para obter detalhes da API, consulte a PutRecordBatch Referência da API AWS SDK for Python (Boto3).

Cenários

Inserção de registros no Firehose

O exemplo de código a seguir mostra como usar o Firehose para processar registros individuais e em lote.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este script insere registros individuais e em lote no Firehose.

```
import json
import logging
import random
from datetime import datetime, timedelta

import backoff
import boto3

from config import get_config


def load_sample_data(path: str) -> dict:
    """
    Load sample data from a JSON file.
```

```
    Args:
        path (str): The file path to the JSON file containing sample data.

    Returns:
        dict: The loaded sample data as a dictionary.
    """
    with open(path, "r") as f:
        return json.load(f)


# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
```

```
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
```

```
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records. Error:
{e}")


    def get_metric_statistics(
        self,
        metric_name: str,
        start_time: datetime,
        end_time: datetime,
        period: int,
        statistics: list = ["Sum"],
    ) -> list:
        """
        Retrieve metric statistics from CloudWatch.

        Args:
            metric_name (str): The name of the metric.
            start_time (datetime): The start time for the metric statistics.
            end_time (datetime): The end time for the metric statistics.
            period (int): The granularity, in seconds, of the returned data points.
            statistics (list): A list of statistics to retrieve. Default is ['Sum'].

        Returns:
            list: List of datapoints containing the metric statistics.
        """
        response = self.cloudwatch.get_metric_statistics(
            Namespace="AWS/Firehose",
            MetricName=metric_name,
            Dimensions=[
                {"Name": "DeliveryStreamName", "Value": self.delivery_stream_name},
            ],
            StartTime=start_time,
            EndTime=end_time,
            Period=period,
            Statistics=statistics,
        )
        return response["Datapoints"]

    def monitor_metrics(self):
```

```
        """
        Monitor Firehose metrics for the last 5 minutes.

        This method retrieves and logs the 'IncomingBytes', 'IncomingRecords', and
'FailedPutCount' metrics
        from CloudWatch for the last 5 minutes.
        """
        end_time = datetime.utcnow()
        start_time = end_time - timedelta(minutes=10)
        period = int((end_time - start_time).total_seconds())

        metrics = {
            "IncomingBytes": self.get_metric_statistics(
                "IncomingBytes", start_time, end_time, period
            ),
            "IncomingRecords": self.get_metric_statistics(
                "IncomingRecords", start_time, end_time, period
            ),
            "FailedPutCount": self.get_metric_statistics(
                "FailedPutCount", start_time, end_time, period
            ),
        }

        for metric, datapoints in metrics.items():
            if datapoints:
                total_sum = sum(datapoint["Sum"] for datapoint in datapoints)
                if metric == "IncomingBytes":
                    logger.info(
                        f"{metric}: {round(total_sum)} ({total_sum / (1024 *
1024):.2f} MB)"
                    )
                else:
                    logger.info(f"{metric}: {round(total_sum)}")
            else:
                logger.info(f"No data found for {metric} over the last 5 minutes")


    def _create_record_entry(self, record: dict) -> dict:
        """
        Create a record entry for Firehose.

        Args:
            record (dict): The data record to be sent.
```

```
        Returns:
            dict: The record entry formatted for Firehose.

        Raises:
            Exception: If a simulated network error occurs.
        """
        if random.random() < 0.2:
            raise Exception("Simulated network error")
        elif random.random() < 0.1:
            return {"Data": '{"malformed": "data"'}
        else:
            return {"Data": json.dumps(record)}

    def _log_response(self, response: dict, entry: dict):
        """
        Log the response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record API call.
            entry (dict): The record entry that was sent.
        """
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            logger.info(f"Sent record: {entry}")
        else:
            logger.info(f"Fail record: {entry}")

    def _log_batch_response(self, response: dict, batch_size: int):
        """
        Log the batch response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record_batch API
call.
            batch_size (int): The number of records in the batch.
        """
        if response.get("FailedPutCount", 0) > 0:
            logger.info(
                f'Failed to send {response["FailedPutCount"]} records in batch of
{batch_size}'
            )
        else:
            logger.info(f"Successfully sent batch of {batch_size} records")
```

```
if __name__ == "__main__":
    config = get_config()
    data = load_sample_data(config.sample_data_file)
    client = FirehoseClient(config)

    # Process the first 100 sample network records
    for record in data[:100]:
        try:
            client.put_record(record)
        except Exception as e:
            logger.info(f"Put record failed after retries and backoff: {e}")
    client.monitor_metrics()

    # Process remaining records using the batch method
    try:
        client.put_record_batch(data[100:])
    except Exception as e:
        logger.info(f"Put record batch failed after retries and backoff: {e}")
    client.monitor_metrics()
```

Este arquivo contém a configuração do script acima.

```
class Config:
    def __init__(self):
        self.delivery_stream_name = "ENTER YOUR DELIVERY STREAM NAME HERE"
        self.region = "us-east-1"
        self.sample_data_file = (
            "../../../../../scenarios/features/firehose/resources/
sample_records.json"
        )


def get_config():
    return Config()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - PutRecord
  - PutRecordBatch

Exemplos de Device Farm usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with Device Farm.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Executar testes de navegador e capturar tela

O exemplo de código a seguir mostra como executar testes de navegador com o Device Farm e fazer capturas de tela.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o PyTest Selenium para navegar até sites específicos, fazer capturas de tela e comparar o conteúdo real do site com o conteúdo esperado.

```
import datetime
import os
import subprocess
import boto3
import pytest
from selenium import webdriver
from selenium.webdriver import DesiredCapabilities
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
```

```
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait


def get_git_hash():
    """
    Get the short Git hash of the current commit of the repository
    """
    try:
        return (
            subprocess.check_output(["git", "rev-parse", "--short", "HEAD"])
            .decode("utf-8")
            .strip()
        )
    except:
        return "norepo"


class TestHelloSuite:
    """
    Our test suite.

    This style of test suite allows us to use setup_method and teardown_method.

    """

    def save_screenshot(self, name):
        self.driver.save_screenshot(os.path.join(self.screenshot_path, name))

    def setup_method(self, method):
        """
        Set up a test.

        This makes sure that the session for an individual test is ready.

        The AWS credentials are read from the default ~/.aws/credentials or from the
        command line by setting the AWS_ACCESS_KEY_ID and AWS_SECRET_KEY environment
        variables.

        The project Amazon Resource Name (ARN) is determined by the PROJECT_ARN
        environment variable.
        """
        devicefarm_client = boto3.client("devicefarm")
        project_arn = os.environ.get("PROJECT_ARN", None)
```

```
        if project_arn is None:
            raise ValueError("Must set PROJECT_ARN")
        # Request a driver hub URL for the Selenium client
        testgrid_url_response = devicefarm_client.create_test_grid_url(
            projectArn=project_arn, expiresInSeconds=300
        )

        # We want a directory to save our files into. We're going to make a
directory
        # in the current directory that holds our results.
        self.screenshot_path = os.path.join(
            ".", "results", get_git_hash() + "-" +
(datetime.date.today().isoformat())
        )
        if not os.path.exists(self.screenshot_path):
            os.makedirs(self.screenshot_path, exist_ok=True)

        # We want a Firefox instance on Windows
        desired_cap = DesiredCapabilities.FIREFOX
        desired_cap["platform"] = "windows"
        desired_cap["BrowserVersion"] = "latest"

        # Configure the webdriver with the appropriate remote endpoint.
        self.driver = webdriver.Remote(testgrid_url_response["url"], desired_cap)

        #
        # Auto-Tagging
        #

        # In order to get the Session ARN, we need to look up the session by the
        # Project ARN and session ID (from the driver).
        testgrid_session_arn_response = devicefarm_client.get_test_grid_session(
            projectArn=project_arn, sessionId=self.driver.session_id
        )

        # Save the session's ARN so we can tag the session.
        self.session_arn = testgrid_session_arn_response["testGridSession"]["arn"]

        # In order to tag it, we're going to use the resourcegroupstaggingapi client
to
        # add a tag to the session ARN that we just got.
        tag_client = boto3.client("resourcegroupstaggingapi")
        tag_client.tag_resources(
            ResourceARNList=[self.session_arn],
```

```
            Tags={"TestSuite": f"testsuite {method.__name__}", "GitId":
get_git_hash()},
        )

    def teardown_method(self, method):
        """
        Clean up resources used by each method.
        """
        # End the Selenium session so we're off the clock.
        self.driver.quit()

    @pytest.mark.parametrize(
        "query,leading",
        [
            pytest.param(
                "Seattle",
                "Seattle (/si#ætəl/ (listen) see-AT-əl) is a seaport city on the
West Coast of the United States.",
            ),
            pytest.param(
                "Selenium",
                "Selenium is a chemical element with the symbol Se and atomic number
34.",
            ),
            pytest.param(
                "Amazon Locker",
                "Amazon Locker is a self-service package delivery service offered by
online retailer Amazon.",
            ),
            pytest.param(
                "Kootenai Falls",
                "Kootenai Falls is a waterfall on the Kootenay River located in
Lincoln County, Montana, just off U.S. Route 2.",
            ),
            pytest.param(
                "Dorayaki",
                "Dorayaki (####, ####, ####, ####) is a type of Japanese
confection.",
            ),
            pytest.param("Robot Face", "<|°_°|> (also known as Robot Face or
Robot)"),
        ],
    )
    def test_first_paragraph_text(self, query, leading):
```

```
        """
        This test looks at the first paragraph of a page on Wikipedia, comparing it
to
        a known leading sentence.

        If the leading sentence matches, the test passes. A screenshot is taken
before
        the final assertion is made, letting us debug if something isn't right.
        """
        # Open the main page of Wikipedia
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
        # Find the search box, enter a query, and press enter
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # Wait for the search box to go stale -- This means we've navigated fully.
        WebDriverWait(self.driver, 5).until(
            expected_conditions.staleness_of(search_input)
        )
        # Get the leading paragraph of the article.
        lead = leading.lower()
        # Find the element...
        lead_para = self.driver.find_element(
            By.XPATH, "//div[@class='mw-parser-output']//p[not(@class)]"
        )
        # ... and copy out its text.
        our_text = lead_para.text.lower()
        our_text = our_text[: len(lead)]
        # Take a screenshot and compare the strings.
        self.save_screenshot(f"leadingpara_{query}.png")
        assert our_text.startswith(lead)

    @pytest.mark.parametrize(
        "query,expected",
        [
            pytest.param("Automation Testing", "Test Automation"),
            pytest.param("DevOps", "DevOps"),
            pytest.param("Jackdaws Love My Big Sphinx Of Quartz", "Pangram"),
            pytest.param("EarthBound", "EarthBound"),
            pytest.param("Covered Bridges Today", "Covered Bridges Today"),
            pytest.param("Kurt Godel", "Kurt Gödel"),
            pytest.param("N//ng language", "N#ng language"),
            pytest.param(
```

```
                "Who the Frick Is Jackson Pollock?", "Who the $&% Is Jackson
Pollock?"
            ),
        ],
    )
    def test_redirect_titles(self, query, expected):
        """
        A test comparing pages we expect to (or not to) redirect on Wikipedia.

        This test checks to see that the page ("query") redirects (or doesn't) to
the
        "expected" page title. Several of these are common synonyms ("Jackdaws...")
        while others are because of characters untypable by most keyboards ("N#ng
language")

        A screenshot is taken just before the final assertion is made to aid in
        debugging and verification.
        """
        # Open the main page of Wikipedia
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
        # Find the search box, enter some text into it, and send an enter key.
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # wait until the page has rolled over -- once the search input handle is
stale,
        # the browser has navigated.
        WebDriverWait(self.driver, 5).until(
            expected_conditions.staleness_of(search_input)
        )
        # Get the first heading & take a screenshot
        our_text = self.driver.find_element(By.ID, "firstHeading").text.lower()
        self.save_screenshot(f"redirect_{query}.png")
        # did it match?
        assert our_text == expected.lower()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateTestGridUrl

- GetTestGridSession

Fazer upload e testar pacotes de dispositivos

O exemplo de código a seguir mostra como fazer upload e testar pacotes de dispositivos móveis com o Device Farm.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de aplicativos Android compilados e pacotes de teste para o Device Farm, inicie um teste, aguarde a conclusão do teste e relate os resultados.

```
import boto3
import os
import requests
import string
import random
import datetime
import time

# Update this dict with your own values before you run the example:
config = {
    # This is our app under test.
    "appFilePath": "app-debug.apk",
    "projectArn": "arn:aws:devicefarm:us-west-2:111222333444:project:581f5703-
e040-4ac9-b7ae-0ba007bfb8e6",
    # Since we care about the most popular devices, we'll use a curated pool.
    "testSpecArn": "arn:aws:devicefarm:us-west-2::upload:20fcf771-eae3-4137-
aa76-92e17fb3131b",
    "poolArn": "arn:aws:devicefarm:us-
west-2::devicepool:4a869d91-6f17-491f-9a95-0a601aee2406",
    "namePrefix": "MyAppTest",
    # This is our test package. This tutorial won't go into how to make these.
    "testPackage": "tests.zip",
}
```

```
client = boto3.client("devicefarm")

unique = (
    config["namePrefix"]
    + "-"
    + (datetime.date.today().isoformat())
    + ("".join(random.sample(string.ascii_letters, 8)))
)

print(
    f"The unique identifier for this run is '{unique}'. All uploads will be prefixed
 "
    f"with this."
)


def upload_df_file(filename, type_, mime="application/octet-stream"):
    upload_response = client.create_upload(
        projectArn=config["projectArn"],
        name=unique + "_" + os.path.basename(filename),
        type=type_,
        contentType=mime,
    )
    upload_arn = upload_response["upload"]["arn"]
    # Extract the URL of the upload and use Requests to upload it.
    upload_url = upload_response["upload"]["url"]
    with open(filename, "rb") as file_stream:
        print(
            f"Uploading {filename} to Device Farm as "
            f"{upload_response['upload']['name']}... ",
            end="",
        )
        put_req = requests.put(
            upload_url, data=file_stream, headers={"content-type": mime}
        )
        print(" done")
        if not put_req.ok:
            raise Exception(f"Couldn't upload. Requests says: {put_req.reason}")
    started = datetime.datetime.now()
    while True:
        print(
            f"Upload of {filename} in state {upload_response['upload']['status']} "
            f"after " + str(datetime.datetime.now() - started)
```

```
        )
        if upload_response["upload"]["status"] == "FAILED":
            raise Exception(
                f"The upload failed processing. Device Farm says the reason is: \n"
                f"{+upload_response['upload']['message']}"
            )
        if upload_response["upload"]["status"] == "SUCCEEDED":
            break
        time.sleep(5)
        upload_response = client.get_upload(arn=upload_arn)
    print("")
    return upload_arn


our_upload_arn = upload_df_file(config["appFilePath"], "ANDROID_APP")
our_test_package_arn = upload_df_file(
    config["testPackage"], "APPIUM_PYTHON_TEST_PACKAGE"
)
print(our_upload_arn, our_test_package_arn)

response = client.schedule_run(
    projectArn=config["projectArn"],
    appArn=our_upload_arn,
    devicePoolArn=config["poolArn"],
    name=unique,
    test={
        "type": "APPIUM_PYTHON",
        "testSpecArn": config["testSpecArn"],
        "testPackageArn": our_test_package_arn,
    },
)
run_arn = response["run"]["arn"]
start_time = datetime.datetime.now()
print(f"Run {unique} is scheduled as arn {run_arn} ")

state = "UNKNOWN"
try:
    while True:
        response = client.get_run(arn=run_arn)
        state = response["run"]["status"]
        if state == "COMPLETED" or state == "ERRORED":
            break
        else:
            print(
```

```
                f" Run {unique} in state {state}, total "
                f"time {datetime.datetime.now() - start_time}"
            )
            time.sleep(10)
except:
    client.stop_run(arn=run_arn)
    exit(1)

print(f"Tests finished in state {state} after {datetime.datetime.now() -
 start_time}")
# Pull all the logs.
jobs_response = client.list_jobs(arn=run_arn)
# Save the output somewhere, using the unique value.
save_path = os.path.join(os.getcwd(), "results", unique)
os.mkdir(save_path)
# Save the last run information.
for job in jobs_response["jobs"]:
    job_name = job["name"]
    os.makedirs(os.path.join(save_path, job_name), exist_ok=True)
    # Get each suite within the job.
    suites = client.list_suites(arn=job["arn"])["suites"]
    for suite in suites:
        for test in client.list_tests(arn=suite["arn"])["tests"]:
            # Get the artifacts.
            for artifact_type in ["FILE", "SCREENSHOT", "LOG"]:
                artifacts = client.list_artifacts(type=artifact_type,
 arn=test["arn"])[
                    "artifacts"
                ]
                for artifact in artifacts:
                    # Replace `:` because it has a special meaning in Windows &
 macOS.
                    path_to = os.path.join(
                        save_path,
                        job_name,
                        suite["name"],
                        test["name"].replace(":", "_"),
                    )
                    os.makedirs(path_to, exist_ok=True)
                    filename = (
                        artifact["type"]
                        + "_"
                        + artifact["name"]
                        + "."
```

```
                        + artifact["extension"]
                    )
                    artifact_save_path = os.path.join(path_to, filename)
                    print(f"Downloading {artifact_save_path}")
                    with open(artifact_save_path, "wb") as fn:
                        with requests.get(
                            artifact["url"], allow_redirects=True
                        ) as request:
                            fn.write(request.content)
print("Finished")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateUpload
  - GetRun
  - GetUpload
  - ListArtifacts
  - ListJobs
  - ListSuites
  - ListTests
  - ScheduleRun
  - StopRun

Exemplos do Amazon DocumentDB usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Python.

```
import json

def lambda_handler(event, context):
    for record in event.get('events', []):
        log_document_db_event(record)
    return 'OK'

def log_document_db_event(record):
    event_data = record.get('event', {})
    operation_type = event_data.get('operationType', 'Unknown')
    db = event_data.get('ns', {}).get('db', 'Unknown')
    collection = event_data.get('ns', {}).get('coll', 'Unknown')
    full_document = event_data.get('fullDocument', {})

    print(f"Operation type: {operation_type}")
    print(f"db: {db}")
    print(f"collection: {collection}")
    print("Full document:", json.dumps(full_document, indent=2))
```

Exemplos de DynamoDB usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3

# Create a DynamoDB client using the default credentials and region
dynamodb = boto3.client("dynamodb")
```

```
# Initialize a paginator for the list_tables operation
paginator = dynamodb.get_paginator("list_tables")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(Limit=10)

# List the tables in the current AWS account
print("Here are the DynamoDB tables in your account:")

# Use pagination to list all tables
table_names = []

for page in page_iterator:
    for table_name in page.get("TableNames", []):
        print(f"- {table_name}")
        table_names.append(table_name)

if not table_names:
    print("You don't have any DynamoDB tables in your account.")
else:
    print(f"\nFound {len(table_names)} tables.")
```

- Para obter detalhes da API, consulte a ListTables Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que encapsule uma tabela do DynamoDB.

```
from decimal import Decimal
from io import BytesIO
import json
import logging
import os
from pprint import pprint
import requests
from zipfile import ZipFile
import boto3
from boto3.dynamodb.conditions import Key
from botocore.exceptions import ClientError
from question import Question

logger = logging.getLogger(__name__)

class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
```

```
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            table.load()
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                exists = False
            else:
                logger.error(
                    "Couldn't check for existence of %s. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            self.table = table
        return exists


    def create_table(self, table_name):
```

```
        """
        Creates an Amazon DynamoDB table that can be used to store movie data.
        The table uses the release year of the movie as the partition key and the
        title as the sort key.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
        try:
            self.table = self.dyn_resource.create_table(
                TableName=table_name,
                KeySchema=[
                    {"AttributeName": "year", "KeyType": "HASH"},  # Partition key
                    {"AttributeName": "title", "KeyType": "RANGE"},  # Sort key
                ],
                AttributeDefinitions=[
                    {"AttributeName": "year", "AttributeType": "N"},
                    {"AttributeName": "title", "AttributeType": "S"},
                ],
                BillingMode='PAY_PER_REQUEST',
            )
            self.table.wait_until_exists()
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.table


    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
```

```
            tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tables


    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at least
                       the keys required by the schema that was specified when the
                       table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
```

```
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    "year": year,
                    "title": title,
                    "info": {"plot": plot, "rating": Decimal(str(rating))},
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Item"]
```

```
    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to the give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={":r": Decimal(str(rating)), ":p": plot},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]


    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
        :return: The list of movies that were released in the specified year.
        """
        try:
            response = self.table.query(KeyConditionExpression=Key("year").eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
```

```
                year,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Items"]


    def scan_movies(self, year_range):
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            "FilterExpression": Key("year").between(
                year_range["first"], year_range["second"]
            ),
            "ProjectionExpression": "#yr, title, info.rating",
            "ExpressionAttributeNames": {"#yr": "year"},
        }
        try:
            done = False
            start_key = None
            while not done:
                if start_key:
                    scan_kwargs["ExclusiveStartKey"] = start_key
                response = self.table.scan(**scan_kwargs)
                movies.extend(response.get("Items", []))
                start_key = response.get("LastEvaluatedKey", None)
                done = start_key is None
        except ClientError as err:
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

        return movies
```

```
    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
def get_sample_movie_data(movie_file_name):
```

```
    """
    Gets sample movie data, either from a local file or by first downloading it from
    the Amazon DynamoDB developer guide.

    :param movie_file_name: The local file name where the movie data is stored in
 JSON format.
    :return: The movie data as a dict.
    """
    if not os.path.isfile(movie_file_name):
        print(f"Downloading {movie_file_name}...")
        movie_content = requests.get(
            "https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
samples/moviedata.zip"
        )
        movie_zip = ZipFile(BytesIO(movie_content.content))
        movie_zip.extractall()

    try:
        with open(movie_file_name) as movie_file:
            movie_data = json.load(movie_file, parse_float=Decimal)
    except FileNotFoundError:
        print(
            f"File {movie_file_name} not found. You must first download the file to
 "
            "run this demo. See the README for instructions."
        )
        raise
    else:
        # The sample file lists over 4000 movies, return only the first 250.
        return movie_data[:250]
```

Execute um cenário interativo para criar a tabela e executar ações nela.

```
def run_scenario(table_name, movie_file_name, dyn_resource):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB getting started demo.")
    print("-" * 88)
```

```
    movies = Movies(dyn_resource)
    movies_exists = movies.exists(table_name)
    if not movies_exists:
        print(f"\nCreating table {table_name}...")
        movies.create_table(table_name)
        print(f"\nCreated table {movies.table.name}.")

    my_movie = Question.ask_questions(
        [
            Question(
                "title", "Enter the title of a movie you want to add to the table: "
            ),
            Question("year", "What year was it released? ", Question.is_int),
            Question(
                "rating",
                "On a scale of 1 - 10, how do you rate it? ",
                Question.is_float,
                Question.in_range(1, 10),
            ),
            Question("plot", "Summarize the plot for me: "),
        ]
    )
    movies.add_movie(**my_movie)
    print(f"\nAdded '{my_movie['title']}' to '{movies.table.name}'.")
    print("-" * 88)

    movie_update = Question.ask_questions(
        [
            Question(
                "rating",
                f"\nLet's update your movie.\nYou rated it {my_movie['rating']},
what new "
                f"rating would you give it? ",
                Question.is_float,
                Question.in_range(1, 10),
            ),
            Question(
                "plot",
                f"You summarized the plot as '{my_movie['plot']}'.\nWhat would you
say now? ",
            ),
        ]
    )
    my_movie.update(movie_update)
```

```
    updated = movies.update_movie(**my_movie)
    print(f"\nUpdated '{my_movie['title']}' with new attributes:")
    pprint(updated)
    print("-" * 88)

    if not movies_exists:
        movie_data = get_sample_movie_data(movie_file_name)
        print(f"\nReading data from '{movie_file_name}' into your table.")
        movies.write_batch(movie_data)
        print(f"\nWrote {len(movie_data)} movies into {movies.table.name}.")
    print("-" * 88)

    title = "The Lord of the Rings: The Fellowship of the Ring"
    if Question.ask_question(
        f"Let's move on...do you want to get info about '{title}'? (y/n) ",
        Question.is_yesno,
    ):
        movie = movies.get_movie(title, 2001)
        print("\nHere's what I found:")
        pprint(movie)
    print("-" * 88)

    ask_for_year = True
    while ask_for_year:
        release_year = Question.ask_question(
            f"\nLet's get a list of movies released in a given year. Enter a year
between "
            f"1972 and 2018: ",
            Question.is_int,
            Question.in_range(1972, 2018),
        )
        releases = movies.query_movies(release_year)
        if releases:
            print(f"There were {len(releases)} movies released in {release_year}:")
            for release in releases:
                print(f"\t{release['title']}")
            ask_for_year = False
        else:
            print(f"I don't know about any movies released in {release_year}!")
            ask_for_year = Question.ask_question(
                "Try another year? (y/n) ", Question.is_yesno
            )
    print("-" * 88)
```

```
    years = Question.ask_questions(
        [
            Question(
                "first",
                f"\nNow let's scan for movies released in a range of years. Enter a
year: ",
                Question.is_int,
                Question.in_range(1972, 2018),
            ),
            Question(
                "second",
                "Now enter another year: ",
                Question.is_int,
                Question.in_range(1972, 2018),
            ),
        ]
    )
    releases = movies.scan_movies(years)
    if releases:
        count = Question.ask_question(
            f"\nFound {len(releases)} movies. How many do you want to see? ",
            Question.is_int,
            Question.in_range(1, len(releases)),
        )
        print(f"\nHere are your {count} movies:\n")
        pprint(releases[:count])
    else:
        print(
            f"I don't know about any movies released between {years['first']} "
            f"and {years['second']}."
        )
    print("-" * 88)

    if Question.ask_question(
        f"\nLet's remove your movie from the table. Do you want to remove "
        f"'{my_movie['title']}'? (y/n)",
        Question.is_yesno,
    ):
        movies.delete_movie(my_movie["title"], my_movie["year"])
        print(f"\nRemoved '{my_movie['title']}' from the table.")
    print("-" * 88)

    if Question.ask_question(f"\nDelete the table? (y/n) ", Question.is_yesno):
        movies.delete_table()
```

```
        print(f"Deleted {table_name}.")
    else:
        print(
            "Don't forget to delete the table when you're done or you might incur "
            "charges on your account."
        )

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        run_scenario(
            "doc-example-table-movies", "moviedata.json", boto3.resource("dynamodb")
        )
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

Esse cenário usa a classe auxiliar a seguir para fazer perguntas em um prompt de comando.

```
class Question:
    """
    A helper class to ask questions at a command prompt and validate and convert
    the answers.
    """

    def __init__(self, key, question, *validators):
        """
        :param key: The key that is used for storing the answer in a dict, when
                    multiple questions are asked in a set.
        :param question: The question to ask.
        :param validators: The answer is passed through the list of validators until
                           one fails or they all pass. Validators may also convert
 the
                           answer to another form, such as from a str to an int.
        """
        self.key = key
        self.question = question
        self.validators = Question.non_empty, *validators

    @staticmethod
```

```
    def ask_questions(questions):
        """
        Asks a set of questions and stores the answers in a dict.

        :param questions: The list of questions to ask.
        :return: A dict of answers.
        """
        answers = {}
        for question in questions:
            answers[question.key] = Question.ask_question(
                question.question, *question.validators
            )
        return answers

    @staticmethod
    def ask_question(question, *validators):
        """
        Asks a single question and validates it against a list of validators.
        When an answer fails validation, the complaint is printed and the question
        is asked again.

        :param question: The question to ask.
        :param validators: The list of validators that the answer must pass.
        :return: The answer, converted to its final form by the validators.
        """
        answer = None
        while answer is None:
            answer = input(question)
            for validator in validators:
                answer, complaint = validator(answer)
                if answer is None:
                    print(complaint)
                    break
        return answer

    @staticmethod
    def non_empty(answer):
        """
        Validates that the answer is not empty.
        :return: The non-empty answer, or None.
        """
        return answer if answer != "" else None, "I need an answer. Please?"

    @staticmethod
```

```
    def is_yesno(answer):
        """
        Validates a yes/no answer.
        :return: True when the answer is 'y'; otherwise, False.
        """
        return answer.lower() == "y", ""

    @staticmethod
    def is_int(answer):
        """
        Validates that the answer can be converted to an int.
        :return: The int answer; otherwise, None.
        """
        try:
            int_answer = int(answer)
        except ValueError:
            int_answer = None
        return int_answer, f"{answer} must be a valid integer."

    @staticmethod
    def is_letter(answer):
        """
        Validates that the answer is a letter.
        :return The letter answer, converted to uppercase; otherwise, None.
        """
        return (
            answer.upper() if answer.isalpha() else None,
            f"{answer} must be a single letter.",
        )

    @staticmethod
    def is_float(answer):
        """
        Validate that the answer can be converted to a float.
        :return The float answer; otherwise, None.
        """
        try:
            float_answer = float(answer)
        except ValueError:
            float_answer = None
        return float_answer, f"{answer} must be a valid float."

    @staticmethod
    def in_range(lower, upper):
```

```
        """
        Validate that the answer is within a range. The answer must be of a type
that can
        be compared to the lower and upper bounds.
        :return: The answer, if it is within the range; otherwise, None.
        """

        def _validate(answer):
            return (
                answer if lower <= answer <= upper else None,
                f"{answer} must be between {lower} and {upper}.",
            )

        return _validate
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`BatchExecuteStatement`

O código de exemplo a seguir mostra como usar `BatchExecuteStatement`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
 to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated with
                           each statement. This list must be in the same order as
 the
                           statements.
        :return: The responses returned from running the statements, if any.
        """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[
                    {"Statement": statement, "Parameters": params}
                    for statement, params in zip(statements, param_list)
                ]
```

```
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the table
"
                    "does not exist."
                )
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why: %s:
%s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

- Para obter detalhes da API, consulte a BatchExecuteStatement Referência da API AWS SDK for Python (Boto3).

`BatchGetItem`

O código de exemplo a seguir mostra como usar `BatchGetItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import decimal
import json
import logging
import os
import pprint
```

```
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
dynamodb = boto3.resource("dynamodb")

MAX_GET_SIZE = 100  # Amazon DynamoDB rejects a get batch larger than 100 items.


def do_batch_get(batch_keys):
    """
    Gets a batch of items from Amazon DynamoDB. Batches can contain keys from
    more than one table.

    When Amazon DynamoDB cannot process all items in a batch, a set of unprocessed
    keys is returned. This function uses an exponential backoff algorithm to retry
    getting the unprocessed keys until all are retrieved or the specified
    number of tries is reached.

    :param batch_keys: The set of keys to retrieve. A batch can contain at most 100
                       keys. Otherwise, Amazon DynamoDB returns an error.
    :return: The dictionary of retrieved items grouped under their respective
             table names.
    """
    tries = 0
    max_tries = 5
    sleepy_time = 1  # Start with 1 second of sleep, then exponentially increase.
    retrieved = {key: [] for key in batch_keys}
    while tries < max_tries:
        response = dynamodb.batch_get_item(RequestItems=batch_keys)
        # Collect any retrieved items and retry unprocessed keys.
        for key in response.get("Responses", []):
            retrieved[key] += response["Responses"][key]
        unprocessed = response["UnprocessedKeys"]
        if len(unprocessed) > 0:
            batch_keys = unprocessed
            unprocessed_count = sum(
                [len(batch_key["Keys"]) for batch_key in batch_keys.values()]
            )
            logger.info(
                "%s unprocessed keys returned. Sleep, then retry.",
 unprocessed_count
            )
```

```
            tries += 1
            if tries < max_tries:
                logger.info("Sleeping for %s seconds.", sleepy_time)
                time.sleep(sleepy_time)
                sleepy_time = min(sleepy_time * 2, 32)
        else:
            break

    return retrieved
```

- Para obter detalhes da API, consulte a BatchGetItem Referência da API AWS SDK for Python (Boto3).

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
```

```
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at least
                       the keys required by the schema that was specified when the
                       table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Para obter detalhes da API, consulte a BatchWriteItem Referência da API AWS SDK for Python (Boto3).

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

Crie uma tabela para armazenar dados de filmes.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
```

```
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def create_table(self, table_name):
        """
        Creates an Amazon DynamoDB table that can be used to store movie data.
        The table uses the release year of the movie as the partition key and the
        title as the sort key.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
        try:
            self.table = self.dyn_resource.create_table(
                TableName=table_name,
                KeySchema=[
                    {"AttributeName": "year", "KeyType": "HASH"},  # Partition key
                    {"AttributeName": "title", "KeyType": "RANGE"},  # Sort key
                ],
                AttributeDefinitions=[
                    {"AttributeName": "year", "AttributeType": "N"},
                    {"AttributeName": "title", "AttributeType": "S"},
                ],
                BillingMode='PAY_PER_REQUEST',
            )
            self.table.wait_until_exists()
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return self.table
```

- Para obter detalhes da API, consulte a CreateTable Referência da API AWS SDK for Python (Boto3).

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """
```

```
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

É possível especificar uma condição para que um item seja excluído somente quando ele atender a determinados critérios.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def delete_underrated_movie(self, title, year, rating):
        """
        Deletes a movie only if it is rated below a specified value. By using a
        condition expression in a delete operation, you can specify that an item is
```

```
        deleted only when it meets certain criteria.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        :param rating: The rating threshold to check before deleting the movie.
        """
        try:
            self.table.delete_item(
                Key={"year": year, "title": title},
                ConditionExpression="info.rating <= :val",
                ExpressionAttributeValues={":val": Decimal(str(rating))},
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't delete %s because its rating is greater than %s.",
                    title,
                    rating,
                )
            else:
                logger.error(
                    "Couldn't delete movie %s. Here's why: %s: %s",
                    title,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteItem Referência da API AWS SDK for Python (Boto3).

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def delete_table(self):
```

```
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTable Referência da API AWS SDK for Python (Boto3).

`DescribeTable`

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
```

```
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            table.load()
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                exists = False
            else:
                logger.error(
                    "Couldn't check for existence of %s. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
```

```
                )
                raise
        else:
            self.table = table
        return exists
```

- Para obter detalhes da API, consulte a DescribeTable Referência da API AWS SDK for Python (Boto3).

`DescribeTimeToLive`

O código de exemplo a seguir mostra como usar `DescribeTimeToLive`.

SDK para Python (Boto3)

Descreva a configuração de TTL em uma tabela existente do DynamoDB usando o AWS SDK para Python (Boto3).

```
import boto3


def describe_ttl(table_name, region):
    """
    Describes TTL on an existing table, as well as a region.

    :param table_name: String representing the name of the table
    :param region: AWS Region of the table - example `us-east-1`
    :return: Time to live description.
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        ttl_description = dynamodb.describe_time_to_live(TableName=table_name)
        print(
            f"TimeToLive for table {table_name} is status
 {ttl_description['TimeToLiveDescription']['TimeToLiveStatus']}"
        )

        return ttl_description
    except Exception as e:
        print(f"Error describing table: {e}")
        raise
```

```
# Enter your own table name and AWS region
describe_ttl("your-table-name", "us-east-1")
```

- Para obter detalhes da API, consulte a DescribeTimeToLive Referência da API AWS SDK for Python (Boto3).

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
 to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.
```

```
        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                       statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not
exist.",
                    statement,
                )
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

- Para obter detalhes da API, consulte a ExecuteStatement Referência da API AWS SDK for Python (Boto3).

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def get_movie(self, title, year):
```

```
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Item"]
```

- Para obter detalhes da API, consulte a GetItem Referência da API AWS SDK for Python (Boto3).

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.
```

```
    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tables
```

- Para obter detalhes da API, consulte a ListTables Referência da API AWS SDK for Python (Boto3).

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
```

```
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    "year": year,
                    "title": title,
                    "info": {"plot": plot, "rating": Decimal(str(rating))},
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a PutItem Referência da API AWS SDK for Python (Boto3).

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Consulte itens usando uma expressão de condição de chave.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """
```

```
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
        :return: The list of movies that were released in the specified year.
        """
        try:
            response = self.table.query(KeyConditionExpression=Key("year").eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
                year,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Items"]
```

Consulte itens e projete-os para retornar um subconjunto de dados.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def query_and_project_movies(self, year, title_bounds):
        """
        Query for movies that were released in a specified year and that have titles
        that start within a range of letters. A projection expression is used
```

```
        to return a subset of data for each movie.

        :param year: The release year to query.
        :param title_bounds: The range of starting letters to query.
        :return: The list of movies.
        """
        try:
            response = self.table.query(
                ProjectionExpression="#yr, title, info.genres, info.actors[0]",
                ExpressionAttributeNames={"#yr": "year"},
                KeyConditionExpression=(
                    Key("year").eq(year)
                    & Key("title").between(
                        title_bounds["first"], title_bounds["second"]
                    )
                ),
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                logger.warning(
                    "There's a validation error. Here's the message: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't query for movies. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return response["Items"]
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def scan_movies(self, year_range):
```

```
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            "FilterExpression": Key("year").between(
                year_range["first"], year_range["second"]
            ),
            "ProjectionExpression": "#yr, title, info.rating",
            "ExpressionAttributeNames": {"#yr": "year"},
        }
        try:
            done = False
            start_key = None
            while not done:
                if start_key:
                    scan_kwargs["ExclusiveStartKey"] = start_key
                response = self.table.scan(**scan_kwargs)
                movies.extend(response.get("Items", []))
                start_key = response.get("LastEvaluatedKey", None)
                done = start_key is None
        except ClientError as err:
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

        return movies
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Python (Boto3).

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualize um item usando uma expressão de atualização.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to the give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={":r": Decimal(str(rating)), ":p": plot},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]
```

Atualize um item usando uma expressão de atualização que inclui uma operação aritmética.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def update_rating(self, title, year, rating_change):
        """
        Updates the quality rating of a movie in the table by using an arithmetic
        operation in the update expression. By specifying an arithmetic operation,
        you can adjust a value in a single request, rather than first getting its
```

```
        value and then setting its new value.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating_change: The amount to add to the current rating for the movie.
        :return: The updated rating.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating = info.rating + :val",
                ExpressionAttributeValues={":val": Decimal(str(rating_change))},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]
```

Atualize um item somente quando ele atender a determinadas condições.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def remove_actors(self, title, year, actor_threshold):
        """
        Removes an actor from a movie, but only when the number of actors is greater
        than a specified threshold. If the movie does not list more than the
 threshold,
        no actors are removed.

        :param title: The title of the movie to update.
```

```
        :param year: The release year of the movie to update.
        :param actor_threshold: The threshold of actors to check.
        :return: The movie data after the update.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="remove info.actors[0]",
                ConditionExpression="size(info.actors) > :num",
                ExpressionAttributeValues={":num": actor_threshold},
                ReturnValues="ALL_NEW",
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't update %s because it has fewer than %s actors.",
                    title,
                    actor_threshold + 1,
                )
            else:
                logger.error(
                    "Couldn't update movie %s. Here's why: %s: %s",
                    title,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["Attributes"]
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

`UpdateTimeToLive`

O código de exemplo a seguir mostra como usar `UpdateTimeToLive`.

SDK para Python (Boto3)

Habilite a TTL em uma tabela existente do DynamoDB.

```
import boto3


def enable_ttl(table_name, ttl_attribute_name):
    """
    Enables TTL on DynamoDB table for a given attribute name
        on success, returns a status code of 200
        on error, throws an exception

    :param table_name: Name of the DynamoDB table
    :param ttl_attribute_name: The name of the TTL attribute being provided to the
 table.
    """
    try:
        dynamodb = boto3.client("dynamodb")

        # Enable TTL on an existing DynamoDB table
        response = dynamodb.update_time_to_live(
            TableName=table_name,
            TimeToLiveSpecification={"Enabled": True, "AttributeName":
 ttl_attribute_name},
        )

        # In the returned response, check for a successful status code.
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            print("TTL has been enabled successfully.")
        else:
            print(
                f"Failed to enable TTL, status code {response['ResponseMetadata']
['HTTPStatusCode']}"
            )
        return response
    except Exception as ex:
        print("Couldn't enable TTL in table %s. Here's why: %s" % (table_name, ex))
        raise


# your values
enable_ttl("your-table-name", "expireAt")
```

Desabilite a TTL em uma tabela existente do DynamoDB.

```
import boto3


def disable_ttl(table_name, ttl_attribute_name):
    """
    Disables TTL on DynamoDB table for a given attribute name
        on success, returns a status code of 200
        on error, throws an exception

    :param table_name: Name of the DynamoDB table being modified
    :param ttl_attribute_name: The name of the TTL attribute being provided to the
 table.
    """
    try:
        dynamodb = boto3.client("dynamodb")

        # Enable TTL on an existing DynamoDB table
        response = dynamodb.update_time_to_live(
            TableName=table_name,
            TimeToLiveSpecification={"Enabled": False, "AttributeName":
 ttl_attribute_name},
        )

        # In the returned response, check for a successful status code.
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            print("TTL has been disabled successfully.")
        else:
            print(
                f"Failed to disable TTL, status code {response['ResponseMetadata']
['HTTPStatusCode']}"
            )
    except Exception as ex:
        print("Couldn't disable TTL in table %s. Here's why: %s" % (table_name, ex))
        raise


# your values
disable_ttl("your-table-name", "expireAt")
```

- Para obter detalhes da API, consulte a UpdateTimeToLive Referência da API AWS SDK for Python (Boto3).

Cenários

Acelerar leituras com o DAX

O exemplo de código a seguir mostra como:

- Criar e gravar dados em uma tabela com os clientes DAX e SDK.
- Obter, consultar e verificar a tabela com ambos os clientes e comparar a respectiva performance.

Para obter mais informações, consulte Desenvolver com o cliente do DynamoDB Accelerator.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma tabela com o cliente do DAX ou Boto3.

```
import boto3


def create_dax_table(dyn_resource=None):
    """
    Creates a DynamoDB table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The newly created table.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table_name = "TryDaxTable"
    params = {
        "TableName": table_name,
        "KeySchema": [
            {"AttributeName": "partition_key", "KeyType": "HASH"},
            {"AttributeName": "sort_key", "KeyType": "RANGE"},
        ],
```

```
        "AttributeDefinitions": [
            {"AttributeName": "partition_key", "AttributeType": "N"},
            {"AttributeName": "sort_key", "AttributeType": "N"},
        ],
        "BillingMode": "PAY_PER_REQUEST",
    }
    table = dyn_resource.create_table(**params)
    print(f"Creating {table_name}...")
    table.wait_until_exists()
    return table


if __name__ == "__main__":
    dax_table = create_dax_table()
    print(f"Created table.")
```

Grave dados de teste na tabela.

```
import boto3


def write_data_to_dax_table(key_count, item_size, dyn_resource=None):
    """
    Writes test data to the demonstration table.

    :param key_count: The number of partition and sort keys to use to populate the
                      table. The total number of items is key_count * key_count.
    :param item_size: The size of non-key data for each test item.
    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    some_data = "X" * item_size

    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.put_item(
                Item={
                    "partition_key": partition_key,
                    "sort_key": sort_key,
```

```
                    "some_data": some_data,
                }
            )
            print(f"Put item ({partition_key}, {sort_key}) succeeded.")


if __name__ == "__main__":
    write_key_count = 10
    write_item_size = 1000
    print(
        f"Writing {write_key_count*write_key_count} items to the table. "
        f"Each item is {write_item_size} characters."
    )
    write_data_to_dax_table(write_key_count, write_item_size)
```

Obtenha itens para várias iterações associadas ao cliente do DAX e ao cliente do Boto3 e relate o tempo gasto para cada um.

```
import argparse
import sys
import time
import amazondax
import boto3


def get_item_test(key_count, iterations, dyn_resource=None):
    """
    Gets items from the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param key_count: The number of items to get from the table in each iteration.
    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    start = time.perf_counter()
    for _ in range(iterations):
```

```
        for partition_key in range(1, key_count + 1):
            for sort_key in range(1, key_count + 1):
                table.get_item(
                    Key={"partition_key": partition_key, "sort_key": sort_key}
                )
                print(".", end="")
                sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_key_count = 10
    test_iterations = 50
    if args.endpoint_url:
        print(
            f"Getting each item from the table {test_iterations} times, "
            f"using the DAX client."
        )
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
 dax:
            test_start, test_end = get_item_test(
                test_key_count, test_iterations, dyn_resource=dax
            )
    else:
        print(
            f"Getting each item from the table {test_iterations} times, "
            f"using the Boto3 client."
        )
        test_start, test_end = get_item_test(test_key_count, test_iterations)
    print(
```

```
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/ test_iterations}."
    )
```

Consulte a tabela para várias iterações associadas ao cliente do DAX e ao cliente do Boto3 e relate o tempo gasto para cada um.

```
import argparse
import time
import sys
import amazondax
import boto3
from boto3.dynamodb.conditions import Key


def query_test(partition_key, sort_keys, iterations, dyn_resource=None):
    """
    Queries the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param partition_key: The partition key value to use in the query. The query
                          returns items that have partition keys equal to this
 value.
    :param sort_keys: The range of sort key values for the query. The query returns
                      items that have sort key values between these two values.
    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    key_condition_expression = Key("partition_key").eq(partition_key) & Key(
        "sort_key"
    ).between(*sort_keys)

    start = time.perf_counter()
    for _ in range(iterations):
        table.query(KeyConditionExpression=key_condition_expression)
        print(".", end="")
```

```
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_partition_key = 5
    test_sort_keys = (2, 9)
    test_iterations = 100
    if args.endpoint_url:
        print(f"Querying the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
 dax:
            test_start, test_end = query_test(
                test_partition_key, test_sort_keys, test_iterations,
 dyn_resource=dax
            )
    else:
        print(f"Querying the table {test_iterations} times, using the Boto3
 client.")
        test_start, test_end = query_test(
            test_partition_key, test_sort_keys, test_iterations
        )

    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/test_iterations}."
    )
```

Verifique a tabela para várias iterações associadas ao cliente do DAX e ao cliente do Boto3 e relate o tempo gasto para cada um.

```
import argparse
import time
import sys
import amazondax
import boto3


def scan_test(iterations, dyn_resource=None):
    """
    Scans the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    start = time.perf_counter()
    for _ in range(iterations):
        table.scan()
        print(".", end="")
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
```

```
    args = parser.parse_args()

    test_iterations = 100
    if args.endpoint_url:
        print(f"Scanning the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
 dax:
            test_start, test_end = scan_test(test_iterations, dyn_resource=dax)
    else:
        print(f"Scanning the table {test_iterations} times, using the Boto3
 client.")
        test_start, test_end = scan_test(test_iterations)
    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/test_iterations}."
    )
```

Exclua a tabela.

```
import boto3


def delete_dax_table(dyn_resource=None):
    """
    Deletes the demonstration table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    table.delete()

    print(f"Deleting {table.name}...")
    table.wait_until_not_exists()


if __name__ == "__main__":
    delete_dax_table()
```

```
    print("Table deleted!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTable
  - DeleteTable
  - GetItem
  - PutItem
  - Query
  - Scan

Comparar vários valores com um único atributo

O exemplo de código a seguir mostra como comparar vários valores com um único atributo no DynamoDB.

- Use o operador IN para comparar vários valores com um único atributo.
- Compare o operador IN com várias condições OR.
- Saiba quais são os benefícios de desempenho e complexidade de expressão oferecidos pelo IN.

SDK para Python (Boto3)

Compare vários valores com um único atributo usando AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key
from typing import Any, Dict, List, Optional


def compare_multiple_values(
    table_name: str,
    attribute_name: str,
    values_list: List[Any],
    partition_key_name: Optional[str] = None,
    partition_key_value: Optional[str] = None,
) -> Dict[str, Any]:
    """
```

```
    Query or scan a DynamoDB table to find items where an attribute matches any
value from a list.

    This function demonstrates the use of the IN operator to compare a single
attribute
    against multiple possible values, which is more efficient than using multiple OR
conditions.

    Args:
        table_name (str): The name of the DynamoDB table.
        attribute_name (str): The name of the attribute to compare against the
values list.
        values_list (List[Any]): List of values to compare the attribute against.
        partition_key_name (Optional[str]): The name of the partition key attribute
for query operations.
        partition_key_value (Optional[str]): The value of the partition key to
query.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the matching items.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create the filter expression using the is_in method
    filter_expression = Attr(attribute_name).is_in(values_list)

    # If partition key is provided, perform a query operation
    if partition_key_name and partition_key_value:
        key_condition = Key(partition_key_name).eq(partition_key_value)
        response = table.query(
            KeyConditionExpression=key_condition, FilterExpression=filter_expression
        )
    else:
        # Otherwise, perform a scan operation
        response = table.scan(FilterExpression=filter_expression)

    # Handle pagination if there are more results
    items = response.get("Items", [])
    while "LastEvaluatedKey" in response:
        if partition_key_name and partition_key_value:
            response = table.query(
                KeyConditionExpression=key_condition,
```

```
                FilterExpression=filter_expression,
                ExclusiveStartKey=response["LastEvaluatedKey"],
            )
        else:
            response = table.scan(
                FilterExpression=filter_expression,
 ExclusiveStartKey=response["LastEvaluatedKey"]
            )
        items.extend(response.get("Items", []))

    # Return the complete result
    return {"Items": items, "Count": len(items)}


def compare_with_or_conditions(
    table_name: str,
    attribute_name: str,
    values_list: List[Any],
    partition_key_name: Optional[str] = None,
    partition_key_value: Optional[str] = None,
) -> Dict[str, Any]:
    """
    Alternative implementation using multiple OR conditions instead of the IN
 operator.

    This function is provided for comparison to show why using the IN operator is
 preferable.
    With many values, this approach becomes verbose and less efficient.

    Args:
        table_name (str): The name of the DynamoDB table.
        attribute_name (str): The name of the attribute to compare against the
 values list.
        values_list (List[Any]): List of values to compare the attribute against.
        partition_key_name (Optional[str]): The name of the partition key attribute
 for query operations.
        partition_key_value (Optional[str]): The value of the partition key to
 query.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the matching items.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
```

```
    table = dynamodb.Table(table_name)

    # Create a filter expression with multiple OR conditions
    filter_expression = None
    for value in values_list:
        condition = Attr(attribute_name).eq(value)
        if filter_expression is None:
            filter_expression = condition
        else:
            filter_expression = filter_expression | condition

    # If partition key is provided, perform a query operation
    if partition_key_name and partition_key_value and filter_expression:
        key_condition = Key(partition_key_name).eq(partition_key_value)
        response = table.query(
            KeyConditionExpression=key_condition, FilterExpression=filter_expression
        )
    elif filter_expression:
        # Otherwise, perform a scan operation
        response = table.scan(FilterExpression=filter_expression)
    else:
        # Return empty response if no values provided
        return {"Items": [], "Count": 0}

    # Handle pagination if there are more results
    items = response.get("Items", [])
    while "LastEvaluatedKey" in response:
        if partition_key_name and partition_key_value:
            response = table.query(
                KeyConditionExpression=key_condition,
                FilterExpression=filter_expression,
                ExclusiveStartKey=response["LastEvaluatedKey"],
            )
        else:
            response = table.scan(
                FilterExpression=filter_expression,
ExclusiveStartKey=response["LastEvaluatedKey"]
            )
        items.extend(response.get("Items", []))

    # Return the complete result
    return {"Items": items, "Count": len(items)}
```

Exemplo de uso de comparação de vários valores com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the compare_multiple_values function."""
    # Example parameters
    table_name = "Products"
    attribute_name = "Category"
    values_list = ["Electronics", "Computers", "Accessories"]

    print(f"Searching for products in any of these categories: {values_list}")

    # Using the IN operator (recommended approach)
    print("\nApproach 1: Using the IN operator")
    response = compare_multiple_values(
        table_name=table_name, attribute_name=attribute_name,
 values_list=values_list
    )

    print(f"Found {response['Count']} products in the specified categories")

    # Using multiple OR conditions (alternative approach)
    print("\nApproach 2: Using multiple OR conditions")
    response2 = compare_with_or_conditions(
        table_name=table_name, attribute_name=attribute_name,
 values_list=values_list
    )

    print(f"Found {response2['Count']} products in the specified categories")

    # Example with a query operation
    print("\nQuerying a specific manufacturer's products in multiple categories")
    partition_key_name = "Manufacturer"
    partition_key_value = "Acme"

    response3 = compare_multiple_values(
        table_name=table_name,
        attribute_name=attribute_name,
        values_list=values_list,
        partition_key_name=partition_key_name,
        partition_key_value=partition_key_value,
    )
```

```
    print(f"Found {response3['Count']} Acme products in the specified categories")

    # Explain the benefits of using the IN operator
    print("\nBenefits of using the IN operator:")
    print("1. More concise expression compared to multiple OR conditions")
    print("2. Better readability and maintainability")
    print("3. Potentially better performance with large value lists")
    print("4. Simpler code that's less prone to errors")
    print("5. Easier to modify when adding or removing values")
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - Query
  - Scan

Atualizar condicionalmente a TTL de um item

O exemplo de código a seguir mostra como atualizar condicionalmente a TTL de um item.

SDK para Python (Boto3)

Atualize a TTL em um item do DynamoDB existente em uma tabela, com uma condição.

```
from datetime import datetime, timedelta

import boto3
from botocore.exceptions import ClientError


def update_dynamodb_item_ttl(table_name, region, primary_key, sort_key,
 ttl_attribute):
    """
    Updates an existing record in a DynamoDB table with a new or updated TTL
 attribute.

    :param table_name: Name of the DynamoDB table
    :param region: AWS Region of the table - example `us-east-1`
    :param primary_key: one attribute known as the partition key.
```

```
    :param sort_key: Also known as a range attribute.
    :param ttl_attribute: name of the TTL attribute in the target DynamoDB table
    :return:
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Generate updated TTL in epoch second format
        updated_expiration_time = int((datetime.now() +
 timedelta(days=90)).timestamp())

        # Define the update expression for adding/updating a new attribute
        update_expression = "SET newAttribute = :val1"

        # Define the condition expression for checking if 'expireAt' is not expired
        condition_expression = "expireAt > :val2"

        # Define the expression attribute values
        expression_attribute_values = {":val1": ttl_attribute, ":val2":
 updated_expiration_time}

        response = table.update_item(
            Key={"primaryKey": primary_key, "sortKey": sort_key},
            UpdateExpression=update_expression,
            ConditionExpression=condition_expression,
            ExpressionAttributeValues=expression_attribute_values,
        )

        print("Item updated successfully.")
        return response["ResponseMetadata"]["HTTPStatusCode"]  # Ideally a 200 OK
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            print("Condition check failed: Item's 'expireAt' is expired.")
        else:
            print(f"Error updating item: {e}")
    except Exception as e:
        print(f"Error updating item: {e}")


# replace with your values
update_dynamodb_item_ttl(
    "your-table-name",
    "us-east-1",
```

```
    "your-partition-key-value",
    "your-sort-key-value",
    "your-ttl-attribute-value",
)
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

Contar operadores de expressão

O exemplo de código a seguir mostra como contar operadores de expressão no DynamoDB.

- Entenda o limite de 300 operadores do DynamoDB.
- Conte operadores em expressões complexas.
- Otimize as expressões para permanecer dentro dos limites.

SDK para Python (Boto3)

Demonstre a contagem de operadores de expressão usando o AWS SDK para Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, List, Optional, Tuple



def create_complex_filter_expression(
    attribute_name: str, values: List[Any], use_or: bool = True
) -> Tuple[str, Dict[str, Any], Dict[str, str], int]:
    """
    Create a complex filter expression with multiple conditions.

    This function demonstrates how to build a complex filter expression
    and count the number of operators used.

    Args:
        attribute_name (str): The name of the attribute to filter on.
        values (List[Any]): List of values to compare against.
        use_or (bool, optional): Whether to use OR between conditions. Defaults to
 True.
```

```
    Returns:
        Tuple[str, Dict[str, Any], Dict[str, str], int]: A tuple containing:
            - The filter expression string
            - Expression attribute values
            - Expression attribute names
            - The number of operators used
    """
    if not values:
        return "", {}, {}, 0

    # Initialize expression components
    filter_expression = ""
    expression_attribute_values = {}
    expression_attribute_names = {"#attr": attribute_name}
    operator_count = 0

    # Build the filter expression
    for i, value in enumerate(values):
        value_placeholder = f":val{i}"
        expression_attribute_values[value_placeholder] = value

        if i > 0:
            # Add OR or AND operator between conditions
            filter_expression += " OR " if use_or else " AND "
            operator_count += 1  # Count the OR/AND operator

        # Add the condition
        filter_expression += f"#attr = {value_placeholder}"
        operator_count += 1  # Count the = operator

    return (
        filter_expression,
        expression_attribute_values,
        expression_attribute_names,
        operator_count,
    )


def create_nested_filter_expression(
    depth: int, conditions_per_level: int
) -> Tuple[str, Dict[str, Any], Dict[str, str], int]:
    """
    Create a deeply nested filter expression with multiple conditions.
```

```
    This function demonstrates how to build a complex nested filter expression
    and count the number of operators used.

    Args:
        depth (int): The depth of nesting.
        conditions_per_level (int): Number of conditions at each level.

    Returns:
        Tuple[str, Dict[str, Any], Dict[str, str], int]: A tuple containing:
            - The filter expression string
            - Expression attribute values
            - Expression attribute names
            - The number of operators used
    """
    if depth <= 0 or conditions_per_level <= 0:
        return "", {}, {}, 0

    # Initialize expression components
    expression_attribute_values = {}
    expression_attribute_names = {}
    operator_count = 0

    def build_nested_expression(current_depth: int, prefix: str) -> str:
        nonlocal operator_count

        if current_depth <= 0:
            return ""

        # Build conditions at this level
        conditions = []
        for i in range(conditions_per_level):
            attr_name = f"attr{prefix}_{i}"
            attr_placeholder = f"#attr{prefix}_{i}"
            val_placeholder = f":val{prefix}_{i}"

            expression_attribute_names[attr_placeholder] = attr_name
            expression_attribute_values[val_placeholder] = i

            conditions.append(f"{attr_placeholder} = {val_placeholder}")
            operator_count += 1  # Count the = operator

        # Join conditions with AND
        level_expression = " AND ".join(conditions)
```

```
        operator_count += max(0, len(conditions) - 1)  # Count the AND operators

        # If not at the deepest level, add nested expressions
        if current_depth > 1:
            nested_expr = build_nested_expression(current_depth - 1,
 f"{prefix}_{current_depth}")
            if nested_expr:
                level_expression = f"({level_expression}) OR ({nested_expr})"
                operator_count += 1  # Count the OR operator

        return level_expression

    # Build the expression starting from the top level
    filter_expression = build_nested_expression(depth, "1")

    return (
        filter_expression,
        expression_attribute_values,
        expression_attribute_names,
        operator_count,
    )


def count_operators_in_update_expression(update_expression: str) -> int:
    """
    Count the number of operators in an update expression.

    This function demonstrates how to count operators in an update expression
    based on DynamoDB's rules.

    Args:
        update_expression (str): The update expression to analyze.

    Returns:
        int: The number of operators in the expression.
    """
    operator_count = 0

    # Count SET operations
    if "SET" in update_expression:
        set_section = (
            update_expression.split("SET")[1].split("REMOVE")[0].split("ADD")
[0].split("DELETE")[0]
        )
```

```
        # Count assignment operators (=)
        operator_count += set_section.count("=")

        # Count arithmetic operators (+, -)
        operator_count += set_section.count("+")
        operator_count += set_section.count("-")

        # Count list_append function calls (each counts as 1 operator)
        operator_count += set_section.lower().count("list_append")

        # Count if_not_exists function calls (each counts as 1 operator)
        operator_count += set_section.lower().count("if_not_exists")

    # Count REMOVE operations (no additional operators)

    # Count ADD operations (each ADD counts as 1 operator)
    if "ADD" in update_expression:
        add_section = (
            update_expression.split("ADD")[1].split("DELETE")[0].split("SET")
[0].split("REMOVE")[0]
        )
        operator_count += add_section.count(",") + 1

    # Count DELETE operations (each DELETE counts as 1 operator)
    if "DELETE" in update_expression:
        delete_section = (
            update_expression.split("DELETE")[1].split("SET")[0].split("ADD")
[0].split("REMOVE")[0]
        )
        operator_count += delete_section.count(",") + 1

    return operator_count


def count_operators_in_condition_expression(condition_expression: str) -> int:
    """
    Count the number of operators in a condition expression.

    This function demonstrates how to count operators in a condition expression
    based on DynamoDB's rules.

    Args:
        condition_expression (str): The condition expression to analyze.
```

```
    Returns:
        int: The number of operators in the expression.
    """
    operator_count = 0

    # Count comparison operators
    comparison_operators = ["=", "<>", "<", "<=", ">", ">="]
    for op in comparison_operators:
        operator_count += condition_expression.count(op)

    # Count logical operators
    operator_count += condition_expression.upper().count(" AND ")
    operator_count += condition_expression.upper().count(" OR ")
    operator_count += condition_expression.upper().count("NOT ")

    # Count BETWEEN operator (counts as 2: BETWEEN + AND)
    between_count = condition_expression.upper().count(" BETWEEN ")
    operator_count += between_count * 2

    # Count IN operator (counts as 1 regardless of number of values)
    operator_count += condition_expression.upper().count(" IN ")

    # Count functions (each counts as 1 operator)
    functions = [
        "attribute_exists",
        "attribute_not_exists",
        "attribute_type",
        "begins_with",
        "contains",
        "size",
    ]
    for func in functions:
        operator_count += condition_expression.lower().count(func)

    return operator_count


# Note: This function is for demonstration purposes only and should be called from
 example_usage()
# It's not meant to be used directly as a test function
def _test_expression_limit(
    table_name: str, key: Dict[str, Any], operator_count: int, attribute_name: str =
 "TestAttribute"
```

```
) -> Tuple[bool, Optional[str]]:
    """
    Test if an expression with a specific number of operators exceeds the limit.

    This function demonstrates how to test the 300 operator limit by creating
    an expression with a specified number of operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        operator_count (int): The number of operators to include in the expression.
        attribute_name (str, optional): The name of the attribute to update.
 Defaults to "TestAttribute".

    Returns:
        Tuple[bool, Optional[str]]: A tuple containing:
            - A boolean indicating if the operation succeeded
            - The error message if it failed, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create an update expression with the specified number of operators
    update_expression = f"SET #{attribute_name} = :val0"
    expression_attribute_names = {f"#{attribute_name}": attribute_name}
    expression_attribute_values = {":val0": 0}

    # Add additional SET operations to reach the desired operator count
    # Each assignment adds 1 operator
    for i in range(1, operator_count):
        attr_name = f"{attribute_name}{i}"
        attr_placeholder = f"#attr{i}"
        val_placeholder = f":val{i}"

        update_expression += f", {attr_placeholder} = {val_placeholder}"
        expression_attribute_names[attr_placeholder] = attr_name
        expression_attribute_values[val_placeholder] = i

    try:
        # Attempt the update operation
        table.update_item(
            Key=key,
            UpdateExpression=update_expression,
```

```
            ExpressionAttributeNames=expression_attribute_names,
            ExpressionAttributeValues=expression_attribute_values,
        )
        return True, None
    except ClientError as e:
        error_message = e.response["Error"]["Message"]

        if "expression contains too many operators" in error_message.lower():
            return False, error_message
        else:
            # Other error occurred
            raise
```

Exemplo de uso da contagem de operadores de expressão com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the expression operator counting functions."""

    print("Example 1: Creating a complex filter expression with multiple
 conditions")
    attribute_name = "Status"
    values = ["Active", "Pending", "Processing", "Shipped", "Delivered"]

    filter_expr, expr_attr_vals, expr_attr_names, op_count =
 create_complex_filter_expression(
        attribute_name=attribute_name, values=values, use_or=True
    )

    print(f"Filter Expression: {filter_expr}")
    print(f"Expression Attribute Values: {expr_attr_vals}")
    print(f"Expression Attribute Names: {expr_attr_names}")
    print(f"Operator Count: {op_count}")

    print("\nExample 2: Creating a nested filter expression")
    nested_expr, nested_vals, nested_names, nested_count =
 create_nested_filter_expression(
        depth=3, conditions_per_level=2
    )

    print(f"Nested Filter Expression: {nested_expr}")
```

```
    print(f"Operator Count: {nested_count}")

    print("\nExample 3: Counting operators in an update expression")
    update_expression = "SET #name = :name, #age = :age + :increment, #address.#city
= :city, #status = if_not_exists(#status, :default_status) REMOVE #old_field ADD
#counter :value DELETE #set_attr :set_val"
    update_op_count = count_operators_in_update_expression(update_expression)

    print(f"Update Expression: {update_expression}")
    print(f"Operator Count: {update_op_count}")

    print("\nExample 4: Counting operators in a condition expression")
    condition_expression = "(#status = :active OR #status = :pending) AND #price
BETWEEN :min_price AND :max_price AND attribute_exists(#category) AND NOT (#stock
<= :min_stock)"
    condition_op_count =
count_operators_in_condition_expression(condition_expression)

    print(f"Condition Expression: {condition_expression}")
    print(f"Operator Count: {condition_op_count}")

    print("\nExample 5: Testing the 300 operator limit")

    # This is just for demonstration - in a real application, you would use your
actual table
    # Note: This function is renamed to _test_expression_limit to avoid pytest
trying to run it
    print("In a real application, you would test with _test_expression_limit
function")
    print("Expression with 250 operators would be under the limit")
    print("Expression with 350 operators would exceed the 300 operator limit")

    print("\nOperator Counting Rules in DynamoDB:")
    print("1. Comparison Operators (=, <>, <, <=, >, >=): 1 operator each")
    print("2. Logical Operators (AND, OR, NOT): 1 operator each")
    print("3. BETWEEN: 2 operators (BETWEEN + AND)")
    print("4. IN: 1 operator (regardless of number of values)")
    print("5. Functions (attribute_exists, begins_with, etc.): 1 operator each")
    print("6. Arithmetic Operators (+, -): 1 operator each")
    print("7. SET assignments (=): 1 operator each")
    print("8. ADD and DELETE operations: 1 operator each")

    print("\nStrategies for Working Within the 300 Operator Limit:")
    print("1. Break operations into multiple requests")
```

```
    print("2. Use DynamoDB Transactions for complex operations")
    print("3. Optimize data model to reduce query complexity")
    print("4. Use application-side filtering for less critical filters")
    print("5. Consider using IN operator instead of multiple OR conditions")
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

Crie uma API REST para rastrear COVID-19 dados

O exemplo de código a seguir mostra como criar uma API REST que simula um sistema para rastrear casos diários de COVID-19 nos Estados Unidos usando dados fictícios.

SDK para Python (Boto3)

Mostra como usar o AWS Chalice com o AWS SDK para Python (Boto3) para criar uma API REST sem servidor que usa o Amazon API Gateway e o Amazon AWS Lambda DynamoDB. A API REST simula um sistema que rastreia casos diários de COVID-19 nos Estados Unidos, usando dados fictícios. Aprenda como:

- Use o AWS Chalice para definir rotas em funções do Lambda que são chamadas para lidar com solicitações REST que vêm por meio do API Gateway.
- Usar as funções do Lambda para recuperar e armazenar dados em uma tabela do DynamoDB para atender a solicitações REST.
- Defina a estrutura da tabela e os recursos da função de segurança em um AWS CloudFormation modelo.
- Use o AWS Chalice e CloudFormation empacote e implante todos os recursos necessários.
- Use CloudFormation para limpar todos os recursos criados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- CloudFormation

- DynamoDB
- Lambda

Criar uma aplicação de mensageiro

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.
- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SQS
- Funções de etapa

Criar uma tabela com o throughput a quente habilitado

O exemplo de código a seguir mostra como criar uma tabela com o throughput a quente habilitado.

SDK para Python (Boto3)

Crie uma tabela do DynamoDB com uma configuração de throughput a quente usando o AWS SDK para Python (Boto3).

```
from boto3 import client
from botocore.exceptions import ClientError


def create_dynamodb_table_warm_throughput(
    table_name,
    partition_key,
    sort_key,
    misc_key_attr,
    non_key_attr,
    table_provisioned_read_units,
    table_provisioned_write_units,
    table_warm_reads,
    table_warm_writes,
    gsi_name,
    gsi_provisioned_read_units,
    gsi_provisioned_write_units,
    gsi_warm_reads,
    gsi_warm_writes,
    region_name="us-east-1",
):
    """
    Creates a DynamoDB table with a warm throughput setting configured.

    :param table_name: The name of the table to be created.
    :param partition_key: The partition key for the table being created.
    :param sort_key: The sort key for the table being created.
    :param misc_key_attr: A miscellaneous key attribute for the table being created.
    :param non_key_attr: A non-key attribute for the table being created.
    :param table_provisioned_read_units: The newly created table's provisioned read
 capacity units.
    :param table_provisioned_write_units: The newly created table's provisioned
 write capacity units.
    :param table_warm_reads: The read units per second setting for the table's warm
 throughput.
    :param table_warm_writes: The write units per second setting for the table's
 warm throughput.
```

```
    :param gsi_name: The name of the Global Secondary Index (GSI) to be created on
the table.
    :param gsi_provisioned_read_units: The configured Global Secondary Index (GSI)
provisioned read capacity units.
    :param gsi_provisioned_write_units: The configured Global Secondary Index (GSI)
provisioned write capacity units.
    :param gsi_warm_reads: The read units per second setting for the Global
Secondary Index (GSI)'s warm throughput.
    :param gsi_warm_writes: The write units per second setting for the Global
Secondary Index (GSI)'s warm throughput.
    :param region_name: The AWS Region name to target. defaults to us-east-1
    """
    try:
        ddb = client("dynamodb", region_name=region_name)

        # Define the table attributes
        attribute_definitions = [
            {"AttributeName": partition_key, "AttributeType": "S"},
            {"AttributeName": sort_key, "AttributeType": "S"},
            {"AttributeName": misc_key_attr, "AttributeType": "N"},
        ]

        # Define the table key schema
        key_schema = [
            {"AttributeName": partition_key, "KeyType": "HASH"},
            {"AttributeName": sort_key, "KeyType": "RANGE"},
        ]

        # Define the provisioned throughput for the table
        provisioned_throughput = {
            "ReadCapacityUnits": table_provisioned_read_units,
            "WriteCapacityUnits": table_provisioned_write_units,
        }

        # Define the global secondary index
        gsi_key_schema = [
            {"AttributeName": sort_key, "KeyType": "HASH"},
            {"AttributeName": misc_key_attr, "KeyType": "RANGE"},
        ]
        gsi_projection = {"ProjectionType": "INCLUDE", "NonKeyAttributes":
[non_key_attr]}
        gsi_provisioned_throughput = {
            "ReadCapacityUnits": gsi_provisioned_read_units,
            "WriteCapacityUnits": gsi_provisioned_write_units,
```

```
        }
        gsi_warm_throughput = {
            "ReadUnitsPerSecond": gsi_warm_reads,
            "WriteUnitsPerSecond": gsi_warm_writes,
        }
        global_secondary_indexes = [
            {
                "IndexName": gsi_name,
                "KeySchema": gsi_key_schema,
                "Projection": gsi_projection,
                "ProvisionedThroughput": gsi_provisioned_throughput,
                "WarmThroughput": gsi_warm_throughput,
            }
        ]

        # Define the warm throughput for the table
        warm_throughput = {
            "ReadUnitsPerSecond": table_warm_reads,
            "WriteUnitsPerSecond": table_warm_writes,
        }

        # Create the DynamoDB client and create the table
        response = ddb.create_table(
            TableName=table_name,
            AttributeDefinitions=attribute_definitions,
            KeySchema=key_schema,
            ProvisionedThroughput=provisioned_throughput,
            GlobalSecondaryIndexes=global_secondary_indexes,
            WarmThroughput=warm_throughput,
        )

        print(response)
        return response
    except ClientError as e:
        print(f"Error creating table: {e}")
        raise e
```

- Para obter detalhes da API, consulte a CreateTable Referência da API AWS SDK for Python (Boto3).

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho no Amazon DynamoDB e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em uma tabela do DynamoDB.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o exemplo completo no Repositório de exemplos de AWS código em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Criar uma aplicação de chat websocket

O exemplo de código a seguir mostra como criar uma aplicação de chat que é atendido por uma API de Websocket criada no Amazon API Gateway.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon API Gateway V2 para criar uma API de websocket que se integre com AWS Lambda o Amazon DynamoDB.

- Crie uma API de Websocket atendida pelo API Gateway.
- Defina um manipulador do Lambda que armazena conexões no DynamoDB e publica mensagens para outros participantes do chat.
- Conecte-se à aplicação de chat websocket e envie mensagens com o pacote Websockets.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Criar um item com TTL

O exemplo de código a seguir mostra como criar um item com TTL.

SDK para Python (Boto3)

```
from datetime import datetime, timedelta

import boto3


def create_dynamodb_item(table_name, region, primary_key, sort_key):
    """
    Creates a DynamoDB item with an attached expiry attribute.

    :param table_name: Table name for the boto3 resource to target when creating an
 item
    :param region: string representing the AWS region. Example: `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :return: Void (nothing)
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Get the current time in epoch second format
        current_time = int(datetime.now().timestamp())

        # Calculate the expiration time (90 days from now) in epoch second format
        expiration_time = int((datetime.now() + timedelta(days=90)).timestamp())

        item = {
```

```
            "primaryKey": primary_key,
            "sortKey": sort_key,
            "creationDate": current_time,
            "expireAt": expiration_time,
        }
        response = table.put_item(Item=item)

        print("Item created successfully.")
        return response
    except Exception as e:
        print(f"Error creating item: {e}")
        raise e


# Use your own values
create_dynamodb_item(
    "your-table-name", "us-west-2", "your-partition-key-value", "your-sort-key-
value"
)
```

- Para obter detalhes da API, consulte a PutItem Referência da API AWS SDK for Python (Boto3).

Executar operações de consulta avançada

O exemplo de código a seguir mostra como realizar operações de consulta avançada no DynamoDB.

- Consulte tabelas usando várias técnicas de filtragem e condição.
- Implemente a paginação para grandes conjuntos de resultados.
- Use índices secundários globais para padrões de acesso alternativos.
- Aplique controles de consistência com base nos requisitos da aplicação.

SDK para Python (Boto3)

Consulte com leituras fortemente consistentes usando AWS SDK para Python (Boto3).

```
import time

import boto3
from boto3.dynamodb.conditions import Key
```

```
def query_with_consistent_read(
    table_name,
    partition_key_name,
    partition_key_value,
    sort_key_name=None,
    sort_key_value=None,
    consistent_read=True,
):
    """
    Query a DynamoDB table with the option for strongly consistent reads.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str, optional): The name of the sort key attribute.
        sort_key_value (str, optional): The value of the sort key to query.
        consistent_read (bool, optional): Whether to use strongly consistent reads.
 Defaults to True.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    if sort_key_name and sort_key_value:
        key_condition = key_condition & Key(sort_key_name).eq(sort_key_value)

    # Perform the query with the consistent read option
    response = table.query(KeyConditionExpression=key_condition,
 ConsistentRead=consistent_read)

    return response
```

Consulte usando um Índice Secundário Global com AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Key


def query_table(table_name, partition_key_name, partition_key_value):
    """
    Query a DynamoDB table using its primary key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the table's primary key
    response =
 table.query(KeyConditionExpression=Key(partition_key_name).eq(partition_key_value))

    return response


def query_gsi(table_name, index_name, partition_key_name, partition_key_value):
    """
    Query a Global Secondary Index (GSI) on a DynamoDB table.

    Args:
        table_name (str): The name of the DynamoDB table.
        index_name (str): The name of the Global Secondary Index.
        partition_key_name (str): The name of the GSI's partition key attribute.
        partition_key_value (str): The value of the GSI's partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
```

```
    table = dynamodb.Table(table_name)

    # Perform the query on the GSI
    response = table.query(
        IndexName=index_name,
 KeyConditionExpression=Key(partition_key_name).eq(partition_key_value)
    )

    return response
```

Consulta com paginação usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_pagination(
    table_name, partition_key_name, partition_key_value, page_size=25,
 max_pages=None
):
    """
    Query a DynamoDB table with pagination to handle large result sets.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page. Defaults
 to 25.
        max_pages (int, optional): The maximum number of pages to retrieve. If None,
 retrieves all pages.

    Returns:
        list: All items retrieved from the query across all pages.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
```

```
    last_evaluated_key = None
    page_count = 0
    all_items = []

    # Paginate through the results
    while True:
        # Check if we've reached the maximum number of pages
        if max_pages is not None and page_count >= max_pages:
            break

        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Process the current page of results
        items = response.get("Items", [])
        all_items.extend(items)

        # Update pagination tracking
        page_count += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if not last_evaluated_key:
            break

    return all_items


def query_with_pagination_generator(
    table_name, partition_key_name, partition_key_value, page_size=25
```

```
):
    """
    Query a DynamoDB table with pagination using a generator to handle large result
 sets.
    This approach is memory-efficient as it yields one page at a time.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page. Defaults
 to 25.

    Yields:
        tuple: A tuple containing (items, page_number, last_page) where:
            - items is a list of items for the current page
            - page_number is the current page number (starting from 1)
            - last_page is a boolean indicating if this is the last page
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_number = 0

    # Paginate through the results
    while True:
        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)
```

```
        # Get the current page of results
        items = response.get("Items", [])
        page_number += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # Determine if this is the last page
        is_last_page = last_evaluated_key is None

        # Yield the current page of results
        yield (items, page_number, is_last_page)

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if is_last_page:
            break
```

Consulte com filtros complexos usando AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_complex_filter(
    table_name,
    partition_key_name,
    partition_key_value,
    min_rating=None,
    status_list=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        min_rating (float, optional): Minimum rating value for filtering.
        status_list (list, optional): List of status values to include.
```

```
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize the filter expression and expression attribute values
    filter_expression = None
    expression_attribute_values = {}

    # Build the filter expression based on provided parameters
    if min_rating is not None:
        filter_expression = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if status_list and len(status_list) > 0:
        status_condition = None
        for i, status in enumerate(status_list):
            status_value_name = f":status{i}"
            expression_attribute_values[status_value_name] = status

            if status_condition is None:
                status_condition = Attr("status").eq(status)
            else:
                status_condition = status_condition | Attr("status").eq(status)

        if filter_expression is None:
            filter_expression = status_condition
        else:
            filter_expression = filter_expression & status_condition

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if filter_expression is None:
            filter_expression = price_condition
        else:
```

```
            filter_expression = filter_expression & price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response


def query_with_complex_filter_and_or(
    table_name,
    partition_key_name,
    partition_key_value,
    category=None,
    min_rating=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression using AND and OR
 operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        category (str, optional): Category value for filtering.
        min_rating (float, optional): Minimum rating value for filtering.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)
```

```
    # Build a complex filter expression with AND and OR operators
    filter_expression = None
    expression_attribute_values = {}

    # Build the category condition
    if category:
        filter_expression = Attr("category").eq(category)
        expression_attribute_values[":category"] = category

    # Build the rating and price condition (rating >= min_rating OR price <=
max_price)
    rating_price_condition = None

    if min_rating is not None:
        rating_price_condition = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if rating_price_condition is None:
            rating_price_condition = price_condition
        else:
            rating_price_condition = rating_price_condition | price_condition

    # Combine the conditions
    if rating_price_condition:
        if filter_expression is None:
            filter_expression = rating_price_condition
        else:
            filter_expression = filter_expression & rating_price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
```

```
    return response
```

Consulte com uma expressão de filtro construída dinamicamente usando AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_dynamic_filter(
    table_name, partition_key_name, partition_key_value, filter_conditions=None
):
    """
    Query a DynamoDB table with a dynamically constructed filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_conditions (dict, optional): A dictionary of filter conditions where
            keys are attribute names and values are dictionaries with 'operator' and
 'value'.
            Example: {'rating': {'operator': '>=', 'value': 4}, 'status':
 {'operator': '=', 'value': 'active'}}

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize variables for the filter expression and attribute values
    filter_expression = None
    expression_attribute_values = {":pk_val": partition_key_value}

    # Dynamically build the filter expression if filter conditions are provided
```

```
    if filter_conditions:
        for attr_name, condition in filter_conditions.items():
            operator = condition.get("operator")
            value = condition.get("value")
            attr_value_name = f":{attr_name}"
            expression_attribute_values[attr_value_name] = value

            # Create the appropriate filter expression based on the operator
            current_condition = None
            if operator == "=":
                current_condition = Attr(attr_name).eq(value)
            elif operator == "!=":
                current_condition = Attr(attr_name).ne(value)
            elif operator == ">":
                current_condition = Attr(attr_name).gt(value)
            elif operator == ">=":
                current_condition = Attr(attr_name).gte(value)
            elif operator == "<":
                current_condition = Attr(attr_name).lt(value)
            elif operator == "<=":
                current_condition = Attr(attr_name).lte(value)
            elif operator == "contains":
                current_condition = Attr(attr_name).contains(value)
            elif operator == "begins_with":
                current_condition = Attr(attr_name).begins_with(value)

            # Combine with existing filter expression using AND
            if current_condition:
                if filter_expression is None:
                    filter_expression = current_condition
                else:
                    filter_expression = filter_expression & current_condition

    # Perform the query with the dynamically built filter expression
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression

    response = table.query(**query_params)
    return response
```

Consulta com uma expressão de filtro e limite de uso AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_filter_and_limit(
    table_name,
    partition_key_name,
    partition_key_value,
    filter_attribute=None,
    filter_value=None,
    limit=10,
):
    """
    Query a DynamoDB table with a filter expression and limit the number of results.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute (str, optional): The attribute name to filter on.
        filter_value (any, optional): The value to compare against in the filter.
        limit (int, optional): The maximum number of items to evaluate. Defaults to
 10.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition, "Limit": limit}

    # Add the filter expression if filter attributes are provided
    if filter_attribute and filter_value is not None:
```

```
        query_params["FilterExpression"] = Attr(filter_attribute).gt(filter_value)
        query_params["ExpressionAttributeValues"] = {":filter_value": filter_value}

    # Execute the query
    response = table.query(**query_params)
    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Executar operações de lista

O exemplo de código a seguir mostra como realizar operações de lista no DynamoDB.

- Adicione elementos a um atributo de lista.
- Remova elementos de um atributo de lista.
- Atualize elementos específicos em uma lista por índice.
- Use as funções list.append e list.index.

SDK para Python (Boto3)

Demonstre as operações de lista usando AWS SDK para Python (Boto3).

```
import boto3
import json
from typing import Any, Dict, List, Optional, Union


def create_list_attribute(
    table_name: str, key: Dict[str, Any], list_name: str, list_values: List[Any]
) -> Dict[str, Any]:
    """
    Create a new list attribute or replace an existing one.

    This function demonstrates how to create a new list attribute or replace
    an existing list with new values.

    Args:
        table_name (str): The name of the DynamoDB table.
```

```
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        list_values (List[Any]): The values to set in the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the SET operation to create or replace the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = :list_values",
        ExpressionAttributeValues={":list_values": list_values},
        ReturnValues="UPDATED_NEW",
    )

    return response


def append_to_list(
    table_name: str, key: Dict[str, Any], list_name: str, values_to_append:
 List[Any]
) -> Dict[str, Any]:
    """
    Append values to the end of a list attribute.

    This function demonstrates how to use the list_append function to add elements
    to the end of a list attribute.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_append (List[Any]): The values to append to the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
```

```
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use list_append to add values to the end of the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = list_append({list_name}, :values)",
        ExpressionAttributeValues={":values": values_to_append},
        ReturnValues="UPDATED_NEW",
    )

    return response


def prepend_to_list(
    table_name: str, key: Dict[str, Any], list_name: str, values_to_prepend:
 List[Any]
) -> Dict[str, Any]:
    """
    Prepend values to the beginning of a list attribute.

    This function demonstrates how to use the list_append function to add elements
    to the beginning of a list attribute.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_prepend (List[Any]): The values to prepend to the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use list_append with reversed order to add values to the beginning of the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = list_append(:values, {list_name})",
        ExpressionAttributeValues={":values": values_to_prepend},
        ReturnValues="UPDATED_NEW",
```

```
    )

    return response


def update_list_element(
    table_name: str, key: Dict[str, Any], list_name: str, index: int, new_value: Any
) -> Dict[str, Any]:
    """
    Update a specific element in a list attribute.

    This function demonstrates how to update a specific element in a list attribute
    using the index notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        index (int): The zero-based index of the element to update.
        new_value (Any): The new value for the element.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the index notation to update a specific element
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name}[{index}] = :value",
        ExpressionAttributeValues={":value": new_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def remove_list_element(
    table_name: str, key: Dict[str, Any], list_name: str, index: int
) -> Dict[str, Any]:
    """
```

```
    Remove a specific element from a list attribute.

    This function demonstrates how to remove a specific element from a list
 attribute
    using the REMOVE action with index notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        index (int): The zero-based index of the element to remove.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the REMOVE action with index notation to remove a specific element
    response = table.update_item(
        Key=key, UpdateExpression=f"REMOVE {list_name}[{index}]",
 ReturnValues="UPDATED_NEW"
    )

    return response


def update_nested_list_element(
    table_name: str, key: Dict[str, Any], path: str, new_value: Any
) -> Dict[str, Any]:
    """
    Update an element in a nested list structure.

    This function demonstrates how to update an element in a nested list structure
    using expression attribute names for the path components.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path (str): The path to the nested element (e.g., "parent[0].child[1]").
        new_value (Any): The new value for the element.
```

```
    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
values.
    """
    # Define a type for path parts
    path_part = Dict[str, Union[str, int]]
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Parse the path to extract attribute names and indices
    path_parts: List[path_part] = []
    current_part = ""
    in_bracket = False

    for char in path:
        if char == "[":
            if current_part:
                path_parts.append({"type": "attribute", "value": current_part})
                current_part = ""
            in_bracket = True
        elif char == "]":
            if current_part:
                # Fix for mypy: Use a properly typed dictionary with Union type
                path_parts.append({"type": "index", "value": int(current_part)})
                current_part = ""
            in_bracket = False
        elif char == "." and not in_bracket:
            if current_part:
                path_parts.append({"type": "attribute", "value": current_part})
                current_part = ""
        else:
            current_part += char

    if current_part:
        path_parts.append({"type": "attribute", "value": current_part})

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
```

```
        if part["type"] == "attribute":
            name_placeholder = f"#attr{i}"
            expression_attribute_names[name_placeholder] = part["value"]

            if path_expression:
                path_expression += "."
            path_expression += name_placeholder
        elif part["type"] == "index":
            path_expression += f"[{part['value']}]"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": new_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def create_list_if_not_exists(
    table_name: str, key: Dict[str, Any], list_name: str, default_values: List[Any]
) -> Dict[str, Any]:
    """
    Create a list attribute if it doesn't exist.

    This function demonstrates how to use if_not_exists to create a list attribute
    with default values if it doesn't already exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        default_values (List[Any]): The default values for the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
```

```
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use if_not_exists to create the list if it doesn't exist
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = if_not_exists({list_name}, :default)",
        ExpressionAttributeValues={":default": default_values},
        ReturnValues="UPDATED_NEW",
    )

    return response


def append_to_list_safely(
    table_name: str,
    key: Dict[str, Any],
    list_name: str,
    values_to_append: List[Any],
    default_values: Optional[List[Any]] = None,
) -> Dict[str, Any]:
    """
    Append values to a list, creating it if it doesn't exist.

    This function demonstrates how to safely append values to a list attribute,
    creating the list with default values if it doesn't exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_append (List[Any]): The values to append to the list.
        default_values (Optional[List[Any]]): The default values if the list doesn't
 exist.
            If not provided, values_to_append will be used as the default.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # If default_values is not provided, use values_to_append
    if default_values is None:
        default_values = values_to_append

    # Use if_not_exists with list_append to safely append to the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} =
 list_append(if_not_exists({list_name}, :default), :values)",
        ExpressionAttributeValues={
            ":default": default_values if default_values else [],
            ":values": values_to_append,
        },
        ReturnValues="UPDATED_NEW",
    )

    return response
```

Exemplo de uso de operações de lista com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use list operations in DynamoDB."""
    # Example parameters
    table_name = "UserData"
    key = {"UserId": "user123"}

    print("Example 1: Creating a list attribute")
    try:
        response = create_list_attribute(
            table_name=table_name,
            key=key,
            list_name="Interests",
            list_values=["Reading", "Hiking", "Photography"],
        )
        print(
            f"List attribute created successfully:
 {json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
```

```
        print(f"Error creating list attribute: {e}")

    print("\nExample 2: Appending values to a list")
    try:
        response = append_to_list(
            table_name=table_name,
            key=key,
            list_name="Interests",
            values_to_append=["Cooking", "Gardening"],
        )
        print(
            f"Values appended to list successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error appending to list: {e}")

    print("\nExample 3: Prepending values to a list")
    try:
        response = prepend_to_list(
            table_name=table_name,
            key=key,
            list_name="Interests",
            values_to_prepend=["Travel", "Music"],
        )
        print(
            f"Values prepended to list successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error prepending to list: {e}")

    print("\nExample 4: Updating a specific list element")
    try:
        response = update_list_element(
            table_name=table_name,
            key=key,
            list_name="Interests",
            index=2,
            new_value="Mountain Hiking",
        )
        print(
            f"List element updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
```

```
        )
    except Exception as e:
        print(f"Error updating list element: {e}")

    print("\nExample 5: Removing a list element")
    try:
        response = remove_list_element(
            table_name=table_name, key=key, list_name="Interests", index=0
        )
        print(
            f"List element removed successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error removing list element: {e}")

    print("\nExample 6: Working with nested lists")
    try:
        # First, create an item with a nested structure
        dynamodb = boto3.resource("dynamodb")
        table = dynamodb.Table(table_name)

        table.update_item(
            Key={"UserId": "user456"},
            UpdateExpression="SET #skills = :skills",
            ExpressionAttributeNames={"#skills": "Skills"},
            ExpressionAttributeValues={
                ":skills": [
                    {"Category": "Programming", "Languages": ["Python", "Java",
"JavaScript"]},
                    {"Category": "Database", "Systems": ["DynamoDB", "MongoDB",
"PostgreSQL"]},
                ]
            },
        )

        # Now update a nested element
        response = update_nested_list_element(
            table_name=table_name,
            key={"UserId": "user456"},
            path="Skills[0].Languages[1]",
            new_value="TypeScript",
        )
        print(
```

```
            f"Nested list element updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error working with nested lists: {e}")

    print("\nExample 7: Creating a list if it doesn't exist")
    try:
        response = create_list_if_not_exists(
            table_name=table_name,
            key={"UserId": "user789"},
            list_name="Preferences",
            default_values=["Default1", "Default2", "Default3"],
        )
        print(
            f"List created with default values:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error creating list with default values: {e}")

    print("\nExample 8: Safely appending to a list")
    try:
        response = append_to_list_safely(
            table_name=table_name,
            key={"UserId": "user789"},
            list_name="Notifications",
            values_to_append=["New message received"],
            default_values=[],
        )
        print(f"Safely appended to list: {json.dumps(response.get('Attributes', {}),
default=str)}")
    except Exception as e:
        print(f"Error safely appending to list: {e}")

    print("\nKey Points About Working with Lists in DynamoDB:")
    print("1. Lists are ordered collections of elements that can be of different
types")
    print("2. Use the SET operation with direct assignment to create or replace a
list")
    print("3. Use list_append() to add elements to a list without replacing the
entire list")
    print("4. To append to the end: list_append(list_name, :values)")
    print("5. To prepend to the beginning: list_append(:values, list_name)")
```

```
    print("6. Use index notation list_name[index] to access or update specific
 elements")
    print("7. Use the REMOVE action with index notation to remove specific
 elements")
    print("8. Lists can contain nested structures like maps and other lists")
    print("9. Use if_not_exists() to create a list with default values if it doesn't
 exist")
    print("10. List indices are zero-based (the first element is at index 0)")
    print("11. Attempting to access an index beyond the list bounds will result in
 an error")
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

Executar operações de mapa

O exemplo de código a seguir mostra como realizar operações de mapa no DynamoDB.

- Adicione e atualize atributos aninhados em estruturas de mapa.
- Remova campos específicos dos mapas.
- Trabalhe com atributos de mapa profundamente aninhados.

SDK para Python (Boto3)

Demonstre as operações do mapa usando AWS SDK para Python (Boto3).

```
"""
Example of updating map attributes in DynamoDB.

This module demonstrates how to update map attributes in DynamoDB, including
handling cases where the map attribute might not exist yet.
"""


import boto3
from typing import Any, Dict, Optional
```

```
def update_map_attribute_safe(
    table_name: str, key: Dict[str, Any], map_name: str, map_key: str, value: Any
) -> Dict[str, Any]:
    """
    Update a specific key in a map attribute, creating the map if it doesn't exist.

    This function demonstrates how to safely update a key within a map attribute,
    even if the map doesn't exist yet in the item.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        map_key (str): The key within the map to update.
        value (Any): The value to set for the map key.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use SET with attribute_not_exists to safely update the map
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #map.#key = :value",
        ExpressionAttributeNames={"#map": map_name, "#key": map_key},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def add_to_nested_map(
    table_name: str, key: Dict[str, Any], path: str, value: Any
) -> Dict[str, Any]:
    """
    Add or update a value in a deeply nested map structure.
```

```
    This function demonstrates how to update a value at a specific path in a
    nested map structure, creating any intermediate maps as needed.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path (str): The path to the nested attribute (e.g.,
"user.preferences.theme").
        value (Any): The value to set at the specified path.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Split the path into components
    path_parts = path.split(".")

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        name_placeholder = f"#attr{i}"
        expression_attribute_names[name_placeholder] = part

        if i == 0:
            path_expression = name_placeholder
        else:
            path_expression += f".{name_placeholder}"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
```

```
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_map_with_if_not_exists(
    table_name: str,
    key: Dict[str, Any],
    map_name: str,
    map_key: str,
    value: Any,
    default_map: Optional[Dict[str, Any]] = None,
) -> Dict[str, Any]:
    """
    Update a key in a map, creating the map with default values if it doesn't exist.

    This function demonstrates how to use if_not_exists to initialize a map with
    default values if it doesn't exist yet, and then update a specific key.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        map_key (str): The key within the map to update.
        value (Any): The value to set for the map key.
        default_map (Optional[Dict[str, Any]]): Default map values if the map
 doesn't exist.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Set default map if not provided
    if default_map is None:
        default_map = {}

    # Create a map with the new key-value pair
    updated_map = default_map.copy()
```

```
    updated_map[map_key] = value

    # Use if_not_exists to initialize the map if it doesn't exist
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #map = if_not_exists(#map, :default_map)",
        ExpressionAttributeNames={"#map": map_name},
        ExpressionAttributeValues={":default_map": updated_map},
        ReturnValues="UPDATED_NEW",
    )

    return response


def merge_into_map(
    table_name: str, key: Dict[str, Any], map_name: str, values_to_merge: Dict[str,
 Any]
) -> Dict[str, Any]:
    """
    Merge multiple key-value pairs into a map attribute.

    This function demonstrates how to update multiple keys in a map attribute
    in a single operation, without overwriting the entire map.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        values_to_merge (Dict[str, Any]): Key-value pairs to merge into the map.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the update expression for each key-value pair
    update_expression = "SET "
    expression_attribute_names = {"#map": map_name}
    expression_attribute_values = {}

    # Add each key-value pair to the update expression
```

```
    for i, (k, v) in enumerate(values_to_merge.items()):
        key_placeholder = f"#key{i}"
        value_placeholder = f":value{i}"

        expression_attribute_names[key_placeholder] = k
        expression_attribute_values[value_placeholder] = v

        if i > 0:
            update_expression += ", "
        update_expression += f"#map.{key_placeholder} = {value_placeholder}"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues="UPDATED_NEW",
    )

    return response




def example_usage():
    """Example of how to use the map attribute update functions."""
    # Example parameters
    table_name = "UserProfiles"
    key = {"UserId": "user123"}

    print("Example 1: Updating a specific key in a map attribute")
    try:
        response = update_map_attribute_safe(
            table_name=table_name, key=key, map_name="Preferences", map_key="Theme",
 value="Dark"
        )
        print(f"Map attribute updated successfully: {response.get('Attributes',
 {})}")
    except Exception as e:
        print(f"Error updating map attribute: {e}")

    print("\nExample 2: Adding a value to a deeply nested map")
    try:
```

```
        response = add_to_nested_map(
            table_name=table_name, key=key, path="Settings.Notifications.Email",
value=True
        )
        print(f"Nested map updated successfully: {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating nested map: {e}")

    print("\nExample 3: Initializing a map with default values if it doesn't exist")
    try:
        default_map = {"Language": "English", "Currency": "USD", "TimeZone": "UTC"}

        response = update_map_with_if_not_exists(
            table_name=table_name,
            key={"UserId": "newuser456"},
            map_name="Preferences",
            map_key="Theme",
            value="Light",
            default_map=default_map,
        )
        print(f"Map initialized with defaults: {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error initializing map: {e}")

    print("\nExample 4: Merging multiple values into a map")
    try:
        values_to_merge = {
            "NotificationsEnabled": True,
            "EmailFrequency": "Daily",
            "PushNotifications": False,
        }

        response = merge_into_map(
            table_name=table_name,
            key=key,
            map_name="NotificationSettings",
            values_to_merge=values_to_merge,
        )
        print(f"Multiple values merged into map: {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error merging values into map: {e}")

    print("\nBest practices for working with map attributes in DynamoDB:")
    print("1. Use dot notation to access and update nested attributes")
```

```
    print("2. Use ExpressionAttributeNames to handle reserved words and special
 characters")
    print("3. Use if_not_exists() to handle cases where attributes might not exist")
    print("4. Update specific map keys rather than overwriting the entire map")
    print("5. Use a single update operation to modify multiple map keys for better
 performance")
    print("6. Consider your data model carefully to minimize the need for deeply
 nested attributes")



if __name__ == "__main__":
    example_usage()
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

Executar operações de conjunto

O exemplo de código a seguir mostra como realizar operações de conjunto no DynamoDB.

- Adicione elementos a um atributo de conjunto.
- Remova elementos de um atributo de conjunto.
- Use as operações ADD e DELETE com conjuntos.

SDK para Python (Boto3)

Demonstre as operações de conjunto usando AWS SDK para Python (Boto3).

```
import boto3
from typing import Any, Dict, List


def create_set_attribute(
    table_name: str,
    key: Dict[str, Any],
    set_name: str,
    set_values: List[Any],
    set_type: str = "string",
) -> Dict[str, Any]:
```

```
    """
    Create a new set attribute or add elements to an existing set.

    This function demonstrates how to use the ADD operation to create a new set
    or add elements to an existing set.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        set_values (List[Any]): The values to add to the set.
        set_type (str, optional): The type of set to create: "string", "number", or
"binary".
            Defaults to "string".

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a DynamoDB set based on the specified type
    if set_type == "string":
        dynamo_set = set(str(value) for value in set_values)
    elif set_type == "number":
        # We need to use actual float values for the DynamoDB API
        # but mypy expects strings in sets, so we need to use type: ignore
        dynamo_set = set(float(value) for value in set_values)  # type: ignore
    else:  # binary set is not directly supported in high-level API, handled
differently
        raise ValueError("Binary sets are not supported in this example")

    # Use the ADD operation to create or update the set
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #set_attr :set_values",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":set_values": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response
```

```
def add_to_set(
    table_name: str, key: Dict[str, Any], set_name: str, values_to_add: List[Any]
) -> Dict[str, Any]:
    """
    Add elements to an existing set attribute.

    This function demonstrates how to use the ADD operation to add elements to an
 existing set.
    If the set doesn't exist, it will be created.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        values_to_add (List[Any]): The values to add to the set.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a set (assuming string set for simplicity)
    dynamo_set = set(str(value) for value in values_to_add)

    # Use the ADD operation to add values to the set
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #set_attr :values_to_add",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":values_to_add": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def remove_from_set(
    table_name: str, key: Dict[str, Any], set_name: str, values_to_remove: List[Any]
) -> Dict[str, Any]:
```

```
    """
    Remove elements from a set attribute.

    This function demonstrates how to use the DELETE operation to remove elements
 from a set.
    If the last element is removed, the attribute will be deleted entirely.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        values_to_remove (List[Any]): The values to remove from the set.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a set (assuming string set for simplicity)
    dynamo_set = set(str(value) for value in values_to_remove)

    # Use the DELETE operation to remove values from the set
    response = table.update_item(
        Key=key,
        UpdateExpression="DELETE #set_attr :values_to_remove",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":values_to_remove": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def check_if_set_exists(table_name: str, key: Dict[str, Any], set_name: str) ->
 bool:
    """
    Check if a set attribute exists in an item.

    This function demonstrates how to check if a set attribute exists after
    potentially removing all elements from it.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to check.
        set_name (str): The name of the set attribute.

    Returns:
        bool: True if the set attribute exists, False otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Get the item
    response = table.get_item(
        Key=key, ProjectionExpression="#set_attr",
 ExpressionAttributeNames={"#set_attr": set_name}
    )

    # Check if the item exists and has the set attribute
    return "Item" in response and set_name in response["Item"]


def demonstrate_last_element_removal(
    table_name: str, key: Dict[str, Any], set_name: str
) -> Dict[str, Any]:
    """
    Demonstrate what happens when you remove the last element from a set.

    This function creates a set with a single element, then removes that element,
    showing that the attribute is completely removed when the last element is
 deleted.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Step 1: Create a set with a single element
    create_response = create_set_attribute(
        table_name=table_name,
        key=key,
```

```
        set_name=set_name,
        set_values=["last_element"],
        set_type="string",
    )

    # Step 2: Check that the set exists
    exists_before = check_if_set_exists(table_name, key, set_name)

    # Step 3: Remove the last element
    delete_response = remove_from_set(
        table_name=table_name, key=key, set_name=set_name,
 values_to_remove=["last_element"]
    )

    # Step 4: Check if the set still exists
    exists_after = check_if_set_exists(table_name, key, set_name)

    # Return the results
    return {
        "create_response": create_response,
        "exists_before": exists_before,
        "delete_response": delete_response,
        "exists_after": exists_after,
    }


def work_with_number_set(
    table_name: str,
    key: Dict[str, Any],
    set_name: str,
    initial_values: List[float],
    values_to_add: List[float],
    values_to_remove: List[float],
) -> Dict[str, Any]:
    """
    Demonstrate working with a number set in DynamoDB.

    This function shows how to create and manipulate a set of numbers.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        initial_values (List[float]): The initial values for the set.
```

```
        values_to_add (List[float]): Values to add to the set.
        values_to_remove (List[float]): Values to remove from the set.

    Returns:
        Dict[str, Any]: A dictionary containing the responses from each operation.
    """
    # Step 1: Create the number set
    create_response = create_set_attribute(
        table_name=table_name,
        key=key,
        set_name=set_name,
        set_values=initial_values,
        set_type="number",
    )

    # Step 2: Add more numbers to the set
    add_response = add_to_set(
        table_name=table_name, key=key, set_name=set_name,
values_to_add=values_to_add
    )

    # Step 3: Remove some numbers from the set
    remove_response = remove_from_set(
        table_name=table_name, key=key, set_name=set_name,
values_to_remove=values_to_remove
    )

    # Step 4: Get the final state
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    get_response = table.get_item(
        Key=key,
        ProjectionExpression=f"#{set_name}",
        ExpressionAttributeNames={f"#{set_name}": set_name},
    )

    # Return all responses
    return {
        "create_response": create_response,
        "add_response": add_response,
        "remove_response": remove_response,
        "final_state": get_response.get("Item", {}),
    }
```

Exemplo de uso de operações definidas com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the set operations functions."""
    # Example parameters
    table_name = "UserPreferences"
    key = {"UserId": "user123"}

    print("Example 1: Creating a string set attribute")
    try:
        response = create_set_attribute(
            table_name=table_name,
            key=key,
            set_name="FavoriteTags",
            set_values=["AWS", "DynamoDB", "NoSQL"],
            set_type="string",
        )
        print(f"Set attribute created successfully: {response.get('Attributes',
 {})}")
    except Exception as e:
        print(f"Error creating set attribute: {e}")

    print("\nExample 2: Adding elements to an existing set")
    try:
        response = add_to_set(
            table_name=table_name,
            key=key,
            set_name="FavoriteTags",
            values_to_add=["Database", "Serverless"],
        )
        print(f"Elements added to set successfully: {response.get('Attributes',
 {})}")
    except Exception as e:
        print(f"Error adding to set: {e}")

    print("\nExample 3: Removing elements from a set")
    try:
        response = remove_from_set(
```

```
            table_name=table_name, key=key, set_name="FavoriteTags",
values_to_remove=["NoSQL"]
        )
        print(f"Elements removed from set successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error removing from set: {e}")

    print("\nExample 4: Demonstrating what happens when you remove the last element
from a set")
    try:
        results = demonstrate_last_element_removal(
            table_name=table_name, key={"UserId": "tempUser"},
set_name="SingleElementSet"
        )

        print(f"Set exists before removal: {results['exists_before']}")
        print(f"Set exists after removal: {results['exists_after']}")

        if not results["exists_after"]:
            print("The set attribute was completely removed when the last element
was deleted.")
        else:
            print("The set attribute still exists after removing the last element.")
    except Exception as e:
        print(f"Error in last element removal demonstration: {e}")

    print("\nExample 5: Working with a number set")
    try:
        results = work_with_number_set(
            table_name=table_name,
            key={"UserId": "user456"},
            set_name="LuckyNumbers",
            initial_values=[7, 13, 42],
            values_to_add=[99, 100],
            values_to_remove=[13],
        )

        print(f"Initial number set: {results['create_response'].get('Attributes',
{})}")
        print(f"After adding numbers: {results['add_response'].get('Attributes',
{})}")
        print(f"After removing numbers:
{results['remove_response'].get('Attributes', {})}")
```

```
        print(f"Final state: {results['final_state']}")
    except Exception as e:
        print(f"Error working with number set: {e}")

    print("\nKey Points About DynamoDB Sets:")
    print("1. Sets can only contain elements of the same type (string, number, or
binary)")
    print("2. Sets automatically eliminate duplicate values")
    print("3. The ADD operation creates a set if it doesn't exist")
    print("4. The DELETE operation removes specified elements from a set")
    print("5. When the last element is removed from a set, the entire attribute is
deleted")
    print("6. Empty sets are not allowed in DynamoDB")
    print("7. Sets are unordered collections")
    print("8. The ADD operation is atomic for sets")
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que possa executar lotes de instruções PartiQL.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
 to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated with
                           each statement. This list must be in the same order as
 the
                           statements.
        :return: The responses returned from running the statements, if any.
        """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[
```

```
                    {"Statement": statement, "Parameters": params}
                    for statement, params in zip(statements, param_list)
                ]
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the table
 "
                    "does not exist."
                )
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why: %s:
 %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

Execute um cenário que crie uma tabela e execute consultas do PartiQL em lotes.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB PartiQL batch statement demo.")
    print("-" * 88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print("-" * 88)

    movie_data = [
        {
            "title": f"House PartiQL",
            "year": datetime.now().year - 5,
```

```
            "info": {
                "plot": "Wacky high jinks result from querying a mysterious
database.",
                "rating": Decimal("8.5"),
            },
        },
        {
            "title": f"House PartiQL 2",
            "year": datetime.now().year - 3,
            "info": {
                "plot": "Moderate high jinks result from querying another mysterious
database.",
                "rating": Decimal("6.5"),
            },
        },
        {
            "title": f"House PartiQL 3",
            "year": datetime.now().year - 1,
            "info": {
                "plot": "Tepid high jinks result from querying yet another
mysterious database.",
                "rating": Decimal("2.5"),
            },
        },
    ]

    print(f"Inserting a batch of movies into table '{table_name}.")
    statements = [
        f'INSERT INTO "{table_name}" ' f"VALUE {{'title': ?, 'year': ?, 'info': ?}}"
    ] * len(movie_data)
    params = [list(movie.values()) for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Getting data for a batch of movies.")
    statements = [f'SELECT * FROM "{table_name}" WHERE title=? AND year=?'] * len(
        movie_data
    )
    params = [[movie["title"], movie["year"]] for movie in movie_data]
    output = wrapper.run_partiql(statements, params)
    for item in output["Responses"]:
        print(f"\n{item['Item']['title']}, {item['Item']['year']}")
        pprint(item["Item"])
```

```
    print("-" * 88)

    ratings = [Decimal("7.7"), Decimal("5.5"), Decimal("1.3")]
    print(f"Updating a batch of movies with new ratings.")
    statements = [
        f'UPDATE "{table_name}" SET info.rating=? ' f"WHERE title=? AND year=?"
    ] * len(movie_data)
    params = [
        [rating, movie["title"], movie["year"]]
        for rating, movie in zip(ratings, movie_data)
    ]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Getting projected data from the table to verify our update.")
    output = wrapper.dyn_resource.meta.client.execute_statement(
        Statement=f'SELECT title, info.rating FROM "{table_name}"'
    )
    pprint(output["Items"])
    print("-" * 88)

    print(f"Deleting a batch of movies from the table.")
    statements = [f'DELETE FROM "{table_name}" WHERE title=? AND year=?'] * len(
        movie_data
    )
    params = [[movie["title"], movie["year"]] for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print("-" * 88)

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        dyn_res = boto3.resource("dynamodb")
        scaffold = Scaffold(dyn_res)
        movies = PartiQLBatchWrapper(dyn_res)
```

```
        run_scenario(scaffold, movies, "doc-example-table-partiql-movies")
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- Para obter detalhes da API, consulte a BatchExecuteStatement Referência da API AWS SDK for Python (Boto3).

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que possa executar instruções PartiQL.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLWrapper:
```

```
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                       statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not
exist.",
                    statement,
                )
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
        else:
            return output
```

Execute um cenário que crie uma tabela e execute consultas do PartiQL.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB PartiQL single statement demo.")
    print("-" * 88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print("-" * 88)

    title = "24 Hour PartiQL People"
    year = datetime.now().year
    plot = "A group of data developers discover a new query language they can't stop
 using."
    rating = Decimal("9.9")

    print(f"Inserting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f"INSERT INTO \"{table_name}\" VALUE {{'title': ?, 'year': ?, 'info': ?}}",
        [title, year, {"plot": plot, "rating": rating}],
    )
    print("Success!")
    print("-" * 88)

    print(f"Getting data for movie '{title}' released in {year}.")
    output = wrapper.run_partiql(
        f'SELECT * FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    for item in output["Items"]:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output["Items"])
    print("-" * 88)

    rating = Decimal("2.4")
```

```
    print(f"Updating movie '{title}' with a rating of {float(rating)}.")
    wrapper.run_partiql(
        f'UPDATE "{table_name}" SET info.rating=? WHERE title=? AND year=?',
        [rating, title, year],
    )
    print("Success!")
    print("-" * 88)

    print(f"Getting data again to verify our update.")
    output = wrapper.run_partiql(
        f'SELECT * FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    for item in output["Items"]:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output["Items"])
    print("-" * 88)

    print(f"Deleting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f'DELETE FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    print("Success!")
    print("-" * 88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print("-" * 88)

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        dyn_res = boto3.resource("dynamodb")
        scaffold = Scaffold(dyn_res)
        movies = PartiQLWrapper(dyn_res)
        run_scenario(scaffold, movies, "doc-example-table-partiql-movies")
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- Para obter detalhes da API, consulte a ExecuteStatement Referência da API AWS SDK for Python (Boto3).

Consultar uma tabela usando um índice secundário global

O exemplo de código a seguir mostra como consultar uma tabela usando um índice secundário global.

- Consulte uma tabela do DynamoDB usando a respectiva chave primária.
- Consulte um índice secundário global (GSI) para obter padrões de acesso alternativos.
- Compare consultas de tabela e consultas de GSI.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB usando sua chave primária e um Índice Secundário Global (GSI) com. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_table(table_name, partition_key_name, partition_key_value):
    """
    Query a DynamoDB table using its primary key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the table's primary key
    response =
 table.query(KeyConditionExpression=Key(partition_key_name).eq(partition_key_value))
```

```
    return response


def query_gsi(table_name, index_name, partition_key_name, partition_key_value):
    """
    Query a Global Secondary Index (GSI) on a DynamoDB table.

    Args:
        table_name (str): The name of the DynamoDB table.
        index_name (str): The name of the Global Secondary Index.
        partition_key_name (str): The name of the GSI's partition key attribute.
        partition_key_value (str): The value of the GSI's partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the GSI
    response = table.query(
        IndexName=index_name,
 KeyConditionExpression=Key(partition_key_name).eq(partition_key_value)
    )

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar uma tabela usando uma condição begins_with

O exemplo de código a seguir mostra como consultar uma tabela usando uma condição begins_with.

- Use a função begins_with em uma expressão de condição de chave.
- Filtre itens com base em um padrão de prefixo na chave de classificação.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB usando uma condição begins_with na chave de classificação por meio do AWS SDK para Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_begins_with(
    table_name, partition_key_name, partition_key_value, sort_key_name, prefix
):
    """
    Query a DynamoDB table with a begins_with condition on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute.
        prefix (str): The prefix to match at the beginning of the sort key.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query with a begins_with condition on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) & Key(
        sort_key_name
    ).begins_with(prefix)
    response = table.query(KeyConditionExpression=key_condition)

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consulte uma tabela usando um intervalo de datas

O exemplo de código a seguir mostra como consultar uma tabela usando um intervalo de datas na chave de classificação.

- Consulte itens em um intervalo de datas específico.
- Use operadores de comparação em chaves de classificação formatadas por data.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB em busca de itens dentro de um intervalo de datas com. AWS SDK para Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_date_range(
    table_name, partition_key_name, partition_key_value, sort_key_name, start_date,
 end_date
):
    """
    Query a DynamoDB table with a date range on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        start_date (datetime): The start date for the query range.
        end_date (datetime): The end date for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Format the date values as ISO 8601 strings
    # DynamoDB works well with ISO format for date values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key using BETWEEN operator
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response


def query_with_date_range_by_month(
    table_name, partition_key_name, partition_key_value, sort_key_name, year, month
):
    """
    Query a DynamoDB table for a specific month's data.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        year (int): The year to query.
        month (int): The month to query (1-12).

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Calculate the start and end dates for the specified month
    if month == 12:
        next_year = year + 1
        next_month = 1
    else:
        next_year = year
        next_month = month + 1

    start_date = datetime(year, month, 1)
    end_date = datetime(next_year, next_month, 1) - timedelta(microseconds=1)

    # Format the date values as ISO 8601 strings
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(KeyConditionExpression=key_condition)

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar uma tabela com uma expressão de filtro complexa

O exemplo de código a seguir mostra como consultar uma tabela com uma expressão de filtro complexa.

- Aplique expressões de filtro complexas aos resultados da consulta.
- Combine várias condições usando operadores lógicos.
- Filtre itens com base em atributos não chave.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com uma expressão de filtro complexa usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_complex_filter(
    table_name,
    partition_key_name,
    partition_key_value,
    min_rating=None,
    status_list=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        min_rating (float, optional): Minimum rating value for filtering.
        status_list (list, optional): List of status values to include.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize the filter expression and expression attribute values
    filter_expression = None
    expression_attribute_values = {}

    # Build the filter expression based on provided parameters
    if min_rating is not None:
```

```
        filter_expression = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if status_list and len(status_list) > 0:
        status_condition = None
        for i, status in enumerate(status_list):
            status_value_name = f":status{i}"
            expression_attribute_values[status_value_name] = status

            if status_condition is None:
                status_condition = Attr("status").eq(status)
            else:
                status_condition = status_condition | Attr("status").eq(status)

        if filter_expression is None:
            filter_expression = status_condition
        else:
            filter_expression = filter_expression & status_condition

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if filter_expression is None:
            filter_expression = price_condition
        else:
            filter_expression = filter_expression & price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response


def query_with_complex_filter_and_or(
    table_name,
    partition_key_name,
```

```
    partition_key_value,
    category=None,
    min_rating=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression using AND and OR
 operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        category (str, optional): Category value for filtering.
        min_rating (float, optional): Minimum rating value for filtering.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build a complex filter expression with AND and OR operators
    filter_expression = None
    expression_attribute_values = {}

    # Build the category condition
    if category:
        filter_expression = Attr("category").eq(category)
        expression_attribute_values[":category"] = category

    # Build the rating and price condition (rating >= min_rating OR price <=
 max_price)
    rating_price_condition = None

    if min_rating is not None:
        rating_price_condition = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating
```

```
    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if rating_price_condition is None:
            rating_price_condition = price_condition
        else:
            rating_price_condition = rating_price_condition | price_condition

    # Combine the conditions
    if rating_price_condition:
        if filter_expression is None:
            filter_expression = rating_price_condition
        else:
            filter_expression = filter_expression & rating_price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar uma tabela com uma expressão de filtro dinâmica

O exemplo de código a seguir mostra como consultar uma tabela com uma expressão de filtro dinâmica.

- Crie expressões de filtro dinamicamente no tempo de execução.
- Crie condições de filtro com base na entrada do usuário ou no estado da aplicação.
- Adicione ou remova critérios de filtro condicionalmente.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com uma expressão de filtro construída dinamicamente usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_dynamic_filter(
    table_name, partition_key_name, partition_key_value, filter_conditions=None
):
    """
    Query a DynamoDB table with a dynamically constructed filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_conditions (dict, optional): A dictionary of filter conditions where
            keys are attribute names and values are dictionaries with 'operator' and
 'value'.
            Example: {'rating': {'operator': '>=', 'value': 4}, 'status':
 {'operator': '=', 'value': 'active'}}

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize variables for the filter expression and attribute values
    filter_expression = None
    expression_attribute_values = {":pk_val": partition_key_value}

    # Dynamically build the filter expression if filter conditions are provided
    if filter_conditions:
        for attr_name, condition in filter_conditions.items():
            operator = condition.get("operator")
            value = condition.get("value")
```

```
            attr_value_name = f":{attr_name}"
            expression_attribute_values[attr_value_name] = value

            # Create the appropriate filter expression based on the operator
            current_condition = None
            if operator == "=":
                current_condition = Attr(attr_name).eq(value)
            elif operator == "!=":
                current_condition = Attr(attr_name).ne(value)
            elif operator == ">":
                current_condition = Attr(attr_name).gt(value)
            elif operator == ">=":
                current_condition = Attr(attr_name).gte(value)
            elif operator == "<":
                current_condition = Attr(attr_name).lt(value)
            elif operator == "<=":
                current_condition = Attr(attr_name).lte(value)
            elif operator == "contains":
                current_condition = Attr(attr_name).contains(value)
            elif operator == "begins_with":
                current_condition = Attr(attr_name).begins_with(value)

            # Combine with existing filter expression using AND
            if current_condition:
                if filter_expression is None:
                    filter_expression = current_condition
                else:
                    filter_expression = filter_expression & current_condition

    # Perform the query with the dynamically built filter expression
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression

    response = table.query(**query_params)
    return response
```

Demonstra como usar expressões de filtro dinâmico com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the query_with_dynamic_filter function."""
    # Example parameters
    table_name = "Products"
    partition_key_name = "Category"
    partition_key_value = "Electronics"

    # Define dynamic filter conditions based on user input or runtime conditions
    user_min_rating = 4  # This could come from user input
    user_status_filter = "active"  # This could come from user input

    filter_conditions = {}

    # Only add conditions that are actually specified
    if user_min_rating is not None:
        filter_conditions["rating"] = {"operator": ">=", "value": user_min_rating}

    if user_status_filter:
        filter_conditions["status"] = {"operator": "=", "value": user_status_filter}

    print(
        f"Querying products in category '{partition_key_value}' with filter
 conditions: {filter_conditions}"
    )

    # Execute the query with dynamic filter
    response = query_with_dynamic_filter(
        table_name, partition_key_name, partition_key_value, filter_conditions
    )

    # Process the results
    items = response.get("Items", [])
    print(f"Found {len(items)} items")

    for item in items:
        print(f"Product: {item}")
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar uma tabela com uma expressão de filtro e limite

O exemplo de código a seguir mostra como consultar uma tabela com uma expressão de filtro e limite.

- Aplique expressões de filtro aos resultados da consulta com um limite nos itens avaliados.
- Entenda como o limite afeta os resultados da consulta filtrada.
- Controle o número máximo de itens processados em uma consulta.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com uma expressão de filtro e limite de uso. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_filter_and_limit(
    table_name,
    partition_key_name,
    partition_key_value,
    filter_attribute=None,
    filter_value=None,
    limit=10,
):
    """
    Query a DynamoDB table with a filter expression and limit the number of results.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute (str, optional): The attribute name to filter on.
        filter_value (any, optional): The value to compare against in the filter.
        limit (int, optional): The maximum number of items to evaluate. Defaults to
 10.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
```

```
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition, "Limit": limit}

    # Add the filter expression if filter attributes are provided
    if filter_attribute and filter_value is not None:
        query_params["FilterExpression"] = Attr(filter_attribute).gt(filter_value)
        query_params["ExpressionAttributeValues"] = {":filter_value": filter_value}

    # Execute the query
    response = table.query(**query_params)
    return response
```

Demonstra como usar expressões de filtro com limites em AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the query_with_filter_and_limit function."""
    # Example parameters
    table_name = "ProductReviews"
    partition_key_name = "ProductId"
    partition_key_value = "P123456"
    filter_attribute = "Rating"
    filter_value = 3  # Filter for ratings > 3
    limit = 5

    print(f"Querying reviews for product '{partition_key_value}' with rating >
 {filter_value}")
    print(f"Limiting to {limit} evaluated items")

    # Execute the query with filter and limit
    response = query_with_filter_and_limit(
        table_name, partition_key_name, partition_key_value, filter_attribute,
 filter_value, limit
    )
```

```
    # Process the results
    items = response.get("Items", [])
    print(f"\nReturned {len(items)} items that passed the filter")

    for item in items:
        print(f"Review: {item}")

    # Explain the difference between Limit and actual results
    explain_limit_vs_results(response)

    # Check if there are more results
    if "LastEvaluatedKey" in response:
        print("\nThere are more results available. Use the LastEvaluatedKey for
 pagination.")
    else:
        print("\nAll matching results have been retrieved.")
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar uma tabela com atributos aninhados

O exemplo de código a seguir mostra como consultar uma tabela com atributos aninhados.

- Acesse e filtre por atributos aninhados nos itens do DynamoDB.
- Use expressões de caminho de documento para se referir a elementos aninhados.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com atributos aninhados usando. AWS SDK para Python (Boto3)

```
from typing import Any, Dict, List

import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_nested_attributes(
    table_name: str,
```

```
    partition_key_name: str,
    partition_key_value: str,
    nested_path: str,
    comparison_operator: str,
    comparison_value: Any,
) -> Dict[str, Any]:
    """
    Query a DynamoDB table and filter by nested attributes.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        nested_path (str): The path to the nested attribute (e.g., 'specs.weight').
        comparison_operator (str): The comparison operator to use ('=', '!=', '<',
 '<=', '>', '>=').
        comparison_value (any): The value to compare against.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build the filter expression based on the nested attribute path and comparison
 operator
    filter_expression = None
    if comparison_operator == "=":
        filter_expression = Attr(nested_path).eq(comparison_value)
    elif comparison_operator == "!=":
        filter_expression = Attr(nested_path).ne(comparison_value)
    elif comparison_operator == "<":
        filter_expression = Attr(nested_path).lt(comparison_value)
    elif comparison_operator == "<=":
        filter_expression = Attr(nested_path).lte(comparison_value)
    elif comparison_operator == ">":
        filter_expression = Attr(nested_path).gt(comparison_value)
    elif comparison_operator == ">=":
        filter_expression = Attr(nested_path).gte(comparison_value)
    elif comparison_operator == "contains":
```

```
        filter_expression = Attr(nested_path).contains(comparison_value)
    elif comparison_operator == "begins_with":
        filter_expression = Attr(nested_path).begins_with(comparison_value)

    # Execute the query with the filter expression
    response = table.query(KeyConditionExpression=key_condition,
 FilterExpression=filter_expression)

    return response


def query_with_multiple_nested_attributes(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    nested_conditions: List[Dict[str, Any]],
) -> Dict[str, Any]:
    """
    Query a DynamoDB table and filter by multiple nested attributes.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        nested_conditions (list): A list of dictionaries, each containing:
            - path (str): The path to the nested attribute
            - operator (str): The comparison operator
            - value (any): The value to compare against

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build the combined filter expression for all nested attributes
    combined_filter = None

    for condition in nested_conditions:
        if not isinstance(condition, dict):
```

```
            continue
        path = condition.get("path", "")
        operator = condition.get("operator", "")
        value = condition.get("value")

        if not path or not operator:
            continue

        # Build the individual filter expression
        current_filter = None
        if operator == "=":
            current_filter = Attr(path).eq(value)
        elif operator == "!=":
            current_filter = Attr(path).ne(value)
        elif operator == "<":
            current_filter = Attr(path).lt(value)
        elif operator == "<=":
            current_filter = Attr(path).lte(value)
        elif operator == ">":
            current_filter = Attr(path).gt(value)
        elif operator == ">=":
            current_filter = Attr(path).gte(value)
        elif operator == "contains":
            current_filter = Attr(path).contains(value)
        elif operator == "begins_with":
            current_filter = Attr(path).begins_with(value)

        # Combine with the existing filter using AND
        if current_filter:
            if combined_filter is None:
                combined_filter = current_filter
            else:
                combined_filter = combined_filter & current_filter

    # Execute the query with the combined filter expression
    response = table.query(KeyConditionExpression=key_condition,
FilterExpression=combined_filter)

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar uma tabela com paginação

O exemplo de código a seguir mostra como consultar uma tabela com paginação.

- Implemente a paginação para os resultados da consulta do DynamoDB.
- Use o LastEvaluatedKey para recuperar as páginas subsequentes.
- Controle o número de itens por página com o parâmetro Limit.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com paginação usando. AWS SDK para Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_pagination(
    table_name, partition_key_name, partition_key_value, page_size=25,
 max_pages=None
):
    """
    Query a DynamoDB table with pagination to handle large result sets.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page. Defaults
 to 25.
        max_pages (int, optional): The maximum number of pages to retrieve. If None,
 retrieves all pages.

    Returns:
        list: All items retrieved from the query across all pages.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_count = 0
```

```
    all_items = []

    # Paginate through the results
    while True:
        # Check if we've reached the maximum number of pages
        if max_pages is not None and page_count >= max_pages:
            break

        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Process the current page of results
        items = response.get("Items", [])
        all_items.extend(items)

        # Update pagination tracking
        page_count += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if not last_evaluated_key:
            break

    return all_items


def query_with_pagination_generator(
    table_name, partition_key_name, partition_key_value, page_size=25
):
    """
```

```
    Query a DynamoDB table with pagination using a generator to handle large result
sets.
    This approach is memory-efficient as it yields one page at a time.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page. Defaults
to 25.

    Yields:
        tuple: A tuple containing (items, page_number, last_page) where:
            - items is a list of items for the current page
            - page_number is the current page number (starting from 1)
            - last_page is a boolean indicating if this is the last page
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_number = 0

    # Paginate through the results
    while True:
        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Get the current page of results
        items = response.get("Items", [])
```

```
        page_number += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # Determine if this is the last page
        is_last_page = last_evaluated_key is None

        # Yield the current page of results
        yield (items, page_number, is_last_page)

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if is_last_page:
            break
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar uma tabela com leituras altamente consistentes

O exemplo de código a seguir mostra como consultar uma tabela com leituras altamente consistentes.

- Configure o nível de consistência das consultas do DynamoDB.
- Use leituras altamente consistentes para obter os dados mais atualizados.
- Entenda as vantagens e desvantagens entre consistência final e consistência forte.

SDK para Python (Boto3)

Consulte uma tabela do DynamoDB com a opção de leituras fortemente consistentes usando. AWS SDK para Python (Boto3)

```
import time

import boto3
from boto3.dynamodb.conditions import Key
```

```
def query_with_consistent_read(
    table_name,
    partition_key_name,
    partition_key_value,
    sort_key_name=None,
    sort_key_value=None,
    consistent_read=True,
):
    """
    Query a DynamoDB table with the option for strongly consistent reads.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str, optional): The name of the sort key attribute.
        sort_key_value (str, optional): The value of the sort key to query.
        consistent_read (bool, optional): Whether to use strongly consistent reads.
 Defaults to True.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    if sort_key_name and sort_key_value:
        key_condition = key_condition & Key(sort_key_name).eq(sort_key_value)

    # Perform the query with the consistent read option
    response = table.query(KeyConditionExpression=key_condition,
 ConsistentRead=consistent_read)

    return response
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar itens com TTL

O exemplo de código a seguir mostra como consultar itens com TTL.

SDK para Python (Boto3)

Consulte a expressão filtrada para reunir itens TTL em uma tabela do DynamoDB usando. AWS SDK para Python (Boto3)

```
from datetime import datetime

import boto3


def query_dynamodb_items(table_name, partition_key):
    """

    :param table_name: Name of the DynamoDB table
    :param partition_key:
    :return:
    """
    try:
        # Initialize a DynamoDB resource
        dynamodb = boto3.resource("dynamodb", region_name="us-east-1")

        # Specify your table
        table = dynamodb.Table(table_name)

        # Get the current time in epoch format
        current_time = int(datetime.now().timestamp())

        # Perform the query operation with a filter expression to exclude expired
 items
        # response = table.query(
        #
 KeyConditionExpression=boto3.dynamodb.conditions.Key('partitionKey').eq(partition_key),
        #
 FilterExpression=boto3.dynamodb.conditions.Attr('expireAt').gt(current_time)
        # )
        response = table.query(

 KeyConditionExpression=dynamodb.conditions.Key("partitionKey").eq(partition_key),
            FilterExpression=dynamodb.conditions.Attr("expireAt").gt(current_time),
        )
```

```
        # Print the items that are not expired
        for item in response["Items"]:
            print(item)

    except Exception as e:
        print(f"Error querying items: {e}")


# Call the function with your values
query_dynamodb_items("Music", "your-partition-key-value")
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Consultar tabelas usando padrões de data e hora

O exemplo de código a seguir mostra como consultar tabelas usando padrões de data e hora.

- Armazene e consulte date/time valores no DynamoDB.
- Implemente consultas de intervalo de datas usando chaves de classificação.
- Formate strings de data para obter uma consulta eficaz.

SDK para Python (Boto3)

Consulte usando intervalos de datas em chaves de classificação com AWS SDK para Python (Boto3).

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_date_range(
    table_name, partition_key_name, partition_key_value, sort_key_name, start_date,
 end_date
):
    """
    Query a DynamoDB table with a date range on the sort key.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        start_date (datetime): The start date for the query range.
        end_date (datetime): The end date for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date values as ISO 8601 strings
    # DynamoDB works well with ISO format for date values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key using BETWEEN operator
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response


def query_with_date_range_by_month(
    table_name, partition_key_name, partition_key_value, sort_key_name, year, month
):
    """
    Query a DynamoDB table for a specific month's data.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
values).
        year (int): The year to query.
        month (int): The month to query (1-12).

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Calculate the start and end dates for the specified month
    if month == 12:
        next_year = year + 1
        next_month = 1
    else:
        next_year = year
        next_month = month + 1

    start_date = datetime(year, month, 1)
    end_date = datetime(next_year, next_month, 1) - timedelta(microseconds=1)

    # Format the date values as ISO 8601 strings
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(KeyConditionExpression=key_condition)

    return response
```

Consulte usando variáveis de data e hora com. AWS SDK para Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_datetime(
    table_name, partition_key_name, partition_key_value, sort_key_name, start_date,
 end_date
):
    """
    Query a DynamoDB table with a date range filter on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date/
time values).
        start_date (datetime): The start date/time for the query range.
        end_date (datetime): The end date/time for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date/time values as ISO 8601 strings
    # DynamoDB works well with ISO format for date/time values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )
```

```
    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response



def example_usage():
    """Example of how to use the query_with_datetime function."""
    # Example parameters
    table_name = "Events"
    partition_key_name = "EventType"
    partition_key_value = "UserLogin"
    sort_key_name = "Timestamp"

    # Create date/time variables for the query
    end_date = datetime.now()
    start_date = end_date - timedelta(days=7)  # Query events from the last 7 days

    print(f"Querying events from {start_date.isoformat()} to
 {end_date.isoformat()}")

    # Execute the query
    response = query_with_datetime(
        table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
    )

    # Process the results
    items = response.get("Items", [])
    print(f"Found {len(items)} items")

    for item in items:
        print(f"Event: {item}")
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Python (Boto3).

Como funciona a ordem da expressão de atualização

O exemplo de código a seguir mostra como funciona a ordem da expressão de atualização.

- Saiba como o DynamoDB processa expressões de atualização.
- Entenda como funciona a ordem das operações nas expressões de atualização.
- Evite resultados inesperados entendendo a avaliação da expressão.

SDK para Python (Boto3)

Demonstre a ordem das expressões de atualização usando AWS SDK para Python (Boto3).

```
import boto3
import json
from typing import Any, Dict, Optional


def update_with_multiple_actions(
    table_name: str,
    key: Dict[str, Any],
    update_expression: str,
    expression_attribute_names: Optional[Dict[str, str]] = None,
    expression_attribute_values: Optional[Dict[str, Any]] = None,
) -> Dict[str, Any]:
    """
    Update an item with multiple actions in a single update expression.

    This function demonstrates how to use multiple actions in a single update
 expression
    and how DynamoDB processes these actions.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        update_expression (str): The update expression with multiple actions.
        expression_attribute_names (Optional[Dict[str, str]]): Expression attribute
 name placeholders.
```

```
        expression_attribute_values (Optional[Dict[str, Any]]): Expression attribute
 value placeholders.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Prepare the update parameters
    update_params = {
        "Key": key,
        "UpdateExpression": update_expression,
        "ReturnValues": "UPDATED_NEW",
    }

    # Add expression attribute names if provided
    if expression_attribute_names:
        update_params["ExpressionAttributeNames"] = expression_attribute_names

    # Add expression attribute values if provided
    if expression_attribute_values:
        update_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the update
    response = table.update_item(**update_params)

    return response


def demonstrate_value_copying(table_name: str, key: Dict[str, Any]) -> Dict[str,
 Any]:
    """
    Demonstrate that variables hold copies of existing values before modifications.

    This function creates an item with initial values, then updates it with an
 expression
    that uses the values of attributes before they are modified in the same
 expression.

    Args:
        table_name (str): The name of the DynamoDB table.
```

```
        key (Dict[str, Any]): The primary key of the item to create and update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Step 1: Create an item with initial values
    initial_item = key.copy()
    initial_item.update({"a": 1, "b": 2, "c": 3})

    table.put_item(Item=initial_item)

    # Step 2: Get the item to verify initial state
    response_before = table.get_item(Key=key)
    item_before = response_before.get("Item", {})

    # Step 3: Update the item with an expression that uses values before they are
 modified
    # This expression removes 'a', then sets 'b' to the value of 'a', and 'c' to the
 value of 'b'
    update_response = table.update_item(
        Key=key, UpdateExpression="REMOVE a SET b = a, c = b",
 ReturnValues="UPDATED_NEW"
    )

    # Step 4: Get the item to verify final state
    response_after = table.get_item(Key=key)
    item_after = response_after.get("Item", {})

    # Return the results
    return {
        "initial_state": item_before,
        "update_response": update_response,
        "final_state": item_after,
    }


def demonstrate_action_order(table_name: str, key: Dict[str, Any]) -> Dict[str,
 Any]:
    """
    Demonstrate the order in which different action types are processed.
```

```
    This function creates an item with initial values, then updates it with an
expression
    that includes multiple action types (SET, REMOVE, ADD, DELETE) to show the order
    in which they are processed.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to create and update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Step 1: Create an item with initial values
    initial_item = key.copy()
    initial_item.update(
        {
            "counter": 10,
            "set_attr": set(["A", "B", "C"]),
            "to_remove": "This will be removed",
            "to_modify": "Original value",
        }
    )

    table.put_item(Item=initial_item)

    # Step 2: Get the item to verify initial state
    response_before = table.get_item(Key=key)
    item_before = response_before.get("Item", {})

    # Step 3: Update the item with multiple action types
    # The actions will be processed in this order: REMOVE, SET, ADD, DELETE
    update_response = table.update_item(
        Key=key,
        UpdateExpression="REMOVE to_remove SET to_modify = :new_value ADD
counter :increment DELETE set_attr :elements",
        ExpressionAttributeValues={
            ":new_value": "Updated value",
            ":increment": 5,
            ":elements": set(["B"]),
```

```
        },
        ReturnValues="UPDATED_NEW",
    )

    # Step 4: Get the item to verify final state
    response_after = table.get_item(Key=key)
    item_after = response_after.get("Item", {})

    # Return the results
    return {
        "initial_state": item_before,
        "update_response": update_response,
        "final_state": item_after,
    }


def update_with_multiple_set_actions(
    table_name: str, key: Dict[str, Any], attributes: Dict[str, Any]
) -> Dict[str, Any]:
    """
    Update multiple attributes with a single SET action.

    This function demonstrates how to update multiple attributes in a single SET
 action,
    which is more efficient than using multiple separate update operations.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        attributes (Dict[str, Any]): The attributes to update and their new values.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the update expression and expression attribute values
    update_expression = "SET "
    expression_attribute_values = {}

    # Add each attribute to the update expression
```

```
    for i, (attr_name, attr_value) in enumerate(attributes.items()):
        value_placeholder = f":val{i}"

        if i > 0:
            update_expression += ", "
        update_expression += f"{attr_name} = {value_placeholder}"

        expression_attribute_values[value_placeholder] = attr_value

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_with_conditional_value_copying(
    table_name: str,
    key: Dict[str, Any],
    source_attribute: str,
    target_attribute: str,
    default_value: Any,
) -> Dict[str, Any]:
    """
    Update an attribute with a value from another attribute or a default value.

    This function demonstrates how to use if_not_exists to conditionally copy a
 value
    from one attribute to another, or use a default value if the source doesn't
 exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        source_attribute (str): The attribute to copy the value from.
        target_attribute (str): The attribute to update.
        default_value (Any): The default value to use if the source attribute
 doesn't exist.

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use if_not_exists to conditionally copy the value
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {target_attribute} =
 if_not_exists({source_attribute}, :default)",
        ExpressionAttributeValues={":default": default_value},
        ReturnValues="UPDATED_NEW",
    )

    return response
```

Exemplo de uso da ordem de atualização da expressão com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use update expression order of operations in DynamoDB."""
    # Example parameters
    table_name = "OrderProcessing"
    key = {"OrderId": "order123"}

    print("Example 1: Demonstrating value copying in update expressions")
    try:
        results = demonstrate_value_copying(table_name=table_name, key=key)

        print(f"Initial state: {json.dumps(results['initial_state'], default=str)}")
        print(f"Update response: {json.dumps(results['update_response'],
 default=str)}")
        print(f"Final state: {json.dumps(results['final_state'], default=str)}")

        print("\nExplanation:")
        print("1. The initial state had a=1, b=2, c=3")
        print("2. The update expression 'REMOVE a SET b = a, c = b' did the
 following:")
        print("   - Copied the value of 'a' (which was 1) to be used for 'b'")
```

```
        print("   - Copied the value of 'b' (which was 2) to be used for 'c'")
        print("   - Removed the attribute 'a'")
        print("3. The final state has b=1, c=2, and 'a' is removed")
        print(
            "4. This demonstrates that DynamoDB uses the values of attributes as
they were BEFORE any modifications"
        )
    except Exception as e:
        print(f"Error demonstrating value copying: {e}")

    print("\nExample 2: Demonstrating the order of different action types")
    try:
        results = demonstrate_action_order(table_name=table_name, key={"OrderId":
"order456"})

        print(f"Initial state: {json.dumps(results['initial_state'], default=str)}")
        print(f"Update response: {json.dumps(results['update_response'],
default=str)}")
        print(f"Final state: {json.dumps(results['final_state'], default=str)}")

        print("\nExplanation:")
        print("1. The update expression contained multiple action types: REMOVE,
SET, ADD, DELETE")
        print("2. DynamoDB processes these actions in this order: REMOVE, SET, ADD,
DELETE")
        print("3. First, 'to_remove' was removed")
        print("4. Then, 'to_modify' was set to a new value")
        print("5. Next, 'counter' was incremented by 5")
        print("6. Finally, 'B' was removed from the set attribute")
    except Exception as e:
        print(f"Error demonstrating action order: {e}")

    print("\nExample 3: Updating multiple attributes in a single SET action")
    try:
        response = update_with_multiple_set_actions(
            table_name=table_name,
            key={"OrderId": "order789"},
            attributes={
                "Status": "Shipped",
                "ShippingDate": "2025-05-14",
                "TrackingNumber": "1Z999AA10123456784",
            },
        )
```

```
        print(
            f"Multiple attributes updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error updating multiple attributes: {e}")

    print("\nExample 4: Conditional value copying with if_not_exists")
    try:
        response = update_with_conditional_value_copying(
            table_name=table_name,
            key={"OrderId": "order101"},
            source_attribute="PreferredShippingMethod",
            target_attribute="ShippingMethod",
            default_value="Standard",
        )

        print(
            f"Conditional value copying result:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error with conditional value copying: {e}")

    print("\nKey Points About Update Expression Order of Operations:")
    print(
        "1. Variables in expressions hold copies of attribute values as they existed
BEFORE any modifications"
    )
    print(
        "2. Multiple actions in an update expression are processed in this order:
REMOVE, SET, ADD, DELETE"
    )
    print("3. Within each action type, operations are processed from left to right")
    print("4. You can reference the same attribute multiple times in an expression")
    print("5. You can use if_not_exists() to conditionally set values based on
attribute existence")
    print(
        "6. Using a single update expression with multiple actions is more efficient
than multiple separate updates"
    )
    print("7. The update expression is atomic - either all actions succeed or none
do")
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

Atualizar a configuração de throughput a quente de uma tabela

O exemplo de código a seguir mostra como atualizar a configuração de throughput a quente de uma tabela.

SDK para Python (Boto3)

Atualize a configuração de throughput a quente em uma tabela existente do DynamoDB usando o AWS SDK para Python (Boto3).

```
from boto3 import client
from botocore.exceptions import ClientError


def update_dynamodb_table_warm_throughput(
    table_name,
    table_read_units,
    table_write_units,
    gsi_name,
    gsi_read_units,
    gsi_write_units,
    region_name="us-east-1",
):
    """
    Updates the warm throughput of a DynamoDB table and a global secondary index.

    :param table_name: The name of the table to update.
    :param table_read_units: The new read units per second for the table's warm
 throughput.
    :param table_write_units: The new write units per second for the table's warm
 throughput.
    :param gsi_name: The name of the global secondary index to update.
    :param gsi_read_units: The new read units per second for the GSI's warm
 throughput.
    :param gsi_write_units: The new write units per second for the GSI's warm
 throughput.
```

```
    :param region_name: The AWS Region name to target. defaults to us-east-1
    :return: The response from the update_table operation
    """
    try:
        ddb = client("dynamodb", region_name=region_name)

        # Update the table's warm throughput
        table_warm_throughput = {
            "ReadUnitsPerSecond": table_read_units,
            "WriteUnitsPerSecond": table_write_units,
        }

        # Update the global secondary index's warm throughput
        gsi_warm_throughput = {
            "ReadUnitsPerSecond": gsi_read_units,
            "WriteUnitsPerSecond": gsi_write_units,
        }

        # Construct the global secondary index update
        global_secondary_index_update = [
            {"Update": {"IndexName": gsi_name, "WarmThroughput":
gsi_warm_throughput}}
        ]

        # Construct the update table request
        update_table_request = {
            "TableName": table_name,
            "GlobalSecondaryIndexUpdates": global_secondary_index_update,
            "WarmThroughput": table_warm_throughput,
        }

        # Update the table
        response = ddb.update_table(**update_table_request)
        print("Table updated successfully!")
        return response  # Make sure to return the response
    except ClientError as e:
        print(f"Error updating table: {e}")
        raise e
```

- Para obter detalhes da API, consulte a UpdateTable Referência da API AWS SDK for Python (Boto3).

Atualiza a TTL de um item

O exemplo de código a seguir mostra como atualizar a TTL de um item.

SDK para Python (Boto3)

```
from datetime import datetime, timedelta

import boto3


def update_dynamodb_item(table_name, region, primary_key, sort_key):
    """
    Update an existing DynamoDB item with a TTL.
    :param table_name: Name of the DynamoDB table
    :param region: AWS Region of the table - example `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :return: Void (nothing)
    """
    try:
        # Create the DynamoDB resource.
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Get the current time in epoch second format
        current_time = int(datetime.now().timestamp())

        # Calculate the expireAt time (90 days from now) in epoch second format
        expire_at = int((datetime.now() + timedelta(days=90)).timestamp())

        table.update_item(
            Key={"partitionKey": primary_key, "sortKey": sort_key},
            UpdateExpression="set updatedAt=:c, expireAt=:e",
            ExpressionAttributeValues={":c": current_time, ":e": expire_at},
        )

        print("Item updated successfully.")
    except Exception as e:
        print(f"Error updating item: {e}")


# Replace with your own values
update_dynamodb_item(
```

```
    "your-table-name", "us-west-2", "your-partition-key-value", "your-sort-key-
value"
)
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para Python (Boto3)

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.
- Use o pacote Requests para enviar solicitações à API REST.
- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar operações de contador atômico

O exemplo de código a seguir mostra como usar operações de contador atômico no DynamoDB.

- Incremente os contadores atomicamente usando as operações ADD e SET.
- Incremente seguramente contadores que talvez não existam.
- Implemente um bloqueio positivo para operações de contador.

SDK para Python (Boto3)

Demonstre operações de contador atômico usando AWS SDK para Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, Union


def increment_counter_with_add(
    table_name: str, key: Dict[str, Any], counter_name: str, increment_value: int =
 1
) -> Dict[str, Any]:
    """
    Increment a counter attribute using the ADD operation.

    This function demonstrates the atomic ADD operation, which is ideal for
    incrementing counters without the risk of race conditions.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to 1.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the ADD operation to atomically increment the counter
```

```
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #counter :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def increment_counter_with_set(
    table_name: str, key: Dict[str, Any], counter_name: str, increment_value: int =
 1
) -> Dict[str, Any]:
    """
    Increment a counter attribute using the SET operation with an expression.

    This function demonstrates using SET with an expression to increment a counter.
    While this works, it's generally recommended to use ADD for simple increments.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to 1.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the SET operation with an expression to increment the counter
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #counter = #counter + :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value},
        ReturnValues="UPDATED_NEW",
    )
```

```
    return response


def increment_counter_safely(
    table_name: str,
    key: Dict[str, Any],
    counter_name: str,
    increment_value: int = 1,
    initial_value: int = 0,
) -> Dict[str, Any]:
    """
    Increment a counter attribute safely, handling the case where it might not
 exist.

    This function demonstrates a best practice for incrementing counters by using
    the if_not_exists function to handle the case where the counter doesn't exist
 yet.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to 1.
        initial_value (int, optional): The initial value if the counter doesn't
 exist. Defaults to 0.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use SET with if_not_exists to safely increment the counter
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #counter = if_not_exists(#counter, :initial)
 + :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value, ":initial":
 initial_value},
        ReturnValues="UPDATED_NEW",
    )
```

```
    return response


def atomic_conditional_increment(
    table_name: str,
    key: Dict[str, Any],
    counter_name: str,
    condition_attribute: str,
    condition_value: Any,
    increment_value: int = 1,
) -> Union[Dict[str, Any], None]:
    """
    Atomically increment a counter only if a condition is met.

    This function demonstrates combining atomic counter operations with
    conditional expressions for more complex update scenarios.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.
        increment_value (int, optional): The value to increment by. Defaults to 1.

    Returns:
        Optional[Dict[str, Any]]: The response from DynamoDB if successful, None if
 condition failed.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        # Use ADD with a condition expression
        response = table.update_item(
            Key=key,
            UpdateExpression="ADD #counter :increment",
            ConditionExpression="#condition = :value",
            ExpressionAttributeNames={"#counter": counter_name, "#condition":
 condition_attribute},
            ExpressionAttributeValues={":increment": increment_value, ":value":
 condition_value},
```

```
            ReturnValues="UPDATED_NEW",
        )
        return response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return None
        else:
            # Other error occurred
            raise
```

Exemplo de uso de operações de contador atômico com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the atomic counter operations functions."""
    # Example parameters
    table_name = "GameScores"
    key = {"UserId": "user123", "GameId": "game456"}
    counter_name = "Score"

    print("Example 1: Incrementing a counter with ADD operation")
    try:
        response = increment_counter_with_add(
            table_name=table_name, key=key, counter_name=counter_name,
 increment_value=10
        )
        print(
            f"Counter incremented successfully. New value:
 {response.get('Attributes', {}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error incrementing counter with ADD: {e}")

    print("\nExample 2: Incrementing a counter with SET operation")
    try:
        response = increment_counter_with_set(
            table_name=table_name, key=key, counter_name=counter_name,
 increment_value=5
        )
        print(
```

```
            f"Counter incremented successfully. New value:
{response.get('Attributes', {}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error incrementing counter with SET: {e}")

    print("\nExample 3: Safely incrementing a counter that might not exist")
    try:
        new_key = {"UserId": "newuser789", "GameId": "game456"}
        response = increment_counter_safely(
            table_name=table_name,
            key=new_key,
            counter_name=counter_name,
            increment_value=15,
            initial_value=100,
        )
        print(
            f"Counter safely incremented. New value: {response.get('Attributes',
{}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error safely incrementing counter: {e}")

    print("\nExample 4: Conditional counter increment")
    try:
        # Fix for mypy: Handle the case where response might be None
        result = atomic_conditional_increment(
            table_name=table_name,
            key=key,
            counter_name="Achievements",
            condition_attribute="Level",
            condition_value=5,
            increment_value=1,
        )

        if result is not None:
            print(
                f"Conditional increment succeeded. New value:
{result.get('Attributes', {}).get('Achievements')}"
            )
        else:
            print("Conditional increment failed because condition was not met.")
        if response:
            print(
```

```
                f"Conditional increment succeeded. New value:
{response.get('Attributes', {}).get('Achievements')}"
            )
        else:
            print("Conditional increment failed because condition was not met.")
    except Exception as e:
        print(f"Error with conditional increment: {e}")

    print("\nComparison of ADD vs SET for counter operations:")
    print("1. ADD is specifically designed for atomic numeric increments and set
operations")
    print("2. SET with an expression can be used for more complex calculations")
    print("3. Both operations are atomic, preventing race conditions")
    print("4. ADD is more concise for simple increments")
    print("5. SET with if_not_exists() is recommended when the attribute might not
exist")
    print("6. For counters, ADD is generally preferred for clarity and simplicity")
```

- Para obter detalhes da API, consulte a UpdateItem Referência da API AWS SDK for Python (Boto3).

Usar operações condicionais

O exemplo de código a seguir mostra como usar operações condicionais no DynamoDB.

- Implemente gravações condicionais para evitar a substituição de dados.
- Use expressões condicionais para impor regras de negócios.
- Lide com falhas de verificação condicional com tranquilidade.

SDK para Python (Boto3)

Demonstre operações condicionais usando AWS SDK para Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, Optional, Tuple, Union
```

```
def conditional_update(
    table_name: str,
    key: Dict[str, Any],
    condition_attribute: str,
    condition_value: Any,
    update_attribute: str,
    update_value: Any,
) -> Tuple[bool, Optional[Dict[str, Any]]]:
    """
    Update an item only if a condition is met.

    This function demonstrates how to perform a conditional update operation
    and determine if the condition was met.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.

    Returns:
        Tuple[bool, Optional[Dict[str, Any]]]: A tuple containing:
            - A boolean indicating if the update succeeded
            - The response from DynamoDB if successful, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        # Perform the conditional update
        response = table.update_item(
            Key=key,
            UpdateExpression="SET #update_attr = :update_val",
            ConditionExpression="#cond_attr = :cond_val",
            ExpressionAttributeNames={
                "#update_attr": update_attribute,
                "#cond_attr": condition_attribute,
            },
            ExpressionAttributeValues={":update_val": update_value, ":cond_val":
 condition_value},
            ReturnValues="UPDATED_NEW",
```

```
        )
        # Update succeeded, condition was met
        return True, response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return False, None
        else:
            # Other error occurred
            raise


def conditional_delete(
    table_name: str, key: Dict[str, Any], condition_attribute: str, condition_value:
 Any
) -> bool:
    """
    Delete an item only if a condition is met.

    This function demonstrates how to perform a conditional delete operation
    and determine if the condition was met.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to delete.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.

    Returns:
        bool: True if the delete succeeded (condition was met), False otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        # Perform the conditional delete
        table.delete_item(
            Key=key,
            ConditionExpression="#attr = :val",
            ExpressionAttributeNames={"#attr": condition_attribute},
            ExpressionAttributeValues={":val": condition_value},
        )
        # Delete succeeded, condition was met
```

```
        return True
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return False
        else:
            # Other error occurred
            raise


def optimistic_locking_update(
    table_name: str,
    key: Dict[str, Any],
    version_attribute: str,
    update_attribute: str,
    update_value: Any,
) -> Tuple[bool, Optional[Dict[str, Any]]]:
    """
    Update an item using optimistic locking with a version attribute.

    This function demonstrates how to implement optimistic locking using
    a version attribute that is incremented with each update.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        version_attribute (str): The name of the version attribute.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.

    Returns:
        Tuple[bool, Optional[Dict[str, Any]]]: A tuple containing:
            - A boolean indicating if the update succeeded
            - The response from DynamoDB if successful, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # First, get the current version
    try:
        response = table.get_item(
            Key=key,
            ProjectionExpression=f"#{version_attribute}",
```

```
            ExpressionAttributeNames={f"#{version_attribute}": version_attribute},
        )

        item = response.get("Item", {})
        current_version = item.get(version_attribute, 0)

        # Now, try to update with a condition on the version
        try:
            update_response = table.update_item(
                Key=key,
                UpdateExpression=f"SET #{update_attribute} = :update_val,
 #{version_attribute} = :new_version",
                ConditionExpression=f"#{version_attribute} = :current_version",
                ExpressionAttributeNames={
                    f"#{update_attribute}": update_attribute,
                    f"#{version_attribute}": version_attribute,
                },
                ExpressionAttributeValues={
                    ":update_val": update_value,
                    ":current_version": current_version,
                    ":new_version": current_version + 1,
                },
                ReturnValues="UPDATED_NEW",
            )
            # Update succeeded
            return True, update_response
        except ClientError as e:
            if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
                # Version has changed, optimistic locking failed
                return False, None
            else:
                # Other error occurred
                raise
    except ClientError:
        # Error getting the item
        raise


def conditional_check_and_update(
    table_name: str,
    key: Dict[str, Any],
    check_attribute: str,
    check_value: Any,
    update_attribute: str,
```

```
    update_value: Any,
    create_if_not_exists: bool = False,
) -> Union[Dict[str, Any], None]:
    """
    Check if an attribute has a specific value and update another attribute if it
 does.

    This function demonstrates a more complex conditional update that can also
    create the item if it doesn't exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        check_attribute (str): The attribute to check in the condition.
        check_value (Any): The value to compare against.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.
        create_if_not_exists (bool, optional): Whether to create the item if it
 doesn't exist.

    Returns:
        Union[Dict[str, Any], None]: The response from DynamoDB if successful, None
 otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        if create_if_not_exists:
            # Use attribute_not_exists to create the item if it doesn't exist
            condition_expression = "attribute_not_exists(#pk) OR #check_attr
 = :check_val"
            update_expression = "SET #update_attr = :update_val, #check_attr =
 if_not_exists(#check_attr, :check_val)"

            # Get the partition key name from the key dictionary
            pk_name = next(iter(key))

            expression_attribute_names = {
                "#pk": pk_name,
                "#check_attr": check_attribute,
                "#update_attr": update_attribute,
            }
```

```
        else:
            # Only update if the check attribute has the expected value
            condition_expression = "#check_attr = :check_val"
            update_expression = "SET #update_attr = :update_val"

            expression_attribute_names = {
                "#check_attr": check_attribute,
                "#update_attr": update_attribute,
            }

        # Perform the conditional update
        response = table.update_item(
            Key=key,
            UpdateExpression=update_expression,
            ConditionExpression=condition_expression,
            ExpressionAttributeNames=expression_attribute_names,
            ExpressionAttributeValues={":check_val": check_value, ":update_val":
 update_value},
            ReturnValues="UPDATED_NEW",
        )
        return response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return None
        else:
            # Other error occurred
            raise
```

Exemplo de uso de operações condicionais com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use the conditional operations functions."""
    # Example parameters
    table_name = "Products"
    key = {"ProductId": "prod123"}

    print("Example 1: Conditional Update")
    try:
        # Update the price only if the current stock is greater than 10
```

```
        success, response = conditional_update(
            table_name=table_name,
            key=key,
            condition_attribute="Stock",
            condition_value=10,
            update_attribute="Price",
            update_value=99.99,
        )

        if success:
            # Fix for mypy: Handle the case where response might be None
            attributes = {} if response is None else response.get("Attributes", {})
            print(f"Update succeeded! New values: {attributes}")
        else:
            print("Update failed because the condition was not met.")
    except Exception as e:
        print(f"Error during conditional update: {e}")

    print("\nExample 2: Conditional Delete")
    try:
        # Delete the product only if it's discontinued
        success = conditional_delete(
            table_name=table_name,
            key=key,
            condition_attribute="Status",
            condition_value="Discontinued",
        )

        if success:
            print("Delete succeeded! The item was deleted.")
        else:
            print("Delete failed because the condition was not met.")
    except Exception as e:
        print(f"Error during conditional delete: {e}")

    print("\nExample 3: Optimistic Locking")
    try:
        # Update with optimistic locking using a version attribute
        success, response = optimistic_locking_update(
            table_name=table_name,
            key=key,
            version_attribute="Version",
            update_attribute="Description",
            update_value="Updated product description",
```

```
        )

        if success:
            # Fix for mypy: Handle the case where response might be None
            attributes = {} if response is None else response.get("Attributes", {})
            print(f"Optimistic locking update succeeded! New values: {attributes}")
        else:
            print("Optimistic locking update failed because the version has
changed.")
    except Exception as e:
        print(f"Error during optimistic locking update: {e}")

    print("\nExample 4: Conditional Check and Update")
    try:
        # Update the featured status if the product is in stock
        response = conditional_check_and_update(
            table_name=table_name,
            key=key,
            check_attribute="InStock",
            check_value=True,
            update_attribute="Featured",
            update_value=True,
            create_if_not_exists=True,
        )

        if response:
            print(
                f"Conditional check and update succeeded! New values:
{response.get('Attributes', {})}"
            )
        else:
            print("Conditional check and update failed because the condition was not
met.")
    except Exception as e:
        print(f"Error during conditional check and update: {e}")

    print("\nUnderstanding Conditional Operations in DynamoDB:")
    print("1. Conditional operations help maintain data integrity")
    print("2. They prevent race conditions in concurrent environments")
    print("3. Failed conditions result in ConditionalCheckFailedException")
    print("4. No DynamoDB capacity is consumed when conditions fail")
    print("5. Optimistic locking is a common pattern using version attributes")
    print("6. Conditions can be combined with logical operators (AND, OR, NOT)")
    print("7. Conditions can use comparison operators (=, <>, <, <=, >, >=)")
```

```
    print(
        "8. attribute_exists() and attribute_not_exists() are useful for checking
 attribute presence"
    )
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DeleteItem
  - PutItem
  - UpdateItem

Usar nomes de atributo de expressão

O exemplo de código a seguir mostra como usar nomes de atributo de expressão no DynamoDB.

- Trabalhe com palavras reservadas nas expressões do DynamoDB.
- Use espaços reservados para nomes de atributo de expressão.
- Manipule caracteres especiais em nomes de atributo.

SDK para Python (Boto3)

Demonstre nomes de atributos de expressão usando AWS SDK para Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, List


def use_reserved_word_attribute(
    table_name: str, key: Dict[str, Any], reserved_word: str, value: Any
) -> Dict[str, Any]:
    """
    Update an attribute whose name is a DynamoDB reserved word.

    This function demonstrates how to use expression attribute names to work with
    attributes that have names that are DynamoDB reserved words.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        reserved_word (str): The reserved word to use as an attribute name.
        value (Any): The value to set for the attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names to handle the reserved word
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #reserved_attr = :value",
        ExpressionAttributeNames={"#reserved_attr": reserved_word},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def use_special_character_attribute(
    table_name: str, key: Dict[str, Any], attribute_with_special_chars: str, value:
 Any
) -> Dict[str, Any]:
    """
    Update an attribute whose name contains special characters.

    This function demonstrates how to use expression attribute names to work with
    attributes that have names containing special characters like spaces, dots, or
 hyphens.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        attribute_with_special_chars (str): The attribute name with special
 characters.
        value (Any): The value to set for the attribute.
```

```
    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names to handle special characters
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #special_attr = :value",
        ExpressionAttributeNames={"#special_attr": attribute_with_special_chars},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def query_with_attribute_names(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    filter_attribute_name: str,
    filter_value: Any,
) -> Dict[str, Any]:
    """
    Query a table using expression attribute names for both key and filter
 attributes.

    This function demonstrates how to use expression attribute names in a query
 operation
    for both the key condition expression and filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute_name (str): The name of the attribute to filter on.
        filter_value (Any): The value to compare against in the filter.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the query results.
```

```
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names for both key condition and filter
    response = table.query(
        KeyConditionExpression="#pk = :pk_val",
        FilterExpression="#filter_attr = :filter_val",
        ExpressionAttributeNames={"#pk": partition_key_name, "#filter_attr":
 filter_attribute_name},
        ExpressionAttributeValues={":pk_val": partition_key_value, ":filter_val":
 filter_value},
    )

    return response


def update_nested_attribute_with_dots(
    table_name: str, key: Dict[str, Any], path_with_dots: str, value: Any
) -> Dict[str, Any]:
    """
    Update a nested attribute using a path with dot notation.

    This function demonstrates how to use expression attribute names to work with
    nested attributes specified using dot notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path_with_dots (str): The path to the nested attribute using dot notation
 (e.g., "a.b.c").
        value (Any): The value to set for the nested attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Split the path into components
    path_parts = path_with_dots.split(".")
```

```
    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        name_placeholder = f"#attr{i}"
        expression_attribute_names[name_placeholder] = part

        if i == 0:
            path_expression = name_placeholder
        else:
            path_expression += f".{name_placeholder}"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def demonstrate_attribute_name_requirements(table_name: str, key: Dict[str, Any]) ->
 Dict[str, Any]:
    """
    Demonstrate the requirements and allowed characters for attribute names.

    This function shows examples of valid and invalid attribute names and how to
    handle them using expression attribute names.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.

    Returns:
```

```
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Examples of attribute names with different characteristics
    examples = {
        "valid_standard": "NormalAttribute",  # Standard attribute name (no
placeholder needed)
        "valid_with_underscore": "Normal_Attribute",  # Underscore is allowed
        "valid_with_number": "Attribute123",  # Numbers are allowed
        "reserved_word": "Timestamp",  # Reserved word (requires placeholder)
        "starts_with_number": "123Attribute",  # Starts with number (valid but may
need placeholder in some contexts)
        "with_space": "Attribute Name",  # Contains space (requires placeholder)
        "with_dot": "Attribute.Name",  # Contains dot (requires placeholder)
        "with_hyphen": "Attribute-Name",  # Contains hyphen (requires placeholder)
        "with_special_chars": "Attribute#$%",  # Contains special characters
(requires placeholder)
    }

    results = {}

    # Try to update each attribute type
    for example_type, attr_name in examples.items():
        try:
            # For attributes that don't need placeholders, try direct reference
            if example_type in ["valid_standard", "valid_with_underscore",
"valid_with_number"]:
                try:
                    # Try without expression attribute names first
                    response = table.update_item(
                        Key=key,
                        UpdateExpression=f"SET {attr_name} = :value",
                        ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                        ReturnValues="UPDATED_NEW",
                    )
                    results[example_type] = {
                        "attribute_name": attr_name,
                        "success": True,
                        "needed_placeholder": False,
                        "response": response,
```

```
                    }
                except ClientError:
                    # If direct reference fails, try with placeholder
                    response = table.update_item(
                        Key=key,
                        UpdateExpression="SET #attr = :value",
                        ExpressionAttributeNames={"#attr": attr_name},
                        ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                        ReturnValues="UPDATED_NEW",
                    )
                    results[example_type] = {
                        "attribute_name": attr_name,
                        "success": True,
                        "needed_placeholder": True,
                        "response": response,
                    }
            else:
                # For attributes that definitely need placeholders
                response = table.update_item(
                    Key=key,
                    UpdateExpression="SET #attr = :value",
                    ExpressionAttributeNames={"#attr": attr_name},
                    ExpressionAttributeValues={":value": f"Value for {attr_name}"},
                    ReturnValues="UPDATED_NEW",
                )
                results[example_type] = {
                    "attribute_name": attr_name,
                    "success": True,
                    "needed_placeholder": True,
                    "response": response,
                }
        except ClientError as e:
            results[example_type] = {"attribute_name": attr_name, "success": False,
"error": str(e)}

    return results
```

Exemplo de uso de nomes de atributos de expressão com AWS SDK para Python (Boto3).

```
def example_usage():
    """Example of how to use expression attribute names in DynamoDB."""
    # Example parameters
    table_name = "Products"
    key = {"ProductId": "prod123"}

    print("Example 1: Using a reserved word as an attribute name")
    try:
        response = use_reserved_word_attribute(
            table_name=table_name, key=key, reserved_word="Timestamp",
 value="2025-05-14T12:00:00Z"
        )
        print(f"Reserved word attribute updated successfully:
 {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating reserved word attribute: {e}")

    print("\nExample 2: Using an attribute name with special characters")
    try:
        response = use_special_character_attribute(
            table_name=table_name,
            key=key,
            attribute_with_special_chars="Product Info",
            value="Special product information",
        )
        print(f"Special character attribute updated successfully:
 {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating special character attribute: {e}")

    print("\nExample 3: Querying with expression attribute names")
    try:
        response = query_with_attribute_names(
            table_name=table_name,
            partition_key_name="Category",
            partition_key_value="Electronics",
            filter_attribute_name="Price",
            filter_value=500,
        )
        print(
            f"Query with expression attribute names returned
 {len(response.get('Items', []))} items"
        )
```

```
    except Exception as e:
        print(f"Error querying with expression attribute names: {e}")

    print("\nExample 4: Updating a nested attribute with dot notation")
    try:
        response = update_nested_attribute_with_dots(
            table_name=table_name,
            key=key,
            path_with_dots="Product.Details.Specifications",
            value={"Weight": "2.5 kg", "Dimensions": "30x20x10 cm"},
        )
        print(f"Nested attribute updated successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error updating nested attribute: {e}")

    print("\nExample 5: Demonstrating attribute name requirements")
    try:
        results = demonstrate_attribute_name_requirements(table_name=table_name,
key=key)

        print("Attribute Name Requirements Results:")
        for example_type, result in results.items():
            if result.get("success", False):
                needed_placeholder = result.get("needed_placeholder", True)
                print(
                    f"  - {example_type}: '{result['attribute_name']}' - {'Requires'
if needed_placeholder else 'Does not require'} placeholder"
                )
            else:
                print(
                    f"  - {example_type}: '{result['attribute_name']}' - Failed:
{result.get('error', 'Unknown error')}"
                )
    except Exception as e:
        print(f"Error demonstrating attribute name requirements: {e}")

    print("\nCommon DynamoDB Reserved Words (sample):")
    reserved_words = get_common_reserved_words()
    print(", ".join(reserved_words[:20]) + "... (and many more)")

    print("\nWhen to Use Expression Attribute Names:")
    print("1. When the attribute name is a DynamoDB reserved word")
```

```
    print("2. When the attribute name contains special characters (spaces, dots,
hyphens)")
    print("3. When the attribute name begins with a number")
    print("4. When working with nested attributes using dot notation")
    print("5. When you need to reference the same attribute multiple times in an
expression")

    print("\nExpression Attribute Name Requirements:")
    print("1. Must begin with a pound sign (#)")
    print("2. After the pound sign, must contain at least one character")
    print("3. Can contain alphanumeric characters and underscore (_)")
    print("4. Are case-sensitive")
    print("5. Must be unique within a single expression")

    print("\nAttribute Name Requirements in DynamoDB:")
    print("1. Can begin with a-z, A-Z, or 0-9")
    print("2. Can contain a-z, A-Z, 0-9, underscore (_), dash (-), and dot (.)")
    print("3. Are case-sensitive")
    print("4. No length restrictions, but practical limits apply")
    print("5. Cannot be a DynamoDB reserved word if used directly in expressions")
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - Consulta
  - UpdateItem

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento agendado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função do Lambda o alvo.

- Concede permissão para permitir que a função Lambda seja EventBridge invocada.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações agendadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos sem servidor

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Python.

```
import json

def lambda_handler(event, context):
    print(json.dumps(event, indent=2))
```

```
    for record in event['Records']:
        log_dynamodb_record(record)

def log_dynamodb_record(record):
    print(record['eventID'])
    print(record['eventName'])
    print(f"DynamoDB Record: {json.dumps(record['dynamodb'])}")
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["dynamodb"]["SequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}
```

```
    return {"batchItemFailures":[]}
```

Exemplos do Amazon EC2 usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon EC2.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def hello_ec2(ec2_client):
    """
    Use the AWS SDK for Python (Boto3) to list the security groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ec2_client: A Boto3 EC2 client. This client provides low-level
                       access to AWS EC2 services.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    try:
        paginator = ec2_client.get_paginator("describe_security_groups")
        response_iterator = paginator.paginate(PaginationConfig={'MaxItems': 10}) #
 List only 10 security groups.
        logging.basicConfig(level=logging.INFO) # Enable logging.
        for page in response_iterator:
            for sg in page["SecurityGroups"]:
                logger.info(f"\t{sg['GroupId']}: {sg['GroupName']}")
    except ClientError as err:
        logger.error("Failed to list security groups.")
        if err.response["Error"]["Code"] == "AccessDeniedException":
            logger.error("You do not have permission to list security groups.")
        raise


if __name__ == "__main__":
    hello_ec2(boto3.client("ec2"))
```

- Para obter detalhes da API, consulte a DescribeSecurityGroups Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class EC2InstanceScenario:
    """
    A scenario that demonstrates how to use Boto3 to manage Amazon EC2 resources.
    Covers creating a key pair, security group, launching an instance, associating
    an Elastic IP, and cleaning up resources.
    """

    def __init__(
        self,
        inst_wrapper: EC2InstanceWrapper,
        key_wrapper: KeyPairWrapper,
        sg_wrapper: SecurityGroupWrapper,
        eip_wrapper: ElasticIpWrapper,
        ssm_client: boto3.client,
        remote_exec: bool = False,
    ):
        """
```

```
        Initializes the EC2InstanceScenario with the necessary AWS service wrappers.

        :param inst_wrapper: Wrapper for EC2 instance operations.
        :param key_wrapper: Wrapper for key pair operations.
        :param sg_wrapper: Wrapper for security group operations.
        :param eip_wrapper: Wrapper for Elastic IP operations.
        :param ssm_client: Boto3 client for accessing SSM to retrieve AMIs.
        :param remote_exec: Flag to indicate if the scenario is running in a remote
execution
                            environment. Defaults to False. If True, the script
won't prompt
                            for user interaction.
        """
        self.inst_wrapper = inst_wrapper
        self.key_wrapper = key_wrapper
        self.sg_wrapper = sg_wrapper
        self.eip_wrapper = eip_wrapper
        self.ssm_client = ssm_client
        self.remote_exec = remote_exec

    def create_and_list_key_pairs(self) -> None:
        """
        Creates an RSA key pair for SSH access to the EC2 instance and lists
available key pairs.
        """
        console.print("**Step 1: Create a Secure Key Pair**", style="bold cyan")
        console.print(
            "Let's create a secure RSA key pair for connecting to your EC2
instance."
        )
        key_name = f"MyUniqueKeyPair-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Key Pair Name**: {key_name}")

        # Create the key pair and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Key Pair") as bar:
            self.key_wrapper.create(key_name)
            time.sleep(0.4)  # Simulate the delay in key creation
            bar()

        console.print(f"- **Private Key Saved to**:
{self.key_wrapper.key_file_path}\n")

        # List key pairs (simulated) and show a progress bar.
        list_keys = True
```

```
        if list_keys:
            console.print("- Listing your key pairs...")
            start_time = time.time()
            with alive_bar(100, title="Listing Key Pairs") as bar:
                while time.time() - start_time < 2:
                    time.sleep(0.2)
                    bar(10)
                self.key_wrapper.list(5)
                if time.time() - start_time > 2:
                    console.print(
                        "Taking longer than expected! Please wait...",
                        style="bold yellow",
                    )

    def create_security_group(self) -> None:
        """
        Creates a security group that controls access to the EC2 instance and adds a
rule
        to allow SSH access from the user's current public IP address.
        """
        console.print("**Step 2: Create a Security Group**", style="bold cyan")
        console.print(
            "Security groups manage access to your instance. Let's create one."
        )
        sg_name = f"MySecurityGroup-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Security Group Name**: {sg_name}")

        # Create the security group and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Security Group") as bar:
            self.sg_wrapper.create(
                sg_name, "Security group for example: get started with instances."
            )
            time.sleep(0.5)
            bar()

        console.print(f"- **Security Group ID**:
{self.sg_wrapper.security_group}\n")

        # Get the current public IP to set up SSH access.
        ip_response = urllib.request.urlopen("http://checkip.amazonaws.com")
        current_ip_address = ip_response.read().decode("utf-8").strip()
        console.print(
            "Let's add a rule to allow SSH only from your current IP address."
        )
```

```
        console.print(f"- **Your Public IP Address**: {current_ip_address}")
        console.print("- Automatically adding SSH rule...")

        # Update security group rules to allow SSH and simulate with a progress bar.
        with alive_bar(1, title="Updating Security Group Rules") as bar:
            response = self.sg_wrapper.authorize_ingress(current_ip_address)
            time.sleep(0.4)
            if response and response.get("Return"):
                console.print("- **Security Group Rules Updated**.")
            else:
                console.print(
                    "- **Error**: Couldn't update security group rules.",
                    style="bold red",
                )
            bar()

        self.sg_wrapper.describe(self.sg_wrapper.security_group)

    def create_instance(self) -> None:
        """
        Launches an EC2 instance using an Amazon Linux 2 AMI and the created key
 pair
        and security group. Displays instance details and SSH connection
 information.
        """
        # Retrieve Amazon Linux 2 AMIs from SSM.
        ami_paginator = self.ssm_client.get_paginator("get_parameters_by_path")
        ami_options = []
        for page in ami_paginator.paginate(Path="/aws/service/ami-amazon-linux-
latest"):
            ami_options += page["Parameters"]
        amzn2_images = self.inst_wrapper.get_images(
            [opt["Value"] for opt in ami_options if "amzn2" in opt["Name"]]
        )
        console.print("\n**Step 3: Launch Your Instance**", style="bold cyan")
        console.print(
            "Let's create an instance from an Amazon Linux 2 AMI. Here are some
 options:"
        )
        image_choice = 0
        console.print(f"- Selected AMI: {amzn2_images[image_choice]['ImageId']}\n")

        # Display instance types compatible with the selected AMI
        inst_types = self.inst_wrapper.get_instance_types(
```

```
            amzn2_images[image_choice]["Architecture"]
        )
        inst_type_choice = 0
        console.print(
            f"- Selected instance type: {inst_types[inst_type_choice]
['InstanceType']}\n"
        )

        console.print("Creating your instance and waiting for it to start...")
        with alive_bar(1, title="Creating Instance") as bar:
            self.inst_wrapper.create(
                amzn2_images[image_choice]["ImageId"],
                inst_types[inst_type_choice]["InstanceType"],
                self.key_wrapper.key_pair["KeyName"],
                [self.sg_wrapper.security_group],
            )
            time.sleep(21)
            bar()

        console.print(f"**Success! Your instance is ready:**\n", style="bold green")
        self.inst_wrapper.display()

        console.print(
            "You can use SSH to connect to your instance. "
            "If the connection attempt times out, you might have to manually update
 "
            "the SSH ingress rule for your IP address in the AWS Management
 Console."
        )
        self._display_ssh_info()

    def _display_ssh_info(self) -> None:
        """
        Displays SSH connection information for the user to connect to the EC2
 instance.
        Handles the case where the instance does or does not have an associated
 public IP address.
        """
        if (
            not self.eip_wrapper.elastic_ips
            or not self.eip_wrapper.elastic_ips[0].allocation_id
        ):
            if self.inst_wrapper.instances:
                instance = self.inst_wrapper.instances[0]
```

```
                instance_id = instance["InstanceId"]

                waiter = self.inst_wrapper.ec2_client.get_waiter("instance_running")
                console.print(
                    "Waiting for the instance to be in a running state with a public
 IP...",
                    style="bold cyan",
                )

                with alive_bar(1, title="Waiting for Instance to Start") as bar:
                    waiter.wait(InstanceIds=[instance_id])
                    time.sleep(20)
                    bar()

                instance = self.inst_wrapper.ec2_client.describe_instances(
                    InstanceIds=[instance_id]
                )["Reservations"][0]["Instances"][0]

                public_ip = instance.get("PublicIpAddress")
                if public_ip:
                    console.print(
                        "\nTo connect via SSH, open another command prompt and run
 the following command:",
                        style="bold cyan",
                    )
                    console.print(
                        f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{public_ip}"
                    )
                else:
                    console.print(
                        "Instance does not have a public IP address assigned.",
                        style="bold red",
                    )
            else:
                console.print(
                    "No instance available to retrieve public IP address.",
                    style="bold red",
                )
        else:
            elastic_ip = self.eip_wrapper.elastic_ips[0]
            elastic_ip_address = elastic_ip.public_ip
            console.print(
```

```
                f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{elastic_ip_address}"
            )

        if not self.remote_exec:
            console.print("\nOpen a new terminal tab to try the above SSH command.")
            input("Press Enter to continue...")

    def associate_elastic_ip(self) -> None:
        """
        Allocates an Elastic IP address and associates it with the EC2 instance.
        Displays the Elastic IP address and SSH connection information.
        """
        console.print("\n**Step 4: Allocate an Elastic IP Address**", style="bold
 cyan")
        console.print(
            "You can allocate an Elastic IP address and associate it with your
 instance\n"
            "to keep a consistent IP address even when your instance restarts."
        )

        with alive_bar(1, title="Allocating Elastic IP") as bar:
            elastic_ip = self.eip_wrapper.allocate()
            time.sleep(0.5)
            bar()

        console.print(
            f"- **Allocated Static Elastic IP Address**: {elastic_ip.public_ip}."
        )

        with alive_bar(1, title="Associating Elastic IP") as bar:
            self.eip_wrapper.associate(
                elastic_ip.allocation_id, self.inst_wrapper.instances[0]
["InstanceId"]
            )
            time.sleep(2)
            bar()

        console.print(f"- **Associated Elastic IP with Your Instance**.")
        console.print(
            "You can now use SSH to connect to your instance by using the Elastic
 IP."
        )
        self._display_ssh_info()
```

```
    def stop_and_start_instance(self) -> None:
        """
        Stops and restarts the EC2 instance. Displays instance state and explains
        changes that occur when the instance is restarted, such as the potential
change
        in the public IP address unless an Elastic IP is associated.
        """
        console.print("\n**Step 5: Stop and Start Your Instance**", style="bold
cyan")
        console.print("Let's stop and start your instance to see what changes.")
        console.print("- **Stopping your instance and waiting until it's
stopped...**")

        with alive_bar(1, title="Stopping Instance") as bar:
            self.inst_wrapper.stop()
            time.sleep(360)
            bar()

        console.print("- **Your instance is stopped. Restarting...**")

        with alive_bar(1, title="Starting Instance") as bar:
            self.inst_wrapper.start()
            time.sleep(20)
            bar()

        console.print("**Your instance is running.**", style="bold green")
        self.inst_wrapper.display()

        elastic_ip = (
            self.eip_wrapper.elastic_ips[0] if self.eip_wrapper.elastic_ips else
None
        )

        if elastic_ip is None or elastic_ip.allocation_id is None:
            console.print(
                "- **Note**: Every time your instance is restarted, its public IP
address changes."
            )
        else:
            console.print(
                f"Because you have associated an Elastic IP with your instance, you
can \n"
```

```
                f"connect by using a consistent IP address after the instance
 restarts: {elastic_ip.public_ip}"
            )

        self._display_ssh_info()

    def cleanup(self) -> None:
        """
        Cleans up all the resources created during the scenario, including
 disassociating
        and releasing the Elastic IP, terminating the instance, deleting the
 security
        group, and deleting the key pair.
        """
        console.print("\n**Step 6: Clean Up Resources**", style="bold cyan")
        console.print("Cleaning up resources:")

        for elastic_ip in self.eip_wrapper.elastic_ips:
            console.print(f"- **Elastic IP**: {elastic_ip.public_ip}")

            with alive_bar(1, title="Disassociating Elastic IP") as bar:
                self.eip_wrapper.disassociate(elastic_ip.allocation_id)
                time.sleep(2)
                bar()

            console.print("\t- **Disassociated Elastic IP from the Instance**")

            with alive_bar(1, title="Releasing Elastic IP") as bar:
                self.eip_wrapper.release(elastic_ip.allocation_id)
                time.sleep(1)
                bar()

            console.print("\t- **Released Elastic IP**")

        console.print(f"- **Instance**: {self.inst_wrapper.instances[0]
['InstanceId']}")

        with alive_bar(1, title="Terminating Instance") as bar:
            self.inst_wrapper.terminate()
            time.sleep(380)
            bar()

        console.print("\t- **Terminated Instance**")
```

```
        console.print(f"- **Security Group**: {self.sg_wrapper.security_group}")

        with alive_bar(1, title="Deleting Security Group") as bar:
            self.sg_wrapper.delete(self.sg_wrapper.security_group)
            time.sleep(1)
            bar()

        console.print("\t- **Deleted Security Group**")

        console.print(f"- **Key Pair**: {self.key_wrapper.key_pair['KeyName']}")

        with alive_bar(1, title="Deleting Key Pair") as bar:
            self.key_wrapper.delete(self.key_wrapper.key_pair["KeyName"])
            time.sleep(0.4)
            bar()

        console.print("\t- **Deleted Key Pair**")

    def run_scenario(self) -> None:
        """
        Executes the entire EC2 instance scenario: creates key pairs, security
groups,
        launches an instance, associates an Elastic IP, and cleans up all resources.
        """
        logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

        console.print("-" * 88)
        console.print(
            "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
with instances demo.",
            style="bold magenta",
        )
        console.print("-" * 88)

        self.create_and_list_key_pairs()
        self.create_security_group()
        self.create_instance()
        self.stop_and_start_instance()
        self.associate_elastic_ip()
        self.stop_and_start_instance()
        self.cleanup()

        console.print("\nThanks for watching!", style="bold green")
        console.print("-" * 88)
```

```
if __name__ == "__main__":
    try:
        scenario = EC2InstanceScenario(
            EC2InstanceWrapper.from_client(),
            KeyPairWrapper.from_client(),
            SecurityGroupWrapper.from_client(),
            ElasticIpWrapper.from_client(),
            boto3.client("ssm"),
        )
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que envolva as ações de pares de chaves.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
```

```
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2 instance.
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that represents
the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
```

```
                )
            raise
        else:
            return self.key_pair


    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.

        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
                      all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}' with
fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise


    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                 if the key pair does not exist.
        """
```

```
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
                    f"The key pair '{key_name}' does not exist and cannot be
 deleted. "
                    "Please verify the key pair name and try again."
                )
            raise
```

Defina uma classe que envolva as ações de grupos de segurança.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
```

```
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of the
current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling for
ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose a
different name."
                )
            raise
        else:
            return self.security_group


    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                               to port 22 over TCP, used for SSH.
```

```
        :return: The response to the authorization request. The 'Return' field of
the
                 response indicates whether the request succeeded or failed, or None
if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling for
ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already exists"
                    f"in security group '{self.security_group}'."
                )
            raise
        else:
            return response


    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                  If None, an open search is performed to describe
all security groups.
        :returns: True if the description is successful.
```

```
        :raises ClientError: If there is an error describing the security group(s),
such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

            for page in page_iterator:
                for security_group in page["SecurityGroups"]:
                    print(f"Security group: {security_group['GroupName']}")
                    print(f"\tID: {security_group['GroupId']}")
                    print(f"\tVPC: {security_group['VpcId']}")
                    if security_group["IpPermissions"]:
                        print("Inbound permissions:")
                        pp(security_group["IpPermissions"])

            return True
        except ClientError as err:
            logger.error("Failed to describe security group(s).")
            if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group {security_group_id} does not exist "
                    f"because the specified security group ID was not found."
                )
            raise


    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.

        :param security_group_id: The ID of the security group to delete. Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an AWS
service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
```

```
            logger.info(f"Successfully deleted security group
 '{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
 '{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted because
 it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted because
 it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise
```

Defina uma classe que envolva as ações de instâncias.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
```

```
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
the instance.
                                   If not specified, the default security group of
the VPC is used.
        :return: A list of dictionaries representing Boto3 Instance objects
representing the newly created instances.
        """
        try:
            instance_params = {
```

```
                "ImageId": image_id,
                "InstanceType": instance_type,
                "KeyName": key_pair_name,
            }
            if security_group_ids is not None:
                instance_params["SecurityGroupIds"] = security_group_ids

            response = self.ec2_client.run_instances(
                **instance_params, MinCount=1, MaxCount=1
            )
            instance = response["Instances"][0]
            self.instances.append(instance)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=[instance["InstanceId"]])
        except ClientError as err:
            params_str = "\n\t".join(
                f"{key}: {value}" for key, value in instance_params.items()
            )
            logger.error(
                f"Failed to complete instance creation request.\nRequest details:
{params_str}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InstanceLimitExceeded":
                logger.error(
                    (
                        f"Insufficient capacity for instance type '{instance_type}'.
 "
                        "Terminate unused instances or contact AWS Support for a
 limit increase."
                    )
                )
            if error_code == "InsufficientInstanceCapacity":
                logger.error(
                    (
                        f"Insufficient capacity for instance type '{instance_type}'.
 "
                        "Select a different instance type or launch in a different
 availability zone."
                    )
                )
            raise
        return self.instances
```

```
    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
 instances in this state
                             will be displayed. Default is 'running'. Example
 states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:
                            continue  # Skip this instance if it doesn't match the
 filter

                        # Create a formatted string with instance details
                        instance_info = (
                            f"• ID: {instance['InstanceId']}\n"
                            f"• Image ID: {instance['ImageId']}\n"
                            f"• Instance type: {instance['InstanceType']}\n"
                            f"• Key name: {instance['KeyName']}\n"
                            f"• VPC ID: {instance['VpcId']}\n"
                            f"• Public IP: {instance.get('PublicIpAddress', 'N/
A')}\n"
                            f"• State: {instance_state}"
                        )
                        print(instance_info)

        except ClientError as err:
            logger.error(
```

```
                f"Failed to display instance(s). : {' '.join(map(str,
instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
                raise


    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
            raise


    def start(self) -> Optional[Dict[str, Any]]:
        """
```

```
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=instance_ids)
            return start_response
        except ClientError as err:
            logger.error(
                f"Failed to start instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't start instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a stopped state before starting
them."
                )
            raise


    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no instances
to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
```

```
            stop_response = self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response


    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str, image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images


    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
"*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
```

```
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
API_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types. Default:
 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support the
 specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture}, {','.join(map(str,
 sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "
                    "Ensure architecture and size strings conform to
 DescribeInstanceTypes API reference."
                )
            raise
        else:
            return inst_types
```

Defina uma classe que envolva as ações de IP elástico.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
 None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the maximum
limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
Support for an increase."
                )
            raise err
        return elastic_ip


    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
        :return: A response that contains the ID of the association, or None if no
Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance ID
is not found.
```

```
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID {allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id  # Track the instance associated with this Elastic IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
                )
            raise
        return response


    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance. When
the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to disassociate.
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(
```

```
                f"No association found for Elastic IP with allocation ID
{allocation_id}."
            )
            return

        try:
            # Retrieve the association ID before disassociating
            response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
            association_id = response["Addresses"][0].get("AssociationId")

            if association_id:
                self.ec2_client.disassociate_address(AssociationId=association_id)
                elastic_ip.instance_id = None  # Remove the instance association
            else:
                logger.info(
                    f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
                )

        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
                logger.error(
                    f"Failed to disassociate Elastic IP {allocation_id} "
                    "because the specified association ID for the Elastic IP address
was not found. "
                    "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                    "resource before attempting to disassociate it."
                )
            raise


    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
```

```
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID {allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip)  # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address "
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise


    @staticmethod
    def get_elastic_ip_by_allocation(
        elastic_ips: List["ElasticIpWrapper.ElasticIp"], allocation_id: str
    ) -> Optional["ElasticIpWrapper.ElasticIp"]:
        """
        Retrieves an Elastic IP object by its allocation ID from a given list of
Elastic IPs.

        :param elastic_ips: A list of ElasticIp objects.
        :param allocation_id: The allocation ID of the Elastic IP to retrieve.
        :return: The ElasticIp object associated with the allocation ID, or None if
not found.
        """
        return next(
            (ip for ip in elastic_ips if ip.allocation_id == allocation_id), None
        )
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AllocateAddress

- AssociateAddress
- AuthorizeSecurityGroupIngress
- CreateKeyPair
- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticIpWrapper:
```

```
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
```

```
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the maximum
limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
Support for an increase."
                )
            raise err
        return elastic_ip
```

- Para obter detalhes da API, consulte a AllocateAddress Referência da API AWS SDK for Python (Boto3).

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
 None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
        :return: A response that contains the ID of the association, or None if no
Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance ID
is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID {allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id  # Track the instance associated with this Elastic IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
                )
            raise
        return response
```

- Para obter detalhes da API, consulte a AssociateAddress Referência da API AWS SDK for Python (Boto3).

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.
```

```
        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                               to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                 response indicates whether the request succeeded or failed, or None
if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling for
ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already exists"
                    f"in security group '{self.security_group}'."
                )
```

```
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a AuthorizeSecurityGroupIngress Referência da API AWS SDK for Python (Boto3).

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is stored.
```

```
                                This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2 instance.
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that represents
the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
                )
            raise
        else:
            return self.key_pair
```

- Para obter detalhes da API, consulte a CreateKeyPair Referência da API AWS SDK for Python (Boto3).

`CreateLaunchTemplate`

O código de exemplo a seguir mostra como usar `CreateLaunchTemplate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo cria um modelo de execução que inclui um perfil de instância que concede permissões específicas à instância e um script Bash de dados do usuário que é executado na instância após sua inicialização.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
```

```
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
```

```
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
```

```
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template
```

- Para obter detalhes da API, consulte a CreateLaunchTemplate Referência da API AWS SDK for Python (Boto3).

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SecurityGroupWrapper:
```

```
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of the
 current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling for
 ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
```

```
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose a
 different name."
                )
            raise
        else:
            return self.security_group
```

- Para obter detalhes da API, consulte a CreateSecurityGroup Referência da API AWS SDK for Python (Boto3).

`CreateVpc`

O código de exemplo a seguir mostra como usar `CreateVpc`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
```

```
    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(self, cidr_block: str) -> str:
        """
        Creates a new Amazon VPC with the specified CIDR block.

        :param cidr_block: The CIDR block for the new VPC, such as '10.0.0.0/16'.
        :return: The ID of the new VPC.
        """
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr_block)
            vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[vpc_id])
            return vpc_id
        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateVpc Referência da API AWS SDK for Python (Boto3).

`CreateVpcEndpoint`

O código de exemplo a seguir mostra como usar `CreateVpcEndpoint`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create_vpc_endpoint(
        self, vpc_id: str, service_name: str, route_table_ids: list[str]
    ) -> Dict[str, Any]:
        """
        Creates a new VPC endpoint for the specified service and associates it with
 the specified route tables.

        :param vpc_id: The ID of the VPC to create the endpoint in.
        :param service_name: The name of the service to create the endpoint for.
```

```
        :param route_table_ids: A list of IDs of the route tables to associate with
 the endpoint.
        :return: A dictionary representing the newly created VPC endpoint.
        """
        try:
            response = self.ec2_client.create_vpc_endpoint(
                VpcId=vpc_id,
                ServiceName=service_name,
                RouteTableIds=route_table_ids,
            )
            return response["VpcEndpoint"]
        except ClientError as err:
            logger.error(
                "Couldn't create VPC endpoint for service %s. Here's why: %s: %s",
                service_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateVpcEndpoint Referência da API AWS SDK for Python (Boto3).

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
```

```
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
```

```
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
                    f"The key pair '{key_name}' does not exist and cannot be
deleted. "
                    "Please verify the key pair name and try again."
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteKeyPair Referência da API AWS SDK for Python (Boto3).

`DeleteLaunchTemplate`

O código de exemplo a seguir mostra como usar `DeleteLaunchTemplate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """
```

```
    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
```

```
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DeleteLaunchTemplate Referência da API AWS SDK for Python (Boto3).

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.
```

```
        :param security_group_id: The ID of the security group to delete. Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an AWS
service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
'{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
'{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted because
it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted because
it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteSecurityGroup Referência da API AWS SDK for Python (Boto3).

`DeleteVpc`

O código de exemplo a seguir mostra como usar `DeleteVpc`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete(self, vpc_id: str) -> None:
        """
        Deletes the specified VPC.

        :param vpc_id: The ID of the VPC to delete.
        """
        try:
            self.ec2_client.delete_vpc(VpcId=vpc_id)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't delete VPC %s. Here's why: %s: %s",
                vpc_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteVpc Referência da API AWS SDK for Python (Boto3).

`DeleteVpcEndpoints`

O código de exemplo a seguir mostra como usar `DeleteVpcEndpoints`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.
```

```
        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete_vpc_endpoints(self, vpc_endpoint_ids: list[str]) -> None:
        """
        Deletes the specified VPC endpoints.

        :param vpc_endpoint_ids: A list of IDs of the VPC endpoints to delete.
        """
        try:
            self.ec2_client.delete_vpc_endpoints(VpcEndpointIds=vpc_endpoint_ids)
        except ClientError as err:
            logger.error(
                "Couldn't delete VPC endpoints %s. Here's why: %s: %s",
                vpc_endpoint_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteVpcEndpoints Referência da API AWS SDK for Python (Boto3).

`DescribeAvailabilityZones`

O código de exemplo a seguir mostra como usar `DescribeAvailabilityZones`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
```

```
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones
```

- Para obter detalhes da API, consulte a DescribeAvailabilityZones Referência da API AWS SDK for Python (Boto3).

`DescribeIamInstanceProfileAssociations`

O código de exemplo a seguir mostra como usar `DescribeIamInstanceProfileAssociations`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
```

```
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DescribeIamInstanceProfileAssociations Referência da API AWS SDK for Python (Boto3).

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
```

```
        Gets information about Amazon Machine Images (AMIs) from a list of AMI IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str, image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images
```

- Para obter detalhes da API, consulte a DescribeImages Referência da API AWS SDK for Python (Boto3).

`DescribeInstanceTypes`

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
```

```
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
 "*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
API_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types. Default:
 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support the
 specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
```

```
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture}, {','.join(map(str,
sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "
                    "Ensure architecture and size strings conform to
DescribeInstanceTypes API reference."
                )
            raise
        else:
            return inst_types
```

- Para obter detalhes da API, consulte a DescribeInstanceTypes Referência da API AWS SDK for Python (Boto3).

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
 instances in this state
                              will be displayed. Default is 'running'. Example
 states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
```

```
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:
                            continue  # Skip this instance if it doesn't match the
 filter

                        # Create a formatted string with instance details
                        instance_info = (
                            f"• ID: {instance['InstanceId']}\n"
                            f"• Image ID: {instance['ImageId']}\n"
                            f"• Instance type: {instance['InstanceType']}\n"
                            f"• Key name: {instance['KeyName']}\n"
                            f"• VPC ID: {instance['VpcId']}\n"
                            f"• Public IP: {instance.get('PublicIpAddress', 'N/
A')}\n"
                            f"• State: {instance_state}"
                        )
                        print(instance_info)

        except ClientError as err:
            logger.error(
                f"Failed to display instance(s). : {' '.join(map(str,
 instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
                raise
```

- Para obter detalhes da API, consulte a DescribeInstances Referência da API AWS SDK for Python (Boto3).

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
```

```
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.

        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
                      all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}' with
fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise
```

- Para obter detalhes da API, consulte a DescribeKeyPairs Referência da API AWS SDK for Python (Boto3).

`DescribeRouteTables`

O código de exemplo a seguir mostra como usar `DescribeRouteTables`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def describe_route_tables(self, vpc_ids: list[str]) -> None:
        """
        Displays information about the route tables in the specified VPC.

        :param vpc_ids: A list of VPC IDs.
        """
        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": vpc_ids}]
            )
            pp(response["RouteTables"])
        except ClientError as err:
            logger.error(
                "Couldn't describe route tables for VPCs %s. Here's why: %s: %s",
                vpc_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeRouteTables Referência da API AWS SDK for Python (Boto3).

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""
```

```
    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
 groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                  If None, an open search is performed to describe
 all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security group(s),
 such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
```

```
            # If no ID is provided, return all security groups.
            page_iterator = paginator.paginate()
        else:
            page_iterator = paginator.paginate(GroupIds=[security_group_id])

        for page in page_iterator:
            for security_group in page["SecurityGroups"]:
                print(f"Security group: {security_group['GroupName']}")
                print(f"\tID: {security_group['GroupId']}")
                print(f"\tVPC: {security_group['VpcId']}")
                if security_group["IpPermissions"]:
                    print("Inbound permissions:")
                    pp(security_group["IpPermissions"])

        return True
    except ClientError as err:
        logger.error("Failed to describe security group(s).")
        if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
            logger.error(
                f"Security group {security_group_id} does not exist "
                f"because the specified security group ID was not found."
            )
        raise
```

- Para obter detalhes da API, consulte a DescribeSecurityGroups Referência da API AWS SDK for Python (Boto3).

`DescribeSubnets`

O código de exemplo a seguir mostra como usar `DescribeSubnets`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
```

```
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
```

```
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DescribeSubnets Referência da API AWS SDK for Python (Boto3).

`DescribeVpcs`

O código de exemplo a seguir mostra como usar `DescribeVpcs`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.
```

```
        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
```

```
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

- Para obter detalhes da API, consulte a DescribeVpcs Referência da API AWS SDK for Python (Boto3).

`DisassociateAddress`

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
 None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance. When
the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to disassociate.
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(
                f"No association found for Elastic IP with allocation ID
{allocation_id}."
            )
            return

        try:
            # Retrieve the association ID before disassociating
            response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
            association_id = response["Addresses"][0].get("AssociationId")

            if association_id:
                self.ec2_client.disassociate_address(AssociationId=association_id)
                elastic_ip.instance_id = None  # Remove the instance association
            else:
                logger.info(
                    f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
                )

        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
                logger.error(
                    f"Failed to disassociate Elastic IP {allocation_id} "
                    "because the specified association ID for the Elastic IP address
was not found. "
                    "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                    "resource before attempting to disassociate it."
```

```
                )
            raise
```

- Para obter detalhes da API, consulte a DisassociateAddress Referência da API AWS SDK for Python (Boto3).

`RebootInstances`

O código de exemplo a seguir mostra como usar `RebootInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
```

```
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.
```

```
        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
```

```
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
 not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a RebootInstances Referência da API AWS SDK for Python (Boto3).

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
 None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
```

```
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID {allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
```

```
            self.elastic_ips.remove(elastic_ip)  # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address "
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise
```

- Para obter detalhes da API, consulte a ReleaseAddress Referência da API AWS SDK for Python (Boto3).

`ReplaceIamInstanceProfileAssociation`

O código de exemplo a seguir mostra como usar `ReplaceIamInstanceProfileAssociation`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo substitui o perfil de instância de uma instância em execução, reinicia a instância e envia um comando para a instância após ela iniciar.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
```

```
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def replace_instance_profile(
```

```
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                            instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
```

```
            )
            log.info(f"Restarted the Python web server on instance
 '{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
 not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a ReplaceIamInstanceProfileAssociation Referência da API AWS SDK for Python (Boto3).

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
```

```
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
```

```
        :param security_group_ids: A list of security group IDs to associate with
 the instance.
                                   If not specified, the default security group of
 the VPC is used.
        :return: A list of dictionaries representing Boto3 Instance objects
 representing the newly created instances.
        """
        try:
            instance_params = {
                "ImageId": image_id,
                "InstanceType": instance_type,
                "KeyName": key_pair_name,
            }
            if security_group_ids is not None:
                instance_params["SecurityGroupIds"] = security_group_ids

            response = self.ec2_client.run_instances(
                **instance_params, MinCount=1, MaxCount=1
            )
            instance = response["Instances"][0]
            self.instances.append(instance)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=[instance["InstanceId"]])
        except ClientError as err:
            params_str = "\n\t".join(
                f"{key}: {value}" for key, value in instance_params.items()
            )
            logger.error(
                f"Failed to complete instance creation request.\nRequest details:
{params_str}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InstanceLimitExceeded":
                logger.error(
                    (
                        f"Insufficient capacity for instance type '{instance_type}'.
 "
                        "Terminate unused instances or contact AWS Support for a
 limit increase."
                    )
                )
            if error_code == "InsufficientInstanceCapacity":
                logger.error(
                    (
```

```
                        f"Insufficient capacity for instance type '{instance_type}'.
"
                        "Select a different instance type or launch in a different
availability zone."
                    )
                )
            raise
        return self.instances
```

- Para obter detalhes da API, consulte a RunInstances Referência da API AWS SDK for Python (Boto3).

`StartInstances`

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
```

```
                            wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def start(self) -> Optional[Dict[str, Any]]:
        """
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=instance_ids)
            return start_response
        except ClientError as err:
            logger.error(
                f"Failed to start instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't start instance(s) because they are in an incorrect
state. "
```

```
                    "Ensure the instances are in a stopped state before starting
 them."
                )
            raise
```

- Para obter detalhes da API, consulte a StartInstances Referência da API AWS SDK for Python (Boto3).

`StopInstances`

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []
```

```
    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no instances
to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response = self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response
```

- Para obter detalhes da API, consulte a StopInstances Referência da API AWS SDK for Python (Boto3).

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
```

```
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
            raise
```

- Para obter detalhes da API, consulte a TerminateInstances Referência da API AWS SDK for Python (Boto3).

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
```

```
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example, such
as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
```

```
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
 balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port is
 open..."
            )
```

```
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access from
this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging from
your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer endpoint.
Please verify your VPC and security group settings."
            )
```

```
    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as sending
requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer endpoint.")
                endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
```

```
                    "Note that it can take a minute or two for the health check to
 update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
 failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
 failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a non-
existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
 self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
 instances."
        )
```

```
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
```

```
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
 Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
 created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up manually
 to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the demo.
    """
```

```
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM policies
and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
```

```
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
```

```
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.
```

```
        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
```

```
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to attach
 to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
```

```
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is assumed
by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
```

```
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
```

```
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
```

```
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
```

```
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
```

```
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
```

```
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
```

```
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.
```

```
        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with availability
zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
```

```
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) -> None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
```

```
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
 '{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
```

```
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
```

```
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress from
this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
```

```
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to either
this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist. Please
check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.
```

```
        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}' on
port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
```

```
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""
```

```
    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
and
        lower thresholds. In production, you might want to decrease the sensitivity
of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group '{target_group_name}'.")
```

```
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )
```

```
    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
```

```
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener to
the
                             load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
```

```
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
```

```
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")
```

```
    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint, retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
```

```
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
 and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books, movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client
```

```
    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended, such
as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5, "WriteCapacityUnits":
5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.
```

```
        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name, data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}: {err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Crie uma classe que envolva as ações do Systems Manager.

```
class ParameterHelper:
```

```
    """
    Encapsulates Systems Manager parameters. This example uses these parameters to
drive
    the demonstration of resilient architecture, such as failure of a dependency or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
```

```
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False. "
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones

- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Exemplos do Amazon ECR usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon ECR.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Amazon ECR

O exemplo de código a seguir mostra como começar a usar o Amazon ECR.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )

    image_names: [str] = []
    for page in page_iterator:
```

```
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Consulte detalhes da API em listImages na Referência de API do AWS SDK para Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um repositório do Amazon ECR.
- Defina políticas de repositório.
- Recupere URIs de repositórios.
- Obtenha tokens de autorização do Amazon ECR.
- Defina políticas de ciclo de vida para repositórios do Amazon ECR.
- Envie por push uma imagem do Docker para um repositório do Amazon ECR.
- Verifique a existência de uma imagem em um repositório do Amazon ECR.
- Liste os repositórios do Amazon ECR da conta e verifique os detalhes sobre eles.
- Exclua repositórios do Amazon ECR.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class ECRGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to perform basic operations using
    Amazon ECR.
    """

    def __init__(
        self,
        ecr_wrapper: ECRWrapper,
        docker_client: docker.DockerClient,
    ):
        self.ecr_wrapper = ecr_wrapper
        self.docker_client = docker_client
        self.tag = "echo-text"
        self.repository_name = "ecr-basics"
        self.docker_image = None
        self.full_tag_name = None
        self.repository = None

    def run(self, role_arn: str) -> None:
        """
        Runs the scenario.
        """
        print(
            """
The Amazon Elastic Container Registry (ECR) is a fully-managed Docker container
 registry
service provided by AWS. It allows developers and organizations to securely
store, manage, and deploy Docker container images.
ECR provides a simple and scalable way to manage container images throughout their
 lifecycle,
from building and testing to production deployment.
```

```
The `ECRWrapper' class is a wrapper for the Boto3 'ecr' client. The 'ecr' client
 provides a set of methods to
programmatically interact with the Amazon ECR service. This allows developers to
automate the storage, retrieval, and management of container images as part of their
 application
deployment pipelines. With ECR, teams can focus on building and deploying their
applications without having to worry about the underlying infrastructure required to
host and manage a container registry.

This scenario walks you through how to perform key operations for this service.
Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(
            f"""
* Create an ECR repository.

An ECR repository is a private Docker container repository provided
by Amazon Web Services (AWS). It is a managed service that makes it easy
to store, manage, and deploy Docker container images.
        """
        )
        print(f"Creating a repository named {self.repository_name}")
        self.repository = self.ecr_wrapper.create_repository(self.repository_name)
        print(f"The ARN of the ECR repository is
 {self.repository['repositoryArn']}")
        repository_uri = self.repository["repositoryUri"]
        press_enter_to_continue()
        print_dashes()

        print(
            f"""
* Build a Docker image.

Create a local Docker image if it does not already exist.
A Python Docker client is used to execute Docker commands.
You must have Docker installed and running.
            """
        )
        print(f"Building a docker image from 'docker_files/Dockerfile'")
        self.full_tag_name = f"{repository_uri}:{self.tag}"
        self.docker_image = self.docker_client.images.build(
```

```
            path="docker_files", tag=self.full_tag_name
        )[0]
        print(f"Docker image {self.full_tag_name} successfully built.")
        press_enter_to_continue()
        print_dashes()

        if role_arn is None:
            print(
                """
* Because an IAM role ARN was not provided, a role policy will not be set for this
 repository.
            """
            )
        else:
            print(
                """
* Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is crucial
 for maintaining
the security and integrity of your container images. The repository policy allows
 you to
define specific rules and restrictions for accessing and managing the images stored
 within your ECR
repository.
        """
            )

            self.grant_role_download_access(role_arn)
            print(f"Download access granted to the IAM role ARN {role_arn}")
            press_enter_to_continue()
            print_dashes()

            print(
                """
* Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
            """
            )

            policy_text =
 self.ecr_wrapper.get_repository_policy(self.repository_name)
            print("Policy Text:")
```

```
            print(f"{policy_text}")
            press_enter_to_continue()
            print_dashes()

        print(
            """
* Retrieve an ECR authorization token.

You need an authorization token to securely access and interact with the Amazon ECR
 registry.
The `get_authorization_token` method of the `ecr` client is responsible for securely
 accessing
and interacting with an Amazon ECR repository. This operation is responsible for
 obtaining a
valid authorization token, which is required to authenticate your requests to the
 ECR service.

Without a valid authorization token, you would not be able to perform any operations
 on the
ECR repository, such as pushing, pulling, or managing your Docker images.
        """
        )

        authorization_token = self.ecr_wrapper.get_authorization_token()
        print("Authorization token retrieved.")
        press_enter_to_continue()
        print_dashes()
        print(
            """
* Get the ECR Repository URI.

The URI  of an Amazon ECR repository is important. When you want to deploy a
 container image to
a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)
or Amazon Elastic Container Service (ECS), you need to specify the full image URI,
which includes the ECR repository URI. This allows the container runtime to pull the
correct container image from the ECR repository.
        """
        )
        repository_descriptions = self.ecr_wrapper.describe_repositories(
            [self.repository_name]
        )
        repository_uri = repository_descriptions[0]["repositoryUri"]
        print(f"Repository URI found: {repository_uri}")
```

```
        press_enter_to_continue()
        print_dashes()

        print(
            """
* Set an ECR Lifecycle Policy.

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored in
 your ECR repositories.
These policies allow you to automatically remove old or unused Docker images from
 your repositories,
freeing up storage space and reducing costs.

This example policy helps to maintain the size and efficiency of the container
 registry
by automatically removing older and potentially unused images, ensuring that the
storage is optimized and the registry remains up-to-date.
            """
        )
        press_enter_to_continue()
        self.put_expiration_policy()
        print(f"An expiration policy was added to the repository.")
        print_dashes()

        print(
            """
* Push a docker image to the Amazon ECR Repository.

The Docker client uses the authorization token is used to authenticate the when
 pushing the image to the
ECR repository.
        """
        )
        decoded_authorization =
 base64.b64decode(authorization_token).decode("utf-8")
        username, password = decoded_authorization.split(":")

        resp = self.docker_client.api.push(
            repository=repository_uri,
            auth_config={"username": username, "password": password},
            tag=self.tag,
            stream=True,
            decode=True,
        )
```

```
        for line in resp:
            print(line)

        print_dashes()

        print("* Verify if the image is in the ECR Repository.")
        image_descriptions = self.ecr_wrapper.describe_images(
            self.repository_name, [self.tag]
        )
        if len(image_descriptions) > 0:
            print("Image found in ECR Repository.")
        else:
            print("Image not found in ECR Repository.")
        press_enter_to_continue()
        print_dashes()

        print(
            "* As an optional step, you can interact with the image in Amazon ECR by
 using the CLI."
        )
        if q.ask(
            "Would you like to view instructions on how to use the CLI to run the
 image? (y/n)",
            q.is_yesno,
        ):
            print(
                f"""
1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you need
 to authenticate with the registry. You can do this using the AWS CLI:

    aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin {repository_uri.split("/")[0]}

2. Describe the image using this command:

   aws ecr describe-images --repository-name {self.repository_name} --image-ids
 imageTag={self.tag}

3. Run the Docker container and view the output using this command:

   docker run --rm {self.full_tag_name}
"""
            )
```

```
        self.cleanup(True)

    def cleanup(self, ask: bool):
        """
        Deletes the resources created in this scenario.
        :param ask: If True, prompts the user to confirm before deleting the
resources.
        """
        if self.repository is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the ECR repository
'{self.repository_name}? (y/n) "
            )
        ):
            print(f"Deleting the ECR repository '{self.repository_name}'.")
            self.ecr_wrapper.delete_repository(self.repository_name)

        if self.full_tag_name is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the local Docker image
'{self.full_tag_name}? (y/n) "
            )
        ):
            print(f"Deleting the docker image '{self.full_tag_name}'.")
            self.docker_client.images.remove(self.full_tag_name)

    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
```

```
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )


    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )



if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Run Amazon ECR getting started scenario."
    )
    parser.add_argument(
        "--iam-role-arn",
        type=str,
        default=None,
        help="an optional IAM role ARN that will be granted access to download
 images from a repository.",
        required=False,
```

```
    )
    parser.add_argument(
        "--no-art",
        action="store_true",
        help="accessibility setting that suppresses art in the console output.",
    )
    args = parser.parse_args()
    no_art = args.no_art
    iam_role_arn = args.iam_role_arn
    demo = None
    a_docker_client = None
    try:
        a_docker_client = docker.from_env()
        if not a_docker_client.ping():
            raise docker.errors.DockerException("Docker is not running.")
    except docker.errors.DockerException as err:
        logging.error(
            """
        The Python Docker client could not be created.
        Do you have Docker installed and running?
        Here is the error message:
        %s
        """,
            err,
        )
        sys.exit("Error with Docker.")
    try:
        an_ecr_wrapper = ECRWrapper.from_client()
        demo = ECRGettingStarted(an_ecr_wrapper, a_docker_client)
        demo.run(iam_role_arn)

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup(False)
```

Classe ECRWrapper que encapsula ações do Amazon ECR.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.
```

```
        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_repository_policy(self, repository_name: str) -> str:
        """
```

```
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.
```

```
        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
```

```
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Ações

`CreateRepository`

O código de exemplo a seguir mostra como usar `CreateRepository`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
 self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryAlreadyExistsException":
```

```
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a CreateRepository Referência da API AWS SDK for Python (Boto3).

`DeleteRepository`

O código de exemplo a seguir mostra como usar `DeleteRepository`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
```

```
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteRepository Referência da API AWS SDK for Python (Boto3).

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Para obter detalhes da API, consulte a DescribeImages Referência da API AWS SDK for Python (Boto3).

`DescribeRepositories`

O código de exemplo a seguir mostra como usar `DescribeRepositories`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
```

```
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeRepositories Referência da API AWS SDK for Python (Boto3).

`GetAuthorizationToken`

O código de exemplo a seguir mostra como usar `GetAuthorizationToken`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
```

```
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a GetAuthorizationToken Referência da API AWS SDK for Python (Boto3).

`GetRepositoryPolicy`

O código de exemplo a seguir mostra como usar `GetRepositoryPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a GetRepositoryPolicy Referência da API AWS SDK for Python (Boto3).

`PutLifeCyclePolicy`

O código de exemplo a seguir mostra como usar `PutLifeCyclePolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
 str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
 policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
```

```
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

Exemplo que coloca uma política de data de validade.

```
    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Para obter detalhes da API, consulte a PutLifeCyclePolicy Referência da API AWS SDK for Python (Boto3).

SetRepositoryPolicy

O código de exemplo a seguir mostra como usar `SetRepositoryPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
```

```
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

Exemplo que concede acesso de download a um perfil do IAM.

```
    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Para obter detalhes da API, consulte a SetRepositoryPolicy Referência da API AWS SDK for Python (Boto3).

Exemplos do Elastic Load Balancing Versão 2 usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with Elastic Load Balancing - Versão 2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Elastic Load Balancing

O exemplo de código a seguir mostra como começar a usar o Elastic Load Balancing.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
```

```
def hello_elbv2(elbv2_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Elastic Load Balancing V2 client
 and list
    up to ten of the load balancers for your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param elbv2_client: A Boto3 Elastic Load Balancing V2 client object.
    """
    print("Hello, Elastic Load Balancing! Let's list some of your load balancers:")
    load_balancers = elbv2_client.describe_load_balancers(PageSize=10).get(
        "LoadBalancers", []
    )
    if load_balancers:
        for lb in load_balancers:
            print(f"\t{lb['LoadBalancerName']}: {lb['DNSName']}")
    else:
        print("Your account doesn't have any load balancers.")


if __name__ == "__main__":
    hello_elbv2(boto3.client("elbv2"))
```

- Para obter detalhes da API, consulte a DescribeLoadBalancers Referência da API AWS SDK for Python (Boto3).

Ações

`CreateListener`

O código de exemplo a seguir mostra como usar `CreateListener`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
 the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
 listener for.
        :param target_group: An existing target group that is added as a listener to
 the
                              load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
```

```
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a CreateListener Referência da API AWS SDK for Python (Boto3).

`CreateLoadBalancer`

O código de exemplo a seguir mostra como usar `CreateLoadBalancer`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
 subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
```

```
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer
```

- Para obter detalhes da API, consulte a CreateLoadBalancer Referência da API AWS SDK for Python (Boto3).

`CreateTargetGroup`

O código de exemplo a seguir mostra como usar `CreateTargetGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
 how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
 and
        lower thresholds. In production, you might want to decrease the sensitivity
 of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
```

```
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a CreateTargetGroup Referência da API AWS SDK for Python (Boto3).

`DeleteLoadBalancer`

O código de exemplo a seguir mostra como usar `DeleteLoadBalancer`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
 {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
```

```
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DeleteLoadBalancer Referência da API AWS SDK for Python (Boto3).

`DeleteTargetGroup`

O código de exemplo a seguir mostra como usar `DeleteTargetGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
 self.elb_client.describe_target_groups(Names=[target_group_name])
```

```
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
```

```
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )
```

- Para obter detalhes da API, consulte a DeleteTargetGroup Referência da API AWS SDK for Python (Boto3).

`DescribeLoadBalancers`

O código de exemplo a seguir mostra como usar `DescribeLoadBalancers`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def get_endpoint(self, load_balancer_name) -> str:
        """
```

```
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte a DescribeLoadBalancers Referência da API AWS SDK for Python (Boto3).

`DescribeTargetHealth`

O código de exemplo a seguir mostra como usar `DescribeTargetHealth`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""
```

```
    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
correct region, "
                    "and ensure it has not been deleted or created in a different
account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

- Para obter detalhes da API, consulte a DescribeTargetHealth Referência da API AWS SDK for Python (Boto3).

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.
- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
```

```
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example, such
as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient service.")
```

```
        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
 balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
```

```
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port is
open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access from
this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging from
your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
```

```
            logging.error(
                "Couldn't get a successful response from the load balancer endpoint.
Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as sending
requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer endpoint.")
                endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
```

```
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check to
 update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
 failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
 failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a non-
existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
 self.recommendation.table_name)

        logging.info(
```

```
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
```

```
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up manually
to avoid unexpected charges."
            )


def main() -> None:
```

```
    """
    Main function to parse arguments and run the appropriate actions for the demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM policies
and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
```

```
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
```

```
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
```

```
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
```

```
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to attach
 to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
```

```
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is assumed
by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
```

```
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
```

```
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
```

```
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
```

```
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")
```

```
    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
```

```
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
```

```
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.
```

```
        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with availability
zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
```

```
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) -> None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
```

```
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
 '{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
```

```
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
```

```
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress from
this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
```

```
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to either
this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist. Please
check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.
```

```
        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}' on
port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.
```

```
        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
class ElasticLoadBalancerWrapper:
```

```
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
and
        lower thresholds. In production, you might want to decrease the sensitivity
of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
```

```
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )
```

```
    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
```

```
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener to
the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
```

```
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
```

```
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")
```

```
    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint, retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
```

```
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
 and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books, movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client
```

```
    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended, such
as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5, "WriteCapacityUnits":
5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.
```

```
        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name, data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}: {err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Crie uma classe que envolva as ações do Systems Manager.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters to
 drive
    the demonstration of resilient architecture, such as failure of a dependency or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                            service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
```

```
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False. "
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups

- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Exemplos do Amazon EMR usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon EMR.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AddJobFlowSteps`

O código de exemplo a seguir mostra como usar `AddJobFlowSteps`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Adicione uma etapa do Spark, que será executada pelo cluster assim que for adicionada.

```
def add_step(cluster_id, name, script_uri, script_args, emr_client):
    """
    Adds a job step to the specified cluster. This example adds a Spark
    step, which is run by the cluster as soon as it is added.

    :param cluster_id: The ID of the cluster.
    :param name: The name of the step.
    :param script_uri: The URI where the Python script is stored.
    :param script_args: Arguments to pass to the Python script.
    :param emr_client: The Boto3 EMR client object.
    :return: The ID of the newly added step.
    """
    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id,
            Steps=[
                {
                    "Name": name,
                    "ActionOnFailure": "CONTINUE",
                    "HadoopJarStep": {
                        "Jar": "command-runner.jar",
                        "Args": [
                            "spark-submit",
                            "--deploy-mode",
                            "cluster",
                            script_uri,
                            *script_args,
```

```
                        ],
                    },
                }
            ],
        )
        step_id = response["StepIds"][0]
        logger.info("Started step with ID %s", step_id)
    except ClientError:
        logger.exception("Couldn't start step %s with URI %s.", name, script_uri)
        raise
    else:
        return step_id
```

Execute um comando do Amazon EMR File System (EMRFS) como uma etapa de trabalho em um cluster. Isso pode ser usado para automatizar comandos do EMRFS em um cluster em vez de executar comandos manualmente por meio de uma conexão SSH.

```
import boto3
from botocore.exceptions import ClientError


def add_emrfs_step(command, bucket_url, cluster_id, emr_client):
    """
    Add an EMRFS command as a job flow step to an existing cluster.

    :param command: The EMRFS command to run.
    :param bucket_url: The URL of a bucket that contains tracking metadata.
    :param cluster_id: The ID of the cluster to update.
    :param emr_client: The Boto3 Amazon EMR client object.
    :return: The ID of the added job flow step. Status can be tracked by calling
             the emr_client.describe_step() function.
    """
    job_flow_step = {
        "Name": "Example EMRFS Command Step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
            "Jar": "command-runner.jar",
            "Args": ["/usr/bin/emrfs", command, bucket_url],
        },
    }
```

```
    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id, Steps=[job_flow_step]
        )
        step_id = response["StepIds"][0]
        print(f"Added step {step_id} to cluster {cluster_id}.")
    except ClientError:
        print(f"Couldn't add a step to cluster {cluster_id}.")
        raise
    else:
        return step_id


def usage_demo():
    emr_client = boto3.client("emr")
    # Assumes the first waiting cluster has EMRFS enabled and has created metadata
    # with the default name of 'EmrFSMetadata'.
    cluster = emr_client.list_clusters(ClusterStates=["WAITING"])["Clusters"][0]
    add_emrfs_step(
        "sync", "s3://elasticmapreduce/samples/cloudfront", cluster["Id"],
 emr_client
    )


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a AddJobFlowSteps Referência da API AWS SDK for Python (Boto3).

`DescribeCluster`

O código de exemplo a seguir mostra como usar `DescribeCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def describe_cluster(cluster_id, emr_client):
    """
    Gets detailed information about a cluster.

    :param cluster_id: The ID of the cluster to describe.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved cluster information.
    """
    try:
        response = emr_client.describe_cluster(ClusterId=cluster_id)
        cluster = response["Cluster"]
        logger.info("Got data for cluster %s.", cluster["Name"])
    except ClientError:
        logger.exception("Couldn't get data for cluster %s.", cluster_id)
        raise
    else:
        return cluster
```

- Para obter detalhes da API, consulte a DescribeCluster Referência da API AWS SDK for Python (Boto3).

`DescribeStep`

O código de exemplo a seguir mostra como usar `DescribeStep`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def describe_step(cluster_id, step_id, emr_client):
    """
    Gets detailed information about the specified step, including the current state
 of
    the step.

    :param cluster_id: The ID of the cluster.
    :param step_id: The ID of the step.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved information about the specified step.
    """
    try:
        response = emr_client.describe_step(ClusterId=cluster_id, StepId=step_id)
        step = response["Step"]
        logger.info("Got data for step %s.", step_id)
    except ClientError:
        logger.exception("Couldn't get data for step %s.", step_id)
        raise
    else:
        return step
```

- Para obter detalhes da API, consulte a DescribeStep Referência da API AWS SDK for Python (Boto3).

`ListSteps`

O código de exemplo a seguir mostra como usar `ListSteps`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_steps(cluster_id, emr_client):
    """
    Gets a list of steps for the specified cluster. In this example, all steps are
    returned, including completed and failed steps.

    :param cluster_id: The ID of the cluster.
    :param emr_client: The Boto3 EMR client object.
    :return: The list of steps for the specified cluster.
    """
    try:
        response = emr_client.list_steps(ClusterId=cluster_id)
        steps = response["Steps"]
        logger.info("Got %s steps for cluster %s.", len(steps), cluster_id)
    except ClientError:
        logger.exception("Couldn't get steps for cluster %s.", cluster_id)
        raise
    else:
        return steps
```

- Para obter detalhes da API, consulte a ListSteps Referência da API AWS SDK for Python (Boto3).

`RunJobFlow`

O código de exemplo a seguir mostra como usar `RunJobFlow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def run_job_flow(
    name,
    log_uri,
    keep_alive,
    applications,
    job_flow_role,
    service_role,
    security_groups,
    steps,
    emr_client,
):
    """
    Runs a job flow with the specified steps. A job flow creates a cluster of
    instances and adds steps to be run on the cluster. Steps added to the cluster
    are run as soon as the cluster is ready.

    This example uses the 'emr-5.30.1' release. A list of recent releases can be
    found here:
        https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-release-
components.html.

    :param name: The name of the cluster.
    :param log_uri: The URI where logs are stored. This can be an Amazon S3 bucket
 URL,
                    such as 's3://my-log-bucket'.
    :param keep_alive: When True, the cluster is put into a Waiting state after all
                       steps are run. When False, the cluster terminates itself when
                       the step queue is empty.
    :param applications: The applications to install on each instance in the
 cluster,
                         such as Hive or Spark.
    :param job_flow_role: The IAM role assumed by the cluster.
    :param service_role: The IAM role assumed by the service.
    :param security_groups: The security groups to assign to the cluster instances.
```

```
                            Amazon EMR adds all needed rules to these groups, so
                            they can be empty if you require only the default rules.
    :param steps: The job flow steps to add to the cluster. These are run in order
                  when the cluster is ready.
    :param emr_client: The Boto3 EMR client object.
    :return: The ID of the newly created cluster.
    """
    try:
        response = emr_client.run_job_flow(
            Name=name,
            LogUri=log_uri,
            ReleaseLabel="emr-5.30.1",
            Instances={
                "MasterInstanceType": "m5.xlarge",
                "SlaveInstanceType": "m5.xlarge",
                "InstanceCount": 3,
                "KeepJobFlowAliveWhenNoSteps": keep_alive,
                "EmrManagedMasterSecurityGroup": security_groups["manager"].id,
                "EmrManagedSlaveSecurityGroup": security_groups["worker"].id,
            },
            Steps=[
                {
                    "Name": step["name"],
                    "ActionOnFailure": "CONTINUE",
                    "HadoopJarStep": {
                        "Jar": "command-runner.jar",
                        "Args": [
                            "spark-submit",
                            "--deploy-mode",
                            "cluster",
                            step["script_uri"],
                            *step["script_args"],
                        ],
                    },
                }
                for step in steps
            ],
            Applications=[{"Name": app} for app in applications],
            JobFlowRole=job_flow_role.name,
            ServiceRole=service_role.name,
            EbsRootVolumeSize=10,
            VisibleToAllUsers=True,
        )
        cluster_id = response["JobFlowId"]
```

```
        logger.info("Created cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't create cluster.")
        raise
    else:
        return cluster_id
```

- Para obter detalhes da API, consulte a RunJobFlow Referência da API AWS SDK for Python (Boto3).

`TerminateJobFlows`

O código de exemplo a seguir mostra como usar `TerminateJobFlows`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def terminate_cluster(cluster_id, emr_client):
    """
    Terminates a cluster. This terminates all instances in the cluster and cannot
    be undone. Any data not saved elsewhere, such as in an Amazon S3 bucket, is
 lost.

    :param cluster_id: The ID of the cluster to terminate.
    :param emr_client: The Boto3 EMR client object.
    """
    try:
        emr_client.terminate_job_flows(JobFlowIds=[cluster_id])
        logger.info("Terminated cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't terminate cluster %s.", cluster_id)
        raise
```

- Para obter detalhes da API, consulte a TerminateJobFlows Referência da API AWS SDK for Python (Boto3).

Cenários

Criar um cluster do Amazon EMR de curta duração e executar uma etapa usando

O exemplo de código a seguir mostra como criar um cluster do Amazon EMR de curta duração que executa uma etapa e termina automaticamente após a conclusão dessa etapa.

SDK para Python (Boto3).

Crie um cluster do Amazon EMR de curta duração que estime o valor de pi usando o Apache Spark para paralelizar um grande número de cálculos. O trabalho grava a saída em logs do Amazon EMR e em um bucket do Amazon Simple Storage Service (Amazon S3). O cluster é encerrado automaticamente após a conclusão do trabalho.

- Crie um bucket do Amazon S3 e carregue um script de trabalho.
- Crie funções AWS Identity and Access Management (IAM).
- Crie grupos de segurança do Amazon Elastic Compute Cloud (Amazon EC2).
- Criar um cluster de curta duração e executar uma única etapa do trabalho.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon EMR

Executar um script shell para instalar bibliotecas

O exemplo de código a seguir mostra como usar AWS Systems Manager para executar um script de shell em instâncias do Amazon EMR que instala bibliotecas adicionais. Dessa forma, é possível automatizar o gerenciamento de instâncias em vez de executar comandos manualmente por meio de uma conexão SSH.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import argparse
import time
import boto3


def install_libraries_on_core_nodes(cluster_id, script_path, emr_client,
 ssm_client):
    """
    Copies and runs a shell script on the core nodes in the cluster.

    :param cluster_id: The ID of the cluster.
    :param script_path: The path to the script, typically an Amazon S3 object URL.
    :param emr_client: The Boto3 Amazon EMR client.
    :param ssm_client: The Boto3 AWS Systems Manager client.
    """
    core_nodes = emr_client.list_instances(
        ClusterId=cluster_id, InstanceGroupTypes=["CORE"]
    )["Instances"]
    core_instance_ids = [node["Ec2InstanceId"] for node in core_nodes]
    print(f"Found core instances: {core_instance_ids}.")

    commands = [
        # Copy the shell script from Amazon S3 to each node instance.
        f"aws s3 cp {script_path} /home/hadoop",
        # Run the shell script to install libraries on each node instance.
        "bash /home/hadoop/install_libraries.sh",
    ]
    for command in commands:
        print(f"Sending '{command}' to core instances...")
        command_id = ssm_client.send_command(
            InstanceIds=core_instance_ids,
            DocumentName="AWS-RunShellScript",
            Parameters={"commands": [command]},
            TimeoutSeconds=3600,
```

```
        )["Command"]["CommandId"]
        while True:
            # Verify the previous step succeeded before running the next step.
            cmd_result = ssm_client.list_commands(CommandId=command_id)["Commands"]
[0]
            if cmd_result["StatusDetails"] == "Success":
                print(f"Command succeeded.")
                break
            elif cmd_result["StatusDetails"] in ["Pending", "InProgress"]:
                print(f"Command status is {cmd_result['StatusDetails']},
 waiting...")
                time.sleep(10)
            else:
                print(f"Command status is {cmd_result['StatusDetails']}, quitting.")
                raise RuntimeError(
                    f"Command {command} failed to run. "
                    f"Details: {cmd_result['StatusDetails']}"
                )


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("cluster_id", help="The ID of the cluster.")
    parser.add_argument("script_path", help="The path to the script in Amazon S3.")
    args = parser.parse_args()

    emr_client = boto3.client("emr")
    ssm_client = boto3.client("ssm")

    install_libraries_on_core_nodes(
        args.cluster_id, args.script_path, emr_client, ssm_client
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ListInstances Referência da API AWS SDK for Python (Boto3).

EventBridge exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with EventBridge.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento agendado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função do Lambda o alvo.
- Concede permissão para permitir que a função Lambda seja EventBridge invocada.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações agendadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

EventBridge Exemplos de agendador usando SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with EventBridge Scheduler.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá EventBridge Scheduler

O exemplo de código a seguir mostra como começar a usar o EventBridge Scheduler.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_scheduler(scheduler_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EventBridge Scheduler
    client and list the schedules in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param scheduler_client: A Boto3 Amazon EventBridge Scheduler Client object.
 This object wraps
                             the low-level Amazon EventBridge Scheduler service API.
    """
    print("Hello, Amazon EventBridge Scheduler! Let's list some of your schedules:
\n")
    paginator = scheduler_client.get_paginator("list_schedules")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

    schedule_names: [str] = []
    for page in page_iterator:
        for schedule in page["Schedules"]:
            schedule_names.append(schedule["Name"])

    print(f"{len(schedule_names)} schedule(s) retrieved.")
    for schedule_name in schedule_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    hello_scheduler(boto3.client("scheduler"))
```

- Para obter detalhes da API, consulte a ListSchedules Referência da API AWS SDK for Python (Boto3).

Ações

`CreateSchedule`

O código de exemplo a seguir mostra como usar `CreateSchedule`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule(
        self,
        name: str,
        schedule_expression: str,
        schedule_group_name: str,
        target_arn: str,
        role_arn: str,
        input: str,
        delete_after_completion: bool = False,
        use_flexible_time_window: bool = False,
    ) -> str:
        """
        Creates a new schedule with the specified parameters.

        :param name: The name of the schedule.
        :param schedule_expression: The expression that defines when the schedule
 runs.
```

```
        :param schedule_group_name: The name of the schedule group.
        :param target_arn: The Amazon Resource Name (ARN) of the target.
        :param role_arn: The Amazon Resource Name (ARN) of the execution IAM role.
        :param input: The input for the target.
        :param delete_after_completion: Whether to delete the schedule after it
completes.
        :param use_flexible_time_window: Whether to use a flexible time window.

        :return The ARN of the created schedule.
        """
        try:
            hours_to_run = 1
            flexible_time_window_minutes = 10
            parameters = {
                "Name": name,
                "ScheduleExpression": schedule_expression,
                "GroupName": schedule_group_name,
                "Target": {"Arn": target_arn, "RoleArn": role_arn, "Input": input},
                "StartDate": datetime.now(timezone.utc),
                "EndDate": datetime.now(timezone.utc) +
timedelta(hours=hours_to_run),
            }

            if delete_after_completion:
                parameters["ActionAfterCompletion"] = "DELETE"

            if use_flexible_time_window:
                parameters["FlexibleTimeWindow"] = {
                    "Mode": "FLEXIBLE",
                    "MaximumWindowInMinutes": flexible_time_window_minutes,
                }
            else:
                parameters["FlexibleTimeWindow"] = {"Mode": "OFF"}

            response = self.scheduler_client.create_schedule(**parameters)
            return response["ScheduleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
```

```
                logger.error(
                    "Error creating schedule: %s", err.response["Error"]["Message"]
                )
            raise
```

- Para obter detalhes da API, consulte a CreateSchedule Referência da API AWS SDK for Python (Boto3).

`CreateScheduleGroup`

O código de exemplo a seguir mostra como usar `CreateScheduleGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule_group(self, name: str) -> str:
        """
```

```
        Creates a new schedule group with the specified name and description.

        :param name: The name of the schedule group.
        :param description: The description of the schedule group.

        :return: The ARN of the created schedule group.
        """
        try:
            response = self.scheduler_client.create_schedule_group(Name=name)
            return response["ScheduleGroupArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule group '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule group: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a CreateScheduleGroup Referência da API AWS SDK for Python (Boto3).

`DeleteSchedule`

O código de exemplo a seguir mostra como usar `DeleteSchedule`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def delete_schedule(self, name: str, schedule_group_name: str) -> None:
        """
        Deletes the schedule with the specified name and schedule group.

        :param name: The name of the schedule.
        :param schedule_group_name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule(
                Name=name, GroupName=schedule_group_name
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule with ID '%s' because the resource was
 not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule: %s", err.response["Error"]["Message"]
                )
                raise
```

- Para obter detalhes da API, consulte a DeleteSchedule Referência da API AWS SDK for Python (Boto3).

`DeleteScheduleGroup`

O código de exemplo a seguir mostra como usar `DeleteScheduleGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def delete_schedule_group(self, name: str) -> None:
        """
        Deletes the schedule group with the specified name.

        :param name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule_group(Name=name)
            logger.info("Schedule group %s deleted successfully.", name)
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule group with ID '%s' because the
resource was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule group: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a DeleteScheduleGroup Referência da API AWS SDK for Python (Boto3).

Cenários

Eventos agendados

O exemplo de código a seguir mostra como:

- Implante uma CloudFormation pilha com os recursos necessários.
- Crie um grupo de EventBridge agendamento do Scheduler.
- Crie uma programação única do EventBridge Scheduler com uma janela de tempo flexível.
- Crie uma programação recorrente do EventBridge Scheduler com uma taxa especificada.
- Exclua o EventBridge Scheduler, a programação e o grupo de agendamentos.
- Limpe os recursos e exclua a pilha.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class SchedulerScenario:
    """
    A scenario that demonstrates how to use Boto3 to schedule and receive events
 using
    the Amazon EventBridge Scheduler.
    """

    def __init__(
        self,
        scheduler_wrapper: SchedulerWrapper,
        cloud_formation_resource: ServiceResource,
    ):
        self.eventbridge_scheduler = scheduler_wrapper
        self.cloud_formation_resource = cloud_formation_resource
        self.stack: ServiceResource = None
        self.schedule_group_name = None
        self.sns_topic_arn = None
        self.role_arn = None

    def run(self) -> None:
        """
        Runs the scenario.
        """

        print(DASHES)
        print("Welcome to the Amazon EventBridge Scheduler Workflow.")
        print(DASHES)

        print(DASHES)
        self.prepare_application()
        print(DASHES)

        print(DASHES)
        self.create_one_time_schedule()
        print(DASHES)

        print(DASHES)
        self.create_recurring_schedule()
        print(DASHES)

        print(DASHES)
        if q.ask(
```

```
            "Do you want to delete all resources created by this workflow? (y/n) ",
            q.is_yesno,
        ):
            self.cleanup()
        print(DASHES)

        print("Amazon EventBridge Scheduler workflow completed.")

    def prepare_application(self) -> None:
        """
        Prepares the application by prompting the user setup information, deploying
a CloudFormation stack and
        creating a schedule group.
        """
        print("Preparing the application...")
        print(
            "\nThis example creates resources in a CloudFormation stack, including
an SNS topic"
            + "\nthat will be subscribed to the EventBridge Scheduler events. "
            + "\n\nYou will need to confirm the subscription in order to receive
event emails. "
        )

        email_address = q.ask("Enter an email address to use for event
subscriptions: ")
        stack_name = q.ask("Enter a name for the AWS Cloud Formation Stack: ")

        template_file = SchedulerScenario.get_template_as_string()

        parameters = [{"ParameterKey": "email", "ParameterValue": email_address}]

        self.stack = self.deploy_cloudformation_stack(
            stack_name, template_file, parameters
        )
        outputs = self.stack.outputs
        for output in outputs:
            if output.get("OutputKey") == "RoleARN":
                self.role_arn = output.get("OutputValue")
            elif output.get("OutputKey") == "SNStopicARN":
                self.sns_topic_arn = output.get("OutputValue")

        if not self.sns_topic_arn or not self.role_arn:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
```

```
            'sns_topic_arn'={self.sns_topic_arn}, 'role_arn'={self.role_arn}
            """
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"Stack output RoleARN: {self.role_arn}")
        print(f"Stack output SNStopicARN: a")
        schedule_group_name = "scenario-schedules-group"
        schedule_group_arn = self.eventbridge_scheduler.create_schedule_group(
            schedule_group_name
        )
        print(
            f"Successfully created schedule group '{self.schedule_group_name}':
{schedule_group_arn}."
        )
        self.schedule_group_name = schedule_group_name
        print("Application preparation complete.")

    def create_one_time_schedule(self) -> None:
        """
        Creates a one-time schedule to send an initial event.
        """
        schedule_name = q.ask("Enter a name for the one-time schedule:")

        scheduled_time = datetime.now(timezone.utc) + timedelta(minutes=1)
        formatted_scheduled_time = scheduled_time.strftime("%Y-%m-%dT%H:%M:%S")

        print(
            f"Creating a one-time schedule named '{schedule_name}' "
            + f"\nto send an initial event in 1 minute with a flexible time
window..."
        )

        schedule_arn = self.eventbridge_scheduler.create_schedule(
            schedule_name,
            f"at({formatted_scheduled_time})",
            self.schedule_group_name,
            self.sns_topic_arn,
            self.role_arn,
            f"One time scheduled event test from schedule {schedule_name}.",
            delete_after_completion=True,
            use_flexible_time_window=True,
        )
        print(
```

```
            f"Successfully created schedule '{schedule_name}' in schedule group
'scenario-schedules-group': {schedule_arn}."
        )
        print(f"Subscription email will receive an email from this event.")
        print(f"You must confirm your subscription to receive event emails.")
        print(f"One-time schedule '{schedule_name}' created successfully.")

    def create_recurring_schedule(self) -> None:
        """
        Create a recurring schedule to send events at a specified rate in minutes.
        """

        print("Creating a recurring schedule to send events for one hour...")
        schedule_name = q.ask("Enter a name for the recurring schedule: ")
        schedule_rate_in_minutes = q.ask(
            "Enter the desired schedule rate (in minutes): ", q.is_int
        )

        schedule_arn = self.eventbridge_scheduler.create_schedule(
            schedule_name,
            f"rate({schedule_rate_in_minutes} minutes)",
            self.schedule_group_name,
            self.sns_topic_arn,
            self.role_arn,
            f"Recurrent event test from schedule {schedule_name}.",
        )

        print(
            f"Successfully created schedule '{schedule_name}' in schedule group
'scenario-schedules-group': {schedule_arn}."
        )
        print(f"Subscription email will receive an email from this event.")
        print(f"You must confirm your subscription to receive event emails.")

        if q.ask(
            f"Are you ready to delete the '{schedule_name}' schedule? (y/n)",
q.is_yesno
        ):
            self.eventbridge_scheduler.delete_schedule(
                schedule_name, self.schedule_group_name
            )

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str, parameters: [dict[str, str]]
```

```
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `cfn_template.yaml` AWS CloudFormation script and
are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :param parameters: The parameters for the CloudFormation stack.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=parameters,
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example resources.
        """
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may take a
few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
```

```
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")

    def cleanup(self) -> None:
        """
        Deletes the CloudFormation stack and the resources created for the demo.
        """

        if self.schedule_group_name:
            schedule_group_name = self.schedule_group_name
            self.schedule_group_name = None
            self.eventbridge_scheduler.delete_schedule_group(schedule_group_name)
            print(f"Successfully deleted schedule group '{schedule_group_name}'.")

        if self.stack is not None:
            stack = self.stack
            self.stack = None
            self.destroy_cloudformation_stack(stack)
        print("Stack deleted, demo complete.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory, "cfn_template.yaml")
        file = open(template_file_path, "r")
        return file.read()


if __name__ == "__main__":
    demo: SchedulerScenario = None
    try:
        scheduler_wrapper = SchedulerWrapper.from_client()
        cloud_formation_resource = resource("cloudformation")
        demo = SchedulerScenario(scheduler_wrapper, cloud_formation_resource)
        demo.run()

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
```

```
            demo.cleanup()
```

SchedulerWrapper classe que envolve as ações do Amazon EventBridge Scheduler.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule(
        self,
        name: str,
        schedule_expression: str,
        schedule_group_name: str,
        target_arn: str,
        role_arn: str,
        input: str,
        delete_after_completion: bool = False,
        use_flexible_time_window: bool = False,
    ) -> str:
        """
        Creates a new schedule with the specified parameters.

        :param name: The name of the schedule.
        :param schedule_expression: The expression that defines when the schedule
 runs.
        :param schedule_group_name: The name of the schedule group.
        :param target_arn: The Amazon Resource Name (ARN) of the target.
        :param role_arn: The Amazon Resource Name (ARN) of the execution IAM role.
```

```
        :param input: The input for the target.
        :param delete_after_completion: Whether to delete the schedule after it
completes.
        :param use_flexible_time_window: Whether to use a flexible time window.

        :return The ARN of the created schedule.
        """
        try:
            hours_to_run = 1
            flexible_time_window_minutes = 10
            parameters = {
                "Name": name,
                "ScheduleExpression": schedule_expression,
                "GroupName": schedule_group_name,
                "Target": {"Arn": target_arn, "RoleArn": role_arn, "Input": input},
                "StartDate": datetime.now(timezone.utc),
                "EndDate": datetime.now(timezone.utc) +
timedelta(hours=hours_to_run),
            }

            if delete_after_completion:
                parameters["ActionAfterCompletion"] = "DELETE"

            if use_flexible_time_window:
                parameters["FlexibleTimeWindow"] = {
                    "Mode": "FLEXIBLE",
                    "MaximumWindowInMinutes": flexible_time_window_minutes,
                }
            else:
                parameters["FlexibleTimeWindow"] = {"Mode": "OFF"}

            response = self.scheduler_client.create_schedule(**parameters)
            return response["ScheduleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule: %s", err.response["Error"]["Message"]
                )
```

```
            raise


    def delete_schedule(self, name: str, schedule_group_name: str) -> None:
        """
        Deletes the schedule with the specified name and schedule group.

        :param name: The name of the schedule.
        :param schedule_group_name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule(
                Name=name, GroupName=schedule_group_name
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule with ID '%s' because the resource was
not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule: %s", err.response["Error"]["Message"]
                )
                raise


    def create_schedule_group(self, name: str) -> str:
        """
        Creates a new schedule group with the specified name and description.

        :param name: The name of the schedule group.
        :param description: The description of the schedule group.

        :return: The ARN of the created schedule group.
        """
        try:
            response = self.scheduler_client.create_schedule_group(Name=name)
            return response["ScheduleGroupArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
```

```
                    "Failed to create schedule group '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule group: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def delete_schedule_group(self, name: str) -> None:
        """
        Deletes the schedule group with the specified name.

        :param name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule_group(Name=name)
            logger.info("Schedule group %s deleted successfully.", name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule group with ID '%s' because the
resource was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule group: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateSchedule

- CreateScheduleGroup
- DeleteSchedule
- DeleteScheduleGroups

Exemplos do Amazon Glacier usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Glacier.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateVault`

O código de exemplo a seguir mostra como usar `CreateVault`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
```

```
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault
```

- Para obter detalhes da API, consulte a CreateVault Referência da API AWS SDK for Python (Boto3).

`DeleteArchive`

O código de exemplo a seguir mostra como usar `DeleteArchive`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id, archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

- Para obter detalhes da API, consulte a DeleteArchive Referência da API AWS SDK for Python (Boto3).

`DeleteVault`

O código de exemplo a seguir mostra como usar `DeleteVault`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteVault Referência da API AWS SDK for Python (Boto3).

`DeleteVaultNotifications`

O código de exemplo a seguir mostra como usar `DeleteVaultNotifications`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def stop_notifications(notification):
        """
        Stops notifications to the configured Amazon SNS topic.

        :param notification: The notification configuration to remove.
        """
        try:
            notification.delete()
            logger.info("Notifications stopped.")
        except ClientError:
            logger.exception("Couldn't stop notifications.")
            raise
```

- Para obter detalhes da API, consulte a DeleteVaultNotifications Referência da API AWS SDK for Python (Boto3).

`DescribeJob`

O código de exemplo a seguir mostra como usar `DescribeJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_status(job):
        """
        Gets the status of a job.

        :param job: The job to query.
        :return: The current status of the job.
        """
        try:
            job.load()
            logger.info(
                "Job %s is performing action %s and has status %s.",
                job.id,
                job.action,
                job.status_code,
            )
        except ClientError:
            logger.exception("Couldn't get status for job %s.", job.id)
            raise
        else:
            return job.status_code
```

- Para obter detalhes da API, consulte a DescribeJob Referência da API AWS SDK for Python (Boto3).

`GetJobOutput`

O código de exemplo a seguir mostra como usar `GetJobOutput`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
 response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes
```

- Para obter detalhes da API, consulte a GetJobOutput Referência da API AWS SDK for Python (Boto3).

`GetVaultNotifications`

O código de exemplo a seguir mostra como usar `GetVaultNotifications`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_notification(vault):
        """
        Gets the currently notification configuration for a vault.

        :param vault: The vault to query.
        :return: The notification configuration for the specified vault.
        """
        try:
            notification = vault.Notification()
            logger.info(
                "Vault %s notifies %s on %s events.",
                vault.name,
                notification.sns_topic,
```

```
                notification.events,
            )
        except ClientError:
            logger.exception("Couldn't get notification data for %s.", vault.name)
            raise
        else:
            return notification
```

- Para obter detalhes da API, consulte a GetVaultNotifications Referência da API AWS SDK for Python (Boto3).

`InitiateJob`

O código de exemplo a seguir mostra como usar `InitiateJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recupere um inventário do cofre.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_inventory_retrieval(vault):
        """
        Initiates an inventory retrieval job. The inventory describes the contents
        of the vault. Standard retrievals typically complete within 3–5 hours.
```

```
        When the job completes, you can get the inventory by calling get_output().

        :param vault: The vault to inventory.
        :return: The inventory retrieval job.
        """
        try:
            job = vault.initiate_inventory_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on vault %s.", vault.name)
            raise
        else:
            return job
```

Recupere um arquivo de um cofre.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically complete
        within 3–5 hours. When the job completes, you can get the archive contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
```

```
            raise
        else:
            return job
```

- Para obter detalhes da API, consulte a InitiateJob Referência da API AWS SDK for Python (Boto3).

ListJobs

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
```

```
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
            raise
        else:
            return job_list
```

- Para obter detalhes da API, consulte a ListJobs Referência da API AWS SDK for Python (Boto3).

`ListVaults`

O código de exemplo a seguir mostra como usar `ListVaults`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
```

```
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

- Para obter detalhes da API, consulte a ListVaults Referência da API AWS SDK for Python (Boto3).

`SetVaultNotifications`

O código de exemplo a seguir mostra como usar `SetVaultNotifications`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
```

```
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn, vault.name
            )
            raise
        else:
            return notification
```

- Para obter detalhes da API, consulte a SetVaultNotifications Referência da API AWS SDK for Python (Boto3).

`UploadArchive`

O código de exemplo a seguir mostra como usar `UploadArchive`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
```

```
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive
```

- Para obter detalhes da API, consulte a UploadArchive Referência da API AWS SDK for Python (Boto3).

Cenários

Arquivar um arquivo, obter notificações e iniciar um trabalho

O exemplo de código a seguir mostra como:

- Criar um cofre do Amazon Glacier.
- Configurar o cofre para publicar notificações em um tópico do Amazon SNS.
- Fazer upload de um arquivo para o cofre.
- Iniciar um trabalho de recuperação de arquivo.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva operações do Amazon Glacier.

```
import argparse
import logging
import os
```

```
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

```
    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically complete
        within 3-5 hours. When the job completes, you can get the archive contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
```

```
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
            raise
        else:
            return job_list


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
```

```
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn, vault.name
            )
            raise
        else:
            return notification
```

Chame funções na classe wrapper para criar um cofre e fazer upload de um arquivo, depois configure o cofre para publicar notificações e iniciar um trabalho para recuperar o arquivo.

```
def upload_demo(glacier, vault_name, topic_arn):
    """
    Shows how to:
    * Create a vault.
    * Configure the vault to publish notifications to an Amazon SNS topic.
    * Upload an archive.
```

```
    * Start a job to retrieve the archive.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to create.
    :param topic_arn: The ARN of an Amazon SNS topic that receives notification of
                      Amazon S3 Glacier events.
    """
    print(f"\nCreating vault {vault_name}.")
    vault = glacier.create_vault(vault_name)
    print("\nList of vaults in your account:")
    glacier.list_vaults()
    print(f"\nUploading glacier_basics.py to {vault.name}.")
    with open("glacier_basics.py", "rb") as upload_file:
        archive = glacier.upload_archive(vault, "glacier_basics.py", upload_file)
    print(
        "\nStarting an archive retrieval request to get the file back from the "
        "vault."
    )
    glacier.initiate_archive_retrieval(archive)
    print("\nListing in progress jobs:")
    glacier.list_jobs(vault, "in_progress")
    print(
        "\nBecause Amazon S3 Glacier is intended for infrequent retrieval, an "
        "archive request with Standard retrieval typically completes within 3-5 "
        "hours."
    )
    if topic_arn:
        notification = glacier.set_notifications(vault, topic_arn)
        print(
            f"\nVault {vault.name} is configured to notify the "
            f"{notification.sns_topic} topic when {notification.events} "
            f"events occur. You can subscribe to this topic to receive "
            f"a message when the archive retrieval completes.\n"
        )
    else:
        print(
            f"\nVault {vault.name} is not configured to notify an Amazon SNS topic "
            f"when the archive retrieval completes so wait a few hours."
        )
    print("\nRetrieve your job output by running this script with the --retrieve
flag.")
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateVault
  - InitiateJob
  - ListJobs
  - ListVaults
  - SetVaultNotifications
  - UploadArchive

Obter o conteúdo do arquivo e excluir o arquivo

O exemplo de código a seguir mostra como:

- Listar trabalhos para um cofre do Amazon Glacier e obter o status do trabalho.
- Obter a saída de um trabalho de recuperação de arquivo concluído.
- Excluir um arquivo.
- Excluir um cofre.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que envolva operações do Amazon Glacier.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
            raise
        else:
            return job_list
```

```
    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id, archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

```
    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

Chame funções na classe wrapper para obter o conteúdo de arquivo de um trabalho concluído e, em seguida, exclua o arquivo.

```
def retrieve_demo(glacier, vault_name):
    """
    Shows how to:
    * List jobs for a vault and get job status.
    * Get the output of a completed archive retrieval job.
    * Delete an archive.
    * Delete a vault.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to query for jobs.
    """
    vault = glacier.glacier_resource.Vault("-", vault_name)
    try:
        vault.load()
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            print(
                f"\nVault {vault_name} doesn't exist. You must first run this script
 "
                f"with the --upload flag to create the vault."
            )
            return
        else:
            raise
```

```
    print(f"\nGetting completed jobs for {vault.name}.")
    jobs = glacier.list_jobs(vault, "completed")
    if not jobs:
        print("\nNo completed jobs found. Give it some time and try again later.")
        return

    retrieval_job = None
    for job in jobs:
        if job.action == "ArchiveRetrieval" and job.status_code == "Succeeded":
            retrieval_job = job
            break
    if retrieval_job is None:
        print(
            "\nNo ArchiveRetrieval jobs found. Give it some time and try again "
            "later."
        )
        return

    print(f"\nGetting output from job {retrieval_job.id}.")
    archive_bytes = glacier.get_job_output(retrieval_job)
    archive_str = archive_bytes.decode("utf-8")
    print("\nGot archive data. Printing the first 10 lines.")
    print(os.linesep.join(archive_str.split(os.linesep)[:10]))

    print(f"\nDeleting the archive from {vault.name}.")
    archive = glacier.glacier_resource.Archive(
        "-", vault.name, retrieval_job.archive_id
    )
    glacier.delete_archive(archive)

    print(f"\nDeleting {vault.name}.")
    glacier.delete_vault(vault)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DeleteArchive
  - DeleteVault
  - GetJobOutput

- ListJobs

AWS Glue exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
from botocore.exceptions import ClientError


def hello_glue():
    """
    Lists the job definitions in your AWS Glue account, using the AWS SDK for Python
 (Boto3).
    """
    try:
        # Create the Glue client
        glue = boto3.client("glue")

        # List the jobs, limiting the results to 10 per page
        paginator = glue.get_paginator("get_jobs")
        response_iterator = paginator.paginate(
            PaginationConfig={"MaxItems": 10, "PageSize": 10}
        )

        # Print the job names
        print("Here are the jobs in your account:")
        for page in response_iterator:
            for job in page["Jobs"]:
                print(f"\t{job['Name']}")

    except ClientError as e:
        print(f"Error: {e}")


if __name__ == "__main__":
    hello_glue()
```

- Para obter detalhes da API, consulte a ListJobs Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que agrupe as AWS Glue funções usadas no cenário.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

```
    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job that
can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is run
as
                                part of the job. The script defines how the data is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
```

```
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_job_run(self, name, input_database, input_table, output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by the
            # Python ETL script to determine the location of input and output data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]


    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_crawler(self, name):
```

```
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Crie uma classe que execute o cenário.

```
class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job and
 use
    them to transform data from CSV to JSON format.
    """

    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
        :param glue_service_role: An AWS Identity and Access Management (IAM) role
                                  that AWS Glue can assume to gain access to the
                                  resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output data
                            from AWS Glue job runs.
        """
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
```

```
    def wait(seconds, tick=12):
        """
        Waits for a specified number of seconds, while also displaying an animated
        spinner.

        :param seconds: The number of seconds to wait.
        :param tick: The number of frames per second used to animate the spinner.
        """
        progress = "|/-\\"
        waited = 0
        while waited < seconds:
            for frame in range(tick):
                sys.stdout.write(f"\r{progress[frame % len(progress)]}")
                sys.stdout.flush()
                time.sleep(1 / tick)
            waited += 1

    def upload_job_script(self, job_script):
        """
        Uploads a Python ETL script to an S3 bucket. The script is used by the AWS
Glue
        job to transform data.

        :param job_script: The relative path to the job script.
        """
        try:
            self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
            print(f"Uploaded job script '{job_script}' to the example bucket.")
        except S3UploadFailedError as err:
            logger.error("Couldn't upload job script. Here's why: %s", err)
            raise

    def run(self, crawler_name, db_name, db_prefix, data_source, job_script,
job_name):
        """
        Runs the scenario. This is an interactive experience that runs at a command
        prompt and asks you for input throughout.

        :param crawler_name: The name of the crawler used in the scenario. If the
                             crawler does not exist, it is created.
        :param db_name: The name to give the metadata database created by the
crawler.
        :param db_prefix: The prefix to give tables added to the database by the
                          crawler.
```

```
        :param data_source: The location of the data source that is targeted by the
                            crawler and extracted during job runs.
        :param job_script: The job script that is used to transform data during job
                           runs.
        :param job_name: The name to give the job definition that is created during
the
                         scenario.
        """
        wrapper = GlueWrapper(self.glue_client)
        print(f"Checking for crawler {crawler_name}.")
        crawler = wrapper.get_crawler(crawler_name)
        if crawler is None:
            print(f"Creating crawler {crawler_name}.")
            wrapper.create_crawler(
                crawler_name,
                self.glue_service_role.arn,
                db_name,
                db_prefix,
                data_source,
            )
            print(f"Created crawler {crawler_name}.")
            crawler = wrapper.get_crawler(crawler_name)
        pprint(crawler)
        print("-" * 88)

        print(
            f"When you run the crawler, it crawls data stored in {data_source} and "
            f"creates a metadata database in the AWS Glue Data Catalog that
describes "
            f"the data in the data source."
        )
        print("In this example, the source data is in CSV format.")
        ready = False
        while not ready:
            ready = Question.ask_question(
                "Ready to start the crawler? (y/n) ", Question.is_yesno
            )
        wrapper.start_crawler(crawler_name)
        print("Let's wait for the crawler to run. This typically takes a few
minutes.")
        crawler_state = None
        while crawler_state != "READY":
            self.wait(10)
            crawler = wrapper.get_crawler(crawler_name)
```

```
            crawler_state = crawler["State"]
            print(f"Crawler is {crawler['State']}.")
        print("-" * 88)

        database = wrapper.get_database(db_name)
        print(f"The crawler created database {db_name}:")
        pprint(database)
        print(f"The database contains these tables:")
        tables = wrapper.get_tables(db_name)
        for index, table in enumerate(tables):
            print(f"\t{index + 1}. {table['Name']}")
        table_index = Question.ask_question(
            f"Enter the number of a table to see more detail: ",
            Question.is_int,
            Question.in_range(1, len(tables)),
        )
        pprint(tables[table_index - 1])
        print("-" * 88)

        print(f"Creating job definition {job_name}.")
        wrapper.create_job(
            job_name,
            "Getting started example job.",
            self.glue_service_role.arn,
            f"s3://{self.glue_bucket.name}/{job_script}",
        )
        print("Created job definition.")
        print(
            f"When you run the job, it extracts data from {data_source}, transforms
it "
            f"by using the {job_script} script, and loads the output into "
            f"S3 bucket {self.glue_bucket.name}."
        )
        print(
            "In this example, the data is transformed from CSV to JSON, and only a
few "
            "fields are included in the output."
        )
        job_run_status = None
        if Question.ask_question(f"Ready to run? (y/n) ", Question.is_yesno):
            job_run_id = wrapper.start_job_run(
                job_name, db_name, tables[0]["Name"], self.glue_bucket.name
            )
            print(f"Job {job_name} started. Let's wait for it to run.")
```

```
            while job_run_status not in ["SUCCEEDED", "STOPPED", "FAILED",
"TIMEOUT"]:
                self.wait(10)
                job_run = wrapper.get_job_run(job_name, job_run_id)
                job_run_status = job_run["JobRunState"]
                print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
        print("-" * 88)

        if job_run_status == "SUCCEEDED":
            print(
                f"Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':"
            )
            try:
                keys = [
                    obj.key for obj in
self.glue_bucket.objects.filter(Prefix="run-")
                ]
                for index, key in enumerate(keys):
                    print(f"\t{index + 1}: {key}")
                lines = 4
                key_index = Question.ask_question(
                    f"Enter the number of a block to download it and see the first
{lines} "
                    f"lines of JSON output in the block: ",
                    Question.is_int,
                    Question.in_range(1, len(keys)),
                )
                job_data = io.BytesIO()
                self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)
                job_data.seek(0)
                for _ in range(lines):
                    print(job_data.readline().decode("utf-8"))
            except ClientError as err:
                logger.error(
                    "Couldn't get job run data. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
            print("-" * 88)

        job_names = wrapper.list_jobs()
        if job_names:
```

```
            print(f"Your account has {len(job_names)} jobs defined:")
            for index, job_name in enumerate(job_names):
                print(f"\t{index + 1}. {job_name}")
            job_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_names)} to see the list of
runs for "
                f"a job: ",
                Question.is_int,
                Question.in_range(1, len(job_names)),
            )
            job_runs = wrapper.get_job_runs(job_names[job_index - 1])
            if job_runs:
                print(f"Found {len(job_runs)} runs for job {job_names[job_index -
1]}:")
                for index, job_run in enumerate(job_runs):
                    print(
                        f"\t{index + 1}. {job_run['JobRunState']} on "
                        f"{job_run['CompletedOn']:%Y-%m-%d %H:%M:%S}"
                    )
                run_index = Question.ask_question(
                    f"Enter a number between 1 and {len(job_runs)} to see details
for a run: ",
                    Question.is_int,
                    Question.in_range(1, len(job_runs)),
                )
                pprint(job_runs[run_index - 1])
            else:
                print(f"No runs found for job {job_names[job_index - 1]}")
        else:
            print("Your account doesn't have any jobs defined.")
        print("-" * 88)

        print(
            f"Let's clean up. During this example we created job definition
'{job_name}'."
        )
        if Question.ask_question(
            "Do you want to delete the definition and all runs? (y/n) ",
            Question.is_yesno,
        ):
            wrapper.delete_job(job_name)
            print(f"Job definition '{job_name}' deleted.")
        tables = wrapper.get_tables(db_name)
        print(f"We also created database '{db_name}' that contains these tables:")
```

```
        for table in tables:
            print(f"\t{table['Name']}")
        if Question.ask_question(
            "Do you want to delete the tables and the database? (y/n) ",
            Question.is_yesno,
        ):
            for table in tables:
                wrapper.delete_table(db_name, table["Name"])
                print(f"Deleted table {table['Name']}.")
            wrapper.delete_database(db_name)
            print(f"Deleted database {db_name}.")
        print(f"We also created crawler '{crawler_name}'.")
        if Question.ask_question(
            "Do you want to delete the crawler? (y/n) ", Question.is_yesno
        ):
            wrapper.delete_crawler(crawler_name)
            print(f"Deleted crawler {crawler_name}.")
        print("-" * 88)


def parse_args(args):
    """
    Parse command line arguments.

    :param args: The command line arguments.
    :return: The parsed arguments.
    """
    parser = argparse.ArgumentParser(
        description="Runs the AWS Glue getting started with crawlers and jobs
 scenario. "
        "Before you run this scenario, set up scaffold resources by running "
        "'python scaffold.py deploy'."
    )
    parser.add_argument(
        "role_name",
        help="The name of an IAM role that AWS Glue can assume. This role must grant
 access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole "
        "managed policy.",
    )
    parser.add_argument(
        "bucket_name",
        help="The name of an S3 bucket that AWS Glue can access to get the job
 script and "
```

```
        "put job results.",
    )
    parser.add_argument(
        "--job_script",
        default="flight_etl_job_script.py",
        help="The name of the job script file that is used in the scenario.",
    )
    return parser.parse_args(args)


def main():
    args = parse_args(sys.argv[1:])
    try:
        print("-" * 88)
        print(
            "Welcome to the AWS Glue getting started with crawlers and jobs
 scenario."
        )
        print("-" * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client("glue"),
            boto3.resource("iam").Role(args.role_name),
            boto3.resource("s3").Bucket(args.bucket_name),
        )
        scenario.upload_job_script(args.job_script)
        scenario.run(
            "doc-example-crawler",
            "doc-example-database",
            "doc-example-",
            "s3://crawler-public-us-east-1/flight/2016/csv",
            args.job_script,
            "doc-example-job",
        )
        print("-" * 88)
        print(
            "To destroy scaffold resources, including the IAM role and S3 bucket "
            "used in this scenario, run 'python scaffold.py destroy'."
        )
        print("\nThanks for watching!")
        print("-" * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")
```

Crie um script ETL que seja usado pelo AWS Glue para extrair, transformar e carregar dados durante a execução do trabalho.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the data
 to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
```

```
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable

- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
```

```
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateCrawler Referência da API AWS SDK for Python (Boto3).

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job that
 can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is run
 as
                                part of the job. The script defines how the data is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
```

```
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateJob Referência da API AWS SDK for Python (Boto3).

`DeleteCrawler`

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_crawler(self, name):
        """
```

```
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteCrawler Referência da API AWS SDK for Python (Boto3).

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteDatabase Referência da API AWS SDK for Python (Boto3).

`DeleteJob`

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
```

```
        """
        self.glue_client = glue_client


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteJob Referência da API AWS SDK for Python (Boto3).

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTable Referência da API AWS SDK for Python (Boto3).

`GetCrawler`

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler
```

- Para obter detalhes da API, consulte a GetCrawler Referência da API AWS SDK for Python (Boto3).

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

- Para obter detalhes da API, consulte a GetDatabase Referência da API AWS SDK for Python (Boto3).

`GetJobRun`

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]
```

- Para obter detalhes da API, consulte a GetJobRun Referência da API AWS SDK for Python (Boto3).

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
```

```
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]
```

- Para obter detalhes da API, consulte a GetJobRuns Referência da API AWS SDK for Python (Boto3).

`GetTables`

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
```

```
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]
```

- Para obter detalhes da API, consulte a GetTables Referência da API AWS SDK for Python (Boto3).

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]
```

- Para obter detalhes da API, consulte a ListJobs Referência da API AWS SDK for Python (Boto3).

`StartCrawler`

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
```

```
        """
        self.glue_client = glue_client


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a StartCrawler Referência da API AWS SDK for Python (Boto3).

`StartJobRun`

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_job_run(self, name, input_database, input_table, output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by the
            # Python ETL script to determine the location of input and output data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return response["JobRunId"]
```

- Para obter detalhes da API, consulte a StartJobRun Referência da API AWS SDK for Python (Boto3).

HealthImaging exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with HealthImaging.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá HealthImaging

O exemplo de código a seguir mostra como começar a usar o HealthImaging.

SDK para Python (Boto3)

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)


def hello_medical_imaging(medical_imaging_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS HealthImaging
    client and list the data stores in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param medical_imaging_client: A Boto3 AWS HealthImaging Client object.
    """
    print("Hello, Amazon Health Imaging! Let's list some of your data stores:\n")
    try:
        paginator = medical_imaging_client.get_paginator("list_datastores")
        page_iterator = paginator.paginate()
        datastore_summaries = []
        for page in page_iterator:
            datastore_summaries.extend(page["datastoreSummaries"])
        print("\tData Stores:")
        for ds in datastore_summaries:
            print(f"\t\tDatastore: {ds['datastoreName']} ID {ds['datastoreId']}")
    except ClientError as err:
        logger.error(
            "Couldn't list data stores. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise


if __name__ == "__main__":
    hello_medical_imaging(boto3.client("medical-imaging"))
```

- Para obter detalhes da API, consulte a ListDatastores Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Ações

`CopyImageSet`

O código de exemplo a seguir mostra como usar `CopyImageSet`.

SDK para Python (Boto3)

Função de utilitário para copiar um conjunto de imagens.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def copy_image_set(
        self,
        datastore_id,
        image_set_id,
        version_id,
        destination_image_set_id=None,
        destination_version_id=None,
        force=False,
        subsets=[],
    ):
        """
        Copy an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param destination_image_set_id: The ID of the optional destination image
 set.
        :param destination_version_id: The ID of the optional destination image set
 version.
        :param force: Force the copy.
        :param subsets: The optional subsets to copy. For example:
 ["12345678901234567890123456789012"].
        :return: The copied image set ID.
        """
        try:
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }
```

```
            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }
            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"][
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }
            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't copy image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return copy_results["destinationImageSetProperties"]["imageSetId"]
```

Copiar um conjunto de imagens sem um destino.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }
```

```
            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Copiar um conjunto de imagens com um destino.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Copiar um subconjunto de um conjunto de imagens.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"][
                        subset
```

```
                ] = {}

            copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                "copiableAttributes": json.dumps(copySubsetsJson)
            }

        copy_results = self.health_imaging_client.copy_image_set(
            datastoreId=datastore_id,
            sourceImageSetId=image_set_id,
            copyImageSetInformation=copy_image_set_information,
            force=force,
        )
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a CopyImageSet Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`CreateDatastore`

O código de exemplo a seguir mostra como usar `CreateDatastore`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def create_datastore(self, name):
```

```
        """
        Create a data store.

        :param name: The name of the data store to create.
        :return: The data store ID.
        """
        try:
            data_store =
self.health_imaging_client.create_datastore(datastoreName=name)
        except ClientError as err:
            logger.error(
                "Couldn't create data store %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreId"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a CreateDatastore Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteDatastore`

O código de exemplo a seguir mostra como usar `DeleteDatastore`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_datastore(self, datastore_id):
        """
        Delete a data store.

        :param datastore_id: The ID of the data store.
        """
        try:
            self.health_imaging_client.delete_datastore(datastoreId=datastore_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete data store %s. Here's why: %s: %s",
                datastore_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a DeleteDatastore Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteImageSet`

O código de exemplo a seguir mostra como usar `DeleteImageSet`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The delete results.
        """
        try:
            delete_results = self.health_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delete_results
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a DeleteImageSet Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDICOMImportJob`

O código de exemplo a seguir mostra como usar `GetDICOMImportJob`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_dicom_import_job(self, datastore_id, job_id):
        """
        Get the properties of a DICOM import job.

        :param datastore_id: The ID of the data store.
        :param job_id: The ID of the job.
        :return: The job properties.
        """
        try:
            job = self.health_imaging_client.get_dicom_import_job(
                jobId=job_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobProperties"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetDICOMImportJob Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDatastore`

O código de exemplo a seguir mostra como usar `GetDatastore`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_datastore_properties(self, datastore_id):
        """
        Get the properties of a data store.

        :param datastore_id: The ID of the data store.
        :return: The data store properties.
        """
        try:
            data_store = self.health_imaging_client.get_datastore(
                datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get data store %s. Here's why: %s: %s",
                id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return data_store["datastoreProperties"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetDatastore Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageFrame`

O código de exemplo a seguir mostra como usar `GetImageFrame`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K encoded
 pixel data.
        :param datastore_id: The ID of the data store.
```

```
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.health_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetImageFrame Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSet`

O código de exemplo a seguir mostra como usar `GetImageSet`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetImageSet Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSetMetadata`

O código de exemplo a seguir mostra como usar `GetImageSetMetadata`.

SDK para Python (Boto3)

Função de utilitário para obter metadados do conjunto de imagens.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """
        try:
            if version_id:
                image_set_metadata =
 self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
```

```
                )
            else:

                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            print(image_set_metadata)
            with open(metadata_file, "wb") as f:
                for chunk in
image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Obter metadados do conjunto de imagens sem versão.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
```

Obter metadados do conjunto de imagens com versão.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a GetImageSetMetadata Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDICOMImportJobs`

O código de exemplo a seguir mostra como usar `ListDICOMImportJobs`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_dicom_import_jobs(self, datastore_id):
        """
        List the DICOM import jobs.

        :param datastore_id: The ID of the data store.
        :return: The list of jobs.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_dicom_import_jobs"
            )
            page_iterator = paginator.paginate(datastoreId=datastore_id)
            job_summaries = []
            for page in page_iterator:
                job_summaries.extend(page["jobSummaries"])
        except ClientError as err:
```

```
            logger.error(
                "Couldn't list DICOM import jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job_summaries
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a ListDICOMImportJobs Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDatastores`

O código de exemplo a seguir mostra como usar `ListDatastores`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_datastores(self):
        """
        List the data stores.

        :return: The list of data stores.
```

```
        """
        try:
            paginator = self.health_imaging_client.get_paginator("list_datastores")
            page_iterator = paginator.paginate()
            datastore_summaries = []
            for page in page_iterator:
                datastore_summaries.extend(page["datastoreSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list data stores. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return datastore_summaries
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a ListDatastores Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListImageSetVersions`

O código de exemplo a seguir mostra como usar `ListImageSetVersions`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
```

```
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_image_set_versions(self, datastore_id, image_set_id):
        """
        List the image set versions.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The list of image set versions.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_image_set_versions"
            )
            page_iterator = paginator.paginate(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
            image_set_properties_list = []
            for page in page_iterator:
                image_set_properties_list.extend(page["imageSetPropertiesList"])
        except ClientError as err:
            logger.error(
                "Couldn't list image set versions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set_properties_list
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a ListImageSetVersions Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListTagsForResource`

O código de exemplo a seguir mostra como usar `ListTagsForResource`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a ListTagsForResource Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`SearchImageSets`

O código de exemplo a seguir mostra como usar `SearchImageSets`.

SDK para Python (Boto3)

A função de utilitário para pesquisar conjuntos de imagens.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
 [{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
 self.health_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
```

```
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries
```

Caso de uso nº 1: operador EQUAL.

```
        search_filter = {
            "filters": [
                {"operator": "EQUAL", "values": [{"DICOMPatientId": patient_id}]}
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(f"Image sets found with EQUAL operator\n{image_sets}")
```

Caso de uso #2: operador BETWEEN usando DICOMStudyDate DICOMStudyTime e.

```
        search_filter = {
            "filters": [
                {
                    "operator": "BETWEEN",
                    "values": [
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "19900101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "20230101",
```

```
                                "DICOMStudyTime": "000000",
                            }
                        },
                    ],
                }
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with BETWEEN operator using DICOMStudyDate and
 DICOMStudyTime\n{image_sets}"
        )
```

Caso de uso nº 3: operador BETWEEN usando o createdAt. Os estudos de tempo foram previamente persistidos.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "createdAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "createdAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                }
            ]
        }

        recent_image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with with BETWEEN operator using createdAt
\n{recent_image_sets}"
        )
```

Caso de uso #4: operador EQUAL on DICOMSeriesInstanceUID e BETWEEN em updatedAt e classifique a resposta na ordem ASC no campo updatedAt.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "updatedAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "updatedAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                },
                {
                    "values": [{"DICOMSeriesInstanceUID": series_instance_uid}],
                    "operator": "EQUAL",
                },
            ],
            "sort": {
                "sortOrder": "ASC",
                "sortField": "updatedAt",
            },
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            "Image sets found with EQUAL operator on DICOMSeriesInstanceUID and
BETWEEN on updatedAt and"
        )
        print(f"sort response in ASC order on updatedAt field\n{image_sets}")
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a SearchImageSets Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

StartDICOMImportJob

O código de exemplo a seguir mostra como usar `StartDICOMImportJob`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def start_dicom_import_job(
        self, job_name, datastore_id, role_arn, input_s3_uri, output_s3_uri
    ):
        """
        Start a DICOM import job.

        :param job_name: The name of the job.
        :param datastore_id: The ID of the data store.
        :param role_arn: The Amazon Resource Name (ARN) of the role to use for the
 job.
        :param input_s3_uri: The S3 bucket input prefix path containing the DICOM
 files.
        :param output_s3_uri: The S3 bucket output prefix path for the result.
        :return: The job ID.
        """
        try:
            job = self.health_imaging_client.start_dicom_import_job(
                jobName=job_name,
                datastoreId=datastore_id,
                dataAccessRoleArn=role_arn,
```

```
                inputS3Uri=input_s3_uri,
                outputS3Uri=output_s3_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a StartDICOMImportJob Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
```

```
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a TagResource Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`UntagResource`

O código de exemplo a seguir mostra como usar `UntagResource`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
```

```
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para obter detalhes da API, consulte a UntagResource Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`UpdateImageSetMetadata`

O código de exemplo a seguir mostra como usar `UpdateImageSetMetadata`.

SDK para Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def update_image_set_metadata(
        self, datastore_id, image_set_id, version_id, metadata, force=False
    ):
        """
        Update the metadata of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param metadata: The image set metadata as a dictionary.
            For example {"DICOMUpdates": {"updatableAttributes":
            "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientName\":
\"Garcia^Gloria\"}}}"}}
        :param: force: Force the update.
        :return: The updated image set metadata.
        """
        try:
            updated_metadata = self.health_imaging_client.update_image_set_metadata(
                imageSetId=image_set_id,
                datastoreId=datastore_id,
                latestVersionId=version_id,
                updateImageSetMetadataUpdates=metadata,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update image set metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return updated_metadata
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

Caso de uso 1: insira ou atualize um atributo.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"updatableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Caso de uso 2: remova um atributo.

```
            # Attribute key and value must match the existing attribute.
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Caso de uso 3: remova uma instância.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "Series": {
                            "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                            }
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Caso de uso 4: reverta para uma versão anterior.

```
            metadata = {"revertToVersionId": "1"}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

- Para obter detalhes da API, consulte a UpdateImageSetMetadata Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cenários

Começar a usar conjuntos de imagens e quadros de imagem

O exemplo de código a seguir mostra como importar arquivos DICOM e baixar quadros de imagem em HealthImaging.

A implementação é estruturada como uma aplicação da linha de comando.

- Configure recursos para uma importação DICOM.
- Importe arquivos DICOM para um armazenamento de dados.
- Recupere os IDs do conjunto de imagens para o trabalho de importação.
- Recupere os IDs do quadro de imagem para o conjunto de imagens.
- Baixe, decodifique e verifique os quadros de imagem.
- Limpe recursos.

SDK para Python (Boto3)

Crie uma CloudFormation pilha com os recursos necessários.

```
    def deploy(self):
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `setup.yaml` AWS CloudFormation script and are
 deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.
        """

        print("\t\tLet's deploy the stack for resource creation.")
        stack_name = q.ask("\t\tEnter a name for the stack: ", q.non_empty)

        data_store_name = q.ask(
            "\t\tEnter a name for the Health Imaging Data Store: ", q.non_empty
        )

        account_id = boto3.client("sts").get_caller_identity()["Account"]

        with open(
            "../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml"
```

```
        ) as setup_file:
            setup_template = setup_file.read()
        print(f"\t\tCreating {stack_name}.")
        stack = self.cf_resource.create_stack(
            StackName=stack_name,
            TemplateBody=setup_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=[
                {
                    "ParameterKey": "datastoreName",
                    "ParameterValue": data_store_name,
                },
                {
                    "ParameterKey": "userAccountID",
                    "ParameterValue": account_id,
                },
            ],
        )
        print("\t\tWaiting for stack to deploy. This typically takes a minute or
two.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_create_complete")
        waiter.wait(StackName=stack.name)
        stack.load()
        print(f"\t\tStack status: {stack.stack_status}")

        outputs_dictionary = {
            output["OutputKey"]: output["OutputValue"] for output in stack.outputs
        }
        self.input_bucket_name = outputs_dictionary["BucketName"]
        self.output_bucket_name = outputs_dictionary["BucketName"]
        self.role_arn = outputs_dictionary["RoleArn"]
        self.data_store_id = outputs_dictionary["DatastoreID"]
        return stack
```

Copie arquivos DICOM para o bucket de importação do Amazon S3.

```
    def copy_single_object(self, key, source_bucket, target_bucket,
target_directory):
        """
        Copies a single object from a source to a target bucket.
```

```
        :param key: The key of the object to copy.
        :param source_bucket: The source bucket for the copy.
        :param target_bucket: The target bucket for the copy.
        :param target_directory: The target directory for the copy.
        """
        new_key = target_directory + "/" + key
        copy_source = {"Bucket": source_bucket, "Key": key}
        self.s3_client.copy_object(
            CopySource=copy_source, Bucket=target_bucket, Key=new_key
        )
        print(f"\n\t\tCopying {key}.")

    def copy_images(
        self, source_bucket, source_directory, target_bucket, target_directory
    ):
        """
        Copies the images from the source to the target bucket using multiple
threads.

        :param source_bucket: The source bucket for the images.
        :param source_directory: Directory within the source bucket.
        :param target_bucket: The target bucket for the images.
        :param target_directory: Directory within the target bucket.
        """

        # Get list of all objects in source bucket.
        list_response = self.s3_client.list_objects_v2(
            Bucket=source_bucket, Prefix=source_directory
        )
        objs = list_response["Contents"]
        keys = [obj["Key"] for obj in objs]

        # Copy the objects in the bucket.
        for key in keys:
            self.copy_single_object(key, source_bucket, target_bucket,
target_directory)

        print("\t\tDone copying all objects.")
```

Importe os arquivos DICOM para o armazenamento de dados do Amazon S3.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def start_dicom_import_job(
        self,
        data_store_id,
        input_bucket_name,
        input_directory,
        output_bucket_name,
        output_directory,
        role_arn,
    ):
        """
        Routine which starts a HealthImaging import job.

        :param data_store_id: The HealthImaging data store ID.
        :param input_bucket_name: The name of the Amazon S3 bucket containing the
 DICOM files.
        :param input_directory: The directory in the S3 bucket containing the DICOM
 files.
        :param output_bucket_name: The name of the S3 bucket for the output.
        :param output_directory: The directory in the S3 bucket to store the output.
        :param role_arn: The ARN of the IAM role with permissions for the import.
        :return: The job ID of the import.
        """
```

```
        input_uri = f"s3://{input_bucket_name}/{input_directory}/"
        output_uri = f"s3://{output_bucket_name}/{output_directory}/"
        try:
            job = self.medical_imaging_client.start_dicom_import_job(
                jobName="examplejob",
                datastoreId=data_store_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_uri,
                outputS3Uri=output_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

Obtenha conjuntos de imagens criados pelo trabalho de importação DICOM.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)
```

```
    def get_image_sets_for_dicom_import_job(self, datastore_id, import_job_id):
        """
        Retrieves the image sets created for an import job.

        :param datastore_id: The HealthImaging data store ID
        :param import_job_id: The import job ID
        :return: List of image set IDs
        """

        import_job = self.medical_imaging_client.get_dicom_import_job(
            datastoreId=datastore_id, jobId=import_job_id
        )

        output_uri = import_job["jobProperties"]["outputS3Uri"]

        bucket = output_uri.split("/")[2]
        key = "/".join(output_uri.split("/")[3:])

        # Try to get the manifest.
        retries = 3
        while retries > 0:
            try:
                obj = self.s3_client.get_object(
                    Bucket=bucket, Key=key + "job-output-manifest.json"
                )
                body = obj["Body"]
                break
            except ClientError as error:
                retries = retries - 1
                time.sleep(3)
        try:
            data = json.load(body)
            expression =
jmespath.compile("jobSummary.imageSetsSummary[].imageSetId")
            image_sets = expression.search(data)
        except json.decoder.JSONDecodeError as error:
            image_sets = import_job["jobProperties"]

        return image_sets


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
```

```
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set
```

Obtenha informações sobre os quadros de imagem de conjuntos de imagens.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
```

```
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_frames_for_image_set(self, datastore_id, image_set_id,
out_directory):
        """
        Get the image frames for an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param out_directory: The directory to save the file.
        :return: The image frames.
        """
        image_frames = []
        file_name = os.path.join(out_directory,
f"{image_set_id}_metadata.json.gzip")
        file_name = file_name.replace("/", "\\\\")
        self.get_image_set_metadata(file_name, datastore_id, image_set_id)
        try:
            with gzip.open(file_name, "rb") as f_in:
                doc = json.load(f_in)
            instances = jmespath.search("Study.Series.*.Instances[].*[]", doc)
            for instance in instances:
                rescale_slope = jmespath.search("DICOM.RescaleSlope", instance)
                rescale_intercept = jmespath.search("DICOM.RescaleIntercept",
instance)
                image_frames_json = jmespath.search("ImageFrames[][]", instance)
                for image_frame in image_frames_json:
                    checksum_json = jmespath.search(
                        "max_by(PixelDataChecksumFromBaseToFullResolution, &Width)",
                        image_frame,
                    )
                    image_frame_info = {
                        "imageSetId": image_set_id,
                        "imageFrameId": image_frame["ID"],
                        "rescaleIntercept": rescale_intercept,
                        "rescaleSlope": rescale_slope,
                        "minPixelValue": image_frame["MinPixelValue"],
```

```
                        "maxPixelValue": image_frame["MaxPixelValue"],
                        "fullResolutionChecksum": checksum_json["Checksum"],
                    }
                    image_frames.append(image_frame_info)
            return image_frames
        except TypeError:
            return {}
        except ClientError as err:
            logger.error(
                "Couldn't get image frames for image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        return image_frames


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """

        try:
            if version_id:
                image_set_metadata =
self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set_metadata =
self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            with open(metadata_file, "wb") as f:
```

```
                for chunk in
 image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Baixe, decodifique e verifique os quadros de imagem.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.
```

```
        :param file_path_to_write: The path to write the image frame's HTJ2K encoded
 pixel data.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.medical_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def download_decode_and_check_image_frames(
        self, data_store_id, image_frames, out_directory
    ):
        """
        Downloads image frames, decodes them, and uses the checksum to validate
        the decoded images.

        :param data_store_id: The HealthImaging data store ID.
        :param image_frames: A list of dicts containing image frame information.
        :param out_directory: A directory for the downloaded images.
        :return: True if the function succeeded; otherwise, False.
        """
        total_result = True
        for image_frame in image_frames:
            image_file_path = f"{out_directory}/
image_{image_frame['imageFrameId']}.jph"
            self.get_pixel_data(
                image_file_path,
                data_store_id,
                image_frame["imageSetId"],
```

```
                image_frame["imageFrameId"],
            )

            image_array = self.jph_image_to_opj_bitmap(image_file_path)
            crc32_checksum = image_frame["fullResolutionChecksum"]
            # Verify checksum.
            crc32_calculated = zlib.crc32(image_array)
            image_result = crc32_checksum == crc32_calculated
            print(
                f"\t\tImage checksum verified for {image_frame['imageFrameId']}:
{image_result }"
            )
            total_result = total_result and image_result
        return total_result

    @staticmethod
    def jph_image_to_opj_bitmap(jph_file):
        """
        Decode the image to a bitmap using an OPENJPEG library.
        :param jph_file: The file to decode.
        :return: The decoded bitmap as an array.
        """
        # Use format 2 for the JPH file.
        params = openjpeg.utils.get_parameters(jph_file, 2)
        print(f"\n\t\tImage parameters for {jph_file}: \n\t\t{params}")

        image_array = openjpeg.utils.decode(jph_file, 2)

        return image_array
```

Limpe recursos.

```
    def destroy(self, stack):
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example resources.
        """
```

```
        print(f"\t\tCleaning up resources and {stack.name}.")
        data_store_id = None
        for oput in stack.outputs:
            if oput["OutputKey"] == "DatastoreID":
                data_store_id = oput["OutputValue"]
        if data_store_id is not None:
            print(f"\t\tDeleting image sets in data store {data_store_id}.")
            image_sets = self.medical_imaging_wrapper.search_image_sets(
                data_store_id, {}
            )
            image_set_ids = [image_set["imageSetId"] for image_set in image_sets]

            for image_set_id in image_set_ids:
                self.medical_imaging_wrapper.delete_image_set(
                    data_store_id, image_set_id
                )
                print(f"\t\tDeleted image set with id : {image_set_id}")

        print(f"\t\tDeleting {stack.name}.")
        stack.delete()
        print("\t\tWaiting for stack removal. This may take a few minutes.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_delete_complete")
        waiter.wait(StackName=stack.name)
        print("\t\tStack delete complete.")




class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)
```

```
    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
[{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
self.medical_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        """
        try:
            delete_results = self.medical_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Marcar um datastore

O exemplo de código a seguir mostra como marcar um armazenamento HealthImaging de dados.

SDK para Python (Boto3)

Marcar um datastore.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.tag_resource(data_store_arn, {"Deployment":
 "Development"})
```

A função de utilitário para marcar um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Listar tags para um datastore.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.list_tags_for_resource(data_store_arn)
```

A função de utilitário para listar as tags de um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
```

```
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

Desmarcar um datastore.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.untag_resource(data_store_arn, ["Deployment"])
```

A função de utilitário para desmarcar um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
```

```
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Marcar um conjunto de imagens

O exemplo de código a seguir mostra como marcar um conjunto de HealthImaging imagens.

SDK para Python (Boto3)

Marcar um conjunto de imagens

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.tag_resource(image_set_arn, {"Deployment":
 "Development"})
```

A função de utilitário para marcar um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Listar tags para um conjunto de imagens

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
```

```
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.list_tags_for_resource(image_set_arn)
```

A função de utilitário para listar as tags de um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

Desmarcar um conjunto de imagens

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )
```

```
    medical_imaging_wrapper.untag_resource(image_set_arn, ["Deployment"])
```

A função de utilitário para desmarcar um recurso.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o MedicalImagingWrapper objeto.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - ListTagsForResource
  - TagResource

- UntagResource

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

HealthLake exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with HealthLake.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`Crie um repositório de dados FHIR`

O código de exemplo a seguir mostra como usar `CreateFHIRDatastore`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
```

```
        return cls(health_lake_client)


    def create_fhir_datastore(
        self,
        datastore_name: str,
        sse_configuration: dict[str, any] = None,
        identity_provider_configuration: dict[str, any] = None,
    ) -> dict[str, str]:
        """
        Creates a new HealthLake data store.
        When creating a SMART on FHIR data store, the following parameters are
required:
        - sse_configuration: The server-side encryption configuration for a SMART on
FHIR-enabled data store.
        - identity_provider_configuration: The identity provider configuration for a
SMART on FHIR-enabled data store.

        :param datastore_name: The name of the data store.
        :param sse_configuration: The server-side encryption configuration for a
SMART on FHIR-enabled data store.
        :param identity_provider_configuration: The identity provider configuration
for a SMART on FHIR-enabled data store.
        :return: A dictionary containing the data store information.
        """
        try:
            parameters = {"DatastoreName": datastore_name, "DatastoreTypeVersion":
"R4"}
            if (
                sse_configuration is not None
                and identity_provider_configuration is not None
            ):
                # Creating a SMART on FHIR-enabled data store
                parameters["SseConfiguration"] = sse_configuration
                parameters[
                    "IdentityProviderConfiguration"
                ] = identity_provider_configuration

            response = self.health_lake_client.create_fhir_datastore(**parameters)
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't create data store %s. Here's why %s",
                datastore_name,
```

```
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir mostra um exemplo de parâmetros para um SMART no armazenamento FHIR-enabled HealthLake de dados.

```
            sse_configuration = {
                "KmsEncryptionConfig": {"CmkType": "AWS_OWNED_KMS_KEY"}
            }
            # TODO: Update the metadata to match your environment.
            metadata = {
                "issuer": "https://ehr.example.com",
                "jwks_uri": "https://ehr.example.com/.well-known/jwks.json",
                "authorization_endpoint": "https://ehr.example.com/auth/authorize",
                "token_endpoint": "https://ehr.token.com/auth/token",
                "token_endpoint_auth_methods_supported": [
                    "client_secret_basic",
                    "foo",
                ],
                "grant_types_supported": ["client_credential", "foo"],
                "registration_endpoint": "https://ehr.example.com/auth/register",
                "scopes_supported": ["openId", "profile", "launch"],
                "response_types_supported": ["code"],
                "management_endpoint": "https://ehr.example.com/user/manage",
                "introspection_endpoint": "https://ehr.example.com/user/introspect",
                "revocation_endpoint": "https://ehr.example.com/user/revoke",
                "code_challenge_methods_supported": ["S256"],
                "capabilities": [
                    "launch-ehr",
                    "sso-openid-connect",
                    "client-public",
                ],
            }
            # TODO: Update the IdpLambdaArn.
            identity_provider_configuration = {
                "AuthorizationStrategy": "SMART_ON_FHIR_V1",
                "FineGrainedAuthorizationEnabled": True,
                "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-
id:function:your-lambda-name",
                "Metadata": json.dumps(metadata),
```

```
            }
            data_store = self.create_fhir_datastore(
                datastore_name, sse_configuration, identity_provider_configuration
            )
```

- Consulte detalhes da API em CreateFHIRDatastore na Referência de API do AWS SDK para Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir armazenamento de dados FHIR

O código de exemplo a seguir mostra como usar `DeleteFHIRDatastore`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def delete_fhir_datastore(self, datastore_id: str) -> None:
        """
        Deletes a HealthLake data store.
        :param datastore_id: The data store ID.
        """
        try:
            self.health_lake_client.delete_fhir_datastore(DatastoreId=datastore_id)
        except ClientError as err:
            logger.exception(
```

```
                "Couldn't delete data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Consulte detalhes da API em DeleteFHIRDatastore na Referência de API do AWS SDK para Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Descreva o armazenamento de dados FHIR

O código de exemplo a seguir mostra como usar `DescribeFHIRDatastore`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_datastore(self, datastore_id: str) -> dict[str, any]:
        """
        Describes a HealthLake data store.
        :param datastore_id: The data store ID.
        :return: The data store description.
        """
        try:
```

```
            response = self.health_lake_client.describe_fhir_datastore(
                DatastoreId=datastore_id
            )
            return response["DatastoreProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Consulte detalhes da API em DescribeFHIRDatastore na Referência de API do AWS SDK para Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DescribeFHIRExportJob`

O código de exemplo a seguir mostra como usar `DescribeFHIRExportJob`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_export_job(
```

```
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake export job.
        :param datastore_id: The data store ID.
        :param job_id: The export job ID.
        :return: The export job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_export_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ExportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe export job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeFHIRExportJob Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DescribeFHIRImportJob`

O código de exemplo a seguir mostra como usar `DescribeFHIRImportJob`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.
```

```
        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_import_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake import job.
        :param datastore_id: The data store ID.
        :param job_id: The import job ID.
        :return: The import job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_import_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ImportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe import job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeFHIRImportJob Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar armazenamentos de dados FHIR

O código de exemplo a seguir mostra como usar `ListFHIRDatastores`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_datastores(self) -> list[dict[str, any]]:
        """
        Lists all HealthLake data stores.
        :return: A list of data store descriptions.
        """
        try:
            next_token = None
            datastores = []

            # Loop through paginated results.
            while True:
                parameters = {}
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_datastores(**parameters)
                datastores.extend(response["DatastorePropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break

            return datastores
        except ClientError as err:
            logger.exception(
```

```
                "Couldn't list data stores. Here's why %s", err.response["Error"]
["Message"]
            )
            raise
```

- Consulte detalhes da API em ListFHIRDatastores na Referência de API do AWS SDK para Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListFHIRExportJobs`

O código de exemplo a seguir mostra como usar `ListFHIRExportJobs`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_export_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
```

```
        Lists HealthLake export jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The export job name.
        :param job_status: The export job status.
        :param submitted_before: The export job submitted before the specified date.
        :param submitted_after: The export job submitted after the specified date.
        :return: A list of export jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
                jobs.extend(response["ExportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListFHIRExportJobs Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListFHIRImportJobs`

O código de exemplo a seguir mostra como usar `ListFHIRImportJobs`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_import_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake import jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The import job name.
        :param job_status: The import job status.
        :param submitted_before: The import job submitted before the specified date.
        :param submitted_after: The import job submitted after the specified date.
        :return: A list of import jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
```

```
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_import_jobs(**parameters)
                jobs.extend(response["ImportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list import jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListFHIRImportJobs Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListTagsForResource`

O código de exemplo a seguir mostra como usar `ListTagsForResource`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
        """
        Lists the tags for a HealthLake resource.
        :param resource_arn: The resource ARN.
        :return: The tags for the resource.
        """
        try:
            response = self.health_lake_client.list_tags_for_resource(
                ResourceARN=resource_arn
            )
            return response["Tags"]
        except ClientError as err:
            logger.exception(
                "Couldn't list tags for resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListTagsForResource Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

StartFHIRExportJob

O código de exemplo a seguir mostra como usar `StartFHIRExportJob`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_export_job(
        self,
        job_name: str,
        datastore_id: str,
        output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake export job.
        :param job_name: The export job name.
        :param datastore_id: The data store ID.
        :param output_s3_uri: The output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The export job.
        """
        try:
            response = self.health_lake_client.start_fhir_export_job(
```

```
                OutputDataConfig={
                    "S3Configuration": {"S3Uri": output_s3_uri, "KmsKeyId":
kms_key_id}
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
                JobName=job_name,
            )

            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start export job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a StartFHIRExportJob Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`StartFHIRImportJob`

O código de exemplo a seguir mostra como usar `StartFHIRImportJob`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
```

```
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_import_job(
        self,
        job_name: str,
        datastore_id: str,
        input_s3_uri: str,
        job_output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake import job.
        :param job_name: The import job name.
        :param datastore_id: The data store ID.
        :param input_s3_uri: The input S3 URI.
        :param job_output_s3_uri: The job output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The import job.
        """
        try:
            response = self.health_lake_client.start_fhir_import_job(
                JobName=job_name,
                InputDataConfig={"S3Uri": input_s3_uri},
                JobOutputDataConfig={
                    "S3Configuration": {
                        "S3Uri": job_output_s3_uri,
                        "KmsKeyId": kms_key_id,
                    }
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
            )
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start import job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a StartFHIRImportJob Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def tag_resource(self, resource_arn: str, tags: list[dict[str, str]]) -> None:
        """
        Tags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tags: The tags to add to the resource.
        """
        try:
            self.health_lake_client.tag_resource(ResourceARN=resource_arn,
Tags=tags)
        except ClientError as err:
            logger.exception(
                "Couldn't tag resource %s. Here's why %s",
                resource_arn,
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a TagResource Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`UntagResource`

O código de exemplo a seguir mostra como usar `UntagResource`.

SDK para Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tag_keys: The tag keys to remove from the resource.
        """
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
```

```
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UntagResource Referência da API AWS SDK for Python (Boto3).

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exemplos de IAM usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def main():
    """
    Lists the managed policies in your AWS account using the AWS SDK for Python
 (Boto3).
    """
    iam = boto3.client("iam")

    try:
        # Get a paginator for the list_policies operation
        paginator = iam.get_paginator("list_policies")

        # Iterate through the pages of results
        for page in paginator.paginate(Scope="All", OnlyAttached=False):
            for policy in page["Policies"]:
                print(f"Policy name: {policy['PolicyName']}")
                print(f"  Policy ARN: {policy['Arn']}")
    except boto3.exceptions.BotoCoreError as e:
        print(f"Encountered an error while listing policies: {e}")
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ListPolicies Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário e um perfil do IAM que conceda permissão para listar os buckets do Amazon S3. O usuário só tem direitos para assumir a função. Após assumir a função, use credenciais temporárias para listar os buckets para a conta.

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, roles, and policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
```

```
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
```

```
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume "
            f"the role."
        )
    except ClientError as error:
        print(
            f"Couldn't create an inline policy for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise
```

```
    print("Give AWS time to propagate these new resources and connections.", end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id, aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
    Uses the temporary credentials from the role to list the buckets that are owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
 role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
```

```
        "sts", aws_access_key_id=user_key.id, aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
 credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise




def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
```

```
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn, "AssumeRoleDemoSession")
```

```
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Anexar uma política a uma função usando o objeto Policy do Boto3.

```
def attach_to_role(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

Anexar uma política a uma função usando o objeto Role do Boto3.

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
```

```
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

- Para obter detalhes da API, consulte a AttachRolePolicy Referência da API AWS SDK for Python (Boto3).

`AttachUserPolicy`

O código de exemplo a seguir mostra como usar `AttachUserPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise
```

- Para obter detalhes da API, consulte a AttachUserPolicy Referência da API AWS SDK for Python (Boto3).

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- Para obter detalhes da API, consulte a CreateAccessKey Referência da API AWS SDK for Python (Boto3).

`CreateAccountAlias`

O código de exemplo a seguir mostra como usar `CreateAccountAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

- Para obter detalhes da API, consulte a CreateAccountAlias Referência da API AWS SDK for Python (Boto3).

`CreateInstanceProfile`

O código de exemplo a seguir mostra como usar `CreateInstanceProfile`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo cria uma política, um perfil e um perfil de instância e vincula uns aos outros.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
```

```
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is assumed
by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
```

```
                                        the role, such as AmazonSSMManagedInstanceCore
to grant
                                        use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
```

```
        return profile_arn
```

- Para obter detalhes da API, consulte a CreateInstanceProfile Referência da API AWS SDK for Python (Boto3).

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
```

```
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- Para obter detalhes da API, consulte a CreatePolicy Referência da API AWS SDK for Python (Boto3).

`CreatePolicyVersion`

O código de exemplo a seguir mostra como usar `CreatePolicyVersion`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
```

```
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version
```

- Para obter detalhes da API, consulte a CreatePolicyVersion Referência da API AWS SDK for Python (Boto3).

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_role(role_name, allowed_services):
```

```
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

- Para obter detalhes da API, consulte a CreateRole Referência da API AWS SDK for Python (Boto3).

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description
        )
        role = iam.Role(response["Role"]["RoleName"])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
 service_name)
        raise
    else:
        return role
```

- Para obter detalhes da API, consulte a CreateServiceLinkedRole Referência da API AWS SDK for Python (Boto3).

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- Para obter detalhes da API, consulte a CreateUser Referência da API AWS SDK for Python (Boto3).

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteAccessKey Referência da API AWS SDK for Python (Boto3).

`DeleteAccountAlias`

O código de exemplo a seguir mostra como usar `DeleteAccountAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
```

```
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise
```

- Para obter detalhes da API, consulte a DeleteAccountAlias Referência da API AWS SDK for Python (Boto3).

`DeleteInstanceProfile`

O código de exemplo a seguir mostra como usar `DeleteInstanceProfile`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo remove o perfil do perfil de instância, separa todas as políticas anexadas ao perfil e exclui todos os recursos.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
```

```
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
```

```
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
```

- Para obter detalhes da API, consulte a DeleteInstanceProfile Referência da API AWS SDK for Python (Boto3).

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

- Para obter detalhes da API, consulte a DeletePolicy Referência da API AWS SDK for Python (Boto3).

`DeleteRole`

O código de exemplo a seguir mostra como usar `DeleteRole`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_role(role_name):
```

```
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteRole Referência da API AWS SDK for Python (Boto3).

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
```

```
        logger.exception("Couldn't delete user %s.", user_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteUser Referência da API AWS SDK for Python (Boto3).

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Desanexar uma política de uma função usando o objeto Policy do Boto3.

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

Desanexar uma política de uma função usando o objeto Role do Boto3.

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

- Para obter detalhes da API, consulte a DetachRolePolicy Referência da API AWS SDK for Python (Boto3).

`DetachUserPolicy`

O código de exemplo a seguir mostra como usar `DetachUserPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
```

```
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

- Para obter detalhes da API, consulte a DetachUserPolicy Referência da API AWS SDK for Python (Boto3).

`GenerateCredentialReport`

O código de exemplo a seguir mostra como usar `GenerateCredentialReport`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
```

```
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your account.")
        raise
    else:
        return response
```

- Para obter detalhes da API, consulte a GenerateCredentialReport Referência da API AWS SDK for Python (Boto3).

`GetAccessKeyLastUsed`

O código de exemplo a seguir mostra como usar `GetAccessKeyLastUsed`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
```

```
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- Para obter detalhes da API, consulte a GetAccessKeyLastUsed Referência da API AWS SDK for Python (Boto3).

`GetAccountAuthorizationDetails`

O código de exemplo a seguir mostra como usar `GetAccountAuthorizationDetails`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
```

```
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- Para obter detalhes da API, consulte a GetAccountAuthorizationDetails Referência da API AWS SDK for Python (Boto3).

`GetAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(
            f"\tallow_users_to_change_password:
 {pw_policy.allow_users_to_change_password}"
        )
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
        print(f"\tpassword_reuse_prevention: {pw_policy.password_reuse_prevention}")
        print(
```

```
            f"\trequire_lowercase_characters:
{pw_policy.require_lowercase_characters}"
        )
        print(f"\trequire_numbers: {pw_policy.require_numbers}")
        print(f"\trequire_symbols: {pw_policy.require_symbols}")
        print(
            f"\trequire_uppercase_characters:
{pw_policy.require_uppercase_characters}"
        )
        printed = True
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchEntity":
            print("The account does not have a password policy set.")
        else:
            logger.exception("Couldn't get account password policy.")
            raise
    else:
        return printed
```

- Para obter detalhes da API, consulte a GetAccountPasswordPolicy Referência da API AWS SDK for Python (Boto3).

`GetAccountSummary`

O código de exemplo a seguir mostra como usar `GetAccountSummary`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_summary():
    """
    Gets a summary of account usage.
```

```
    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- Para obter detalhes da API, consulte a GetAccountSummary Referência da API AWS SDK for Python (Boto3).

`GetCredentialReport`

O código de exemplo a seguir mostra como usar `GetCredentialReport`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
```

```
    else:
        return response["Content"]
```

- Para obter detalhes da API, consulte a GetCredentialReport Referência da API AWS SDK for Python (Boto3).

GetPolicy

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- Para obter detalhes da API, consulte a GetPolicy Referência da API AWS SDK for Python (Boto3).

GetPolicyVersion

O código de exemplo a seguir mostra como usar `GetPolicyVersion`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- Para obter detalhes da API, consulte a GetPolicyVersion Referência da API AWS SDK for Python (Boto3).

`GetRole`

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load()  # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- Para obter detalhes da API, consulte a GetRole Referência da API AWS SDK for Python (Boto3).

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- Para obter detalhes da API, consulte a ListAccessKeys Referência da API AWS SDK for Python (Boto3).

`ListAccountAliases`

O código de exemplo a seguir mostra como usar `ListAccountAliases`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

- Para obter detalhes da API, consulte a ListAccountAliases Referência da API AWS SDK for Python (Boto3).

`ListAttachedRolePolicies`

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- Para obter detalhes da API, consulte a ListAttachedRolePolicies Referência da API AWS SDK for Python (Boto3).

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
    """
    try:
        for group in iam.groups.limit(count):
            logger.info("Group: %s", group.name)
    except ClientError:
        logger.exception("Couldn't list groups for the account.")
        raise
```

- Para obter detalhes da API, consulte a ListGroups Referência da API AWS SDK for Python (Boto3).

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
```

```
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- Para obter detalhes da API, consulte a ListPolicies Referência da API AWS SDK for Python (Boto3).

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- Para obter detalhes da API, consulte a ListRolePolicies Referência da API AWS SDK for Python (Boto3).

ListRoles

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- Para obter detalhes da API, consulte a ListRoles Referência da API AWS SDK for Python (Boto3).

Listar provedores de SAML

O código de exemplo a seguir mostra como usar `ListSAMLProviders`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info("Got SAML provider %s.", provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK for Python (Boto3).

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- Para obter detalhes da API, consulte a ListUsers Referência da API AWS SDK for Python (Boto3).

`UpdateAccessKey`

O código de exemplo a seguir mostra como usar `UpdateAccessKey`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate", key_id
        )
        raise
```

- Para obter detalhes da API, consulte a UpdateAccessKey Referência da API AWS SDK for Python (Boto3).

`UpdateUser`

O código de exemplo a seguir mostra como usar `UpdateUser`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

- Para obter detalhes da API, consulte a UpdateUser Referência da API AWS SDK for Python (Boto3).

Cenários

Criar e gerenciar um serviço resiliente

O exemplo de código a seguir mostra como criar um serviço web com balanceamento de carga que retorna recomendações de livros, filmes e músicas. O exemplo mostra como o serviço responde a falhas e como é possível reestruturá-lo para gerar mais resiliência em caso de falhas.

- Use um grupo do Amazon EC2 Auto Scaling para criar instâncias do Amazon Elastic Compute Cloud (Amazon EC2) com base em um modelo de execução e para manter o número de instâncias em um intervalo especificado.
- Gerencie e distribua solicitações HTTP com o Elastic Load Balancing.
- Monitore a integridade das instâncias em um grupo do Auto Scaling e encaminhe solicitações somente para instâncias íntegras.
- Execute um servidor Web Python em cada instância do EC2 para lidar com solicitações HTTP. O servidor Web responde com recomendações e verificações de integridade.

- Simule um serviço de recomendação com uma tabela do Amazon DynamoDB.
- Controle a resposta do servidor web às solicitações e verificações de integridade atualizando AWS Systems Manager os parâmetros.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute o cenário interativo em um prompt de comando.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example, such
 as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
```

```
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()
```

```
        logging.info("Creating Elastic Load Balancing target group and load
 balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port is
 open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access from
 this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
```

```
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging from
your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer endpoint.
Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as sending
requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer endpoint.")
```

```
                endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check to
update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
```

```
            "Starting demonstration of the service's resilience under various
 failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a non-
existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
 self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
 instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
```

```
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
```

```
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up manually
 to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM policies
 and\n"
        "instance scripts.",
    )
    args = parser.parse_args()
```

```
    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
```

```
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Crie uma classe que envolva ações do Auto Scaling e do Amazon EC2.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
```

```
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
```

```
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
```

```
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to attach
 to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
 created by
        this class. An instance's associated profile defines a role that is assumed
 by the
        instance. The role has attached policies that specify the AWS permissions
 granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
 definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
```

```
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
```

```
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.
```

```
        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
```

```
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
```

```
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
```

```
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
```

```
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")
```

```
    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with availability
zones {zones}."
```

```
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
```

```
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) -> None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")
```

```
    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")
```

```
    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
 delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
```

```
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress from
this
```

```
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to either
this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
```

```
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist. Please
check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
```

```
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}' on
port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])
```

```
            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Crie uma classe que envolva ações do Elastic Load Balancing.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
 how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
 and
```

```
        lower thresholds. In production, you might want to decrease the sensitivity
of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
```

```
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
```

```
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")
```

```
            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
the
        specified target group.
```

```
        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener to
the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
```

```
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

```
    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
```

```
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint, retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
correct region, "
                    "and ensure it has not been deleted or created in a different
account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
```

```
            return health_response["TargetHealthDescriptions"]
```

Crie uma classe que use o DynamoDB para simular um serviço de recomendação.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books, movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended, such
 as
        Book or Movie, and a range key named 'ItemId' that, combined with the
 MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
```

```
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5, "WriteCapacityUnits":
5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name, data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}: {err}"
            )

    def destroy(self) -> None:
        """
```

```
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Crie uma classe que envolva as ações do Systems Manager.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters to
 drive
    the demonstration of resilient architecture, such as failure of a dependency or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
```

```
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False. "
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Criar usuários somente leitura e leitura/gravação usando

Os exemplos de código a seguir mostram como criar usuários e anexar políticas a eles.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Criar dois usuários do IAM.
- Anexe uma política para um usuário obter e colocar objetos em um bucket do Amazon S3.
- Anexar uma política para o segundo usuário para obter objetos do bucket.
- Obter outras permissões para o bucket com base nas credenciais do usuário.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações do usuário do IAM.

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_user(user_name):
    """
```

```
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user



def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user



def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
```

```
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users



def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise



def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise



def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.
```

```
    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

Crie a funções que envolvam ações de política do IAM.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
```

```
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

Crie funções que envolvam ações de chave de acesso do IAM.

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Use as funções de wrapper para criar usuários com políticas diferentes e use as credenciais deles para acessar um bucket do Amazon S3.

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
    The demo then shows how the users can perform only the actions they are
 permitted
    to perform.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print("-" * 88)
    print(
        "Users can have policies and roles attached to grant them specific "
        "permissions."
    )
    s3 = boto3.resource("s3")
    bucket = s3.create_bucket(
        Bucket=f"demo-iam-bucket-{time.time_ns()}",
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user("demo-iam-read-writer")
    user_reader = create_user("demo-iam-reader")
    print(f"Created two IAM users: {user_read_writer.name} and {user_reader.name}")
    update_user(user_read_writer.name, "demo-iam-creator")
    update_user(user_reader.name, "demo-iam-getter")
    users = list_users()
    user_read_writer = next(
        user for user in users if user.user_id == user_read_writer.user_id
    )
    user_reader = next(user for user in users if user.user_id ==
 user_reader.user_id)
    print(
```

```
        f"Changed the names of the users to {user_read_writer.name} "
        f"and {user_reader.name}."
    )

    read_write_policy = policy_wrapper.create_policy(
        "demo-iam-read-write-policy",
        "Grants rights to create and get an object in the demo bucket.",
        ["s3:PutObject", "s3:GetObject"],
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(
        f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}"
    )
    print(read_write_policy.description)
    read_policy = policy_wrapper.create_policy(
        "demo-iam-read-policy",
        "Grants rights to get an object from the demo bucket.",
        "s3:GetObject",
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(f"Created policy {read_policy.policy_name} with ARN: {read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to {user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}.")
    user_reader_key = access_key_wrapper.create_key(user_reader.name)
    print(f"Created access key pair for {user_reader.name}.")

    s3_read_writer_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_read_writer_key.id,
        aws_secret_access_key=user_read_writer_key.secret,
    )
    demo_object_key = f"object-{time.time_ns()}"
    demo_object = None
    while demo_object is None:
        try:
            demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
                Key=demo_object_key, Body=b"AWS IAM demo object content!"
```

```
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise
    print(
        f"Put {demo_object_key} into {bucket.name} using "
        f"{user_read_writer.name}'s credentials."
    )

    read_writer_object = s3_read_writer_resource.Bucket(bucket.name).Object(
        demo_object_key
    )
    read_writer_content = read_writer_object.get()["Body"].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret,
    )
    demo_content = None
    while demo_content is None:
        try:
            demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
            demo_content = demo_object.get()["Body"].read()
            print(f"Got object {demo_object.key} using reader user's credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
```

```
            print("-" * 88)
            print(
                "Tried to delete the object using the reader user's credentials. "
                "Got expected AccessDenied error because the reader is not "
                "allowed to delete objects."
            )
            print("-" * 88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
    policy_wrapper.delete_policy(read_policy.arn)
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(
        f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}."
    )

    bucket.objects.delete()
    bucket.delete()
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachUserPolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy

- DeleteUser
- DetachUserPolicy
- ListUsers
- UpdateUser

Gerenciar chaves de acesso

O exemplo de código a seguir mostra como gerenciar chaves de acesso.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Criar e listar chaves de acesso.
- Descobrir quando e como a chave de acesso foi usada pela última vez.
- Atualizar e excluir chaves de acesso.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações de chave de acesso do IAM.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")
```

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys



def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.
```

```
    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response



def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate", key_id
```

```
        )
        raise



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Use a funções de wrapper para executar ações de chave de acesso para o usuário atual.

```
def usage_demo():
    """Shows how to create and manage access keys."""

    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep="\n")

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print("-" * 88)
    current_user_name = iam.CurrentUser().user_name
    print(
        f"This demo creates an access key for the current user "
        f"({current_user_name}), manipulates the key in a few ways, and then "
```

```
        f"deletes it."
    )
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
        print(
            "The current user already has the maximum of 2 access keys. To run "
            "this demo, either delete one of the access keys or use a user "
            "that has only 1 access key."
        )
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret
{new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)["AccessKeyLastUsed"]
        print(
            f"Key {all_keys[0].id} was last used to access {last_use['ServiceName']}
"
            f"on {last_use['LastUsedDate']}"
        )
        update_key(current_user_name, new_key.id, False)
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()
        delete_key(current_user_name, new_key.id)
        print_keys()
        print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAccessKey
  - DeleteAccessKey
  - GetAccessKeyLastUsed
  - ListAccessKeys
  - UpdateAccessKey

Políticas gerenciadas

O exemplo de código a seguir mostra como:

- Criar e listar políticas.
- Criar e obter versões de políticas.
- Reverter uma política para uma versão anterior.
- Excluir políticas.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie a funções que envolvam ações de política do IAM.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                         applies to. This ARN can contain wildcards, such as
```

```
                        'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                        in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version



def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
```

```
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement



def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
```

```
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.", rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
 default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists, so
 "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

Use a funções de wrapper para criar políticas, atualizar versões e obter informações sobre elas.

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print("-" * 88)
    print(
        "Policies let you define sets of permissions that can be attached to "
        "other IAM resources, like users and roles."
    )
    bucket_arn = f"arn:aws:s3:::amzn-s3-demo-bucket"
    policy = create_policy(
        "demo-iam-policy",
        "Policy for IAM demonstration.",
        ["s3:ListObjects"],
        bucket_arn,
    )
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies("Local")
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=", ")
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ["s3:PutObject"], bucket_arn, True
    )
    print(
        f"Added policy version {policy_version.version_id} to policy "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(
        f"Rolled back to version {rollback_version.version_id} for "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
    print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreatePolicy
  - CreatePolicyVersion
  - DeletePolicy
  - DeletePolicyVersion
  - GetPolicyVersion
  - ListPolicies
  - ListPolicyVersions
  - SetDefaultPolicyVersion

Gerenciar funções

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM.
- Anexar e separar políticas para um perfil.
- Excluir um perfil.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar perfis que envolvam ações de perfil do IAM.

```
import json
import logging
import pprint
```

```
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role



def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
```

```
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise



def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise



def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

Use a funções de wrapper para criar uma função, depois, anexe e desanexe uma política.

```
def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print("-" * 88)
    print(
        "Roles let you define sets of permissions and can be assumed by "
        "other entities, like users and services."
    )
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        "demo-iam-role", ["lambda.amazonaws.com",
 "batchoperations.s3.amazonaws.com"]
    )
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    attach_policy(role.name, policy_arn)
    print(f"Attached policy {policy_arn} to {role.name}.")
    print(f"Policies attached to role {role.name} are:")
    list_attached_policies(role.name)
    detach_policy(role.name, policy_arn)
    print(f"Detached policy {policy_arn} from {role.name}.")
    delete_role(role.name)
    print(f"Deleted {role.name}.")
    print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AttachRolePolicy
  - CreateRole

- DeleteRole
- DetachRolePolicy

Gerenciar sua conta

O exemplo de código a seguir mostra como:

- Obter e atualizar o alias da conta.
- Gerar um relatório de usuários e credenciais.
- Obter um resumo da utilização da conta.
- Obter detalhes de todos os usuários, grupos, perfis e políticas em sua conta, incluindo as relações uns com os outros.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que envolvam ações de conta do IAM.

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
```

```
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]



def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise



def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
```

```
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise



def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your account.")
        raise
    else:
        return response



def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]
```

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map



def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

Chame funções de wrapper para alterar o alias da conta e obter relatórios sobre a conta.

```
def usage_demo():
```

```
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print("-" * 88)
    print(
        "Setting an account alias lets you use the alias in your sign-in URL "
        "instead of your account number."
    )
    old_aliases = list_aliases()
    if len(old_aliases) > 0:
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f"alias-{index}-{time.time_ns()}"
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print("-" * 88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
    report_state = None
    while report_state != "COMPLETE":
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response["State"]
        if report_state != old_report_state:
            print(report_state, sep="")
        else:
            print(".", sep="")
        sys.stdout.flush()
        time.sleep(1)
    print()
    cred_report = get_credential_report()
    col_count = 3
    print(f"Got credentials report. Showing only the first {col_count} columns.")
```

```
    cred_lines = [
        line.split(",")[:col_count] for line in
 cred_report.decode("utf-8").split("\n")
    ]
    col_width = max([len(item) for line in cred_lines for item in line]) + 2
    for line in cred_report.decode("utf-8").split("\n"):
        print(
            "".join(element.ljust(col_width) for element in line.split(",")
[:col_count])
        )

    print("-" * 88)
    print("Let's get an account summary.")
    summary = get_summary()
    print("Here's your summary:")
    pprint.pprint(summary)

    print("-" * 88)
    print("Let's get authorization details!")
    details = get_authorization_details([])
    see_details = input("These are pretty long, do you want to see them (y/n)? ")
    if see_details.lower() == "y":
        pprint.pprint(details)

    print("-" * 88)
    pw_policy_created = None
    see_pw_policy = input("Want to see the password policy for the account (y/n)? ")
    if see_pw_policy.lower() == "y":
        while True:
            if print_password_policy():
                break
            else:
                answer = input(
                    "Do you want to create a default password policy (y/n)? "
                )
                if answer.lower() == "y":
                    pw_policy_created = iam.create_account_password_policy()
                else:
                    break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == "y":
            pw_policy_created.delete()
            print("Password policy deleted.")
```

```
    print("The SAML providers for your account are:")
    list_saml_providers(10)

    print("-" * 88)
    print("Thanks for watching.")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAccountAlias
  - DeleteAccountAlias
  - GenerateCredentialReport
  - GetAccountAuthorizationDetails
  - GetAccountSummary
  - GetCredentialReport
  - ListAccountAliases

Reverter uma versão de política

O exemplo de código a seguir mostra como:

- Obter a lista de versões da política em ordem por data.
- Encontrar a versão da política padrão.
- Tornar a versão da política anterior a padrão.
- Excluir a versão padrão antiga.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.", rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
 default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists, so
 "
                "nothing to roll back to.",
```

```
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - DeletePolicyVersion
  - ListPolicyVersions
  - SetDefaultPolicyVersion

AWS IoT exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with AWS IoT.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def hello_iot():
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT client and list
    up to 10 things in your AWS IoT account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    try:
        iot_client = boto3.client("iot")
        response = iot_client.list_things(maxResults=10)
        things = response.get("things", [])

        print("Hello, AWS IoT! Here are your things:")
        if things:
            for i, thing in enumerate(things, 1):
                print(f"{i}. {thing['thingName']}")
        else:
            print("No things found in your AWS IoT account.")
    except ClientError as e:
        if e.response["Error"]["Code"] == "UnauthorizedException":
            print("You don't have permission to access AWS IoT.")
        else:
            print(f"Couldn't access AWS IoT. Error: {e}")
    except NoCredentialsError:
        print("No AWS credentials found. Please configure your credentials.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

- Para obter detalhes da API, consulte listThings na Referência da API AWS SDK para Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie uma AWS IoT coisa.
- Gerar um certificado de dispositivo.
- Atualize AWS IoT qualquer coisa com atributos.
- Exibir um endpoint exclusivo.
- Liste seus AWS IoT certificados.
- Atualize uma AWS IoT sombra.
- Gravar informações do estado.
- Cria uma regra.
- Listar suas regras.
- Pesquisar coisas usando o nome da coisa.
- Exclua qualquer AWS IoT coisa.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe de invólucro de IoT para gerenciar as operações.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
```

```
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)


    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def list_things(self):
        """
        Lists AWS IoT things.
```

```
        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response = self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

```
    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal, thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response = self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
            raise
        else:
            return response["endpointAddress"]


    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
```

```
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn": role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_topic_rules(self):
        """
        Lists AWS IoT topic rules.

        :return: The list of topic rules.
        """
        try:
            rules = []
            paginator = self.iot_client.get_paginator("list_topic_rules")
            for page in paginator.paginate():
                rules.extend(page["rules"])
            logger.info("Retrieved %s topic rules.", len(rules))
            return rules
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list topic rules. Here's why: %s: %s",
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
            raise



    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])


    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise


    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_thing_shadow(self, thing_name, shadow_state):
```

```
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
            response = self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return shadow
```

Execute um cenário interativo demonstrando os fundamentos da IoT.

```
class IoTScenario:
    """Runs an interactive scenario that shows how to use AWS IoT."""

    is_interactive = True

    def __init__(self, iot_wrapper, iot_data_client, cfn_client,
 stack_name="IoTBasicsStack", template_path=None):
        """
        :param iot_wrapper: An instance of the IoTWrapper class.
        :param iot_data_client: A Boto3 IoT Data Plane client.
        :param cfn_client: A Boto3 CloudFormation client.
        :param stack_name: Name for the CloudFormation stack.
        :param template_path: Path to the CloudFormation template file.
        """
        self.iot_wrapper = iot_wrapper
        self.iot_data_client = iot_data_client
        self.cfn_client = cfn_client
        self.thing_name = None
        self.certificate_arn = None
        self.certificate_id = None
        self.rule_name = None
        self.stack_name = stack_name
        self.template_path = template_path or "../../../scenarios/basics/iot/
iot_usecase/resources/cfn_template.yaml"

    def _deploy_stack(self):
        """Deploy CloudFormation stack and return outputs."""
        with open(self.template_path, "r") as f:
            template_body = f.read()

        try:
            self.cfn_client.create_stack(
                StackName=self.stack_name,
                TemplateBody=template_body,
```

```
                Capabilities=["CAPABILITY_NAMED_IAM"]
            )

            waiter = self.cfn_client.get_waiter("stack_create_complete")
            waiter.wait(StackName=self.stack_name)

            response = self.cfn_client.describe_stacks(StackName=self.stack_name)
            outputs = {output["OutputKey"]: output["OutputValue"]
                      for output in response["Stacks"][0]["Outputs"]}
            return outputs["SNSTopicArn"], outputs["RoleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                response =
 self.cfn_client.describe_stacks(StackName=self.stack_name)
                outputs = {output["OutputKey"]: output["OutputValue"]
                          for output in response["Stacks"][0]["Outputs"]}
                return outputs["SNSTopicArn"], outputs["RoleArn"]
            raise

    def _cleanup_stack(self):
        """Delete CloudFormation stack."""
        try:
            self.cfn_client.delete_stack(StackName=self.stack_name)
            waiter = self.cfn_client.get_waiter("stack_delete_complete")
            waiter.wait(StackName=self.stack_name)
            print("CloudFormation stack deleted successfully.")
        except ClientError as err:
            logger.error(f"Failed to delete stack: {err}")

    def run_scenario(self, thing_name, rule_name):
        """
        Runs the IoT basics scenario.

        :param thing_name: The name of the thing to create.
        :param rule_name: The name of the topic rule to create.
        """
        print("-" * 88)
        print("Welcome to the AWS IoT basics scenario!")
        print("-" * 88)
        print(
            "This scenario demonstrates how to interact with AWS IoT using the AWS
 SDK for Python (Boto3).\n"
            "AWS IoT provides secure, bi-directional communication between Internet-
connected devices\n"
```

```
            "and the AWS cloud. You can manage device connections, process device
data, and build IoT applications.\n"
        )

        self.thing_name = thing_name
        self.rule_name = rule_name

        try:
            print("\nDeploying CloudFormation stack...")
            sns_topic_arn, role_arn = self._deploy_stack()
            print(f"Stack deployed. SNS Topic: {sns_topic_arn}")

            input("\nNext, we'll create an AWS IoT thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("1. Create an AWS IoT thing")
            print("-" * 88)
            response = self.iot_wrapper.create_thing(thing_name)
            print(f"Created thing: {response['thingName']}")
            print(f"Thing ARN: {response['thingArn']}")

            input("\nNext, we'll list things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("2. List things")
            print("-" * 88)
            things = self.iot_wrapper.list_things()
            print(f"Found {len(things)} thing(s) in your account")
            for thing in things[:5]:  # Show first 5
                print(f"  Thing name: {thing['thingName']}")

            input("\nNext, we'll generate a device certificate. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("3. Generate a device certificate")
            print("-" * 88)
            cert_response = self.iot_wrapper.create_keys_and_certificate()
            self.certificate_arn = cert_response["certificateArn"]
            self.certificate_id = cert_response["certificateId"]
            print(f"Created certificate: {self.certificate_id}")

            input("\nNext, we'll attach the certificate to the thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
```

```
            print("4. Attach the certificate to the thing")
            print("-" * 88)
            self.iot_wrapper.attach_thing_principal(thing_name,
self.certificate_arn)
            print(f"Attached certificate to thing: {thing_name}")

            input("\nNext, we'll update the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("5. Update the thing shadow")
            print("-" * 88)
            shadow_state = {"state": {"reported": {"temperature": 25, "humidity":
50}}}
            self.iot_wrapper.update_thing_shadow(thing_name, shadow_state)
            print(f"Updated shadow for thing: {thing_name}")

            input("\nNext, we'll get the thing shadow. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("6. Get the thing shadow")
            print("-" * 88)
            shadow = self.iot_wrapper.get_thing_shadow(thing_name)
            print(f"Shadow state: {json.dumps(shadow['state'], indent=2)}")

            input("\nNext, we'll get the AWS IoT endpoint. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("7. Get the AWS IoT endpoint")
            print("-" * 88)
            endpoint = self.iot_wrapper.describe_endpoint()
            print(f"IoT endpoint: {endpoint}")

            input("\nNext, we'll list certificates. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("8. List certificates")
            print("-" * 88)
            certificates = self.iot_wrapper.list_certificates()
            print(f"Found {len(certificates)} certificate(s)")
            for cert in certificates:
                print(f"  Certificate ID: {cert['certificateId']}")
                print(f"  Certificate ARN: {cert['certificateArn']}")
                print(f"  Status: {cert['status']}")
```

```
            input("\nNext, we'll create a topic rule. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("9. Create a topic rule")
            print("-" * 88)
            self.iot_wrapper.create_topic_rule(
                rule_name, f"device/{thing_name}/data", sns_topic_arn, role_arn
            )
            print(f"Created topic rule: {rule_name}")

            input("\nNext, we'll list topic rules. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("10. List topic rules")
            print("-" * 88)
            rules = self.iot_wrapper.list_topic_rules()
            print(f"Found {len(rules)} topic rule(s)")
            for rule in rules:
                print(f"  Rule name: {rule['ruleName']}")
                print(f"  Rule ARN: {rule['ruleArn']}")

            input("\nNext, we'll configure thing indexing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("11. Configure thing indexing")
            print("-" * 88)
            self.iot_wrapper.update_indexing_configuration()
            print("Enabled thing indexing")
            print("Waiting for indexing to be ready...")
            time.sleep(10)

            input("\nNext, we'll search for things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("12. Search for things")
            print("-" * 88)
            try:
                things = self.iot_wrapper.search_index(f"thingName:{thing_name}")
                if things:
                    print(f"Found {len(things)} thing(s) matching the query")
                    for thing in things:
                        print(f"  Thing name: {thing.get('thingName', 'N/A')}")
                        print(f"  Thing ID: {thing.get('thingId', 'N/A')}")
                else:
```

```
                    print("No things found. Indexing may take a few minutes.")
            except ClientError as err:
                if err.response["Error"]["Code"] in [
                    "IndexNotReadyException",
                    "InvalidRequestException",
                ]:
                    print("Search index not ready yet. This is expected.")
                else:
                    raise

        except ClientError as err:
            logger.error(
                "Scenario failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        finally:
            self._cleanup()

    def _cleanup(self):
        """Cleans up resources created during the scenario."""
        if not self.thing_name:
            return

        print("\n" + "-" * 88)
        print("Cleanup")
        print("-" * 88)

        if q.ask("Do you want to delete the resources? (y/n) ", q.is_yesno):
            try:
                if self.certificate_arn:
                    print(f"Detaching certificate from thing: {self.thing_name}")
                    self.iot_wrapper.detach_thing_principal(
                        self.thing_name, self.certificate_arn
                    )

                if self.certificate_id:
                    print(f"Deleting certificate: {self.certificate_id}")
                    self.iot_wrapper.delete_certificate(self.certificate_id)

                if self.thing_name:
                    print(f"Deleting thing: {self.thing_name}")
                    self.iot_wrapper.delete_thing(self.thing_name)
```

```
            if self.rule_name:
                print(f"Deleting topic rule: {self.rule_name}")
                self.iot_wrapper.delete_topic_rule(self.rule_name)

            self._cleanup_stack()
            print("Resources deleted successfully.")
        except ClientError as err:
            logger.error(
                "Cleanup failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
    else:
        print("Resources will remain in your account.")

    print("\n" + "-" * 88)
    print("Thanks for using AWS IoT!")
    print("-" * 88)
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex

- UpdateIndexingConfiguration
- UpdateThing

Ações

`AttachThingPrincipal`

O código de exemplo a seguir mostra como usar `AttachThingPrincipal`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
```

```
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal, thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a AttachThingPrincipal Referência da API AWS SDK for Python (Boto3).

`CreateKeysAndCertificate`

O código de exemplo a seguir mostra como usar `CreateKeysAndCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
```

```
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response = self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateKeysAndCertificate Referência da API AWS SDK for Python (Boto3).

`CreateThing`

O código de exemplo a seguir mostra como usar `CreateThing`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
 thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
```

```
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateThing Referência da API AWS SDK for Python (Boto3).

`CreateTopicRule`

O código de exemplo a seguir mostra como usar `CreateTopicRule`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)
```

```
    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn": role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateTopicRule Referência da API AWS SDK for Python (Boto3).

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
```

```
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteCertificate Referência da API AWS SDK for Python (Boto3).

`DeleteThing`

O código de exemplo a seguir mostra como usar `DeleteThing`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_thing(self, thing_name):
```

```
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteThing Referência da API AWS SDK for Python (Boto3).

`DeleteTopicRule`

O código de exemplo a seguir mostra como usar `DeleteTopicRule`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
```

```
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTopicRule Referência da API AWS SDK for Python (Boto3).

`DescribeEndpoint`

O código de exemplo a seguir mostra como usar `DescribeEndpoint`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response = self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]
```

- Para obter detalhes da API, consulte a DescribeEndpoint Referência da API AWS SDK for Python (Boto3).

`DetachThingPrincipal`

O código de exemplo a seguir mostra como usar `DetachThingPrincipal`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def detach_thing_principal(self, thing_name, principal):
```

```
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DetachThingPrincipal Referência da API AWS SDK for Python (Boto3).

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
```

```
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListCertificates Referência da API AWS SDK for Python (Boto3).

`ListThings`

O código de exemplo a seguir mostra como usar `ListThings`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
```

```
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListThings Referência da API AWS SDK for Python (Boto3).

`SearchIndex`

O código de exemplo a seguir mostra como usar `SearchIndex`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
```

```
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])
```

- Para obter detalhes da API, consulte a SearchIndex Referência da API AWS SDK for Python (Boto3).

`UpdateIndexingConfiguration`

O código de exemplo a seguir mostra como usar `UpdateIndexingConfiguration`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateIndexingConfiguration Referência da API AWS SDK for Python (Boto3).

AWS IoT data exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with AWS IoT data.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetThingShadow`

O código de exemplo a seguir mostra como usar `GetThingShadow`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
```

```
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
            response = self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

- Para obter detalhes da API, consulte a GetThingShadow Referência da API AWS SDK for Python (Boto3).

`UpdateThingShadow`

O código de exemplo a seguir mostra como usar `UpdateThingShadow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateThingShadow Referência da API AWS SDK for Python (Boto3).

AWS IoT SiteWise exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with AWS IoT SiteWise.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT SiteWise

O exemplo de código a seguir mostra como começar a usar o AWS IoT SiteWise.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_iot_sitewise(iot_sitewise_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT SiteWise
    client and list the asset models in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param iot_sitewise_client: A Boto3 AWS IoT SiteWise Client object. This object
 wraps
                                the low-level AWS IoT SiteWise service API.
    """
    print("Hello, AWS IoT SiteWise! Let's list some of your asset models:\n")
    paginator = iot_sitewise_client.get_paginator("list_asset_models")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

    asset_model_names: [str] = []
    for page in page_iterator:
        for asset_model in page["assetModelSummaries"]:
            asset_model_names.append(asset_model["name"])

    print(f"{len(asset_model_names)} asset model(s) retrieved.")
    for asset_model_name in asset_model_names:
        print(f"\t{asset_model_name}")


if __name__ == "__main__":
    hello_iot_sitewise(boto3.client("iotsitewise"))
```

- Para obter detalhes da API, consulte a ListAssetModels Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um modelo AWS IoT SiteWise de ativos.
- Crie um AWS IoT SiteWise ativo.
- Recuperar os valores de ID da propriedade.
- Envie dados para um AWS IoT SiteWise ativo.
- Recupere o valor da propriedade AWS IoT SiteWise Asset.
- Crie um AWS IoT SiteWise portal.
- Crie um AWS IoT SiteWise gateway.
- Descreva o AWS IoT SiteWise Gateway.
- Exclua os AWS IoT SiteWise ativos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class IoTSitewiseGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to manage IoT physical assets
 using
    the AWS IoT SiteWise.
    """

    def __init__(
        self,
        iot_sitewise_wrapper: IoTSitewiseWrapper,
        cloud_formation_resource: ServiceResource,
    ):
```

```
        self.iot_sitewise_wrapper = iot_sitewise_wrapper
        self.cloud_formation_resource = cloud_formation_resource
        self.stack = None
        self.asset_model_id = None
        self.asset_id = None
        self.portal_id = None
        self.gateway_id = None

    def run(self) -> None:
        """
        Runs the scenario.
        """
        print(
            """
AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes.
It is designed to help industrial and manufacturing organizations collect data from
 their equipment and
processes, and use that data to make informed decisions about their operations.

One of the key features of AWS IoT SiteWise is its ability to connect to a wide
 range of industrial
equipment and systems, including programmable logic controllers (PLCs), sensors, and
 other
industrial devices. It can collect data from these devices and organize it into a
 unified data model,
making it easier to analyze and gain insights from the data. AWS IoT SiteWise also
 provides tools for
visualizing the data, setting up alarms and alerts, and generating reports.

Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data.
It can collect and store data from thousands of devices and process millions of data
 points per second,
making it suitable for large-scale industrial operations. Additionally, AWS IoT
 SiteWise is designed
to be secure and compliant, with features like role-based access controls, data
 encryption,
and integration with other AWS services for additional security and compliance
 features.

Let's get started...
        """
```

```
        )
        press_enter_to_continue()
        print_dashes()
        print(f"")
        print(
            f"Use AWS CloudFormation to create an IAM role that is required for this
 scenario."
        )
        template_file = IoTSitewiseGettingStarted.get_template_as_string()

        self.stack = self.deploy_cloudformation_stack(
            "python-iot-sitewise-basics", template_file
        )
        outputs = self.stack.outputs
        iam_role = None

        for output in outputs:
            if output.get("OutputKey") == "SitewiseRoleArn":
                iam_role = output.get("OutputValue")

        if iam_role is None:
            error_string = f"Failed to retrieve iam_role from CloudFormation stack."
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"The ARN of the IAM role is {iam_role}")
        print_dashes()
        print_dashes()
        print(f"1. Create an AWS SiteWise Asset Model")
        print(
            """
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such as
 equipment,
processes, and systems, that exist in an industrial environment. This model provides
 a structured and
hierarchical representation of these assets, allowing users to define the
 relationships and values
of each asset.

This scenario creates two asset model values: temperature and humidity.
        """
        )
        press_enter_to_continue()
        asset_model_name = "MyAssetModel1"
```

```
        temperature_property_name = "temperature"
        humidity_property_name = "humidity"
        try:
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
            ]
            self.asset_model_id = self.iot_sitewise_wrapper.create_asset_model(
                asset_model_name, properties
            )
            print(
                f"Asset Model successfully created. Asset Model ID:
 {self.asset_model_id}. "
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                self.asset_model_id =
 self.get_model_id_for_model_name(asset_model_name)
                print(
                    f"Asset Model {asset_model_name} already exists. Asset Model ID:
 {self.asset_model_id}. "
                )
            else:
                raise

        press_enter_to_continue()
        print_dashes()
        print(f"2. Create an AWS IoT SiteWise Asset")
        print(
            """
The IoT SiteWise model that we just created defines the structure and metadata for
 your physical assets.
```

```
Now we create an asset from the asset model.

        """
        )
        press_enter_to_continue()

        self.asset_id = self.iot_sitewise_wrapper.create_asset(
            "MyAsset1", self.asset_model_id
        )

        print(f"Asset created with ID: {self.asset_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()
        print(f"3. Retrieve the property ID values")
        print(
            """
To send data to an asset, we need to get the property ID values. In this scenario,
 we access the
temperature and humidity property ID values.
        """
        )
        press_enter_to_continue()
        property_ids = self.iot_sitewise_wrapper.list_asset_model_properties(
            self.asset_model_id
        )
        humidity_property_id = None
        temperature_property_id = None
        for property_id in property_ids:
            if property_id.get("name") == humidity_property_name:
                humidity_property_id = property_id.get("id")
            elif property_id.get("name") == temperature_property_name:
                temperature_property_id = property_id.get("id")
        if humidity_property_id is None or temperature_property_id is None:
            error_string = f"Failed to retrieve property IDs from Asset Model."
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"The Humidity property Id is {humidity_property_id}")
        print(f"The Temperature property Id is {temperature_property_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()
```

```
        print(f"4. Send data to an AWS IoT SiteWise Asset")
        print(
            """
By sending data to an IoT SiteWise Asset, you can aggregate data from
multiple sources, normalize the data into a standard format, and store it in a
centralized location. This makes it easier to analyze and gain insights from the
 data.

In this example, we generate sample temperature and humidity data and send it to the
 AWS IoT SiteWise asset.

        """
        )
        press_enter_to_continue()

        values = [
            {
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
        self.iot_sitewise_wrapper.batch_put_asset_property_value(self.asset_id,
 values)
        print(f"Data sent successfully.")

        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"5. Retrieve the value of the IoT SiteWise Asset property")
        print(
            """
IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data
from connected equipment and sensors. One of the key benefits of reading an IoT
 SiteWise property
is the ability to gain valuable insights from your industrial data.
```

```
        """
        )
        press_enter_to_continue()

        property_value = self.iot_sitewise_wrapper.get_asset_property_value(
            self.asset_id, temperature_property_id
        )
        print(f"The property name is '{temperature_property_name}'.")

        print(
            f"The value of this property is: {property_value['value']
['doubleValue']}"
        )
        press_enter_to_continue()

        property_value = self.iot_sitewise_wrapper.get_asset_property_value(
            self.asset_id, humidity_property_id
        )
        print(f"The property name is '{humidity_property_name}'.")
        print(
            f"The value of this property is: {property_value['value']
['doubleValue']}"
        )
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"6. Create an IoT SiteWise Portal")
        print(
            """
An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources,
such as sensors, equipment, and control systems, into a centralized platform.
        """
        )

        press_enter_to_continue()
        contact_email = q.ask("Enter a contact email for the portal:", q.non_empty)
        print("Creating the portal. The portal may take a while to become active.")
        self.portal_id = self.iot_sitewise_wrapper.create_portal(
            "MyPortal1", iam_role, contact_email
        )
        print(f"Portal created successfully. Portal ID {self.portal_id}")
        press_enter_to_continue()
```

```
        print_dashes()
        print_dashes()

        print(f"7. Describe the Portal")
        print(
            """
In this step, we get a description of the portal and display the portal URL.
        """
        )
        press_enter_to_continue()
        portal_description =
 self.iot_sitewise_wrapper.describe_portal(self.portal_id)
        print(f"Portal URL: {portal_description['portalStartUrl']}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"8. Create an IoT SiteWise Gateway")
        press_enter_to_continue()
        self.gateway_id = self.iot_sitewise_wrapper.create_gateway(
            "MyGateway1", "MyThing1"
        )
        print(f"Gateway creation completed successfully. id is {self.gateway_id}")
        print_dashes()
        print_dashes()
        print(f"9. Describe the IoT SiteWise Gateway")
        press_enter_to_continue()

        gateway_description = self.iot_sitewise_wrapper.describe_gateway(
            self.gateway_id
        )
        print(f"Gateway Name: {gateway_description['gatewayName']}")
        print(f"Gateway ARN: {gateway_description['gatewayArn']}")
        print(f"Gateway Platform:\n{gateway_description['gatewayPlatform']}")
        print(f"Gateway Creation Date: {gateway_description['gatewayArn']}")
        print_dashes()
        print_dashes()

        print(f"10. Delete the AWS IoT SiteWise Assets")
        if q.ask("Would you like to delete the IoT SiteWise Assets? (y/n)",
 q.is_yesno):
            self.cleanup()
        else:
            print(f"The resources will not be deleted.")
```

```
        print_dashes()
        print_dashes()
        print(f"This concludes the AWS IoT SiteWise Scenario")

    def cleanup(self) -> None:
        """
        Deletes the CloudFormation stack and the resources created for the demo.
        """

        if self.gateway_id is not None:
            self.iot_sitewise_wrapper.delete_gateway(self.gateway_id)
            print(f"Deleted gateway with id {self.gateway_id}.")
            self.gateway_id = None
        if self.portal_id is not None:
            self.iot_sitewise_wrapper.delete_portal(self.portal_id)
            print(f"Deleted portal with id {self.portal_id}.")
            self.portal_id = None
        if self.asset_id is not None:
            self.iot_sitewise_wrapper.delete_asset(self.asset_id)
            print(f"Deleted asset with id {self.asset_id}.")
            self.iot_sitewise_wrapper.wait_asset_deleted(self.asset_id)
            self.asset_id = None
        if self.asset_model_id is not None:
            self.iot_sitewise_wrapper.delete_asset_model(self.asset_model_id)
            print(f"Deleted asset model with id {self.asset_model_id}.")
            self.asset_model_id = None
        if self.stack is not None:
            stack = self.stack
            self.stack = None
            self.destroy_cloudformation_stack(stack)

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `SitewiseRoles-template.yaml` AWS CloudFormation
script and are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
```

```
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example resources.
        """
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may take a
few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        template_file_path = os.path.join(script_dir, "SitewiseRoles-template.yaml")
        file = open(template_file_path, "r")
        return file.read()
```

```
    def get_model_id_for_model_name(self, model_name: str) -> str:
        """
        Returns the model ID for the given model name.

        :param model_name: The name of the model.
        :return: The model ID.
        """
        model_id = None
        asset_models = self.iot_sitewise_wrapper.list_asset_models()
        for asset_model in asset_models:
            if asset_model["name"] == model_name:
                model_id = asset_model["id"]
                break
        return model_id
```

IoTSitewiseWrapper classe que envolve AWS IoT SiteWise ações.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.
```

```
        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset_model(
        self, asset_model_name: str, properties: List[Dict[str, Any]]
    ) -> str:
        """
        Creates an AWS IoT SiteWise Asset Model.

        :param asset_model_name: The name of the asset model to create.
        :param properties: The property definitions of the asset model.
        :return: The ID of the created asset model.
        """
        try:
            response = self.iotsitewise_client.create_asset_model(
                assetModelName=asset_model_name,
                assetModelDescription="This is a sample asset model description.",
                assetModelProperties=properties,
            )
            asset_model_id = response["assetModelId"]
            waiter = self.iotsitewise_client.get_waiter("asset_model_active")
            waiter.wait(assetModelId=asset_model_id)
            return asset_model_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Asset model %s already exists.", asset_model_name)
            else:
                logger.error(
                    "Error creating asset model %s. Here's why %s",
                    asset_model_name,
                    err.response["Error"]["Message"],
                )
            raise


    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
```

```
        :param asset_model_id: The ID of the asset model to associate with the
asset.
        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
            raise


    def list_asset_models(self) -> List[Dict[str, Any]]:
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
model.

        """
        try:
            asset_models = []
            paginator = self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
```

```
            raise


    def list_asset_model_properties(self, asset_model_id: str) -> List[Dict[str,
 Any]]:
        """
        Lists all AWS IoT SiteWise Asset Model Properties.

        :param asset_model_id: The ID of the asset model to list values for.
        :return: A list of dictionaries containing information about each asset
 model property.
        """
        try:
            asset_model_properties = []
            paginator = self.iotsitewise_client.get_paginator(
                "list_asset_model_properties"
            )
            pages = paginator.paginate(assetModelId=asset_model_id)
            for page in pages:
                asset_model_properties.extend(page["assetModelPropertySummaries"])
            return asset_model_properties
        except ClientError as err:
            logger.error(
                "Error listing asset model values. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def batch_put_asset_property_value(
        self, asset_id: str, values: List[Dict[str, str]]
    ) -> None:
        """
        Sends data to an AWS IoT SiteWise Asset.

        :param asset_id: The asset ID.
        :param values: A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        """
        try:
            entries = self.properties_to_values(asset_id, values)
```

```
            self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Asset %s does not exist.", asset_id)
            else:
                logger.error(
                    "Error sending data to asset. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise


    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
```

```
                "propertyId": value["propertyId"],
                "propertyValues": [
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
                ],
            }
            entries.append(entry)
        return entries


    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
        """
        Gets the value of an AWS IoT SiteWise Asset Property.

        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Asset %s or property %s does not exist.", asset_id, property_id
                )
            else:
                logger.error(
                    "Error getting asset property value. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise


    def create_portal(
```

```
        self, portal_name: str, iam_role_arn: str, portal_contact_email: str
    ) -> str:
        """
        Creates an AWS IoT SiteWise Portal.

        :param portal_name: The name of the portal to create.
        :param iam_role_arn: The ARN of an IAM role.
        :param portal_contact_email: The contact email of the portal.
        :return: The ID of the created portal.
        """
        try:
            response = self.iotsitewise_client.create_portal(
                portalName=portal_name,
                roleArn=iam_role_arn,
                portalContactEmail=portal_contact_email,
            )
            portal_id = response["portalId"]
            waiter = self.iotsitewise_client.get_waiter("portal_active")
            waiter.wait(portalId=portal_id, WaiterConfig={"MaxAttempts": 40})
            return portal_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Portal %s already exists.", portal_name)
            else:
                logger.error(
                    "Error creating portal %s. Here's why %s",
                    portal_name,
                    err.response["Error"]["Message"],
                )
            raise


    def describe_portal(self, portal_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to describe.
        :return: A dictionary containing information about the portal.
        """
        try:
            response = self.iotsitewise_client.describe_portal(portalId=portal_id)
            return response
        except ClientError as err:
            logger.error(
```

```
                "Error describing portal %s. Here's why %s",
                portal_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_gateway(self, gateway_name: str, my_thing: str) -> str:
        """
        Creates an AWS IoT SiteWise Gateway.

        :param gateway_name: The name of the gateway to create.
        :param my_thing: The core device thing name.
        :return: The ID of the created gateway.
        """
        try:
            response = self.iotsitewise_client.create_gateway(
                gatewayName=gateway_name,
                gatewayPlatform={
                    "greengrassV2": {"coreDeviceThingName": my_thing},
                },
                tags={"Environment": "Production"},
            )
            gateway_id = response["gatewayId"]
            return gateway_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Gateway %s already exists.", gateway_name)
            else:
                logger.error(
                    "Error creating gateway %s. Here's why %s",
                    gateway_name,
                    err.response["Error"]["Message"],
                )
            raise


    def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to describe.
        :return: A dictionary containing information about the gateway.
        """
```

```
        try:
            response =
self.iotsitewise_client.describe_gateway(gateway_id=gateway_id)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error describing gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_gateway(self, gateway_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to delete.
        """
        try:
            self.iotsitewise_client.delete_gateway(gateway_id=gateway_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error deleting gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_portal(self, portal_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to delete.
        """
        try:
            self.iotsitewise_client.delete_portal(portalId=portal_id)
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Portal %s does not exist.", portal_id)
            else:
                logger.error(
                    "Error deleting portal %s. Here's why %s",
                    portal_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_asset(self, asset_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset.

        :param asset_id: The ID of the asset to delete.
        """
        try:
            self.iotsitewise_client.delete_asset(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset %s. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.

        :param asset_model_id: The ID of the asset model to delete.
        """
        try:
            self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise
```

```
    def wait_asset_deleted(self, asset_id: str) -> None:
        """
        Waits for an AWS IoT SiteWise Asset to be deleted.

        :param asset_id: The ID of the asset to wait for.
        """
        try:
            waiter = self.iotsitewise_client.get_waiter("asset_not_exists")
            waiter.wait(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error waiting for asset %s to be deleted. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties
  - ListAssetModels

Ações

`BatchPutAssetPropertyValue`

O código de exemplo a seguir mostra como usar `BatchPutAssetPropertyValue`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def batch_put_asset_property_value(
```

```
        self, asset_id: str, values: List[Dict[str, str]]
    ) -> None:
        """
        Sends data to an AWS IoT SiteWise Asset.

        :param asset_id: The asset ID.
        :param values: A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        """
        try:
            entries = self.properties_to_values(asset_id, values)
            self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Asset %s does not exist.", asset_id)
            else:
                logger.error(
                    "Error sending data to asset. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Uma função auxiliar para gerar o parâmetro de entradas com base em uma lista de valores.

```
    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
```

```
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
                "propertyId": value["propertyId"],
                "propertyValues": [
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
                ],
            }
            entries.append(entry)
        return entries
```

Aqui está um exemplo de uma lista de valores para passar para a função auxiliar.

```
        values = [
            {
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
```

```
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
```

- Para obter detalhes da API, consulte a BatchPutAssetPropertyValue Referência da API AWS SDK for Python (Boto3).

`CreateAsset`

O código de exemplo a seguir mostra como usar `CreateAsset`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
```

```
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
        :param asset_model_id: The ID of the asset model to associate with the
asset.
        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a CreateAsset Referência da API AWS SDK for Python (Boto3).

`CreateAssetModel`

O código de exemplo a seguir mostra como usar `CreateAssetModel`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset_model(
        self, asset_model_name: str, properties: List[Dict[str, Any]]
    ) -> str:
```

```
        """
        Creates an AWS IoT SiteWise Asset Model.

        :param asset_model_name: The name of the asset model to create.
        :param properties: The property definitions of the asset model.
        :return: The ID of the created asset model.
        """
        try:
            response = self.iotsitewise_client.create_asset_model(
                assetModelName=asset_model_name,
                assetModelDescription="This is a sample asset model description.",
                assetModelProperties=properties,
            )
            asset_model_id = response["assetModelId"]
            waiter = self.iotsitewise_client.get_waiter("asset_model_active")
            waiter.wait(assetModelId=asset_model_id)
            return asset_model_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Asset model %s already exists.", asset_model_name)
            else:
                logger.error(
                    "Error creating asset model %s. Here's why %s",
                    asset_model_name,
                    err.response["Error"]["Message"],
                )
            raise
```

Aqui está um exemplo de uma lista de propriedades para passar para a função auxiliar.

```
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
```

```
                    "type": {
                        "measurement": {},
                    },
                },
            ]
```

- Para obter detalhes da API, consulte a CreateAssetModel Referência da API AWS SDK for Python (Boto3).

`CreateGateway`

O código de exemplo a seguir mostra como usar `CreateGateway`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
```

```
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_gateway(self, gateway_name: str, my_thing: str) -> str:
        """
        Creates an AWS IoT SiteWise Gateway.

        :param gateway_name: The name of the gateway to create.
        :param my_thing: The core device thing name.
        :return: The ID of the created gateway.
        """
        try:
            response = self.iotsitewise_client.create_gateway(
                gatewayName=gateway_name,
                gatewayPlatform={
                    "greengrassV2": {"coreDeviceThingName": my_thing},
                },
                tags={"Environment": "Production"},
            )
            gateway_id = response["gatewayId"]
            return gateway_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Gateway %s already exists.", gateway_name)
            else:
                logger.error(
                    "Error creating gateway %s. Here's why %s",
                    gateway_name,
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a CreateGateway Referência da API AWS SDK for Python (Boto3).

`DeleteAsset`

O código de exemplo a seguir mostra como usar `DeleteAsset`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_asset(self, asset_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset.
```

```
        :param asset_id: The ID of the asset to delete.
        """
        try:
            self.iotsitewise_client.delete_asset(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset %s. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteAsset Referência da API AWS SDK for Python (Boto3).

`DeleteAssetModel`

O código de exemplo a seguir mostra como usar `DeleteAssetModel`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
```

```
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.

        :param asset_model_id: The ID of the asset model to delete.
        """
        try:
            self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteAssetModel Referência da API AWS SDK for Python (Boto3).

`DeleteGateway`

O código de exemplo a seguir mostra como usar `DeleteGateway`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_gateway(self, gateway_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to delete.
        """
```

```
        try:
            self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error deleting gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DeleteGateway Referência da API AWS SDK for Python (Boto3).

`DescribeGateway`

O código de exemplo a seguir mostra como usar `DescribeGateway`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
```

```
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to describe.
        :return: A dictionary containing information about the gateway.
        """
        try:
            response =
self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error describing gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a DescribeGateway Referência da API AWS SDK for Python (Boto3).

`GetAssetPropertyValue`

O código de exemplo a seguir mostra como usar `GetAssetPropertyValue`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
```

```
        """
        Gets the value of an AWS IoT SiteWise Asset Property.

        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Asset %s or property %s does not exist.", asset_id, property_id
                )
            else:
                logger.error(
                    "Error getting asset property value. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a GetAssetPropertyValue Referência da API AWS SDK for Python (Boto3).

`ListAssetModels`

O código de exemplo a seguir mostra como usar `ListAssetModels`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def list_asset_models(self) -> List[Dict[str, Any]]:
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
 model.

        """
        try:
            asset_models = []
            paginator = self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
```

```
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListAssetModels Referência da API AWS SDK for Python (Boto3).

Exemplos do Amazon Keyspaces usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com Amazon Keyspaces.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Keyspaces

O exemplo de código a seguir mostra como começar a usar o Amazon Keyspaces.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_keyspaces(keyspaces_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Keyspaces (for Apache
 Cassandra)
    client and list the keyspaces in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param keyspaces_client: A Boto3 Amazon Keyspaces Client object. This object
 wraps
                             the low-level Amazon Keyspaces service API.
    """
    print("Hello, Amazon Keyspaces! Let's list some of your keyspaces:\n")
    for ks in keyspaces_client.list_keyspaces(maxResults=5).get("keyspaces", []):
        print(ks["keyspaceName"])
        print(f"\t{ks['resourceArn']}")


if __name__ == "__main__":
    hello_keyspaces(boto3.client("keyspaces"))
```

- Para obter detalhes da API, consulte a ListKeyspaces Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um keyspace e uma tabela. O esquema da tabela contém dados do filme e tem a recuperação para um ponto no tempo.
- Conectar-se ao keyspace usando uma conexão TLS segura com autenticação SigV4.
- Consultar a tabela. Adicionar, recuperar e atualizar dados do filme.
- Atualizar a tabela. Adicionar uma coluna para rastrear os filmes assistidos.
- Restaurar a tabela ao estado anterior e limpar os recursos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class KeyspaceScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Keyspaces."""

    def __init__(self, ks_wrapper):
        """
        :param ks_wrapper: An object that wraps Amazon Keyspace actions.
        """
        self.ks_wrapper = ks_wrapper

    @demo_func
    def create_keyspace(self):
        """
        1. Creates a keyspace.
        2. Lists up to 10 keyspaces in your account.
        """
        print("Let's create a keyspace.")
        ks_name = q.ask(
            "Enter a name for your new keyspace.\nThe name can contain only letters,
 "
            "numbers and underscores: ",
            q.non_empty,
        )
```

```
        if self.ks_wrapper.exists_keyspace(ks_name):
            print(f"A keyspace named {ks_name} exists.")
        else:
            ks_arn = self.ks_wrapper.create_keyspace(ks_name)
            ks_exists = False
            while not ks_exists:
                wait(3)
                ks_exists = self.ks_wrapper.exists_keyspace(ks_name)
            print(f"Created a new keyspace.\n\t{ks_arn}.")
        print("The first 10 keyspaces in your account are:\n")
        self.ks_wrapper.list_keyspaces(10)

    @demo_func
    def create_table(self):
        """
        1. Creates a table in the keyspace. The table is configured with a schema to
hold
           movie data and has point-in-time recovery enabled.
        2. Waits for the table to be in an active state.
        3. Displays schema information for the table.
        4. Lists tables in the keyspace.
        """
        print("Let's create a table for movies in your keyspace.")
        table_name = q.ask("Enter a name for your table: ", q.non_empty)
        table = self.ks_wrapper.get_table(table_name)
        if table is not None:
            print(
                f"A table named {table_name} already exists in keyspace "
                f"{self.ks_wrapper.ks_name}."
            )
        else:
            table_arn = self.ks_wrapper.create_table(table_name)
            print(f"Created table {table_name}:\n\t{table_arn}")
            table = {"status": None}
            print("Waiting for your table to be ready...")
            while table["status"] != "ACTIVE":
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
        print(f"Your table is {table['status']}. Its schema is:")
        pp(table["schemaDefinition"])
        print("\nThe tables in your keyspace are:\n")
        self.ks_wrapper.list_tables()

    @demo_func
```

```
    def ensure_tls_cert(self):
        """
        Ensures you have a TLS certificate available to use to secure the connection
        to the keyspace. This function downloads a default certificate or lets you
        specify your own.
        """
        print("To connect to your keyspace, you must have a TLS certificate.")
        print("Checking for TLS certificate...")
        cert_path = os.path.join(
            os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
        )
        if not os.path.exists(cert_path):
            cert_choice = q.ask(
                f"Press enter to download a certificate from {QueryManager.CERT_URL}
"
                f"or enter the full path to the certificate you want to use: "
            )
            if cert_choice:
                cert_path = cert_choice
            else:
                cert = requests.get(QueryManager.CERT_URL).text
                with open(cert_path, "w") as cert_file:
                    cert_file.write(cert)
        else:
            q.ask(f"Certificate {cert_path} found. Press Enter to continue.")
        print(
            f"Certificate {cert_path} will be used to secure the connection to your
keyspace."
        )
        return cert_path

    @demo_func
    def query_table(self, qm, movie_file):
        """
        1. Adds movies to the table from a sample movie data file.
        2. Gets a list of movies from the table and lets you select one.
        3. Displays more information about the selected movie.
        """
        qm.add_movies(self.ks_wrapper.table_name, movie_file)
        movies = qm.get_movies(self.ks_wrapper.table_name)
        print(f"Added {len(movies)} movies to the table:")
        sel = q.choose("Pick one to learn more about it: ", [m.title for m in
movies])
        movie_choice = qm.get_movie(
```

```
            self.ks_wrapper.table_name, movies[sel].title, movies[sel].year
        )
        print(movie_choice.title)
        print(f"\tReleased: {movie_choice.release_date}")
        print(f"\tPlot: {movie_choice.plot}")

    @demo_func
    def update_and_restore_table(self, qm):
        """
        1. Updates the table by adding a column to track watched movies.
        2. Marks some of the movies as watched.
        3. Gets the list of watched movies from the table.
        4. Restores to a movies_restored table at a previous point in time.
        5. Gets the list of movies from the restored table.
        """
        print("Let's add a column to record which movies you've watched.")
        pre_update_timestamp = datetime.utcnow()
        print(
            f"Recorded the current UTC time of {pre_update_timestamp} so we can
restore the table later."
        )
        self.ks_wrapper.update_table()
        print("Waiting for your table to update...")
        table = {"status": "UPDATING"}
        while table["status"] != "ACTIVE":
            wait(5)
            table = self.ks_wrapper.get_table(self.ks_wrapper.table_name)
        print("Column 'watched' added to table.")
        q.ask(
            "Let's mark some of the movies as watched. Press Enter when you're
ready.\n"
        )
        movies = qm.get_movies(self.ks_wrapper.table_name)
        for movie in movies[:10]:
            qm.watched_movie(self.ks_wrapper.table_name, movie.title, movie.year)
            print(f"Marked {movie.title} as watched.")
        movies = qm.get_movies(self.ks_wrapper.table_name, watched=True)
        print("-" * 88)
        print("The watched movies in our table are:\n")
        for movie in movies:
            print(movie.title)
        print("-" * 88)
        if q.ask(
```

```
            "Do you want to restore the table to the way it was before all of these
\n"
            "updates? Keep in mind, this can take up to 20 minutes. (y/n) ",
            q.is_yesno,
        ):
            starting_table_name = self.ks_wrapper.table_name
            table_name_restored =
 self.ks_wrapper.restore_table(pre_update_timestamp)
            table = {"status": "RESTORING"}
            while table["status"] != "ACTIVE":
                wait(10)
                table = self.ks_wrapper.get_table(table_name_restored)
            print(
                f"Restored {starting_table_name} to {table_name_restored} "
                f"at a point in time of {pre_update_timestamp}."
            )
            movies = qm.get_movies(table_name_restored)
            print("Now the movies in our table are:")
            for movie in movies:
                print(movie.title)

    def cleanup(self, cert_path):
        """
        1. Deletes the table and waits for it to be removed.
        2. Deletes the keyspace.

        :param cert_path: The path of the TLS certificate used in the demo. If the
                          certificate was downloaded during the demo, it is removed.
        """
        if q.ask(
            f"Do you want to delete your {self.ks_wrapper.table_name} table and "
            f"{self.ks_wrapper.ks_name} keyspace? (y/n) ",
            q.is_yesno,
        ):
            table_name = self.ks_wrapper.table_name
            self.ks_wrapper.delete_table()
            table = self.ks_wrapper.get_table(table_name)
            print("Waiting for the table to be deleted.")
            while table is not None:
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
            print("Table deleted.")
            self.ks_wrapper.delete_keyspace()
            print(
```

```
                "Keyspace deleted. If you chose to restore your table during the "
                "demo, the original table is also deleted."
            )
            if cert_path == os.path.join(
                os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
            ) and os.path.exists(cert_path):
                os.remove(cert_path)
                print("Removed certificate that was downloaded for this demo.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

        print("-" * 88)
        print("Welcome to the Amazon Keyspaces (for Apache Cassandra) demo.")
        print("-" * 88)

        self.create_keyspace()
        self.create_table()
        cert_file_path = self.ensure_tls_cert()
        # Use a context manager to ensure the connection to the keyspace is closed.
        with QueryManager(
            cert_file_path, boto3.DEFAULT_SESSION, self.ks_wrapper.ks_name
        ) as qm:
            self.query_table(qm, "../../../resources/sample_files/movies.json")
            self.update_and_restore_table(qm)
        self.cleanup(cert_file_path)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = KeyspaceScenario(KeyspaceWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que envolva ações de keyspace e tabelas.

```
class KeyspaceWrapper:
```

```
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.
```

```
        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems": limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
```

```
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response


    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_table(self):
        """
        Updates the schema of the table.
```

```
        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return restored_table_name


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Defina uma classe que crie uma conexão TLS com um keyspace, autentique com SigV4 e envie consultas CQL para uma tabela no keyspace.

```
class QueryManager:
    """
    Manages queries to an Amazon Keyspaces (for Apache Cassandra) keyspace.
    Queries are secured by TLS and authenticated by using the Signature V4 (SigV4)
    AWS signing protocol. This is more secure than sending username and password
    with a plain-text authentication provider.

    This example downloads a default certificate to secure TLS, or lets you specify
    your own.

    This example uses a table of movie data to demonstrate basic queries.
    """

    DEFAULT_CERT_FILE = "sf-class2-root.crt"
    CERT_URL = f"https://certs.secureserver.net/repository/sf-class2-root.crt"

    def __init__(self, cert_file_path, boto_session, keyspace_name):
        """
        :param cert_file_path: The path and file name of the certificate used for
 TLS.
        :param boto_session: A Boto3 session. This is used to acquire your AWS
 credentials.
        :param keyspace_name: The name of the keyspace to connect.
        """
        self.cert_file_path = cert_file_path
        self.boto_session = boto_session
        self.ks_name = keyspace_name
        self.cluster = None
        self.session = None

    def __enter__(self):
        """
        Creates a session connection to the keyspace that is secured by TLS and
        authenticated by SigV4.
        """
        ssl_context = SSLContext(PROTOCOL_TLSv1_2)
        ssl_context.load_verify_locations(self.cert_file_path)
        ssl_context.verify_mode = CERT_REQUIRED
```

```
        auth_provider = SigV4AuthProvider(self.boto_session)
        contact_point = f"cassandra.{self.boto_session.region_name}.amazonaws.com"
        exec_profile = ExecutionProfile(
            consistency_level=ConsistencyLevel.LOCAL_QUORUM,
            load_balancing_policy=DCAwareRoundRobinPolicy(),
        )
        self.cluster = Cluster(
            [contact_point],
            ssl_context=ssl_context,
            auth_provider=auth_provider,
            port=9142,
            execution_profiles={EXEC_PROFILE_DEFAULT: exec_profile},
            protocol_version=4,
        )
        self.cluster.__enter__()
        self.session = self.cluster.connect(self.ks_name)
        return self

    def __exit__(self, *args):
        """
        Exits the cluster. This shuts down all existing session connections.
        """
        self.cluster.__exit__(*args)

    def add_movies(self, table_name, movie_file_path):
        """
        Gets movies from a JSON file and adds them to a table in the keyspace.

        :param table_name: The name of the table.
        :param movie_file_path: The path and file name of a JSON file that contains
movie data.
        """
        with open(movie_file_path, "r") as movie_file:
            movies = json.loads(movie_file.read())
        stmt = self.session.prepare(
            f"INSERT INTO {table_name} (year, title, release_date, plot) VALUES
(?, ?, ?, ?);"
        )
        for movie in movies[:20]:
            self.session.execute(
                stmt,
                parameters=[
                    movie["year"],
                    movie["title"],
```

```
                    date.fromisoformat(movie["info"]["release_date"].partition("T")
[0]),
                    movie["info"]["plot"],
                ],
            )

    def get_movies(self, table_name, watched=None):
        """
        Gets the title and year of the full list of movies from the table.

        :param table_name: The name of the movie table.
        :param watched: When specified, the returned list of movies is filtered to
                        either movies that have been watched or movies that have not
                        been watched. Otherwise, all movies are returned.
        :return: A list of movies in the table.
        """
        if watched is None:
            stmt = SimpleStatement(f"SELECT title, year from {table_name}")
            params = None
        else:
            stmt = SimpleStatement(
                f"SELECT title, year from {table_name} WHERE watched = %s ALLOW
 FILTERING"
            )
            params = [watched]
        return self.session.execute(stmt, parameters=params).all()

    def get_movie(self, table_name, title, year):
        """
        Gets a single movie from the table, by title and year.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        :return: The requested movie.
        """
        return self.session.execute(
            SimpleStatement(
                f"SELECT * from {table_name} WHERE title = %s AND year = %s"
            ),
            parameters=[title, year],
        ).one()

    def watched_movie(self, table_name, title, year):
```

```
        """
        Updates a movie as having been watched.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        """
        self.session.execute(
            SimpleStatement(
                f"UPDATE {table_name} SET watched=true WHERE title = %s AND year =
%s"
            ),
            parameters=[title, year],
        )
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Ações

`CreateKeyspace`

O código de exemplo a seguir mostra como usar `CreateKeyspace`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn
```

- Para obter detalhes da API, consulte a CreateKeyspace Referência da API AWS SDK for Python (Boto3).

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)
```

```
    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]
```

- Para obter detalhes da API, consulte a CreateTable Referência da API AWS SDK for Python (Boto3).

`DeleteKeyspace`

O código de exemplo a seguir mostra como usar `DeleteKeyspace`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteKeyspace Referência da API AWS SDK for Python (Boto3).

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def delete_table(self):
```

```
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTable Referência da API AWS SDK for Python (Boto3).

`GetKeyspace`

O código de exemplo a seguir mostra como usar `GetKeyspace`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
```

```
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists
```

- Para obter detalhes da API, consulte a GetKeyspace Referência da API AWS SDK for Python (Boto3).

`GetTable`

O código de exemplo a seguir mostra como usar `GetTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- Para obter detalhes da API, consulte a GetTable Referência da API AWS SDK for Python (Boto3).

`ListKeyspaces`

O código de exemplo a seguir mostra como usar `ListKeyspaces`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
```

```
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems": limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListKeyspaces Referência da API AWS SDK for Python (Boto3).

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListTables Referência da API AWS SDK for Python (Boto3).

`RestoreTable`

O código de exemplo a seguir mostra como usar `RestoreTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def restore_table(self, restore_timestamp):
```

```
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return restored_table_name
```

- Para obter detalhes da API, consulte a RestoreTable Referência da API AWS SDK for Python (Boto3).

`UpdateTable`

O código de exemplo a seguir mostra como usar `UpdateTable`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateTable Referência da API AWS SDK for Python (Boto3).

Exemplos do Kinesis usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Kinesis.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Exemplos sem servidor

Ações

`CreateStream`

O código de exemplo a seguir mostra como usar `CreateStream`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def create(self, name, wait_until_exists=True):
        """
        Creates a stream.

        :param name: The name of the stream.
        :param wait_until_exists: When True, waits until the service reports that
                                  the stream exists, then queries for its metadata.
        """
        try:
            self.kinesis_client.create_stream(StreamName=name, ShardCount=1)
            self.name = name
            logger.info("Created stream %s.", name)
            if wait_until_exists:
                logger.info("Waiting until exists.")
                self.stream_exists_waiter.wait(StreamName=name)
                self.describe(name)
        except ClientError:
            logger.exception("Couldn't create stream %s.", name)
            raise
```

- Para obter detalhes da API, consulte a CreateStream Referência da API AWS SDK for Python (Boto3).

`DeleteStream`

O código de exemplo a seguir mostra como usar `DeleteStream`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def delete(self):
        """
        Deletes a stream.
        """
        try:
            self.kinesis_client.delete_stream(StreamName=self.name)
            self._clear()
            logger.info("Deleted stream %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete stream %s.", self.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteStream Referência da API AWS SDK for Python (Boto3).

`DescribeStream`

O código de exemplo a seguir mostra como usar `DescribeStream`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response["StreamDescription"]
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
```

```
            raise
        else:
            return self.details
```

- Para obter detalhes da API, consulte a DescribeStream Referência da API AWS SDK for Python (Boto3).

GetRecords

O código de exemplo a seguir mostra como usar `GetRecords`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def get_records(self, max_records):
        """
        Gets records from the stream. This function is a generator that first gets
        a shard iterator for the stream, then uses the shard iterator to get records
        in batches from the stream. The shard iterator can be accessed through the
        'details' property, which is populated using the 'describe' function of this
 class.
```

```
        Each batch of records is yielded back to the caller until the specified
        maximum number of records has been retrieved.

        :param max_records: The maximum number of records to retrieve.
        :return: Yields the current batch of retrieved records.
        """
        try:
            response = self.kinesis_client.get_shard_iterator(
                StreamName=self.name,
                ShardId=self.details["Shards"][0]["ShardId"],
                ShardIteratorType="LATEST",
            )
            shard_iter = response["ShardIterator"]
            record_count = 0
            while record_count < max_records:
                response = self.kinesis_client.get_records(
                    ShardIterator=shard_iter, Limit=10
                )
                shard_iter = response["NextShardIterator"]
                records = response["Records"]
                logger.info("Got %s records.", len(records))
                record_count += len(records)
                yield records
        except ClientError:
            logger.exception("Couldn't get records from stream %s.", self.name)
            raise



    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response["StreamDescription"]
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
            raise
```

```
        else:
            return self.details
```

- Para obter detalhes da API, consulte a GetRecords Referência da API AWS SDK for Python (Boto3).

PutRecord

O código de exemplo a seguir mostra como usar `PutRecord`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def put_record(self, data, partition_key):
        """
        Puts data into the stream. The data is formatted as JSON before it is passed
        to the stream.

        :param data: The data to put in the stream.
        :param partition_key: The partition key to use for the data.
```

```
        :return: Metadata about the record, including its shard ID and sequence
number.
        """
        try:
            response = self.kinesis_client.put_record(
                StreamName=self.name, Data=json.dumps(data),
PartitionKey=partition_key
            )
            logger.info("Put record in stream %s.", self.name)
        except ClientError:
            logger.exception("Couldn't put record in stream %s.", self.name)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a PutRecord Referência da API AWS SDK for Python (Boto3).

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import base64
```

```
def lambda_handler(event, context):

    for record in event['Records']:
        try:
            print(f"Processed Kinesis Event - EventID: {record['eventID']}")
            record_data = base64.b64decode(record['kinesis']
['data']).decode('utf-8')
            print(f"Record Data: {record_data}")
            # TODO: Do interesting work based on the new data
        except Exception as e:
            print(f"An error occurred {e}")
            raise e
    print(f"Successfully processed {len(event['Records'])} records.")
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["kinesis"]["sequenceNumber"]
```

```
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

AWS KMS exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with AWS KMS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma chave do KMS.
- Listar chaves KMS para sua conta e obter detalhes sobre elas.
- Habilitar e desabilitar chaves do KMS.
- Gerar uma chave de dados simétrica que possa ser usada para criptografia do lado do cliente.
- Gere uma chave assimétrica usada para assinar dados digitalmente.

- Marcar chaves com tags.
- Excluir chaves do KMS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KMSScenario:
    """Runs an interactive scenario that shows how to get started with KMS."""

    def __init__(
        self,
        key_manager: KeyManager,
        key_encryption: KeyEncrypt,
        alias_manager: AliasManager,
        grant_manager: GrantManager,
        key_policy: KeyPolicy,
    ):
        self.key_manager = key_manager
        self.key_encryption = key_encryption
        self.alias_manager = alias_manager
        self.grant_manager = grant_manager
        self.key_policy = key_policy
        self.key_id = ""
        self.alias_name = ""
        self.asymmetric_key_id = ""

    def kms_scenario(self):
        key_description = "Created by the AWS KMS API"

        print(DASHES)
        print(
            """
Welcome to the AWS Key Management SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management using the AWS SDK
 for Python (Boto3).
```

```
The AWS Key Management Service (KMS) is a secure and highly available service that
 allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys, making
 it easier to meet your
data protection and regulatory compliance requirements.

This Basics scenario creates two key types:

- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...
        """
        )
        q.ask("Press Enter to continue...")

        print(DASHES)
        print(f"1. Create a symmetric KMS key\n")
        print(
            f"First, the program will creates a symmetric KMS key that you can used
 to encrypt and decrypt data."
        )
        q.ask("Press Enter to continue...")
        self.key_id = self.key_manager.create_key(key_description)["KeyId"]
        print(f"A symmetric key was successfully created {self.key_id}.")
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(
            """
2. Enable a KMS key

By default, when the SDK creates an AWS key, it is enabled. The next bit of code
 checks to
determine if the key is enabled.
        """
        )
        q.ask("Press Enter to continue...")
        is_enabled = self.is_key_enabled(self.key_id)
        print(f"Is the key enabled? {is_enabled}")
        if not is_enabled:
            self.key_manager.enable_key(self.key_id)
        q.ask("Press Enter to continue...")
```

```
        print(DASHES)
        print(f"3. Encrypt data using the symmetric KMS key")
        plain_text = "Hello, AWS KMS!"
        print(
            f"""
One of the main uses of symmetric keys is to encrypt and decrypt data.
Next, the code encrypts the string "{plain_text}" with the SYMMETRIC_DEFAULT
 encryption algorithm.
        """
        )
        q.ask("Press Enter to continue...")
        encrypted_text = self.key_encryption.encrypt(self.key_id, plain_text)
        print(DASHES)
        print(f"4. Create an alias")
        print(
            """
Now, the program will create an alias for the KMS key. An alias is a friendly name
 that you
can associate with a KMS key. The alias name should be prefixed with 'alias/'.
        """
        )
        alias_name = q.ask("Enter an alias name: ", q.non_empty)
        self.alias_manager.create_alias(self.key_id, alias_name)
        print(f"{alias_name} was successfully created.")
        self.alias_name = alias_name
        print(DASHES)
        print(f"5. List all of your aliases")
        q.ask("Press Enter to continue...")
        self.alias_manager.list_aliases(10)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"6. Enable automatic rotation of the KMS key")
        print(
            """

By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days) from
 the enable date and every year
thereafter.
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.enable_key_rotation(self.key_id)
        print(DASHES)
```

```
        print(f"Key rotation has been enabled for key with id {self.key_id}")
        print(
            """
7. Create a grant

A grant is a policy instrument that allows Amazon Web Services principals to use KMS
 keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage grants.
When authorizing access to a KMS key, grants are considered along with key policies
 and IAM policies.
        """
        )
        print(
            """
To create a grant you must specify a account_id. To specify the grantee account_id,
 use the Amazon Resource Name (ARN)
of an AWS account_id. Valid principals include AWS accounts, IAM users, IAM roles,
 federated users,
and assumed role users.
        """
        )
        account_id = q.ask(
            "Enter an account_id, or press enter to skip creating a grant... "
        )
        grant = None
        if account_id != "":
            grant = self.grant_manager.create_grant(
                self.key_id,
                account_id,
                [
                    "Encrypt",
                    "Decrypt",
                    "DescribeKey",
                ],
            )
            print(f"Grant created successfully with ID: {grant['GrantId']}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(DASHES)
        print(f"8. List grants for the KMS key")
        q.ask("Press Enter to continue...")
        self.grant_manager.list_grants(self.key_id)
        q.ask("Press Enter to continue...")
```

```
        print(DASHES)
        print(f"9. Revoke the grant")
        print(
            """
The revocation of a grant immediately removes the permissions and access that the
 grant had provided.
This means that any account_id (user, role, or service) that was granted access to
 perform specific
KMS operations on a KMS key will no longer be able to perform those operations.
        """
        )
        q.ask("Press Enter to continue...")

        if grant is not None:
            self.grant_manager.revoke_grant(self.key_id, grant["GrantId"])
            print(f"Grant ID: {grant['GrantId']} was successfully revoked!")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"10. Decrypt the data\n")
        print(
            """
Lets decrypt the data that was encrypted in an early step.
The code uses the same key to decrypt the string that we encrypted earlier in the
 program.
        """
        )
        q.ask("Press Enter to continue...")
        decrypted_data = self.key_encryption.decrypt(self.key_id, encrypted_text)
        print(f"Data decrypted successfully for key ID: {self.key_id}")
        print(f"Decrypted data: {decrypted_data}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"11. Replace a key policy\n")
        print(
            """
A key policy is a resource policy for a KMS key. Key policies are the primary way to
 control
access to KMS keys. Every KMS key must have exactly one key policy. The statements
 in the key policy
determine who has permission to use the KMS key and how they can use it.
You can also use IAM policies and grants to control access to the KMS key, but every
 KMS key
```

```
must have a key policy.

By default, when you create a key by using the SDK, a policy is created that
gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.
{
"Version":"2012-10-17",
"Statement": [{
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::0000000000:root"},
"Action": "kms:*",
"Resource": "*"
}]
}
        """
        )
        account_id = q.ask("Enter your account ID or press enter to skip: ")
        if account_id != "":
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": f"arn:aws:iam::{account_id}:root"},
                        "Action": "kms:*",
                        "Resource": "*",
                    }
                ],
            }

            self.key_policy.set_new_policy(self.key_id, policy)
            print("Key policy replacement succeeded.")
            q.ask("Press Enter to continue...")
        else:
            print("Skipping replacing the key policy.")

        print(DASHES)
        print(f"12. Get the key policy\n")
        print(
            f"The next bit of code that runs gets the key policy to make sure it
 exists."
        )
        q.ask("Press Enter to continue...")
```

```
        policy = self.key_policy.get_policy(self.key_id)
        print(f"The key policy is: {policy}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"13. Create an asymmetric KMS key and sign your data\n")
        print(
            """
        Signing your data with an AWS key can provide several benefits that make it
an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
        """
        )
        q.ask("Press Enter to continue...")
        print(f"Sign and verify data operation succeeded.")
        self.asymmetric_key_id = self.key_manager.create_asymmetric_key()
        message = "Here is the message that will be digitally signed"
        signature = self.key_encryption.sign(self.asymmetric_key_id, message)
        if self.key_encryption.verify(self.asymmetric_key_id, message, signature):
            print("Signature verification succeeded.")
        else:
            print("Signature verification failed.")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"14. Tag your symmetric KMS Key\n")
        print(
            """
        By using tags, you can improve the overall management, security, and
governance of your
        KMS keys, making it easier to organize, track, and control access to your
encrypted data within
        your AWS environment
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.tag_resource(self.key_id, "Environment", "Production")
        self.clean_up()

    def is_key_enabled(self, key_id: str) -> bool:
```

```
        """
        Check if the key is enabled or not.

        :param key_id: The key to check.
        :return: True if the key is enabled, otherwise False.
        """
        response = self.key_manager.describe_key(key_id)
        return response["Enabled"] is True

    def clean_up(self):
        """
        Delete resources created by this scenario.
        """
        if self.alias_name != "":
            print(f"Deleting the alias {self.alias_name}.")
            self.alias_manager.delete_alias(self.alias_name)
        window = 7  # The window in days for a scheduled deletion.
        if self.key_id != "":
            print(
                """
Warning:
Deleting a KMS key is a destructive and potentially dangerous operation. When a KMS
 key is deleted,
all data that was encrypted under the KMS key is unrecoverable.
                """
            )
            if q.ask(
                f"Do you want to delete the key with ID {self.key_id} (y/n)?",
                q.is_yesno,
            ):
                print(
                    f"The key {self.key_id} will be deleted with a window of
 {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.key_id, window)
                self.key_id = ""

        if self.asymmetric_key_id != "":
            if q.ask(
                f"Do you want to delete the asymmetric key with ID
 {self.asymmetric_key_id} (y/n)?",
                q.is_yesno,
            ):
```

```
                print(
                    f"The key {self.asymmetric_key_id} will be deleted with a window
 of {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.asymmetric_key_id, window)
                self.asymmetric_key_id = ""


if __name__ == "__main__":
    kms_scenario = None
    try:
        kms_client = boto3.client("kms")
        a_key_manager = KeyManager(kms_client)
        a_key_encrypt = KeyEncrypt(kms_client)
        an_alias_manager = AliasManager(kms_client)
        a_grant_manager = GrantManager(kms_client)
        a_key_policy = KeyPolicy(kms_client)
        kms_scenario = KMSScenario(
            key_manager=a_key_manager,
            key_encryption=a_key_encrypt,
            alias_manager=an_alias_manager,
            grant_manager=a_grant_manager,
            key_policy=a_key_policy,
        )
        kms_scenario.kms_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo!")
        if kms_scenario is not None:
            kms_scenario.clean_up()
```

Classe e métodos do wrapper para gerenciamento de chaves do KMS.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
```

```
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
```

```
            )
            raise


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_asymmetric_key(self) -> str:
        """
        Creates an asymmetric key in AWS KMS for signing messages.

        :return: The ID of the created key.
        """
        try:
            key = self.kms_client.create_key(
                KeySpec="RSA_2048", KeyUsage="SIGN_VERIFY", Origin="AWS_KMS"
            )["KeyMetadata"]
            self.created_keys.append(key)
            return key["KeyId"]
        except ClientError as err:
            logger.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
        """
        Add or edit tags on a customer managed key.
```

```
        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

Classe e métodos do wrapper para aliases de chaves do KMS.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

```
    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

Classe e métodos do wrapper para criptografia de chaves do KMS.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
 Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
 {response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
```

```
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id, CiphertextBlob=cipher_text)
[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
```

```
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Classe e métodos do wrapper para concessões de chaves do KMS.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
```

```
    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
```

```
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

Classe e métodos do wrapper para políticas de chaves do KMS.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_new_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the existing
        policy, so care is taken to add a statement to the existing list of
 statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: A new key policy. The key policy must allow the calling
 principal to make a subsequent
                       PutKeyPolicy request on the KMS key. This reduces the risk
 that the KMS key becomes unmanageable
        """

        try:
            self.kms_client.put_key_policy(KeyId=key_id, Policy=json.dumps(policy))
        except ClientError as err:
            logger.error(
                "Couldn't set policy for key %s. Here's why %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise



    def get_policy(self, key_id: str) -> dict[str, str]:
        """
```

```
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants

- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

Ações

`CreateAlias`

O código de exemplo a seguir mostra como usar `CreateAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.
```

```
        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Para obter detalhes da API, consulte a CreateAlias Referência da API AWS SDK for Python (Boto3).

`CreateGrant`

O código de exemplo a seguir mostra como usar `CreateGrant`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
```

```
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateGrant Referência da API AWS SDK for Python (Boto3).

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateKey Referência da API AWS SDK for Python (Boto3).

`Decrypt`

O código de exemplo a seguir mostra como usar `Decrypt`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
```

```
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id, CiphertextBlob=cipher_text)
[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK para Python (Boto3).

`DeleteAlias`

O código de exemplo a seguir mostra como usar `DeleteAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.
```

```
        :return: An instance of AliasManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteAlias Referência da API AWS SDK for Python (Boto3).

`DescribeKey`

O código de exemplo a seguir mostra como usar `DescribeKey`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeKey Referência da API AWS SDK for Python (Boto3).

`DisableKey`

O código de exemplo a seguir mostra como usar `DisableKey`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def disable_key(self, key_id: str) -> None:
        try:
            self.kms_client.disable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't disable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DisableKey Referência da API AWS SDK for Python (Boto3).

EnableKey

O código de exemplo a seguir mostra como usar `EnableKey`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key(self, key_id: str) -> None:
        """
        Enables a key. Gets the key state after each state change.

        :param key_id: The ARN or ID of the key to enable.
        """
        try:
            self.kms_client.enable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable key '%s'. Here's why: %s",
```

```
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a EnableKey Referência da API AWS SDK for Python (Boto3).

`EnableKeyRotation`

O código de exemplo a seguir mostra como usar `EnableKeyRotation`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.
```

```
        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a EnableKeyRotation Referência da API AWS SDK for Python (Boto3).

`Encrypt`

O código de exemplo a seguir mostra como usar `Encrypt`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
```

```
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
{response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK para Python (Boto3).

`GenerateDataKey`

O código de exemplo a seguir mostra como usar `GenerateDataKey`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def generate_data_key(self, key_id):
        """
        Generates a symmetric data key that can be used for client-side encryption.
        """
        answer = input(
            f"Do you want to generate a symmetric data key from key {key_id} (y/n)?
 "
        )
        if answer.lower() == "y":
            try:
                data_key = self.kms_client.generate_data_key(
                    KeyId=key_id, KeySpec="AES_256"
                )
            except ClientError as err:
                logger.error(
                    "Couldn't generate a data key for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
```

```
                )
            else:
                pprint(data_key)
```

- Para obter detalhes da API, consulte a GenerateDataKey Referência da API AWS SDK for Python (Boto3).

`GetKeyPolicy`

O código de exemplo a seguir mostra como usar `GetKeyPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
```

```
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Para obter detalhes da API, consulte a GetKeyPolicy Referência da API AWS SDK for Python (Boto3).

`ListAliases`

O código de exemplo a seguir mostra como usar `ListAliases`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
```

```
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListAliases Referência da API AWS SDK for Python (Boto3).

`ListGrants`

O código de exemplo a seguir mostra como usar `ListGrants`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
```

```
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListGrants Referência da API AWS SDK for Python (Boto3).

`ListKeyPolicies`

O código de exemplo a seguir mostra como usar `ListKeyPolicies`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS client.
```

```
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)[
                "PolicyNames"
            ]
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- Para obter detalhes da API, consulte a ListKeyPolicies Referência da API AWS SDK for Python (Boto3).

`ListKeys`

O código de exemplo a seguir mostra como usar `ListKeys`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            key_paginator = self.kms_client.get_paginator("list_keys")
            for key_page in key_paginator.paginate(PaginationConfig={"PageSize":
 10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page["Keys"])
                if key_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} keys (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s",
                err.response["Error"]["Message"],
            )
```

- Para obter detalhes da API, consulte a ListKeys Referência da API AWS SDK for Python (Boto3).

PutKeyPolicy

O código de exemplo a seguir mostra como usar `PutKeyPolicy`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the existing
        policy, so care is taken to add a statement to the existing list of
 statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: The existing policy of the key.
        :return: None
        """
        principal = input(
```

```
            "Enter the ARN of an IAM role to set as the principal on the policy: "
        )
        if key_id != "" and principal != "":
            # The updated policy replaces the existing policy. Add a new statement
to
            # the list along with the original policy statements.
            policy["Statement"].append(
                {
                    "Sid": "Allow access for ExampleRole",
                    "Effect": "Allow",
                    "Principal": {"AWS": principal},
                    "Action": [
                        "kms:Encrypt",
                        "kms:GenerateDataKey*",
                        "kms:Decrypt",
                        "kms:DescribeKey",
                        "kms:ReEncrypt*",
                    ],
                    "Resource": "*",
                }
            )
            try:
                self.kms_client.put_key_policy(KeyId=key_id,
Policy=json.dumps(policy))
            except ClientError as err:
                logger.error(
                    "Couldn't set policy for key %s. Here's why %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                print(f"Set policy for key {key_id}.")
        else:
            print("Skipping set policy demo.")
```

- Para obter detalhes da API, consulte a PutKeyPolicy Referência da API AWS SDK for Python (Boto3).

`ReEncrypt`

O código de exemplo a seguir mostra como usar `ReEncrypt`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def re_encrypt(self, source_key_id, cipher_text):
        """
        Takes ciphertext previously encrypted with one key and reencrypt it by using
        another key.

        :param source_key_id: The ARN or ID of the original key used to encrypt the
                              ciphertext.
        :param cipher_text: The encrypted ciphertext.
        :return: The ciphertext encrypted by the second key.
        """
        destination_key_id = input(
            f"Your ciphertext is currently encrypted with key {source_key_id}. "
            f"Enter another key ID or ARN to reencrypt it: "
        )
        if destination_key_id != "":
            try:
```

```
                cipher_text = self.kms_client.re_encrypt(
                    SourceKeyId=source_key_id,
                    DestinationKeyId=destination_key_id,
                    CiphertextBlob=cipher_text,
                )["CiphertextBlob"]
            except ClientError as err:
                logger.error(
                    "Couldn't reencrypt your ciphertext. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Reencrypted your ciphertext as: {cipher_text}")
                return cipher_text
        else:
            print("Skipping reencryption demo.")
```

- Para obter detalhes da API, consulte a ReEncrypt Referência da API AWS SDK for Python (Boto3).

`RetireGrant`

O código de exemplo a seguir mostra como usar `RetireGrant`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.
```

```
        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def retire_grant(self, grant):
        """
        Retires a grant so that it can no longer be used.

        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant["GrantToken"])
        except ClientError as err:
            logger.error(
                "Couldn't retire grant %s. Here's why: %s",
                grant["GrantId"],
                err.response["Error"]["Message"],
            )
        else:
            print(f"Grant {grant['GrantId']} retired.")
```

- Para obter detalhes da API, consulte a RetireGrant Referência da API AWS SDK for Python (Boto3).

`RevokeGrant`

O código de exemplo a seguir mostra como usar `RevokeGrant`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a RevokeGrant Referência da API AWS SDK for Python (Boto3).

`ScheduleKeyDeletion`

O código de exemplo a seguir mostra como usar `ScheduleKeyDeletion`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
 When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
```

```
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ScheduleKeyDeletion Referência da API AWS SDK for Python (Boto3).

`Sign`

O código de exemplo a seguir mostra como usar `Sign`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def sign(self, key_id: str, message: str) -> str:
        """
```

```
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Consulte detalhes da API em Sign na Referência API do AWS SDK para Python (Boto3).

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
```

```
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a TagResource Referência da API AWS SDK for Python (Boto3).

`UpdateAlias`

O código de exemplo a seguir mostra como usar `UpdateAlias`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def update_alias(self, alias, current_key_id):
        """
        Updates an alias by assigning it to another key.

        :param alias: The alias to reassign.
        :param current_key_id: The ARN or ID of the key currently associated with
 the alias.
        """
        new_key_id = input(
            f"Alias {alias} is currently associated with {current_key_id}. "
            f"Enter another key ID or ARN that you want to associate with {alias}: "
        )
        if new_key_id != "":
            try:
                self.kms_client.update_alias(AliasName=alias,
 TargetKeyId=new_key_id)
            except ClientError as err:
```

```
                logger.error(
                    "Couldn't associate alias %s with key %s. Here's why: %s",
                    alias,
                    new_key_id,
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Alias {alias} is now associated with key {new_key_id}.")
        else:
            print("Skipping alias update.")
```

- Para obter detalhes da API, consulte a UpdateAlias Referência da API AWS SDK for Python (Boto3).

`Verificar`

O código de exemplo a seguir mostra como usar `Verify`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Consulte detalhes da API em Verify na Referência da API do AWS SDK para Python (Boto3).

Exemplos de Lambda usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Lambda

O exemplo de código a seguir mostra como começar a usar o Lambda.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def main():
    """
```

```
    List the Lambda functions in your AWS account.
    """
    # Create the Lambda client
    lambda_client = boto3.client("lambda")

    # Use the paginator to list the functions
    paginator = lambda_client.get_paginator("list_functions")
    response_iterator = paginator.paginate()

    print("Here are the Lambda functions in your account:")
    for page in response_iterator:
        for function in page["Functions"]:
            print(f"  {function['FunctionName']}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ListFunctions Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina um manipulador do Lambda que aumente um número.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)


def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
 number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get("action")
    if action == "increment":
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Defina um segundo manipulador do Lambda que faça operações aritméticas.

```
import logging
import os


logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
 function.
ACTIONS = {
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}


def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)

    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)
```

```
    response = {"result": result}
    return response
```

Crie funções que envolvam ações do Lambda.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    @staticmethod
    def create_deployment_package(source_file, destination_file):
        """
        Creates a Lambda deployment package in .zip format in an in-memory buffer.
 This
        buffer can be passed directly to Lambda when creating the function.

        :param source_file: The name of the file that contains the Lambda handler
                            function.
        :param destination_file: The name to give the file when it's deployed to
 Lambda.
        :return: The deployment package.
        """
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, "w") as zipped:
            zipped.write(source_file, destination_file)
        buffer.seek(0)
        return buffer.read()

    def get_iam_role(self, iam_role_name):
        """
        Get an AWS Identity and Access Management (IAM) role.

        :param iam_role_name: The name of the role to retrieve.
        :return: The IAM role.
        """
        role = None
        try:
```

```
            temp_role = self.iam_resource.Role(iam_role_name)
            temp_role.load()
            role = temp_role
            logger.info("Got IAM role %s", role.name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                logger.info("IAM role %s does not exist.", iam_role_name)
            else:
                logger.error(
                    "Couldn't get IAM role %s. Here's why: %s: %s",
                    iam_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return role

    def create_iam_role_for_lambda(self, iam_role_name):
        """
        Creates an IAM role that grants the Lambda function basic permissions. If a
        role with the specified name already exists, it is used for the demo.

        :param iam_role_name: The name of the role to create.
        :return: The role and a value that indicates whether the role is newly
 created.
        """
        role = self.get_iam_role(iam_role_name)
        if role is not None:
            return role, False

        lambda_assume_role_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "lambda.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"

        try:
```

```
            role = self.iam_resource.create_role(
                RoleName=iam_role_name,
                AssumeRolePolicyDocument=json.dumps(lambda_assume_role_policy),
            )
            logger.info("Created role %s.", role.name)
            role.attach_policy(PolicyArn=policy_arn)
            logger.info("Attached basic execution policy to role %s.", role.name)
        except ClientError as error:
            if error.response["Error"]["Code"] == "EntityAlreadyExists":
                role = self.iam_resource.Role(iam_role_name)
                logger.warning("The role %s already exists. Using it.",
iam_role_name)
            else:
                logger.exception(
                    "Couldn't create role %s or attach policy %s.",
                    iam_role_name,
                    policy_arn,
                )
                raise

        return role, True

    def get_function(self, function_name):
        """
        Gets data about a Lambda function.

        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response = self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

```
    def create_function(
        self, function_name, handler_name, iam_role, deployment_package
    ):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function. This
                             must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the function
                                   code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime="python3.9",
                Role=iam_role.arn,
                Handler=handler_name,
                Code={"ZipFile": deployment_package},
                Publish=True,
            )
            function_arn = response["FunctionArn"]
            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)
            logger.info(
                "Created function '%s' with ARN: '%s'.",
                function_name,
                response["FunctionArn"],
            )
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn


    def delete_function(self, function_name):
        """
        Deletes a Lambda function.
```

```
        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise


    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This dict
                                is serialized to JSON before it is sent to Lambda.
        :param get_log: When true, the last 4 KB of the execution log are included
in
                        the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType="Tail" if get_log else "None",
            )
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response


    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
contains
        the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes in
                                   .zip format.
```

```
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={"Variables": env_vars}
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def list_functions(self):
        """
```

```
        Lists the Lambda functions for the current account.
        """
        try:
            func_paginator = self.lambda_client.get_paginator("list_functions")
            for func_page in func_paginator.paginate():
                for func in func_page["Functions"]:
                    print(func["FunctionName"])
                    desc = func.get("Description")
                    if desc:
                        print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
        except ClientError as err:
            logger.error(
                "Couldn't list functions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Crie uma função que execute o cenário.

```
class UpdateFunctionWaiter(CustomWaiter):
    """A custom waiter that waits until a function is successfully updated."""

    def __init__(self, client):
        super().__init__(
            "UpdateSuccess",
            "GetFunction",
            "Configuration.LastUpdateStatus",
            {"Successful": WaitState.SUCCESS, "Failed": WaitState.FAILURE},
            client,
        )

    def wait(self, function_name):
        self._wait(FunctionName=function_name)


def run_scenario(lambda_client, iam_resource, basic_file, calculator_file,
 lambda_name):
```

```
    """
    Runs the scenario.

    :param lambda_client: A Boto3 Lambda client.
    :param iam_resource: A Boto3 IAM resource.
    :param basic_file: The name of the file that contains the basic Lambda handler.
    :param calculator_file: The name of the file that contains the calculator Lambda
handler.
    :param lambda_name: The name to give resources created for the scenario, such as
the
                        IAM role and the Lambda function.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the AWS Lambda getting started with functions demo.")
    print("-" * 88)

    wrapper = LambdaWrapper(lambda_client, iam_resource)

    print("Checking for IAM role for Lambda...")
    iam_role, should_wait = wrapper.create_iam_role_for_lambda(lambda_name)
    if should_wait:
        logger.info("Giving AWS time to create resources...")
        wait(10)

    print(f"Looking for function {lambda_name}...")
    function = wrapper.get_function(lambda_name)
    if function is None:
        print("Zipping the Python script into a deployment package...")
        deployment_package = wrapper.create_deployment_package(
            basic_file, f"{lambda_name}.py"
        )
        print(f"...and creating the {lambda_name} Lambda function.")
        wrapper.create_function(
            lambda_name, f"{lambda_name}.lambda_handler", iam_role,
deployment_package
        )
    else:
        print(f"Function {lambda_name} already exists.")
    print("-" * 88)

    print(f"Let's invoke {lambda_name}. This function increments a number.")
    action_params = {
```

```
        "action": "increment",
        "number": q.ask("Give me a number to increment: ", q.is_int),
    }
    print(f"Invoking {lambda_name}...")
    response = wrapper.invoke_function(lambda_name, action_params)
    print(
        f"Incrementing {action_params['number']} resulted in "
        f"{json.load(response['Payload'])}"
    )
    print("-" * 88)

    print(f"Let's update the function to an arithmetic calculator.")
    q.ask("Press Enter when you're ready.")
    print("Creating a new deployment package...")
    deployment_package = wrapper.create_deployment_package(
        calculator_file, f"{lambda_name}.py"
    )
    print(f"...and updating the {lambda_name} Lambda function.")
    update_waiter = UpdateFunctionWaiter(lambda_client)
    wrapper.update_function_code(lambda_name, deployment_package)
    update_waiter.wait(lambda_name)
    print(f"This function uses an environment variable to control logging level.")
    print(f"Let's set it to DEBUG to get the most logging.")
    wrapper.update_function_configuration(
        lambda_name, {"LOG_LEVEL": logging.getLevelName(logging.DEBUG)}
    )

    actions = ["plus", "minus", "times", "divided-by"]
    want_invoke = True
    while want_invoke:
        print(f"Let's invoke {lambda_name}. You can invoke these actions:")
        for index, action in enumerate(actions):
            print(f"{index + 1}: {action}")
        action_params = {}
        action_index = q.ask(
            "Enter the number of the action you want to take: ",
            q.is_int,
            q.in_range(1, len(actions)),
        )
        action_params["action"] = actions[action_index - 1]
        print(f"You've chosen to invoke 'x {action_params['action']} y'.")
        action_params["x"] = q.ask("Enter a value for x: ", q.is_int)
        action_params["y"] = q.ask("Enter a value for y: ", q.is_int)
        print(f"Invoking {lambda_name}...")
```

```
        response = wrapper.invoke_function(lambda_name, action_params, True)
        print(
            f"Calculating {action_params['x']} {action_params['action']}
 {action_params['y']} "
            f"resulted in {json.load(response['Payload'])}"
        )
        q.ask("Press Enter to see the logs from the call.")
        print(base64.b64decode(response["LogResult"]).decode())
        want_invoke = q.ask("That was fun. Shall we do it again? (y/n) ",
 q.is_yesno)
    print("-" * 88)

    if q.ask(
        "Do you want to list all of the functions in your account? (y/n) ",
 q.is_yesno
    ):
        wrapper.list_functions()
    print("-" * 88)

    if q.ask("Ready to delete the function and role? (y/n) ", q.is_yesno):
        for policy in iam_role.attached_policies.all():
            policy.detach_role(RoleName=iam_role.name)
        iam_role.delete()
        print(f"Deleted role {lambda_name}.")
        wrapper.delete_function(lambda_name)
        print(f"Deleted function {lambda_name}.")

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        run_scenario(
            boto3.client("lambda"),
            boto3.resource("iam"),
            "lambda_handler_basic.py",
            "lambda_handler_calculator.py",
            "doc_example_lambda_calculator",
        )
    except Exception:
        logging.exception("Something went wrong with the demo!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def create_function(
        self, function_name, handler_name, iam_role, deployment_package
    ):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function. This
```

```
                                must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the function
                                   code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime="python3.9",
                Role=iam_role.arn,
                Handler=handler_name,
                Code={"ZipFile": deployment_package},
                Publish=True,
            )
            function_arn = response["FunctionArn"]
            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)
            logger.info(
                "Created function '%s' with ARN: '%s'.",
                function_name,
                response["FunctionArn"],
            )
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn
```

- Para obter detalhes da API, consulte a CreateFunction Referência da API AWS SDK for Python (Boto3).

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def delete_function(self, function_name):
        """
        Deletes a Lambda function.

        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise
```

- Para obter detalhes da API, consulte a DeleteFunction Referência da API AWS SDK for Python (Boto3).

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def get_function(self, function_name):
        """
        Gets data about a Lambda function.

        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response = self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- Para obter detalhes da API, consulte a GetFunction Referência da API AWS SDK for Python (Boto3).

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This dict
                                is serialized to JSON before it is sent to Lambda.
        :param get_log: When true, the last 4 KB of the execution log are included
 in
                        the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType="Tail" if get_log else "None",
            )
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para Python (Boto3).

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def list_functions(self):
        """
        Lists the Lambda functions for the current account.
        """
        try:
            func_paginator = self.lambda_client.get_paginator("list_functions")
            for func_page in func_paginator.paginate():
                for func in func_page["Functions"]:
                    print(func["FunctionName"])
                    desc = func.get("Description")
                    if desc:
                        print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
        except ClientError as err:
            logger.error(
                "Couldn't list functions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Para obter detalhes da API, consulte a ListFunctions Referência da API AWS SDK for Python (Boto3).

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
 contains
        the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes in
                                   .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a UpdateFunctionCode Referência da API AWS SDK for Python (Boto3).

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
```

```
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={"Variables": env_vars}
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a UpdateFunctionConfiguration Referência da API AWS SDK for Python (Boto3).

Cenários

Crie uma API REST para rastrear COVID-19 dados

O exemplo de código a seguir mostra como criar uma API REST que simula um sistema para rastrear casos diários de COVID-19 nos Estados Unidos usando dados fictícios.

SDK para Python (Boto3)

Mostra como usar o AWS Chalice com o AWS SDK para Python (Boto3) para criar uma API REST sem servidor que usa o Amazon API Gateway e o Amazon AWS Lambda DynamoDB. A API REST simula um sistema que rastreia casos diários de COVID-19 nos Estados Unidos, usando dados fictícios. Aprenda como:

- Use o AWS Chalice para definir rotas em funções do Lambda que são chamadas para lidar com solicitações REST que vêm por meio do API Gateway.
- Usar as funções do Lambda para recuperar e armazenar dados em uma tabela do DynamoDB para atender a solicitações REST.
- Defina a estrutura da tabela e os recursos da função de segurança em um AWS CloudFormation modelo.

- Use o AWS Chalice e CloudFormation empacote e implante todos os recursos necessários.
- Use CloudFormation para limpar todos os recursos criados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice para criar uma API REST apoiada por um banco de dados Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Use AWS Secrets Manager para gerenciar as credenciais do banco de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST sem servidor no Amazon API Gateway e. AWS Lambda
- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Criar uma aplicação de mensageiro

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.
- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SQS
- Funções de etapa

Criar uma aplicação de chat websocket

O exemplo de código a seguir mostra como criar uma aplicação de chat que é atendido por uma API de Websocket criada no Amazon API Gateway.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon API Gateway V2 para criar uma API de websocket que se integre com AWS Lambda o Amazon DynamoDB.

- Crie uma API de Websocket atendida pelo API Gateway.
- Defina um manipulador do Lambda que armazena conexões no DynamoDB e publica mensagens para outros participantes do chat.
- Conecte-se à aplicação de chat websocket e envie mensagens com o pacote Websockets.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para Python (Boto3)

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.

- Use o pacote Requests para enviar solicitações à API REST.
- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento agendado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função do Lambda o alvo.
- Concede permissão para permitir que a função Lambda seja EventBridge invocada.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações agendadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB

- EventBridge
- Lambda
- Amazon SNS

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Python.

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
```

```
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import base64
def lambda_handler(event, context):

    for record in event['Records']:
        try:
            print(f"Processed Kinesis Event - EventID: {record['eventID']}")
            record_data = base64.b64decode(record['kinesis']
['data']).decode('utf-8')
            print(f"Record Data: {record_data}")
            # TODO: Do interesting work based on the new data
        except Exception as e:
            print(f"An error occurred {e}")
            raise e
    print(f"Successfully processed {len(event['Records'])} records.")
```

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Python.

```
import json

def lambda_handler(event, context):
    print(json.dumps(event, indent=2))
```

```
    for record in event['Records']:
        log_dynamodb_record(record)

def log_dynamodb_record(record):
    print(record['eventID'])
    print(record['eventName'])
    print(f"DynamoDB Record: {json.dumps(record['dynamodb'])}")
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Python.

```
import json

def lambda_handler(event, context):
    for record in event.get('events', []):
        log_document_db_event(record)
    return 'OK'

def log_document_db_event(record):
    event_data = record.get('event', {})
    operation_type = event_data.get('operationType', 'Unknown')
    db = event_data.get('ns', {}).get('db', 'Unknown')
    collection = event_data.get('ns', {}).get('coll', 'Unknown')
    full_document = event_data.get('fullDocument', {})

    print(f"Operation type: {operation_type}")
```

```
    print(f"db: {db}")
    print(f"collection: {collection}")
    print("Full document:", json.dumps(full_document, indent=2))
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Python.

```
import base64

def lambda_handler(event, context):
    # Iterate through keys
    for key in event['records']:
        print('Key:', key)
        # Iterate through records
        for record in event['records'][key]:
            print('Record:', record)
            # Decode base64
            msg = base64.b64decode(record['value']).decode('utf-8')
            print('Message:', msg)
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your
 bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do Kinesis com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["kinesis"]["sequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""
```

```
    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["dynamodb"]["SequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

def lambda_handler(event, context):
    if event:
        batch_item_failures = []
        sqs_batch_response = {}

        for record in event["Records"]:
            try:
                print(f"Processed message: {record['body']}")
            except Exception as e:
```

```
                batch_item_failures.append({"itemIdentifier": record['messageId']})

        sqs_batch_response["batchItemFailures"] = batch_item_failures
        return sqs_batch_response
```

Exemplos de Serviço gerenciado para Apache Flink usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o serviço gerenciado AWS SDK para Python (Boto3) com o Apache Flink.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Gerador de dados

Ações

`AddApplicationInput`

O código de exemplo a seguir mostra como usar `AddApplicationInput`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
```

```
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def add_input(self, input_prefix, stream_arn, input_schema):
        """
        Adds an input stream to the application. The input stream data is mapped
        to an in-application stream that can be processed by your code running in
        Kinesis Data Analytics.

        :param input_prefix: The prefix prepended to in-application input stream
names.
        :param stream_arn: The ARN of the input stream.
        :param input_schema: A schema that maps the data in the input stream to the
                             runtime environment. This can be automatically
generated
                             by using `discover_input_schema` or you can create it
                             yourself.
        :return: Metadata about the newly added input.
        """
        try:
            response = self.analytics_client.add_application_input(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Input={
                    "NamePrefix": input_prefix,
                    "KinesisStreamsInput": {"ResourceARN": stream_arn},
                    "InputSchema": input_schema,
                },
            )
            self.version_id = response["ApplicationVersionId"]
            logger.info("Add input stream %s to application %s.", stream_arn,
self.name)
        except ClientError:
            logger.exception(
```

```
                "Couldn't add input stream %s to application %s.", stream_arn,
 self.name
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a AddApplicationInput Referência da API AWS SDK for Python (Boto3).

`AddApplicationOutput`

O código de exemplo a seguir mostra como usar `AddApplicationOutput`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def add_output(self, in_app_stream_name, output_arn):
        """
        Adds an output stream to the application. Kinesis Data Analytics maps data
```

```
        from the specified in-application stream to the output stream.

        :param in_app_stream_name: The name of the in-application stream to map
                                   to the output stream.
        :param output_arn: The ARN of the output stream.
        :return: A list of metadata about the output resources currently assigned
                 to the application.
        """
        try:
            response = self.analytics_client.add_application_output(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Output={
                    "Name": in_app_stream_name,
                    "KinesisStreamsOutput": {"ResourceARN": output_arn},
                    "DestinationSchema": {"RecordFormatType": "JSON"},
                },
            )
            outputs = response["OutputDescriptions"]
            self.version_id = response["ApplicationVersionId"]
            logging.info(
                "Added output %s to %s, which now has %s outputs.",
                output_arn,
                self.name,
                len(outputs),
            )
        except ClientError:
            logger.exception("Couldn't add output %s to %s.", output_arn, self.name)
            raise
        else:
            return outputs
```

- Para obter detalhes da API, consulte a AddApplicationOutput Referência da API AWS SDK for Python (Boto3).

`CreateApplication`

O código de exemplo a seguir mostra como usar `CreateApplication`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def create(self, app_name, role_arn, env="SQL-1_0"):
        """
        Creates a Kinesis Data Analytics application.

        :param app_name: The name of the application.
        :param role_arn: The ARN of a role that can be assumed by Kinesis Data
                         Analytics and grants needed permissions.
        :param env: The runtime environment of the application, such as SQL. Code
                    uploaded to the application runs in this environment.
        :return: Metadata about the newly created application.
        """
        try:
            response = self.analytics_client.create_application(
                ApplicationName=app_name,
                RuntimeEnvironment=env,
                ServiceExecutionRole=role_arn,
            )
            details = response["ApplicationDetail"]
            self._update_details(details)
            logger.info("Application %s created.", app_name)
```

```
        except ClientError:
            logger.exception("Couldn't create application %s.", app_name)
            raise
        else:
            return details
```

- Para obter detalhes da API, consulte a CreateApplication Referência da API AWS SDK for Python (Boto3).

`DeleteApplication`

O código de exemplo a seguir mostra como usar `DeleteApplication`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def delete(self):
        """
        Deletes an application.
        """
```

```
        try:
            self.analytics_client.delete_application(
                ApplicationName=self.name, CreateTimestamp=self.create_timestamp
            )
            logger.info("Deleted application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete application %s.", self.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteApplication Referência da API AWS SDK for Python (Boto3).

`DescribeApplication`

O código de exemplo a seguir mostra como usar `DescribeApplication`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def describe(self, name):
```

```
        """
        Gets metadata about an application.

        :param name: The name of the application to look up.
        :return: Metadata about the application.
        """
        try:
            response =
 self.analytics_client.describe_application(ApplicationName=name)
            details = response["ApplicationDetail"]
            self._update_details(details)
            logger.info("Got metadata for application %s.", name)
        except ClientError:
            logger.exception("Couldn't get metadata for application %s.", name)
            raise
        else:
            return details
```

- Para obter detalhes da API, consulte a DescribeApplication Referência da API AWS SDK for Python (Boto3).

`DescribeApplicationSnapshot`

O código de exemplo a seguir mostra como usar `DescribeApplicationSnapshot`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
```

```
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def describe_snapshot(self, application_name, snapshot_name):
        """
        Gets metadata about a previously saved application snapshot.

        :param application_name: The name of the application.
        :param snapshot_name: The name of the snapshot.
        :return: Metadata about the snapshot.
        """
        try:
            response = self.analytics_client.describe_application_snapshot(
                ApplicationName=application_name, SnapshotName=snapshot_name
            )
            snapshot = response["SnapshotDetails"]
            logger.info(
                "Got metadata for snapshot %s of application %s.",
                snapshot_name,
                application_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get metadata for snapshot %s of application %s.",
                snapshot_name,
                application_name,
            )
            raise
        else:
            return snapshot
```

- Para obter detalhes da API, consulte a DescribeApplicationSnapshot Referência da API AWS SDK for Python (Boto3).

`DiscoverInputSchema`

O código de exemplo a seguir mostra como usar `DiscoverInputSchema`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def discover_input_schema(self, stream_arn, role_arn):
        """
        Discovers a schema that maps data in a stream to a format that is usable by
        an application's runtime environment. The stream must be active and have
        enough data moving through it for the service to sample. The returned schema
        can be used when you add the stream as an input to the application or you
 can
        write your own schema.

        :param stream_arn: The ARN of the stream to map.
        :param role_arn: A role that lets Kinesis Data Analytics read from the
 stream.
        :return: The discovered schema of the data in the input stream.
        """
        try:
            response = self.analytics_client.discover_input_schema(
                ResourceARN=stream_arn,
```

```
                ServiceExecutionRole=role_arn,
                InputStartingPositionConfiguration={"InputStartingPosition": "NOW"},
            )
            schema = response["InputSchema"]
            logger.info("Discovered input schema for stream %s.", stream_arn)
        except ClientError:
            logger.exception(
                "Couldn't discover input schema for stream %s.", stream_arn
            )
            raise
        else:
            return schema
```

- Para obter detalhes da API, consulte a DiscoverInputSchema Referência da API AWS SDK for Python (Boto3).

`StartApplication`

O código de exemplo a seguir mostra como usar `StartApplication`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
```

```
        self.create_timestamp = None


    def start(self, input_id):
        """
        Starts an application. After the application is running, it reads from the
        specified input stream and runs the application code on the incoming data.

        :param input_id: The ID of the input to read.
        """
        try:
            self.analytics_client.start_application(
                ApplicationName=self.name,
                RunConfiguration={
                    "SqlRunConfigurations": [
                        {
                            "InputId": input_id,
                            "InputStartingPositionConfiguration": {
                                "InputStartingPosition": "NOW"
                            },
                        }
                    ]
                },
            )
            logger.info("Started application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't start application %s.", self.name)
            raise
```

- Para obter detalhes da API, consulte a StartApplication Referência da API AWS SDK for Python (Boto3).

StopApplication

O código de exemplo a seguir mostra como usar `StopApplication`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def stop(self):
        """
        Stops an application. This stops the application from processing data but
        does not delete any resources.
        """
        try:
            self.analytics_client.stop_application(ApplicationName=self.name)
            logger.info("Stopping application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't stop application %s.", self.name)
            raise
```

- Para obter detalhes da API, consulte a StopApplication Referência da API AWS SDK for Python (Boto3).

`UpdateApplication`

O código de exemplo a seguir mostra como usar `UpdateApplication`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo atualiza o código que é executado em um aplicativo existente.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def update_code(self, code):
        """
        Updates the code that runs in the application. The code must run in the
        runtime environment of the application, such as SQL. Application code
        typically reads data from in-application streams and transforms it in some
 way.

        :param code: The code to upload. This completely replaces any existing code
                     in the application.
        :return: Metadata about the application.
        """
        try:
            response = self.analytics_client.update_application(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
```

```
                ApplicationConfigurationUpdate={
                    "ApplicationCodeConfigurationUpdate": {
                        "CodeContentTypeUpdate": "PLAINTEXT",
                        "CodeContentUpdate": {"TextContentUpdate": code},
                    }
                },
            )
            details = response["ApplicationDetail"]
            self.version_id = details["ApplicationVersionId"]
            logger.info("Update code for application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't update code for application %s.", self.name)
            raise
        else:
            return details
```

- Para obter detalhes da API, consulte a UpdateApplication Referência da API AWS SDK for Python (Boto3).

Gerador de dados

Gerar um stream com um referenciador

O exemplo de código a seguir mostra como gerar um stream do Kinesis com um referenciador.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"
```

```
def get_data():
    return {"REFERRER": "http://www.amazon.com"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gerar um stream com anomalias da pressão arterial

O exemplo de código a seguir mostra como gerar um stream do Kinesis com anomalias de pressão arterial.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"
```

```
class PressureType(Enum):
    low = "LOW"
    normal = "NORMAL"
    high = "HIGH"


def get_blood_pressure(pressure_type):
    pressure = {"BloodPressureLevel": pressure_type.value}
    if pressure_type == PressureType.low:
        pressure["Systolic"] = random.randint(50, 80)
        pressure["Diastolic"] = random.randint(30, 50)
    elif pressure_type == PressureType.normal:
        pressure["Systolic"] = random.randint(90, 120)
        pressure["Diastolic"] = random.randint(60, 80)
    elif pressure_type == PressureType.high:
        pressure["Systolic"] = random.randint(130, 200)
        pressure["Diastolic"] = random.randint(90, 150)
    else:
        raise TypeError
    return pressure


def generate(stream_name, kinesis_client):
    while True:
        rnd = random.random()
        pressure_type = (
            PressureType.low
            if rnd < 0.005
            else PressureType.high
            if rnd > 0.995
            else PressureType.normal
        )
        blood_pressure = get_blood_pressure(pressure_type)
        print(blood_pressure)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(blood_pressure),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gerar um stream com dados em colunas

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dados em colunas.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"Col_A": "a", "Col_B": "b", "Col_C": "c", "Col_E_Unstructured": "x,y,z"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gerar um stream com anomalias da frequência cardíaca

O exemplo de código a seguir mostra como gerar um stream do Kinesis com anomalias da frequência cardíaca.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


class RateType(Enum):
    normal = "NORMAL"
    high = "HIGH"


def get_heart_rate(rate_type):
    if rate_type == RateType.normal:
        rate = random.randint(60, 100)
    elif rate_type == RateType.high:
        rate = random.randint(150, 200)
    else:
        raise TypeError
    return {"heartRate": rate, "rateType": rate_type.value}


def generate(stream_name, kinesis_client, output=True):
    while True:
        rnd = random.random()
        rate_type = RateType.high if rnd < 0.01 else RateType.normal
        heart_rate = get_heart_rate(rate_type)
```

```
        if output:
            print(heart_rate)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(heart_rate),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gerar um stream com hotspots

O exemplo de código a seguir mostra como gerar um stream do Kinesis com hotspots.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
from pprint import pprint
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_hotspot(field, spot_size):
    hotspot = {
        "left": field["left"] + random.random() * (field["width"] - spot_size),
        "width": spot_size,
        "top": field["top"] + random.random() * (field["height"] - spot_size),
        "height": spot_size,
```

```
    }
    return hotspot


def get_record(field, hotspot, hotspot_weight):
    rectangle = hotspot if random.random() < hotspot_weight else field
    point = {
        "x": rectangle["left"] + random.random() * rectangle["width"],
        "y": rectangle["top"] + random.random() * rectangle["height"],
        "is_hot": "Y" if rectangle is hotspot else "N",
    }
    return {"Data": json.dumps(point), "PartitionKey": "partition_key"}


def generate(
    stream_name, field, hotspot_size, hotspot_weight, batch_size, kinesis_client
):
    """
    Generates points used as input to a hotspot detection algorithm.
    With probability hotspot_weight (20%), a point is drawn from the hotspot;
    otherwise, it is drawn from the base field. The location of the hotspot
    changes for every 1000 points generated.
    """
    points_generated = 0
    hotspot = None
    while True:
        if points_generated % 1000 == 0:
            hotspot = get_hotspot(field, hotspot_size)
        records = [
            get_record(field, hotspot, hotspot_weight) for _ in range(batch_size)
        ]
        points_generated += len(records)
        pprint(records)
        kinesis_client.put_records(StreamName=stream_name, Records=records)

        time.sleep(0.1)


if __name__ == "__main__":
    generate(
        stream_name=STREAM_NAME,
        field={"left": 0, "width": 10, "top": 0, "height": 10},
        hotspot_size=1,
        hotspot_weight=0.2,
```

```
        batch_size=10,
        kinesis_client=boto3.client("kinesis"),
    )
```

Gerar um stream com entradas de log

O exemplo de código a seguir mostra como gerar um stream do Kinesis com entradas de log.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "
        '"GET /index.php HTTP/1.1" 200 125 "-" '
        '"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0"'
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )
```

```
if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gerar um stream com dados escalonados

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dados escalonados.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import datetime
import json
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    event_time = datetime.datetime.utcnow() - datetime.timedelta(seconds=10)
    return {
        "EVENT_TIME": event_time.isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        # Send six records, ten seconds apart, with the same event time and ticker
        for _ in range(6):
            print(data)
```

```
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey",
            )
            time.sleep(10)


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gerar um stream com dados de cotação da bolsa

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dados de cotação da bolsa.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }
```

```
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gerar um stream com dois tipos de dados

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dois tipos de dados.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import random
import boto3

STREAM_NAME = "OrdersAndTradesStream"
PARTITION_KEY = "partition_key"


def get_order(order_id, ticker):
    return {
        "RecordType": "Order",
        "Oid": order_id,
        "Oticker": ticker,
```

```
        "Oprice": random.randint(500, 10000),
        "Otype": "Sell",
    }


def get_trade(order_id, trade_id, ticker):
    return {
        "RecordType": "Trade",
        "Tid": trade_id,
        "Toid": order_id,
        "Tticker": ticker,
        "Tprice": random.randint(0, 3000),
    }


def generate(stream_name, kinesis_client):
    order_id = 1
    while True:
        ticker = random.choice(["AAAA", "BBBB", "CCCC"])
        order = get_order(order_id, ticker)
        print(order)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(order),
 PartitionKey=PARTITION_KEY
        )
        for trade_id in range(1, random.randint(0, 6)):
            trade = get_trade(order_id, trade_id, ticker)
            print(trade)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(trade),
                PartitionKey=PARTITION_KEY,
            )
        order_id += 1


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Gerar um stream com dados de log da web

O exemplo de código a seguir mostra como gerar um stream do Kinesis com dados de log da web.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "log": "192.168.254.30 - John [24/May/2004:22:01:02 -0700] "
        '"GET /icons/apache_pb.gif HTTP/1.1" 304 0'
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

AWS Marketplace Exemplos de API de catálogo usando SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS SDK para Python (Boto3) with AWS Marketplace Catalog.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Produtos de AMI
- Ofertas de parceiros de canal
- Produtos de contêiner
- Entidades
- Ofertas
- Produtos
- Autorização de revenda
- Produtos de SaaS do
- Serviços públicos

Produtos de AMI

Adicionar uma dimensão a um produto existente da AMI e atualize os termos de preços da oferta

O exemplo de código a seguir mostra como adicionar uma dimensão a um produto de AMI existente e atualizar os termos de preços da oferta.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
```

```
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to add a dimension to an existing
 AMI product and update the offer pricing terms.
CAPI-23
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
```

```
    sc.usage_demo(change_set, "Add dimension for AMI product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Adicionar uma região onde um produto de AMI é implantado

O exemplo de código a seguir mostra como adicionar uma região em que um produto de AMI é implantado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
```

```
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to add a region where my
AMI product is deployed
CAPI-25A
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Add a region where my AMI product is deployed",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar um produto de AMI público ou limitado e uma oferta pública com preços anuais calculados por hora

O exemplo de código a seguir mostra como criar um produto de AMI público ou limitado e uma oferta pública com preços anuais calculados por hora. Este exemplo cria um EULA padrão ou personalizado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
```

```
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [],
                "AdditionalResources": [],
                "SupportDescription": "For support, contact support@example.com"
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
```

```
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
```

```
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
```

```
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public or limited AMI
product and public offer with hourly-annual pricing and standard or custom EULA
CAPI-06
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
```

```
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited AMI product and public offer with hourly-annual pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar um produto de AMI público ou limitado e uma oferta pública com preços mensais calculados por hora

O exemplo de código a seguir mostra como criar um produto de AMI público ou limitado e uma oferta pública com preços mensais calculados por hora. Este exemplo cria um EULA padrão ou personalizado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
```

```
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
```

```
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
```

```
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "RecurringPaymentTerm",
                        "CurrencyCode": "USD",
                        "BillingPeriod": "Monthly",
                        "Price": "15.0"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public or limited AMI
product and public offer with hourly-monthly pricing and standard or custom EULA
CAPI-08
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
```

```
        change_set,
        "create limited AMI product and public offer with hourly-monthly pricing and
 standard EULA",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar um produto de AMI público ou limitado e uma oferta pública com preços calculados por hora

O exemplo de código a seguir mostra como criar um produto de AMI público ou limitado e uma oferta pública com preços por hora. Este exemplo cria um EULA padrão ou personalizado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
```

```
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
```

```
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly pricing for AmiProduct
using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to Create a public or limited AMI
 product
and public offer with hourly pricing and standard or custom EULA
CAPI-07
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited AMI product and public offer with hourly pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
```

```
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar o rascunho de um produto de AMI com um rascunho de oferta pública

O exemplo de código a seguir mostra como criar um rascunho de produto de AMI com um rascunho de oferta pública.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
```

```
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an AMI draft product
with a draft public offer.
CAPI-02
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "AMI draft product with draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Restringir uma região onde um produto de AMI é implantado

O exemplo de código a seguir mostra como restringir uma região em que um produto de AMI é implantado.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to restrict a region where my
AMI product is deployed
CAPI-25B
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Add a region where my AMI product is deployed",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Limitar a visibilidade do produto

O exemplo de código a seguir mostra como restringir a visibilidade do produto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to change a product visibility to
 restricted
CAPI-17
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Restrict existing AMI")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Especificar se os ativos da AMI são implantados em novas regiões

O exemplo de código a seguir mostra como especificar se os ativos da AMI serão implantados em novas regiões criadas AWS para oferecer suporte a futuras regiões.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
```

```
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to modify a product to support all
 future regions
CAPI-26
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update future region support")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Ofertas de parceiros de canal

Criar um rascunho de CPPO para qualquer tipo de produto

O exemplo de código a seguir mostra como criar um rascunho de CPPO para qualquer tipo de produto a fim de analisá-lo internamente antes de publicar para os compradores.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer name"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” CPPO
for any product type (AMI/SaaS/Container) that can be reviewed internally
```

```
before publishing to buyers
CAPI-60
"""
import os

import utils.start_changeset as sc  # noqa: E402
import utils.stringify_details as sd  # noqa: E402

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Create a draft CPPO offer for a product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada de substituição de autorização de revenda com preços contratuais

O exemplo de código a seguir mostra como criar uma oferta privada de substituição de autorização de revenda com base em um contrato existente com preços contratuais.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private resale replacement offer with contract
 pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a resale authorization
 replacement private offer
from an existing agreement with contract pricing
CAPI-96
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create resale authorization replacement private offer with contract
 pricing",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Listar todas as CPPOs criadas por um parceiro de canal

O exemplo de código a seguir mostra como listar todos os CPPOs criados por um parceiro de canal.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to list all Channel Partner Offers
in an account

Program executed with no arguments:
ie. python3 list_all_cppo_offers.py

CAPI-93
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-catalog")


def get_offer_entities():
    """
    Returns a list of all offers in the account
    """

    next_token = ""  # nosec: B105
    response_list = []

    try:
        response = mp_client.list_entities(Catalog="AWSMarketplace",
 EntityType="Offer")
```

```
    except ClientError as e:
        logging.exception(f"Couldn't list entities. {e}")
        raise

    response_list.append(response)

    # Results are paginated depending on number of entities returned
    while "NextToken" in response:
        next_token = response["NextToken"]

        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType="Offer",
                NextToken=next_token,
            )
        except ClientError as e:
            logging.exception(f"Couldn't list entities. {e}")
            raise

        if "NextToken" in response:
            response_list.append(response)

    return response_list


def build_offer_list(response_list):
    """
    Cleans up list_entities response list with just list of offer IDs
    """
    offer_list = []

    for response in response_list:
        for entity in response["EntitySummaryList"]:
            offer_list.append(entity["EntityId"])

    return offer_list


def check_offer_resaleauth(offer_id):
    """
    Checks to see if an offer is based on a resale authorization
    """
    offer_response = describe_entity(offer_id)
```

```
    offer_details = json.loads(offer_response["Details"])
    if offer_details is None:
        offer_details = offer_response["DetailsDocument"]
    if "ResaleAuthorizationId" in offer_details and
 offer_details["ResaleAuthorizationId"] is not None:
        return offer_id
    else:
        return None


def describe_entity(entity_id):
    """
    General purpose describe entity call
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )
    except ClientError as e:
        logging.exception(f"Couldn't describe entity. {e}")
        raise

    return response


def get_resaleauth_offers():
    """
    Returns a list of all offers in the account that are
    based on a resale authorization
    """
    resale_offer_list = []

    response_list = get_offer_entities()
    offer_list = build_offer_list(response_list)
    for offer in offer_list:
        print ("offer id " + offer)
        offer_info = check_offer_resaleauth(offer)
        if offer_info is not None:
            resale_offer_list.append(offer_info)

    return resale_offer_list
```

```
if __name__ == "__main__":
    print(get_resaleauth_offers())
```

- Para obter detalhes da API, consulte a ListEntities Referência da API AWS SDK for Python (Boto3).

Listar todas as autorizações de revenda compartilhadas disponíveis para um parceiro de canal

O exemplo de código a seguir mostra como listar todas as autorizações de revenda compartilhadas disponíveis para um parceiro de canal.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to list all resale authorizations
shared to an account

Program executed with no arguments:
ie. python3 list_all_resale_authorizations.py

CAPI-94
"""

import logging

import boto3
import utils.helpers as hlp  # noqa: E402
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-catalog")
```

```
def get_shared_entities():
    next_token = ""  # nosec: B105
    response_list = []

    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType="ResaleAuthorization",
            OwnershipType="SHARED",
        )
    except ClientError as e:
        logging.exception(f"Couldn't list entities. {e}")
        raise

    response_list.append(response)

    # Results can be paginated depending on number of entities returned
    while "NextToken" in response:
        next_token = response["NextToken"]

        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType="ResaleAuthorization",
                OwnershipType="SHARED",
                NextToken=next_token,
            )
        except ClientError as e:
            logging.exception(f"Couldn't list entities. {e}")
            raise

        if "NextToken" in response:
            response_list.append(response)

    return response_list


if __name__ == "__main__":
    response_list = get_shared_entities()
    hlp.pretty_print_datetime(response_list)
```

- Para obter detalhes da API, consulte a ListEntities Referência da API AWS SDK for Python (Boto3).

Publicar uma CPPO e anexar um EULA do comprador

O exemplo de código a seguir mostra como publicar uma CPPO e acrescentar um EULA do comprador.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
```

```
                "Type": "LegalTerm",
                "Documents": [
                 {
                  "Type": "CustomEula",
                  "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                 }
                ]
            }
        ]
          }
         },
         {
             "ChangeType": "UpdateTargeting",
             "Entity": {
                 "Type": "Offer@1.0",
                 "Identifier": "$CreateCPPOoffer.Entity.Identifier"
             },
             "DetailsDocument": {
                 "PositiveTargeting": {
                     "BuyerAccounts": ["222222222222"]
                     }
             }
         },
         {
             "ChangeType": "UpdateAvailability",
             "Entity": {
                 "Type": "Offer@1.0",
                 "Identifier": "$CreateCPPOoffer.Entity.Identifier"
             },
             "DetailsDocument": {
                 "AvailabilityEndDate": "2023-07-31"
                 }
         },
         {
             "ChangeType": "UpdateValidityTerms",
             "Entity": {
                 "Type": "Offer@1.0",
                 "Identifier": "$CreateCPPOoffer.Entity.Identifier"
             },
             "DetailsDocument": {
                 "Terms": [
                     {
                         "Type": "ValidityTerm",
                         "AgreementDuration": "P450D"
```

```
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to publish CPPO
for any product type (AMI/SaaS/Container) and append buyer EULA
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
```

```
        stringified_change_set,
        "Publish CPPO for any product type and append buyer EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar uma CPPO usando autorização única de revenda e atualizar a margem de preço

O exemplo de código a seguir mostra como publicar uma CPPO usando autorização única de revenda em produtos de AMI, de SaaS ou de contêiner e atualizar a margem de preço.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPO",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111"
            }
        },
```

```
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test Offer name",
                "Description":"Test Offer description"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["111111111111"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
                }
        },
        {
```

```
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to Create CPPO using one-time resale
authorization on AMI, SaaS or Container products and update price markup
CAPI-63
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create CPPO using one-time resale authorization and update price markup"
    )
```

```
    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar um rascunho de CPPO e atualizar a margem de preço

O exemplo de código a seguir mostra como publicar um rascunho de CPPO e atualizar a margem de preço.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPO",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test Offer name",
                "Description":"Test Offer description"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["111111111111"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to publish “draft” CPPO
for any product type (AMI/SaaS/Container) and update price markup
CAPI-72
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
```

```
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Publish draft CPPO for any product type adn update price markup"
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar a data de validade de uma CPPO

O exemplo de código a seguir mostra como atualizar a data de validade de uma CPPO para dar aos compradores mais tempo para avaliar e aceitar a oferta.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update the expiry
date of a CPPO offer
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Update the expiry date of a CPPO offer"
    )

    return response
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Produtos de contêiner

Criar o rascunho de um produto de contêiner com um rascunho de oferta pública

O exemplo de código a seguir mostra como criar um rascunho de produto de contêiner com um rascunho de oferta pública.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
```

```
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an container draft product
with a draft public offer.
CAPI-03
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create a draft container product with a draft public offer",
    )

    return response
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar um produto de contêiner limitado com uma oferta pública e preços calculados por contrato

O exemplo de código a seguir mostra como criar um produto de contêiner limitado com uma oferta pública, preços contratuais e EULA padrão.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
```

```
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
                    "Name": "Cores",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
```

```
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
                "Description": "Some interesting container offer description"
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create limited container
product with public offer, contract pricing and standard EULA
CAPI-15
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
```

```
        "Create limited container product with public offer contract pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Entidades

Descrever todas as entidades em uma única chamada

O exemplo de código a seguir mostra como descrever todas as entidades em uma única chamada.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to describe for multiple entities
 information in the AWS Marketplace Catalog
CAPI-98
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

PRODUCT_ID = "prod-1111111111111"
OFFER_ID = "offer-1111111111111"
MARKETPLACE_CATALOG = "AWSMarketplace"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_entities_information(mp_client):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    entity_request_list_param = [
        {'EntityId': PRODUCT_ID, 'Catalog': MARKETPLACE_CATALOG},
        {'EntityId': OFFER_ID, 'Catalog': MARKETPLACE_CATALOG}
    ]
    try:
        response = mp_client.batch_describe_entities(
            EntityRequestList=entity_request_list_param
        )

        return response

    except ClientError as e:
        logger.exception("Unexpected error: %s", e)
        raise


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for entities in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")
```

```
    response = get_entities_information(mp_client)
    print("Successful entities response -")
    pretty_print(response["EntityDetails"])
    print("Failed entities response -")
    pretty_print(response["Errors"])


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a BatchDescribeEntities Referência da API AWS SDK for Python (Boto3).

Listar e descrever todas as ofertas associadas a um produto

O exemplo de código a seguir mostra como listar e descrever todas as ofertas associadas a um produto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to retrieve all offer information
related to a single product
CAPI-97
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError
```

```
from utils import helpers

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-catalog")


def get_entity_information(entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def list_entity_details(entity_type, entity_id):
    """
    Returns details about a given entity and entity type
    """

    entity_summary_list = []

    # filter will return details for given entity_id with BuyerAccounts targeting
    filter_list_param = {
        'OfferFilters':{
            'ProductId':{
                'ValueList':[entity_id]
            },
            'Targeting': {
                'ValueList': ["BuyerAccounts"]
```

```
            }
        }
    }

    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType=entity_type,
            EntityTypeFilters = filter_list_param,
            MaxResults=10
        )

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)

    # add results to entity_summary_list
    entity_summary_list.extend(response["EntitySummaryList"])

    # if there are more than 10 offers, paginate through the results
    while "NextToken" in response and response["NextToken"] is not None:
        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType=entity_type,
                EntityTypeFilters = filter_list_param,
                NextToken=response["NextToken"],
                MaxResults=10
            )

        except ClientError as e:
            if e.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Entity ID %s not found.", entity_id)
            else:
                logger.error("Unexpected error: %s", e)

        # add results to entity_summary_list
        entity_summary_list.extend(response["EntitySummaryList"])

        return entity_summary_list

if __name__ == "__main__":
```

```
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "--entity-id",
        "-eid",
        help="Provide Entity ID corresponding to a product to filter offers on",
        required=True,
    )

    args = parser.parse_args()

    # Gets a offers associated with the entity_id
    response = list_entity_details(
        "Offer",
        entity_id=args.entity_id
    )

    if response: # if response is not empty

        # list_entity_details returns a list of offers
        for offer in response:

            print("-"*128)
            print(f"Terms for Offer ID: {offer['EntityId']}")
            print("-"*128)

            #retrieve offer information for each offer
            entity_information = get_entity_information(offer["EntityId"])

            helpers.pretty_print_datetime(entity_information)

    else:
        print(f"No information found for Entity ID: {args.entity_id}")
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DescribeEntity
  - ListEntities

Ofertas

Criar uma dimensão personalizada para um produto de SaaS e criar uma oferta privada

O exemplo de código a seguir mostra como criar uma dimensão personalizada para um produto de SaaS e criar uma oferta privada.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
 defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
```

```
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
```

```
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
```

```
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a SaaS product custom
 dimension and private offer
CAPI-91
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
```

```
    sc.usage_demo(
        change_set, "Create a SaaS product custom dimension and private offer"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar um rascunho de oferta privada para um produto de AMI ou SaaS

O exemplo de código a seguir mostra como criar um rascunho de oferta privada para um produto de AMI ou de SaaS a fim de analisá-lo internamente antes de publicá-lo para os compradores.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” Private Offer
for any AMI or SAAS product type that can be reviewed internally
before publishing to buyers
CAPI-30
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Private offer for AMI product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Crie uma oferta privada com contrato e Pay-As-You-Go preços para um produto SaaS

O exemplo de código a seguir mostra como criar uma oferta privada com contrato e Pay-As-You-Go preços para um produto SaaS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
```

```
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 contract with PAYG pricing for my SaaS product
CAPI-34
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
```

```
        change_set,
        "Create private offer with contract with PAYG pricing for my SaaS product",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada com preços contratuais e uma programação de pagamento flexível para um produto de SaaS

O exemplo de código a seguir mostra como criar uma oferta privada com preços contratuais e uma programação de pagamento flexível para um produto de SaaS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
```

```
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Some kind of refund policy description"
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer
with contract pricing and flexible payment schedule for my SaaS product
CAPI-39
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)
```

```
    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create private offer with contract pricing and flexible payment schedule
 for my SaaS product",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada com preços contratuais para um produto de contêiner

O exemplo de código a seguir mostra como criar uma oferta privada com preços contratuais para um produto de contêiner.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
```

```
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
 Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
 contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer (target
 buyers)
for my Container product with contract pricing
CAPI-36
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
```

```
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "publish a private offer for my Container product with contract pricing",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada com preços contratuais para um produto de AMI

O exemplo de código a seguir mostra como criar uma oferta privada com preços contratuais para um produto de AMI.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
```

```
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 contract pricing for my AMI product
CAPI-35
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)
```

```
change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with contract pricing for my AMI product"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada com preços anuais calculados por hora e uma programação de pagamento flexível para um produto de AMI

O exemplo de código a seguir mostra como criar uma oferta privada com preços anuais calculados por hora e uma programação de pagamento flexível para um produto de AMI.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
```

```
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "FixedUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P365D",
                        "Grants": [
                            {
                                "DimensionKey": "t2.micro",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
```

```
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly annual pricing and flexible payment schedule for my AMI product
CAPI-XX
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(
        change_set,
        "Create private offer with hourly annual pricing and flexible payment
 schedule for my AMI product",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada com preços anuais calculados por hora para um produto de AMI

O exemplo de código a seguir mostra como criar uma oferta privada com preços anuais calculados por hora para um produto de AMI.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
```

```
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
```

```
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-12-31"
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Usage",
            "Terms": [
                {
                    "Type": "UsageBasedPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "0.17"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
```

```
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly annual pricing for my AMI product
CAPI-31
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with hourly annual pricing for my AMI
 product"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada com preços por hora para um produto de AMI

O exemplo de código a seguir mostra como criar uma oferta privada com preços calculados por hora para um produto de AMI.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-01-01"
            }
        },
        {
```

```
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly pricing for my AMI product
CAPI-32
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with hourly pricing for my AMI product"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada com preços de assinatura para um produto de SaaS

O exemplo de código a seguir mostra como criar uma oferta privada com preços de assinatura para um produto de SaaS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
```

```
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 subscription pricing for my SaaS product
CAPI-33
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "Create private offer with subscription pricing for my SaaS
 product"
    )


if __name__ == "__main__":
```

```
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada com preços contratuais por níveis para um produto de SaaS

O exemplo de código a seguir mostra como criar uma oferta privada com preços contratuais por níveis para um produto de SaaS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
```

```
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
```

```
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 tiered contract pricing for my SaaS product
CAPI-XX
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create private offer with tiered contract pricing for my SaaS product",
    )
```

```
if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta pública de teste gratuito com preços de assinatura para um produto de SaaS

O exemplo de código a seguir mostra como criar uma oferta pública de teste gratuito com preços de assinatura de um produto de SaaS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test public free trial offer for SaaSProduct using AWS
Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public free trial offer
 with subscription pricing for SaaS product
CAPI-13
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
```

```
        "Create public free trial offer with subscription pricing for SaaS product",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar uma oferta privada de substituição com preços contratuais

O exemplo de código a seguir mostra como criar uma oferta privada de substituição com base em um contrato existente com preços contratuais.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract pricing
for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a replacement private
 offer
from an existing agreement with contract pricing
CAPI-95
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
```

```
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create replacement private offer with contract pricing..",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Descrever uma oferta pública

O exemplo de código a seguir mostra como descrever uma oferta pública.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) search for offer information in the
 AWS Marketplace Catalog
CAPI-29
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

OFFER_ID = "offer-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_offer_information(mp_client, entity_id):
    """
    Returns information about a given offer
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of offer information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Offer with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an offer in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_offer_information(mp_client, OFFER_ID))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeEntity Referência da API AWS SDK for Python (Boto3).

Expirar um rascunho de oferta privada

O exemplo de código a seguir mostra como definir a data de validade de uma oferta privada como uma data anterior para que os compradores não vejam mais a oferta.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
```

```
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” Private Offer
for any AMI or SAAS product type that can be reviewed internally
before publishing to buyers
CAPI-30
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Private offer for AMI product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Listar todas as ofertas privadas

O exemplo de código a seguir mostra como listar todas as ofertas privadas.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) for listing offers in the AWS
 Marketplace Catalog
CAPI-40
"""
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# Constants
MAX_RESULTS = 10
CATALOG = "AWSMarketplace"
ENTITY_TYPE = "Offer"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def list_private_offers(mp_client, return_all_private_offers):
    """
```

```
    This method retrieves list of all Private Offers for this account.
    """
    entity_summary_list = []
    filter_list_param = {
        'OfferFilters': {
            'Targeting': {
                'ValueList': ["BuyerAccounts"]
            }
        }
    }
    try:
        response = mp_client.list_entities(
            Catalog=CATALOG,
            EntityType=ENTITY_TYPE,
            EntityTypeFilters=filter_list_param,
            MaxResults=MAX_RESULTS
        )
    except ClientError as e:
        logger.error("Could not complete list_entities request: %s", e)
        raise

    entity_summary_list.extend(response["EntitySummaryList"])
    logger.info("Number of results in first iteration: %d " %
len(entity_summary_list))

    # Get subsequent pages of results if previous response contained a NextToken
    while "NextToken" in response and return_all_private_offers:
        try:
            logger.info("Getting Next Token results: %s " % response["NextToken"])
            response = mp_client.list_entities(
                Catalog=CATALOG,
                EntityType=ENTITY_TYPE,
                EntityTypeFilters=filter_list_param,
                MaxResults=MAX_RESULTS,
                NextToken=response["NextToken"]
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request: %s", e)
            raise

        entity_summary_list.extend(response["EntitySummaryList"])
        logger.info(
            "Number of results in the current iteration: %d "
            % len(response["EntitySummaryList"])
```

```
        )

    return entity_summary_list


def get_offer_details(mp_client, offer):
    """
    Describe the details of the Offer.
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace", EntityId=offer["EntityId"]
        )

        return response
    except ClientError:
        logger.exception("Error: Couldn't get details of the Offer.")
        raise


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Demo  - List Private offers.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    # Get list of all Offers.
    private_offers = list_private_offers(mp_client, False)
    count = len(private_offers)

    logger.info("Number of Offers: %d " % count)
    offer_counter = 0
    # Display details of each Offer.
    for offer in private_offers:
        print("-" * 88)
        offer_counter += 1
        print("Displaying Offer details for Offer# %d" % offer_counter)
        entity = get_offer_details(mp_client, offer)
        pretty_print(entity)

    print("-" * 88)
```

```
if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar uma oferta para aplicar um contrato com Pay-As-You-Go preços

O exemplo de código a seguir mostra como atualizar uma oferta para aplicar um contrato com Pay-As-You-Go preços.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply contract
 with PAYG pricing
CAPI-21
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer to apply contract with PAYG pricing")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar uma oferta para aplicar preços anuais calculados por hora

O exemplo de código a seguir mostra como atualizar uma oferta para aplicar preços anuais calculados por hora.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply hourly
 annual pricing
CAPI-20
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "Update offer with hourly annual pricing")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar uma oferta para aplicar a segmentação a regiões geográficas específicas

O exemplo de código a seguir mostra como atualizar uma oferta para aplicar segmentação a regiões geográficas específicas.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
```

```
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply
 targeting to certain geographic regions.
CAPI-19
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer targeting")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar nome e descrição de uma oferta pública

O exemplo de código a seguir mostra como atualizar o nome e a descrição de uma oferta pública.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update name and description of my
 offer
CAPI-18
"""
```

```
import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update name and description of my offer")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar o EULA de uma oferta

O exemplo de código a seguir mostra como atualizar o EULA de uma oferta.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update EULA of my offer
CAPI-18
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "Update EULA of my offer")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar a data de validade de uma oferta privada para uma data futura

O exemplo de código a seguir mostra como atualizar a data de validade de uma oferta privada para uma data futura a fim de dar aos compradores mais tempo para avaliar e aceitar a oferta.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to set expiry date of a private
 offer to a date in the future so that my buyers get more time to evaluate and
 accept the offer.
CAPI-37
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer expiration date")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar a duração do teste gratuito de uma oferta pública de teste gratuito de um produto de SaaS

O exemplo de código a seguir mostra como atualizar a duração do teste gratuito de uma oferta pública de teste gratuito de um produto de SaaS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to change free trial duration for my
 SaaS product by modifying my free trial public offer
CAPI-14
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Change free trial duration for SaaS product")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar a política de reembolso de uma oferta

O exemplo de código a seguir mostra como atualizar a política de reembolso de uma oferta.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update refund policy of my offer
CAPI-18
"""

import os
```

```
import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update refund policy of my public offer")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Produtos

Descrever um produto de AMI, SaaS ou contêiner

O exemplo de código a seguir mostra como descrever um produto de AMI, de SaaS ou de contêiner e verificar se ele contém todas as informações que você deseja saber sobre o produto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for product information in
 the AWS Marketplace Catalog
```

```
CAPI-28
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

PRODUCT_ID = "prod-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_product_information(mp_client, entity_id):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Product with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    print("-" * 88)
    print("Looking for a product in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_product_information(mp_client, PRODUCT_ID))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeEntity Referência da API AWS SDK for Python (Boto3).

Listar todos os produtos de AMI, SaaS ou Container e ofertas públicas associadas

O exemplo de código a seguir mostra como listar todos os produtos de AMI, de SaaS ou de contêiner e ofertas públicas associadas.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to display information about AMI
 products and their associated offers in the AWS Marketplace Catalog
CAPI-27
"""

import json
import logging

import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

try:
    mp_client = boto3.client("marketplace-catalog")
except ClientError as e:
    logger.error("Could not create boto3 client.")
    raise


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def describe_entity(entity_id):
    """
    Returns entity details
    Args: entity_id str: The entity ID of the product or offer
    Returns: dict: The entity details
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace", EntityId=entity_id
        )
    except ClientError as e:
        logger.error("Could not complete describe_entity request.")
        raise

    # De-stringify the details
    response["Details"] = json.loads(response["Details"])

    return response


def get_entities(entity_type, visibility=None):
    """
    Returns list of entities for provided entity_type
    Args: entity_type str: Type of entity list to return, in our case AmiProduct or
 Offer
    Returns: list: Abbreviated list of entity information
```

```
    """
    EntitySummaryList = []

    # Get the first page of results
    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType=entity_type,
            MaxResults=MAX_PAGE_RESULTS,
        )
    except ClientError as e:
        logger.error("Could not complete list_entities request.")
        raise

    EntitySummaryList.extend(response["EntitySummaryList"])

    # Get subsequent pages of results if previous response contained a NextToken
    while "NextToken" in response:
        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType=entity_type,
                MaxResults=MAX_PAGE_RESULTS,
                NextToken=response["NextToken"],
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request.")
            raise

        EntitySummaryList.extend(response["EntitySummaryList"])

    # if visibility is provided, filter the list to only include entities with that
 visibility
    if visibility is not None:
        EntitySummaryList = [
            entity for entity in EntitySummaryList if entity["Visibility"] ==
 visibility
        ]

    return EntitySummaryList


def get_enhanced_product_list(entity_type):
    """
```

```
    Returns an enhanced list of products with product details and offer details
    Args: entity_type str: Type of entity list to return, in our case AmiProduct
    Returns: list: Enhanced list of dictionary objects containing product and offer
 details
    """

    product_list = get_entities(entity_type)

    # Loop through product list and append product details to each product
    for product in product_list:
        # appends product details to product dictionary
        product["ProductDetails"] = describe_entity(product["EntityId"])["Details"]
        # creating an empty list for offer details
        product["OfferDetailsList"] = []

    return product_list


def attach_offer_details(product_list):
    """
    Loops through offer information and appends offer details to product list
    Args: product_list list: List of product dictionaries
    Returns: list: Enhanced list of dictionary objects containing product and offer
 details
    """
    offer_list = get_entities("Offer", "Public")

    # Loop through offer list and append offer details to each product
    for offer in offer_list:
        offer["OfferDetails"] = describe_entity(offer["EntityId"])["Details"]

        # Extracts product-id from offer
        product_id = offer["OfferDetails"]["ProductId"]

        # Determines if product-id referenced in offer matches product-id in product
 list
        product_dict = next(
            filter(lambda product: product["EntityId"] == product_id, product_list),
            None,
        )

        # If product-id matches, appends offer details to product dictionary
        if product_dict is not None:
```

```
            # logger.info(f"Offer product Id {offer['OfferDetails']['ProductId']}
 found in product dictionary. Updating product dictionary with offer details")
            product_dict["OfferDetailsList"].append(offer["OfferDetails"])

        else:
            # logger.info("Offer product Id {offer['OfferDetails']['ProductId']} not
 found. Skipping offer details update")
            pass

    return product_list


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Retrieving products and public offer information....")
    print("-" * 88)

    # Builds a list of products and their details
    product_list = get_enhanced_product_list("AmiProduct")

    # Queries offer information and attaches it to the product list
    product_offer_list = attach_offer_details(product_list)

    pretty_print(product_offer_list)
    return product_offer_list


if __name__ == "__main__":
    usage_demo()
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - DescribeEntity
  - ListEntities

Autorização de revenda

Criar um rascunho de autorização de revenda

O exemplo de código a seguir mostra como criar um rascunho de autorização de revenda para qualquer tipo de produto a fim de analisá-lo internamente antes de publicar para um parceiro de canal.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product so my CP
 can use that to create Channel Partner Private Offer (CPPO)
CAPI-41
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "draft resale auth")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Descrever uma autorização de revenda

O exemplo de código a seguir mostra como descrever uma autorização de revenda.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for product information in
 the AWS Marketplace Catalog
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

resaleAuthorizationId = "resaleauthz-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_product_information(mp_client, entity_id):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Product with ID %s not found.", entity_id)
        else:
```

```
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for a product in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_product_information(mp_client, resaleAuthorizationId))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeEntity Referência da API AWS SDK for Python (Boto3).

Publicar uma autorização única de revenda com uma oferta privada

O exemplo de código a seguir mostra como publicar uma autorização de revenda única com uma oferta privada para que um parceiro de canal possa usá-la a fim de criar uma oferta privada de parceiro de canal (CPPO).

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product so my CP
can use that to create Channel Partner Private Offer (CPPO)
CAPI-42
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set, "onetime resale auth with private offer"
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar a autorização de revenda multiuso com data de validade

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso com uma data de validade para um produto de AMI com preços anuais calculados por hora a fim de que um parceiro de canal possa usá-la para criar uma CPPO.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
```

```
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI product so
 my CP can use that to create Channel Partner Private Offer (CPPO)
CAPI-48
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "multi-use resale auth with expiry date")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar a autorização de revenda multiuso com data de validade e um EULA

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
```

```
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Publish a multi-use resale authorization with expiry date on my SaaS/AMI/Container
 product and add custom EULA to be sent to the buyer
CAPI-56
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multiuse resale auth with expiry date and custom
 EULA")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar a autorização de revenda multiuso com data de validade e documentação do contrato de revendedor

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleUsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
```

```
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI/Container
 product
and add reseller contract documentation between the ISV and channel partner
CAPI-57
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
```

```
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "multi use resale auth with contract doc",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar a autorização de revenda multiuso com data de validade e adicionar uma conta de comprador específica

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso com uma data de validade para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish multi-use resale authorization with expiry date for any product type (AMI/
SaaS/Container) and add specific buyer account for the resale
CAPI-82
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(
        change_set,
        "Publish multi-use resale authorization with expiry date and add specific
 buyer account",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar a autorização de revenda multiuso sem data de validade

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso sem data de validade para um produto de AMI com preços anuais calculados por hora para que um CP possa usá-la para criar uma CPPO.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
```

```
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI product
 so my CP can use that to create Channel Partner Private Offer (CPPO)
CAPI-52
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multi use resale auth with no expiry date")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar a autorização de revenda multiuso sem data de validade e um EULA

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
```

```
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI/
Container product and add custom EULA to be sent to the buyer
CAPI-58
"""

import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "multi use resale auth with no expiry date and custom EULA"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar a autorização de revenda multiuso sem data de validade e documentação do contrato de revendedor

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
```

```
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI/
Container product and add reseller contract documentation between the ISV and
 channel partner
CAPI-59
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "multi use resale auth with no expiry date and contract doc"
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar a autorização de revenda multiuso sem data de validade e adicionar uma conta de comprador específica

O exemplo de código a seguir mostra como publicar uma autorização de revenda multiuso sem uma data de validade para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
```

```
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish multi-use resale authorization without expiry date for any product type
 (AMI/SaaS/Container) and add specific buyer account for the resale
CAPI-83
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "publish multi-use resale authorization without expiry date and add specific
 buyer account",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar uma autorização de revenda única e adicionar uma programação de pagamento flexível

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar uma programação de pagamento flexível.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
                            "DimensionKey": "Users",
                            "MaxQuantity": 10
                          }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ResalePaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2023-09-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2023-12-01",
                                "ChargeAmount": "250.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-06-30",
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container) and
 add Flexible payment schedule
CAPI-78
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "onetime resale auth with flexible payment schedule")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar uma autorização de revenda única e adicionar um EULA

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar um EULA personalizado para ser enviado ao comprador.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
```

```
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product and add
 custom EULA to be sent to the buyer
```

```
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with custom EULA")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar uma autorização de revenda única e adicionar uma conta de comprador específica

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar uma conta de comprador específica para a revenda.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container) and
 add specific buyer account for the resale
CAPI-81
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "onetime resale authorization for specific buyer
 account")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar uma autorização de revenda única e adicionar a documentação do contrato de revendedor

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar a documentação do contrato de revendedor entre o ISV e o parceiro de canal.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
```

```
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
```

```
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product and add
 reseller contract documentation between the ISV and channel partner
CAPI-47
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with reseller contract doc")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar uma autorização de revenda única e adicionar se é uma renovação

O exemplo de código a seguir mostra como publicar uma autorização de revenda única para qualquer tipo de produto e adicionar se é uma renovação.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container)
and add whether it is renewal or not
CAPI-90
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(stringified_change_set, "onetime resale auth renewal")

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Restringir autorização de revenda

O exemplo de código a seguir mostra como restringir a autorização de revenda.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Restrict a  authorization for any product type (AMI/SaaS/Container)
CAPI-84
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Restrict resale authorization")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar o nome e a descrição da autorização de revenda única ou multiuso

O exemplo de código a seguir mostra como atualizar o nome e a descrição da autorização de revenda única ou multiuso antes da publicação de qualquer tipo de produto.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
```

```
Purpose
Update name/description of one-time or multi-use resale authorization before
 publishing for any product type (AMI/SaaS/Container)
CAPI-77
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "update name and description of one-time or multi-use resale authorization
 before publishing",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Produtos de SaaS do

Criar um rascunho de um produto de SaaS com um rascunho de oferta pública

O exemplo de código a seguir mostra como criar um rascunho de produto de SaaS com um rascunho de oferta pública.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to create an SaaS draft product
with a draft public offer.
CAPI-04
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create a draft saas product with a draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar um produto de SaaS público ou limitado e uma oferta pública com preços calculados por contrato

O exemplo de código a seguir mostra como criar um produto de SaaS público ou limitado e uma oferta pública com preços contratuais. Este exemplo cria um EULA padrão ou personalizado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
```

```
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with contract pricing and standard
 EULA
CAPI-11
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create a limited saas product with a public offer with contract pricing",
    )


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Crie um produto SaaS público ou limitado e uma oferta pública com Pay-As-You-Go contrato e preços

O exemplo de código a seguir mostra como criar um produto SaaS público ou limitado e uma oferta pública com um contrato com Pay-As-You-Go preços. Este exemplo cria um EULA padrão ou personalizado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
```

```
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
```

```
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with contract with PAYG pricing and
 standard EULA
CAPI-10
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create limited SaaS product with public offer with contract with payg
 pricing",
    )


if __name__ == "__main__":
```

```
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Criar um produto de SaaS público ou limitado e uma oferta pública com preços de assinatura

O exemplo de código a seguir mostra como criar um produto de SaaS público ou limitado e uma oferta pública com preços de assinatura. Este exemplo cria um EULA padrão ou personalizado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
```

```
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
```

```
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with subscription(usage) pricing and
 standard EULA
CAPI-09
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "public saas public offer with subscription pricing")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Publicar um produto de SaaS e uma oferta pública associada

O exemplo de código a seguir mostra como publicar um produto de SaaS e a oferta pública associada. Por padrão, o produto estará em um estado limitado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
```

```
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
```

```
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
```

```
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish my SaaS product and associated public offer (product will be in limited
 state by default)
CAPI-05A
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
def main(change_set=None):
    if change_set is None:
        fname = "changeset1.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "publish saas product and associated public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Atualizar dimensões em um produto de AMI ou de SaaS

O exemplo de código a seguir mostra como atualizar dimensões em um produto de AMI ou de SaaS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

Execute esse script para iniciar o conjunto de alterações. As funções auxiliares são definidas em Utilitários para iniciar um conjunto de alterações na seção Utilitários.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update (e.g name) dimensions on
 my AMI or SaaS product
CAPI-24
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
```

```
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Update name dimensions on my AMI or SaaS product",
    )

    return response


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

Serviços públicos

Utilitários para iniciar um conjunto de alterações

O exemplo de código a seguir mostra como definir utilitários para iniciar um conjunto de alterações.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

Utilitário para iniciar um conjunto de alterações.

```
"""
Purpose:
```

```
Generic function to start a changeset
"""

import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def generate_changeset(mp_client, change_set, change_set_name):
    """
    Start changeset
    """
    try:
        response = mp_client.start_change_set(
            Catalog="AWSMarketplace",
            ChangeSet=change_set,
            ChangeSetName=change_set_name,
        )
        logger.info("Changeset created!")
        logger.info("ChangeSet ID: %s", response["ChangeSetId"])
        logger.info("ChangeSet ARN: %s", response["ChangeSetArn"])

        return response

    except ClientError as e:
        logger.exception("Unexpected error: %s", e)
        raise


def usage_demo(change_set, change_set_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Executing changeset: " + change_set_name)
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    response = generate_changeset(mp_client, change_set, change_set_name)

    return response
```

```
    print("-" * 88)
```

Utilitário para carregar um conjunto de alterações de um arquivo JSON.

```
"""
Purpose:

This module will stringify the details sections of a changeset file.
"""

import json


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


# open json file from path
def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


def stringify_details_sections(json_object):
    """
    Loops through every change type in the changeset to look for non-empty
    details section and stringifies them
    """
    for change_type in json_object["ChangeSet"]:
        # Only stringify details section if it is not empty
        if "Details" in change_type and change_type["Details"] != "{}":
            string_details = json.dumps(change_type["Details"])
            change_type["Details"] = string_details
        else:
            pass

    return json_object["ChangeSet"]


def stringify_changeset(file_path):
```

```
    changeset_file = open_json_file(file_path)
    changeset_stringified = stringify_details_sections(changeset_file)

    return changeset_stringified
```

- Para obter detalhes da API, consulte a StartChangeSet Referência da API AWS SDK for Python (Boto3).

AWS Marketplace Exemplos de API de contrato usando SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS SDK para Python (Boto3) with AWS Marketplace Agreement.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Contratos

Contratos

Aceitar uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como aceitar uma solicitação de cancelamento de contrato iniciada pelo vendedor.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
```

```
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class AcceptAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID HERE>"

    @staticmethod
    def accept_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.accept_agreement_cancellation_request(
            agreementId=AcceptAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=AcceptAgreementCancellationRequest.AGREEMENT_CANCELLATION_RE
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Description: " + str(response.get("description", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    AcceptAgreementCancellationRequest.accept_agreement_cancellation_request()
```

- Para obter detalhes da API, consulte a AcceptAgreementCancellationRequest Referência da API AWS SDK for Python (Boto3).

Aceitar uma solicitação de pagamento por contrato

O exemplo de código a seguir mostra como aceitar uma solicitação de pagamento por contrato iniciada pelo vendedor.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class AcceptAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"
    PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER REFERENCE HERE>"

    @staticmethod
    def accept_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.accept_agreement_payment_request(
            agreementId=AcceptAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=AcceptAgreementPaymentRequest.PAYMENT_REQUEST_ID,

 purchaseOrderReference=AcceptAgreementPaymentRequest.PURCHASE_ORDER_REFERENCE,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))
```

```
if __name__ == "__main__":
    AcceptAgreementPaymentRequest.accept_agreement_payment_request()
```

- Para obter detalhes da API, consulte a AcceptAgreementPaymentRequest Referência da API AWS SDK for Python (Boto3).

Alterar um termo de renovação do contrato de SaaS

O exemplo de código a seguir mostra como alterar um contrato de SaaS para adicionar ou atualizar seu prazo de renovação.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create a SaaS agreement with CONTRACT pricing model and then
 turn on
the auto-renewal setting using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product using a public offer that supports
auto-renewal. After acceptance, the buyer decides to amend the agreement to enable
auto-renewal via the RenewalTerm configuration.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P1M for 1 month).
  - DIMENSION_1_KEY — the dimension key defined in the offer.
"""

import boto3
```

```
from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class AmendSaaSContractRenewalTerm:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
    SELECTOR_VALUE = "<your-selector-value>"

    # The dimension key defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the RenewalTerm in your offer.
    RENEWAL_TERM_ID = "<your-renewal-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the SupportTerm in your offer.
    SUPPORT_TERM_ID = "<your-support-term-id>"

    @staticmethod
    def amend_saas_contract_agreement_renewal_term():
        """
        Full end-to-end flow:
        1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
 disabled.
        2. Wait for entitlements to become active.
        3. Amend the agreement to enable auto-renewal.
        """
```

```
        client = boto3.client("marketplace-agreement")
        cls = AmendSaaSContractRenewalTerm

        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        # Initial agreement: auto-renewal disabled.
        renewal_term = {
            "id": cls.RENEWAL_TERM_ID,
            "configuration": {
                "renewalTermConfiguration": {
                    "enableAutoRenew": False,
                }
            },
        }

        legal_term = {"id": cls.LEGAL_TERM_ID}
        support_term = {"id": cls.SUPPORT_TERM_ID}

        # --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, renewal_term,
legal_term, support_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)
```

```
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request accepted. AgreementId: " + agreement_id)

        # Wait for entitlements to become active before amending.
        print("Waiting for entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
 agreement_id)
        print("Entitlements are now active.")
        format_output(entitlements_response)

        # --- Amend: enable auto-renewal ---
        renewal_term_amended = {
            "id": cls.RENEWAL_TERM_ID,
            "configuration": {
                "renewalTermConfiguration": {
                    "enableAutoRenew": True,
                }
            },
        }

        # Use Intent.AMEND and sourceAgreementIdentifier to target the existing
 agreement.
        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="AMEND",
            requestedTerms=[configurable_upfront_pricing_term, renewal_term_amended,
 legal_term, support_term],
            sourceAgreementIdentifier=agreement_id,
        )
        print("Amend agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        print("Amendment accepted. Auto-renewal enabled. New AgreementId: " +
 aar_response["agreementId"])


if __name__ == "__main__":
    AmendSaaSContractRenewalTerm.amend_saas_contract_agreement_renewal_term()
```

- Para obter detalhes da API, consulte a CreateAgreementRequest Referência da API AWS SDK for Python (Boto3).

Alterar uma AMI ConfigurableUpfrontPricingTerm para o modelo de preços de uso

O exemplo de código a seguir mostra como alterar um contrato de AMI ConfigurableUpfrontPricingTerm atualizando a quantidade da dimensão para o modelo de preços de uso.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create an AMI agreement with ConfigurableUpfrontPricingTerm and
 then amend
the dimension quantity using the AWS Marketplace Agreement Service APIs.

Scenario: An AMI product with USAGE pricing requires two agreements:
  1. An agreement with usageBasedPricingTerm (UBPT) — accepted first to establish
 the base agreement.
  2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
 UBPT agreement
     entitlements are active.
Once both agreement entitlements are available, this sample shows how to amend the
 agreement
with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P365D for one year).
  - DIMENSION_1_KEY — the dimension key defined in the offer (e.g., instance type).
```

```
  - DIMENSION_1_VALUE — initial quantity; NEW_DIMENSION_1_VALUE — amended quantity.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P365D" for 365 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Initial quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the UsageBasedPricingTerm in your offer.
    USAGE_TERM_ID = "<your-usage-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in your offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # New quantity to use when amending the dimension of CUPT.
    NEW_DIMENSION_1_VALUE = 5

    @staticmethod
```

```
    def amend_ami_cupt_agreement():
        """
        Full end-to-end flow:
        1. Create and accept an agreement request with usageBasedPricingTerm.
        2. Wait for entitlements to become active, then create and accept an
agreement request
           with configurableUpfrontPricingTerm (CUPT).
        3. Wait for CUPT entitlements to become active, then amend the dimension
quantity.
        """
        client = boto3.client("marketplace-agreement")
        cls = AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel

        usage_term = {"id": cls.USAGE_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Agreement with UBPT ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_term, legal_term, validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " + agreement_id)

        # Wait for entitlements to become active before creating the agreement with
CUPT.
        print("Waiting for UBPT agreement entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("UBPT agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
        configurable_upfront_pricing_term = {
```

```
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        print("Agreement request with CUPT created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        cupt_agreement_id = aar_response["agreementId"]
        print("Agreement request with CUPT accepted. AgreementId: " +
cupt_agreement_id)

        # Wait for entitlements to become active before amending.
        print("Waiting for CUPT agreement entitlements to become active...")
        cupt_entitlements_response = poll_until_entitlements_available(client,
cupt_agreement_id)
        print("CUPT agreement entitlements are now active.")
        format_output(cupt_entitlements_response)

        # --- Amend Agreement with CUPT ---
        # Increase the dimension quantity using Intent.AMEND and
sourceAgreementIdentifier.
        new_config = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
```

```
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.NEW_DIMENSION_1_VALUE,  # Increase
 quantity for this dimension key
                        },
                    ],
                }
            },
        }

        car_amend_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="AMEND",
            requestedTerms=[new_config, legal_term, validity_term],
            sourceAgreementIdentifier=cupt_agreement_id,
        )
        print(
            "Amendment of CUPT agreement request created. AgreementRequestId: "
            + car_amend_response["agreementRequestId"]
        )

        aar_amend_response = client.accept_agreement_request(
            agreementRequestId=car_amend_response["agreementRequestId"]
        )
        print("Amendment accepted. New AgreementId: " +
 aar_amend_response["agreementId"])


if __name__ == "__main__":

 AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel.amend_ami_cupt_agreement()
```

- Para obter detalhes da API, consulte a CreateAgreementRequest Referência da API AWS SDK for Python (Boto3).

Crie um novo contrato de teste gratuito da AMI

O exemplo de código a seguir mostra como criar um novo contrato de teste gratuito da AMI.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create an AMI Free Trial agreement
using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to an AMI product that offers a free trial period.
The free trial includes a FreeTrialPricingTerm alongside a UsageBasedPricingTerm.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
"""

import boto3

from utils.agreement_api_utils import generate_client_token


class NewAmiFreeTrial:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the FreeTrialPricingTerm in your offer.
    FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>"

    # Term ID for the UsageBasedPricingTerm in your offer (applies after the trial
 ends).
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the SupportTerm in your offer.
    SUPPORT_TERM_ID = "<your-support-term-id>"
```

```
    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    @staticmethod
    def create_and_accept_ami_free_trial_agreement_request():
        """
        Create an AMI Free Trial agreement.

        The FreeTrialPricingTerm grants access at no cost for the trial period.
        The UsageBasedPricingTerm defines the charges that apply once the trial
 ends.
        """
        client = boto3.client("marketplace-agreement")

        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[
                {"id": NewAmiFreeTrial.FREE_TRIAL_PRICING_TERM_ID},
                {"id": NewAmiFreeTrial.USAGE_BASED_PRICING_TERM_ID},
                {"id": NewAmiFreeTrial.SUPPORT_TERM_ID},
                {"id": NewAmiFreeTrial.LEGAL_TERM_ID},
            ],

 agreementProposalIdentifier=NewAmiFreeTrial.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print(
            "Agreement request with freeTrialPricingTerm accepted. AgreementId: "
            + accept_response["agreementId"]
        )


if __name__ == "__main__":
    NewAmiFreeTrial.create_and_accept_ami_free_trial_agreement_request()
```

- Para obter detalhes da API, consulte a CreateAgreementRequest Referência da API AWS SDK for Python (Boto3).

Crie um novo contrato de SaaS com pagamento adiantado

O exemplo de código a seguir mostra como criar um novo contrato de SaaS com pagamento adiantado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create a SaaS agreement with CONTRACT pricing model with upfront
 payment
using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
selecting an agreement duration and specifying quantities for multiple dimensions.
Tax estimation is enabled at the time of agreement creation.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P12M for 12 months).
  - DIMENSION_1_KEY, DIMENSION_2_KEY — dimension keys defined in the offer.
  - DIMENSION_1_VALUE, DIMENSION_2_VALUE — quantities for each dimension.
"""

import boto3

from utils.agreement_api_utils import generate_client_token
```

```
class NewSaaSContractWithUpfrontPayment:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P12M" for 12 months).
    SELECTOR_VALUE = "<your-selector-value>"

    # First dimension key and quantity defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-1-key>"
    DIMENSION_1_VALUE = 10

    # Second dimension key and quantity defined in your offer.
    DIMENSION_2_KEY = "<your-dimension-2-key>"
    DIMENSION_2_VALUE = 20

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Tax estimation setting: "ENABLED" to include estimated taxes in the agreement.
    TAX_ESTIMATION = "ENABLED"

    @staticmethod
    def create_saas_contract_agreement():
        """
        Create a SaaS agreement with CONTRACT pricing model with configurable
 upfront pricing
        for multiple dimensions and tax estimation.
        """
        client = boto3.client("marketplace-agreement")

        configurable_upfront_pricing_term = {
            "id":
 NewSaaSContractWithUpfrontPayment.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue":
 NewSaaSContractWithUpfrontPayment.SELECTOR_VALUE,
                    "dimensions": [
                        {
```

```
                            "dimensionKey":
 NewSaaSContractWithUpfrontPayment.DIMENSION_1_KEY,
                            "dimensionValue":
 NewSaaSContractWithUpfrontPayment.DIMENSION_1_VALUE,
                        },
                        {
                            "dimensionKey":
 NewSaaSContractWithUpfrontPayment.DIMENSION_2_KEY,
                            "dimensionValue":
 NewSaaSContractWithUpfrontPayment.DIMENSION_2_VALUE,
                        },
                    ],
                }
            },
        }

        legal_term = {"id": NewSaaSContractWithUpfrontPayment.LEGAL_TERM_ID}

        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term],
            taxConfiguration={"taxEstimation":
 NewSaaSContractWithUpfrontPayment.TAX_ESTIMATION},

 agreementProposalIdentifier=NewSaaSContractWithUpfrontPayment.AGREEMENT_PROPOSAL_IDENTIFIER
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print(
            "SaaS agreement request with CONTRACT pricing model accepted.
 AgreementId: "
            + accept_response["agreementId"]
        )


if __name__ == "__main__":
    NewSaaSContractWithUpfrontPayment.create_saas_contract_agreement()
```

- Para obter detalhes da API, consulte a CreateAgreementRequest Referência da API AWS SDK for Python (Boto3).

Obter todos os IDs de contratos

O exemplo de código a seguir mostra como obter todos os IDs de contratos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreement ids
AG-09
"""

import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10


def get_agreements():
    AgreementSummaryList = []
    agreement_id_list = []
```

```
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you are
 the proposer.
            # Change to "Acceptor" to return agreements where you are the acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise

    AgreementSummaryList.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreements["agreementViewSummaries"])

    for agreement in AgreementSummaryList:
        agreement_id_list.append(agreement["agreementId"])

    return agreement_id_list


if __name__ == "__main__":
    agreement_id_list = get_agreements()
```

```
    print(agreement_id_list)
```

- Para obter detalhes da API, consulte a SearchAgreements Referência da API AWS SDK for Python (Boto3).

Obter todos os contratos

O exemplo de código a seguir mostra como obter todos os contratos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreements
AG-01
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")
```

```
logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

party_type_list = ["Acceptor"]
agreement_type_list = ["PurchaseAgreement"]

filter_list = [
    {"name": "PartyType", "values": party_type_list},
    {"name": "AgreementType", "values": agreement_type_list},
]

agreement_results_list = []


def get_agreements(filter_list=filter_list):
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            filters=filter_list,
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    agreement_results_list.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=filter_list,
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        agreement_results_list.extend(agreements["agreementViewSummaries"])

    return agreement_results_list
```

```
if __name__ == "__main__":
    agreements_list = get_agreements(filter_list)
    helper.pretty_print_datetime(agreements_list)
```

- Para obter detalhes da API, consulte a SearchAgreements Referência da API AWS SDK for Python (Boto3).

Obter o ID do cliente de um contrato

O exemplo de código a seguir mostra como obter um ID de cliente de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get customer AWS account id
from a given agreement
AG-08
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)
```

```
def get_agreement_information(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
            raise e
        else:
            logger.error("Unexpected error: %s", e)
            raise e

    return response


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement_information(agreement_id=args.agreement_id)

    print(f"Customer account: {response['acceptor']['accountId']}")
```

- Para obter detalhes da API, consulte a DescribeAgreement Referência da API AWS SDK for Python (Boto3).

Obtenha detalhes de uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como obter detalhes de uma solicitação específica de cancelamento de contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class GetAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID HERE>"

    @staticmethod
    def get_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.get_agreement_cancellation_request(
            agreementId=GetAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=GetAgreementCancellationRequest.AGREEMENT_CANCELLATION_REQUE
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))
```

```
if __name__ == "__main__":
    GetAgreementCancellationRequest.get_agreement_cancellation_request()
```

- Para obter detalhes da API, consulte a GetAgreementCancellationRequest Referência da API AWS SDK for Python (Boto3).

Obtenha detalhes de uma solicitação de pagamento por contrato

O exemplo de código a seguir mostra como obter detalhes de uma solicitação de pagamento de um contrato específico.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class GetAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"

    @staticmethod
    def get_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.get_agreement_payment_request(
            agreementId=GetAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=GetAgreementPaymentRequest.PAYMENT_REQUEST_ID,
```

```
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge ID: " + str(response.get("chargeId", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    GetAgreementPaymentRequest.get_agreement_payment_request()
```

- Para obter detalhes da API, consulte a GetAgreementPaymentRequest Referência da API AWS SDK for Python (Boto3).

Obter detalhes financeiros de um contrato

O exemplo de código a seguir mostra como obter os detalhes financeiros de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain financial details, such as Total Contract Value of the agreementfrom a given
 agreement
AG-14
```

```
Example Usage: python3 get_agreement_financial_details.py --agreement-id <agreement-
id>
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_information(agreement_id):
    try:
        agreement = mp_client.describe_agreement(agreementId=agreement_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_information(args.agreement_id)

    if agreement is not None:
```

```
        print(f"Agreement Id: {args.agreement_id}")
        print(
            f"Agreement Value: {agreement['estimatedCharges']['currencyCode']}
 {agreement['estimatedCharges']['agreementValue']}"
        )

    else:
        print(f"Agreement with ID {args.agreement_id} is not found")
```

- Para obter detalhes da API, consulte a DescribeAgreement Referência da API AWS SDK for Python (Boto3).

Obter detalhes do teste gratuito de um contrato

O exemplo de código a seguir mostra como obter os detalhes do teste gratuito de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the details from an agreement of a free trial I have provided to the customer
AG-20

Example Usage: python3 get_agreement_free_trial_details.py --agreement-id
 <agreement-id>
"""

import argparse
import logging

import boto3
```

```
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_terms(agreement_id):
    try:
        agreement = mp_client.get_agreement_terms(agreementId=agreement_id)
        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)

        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_terms(agreement_id=args.agreement_id)

    if agreement is not None:
        freetrial_found = False

        for term in agreement["acceptedTerms"]:
```

```
            if "freeTrialPricingTerm" in term.keys():
                helper.pretty_print_datetime(term)
                freetrial_found = True

        if not freetrial_found:
            print(f"No free trial term found for agreement: {args.agreement_id}")
```

- Para obter detalhes da API, consulte a DescribeAgreement Referência da API AWS SDK for Python (Boto3).

Obter informações sobre um contrato

O exemplo de código a seguir mostra como obter informações sobre um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get agreement information
AG-07
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))
```

```
import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)


def get_agreement_information(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
            raise e
        else:
            logger.error("Unexpected error: %s", e)
            raise e

    return response


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement_information(agreement_id=args.agreement_id)

    helper.pretty_print_datetime(response)
```

- Para obter detalhes da API, consulte a DescribeAgreement Referência da API AWS SDK for Python (Boto3).

Obter detalhes do produto e da oferta de um contrato

O exemplo de código a seguir mostra como obter os detalhes de produtos e ofertas com base em um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get product and offer details in
 a given agreement
AG-10
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mpa_client = boto3.client("marketplace-agreement")
mpc_client = boto3.client("marketplace-catalog")

logger = logging.getLogger(__name__)


def get_agreement_information(agreement_id):
```

```
    """
    Returns information about a given agreement
    Args: agreement_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        agreement = mpa_client.describe_agreement(agreementId=agreement_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_entity_information(entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mpc_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_agreement_components(agreement_id):
    agreement_component_list = []

    agreement = get_agreement_information(agreement_id)
```

```
    if agreement is not None:
        productIds = []
        for resource in agreement["proposalSummary"]["resources"]:
            productIds.append(resource["id"])

        for product_id in productIds:
            product_document = get_entity_information(product_id)

            product_document_dict = {}
            product_document_dict["product_id"] = product_id
            product_document_dict["document"] = product_document
            agreement_component_list.append(product_document_dict)

        offerId = agreement["proposalSummary"]["offerId"]

        offer_document = get_entity_information(offerId)

        offer_document_dict = {}
        offer_document_dict["offer_id"] = offerId
        offer_document_dict["document"] = offer_document
        agreement_component_list.append(offer_document_dict)

        return agreement_component_list

    else:
        print("Agreement with ID " + args.agreement_id + " is not found")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement_id",
        "-aid",
        help="Provide agreement ID to search for product and offer detail",
        required=True,
    )
    args = parser.parse_args()

    product_offer_detail = get_agreement_components(agreement_id=args.agreement_id)

    helper.pretty_print_datetime(product_offer_detail)
```

- Para obter detalhes da API, consulte a DescribeAgreement Referência da API AWS SDK for Python (Boto3).

Obter o EULA de um contrato

O exemplo de código a seguir mostra como obter o EULA de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the EULA I have entered into with my customer via the agreement
AG-18
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
# agreement id
AGREEMENT_ID = "agmt-11111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "legalTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "documents"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        legalEulaArray = []
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term and ATTRIBUTE_NAME in term[TERM_NAME]:
                docs = term[TERM_NAME][ATTRIBUTE_NAME]
                for doc in docs:
                    if "type" in doc:
                        legalEulaArray.append(doc)
        return legalEulaArray

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)
```

```
def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTerms Referência da API AWS SDK for Python (Boto3).

Obter os termos de renovação automática de um contrato

O exemplo de código a seguir mostra como obter os termos de renovação automática de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the auto-renewal status of the agreement
AG-15
"""

import json
import logging
import os
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper


import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-1111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "renewalTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "enableAutoRenew"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """
```

```
    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        auto_renewal = "No Auto Renewal"
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    auto_renewal = term[TERM_NAME][CONFIG_ELEM][ATTRIBUTE_NAME]
                    break
        return auto_renewal

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    agreement = get_agreement_information(mp_client, AGREEMENT_ID)

    if agreement is not None:
        print(f"Auto Renewal is {agreement}")
    else:
        print("Agreement with ID " + AGREEMENT_ID + " is not found")


# open json file from path
def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)
```

```
if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTerms Referência da API AWS SDK for Python (Boto3).

Obter as dimensões adquiridas em um contrato

O exemplo de código a seguir mostra como obter as dimensões compradas em um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the dimensions the buyer has purchased from me via the agreement
AG-28
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        dimensionKeys = []

        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    confParam = term[TERM_NAME][CONFIG_ELEM]
                    if ATTRIBUTE_NAME in confParam:
                        if "dimensions" in confParam:
                            for dimension in confParam["dimensions"]:
                                if "dimensionKey" in dimension:
                                    dimensionKey = dimension["dimensionKey"]
                                    print(f"Dimension Key: {dimensionKey}")
                                    dimensionKeys.append(dimensionKey)
```

```
        return dimensionKeys

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTerms Referência da API AWS SDK for Python (Boto3).

Obter as instâncias de cada dimensão adquirida em um contrato

O exemplo de código a seguir mostra como obter as instâncias de cada dimensão compradas em um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain instances of each dimension that buyer has purchased in the agreement
AG-30
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"

logger = logging.getLogger(__name__)
```

```
def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        terms = mp_client.get_agreement_terms(agreementId=entity_id)
        dimensionKeyValueMap = {}
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    confParam = term[TERM_NAME][CONFIG_ELEM]
                    if ATTRIBUTE_NAME in confParam:
                        selectValue = confParam["selectorValue"]
                        dimensionKeyValueMap["selectorValue"] = selectValue
                        if "dimensions" in confParam:
                            dimensionKeyValueMap["dimensions"] =
 confParam["dimensions"]
                            """
                            for dimension in confParam['dimensions']:
                                if 'dimensionKey' in dimension:

                                    dimensionValue = dimension['dimensionValue']
                                    dimensionKey = dimension['dimensionKey']
                                    print(f"Selector: {selectValue}, Dimension Key:
 {dimensionKey}, Dimension Value: {dimensionValue}")
                                    dimensionKeyValueMap[dimensionKey] =
 dimensionValue
                            """
        return dimensionKeyValueMap

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
```

```
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTerms Referência da API AWS SDK for Python (Boto3).

Obter a programação de pagamento de um contrato

O exemplo de código a seguir mostra como obter a programação de pagamento de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the payment schedule I have agreed to with the agreement, including the
 invoice date and invoice amount
AG-17
"""

import json
import logging
import os
```

```
import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "paymentScheduleTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        paymentScheduleArray = []
        currencyCode = ""
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                paymentSchedule = term[TERM_NAME]
```

```
                if "currencyCode" in paymentSchedule:
                    currencyCode = paymentSchedule["currencyCode"]
                if "schedule" in paymentSchedule:
                    for sch in paymentSchedule["schedule"]:
                        if "chargeDate" in sch:
                            chargeDate = sch["chargeDate"]
                            chargeAmount = sch["chargeAmount"]
                            # print(f"chargeDate: {chargeDate}, chargeAmount:
 {chargeAmount}")
                            schedule = {
                                "currencyCode": currencyCode,
                                "chargeDate": chargeDate,
                                "chargeAmount": chargeAmount,
                            }
                            paymentScheduleArray.append(schedule)

        return paymentScheduleArray

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)
```

```
if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTerms Referência da API AWS SDK for Python (Boto3).

Obter o preço por dimensão em um contrato

O exemplo de código a seguir mostra como obter os preços por dimensão em um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain pricing per each dimension in the agreement
AG-29
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"

TERMS_TO_SEARCH = [
    "configurableUpfrontPricingTerm",
    "usageBasedPricingTerm",
    "fixedUpfrontPricingTerm",
]


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        dimentions = []
        for term in terms[ROOT_ELEM]:
            for t in TERMS_TO_SEARCH:
                rateInfo = []
                if t in term:
                    if "type" in term[t]:
                        rateInfo.append(term[t]["type"])
                    if "currencyCode" in term[t]:
```

```
                        rateInfo.append(term[t]["currencyCode"])
                    if "rateCards" in term[t]:
                        rateInfo.append(term[t]["rateCards"])
                    dimentions.append(rateInfo)
        return dimentions

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAgreementTerms Referência da API AWS SDK for Python (Boto3).

Obter o tipo de preço de um contrato

O exemplo de código a seguir mostra como obter o tipo de preço de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
AG-16
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

# To search by offer id: OfferId; by product id: ResourceIdentifier; by product
 type: ResourceType
idType = "OfferId"

# replace id value as needed
idValue = "offer-1111111111111"

MAX_PAGE_RESULTS = 10

# catalog; switch to AWSMarketplace for release
AWSMPCATALOG = "AWSMarketplace"

# product types

SaaSProduct = "SaaSProduct"
AmiProduct = "AmiProduct"
MLProduct = "MachineLearningProduct"
ContainerProduct = "ContainerProduct"
DataProduct = "DataProduct"
```

```
ProServiceProduct = "ProfessionalServicesProduct"
AiqProduct = "AiqProduct"

# Define pricing types
CCP = "CCP"
Annual = "Annual"
Contract = "Contract"
SFT = "SaaS Free Trial"
HMA = "Hourly and Monthly Agreements"
Hourly = "Hourly"
Monthly = "Monthly"
AFPS = "Annual FPS"
CFPS = "Contract FPS"
CCPFPS = "CCP with FPS"
BYOL = "BYOL"
Free = "Free"
FTH = "Free Trials and Hourly"

# Define Agreement Term Types
legal = ["LegalTerm"]
config = ["ConfigurableUpfrontPricingTerm"]
usage = ["UsageBasedPricingTerm"]
config_usage = ["ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm"]
freeTrial = ["FreeTrialPricingTerm"]
recur = ["RecurringPaymentTerm"]
usage_recur = ("UsageBasedPricingTerm", "RecurringPaymentTerm")
fixed_payment = ["FixedUpfrontPricingTerm", "PaymentScheduleTerm"]
fixed_payment_usage = [
    "FixedUpfrontPricingTerm",
    "PaymentScheduleTerm",
    "UsageBasedPricingTerm",
]
byol = ["ByolPricingTerm"]
freeTrial_usage = ("FreeTrialPricingTerm", "UsageBasedPricingTerm")
all_agreement_types_combination = (
    legal,
    config,
    usage,
    config_usage,
    freeTrial,
    recur,
    usage_recur,
    fixed_payment,
    fixed_payment_usage,
```

```
    byol,
    freeTrial_usage,
)


# get pricing type method given product type, agreement temr type and offer type if
 needed
def get_pricing_type(product_type, agreement_term_type, offer_type):
    pricing_types = {
        (SaaSProduct, frozenset(config_usage), frozenset("")): CCP,
        (DataProduct, frozenset(config_usage), frozenset("")): CCP,
        (ContainerProduct, frozenset(config), frozenset(config_usage)): Annual,
        (AmiProduct, frozenset(config), frozenset(config_usage)): Annual,
        (MLProduct, frozenset(config), frozenset(config_usage)): Annual,
        (ContainerProduct, frozenset(config), frozenset(config)): Contract,
        (AmiProduct, frozenset(config), frozenset(config)): Contract,
        (SaaSProduct, frozenset(config), frozenset("")): Contract,
        (DataProduct, frozenset(config), frozenset("")): Contract,
        (AiqProduct, frozenset(config), frozenset("")): Contract,
        (ProServiceProduct, frozenset(config), frozenset("")): Contract,
        (SaaSProduct, frozenset(freeTrial), frozenset("")): SFT,
        (AmiProduct, frozenset(usage_recur), frozenset("")): HMA,
        (SaaSProduct, frozenset(usage), frozenset("")): Hourly,
        (AmiProduct, frozenset(usage), frozenset("")): Hourly,
        (ContainerProduct, frozenset(usage), frozenset("")): Hourly,
        (MLProduct, frozenset(usage), frozenset("")): Hourly,
        (ContainerProduct, frozenset(recur), frozenset("")): Monthly,
        (AmiProduct, frozenset(recur), frozenset("")): Monthly,
        (
            ContainerProduct,
            frozenset(fixed_payment),
            frozenset(fixed_payment_usage),
        ): AFPS,
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment_usage)):
 AFPS,
        (MLProduct, frozenset(fixed_payment), frozenset("")): AFPS,
        (ContainerProduct, frozenset(fixed_payment), frozenset(fixed_payment)):
 CFPS,
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment)): CFPS,
        (SaaSProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (DataProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (AiqProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (ProServiceProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (SaaSProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
```

```
        (DataProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (AiqProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (ProServiceProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (AmiProduct, frozenset(byol), frozenset("")): BYOL,
        (SaaSProduct, frozenset(byol), frozenset("")): BYOL,
        (ProServiceProduct, frozenset(byol), frozenset("")): BYOL,
        (AiqProduct, frozenset(byol), frozenset("")): BYOL,
        (MLProduct, frozenset(byol), frozenset("")): BYOL,
        (ContainerProduct, frozenset(byol), frozenset("")): BYOL,
        (DataProduct, frozenset(byol), frozenset("")): BYOL,
        (ContainerProduct, frozenset(legal), frozenset("")): Free,
        (AmiProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
        (ContainerProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
        (MLProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
    }

    key = (product_type, agreement_term_type, offer_type)
    if key in pricing_types:
        return pricing_types[key]
    else:
        return "Unknown"


# Example usage for testing purpose
"""
product_type = SaaSProduct
agreement_term_type = frozenset(config_usage)
offer_type = frozenset('')
pricing_type = get_pricing_type(product_type, agreement_term_type, offer_type)
print("pricing type = " + pricing_type)  # Output: CCP
"""


# check if offer term types are needed; if Y, needed
def get_offer_term_type(product_type, agreement_term_type):
    offer_term_types = {
        (ContainerProduct, frozenset(config)): "Y",
        (AmiProduct, frozenset(config)): "Y",
        (ContainerProduct, frozenset(fixed_payment)): "Y",
        (AmiProduct, frozenset(fixed_payment)): "Y",
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment)): "Y",
    }

    key = (product_type, agreement_term_type)
```

```
    if key in offer_term_types:
        return offer_term_types[key]
    else:
        return


logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog=AWSMPCATALOG,
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": idType, "values": [idValue]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog=AWSMPCATALOG,
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": idType, "values": [idValue]},
                        {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
```

```
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    # find all agreements matching the specified idType and idValue
    agreements = get_agreements(mp_client)

    for item in agreements:
        pricingType = ""
        agreement_id = item["agreementId"]

        # get term types inside offer
        offer_term_types = get_offer_term_types(item)

        # even though multiple product types are allowed for one agreement, only
 need the first one
        productType = item["proposalSummary"]["resources"][0]["type"]

        # get agreement terms types
        agreementTerm = mp_client.get_agreement_terms(agreementId=agreement_id)

        agreementTermTypes = get_agreement_term_types(agreementTerm)

        # match with agreement term type group
        matchedTermType = getMatchedTermTypesCombination(agreementTermTypes)

        # check if offer term type is needed.
        offer_term_type_needed = get_offer_term_type(
            productType, frozenset(matchedTermType)
        )
```

```
        # get pricing type given product type, agreement term types and offer type
 if needed;
        # one excpetion is Container with Legal term. LegalTerm needs to be the only
 term present
        if offer_term_type_needed is not None:
            matchedOfferTermTypes = getMatchedTermTypesCombination(offer_term_types)
            print(f"matchedOfferTermType = {matchedOfferTermTypes}")
            pricingType = get_pricing_type(
                productType,
                frozenset(matchedTermType),
                frozenset(matchedOfferTermTypes),
            )
        elif set(matchedTermType) == set(legal):
            pricingType = Free
        else:
            pricingType = get_pricing_type(
                productType, frozenset(matchedTermType), frozenset("")
            )

        print(
            f"agreementId={agreement_id};productType={productType};
 agreementTermTypes={agreementTermTypes}; matchedTermType={matchedTermType};
 offerTermTypeNeeded={offer_term_type_needed}; offer_term_types={offer_term_types}"
        )
        print(f"pricing type={pricingType}")


def getMatchedTermTypesCombination(agreementTermTypes):
    matchedCombination = ()
    for element in all_agreement_types_combination:
        if check_elements(agreementTermTypes, element):
            matchedCombination = element
    return matchedCombination


def get_offer_term_types(item):
    offer_id = item["proposalSummary"]["offerId"]
    mp_catalogAPI_client = boto3.client("marketplace-catalog")
    offer_document = get_entity_information(mp_catalogAPI_client, offer_id)
    offerDetail = offer_document["Details"]
    offerDetail_json_object = json.loads(offerDetail)
    offer_term_types = [term["Type"] for term in offerDetail_json_object["Terms"]]
    return offer_term_types
```

```
# make sure all elements in array2 exist in array1
def check_elements(array1, array2):
    for element in array2:
        if element not in array1:
            return False
    return True


def get_entity_information(mp_client, entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_agreement_term_types(agreementTerm):
    types = []
    for term in agreementTerm["acceptedTerms"]:
        for value in term.values():
            if isinstance(value, dict) and "type" in value:
                types.append(value["type"])
    return types


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeAgreement Referência da API AWS SDK for Python (Boto3).

Obter o tipo de produto de um contrato

O exemplo de código a seguir mostra como obter o tipo de produto de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the Product Type of the product the agreement was created on
AG-11
"""

import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"


def get_agreement_information(mp_client, entity_id):
    """
    Returns information about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """
```

```
    try:
        agreement = mp_client.describe_agreement(agreementId=entity_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for offer and product details in a given agreement by agreement
 id.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    agreement = get_agreement_information(mp_client, AGREEMENT_ID)

    if agreement is not None:
        for resource in agreement["proposalSummary"]["resources"]:
            print(f"Product ID: {resource['id']}  |  Product Type:
 {resource['type']}")
    else:
        print("Agreement with ID " + AGREEMENT_ID + " is not found")


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DescribeAgreement Referência da API AWS SDK for Python (Boto3).

Obter o status de um contrato

O exemplo de código a seguir mostra como obter o status de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreement status
AG-13

Example Usage: python3 get_agreement_status.py --agreement-id <agreement-id>
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)


def get_agreement(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
        return response
    except ClientError as e:
        logger.error(f"Could not complete search_agreements request. {e}")

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
```

```
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement(agreement_id=args.agreement_id)

    if response is not None:
        print(f"Agreement status: {response['status']}")
    else:
        print(f"No agreement found for {args.agreement_id}")
```

- Para obter detalhes da API, consulte a DescribeAgreement Referência da API AWS SDK for Python (Boto3).

Obter os termos de suporte de um contrato

O exemplo de código a seguir mostra como obter os termos de suporte de um contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the support and refund policy I have provided to the customer for an
 agreement
AG-19

Example Usage: python3 get_agreement_support_terms.py --agreement-id <agreement-id>
```

```
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_terms(agreement_id):
    try:
        agreement = mp_client.get_agreement_terms(agreementId=agreement_id)
        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)

        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()
```

```
    agreement = get_agreement_terms(agreement_id=args.agreement_id)

    if agreement is not None:
        support_found = False

        for term in agreement["acceptedTerms"]:
            if "supportTerm" in term.keys():
                helper.pretty_print_datetime(term)
                support_found = True

        if not support_found:
            print(f"No support term found for agreement: {args.agreement_id}")
```

- Para obter detalhes da API, consulte a GetAgreementTerms Referência da API AWS SDK for Python (Boto3).

Listar solicitações de cancelamento de contratos

O exemplo de código a seguir mostra como listar solicitações de cancelamento de contratos dos quais participo como aceitante.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class ListAgreementCancellationRequests:
```

```
    PARTY_TYPE = "Proposer"

    @staticmethod
    def list_agreement_cancellation_requests():
        client = boto3.client("marketplace-agreement")

        next_token = None

        while True:
            kwargs = {"partyType": ListAgreementCancellationRequests.PARTY_TYPE}
            if next_token:
                kwargs["nextToken"] = next_token

            response = client.list_agreement_cancellation_requests(**kwargs)

            for summary in response.get("items", []):
                print("Cancellation Request ID: " +
 summary["agreementCancellationRequestId"])
                print("Agreement ID: " + summary["agreementId"])
                print("Status: " + str(summary.get("status", "")))
                print("Reason Code: " + str(summary.get("reasonCode", "")))
                print("Agreement Type: " + str(summary.get("agreementType", "")))
                print("Catalog: " + str(summary.get("catalog", "")))
                print("Created At: " + str(summary.get("createdAt", "")))
                print("Updated At: " + str(summary.get("updatedAt", "")))
                print("---")

            next_token = response.get("nextToken")
            if not next_token:
                break


if __name__ == "__main__":
    ListAgreementCancellationRequests.list_agreement_cancellation_requests()
```

- Para obter detalhes da API, consulte a ListAgreementCancellationRequests Referência da API AWS SDK for Python (Boto3).

Listar solicitações de pagamento de contratos

O exemplo de código a seguir mostra como listar solicitações de pagamento de contratos dos quais participo como aceitante.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class ListAgreementPaymentRequests:

    PARTY_TYPE = "Proposer"

    @staticmethod
    def list_agreement_payment_requests():
        client = boto3.client("marketplace-agreement")

        next_token = None

        while True:
            kwargs = {"partyType": ListAgreementPaymentRequests.PARTY_TYPE}
            if next_token:
                kwargs["nextToken"] = next_token

            response = client.list_agreement_payment_requests(**kwargs)

            for summary in response.get("items", []):
                print("Payment Request ID: " + summary["paymentRequestId"])
                print("Agreement ID: " + summary["agreementId"])
                print("Status: " + str(summary.get("status", "")))
                print("Name: " + str(summary.get("name", "")))
                print("Charge ID: " + str(summary.get("chargeId", "")))
                print("Charge Amount: " + str(summary.get("chargeAmount", "")))
                print("Currency Code: " + str(summary.get("currencyCode", "")))
```

```
                print("Created At: " + str(summary.get("createdAt", "")))
                print("Updated At: " + str(summary.get("updatedAt", "")))
                print("---")

            next_token = response.get("nextToken")
            if not next_token:
                break


if __name__ == "__main__":
    ListAgreementPaymentRequests.list_agreement_payment_requests()
```

- Para obter detalhes da API, consulte a ListAgreementPaymentRequests Referência da API AWS SDK for Python (Boto3).

Rejeitar uma solicitação de cancelamento de contrato

O exemplo de código a seguir mostra como rejeitar uma solicitação de cancelamento de contrato iniciada pelo vendedor.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class RejectAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
```

```
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID HERE>"
    REJECTION_REASON = "<REJECTION REASON HERE>"

    @staticmethod
    def reject_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.reject_agreement_cancellation_request(
            agreementId=RejectAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=RejectAgreementCancellationRequest.AGREEMENT_CANCELLATION_RE
            rejectionReason=RejectAgreementCancellationRequest.REJECTION_REASON,
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    RejectAgreementCancellationRequest.reject_agreement_cancellation_request()
```

- Para obter detalhes da API, consulte a RejectAgreementCancellationRequest Referência da API AWS SDK for Python (Boto3).

Rejeitar uma solicitação de pagamento de contrato

O exemplo de código a seguir mostra como rejeitar uma solicitação de pagamento de contrato iniciada pelo vendedor.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class RejectAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"
    REJECTION_REASON = "<REJECTION REASON HERE>"

    @staticmethod
    def reject_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.reject_agreement_payment_request(
            agreementId=RejectAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=RejectAgreementPaymentRequest.PAYMENT_REQUEST_ID,
            rejectionReason=RejectAgreementPaymentRequest.REJECTION_REASON,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))
```

```
if __name__ == "__main__":
    RejectAgreementPaymentRequest.reject_agreement_payment_request()
```

- Para obter detalhes da API, consulte a RejectAgreementPaymentRequest Referência da API AWS SDK for Python (Boto3).

Substitua um contrato de SaaS por uma oferta baseada em contrato

O exemplo de código a seguir mostra como substituir um contrato de SaaS por uma nova oferta baseada em contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing model
 with a new
Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
Agreement-Based Offer from the seller. The buyer replaces the existing agreement
with the new ABO offer, which may include updated pricing terms and payment
 schedule.

Note: Unlike other Replace samples, this example starts from an already-active
EXISTING_AGREEMENT_ID rather than creating a new agreement first. Use this
pattern when you already have an agreement ID and want to replace it directly.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offers:
```

```
  - EXISTING_AGREEMENT_ID — the agreement ID of the active agreement to replace
 (starts with agmt-).
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new ABO
 offer.
  - Term IDs (starting with term-) — found in the new offer's term list.
"""

import boto3

from utils.agreement_api_utils import generate_client_token


class ReplaceSaaSContractWithAgreementBasedOffer:

    # The agreement ID of the active SaaS agreement with CONTRACT pricing model to
 replace (starts with "agmt-").
    EXISTING_AGREEMENT_ID = "<your-existing-agreement-id>"

    # The agreementProposalId from the new Agreement-Based Offer.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-identifier>"

    # Term ID for the LegalTerm in the new offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the FixedUpfrontPricingTerm in the new offer.
    FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>"

    # Term ID for the PaymentScheduleTerm in the new offer.
    PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>"

    @staticmethod
    def replace_existing_agreement():
        """
        Replace an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.

        Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSContractWithAgreementBasedOffer
```

```
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        fixed_upfront_pricing_term = {"id": cls.FIXED_UPFRONT_PRICING_TERM_ID}
        payment_schedule_term = {"id": cls.PAYMENT_SCHEDULE_TERM_ID}

        # Replace the agreement with the new offer
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[fixed_upfront_pricing_term, payment_schedule_term,
 legal_term, validity_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=cls.EXISTING_AGREEMENT_ID,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Replace agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print("Agreement replaced with ABO. New AgreementId: " +
 accept_response["agreementId"])


if __name__ == "__main__":
    ReplaceSaaSContractWithAgreementBasedOffer.replace_existing_agreement()
```

- Para obter detalhes da API, consulte a CreateAgreementRequest Referência da API AWS SDK for Python (Boto3).

Substitua um teste gratuito de SaaS por um contrato com preços de consumo

O exemplo de código a seguir mostra como substituir um teste gratuito de SaaS por um Contrato com Preços de Consumo (CCP).

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create a SaaS free trial agreement and then replace it with a
paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.

Scenario: A buyer first starts a free trial on a SaaS product. Once the trial is
 active,
they decide to convert by replacing it with a paid offer that includes a
FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.

Flow:
  1. Create a SaaS free trial agreement with freeTrialPricingTerm.
  2. Wait for freeTrialPricingTerm agreement entitlements to become active.
  3. Replace the free trial agreement with the paid CCP offer using Intent.REPLACE.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the free trial
 offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the paid CCP
 offer.
  - Term IDs (starting with term-) — found in each offer's term list.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)
```

```
class ReplaceSaaSFreeTrialWithCCP:

    # The agreementProposalId from the free trial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-agreement-proposal-
identifier>"

    # Term ID for the FreeTrialPricingTerm in the free trial offer.
    FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>"

    # Term ID for the LegalTerm in the free trial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # The agreementProposalId from the paid CCP offer to convert to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-agreement-proposal-identifier>"

    # Term ID for the FixedUpfrontPricingTerm in the CCP offer.
    FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>"

    # Term ID for the PaymentScheduleTerm in the CCP offer.
    PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>"

    # Term ID for the UsageBasedPricingTerm in the CCP offer.
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the CCP offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the LegalTerm in the CCP offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    @staticmethod
    def create_saas_free_trial_and_replace_with_ccp():
        """
        Full end-to-end flow:
        1. Create a SaaS free trial agreement with freeTrialPricingTerm.
        2. Wait for freeTrialPricingTerm agreement entitlements to become active.
        3. Replace the free trial agreement with the paid CCP offer.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSFreeTrialWithCCP

        legal_term = {"id": cls.LEGAL_TERM_ID}
        free_trial_pricing_term = {"id": cls.FREE_TRIAL_PRICING_TERM_ID}
```

```
        # --- Step 1: Agreement with freeTrialPricingTerm ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[free_trial_pricing_term, legal_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with freeTrialPricingTerm created.
AgreementRequestId: " + agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with freeTrialPricingTerm accepted. AgreementId: "
+ agreement_id)

        # Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
        print("Waiting for freeTrialPricingTerm agreement entitlements to become
active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("freeTrialPricingTerm agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
        # Use Intent.REPLACE and sourceAgreementIdentifier to replace the free trial
agreement.
        usage_based_pricing_term = {"id": cls.USAGE_BASED_PRICING_TERM_ID}
        fixed_upfront_pricing_term = {"id": cls.FIXED_UPFRONT_PRICING_TERM_ID}
        payment_schedule_term = {"id": cls.PAYMENT_SCHEDULE_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[
                usage_based_pricing_term, fixed_upfront_pricing_term,
                payment_schedule_term, validity_term, new_legal_term,
            ],
```

```
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        print(
            "Agreement with freeTrialPricingTerm replaced with paid CCP offer. New
 AgreementId: "
            + aar_response["agreementId"]
        )


if __name__ == "__main__":
    ReplaceSaaSFreeTrialWithCCP.create_saas_free_trial_and_replace_with_ccp()
```

- Para obter detalhes da API, consulte a CreateAgreementRequest Referência da API AWS SDK for Python (Boto3).

Substitua um termo de preço baseado no uso do SaaS e cancele o contrato anterior

O exemplo de código a seguir mostra como substituir um termo de preço baseado no uso do SaaS e cancelar o contrato anterior.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT) and
 then replace it
with a new offer using the AWS Marketplace Agreement Service APIs.
```

```
Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
The buyer then converts to a different offer by replacing the existing agreement.

Flow:
  1. Create and accept the initial agreement request with UBPT.
  2. Wait for entitlements to become active.
  3. Replace the agreement with a new offer using Intent.REPLACE.
  4. Cancel the new agreement using CancelAgreement.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new offer
 to replace to.
  - Term IDs (starting with term-) — found in each offer's term list.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class ReplaceSaaSUsageBasedPricingTermAndCancel:

    # The agreementProposalId from the initial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the UsageBasedPricingTerm in the initial offer.
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the initial offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the LegalTerm in the initial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # The agreementProposalId from the new offer to replace to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-identifier>"
```

```
    # Term ID for the UsageBasedPricingTerm in the new offer.
    NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>"

    # Term ID for the LegalTerm in the new offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    @staticmethod
    def replace_saas_ubpt_and_cancel():
        """
        Full end-to-end flow:
        1. Create and accept the initial agreement request with
UsageBasedPricingTerm.
        2. Wait for entitlements to become active.
        3. Replace the agreement with a new offer.
        4. Cancel the new agreement.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSUsageBasedPricingTermAndCancel

        usage_based_pricing_term = {"id": cls.USAGE_BASED_PRICING_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}

        # --- Step 1: Create and accept the initial UBPT agreement request ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_based_pricing_term, validity_term, legal_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " + agreement_id)

        # Wait for entitlements to become active before replacing.
```

```
        print("Waiting for entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
 agreement_id)
        print("Entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Replace the UBPT agreement with a new offer ---
        # Use Intent.REPLACE and sourceAgreementIdentifier to replace the existing
 agreement.
        new_usage_based_pricing_term = {"id": cls.NEW_USAGE_BASED_PRICING_TERM_ID}
        new_validity_term = {"id": cls.NEW_VALIDITY_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[new_usage_based_pricing_term, new_validity_term,
 new_legal_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        agreement_id_with_new_offer = aar_response["agreementId"]
        print("UBPT agreement replaced. New AgreementId: " +
 agreement_id_with_new_offer)

        # --- Step 3: Cancel the new agreement ---
        client.cancel_agreement(agreementId=agreement_id_with_new_offer)
        print("The new agreement has been cancelled. AgreementId: " +
 agreement_id_with_new_offer)


if __name__ == "__main__":
    ReplaceSaaSUsageBasedPricingTermAndCancel.replace_saas_ubpt_and_cancel()
```

- Para obter detalhes da API, consulte a CreateAgreementRequest Referência da API AWS SDK for Python (Boto3).

Substitua um termo de preço baseado no uso da AMI, mas mantenha o ConfigurableUpfrontPricingTerm

O exemplo de código a seguir mostra como substituir um termo de preço baseado no uso da AMI e, ao mesmo tempo, manter o existente. ConfigurableUpfrontPricingTerm

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a new
 offer while
the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active, using the
AWS Marketplace Agreement Service APIs.

Scenario: An AMI product with USAGE pricing requires two agreements:
  1. An agreement with usageBasedPricingTerm — accepted first to establish the base
 agreement.
  2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
     usageBasedPricingTerm agreement entitlements are active.
This sample shows how to replace only the agreement with usageBasedPricingTerm with
 a
new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).

Flow:
  1. Create and accept the initial agreement request with usageBasedPricingTerm.
  2. Create and accept the agreement request with configurableUpfrontPricingTerm
 (CUPT)
     (after usageBasedPricingTerm agreement entitlements are active).
  3. Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial offer.
```

```
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new offer
 to replace to.
  - Term IDs (starting with term-) — found in each offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P365D).
  - DIMENSION_1_KEY — dimension key defined in the CUPT term.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class ReplaceAmiUsageBasedPricingTermButNotCupt:

    # The agreementProposalId from the initial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P365D" for 365 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # Dimension key defined in the CUPT term.
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the UsageBasedPricingTerm in the initial offer.
    USAGE_TERM_ID = "<your-usage-term-id>"

    # Term ID for the LegalTerm in the initial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in the initial offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # The agreementProposalId from the new offer to replace to.
```

```
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-identifier>"

    # Term ID for the UsageBasedPricingTerm in the new offer.
    NEW_USAGE_TERM_ID = "<your-new-usage-term-id>"

    # Term ID for the LegalTerm in the new offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>"

    @staticmethod
    def replace_ami_usage_when_cupt_exists():
        """
        Full end-to-end flow:
        1. Create and accept an agreement request with usageBasedPricingTerm, then
wait for entitlements.
        2. Create and accept an agreement request with CUPT, then wait for
entitlements.
        3. Replace the agreement with usageBasedPricingTerm with a new offer
           (agreement with CUPT is unaffected).
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceAmiUsageBasedPricingTermButNotCupt

        usage_term = {"id": cls.USAGE_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Step 1: Agreement with UBPT ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_term, legal_term, validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        usage_agreement_id = accept_response["agreementId"]
```

```
        print("Agreement request with UBPT accepted. AgreementId: " +
usage_agreement_id)

        # Wait for UBPT agreement entitlements to become active before creating the
agreement with CUPT.
        print("Waiting for UBPT agreement entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
usage_agreement_id)
        print("UBPT agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        print("Agreement request with CUPT created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        cupt_agreement_id = aar_response["agreementId"]
        print("Agreement request with CUPT accepted. AgreementId: " +
cupt_agreement_id)
```

```
        # Wait for CUPT agreement entitlements to become active before replacing
 usage.
        print("Waiting for CUPT agreement entitlements to become active...")
        cupt_entitlements_response = poll_until_entitlements_available(client,
 cupt_agreement_id)
        print("CUPT agreement entitlements are now active.")
        format_output(cupt_entitlements_response)

        # --- Step 3: Replace Agreement with usageBasedPricingTerm with a new offer
 ---
        new_usage_term = {"id": cls.NEW_USAGE_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}
        new_validity_term = {"id": cls.NEW_VALIDITY_TERM_ID}

        # Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
 usageBasedPricingTerm agreement only.
        # The agreement with configurableUpfrontPricingTerm (CUPT) is NOT affected
 by this replacement.
        car_replace_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[new_usage_term, new_legal_term, new_validity_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=usage_agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
 car_replace_response["agreementRequestId"])

        aar_replace_response = client.accept_agreement_request(
            agreementRequestId=car_replace_response["agreementRequestId"]
        )
        print("Agreement with UBPT replaced. New AgreementId: " +
 aar_replace_response["agreementId"])


if __name__ == "__main__":
    ReplaceAmiUsageBasedPricingTermButNotCupt.replace_ami_usage_when_cupt_exists()
```

- Para obter detalhes da API, consulte a CreateAgreementRequest Referência da API AWS SDK for Python (Boto3).

Pesquisar contratos por ID da conta

O exemplo de código a seguir mostra como pesquisar contratos por ID da conta.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get agreement by customer AWS
 account ID
AG-02
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")
logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10


def get_agreements(account_id):
    AgreementSummaryList = []
```

```
    try:
        agreement = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you are
 the proposer.
            # Change to "Acceptor" to return agreements where you are the acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": "AcceptorId", "values": [account_id]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    while "nextToken" in agreement and agreement["nextToken"] is not None:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreement["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {"name": "AcceptorId", "values": [account_id]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
```

```
        "--account_id",
        "-aid",
        help="Provide accepting account ID to search for agreements",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreements(account_id=args.account_id)

    helper.pretty_print_datetime(response)
```

- Para obter detalhes da API, consulte a SearchAgreements Referência da API AWS SDK for Python (Boto3).

Pesquisar contratos por ID do contrato

O exemplo de código a seguir mostra como pesquisar contratos por ID do contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreements give id
 information
AG-02-A
"""


import logging

import boto3
```

```
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# To search by offer id: OfferId; by product id: ResourceIdentifier; by product
 type: ResourceType
idType = "ResourceType"

# replace id value as needed
idValue = "SaaSProduct"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": idType, "values": [idValue]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
```

```
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": idType, "values": [idValue]},
                        {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise e

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a SearchAgreements Referência da API AWS SDK for Python (Boto3).

Pesquisar contratos por data de término

O exemplo de código a seguir mostra como pesquisar contratos por data de término.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreement information
 before or after end date
AG-03
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

# change to 'AfterEndTime' if after endtime is desired
beforeOrAfterEndtimeFilterName = "BeforeEndTime"

# Make sure to use the same date format as below
cutoffDate = "2322-11-18T00:00:00Z"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)
```

```
def get_agreements():
    AgreementSummaryList = []

    try:
        agreement = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you are
 the proposer.
            # Change to "Acceptor" to return agreements where you are the acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": beforeOrAfterEndtimeFilterName, "values": [cutoffDate]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise

    AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    while "nextToken" in agreement:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreement["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {
                        "name": beforeOrAfterEndtimeFilterName,
                        "values": [cutoffDate],
                    },
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList
```

```
if __name__ == "__main__":
    agreements = get_agreements()
    helper.pretty_print_datetime(agreements)
```

- Para obter detalhes da API, consulte a SearchAgreements Referência da API AWS SDK for Python (Boto3).

Pesquisar contratos por ID da oferta

O exemplo de código a seguir mostra como pesquisar contratos por ID da oferta.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreements by offer id
AG-0
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError
```

```
# offer id to search by
offerId = "11111111111111111111111111"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": "OfferId", "values": [offerId]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": "OfferId", "values": [offerId]},
                        {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
```

```
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement by offer id.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a SearchAgreements Referência da API AWS SDK for Python (Boto3).

Pesquisar contratos por ID do produto

O exemplo de código a seguir mostra como pesquisar contratos por ID do produto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreement by product
 id
AG-02
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# product id to search by
resourceId = "prod-1111111111111"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": "ResourceIdentifier", "values": [resourceId]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
```

```
            logger.error("Could not complete list_entities request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": "ResourceIdentifier", "values": [resourceId]},
                        {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a SearchAgreements Referência da API AWS SDK for Python (Boto3).

Pesquisar contratos por status

O exemplo de código a seguir mostra como pesquisar contratos por status.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to filter agreements by status
AG-04

Example Usage: python3 search_agreements_by_status.py
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)
```

```
MAX_PAGE_RESULTS = 10

# Set PartyType to "Proposer" to return agreements where you are the proposer.
# Change to "Acceptor" to return agreements where you are the acceptor.
party_type_list = ["Proposer"]
agreement_type_list = ["PurchaseAgreement"]

# Accepted values: "ACTIVE", "TERMINATED", "CANCELED", "EXPIRED", "REPLACED",
 "RENEWED"
status_list = ["ACTIVE"]

filter_list = [
    {"name": "PartyType", "values": party_type_list},
    {"name": "AgreementType", "values": agreement_type_list},
    {"name": "Status", "values": status_list},
]

agreement_results_list = []


def get_agreements(filter_list=filter_list):
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            filters=filter_list,
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    agreement_results_list.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=filter_list,
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e
```

```
        agreement_results_list.extend(agreements["agreementViewSummaries"])

    helper.pretty_print_datetime(agreement_results_list)
    return agreement_results_list


if __name__ == "__main__":
    agreements_list = get_agreements(filter_list)
```

- Para obter detalhes da API, consulte a SearchAgreements Referência da API AWS SDK for Python (Boto3).

Atualizar pedidos de compra após a aceitação do contrato

O exemplo de código a seguir mostra como atualizar pedidos de compra após a aceitação do contrato.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório da Biblioteca de códigos da referência de API do AWS Marketplace.

```
"""
Demonstrates how to associate a purchase order reference with a SaaS agreement with
 CONTRACT
pricing model using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model and
 provides
a purchase order reference in AcceptAgreementRequest. After acceptance, the buyer
 lists the
resulting charges via ListAgreementCharges and associates the purchase order
 reference with a
specific charge via UpdatePurchaseOrders.
```

```
Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement.
  - DIMENSION_1_KEY — dimension key defined in the offer.
  - PURCHASE_ORDER_REFERENCE — your internal purchase order number (e.g.,
 po-123456).
"""

import boto3

from utils.agreement_api_utils import format_output, generate_client_token


class UpdatePurchaseOrdersAfterAgreementAcceptance:

    # Your internal purchase order reference number (e.g., "po-123456").
    PURCHASE_ORDER_REFERENCE = "po-123456"

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P366D" for 366 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # Dimension key and quantity defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-key>"
    DIMENSION_1_VALUE = 1

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in your offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    @staticmethod
    def list_agreement_charges_and_update_purchase_orders():
        """
        Full end-to-end flow:
```

```
        1. Create a SaaS agreement with CONTRACT pricing model with a purchase order
reference.
        2. List charges to retrieve charge IDs and revisions.
        3. Associate the purchase order reference with a specific charge via
UpdatePurchaseOrders.
        4. List charges again to confirm the update.
        """
        client = boto3.client("marketplace-agreement")
        cls = UpdatePurchaseOrdersAfterAgreementAcceptance

        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Create Agreement ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)

        # --- Accept Agreement Request with Purchase Order ---
        # The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
        charge_id = create_response["chargeSummary"]["expectedCharges"][0]["id"]
```

```
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id,
            purchaseOrders=[
                {
                    "chargeId": charge_id,
                    "purchaseOrderReference": cls.PURCHASE_ORDER_REFERENCE,
                }
            ],
        )
        agreement_id = accept_response["agreementId"]
        print(
            "Agreement request accepted with purchase order reference '"
            + cls.PURCHASE_ORDER_REFERENCE + "'. AgreementId: " + agreement_id
        )

        # --- List Agreement Charges ---
        list_charges_response =
client.list_agreement_charges(agreementId=agreement_id)
        print("All charges for agreement " + agreement_id + ":")
        format_output(list_charges_response)

        # --- Update Purchase Order ---
        first_charge = list_charges_response["items"][0]
        client.update_purchase_orders(
            purchaseOrders=[
                {
                    "agreementId": agreement_id,
                    "purchaseOrderReference": cls.PURCHASE_ORDER_REFERENCE,
                    "chargeRevision": first_charge["revision"],
                    "chargeId": first_charge["id"],
                }
            ],
        )
        print(
            "Purchase order reference '" + cls.PURCHASE_ORDER_REFERENCE
            + "' updated for ChargeId: " + first_charge["id"]
        )

        # --- Verify Update ---
        lac_response = client.list_agreement_charges(agreementId=agreement_id)
        print("Verified updated charge:")
        format_output(lac_response["items"][0])
```

```
if __name__ == "__main__":

 UpdatePurchaseOrdersAfterAgreementAcceptance.list_agreement_charges_and_update_purchase_ord
```

- Para obter detalhes da API, consulte a UpdatePurchaseOrders Referência da API AWS SDK for Python (Boto3).

Exemplos do Amazon MSK usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon MSK.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Python.

```
import base64

def lambda_handler(event, context):
    # Iterate through keys
    for key in event['records']:
        print('Key:', key)
        # Iterate through records
        for record in event['records'][key]:
            print('Record:', record)
            # Decode base64
            msg = base64.b64decode(record['value']).decode('utf-8')
            print('Message:', msg)
```

Exemplos do Neptune usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with Neptune.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Neptune

O exemplo de código a seguir mostra como começar a usar o Neptune.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
from botocore.exceptions import ClientError


def describe_db_clusters(neptune_client):
    """
    Describes the Amazon Neptune DB clusters using a paginator to handle multiple
 pages.
    Raises ClientError with 'ResourceNotFoundException' if no clusters are found.
    """
    paginator = neptune_client.get_paginator("describe_db_clusters")
    clusters_found = False

    for page in paginator.paginate():
        for cluster in page.get("DBClusters", []):
            clusters_found = True
            print(f"Cluster Identifier: {cluster['DBClusterIdentifier']}")
            print(f"Status: {cluster['Status']}")

    if not clusters_found:
        raise ClientError(
            {
                "Error": {
                    "Code": "ResourceNotFoundException",
                    "Message": "No Neptune DB clusters found."
                }
            },
            operation_name="DescribeDBClusters"
        )

def main():
    """
    Main entry point: creates the Neptune client and calls the describe operation.
    """
```

```
    neptune_client = boto3.client("neptune")
    try:
        describe_db_clusters(neptune_client)
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "ResourceNotFoundException":
            print(f"Resource not found: {e.response['Error']['Message']}")
        else:
            print(f"Unexpected ClientError: {e.response['Error']['Message']}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a DescribeDBClustersPaginator Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de sub-redes do Amazon Neptune.
- Criar um cluster do Neptune.
- Criar uma instância do Neptune.
- Conferir o status da instância do Neptune.
- Mostrar detalhes do cluster do Neptune.
- Interromper o cluster do Neptune.
- Iniciar o cluster do Neptune.
- Excluir os ativos do Neptune.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
import time
from botocore.exceptions import ClientError

# Constants used in this scenario
POLL_INTERVAL_SECONDS = 10
TIMEOUT_SECONDS = 1200  # 20 minutes

def delete_db_cluster(neptune_client, cluster_id: str):
    """
    Deletes a Neptune DB cluster and throws exceptions to the caller.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        cluster_id (str): The ID of the Neptune DB cluster to be deleted.

    Raises:
        ClientError: If the delete operation fails.
    """
    request = {
        'DBClusterIdentifier': cluster_id,
        'SkipFinalSnapshot': True
    }

    try:
        print(f"Deleting DB Cluster: {cluster_id}")
        neptune_client.delete_db_cluster(**request)

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBClusterNotFoundFault":
            print(f"Cluster '{cluster_id}' not found or already deleted.")
```

```
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB cluster. {code}: {message}")
        raise

def format_elapsed_time(seconds: int) -> str:
    mins, secs = divmod(seconds, 60)
    hours, mins = divmod(mins, 60)
    return f"{hours:02}:{mins:02}:{secs:02}"


def delete_db_instance(neptune_client, instance_id: str):
    """
    Deletes a Neptune DB instance and waits for its deletion to complete.
    Raises exception to be handled by calling code.
    """
    print(f"Initiating deletion of DB Instance: {instance_id}")
    try:
        neptune_client.delete_db_instance(
            DBInstanceIdentifier=instance_id,
            SkipFinalSnapshot=True
        )

        print(f"Waiting for DB Instance '{instance_id}' to be deleted...")
        waiter = neptune_client.get_waiter('db_instance_deleted')
        waiter.wait(
            DBInstanceIdentifier=instance_id,
            WaiterConfig={
                'Delay': 30,
                'MaxAttempts': 40
            }
        )

        print(f"DB Instance '{instance_id}' successfully deleted.")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBInstanceNotFoundFault":
            print(f"Instance '{instance_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
```

```
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB instance. {code}: {message}")
        raise

def delete_db_subnet_group(neptune_client, subnet_group_name):
    """
    Deletes a Neptune DB subnet group synchronously using Boto3.

    Args:
        neptune_client (boto3.client): The Neptune client.
        subnet_group_name (str): The name of the DB subnet group to delete.

    Raises:
        ClientError: If the delete operation fails.
    """
    delete_group_request = {
        'DBSubnetGroupName': subnet_group_name
    }

    try:
        neptune_client.delete_db_subnet_group(**delete_group_request)
        print(f"# Deleting Subnet Group: {subnet_group_name}")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found or already
 deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete subnet group. {code}: {message}")
        raise

def wait_for_cluster_status(
        neptune_client,
        cluster_id: str,
        desired_status: str,
        timeout_seconds: int = TIMEOUT_SECONDS,
```

```
        poll_interval_seconds: int = POLL_INTERVAL_SECONDS
):
    """
    Waits for a Neptune DB cluster to reach a desired status.

    Args:
        neptune_client (boto3.client): The Amazon Neptune client.
        cluster_id (str): The identifier of the Neptune DB cluster.
        desired_status (str): The target status (e.g., "available", "stopped").
        timeout_seconds (int): Max time to wait in seconds (default: 1200).
        poll_interval_seconds (int): Polling interval in seconds (default: 10).

    Raises:
        RuntimeError: If the desired status is not reached before timeout.
    """
    print(f"Waiting for cluster '{cluster_id}' to reach status
'{desired_status}'...")
    start_time = time.time()

    while True:
        # Prepare request object
        describe_cluster_request = {
            'DBClusterIdentifier': cluster_id
        }

        # Call the Neptune API
        response = neptune_client.describe_db_clusters(**describe_cluster_request)
        clusters = response.get('DBClusters', [])
        current_status = clusters[0].get('Status') if clusters else None
        elapsed_seconds = int(time.time() - start_time)

        status_str = current_status if current_status else "Unknown"
        print(
            f"\r Elapsed: {format_elapsed_time(elapsed_seconds):<20}  Cluster
status: {status_str:<20}",
            end="", flush=True
        )

        if current_status and current_status.lower() == desired_status.lower():
            print(
                f"\nNeptune cluster reached desired status '{desired_status}' after
{format_elapsed_time(elapsed_seconds)}."
            )
            return
```

```
        if elapsed_seconds > timeout_seconds:
            raise RuntimeError(f"Timeout waiting for Neptune cluster to reach
 status: {desired_status}")

        time.sleep(poll_interval_seconds)


def start_db_cluster(neptune_client, cluster_identifier: str):
    """
    Starts an Amazon Neptune DB cluster and waits until it reaches 'available'.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: Propagates AWS API issues like resource not found.
        RuntimeError: If cluster doesn't reach 'available' within timeout.
    """
    try:
        # Initial wait in case the cluster was just stopped
        time.sleep(30)
        neptune_client.start_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't start DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
```

```
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling. It
 may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
 {message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s – Cluster status: {status}", end="",
 flush=True)

        if status and status.lower() == 'available':
            print(f"\n# Cluster '{cluster_identifier}' is available.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster '{cluster_identifier}'
 to become available.")

        time.sleep(POLL_INTERVAL_SECONDS)


def stop_db_cluster(neptune_client, cluster_identifier: str):
    """
    Stops an Amazon Neptune DB cluster and waits until it's fully stopped.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: For AWS API errors (e.g., resource not found).
        RuntimeError: If the cluster doesn't stop within the timeout.
    """
    try:
        neptune_client.stop_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
```

```
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't stop DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling. It
may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'stopped':
            print(f"\nCluster '{cluster_identifier}' is now stopped.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster '{cluster_identifier}'
to stop.")
```

```
        time.sleep(POLL_INTERVAL_SECONDS)



def describe_db_clusters(neptune_client, cluster_id: str):
    """
    Describes details of a Neptune DB cluster, paginating if needed.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_id (str): The ID of the cluster to describe.

    Raises:
        ClientError: If there's an AWS API error (e.g., cluster not found).
    """
    paginator = neptune_client.get_paginator('describe_db_clusters')

    try:
        pages = paginator.paginate(DBClusterIdentifier=cluster_id)

        found = False
        for page in pages:
            for cluster in page.get('DBClusters', []):
                found = True
                print(f"Cluster Identifier: {cluster.get('DBClusterIdentifier')}")
                print(f"Status: {cluster.get('Status')}")
                print(f"Engine: {cluster.get('Engine')}")
                print(f"Engine Version: {cluster.get('EngineVersion')}")
                print(f"Endpoint: {cluster.get('Endpoint')}")
                print(f"Reader Endpoint: {cluster.get('ReaderEndpoint')}")
                print(f"Availability Zones: {cluster.get('AvailabilityZones')}")
                print(f"Subnet Group: {cluster.get('DBSubnetGroup')}")
                print("VPC Security Groups:")
                for vpc_group in cluster.get('VpcSecurityGroups', []):
                    print(f"  - {vpc_group.get('VpcSecurityGroupId')}")
                print(f"Storage Encrypted: {cluster.get('StorageEncrypted')}")
                print(f"IAM Auth Enabled:
 {cluster.get('IAMDatabaseAuthenticationEnabled')}")
                print(f"Backup Retention Period:
 {cluster.get('BackupRetentionPeriod')} days")
                print(f"Preferred Backup Window:
 {cluster.get('PreferredBackupWindow')}")
```

```
                print(f"Preferred Maintenance Window:
 {cluster.get('PreferredMaintenanceWindow')}")
                print("------")

        if not found:
            # Treat empty response as cluster not found
            raise ClientError(
                {"Error": {"Code": "DBClusterNotFound", "Message": f"No cluster
 found with ID '{cluster_id}'"}},
                "DescribeDBClusters"
            )

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        elif code == "DBClusterNotFound":
            print(f"Cluster '{cluster_id}' not found. Please verify the cluster
 ID.")
        else:
            print(f"Couldn't describe DB cluster. Here's why: {code}: {message}")
        raise

def check_instance_status(neptune_client, instance_id: str, desired_status: str):
    """
    Polls the status of a Neptune DB instance until it reaches desired_status.
    Uses pagination via describe_db_instances — even for a single instance.

    Raises:
      ClientError: If describe_db_instances fails (e.g., instance not found).
      RuntimeError: If timeout expires before reaching desired status.
    """
    paginator = neptune_client.get_paginator('describe_db_instances')
    start_time = time.time()

    while True:
        try:
            pages = paginator.paginate(DBInstanceIdentifier=instance_id)
            instances = []
            for page in pages:
                instances.extend(page.get('DBInstances', []))
```

```
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBInstanceNotFound":
                print(f"Instance '{instance_id}' not found. Please verify the
 instance ID.")
            else:
                print(f"Failed to describe DB instance. {code}: {message}")
            raise

        current_status = instances[0].get('DBInstanceStatus') if instances else None
        elapsed = int(time.time() - start_time)

        print(f"\rElapsed: {format_elapsed_time(elapsed)}  Status:
 {current_status}", end="", flush=True)

        if current_status and current_status.lower() == desired_status.lower():
            print(f"\nInstance '{instance_id}' reached '{desired_status}' in
 {format_elapsed_time(elapsed)}.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for '{instance_id}' to reach
 '{desired_status}'")

        time.sleep(POLL_INTERVAL_SECONDS)


def create_db_instance(neptune_client, db_instance_id: str, db_cluster_id: str) ->
 str:
    try:
        request = {
            'DBInstanceIdentifier': db_instance_id,
            'DBInstanceClass': 'db.r5.large',
            'Engine': 'neptune',
            'DBClusterIdentifier': db_cluster_id
        }

        print(f"Creating Neptune DB Instance: {db_instance_id}")
        response = neptune_client.create_db_instance(**request)

        instance = response.get('DBInstance')
```

```
        if not instance or 'DBInstanceIdentifier' not in instance:
            raise RuntimeError("Instance creation succeeded but no ID returned.")

        print(f"Waiting for DB Instance '{db_instance_id}' to become available...")
        waiter = neptune_client.get_waiter('db_instance_available')
        waiter.wait(
            DBInstanceIdentifier=db_instance_id,
            WaiterConfig={'Delay': 30, 'MaxAttempts': 40}
        )

        print(f"DB Instance '{db_instance_id}' is now available.")
        return instance['DBInstanceIdentifier']

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't create DB instance. Here's why: {code}: {message}")
        raise

    except Exception as e:
        print(f"Unexpected error creating DB instance '{db_instance_id}': {e}")
        raise RuntimeError(f"Unexpected error creating DB instance
 '{db_instance_id}': {e}") from e


def create_db_cluster(neptune_client, db_name: str) -> str:
    """
    Creates a Neptune DB cluster and returns its identifier.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        db_name (str): The desired cluster identifier.

    Returns:
        str: The DB cluster identifier.

    Raises:
        RuntimeError: For any failure or AWS error, with a user-friendly message.
    """
```

```
    request = {
        'DBClusterIdentifier': db_name,
        'Engine': 'neptune',
        'DeletionProtection': False,
        'BackupRetentionPeriod': 1
    }

    try:
        response = neptune_client.create_db_cluster(**request)
        cluster = response.get('DBCluster') or {}

        cluster_id = cluster.get('DBClusterIdentifier')
        if not cluster_id:
            raise RuntimeError("Cluster created but no ID returned.")

        print(f"DB Cluster created: {cluster_id}")
        return cluster_id

    except ClientError as e:
        code = e.response["Error"]["Code"]
        message = e.response["Error"]["Message"]

        if code in ("ServiceQuotaExceededException", "DBClusterQuotaExceededFault"):
            raise RuntimeError("You have exceeded the quota for Neptune DB
 clusters.") from e
        else:
            raise RuntimeError(f"AWS error [{code}]: {message}") from e

    except Exception as e:
        raise RuntimeError(f"Unexpected error creating DB cluster '{db_name}': {e}")
 from e

def get_subnet_ids(vpc_id: str) -> list[str]:
    ec2_client = boto3.client('ec2')

    describe_subnets_request = {
        'Filters': [{'Name': 'vpc-id', 'Values': [vpc_id]}]
    }

    response = ec2_client.describe_subnets(**describe_subnets_request)
    subnets = response.get('Subnets', [])
    subnet_ids = [subnet['SubnetId'] for subnet in subnets if 'SubnetId' in subnet]
    return subnet_ids
```

```
def get_default_vpc_id() -> str:
    ec2_client = boto3.client('ec2')
    describe_vpcs_request = {
        'Filters': [{'Name': 'isDefault', 'Values': ['true']}]
    }

    response = ec2_client.describe_vpcs(**describe_vpcs_request)
    vpcs = response.get('Vpcs', [])
    if not vpcs:
        raise RuntimeError("No default VPC found in this region.")

    default_vpc_id = vpcs[0]['VpcId']
    print(f"Default VPC ID: {default_vpc_id}")
    return default_vpc_id


def create_subnet_group(neptune_client, group_name: str):
    """
    Creates a Neptune DB subnet group and returns its name and ARN.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        group_name (str): The desired name of the subnet group.

    Returns:
        tuple(str, str): (subnet_group_name, subnet_group_arn)

    Raises:
        RuntimeError: For quota errors or other AWS-related failures.
    """
    vpc_id = get_default_vpc_id()
    subnet_ids = get_subnet_ids(vpc_id)

    request = {
        'DBSubnetGroupName': group_name,
        'DBSubnetGroupDescription': 'My Neptune subnet group',
        'SubnetIds': subnet_ids,
        'Tags': [{'Key': 'Environment', 'Value': 'Dev'}]
    }

    try:
        response = neptune_client.create_db_subnet_group(**request)
        sg = response.get("DBSubnetGroup", {})
```

```
        name = sg.get("DBSubnetGroupName")
        arn = sg.get("DBSubnetGroupArn")

        if not name or not arn:
            raise RuntimeError("Response missing subnet group name or ARN.")

        print(f"Subnet group created: {name}")
        print(f"ARN: {arn}")
        return name, arn

    except ClientError as e:
        code = e.response["Error"]["Code"]
        msg = e.response["Error"]["Message"]

        if code == "ServiceQuotaExceededException":
            print("Subnet group quota exceeded.")
            raise RuntimeError("Subnet group quota exceeded.") from e
        else:
            print(f"AWS error [{code}]: {msg}")
            raise RuntimeError(f"AWS error [{code}]: {msg}") from e

    except Exception as e:
        print(f"Unexpected error creating subnet group '{group_name}': {e}")
        raise RuntimeError(f"Unexpected error creating subnet group '{group_name}':
 {e}") from e

def wait_for_input_to_continue():
    input("\nPress <ENTER> to continue...")
    print("Continuing with the program...\n")


def run_scenario(neptune_client, subnet_group_name: str, db_instance_id: str,
 cluster_name: str):
    print("-" * 88)
    print("1. Create a Neptune DB Subnet Group")
    wait_for_input_to_continue()

    try:
        name, arn = create_subnet_group(neptune_client, subnet_group_name)
        print(f"Subnet group successfully created: {name}")

        print("-" * 88)
        print("2. Create a Neptune Cluster")
        wait_for_input_to_continue()
```

```
        db_cluster_id = create_db_cluster(neptune_client, cluster_name)

        print("-" * 88)
        print("3. Create a Neptune DB Instance")
        wait_for_input_to_continue()
        create_db_instance(neptune_client, db_instance_id, cluster_name)

        print("-" * 88)
        print("4. Check the status of the Neptune DB Instance")
        print("""
        Even though you're targeting a single DB instance,
        describe_db_instances supports pagination and can return multiple pages.

        Handling paginated responses ensures your method continues to work reliably
        even if AWS returns large or paged results.
        """)
        wait_for_input_to_continue()
        check_instance_status(neptune_client, db_instance_id, "available")

        print("-" * 88)
        print("5. Show Neptune Cluster details")
        wait_for_input_to_continue()
        describe_db_clusters(neptune_client, db_cluster_id)

        print("-" * 88)
        print("6. Stop the Amazon Neptune cluster")
        print("""
            Boto3 doesn't currently offer a
            built-in waiter for stop_db_cluster,
            This example implements a custom polling
            strategy until the cluster is in a stopped state.
        """)
        wait_for_input_to_continue()
        stop_db_cluster(neptune_client, db_cluster_id)
        check_instance_status(neptune_client, db_instance_id, "stopped")

        print("-" * 88)
        print("7. Start the Amazon Neptune cluster")
        print("""
            Boto3 doesn't currently offer a
            built-in waiter for start_db_cluster,
            This example implements a custom polling
            strategy until the cluster is in an available state.
        """)
```

```
        wait_for_input_to_continue()
        start_db_cluster(neptune_client, db_cluster_id)
        wait_for_cluster_status(neptune_client, db_cluster_id, "available")
        check_instance_status(neptune_client, db_instance_id, "available")

        print("All Neptune resources are now available.")
        print("-" * 88)

        print("-" * 88)
        print("8. Delete the Neptune Assets")
        print("Would you like to delete the Neptune Assets? (y/n)")
        del_ans = input().strip().lower()

        if del_ans == "y":
            print("You selected to delete the Neptune assets.")

            delete_db_instance(neptune_client, db_instance_id)
            delete_db_cluster(neptune_client, db_cluster_id)
            delete_db_subnet_group(neptune_client, subnet_group_name)

            print("Neptune resources deleted successfully")

    except ClientError as ce:
        code = ce.response["Error"]["Code"]

        if code in ("DBInstanceNotFound", "DBInstanceNotFoundFault",
"ResourceNotFound"):
            print(f"Instance '{db_instance_id}' not found.")
        elif code in ("DBClusterNotFound", "DBClusterNotFoundFault",
"ResourceNotFoundFault"):
            print(f"Cluster '{cluster_name}' not found.")
        elif code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"AWS error [{code}]: {ce.response['Error']['Message']}")
            raise  # re-raise unexpected errors

    except RuntimeError as re:
        print(f"Runtime error or timeout: {re}")
```

```
def main():
    neptune_client = boto3.client('neptune')

    # Customize the following names to match your Neptune setup
    # (You must change these to unique values for your environment)
    subnet_group_name = "neptuneSubnetGroup111"
    cluster_name = "neptuneCluster111"
    db_instance_id = "neptuneDB111"

    print("""
    Amazon Neptune is a fully managed graph database service by AWS...
    Let's get started!
    """)
    wait_for_input_to_continue()
    run_scenario(neptune_client, subnet_group_name, db_instance_id, cluster_name)

    print("""
    Thank you for checking out the Amazon Neptune Service Use demo.
    For more AWS code examples, visit:
    https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-library.html
    """)

if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDBCluster
  - CreateDBInstance
  - CreateDBSubnetGroup
  - CreateGraph
  - DeleteDBCluster
  - DeleteDBInstance
  - DeleteDBSubnetGroup
  - DescribeDBClusters
  - DescribeDBInstances
  - ExecuteGremlinProfileQuery
  - ExecuteGremlinQuery

- ExecuteOpenCypherExplainQuery
- ExecuteQuery
- Iniciar cluster de banco de dados
- Interrompa o cluster de banco

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_db_cluster(neptune_client, db_name: str) -> str:
    """
    Creates a Neptune DB cluster and returns its identifier.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        db_name (str): The desired cluster identifier.

    Returns:
        str: The DB cluster identifier.

    Raises:
        RuntimeError: For any failure or AWS error, with a user-friendly message.
    """
    request = {
        'DBClusterIdentifier': db_name,
        'Engine': 'neptune',
        'DeletionProtection': False,
        'BackupRetentionPeriod': 1
    }
```

```
    try:
        response = neptune_client.create_db_cluster(**request)
        cluster = response.get('DBCluster') or {}

        cluster_id = cluster.get('DBClusterIdentifier')
        if not cluster_id:
            raise RuntimeError("Cluster created but no ID returned.")

        print(f"DB Cluster created: {cluster_id}")
        return cluster_id

    except ClientError as e:
        code = e.response["Error"]["Code"]
        message = e.response["Error"]["Message"]

        if code in ("ServiceQuotaExceededException", "DBClusterQuotaExceededFault"):
            raise RuntimeError("You have exceeded the quota for Neptune DB
clusters.") from e
        else:
            raise RuntimeError(f"AWS error [{code}]: {message}") from e

    except Exception as e:
        raise RuntimeError(f"Unexpected error creating DB cluster '{db_name}': {e}")
from e
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Python (Boto3).

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_db_instance(neptune_client, db_instance_id: str, db_cluster_id: str) ->
 str:
    try:
        request = {
            'DBInstanceIdentifier': db_instance_id,
            'DBInstanceClass': 'db.r5.large',
            'Engine': 'neptune',
            'DBClusterIdentifier': db_cluster_id
        }

        print(f"Creating Neptune DB Instance: {db_instance_id}")
        response = neptune_client.create_db_instance(**request)

        instance = response.get('DBInstance')
        if not instance or 'DBInstanceIdentifier' not in instance:
            raise RuntimeError("Instance creation succeeded but no ID returned.")

        print(f"Waiting for DB Instance '{db_instance_id}' to become available...")
        waiter = neptune_client.get_waiter('db_instance_available')
        waiter.wait(
            DBInstanceIdentifier=db_instance_id,
            WaiterConfig={'Delay': 30, 'MaxAttempts': 40}
        )

        print(f"DB Instance '{db_instance_id}' is now available.")
        return instance['DBInstanceIdentifier']

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't create DB instance. Here's why: {code}: {message}")
        raise

    except Exception as e:
        print(f"Unexpected error creating DB instance '{db_instance_id}': {e}")
        raise RuntimeError(f"Unexpected error creating DB instance
 '{db_instance_id}': {e}") from e
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Python (Boto3).

`CreateDBSubnetGroup`

O código de exemplo a seguir mostra como usar `CreateDBSubnetGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_subnet_group(neptune_client, group_name: str):
    """
    Creates a Neptune DB subnet group and returns its name and ARN.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        group_name (str): The desired name of the subnet group.

    Returns:
        tuple(str, str): (subnet_group_name, subnet_group_arn)

    Raises:
        RuntimeError: For quota errors or other AWS-related failures.
    """
    vpc_id = get_default_vpc_id()
    subnet_ids = get_subnet_ids(vpc_id)

    request = {
        'DBSubnetGroupName': group_name,
        'DBSubnetGroupDescription': 'My Neptune subnet group',
        'SubnetIds': subnet_ids,
        'Tags': [{'Key': 'Environment', 'Value': 'Dev'}]
    }
```

```
    try:
        response = neptune_client.create_db_subnet_group(**request)
        sg = response.get("DBSubnetGroup", {})
        name = sg.get("DBSubnetGroupName")
        arn = sg.get("DBSubnetGroupArn")

        if not name or not arn:
            raise RuntimeError("Response missing subnet group name or ARN.")

        print(f"Subnet group created: {name}")
        print(f"ARN: {arn}")
        return name, arn

    except ClientError as e:
        code = e.response["Error"]["Code"]
        msg = e.response["Error"]["Message"]

        if code == "ServiceQuotaExceededException":
            print("Subnet group quota exceeded.")
            raise RuntimeError("Subnet group quota exceeded.") from e
        else:
            print(f"AWS error [{code}]: {msg}")
            raise RuntimeError(f"AWS error [{code}]: {msg}") from e

    except Exception as e:
        print(f"Unexpected error creating subnet group '{group_name}': {e}")
        raise RuntimeError(f"Unexpected error creating subnet group '{group_name}':
{e}") from e
```

- Para obter detalhes da API, consulte a CreateDBSubnetGroup Referência da API AWS SDK for Python (Boto3).

`CreateGraph`

O código de exemplo a seguir mostra como usar `CreateGraph`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a **peered
 VPC**

"""

GRAPH_NAME = "sample-analytics-graph"

def main():
    config = Config(retries={"total_max_attempts": 1, "mode": "standard"},
 read_timeout=None)
    client = boto3.client("neptune-graph", config=config)
    execute_create_graph(client, GRAPH_NAME)

def execute_create_graph(client, graph_name):
    try:
        print("Creating Neptune graph...")
        response = client.create_graph(
            graphName=graph_name,
            provisionedMemory = 16
        )

        created_graph_name = response.get("name")
        graph_arn = response.get("arn")
```

```
        graph_endpoint = response.get("endpoint")

        print("Graph created successfully!")
        print(f"Graph Name: {created_graph_name}")
        print(f"Graph ARN: {graph_arn}")
        print(f"Graph Endpoint: {graph_endpoint}")

    except ClientError as e:
        print(f"Failed to create graph: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"Failed to create graph: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a CreateGraph Referência da API AWS SDK for Python (Boto3).

`DeleteDBCluster`

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_db_cluster(neptune_client, cluster_id: str):
    """
    Deletes a Neptune DB cluster and throws exceptions to the caller.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        cluster_id (str): The ID of the Neptune DB cluster to be deleted.
```

```
    Raises:
        ClientError: If the delete operation fails.
    """
    request = {
        'DBClusterIdentifier': cluster_id,
        'SkipFinalSnapshot': True
    }

    try:
        print(f"Deleting DB Cluster: {cluster_id}")
        neptune_client.delete_db_cluster(**request)

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBClusterNotFoundFault":
            print(f"Cluster '{cluster_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't delete DB cluster. {code}: {message}")
        raise
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Python (Boto3).

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_db_instance(neptune_client, instance_id: str):
    """
    Deletes a Neptune DB instance and waits for its deletion to complete.
    Raises exception to be handled by calling code.
    """
    print(f"Initiating deletion of DB Instance: {instance_id}")
    try:
        neptune_client.delete_db_instance(
            DBInstanceIdentifier=instance_id,
            SkipFinalSnapshot=True
        )

        print(f"Waiting for DB Instance '{instance_id}' to be deleted...")
        waiter = neptune_client.get_waiter('db_instance_deleted')
        waiter.wait(
            DBInstanceIdentifier=instance_id,
            WaiterConfig={
                'Delay': 30,
                'MaxAttempts': 40
            }
        )

        print(f"DB Instance '{instance_id}' successfully deleted.")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBInstanceNotFoundFault":
            print(f"Instance '{instance_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB instance. {code}: {message}")
        raise
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Python (Boto3).

`DeleteDBSubnetGroup`

O código de exemplo a seguir mostra como usar `DeleteDBSubnetGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_db_subnet_group(neptune_client, subnet_group_name):
    """
    Deletes a Neptune DB subnet group synchronously using Boto3.

    Args:
        neptune_client (boto3.client): The Neptune client.
        subnet_group_name (str): The name of the DB subnet group to delete.

    Raises:
        ClientError: If the delete operation fails.
    """
    delete_group_request = {
        'DBSubnetGroupName': subnet_group_name
    }

    try:
        neptune_client.delete_db_subnet_group(**delete_group_request)
        print(f"# Deleting Subnet Group: {subnet_group_name}")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found or already
 deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete subnet group. {code}: {message}")
```

```
        raise
```

- Para obter detalhes da API, consulte a DeleteDBSubnetGroup Referência da API AWS SDK for Python (Boto3).

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def describe_db_clusters(neptune_client, cluster_id: str):
    """
    Describes details of a Neptune DB cluster, paginating if needed.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_id (str): The ID of the cluster to describe.

    Raises:
        ClientError: If there's an AWS API error (e.g., cluster not found).
    """
    paginator = neptune_client.get_paginator('describe_db_clusters')

    try:
        pages = paginator.paginate(DBClusterIdentifier=cluster_id)

        found = False
        for page in pages:
            for cluster in page.get('DBClusters', []):
                found = True
                print(f"Cluster Identifier: {cluster.get('DBClusterIdentifier')}")
                print(f"Status: {cluster.get('Status')}")
                print(f"Engine: {cluster.get('Engine')}")
```

```
                print(f"Engine Version: {cluster.get('EngineVersion')}")
                print(f"Endpoint: {cluster.get('Endpoint')}")
                print(f"Reader Endpoint: {cluster.get('ReaderEndpoint')}")
                print(f"Availability Zones: {cluster.get('AvailabilityZones')}")
                print(f"Subnet Group: {cluster.get('DBSubnetGroup')}")
                print("VPC Security Groups:")
                for vpc_group in cluster.get('VpcSecurityGroups', []):
                    print(f"  - {vpc_group.get('VpcSecurityGroupId')}")
                print(f"Storage Encrypted: {cluster.get('StorageEncrypted')}")
                print(f"IAM Auth Enabled:
{cluster.get('IAMDatabaseAuthenticationEnabled')}")
                print(f"Backup Retention Period:
{cluster.get('BackupRetentionPeriod')} days")
                print(f"Preferred Backup Window:
{cluster.get('PreferredBackupWindow')}")
                print(f"Preferred Maintenance Window:
{cluster.get('PreferredMaintenanceWindow')}")
                print("------")

        if not found:
            # Treat empty response as cluster not found
            raise ClientError(
                {"Error": {"Code": "DBClusterNotFound", "Message": f"No cluster
found with ID '{cluster_id}'"}},
                "DescribeDBClusters"
            )

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        elif code == "DBClusterNotFound":
            print(f"Cluster '{cluster_id}' not found. Please verify the cluster
ID.")
        else:
            print(f"Couldn't describe DB cluster. Here's why: {code}: {message}")
        raise
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Python (Boto3).

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def check_instance_status(neptune_client, instance_id: str, desired_status: str):
    """
    Polls the status of a Neptune DB instance until it reaches desired_status.
    Uses pagination via describe_db_instances — even for a single instance.

    Raises:
      ClientError: If describe_db_instances fails (e.g., instance not found).
      RuntimeError: If timeout expires before reaching desired status.
    """
    paginator = neptune_client.get_paginator('describe_db_instances')
    start_time = time.time()

    while True:
        try:
            pages = paginator.paginate(DBInstanceIdentifier=instance_id)
            instances = []
            for page in pages:
                instances.extend(page.get('DBInstances', []))

        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBInstanceNotFound":
                print(f"Instance '{instance_id}' not found. Please verify the
 instance ID.")
            else:
```

```
                print(f"Failed to describe DB instance. {code}: {message}")
            raise

        current_status = instances[0].get('DBInstanceStatus') if instances else None
        elapsed = int(time.time() - start_time)

        print(f"\rElapsed: {format_elapsed_time(elapsed)}  Status:
{current_status}", end="", flush=True)

        if current_status and current_status.lower() == desired_status.lower():
            print(f"\nInstance '{instance_id}' reached '{desired_status}' in
{format_elapsed_time(elapsed)}.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for '{instance_id}' to reach
'{desired_status}'")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

`ExecuteGremlinProfileQuery`

O código de exemplo a seguir mostra como usar `ExecuteGremlinProfileQuery`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Replace this with your actual Neptune endpoint
NEPTUNE_ENDPOINT = "https://[Specify Endpoint]:8182"

def main():
```

```
    """
    Entry point of the program. Initializes the Neptune client and executes the
 Gremlin query.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    execute_gremlin_query(neptune_client)


def execute_gremlin_query(neptune_client):
    """
    Executes a Gremlin query against an Amazon Neptune database.
    """
    try:
        print("Querying Neptune...")

        response = neptune_client.execute_gremlin_explain_query(
            gremlinQuery="g.V().has('code', 'ANC')"
        )

        print("Full Response:")
        print(response['output'].read().decode('UTF-8'))

    except ClientError as e:
        print(f"Error calling Neptune: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ExecuteGremlinProfileQuery Referência da API AWS SDK for Python (Boto3).

`ExecuteGremlinQuery`

O código de exemplo a seguir mostra como usar `ExecuteGremlinQuery`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a **peered
 VPC**

"""

# Replace with your actual Neptune endpoint
NEPTUNE_ENDPOINT = "https://[Specify-Your-Endpoint]:8182"

def main():
    """
    Entry point of the program. Initializes the Neptune client and runs both EXPLAIN
 and PROFILE queries.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})
```

```
    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    try:
        run_profile_query(neptune_client)
    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

def run_profile_query(neptune_client):
    """
    Runs a PROFILE query on the Neptune graph database.
    """
    print("Running Gremlin PROFILE query...")

    try:
        response = neptune_client.execute_gremlin_profile_query(
            gremlinQuery="g.V().has('code', 'ANC')"
        )
        print("Profile Query Result:")
        output = response.get("output")
        if output:
            print(output.read().decode('utf-8'))
        else:
            print("No explain output returned.")
    except Exception as e:
        print(f"Failed to execute PROFILE query: {str(e)}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ExecuteGremlinQuery Referência da API AWS SDK for Python (Boto3).

ExecuteOpenCypherExplainQuery

O código de exemplo a seguir mostra como usar `ExecuteOpenCypherExplainQuery`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Replace with your actual Neptune endpoint URL
NEPTUNE_ENDPOINT = "https://<your-neptune-endpoint>:8182"

def main():
    """
    Entry point: Create Neptune client and execute different OpenCypher queries.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    execute_open_cypher_query_without_params(neptune_client)
    execute_open_cypher_query_with_params(neptune_client)
    execute_open_cypher_explain_query(neptune_client)

def execute_open_cypher_query_without_params(client):
    """
    Executes a simple OpenCypher query without parameters.
    """
    try:
        print("\nRunning OpenCypher query without parameters...")
        resp = client.execute_open_cypher_query(
            openCypherQuery="MATCH (n {code: 'ANC'}) RETURN n"
        )
        print("Results:")
```

```
        print(resp['results'])

    except Exception as e:
        print(f"Error in simple OpenCypher query: {str(e)}")


def execute_open_cypher_query_with_params(client):
    """
    Executes an OpenCypher query using parameters.
    """
    try:
        print("\nRunning OpenCypher query with parameters...")
        parameters = {'code': 'ANC'}
        resp = client.execute_open_cypher_query(
            openCypherQuery="MATCH (n {code: $code}) RETURN n",
            parameters=json.dumps(parameters)
        )
        print("Results:")
        print(resp['results'])

    except Exception as e:
        print(f"Error in parameterized OpenCypher query: {str(e)}")

def execute_open_cypher_explain_query(client):
    """
    Runs an OpenCypher EXPLAIN query in debug mode.
    """
    try:
        print("\nRunning OpenCypher EXPLAIN query (debug mode)...")
        resp = client.execute_open_cypher_explain_query(
            openCypherQuery="MATCH (n {code: 'ANC'}) RETURN n",
            explainMode="details"
        )
        results = resp.get('results')
        if results is None:
            print("No explain results returned.")
        else:
            try:
                print("Explain Results:")
                print(results.read().decode('UTF-8'))
            except Exception as e:
                print(f"Error in OpenCypher EXPLAIN query: {str(e)}")

    except ClientError as e:
```

```
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ExecuteOpenCypherExplainQuery Referência da API AWS SDK for Python (Boto3).

`ExecuteQuery`

O código de exemplo a seguir mostra como usar `ExecuteQuery`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a **peered
 VPC**
"""
```

```
GRAPH_ID = "<your-graph-id>"

def main():
    config = Config(retries={"total_max_attempts": 1, "mode": "standard"},
 read_timeout=None)
    client = boto3.client("neptune-graph", config=config)

    try:
        print("\n--- Running OpenCypher query without parameters ---")
        run_open_cypher_query(client, GRAPH_ID)

        print("\n--- Running OpenCypher query with parameters ---")
        run_open_cypher_query_with_params(client, GRAPH_ID)

        print("\n--- Running OpenCypher explain query ---")
        run_open_cypher_explain_query(client, GRAPH_ID)

    except Exception as e:
        print(f"Unexpected error in main: {e}")

def run_open_cypher_query(client, graph_id):
    """
    Run an OpenCypher query without parameters.
    """
    try:
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: 'ANC'}) RETURN n",
            language='OPEN_CYPHER'
        )
        print(resp['payload'].read().decode('UTF-8'))

    except client.exceptions.InternalServerException as e:
        print(f"InternalServerException: {e.response['Error']['Message']}")
    except ClientError as e:
        print(f"ClientError: {e.response['Error']['Message']}")
    except Exception as e:  # <--- ADD THIS BLOCK
        print(f"Unexpected error: {e}")

def run_open_cypher_query_with_params(client, graph_id):
    """
    Run an OpenCypher query with parameters.
    """
    try:
```

```
        parameters = {'code': 'ANC'}
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: $code}) RETURN n",
            language='OPEN_CYPHER',
            parameters=parameters
        )
        print(resp['payload'].read().decode('UTF-8'))

    except client.exceptions.InternalServerException as e:
        print(f"InternalServerException: {e.response['Error']['Message']}")
    except ClientError as e:
        print(f"ClientError: {e.response['Error']['Message']}")
    except Exception as e:  # <--- ADD THIS BLOCK
        print(f"Unexpected error: {e}")

def run_open_cypher_explain_query(client, graph_id):
    """
    Run an OpenCypher explain query (explainMode = "debug").
    """
    try:
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: 'ANC'}) RETURN n",
            language='OPEN_CYPHER',
            explainMode='DETAILS'
        )
        print(resp['payload'].read().decode('UTF-8'))

    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"Unexpected Boto3 error: {str(e)}")
    except Exception as e:  # <-- Add this generic catch
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte a ExecuteQuery Referência da API AWS SDK for Python (Boto3).

StartDBCluster

O código de exemplo a seguir mostra como usar `StartDBCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def start_db_cluster(neptune_client, cluster_identifier: str):
    """
    Starts an Amazon Neptune DB cluster and waits until it reaches 'available'.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: Propagates AWS API issues like resource not found.
        RuntimeError: If cluster doesn't reach 'available' within timeout.
    """
    try:
        # Initial wait in case the cluster was just stopped
        time.sleep(30)
        neptune_client.start_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't start DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
```

```
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling. It
may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s – Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'available':
            print(f"\n# Cluster '{cluster_identifier}' is available.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster '{cluster_identifier}'
to become available.")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Consulte detalhes da API em StartDBCluster na Referência de API do AWS SDK para Python (Boto3).

`StopDBCluster`

O código de exemplo a seguir mostra como usar `StopDBCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def stop_db_cluster(neptune_client, cluster_identifier: str):
    """
    Stops an Amazon Neptune DB cluster and waits until it's fully stopped.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: For AWS API errors (e.g., resource not found).
        RuntimeError: If the cluster doesn't stop within the timeout.
    """
    try:
        neptune_client.stop_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't stop DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
```

```
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling. It
may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s – Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'stopped':
            print(f"\nCluster '{cluster_identifier}' is now stopped.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster '{cluster_identifier}'
to stop.")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Consulte detalhes da API em StopDBCluster na Referência de API do AWS SDK para Python (Boto3).

Exemplos do Organizations usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with Organizations.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AttachPolicy`

O código de exemplo a seguir mostra como usar `AttachPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def attach_policy(policy_id, target_id, orgs_client):
    """
    Attaches a policy to a target. The target is an organization root, account, or
    organizational unit.

    :param policy_id: The ID of the policy to attach.
    :param target_id: The ID of the resources to attach the policy to.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.attach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Attached policy %s to target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't attach policy %s to target %s.", policy_id, target_id
        )
        raise
```

- Para obter detalhes da API, consulte a AttachPolicy Referência da API AWS SDK for Python (Boto3).

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_policy(name, description, content, policy_type, orgs_client):
    """
    Creates a policy.

    :param name: The name of the policy.
    :param description: The description of the policy.
    :param content: The policy content as a dict. This is converted to JSON before
                    it is sent to AWS. The specific format depends on the policy
 type.
    :param policy_type: The type of the policy.
    :param orgs_client: The Boto3 Organizations client.
    :return: The newly created policy.
    """
    try:
        response = orgs_client.create_policy(
            Name=name,
            Description=description,
            Content=json.dumps(content),
            Type=policy_type,
        )
        policy = response["Policy"]
        logger.info("Created policy %s.", name)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- Para obter detalhes da API, consulte a CreatePolicy Referência da API AWS SDK for Python (Boto3).

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_policy(policy_id, orgs_client):
    """
    Deletes a policy.

    :param policy_id: The ID of the policy to delete.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.delete_policy(PolicyId=policy_id)
        logger.info("Deleted policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_id)
        raise
```

- Para obter detalhes da API, consulte a DeletePolicy Referência da API AWS SDK for Python (Boto3).

`DescribePolicy`

O código de exemplo a seguir mostra como usar `DescribePolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def describe_policy(policy_id, orgs_client):
    """
    Describes a policy.

    :param policy_id: The ID of the policy to describe.
    :param orgs_client: The Boto3 Organizations client.
    :return: The description of the policy.
    """
    try:
        response = orgs_client.describe_policy(PolicyId=policy_id)
        policy = response["Policy"]
        logger.info("Got policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't get policy %s.", policy_id)
        raise
    else:
        return policy
```

- Para obter detalhes da API, consulte a DescribePolicy Referência da API AWS SDK for Python (Boto3).

`DetachPolicy`

O código de exemplo a seguir mostra como usar `DetachPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def detach_policy(policy_id, target_id, orgs_client):
    """
    Detaches a policy from a target.

    :param policy_id: The ID of the policy to detach.
    :param target_id: The ID of the resource where the policy is currently attached.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.detach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Detached policy %s from target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from target %s.", policy_id, target_id
        )
        raise
```

- Para obter detalhes da API, consulte a DetachPolicy Referência da API AWS SDK for Python (Boto3).

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_policies(policy_filter, orgs_client):
    """
    Lists the policies for the account, limited to the specified filter.

    :param policy_filter: The kind of policies to return.
    :param orgs_client: The Boto3 Organizations client.
    :return: The list of policies found.
    """
    try:
        response = orgs_client.list_policies(Filter=policy_filter)
        policies = response["Policies"]
        logger.info("Found %s %s policies.", len(policies), policy_filter)
    except ClientError:
        logger.exception("Couldn't get %s policies.", policy_filter)
        raise
    else:
        return policies
```

- Para obter detalhes da API, consulte a ListPolicies Referência da API AWS SDK for Python (Boto3).

Exemplos da Central de Parceiros usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Partner Central.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AssignOpportunity`

O código de exemplo a seguir mostra como usar `AssignOpportunity`.

SDK para Python (Boto3)

Reatribua uma oportunidade existente a outro usuário.

```
#!/usr/bin/env python

"""
Purpose
PC-API-07 Assigning a new owner
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)
```

```
def assign_opportunity(identifier):
    assign_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "Assignee": {
            "BusinessTitle": "OpportunityOwner",
            "Email": "test@test.com",
            "FirstName": "John",
            "LastName": "Doe"
        }
    }
    try:
        # Perform an API call
        response =
 partner_central_client.assign_opportunity(**assign_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O4236468"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Assigning a new owner to an opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(assign_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a AssignOpportunity Referência da API AWS SDK for Python (Boto3).

`AssociateOpportunity`

O código de exemplo a seguir mostra como usar `AssociateOpportunity`.

SDK para Python (Boto3)

Crie uma associação formal entre uma oportunidade e várias entidades relacionadas.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def associate_opportunity(entity_type, entity_identifier, opportunityIdentifier):
    associate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)
```

```
def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0059717"
    opportunityIdentifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Associate Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(associate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a AssociateOpportunity Referência da API AWS SDK for Python (Boto3).

`CreateOpportunity`

O código de exemplo a seguir mostra como usar `CreateOpportunity`.

SDK para Python (Boto3)

Crie uma oportunidade.

```
#!/usr/bin/env python
import boto3
import logging
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
import utils.stringify_details as sd
from botocore.client import ClientError
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"
```

```
def create_opportunity(partner_central_client):
    create_opportunity_request = helper.remove_nulls(sd.stringify_json("src/
create_opportunity/createOpportunity.json"))
    try:
        # Perform an API call
        response =
 partner_central_client.create_opportunity(**create_opportunity_request)

        helper.pretty_print_datetime(response)

        # Retrieve the opportunity details
        get_response = partner_central_client.get_opportunity(
            Identifier=response["Id"],
            Catalog=CATALOG_TO_USE
        )
        helper.pretty_print_datetime(get_response)
        return response
    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Create Opportunity.")
    print("-" * 88)

    partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
    )

    create_opportunity(partner_central_client)

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a CreateOpportunity Referência da API AWS SDK for Python (Boto3).

`DisassociateOpportunity`

O código de exemplo a seguir mostra como usar `DisassociateOpportunity`.

SDK para Python (Boto3)

Remova uma associação existente entre uma oportunidade e entidades relacionadas.

```
#!/usr/bin/env python

"""
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def disassociate_opportunity(entity_type, entity_identifier, opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        return response
```

```
    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0049999"
    opportunityIdentifier = "O4397574"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(disassociate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a DisassociateOpportunity Referência da API AWS SDK for Python (Boto3).

`GetAwsOpportunitySummary`

O código de exemplo a seguir mostra como usar `GetAwsOpportunitySummary`.

SDK para Python (Boto3)

Recupera um resumo de uma AWS oportunidade.

```
#!/usr/bin/env python

"""
Purpose
PC-API-25 Retrieves a summary of an AWS Opportunity. LifeCycle.ReviewStatus=Approved
"""
import logging
import boto3
```

```
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "RelatedOpportunityIdentifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_aws_opportunity_summary(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get AWS Opportunity summary.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetAwsOpportunitySummary Referência da API AWS SDK for Python (Boto3).

GetEngagementInvitation

O código de exemplo a seguir mostra como usar `GetEngagementInvitation`.

SDK para Python (Boto3)

Recupera os detalhes de um convite de noivado compartilhado AWS com um parceiro.

```
#!/usr/bin/env python

"""
Purpose
PC-API-22  GetOpportunityEngagementInvitation - Retrieves details of a specific
 engagement invitation.
This operation allows partners to view the invitation and its associated
 information,
such as the customer, project, and lifecycle details.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity_engagement_invitation(identifier):
    get_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
```

```
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_engagement_invitation_re
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral-selling:us-east-1:aws:catalog/Sandbox/
engagement-invitation/engi-0000000IS0Qga"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Given the ARN identifier, retrieve details of Opportunity Engagement
 Invitation.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity_engagement_invitation(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetEngagementInvitation Referência da API AWS SDK for Python (Boto3).

`GetOpportunity`

O código de exemplo a seguir mostra como usar `GetOpportunity`.

SDK para Python (Boto3)

Tenha uma oportunidade.

```
#!/usr/bin/env python

"""
Purpose
PC-API -08 Get updated Opportunity given opportunity id
"""
```

```
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
        response = partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a GetOpportunity Referência da API AWS SDK for Python (Boto3).

`ListEngagementInvitations`

O código de exemplo a seguir mostra como usar `ListEngagementInvitations`.

SDK para Python (Boto3)

Recupera uma lista de convites de engajamento enviados ao parceiro.

```
#!/usr/bin/env python

"""
Purpose
PC-API-21 ListEngagementInvitations - Retrieves a list of engagement invitations
 based on specified criteria.
This operation allows partners to view all invitations to engagement.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def list_engagement_invitations():
    list_engagement_invitations_request ={
        "Catalog": CATALOG_TO_USE,
        "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_engagement_invitations(**list_engagement_invitations_request)
```

```
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Retrieve list of Engagement Invitations.")
    print("-" * 88)

    helper.pretty_print_datetime(list_engagement_invitations())

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a ListEngagementInvitations Referência da API AWS SDK for Python (Boto3).

`ListOpportunities`

O código de exemplo a seguir mostra como usar `ListOpportunities`.

SDK para Python (Boto3)

Liste oportunidades.

```
#!/usr/bin/env python

"""
Purpose
PC-API -18 Getting list of Opportunities
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE
```

```
serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_opportunities():

    opportunity_list = []

    list_opportunities_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_opportunities(**list_opportunities_request)
        opportunity_list.extend(response["OpportunitySummaries"])

        while "NextToken" in response and response["NextToken"] is not None:
            list_opportunities_request["NextToken"] = response["NextToken"]
            response =
 partner_central_client.list_opportunities(**list_opportunities_request)
            opportunity_list.extend(response["OpportunitySummaries"])

        return opportunity_list

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Getting list of Opportunities.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_opportunities())

if __name__ == "__main__":
```

```
    usage_demo()
```

- Para obter detalhes da API, consulte a ListOpportunities Referência da API AWS SDK for Python (Boto3).

ListSolutions

O código de exemplo a seguir mostra como usar `ListSolutions`.

SDK para Python (Boto3)

Recupera uma lista de soluções de parceiros que o parceiro registrou na Central de Parceiros.

```
#!/usr/bin/env python

"""
Purpose
PC-API-10 Getting list of solutions
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_solutions():
    list_solutions_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
    }
    try:
        # Perform an API call
        response = partner_central_client.list_solutions(**list_solutions_request)
```

```
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Getting list of solutions.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_solutions())

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a ListSolutions Referência da API AWS SDK for Python (Boto3).

`RejectEngagementInvitation`

O código de exemplo a seguir mostra como usar `RejectEngagementInvitation`.

SDK para Python (Boto3)

Rejeita e EngagementInvitation isso é AWS compartilhado.

```
#!/usr/bin/env python

"""
Purpose
PC-API-05 AWS Originated AO rejection - RejectOpportunityEngagementInvitation -
 Rejects a engagement invitation.
This action indicates that the partner does not wish to participate in the
 engagement and
provides a reason for the rejection.
Upon rejection, a OpportunityEngagementInvitationRejected event is triggered.
Subsequently, the invitation will no longer be available for the partner to act on.
"""
```

```
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def reject_opportunity_engagement_invitation(identifier, reject_reason):
    reject_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "RejectionReason": reject_reason
    }
    try:
        # Perform an API call
        response =
 partner_central_client.reject_engagement_invitation(**reject_opportunity_engagement_invitat
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isviga"
    reject_reason = "Customer problem unclear"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Given the ARN identifier and reject reason, reject the Opportunity
 Engagement Invitation.")
    print("-" * 88)
```

```
 helper.pretty_print_datetime(reject_opportunity_engagement_invitation(identifier,
 reject_reason))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a RejectEngagementInvitation Referência da API AWS SDK for Python (Boto3).

StartEngagementByAcceptingInvitationTask

O código de exemplo a seguir mostra como usar `StartEngagementByAcceptingInvitationTask`.

SDK para Python (Boto3)

Inicia o noivado aceitando um EngagementInvitation.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)
```

```
def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def start_engagement_by_accepting_invitation_task(identifier):

    response = get_opportunity(identifier)

    if ( response['Status'] == 'PENDING') :
        accept_opportunity_engagement_invitation_request ={
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-123456"
        }
        try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_by_accepting_invitation_task(**accept_opportunity_e
            return response

        except ClientError as err:
            # Catch all client exceptions
            print(err.response)
            return None
    else:
        return None

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isusga"
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
```

```
    print("Get updated Opportunity.")
    print("-" * 88)


 helper.pretty_print_datetime(start_engagement_by_accepting_invitation_task(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a StartEngagementByAcceptingInvitationTask Referência da API AWS SDK for Python (Boto3).

`StartEngagementFromOpportunityTask`

O código de exemplo a seguir mostra como usar `StartEngagementFromOpportunityTask`.

SDK para Python (Boto3)

Inicia o processo de engajamento com base em uma oportunidade existente aceitando o convite de engajamento e criando uma oportunidade correspondente no sistema do parceiro.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
```

```
        region_name='us-east-1'
)

def start_engagement_from_opportunity_task(identifier):

    start_engagement_from_opportunity_task_request ={
            "AwsSubmission": {
                "InvolvementType": "Co-Sell",
                "Visibility": "Full"
            },
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-annjqwesdsd99"
    }
    try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_from_opportunity_task(**start_engagement_from_oppor
            return response

    except ClientError as err:
            # Catch all client exceptions
            print(err.response)
            return None

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Start Engagement from Opportunity Task.")
    print("-" * 88)

    helper.pretty_print_datetime(start_engagement_from_opportunity_task(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a StartEngagementFromOpportunityTask Referência da API AWS SDK for Python (Boto3).

`UpdateOpportunity`

O código de exemplo a seguir mostra como usar `UpdateOpportunity`.

SDK para Python (Boto3)

Atualize uma oportunidade.

```
#!/usr/bin/env python

"""
Purpose
PC-API-2  Updating Partner Originated Opportunity
"""
import logging
import boto3
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
from botocore.client import ClientError
import utils.stringify_details as sd
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response = partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
```

```
        print(err.response)

def update_opportunity():
    update_opportunity_request_orig = sd.stringify_json("src/update_opportunity/
update_opportunity_technical_validation.json")
    update_opportunity_request =
 helper.remove_nulls(update_opportunity_request_orig)

    try:
        # Perform an API call
        response =
 partner_central_client.update_opportunity(**update_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def update_opportunity_if_eligible(identifier):
    response = get_opportunity(identifier)
    if response is not None:
        return update_opportunity()
    else:
        print("Failed to retrieve opportunity details")

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Updating opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(update_opportunity_if_eligible(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Para obter detalhes da API, consulte a UpdateOpportunity Referência da API AWS SDK for Python (Boto3).

Cenários

Atualizar a entidade associada de uma oportunidade

O exemplo de código a seguir mostra como:

- Desassociar uma entidade antiga.
- Associar uma entidade nova.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Atualizar a entidade associada de uma oportunidade

```
#!/usr/bin/env python

"""
Purpose
PC-API -17 Replacing a solution
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def replace_solution(original_entity_identifier, new_entity_identifier,
 opportunityIdentifier):
```

```
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : original_entity_identifier
    }

    associate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : new_entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    original_entity_identifier = "S-0049999"
    new_entity_identifier = "S-0050014"
    opportunityIdentifier = "O4397574"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Replacing a solution.")
    print("-" * 88)

    helper.pretty_print_datetime(replace_solution(original_entity_identifier,
 new_entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Python (Boto3).
  - AssociateOpportunity
  - DisassociateOpportunity

Exemplos do Amazon Pinpoint usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendMessages`

O código de exemplo a seguir mostra como usar `SendMessages`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
import logging
import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_email_message(
    pinpoint_client,
    app_id,
    sender,
    to_addresses,
    char_set,
    subject,
    html_message,
    text_message,
):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project ID to use when you send this message.
    :param sender: The "From" address. This address must be verified in
                   Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
 account
                         is in the sandbox, these addresses must be verified.
    :param char_set: The character encoding to use for the subject line and message
                     body of the email.
    :param subject: The subject line of the email.
    :param html_message: The body of the email for recipients whose email clients
 can
                         display HTML content.
    :param text_message: The body of the email for recipients whose email clients
                         don't support HTML content.
    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                "Addresses": {
                    to_address: {"ChannelType": "EMAIL"} for to_address in
 to_addresses
                },
                "MessageConfiguration": {
```

```
                    "EmailMessage": {
                        "FromAddress": sender,
                        "SimpleEmail": {
                            "Subject": {"Charset": char_set, "Data": subject},
                            "HtmlPart": {"Charset": char_set, "Data": html_message},
                            "TextPart": {"Charset": char_set, "Data": text_message},
                        },
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
    else:
        return {
            to_address: message["MessageId"]
            for to_address, message in response["MessageResponse"]["Result"].items()
        }


def main():
    app_id = "ce796be37f32f178af652b26eexample"
    sender = "sender@example.com"
    to_address = "recipient@example.com"
    char_set = "UTF-8"
    subject = "Amazon Pinpoint Test (SDK for Python (Boto3))"
    text_message = """Amazon Pinpoint Test (SDK for Python)
    -------------------------------------
    This email was sent with Amazon Pinpoint using the AWS SDK for Python (Boto3).
    For more information, see https://aws.amazon.com/sdk-for-python/
                """
    html_message = """<html>
    <head></head>
    <body>
      <h1>Amazon Pinpoint Test (SDK for Python (Boto3)</h1>
      <p>This email was sent with
        <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> using the
        <a href='https://aws.amazon.com/sdk-for-python/'>
          AWS SDK for Python (Boto3)</a>.</p>
    </body>
    </html>
                """
```

```
    print("Sending email.")
    message_ids = send_email_message(
        boto3.client("pinpoint"),
        app_id,
        sender,
        [to_address],
        char_set,
        subject,
        html_message,
        text_message,
    )
    print(f"Message sent! Message IDs: {message_ids}")


if __name__ == "__main__":
    main()
```

Envie uma mensagem SMS.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_sms_message(
    pinpoint_client,
    app_id,
    origination_number,
    destination_number,
    message,
    message_type,
):
    """
    Sends an SMS message with Amazon Pinpoint.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project/application ID to use when you send
                   this message. The SMS channel must be enabled for the project or
                   application.
```

```
    :param destination_number: The recipient's phone number in E.164 format.
    :param origination_number: The phone number to send the message from. This phone
                               number must be associated with your Amazon Pinpoint
                               account and be in E.164 format.
    :param message: The content of the SMS message.
    :param message_type: The type of SMS message that you want to send. If you send
                         time-sensitive content, specify TRANSACTIONAL. If you send
                         marketing-related content, specify PROMOTIONAL.
    :return: The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                "Addresses": {destination_number: {"ChannelType": "SMS"}},
                "MessageConfiguration": {
                    "SMSMessage": {
                        "Body": message,
                        "MessageType": message_type,
                        "OriginationNumber": origination_number,
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response["MessageResponse"]["Result"][destination_number]
["MessageId"]


def main():
    app_id = "ce796be37f32f178af652b26eexample"
    origination_number = "+12065550199"
    destination_number = "+14255550142"
    message = (
        "This is a sample message sent from Amazon Pinpoint by using the AWS SDK for
 "
        "Python (Boto 3)."
    )
    message_type = "TRANSACTIONAL"

    print("Sending SMS message.")
```

```
    message_id = send_sms_message(
        boto3.client("pinpoint"),
        app_id,
        origination_number,
        destination_number,
        message,
        message_type,
    )
    print(f"Message sent! Message ID: {message_id}.")


if __name__ == "__main__":
    main()
```

Enviar uma mensagem com um modelo de e-mail existente.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_templated_email_message(
    pinpoint_client, project_id, sender, to_addresses, template_name,
 template_version
):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: The Amazon Pinpoint project ID to use when you send this
 message.
    :param sender: The "From" address. This address must be verified in
                   Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
                         account is in the sandbox, these addresses must be
 verified.
    :param template_name: The name of the email template to use when sending the
 message.
    :param template_version: The version number of the message template.
```

```
    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
            MessageRequest={
                "Addresses": {
                    to_address: {"ChannelType": "EMAIL"} for to_address in
 to_addresses
                },
                "MessageConfiguration": {"EmailMessage": {"FromAddress": sender}},
                "TemplateConfiguration": {
                    "EmailTemplate": {
                        "Name": template_name,
                        "Version": template_version,
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
    else:
        return {
            to_address: message["MessageId"]
            for to_address, message in response["MessageResponse"]["Result"].items()
        }


def main():
    project_id = "296b04b342374fceb661bf494example"
    sender = "sender@example.com"
    to_addresses = ["recipient@example.com"]
    template_name = "My_Email_Template"
    template_version = "1"

    print("Sending email.")
    message_ids = send_templated_email_message(
        boto3.client("pinpoint"),
        project_id,
        sender,
        to_addresses,
        template_name,
        template_version,
```

```
    )
    print(f"Message sent! Message IDs: {message_ids}")


if __name__ == "__main__":
    main()
```

Envie uma mensagem de texto com um modelo de SMS existente.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_templated_sms_message(
    pinpoint_client,
    project_id,
    destination_number,
    message_type,
    origination_number,
    template_name,
    template_version,
):
    """
    Sends an SMS message to a specific phone number using a pre-defined template.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: An Amazon Pinpoint project (application) ID.
    :param destination_number: The phone number to send the message to.
    :param message_type: The type of SMS message (promotional or transactional).
    :param origination_number: The phone number that the message is sent from.
    :param template_name: The name of the SMS template to use when sending the
 message.
    :param template_version: The version number of the message template.

    :return The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
```

```
            MessageRequest={
                "Addresses": {destination_number: {"ChannelType": "SMS"}},
                "MessageConfiguration": {
                    "SMSMessage": {
                        "MessageType": message_type,
                        "OriginationNumber": origination_number,
                    }
                },
                "TemplateConfiguration": {
                    "SMSTemplate": {"Name": template_name, "Version":
 template_version}
                },
            },
        )

    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response["MessageResponse"]["Result"][destination_number]
["MessageId"]


def main():
    region = "us-east-1"
    origination_number = "+18555550001"
    destination_number = "+14255550142"
    project_id = "7353f53e6885409fa32d07cedexample"
    message_type = "TRANSACTIONAL"
    template_name = "My_SMS_Template"
    template_version = "1"
    message_id = send_templated_sms_message(
        boto3.client("pinpoint", region_name=region),
        project_id,
        destination_number,
        message_type,
        origination_number,
        template_name,
        template_version,
    )
    print(f"Message sent! Message ID: {message_id}.")


if __name__ == "__main__":
```

```
    main()
```

- Para obter detalhes da API, consulte a SendMessages Referência da API AWS SDK for Python (Boto3).

Exemplos da API SMS and Voice do Amazon Pinpoint usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API AWS SDK para Python (Boto3) de SMS e voz do Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendVoiceMessage`

O código de exemplo a seguir mostra como usar `SendVoiceMessage`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import logging
import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_voice_message(
    sms_voice_client,
    origination_number,
    caller_id,
    destination_number,
    language_code,
    voice_id,
    ssml_message,
):
    """
    Sends a voice message using speech synthesis provided by Amazon Polly.

    :param sms_voice_client: A Boto3 PinpointSMSVoice client.
    :param origination_number: The phone number that the message is sent from.
                               The phone number must be associated with your Amazon
                               Pinpoint account and be in E.164 format.
    :param caller_id: The phone number that you want to appear on the recipient's
                      device. The phone number must be associated with your Amazon
                      Pinpoint account and be in E.164 format.
    :param destination_number: The recipient's phone number. Specify the phone
                               number in E.164 format.
    :param language_code: The language to use when sending the message.
    :param voice_id: The Amazon Polly voice that you want to use to send the
 message.
    :param ssml_message: The content of the message. This example uses SSML to
 control
                         certain aspects of the message, such as the volume and the
                         speech rate. The message must not contain line breaks.
    :return: The ID of the message.
    """
    try:
        response = sms_voice_client.send_voice_message(
            DestinationPhoneNumber=destination_number,
            OriginationPhoneNumber=origination_number,
            CallerId=caller_id,
            Content={
                "SSMLMessage": {
                    "LanguageCode": language_code,
                    "VoiceId": voice_id,
```

```
                    "Text": ssml_message,
                }
            },
        )
    except ClientError:
        logger.exception(
            "Couldn't send message from %s to %s.",
            origination_number,
            destination_number,
        )
        raise
    else:
        return response["MessageId"]


def main():
    origination_number = "+12065550110"
    caller_id = "+12065550199"
    destination_number = "+12065550142"
    language_code = "en-US"
    voice_id = "Matthew"
    ssml_message = (
        "<speak>"
        "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "
        "using the <break strength='weak'/>AWS SDK for Python (Boto3). "
        "<amazon:effect phonation='soft'>Thank you for listening."
        "</amazon:effect>"
        "</speak>"
    )
    print(f"Sending voice message from {origination_number} to
 {destination_number}.")
    message_id = send_voice_message(
        boto3.client("pinpoint-sms-voice"),
        origination_number,
        caller_id,
        destination_number,
        language_code,
        voice_id,
        ssml_message,
    )
    print(f"Message sent!\nMessage ID: {message_id}")


if __name__ == "__main__":
```

```
    main()
```

- Para obter detalhes da API, consulte a SendVoiceMessage Referência da API AWS SDK for Python (Boto3).

Exemplos do Amazon Polly usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Polly.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DescribeVoices`

O código de exemplo a seguir mostra como usar `DescribeVoices`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def describe_voices(self):
        """
        Gets metadata about available voices.

        :return: The list of voice metadata.
        """
        try:
            response = self.polly_client.describe_voices()
            self.voice_metadata = response["Voices"]
            logger.info("Got metadata about %s voices.", len(self.voice_metadata))
        except ClientError:
            logger.exception("Couldn't get voice metadata.")
            raise
        else:
            return self.voice_metadata
```

- Para obter detalhes da API, consulte a DescribeVoices Referência da API AWS SDK for Python (Boto3).

`GetLexicon`

O código de exemplo a seguir mostra como usar `GetLexicon`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def get_lexicon(self, name):
        """
        Gets metadata and contents of an existing lexicon.

        :param name: The name of the lexicon to retrieve.
        :return: The retrieved lexicon.
        """
        try:
            response = self.polly_client.get_lexicon(Name=name)
            logger.info("Got lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't get lexicon %s.", name)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a GetLexicon Referência da API AWS SDK for Python (Boto3).

`GetSpeechSynthesisTask`

O código de exemplo a seguir mostra como usar `GetSpeechSynthesisTask`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def get_speech_synthesis_task(self, task_id):
        """
        Gets metadata about an asynchronous speech synthesis task, such as its
 status.

        :param task_id: The ID of the task to retrieve.
        :return: Metadata about the task.
        """
        try:
            response = self.polly_client.get_speech_synthesis_task(TaskId=task_id)
            task = response["SynthesisTask"]
            logger.info("Got synthesis task. Status is %s.", task["TaskStatus"])
        except ClientError:
            logger.exception("Couldn't get synthesis task %s.", task_id)
            raise
        else:
            return task
```

- Para obter detalhes da API, consulte a GetSpeechSynthesisTask Referência da API AWS SDK for Python (Boto3).

`ListLexicons`

O código de exemplo a seguir mostra como usar `ListLexicons`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def list_lexicons(self):
        """
        Lists lexicons in the current account.

        :return: The list of lexicons.
        """
        try:
            response = self.polly_client.list_lexicons()
            lexicons = response["Lexicons"]
```

```
            logger.info("Got %s lexicons.", len(lexicons))
        except ClientError:
            logger.exception(
                "Couldn't get  %s.",
            )
            raise
        else:
            return lexicons
```

- Para obter detalhes da API, consulte a ListLexicons Referência da API AWS SDK for Python (Boto3).

`PutLexicon`

O código de exemplo a seguir mostra como usar `PutLexicon`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def create_lexicon(self, name, content):
```

```
        """
        Creates a lexicon with the specified content. A lexicon contains custom
        pronunciations.

        :param name: The name of the lexicon.
        :param content: The content of the lexicon.
        """
        try:
            self.polly_client.put_lexicon(Name=name, Content=content)
            logger.info("Created lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't create lexicon %s.")
            raise
```

- Para obter detalhes da API, consulte a PutLexicon Referência da API AWS SDK for Python (Boto3).

`StartSpeechSynthesisTask`

O código de exemplo a seguir mostra como usar `StartSpeechSynthesisTask`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
```

```
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def do_synthesis_task(
        self,
        text,
        engine,
        voice,
        audio_format,
        s3_bucket,
        lang_code=None,
        include_visemes=False,
        wait_callback=None,
    ):
        """
        Start an asynchronous task to synthesize speech or speech marks, wait for
        the task to complete, retrieve the output from Amazon S3, and return the
        data.

        An asynchronous task is required when the text is too long for near-real
time
        synthesis.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech. When
                             speech marks are synthesized, the output format is
JSON.
        :param s3_bucket: The name of an existing Amazon S3 bucket that you have
                          write access to. Synthesis output is written to this
bucket.
        :param lang_code: The language code of the voice to use. This has an effect
                          only when a bilingual voice is selected.
        :param include_visemes: When True, a second request is made to Amazon Polly
                                to synthesize a list of visemes, using the specified
                                text and voice. A viseme represents the visual
position
                                of the face and mouth when saying part of a word.
        :param wait_callback: A callback function that is called periodically during
                              task processing, to give the caller an opportunity to
                              take action, such as to display status.
        :return: The audio stream that contains the synthesized speech and a list
```

```
                 of visemes that are associated with the speech audio.
        """
        try:
            kwargs = {
                "Engine": engine,
                "OutputFormat": audio_format,
                "OutputS3BucketName": s3_bucket,
                "Text": text,
                "VoiceId": voice,
            }
            if lang_code is not None:
                kwargs["LanguageCode"] = lang_code
            response = self.polly_client.start_speech_synthesis_task(**kwargs)
            speech_task = response["SynthesisTask"]
            logger.info("Started speech synthesis task %s.", speech_task["TaskId"])

            viseme_task = None
            if include_visemes:
                kwargs["OutputFormat"] = "json"
                kwargs["SpeechMarkTypes"] = ["viseme"]
                response = self.polly_client.start_speech_synthesis_task(**kwargs)
                viseme_task = response["SynthesisTask"]
                logger.info("Started viseme synthesis task %s.",
viseme_task["TaskId"])
        except ClientError:
            logger.exception("Couldn't start synthesis task.")
            raise
        else:
            bucket = self.s3_resource.Bucket(s3_bucket)
            audio_stream = self._wait_for_task(
                10, speech_task["TaskId"], "speech", wait_callback, bucket
            )

            visemes = None
            if include_visemes:
                viseme_data = self._wait_for_task(
                    10, viseme_task["TaskId"], "viseme", wait_callback, bucket
                )
                visemes = [
                    json.loads(v) for v in viseme_data.read().decode().split() if v
                ]

            return audio_stream, visemes
```

- Para obter detalhes da API, consulte a StartSpeechSynthesisTask Referência da API AWS SDK for Python (Boto3).

`SynthesizeSpeech`

O código de exemplo a seguir mostra como usar `SynthesizeSpeech`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def synthesize(
        self, text, engine, voice, audio_format, lang_code=None,
 include_visemes=False
    ):
        """
        Synthesizes speech or speech marks from text, using the specified voice.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
```

```
        :param audio_format: The audio format to return for synthesized speech. When
                             speech marks are synthesized, the output format is
JSON.
        :param lang_code: The language code of the voice to use. This has an effect
                          only when a bilingual voice is selected.
        :param include_visemes: When True, a second request is made to Amazon Polly
                                to synthesize a list of visemes, using the specified
                                text and voice. A viseme represents the visual
position
                                of the face and mouth when saying part of a word.
        :return: The audio stream that contains the synthesized speech and a list
                 of visemes that are associated with the speech audio.
        """
        try:
            kwargs = {
                "Engine": engine,
                "OutputFormat": audio_format,
                "Text": text,
                "VoiceId": voice,
            }
            if lang_code is not None:
                kwargs["LanguageCode"] = lang_code
            response = self.polly_client.synthesize_speech(**kwargs)
            audio_stream = response["AudioStream"]
            logger.info("Got audio stream spoken by %s.", voice)
            visemes = None
            if include_visemes:
                kwargs["OutputFormat"] = "json"
                kwargs["SpeechMarkTypes"] = ["viseme"]
                response = self.polly_client.synthesize_speech(**kwargs)
                visemes = [
                    json.loads(v)
                    for v in response["AudioStream"].read().decode().split()
                    if v
                ]
                logger.info("Got %s visemes.", len(visemes))
        except ClientError:
            logger.exception("Couldn't get audio stream.")
            raise
        else:
            return audio_stream, visemes
```

- Para obter detalhes da API, consulte a SynthesizeSpeech Referência da API AWS SDK for Python (Boto3).

Cenários

Criar um aplicativo de sincronização labial

O exemplo de código a seguir mostra como criar um aplicativo de sincronização de lábios com o Amazon Polly.

SDK para Python (Boto3)

Mostra como usar o Amazon Polly e o Tkinter para criar um aplicativo de sincronização labial que exibe um rosto animado falando junto com a fala sintetizada pelo Amazon Polly. Lip-sync é realizado solicitando uma lista de visemes do Amazon Polly que coincidem com a fala sintetizada.

- Obter metadados de voz do Amazon Polly e exibi-os em um aplicativo Tkinter.
- Obter áudio de fala sintetizado e respectivas marcas de fala de fonema do Amazon Polly.
- Reproduzir o áudio com movimentos sincronizados da boca em um rosto animado.
- Enviar tarefas de síntese assíncrona para textos longos e recuperar a saída de um bucket do Amazon Simple Storage Service (Amazon S3).

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Polly

Exemplos do Amazon RDS usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon RDS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon RDS

O exemplo de código a seguir mostra como começar a usar o Amazon RDS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"""
Purpose

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational Database
 Service
(Amazon RDS) to list the databases in your account.
"""

import boto3
from botocore.exceptions import ClientError

# Create an RDS client
```

```
rds_client = boto3.client("rds")

# Create a paginator for the describe_db_instances operation
paginator = rds_client.get_paginator("describe_db_instances")

try:
    # Use the paginator to get a list of DB instances
    response_iterator = paginator.paginate(
        PaginationConfig={
            "MaxItems": 123,
            "PageSize": 50,  # Adjust PageSize as needed
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response
    instances_found = False
    for page in response_iterator:
        if "DBInstances" in page and page["DBInstances"]:
            instances_found = True
            print("Your RDS instances are:")
            for db in page["DBInstances"]:
                print(db["DBInstanceIdentifier"])

    if not instances_found:
        print("No RDS instances found!")

except ClientError as e:
    print(f"Couldn't list RDS instances. Here's why: {e.response['Error']
['Message']}")
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de banco de dados e definir os valores dos parâmetros.

- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.
- Exclua a instância e o grupo de parâmetros.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class RdsInstanceScenario:
    """Runs a scenario that shows how to get started using Amazon RDS DB
 instances."""

    def __init__(self, instance_wrapper):
        """
        :param instance_wrapper: An object that wraps Amazon RDS DB instance
 actions.
        """
        self.instance_wrapper = instance_wrapper

    def create_parameter_group(self, parameter_group_name, db_engine):
        """
        Shows how to get available engine versions for a specified database engine
 and
        create a DB parameter group that is compatible with a selected engine
 family.

        :param parameter_group_name: The name given to the newly created parameter
 group.
        :param db_engine: The database engine to use as a basis.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB instance parameter group named
 {parameter_group_name}."
```

```
        )
        parameter_group = self.instance_wrapper.get_parameter_group(
            parameter_group_name
        )
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.instance_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ", families)
            print(f"Creating a parameter group.")
            self.instance_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.instance_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group {parameter_group['DBParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def update_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
```

```
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.instance_wrapper.update_parameters(parameter_group_name, update_params)
        print(
            "You can get a list of parameters you've set by specifying a source of
'user'."
        )
        user_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_instance(self, instance_name, db_name, db_engine, parameter_group):
        """
        Shows how to create a DB instance that contains a database of a specified
        type and is configured to use a custom DB parameter group.

        :param instance_name: The name given to the newly created DB instance.
        :param db_name: The name given to the created database.
        :param db_engine: The engine of the created database.
        :param parameter_group: The parameter group that is associated with the DB
instance.
        :return: The newly created DB instance.
        """
        print("Checking for an existing DB instance.")
        db_inst = self.instance_wrapper.get_db_instance(instance_name)
        if db_inst is None:
            print("Let's create a DB instance.")
            admin_username = q.ask(
                "Enter an administrator user name for the database: ", q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters): ",
                q.non_empty,
            )
            engine_versions = self.instance_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
```

```
            engine_choices = [ver["EngineVersion"] for ver in engine_versions]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            engine_selection = engine_versions[engine_index]
            print(
                "The available micro DB instance classes for your database engine
are:"
            )
            inst_opts = self.instance_wrapper.get_orderable_instances(
                engine_selection["Engine"], engine_selection["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"]
                    for opt in inst_opts
                    if "micro" in opt["DBInstanceClass"]
                }
            )
            inst_index = q.choose(
                "Which micro DB instance class do you want to use? ", inst_choices
            )
            group_name = parameter_group["DBParameterGroupName"]
            storage_type = "standard"
            allocated_storage = 5
            print(
                f"Creating a DB instance named {instance_name} and database
{db_name}.\n"
                f"The DB instance is configured to use your custom parameter group
{group_name},\n"
                f"selected engine {engine_selection['EngineVersion']},\n"
                f"selected DB instance class {inst_choices[inst_index]},"
                f"and {allocated_storage} GiB of {storage_type} storage.\n"
                f"This typically takes several minutes."
            )
            db_inst = self.instance_wrapper.create_db_instance(
                db_name,
                instance_name,
                group_name,
                engine_selection["Engine"],
                engine_selection["EngineVersion"],
                inst_choices[inst_index],
                storage_type,
                allocated_storage,
```

```
                admin_username,
                admin_password,
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(instance_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(db_inst):
        """
        Displays connection information about a DB instance and tips on how to
        connect to it.

        :param db_inst: The DB instance to display.
        """
        print(
            "You can now connect to your database using your favorite MySql client.
\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your DB instance. Pass the endpoint,
\n"
            "port, and administrator user name to 'mysql' and enter your password\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {db_inst['Endpoint']['Address']} -P {db_inst['Endpoint']
['Port']} "
            f"-u {db_inst['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Amazon RDS:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL"
        )
        print("-" * 88)

    def create_snapshot(self, instance_name):
        """
        Shows how to create a DB instance snapshot and wait until it's available.
```

```
        :param instance_name: The name of a DB instance to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB instance (y/n)? ",
q.is_yesno
        ):
            snapshot_id = f"{instance_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
few minutes."
            )
            snapshot = self.instance_wrapper.create_snapshot(snapshot_id,
instance_name)
            while snapshot.get("Status") != "available":
                wait(10)
                snapshot = self.instance_wrapper.get_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)

    def cleanup(self, db_inst, parameter_group_name):
        """
        Shows how to clean up a DB instance and parameter group.
        Before the parameter group can be deleted, all associated DB instances must
first
        be deleted.

        :param db_inst: The DB instance to delete.
        :param parameter_group_name: The DB parameter group to delete.
        """
        if q.ask(
            "\nDo you want to delete the DB instance and parameter group (y/n)? ",
            q.is_yesno,
        ):
            print(f"Deleting DB instance {db_inst['DBInstanceIdentifier']}.")

self.instance_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(
                "Waiting for the DB instance to delete. This typically takes several
minutes."
            )
            while db_inst is not None:
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(
```

```
                    db_inst["DBInstanceIdentifier"]
                )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.instance_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, instance_name, db_name):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS)\n"
            "get started with DB instances demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(parameter_group_name,
 db_engine)
        self.update_parameters(parameter_group_name)
        db_inst = self.create_instance(
            instance_name, db_name, db_engine, parameter_group
        )
        self.display_connection(db_inst)
        self.create_snapshot(instance_name)
        self.cleanup(db_inst, parameter_group_name)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = RdsInstanceScenario(InstanceWrapper.from_client())
        scenario.run_scenario(
            "mysql",
            "doc-example-parameter-group",
            "doc-example-instance",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina as funções que são chamadas pelo cenário para gerenciar as ações do Amazon RDS.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
 parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
```

```
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
```

```
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id, DBInstanceIdentifier=instance_id
            )
```

```
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
```

```
        :param parameter_group_family: When specified, restricts the returned list
of
                                       engine versions to those that are compatible
with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by the
DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
```

```
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def create_db_instance(
        self,
        db_name,
        instance_id,
```

```
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
```

```
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateDBInstance

- CreateDBParameterGroup
- CreateDBSnapshot
- DeleteDBInstance
- DeleteDBParameterGroup
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Ações

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
```

```
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
```

```
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Python (Boto3).

`CreateDBParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBParameterGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
```

```
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateDBParameterGroup Referência da API AWS SDK for Python (Boto3).

`CreateDBSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBSnapshot`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
```

```
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id, DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para Python (Boto3).

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
```

```
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Python (Boto3).

`DeleteDBParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBParameterGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteDBParameterGroup Referência da API AWS SDK for Python (Boto3).

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list
of
                                       engine versions to those that are compatible
with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersions Referência da API AWS SDK for Python (Boto3).

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
```

```
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Python (Boto3).

`DescribeDBParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBParameterGroups`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
```

```
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- Para obter detalhes da API, consulte a DescribeDBParameterGroups Referência da API AWS SDK for Python (Boto3).

`DescribeDBParameters`

O código de exemplo a seguir mostra como usar `DescribeDBParameters`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
 filtered
                            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
 retrieved.
                       For example, a source of 'user' retrieves only parameters
 that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
```

```
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para Python (Boto3).

`DescribeDBSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBSnapshots`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Consulte detalhes da API em DescribeDBSnapshots na referência da API do AWS SDK para Python (Boto3).

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
 instance.
        :param db_engine_version: The engine version that must be supported by the
 DB instance.
        :return: The list of DB instance options that can be used to create a
 compatible DB instance.
        """
        try:
```

```
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptions Referência da API AWS SDK for Python (Boto3).

`ModifyDBParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBParameterGroup`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
```

```
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a ModifyDBParameterGroup Referência da API AWS SDK for Python (Boto3).

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um AWS Secrets Manager segredo que contenha credenciais do banco de dados e use-o para autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Python.

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
```

```
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
occurs
```

Exemplos do Amazon RDS Data Service usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) Amazon RDS Data Service.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um AWS Secrets Manager segredo que contenha credenciais do banco de dados e use-o para autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Exemplos do Amazon Redshift usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Redshift.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon Redshift

O exemplo de código a seguir mostra como começar a usar o Amazon Redshift.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_redshift(redshift_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Redshift client and list
    the clusters in your account. This list might be empty if you haven't created
    any clusters.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param redshift_client: A Boto3 Redshift Client object.
    """
    print("Hello, Redshift! Let's list your clusters:")
    paginator = redshift_client.get_paginator("describe_clusters")
    clusters = []
    for page in paginator.paginate():
        clusters.extend(page["Clusters"])

    print(f"{len(clusters)} cluster(s) were found.")
```

```
    for cluster in clusters:
        print(f"  {cluster['ClusterIdentifier']}")


if __name__ == "__main__":
    hello_redshift(boto3.client("redshift"))
```

- Para obter detalhes da API, consulte a DescribeClusters Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um cluster do Redshift.
- Listar bancos de dados no cluster.
- Criar uma tabela chamada Filmes.
- Preencher a tabela Filmes.
- Consultar a tabela Filmes por ano.
- Modificar o cluster do Redshift.
- Excluir o cluster do Amazon Redshift.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftScenario:
    """Runs an interactive scenario that shows how to get started with Redshift."""

    def __init__(self, redshift_wrapper, redshift_data_wrapper):
        self.redshift_wrapper = redshift_wrapper
```

```
        self.redshift_data_wrapper = redshift_data_wrapper

    def redhift_scenario(self, json_file_path):
        database_name = "dev"

        print(DASHES)
        print("Welcome to the Amazon Redshift SDK Getting Started example.")
        print(
            """
      This Python program demonstrates how to interact with Amazon Redshift
      using the AWS SDK for Python (Boto3).

      Amazon Redshift is a fully managed, petabyte-scale data warehouse
      service hosted in the cloud.

      The program's primary functionalities include cluster creation,
      verification of cluster readiness, listing databases, table creation,
      populating data within the table, and executing SQL statements.

      It also demonstrates querying data from the Movies table.

      Upon completion, all AWS resources are cleaned up.
    """
        )
        if not os.path.isfile(json_file_path):
            logging.error(f"The file {json_file_path} does not exist.")
            return

        print("Let's get started...")
        user_name = q.ask("Please enter your user name (default is awsuser):")
        user_name = user_name if user_name else "awsuser"

        print(DASHES)
        user_password = q.ask(
            "Please enter your user password (default is AwsUser1000):"
        )
        user_password = user_password if user_password else "AwsUser1000"

        print(DASHES)
        print(
            """A Redshift cluster refers to the collection of computing resources
and storage that work
            together to process and analyze large volumes of data."""
        )
```

```
        cluster_id = q.ask(
            "Enter a cluster identifier value (default is redshift-cluster-movies):
"
        )
        cluster_id = cluster_id if cluster_id else "redshift-cluster-movies"

        self.redshift_wrapper.create_cluster(
            cluster_id, "ra3.4xlarge", user_name, user_password, True, 2
        )

        print(DASHES)
        print(f"Wait until {cluster_id} is available. This may take a few
minutes...")
        q.ask("Press Enter to continue...")

        self.wait_cluster_available(cluster_id)

        print(DASHES)

        print(
            f"""
      When you created {cluster_id}, the dev database is created by default and
used in this scenario.

      To create a custom database, you need to have a CREATEDB privilege.
      For more information, see the documentation here:
      https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
     """
        )
        q.ask("Press Enter to continue...")
        print(DASHES)

        print(DASHES)
        print(f"List databases in {cluster_id}")
        q.ask("Press Enter to continue...")
        databases = self.redshift_data_wrapper.list_databases(
            cluster_id, database_name, user_name
        )
        print(f"The cluster contains {len(databases)} database(s).")
        for database in databases:
            print(f"    Database: {database}")
        print(DASHES)

        print(DASHES)
```

```
        print("Now you will create a table named Movies.")
        q.ask("Press Enter to continue...")

        self.create_table(cluster_id, database_name, user_name)

        print(DASHES)

        print("Populate the Movies table using the Movies.json file.")
        print(
            "Specify the number of records you would like to add to the Movies
Table."
        )
        print("Please enter a value between 50 and 200.")

        while True:
            try:
                num_records = int(q.ask("Enter a value: ", q.is_int))
                if 50 <= num_records <= 200:
                    break
                else:
                    print("Invalid input. Please enter a value between 50 and 200.")
            except ValueError:
                print("Invalid input. Please enter a value between 50 and 200.")

        self.populate_table(
            cluster_id, database_name, user_name, json_file_path, num_records
        )

        print(DASHES)
        print("Query the Movies table by year. Enter a value between 2012-2014.")

        while True:
            movie_year = int(q.ask("Enter a year: ", q.is_int))
            if 2012 <= movie_year <= 2014:
                break
            else:
                print("Invalid input. Please enter a valid year between 2012 and
2014.")

        # Function to query database
        sql_id = self.query_movies_by_year(
            database_name, user_name, movie_year, cluster_id
        )
```

```
        print(f"The identifier of the statement is {sql_id}")

        print("Checking statement status...")
        self.wait_statement_finished(sql_id)
        result = self.redshift_data_wrapper.get_statement_result(sql_id)

        self.display_movies(result)

        print(DASHES)

        print(DASHES)
        print("Now you will modify the Redshift cluster.")
        q.ask("Press Enter to continue...")

        preferred_maintenance_window = "wed:07:30-wed:08:00"
        self.redshift_wrapper.modify_cluster(cluster_id,
preferred_maintenance_window)

        print(DASHES)

        print(DASHES)
        delete = q.ask("Do you want to delete the cluster? (y/n) ", q.is_yesno)

        if delete:
            print(f"You selected to delete {cluster_id}")
            q.ask("Press Enter to continue...")
            self.redshift_wrapper.delete_cluster(cluster_id)
        else:
            print(f"Cluster {cluster_id}cluster_id was not deleted")

        print(DASHES)
        print("This concludes the Amazon Redshift SDK Getting Started scenario.")
        print(DASHES)

    def create_table(self, cluster_id, database, username):
        self.redshift_data_wrapper.execute_statement(
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql="CREATE TABLE Movies (statement_id INT PRIMARY KEY, title
VARCHAR(100), year INT)",
        )

        print("Table created: Movies")
```

```
    def populate_table(self, cluster_id, database, username, file_name, number):
        with open(file_name) as f:
            data = json.load(f)

        i = 0
        for record in data:
            if i == number:
                break

            statement_id = i
            title = record["title"]
            year = record["year"]
            i = i + 1
            parameters = [
                {"name": "statement_id", "value": str(statement_id)},
                {"name": "title", "value": title},
                {"name": "year", "value": str(year)},
            ]

            self.redshift_data_wrapper.execute_statement(
                cluster_identifier=cluster_id,
                database_name=database,
                user_name=username,
                sql="INSERT INTO Movies VALUES(:statement_id, :title, :year)",
                parameter_list=parameters,
            )

        print(f"{i} records inserted into Movies table")

    def wait_cluster_available(self, cluster_id):
        """
        Waits for a cluster to be available.

        :param cluster_id: The cluster identifier.

        Note: The cluster_available waiter can also be used.
        It is not used in this case to allow an elapsed time message.
        """
        cluster_ready = False
        start_time = time.time()

        while not cluster_ready:
```

```
            time.sleep(30)
            cluster = self.redshift_wrapper.describe_clusters(cluster_id)
            status = cluster[0]["ClusterStatus"]
            if status == "available":
                cluster_ready = True
            elif status != "creating":
                raise Exception(
                    f"Cluster {cluster_id} creation failed with status {status}."
                )

            elapsed_seconds = int(round(time.time() - start_time))
            minutes = int(elapsed_seconds // 60)
            seconds = int(elapsed_seconds % 60)

            print(f"Elapsed Time: {minutes}:{seconds:02d} - status {status}...")

            if minutes > 30:
                raise Exception(
                    f"Cluster {cluster_id} is not available after 30 minutes."
                )

    def query_movies_by_year(self, database, username, year, cluster_id):
        sql = "SELECT * FROM Movies WHERE year = :year"

        params = [{"name": "year", "value": str(year)}]

        response = self.redshift_data_wrapper.execute_statement(
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql=sql,
            parameter_list=params,
        )

        return response["Id"]

    @staticmethod
    def display_movies(response):
        metadata = response["ColumnMetadata"]
        records = response["Records"]

        title_column_index = None
        for i in range(len(metadata)):
            if metadata[i]["name"] == "title":
```

```
                title_column_index = i
                break

        if title_column_index is None:
            print("No title column found.")
            return

        print(f"Found {len(records)} movie(s).")
        for record in records:
            print(f"   {record[title_column_index]['stringValue']}")

    def wait_statement_finished(self, sql_id):
        while True:
            time.sleep(1)
            response = self.redshift_data_wrapper.describe_statement(sql_id)
            status = response["Status"]
            print(f"Statement status is {status}.")

            if status == "FAILED":
                print(f"The query failed because {response['Error']}. Ending
 program")
                raise Exception("The Query Failed. Ending program")
            elif status == "FINISHED":
                break
```

Função principal mostrando a implementação do cenário.

```
def main():
    redshift_client = boto3.client("redshift")
    redshift_data_client = boto3.client("redshift-data")
    redshift_wrapper = RedshiftWrapper(redshift_client)
    redshift_data_wrapper = RedshiftDataWrapper(redshift_data_client)
    redshift_scenario = RedshiftScenario(redshift_wrapper, redshift_data_wrapper)
    redshift_scenario.redhift_scenario(
        f"{os.path.dirname(__file__)}/../../../resources/sample_files/movies.json"
    )
```

As funções wrapper usadas no cenário.

```
    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """

        try:
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_clusters(self, cluster_identifier):
        """
```

```
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def execute_statement(
        self, cluster_identifier, database_name, user_name, sql, parameter_list=None
    ):
        """
        Executes a SQL statement.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
        :param user_name: The user's name.
        :param sql: The SQL statement.
        :param parameter_list: The optional SQL statement parameters.
        :return: The SQL statement result.
        """

        try:
            kwargs = {
                "ClusterIdentifier": cluster_identifier,
                "Database": database_name,
```

```
                "DbUser": user_name,
                "Sql": sql,
            }
            if parameter_list:
                kwargs["Parameters"] = parameter_list
            response = self.client.execute_statement(**kwargs)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't execute statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_statement_result(self, statement_id):
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
```

```
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_databases(self, cluster_identifier, database_name, database_user):
        """
        Lists databases in a cluster.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
```

```
        :param database_user: The database user.
        :return: The list of databases.
        """
        try:
            paginator = self.client.get_paginator("list_databases")
            databases = []
            for page in paginator.paginate(
                ClusterIdentifier=cluster_identifier,
                Database=database_name,
                DbUser=database_user,
            ):
                databases.extend(page["Databases"])

            return databases
        except ClientError as err:
            logging.error(
                "Couldn't list databases. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier, SkipFinalClusterSnapshot=True
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir em AWS SDK for Python (Boto3) API Reference.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def create_cluster(
        self,
```

```
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """

        try:
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftWrapper objeto.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Para obter detalhes da API, consulte a CreateCluster Referência da API AWS SDK for Python (Boto3).

`DeleteCluster`

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier, SkipFinalClusterSnapshot=True
            )
        except ClientError as err:
            logging.error(
```

```
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftWrapper objeto.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Para obter detalhes da API, consulte a DeleteCluster Referência da API AWS SDK for Python (Boto3).

`DescribeClusters`

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client
```

```
    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftWrapper objeto.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Para obter detalhes da API, consulte a DescribeClusters Referência da API AWS SDK for Python (Boto3).

`DescribeStatement`

O código de exemplo a seguir mostra como usar `DescribeStatement`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftDataWrapper objeto.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- Para obter detalhes da API, consulte a DescribeStatement Referência da API AWS SDK for Python (Boto3).

`GetStatementResult`

O código de exemplo a seguir mostra como usar `GetStatementResult`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def get_statement_result(self, statement_id):
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
```

```
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftDataWrapper objeto.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- Para obter detalhes da API, consulte a GetStatementResult Referência da API AWS SDK for Python (Boto3).

`ModifyCluster`

O código de exemplo a seguir mostra como usar `ModifyCluster`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
```

```
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

O código a seguir instancia o RedshiftWrapper objeto.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Para obter detalhes da API, consulte a ModifyCluster Referência da API AWS SDK for Python (Boto3).

Exemplos do Amazon Rekognition usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Rekognition.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CompareFaces`

O código de exemplo a seguir mostra como usar `CompareFaces`.

Para obter mais informações, consulte Comparação de faces em imagens.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
```

```
                        an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
```

```
            return matches, unmatches
```

- Para obter detalhes da API, consulte a CompareFaces Referência da API AWS SDK for Python (Boto3).

`CreateCollection`

O código de exemplo a seguir mostra como usar `CreateCollection`.

Para obter mais informações, consulte Criar uma coleção.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
```

```
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection
```

- Para obter detalhes da API, consulte a CreateCollection Referência da API AWS SDK for Python (Boto3).

`DeleteCollection`

O código de exemplo a seguir mostra como usar `DeleteCollection`.

Para obter mais informações, consulte Excluir uma coleção.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.
```

```
        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.", self.collection_id)
            raise
```

- Para obter detalhes da API, consulte a DeleteCollection Referência da API AWS SDK for Python (Boto3).

DeleteFaces

O código de exemplo a seguir mostra como usar `DeleteFaces`.

Para obter mais informações, consulte Excluir faces de uma coleção.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
```

```
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_faces(self, face_ids):
        """
        Deletes faces from the collection.

        :param face_ids: The list of IDs of faces to delete.
        :return: The list of IDs of faces that were deleted.
        """
        try:
            response = self.rekognition_client.delete_faces(
                CollectionId=self.collection_id, FaceIds=face_ids
            )
            deleted_ids = response["DeletedFaces"]
            logger.info(
                "Deleted %s faces from %s.", len(deleted_ids), self.collection_id
            )
        except ClientError:
            logger.exception("Couldn't delete faces from %s.", self.collection_id)
            raise
        else:
            return deleted_ids
```

- Para obter detalhes da API, consulte a DeleteFaces Referência da API AWS SDK for Python (Boto3).

`DescribeCollection`

O código de exemplo a seguir mostra como usar `DescribeCollection`.

Para obter mais informações, consulte Descrever uma coleção.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()
```

- Para obter detalhes da API, consulte a DescribeCollection Referência da API AWS SDK for Python (Boto3).

`DetectFaces`

O código de exemplo a seguir mostra como usar `DetectFaces`.

Para obter mais informações, consulte Detectar faces em uma imagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
```

```
            raise
        else:
            return faces
```

- Para obter detalhes da API, consulte a DetectFaces Referência da API AWS SDK for Python (Boto3).

`DetectLabels`

O código de exemplo a seguir mostra como usar `DetectLabels`.

Para obter mais informações, consulte Detectar rótulos em uma imagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client
```

```
    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels
```

- Para obter detalhes da API, consulte a DetectLabels Referência da API AWS SDK for Python (Boto3).

`DetectModerationLabels`

O código de exemplo a seguir mostra como usar `DetectModerationLabels`.

Para obter mais informações, consulte Detectar imagens impróprias.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
```

```
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

- Para obter detalhes da API, consulte a DetectModerationLabels Referência da API AWS SDK for Python (Boto3).

`DetectText`

O código de exemplo a seguir mostra como usar `DetectText`.

Para obter mais informações, consulte Detectar texto em uma imagem.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_text(self):
        """
        Detects text in the image.
```

```
        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

- Para obter detalhes da API, consulte a DetectText Referência da API AWS SDK for Python (Boto3).

`IndexFaces`

O código de exemplo a seguir mostra como usar `IndexFaces`.

Para obter mais informações, consulte Adicionar faces a uma coleção.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.
```

```
        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
```

```
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.", image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces
```

- Para obter detalhes da API, consulte a IndexFaces Referência da API AWS SDK for Python (Boto3).

`ListCollections`

O código de exemplo a seguir mostra como usar `ListCollections`.

Para obter mais informações, consulte Listar coleções.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollectionManager:
    """
```

```
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections
```

- Para obter detalhes da API, consulte a ListCollections Referência da API AWS SDK for Python (Boto3).

`ListFaces`

O código de exemplo a seguir mostra como usar `ListFaces`.

Para obter mais informações, consulte Listar faces em uma coleção.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces), self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces
```

- Para obter detalhes da API, consulte a ListFaces Referência da API AWS SDK for Python (Boto3).

`RecognizeCelebrities`

O código de exemplo a seguir mostra como usar `RecognizeCelebrities`.

Para obter mais informações, consulte Reconhecer celebridades em uma imagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
 self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in response["CelebrityFaces"]
            ]
```

```
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.", self.image_name)
            raise
        else:
            return celebrities, other_faces
```

- Para obter detalhes da API, consulte a RecognizeCelebrities Referência da API AWS SDK for Python (Boto3).

`SearchFaces`

O código de exemplo a seguir mostra como usar `SearchFaces`.

Para obter mais informações, consulte Pesquisar uma face (face ID).

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
```

```
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
```

```
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces
```

- Para obter detalhes da API, consulte a SearchFaces Referência da API AWS SDK for Python (Boto3).

`SearchFacesByImage`

O código de exemplo a seguir mostra como usar `SearchFacesByImage`.

Para obter mais informações, consulte Pesquisar uma face (imagem).

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
```

```
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces
```

- Para obter detalhes da API, consulte a SearchFacesByImage Referência da API AWS SDK for Python (Boto3).

Cenários

Criar uma coleção e encontrar faces nela

O exemplo de código a seguir mostra como:

- Criar uma coleção do Amazon Rekognition.
- Adicionar imagens à coleção e detectar faces nela.
- Pesquisar na coleção faces que correspondam a uma imagem de referência.
- Excluir uma coleção.

Para obter mais informações, consulte Pesquisar faces em uma coleção.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar classes que envolvam as funções do Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
```

```
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened and
 its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, rekognition_client):
        """
```

```
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
```

```
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections



class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )
```

```
    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
        rendering = {
            "collection_id": self.collection_id,
            "collection_arn": self.collection_arn,
            "face_count": self.face_count,
            "created": self.created,
        }
        return rendering


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()


    def delete_collection(self):
        """
        Deletes the collection.
```

```
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.", self.collection_id)
            raise


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
```

```
            logger.exception("Couldn't index faces in image %s.", image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces), self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
```

```
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
```

```
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces


class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
```

```
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
```

```
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering
```

Use as classes wrapper para criar uma coleção de faces a partir de um conjunto de imagens e, em seguida, pesquisar faces na coleção.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition face collection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    rekognition_client = boto3.client("rekognition")
    images = [
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128316.jpg",
            rekognition_client,
            image_name="sitting",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128317.jpg",
            rekognition_client,
            image_name="hopping",
        ),
```

```
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128318.jpg",
            rekognition_client,
            image_name="biking",
        ),
    ]

    collection_mgr = RekognitionCollectionManager(rekognition_client)
    collection = collection_mgr.create_collection("doc-example-collection-demo")
    print(f"Created collection {collection.collection_id}:")
    pprint(collection.describe_collection())

    print("Indexing faces from three images:")
    for image in images:
        collection.index_faces(image, 10)
    print("Listing faces in collection:")
    faces = collection.list_faces(10)
    for face in faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the first face in the "
        f"list (Face ID: {faces[0].face_id}."
    )
    found_faces = collection.search_faces(faces[0].face_id, 80, 10)
    print(f"Found {len(found_faces)} matching faces.")
    for face in found_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the largest face in "
        f"{images[0].image_name}."
    )
    image_face, match_faces = collection.search_faces_by_image(images[0], 80, 10)
    print(f"The largest face in {images[0].image_name} is:")
    pprint(image_face.to_dict())
    print(f"Found {len(match_faces)} matching faces.")
    for face in match_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    collection.delete_collection()
```

```
    print("Thanks for watching!")
    print("-" * 88)
```

Detectar e exibir elementos em imagens

O exemplo de código a seguir mostra como:

- Detectar elementos em imagens usando o Amazon Rekognition.
- Exibir imagens e desenhar caixas delimitadoras ao redor dos elementos detectados.

Para obter mais informações, consulte Exibir caixas delimitadoras.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar classes para agrupar as funções do Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace,
    RekognitionCelebrity,
    RekognitionLabel,
    RekognitionModerationLabel,
    RekognitionText,
    show_bounding_boxes,
    show_polygons,
)
```

```
logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened and
 its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)
```

```
    @classmethod
    def from_bucket(cls, s3_object, rekognition_client):
        """
        Creates a RekognitionImage object from an Amazon S3 object.

        :param s3_object: An Amazon S3 object that identifies the image. The image
                          is not retrieved until needed for a later call.
        :param rekognition_client: A Boto3 Rekognition client.
        :return: The RekognitionImage object, initialized with Amazon S3 object
data.
        """
        image = {"S3Object": {"Bucket": s3_object.bucket_name, "Name":
s3_object.key}}
        return cls(image, s3_object.key, rekognition_client)


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
```

```
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.", self.image_name)
            raise
        else:
            return celebrities, other_faces



    def compare_faces(self, target_image, similarity):
        """
```

```
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
```

```
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

Criar funções auxiliares para desenhar caixas delimitadoras e polígonos.

```
import io
import logging
from PIL import Image, ImageDraw
```

```
logger = logging.getLogger(__name__)


def show_bounding_boxes(image_bytes, box_sets, colors):
    """
    Draws bounding boxes on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box["Left"]
            top = image.height * box["Top"]
            right = (image.width * box["Width"]) + left
            bottom = (image.height * box["Height"]) + top
            draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()



def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param polygons: The list of polygons to draw on the image.
    :param color: The color to use to draw the polygons.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for polygon in polygons:
        draw.polygon(
            [
                (image.width * point["X"], image.height * point["Y"])
                for point in polygon
            ],
            outline=color,
        )
    image.show()
```

Criar classes para analisar objetos retornados pelo Amazon Rekognition.

```
class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp
```

```
    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering



class RekognitionCelebrity:
```

```
    """Encapsulates an Amazon Rekognition celebrity."""

    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.

        :param celebrity: Celebrity data, in the format returned by Amazon
 Rekognition
                          functions.
        :param timestamp: The time when the celebrity was detected, if the celebrity
                          was detected in a video.
        """
        self.info_urls = celebrity.get("Urls")
        self.name = celebrity.get("Name")
        self.id = celebrity.get("Id")
        self.face = RekognitionFace(celebrity.get("Face"))
        self.confidence = celebrity.get("MatchConfidence")
        self.bounding_box = celebrity.get("BoundingBox")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
        if self.name is not None:
            rendering["name"] = self.name
        if self.info_urls:
            rendering["info URLs"] = self.info_urls
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""

    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.
```

```
        :param person: Person data, in the format returned by Amazon Rekognition
                       functions.
        :param timestamp: The time when the person was detected, if the person
                          was detected in a video.
        """
        self.index = person.get("Index")
        self.bounding_box = person.get("BoundingBox")
        face = person.get("Face")
        self.face = RekognitionFace(face) if face is not None else None
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the person data to a dict.

        :return: A dict that contains the person data.
        """
        rendering = self.face.to_dict() if self.face is not None else {}
        if self.index is not None:
            rendering["index"] = self.index
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the label was detected, if the label
                          was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.instances = label.get("Instances")
        self.parents = label.get("Parents")
```

```
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the label data to a dict.

        :return: A dict that contains the label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionModerationLabel:
    """Encapsulates an Amazon Rekognition moderation label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the moderation label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the moderation label was detected, if the
                          label was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.parent_name = label.get("ParentName")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the moderation label data to a dict.

        :return: A dict that contains the moderation label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.parent_name is not None:
```

```
            rendering["parent_name"] = self.parent_name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionText:
    """Encapsulates an Amazon Rekognition text element."""

    def __init__(self, text_data):
        """
        Initializes the text object.

        :param text_data: Text data, in the format returned by Amazon Rekognition
                          functions.
        """
        self.text = text_data.get("DetectedText")
        self.kind = text_data.get("Type")
        self.id = text_data.get("Id")
        self.parent_id = text_data.get("ParentId")
        self.confidence = text_data.get("Confidence")
        self.geometry = text_data.get("Geometry")

    def to_dict(self):
        """
        Renders some of the text data to a dict.

        :return: A dict that contains the text data.
        """
        rendering = {}
        if self.text is not None:
            rendering["text"] = self.text
        if self.kind is not None:
            rendering["kind"] = self.kind
        if self.geometry is not None:
            rendering["polygon"] = self.geometry.get("Polygon")
        return rendering
```

Use as classes wrapper para detectar elementos em imagens e exibir suas caixas delimitadoras. As imagens usadas neste exemplo podem ser encontradas GitHub junto com instruções e mais códigos.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition image detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    rekognition_client = boto3.client("rekognition")
    street_scene_file_name = ".media/pexels-kaique-rocha-109919.jpg"
    celebrity_file_name = ".media/pexels-pixabay-53370.jpg"
    one_girl_url = "https://dhei5unw3vrsx.cloudfront.net/images/source3_resized.jpg"
    three_girls_url = "https://dhei5unw3vrsx.cloudfront.net/images/
target3_resized.jpg"
    swimwear_object = boto3.resource("s3").Object(
        "console-sample-images-pdx", "yoga_swimwear.jpg"
    )
    book_file_name = ".media/pexels-christina-morillo-1181671.jpg"

    street_scene_image = RekognitionImage.from_file(
        street_scene_file_name, rekognition_client
    )
    print(f"Detecting faces in {street_scene_image.image_name}...")
    faces = street_scene_image.detect_faces()
    print(f"Found {len(faces)} faces, here are the first three.")
    for face in faces[:3]:
        pprint(face.to_dict())
    show_bounding_boxes(
        street_scene_image.image["Bytes"],
        [[face.bounding_box for face in faces]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    print(f"Detecting labels in {street_scene_image.image_name}...")
    labels = street_scene_image.detect_labels(100)
    print(f"Found {len(labels)} labels.")
    for label in labels:
        pprint(label.to_dict())
    names = []
    box_sets = []
```

```
    colors = ["aqua", "red", "white", "blue", "yellow", "green"]
    for label in labels:
        if label.instances:
            names.append(label.name)
            box_sets.append([inst["BoundingBox"] for inst in label.instances])
    print(f"Showing bounding boxes for {names} in {colors[:len(names)]}.")
    show_bounding_boxes(
        street_scene_image.image["Bytes"], box_sets, colors[: len(names)]
    )
    input("Press Enter to continue.")

    celebrity_image = RekognitionImage.from_file(
        celebrity_file_name, rekognition_client
    )
    print(f"Detecting celebrities in {celebrity_image.image_name}...")
    celebs, others = celebrity_image.recognize_celebrities()
    print(f"Found {len(celebs)} celebrities.")
    for celeb in celebs:
        pprint(celeb.to_dict())
    show_bounding_boxes(
        celebrity_image.image["Bytes"],
        [[celeb.face.bounding_box for celeb in celebs]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    girl_image_response = requests.get(one_girl_url)
    girl_image = RekognitionImage(
        {"Bytes": girl_image_response.content}, "one-girl", rekognition_client
    )
    group_image_response = requests.get(three_girls_url)
    group_image = RekognitionImage(
        {"Bytes": group_image_response.content}, "three-girls", rekognition_client
    )
    print("Comparing reference face to group of faces...")
    matches, unmatches = girl_image.compare_faces(group_image, 80)
    print(f"Found {len(matches)} face matching the reference face.")
    show_bounding_boxes(
        group_image.image["Bytes"],
        [[match.bounding_box for match in matches]],
        ["aqua"],
    )
    input("Press Enter to continue.")
```

```
    swimwear_image = RekognitionImage.from_bucket(swimwear_object,
rekognition_client)
    print(f"Detecting suggestive content in {swimwear_object.key}...")
    labels = swimwear_image.detect_moderation_labels()
    print(f"Found {len(labels)} moderation labels.")
    for label in labels:
        pprint(label.to_dict())
    input("Press Enter to continue.")

    book_image = RekognitionImage.from_file(book_file_name, rekognition_client)
    print(f"Detecting text in {book_image.image_name}...")
    texts = book_image.detect_text()
    print(f"Found {len(texts)} text instances. Here are the first seven:")
    for text in texts[:7]:
        pprint(text.to_dict())
    show_polygons(
        book_image.image["Bytes"], [text.geometry["Polygon"] for text in texts],
"aqua"
    )

    print("Thanks for watching!")
    print("-" * 88)
```

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um aplicativo web que permite fazer o seguinte:

- Carregar fotos em um bucket do Amazon Simple Storage Service (Amazon S3).
- Usar o Amazon Rekognition para analisar e rotular as fotos.
- Usar o Amazon Simple Email Service (Amazon SES) para enviar relatórios de análise da imagem por e-mail.

Este exemplo contém dois componentes principais: uma página da web escrita JavaScript que é construída com React e um serviço REST escrito em Python que é construído com. Flask-RESTful

Você pode usar a página da Web do React para:

- Exibir uma lista de imagens que estão armazenadas no bucket do S3.
- Carregar imagens do computador para o bucket do S3.
- Exibir imagens e rótulos que identificam os itens detectados na imagem.
- Obter um relatório de todas as imagens no bucket do S3 e enviar um relatório por e-mail.

A página da Web chama o serviço REST. O serviço envia solicitações à AWS para realizar as seguintes ações:

- Obter e filtrar a lista de imagens no bucket do S3.
- Carregar fotos no bucket do S3.
- Usar o Amazon Rekognition para analisar fotos individuais e obter uma lista dos rótulos que identifiquem os itens detectados nas fotos.
- Analisar todas as fotos no bucket do S3 e usar o Amazon SES para enviar um relatório por e-mail.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são

concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Exemplos do Amazon S3 usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

- Exemplos sem servidor

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")

    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
            "PageSize": 50,  # Adjust PageSize as needed.
            "StartingToken": None,
        }
    )
```

```
    # Iterate through the pages of the response.
    buckets_found = False
    for page in response_iterator:
        if "Buckets" in page and page["Buckets"]:
            buckets_found = True
            for bucket in page["Buckets"]:
                print(f"\t{bucket['Name']}")

    if not buckets_found:
        print("No buckets found!")


if __name__ == "__main__":
    hello_s3()
```

- Para obter detalhes da API, consulte a ListBuckets Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)

    bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            }
        )
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
        return

    file_name = None
    while file_name is None:
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
            file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(
            f"Uploaded file {file_name} into bucket {bucket.name} with key
 {obj.key}."
```

```
        )
    except S3UploadFailedError as err:
        print(f"Couldn't upload file {file_name} to {bucket.name}.")
        print(f"\t{err}")

    answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
    if answer.lower() == "y":
        data = io.BytesIO()
        try:
            obj.download_fileobj(data)
            data.seek(0)
            print(f"Got your object. Here are the first 20 bytes:\n")
            print(f"\t{data.read(20)}")
        except ClientError as err:
            print(f"Couldn't download {obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
```

```
    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)? "
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                            that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
 key.
                              This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
 self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Copie um objeto usando uma solicitação condicional.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
 request.

        :param source_key: The key of the source object to copy.
```

```
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Para obter detalhes da API, consulte a CopyObject Referência da API AWS SDK for Python (Boto3).

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket com as configurações padrão.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this is
                                not specified, the Region configured in your shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
 region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
 region)
```

```
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error
```

Crie um bucket versionado com uma configuração de ciclo de vida.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
```

```
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

- Para obter detalhes da API, consulte a CreateBucket Referência da API AWS SDK for Python (Boto3).

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteBucket Referência da API AWS SDK for Python (Boto3).

`DeleteBucketCors`

O código de exemplo a seguir mostra como usar `DeleteBucketCors`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
 self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a DeleteBucketCors Referência da API AWS SDK for Python (Boto3).

`DeleteBucketLifecycle`

O código de exemplo a seguir mostra como usar `DeleteBucketLifecycle`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.", self.bucket.name
            )
        except ClientError:
            logger.exception(
```

```
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteBucketLifecycle Referência da API AWS SDK for Python (Boto3).

`DeleteBucketPolicy`

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
```

```
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteBucketPolicy Referência da API AWS SDK for Python (Boto3).

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um objeto.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def delete(self):
        """
        Deletes the object.
```

```
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
```

Reverta um objeto para uma versão anterior excluindo versões posteriores do objeto.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
```

```
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )
```

Restaure um objeto excluído removendo o marcador de exclusão ativo do objeto.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
```

```
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

Crie um manipulador do Lambda que remova um marcador de exclusão de um objeto do S3. Esse manipulador pode ser usado para limpar com eficiência marcadores de exclusão estranhos em um bucket versionado.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
 obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
 delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
```

```
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
```

```
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

- Para obter detalhes da API, consulte a DeleteObject Referência da API AWS SDK for Python (Boto3).

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exclua um conjunto de objetos de uma lista de chaves.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
```

```
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3 Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                 and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
        except ClientError:
```

```
            logger.exception("Couldn't delete any objects from bucket %s.",
 bucket.name)
            raise
        else:
            return response
```

Exclua todos os objetos em um bucket.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                             that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise
```

Exclua permanentemente um objeto versionado excluindo todas as suas versões.

```
def permanently_delete_object(bucket, object_key):
    """
```

```
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- Para obter detalhes da API, consulte a DeleteObjects Referência da API AWS SDK for Python (Boto3).

`GetBucketAcl`

O código de exemplo a seguir mostra como usar `GetBucketAcl`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
```

```
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name, acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- Para obter detalhes da API, consulte a GetBucketAcl Referência da API AWS SDK for Python (Boto3).

`GetBucketCors`

O código de exemplo a seguir mostra como usar `GetBucketCors`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
```

```
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
self.bucket.name
            )
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.", self.bucket.name))
            raise
        else:
            return cors
```

- Para obter detalhes da API, consulte a GetBucketCors Referência da API AWS SDK for Python (Boto3).

`GetBucketLifecycleConfiguration`

O código de exemplo a seguir mostra como usar `GetBucketLifecycleConfiguration`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
            return config.rules
```

- Para obter detalhes da API, consulte a GetBucketLifecycleConfiguration Referência da API AWS SDK for Python (Boto3).

`GetBucketPolicy`

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
```

```
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
 self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```

- Para obter detalhes da API, consulte a GetBucketPolicy Referência da API AWS SDK for Python (Boto3).

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get(self):
        """
        Gets the object.
```

```
        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        else:
            return body
```

Obtenha um objeto usando uma solicitação condicional.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
```

```
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of the
object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Para obter detalhes da API, consulte a GetObject Referência da API AWS SDK for Python (Boto3).

`GetObjectAcl`

O código de exemplo a seguir mostra como usar `GetObjectAcl`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key,
                acl.owner["DisplayName"],
            )
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- Para obter detalhes da API, consulte a GetObjectAcl Referência da API AWS SDK for Python (Boto3).

GetObjectLegalHold

O código de exemplo a seguir mostra como usar `GetObjectLegalHold`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a retenção legal de objetos.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
 bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
```

```
            bucket,
            e,
        )
```

- Para obter detalhes da API, consulte a GetObjectLegalHold Referência da API AWS SDK for Python (Boto3).

`GetObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `GetObjectLockConfiguration`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de bloqueio de objetos.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:
    """
    Check if object lock is enabled for a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket to check.

    Returns:
        True if object lock is enabled, False otherwise.
    """
    try:
        response = s3_client.get_object_lock_configuration(Bucket=bucket)
        return (
            "ObjectLockConfiguration" in response
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
 "Enabled"
        )
```

```
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
            return False
        else:
            raise
```

- Para obter detalhes da API, consulte a GetObjectLockConfiguration Referência da API AWS SDK for Python (Boto3).

`HeadBucket`

O código de exemplo a seguir mostra como usar `HeadBucket`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.
```

```
        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
            exists = False
        return exists
```

- Para obter detalhes da API, consulte a HeadBucket Referência da API AWS SDK for Python (Boto3).

`ListBuckets`

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
```

```
        self.name = bucket.name


    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
Boto3
                            that contains collections and factory methods to create
                            other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets
```

- Para obter detalhes da API, consulte a ListBuckets Referência da API AWS SDK for Python (Boto3).

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
```

```
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix are
listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.", bucket.name)
            raise
        else:
            return objects
```

- Para obter detalhes da API, consulte ListObjects V2 in AWS SDK for Python (Boto3) API Reference.

`PutBucketAcl`

O código de exemplo a seguir mostra como usar `PutBucketAcl`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
 recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
```

```
                    "Permission": "WRITE",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
 self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.", self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a PutBucketAcl Referência da API AWS SDK for Python (Boto3).

`PutBucketCors`

O código de exemplo a seguir mostra como usar `PutBucketCors`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_cors(self, cors_rules):
```

```
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
 self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a PutBucketCors Referência da API AWS SDK for Python (Boto3).

`PutBucketLifecycleConfiguration`

O código de exemplo a seguir mostra como usar `PutBucketLifecycleConfiguration`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
```

```
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle configuration
can
        be used to archive or delete the objects in the bucket according to
specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
```

- Para obter detalhes da API, consulte a PutBucketLifecycleConfiguration Referência da API AWS SDK for Python (Boto3).

`PutBucketPolicy`

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy, self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
 self.bucket.name)
            raise
```

- Para obter detalhes da API, consulte a PutBucketPolicy Referência da API AWS SDK for Python (Boto3).

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string. When
 this
                     argument is a string, it is interpreted as a file name, which
 is
                     opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
 data)
                raise
```

```
        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
                put_data.close()
```

Faça upload de um objeto usando uma solicitação condicional.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def put_object_conditional(self, object_key: str, source_bucket: str, data:
 bytes):
        """
```

```
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional write failed: Precondition failed")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Para obter detalhes da API, consulte a PutObject Referência da API AWS SDK for Python (Boto3).

`PutObjectAcl`

O código de exemplo a seguir mostra como usar `PutObjectAcl`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
 identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
 email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- Para obter detalhes da API, consulte a PutObjectAcl Referência da API AWS SDK for Python (Boto3).

`PutObjectLegalHold`

O código de exemplo a seguir mostra como usar `PutObjectLegalHold`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a retenção legal de objetos.

```
def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()
    logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
    try:
        before_status = "OFF"
        after_status = "ON"
        s3_client.put_object_legal_hold(
            Bucket=bucket, Key=key, LegalHold={"Status": after_status}
        )
        logger.debug(
            "Legal hold set successfully on file [%s] in bucket [%s]", key, bucket
        )
        _print_legal_hold_update(bucket, key, before_status, after_status)
    except Exception as e:
        logger.error(
            "Failed to set legal hold on file [%s] in bucket [%s]: %s", key, bucket,
 e
        )
```

- Para obter detalhes da API, consulte a PutObjectLegalHold Referência da API AWS SDK for Python (Boto3).

`PutObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `PutObjectLockConfiguration`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a configuração de bloqueio de objetos.

```
        s3_client.put_object_lock_configuration(
            Bucket=bucket,
            ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule": {}},
        )
```

- Para obter detalhes da API, consulte a PutObjectLockConfiguration Referência da API AWS SDK for Python (Boto3).

`PutObjectRetention`

O código de exemplo a seguir mostra como usar `PutObjectRetention`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina uma retenção de objetos.

```
            s3_client.put_object_retention(
                Bucket=bucket,
                Key=key,
                VersionId=version_id,
                Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
far_future_date},
                BypassGovernanceRetention=True,
            )
```

- Para obter detalhes da API, consulte a PutObjectRetention Referência da API AWS SDK for Python (Boto3).

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Gere um URL pré-assinado que possa realizar uma ação do S3 por tempo limitado. Use o pacote Requests para fazer uma solicitação com o URL.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)
```

```
def generate_presigned_url(s3_client, client_method, method_parameters, expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
 ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
        )
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to
 perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
```

```
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
        if response.status_code == 200:
            with open(args.key.split("/")[-1], 'wb') as object_file:
                object_file.write(response.content)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "rb") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be the
 "
                f"name of a file that exists on your computer."
            )

    if response is not None:
        print(f"Status: {response.status_code}\nReason: {response.reason}")

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

Gere uma solicitação POST pré-assinada para carregar um arquivo.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
```

```
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                 required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response
```

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Detectar entidades em texto extraído de uma imagem

O exemplo de código a seguir mostra como usar o Amazon Comprehend para detectar entidades em texto extraído pelo Amazon Textract de uma imagem armazenada no Amazon S3.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) em um notebook Jupyter para detectar entidades no texto que é extraído de uma imagem. Este exemplo usa o Amazon Textract para extrair texto de uma imagem armazenada no Amazon Simple Storage Service (Amazon S3) e no Amazon Comprehend para detectar entidades no texto extraído.

Este exemplo é um caderno Jupyter e deve ser executado em um ambiente que possa hospedar blocos de anotações. Para obter instruções sobre como executar o exemplo usando o Amazon SageMaker AI, consulte as instruções em TextractAndComprehendNotebook.ipynb.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um aplicativo web que permite fazer o seguinte:

- Carregar fotos em um bucket do Amazon Simple Storage Service (Amazon S3).
- Usar o Amazon Rekognition para analisar e rotular as fotos.
- Usar o Amazon Simple Email Service (Amazon SES) para enviar relatórios de análise da imagem por e-mail.

Este exemplo contém dois componentes principais: uma página da web escrita JavaScript que é construída com React e um serviço REST escrito em Python que é construído com. Flask-RESTful

Você pode usar a página da Web do React para:

- Exibir uma lista de imagens que estão armazenadas no bucket do S3.
- Carregar imagens do computador para o bucket do S3.
- Exibir imagens e rótulos que identificam os itens detectados na imagem.
- Obter um relatório de todas as imagens no bucket do S3 e enviar um relatório por e-mail.

A página da Web chama o serviço REST. O serviço envia solicitações à AWS para realizar as seguintes ações:

- Obter e filtrar a lista de imagens no bucket do S3.
- Carregar fotos no bucket do S3.
- Usar o Amazon Rekognition para analisar fotos individuais e obter uma lista dos rótulos que identifiquem os itens detectados nas fotos.
- Analisar todas as fotos no bucket do S3 e usar o Amazon SES para enviar um relatório por e-mail.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS

- Amazon SQS

Fazer solicitações condicionais

O exemplo de código a seguir mostra como adicionar pré-condições a solicitações do Amazon S3.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo que demonstre as solicitações condicionais do Amazon S3.

```
"""
Purpose

Shows how to use AWS SDK for Python (Boto3) to get started using conditional
 requests for
Amazon Simple Storage Service (Amazon S3).

"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
 setup.
sys.path.append("../../../..")
import demo_tools.question as q  # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
```

```
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)


class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
 actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key: str):
        """
        Sets up the scenario by creating a source and destination bucket.
        Prompts the user to provide a bucket name prefix.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The name of a test file to add to the source bucket.
        """

        # Create the buckets.
        try:
            self.s3_client.create_bucket(Bucket=source_bucket)
            self.s3_client.create_bucket(Bucket=dest_bucket)
            print(
                f"Created source bucket: {source_bucket} and destination bucket:
 {dest_bucket}"
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            logger.error(f"Error creating buckets: {error_code}")
            raise

        # Upload test file into the source bucket.
        try:
            print(f"Uploading file {object_key} to bucket {source_bucket}")
            response = self.s3_client.put_object(
```

```
                Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
            )
            object_etag = response["ETag"]
            return object_etag

        except Exception as e:
            logger.error(
                f"Failed to upload file {object_key} to bucket {source_bucket}: {e}"
            )


    def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
        """
        Cleans up the scenario by deleting the source and destination buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.cleanup_bucket(source_bucket)
        self.cleanup_bucket(dest_bucket)

    def cleanup_bucket(self, bucket_name: str):
        """
        Cleans up the bucket by deleting all objects and then the bucket itself.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            for obj in objs:
                key = obj["Key"]
                self.s3_client.delete_object(Bucket=bucket_name, Key=key)
            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Cleaned up bucket: {bucket_name}.")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
            else:
                logger.error(f"Error deleting bucket: {error_code}")
                raise
```

```
    def display_buckets(self, source_bucket: str, dest_bucket: str):
        """
        Display a list of the objects in the test buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.list_bucket_contents(source_bucket)
        self.list_bucket_contents(dest_bucket)

    def list_bucket_contents(self, bucket_name):
        """
        Display a list of the objects in the bucket.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            print(f"\t Items in bucket {bucket_name}")
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            if not objs:
                print("\t\tNo objects found.")
            for obj in objs:
                key = obj["Key"]
                print(f"\t\t object: {key} ETag {obj['ETag']}")
            return objs
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist.")
            else:
                logger.error(f"Error listing bucket and objects: {error_code}")
                raise


    def display_menu(
        self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
    ):
        """
        Displays the menu of conditional request options for the user.
```

```
        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The key of the test object in the source bucket.
        :param etag: The etag of the test object in the source bucket.
        """

        actions = [
            "Print list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit.",
        ]

        conditions = [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ]

        condition_types = [
            "IfMatch",
            "IfNoneMatch",
            "IfModifiedSince",
            "IfUnmodifiedSince",
        ]
        copy_condition_types = [
            "CopySourceIfMatch",
            "CopySourceIfNoneMatch",
            "CopySourceIfModifiedSince",
            "CopySourceIfUnmodifiedSince",
        ]

        yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

        choice = 0
        while choice != 4:
            print("-" * 88)
            print("Choose an action to explore some example conditional requests.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
```

```
                print("Listing the objects and buckets.")
                self.display_buckets(source_bucket, dest_bucket)
            elif choice == 1:
                print("Perform a conditional read.")
                condition_type = q.choose("Enter the condition type : ", conditions)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.get_object_conditional(
                        object_key, source_bucket, condition_types[condition_type],
etag
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.get_object_conditional(
                        object_key,
                        source_bucket,
                        condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 2:
                print("Perform a conditional copy.")
                condition_type = q.choose("Enter the condition type : ", conditions)
                dest_key = q.ask("Enter an object key: ", q.non_empty)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.copy_object_conditional(
                        object_key,
                        dest_key,
                        source_bucket,
                        dest_bucket,
                        copy_condition_types[condition_type],
                        etag,
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.copy_object_conditional(
                        object_key,
                        dest_key,
                        copy_condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 3:
                print(
                    "Perform a conditional write using IfNoneMatch condition on the
object key."
                )
                print("If the key is a duplicate, the write will fail.")
                object_key = q.ask("Enter an object key: ", q.non_empty)
```

```
                self.conditional_requests.put_object_conditional(
                    object_key, source_bucket, b"Conditional write example data."
                )
            elif choice == 4:
                print("Proceeding to cleanup.")


    def run_scenario(self):
        """
        Runs the interactive scenario.
        """
        print("-" * 88)
        print("Welcome to the Amazon S3 conditional requests example.")
        print("-" * 88)

        print(
            f"""\
        This example demonstrates the use of conditional requests for S3 operations.
        You can use conditional requests to add preconditions to S3 read requests to
return or copy
        an object based on its Entity tag (ETag), or last modified date.
        You can use a conditional write requests to prevent overwrites by ensuring
        there is no existing object with the same key.

        This example will allow you to perform conditional reads
        and writes that will succeed or fail based on your selected options.

        Sample buckets and a sample object will be created as part of the example.
        """
        )

        bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
        source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
        dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
        object_key = "test-upload-file.txt"

        try:
            etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
object_key)
            self.display_menu(source_bucket_name, dest_bucket_name, object_key,
etag)
        finally:
            self.cleanup_scenario(source_bucket_name, dest_bucket_name)
```

```
        print("-" * 88)
        print("Thanks for watching.")
        print("-" * 88)


if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()
```

Uma classe de wrapper que define as operações de solicitação condicional.

```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)


class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def get_object_conditional(
        self,
        object_key: str,
```

```
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of the
object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise



    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.
```

```
        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional write failed: Precondition failed")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise


    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
request.

        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
```

```
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CopyObject
  - GetObject
  - PutObject

Gerenciar objetos versionados em lotes com uma função do Lambda

O exemplo de código a seguir mostra como gerenciar objetos do S3 versionados em lotes com uma função do Lambda.

SDK para Python (Boto3)

Mostra como manipular objetos versionados do Amazon Simple Storage Service (Amazon S3) em lotes criando trabalhos que chamam AWS Lambda funções para realizar o processamento. Este exemplo cria um bucket habilitado para versão, carrega as estrofes do poema Você é velho, padre William, de Lewis Carroll, e usa trabalhos em lote do Amazon S3 para distorcer o poema de várias maneiras.

Aprenda como:

- Criar funções do Lambda que operam em objetos versionados.
- Criar um manifesto de objetos para atualizar.
- Cria trabalhos em lote que invocam funções do Lambda para atualizar objetos.
- Excluir funções do Lambda.
- Esvaziar e excluir um bucket versionado.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon S3

Fazer upload ou download de arquivos grandes

O exemplo de código a seguir mostra como fazer upload ou download de arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte Carregar um objeto usando carregamento fracionado.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que transfiram arquivos usando várias configurações do gerenciador de transferências disponíveis. Use uma classe de retorno de chamada para gravar o andamento do retorno de chamada durante a transferência de arquivos.

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig
```

```
MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
```

```
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
```

```
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
```

```
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
```

```
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

Demonstre as funções do gerenciador de transferências e relate resultados.

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
```

```
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")


class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
 files? "
            )
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? "
```

```
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
```

```
                self.demo_bucket,
                object_key,
                self.file_size_mb,
                **upload_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Downloading", object_key, download_file_path, **download_args
            )
            start_time = time.perf_counter()
            thread_info = download_func(
                self.demo_bucket,
                object_key,
                download_file_path,
                self.file_size_mb,
                **download_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
```

```
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:
                    self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
            self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported."
            )

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.

        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1
```

```
        local_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, local_suffix)
        )

        s3_object_key = file_name_template.format(file_tag, object_suffix)

        downloaded_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, download_suffix)
        )

        filled_cmd = self._create_file_cmd.format(
            local_file_path, self.file_size_mb * self._size_multiplier
        )

        print(
            f"Creating file of size {self.file_size_mb} MB "
            f"in {self.demo_folder} by running:"
        )
        print(f"{'':4}{filled_cmd}")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print("-" * self._terminal_width)
        print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
        if kwargs:
            print("With extra args:")
            for arg, value in kwargs.items():
                print(f'{"":4}{arg:<20}: {value}')

    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"
```

```
    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
            print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")


def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
```

```
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )

    # Download without specifying an encryption key to show that the
    # encryption key must be included to download an encrypted object.
    if demo_manager.ask_user(
        "Do you want to try to download the encrypted "
        "object without sending the required key?"
    ):
        try:
            _, object_key, download_file_path = demo_manager.last_name_set()
            file_transfer.download_with_default_configuration(
                demo_manager.demo_bucket,
                object_key,
```

```
                download_file_path,
                demo_manager.file_size_mb,
            )
        except ClientError as err:
            print(
                "Got expected error when trying to download an encrypted "
                "object without specifying encryption info:"
            )
            print(f"{'':4}{err}")

    # Remove all created and downloaded files, remove all objects from
    # S3 storage.
    if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files " "and S3
 objects?"
    ):
        demo_manager.cleanup()


if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Trabalhar com objetos versionados

O exemplo de código a seguir mostra como:

- Criar um bucket do S3 versionado.
- Obter todas as versões de um objeto.
- Reverter um objeto para uma versão anterior.
- Excluir e restaurar um objeto versionado.
- Excluir, permanentemente, todas as versões de um objeto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie funções que envolvam ações do S3.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise
```

```
    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
 expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket



def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.
```

```
    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

Carregue uma estrofe de poema para um objeto versionado e realize uma série de ações nela.

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
```

```
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
    print(
        "The latest version of the stanza is now:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions = bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[
            f"    {version.version_id}, last modified {version.last_modified}"
            for version in obj_stanza_1_versions
        ],
        sep="\n",
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print(
        "The latest version of the stanza:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
    try:
```

```
        obj_stanza_1.get()
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            print("The stanza is now deleted (as expected).")
        else:
            raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions = bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateBucket
  - DeleteObject
  - ListObjectVersions
  - PutBucketLifecycleConfiguration

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
```

```
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your
bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Exemplos de controle do Amazon S3 usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) Amazon S3 Control.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon S3 Control

O exemplo de código a seguir mostra como começar a usar o Amazon S3 Control.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def list_jobs(self, account_id: str) -> None:
        """
        List all batch jobs for the account.

        Args:
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.list_jobs(
                AccountId=account_id,
                JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
            )
            jobs = response.get('Jobs', [])
            for job in jobs:
                print(f"The job id is {job['JobId']}")
                print(f"The job priority is {job['Priority']}")
        except ClientError as e:
            print(f"Error listing jobs: {e}")
            raise
```

- Para obter detalhes da API, consulte a ListJobs Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como aprender as principais operações do Amazon S3 Control.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Conheça o cenário básico do S3 Batch.

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
 None:
        """
        Initializes the S3BatchWrapper with AWS service clients.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-level
                          access to AWS S3 services.
        :param s3control_client: A Boto3 Amazon S3 Control client. This client
 provides
                                 low-level access to AWS S3 Control services.
        :param sts_client: A Boto3 AWS STS client. This client provides low-level
                           access to AWS STS services.
        """
        self.s3_client = s3_client
        self.s3control_client = s3control_client
        self.sts_client = sts_client
        # Get region from the client for bucket creation logic
        self.region_name = self.s3_client.meta.region_name

    def get_account_id(self) -> str:
        """
        Get AWS account ID.

        Returns:
            str: AWS account ID
        """
        return self.sts_client.get_caller_identity()["Account"]

    def create_bucket(self, bucket_name: str) -> None:
        """
        Create an S3 bucket.

        Args:
            bucket_name (str): Name of the bucket to create

        Raises:
            ClientError: If bucket creation fails
        """
        try:
            if self.region_name and self.region_name != 'us-east-1':
```

```
                self.s3_client.create_bucket(
                    Bucket=bucket_name,
                    CreateBucketConfiguration={
                        'LocationConstraint': self.region_name
                    }
                )
            else:
                self.s3_client.create_bucket(Bucket=bucket_name)
            print(f"Created bucket: {bucket_name}")
        except ClientError as e:
            print(f"Error creating bucket: {e}")
            raise

    def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) ->
str:
        """
        Upload files to S3 bucket including manifest file.

        Args:
            bucket_name (str): Target bucket name
            file_names (list): List of file names to upload

        Returns:
            str: ETag of the manifest file

        Raises:
            ClientError: If file upload fails
        """
        try:
            for file_name in file_names:
                if file_name != "job-manifest.csv":
                    content = f"Content for {file_name}"
                    self.s3_client.put_object(
                        Bucket=bucket_name,
                        Key=file_name,
                        Body=content.encode('utf-8')
                    )
                    print(f"Uploaded {file_name} to {bucket_name}")

            manifest_content = ""
            for file_name in file_names:
                if file_name != "job-manifest.csv":
                    manifest_content += f"{bucket_name},{file_name}\n"
```

```
            manifest_response = self.s3_client.put_object(
                Bucket=bucket_name,
                Key="job-manifest.csv",
                Body=manifest_content.encode('utf-8')
            )
            print(f"Uploaded manifest file to {bucket_name}")
            print(f"Manifest content:\n{manifest_content}")
            return manifest_response['ETag'].strip('"')

        except ClientError as e:
            print(f"Error uploading files: {e}")
            raise

    def create_s3_batch_job(self, account_id: str, role_arn: str, manifest_location:
str,
                            report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports

        Returns:
            str: Job ID

        Raises:
            ClientError: If job creation fails
        """
        try:
            bucket_name = manifest_location.split(':::')[1].split('/')[0]
            manifest_key = 'job-manifest.csv'
            manifest_obj = self.s3_client.head_object(
                Bucket=bucket_name,
                Key=manifest_key
            )
            etag = manifest_obj['ETag'].strip('"')

            response = self.s3control_client.create_job(
                AccountId=account_id,
                Operation={
                    'S3PutObjectTagging': {
```

```
                        'TagSet': [
                            {
                                'Key': 'BatchTag',
                                'Value': 'BatchValue'
                            },
                        ]
                    }
                },
                Report={
                    'Bucket': report_bucket_name,
                    'Format': 'Report_CSV_20180820',
                    'Enabled': True,
                    'Prefix': 'batch-op-reports',
                    'ReportScope': 'AllTasks'
                },
                Manifest={
                    'Spec': {
                        'Format': 'S3BatchOperations_CSV_20180820',
                        'Fields': ['Bucket', 'Key']
                    },
                    'Location': {
                        'ObjectArn': manifest_location,
                        'ETag': etag
                    }
                },
                Priority=10,
                RoleArn=role_arn,
                Description='Batch job for tagging objects',
                ConfirmationRequired=True
            )
            job_id = response['JobId']
            print(f"The Job id is {job_id}")
            return job_id
        except ClientError as e:
            print(f"Error creating batch job: {e}")
            if 'Message' in str(e):
                print(f"Detailed error message: {e.response['Message']}")
            raise

    def check_job_failure_reasons(self, job_id: str, account_id: str) ->
List[Dict[str, Any]]:
        """
        Check for any failure reasons of a batch job.
```

```
        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID

        Returns:
            list: List of failure reasons

        Raises:
            ClientError: If checking job failure reasons fails
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            if 'FailureReasons' in response['Job']:
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return response['Job'].get('FailureReasons', [])
        except ClientError as e:
            print(f"Error checking job failure reasons: {e}")
            raise

    def wait_for_job_ready(self, job_id: str, account_id: str, desired_status: str =
'Ready') -> bool:
        """
        Wait for a job to reach the desired status.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
            desired_status (str): Target status to wait for

        Returns:
            bool: True if desired status is reached, False otherwise

        Raises:
            ClientError: If checking job status fails
        """
        print(f"Waiting for job to become {desired_status}...")
        max_attempts = 60
        attempt = 0
        while attempt < max_attempts:
            try:
```

```
                response = self.s3control_client.describe_job(
                    AccountId=account_id,
                    JobId=job_id
                )
                current_status = response['Job']['Status']
                print(f"Current job status: {current_status}")
                if current_status == desired_status:
                    return True
                if current_status == 'Suspended':
                    print("Job is in Suspended state, can proceed with activation")
                    return True
                if current_status in ['Active', 'Failed', 'Cancelled', 'Complete']:
                    print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
                    if 'FailureReasons' in response['Job']:
                        print("Failure reasons:")
                        for reason in response['Job']['FailureReasons']:
                            print(f"- {reason}")
                    return False

                time.sleep(20)
                attempt += 1
            except ClientError as e:
                print(f"Error checking job status: {e}")
                raise
        print(f"Timeout waiting for job to become {desired_status}")
        return False

    def update_job_priority(self, job_id: str, account_id: str) -> None:
        """
        Update the priority of a batch job and start it.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")
```

```
            if current_status in ['Ready', 'Suspended']:
                self.s3control_client.update_job_priority(
                    AccountId=account_id,
                    JobId=job_id,
                    Priority=60
                )
                print("The job priority was updated")

                try:
                    self.s3control_client.update_job_status(
                        AccountId=account_id,
                        JobId=job_id,
                        RequestedJobStatus='Ready'
                    )
                    print("Job activated successfully")
                except ClientError as activation_error:
                    print(f"Note: Could not activate job automatically:
{activation_error}")
                    print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return

    def cancel_job(self, job_id: str, account_id: str) -> None:
        """
        Cancel an S3 batch job.

        Args:
            job_id (str): ID of the batch job
```

```
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended', 'Active']:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Cancelled'
                )
                print(f"Job {job_id} was successfully canceled.")
            elif current_status in ['Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - cannot be cancelled")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
            else:
                print(f"Job is in '{current_status}' state - cancel not allowed")
        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise

    def describe_job_details(self, job_id: str, account_id: str) -> None:
        """
        Describe detailed information about a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            job = response['Job']
            print(f"Job ID: {job['JobId']}")
```

```
            print(f"Description: {job.get('Description', 'N/A')}")
            print(f"Status: {job['Status']}")
            print(f"Role ARN: {job['RoleArn']}")
            print(f"Priority: {job['Priority']}")
            if 'ProgressSummary' in job:
                progress = job['ProgressSummary']
                print(f"Progress Summary: Total={progress.get('TotalNumberOfTasks',
0)}, "
                      f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                      f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise

    def get_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Get tags associated with a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.get_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            tags = response.get('Tags', [])
            if tags:
                print(f"Tags for job {job_id}:")
                for tag in tags:
                    print(f"  {tag['Key']}: {tag['Value']}")
            else:
                print(f"No tags found for job ID: {job_id}")
        except ClientError as e:
            print(f"Error getting job tags: {e}")
            raise

    def put_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Add tags to a batch job.

        Args:
            job_id (str): ID of the batch job
```

```
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.put_job_tagging(
                AccountId=account_id,
                JobId=job_id,
                Tags=[
                    {'Key': 'Environment', 'Value': 'Development'},
                    {'Key': 'Team', 'Value': 'DataProcessing'}
                ]
            )
            print(f"Additional tags were added to job {job_id}")
        except ClientError as e:
            print(f"Error adding job tags: {e}")
            raise

    def list_jobs(self, account_id: str) -> None:
        """
        List all batch jobs for the account.

        Args:
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.list_jobs(
                AccountId=account_id,
                JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
 'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
            )
            jobs = response.get('Jobs', [])
            for job in jobs:
                print(f"The job id is {job['JobId']}")
                print(f"The job priority is {job['Priority']}")
        except ClientError as e:
            print(f"Error listing jobs: {e}")
            raise

    def delete_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Delete all tags from a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
```

```
        """
        try:
            self.s3control_client.delete_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            print(f"You have successfully deleted {job_id} tagging.")
        except ClientError as e:
            print(f"Error deleting job tags: {e}")
            raise

    def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
        """
        Clean up all resources created during the scenario.

        Args:
            bucket_name (str): Name of the bucket to clean up
            file_names (list): List of files to delete

        Raises:
            ClientError: If cleanup fails
        """
        try:
            for file_name in file_names:
                self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
                print(f"Deleted {file_name}")

            response = self.s3_client.list_objects_v2(
                Bucket=bucket_name,
                Prefix='batch-op-reports/'
            )
            if 'Contents' in response:
                for obj in response['Contents']:
                    self.s3_client.delete_object(
                        Bucket=bucket_name,
                        Key=obj['Key']
                    )
                    print(f"Deleted {obj['Key']}")

            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Deleted bucket {bucket_name}")
        except ClientError as e:
            print(f"Error in cleanup: {e}")
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority
  - UpdateJobStatus

Ações

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def create_s3_batch_job(self, account_id: str, role_arn: str, manifest_location:
str,
                            report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports
```

```
        Returns:
            str: Job ID

        Raises:
            ClientError: If job creation fails
        """
        try:
            bucket_name = manifest_location.split(':::')[1].split('/')[0]
            manifest_key = 'job-manifest.csv'
            manifest_obj = self.s3_client.head_object(
                Bucket=bucket_name,
                Key=manifest_key
            )
            etag = manifest_obj['ETag'].strip('"')

            response = self.s3control_client.create_job(
                AccountId=account_id,
                Operation={
                    'S3PutObjectTagging': {
                        'TagSet': [
                            {
                                'Key': 'BatchTag',
                                'Value': 'BatchValue'
                            },
                        ]
                    }
                },
                Report={
                    'Bucket': report_bucket_name,
                    'Format': 'Report_CSV_20180820',
                    'Enabled': True,
                    'Prefix': 'batch-op-reports',
                    'ReportScope': 'AllTasks'
                },
                Manifest={
                    'Spec': {
                        'Format': 'S3BatchOperations_CSV_20180820',
                        'Fields': ['Bucket', 'Key']
                    },
                    'Location': {
                        'ObjectArn': manifest_location,
                        'ETag': etag
                    }
```

```
                },
                Priority=10,
                RoleArn=role_arn,
                Description='Batch job for tagging objects',
                ConfirmationRequired=True
            )
            job_id = response['JobId']
            print(f"The Job id is {job_id}")
            return job_id
        except ClientError as e:
            print(f"Error creating batch job: {e}")
            if 'Message' in str(e):
                print(f"Detailed error message: {e.response['Message']}")
            raise
```

- Para obter detalhes da API, consulte a CreateJob Referência da API AWS SDK for Python (Boto3).

`DeleteJobTagging`

O código de exemplo a seguir mostra como usar `DeleteJobTagging`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def delete_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Delete all tags from a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.delete_job_tagging(
```

```
                AccountId=account_id,
                JobId=job_id
            )
            print(f"You have successfully deleted {job_id} tagging.")
        except ClientError as e:
            print(f"Error deleting job tags: {e}")
            raise
```

- Para obter detalhes da API, consulte a DeleteJobTagging Referência da API AWS SDK for Python (Boto3).

`DescribeJob`

O código de exemplo a seguir mostra como usar `DescribeJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def describe_job_details(self, job_id: str, account_id: str) -> None:
        """
        Describe detailed information about a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            job = response['Job']
            print(f"Job ID: {job['JobId']}")
            print(f"Description: {job.get('Description', 'N/A')}")
            print(f"Status: {job['Status']}")
```

```
            print(f"Role ARN: {job['RoleArn']}")
            print(f"Priority: {job['Priority']}")
            if 'ProgressSummary' in job:
                progress = job['ProgressSummary']
                print(f"Progress Summary: Total={progress.get('TotalNumberOfTasks',
0)}, "
                      f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                      f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise
```

- Para obter detalhes da API, consulte a DescribeJob Referência da API AWS SDK for Python (Boto3).

`GetJobTagging`

O código de exemplo a seguir mostra como usar `GetJobTagging`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def get_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Get tags associated with a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.get_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
```

```
            tags = response.get('Tags', [])
            if tags:
                print(f"Tags for job {job_id}:")
                for tag in tags:
                    print(f"  {tag['Key']}: {tag['Value']}")
            else:
                print(f"No tags found for job ID: {job_id}")
        except ClientError as e:
            print(f"Error getting job tags: {e}")
            raise
```

- Para obter detalhes da API, consulte a GetJobTagging Referência da API AWS SDK for Python (Boto3).

`PutJobTagging`

O código de exemplo a seguir mostra como usar `PutJobTagging`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def put_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Add tags to a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.put_job_tagging(
                AccountId=account_id,
                JobId=job_id,
                Tags=[
                    {'Key': 'Environment', 'Value': 'Development'},
```

```
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise
```

- Para obter detalhes da API, consulte a PutJobTagging Referência da API AWS SDK for Python (Boto3).

`UpdateJobPriority`

O código de exemplo a seguir mostra como usar `UpdateJobPriority`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def update_job_priority(self, job_id: str, account_id: str) -> None:
        """
        Update the priority of a batch job and start it.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended']:
```

```
                self.s3control_client.update_job_priority(
                    AccountId=account_id,
                    JobId=job_id,
                    Priority=60
                )
                print("The job priority was updated")

                try:
                    self.s3control_client.update_job_status(
                        AccountId=account_id,
                        JobId=job_id,
                        RequestedJobStatus='Ready'
                    )
                    print("Job activated successfully")
                except ClientError as activation_error:
                    print(f"Note: Could not activate job automatically:
{activation_error}")
                    print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return
```

- Para obter detalhes da API, consulte a UpdateJobPriority Referência da API AWS SDK for Python (Boto3).

`UpdateJobStatus`

O código de exemplo a seguir mostra como usar `UpdateJobStatus`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def cancel_job(self, job_id: str, account_id: str) -> None:
        """
        Cancel an S3 batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended', 'Active']:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Cancelled'
                )
                print(f"Job {job_id} was successfully canceled.")
            elif current_status in ['Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - cannot be cancelled")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
            else:
                print(f"Job is in '{current_status}' state - cancel not allowed")
```

```
        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise
```

- Para obter detalhes da API, consulte a UpdateJobStatus Referência da API AWS SDK for Python (Boto3).

Exemplos de buckets de diretório do S3 usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com S3 Directory Buckets.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Configure uma VPC e um endpoint da VPC.
- Configure as políticas, os perfis e o usuário para trabalhar com os buckets de diretório do S3 e a classe de armazenamento S3 Express One Zone.
- Crie dois clientes do S3.
- Crie dois buckets.

- Crie um objeto e copie-o.
- Demonstre a diferença de desempenho.
- Preencha os buckets para mostrar a diferença lexicográfica.
- Confirme se o usuário deseja limpar os recursos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário demonstrando os conceitos básicos sobre os buckets de diretório do Amazon S3 e a classe S3 Express One Zone.

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
 Express."""

    def __init__(
        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
```

```
        Runs the scenario.
        """
        print("")
        print_dashes()
        print("Welcome to the Amazon S3 Express Basics demo using Python (Boto 3)!")
        print_dashes()
        print(
            """
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results in
 this example and when you implement
Directory buckets into your infrastructure, it is best to put your compute resources
 in the same AZ as your Directory
bucket.
    """
        )
        press_enter_to_continue()

        # Create an optional VPC and create 2 IAM users.
        express_user_name, regular_user_name = self.create_vpc_and_users()

        # Set up two S3 clients, one regular and one express, and two buckets, one
 regular and one express.
        self.setup_clients_and_buckets(express_user_name, regular_user_name)

        # Create an S3 session for the express S3 client and add objects to the
 buckets.
        bucket_object = self.create_session_and_add_objects()

        # Demonstrate performance differences between regular and express buckets.
        self.demonstrate_performance(bucket_object)

        # Populate the buckets to show the lexicographical difference between
 regular and express buckets.
        self.show_lexicographical_differences(bucket_object)

        print("")
        print("That's it for our tour of the basic operations for S3 Express One
 Zone.")

        if q.ask(
            "Would you like to delete all the resources created during this demo (y/
n)? ",
```

```
            q.is_yesno,
        ):
            self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
 without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to allow
 S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in an
 EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
 same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in production!")
        print(
            """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
 permissions to do S3 Express One Zone actions.
            """
        )
        press_enter_to_continue()
        stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
        template_as_string = S3ExpressScenario.get_template_as_string()
        self.stack = self.deploy_cloudformation_stack(stack_name,
 template_as_string)
        regular_user_name = None
        express_user_name = None
```

```
        outputs = self.stack.outputs
        for output in outputs:
            if output.get("OutputKey") == "RegularUser":
                regular_user_name = output.get("OutputValue")
            elif output.get("OutputKey") == "ExpressUser":
                express_user_name = output.get("OutputValue")
        if not regular_user_name or not express_user_name:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
            'regular_user_name'={regular_user_name},
 'express_user_name'={express_user_name}
            """
            logger.error(error_string)
            raise ValueError(error_string)
        return express_user_name, regular_user_name

    def setup_clients_and_buckets(
        self, express_user_name: str, regular_user_name: str
    ) -> None:
        """
        Set up two S3 clients, one regular and one express, and two buckets, one
 regular and one express.
        :param express_user_name: The name of the user with S3 Express permissions.
        :param regular_user_name: The name of the user with regular S3 permissions.
        """
        regular_credentials = self.create_access_key(regular_user_name)
        express_credentials = self.create_access_key(express_user_name)
        # 3. Create an additional client using the credentials with S3 Express
 permissions.
        print(
            """
3. Create an additional client using the credentials with S3 Express permissions.
 This client is created with the
credentials associated with the user account with the S3 Express policy attached, so
 it can perform S3 Express operations.
"""
        )
        press_enter_to_continue()
        s3_regular_client = self.create_s3__client_with_access_key_credentials(
            regular_credentials
        )
        self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
        s3_express_client = self.create_s3__client_with_access_key_credentials(
            express_credentials
```

```
        )
        self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
        print(
            """
All the roles and policies were created and attached to the user. Then a new S3
 Client were created using
that user's credentials. We can now use this client to make calls to S3 Express
 operations. Keeping permissions in mind
(and adhering to least-privilege) is crucial to S3 Express.
 """
        )
        press_enter_to_continue()
        # 4. Create two buckets.
        print(
            """
3. Create two buckets.
Now we will create a Directory bucket which is the linchpin of the S3 Express One
 Zone service. Directory buckets
behave in different ways from regular S3 buckets which we will explore here. We'll
 also create a normal bucket, put
an object into the normal bucket, and copy it over to the Directory bucket.
"""
        )

        # Create a directory bucket. These are different from normal S3 buckets in
 subtle ways.
        bucket_prefix = q.ask(
            "Enter a bucket name prefix that will be used for both buckets: ",
            q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
        )

        # Some availability zones are not supported for Directory buckets. We'll
 choose one that is supported.
        print(
            "Now, let's choose an availability zone for the Directory bucket. We'll
 choose one that is supported."
        )
        while True:
            availability_zone = self.select_availability_zone_id(self.region)
            # Construct the parts of a directory bucket name that is made unique
 with a UUID string.
            directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"
            max_uuid_length = 63 - len(bucket_prefix) - len(directory_bucket_suffix)
 - 1
```

```
            bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
            directory_bucket_name = (
                f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
            )
            regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
            configuration = {
                "Bucket": {
                    "Type": "Directory",
                    "DataRedundancy": "SingleAvailabilityZone",
                },
                "Location": {
                    "Name": availability_zone["ZoneId"],
                    "Type": "AvailabilityZone",
                },
            }
            press_enter_to_continue()
            print(
                "Now, let's create the actual Directory bucket, as well as a regular
bucket."
            )
            press_enter_to_continue()
            try:
                self.s3_express_wrapper.create_bucket(
                    directory_bucket_name, configuration
                )
                break
            except ClientError as client_error:
                if client_error.response["Error"]["Code"] == "InvalidBucketName":
                    print(
                        f"Bucket '{directory_bucket_name}' is invalid. This may be
because of selected availability zone."
                    )
                    if q.ask(
                        "Would you like to select a different availability zone? ",
                        q.is_yesno,
                    ):
                        continue
                    else:
                        raise
                else:
                    raise
        print(f"Created directory bucket, '{directory_bucket_name}'")
        self.directory_bucket_name = directory_bucket_name
```

```
        self.s3_regular_wrapper.create_bucket(regular_bucket_name)
        print(f"Created regular bucket, '{regular_bucket_name}'")
        self.regular_bucket_name = regular_bucket_name
        print("Great! Both buckets were created.")
        press_enter_to_continue()

    def create_session_and_add_objects(self) -> None:
        """
        Create a session for the express S3 client and add objects to the buckets.
        """
        print(
            """
5. Create an object and copy it over.
We'll create a basic object consisting of some text and upload it to the normal
 bucket. Next we'll copy the object
into the Directory bucket using the regular client. This works fine because copy
 operations are not restricted for
Directory buckets.
        """
        )
        press_enter_to_continue()
        bucket_object = "basic-text-object"
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
        )
        self.s3_express_wrapper.create_session(self.directory_bucket_name)
        self.s3_express_wrapper.copy_object(
            self.regular_bucket_name,
            bucket_object,
            self.directory_bucket_name,
            bucket_object,
        )
        print(
            """
It worked! It's important to remember the user permissions when interacting with
 Directory buckets. Instead of validating
permissions on every call as normal buckets do, Directory buckets utilize the user
 credentials and session token to validate.
This allows for much faster connection speeds on every call. For single calls, this
 is low, but for many concurrent calls
this adds up to a lot of time saved.
"""
        )
        press_enter_to_continue()
```

```
        return bucket_object

    def demonstrate_performance(self, bucket_object: str) -> None:
        """
        Demonstrate performance differences between regular and Directory buckets.
        :param bucket_object: The name of the object to download from each bucket.
        """
        print("")
        print("6. Demonstrate performance difference.")
        print(
            """
Now, let's do a performance test. We'll download the same object from each bucket
 'downloads' times
and compare the total time needed. Note: the performance difference will be much
 more pronounced if this
example is run in an EC2 instance in the same Availability Zone as the bucket.
"""
        )
        downloads = 1000
        print(
            f"The number of downloads of the same object for this example is set at
 {downloads}."
        )
        if q.ask("Would you like to download a different number? (y/n) ",
 q.is_yesno):
            max_downloads = 1000000
            downloads = q.ask(
                f"Enter a number between 1 and {max_downloads} for the number of
 downloads: ",
                q.is_int,
                q.in_range(1, max_downloads),
            )
        # Download the object 'downloads' times from each bucket and time it to
 demonstrate the speed difference.
        print("Downloading from the Directory bucket.")
        directory_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")

            self.s3_express_wrapper.get_object(
                self.directory_bucket_name, bucket_object
            )
```

```
        directory_time_difference = time.time_ns() - directory_time_start
        print("Downloading from the normal bucket.")
        normal_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")
            self.s3_regular_wrapper.get_object(self.regular_bucket_name,
 bucket_object)

        normal_time_difference = time.time_ns() - normal_time_start
        print(
            f"The directory bucket took {directory_time_difference} nanoseconds,
 while the normal bucket took {normal_time_difference}."
        )
        difference = normal_time_difference - directory_time_difference
        print(f"That's a difference of {difference} nanoseconds, or")
        print(f"{(difference) / 1000000000} seconds.")
        if difference < 0:
            print(
                "The directory buckets were slower. This can happen if you are not
 running on the cloud within a vpc."
            )
        press_enter_to_continue()

    def show_lexicographical_differences(self, bucket_object: str) -> None:
        """
        Show the lexicographical difference between Directory buckets and regular
 buckets.
        This is done by creating a few objects in each bucket and listing them to
 show the difference.
        :param bucket_object: The object to use for the listing operations.
        """
        print(
            """
7. Populate the buckets to show the lexicographical difference.
Now let's explore how Directory buckets store objects in a different manner to
 regular buckets. The key is in the name
"Directory". Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual
directories/folders. This allows for more rapid indexing, traversing, and therefore
 retrieval times! The more segmented
```

```
your bucket is, with lots of directories, sub-directories, and objects, the more
 efficient it becomes. This structural
difference also causes ListObjects to behave differently, which can cause unexpected
 results. Let's add a few more
objects with layered directories to see how the output of ListObjects changes.
        """
        )
        press_enter_to_continue()
        # Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        other_object = f"other/{bucket_object}"
        alt_object = f"alt/{bucket_object}"
        other_alt_object = f"other/alt/{bucket_object}"
        self.s3_regular_wrapper.put_object(self.regular_bucket_name, other_object,
 "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name, other_object,
 "")
        self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object, "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name, alt_object,
 "")
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, other_alt_object, ""
        )
        self.s3_express_wrapper.put_object(
            self.directory_bucket_name, other_alt_object, ""
        )
        directory_bucket_objects = self.s3_express_wrapper.list_objects(
            self.directory_bucket_name
        )

        regular_bucket_objects = self.s3_regular_wrapper.list_objects(
            self.regular_bucket_name
        )

        print("Directory bucket content")
        for bucket_object in directory_bucket_objects:
            print(f"   {bucket_object['Key']}")
        print("Normal bucket content")
        for bucket_object in regular_bucket_objects:
            print(f"   {bucket_object['Key']}")
        print(
            """
Notice how the normal bucket lists objects in lexicographical order, while the
 directory bucket does not. This is
```

```
because the normal bucket considers the whole "key" to be the object identifier,
 while the directory bucket actually
creates directories and uses the object "key" as a path to the object.
            """
        )
        press_enter_to_continue()

    def cleanup(self) -> None:
        """
        Delete resources created by this scenario.
        """
        if self.directory_bucket_name is not None:
            self.s3_express_wrapper.delete_bucket_and_objects(
                self.directory_bucket_name
            )
            print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
            self.directory_bucket_name = None

        if self.regular_bucket_name is not None:

 self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
            print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
            self.regular_bucket_name = None

        if self.stack is not None:
            self.destroy_cloudformation_stack(self.stack)
            self.stack = None

        self.tear_done_vpc()

    def create_access_key(self, user_name: str) -> dict[str, any]:
        """
        Creates an access key for the user.
        :param user_name: The name of the user.
        :return: The access key for the user.
        """
        try:
            access_key = self.iam_client.create_access_key(UserName=user_name)
            return access_key["AccessKey"]
        except ClientError as client_error:
            logging.error(
                "Couldn't create the access key. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
```

```
            raise

    def create_s3__client_with_access_key_credentials(
        self, access_key: dict[str, any]
    ) -> client:
        """
        Creates an S3 client with access key credentials.
        :param access_key: The access key for the user.
        :return: The S3 Express One Zone client.
        """
        try:
            s3_express_client = boto3.client(
                "s3",
                aws_access_key_id=access_key["AccessKeyId"],
                aws_secret_access_key=access_key["SecretAccessKey"],
                region_name=self.region,
            )
            return s3_express_client
        except ClientError as client_error:
            logging.error(
                "Couldn't create the S3 Express One Zone client. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def select_availability_zone_id(self, region: str) -> dict[str, any]:
        """
        Selects an availability zone.
        :param region: The region to select the availability zone from.
        :return: The availability zone dictionary.
        """
        try:
            response = self.ec2_client.describe_availability_zones(
                Filters=[{"Name": "region-name", "Values": [region]}]
            )
            availability_zones = response["AvailabilityZones"]
            zone_names = [zone["ZoneName"] for zone in availability_zones]
            index = q.choose("Select an availability zone: ", zone_names)
            return availability_zones[index]
        except ClientError as client_error:
            logging.error(
                "Couldn't describe availability zones. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
```

```
            raise

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `cfn_template.yaml` AWS CloudFormation script and
are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example resources.
        """
        try:
            print(
                f"CloudFormation stack '{stack.name}' is being deleted. This may
take a few minutes."
```

```
            )
            stack.delete()
            waiter = self.cloud_formation_resource.meta.client.get_waiter(
                "stack_delete_complete"
            )
            waiter.wait(StackName=stack.name)
            print(f"CloudFormation stack '{stack.name}' has been deleted.")
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the CloudFormation stack. Here's why: %s",
                client_error.response["Error"]["Message"],
            )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
"s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
            )
```

```
            route_table_id = response["RouteTables"][0]["RouteTableId"]
            service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

            response = self.ec2_client.create_vpc_endpoint(
                VpcId=self.vpc_id,
                RouteTableIds=[route_table_id],
                ServiceName=service_name,
            )
            self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
            print(f"Created vpc endpoint {self.vpc_endpoint_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc endpoint. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def tear_done_vpc(self) -> None:
        if self.vpc_endpoint_id is not None:
            try:
                self.ec2_client.delete_vpc_endpoints(
                    VpcEndpointIds=[self.vpc_endpoint_id]
                )
                print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
                self.vpc_endpoint_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc endpoint %s. Here's why: %s",
                    self.vpc_endpoint_id,
                    client_error.response["Error"]["Message"],
                )
        if self.vpc_id is not None:
            try:
                self.ec2_client.delete_vpc(VpcId=self.vpc_id)
                print(f"Deleted vpc {self.vpc_id}")
                self.vpc_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc %s. Here's why: %s",
                    self.vpc_id,
                    client_error.response["Error"]["Message"],
                )
```

Uma classe de wrapper para funções de SDK do Amazon S3 Express.

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
 interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-level
                          access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def create_bucket(
        self, bucket_name: str, bucket_configuration: dict[str, any] = None
    ) -> None:
        """
        Creates a bucket.
        :param bucket_name: The name of the bucket.
        :param bucket_configuration: The optional configuration for the bucket.
        """
        try:
            params = {"Bucket": bucket_name}
            if bucket_configuration:
                params["CreateBucketConfiguration"] = bucket_configuration
```

```
            self.s3_client.create_bucket(**params)
        except ClientError as client_error:
            # Do not log InvalidBucketName error because it is logged elsewhere.
            if client_error.response["Error"]["Code"] != "InvalidBucketName":
                logging.error(
                    "Couldn't create the bucket %s. Here's why: %s",
                    bucket_name,
                    client_error.response["Error"]["Message"],
                )
            raise

    def delete_bucket_and_objects(self, bucket_name: str) -> None:
        """
        Deletes a bucket and its objects.
         :param bucket_name: The name of the bucket.
        """
        try:
            # Delete the objects in the bucket first. This is required for a bucket
to be deleted.
            paginator = self.s3_client.get_paginator("list_objects_v2")
            page_iterator = paginator.paginate(Bucket=bucket_name)
            for page in page_iterator:
                if "Contents" in page:
                    delete_keys = {
                        "Objects": [{"Key": obj["Key"]} for obj in page["Contents"]]
                    }
                    response = self.s3_client.delete_objects(
                        Bucket=bucket_name, Delete=delete_keys
                    )
                    if "Errors" in response:
                        for error in response["Errors"]:
                            logging.error(
                                "Couldn't delete object %s. Here's why: %s",
                                error["Key"],
                                error["Message"],
                            )

            self.s3_client.delete_bucket(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
```

```
            )

    def put_object(self, bucket_name: str, object_key: str, content: str) -> None:
        """
        Puts an object into a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        :param content: The content of the object.
        """
        try:
            self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't put the object %s into bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

    def list_objects(self, bucket: str) -> list[str]:
        """
        Lists objects in a bucket.
        :param bucket: The name of the bucket.
        :return: The list of objects in the bucket.
        """
        try:
            response = self.s3_client.list_objects_v2(Bucket=bucket)
            return response.get("Contents", [])
        except ClientError as client_error:
            logging.error(
                "Couldn't list objects in bucket %s. Here's why: %s",
                bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def copy_object(
        self,
        source_bucket: str,
        source_key: str,
        destination_bucket: str,
        destination_key: str,
```

```
    ) -> None:
        """
        Copies an object from one bucket to another.
        :param source_bucket: The source bucket.
        :param source_key: The source key.
        :param destination_bucket: The destination bucket.
        :param destination_key: The destination key.
        :return: None
        """
        try:
            self.s3_client.copy_object(
                CopySource={"Bucket": source_bucket, "Key": source_key},
                Bucket=destination_bucket,
                Key=destination_key,
            )
        except ClientError as client_error:
            logging.error(
                "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",
                source_key,
                source_bucket,
                destination_bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise


    def get_object(self, bucket_name: str, object_key: str) -> None:
        """
```

```
        Gets an object from a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        """
        try:
            self.s3_client.get_object(Bucket=bucket_name, Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't get the object %s from bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Ações

`CreateSession`

O código de exemplo a seguir mostra como usar `CreateSession`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
 interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
```

```
            "Couldn't create the express session for bucket %s. Here's why: %s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a CreateSession Referência da API AWS SDK for Python (Boto3).

Exemplos do Secrets Manager usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`BatchGetSecretValue`

O código de exemplo a seguir mostra como usar `BatchGetSecretValue`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class BatchGetSecretsWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def batch_get_secrets(self, filter_name):
        """
        Retrieve multiple secrets from AWS Secrets Manager using the
 batch_get_secret_value API.
        This function assumes the stack mentioned in the source code README has been
 successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param filter_name: The full or partial name of secrets to be fetched.
        :type filter_name: str
        """
        try:
            secrets = []
            response = self.client.batch_get_secret_value(
                Filters=[{"Key": "name", "Values": [f"{filter_name}"]}]
            )
            for secret in response["SecretValues"]:
                secrets.append(json.loads(secret["SecretString"]))
            if secrets:
                logger.info("Secrets retrieved successfully.")
            else:
                logger.info("Zero secrets returned without error.")
            return secrets
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"One or more requested secrets were not found with filter:
 {filter_name}"
            logger.info(msg)
            return msg
```

```
        except Exception as e:
            logger.error(f"An unknown error occurred:\n{str(e)}.")
            raise
```

- Para obter detalhes da API, consulte a BatchGetSecretValue Referência da API AWS SDK for Python (Boto3).

GetSecretValue

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class GetSecretWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def get_secret(self, secret_name):
        """
        Retrieve individual secrets from AWS Secrets Manager using the
 get_secret_value API.
        This function assumes the stack mentioned in the source code README has been
 successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param secret_name: The name of the secret fetched.
        :type secret_name: str
        """
        try:
            get_secret_value_response = self.client.get_secret_value(
```

```
                SecretId=secret_name
            )
            logging.info("Secret retrieved successfully.")
            return get_secret_value_response["SecretString"]
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"The requested secret {secret_name} was not found."
            logger.info(msg)
            return msg
        except Exception as e:
            logger.error(f"An unknown error occurred: {str(e)}.")
            raise
```

- Para obter detalhes da API, consulte a GetSecretValue Referência da API AWS SDK for Python (Boto3).

Cenários

Criar uma API REST de biblioteca de empréstimos

O exemplo de código abaixo mostra como criar uma biblioteca de empréstimos na qual os clientes possam pegar e devolver livros emprestados usando uma API REST com suporte por um banco de dados do Amazon Aurora.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com a API do Amazon Relational Database Service (Amazon RDS) e o AWS Chalice para criar uma API REST apoiada por um banco de dados Amazon Aurora. O serviço da Web é uma tecnologia sem servidor e representa uma biblioteca de empréstimos simples, na qual os clientes podem pegar e devolver livros emprestados. Aprenda como:

- Crie e gerencie um cluster de banco de dados Aurora com tecnologia sem servidor.
- Use AWS Secrets Manager para gerenciar as credenciais do banco de dados.
- Implemente uma camada de armazenamento de dados que use o Amazon RDS para mover dados para dentro e fora do banco de dados.
- Use o AWS Chalice para implantar uma API REST sem servidor no Amazon API Gateway e. AWS Lambda

- Use o pacote Requests para enviar solicitações ao serviço Web.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Exemplos do Amazon SES usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateReceiptFilter`

O código de exemplo a seguir mostra como usar `CreateReceiptFilter`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_filter(self, filter_name, ip_address_or_range, allow):
        """
        Creates a filter that allows or blocks incoming mail from an IP address or
        range.

        :param filter_name: The name to give the filter.
        :param ip_address_or_range: The IP address or range to block or allow.
        :param allow: When True, incoming mail is allowed from the specified IP
                      address or range; otherwise, it is blocked.
        """
        try:
            policy = "Allow" if allow else "Block"
            self.ses_client.create_receipt_filter(
                Filter={
                    "Name": filter_name,
                    "IpFilter": {"Cidr": ip_address_or_range, "Policy": policy},
                }
            )
            logger.info(
                "Created receipt filter %s to %s IP of %s.",
                filter_name,
                policy,
```

```
                ip_address_or_range,
            )
        except ClientError:
            logger.exception("Couldn't create receipt filter %s.", filter_name)
            raise
```

- Para obter detalhes da API, consulte a CreateReceiptFilter Referência da API AWS SDK for Python (Boto3).

`CreateReceiptRule`

O código de exemplo a seguir mostra como usar `CreateReceiptRule`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um bucket do Amazon S3 no qual o Amazon SES possa colocar cópias de e-mails recebidos e crie uma regra que copia para o bucket os e-mails recebidos de uma lista específica de destinatários.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_bucket_for_copy(self, bucket_name):
        """
```

```
        Creates a bucket that can receive copies of emails from Amazon SES. This
        includes adding a policy to the bucket that grants Amazon SES permission
        to put objects in the bucket.

        :param bucket_name: The name of the bucket to create.
        :return: The newly created bucket.
        """
        allow_ses_put_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowSESPut",
                    "Effect": "Allow",
                    "Principal": {"Service": "ses.amazonaws.com"},
                    "Action": "s3:PutObject",
                    "Resource": f"arn:aws:s3:::{bucket_name}/*",
                }
            ],
        }
        bucket = None
        try:
            bucket = self.s3_resource.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    "LocationConstraint":
self.s3_resource.meta.client.meta.region_name
                },
            )
            bucket.wait_until_exists()
            bucket.Policy().put(Policy=json.dumps(allow_ses_put_policy))
            logger.info("Created bucket %s to receive copies of emails.",
bucket_name)
        except ClientError:
            logger.exception("Couldn't create bucket to receive copies of emails.")
            if bucket is not None:
                bucket.delete()
            raise
        else:
            return bucket


    def create_s3_copy_rule(
        self, rule_set_name, rule_name, recipients, bucket_name, prefix
    ):
```

```
        """
        Creates a rule so that all emails received by the specified recipients are
        copied to an Amazon S3 bucket.

        :param rule_set_name: The name of a previously created rule set to contain
                              this rule.
        :param rule_name: The name to give the rule.
        :param recipients: When an email is received by one of these recipients, it
                           is copied to the Amazon S3 bucket.
        :param bucket_name: The name of the bucket to receive email copies. This
                            bucket must allow Amazon SES to put objects into it.
        :param prefix: An object key prefix to give the emails copied to the bucket.
        """
        try:
            self.ses_client.create_receipt_rule(
                RuleSetName=rule_set_name,
                Rule={
                    "Name": rule_name,
                    "Enabled": True,
                    "Recipients": recipients,
                    "Actions": [
                        {
                            "S3Action": {
                                "BucketName": bucket_name,
                                "ObjectKeyPrefix": prefix,
                            }
                        }
                    ],
                },
            )
            logger.info(
                "Created rule %s to copy mail received by %s to bucket %s.",
                rule_name,
                recipients,
                bucket_name,
            )
        except ClientError:
            logger.exception("Couldn't create rule %s.", rule_name)
            raise
```

- Para obter detalhes da API, consulte a CreateReceiptRule Referência da API AWS SDK for Python (Boto3).

`CreateReceiptRuleSet`

O código de exemplo a seguir mostra como usar `CreateReceiptRuleSet`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_rule_set(self, rule_set_name):
        """
        Creates an empty rule set. Rule sets contain individual rules and can be
        used to organize rules.

        :param rule_set_name: The name to give the rule set.
        """
        try:
            self.ses_client.create_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Created receipt rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't create receipt rule set %s.", rule_set_name)
            raise
```

- Para obter detalhes da API, consulte a CreateReceiptRuleSet Referência da API AWS SDK for Python (Boto3).

`CreateTemplate`

O código de exemplo a seguir mostra como usar `CreateTemplate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
```

```
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise
```

- Para obter detalhes da API, consulte a CreateTemplate Referência da API AWS SDK for Python (Boto3).

`DeleteIdentity`

O código de exemplo a seguir mostra como usar `DeleteIdentity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
```

```
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise
```

- Para obter detalhes da API, consulte a DeleteIdentity Referência da API AWS SDK for Python (Boto3).

`DeleteReceiptFilter`

O código de exemplo a seguir mostra como usar `DeleteReceiptFilter`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""
```

```
    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_filter(self, filter_name):
        """
        Deletes a receipt filter.

        :param filter_name: The name of the filter to delete.
        """
        try:
            self.ses_client.delete_receipt_filter(FilterName=filter_name)
            logger.info("Deleted receipt filter %s.", filter_name)
        except ClientError:
            logger.exception("Couldn't delete receipt filter %s.", filter_name)
            raise
```

- Para obter detalhes da API, consulte a DeleteReceiptFilter Referência da API AWS SDK for Python (Boto3).

`DeleteReceiptRule`

O código de exemplo a seguir mostra como usar `DeleteReceiptRule`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
```

```
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule(self, rule_set_name, rule_name):
        """
        Deletes a rule.

        :param rule_set_name: The rule set that contains the rule to delete.
        :param rule_name: The rule to delete.
        """
        try:
            self.ses_client.delete_receipt_rule(
                RuleSetName=rule_set_name, RuleName=rule_name
            )
            logger.info("Removed rule %s from rule set %s.", rule_name,
rule_set_name)
        except ClientError:
            logger.exception(
                "Couldn't remove rule %s from rule set %s.", rule_name,
rule_set_name
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteReceiptRule Referência da API AWS SDK for Python (Boto3).

`DeleteReceiptRuleSet`

O código de exemplo a seguir mostra como usar `DeleteReceiptRuleSet`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule_set(self, rule_set_name):
        """
        Deletes a rule set. When a rule set is deleted, all of the rules it contains
        are also deleted.

        :param rule_set_name: The name of the rule set to delete.
        """
        try:
            self.ses_client.delete_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Deleted rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't delete rule set %s.", rule_set_name)
            raise
```

- Para obter detalhes da API, consulte a DeleteReceiptRuleSet Referência da API AWS SDK for Python (Boto3).

`DeleteTemplate`

O código de exemplo a seguir mostra como usar `DeleteTemplate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

 self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
```

```
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTemplate Referência da API AWS SDK for Python (Boto3).

`DescribeReceiptRuleSet`

O código de exemplo a seguir mostra como usar `DescribeReceiptRuleSet`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def describe_receipt_rule_set(self, rule_set_name):
        """
        Gets data about a rule set.
```

```
        :param rule_set_name: The name of the rule set to retrieve.
        :return: Data about the rule set.
        """
        try:
            response = self.ses_client.describe_receipt_rule_set(
                RuleSetName=rule_set_name
            )
            logger.info("Got data for rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't get data for rule set %s.", rule_set_name)
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DescribeReceiptRuleSet Referência da API AWS SDK for Python (Boto3).

`GetIdentityVerificationAttributes`

O código de exemplo a seguir mostra como usar `GetIdentityVerificationAttributes`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
```

```
    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status
```

- Para obter detalhes da API, consulte a GetIdentityVerificationAttributes Referência da API AWS SDK for Python (Boto3).

`GetTemplate`

O código de exemplo a seguir mostra como usar `GetTemplate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
```

```
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template
```

- Para obter detalhes da API, consulte a GetTemplate Referência da API AWS SDK for Python (Boto3).

`ListIdentities`

O código de exemplo a seguir mostra como usar `ListIdentities`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
 EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
```

```
            logger.info("Got %s identities for the current account.",
len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

- Para obter detalhes da API, consulte a ListIdentities Referência da API AWS SDK for Python (Boto3).

`ListReceiptFilters`

O código de exemplo a seguir mostra como usar `ListReceiptFilters`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def list_receipt_filters(self):
        """
        Gets the list of receipt filters for the current account.
```

```
        :return: The list of receipt filters.
        """
        try:
            response = self.ses_client.list_receipt_filters()
            filters = response["Filters"]
            logger.info("Got %s receipt filters.", len(filters))
        except ClientError:
            logger.exception("Couldn't get receipt filters.")
            raise
        else:
            return filters
```

- Para obter detalhes da API, consulte a ListReceiptFilters Referência da API AWS SDK for Python (Boto3).

`ListTemplates`

O código de exemplo a seguir mostra como usar `ListTemplates`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()
```

```
    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates
```

- Para obter detalhes da API, consulte a ListTemplates Referência da API AWS SDK for Python (Boto3).

SendEmail

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html, reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
```

```
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source, destination.tos
            )
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id
```

- Para obter detalhes da API, consulte a SendEmail Referência da API AWS SDK for Python (Boto3).

`SendTemplatedEmail`

O código de exemplo a seguir mostra como usar `SendTemplatedEmail`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
```

```
    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
passed
        in this function contains key-value pairs that define the values to insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement values
                              that are inserted in the template before it is sent.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
```

```
            raise
        else:
            return message_id
```

- Para obter detalhes da API, consulte a SendTemplatedEmail Referência da API AWS SDK for Python (Boto3).

`UpdateTemplate`

O código de exemplo a seguir mostra como usar `UpdateTemplate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
```

```
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

- Para obter detalhes da API, consulte a UpdateTemplate Referência da API AWS SDK for Python (Boto3).

`VerifyDomainIdentity`

O código de exemplo a seguir mostra como usar `VerifyDomainIdentity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
        """
        Starts verification of a domain identity. To complete verification, you must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token
```

- Para obter detalhes da API, consulte a VerifyDomainIdentity Referência da API AWS SDK for Python (Boto3).

`VerifyEmailIdentity`

O código de exemplo a seguir mostra como usar `VerifyEmailIdentity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise
```

- Para obter detalhes da API, consulte a VerifyEmailIdentity Referência da API AWS SDK for Python (Boto3).

Cenários

Copiar identidades de domínio e e-mail entre regiões

O exemplo de código a seguir mostra como copiar identidades de e-mail e domínio do Amazon SES de uma AWS região para outra. Quando as identidades de domínio são gerenciadas pelo Route 53, os registros de verificação são copiados para o domínio da região de destino.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_identities(ses_client):
    """
    Gets the identities for the current Region. The Region is specified in the
    Boto3 Amazon SES client object.

    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of email identities and the list of domain identities.
    """
    email_identities = []
    domain_identities = []
    try:
        identity_paginator = ses_client.get_paginator("list_identities")
```

```
        identity_iterator = identity_paginator.paginate(
            PaginationConfig={"PageSize": 20}
        )
        for identity_page in identity_iterator:
            for identity in identity_page["Identities"]:
                if "@" in identity:
                    email_identities.append(identity)
                else:
                    domain_identities.append(identity)
        logger.info(
            "Found %s email and %s domain identities.",
            len(email_identities),
            len(domain_identities),
        )
    except ClientError:
        logger.exception("Couldn't get identities.")
        raise
    else:
        return email_identities, domain_identities


def verify_emails(email_list, ses_client):
    """
    Starts verification of a list of email addresses. Verification causes an email
    to be sent to each address. To complete verification, the recipient must follow
    the instructions in the email.

    :param email_list: The list of email addresses to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of emails that were successfully submitted for verification.
    """
    verified_emails = []
    for email in email_list:
        try:
            ses_client.verify_email_identity(EmailAddress=email)
            verified_emails.append(email)
            logger.info("Started verification of %s.", email)
        except ClientError:
            logger.warning("Couldn't start verification of %s.", email)
    return verified_emails


def verify_domains(domain_list, ses_client):
    """
```

```
    Starts verification for a list of domain identities. This returns a token for
    each domain, which must be registered as a TXT record with the DNS provider for
    the domain.

    :param domain_list: The list of domains to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The generated domain tokens to use to completed verification.
    """
    domain_tokens = {}
    for domain in domain_list:
        try:
            response = ses_client.verify_domain_identity(Domain=domain)
            token = response["VerificationToken"]
            domain_tokens[domain] = token
            logger.info("Got verification token %s for domain %s.", token, domain)
        except ClientError:
            logger.warning("Couldn't get verification token for domain %s.", domain)
    return domain_tokens


def get_hosted_zones(route53_client):
    """
    Gets the Amazon Route 53 hosted zones for the current account.

    :param route53_client: A Boto3 Route 53 client.
    :return: The list of hosted zones.
    """
    zones = []
    try:
        zone_paginator = route53_client.get_paginator("list_hosted_zones")
        zone_iterator = zone_paginator.paginate(PaginationConfig={"PageSize": 20})
        zones = [
            zone for zone_page in zone_iterator for zone in zone_page["HostedZones"]
        ]
        logger.info("Found %s hosted zones.", len(zones))
    except ClientError:
        logger.warning("Couldn't get hosted zones.")
    return zones


def find_domain_zone_matches(domains, zones):
    """
    Finds matches between Amazon SES verified domains and Route 53 hosted zones.
    Subdomain matches are taken when found, otherwise root domain matches are taken.
```

```
    :param domains: The list of domains to match.
    :param zones: The list of hosted zones to match.
    :return: The set of matched domain-zone pairs. When a match is not found, the
             domain is included in the set with a zone value of None.
    """
    domain_zones = {}
    for domain in domains:
        domain_zones[domain] = None
        # Start at the most specific sub-domain and walk up to the root domain until
 a
        # zone match is found.
        domain_split = domain.split(".")
        for index in range(0, len(domain_split) - 1):
            sub_domain = ".".join(domain_split[index:])
            for zone in zones:
                # Normalize the zone name from Route 53 by removing the trailing
 '.'.
                zone_name = zone["Name"][:-1]
                if sub_domain == zone_name:
                    domain_zones[domain] = zone
                    break
            if domain_zones[domain] is not None:
                break
    return domain_zones


def add_route53_verification_record(domain, token, zone, route53_client):
    """
    Adds a domain verification TXT record to the specified Route 53 hosted zone.
    When a TXT record already exists in the hosted zone for the specified domain,
    the existing values are preserved and the new token is added to the list.

    :param domain: The domain to add.
    :param token: The verification token for the domain.
    :param zone: The hosted zone where the domain verification record is added.
    :param route53_client: A Boto3 Route 53 client.
    """
    domain_token_record_set_name = f"_amazonses.{domain}"
    record_set_paginator = route53_client.get_paginator("list_resource_record_sets")
    record_set_iterator = record_set_paginator.paginate(
        HostedZoneId=zone["Id"], PaginationConfig={"PageSize": 20}
    )
    records = []
```

```
    for record_set_page in record_set_iterator:
        try:
            txt_record_set = next(
                record_set
                for record_set in record_set_page["ResourceRecordSets"]
                if record_set["Name"][:-1] == domain_token_record_set_name
                and record_set["Type"] == "TXT"
            )
            records = txt_record_set["ResourceRecords"]
            logger.info(
                "Existing TXT record found in set %s for zone %s.",
                domain_token_record_set_name,
                zone["Name"],
            )
            break
        except StopIteration:
            pass
    records.append({"Value": json.dumps(token)})
    changes = [
        {
            "Action": "UPSERT",
            "ResourceRecordSet": {
                "Name": domain_token_record_set_name,
                "Type": "TXT",
                "TTL": 1800,
                "ResourceRecords": records,
            },
        }
    ]
    try:
        route53_client.change_resource_record_sets(
            HostedZoneId=zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Created or updated the TXT record in set %s for zone %s.",
            domain_token_record_set_name,
            zone["Name"],
        )
    except ClientError as err:
        logger.warning(
            "Got error %s. Couldn't create or update the TXT record for zone %s.",
            err.response["Error"]["Code"],
            zone["Name"],
        )
```

```
def generate_dkim_tokens(domain, ses_client):
    """
    Generates DKIM tokens for a domain. These must be added as CNAME records to the
    DNS provider for the domain.

    :param domain: The domain to generate tokens for.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of generated DKIM tokens.
    """
    dkim_tokens = []
    try:
        dkim_tokens = ses_client.verify_domain_dkim(Domain=domain)["DkimTokens"]
        logger.info("Generated %s DKIM tokens for domain %s.", len(dkim_tokens),
 domain)
    except ClientError:
        logger.warning("Couldn't generate DKIM tokens for domain %s.", domain)
    return dkim_tokens


def add_dkim_domain_tokens(hosted_zone, domain, tokens, route53_client):
    """
    Adds DKIM domain token CNAME records to a Route 53 hosted zone.

    :param hosted_zone: The hosted zone where the records are added.
    :param domain: The domain to add.
    :param tokens: The DKIM tokens for the domain to add.
    :param route53_client: A Boto3 Route 53 client.
    """
    try:
        changes = [
            {
                "Action": "UPSERT",
                "ResourceRecordSet": {
                    "Name": f"{token}._domainkey.{domain}",
                    "Type": "CNAME",
                    "TTL": 1800,
                    "ResourceRecords": [{"Value": f"{token}.dkim.amazonses.com"}],
                },
            }
            for token in tokens
        ]
        route53_client.change_resource_record_sets(
```

```
            HostedZoneId=hosted_zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Added %s DKIM CNAME records to %s in zone %s.",
            len(tokens),
            domain,
            hosted_zone["Name"],
        )
    except ClientError:
        logger.warning(
            "Couldn't add DKIM CNAME records for %s to zone %s.",
            domain,
            hosted_zone["Name"],
        )


def configure_sns_topics(identity, topics, ses_client):
    """
    Configures Amazon Simple Notification Service (Amazon SNS) notifications for
    an identity. The Amazon SNS topics must already exist.

    :param identity: The identity to configure.
    :param topics: The list of topics to configure. The choices are Bounce,
 Delivery,
                   or Complaint.
    :param ses_client: A Boto3 Amazon SES client.
    """
    for topic in topics:
        topic_arn = input(
            f"Enter the Amazon Resource Name (ARN) of the {topic} topic or press "
            f"Enter to skip: "
        )
        if topic_arn != "":
            try:
                ses_client.set_identity_notification_topic(
                    Identity=identity, NotificationType=topic, SnsTopic=topic_arn
                )
                logger.info("Configured %s for %s notifications.", identity, topic)
            except ClientError:
                logger.warning(
                    "Couldn't configure %s for %s notifications.", identity, topic
                )
```

```
def replicate(source_client, destination_client, route53_client):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print(
        f"Replicating Amazon SES identities and other configuration from "
        f"{source_client.meta.region_name} to
 {destination_client.meta.region_name}."
    )
    print("-" * 88)

    print(f"Retrieving identities from {source_client.meta.region_name}.")
    source_emails, source_domains = get_identities(source_client)
    print("Email addresses found:")
    print(*source_emails)
    print("Domains found:")
    print(*source_domains)

    print("Starting verification for email identities.")
    dest_emails = verify_emails(source_emails, destination_client)
    print("Getting domain tokens for domain identities.")
    dest_domain_tokens = verify_domains(source_domains, destination_client)

    # Get Route 53 hosted zones and match them with Amazon SES domains.
    answer = input(
        "Is the DNS configuration for your domains managed by Amazon Route 53 (y/n)?
 "
    )
    use_route53 = answer.lower() == "y"
    hosted_zones = get_hosted_zones(route53_client) if use_route53 else []
    if use_route53:
        print("Adding or updating Route 53 TXT records for your domains.")
        domain_zones = find_domain_zone_matches(dest_domain_tokens.keys(),
 hosted_zones)
        for domain in domain_zones:
            add_route53_verification_record(
                domain, dest_domain_tokens[domain], domain_zones[domain],
 route53_client
            )
    else:
        print(
            "Use these verification tokens to create TXT records through your DNS "
            "provider:"
        )
```

```
        pprint(dest_domain_tokens)

    answer = input("Do you want to configure DKIM signing for your identities (y/n)?
 ")
    if answer.lower() == "y":
        # Build a set of unique domains from email and domain identities.
        domains = {email.split("@")[1] for email in dest_emails}
        domains.update(dest_domain_tokens)
        domain_zones = find_domain_zone_matches(domains, hosted_zones)
        for domain, zone in domain_zones.items():
            answer = input(
                f"Do you want to configure DKIM signing for {domain} (y/n)? "
            )
            if answer.lower() == "y":
                dkim_tokens = generate_dkim_tokens(domain, destination_client)
                if use_route53 and zone is not None:
                    add_dkim_domain_tokens(zone, domain, dkim_tokens,
 route53_client)
                else:
                    print(
                        "Add the following DKIM tokens as CNAME records through your
 "
                        "DNS provider:"
                    )
                    print(*dkim_tokens, sep="\n")

    answer = input(
        "Do you want to configure Amazon SNS notifications for your identities (y/
n)? "
    )
    if answer.lower() == "y":
        for identity in dest_emails + list(dest_domain_tokens.keys()):
            answer = input(
                f"Do you want to configure Amazon SNS topics for {identity} (y/n)? "
            )
            if answer.lower() == "y":
                configure_sns_topics(
                    identity, ["Bounce", "Delivery", "Complaint"],
 destination_client
                )

    print(f"Replication complete for {destination_client.meta.region_name}.")
    print("-" * 88)
```

```
def main():
    boto3_session = boto3.Session()
    ses_regions = boto3_session.get_available_regions("ses")
    parser = argparse.ArgumentParser(
        description="Copies email address and domain identities from one AWS Region
 to "
        "another. Optionally adds records for domain verification and DKIM "
        "signing to domains that are managed by Amazon Route 53, "
        "and sets up Amazon SNS notifications for events of interest."
    )
    parser.add_argument(
        "source_region", choices=ses_regions, help="The region to copy from."
    )
    parser.add_argument(
        "destination_region", choices=ses_regions, help="The region to copy to."
    )
    args = parser.parse_args()
    source_client = boto3.client("ses", region_name=args.source_region)
    destination_client = boto3.client("ses", region_name=args.destination_region)
    route53_client = boto3.client("route53")
    replicate(source_client, destination_client, route53_client)


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - ListIdentities
  - SetIdentityNotificationTopic
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Criar uma aplicação Web para monitorar dados do DynamoDB

O exemplo de código a seguir mostra como criar uma aplicação Web que monitora itens de trabalho em uma tabela do Amazon DynamoDB e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho no Amazon DynamoDB e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS
- Leia, grave e atualize itens de trabalho armazenados em uma tabela do DynamoDB.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, consulte o exemplo completo no Repositório de exemplos de AWS código em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon SES

Crie um rastreador de itens de trabalho do Aurora Sem Servidor

O exemplo de código a seguir mostra como criar uma aplicação Web que rastreia os itens de trabalho em um banco de dados do Amazon Aurora Sem Servidor e usa o Amazon Simple Email Service (Amazon SES) para enviar relatórios.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um serviço REST que rastreia itens de trabalho em um banco de dados Amazon Aurora Serverless e envia relatórios por e-mail usando o Amazon Simple Email Service (Amazon SES). Este exemplo usa a estrutura web Flask para lidar com o roteamento HTTP e se integra a uma página da Web do React para apresentar uma aplicação Web totalmente funcional.

- Crie um serviço Flask REST que se integre com o. Serviços da AWS

- Leia, grave e atualize itens de trabalho armazenados em um banco de dados do Aurora Sem Servidor.
- Crie um AWS Secrets Manager segredo que contenha credenciais do banco de dados e use-o para autenticar chamadas para o banco de dados.
- Use o Amazon SES para enviar relatórios por e-mail de itens de trabalho.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Aurora
- Amazon RDS
- Serviços de dados do Amazon RDS
- Amazon SES

Detectar objetos em imagens

O exemplo de código a seguir mostra como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) para criar um aplicativo web que permite fazer o seguinte:

- Carregar fotos em um bucket do Amazon Simple Storage Service (Amazon S3).
- Usar o Amazon Rekognition para analisar e rotular as fotos.
- Usar o Amazon Simple Email Service (Amazon SES) para enviar relatórios de análise da imagem por e-mail.

Este exemplo contém dois componentes principais: uma página da web escrita JavaScript que é construída com React e um serviço REST escrito em Python que é construído com. Flask-RESTful

Você pode usar a página da Web do React para:

- Exibir uma lista de imagens que estão armazenadas no bucket do S3.
- Carregar imagens do computador para o bucket do S3.
- Exibir imagens e rótulos que identificam os itens detectados na imagem.

- Obter um relatório de todas as imagens no bucket do S3 e enviar um relatório por e-mail.

A página da Web chama o serviço REST. O serviço envia solicitações à AWS para realizar as seguintes ações:

- Obter e filtrar a lista de imagens no bucket do S3.
- Carregar fotos no bucket do S3.
- Usar o Amazon Rekognition para analisar fotos individuais e obter uma lista dos rótulos que identifiquem os itens detectados nas fotos.
- Analisar todas as fotos no bucket do S3 e usar o Amazon SES para enviar um relatório por e-mail.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition

- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Gerar credenciais para estabelecer conexão com um endpoint SMTP

O exemplo de código a seguir mostra como gerar credenciais para estabelecer conexão com um endpoint SMTP do Amazon SES.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
#!/usr/bin/env python3

import hmac
import hashlib
import base64
import argparse

SMTP_REGIONS = [
    "us-east-2",  # US East (Ohio)
    "us-east-1",  # US East (N. Virginia)
    "us-west-2",  # US West (Oregon)
    "ap-south-1",  # Asia Pacific (Mumbai)
    "ap-northeast-2",  # Asia Pacific (Seoul)
    "ap-southeast-1",  # Asia Pacific (Singapore)
    "ap-southeast-2",  # Asia Pacific (Sydney)
    "ap-northeast-1",  # Asia Pacific (Tokyo)
    "ca-central-1",  # Canada (Central)
    "eu-central-1",  # Europe (Frankfurt)
    "eu-west-1",  # Europe (Ireland)
    "eu-west-2",  # Europe (London)
    "eu-south-1",  # Europe (Milan)
    "eu-north-1",  # Europe (Stockholm)
```

```
    "sa-east-1",  # South America (Sao Paulo)
    "us-gov-west-1",  # AWS GovCloud (US)
    "us-gov-east-1",  # AWS GovCloud (US)
]

# These values are required to calculate the signature. Do not change them.
DATE = "11111111"
SERVICE = "ses"
MESSAGE = "SendRawEmail"
TERMINAL = "aws4_request"
VERSION = 0x04


def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def calculate_key(secret_access_key, region):
    if region not in SMTP_REGIONS:
        raise ValueError(f"The {region} Region doesn't have an SMTP endpoint.")

    signature = sign(("AWS4" + secret_access_key).encode("utf-8"), DATE)
    signature = sign(signature, region)
    signature = sign(signature, SERVICE)
    signature = sign(signature, TERMINAL)
    signature = sign(signature, MESSAGE)
    signature_and_version = bytes([VERSION]) + signature
    smtp_password = base64.b64encode(signature_and_version)
    return smtp_password.decode("utf-8")


def main():
    parser = argparse.ArgumentParser(
        description="Convert a Secret Access Key to an SMTP password."
    )
    parser.add_argument("secret", help="The Secret Access Key to convert.")
    parser.add_argument(
        "region",
        help="The AWS Region where the SMTP password will be used.",
        choices=SMTP_REGIONS,
    )
    args = parser.parse_args()
    print(calculate_key(args.secret, args.region))
```

```
if __name__ == "__main__":
    main()
```

Verificar uma identidade de e-mail e enviar mensagens

O exemplo de código a seguir mostra como:

- Adicionar e verificar um endereço de e-mail com o Amazon SES.
- Enviar uma mensagem de e-mail padrão.
- Criar um modelo e envie uma mensagem de e-mail com modelo.
- Enviar uma mensagem usando um servidor SMTP do Amazon SES.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Verifique um endereço de e-mail com o Amazon SES e envie mensagens.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Simple Email Service (Amazon SES) email demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    ses_client = boto3.client("ses")
    ses_identity = SesIdentity(ses_client)
    ses_mail_sender = SesMailSender(ses_client)
    ses_template = SesTemplate(ses_client)
    email = input("Enter an email address to send mail with Amazon SES: ")
    status = ses_identity.get_identity_status(email)
    verified = status == "Success"
    if not verified:
        answer = input(
```

```
            f"The address '{email}' is not verified with Amazon SES. Unless your "
            f"Amazon SES account is out of sandbox, you can send mail only from "
            f"and to verified accounts. Do you want to verify this account for use "
            f"with Amazon SES? If yes, the address will receive a verification "
            f"email (y/n): "
        )
        if answer.lower() == "y":
            ses_identity.verify_email_identity(email)
            print(f"Follow the steps in the email to {email} to complete
 verification.")
            print("Waiting for verification...")
            try:
                ses_identity.wait_until_identity_exists(email)
                print(f"Identity verified for {email}.")
                verified = True
            except WaiterError:
                print(
                    f"Verification timeout exceeded. You must complete the "
                    f"steps in the email sent to {email} to verify the address."
                )

    if verified:
        test_message_text = "Hello from the Amazon SES mail demo!"
        test_message_html = "<p>Hello!</p><p>From the <b>Amazon SES</b> mail demo!</
p>"

        print(f"Sending mail from {email} to {email}.")
        ses_mail_sender.send_email(
            email,
            SesDestination([email]),
            "Amazon SES demo",
            test_message_text,
            test_message_html,
        )
        input("Mail sent. Check your inbox and press Enter to continue.")

        template = {
            "name": "doc-example-template",
            "subject": "Example of an email template.",
            "text": "This is what {{name}} will {{action}} if {{name}} can't display
 "
            "HTML.",
            "html": "<p><i>This</i> is what {{name}} will {{action}} if {{name}} "
            "<b>can</b> display HTML.</p>",
```

```
        }
        print("Creating a template and sending a templated email.")
        ses_template.create_template(**template)
        template_data = {"name": email.split("@")[0], "action": "read"}
        if ses_template.verify_tags(template_data):
            ses_mail_sender.send_templated_email(
                email, SesDestination([email]), ses_template.name(), template_data
            )
            input("Mail sent. Check your inbox and press Enter to continue.")

        print("Sending mail through the Amazon SES SMTP server.")
        boto3_session = boto3.Session()
        region = boto3_session.region_name
        credentials = boto3_session.get_credentials()
        port = 587
        smtp_server = f"email-smtp.{region}.amazonaws.com"
        password = calculate_key(credentials.secret_key, region)
        message = """
Subject: Hi there

This message is sent from the Amazon SES SMTP mail demo."""
        context = ssl.create_default_context()
        with smtplib.SMTP(smtp_server, port) as server:
            server.starttls(context=context)
            server.login(credentials.access_key, password)
            server.sendmail(email, email, message)
        print("Mail sent. Check your inbox!")

    if ses_template.template is not None:
        print("Deleting demo template.")
        ses_template.delete_template()
    if verified:
        answer = input(f"Do you want to remove {email} from Amazon SES (y/n)? ")
        if answer.lower() == "y":
            ses_identity.delete_identity(email)
    print("Thanks for watching!")
    print("-" * 88)
```

Crie funções para encapsular ações de identidade do Amazon SES.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
        """
        Starts verification of a domain identity. To complete verification, you must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
```

```
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise


    def wait_until_identity_exists(self, identity):
        """
        Waits until an identity exists. The waiter polls Amazon SES until the
        identity has been successfully verified or until it exceeds its maximum
time.

        :param identity: The identity to wait for.
        """
        try:
            waiter = self.ses_client.get_waiter("identity_exists")
            logger.info("Waiting until %s exists.", identity)
            waiter.wait(Identities=[identity])
        except WaiterError:
            logger.error("Waiting for identity %s failed or timed out.", identity)
            raise


    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
```

```
            return status


    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

Crie funções para encapsular ações com modelo do Amazon SES.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
```

```
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise


    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template
```

```
    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

Crie funções para encapsular ações de e-mail do Amazon SES.

```
class SesDestination:
    """Contains data about an email destination."""

    def __init__(self, tos, ccs=None, bccs=None):
        """
        :param tos: The list of recipients on the 'To:' line.
        :param ccs: The list of recipients on the 'CC:' line.
        :param bccs: The list of recipients on the 'BCC:' line.
        """
        self.tos = tos
        self.ccs = ccs
        self.bccs = bccs

    def to_service_format(self):
        """
        :return: The destination data in the format expected by Amazon SES.
        """
        svc_format = {"ToAddresses": self.tos}
        if self.ccs is not None:
            svc_format["CcAddresses"] = self.ccs
        if self.bccs is not None:
            svc_format["BccAddresses"] = self.bccs
        return svc_format



class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html, reply_tos=None):
        """
        Sends an email.
```

```
        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source, destination.tos
            )
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id


    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
```

```
        each enclosed in two curly braces, such as {{name}}. The template data
passed
        in this function contains key-value pairs that define the values to insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement values
                              that are inserted in the template before it is sent.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
            raise
        else:
            return message_id
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTemplate
  - DeleteIdentity
  - DeleteTemplate
  - GetIdentityVerificationAttributes
  - GetTemplate
  - ListIdentities
  - ListTemplates
  - SendEmail
  - SendTemplatedEmail
  - UpdateTemplate
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Exemplos da API v2 do Amazon SES usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API v2 AWS SDK para Python (Boto3) com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, API Amazon SES v2

O exemplo de código a seguir mostra como começar a usar a API v2 do Amazon SES.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def hello_sesv2(sesv2_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon SESv2 client and
    list the email identities in your account. This example uses the default
    settings specified in your shared credentials and config files.

    :param sesv2_client: A Boto3 SESv2 client object.
    """
    print("Hello, Amazon SESv2. Let's list up to 5 email identities:\n")

    try:
        response = sesv2_client.list_email_identities(PageSize=5)
        identities = response["EmailIdentities"]

        if not identities:
            print(
                "No email identities found. "
                "Use CreateEmailIdentity to add one."
            )
        else:
            for identity in identities:
                print(
                    f"  Identity: {identity['IdentityName']}"
                    f"  Type: {identity['IdentityType']}"
                    f"  Status: {identity['VerificationStatus']}"
                    f"  Sending: {'Enabled' if identity['SendingEnabled'] else
 'Disabled'}"
                )
            print(f"\nShowing {len(identities)} email identity(ies).")
```

```
    except ClientError as err:
        logger.error(
            "Couldn't list email identities. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a ListEmailIdentities Referência da API AWS SDK for Python (Boto3).

Ações

`CreateContact`

O código de exemplo a seguir mostra como usar `CreateContact`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
```

```
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
```

```
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
 Skipping...")
                else:
                    raise e
```

- Para obter detalhes da API, consulte a CreateContact Referência da API AWS SDK for Python (Boto3).

`CreateContactList`

O código de exemplo a seguir mostra como usar `CreateContactList`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()
```

```
class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e
```

- Para obter detalhes da API, consulte a CreateContactList Referência da API AWS SDK for Python (Boto3).

`CreateEmailIdentity`

O código de exemplo a seguir mostra como usar `CreateEmailIdentity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
```

```
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e
```

- Para obter detalhes da API, consulte a CreateEmailIdentity Referência da API AWS SDK for Python (Boto3).

`CreateEmailTemplate`

O código de exemplo a seguir mostra como usar `CreateEmailTemplate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
```

```
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e
```

- Para obter detalhes da API, consulte a CreateEmailTemplate Referência da API AWS SDK for Python (Boto3).

`DeleteContactList`

O código de exemplo a seguir mostra como usar `DeleteContactList`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
```

```
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)
```

- Para obter detalhes da API, consulte a DeleteContactList Referência da API AWS SDK for Python (Boto3).

`DeleteEmailIdentity`

O código de exemplo a seguir mostra como usar `DeleteEmailIdentity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
```

```
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:

 self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
 successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not exist.")
                else:
                    print(e)
```

- Para obter detalhes da API, consulte a DeleteEmailIdentity Referência da API AWS SDK for Python (Boto3).

`DeleteEmailTemplate`

O código de exemplo a seguir mostra como usar `DeleteEmailTemplate`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
```

```
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

- Para obter detalhes da API, consulte a DeleteEmailTemplate Referência da API AWS SDK for Python (Boto3).

`GetEmailIdentity`

O código de exemplo a seguir mostra como usar `GetEmailIdentity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
```

```
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def get_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Gets information about an email identity, including its verification status.

        :param email_address: The email address or domain to look up.
        :return: A dictionary with identity information including verification
status.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            response = self.sesv2_client.get_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Got email identity for %s.", email_address)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found.", email_address
                )
            else:
                logger.error(
                    "Couldn't get email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a GetEmailIdentity Referência da API AWS SDK for Python (Boto3).

`ListContacts`

O código de exemplo a seguir mostra como usar `ListContacts`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
```

```
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e
```

- Para obter detalhes da API, consulte a ListContacts Referência da API AWS SDK for Python (Boto3).

`SendBulkEmail`

O código de exemplo a seguir mostra como usar `SendBulkEmail`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)
```

```
    def send_bulk_email(
        self,
        from_address: str,
        template_name: str,
        default_template_data: str,
        bulk_entries: List[Dict[str, Any]],
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Sends a templated email to multiple recipients in a single API call.

        All recipients receive the same attachment(s) defined in the default
        content, while template data can be personalized per recipient.

        :param from_address: The verified sender email address.
        :param template_name: The name of an existing email template.
        :param default_template_data: Default JSON template data string.
        :param bulk_entries: A list of BulkEmailEntry dicts, each containing
            'Destination' and optionally 'ReplacementEmailContent'.
        :param attachments: An optional list of attachment dicts for all
            recipients.
        :return: A list of BulkEmailEntryResult dicts with status and MessageId.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            template_content: Dict[str, Any] = {
                "TemplateName": template_name,
                "TemplateData": default_template_data,
            }

            if attachments:
                template_content["Attachments"] = attachments

            response = self.sesv2_client.send_bulk_email(
                FromEmailAddress=from_address,
                DefaultContent={"Template": template_content},
                BulkEmailEntries=bulk_entries,
            )
            results = response.get("BulkEmailEntryResults", [])
            logger.info(
                "Sent bulk email from %s to %d recipients.",
                from_address,
```

```
                len(bulk_entries),
            )
            return results
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Bulk message was rejected. Check that the template "
                    "exists, attachment file types are supported, and "
                    "total message size is within limits. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send bulk email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a SendBulkEmail Referência da API AWS SDK for Python (Boto3).

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem a todos os membros da lista de contatos.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
```

```
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
```

Envia uma mensagem a todos os membros da lista de contatos usando um modelo.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )
```

- Para obter detalhes da API, consulte a SendEmail Referência da API AWS SDK for Python (Boto3).

Cenários

Cenário de anexos de e-mail

O exemplo de código a seguir mostra como enviar e-mails com anexos usando a API v2 do Amazon SES.

- Crie um modelo de e-mail para envios em massa.
- Envie um e-mail simples com um anexo de arquivo.
- Envie um e-mail simples com uma imagem embutida.
- Envie e-mails modelados em massa com anexos.
- Limpe recursos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando anexos de e-mail.

```
class SESv2EmailAttachmentsScenario:
    """
    Demonstrates sending emails with attachments using Amazon SESv2.

    This scenario demonstrates:
    1. Setting up an email identity and template.
    2. Sending a simple email with a file attachment.
    3. Sending a simple email with an inline image.
    4. Sending bulk templated emails with attachments.
    5. Cleaning up created resources.
    """

    TEMPLATE_NAME = "AttachmentDemoTemplate"

    def __init__(self, sesv2_wrapper: SESv2Wrapper) -> None:
        """
        :param sesv2_wrapper: An instance of the SESv2Wrapper class.
```

```
        """
        self.sesv2_wrapper = sesv2_wrapper
        self.sender_email = ""
        self.recipient_emails: list = []
        self.identity_was_created = False

    def run_scenario(self) -> None:
        """Runs the SESv2 email attachments scenario."""
        print("-" * 88)
        print("Welcome to the Amazon SESv2 Email Attachments Scenario.")
        print("-" * 88)
        print(
            "This scenario demonstrates how to send emails with attachments\n"
            "using SESv2 attachment support. SES handles MIME\n"
            "construction automatically, so you don't need to build raw\n"
            "MIME messages.\n"
        )

        try:
            self._setup()
            self._step1_send_email_with_attachment()
            self._step2_send_email_with_inline_image()
            self._step3_send_bulk_email_with_attachments()
        except Exception as e:
            logger.error("Scenario failed: %s", e)
            print(f"\nThe scenario encountered an error: {e}")
        finally:
            self._cleanup()

    # ---------- Setup ----------

    def _setup(self) -> None:
        """
        Prompts for configuration, verifies the sender identity, prepares a
        sample attachment, and creates an email template.
        """
        print("\n--- Setup ---\n")

        # Prompt for sender and recipient addresses.
        print(
            "Both sender and recipient addresses must be verified if your\n"
            "account is in the SES sandbox.\n"
        )
        self.sender_email = q.ask(
```

```
            "Enter a verified sender email address: "
        )
        recipient_input = q.ask(
            "Enter one or more recipient email addresses (comma-separated): "
        )
        self.recipient_emails = [
            addr.strip() for addr in recipient_input.split(",") if addr.strip()
        ]

        # Verify the sender identity.
        print(f"\nChecking identity for {self.sender_email}...")
        try:
            identity_info = self.sesv2_wrapper.get_email_identity(
                self.sender_email
            )
            verified = identity_info.get("VerifiedForSendingStatus", False)
            if verified:
                print(f"  {self.sender_email} is verified and ready to send.")
            else:
                print(
                    f"  {self.sender_email} exists but is not yet verified."
                )
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                print(
                    f"  Identity {self.sender_email} not found. "
                    "Creating it now..."
                )
                result = self.sesv2_wrapper.create_email_identity(
                    self.sender_email
                )
                self.identity_was_created = True
                print(
                    f"  Identity created. Verification status: "
                    f"{result.get('VerifiedForSendingStatus', False)}"
                )
                print(
                    "  Check your inbox and click the verification link "
                    "before continuing."
                )
                q.ask("Press Enter when you have verified the address...")
            else:
                raise
```

```
        # Create the email template for the bulk-send step.
        print("\nCreating email template for the bulk email step...")
        try:
            self.sesv2_wrapper.create_email_template(
                template_name=self.TEMPLATE_NAME,
                subject="Bulk Email with Attachment for {{name}}",
                html_body=(
                    "<h1>Hello {{name}}</h1>"
                    "<p>Please find the attached document.</p>"
                ),
                text_body=(
                    "Hello {{name}}, Please find the attached document."
                ),
            )
            print(f"  Template '{self.TEMPLATE_NAME}' created.\n")
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                print(
                    f"  Template '{self.TEMPLATE_NAME}' already exists. "
                    "Using it.\n"
                )
            else:
                raise

    # ---------- Step 1: Simple email with file attachment ----------

    def _step1_send_email_with_attachment(self) -> None:
        """Sends a simple email with a text file attachment."""
        print("\n--- Step 1: Send a Simple Email with a File Attachment ---\n")
        print(
            "Creating a sample text file attachment and sending it with\n"
            "the Simple email content type. SES constructs the MIME message\n"
            "automatically.\n"
        )

        # Prepare a sample text file as bytes.
        sample_content = b"This is a sample report attachment."

        attachment = {
            "RawContent": sample_content,
            "FileName": "sample-report.txt",
            "ContentType": "text/plain",
            "ContentDisposition": "ATTACHMENT",
            "ContentDescription": "Sample report text file",
```

```
            "ContentTransferEncoding": "BASE64",
        }

        print(
            "Note: When using an AWS SDK, the SDK handles base64 encoding\n"
            "automatically. Direct API callers must encode content themselves.\n"
        )

        message_id = self.sesv2_wrapper.send_email(
            from_address=self.sender_email,
            to_addresses=self.recipient_emails,
            subject="SESv2 Attachment Demo - Simple Email with Attachment",
            html_body=(
                "<h1>Attachment Demo</h1>"
                "<p>Please see the attached <b>report document</b>.</p>"
            ),
            text_body="Please see the attached report document.",
            attachments=[attachment],
        )

        print(f"  Email sent. MessageId: {message_id}")
        print(
            "  SES automatically constructed the MIME message with the "
            "attachment.\n"
        )

    # ---------- Step 2: Simple email with inline image ----------

    def _step2_send_email_with_inline_image(self) -> None:
        """Sends a simple email with an inline image that renders in HTML."""
        print("\n--- Step 2: Send a Simple Email with an Inline Image ---\n")
        print(
            "This step demonstrates INLINE disposition. The image renders\n"
            "directly in the HTML body using a 'cid:' reference instead of\n"
            "appearing as a downloadable attachment.\n"
        )

        # Create a minimal 1x1 red PNG (valid PNG file).
        sample_image = (
            b"\x89PNG\r\n\x1a\n"  # PNG signature
            b"\x00\x00\x00\rIHDR\x00\x00\x00\x01\x00\x00\x00\x01"
            b"\x08\x02\x00\x00\x00\x90wS\xde"  # 1x1 RGB
            b"\x00\x00\x00\x0cIDATx\x9cc\xf8\x0f\x00\x00\x01\x01"
            b"\x00\x05\x18\xd8N"  # compressed data
```

```
            b"\x00\x00\x00\x00IEND\xaeB`\x82"  # IEND
        )

        attachment = {
            "RawContent": sample_image,
            "FileName": "logo.png",
            "ContentType": "image/png",
            "ContentDisposition": "INLINE",
            "ContentId": "logo123",
            "ContentDescription": "Company logo",
            "ContentTransferEncoding": "BASE64",
        }

        html_body = (
            "<html><body>"
            "<h1>Inline Image Demo</h1>"
            "<p>Here is our logo:</p>"
            '<img src="cid:logo123" alt="Company Logo">'
            "</body></html>"
        )

        message_id = self.sesv2_wrapper.send_email(
            from_address=self.sender_email,
            to_addresses=self.recipient_emails,
            subject="SESv2 Attachment Demo - Inline Image",
            html_body=html_body,
            text_body=(
                "Inline Image Demo - Please view this email in an "
                "HTML-capable client to see the embedded image."
            ),
            attachments=[attachment],
        )

        print(f"  Email sent. MessageId: {message_id}")
        print(
            "  The ContentId 'logo123' is referenced in the HTML body via\n"
            "  'cid:logo123', which lets the image render inline.\n"
        )

    # ---------- Step 3: Bulk templated email with attachments ----------

    def _step3_send_bulk_email_with_attachments(self) -> None:
        """Sends bulk templated emails with attachments to multiple recipients."""
        print("\n--- Step 3: Send Bulk Templated Emails with Attachments ---\n")
```

```
        print(
            "Using SendBulkEmail to send a templated email with an attachment\n"
            "to multiple recipients in a single API call. Each recipient gets\n"
            "personalized content via template data.\n"
        )

        sample_content = b"This is a sample report attachment."

        attachment = {
            "RawContent": sample_content,
            "FileName": "sample-report.txt",
            "ContentType": "text/plain",
            "ContentDisposition": "ATTACHMENT",
            "ContentDescription": "Sample report for bulk recipients",
            "ContentTransferEncoding": "BASE64",
        }

        # Build one entry per recipient with personalized names.
        names = ["Alice", "Bob", "Charlie", "Diana", "Eve"]
        bulk_entries = []
        for i, email in enumerate(self.recipient_emails):
            name = names[i] if i < len(names) else f"Recipient{i + 1}"
            bulk_entries.append(
                {
                    "Destination": {"ToAddresses": [email]},
                    "ReplacementEmailContent": {
                        "ReplacementTemplate": {
                            "ReplacementTemplateData": json.dumps(
                                {"name": name}
                            )
                        }
                    },
                }
            )

        results = self.sesv2_wrapper.send_bulk_email(
            from_address=self.sender_email,
            template_name=self.TEMPLATE_NAME,
            default_template_data='{"name": "Valued Customer"}',
            bulk_entries=bulk_entries,
            attachments=[attachment],
        )

        print("  Bulk email results:")
```

```
        for idx, result in enumerate(results):
            status = result.get("Status", "Unknown")
            msg_id = result.get("MessageId", "N/A")
            error = result.get("Error", "")
            recipient = (
                self.recipient_emails[idx]
                if idx < len(self.recipient_emails)
                else "Unknown"
            )
            print(f"    {recipient}: Status={status}, MessageId={msg_id}")
            if error:
                print(f"      Error: {error}")

        print(
            "\n  All recipients receive the same attachment(s) defined in\n"
            "  DefaultContent. Template data is personalized per recipient.\n"
        )

    # ---------- Cleanup ----------

    def _cleanup(self) -> None:
        """Deletes the email template and optionally the email identity."""
        print("\n--- Cleanup ---\n")

        # Delete the email template.
        try:
            self.sesv2_wrapper.delete_email_template(self.TEMPLATE_NAME)
            print(f"  Template '{self.TEMPLATE_NAME}' deleted.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                print(
                    f"  Template '{self.TEMPLATE_NAME}' was already deleted."
                )
            else:
                logger.error("Failed to delete template: %s", err)

        # Optionally delete the email identity.
        if self.identity_was_created and self.sender_email:
            delete_identity = q.ask(
                f"Delete the email identity '{self.sender_email}'? (y/n) ",
                q.is_yesno,
            )
            if delete_identity:
                try:
```

```
                    self.sesv2_wrapper.delete_email_identity(
                        self.sender_email
                    )
                    print(
                        f"  Email identity '{self.sender_email}' deleted."
                    )
                except ClientError as err:
                    if err.response["Error"]["Code"] == "NotFoundException":
                        print(
                            f"  Identity '{self.sender_email}' was "
                            "already deleted."
                        )
                    else:
                        logger.error(
                            "Failed to delete identity: %s", err
                        )
            else:
                print(
                    f"  Skipping identity deletion for {self.sender_email}."
                )
        else:
            print(
                "  Sender identity was pre-existing. Skipping deletion."
            )

        print("\nAll resources have been cleaned up.")
        print("-" * 88)
```

Crie uma classe de wrapper SESv2 para gerenciar as operações.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client
```

```
    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def get_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Gets information about an email identity, including its verification status.

        :param email_address: The email address or domain to look up.
        :return: A dictionary with identity information including verification
status.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            response = self.sesv2_client.get_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Got email identity for %s.", email_address)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found.", email_address
                )
            else:
                logger.error(
                    "Couldn't get email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def create_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
```

```
        Starts the process of verifying an email identity (email address or domain).

        :param email_address: The email address or domain to verify.
        :return: A dictionary with the identity type and verification status.
        :raises ClientError: If the limit is exceeded (LimitExceededException).
        """
        try:
            response = self.sesv2_client.create_email_identity(
                EmailIdentity=email_address
            )
            logger.info(
                "Started verification for email identity %s.", email_address
            )
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email identity %s. You have exceeded "
                    "the maximum number of email identities. "
                    "Use an existing verified identity.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't create email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def create_email_template(
        self,
        template_name: str,
        subject: str,
        html_body: str,
        text_body: str,
    ) -> None:
        """
        Creates an email template for use with templated and bulk email sends.

        :param template_name: The name for the new template.
```

```
        :param subject: The subject line of the template. May include
{{placeholders}}.
        :param html_body: The HTML body of the template.
        :param text_body: The plain text body of the template.
        :raises ClientError: If the template limit is exceeded
(LimitExceededException).
        """
        try:
            self.sesv2_client.create_email_template(
                TemplateName=template_name,
                TemplateContent={
                    "Subject": subject,
                    "Html": html_body,
                    "Text": text_body,
                },
            )
            logger.info("Created email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email template %s. You have exceeded "
                    "the maximum number of email templates. "
                    "Delete unused templates first.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't create email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def send_email(
        self,
        from_address: str,
        to_addresses: List[str],
        subject: str,
        html_body: str,
        text_body: str,
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> str:
```

```
        """
        Sends a simple email message with optional attachments.

        SES handles MIME construction automatically when using attachments
        with the Simple content type, so developers don't need to build
        raw MIME messages.

        :param from_address: The verified sender email address.
        :param to_addresses: A list of recipient email addresses.
        :param subject: The subject line of the email.
        :param html_body: The HTML body content.
        :param text_body: The plain text body content.
        :param attachments: An optional list of attachment dictionaries. Each
            attachment should contain 'RawContent' (bytes), 'FileName' (str),
            and optionally 'ContentType', 'ContentDisposition', 'ContentId',
            'ContentDescription', and 'ContentTransferEncoding'.
        :return: The MessageId of the sent email.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            simple_message: Dict[str, Any] = {
                "Subject": {"Data": subject},
                "Body": {
                    "Html": {"Data": html_body},
                    "Text": {"Data": text_body},
                },
            }

            if attachments:
                simple_message["Attachments"] = attachments

            response = self.sesv2_client.send_email(
                FromEmailAddress=from_address,
                Destination={"ToAddresses": to_addresses},
                Content={"Simple": simple_message},
            )
            message_id = response["MessageId"]
            logger.info(
                "Sent email from %s to %s. MessageId: %s",
                from_address,
                to_addresses,
                message_id,
            )
            return message_id
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Message was rejected. Check that attachments use "
                    "supported file types and total message size is "
                    "under 40 MB. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def send_bulk_email(
        self,
        from_address: str,
        template_name: str,
        default_template_data: str,
        bulk_entries: List[Dict[str, Any]],
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Sends a templated email to multiple recipients in a single API call.

        All recipients receive the same attachment(s) defined in the default
        content, while template data can be personalized per recipient.

        :param from_address: The verified sender email address.
        :param template_name: The name of an existing email template.
        :param default_template_data: Default JSON template data string.
        :param bulk_entries: A list of BulkEmailEntry dicts, each containing
            'Destination' and optionally 'ReplacementEmailContent'.
        :param attachments: An optional list of attachment dicts for all
            recipients.
        :return: A list of BulkEmailEntryResult dicts with status and MessageId.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            template_content: Dict[str, Any] = {
                "TemplateName": template_name,
```

```
                "TemplateData": default_template_data,
            }

            if attachments:
                template_content["Attachments"] = attachments

            response = self.sesv2_client.send_bulk_email(
                FromEmailAddress=from_address,
                DefaultContent={"Template": template_content},
                BulkEmailEntries=bulk_entries,
            )
            results = response.get("BulkEmailEntryResults", [])
            logger.info(
                "Sent bulk email from %s to %d recipients.",
                from_address,
                len(bulk_entries),
            )
            return results
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Bulk message was rejected. Check that the template "
                    "exists, attachment file types are supported, and "
                    "total message size is within limits. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send bulk email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def delete_email_template(self, template_name: str) -> None:
        """
        Deletes an email template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If the template is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_template(
```

```
                TemplateName=template_name
            )
            logger.info("Deleted email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email template %s not found or already deleted.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't delete email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def delete_email_identity(self, email_address: str) -> None:
        """
        Deletes an email identity.

        :param email_address: The email address or domain to delete.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Deleted email identity %s.", email_address)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found or already deleted.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't delete email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - GetEmailIdentity
  - SendBulkEmail
  - SendEmail

Cenário de boletim informativo

O exemplo de código a seguir mostra como executar o cenário do boletim informativo da API v2 do Amazon SES.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
```

```
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()




class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e

            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons Newsletter"
```

```
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
Skipping...")
                else:
                    raise e

        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
```

```
                    },
                )
                print(f"Welcome email sent to '{email}'.")

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )

        try:
            self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e

        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e

        try:
```

```
            self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)

            try:

self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not exist.")
                else:
                    print(e)

        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity

- DeleteEmailTemplate
- ListContacts
- SendEmail.simple
- SendEmail.template

Exemplos do Amazon SNS usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários
- Exemplos sem servidor

Ações

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.
```

```
        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
```

```
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a CreateTopic Referência da API AWS SDK for Python (Boto3).

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
```

```
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
```

```
            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise
```

- Para obter detalhes da API, consulte a DeleteTopic Referência da API AWS SDK for Python (Boto3).

`ListSubscriptions`

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are returned.
        :return: An iterator that yields the subscriptions.
```

```
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- Para obter detalhes da API, consulte a ListSubscriptions Referência da API AWS SDK for Python (Boto3).

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_topics(self):
```

```
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- Para obter detalhes da API, consulte a ListTopics Referência da API AWS SDK for Python (Boto3).

`Publicar`

O código de exemplo a seguir mostra como usar `Publish`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Publique uma mensagem com atributos para que uma assinatura possa filtrar com base em atributos.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
```

```
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message. Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

Publique uma mensagem que assume diferentes formas com base no protocolo do assinante.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version is
                                sent to subscribers that have protocols that are not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
```

```
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} - {e}")
            raise
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Python (Boto3).

`SetSubscriptionAttributes`

O código de exemplo a seguir mostra como usar `SetSubscriptionAttributes`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must have
 an
        attribute that passes the filter or it will not be sent to the subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy", AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- Para obter detalhes da API, consulte a SetSubscriptionAttributes Referência da API AWS SDK for Python (Boto3).

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone number
                         (in E.164 format) for SMS messages, or an email address for
                         email messages.
        :return: The newly added subscription.
        """
```

```
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

Inscreva uma fila em um tópico com filtros opcionais.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def subscribe_queue_to_topic(
        self,
```

```
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise
```

- Consulte detalhes da API em Subscribe na Referência da API do AWS SDK for Python (Boto3).

Cancelar assinatura

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.", subscription.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client
```

```
    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
                raise
```

- Consulte detalhes da API em Unsubscribe na Referência da API AWS SDK for Python (Boto3).

Cenários

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Criar e publicar em um tópico FIFO

O exemplo de código a seguir mostra como criar e publicar em um tópico FIFO do Amazon SNS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO do Amazon SNS, inscreva filas padrão e FIFO do Amazon SQS no tópico e publique uma mensagem no tópico.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
```

```
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)
```

```
    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII letters,
        numbers, underscores, and hyphens, and must be between 1 and 256 characters
 long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
```

```
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version":"2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
```

```
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
```

```
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTopic
  - Publicar
  - Assinar

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico.

Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.
```

```
        :param phone_number: The phone number that receives the message. This must
be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Python (Boto3).

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
        self.queue_count = 2
        self.queue_urls = []
        self.subscription_arns = []
        self.tones = ["cheerful", "funny", "serious", "sincere"]

    def run_scenario(self) -> None:
        """Run the Topics and Queues feature scenario."""
        print(self.DASHES)
        print("Welcome to messaging with topics and queues.")
        print(self.DASHES)
        print(f"""
    In this scenario, you will create an SNS topic and subscribe {self.queue_count}
 SQS queues to the topic.
    You can select from several options for configuring the topic and the
 subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
        """)

        try:
            # Setup Phase
            print(self.DASHES)
            self._setup_topic()
```

```
            print(self.DASHES)

            self._setup_queues()
            print(self.DASHES)

            # Demonstration Phase
            self._publish_messages()
            print(self.DASHES)

            # Examination Phase
            self._poll_queues_for_messages()
            print(self.DASHES)

            # Cleanup Phase
            self._cleanup_resources()
            print(self.DASHES)

        except Exception as e:
            logger.error(f"Scenario failed: {e}")
            print(f"There was a problem with the scenario: {e}")
            print("\nInitiating cleanup...")
            try:
                self._cleanup_resources()
            except Exception as cleanup_error:
                logger.error(f"Error during cleanup: {cleanup_error}")

        print("Messaging with topics and queues scenario is complete.")
        print(self.DASHES)

    def _setup_topic(self) -> None:
        """Set up the SNS topic to be used with the queues."""
        print("SNS topics can be configured as FIFO (First-In-First-Out).")
        print("FIFO topics deliver messages in order and support deduplication and
message filtering.")
        print()

        self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/n):
", q.is_yesno)

        if self.use_fifo_topic:
            print(self.DASHES)
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)
```

```
            print("Because you have selected a FIFO topic, '.fifo' must be appended
to the topic name.")
            print()

            print(self.DASHES)
            print("""
   Because you have chosen a FIFO topic, deduplication is supported.
   Deduplication IDs are either set in the message or automatically generated
   from content using a hash function.

   If a message is successfully published to an SNS FIFO topic, any message
   published and determined to have the same deduplication ID,
   within the five-minute deduplication interval, is accepted but not delivered.

   For more information about deduplication,
   see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
            """)

            self.use_content_based_deduplication = q.ask(
                "Use content-based deduplication instead of entering a deduplication
ID? (y/n): ",
                q.is_yesno
            )
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)

        print(self.DASHES)

        # Create the topic
        self.topic_arn = self.sns_wrapper.create_topic(
            self.topic_name,
            self.use_fifo_topic,
            self.use_content_based_deduplication
        )

        print(f"Your new topic with the name {self.topic_name}")
        print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
        print(f"  has been created.")
        print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
```

```
        print(f"Now you will create {self.queue_count} Amazon Simple Queue Service
(Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ", q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)

            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()

            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")
                print(self.DASHES)

            # Get queue ARN
            queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

            if i == 0:
                print("An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue,")
                print("enabling it to receive messages from an SNS topic.")

            # Set queue policy to allow SNS to send messages
            self.sqs_wrapper.set_queue_policy_for_topic(queue_arn, self.topic_arn,
queue_url)

            # Set up message filtering if using FIFO
            subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
queue_name)
            self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
queue_name: str) -> str:
```

```
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
 filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
                print()
                print("For this example, you can filter messages by a TONE
 attribute.")

            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription to
 the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
 {self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

    def _create_filter_policy(self) -> str:
        """Create a message filter policy based on user selections."""
        print(self.DASHES)
        print("You can filter messages by one or more of the following TONE
 attributes.")

        filter_selections = []
        selection_number = 0

        while True:
```

```
            print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
            for i, tone in enumerate(self.tones, 1):
                print(f"  {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

            if selection == 0:
                break
            elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
                filter_selections.append(self.tones[selection - 1])
                print(f"Added '{self.tones[selection - 1]}' to filter list.")

        if filter_selections:
            filters = {"tone": filter_selections}
            return json.dumps(filters)
        return None

    def _publish_messages(self) -> None:
        """Publish messages to the topic with various options."""
        print("Now we can publish messages.")

        keep_sending = True
        while keep_sending:
            print()
            message = q.ask("Enter a message to publish: ", q.non_empty)

            message_group_id = None
            deduplication_id = None
            tone_attribute = None

            if self.use_fifo_topic:
                print("Because you are using a FIFO topic, you must set a message
group ID.")
                print("All messages within the same group will be received in the
order they were published.")
                print()
                message_group_id = q.ask("Enter a message group ID for this message:
", q.non_empty)

                if not self.use_content_based_deduplication:
                    print("Because you are not using content-based deduplication,")
```

```
                    print("you must enter a deduplication ID.")
                    deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

                # Ask about tone attribute
                add_attribute = q.ask("Add an attribute to this message? (y/n): ",
q.is_yesno)
                if add_attribute:
                    print("Enter a number for an attribute:")
                    for i, tone in enumerate(self.tones, 1):
                        print(f"  {i}. {tone}")

                    selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
len(self.tones)))
                    if 1 <= selection <= len(self.tones):
                        tone_attribute = self.tones[selection - 1]

            # Publish the message
            message_id = self.sns_wrapper.publish_message(
                self.topic_arn,
                message,
                tone_attribute,
                deduplication_id,
                message_group_id
            )

            print(f"Message published with ID: {message_id}")

            keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

    def _poll_queues_for_messages(self) -> None:
        """Poll all queues for messages and display results."""
        for i, queue_url in enumerate(self.queue_urls):
            print(f"Polling queue #{i+1} at {queue_url} for messages...")

            q.ask("Press Enter to continue...")

            messages = self._poll_queue_for_messages(queue_url)

            if messages:
                print(f"{len(messages)} message(s) were received by queue #{i+1}")
                for j, message in enumerate(messages, 1):
                    print(f"  Message {j}:")
                    # Parse the SNS message body to get the actual message
```

```
                    try:
                        sns_message = json.loads(message['Body'])
                        actual_message = sns_message.get('Message', message['Body'])
                        print(f"    {actual_message}")
                    except (json.JSONDecodeError, KeyError):
                        print(f"    {message['Body']}")

                # Delete the messages
                self.sqs_wrapper.delete_messages(queue_url, messages)
                print(f"Messages deleted from queue #{i+1}")
            else:
                print(f"No messages received by queue #{i+1}")

            print(self.DASHES)

    def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
        """Poll a single queue for messages."""
        all_messages = []
        max_polls = 3  # Limit polling to avoid infinite loops

        for poll_count in range(max_polls):
            messages = self.sqs_wrapper.receive_messages(queue_url, 10)

            if messages:
                all_messages.extend(messages)
                print(f"  Received {len(messages)} messages in poll {poll_count +
 1}")
                # Small delay between polls
                time.sleep(1)
            else:
                print(f"  No messages in poll {poll_count + 1}")
                break

        return all_messages

    def _cleanup_resources(self) -> None:
        """Clean up all resources created during the scenario."""
        print("Cleaning up resources...")

        # Delete queues
        for i, queue_url in enumerate(self.queue_urls):
            if queue_url:
                delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}? (y/
n): ", q.is_yesno)
```

```
                if delete_queue:
                    try:
                        self.sqs_wrapper.delete_queue(queue_url)
                        print(f"Deleted queue #{i+1}")
                    except Exception as e:
                        print(f"Error deleting queue #{i+1}: {e}")

        # Unsubscribe from topic
        for i, subscription_arn in enumerate(self.subscription_arns):
            if subscription_arn:
                try:
                    self.sns_wrapper.unsubscribe(subscription_arn)
                    print(f"Unsubscribed subscription #{i+1}")
                except Exception as e:
                    print(f"Error unsubscribing #{i+1}: {e}")

        # Delete topic
        if self.topic_arn:
            delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
 q.is_yesno)
            if delete_topic:
                try:
                    self.sns_wrapper.delete_topic(self.topic_arn)
                    print(f"Deleted topic {self.topic_name}")
                except Exception as e:
                    print(f"Error deleting topic: {e}")

        print("Resource cleanup complete.")
```

Crie classes que envolvam as operações do Amazon SNS e do Amazon SQS para uso no cenário.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
```

```
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )
```

```
            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} - {e}")
            raise


    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
```

```
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id
```

```
            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} - {e}")
            raise


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
                raise


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
```

```
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise


    def list_topics(self) -> list:
        """
        List all SNS topics in the account using pagination.

        :return: List of topic ARNs.
        :raises ClientError: If listing topics fails.
        """
        try:
            topics = []
            paginator = self.sns_client.get_paginator('list_topics')

            for page in paginator.paginate():
                topics.extend([topic['TopicArn'] for topic in page.get('Topics',
[])])

            logger.info(f"Found {len(topics)} topics")
            return topics

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AuthorizationError':
                logger.error("Authorization error listing topics - check IAM
permissions")
            else:
```

```
                logger.error(f"Error listing topics: {error_code} - {e}")
            raise


class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'
```

```
            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
            raise


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str, queue_url:
str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.
```

```
        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise
```

```
    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True
```

```
            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)
```

```
            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise


    def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
        """
        List all SQS queues in the account using pagination.

        :param queue_name_prefix: Optional prefix to filter queue names.
        :return: List of queue URLs.
        :raises ClientError: If listing queues fails.
        """
        try:
            queue_urls = []
            paginator = self.sqs_client.get_paginator('list_queues')

            page_params = {}
            if queue_name_prefix:
                page_params['QueueNamePrefix'] = queue_name_prefix

            for page in paginator.paginate(**page_params):
                queue_urls.extend(page.get('QueueUrls', []))

            logger.info(f"Found {len(queue_urls)} queues")
            return queue_urls

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AccessDenied':
                logger.error("Access denied listing queues - check IAM permissions")
            else:
                logger.error(f"Error listing queues: {error_code} - {e}")
            raise
```

```
    def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
        """
        Send a message to an SQS queue.

        :param queue_url: The URL of the queue.
        :param message_body: The message content.
        :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
        :return: The message ID.
        :raises ClientError: If sending the message fails.
        """
        try:
            send_params = {
                'QueueUrl': queue_url,
                'MessageBody': message_body,
                **kwargs
            }

            response = self.sqs_client.send_message(**send_params)

            message_id = response['MessageId']
            logger.info(f"Sent message to {queue_url} with ID: {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error sending message: {error_code} - {e}")
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar

- ReceiveMessage
- SetQueueAttributes
- Assinar
- Cancelar assinatura

Usar o API Gateway para invocar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada pelo Amazon API Gateway.

SDK para Python (Boto3)

Este exemplo mostra como criar e usar uma API REST do Amazon API Gateway cujo alvo é uma função do AWS Lambda . O manipulador do Lambda mostra como rotear com base em métodos HTTP; como obter dados da string de consulta, do cabeçalho e do corpo e como retornar uma resposta JSON.

- Implante uma função do Lambda.
- Crie uma API REST do API Gateway.
- Criar um recurso REST cujo alvo seja a função do Lambda.
- Conceda permissão para que o API Gateway possa invocar a função do Lambda.
- Use o pacote Requests para enviar solicitações à API REST.
- Limpe todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Usar eventos programados para chamar uma função do Lambda

O exemplo de código a seguir mostra como criar uma AWS Lambda função invocada por um evento EventBridge agendado pela Amazon.

SDK para Python (Boto3)

Este exemplo mostra como registrar uma AWS Lambda função como alvo de um EventBridge evento agendado da Amazon. O manipulador do Lambda grava uma mensagem amigável e os dados completos do evento no Amazon CloudWatch Logs para recuperação posterior.

- Implanta uma função do Lambda.
- Cria um evento EventBridge agendado e torna a função do Lambda o alvo.
- Concede permissão para permitir que a função Lambda seja EventBridge invocada.
- Imprime os dados mais recentes do CloudWatch Logs para mostrar o resultado das invocações agendadas.
- Limpa todos os recursos criados durante a demonstração.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- CloudWatch Registros
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Exemplos do Amazon SQS usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários
- Exemplos sem servidor

Ações

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_queue(name, attributes=None):
    """
    Creates an Amazon SQS queue.

    :param name: The name of the queue. This is part of the URL assigned to the
 queue.
    :param attributes: The attributes of the queue, such as maximum message size or
                       whether it's a FIFO queue.
    :return: A Queue object that contains metadata about the queue and that can be
 used
             to perform queue operations like sending and receiving messages.
    """
    if not attributes:
        attributes = {}

    try:
        queue = sqs.create_queue(QueueName=name, Attributes=attributes)
        logger.info("Created queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't create queue named '%s'.", name)
        raise error
    else:
        return queue
```

```
class SqsWrapper:
```

```
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )
```

```
            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a CreateQueue Referência da API AWS SDK for Python (Boto3).

`DeleteMessage`

O código de exemplo a seguir mostra como usar `DeleteMessage`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_message(message):
    """
    Delete a message from a queue. Clients must delete messages after they
    are received and processed to remove them from the queue.

    :param message: The message to delete. The message's queue URL is contained in
                    the message's metadata.
    :return: None
    """
    try:
        message.delete()
        logger.info("Deleted message: %s", message.message_id)
    except ClientError as error:
        logger.exception("Couldn't delete message: %s", message.message_id)
        raise error
```

- Para obter detalhes da API, consulte a DeleteMessage Referência da API AWS SDK for Python (Boto3).

`DeleteMessageBatch`

O código de exemplo a seguir mostra como usar `DeleteMessageBatch`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and failed
             message deletions.
    """
    try:
        entries = [
            {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
            for ind, msg in enumerate(messages)
        ]
        response = queue.delete_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info("Deleted %s",
 messages[int(msg_meta["Id"])].receipt_handle)
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
```

```
                    "Could not delete %s",
 messages[int(msg_meta["Id"])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
 bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
```

```
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a DeleteMessageBatch Referência da API AWS SDK for Python (Boto3).

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def remove_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.
```

```
        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise
```

- Para obter detalhes da API, consulte a DeleteQueue Referência da API AWS SDK for Python (Boto3).

`GetQueueAttributes`

O código de exemplo a seguir mostra como usar `GetQueueAttributes`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
```

```
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a GetQueueAttributes Referência da API AWS SDK for Python (Boto3).

`GetQueueUrl`

O código de exemplo a seguir mostra como usar `GetQueueUrl`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_queue(name):
    """
    Gets an SQS queue by name.

    :param name: The name that was used to create the queue.
    :return: A Queue object.
    """
    try:
        queue = sqs.get_queue_by_name(QueueName=name)
        logger.info("Got queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't get queue named %s.", name)
        raise error
```

```
    else:
        return queue
```

- Para obter detalhes da API, consulte a GetQueueUrl Referência da API AWS SDK for Python (Boto3).

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_queues(prefix=None):
    """
    Gets a list of SQS queues. When a prefix is specified, only queues with names
    that start with the prefix are returned.

    :param prefix: The prefix used to restrict the list of returned queues.
    :return: A list of Queue objects.
    """
    if prefix:
        queue_iter = sqs.queues.filter(QueueNamePrefix=prefix)
    else:
        queue_iter = sqs.queues.all()
    queues = list(queue_iter)
    if queues:
        logger.info("Got queues: %s", ", ".join([q.url for q in queues]))
    else:
        logger.warning("No queues found.")
    return queues
```

- Para obter detalhes da API, consulte a ListQueues Referência da API AWS SDK for Python (Boto3).

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning. When
                      this number is greater than zero, long polling is used. This
                      can result in reduced costs and fewer false empty responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
```

```
    else:
        return messages
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
 List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
```

```
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a ReceiveMessage Referência da API AWS SDK for Python (Boto3).

`SendMessage`

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def send_message(queue, message_body, message_attributes=None):
    """
    Send a message to an Amazon SQS queue.

    :param queue: The queue that receives the message.
    :param message_body: The body text of the message.
    :param message_attributes: Custom attributes of the message. These are key-value
                               pairs that can be whatever you want.
    :return: The response from SQS that contains the assigned message ID.
```

```
    """
    if not message_attributes:
        message_attributes = {}

    try:
        response = queue.send_message(
            MessageBody=message_body, MessageAttributes=message_attributes
        )
    except ClientError as error:
        logger.exception("Send message failed: %s", message_body)
        raise error
    else:
        return response
```

- Para obter detalhes da API, consulte a SendMessage Referência da API AWS SDK for Python (Boto3).

`SendMessageBatch`

O código de exemplo a seguir mostra como usar `SendMessageBatch`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
```

```
                    contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and failed
             messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
            for ind, msg in enumerate(messages)
        ]
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response
```

- Para obter detalhes da API, consulte a SendMessageBatch Referência da API AWS SDK for Python (Boto3).

`SetQueueAttributes`

O código de exemplo a seguir mostra como usar `SetQueueAttributes`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Definir o atributo de política de uma fila para um tópico.

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str, queue_url:
 str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
```

```
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise
```

- Para obter detalhes da API, consulte a SetQueueAttributes Referência da API AWS SDK for Python (Boto3).

Cenários

Criar uma aplicação de mensageiro

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.
- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SQS
- Funções de etapa

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Criar e publicar em um tópico FIFO

O exemplo de código a seguir mostra como criar e publicar em um tópico FIFO do Amazon SNS.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO do Amazon SNS, inscreva filas padrão e FIFO do Amazon SQS no tópico e publique uma mensagem no tópico.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")
```

```
    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)
```

```
class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII letters,
        numbers, underscores, and hyphens, and must be between 1 and 256 characters
 long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.
```

```
        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version":"2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
```

```
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.
```

```
        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateTopic
  - Publicar
  - Assinar

Detectar pessoas e objetos em um vídeo

O exemplo de código a seguir mostra como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

SDK para Python (Boto3)

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado em GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition

- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper
```

```
        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
        self.queue_count = 2
        self.queue_urls = []
        self.subscription_arns = []
        self.tones = ["cheerful", "funny", "serious", "sincere"]

    def run_scenario(self) -> None:
        """Run the Topics and Queues feature scenario."""
        print(self.DASHES)
        print("Welcome to messaging with topics and queues.")
        print(self.DASHES)
        print(f"""
    In this scenario, you will create an SNS topic and subscribe {self.queue_count}
SQS queues to the topic.
    You can select from several options for configuring the topic and the
subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
        """)

        try:
            # Setup Phase
            print(self.DASHES)
            self._setup_topic()
            print(self.DASHES)

            self._setup_queues()
            print(self.DASHES)

            # Demonstration Phase
            self._publish_messages()
            print(self.DASHES)

            # Examination Phase
            self._poll_queues_for_messages()
            print(self.DASHES)

            # Cleanup Phase
            self._cleanup_resources()
            print(self.DASHES)
```

```
        except Exception as e:
            logger.error(f"Scenario failed: {e}")
            print(f"There was a problem with the scenario: {e}")
            print("\nInitiating cleanup...")
            try:
                self._cleanup_resources()
            except Exception as cleanup_error:
                logger.error(f"Error during cleanup: {cleanup_error}")

        print("Messaging with topics and queues scenario is complete.")
        print(self.DASHES)

    def _setup_topic(self) -> None:
        """Set up the SNS topic to be used with the queues."""
        print("SNS topics can be configured as FIFO (First-In-First-Out).")
        print("FIFO topics deliver messages in order and support deduplication and
message filtering.")
        print()

        self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/n):
", q.is_yesno)

        if self.use_fifo_topic:
            print(self.DASHES)
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)
            print("Because you have selected a FIFO topic, '.fifo' must be appended
to the topic name.")
            print()

            print(self.DASHES)
            print("""
    Because you have chosen a FIFO topic, deduplication is supported.
    Deduplication IDs are either set in the message or automatically generated
    from content using a hash function.

    If a message is successfully published to an SNS FIFO topic, any message
    published and determined to have the same deduplication ID,
    within the five-minute deduplication interval, is accepted but not delivered.

    For more information about deduplication,
    see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
            """)
```

```
            self.use_content_based_deduplication = q.ask(
                "Use content-based deduplication instead of entering a deduplication
ID? (y/n): ",
                q.is_yesno
            )
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)

        print(self.DASHES)

        # Create the topic
        self.topic_arn = self.sns_wrapper.create_topic(
            self.topic_name,
            self.use_fifo_topic,
            self.use_content_based_deduplication
        )

        print(f"Your new topic with the name {self.topic_name}")
        print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
        print(f"  has been created.")
        print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
        print(f"Now you will create {self.queue_count} Amazon Simple Queue Service
(Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ", q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)

            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()
```

```
            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")
                print(self.DASHES)

            # Get queue ARN
            queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

            if i == 0:
                print("An AWS Identity and Access Management (IAM) policy must be
 attached to an SQS queue,")
                print("enabling it to receive messages from an SNS topic.")

            # Set queue policy to allow SNS to send messages
            self.sqs_wrapper.set_queue_policy_for_topic(queue_arn, self.topic_arn,
 queue_url)

            # Set up message filtering if using FIFO
            subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
 queue_name)
            self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
 queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
 filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
                print()
                print("For this example, you can filter messages by a TONE
 attribute.")
```

```
            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription to
the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
{self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

    def _create_filter_policy(self) -> str:
        """Create a message filter policy based on user selections."""
        print(self.DASHES)
        print("You can filter messages by one or more of the following TONE
attributes.")

        filter_selections = []
        selection_number = 0

        while True:
            print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
            for i, tone in enumerate(self.tones, 1):
                print(f"  {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

            if selection == 0:
                break
            elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
                filter_selections.append(self.tones[selection - 1])
                print(f"Added '{self.tones[selection - 1]}' to filter list.")

        if filter_selections:
            filters = {"tone": filter_selections}
            return json.dumps(filters)
```

```
        return None

    def _publish_messages(self) -> None:
        """Publish messages to the topic with various options."""
        print("Now we can publish messages.")

        keep_sending = True
        while keep_sending:
            print()
            message = q.ask("Enter a message to publish: ", q.non_empty)

            message_group_id = None
            deduplication_id = None
            tone_attribute = None

            if self.use_fifo_topic:
                print("Because you are using a FIFO topic, you must set a message
group ID.")
                print("All messages within the same group will be received in the
order they were published.")
                print()
                message_group_id = q.ask("Enter a message group ID for this message:
", q.non_empty)

                if not self.use_content_based_deduplication:
                    print("Because you are not using content-based deduplication,")
                    print("you must enter a deduplication ID.")
                    deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

                # Ask about tone attribute
                add_attribute = q.ask("Add an attribute to this message? (y/n): ",
q.is_yesno)
                if add_attribute:
                    print("Enter a number for an attribute:")
                    for i, tone in enumerate(self.tones, 1):
                        print(f"  {i}. {tone}")

                    selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
len(self.tones)))
                    if 1 <= selection <= len(self.tones):
                        tone_attribute = self.tones[selection - 1]

            # Publish the message
```

```
            message_id = self.sns_wrapper.publish_message(
                self.topic_arn,
                message,
                tone_attribute,
                deduplication_id,
                message_group_id
            )

            print(f"Message published with ID: {message_id}")

            keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

    def _poll_queues_for_messages(self) -> None:
        """Poll all queues for messages and display results."""
        for i, queue_url in enumerate(self.queue_urls):
            print(f"Polling queue #{i+1} at {queue_url} for messages...")

            q.ask("Press Enter to continue...")

            messages = self._poll_queue_for_messages(queue_url)

            if messages:
                print(f"{len(messages)} message(s) were received by queue #{i+1}")
                for j, message in enumerate(messages, 1):
                    print(f"  Message {j}:")
                    # Parse the SNS message body to get the actual message
                    try:
                        sns_message = json.loads(message['Body'])
                        actual_message = sns_message.get('Message', message['Body'])
                        print(f"    {actual_message}")
                    except (json.JSONDecodeError, KeyError):
                        print(f"    {message['Body']}")

                # Delete the messages
                self.sqs_wrapper.delete_messages(queue_url, messages)
                print(f"Messages deleted from queue #{i+1}")
            else:
                print(f"No messages received by queue #{i+1}")

            print(self.DASHES)

    def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
        """Poll a single queue for messages."""
        all_messages = []
```

```
        max_polls = 3  # Limit polling to avoid infinite loops

        for poll_count in range(max_polls):
            messages = self.sqs_wrapper.receive_messages(queue_url, 10)

            if messages:
                all_messages.extend(messages)
                print(f"  Received {len(messages)} messages in poll {poll_count +
 1}")
                # Small delay between polls
                time.sleep(1)
            else:
                print(f"  No messages in poll {poll_count + 1}")
                break

        return all_messages

    def _cleanup_resources(self) -> None:
        """Clean up all resources created during the scenario."""
        print("Cleaning up resources...")

        # Delete queues
        for i, queue_url in enumerate(self.queue_urls):
            if queue_url:
                delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}? (y/
n): ", q.is_yesno)
                if delete_queue:
                    try:
                        self.sqs_wrapper.delete_queue(queue_url)
                        print(f"Deleted queue #{i+1}")
                    except Exception as e:
                        print(f"Error deleting queue #{i+1}: {e}")

        # Unsubscribe from topic
        for i, subscription_arn in enumerate(self.subscription_arns):
            if subscription_arn:
                try:
                    self.sns_wrapper.unsubscribe(subscription_arn)
                    print(f"Unsubscribed subscription #{i+1}")
                except Exception as e:
                    print(f"Error unsubscribing #{i+1}: {e}")

        # Delete topic
        if self.topic_arn:
```

```
            delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
 q.is_yesno)
            if delete_topic:
                try:
                    self.sns_wrapper.delete_topic(self.topic_arn)
                    print(f"Deleted topic {self.topic_name}")
                except Exception as e:
                    print(f"Error deleting topic: {e}")

        print("Resource cleanup complete.")
```

Crie classes que envolvam as operações do Amazon SNS e do Amazon SQS para uso no cenário.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
```

```
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} - {e}")
            raise


    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
```

```
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
```

```
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} - {e}")
            raise


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.
```

```
        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
                raise


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise
```

```
    def list_topics(self) -> list:
        """
        List all SNS topics in the account using pagination.

        :return: List of topic ARNs.
        :raises ClientError: If listing topics fails.
        """
        try:
            topics = []
            paginator = self.sns_client.get_paginator('list_topics')

            for page in paginator.paginate():
                topics.extend([topic['TopicArn'] for topic in page.get('Topics',
 [])])

            logger.info(f"Found {len(topics)} topics")
            return topics

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AuthorizationError':
                logger.error("Authorization error listing topics - check IAM
 permissions")
            else:
                logger.error(f"Error listing topics: {error_code} - {e}")
            raise


class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
```

```
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
            raise


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.
```

```
        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str, queue_url:
str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
```

```
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
```

```
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )
```

```
            # Check for failures
            if 'Failed' in response and response['Failed']:
                for failed in response['Failed']:
                    logger.warning(f"Failed to delete message: {failed}")

        logger.info(f"Deleted {len(messages)} messages from {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error deleting messages: {error_code} - {e}")
        raise


def delete_queue(self, queue_url: str) -> bool:
    """
    Delete an SQS queue.

    :param queue_url: The URL of the queue to delete.
    :return: True if successful.
    :raises ClientError: If the queue deletion fails.
    """
    try:
        self.sqs_client.delete_queue(QueueUrl=queue_url)

        logger.info(f"Deleted queue: {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
            logger.warning(f"Queue not found: {queue_url}")
            return True  # Already deleted
        else:
            logger.error(f"Error deleting queue: {error_code} - {e}")
            raise


def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
    """
    List all SQS queues in the account using pagination.
```

```
        :param queue_name_prefix: Optional prefix to filter queue names.
        :return: List of queue URLs.
        :raises ClientError: If listing queues fails.
        """
        try:
            queue_urls = []
            paginator = self.sqs_client.get_paginator('list_queues')

            page_params = {}
            if queue_name_prefix:
                page_params['QueueNamePrefix'] = queue_name_prefix

            for page in paginator.paginate(**page_params):
                queue_urls.extend(page.get('QueueUrls', []))

            logger.info(f"Found {len(queue_urls)} queues")
            return queue_urls

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AccessDenied':
                logger.error("Access denied listing queues - check IAM permissions")
            else:
                logger.error(f"Error listing queues: {error_code} - {e}")
            raise

    def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
        """
        Send a message to an SQS queue.

        :param queue_url: The URL of the queue.
        :param message_body: The message content.
        :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
        :return: The message ID.
        :raises ClientError: If sending the message fails.
        """
        try:
            send_params = {
                'QueueUrl': queue_url,
                'MessageBody': message_body,
                **kwargs
            }
```

```
            response = self.sqs_client.send_message(**send_params)

            message_id = response['MessageId']
            logger.info(f"Sent message to {queue_url} with ID: {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error sending message: {error_code} - {e}")
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

Enviar e receber lotes de mensagens

O exemplo de código a seguir mostra como:

- Criar uma fila do Amazon SQS.
- Enviar lotes de mensagens para a fila.
- Receber lotes de mensagens de uma fila.
- Excluir lotes de mensagens de uma fila.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Criar funções para encapsular funções de mensagem do Amazon SQS.

```
import logging
import sys

import boto3
from botocore.exceptions import ClientError

import queue_wrapper

logger = logging.getLogger(__name__)
sqs = boto3.resource("sqs")

def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and failed
             messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
            for ind, msg in enumerate(messages)
        ]
```

```
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response



def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning. When
                      this number is greater than zero, long polling is used. This
                      can result in reduced costs and fewer false empty responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
```

```
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return messages



def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and failed
             message deletions.
    """
    try:
        entries = [
            {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
            for ind, msg in enumerate(messages)
        ]
        response = queue.delete_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info("Deleted %s",
 messages[int(msg_meta["Id"])].receipt_handle)
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Could not delete %s",
 messages[int(msg_meta["Id"])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

Use as funções de wrapper para enviar e receber mensagens em lotes.

```
def usage_demo():
    """
    Shows how to:
    * Read the lines from this Python file and send the lines in
      batches of 10 as messages to a queue.
    * Receive the messages in batches until the queue is empty.
    * Reassemble the lines of the file and verify they match the original file.
    """

    def pack_message(msg_path, msg_body, msg_line):
        return {
            "body": msg_body,
            "attributes": {
                "path": {"StringValue": msg_path, "DataType": "String"},
                "line": {"StringValue": str(msg_line), "DataType": "String"},
            },
        }

    def unpack_message(msg):
        return (
            msg.message_attributes["path"]["StringValue"],
            msg.body,
            int(msg.message_attributes["line"]["StringValue"]),
        )

    print("-" * 88)
    print("Welcome to the Amazon Simple Queue Service (Amazon SQS) demo!")
    print("-" * 88)

    queue = queue_wrapper.create_queue("sqs-usage-demo-message-wrapper")

    with open(__file__) as file:
        lines = file.readlines()

    line = 0
    batch_size = 10
    received_lines = [None] * len(lines)
    print(f"Sending file lines in batches of {batch_size} as messages.")
    while line < len(lines):
        messages = [
            pack_message(__file__, lines[index], index)
            for index in range(line, min(line + batch_size, len(lines)))
        ]
```

```
        line = line + batch_size
        send_messages(queue, messages)
        print(".", end="")
        sys.stdout.flush()
    print(f"Done. Sent {len(lines) - 1} messages.")

    print(f"Receiving, handling, and deleting messages in batches of {batch_size}.")
    more_messages = True
    while more_messages:
        received_messages = receive_messages(queue, batch_size, 2)
        print(".", end="")
        sys.stdout.flush()
        for message in received_messages:
            path, body, line = unpack_message(message)
            received_lines[line] = body
        if received_messages:
            delete_messages(queue, received_messages)
        else:
            more_messages = False
    print("Done.")

    if all([lines[index] == received_lines[index] for index in range(len(lines))]):
        print(f"Successfully reassembled all file lines!")
    else:
        print(f"Uh oh, some lines were missed!")

    queue.delete()

    print("Thanks for watching!")
    print("-" * 88)
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage

- SendMessage
- SendMessageBatch

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

def lambda_handler(event, context):
    if event:
        batch_item_failures = []
        sqs_batch_response = {}

        for record in event["Records"]:
            try:
                print(f"Processed message: {record['body']}")
            except Exception as e:
                batch_item_failures.append({"itemIdentifier": record['messageId']})

        sqs_batch_response["batchItemFailures"] = batch_item_failures
        return sqs_batch_response
```

Exemplos do Step Functions usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with Step Functions.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Step Functions

O exemplo de código a seguir mostra como começar a usar o Step Functions.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3


def hello_stepfunctions(stepfunctions_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Step Functions client and
 list
    the state machines in your account. This list might be empty if you haven't
 created
    any state machines.
```

```
    This example uses the default settings specified in your shared credentials
    and config files.

    :param stepfunctions_client: A Boto3 Step Functions Client object.
    """
    print("Hello, Step Functions! Let's list up to 10 of your state machines:")
    state_machines = stepfunctions_client.list_state_machines(maxResults=10)
    for sm in state_machines["stateMachines"]:
        print(f"\t{sm['name']}: {sm['stateMachineArn']}")


if __name__ == "__main__":
    hello_stepfunctions(boto3.client("stepfunctions"))
```

- Para obter detalhes da API, consulte a ListStateMachines Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma atividade.
- Criar uma máquina de estado a partir de uma definição da Amazon States Language que contenha a atividade criada anteriormente como uma etapa.
- Executar a máquina de estado e responder à atividade com entrada do usuário.
- Obter o status e a saída finais após a conclusão da execução e, em seguida, limpar os recursos.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class StateMachineScenario:
    """Runs an interactive scenario that shows how to get started using Step
 Functions."""

    def __init__(self, activity, state_machine, iam_client):
        """
        :param activity: An object that wraps activity actions.
        :param state_machine: An object that wraps state machine actions.
        :param iam_client: A Boto3 AWS Identity and Access Management (IAM) client.
        """
        self.activity = activity
        self.state_machine = state_machine
        self.iam_client = iam_client
        self.state_machine_role = None

    def prerequisites(self, state_machine_role_name):
        """
        Finds or creates an IAM role that can be assumed by Step Functions.
        A role of this kind is required to create a state machine.
        The state machine used in this example does not call any additional
 services,
        so it needs no additional permissions.

        :param state_machine_role_name: The name of the role.
        :return: Data about the role.
        """
        trust_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "",
                    "Effect": "Allow",
                    "Principal": {"Service": "states.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        try:
            role = self.iam_client.get_role(RoleName=state_machine_role_name)
            print(f"Prerequisite IAM role {state_machine_role_name} already
 exists.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
```

```
                role = None
            else:
                logger.error(
                    "Couldn't get prerequisite IAM role %s. Here's why: %s: %s",
                    state_machine_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        if role is None:
            try:
                role = self.iam_client.create_role(
                    RoleName=state_machine_role_name,
                    AssumeRolePolicyDocument=json.dumps(trust_policy),
                )
            except ClientError as err:
                logger.error(
                    "Couldn't create prerequisite IAM role %s. Here's why: %s: %s",
                    state_machine_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        self.state_machine_role = role["Role"]

    def find_or_create_activity(self, activity_name):
        """
        Finds or creates a Step Functions activity.

        :param activity_name: The name of the activity.
        :return: The Amazon Resource Name (ARN) of the activity.
        """
        print("First, let's set up an activity and state machine.")
        activity_arn = self.activity.find(activity_name)
        if activity_arn is None:
            activity_arn = self.activity.create(activity_name)
            print(
                f"Activity {activity_name} created. Its Amazon Resource Name (ARN)
is "
                f"{activity_arn}."
            )
        else:
            print(f"Activity {activity_name} already exists.")
        return activity_arn
```

```
    def find_or_create_state_machine(
        self, state_machine_name, activity_arn, state_machine_file
    ):
        """
        Finds or creates a Step Functions state machine.

        :param state_machine_name: The name of the state machine.
        :param activity_arn: The ARN of an activity that is used as a step in the
state
                             machine. This ARN is injected into the state machine
                             definition that's used to create the state machine.
        :param state_machine_file: The path to a file containing the state machine
                                   definition.
        :return: The ARN of the state machine.
        """
        state_machine_arn = self.state_machine.find(state_machine_name)
        if state_machine_arn is None:
            with open(state_machine_file) as state_machine_file:
                state_machine_def = state_machine_file.read().replace(
                    "{{DOC_EXAMPLE_ACTIVITY_ARN}}", activity_arn
                )
                state_machine_arn = self.state_machine.create(
                    state_machine_name,
                    state_machine_def,
                    self.state_machine_role["Arn"],
                )
            print(f"State machine {state_machine_name} created.")
        else:
            print(f"State machine {state_machine_name} already exists.")
        print("-" * 88)
        print(f"Here's some information about state machine {state_machine_name}:")
        state_machine_info = self.state_machine.describe(state_machine_arn)
        for field in ["name", "status", "stateMachineArn", "roleArn"]:
            print(f"\t{field}: {state_machine_info[field]}")
        return state_machine_arn

    def run_state_machine(self, state_machine_arn, activity_arn):
        """
        Run the state machine. The state machine used in this example is a simple
        chat simulation. It contains an activity step in a loop that is used for
user
        interaction. When the state machine gets to the activity step, it waits for
```

```
        an external application to get task data and submit a response. This
function
        acts as the activity application by getting task input and responding with
        user input.

        :param state_machine_arn: The ARN of the state machine.
        :param activity_arn: The ARN of the activity used as a step in the state
machine.
        :return: The ARN of the run.
        """
        print(
            f"Let's run the state machine. It's a simplistic, non-AI chat simulator
"
            f"we'll call ChatSFN."
        )
        user_name = q.ask("What should ChatSFN call you? ", q.non_empty)
        run_input = {"name": user_name}
        print("Starting state machine...")
        run_arn = self.state_machine.start(state_machine_arn, json.dumps(run_input))
        action = None
        while action != "done":
            activity_task = self.activity.get_task(activity_arn)
            task_input = json.loads(activity_task["input"])
            print(f"ChatSFN: {task_input['message']}")
            action = task_input["actions"][
                q.choose("What now? ", task_input["actions"])
            ]
            task_response = {"action": action}
            self.activity.send_task_success(
                activity_task["taskToken"], json.dumps(task_response)
            )
        return run_arn

    def finish_state_machine_run(self, run_arn):
        """
        Wait for the state machine run to finish, then print final status and
output.

        :param run_arn: The ARN of the run to retrieve.
        """
        print(f"Let's get the final output from the state machine:")
        status = "RUNNING"
        while status == "RUNNING":
            run_output = self.state_machine.describe_run(run_arn)
```

```
            status = run_output["status"]
            if status == "RUNNING":
                print(
                    "The state machine is still running, let's wait for it to
finish."
                )
                wait(1)
            elif status == "SUCCEEDED":
                print(f"ChatSFN: {json.loads(run_output['output'])['message']}")
            else:
                print(f"Run status: {status}.")

    def cleanup(
        self,
        state_machine_name,
        state_machine_arn,
        activity_name,
        activity_arn,
        state_machine_role_name,
    ):
        """
        Clean up resources created by this example.

        :param state_machine_name: The name of the state machine.
        :param state_machine_arn: The ARN of the state machine.
        :param activity_name: The name of the activity.
        :param activity_arn: The ARN of the activity.
        :param state_machine_role_name: The name of the role used by the state
machine.
        """
        if q.ask(
            "Do you want to delete the state machine, activity, and role created for
this "
            "example? (y/n) ",
            q.is_yesno,
        ):
            self.state_machine.delete(state_machine_arn)
            print(f"Deleted state machine {state_machine_name}.")
            self.activity.delete(activity_arn)
            print(f"Deleted activity {activity_name}.")
            self.iam_client.delete_role(RoleName=state_machine_role_name)
            print(f"Deleted role {state_machine_role_name}.")

    def run_scenario(self, activity_name, state_machine_name):
```

```
        print("-" * 88)
        print("Welcome to the AWS Step Functions state machines demo.")
        print("-" * 88)

        activity_arn = self.find_or_create_activity(activity_name)
        state_machine_arn = self.find_or_create_state_machine(
            state_machine_name,
            activity_arn,
            "../../../resources/sample_files/chat_sfn_state_machine.json",
        )
        print("-" * 88)
        run_arn = self.run_state_machine(state_machine_arn, activity_arn)
        print("-" * 88)
        self.finish_state_machine_run(run_arn)
        print("-" * 88)
        self.cleanup(
            state_machine_name,
            state_machine_arn,
            activity_name,
            activity_arn,
            self.state_machine_role["RoleName"],
        )

        print("-" * 88)
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        stepfunctions_client = boto3.client("stepfunctions")
        iam_client = boto3.client("iam")
        scenario = StateMachineScenario(
            Activity(stepfunctions_client),
            StateMachine(stepfunctions_client),
            iam_client,
        )
        scenario.prerequisites("doc-example-state-machine-chat")
        scenario.run_scenario("doc-example-activity", "doc-example-state-machine")
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que envolva ações de máquina de estado.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name, definition, role_arn):
        """
        Creates a state machine with the specific definition. The state machine
 assumes
        the provided role before it starts a run.

        :param name: The name to give the state machine.
        :param definition: The Amazon States Language definition of the steps in the
                           the state machine.
        :param role_arn: The Amazon Resource Name (ARN) of the role that is assumed
 by
                         Step Functions when the state machine is run.
        :return: The ARN of the newly created state machine.
        """
        try:
            response = self.stepfunctions_client.create_state_machine(
                name=name, definition=definition, roleArn=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't create state machine %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["stateMachineArn"]


    def find(self, name):
        """
```

```
        Find a state machine by name. This requires listing the state machines until
        one is found with a matching name.

        :param name: The name of the state machine to search for.
        :return: The ARN of the state machine if found; otherwise, None.
        """
        try:
            paginator =
self.stepfunctions_client.get_paginator("list_state_machines")
            for page in paginator.paginate():
                for state_machine in page.get("stateMachines", []):
                    if state_machine["name"] == name:
                        return state_machine["stateMachineArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list state machines. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self, state_machine_arn):
        """
        Get data about a state machine.

        :param state_machine_arn: The ARN of the state machine to look up.
        :return: The retrieved state machine data.
        """
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

```
    def start(self, state_machine_arn, run_input):
        """
        Start a run of a state machine with a specified input. A run is also known
        as an "execution" in Step Functions.

        :param state_machine_arn: The ARN of the state machine to run.
        :param run_input: The input to the state machine, in JSON format.
        :return: The ARN of the run. This can be used to get information about the
run,
                 including its current status and final output.
        """
        try:
            response = self.stepfunctions_client.start_execution(
                stateMachineArn=state_machine_arn, input=run_input
            )
        except ClientError as err:
            logger.error(
                "Couldn't start state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["executionArn"]


    def describe_run(self, run_arn):
        """
        Get data about a state machine run, such as its current status or final
output.

        :param run_arn: The ARN of the run to look up.
        :return: The retrieved run data.
        """
        try:
            response = self.stepfunctions_client.describe_execution(
                executionArn=run_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe run %s. Here's why: %s: %s",
                run_arn,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete(self, state_machine_arn):
        """
        Delete a state machine and all of its run data.

        :param state_machine_arn: The ARN of the state machine to delete.
        """
        try:
            response = self.stepfunctions_client.delete_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

Defina uma classe que envolva ações de atividade.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
```

```
    def create(self, name):
        """
        Create an activity.

        :param name: The name of the activity to create.
        :return: The Amazon Resource Name (ARN) of the newly created activity.
        """
        try:
            response = self.stepfunctions_client.create_activity(name=name)
        except ClientError as err:
            logger.error(
                "Couldn't create activity %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["activityArn"]


    def find(self, name):
        """
        Find an activity by name. This requires listing activities until one is
found
        with a matching name.

        :param name: The name of the activity to search for.
        :return: If found, the ARN of the activity; otherwise, None.
        """
        try:
            paginator = self.stepfunctions_client.get_paginator("list_activities")
            for page in paginator.paginate():
                for activity in page.get("activities", []):
                    if activity["name"] == name:
                        return activity["activityArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list activities. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise


    def get_task(self, activity_arn):
        """
        Gets task data for an activity. When a state machine is waiting for the
        specified activity, a response is returned with data from the state machine.
        When a state machine is not waiting, this call blocks for 60 seconds.

        :param activity_arn: The ARN of the activity to get task data for.
        :return: The task data for the activity.
        """
        try:
            response = self.stepfunctions_client.get_activity_task(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't get a task for activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def send_task_success(self, task_token, task_response):
        """
        Sends a success response to a waiting activity step. A state machine with an
        activity step waits for the activity to get task data and then respond with
        either success or failure before it resumes processing.

        :param task_token: The token associated with the task. This is included in
the
                           response to the get_activity_task action and must be sent
                           without modification.
        :param task_response: The response data from the activity. This data is
                              received and processed by the state machine.
        """
        try:
            self.stepfunctions_client.send_task_success(
                taskToken=task_token, output=task_response
```

```
            )
        except ClientError as err:
            logger.error(
                "Couldn't send task success. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self, activity_arn):
        """
        Delete an activity.

        :param activity_arn: The ARN of the activity to delete.
        """
        try:
            response = self.stepfunctions_client.delete_activity(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine

- DescribeExecution
- DescribeStateMachine
- GetActivityTask
- ListActivities
- ListStateMachines
- SendTaskSuccess
- StartExecution
- StopExecution

Ações

`CreateActivity`

O código de exemplo a seguir mostra como usar `CreateActivity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name):
        """
        Create an activity.

        :param name: The name of the activity to create.
```

```
        :return: The Amazon Resource Name (ARN) of the newly created activity.
        """
        try:
            response = self.stepfunctions_client.create_activity(name=name)
        except ClientError as err:
            logger.error(
                "Couldn't create activity %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["activityArn"]
```

- Para obter detalhes da API, consulte a CreateActivity Referência da API AWS SDK for Python (Boto3).

`CreateStateMachine`

O código de exemplo a seguir mostra como usar `CreateStateMachine`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
```

```
    def create(self, name, definition, role_arn):
        """
        Creates a state machine with the specific definition. The state machine
assumes
        the provided role before it starts a run.

        :param name: The name to give the state machine.
        :param definition: The Amazon States Language definition of the steps in the
                           the state machine.
        :param role_arn: The Amazon Resource Name (ARN) of the role that is assumed
by
                         Step Functions when the state machine is run.
        :return: The ARN of the newly created state machine.
        """
        try:
            response = self.stepfunctions_client.create_state_machine(
                name=name, definition=definition, roleArn=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't create state machine %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["stateMachineArn"]
```

- Para obter detalhes da API, consulte a CreateStateMachine Referência da API AWS SDK for Python (Boto3).

`DeleteActivity`

O código de exemplo a seguir mostra como usar `DeleteActivity`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def delete(self, activity_arn):
        """
        Delete an activity.

        :param activity_arn: The ARN of the activity to delete.
        """
        try:
            response = self.stepfunctions_client.delete_activity(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteActivity Referência da API AWS SDK for Python (Boto3).

`DeleteStateMachine`

O código de exemplo a seguir mostra como usar `DeleteStateMachine`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def delete(self, state_machine_arn):
        """
        Delete a state machine and all of its run data.

        :param state_machine_arn: The ARN of the state machine to delete.
        """
        try:
            response = self.stepfunctions_client.delete_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DeleteStateMachine Referência da API AWS SDK for Python (Boto3).

`DescribeExecution`

O código de exemplo a seguir mostra como usar `DescribeExecution`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    def describe_run(self, run_arn):
        """
        Get data about a state machine run, such as its current status or final
output.

        :param run_arn: The ARN of the run to look up.
        :return: The retrieved run data.
        """
        try:
            response = self.stepfunctions_client.describe_execution(
                executionArn=run_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe run %s. Here's why: %s: %s",
                run_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return response
```

- Para obter detalhes da API, consulte a DescribeExecution Referência da API AWS SDK for Python (Boto3).

`DescribeStateMachine`

O código de exemplo a seguir mostra como usar `DescribeStateMachine`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def describe(self, state_machine_arn):
        """
        Get data about a state machine.

        :param state_machine_arn: The ARN of the state machine to look up.
        :return: The retrieved state machine data.
        """
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=state_machine_arn
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't describe state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DescribeStateMachine Referência da API AWS SDK for Python (Boto3).

`GetActivityTask`

O código de exemplo a seguir mostra como usar `GetActivityTask`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def get_task(self, activity_arn):
        """
        Gets task data for an activity. When a state machine is waiting for the
```

```
        specified activity, a response is returned with data from the state machine.
        When a state machine is not waiting, this call blocks for 60 seconds.

        :param activity_arn: The ARN of the activity to get task data for.
        :return: The task data for the activity.
        """
        try:
            response = self.stepfunctions_client.get_activity_task(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't get a task for activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a GetActivityTask Referência da API AWS SDK for Python (Boto3).

`ListActivities`

O código de exemplo a seguir mostra como usar `ListActivities`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""
```

```
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def find(self, name):
        """
        Find an activity by name. This requires listing activities until one is
found
        with a matching name.

        :param name: The name of the activity to search for.
        :return: If found, the ARN of the activity; otherwise, None.
        """
        try:
            paginator = self.stepfunctions_client.get_paginator("list_activities")
            for page in paginator.paginate():
                for activity in page.get("activities", []):
                    if activity["name"] == name:
                        return activity["activityArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list activities. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListActivities Referência da API AWS SDK for Python (Boto3).

`ListStateMachines`

O código de exemplo a seguir mostra como usar `ListStateMachines`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Encontre uma máquina de estado pelo nome pesquisando a lista de máquinas de estado da conta.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def find(self, name):
        """
        Find a state machine by name. This requires listing the state machines until
        one is found with a matching name.

        :param name: The name of the state machine to search for.
        :return: The ARN of the state machine if found; otherwise, None.
        """
        try:
            paginator =
 self.stepfunctions_client.get_paginator("list_state_machines")
            for page in paginator.paginate():
                for state_machine in page.get("stateMachines", []):
                    if state_machine["name"] == name:
                        return state_machine["stateMachineArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list state machines. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Para obter detalhes da API, consulte a ListStateMachines Referência da API AWS SDK for Python (Boto3).

`SendTaskSuccess`

O código de exemplo a seguir mostra como usar `SendTaskSuccess`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def send_task_success(self, task_token, task_response):
        """
        Sends a success response to a waiting activity step. A state machine with an
        activity step waits for the activity to get task data and then respond with
        either success or failure before it resumes processing.

        :param task_token: The token associated with the task. This is included in
 the
                           response to the get_activity_task action and must be sent
                           without modification.
        :param task_response: The response data from the activity. This data is
                              received and processed by the state machine.
```

```
        """
        try:
            self.stepfunctions_client.send_task_success(
                taskToken=task_token, output=task_response
            )
        except ClientError as err:
            logger.error(
                "Couldn't send task success. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a SendTaskSuccess Referência da API AWS SDK for Python (Boto3).

`StartExecution`

O código de exemplo a seguir mostra como usar `StartExecution`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def start(self, state_machine_arn, run_input):
```

```
        """
        Start a run of a state machine with a specified input. A run is also known
        as an "execution" in Step Functions.

        :param state_machine_arn: The ARN of the state machine to run.
        :param run_input: The input to the state machine, in JSON format.
        :return: The ARN of the run. This can be used to get information about the
run,
                 including its current status and final output.
        """
        try:
            response = self.stepfunctions_client.start_execution(
                stateMachineArn=state_machine_arn, input=run_input
            )
        except ClientError as err:
            logger.error(
                "Couldn't start state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["executionArn"]
```

- Para obter detalhes da API, consulte a StartExecution Referência da API AWS SDK for Python (Boto3).

Cenários

Criar uma aplicação de mensageiro

O exemplo de código a seguir mostra como criar um aplicativo de AWS Step Functions mensagens que recupera registros de mensagens de uma tabela de banco de dados.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) with AWS Step Functions para criar um aplicativo de mensagens que recupera registros de mensagens de uma tabela do Amazon DynamoDB e os envia com o Amazon Simple Queue Service (Amazon SQS). A máquina de

estado se integra a uma AWS Lambda função para verificar o banco de dados em busca de mensagens não enviadas.

- Crie uma máquina de estado que recupere e atualize registros de mensagens de uma tabela do Amazon DynamoDB.
- Atualize a definição de máquina de estado para enviar mensagens ao Amazon Simple Queue Service (Amazon SQS).
- Inicie e interrompa execuções da máquina de estado.
- Conecte-se ao Lambda, ao DynamoDB e ao Amazon SQS por meio de uma máquina de estado usando integrações de serviço.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Lambda
- Amazon SQS
- Funções de etapa

Orquestrar aplicações de IA generativa com o Step Functions

O exemplo de código a seguir mostra como criar e orquestrar aplicações de IA generativa com o Amazon Bedrock e o Step Functions.

SDK para Python (Boto3)

O cenário de encadeamento de prompts do Amazon Bedrock Sem Servidor demonstra como o AWS Step Functions, o Amazon Bedrock e a documentação https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html podem ser usados para criar e orquestrar aplicações de IA generativa complexas, sem servidor e altamente escaláveis. Ele contém os seguintes exemplos de trabalho:

- Escrever uma análise de um determinado romance para um blog de literatura. Este exemplo ilustra uma cadeia de prompts simples e sequencial.
- Gerar uma história curta sobre um determinado tópico. Este exemplo ilustra como a IA pode processar uma lista de itens gerada anteriormente de forma iterativa.

- Criar um itinerário para férias de fim de semana em um determinado destino. Este exemplo ilustra como paralelizar vários prompts distintos.
- Lançar ideias de filmes para um usuário humano que atua como produtor de filmes. Este exemplo ilustra como paralelizar o mesmo prompt com diferentes parâmetros de inferência, como voltar a uma etapa anterior na cadeia e como incluir a entrada humana como parte do fluxo de trabalho.
- Planejar uma refeição com base nos ingredientes que o usuário tem em mãos. Este exemplo ilustra como as cadeias de prompts podem incorporar duas conversas distintas de IA, com duas personas de IA participando de um debate entre si para melhorar o resultado final.
- Encontre e resuma o repositório mais popular GitHub da atualidade. Este exemplo ilustra o encadeamento de vários agentes de IA que interagem com APIs externas.

Para obter o código-fonte completo e as instruções de configuração e execução, consulte o projeto completo em GitHub.

Serviços usados neste exemplo

- Amazon Bedrock
- Amazon Bedrock Runtime
- Amazon Bedrock Agents
- Amazon Bedrock Agents Runtime
- Funções de etapa

AWS STS exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Assuma um perfil do IAM que exija um token de MFA e use credenciais temporárias para listar os buckets do Amazon S3 para a conta.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
```

```
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- Para obter detalhes da API, consulte a AssumeRole Referência da API AWS SDK for Python (Boto3).

`GetSessionToken`

O código de exemplo a seguir mostra como usar `GetSessionToken`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um token de sessão passando um token de MFA e use-o para listar os buckets do Amazon S3 para a conta.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
```

```
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- Para obter detalhes da API, consulte a GetSessionToken Referência da API AWS SDK for Python (Boto3).

Cenários

Assumir um perfil do IAM que exija um token de MFA

O exemplo de código a seguir mostra como assumir um perfil que exige um token de MFA.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Criar um perfil do IAM que conceda permissão para listar os buckets do Amazon S3.
- Criar um usuário do IAM que tenha permissão para assumir o perfil somente quando as credenciais de MFA forem fornecidas.
- Registrar um dispositivo MFA para o usuário.
- Assumir o perfil e usar credenciais temporárias para listar os buckets do S3.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário do IAM, registre um dispositivo de MFA e crie um perfil que conceda permissão para listar os buckets do Amazon S3. O usuário só tem direitos para assumir a função.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
 MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the role,
```

```
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, roles, and policies
                         in the account.
    :return: The newly created user, user key, virtual MFA device, and role.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)
```

```
    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": user.arn},
                        "Action": "sts:AssumeRole",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent": True}},
                    }
                ],
            }
        ),
    )
    print(f"Created role {role.name} that requires MFA.")

    policy = iam_resource.create_policy(
        PolicyName=unique_name("policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
```

```
                        "Action": "sts:AssumeRole",
                        "Resource": role.arn,
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user assume "
        f"the role."
    )

    print("Give AWS time to propagate these new resources and connections.", end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device, role
```

Mostre que não é permitido assumir uma função sem um token de MFA.

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
 RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Got AccessDenied.")
        else:
            raise
```

Assuma o perfil que concede permissão para listar os buckets do S3 passando o token de MFA necessário e mostre que os buckets podem ser listados.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Destrua os recursos criados para a demonstração.

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Execute esse cenário usando a funções definidas anteriormente.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
```

```
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
        )
        try_to_assume_role_without_mfa(role.arn, "demo-sts-session", sts_client)
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_from_assumed_role_with_mfa(
            role.arn,
            "demo-sts-session",
            virtual_mfa_device.serial_number,
            mfa_totp,
            sts_client,
        )
    finally:
        teardown(user, virtual_mfa_device, role)
        print("Thanks for watching!")
```

- Para obter detalhes da API, consulte a AssumeRole Referência da API AWS SDK for Python (Boto3).

Criar um URL para usuários federados usando

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM que conceda acesso somente leitura aos recursos do Amazon S3 da conta atual.
- Obtenha um token de segurança do endpoint da AWS federação.
- Criar um URL que possa ser usado para acessar o console com credenciais federadas.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um perfil que conceda acesso somente leitura aos recursos do S3 da conta atual.

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) instance
                         that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": iam_resource.CurrentUser().arn},
                        "Action": "sts:AssumeRole",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn="arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess")
    print(f"Created role {role.name}.")

    print("Give AWS time to propagate these new resources and connections.", end="")
    progress_bar(10)

    return role
```

Obtenha um token de segurança do endpoint de AWS federação e construa uma URL que possa ser usada para acessar o console com credenciais federadas.

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS) that
       can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
       AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS federation
       endpoint, includes the sign-in token for authentication, and redirects to
       the AWS Management Console with permissions defined by the role that was
       specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are
 granted.
                            The current user must have permission to assume the
 role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
        "sessionId": temp_credentials["AccessKeyId"],
        "sessionKey": temp_credentials["SecretAccessKey"],
        "sessionToken": temp_credentials["SessionToken"],
    }
    aws_federated_signin_endpoint = "https://signin.aws.amazon.com/federation"

    # Make a request to the AWS federation endpoint to get a sign-in token.
```

```
    # The requests.get function URL-encodes the parameters and builds the query
 string
    # before making the request.
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            "Action": "getSigninToken",
            "SessionDuration": str(datetime.timedelta(hours=12).seconds),
            "Session": json.dumps(session_data),
        },
    )
    signin_token = json.loads(response.text)
    print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

    # Make a federated URL that can be used to sign into the AWS Management Console.
    query_string = urllib.parse.urlencode(
        {
            "Action": "login",
            "Issuer": issuer,
            "Destination": "https://console.aws.amazon.com/",
            "SigninToken": signin_token["SigninToken"],
        }
    )
    federated_url = f"{aws_federated_signin_endpoint}?{query_string}"
    return federated_url
```

Destrua os recursos criados para a demonstração.

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

Execute esse cenário usando a funções definidas anteriormente.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    role = setup(iam_resource)
    sts_client = boto3.client("sts")
    try:
        federated_url = construct_federated_url(
            role.arn, "AssumeRoleDemoSession", "example.org", sts_client
        )
        print(
            "Constructed a federated URL that can be used to connect to the "
            "AWS Management Console with role-defined permissions:"
        )
        print("-" * 88)
        print(federated_url)
        print("-" * 88)
        _ = input(
            "Copy and paste the above URL into a browser to open the AWS "
            "Management Console with limited permissions. When done, press "
            "Enter to clean up and complete this demo."
        )
    finally:
        teardown(role)
        print("Thanks for watching!")
```

- Para obter detalhes da API, consulte a AssumeRole Referência da API AWS SDK for Python (Boto3).

Obtenha um token de sessão que requeira um token de MFA

O exemplo de código a seguir mostra como obter um token de sessão que exige um token de MFA.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Criar um perfil do IAM que conceda permissão para listar os buckets do Amazon S3.
- Criar um usuário do IAM que tenha permissão para assumir o perfil somente quando as credenciais de MFA forem fornecidas.
- Registrar um dispositivo MFA para o usuário.
- Forneça credenciais de MFA para obter um token de sessão e use credenciais temporárias para listar os buckets do S3.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário do IAM, registre um dispositivo de MFA e crie um perfil que conceda permissão para deixar o usuário listar os buckets do S3 somente quando credenciais de MFA forem usadas.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3 buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.
```

```
    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, MFA devices, and
                         policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
```

```
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent": True}},
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user list buckets,
 "
        f"but only when MFA credentials are present."
    )

    print("Give AWS time to propagate these new resources and connections.", end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device
```

Obtenha credenciais de sessão temporárias passando um token de MFA e use-as para listar os buckets do S3 para a conta.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
```

```
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Destrua os recursos criados para a demonstração.

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Execute esse cenário usando a funções definidas anteriormente.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
        )
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response["Error"]["Code"] == "AccessDenied":
                print("Got expected AccessDenied error.")
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_with_session_token_with_mfa(
            virtual_mfa_device.serial_number, mfa_totp, sts_client
        )
    finally:
        teardown(user, virtual_mfa_device)
        print("Thanks for watching!")
```

- Para obter detalhes da API, consulte a GetSessionToken Referência da API AWS SDK for Python (Boto3).

Suporte exemplos de uso do SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) with Suporte.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Suporte

O exemplo de código a seguir mostra como começar a usar o Suporte.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def hello_support(support_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Support client and count
```

```
    the available services in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param support_client: A Boto3 Support Client object.
    """
    try:
        print("Hello, AWS Support! Let's count the available Support services:")
        response = support_client.describe_services()
        print(f"There are {len(response['services'])} services available.")
    except ClientError as err:
        if err.response["Error"]["Code"] == "SubscriptionRequiredException":
            logger.info(
                "You must have a Business, Enterprise On-Ramp, or Enterprise Support
 "
                "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                "examples."
            )
        else:
            logger.error(
                "Couldn't count services. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


if __name__ == "__main__":
    hello_support(boto3.client("support"))
```

- Para obter detalhes da API, consulte a DescribeServices Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Obter e exibir os serviços disponíveis e os níveis de gravidade dos casos.

- Criar um caso de suporte usando um serviço, uma categoria e um nível de gravidade selecionados.
- Obter e exibir uma lista de casos em aberto para o dia atual.
- Adicionar um conjunto de anexos e uma comunicação ao novo caso.
- Descrever o novo anexo e a comunicação para o caso.
- Resolver o caso.
- Obtenha e exiba uma lista de casos resolvidos para o dia atual.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class SupportCasesScenario:
    """Runs an interactive scenario that shows how to get started using AWS
 Support."""

    def __init__(self, support_wrapper):
        """
        :param support_wrapper: An object that wraps AWS Support actions.
        """
        self.support_wrapper = support_wrapper

    def display_and_select_service(self):
        """
        Lists support services and prompts the user to select one.

        :return: The support service selected by the user.
        """
        print("-" * 88)
        services_list = self.support_wrapper.describe_services("en")
        print(f"AWS Support client returned {len(services_list)} services.")
        print("Displaying first 10 services:")
```

```
        service_choices = [svc["name"] for svc in services_list[:10]]
        selected_index = q.choose(
            "Select an example support service by entering a number from the
preceding list:",
            service_choices,
        )
        selected_service = services_list[selected_index]
        print("-" * 88)
        return selected_service

    def display_and_select_category(self, service):
        """
        Lists categories for a support service and prompts the user to select one.

        :param service: The service of the categories.
        :return: The selected category.
        """
        print("-" * 88)
        print(
            f"Available support categories for Service {service['name']}
{len(service['categories'])}:"
        )
        categories_choices = [category["name"] for category in
service["categories"]]
        selected_index = q.choose(
            "Select an example support category by entering a number from the
preceding list:",
            categories_choices,
        )
        selected_category = service["categories"][selected_index]
        print("-" * 88)
        return selected_category

    def display_and_select_severity(self):
        """
        Lists available severity levels and prompts the user to select one.

        :return: The selected severity level.
        """
        print("-" * 88)
        severity_levels_list = self.support_wrapper.describe_severity_levels("en")
        print(f"Available severity levels:")
        severity_choices = [level["name"] for level in severity_levels_list]
        selected_index = q.choose(
```

```
            "Select an example severity level by entering a number from the
preceding list:",
            severity_choices,
        )
        selected_severity = severity_levels_list[selected_index]
        print("-" * 88)
        return selected_severity

    def create_example_case(self, service, category, severity_level):
        """
        Creates an example support case with the user's selections.

        :param service: The service for the new case.
        :param category: The category for the new case.
        :param severity_level: The severity level for the new case.
        :return: The caseId of the new support case.
        """
        print("-" * 88)
        print(f"Creating new case for service {service['name']}.")
        case_id = self.support_wrapper.create_case(service, category,
severity_level)
        print(f"\tNew case created with ID {case_id}.")
        print("-" * 88)
        return case_id

    def list_open_cases(self):
        """
        List the open cases for the current day.
        """
        print("-" * 88)
        print("Let's list the open cases for the current day.")
        start_time = str(datetime.utcnow().date())
        end_time = str(datetime.utcnow().date() + timedelta(days=1))
        open_cases = self.support_wrapper.describe_cases(start_time, end_time,
False)
        for case in open_cases:
            print(f"\tCase: {case['caseId']}: status {case['status']}.")
        print("-" * 88)

    def create_attachment_set(self):
        """
        Create an attachment set with a sample file.

        :return: The attachment set ID of the new attachment set.
```

```
        """
        print("-" * 88)
        print("Creating attachment set with a sample file.")
        attachment_set_id = self.support_wrapper.add_attachment_to_set()
        print(f"\tNew attachment set created with ID {attachment_set_id}.")
        print("-" * 88)
        return attachment_set_id

    def add_communication(self, case_id, attachment_set_id):
        """
        Add a communication with an attachment set to the case.

        :param case_id: The ID of the case for the communication.
        :param attachment_set_id: The ID of the attachment set to
        add to the communication.
        """
        print("-" * 88)
        print(f"Adding a communication and attachment set to the case.")
        self.support_wrapper.add_communication_to_case(attachment_set_id, case_id)
        print(
            f"Added a communication and attachment set {attachment_set_id} to the
case {case_id}."
        )
        print("-" * 88)

    def list_communications(self, case_id):
        """
        List the communications associated with a case.

        :param case_id: The ID of the case.
        :return: The attachment ID of an attachment.
        """
        print("-" * 88)
        print("Let's list the communications for our case.")
        attachment_id = ""
        communications =
self.support_wrapper.describe_all_case_communications(case_id)
        for communication in communications:
            print(
                f"\tCommunication created on {communication['timeCreated']} "
                f"has {len(communication['attachmentSet'])} attachments."
            )
            if len(communication["attachmentSet"]) > 0:
                attachment_id = communication["attachmentSet"][0]["attachmentId"]
```

```
        print("-" * 88)
        return attachment_id

    def describe_case_attachment(self, attachment_id):
        """
        Describe an attachment associated with a case.

        :param attachment_id: The ID of the attachment.
        """
        print("-" * 88)
        print("Let's list the communications for our case.")
        attached_file = self.support_wrapper.describe_attachment(attachment_id)
        print(f"\tAttachment includes file {attached_file}.")
        print("-" * 88)

    def resolve_case(self, case_id):
        """
        Shows how to resolve an AWS Support case by its ID.

        :param case_id: The ID of the case to resolve.
        """
        print("-" * 88)
        print(f"Resolving case with ID {case_id}.")
        case_status = self.support_wrapper.resolve_case(case_id)
        print(f"\tFinal case status is {case_status}.")
        print("-" * 88)

    def list_resolved_cases(self):
        """
        List the resolved cases for the current day.
        """
        print("-" * 88)
        print("Let's list the resolved cases for the current day.")
        start_time = str(datetime.utcnow().date())
        end_time = str(datetime.utcnow().date() + timedelta(days=1))
        resolved_cases = self.support_wrapper.describe_cases(start_time, end_time,
True)
        for case in resolved_cases:
            print(f"\tCase: {case['caseId']}: status {case['status']}.")
        print("-" * 88)

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
        print("-" * 88)
        print("Welcome to the AWS Support get started with support cases demo.")
        print("-" * 88)

        selected_service = self.display_and_select_service()
        selected_category = self.display_and_select_category(selected_service)
        selected_severity = self.display_and_select_severity()
        new_case_id = self.create_example_case(
            selected_service, selected_category, selected_severity
        )
        wait(10)
        self.list_open_cases()
        new_attachment_set_id = self.create_attachment_set()
        self.add_communication(new_case_id, new_attachment_set_id)
        new_attachment_id = self.list_communications(new_case_id)
        self.describe_case_attachment(new_attachment_id)
        self.resolve_case(new_case_id)
        wait(10)
        self.list_resolved_cases()

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = SupportCasesScenario(SupportWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que envolva ações de suporte ao cliente.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_services(self, language):
        """
        Get the descriptions of AWS services available for support for a language.

        :param language: The language for support services.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of AWS service descriptions.
        """
        try:
            response = self.support_client.describe_services(language=language)
            services = response["services"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get Support services for language %s. Here's why: %s:
%s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return services


    def describe_severity_levels(self, language):
        """
```

```
        Get the descriptions of available severity levels for support cases for a
language.

        :param language: The language for support severity levels.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of severity levels.
        """
        try:
            response =
self.support_client.describe_severity_levels(language=language)
            severity_levels = response["severityLevels"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get severity levels for language %s. Here's why: %s:
%s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return severity_levels


    def create_case(self, service, category, severity):
        """
        Create a new support case.

        :param service: The service to use for the new case.
        :param category: The category to use for the new case.
        :param severity: The severity to use for the new case.
        :return: The caseId of the new case.
        """
        try:
            response = self.support_client.create_case(
```

```
                subject="Example case for testing, ignore.",
                serviceCode=service["code"],
                severityCode=severity["code"],
                categoryCode=category["code"],
                communicationBody="Example support case body.",
                language="en",
                issueType="customer-service",
            )
            case_id = response["caseId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't create case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return case_id


    def add_attachment_to_set(self):
        """
        Add an attachment to a set, or create a new attachment set if one does not
exist.

        :return: The attachment set ID.
        """
        try:
            response = self.support_client.add_attachments_to_set(
                attachments=[
                    {
                        "fileName": "attachment_file.txt",
                        "data": b"This is a sample file for attachment to a support
case.",
                    }
```

```
                ]
            )
            new_set_id = response["attachmentSetId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add attachment. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return new_set_id


    def add_communication_to_case(self, attachment_set_id, case_id):
        """
        Add a communication and an attachment set to a case.

        :param attachment_set_id: The ID of an existing attachment set.
        :param case_id: The ID of the case.
        """
        try:
            self.support_client.add_communication_to_case(
                caseId=case_id,
                communicationBody="This is an example communication added to a
support case.",
                attachmentSetId=attachment_set_id,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
```

```
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add communication. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise


    def describe_all_case_communications(self, case_id):
        """
        Describe all the communications for a case using a paginator.

        :param case_id: The ID of the case.
        :return: The communications for the case.
        """
        try:
            communications = []
            paginator = self.support_client.get_paginator("describe_communications")
            for page in paginator.paginate(caseId=case_id):
                communications += page["communications"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe communications. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return communications


    def describe_attachment(self, attachment_id):
```

```
        """
        Get information about an attachment by its attachmentID.

        :param attachment_id: The ID of the attachment.
        :return: The name of the attached file.
        """
        try:
            response = self.support_client.describe_attachment(
                attachmentId=attachment_id
            )
            attached_file = response["attachment"]["fileName"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get attachment description. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return attached_file


    def resolve_case(self, case_id):
        """
        Resolve a support case by its caseId.

        :param case_id: The ID of the case to resolve.
        :return: The final status of the case.
        """
        try:
            response = self.support_client.resolve_case(caseId=case_id)
            final_status = response["finalCaseStatus"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
```

```
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't resolve case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return final_status


    def describe_cases(self, after_time, before_time, resolved):
        """
        Describe support cases over a period of time, optionally filtering
        by status.

        :param after_time: The start time to include for cases.
        :param before_time: The end time to include for cases.
        :param resolved: True to include resolved cases in the results,
            otherwise results are open cases.
        :return: The final status of the case.
        """
        try:
            cases = []
            paginator = self.support_client.get_paginator("describe_cases")
            for page in paginator.paginate(
                afterTime=after_time,
                beforeTime=before_time,
                includeResolvedCases=resolved,
                language="en",
            ):
                cases += page["cases"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
```

```
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe cases. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            if resolved:
                cases = filter(lambda case: case["status"] == "resolved", cases)
            return cases
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Ações

`AddAttachmentsToSet`

O código de exemplo a seguir mostra como usar `AddAttachmentsToSet`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def add_attachment_to_set(self):
        """
        Add an attachment to a set, or create a new attachment set if one does not
 exist.

        :return: The attachment set ID.
        """
        try:
            response = self.support_client.add_attachments_to_set(
                attachments=[
                    {
                        "fileName": "attachment_file.txt",
                        "data": b"This is a sample file for attachment to a support
 case.",
                    }
                ]
```

```
            )
            new_set_id = response["attachmentSetId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add attachment. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return new_set_id
```

- Para obter detalhes da API, consulte a AddAttachmentsToSet Referência da API AWS SDK for Python (Boto3).

`AddCommunicationToCase`

O código de exemplo a seguir mostra como usar `AddCommunicationToCase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""
```

```
    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def add_communication_to_case(self, attachment_set_id, case_id):
        """
        Add a communication and an attachment set to a case.

        :param attachment_set_id: The ID of an existing attachment set.
        :param case_id: The ID of the case.
        """
        try:
            self.support_client.add_communication_to_case(
                caseId=case_id,
                communicationBody="This is an example communication added to a
support case.",
                attachmentSetId=attachment_set_id,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add communication. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
                raise
```

- Para obter detalhes da API, consulte a AddCommunicationToCase Referência da API AWS SDK for Python (Boto3).

`CreateCase`

O código de exemplo a seguir mostra como usar `CreateCase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def create_case(self, service, category, severity):
        """
        Create a new support case.
```

```
        :param service: The service to use for the new case.
        :param category: The category to use for the new case.
        :param severity: The severity to use for the new case.
        :return: The caseId of the new case.
        """
        try:
            response = self.support_client.create_case(
                subject="Example case for testing, ignore.",
                serviceCode=service["code"],
                severityCode=severity["code"],
                categoryCode=category["code"],
                communicationBody="Example support case body.",
                language="en",
                issueType="customer-service",
            )
            case_id = response["caseId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't create case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return case_id
```

- Para obter detalhes da API, consulte a CreateCase Referência da API AWS SDK for Python (Boto3).

`DescribeAttachment`

O código de exemplo a seguir mostra como usar `DescribeAttachment`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_attachment(self, attachment_id):
        """
        Get information about an attachment by its attachmentID.

        :param attachment_id: The ID of the attachment.
        :return: The name of the attached file.
        """
        try:
            response = self.support_client.describe_attachment(
                attachmentId=attachment_id
            )
            attached_file = response["attachment"]["fileName"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
```

```
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get attachment description. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return attached_file
```

- Para obter detalhes da API, consulte a DescribeAttachment Referência da API AWS SDK for Python (Boto3).

`DescribeCases`

O código de exemplo a seguir mostra como usar `DescribeCases`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_cases(self, after_time, before_time, resolved):
        """
        Describe support cases over a period of time, optionally filtering
        by status.

        :param after_time: The start time to include for cases.
        :param before_time: The end time to include for cases.
        :param resolved: True to include resolved cases in the results,
            otherwise results are open cases.
        :return: The final status of the case.
        """
        try:
            cases = []
            paginator = self.support_client.get_paginator("describe_cases")
            for page in paginator.paginate(
                afterTime=after_time,
                beforeTime=before_time,
                includeResolvedCases=resolved,
                language="en",
            ):
                cases += page["cases"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe cases. Here's why: %s: %s",
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
                raise
        else:
            if resolved:
                cases = filter(lambda case: case["status"] == "resolved", cases)
            return cases
```

- Para obter detalhes da API, consulte a DescribeCases Referência da API AWS SDK for Python (Boto3).

`DescribeCommunications`

O código de exemplo a seguir mostra como usar `DescribeCommunications`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)
```

```
    def describe_all_case_communications(self, case_id):
        """
        Describe all the communications for a case using a paginator.

        :param case_id: The ID of the case.
        :return: The communications for the case.
        """
        try:
            communications = []
            paginator = self.support_client.get_paginator("describe_communications")
            for page in paginator.paginate(caseId=case_id):
                communications += page["communications"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe communications. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return communications
```

- Para obter detalhes da API, consulte a DescribeCommunications Referência da API AWS SDK for Python (Boto3).

`DescribeServices`

O código de exemplo a seguir mostra como usar `DescribeServices`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_services(self, language):
        """
        Get the descriptions of AWS services available for support for a language.

        :param language: The language for support services.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of AWS service descriptions.
        """
        try:
            response = self.support_client.describe_services(language=language)
            services = response["services"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
```

```
                    "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get Support services for language %s. Here's why: %s:
 %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return services
```

- Para obter detalhes da API, consulte a DescribeServices Referência da API AWS SDK for Python (Boto3).

`DescribeSeverityLevels`

O código de exemplo a seguir mostra como usar `DescribeSeverityLevels`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_severity_levels(self, language):
        """
        Get the descriptions of available severity levels for support cases for a
language.

        :param language: The language for support severity levels.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of severity levels.
        """
        try:
            response =
self.support_client.describe_severity_levels(language=language)
            severity_levels = response["severityLevels"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get severity levels for language %s. Here's why: %s:
%s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return severity_levels
```

- Para obter detalhes da API, consulte a DescribeSeverityLevels Referência da API AWS SDK for Python (Boto3).

`ResolveCase`

O código de exemplo a seguir mostra como usar `ResolveCase`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def resolve_case(self, case_id):
        """
        Resolve a support case by its caseId.

        :param case_id: The ID of the case to resolve.
        :return: The final status of the case.
```

```
        """
        try:
            response = self.support_client.resolve_case(caseId=case_id)
            final_status = response["finalCaseStatus"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't resolve case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return final_status
```

- Para obter detalhes da API, consulte a ResolveCase Referência da API AWS SDK for Python (Boto3).

Exemplos do Systems Manager usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Systems Manager.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello Systems Manager

O exemplo de código a seguir mostra como começar a usar o Systems Manager.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import boto3
from botocore.exceptions import ClientError


def hello_systems_manager(ssm_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Systems Manager
    client and list the first 5 documents in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ssm_client: A Boto3 AWS Systems Manager Client object. This object wraps
                           the low-level AWS Systems Manager service API.
    """
    print("Hello, AWS Systems Manager! Let's list some of your documents:\n")

    paginator = ssm_client.get_paginator("list_documents")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 5})
    for page in page_iterator:
        for document in page["DocumentIdentifiers"]:
            print(f"  {document['Name']}")
```

```
if __name__ == "__main__":
    try:
        hello_systems_manager(boto3.client("ssm"))
    except ClientError as err:
        print("Hello systems manager had an error.")
        print(err.response["Error"]["Code"])
        print(err.response["Error"]["Message"])
```

- Para obter detalhes da API, consulte a ListDocuments Referência da API AWS SDK for Python (Boto3).

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma janela de manutenção.
- Modificar a programação da janela de manutenção.
- Criar um documento.
- Enviar um comando para uma instância do EC2 especificada.
- Crie um OpsItem.
- Atualize e resolva OpsItem o.
- Exclua a janela de manutenção OpsItem e o documento.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo em um prompt de comando.

```
class SystemsManagerScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Systems Manager."""

    def __init__(self, document_wrapper, maintenance_window_wrapper,
 ops_item_wrapper):
        """
        :param document_wrapper: An object that wraps Systems Manager document
 functions.
        :param maintenance_window_wrapper: An object that wraps Systems Manager
 maintenance window functions.
        :param ops_item_wrapper: An object that wraps Systems Manager OpsItem
 functions.
        """
        self.document_wrapper = document_wrapper
        self.maintenance_window_wrapper = maintenance_window_wrapper
        self.ops_item_wrapper = ops_item_wrapper

    def run(self):
        """Demonstrates how to use the AWS SDK for Python (Boto3) to get started
 with Systems Manager."""
        try:
            print("-" * 88)
            print(
                """
Welcome to the AWS Systems Manager SDK Getting Started scenario.
This program demonstrates how to interact with Systems Manager using the AWS SDK for
 Python (Boto3).
Systems Manager is the operations hub for your AWS applications and resources and a
 secure end-to-end management
solution. The program's primary functions include creating a maintenance window,
 creating a document, sending a
command to a document, listing documents, listing commands, creating an OpsItem,
 modifying an OpsItem, and deleting
Systems Manager resources. Upon completion of the program, all AWS resources are
 cleaned up.
Let's get started..."""
            )
            q.ask("Please hit Enter")

            print("-" * 88)
            print("Create a Systems Manager maintenance window.")
            maintenance_window_name = q.ask(
```

```
                "Please enter the maintenance window name (default is ssm-
maintenance-window):",
            )
            if not maintenance_window_name:
                maintenance_window_name = "ssm-maintenance-window"

            self.maintenance_window_wrapper.create(
                name=maintenance_window_name,
                schedule="cron(0 10 ? * MON-FRI *)",
                duration=2,
                cutoff=1,
                allow_unassociated_targets=True,
            )

            print("-" * 88)
            print("Modify the maintenance window by changing the schedule")
            q.ask("Please hit Enter")

            self.maintenance_window_wrapper.update(
                name=maintenance_window_name,
                schedule="cron(0 0 ? * MON *)",
                duration=24,
                cutoff=1,
                allow_unassociated_targets=True,
                enabled=True,
            )

            print("-" * 88)
            print(
                "Create a document that defines the actions that Systems Manager
 performs on your EC2 instance."
            )
            document_name = q.ask(
                "Please enter the document name (default is ssmdocument):"
            )

            if not document_name:
                document_name = "ssmdocument"

            self.document_wrapper.create(
                name=document_name,
                content="""
{
    "schemaVersion": "2.2",
```

```
    "description": "Run a simple shell command",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runEchoCommand",
            "inputs": {
              "runCommand": [
                "echo 'Hello, world!'"
              ]
            }
        }
    ]
}
            """,
            )

            self.document_wrapper.wait_until_active()

            print(
                """
Now you have the option of running a command on an EC2 instance that echoes 'Hello,
 world!'.
In order to run this command, you must provide the instance ID of a Linux EC2
 instance. If you do
not already have a running Linux EC2 instance in your account, you can create one
 using the AWS console.
For information about creating an EC2 instance, see
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-instance-wizard.html.
            """
            )

            if q.ask(
                "Would you like to run a command on an EC2 instance? (y/n)",
                q.is_yesno,
            ):
                instance_id = q.ask(
                    "Please enter the instance ID of the EC2 instance:", q.non_empty
                )
                command_id = self.document_wrapper.send_command(
                    instance_ids=[instance_id]
                )

                self.document_wrapper.wait_command_executed(
                    command_id=command_id, instance_id=instance_id
```

```
                )

                print("-" * 88)
                print(
                    "Lets get the time when the specific command was sent to the
 specific managed node"
                )
                q.ask("Please hit Enter")


 self.document_wrapper.list_command_invocations(instance_id=instance_id)

            print("-" * 88)
            print("-" * 88)
            print(
                """
Now we will create a  Systems Manager OpsItem.
An OpsItem is a feature provided by the Systems Manager service.
It is a type of operational data item that allows you to manage and track various
 operational issues,
events, or tasks within your AWS environment.

You can create OpsItems to track and manage operational issues as they arise.
For example, you could create an OpsItem whenever your application detects a
 critical error
or an anomaly in your infrastructure.
            """
            )
            q.ask("Please hit Enter")

            self.ops_item_wrapper.create(
                title="Disk Space Alert",
                description="Created by the Systems Manager Python (Boto3) API",
                source="EC2",
                category="Performance",
                severity="2",
            )

            print("-" * 88)
            print("-" * 88)
            print(f"Now we will update  the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(
```

```
                title="Disk Space Alert",
                description=f"An update to {self.ops_item_wrapper.id}",
            )

            print(
                f"Now we will get the status of the OpsItem
{self.ops_item_wrapper.id}"
            )
            q.ask("Please hit Enter")

            # It may take a second for the ops item to be available
            counter = 0
            while not self.ops_item_wrapper.describe() and counter < 5:
                counter += 1
                time.sleep(1)

            print(f"Now we will resolve the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(status="Resolved")

            print("-" * 88)
            print("-" * 88)
            if q.ask(
                "Would you like to delete the Systems Manager resources? (y/n)",
                q.is_yesno,
            ):
                print("You selected to delete the resources.")
                self.cleanup()
            else:
                print("The Systems Manager resources will not be deleted")

            print("-" * 88)
            print("This concludes the Systems Manager SDK Getting Started
scenario.")
            print("-" * 88)

        except Exception:
            self.cleanup()
            raise

    def cleanup(self):
        self.maintenance_window_wrapper.delete()
        self.ops_item_wrapper.delete()
```

```
        self.document_wrapper.delete()


if __name__ == "__main__":
    try:
        scenario = SystemsManagerScenario(
            DocumentWrapper.from_client(),
            MaintenanceWindowWrapper.from_client(),
            OpsItemWrapper.from_client(),
        )
        scenario.run()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Defina uma classe que englobe as ações de documentos e comandos.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
```

```
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
```

```
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
TimeoutSeconds=3600
            )
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self):
        """
        Describes the document.

        :return: Document status.
        """
        try:
            response = self.ssm_client.describe_document(Name=self.name)
            return response["Document"]["Status"]
        except ClientError as err:
            logger.error(
                "Couldn't get %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def wait_until_active(self, max_attempts=20, delay=5):
        """
        Waits until the document is active.

        :param max_attempts: The maximum number of attempts for checking the status.
        :param delay: The delay in seconds between each check.
        """
        attempt = 0
        status = ""
        while attempt <= max_attempts:
```

```
            status = self.describe()
            if status == "Active":
                break
            attempt += 1
            time.sleep(delay)

        if status != "Active":
            logger.error("Document is not active.")
        else:
            logger.info("Document is active.")

    def wait_command_executed(self, command_id, instance_id):
        """
        Waits until the command is executed on the instance.

        :param command_id: The ID of the command.
        :param instance_id: The ID of the instance.
        """

        waiter = self.ssm_client.get_waiter("command_executed")
        waiter.wait(CommandId=command_id, InstanceId=instance_id)

    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
```

```
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
 {command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Defina uma classe que englobe as ações de itens de operação.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem
```

```
        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self):
        """
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator": "Equal"}
                ]
            ):
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
                print(
                    f"The item title is {item['Title']} and the status is
{item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.
```

```
        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
            args["Description"] = description
        if status is not None:
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Defina uma classe que englobe as ações de janelas de manutenção.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
```

```
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff, allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes as
targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance window
%s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(
        self, name, enabled, schedule, duration, cutoff, allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on managed
nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes as
targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
```

```
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to update maintenance window
%s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Ações

`CreateDocument`

O código de exemplo a seguir mostra como usar `CreateDocument`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
```

```
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateDocument Referência da API AWS SDK for Python (Boto3).

`CreateMaintenanceWindow`

O código de exemplo a seguir mostra como usar `CreateMaintenanceWindow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
```

```
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff, allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes as
targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance window
%s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateMaintenanceWindow Referência da API AWS SDK for Python (Boto3).

`CreateOpsItem`

O código de exemplo a seguir mostra como usar `CreateOpsItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem
```

```
        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateOpsItem Referência da API AWS SDK for Python (Boto3).

`DeleteDocument`

O código de exemplo a seguir mostra como usar `DeleteDocument`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Para obter detalhes da API, consulte a DeleteDocument Referência da API AWS SDK for Python (Boto3).

`DeleteMaintenanceWindow`

O código de exemplo a seguir mostra como usar `DeleteMaintenanceWindow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
```

```
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteMaintenanceWindow Referência da API AWS SDK for Python (Boto3).

`DeleteOpsItem`

O código de exemplo a seguir mostra como usar `DeleteOpsItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
```

```
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteOpsItem Referência da API AWS SDK for Python (Boto3).

`DescribeOpsItems`

O código de exemplo a seguir mostra como usar `DescribeOpsItems`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def describe(self):
        """
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator": "Equal"}
                ]
            ):
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
```

```
                print(
                    f"The item title is {item['Title']} and the status is
 {item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DescribeOpsItems Referência da API AWS SDK for Python (Boto3).

`ListCommandInvocations`

O código de exemplo a seguir mostra como usar `ListCommandInvocations`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None
```

```
    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a ListCommandInvocations Referência da API AWS SDK for Python (Boto3).

SendCommand

O código de exemplo a seguir mostra como usar SendCommand.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
 TimeoutSeconds=3600
```

```
            )
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a SendCommand Referência da API AWS SDK for Python (Boto3).

`UpdateMaintenanceWindow`

O código de exemplo a seguir mostra como usar `UpdateMaintenanceWindow`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
```

```
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def update(
        self, name, enabled, schedule, duration, cutoff, allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on managed
nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes as
targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to update maintenance window
%s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
```

```
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateMaintenanceWindow Referência da API AWS SDK for Python (Boto3).

`UpdateOpsItem`

O código de exemplo a seguir mostra como usar `UpdateOpsItem`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)
```

```
    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
            args["Description"] = description
        if status is not None:
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateOpsItem Referência da API AWS SDK for Python (Boto3).

Exemplos do Amazon Textract usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Textract.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AnalyzeDocument`

O código de exemplo a seguir mostra como usar `AnalyzeDocument`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def analyze_file(
        self, feature_types, *, document_file_name=None, document_bytes=None
    ):
```

```
        """
        Detects text and additional elements, such as forms or tables, in a local
image
        file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, "rb") as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={"Bytes": document_bytes}, FeatureTypes=feature_types
            )
            logger.info("Detected %s blocks.", len(response["Blocks"]))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a AnalyzeDocument Referência da API AWS SDK for Python (Boto3).

`DetectDocumentText`

O código de exemplo a seguir mostra como usar `DetectDocumentText`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, "rb") as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.detect_document_text(
                Document={"Bytes": document_bytes}
            )
            logger.info("Detected %s blocks.", len(response["Blocks"]))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a DetectDocumentText Referência da API AWS SDK for Python (Boto3).

`GetDocumentAnalysis`

O código de exemplo a seguir mostra como usar `GetDocumentAnalysis`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                 detected in the image.
        """
        try:
            response = self.textract_client.get_document_analysis(JobId=job_id)
            job_status = response["JobStatus"]
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
```

```
            return response
```

- Para obter detalhes da API, consulte a GetDocumentAnalysis Referência da API AWS SDK for Python (Boto3).

StartDocumentAnalysis

O código de exemplo a seguir mostra como usar `StartDocumentAnalysis`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Iniciar um trabalho assíncrono para analisar um documento.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def start_analysis_job(
        self,
        bucket_name,
        document_file_name,
        feature_types,
        sns_topic_arn,
```

```
        sns_role_arn,
    ):
        """
        Starts an asynchronous job to detect text and additional elements, such as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract
publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
image.
        :param document_file_name: The name of the document image stored in Amazon
S3.
        :param feature_types: The types of additional document features to detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS topic
                              where job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management (IAM)
                             role that can be assumed by Textract and grants
permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    "S3Object": {"Bucket": bucket_name, "Name": document_file_name}
                },
                NotificationChannel={
                    "SNSTopicArn": sns_topic_arn,
                    "RoleArn": sns_role_arn,
                },
                FeatureTypes=feature_types,
            )
            job_id = response["JobId"]
            logger.info(
                "Started text analysis job %s on %s.", job_id, document_file_name
            )
        except ClientError:
            logger.exception("Couldn't analyze text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- Para obter detalhes da API, consulte a StartDocumentAnalysis Referência da API AWS SDK for Python (Boto3).

StartDocumentTextDetection

O código de exemplo a seguir mostra como usar `StartDocumentTextDetection`.

SDK para Python (Boto3)

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Iniciar um trabalho assíncrono para detectar texto em um documento.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def start_detection_job(
        self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn
    ):
        """
        Starts an asynchronous job to detect text elements in an image stored in an
        Amazon S3 bucket. Textract publishes a notification to the specified Amazon
 SNS
        topic when the job completes.
```

```
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
image.
        :param document_file_name: The name of the document image stored in Amazon
S3.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS topic
                              where the job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management (IAM)
                             role that can be assumed by Textract and grants
permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_text_detection(
                DocumentLocation={
                    "S3Object": {"Bucket": bucket_name, "Name": document_file_name}
                },
                NotificationChannel={
                    "SNSTopicArn": sns_topic_arn,
                    "RoleArn": sns_role_arn,
                },
            )
            job_id = response["JobId"]
            logger.info(
                "Started text detection job %s on %s.", job_id, document_file_name
            )
        except ClientError:
            logger.exception("Couldn't detect text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- Para obter detalhes da API, consulte a StartDocumentTextDetection Referência da API AWS SDK for Python (Boto3).

Cenários

Criar uma aplicação de exploração do Amazon Textract

O exemplo de código a seguir mostra como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulário e tabela em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.
- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Identidade do Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Detectar entidades em texto extraído de uma imagem

O exemplo de código a seguir mostra como usar o Amazon Comprehend para detectar entidades em texto extraído pelo Amazon Textract de uma imagem armazenada no Amazon S3.

SDK para Python (Boto3)

Mostra como usar o AWS SDK para Python (Boto3) em um notebook Jupyter para detectar entidades no texto que é extraído de uma imagem. Este exemplo usa o Amazon Textract para extrair texto de uma imagem armazenada no Amazon Simple Storage Service (Amazon S3) e no Amazon Comprehend para detectar entidades no texto extraído.

Este exemplo é um caderno Jupyter e deve ser executado em um ambiente que possa hospedar blocos de anotações. Para obter instruções sobre como executar o exemplo usando o Amazon SageMaker AI, consulte as instruções em TextractAndComprehendNotebook.ipynb.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Exemplos do Amazon Transcribe usando o SDK para Python (Boto3)

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Python (Boto3) com o Amazon Transcribe.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateVocabulary`

O código de exemplo a seguir mostra como usar `CreateVocabulary`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None, table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
```

```
        logger.exception("Couldn't create custom vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- Para obter detalhes da API, consulte a CreateVocabulary Referência da API AWS SDK for Python (Boto3).

`DeleteTranscriptionJob`

O código de exemplo a seguir mostra como usar `DeleteTranscriptionJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteTranscriptionJob Referência da API AWS SDK for Python (Boto3).

`DeleteVocabulary`

O código de exemplo a seguir mostra como usar `DeleteVocabulary`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

- Para obter detalhes da API, consulte a DeleteVocabulary Referência da API AWS SDK for Python (Boto3).

`GetTranscriptionJob`

O código de exemplo a seguir mostra como usar `GetTranscriptionJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job
```

- Para obter detalhes da API, consulte a GetTranscriptionJob Referência da API AWS SDK for Python (Boto3).

`GetVocabulary`

O código de exemplo a seguir mostra como usar `GetVocabulary`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response = transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- Para obter detalhes da API, consulte a GetVocabulary Referência da API AWS SDK for Python (Boto3).

`ListTranscriptionJobs`

O código de exemplo a seguir mostra como usar `ListTranscriptionJobs`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_jobs(job_filter, transcribe_client):
    """
    Lists summaries of the transcription jobs for the current AWS account.

    :param job_filter: The list of returned jobs must contain this string in their
                       names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved transcription job summaries.
    """
    try:
        response =
 transcribe_client.list_transcription_jobs(JobNameContains=job_filter)
        jobs = response["TranscriptionJobSummaries"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_transcription_jobs(
                JobNameContains=job_filter, NextToken=next_token
            )
            jobs += response["TranscriptionJobSummaries"]
            next_token = response.get("NextToken")
        logger.info("Got %s jobs with filter %s.", len(jobs), job_filter)
    except ClientError:
        logger.exception("Couldn't get jobs with filter %s.", job_filter)
        raise
    else:
        return jobs
```

- Para obter detalhes da API, consulte a ListTranscriptionJobs Referência da API AWS SDK for Python (Boto3).

`ListVocabularies`

O código de exemplo a seguir mostra como usar `ListVocabularies`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs
```

- Para obter detalhes da API, consulte a ListVocabularies Referência da API AWS SDK for Python (Boto3).

StartTranscriptionJob

O código de exemplo a seguir mostra como usar `StartTranscriptionJob`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when transcribing
                            the audio file.
```

```
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job
```

- Para obter detalhes da API, consulte a StartTranscriptionJob Referência da API AWS SDK for Python (Boto3).

`UpdateVocabulary`

O código de exemplo a seguir mostra como usar `UpdateVocabulary`.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None, table_uri=None
):
```

```
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.", vocabulary_name)
        raise
```

- Para obter detalhes da API, consulte a UpdateVocabulary Referência da API AWS SDK for Python (Boto3).

Cenários

Criar e refinar um vocabulário personalizado

O exemplo de código a seguir mostra como:

- Fazer upload de um arquivo de áudio para o Amazon S3.
- Executar um trabalho do Amazon Transcribe para transcrever o arquivo e obter os resultados.
- Criar e refinar um vocabulário personalizado para melhorar a precisão da transcrição.
- Executar trabalhos com vocabulários personalizados e obter os resultados.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Transcreva um arquivo de áudio que contenha uma leitura de Jabberwocky de Lewis Carroll. Comece criando funções que envolvam as ações do Amazon Transcribe.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
```

```
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job



def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job



def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
```

```
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise



def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None, table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response = transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response



def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None, table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.", vocabulary_name)
```

```
        raise



def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs



def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
```

```
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

Chame as funções do wrapper para transcrever áudio sem um vocabulário personalizado e, em seguida, com diferentes versões de um vocabulário personalizado para ver os melhores resultados.

```
def usage_demo():
    """Shows how to use the Amazon Transcribe service."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    s3_resource = boto3.resource("s3")
    transcribe_client = boto3.client("transcribe")

    print("-" * 88)
    print("Welcome to the Amazon Transcribe demo!")
    print("-" * 88)

    bucket_name = f"jabber-bucket-{time.time_ns()}"
    print(f"Creating bucket {bucket_name}.")
    bucket = s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": transcribe_client.meta.region_name
        },
    )
    media_file_name = ".media/Jabberwocky.mp3"
    media_object_key = "Jabberwocky.mp3"
    print(f"Uploading media file {media_file_name}.")
    bucket.upload_file(media_file_name, media_object_key)
    media_uri = f"s3://{bucket.name}/{media_object_key}"

    job_name_simple = f"Jabber-{time.time_ns()}"
    print(f"Starting transcription job {job_name_simple}.")
    start_job(
        job_name_simple,
```

```
        f"s3://{bucket_name}/{media_object_key}",
        "mp3",
        "en-US",
        transcribe_client,
    )
    transcribe_waiter = TranscribeCompleteWaiter(transcribe_client)
    transcribe_waiter.wait(job_name_simple)
    job_simple = get_job(job_name_simple, transcribe_client)
    transcript_simple = requests.get(
        job_simple["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_simple['jobName']}:")
    print(transcript_simple["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Creating a custom vocabulary that lists the nonsense words to try to "
        "improve the transcription."
    )
    vocabulary_name = f"Jabber-vocabulary-{time.time_ns()}"
    create_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        phrases=[
            "brillig",
            "slithy",
            "borogoves",
            "mome",
            "raths",
            "Jub-Jub",
            "frumious",
            "manxome",
            "Tumtum",
            "uffish",
            "whiffling",
            "tulgey",
            "thou",
            "frabjous",
            "callooh",
            "callay",
            "chortled",
        ],
    )
```

```
    vocabulary_ready_waiter = VocabularyReadyWaiter(transcribe_client)
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocabulary_list = f"Jabber-vocabulary-list-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocabulary_list}.")
    start_job(
        job_name_vocabulary_list,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocabulary_list)
    job_vocabulary_list = get_job(job_name_vocabulary_list, transcribe_client)
    transcript_vocabulary_list = requests.get(
        job_vocabulary_list["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocabulary_list['jobName']}:")
    print(transcript_vocabulary_list["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Updating the custom vocabulary with table data that provides additional "
        "pronunciation hints."
    )
    table_vocab_file = "jabber-vocabulary-table.txt"
    bucket.upload_file(table_vocab_file, table_vocab_file)
    update_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        table_uri=f"s3://{bucket.name}/{table_vocab_file}",
    )
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocab_table = f"Jabber-vocab-table-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocab_table}.")
    start_job(
        job_name_vocab_table,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
```

```
        vocabulary_name=vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocab_table)
    job_vocab_table = get_job(job_name_vocab_table, transcribe_client)
    transcript_vocab_table = requests.get(
        job_vocab_table["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocab_table['jobName']}:")
    print(transcript_vocab_table["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print("Getting data for jobs and vocabularies.")
    jabber_jobs = list_jobs("Jabber", transcribe_client)
    print(f"Found {len(jabber_jobs)} jobs:")
    for job_sum in jabber_jobs:
        job = get_job(job_sum["TranscriptionJobName"], transcribe_client)
        print(
            f"\t{job['TranscriptionJobName']}, {job['Media']['MediaFileUri']}, "
            f"{job['Settings'].get('VocabularyName')}"
        )

    jabber_vocabs = list_vocabularies("Jabber", transcribe_client)
    print(f"Found {len(jabber_vocabs)} vocabularies:")
    for vocab_sum in jabber_vocabs:
        vocab = get_vocabulary(vocab_sum["VocabularyName"], transcribe_client)
        vocab_content = requests.get(vocab["DownloadUri"]).text
        print(f"\t{vocab['VocabularyName']} contents:")
        print(vocab_content)

    print("-" * 88)
    print("Deleting demo jobs.")
    for job_name in [job_name_simple, job_name_vocabulary_list,
job_name_vocab_table]:
        delete_job(job_name, transcribe_client)
    print("Deleting demo vocabulary.")
    delete_vocabulary(vocabulary_name, transcribe_client)
    print("Deleting demo bucket.")
    bucket.objects.delete()
    bucket.delete()
    print("Thanks for watching!")
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - CreateVocabulary
  - DeleteTranscriptionJob
  - DeleteVocabulary
  - GetTranscriptionJob
  - GetVocabulary
  - ListVocabularies
  - StartTranscriptionJob
  - UpdateVocabulary

Transcrever áudio e obter dados do trabalho

O exemplo de código a seguir mostra como:

- Iniciar um trabalho de transcrição com o Amazon Transcribe.
- Aguardar a conclusão do trabalho.
- Obter o URI em que a transcrição está armazenada.

Para obter mais informações, consulte Começar a usar o Amazon Transcribe.

SDK para Python (Boto3)

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import time
import boto3


def transcribe_file(job_name, file_uri, transcribe_client):
    transcribe_client.start_transcription_job(
        TranscriptionJobName=job_name,
```

```
        Media={"MediaFileUri": file_uri},
        MediaFormat="wav",
        LanguageCode="en-US",
    )

    max_tries = 60
    while max_tries > 0:
        max_tries -= 1
        job = transcribe_client.get_transcription_job(TranscriptionJobName=job_name)
        job_status = job["TranscriptionJob"]["TranscriptionJobStatus"]
        if job_status in ["COMPLETED", "FAILED"]:
            print(f"Job {job_name} is {job_status}.")
            if job_status == "COMPLETED":
                print(
                    f"Download the transcript from\n"
                    f"\t{job['TranscriptionJob']['Transcript']
['TranscriptFileUri']}."
                )
            break
        else:
            print(f"Waiting for {job_name}. Current status is {job_status}.")
        time.sleep(10)


def main():
    transcribe_client = boto3.client("transcribe")
    file_uri = "s3://test-transcribe/answer2.wav"
    transcribe_file("Example-job", file_uri, transcribe_client)


if __name__ == "__main__":
    main()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
  - GetTranscriptionJob
  - StartTranscriptionJob

Exemplos de código do SDK para Ruby

Os exemplos de código a seguir mostram como usar o AWS SDK para Ruby with AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para Ruby — Saiba mais sobre como usar o Ruby com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos de Aurora usando o SDK para Ruby
- Exemplos do Auto Scaling usando o SDK para Ruby
- CloudTrail exemplos de uso do SDK para Ruby
- CloudWatch exemplos de uso do SDK para Ruby
- Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Ruby
- Exemplos do Amazon Comprehend usando o SDK para Ruby
- Exemplos do Amazon DocumentDB usando o SDK para Ruby
- Exemplos de código do DynamoDB usando o SDK para Ruby
- Exemplos do Amazon EC2 usando o SDK para Ruby
- Exemplos do Elastic Beanstalk usando o SDK para Ruby

- EventBridge exemplos de uso do SDK para Ruby
- AWS Glue exemplos de uso do SDK para Ruby
- Exemplos do IAM usando o SDK para Ruby
- Exemplos do Kinesis usando o SDK para Ruby
- AWS KMS exemplos de uso do SDK para Ruby
- Exemplos de Lambda usando o SDK para Ruby
- Exemplos do Amazon MSK usando o SDK para Ruby
- Exemplos do Amazon Polly usando o SDK para Ruby
- Exemplos de código para o Amazon RDS usando o SDK para Ruby
- Exemplos de código do Amazon S3 usando o SDK para Ruby
- Exemplos do Amazon SES usando o SDK para Ruby
- Exemplos da API v2 do Amazon SES usando o SDK para Ruby
- Exemplos do Amazon SNS usando o SDK para Ruby
- Exemplos do Amazon SQS usando o SDK para Ruby
- AWS STS exemplos de uso do SDK para Ruby
- Exemplos do Amazon Textract usando o SDK para Ruby
- Exemplos do Amazon Translate usando o SDK para Ruby

Exemplos de Aurora usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Aurora.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Olá, Aurora

O exemplo de código a seguir mostra como começar a usar o Aurora.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Ruby .

Exemplos do Auto Scaling usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com Auto Scaling.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Olá, Auto Scaling

O exemplo de código a seguir mostra como começar a usar o Auto Scaling.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
```

```
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:       #{group.min_size}/#{group.max_size}/
#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- Para obter detalhes da API, consulte DescribeAutoScalingGroups em Referência AWS SDK para Ruby da API.

CloudTrail exemplos de uso do SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby with CloudTrail.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateTrail`

O código de exemplo a seguir mostra como usar `CreateTrail`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'
require 'aws-sdk-s3'
require 'aws-sdk-sts'

def create_trail_example(s3_client, sts_client, cloudtrail_client, trail_name,
 bucket_name)
  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to an Amazon Simple Storage Service (S3) bucket.
  s3_client.create_bucket(bucket: bucket_name)
  begin
    policy = {
      'Version' => '2012-10-17',
      'Statement' => [
        {
          'Sid' => 'AWSCloudTrailAclCheck20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:GetBucketAcl',
          'Resource' => "arn:aws:s3:::#{bucket_name}"
        },
        {
          'Sid' => 'AWSCloudTrailWrite20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:PutObject',
          'Resource' => "arn:aws:s3:::#{bucket_name}/AWSLogs/#{account_id}/*",
          'Condition' => {
            'StringEquals' => {
```

```
              's3:x-amz-acl' => 'bucket-owner-full-control'
            }
          }
        }
      ]
    }.to_json

    s3_client.put_bucket_policy(
      bucket: bucket_name,
      policy: policy
    )
    puts "Successfully added policy to bucket #{bucket_name}"
  end

  begin
    cloudtrail_client.create_trail({
                                     name: trail_name, # required
                                     s3_bucket_name: bucket_name # required
                                   })

    puts "Successfully created trail: #{trail_name}."
  rescue StandardError => e
    puts "Got error trying to create trail #{trail_name}:\n #{e}"
    puts e
    exit 1
  end
```

- Para obter detalhes da API, consulte CreateTrail em Referência AWS SDK para Ruby da API.

`DeleteTrail`

O código de exemplo a seguir mostra como usar `DeleteTrail`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  client.delete_trail({
                        name: trail_name # required
                      })
  puts "Successfully deleted trail: #{trail_name}"
rescue StandardError => e
  puts "Got error trying to delete trail: #{trail_name}:"
  puts e
  exit 1
end
```

- Para obter detalhes da API, consulte DeleteTrail em Referência AWS SDK para Ruby da API.

`ListTrails`

O código de exemplo a seguir mostra como usar `ListTrails`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

def describe_trails_example(client)
  resp = client.describe_trails({})
  puts "Found #{resp.trail_list.count} trail(s)."

  resp.trail_list.each do |trail|
    puts "Name:           #{trail.name}"
    puts "S3 bucket name: #{trail.s3_bucket_name}"
    puts
  end
```

- Para obter detalhes da API, consulte ListTrails em Referência AWS SDK para Ruby da API.

`LookupEvents`

O código de exemplo a seguir mostra como usar `LookupEvents`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

# @param [Object] client
def lookup_events_example(client)
  resp = client.lookup_events
  puts "Found #{resp.events.count} events:"
  resp.events.each do |e|
    puts "Event name:   #{e.event_name}"
    puts "Event ID:     #{e.event_id}"
    puts "Event time:   #{e.event_time}"
    puts 'Resources:'

    e.resources.each do |r|
      puts "  Name:       #{r.resource_name}"
      puts "  Type:       #{r.resource_type}"
      puts ''
    end
  end
end
```

- Para obter detalhes da API, consulte LookupEvents em Referência AWS SDK para Ruby da API.

CloudWatch exemplos de uso do SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby with CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DescribeAlarms`

O código de exemplo a seguir mostra como usar `DescribeAlarms`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudwatch'

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts 'No alarms found.'
  end
```

```
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- Para obter detalhes da API, consulte DescribeAlarms em Referência AWS SDK para Ruby da API.

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts '-' * 16
      puts "Name:           #{alarm.alarm_name}"
      puts "State value:    #{alarm.state_value}"
      puts "State reason:   #{alarm.state_reason}"
      puts "Metric:         #{alarm.metric_name}"
      puts "Namespace:      #{alarm.namespace}"
      puts "Statistic:      #{alarm.statistic}"
      puts "Period:         #{alarm.period}"
      puts "Unit:           #{alarm.unit}"
```

```
      puts "Eval. periods:  #{alarm.evaluation_periods}"
      puts "Threshold:      #{alarm.threshold}"
      puts "Comp. operator: #{alarm.comparison_operator}"

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts 'OK actions:'
        alarm.ok_actions.each do |a|
          puts "  #{a}"
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
        puts 'Alarm actions:'
        alarm.alarm_actions.each do |a|
          puts "  #{a}"
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
         alarm.insufficient_data_actions.count.positive?
        puts 'Insufficient data actions:'
        alarm.insufficient_data_actions.each do |a|
          puts "  #{a}"
        end
      end

      puts 'Dimensions:'
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: #{d.name}, Value: #{d.value}"
        end
      else
        puts '  None for this alarm.'
      end
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
```

```
  region = ''

  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby cw-ruby-example-show-alarms.rb REGION'
    puts 'Example: ruby cw-ruby-example-show-alarms.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts 'Available alarms:'
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte DescribeAlarmsForMetric em Referência AWS SDK para Ruby da API.

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Disables an alarm in Amazon CloudWatch.
#
```

```
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  false
end

# Example usage:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: "BucketName",
      value: "amzn-s3-demo-bucket"
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
```

```
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

  if alarm_actions_disabled?(cloudwatch_client, alarm_name)
    puts "Alarm '#{alarm_name}' disabled."
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte DisableAlarmActions em Referência AWS SDK para Ruby da API.

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts '    Dimensions:'
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts 'No dimensions found.'
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      'Note that it could take up to 15 minutes for recently-added metrics ' \
      'to become available.'
  end
end
```

```
# Example usage:
def run_me
  metric_namespace = 'SITE/TRAFFIC'
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisitors',
    'SiteName',
    'example.com',
    5_885.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisits',
    'SiteName',
    'example.com',
    8_628.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'PageViews',
    'PageURL',
    'example.html',
    18_057.0,
    'Count'
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte ListMetrics em Referência AWS SDK para Ruby da API.

PutMetricAlarm

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
```

```
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'amzn-s3-demo-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
```

```
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  false
end
```

- Para obter detalhes da API, consulte PutMetricAlarm em Referência AWS SDK para Ruby da API.

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cloudwatch'

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
```

```
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  true
```

```
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  false
end
```

- Para obter detalhes da API, consulte PutMetricData em Referência AWS SDK para Ruby da API.

Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby Amazon Cognito Identity Provider.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

Conceitos básicos

Olá, Amazon Cognito

O exemplo de código a seguir mostra como começar a usar o Amazon Cognito.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

- Para obter detalhes da API, consulte ListUserPools em Referência AWS SDK para Ruby da API.

Exemplos do Amazon Comprehend usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon Comprehend.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon DocumentDB usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
```

```
    log_document_db_event(record)
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

Exemplos de código do DynamoDB usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

SDK para Ruby

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-dynamodb'
require 'logger'

# DynamoDBManager is a class responsible for managing DynamoDB operations
# such as listing all tables in the current AWS account.
class DynamoDBManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all DynamoDB tables in the current AWS account.
  def list_tables
    @logger.info('Here are the DynamoDB tables in your account:')

    paginator = @client.list_tables(limit: 10)
    table_names = []

    paginator.each_page do |page|
      page.table_names.each do |table_name|
        @logger.info("- #{table_name}")
        table_names << table_name
      end
```

```
    end

    if table_names.empty?
      @logger.info("You don't have any DynamoDB tables in your account.")
    else
      @logger.info("\nFound #{table_names.length} tables.")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  dynamodb_client = Aws::DynamoDB::Client.new
  manager = DynamoDBManager.new(dynamodb_client)
  manager.list_tables
end
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para Ruby da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que encapsule uma tabela do DynamoDB.

```
  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        { attribute_name: 'year', key_type: 'HASH' }, # Partition key
        { attribute_name: 'title', key_type: 'RANGE' } # Sort key
      ],
      attribute_definitions: [
        { attribute_name: 'year', attribute_type: 'N' },
        { attribute_name: 'title', attribute_type: 'S' }
      ],
      billing_mode: 'PAY_PER_REQUEST'
    )
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

Crie uma função auxiliar para baixar e extrair o arquivo JSON de exemplo.

```
  # Gets sample movie data, either from a local file or by first downloading it from
  # the Amazon DynamoDB Developer Guide.
  #
  # @param movie_file_name [String] The local file name where the movie data is
 stored in JSON format.
  # @return [Hash] The movie data as a Hash.
  def fetch_movie_data(movie_file_name)
    if !File.file?(movie_file_name)
      @logger.debug("Downloading #{movie_file_name}...")
      movie_content = URI.open(
        'https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
moviedata.zip'
```

```
      )
      movie_json = ''
      Zip::File.open_buffer(movie_content) do |zip|
        zip.each do |entry|
          movie_json = entry.get_input_stream.read
        end
      end
    else
      movie_json = File.read(movie_file_name)
    end
    movie_data = JSON.parse(movie_json)
    # The sample file lists over 4000 movies. This returns only the first 250.
    movie_data.slice(0, 250)
  rescue StandardError => e
    puts("Failure downloading movie data:\n#{e}")
    raise
  end
```

Execute um cenário interativo para criar a tabela e executar ações nela.

```
  table_name = "doc-example-table-movies-#{rand(10**4)}"
  scaffold = Scaffold.new(table_name)
  dynamodb_wrapper = DynamoDBBasics.new(table_name)

  new_step(1, 'Create a new DynamoDB table if none already exists.')
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, 'Add a new record to the DynamoDB table.')
  my_movie = {}
  my_movie[:title] = CLI::UI::Prompt.ask('Enter the title of a movie to add to the
table. E.g. The Matrix')
  my_movie[:year] = CLI::UI::Prompt.ask('What year was it released? E.g. 1989').to_i
  my_movie[:rating] = CLI::UI::Prompt.ask('On a scale of 1 - 10, how do you rate it?
E.g. 7').to_i
  my_movie[:plot] = CLI::UI::Prompt.ask('Enter a brief summary of the plot. E.g. A
man awakens to a new reality.')
  dynamodb_wrapper.add_item(my_movie)
  puts("\nNew record added:")
```

```
 puts JSON.pretty_generate(my_movie).green
 print "Done!\n".green

 new_step(3, 'Update a record in the DynamoDB table.')
 my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with a
new rating, e.g. 3:").to_i
 response = dynamodb_wrapper.update_item(my_movie)
 puts("Updated '#{my_movie[:title]}' with new attributes:")
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(4, 'Get a record from the DynamoDB table.')
 puts("Searching for #{my_movie[:title]} (#{my_movie[:year]})...")
 response = dynamodb_wrapper.get_item(my_movie[:title], my_movie[:year])
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(5, 'Write a batch of items into the DynamoDB table.')
 download_file = 'moviedata.json'
 puts("Downloading movie database to #{download_file}...")
 movie_data = scaffold.fetch_movie_data(download_file)
 puts("Writing movie data from #{download_file} into your table...")
 scaffold.write_batch(movie_data)
 puts("Records added: #{movie_data.length}.")
 print "Done!\n".green

 new_step(5, 'Query for a batch of items by key.')
 loop do
   release_year = CLI::UI::Prompt.ask('Enter a year between 1972 and 2018, e.g.
1999:').to_i
   results = dynamodb_wrapper.query_items(release_year)
   if results.any?
     puts("There were #{results.length} movies released in #{release_year}:")
     results.each do |movie|
       print "\t #{movie['title']}".green
     end
     break
   else
     continue = CLI::UI::Prompt.ask("Found no movies released in #{release_year}!
Try another year? (y/n)")
     break unless continue.eql?('y')
   end
 end
 print "\nDone!\n".green
```

```
  new_step(6, 'Scan for a batch of items using a filter expression.')
  years = {}
  years[:start] = CLI::UI::Prompt.ask('Enter a starting year between 1972 and
 2018:')
  years[:end] = CLI::UI::Prompt.ask('Enter an ending year between 1972 and 2018:')
  releases = dynamodb_wrapper.scan_items(years)
  if !releases.empty?
    puts("Found #{releases.length} movies.")
    count = Question.ask(
      'How many do you want to see? ', method(:is_int), in_range(1, releases.length)
    )
    puts("Here are your #{count} movies:")
    releases.take(count).each do |release|
      puts("\t#{release['title']}")
    end
  else
    puts("I don't know about any movies released between #{years[:start]} "\
         "and #{years[:end]}.")
  end
  print "\nDone!\n".green

  new_step(7, 'Delete an item from the DynamoDB table.')
  answer = CLI::UI::Prompt.ask("Do you want to remove '#{my_movie[:title]}'? (y/n)
 ")
  if answer.eql?('y')
    dynamodb_wrapper.delete_item(my_movie[:title], my_movie[:year])
    puts("Removed '#{my_movie[:title]}' from the table.")
    print "\nDone!\n".green
  end

  new_step(8, 'Delete the DynamoDB table.')
  answer = CLI::UI::Prompt.ask('Delete the table? (y/n)')
  if answer.eql?('y')
    scaffold.delete_table
    puts("Deleted #{table_name}.")
  else
    puts("Don't forget to delete the table when you're done!")
  end
  print "\nThanks for watching!\n".green
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo.')
rescue Errno::ENOENT
  true
```

```
end
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`BatchExecuteStatement`

O código de exemplo a seguir mostra como usar `BatchExecuteStatement`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Leia um lote de itens usando o PartiQL.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
```

```
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Selects a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_select(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "SELECT * FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

Exclua um lote de itens usando o PartiQL.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_write(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para Ruby da API.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Fills an Amazon DynamoDB table with the specified data. Items are sent in
  # batches of 25 until all items are written.
  #
  # @param movies [Enumerable] The data to put in the table. Each item must contain
 at least
  #                            the keys required by the schema that was specified
 when the
  #                            table was created.
  def write_batch(movies)
    index = 0
    slice_size = 25
    while index < movies.length
      movie_items = []
      movies[index, slice_size].each do |movie|
        movie_items.append({ put_request: { item: movie } })
```

```
      end
      @dynamo_resource.client.batch_write_item({ request_items: { @table.name =>
 movie_items } })
      index += slice_size
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts(
      "Couldn't load data into table #{@table.name}. Here's why:"
    )
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte BatchWriteItem em Referência AWS SDK para Ruby da API.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end
```

```
# Creates an Amazon DynamoDB table that can be used to store movie data.
# The table uses the release year of the movie as the partition key and the
# title as the sort key.
#
# @param table_name [String] The name of the table to create.
# @return [Aws::DynamoDB::Table] The newly created table.
def create_table(table_name)
  @table = @dynamo_resource.create_table(
    table_name: table_name,
    key_schema: [
      { attribute_name: 'year', key_type: 'HASH' }, # Partition key
      { attribute_name: 'title', key_type: 'RANGE' } # Sort key
    ],
    attribute_definitions: [
      { attribute_name: 'year', attribute_type: 'N' },
      { attribute_name: 'title', attribute_type: 'S' }
    ],
    billing_mode: 'PAY_PER_REQUEST'
  )
  @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
  @table
rescue Aws::DynamoDB::Errors::ServiceError => e
  @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
  raise
end
```

- Para obter detalhes da API, consulte CreateTable em Referência AWS SDK para Ruby da API.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
  # @param year [Integer] The release year of the movie to delete.
  def delete_item(title, year)
    @table.delete_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteItem em Referência AWS SDK para Ruby da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
```

```
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Deletes the table.
  def delete_table
    @table.delete
    @table = nil
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete table. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para Ruby da API.

`DescribeTable`

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
```

```
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DescribeTable em Referência AWS SDK para Ruby da API.

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Selecione um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle
```

```
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Gets a single record from a table using PartiQL.
  # Note: To perform more fine-grained selects,
  # use the Client.query instance method instead.
  #
  # @param title [String] The title of the movie to search.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def select_item_by_title(title)
    request = {
      statement: "SELECT * FROM \"#{@table.name}\" WHERE title=?",
      parameters: [title]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Atualize um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Updates a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def update_rating_by_title(title, year, rating)
    request = {
```

```
      statement: "UPDATE \"#{@table.name}\" SET info.rating=? WHERE title=? and
 year=?",
      parameters: [{ "N": rating }, title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Adicione um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Adds a single record to a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param plot [String] The plot of the movie.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def insert_item(title, year, plot, rating)
    request = {
      statement: "INSERT INTO \"#{@table.name}\" VALUE {'title': ?, 'year': ?,
 'info': ?}",
      parameters: [title, year, { 'plot': plot, 'rating': rating }]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Exclua um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
```

```
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def delete_item_by_title(title, year)
    request = {
      statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
      parameters: [title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para Ruby da API.

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end
```

```
  # Gets movie data from the table for a specific movie.
  #
  # @param title [String] The title of the movie.
  # @param year [Integer] The release year of the movie.
  # @return [Hash] The data about the requested movie.
  def get_item(title, year)
    @table.get_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't get movie #{title} (#{year}) from table #{@table.name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte GetItem em Referência AWS SDK para Ruby da API.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Determine se uma tabela existe.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
```

```
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte ListTables em Referência AWS SDK para Ruby da API.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
```

```
    @table = @dynamo_resource.table(table_name)
  end

  # Adds a movie to the table.
  #
  # @param movie [Hash] The title, year, plot, and rating of the movie.
  def add_item(movie)
    @table.put_item(
      item: {
        'year' => movie[:year],
        'title' => movie[:title],
        'info' => { 'plot' => movie[:plot], 'rating' => movie[:rating] }
      }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte PutItem em Referência AWS SDK para Ruby da API.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
```

```
    @table = @dynamo_resource.table(table_name)
  end

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.
  # @return [Array] The list of movies that were released in the specified year.
  def query_items(year)
    response = @table.query(
      key_condition_expression: '#yr = :year',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: { ':year' => year }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't query for movies released in #{year}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.items
  end
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Ruby .

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
```

```
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
    movies = []
    scan_hash = {
      filter_expression: '#yr between :start_yr and :end_yr',
      projection_expression: '#yr, title, info.rating',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: {
        ':start_yr' => year_range[:start], ':end_yr' => year_range[:end]
      }
    }
    done = false
    start_key = nil
    until done
      scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
      response = @table.scan(scan_hash)
      movies.concat(response.items) unless response.items.empty?
      start_key = response.last_evaluated_key
      done = start_key.nil?
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't scan for movies. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    movies
  end
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Ruby .

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Updates rating and plot data for a movie in the table.
  #
  # @param movie [Hash] The title, year, plot, rating of the movie.
  def update_item(movie)
    response = @table.update_item(
      key: { 'year' => movie[:year], 'title' => movie[:title] },
      update_expression: 'set info.rating=:r',
      expression_attribute_values: { ':r' => movie[:rating] },
      return_values: 'UPDATED_NEW'
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't update movie #{movie[:title]} (#{movie[:year]}) in table
 #{@table.name}\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.attributes
  end
```

- Para obter detalhes da API, consulte UpdateItem em Referência AWS SDK para Ruby da API.

Cenários

Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obter um lote de itens executando várias instruções SELECT.
- Adicionar um lote de itens executando várias instruções INSERT.
- Atualizar um lote de itens executando várias instruções UPDATE.
- Excluir um lote de itens executando várias instruções DELETE.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário que crie uma tabela e execute consultas do PartiQL em lotes.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**4)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLBatch.new(table_name)

new_step(1, 'Create a new DynamoDB table if none already exists.')
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end

new_step(2, 'Populate DynamoDB table with movie data.')
download_file = 'moviedata.json'
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
scaffold.write_batch(movie_data)
puts("Records added: #{movie_data.length}.")
print "Done!\n".green
```

```
  new_step(3, 'Select a batch of items from the movies table.')
  puts "Let's select some popular movies for side-by-side comparison."
  response = sdk.batch_execute_select([['Mean Girls', 2004], ['Goodfellas', 1977],
 ['The Prancing of the Lambs', 2005]])
  puts("Items selected: #{response['responses'].length}\n")
  print "\nDone!\n".green

  new_step(4, 'Delete a batch of items from the movies table.')
  sdk.batch_execute_write([['Mean Girls', 2004], ['Goodfellas', 1977], ['The
 Prancing of the Lambs', 2005]])
  print "\nDone!\n".green

  new_step(5, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- Para obter detalhes da API, consulte BatchExecuteStatement em Referência AWS SDK para Ruby da API.

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário que crie uma tabela e execute consultas do PartiQL.

```
  table_name = "doc-example-table-movies-partiql-#{rand(10**8)}"
  scaffold = Scaffold.new(table_name)
  sdk = DynamoDBPartiQLSingle.new(table_name)

  new_step(1, 'Create a new DynamoDB table if none already exists.')
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, 'Populate DynamoDB table with movie data.')
  download_file = 'moviedata.json'
  puts("Downloading movie database to #{download_file}...")
  movie_data = scaffold.fetch_movie_data(download_file)
  puts("Writing movie data from #{download_file} into your table...")
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(3, 'Select a single item from the movies table.')
  response = sdk.select_item_by_title('Star Wars')
  puts("Items selected for title 'Star Wars': #{response.items.length}\n")
  print response.items.first.to_s.yellow
  print "\n\nDone!\n".green

  new_step(4, 'Update a single item from the movies table.')
  puts "Let's correct the rating on The Big Lebowski to 10.0."
  sdk.update_rating_by_title('The Big Lebowski', 1998, 10.0)
  print "\nDone!\n".green

  new_step(5, 'Delete a single item from the movies table.')
  puts "Let's delete The Silence of the Lambs because it's just too scary."
  sdk.delete_item_by_title('The Silence of the Lambs', 1991)
  print "\nDone!\n".green

  new_step(6, 'Insert a new item into the movies table.')
  puts "Let's create a less-scary movie called The Prancing of the Lambs."
  sdk.insert_item('The Prancing of the Lambs', 2005, 'A movie about happy
livestock.', 5.0)
  print "\nDone!\n".green

  new_step(7, 'Delete the table.')
```

```
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- Para obter detalhes da API, consulte ExecuteStatement em Referência AWS SDK para Ruby da API.

Exemplos sem servidor

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
```

```
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

Exemplos do Amazon EC2 usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon EC2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'
require 'logger'

# EC2Manager is a class responsible for managing EC2 operations
# such as listing all EC2 instances in the current AWS account.
class EC2Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
```

```
  end

  # Lists and prints all EC2 instances in the current AWS account.
  def list_instances
    @logger.info('Listing instances')

    instances = fetch_instances

    if instances.empty?
      @logger.info('You have no instances')
    else
      print_instances(instances)
    end
  end

  private

  # Fetches all EC2 instances using pagination.
  #
  # @return [Array<Aws::EC2::Types::Instance>] List of EC2 instances.
  def fetch_instances
    paginator = @client.describe_instances
    instances = []

    paginator.each_page do |page|
      page.reservations.each do |reservation|
        reservation.instances.each do |instance|
          instances << instance
        end
      end
    end

    instances
  end

  # Prints details of the given EC2 instances.
  #
  # @param instances [Array<Aws::EC2::Types::Instance>] List of EC2 instances to
print.
  def print_instances(instances)
    instances.each do |instance|
      @logger.info("Instance ID: #{instance.instance_id}")
      @logger.info("Instance Type: #{instance.instance_type}")
      @logger.info("Public IP: #{instance.public_ip_address}")
```

```
      @logger.info("Public DNS Name: #{instance.public_dns_name}")
      @logger.info("\n")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
  manager = EC2Manager.new(ec2_client)
  manager.list_instances
end
```

- Para obter detalhes da API, consulte DescribeSecurityGroups em Referência AWS SDK para Ruby da API.

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  response.allocation_id
rescue StandardError => e
```

```
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end
```

- Para obter detalhes da API, consulte AllocateAddress em Referência AWS SDK para Ruby da API.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
```

```
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  'Error'
end
```

- Para obter detalhes da API, consulte AssociateAddress em Referência AWS SDK para Ruby da API.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
```

```
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  false
end

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts 'No key pairs found.'
  else
    puts 'Key pair names:'
    result.key_pairs.each do |key_pair|
      puts key_pair.key_name
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end
```

```
# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end

# Example usage:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)
```

```
  puts 'Displaying existing key pair names before creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Creating key pair...'
  unless key_pair_created?(ec2_client, key_pair_name)
    puts 'Stopping program.'
    exit 1
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Deleting key pair...'
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts 'Stopping program. You must delete the key pair yourself.'
    exit 1
  end
  puts 'Key pair deleted.'

  puts '-' * 10
  puts 'Now that the key pair is deleted, ' \
    'also deleting the related private key pair file...'
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.delete(filename)
  if File.exist?(filename)
    puts "Could not delete file at '#{filename}'. You must delete it yourself."
  else
    puts 'File deleted.'
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after deleting this key pair...'
  describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateKeyPair em Referência AWS SDK para Ruby da API.

`CreateRouteTable`

O código de exemplo a seguir mostra como usar `CreateRouteTable`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
```

```
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts 'Added tags to route table.'
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts 'Created route with destination CIDR block ' \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
```

```
# Print usage information and then stop.
if ARGV[0] == '--help' || ARGV[0] == '-h'
  puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
    'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
    'TAG_KEY TAG_VALUE REGION'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
    'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
    "'0.0.0.0/0' my-key my-value us-west-2"
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  vpc_id = 'vpc-0b6f769731EXAMPLE'
  subnet_id = 'subnet-03d9303b57EXAMPLE'
  gateway_id = 'igw-06ca90c011EXAMPLE'
  destination_cidr_block = '0.0.0.0/0'
  tag_key = 'my-key'
  tag_value = 'my-value'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  subnet_id = ARGV[1]
  gateway_id = ARGV[2]
  destination_cidr_block = ARGV[3]
  tag_key = ARGV[4]
  tag_value = ARGV[5]
  region = ARGV[6]
end

ec2_resource = Aws::EC2::Resource.new(region: region)

if route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  puts 'Route table created and associated.'
else
```

```
    puts 'Route table not created or not associated.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateRouteTable em Referência AWS SDK para Ruby da API.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
```

```
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(ec2_client, group_name, description, vpc_id)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
```

```
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  false
end

# Refactored method to simplify complexity for describing security group permissions
def format_port_information(perm)
  from_port_str = perm.from_port == '-1' || perm.from_port == -1 ? 'All' :
 perm.from_port.to_s
  to_port_str = perm.to_port == '-1' || perm.to_port == -1 ? 'All' :
 perm.to_port.to_s
  { from_port: from_port_str, to_port: to_port_str }
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_group_permissions(perm)
  ports = format_port_information(perm)
```

```
  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"
  print ", From: #{ports[:from_port]}, To: #{ports[:to_port]}"

  print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}" if perm.key?
(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?

  print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}" if perm.key?(:ip_ranges) &&
 perm.ip_ranges.count.positive?
  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      display_group_details(sg)
    end
  else
    puts 'No security groups found.'
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Helper method to display the details of security groups
def display_group_details(sg)
  puts '-' * (sg.group_name.length + 13)
  puts "Name:        #{sg.group_name}"
  puts "Description: #{sg.description}"
  puts "Group ID:    #{sg.group_id}"
  puts "Owner ID:    #{sg.owner_id}"
  puts "VPC ID:      #{sg.vpc_id}"

  display_group_tags(sg.tags) if sg.tags.count.positive?
  display_group_permissions(sg)
end

def display_group_tags(tags)
  puts 'Tags:'
  tags.each do |tag|
    puts "  Key: #{tag.key}, Value: #{tag.value}"
```

```
  end
end

def display_group_permissions(sg)
  if sg.ip_permissions.count.positive?
    puts 'Inbound rules:'
    sg.ip_permissions.each do |p|
      describe_security_group_permissions(p)
    end
  end

  return if sg.ip_permissions_egress.empty?

  puts 'Outbound rules:'
  sg.ip_permissions_egress.each do |p|
    describe_security_group_permissions(p)
  end
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  false
end

# Example usage with refactored run_me to reduce complexity
def run_me
  group_name, description, vpc_id, ip_protocol_http, from_port_http, to_port_http, \
  cidr_ip_range_http, ip_protocol_ssh, from_port_ssh, to_port_ssh, \
  cidr_ip_range_ssh, region = process_arguments
  ec2_client = Aws::EC2::Client.new(region: region)

  security_group_id = attempt_create_security_group(ec2_client, group_name,
 description, vpc_id)
  security_group_exists = security_group_id != 'Error'

  if security_group_exists
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_http,
 from_port_http, to_port_http, cidr_ip_range_http)
```

```
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_ssh, from_port_ssh,
 to_port_ssh, cidr_ip_range_ssh)
  end

  describe_security_groups(ec2_client)
  attempt_delete_security_group(ec2_client, security_group_id) if
 security_group_exists
end

def process_arguments
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    display_help
    exit 1
  elsif ARGV.count.zero?
    default_values
  else
    ARGV
  end
end

def attempt_create_security_group(ec2_client, group_name, description, vpc_id)
  puts 'Attempting to create security group...'
  security_group_id = create_security_group(ec2_client, group_name, description,
 vpc_id)
  puts 'Could not create security group. Skipping this step.' if security_group_id
 == 'Error'
  security_group_id
end

def add_inbound_rules(ec2_client, security_group_id, ip_protocol, from_port,
 to_port, cidr_ip_range)
  puts 'Attempting to add inbound rules to security group...'
  return if security_group_ingress_authorized?(ec2_client, security_group_id,
 ip_protocol, from_port, to_port,
                                               cidr_ip_range)

  puts 'Could not add inbound rule to security group. Skipping this step.'
end

def attempt_delete_security_group(ec2_client, security_group_id)
  puts "\nAttempting to delete security group..."
  return if security_group_deleted?(ec2_client, security_group_id)

  puts 'Could not delete security group. You must delete it yourself.'
```

```
end

def display_help
  puts 'Usage:   ruby ec2-ruby-example-security-group.rb ' \
    'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
    'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
    'CIDR_IP_RANGE_2 REGION'
  puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
    "my-security-group 'This is my security group.' vpc-6713dfEX " \
    "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
end

def default_values
  [
    'my-security-group', 'This is my security group.', 'vpc-6713dfEX', 'tcp', '80',
 '80',
    '0.0.0.0/0', 'tcp', '22', '22', '0.0.0.0/0', 'us-west-2'
  ]
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateSecurityGroup em Referência AWS SDK para Ruby da API.

`CreateSubnet`

O código de exemplo a seguir mostra como usar `CreateSubnet`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'
```

```
# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
```

```
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

# Example usage:
def run_me
  vpc_id = ''
  cidr_block = ''
  availability_zone = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-subnet.rb ' \
      'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
      'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    cidr_block = '10.0.0.0/24'
    availability_zone = 'us-west-2a'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
```

```
    tag_key = ARGV[3]
    tag_value = ARGV[4]
    region = ARGV[5]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if subnet_created_and_tagged?(
    ec2_resource,
    vpc_id,
    cidr_block,
    availability_zone,
    tag_key,
    tag_value
  )
    puts 'Subnet created and tagged.'
  else
    puts 'Subnet not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateSubnet em Referência AWS SDK para Ruby da API.

`CreateVpc`

O código de exemplo a seguir mostra como usar `CreateVpc`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'
```

```
# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''
```

```
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-vpc.rb ' \
      'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = '10.0.0.0/24'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,
    cidr_block,
    tag_key,
    tag_value
  )
    puts 'VPC created and tagged.'
  else
    puts 'VPC not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateVpc em Referência AWS SDK para Ruby da API.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
```

```
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte DescribeInstances em Referência AWS SDK para Ruby da API.

`DescribeRegions`

O código de exemplo a seguir mostra como usar `DescribeRegions`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print 'Region'
```

```
  print ' ' * (max_region_string_length - 'Region'.length)
  print "  Endpoint\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print ' ' * (max_region_string_length - region.region_name.length)
    print '  '
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print '  Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print "  State\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_zone_string_length
  print '  '
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
```

```
    print ' ' * (max_region_string_length - zone.region_name.length)
    print '  '
    print zone.zone_name
    print ' ' * (max_zone_string_length - zone.zone_name.length)
    print '  '
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts '  Messages for this zone:'
      zone.messages.each do |message|
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'AWS Regions for Amazon EC2 that are available to you:'
  list_regions_endpoints(ec2_client)
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS Region
 '#{region}':"
  list_availability_zones(ec2_client)
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte DescribeRegions em Referência AWS SDK para Ruby da API.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
```

```
  false
end
```

- Para obter detalhes da API, consulte ReleaseAddress em Referência AWS SDK para Ruby da API.

StartInstances

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Attempts to start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was started; otherwise, false.
# @example
#   exit 1 unless instance_started?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
```

```
    case state
    when 'pending'
      puts 'Error starting instance: the instance is pending. Try again later.'
      return false
    when 'running'
      puts 'The instance is already running.'
      return true
    when 'terminated'
      puts 'Error starting instance: ' \
        'the instance is terminated, so you cannot start it.'
      return false
    end
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance started.'
  true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
```

```
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to start instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_started?(ec2_client, instance_id)

  puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte StartInstances em Referência AWS SDK para Ruby da API.

StopInstances

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
```

```
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'stopping'
      puts 'The instance is already stopping.'
      return true
    when 'stopped'
      puts 'The instance is already stopped.'
      return true
    when 'terminated'
      puts 'Error stopping instance: ' \
        'the instance is terminated, so you cannot stop it.'
      return false
    end
  end

  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
```

```
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_stopped?(ec2_client, instance_id)

  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte StopInstances em Referência AWS SDK para Ruby da API.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ec2'

# Prerequisites:
```

```
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
     response.instance_statuses[0].instance_state.name == 'terminated'

    puts 'The instance is already terminated.'
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
  true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
```

```
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_terminated?(ec2_client, instance_id)

  puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte TerminateInstances em Referência AWS SDK para Ruby da API.

Exemplos do Elastic Beanstalk usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Elastic Beanstalk.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DescribeApplications`

O código de exemplo a seguir mostra como usar `DescribeApplications`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Class to manage Elastic Beanstalk applications
class ElasticBeanstalkManager
  def initialize(eb_client, logger: Logger.new($stdout))
    @eb_client = eb_client
    @logger = logger
  end

  # Lists applications and their environments
  def list_applications
    @eb_client.describe_applications.applications.each do |application|
      log_application_details(application)
      list_environments(application.application_name)
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Elastic Beanstalk Service Error: #{e.message}")
  end

  private

  # Logs application details
  def log_application_details(application)
    @logger.info("Name:        #{application.application_name}")
    @logger.info("Description: #{application.description}")
  end

  # Lists and logs details of environments for a given application
  def list_environments(application_name)
    @eb_client.describe_environments(application_name:
 application_name).environments.each do |env|
```

```
      @logger.info("  Environment:  #{env.environment_name}")
      @logger.info("    URL:        #{env.cname}")
      @logger.info("    Health:     #{env.health}")
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Error listing environments for application #{application_name}:
 #{e.message}")
  end
end
```

- Para obter detalhes da API, consulte DescribeApplications em Referência AWS SDK para Ruby da API.

`ListAvailableSolutionStacks`

O código de exemplo a seguir mostra como usar `ListAvailableSolutionStacks`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Manages listing of AWS Elastic Beanstalk solution stacks
# @param [Aws::ElasticBeanstalk::Client] eb_client
# @param [String] filter - Returns subset of results based on match
# @param [Logger] logger
class StackLister
  # Initialize with AWS Elastic Beanstalk client
  def initialize(eb_client, filter, logger: Logger.new($stdout))
    @eb_client = eb_client
    @filter = filter.downcase
    @logger = logger
  end

  # Lists and logs Elastic Beanstalk solution stacks
  def list_stacks
    stacks = @eb_client.list_available_solution_stacks.solution_stacks
```

```
    orig_length = stacks.length
    filtered_length = 0

    stacks.each do |stack|
      if @filter.empty? || stack.downcase.include?(@filter)
        @logger.info(stack)
        filtered_length += 1
      end
    end

    log_summary(filtered_length, orig_length)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Error listing solution stacks: #{e.message}")
  end

  private

  # Logs summary of listed stacks
  def log_summary(filtered_length, orig_length)
    if @filter.empty?
      @logger.info("Showed #{orig_length} stack(s)")
    else
      @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
    end
  end
end
```

- Para obter detalhes da API, consulte ListAvailableSolutionStacks em Referência AWS SDK para Ruby da API.

`UpdateApplication`

O código de exemplo a seguir mostra como usar `UpdateApplication`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Manages deployment of Rails applications to AWS Elastic Beanstalk
class RailsAppDeployer
  def initialize(eb_client, s3_client, app_name, logger: Logger.new($stdout))
    @eb_client = eb_client
    @s3_client = s3_client
    @app_name = app_name
    @logger = logger
  end

  # Deploys the latest application version to Elastic Beanstalk
  def deploy
    create_storage_location
    zip_file_name = create_zip_file
    upload_zip_to_s3(zip_file_name)
    create_and_deploy_new_application_version(zip_file_name)
  end

  private

  # Creates a new S3 storage location for the application
  def create_storage_location
    resp = @eb_client.create_storage_location
    @logger.info("Created storage location in bucket #{resp.s3_bucket}")
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create storage location: #{e.message}")
  end

  # Creates a ZIP file of the application using git
  def create_zip_file
    zip_file_basename = SecureRandom.urlsafe_base64
    zip_file_name = "#{zip_file_basename}.zip"
    `git archive --format=zip -o #{zip_file_name} HEAD`
    zip_file_name
  end

  # Uploads the ZIP file to the S3 bucket
  def upload_zip_to_s3(zip_file_name)
    zip_contents = File.read(zip_file_name)
    key = "#{@app_name}/#{zip_file_name}"
    @s3_client.put_object(body: zip_contents, bucket: fetch_bucket_name, key: key)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to upload ZIP file to S3: #{e.message}")
  end
```

```
  # Fetches the S3 bucket name from Elastic Beanstalk application versions
  def fetch_bucket_name
    app_versions = @eb_client.describe_application_versions(application_name:
@app_name)
    av = app_versions.application_versions.first
    av.source_bundle.s3_bucket
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch bucket name: #{e.message}")
    raise
  end

  # Creates a new application version and deploys it
  def create_and_deploy_new_application_version(zip_file_name)
    version_label = File.basename(zip_file_name, '.zip')
    @eb_client.create_application_version(
      process: false,
      application_name: @app_name,
      version_label: version_label,
      source_bundle: {
        s3_bucket: fetch_bucket_name,
        s3_key: "#{@app_name}/#{zip_file_name}"
      },
      description: "Updated #{Time.now.strftime('%d/%m/%Y')}"
    )
    update_environment(version_label)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create or deploy application version: #{e.message}")
  end

  # Updates the environment to the new application version
  def update_environment(version_label)
    env_name = fetch_environment_name
    @eb_client.update_environment(
      environment_name: env_name,
      version_label: version_label
    )
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to update environment: #{e.message}")
  end

  # Fetches the environment name of the application
  def fetch_environment_name
    envs = @eb_client.describe_environments(application_name: @app_name)
```

```
    envs.environments.first.environment_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch environment name: #{e.message}")
    raise
  end
end
```

- Para obter detalhes da API, consulte UpdateApplication em Referência AWS SDK para Ruby da API.

EventBridge exemplos de uso do SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby with EventBridge.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar e acionar uma regra

O exemplo de código a seguir mostra como criar e acionar uma regra na Amazon EventBridge.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Chame as funções na ordem correta.

```
require 'aws-sdk-sns'
require 'aws-sdk-iam'
require 'aws-sdk-cloudwatchevents'
require 'aws-sdk-ec2'
require 'aws-sdk-cloudwatch'
require 'aws-sdk-cloudwatchlogs'
require 'securerandom'
```

Verifica se o tópico do Amazon Simple Notification Service (Amazon SNS) existe dentre aqueles fornecidos para essa função.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  false
end
```

Verifica se o tópico especificado existe dentre aqueles disponíveis para o chamador no Amazon SNS.

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
```

```
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts 'Topic found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.topics.count.positive?

      if topic_found?(response.topics, topic_arn)
        puts 'Topic found.'
        return true
      end
    end
  end
  puts 'Topic not found.'
  false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  false
end
```

Crie um tópico no Amazon SNS e, em seguida, assine um endereço de e-mail para receber notificações sobre esse tópico.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
```

```
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: 'email',
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts 'Subscription created with ARN ' \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    'and confirm the subscription to start receiving notification emails.'
  topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  'Error'
end
```

Verifique se a função especificada AWS Identity and Access Management (IAM) existe entre as fornecidas para essa função.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
```

```
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  false
end
```

Verificar se o perfil especificado existe dentre aqueles disponíveis para o chamador no IAM.

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts 'Role found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.roles.count.positive?

      if role_found?(response.roles, role_arn)
        puts 'Role found.'
        return true
      end
    end
  end
```

```
  puts 'Role not found.'
  false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  false
end
```

Criar um perfil do IAM.

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'
          },
          'Action': 'sts:AssumeRole'
        }
      ]
    }.to_json,
    path: '/',
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
```

```
  puts 'Adding access policy to role...'
  iam_client.put_role_policy(
    policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': 'CloudWatchEventsFullAccess',
          'Effect': 'Allow',
          'Resource': '*',
          'Action': 'events:*'
        },
        {
          'Sid': 'IAMPassRoleForCloudWatchEvents',
          'Effect': 'Allow',
          'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
          'Action': 'iam:PassRole'
        }
      ]
    }.to_json,
    policy_name: 'CloudWatchEventsPolicy',
    role_name: role_name
  )
  puts 'Access policy added to role.'
  response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts 'If the role was created, you must add the access policy ' \
    'to the role yourself, or delete the role yourself and try again.'
  'Error'
end
```

Verifica se a EventBridge regra especificada existe entre as fornecidas para essa função.

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
```

```
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  false
end
```

Verifica se a regra especificada existe entre as disponíveis para o chamador. EventBridge

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts 'Rule found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.rules.count.positive?

      if rule_found?(response.rules, rule_name)
        puts 'Rule found.'
        return true
      end
    end
```

```
  end
  puts 'Rule not found.'
  false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  false
end
```

Crie uma regra em EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
```

```
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        'aws.ec2'
      ],
      'detail-type': [
        'EC2 Instance State-change Notification'
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: 'ENABLED',
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
     put_targets_response.failed_entry_count.positive?
    puts 'Error(s) adding target to rule:'
    put_targets_response.failed_entries.each do |failure|
```

```
      puts failure.error_message
    end
    false
  else
    true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts 'If the rule was created, you must add the target ' \
    'to the rule yourself, or delete the rule yourself and try again.'
  false
end
```

Verifique se o grupo de logs especificado existe entre aqueles disponíveis para o chamador no Amazon CloudWatch Logs.

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts 'Log group found.'
        return true
      end
    end
  end
  puts 'Log group not found.'
```

```
  false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  false
end
```

Crie um grupo de CloudWatch registros em Registros.

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts 'Log group created.'
  true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  false
end
```

Grave um evento em um stream de log em CloudWatch Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
```

```
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '49542672486831074009579604567656788214806863282469607 3EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  event[:sequence_token] = sequence_token unless sequence_token.empty?

  response = cloudwatchlogs_client.put_log_events(event)
  puts 'Message logged.'
  response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end
```

Reinicie uma instância do Amazon Elastic Compute Cloud (Amazon EC2) e adicione informações sobre a atividade relacionada a um fluxo de log no CloudWatch Logs.

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ''
```

```
  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    'This might take a few minutes...'
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts 'Attempting to restart the instance. This might take a few minutes...'
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance restarted.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  true
rescue StandardError => e
  puts 'Error creating log stream or stopping or restarting the instance: ' \
    "#{e.message}"
  log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  false
end
```

Exibir informações sobre a atividade de uma regra em EventBridge.

```
# Displays information about activity for a rule in Amazon EventBridge.
```

```
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts 'Attempting to display rule activity...'
  response = cloudwatch_client.get_metric_statistics(
    namespace: 'AWS/Events',
    metric_name: 'Invocations',
    dimensions: [
      {
        name: 'RuleName',
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ['Sum'],
    unit: 'Count'
  )
```

```
  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      'specified time period.'
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Exiba as informações de registro de todos os fluxos de registro em um grupo de CloudWatch registros de registros.

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts 'Attempting to display log stream data for the log group ' \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: 'LastEventTime',
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
     describe_log_streams_response.log_streams.count.positive?
```

```
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts '-' * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
         get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts 'No log messages for this log stream.'
      end
    end
  end
rescue StandardError => e
  puts 'Error getting information about the log streams or their messages: ' \
    "#{e.message}"
end
```

Exiba um lembrete para que o chamador limpe manualmente todos AWS os recursos associados de que ele não precise mais.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
```

```
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts '-' * 10
  puts 'Some of the following AWS resources might still exist in your account.'
  puts 'If you no longer want to use this code example, then to clean up'
  puts 'your AWS account and avoid unexpected costs, you might want to'
  puts 'manually delete any of the following resources if they exist:'
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - PutEvents
  - PutRule

AWS Glue exemplos de uso do SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby with AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-glue'
require 'logger'

# GlueManager is a class responsible for managing AWS Glue operations
# such as listing all Glue jobs in the current AWS account.
class GlueManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all Glue jobs in the current AWS account.
  def list_jobs
    @logger.info('Here are the Glue jobs in your account:')

    paginator = @client.get_jobs(max_results: 10)
    jobs = []

    paginator.each_page do |page|
      jobs.concat(page.jobs)
    end

    if jobs.empty?
      @logger.info("You don't have any Glue jobs.")
    else
      jobs.each do |job|
        @logger.info("- #{job.name}")
```

```
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  glue_client = Aws::Glue::Client.new
  manager = GlueManager.new(glue_client)
  manager.list_jobs
end
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para Ruby da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe que agrupe as AWS Glue funções usadas no cenário.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil if
 not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
 crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
```

```
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
```

```
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
```

```
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
```

```
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
file to.
  # @return [void]
  def upload_job_script(file_path, bucket_resource)
```

```
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
                                          body: file,
                                          bucket: bucket_resource.name,
                                          key: file_path
                                        })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end
end
```

Crie uma classe que execute o cenário.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
  end

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
  end

  private

  def setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    new_step(1, 'Create a crawler')
    crawler = wrapper.get_crawler(crawler_name)
    unless crawler
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
 db_prefix, data_source)
      puts "Successfully created #{crawler_name}."
    end
    wrapper.start_crawler(crawler_name)
```

```
    monitor_crawler(wrapper, crawler_name)
  end

  def monitor_crawler(wrapper, crawler_name)
    new_step(2, 'Monitor Crawler')
    crawler_state = nil
    until crawler_state == 'READY'
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]['state']
      print "Crawler status: #{crawler_state}".yellow
    end
  end

  def query_database(wrapper, _crawler_name, db_name)
    new_step(3, 'Query the database.')
    wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    puts "Database contains tables: #{wrapper.get_tables(db_name).map { |t|
 t['name'] }}"
  end

  def create_and_run_job(wrapper, job_script, job_name, db_name)
    new_step(4, 'Create and run job.')
    wrapper.upload_job_script(job_script, @glue_bucket)
    wrapper.create_job(job_name, 'ETL Job', @glue_service_role.arn, "s3://
#{@glue_bucket.name}/#{job_script}")
    run_job(wrapper, job_name, db_name)
  end

  def run_job(wrapper, job_name, db_name)
    new_step(5, 'Run the job.')
    wrapper.start_job_run(job_name, db_name, wrapper.get_tables(db_name)[0]['name'],
 @glue_bucket.name)
    job_run_status = nil
    until %w[SUCCEEDED FAILED STOPPED].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]['job_run_state']
      print "Job #{job_name} status: #{job_run_status}".yellow
    end
  end
end
```

```
def main
  banner('../../helpers/banner.txt')
  puts 'Starting AWS Glue demo...'

  # Load resource names from YAML.
  resource_names = YAML.load_file('resource_names.yaml')

  # Setup services and resources.
  iam_role = Aws::IAM::Resource.new(region: 'us-
east-1').role(resource_names['glue_service_role'])
  s3_bucket = Aws::S3::Resource.new(region: 'us-
east-1').bucket(resource_names['glue_bucket'])

  # Instantiate scenario and run.
  scenario = GlueCrawlerJobScenario.new(Aws::Glue::Client.new(region: 'us-east-1'),
 iam_role, s3_bucket, @logger)
  random_suffix = rand(10**4)
  scenario.run("crawler-#{random_suffix}", "db-#{random_suffix}", "prefix-
#{random_suffix}-", 's3://data_source',
               'job_script.py', "job-#{random_suffix}")

  puts 'Demo complete.'
end
```

Crie um script ETL que seja usado pelo AWS Glue para extrair, transformar e carregar dados durante a execução do trabalho.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the data
 to
```

```
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)
```

```
# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
 crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
```

```
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte CreateCrawler em Referência AWS SDK para Ruby da API.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
```

```
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte CreateJob em Referência AWS SDK para Ruby da API.

`DeleteCrawler`

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
```

```
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteCrawler em Referência AWS SDK para Ruby da API.

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
```

```
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte DeleteDatabase em Referência AWS SDK para Ruby da API.

`DeleteJob`

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
```

```
    @logger = logger
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte DeleteJob em Referência AWS SDK para Ruby da API.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end
```

```
  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte DeleteTable em Referência AWS SDK para Ruby da API.

`GetCrawler`

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end
```

```
  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil if
 not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte GetCrawler em Referência AWS SDK para Ruby da API.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
```

```
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
 if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte GetDatabase em Referência AWS SDK para Ruby da API.

`GetJobRun`

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
```

```
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte GetJobRun em Referência AWS SDK para Ruby da API.

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end
```

```
  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Para obter detalhes da API, consulte GetJobRuns em Referência AWS SDK para Ruby da API.

`GetTables`

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
```

```
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte GetTables em Referência AWS SDK para Ruby da API.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of jobs in AWS Glue.
  #
```

```
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte ListJobs em Referência AWS SDK para Ruby da API.

StartCrawler

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
```

```
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte StartCrawler em Referência AWS SDK para Ruby da API.

`StartJobRun`

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
```

```
# @return [String] The ID of the started job run.
def start_job_run(name, input_database, input_table, output_bucket_name)
  response = @glue_client.start_job_run(
    job_name: name,
    arguments: {
      '--input_database': input_database,
      '--input_table': input_table,
      '--output_bucket_url': "s3://#{output_bucket_name}/"
    }
  )
  response.job_run_id
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not start job run #{name}: \n#{e.message}")
  raise
end
```

- Para obter detalhes da API, consulte StartJobRun em Referência AWS SDK para Ruby da API.

Exemplos do IAM usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-iam'
require 'logger'

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')

    paginator = @client.list_policies
    policies = []

    paginator.each_page do |page|
      policies.concat(page.policies)
    end

    if policies.empty?
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
      end
```

```
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para Ruby da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para Ruby

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um usuário e um perfil do IAM que conceda permissão para listar os buckets do Amazon S3. O usuário só tem direitos para assumir a função. Após assumir a função, use credenciais temporárias para listar os buckets para a conta.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts('Give AWS time to propagate resources...')
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info('Tried and failed to create demo user.')
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
```

```
    user_key = @iam_client.create_access_key(user_name: user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Principal: { 'AWS': user.arn },
        Action: 'sts:AssumeRole'
      }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account, and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
```

```
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 's3:ListAllMyBuckets',
        Resource: 'arn:aws:s3:::*'
      }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role #{role.role_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 'sts:AssumeRole',
        Resource: role.arn
      }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
```

```
      policy_document: policy_document
    )
    puts("Created an inline policy for #{user.user_name} that lets the user assume
role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an Amazon S3 resource with specified credentials. This is separated into
a
  # factory function so that it can be mocked for unit testing.
  #
  # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
  def create_s3_resource(credentials)
    Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
  end

  # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
  # Because the resource uses credentials with limited access, it may not be able to
  # list the S3 buckets.
  #
  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def list_buckets(s3_resource)
    count = 10
    s3_resource.buckets.each do |bucket|
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == 'AccessDenied'
      puts('Attempt to list buckets with no permissions: AccessDenied.')
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end
```

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: 'create-use-assume-role-scenario'
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end

 # Deletes a role. If the role has policies attached, they are detached and
 # deleted before the role is deleted.
 #
 # @param role_name [String] The name of the role to delete.
 def delete_role(role_name)
   @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
     @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
     @iam_client.delete_policy(policy_arn: policy.policy_arn)
     @logger.info("Detached and deleted policy #{policy.policy_name}.")
   end
   @iam_client.delete_role({ role_name: role_name })
   @logger.info("Role deleted: #{role_name}.")
```

```
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
 why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
 user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
 role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
```

```
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy

- PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para Ruby

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
```

```
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte AttachRolePolicy em Referência AWS SDK para Ruby da API.

`AttachUserPolicy`

O código de exemplo a seguir mostra como usar `AttachUserPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Attaches a policy to a user
  #
  # @param user_name [String] The name of the user
```

```
  # @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_user(user_name, policy_arn)
    @iam_client.attach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
  end
```

- Para obter detalhes da API, consulte AttachUserPolicy em Referência AWS SDK para Ruby da API.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
```

```
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
```

```
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte CreateAccessKey em Referência AWS SDK para Ruby da API.

`CreateAccountAlias`

O código de exemplo a seguir mostra como usar `CreateAccountAlias`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, criar e excluir aliases da conta.

```
class IAMAliasManager
```

```
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
#{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
```

```
    false
  end
end
```

- Para obter detalhes da API, consulte CreateAccountAlias em Referência AWS SDK para Ruby da API.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
```

```
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
```

```
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte CreatePolicy em Referência AWS SDK para Ruby da API.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Creates a role and attaches policies to it.
 #
 # @param role_name [String] The name of the role.
 # @param assume_role_policy_document [Hash] The trust relationship policy
document.
 # @param policy_arns [Array<String>] The ARNs of the policies to attach.
 # @return [String, nil] The ARN of the new role if successful, or nil if an error
occurred.
 def create_role(role_name, assume_role_policy_document, policy_arns)
   response = @iam_client.create_role(
     role_name: role_name,
     assume_role_policy_document: assume_role_policy_document.to_json
   )
   role_arn = response.role.arn

   policy_arns.each do |policy_arn|
     @iam_client.attach_role_policy(
       role_name: role_name,
       policy_arn: policy_arn
     )
   end

   role_arn
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error creating role: #{e.message}")
   nil
 end
```

- Para obter detalhes da API, consulte CreateRole em Referência AWS SDK para Ruby da API.

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Creates a service-linked role
  #
  # @param service_name [String] The service name to create the role for.
  # @param description [String] The description of the service-linked role.
  # @param suffix [String] Suffix for customizing role name.
  # @return [String] The name of the created role
  def create_service_linked_role(service_name, description, suffix)
    response = @iam_client.create_service_linked_role(
      aws_service_name: service_name, description: description, custom_suffix:
suffix
    )
    role_name = response.role.role_name
    @logger.info("Created service-linked role #{role_name}.")
    role_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't create service-linked role for #{service_name}. Here's
why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte CreateServiceLinkedRole em Referência AWS SDK para Ruby da API.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Creates a user and their login profile
  #
  # @param user_name [String] The name of the user
  # @param initial_password [String] The initial password for the user
```

```
  # @return [String, nil] The ID of the user if created, or nil if an error occurred
  def create_user(user_name, initial_password)
    response = @iam_client.create_user(user_name: user_name)
    @iam_client.wait_until(:user_exists, user_name: user_name)
    @iam_client.create_login_profile(
      user_name: user_name,
      password: initial_password,
      password_reset_required: true
    )
    @logger.info("User '#{user_name}' created successfully.")
    response.user.user_id
  rescue Aws::IAM::Errors::EntityAlreadyExists
    @logger.error("Error creating user '#{user_name}': user already exists.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating user '#{user_name}': #{e.message}")
    nil
  end
```

- Para obter detalhes da API, consulte CreateUser em Referência AWS SDK para Ruby da API.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
```

```
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
```

```
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte DeleteAccessKey em Referência AWS SDK para Ruby da API.

`DeleteAccountAlias`

O código de exemplo a seguir mostra como usar `DeleteAccountAlias`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, criar e excluir aliases da conta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
```

```
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte DeleteAccountAlias em Referência AWS SDK para Ruby da API.

`DeleteRole`

O código de exemplo a seguir mostra como usar `DeleteRole`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Deletes a role and its attached policies.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    # Detach and delete attached policies
    @iam_client.list_attached_role_policies(role_name: role_name).each do |response|
      response.attached_policies.each do |policy|
        @iam_client.detach_role_policy({
                                          role_name: role_name,
```

```
                                        policy_arn: policy.policy_arn
                                      })
        # Check if the policy is a customer managed policy (not AWS managed)
        unless policy.policy_arn.include?('aws:policy/')
          @iam_client.delete_policy({ policy_arn: policy.policy_arn })
          @logger.info("Deleted customer managed policy #{policy.policy_name}.")
        end
      end
    end

    # Delete the role
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Deleted role #{role_name}.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteRole em Referência AWS SDK para Ruby da API.

`DeleteServerCertificate`

O código de exemplo a seguir mostra como usar `DeleteServerCertificate`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, atualizar e excluir certificados de servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
```

```
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
```

```
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte DeleteServerCertificate em Referência AWS SDK para Ruby da API.

`DeleteServiceLinkedRole`

O código de exemplo a seguir mostra como usar `DeleteServiceLinkedRole`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Deletes a service-linked role.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_service_linked_role(role_name)
    response = @iam_client.delete_service_linked_role(role_name: role_name)
    task_id = response.deletion_task_id
    check_deletion_status(role_name, task_id)
  rescue Aws::Errors::ServiceError => e
```

```
    handle_deletion_error(e, role_name)
  end

  private

  # Checks the deletion status of a service-linked role
  #
  # @param role_name [String] The name of the role being deleted
  # @param task_id [String] The task ID for the deletion process
  def check_deletion_status(role_name, task_id)
    loop do
      response = @iam_client.get_service_linked_role_deletion_status(
        deletion_task_id: task_id
      )
      status = response.status
      @logger.info("Deletion of #{role_name} #{status}.")
      break if %w[SUCCEEDED FAILED].include?(status)

      sleep(3)
    end
  end

  # Handles deletion error
  #
  # @param e [Aws::Errors::ServiceError] The error encountered during deletion
  # @param role_name [String] The name of the role attempted to delete
  def handle_deletion_error(e, role_name)
    return if e.code == 'NoSuchEntity'

    @logger.error("Couldn't delete #{role_name}. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteServiceLinkedRole em Referência AWS SDK para Ruby da API.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
 end
```

- Para obter detalhes da API, consulte DeleteUser em Referência AWS SDK para Ruby da API.

`DeleteUserPolicy`

O código de exemplo a seguir mostra como usar `DeleteUserPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Deletes a user and their associated resources
  #
  # @param user_name [String] The name of the user to delete
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
 user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
```

- Para obter detalhes da API, consulte DeleteUserPolicy em Referência AWS SDK para Ruby da API.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
```

```
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte DetachRolePolicy em Referência AWS SDK para Ruby da API.

`DetachUserPolicy`

O código de exemplo a seguir mostra como usar `DetachUserPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Detaches a policy from a user
#
# @param user_name [String] The name of the user
# @param policy_arn [String] The ARN of the policy to detach
# @return [Boolean] true if the policy was successfully detached, false otherwise
def detach_user_policy(user_name, policy_arn)
  @iam_client.detach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
  @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
successfully.")
  true
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error('Error detaching policy: Policy or user does not exist.')
  false
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error detaching policy from user '#{user_name}': #{e.message}")
  false
end
```

- Para obter detalhes da API, consulte DetachUserPolicy em Referência AWS SDK para Ruby da API.

`GetAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'IAMPolicyManager'
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
    response = @iam_client.get_account_password_policy
    @logger.info("The account password policy is: #{response.password_policy.to_h}")
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.info('The account does not have a password policy.')
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't print the account password policy. Error: #{e.code} -
 #{e.message}")
    raise
  end
end
```

- Para obter detalhes da API, consulte GetAccountPasswordPolicy em Referência AWS SDK para Ruby da API.

GetPolicy

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Fetches an IAM policy by its ARN
 # @param policy_arn [String] the ARN of the IAM policy to retrieve
 # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
 def get_policy(policy_arn)
   response = @iam_client.get_policy(policy_arn: policy_arn)
   policy = response.policy
   @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
   policy
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
   raise
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
   raise
 end
```

- Para obter detalhes da API, consulte GetPolicy em Referência AWS SDK para Ruby da API.

GetRole

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                              role_name: name
                            }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- Para obter detalhes da API, consulte GetRole em Referência AWS SDK para Ruby da API.

`GetUser`

O código de exemplo a seguir mostra como usar `GetUser`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Retrieves a user's details
  #
  # @param user_name [String] The name of the user to retrieve
  # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
 error occurred
  def get_user(user_name)
    response = @iam_client.get_user(user_name: user_name)
    response.user
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("User '#{user_name}' not found.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error retrieving user '#{user_name}': #{e.message}")
    nil
  end
```

- Para obter detalhes da API, consulte GetUser em Referência AWS SDK para Ruby da API.

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end
```

```
  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
```

```
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListAccessKeys em Referência AWS SDK para Ruby da API.

`ListAccountAliases`

O código de exemplo a seguir mostra como usar `ListAccountAliases`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, criar e excluir aliases da conta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
```

```
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListAccountAliases em Referência AWS SDK para Ruby da API.

`ListAttachedRolePolicies`

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
```

```
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end
```

```
  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListAttachedRolePolicies em Referência AWS SDK para Ruby da API.

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Para obter detalhes da API, consulte ListGroups em Referência AWS SDK para Ruby da API.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
```

```
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListPolicies em Referência AWS SDK para Ruby da API.

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

- Para obter detalhes da API, consulte ListRolePolicies em Referência AWS SDK para Ruby da API.

ListRoles

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count

      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end

  role_names
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't list roles for the account. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- Para obter detalhes da API, consulte ListRoles em Referência AWS SDK para Ruby da API.

Listar provedores de SAML

O código de exemplo a seguir mostra como usar `ListSAMLProviders`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Consulte detalhes da API em ListSAMLProvider na Referência da API AWS SDK para Ruby .

`ListServerCertificates`

O código de exemplo a seguir mostra como usar `ListServerCertificates`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, atualizar e excluir certificados de servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
```

```
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte ListServerCertificates em Referência AWS SDK para Ruby da API.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing users: #{e.message}")
  []
end
```

- Para obter detalhes da API, consulte ListUsers em Referência AWS SDK para Ruby da API.

`PutUserPolicy`

O código de exemplo a seguir mostra como usar `PutUserPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Creates an inline policy for a specified user.
# @param username [String] The name of the IAM user.
# @param policy_name [String] The name of the policy to create.
# @param policy_document [String] The JSON policy document.
# @return [Boolean]
def create_user_policy(username, policy_name, policy_document)
  @iam_client.put_user_policy({
                                user_name: username,
                                policy_name: policy_name,
                                policy_document: policy_document
                              })
  @logger.info("Policy #{policy_name} created for user #{username}.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't create policy #{policy_name} for user #{username}.
Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  false
end
```

- Para obter detalhes da API, consulte PutUserPolicy em Referência AWS SDK para Ruby da API.

`UpdateServerCertificate`

O código de exemplo a seguir mostra como usar `UpdateServerCertificate`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar, atualizar e excluir certificados de servidor.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end
```

```
    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Para obter detalhes da API, consulte UpdateServerCertificate em Referência AWS SDK para Ruby da API.

`UpdateUser`

O código de exemplo a seguir mostra como usar `UpdateUser`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Updates an IAM user's name
#
# @param current_name [String] The current name of the user
# @param new_name [String] The new name of the user
def update_user_name(current_name, new_name)
  @iam_client.update_user(user_name: current_name, new_user_name: new_name)
  true
rescue StandardError => e
  @logger.error("Error updating user name from '#{current_name}' to '#{new_name}':
#{e.message}")
  false
end
```

- Para obter detalhes da API, consulte UpdateUser em Referência AWS SDK para Ruby da API.

Exemplos do Kinesis usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Kinesis.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
```

```
  return data
end
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de item em lote do Kinesis com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
```

```
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
end
```

AWS KMS exemplos de uso do SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby with AWS KMS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: 'CreatedBy',
                               tag_value: 'ExampleUser'
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

- Para obter detalhes da API, consulte CreateKey em Referência AWS SDK para Ruby da API.

`Decrypt`

O código de exemplo a seguir mostra como usar `Decrypt`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Decrypted blob

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596
blob_packed = [blob].pack('H*')

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts 'Raw text: '
puts resp.plaintext
```

- Consulte detalhes da API em Decrypt na Referência da API AWS SDK para Ruby .

`Encrypt`

O código de exemplo a seguir mostra como usar `Encrypt`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

text = '1234567890'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.encrypt({
                        key_id: keyId,
                        plaintext: text
                      })

# Display a readable version of the resulting encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Consulte detalhes da API em Encrypt na Referência da API AWS SDK para Ruby .

`ReEncrypt`

O código de exemplo a seguir mostra como usar `ReEncrypt`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.
```

```
blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596
sourceCiphertextBlob = [blob].pack('H*')

# Replace the fictitious key ARN with a valid key ID

destinationKeyId = 'arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.re_encrypt({
                           ciphertext_blob: sourceCiphertextBlob,
                           destination_key_id: destinationKeyId
                         })

# Display a readable version of the resulting re-encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Para obter detalhes da API, consulte ReEncrypt em Referência AWS SDK para Ruby da API.

Exemplos de Lambda usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Lambda

O exemplo de código a seguir mostra como começar a usar o Lambda.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-lambda'

# Creates an AWS Lambda client using the default credentials and configuration
def lambda_client
  Aws::Lambda::Client.new
end

# Lists the Lambda functions in your AWS account, paginating the results if
 necessary
def list_lambda_functions
  lambda = lambda_client

  # Use a pagination iterator to list all functions
  functions = []
  lambda.list_functions.each_page do |page|
    functions.concat(page.functions)
  end
```

```
  # Print the name and ARN of each function
  functions.each do |function|
    puts "Function name: #{function.function_name}"
    puts "Function ARN: #{function.function_arn}"
    puts
  end

  puts "Total functions: #{functions.count}"
end

list_lambda_functions if __FILE__ == $PROGRAM_NAME
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para Ruby da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Configure as permissões de pré-requisitos do IAM para uma função do Lambda capaz de gravar logs.

```
  # Get an AWS Identity and Access Management (IAM) role.
  #
  # @param iam_role_name: The name of the role to retrieve.
  # @param action: Whether to create or destroy the IAM apparatus.
  # @return: The IAM role.
  def manage_iam(iam_role_name, action)
    case action
    when 'create'
      create_iam_role(iam_role_name)
    when 'destroy'
      destroy_iam_role(iam_role_name)
    else
      raise "Incorrect action provided. Must provide 'create' or 'destroy'"
    end
  end

  private

  def create_iam_role(iam_role_name)
    role_policy = {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Effect': 'Allow',
          'Principal': { 'Service': 'lambda.amazonaws.com' },
          'Action': 'sts:AssumeRole'
        }
      ]
    }
    role = @iam_client.create_role(
      role_name: iam_role_name,
      assume_role_policy_document: role_policy.to_json
    )
    @iam_client.attach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
```

```
    wait_for_role_to_exist(iam_role_name)
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    sleep(10)
    [role, role_policy.to_json]
  end

  def destroy_iam_role(iam_role_name)
    @iam_client.detach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    @iam_client.delete_role(role_name: iam_role_name)
    @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
  end

  def wait_for_role_to_exist(iam_role_name)
    @iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  end
```

Defina um manipulador do Lambda que incremente um número fornecido como um parâmetro de invocação.

```
require 'logger'

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
# @param event [Hash] Parameters sent when the function is invoked
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
  logger = Logger.new($stdout)
  log_level = ENV['LOG_LEVEL']
  logger.level = case log_level
                 when 'debug'
```

```
                    Logger::DEBUG
                  when 'info'
                    Logger::INFO
                  else
                    Logger::ERROR
                  end
  logger.debug('This is a debug log message.')
  logger.info('This is an info log message. Code executed successfully!')
  number = event['number'].to_i
  incremented_number = number + 1
  logger.info("You provided #{number.round} and it was incremented to
 #{incremented_number.round}")
  incremented_number.round.to_s
end
```

Compacte a função do Lambda em um pacote de implantação.

```
  # Creates a Lambda deployment package in .zip format.
  #
  # @param source_file: The name of the object, without suffix, for the Lambda file
 and zip.
  # @return: The deployment package.
  def create_deployment_package(source_file)
    Dir.chdir(File.dirname(__FILE__))
    if File.exist?('lambda_function.zip')
      File.delete('lambda_function.zip')
      @logger.debug('Deleting old zip: lambda_function.zip')
    end
    Zip::File.open('lambda_function.zip', create: true) do |zipfile|
      zipfile.add('lambda_function.rb', "#{source_file}.rb")
    end
    @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
    File.read('lambda_function.zip').to_s
  rescue StandardError => e
    @logger.error("There was an error creating deployment package:\n #{e.message}")
  end
```

Crie uma nova função do Lambda.

```
  # Deploys a Lambda function.
  #
```

```
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name })
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Invoque a função do Lambda com parâmetros de runtime opcionais.

```
  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
```

```
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

Atualize a configuração da função do Lambda para injetar uma nova variável de ambiente.

```
  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                    function_name: function_name,
                                                    environment: {
                                                      variables: {
                                                        'LOG_LEVEL' => log_level
                                                      }
                                                    }
                                                  })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Atualize o código da função do Lambda com um pacote de implantação diferente que contenha um código diferente.

```
  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
```

```
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

Liste todas as funções do Lambda existentes usando o paginador integrado.

```
  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response['functions'].each do |function|
        functions.append(function['function_name'])
      end
    end
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error listing functions:\n #{e.message}")
  end
```

Exclua uma função do Lambda específica.

```
  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
```

```
    @lambda_client.delete_function(
      function_name: function_name
    )
    print 'Done!'.green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
```

```
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name })
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- Para obter detalhes da API, consulte CreateFunction em Referência AWS SDK para Ruby da API.

DeleteFunction

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name: function_name
    )
    print 'Done!'.green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
```

- Para obter detalhes da API, consulte DeleteFunction em Referência AWS SDK para Ruby da API.

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
```

- Para obter detalhes da API, consulte GetFunction em Referência AWS SDK para Ruby da API.

`Invocar`

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

- Consulte detalhes da API em Invoke, na Referência da API AWS SDK para Ruby .

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response['functions'].each do |function|
        functions.append(function['function_name'])
      end
    end
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error listing functions:\n #{e.message}")
  end
```

- Para obter detalhes da API, consulte ListFunctions em Referência AWS SDK para Ruby da API.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
```

```
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

- Para obter detalhes da API, consulte UpdateFunctionCode em Referência AWS SDK para Ruby da API.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                   function_name: function_name,
                                                   environment: {
                                                     variables: {
                                                       'LOG_LEVEL' => log_level
                                                     }
```

```
                                                    }
                                                  })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- Para obter detalhes da API, consulte UpdateFunctionConfiguration em Referência AWS SDK para Ruby da API.

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
```

```
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Kinesis com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do S3 com o Lambda usando Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de item em lote do Kinesis com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
```

```
end
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

Exemplos do Amazon MSK usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon MSK.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon MSK com o Lambda usando Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"
```

```
      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Exemplos do Amazon Polly usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon Polly.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DescribeVoices`

O código de exemplo a seguir mostra como usar `DescribeVoices`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts "  #{v.gender}"
    puts
  end
rescue StandardError => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- Para obter detalhes da API, consulte DescribeVoices em Referência AWS SDK para Ruby da API.

`ListLexicons`

O código de exemplo a seguir mostra como usar `ListLexicons`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'
```

```
begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:#{l.attributes.alphabet}"
    puts "  Language:#{l.attributes.language}"
    puts
  end
rescue StandardError => e
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts e.message
end
```

- Para obter detalhes da API, consulte ListLexicons em Referência AWS SDK para Ruby da API.

`SynthesizeSpeech`

O código de exemplo a seguir mostra como usar `SynthesizeSpeech`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
```

```
  # Get the filename from the command line
  if ARGV.empty?
    puts 'You must supply a filename'
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: #{filename}"
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
                                   output_format: 'mp3',
                                   text: contents,
                                   voice_id: 'Joanna'
                                 })

  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = "#{first_part}.mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: #{mp3_file}"
rescue StandardError => e
  puts 'Got error:'
  puts 'Error message:'
  puts e.message
```

```
end
```

- Para obter detalhes da API, consulte SynthesizeSpeech em Referência AWS SDK para Ruby da API.

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos de código para o Amazon RDS usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon RDS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon RDS

O exemplo de código a seguir mostra como começar a usar o Amazon RDS.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds'
require 'logger'

# RDSManager is a class responsible for managing RDS operations
# such as listing all RDS DB instances in the current AWS account.
class RDSManager
  def initialize(client)
```

```
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all RDS DB instances in the current AWS account.
  def list_db_instances
    @logger.info('Listing RDS DB instances')

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
 (#{instance.db_instance_status})")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- Para obter detalhes da API, consulte DescribeDBInstances na Referência da API AWS SDK para Ruby .

Ações

`CreateDBSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBSnapshot`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
                                db_snapshot_identifier: id
                              })
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- Consulte detalhes da API em CreateDBSnapshot na Referência da API AWS SDK para Ruby .

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
                          "name": i.id,
                          "status": i.db_instance_status
                        })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para Ruby .

`DescribeDBParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBParameterGroups`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
```

```
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Para obter detalhes da API, consulte DescribeDBParameterGroups em Referência AWS SDK para Ruby da API.

`DescribeDBParameters`

O código de exemplo a seguir mostra como usar `DescribeDBParameters`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
```

```
                              })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Consulte detalhes da API em DescribeDBParameters na Referência da API AWS SDK para Ruby .

`DescribeDBSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBSnapshots`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instance
# snapshots.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return instance_snapshots [Array, nil] All instance snapshots, or nil if error.
def list_instance_snapshots(rds_resource)
  instance_snapshots = []
  rds_resource.db_snapshots.each do |s|
    instance_snapshots.append({
                                "id": s.snapshot_id,
                                "status": s.status
                              })
  end
  instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
```

```
end
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência da API AWS SDK para Ruby .

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
```

```
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Exemplos de código do Amazon S3 usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
```

```
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para Ruby da API.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.

- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
```

```
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded file.
  def upload_file(bucket)
    File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
    s3_object = bucket.object(File.basename('demo.txt'))
    s3_object.upload_file('demo.txt')
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      puts('Enter a name for the downloaded file: ')
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
```

```
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
```

```
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME ==
 __FILE__
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects

- GetObject
- ListObjectsV2
- PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Copie um objeto.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
```

```
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copie um objeto e adicione criptografia do lado do servidor ao objeto de destino.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                       copy actions.
  def initialize(source_object)
    @source_object = source_object
  end
```

```
  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CopyObject em Referência AWS SDK para Ruby da API.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
```

```
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateBucket em Referência AWS SDK para Ruby da API.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
```

```
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Para obter detalhes da API, consulte DeleteBucket em Referência AWS SDK para Ruby da API.

`DeleteBucketCors`

O código de exemplo a seguir mostra como usar `DeleteBucketCors`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end
```

```
  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obter detalhes da API, consulte DeleteBucketCors em Referência AWS SDK para Ruby da API.

`DeleteBucketPolicy`

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
```

```
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obter detalhes da API, consulte DeleteBucketPolicy em Referência AWS SDK para Ruby da API.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
```

```
  end
```

- Para obter detalhes da API, consulte DeleteObjects em Referência AWS SDK para Ruby da API.

`GetBucketCors`

O código de exemplo a seguir mostra como usar `GetBucketCors`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
 for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end
```

```
end
```

- Para obter detalhes da API, consulte GetBucketCors em Referência AWS SDK para Ruby da API.

GetBucketPolicy

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end
```

```
end
```

- Para obter detalhes da API, consulte GetBucketPolicy em Referência AWS SDK para Ruby da API.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Obtenha um objeto.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Obtenha um objeto e relate seu estado de criptografia do lado do servidor.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte GetObject em Referência AWS SDK para Ruby da API.

`HeadObject`

O código de exemplo a seguir mostra como usar `HeadObject`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end
```

```
  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
 Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte HeadObject em Referência AWS SDK para Ruby da API.

`ListBuckets`

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'
```

```
# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte ListBuckets em Referência AWS SDK para Ruby da API.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why: #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
```

```
  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte ListObjects V2 em Referência de AWS SDK para Ruby API.

`PutBucketCors`

O código de exemplo a seguir mostra como usar `PutBucketCors`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
```

```
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obter detalhes da API, consulte PutBucketCors em Referência AWS SDK para Ruby da API.

`PutBucketPolicy`

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy
```

```
  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Para obter detalhes da API, consulte PutBucketPolicy em Referência AWS SDK para Ruby da API.

`PutBucketWebsite`

O código de exemplo a seguir mostra como usar `PutBucketWebsite`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website
```

```
  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte PutBucketWebsite em Referência AWS SDK para Ruby da API.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Faça upload de um arquivo usando um carregador gerenciado (Object.upload_file).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Faça upload de um arquivo usando Object.put.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
```

```
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Faça upload de um arquivo usando Object.put e adicione criptografia do lado do servidor.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"
```

```
  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte PutObject em Referência AWS SDK para Ruby da API.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
```

```
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:
 #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
 'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do S3 com o Lambda usando Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Exemplos do Amazon SES usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetIdentityVerificationAttributes`

O código de exemplo a seguir mostra como usar `GetIdentityVerificationAttributes`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                              identity_type: 'EmailAddress'
                            })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })
```

```
  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Para obter detalhes da API, consulte GetIdentityVerificationAttributes em Referência AWS SDK para Ruby da API.

`ListIdentities`

O código de exemplo a seguir mostra como usar `ListIdentities`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                              identity_type: 'EmailAddress'
                            })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status
```

```
  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Para obter detalhes da API, consulte ListIdentities em Referência AWS SDK para Ruby da API.

SendEmail

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
```

```
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )
```

```
  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- Para obter detalhes da API, consulte SendEmail em Referência AWS SDK para Ruby da API.

`VerifyEmailIdentity`

O código de exemplo a seguir mostra como usar `VerifyEmailIdentity`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
                              email_address: recipient
                            })

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
```

```
  puts "Email not sent. Error message: #{e}"
end
```

- Para obter detalhes da API, consulte VerifyEmailIdentity em Referência AWS SDK para Ruby da API.

Exemplos da API v2 do Amazon SES usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando a API v2 AWS SDK para Ruby com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sesv2'
require_relative 'config' # Recipient and sender email addresses.
```

```
# Set up the SESv2 client.
client = Aws::SESV2::Client.new(region: AWS_REGION)

def send_email(client, sender_email, recipient_email)
  response = client.send_email(
    {
      from_email_address: sender_email,
      destination: {
        to_addresses: [recipient_email]
      },
      content: {
        simple: {
          subject: {
            data: 'Test email subject'
          },
          body: {
            text: {
              data: 'Test email body'
            }
          }
        }
      }
    }
  )
  puts "Email sent from #{SENDER_EMAIL} to #{RECIPIENT_EMAIL} with message ID:
 #{response.message_id}"
end

send_email(client, SENDER_EMAIL, RECIPIENT_EMAIL)
```

- Para obter detalhes da API, consulte SendEmail em Referência AWS SDK para Ruby da API.

Exemplos do Amazon SNS usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Exemplos sem servidor

Ações

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# This class demonstrates how to create an Amazon Simple Notification Service (SNS)
 topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
```

```
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = 'YourTopicName' # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts 'The topic was not created. Stopping program.'
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte CreateTopic em Referência AWS SDK para Ruby da API.

`ListSubscriptions`

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# This class demonstrates how to list subscriptions to an Amazon Simple Notification
 Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
```

```
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = 'SNS_TOPIC_ARN' # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte ListSubscriptions em Referência AWS SDK para Ruby da API.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
require 'aws-sdk-sns' # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte ListTopics em Referência AWS SDK para Ruby da API.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# Service class for sending messages using Amazon Simple Notification Service (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'SNS_TOPIC_ARN' # Should be replaced with a real topic ARN
  message = 'MESSAGE'         # Should be replaced with the actual message content

  sns_client = Aws::SNS::Client.new
```

```
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info('Sending message.')
  unless message_sender.send_message(topic_arn, message)
    @logger.error('Message sending failed. Stopping program.')
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Consulte detalhes da API em Publish na Referência da API AWS SDK para Ruby .

`SetTopicAttributes`

O código de exemplo a seguir mostra como usar `SetTopicAttributes`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no Repositório de exemplos de código da AWS.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
```

```
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: '2008-10-17',
      Id: '__default_policy_ID',
      Statement: [{
        Sid: '__default_statement_ID',
        Effect: 'Allow',
        Principal: { "AWS": '*' },
        Action: ['SNS:Publish'],
        Resource: topic_arn,
        Condition: {
          ArnEquals: {
            "AWS:SourceArn": resource_arn
          }
        }
      }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'MY_TOPIC_ARN' # Should be replaced with a real topic ARN
  resource_arn = 'MY_RESOURCE_ARN' # Should be replaced with a real resource ARN
  policy_name = 'POLICY_NAME' # Typically, this is "Policy"
```

```
  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte SetTopicAttributes em Referência AWS SDK para Ruby da API.

`Assinar`

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
require 'aws-sdk-sns'
require 'logger'

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info('Subscription created successfully.')
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = 'email'
  endpoint = 'EMAIL_ADDRESS' # Should be replaced with a real email address
  topic_arn = 'TOPIC_ARN'    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info('Creating the subscription.')
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error('Subscription creation failed. Stopping program.')
    exit 1
  end
end
```

- Para obter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Ruby.
- Para obter detalhes da API, consulte Subscribe na Referência da API do AWS SDK para Ruby .

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SNS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Exemplos do Amazon SQS usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Exemplos sem servidor

Ações

`ChangeMessageVisibility`

O código de exemplo a seguir mostra como usar `ChangeMessageVisibility`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

begin
  queue_name = 'my-queue'
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Receive up to 10 messages
  receive_message_result_before = sqs.receive_message({
                                                        queue_url: queue_url,
                                                        max_number_of_messages: 10
                                                      })
```

```
  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
                                    queue_url: queue_url,
                                    receipt_handle: message.receipt_handle,
                                    visibility_timeout: 30 # This message will not
 be visible for 30 seconds after first receipt.
                                  })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
                                                       queue_url: queue_url,
                                                       max_number_of_messages: 10
                                                     })

  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
 exist."
end
```

- Para obter detalhes da API, consulte ChangeMessageVisibility em Referência AWS SDK para Ruby da API.

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
# This code example demonstrates how to create a queue in Amazon Simple Queue
 Service (Amazon SQS).

require 'aws-sdk-sqs'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts 'Queue created.'
  else
    puts 'Queue not created.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte CreateQueue em Referência AWS SDK para Ruby da API.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

- Para obter detalhes da API, consulte DeleteQueue em Referência AWS SDK para Ruby da API.

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
```

```
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-west-2'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: ['All']
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end
rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'

  sqs_client = Aws::SQS::Client.new(region: region)

  puts 'Listing available queue URLs...'
  list_queue_urls(sqs_client)
```

```
  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end
```

- Para obter detalhes da API, consulte ListQueues em Referência AWS SDK para Ruby da API.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     10
```

```
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)
  if max_number_of_messages > 10
    puts 'Maximum number of messages to receive must be 10 or less. ' \
      'Stopping program.'
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )

  if response.messages.count.zero?
    puts 'No messages to receive, or all messages have already ' \
      'been previously received.'
    return
  end

  response.messages.each do |message|
    puts '-' * 20
    puts "Message body: #{message.body}"
    puts "Message ID:   #{message.message_id}"
  end
rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)
```

```
  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte ReceiveMessage em Referência AWS SDK para Ruby da API.

`SendMessage`

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
```

```
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  message_body = 'This is my message.'

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending a message to the queue named '#{queue_name}'..."

  if message_sent?(sqs_client, queue_url, message_body)
    puts 'Message sent.'
  else
    puts 'Message not sent.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter detalhes da API, consulte SendMessage em Referência AWS SDK para Ruby da API.

`SendMessageBatch`

O código de exemplo a seguir mostra como usar `SendMessageBatch`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
```

```
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
  end
end
```

- Para obter detalhes da API, consulte SendMessageBatch em Referência AWS SDK para Ruby da API.

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

AWS STS exemplos de uso do SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby with AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para Ruby

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
```

```
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: 'create-use-assume-role-scenario'
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- Para obter detalhes da API, consulte AssumeRole em Referência AWS SDK para Ruby da API.

Exemplos do Amazon Textract usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon Textract.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos do Amazon Translate usando o SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Ruby com o Amazon Translate.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter instruções sobre o código-fonte e a implantação, consulte o projeto em GitHub.

Serviços usados neste exemplo

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Exemplos de código para SDK para Rust

Os exemplos de código a seguir mostram como usar o AWS SDK para Rust com. AWS

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para Rust — Saiba mais sobre como usar o Rust com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos da API Gateway usando o SDK para Rust
- A API de gerenciamento do API Gateway usando o SDK para Rust
- Exemplos do aplicativo Auto Scaling usando o SDK para Rust
- Exemplos de Aurora usando o SDK para Rust
- Exemplos do Auto Scaling usando o SDK para Rust
- Exemplos do Amazon Bedrock Runtime usando o SDK para Rust
- Exemplos do Amazon Bedrock Agents Runtime usando o SDK para Rust
- Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Rust
- Exemplos do Amazon Cognito Sync usando o SDK para Rust
- Exemplos do Firehose usando o SDK para Rust

- Exemplos do Amazon DocumentDB usando o SDK para Rust
- Exemplos do DynamoDB usando o SDK para Rust
- Exemplos do Amazon EBS usando o SDK para Rust
- Exemplos do Amazon EC2 usando o SDK para Rust
- Exemplos do Amazon ECR usando o SDK para Rust
- Exemplos do Amazon ECS usando o SDK para Rust
- Exemplos do Amazon EKS usando o SDK para Rust
- AWS Glue exemplos de uso do SDK para Rust
- Exemplos do IAM usando o SDK para Rust
- AWS IoT exemplos de uso do SDK para Rust
- Exemplos do Kinesis usando o SDK para Rust
- AWS KMS exemplos de uso do SDK para Rust
- Exemplos de Lambda usando SDKs para Rust
- MediaLive exemplos de uso do SDK para Rust
- MediaPackage exemplos de uso do SDK para Rust
- Exemplos do Amazon MSK usando o SDK para Rust
- Exemplos do Amazon Polly usando o SDK para Rust
- Exemplos do Amazon RDS usando o SDK para Rust
- Exemplos do Amazon RDS Data Service usando o SDK para Rust
- Exemplos do Amazon Rekognition usando o SDK para Rust
- Exemplos de Route 53 usando o SDK para Rust
- Exemplos do Amazon S3 usando o SDK para Rust
- SageMaker Exemplos de IA usando o SDK para Rust
- Exemplos de Secrets Manager usando o SDK para Rust
- Exemplos API v2 do Amazon SES usando o SDK para Rust
- Exemplos do Amazon SNS usando o SDK para Rust
- Exemplos do Amazon SQS usando o SDK para Rust
- AWS STS exemplos de uso do SDK para Rust
- Exemplos do Systems Manager usando o SDK para Rust
- Exemplos do Amazon Transcribe usando o SDK para Rust

Exemplos da API Gateway usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o API Gateway.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários
- AWS contribuições da comunidade

Ações

`GetRestApis`

O código de exemplo a seguir mostra como usar `GetRestApis`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Exiba as APIs REST do Amazon API Gateway na região.

```
async fn show_apis(client: &Client) -> Result<(), Error> {
```

```
    let resp = client.get_rest_apis().send().await?;

    for api in resp.items() {
        println!("ID:          {}", api.id().unwrap_or_default());
        println!("Name:        {}", api.name().unwrap_or_default());
        println!("Description: {}", api.description().unwrap_or_default());
        println!("Version:     {}", api.version().unwrap_or_default());
        println!(
            "Created:     {}",
            api.created_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetRestApis referência da API AWS SDK for Rust.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Rust

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Rust.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

A API de gerenciamento do API Gateway usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com a API Gateway Management API.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PostToConnection`

O código de exemplo a seguir mostra como usar `PostToConnection`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn send_data(
    client: &aws_sdk_apigatewaymanagement::Client,
    con_id: &str,
    data: &str,
) -> Result<(), aws_sdk_apigatewaymanagement::Error> {
    client
        .post_to_connection()
        .connection_id(con_id)
        .data(Blob::new(data))
        .send()
        .await?;

    Ok(())
}

    let endpoint_url = format!(
        "https://{api_id}.execute-api.{region}.amazonaws.com/{stage}",
        api_id = api_id,
        region = region,
        stage = stage
    );

    let shared_config = aws_config::from_env().region(region_provider).load().await;
    let api_management_config = config::Builder::from(&shared_config)
        .endpoint_url(endpoint_url)
        .build();
```

```
    let client = Client::from_conf(api_management_config);
```

- Para obter detalhes da API, consulte a PostToConnection referência da API AWS SDK for Rust.

Exemplos do aplicativo Auto Scaling usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Application Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DescribeScalingPolicies`

O código de exemplo a seguir mostra como usar `DescribeScalingPolicies`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_policies(client: &Client) -> Result<(), Error> {
    let response = client
        .describe_scaling_policies()
        .service_namespace(ServiceNamespace::Ec2)
```

```
        .send()
        .await?;
    println!("Auto Scaling Policies:");
    for policy in response.scaling_policies() {
        println!("{:?}\n", policy);
    }
    println!("Next token: {:?}", response.next_token());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeScalingPolicies referência da API AWS SDK for Rust.

Exemplos de Aurora usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Aurora.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Aurora

O exemplo de código a seguir mostra como começar a usar o Aurora.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
```

```
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Rust.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de cluster do banco de dados do Aurora e definir os valores dos parâmetros.
- Criar um cluster de banco de dados que use o grupo de parâmetros.
- Criar uma instância de banco de dados que contenha um banco de dados.
- Crie um snapshot do cluster do banco de dados e limpe os recursos.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Uma biblioteca contendo as funções específicas para o cenário do Aurora.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
```

```
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str = "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}
```

```
#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) -> Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
```

```
            f,
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }
```

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");

       if let Err(err) = describe_db_engine_versions {
           return Err(ScenarioError::new(
               "Failed to retrieve DB Engine Versions",
               &err,
           ));
       };

       let version_count = describe_db_engine_versions
           .as_ref()
           .map(|o| o.db_engine_versions().len())
           .unwrap_or_default();
       info!(version_count, "got list of versions");

       // Create a map of engine families to their available versions.
       let mut versions = HashMap::<String, Vec<String>>::new();
       describe_db_engine_versions
           .unwrap()
           .db_engine_versions()
           .iter()
           .filter_map(
               |v| match (&v.db_parameter_group_family, &v.engine_version) {
                   (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                   _ => None,
               },
           )
           .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

       Ok(versions)
   }

   pub async fn get_instance_classes(&self) -> Result<Vec<String>, ScenarioError> {
       let describe_orderable_db_instance_options_items = self
           .rds
```

```
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o| o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
```

```
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
```

```
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
    // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
    // Parse the ParameterName, Description, and AllowedValues values and display
them.
    pub async fn cluster_parameters(&self) -> Result<Vec<AuroraScenarioParameter>,
ScenarioError> {
        let parameters_output = self
            .rds
            .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
            .await;

        if let Err(err) = parameters_output {
            return Err(ScenarioError::new(
                format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
                &err,
            ));
        }

        let parameters = parameters_output
            .unwrap()
            .into_iter()
            .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
            .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
            .map(AuroraScenarioParameter::from)
            .collect::<Vec<_>>();
```

```
        Ok(parameters)
    }

    // Modify both the auto_increment_offset and auto_increment_increment parameters
in one call in the custom parameter group. Set their ParameterValue fields to a new
allowable value. rds.ModifyDbClusterParameterGroup.
    pub async fn update_auto_increment(
        &self,
        offset: u8,
        increment: u8,
    ) -> Result<(), ScenarioError> {
        let modify_db_cluster_parameter_group = self
            .rds
            .modify_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
```

```
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
```

```
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;
```

```
            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
```

```
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until Status
== 'available'.
    pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
        let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
        let snapshot = self
            .rds
            .snapshot_cluster(id, format!("{id}_{name}").as_str())
            .await;
        match snapshot {
            Ok(output) => match output.db_cluster_snapshot {
                Some(snapshot) => Ok(snapshot),
                None => Err(ScenarioError::with("Missing Snapshot")),
            },
            Err(err) => Err(ScenarioError::new("Failed to create snapshot", &err)),
        }
    }

    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
```

```
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances = self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
```

```
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
```

```
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}
```

```
#[cfg(test)]
pub mod tests;
```

Testes da biblioteca usando automocks em torno do wrapper do RDS Client.

```
use crate::rds::MockRdsImpl;

use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError, DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError, ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
```

```
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
```

```
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                   DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
```

```
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
```

```
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to retrieve
 DB Engine Versions"
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
```

```
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()
```

```
 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if message
 == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if message
 == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
```

```
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
```

```
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
```

```
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(
                ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message ==
 "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
```

```
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
```

```
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
```

```
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
```

```
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
 message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}

#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
```

```
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Um binário para executar o cenário do início ao fim usando o inquiridor para que o usuário possa tomar algumas decisões.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }

    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
```

```
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to the
 Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario, anyhow::Error> {
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
 rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
```

```
    )?;

    let set_engine = scenario.set_engine(engine.as_str(), version.as_str()).await;
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for engine:
 {err}")),
    }

    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings) ->
 Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings, "auto_increment_increment",
 3);

    // Modify both the auto_increment_offset and auto_increment_increment parameters
 in one call in the custom parameter group. Set their ParameterValue fields to a new
 allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }
```

```
   // Get and display the updated parameters. Specify Source of 'user' to get just
the modified parameters. rds.DescribeDbClusterParameters(Source='user')
   show_parameters(scenario, warnings).await;

   let username = inquire::Text::new("Username for the database (default
'testuser')")
       .with_default("testuser")
       .with_initial_value("testuser")
       .prompt();

   if let Err(error) = username {
       warnings.push(
           "Failed to get username, using default",
           ScenarioError::with(format!("Error from inquirer: {error}")),
       );
       return Err(());
   }
   let username = username.unwrap();

   let password = inquire::Text::new("Password for the database (minimum 8
characters)")
       .with_validator(|i: &str| {
           if i.len() >= 8 {
               Ok(inquire::validator::Validation::Valid)
           } else {
               Ok(inquire::validator::Validation::Invalid(
                   "Password must be at least 8 characters".into(),
               ))
           }
       })
       .prompt();

   let password: Option<SecretString> = match password {
       Ok(password) => Some(SecretString::from(password)),
       Err(error) => {
           warnings.push(
               "Failed to get password, using none (and not starting a DB)",
               ScenarioError::with(format!("Error from inquirer: {error}")),
           );
           return Err(());
       }
   };
```

```
    scenario.set_login(Some(username), password);

    Ok(())
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError> {
    // Create an Aurora DB cluster database cluster that contains a MySql database
 and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
 DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until Status
 == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;
```

```
    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();

    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the instance
 and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
```

```
    let parameters = scenario.cluster_parameters().await;

    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) -> u8
 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Um wrapper do serviço Amazon RDS que permite automocking para testes.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError, DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError, DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError, ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;

#[cfg(test)]
```

```
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
 SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }
```

```
    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

    pub async fn modify_db_cluster_parameter_group(
```

```
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
```

```
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
```

```
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }
}
```

As Cargo.toml com dependências usadas nesse cenário.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/
manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance

- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Ações

`CreateDBCluster`

O código de exemplo a seguir mostra como usar `CreateDBCluster`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
```

```
        if self.password.is_none() {
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
            ));
        }
        let create_db_cluster = self
            .rds
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
```

```
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
```

```
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }
```

```
    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
```

```
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
```

```
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
```

```
                    "create db cluster error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
```

```
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
```

```
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em CreateDBCluster na Referência da API AWS SDK para Rust.

`CreateDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
       let create_db_cluster_parameter_group = self
           .rds
           .create_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
               engine,
           )
           .await;

       match create_db_cluster_parameter_group {
           Ok(CreateDbClusterParameterGroupOutput {
               db_cluster_parameter_group: None,
```

```
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
 SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert!(set_engine.is_err());
}
```

```
#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                   DbParameterGroupAlreadyExistsFault::builder().build(),
               ),
               Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- Para obter detalhes da API, consulte a CreateDBClusterParameterGroup referência da API AWS SDK for Rust.

`CreateDBClusterSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBClusterSnapshot`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
```

```
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
```

```
                .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = cluster {
            warn!(?err, "Failed to describe cluster while waiting for ready");
            continue;
        }

        let instance = self
            .rds
            .describe_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = instance {
            return Err(ScenarioError::new(
                "Failed to find instance for cluster",
                &err,
            ));
        }

        let instances_available = instance
            .unwrap()
            .db_instances()
            .iter()
            .all(|instance| instance.db_instance_status() == Some("Available"));

        let endpoints = self
            .rds
            .describe_db_cluster_endpoints(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = endpoints {
            return Err(ScenarioError::new(
                "Failed to find endpoint for cluster",
                &err,
            ));
```

```
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
 SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
```

```
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
```

```
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
```

```
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
```

```
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte a CreateDBClusterSnapshot referência da API AWS SDK for Rust.

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
```

```
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }

       info!(
```

```
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
                continue;
            }
```

```
            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));
```

```
            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
```

```
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
```

```
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
```

```
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em CreateDBInstance na Referência da API AWS SDK para Rust.

`DeleteDBCluster`

O código de exemplo a seguir mostra como usar `DeleteDBCluster`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
```

```
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances = self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }
```

```
        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
```

```
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
```

```
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
```

```
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
Cluster
```

```
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
```

```
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
 message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DeleteDBCluster na Referência da API AWS SDK para Rust.

`DeleteDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBClusterParameterGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
```

```
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances = self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
```

```
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
```

```
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
```

```
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));
```

```
    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
```

```
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
 message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Para obter detalhes da API, consulte a DeleteDBClusterParameterGroup referência da API AWS SDK for Rust.

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
    let mut clean_up_errors: Vec<ScenarioError> = vec![];

    // Delete the instance. rds.DeleteDbInstance.
    let delete_db_instance = self
        .rds
        .delete_db_instance(
            self.db_instance_identifier
                .as_deref()
                .expect("instance identifier"),
        )
        .await;
    if let Err(err) = delete_db_instance {
        let identifier = self
            .db_instance_identifier
            .as_deref()
            .unwrap_or("Missing Instance Identifier");
        let message = format!("failed to delete db instance {identifier}");
        clean_up_errors.push(ScenarioError::new(message, &err));
    } else {
        // Wait for the instance to delete
        let waiter = Waiter::default();
        while waiter.sleep().await.is_ok() {
            let describe_db_instances = self.rds.describe_db_instances().await;
            if let Err(err) = describe_db_instances {
                clean_up_errors.push(ScenarioError::new(
                    "Failed to check instance state during deletion",
                    &err,
                ));
                break;
            }
            let db_instances = describe_db_instances
                .unwrap()
                .db_instances()
```

```
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
```

```
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
```

```
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
```

```
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
```

```
                    "describe db clusters error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
 message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DeleteDBInstance na Referência da API AWS SDK para Rust.

`DescribeDBClusterParameters`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameters`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) -> Result<Vec<AuroraScenarioParameter>,
ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
```

```
            .filter(|p|
 FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
            .map(AuroraScenarioParameter::from)
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
 SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
```

```
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameters referência da API AWS SDK for Rust.

`DescribeDBClusters`

O código de exemplo a seguir mostra como usar `DescribeDBClusters`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
```

```
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
```

```
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
```

```
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
```

```
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
```

```
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
```

```
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
```

```
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
```

```
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });
```

```
    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DescribeDBClusters na Referência da API AWS SDK para Rust.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");

       if let Err(err) = describe_db_engine_versions {
           return Err(ScenarioError::new(
               "Failed to retrieve DB Engine Versions",
               &err,
           ));
       };

       let version_count = describe_db_engine_versions
           .as_ref()
           .map(|o| o.db_engine_versions().len())
           .unwrap_or_default();
       info!(version_count, "got list of versions");

       // Create a map of engine families to their available versions.
       let mut versions = HashMap::<String, Vec<String>>::new();
       describe_db_engine_versions
           .unwrap()
           .db_engine_versions()
           .iter()
           .filter_map(
               |v| match (&v.db_parameter_group_family, &v.engine_version) {
                   (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                   _ => None,
               },
           )
           .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

       Ok(versions)
   }

   pub async fn describe_db_engine_versions(
       &self,
```

```
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
```

```
    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to retrieve
 DB Engine Versions"
    );
}
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersions referência da API AWS SDK for Rust.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
    let mut clean_up_errors: Vec<ScenarioError> = vec![];

    // Delete the instance. rds.DeleteDbInstance.
    let delete_db_instance = self
        .rds
        .delete_db_instance(
            self.db_instance_identifier
                .as_deref()
                .expect("instance identifier"),
        )
        .await;
    if let Err(err) = delete_db_instance {
        let identifier = self
            .db_instance_identifier
            .as_deref()
            .unwrap_or("Missing Instance Identifier");
        let message = format!("failed to delete db instance {identifier}");
        clean_up_errors.push(ScenarioError::new(message, &err));
    } else {
        // Wait for the instance to delete
        let waiter = Waiter::default();
        while waiter.sleep().await.is_ok() {
            let describe_db_instances = self.rds.describe_db_instances().await;
            if let Err(err) = describe_db_instances {
                clean_up_errors.push(ScenarioError::new(
                    "Failed to check instance state during deletion",
                    &err,
                ));
                break;
            }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
```

```
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
```

```
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
```

```
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
```

```
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
```

```
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
```

```
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
 message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Consulte detalhes da API em DescribeDBInstances na Referência da API AWS SDK para Rust.

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn get_instance_classes(&self) -> Result<Vec<String>, ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o| o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
```

```
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
```

```
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
```

```
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptions referência da API AWS SDK for Rust.

`ModifyDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // Modify both the auto_increment_offset and auto_increment_increment parameters
 in one call in the custom parameter group. Set their ParameterValue fields to a new
 allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               vec![
                   Parameter::builder()
                       .parameter_name("auto_increment_offset")
                       .parameter_value(format!("{offset}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
                   Parameter::builder()
                       .parameter_name("auto_increment_increment")
```

```
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
 SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
```

```
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(
                ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
```

```
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message ==
 "Failed to modify cluster parameter group");
}
```

- Para obter detalhes da API, consulte a ModifyDBClusterParameterGroup referência da API AWS SDK for Rust.

Exemplos do Auto Scaling usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com Auto Scaling.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Auto Scaling

O exemplo de código a seguir mostra como começar a usar o Auto Scaling.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroups referência da API AWS SDK for Rust.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo do Amazon EC2 Auto Scaling com um modelo de inicialização e zonas de disponibilidade e obter informações sobre instâncias em execução.
- Habilite a coleta de CloudWatch métricas da Amazon.
- Atualizar a capacidade desejada do grupo e aguardar a inicialização de uma instância.
- Encerrar uma instância no grupo.
- Listar as atividades de ajuste de escala que ocorrem em resposta às solicitações do usuário e às mudanças de capacidade.
- Obtenha estatísticas para CloudWatch métricas e, em seguida, limpe os recursos.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
[package]
name = "autoscaling-code-examples"
version = "0.1.0"
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
 <dpsouth@amazon.com>"]
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/
manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
```

```
clap = { version = "4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"


use std::{collections::BTreeSet, fmt::Display};

use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

#[tokio::main]
```

```
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();

    // 1. Create an EC2 launch template that you'll use to create an auto scaling
 group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch template.
    // 2. CreateAutoScalingGroup: pass it the launch template you created in step 0.
 Give it min/max of 1 instance.
    // 4. EnableMetricsCollection: enable all metrics or a subset.
    let scenario = match AutoScalingScenario::prepare_scenario(&shared_config).await
 {
        Ok(scenario) => scenario,
        Err(errs) => {
            let err_str = errs
                .into_iter()
                .map(|e| e.to_string())
                .collect::<Vec<String>>()
                .join(", ");
            return Err(anyhow!("Failed to initialize scenario: {err_str}"));
        }
    };

    info!("Prepared autoscaling scenario:\n{scenario}");

    let stable = scenario.wait_for_stable(1).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 3. DescribeAutoScalingInstances: show that one instance has launched.
    show_scenario_description(
        &scenario,
        "show that the group was created and one instance has launched",
    )
    .await;

    // 5. UpdateAutoScalingGroup: update max size to 3.
    let scale_max_size = scenario.scale_max_size(3).await;
```

```
    if let Err(err) = scale_max_size {
        warnings.push("There was a problem scaling max size", err);
    }

    // 6. DescribeAutoScalingGroups: the current state of the group
    show_scenario_description(
        &scenario,
        "show the current state of the group after setting max size",
    )
    .await;

    // 7. SetDesiredCapacity: set desired capacity to 2.
    let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
    if let Err(err) = scale_desired_capacity {
        warnings.push("There was a problem setting desired capacity", err);
    }

    //   Wait for a second instance to launch.
    let stable = scenario.wait_for_stable(2).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 8. DescribeAutoScalingInstances: show that two instances are launched.
    show_scenario_description(
        &scenario,
        "show that two instances are launched after setting desired capacity",
    )
    .await;

    let ids_before = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    // 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in the
group.
    let terminate_some_instance = scenario.terminate_some_instance().await;
    if let Err(err) = terminate_some_instance {
        warnings.push("There was a problem replacing an instance", err);
```

```
    }

    let wait_after_terminate = scenario.wait_for_stable(1).await;
    if let Err(err) = wait_after_terminate {
        warnings.push(
            "There was a problem waiting after terminating an instance",
            err,
        );
    }

    let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
    if let Err(err) = wait_scale_up_after_terminate {
        warnings.push(
            "There was a problem waiting for scale up after terminating an
instance",
            err,
        );
    }

    let ids_after = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    let difference = ids_after.intersection(&ids_before).count();
    if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
        warnings.push(
            "Before and after set not different",
            ScenarioError::with(format!("{difference}")),
        );
    }

    // 10. DescribeScalingActivities: list the scaling activities that have occurred
for the group so far.
    show_scenario_description(
        &scenario,
        "list the scaling activities that have occurred for the group so far",
    )
    .await;

    // 11. DisableMetricsCollection
    let scale_group = scenario.scale_group_to_zero().await;
    if let Err(err) = scale_group {
```

```
        warnings.push("There was a problem scaling the group to 0", err);
    }
    show_scenario_description(&scenario, "Scenario scaled to 0").await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all instances):
    // 13. Delete LaunchTemplate.
    let clean_scenario = scenario.clean_scenario().await;
    if let Err(errs) = clean_scenario {
        for err in errs {
            warnings.push("There was a problem cleaning the scenario", err);
        }
    } else {
        info!("The scenario has been cleaned up!");
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        Err(anyhow!(
            "There were warnings during scenario execution:\n{warnings}"
        ))
    }
}

pub mod scenario;


use std::{
    error::Error,
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
```

```
const AUTOSCALING_GROUP_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25 seconds.
const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at a
 time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
            ))
        } else {
            tokio::time::sleep(WAIT_TIME).await;
            Ok(())
        }
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
```

```
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\n",
            self.launch_template_arn
        ))?;
        f.write_fmt(format_args!(
            "\tScaling Group Name: {}\n",
            self.auto_scaling_group_name
        ))?;

        Ok(())
    }
}

pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t      Group status:")?;
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t- {status}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! - {e}")?,
        }
        writeln!(f, "\t         Instances:")?;
        match &self.instances {
            Ok(instances) => {
                for instance in instances {
                    writeln!(f, "\t\t- {instance}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        writeln!(f, "\t        Activities:")?;
        match &self.activities {
            Ok(activities) => {
                for activity in activities {
```

```
                    writeln!(
                        f,
                        "\t\t- {} Progress: {}% Status: {:?} End: {:?}",
                        activity.cause().unwrap_or("Unknown"),
                        activity.progress.unwrap_or(-1),
                        activity.status_code(),
                        // activity.status_message().unwrap_or_default()
                        activity.end_time(),
                    )?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        Ok(())
    }
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(|s| s.to_string()),
            code: err.code().map(|s| s.to_string()),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}
```

```
#[derive(Debug)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) -> Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
 for this example
    // the ScenarioError captures the underlying information in MetadataError and
 treats it as a
    // single Error from this Crate. In other contexts, it may be appropriate to
 model the error
    // as including the SdkError as its source.
}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
 Vec<ScenarioError>> {
```

```
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);

        let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

        // Before creating any resources, prepare the list of AZs
        let availablity_zones = ec2.describe_availability_zones().send().await;
        if let Err(err) = availablity_zones {
            return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
        }

        let availability_zones: Vec<String> = availablity_zones
            .unwrap()
            .availability_zones
            .unwrap_or_default()
            .iter()
            .take(3)
            .map(|z| z.zone_name.clone().unwrap())
            .collect();

        // 1. Create an EC2 launch template that you'll use to create an auto
scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
template.
        //   * Recommended: InstanceType='t1.micro', ImageId='ami-0ca285d4c2cda3300'
        let create_launch_template = ec2
            .create_launch_template()
            .launch_template_name(LAUNCH_TEMPLATE_NAME)
            .launch_template_data(
                RequestLaunchTemplateData::builder()
                    .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
                    .image_id("ami-0ca285d4c2cda3300")
                    .build(),
            )
            .send()
            .await
            .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)])?;

        let launch_template_arn = match create_launch_template.launch_template {
            Some(launch_template) =>
launch_template.launch_template_id.unwrap_or_default(),
            None => {
                // Try to delete the launch template
                let _ = ec2
```

```
                    .delete_launch_template()
                    .launch_template_name(LAUNCH_TEMPLATE_NAME)
                    .send()
                    .await;
                return Err(vec![ScenarioError::with("Failed to load launch
template")]);
            }
        };

        // 2. CreateAutoScalingGroup: pass it the launch template you created in
step 0. Give it min/max of 1 instance.
        //   You can use EC2.describe_availability_zones() to get a list of AZs (you
have to specify an AZ when you create the group).
        //   Wait for instance to launch. Use a waiter if you have one, otherwise
DescribeAutoScalingInstances until LifecycleState='InService'
        if let Err(err) = autoscaling
            .create_auto_scaling_group()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .launch_template(
                LaunchTemplateSpecification::builder()
                    .launch_template_id(launch_template_arn.clone())
                    .version("$Latest")
                    .build(),
            )
            .max_size(1)
            .min_size(1)
            .set_availability_zones(Some(availability_zones))
            .send()
            .await
        {
            let mut errs = vec![ScenarioError::new(
                "Failed to create autoscaling group",
                &err,
            )];

            if let Err(err) = autoscaling
                .delete_auto_scaling_group()
                .auto_scaling_group_name(auto_scaling_group_name.as_str())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up autoscaling group",
                    &err,
```

```
                ));
            }

            if let Err(err) = ec2
                .delete_launch_template()
                .launch_template_id(launch_template_arn.clone())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up launch template",
                    &err,
                ));
            }
            return Err(errs);
        }

        let scenario = AutoScalingScenario {
            ec2,
            autoscaling: autoscaling.clone(), // Clients are cheap so cloning here
to prevent a move is ok.
            auto_scaling_group_name: auto_scaling_group_name.clone(),
            launch_template_arn,
        };

        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;

        match enable_metrics_collection {
            Ok(_) => Ok(scenario),
            Err(err) => {
                scenario.clean_scenario().await?;
                Err(vec![ScenarioError::new(
```

```
                    "Failed to enable metrics collections for group",
                    &err,
                )])
            }
        }
    }

    pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
        let _ = self.wait_for_no_scaling().await;
        let delete_group = self
            .autoscaling
            .delete_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 14. Delete LaunchTemplate.
        let delete_launch_template = self
            .ec2
            .delete_launch_template()
            .launch_template_id(self.launch_template_arn.clone())
            .send()
            .await;

        let early_exit = match (delete_group, delete_launch_template) {
            (Ok(_), Ok(_)) => Ok(()),
            (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the launch template",
                &e,
            )]),
            (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the scale group",
                &e,
            )]),
            (Err(e1), Err(e2)) => Err(vec![
                ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
                ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
            ]),
        };

        if early_exit.is_err() {
            early_exit
```

```
        } else {
            // Wait for delete_group to finish
            let waiter = Waiter::new();
            let mut errors = Vec::<ScenarioError>::new();
            while errors.len() < 3 {
                if let Err(e) = waiter.sleep().await {
                    errors.push(e);
                    continue;
                }
                let describe_group = self
                    .autoscaling
                    .describe_auto_scaling_groups()
                    .auto_scaling_group_names(self.auto_scaling_group_name.clone())
                    .send()
                    .await;
                match describe_group {
                    Ok(group) => match group.auto_scaling_groups().first() {
                        Some(group) => {
                            if group.status() != Some("Delete in progress") {
                                errors.push(ScenarioError::with(format!(
                                    "Group in an unknown state while deleting: {}",
                                    group.status().unwrap_or("unknown error")
                                )));
                                return Err(errors);
                            }
                        }
                        None => return Ok(()),
                    },
                    Err(err) => {
                        errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
                    }
                }
                if errors.len() > 3 {
                    return Err(errors);
                }
            }
            Err(vec![ScenarioError::with(
                "Exited cleanup wait loop without retuning success or failing after
three rounds",
            )])
        }
    }
```

```
    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times must
be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
```

```
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
```

```
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }

    pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
        let waiter = Waiter::new();
        let mut scaling = true;
        while scaling {
            waiter.sleep().await?;
            let describe_activities = self
                .autoscaling
                .describe_scaling_activities()
                .auto_scaling_group_name(self.auto_scaling_group_name.clone())
                .send()
                .await
                .map_err(|e| {
                    ScenarioError::new("Failed to get autoscaling activities for
group", &e)
                })?;
            let activities = describe_activities.activities();
            trace!(
                "Waiting for no scaling found {} activities",
                activities.len()
            );
            scaling = activities.iter().any(|a| a.progress() < Some(100));
        }
        Ok(())
    }

    pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError> {
        self.wait_for_no_scaling().await?;

        let mut group = self.get_group().await?;
        let mut count = count_group_instances(&group);

        let waiter = Waiter::new();
        while count != size {
            trace!("Waiting for stable {size} (current: {count})");
            waiter.sleep().await?;
            group = self.get_group().await?;
            count = count_group_instances(&group);
```

```
        }

        Ok(())
    }

    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using DescribeAutoScalingGroup's
instances property. However, this returns a Vec<Instance>, as opposed to a
Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|i|
i.instance_id.clone().unwrap_or_default()).filter(|id| !id.is_empty()).collect())

        // Alternatively, and for the sake of example, DescribeAutoScalingInstances
returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }

    pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(size)
            .send()
            .await;
```

```
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failer to update group to min size ({size}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
        // 5. UpdateAutoScalingGroup: update max size to 3.
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .max_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to max size ({size})").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
        // 7. SetDesiredCapacity: set desired capacity to 2.
        //   Wait for a second instance to launch.
        let update_group = self
            .autoscaling
            .set_desired_capacity()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .desired_capacity(capacity)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to desired capacity
({capacity}))").as_str(),
                &err,
            ));
```

```
        }
        Ok(())
    }

    pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
        //   UpdateAutoScalingGroup with MinSize=0
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(0)
            .desired_capacity(0)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                "Failed to update group for scaling down&",
                &err,
            ));
        }

        let stable = self.wait_for_stable(0).await;
        if let Err(err) = stable {
            return Err(ScenarioError::with(format!(
                "Error while waiting for group to be stable on scale down: {err}"
            )));
        }

        Ok(())
    }

    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
```

```
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances

- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Ações

`CreateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `CreateAutoScalingGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error> {
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateAutoScalingGroup referência da API AWS SDK for Rust.

`DeleteAutoScalingGroup`

O código de exemplo a seguir mostra como usar `DeleteAutoScalingGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(), Error>
 {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteAutoScalingGroup referência da API AWS SDK for Rust.

`DescribeAutoScalingGroups`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingGroups`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroups referência da API AWS SDK for Rust.

`DescribeAutoScalingInstances`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
       // The direct way to list instances is by using DescribeAutoScalingGroup's
instances property. However, this returns a Vec<Instance>, as opposed to a
Vec<AutoScalingInstanceDetails>.
       // Ok(self.get_group().await?.instances.unwrap_or_default().map(|i|
i.instance_id.clone().unwrap_or_default()).filter(|id| !id.is_empty()).collect())

       // Alternatively, and for the sake of example, DescribeAutoScalingInstances
returns a list that can be filtered by the client.
       self.autoscaling
           .describe_auto_scaling_instances()
           .into_paginator()
           .items()
           .send()
           .try_collect()
           .await
           .map(|items| {
               items
                   .into_iter()
                   .filter(|i| {
                       i.auto_scaling_group_name.as_deref()
                           == Some(self.auto_scaling_group_name.as_str())
                   })
                   .map(|i| i.instance_id.unwrap_or_default())
                   .filter(|id| !id.is_empty())
                   .collect::<Vec<String>>()
           })
           .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
   }
```

- Para obter detalhes da API, consulte a DescribeAutoScalingInstances referência da API AWS SDK for Rust.

`DescribeScalingActivities`

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
```

```
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times must
be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }
```

- Para obter detalhes da API, consulte a DescribeScalingActivities referência da API AWS SDK for Rust.

`DisableMetricsCollection`

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

- Para obter detalhes da API, consulte a DisableMetricsCollection referência da API AWS SDK for Rust.

`EnableMetricsCollection`

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;
```

- Para obter detalhes da API, consulte a EnableMetricsCollection referência da API AWS SDK for Rust.

`SetDesiredCapacity`

O código de exemplo a seguir mostra como usar `SetDesiredCapacity`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
       // 7. SetDesiredCapacity: set desired capacity to 2.
       //   Wait for a second instance to launch.
       let update_group = self
           .autoscaling
           .set_desired_capacity()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .desired_capacity(capacity)
           .send()
           .await;
       if let Err(err) = update_group {
           return Err(ScenarioError::new(
               format!("Failed to update group to desired capacity
({capacity}))").as_str(),
               &err,
           ));
       }
       Ok(())
   }
```

- Para obter detalhes da API, consulte a SetDesiredCapacity referência da API AWS SDK for Rust.

`TerminateInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
    // Retrieve a list of instances in the auto scaling group.
    let auto_scaling_group = self.get_group().await?;
    let instances = auto_scaling_group.instances();
    // Or use other logic to find an instance to terminate.
    let instance = instances.first();
    if let Some(instance) = instance {
        let instance_id = if let Some(instance_id) = instance.instance_id() {
            instance_id
        } else {
            return Err(ScenarioError::with("Missing instance id"));
        };
        let termination = self
            .ec2
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await;
        if let Err(err) = termination {
            Err(ScenarioError::new(
                "There was a problem terminating an instance",
                &err,
            ))
        } else {
            Ok(())
        }
    } else {
        Err(ScenarioError::with("There was no instance to terminate"))
    }
}
```

```
    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }
```

- Para obter detalhes da API, consulte a TerminateInstanceInAutoScalingGroup referência da API AWS SDK for Rust.

`UpdateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `UpdateAutoScalingGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(), Error> {
    client
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
        .send()
        .await?;

    println!("Updated AutoScaling group");

    Ok(())
}
```

- Para obter detalhes da API, consulte a UpdateAutoScalingGroup referência da API AWS SDK for Rust.

Exemplos do Amazon Bedrock Runtime usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon Bedrock Runtime.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários
- Claude da Anthropic

Cenários

Uso de ferramenta com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O cenário principal e a lógica da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
#[derive(Debug)]
#[allow(dead_code)]
struct InvokeToolResult(String, ToolResultBlock);
struct ToolUseScenario {
    client: Client,
    conversation: Vec<Message>,
    system_prompt: SystemContentBlock,
    tool_config: ToolConfiguration,
}

impl ToolUseScenario {
    fn new(client: Client) -> Self {
        let system_prompt = SystemContentBlock::Text(SYSTEM_PROMPT.into());
        let tool_config = ToolConfiguration::builder()
            .tools(Tool::ToolSpec(
                ToolSpecification::builder()
                    .name(TOOL_NAME)
                    .description(TOOL_DESCRIPTION)
                    .input_schema(ToolInputSchema::Json(make_tool_schema()))
                    .build()
                    .unwrap(),
```

```
            ))
            .build()
            .unwrap();

        ToolUseScenario {
            client,
            conversation: vec![],
            system_prompt,
            tool_config,
        }
    }

    async fn run(&mut self) -> Result<(), ToolUseScenarioError> {
        loop {
            let input = get_input().await?;
            if input.is_none() {
                break;
            }

            let message = Message::builder()
                .role(User)
                .content(ContentBlock::Text(input.unwrap()))
                .build()
                .map_err(ToolUseScenarioError::from)?;
            self.conversation.push(message);

            let response = self.send_to_bedrock().await?;

            self.process_model_response(response).await?;
        }

        Ok(())
    }

    async fn send_to_bedrock(&mut self) -> Result<ConverseOutput,
ToolUseScenarioError> {
        debug!("Sending conversation to bedrock");
        self.client
            .converse()
            .model_id(MODEL_ID)
            .set_messages(Some(self.conversation.clone()))
            .system(self.system_prompt.clone())
            .tool_config(self.tool_config.clone())
            .send()
```

```
            .await
            .map_err(ToolUseScenarioError::from)
    }

    async fn process_model_response(
        &mut self,
        mut response: ConverseOutput,
    ) -> Result<(), ToolUseScenarioError> {
        let mut iteration = 0;

        while iteration < MAX_RECURSIONS {
            iteration += 1;
            let message = if let Some(ref output) = response.output {
                if output.is_message() {
                    Ok(output.as_message().unwrap().clone())
                } else {
                    Err(ToolUseScenarioError(
                        "Converse Output is not a message".into(),
                    ))
                }
            } else {
                Err(ToolUseScenarioError("Missing Converse Output".into()))
            }?;

            self.conversation.push(message.clone());

            match response.stop_reason {
                StopReason::ToolUse => {
                    response = self.handle_tool_use(&message).await?;
                }
                StopReason::EndTurn => {
                    print_model_response(&message.content[0])?;
                    return Ok(());
                }
                _ => (),
            }
        }

        Err(ToolUseScenarioError(
            "Exceeded MAX_ITERATIONS when calling tools".into(),
        ))
    }

    async fn handle_tool_use(
```

```
        &mut self,
        message: &Message,
    ) -> Result<ConverseOutput, ToolUseScenarioError> {
        let mut tool_results: Vec<ContentBlock> = vec![];

        for block in &message.content {
            match block {
                ContentBlock::Text(_) => print_model_response(block)?,
                ContentBlock::ToolUse(tool) => {
                    let tool_response = self.invoke_tool(tool).await?;
                    tool_results.push(ContentBlock::ToolResult(tool_response.1));
                }
                _ => (),
            };
        }

        let message = Message::builder()
            .role(User)
            .set_content(Some(tool_results))
            .build()?;
        self.conversation.push(message);

        self.send_to_bedrock().await
    }

    async fn invoke_tool(
        &mut self,
        tool: &ToolUseBlock,
    ) -> Result<InvokeToolResult, ToolUseScenarioError> {
        match tool.name() {
            TOOL_NAME => {
                println!(
                    "\x1b[0;90mExecuting tool: {TOOL_NAME} with input: {:?}...
\x1b[0m",
                    tool.input()
                );
                let content = fetch_weather_data(tool).await?;
                println!(
                    "\x1b[0;90mTool responded with {:?}\x1b[0m",
                    content.content()
                );
                Ok(InvokeToolResult(tool.tool_use_id.clone(), content))
            }
            _ => Err(ToolUseScenarioError(format!(
```

```
                "The requested tool with name {} does not exist",
                tool.name()
            ))),
        }
    }
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let mut scenario = ToolUseScenario::new(client);

    header();
    if let Err(err) = scenario.run().await {
        println!("There was an error running the scenario! {}", err.0)
    }
    footer();
}
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
const ENDPOINT: &str = "https://api.open-meteo.com/v1/forecast";
async fn fetch_weather_data(
    tool_use: &ToolUseBlock,
) -> Result<ToolResultBlock, ToolUseScenarioError> {
    let input = tool_use.input();
    let latitude = input
        .as_object()
        .unwrap()
        .get("latitude")
        .unwrap()
        .as_string()
        .unwrap();
    let longitude = input
```

```
        .as_object()
        .unwrap()
        .get("longitude")
        .unwrap()
        .as_string()
        .unwrap();
    let params = [
        ("latitude", latitude),
        ("longitude", longitude),
        ("current_weather", "true"),
    ];

    debug!("Calling {ENDPOINT} with {params:?}");

    let response = reqwest::Client::new()
        .get(ENDPOINT)
        .query(&params)
        .send()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error requesting weather:
 {e:?}")))?
        .error_for_status()
        .map_err(|e| ToolUseScenarioError(format!("Failed to request weather:
 {e:?}")))?;

    debug!("Response: {response:?}");

    let bytes = response
        .bytes()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error reading response:
 {e:?}")))?;

    let result = String::from_utf8(bytes.to_vec())
        .map_err(|_| ToolUseScenarioError("Response was not utf8".into()))?;

    Ok(ToolResultBlock::builder()
        .tool_use_id(tool_use.tool_use_id())
        .content(ToolResultContentBlock::Text(result))
        .build()?)
}
```

Utilitários para imprimir os blocos de conteúdo da mensagem.

```
fn print_model_response(block: &ContentBlock) -> Result<(), ToolUseScenarioError> {
    if block.is_text() {
        let text = block.as_text().unwrap();
        println!("\x1b[0;90mThe model's response:\x1b[0m\n{text}");
        Ok(())
    } else {
        Err(ToolUseScenarioError(format!(
            "Content block is not text ({block:?})"
        )))
    }
}
```

Use instruções, utilitário de erro e constantes.

```
use std::{collections::HashMap, io::stdin};

use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::{BuildError, SdkError},
    operation::converse::{ConverseError, ConverseOutput},
    types::{
        ContentBlock, ConversationRole::User, Message, StopReason,
 SystemContentBlock, Tool,
        ToolConfiguration, ToolInputSchema, ToolResultBlock, ToolResultContentBlock,
        ToolSpecification, ToolUseBlock,
    },
    Client,
};
use aws_smithy_runtime_api::http::Response;
use aws_smithy_types::Document;
use tracing::debug;

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

const SYSTEM_PROMPT: &str = "You are a weather assistant that provides current
 weather data for user-specified locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
```

```
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
";

// The maximum number of recursive calls allowed in the tool_use_demo function.
// This helps prevent infinite loops and potential performance issues.
const MAX_RECURSIONS: i8 = 5;

const TOOL_NAME: &str = "Weather_Tool";
const TOOL_DESCRIPTION: &str =
    "Get the current weather for a given location, based on its WGS84 coordinates.";
fn make_tool_schema() -> Document {
    Document::Object(HashMap::<String, Document>::from([
        ("type".into(), Document::String("object".into())),
        (
            "properties".into(),
            Document::Object(HashMap::from([
                (
                    "latitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String("Geographical WGS84 latitude of the
 location.".into()),
                        ),
                    ])),
                ),
                (
```

```
                    "longitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String(
                                "Geographical WGS84 longitude of the
 location.".into(),
                            ),
                        ),
                    ])),
                ),
            ])),
        ),
        (
            "required".into(),
            Document::Array(vec![
                Document::String("latitude".into()),
                Document::String("longitude".into()),
            ]),
        ),
    ]))
}

#[derive(Debug)]
struct ToolUseScenarioError(String);
impl std::fmt::Display for ToolUseScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Tool use error with '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl From<&str> for ToolUseScenarioError {
    fn from(value: &str) -> Self {
        ToolUseScenarioError(value.into())
    }
}
impl From<BuildError> for ToolUseScenarioError {
    fn from(value: BuildError) -> Self {
        ToolUseScenarioError(value.to_string().clone())
    }
}
impl From<SdkError<ConverseError, Response>> for ToolUseScenarioError {
    fn from(value: SdkError<ConverseError, Response>) -> Self {
        ToolUseScenarioError(match value.as_service_error() {
```

```
            Some(value) => value.meta().message().unwrap_or("Unknown").into(),
            None => "Unknown".into(),
        })
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Rust.

Claude da Anthropic

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
#[tokio::main]
async fn main() -> Result<(), BedrockConverseError> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let response = client
        .converse()
        .model_id(MODEL_ID)
        .messages(
            Message::builder()
                .role(ConversationRole::User)
```

```
                .content(ContentBlock::Text(USER_MESSAGE.to_string()))
                .build()
                .map_err(|_| "failed to build message")?,
        )
        .send()
        .await;

    match response {
        Ok(output) => {
            let text = get_converse_output_text(output)?;
            println!("{}", text);
            Ok(())
        }
        Err(e) => Err(e
            .as_service_error()
            .map(BedrockConverseError::from)
            .unwrap_or_else(|| BedrockConverseError("Unknown service
 error".into()))),
    }
}

fn get_converse_output_text(output: ConverseOutput) -> Result<String,
 BedrockConverseError> {
    let text = output
        .output()
        .ok_or("no output")?
        .as_message()
        .map_err(|_| "output not a message")?
        .content()
        .first()
        .ok_or("no content in message")?
        .as_text()
        .map_err(|_| "content is not text")?
        .to_string();
    Ok(text)
}
```

Use instruções, utilitário de erro e constantes.

```
use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    operation::converse::{ConverseError, ConverseOutput},
```

```
    types::{ContentBlock, ConversationRole, Message},
    Client,
};

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

// Start a conversation with the user message.
const USER_MESSAGE: &str = "Describe the purpose of a 'hello world' program in one
 line.";

#[derive(Debug)]
struct BedrockConverseError(String);
impl std::fmt::Display for BedrockConverseError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Can't invoke '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl std::error::Error for BedrockConverseError {}
impl From<&str> for BedrockConverseError {
    fn from(value: &str) -> Self {
        BedrockConverseError(value.to_string())
    }
}
impl From<&ConverseError> for BedrockConverseError {
    fn from(value: &ConverseError) -> Self {
        BedrockConverseError::from(match value {
            ConverseError::ModelTimeoutException(_) => "Model took too long",
            ConverseError::ModelNotReadyException(_) => "Model is not ready",
            _ => "Unknown",
        })
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Rust.

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para Anthropic Claude e transmita tokens de resposta usando a API da Bedrock. ConverseStream

```
#[tokio::main]
async fn main() -> Result<(), BedrockConverseStreamError> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let response = client
        .converse_stream()
        .model_id(MODEL_ID)
        .messages(
            Message::builder()
                .role(ConversationRole::User)
                .content(ContentBlock::Text(USER_MESSAGE.to_string()))
                .build()
                .map_err(|_| "failed to build message")?,
        )
        .send()
        .await;

    let mut stream = match response {
        Ok(output) => Ok(output.stream),
        Err(e) => Err(BedrockConverseStreamError::from(
            e.as_service_error().unwrap(),
        )),
    }?;

    loop {
        let token = stream.recv().await;
```

```
        match token {
            Ok(Some(text)) => {
                let next = get_converse_output_text(text)?;
                print!("{}", next);
                Ok(())
            }
            Ok(None) => break,
            Err(e) => Err(e
                .as_service_error()
                .map(BedrockConverseStreamError::from)
                .unwrap_or(BedrockConverseStreamError(
                    "Unknown error receiving stream".into(),
                ))),
        }?
    }

    println!();

    Ok(())
}

fn get_converse_output_text(
    output: ConverseStreamOutputType,
) -> Result<String, BedrockConverseStreamError> {
    Ok(match output {
        ConverseStreamOutputType::ContentBlockDelta(event) => match event.delta() {
            Some(delta) => delta.as_text().cloned().unwrap_or_else(|_| "".into()),
            None => "".into(),
        },
        _ => "".into(),
    })
}
```

Use instruções, utilitário de erro e constantes.

```
use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::ProvideErrorMetadata,
    operation::converse_stream::ConverseStreamError,
    types::{
        error::ConverseStreamOutputError, ContentBlock, ConversationRole,
```

```
        ConverseStreamOutput as ConverseStreamOutputType, Message,
    },
    Client,
};

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

// Start a conversation with the user message.
const USER_MESSAGE: &str = "Describe the purpose of a 'hello world' program in one
 line.";

#[derive(Debug)]
struct BedrockConverseStreamError(String);
impl std::fmt::Display for BedrockConverseStreamError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Can't invoke '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl std::error::Error for BedrockConverseStreamError {}
impl From<&str> for BedrockConverseStreamError {
    fn from(value: &str) -> Self {
        BedrockConverseStreamError(value.into())
    }
}

impl From<&ConverseStreamError> for BedrockConverseStreamError {
    fn from(value: &ConverseStreamError) -> Self {
        BedrockConverseStreamError(
            match value {
                ConverseStreamError::ModelTimeoutException(_) => "Model took too
 long",
                ConverseStreamError::ModelNotReadyException(_) => "Model is not
 ready",
                _ => "Unknown",
            }
            .into(),
        )
    }
}

impl From<&ConverseStreamOutputError> for BedrockConverseStreamError {
    fn from(value: &ConverseStreamOutputError) -> Self {
```

```
        match value {
            ConverseStreamOutputError::ValidationException(ve) =>
 BedrockConverseStreamError(
                ve.message().unwrap_or("Unknown ValidationException").into(),
            ),
            ConverseStreamOutputError::ThrottlingException(te) =>
 BedrockConverseStreamError(
                te.message().unwrap_or("Unknown ThrottlingException").into(),
            ),
            value => BedrockConverseStreamError(
                value
                    .message()
                    .unwrap_or("Unknown StreamOutput exception")
                    .into(),
            ),
        }
    }
}
```

- Para obter detalhes da API, consulte a ConverseStream referência da API AWS SDK for Rust.

Cenário: uso de ferramentas com a API Converse

O exemplo de código a seguir mostra como criar uma interação típica entre uma aplicação, um modelo de IA generativa e ferramentas conectadas ou APIs para mediar as interações entre a IA e o mundo externo. Ele usa o exemplo de conectar uma API de meteorologia externa ao modelo de IA para que possa fornecer informações de meteorologia em tempo real com base na entrada do usuário.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O cenário principal e a lógica da demonstração. Esse script orquestra a conversa entre o usuário, a API Converse do Amazon Bedrock e uma ferramenta de meteorologia.

```
#[derive(Debug)]
#[allow(dead_code)]
struct InvokeToolResult(String, ToolResultBlock);
struct ToolUseScenario {
    client: Client,
    conversation: Vec<Message>,
    system_prompt: SystemContentBlock,
    tool_config: ToolConfiguration,
}

impl ToolUseScenario {
    fn new(client: Client) -> Self {
        let system_prompt = SystemContentBlock::Text(SYSTEM_PROMPT.into());
        let tool_config = ToolConfiguration::builder()
            .tools(Tool::ToolSpec(
                ToolSpecification::builder()
                    .name(TOOL_NAME)
                    .description(TOOL_DESCRIPTION)
                    .input_schema(ToolInputSchema::Json(make_tool_schema()))
                    .build()
                    .unwrap(),
            ))
            .build()
            .unwrap();

        ToolUseScenario {
            client,
            conversation: vec![],
            system_prompt,
            tool_config,
        }
    }

    async fn run(&mut self) -> Result<(), ToolUseScenarioError> {
        loop {
            let input = get_input().await?;
            if input.is_none() {
                break;
            }

            let message = Message::builder()
                .role(User)
                .content(ContentBlock::Text(input.unwrap()))
```

```
                .build()
                .map_err(ToolUseScenarioError::from)?;
            self.conversation.push(message);

            let response = self.send_to_bedrock().await?;

            self.process_model_response(response).await?;
        }

        Ok(())
    }

    async fn send_to_bedrock(&mut self) -> Result<ConverseOutput,
ToolUseScenarioError> {
        debug!("Sending conversation to bedrock");
        self.client
            .converse()
            .model_id(MODEL_ID)
            .set_messages(Some(self.conversation.clone()))
            .system(self.system_prompt.clone())
            .tool_config(self.tool_config.clone())
            .send()
            .await
            .map_err(ToolUseScenarioError::from)
    }

    async fn process_model_response(
        &mut self,
        mut response: ConverseOutput,
    ) -> Result<(), ToolUseScenarioError> {
        let mut iteration = 0;

        while iteration < MAX_RECURSIONS {
            iteration += 1;
            let message = if let Some(ref output) = response.output {
                if output.is_message() {
                    Ok(output.as_message().unwrap().clone())
                } else {
                    Err(ToolUseScenarioError(
                        "Converse Output is not a message".into(),
                    ))
                }
            } else {
                Err(ToolUseScenarioError("Missing Converse Output".into()))
```

```
            }?;

            self.conversation.push(message.clone());

            match response.stop_reason {
                StopReason::ToolUse => {
                    response = self.handle_tool_use(&message).await?;
                }
                StopReason::EndTurn => {
                    print_model_response(&message.content[0])?;
                    return Ok(());
                }
                _ => (),
            }
        }

        Err(ToolUseScenarioError(
            "Exceeded MAX_ITERATIONS when calling tools".into(),
        ))
    }

    async fn handle_tool_use(
        &mut self,
        message: &Message,
    ) -> Result<ConverseOutput, ToolUseScenarioError> {
        let mut tool_results: Vec<ContentBlock> = vec![];

        for block in &message.content {
            match block {
                ContentBlock::Text(_) => print_model_response(block)?,
                ContentBlock::ToolUse(tool) => {
                    let tool_response = self.invoke_tool(tool).await?;
                    tool_results.push(ContentBlock::ToolResult(tool_response.1));
                }
                _ => (),
            };
        }

        let message = Message::builder()
            .role(User)
            .set_content(Some(tool_results))
            .build()?;
        self.conversation.push(message);
```

```
        self.send_to_bedrock().await
    }

    async fn invoke_tool(
        &mut self,
        tool: &ToolUseBlock,
    ) -> Result<InvokeToolResult, ToolUseScenarioError> {
        match tool.name() {
            TOOL_NAME => {
                println!(
                    "\x1b[0;90mExecuting tool: {TOOL_NAME} with input: {:?}...
\x1b[0m",
                    tool.input()
                );
                let content = fetch_weather_data(tool).await?;
                println!(
                    "\x1b[0;90mTool responded with {:?}\x1b[0m",
                    content.content()
                );
                Ok(InvokeToolResult(tool.tool_use_id.clone(), content))
            }
            _ => Err(ToolUseScenarioError(format!(
                "The requested tool with name {} does not exist",
                tool.name()
            ))),
        }
    }
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let mut scenario = ToolUseScenario::new(client);

    header();
    if let Err(err) = scenario.run().await {
        println!("There was an error running the scenario! {}", err.0)
    }
```

```
    footer();
}
```

A ferramenta de meteorologia usada pela demonstração. Esse script define a especificação da ferramenta e implementa a lógica para recuperar dados meteorológicos usando a Open-Meteo API.

```
const ENDPOINT: &str = "https://api.open-meteo.com/v1/forecast";
async fn fetch_weather_data(
    tool_use: &ToolUseBlock,
) -> Result<ToolResultBlock, ToolUseScenarioError> {
    let input = tool_use.input();
    let latitude = input
        .as_object()
        .unwrap()
        .get("latitude")
        .unwrap()
        .as_string()
        .unwrap();
    let longitude = input
        .as_object()
        .unwrap()
        .get("longitude")
        .unwrap()
        .as_string()
        .unwrap();
    let params = [
        ("latitude", latitude),
        ("longitude", longitude),
        ("current_weather", "true"),
    ];

    debug!("Calling {ENDPOINT} with {params:?}");

    let response = reqwest::Client::new()
        .get(ENDPOINT)
        .query(&params)
        .send()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error requesting weather:
 {e:?}")))?
        .error_for_status()
```

```
        .map_err(|e| ToolUseScenarioError(format!("Failed to request weather:
 {e:?}")))?;

    debug!("Response: {response:?}");

    let bytes = response
        .bytes()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error reading response:
 {e:?}")))?;

    let result = String::from_utf8(bytes.to_vec())
        .map_err(|_| ToolUseScenarioError("Response was not utf8".into()))?;

    Ok(ToolResultBlock::builder()
        .tool_use_id(tool_use.tool_use_id())
        .content(ToolResultContentBlock::Text(result))
        .build()?)
}
```

Utilitários para imprimir os blocos de conteúdo da mensagem.

```
fn print_model_response(block: &ContentBlock) -> Result<(), ToolUseScenarioError> {
    if block.is_text() {
        let text = block.as_text().unwrap();
        println!("\x1b[0;90mThe model's response:\x1b[0m\n{text}");
        Ok(())
    } else {
        Err(ToolUseScenarioError(format!(
            "Content block is not text ({block:?})"
        )))
    }
}
```

Use instruções, utilitário de erro e constantes.

```
use std::{collections::HashMap, io::stdin};

use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::{BuildError, SdkError},
```

```
    operation::converse::{ConverseError, ConverseOutput},
    types::{
        ContentBlock, ConversationRole::User, Message, StopReason,
 SystemContentBlock, Tool,
        ToolConfiguration, ToolInputSchema, ToolResultBlock, ToolResultContentBlock,
        ToolSpecification, ToolUseBlock,
    },
    Client,
};
use aws_smithy_runtime_api::http::Response;
use aws_smithy_types::Document;
use tracing::debug;

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

const SYSTEM_PROMPT: &str = "You are a weather assistant that provides current
 weather data for user-specified locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
";

// The maximum number of recursive calls allowed in the tool_use_demo function.
// This helps prevent infinite loops and potential performance issues.
const MAX_RECURSIONS: i8 = 5;
```

```
const TOOL_NAME: &str = "Weather_Tool";
const TOOL_DESCRIPTION: &str =
    "Get the current weather for a given location, based on its WGS84 coordinates.";
fn make_tool_schema() -> Document {
    Document::Object(HashMap::<String, Document>::from([
        ("type".into(), Document::String("object".into())),
        (
            "properties".into(),
            Document::Object(HashMap::from([
                (
                    "latitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String("Geographical WGS84 latitude of the
 location.".into()),
                        ),
                    ])),
                ),
                (
                    "longitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String(
                                "Geographical WGS84 longitude of the
 location.".into(),
                            ),
                        ),
                    ])),
                ),
            ])),
        ),
        (
            "required".into(),
            Document::Array(vec![
                Document::String("latitude".into()),
                Document::String("longitude".into()),
            ]),
        ),
    ]))
}
```

```
#[derive(Debug)]
struct ToolUseScenarioError(String);
impl std::fmt::Display for ToolUseScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Tool use error with '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl From<&str> for ToolUseScenarioError {
    fn from(value: &str) -> Self {
        ToolUseScenarioError(value.into())
    }
}
impl From<BuildError> for ToolUseScenarioError {
    fn from(value: BuildError) -> Self {
        ToolUseScenarioError(value.to_string().clone())
    }
}
impl From<SdkError<ConverseError, Response>> for ToolUseScenarioError {
    fn from(value: SdkError<ConverseError, Response>) -> Self {
        ToolUseScenarioError(match value.as_service_error() {
            Some(value) => value.meta().message().unwrap_or("Unknown").into(),
            None => "Unknown".into(),
        })
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Rust.

Exemplos do Amazon Bedrock Agents Runtime usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon Bedrock Agents Runtime.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`InvokeAgent`

O código de exemplo a seguir mostra como usar `InvokeAgent`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use aws_config::{BehaviorVersion, SdkConfig};
use aws_sdk_bedrockagentruntime::{
    self as bedrockagentruntime,
    types::{error::ResponseStreamError, ResponseStream},
};
#[allow(unused_imports)]
use mockall::automock;

const BEDROCK_AGENT_ID: &str = "AJBHXXILZN";
const BEDROCK_AGENT_ALIAS_ID: &str = "AVKP1ITZAA";
const BEDROCK_AGENT_REGION: &str = "us-east-1";

#[cfg(not(test))]
pub use EventReceiverImpl as EventReceiver;
#[cfg(test)]
pub use MockEventReceiverImpl as EventReceiver;

pub struct EventReceiverImpl {
    inner: aws_sdk_bedrockagentruntime::primitives::event_stream::EventReceiver<
        ResponseStream,
        ResponseStreamError,
    >,
}
```

```
#[cfg_attr(test, automock)]
impl EventReceiverImpl {
    #[allow(dead_code)]
    pub fn new(
        inner: aws_sdk_bedrockagentruntime::primitives::event_stream::EventReceiver<
            ResponseStream,
            ResponseStreamError,
        >,
    ) -> Self {
        Self { inner }
    }

    pub async fn recv(
        &mut self,
    ) -> Result<
        Option<ResponseStream>,
        aws_sdk_bedrockagentruntime::error::SdkError<
            ResponseStreamError,
            aws_smithy_types::event_stream::RawMessage,
        >,
    > {
        self.inner.recv().await
    }
}

#[tokio::main]
async fn main() -> Result<(), Box<bedrockagentruntime::Error>> {
    let result = invoke_bedrock_agent("I need help.".to_string(),
 "123".to_string()).await?;
    println!("{}", result);
    Ok(())
}

async fn invoke_bedrock_agent(
    prompt: String,
    session_id: String,
) -> Result<String, bedrockagentruntime::Error> {
    let sdk_config: SdkConfig = aws_config::defaults(BehaviorVersion::latest())
        .region(BEDROCK_AGENT_REGION)
        .load()
        .await;
    let bedrock_client = bedrockagentruntime::Client::new(&sdk_config);
```

```
    let command_builder = bedrock_client
        .invoke_agent()
        .agent_id(BEDROCK_AGENT_ID)
        .agent_alias_id(BEDROCK_AGENT_ALIAS_ID)
        .session_id(session_id)
        .input_text(prompt);

    let response = command_builder.send().await?;

    let response_stream = response.completion;

    let event_receiver = EventReceiver::new(response_stream);

    process_agent_response_stream(event_receiver).await
}

async fn process_agent_response_stream(
    mut event_receiver: EventReceiver,
) -> Result<String, bedrockagentruntime::Error> {
    let mut full_agent_text_response = String::new();

    while let Some(event_result) = event_receiver.recv().await? {
        match event_result {
            ResponseStream::Chunk(chunk) => {
                if let Some(bytes) = chunk.bytes {
                    match String::from_utf8(bytes.into_inner()) {
                        Ok(text_chunk) => {
                            full_agent_text_response.push_str(&text_chunk);
                        }
                        Err(e) => {
                            eprintln!("UTF-8 decoding error for chunk: {}", e);
                        }
                    }
                }
            }
            _ => {
                panic!("received an unhandled event type from Bedrock stream",);
            }
        }
    }
    Ok(full_agent_text_response)
}

#[cfg(test)]
```

```
mod test {

    use super::*;

    #[tokio::test]
    async fn test_process_agent_response_stream() {
        let mut mock = MockEventReceiverImpl::default();
        mock.expect_recv().times(1).returning(|| {
            Ok(Some(
                aws_sdk_bedrockagentruntime::types::ResponseStream::Chunk(
                    aws_sdk_bedrockagentruntime::types::PayloadPart::builder()
                        .set_bytes(Some(aws_smithy_types::Blob::new(vec![
                            116, 101, 115, 116, 32, 99, 111, 109, 112, 108, 101,
 116, 105, 111, 110,
                        ])))
                        .build(),
                ),
            ))
        });

        // end the stream
        mock.expect_recv().times(1).returning(|| Ok(None));

        let response = process_agent_response_stream(mock).await.unwrap();

        assert_eq!("test completion", response);
    }

    #[tokio::test]
    #[should_panic(expected = "received an unhandled event type from Bedrock
 stream")]
    async fn test_process_agent_response_stream_error() {
        let mut mock = MockEventReceiverImpl::default();
        mock.expect_recv().times(1).returning(|| {
            Ok(Some(
                aws_sdk_bedrockagentruntime::types::ResponseStream::Trace(

 aws_sdk_bedrockagentruntime::types::TracePart::builder().build(),
                ),
            ))
        });

        let _ = process_agent_response_stream(mock).await.unwrap();
    }
```

```
}
```

- Para obter detalhes da API, consulte a InvokeAgent referência da API AWS SDK for Rust.

Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListUserPools`

O código de exemplo a seguir mostra como usar `ListUserPools`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
```

```
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListUserPools referência da API AWS SDK for Rust.

Exemplos do Amazon Cognito Sync usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon Cognito Sync.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListIdentityPoolUsage`

O código de exemplo a seguir mostra como usar `ListIdentityPoolUsage`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
```

```
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListIdentityPoolUsage referência da API AWS SDK for Rust.

Exemplos do Firehose usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com Firehose.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutRecordBatch`

O código de exemplo a seguir mostra como usar `PutRecordBatch`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn put_record_batch(
    client: &Client,
    stream: &str,
    data: Vec<Record>,
) -> Result<PutRecordBatchOutput, SdkError<PutRecordBatchError>> {
    client
        .put_record_batch()
        .delivery_stream_name(stream)
        .set_records(Some(data))
        .send()
        .await
}
```

- Para obter detalhes da API, consulte a PutRecordBatch referência da API AWS SDK for Rust.

Exemplos do Amazon DocumentDB usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::documentdb::{DocumentDbEvent, DocumentDbInnerEvent},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<DocumentDbEvent>) ->Result<(), Error> {

    tracing::info!("Event Source ARN: {:?}", event.payload.event_source_arn);
    tracing::info!("Event Source: {:?}", event.payload.event_source);

    let records = &event.payload.events;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_document_db_event(record);
    }

    tracing::info!("Document db records processed");
```

```
    // Prepare the response
    Ok(())

}

fn log_document_db_event(record: &DocumentDbInnerEvent)-> Result<(), Error>{
    tracing::info!("Change Event: {:?}", record.event);

    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Exemplos do DynamoDB usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o DynamoDB.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários
- Exemplos sem servidor
- AWS contribuições da comunidade

Ações

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_table(
    client: &Client,
    table: &str,
    key: &str,
) -> Result<CreateTableOutput, Error> {
    let a_name: String = key.into();
    let table_name: String = table.into();

    let ad = AttributeDefinition::builder()
        .attribute_name(&a_name)
        .attribute_type(ScalarAttributeType::S)
        .build()
        .map_err(Error::BuildError)?;

    let ks = KeySchemaElement::builder()
        .attribute_name(&a_name)
        .key_type(KeyType::Hash)
```

```
        .build()
        .map_err(Error::BuildError)?;

    let create_table_response = client
        .create_table()
        .table_name(table_name)
        .key_schema(ks)
        .attribute_definitions(ad)
        .billing_mode(BillingMode::PayPerRequest)
        .send()
        .await;

    match create_table_response {
        Ok(out) => {
            println!("Added table {} with key {}", table, key);
            Ok(out)
        }
        Err(e) => {
            eprintln!("Got an error creating table:");
            eprintln!("{}", e);
            Err(Error::unhandled(e))
        }
    }
}
```

- Para obter detalhes da API, consulte a CreateTable referência da API AWS SDK for Rust.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_item(
    client: &Client,
```

```
    table: &str,
    key: &str,
    value: &str,
) -> Result<DeleteItemOutput, Error> {
    match client
        .delete_item()
        .table_name(table)
        .key(key, AttributeValue::S(value.into()))
        .send()
        .await
    {
        Ok(out) => {
            println!("Deleted item from table");
            Ok(out)
        }
        Err(e) => Err(Error::unhandled(e)),
    }
}
```

- Para obter detalhes da API, consulte a DeleteItem referência da API AWS SDK for Rust.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_table(client: &Client, table: &str) -> Result<DeleteTableOutput,
 Error> {
    let resp = client.delete_table().table_name(table).send().await;

    match resp {
        Ok(out) => {
            println!("Deleted table");
            Ok(out)
```

```
        }
        Err(e) => Err(Error::Unhandled(e.into())),
    }
}
```

- Para obter detalhes da API, consulte a DeleteTable referência da API AWS SDK for Rust.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_tables(client: &Client) -> Result<Vec<String>, Error> {
    let paginator = client.list_tables().into_paginator().items().send();
    let table_names = paginator.collect::<Result<Vec<_>, _>>().await?;

    println!("Tables:");

    for name in &table_names {
        println!("  {}", name);
    }

    println!("Found {} tables", table_names.len());
    Ok(table_names)
}
```

Determine se a tabela existe.

```
pub async fn table_exists(client: &Client, table: &str) -> Result<bool, Error> {
    debug!("Checking for table: {table}");
    let table_list = client.list_tables().send().await;
```

```
    match table_list {
        Ok(list) => Ok(list.table_names().contains(&table.into())),
        Err(e) => Err(e.into()),
    }
}
```

- Para obter detalhes da API, consulte a ListTables referência da API AWS SDK for Rust.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn add_item(client: &Client, item: Item, table: &String) ->
 Result<ItemOut, Error> {
    let user_av = AttributeValue::S(item.username);
    let type_av = AttributeValue::S(item.p_type);
    let age_av = AttributeValue::S(item.age);
    let first_av = AttributeValue::S(item.first);
    let last_av = AttributeValue::S(item.last);

    let request = client
        .put_item()
        .table_name(table)
        .item("username", user_av)
        .item("account_type", type_av)
        .item("age", age_av)
        .item("first_name", first_av)
        .item("last_name", last_av);

    println!("Executing request [{request:?}] to add item...");

    let resp = request.send().await?;
```

```
    let attributes = resp.attributes().unwrap();

    let username = attributes.get("username").cloned();
    let first_name = attributes.get("first_name").cloned();
    let last_name = attributes.get("last_name").cloned();
    let age = attributes.get("age").cloned();
    let p_type = attributes.get("p_type").cloned();

    println!(
        "Added user {:?}, {:?} {:?}, age {:?} as {:?} user",
        username, first_name, last_name, age, p_type
    );

    Ok(ItemOut {
        p_type,
        age,
        username,
        first_name,
        last_name,
    })
}
```

- Para obter detalhes da API, consulte a PutItem referência da API AWS SDK for Rust.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Encontre os filmes feitos no ano especificado.

```
pub async fn movies_in_year(
    client: &Client,
    table_name: &str,
```

```
    year: u16,
) -> Result<Vec<Movie>, MovieError> {
    let results = client
        .query()
        .table_name(table_name)
        .key_condition_expression("#yr = :yyyy")
        .expression_attribute_names("#yr", "year")
        .expression_attribute_values(":yyyy", AttributeValue::N(year.to_string()))
        .send()
        .await?;

    if let Some(items) = results.items {
        let movies = items.iter().map(|v| v.into()).collect();
        Ok(movies)
    } else {
        Ok(vec![])
    }
}
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Rust.

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_items(client: &Client, table: &str, page_size: Option<i32>) ->
 Result<(), Error> {
    let page_size = page_size.unwrap_or(10);
    let items: Result<Vec<_>, _> = client
        .scan()
        .table_name(table)
        .limit(page_size)
        .into_paginator()
```

```
        .items()
        .send()
        .collect()
        .await;

    println!("Items in table (up to {page_size}):");
    for item in items? {
        println!("   {:?}", item);
    }

    Ok(())
}
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Rust.

Cenários

Conecte-se a uma instância local

O exemplo de código a seguir mostra como substituir uma URL de endpoint para conectar-se a uma implantação de desenvolvimento local do DynamoDB e de um SDK. AWS

Para obter mais informações, consulte DynamoDB Local.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// Lists your tables from a local DynamoDB instance by setting the SDK Config's
/// endpoint_url and test_credentials.
#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();

    let config = aws_config::defaults(aws_config::BehaviorVersion::latest())
        .test_credentials()
```

```
        // DynamoDB run locally uses port 8000 by default.
        .endpoint_url("http://localhost:8000")
        .load()
        .await;
    let dynamodb_local_config =
 aws_sdk_dynamodb::config::Builder::from(&config).build();

    let client = aws_sdk_dynamodb::Client::from_conf(dynamodb_local_config);

    let list_resp = client.list_tables().send().await;
    match list_resp {
        Ok(resp) => {
            println!("Found {} tables", resp.table_names().len());
            for name in resp.table_names() {
                println!("  {}", name);
            }
        }
        Err(err) => eprintln!("Failed to list local dynamodb tables: {err:?}"),
    }
}
```

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obter um item executando uma instrução SELECT.
- Adicionar um item executando uma instrução INSERT.
- Atualizar um item executando a instrução UPDATE.
- Excluir um item executando uma instrução DELETE.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_table(
    client: &Client,
    table: &str,
    key: &str,
) -> Result<(), SdkError<CreateTableError>> {
    let ad = AttributeDefinition::builder()
        .attribute_name(key)
        .attribute_type(ScalarAttributeType::S)
        .build()
        .expect("creating AttributeDefinition");

    let ks = KeySchemaElement::builder()
        .attribute_name(key)
        .key_type(KeyType::Hash)
        .build()
        .expect("creating KeySchemaElement");
```

```
    match client
        .create_table()
        .table_name(table)
        .key_schema(ks)
        .attribute_definitions(ad)
        .billing_mode(BillingMode::PayPerRequest)
        .send()
        .await
    {
        Ok(_) => Ok(()),
        Err(e) => Err(e),
    }
}

async fn add_item(client: &Client, item: Item) -> Result<(),
 SdkError<ExecuteStatementError>> {
    match client
        .execute_statement()
        .statement(format!(
            r#"INSERT INTO "{}" VALUE {{
                "{}": ?,
                "acount_type": ?,
                "age": ?,
                "first_name": ?,
                "last_name": ?
        }} "#,
            item.table, item.key
        ))
        .set_parameters(Some(vec![
            AttributeValue::S(item.utype),
            AttributeValue::S(item.age),
            AttributeValue::S(item.first_name),
            AttributeValue::S(item.last_name),
        ]))
        .send()
        .await
    {
        Ok(_) => Ok(()),
        Err(e) => Err(e),
    }
}

async fn query_item(client: &Client, item: Item) -> bool {
```

```
    match client
        .execute_statement()
        .statement(format!(
            r#"SELECT * FROM "{}" WHERE "{}" = ?"#,
            item.table, item.key
        ))
        .set_parameters(Some(vec![AttributeValue::S(item.value)]))
        .send()
        .await
    {
        Ok(resp) => {
            if !resp.items().is_empty() {
                println!("Found a matching entry in the table:");
                println!("{:?}", resp.items.unwrap_or_default().pop());
                true
            } else {
                println!("Did not find a match.");
                false
            }
        }
        Err(e) => {
            println!("Got an error querying table:");
            println!("{}", e);
            process::exit(1);
        }
    }
}

async fn remove_item(client: &Client, table: &str, key: &str, value: String) ->
 Result<(), Error> {
    client
        .execute_statement()
        .statement(format!(r#"DELETE FROM "{table}" WHERE "{key}" = ?"#))
        .set_parameters(Some(vec![AttributeValue::S(value)]))
        .send()
        .await?;

    println!("Deleted item.");

    Ok(())
}

async fn remove_table(client: &Client, table: &str) -> Result<(), Error> {
    client.delete_table().table_name(table).send().await?;
```

```
    Ok(())
}
```

- Para obter detalhes da API, consulte a ExecuteStatement referência da API AWS SDK for Rust.

Salvar o EXIF e outras informações de imagem

O exemplo de código a seguir mostra como:

- Obter informações de EXIF de um arquivo JPG, JPEG ou PNG.
- Fazer upload do arquivo de imagem para um bucket do Amazon S3.
- Usar o Amazon Rekognition para identificar os três principais atributos (rótulos) no arquivo.
- Adicionar as informações de EXIF e rótulo a uma tabela do Amazon DynamoDB na região.

SDK para Rust

Obtenha informações de EXIF de um arquivo JPG, JPEG ou PNG, faça upload do arquivo de imagem para um bucket do Amazon S3, use o Amazon Rekognition para identificar os três principais atributos (rótulos no Amazon Rekognition) no arquivo e adicione as informações de EXIF e de rótulo a uma tabela do Amazon DynamoDB na região.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Exemplos sem servidor

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<Event>) ->Result<(), Error> {

    let records = &event.payload.records;
    tracing::info!("event payload: {:?}",records);
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }
```

```
    for record in records{
        log_dynamo_dbrecord(record);
    }

    tracing::info!("Dynamo db records processed");

    // Prepare the response
    Ok(())

}

fn log_dynamo_dbrecord(record: &EventRecord)-> Result<(), Error>{
    tracing::info!("EventId: {}", record.event_id);
    tracing::info!("EventName: {}", record.event_name);
    tracing::info!("DynamoDB Record: {:?}", record.change );
    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Rust.

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord, StreamRecord},
    streams::{DynamoDbBatchItemFailure, DynamoDbEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Process the stream record
fn process_record(record: &EventRecord) -> Result<(), Error> {
    let stream_record: &StreamRecord = &record.change;

    // process your stream record here...
    tracing::info!("Data: {:?}", stream_record);

    Ok(())
}

/// Main Lambda handler here...
async fn function_handler(event: LambdaEvent<Event>) ->
 Result<DynamoDbEventResponse, Error> {
    let mut response = DynamoDbEventResponse {
        batch_item_failures: vec![],
    };

    let records = &event.payload.records;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in records {
        tracing::info!("EventId: {}", record.event_id);
```

```
        // Couldn't find a sequence number
        if record.change.sequence_number.is_none() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: Some("".to_string()),
            });
            return Ok(response);
        }

        // Process your record here...
        if process_record(record).is_err() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: record.change.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed item
 onwards. */
            return Ok(response);
        }
    }

    tracing::info!("Successfully processed {} record(s)", records.len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Rust

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Rust.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

Exemplos do Amazon EBS usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon EBS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CompleteSnapshot`

O código de exemplo a seguir mostra como usar `CompleteSnapshot`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn finish(client: &Client, id: &str) -> Result<(), Error> {
    client
        .complete_snapshot()
        .changed_blocks_count(2)
        .snapshot_id(id)
        .send()
        .await?;

    println!("Snapshot ID {}", id);
    println!("The state is 'completed' when all of the modified blocks have been
 transferred to Amazon S3.");
    println!("Use the get-snapshot-state code example to get the state of the
 snapshot.");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CompleteSnapshot referência da API AWS SDK for Rust.

`PutSnapshotBlock`

O código de exemplo a seguir mostra como usar `PutSnapshotBlock`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn add_block(
    client: &Client,
    id: &str,
    idx: usize,
    block: Vec<u8>,
    checksum: &str,
) -> Result<(), Error> {
    client
        .put_snapshot_block()
        .snapshot_id(id)
        .block_index(idx as i32)
        .block_data(ByteStream::from(block))
        .checksum(checksum)
        .checksum_algorithm(ChecksumAlgorithm::ChecksumAlgorithmSha256)
        .data_length(EBS_BLOCK_SIZE as i32)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a PutSnapshotBlock referência da API AWS SDK for Rust.

`StartSnapshot`

O código de exemplo a seguir mostra como usar `StartSnapshot`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn start(client: &Client, description: &str) -> Result<String, Error> {
    let snapshot = client
        .start_snapshot()
        .description(description)
```

```
        .encrypted(false)
        .volume_size(1)
        .send()
        .await?;

    Ok(snapshot.snapshot_id.unwrap())
}
```

- Para obter detalhes da API, consulte a StartSnapshot referência da API AWS SDK for Rust.

Exemplos do Amazon EC2 usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon EC2.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids: Vec<String>)
 {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSecurityGroups referência da API AWS SDK for Rust.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.
- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

A EC2InstanceScenario implementação contém lógica para executar o exemplo como um todo.

```
//! Scenario that uses the AWS SDK for Rust (the SDK) with Amazon Elastic Compute
 Cloud
//! (Amazon EC2) to do the following:
//!
//! * Create a key pair that is used to secure SSH communication between your
 computer and
//!   an EC2 instance.
//! * Create a security group that acts as a virtual firewall for your EC2 instances
 to
//!   control incoming and outgoing traffic.
//! * Find an Amazon Machine Image (AMI) and a compatible instance type.
//! * Create an instance that is created from the instance type and AMI you select,
 and
//!   is configured to use the security group and key pair created in this example.
//! * Stop and restart the instance.
```

```
//! * Create an Elastic IP address and associate it as a consistent IP address for
 your instance.
//! * Connect to your instance with SSH, using both its public IP address and your
 Elastic IP
//!   address.
//! * Clean up all of the resources created by this example.

use std::net::Ipv4Addr;

use crate::{
    ec2::{EC2Error, EC2},
    getting_started::{key_pair::KeyPairManager, util::Util},
    ssm::SSM,
};
use aws_sdk_ssm::types::Parameter;

use super::{
    elastic_ip::ElasticIpManager, instance::InstanceManager,
 security_group::SecurityGroupManager,
    util::ScenarioImage,
};

pub struct Ec2InstanceScenario {
    ec2: EC2,
    ssm: SSM,
    util: Util,
    key_pair_manager: KeyPairManager,
    security_group_manager: SecurityGroupManager,
    instance_manager: InstanceManager,
    elastic_ip_manager: ElasticIpManager,
}

impl Ec2InstanceScenario {
    pub fn new(ec2: EC2, ssm: SSM, util: Util) -> Self {
        Ec2InstanceScenario {
            ec2,
            ssm,
            util,
            key_pair_manager: Default::default(),
            security_group_manager: Default::default(),
            instance_manager: Default::default(),
            elastic_ip_manager: Default::default(),
        }
    }
```

```
    pub async fn run(&mut self) -> Result<(), EC2Error> {
        self.create_and_list_key_pairs().await?;
        self.create_security_group().await?;
        self.create_instance().await?;
        self.stop_and_start_instance().await?;
        self.associate_elastic_ip().await?;
        self.stop_and_start_instance().await?;
        Ok(())
    }

    /// 1. Creates an RSA key pair and saves its private key data as a .pem file in
secure
    ///    temporary storage. The private key data is deleted after the example
completes.
    /// 2. Optionally, lists the first five key pairs for the current account.
    pub async fn create_and_list_key_pairs(&mut self) -> Result<(), EC2Error> {
        println!( "Let's create an RSA key pair that you can be use to securely
connect to your EC2 instance.");

        let key_name = self.util.prompt_key_name()?;

        self.key_pair_manager
            .create(&self.ec2, &self.util, key_name)
            .await?;

        println!(
            "Created a key pair {} and saved the private key to {:?}.",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .ok_or_else(|| EC2Error::new("No key name after creating key"))?,
            self.key_pair_manager
                .key_file_path()
                .ok_or_else(|| EC2Error::new("No key file after creating key"))?
        );

        if self.util.should_list_key_pairs()? {
            for pair in self.key_pair_manager.list(&self.ec2).await? {
                println!(
                    "Found {:?} key {} with fingerprint:\t{:?}",
                    pair.key_type(),
                    pair.key_name().unwrap_or("Unknown"),
                    pair.key_fingerprint()
```

```
                );
            }
        }

        Ok(())
    }

    /// 1. Creates a security group for the default VPC.
    /// 2. Adds an inbound rule to allow SSH. The SSH rule allows only
    ///    inbound traffic from the current computer’s public IPv4 address.
    /// 3. Displays information about the security group.
    ///
    /// This function uses <http://checkip.amazonaws.com> to get the current public
IP
    /// address of the computer that is running the example. This method works in
most
    /// cases. However, depending on how your computer connects to the internet, you
    /// might have to manually add your public IP address to the security group by
using
    /// the AWS Management Console.
    pub async fn create_security_group(&mut self) -> Result<(), EC2Error> {
        println!("Let's create a security group to manage access to your
instance.");
        let group_name = self.util.prompt_security_group_name()?;

        self.security_group_manager
            .create(
                &self.ec2,
                &group_name,
                "Security group for example: get started with instances.",
            )
            .await?;

        println!(
            "Created security group {} in your default VPC {}.",
            self.security_group_manager.group_name(),
            self.security_group_manager
                .vpc_id()
                .unwrap_or("(unknown vpc)")
        );

        let check_ip = self.util.do_get("https://checkip.amazonaws.com").await?;
        let current_ip_address: Ipv4Addr = check_ip.trim().parse().map_err(|e| {
            EC2Error::new(format!(
```

```
                "Failed to convert response {} to IP Address: {e:?}",
                check_ip
            ))
        })?;

        println!("Your public IP address seems to be {current_ip_address}");
        if self.util.should_add_to_security_group() {
            match self
                .security_group_manager
                .authorize_ingress(&self.ec2, current_ip_address)
                .await
            {
                Ok(_) => println!("Security group rules updated"),
                Err(err) => eprintln!("Couldn't update security group rules:
{err:?}"),
            }
        }
        println!("{}", self.security_group_manager);

        Ok(())
    }

    /// 1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager. Specifying
the
    ///    '/aws/service/ami-amazon-linux-latest' path returns only the latest AMIs.
    /// 2. Gets and displays information about the available AMIs and lets you
select one.
    /// 3. Gets a list of instance types that are compatible with the selected AMI
and
    ///    lets you select one.
    /// 4. Creates an instance with the previously created key pair and security
group,
    ///    and the selected AMI and instance type.
    /// 5. Waits for the instance to be running and then displays its information.
    pub async fn create_instance(&mut self) -> Result<(), EC2Error> {
        let ami = self.find_image().await?;

        let instance_types = self
            .ec2
            .list_instance_types(&ami.0)
            .await
            .map_err(|e| e.add_message("Could not find instance types"))?;
        println!(
```

```
            "There are several instance types that support the {} architecture of
the image.",
            ami.0
                .architecture
                .as_ref()
                .ok_or_else(|| EC2Error::new(format!("Missing architecture in {:?}",
ami.0)))?
        );
        let instance_type = self.util.select_instance_type(instance_types)?;

        println!("Creating your instance and waiting for it to start...");
        self.instance_manager
            .create(
                &self.ec2,
                ami.0
                    .image_id()
                    .ok_or_else(|| EC2Error::new("Could not find image ID"))?,
                instance_type,
                self.key_pair_manager.key_pair(),
                self.security_group_manager
                    .security_group()
                    .map(|sg| vec![sg])
                    .ok_or_else(|| EC2Error::new("Could not find security group"))?,
            )
            .await
            .map_err(|e| e.add_message("Scenario failed to create instance"))?;

        while let Err(err) = self
            .ec2
            .wait_for_instance_ready(self.instance_manager.instance_id(), None)
            .await
        {
            println!("{err}");
            if !self.util.should_continue_waiting() {
                return Err(err);
            }
        }

        println!("Your instance is ready:\n{}", self.instance_manager);

        self.display_ssh_info();

        Ok(())
    }
```

```
    async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
        let params: Vec<Parameter> = self
            .ssm
            .list_path("/aws/service/ami-amazon-linux-latest")
            .await
            .map_err(|e| e.add_message("Could not find parameters for available
images"))?
            .into_iter()
            .filter(|param| param.name().is_some_and(|name| name.contains("amzn2")))
            .collect();
        let amzn2_images: Vec<ScenarioImage> = self
            .ec2
            .list_images(params)
            .await
            .map_err(|e| e.add_message("Could not find images"))?
            .into_iter()
            .map(ScenarioImage::from)
            .collect();
        println!("We will now create an instance from an Amazon Linux 2 AMI");
        let ami = self.util.select_scenario_image(amzn2_images)?;
        Ok(ami)
    }

    // 1. Stops the instance and waits for it to stop.
    // 2. Starts the instance and waits for it to start.
    // 3. Displays information about the instance.
    // 4. Displays an SSH connection string. When an Elastic IP address is
associated
    //    with the instance, the IP address stays consistent when the instance stops
    //    and starts.
    pub async fn stop_and_start_instance(&self) -> Result<(), EC2Error> {
        println!("Let's stop and start your instance to see what changes.");
        println!("Stopping your instance and waiting until it's stopped...");
        self.instance_manager.stop(&self.ec2).await?;
        println!("Your instance is stopped. Restarting...");
        self.instance_manager.start(&self.ec2).await?;
        println!("Your instance is running.");
        println!("{}", self.instance_manager);
        if self.elastic_ip_manager.public_ip() == "0.0.0.0" {
            println!("Every time your instance is restarted, its public IP address
changes.");
        } else {
            println!(
```

```
                "Because you have associated an Elastic IP with your instance, you
can connect by using a consistent IP address after the instance restarts."
            );
        }
        self.display_ssh_info();
        Ok(())
    }

    /// 1. Allocates an Elastic IP address and associates it with the instance.
    /// 2. Displays an SSH connection string that uses the Elastic IP address.
    async fn associate_elastic_ip(&mut self) -> Result<(), EC2Error> {
        self.elastic_ip_manager.allocate(&self.ec2).await?;
        println!(
            "Allocated static Elastic IP address: {}",
            self.elastic_ip_manager.public_ip()
        );

        self.elastic_ip_manager
            .associate(&self.ec2, self.instance_manager.instance_id())
            .await?;
        println!("Associated your Elastic IP with your instance.");
        println!("You can now use SSH to connect to your instance by using the
Elastic IP.");
        self.display_ssh_info();
        Ok(())
    }

    /// Displays an SSH connection string that can be used to connect to a running
    /// instance.
    fn display_ssh_info(&self) {
        let ip_addr = if self.elastic_ip_manager.has_allocation() {
            self.elastic_ip_manager.public_ip()
        } else {
            self.instance_manager.instance_ip()
        };
        let key_file_path = self.key_pair_manager.key_file_path().unwrap();
        println!("To connect, open another command prompt and run the following
command:");
        println!("\nssh -i {} ec2-user@{ip_addr}\n", key_file_path.display());
        let _ = self.util.enter_to_continue();
    }

    /// 1. Disassociate and delete the previously created Elastic IP.
    /// 2. Terminate the previously created instance.
```

```
    /// 3. Delete the previously created security group.
    /// 4. Delete the previously created key pair.
    pub async fn clean_up(self) {
        println!("Let's clean everything up. This example created these
 resources:");
        println!(
            "\tKey pair: {}",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .unwrap_or("(unknown key pair)")
        );
        println!(
            "\tSecurity group: {}",
            self.security_group_manager.group_name()
        );
        println!(
            "\tInstance: {}",
            self.instance_manager.instance_display_name()
        );
        if self.util.should_clean_resources() {
            if let Err(err) = self.elastic_ip_manager.remove(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.instance_manager.delete(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.security_group_manager.delete(&self.ec2).await {
                eprintln!("{err}");
            }
            if let Err(err) = self.key_pair_manager.delete(&self.ec2,
 &self.util).await {
                eprintln!("{err}");
            }
        } else {
            println!("Ok, not cleaning up any resources!");
        }
    }
}

pub async fn run(mut scenario: Ec2InstanceScenario) {
    println!
("--------------------------------------------------------------------------------");
    println!(
```

```
        "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started with
 instances demo."
    );
    println!
("------------------------------------------------------------------------------");

    if let Err(err) = scenario.run().await {
        eprintln!("There was an error running the scenario: {err}")
    }

    println!
("------------------------------------------------------------------------------");

    scenario.clean_up().await;

    println!("Thanks for running!");
    println!
("------------------------------------------------------------------------------");
}
```

O struct EC2Impl serve como um ponto de controle automático para testes e suas funções envolvem as chamadas do SDK do EC2.

```
use std::{net::Ipv4Addr, time::Duration};

use aws_sdk_ec2::{
    client::Waiters,
    error::ProvideErrorMetadata,
    operation::{
        allocate_address::AllocateAddressOutput,
 associate_address::AssociateAddressOutput,
    },
    types::{
        DomainType, Filter, Image, Instance, InstanceType, IpPermission, IpRange,
 KeyPairInfo,
        SecurityGroup, Tag,
    },
    Client as EC2Client,
};
use aws_sdk_ssm::types::Parameter;
use aws_smithy_runtime_api::client::waiters::error::WaiterError;
```

```
#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use EC2Impl as EC2;

#[cfg(test)]
pub use MockEC2Impl as EC2;

#[derive(Clone)]
pub struct EC2Impl {
    pub client: EC2Client,
}

#[cfg_attr(test, automock)]
impl EC2Impl {
    pub fn new(client: EC2Client) -> Self {
        EC2Impl { client }
    }

    pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
 String), EC2Error> {
        tracing::info!("Creating key pair {name}");
        let output = self.client.create_key_pair().key_name(name).send().await?;
        let info = KeyPairInfo::builder()
            .set_key_name(output.key_name)
            .set_key_fingerprint(output.key_fingerprint)
            .set_key_pair_id(output.key_pair_id)
            .build();
        let material = output
            .key_material
            .ok_or_else(|| EC2Error::new("Create Key Pair has no key material"))?;
        Ok((info, material))
    }

    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }

    pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
        let key_name: String = key_name.into();
        tracing::info!("Deleting key pair {key_name}");
```

```
        self.client
            .delete_key_pair()
            .key_name(key_name)
            .send()
            .await?;
        Ok(())
    }

    pub async fn create_security_group(
        &self,
        name: &str,
        description: &str,
    ) -> Result<SecurityGroup, EC2Error> {
        tracing::info!("Creating security group {name}");
        let create_output = self
            .client
            .create_security_group()
            .group_name(name)
            .description(description)
            .send()
            .await
            .map_err(EC2Error::from)?;

        let group_id = create_output
            .group_id
            .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

        let group = self
            .describe_security_group(&group_id)
            .await?
            .ok_or_else(|| {
                EC2Error::new(format!("Could not find security group with id
{group_id}"))
            })?;

        tracing::info!("Created security group {name} as {group_id}");

        Ok(group)
    }

    /// Find a single security group, by ID. Returns Err if multiple groups are
found.
    pub async fn describe_security_group(
```

```
        &self,
        group_id: &str,
    ) -> Result<Option<SecurityGroup>, EC2Error> {
        let group_id: String = group_id.into();
        let describe_output = self
            .client
            .describe_security_groups()
            .group_ids(&group_id)
            .send()
            .await?;

        let mut groups = describe_output.security_groups.unwrap_or_default();

        match groups.len() {
            0 => Ok(None),
            1 => Ok(Some(groups.remove(0))),
            _ => Err(EC2Error::new(format!(
                "Expected single group for {group_id}"
            ))),
        }
    }

    /// Add an ingress rule to a security group explicitly allowing IPv4 address
    /// as {ip}/32 over TCP port 22.
    pub async fn authorize_security_group_ssh_ingress(
        &self,
        group_id: &str,
        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
                            .build()
```

```
                    })
                    .collect(),
            ))
            .send()
            .await?;
        Ok(())
    }

    pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Deleting security group {group_id}");
        self.client
            .delete_security_group()
            .group_id(group_id)
            .send()
            .await?;
        Ok(())
    }

    pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
        let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
        let output = self
            .client
            .describe_images()
            .set_image_ids(Some(image_ids))
            .send()
            .await?;

        let images = output.images.unwrap_or_default();
        if images.is_empty() {
            Err(EC2Error::new("No images for selected AMIs"))
        } else {
            Ok(images)
        }
    }

    /// List instance types that match an image's architecture and are free tier
eligible.
    pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
        let architecture = format!(
            "{}",
            image.architecture().ok_or_else(|| EC2Error::new(format!(
```

```
                "Image {:?} does not have a listed architecture",
                image.image_id()
            )))?
        );
        let free_tier_eligible_filter = Filter::builder()
            .name("free-tier-eligible")
            .values("false")
            .build();
        let supported_architecture_filter = Filter::builder()
            .name("processor-info.supported-architecture")
            .values(architecture)
            .build();
        let response = self
            .client
            .describe_instance_types()
            .filters(free_tier_eligible_filter)
            .filters(supported_architecture_filter)
            .send()
            .await?;

        Ok(response
            .instance_types
            .unwrap_or_default()
            .into_iter()
            .filter_map(|iti| iti.instance_type)
            .collect())
    }

    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
```

```
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }
```

```
    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
        let response = self
            .client
            .describe_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        let instance = response
            .reservations()
            .first()
            .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
            .instances()
            .first()
            .ok_or_else(|| {
                EC2Error::new(format!("No instances in reservation for
{instance_id}"))
            })?;

        Ok(instance.clone())
```

```
    }

    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }

    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }

    pub async fn reboot_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Rebooting instance {instance_id}");

        self.client
            .reboot_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        Ok(())
    }
```

```
    pub async fn wait_for_instance_stopped(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_stopped()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to stop.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn delete_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Deleting instance with id {instance_id}");
        self.stop_instance(instance_id).await?;
        self.client
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await?;
        self.wait_for_instance_terminated(instance_id).await?;
        tracing::info!("Terminated instance with id {instance_id}");
        Ok(())
    }

    async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
        self.client
            .wait_until_instance_terminated()
            .instance_ids(instance_id)
            .wait(Duration::from_secs(60))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to terminate.",
                    exceeded.max_wait().as_secs(),
```

```
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
        self.client
            .allocate_address()
            .domain(DomainType::Vpc)
            .send()
            .await
            .map_err(EC2Error::from)
    }

    pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
        self.client
            .release_address()
            .allocation_id(allocation_id)
            .send()
            .await?;
        Ok(())
    }

    pub async fn associate_ip_address(
        &self,
        allocation_id: &str,
        instance_id: &str,
    ) -> Result<AssociateAddressOutput, EC2Error> {
        let response = self
            .client
            .associate_address()
            .allocation_id(allocation_id)
            .instance_id(instance_id)
            .send()
            .await?;
        Ok(response)
    }

    pub async fn disassociate_ip_address(&self, association_id: &str) -> Result<(),
EC2Error> {
        self.client
```

```
            .disassociate_address()
            .association_id(association_id)
            .send()
            .await?;
        Ok(())
    }
}

#[derive(Debug)]
pub struct EC2Error(String);
impl EC2Error {
    pub fn new(value: impl Into<String>) -> Self {
        EC2Error(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        EC2Error(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: ProvideErrorMetadata> From<T> for EC2Error {
    fn from(value: T) -> Self {
        EC2Error(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for EC2Error {}

impl std::fmt::Display for EC2Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

O struct SSM serve como um ponto de controle automático para testes e suas funções envolvem as chamadas do SDK do SSM.

```
use aws_sdk_ssm::{types::Parameter, Client};
use aws_smithy_async::future::pagination_stream::TryFlatMap;

use crate::ec2::EC2Error;

#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use SSMImpl as SSM;

#[cfg(test)]
pub use MockSSMImpl as SSM;

pub struct SSMImpl {
    inner: Client,
}

#[cfg_attr(test, automock)]
impl SSMImpl {
    pub fn new(inner: Client) -> Self {
        SSMImpl { inner }
    }

    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error> {
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
        .await;
        // Fail on the first error
        let params = maybe_params
```

```
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
    }
}
```

O cenário usa várias estruturas no estilo “Gerenciador” para lidar com o acesso aos recursos que são criados e excluídos em todo o cenário.

```
use aws_sdk_ec2::operation::{
    allocate_address::AllocateAddressOutput,
 associate_address::AssociateAddressOutput,
};

use crate::ec2::{EC2Error, EC2};

/// ElasticIpManager tracks the lifecycle of a public IP address, including its
/// allocation from the global pool and association with a specific instance.
#[derive(Debug, Default)]
pub struct ElasticIpManager {
    elastic_ip: Option<AllocateAddressOutput>,
    association: Option<AssociateAddressOutput>,
}

impl ElasticIpManager {
    pub fn has_allocation(&self) -> bool {
        self.elastic_ip.is_some()
    }

    pub fn public_ip(&self) -> &str {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(addr) = allocation.public_ip() {
                return addr;
            }
        }
        "0.0.0.0"
    }

    pub async fn allocate(&mut self, ec2: &EC2) -> Result<(), EC2Error> {
        let allocation = ec2.allocate_ip_address().await?;
        self.elastic_ip = Some(allocation);
```

```
        Ok(())
    }

    pub async fn associate(&mut self, ec2: &EC2, instance_id: &str) -> Result<(),
 EC2Error> {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                let association = ec2.associate_ip_address(allocation_id,
 instance_id).await?;
                self.association = Some(association);
                return Ok(());
            }
        }
        Err(EC2Error::new("No ip address allocation to associate"))
    }

    pub async fn remove(mut self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(association) = &self.association {
            if let Some(association_id) = association.association_id() {
                ec2.disassociate_ip_address(association_id).await?;
            }
        }
        self.association = None;
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                ec2.deallocate_ip_address(allocation_id).await?;
            }
        }
        self.elastic_ip = None;
        Ok(())
    }
}


use std::fmt::Display;

use aws_sdk_ec2::types::{Instance, InstanceType, KeyPairInfo, SecurityGroup};

use crate::ec2::{EC2Error, EC2};

/// InstanceManager wraps the lifecycle of an EC2 Instance.
#[derive(Debug, Default)]
pub struct InstanceManager {
    instance: Option<Instance>,
```

```
}

impl InstanceManager {
    pub fn instance_id(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(id) = instance.instance_id() {
                return id;
            }
        }
        "Unknown"
    }

    pub fn instance_name(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(tag) = instance.tags().iter().find(|e| e.key() ==
 Some("Name")) {
                if let Some(value) = tag.value() {
                    return value;
                }
            }
        }
        "Unknown"
    }

    pub fn instance_ip(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(public_ip_address) = instance.public_ip_address() {
                return public_ip_address;
            }
        }
        "0.0.0.0"
    }

    pub fn instance_display_name(&self) -> String {
        format!("{} ({})", self.instance_name(), self.instance_id())
    }

    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        image_id: &str,
        instance_type: InstanceType,
```

```
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }

    /// Start the managed EC2 instance, if present.
    pub async fn start(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.start_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    /// Stop the managed EC2 instance, if present.
    pub async fn stop(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.stop_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.reboot_instance(self.instance_id()).await?;
            ec2.wait_for_instance_stopped(self.instance_id(), None)
                .await?;
            ec2.wait_for_instance_ready(self.instance_id(), None)
                .await?;
        }
        Ok(())
    }

    /// Terminate and delete the managed EC2 instance, if present.
    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.delete_instance(self.instance_id()).await?;
        }
        Ok(())
```

```
    }
}

impl Display for InstanceManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        if let Some(instance) = &self.instance {
            writeln!(f, "\tID: {}", instance.instance_id().unwrap_or("(Unknown)"))?;
            writeln!(
                f,
                "\tImage ID: {}",
                instance.image_id().unwrap_or("(Unknown)")
            )?;
            writeln!(
                f,
                "\tInstance type: {}",
                instance
                    .instance_type()
                    .map(|it| format!("{it}"))
                    .unwrap_or("(Unknown)".to_string())
            )?;
            writeln!(
                f,
                "\tKey name: {}",
                instance.key_name().unwrap_or("(Unknown)")
            )?;
            writeln!(f, "\tVPC ID: {}", instance.vpc_id().unwrap_or("(Unknown)"))?;
            writeln!(
                f,
                "\tPublic IP: {}",
                instance.public_ip_address().unwrap_or("(Unknown)")
            )?;
            let instance_state = instance
                .state
                .as_ref()
                .map(|is| {
                    is.name()
                        .map(|isn| format!("{isn}"))
                        .unwrap_or("(Unknown)".to_string())
                })
                .unwrap_or("(Unknown)".to_string());
            writeln!(f, "\tState: {instance_state}")?;
        } else {
            writeln!(f, "\tNo loaded instance")?;
        }
```

```
        Ok(())
    }
}


use std::{env, path::PathBuf};

use aws_sdk_ec2::types::KeyPairInfo;

use crate::ec2::{EC2Error, EC2};

use super::util::Util;

/// KeyPairManager tracks a KeyPairInfo and the path the private key has been
/// written to, if it's been created.
#[derive(Debug)]
pub struct KeyPairManager {
    key_pair: KeyPairInfo,
    key_file_path: Option<PathBuf>,
    key_file_dir: PathBuf,
}

impl KeyPairManager {
    pub fn new() -> Self {
        Self::default()
    }

    pub fn key_pair(&self) -> &KeyPairInfo {
        &self.key_pair
    }

    pub fn key_file_path(&self) -> Option<&PathBuf> {
        self.key_file_path.as_ref()
    }

    pub fn key_file_dir(&self) -> &PathBuf {
        &self.key_file_dir
    }

    /// Creates a key pair that can be used to securely connect to an EC2 instance.
    /// The returned key pair contains private key information that cannot be
 retrieved
    /// again. The private key data is stored as a .pem file.
    ///
```

```
    /// :param key_name: The name of the key pair to create.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        util: &Util,
        key_name: String,
    ) -> Result<KeyPairInfo, EC2Error> {
        let (key_pair, material) =
 ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
            self.key_pair =
 KeyPairInfo::builder().key_name(key_name.clone()).build();
            e.add_message(format!("Couldn't create key {key_name}"))
        })?;

        let path = self.key_file_dir.join(format!("{key_name}.pem"));

        // Save the key_pair information immediately, so it can get cleaned up if
 write_secure fails.
        self.key_file_path = Some(path.clone());
        self.key_pair = key_pair.clone();

        util.write_secure(&key_name, &path, material)?;

        Ok(key_pair)
    }

    pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
        if let Some(key_name) = self.key_pair.key_name() {
            ec2.delete_key_pair(key_name).await?;
            if let Some(key_path) = self.key_file_path() {
                if let Err(err) = util.remove(key_path) {
                    eprintln!("Failed to remove {key_path:?} ({err:?})");
                }
            }
        }
        Ok(())
    }

    pub async fn list(&self, ec2: &EC2) -> Result<Vec<KeyPairInfo>, EC2Error> {
        ec2.list_key_pair().await
    }
}

impl Default for KeyPairManager {
```

```
    fn default() -> Self {
        KeyPairManager {
            key_pair: KeyPairInfo::builder().build(),
            key_file_path: Default::default(),
            key_file_dir: env::temp_dir(),
        }
    }
}


use std::net::Ipv4Addr;

use aws_sdk_ec2::types::SecurityGroup;

use crate::ec2::{EC2Error, EC2};

/// SecurityGroupManager tracks the lifecycle of a SecurityGroup for an instance,
/// including adding a rule to allow SSH from a public IP address.
#[derive(Debug, Default)]
pub struct SecurityGroupManager {
    group_name: String,
    group_description: String,
    security_group: Option<SecurityGroup>,
}

impl SecurityGroupManager {
    pub async fn create(
        &mut self,
        ec2: &EC2,
        group_name: &str,
        group_description: &str,
    ) -> Result<(), EC2Error> {
        self.group_name = group_name.into();
        self.group_description = group_description.into();

        self.security_group = Some(
            ec2.create_security_group(group_name, group_description)
                .await
                .map_err(|e| e.add_message("Couldn't create security group"))?,
        );

        Ok(())
    }
```

```
    pub async fn authorize_ingress(&self, ec2: &EC2, ip_address: Ipv4Addr) ->
 Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.authorize_security_group_ssh_ingress(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
                vec![ip_address],
            )
            .await?;
        };

        Ok(())
    }

    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.delete_security_group(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
            )
            .await?;
        };

        Ok(())
    }

    pub fn group_name(&self) -> &str {
        &self.group_name
    }

    pub fn vpc_id(&self) -> Option<&str> {
        self.security_group.as_ref().and_then(|sg| sg.vpc_id())
    }

    pub fn security_group(&self) -> Option<&SecurityGroup> {
        self.security_group.as_ref()
    }
}

impl std::fmt::Display for SecurityGroupManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.security_group {
            Some(sg) => {
                writeln!(
```

```
                    f,
                    "Security group: {}",
                    sg.group_name().unwrap_or("(unknown group)")
                )?;
                writeln!(f, "\tID: {}", sg.group_id().unwrap_or("(unknown group
 id)"))?;
                writeln!(f, "\tVPC: {}", sg.vpc_id().unwrap_or("(unknown group
 vpc)"))?;
                if !sg.ip_permissions().is_empty() {
                    writeln!(f, "\tInbound Permissions:")?;
                    for permission in sg.ip_permissions() {
                        writeln!(f, "\t\t{permission:?}")?;
                    }
                }
                Ok(())
            }
            None => writeln!(f, "No security group loaded."),
        }
    }
}
```

O principal ponto de entrada para o cenário.

```
use ec2_code_examples::{
    ec2::EC2,
    getting_started::{
        scenario::{run, Ec2InstanceScenario},
        util::UtilImpl,
    },
    ssm::SSM,
};

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::load_from_env().await;
    let ec2 = EC2::new(aws_sdk_ec2::Client::new(&sdk_config));
    let ssm = SSM::new(aws_sdk_ssm::Client::new(&sdk_config));
    let util = UtilImpl {};
    let scenario = Ec2InstanceScenario::new(ec2, ssm, util);
    run(scenario).await;
```

```
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
       self.client
           .allocate_address()
           .domain(DomainType::Vpc)
           .send()
           .await
           .map_err(EC2Error::from)
   }
```

- Para obter detalhes da API, consulte a AllocateAddress referência da API AWS SDK for Rust.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn associate_ip_address(
       &self,
       allocation_id: &str,
       instance_id: &str,
   ) -> Result<AssociateAddressOutput, EC2Error> {
       let response = self
```

```
            .client
            .associate_address()
            .allocation_id(allocation_id)
            .instance_id(instance_id)
            .send()
            .await?;
        Ok(response)
    }
```

- Para obter detalhes da API, consulte a AssociateAddress referência da API AWS SDK for Rust.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// Add an ingress rule to a security group explicitly allowing IPv4 address
    /// as {ip}/32 over TCP port 22.
    pub async fn authorize_security_group_ssh_ingress(
        &self,
        group_id: &str,
        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
```

```
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
                            .build()
                    })
                    .collect(),
            ))
            .send()
            .await?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a AuthorizeSecurityGroupIngress referência da API AWS SDK for Rust.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Implementação do Rust que chama o create_key_pair do cliente do EC2 e extrai o material retornado.

```
    pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
String), EC2Error> {
        tracing::info!("Creating key pair {name}");
        let output = self.client.create_key_pair().key_name(name).send().await?;
        let info = KeyPairInfo::builder()
            .set_key_name(output.key_name)
            .set_key_fingerprint(output.key_fingerprint)
            .set_key_pair_id(output.key_pair_id)
```

```
            .build();
        let material = output
            .key_material
            .ok_or_else(|| EC2Error::new("Create Key Pair has no key material"))?;
        Ok((info, material))
    }
```

Uma função que chama create_key impl e salva com segurança a chave privada do PEM.

```
    /// Creates a key pair that can be used to securely connect to an EC2 instance.
    /// The returned key pair contains private key information that cannot be
retrieved
    /// again. The private key data is stored as a .pem file.
    ///
    /// :param key_name: The name of the key pair to create.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        util: &Util,
        key_name: String,
    ) -> Result<KeyPairInfo, EC2Error> {
        let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
            self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
            e.add_message(format!("Couldn't create key {key_name}"))
        })?;

        let path = self.key_file_dir.join(format!("{key_name}.pem"));

        // Save the key_pair information immediately, so it can get cleaned up if
write_secure fails.
        self.key_file_path = Some(path.clone());
        self.key_pair = key_pair.clone();

        util.write_secure(&key_name, &path, material)?;

        Ok(key_pair)
    }
```

- Para obter detalhes da API, consulte a CreateKeyPair referência da API AWS SDK for Rust.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn create_security_group(
        &self,
        name: &str,
        description: &str,
    ) -> Result<SecurityGroup, EC2Error> {
        tracing::info!("Creating security group {name}");
        let create_output = self
            .client
            .create_security_group()
            .group_name(name)
            .description(description)
            .send()
            .await
            .map_err(EC2Error::from)?;

        let group_id = create_output
            .group_id
            .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

        let group = self
            .describe_security_group(&group_id)
            .await?
            .ok_or_else(|| {
                EC2Error::new(format!("Could not find security group with id
{group_id}"))
            })?;

        tracing::info!("Created security group {name} as {group_id}");

        Ok(group)
```

```
    }
```

- Para obter detalhes da API, consulte a CreateSecurityGroup referência da API AWS SDK for Rust.

`CreateTags`

O código de exemplo a seguir mostra como usar `CreateTags`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Esse exemplo aplica a tag Name depois de criar uma instância.

```
    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
```

```
                .filter_map(|sg| sg.group_id.clone())
                .collect(),
        ))
        .min_count(1)
        .max_count(1)
        .send()
        .await?;

    if run_instances.instances().is_empty() {
        return Err(EC2Error::new("Failed to create instance"));
    }

    let instance_id = run_instances.instances()[0].instance_id().unwrap();
    let response = self
        .client
        .create_tags()
        .resources(instance_id)
        .tags(
            Tag::builder()
                .key("Name")
                .value("From SDK Examples")
                .build(),
        )
        .send()
        .await;

    match response {
        Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
        Err(err) => {
            tracing::info!("Error applying tags to {instance_id}: {err:?}");
            return Err(err.into());
        }
    }

    tracing::info!("Instance is created.");

    Ok(instance_id.to_string())
}
```

- Para obter detalhes da API, consulte a CreateTags referência da API AWS SDK for Rust.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envolva delete_key que também remove a chave PEM privada de apoio.

```
pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
    if let Some(key_name) = self.key_pair.key_name() {
        ec2.delete_key_pair(key_name).await?;
        if let Some(key_path) = self.key_file_path() {
            if let Err(err) = util.remove(key_path) {
                eprintln!("Failed to remove {key_path:?} ({err:?})");
            }
        }
    }
    Ok(())
}
```

```
pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
    let key_name: String = key_name.into();
    tracing::info!("Deleting key pair {key_name}");
    self.client
        .delete_key_pair()
        .key_name(key_name)
        .send()
        .await?;
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteKeyPair referência da API AWS SDK for Rust.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
       tracing::info!("Deleting security group {group_id}");
       self.client
           .delete_security_group()
           .group_id(group_id)
           .send()
           .await?;
       Ok(())
   }
```

- Para obter detalhes da API, consulte a DeleteSecurityGroup referência da API AWS SDK for Rust.

`DeleteSnapshot`

O código de exemplo a seguir mostra como usar `DeleteSnapshot`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
```

```
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteSnapshot referência da API AWS SDK for Rust.

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
       let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
       let output = self
           .client
           .describe_images()
           .set_image_ids(Some(image_ids))
           .send()
           .await?;

       let images = output.images.unwrap_or_default();
       if images.is_empty() {
           Err(EC2Error::new("No images for selected AMIs"))
       } else {
           Ok(images)
       }
   }
```

Usar a função list_images com o SSM para limitar com base em seu ambiente. Para obter mais detalhes sobre o SSM, consulte https://docs.aws.amazon.com/systems-manager/latest/userguide/example_ssm_GetParameters_section.html.

```
    async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
        let params: Vec<Parameter> = self
            .ssm
            .list_path("/aws/service/ami-amazon-linux-latest")
            .await
            .map_err(|e| e.add_message("Could not find parameters for available
images"))?
            .into_iter()
            .filter(|param| param.name().is_some_and(|name| name.contains("amzn2")))
            .collect();
        let amzn2_images: Vec<ScenarioImage> = self
            .ec2
            .list_images(params)
            .await
            .map_err(|e| e.add_message("Could not find images"))?
            .into_iter()
            .map(ScenarioImage::from)
            .collect();
        println!("We will now create an instance from an Amazon Linux 2 AMI");
        let ami = self.util.select_scenario_image(amzn2_images)?;
        Ok(ami)
    }
```

- Para obter detalhes da API, consulte a DescribeImages referência da API AWS SDK for Rust.

`DescribeInstanceStatus`

O código de exemplo a seguir mostra como usar `DescribeInstanceStatus`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_all_events(client: &Client) -> Result<(), Error> {
    let resp = client.describe_regions().send().await.unwrap();

    for region in resp.regions.unwrap_or_default() {
        let reg: &'static str = Box::leak(Box::from(region.region_name().unwrap()));
        let region_provider = RegionProviderChain::default_provider().or_else(reg);
        let config = aws_config::from_env().region(region_provider).load().await;
        let new_client = Client::new(&config);

        let resp = new_client.describe_instance_status().send().await;

        println!("Instances in region {}:", reg);
        println!();

        for status in resp.unwrap().instance_statuses() {
            println!(
                "  Events scheduled for instance ID: {}",
                status.instance_id().unwrap_or_default()
            );
            for event in status.events() {
                println!("    Event ID:     {}",
 event.instance_event_id().unwrap());
                println!("    Description:  {}", event.description().unwrap());
                println!("    Event code:   {}", event.code().unwrap().as_ref());
                println!();
            }
        }
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeInstanceStatus referência da API AWS SDK for Rust.

`DescribeInstanceTypes`

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// List instance types that match an image's architecture and are free tier
eligible.
   pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
       let architecture = format!(
           "{}",
           image.architecture().ok_or_else(|| EC2Error::new(format!(
               "Image {:?} does not have a listed architecture",
               image.image_id()
           )))?
       );
       let free_tier_eligible_filter = Filter::builder()
           .name("free-tier-eligible")
           .values("false")
           .build();
       let supported_architecture_filter = Filter::builder()
           .name("processor-info.supported-architecture")
           .values(architecture)
           .build();
       let response = self
           .client
           .describe_instance_types()
           .filters(free_tier_eligible_filter)
           .filters(supported_architecture_filter)
           .send()
           .await?;

       Ok(response
           .instance_types
           .unwrap_or_default()
           .into_iter()
           .filter_map(|iti| iti.instance_type)
           .collect())
   }
```

- Para obter detalhes da API, consulte a DescribeInstanceTypes referência da API AWS SDK for Rust.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recupere detalhes de uma instância do EC2.

```
   pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
       let response = self
           .client
           .describe_instances()
           .instance_ids(instance_id)
           .send()
           .await?;

       let instance = response
           .reservations()
           .first()
           .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
           .instances()
           .first()
           .ok_or_else(|| {
               EC2Error::new(format!("No instances in reservation for
{instance_id}"))
           })?;

       Ok(instance.clone())
```

```
    }
```

Depois de criar uma instância do EC2, recupere e armazene seus detalhes.

```
    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        image_id: &str,
        instance_type: InstanceType,
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }
```

- Para obter detalhes da API, consulte a DescribeInstances referência da API AWS SDK for Rust.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
```

```
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }
```

- Para obter detalhes da API, consulte a DescribeKeyPairs referência da API AWS SDK for Rust.

`DescribeRegions`

O código de exemplo a seguir mostra como usar `DescribeRegions`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_regions(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_regions().send().await?;

    println!("Regions:");
    for region in rsp.regions() {
        println!("  {}", region.region_name().unwrap());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeRegions referência da API AWS SDK for Rust.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids: Vec<String>)
 {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
        }
    }
}
```

- Para obter detalhes da API, consulte a DescribeSecurityGroups referência da API AWS SDK for Rust.

`DescribeSnapshots`

O código de exemplo a seguir mostra como usar `DescribeSnapshots`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Mostra o estado de um snapshot.

```
async fn show_state(client: &Client, id: &str) -> Result<(), Error> {
    let resp = client
        .describe_snapshots()
        .filters(Filter::builder().name("snapshot-id").values(id).build())
        .send()
        .await?;

    println!(
        "State: {}",
        resp.snapshots().first().unwrap().state().unwrap().as_ref()
    );

    Ok(())
}
```

```
async fn show_snapshots(client: &Client) -> Result<(), Error> {
    // "self" represents your account ID.
    // You can list the snapshots for any account by replacing
    // "self" with that account ID.
    let resp = client.describe_snapshots().owner_ids("self").send().await?;
    let snapshots = resp.snapshots();
    let length = snapshots.len();

    for snapshot in snapshots {
        println!(
            "ID:          {}",
            snapshot.snapshot_id().unwrap_or_default()
```

```
        );
        println!(
            "Description: {}",
            snapshot.description().unwrap_or_default()
        );
        println!("State:       {}", snapshot.state().unwrap().as_ref());
        println!();
    }

    println!();
    println!("Found {} snapshot(s)", length);
    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeSnapshots referência da API AWS SDK for Rust.

`DisassociateAddress`

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn disassociate_ip_address(&self, association_id: &str) -> Result<(),
 EC2Error> {
        self.client
            .disassociate_address()
            .association_id(association_id)
            .send()
            .await?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a DisassociateAddress referência da API AWS SDK for Rust.

`RebootInstances`

O código de exemplo a seguir mostra como usar `RebootInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
    if self.instance.is_some() {
        ec2.reboot_instance(self.instance_id()).await?;
        ec2.wait_for_instance_stopped(self.instance_id(), None)
            .await?;
        ec2.wait_for_instance_ready(self.instance_id(), None)
            .await?;
    }
    Ok(())
}
```

```
pub async fn reboot_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
    tracing::info!("Rebooting instance {instance_id}");

    self.client
        .reboot_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    Ok(())
}
```

Os waiters, por exemplo, devem estar nos estados interrompido e pronto, usando a API Waiters. O uso da API Waiters requer “use aws_sdk_ec2::client::Waiters” no arquivo rust.

```
    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn wait_for_instance_stopped(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_stopped()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to stop.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
```

```
        Ok(())
    }
```

- Para obter detalhes da API, consulte a RebootInstances referência da API AWS SDK for Rust.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
        self.client
            .release_address()
            .allocation_id(allocation_id)
            .send()
            .await?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a ReleaseAddress referência da API AWS SDK for Rust.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
```

```
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }
```

- Para obter detalhes da API, consulte a RunInstances referência da API AWS SDK for Rust.

`StartInstances`

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inicie uma instância do EC2 por ID de instância.

```
    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }
```

Aguarde até que uma instância esteja pronta e com status OK, usando a API Waiters. O uso da API Waiters requer “use aws_sdk_ec2::client::Waiters” no arquivo rust.

```
    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a StartInstances referência da API AWS SDK for Rust.

`StopInstances`

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
    tracing::info!("Stopping instance {instance_id}");

    self.client
        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    self.wait_for_instance_stopped(instance_id, None).await?;

    tracing::info!("Stopped instance.");

    Ok(())
}
```

Aguarde até que uma instância esteja no estado interrompido, usando a API Waiters. O uso da API Waiters requer “use aws_sdk_ec2::client::Waiters” no arquivo rust.

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
    tracing::info!("Stopping instance {instance_id}");

    self.client
        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    self.wait_for_instance_stopped(instance_id, None).await?;
```

```
        tracing::info!("Stopped instance.");

        Ok(())
    }
```

- Para obter detalhes da API, consulte a StopInstances referência da API AWS SDK for Rust.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    pub async fn delete_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Deleting instance with id {instance_id}");
        self.stop_instance(instance_id).await?;
        self.client
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await?;
        self.wait_for_instance_terminated(instance_id).await?;
        tracing::info!("Terminated instance with id {instance_id}");
        Ok(())
    }
```

Aguarde até que uma instância esteja no estado encerrado, usando a API Waiters. O uso da API Waiters requer “use aws_sdk_ec2::client::Waiters” no arquivo rust.

```
    async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
```

```
        self.client
            .wait_until_instance_terminated()
            .instance_ids(instance_id)
            .wait(Duration::from_secs(60))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to terminate.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- Para obter detalhes da API, consulte a TerminateInstances referência da API AWS SDK for Rust.

Exemplos do Amazon ECR usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon ECR.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DescribeRepositories`

O código de exemplo a seguir mostra como usar `DescribeRepositories`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(), aws_sdk_ecr::Error>
 {
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!("  ARN:  {}", repo.repository_arn().unwrap());
        println!("  Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeRepositories referência da API AWS SDK for Rust.

`ListImages`

O código de exemplo a seguir mostra como usar `ListImages`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_images(
    client: &aws_sdk_ecr::Client,
    repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
    let rsp = client
        .list_images()
        .repository_name(repository)
        .send()
        .await?;

    let images = rsp.image_ids();

    println!("found {} images", images.len());

    for image in images {
        println!(
            "image: {}:{}",
            image.image_tag().unwrap(),
            image.image_digest().unwrap()
        );
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListImages referência da API AWS SDK for Rust.

Exemplos do Amazon ECS usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon ECS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_cluster(client: &aws_sdk_ecs::Client, name: &str) -> Result<(),
 aws_sdk_ecs::Error> {
    let cluster = client.create_cluster().cluster_name(name).send().await?;
    println!("cluster created: {:?}", cluster);

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateCluster referência da API AWS SDK for Rust.

`DeleteCluster`

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn remove_cluster(
```

```
    client: &aws_sdk_ecs::Client,
    name: &str,
) -> Result<(), aws_sdk_ecs::Error> {
    let cluster_deleted = client.delete_cluster().cluster(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteCluster referência da API AWS SDK for Rust.

`DescribeClusters`

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_clusters(client: &aws_sdk_ecs::Client) -> Result<(),
 aws_sdk_ecs::Error> {
    let resp = client.list_clusters().send().await?;

    let cluster_arns = resp.cluster_arns();
    println!("Found {} clusters:", cluster_arns.len());

    let clusters = client
        .describe_clusters()
        .set_clusters(Some(cluster_arns.into()))
        .send()
        .await?;

    for cluster in clusters.clusters() {
        println!("  ARN:  {}", cluster.cluster_arn().unwrap());
        println!("  Name: {}", cluster.cluster_name().unwrap());
    }
```

```
    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeClusters referência da API AWS SDK for Rust.

Exemplos do Amazon EKS usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon EKS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
    arn: &str,
```

```
    subnet_ids: Vec<String>,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster = client
        .create_cluster()
        .name(name)
        .role_arn(arn)
        .resources_vpc_config(
            VpcConfigRequest::builder()
                .set_subnet_ids(Some(subnet_ids))
                .build(),
        )
        .send()
        .await?;
    println!("cluster created: {:?}", cluster);

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateCluster referência da API AWS SDK for Rust.

`DeleteCluster`

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn remove_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster_deleted = client.delete_cluster().name(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteCluster referência da API AWS SDK for Rust.

AWS Glue exemplos de uso do SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com. AWS Glue

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
```

```
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- Para obter detalhes da API, consulte a ListJobs referência da API AWS SDK for Rust.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie e execute um rastreador que rastreia um bucket público do Amazon Simple Storage Service (Amazon S3) e gera um banco de dados de metadados que descreve os dados encontrados. CSV-formatted

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;

        let start_crawler = glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;

        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

Crie e execute um trabalho que extraia dados CSV do bucket de origem do Amazon S3, os transforme removendo e renomeando campos e carregue a JSON-formatted saída em outro bucket do Amazon S3.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating job".into())
        })?;

        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just started
job".into()))?
            .to_string();
```

Exclua todos os recursos criados pela demonstração.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        }

        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;
```

- Para obter detalhes da API, consulte a CreateCrawler referência da API AWS SDK for Rust.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating job".into())
        })?;
```

- Para obter detalhes da API, consulte a CreateJob referência da API AWS SDK for Rust.

`DeleteCrawler`

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteCrawler referência da API AWS SDK for Rust.

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteDatabase referência da API AWS SDK for Rust.

`DeleteJob`

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a DeleteJob referência da API AWS SDK for Rust.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        }
```

- Para obter detalhes da API, consulte a DeleteTable referência da API AWS SDK for Rust.

`GetCrawler`

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
            let tmp_crawler = glue
                .get_crawler()
                .name(self.crawler())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obter detalhes da API, consulte a GetCrawler referência da API AWS SDK for Rust.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let database = glue
```

```
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;
```

- Para obter detalhes da API, consulte a GetDatabase referência da API AWS SDK for Rust.

`GetJobRun`

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let get_job_run = || async {
            Ok::<JobRun, GlueMvpError>(
                glue.get_job_run()
                    .job_name(self.job())
                    .run_id(job_run_id.to_string())
                    .send()
                    .await
                    .map_err(GlueMvpError::from_glue_sdk)?
                    .job_run()
                    .ok_or_else(|| GlueMvpError::Unknown("Failed to get
job_run".into()))?
                    .to_owned(),
            )
        };
```

```
        let mut job_run = get_job_run().await?;
        let mut state =
job_run.job_run_state().unwrap_or(&unknown_state).to_owned();

        while matches!(
            state,
            JobRunState::Starting | JobRunState::Stopping | JobRunState::Running
        ) {
            info!(?state, "Waiting for job to finish");
            tokio::time::sleep(self.wait_delay).await;

            job_run = get_job_run().await?;
            state = job_run.job_run_state().unwrap_or(&unknown_state).to_owned();
        }
```

- Para obter detalhes da API, consulte a GetJobRun referência da API AWS SDK for Rust.

`GetTables`

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

- Para obter detalhes da API, consulte a GetTables referência da API AWS SDK for Rust.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- Para obter detalhes da API, consulte a ListJobs referência da API AWS SDK for Rust.

`StartCrawler`

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let start_crawler = glue.start_crawler().name(self.crawler()).send().await;
```

```
        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

- Para obter detalhes da API, consulte a StartCrawler referência da API AWS SDK for Rust.

`StartJobRun`

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
```

```
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just started
job".into()))?
            .to_string();
```

- Para obter detalhes da API, consulte a StartJobRun referência da API AWS SDK for Rust.

Exemplos do IAM usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o IAM.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, IAM

O exemplo de código a seguir mostra como começar a usar o IAM.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

De src/bin /hello.rs.

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
struct HelloScenarioArgs {
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}
```

De src/iam -service-lib.rs.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
```

```
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- Para obter detalhes da API, consulte a ListPolicies referência da API AWS SDK for Rust.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

> **Warning**
>
> Para evitar riscos de segurança, não use usuários do IAM para autenticação ao desenvolver software com propósito específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como Centro de Identidade do AWS IAM.

- Crie um usuário sem permissões.

- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.
- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client as
 iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document) =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}
```

```
async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));

    let shared_config = aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}", "iam_demo_user_",
 uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
```

```
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );

    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
            .await?;
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
    let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
        .await?;
```

```
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
        }
    }

    let sts_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    let sts_client: stsClient = stsClient::new(&sts_config);
    sleep(Duration::from_secs(10)).await;
    let assumed_role = sts_client
        .assume_role()
        .role_arn(assume_role_role.arn())
        .role_session_name(format!("iam_demo_assumerole_session_{uuid}"))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
        assumed_role
```

```
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
```

```
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name())
        .policy_arn(policy.arn().unwrap_or_default())
        .send()
        .await
}
```

- Para obter detalhes da API, consulte a AttachRolePolicy referência da API AWS SDK for Rust.

`AttachUserPolicy`

O código de exemplo a seguir mostra como usar `AttachUserPolicy`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
```

```
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a AttachUserPolicy referência da API AWS SDK for Rust.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
 Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
 loop {
        match client.create_access_key().user_name(user_name).send().await {
            Ok(inner_response) => {
                break Ok(inner_response);
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
```

```
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    Ok(response.unwrap().access_key.unwrap())
}
```

- Para obter detalhes da API, consulte a CreateAccessKey referência da API AWS SDK for Rust.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- Para obter detalhes da API, consulte a CreatePolicy referência da API AWS SDK for Rust.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    };

    Ok(response.role.unwrap())
}
```

- Para obter detalhes da API, consulte a CreateRole referência da API AWS SDK for Rust.

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput, SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a CreateServiceLinkedRole referência da API AWS SDK for Rust.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
 iamError> {
    let response = client.create_user().user_name(user_name).send().await?;

    Ok(response.user.unwrap())
}
```

- Para obter detalhes da API, consulte a CreateUser referência da API AWS SDK for Rust.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_access_key(
    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
            .delete_access_key()
            .user_name(user.user_name())
            .access_key_id(key.access_key_id())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(e) => {
                println!("Can't delete the access key: {:?}", e);
                sleep(Duration::from_secs(2)).await;
```

```
            }
        }
    }
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteAccessKey referência da API AWS SDK for Rust.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
 iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeletePolicy referência da API AWS SDK for Rust.

`DeleteRole`

O código de exemplo a seguir mostra como usar `DeleteRole`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError> {
    let role = role.clone();
    while client
        .delete_role()
        .role_name(role.role_name())
        .send()
        .await
        .is_err()
    {
        sleep(Duration::from_secs(2)).await;
    }
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteRole referência da API AWS SDK for Rust.

`DeleteServiceLinkedRole`

O código de exemplo a seguir mostra como usar `DeleteServiceLinkedRole`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
```

```
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteServiceLinkedRole referência da API AWS SDK for Rust.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
 SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<(), SdkError<DeleteUserError>> = loop {
        match client
            .delete_user()
            .user_name(user.user_name())
            .send()
            .await
        {
            Ok(_) => {
                break Ok(());
```

```
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    response
}
```

- Para obter detalhes da API, consulte a DeleteUser referência da API AWS SDK for Rust.

`DeleteUserPolicy`

O código de exemplo a seguir mostra como usar `DeleteUserPolicy`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
```

```
}
```

- Para obter detalhes da API, consulte a DeleteUserPolicy referência da API AWS SDK for Rust.

DetachRolePolicy

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a DetachRolePolicy referência da API AWS SDK for Rust.

DetachUserPolicy

O código de exemplo a seguir mostra como usar `DetachUserPolicy`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a DetachUserPolicy referência da API AWS SDK for Rust.

`GetAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn get_account_password_policy(
```

```
    client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput, SdkError<GetAccountPasswordPolicyError>>
 {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a GetAccountPasswordPolicy referência da API AWS SDK for Rust.

`GetRole`

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- Para obter detalhes da API, consulte a GetRole referência da API AWS SDK for Rust.

`ListAttachedRolePolicies`

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput, SdkError<ListAttachedRolePoliciesError>>
 {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListAttachedRolePolicies referência da API AWS SDK for Rust.

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListGroups referência da API AWS SDK for Rust.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- Para obter detalhes da API, consulte a ListPolicies referência da API AWS SDK for Rust.

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListRolePolicies referência da API AWS SDK for Rust.

`ListRoles`

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
```

```
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListRoles referência da API AWS SDK for Rust.

Listar provedores de SAML

O código de exemplo a seguir mostra como usar `ListSAMLProviders`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_saml_providers(
    client: &Client,
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {
    let response = client.list_saml_providers().send().await?;

    Ok(response)
}
```

- Consulte detalhes da API em ListSAMLProvider na referência da API do AWS SDK for Rust.

ListUsers

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- Para obter detalhes da API, consulte a ListUsers referência da API AWS SDK for Rust.

AWS IoT exemplos de uso do SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com. AWS IoT

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DescribeEndpoint`

O código de exemplo a seguir mostra como usar `DescribeEndpoint`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error> {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;

    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeEndpoint referência da API AWS SDK for Rust.

`ListThings`

O código de exemplo a seguir mostra como usar `ListThings`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name:  {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type:  {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
        println!(
            "  ARN:   {}",
            thing.thing_arn.as_deref().unwrap_or_default()
        );
        println!();
    }

    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListThings referência da API AWS SDK for Rust.

Exemplos do Kinesis usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Kinesis.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Exemplos sem servidor

Ações

`CreateStream`

O código de exemplo a seguir mostra como usar `CreateStream`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client
        .create_stream()
        .stream_name(stream)
        .shard_count(4)
        .send()
        .await?;

    println!("Created stream");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateStream referência da API AWS SDK for Rust.

`DeleteStream`

O código de exemplo a seguir mostra como usar `DeleteStream`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn remove_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client.delete_stream().stream_name(stream).send().await?;

    println!("Deleted stream.");

    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteStream referência da API AWS SDK for Rust.

`DescribeStream`

O código de exemplo a seguir mostra como usar `DescribeStream`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_stream(client: &Client, stream: &str) -> Result<(), Error> {
    let resp = client.describe_stream().stream_name(stream).send().await?;

    let desc = resp.stream_description.unwrap();
```

```
    println!("Stream description:");
    println!("  Name:              {}:", desc.stream_name());
    println!("  Status:            {:?}", desc.stream_status());
    println!("  Open shards:       {:?}", desc.shards.len());
    println!("  Retention (hours): {}", desc.retention_period_hours());
    println!("  Encryption:        {:?}", desc.encryption_type.unwrap());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeStream referência da API AWS SDK for Rust.

`ListStreams`

O código de exemplo a seguir mostra como usar `ListStreams`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_streams(client: &Client) -> Result<(), Error> {
    let resp = client.list_streams().send().await?;

    println!("Stream names:");

    let streams = resp.stream_names;
    for stream in &streams {
        println!("  {}", stream);
    }

    println!("Found {} stream(s)", streams.len());

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListStreams referência da API AWS SDK for Rust.

`PutRecord`

O código de exemplo a seguir mostra como usar `PutRecord`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn add_record(client: &Client, stream: &str, key: &str, data: &str) ->
 Result<(), Error> {
    let blob = Blob::new(data);

    client
        .put_record()
        .data(blob)
        .partition_key(key)
        .stream_name(stream)
        .send()
        .await?;

    println!("Put data into stream.");

    Ok(())
}
```

- Para obter detalhes da API, consulte a PutRecord referência da API AWS SDK for Rust.

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do Kinesis com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::kinesis::KinesisEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) -> Result<(), Error> {
    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    event.payload.records.iter().for_each(|record| {
        tracing::info!("EventId:
 {}",record.event_id.as_deref().unwrap_or_default());

        let record_data = std::str::from_utf8(&record.kinesis.data);

        match record_data {
            Ok(data) => {
                // log the record data
                tracing::info!("Data: {}", data);
            }
            Err(e) => {
                tracing::error!("Error: {}", e);
```

```
            }
        }
    });

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relate falhas de itens em lote do Kinesis com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::kinesis::KinesisEvent,
    kinesis::KinesisEventRecord,
    streams::{KinesisBatchItemFailure, KinesisEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) ->
 Result<KinesisEventResponse, Error> {
    let mut response = KinesisEventResponse {
        batch_item_failures: vec![],
    };

    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in &event.payload.records {
        tracing::info!(
            "EventId: {}",
            record.event_id.as_deref().unwrap_or_default()
        );

        let record_processing_result = process_record(record);

        if record_processing_result.is_err() {
            response.batch_item_failures.push(KinesisBatchItemFailure {
                item_identifier: record.kinesis.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed item
 onwards. */
            return Ok(response);
        }
    }

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
```

```
    );

    Ok(response)
}

fn process_record(record: &KinesisEventRecord) -> Result<(), Error> {
    let record_data = std::str::from_utf8(record.kinesis.data.as_slice());

    if let Some(err) = record_data.err() {
        tracing::error!("Error: {}", err);
        return Err(Error::from(err));
    }

    let record_data = record_data.unwrap_or_default();

    // do something interesting with the data
    tracing::info!("Data: {}", record_data);

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

AWS KMS exemplos de uso do SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com. AWS KMS

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_key(client: &Client) -> Result<(), Error> {
    let resp = client.create_key().send().await?;

    let id = resp.key_metadata.as_ref().unwrap().key_id();

    println!("Key: {}", id);

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateKey referência da API AWS SDK for Rust.

`Decrypt`

O código de exemplo a seguir mostra como usar `Decrypt`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(),
 Error> {
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(filename)
        .map(|input| {
            base64::decode(input).expect("Input file does not contain valid base 64
 characters.")
        })
        .map(Blob::new);

    let resp = client
        .decrypt()
        .key_id(key)
        .ciphertext_blob(data.unwrap())
        .send()
        .await?;

    let inner = resp.plaintext.unwrap();
    let bytes = inner.as_ref();

    let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to UTF-8");

    println!();
    println!("Decoded string:");
    println!("{}", s);

    Ok(())
}
```

- Consulte detalhes da API em Descriptografar na Referência da API AWS SDK para Rust.

`Encrypt`

O código de exemplo a seguir mostra como usar `Encrypt`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn encrypt_string(
    verbose: bool,
    client: &Client,
    text: &str,
    key: &str,
    out_file: &str,
) -> Result<(), Error> {
    let blob = Blob::new(text.as_bytes());

    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    let mut ofile = File::create(out_file).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {:?}", out_file);
        println!("{}", s);
    }

    Ok(())
}
```

- Consulte detalhes da API em Criptografar na Referência da API AWS SDK para Rust.

`GenerateDataKey`

O código de exemplo a seguir mostra como usar `GenerateDataKey`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Para obter detalhes da API, consulte a GenerateDataKey referência da API AWS SDK for Rust.

`GenerateDataKeyWithoutPlaintext`

O código de exemplo a seguir mostra como usar `GenerateDataKeyWithoutPlaintext`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Para obter detalhes da API, consulte a GenerateDataKeyWithoutPlaintext referência da API AWS SDK for Rust.

`GenerateRandom`

O código de exemplo a seguir mostra como usar `GenerateRandom`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
        .number_of_bytes(length)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.plaintext.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Para obter detalhes da API, consulte a GenerateRandom referência da API AWS SDK for Rust.

`ListKeys`

O código de exemplo a seguir mostra como usar `ListKeys`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListKeys referência da API AWS SDK for Rust.

`ReEncrypt`

O código de exemplo a seguir mostra como usar `ReEncrypt`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn reencrypt_string(
    verbose: bool,
    client: &Client,
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client
        .re_encrypt()
        .ciphertext_blob(data.unwrap())
        .source_key_id(first_key)
        .destination_key_id(new_key)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);
    let o = &output_file;

    let mut ofile = File::create(o).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {}:", output_file);
        println!("{}", s);
    } else {
        println!("Wrote base64-encoded output to {}", output_file);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ReEncrypt referência da API AWS SDK for Rust.

Exemplos de Lambda usando SDKs para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

AWS contribuições da comunidade são exemplos que foram criados e mantidos por várias equipes em todo o mundo AWS. Para deixar seu feedback, use o mecanismo fornecido nos repositórios vinculados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor
- AWS contribuições da comunidade

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.

- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

As Cargo.toml com dependências usadas nesse cenário.

```
[package]
name = "lambda-code-examples"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/
manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-ec2 = { version = "1.3.0" }
aws-sdk-iam = { version = "1.3.0" }
aws-sdk-lambda = { version = "1.3.0" }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
tracing = "0.1.37"
serde_json = "1.0.94"
anyhow = "1.0.71"
uuid = { version = "1.3.3", features = ["v4"] }
lambda_runtime = "0.8.0"
serde = "1.0.164"
```

Uma coleção de utilitários que simplificam chamar o Lambda para este cenário. Esse arquivo é src/ations .rs na caixa.

```
use anyhow::anyhow;
use aws_sdk_iam::operation::{create_role::CreateRoleError,
 delete_role::DeleteRoleOutput};
use aws_sdk_lambda::{
    operation::{
        delete_function::DeleteFunctionOutput, get_function::GetFunctionOutput,
        invoke::InvokeOutput, list_functions::ListFunctionsOutput,
        update_function_code::UpdateFunctionCodeOutput,
        update_function_configuration::UpdateFunctionConfigurationOutput,
    },
    primitives::ByteStream,
    types::{Environment, FunctionCode, LastUpdateStatus, State},
};
use aws_sdk_s3::{
    error::ErrorMetadata,
    operation::{delete_bucket::DeleteBucketOutput,
 delete_object::DeleteObjectOutput},
    types::CreateBucketConfiguration,
};
use aws_smithy_types::Blob;
use serde::{ser::SerializeMap, Serialize};
use std::{fmt::Display, path::PathBuf, str::FromStr, time::Duration};
use tracing::{debug, info, warn};

/* Operation describes  */
#[derive(Clone, Copy, Debug, Serialize)]
pub enum Operation {
    #[serde(rename = "plus")]
    Plus,
    #[serde(rename = "minus")]
    Minus,
    #[serde(rename = "times")]
    Times,
    #[serde(rename = "divided-by")]
    DividedBy,
}
```

```
impl FromStr for Operation {
    type Err = anyhow::Error;

    fn from_str(s: &str) -> Result<Self, Self::Err> {
        match s {
            "plus" => Ok(Operation::Plus),
            "minus" => Ok(Operation::Minus),
            "times" => Ok(Operation::Times),
            "divided-by" => Ok(Operation::DividedBy),
            _ => Err(anyhow!("Unknown operation {s}")),
        }
    }
}

impl Display for Operation {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match self {
            Operation::Plus => write!(f, "plus"),
            Operation::Minus => write!(f, "minus"),
            Operation::Times => write!(f, "times"),
            Operation::DividedBy => write!(f, "divided-by"),
        }
    }
}

/**
 * InvokeArgs will be serialized as JSON and sent to the AWS Lambda handler.
 */
#[derive(Debug)]
pub enum InvokeArgs {
    Increment(i32),
    Arithmetic(Operation, i32, i32),
}

impl Serialize for InvokeArgs {
    fn serialize<S>(&self, serializer: S) -> Result<S::Ok, S::Error>
    where
        S: serde::Serializer,
    {
        match self {
            InvokeArgs::Increment(i) => serializer.serialize_i32(*i),
            InvokeArgs::Arithmetic(o, i, j) => {
                let mut map: S::SerializeMap = serializer.serialize_map(Some(3))?;
                map.serialize_key(&"op".to_string())?;
```

```
                map.serialize_value(&o.to_string())?;
                map.serialize_key(&"i".to_string())?;
                map.serialize_value(&i)?;
                map.serialize_key(&"j".to_string())?;
                map.serialize_value(&j)?;
                map.end()
            }
        }
    }
}

/** A policy document allowing Lambda to execute this function on the account's
 behalf. */
const ROLE_POLICY_DOCUMENT: &str = r#"{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "Service": "lambda.amazonaws.com" },
            "Action": "sts:AssumeRole"
        }
    ]
}"#;

/**
 * A LambdaManager gathers all the resources necessary to run the Lambda example
 scenario.
 * This includes instantiated aws_sdk clients and details of resource names.
 */
pub struct LambdaManager {
    iam_client: aws_sdk_iam::Client,
    lambda_client: aws_sdk_lambda::Client,
    s3_client: aws_sdk_s3::Client,
    lambda_name: String,
    role_name: String,
    bucket: String,
    own_bucket: bool,
}

// These unit type structs provide nominal typing on top of String parameters for
 LambdaManager::new
pub struct LambdaName(pub String);
pub struct RoleName(pub String);
pub struct Bucket(pub String);
```

```
pub struct OwnBucket(pub bool);

impl LambdaManager {
    pub fn new(
        iam_client: aws_sdk_iam::Client,
        lambda_client: aws_sdk_lambda::Client,
        s3_client: aws_sdk_s3::Client,
        lambda_name: LambdaName,
        role_name: RoleName,
        bucket: Bucket,
        own_bucket: OwnBucket,
    ) -> Self {
        Self {
            iam_client,
            lambda_client,
            s3_client,
            lambda_name: lambda_name.0,
            role_name: role_name.0,
            bucket: bucket.0,
            own_bucket: own_bucket.0,
        }
    }

    /**
     * Load the AWS configuration from the environment.
     * Look up lambda_name and bucket if none are given, or generate a random name
 if not present in the environment.
     * If the bucket name is provided, the caller needs to have created the bucket.
     * If the bucket name is generated, it will be created.
     */
    pub async fn load_from_env(lambda_name: Option<String>, bucket: Option<String>)
 -> Self {
        let sdk_config = aws_config::load_from_env().await;
        let lambda_name = LambdaName(lambda_name.unwrap_or_else(|| {
            std::env::var("LAMBDA_NAME").unwrap_or_else(|_|
 "rust_lambda_example".to_string())
        }));
        let role_name = RoleName(format!("{}_role", lambda_name.0));
        let (bucket, own_bucket) =
            match bucket {
                Some(bucket) => (Bucket(bucket), false),
                None => (
                    Bucket(std::env::var("LAMBDA_BUCKET").unwrap_or_else(|_| {
                        format!("rust-lambda-example-{}", uuid::Uuid::new_v4())
```

```
                    })),
                    true,
                ),
            };

        let s3_client = aws_sdk_s3::Client::new(&sdk_config);

        if own_bucket {
            info!("Creating bucket for demo: {}", bucket.0);
            s3_client
                .create_bucket()
                .bucket(bucket.0.clone())
                .create_bucket_configuration(
                    CreateBucketConfiguration::builder()

 .location_constraint(aws_sdk_s3::types::BucketLocationConstraint::from(
                            sdk_config.region().unwrap().as_ref(),
                        ))
                        .build(),
                )
                .send()
                .await
                .unwrap();
        }

        Self::new(
            aws_sdk_iam::Client::new(&sdk_config),
            aws_sdk_lambda::Client::new(&sdk_config),
            s3_client,
            lambda_name,
            role_name,
            bucket,
            OwnBucket(own_bucket),
        )
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --output-
format Zip`.
     */
    async fn prepare_function(
        &self,
```

```
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }

    /**
     * Create a function, uploading from a zip file.
     */
    pub async fn create_function(&self, zip_file: PathBuf) -> Result<String,
anyhow::Error> {
        let code = self.prepare_function(zip_file, None).await?;

        let key = code.s3_key().unwrap().to_string();

        let role = self.create_role().await.map_err(|e| anyhow!(e))?;

        info!("Created iam role, waiting 15s for it to become active");
        tokio::time::sleep(Duration::from_secs(15)).await;

        info!("Creating lambda function {}", self.lambda_name);
        let _ = self
            .lambda_client
            .create_function()
            .function_name(self.lambda_name.clone())
            .code(code)
            .role(role.arn())
```

```
            .runtime(aws_sdk_lambda::types::Runtime::Providedal2)
            .handler("_unused")
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        self.lambda_client
            .publish_version()
            .function_name(self.lambda_name.clone())
            .send()
            .await?;

        Ok(key)
    }

    /**
     * Create an IAM execution role for the managed Lambda function.
     * If the role already exists, use that instead.
     */
    async fn create_role(&self) -> Result<aws_sdk_iam::types::Role, CreateRoleError>
{
        info!("Creating execution role for function");
        let get_role = self
            .iam_client
            .get_role()
            .role_name(self.role_name.clone())
            .send()
            .await;
        if let Ok(get_role) = get_role {
            if let Some(role) = get_role.role {
                return Ok(role);
            }
        }

        let create_role = self
            .iam_client
            .create_role()
            .role_name(self.role_name.clone())
            .assume_role_policy_document(ROLE_POLICY_DOCUMENT)
            .send()
            .await;
```

```
        match create_role {
            Ok(create_role) => match create_role.role {
                Some(role) => Ok(role),
                None => Err(CreateRoleError::generic(
                    ErrorMetadata::builder()
                        .message("CreateRole returned empty success")
                        .build(),
                )),
            },
            Err(err) => Err(err.into_service_error()),
        }
    }

    /**
     * Poll `is_function_ready` with a 1-second delay. It returns when the function
is ready or when there's an error checking the function's state.
     */
    pub async fn wait_for_function_ready(&self) -> Result<(), anyhow::Error> {
        info!("Waiting for function");
        while !self.is_function_ready(None).await? {
            info!("Function is not ready, sleeping 1s");
            tokio::time::sleep(Duration::from_secs(1)).await;
        }
        Ok(())
    }

    /**
     * Check if a Lambda function is ready to be invoked.
     * A Lambda function is ready for this scenario when its state is active and its
LastUpdateStatus is Successful.
     * Additionally, if a sha256 is provided, the function must have that as its
current code hash.
     * Any missing properties or failed requests will be reported as an Err.
     */
    async fn is_function_ready(
        &self,
        expected_code_sha256: Option<&str>,
    ) -> Result<bool, anyhow::Error> {
        match self.get_function().await {
            Ok(func) => {
                if let Some(config) = func.configuration() {
                    if let Some(state) = config.state() {
                        info!(?state, "Checking if function is active");
                        if !matches!(state, State::Active) {
```

```
                            return Ok(false);
                        }
                    }
                    match config.last_update_status() {
                        Some(last_update_status) => {
                            info!(?last_update_status, "Checking if function is
ready");
                            match last_update_status {
                                LastUpdateStatus::Successful => {
                                    // continue
                                }
                                LastUpdateStatus::Failed |
LastUpdateStatus::InProgress => {
                                    return Ok(false);
                                }
                                unknown => {
                                    warn!(
                                        status_variant = unknown.as_str(),
                                        "LastUpdateStatus unknown"
                                    );
                                    return Err(anyhow!(
                                        "Unknown LastUpdateStatus, fn config is
{config:?}"
                                    ));
                                }
                            }
                        }
                        None => {
                            warn!("Missing last update status");
                            return Ok(false);
                        }
                    };
                    if expected_code_sha256.is_none() {
                        return Ok(true);
                    }
                    if let Some(code_sha256) = config.code_sha256() {
                        return Ok(code_sha256 ==
expected_code_sha256.unwrap_or_default());
                    }
                }
            }
            Err(e) => {
                warn!(?e, "Could not get function while waiting");
            }
```

```
        }
        Ok(false)
    }

    /** Get the Lambda function with this Manager's name. */
    pub async fn get_function(&self) -> Result<GetFunctionOutput, anyhow::Error> {
        info!("Getting lambda function");
        self.lambda_client
            .get_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

    /** List all Lambda functions in the current Region. */
    pub async fn list_functions(&self) -> Result<ListFunctionsOutput, anyhow::Error>
{
        info!("Listing lambda functions");
        self.lambda_client
            .list_functions()
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

    /** Invoke the lambda function using calculator InvokeArgs. */
    pub async fn invoke(&self, args: InvokeArgs) -> Result<InvokeOutput,
anyhow::Error> {
        info!(?args, "Invoking {}", self.lambda_name);
        let payload = serde_json::to_string(&args)?;
        debug!(?payload, "Sending payload");
        self.lambda_client
            .invoke()
            .function_name(self.lambda_name.clone())
            .payload(Blob::new(payload))
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

    /** Given a Path to a zip file, update the function's code and wait for the
update to finish. */
    pub async fn update_function_code(
```

```
        &self,
        zip_file: PathBuf,
        key: String,
    ) -> Result<UpdateFunctionCodeOutput, anyhow::Error> {
        let function_code = self.prepare_function(zip_file, Some(key)).await?;

        info!("Updating code for {}", self.lambda_name);
        let update = self
            .lambda_client
            .update_function_code()
            .function_name(self.lambda_name.clone())
            .s3_bucket(self.bucket.clone())
            .s3_key(function_code.s3_key().unwrap().to_string())
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(update)
    }

    /** Update the environment for a function. */
    pub async fn update_function_configuration(
        &self,
        environment: Environment,
    ) -> Result<UpdateFunctionConfigurationOutput, anyhow::Error> {
        info!(
            ?environment,
            "Updating environment for {}", self.lambda_name
        );
        let updated = self
            .lambda_client
            .update_function_configuration()
            .function_name(self.lambda_name.clone())
            .environment(environment)
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(updated)
    }
```

```
    /** Delete a function and its role, and if possible or necessary, its associated
code object and bucket. */
    pub async fn delete_function(
        &self,
        location: Option<String>,
    ) -> (
        Result<DeleteFunctionOutput, anyhow::Error>,
        Result<DeleteRoleOutput, anyhow::Error>,
        Option<Result<DeleteObjectOutput, anyhow::Error>>,
    ) {
        info!("Deleting lambda function {}", self.lambda_name);
        let delete_function = self
            .lambda_client
            .delete_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        info!("Deleting iam role {}", self.role_name);
        let delete_role = self
            .iam_client
            .delete_role()
            .role_name(self.role_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        let delete_object: Option<Result<DeleteObjectOutput, anyhow::Error>> =
            if let Some(location) = location {
                info!("Deleting object {location}");
                Some(
                    self.s3_client
                        .delete_object()
                        .bucket(self.bucket.clone())
                        .key(location)
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                info!(?location, "Skipping delete object");
                None
```

```
            };

        (delete_function, delete_role, delete_object)
    }

    pub async fn cleanup(
        &self,
        location: Option<String>,
    ) -> (
        (
            Result<DeleteFunctionOutput, anyhow::Error>,
            Result<DeleteRoleOutput, anyhow::Error>,
            Option<Result<DeleteObjectOutput, anyhow::Error>>,
        ),
        Option<Result<DeleteBucketOutput, anyhow::Error>>,
    ) {
        let delete_function = self.delete_function(location).await;

        let delete_bucket = if self.own_bucket {
            info!("Deleting bucket {}", self.bucket);
            if delete_function.2.is_none() ||
 delete_function.2.as_ref().unwrap().is_ok() {
                Some(
                    self.s3_client
                        .delete_bucket()
                        .bucket(self.bucket.clone())
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                None
            }
        } else {
            info!("No bucket to clean up");
            None
        };

        (delete_function, delete_bucket)
    }
}

/**
```

```
 * Testing occurs primarily as an integration test running the `scenario` bin
 successfully.
 * Each action relies deeply on the internal workings and state of Amazon Simple
 Storage Service (Amazon S3), Lambda, and IAM working together.
 * It is therefore infeasible to mock the clients to test the individual actions.
 */
#[cfg(test)]
mod test {
    use super::{InvokeArgs, Operation};
    use serde_json::json;

    /** Make sure that the JSON output of serializing InvokeArgs is what's expected
 by the calculator. */
    #[test]
    fn test_serialize() {
        assert_eq!(json!(InvokeArgs::Increment(5)), 5);
        assert_eq!(
            json!(InvokeArgs::Arithmetic(Operation::Plus, 5, 7)).to_string(),
            r#"{"op":"plus","i":5,"j":7}"#.to_string(),
        );
    }
}
```

Um binário para executar o cenário do início ao fim usando sinalizadores de linha de comando para controlar algum comportamento. Esse arquivo é src/bin /scenario.rs na caixa.

```
/*
## Service actions

Service actions wrap the SDK call, taking a client and any specific parameters
 necessary for the call.

* CreateFunction
* GetFunction
* ListFunctions
* Invoke
* UpdateFunctionCode
* UpdateFunctionConfiguration
* DeleteFunction

## Scenario
```

```
A scenario runs at a command prompt and prints output to the user on the result
 of each service action. A scenario can run in one of two ways: straight through,
 printing out progress as it goes, or as an interactive question/answer script.

## Getting started with functions

Use an SDK to manage AWS Lambda functions: create a function, invoke it, update its
 code, invoke it again, view its output and logs, and delete it.

This scenario uses two Lambda handlers:
_Note: Handlers don't use AWS SDK API calls._

The increment handler is straightforward:

1. It accepts a number, increments it, and returns the new value.
2. It performs simple logging of the result.

The arithmetic handler is more complex:
1. It accepts a set of actions ['plus', 'minus', 'times', 'divided-by'] and two
 numbers, and returns the result of the calculation.
2. It uses an environment variable to control log level (such as DEBUG, INFO,
 WARNING, ERROR).
It logs a few things at different levels, such as:
    * DEBUG: Full event data.
    * INFO: The calculation result.
    * WARN~ING~: When a divide by zero error occurs.
    * This will be the typical `RUST_LOG` variable.


The steps of the scenario are:

1. Create an AWS Identity and Access Management (IAM) role that meets the following
 requirements:
    * Has an assume_role policy that grants 'lambda.amazonaws.com' the
 'sts:AssumeRole' action.
    * Attaches the 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' managed role.
    * _You must wait for ~10 seconds after the role is created before you can use
 it!_
2. Create a function (CreateFunction) for the increment handler by packaging it as a
 zip and doing one of the following:
    * Adding it with CreateFunction Code.ZipFile.
    * --or--
```

```
    * Uploading it to Amazon Simple Storage Service (Amazon S3) and adding it with
 CreateFunction Code.S3Bucket/S3Key.
    * _Note: Zipping the file does not have to be done in code._
    * If you have a waiter, use it to wait until the function is active. Otherwise,
 call GetFunction until State is Active.
3. Invoke the function with a number and print the result.
4. Update the function (UpdateFunctionCode) to the arithmetic handler by packaging
 it as a zip and doing one of the following:
    * Adding it with UpdateFunctionCode ZipFile.
    * --or--
    * Uploading it to Amazon S3 and adding it with UpdateFunctionCode S3Bucket/
S3Key.
5. Call GetFunction until Configuration.LastUpdateStatus is 'Successful' (or
 'Failed').
6. Update the environment variable by calling UpdateFunctionConfiguration and pass
 it a log level, such as:
    * Environment={'Variables': {'RUST_LOG': 'TRACE'}}
7. Invoke the function with an action from the list and a couple of values. Include
 LogType='Tail' to get logs in the result. Print the result of the calculation and
 the log.
8. [Optional] Invoke the function to provoke a divide-by-zero error and show the log
 result.
9. List all functions for the account, using pagination (ListFunctions).
10. Delete the function (DeleteFunction).
11. Delete the role.

Each step should use the function created in Service Actions to abstract calling the
 SDK.
 */

use aws_sdk_lambda::{operation::invoke::InvokeOutput, types::Environment};
use clap::Parser;
use std::{collections::HashMap, path::PathBuf};
use tracing::{debug, info, warn};
use tracing_subscriber::EnvFilter;

use lambda_code_examples::actions::{
    InvokeArgs::{Arithmetic, Increment},
    LambdaManager, Operation,
};

#[derive(Debug, Parser)]
pub struct Opt {
    /// The AWS Region.
```

```
    #[structopt(short, long)]
    pub region: Option<String>,

    // The bucket to use for the FunctionCode.
    #[structopt(short, long)]
    pub bucket: Option<String>,

    // The name of the Lambda function.
    #[structopt(short, long)]
    pub lambda_name: Option<String>,

    // The number to increment.
    #[structopt(short, long, default_value = "12")]
    pub inc: i32,

    // The left operand.
    #[structopt(long, default_value = "19")]
    pub num_a: i32,

    // The right operand.
    #[structopt(long, default_value = "23")]
    pub num_b: i32,

    // The arithmetic operation.
    #[structopt(short, long, default_value = "plus")]
    pub operation: Operation,

    #[structopt(long)]
    pub cleanup: Option<bool>,

    #[structopt(long)]
    pub no_cleanup: Option<bool>,
}

fn code_path(lambda: &str) -> PathBuf {
    PathBuf::from(format!("../target/lambda/{lambda}/bootstrap.zip"))
}

fn log_invoke_output(invoke: &InvokeOutput, message: &str) {
    if let Some(payload) = invoke.payload().cloned() {
        let payload = String::from_utf8(payload.into_inner());
        info!(?payload, message);
    } else {
        info!("Could not extract payload")
```

```
    }
    if let Some(logs) = invoke.log_result() {
        debug!(?logs, "Invoked function logs")
    } else {
        debug!("Invoked function had no logs")
    }
}

async fn main_block(
    opt: &Opt,
    manager: &LambdaManager,
    code_location: String,
) -> Result<(), anyhow::Error> {
    let invoke = manager.invoke(Increment(opt.inc)).await?;
    log_invoke_output(&invoke, "Invoked function configured as increment");

    let update_code = manager
        .update_function_code(code_path("arithmetic"), code_location.clone())
        .await?;

    let code_sha256 = update_code.code_sha256().unwrap_or("Unknown SHA");
    info!(?code_sha256, "Updated function code with arithmetic.zip");

    let arithmetic_args = Arithmetic(opt.operation, opt.num_a, opt.num_b);
    let invoke = manager.invoke(arithmetic_args).await?;
    log_invoke_output(&invoke, "Invoked function configured as arithmetic");

    let update = manager
        .update_function_configuration(
            Environment::builder()
                .set_variables(Some(HashMap::from([(
                    "RUST_LOG".to_string(),
                    "trace".to_string(),
                )])))
                .build(),
        )
        .await?;
    let updated_environment = update.environment();
    info!(?updated_environment, "Updated function configuration");

    let invoke = manager
        .invoke(Arithmetic(opt.operation, opt.num_a, opt.num_b))
        .await?;
    log_invoke_output(
```

```
        &invoke,
        "Invoked function configured as arithmetic with increased logging",
    );

    let invoke = manager
        .invoke(Arithmetic(Operation::DividedBy, opt.num_a, 0))
        .await?;
    log_invoke_output(
        &invoke,
        "Invoked function configured as arithmetic with divide by zero",
    );

    Ok::<(), anyhow::Error>(())
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt()
        .without_time()
        .with_file(true)
        .with_line_number(true)
        .with_env_filter(EnvFilter::from_default_env())
        .init();

    let opt = Opt::parse();
    let manager = LambdaManager::load_from_env(opt.lambda_name.clone(),
 opt.bucket.clone()).await;

    let key = match manager.create_function(code_path("increment")).await {
        Ok(init) => {
            info!(?init, "Created function, initially with increment.zip");
            let run_block = main_block(&opt, &manager, init.clone()).await;
            info!(?run_block, "Finished running example, cleaning up");
            Some(init)
        }
        Err(err) => {
            warn!(?err, "Error happened when initializing function");
            None
        }
    };

    if Some(false) == opt.cleanup || Some(true) == opt.no_cleanup {
        info!("Skipping cleanup")
    } else {
```

```
        let delete = manager.cleanup(key).await;
        info!(?delete, "Deleted function & cleaned up resources");
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Create a function, uploading from a zip file.
     */
    pub async fn create_function(&self, zip_file: PathBuf) -> Result<String,
anyhow::Error> {
        let code = self.prepare_function(zip_file, None).await?;

        let key = code.s3_key().unwrap().to_string();

        let role = self.create_role().await.map_err(|e| anyhow!(e))?;
```

```
        info!("Created iam role, waiting 15s for it to become active");
        tokio::time::sleep(Duration::from_secs(15)).await;

        info!("Creating lambda function {}", self.lambda_name);
        let _ = self
            .lambda_client
            .create_function()
            .function_name(self.lambda_name.clone())
            .code(code)
            .role(role.arn())
            .runtime(aws_sdk_lambda::types::Runtime::Providedal2)
            .handler("_unused")
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        self.lambda_client
            .publish_version()
            .function_name(self.lambda_name.clone())
            .send()
            .await?;

        Ok(key)
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --output-
format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
```

```
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }
```

- Para obter detalhes da API, consulte a CreateFunction referência da API AWS SDK for Rust.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** Delete a function and its role, and if possible or necessary, its associated
code object and bucket. */
    pub async fn delete_function(
        &self,
        location: Option<String>,
    ) -> (
        Result<DeleteFunctionOutput, anyhow::Error>,
        Result<DeleteRoleOutput, anyhow::Error>,
        Option<Result<DeleteObjectOutput, anyhow::Error>>,
    ) {
        info!("Deleting lambda function {}", self.lambda_name);
```

```
        let delete_function = self
            .lambda_client
            .delete_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        info!("Deleting iam role {}", self.role_name);
        let delete_role = self
            .iam_client
            .delete_role()
            .role_name(self.role_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        let delete_object: Option<Result<DeleteObjectOutput, anyhow::Error>> =
            if let Some(location) = location {
                info!("Deleting object {location}");
                Some(
                    self.s3_client
                        .delete_object()
                        .bucket(self.bucket.clone())
                        .key(location)
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                info!(?location, "Skipping delete object");
                None
            };

        (delete_function, delete_role, delete_object)
    }
```

- Para obter detalhes da API, consulte a DeleteFunction referência da API AWS SDK for Rust.

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** Get the Lambda function with this Manager's name. */
    pub async fn get_function(&self) -> Result<GetFunctionOutput, anyhow::Error> {
        info!("Getting lambda function");
        self.lambda_client
            .get_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from)
    }
```

- Para obter detalhes da API, consulte a GetFunction referência da API AWS SDK for Rust.

`Invocar`

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** Invoke the lambda function using calculator InvokeArgs. */
    pub async fn invoke(&self, args: InvokeArgs) -> Result<InvokeOutput,
anyhow::Error> {
        info!(?args, "Invoking {}", self.lambda_name);
        let payload = serde_json::to_string(&args)?;
        debug!(?payload, "Sending payload");
```

```
        self.lambda_client
            .invoke()
            .function_name(self.lambda_name.clone())
            .payload(Blob::new(payload))
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

fn log_invoke_output(invoke: &InvokeOutput, message: &str) {
    if let Some(payload) = invoke.payload().cloned() {
        let payload = String::from_utf8(payload.into_inner());
        info!(?payload, message);
    } else {
        info!("Could not extract payload")
    }
    if let Some(logs) = invoke.log_result() {
        debug!(?logs, "Invoked function logs")
    } else {
        debug!("Invoked function had no logs")
    }
}
```

- Consulte detalhes da API em Invoke na Referência da API AWS SDK para Rust.

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** List all Lambda functions in the current Region. */
    pub async fn list_functions(&self) -> Result<ListFunctionsOutput, anyhow::Error>
 {
        info!("Listing lambda functions");
```

```
        self.lambda_client
            .list_functions()
            .send()
            .await
            .map_err(anyhow::Error::from)
    }
```

- Para obter detalhes da API, consulte a ListFunctions referência da API AWS SDK for Rust.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /** Given a Path to a zip file, update the function's code and wait for the
update to finish. */
    pub async fn update_function_code(
        &self,
        zip_file: PathBuf,
        key: String,
    ) -> Result<UpdateFunctionCodeOutput, anyhow::Error> {
        let function_code = self.prepare_function(zip_file, Some(key)).await?;

        info!("Updating code for {}", self.lambda_name);
        let update = self
            .lambda_client
            .update_function_code()
            .function_name(self.lambda_name.clone())
            .s3_bucket(self.bucket.clone())
            .s3_key(function_code.s3_key().unwrap().to_string())
            .send()
            .await
            .map_err(anyhow::Error::from)?;
```

```
        self.wait_for_function_ready().await?;

        Ok(update)
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --output-
format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }
```

- Para obter detalhes da API, consulte a UpdateFunctionCode referência da API AWS SDK for Rust.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/** Update the environment for a function. */
pub async fn update_function_configuration(
    &self,
    environment: Environment,
) -> Result<UpdateFunctionConfigurationOutput, anyhow::Error> {
    info!(
        ?environment,
        "Updating environment for {}", self.lambda_name
    );
    let updated = self
        .lambda_client
        .update_function_configuration()
        .function_name(self.lambda_name.clone())
        .environment(environment)
        .send()
        .await
        .map_err(anyhow::Error::from)?;

    self.wait_for_function_ready().await?;

    Ok(updated)
}
```

- Para obter detalhes da API, consulte a UpdateFunctionConfiguration referência da API AWS SDK for Rust.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Rust.

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
```

```
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
```

```
        port = port,
        db_user = db_username
    );

    let signable_request =
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
```

```
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do Kinesis com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
use aws_lambda_events::event::kinesis::KinesisEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) -> Result<(), Error> {
    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    event.payload.records.iter().for_each(|record| {
        tracing::info!("EventId:
 {}",record.event_id.as_deref().unwrap_or_default());

        let record_data = std::str::from_utf8(&record.kinesis.data);

        match record_data {
            Ok(data) => {
                // log the record data
                tracing::info!("Data: {}", data);
            }
            Err(e) => {
                tracing::error!("Error: {}", e);
            }
        }
    });

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();
```

```
    run(service_fn(function_handler)).await
}
```

Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como consumir um evento do DynamoDB com o Lambda usando Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<Event>) ->Result<(), Error> {

    let records = &event.payload.records;
    tracing::info!("event payload: {:?}",records);
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
```

```
        return Ok(());
    }

    for record in records{
        log_dynamo_dbrecord(record);
    }

    tracing::info!("Dynamo db records processed");

    // Prepare the response
    Ok(())

}

fn log_dynamo_dbrecord(record: &EventRecord)-> Result<(), Error>{
    tracing::info!("EventId: {}", record.event_id);
    tracing::info!("EventName: {}", record.event_name);
    tracing::info!("DynamoDB Record: {:?}", record.change );
    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um fluxo de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do Amazon DocumentDB com o Lambda usando Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::documentdb::{DocumentDbEvent, DocumentDbInnerEvent},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<DocumentDbEvent>) ->Result<(), Error> {

    tracing::info!("Event Source ARN: {:?}", event.payload.event_source_arn);
    tracing::info!("Event Source: {:?}", event.payload.event_source);

    let records = &event.payload.events;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_document_db_event(record);
    }

    tracing::info!("Document db records processed");
```

```
    // Prepare the response
    Ok(())

}

fn log_document_db_event(record: &DocumentDbInnerEvent)-> Result<(), Error>{
    tracing::info!("Change Event: {:?}", record.event);

    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando o Rust.

```
use aws_lambda_events::event::kafka::KafkaEvent;
use lambda_runtime::{run, service_fn, tracing, Error, LambdaEvent};
use base64::prelude::*;
use serde_json::{Value};
use tracing::{info};

/// Pre-Requisites:
/// 1. Install Cargo Lambda - see https://www.cargo-lambda.info/guide/getting-
started.html
/// 2. Add packages tracing, tracing-subscriber, serde_json, base64
///
/// This is the main body for the function.
/// Write your code inside it.
/// There are some code example in the following URLs:
/// - https://github.com/awslabs/aws-lambda-rust-runtime/tree/main/examples
/// - https://github.com/aws-samples/serverless-rust-demo/

async fn function_handler(event: LambdaEvent<KafkaEvent>) -> Result<Value, Error> {

    let payload = event.payload.records;

    for (_name, records) in payload.iter() {

        for record in records {

         let record_text = record.value.as_ref().ok_or("Value is None")?;
         info!("Record: {}", &record_text);

         // perform Base64 decoding
         let record_bytes = BASE64_STANDARD.decode(record_text)?;
         let message = std::str::from_utf8(&record_bytes)?;

         info!("Message: {}", message);
        }

    }

    Ok(().into())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
```

```
    // required to enable CloudWatch error logging by the runtime
    tracing::init_default_subscriber();
    info!("Setup CW subscriber!");

    run(service_fn(function_handler)).await
}
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();
```

```
    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket name
 to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
```

```
}
```

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SNS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features =
 ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}
```

```
fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
```

```
async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
 record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relate falhas de itens em lote do Kinesis com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
use aws_lambda_events::{
    event::kinesis::KinesisEvent,
    kinesis::KinesisEventRecord,
    streams::{KinesisBatchItemFailure, KinesisEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) ->
 Result<KinesisEventResponse, Error> {
    let mut response = KinesisEventResponse {
        batch_item_failures: vec![],
    };

    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in &event.payload.records {
        tracing::info!(
            "EventId: {}",
            record.event_id.as_deref().unwrap_or_default()
        );

        let record_processing_result = process_record(record);

        if record_processing_result.is_err() {
            response.batch_item_failures.push(KinesisBatchItemFailure {
                item_identifier: record.kinesis.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed item
 onwards. */
            return Ok(response);
        }
    }

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(response)
```

```
}

fn process_record(record: &KinesisEventRecord) -> Result<(), Error> {
    let record_data = std::str::from_utf8(record.kinesis.data.as_slice());

    if let Some(err) = record_data.err() {
        tracing::error!("Error: {}", err);
        return Err(Error::from(err));
    }

    let record_data = record_data.unwrap_or_default();

    // do something interesting with the data
    tracing::info!("Data: {}", record_data);

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um fluxo do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando Rust.

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord, StreamRecord},
    streams::{DynamoDbBatchItemFailure, DynamoDbEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Process the stream record
fn process_record(record: &EventRecord) -> Result<(), Error> {
    let stream_record: &StreamRecord = &record.change;

    // process your stream record here...
    tracing::info!("Data: {:?}", stream_record);

    Ok(())
}

/// Main Lambda handler here...
async fn function_handler(event: LambdaEvent<Event>) ->
 Result<DynamoDbEventResponse, Error> {
    let mut response = DynamoDbEventResponse {
        batch_item_failures: vec![],
    };

    let records = &event.payload.records;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in records {
        tracing::info!("EventId: {}", record.event_id);
```

```
        // Couldn't find a sequence number
        if record.change.sequence_number.is_none() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: Some("".to_string()),
            });
            return Ok(response);
        }

        // Process your record here...
        if process_record(record).is_err() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: record.change.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed item
 onwards. */
            return Ok(response);
        }
    }

    tracing::info!("Successfully processed {} record(s)", records.len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<SqsBatchResponse,
 Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
        }
    }

    Ok(SqsBatchResponse {
        batch_item_failures,
```

```
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(function_handler)).await
}
```

AWS contribuições da comunidade

Compilar e testar uma aplicação com tecnologia sem servidor

O exemplo de código a seguir mostra como criar e testar uma aplicação com tecnologia sem servidor usando o API Gateway com o Lambda e o DynamoDB.

SDK para Rust

Mostra como compilar e testar uma aplicação com tecnologia sem servidor que consiste em um API Gateway com o Lambda e o DynamoDB usando o SDK Rust.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda

MediaLive exemplos de uso do SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com. MediaLive

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListInputs`

O código de exemplo a seguir mostra como usar `ListInputs`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Liste seus nomes MediaLive de entrada e ARNs na região.

```
async fn show_inputs(client: &Client) -> Result<(), Error> {
    let input_list = client.list_inputs().send().await?;

    for i in input_list.inputs() {
        let input_arn = i.arn().unwrap_or_default();
        let input_name = i.name().unwrap_or_default();

        println!("Input Name : {}", input_name);
        println!("Input ARN : {}", input_arn);
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListInputs referência da API AWS SDK for Rust.

MediaPackage exemplos de uso do SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com. MediaPackage

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListChannels`

O código de exemplo a seguir mostra como usar `ListChannels`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar canais ARNs e as descrições.

```
async fn show_channels(client: &Client) -> Result<(), Error> {
    let list_channels = client.list_channels().send().await?;

    println!("Channels:");

    for c in list_channels.channels() {
        let description = c.description().unwrap_or_default();
        let arn = c.arn().unwrap_or_default();
```

```
        println!("  Description : {}", description);
        println!("  ARN :         {}", arn);
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListChannels referência da API AWS SDK for Rust.

`ListOriginEndpoints`

O código de exemplo a seguir mostra como usar `ListOriginEndpoints`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Listar descrições de endpoint e URLs.

```
async fn show_endpoints(client: &Client) -> Result<(), Error> {
    let or_endpoints = client.list_origin_endpoints().send().await?;

    println!("Endpoints:");

    for e in or_endpoints.origin_endpoints() {
        let endpoint_url = e.url().unwrap_or_default();
        let endpoint_description = e.description().unwrap_or_default();
        println!("  Description: {}", endpoint_description);
        println!("  URL :        {}", endpoint_url);
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListOriginEndpoints referência da API AWS SDK for Rust.

Exemplos do Amazon MSK usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon MSK.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Invocar uma função do Lambda em um gatinho do Amazon MSK

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de registros de um cluster do Amazon MSK. A função recupera a carga útil do MSK e registra em log o conteúdo dos registros.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumo de um evento do Amazon MSK com o Lambda usando o Rust.

```
use aws_lambda_events::event::kafka::KafkaEvent;
use lambda_runtime::{run, service_fn, tracing, Error, LambdaEvent};
use base64::prelude::*;
use serde_json::{Value};
use tracing::{info};

/// Pre-Requisites:
```

```
/// 1. Install Cargo Lambda - see https://www.cargo-lambda.info/guide/getting-
started.html
/// 2. Add packages tracing, tracing-subscriber, serde_json, base64
///
/// This is the main body for the function.
/// Write your code inside it.
/// There are some code example in the following URLs:
/// - https://github.com/awslabs/aws-lambda-rust-runtime/tree/main/examples
/// - https://github.com/aws-samples/serverless-rust-demo/

async fn function_handler(event: LambdaEvent<KafkaEvent>) -> Result<Value, Error> {

    let payload = event.payload.records;

    for (_name, records) in payload.iter() {

        for record in records {

         let record_text = record.value.as_ref().ok_or("Value is None")?;
         info!("Record: {}", &record_text);

         // perform Base64 decoding
         let record_bytes = BASE64_STANDARD.decode(record_text)?;
         let message = std::str::from_utf8(&record_bytes)?;

         info!("Message: {}", message);
        }

    }

    Ok(().into())
}

#[tokio::main]
async fn main() -> Result<(), Error> {

    // required to enable CloudWatch error logging by the runtime
    tracing::init_default_subscriber();
    info!("Setup CW subscriber!");

    run(service_fn(function_handler)).await
}
```

Exemplos do Amazon Polly usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon Polly.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DescribeVoices`

O código de exemplo a seguir mostra como usar `DescribeVoices`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn list_voices(client: &Client) -> Result<(), Error> {
    let resp = client.describe_voices().send().await?;

    println!("Voices:");
```

```
    let voices = resp.voices();
    for voice in voices {
        println!("  Name:     {}", voice.name().unwrap_or("No name!"));
        println!(
            "  Language: {}",
            voice.language_name().unwrap_or("No language!")
        );

        println!();
    }

    println!("Found {} voices", voices.len());

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeVoices referência da API AWS SDK for Rust.

`ListLexicons`

O código de exemplo a seguir mostra como usar `ListLexicons`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_lexicons(client: &Client) -> Result<(), Error> {
    let resp = client.list_lexicons().send().await?;

    println!("Lexicons:");

    let lexicons = resp.lexicons();

    for lexicon in lexicons {
        println!("  Name:     {}", lexicon.name().unwrap_or_default());
        println!(
            "  Language: {:?}\n",
```

```
            lexicon
                .attributes()
                .as_ref()
                .map(|attrib| attrib
                    .language_code
                    .as_ref()
                    .expect("languages must have language codes"))
                .expect("languages must have attributes")
        );
    }

    println!();
    println!("Found {} lexicons.", lexicons.len());
    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListLexicons referência da API AWS SDK for Rust.

`PutLexicon`

O código de exemplo a seguir mostra como usar `PutLexicon`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_lexicon(client: &Client, name: &str, from: &str, to: &str) ->
 Result<(), Error> {
    let content = format!("<?xml version=\"1.0\" encoding=\"UTF-8\"?>
    <lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/pronunciation-lexicon
\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
    xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon http://
www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\"
    alphabet=\"ipa\" xml:lang=\"en-US\">
    <lexeme><grapheme>{}</grapheme><alias>{}</alias></lexeme>
```

```
    </lexicon>", from, to);

    client
        .put_lexicon()
        .name(name)
        .content(content)
        .send()
        .await?;

    println!("Added lexicon");

    Ok(())
}
```

- Para obter detalhes da API, consulte a PutLexicon referência da API AWS SDK for Rust.

`SynthesizeSpeech`

O código de exemplo a seguir mostra como usar `SynthesizeSpeech`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn synthesize(client: &Client, filename: &str) -> Result<(), Error> {
    let content = fs::read_to_string(filename);

    let resp = client
        .synthesize_speech()
        .output_format(OutputFormat::Mp3)
        .text(content.unwrap())
        .voice_id(VoiceId::Joanna)
        .send()
        .await?;

    // Get MP3 data from response and save it
    let mut blob = resp
```

```
        .audio_stream
        .collect()
        .await
        .expect("failed to read data");

    let parts: Vec<&str> = filename.split('.').collect();
    let out_file = format!("{}{}", String::from(parts[0]), ".mp3");

    let mut file = tokio::fs::File::create(out_file)
        .await
        .expect("failed to create file");

    file.write_all_buf(&mut blob)
        .await
        .expect("failed to write to file");

    Ok(())
}
```

- Para obter detalhes da API, consulte a SynthesizeSpeech referência da API AWS SDK for Rust.

Cenários

Converter texto em fala e de volta em texto

O exemplo de código a seguir mostra como:

- Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio.
- Fazer upload do arquivo de áudio para um bucket do Amazon S3.
- Usar o Amazon Transcribe para converter o arquivo de áudio em texto.
- Exibir o texto.

SDK para Rust

Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio, fazer upload do arquivo de áudio em um bucket do Amazon S3, usar o Amazon Transcribe para converter esse arquivo de áudio em texto e exibir o texto.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Exemplos do Amazon RDS usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon RDS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Exemplos sem servidor

Exemplos sem servidor

Como se conectar a um banco de dados do Amazon RDS em uma função do Lambda

O exemplo de código a seguir mostra como implementar uma função do Lambda que se conecte a um banco de dados do RDS. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Conectar-se a um banco de dados do Amazon RDS em uma função do Lambda usando Rust.

```
use aws_config::BehaviorVersion;
```

```
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
```

```
        db_hostname = db_hostname,
        port = port,
        db_user = db_username
    );

    let signable_request =
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
```

```
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

Exemplos do Amazon RDS Data Service usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon RDS Data Service.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn query_cluster(
    client: &Client,
    cluster_arn: &str,
    query: &str,
    secret_arn: &str,
) -> Result<(), Error> {
    let st = client
        .execute_statement()
        .resource_arn(cluster_arn)
        .database("postgres") // Do not confuse this with db instance name
        .sql(query)
        .secret_arn(secret_arn);

    let result = st.send().await?;

    println!("{:?}", result);
    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a ExecuteStatement referência da API AWS SDK for Rust.

Exemplos do Amazon Rekognition usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon Rekognition.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar faces em uma imagem

O exemplo de código a seguir mostra como:

- Salvar uma imagem em um bucket do Amazon S3.
- Usar o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.).
- Exibir esses detalhes.

SDK para Rust

Salve a imagem em um bucket do Amazon S3 com um prefixo uploads, use o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.), e exiba esses detalhes.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3

Salvar o EXIF e outras informações de imagem

O exemplo de código a seguir mostra como:

- Obter informações de EXIF de um arquivo JPG, JPEG ou PNG.
- Fazer upload do arquivo de imagem para um bucket do Amazon S3.
- Usar o Amazon Rekognition para identificar os três principais atributos (rótulos) no arquivo.
- Adicionar as informações de EXIF e rótulo a uma tabela do Amazon DynamoDB na região.

SDK para Rust

Obtenha informações de EXIF de um arquivo JPG, JPEG ou PNG, faça upload do arquivo de imagem para um bucket do Amazon S3, use o Amazon Rekognition para identificar os três principais atributos (rótulos no Amazon Rekognition) no arquivo e adicione as informações de EXIF e de rótulo a uma tabela do Amazon DynamoDB na região.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Exemplos de Route 53 usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Route 53.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListHostedZones`

O código de exemplo a seguir mostra como usar `ListHostedZones`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_host_info(client: &aws_sdk_route53::Client) -> Result<(),
 aws_sdk_route53::Error> {
    let hosted_zone_count = client.get_hosted_zone_count().send().await?;

    println!(
        "Number of hosted zones in region : {}",
        hosted_zone_count.hosted_zone_count(),
    );

    let hosted_zones = client.list_hosted_zones().send().await?;

    println!("Zones:");

    for hz in hosted_zones.hosted_zones() {
        let zone_name = hz.name();
        let zone_id = hz.id();

        println!("  ID :   {}", zone_id);
        println!("  Name : {}", zone_name);
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListHostedZones referência da API AWS SDK for Rust.

Exemplos do Amazon S3 usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações
- Cenários
- Exemplos sem servidor

Conceitos básicos

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
 present in the region the `client` is configured to use.
```

```
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the bucket.
async fn list_bucket_and_upload_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
        .bucket(bucket)
        .into_paginator()
        .send();

    println!("key\tetag\tlast_modified\tstorage_class");
    while let Some(Ok(object)) = objects.next().await {
        for item in object.contents() {
            println!(
                "{}\t{}\t{}\t{}",
                item.key().unwrap_or_default(),
                item.e_tag().unwrap_or_default(),
                item.last_modified()
                    .map(|lm| format!("{lm}"))
                    .unwrap_or_default(),
                item.storage_class()
                    .map(|sc| format!("{sc}"))
                    .unwrap_or_default()
            );
        }
    }

    // Prepare a ByteStream around the file, and upload the object using that
 ByteStream.
    let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
        .await
        .map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to create bytestream for {filepath:?} ({err:?})"
            ))
        })?;
    let resp = client
        .put_object()
        .bucket(bucket)
```

```
        .key(key)
        .body(body)
        .send()
        .await?;

    println!(
        "Upload success. Version: {:?}",
        resp.version_id()
            .expect("S3 Object upload missing version ID")
    );

    // Retrieve the just-uploaded object.
    let resp = client.get_object().bucket(bucket).key(key).send().await?;
    println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
    println!("version: {}", resp.version_id().unwrap_or("(missing)"));

    Ok(())
}
```

S3ExampleError utilitários.

```
/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
```

```
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

- Para obter detalhes da API, consulte a ListBuckets referência da API AWS SDK for Rust.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Código para a caixa binária que executa o cenário.

```
#![allow(clippy::result_large_err)]

//!  Purpose
//!  Shows how to use the AWS SDK for Rust to get started using
//!  Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects into
 and out of it,
//!  and delete all resources at the end of the demo.
//!
//!  This example follows the steps in "Getting started with Amazon S3" in the
 Amazon S3
//!  user guide.
//!  - https://docs.aws.amazon.com/AmazonS3/latest/userguide/GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config = aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name, key,
 target_key).await
    {
```

```
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), S3ExampleError> {
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
    let run_example: Result<(), S3ExampleError> = (async {
        s3_code_examples::upload_object(&client, &bucket_name, &file_name,
 &key).await?;
        let _object = s3_code_examples::download_object(&client, &bucket_name,
 &key).await;
        s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
 &target_key)
            .await?;
        s3_code_examples::list_objects(&client, &bucket_name).await?;
        s3_code_examples::clear_bucket(&client, &bucket_name).await?;
        Ok(())
    })
    .await;
    if let Err(err) = run_example {
        eprintln!("Failed to complete getting-started example: {err:?}");
    }
    s3_code_examples::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}
```

Ações comuns usadas na situação.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
```

```
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
 S3ExampleError> {
    let constraint =
 aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
 task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
 se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
 aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
```

```
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {
    client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await
        .map_err(S3ExampleError::from)
}

/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object} with
 etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
 aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
```

```
    );
    Ok(())
}

pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) -> Result<(),
 S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();
```

```
    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;

    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(S3ExampleError::new(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Rust.

- CopyObject
- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Ações

`CompleteMultipartUpload`

O código de exemplo a seguir mostra como usar `CompleteMultipartUpload`.

SDK para Rust

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
   let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
       .set_parts(Some(upload_parts))
       .build();

   let _complete_multipart_upload_res = client
       .complete_multipart_upload()
       .bucket(&bucket_name)
       .key(&key)
       .multipart_upload(completed_multipart_upload)
       .upload_id(upload_id)
       .send()
       .await?;
```

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
```

```
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

- Para obter detalhes da API, consulte a CompleteMultipartUpload referência da API AWS SDK for Rust.

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
```

```
        "Copied from {source_key} to {destination_bucket}/{destination_object} with
 etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
 aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}
```

- Para obter detalhes da API, consulte a CopyObject referência da API AWS SDK for Rust.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
 S3ExampleError> {
    let constraint =
 aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
```

```
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
 task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
 se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}
```

- Para obter detalhes da API, consulte a CreateBucket referência da API AWS SDK for Rust.

`CreateMultipartUpload`

O código de exemplo a seguir mostra como usar `CreateMultipartUpload`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
```

```
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
 CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

- Para obter detalhes da API, consulte a CreateMultipartUpload referência da API AWS SDK for Rust.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
```

```
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
```

- Para obter detalhes da API, consulte a DeleteBucket referência da API AWS SDK for Rust.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// Delete an object from a bucket.
pub async fn remove_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    key: &str,
) -> Result<(), S3ExampleError> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    // There are no modeled errors to handle when deleting an object.
```

```
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteObject referência da API AWS SDK for Rust.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use `?` early return errors while building object
 keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for delete_object:
 {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }

    client
        .delete_objects()
        .bucket(bucket_name)
```

```
        .delete(
            aws_sdk_s3::types::Delete::builder()
                .set_objects(Some(delete_object_ids))
                .build()
                .map_err(|err| {
                    S3ExampleError::new(format!("Failed to build delete_object input
 {err:?}"))
                })?,
        )
        .send()
        .await?;
    Ok(())
}
```

- Para obter detalhes da API, consulte a DeleteObjects referência da API AWS SDK for Rust.

`GetBucketLocation`

O código de exemplo a seguir mostra como usar `GetBucketLocation`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
```

```
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetBucketLocation referência da API AWS SDK for Rust.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}"
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
 {err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }

    Ok(byte_count)
```

```
}
```

- Para obter detalhes da API, consulte a GetObject referência da API AWS SDK for Rust.

`ListBuckets`

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
```

```
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListBuckets referência da API AWS SDK for Rust.

`ListObjectVersions`

O código de exemplo a seguir mostra como usar `ListObjectVersions`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
```

```
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListObjectVersions referência da API AWS SDK for Rust.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) -> Result<(),
 S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
```

```
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte ListObjects V2 no AWS SDK para referência da API Rust.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
 aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}
```

- Para obter detalhes da API, consulte a PutObject referência da API AWS SDK for Rust.

`UploadPart`

O código de exemplo a seguir mostra como usar `UploadPart`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
```

```
                .part_number(part_number)
                .build(),
        );
    }
```

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

- Para obter detalhes da API, consulte a UploadPart referência da API AWS SDK for Rust.

Cenários

Converter texto em fala e de volta em texto

O exemplo de código a seguir mostra como:

- Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio.
- Fazer upload do arquivo de áudio para um bucket do Amazon S3.
- Usar o Amazon Transcribe para converter o arquivo de áudio em texto.
- Exibir o texto.

SDK para Rust

Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio, fazer upload do arquivo de áudio em um bucket do Amazon S3, usar o Amazon Transcribe para converter esse arquivo de áudio em texto e exibir o texto.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie solicitações de pré-assinatura para objetos GET do S3.

```
/// Generate a URL for a presigned GET request.
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());
    let valid_until = chrono::offset::Local::now() + expires_in;
    println!("Valid until: {valid_until}");

    Ok(())
}
```

Crie solicitações de pré-assinatura para objetos PUT do S3.

```
async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<String, S3ExampleError> {
    let expires_in: std::time::Duration =
 std::time::Duration::from_secs(expires_in);
    let expires_in: aws_sdk_s3::presigning::PresigningConfig =
        PresigningConfig::expires_in(expires_in).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to convert expiration to PresigningConfig: {err:?}"
            ))
        })?;
    let presigned_request = client
        .put_object()
```

```
        .bucket(bucket)
        .key(object)
        .presigned(expires_in)
        .await?;

    Ok(presigned_request.uri().into())
}
```

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Detectar faces em uma imagem

O exemplo de código a seguir mostra como:

- Salvar uma imagem em um bucket do Amazon S3.

- Usar o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.).
- Exibir esses detalhes.

SDK para Rust

Salve a imagem em um bucket do Amazon S3 com um prefixo uploads, use o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.), e exiba esses detalhes.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3

Obter um objeto de um bucket que foi modificado

O exemplo de código a seguir mostra como ler dados de um objeto em um bucket do S3, mas somente se esse bucket não tiver sido modificado desde a última recuperação.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};
use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};

const KEY: &str = "key";
```

```
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{uuid}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
```

```
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };

    // Request the object's headers.
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await;

    // If the `HeadObject` request succeeded, create a tuple containing the
    // values of the headers `last-modified` and `etag`. If the request
    // failed, return the error in a tuple instead.
    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => (Err(err), String::new()),
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and first GetObject had differing eTags"
    );

    println!("First value of last_modified: {last_modified:?}");
    println!("First tag: {}\n", e_tag_1);

    // Send a second `HeadObject` request. This time, the `if_modified_since`
    // option is specified, giving the `last_modified` value returned by the
    // first call to `HeadObject`.
    //
    // Since the object hasn't been changed, and there are no other objects in
    // the bucket, there should be no matching objects.

    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
```

```
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

    // If the `HeadObject` request succeeded, the result is a typle containing
    // the `last_modified` and `e_tag_1` properties. This is _not_ the expected
    // result.
    //
    // The _expected_ result of the second call to `HeadObject` is an
    // `SdkError::ServiceError` containing the HTTP error response. If that's
    // the case and the HTTP status is 304 (not modified), the output is a
    // tuple containing the values of the HTTP `last-modified` and `etag`
    // headers.
    //
    // If any other HTTP error occurred, the error is returned as an
    // `SdkError::ServiceError`.

    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => match err {
            SdkError::ServiceError(err) => {
                // Get the raw HTTP response. If its status is 304, the
                // object has not changed. This is the expected code path.
                let http = err.raw();
                match http.status().as_u16() {
                    // If the HTTP status is 304: Not Modified, return a
                    // tuple containing the values of the HTTP
                    // `last-modified` and `etag` headers.
                    304 => (
                        Ok(DateTime::from_str(
                            http.headers().get("last-modified").unwrap(),
                            DateTimeFormat::HttpDate,
                        )
                        .unwrap()),
                        http.headers().get("etag").map(|t| t.into()).unwrap(),
                    ),
                    // Any other HTTP status code is returned as an
                    // `SdkError::ServiceError`.
                    _ => (Err(SdkError::ServiceError(err)), String::new()),
                }
```

```
            }
            // Any other kind of error is returned in a tuple containing the
            // error and an empty string.
            _ => (Err(err), String::new()),
        },
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and second HeadObject had different eTags"
    );

    println!("Second value of last modified: {last_modified:?}");
    println!("Second tag: {}", e_tag_2);

    // Clean up by deleting the object and the bucket.
    client
        .delete_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await?;

    client
        .delete_bucket()
        .bucket(bucket_name.as_str())
        .send()
        .await?;

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetObject referência da API AWS SDK for Rust.

Salvar o EXIF e outras informações de imagem

O exemplo de código a seguir mostra como:

- Obter informações de EXIF de um arquivo JPG, JPEG ou PNG.
- Fazer upload do arquivo de imagem para um bucket do Amazon S3.

- Usar o Amazon Rekognition para identificar os três principais atributos (rótulos) no arquivo.
- Adicionar as informações de EXIF e rótulo a uma tabela do Amazon DynamoDB na região.

SDK para Rust

Obtenha informações de EXIF de um arquivo JPG, JPEG ou PNG, faça upload do arquivo de imagem para um bucket do Amazon S3, use o Amazon Rekognition para identificar os três principais atributos (rótulos no Amazon Rekognition) no arquivo e adicione as informações de EXIF e de rótulo a uma tabela do Amazon DynamoDB na região.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Teste de unidade e integração com SDK

O exemplo de código a seguir mostra exemplos de técnicas de melhores práticas ao escrever testes de unidade e de integração usando um AWS SDK.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Cargo.toml para testar exemplos.

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
```

```
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"

[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-
credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
path = "src/main.rs"
```

Exemplo de teste de unidade usando automock e um wrapper de serviço.

```
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
```

```
}

#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
```

```
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
```

```
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .set_next_continuation_token(Some("next".to_string()))
                    .build())
            });
        mock.expect_list_objects()
            .with(
                eq("test-bucket"),
                eq("test-prefix"),
                eq(Some("next".to_string())),
            )
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(3).build(),
                        s3::types::Object::builder().size(9).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
    }
}
```

Exemplo de teste de integração usando StaticReplayClient.

```
use aws_sdk_s3 as s3;
```

```
#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}
```

```
#[cfg(test)]
mod test {
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
 StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/response_1.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }

    #[tokio::test]
```

```
    async fn test_multiple_pages() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
                    .unwrap(),
            );
        let page_2 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
```

```
        replay_client.assert_requests_match(&[]);
    }
}
```

Fazer upload ou download de arquivos grandes

O exemplo de código a seguir mostra como fazer upload ou download de arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte Carregar um objeto usando carregamento fracionado.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
 get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
```

```
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}

async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
```

```
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }

    if file_size == 0 {
        return Err(S3ExampleError::new("Bad file size."));
    }
    if chunk_count > MAX_CHUNKS {
        return Err(S3ExampleError::new(
            "Too many chunks! Try increasing your chunk size.",
        ));
    }

    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();
```

```
        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }

    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;

    let data: GetObjectOutput =
        s3_code_examples::download_object(&client, &bucket_name, &key).await?;
    let data_length: u64 = data
        .content_length()
        .unwrap_or_default()
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
```

```
    } else {
        println!("The data was not the same size!");
    }

    s3_code_examples::clear_bucket(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_code_examples::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}
```

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consumir um evento do S3 com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
```

```
/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket name
 to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;
```

```
    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

SageMaker Exemplos de IA usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com SageMaker IA.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListNotebookInstances`

O código de exemplo a seguir mostra como usar `ListNotebookInstances`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_instances(client: &Client) -> Result<(), Error> {
    let notebooks = client.list_notebook_instances().send().await?;

    println!("Notebooks:");

    for n in notebooks.notebook_instances() {
        let n_instance_type = n.instance_type().unwrap();
        let n_status = n.notebook_instance_status().unwrap();
        let n_name = n.notebook_instance_name();

        println!("  Name :          {}", n_name.unwrap_or("Unknown"));
        println!("  Status :        {}", n_status.as_ref());
        println!("  Instance Type : {}", n_instance_type.as_ref());
        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListNotebookInstances referência da API AWS SDK for Rust.

`ListTrainingJobs`

O código de exemplo a seguir mostra como usar `ListTrainingJobs`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_jobs(client: &Client) -> Result<(), Error> {
    let job_details = client.list_training_jobs().send().await?;

    println!("Jobs:");

    for j in job_details.training_job_summaries() {
```

```
        let name = j.training_job_name().unwrap_or("Unknown");
        let creation_time = j.creation_time().expect("creation
 time").to_chrono_utc()?;
        let training_end_time = j
            .training_end_time()
            .expect("Training end time")
            .to_chrono_utc()?;

        let status = j.training_job_status().expect("training status");
        let duration = training_end_time - creation_time;

        println!("  Name:               {}", name);
        println!(
            "  Creation date/time: {}",
            creation_time.format("%Y-%m-%d@%H:%M:%S")
        );
        println!("  Duration (seconds): {}", duration.num_seconds());
        println!("  Status:             {:?}", status);

        println!();
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListTrainingJobs referência da API AWS SDK for Rust.

Exemplos de Secrets Manager usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetSecretValue`

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_secret(client: &Client, name: &str) -> Result<(), Error> {
    let resp = client.get_secret_value().secret_id(name).send().await?;

    println!("Value: {}", resp.secret_string().unwrap_or("No value!"));

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetSecretValue referência da API AWS SDK for Rust.

Exemplos API v2 do Amazon SES usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com a API Amazon SES v2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateContact`

O código de exemplo a seguir mostra como usar `CreateContact`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn add_contact(client: &Client, list: &str, email: &str) -> Result<(), Error>
 {
    client
        .create_contact()
        .contact_list_name(list)
        .email_address(email)
        .send()
        .await?;

    println!("Created contact");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateContact referência da API AWS SDK for Rust.

`CreateContactList`

O código de exemplo a seguir mostra como usar `CreateContactList`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_list(client: &Client, contact_list: &str) -> Result<(), Error> {
    client
        .create_contact_list()
        .contact_list_name(contact_list)
        .send()
        .await?;

    println!("Created contact list.");

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateContactList referência da API AWS SDK for Rust.

`CreateEmailIdentity`

O código de exemplo a seguir mostra como usar `CreateEmailIdentity`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
```

```
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }
```

- Para obter detalhes da API, consulte a CreateEmailIdentity referência da API AWS SDK for Rust.

`CreateEmailTemplate`

O código de exemplo a seguir mostra como usar `CreateEmailTemplate`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
```

```
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();

        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }
```

- Para obter detalhes da API, consulte a CreateEmailTemplate referência da API AWS SDK for Rust.

`DeleteContactList`

O código de exemplo a seguir mostra como usar `DeleteContactList`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }
```

- Para obter detalhes da API, consulte a DeleteContactList referência da API AWS SDK for Rust.

`DeleteEmailIdentity`

O código de exemplo a seguir mostra como usar `DeleteEmailIdentity`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }
```

- Para obter detalhes da API, consulte a DeleteEmailIdentity referência da API AWS SDK for Rust.

`DeleteEmailTemplate`

O código de exemplo a seguir mostra como usar `DeleteEmailTemplate`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        match self
            .client
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted successfully.")?,
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- Para obter detalhes da API, consulte a DeleteEmailTemplate referência da API AWS SDK for Rust.

`GetEmailIdentity`

O código de exemplo a seguir mostra como usar `GetEmailIdentity`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Determina se um endereço de e-mail foi verificado.

```
async fn is_verified(client: &Client, email: &str) -> Result<(), Error> {
    let resp = client
        .get_email_identity()
        .email_identity(email)
        .send()
        .await?;

    if resp.verified_for_sending_status() {
        println!("The address is verified");
    } else {
        println!("The address is not verified");
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a GetEmailIdentity referência da API AWS SDK for Rust.

`ListContactLists`

O código de exemplo a seguir mostra como usar `ListContactLists`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_lists(client: &Client) -> Result<(), Error> {
    let resp = client.list_contact_lists().send().await?;

    println!("Contact lists:");

    for list in resp.contact_lists() {
        println!("  {}", list.contact_list_name().unwrap_or_default());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListContactLists referência da API AWS SDK for Rust.

`ListContacts`

O código de exemplo a seguir mostra como usar `ListContacts`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_contacts(client: &Client, list: &str) -> Result<(), Error> {
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    println!("Contacts:");

    for contact in resp.contacts() {
        println!("  {}", contact.email_address().unwrap_or_default());
    }

    Ok(())
```

```
}
```

- Para obter detalhes da API, consulte a ListContacts referência da API AWS SDK for Rust.

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem a todos os membros da lista de contatos.

```
async fn send_message(
    client: &Client,
    list: &str,
    from: &str,
    subject: &str,
    message: &str,
) -> Result<(), Error> {
    // Get list of email addresses from contact list.
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    let contacts = resp.contacts();

    let cs: Vec<String> = contacts
        .iter()
        .map(|i| i.email_address().unwrap_or_default().to_string())
        .collect();

    let mut dest: Destination = Destination::builder().build();
    dest.to_addresses = Some(cs);
```

```
    let subject_content = Content::builder()
        .data(subject)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body_content = Content::builder()
        .data(message)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body = Body::builder().text(body_content).build();

    let msg = Message::builder()
        .subject(subject_content)
        .body(body)
        .build();

    let email_content = EmailContent::builder().simple(msg).build();

    client
        .send_email()
        .from_email_address(from)
        .destination(dest)
        .content(email_content)
        .send()
        .await?;

    println!("Email sent to list");

    Ok(())
}
```

Envia uma mensagem a todos os membros da lista de contatos usando um modelo.

```
            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
```

```
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

.destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}: {}",
email, e)),
            }
```

- Para obter detalhes da API, consulte a SendEmail referência da API AWS SDK for Rust.

Cenários

Cenário de boletim informativo

O exemplo de código a seguir mostra como executar o cenário do boletim informativo da API v2 do Amazon SES.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        match self
            .client
            .create_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateContactListError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Contact list already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating contact list: {}", e)),
            },
        }

            match self
                .client
                .create_contact()
                .contact_list_name(CONTACT_LIST_NAME)
                .email_address(email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Contact created for {}", email)?,
                Err(e) => match e.into_service_error() {
                    CreateContactError::AlreadyExistsException(_) => writeln!(
                        self.stdout,
                        "Contact already exists for {}, skipping creation.",
                        email
                    )?,
```

```
                    e => return Err(anyhow!("Error creating contact for {}: {}",
 email, e)),
                },
            }

        let contacts: Vec<Contact> = match self
            .client
            .list_contacts()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(list_contacts_output) => {
                list_contacts_output.contacts.unwrap().into_iter().collect()
            }
            Err(e) => {
                return Err(anyhow!(
                    "Error retrieving contact list {}: {}",
                    CONTACT_LIST_NAME,
                    e
                ))
            }
        };

            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

 .destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
```

```
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}: {}",
 email, e)),
            }

        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }

        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
```

```
            .unwrap_or_else(|_| "Missing coupon-newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-newsletter.txt")
            .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();

        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }

        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }

            match self
```

```
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }

        match self
            .client
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted successfully.")?,
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Exemplos do Amazon SNS usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários
- Exemplos sem servidor

Ações

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
```

```
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Para obter detalhes da API, consulte a CreateTopic referência da API AWS SDK for Rust.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListTopics referência da API AWS SDK for Rust.

`Publicar`

O código de exemplo a seguir mostra como usar `Publish`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Consulte detalhes da API em Publish na Referência da API AWS SDK for Rust.

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Consulte detalhes da API em Subscribe na Referência da API AWS SDK para Rust.

Cenários

Criar uma aplicação com tecnologia sem servidor para gerenciar fotos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Para uma análise detalhada da origem desse exemplo, veja a publicação na Comunidade da AWS.

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Exemplos sem servidor

Invocar uma função do Lambda em um acionador do Amazon SNS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um tópico do SNS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SNS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features =
 ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
```

```
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Exemplos do Amazon SQS usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Exemplos sem servidor

Ações

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Recuperar a primeira fila do Amazon SQS listada na região.

```
async fn find_first_queue(client: &Client) -> Result<String, Error> {
    let queues = client.list_queues().send().await?;
    let queue_urls = queues.queue_urls();
    Ok(queue_urls
        .first()
        .expect("No queues in this account and Region. Create a queue to proceed.")
        .to_string())
}
```

- Para obter detalhes da API, consulte a ListQueues referência da API AWS SDK for Rust.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn receive(client: &Client, queue_url: &String) -> Result<(), Error> {
    let rcv_message_output =
 client.receive_message().queue_url(queue_url).send().await?;

    println!("Messages from queue with url: {}", queue_url);

    for message in rcv_message_output.messages.unwrap_or_default() {
        println!("Got the message: {:#?}", message);
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a ReceiveMessage referência da API AWS SDK for Rust.

SendMessage

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn send(client: &Client, queue_url: &String, message: &SQSMessage) ->
 Result<(), Error> {
    println!("Sending message to queue with URL: {}", queue_url);

    let rsp = client
        .send_message()
        .queue_url(queue_url)
        .message_body(&message.body)
        // If the queue is FIFO, you need to set .message_deduplication_id
        // and message_group_id or configure the queue for
 ContentBasedDeduplication.
        .send()
        .await?;

    println!("Send message to the queue: {:#?}", rsp);

    Ok(())
}
```

- Para obter detalhes da API, consulte a SendMessage referência da API AWS SDK for Rust.

Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de uma fila do SQS. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Consuma um evento do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
 record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();
```

```
    run(service_fn(function_handler)).await
}
```

Relatar falhas de itens em lote para funções do Lambda com um trigger do Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de uma fila do SQS. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos Exemplos sem servidor.

Relatar falhas de itens em lote do SQS com o Lambda usando Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<SqsBatchResponse,
 Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
```

```
        }
    }

    Ok(SqsBatchResponse {
        batch_item_failures,
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(function_handler)).await
}
```

AWS STS exemplos de uso do SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com. AWS STS

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}", e.user_id().unwrap_or_default());
            println!("Account:               {}", e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- Para obter detalhes da API, consulte a AssumeRole referência da API AWS SDK for Rust.

Exemplos do Systems Manager usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Systems Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DescribeParameters`

O código de exemplo a seguir mostra como usar `DescribeParameters`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_parameters(client: &Client) -> Result<(), Error> {
    let resp = client.describe_parameters().send().await?;

    for param in resp.parameters() {
        println!("  {}", param.name().unwrap_or_default());
    }

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeParameters referência da API AWS SDK for Rust.

GetParameter

O código de exemplo a seguir mostra como usar `GetParameter`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error> {
    let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
        self.inner
            .get_parameters_by_path()
            .path(path)
            .into_paginator()
            .send(),
    )
    .flat_map(|item| item.parameters.unwrap_or_default())
    .collect()
    .await;
    // Fail on the first error
    let params = maybe_params
        .into_iter()
        .collect::<Result<Vec<Parameter>, _>>()?;
    Ok(params)
}
```

- Para obter detalhes da API, consulte a GetParameter referência da API AWS SDK for Rust.

PutParameter

O código de exemplo a seguir mostra como usar `PutParameter`.

SDK para Rust

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn make_parameter(
    client: &Client,
    name: &str,
    value: &str,
    description: &str,
) -> Result<(), Error> {
    let resp = client
        .put_parameter()
        .overwrite(true)
        .r#type(ParameterType::String)
        .name(name)
        .value(value)
        .description(description)
        .send()
        .await?;

    println!("Success! Parameter now has version: {}", resp.version());

    Ok(())
}
```

- Para obter detalhes da API, consulte a PutParameter referência da API AWS SDK for Rust.

Exemplos do Amazon Transcribe usando o SDK para Rust

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Rust com o Amazon Transcribe.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Cenários

Cenários

Converter texto em fala e de volta em texto

O exemplo de código a seguir mostra como:

- Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio.
- Fazer upload do arquivo de áudio para um bucket do Amazon S3.
- Usar o Amazon Transcribe para converter o arquivo de áudio em texto.
- Exibir o texto.

SDK para Rust

Use o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) em um arquivo de áudio, fazer upload do arquivo de áudio em um bucket do Amazon S3, usar o Amazon Transcribe para converter esse arquivo de áudio em texto e exibir o texto.

Para obter o código-fonte completo e instruções sobre como configurar e executar, veja o exemplo completo em GitHub.

Serviços usados neste exemplo

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Exemplos de código para o SDK para SAP ABAP.

Os exemplos de código a seguir mostram como usar o AWS SDK para SAP ABAP com. AWS

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para SAP ABAP — Saiba mais sobre como usar o SAP ABAP com. AWS
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.
- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos de ACM usando o SDK para SAP ABAP
- Exemplos de API Gateway usando o SDK para SAP ABAP
- Exemplos do Application Recovery Controller usando o SDK para SAP ABAP
- Exemplos do Aurora usando o SDK para SAP ABAP
- Exemplos de Auto Scaling usando o SDK para SAP ABAP
- Exemplos de Amazon Bedrock Runtime usando o SDK para SAP ABAP
- Exemplos do Amazon Bedrock Agents Runtime usando o SDK para SAP ABAP
- CloudFront exemplos de uso do SDK para SAP ABAP
- CloudWatch exemplos de uso do SDK para SAP ABAP
- CloudWatch Exemplos de registros usando o SDK para SAP ABAP
- Exemplos do Amazon Cognito Identity Provider usando o SDK para SAP ABAP
- Exemplos do Amazon Comprehend usando o SDK para SAP ABAP

- AWS Config exemplos de uso do SDK para SAP ABAP
- AWS Control Tower exemplos de uso do SDK para SAP ABAP
- Exemplos do Firehose usando o SDK para SAP ABAP
- Exemplos do DynamoDB usando o SDK para SAP ABAP
- Exemplos do Amazon EC2 usando o SDK para SAP ABAP
- Exemplos do Amazon ECR usando o SDK para SAP ABAP
- Exemplos do Amazon EMR usando o SDK para SAP ABAP
- EventBridge Exemplos de agendadores usando o SDK para SAP ABAP
- AWS Glue exemplos de uso do SDK para SAP ABAP
- HealthImaging exemplos de uso do SDK para SAP ABAP
- HealthLake exemplos de uso do SDK para SAP ABAP
- Exemplos de IAM usando SDK para SAP ABAP
- AWS IoT exemplos de uso do SDK para SAP ABAP
- AWS IoT data exemplos de uso do SDK para SAP ABAP
- AWS IoT SiteWise exemplos de uso do SDK para SAP ABAP
- Exemplos do Amazon Keyspaces usando o SDK para SAP ABAP
- Exemplos do Kinesis usando o SDK para SAP ABAP
- AWS KMS exemplos de uso do SDK para SAP ABAP
- Exemplos do Lambda usando o SDK para SAP ABAP
- Exemplos de organizações usando o SDK para SAP ABAP
- Exemplos do Amazon Pinpoint usando o SDK para SAP ABAP
- Exemplos da API de SMS e voz do Amazon Pinpoint usando o SDK para SAP ABAP
- Exemplos do Amazon Polly usando o SDK para SAP ABAP
- Exemplos do Amazon RDS usando o SDK para SAP ABAP
- Exemplos do Amazon Redshift usando o SDK para SAP ABAP
- Exemplos do Amazon Rekognition usando o SDK para SAP ABAP
- Exemplos do Amazon S3 usando o SDK para SAP ABAP
- Exemplos de controle do Amazon S3 usando o SDK para SAP ABAP
- SageMaker Exemplos de IA usando SDK para SAP ABAP
- Exemplos do Secrets Manager usando o SDK para SAP ABAP

- Exemplos do Amazon SES usando o SDK para SAP ABAP
- Exemplos da API Amazon SES v2 usando o SDK para SAP ABAP
- Exemplos do Amazon SNS usando o SDK para SAP ABAP
- Exemplos do Amazon SQS usando o SDK para SAP ABAP
- Exemplos do Step Functions usando o SDK para SAP ABAP
- Exemplos do Systems Manager usando o SDK para SAP ABAP
- Exemplos do Amazon Textract usando o SDK para SAP ABAP
- Exemplos do Amazon Transcribe usando o SDK para SAP ABAP
- Exemplos do Amazon Translate usando o SDK para SAP ABAP

Exemplos de ACM usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com ACM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AddTagsToCertificate`

O código de exemplo a seguir mostra como usar `AddTagsToCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->addtagstocertificate(
          iv_certificatearn = iv_certificate_arn
          it_tags = it_tags
        ).
        MESSAGE 'Tags added to certificate successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidtagex.
        MESSAGE 'Invalid tag provided.' TYPE 'I'.
      CATCH /aws1/cx_acmtoomanytagsex.
        MESSAGE 'Too many tags for certificate.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AddTagsToCertificate referência da API AWS SDK for SAP ABAP.

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->deletecertificate( iv_certificatearn = iv_certificate_arn ).
        MESSAGE 'Certificate deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acmresourceinuseex.
        MESSAGE 'Certificate is in use and cannot be deleted.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCertificate referência da API AWS SDK for SAP ABAP.

`DescribeCertificate`

O código de exemplo a seguir mostra como usar `DescribeCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        oo_result = lo_acm->describecertificate( iv_certificatearn =
 iv_certificate_arn ).
        MESSAGE 'Certificate details retrieved.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeCertificate referência da API AWS SDK for SAP ABAP.

`GetCertificate`

O código de exemplo a seguir mostra como usar `GetCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        oo_result = lo_acm->getcertificate( iv_certificatearn =
 iv_certificate_arn ).
        MESSAGE 'Certificate body and chain retrieved.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acmrequestinprgssex.
        MESSAGE 'Certificate request is in progress.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetCertificate referência da API AWS SDK for SAP ABAP.

`ImportCertificate`

O código de exemplo a seguir mostra como usar `ImportCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Only pass certificate chain if it's provided (it's optional)
        IF iv_certificate_chain IS NOT INITIAL.
          DATA(lo_result) = lo_acm->importcertificate(
            iv_certificate = iv_certificate
            iv_privatekey = iv_private_key
            iv_certificatechain = iv_certificate_chain
          ).
        ELSE.
          lo_result = lo_acm->importcertificate(
            iv_certificate = iv_certificate
            iv_privatekey = iv_private_key
          ).
        ENDIF.
        ov_certificate_arn = lo_result->get_certificatearn( ).
        MESSAGE 'Certificate imported successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'I'.
      CATCH /aws1/cx_acmlimitexceededex.
        MESSAGE 'Certificate limit exceeded.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ImportCertificate referência da API AWS SDK for SAP ABAP.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_acm->listcertificates(
          iv_maxitems = iv_max_items
          it_certificatestatuses = it_statuses
          io_includes = io_includes
        ).
        MESSAGE 'Certificates listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidargsex.
        MESSAGE 'Invalid arguments provided.' TYPE 'I'.
      CATCH /aws1/cx_acmvalidationex.
        MESSAGE 'Validation error occurred.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListCertificates referência da API AWS SDK for SAP ABAP.

`ListTagsForCertificate`

O código de exemplo a seguir mostra como usar `ListTagsForCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
```

```
        DATA(lo_result) = lo_acm->listtagsforcertificate(
          iv_certificatearn = iv_certificate_arn
        ).
        ot_tags = lo_result->get_tags( ).
        MESSAGE 'Certificate tags retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTagsForCertificate referência da API AWS SDK for SAP ABAP.

`RemoveTagsFromCertificate`

O código de exemplo a seguir mostra como usar `RemoveTagsFromCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->removetagsfromcertificate(
          iv_certificatearn = iv_certificate_arn
          it_tags = it_tags
        ).
        MESSAGE 'Tags removed from certificate successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidtagex.
        MESSAGE 'Invalid tag provided.' TYPE 'I'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a RemoveTagsFromCertificate referência da API AWS SDK for SAP ABAP.

`RequestCertificate`

O código de exemplo a seguir mostra como usar `RequestCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_domain_name = 'example.com'
        " iv_validation_method = 'DNS' or 'EMAIL'
        DATA(lo_result) = lo_acm->requestcertificate(
          iv_domainname = iv_domain_name
          it_subjectalternativenames = COND #( WHEN it_alternate_domains IS NOT
INITIAL
                                                THEN it_alternate_domains )
          iv_validationmethod = iv_validation_method
        ).
        ov_certificate_arn = lo_result->get_certificatearn( ).
        MESSAGE 'Certificate requested successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'I'.
      CATCH /aws1/cx_acmlimitexceededex.
        MESSAGE 'Certificate limit exceeded.' TYPE 'I'.
      CATCH /aws1/cx_acminvdomvationoptsex.
        MESSAGE 'Invalid domain validation options.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RequestCertificate referência da API AWS SDK for SAP ABAP.

`ResendValidationEmail`

O código de exemplo a seguir mostra como usar `ResendValidationEmail`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        " iv_domain = 'example.com'
        " iv_validation_domain = 'example.com'
        lo_acm->resendvalidationemail(
          iv_certificatearn = iv_certificate_arn
          iv_domain = iv_domain
          iv_validationdomain = iv_validation_domain
        ).
        MESSAGE 'Validation email resent successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidstateex.
        MESSAGE 'Certificate is not in a valid state.' TYPE 'I'.
      CATCH /aws1/cx_acminvdomvationoptsex.
        MESSAGE 'Invalid domain validation options.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ResendValidationEmail referência da API AWS SDK for SAP ABAP.

Exemplos de API Gateway usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o API Gateway.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateDeployment`

O código de exemplo a seguir mostra como usar `CreateDeployment`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_agw->createdeployment(
          iv_restapiid = iv_rest_api_id
          iv_stagename = iv_stage_name
          iv_description = 'Deployment created by ABAP SDK' ).
        DATA(lv_deployment_id) = oo_result->get_id( ).
        MESSAGE 'Deployment created with ID: ' && lv_deployment_id TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDeployment referência da API AWS SDK for SAP ABAP.

`CreateResource`

O código de exemplo a seguir mostra como usar `CreateResource`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_agw->createresource(
          iv_restapiid = iv_rest_api_id
          iv_parentid = iv_parent_id
          iv_pathpart = iv_resource_path ).
        DATA(lv_resource_id) = oo_result->get_id( ).
        MESSAGE 'Resource created with ID: ' && lv_resource_id TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateResource referência da API AWS SDK for SAP ABAP.

`CreateRestApi`

O código de exemplo a seguir mostra como usar `CreateRestApi`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_agw->createrestapi(
      iv_name = iv_api_name
      iv_description = 'Sample REST API created by ABAP SDK' ).
    DATA(lv_api_id) = oo_result->get_id( ).
    MESSAGE 'REST API created with ID: ' && lv_api_id TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
  CATCH /aws1/cx_agwunauthorizedex INTO DATA(lo_unauthorized).
    MESSAGE lo_unauthorized->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_unauthorized.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateRestApi referência da API AWS SDK for SAP ABAP.

`DeleteRestApi`

O código de exemplo a seguir mostra como usar `DeleteRestApi`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_agw->deleterestapi(
      iv_restapiid = iv_rest_api_id ).
    MESSAGE 'REST API deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteRestApi referência da API AWS SDK for SAP ABAP.

`GetResources`

O código de exemplo a seguir mostra como usar `GetResources`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
```

```
      oo_result = lo_agw->getresources(
        iv_restapiid = iv_rest_api_id ).
      DATA(lt_resources) = oo_result->get_items( ).
      DATA(lv_count) = lines( lt_resources ).
      MESSAGE 'Found ' && lv_count && ' resources' TYPE 'I'.
    CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
      MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
    CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
      MESSAGE lo_not_found->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_not_found.
    CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
      MESSAGE lo_too_many->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetResources referência da API AWS SDK for SAP ABAP.

`GetRestApis`

O código de exemplo a seguir mostra como usar `GetRestApis`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_agw->getrestapis( ).
      DATA(lt_apis) = oo_result->get_items( ).
      DATA(lv_count) = lines( lt_apis ).
      MESSAGE 'Found ' && lv_count && ' REST APIs' TYPE 'I'.
    CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
      MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
    CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
```

```
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetRestApis referência da API AWS SDK for SAP ABAP.

`PutIntegration`

O código de exemplo a seguir mostra como usar `PutIntegration`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_agw->putintegration(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_type = 'AWS_PROXY'
          iv_integrationhttpmethod = 'POST'
          iv_uri = iv_integration_uri ).
        MESSAGE 'Integration configured for method' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutIntegration referência da API AWS SDK for SAP ABAP.

`PutIntegrationResponse`

O código de exemplo a seguir mostra como usar `PutIntegrationResponse`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_agw->putintegrationresponse(
      iv_restapiid = iv_rest_api_id
      iv_resourceid = iv_resource_id
      iv_httpmethod = iv_http_method
      iv_statuscode = '200' ).
    MESSAGE 'Integration response configured for status 200' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutIntegrationResponse referência da API AWS SDK for SAP ABAP.

`PutMethod`

O código de exemplo a seguir mostra como usar `PutMethod`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_agw->putmethod(
      iv_restapiid = iv_rest_api_id
      iv_resourceid = iv_resource_id
      iv_httpmethod = iv_http_method
      iv_authorizationtype = 'NONE' ).
    MESSAGE 'Method ' && iv_http_method && ' added to resource' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutMethod referência da API AWS SDK for SAP ABAP.

`PutMethodResponse`

O código de exemplo a seguir mostra como usar `PutMethodResponse`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_agw->putmethodresponse(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_statuscode = '200' ).
        MESSAGE 'Method response configured for status 200' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutMethodResponse referência da API AWS SDK for SAP ABAP.

Exemplos do Application Recovery Controller usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Application Recovery Controller.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetRoutingControlState`

O código de exemplo a seguir mostra como usar `GetRoutingControlState`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA lo_exception TYPE REF TO /aws1/cx_rt_generic.
    DATA lo_session TYPE REF TO /aws1/cl_rt_session_base.
    DATA lo_client TYPE REF TO /aws1/if_r5v.
    DATA lt_endpoints TYPE TABLE OF string.
    DATA lv_endpoint TYPE string.
    DATA lv_region TYPE /aws1/rt_region_id.

    " Parse the comma-separated cluster endpoints
    " Expected format: "https://endpoint1.com|us-west-2,https://endpoint2.com|us-
east-1"
    SPLIT iv_cluster_endpoints AT ',' INTO TABLE lt_endpoints.

    " As a best practice, shuffle cluster endpoints to distribute load
    " For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
    " For simplicity, we'll try them in order (shuffling can be added if needed)

    " Try each endpoint in order
    LOOP AT lt_endpoints INTO lv_endpoint.
      TRY.
          " Parse endpoint and region from the format "url|region"
          DATA(lv_pos) = find( val = lv_endpoint sub = '|' ).
          IF lv_pos > 0.
            DATA(lv_url) = substring( val = lv_endpoint len = lv_pos ).
            lv_region = substring( val = lv_endpoint off = lv_pos + 1 ).
          ELSE.
            " If no region specified, use default
            lv_url = lv_endpoint.
```

```
            lv_region = 'us-east-1'.
          ENDIF.

          " Create session for this region
          lo_session = /aws1/cl_rt_session_aws=>create( cv_pfl ).

          " Create client with the specific endpoint
          lo_client = create_recovery_client(
            iv_endpoint = lv_url
            iv_region   = lv_region
            io_session  = lo_session ).

          " Try to get the routing control state
          oo_result = lo_client->getroutingcontrolstate(
            iv_routingcontrolarn = iv_routing_control_arn ).

          " If successful, return the result
          RETURN.

        CATCH /aws1/cx_r5vendpttmpyunavailex INTO DATA(lo_endpoint_ex).
          " This endpoint is temporarily unavailable, try the next one
          lo_exception = lo_endpoint_ex.
          CONTINUE.

        CATCH /aws1/cx_r5vaccessdeniedex
              /aws1/cx_r5vinternalserverex
              /aws1/cx_r5vresourcenotfoundex
              /aws1/cx_r5vthrottlingex
              /aws1/cx_r5vvalidationex
              /aws1/cx_rt_generic INTO lo_exception.
          " For other errors, re-raise immediately
          RAISE EXCEPTION lo_exception.
      ENDTRY.
    ENDLOOP.

    " If we get here, all endpoints failed - re-raise the last exception
    IF lo_exception IS BOUND.
      RAISE EXCEPTION lo_exception.
    ENDIF.
```

- Para obter detalhes da API, consulte a GetRoutingControlState referência da API AWS SDK for SAP ABAP.

`UpdateRoutingControlState`

O código de exemplo a seguir mostra como usar `UpdateRoutingControlState`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA lo_exception TYPE REF TO /aws1/cx_rt_generic.
    DATA lo_session TYPE REF TO /aws1/cl_rt_session_base.
    DATA lo_client TYPE REF TO /aws1/if_r5v.
    DATA lt_endpoints TYPE TABLE OF string.
    DATA lv_endpoint TYPE string.
    DATA lv_region TYPE /aws1/rt_region_id.

    " Parse the comma-separated cluster endpoints
    " Expected format: "https://endpoint1.com|us-west-2,https://endpoint2.com|us-
east-1"
    SPLIT iv_cluster_endpoints AT ',' INTO TABLE lt_endpoints.

    " As a best practice, shuffle cluster endpoints to distribute load
    " For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
    " For simplicity, we'll try them in order (shuffling can be added if needed)

    " Try each endpoint in order
    LOOP AT lt_endpoints INTO lv_endpoint.
      TRY.
          " Parse endpoint and region from the format "url|region"
          DATA(lv_pos) = find( val = lv_endpoint sub = '|' ).
          IF lv_pos > 0.
            DATA(lv_url) = substring( val = lv_endpoint len = lv_pos ).
            lv_region = substring( val = lv_endpoint off = lv_pos + 1 ).
          ELSE.
            " If no region specified, use default
            lv_url = lv_endpoint.
            lv_region = 'us-east-1'.
          ENDIF.
```

```
        " Create session for this region
        lo_session = /aws1/cl_rt_session_aws=>create( cv_pfl ).

        " Create client with the specific endpoint
        lo_client = create_recovery_client(
          iv_endpoint = lv_url
          iv_region   = lv_region
          io_session  = lo_session ).

        " Try to update the routing control state
        oo_result = lo_client->updateroutingcontrolstate(
          iv_routingcontrolarn     = iv_routing_control_arn
          iv_routingcontrolstate   = iv_routing_control_state
          it_safetyrulestooverride = it_safety_rules_override ).

        " If successful, return the result
        RETURN.

      CATCH /aws1/cx_r5vendpttmpyunavailex INTO DATA(lo_endpoint_ex).
        " This endpoint is temporarily unavailable, try the next one
        lo_exception = lo_endpoint_ex.
        CONTINUE.

      CATCH /aws1/cx_r5vaccessdeniedex
            /aws1/cx_r5vconflictexception
            /aws1/cx_r5vinternalserverex
            /aws1/cx_r5vresourcenotfoundex
            /aws1/cx_r5vthrottlingex
            /aws1/cx_r5vvalidationex
            /aws1/cx_rt_generic INTO lo_exception.
        " For other errors, re-raise immediately
        RAISE EXCEPTION lo_exception.
    ENDTRY.
  ENDLOOP.

  " If we get here, all endpoints failed - re-raise the last exception
  IF lo_exception IS BOUND.
    RAISE EXCEPTION lo_exception.
  ENDIF.
```

- Para obter detalhes da API, consulte a UpdateRoutingControlState referência da API AWS SDK for SAP ABAP.

Exemplos do Aurora usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Aurora.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBClusterParameterGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_output) = lo_rds->createdbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
      iv_dbparametergroupfamily = iv_param_group_family
      iv_description = iv_description
    ).
```

```
      oo_result = lo_output->get_dbclusterparametergroup( ).
    CATCH /aws1/cx_rdsdbparmgralrexfault.
      " Re-raise exception - parameter group already exists
      RAISE EXCEPTION TYPE /aws1/cx_rdsdbparmgralrexfault.
    CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
      " Re-raise exception - quota exceeded
      RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrquotaexcd00.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDBClusterParameterGroup referência da API AWS SDK for SAP ABAP.

`DeleteDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBClusterParameterGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_rds->deletedbclusterparamgroup(
        iv_dbclusterparamgroupname = iv_param_group_name
      ).
    CATCH /aws1/cx_rdsdbprmgrnotfndfault.
      " Re-raise exception - parameter group not found
      RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
    CATCH /aws1/cx_rdsinvdbprmgrstatef00.
      " Re-raise exception - invalid state
      RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDBClusterParameterGroup referência da API AWS SDK for SAP ABAP.

`DescribeDBClusterParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameterGroups`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_output) = lo_rds->describedbclusterparamgroups(
      iv_dbclusterparamgroupname = iv_param_group_name
    ).
    DATA(lt_param_groups) = lo_output->get_dbclusterparametergroups( ).
    IF lines( lt_param_groups ) > 0.
      oo_result = lt_param_groups[ 1 ].
    ENDIF.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameterGroups referência da API AWS SDK for SAP ABAP.

`DescribeDBClusterParameters`

O código de exemplo a seguir mostra como usar `DescribeDBClusterParameters`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
```

```
      DATA lv_marker TYPE /aws1/rdsstring VALUE ''.
      DATA lt_all_parameters TYPE /aws1/cl_rdsparameter=>tt_parameterslist.

      DO.
        DATA(lo_output) = lo_rds->describedbclusterparameters(
          iv_dbclusterparamgroupname = iv_param_group_name
          iv_source = iv_source
          iv_marker = lv_marker
        ).

        LOOP AT lo_output->get_parameters( ) INTO DATA(lo_param).
          IF iv_name_prefix IS INITIAL OR
             lo_param->get_parametername( ) CP |{ iv_name_prefix }*|.
            APPEND lo_param TO lt_all_parameters.
          ENDIF.
        ENDLOOP.

        lv_marker = lo_output->get_marker( ).
        IF lv_marker IS INITIAL.
          EXIT.
        ENDIF.
      ENDDO.

      ot_parameters = lt_all_parameters.
    CATCH /aws1/cx_rdsdbprmgrnotfndfault.
      " Re-raise exception - parameter group not found
      RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBClusterParameters referência da API AWS SDK for SAP ABAP.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " iv_engine                 = 'mysql'
   " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter group
family)
   TRY.
       oo_result = lo_rds->describedbengineversions(
         iv_engine                 = iv_engine
         iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
       DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
       MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersions referência da API AWS SDK for SAP ABAP.

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " iv_engine        = 'mysql'
   " iv_engineversion = '8.0.35'
   TRY.
       oo_result = lo_rds->descrorderabledbinstoptions(
```

```
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result->get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.| TYPE
'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptions referência da API AWS SDK for SAP ABAP.

`ModifyDBClusterParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBClusterParameterGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_rds->modifydbclusterparamgroup(
          iv_dbclusterparamgroupname = iv_param_group_name
          it_parameters = it_update_parameters
        ).
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        " Re-raise exception - parameter group not found
        RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        " Re-raise exception - invalid state
        RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ModifyDBClusterParameterGroup referência da API AWS SDK for SAP ABAP.

Exemplos de Auto Scaling usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com Auto Scaling.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `CreateAutoScalingGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA lo_launch_template TYPE REF TO /aws1/cl_asclaunchtemplatespec.

" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_launch_template_name = 'my-launch-template'
" Example: iv_min_size = 1
" Example: iv_max_size = 3
" Example: iv_vpc_zone_identifier = 'subnet-12345,subnet-67890' (for VPC)

TRY.
    " Create launch template specification
```

```
        lo_launch_template = NEW /aws1/cl_asclaunchtemplatespec(
          iv_launchtemplatename = iv_launch_template_name
          iv_version = '$Default' ).

        " Create the Auto Scaling group
        " Use VPCZoneIdentifier for VPC subnets, or AvailabilityZones for EC2-
Classic
        IF iv_vpc_zone_identifier IS NOT INITIAL.
          " VPC-based deployment - use subnet IDs
          ao_asc->createautoscalinggroup(
            iv_autoscalinggroupname = iv_group_name
            iv_vpczoneidentifier = iv_vpc_zone_identifier
            io_launchtemplate = lo_launch_template
            iv_minsize = iv_min_size
            iv_maxsize = iv_max_size ).
        ELSE.
          " EC2-Classic or default VPC - use availability zones
          ao_asc->createautoscalinggroup(
            iv_autoscalinggroupname = iv_group_name
            it_availabilityzones = it_group_zones
            io_launchtemplate = lo_launch_template
            iv_minsize = iv_min_size
            iv_maxsize = iv_max_size ).
        ENDIF.

        " Wait for the group to be created (simplified - in production use proper
 polling)
        WAIT UP TO 10 SECONDS.

        MESSAGE 'Auto Scaling group created successfully' TYPE 'I'.

      CATCH /aws1/cx_ascalreadyexistsfault INTO DATA(lo_already_exists).
        RAISE EXCEPTION lo_already_exists.
      CATCH /aws1/cx_asclimitexceededfault INTO DATA(lo_limit_exceeded).
        RAISE EXCEPTION lo_limit_exceeded.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAutoScalingGroup referência da API AWS SDK for SAP ABAP.

`DeleteAutoScalingGroup`

O código de exemplo a seguir mostra como usar `DeleteAutoScalingGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " Example: iv_group_name = 'my-auto-scaling-group'

   TRY.
       ao_asc->deleteautoscalinggroup(
         iv_autoscalinggroupname = iv_group_name ).

       " Wait for the group to be deleted (simplified - in production use proper
polling)
       WAIT UP TO 10 SECONDS.

       MESSAGE 'Auto Scaling group deleted successfully' TYPE 'I'.

     CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
       RAISE EXCEPTION lo_activity_in_progress.
     CATCH /aws1/cx_ascresourceinusefault INTO DATA(lo_resource_in_use).
       RAISE EXCEPTION lo_resource_in_use.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
       RAISE EXCEPTION lo_generic_exception.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAutoScalingGroup referência da API AWS SDK for SAP ABAP.

`DescribeAutoScalingGroups`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingGroups`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_group_names TYPE /aws1/
cl_ascautoscgroupnames_w=>tt_autoscalinggroupnames.
    DATA lo_group_name TYPE REF TO /aws1/cl_ascautoscgroupnames_w.

    " Example: iv_group_name = 'my-auto-scaling-group'

    TRY.
        " Build group names parameter
        CREATE OBJECT lo_group_name
          EXPORTING
            iv_value = iv_group_name.
        APPEND lo_group_name TO lt_group_names.

        " Describe the Auto Scaling group
        DATA(lo_output) = ao_asc->describeautoscalinggroups(
          it_autoscalinggroupnames = lt_group_names ).

        " Return the first (and only) group in the result
        DATA(lt_groups) = lo_output->get_autoscalinggroups( ).
        IF lines( lt_groups ) > 0.
          READ TABLE lt_groups INDEX 1 INTO DATA(lo_group).
          oo_output = lo_group.
        ENDIF.

        MESSAGE 'Auto Scaling group information retrieved successfully' TYPE 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAutoScalingGroups referência da API AWS SDK for SAP ABAP.

`DescribeAutoScalingInstances`

O código de exemplo a seguir mostra como usar `DescribeAutoScalingInstances`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " Example: it_instance_ids contains a list of instance IDs

   TRY.
       DATA(lo_output) = ao_asc->describeautoscalinginstances(
         it_instanceids = it_instance_ids ).

       ot_output = lo_output->get_autoscalinginstances( ).

       MESSAGE 'Auto Scaling instances information retrieved successfully' TYPE
'I'.

     CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
       RAISE EXCEPTION lo_contention.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
       RAISE EXCEPTION lo_generic_exception.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAutoScalingInstances referência da API AWS SDK for SAP ABAP.

`DescribeScalingActivities`

O código de exemplo a seguir mostra como usar `DescribeScalingActivities`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
    DATA(lo_output) = ao_asc->describescalingactivities(
      iv_autoscalinggroupname = iv_group_name ).

    ot_output = lo_output->get_activities( ).

    MESSAGE 'Scaling activities retrieved successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeScalingActivities referência da API AWS SDK for SAP ABAP.

`DisableMetricsCollection`

O código de exemplo a seguir mostra como usar `DisableMetricsCollection`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Example: iv_group_name = 'my-auto-scaling-group'

    TRY.
        ao_asc->disablemetricscollection(
          iv_autoscalinggroupname = iv_group_name ).

        MESSAGE 'Metrics collection disabled successfully' TYPE 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DisableMetricsCollection referência da API AWS SDK for SAP ABAP.

`EnableMetricsCollection`

O código de exemplo a seguir mostra como usar `EnableMetricsCollection`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Example: iv_group_name = 'my-auto-scaling-group'
    " Example: it_metrics contains list of metrics like 'GroupMinSize',
 'GroupMaxSize', etc.

    TRY.
        ao_asc->enablemetricscollection(
          iv_autoscalinggroupname = iv_group_name
          it_metrics = it_metrics
          iv_granularity = '1Minute' ).

        MESSAGE 'Metrics collection enabled successfully' TYPE 'I'.
```

```
    CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
      RAISE EXCEPTION lo_contention.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
      RAISE EXCEPTION lo_generic_exception.
  ENDTRY.
```

- Para obter detalhes da API, consulte a EnableMetricsCollection referência da API AWS SDK for SAP ABAP.

`SetDesiredCapacity`

O código de exemplo a seguir mostra como usar `SetDesiredCapacity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  " Example: iv_group_name = 'my-auto-scaling-group'
  " Example: iv_capacity = 2

  TRY.
      ao_asc->setdesiredcapacity(
        iv_autoscalinggroupname = iv_group_name
        iv_desiredcapacity = iv_capacity
        iv_honorcooldown = abap_false ).

      MESSAGE 'Desired capacity set successfully' TYPE 'I'.

    CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
      RAISE EXCEPTION lo_activity_in_progress.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
      RAISE EXCEPTION lo_generic_exception.
  ENDTRY.
```

- Para obter detalhes da API, consulte a SetDesiredCapacity referência da API AWS SDK for SAP ABAP.

`TerminateInstanceInAutoScalingGroup`

O código de exemplo a seguir mostra como usar `TerminateInstanceInAutoScalingGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Example: iv_instance_id = 'i-1234567890abcdef0'
" Example: iv_decrease_capacity = abap_true

TRY.
    DATA(lo_output) = ao_asc->terminateinstinautoscgroup(
      iv_instanceid = iv_instance_id
      iv_shoulddecrementdesiredcap = iv_decrease_capacity ).

    oo_output = lo_output->get_activity( ).

    MESSAGE 'Instance terminated successfully' TYPE 'I'.

  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Para obter detalhes da API, consulte a TerminateInstanceInAutoScalingGroup referência da API AWS SDK for SAP ABAP.

`UpdateAutoScalingGroup`

O código de exemplo a seguir mostra como usar `UpdateAutoScalingGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_max_size = 5

TRY.
    ao_asc->updateautoscalinggroup(
      iv_autoscalinggroupname = iv_group_name
      iv_maxsize = iv_max_size
      iv_minsize = iv_min_size ).

    MESSAGE 'Auto Scaling group updated successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateAutoScalingGroup referência da API AWS SDK for SAP ABAP.

Exemplos de Amazon Bedrock Runtime usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Bedrock Runtime.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Stable Image Core

Stable Image Core

InvokeModel

O exemplo de código a seguir mostra como invocar o Stability.ai Stable Image Core no Amazon Bedrock para gerar uma imagem.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Stable Diffusion.

```
    "Stable Image Core Input Parameters should be in a format like this:
*   {
*     "prompt": "Draw a dolphin with a mustache, photorealistic",
*     "aspect_ratio": "1:1",
*     "seed": 0,
*     "output_format": "png"
*   }
    DATA: BEGIN OF ls_input,
            prompt        TYPE /aws1/rt_shape_string,
            aspect_ratio  TYPE /aws1/rt_shape_string,
            seed          TYPE /aws1/rt_shape_integer,
            output_format TYPE /aws1/rt_shape_string,
          END OF ls_input.

    ls_input-prompt = iv_prompt.
    ls_input-aspect_ratio = '1:1'.
    ls_input-seed = 0. "or better, choose a random integer.
    ls_input-output_format = 'png'.
```

```
    DATA(lv_json) = /ui2/cl_json=>serialize(
      data = ls_input
                pretty_name   = /ui2/cl_json=>pretty_mode-low_case ).

    TRY.
        DATA(lo_response) = lo_bdr->invokemodel(
          iv_body = /aws1/cl_rt_util=>string_to_xstring( lv_json )
          iv_modelid = 'stability.stable-image-core-v1:1'
          iv_accept = 'application/json'
          iv_contenttype = 'application/json' ).

        "Stable Image Core Result Format:
*       {
*         "seeds": ["0"],
*         "finish_reasons": [null],
*         "images": ["iVBORw0KGgoAAAANSUhEUgAAAgAAA...."]
*       }
        DATA: BEGIN OF ls_response,
                images TYPE STANDARD TABLE OF /aws1/rt_shape_string,
              END OF ls_response.

        /ui2/cl_json=>deserialize(
          EXPORTING jsonx = lo_response->get_body( )
                    pretty_name = /ui2/cl_json=>pretty_mode-camel_case
          CHANGING  data  = ls_response ).
        IF ls_response-images IS NOT INITIAL.
          DATA(lv_image) =
 cl_http_utility=>if_http_utility~decode_x_base64( ls_response-images[ 1 ] ).
        ENDIF.
      CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

Invoque o modelo básico Stability.ai Stable Image Core para gerar imagens usando o cliente de alto nível L2.

```
    TRY.
```

```
        DATA(lo_bdr_l2_sd) = /aws1/
cl_bdr_l2_factory=>create_stable_diffusion_xl_1( lo_bdr ).
        " iv_prompt contains a prompt like 'Show me a picture of a unicorn reading
 an enterprise financial report'.
        DATA(lv_image) = lo_bdr_l2_sd->text_to_image( iv_prompt ).
      CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

- Para obter detalhes da API, consulte a InvokeModel referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Bedrock Agents Runtime usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Bedrock Agents Runtime.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`InvokeAgent`

O código de exemplo a seguir mostra como usar `InvokeAgent`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA(lo_result) = lo_bdz->invokeagent(
  iv_agentid      = iv_agentid
    iv_agentaliasid = iv_agentaliasid
    iv_enabletrace  = abap_true
    iv_sessionid    = CONV #( cl_system_uuid=>create_uuid_c26_static( ) )
    iv_inputtext    = |Let's play "rock, paper, scissors".  I choose rock.| ).
DATA(lo_stream) = lo_result->get_completion( ).
TRY.
    " loop while there are still events in the stream
    WHILE lo_stream->/aws1/if_rt_stream_reader~data_available( ) = abap_true.
      DATA(lo_evt) = lo_stream->read( ).
      " each /AWS1/CL_BDZRESPONSESTREAM_EV event contains exactly one member
      " all others are INITIAL.  For each event, process the non-initial
      " member if desired
      IF lo_evt->get_chunk( ) IS NOT INITIAL.
        " Process a Chunk event
        DATA(lv_xstr) = lo_evt->get_chunk( )->get_bytes( ).
        DATA(lv_answer) = /aws1/cl_rt_util=>xstring_to_string( lv_xstr ).
        " the answer says something like "I chose paper, so you lost"
      ELSEIF lo_evt->get_files( ) IS NOT INITIAL.
        " process a Files event if desired
      ELSEIF lo_evt->get_returncontrol( ) IS NOT INITIAL.
        " process a ReturnControl event if desired
      ELSEIF lo_evt->get_trace( ) IS NOT INITIAL.
        " process a Trace event if desired
      ENDIF.
    ENDWHILE.
    " the stream of events can possibly contain an exception
    " which will be raised to break the loop
    " catch /AWS1/CX_BDZACCESSDENIEDEX.
    " catch /AWS1/CX_BDZINTERNALSERVEREX.
    " catch /AWS1/CX_BDZMODELNOTREADYEX.
    " catch /AWS1/CX_BDZVALIDATIONEX.
```

```
        " catch /AWS1/CX_BDZTHROTTLINGEX.
        " catch /AWS1/CX_BDZDEPENDENCYFAILEDEX.
        " catch /AWS1/CX_BDZBADGATEWAYEX.
        " catch /AWS1/CX_BDZRESOURCENOTFOUNDEX.
        " catch /AWS1/CX_BDZSERVICEQUOTAEXCDEX.
        " catch /AWS1/CX_BDZCONFLICTEXCEPTION.
    ENDTRY.
```

- Para obter detalhes da API, consulte a InvokeAgent referência da API AWS SDK for SAP ABAP.

CloudFront exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. CloudFront

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`ListDistributions`

O código de exemplo a seguir mostra como usar `ListDistributions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_fnt->listdistributions( ). " oo_result is returned for
 testing purposes. "
        MESSAGE 'Retrieved list of CloudFront distributions.' TYPE 'I'.
      CATCH /aws1/cx_fntinvalidargument.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListDistributions referência da API AWS SDK for SAP ABAP.

`UpdateDistribution`

O código de exemplo a seguir mostra como usar `UpdateDistribution`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Get the current distribution configuration and ETag "
        DATA(lo_distribution_config_result) = lo_fnt->getdistributionconfig( iv_id =
 iv_distribution_id ).
        DATA(lo_old_config) = lo_distribution_config_result-
>get_distributionconfig( ).
        DATA(lv_etag) = lo_distribution_config_result->get_etag( ).

        " Create a new distribution config with the updated comment "
        " Since the config object is immutable, we need to create a new one with all
 existing values "
        DATA(lo_new_config) = NEW /aws1/cl_fntdistributionconfig(
          iv_callerreference = lo_old_config->get_callerreference( )
          io_aliases = lo_old_config->get_aliases( )
          iv_defaultrootobject = lo_old_config->get_defaultrootobject( )
          io_origins = lo_old_config->get_origins( )
          io_origingroups = lo_old_config->get_origingroups( )
```

```
          io_defaultcachebehavior = lo_old_config->get_defaultcachebehavior( )
          io_cachebehaviors = lo_old_config->get_cachebehaviors( )
          io_customerrorresponses = lo_old_config->get_customerrorresponses( )
          iv_comment = iv_comment
          io_logging = lo_old_config->get_logging( )
          iv_priceclass = lo_old_config->get_priceclass( )
          iv_enabled = lo_old_config->get_enabled( )
          io_viewercertificate = lo_old_config->get_viewercertificate( )
          io_restrictions = lo_old_config->get_restrictions( )
          iv_webaclid = lo_old_config->get_webaclid( )
          iv_httpversion = lo_old_config->get_httpversion( )
          iv_isipv6enabled = lo_old_config->get_isipv6enabled( ) ).

        " Update the distribution with the modified configuration "
        lo_fnt->updatedistribution(
          io_distributionconfig = lo_new_config
          iv_id = iv_distribution_id
          iv_ifmatch = lv_etag ).
        MESSAGE 'CloudFront distribution updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_fntnosuchdistribution.
        MESSAGE 'Distribution does not exist.' TYPE 'E'.
      CATCH /aws1/cx_fntpreconditionfailed.
        MESSAGE 'Precondition failed - ETag mismatch.' TYPE 'E'.
      CATCH /aws1/cx_fntinvalidifmatchvrs.
        MESSAGE 'Invalid If-Match version.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateDistribution referência da API AWS SDK for SAP ABAP.

CloudWatch exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. CloudWatch

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DeleteAlarms`

O código de exemplo a seguir mostra como usar `DeleteAlarms`.

SDK para SAP ABAP

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_cwt->deletealarms(
      it_alarmnames = it_alarm_names ).
    MESSAGE 'Alarms deleted.' TYPE 'I'.
  CATCH /aws1/cx_cwtresourcenotfound.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAlarms referência da API AWS SDK for SAP ABAP.

`DescribeAlarms`

O código de exemplo a seguir mostra como usar `DescribeAlarms`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cwt->describealarms(                " oo_result is returned
 for testing purposes. "
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarms retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAlarms referência da API AWS SDK for SAP ABAP.

`DescribeAlarmsForMetric`

O código de exemplo a seguir mostra como usar `DescribeAlarmsForMetric`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cwt->describealarmsformetric(          " oo_result is
returned for testing purposes. "
          iv_namespace   = iv_namespace
```

```
          iv_metricname  = iv_metric_name
          it_dimensions  = it_dimensions
          iv_statistic   = iv_statistic
          iv_period      = iv_period
          iv_unit        = iv_unit ).
        MESSAGE 'Alarms for metric retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAlarmsForMetric referência da API AWS SDK for SAP ABAP.

`DisableAlarmActions`

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Disables actions on the specified alarm. "
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarm actions disabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DisableAlarmActions referência da API AWS SDK for SAP ABAP.

EnableAlarmActions

O código de exemplo a seguir mostra como usar `EnableAlarmActions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Enable actions on the specified alarm."
    TRY.
        lo_cwt->enablealarmactions(
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarm actions enabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a EnableAlarmActions referência da API AWS SDK for SAP ABAP.

GetMetricStatistics

O código de exemplo a seguir mostra como usar `GetMetricStatistics`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_cwt->getmetricstatistics(              " oo_result is
returned for testing purposes. "
         iv_namespace   = iv_namespace
         iv_metricname  = iv_metric_name
         iv_starttime   = iv_start_time
         iv_endtime     = iv_end_time
         iv_period      = iv_period
         it_statistics  = it_statistics ).
       MESSAGE 'Metric statistics retrieved.' TYPE 'I'.
     CATCH /aws1/cx_cwtinvparamvalueex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetMetricStatistics referência da API AWS SDK for SAP ABAP.

`ListMetrics`

O código de exemplo a seguir mostra como usar `ListMetrics`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "The following list-metrics example displays the metrics for Amazon CloudWatch."
```

```
    TRY.
        oo_result = lo_cwt->listmetrics(            " oo_result is returned for
testing purposes. "
          iv_namespace = iv_namespace ).
        DATA(lt_metrics) = oo_result->get_metrics( ).
        MESSAGE 'Metrics retrieved.' TYPE 'I'.
      CATCH /aws1/cx_cwtinvparamvalueex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListMetrics referência da API AWS SDK for SAP ABAP.

`PutMetricAlarm`

O código de exemplo a seguir mostra como usar `PutMetricAlarm`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_cwt->putmetricalarm(
          iv_alarmname                 = iv_alarm_name
          iv_comparisonoperator        = iv_comparison_operator
          iv_evaluationperiods         = iv_evaluation_periods
          iv_metricname                = iv_metric_name
          iv_namespace                 = iv_namespace
          iv_statistic                 = iv_statistic
          iv_threshold                 = iv_threshold
          iv_actionsenabled            = iv_actions_enabled
          iv_alarmdescription          = iv_alarm_description
          iv_unit                      = iv_unit
          iv_period                    = iv_period
          it_dimensions                = it_dimensions ).
        MESSAGE 'Alarm created.' TYPE 'I'.
```

```
    CATCH /aws1/cx_cwtlimitexceededfault.
      MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a PutMetricAlarm referência da API AWS SDK for SAP ABAP.

`PutMetricData`

O código de exemplo a seguir mostra como usar `PutMetricData`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  DATA lt_metricdata TYPE /aws1/cl_cwtmetricdatum=>tt_metricdata.

  "Create metric data object.
  DATA(lo_metricdatum) = NEW /aws1/cl_cwtmetricdatum(
    iv_metricname = iv_metric_name
    iv_value      = iv_value
    iv_unit       = iv_unit ).

  INSERT lo_metricdatum INTO TABLE lt_metricdata.

  TRY.
      lo_cwt->putmetricdata(
        iv_namespace   = iv_namespace
        it_metricdata  = lt_metricdata ).
      MESSAGE 'Metric data added.' TYPE 'I'.
    CATCH /aws1/cx_cwtinvparamvalueex.
      MESSAGE 'The specified argument was not valid.' TYPE 'E'.
  ENDTRY.
```

Coloque um conjunto de dados em uma CloudWatch métrica.

```
DATA lt_metricdata TYPE /aws1/cl_cwtmetricdatum=>tt_metricdata.

"Create metric data object with values and counts.
DATA(lo_metricdatum) = NEW /aws1/cl_cwtmetricdatum(
  iv_metricname = iv_metric_name
  iv_timestamp  = iv_timestamp
  iv_unit       = iv_unit
  it_values     = it_values
  it_counts     = it_counts ).

INSERT lo_metricdatum INTO TABLE lt_metricdata.

TRY.
    lo_cwt->putmetricdata(
      iv_namespace   = iv_namespace
      it_metricdata  = lt_metricdata ).
    MESSAGE 'Metric data set added.' TYPE 'I'.
  CATCH /aws1/cx_cwtinvparamvalueex.
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutMetricData referência da API AWS SDK for SAP ABAP.

Cenários

Começar a usar alarmes

O exemplo de código a seguir mostra como:

- Criar um alarme.
- Desabilitar ações de alarme.
- Descrever um alarme.
- Excluir um alarme.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.

"Create an alarm"
TRY.
    lo_cwt->putmetricalarm(
      iv_alarmname                = iv_alarm_name
      iv_comparisonoperator       = iv_comparison_operator
      iv_evaluationperiods        = iv_evaluation_periods
      iv_metricname               = iv_metric_name
      iv_namespace                = iv_namespace
      iv_statistic                = iv_statistic
      iv_threshold                = iv_threshold
      iv_actionsenabled           = iv_actions_enabled
      iv_alarmdescription         = iv_alarm_description
      iv_unit                     = iv_unit
      iv_period                   = iv_period
      it_dimensions               = it_dimensions ).
    MESSAGE 'Alarm created' TYPE 'I'.
  CATCH /aws1/cx_cwtlimitexceededfault.
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.

"Create an ABAP internal table for the created alarm."
lo_alarmname = NEW #( iv_value = iv_alarm_name ).
INSERT lo_alarmname INTO TABLE lt_alarmnames.

"Disable alarm actions."
TRY.
    lo_cwt->disablealarmactions(
      it_alarmnames               = lt_alarmnames ).
    MESSAGE 'Alarm actions disabled' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
```

```
        DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception->av_err_code }"
 - { lo_disablealarm_exception->av_err_msg }|.
        MESSAGE lv_disablealarm_error TYPE 'E'.
    ENDTRY.

    "Describe alarm using the same ABAP internal table."
    TRY.
        oo_result = lo_cwt->describealarms(                       " oo_result is
 returned for testing purpose "
          it_alarmnames                = lt_alarmnames ).
        MESSAGE 'Alarms retrieved' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
        DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
        MESSAGE lv_describealarms_error TYPE 'E'.
    ENDTRY.

    "Delete alarm."
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = lt_alarmnames ).
        MESSAGE 'Alarms deleted' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound.
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - DeleteAlarms
  - DescribeAlarms
  - DisableAlarmActions
  - PutMetricAlarm

CloudWatch Exemplos de registros usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP with CloudWatch Logs.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetQueryResults`

O código de exemplo a seguir mostra como usar `GetQueryResults`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cwl->getqueryresults(
          iv_queryid = iv_query_id ).

        " Display query status and result count
        DATA(lv_status) = oo_result->get_status( ).
        DATA(lt_results) = oo_result->get_results( ).
        DATA(lv_result_count) = lines( lt_results ).

        MESSAGE |Query status: { lv_status }. Retrieved { lv_result_count } log
event(s).| TYPE 'I'.
      CATCH /aws1/cx_cwlinvalidparameterex.
        MESSAGE 'Invalid parameter.' TYPE 'E'.
      CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cwlserviceunavailex.
```

```
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetQueryResults referência da API AWS SDK for SAP ABAP.

StartQuery

O código de exemplo a seguir mostra como usar `StartQuery`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_log_group_name = '/aws/lambda/my-function'
        " iv_query_string = 'fields @timestamp, @message | sort @timestamp desc |
limit 20'
        " iv_start_time and iv_end_time must be in Unix epoch milliseconds (ms since
Jan 1, 1970 00:00:00 UTC)
        oo_result = lo_cwl->startquery(
          iv_loggroupname = iv_log_group_name
          iv_starttime    = iv_start_time
          iv_endtime      = iv_end_time
          iv_querystring  = iv_query_string
          iv_limit        = iv_limit ).

        " Display the query ID for tracking
        DATA(lv_query_id) = oo_result->get_queryid( ).
        MESSAGE |Query started successfully with ID: { lv_query_id }| TYPE 'I'.
      CATCH /aws1/cx_cwlinvalidparameterex.
        MESSAGE 'Invalid parameter.' TYPE 'E'.
      CATCH /aws1/cx_cwllimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_cwlmalformedqueryex.
        MESSAGE 'Malformed query.' TYPE 'E'.
```

```
      CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cwlserviceunavailex.
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartQuery referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Cognito Identity Provider usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AdminInitiateAuth`

O código de exemplo a seguir mostra como usar `AdminInitiateAuth`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Set up authentication parameters
        DATA(lt_auth_params) = VALUE /aws1/
cl_cgpauthparamstype_w=>tt_authparameterstype(
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'PASSWORD'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_password ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpauthparamstype_w( iv_secret_hash )
          ) INTO TABLE lt_auth_params.
        ENDIF.

        oo_result = lo_cgp->admininitiateauth(
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_authflow = 'ADMIN_USER_PASSWORD_AUTH'
          it_authparameters = lt_auth_params
        ).

        DATA(lv_challenge) = oo_result->get_challengename( ).

        IF lv_challenge IS INITIAL.
          MESSAGE 'User successfully signed in.' TYPE 'I'.
        ELSE.
          MESSAGE |Authentication challenge required: { lv_challenge }.| TYPE 'I'.
        ENDIF.

      CATCH /aws1/cx_cgpusernotfoundex INTO DATA(lo_user_ex).
        MESSAGE |User { iv_user_name } not found.| TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized. Check credentials.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AdminInitiateAuth referência da API AWS SDK for SAP ABAP.

`AdminRespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `AdminRespondToAuthChallenge`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Build challenge responses
        DATA(lt_challenge_responses) = VALUE /aws1/
cl_cgpchallengerspstyp00=>tt_challengeresponsestype(
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpchallengerspstyp00( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'SOFTWARE_TOKEN_MFA_CODE'
              value = NEW /aws1/cl_cgpchallengerspstyp00( iv_mfa_code ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpchallengerspstyp00( iv_secret_hash )
          ) INTO TABLE lt_challenge_responses.
        ENDIF.

        DATA(lo_result) = lo_cgp->adminrespondtoauthchallenge(
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_challengename = 'SOFTWARE_TOKEN_MFA'
          it_challengeresponses = lt_challenge_responses
          iv_session = iv_session
        ).
```

```
      oo_auth_result = lo_result->get_authenticationresult( ).

      IF oo_auth_result IS BOUND.
        MESSAGE 'MFA challenge completed successfully.' TYPE 'I'.
      ELSE.
        " Another challenge might be required
        DATA(lv_next_challenge) = lo_result->get_challengename( ).
        MESSAGE |Additional challenge required: { lv_next_challenge }.| TYPE 'I'.
      ENDIF.

    CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
      MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

    CATCH /aws1/cx_cgpexpiredcodeex INTO DATA(lo_expired_ex).
      MESSAGE 'MFA code has expired.' TYPE 'E'.

    CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
      MESSAGE 'Not authorized. Check MFA configuration.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a AdminRespondToAuthChallenge referência da API AWS SDK for SAP ABAP.

`AssociateSoftwareToken`

O código de exemplo a seguir mostra como usar `AssociateSoftwareToken`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      DATA(lo_result) = lo_cgp->associatesoftwaretoken(
        iv_session = iv_session
      ).
```

```
      ov_secret_code = lo_result->get_secretcode( ).

      MESSAGE 'MFA secret code generated successfully.' TYPE 'I'.

    CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
      MESSAGE 'Session not found or expired.' TYPE 'E'.

    CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
      MESSAGE 'Not authorized to associate software token.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a AssociateSoftwareToken referência da API AWS SDK for SAP ABAP.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      DATA(lo_result) = lo_cgp->listusers(
        iv_userpoolid = iv_user_pool_id
      ).

      ot_users = lo_result->get_users( ).

      MESSAGE |Found { lines( ot_users ) } users in the pool.| TYPE 'I'.

    CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
      MESSAGE |User pool { iv_user_pool_id } not found.| TYPE 'E'.

    CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
```

```
        MESSAGE 'Not authorized to list users.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListUsers referência da API AWS SDK for SAP ABAP.

`VerifySoftwareToken`

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_cgp->verifysoftwaretoken(
          iv_session = iv_session
          iv_usercode = iv_user_code
        ).

        ov_status = lo_result->get_status( ).

        IF ov_status = 'SUCCESS'.
          MESSAGE 'MFA token verified successfully.' TYPE 'I'.
        ELSE.
          MESSAGE |MFA verification status: { ov_status }.| TYPE 'I'.
        ENDIF.

      CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
        MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

      CATCH /aws1/cx_cgpenbsoftwaretokmf00 INTO DATA(lo_enabled_ex).
        MESSAGE 'Software token MFA is already enabled.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a VerifySoftwareToken referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Comprehend usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Comprehend.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateDocumentClassifier`

O código de exemplo a seguir mostra como usar `CreateDocumentClassifier`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->createdocumentclassifier(
      iv_documentclassifiername = iv_classifier_name
      iv_languagecode = iv_language_code
      io_inputdataconfig = NEW /aws1/cl_cpddocclifierinpdat00(
        iv_s3uri = iv_training_s3_uri
      )
      iv_dataaccessrolearn = iv_data_access_role_arn
```

```
          iv_mode = iv_mode
        ).
        MESSAGE 'Document classifier creation started.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdresrclimitexcdex.
        MESSAGE 'Resource limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanytagsex.
        MESSAGE 'Too many tags.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDocumentClassifier referência da API AWS SDK for SAP ABAP.

`DeleteDocumentClassifier`

O código de exemplo a seguir mostra como usar `DeleteDocumentClassifier`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->deletedocumentclassifier(
          iv_documentclassifierarn = iv_classifier_arn
        ).
        MESSAGE 'Document classifier deleted.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourcenotfoundex.
```

```
      MESSAGE 'Resource not found.' TYPE 'E'.
    CATCH /aws1/cx_cpdresourceinuseex.
      MESSAGE 'Resource in use.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDocumentClassifier referência da API AWS SDK for SAP ABAP.

`DescribeDocumentClassificationJob`

O código de exemplo a seguir mostra como usar `DescribeDocumentClassificationJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_cpd->describedocclassificationjob(
        iv_jobid = iv_job_id
      ).
      MESSAGE 'Document classification job described.' TYPE 'I'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
    CATCH /aws1/cx_cpdjobnotfoundex.
      MESSAGE 'Job not found.' TYPE 'E'.
    CATCH /aws1/cx_cpdtoomanyrequestsex.
      MESSAGE 'Too many requests.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDocumentClassificationJob referência da API AWS SDK for SAP ABAP.

`DescribeDocumentClassifier`

O código de exemplo a seguir mostra como usar `DescribeDocumentClassifier`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->describedocumentclassifier(
      iv_documentclassifierarn = iv_classifier_arn
    ).
    MESSAGE 'Document classifier described.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdresourcenotfoundex.
    MESSAGE 'Resource not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDocumentClassifier referência da API AWS SDK for SAP ABAP.

`DescribeTopicsDetectionJob`

O código de exemplo a seguir mostra como usar `DescribeTopicsDetectionJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->describetopicsdetectionjob(
      iv_jobid = iv_job_id
    ).
    MESSAGE 'Topics detection job described.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdjobnotfoundex.
    MESSAGE 'Job not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeTopicsDetectionJob referência da API AWS SDK for SAP ABAP.

`DetectDominantLanguage`

O código de exemplo a seguir mostra como usar `DetectDominantLanguage`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
```

```
      oo_result = lo_cpd->detectdominantlanguage( iv_text = iv_text ).
      MESSAGE 'Languages detected.' TYPE 'I'.
    CATCH /aws1/cx_cpdtextsizelmtexcdex.
      MESSAGE 'Text size exceeds limit.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DetectDominantLanguage referência da API AWS SDK for SAP ABAP.

`DetectEntities`

O código de exemplo a seguir mostra como usar `DetectEntities`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_cpd->detectentities(
        iv_text = iv_text
        iv_languagecode = iv_language_code
      ).
      MESSAGE 'Entities detected.' TYPE 'I'.
    CATCH /aws1/cx_cpdtextsizelmtexcdex.
      MESSAGE 'Text size exceeds limit.' TYPE 'E'.
    CATCH /aws1/cx_cpdunsuppedlanguageex.
      MESSAGE 'Unsupported language.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DetectEntities referência da API AWS SDK for SAP ABAP.

`DetectKeyPhrases`

O código de exemplo a seguir mostra como usar `DetectKeyPhrases`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->detectkeyphrases(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'Key phrases detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectKeyPhrases referência da API AWS SDK for SAP ABAP.

`DetectPiiEntities`

O código de exemplo a seguir mostra como usar `DetectPiiEntities`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->detectpiientities(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'PII entities detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectPiiEntities referência da API AWS SDK for SAP ABAP.

`DetectSentiment`

O código de exemplo a seguir mostra como usar `DetectSentiment`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->detectsentiment(
          iv_text = iv_text
          iv_languagecode = iv_language_code
        ).
        MESSAGE 'Sentiment detected.' TYPE 'I'.
      CATCH /aws1/cx_cpdtextsizelmtexcdex.
        MESSAGE 'Text size exceeds limit.' TYPE 'E'.
      CATCH /aws1/cx_cpdunsuppedlanguageex.
        MESSAGE 'Unsupported language.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetectSentiment referência da API AWS SDK for SAP ABAP.

`DetectSyntax`

O código de exemplo a seguir mostra como usar `DetectSyntax`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->detectsyntax(
          iv_text = iv_text
          iv_languagecode = iv_language_code
        ).
        MESSAGE 'Syntax tokens detected.' TYPE 'I'.
      CATCH /aws1/cx_cpdtextsizelmtexcdex.
        MESSAGE 'Text size exceeds limit.' TYPE 'E'.
      CATCH /aws1/cx_cpdunsuppedlanguageex.
```

```
      MESSAGE 'Unsupported language.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DetectSyntax referência da API AWS SDK for SAP ABAP.

`ListDocumentClassificationJobs`

O código de exemplo a seguir mostra como usar `ListDocumentClassificationJobs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_cpd->listdocclassificationjobs( ).
      MESSAGE 'Document classification jobs listed.' TYPE 'I'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
    CATCH /aws1/cx_cpdtoomanyrequestsex.
      MESSAGE 'Too many requests.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidfilterex.
      MESSAGE 'Invalid filter.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListDocumentClassificationJobs referência da API AWS SDK for SAP ABAP.

`ListDocumentClassifiers`

O código de exemplo a seguir mostra como usar `ListDocumentClassifiers`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_cpd->listdocumentclassifiers( ).
    MESSAGE 'Document classifiers listed.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidfilterex.
    MESSAGE 'Invalid filter.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListDocumentClassifiers referência da API AWS SDK for SAP ABAP.

`ListTopicsDetectionJobs`

O código de exemplo a seguir mostra como usar `ListTopicsDetectionJobs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->listtopicsdetectionjobs( ).
        MESSAGE 'Topics detection jobs listed.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidfilterex.
        MESSAGE 'Invalid filter.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTopicsDetectionJobs referência da API AWS SDK for SAP ABAP.

`StartDocumentClassificationJob`

O código de exemplo a seguir mostra como usar `StartDocumentClassificationJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_cpd->startdocclassificationjob(
          iv_jobname = iv_job_name
          iv_documentclassifierarn = iv_classifier_arn
          io_inputdataconfig = NEW /aws1/cl_cpdinputdataconfig(
            iv_s3uri = iv_input_s3_uri
            iv_inputformat = iv_input_format
          )
          io_outputdataconfig = NEW /aws1/cl_cpdoutputdataconfig(
            iv_s3uri = iv_output_s3_uri
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
```

```
      ).
      MESSAGE 'Document classification job started.' TYPE 'I'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
    CATCH /aws1/cx_cpdtoomanyrequestsex.
      MESSAGE 'Too many requests.' TYPE 'E'.
    CATCH /aws1/cx_cpdresourcenotfoundex.
      MESSAGE 'Resource not found.' TYPE 'E'.
    CATCH /aws1/cx_cpdresourceunavailex.
      MESSAGE 'Resource unavailable.' TYPE 'E'.
    CATCH /aws1/cx_cpdkmskeyvalidationex.
      MESSAGE 'KMS key validation error.' TYPE 'E'.
    CATCH /aws1/cx_cpdtoomanytagsex.
      MESSAGE 'Too many tags.' TYPE 'E'.
    CATCH /aws1/cx_cpdresrclimitexcdex.
      MESSAGE 'Resource limit exceeded.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a StartDocumentClassificationJob referência da API AWS SDK for SAP ABAP.

StartTopicsDetectionJob

O código de exemplo a seguir mostra como usar `StartTopicsDetectionJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_cpd->starttopicsdetectionjob(
        iv_jobname = iv_job_name
        io_inputdataconfig = NEW /aws1/cl_cpdinputdataconfig(
          iv_s3uri = iv_input_s3_uri
```

```
            iv_inputformat = iv_input_format
          )
          io_outputdataconfig = NEW /aws1/cl_cpdoutputdataconfig(
            iv_s3uri = iv_output_s3_uri
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
        ).
        MESSAGE 'Topics detection job started.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdkmskeyvalidationex.
        MESSAGE 'KMS key validation error.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanytagsex.
        MESSAGE 'Too many tags.' TYPE 'E'.
      CATCH /aws1/cx_cpdresrclimitexcdex.
        MESSAGE 'Resource limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartTopicsDetectionJob referência da API AWS SDK for SAP ABAP.

AWS Config exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. AWS Config

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteConfigRule`

O código de exemplo a seguir mostra como usar `DeleteConfigRule`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
lo_cfs->deleteconfigrule( iv_rule_name ).
MESSAGE 'Deleted AWS Config rule.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a DeleteConfigRule referência da API AWS SDK for SAP ABAP.

`DescribeConfigRules`

O código de exemplo a seguir mostra como usar `DescribeConfigRules`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA(lo_result) = lo_cfs->describeconfigrules(
  it_configrulenames = VALUE /aws1/cl_cfsconfigrulenames_w=>tt_configrulenames(
    ( NEW /aws1/cl_cfsconfigrulenames_w( iv_rule_name ) )
  )
).
ot_cfg_rules = lo_result->get_configrules( ).
MESSAGE 'Retrieved AWS Config rule data.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a DescribeConfigRules referência da API AWS SDK for SAP ABAP.

`PutConfigRule`

O código de exemplo a seguir mostra como usar `PutConfigRule`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create a config rule for S3 bucket public read prohibition
    lo_cfs->putconfigrule(
      io_configrule = NEW /aws1/cl_cfsconfigrule(
        iv_configrulename = iv_rule_name
        iv_description = |S3 Public Read Prohibited Bucket Rule|
        io_scope = NEW /aws1/cl_cfsscope(
          it_complianceresourcetypes = VALUE /aws1/
cl_cfscplncresrctypes_w=>tt_complianceresourcetypes(
            ( NEW /aws1/cl_cfscplncresrctypes_w( |AWS::S3::Bucket| ) )
          )
        )
        io_source = NEW /aws1/cl_cfssource(
          iv_owner = |AWS|
          iv_sourceidentifier = |S3_BUCKET_PUBLIC_READ_PROHIBITED|
        )
        iv_inputparameters = '{}'
        iv_configrulestate = |ACTIVE|
      )
    ).
    MESSAGE 'Created AWS Config rule.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a PutConfigRule referência da API AWS SDK for SAP ABAP.

AWS Control Tower exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. AWS Control Tower

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DisableBaseline`

O código de exemplo a seguir mostra como usar `DisableBaseline`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Disable the baseline
    DATA(lo_output) = io_ctt->disablebaseline(
      iv_enabledbaselineidentifier = iv_enabled_baseline_identifier
    ).

    DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttbaselineopstatus.
```

```
        DO 100 TIMES.
          lv_status = get_baseline_operation(
            io_ctt = io_ctt
            iv_operation_id = lv_operation_id
          ).

          DATA(lv_msg) = |Baseline operation status: { lv_status }|.
          MESSAGE lv_msg TYPE 'I'.

          IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
            EXIT.
          ENDIF.

          " Wait 30 seconds
          WAIT UP TO 30 SECONDS.
        ENDDO.

        ov_operation_id = lv_operation_id.
        MESSAGE 'Baseline disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_cttconflictexception INTO DATA(lo_conflict).
        " Log conflict but don't fail - return empty operation ID
        DATA(lv_msg2) = |Conflict disabling baseline: { lo_conflict->get_text( ) }.
Skipping disable step.|.
        MESSAGE lv_msg2 TYPE 'I'.
        CLEAR ov_operation_id.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DisableBaseline referência da API AWS SDK for SAP ABAP.

`DisableControl`

O código de exemplo a seguir mostra como usar `DisableControl`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Disable the control
    DATA(lo_output) = io_ctt->disablecontrol(
      iv_controlidentifier = iv_control_arn
      iv_targetidentifier  = iv_target_identifier
    ).

    DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttcontrolopstatus.
    DO 100 TIMES.
      lv_status = get_control_operation(
        io_ctt = io_ctt
        iv_operation_id = lv_operation_id
      ).

      DATA(lv_msg) = |Control operation status: { lv_status }|.
      MESSAGE lv_msg TYPE 'I'.

      IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
        EXIT.
      ENDIF.

      " Wait 30 seconds
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Control disabled successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a DisableControl referência da API AWS SDK for SAP ABAP.

`EnableBaseline`

O código de exemplo a seguir mostra como usar `EnableBaseline`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Prepare parameters for enabling baseline
    DATA lt_parameters TYPE /aws1/
cl_cttenbdbaselineparam=>tt_enabledbaselineparameters.

    " Add Identity Center baseline parameter if provided
    IF iv_identity_center_baseline IS NOT INITIAL.
      " Create a JSON document with the baseline ARN value
      DATA(lv_json) = |\{ "IdentityCenterEnabledBaselineArn":
 "{ iv_identity_center_baseline }" \}|.
      DATA(lo_param) = NEW /aws1/cl_cttenbdbaselineparam(
        iv_key = 'IdentityCenterEnabledBaselineArn'
        io_value = /aws1/cl_rt_document=>from_json_str( lv_json )
      ).
      APPEND lo_param TO lt_parameters.
    ENDIF.

    " Enable the baseline
    DATA(lo_output) = io_ctt->enablebaseline(
      iv_baselineidentifier = iv_baseline_identifier
      iv_baselineversion    = iv_baseline_version
      iv_targetidentifier   = iv_target_identifier
      it_parameters         = lt_parameters
    ).

    DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttbaselineopstatus.
    DO 100 TIMES.
      lv_status = get_baseline_operation(
        io_ctt = io_ctt
        iv_operation_id = lv_operation_id
      ).
```

```
    DATA(lv_msg) = |Baseline operation status: { lv_status }|.
    MESSAGE lv_msg TYPE 'I'.

    IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
      EXIT.
    ENDIF.

    " Wait 30 seconds
    WAIT UP TO 30 SECONDS.
  ENDDO.

  ov_enabled_baseline_arn = lo_output->get_arn( ).
  MESSAGE 'Baseline enabled successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a EnableBaseline referência da API AWS SDK for SAP ABAP.

`EnableControl`

O código de exemplo a seguir mostra como usar `EnableControl`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  " Enable the control
  DATA(lo_output) = io_ctt->enablecontrol(
    iv_controlidentifier = iv_control_arn
    iv_targetidentifier  = iv_target_identifier
  ).

  DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

  " Wait for operation to complete
  DATA lv_status TYPE /aws1/cttcontrolopstatus.
  DO 100 TIMES.
```

```
    lv_status = get_control_operation(
      io_ctt = io_ctt
      iv_operation_id = lv_operation_id
    ).

    DATA(lv_msg) = |Control operation status: { lv_status }|.
    MESSAGE lv_msg TYPE 'I'.

    IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
      EXIT.
    ENDIF.

    " Wait 30 seconds
    WAIT UP TO 30 SECONDS.
  ENDDO.

  ov_operation_id = lv_operation_id.
  MESSAGE 'Control enabled successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a EnableControl referência da API AWS SDK for SAP ABAP.

`GetBaselineOperation`

O código de exemplo a seguir mostra como usar `GetBaselineOperation`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  DATA(lo_output) = io_ctt->getbaselineoperation(
    iv_operationidentifier = iv_operation_id
  ).

  ov_status = lo_output->get_baselineoperation( )->get_status( ).
```

- Para obter detalhes da API, consulte a GetBaselineOperation referência da API AWS SDK for SAP ABAP.

`GetControlOperation`

O código de exemplo a seguir mostra como usar `GetControlOperation`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA(lo_output) = io_ctt->getcontroloperation(
  iv_operationidentifier = iv_operation_id
).

ov_status = lo_output->get_controloperation( )->get_status( ).
```

- Para obter detalhes da API, consulte a GetControlOperation referência da API AWS SDK for SAP ABAP.

`ListBaselines`

O código de exemplo a seguir mostra como usar `ListBaselines`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA lt_baselines TYPE /aws1/cl_cttbaselinesummary=>tt_baselines.
DATA lv_nexttoken TYPE /aws1/cttstring.
```

```
    " List all baselines using pagination
    DO.
      DATA(lo_output) = io_ctt->listbaselines(
        iv_nexttoken = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_baselines( ) TO lt_baselines.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_baselines = lt_baselines.
    MESSAGE 'Listed baselines successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ListBaselines referência da API AWS SDK for SAP ABAP.

`ListEnabledBaselines`

O código de exemplo a seguir mostra como usar `ListEnabledBaselines`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_enabled_baselines TYPE /aws1/
cl_cttenbdbaselinesumm=>tt_enabledbaselines.
    DATA lv_nexttoken TYPE /aws1/cttlstenbdbaselinesnex00.

    " List all enabled baselines using pagination
    DO.
      DATA(lo_output) = io_ctt->listenabledbaselines(
```

```
        iv_nexttoken = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_enabledbaselines( ) TO lt_enabled_baselines.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_enabled_baselines = lt_enabled_baselines.
    MESSAGE 'Listed enabled baselines successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ListEnabledBaselines referência da API AWS SDK for SAP ABAP.

`ListEnabledControls`

O código de exemplo a seguir mostra como usar `ListEnabledControls`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_enabled_controls TYPE /aws1/
cl_cttenabledcontrolsumm=>tt_enabledcontrols.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all enabled controls using pagination
    DO.
      DATA(lo_output) = io_ctt->listenabledcontrols(
        iv_targetidentifier = iv_target_identifier
        iv_nexttoken        = lv_nexttoken
      ).
```

```
      APPEND LINES OF lo_output->get_enabledcontrols( ) TO lt_enabled_controls.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_enabled_controls = lt_enabled_controls.
    MESSAGE 'Listed enabled controls successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ListEnabledControls referência da API AWS SDK for SAP ABAP.

`ListLandingZones`

O código de exemplo a seguir mostra como usar `ListLandingZones`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_landing_zones TYPE /aws1/
cl_cttlandingzonesummary=>tt_landingzonesummaries.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all landing zones using pagination
    DO.
      DATA(lo_output) = io_ctt->listlandingzones(
        iv_nexttoken = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_landingzones( ) TO lt_landing_zones.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
```

```
      EXIT.
    ENDIF.
  ENDDO.

  ot_landing_zones = lt_landing_zones.
  MESSAGE 'Listed landing zones successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ListLandingZones referência da API AWS SDK for SAP ABAP.

`ResetEnabledBaseline`

O código de exemplo a seguir mostra como usar `ResetEnabledBaseline`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  " Reset the enabled baseline
  DATA(lo_output) = io_ctt->resetenabledbaseline(
    iv_enabledbaselineidentifier = iv_enabled_baseline_identifier
  ).

  DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

  " Wait for operation to complete
  DATA lv_status TYPE /aws1/cttbaselineopstatus.
  DO 100 TIMES.
    lv_status = get_baseline_operation(
      io_ctt = io_ctt
      iv_operation_id = lv_operation_id
    ).

    DATA(lv_msg) = |Baseline operation status: { lv_status }|.
    MESSAGE lv_msg TYPE 'I'.
```

```
      IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
        EXIT.
      ENDIF.

      " Wait 30 seconds
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Baseline reset successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a ResetEnabledBaseline referência da API AWS SDK for SAP ABAP.

Exemplos do Firehose usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com Firehose.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`PutRecord`

O código de exemplo a seguir mostra como usar `PutRecord`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_record) = NEW /aws1/cl_frhrecord( iv_data = iv_data ).

    DATA(lo_result) = lo_frh->putrecord(
      iv_deliverystreamname = iv_deliv_stream_name
      io_record             = lo_record ).

    MESSAGE 'Record sent to Firehose delivery stream.' TYPE 'I'.
  CATCH /aws1/cx_frhresourcenotfoundex.
    MESSAGE 'Delivery stream not found.' TYPE 'E'.
  CATCH /aws1/cx_frhinvalidargumentex.
    MESSAGE 'Invalid argument provided.' TYPE 'E'.
  CATCH /aws1/cx_frhserviceunavailex.
    MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutRecord referência da API AWS SDK for SAP ABAP.

`PutRecordBatch`

O código de exemplo a seguir mostra como usar `PutRecordBatch`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
```

```
        DATA(lo_result) = lo_frh->putrecordbatch(
          iv_deliverystreamname = iv_deliv_stream_name
          it_records            = it_records ).

        DATA(lv_failed_count) = lo_result->get_failedputcount( ).

        IF lv_failed_count > 0.
          MESSAGE |{ lv_failed_count } records failed to send.| TYPE 'I'.
        ELSE.
          MESSAGE 'All records sent successfully to Firehose delivery stream.' TYPE
'I'.
        ENDIF.
      CATCH /aws1/cx_frhresourcenotfoundex.
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
      CATCH /aws1/cx_frhinvalidargumentex.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
      CATCH /aws1/cx_frhserviceunavailex.
        MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutRecordBatch referência da API AWS SDK for SAP ABAP.

Exemplos do DynamoDB usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create an Amazon Dynamo DB table.

    TRY.
        DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
        DATA(lo_dyn) = /aws1/cl_dyn_factory=>create( lo_session ).
        DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                              iv_keytype = 'HASH' ) )
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                              iv_keytype = 'RANGE' ) ) ).
        DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                           iv_attributetype = 'N' ) )
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                           iv_attributetype = 'S' ) ) ).
```

```
        " Adjust read/write capacities as desired.
        DATA(lo_dynprovthroughput)  = NEW /aws1/cl_dynprovthroughput(
          iv_readcapacityunits = 5
          iv_writecapacityunits = 5 ).
        DATA(oo_result) = lo_dyn->createtable(
          it_keyschema = lt_keyschema
          iv_tablename = iv_table_name
          it_attributedefinitions = lt_attributedefinitions
          io_provisionedthroughput = lo_dynprovthroughput ).
        " Table creation can take some time. Wait till table exists before
 returning.
        lo_dyn->get_waiter( )->tableexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
        " It throws exception if the table already exists.
      CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
 { lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

    " Describe table
    TRY.
        DATA(lo_table) = lo_dyn->describetable( iv_tablename = iv_table_name ).
        DATA(lv_tablename) = lo_table->get_table( )->ask_tablename( ).
        MESSAGE 'The table name is ' && lv_tablename TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table does not exist' TYPE 'E'.
    ENDTRY.

    " Put items into the table.
    TRY.
        DATA(lo_resp_putitem) = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Jaws' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1975' }| ) ) )
```

```
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.5' }| ) ) )
          ) ).
        lo_resp_putitem = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s = 'Star
 Wars' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1978' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '8.1' }| ) ) )
          ) ).
        lo_resp_putitem = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Speed' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1994' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.9' }| ) ) )
          ) ).
        " TYPE REF TO /AWSEX/CL_AWS1_dyn_PUT_ITEM_OUTPUT
        MESSAGE '3 rows inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Get item from table.
```

```
    TRY.
        DATA(lo_resp_getitem) = lo_dyn->getitem(
          iv_tablename                = iv_table_name
          it_key                      = VALUE /aws1/cl_dynattributevalue=>tt_key(
           ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
             key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Jaws' ) ) )
           ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
             key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n =
 '1975' ) ) )
          ) ).
        DATA(lt_attr) = lo_resp_getitem->get_item( ).
        DATA(lo_title) = lt_attr[ key = 'title' ]-value.
        DATA(lo_year) = lt_attr[ key = 'year' ]-value.
        DATA(lo_rating) = lt_attr[ key = 'year' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Query item from table.
    TRY.
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
              ( NEW /aws1/cl_dynattributevalue( iv_n = '1975' ) ) ).
        DATA(lt_keyconditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
          ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
          key = 'year'
          value = NEW /aws1/cl_dyncondition(
            it_attributevaluelist = lt_attributelist
            iv_comparisonoperator = |EQ|
          ) ) ) ).
        DATA(lo_query_result) = lo_dyn->query(
          iv_tablename = iv_table_name
          it_keyconditions = lt_keyconditions ).
        DATA(lt_items) = lo_query_result->get_items( ).
        READ TABLE lo_query_result->get_items( ) INTO DATA(lt_item) INDEX 1.
        lo_title = lt_item[ key = 'title' ]-value.
        lo_year = lt_item[ key = 'year' ]-value.
        lo_rating = lt_item[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
```

```
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Scan items from table.
    TRY.
        DATA(lo_scan_result) = lo_dyn->scan( iv_tablename = iv_table_name ).
        lt_items = lo_scan_result->get_items( ).
        " Read the first item and display the attributes.
        READ TABLE lo_query_result->get_items( ) INTO lt_item INDEX 1.
        lo_title = lt_item[ key = 'title' ]-value.
        lo_year = lt_item[ key = 'year' ]-value.
        lo_rating = lt_item[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Update items from table.
    TRY.
        DATA(lt_attributeupdates) = VALUE /aws1/
cl_dynattrvalueupdate=>tt_attributeupdates(
          ( VALUE /aws1/cl_dynattrvalueupdate=>ts_attributeupdates_maprow(
          key = 'rating' value = NEW /aws1/cl_dynattrvalueupdate(
            io_value  = NEW /aws1/cl_dynattributevalue( iv_n = '7.6' )
            iv_action = |PUT| ) ) ) ).
        DATA(lt_key) = VALUE /aws1/cl_dynattributevalue=>tt_key(
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = '1975' ) ) )
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 '1980' ) ) ) ).
        DATA(lo_resp) = lo_dyn->updateitem(
          iv_tablename        = iv_table_name
          it_key              = lt_key
          it_attributeupdates = lt_attributeupdates ).
        MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
```

```
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Delete table.
    TRY.
        lo_dyn->deletetable( iv_tablename = iv_table_name ).
        lo_dyn->get_waiter( )->tablenotexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table deleted.' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dynresourceinuseex.
        MESSAGE 'The table cannot be deleted as it is in use' TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                              iv_keytype = 'HASH' ) )
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                              iv_keytype = 'RANGE' ) ) ).
        DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                           iv_attributetype = 'N' ) )
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                           iv_attributetype = 'S' ) ) ).

        " Adjust read/write capacities as desired.
        DATA(lo_dynprovthroughput)  = NEW /aws1/cl_dynprovthroughput(
          iv_readcapacityunits = 5
          iv_writecapacityunits = 5 ).
        oo_result = lo_dyn->createtable(
          it_keyschema = lt_keyschema
          iv_tablename = iv_table_name
          it_attributedefinitions = lt_attributedefinitions
          io_provisionedthroughput = lo_dynprovthroughput ).
        " Table creation can take some time. Wait till table exists before
 returning.
        lo_dyn->get_waiter( )->tableexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
        " This exception can happen if the table already exists.
      CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
 { lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTable referência da API AWS SDK for SAP ABAP.

DeleteItem

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       DATA(lo_resp) = lo_dyn->deleteitem(
         iv_tablename                = iv_table_name
         it_key                      = it_key_input ).
       MESSAGE 'Deleted one item.' TYPE 'I'.
     CATCH /aws1/cx_dyncondalcheckfaile00.
       MESSAGE 'A condition specified in the operation could not be evaluated.'
TYPE 'E'.
     CATCH /aws1/cx_dynresourcenotfoundex.
       MESSAGE 'The table or index does not exist' TYPE 'E'.
     CATCH /aws1/cx_dyntransactconflictex.
       MESSAGE 'Another transaction is using the item' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteItem referência da API AWS SDK for SAP ABAP.

DeleteTable

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_dyn->deletetable( iv_tablename = iv_table_name ).
    " Wait till the table is actually deleted.
    lo_dyn->get_waiter( )->tablenotexists(
      iv_max_wait_time = 200
      iv_tablename     = iv_table_name ).
    MESSAGE 'Table ' && iv_table_name && ' deleted.' TYPE 'I'.
  CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table ' && iv_table_name && ' does not exist' TYPE 'E'.
  CATCH /aws1/cx_dynresourceinuseex.
    MESSAGE 'The table cannot be deleted since it is in use' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTable referência da API AWS SDK for SAP ABAP.

`DescribeTable`

O código de exemplo a seguir mostra como usar `DescribeTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_dyn->describetable( iv_tablename = iv_table_name ).
    DATA(lv_tablename) = oo_result->get_table( )->ask_tablename( ).
```

```
        DATA(lv_tablearn) = oo_result->get_table( )->ask_tablearn( ).
        DATA(lv_tablestatus) = oo_result->get_table( )->ask_tablestatus( ).
        DATA(lv_itemcount) = oo_result->get_table( )->ask_itemcount( ).
        MESSAGE 'The table name is ' && lv_tablename
            && '. The table ARN is ' && lv_tablearn
            && '. The tablestatus is ' && lv_tablestatus
            && '. Item count is ' && lv_itemcount TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table ' && lv_tablename && ' does not exist' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeTable referência da API AWS SDK for SAP ABAP.

`GetItem`

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_item = lo_dyn->getitem(
          iv_tablename                = iv_table_name
          it_key                      = it_key ).
        DATA(lt_attr) = oo_item->get_item( ).
        DATA(lo_title) = lt_attr[ key = 'title' ]-value.
        DATA(lo_year) = lt_attr[ key = 'year' ]-value.
        DATA(lo_rating) = lt_attr[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( )
          && 'Movie year is: ' && lo_year->get_n( )
          && 'Moving rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetItem referência da API AWS SDK for SAP ABAP.

ListTables

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_dyn->listtables( ).
        " You can loop over the oo_result to get table properties like this.
        LOOP AT oo_result->get_tablenames( ) INTO DATA(lo_table_name).
          DATA(lv_tablename) = lo_table_name->get_value( ).
        ENDLOOP.
        DATA(lv_tablecount) = lines( oo_result->get_tablenames( ) ).
        MESSAGE 'Found ' && lv_tablecount && ' tables' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTables referência da API AWS SDK for SAP ABAP.

PutItem

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       DATA(lo_resp) = lo_dyn->putitem(
         iv_tablename = iv_table_name
         it_item      = it_item ).
       MESSAGE '1 row inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
     CATCH /aws1/cx_dyncondalcheckfaile00.
       MESSAGE 'A condition specified in the operation could not be evaluated.'
TYPE 'E'.
     CATCH /aws1/cx_dynresourcenotfoundex.
       MESSAGE 'The table or index does not exist' TYPE 'E'.
     CATCH /aws1/cx_dyntransactconflictex.
       MESSAGE 'Another transaction is using the item' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a PutItem referência da API AWS SDK for SAP ABAP.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
```

```
        " Query movies for a given year .
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
            ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_year }| ) ) ).
        DATA(lt_key_conditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
          ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
          key = 'year'
          value = NEW /aws1/cl_dyncondition(
          it_attributevaluelist = lt_attributelist
          iv_comparisonoperator = |EQ|
          ) ) ) ).
        oo_result = lo_dyn->query(
          iv_tablename = iv_table_name
          it_keyconditions = lt_key_conditions ).
        DATA(lt_items) = oo_result->get_items( ).
        "You can loop over the results to get item attributes.
        LOOP AT lt_items INTO DATA(lt_item).
          DATA(lo_title) = lt_item[ key = 'title' ]-value.
          DATA(lo_year) = lt_item[ key = 'year' ]-value.
        ENDLOOP.
        DATA(lv_count) = oo_result->get_count( ).
        MESSAGE 'Item count is: ' && lv_count TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Query na Referência da API AWS SDK para SAP ABAP.

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Scan movies for rating greater than or equal to the rating specified
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
            ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_rating }| ) ) ).
        DATA(lt_filter_conditions) = VALUE /aws1/
cl_dyncondition=>tt_filterconditionmap(
          ( VALUE /aws1/cl_dyncondition=>ts_filterconditionmap_maprow(
          key = 'rating'
          value = NEW /aws1/cl_dyncondition(
          it_attributevaluelist = lt_attributelist
          iv_comparisonoperator = |GE|
          ) ) ) ).
        oo_scan_result = lo_dyn->scan( iv_tablename = iv_table_name
          it_scanfilter = lt_filter_conditions ).
        DATA(lt_items) = oo_scan_result->get_items( ).
        LOOP AT lt_items INTO DATA(lo_item).
          " You can loop over to get individual attributes.
          DATA(lo_title) = lo_item[ key = 'title' ]-value.
          DATA(lo_year) = lo_item[ key = 'year' ]-value.
        ENDLOOP.
        DATA(lv_count) = oo_scan_result->get_count( ).
        MESSAGE 'Found ' && lv_count && ' items' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Scan na Referência da API AWS SDK para SAP ABAP.

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_output = lo_dyn->updateitem(
          iv_tablename        = iv_table_name
          it_key              = it_item_key
          it_attributeupdates = it_attribute_updates ).
        MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateItem referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon EC2 usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon EC2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).   " oo_result is
 returned for testing purposes. "
        MESSAGE 'Allocated an Elastic IP address.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AllocateAddress referência da API AWS SDK for SAP ABAP.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->associateaddress(                        " oo_result is
 returned for testing purposes. "
            iv_allocationid = iv_allocation_id
            iv_instanceid = iv_instance_id ).
```

```
        MESSAGE 'Associated an Elastic IP address with an EC2 instance.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AssociateAddress referência da API AWS SDK for SAP ABAP.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create IP permissions for SSH access (port 22)
    " iv_cidr_ip = '192.0.2.0/24'
    DATA lt_ip_permissions TYPE /aws1/cl_ec2ippermission=>tt_ippermissionlist.
    DATA(lo_ip_permission) = NEW /aws1/cl_ec2ippermission(
      iv_ipprotocol = 'tcp'
      iv_fromport = 22
      iv_toport = 22
      it_ipranges = VALUE /aws1/cl_ec2iprange=>tt_iprangelist(
        ( NEW /aws1/cl_ec2iprange( iv_cidrip = iv_cidr_ip ) )
      )
    ).
    APPEND lo_ip_permission TO lt_ip_permissions.

    TRY.
        oo_result = lo_ec2->authsecuritygroupingress(             " oo_result is
returned for testing purposes. "
          iv_groupid = iv_group_id
          it_ippermissions = lt_ip_permissions ).
```

```
        MESSAGE 'Authorized ingress rule for security group.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AuthorizeSecurityGroupIngress referência da API AWS SDK for SAP ABAP.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).
              " oo_result is returned for testing purposes. "
        MESSAGE 'Amazon EC2 key pair created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateKeyPair referência da API AWS SDK for SAP ABAP.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->createsecuritygroup(                " oo_result is
returned for testing purposes. "
          iv_description = 'Security group example'
          iv_groupname = iv_security_group_name
          iv_vpcid = iv_vpc_id ).
        MESSAGE 'Security group created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateSecurityGroup referência da API AWS SDK for SAP ABAP.

`CreateVpc`

O código de exemplo a seguir mostra como usar `CreateVpc`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_cidr_block = '10.0.0.0/16'
    TRY.
```

```
        oo_result = lo_ec2->createvpc( iv_cidrblock = iv_cidr_block ).             "
 oo_result is returned for testing purposes. "
        DATA(lv_vpc_id) = oo_result->get_vpc( )->get_vpcid( ).
        MESSAGE 'Created VPC.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateVpc referência da API AWS SDK for SAP ABAP.

`CreateVpcEndpoint`

O código de exemplo a seguir mostra como usar `CreateVpcEndpoint`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_vpc_id = 'vpc-abc123'
    " iv_service_name = 'com.amazonaws.region.service'
    TRY.
        oo_result = lo_ec2->createvpcendpoint(             " oo_result is returned
 for testing purposes. "
          iv_vpcid = iv_vpc_id
          iv_servicename = iv_service_name
          it_routetableids = it_route_table_ids ).
        DATA(lv_vpc_endpoint_id) = oo_result->get_vpcendpoint( )-
>get_vpcendpointid( ).
        MESSAGE 'Created VPC endpoint.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateVpcEndpoint referência da API AWS SDK for SAP ABAP.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->deletekeypair( iv_keyname = iv_key_name ).
        MESSAGE 'Amazon EC2 key pair deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteKeyPair referência da API AWS SDK for SAP ABAP.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).
        MESSAGE 'Security group deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteSecurityGroup referência da API AWS SDK for SAP ABAP.

`DeleteVpc`

O código de exemplo a seguir mostra como usar `DeleteVpc`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->deletevpc( iv_vpcid = iv_vpc_id ).
        MESSAGE 'Deleted VPC.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
```

```
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteVpc referência da API AWS SDK for SAP ABAP.

DeleteVpcEndpoints

O código de exemplo a seguir mostra como usar `DeleteVpcEndpoints`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->deletevpcendpoints( it_vpcendpointids = it_vpc_endpoint_ids ).
        MESSAGE 'Deleted VPC endpoint(s).' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteVpcEndpoints referência da API AWS SDK for SAP ABAP.

DescribeAddresses

O código de exemplo a seguir mostra como usar `DescribeAddresses`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeaddresses( ).                       " oo_result
 is returned for testing purposes. "
        DATA(lt_addresses) = oo_result->get_addresses( ).
        MESSAGE 'Retrieved information about Elastic IP addresses.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAddresses referência da API AWS SDK for SAP ABAP.

`DescribeAvailabilityZones`

O código de exemplo a seguir mostra como usar `DescribeAvailabilityZones`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeavailabilityzones( ).                       "
 oo_result is returned for testing purposes. "
        DATA(lt_zones) = oo_result->get_availabilityzones( ).
```

```
        MESSAGE 'Retrieved information about Availability Zones.' TYPE 'I'.

      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeAvailabilityZones referência da API AWS SDK for SAP ABAP.

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeimages( it_imageids = it_image_ids ).
  " oo_result is returned for testing purposes. "
        DATA(lt_images) = oo_result->get_images( ).
        MESSAGE 'Retrieved information about Amazon Machine Images (AMIs).' TYPE
 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeImages referência da API AWS SDK for SAP ABAP.

`DescribeInstanceTypes`

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create filters for architecture and instance type patterns
    " iv_architecture = 'x86_64'
    DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
    APPEND NEW /aws1/cl_ec2filter(
      iv_name = 'processor-info.supported-architecture'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( iv_architecture ) )
      )
    ) TO lt_filters.
    " Filter for instance type patterns like '*.micro', '*.small'
    APPEND NEW /aws1/cl_ec2filter(
      iv_name = 'instance-type'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( '*.micro' ) )
        ( NEW /aws1/cl_ec2valuestringlist_w( '*.small' ) )
      )
    ) TO lt_filters.

    TRY.
        oo_result = lo_ec2->describeinstancetypes( it_filters = lt_filters ).
      " oo_result is returned for testing purposes. "
        DATA(lt_instance_types) = oo_result->get_instancetypes( ).
        MESSAGE 'Retrieved information about EC2 instance types.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeInstanceTypes referência da API AWS SDK for SAP ABAP.

`DescribeInstances`

O código de exemplo a seguir mostra como usar `DescribeInstances`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeinstances( ).                       " oo_result
 is returned for testing purposes. "

        " Retrieving details of EC2 instances. "
        DATA: lv_istance_id    TYPE /aws1/ec2string,
              lv_status        TYPE /aws1/ec2instancestatename,
              lv_instance_type TYPE /aws1/ec2instancetype,
              lv_image_id      TYPE /aws1/ec2string.
        LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
          LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
            lv_istance_id = lo_instance->get_instanceid( ).
            lv_status = lo_instance->get_state( )->get_name( ).
            lv_instance_type = lo_instance->get_instancetype( ).
            lv_image_id = lo_instance->get_imageid( ).
          ENDLOOP.
        ENDLOOP.
        MESSAGE 'Retrieved information about EC2 instances.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeInstances referência da API AWS SDK for SAP ABAP.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describekeypairs( ).                        " oo_result
 is returned for testing purposes. "
        DATA(lt_key_pairs) = oo_result->get_keypairs( ).
        MESSAGE 'Retrieved information about key pairs.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeKeyPairs referência da API AWS SDK for SAP ABAP.

`DescribeRegions`

O código de exemplo a seguir mostra como usar `DescribeRegions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ec2->describeregions( ).                        " oo_result
 is returned for testing purposes. "
        DATA(lt_regions) = oo_result->get_regions( ).
        MESSAGE 'Retrieved information about Regions.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeRegions referência da API AWS SDK for SAP ABAP.

`DescribeRouteTables`

O código de exemplo a seguir mostra como usar `DescribeRouteTables`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create filter for VPC ID
    " iv_vpc_id = 'vpc-abc123'
    DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
    APPEND NEW /aws1/cl_ec2filter(
```

```
      iv_name = 'vpc-id'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( iv_vpc_id ) )
      )
    ) TO lt_filters.

    TRY.
        oo_result = lo_ec2->describeroutetables( it_filters = lt_filters ).
    " oo_result is returned for testing purposes. "
        DATA(lt_route_tables) = oo_result->get_routetables( ).
        MESSAGE 'Retrieved information about route tables.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeRouteTables referência da API AWS SDK for SAP ABAP.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA lt_group_ids TYPE /aws1/cl_ec2groupidstrlist_w=>tt_groupidstringlist.
        APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO
 lt_group_ids.
        oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).
      " oo_result is returned for testing purposes. "
        DATA(lt_security_groups) = oo_result->get_securitygroups( ).
        MESSAGE 'Retrieved information about security groups.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeSecurityGroups referência da API AWS SDK for SAP ABAP.

`MonitorInstances`

O código de exemplo a seguir mostra como usar `MonitorInstances`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to monitor
 the instance without actually making the request. "
        lo_ec2->monitorinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to monitor this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to enable detailed monitoring completed.' TYPE 'I'.
          " DryRun is set to false to enable detailed monitoring. "
```

```
          lo_ec2->monitorinstances(
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Detailed monitoring enabled.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to monitor this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to enable detailed monitoring failed. User does not have
 the permissions to monitor the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a MonitorInstances referência da API AWS SDK for SAP ABAP.

`RebootInstances`

O código de exemplo a seguir mostra como usar `RebootInstances`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to reboot
 the instance without actually making the request. "
```

```
        lo_ec2->rebootinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to reboot this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to reboot instance completed.' TYPE 'I'.
          " DryRun is set to false to make a reboot request. "
          lo_ec2->rebootinstances(
             it_instanceids = lt_instance_ids
             iv_dryrun = abap_false ).
          MESSAGE 'Instance rebooted.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to reboot this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to reboot instance failed. User does not have permissions
 to reboot the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RebootInstances referência da API AWS SDK for SAP ABAP.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).
        MESSAGE 'Elastic IP address released.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ReleaseAddress referência da API AWS SDK for SAP ABAP.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create tags for resource created during instance launch. "
    DATA lt_tagspecifications TYPE /aws1/
cl_ec2tagspecification=>tt_tagspecificationlist.
    DATA ls_tagspecifications LIKE LINE OF lt_tagspecifications.
    ls_tagspecifications = NEW /aws1/cl_ec2tagspecification(
      iv_resourcetype = 'instance'
      it_tags = VALUE /aws1/cl_ec2tag=>tt_taglist(
        ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
      ) ).
    APPEND ls_tagspecifications TO lt_tagspecifications.

    TRY.
        " Create/launch Amazon Elastic Compute Cloud (Amazon EC2) instance. "
        oo_result = lo_ec2->runinstances(                           " oo_result is
 returned for testing purposes. "
```

```
          iv_imageid = iv_ami_id
          iv_instancetype = 't3.micro'
          iv_maxcount = 1
          iv_mincount = 1
          it_tagspecifications = lt_tagspecifications
          iv_subnetid = iv_subnet_id ).
        MESSAGE 'EC2 instance created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RunInstances referência da API AWS SDK for SAP ABAP.

`StartInstances`

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to start
 the instance without actually making the request. "
        lo_ec2->startinstances(
          it_instanceids = lt_instance_ids
```

```
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to start this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to start instance completed.' TYPE 'I'.
          " DryRun is set to false to start instance. "
          oo_result = lo_ec2->startinstances(           " oo_result is returned for
 testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Successfully started the EC2 instance.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to start this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to start instance failed. User does not have permissions
 to start the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartInstances referência da API AWS SDK for SAP ABAP.

`StopInstances`

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to stop
 the instance without actually making the request. "
        lo_ec2->stopinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to stop this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to stop instance completed.' TYPE 'I'.
          " DryRun is set to false to stop instance. "
          oo_result = lo_ec2->stopinstances(           " oo_result is returned for
 testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Successfully stopped the EC2 instance.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to stop this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to stop instance failed. User does not have permissions
 to stop the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StopInstances referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon ECR usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon ECR.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateRepository`

O código de exemplo a seguir mostra como usar `CreateRepository`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_repository_name = 'my-repository'
        oo_result = lo_ecr->createrepository(
          iv_repositoryname = iv_repository_name ).
        DATA(lv_repository_uri) = oo_result->get_repository( )-
>get_repositoryuri( ).
        MESSAGE |Repository created with URI: { lv_repository_uri }| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositoryalrexex.
        " If repository already exists, retrieve it
        DATA lt_repo_names TYPE /aws1/
cl_ecrrepositorynamels00=>tt_repositorynamelist.
```

```
        APPEND NEW /aws1/cl_ecrrepositorynamels00( iv_value = iv_repository_name )
TO lt_repo_names.
        DATA(lo_describe_result) = lo_ecr->describerepositories( it_repositorynames
= lt_repo_names ).
        DATA(lt_repos) = lo_describe_result->get_repositories( ).
        IF lines( lt_repos ) > 0.
          READ TABLE lt_repos INDEX 1 INTO DATA(lo_repo).
          oo_result = NEW /aws1/cl_ecrcrerepositoryrsp( io_repository = lo_repo ).
          MESSAGE |Repository { iv_repository_name } already exists.| TYPE 'I'.
        ENDIF.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateRepository referência da API AWS SDK for SAP ABAP.

`DeleteRepository`

O código de exemplo a seguir mostra como usar `DeleteRepository`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_repository_name = 'my-repository'
        lo_ecr->deleterepository(
          iv_repositoryname = iv_repository_name
          iv_force = abap_true ).
        MESSAGE |Repository { iv_repository_name } deleted.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteRepository referência da API AWS SDK for SAP ABAP.

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       " iv_repository_name = 'my-repository'
       " it_image_ids = VALUE #( ( NEW /aws1/cl_ecrimageidentifier( iv_imagetag =
'latest' ) ) )
       IF it_image_ids IS NOT INITIAL.
         oo_result = lo_ecr->describeimages(
           iv_repositoryname = iv_repository_name
           it_imageids = it_image_ids ).
       ELSE.
         oo_result = lo_ecr->describeimages(
           iv_repositoryname = iv_repository_name ).
       ENDIF.
       DATA(lt_image_details) = oo_result->get_imagedetails( ).
       MESSAGE |Found { lines( lt_image_details ) } images in repository.| TYPE
'I'.
     CATCH /aws1/cx_ecrrepositorynotfndex.
       MESSAGE 'Repository not found.' TYPE 'I'.
     CATCH /aws1/cx_ecrimagenotfoundex.
       MESSAGE 'Image not found.' TYPE 'I'.
     CATCH /aws1/cx_ecrinvalidparameterex.
       MESSAGE 'Invalid parameter provided.' TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeImages referência da API AWS SDK for SAP ABAP.

`DescribeRepositories`

O código de exemplo a seguir mostra como usar `DescribeRepositories`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " it_repository_names = VALUE #( ( NEW /aws1/
cl_ecrrepositorynamels00( iv_value = 'my-repository' ) ) )
        oo_result = lo_ecr->describerepositories(
          it_repositorynames = it_repository_names ).
        DATA(lt_repositories) = oo_result->get_repositories( ).
        MESSAGE |Found { lines( lt_repositories ) } repositories.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeRepositories referência da API AWS SDK for SAP ABAP.

`GetAuthorizationToken`

O código de exemplo a seguir mostra como usar `GetAuthorizationToken`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
      oo_result = lo_ecr->getauthorizationtoken( ).
      DATA(lt_auth_data) = oo_result->get_authorizationdata( ).
      IF lines( lt_auth_data ) > 0.
        READ TABLE lt_auth_data INDEX 1 INTO DATA(lo_auth_data).
        DATA(lv_token) = lo_auth_data->get_authorizationtoken( ).
        MESSAGE 'Authorization token retrieved.' TYPE 'I'.
      ENDIF.
    CATCH /aws1/cx_ecrserverexception.
      MESSAGE 'Server exception occurred.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetAuthorizationToken referência da API AWS SDK for SAP ABAP.

`GetRepositoryPolicy`

O código de exemplo a seguir mostra como usar `GetRepositoryPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " iv_repository_name = 'my-repository'
      oo_result = lo_ecr->getrepositorypolicy(
        iv_repositoryname = iv_repository_name ).
      DATA(lv_policy_text) = oo_result->get_policytext( ).
      MESSAGE 'Repository policy retrieved.' TYPE 'I'.
    CATCH /aws1/cx_ecrrepositorynotfndex.
      MESSAGE 'Repository not found.' TYPE 'I'.
    CATCH /aws1/cx_ecrrepositoryplynot00.
      MESSAGE 'Repository policy not found.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetRepositoryPolicy referência da API AWS SDK for SAP ABAP.

PutLifeCyclePolicy

O código de exemplo a seguir mostra como usar `PutLifeCyclePolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_repository_name = 'my-repository'
        " iv_lifecycle_policy_text = '{"rules":
[{"rulePriority":1,"description":"Expire images older than 14 days",...}]}'
        lo_ecr->putlifecyclepolicy(
          iv_repositoryname = iv_repository_name
          iv_lifecyclepolicytext = iv_lifecycle_policy_text ).
        MESSAGE |Lifecycle policy set for repository { iv_repository_name }.| TYPE
 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrvalidationex.
        MESSAGE 'Invalid lifecycle policy format.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutLifeCyclePolicy referência da API AWS SDK for SAP ABAP.

SetRepositoryPolicy

O código de exemplo a seguir mostra como usar `SetRepositoryPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_repository_name = 'my-repository'
    " iv_policy_text = '{"Version":"2012-10-17",       "Statement":[...]}'
    lo_ecr->setrepositorypolicy(
      iv_repositoryname = iv_repository_name
      iv_policytext = iv_policy_text ).
    MESSAGE |Policy set for repository { iv_repository_name }.| TYPE 'I'.
  CATCH /aws1/cx_ecrrepositorynotfndex.
    MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a SetRepositoryPolicy referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon EMR usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon EMR.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AddJobFlowSteps`

O código de exemplo a seguir mostra como usar `AddJobFlowSteps`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Build args list for Spark submit
    DATA lt_args TYPE /aws1/cl_emrxmlstringlist_w=>tt_xmlstringlist.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( 'spark-submit' ) TO lt_args.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( '--deploy-mode' ) TO lt_args.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( 'cluster' ) TO lt_args.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( iv_script_uri ) TO lt_args.
    APPEND LINES OF it_script_args TO lt_args.

    " Create step configuration
    DATA(lo_hadoop_jar_step) = NEW /aws1/cl_emrhadoopjarstepcfg(
      iv_jar = 'command-runner.jar'
      it_args = lt_args
    ).

    DATA(lo_step_config) = NEW /aws1/cl_emrstepconfig(
      iv_name = iv_name
      iv_actiononfailure = 'CONTINUE'
      io_hadoopjarstep = lo_hadoop_jar_step
    ).

    DATA lt_steps TYPE /aws1/cl_emrstepconfig=>tt_stepconfiglist.
    APPEND lo_step_config TO lt_steps.

    DATA(lo_result) = lo_emr->addjobflowsteps(
      iv_jobflowid = iv_cluster_id
      it_steps = lt_steps
    ).
```

```
      " Get first step ID
      DATA(lt_step_ids) = lo_result->get_stepids( ).
      READ TABLE lt_step_ids INDEX 1 INTO DATA(lo_step_id_obj).
      IF sy-subrc = 0.
        ov_step_id = lo_step_id_obj->get_value( ).
        MESSAGE |Step added with ID { ov_step_id }| TYPE 'I'.
      ENDIF.
    CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
      DATA(lv_error) = lo_internal_error->if_message~get_text( ).
      MESSAGE lv_error TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a AddJobFlowSteps referência da API AWS SDK for SAP ABAP.

`DescribeCluster`

O código de exemplo a seguir mostra como usar `DescribeCluster`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_emr->describecluster(
        iv_clusterid = iv_cluster_id
      ).
      DATA(lo_cluster) = oo_result->get_cluster( ).
      DATA(lv_cluster_name) = lo_cluster->get_name( ).
      MESSAGE |Retrieved cluster information for { lv_cluster_name }| TYPE 'I'.
    CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
      DATA(lv_error) = lo_internal_error->if_message~get_text( ).
      MESSAGE lv_error TYPE 'E'.
    CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
      lv_error = lo_invalid_error->if_message~get_text( ).
      MESSAGE lv_error TYPE 'E'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeCluster referência da API AWS SDK for SAP ABAP.

DescribeStep

O código de exemplo a seguir mostra como usar `DescribeStep`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_emr->describestep(
          iv_clusterid = iv_cluster_id
          iv_stepid = iv_step_id
        ).
        DATA(lo_step) = oo_result->get_step( ).
        DATA(lv_step_name) = lo_step->get_name( ).
        MESSAGE |Retrieved step information for { lv_step_name }| TYPE 'I'.
      CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
      CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
        lv_error = lo_invalid_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeStep referência da API AWS SDK for SAP ABAP.

`ListSteps`

O código de exemplo a seguir mostra como usar `ListSteps`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_emr->liststeps(
      iv_clusterid = iv_cluster_id
    ).
    DATA(lt_steps) = oo_result->get_steps( ).
    DATA(lv_step_count) = lines( lt_steps ).
    MESSAGE |Retrieved { lv_step_count } steps for cluster| TYPE 'I'.
  CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
  CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
    lv_error = lo_invalid_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListSteps referência da API AWS SDK for SAP ABAP.

`RunJobFlow`

O código de exemplo a seguir mostra como usar `RunJobFlow`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create instances configuration
        DATA(lo_instances) = NEW /aws1/cl_emrjobflowinstsconfig(
          iv_masterinstancetype = 'm5.xlarge'
          iv_slaveinstancetype = 'm5.xlarge'
          iv_instancecount = 3
          iv_keepjobflowalivewhennos00 = iv_keep_alive
          iv_emrmanagedmastersecgroup = iv_primary_sec_grp
          iv_emrmanagedslavesecgroup = iv_secondary_sec_grp
        ).

        DATA(lo_result) = lo_emr->runjobflow(
          iv_name = iv_name
          iv_loguri = iv_log_uri
          iv_releaselabel = 'emr-5.30.1'
          io_instances = lo_instances
          it_steps = it_steps
          it_applications = it_applications
          iv_jobflowrole = iv_job_flow_role
          iv_servicerole = iv_service_role
          iv_ebsrootvolumesize = 10
          iv_visibletoallusers = abap_true
        ).

        ov_cluster_id = lo_result->get_jobflowid( ).
        MESSAGE 'EMR cluster created successfully.' TYPE 'I'.
      CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
      CATCH /aws1/cx_emrclientexc INTO DATA(lo_client_error).
        lv_error = lo_client_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RunJobFlow referência da API AWS SDK for SAP ABAP.

`TerminateJobFlows`

O código de exemplo a seguir mostra como usar `TerminateJobFlows`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA lt_cluster_ids TYPE /aws1/cl_emrxmlstringlist_w=>tt_xmlstringlist.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( iv_cluster_id ) TO lt_cluster_ids.

    lo_emr->terminatejobflows(
      it_jobflowids = lt_cluster_ids
    ).
    MESSAGE 'EMR cluster terminated successfully.' TYPE 'I'.
  CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a TerminateJobFlows referência da API AWS SDK for SAP ABAP.

EventBridge Exemplos de agendadores usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com EventBridge o Scheduler.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateSchedule`

O código de exemplo a seguir mostra como usar `CreateSchedule`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Constants for time calculations
        DATA lv_start_date TYPE /aws1/scdstartdate.
        DATA lv_end_date TYPE /aws1/scdenddate.
        DATA lv_start_timestamp TYPE timestamp.
        DATA lv_end_timestamp TYPE timestamp.
        DATA lv_hours_to_run TYPE i VALUE 1.

        " Get current timestamp
        GET TIME STAMP FIELD lv_start_timestamp.

        " Add 1 hour to the current timestamp using CL_ABAP_TSTMP
        lv_end_timestamp = cl_abap_tstmp=>add(
          tstmp = lv_start_timestamp
          secs = lv_hours_to_run * 3600 ).

        " Convert timestamps to decimal format for AWS API
        lv_start_date = lv_start_timestamp.
        lv_end_date = lv_end_timestamp.

        " Prepare flexible time window configuration
        DATA lo_flexible_time_window TYPE REF TO /aws1/cl_scdflexibletimewindow.
        IF iv_use_flexible_time_win = abap_true.
          " iv_use_flexible_time_win = ABAP_TRUE
          " Example: Set MaximumWindowInMinutes to 10 for flexible window
          lo_flexible_time_window = NEW /aws1/cl_scdflexibletimewindow(
            iv_mode = 'FLEXIBLE'
            iv_maximumwindowinminutes = 10 ).
        ELSE.
```

```
          lo_flexible_time_window = NEW /aws1/cl_scdflexibletimewindow(
            iv_mode = 'OFF' ).
        ENDIF.

        " Prepare target configuration
        " Example iv_target_arn = 'arn:aws:sqs:us-east-1:123456789012:my-queue'
        " Example iv_role_arn = 'arn:aws:iam::123456789012:role/SchedulerRole'
        " Example iv_input = '{"message": "Hello from EventBridge Scheduler"}'
        DATA(lo_target) = NEW /aws1/cl_scdtarget(
          iv_arn = iv_target_arn
          iv_rolearn = iv_role_arn
          iv_input = iv_input ).

        " Set action after completion if needed
        DATA lv_action_after_completion TYPE /aws1/scdactionaftercompletion.
        IF iv_delete_after_completion = abap_true.
          " iv_delete_after_completion = ABAP_TRUE
          lv_action_after_completion = 'DELETE'.
        ELSE.
          lv_action_after_completion = 'NONE'.
        ENDIF.

        " Create the schedule
        " Example iv_name = 'my-schedule'
        " Example iv_schedule_expression = 'rate(15 minutes)'
        " Example iv_schedule_group_name = 'my-schedule-group'
        DATA(lo_result) = lo_scd->createschedule(
          iv_name = iv_name
          iv_scheduleexpression = iv_schedule_expression
          iv_groupname = iv_schedule_group_name
          io_target = lo_target
          io_flexibletimewindow = lo_flexible_time_window
          iv_startdate = lv_start_date
          iv_enddate = lv_end_date
          iv_actionaftercompletion = lv_action_after_completion ).

        ov_schedule_arn = lo_result->get_schedulearn( ).
        MESSAGE 'Schedule created successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdconflictexception INTO DATA(lo_conflict_ex).
        DATA(lv_error) = |Conflict creating schedule: { lo_conflict_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
```

```
        DATA(lv_generic_error) = |Error creating schedule: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateSchedule referência da API AWS SDK for SAP ABAP.

`CreateScheduleGroup`

O código de exemplo a seguir mostra como usar `CreateScheduleGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example iv_name = 'my-schedule-group'
        DATA(lo_result) = lo_scd->createschedulegroup(
          iv_name = iv_name ).

        ov_schedule_group_arn = lo_result->get_schedulegrouparn( ).
        MESSAGE 'Schedule group created successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdconflictexception INTO DATA(lo_conflict_ex).
        DATA(lv_error) = |Conflict creating schedule group: { lo_conflict_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error creating schedule group: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateScheduleGroup referência da API AWS SDK for SAP ABAP.

`DeleteSchedule`

O código de exemplo a seguir mostra como usar `DeleteSchedule`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example iv_name = 'my-schedule'
        " Example iv_schedule_group_name = 'my-schedule-group'
        lo_scd->deleteschedule(
          iv_name = iv_name
          iv_groupname = iv_schedule_group_name ).
        MESSAGE 'Schedule deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdresourcenotfoundex INTO DATA(lo_not_found_ex).
        DATA(lv_error) = |Schedule not found: { lo_not_found_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error deleting schedule: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteSchedule referência da API AWS SDK for SAP ABAP.

`DeleteScheduleGroup`

O código de exemplo a seguir mostra como usar `DeleteScheduleGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example iv_name = 'my-schedule-group'
        lo_scd->deleteschedulegroup(
          iv_name = iv_name ).
        MESSAGE 'Schedule group deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdresourcenotfoundex INTO DATA(lo_not_found_ex).
        DATA(lv_error) = |Schedule group not found: { lo_not_found_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error deleting schedule group: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteScheduleGroup referência da API AWS SDK for SAP ABAP.

AWS Glue exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. AWS Glue

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/AWSGlueServiceRole-Test'
        " iv_database_name = 'my-database'
        " iv_table_prefix = 'test_'
        " iv_s3_target = 's3://example-bucket/data/'

        DATA(lt_s3_targets) = VALUE /aws1/cl_glus3target=>tt_s3targetlist(
          ( NEW /aws1/cl_glus3target( iv_path = iv_s3_target ) ) ).

        DATA(lo_targets) = NEW /aws1/cl_glucrawlertargets(
          it_s3targets = lt_s3_targets ).

        lo_glu->createcrawler(
          iv_name = iv_crawler_name
          iv_role = iv_role_arn
          iv_databasename = iv_database_name
          iv_tableprefix = iv_table_prefix
          io_targets = lo_targets ).
        MESSAGE 'Crawler created successfully.' TYPE 'I'.
      CATCH /aws1/cx_glualreadyexistsex.
        MESSAGE 'Crawler already exists.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
```

```
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateCrawler referência da API AWS SDK for SAP ABAP.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        " iv_description = 'ETL job for data transformation'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/AWSGlueServiceRole-Test'
        " iv_script_location = 's3://example-bucket/scripts/my-script.py'

        DATA(lo_command) = NEW /aws1/cl_glujobcommand(
          iv_name = 'glueetl'
          iv_scriptlocation = iv_script_location
          iv_pythonversion = '3' ).

        lo_glu->createjob(
          iv_name = iv_job_name
          iv_description = iv_description
          iv_role = iv_role_arn
          io_command = lo_command
```

```
        iv_glueversion = '3.0' ).
      MESSAGE 'Job created successfully.' TYPE 'I'.
    CATCH /aws1/cx_glualreadyexistsex.
      MESSAGE 'Job already exists.' TYPE 'E'.
    CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
      DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
      MESSAGE lv_invalid_error TYPE 'E'.
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
    CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
      DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
      MESSAGE lv_limit_error TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateJob referência da API AWS SDK for SAP ABAP.

`DeleteCrawler`

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " iv_crawler_name = 'my-crawler'
      lo_glu->deletecrawler( iv_name = iv_crawler_name ).
      MESSAGE 'Crawler deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_glucrawlerrunningex.
      MESSAGE 'Crawler is currently running.' TYPE 'E'.
    CATCH /aws1/cx_gluentitynotfoundex.
      MESSAGE 'Crawler does not exist.' TYPE 'E'.
```

```
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
    CATCH /aws1/cx_gluschdrtransingex.
      MESSAGE 'Scheduler is transitioning.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCrawler referência da API AWS SDK for SAP ABAP.

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " iv_database_name = 'my-database'
      lo_glu->deletedatabase( iv_name = iv_database_name ).
      MESSAGE 'Database deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_gluentitynotfoundex.
      MESSAGE 'Database does not exist.' TYPE 'E'.
    CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
      DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
      MESSAGE lv_invalid_error TYPE 'E'.
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDatabase referência da API AWS SDK for SAP ABAP.

DeleteJob

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_job_name = 'my-etl-job'
    lo_glu->deletejob( iv_jobname = iv_job_name ).
    MESSAGE 'Job deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteJob referência da API AWS SDK for SAP ABAP.

DeleteTable

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_database_name = 'my-database'
        " iv_table_name = 'my-table'
        lo_glu->deletetable(
          iv_databasename = iv_database_name
          iv_name = iv_table_name ).
        MESSAGE 'Table deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Table or database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTable referência da API AWS SDK for SAP ABAP.

`GetCrawler`

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_crawler_name = 'my-crawler'
    oo_result = lo_glu->getcrawler( iv_name = iv_crawler_name ).
    DATA(lo_crawler) = oo_result->get_crawler( ).
    MESSAGE 'Crawler information retrieved.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Crawler does not exist.' TYPE 'I'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetCrawler referência da API AWS SDK for SAP ABAP.

`GetDatabase`

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_database_name = 'my-database'
    oo_result = lo_glu->getdatabase( iv_name = iv_database_name ).
    DATA(lo_database) = oo_result->get_database( ).
```

```
      MESSAGE 'Database information retrieved.' TYPE 'I'.
    CATCH /aws1/cx_gluentitynotfoundex.
      MESSAGE 'Database does not exist.' TYPE 'E'.
    CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
      DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
      MESSAGE lv_invalid_error TYPE 'E'.
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetDatabase referência da API AWS SDK for SAP ABAP.

`GetJobRun`

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " iv_job_name = 'my-etl-job'
      " iv_run_id = 'jr_abcd1234567890abcdef1234567890abcdef12345678'
      oo_result = lo_glu->getjobrun(
        iv_jobname = iv_job_name
        iv_runid = iv_run_id ).
      DATA(lo_job_run) = oo_result->get_jobrun( ).
      MESSAGE 'Job run information retrieved.' TYPE 'I'.
    CATCH /aws1/cx_gluentitynotfoundex.
      MESSAGE 'Job or job run does not exist.' TYPE 'E'.
    CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
```

```
      DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
      MESSAGE lv_invalid_error TYPE 'E'.
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetJobRun referência da API AWS SDK for SAP ABAP.

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " iv_job_name = 'my-etl-job'
      oo_result = lo_glu->getjobruns( iv_jobname = iv_job_name ).
      DATA(lt_job_runs) = oo_result->get_jobruns( ).
      MESSAGE 'Job runs retrieved successfully.' TYPE 'I'.
    CATCH /aws1/cx_gluentitynotfoundex.
      MESSAGE 'Job does not exist.' TYPE 'E'.
    CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
      DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
      MESSAGE lv_invalid_error TYPE 'E'.
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
```

```
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetJobRuns referência da API AWS SDK for SAP ABAP.

`GetTables`

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_database_name = 'my-database'
        oo_result = lo_glu->gettables( iv_databasename = iv_database_name ).
        DATA(lt_tables) = oo_result->get_tablelist( ).
        MESSAGE 'Tables retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetTables referência da API AWS SDK for SAP ABAP.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_glu->listjobs( ).
    DATA(lt_job_names) = oo_result->get_jobnames( ).
    MESSAGE 'Job list retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'No jobs found.' TYPE 'I'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListJobs referência da API AWS SDK for SAP ABAP.

`StartCrawler`

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_crawler_name = 'my-crawler'
    lo_glu->startcrawler( iv_name = iv_crawler_name ).
    MESSAGE 'Crawler started successfully.' TYPE 'I'.
  CATCH /aws1/cx_glucrawlerrunningex.
    MESSAGE 'Crawler is already running.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Crawler does not exist.' TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a StartCrawler referência da API AWS SDK for SAP ABAP.

`StartJobRun`

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_job_name = 'my-etl-job'
```

```
        " iv_input_database = 'my-database'
        " iv_input_table = 'my-table'
        " iv_output_bucket_url = 's3://example-output-bucket/'

        DATA lt_arguments TYPE /aws1/cl_glugenericmap_w=>tt_genericmap.
        lt_arguments = VALUE #(
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--input_database'
            value = NEW /aws1/cl_glugenericmap_w( iv_value = iv_input_database ) ) )
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--input_table'
            value = NEW /aws1/cl_glugenericmap_w( iv_value = iv_input_table ) ) )
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--output_bucket_url'
            value = NEW /aws1/cl_glugenericmap_w( iv_value =
iv_output_bucket_url ) ) ) ).

        DATA(oo_result) = lo_glu->startjobrun(
          iv_jobname = iv_job_name
          it_arguments = lt_arguments ).
        ov_job_run_id = oo_result->get_jobrunid( ).
        MESSAGE 'Job run started successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluconcurrentrunsex00.
        MESSAGE 'Maximum concurrent runs exceeded.' TYPE 'E'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Job does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartJobRun referência da API AWS SDK for SAP ABAP.

HealthImaging exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. HealthImaging

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CopyImageSet`

O código de exemplo a seguir mostra como usar `CopyImageSet`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_source_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_source_version_id = '1'
        " iv_destination_image_set_id =
'12345678901234567890123456789012345678 90' (optional)
        " iv_destination_version_id = '1' (optional)
        " iv_force = abap_false
        DATA(lo_source_info) = NEW /aws1/cl_migcpsrcimagesetinf00(
          iv_latestversionid = iv_source_version_id ).
        DATA(lo_copy_info) = NEW /aws1/cl_migcpimagesetinfmtion(
          io_sourceimageset = lo_source_info ).
        IF iv_destination_image_set_id IS NOT INITIAL AND
           iv_destination_version_id IS NOT INITIAL.
          DATA(lo_dest_info) = NEW /aws1/cl_migcopydstimageset(
            iv_imagesetid = iv_destination_image_set_id
            iv_latestversionid = iv_destination_version_id ).
          lo_copy_info = NEW /aws1/cl_migcpimagesetinfmtion(
```

```
            io_sourceimageset = lo_source_info
            io_destinationimageset = lo_dest_info ).
        ENDIF.
        oo_result = lo_mig->copyimageset(
          iv_datastoreid = iv_datastore_id
          iv_sourceimagesetid = iv_source_image_set_id
          io_copyimagesetinformation = lo_copy_info
          iv_force = iv_force ).
        DATA(lo_dest_props) = oo_result->get_dstimagesetproperties( ).
        DATA(lv_new_id) = lo_dest_props->get_imagesetid( ).
        MESSAGE |Image set copied with new ID: { lv_new_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CopyImageSet referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`CreateDatastore`

O código de exemplo a seguir mostra como usar `CreateDatastore`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_name = 'my-datastore-name'
        oo_result = lo_mig->createdatastore( iv_datastorename = iv_datastore_name ).
        DATA(lv_datastore_id) = oo_result->get_datastoreid( ).
        MESSAGE 'Data store created.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict. Data store may already exist.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDatastore referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteDatastore`

O código de exemplo a seguir mostra como usar `DeleteDatastore`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '1234567890123456789012345678901234567890'
        oo_result = lo_mig->deletedatastore( iv_datastoreid = iv_datastore_id ).
        MESSAGE 'Data store deleted.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
```

```
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict. Data store may contain resources.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Data store not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDatastore referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`DeleteImageSet`

O código de exemplo a seguir mostra como usar `DeleteImageSet`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->deleteimageset(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id ).
        MESSAGE 'Image set deleted.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
```

```
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteImageSet referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDICOMImportJob`

O código de exemplo a seguir mostra como usar `GetDICOMImportJob`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_job_id = '12345678901234567890123456789012'
        oo_result = lo_mig->getdicomimportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id ).
        DATA(lo_job_props) = oo_result->get_jobproperties( ).
        DATA(lv_job_status) = lo_job_props->get_jobstatus( ).
        MESSAGE |Job status: { lv_job_status }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Job not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
```

```
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetDICOMImportJob referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetDatastore`

O código de exemplo a seguir mostra como usar `GetDatastore`.

SDK para SAP ABAP

```
  TRY.
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      oo_result = lo_mig->getdatastore( iv_datastoreid = iv_datastore_id ).
      DATA(lo_properties) = oo_result->get_datastoreproperties( ).
      DATA(lv_name) = lo_properties->get_datastorename( ).
      DATA(lv_status) = lo_properties->get_datastorestatus( ).
      MESSAGE 'Data store properties retrieved.' TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Data store not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetDatastore referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageFrame`

O código de exemplo a seguir mostra como usar `GetImageFrame`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_image_frame_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->getimageframe(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          io_imageframeinformation = NEW /aws1/cl_migimageframeinfmtion(
            iv_imageframeid = iv_image_frame_id ) ).
        DATA(lv_frame_blob) = oo_result->get_imageframeblob( ).
        MESSAGE 'Image frame retrieved.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image frame not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetImageFrame referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSet`

O código de exemplo a seguir mostra como usar `GetImageSet`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_state) = oo_result->get_imagesetstate( ).
        MESSAGE |Image set retrieved with state: { lv_state }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
```

```
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetImageSet referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`GetImageSetMetadata`

O código de exemplo a seguir mostra como usar `GetImageSetMetadata`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimagesetmetadata(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimagesetmetadata(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_metadata_blob) = oo_result->get_imagesetmetadatablob( ).
        MESSAGE 'Image set metadata retrieved.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
```

```
      MESSAGE 'Image set not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetImageSetMetadata referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDICOMImportJobs`

O código de exemplo a seguir mostra como usar `ListDICOMImportJobs`.

SDK para SAP ABAP

```
  TRY.
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      oo_result = lo_mig->listdicomimportjobs( iv_datastoreid = iv_datastore_id ).
      DATA(lt_jobs) = oo_result->get_jobsummaries( ).
      DATA(lv_count) = lines( lt_jobs ).
      MESSAGE |Found { lv_count } DICOM import jobs.| TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Resource not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListDICOMImportJobs referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListDatastores`

O código de exemplo a seguir mostra como usar `ListDatastores`.

SDK para SAP ABAP

```
TRY.
    oo_result = lo_mig->listdatastores( ).
    DATA(lt_datastores) = oo_result->get_datastoresummaries( ).
    DATA(lv_count) = lines( lt_datastores ).
    MESSAGE |Found { lv_count } data stores.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListDatastores referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListImageSetVersions`

O código de exemplo a seguir mostra como usar `ListImageSetVersions`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->listimagesetversions(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id ).
        DATA(lt_versions) = oo_result->get_imagesetpropertieslist( ).
        DATA(lv_count) = lines( lt_versions ).
        MESSAGE |Found { lv_count } image set versions.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListImageSetVersions referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`ListTagsForResource`

O código de exemplo a seguir mostra como usar `ListTagsForResource`.

SDK para SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        oo_result = lo_mig->listtagsforresource( iv_resourcearn = iv_resource_arn ).
        DATA(lt_tags) = oo_result->get_tags( ).
        DATA(lv_count) = lines( lt_tags ).
        MESSAGE |Found { lv_count } tags for resource.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTagsForResource referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`SearchImageSets`

O código de exemplo a seguir mostra como usar `SearchImageSets`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->searchimagesets(
          iv_datastoreid = iv_datastore_id
          io_searchcriteria = io_search_criteria ).
```

```
      DATA(lt_imagesets) = oo_result->get_imagesetsmetadatasums( ).
      DATA(lv_count) = lines( lt_imagesets ).
      MESSAGE |Found { lv_count } image sets.| TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Resource not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a SearchImageSets referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`StartDICOMImportJob`

O código de exemplo a seguir mostra como usar `StartDICOMImportJob`.

SDK para SAP ABAP

```
  TRY.
      " iv_job_name = 'import-job-1'
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      " iv_role_arn = 'arn:aws:iam::123456789012:role/ImportJobRole'
      " iv_input_s3_uri = 's3://my-bucket/input/'
      " iv_output_s3_uri = 's3://my-bucket/output/'
      oo_result = lo_mig->startdicomimportjob(
        iv_jobname = iv_job_name
        iv_datastoreid = iv_datastore_id
```

```
          iv_dataaccessrolearn = iv_role_arn
          iv_inputs3uri = iv_input_s3_uri
          iv_outputs3uri = iv_output_s3_uri ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |DICOM import job started with ID: { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartDICOMImportJob referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->tagresource(
          iv_resourcearn = iv_resource_arn
```

```
          it_tags = it_tags ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a TagResource referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`UntagResource`

O código de exemplo a seguir mostra como usar `UntagResource`.

SDK para SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
```

```
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UntagResource referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

`UpdateImageSetMetadata`

O código de exemplo a seguir mostra como usar `UpdateImageSetMetadata`.

SDK para SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_latest_version_id = '1'
        " iv_force = abap_false
        oo_result = lo_mig->updateimagesetmetadata(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          iv_latestversionid = iv_latest_version_id
          io_updateimagesetmetupdates = io_metadata_updates
          iv_force = iv_force ).
        DATA(lv_new_version) = oo_result->get_latestversionid( ).
        MESSAGE |Image set metadata updated to version: { lv_new_version }.| TYPE
'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
```

```
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateImageSetMetadata referência da API AWS SDK for SAP ABAP.

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

HealthLake exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. HealthLake

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

Crie um repositório de dados FHIR

O código de exemplo a seguir mostra como usar `CreateFHIRDatastore`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_name = 'MyHealthLakeDataStore'
        oo_result = lo_hll->createfhirdatastore(
          iv_datastorename = iv_datastore_name
          iv_datastoretypeversion = 'R4'
        ).
        MESSAGE 'Data store created successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllinternalserverex INTO DATA(lo_internal_ex).
        lv_error = |Internal server error: { lo_internal_ex->av_err_code }-
{ lo_internal_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte CreateFhirDatastore no AWS SDK para referência da API SAP ABAP.

Excluir armazenamento de dados FHIR

O código de exemplo a seguir mostra como usar `DeleteFHIRDatastore`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->deletefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        MESSAGE 'Data store deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        DATA(lv_error) = |Access denied: { lo_access_ex->av_err_code }-
{ lo_access_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
      CATCH /aws1/cx_hllconflictexception INTO DATA(lo_conflict_ex).
        lv_error = |Conflict error: { lo_conflict_ex->av_err_code }-
{ lo_conflict_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a referência da API DeleteFhirDatastore no AWS SDK para SAP ABAP.

Descreva o armazenamento de dados FHIR

O código de exemplo a seguir mostra como usar `DescribeFHIRDatastore`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lo_datastore_properties) = oo_result->get_datastoreproperties( ).
        IF lo_datastore_properties IS BOUND.
          DATA(lv_datastore_name) = lo_datastore_properties->get_datastorename( ).
          DATA(lv_datastore_status) = lo_datastore_properties-
>get_datastorestatus( ).
          MESSAGE 'Data store described successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte DescribeFhirDatastore no AWS SDK para referência da API SAP ABAP.

`DescribeFHIRExportJob`

O código de exemplo a seguir mostra como usar `DescribeFHIRExportJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirexportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_export_job_properties) = oo_result->get_exportjobproperties( ).
        IF lo_export_job_properties IS BOUND.
          DATA(lv_job_status) = lo_export_job_properties->get_jobstatus( ).
          MESSAGE |Export job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeFHIRExportJob referência da API AWS SDK for SAP ABAP.

`DescribeFHIRImportJob`

O código de exemplo a seguir mostra como usar `DescribeFHIRImportJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirimportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_import_job_properties) = oo_result->get_importjobproperties( ).
        IF lo_import_job_properties IS BOUND.
          DATA(lv_job_status) = lo_import_job_properties->get_jobstatus( ).
          MESSAGE |Import job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeFHIRImportJob referência da API AWS SDK for SAP ABAP.

Listar armazenamentos de dados FHIR

O código de exemplo a seguir mostra como usar `ListFHIRDatastores`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_hll->listfhirdatastores( ).
        DATA(lt_datastores) = oo_result->get_datastorepropertieslist( ).
        DATA(lv_datastore_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_datastore_count } data store(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte ListFhirDatastores no AWS SDK para referência da API SAP ABAP.

ListFHIRExportJobs

O código de exemplo a seguir mostra como usar `ListFHIRExportJobs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_export_jobs) = oo_result->get_exportjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_export_jobs ).
        MESSAGE |Found { lv_job_count } export job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListFHIRExportJobs referência da API AWS SDK for SAP ABAP.

`ListFHIRImportJobs`

O código de exemplo a seguir mostra como usar `ListFHIRImportJobs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_import_jobs) = oo_result->get_importjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_import_jobs ).
        MESSAGE |Found { lv_job_count } import job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListFHIRImportJobs referência da API AWS SDK for SAP ABAP.

`ListTagsForResource`

O código de exemplo a seguir mostra como usar `ListTagsForResource`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        DATA(lo_result) = lo_hll->listtagsforresource(
          iv_resourcearn = iv_resource_arn
        ).
        ot_tags = lo_result->get_tags( ).
        DATA(lv_tag_count) = lines( ot_tags ).
        MESSAGE |Found { lv_tag_count } tag(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTagsForResource referência da API AWS SDK for SAP ABAP.

`StartFHIRExportJob`

O código de exemplo a seguir mostra como usar `StartFHIRExportJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_job_name = 'MyExportJob'
        " iv_output_s3_uri = 's3://my-bucket/export/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeExportRole'
        oo_result = lo_hll->startfhirexportjob(
          iv_jobname = iv_job_name
          io_outputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Export job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartFHIRExportJob referência da API AWS SDK for SAP ABAP.

StartFHIRImportJob

O código de exemplo a seguir mostra como usar `StartFHIRImportJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_job_name = 'MyImportJob'
        " iv_input_s3_uri = 's3://my-bucket/import/data.ndjson'
        " iv_job_output_s3_uri = 's3://my-bucket/import/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeImportRole'
        oo_result = lo_hll->startfhirimportjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_hllinputdataconfig( iv_s3uri =
 iv_input_s3_uri )
          io_joboutputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
```

```
        MESSAGE |Import job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartFHIRImportJob referência da API AWS SDK for SAP ABAP.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags
        ).
```

```
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a TagResource referência da API AWS SDK for SAP ABAP.

`UntagResource`

O código de exemplo a seguir mostra como usar `UntagResource`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UntagResource referência da API AWS SDK for SAP ABAP.

Exemplos de IAM usando SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com IAM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->attachrolepolicy(
          iv_rolename = iv_role_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy attached to role successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role or policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AttachRolePolicy referência da API AWS SDK for SAP ABAP.

`AttachUserPolicy`

O código de exemplo a seguir mostra como usar `AttachUserPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->attachuserpolicy(
          iv_username = iv_user_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy attached to user successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User or policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AttachUserPolicy referência da API AWS SDK for SAP ABAP.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->createaccesskey(
      iv_username = iv_user_name ).
    MESSAGE 'Access key created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Maximum number of access keys reached.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAccessKey referência da API AWS SDK for SAP ABAP.

`CreateAccountAlias`

O código de exemplo a seguir mostra como usar `CreateAccountAlias`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_iam->createaccountalias(
          iv_accountalias = iv_account_alias ).
        MESSAGE 'Account alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Account alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Account alias limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAccountAlias referência da API AWS SDK for SAP ABAP.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->createpolicy(
          iv_policyname = iv_policy_name
          iv_policydocument = iv_policy_document
          iv_description = iv_description ).
        MESSAGE 'Policy created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Policy already exists.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreatePolicy referência da API AWS SDK for SAP ABAP.

`CreatePolicyVersion`

O código de exemplo a seguir mostra como usar `CreatePolicyVersion`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->createpolicyversion(
      iv_policyarn = iv_policy_arn
      iv_policydocument = iv_policy_document
      iv_setasdefault = iv_set_as_default ).
    MESSAGE 'Policy version created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iammalformedplydocex.
    MESSAGE 'Policy document is malformed.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy version limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreatePolicyVersion referência da API AWS SDK for SAP ABAP.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->createrole(
      iv_rolename = iv_role_name
      iv_assumerolepolicydocument = iv_assume_role_policy_document ).
    MESSAGE 'Role created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamentityalrdyexex.
    MESSAGE 'Role already exists.' TYPE 'E'.
  CATCH /aws1/cx_iammalformedplydocex.
    MESSAGE 'Assume role policy document is malformed.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Role limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateRole referência da API AWS SDK for SAP ABAP.

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->listpolicyversions(
```

```
        iv_policyarn = iv_policy_arn ).
      MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Policy does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iamservicefailureex.
      MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateServiceLinkedRole referência da API AWS SDK for SAP ABAP.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_iam->createuser(
        iv_username = iv_user_name ).
      MESSAGE 'User created successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamentityalrdyexex.
      MESSAGE 'User already exists.' TYPE 'E'.
    CATCH /aws1/cx_iamlimitexceededex.
      MESSAGE 'Limit exceeded for IAM users.' TYPE 'E'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Entity does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateUser referência da API AWS SDK for SAP ABAP.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->deleteaccesskey(
      iv_accesskeyid = iv_access_key_id
      iv_username = iv_user_name ).
    MESSAGE 'Access key deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Access key or user does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAccessKey referência da API AWS SDK for SAP ABAP.

`DeleteAccountAlias`

O código de exemplo a seguir mostra como usar `DeleteAccountAlias`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->deleteaccountalias(
```

```
        iv_accountalias = iv_account_alias ).
      MESSAGE 'Account alias deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Account alias does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAccountAlias referência da API AWS SDK for SAP ABAP.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_iam->deletepolicy( iv_policyarn = iv_policy_arn ).
      MESSAGE 'Policy deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Policy does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iamdeleteconflictex.
      MESSAGE 'Policy cannot be deleted due to attachments.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeletePolicy referência da API AWS SDK for SAP ABAP.

`DeletePolicyVersion`

O código de exemplo a seguir mostra como usar `DeletePolicyVersion`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->deletepolicyversion(
      iv_policyarn = iv_policy_arn
      iv_versionid = iv_version_id ).
    MESSAGE 'Policy version deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy or version does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamdeleteconflictex.
    MESSAGE 'Cannot delete default policy version.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeletePolicyVersion referência da API AWS SDK for SAP ABAP.

`DeleteRole`

O código de exemplo a seguir mostra como usar `DeleteRole`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->deleterole( iv_rolename = iv_role_name ).
```

```
      MESSAGE 'Role deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Role does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iamdeleteconflictex.
      MESSAGE 'Role cannot be deleted due to attached resources.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteRole referência da API AWS SDK for SAP ABAP.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_iam->deleteuser( iv_username = iv_user_name ).
      MESSAGE 'User deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'User does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iamdeleteconflictex.
      MESSAGE 'User cannot be deleted due to attached resources.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteUser referência da API AWS SDK for SAP ABAP.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->detachrolepolicy(
      iv_rolename = iv_role_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy detached from role successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role or policy does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetachRolePolicy referência da API AWS SDK for SAP ABAP.

`DetachUserPolicy`

O código de exemplo a seguir mostra como usar `DetachUserPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->detachuserpolicy(
      iv_username = iv_user_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy detached from user successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
```

```
    MESSAGE 'User or policy does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetachUserPolicy referência da API AWS SDK for SAP ABAP.

GenerateCredentialReport

O código de exemplo a seguir mostra como usar `GenerateCredentialReport`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->generatecredentialreport( ).
    MESSAGE 'Credential report generation started.' TYPE 'I'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Report generation limit exceeded.' TYPE 'E'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when generating credential report.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GenerateCredentialReport referência da API AWS SDK for SAP ABAP.

GetAccessKeyLastUsed

O código de exemplo a seguir mostra como usar `GetAccessKeyLastUsed`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->getaccesskeylastused(
          iv_accesskeyid = iv_access_key_id ).
        MESSAGE 'Retrieved access key last used information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Access key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetAccessKeyLastUsed referência da API AWS SDK for SAP ABAP.

`GetAccountAuthorizationDetails`

O código de exemplo a seguir mostra como usar `GetAccountAuthorizationDetails`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->getaccountauthdetails( ).
        MESSAGE 'Retrieved account authorization details.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting account authorization details.' TYPE
'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetAccountAuthorizationDetails referência da API AWS SDK for SAP ABAP.

`GetAccountPasswordPolicy`

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->getaccountpasswordpolicy( ).
    MESSAGE 'Retrieved account password policy.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'No password policy exists.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when getting password policy.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetAccountPasswordPolicy referência da API AWS SDK for SAP ABAP.

`GetAccountSummary`

O código de exemplo a seguir mostra como usar `GetAccountSummary`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->getaccountsummary( ).
        MESSAGE 'Retrieved account summary.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting account summary.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetAccountSummary referência da API AWS SDK for SAP ABAP.

`GetCredentialReport`

O código de exemplo a seguir mostra como usar `GetCredentialReport`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->getcredentialreport( ).
        MESSAGE 'Retrieved credential report.' TYPE 'I'.
      CATCH /aws1/cx_iamcredrptnotpresen00.
        MESSAGE 'Credential report not present.' TYPE 'E'.
      CATCH /aws1/cx_iamcredrptexpiredex.
        MESSAGE 'Credential report expired.' TYPE 'E'.
      CATCH /aws1/cx_iamcredrptnotreadyex.
        MESSAGE 'Credential report not ready.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting credential report.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetCredentialReport referência da API AWS SDK for SAP ABAP.

`GetPolicy`

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->getpolicy( iv_policyarn = iv_policy_arn ).
    MESSAGE 'Retrieved policy information.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetPolicy referência da API AWS SDK for SAP ABAP.

`GetPolicyVersion`

O código de exemplo a seguir mostra como usar `GetPolicyVersion`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->getpolicyversion(
      iv_policyarn = iv_policy_arn
      iv_versionid = iv_version_id ).
    MESSAGE 'Retrieved policy version information.' TYPE 'I'.
```

```
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Policy or version does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iaminvalidinputex.
      MESSAGE 'Invalid input provided.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetPolicyVersion referência da API AWS SDK for SAP ABAP.

`GetRole`

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_iam->getrole( iv_rolename = iv_role_name ).
      MESSAGE 'Retrieved role information.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Role does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetRole referência da API AWS SDK for SAP ABAP.

`ListAccessKeys`

O código de exemplo a seguir mostra como usar `ListAccessKeys`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->listaccesskeys(
      iv_username = iv_user_name ).
    MESSAGE 'Retrieved access key list.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListAccessKeys referência da API AWS SDK for SAP ABAP.

`ListAccountAliases`

O código de exemplo a seguir mostra como usar `ListAccountAliases`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->listaccountaliases( ).
    MESSAGE 'Retrieved account alias list.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing account aliases.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListAccountAliases referência da API AWS SDK for SAP ABAP.

`ListAttachedRolePolicies`

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->listattachedrolepolicies(
      iv_rolename = iv_role_name ).
    MESSAGE 'Retrieved attached policy list for role.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListAttachedRolePolicies referência da API AWS SDK for SAP ABAP.

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listgroups( ).
        MESSAGE 'Retrieved group list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing groups.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListGroups referência da API AWS SDK for SAP ABAP.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listpolicies( iv_scope = iv_scope ).
        MESSAGE 'Retrieved policy list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policies.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListPolicies referência da API AWS SDK for SAP ABAP.

`ListPolicyVersions`

O código de exemplo a seguir mostra como usar `ListPolicyVersions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->listpolicyversions(
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListPolicyVersions referência da API AWS SDK for SAP ABAP.

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_iam->listrolepolicies(
      iv_rolename = iv_role_name ).
    MESSAGE 'Retrieved inline policy list for role.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
```

```
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListRolePolicies referência da API AWS SDK for SAP ABAP.

`ListRoles`

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listroles( ).
        MESSAGE 'Retrieved role list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing roles.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListRoles referência da API AWS SDK for SAP ABAP.

`Listar provedores de SAML`

O código de exemplo a seguir mostra como usar `ListSAMLProviders`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listsamlproviders( ).
        MESSAGE 'Retrieved SAML provider list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing SAML providers.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte ListSAMLProviders na Referência de APIs do AWS SDK para SAP ABAP.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_iam->listusers( ).
        MESSAGE 'Retrieved user list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing users.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListUsers referência da API AWS SDK for SAP ABAP.

`SetDefaultPolicyVersion`

O código de exemplo a seguir mostra como usar `SetDefaultPolicyVersion`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->setdefaultpolicyversion(
      iv_policyarn = iv_policy_arn
      iv_versionid = iv_version_id ).
    MESSAGE 'Default policy version set successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy or version does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iaminvalidinputex.
    MESSAGE 'Invalid input provided.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a SetDefaultPolicyVersion referência da API AWS SDK for SAP ABAP.

`UpdateAccessKey`

O código de exemplo a seguir mostra como usar `UpdateAccessKey`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_iam->updateaccesskey(
```

```
        iv_accesskeyid = iv_access_key_id
        iv_status = iv_status
        iv_username = iv_user_name ).
      MESSAGE 'Access key updated successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Access key or user does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateAccessKey referência da API AWS SDK for SAP ABAP.

`UpdateUser`

O código de exemplo a seguir mostra como usar `UpdateUser`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_iam->updateuser(
        iv_username = iv_user_name
        iv_newusername = iv_new_user_name ).
      MESSAGE 'User updated successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'User does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iamentityalrdyexex.
      MESSAGE 'New user name already exists.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateUser referência da API AWS SDK for SAP ABAP.

AWS IoT exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. AWS IoT

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AttachThingPrincipal`

O código de exemplo a seguir mostra como usar `AttachThingPrincipal`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        lo_iot->attachthingprincipal(
          iv_thingname = iv_thing_name
          iv_principal = iv_principal ).
        MESSAGE |Principal attached to IoT thing '{ iv_thing_name }'.| TYPE 'I'.
      CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
        MESSAGE |Resource not found when attaching principal to
'{ iv_thing_name }'.| TYPE 'I'.
```

```
      RAISE EXCEPTION lo_ex.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
      MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_svc_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a AttachThingPrincipal referência da API AWS SDK for SAP ABAP.

`CreateKeysAndCertificate`

O código de exemplo a seguir mostra como usar `CreateKeysAndCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
  DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
  DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
  TRY.
      oo_result = lo_iot->createkeysandcertificate( iv_setasactive = abap_true ).
      MESSAGE |Certificate created: { oo_result->get_certificateid( ) }| TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
      MESSAGE lo_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateKeysAndCertificate referência da API AWS SDK for SAP ABAP.

`CreateThing`

O código de exemplo a seguir mostra como usar `CreateThing`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       oo_result = lo_iot->creatething(
         iv_thingname = iv_thing_name ).
       MESSAGE |IoT thing created: { oo_result->get_thingname( ) } ARN:
{ oo_result->get_thingarn( ) }| TYPE 'I'.
     CATCH /aws1/cx_iotresrcalrdyexistsex.
       MESSAGE |IoT thing '{ iv_thing_name }' already exists.| TYPE 'I'.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateThing referência da API AWS SDK for SAP ABAP.

`CreateTopicRule`

O código de exemplo a seguir mostra como usar `CreateTopicRule`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
```

```
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        " Build the SNS action that will receive messages matching the rule.
        DATA lo_sns_action TYPE REF TO /aws1/cl_iotsnsaction.
        CREATE OBJECT lo_sns_action
          EXPORTING
            iv_targetarn = iv_sns_action_arn
            iv_rolearn   = iv_role_arn.

        DATA lo_action TYPE REF TO /aws1/cl_iotaction.
        CREATE OBJECT lo_action
          EXPORTING
            io_sns = lo_sns_action.

        DATA lt_actions TYPE /aws1/cl_iotaction=>tt_actionlist.
        APPEND lo_action TO lt_actions.

        " iv_topic = 'my/iot/topic' - The MQTT topic pattern to match
        DATA lo_payload TYPE REF TO /aws1/cl_iottopicrulepayload.
        CREATE OBJECT lo_payload
          EXPORTING
            iv_sql     = |SELECT * FROM '{ iv_topic }'|
            it_actions = lt_actions.

        lo_iot->createtopicrule(
          iv_rulename         = iv_rule_name
          io_topicrulepayload = lo_payload ).
        MESSAGE |IoT topic rule created: { iv_rule_name }| TYPE 'I'.
      CATCH /aws1/cx_iotresrcalrdyexistsex.
        MESSAGE |Topic rule '{ iv_rule_name }' already exists.| TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTopicRule referência da API AWS SDK for SAP ABAP.

`DeleteCertificate`

O código de exemplo a seguir mostra como usar `DeleteCertificate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Certificates must be deactivated before they can be deleted.
    lo_iot->updatecertificate(
      iv_certificateid = iv_certificate_id
      iv_newstatus     = 'INACTIVE' ).
    lo_iot->deletecertificate( iv_certificateid = iv_certificate_id ).
    MESSAGE |Certificate deleted: { iv_certificate_id }| TYPE 'I'.
  CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE |Certificate '{ iv_certificate_id }' not found.| TYPE 'I'.
    RAISE EXCEPTION lo_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
    MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_svc_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCertificate referência da API AWS SDK for SAP ABAP.

`DeleteThing`

O código de exemplo a seguir mostra como usar `DeleteThing`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    lo_iot->deletething( iv_thingname = iv_thing_name ).
    MESSAGE |IoT thing deleted: { iv_thing_name }| TYPE 'I'.
  CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE |IoT thing '{ iv_thing_name }' not found.| TYPE 'I'.
    RAISE EXCEPTION lo_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
    MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_svc_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteThing referência da API AWS SDK for SAP ABAP.

`DeleteTopicRule`

O código de exemplo a seguir mostra como usar `DeleteTopicRule`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    lo_iot->deletetopicrule( iv_rulename = iv_rule_name ).
    MESSAGE |IoT topic rule deleted: { iv_rule_name }| TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTopicRule referência da API AWS SDK for SAP ABAP.

`DescribeEndpoint`

O código de exemplo a seguir mostra como usar `DescribeEndpoint`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   " iv_endpoint_type = 'iot:Data-ATS' - Endpoint type for data operations
   TRY.
       DATA(lo_result) = lo_iot->describeendpoint( iv_endpointtype =
iv_endpoint_type ).
       ov_endpoint_address = lo_result->get_endpointaddress( ).
       MESSAGE |Endpoint address: { ov_endpoint_address }| TYPE 'I'.
     CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
       MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
       RAISE EXCEPTION lo_throttle_ex.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeEndpoint referência da API AWS SDK for SAP ABAP.

`DetachThingPrincipal`

O código de exemplo a seguir mostra como usar `DetachThingPrincipal`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        lo_iot->detachthingprincipal(
          iv_thingname = iv_thing_name
          iv_principal = iv_principal ).
        MESSAGE |Principal detached from IoT thing '{ iv_thing_name }'.| TYPE 'I'.
      CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
        MESSAGE |Resource not found when detaching principal from
'{ iv_thing_name }'.| TYPE 'I'.
        RAISE EXCEPTION lo_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
        MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_svc_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetachThingPrincipal referência da API AWS SDK for SAP ABAP.

`ListCertificates`

O código de exemplo a seguir mostra como usar `ListCertificates`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Collect all certificates by following the pagination marker.
    DATA lv_marker TYPE /aws1/iotmarker.
    DATA lv_count  TYPE i.

    DO.
      oo_result = lo_iot->listcertificates( iv_marker = lv_marker ).
      lv_count  = lv_count + lines( oo_result->get_certificates( ) ).
      lv_marker = oo_result->get_nextmarker( ).
      IF lv_marker IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    MESSAGE |Retrieved { lv_count } IoT certificates.| TYPE 'I'.
  CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
    MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
    RAISE EXCEPTION lo_throttle_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListCertificates referência da API AWS SDK for SAP ABAP.

`ListThings`

O código de exemplo a seguir mostra como usar `ListThings`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Collect all things by following the pagination token.
    DATA lv_nexttoken TYPE /aws1/iotnexttoken.
    DATA lv_count     TYPE i.

    DO.
      oo_result    = lo_iot->listthings( iv_nexttoken = lv_nexttoken ).
      lv_count     = lv_count + lines( oo_result->get_things( ) ).
      lv_nexttoken = oo_result->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    MESSAGE |Retrieved { lv_count } IoT things.| TYPE 'I'.
  CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
    MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
    RAISE EXCEPTION lo_throttle_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListThings referência da API AWS SDK for SAP ABAP.

`SearchIndex`

O código de exemplo a seguir mostra como usar `SearchIndex`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   " iv_query_string = 'thingName:MyThing*' - Fleet indexing query string
   TRY.
       oo_result = lo_iot->searchindex( iv_querystring = iv_query_string ).
       MESSAGE |Found { lines( oo_result->get_things( ) ) } IoT things matching
query.| TYPE 'I'.
     CATCH /aws1/cx_iotindexnotreadyex INTO DATA(lo_idx_ex).
       MESSAGE 'Fleet indexing is not ready. Enable indexing with
UpdateIndexingConfiguration first.' TYPE 'I'.
       RAISE EXCEPTION lo_idx_ex.
     CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
       MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
       RAISE EXCEPTION lo_throttle_ex.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a SearchIndex referência da API AWS SDK for SAP ABAP.

`UpdateIndexingConfiguration`

O código de exemplo a seguir mostra como usar `UpdateIndexingConfiguration`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       DATA lo_idx_conf TYPE REF TO /aws1/cl_iotthingindexingconf.
       CREATE OBJECT lo_idx_conf
         EXPORTING
           iv_thingindexingmode = 'REGISTRY'.
       lo_iot->updateindexingconfiguration(
         io_thingindexingconf = lo_idx_conf ).
       MESSAGE 'IoT thing indexing configuration updated to REGISTRY mode.' TYPE
'I'.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateIndexingConfiguration referência da API AWS SDK for SAP ABAP.

AWS IoT data exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. AWS IoT data

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`GetThingShadow`

O código de exemplo a seguir mostra como usar `GetThingShadow`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_iop->getthingshadow( iv_thingname = iv_thing_name ).

    " Convert xstring payload to JSON string
    DATA(lv_payload) = lo_result->get_payload( ).
    ov_shadow = /aws1/cl_rt_util=>xstring_to_string( lv_payload ).
    MESSAGE 'Thing shadow retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_iopresourcenotfoundex.
    MESSAGE 'Thing shadow not found.' TYPE 'E'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_exception).
    DATA(lv_error) = |{ lo_exception->get_text( ) }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetThingShadow referência da API AWS SDK for SAP ABAP.

`UpdateThingShadow`

O código de exemplo a seguir mostra como usar `UpdateThingShadow`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Convert JSON string to xstring for payload
    DATA(lv_payload) = /aws1/cl_rt_util=>string_to_xstring( iv_shadow_state ).

    lo_iop->updatethingshadow(
      iv_thingname = iv_thing_name
      iv_payload = lv_payload ).
    MESSAGE 'Thing shadow updated successfully.' TYPE 'I'.
  CATCH /aws1/cx_iopresourcenotfoundex.
    MESSAGE 'Thing not found.' TYPE 'E'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_exception).
    DATA(lv_error) = |{ lo_exception->get_text( ) }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateThingShadow referência da API AWS SDK for SAP ABAP.

AWS IoT SiteWise exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. AWS IoT SiteWise

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS IoT SiteWise

O exemplo de código a seguir mostra como começar a usar o AWS IoT SiteWise.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_ios->listassetmodels( ). " oo_result is returned for testing
purposes. "
       DATA(lt_asset_models) = oo_result->get_assetmodelsummaries( ).
       MESSAGE 'Retrieved list of asset models.' TYPE 'I'.
     CATCH /aws1/cx_rt_generic.
       MESSAGE 'Unable to list asset models.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListAssetModels referência da API AWS SDK for SAP ABAP.

Ações

`BatchPutAssetPropertyValue`

O código de exemplo a seguir mostra como usar `BatchPutAssetPropertyValue`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ios->batchputassetpropertyvalue(
      it_entries = it_entries
    ).
    MESSAGE 'Data sent to IoT SiteWise asset successfully.' TYPE 'I'.
  CATCH /aws1/cx_iosresourcenotfoundex.
    MESSAGE 'Asset does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a BatchPutAssetPropertyValue referência da API AWS SDK for SAP ABAP.

`CreateAsset`

O código de exemplo a seguir mostra como usar `CreateAsset`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ios->createasset(
      iv_assetname = iv_asset_name
      iv_assetmodelid = iv_asset_model_id
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'IoT SiteWise asset created' TYPE 'I'.
```

```
    CATCH /aws1/cx_iosresourcenotfoundex.
      MESSAGE 'Asset model does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAsset referência da API AWS SDK for SAP ABAP.

`CreateAssetModel`

O código de exemplo a seguir mostra como usar `CreateAssetModel`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_ios->createassetmodel(
        iv_assetmodelname = iv_asset_model_name
        iv_assetmodeldescription = 'This is a sample asset model description.'
        it_assetmodelproperties = it_properties
      ). " oo_result is returned for testing purposes. "
      MESSAGE 'IoT SiteWise asset model created' TYPE 'I'.
    CATCH /aws1/cx_iosresrcalrdyexistsex.
      MESSAGE 'Asset model already exists.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAssetModel referência da API AWS SDK for SAP ABAP.

`CreateGateway`

O código de exemplo a seguir mostra como usar `CreateGateway`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ios->creategateway(
      iv_gatewayname = iv_gateway_name
      io_gatewayplatform = NEW /aws1/cl_iosgatewayplatform(
        io_greengrassv2 = NEW /aws1/cl_iosgreengrassv2(
          iv_coredevicethingname = iv_core_device_thing_name
        )
      )
      it_tags = VALUE /aws1/cl_iostagmap_w=>tt_tagmap(
        (
          VALUE /aws1/cl_iostagmap_w=>ts_tagmap_maprow(
            key = 'Environment'
            value = NEW /aws1/cl_iostagmap_w( 'Production' )
          )
        )
      )
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'IoT SiteWise gateway created' TYPE 'I'.
  CATCH /aws1/cx_iosresrcalrdyexistsex.
    MESSAGE 'Gateway already exists.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateGateway referência da API AWS SDK for SAP ABAP.

`DeleteAsset`

O código de exemplo a seguir mostra como usar `DeleteAsset`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ios->deleteasset(
      iv_assetid = iv_asset_id
    ).
    MESSAGE 'IoT SiteWise asset deleted.' TYPE 'I'.
  CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to delete asset.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAsset referência da API AWS SDK for SAP ABAP.

`DeleteAssetModel`

O código de exemplo a seguir mostra como usar `DeleteAssetModel`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ios->deleteassetmodel(
      iv_assetmodelid = iv_asset_model_id
    ).
    MESSAGE 'IoT SiteWise asset model deleted.' TYPE 'I'.
  CATCH /aws1/cx_rt_generic.
```

```
        MESSAGE 'Unable to delete asset model.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAssetModel referência da API AWS SDK for SAP ABAP.

DeleteGateway

O código de exemplo a seguir mostra como usar `DeleteGateway`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ios->deletegateway(
          iv_gatewayid = iv_gateway_id
        ).
        MESSAGE 'IoT SiteWise gateway deleted.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Gateway does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteGateway referência da API AWS SDK for SAP ABAP.

DescribeGateway

O código de exemplo a seguir mostra como usar `DescribeGateway`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ios->describegateway(
      iv_gatewayid = iv_gateway_id
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'Retrieved gateway description.' TYPE 'I'.
  CATCH /aws1/cx_iosresourcenotfoundex.
    MESSAGE 'Gateway does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeGateway referência da API AWS SDK for SAP ABAP.

`GetAssetPropertyValue`

O código de exemplo a seguir mostra como usar `GetAssetPropertyValue`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ios->getassetpropertyvalue(
      iv_assetid = iv_asset_id
      iv_propertyid = iv_property_id
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'Retrieved asset property value.' TYPE 'I'.
```

```
    CATCH /aws1/cx_iosresourcenotfoundex.
      MESSAGE 'Asset or property does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetAssetPropertyValue referência da API AWS SDK for SAP ABAP.

ListAssetModelProperties

O código de exemplo a seguir mostra como usar `ListAssetModelProperties`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_ios->listassetmodelproperties(
        iv_assetmodelid = iv_asset_model_id
      ). " oo_result is returned for testing purposes. "
      DATA(lt_properties) = oo_result->get_assetmodelpropertysums( ).
      MESSAGE 'Retrieved list of asset model properties.' TYPE 'I'.
    CATCH /aws1/cx_rt_generic.
      MESSAGE 'Unable to list asset model properties.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListAssetModelProperties referência da API AWS SDK for SAP ABAP.

ListAssetModels

O código de exemplo a seguir mostra como usar `ListAssetModels`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_ios->listassetmodels( ). " oo_result is returned for testing
purposes. "
       DATA(lt_asset_models) = oo_result->get_assetmodelsummaries( ).
       MESSAGE 'Retrieved list of asset models.' TYPE 'I'.
     CATCH /aws1/cx_rt_generic.
       MESSAGE 'Unable to list asset models.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListAssetModels referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Keyspaces usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Keyspaces.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateKeyspace`

O código de exemplo a seguir mostra como usar `CreateKeyspace`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->createkeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysconflictexception.
        MESSAGE 'Keyspace already exists.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateKeyspace referência da API AWS SDK for SAP ABAP.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Define schema with columns
        DATA(lt_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'title' iv_type = 'text' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'year' iv_type = 'int' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'release_date' iv_type =
 'timestamp' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'plot' iv_type = 'text' ) )
        ).

        " Define partition keys
        DATA(lt_partition_keys) = VALUE /aws1/
cl_kyspartitionkey=>tt_partitionkeylist(
          ( NEW /aws1/cl_kyspartitionkey( iv_name = 'year' ) )
          ( NEW /aws1/cl_kyspartitionkey( iv_name = 'title' ) )
        ).

        " Create schema definition
        DATA(lo_schema) = NEW /aws1/cl_kysschemadefinition(
          it_allcolumns = lt_columns
          it_partitionkeys = lt_partition_keys ).

        " Enable point-in-time recovery
        DATA(lo_pitr) = NEW /aws1/cl_kyspointintimerec(
          iv_status = 'ENABLED' ).

        oo_result = lo_kys->createtable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name
          io_schemadefinition = lo_schema
          io_pointintimerecovery = lo_pitr ).
        MESSAGE 'Table created successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTable referência da API AWS SDK for SAP ABAP.

`DeleteKeyspace`

O código de exemplo a seguir mostra como usar `DeleteKeyspace`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_kys->deletekeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteKeyspace referência da API AWS SDK for SAP ABAP.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_kys->deletetable(
```

```
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name ).
        MESSAGE 'Table deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTable referência da API AWS SDK for SAP ABAP.

`GetKeyspace`

O código de exemplo a seguir mostra como usar `GetKeyspace`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->getkeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysresourcenotfoundex.
        MESSAGE 'Keyspace does not exist.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetKeyspace referência da API AWS SDK for SAP ABAP.

`GetTable`

O código de exemplo a seguir mostra como usar `GetTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->gettable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name ).
        MESSAGE 'Table information retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysresourcenotfoundex.
        MESSAGE 'Table does not exist.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetTable referência da API AWS SDK for SAP ABAP.

`ListKeyspaces`

O código de exemplo a seguir mostra como usar `ListKeyspaces`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->listkeyspaces(
          iv_maxresults = iv_max_results ).
        MESSAGE 'Keyspaces listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListKeyspaces referência da API AWS SDK for SAP ABAP.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->listtables(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Tables listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTables referência da API AWS SDK for SAP ABAP.

`RestoreTable`

O código de exemplo a seguir mostra como usar `RestoreTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kys->restoretable(
          iv_sourcekeyspacename = iv_source_keyspace_name
          iv_sourcetablename = iv_source_table_name
          iv_targetkeyspacename = iv_target_keyspace_name
          iv_targettablename = iv_target_table_name
          iv_restoretimestamp = iv_restore_timestamp ).
        MESSAGE 'Table restore initiated successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RestoreTable referência da API AWS SDK for SAP ABAP.

`UpdateTable`

O código de exemplo a seguir mostra como usar `UpdateTable`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Add a new column to track watched movies
        DATA(lt_add_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'watched' iv_type =
 'boolean' ) )
        ).

        oo_result = lo_kys->updatetable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name
          it_addcolumns = lt_add_columns ).
        MESSAGE 'Table updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateTable referência da API AWS SDK for SAP ABAP.

Exemplos do Kinesis usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com Kinesis.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um fluxo e inserir um registro nele.
- Criar um iterador de fragmento.
- Ler o registro e, em seguida, limpar os recursos.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   DATA lo_stream_describe_result TYPE REF TO /aws1/cl_knsdescrstreamoutput.
   DATA lo_stream_description TYPE REF TO /aws1/cl_knsstreamdescription.
   DATA lo_sharditerator TYPE REF TO /aws1/cl_knsgetsharditerator01.
   DATA lo_record_result TYPE REF TO /aws1/cl_knsputrecordoutput.

   "Create stream."
   TRY.
       lo_kns->createstream(
           iv_streamname = iv_stream_name
           iv_shardcount = iv_shard_count ).
       MESSAGE 'Stream created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
```

```
   ENDTRY.

   "Wait for stream to becomes active."
   lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
   lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
   WHILE lo_stream_description->get_streamstatus( ) <> 'ACTIVE'.
     IF sy-index = 30.
       EXIT.               "maximum 5 minutes"
     ENDIF.
     WAIT UP TO 10 SECONDS.
     lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
     lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
   ENDWHILE.

   "Create record."
   TRY.
       lo_record_result = lo_kns->putrecord(
           iv_streamname = iv_stream_name
           iv_data       = iv_data
           iv_partitionkey = iv_partition_key ).
       MESSAGE 'Record created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_knsresourcenotfoundex.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
```

```
   ENDTRY.

   "Create a shard iterator in order to read the record."
   TRY.
       lo_sharditerator = lo_kns->getsharditerator(
         iv_shardid = lo_record_result->get_shardid( )
         iv_sharditeratortype = iv_sharditeratortype
         iv_streamname = iv_stream_name ).
       MESSAGE 'Shard iterator created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.

   "Read the record."
   TRY.
       oo_result = lo_kns->getrecords(                    " oo_result is returned
for testing purposes. "
           iv_sharditerator   = lo_sharditerator->get_sharditerator( ) ).
       MESSAGE 'Shard iterator created.' TYPE 'I'.
     CATCH /aws1/cx_knsexpirediteratorex.
       MESSAGE 'Iterator expired.' TYPE 'E'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
```

```
        MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.

    "Delete stream."
    TRY.
        lo_kns->deletestream(
            iv_streamname = iv_stream_name ).
        MESSAGE 'Stream deleted.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourceinuseex.
        MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - CreateStream
  - DeleteStream
  - GetRecords
  - GetShardIterator
  - PutRecord

Ações

`CreateStream`

O código de exemplo a seguir mostra como usar `CreateStream`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_kns->createstream(
           iv_streamname = iv_stream_name
           iv_shardcount = iv_shard_count ).
       MESSAGE 'Stream created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateStream referência da API AWS SDK for SAP ABAP.

`DeleteStream`

O código de exemplo a seguir mostra como usar `DeleteStream`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
```

```
        lo_kns->deletestream(
            iv_streamname = iv_stream_name ).
        MESSAGE 'Stream deleted.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourceinuseex.
        MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteStream referência da API AWS SDK for SAP ABAP.

`DescribeStream`

O código de exemplo a seguir mostra como usar `DescribeStream`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kns->describestream(
            iv_streamname = iv_stream_name ).
        DATA(lt_stream_description) = oo_result->get_streamdescription( ).
        MESSAGE 'Streams retrieved.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeStream referência da API AWS SDK for SAP ABAP.

`GetRecords`

O código de exemplo a seguir mostra como usar `GetRecords`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_kns->getrecords(             " oo_result is returned for
testing purposes. "
           iv_sharditerator = iv_shard_iterator ).
       DATA(lt_records) = oo_result->get_records( ).
       MESSAGE 'Record retrieved.' TYPE 'I'.
     CATCH /aws1/cx_knsexpirediteratorex.
       MESSAGE 'Iterator expired.' TYPE 'E'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
```

```
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetRecords referência da API AWS SDK for SAP ABAP.

ListStreams

O código de exemplo a seguir mostra como usar `ListStreams`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_kns->liststreams(        " oo_result is returned for testing
purposes. "
            "Set Limit to specify that a maximum of streams should be returned."
            iv_limit = iv_limit ).
        DATA(lt_streams) = oo_result->get_streamnames( ).
        MESSAGE 'Streams listed.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListStreams referência da API AWS SDK for SAP ABAP.

PutRecord

O código de exemplo a seguir mostra como usar `PutRecord`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_kns->putrecord(            " oo_result is returned for
testing purposes. "
           iv_streamname = iv_stream_name
           iv_data       = iv_data
           iv_partitionkey = iv_partition_key ).
       MESSAGE 'Record created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_knsresourcenotfoundex.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a PutRecord referência da API AWS SDK for SAP ABAP.

`RegisterStreamConsumer`

O código de exemplo a seguir mostra como usar `RegisterStreamConsumer`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_kns->registerstreamconsumer(       " oo_result is returned
for testing purposes. "
           iv_streamarn = iv_stream_arn
           iv_consumername = iv_consumer_name ).
       MESSAGE 'Stream consumer registered.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a RegisterStreamConsumer referência da API AWS SDK for SAP ABAP.

AWS KMS exemplos de uso do SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com. AWS KMS

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Hello AWS KMS

O exemplo de código a seguir mostra como começar a usar o AWS Key Management Service.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_kms->listkeys( ).
    MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListKeys referência da API AWS SDK for SAP ABAP.

Ações

`CreateAlias`

O código de exemplo a seguir mostra como usar `CreateAlias`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->createalias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_key_id
        ).
        MESSAGE 'Alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsalreadyexistsex.
        MESSAGE 'Alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidaliasnameex.
        MESSAGE 'Invalid alias name.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateAlias referência da API AWS SDK for SAP ABAP.

`CreateGrant`

O código de exemplo a seguir mostra como usar `CreateGrant`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grantee_principal = 'arn:aws:iam::123456789012:role/my-role'
        " it_operations contains 'Encrypt', 'Decrypt', 'GenerateDataKey'
        oo_result = lo_kms->creategrant(
          iv_keyid = iv_key_id
          iv_granteeprincipal = iv_grantee_principal
          it_operations = it_operations
        ).
        MESSAGE 'Grant created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateGrant referência da API AWS SDK for SAP ABAP.

`CreateKey`

O código de exemplo a seguir mostra como usar `CreateKey`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_description = 'Created by the AWS SDK for SAP ABAP'
        oo_result = lo_kms->createkey( iv_description = iv_description ).
        MESSAGE 'KMS key created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
      CATCH /aws1/cx_kmslimitexceededex.
        MESSAGE 'Limit exceeded for KMS resources.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateKey referência da API AWS SDK for SAP ABAP.

`Decrypt`

O código de exemplo a seguir mostra como usar `Decrypt`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->decrypt(
          iv_keyid = iv_key_id
          iv_ciphertextblob = iv_ciphertext_blob
```

```
      ).
      MESSAGE 'Text decrypted successfully.' TYPE 'I'.
    CATCH /aws1/cx_kmsdisabledexception.
      MESSAGE 'The key is disabled.' TYPE 'E'.
    CATCH /aws1/cx_kmsincorrectkeyex.
      MESSAGE 'Incorrect key for decryption.' TYPE 'E'.
    CATCH /aws1/cx_kmsnotfoundexception.
      MESSAGE 'Key not found.' TYPE 'E'.
    CATCH /aws1/cx_kmskmsinternalex.
      MESSAGE 'An internal error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte Decrypt in AWS SDK for SAP ABAP API reference.

`DeleteAlias`

O código de exemplo a seguir mostra como usar `DeleteAlias`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " iv_alias_name = 'alias/my-key-alias'
      lo_kms->deletealias( iv_aliasname = iv_alias_name ).
      MESSAGE 'Alias deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_kmsnotfoundexception.
      MESSAGE 'Alias not found.' TYPE 'E'.
    CATCH /aws1/cx_kmskmsinternalex.
      MESSAGE 'An internal error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteAlias referência da API AWS SDK for SAP ABAP.

`DescribeKey`

O código de exemplo a seguir mostra como usar `DescribeKey`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->describekey( iv_keyid = iv_key_id ).
        DATA(lo_key) = oo_result->get_keymetadata( ).
        MESSAGE 'Retrieved key information successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeKey referência da API AWS SDK for SAP ABAP.

`DisableKey`

O código de exemplo a seguir mostra como usar `DisableKey`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->disablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DisableKey referência da API AWS SDK for SAP ABAP.

`EnableKey`

O código de exemplo a seguir mostra como usar `EnableKey`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a EnableKey referência da API AWS SDK for SAP ABAP.

`EnableKeyRotation`

O código de exemplo a seguir mostra como usar `EnableKeyRotation`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekeyrotation( iv_keyid = iv_key_id ).
        MESSAGE 'Key rotation enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsunsupportedopex.
        MESSAGE 'Operation not supported for this key.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a EnableKeyRotation referência da API AWS SDK for SAP ABAP.

`Encrypt`

O código de exemplo a seguir mostra como usar `Encrypt`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_plaintext contains the data to encrypt
        oo_result = lo_kms->encrypt(
          iv_keyid = iv_key_id
          iv_plaintext = iv_plaintext
        ).
        MESSAGE 'Text encrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte Criptografar no AWS SDK para referência da API SAP ABAP.

`GenerateDataKey`

O código de exemplo a seguir mostra como usar `GenerateDataKey`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_keyspec = 'AES_256'
        oo_result = lo_kms->generatedatakey(
          iv_keyid = iv_key_id
          iv_keyspec = 'AES_256'
        ).
        MESSAGE 'Data key generated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GenerateDataKey referência da API AWS SDK for SAP ABAP.

`GetKeyPolicy`

O código de exemplo a seguir mostra como usar `GetKeyPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->getkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
        ).
        MESSAGE 'Retrieved key policy successfully.' TYPE 'I'.
```

```
    CATCH /aws1/cx_kmsnotfoundexception.
      MESSAGE 'Key not found.' TYPE 'E'.
    CATCH /aws1/cx_kmskmsinternalex.
      MESSAGE 'An internal error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetKeyPolicy referência da API AWS SDK for SAP ABAP.

`ListAliases`

O código de exemplo a seguir mostra como usar `ListAliases`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_kms->listaliases( ).
      MESSAGE 'Retrieved KMS aliases list.' TYPE 'I'.
    CATCH /aws1/cx_kmskmsinternalex.
      MESSAGE 'An internal error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListAliases referência da API AWS SDK for SAP ABAP.

`ListGrants`

O código de exemplo a seguir mostra como usar `ListGrants`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listgrants( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved grants list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListGrants referência da API AWS SDK for SAP ABAP.

`ListKeyPolicies`

O código de exemplo a seguir mostra como usar `ListKeyPolicies`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listkeypolicies( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved key policies list.' TYPE 'I'.
```

```
    CATCH /aws1/cx_kmsnotfoundexception.
      MESSAGE 'Key not found.' TYPE 'E'.
    CATCH /aws1/cx_kmskmsinternalex.
      MESSAGE 'An internal error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListKeyPolicies referência da API AWS SDK for SAP ABAP.

`ListKeys`

O código de exemplo a seguir mostra como usar `ListKeys`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_kms->listkeys( ).
      MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
    CATCH /aws1/cx_kmskmsinternalex.
      MESSAGE 'An internal error occurred.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListKeys referência da API AWS SDK for SAP ABAP.

`PutKeyPolicy`

O código de exemplo a seguir mostra como usar `PutKeyPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_policy = '{"Version":"2012-10-17",        "Statement": [...]}'
        lo_kms->putkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
          iv_policy = iv_policy
        ).
        MESSAGE 'Key policy updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsmalformedplydocex.
        MESSAGE 'Malformed policy document.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutKeyPolicy referência da API AWS SDK for SAP ABAP.

`ReEncrypt`

O código de exemplo a seguir mostra como usar `ReEncrypt`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_source_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_destination_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->reencrypt(
          iv_sourcekeyid = iv_source_key_id
          iv_destinationkeyid = iv_destination_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Ciphertext reencrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect source key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ReEncrypt referência da API AWS SDK for SAP ABAP.

`RetireGrant`

O código de exemplo a seguir mostra como usar `RetireGrant`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_grant_token = 'AQpAM2RhZ...'
        lo_kms->retiregrant( iv_granttoken = iv_grant_token ).
        MESSAGE 'Grant retired successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Grant not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvgranttokenex.
        MESSAGE 'Invalid grant token.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RetireGrant referência da API AWS SDK for SAP ABAP.

`RevokeGrant`

O código de exemplo a seguir mostra como usar `RevokeGrant`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grant_id = '1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p'
        lo_kms->revokegrant(
          iv_keyid = iv_key_id
          iv_grantid = iv_grant_id
        ).
        MESSAGE 'Grant revoked successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Grant or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidgrantidex.
        MESSAGE 'Invalid grant ID.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RevokeGrant referência da API AWS SDK for SAP ABAP.

`ScheduleKeyDeletion`

O código de exemplo a seguir mostra como usar `ScheduleKeyDeletion`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_pending_window_days = 7
        oo_result = lo_kms->schedulekeydeletion(
          iv_keyid = iv_key_id
          iv_pendingwindowindays = iv_pending_window_days
        ).
        MESSAGE 'Key scheduled for deletion.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ScheduleKeyDeletion referência da API AWS SDK for SAP ABAP.

`Sign`

O código de exemplo a seguir mostra como usar `Sign`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the message to sign
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->sign(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signingalgorithm = iv_signing_algorithm
        ).
        MESSAGE 'Message signed successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidkeyusageex.
        MESSAGE 'Key cannot be used for signing.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte Referência da API Sign in AWS SDK for SAP ABAP.

`TagResource`

O código de exemplo a seguir mostra como usar `TagResource`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_tags TYPE /aws1/cl_kmstag=>tt_taglist.

    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_tag_key = 'Environment'
        " iv_tag_value = 'Production'
        APPEND NEW /aws1/cl_kmstag(
          iv_tagkey = iv_tag_key
          iv_tagvalue = iv_tag_value
        ) TO lt_tags.

        lo_kms->tagresource(
          iv_keyid = iv_key_id
          it_tags = lt_tags
        ).
        MESSAGE 'Tag added to KMS key successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmstagexception.
        MESSAGE 'Invalid tag format.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a TagResource referência da API AWS SDK for SAP ABAP.

`UpdateAlias`

O código de exemplo a seguir mostra como usar `UpdateAlias`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_target_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        lo_kms->updatealias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_target_key_id
        ).
        MESSAGE 'Alias updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Alias or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateAlias referência da API AWS SDK for SAP ABAP.

`Verificar`

O código de exemplo a seguir mostra como usar `Verify`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the original message
        " iv_signature contains the signature to verify
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->verify(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signature = iv_signature
          iv_signingalgorithm = iv_signing_algorithm
        ).
        DATA(lv_valid) = oo_result->get_signaturevalid( ).
        IF lv_valid = abap_true.
          MESSAGE 'Signature is valid.' TYPE 'I'.
        ELSE.
          MESSAGE 'Signature is invalid.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinvalidsigex.
        MESSAGE 'Invalid signature.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte Verify in AWS SDK for SAP ABAP API reference.

Exemplos do Lambda usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       "Create an AWS Identity and Access Management (IAM) role that grants AWS
Lambda permission to write to logs."
       DATA(lv_policy_document) = `{` &&
           `"Version":"2012-10-17",        ` &&
                 `"Statement": [` &&
                   `{` &&
                     `"Effect": "Allow",` &&
```

```
                      `"Action": [` &&
                        `"sts:AssumeRole"` &&
                      `],` &&
                      `"Principal": {` &&
                        `"Service": [` &&
                          `"lambda.amazonaws.com"` &&
                        `]` &&
                      `}` &&
                    `}` &&
                  `]` &&
                `}`.
        TRY.
            DATA(lo_create_role_output) = lo_iam->createrole(
                    iv_rolename = iv_role_name
                    iv_assumerolepolicydocument = lv_policy_document
                    iv_description = 'Grant lambda permission to write to logs' ).
            DATA(lv_role_arn) = lo_create_role_output->get_role( )->get_arn( ).
            MESSAGE 'IAM role created.' TYPE 'I'.
            WAIT UP TO 10 SECONDS.            " Make sure that the IAM role is ready
 for use. "
          CATCH /aws1/cx_iamentityalrdyexex.
            DATA(lo_role) = lo_iam->getrole( iv_rolename = iv_role_name ).
            lv_role_arn = lo_role->get_role( )->get_arn( ).
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iammalformedplydocex.
            MESSAGE 'Policy document in the request is malformed.' TYPE 'E'.
        ENDTRY.

        TRY.
            lo_iam->attachrolepolicy(
                iv_rolename  = iv_role_name
                iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' ).
            MESSAGE 'Attached policy to the IAM role.' TYPE 'I'.
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'E'.
          CATCH /aws1/cx_iamplynotattachableex.
            MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
          CATCH /aws1/cx_iamunmodableentityex.
```

```
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.

        " Create a Lambda function and upload handler code. "
        " Lambda function performs 'increment' action on a number. "
        TRY.
            lo_lmd->createfunction(
                 iv_functionname = iv_function_name
                 iv_runtime = `python3.9`
                 iv_role = lv_role_arn
                 iv_handler = iv_handler
                 io_code = io_initial_zip_file
                 iv_description = 'AWS Lambda code example' ).
            MESSAGE 'Lambda function created.' TYPE 'I'.
          CATCH /aws1/cx_lmdcodestorageexcdex.
            MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        " Verify the function is in Active state "
        WHILE lo_lmd->getfunction( iv_functionname = iv_function_name )-
>get_configuration( )->ask_state( ) <> 'Active'.
          IF sy-index = 10.
            EXIT.               " Maximum 10 seconds. "
          ENDIF.
          WAIT UP TO 1 SECONDS.
        ENDWHILE.

        "Invoke the function with a single parameter and get results."
        TRY.
            DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
              `{`  &&
                `"action": "increment",`  &&
                `"number": 10` &&
              `}` ).
            DATA(lo_initial_invoke_output) = lo_lmd->invoke(
                       iv_functionname = iv_function_name
                       iv_payload = lv_json ).
            ov_initial_invoke_payload = lo_initial_invoke_output->get_payload( ).
        " ov_initial_invoke_payload is returned for testing purposes. "
```

```
            DATA(lo_writer_json) = cl_sxml_string_writer=>create( type =
 if_sxml=>co_xt_json ).
            CALL TRANSFORMATION id SOURCE XML ov_initial_invoke_payload RESULT XML
 lo_writer_json.
            DATA(lv_result) = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
            MESSAGE 'Lambda function invoked.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
          CATCH /aws1/cx_lmdunsuppedmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
        ENDTRY.

        " Update the function code and configure its Lambda environment with an
 environment variable. "
        " Lambda function is updated to perform 'decrement' action also. "
        TRY.
            lo_lmd->updatefunctioncode(
                  iv_functionname = iv_function_name
                  iv_zipfile = io_updated_zip_file ).
            WAIT UP TO 10 SECONDS.            " Make sure that the update is
 completed. "
            MESSAGE 'Lambda function code updated.' TYPE 'I'.
          CATCH /aws1/cx_lmdcodestorageexcdex.
            MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        TRY.
            DATA lt_variables TYPE /aws1/
cl_lmdenvironmentvaria00=>tt_environmentvariables.
            DATA ls_variable LIKE LINE OF lt_variables.
            ls_variable-key = 'LOG_LEVEL'.
            ls_variable-value = NEW /aws1/cl_lmdenvironmentvaria00( iv_value =
 'info' ).
            INSERT ls_variable INTO TABLE lt_variables.
```

```
            lo_lmd->updatefunctionconfiguration(
                  iv_functionname = iv_function_name
                  io_environment = NEW /aws1/cl_lmdenvironment( it_variables =
 lt_variables ) ).
            WAIT UP TO 10 SECONDS.            " Make sure that the update is
 completed. "
            MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourceconflictex.
            MESSAGE 'Resource already exists or another operation is in progress.'
 TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        "Invoke the function with new parameters and get results. Display the
 execution log that's returned from the invocation."
        TRY.
            lv_json = /aws1/cl_rt_util=>string_to_xstring(
              `{`  &&
                `"action": "decrement",`  &&
                `"number": 10` &&
              `}` ).
            DATA(lo_updated_invoke_output) = lo_lmd->invoke(
                       iv_functionname = iv_function_name
                       iv_payload = lv_json ).
            ov_updated_invoke_payload = lo_updated_invoke_output->get_payload( ).
        " ov_updated_invoke_payload is returned for testing purposes. "
            lo_writer_json = cl_sxml_string_writer=>create( type =
 if_sxml=>co_xt_json ).
            CALL TRANSFORMATION id SOURCE XML ov_updated_invoke_payload RESULT XML
 lo_writer_json.
            lv_result = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
            MESSAGE 'Lambda function invoked.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
          CATCH /aws1/cx_lmdunsuppedmediatyp00.
```

```
            MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
        ENDTRY.

        " List the functions for your account. "
        TRY.
            DATA(lo_list_output) = lo_lmd->listfunctions( ).
            DATA(lt_functions) = lo_list_output->get_functions( ).
            MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
        ENDTRY.

        " Delete the Lambda function. "
        TRY.
            lo_lmd->deletefunction( iv_functionname = iv_function_name ).
            MESSAGE 'Lambda function deleted.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'W'.
        ENDTRY.

        " Detach role policy. "
        TRY.
            lo_iam->detachrolepolicy(
                iv_rolename  = iv_role_name
                iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' ).
            MESSAGE 'Detached policy from the IAM role.' TYPE 'I'.
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'W'.
          CATCH /aws1/cx_iamplynotattachableex.
            MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.

        " Delete the IAM role. "
        TRY.
```

```
            lo_iam->deleterole( iv_rolename = iv_role_name ).
            MESSAGE 'IAM role deleted.' TYPE 'I'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'W'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
modifiable.' TYPE 'E'.
        ENDTRY.

      CATCH /aws1/cx_rt_service_generic INTO lo_exception.
        DATA(lv_error) = lo_exception->get_longtext( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_lmd->createfunction(
           iv_functionname = iv_function_name
           iv_runtime = `python3.9`
           iv_role = iv_role_arn
           iv_handler = iv_handler
           io_code = io_zip_file
           iv_description = 'AWS Lambda code example' ).
       MESSAGE 'Lambda function created.' TYPE 'I'.
     CATCH /aws1/cx_lmdcodesigningcfgno00.
       MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodestorageexcdex.
       MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodeverification00.
       MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvalidcodesigex.
       MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateFunction referência da API AWS SDK for SAP ABAP.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_lmd->deletefunction( iv_functionname = iv_function_name ).
       MESSAGE 'Lambda function deleted.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteFunction referência da API AWS SDK for SAP ABAP.

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_lmd->getfunction( iv_functionname = iv_function_name ).
" oo_result is returned for testing purposes. "
       MESSAGE 'Lambda function information retrieved.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetFunction referência da API AWS SDK for SAP ABAP.

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
         `{`  &&
           `"action": "increment",`  &&
           `"number": 10` &&
         `}` ).
       oo_result = lo_lmd->invoke(                  " oo_result is returned for
testing purposes. "
                iv_functionname = iv_function_name
                iv_payload = lv_json ).
       MESSAGE 'Lambda function invoked.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
```

```
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvrequestcontex.
        MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidzipfileex.
        MESSAGE 'The deployment package could not be unzipped.' TYPE 'E'.
      CATCH /aws1/cx_lmdrequesttoolargeex.
        MESSAGE 'Invoke request body JSON input limit was exceeded by the request
payload.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
      CATCH /aws1/cx_lmdunsuppedmediatyp00.
        MESSAGE 'Invoke request body does not have JSON as its content type.' TYPE
'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Invoke na Referência da API do AWS SDK para SAP ABAP.

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
```

```
        oo_result = lo_lmd->listfunctions( ).       " oo_result is returned for
testing purposes. "
        DATA(lt_functions) = oo_result->get_functions( ).
        MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListFunctions referência da API AWS SDK for SAP ABAP.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_lmd->updatefunctioncode(     " oo_result is returned for
testing purposes. "
              iv_functionname = iv_function_name
              iv_zipfile = io_zip_file ).

        MESSAGE 'Lambda function code updated.' TYPE 'I'.
      CATCH /aws1/cx_lmdcodesigningcfgno00.
        MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdcodestorageexcdex.
        MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
      CATCH /aws1/cx_lmdcodeverification00.
```

```
        MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidcodesigex.
        MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateFunctionCode referência da API AWS SDK for SAP ABAP.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_lmd->updatefunctionconfiguration(     " oo_result is returned
for testing purposes. "
              iv_functionname = iv_function_name
              iv_runtime = iv_runtime
              iv_description  = 'Updated Lambda function'
              iv_memorysize  = iv_memory_size ).
```

```
        MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
      CATCH /aws1/cx_lmdcodesigningcfgno00.
        MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdcodeverification00.
        MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidcodesigex.
        MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateFunctionConfiguration referência da API AWS SDK for SAP ABAP.

Exemplos de organizações usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP with Organizations.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AttachPolicy`

O código de exemplo a seguir mostra como usar `AttachPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_org->attachpolicy(
      iv_policyid = iv_policy_id
      iv_targetid = iv_target_id ).
    MESSAGE 'Policy attached to target.' TYPE 'I'.
  CATCH /aws1/cx_orgaccessdeniedex.
    MESSAGE 'You do not have permission to attach the policy.' TYPE 'E'.
  CATCH /aws1/cx_orgpolicynotfoundex.
    MESSAGE 'The specified policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgtargetnotfoundex.
    MESSAGE 'The specified target does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgduplicateplyatta00.
    MESSAGE 'The policy is already attached to the target.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a AttachPolicy referência da API AWS SDK for SAP ABAP.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_org->createpolicy(       " oo_result is returned for testing
purposes. "
         iv_name        = iv_policy_name
         iv_description = iv_policy_description
         iv_content     = iv_policy_content
         iv_type        = iv_policy_type ).
       MESSAGE 'Policy created.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to create a policy.' TYPE 'E'.
     CATCH /aws1/cx_orgduplicatepolicyex.
       MESSAGE 'A policy with this name already exists.' TYPE 'E'.
     CATCH /aws1/cx_orgmalformedplydocex.
       MESSAGE 'The policy content is malformed.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreatePolicy referência da API AWS SDK for SAP ABAP.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_org->deletepolicy(
          iv_policyid = iv_policy_id ).
        MESSAGE 'Policy deleted.' TYPE 'I'.
      CATCH /aws1/cx_orgaccessdeniedex.
        MESSAGE 'You do not have permission to delete the policy.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicynotfoundex.
        MESSAGE 'The specified policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicyinuseex.
        MESSAGE 'The policy is still attached to one or more targets.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeletePolicy referência da API AWS SDK for SAP ABAP.

`DescribePolicy`

O código de exemplo a seguir mostra como usar `DescribePolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_org->describepolicy(     " oo_result is returned for testing
purposes. "
          iv_policyid = iv_policy_id ).
        DATA(lo_policy) = oo_result->get_policy( ).
        MESSAGE 'Retrieved policy details.' TYPE 'I'.
      CATCH /aws1/cx_orgaccessdeniedex.
        MESSAGE 'You do not have permission to describe the policy.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicynotfoundex.
        MESSAGE 'The specified policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribePolicy referência da API AWS SDK for SAP ABAP.

DetachPolicy

O código de exemplo a seguir mostra como usar `DetachPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_org->detachpolicy(
      iv_policyid = iv_policy_id
      iv_targetid = iv_target_id ).
    MESSAGE 'Policy detached from target.' TYPE 'I'.
  CATCH /aws1/cx_orgaccessdeniedex.
    MESSAGE 'You do not have permission to detach the policy.' TYPE 'E'.
  CATCH /aws1/cx_orgpolicynotfoundex.
    MESSAGE 'The specified policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgtargetnotfoundex.
    MESSAGE 'The specified target does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgpolicynotattex.
    MESSAGE 'The policy is not attached to the target.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetachPolicy referência da API AWS SDK for SAP ABAP.

ListPolicies

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_org->listpolicies(       " oo_result is returned for testing
purposes. "
         iv_filter = iv_filter ).
       DATA(lt_policies) = oo_result->get_policies( ).
       MESSAGE 'Retrieved list of policies.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to list policies.' TYPE 'E'.
     CATCH /aws1/cx_orgawsorgsnotinuseex.
       MESSAGE 'Your account is not a member of an organization.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListPolicies referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Pinpoint usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`SendMessages`

O código de exemplo a seguir mostra como usar `SendMessages`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de e-mail.

```
" Build the addresses map from the list of to_addresses
DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
LOOP AT it_to_addresses INTO DATA(lo_address).
  INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
    key = lo_address->get_value( )
    value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'EMAIL' )
  ) INTO TABLE lt_addresses.
ENDLOOP.

" Send the email message
DATA(lo_result) = lo_ppt->sendmessages(
  iv_applicationid = iv_app_id
  io_messagerequest = NEW /aws1/cl_pptmessagerequest(
    it_addresses = lt_addresses
    io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
      io_emailmessage = NEW /aws1/cl_pptemailmessage(
        iv_fromaddress = iv_sender
        io_simpleemail = NEW /aws1/cl_pptsimpleemail(
          io_subject = NEW /aws1/cl_pptsimpleemailpart(
            iv_charset = iv_char_set
            iv_data = iv_subject
          )
          io_htmlpart = NEW /aws1/cl_pptsimpleemailpart(
            iv_charset = iv_char_set
            iv_data = iv_html_message
          )
          io_textpart = NEW /aws1/cl_pptsimpleemailpart(
```

```
                iv_charset = iv_char_set
                iv_data = iv_text_message
              )
            )
          )
        )
      )
    ).

    " Extract message IDs from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    ot_message_ids = lo_message_response->get_result( ).

    MESSAGE 'Email message sent successfully.' TYPE 'I'.
```

Envie uma mensagem SMS.

```
    " Build the addresses map for the destination number
    DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
    INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
      key = iv_destination_number
      value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'SMS' )
    ) INTO TABLE lt_addresses.

    " Send the SMS message
    DATA(lo_result) = lo_ppt->sendmessages(
      iv_applicationid = iv_app_id
      io_messagerequest = NEW /aws1/cl_pptmessagerequest(
        it_addresses = lt_addresses
        io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
          io_smsmessage = NEW /aws1/cl_pptsmsmessage(
            iv_body = iv_message
            iv_messagetype = iv_message_type
            iv_originationnumber = iv_origination_number
          )
        )
      )
    ).

    " Extract message ID from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    DATA(lt_results) = lo_message_response->get_result( ).
```

```
    LOOP AT lt_results INTO DATA(ls_result).
      IF ls_result-key = iv_destination_number.
        ov_message_id = ls_result-value->get_messageid( ).
        EXIT.
      ENDIF.
    ENDLOOP.

    MESSAGE 'SMS message sent successfully.' TYPE 'I'.
```

Enviar uma mensagem com um modelo de e-mail existente.

```
    " Build the addresses map from the list of to_addresses
    DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
    LOOP AT it_to_addresses INTO DATA(lo_address).
      INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
        key = lo_address->get_value( )
        value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'EMAIL' )
      ) INTO TABLE lt_addresses.
    ENDLOOP.

    " Send the email message using a template
    DATA(lo_result) = lo_ppt->sendmessages(
      iv_applicationid = iv_app_id
      io_messagerequest = NEW /aws1/cl_pptmessagerequest(
        it_addresses = lt_addresses
        io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
          io_emailmessage = NEW /aws1/cl_pptemailmessage(
            iv_fromaddress = iv_sender
          )
        )
        io_templateconfiguration = NEW /aws1/cl_ppttemplateconf(
          io_emailtemplate = NEW /aws1/cl_ppttemplate(
            iv_name = iv_template_name
            iv_version = iv_template_version
          )
        )
      )
    ).

    " Extract message IDs from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    ot_message_ids = lo_message_response->get_result( ).
```

```
    MESSAGE 'Templated email message sent successfully.' TYPE 'I'.
```

Envie uma mensagem de texto com um modelo de SMS existente.

```
    " Build the addresses map for the destination number
    DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
    INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
      key = iv_destination_number
      value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'SMS' )
    ) INTO TABLE lt_addresses.

    " Send the SMS message using a template
    DATA(lo_result) = lo_ppt->sendmessages(
      iv_applicationid = iv_app_id
      io_messagerequest = NEW /aws1/cl_pptmessagerequest(
        it_addresses = lt_addresses
        io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
          io_smsmessage = NEW /aws1/cl_pptsmsmessage(
            iv_messagetype = iv_message_type
            iv_originationnumber = iv_origination_number
          )
        )
        io_templateconfiguration = NEW /aws1/cl_ppttemplateconf(
          io_smstemplate = NEW /aws1/cl_ppttemplate(
            iv_name = iv_template_name
            iv_version = iv_template_version
          )
        )
      )
    ).

    " Extract message ID from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    DATA(lt_results) = lo_message_response->get_result( ).
    LOOP AT lt_results INTO DATA(ls_result).
      IF ls_result-key = iv_destination_number.
        ov_message_id = ls_result-value->get_messageid( ).
        EXIT.
      ENDIF.
    ENDLOOP.
```

```
    MESSAGE 'Templated SMS message sent successfully.' TYPE 'I'.
```

- Para obter detalhes da API, consulte a SendMessages referência da API AWS SDK for SAP ABAP.

Exemplos da API de SMS e voz do Amazon Pinpoint usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com a API de voz e SMS do Amazon Pinpoint.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateConfigurationSet`

O código de exemplo a seguir mostra como usar `CreateConfigurationSet`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create a new configuration set
        lo_pps->createconfigurationset(
```

```
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
        ).

        MESSAGE 'Configuration set created successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsalreadyexistsex INTO DATA(lo_already_exists_ex).
        MESSAGE lo_already_exists_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_already_exists_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppslimitexceededex INTO DATA(lo_limit_exceeded_ex).
        MESSAGE lo_limit_exceeded_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_limit_exceeded_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateConfigurationSet referência da API AWS SDK for SAP ABAP.

`CreateConfigurationSetEventDestination`

O código de exemplo a seguir mostra como usar `CreateConfigurationSetEventDestination`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create event destination for the configuration set
```

```
        lo_pps->createconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-event-
dest'
          io_eventdestination = io_event_destination
        ).

        MESSAGE 'Event destination created successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsalreadyexistsex INTO DATA(lo_already_exists_ex).
        MESSAGE lo_already_exists_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_already_exists_ex.
      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppslimitexceededex INTO DATA(lo_limit_exceeded_ex).
        MESSAGE lo_limit_exceeded_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_limit_exceeded_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateConfigurationSetEventDestination referência da API AWS SDK for SAP ABAP.

`DeleteConfigurationSet`

O código de exemplo a seguir mostra como usar `DeleteConfigurationSet`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Delete the configuration set
        lo_pps->deleteconfigurationset(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
        ).

        MESSAGE 'Configuration set deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteConfigurationSet referência da API AWS SDK for SAP ABAP.

`DeleteConfigurationSetEventDestination`

O código de exemplo a seguir mostra como usar `DeleteConfigurationSetEventDestination`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Delete the event destination
        lo_pps->deleteconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-event-
dest'
        ).

        MESSAGE 'Event destination deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteConfigurationSetEventDestination referência da API AWS SDK for SAP ABAP.

`GetConfigurationSetEventDestinations`

O código de exemplo a seguir mostra como usar `GetConfigurationSetEventDestinations`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Get event destinations for the configuration set
        oo_result = lo_pps->getconfseteventdestinations(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
        ).

        " Process the event destinations
        LOOP AT oo_result->get_eventdestinations( ) INTO DATA(lo_event_dest).
          DATA(lv_dest_name) = lo_event_dest->get_name( ).
          DATA(lv_enabled) = lo_event_dest->get_enabled( ).

          MESSAGE |Event destination: { lv_dest_name }, Enabled: { lv_enabled }|
 TYPE 'I'.

          " Check for CloudWatch Logs destination
          DATA(lo_cloudwatch_dest) = lo_event_dest->get_cloudwatchlogsdst( ).
          IF lo_cloudwatch_dest IS NOT INITIAL.
            DATA(lv_log_group_arn) = lo_cloudwatch_dest->get_loggrouparn( ).
            MESSAGE |  CloudWatch Logs destination: { lv_log_group_arn }| TYPE 'I'.
          ENDIF.

          " Check for Kinesis Firehose destination
          DATA(lo_firehose_dest) = lo_event_dest->get_kinesisfirehosedst( ).
          IF lo_firehose_dest IS NOT INITIAL.
            DATA(lv_delivery_stream) = lo_firehose_dest->get_deliverystreamarn( ).
            MESSAGE |  Kinesis Firehose destination: { lv_delivery_stream }| TYPE
 'I'.
          ENDIF.

          " Check for SNS destination
          DATA(lo_sns_dest) = lo_event_dest->get_snsdestination( ).
          IF lo_sns_dest IS NOT INITIAL.
            DATA(lv_topic_arn) = lo_sns_dest->get_topicarn( ).
```

```
            MESSAGE |  SNS destination: { lv_topic_arn }| TYPE 'I'.
          ENDIF.
        ENDLOOP.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetConfigurationSetEventDestinations referência da API AWS SDK for SAP ABAP.

`ListConfigurationSets`

O código de exemplo a seguir mostra como usar `ListConfigurationSets`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " List all configuration sets
        oo_result = lo_pps->listconfigurationsets(
          iv_nexttoken = iv_next_token    " Optional: Token for pagination
          iv_pagesize = iv_page_size      " Optional: Number of results per page,
e.g., '10'
        ).
```

```
      " Process the configuration sets
      LOOP AT oo_result->get_configurationsets( ) INTO DATA(lo_config_set).
        DATA(lv_config_set_name) = lo_config_set->get_value( ).
        MESSAGE |Configuration set: { lv_config_set_name }| TYPE 'I'.
      ENDLOOP.

      " Check if there are more results
      DATA(lv_next_token) = oo_result->get_nexttoken( ).
      IF lv_next_token IS NOT INITIAL.
        MESSAGE |More results available. Next token: { lv_next_token }| TYPE 'I'.
      ENDIF.

    CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
      MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_internal_error_ex.
    CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
      MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many_requests_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListConfigurationSets referência da API AWS SDK for SAP ABAP.

`SendVoiceMessage`

O código de exemplo a seguir mostra como usar `SendVoiceMessage`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
```

```
      " Create SSML message type object with voice parameters
      DATA(lo_ssml_message) = NEW /aws1/cl_ppsssmlmessagetype(
        iv_languagecode = iv_language_code    " e.g., 'en-US'
        iv_voiceid = iv_voice_id              " e.g., 'Matthew'
        iv_text = iv_ssml_message             " SSML formatted message text
      ).

      " Create voice message content with the SSML message
      DATA(lo_content) = NEW /aws1/cl_ppsvoicemessagecont(
        io_ssmlmessage = lo_ssml_message
      ).

      " Send the voice message
      DATA(lo_result) = lo_pps->sendvoicemessage(
        iv_originationphonenumber = iv_origination_number  " e.g., '+12065550110'
        iv_callerid = iv_caller_id                         " e.g., '+12065550199'
        iv_destinationphonenumber = iv_destination_number  " e.g., '+12065550142'
        io_content = lo_content
      ).

      " Retrieve the message ID from the response
      ov_message_id = lo_result->get_messageid( ).

      MESSAGE 'Voice message sent successfully.' TYPE 'I'.

    CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
      MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_internal_error_ex.
    CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
      MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many_requests_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a SendVoiceMessage referência da API AWS SDK for SAP ABAP.

`UpdateConfigurationSetEventDestination`

O código de exemplo a seguir mostra como usar `UpdateConfigurationSetEventDestination`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Update the event destination
        lo_pps->updateconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-event-
dest'
          io_eventdestination = io_event_destination
        ).

        MESSAGE 'Event destination updated successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateConfigurationSetEventDestination referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Polly usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Polly.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`DeleteLexicon`

O código de exemplo a seguir mostra como usar `DeleteLexicon`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ply->deletelexicon( iv_name ).
    MESSAGE 'Lexicon deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_plylexiconnotfoundex.
    MESSAGE 'Lexicon not found.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteLexicon referência da API AWS SDK for SAP ABAP.

`DescribeVoices`

O código de exemplo a seguir mostra como usar `DescribeVoices`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Only pass optional parameters if they have values
    IF iv_engine IS NOT INITIAL AND iv_language IS NOT INITIAL.
      oo_result = lo_ply->describevoices(
        iv_engine = iv_engine
        iv_languagecode = iv_language ).
    ELSEIF iv_engine IS NOT INITIAL.
      oo_result = lo_ply->describevoices(
        iv_engine = iv_engine ).
    ELSEIF iv_language IS NOT INITIAL.
      oo_result = lo_ply->describevoices(
        iv_languagecode = iv_language ).
    ELSE.
      oo_result = lo_ply->describevoices( ).
    ENDIF.
    MESSAGE 'Retrieved voice metadata.' TYPE 'I'.
  CATCH /aws1/cx_plyinvalidnexttokenex.
    MESSAGE 'The NextToken is invalid.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeVoices referência da API AWS SDK for SAP ABAP.

`GetLexicon`

O código de exemplo a seguir mostra como usar `GetLexicon`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ply->getlexicon( iv_name ).
    DATA(lo_lexicon) = oo_result->get_lexicon( ).
    IF lo_lexicon IS BOUND.
      DATA(lv_lex_name) = lo_lexicon->get_name( ).
      MESSAGE |Retrieved lexicon: { lv_lex_name }| TYPE 'I'.
    ENDIF.
  CATCH /aws1/cx_plylexiconnotfoundex.
    MESSAGE 'Lexicon not found.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetLexicon referência da API AWS SDK for SAP ABAP.

`GetSpeechSynthesisTask`

O código de exemplo a seguir mostra como usar `GetSpeechSynthesisTask`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ply->getspeechsynthesistask( iv_task_id ).
        DATA(lo_task) = oo_result->get_synthesistask( ).
        IF lo_task IS BOUND.
          DATA(lv_status) = lo_task->get_taskstatus( ).
          MESSAGE |Task status: { lv_status }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_plyinvalidtaskidex.
        MESSAGE 'Invalid task ID.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plysynthesistsknotf00.
        MESSAGE 'Synthesis task not found.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetSpeechSynthesisTask referência da API AWS SDK for SAP ABAP.

`ListLexicons`

O código de exemplo a seguir mostra como usar `ListLexicons`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_ply->listlexicons( ).
        DATA(lt_lexicons) = oo_result->get_lexicons( ).
        DATA(lv_count) = lines( lt_lexicons ).
        MESSAGE |Found { lv_count } lexicons| TYPE 'I'.
      CATCH /aws1/cx_plyinvalidnexttokenex.
        MESSAGE 'Invalid NextToken.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListLexicons referência da API AWS SDK for SAP ABAP.

`ListSpeechSynthesisTasks`

O código de exemplo a seguir mostra como usar `ListSpeechSynthesisTasks`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Only pass optional parameters if they have values
    IF iv_max_results IS NOT INITIAL AND iv_status IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_maxresults = iv_max_results
        iv_status = iv_status ).
    ELSEIF iv_max_results IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_maxresults = iv_max_results ).
    ELSEIF iv_status IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_status = iv_status ).
    ELSE.
      oo_result = lo_ply->listspeechsynthesistasks( ).
    ENDIF.
    DATA(lt_tasks) = oo_result->get_synthesistasks( ).
    DATA(lv_count) = lines( lt_tasks ).
    MESSAGE |Found { lv_count } synthesis tasks| TYPE 'I'.
  CATCH /aws1/cx_plyinvalidnexttokenex.
    MESSAGE 'Invalid NextToken.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListSpeechSynthesisTasks referência da API AWS SDK for SAP ABAP.

`PutLexicon`

O código de exemplo a seguir mostra como usar `PutLexicon`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ply->putlexicon(
      iv_name = iv_name
      iv_content = iv_content ).
    MESSAGE 'Lexicon created successfully.' TYPE 'I'.
  CATCH /aws1/cx_plyinvalidlexiconex.
    MESSAGE 'Invalid lexicon.' TYPE 'E'.
  CATCH /aws1/cx_plylexiconsizeexcdex.
    MESSAGE 'Lexicon size exceeded.' TYPE 'E'.
  CATCH /aws1/cx_plymaxlexemelengthe00.
    MESSAGE 'Maximum lexeme length exceeded.' TYPE 'E'.
  CATCH /aws1/cx_plymaxlexiconsnoexc00.
    MESSAGE 'Maximum number of lexicons exceeded.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
  CATCH /aws1/cx_plyunsuppedplsalpha00.
    MESSAGE 'Unsupported PLS alphabet.' TYPE 'E'.
  CATCH /aws1/cx_plyunsuppedplslangu00.
    MESSAGE 'Unsupported PLS language.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutLexicon referência da API AWS SDK for SAP ABAP.

StartSpeechSynthesisTask

O código de exemplo a seguir mostra como usar `StartSpeechSynthesisTask`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Only pass optional parameters if they have values
        IF iv_lang_code IS NOT INITIAL AND iv_s3_key_prefix IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_outputs3keyprefix = iv_s3_key_prefix
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSEIF iv_lang_code IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSEIF iv_s3_key_prefix IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_outputs3keyprefix = iv_s3_key_prefix
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ELSE.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
```

```
            iv_outputs3bucketname = iv_s3_bucket
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ENDIF.
        MESSAGE 'Speech synthesis task started.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalids3bucketex.
        MESSAGE 'Invalid S3 bucket.' TYPE 'E'.
      CATCH /aws1/cx_plyinvalidssmlex.
        MESSAGE 'Invalid SSML.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plytextlengthexcdex.
        MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartSpeechSynthesisTask referência da API AWS SDK for SAP ABAP.

`SynthesizeSpeech`

O código de exemplo a seguir mostra como usar `SynthesizeSpeech`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Only pass optional language code if it has a value
        IF iv_lang_code IS NOT INITIAL.
          oo_result = lo_ply->synthesizespeech(
            iv_engine = iv_engine
            iv_outputformat = iv_output_fmt
            iv_text = iv_text
            iv_voiceid = iv_voice_id
```

```
            iv_languagecode = iv_lang_code ).
        ELSE.
          oo_result = lo_ply->synthesizespeech(
            iv_engine = iv_engine
            iv_outputformat = iv_output_fmt
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ENDIF.
        MESSAGE 'Speech synthesized successfully.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalidssmlex.
        MESSAGE 'Invalid SSML.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plytextlengthexcdex.
        MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SynthesizeSpeech referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon RDS usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon RDS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateDBParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBParameterGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" iv_dbparametergroupname   = 'mydbparametergroup'
" iv_dbparametergroupfamily = 'mysql8.0'
" iv_description            = 'My custom DB parameter group for MySQL 8.0'
TRY.
    oo_result = lo_rds->createdbparametergroup(
      iv_dbparametergroupname   = iv_dbparametergroupname
      iv_dbparametergroupfamily = iv_dbparametergroupfamily
      iv_description            = iv_description ).
    MESSAGE 'DB parameter group created.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbparmgralrexfault.
    MESSAGE 'DB parameter group already exists.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
    MESSAGE 'DB parameter group quota exceeded.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDBParameterGroup referência da API AWS SDK for SAP ABAP.

`DeleteDBParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBParameterGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " iv_dbparametergroupname = 'mydbparametergroup'
   TRY.
       lo_rds->deletedbparametergroup(
         iv_dbparametergroupname = iv_dbparametergroupname ).
       MESSAGE 'DB parameter group deleted.' TYPE 'I'.
     CATCH /aws1/cx_rdsdbprmgrnotfndfault.
       MESSAGE 'DB parameter group not found.' TYPE 'I'.
     CATCH /aws1/cx_rdsinvdbprmgrstatef00.
       MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDBParameterGroup referência da API AWS SDK for SAP ABAP.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   " iv_engine                 = 'mysql'
   " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter group
family)
   TRY.
```

```
    oo_result = lo_rds->describedbengineversions(
      iv_engine                 = iv_engine
      iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
    DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
    MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBEngineVersions referência da API AWS SDK for SAP ABAP.

`DescribeDBParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBParameterGroups`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" iv_dbparametergroupname = 'mydbparametergroup'
TRY.
    oo_result = lo_rds->describedbparametergroups(
      iv_dbparametergroupname = iv_dbparametergroupname ).
    MESSAGE 'DB parameter group retrieved.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    MESSAGE 'DB parameter group not found.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDBParameterGroups referência da API AWS SDK for SAP ABAP.

`DescribeDBParameters`

O código de exemplo a seguir mostra como usar `DescribeDBParameters`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" iv_dbparametergroupname = 'mydbparametergroup'
" iv_source               = 'user' (optional - filters by parameter source)
TRY.
    oo_result = lo_rds->describedbparameters(
      iv_dbparametergroupname = iv_dbparametergroupname
      iv_source               = iv_source ).
    DATA(lv_param_count) = lines( oo_result->get_parameters( ) ).
    MESSAGE |Retrieved { lv_param_count } parameters.| TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    MESSAGE 'DB parameter group not found.' TYPE 'I'.
ENDTRY.
```

- Para ver detalhes da API, consulte DescribeDBParameters na Referência de API do SDK da AWS para SAP ABAP.

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" iv_engine        = 'mysql'
" iv_engineversion = '8.0.35'
TRY.
```

```
        oo_result = lo_rds->descrorderabledbinstoptions(
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result->get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.| TYPE
'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeOrderableDBInstanceOptions referência da API AWS SDK for SAP ABAP.

`ModifyDBParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBParameterGroup`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    " it_parameters - table containing parameter objects with:
    "   - parametername = 'max_connections'
    "   - parametervalue = '100'
    "   - applymethod = 'immediate' or 'pending-reboot'
    TRY.
        oo_result = lo_rds->modifydbparametergroup(
          iv_dbparametergroupname = iv_dbparametergroupname
          it_parameters           = it_parameters ).
        MESSAGE 'DB parameter group modified.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ModifyDBParameterGroup referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Redshift usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Redshift.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateCluster`

O código de exemplo a seguir mostra como usar `CreateCluster`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie o cluster.

```
TRY.
    " Example values: iv_cluster_identifier = 'my-redshift-cluster'
    " Example values: iv_node_type = 'ra3.4xlarge'
    " Example values: iv_master_username = 'awsuser'
```

```
      " Example values: iv_master_password = 'AwsUser1000'
      " Example values: iv_publicly_accessible = abap_true
      " Example values: iv_number_of_nodes = 2
      oo_result = lo_rsh->createcluster(
        iv_clusteridentifier = iv_cluster_identifier
        iv_nodetype = iv_node_type
        iv_masterusername = iv_master_username
        iv_masteruserpassword = iv_master_password
        iv_publiclyaccessible = iv_publicly_accessible
        iv_numberofnodes = iv_number_of_nodes
      ).
      MESSAGE 'Redshift cluster created successfully.' TYPE 'I'.
    CATCH /aws1/cx_rshclustalrdyexfault.
      MESSAGE 'Cluster already exists.' TYPE 'I'.
    CATCH /aws1/cx_rshclstquotaexcdfault.
      MESSAGE 'Cluster quota exceeded.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateCluster referência da API AWS SDK for SAP ABAP.

`DeleteCluster`

O código de exemplo a seguir mostra como usar `DeleteCluster`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Excluir o cluster.

```
  TRY.
      " Example values: iv_cluster_identifier = 'my-redshift-cluster'
      lo_rsh->deletecluster(
        iv_clusteridentifier = iv_cluster_identifier
        iv_skipfinalclustersnapshot = abap_true
```

```
      ).
      MESSAGE 'Redshift cluster deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_rshclustnotfoundfault.
      MESSAGE 'Cluster not found.' TYPE 'I'.
    CATCH /aws1/cx_rshinvcluststatefault.
      MESSAGE 'Invalid cluster state for deletion.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCluster referência da API AWS SDK for SAP ABAP.

`DescribeClusters`

O código de exemplo a seguir mostra como usar `DescribeClusters`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Descrever o cluster.

```
  TRY.
      " Example values: iv_cluster_identifier = 'my-redshift-cluster' (optional)
      oo_result = lo_rsh->describeclusters(
        iv_clusteridentifier = iv_cluster_identifier
      ).
      lt_clusters = oo_result->get_clusters( ).
      lv_cluster_count = lines( lt_clusters ).
      MESSAGE |Retrieved { lv_cluster_count } cluster(s).| TYPE 'I'.
    CATCH /aws1/cx_rshclustnotfoundfault.
      MESSAGE 'Cluster not found.' TYPE 'I'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeClusters referência da API AWS SDK for SAP ABAP.

`DescribeStatement`

O código de exemplo a seguir mostra como usar `DescribeStatement`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
    oo_result = lo_rsd->describestatement(
      iv_id = iv_statement_id
    ).
    lv_status = oo_result->get_status( ).
    MESSAGE |Statement status: { lv_status }| TYPE 'I'.
  CATCH /aws1/cx_rsdresourcenotfoundex.
    MESSAGE 'Statement not found.' TYPE 'I'.
  CATCH /aws1/cx_rsdinternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeStatement referência da API AWS SDK for SAP ABAP.

`ExecuteStatement`

O código de exemplo a seguir mostra como usar `ExecuteStatement`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
        " Example values: iv_database_name = 'dev'
        " Example values: iv_user_name = 'awsuser'
        " Example values: iv_sql = 'SELECT * FROM movies WHERE year = :year'
        " Example values: it_parameter_list - SQL parameters for parameterized
queries

        " Only pass parameters if the list is not empty
        IF it_parameter_list IS NOT INITIAL.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
            it_parameters = it_parameter_list
          ).
        ELSE.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
          ).
        ENDIF.

        lv_statement_id = oo_result->get_id( ).
        MESSAGE |Statement executed. ID: { lv_statement_id }| TYPE 'I'.
      CATCH /aws1/cx_rsdexecutestatementex.
        MESSAGE 'Statement execution error.' TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ExecuteStatement referência da API AWS SDK for SAP ABAP.

`GetStatementResult`

O código de exemplo a seguir mostra como usar `GetStatementResult`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Verifique o resultado da instrução

```
    TRY.
        " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
        " Handle pagination for large result sets

        DO.
          lo_result_page = lo_rsd->getstatementresult(
            iv_id = iv_statement_id
            iv_nexttoken = lv_next_token
          ).

          " Collect records from this page
          lt_page_records = lo_result_page->get_records( ).
          APPEND LINES OF lt_page_records TO lt_all_records.

          " Check if there are more pages
          lv_next_token = lo_result_page->get_nexttoken( ).
          IF lv_next_token IS INITIAL.
            EXIT. " No more pages
          ENDIF.
        ENDDO.

        " For the last call, set oo_result for return value
        oo_result = lo_result_page.
        lv_record_count = lines( lt_all_records ).
        MESSAGE |Retrieved { lv_record_count } record(s).| TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Statement not found or results not available.' TYPE 'I'.
      CATCH /aws1/cx_rsdinternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetStatementResult referência da API AWS SDK for SAP ABAP.

`ListDatabases`

O código de exemplo a seguir mostra como usar `ListDatabases`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
    " Example values: iv_database_name = 'dev'
    " Example values: iv_database_user = 'awsuser'
    oo_result = lo_rsd->listdatabases(
      iv_clusteridentifier = iv_cluster_identifier
      iv_database = iv_database_name
      iv_dbuser = iv_database_user
    ).
    lt_databases = oo_result->get_databases( ).
    lv_db_count = lines( lt_databases ).
    MESSAGE |Retrieved { lv_db_count } database(s).| TYPE 'I'.
  CATCH /aws1/cx_rsddatabaseconnex.
    MESSAGE 'Database connection error.' TYPE 'I'.
  CATCH /aws1/cx_rsdresourcenotfoundex.
    MESSAGE 'Cluster not found.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListDatabases referência da API AWS SDK for SAP ABAP.

`ModifyCluster`

O código de exemplo a seguir mostra como usar `ModifyCluster`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Modificar um cluster.

```
TRY.
    " Example values: iv_cluster_identifier = 'my-redshift-cluster'
    " Example values: iv_pref_maintenance_wn = 'wed:07:30-wed:08:00'
    lo_rsh->modifycluster(
      iv_clusteridentifier = iv_cluster_identifier
      iv_preferredmaintenancewin00 = iv_pref_maintenance_wn
    ).
    MESSAGE 'Redshift cluster modified successfully.' TYPE 'I'.
  CATCH /aws1/cx_rshclustnotfoundfault.
    MESSAGE 'Cluster not found.' TYPE 'I'.
  CATCH /aws1/cx_rshinvcluststatefault.
    MESSAGE 'Invalid cluster state for modification.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ModifyCluster referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Rekognition usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Rekognition.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CompareFaces`

O código de exemplo a seguir mostra como usar `CompareFaces`.

Para obter mais informações, consulte Comparação de faces em imagens.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Create S3 object reference for the source image
    DATA(lo_source_s3obj) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_source_s3_bucket
      iv_name = iv_source_s3_key ).

    " Create source image object
    DATA(lo_source_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_source_s3obj ).

    " Create S3 object reference for the target image
    DATA(lo_target_s3obj) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_target_s3_bucket
      iv_name = iv_target_s3_key ).

    " Create target image object
    DATA(lo_target_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_target_s3obj ).

    " Compare faces
    oo_result = lo_rek->comparefaces(
      io_sourceimage = lo_source_image
      io_targetimage = lo_target_image
      iv_similaritythreshold = iv_similarity ).
```

```
      DATA(lt_face_matches) = oo_result->get_facematches( ).
      DATA(lt_unmatched_faces) = oo_result->get_unmatchedfaces( ).

      " Get counts of matched and unmatched faces
      DATA(lv_matched_count) = lines( lt_face_matches ).
      DATA(lv_unmatched_count) = lines( lt_unmatched_faces ).

      " Output detailed comparison results
      DATA(lv_message) = |Face comparison completed: | &&
                         |{ lv_matched_count } matched face(s), | &&
                         |{ lv_unmatched_count } unmatched face(s).|.
      MESSAGE lv_message TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CompareFaces referência da API AWS SDK for SAP ABAP.

`CreateCollection`

O código de exemplo a seguir mostra como usar `CreateCollection`.

Para obter mais informações, consulte Criar uma coleção.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_rek->createcollection(
        iv_collectionid = iv_collection_id ).
      MESSAGE 'Collection created successfully.' TYPE 'I'.
    CATCH /aws1/cx_rekresrcalrdyexistsex.
```

```
      MESSAGE 'Collection already exists.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateCollection referência da API AWS SDK for SAP ABAP.

`DeleteCollection`

O código de exemplo a seguir mostra como usar `DeleteCollection`.

Para obter mais informações, consulte Excluir uma coleção.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_rek->deletecollection(
        iv_collectionid = iv_collection_id ).
      MESSAGE 'Collection deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_rekresourcenotfoundex.
      MESSAGE 'Collection not found.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteCollection referência da API AWS SDK for SAP ABAP.

`DeleteFaces`

O código de exemplo a seguir mostra como usar `DeleteFaces`.

Para obter mais informações, consulte Excluir faces de uma coleção.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_rek->deletefaces(
      iv_collectionid = iv_collection_id
      it_faceids = it_face_ids ).

    DATA(lt_deleted_faces) = oo_result->get_deletedfaces( ).
    DATA(lv_deleted_count) = lines( lt_deleted_faces ).
    DATA(lv_msg6) = |{ lv_deleted_count } face(s) deleted successfully.|.
    MESSAGE lv_msg6 TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteFaces referência da API AWS SDK for SAP ABAP.

`DescribeCollection`

O código de exemplo a seguir mostra como usar `DescribeCollection`.

Para obter mais informações, consulte Descrever uma coleção.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_rek->describecollection(
      iv_collectionid = iv_collection_id ).
    DATA(lv_face_count) = oo_result->get_facecount( ).
    DATA(lv_msg) = |Collection described: { lv_face_count } face(s) indexed.|.
    MESSAGE lv_msg TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeCollection referência da API AWS SDK for SAP ABAP.

`DetectFaces`

O código de exemplo a seguir mostra como usar `DetectFaces`.

Para obter mais informações, consulte Detectar faces em uma imagem.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
```

```
      " Create S3 object reference for the image
      DATA(lo_s3object) = NEW /aws1/cl_reks3object(
        iv_bucket = iv_s3_bucket
        iv_name = iv_s3_key ).

      " Create image object
      DATA(lo_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_s3object ).

      " Detect faces in the image with all attributes
      DATA(lt_attributes) = VALUE /aws1/cl_rekattributes_w=>tt_attributes( ).
      DATA(lo_attr_wrapper) = NEW /aws1/cl_rekattributes_w( iv_value = 'ALL' ).
      INSERT lo_attr_wrapper INTO TABLE lt_attributes.

      oo_result = lo_rek->detectfaces(
        io_image = lo_image
        it_attributes = lt_attributes ).

      DATA(lt_face_details) = oo_result->get_facedetails( ).
      DATA(lv_detected_count) = lines( lt_face_details ).
      DATA(lv_msg8) = |{ lv_detected_count } face(s) detected in image.|.
      MESSAGE lv_msg8 TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DetectFaces referência da API AWS SDK for SAP ABAP.

`DetectLabels`

O código de exemplo a seguir mostra como usar `DetectLabels`.

Para obter mais informações, consulte Detectar rótulos em uma imagem.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect labels in the image
        oo_result = lo_rek->detectlabels(
          io_image = lo_image
          iv_maxlabels = iv_max_labels ).

        DATA(lt_labels) = oo_result->get_labels( ).
        DATA(lv_label_count) = lines( lt_labels ).
        DATA(lv_msg9) = |{ lv_label_count } label(s) detected in image.|.
        MESSAGE lv_msg9 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetectLabels referência da API AWS SDK for SAP ABAP.

`DetectModerationLabels`

O código de exemplo a seguir mostra como usar `DetectModerationLabels`.

Para obter mais informações, consulte Detectar imagens impróprias.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Detect moderation labels
    oo_result = lo_rek->detectmoderationlabels(
      io_image = lo_image ).

    DATA(lt_moderation_labels) = oo_result->get_moderationlabels( ).
    DATA(lv_mod_count) = lines( lt_moderation_labels ).
    DATA(lv_msg10) = |{ lv_mod_count } moderation label(s) detected.|.
    MESSAGE lv_msg10 TYPE 'I'.
  CATCH /aws1/cx_rekinvalids3objectex.
    MESSAGE 'Invalid S3 object.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectModerationLabels referência da API AWS SDK for SAP ABAP.

`DetectText`

O código de exemplo a seguir mostra como usar `DetectText`.

Para obter mais informações, consulte Detectar texto em uma imagem.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Detect text in the image
    oo_result = lo_rek->detecttext(
      io_image = lo_image ).

    DATA(lt_text_detections) = oo_result->get_textdetections( ).
    DATA(lv_text_count) = lines( lt_text_detections ).
    DATA(lv_msg11) = |{ lv_text_count } text detection(s) found.|.
    MESSAGE lv_msg11 TYPE 'I'.
  CATCH /aws1/cx_rekinvalids3objectex.
    MESSAGE 'Invalid S3 object.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DetectText referência da API AWS SDK for SAP ABAP.

`IndexFaces`

O código de exemplo a seguir mostra como usar `IndexFaces`.

Para obter mais informações, consulte Adicionar faces a uma coleção.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Index faces in the image
        oo_result = lo_rek->indexfaces(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_externalimageid = iv_external_id
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_records) = oo_result->get_facerecords( ).
        DATA(lv_indexed_count) = lines( lt_face_records ).
        DATA(lv_msg2) = |{ lv_indexed_count } face(s) indexed successfully.|.
        MESSAGE lv_msg2 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a IndexFaces referência da API AWS SDK for SAP ABAP.

`ListCollections`

O código de exemplo a seguir mostra como usar `ListCollections`.

Para obter mais informações, consulte Listar coleções.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_rek->listcollections(
      iv_maxresults = iv_max_results ).

    DATA(lt_collection_ids) = oo_result->get_collectionids( ).
    DATA(lv_coll_count) = lines( lt_collection_ids ).
    DATA(lv_msg7) = |{ lv_coll_count } collection(s) found.|.
    MESSAGE lv_msg7 TYPE 'I'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListCollections referência da API AWS SDK for SAP ABAP.

`ListFaces`

O código de exemplo a seguir mostra como usar `ListFaces`.

Para obter mais informações, consulte Listar faces em uma coleção.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_rek->listfaces(
      iv_collectionid = iv_collection_id
      iv_maxresults = iv_max_results ).

    DATA(lt_faces) = oo_result->get_faces( ).
    DATA(lv_face_count2) = lines( lt_faces ).
    DATA(lv_msg3) = |{ lv_face_count2 } face(s) found in collection.|.
    MESSAGE lv_msg3 TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListFaces referência da API AWS SDK for SAP ABAP.

`RecognizeCelebrities`

O código de exemplo a seguir mostra como usar `RecognizeCelebrities`.

Para obter mais informações, consulte Reconhecer celebridades em uma imagem.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Recognize celebrities
        oo_result = lo_rek->recognizecelebrities(
          io_image = lo_image ).

        DATA(lt_celebrity_faces) = oo_result->get_celebrityfaces( ).
        DATA(lv_celeb_count) = lines( lt_celebrity_faces ).
        DATA(lv_msg12) = |{ lv_celeb_count } celebrity/celebrities recognized.|.
        MESSAGE lv_msg12 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a RecognizeCelebrities referência da API AWS SDK for SAP ABAP.

`SearchFaces`

O código de exemplo a seguir mostra como usar `SearchFaces`.

Para obter mais informações, consulte Pesquisar uma face (face ID).

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_rek->searchfaces(
          iv_collectionid = iv_collection_id
          iv_faceid = iv_face_id
          iv_facematchthreshold = iv_threshold
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_matches) = oo_result->get_facematches( ).
        DATA(lv_match_count2) = lines( lt_face_matches ).
        DATA(lv_msg5) = |Face search completed: { lv_match_count2 } match(es)
found.|.
        MESSAGE lv_msg5 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection or face not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SearchFaces referência da API AWS SDK for SAP ABAP.

`SearchFacesByImage`

O código de exemplo a seguir mostra como usar `SearchFacesByImage`.

Para obter mais informações, consulte Pesquisar uma face (imagem).

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).
```

```
        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Search for matching faces
        oo_result = lo_rek->searchfacesbyimage(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_facematchthreshold = iv_threshold
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_matches) = oo_result->get_facematches( ).
        DATA(lv_match_count) = lines( lt_face_matches ).
        DATA(lv_msg4) = |Face search completed: { lv_match_count } match(es)
found.|.
        MESSAGE lv_msg4 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SearchFacesByImage referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon S3 usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    " determine our region from our session
```

```
        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
        DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
        " When in the us-east-1 region, you must not specify a constraint
        " In all other regions, specify the region as the constraint
        IF lv_region = 'us-east-1'.
          CLEAR lo_constraint.
        ELSE.
          lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
        ENDIF.

        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
            io_createbucketconfiguration  = lo_constraint ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.


    "Upload an object to an S3 bucket."
    TRY.
        "Get contents of file from application server."
        DATA lv_file_content TYPE xstring.
        OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
        READ DATASET iv_key INTO lv_file_content.
        CLOSE DATASET iv_key.

        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key
            iv_body = lv_file_content ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.

    " Get an object from a bucket. "
    TRY.
        DATA(lo_result) = lo_s3->getobject(
                    iv_bucket = iv_bucket_name
                    iv_key = iv_key ).
```

```
        DATA(lv_object_data) = lo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " Copy an object to a subfolder in a bucket. "
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_bucket_name
          iv_key = |{ iv_copy_to_folder }/{ iv_key }|
          iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
        MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " List objects in the bucket. "
    TRY.
        DATA(lo_list) = lo_s3->listobjects(
           iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
    DATA text TYPE string VALUE 'Object List - '.
    DATA lv_object_key TYPE /aws1/s3_objectkey.
    LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
      lv_object_key = lo_object->get_key( ).
      CONCATENATE lv_object_key ', ' INTO text.
    ENDLOOP.
    MESSAGE text TYPE'I'.

    " Delete the objects in a bucket. "
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key ).
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
```

```
            iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.


    " Delete the bucket. "
    TRY.
        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_dest_bucket
          iv_key = iv_dest_object
          iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).
        MESSAGE 'Object copied to another bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CopyObject referência da API AWS SDK for SAP ABAP.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " determine our region from our session
        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
```

```
      DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
      " When in the us-east-1 region, you must not specify a constraint
      " In all other regions, specify the region as the constraint
      IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
      ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
      ENDIF.

      lo_s3->createbucket(
          iv_bucket = iv_bucket_name
          io_createbucketconfiguration  = lo_constraint ).
      MESSAGE 'S3 bucket created.' TYPE 'I'.
    CATCH /aws1/cx_s3_bucketalrdyexists.
      MESSAGE 'Bucket name already exists.' TYPE 'E'.
    CATCH /aws1/cx_s3_bktalrdyownedbyyou.
      MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateBucket referência da API AWS SDK for SAP ABAP.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.

      lo_s3->deletebucket(
          iv_bucket = iv_bucket_name ).
      MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
```

```
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteBucket referência da API AWS SDK for SAP ABAP.

`DeleteBucketCors`

O código de exemplo a seguir mostra como usar `DeleteBucketCors`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_s3->deletebucketcors(
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteBucketCors referência da API AWS SDK for SAP ABAP.

`DeleteBucketLifecycle`

O código de exemplo a seguir mostra como usar `DeleteBucketLifecycle`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_s3->deletebucketlifecycle(
      iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteBucketLifecycle referência da API AWS SDK for SAP ABAP.

`DeleteBucketPolicy`

O código de exemplo a seguir mostra como usar `DeleteBucketPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_s3->deletebucketpolicy(
      iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket policy deleted.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteBucketPolicy referência da API AWS SDK for SAP ABAP.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteObject referência da API AWS SDK for SAP ABAP.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->deleteobjects(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
          io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ) ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteObjects referência da API AWS SDK for SAP ABAP.

`GetBucketAcl`

O código de exemplo a seguir mostra como usar `GetBucketAcl`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->getbucketacl(         " oo_result is returned for testing
purposes. "
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetBucketAcl referência da API AWS SDK for SAP ABAP.

`GetBucketCors`

O código de exemplo a seguir mostra como usar `GetBucketCors`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getbucketcors(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetBucketCors referência da API AWS SDK for SAP ABAP.

`GetBucketLifecycleConfiguration`

O código de exemplo a seguir mostra como usar `GetBucketLifecycleConfiguration`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getbucketlifecycleconf(         " oo_result is returned
for testing purposes. "
```

```
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetBucketLifecycleConfiguration referência da API AWS SDK for SAP ABAP.

`GetBucketPolicy`

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->getbucketpolicy(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name ).
        DATA(lv_policy) = oo_result->get_policy( ).
        MESSAGE 'Retrieved bucket policy.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetBucketPolicy referência da API AWS SDK for SAP ABAP.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getobject(           " oo_result is returned for testing
purposes. "
                 iv_bucket = iv_bucket_name
                 iv_key = iv_object_key ).
       DATA(lv_object_data) = oo_result->get_body( ).
       MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetObject referência da API AWS SDK for SAP ABAP.

`GetObjectAcl`

O código de exemplo a seguir mostra como usar `GetObjectAcl`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getobjectacl(         " oo_result is returned for testing
purposes. "
```

```
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key ).
        MESSAGE 'Retrieved object ACL.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetObjectAcl referência da API AWS SDK for SAP ABAP.

`GetObjectLegalHold`

O código de exemplo a seguir mostra como usar `GetObjectLegalHold`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->getobjectlegalhold(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key ).
        MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetObjectLegalHold referência da API AWS SDK for SAP ABAP.

`GetObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `GetObjectLockConfiguration`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->getobjectlockconfiguration(         " oo_result is
returned for testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetObjectLockConfiguration referência da API AWS SDK for SAP ABAP.

`HeadBucket`

O código de exemplo a seguir mostra como usar `HeadBucket`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->headbucket(         " oo_result is returned for testing
purposes. "
```

```
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a HeadBucket referência da API AWS SDK for SAP ABAP.

`ListBuckets`

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3->listbuckets(         " oo_result is returned for testing
purposes. "
        ).
        DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).
        MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list buckets.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListBuckets referência da API AWS SDK for SAP ABAP.

`ListObjectVersions`

O código de exemplo a seguir mostra como usar `ListObjectVersions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->listobjectversions(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name
         iv_prefix = iv_prefix ).
       MESSAGE 'Retrieved object versions.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListObjectVersions referência da API AWS SDK for SAP ABAP.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3->listobjectsv2(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
```

```
    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte ListObjects V2 no AWS SDK para referência da API SAP ABAP.

PutBucketAcl

O código de exemplo a seguir mostra como usar `PutBucketAcl`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " Example: Grant log delivery access to a bucket
      " iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'
      lo_s3->putbucketacl(
        iv_bucket = iv_bucket_name
        iv_grantwrite = iv_grantwrite ).
      MESSAGE 'Bucket ACL updated.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketAcl referência da API AWS SDK for SAP ABAP.

PutBucketCors

O código de exemplo a seguir mostra como usar `PutBucketCors`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example: Allow PUT, POST, DELETE methods from http://www.example.com
    lo_s3->putbucketcors(
      iv_bucket = iv_bucket_name
      io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(
        it_corsrules = it_cors_rules ) ).
    MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketCors referência da API AWS SDK for SAP ABAP.

`PutBucketLifecycleConfiguration`

O código de exemplo a seguir mostra como usar `PutBucketLifecycleConfiguration`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example: Expire objects with prefix 'logs/' after 30 days
    lo_s3->putbucketlifecycleconf(
      iv_bucket = iv_bucket_name
```

```
          io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
            it_rules = it_lifecycle_rule ) ).
        MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketLifecycleConfiguration referência da API AWS SDK for SAP ABAP.

`PutBucketPolicy`

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example policy JSON string
        " iv_policy = '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::123456789012:user/
user"},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::bucketname/*"]}]}'
        lo_s3->putbucketpolicy(
          iv_bucket = iv_bucket_name
          iv_policy = iv_policy ).
        MESSAGE 'Bucket policy set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketPolicy referência da API AWS SDK for SAP ABAP.

`PutBucketVersioning`

O código de exemplo a seguir mostra como usar `PutBucketVersioning`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example: Enable versioning on a bucket
    " iv_status = 'Enabled'
    lo_s3->putbucketversioning(
      iv_bucket = iv_bucket_name
      io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(
        iv_status = iv_status ) ).
    MESSAGE 'Bucket versioning enabled.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutBucketVersioning referência da API AWS SDK for SAP ABAP.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
"Get contents of file from application server."
DATA lv_body TYPE xstring.
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
READ DATASET iv_file_name INTO lv_body.
CLOSE DATASET iv_file_name.

"Upload/put an object to an S3 bucket."
TRY.
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_file_name
        iv_body = lv_body ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutObject referência da API AWS SDK for SAP ABAP.

`PutObjectAcl`

O código de exemplo a seguir mostra como usar `PutObjectAcl`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example: Grant read access to an AWS user
    " iv_grantread = 'emailAddress=user@example.com'
    lo_s3->putobjectacl(
      iv_bucket = iv_bucket_name
      iv_key = iv_object_key
      iv_grantread = iv_grantread ).
```

```
      MESSAGE 'Object ACL updated.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
      MESSAGE 'Object key does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectAcl referência da API AWS SDK for SAP ABAP.

`PutObjectLegalHold`

O código de exemplo a seguir mostra como usar `PutObjectLegalHold`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " Example: Set legal hold status to ON
      " iv_status = 'ON'
      lo_s3->putobjectlegalhold(
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key
        io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(
          iv_status = iv_status ) ).
      MESSAGE 'Object legal hold status set.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
      MESSAGE 'Object key does not exist.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectLegalHold referência da API AWS SDK for SAP ABAP.

`PutObjectLockConfiguration`

O código de exemplo a seguir mostra como usar `PutObjectLockConfiguration`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " Example: Enable object lock with default retention
    " iv_enabled = 'Enabled'
    lo_s3->putobjectlockconfiguration(
      iv_bucket = iv_bucket_name
      io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(
        iv_objectlockenabled = iv_enabled ) ).
    MESSAGE 'Object lock configuration set.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectLockConfiguration referência da API AWS SDK for SAP ABAP.

`PutObjectRetention`

O código de exemplo a seguir mostra como usar `PutObjectRetention`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Example: Set retention mode to GOVERNANCE for 30 days
        " iv_mode = 'GOVERNANCE'
        " iv_retain_date should be a timestamp in the future
        lo_s3->putobjectretention(
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key
          io_retention = NEW /aws1/cl_s3_objectlockret(
            iv_mode = iv_mode
            iv_retainuntildate = iv_retain_date )
          iv_bypassgovernanceretention = abap_true ).
        MESSAGE 'Object retention set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a PutObjectRetention referência da API AWS SDK for SAP ABAP.

Cenários

Criar um URL pré-assinado

O exemplo de código a seguir mostra como criar um URL pré-assinado para o Amazon S3 e fazer upload de um objeto.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie solicitações pré-assinadas para objetos GET do S3.

```
    " iv_bucket_name is the bucket name
    " iv_key is the object name like "myfile.txt"
```

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
    DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call.
" to the service.  It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
  iv_bucket = iv_bucket_name
  iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file.  The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).
```

Exemplos de controle do Amazon S3 usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon S3 Control.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_manifest_arn  = 'arn:aws:s3:::my-bucket/job-manifest.csv'
        " iv_manifest_etag = 'abc123def456'
        " iv_report_bucket = 'arn:aws:s3:::my-report-bucket'
        DATA(lo_result) = lo_s3c->createjob(
          iv_accountid            = iv_account_id
          iv_rolearn              = iv_role_arn
          iv_confirmationrequired = abap_true
          iv_priority             = 10
          iv_description          = 'Batch job for tagging objects'
          io_operation            = NEW /aws1/cl_s3cjoboperation(
            io_s3putobjecttagging = NEW /aws1/cl_s3cs3setobjecttagop(
              it_tagset = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
                ( NEW /aws1/cl_s3cs3tag(
                    iv_key   = 'BatchTag'
                    iv_value = 'BatchValue' ) )
              )
            )
          )
          io_manifest             = NEW /aws1/cl_s3cjobmanifest(
            io_spec     = NEW /aws1/cl_s3cjobmanifestspec(
              iv_format = 'S3BatchOperations_CSV_20180820'
              it_fields = VALUE /aws1/
cl_s3cjobmanifestfield00=>tt_jobmanifestfieldlist(
```

```
                ( NEW /aws1/cl_s3cjobmanifestfield00( 'Bucket' ) )
                ( NEW /aws1/cl_s3cjobmanifestfield00( 'Key' ) )
              )
            )
            io_location = NEW /aws1/cl_s3cjobmanifestloc(
              iv_objectarn = iv_manifest_arn
              iv_etag      = iv_manifest_etag
            )
          )
          io_report               = NEW /aws1/cl_s3cjobreport(
            iv_bucket      = iv_report_bucket
            iv_format      = 'Report_CSV_20180820'
            iv_enabled     = abap_true
            iv_prefix      = 'batch-op-reports'
            iv_reportscope = 'AllTasks'
          )
        ).
        ov_job_id = lo_result->get_jobid( ).
        MESSAGE |S3 Batch Operations job created: { ov_job_id }| TYPE 'I'.
      CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
        MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_bad.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateJob referência da API AWS SDK for SAP ABAP.

DeleteJobTagging

O código de exemplo a seguir mostra como usar `DeleteJobTagging`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_s3c->deletejobtagging(
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
        ).
        MESSAGE |Tags deleted from job { iv_job_id }| TYPE 'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteJobTagging referência da API AWS SDK for SAP ABAP.

`DescribeJob`

O código de exemplo a seguir mostra como usar `DescribeJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3c->describejob(         " oo_result is returned for testing
purposes.
         iv_accountid = iv_account_id
         iv_jobid     = iv_job_id
       ).
       DATA(lo_job) = oo_result->get_job( ).
       DATA(lv_status) = lo_job->get_status( ).
       DATA(lv_priority) = lo_job->get_priority( ).
       DATA(lo_progress) = lo_job->get_progresssummary( ).
       IF lo_progress IS NOT INITIAL.
         MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }, | &&
                 |total={ lo_progress->get_totalnumberoftasks( ) }, | &&
                 |succeeded={ lo_progress->get_numberoftaskssucceeded( ) }, | &&
                 |failed={ lo_progress->get_numberoftasksfailed( ) }| TYPE 'I'.
       ELSE.
         MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }| TYPE 'I'.
       ENDIF.
     CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
       MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_nf.
     CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
       MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_cli.
     CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
       MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_srv.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeJob referência da API AWS SDK for SAP ABAP.

`GetJobTagging`

O código de exemplo a seguir mostra como usar `GetJobTagging`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_s3c->getjobtagging(       " oo_result is returned for testing
purposes.
         iv_accountid = iv_account_id
         iv_jobid     = iv_job_id
       ).
       DATA(lt_tags) = oo_result->get_tags( ).
       MESSAGE |Retrieved { lines( lt_tags ) } tag(s) for job { iv_job_id }| TYPE
'I'.
     CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
       MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_nf.
     CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
       MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_cli.
     CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
       MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_srv.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetJobTagging referência da API AWS SDK for SAP ABAP.

`PutJobTagging`

O código de exemplo a seguir mostra como usar `PutJobTagging`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_s3c->putjobtagging(
      iv_accountid = iv_account_id
      iv_jobid     = iv_job_id
      it_tags      = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
        ( NEW /aws1/cl_s3cs3tag(
            iv_key   = 'Environment'
            iv_value = 'Development' ) )
        ( NEW /aws1/cl_s3cs3tag(
            iv_key   = 'Team'
            iv_value = 'DataProcessing' ) )
      )
    ).
    MESSAGE |Tags added to job { iv_job_id }| TYPE 'I'.
  CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_nf.
  CATCH /aws1/cx_s3ctoomanytagsex INTO DATA(lo_ex_tags).
    MESSAGE lo_ex_tags->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_tags.
  CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_cli.
  CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_srv.
ENDTRY.
```

- Para obter detalhes da API, consulte a PutJobTagging referência da API AWS SDK for SAP ABAP.

`UpdateJobPriority`

O código de exemplo a seguir mostra como usar `UpdateJobPriority`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_s3c->updatejobpriority(   " oo_result is returned for testing
 purposes.
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
          iv_priority  = 60
        ).
        MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
        MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_bad.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
```

```
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateJobPriority referência da API AWS SDK for SAP ABAP.

`UpdateJobStatus`

O código de exemplo a seguir mostra como usar `UpdateJobStatus`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_requested_status = 'Cancelled'
        oo_result = lo_s3c->updatejobstatus(     " oo_result is returned for testing
 purposes.
          iv_accountid          = iv_account_id
          iv_jobid              = iv_job_id
          iv_requestedjobstatus = iv_requested_status
        ).
        MESSAGE |Job { oo_result->get_jobid( ) } status updated to { oo_result-
>get_status( ) }| TYPE 'I'.
      CATCH /aws1/cx_s3cjobstatusexception INTO DATA(lo_ex_js).
        MESSAGE lo_ex_js->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_js.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
```

```
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateJobStatus referência da API AWS SDK for SAP ABAP.

SageMaker Exemplos de IA usando SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com SageMaker IA.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateEndpoint`

O código de exemplo a seguir mostra como usar `CreateEndpoint`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
    DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
    DATA oo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.

    "Create a production variant as an ABAP object."
    "Identifies a model that you want to host and the resources chosen to deploy for
 hosting it."
    lo_production_variants = NEW #( iv_variantname = iv_variant_name
                                    iv_modelname = iv_model_name
                                    iv_initialinstancecount =
 iv_initial_instance_count
                                    iv_instancetype = iv_instance_type ).

    INSERT lo_production_variants INTO TABLE lt_production_variants.

    "Create an endpoint configuration."
    TRY.
        oo_ep_config_result = lo_sgm->createendpointconfig(
          iv_endpointconfigname = iv_endpoint_config_name
          it_productionvariants = lt_production_variants ).
        MESSAGE 'Endpoint configuration created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.

    "Create an endpoint."
    TRY.
        oo_result = lo_sgm->createendpoint(     " oo_result is returned for testing
 purposes. "
            iv_endpointconfigname = iv_endpoint_config_name
            iv_endpointname = iv_endpoint_name ).
        MESSAGE 'Endpoint created.' TYPE 'I'.
```

```
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateEndpoint referência da API AWS SDK for SAP ABAP.

`CreateModel`

O código de exemplo a seguir mostra como usar `CreateModel`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.

    "Create an ABAP object for the container image based on input variables."
    lo_primarycontainer = NEW #( iv_image = iv_container_image
                                 iv_modeldataurl = iv_model_data_url ).

    "Create an Amazon SageMaker model."
    TRY.
        oo_result = lo_sgm->createmodel(        " oo_result is returned for testing
purposes. "
          iv_executionrolearn = iv_execution_role_arn
          iv_modelname = iv_model_name
          io_primarycontainer = lo_primarycontainer ).
        MESSAGE 'Model created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateModel referência da API AWS SDK for SAP ABAP.

`CreateTrainingJob`

O código de exemplo a seguir mostra como usar `CreateTrainingJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
   DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
   DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
   DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
   DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
   DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
   DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
   DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
   DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
   DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
   DATA lo_resource_config  TYPE REF TO /aws1/cl_sgmresourceconfig.
   DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
   DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.

   "Create ABAP internal table for hyperparameters based on input variables."
   "These hyperparameters are based on the Amazon SageMaker built-in algorithm,
XGBoost."
   lo_hyperparameters_w = NEW #( iv_value = iv_hp_max_depth ).
   INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eta ).
   INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eval_metric ).
```

```
   INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_scale_pos_weight ).
   INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_subsample ).
   INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_objective ).
   INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_num_round ).
   INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   "Create ABAP objects for training data sources."
   lo_trn_s3datasource = NEW #( iv_s3datatype = iv_trn_data_s3datatype
                                iv_s3datadistributiontype =
iv_trn_data_s3datadistribution
                                iv_s3uri = iv_trn_data_s3uri ).

   lo_trn_datasource = NEW #( io_s3datasource = lo_trn_s3datasource ).

   lo_trn_channel = NEW #( iv_channelname = 'train'
                           io_datasource = lo_trn_datasource
                           iv_compressiontype = iv_trn_data_compressiontype
                           iv_contenttype = iv_trn_data_contenttype ).

   INSERT lo_trn_channel INTO TABLE lt_input_data_config.

   "Create ABAP objects for validation data sources."
   lo_val_s3datasource = NEW #( iv_s3datatype = iv_val_data_s3datatype
                                iv_s3datadistributiontype =
iv_val_data_s3datadistribution
                                iv_s3uri = iv_val_data_s3uri ).

   lo_val_datasource = NEW #( io_s3datasource = lo_val_s3datasource ).

   lo_val_channel = NEW #( iv_channelname = 'validation'
                           io_datasource = lo_val_datasource
```

```
                            iv_compressiontype = iv_val_data_compressiontype
                            iv_contenttype = iv_val_data_contenttype ).

   INSERT lo_val_channel INTO TABLE lt_input_data_config.

   "Create an ABAP object for algorithm specification."
   lo_algorithm_specification = NEW #( iv_trainingimage = iv_training_image
                                       iv_traininginputmode =
iv_training_input_mode ).

   "Create an ABAP object for resource configuration."
   lo_resource_config = NEW #( iv_instancecount = iv_instance_count
                               iv_instancetype = iv_instance_type
                               iv_volumesizeingb = iv_volume_sizeingb ).

   "Create an ABAP object for output data configuration."
   lo_output_data_config = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create an ABAP object for stopping condition."
   lo_stopping_condition = NEW #( iv_maxruntimeinseconds =
iv_max_runtime_in_seconds ).

   "Create a training job."
   TRY.
       oo_result = lo_sgm->createtrainingjob(    " oo_result is returned for
testing purposes. "
         iv_trainingjobname           = iv_training_job_name
         iv_rolearn                   = iv_role_arn
         it_hyperparameters           = lt_hyperparameters
         it_inputdataconfig           = lt_input_data_config
         io_algorithmspecification    = lo_algorithm_specification
         io_outputdataconfig          = lo_output_data_config
         io_resourceconfig            = lo_resource_config
         io_stoppingcondition         = lo_stopping_condition ).
       MESSAGE 'Training job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTrainingJob referência da API AWS SDK for SAP ABAP.

`CreateTransformJob`

O código de exemplo a seguir mostra como usar `CreateTransformJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   DATA lo_transforminput TYPE REF TO /aws1/cl_sgmtransforminput.
   DATA lo_transformoutput TYPE REF TO /aws1/cl_sgmtransformoutput.
   DATA lo_transformresources TYPE REF TO /aws1/cl_sgmtransformresources.
   DATA lo_datasource  TYPE REF TO /aws1/cl_sgmtransformdatasrc.
   DATA lo_s3datasource  TYPE REF TO /aws1/cl_sgmtransforms3datasrc.

   "Create an ABAP object for an Amazon Simple Storage Service (Amazon S3) data
source."
   lo_s3datasource = NEW #( iv_s3uri = iv_tf_data_s3uri
                            iv_s3datatype = iv_tf_data_s3datatype ).

   "Create an ABAP object for data source."
   lo_datasource = NEW #( io_s3datasource = lo_s3datasource ).

   "Create an ABAP object for transform data source."
   lo_transforminput = NEW #( io_datasource = lo_datasource
                              iv_contenttype = iv_tf_data_contenttype
                              iv_compressiontype = iv_tf_data_compressiontype ).

   "Create an ABAP object for resource configuration."
   lo_transformresources = NEW #( iv_instancecount = iv_instance_count
                                  iv_instancetype = iv_instance_type ).

   "Create an ABAP object for output data configuration."
   lo_transformoutput = NEW #( iv_s3outputpath = iv_s3_output_path ).
```

```
    "Create a transform job."
    TRY.
        oo_result = lo_sgm->createtransformjob(     " oo_result is returned for
testing purposes. "
            iv_modelname = iv_tf_model_name
            iv_transformjobname = iv_tf_job_name
            io_transforminput = lo_transforminput
            io_transformoutput = lo_transformoutput
            io_transformresources = lo_transformresources ).
        MESSAGE 'Transform job created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourceinuse.
        MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcenotfound.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTransformJob referência da API AWS SDK for SAP ABAP.

`DeleteEndpoint`

O código de exemplo a seguir mostra como usar `DeleteEndpoint`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Delete an endpoint."
    TRY.
        lo_sgm->deleteendpoint(
            iv_endpointname = iv_endpoint_name ).
        MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpoint_exception).
```

```
        DATA(lv_endpoint_error) = |"{ lo_endpoint_exception->av_err_code }" -
 { lo_endpoint_exception->av_err_msg }|.
        MESSAGE lv_endpoint_error TYPE 'E'.
    ENDTRY.

    "Delete an endpoint configuration."
    TRY.
        lo_sgm->deleteendpointconfig(
          iv_endpointconfigname = iv_endpoint_config_name ).
        MESSAGE 'Endpoint deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteEndpoint referência da API AWS SDK for SAP ABAP.

`DeleteModel`

O código de exemplo a seguir mostra como usar `DeleteModel`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sgm->deletemodel(
                  iv_modelname = iv_model_name ).
        MESSAGE 'Model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteModel referência da API AWS SDK for SAP ABAP.

DescribeTrainingJob

O código de exemplo a seguir mostra como usar `DescribeTrainingJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->describetrainingjob(      " oo_result is returned for
 testing purposes. "
          iv_trainingjobname = iv_training_job_name ).
        MESSAGE 'Retrieved description of training job.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeTrainingJob referência da API AWS SDK for SAP ABAP.

ListAlgorithms

O código de exemplo a seguir mostra como usar `ListAlgorithms`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->listalgorithms(         " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of algorithms.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListAlgorithms referência da API AWS SDK for SAP ABAP.

`ListModels`

O código de exemplo a seguir mostra como usar `ListModels`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->listmodels(           " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of models.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListModels referência da API AWS SDK for SAP ABAP.

`ListNotebookInstances`

O código de exemplo a seguir mostra como usar `ListNotebookInstances`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->listnotebookinstances(        " oo_result is returned
 for testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of notebook instances.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListNotebookInstances referência da API AWS SDK for SAP ABAP.

`ListTrainingJobs`

O código de exemplo a seguir mostra como usar `ListTrainingJobs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sgm->listtrainingjobs(       " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains
          iv_maxresults = iv_max_results ).
        MESSAGE 'Retrieved list of training jobs.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTrainingJobs referência da API AWS SDK for SAP ABAP.

Cenários

Conceitos básicos de usar modelos e endpoints

O exemplo de código a seguir mostra como:

- Comece um trabalho de treinamento e crie um modelo de SageMaker IA.
- Criar uma configuração de endpoint.
- Criar um endpoint e, em seguida, limpar os recursos.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
    DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
    DATA lo_resource_config  TYPE REF TO /aws1/cl_sgmresourceconfig.
    DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
    DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.
    DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.
    DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
    DATA lo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.
    DATA lo_training_result TYPE REF TO /aws1/cl_sgmdescrtrnjobrsp.
    DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
    DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
    DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
    DATA lv_model_data_url TYPE /aws1/sgmurl.

    lv_model_data_url = iv_s3_output_path && iv_training_job_name && '/output/
model.tar.gz'.

    "Create ABAP internal table for hyperparameters based on input variables."
    "These hyperparameters are based on Amazon SageMaker built-in algorithm -
 XGBoost"
    lo_hyperparameters_w = NEW #( iv_value = iv_hp_max_depth ).
    INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_eta ).
```

```
   INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eval_metric ).
   INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_scale_pos_weight ).
   INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_subsample ).
   INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_objective ).
   INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_num_round ).
   INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   "Create ABAP internal table for data based on input variables."
   "Training data."
   lo_trn_s3datasource = NEW #( iv_s3datatype = iv_trn_data_s3datatype
                                iv_s3datadistributiontype =
iv_trn_data_s3datadistribution
                                iv_s3uri = iv_trn_data_s3uri ).

   lo_trn_datasource = NEW #( io_s3datasource = lo_trn_s3datasource ).

   lo_trn_channel = NEW #( iv_channelname = 'train'
                           io_datasource = lo_trn_datasource
                           iv_compressiontype = iv_trn_data_compressiontype
                           iv_contenttype = iv_trn_data_contenttype ).
   INSERT lo_trn_channel INTO TABLE lt_input_data_config.

   "Validation data."
   lo_val_s3datasource = NEW #( iv_s3datatype = iv_val_data_s3datatype
                                iv_s3datadistributiontype =
iv_val_data_s3datadistribution
                                iv_s3uri = iv_val_data_s3uri ).
```

```
   lo_val_datasource = NEW #( io_s3datasource = lo_val_s3datasource ).

   lo_val_channel = NEW #( iv_channelname = 'validation'
                           io_datasource = lo_val_datasource
                           iv_compressiontype = iv_val_data_compressiontype
                           iv_contenttype = iv_val_data_contenttype ).
   INSERT lo_val_channel INTO TABLE lt_input_data_config.

   "Create an ABAP object for algorithm specification based on input variables."
   lo_algorithm_specification = NEW #( iv_trainingimage = iv_training_image
                                       iv_traininginputmode =
iv_training_input_mode ).

   "Create an ABAP object for resource configuration."
   lo_resource_config = NEW #( iv_instancecount = iv_instance_count
                               iv_instancetype = iv_instance_type
                               iv_volumesizeingb = iv_volume_sizeingb ).

   "Create an ABAP object for output data configuration."
   lo_output_data_config = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create an ABAP object for stopping condition."
   lo_stopping_condition = NEW #( iv_maxruntimeinseconds =
iv_max_runtime_in_seconds ).

   TRY.
       lo_sgm->createtrainingjob(
         iv_trainingjobname           = iv_training_job_name
         iv_rolearn                   = iv_role_arn
         it_hyperparameters           = lt_hyperparameters
         it_inputdataconfig           = lt_input_data_config
         io_algorithmspecification    = lo_algorithm_specification
         io_outputdataconfig          = lo_output_data_config
         io_resourceconfig            = lo_resource_config
         io_stoppingcondition         = lo_stopping_condition ).
       MESSAGE 'Training job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
   ENDTRY.
```

```
   "Wait for training job to be completed."
   lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
   WHILE lo_training_result->get_trainingjobstatus( ) <> 'Completed'.
     IF sy-index = 30.
       EXIT.               "Maximum 900 seconds."
     ENDIF.
     WAIT UP TO 30 SECONDS.
     lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
   ENDWHILE.

   "Create ABAP object for the container image based on input variables."
   lo_primarycontainer = NEW #( iv_image = iv_training_image
                                iv_modeldataurl = lv_model_data_url ).

   "Create an Amazon SageMaker model."
   TRY.
       lo_sgm->createmodel(
         iv_executionrolearn = iv_role_arn
         iv_modelname = iv_model_name
         io_primarycontainer = lo_primarycontainer ).
       MESSAGE 'Model created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
   ENDTRY.

   "Create an endpoint production variant."
   lo_production_variants = NEW #( iv_variantname = iv_ep_variant_name
                                   iv_modelname = iv_model_name
                                   iv_initialinstancecount =
iv_ep_initial_instance_count
                                   iv_instancetype = iv_ep_instance_type ).
   INSERT lo_production_variants INTO TABLE lt_production_variants.

   TRY.
       "Create an endpoint configuration."
       lo_ep_config_result = lo_sgm->createendpointconfig(
         iv_endpointconfigname = iv_ep_cfg_name
         it_productionvariants = lt_production_variants ).
       MESSAGE 'Endpoint configuration created.' TYPE 'I'.

       "Create an endpoint."
```

```
        oo_ep_output = lo_sgm->createendpoint(        " oo_ep_output is returned for
 testing purposes. "
            iv_endpointconfigname = iv_ep_cfg_name
            iv_endpointname = iv_ep_name ).
        MESSAGE 'Endpoint created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.

    "Wait for endpoint creation to be completed."
    DATA(lo_endpoint_result) = lo_sgm->describeendpoint( iv_endpointname =
 iv_ep_name ).
    WHILE lo_endpoint_result->get_endpointstatus( ) <> 'InService'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_endpoint_result = lo_sgm->describeendpoint( iv_endpointname = iv_ep_name ).
    ENDWHILE.

    TRY.
        "Delete an endpoint."
        lo_sgm->deleteendpoint(
            iv_endpointname = iv_ep_name ).
        MESSAGE 'Endpoint deleted' TYPE 'I'.

        "Delete an endpoint configuration."
        lo_sgm->deleteendpointconfig(
          iv_endpointconfigname = iv_ep_cfg_name ).
        MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.

        "Delete model."
        lo_sgm->deletemodel(
                  iv_modelname = iv_model_name ).
        MESSAGE 'Model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - CreateEndpoint
  - CreateEndpointConfig
  - CreateModel
  - CreateTrainingJob
  - DeleteEndpoint
  - DeleteEndpointConfig
  - DeleteModel
  - DescribeEndpoint
  - DescribeTrainingJob

Exemplos do Secrets Manager usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Secrets Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`BatchGetSecretValue`

O código de exemplo a seguir mostra como usar `BatchGetSecretValue`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " iv_filter_name = 'mySecret'
        DATA(lo_result) = lo_smr->batchgetsecretvalue(
          it_filters = VALUE /aws1/cl_smrfilter=>tt_filterslisttype(
            (
              NEW /aws1/cl_smrfilter(
                iv_key = 'name'
                it_values = VALUE /aws1/
cl_smrfiltvalsstrlist_w=>tt_filtervaluesstringlist(
                  ( NEW /aws1/cl_smrfiltvalsstrlist_w( iv_value = iv_filter_name ) )
                )
              )
            )
          )
        ).
        ot_secret_values = lo_result->get_secretvalues( ).
        MESSAGE 'Secrets retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_smrresourcenotfoundex.
        MESSAGE 'One or more requested secrets were not found.' TYPE 'E'.
      CATCH /aws1/cx_smrdecryptionfailure.
        MESSAGE 'Failed to decrypt one or more secrets.' TYPE 'E'.
      CATCH /aws1/cx_smrinvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'E'.
      CATCH /aws1/cx_smrinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a BatchGetSecretValue referência da API AWS SDK for SAP ABAP.

`GetSecretValue`

O código de exemplo a seguir mostra como usar `GetSecretValue`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    " iv_secret_name = 'MySecretName'
    DATA(lo_result) = lo_smr->getsecretvalue( iv_secretid = iv_secret_name ).
    ov_secret_value = lo_result->get_secretstring( ).
    MESSAGE 'Secret value retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_smrresourcenotfoundex.
    MESSAGE 'The requested secret was not found.' TYPE 'E'.
  CATCH /aws1/cx_smrdecryptionfailure.
    MESSAGE 'Failed to decrypt the secret.' TYPE 'E'.
  CATCH /aws1/cx_smrinvalidparameterex.
    MESSAGE 'Invalid parameter provided.' TYPE 'E'.
  CATCH /aws1/cx_smrinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetSecretValue referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon SES usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon SES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateReceiptFilter`

O código de exemplo a seguir mostra como usar `CreateReceiptFilter`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
" iv_allow = abap_true means 'Allow', abap_false means 'Block'
DATA(lv_policy) = COND /aws1/sesreceiptfilterpolicy(
  WHEN iv_allow = abap_true THEN 'Allow'
  ELSE 'Block'
).

DATA(lo_ip_filter) = NEW /aws1/cl_sesreceiptipfilter(
  iv_policy = lv_policy
  iv_cidr = iv_ip_address_or_range
).

DATA(lo_filter) = NEW /aws1/cl_sesreceiptfilter(
  iv_name = iv_filter_name
  io_ipfilter = lo_ip_filter
).

TRY.
    lo_ses->createreceiptfilter( io_filter = lo_filter ).
    MESSAGE 'Receipt filter created successfully' TYPE 'I'.
  CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
    DATA(lv_error) = |Filter already exists: { lo_ex1->get_text( ) }|.
```

```
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateReceiptFilter referência da API AWS SDK for SAP ABAP.

`CreateReceiptRule`

O código de exemplo a seguir mostra como usar `CreateReceiptRule`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  " Create S3 action for copying emails to S3
  DATA(lo_s3_action) = NEW /aws1/cl_sess3action(
    iv_bucketname = iv_bucket_name
    iv_objectkeyprefix = iv_prefix
  ).

  " Create receipt action with S3 action
  DATA(lo_action) = NEW /aws1/cl_sesreceiptaction(
    io_s3action = lo_s3_action
  ).

  " Create list of actions
  DATA lt_actions TYPE /aws1/cl_sesreceiptaction=>tt_receiptactionslist.
  APPEND lo_action TO lt_actions.

  " Create receipt rule
  DATA(lo_rule) = NEW /aws1/cl_sesreceiptrule(
```

```
      iv_name = iv_rule_name
      iv_enabled = abap_true
      it_recipients = it_recipients
      it_actions = lt_actions
    ).

    TRY.
        lo_ses->createreceiptrule(
          iv_rulesetname = iv_rule_set_name
          io_rule = lo_rule
        ).
        MESSAGE 'Receipt rule created successfully' TYPE 'I'.
      CATCH /aws1/cx_sesinvalids3confex INTO DATA(lo_ex1).
        DATA(lv_error) = |Invalid S3 configuration: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateReceiptRule referência da API AWS SDK for SAP ABAP.

`CreateReceiptRuleSet`

O código de exemplo a seguir mostra como usar `CreateReceiptRuleSet`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ses->createreceiptruleset( iv_rulesetname = iv_rule_set_name ).
        MESSAGE 'Receipt rule set created successfully' TYPE 'I'.
```

```
    CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
      DATA(lv_error) = |Rule set already exists: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateReceiptRuleSet referência da API AWS SDK for SAP ABAP.

`CreateTemplate`

O código de exemplo a seguir mostra como usar `CreateTemplate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  DATA(lo_template) = NEW /aws1/cl_sestemplate(
    iv_templatename = iv_name
    iv_subjectpart = iv_subject
    iv_textpart = iv_text
    iv_htmlpart = iv_html
  ).

  TRY.
      lo_ses->createtemplate( io_template = lo_template ).
      MESSAGE 'Template created successfully' TYPE 'I'.
    CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
      DATA(lv_error) = |Template already exists: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_sesinvalidtemplateex INTO DATA(lo_ex2).
```

```
      lv_error = |Invalid template: { lo_ex2->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex2.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTemplate referência da API AWS SDK for SAP ABAP.

`DeleteIdentity`

O código de exemplo a seguir mostra como usar `DeleteIdentity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_ses->deleteidentity( iv_identity = iv_identity ).
      MESSAGE 'Identity deleted successfully' TYPE 'I'.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
      DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteIdentity referência da API AWS SDK for SAP ABAP.

`DeleteReceiptFilter`

O código de exemplo a seguir mostra como usar `DeleteReceiptFilter`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ses->deletereceiptfilter( iv_filtername = iv_filter_name ).
    MESSAGE 'Receipt filter deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteReceiptFilter referência da API AWS SDK for SAP ABAP.

`DeleteReceiptRule`

O código de exemplo a seguir mostra como usar `DeleteReceiptRule`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ses->deletereceiptrule(
      iv_rulesetname = iv_rule_set_name
```

```
        iv_rulename = iv_rule_name
      ).
      MESSAGE 'Receipt rule deleted successfully' TYPE 'I'.
    CATCH /aws1/cx_sesrulesetdoesnotexex INTO DATA(lo_ex1).
      DATA(lv_error) = |Rule set does not exist: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteReceiptRule referência da API AWS SDK for SAP ABAP.

`DeleteReceiptRuleSet`

O código de exemplo a seguir mostra como usar `DeleteReceiptRuleSet`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_ses->deletereceiptruleset( iv_rulesetname = iv_rule_set_name ).
      MESSAGE 'Receipt rule set deleted successfully' TYPE 'I'.
    CATCH /aws1/cx_sescannotdeleteex INTO DATA(lo_ex1).
      DATA(lv_error) = |Cannot delete rule set: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteReceiptRuleSet referência da API AWS SDK for SAP ABAP.

`DeleteTemplate`

O código de exemplo a seguir mostra como usar `DeleteTemplate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ses->deletetemplate( iv_templatename = iv_template_name ).
    MESSAGE 'Template deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTemplate referência da API AWS SDK for SAP ABAP.

`DescribeReceiptRuleSet`

O código de exemplo a seguir mostra como usar `DescribeReceiptRuleSet`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ses->describereceiptruleset(
      iv_rulesetname = iv_rule_set_name
    ).
    MESSAGE 'Receipt rule set described successfully' TYPE 'I'.
  CATCH /aws1/cx_sesrulesetdoesnotexex INTO DATA(lo_ex1).
    DATA(lv_error) = |Rule set does not exist: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeReceiptRuleSet referência da API AWS SDK for SAP ABAP.

`GetIdentityVerificationAttributes`

O código de exemplo a seguir mostra como usar `GetIdentityVerificationAttributes`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
DATA lt_identities TYPE /aws1/cl_sesidentitylist_w=>tt_identitylist.
```

```
    APPEND NEW /aws1/cl_sesidentitylist_w( iv_value = iv_identity ) TO
lt_identities.

    TRY.
        DATA(lo_result) = lo_ses->getidentityverificationattrs(
          it_identities = lt_identities
        ).

        DATA(lt_attrs) = lo_result->get_verificationattributes( ).
        IF lt_attrs IS NOT INITIAL.
          LOOP AT lt_attrs ASSIGNING FIELD-SYMBOL(<ls_attr>).
            ov_status = <ls_attr>-value->get_verificationstatus( ).
            EXIT.
          ENDLOOP.
        ELSE.
          ov_status = 'NotFound'.
        ENDIF.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetIdentityVerificationAttributes referência da API AWS SDK for SAP ABAP.

`GetTemplate`

O código de exemplo a seguir mostra como usar `GetTemplate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_ses->gettemplate( iv_templatename = iv_template_name ).
```

```
      oo_template = lo_result->get_template( ).
      MESSAGE 'Template retrieved successfully' TYPE 'I'.
    CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
      DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetTemplate referência da API AWS SDK for SAP ABAP.

`ListIdentities`

O código de exemplo a seguir mostra como usar `ListIdentities`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      DATA(lo_result) = lo_ses->listidentities(
        iv_identitytype = iv_identity_type
        iv_maxitems = iv_max_items
      ).
      ot_identities = lo_result->get_identities( ).
      MESSAGE 'Identities retrieved successfully' TYPE 'I'.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
      DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListIdentities referência da API AWS SDK for SAP ABAP.

`ListReceiptFilters`

O código de exemplo a seguir mostra como usar `ListReceiptFilters`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_ses->listreceiptfilters( ).
    ot_filters = lo_result->get_filters( ).
    MESSAGE 'Receipt filters retrieved successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListReceiptFilters referência da API AWS SDK for SAP ABAP.

`ListTemplates`

O código de exemplo a seguir mostra como usar `ListTemplates`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_ses->listtemplates( iv_maxitems = iv_max_items ).
        ot_templates = lo_result->get_templatesmetadata( ).
        MESSAGE 'Templates retrieved successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTemplates referência da API AWS SDK for SAP ABAP.

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " Create message object
    DATA(lo_subject) = NEW /aws1/cl_sescontent( iv_data = iv_subject ).
    DATA(lo_text_body) = NEW /aws1/cl_sescontent( iv_data = iv_text ).
    DATA(lo_html_body) = NEW /aws1/cl_sescontent( iv_data = iv_html ).
    DATA(lo_body) = NEW /aws1/cl_sesbody(
      io_text = lo_text_body
      io_html = lo_html_body
    ).
    DATA(lo_message) = NEW /aws1/cl_sesmessage(
      io_subject = lo_subject
      io_body = lo_body
    ).

    TRY.
        " Send email
```

```
      DATA(lo_result) = lo_ses->sendemail(
        iv_source = iv_source
        io_destination = io_destination
        io_message = lo_message
        it_replytoaddresses = it_reply_tos
      ).
      ov_msg_id = lo_result->get_messageid( ).
      MESSAGE 'Email sent successfully' TYPE 'I'.
    CATCH /aws1/cx_sesacctsendingpause00 INTO DATA(lo_ex1).
      DATA(lv_error) = |Account sending paused: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_sesmessagerejected INTO DATA(lo_ex2).
      lv_error = |Message rejected: { lo_ex2->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex2.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- Para obter detalhes da API, consulte a SendEmail referência da API AWS SDK for SAP ABAP.

`SendTemplatedEmail`

O código de exemplo a seguir mostra como usar `SendTemplatedEmail`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      " Send templated email
      DATA(lo_result) = lo_ses->sendtemplatedemail(
```

```
        iv_source = iv_source
        io_destination = io_destination
        iv_template = iv_template_name
        iv_templatedata = iv_template_data
        it_replytoaddresses = it_reply_tos
      ).
      ov_msg_id = lo_result->get_messageid( ).
      MESSAGE 'Templated email sent successfully' TYPE 'I'.
    CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
      DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- Para obter detalhes da API, consulte a SendTemplatedEmail referência da API AWS SDK for SAP ABAP.

`UpdateTemplate`

O código de exemplo a seguir mostra como usar `UpdateTemplate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  DATA(lo_template) = NEW /aws1/cl_sestemplate(
    iv_templatename = iv_name
    iv_subjectpart = iv_subject
    iv_textpart = iv_text
    iv_htmlpart = iv_html
  ).
```

```
    TRY.
        lo_ses->updatetemplate( io_template = lo_template ).
        MESSAGE 'Template updated successfully' TYPE 'I'.
      CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
        DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_sesinvalidtemplateex INTO DATA(lo_ex2).
        lv_error = |Invalid template: { lo_ex2->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex2.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateTemplate referência da API AWS SDK for SAP ABAP.

`VerifyDomainIdentity`

O código de exemplo a seguir mostra como usar `VerifyDomainIdentity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_ses->verifydomainidentity( iv_domain =
iv_domain_name ).
        ov_token = lo_result->get_verificationtoken( ).
        MESSAGE 'Domain verification initiated' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a VerifyDomainIdentity referência da API AWS SDK for SAP ABAP.

`VerifyEmailIdentity`

O código de exemplo a seguir mostra como usar `VerifyEmailIdentity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ses->verifyemailidentity( iv_emailaddress = iv_email_address ).
        MESSAGE 'Email verification initiated' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a VerifyEmailIdentity referência da API AWS SDK for SAP ABAP.

Exemplos da API Amazon SES v2 usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com a API Amazon SES v2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateContact`

O código de exemplo a seguir mostra como usar `CreateContact`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_se2->createcontact(
      iv_contactlistname = iv_contact_list_name
      iv_emailaddress = iv_email_address ).
    MESSAGE 'Contact created successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2alreadyexistsex.
    MESSAGE 'Contact already exists.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex.
    MESSAGE 'Bad request.' TYPE 'E'.
  CATCH /aws1/cx_se2notfoundexception.
    MESSAGE 'Contact list not found.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateContact referência da API AWS SDK for SAP ABAP.

`CreateContactList`

O código de exemplo a seguir mostra como usar `CreateContactList`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_se2->createcontactlist(
      iv_contactlistname = iv_contact_list_name ).
    MESSAGE 'Contact list created successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2alreadyexistsex.
    MESSAGE 'Contact list already exists.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE 'Bad request - contact list limit may be reached.' TYPE 'I'.
    " Re-raise the exception so the caller can handle it
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_se2limitexceededex INTO DATA(lo_limit_exceeded).
    MESSAGE 'Limit exceeded - contact list limit reached.' TYPE 'I'.
    " Re-raise the exception so the caller can handle it
    RAISE EXCEPTION lo_limit_exceeded.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateContactList referência da API AWS SDK for SAP ABAP.

`CreateEmailIdentity`

O código de exemplo a seguir mostra como usar `CreateEmailIdentity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_se2->createemailidentity(
          iv_emailidentity = iv_email_identity ).
        MESSAGE 'Email identity created successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2alreadyexistsex.
        MESSAGE 'Email identity already exists.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
      CATCH /aws1/cx_se2limitexceededex INTO DATA(lo_limit_exceeded).
        MESSAGE lo_limit_exceeded TYPE 'I' DISPLAY LIKE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateEmailIdentity referência da API AWS SDK for SAP ABAP.

`CreateEmailTemplate`

O código de exemplo a seguir mostra como usar `CreateEmailTemplate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_template_content) = NEW /aws1/cl_se2emailtmplcontent(
          iv_subject = iv_subject
          iv_html = iv_html
          iv_text = iv_text ).

        lo_se2->createemailtemplate(
          iv_templatename = iv_template_name
          io_templatecontent = lo_template_content ).
        MESSAGE 'Email template created successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2alreadyexistsex.
        MESSAGE 'Email template already exists.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex.
```

```
      MESSAGE 'Bad request.' TYPE 'E'.
    CATCH /aws1/cx_se2limitexceededex.
      MESSAGE 'Limit exceeded.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateEmailTemplate referência da API AWS SDK for SAP ABAP.

`DeleteContactList`

O código de exemplo a seguir mostra como usar `DeleteContactList`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_se2->deletecontactlist(
        iv_contactlistname = iv_contact_list_name ).
      MESSAGE 'Contact list deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_se2notfoundexception.
      MESSAGE 'Contact list not found.' TYPE 'I'.
    CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
      MESSAGE 'Bad request.' TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteContactList referência da API AWS SDK for SAP ABAP.

`DeleteEmailIdentity`

O código de exemplo a seguir mostra como usar `DeleteEmailIdentity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_se2->deleteemailidentity(
      iv_emailidentity = iv_email_identity ).
    MESSAGE 'Email identity deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2notfoundexception.
    MESSAGE 'Email identity not found.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE 'Bad request.' TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteEmailIdentity referência da API AWS SDK for SAP ABAP.

`DeleteEmailTemplate`

O código de exemplo a seguir mostra como usar `DeleteEmailTemplate`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_se2->deleteemailtemplate(
      iv_templatename = iv_template_name ).
    MESSAGE 'Email template deleted successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE 'Email template not found.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteEmailTemplate referência da API AWS SDK for SAP ABAP.

`GetEmailIdentity`

O código de exemplo a seguir mostra como usar `GetEmailIdentity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_se2->getemailidentity(
          iv_emailidentity = iv_email_identity ).
        MESSAGE |Identity type: { oo_result->get_identitytype( ) }, | &&
                |verified for sending: { oo_result-
>get_verifiedforsendingstatus( ) }| TYPE 'I'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE |Email identity { iv_email_identity } not found.| TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Para obter detalhes da API, consulte a GetEmailIdentity referência da API AWS SDK for SAP ABAP.

`ListContacts`

O código de exemplo a seguir mostra como usar `ListContacts`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_se2->listcontacts(
      iv_contactlistname = iv_contact_list_name ).
    DATA(lv_count) = lines( oo_result->get_contacts( ) ).
    MESSAGE |Retrieved { lv_count } contacts from list.| TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE 'Bad request.' TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_se2notfoundexception INTO DATA(lo_not_found).
    MESSAGE 'Contact list not found.' TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
ENDTRY.
```

- Para obter detalhes da API, consulte a ListContacts referência da API AWS SDK for SAP ABAP.

`SendBulkEmail`

O código de exemplo a seguir mostra como usar `SendBulkEmail`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Build the default template content used for all bulk recipients
        DATA(lo_template) = NEW /aws1/cl_se2template(
          iv_templatename = iv_template_name
          iv_templatedata = iv_template_data ).

        DATA(lo_default_content) = NEW /aws1/cl_se2bulkemailcontent(
          io_template = lo_template ).

        DATA(lo_result) = lo_se2->sendbulkemail(
          iv_fromemailaddress = iv_from_address
          io_defaultcontent   = lo_default_content
          it_bulkemailentries = it_bulk_entries ).

        ot_results = lo_result->get_bulkemailentryresults( ).
        MESSAGE |Bulk email sent to { lines( it_bulk_entries ) } recipient(s).| TYPE
'I'.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_rejected).
        MESSAGE lo_rejected TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_rejected.
      CATCH /aws1/cx_se2mailfrmdomnotver00 INTO DATA(lo_not_verified).
        MESSAGE lo_not_verified TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_not_verified.
      CATCH /aws1/cx_se2notfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendBulkEmail referência da API AWS SDK for SAP ABAP.

`SendEmail`

O código de exemplo a seguir mostra como usar `SendEmail`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envia uma mensagem.

```
    TRY.
        " Create destination with recipient address
        DATA lt_to_addresses TYPE /aws1/
cl_se2emailaddresslist_w=>tt_emailaddresslist.
        APPEND NEW /aws1/cl_se2emailaddresslist_w( iv_value = iv_to_email_address )
 TO lt_to_addresses.
        DATA(lo_destination) = NEW /aws1/cl_se2destination(
          it_toaddresses = lt_to_addresses ).

        " Create message content
        DATA(lo_subject) = NEW /aws1/cl_se2content( iv_data = iv_subject ).
        DATA(lo_text_body) = NEW /aws1/cl_se2content( iv_data = iv_text_body ).
        DATA(lo_html_body) = NEW /aws1/cl_se2content( iv_data = iv_html_body ).
        DATA(lo_body) = NEW /aws1/cl_se2body(
          io_text = lo_text_body
          io_html = lo_html_body ).
        DATA(lo_message) = NEW /aws1/cl_se2message(
          io_subject = lo_subject
          io_body = lo_body ).

        DATA(lo_content) = NEW /aws1/cl_se2emailcontent(
          io_simple = lo_message ).

        " Send the email
        lo_se2->sendemail(
          iv_fromemailaddress = iv_from_email_address
          io_destination = lo_destination
          io_content = lo_content ).
        MESSAGE 'Email sent successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2accountsuspendedex INTO DATA(lo_account_suspended).
        MESSAGE 'Account suspended.' TYPE 'I'.
        RAISE EXCEPTION lo_account_suspended.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
```

```
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_message_rejected).
        MESSAGE 'Message rejected - check email verification.' TYPE 'I'.
        RAISE EXCEPTION lo_message_rejected.
    ENDTRY.
```

Envia uma mensagem usando um modelo.

```
    TRY.
        " Create destination with recipient address
        DATA lt_to_addresses TYPE /aws1/
cl_se2emailaddresslist_w=>tt_emailaddresslist.
        APPEND NEW /aws1/cl_se2emailaddresslist_w( iv_value = iv_to_email_address )
 TO lt_to_addresses.
        DATA(lo_destination) = NEW /aws1/cl_se2destination(
          it_toaddresses = lt_to_addresses ).

        " Create template reference
        DATA(lo_template) = NEW /aws1/cl_se2template(
          iv_templatename = iv_template_name
          iv_templatedata = iv_template_data ).

        DATA(lo_content) = NEW /aws1/cl_se2emailcontent(
          io_template = lo_template ).

        " Create list management options
        DATA(lo_list_mgmt) = NEW /aws1/cl_se2listmanagementopts(
          iv_contactlistname = iv_contact_list_name ).

        " Send the email using template
        lo_se2->sendemail(
          iv_fromemailaddress = iv_from_email_address
          io_destination = lo_destination
          io_content = lo_content
          io_listmanagementoptions = lo_list_mgmt ).
        MESSAGE 'Email sent using template successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2accountsuspendedex INTO DATA(lo_account_suspended).
        MESSAGE 'Account suspended.' TYPE 'I'.
        RAISE EXCEPTION lo_account_suspended.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
```

```
      RAISE EXCEPTION lo_bad_request.
    CATCH /aws1/cx_se2messagerejected INTO DATA(lo_message_rejected).
      MESSAGE 'Message rejected - check email verification.' TYPE 'I'.
      RAISE EXCEPTION lo_message_rejected.
  ENDTRY.
```

- Para obter detalhes da API, consulte a SendEmail referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon SNS usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result is
returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum number
of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a CreateTopic referência da API AWS SDK for SAP ABAP.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
       MESSAGE 'SNS topic deleted.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTopic referência da API AWS SDK for SAP ABAP.

`GetTopicAttributes`

O código de exemplo a seguir mostra como usar `GetTopicAttributes`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
       DATA(lt_attributes) = oo_result->get_attributes( ).
       MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetTopicAttributes referência da API AWS SDK for SAP ABAP.

`ListSubscriptions`

O código de exemplo a seguir mostra como usar `ListSubscriptions`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sns->listsubscriptions( ).                " oo_result is
returned for testing purposes. "
        DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
        MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list subscribers.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListSubscriptions referência da API AWS SDK for SAP ABAP.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sns->listtopics( ).            " oo_result is returned for
testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTopics referência da API AWS SDK for SAP ABAP.

`Publicar`

O código de exemplo a seguir mostra como usar `Publish`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
         iv_topicarn = iv_topic_arn
         iv_message = iv_message ).
       MESSAGE 'Message published to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
   ENDTRY.
```

Publique uma mensagem com atributos para um tópico.

```
   TRY.
       oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
         iv_topicarn = iv_topic_arn
         iv_message = iv_message
         it_messageattributes = it_msg_attrs ).
       MESSAGE 'Message with attributes published to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
   ENDTRY.
```

Publique uma mensagem em vários formatos em um tópico.

```
   " Build JSON message structure for multi-format message
   DATA(lv_json_message) = |\{ "default": "{ iv_default_message }", "sms":
"{ iv_sms_message }", "email": "{ iv_email_message }" \}|.

   TRY.
```

```
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = lv_json_message
          iv_subject = iv_subject
          iv_messagestructure = 'json' ).
        MESSAGE 'Multi-format message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Publicar na Referência da API do AWS SDK para SAP ABAP.

`SetSubscriptionAttributes`

O código de exemplo a seguir mostra como usar `SetSubscriptionAttributes`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sns->setsubscriptionattributes(
            iv_subscriptionarn = iv_subscription_arn
            iv_attributename  = 'FilterPolicy'
            iv_attributevalue = iv_filter_policy ).
        MESSAGE 'Added filter policy to subscription.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SetSubscriptionAttributes referência da API AWS SDK for SAP ABAP.

`SetTopicAttributes`

O código de exemplo a seguir mostra como usar `SetTopicAttributes`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename  = iv_attribute_name
        iv_attributevalue = iv_attribute_value ).
    MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a SetTopicAttributes referência da API AWS SDK for SAP ABAP.

`Assinar`

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
    TRY.
        oo_result = lo_sns->subscribe(                      "oo_result is returned
 for testing purposes."
                iv_topicarn = iv_topic_arn
                iv_protocol = 'email'
                iv_endpoint = iv_email_address
                iv_returnsubscriptionarn = abap_true ).
        MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum number
 of subscriptions allowed.' TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API em Inscrever-se na Referência da API do AWS SDK para SAP ABAP.

Cancelar assinatura

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be deleted/
unsubscribed. Confirm subscription before performing unsubscribe operation.' TYPE
 'E'.
    ENDTRY.
```

- Consulte detalhes da API em Cancelar assinatura na Referência da API do AWS SDK para SAP ABAP.

Cenários

Criar e publicar em um tópico FIFO

O exemplo de código a seguir mostra como criar e publicar em um tópico FIFO do Amazon SNS.

SDK para SAP ABAP

> **Note**
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO, inscreva uma fila FIFO do Amazon SQS no tópico e publique uma mensagem em um tópico do Amazon SNS.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value = 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum number
 of topics allowed.' TYPE 'E'.
```

```
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon SNS)
 topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed to
 an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum number
 of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String'
                                                                    iv_stringvalue =
 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from $19
 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes ).
```

```
        ov_message_id = lo_result->get_messageid( ).                    "
ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para SAP ABAP.
  - CreateTopic
  - Publicar
  - Assinar

Publicar uma mensagem de texto SMS

O exemplo de código a seguir mostra como publicar mensagens SMS usando o Amazon SNS.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    " iv_phone_number = '+12065550101' - Phone number in E.164 format
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_phonenumber = iv_phone_number
          iv_message = iv_message ).
        MESSAGE 'Message published to phone number.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Phone number does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter os detalhes da API, consulte Publicar na Referência da API do AWS SDK para SAP ABAP.

Exemplos do Amazon SQS usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma fila padrão do Amazon SQS.

```
TRY.
```

```
        oo_result = lo_sqs->createqueue( iv_queuename = iv_queue_name ).        "
 oo_result is returned for testing purposes. "
        MESSAGE 'SQS queue created.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
 queue with the same name.' TYPE 'E'.
      CATCH /aws1/cx_sqsqueuenameexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.
```

Crie uma fila do Amazon SQS que aguarda a chegada de uma mensagem.

```
    TRY.
        DATA lt_attributes TYPE /aws1/cl_sqsqueueattrmap_w=>tt_queueattributemap.
        DATA ls_attribute TYPE /aws1/
cl_sqsqueueattrmap_w=>ts_queueattributemap_maprow.
        ls_attribute-key = 'ReceiveMessageWaitTimeSeconds'.               " Time in
 seconds for long polling, such as how long the call waits for a message to arrive
 in the queue before returning. "
        ls_attribute-value = NEW /aws1/cl_sqsqueueattrmap_w( iv_value =
 iv_wait_time ).
        INSERT ls_attribute INTO TABLE lt_attributes.
        oo_result = lo_sqs->createqueue(                  " oo_result is returned
 for testing purposes. "
                iv_queuename = iv_queue_name
                it_attributes = lt_attributes ).
        MESSAGE 'SQS queue created.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
 queue with the same name.' TYPE 'E'.
      CATCH /aws1/cx_sqsqueuenameexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateQueue referência da API AWS SDK for SAP ABAP.

`DeleteMessage`

O código de exemplo a seguir mostra como usar `DeleteMessage`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       lo_sqs->deletemessage(
          iv_queueurl = iv_queue_url
          iv_receipthandle = iv_receipt_handle ).
       MESSAGE 'Message deleted from SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsinvalididformat.
       MESSAGE 'The specified receipt handle is not valid.' TYPE 'E'.
     CATCH /aws1/cx_sqsreceipthandleisinv.
       MESSAGE 'The specified receipt handle is not valid for the current version.'
TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteMessage referência da API AWS SDK for SAP ABAP.

`DeleteMessageBatch`

O código de exemplo a seguir mostra como usar `DeleteMessageBatch`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sqs->deletemessagebatch(       " oo_result is returned for
testing purposes. "
```

```
          iv_queueurl = iv_queue_url
          it_entries = it_entries ).
        MESSAGE 'Messages deleted from SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsbtcentidsnotdist00.
        MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
      CATCH /aws1/cx_sqsemptybatchrequest.
        MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
      CATCH /aws1/cx_sqsinvbatchentryid.
        MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE 'E'.
      CATCH /aws1/cx_sqstoomanyentriesin00.
        MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteMessageBatch referência da API AWS SDK for SAP ABAP.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_sqs->deletequeue( iv_queueurl = iv_queue_url ).
        MESSAGE 'SQS queue deleted' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteQueue referência da API AWS SDK for SAP ABAP.

`GetQueueUrl`

O código de exemplo a seguir mostra como usar `GetQueueUrl`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sqs->getqueueurl( iv_queuename = iv_queue_name ).        "
oo_result is returned for testing purposes. "
       MESSAGE 'Queue URL retrieved.' TYPE 'I'.
     CATCH /aws1/cx_sqsqueuedoesnotexist.
       MESSAGE 'The requested queue does not exist.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a GetQueueUrl referência da API AWS SDK for SAP ABAP.

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sqs->listqueues( ).        " oo_result is returned for
testing purposes. "
       MESSAGE 'Retrieved list of queues.' TYPE 'I'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListQueues referência da API AWS SDK for SAP ABAP.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Receba uma mensagem de uma fila do Amazon SQS.

```
    TRY.
        oo_result = lo_sqs->receivemessage( iv_queueurl = iv_queue_url ).    "
oo_result is returned for testing purposes. "
        DATA(lt_messages) = oo_result->get_messages( ).
        MESSAGE 'Message received from SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsoverlimit.
        MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
    ENDTRY.
```

Receba uma mensagem de uma fila do Amazon SQS usando o suporte de sondagem longa.

```
    TRY.
        oo_result = lo_sqs->receivemessage(           " oo_result is returned for
testing purposes. "
                iv_queueurl = iv_queue_url
                iv_waittimeseconds = iv_wait_time ).    " Time in seconds for long
polling, such as how long the call waits for a message to arrive in the queue
before returning. " ).
        DATA(lt_messages) = oo_result->get_messages( ).
        MESSAGE 'Message received from SQS queue.' TYPE 'I'.
```

```
      CATCH /aws1/cx_sqsoverlimit.
        MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ReceiveMessage referência da API AWS SDK for SAP ABAP.

SendMessage

O código de exemplo a seguir mostra como usar `SendMessage`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        oo_result = lo_sqs->sendmessage(              " oo_result is returned for
testing purposes. "
          iv_queueurl = iv_queue_url
          iv_messagebody = iv_message ).
        MESSAGE 'Message sent to SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsinvalidmsgconts.
        MESSAGE 'Message contains non-valid characters.' TYPE 'E'.
      CATCH /aws1/cx_sqsunsupportedop.
        MESSAGE 'Operation not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendMessage referência da API AWS SDK for SAP ABAP.

SendMessageBatch

O código de exemplo a seguir mostra como usar `SendMessageBatch`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   TRY.
       oo_result = lo_sqs->sendmessagebatch(         " oo_result is returned for
testing purposes. "
          iv_queueurl = iv_queue_url
          it_entries = it_messages ).
       MESSAGE 'Messages sent to SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsbtcentidsnotdist00.
       MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
     CATCH /aws1/cx_sqsbatchreqtoolong.
       MESSAGE 'The length of all the messages put together is more than the
limit.' TYPE 'E'.
     CATCH /aws1/cx_sqsemptybatchrequest.
       MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
     CATCH /aws1/cx_sqsinvbatchentryid.
       MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE 'E'.
     CATCH /aws1/cx_sqstoomanyentriesin00.
       MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
     CATCH /aws1/cx_sqsunsupportedop.
       MESSAGE 'Operation not supported.' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a SendMessageBatch referência da API AWS SDK for SAP ABAP.

Cenários

Criar e publicar em um tópico FIFO

O exemplo de código a seguir mostra como criar e publicar em um tópico FIFO do Amazon SNS.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie um tópico FIFO, inscreva uma fila FIFO do Amazon SQS no tópico e publique uma mensagem em um tópico do Amazon SNS.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value = 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum number
 of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon SNS)
 topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed to
 an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn ).
```

```
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum number
 of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String'
                                                                    iv_stringvalue =
 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from $19
 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes ).
        ov_message_id = lo_result->get_messageid( ).                    "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para SAP ABAP.

- CreateTopic
- Publicar
- Assinar

Exemplos do Step Functions usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com Step Functions.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateActivity`

O código de exemplo a seguir mostra como usar `CreateActivity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_sfn->createactivity(
      iv_name = iv_name
    ).
    ov_activity_arn = lo_result->get_activityarn( ).
    MESSAGE 'Activity created successfully.' TYPE 'I'.
```

```
    CATCH /aws1/cx_sfnactivityalrdyex.
      MESSAGE 'Activity already exists.' TYPE 'E'.
    CATCH /aws1/cx_sfninvalidname.
      MESSAGE 'Invalid activity name.' TYPE 'E'.
    CATCH /aws1/cx_sfnactivitylimitexcd.
      MESSAGE 'Activity limit exceeded.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateActivity referência da API AWS SDK for SAP ABAP.

`CreateStateMachine`

O código de exemplo a seguir mostra como usar `CreateStateMachine`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      DATA(lo_result) = lo_sfn->createstatemachine(
        iv_name = iv_name
        iv_definition = iv_definition
        iv_rolearn = iv_role_arn
      ).
      ov_state_machine_arn = lo_result->get_statemachinearn( ).
      MESSAGE 'State machine created successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfnstatemachinealrex.
      MESSAGE 'State machine already exists.' TYPE 'E'.
    CATCH /aws1/cx_sfninvaliddefinition.
      MESSAGE 'Invalid state machine definition.' TYPE 'E'.
    CATCH /aws1/cx_sfninvalidname.
      MESSAGE 'Invalid state machine name.' TYPE 'E'.
    CATCH /aws1/cx_sfninvalidarn.
      MESSAGE 'Invalid role ARN.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateStateMachine referência da API AWS SDK for SAP ABAP.

`DeleteActivity`

O código de exemplo a seguir mostra como usar `DeleteActivity`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_sfn->deleteactivity(
      iv_activityarn = iv_activity_arn
    ).
    MESSAGE 'Activity deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid activity ARN.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteActivity referência da API AWS SDK for SAP ABAP.

`DeleteStateMachine`

O código de exemplo a seguir mostra como usar `DeleteStateMachine`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_sfn->deletestatemachine(
      iv_statemachinearn = iv_state_machine_arn
    ).
    MESSAGE 'State machine deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfnvalidationex.
    MESSAGE 'Validation error occurred.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteStateMachine referência da API AWS SDK for SAP ABAP.

`DescribeExecution`

O código de exemplo a seguir mostra como usar `DescribeExecution`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_sfn->describeexecution(
      iv_executionarn = iv_execution_arn
    ).
    MESSAGE 'Execution described successfully.' TYPE 'I'.
```

```
    CATCH /aws1/cx_sfnexecdoesnotexist.
      MESSAGE 'Execution does not exist.' TYPE 'E'.
    CATCH /aws1/cx_sfninvalidarn.
      MESSAGE 'Invalid execution ARN.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeExecution referência da API AWS SDK for SAP ABAP.

`DescribeStateMachine`

O código de exemplo a seguir mostra como usar `DescribeStateMachine`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      oo_result = lo_sfn->describestatemachine(
        iv_statemachinearn = iv_state_machine_arn
      ).
      MESSAGE 'State machine described successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfnstatemachinedoes00.
      MESSAGE 'State machine does not exist.' TYPE 'E'.
    CATCH /aws1/cx_sfninvalidarn.
      MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeStateMachine referência da API AWS SDK for SAP ABAP.

`GetActivityTask`

O código de exemplo a seguir mostra como usar `GetActivityTask`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_sfn->getactivitytask(
      iv_activityarn = iv_activity_arn
    ).
    MESSAGE 'Activity task retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnactivitydoesnotex.
    MESSAGE 'Activity does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid activity ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfnactivityworkerlm00.
    MESSAGE 'Activity worker limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetActivityTask referência da API AWS SDK for SAP ABAP.

`ListActivities`

O código de exemplo a seguir mostra como usar `ListActivities`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_sfn->listactivities( ).
```

```
      DATA(lt_activities) = lo_result->get_activities( ).
      LOOP AT lt_activities INTO DATA(lo_activity).
        IF lo_activity->get_name( ) = iv_name.
          ov_activity_arn = lo_activity->get_activityarn( ).
          EXIT.
        ENDIF.
      ENDLOOP.
      MESSAGE 'Activities listed successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfninvalidtoken.
      MESSAGE 'Invalid pagination token.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListActivities referência da API AWS SDK for SAP ABAP.

`ListStateMachines`

O código de exemplo a seguir mostra como usar `ListStateMachines`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      DATA(lo_result) = lo_sfn->liststatemachines( ).
      DATA(lt_state_machines) = lo_result->get_statemachines( ).
      LOOP AT lt_state_machines INTO DATA(lo_state_machine).
        IF lo_state_machine->get_name( ) = iv_name.
          ov_state_machine_arn = lo_state_machine->get_statemachinearn( ).
          EXIT.
        ENDIF.
      ENDLOOP.
      MESSAGE 'State machines listed successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfninvalidtoken.
      MESSAGE 'Invalid pagination token.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a ListStateMachines referência da API AWS SDK for SAP ABAP.

`SendTaskSuccess`

O código de exemplo a seguir mostra como usar `SendTaskSuccess`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_sfn->sendtasksuccess(
      iv_tasktoken = iv_task_token
      iv_output = iv_task_response
    ).
    MESSAGE 'Task success sent successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfninvalidtoken.
    MESSAGE 'Invalid task token.' TYPE 'E'.
  CATCH /aws1/cx_sfntaskdoesnotexist.
    MESSAGE 'Task does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidoutput.
    MESSAGE 'Invalid task output.' TYPE 'E'.
  CATCH /aws1/cx_sfntasktimedout.
    MESSAGE 'Task timed out.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a SendTaskSuccess referência da API AWS SDK for SAP ABAP.

`StartExecution`

O código de exemplo a seguir mostra como usar `StartExecution`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    DATA(lo_result) = lo_sfn->startexecution(
      iv_statemachinearn = iv_state_machine_arn
      iv_input = iv_input
    ).
    ov_execution_arn = lo_result->get_executionarn( ).
    MESSAGE 'Execution started successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnstatemachinedoes00.
    MESSAGE 'State machine does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidexecinput.
    MESSAGE 'Invalid execution input.' TYPE 'E'.
  CATCH /aws1/cx_sfnexeclimitexceeded.
    MESSAGE 'Execution limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte a StartExecution referência da API AWS SDK for SAP ABAP.

Exemplos do Systems Manager usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Systems Manager.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateDocument`

O código de exemplo a seguir mostra como usar `CreateDocument`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ssm->createdocument(
        iv_name = iv_name
        iv_content = iv_content
        iv_documenttype = 'Command' ).
    MESSAGE 'Document created.' TYPE 'I'.
  CATCH /aws1/cx_ssmdocalreadyexists.
    MESSAGE 'Document already exists.' TYPE 'I'.
  CATCH /aws1/cx_ssminvaliddoccontent.
    MESSAGE 'Invalid document content.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateDocument referência da API AWS SDK for SAP ABAP.

`CreateMaintenanceWindow`

O código de exemplo a seguir mostra como usar `CreateMaintenanceWindow`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ssm->createmaintenancewindow(
        iv_name = iv_name
        iv_schedule = iv_schedule
        iv_duration = iv_duration
        iv_cutoff = iv_cutoff
        iv_allowunassociatedtargets = iv_allow_unassociated_targets ).
    MESSAGE 'Maintenance window created.' TYPE 'I'.
  CATCH /aws1/cx_ssmresrclimitexcdex.
    MESSAGE 'Resource limit exceeded.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a CreateMaintenanceWindow referência da API AWS SDK for SAP ABAP.

`CreateOpsItem`

O código de exemplo a seguir mostra como usar `CreateOpsItem`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_ssm->createopsitem(
        iv_title = iv_title
```

```
            iv_source = iv_source
            iv_category = iv_category
            iv_severity = iv_severity
            iv_description = iv_description ).
        MESSAGE 'OpsItem created.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsitemlimitexcdex.
        MESSAGE 'You have exceeded your open OpsItem limit.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsitemalrdyexex.
        MESSAGE 'OpsItem already exists.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a CreateOpsItem referência da API AWS SDK for SAP ABAP.

`DeleteDocument`

O código de exemplo a seguir mostra como usar `DeleteDocument`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ssm->deletedocument( iv_name = iv_name ).
        MESSAGE 'Document deleted.' TYPE 'I'.
      CATCH /aws1/cx_ssminvaliddocument.
        MESSAGE 'Invalid document.' TYPE 'I'.
      CATCH /aws1/cx_ssmassocdinstances.
        MESSAGE 'Document has associated instances.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteDocument referência da API AWS SDK for SAP ABAP.

`DeleteMaintenanceWindow`

O código de exemplo a seguir mostra como usar `DeleteMaintenanceWindow`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ssm->deletemaintenancewindow( iv_windowid = iv_window_id ).
    MESSAGE 'Maintenance window deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssminternalservererr.
    MESSAGE 'Internal server error occurred.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteMaintenanceWindow referência da API AWS SDK for SAP ABAP.

`DeleteOpsItem`

O código de exemplo a seguir mostra como usar `DeleteOpsItem`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ssm->deleteopsitem( iv_opsitemid = iv_ops_item_id ).
    MESSAGE 'OpsItem deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssmopsiteminvparamex.
```

```
        MESSAGE 'Invalid OpsItem parameter.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteOpsItem referência da API AWS SDK for SAP ABAP.

`DescribeDocument`

O código de exemplo a seguir mostra como usar `DescribeDocument`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_ssm->describedocument( iv_name = iv_name ).
        DATA(lo_document) = lo_result->get_document( ).
        IF lo_document IS BOUND.
          rv_status = lo_document->get_status( ).
          MESSAGE |Document status: { rv_status }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_ssminvaliddocument.
        MESSAGE 'Invalid document.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeDocument referência da API AWS SDK for SAP ABAP.

`DescribeOpsItems`

O código de exemplo a seguir mostra como usar `DescribeOpsItems`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Create filter for OpsItem ID
        DATA(lt_filters) = VALUE /aws1/cl_ssmopsitemfilter=>tt_opsitemfilters(
          ( NEW /aws1/cl_ssmopsitemfilter(
              iv_key = 'OpsItemId'
              it_values = VALUE /aws1/
cl_ssmopsitemfiltvals_w=>tt_opsitemfiltervalues(
                ( NEW /aws1/cl_ssmopsitemfiltvals_w( iv_value = iv_ops_item_id ) )
              )
              iv_operator = 'Equal'
            ) )
        ).

        " Use paginator to get all results
        DATA(lo_paginator) = lo_ssm->get_paginator( ).
        DATA(lo_iterator) = lo_paginator->describeopsitems(
          it_opsitemfilters = lt_filters ).

        rv_found = abap_false.

        WHILE lo_iterator->has_next( ).
          DATA(lo_result) = CAST /aws1/cl_ssmdescropsitemsrsp( lo_iterator-
>get_next( ) ).
          LOOP AT lo_result->get_opsitemsummaries( ) INTO DATA(lo_item).
            DATA(lv_title) = lo_item->get_title( ).
            DATA(lv_status) = lo_item->get_status( ).
            MESSAGE |The OpsItem title is { lv_title } and the status is
 { lv_status }| TYPE 'I'.
            rv_found = abap_true.
          ENDLOOP.
        ENDWHILE.
      CATCH /aws1/cx_ssminternalservererr.
        MESSAGE 'Internal server error occurred.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeOpsItems referência da API AWS SDK for SAP ABAP.

`ListCommandInvocations`

O código de exemplo a seguir mostra como usar `ListCommandInvocations`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        " Use paginator to get all results
        DATA(lo_paginator) = lo_ssm->get_paginator( ).
        DATA(lo_iterator) = lo_paginator->listcommandinvocations(
          iv_instanceid = iv_instance_id ).

        DATA lv_count TYPE i VALUE 0.

        WHILE lo_iterator->has_next( ).
          DATA(lo_result) = CAST /aws1/cl_ssmlistcmdinvcsresult( lo_iterator-
>get_next( ) ).
          LOOP AT lo_result->get_commandinvocations( ) INTO DATA(lo_invocation).
            lv_count = lv_count + 1.
            DATA(lv_requested_datetime) = lo_invocation->get_requesteddatetime( ).
            MESSAGE |Command invocation requested at: { lv_requested_datetime }|
 TYPE 'I'.
          ENDLOOP.
        ENDWHILE.

        MESSAGE |{ lv_count } command invocation(s) found for instance
 { iv_instance_id }.| TYPE 'I'.
      CATCH /aws1/cx_ssminvalidinstanceid.
        MESSAGE 'Invalid instance ID.' TYPE 'I'.
      CATCH /aws1/cx_ssminvalidcommandid.
```

```
        MESSAGE 'Invalid command ID.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListCommandInvocations referência da API AWS SDK for SAP ABAP.

SendCommand

O código de exemplo a seguir mostra como usar `SendCommand`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_result) = lo_ssm->sendcommand(
            it_instanceids = it_instance_ids
            iv_documentname = iv_document_name
            iv_timeoutseconds = 3600 ).
        DATA(lo_command) = lo_result->get_command( ).
        IF lo_command IS BOUND.
          rv_command_id = lo_command->get_commandid( ).
          MESSAGE 'Command sent successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_ssminvaliddocument.
        MESSAGE 'Invalid document.' TYPE 'I'.
      CATCH /aws1/cx_ssminvalidinstanceid.
        MESSAGE 'Invalid instance ID.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a SendCommand referência da API AWS SDK for SAP ABAP.

`UpdateMaintenanceWindow`

O código de exemplo a seguir mostra como usar `UpdateMaintenanceWindow`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    lo_ssm->updatemaintenancewindow(
        iv_windowid = iv_window_id
        iv_name = iv_name
        iv_enabled = iv_enabled
        iv_schedule = iv_schedule
        iv_duration = iv_duration
        iv_cutoff = iv_cutoff
        iv_allowunassociatedtargets = iv_allow_unassociated_targets ).
    MESSAGE 'Maintenance window updated.' TYPE 'I'.
  CATCH /aws1/cx_ssmdoesnotexistex.
    MESSAGE 'Maintenance window does not exist.' TYPE 'I'.
ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateMaintenanceWindow referência da API AWS SDK for SAP ABAP.

`UpdateOpsItem`

O código de exemplo a seguir mostra como usar `UpdateOpsItem`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        lo_ssm->updateopsitem(
            iv_opsitemid = iv_ops_item_id
            iv_title = iv_title
            iv_description = iv_description
            iv_status = iv_status ).
        MESSAGE 'OpsItem updated.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsitemnotfoundex.
        MESSAGE 'OpsItem not found.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsiteminvparamex.
        MESSAGE 'Invalid OpsItem parameter.' TYPE 'I'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateOpsItem referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Textract usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Textract.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AnalyzeDocument`

O código de exemplo a seguir mostra como usar `AnalyzeDocument`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Detects text and additional elements, such as forms or tables,"
   "in a local image file or from in-memory byte data."
   "The image must be in PNG or JPG format."


   "Create ABAP objects for feature type."
   "Add TABLES to return information about the tables."
   "Add FORMS to return detected form data."
   "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).

   "Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).

   "Analyze document stored in Amazon S3."
   TRY.
       oo_result = lo_tex->analyzedocument(      "oo_result is returned for testing
purposes."
         io_document        = lo_document
```

```
          it_featuretypes    = lt_featuretypes ).
        LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
          IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
            MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
          ENDIF.
        ENDLOOP.
        MESSAGE 'Analyze document completed.' TYPE 'I'.
      CATCH /aws1/cx_texaccessdeniedex.
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
      CATCH /aws1/cx_texbaddocumentex.
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
      CATCH /aws1/cx_texdocumenttoolargeex.
        MESSAGE 'The document is too large.' TYPE 'E'.
      CATCH /aws1/cx_texhlquotaexceededex.
        MESSAGE 'Human loop quota exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.

      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a AnalyzeDocument referência da API AWS SDK for SAP ABAP.

`DetectDocumentText`

O código de exemplo a seguir mostra como usar `DetectDocumentText`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Detects text in the input document."
   "Amazon Textract can detect lines of text and the words that make up a line of
 text."
   "The input document must be in one of the following image formats: JPEG, PNG,
 PDF, or TIFF."

   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).
   "Analyze document stored in Amazon S3."
   TRY.
       oo_result = lo_tex->detectdocumenttext( io_document = lo_document ).
 "oo_result is returned for testing purposes."
       LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
         IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
           MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
         ENDIF.
       ENDLOOP.
       DATA(lo_metadata) = oo_result->get_documentmetadata( ).
       MESSAGE 'The number of pages in the document is ' && lo_metadata-
>ask_pages( ) TYPE 'I'.
       MESSAGE 'Detect document text completed.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
```

```
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a DetectDocumentText referência da API AWS SDK for SAP ABAP.

`GetDocumentAnalysis`

O código de exemplo a seguir mostra como usar `GetDocumentAnalysis`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Gets the results for an Amazon Textract"
   "asynchronous operation that analyzes text in a document."
   TRY.
       oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
"oo_result is returned for testing purposes."
       WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
         IF sy-index = 10.
           EXIT.               "Maximum 300 seconds.
         ENDIF.
         WAIT UP TO 30 SECONDS.
         oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
       ENDWHILE.
```

```
      DATA(lt_blocks) = oo_result->get_blocks( ).
      LOOP AT lt_blocks INTO DATA(lo_block).
        IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
          MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
        ENDIF.
      ENDLOOP.
      MESSAGE 'Document analysis retrieved.' TYPE 'I'.
    CATCH /aws1/cx_texaccessdeniedex.
      MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
    CATCH /aws1/cx_texinternalservererr.
      MESSAGE 'Internal server error.' TYPE 'E'.
    CATCH /aws1/cx_texinvalidjobidex.
      MESSAGE 'Job ID is not valid.' TYPE 'E'.
    CATCH /aws1/cx_texinvalidkmskeyex.
      MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
    CATCH /aws1/cx_texinvalidparameterex.
      MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
    CATCH /aws1/cx_texinvalids3objectex.
      MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
    CATCH /aws1/cx_texprovthruputexcdex.
      MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
    CATCH /aws1/cx_texthrottlingex.
      MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a GetDocumentAnalysis referência da API AWS SDK for SAP ABAP.

`StartDocumentAnalysis`

O código de exemplo a seguir mostra como usar `StartDocumentAnalysis`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Starts the asynchronous analysis of an input document for relationships"
   "between detected items such as key-value pairs, tables, and selection
elements."

   "Create ABAP objects for feature type."
   "Add TABLES to return information about the tables."
   "Add FORMS to return detected form data."
   "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).
   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).
   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

   "Start async document analysis."
   TRY.
       oo_result = lo_tex->startdocumentanalysis(      "oo_result is returned for
testing purposes."
         io_documentlocation     = lo_documentlocation
         it_featuretypes         = lt_featuretypes ).
       DATA(lv_jobid) = oo_result->get_jobid( ).

       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
```

```
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texlimitexceededex.
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartDocumentAnalysis referência da API AWS SDK for SAP ABAP.

`StartDocumentTextDetection`

O código de exemplo a seguir mostra como usar `StartDocumentTextDetection`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Starts the asynchronous detection of text in a document."
    "Amazon Textract can detect lines of text and the words that make up a line of
text."

    "Create an ABAP object for the Amazon S3 object."
    DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
      iv_name   = iv_s3object ).
    "Create an ABAP object for the document."
    DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).
    "Start document analysis."
    TRY.
```

```
      oo_result = lo_tex->startdocumenttextdetection( io_documentlocation =
lo_documentlocation ).
      DATA(lv_jobid) = oo_result->get_jobid( ).            "oo_result is returned
for testing purposes."
      MESSAGE 'Document analysis started.' TYPE 'I'.
    CATCH /aws1/cx_texaccessdeniedex.
      MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
    CATCH /aws1/cx_texbaddocumentex.
      MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
    CATCH /aws1/cx_texdocumenttoolargeex.
      MESSAGE 'The document is too large.' TYPE 'E'.
    CATCH /aws1/cx_texidempotentprmmis00.
      MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
    CATCH /aws1/cx_texinternalservererr.
      MESSAGE 'Internal server error.' TYPE 'E'.
    CATCH /aws1/cx_texinvalidkmskeyex.
      MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
    CATCH /aws1/cx_texinvalidparameterex.
      MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
    CATCH /aws1/cx_texinvalids3objectex.
      MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
    CATCH /aws1/cx_texlimitexceededex.
      MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
    CATCH /aws1/cx_texprovthruputexcdex.
      MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
    CATCH /aws1/cx_texthrottlingex.
      MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
    CATCH /aws1/cx_texunsupporteddocex.
      MESSAGE 'The document is not supported.' TYPE 'E'.
  ENDTRY.
```

- Para obter detalhes da API, consulte a StartDocumentTextDetection referência da API AWS SDK for SAP ABAP.

Cenários

Conceitos básicos de usar análise de documentos

O exemplo de código a seguir mostra como:

- Iniciar análise assíncrona.

- Obter análise de documento.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Create ABAP objects for feature type."
   "Add TABLES to return information about the tables."
   "Add FORMS to return detected form data."
   "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).

   "Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

   "Start document analysis."
   TRY.
       DATA(lo_start_result) = lo_tex->startdocumentanalysis(
         io_documentlocation     = lo_documentlocation
         it_featuretypes         = lt_featuretypes ).
       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
```

```
        MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidkmskeyex.
        MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texlimitexceededex.
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.

    "Get job ID from the output."
    DATA(lv_jobid) = lo_start_result->get_jobid( ).

    "Wait for job to complete."
    oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).     " oo_result
is returned for testing purposes. "
    WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
      IF sy-index = 10.
        EXIT.               "Maximum 300 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).
    ENDWHILE.

    DATA(lt_blocks) = oo_result->get_blocks( ).
    LOOP AT lt_blocks INTO DATA(lo_block).
      IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
        MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
      ENDIF.
    ENDLOOP.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.

- GetDocumentAnalysis
- StartDocumentAnalysis

Exemplos do Amazon Transcribe usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Transcribe.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateVocabulary`

O código de exemplo a seguir mostra como usar `CreateVocabulary`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    IF it_phrases IS NOT INITIAL.
      oo_result = lo_tnb->createvocabulary(
        iv_vocabularyname = iv_vocabulary_name
        iv_languagecode = iv_language_code
        it_phrases = it_phrases ).
    ELSEIF iv_vocab_file_uri IS NOT INITIAL.
```

```
        oo_result = lo_tnb->createvocabulary(
          iv_vocabularyname = iv_vocabulary_name
          iv_languagecode = iv_language_code
          iv_vocabularyfileuri = iv_vocab_file_uri ).
      ENDIF.
      MESSAGE 'Custom vocabulary created.' TYPE 'I'.
    CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex TYPE 'I'.
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
      MESSAGE lo_limit_ex TYPE 'I'.
      RAISE EXCEPTION lo_limit_ex.
    CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
      MESSAGE lo_internal_ex TYPE 'I'.
      RAISE EXCEPTION lo_internal_ex.
    CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
      MESSAGE lo_conflict_ex TYPE 'I'.
      RAISE EXCEPTION lo_conflict_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a CreateVocabulary referência da API AWS SDK for SAP ABAP.

`DeleteTranscriptionJob`

O código de exemplo a seguir mostra como usar `DeleteTranscriptionJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_tnb->deletetranscriptionjob( iv_job_name ).
      MESSAGE 'Transcription job deleted.' TYPE 'I'.
    CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex TYPE 'I'.
```

```
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
      MESSAGE lo_limit_ex TYPE 'I'.
      RAISE EXCEPTION lo_limit_ex.
    CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
      MESSAGE lo_internal_ex TYPE 'I'.
      RAISE EXCEPTION lo_internal_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteTranscriptionJob referência da API AWS SDK for SAP ABAP.

`DeleteVocabulary`

O código de exemplo a seguir mostra como usar `DeleteVocabulary`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
  TRY.
      lo_tnb->deletevocabulary( iv_vocabulary_name ).
      MESSAGE 'Vocabulary deleted.' TYPE 'I'.
    CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex TYPE 'I'.
    CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
      MESSAGE lo_limit_ex TYPE 'I'.
      RAISE EXCEPTION lo_limit_ex.
    CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
      MESSAGE lo_not_found_ex TYPE 'I'.
    CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
      MESSAGE lo_internal_ex TYPE 'I'.
      RAISE EXCEPTION lo_internal_ex.
  ENDTRY.
```

- Para obter detalhes da API, consulte a DeleteVocabulary referência da API AWS SDK for SAP ABAP.

GetTranscriptionJob

O código de exemplo a seguir mostra como usar `GetTranscriptionJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_tnb->gettranscriptionjob( iv_job_name ).
    DATA(lo_job) = oo_result->get_transcriptionjob( ).
    MESSAGE 'Retrieved transcription job details.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
    RAISE EXCEPTION lo_bad_request_ex.
  CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
    MESSAGE lo_not_found_ex TYPE 'I'.
    RAISE EXCEPTION lo_not_found_ex.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetTranscriptionJob referência da API AWS SDK for SAP ABAP.

GetVocabulary

O código de exemplo a seguir mostra como usar `GetVocabulary`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    oo_result = lo_tnb->getvocabulary( iv_vocabulary_name ).
    MESSAGE 'Retrieved vocabulary details.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
    RAISE EXCEPTION lo_bad_request_ex.
  CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
    MESSAGE lo_not_found_ex TYPE 'I'.
    RAISE EXCEPTION lo_not_found_ex.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
ENDTRY.
```

- Para obter detalhes da API, consulte a GetVocabulary referência da API AWS SDK for SAP ABAP.

`ListTranscriptionJobs`

O código de exemplo a seguir mostra como usar `ListTranscriptionJobs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
TRY.
    IF iv_job_filter IS NOT INITIAL.
```

```
          oo_result = lo_tnb->listtranscriptionjobs( iv_jobnamecontains =
iv_job_filter ).
        ELSE.
          oo_result = lo_tnb->listtranscriptionjobs( ).
        ENDIF.
        MESSAGE 'Retrieved transcription jobs list.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListTranscriptionJobs referência da API AWS SDK for SAP ABAP.

`ListVocabularies`

O código de exemplo a seguir mostra como usar `ListVocabularies`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        IF iv_vocab_filter IS NOT INITIAL.
          oo_result = lo_tnb->listvocabularies( iv_namecontains = iv_vocab_filter ).
        ELSE.
          oo_result = lo_tnb->listvocabularies( ).
        ENDIF.
        MESSAGE 'Retrieved vocabularies list.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
```

```
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a ListVocabularies referência da API AWS SDK for SAP ABAP.

`StartTranscriptionJob`

O código de exemplo a seguir mostra como usar `StartTranscriptionJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        DATA(lo_media) = NEW /aws1/cl_tnbmedia( iv_mediafileuri = iv_media_uri ).
        DATA(lo_settings) = NEW /aws1/cl_tnbsettings( ).
        IF iv_vocabulary_name IS NOT INITIAL.
          lo_settings = NEW /aws1/cl_tnbsettings( iv_vocabularyname =
iv_vocabulary_name ).
        ENDIF.

        oo_result = lo_tnb->starttranscriptionjob(
          iv_transcriptionjobname = iv_job_name
          io_media = lo_media
          iv_mediaformat = iv_media_format
          iv_languagecode = iv_language_code
          io_settings = lo_settings ).

        MESSAGE 'Transcription job started.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
```

```
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartTranscriptionJob referência da API AWS SDK for SAP ABAP.

`UpdateVocabulary`

O código de exemplo a seguir mostra como usar `UpdateVocabulary`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    TRY.
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Vocabulary updated.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
```

```
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- Para obter detalhes da API, consulte a UpdateVocabulary referência da API AWS SDK for SAP ABAP.

Exemplos do Amazon Translate usando o SDK para SAP ABAP

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para SAP ABAP com o Amazon Translate.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`DescribeTextTranslationJob`

O código de exemplo a seguir mostra como usar `DescribeTextTranslationJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Gets the properties associated with an asynchronous batch translation job."
   "Includes properties such as name, ID, status, source and target languages, and
input/output Amazon Simple Storage Service (Amazon S3) buckets."
   TRY.
       oo_result = lo_xl8->describetexttranslationjob(      "oo_result is returned
for testing purposes."
         iv_jobid        = iv_jobid ).
       MESSAGE 'Job description retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a DescribeTextTranslationJob referência da API AWS SDK for SAP ABAP.

`ListTextTranslationJobs`

O código de exemplo a seguir mostra como usar `ListTextTranslationJobs`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Gets a list of the batch translation jobs that you have submitted."

   DATA lo_filter TYPE REF TO /aws1/cl_xl8textxlationjobfilt.

   "Create an ABAP object for filtering using jobname."
   lo_filter = NEW #( iv_jobname = iv_jobname ).

   TRY.
       oo_result = lo_xl8->listtexttranslationjobs(      "oo_result is returned for
testing purposes."
         io_filter        = lo_filter ).
       MESSAGE 'Jobs retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidfilterex.
       MESSAGE 'The filter specified for the operation is not valid. Specify a
different filter.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a ListTextTranslationJobs referência da API AWS SDK for SAP ABAP.

`StartTextTranslationJob`

O código de exemplo a seguir mostra como usar `StartTextTranslationJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Starts an asynchronous batch translation job."
```

```
    "Use batch translation jobs to translate large volumes of text across multiple
 documents at once."

    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    lo_inputdataconfig = NEW #( iv_s3uri = iv_input_data_s3uri
                                iv_contenttype = iv_input_data_contenttype ).

    "Create an ABAP object for the output data config."
    lo_outputdataconfig = NEW #( iv_s3uri = iv_output_data_s3uri ).

    "Create an internal table for target languages."
    lo_targetlanguagecodes = NEW #( iv_value = iv_targetlanguagecode ).
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
        oo_result = lo_xl8->starttexttranslationjob(      "oo_result is returned for
 testing purposes."
          io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode ).
        MESSAGE 'Translation job started.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
      CATCH /aws1/cx_xl8invparamvalueex.
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
 TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00.
        MESSAGE 'Amazon Translate does not support translation from the language of
 the source text into the requested target language.' TYPE 'E'.
```

```
    ENDTRY.
```

- Para obter detalhes da API, consulte a StartTextTranslationJob referência da API AWS SDK for SAP ABAP.

StopTextTranslationJob

O código de exemplo a seguir mostra como usar `StopTextTranslationJob`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    "Stops an asynchronous batch translation job that is in progress."

    TRY.
        oo_result = lo_xl8->stoptexttranslationjob(      "oo_result is returned for
testing purposes."
          iv_jobid        = iv_jobid ).
        MESSAGE 'Translation job stopped.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    ENDTRY.
```

- Para obter detalhes da API, consulte a StopTextTranslationJob referência da API AWS SDK for SAP ABAP.

`TranslateText`

O código de exemplo a seguir mostra como usar `TranslateText`.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   "Translates input text from the source language to the target language."
   TRY.
       oo_result = lo_xl8->translatetext(      "oo_result is returned for testing
purposes."
         iv_text        = iv_text
           iv_sourcelanguagecode = iv_sourcelanguagecode
           iv_targetlanguagecode = iv_targetlanguagecode ).
       MESSAGE 'Translation completed.' TYPE 'I'.
     CATCH /aws1/cx_xl8detectedlanguage00.
       MESSAGE 'The confidence that Amazon Comprehend accurately detected the
source language is low.' TYPE 'E'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8serviceunavailex.
       MESSAGE 'The Amazon Translate service is temporarily unavailable.' TYPE 'E'.
     CATCH /aws1/cx_xl8textsizelmtexcdex.
       MESSAGE 'The size of the text you submitted exceeds the size limit. ' TYPE
'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
     CATCH /aws1/cx_xl8unsuppedlanguage00.
       MESSAGE 'Amazon Translate does not support translation from the language of
the source text into the requested target language. ' TYPE 'E'.
   ENDTRY.
```

- Para obter detalhes da API, consulte a TranslateText referência da API AWS SDK for SAP ABAP.

Cenários

Conceitos básicos de usar trabalhos de tradução

O exemplo de código a seguir mostra como:

- Iniciar um trabalho assíncrono de tradução em lote.
- Aguardar a conclusão do trabalho assíncrono.
- Descrever o trabalho assíncrono.

SDK para SAP ABAP

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    lo_inputdataconfig = NEW #( iv_s3uri = iv_input_data_s3uri
                                iv_contenttype = iv_input_data_contenttype ).

    "Create an ABAP object for the output data config."
    lo_outputdataconfig = NEW #( iv_s3uri = iv_output_data_s3uri ).

    "Create an internal table for target languages."
    lo_targetlanguagecodes = NEW #( iv_value = iv_targetlanguagecode ).
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
```

```
        DATA(lo_translationjob_result) = lo_xl8->starttexttranslationjob(
          io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode ).
        MESSAGE 'Translation job started.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
      CATCH /aws1/cx_xl8invparamvalueex.
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time. '
 TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00.
        MESSAGE 'Amazon Translate does not support translation from the language of
 the source text into the requested target language.' TYPE 'E'.
    ENDTRY.

    "Get the job ID."
    DATA(lv_jobid) = lo_translationjob_result->get_jobid( ).

    "Wait for translate job to complete."
    DATA(lo_des_translation_result) = lo_xl8->describetexttranslationjob( iv_jobid =
 lv_jobid ).
    WHILE lo_des_translation_result->get_textxlationjobproperties( )-
>get_jobstatus( ) <> 'COMPLETED'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_des_translation_result = lo_xl8->describetexttranslationjob( iv_jobid =
 lv_jobid ).
    ENDWHILE.

    TRY.
        oo_result = lo_xl8->describetexttranslationjob(      "oo_result is returned
 for testing purposes."
          iv_jobid        = lv_jobid ).
```

```
        MESSAGE 'Job description retrieved.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    ENDTRY.
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para SAP ABAP.
  - DescribeTextTranslationJob
  - StartTextTranslationJob

Exemplos de código para o SDK para Swift.

Os exemplos de código a seguir mostram como usar o AWS SDK para Swift com AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Alguns serviços contêm categorias de exemplo adicionais que mostram como utilizar bibliotecas ou funções específicas do serviço.

Mais atributos

- Guia do desenvolvedor do SDK para Swift — Saiba mais sobre como usar o Swift com AWS.
- Centro do desenvolvedor da AWS : exemplos de código que você pode filtrar por categoria ou pesquisa de texto completo.

- AWS Exemplos de SDK — GitHub repositório com código completo nos idiomas preferidos. Inclui instruções para configurar e executar o código.

Services

- Exemplos do Amazon Bedrock usando o SDK para Swift
- Exemplos do Amazon Bedrock Runtime usando o SDK para Swift
- Exemplos do Amazon Cognito Identity usando o SDK para Swift
- Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Swift
- Exemplos de código do DynamoDB usando o SDK para Swift
- Exemplos de código do Amazon EC2 usando o SDK para Swift
- AWS Glue exemplos de uso do SDK para Swift
- Exemplos do IAM usando o SDK para Swift
- Exemplos do Lambda usando o SDK para Swift
- Exemplos de código do Amazon RDS usando o SDK para Swift
- Exemplos de código do Amazon S3 usando o SDK para Swift
- Exemplos do Amazon SNS usando o SDK para Swift
- Exemplos de código do Amazon SQS usando o SDK para Swift
- AWS STS exemplos de uso do SDK para Swift
- Exemplos do Amazon Transcribe Streaming usando o SDK para Swift

Exemplos do Amazon Bedrock usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon Bedrock.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos

- Ações

Conceitos básicos

Olá, Amazon Bedrock

O exemplo de código a seguir mostra como começar a usar o Amazon Bedrock.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import Foundation

import AWSBedrock

struct ExampleCommand: ParsableCommand {
    static var configuration = CommandConfiguration(
        commandName: "ListFoundationModels",
        abstract: """
        This example demonstrates how to retrieve a list of the available
        foundation models from Amazon Bedrock.
        """,
        discussion: """
        """
    )

    /// Construct a string listing the specified modalities.
    ///
    /// - Parameter modalities: An array of the modalities to list.
    ///
    /// - Returns: A string with a human-readable list of modalities.
    func buildModalityList(modalities: [BedrockClientTypes.ModelModality]?) ->
 String {
        var first = true
        var str = ""
```

```
        if modalities == nil {
            return "<none>"
        }

        for modality in modalities! {
            if !first {
                str += ", "
            }
            first = false
            str += modality.rawValue
        }

        return str
    }

    /// Construct a string listing the specified customizations.
    ///
    /// - Parameter customizations: An array of the customizations to list.
    ///
    /// - Returns: A string listing the customizations.
    func buildCustomizationList(customizations:
[BedrockClientTypes.ModelCustomization]?) -> String {
        var first = true
        var str = ""

        if customizations == nil {
            return "<none>"
        }

        for customization in customizations! {
            if !first {
                str += ", "
            }
            first = false
            str += customization.rawValue
        }

        return str
    }

    /// Construct a string listing the specified inferences.
    ///
    /// - Parameter inferences: An array of inferences to list.
```

```
    ///
    /// - Returns: A string listing the specified inferences.
    func buildInferenceList(inferences: [BedrockClientTypes.InferenceType]?) ->
 String {
        var first = true
        var str = ""

        if inferences == nil {
            return "<none>"
        }

        for inference in inferences! {
            if !first {
                str += ", "
            }
            first = false
            str += inference.rawValue
        }

        return str
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // Always use the Region "us-east-1" to have access to the most models.
        let config = try await BedrockClient.BedrockClientConfiguration(region: "us-
east-1")
        let bedrockClient = BedrockClient(config: config)

        let output = try await bedrockClient.listFoundationModels(
            input: ListFoundationModelsInput()
        )

        guard let summaries = output.modelSummaries else {
            print("No models returned.")
            return
        }

        // Output a list of the models with their details.
        for summary in summaries {
            print("==========================================")
            print(" Model ID: \(summary.modelId ?? "<unknown>")")
            print("------------------------------------------")
            print(" Name: \(summary.modelName ?? "<unknown>")")
```

```
            print(" Provider: \(summary.providerName ?? "<unknown>")")
            print(" Input modalities: \(buildModalityList(modalities:
 summary.inputModalities))")
            print(" Output modalities: \(buildModalityList(modalities:
 summary.outputModalities))")
            print(" Supported customizations:
 \(buildCustomizationList(customizations: summary.customizationsSupported ))")
            print(" Supported inference types: \(buildInferenceList(inferences:
 summary.inferenceTypesSupported))")
            print("----------------------------------------\n")
        }

        print("\(summaries.count) models available.")
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte ListFoundationModels a referência da API AWS SDK for Swift.

Ações

`ListFoundationModels`

O código de exemplo a seguir mostra como usar `ListFoundationModels`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSBedrock

        // Always use the Region "us-east-1" to have access to the most models.
        let config = try await BedrockClient.BedrockClientConfiguration(region: "us-
east-1")
        let bedrockClient = BedrockClient(config: config)

        let output = try await bedrockClient.listFoundationModels(
            input: ListFoundationModelsInput()
        )

        guard let summaries = output.modelSummaries else {
            print("No models returned.")
            return
        }

        // Output a list of the models with their details.
        for summary in summaries {
            print("==========================================")
            print(" Model ID: \(summary.modelId ?? "<unknown>")")
            print("------------------------------------------")
            print(" Name: \(summary.modelName ?? "<unknown>")")
            print(" Provider: \(summary.providerName ?? "<unknown>")")
            print(" Input modalities: \(buildModalityList(modalities:
 summary.inputModalities))")
            print(" Output modalities: \(buildModalityList(modalities:
 summary.outputModalities))")
            print(" Supported customizations:
 \(buildCustomizationList(customizations: summary.customizationsSupported ))")
            print(" Supported inference types: \(buildInferenceList(inferences:
 summary.inferenceTypesSupported))")
            print("------------------------------------------\n")
        }
```

- Para obter detalhes da API, consulte ListFoundationModels a referência da API AWS SDK for Swift.

Exemplos do Amazon Bedrock Runtime usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon Bedrock Runtime.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Amazônia Nova
- Amazon Nova Canvas
- Amazon Nova Reel
- Claude da Anthropic
- Llama da Meta

Amazônia Nova

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Amazon Nova.
```

```
import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "amazon.nova-micro-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
```

```
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Swift.

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processar o fluxo de respostas em tempo real.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto para o Amazon Nova usando a API Converse do Bedrock e processe o fluxo de respostas em tempo real.

```
// An example demonstrating how to use the Conversation API to send a text message
// to Amazon Nova and print the response stream

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "amazon.nova-lite-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )
```

```
    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nNova Lite:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStream a referência da API AWS SDK for Swift.

Amazon Nova Canvas

InvokeModel

O exemplo de código a seguir mostra como invocar o Amazon Nova Canvas no Amazon Bedrock para gerar uma imagem.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma imagem com o Amazon Nova Canvas.

```
// Use the native inference API to create an image with Amazon Nova Canvas

import AWSBedrockRuntime
import AWSSDKIdentity
import Foundation

struct NovaImageOutput: Decodable {
    let images: [Data]
}

func generateImage(_ textPrompt: String) async throws {
    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    config.awsCredentialIdentityResolver = try SSOAWSCredentialIdentityResolver()

    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "amazon.nova-canvas-v1:0"
```

```
    //  Format the request payload using the model's native structure.
    let input = InvokeModelInput(
        accept: "application/json",
        body: """
            {
                "textToImageParams": {
                    "text": "\(textPrompt)"
                },
                "taskType": "TEXT_IMAGE",
                "imageGenerationConfig": {
                    "seed": 42,
                    "quality": "standard",
                    "width": 512,
                    "height": 512,
                    "numberOfImages": 1
                }
            }
            """.data(using: .utf8),
        modelId: modelId
    )

    // Invoke the model with the request.
    let response = try await client.invokeModel(input: input)

    // Decode the response body.
    let output = try JSONDecoder().decode(NovaImageOutput.self, from:
 response.body!)

    // Extract the image data.
    guard let data = output.images.first else {
        print("No image data found")
        return
    }

    // Save the generated image to a local folder.
    let fileURL = URL.documentsDirectory.appending(path: "nova_canvas.png")
    print(fileURL)
    try data.write(to: fileURL)
    print("Image is saved at \(fileURL)")
}
```

- Para obter detalhes da API, consulte InvokeModel a referência da API AWS SDK for Swift.

Amazon Nova Reel

Text-to-video

O exemplo de código a seguir mostra como usar o Amazon Nova Reel para gerar um vídeo de um prompt de texto.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o Amazon Nova Reel para gerar um vídeo de um prompt de texto.

```
// This example demonstrates how to use Amazon Nova Reel to generate a video from a
 text prompt.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Configure a text-to-video request
// - Submit an asynchronous job for video generation
// - Poll for job completion status
// - Access the generated video from S3

import AWSBedrockRuntime
import Foundation
import Smithy

func startTextToVideoGenerationJob(
    bedrockRuntimeClient: BedrockRuntimeClient, prompt: String, outputS3Uri: String
) async throws -> String? {
    // Specify the model ID for text-to-video generation
    let modelId = "amazon.nova-reel-v1:0"

    // Configure the video generation request with additional parameters
    let modelInputSource: [String: Any] = [
        "taskType": "TEXT_VIDEO",
        "textToVideoParams": [
            "text": "\(prompt)"
        ],
        "videoGenerationConfig": [
```

```
            "durationSeconds": 6,
            "fps": 24,
            "dimension": "1280x720",
        ],
    ]

    let modelInput = try Document.make(from: modelInputSource)

    let input = StartAsyncInvokeInput(
        modelId: modelId,
        modelInput: modelInput,
        outputDataConfig: .s3outputdataconfig(
            BedrockRuntimeClientTypes.AsyncInvokeS3OutputDataConfig(
                s3Uri: outputS3Uri
            )
        )
    )

    // Invoke the model asynchronously
    let output = try await bedrockRuntimeClient.startAsyncInvoke(input: input)
    return output.invocationArn
}

func queryJobStatus(
    bedrockRuntimeClient: BedrockRuntimeClient,
    invocationArn: String?
) async throws -> GetAsyncInvokeOutput {
    try await bedrockRuntimeClient.getAsyncInvoke(
        input: GetAsyncInvokeInput(invocationArn: invocationArn))
}

func main() async throws {
    // Create a Bedrock Runtime client
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Specify the S3 location for the output video
    let bucket = "s3://REPLACE-WITH-YOUR-S3-BUCKET-NAM"

    print("Submitting video generation job...")
    let invocationArn = try await startTextToVideoGenerationJob(
        bedrockRuntimeClient: client,
        prompt: "A pomegranate juice in a railway station",
```

```
        outputS3Uri: bucket
    )
    print("Job started with invocation ARN: \(String(describing:invocationArn))")

    // Poll for job completion
    var status: BedrockRuntimeClientTypes.AsyncInvokeStatus?
    var isReady = false
    var hasFailed = false

    while !isReady && !hasFailed {
        print("\nPolling job status...")
        status = try await queryJobStatus(
            bedrockRuntimeClient: client, invocationArn: invocationArn
        ).status
        switch status {
        case .completed:
            isReady = true
            print("Video is ready\nCheck S3 bucket: \(bucket)")
        case .failed:
            hasFailed = true
            print("Something went wrong")
        case .inProgress:
            print("Job is in progress...")
            try await Task.sleep(nanoseconds: 15 * 1_000_000_000)  // 15 seconds
        default:
            isReady = true
        }
    }
}

do {
    try await main()
} catch {
    print("An error occurred: \(error)")
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - GetAsyncInvoke
  - StartAsyncInvoke

Claude da Anthropic

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Anthropic Claude.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "anthropic.claude-3-haiku-20240307-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
```

```
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Swift.

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Claude da Anthropic usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// An example demonstrating how to use the Conversation API to send a text message
// to Anthropic Claude and print the response stream

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "anthropic.claude-3-haiku-20240307-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }
```

```
    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nAnthropic Claude:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStream a referência da API AWS SDK for Swift.

Llama da Meta

Converse

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Meta Llama.
```

```
import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "meta.llama3-8b-instruct-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
```

```
}
```

- Consulte detalhes da API em Converse na Referência de API do AWS SDK para Swift.

ConverseStream

O exemplo de código a seguir mostra como enviar uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processar o fluxo de resposta em tempo real.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Envie uma mensagem de texto ao Llama da Meta usando a API Converse do Bedrock e processe o fluxo de resposta em tempo real.

```
// An example demonstrating how to use the Conversation API to send a text message
// to Meta Llama and print the response stream.

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "meta.llama3-8b-instruct-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )
```

```
    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nMeta Llama:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- Para obter detalhes da API, consulte ConverseStream a referência da API AWS SDK for Swift.

Exemplos do Amazon Cognito Identity usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon Cognito Identity.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`CreateIdentityPool`

O código de exemplo a seguir mostra como usar `CreateIdentityPool`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSCognitoIdentity


    /// Create a new identity pool and return its ID.
    ///
    /// - Parameters:
    ///     - name: The name to give the new identity pool.
    ///
```

```
    /// - Returns: A string containing the newly created pool's ID, or `nil`
    ///   if an error occurred.
    ///
    func createIdentityPool(name: String) async throws -> String? {
        do {
            let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
"com.exampleco.CognitoIdentityDemo",
                                                           identityPoolName: name)

            let result = try await cognitoIdentityClient.createIdentityPool(input:
cognitoInputCall)
            guard let poolId = result.identityPoolId else {
                return nil
            }

            return poolId
        } catch {
            print("ERROR: createIdentityPool:", dump(error))
            throw error
        }
    }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Swift.
- Para obter detalhes da API, consulte CreateIdentityPool a referência da API AWS SDK for Swift.

`DeleteIdentityPool`

O código de exemplo a seguir mostra como usar `DeleteIdentityPool`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSCognitoIdentity
```

```
    /// Delete the specified identity pool.
    ///
    /// - Parameters:
    ///   - id: The ID of the identity pool to delete.
    ///
    func deleteIdentityPool(id: String) async throws {
        do {
            let input = DeleteIdentityPoolInput(
                identityPoolId: id
            )

            _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
        } catch {
            print("ERROR: deleteIdentityPool:", dump(error))
            throw error
        }
    }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Swift.
- Para obter detalhes da API, consulte DeleteIdentityPool a referência da API AWS SDK for Swift.

`ListIdentityPools`

O código de exemplo a seguir mostra como usar `ListIdentityPools`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
```

```
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        let listPoolsInput = ListIdentityPoolsInput(maxResults: 25)
        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'nextToken' field in
"ListIdentityPoolsOutput".
        let pages = cognitoIdentityClient.listIdentityPoolsPaginated(input:
listPoolsInput)

        do {
            for try await page in pages {
                guard let identityPools = page.identityPools else {
                    print("ERROR: listIdentityPoolsPaginated returned nil
contents.")
                    continue
                }

                /// Read pages of identity pools from Cognito until one is found
                /// whose name matches the one specified in the `name` parameter.
                /// Return the matching pool's ID.

                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }
        } catch {
            print("ERROR: getIdentityPoolID:", dump(error))
            throw error
        }

        return nil
    }
```

Obtenha o ID de um banco de identidades existente ou crie-o se ainda não existir.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned
    ///
    /// - Returns: A string containing the ID of the specified identity pool.
    ///   Returns `nil` if there's an error or if the pool isn't found.
    ///
    public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
        // See if the pool already exists. If it doesn't, create it.

        do {
            guard let poolId = try await getIdentityPoolID(name: name) else {
                return try await createIdentityPool(name: name)
            }

            return poolId
        } catch {
            print("ERROR: getOrCreateIdentityPoolID:", dump(error))
            throw error
        }
    }
```

- Para ter mais informações, consulte o Guia do desenvolvedor do AWS SDK para Swift.
- Para obter detalhes da API, consulte ListIdentityPools a referência da API AWS SDK for Swift.

Exemplos de código do Provedor de Identidade do Amazon Cognito usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon Cognito Identity Provider.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`AdminGetUser`

O código de exemplo a seguir mostra como usar `AdminGetUser`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
```

```
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }
```

- Para obter detalhes da API, consulte AdminGetUser a referência da API AWS SDK for Swift.

`AdminInitiateAuth`

O código de exemplo a seguir mostra como usar `AdminInitiateAuth`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Begin an authentication session.
    ///
```

```
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
```

```
            return nil
        }
    }
```

- Para obter detalhes da API, consulte AdminInitiateAuth a referência da API AWS SDK for Swift.

`AdminRespondToAuthChallenge`

O código de exemplo a seguir mostra como usar `AdminRespondToAuthChallenge`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
 userName: String,
                                     clientId: String, userPoolId: String, mfaCode:
 String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")
```

```
        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }
```

- Para obter detalhes da API, consulte AdminRespondToAuthChallenge a referência da API AWS SDK for Swift.

`AssociateSoftwareToken`

O código de exemplo a seguir mostra como usar `AssociateSoftwareToken`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient, authSession:
 String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
```

```
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the app's
MFA.")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte AssociateSoftwareToken a referência da API AWS SDK for Swift.

`ConfirmSignUp`

O código de exemplo a seguir mostra como usar `ConfirmSignUp`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
```

```
///   - clientId: The app client ID the user is signing up for.
///   - userName: The username of the user whose code is being sent.
///   - code: The user's confirmation code.
///
/// - Returns: `true` if the code was successfully confirmed; otherwise `false`.
func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId: String,
                   userName: String, code: String) async -> Bool {
    do {
        _ = try await cipClient.confirmSignUp(
            input: ConfirmSignUpInput(
                clientId: clientId,
                confirmationCode: code,
                username: userName
            )
        )

        print("=====> \(userName) has been confirmed.")
        return true
    } catch {
        print("=====> \(userName)'s code was entered incorrectly.")
        return false
    }
}
```

- Para obter detalhes da API, consulte ConfirmSignUp a referência da API AWS SDK for Swift.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    do {
        let output = try await cognitoClient.listUsers(
```

```
                input: ListUsersInput(
                    userPoolId: poolId
                )
            )

            guard let users = output.users else {
                print("No users found.")
                return
            }

            print("\(users.count) user(s) found.")
            for user in users {
                print("  \(user.username ?? "<unknown>")")
            }
        } catch _ as NotAuthorizedException {
            print("*** Please authenticate with AWS before using this command.")
            return
        } catch _ as ResourceNotFoundException {
            print("*** The specified User Pool was not found.")
            return
        } catch {
            print("*** An unexpected type of error occurred.")
            return
        }
```

- Para obter detalhes da API, consulte ListUsers a referência da API AWS SDK for Swift.

`ResendConfirmationCode`

O código de exemplo a seguir mostra como usar `ResendConfirmationCode`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider
```

```
    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient, clientId:
String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
else {
                print("*** Unable to get the delivery method for the resent code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user \(userName).")
            return false
        }
    }
```

- Para obter detalhes da API, consulte ResendConfirmationCode a referência da API AWS SDK for Swift.

SignUp

O código de exemplo a seguir mostra como usar `SignUp`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
 userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )

            print("=====> User \(userName) signed up.")
```

```
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException {
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }
```

- Para obter detalhes da API, consulte SignUp a referência da API AWS SDK for Swift.

`VerifySoftwareToken`

O código de exemplo a seguir mostra como usar `VerifySoftwareToken`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
```

```
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }
```

- Para obter detalhes da API, consulte VerifySoftwareToken a referência da API AWS SDK for Swift.

Cenários

Inscrever um usuário em um grupo de usuários que exija MFA

O exemplo de código a seguir mostra como:

- Inscrever e confirmar um usuário com nome de usuário, senha e endereço de e-mail.
- Configurar a autenticação multifator associando uma aplicação de MFA ao usuário.
- Faça login usando uma senha e um código de MFA.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "cognito-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
```

```
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "cognito-scenario",
            dependencies: [
                .product(name: "AWSCognitoIdentityProvider", package: "aws-sdk-
swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo de código do Swift.

```
// An example demonstrating various features of Amazon Cognito. Before running
// this Swift code example, set up your development environment, including
// your credentials.
//
// For more information, see the following documentation:
// https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
//
// TIP: To set up the required user pool, run the AWS Cloud Development Kit
// (AWS CDK) script provided in this GitHub repo at
// resources/cdk/cognito_scenario_user_pool_with_mfa.
//
// This example performs the following functions:
//
// 1. Invokes the signUp method to sign up a user.
// 2. Invokes the adminGetUser method to get the user's confirmation status.
// 3. Invokes the ResendConfirmationCode method if the user requested another
//    code.
// 4. Invokes the confirmSignUp method.
// 5. Invokes the initiateAuth to sign in. This results in being prompted to
//    set up TOTP (time-based one-time password). (The response is
```

```
//    “ChallengeName”: “MFA_SETUP”).
// 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
//    key. This can be used with Google Authenticator.
// 7. Invokes the VerifySoftwareToken method to verify the TOTP and register
//    for MFA.
// 8. Invokes the AdminInitiateAuth to sign in again. This results in being
//    prompted to submit a TOTP (Response: “ChallengeName”:
//    “SOFTWARE_TOKEN_MFA”).
// 9. Invokes the AdminRespondToAuthChallenge to get back a token.

import ArgumentParser
import Foundation

import AWSClientRuntime
import AWSCognitoIdentityProvider

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The application clientId.")
    var clientId: String
    @Argument(help: "The user pool ID to use.")
    var poolId: String
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "cognito-scenario",
        abstract: """
        Demonstrates various features of Amazon Cognito.
        """,
        discussion: """
        """
    )

    /// Prompt for an input string of at least a minimum length.
    ///
    /// - Parameters:
    ///   - prompt: The prompt string to display.
    ///   - minLength: The minimum number of characters to allow in the
    ///     response. Default value is 0.
    ///
    /// - Returns: The entered string.
    func stringRequest(_ prompt: String, minLength: Int = 1) -> String {
        while true {
            print(prompt, terminator: "")
```

```
            let str = readLine()

            guard let str else {
                continue
            }
            if str.count >= minLength {
                return str
            } else {
                print("*** Response must be at least \(minLength) character(s)
long.")
            }
        }
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(_ prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
```

```
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )
```

```
            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException {
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient, clientId:
String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
else {
```

```
                print("*** Unable to get the delivery method for the resent code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user \(userName).")
            return false
        }
    }

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise `false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId: String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }

    /// Begin an authentication session.
    ///
```

```
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
```

```
            return nil
        }
    }

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient, authSession:
String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the app's
MFA.")
            return nil
        }
    }

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
```

```
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
```

```
   ///   - cipClient: The `CognitoIdentityProviderClient` to use.
   ///   - userName: The user's username.
   ///   - clientId: The app client ID.
   ///   - userPoolId: The user pool to sign into.
   ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
   ///     authenticator.
   ///   - session: The authentication session to continue processing.
   func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                    clientId: String, userPoolId: String, mfaCode:
String,
                                    session: String) async {
       print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

       var challengeResponsesOb: [String: String] = [:]
       challengeResponsesOb["USERNAME"] = userName
       challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

       do {
           let output = try await cipClient.adminRespondToAuthChallenge(
               input: AdminRespondToAuthChallengeInput(
                   challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                   challengeResponses: challengeResponsesOb,
                   clientId: clientId,
                   session: session,
                   userPoolId: userPoolId
               )
           )

           guard let authenticationResult = output.authenticationResult else {
               print("*** Unable to get authentication result.")
               return
           }

           print("=====> Authentication result (JWTs are redacted):")
           print(authenticationResult)
       } catch _ as SoftwareTokenMFANotFoundException {
           print("*** The specified user pool isn't configured for MFA.")
           return
       } catch _ as CodeMismatchException {
           print("*** The specified MFA code doesn't match the expected value.")
           return
       } catch _ as UserNotFoundException {
```

```
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await
CognitoIdentityProviderClient.CognitoIdentityProviderClientConfiguration(region:
region)
        let cipClient = CognitoIdentityProviderClient(config: config)

        print("""
              This example collects information about a user, then creates that user
in the
              specified user pool. Then, it enables Multi-Factor Authentication
(MFA) for that
              user by associating an authenticator application (such as Google
Authenticator
              or a password manager that supports TOTP). Then, the user uses a code
from their
              authenticator application to sign in.

              """)

        let userName = stringRequest("Please enter a new username: ")
        let password = stringRequest("Enter a password: ")
        let email = stringRequest("Enter your email address: ", minLength: 5)

        // Submit the sign-up request to AWS.

        print("==> Signing up user \(userName)...")
        if await signUp(cipClient: cipClient, clientId: clientId,
                        userName: userName, password: password,
                        email: email) == false {
```

```
            return
        }

        // Check the user's status. This time, it should come back "unconfirmed".

        print("==> Getting the status of user \(userName) from the user pool (should
be 'unconfirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName, userPoolId:
poolId) == false {
            return
        }

        // Ask the user if they want a replacement code sent, such as if the
        // code hasn't arrived yet. If the user responds with a "yes," send a
        // new code.

        if yesNoRequest("==> A confirmation code was sent to \(userName). Would you
like to send a new code (Y/N)? ") {
            print("==> Sending a new confirmation code...")
            if await resendConfirmationCode(cipClient: cipClient, clientId:
clientId, userName: userName) == false {
                return
            }
        }

        // Ask the user to enter the confirmation code, then send it to Amazon
        // Cognito to verify it.

        let code = stringRequest("==> Enter the confirmation code sent to
\(userName): ")
        if await confirmSignUp(cipClient: cipClient, clientId: clientId, userName:
userName, code: code) == false {
            // The code didn't match. Your application may wish to offer to
            // re-send the confirmation code here and try again.
            return
        }

        // Check the user's status again. This time it should come back
        // "confirmed".

        print("==> Rechecking status of user \(userName) in the user pool (should be
'confirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName, userPoolId:
poolId) == false {
```

```
            return
        }
        // Check the challenge mode. Here, it should be "mfaSetup", indicating
        // that the user needs to add MFA before using it. This returns a
        // session that can be used to register MFA, or nil if an error occurs.

        let authSession = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                             userName: userName, password:
password,
                                             userPoolId: poolId)
        if authSession == nil {
            return
        }

        // Ask Cognito for an MFA secret token that the user should enter into
        // their authenticator software (such as Google Authenticator) or
        // password manager to configure it for this user account. This
        // returns a new session that should be used for the new stage of the
        // authentication process.

        let newSession = await getSecretForAppMFA(cipClient: cipClient, authSession:
authSession)
        if newSession == nil {
            return
        }

        // Ask the user to enter the current 6-digit code displayed by their
        // authenticator. Then verify that it matches the value expected for
        // the session.

        let mfaCode1 = stringRequest("==> Enter the 6-digit code displayed in your
authenticator: ",
                                     minLength: 6)
        await verifyTOTP(cipClient: cipClient, session: newSession, mfaCode:
mfaCode1)

        // Ask the user to authenticate now that the authenticator has been
        // configured. This creates a new session using the user's username
        // and password as already entered.

        print("\nNow starting the sign-in process for user \(userName)...\n")

        let session2 = await initiateAuth(cipClient: cipClient, clientId: clientId,
```

```
                                    userName: userName, password: password,
 userPoolId: poolId)
        guard let session2 else {
            return
        }

        // Now that we have a new auth session, `session2`, ask the user for a
        // new 6-digit code from their authenticator, and send it to the auth
        // session.

        let mfaCode2 = stringRequest("==> Wait for your authenticator to show a new
 6-digit code, then enter it: ",
                                    minLength: 6)
        await adminRespondToAuthChallenge(cipClient: cipClient, userName: userName,
                                          clientId: clientId, userPoolId: poolId,
                                          mfaCode: mfaCode2, session: session2)
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice

- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Exemplos de código do DynamoDB usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.
- Gravar dados de filmes na tabela usando um arquivo JSON de exemplo.

- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Uma classe Swift que gerencia chamadas do DynamoDB para o SDK para Swift.

```
import AWSDynamoDB
import Foundation

/// An enumeration of error codes representing issues that can arise when using
/// the `MovieTable` class.
enum MoviesError: Error {
    /// The specified table wasn't found or couldn't be created.
    case TableNotFound
    /// The specified item wasn't found or couldn't be created.
    case ItemNotFound
    /// The Amazon DynamoDB client is not properly initialized.
    case UninitializedClient
    /// The table status reported by Amazon DynamoDB is not recognized.
    case StatusUnknown
    /// One or more specified attribute values are invalid or missing.
    case InvalidAttributes
}

/// A class representing an Amazon DynamoDB table containing movie
/// information.
public class MovieTable {
    var ddbClient: DynamoDBClient?
    let tableName: String

    /// Create an object representing a movie table in an Amazon DynamoDB
    /// database.
    ///
```

```
    /// - Parameters:
    ///   - region: The optional Amazon Region to create the database in.
    ///   - tableName: The name to assign to the table. If not specified, a
    ///     random table name is generated automatically.
    ///
    /// > Note: The table is not necessarily available when this function
    /// returns. Use `tableExists()` to check for its availability, or
    /// `awaitTableActive()` to wait until the table's status is reported as
    /// ready to use by Amazon DynamoDB.
    ///
    init(region: String? = nil, tableName: String) async throws {
        do {
            let config = try await DynamoDBClient.DynamoDBClientConfiguration()
            if let region = region {
                config.region = region
            }

            self.ddbClient = DynamoDBClient(config: config)
            self.tableName = tableName

            try await self.createTable()
        } catch {
            print("ERROR: ", dump(error, name: "Initializing Amazon DynamoDBClient
client"))
            throw error
        }
    }

    ///
    /// Create a movie table in the Amazon DynamoDB data store.
    ///
    private func createTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = CreateTableInput(
                attributeDefinitions: [
                    DynamoDBClientTypes.AttributeDefinition(attributeName: "year",
attributeType: .n),
                    DynamoDBClientTypes.AttributeDefinition(attributeName: "title",
attributeType: .s)
                ],
```

```
                billingMode: DynamoDBClientTypes.BillingMode.payPerRequest,
                keySchema: [
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "year",
keyType: .hash),
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "title",
keyType: .range)
                ],
                tableName: self.tableName
            )
            let output = try await client.createTable(input: input)
            if output.tableDescription == nil {
                throw MoviesError.TableNotFound
            }
        } catch {
            print("ERROR: createTable:", dump(error))
            throw error
        }
    }


    /// Check to see if the table exists online yet.
    ///
    /// - Returns: `true` if the table exists, or `false` if not.
    ///
    func tableExists() async throws -> Bool {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DescribeTableInput(
                tableName: tableName
            )
            let output = try await client.describeTable(input: input)
            guard let description = output.table else {
                throw MoviesError.TableNotFound
            }

            return description.tableName == self.tableName
        } catch {
            print("ERROR: tableExists:", dump(error))
            throw error
        }
    }
```

```
    ///
    /// Waits for the table to exist and for its status to be active.
    ///
    func awaitTableActive() async throws {
        while try (await self.tableExists() == false) {
            do {
                let duration = UInt64(0.25 * 1_000_000_000) // Convert .25 seconds
to nanoseconds.
                try await Task.sleep(nanoseconds: duration)
            } catch {
                print("Sleep error:", dump(error))
            }
        }

        while try (await self.getTableStatus() != .active) {
            do {
                let duration = UInt64(0.25 * 1_000_000_000) // Convert .25 seconds
to nanoseconds.
                try await Task.sleep(nanoseconds: duration)
            } catch {
                print("Sleep error:", dump(error))
            }
        }
    }


    ///
    /// Deletes the table from Amazon DynamoDB.
    ///
    func deleteTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteTableInput(
                tableName: self.tableName
            )
            _ = try await client.deleteTable(input: input)
        } catch {
            print("ERROR: deleteTable:", dump(error))
            throw error
```

```
        }
    }


    /// Get the table's status.
    ///
    /// - Returns: The table status, as defined by the
    ///   `DynamoDBClientTypes.TableStatus` enum.
    ///
    func getTableStatus() async throws -> DynamoDBClientTypes.TableStatus {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DescribeTableInput(
                tableName: self.tableName
            )
            let output = try await client.describeTable(input: input)
            guard let description = output.table else {
                throw MoviesError.TableNotFound
            }
            guard let status = description.tableStatus else {
                throw MoviesError.StatusUnknown
            }
            return status
        } catch {
            print("ERROR: getTableStatus:", dump(error))
            throw error
        }
    }


    /// Populate the movie database from the specified JSON file.
    ///
    /// - Parameter jsonPath: Path to a JSON file containing movie data.
    ///
    func populate(jsonPath: String) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Create a Swift `URL` and use it to load the file into a `Data`
```

```
            // object. Then decode the JSON into an array of `Movie` objects.

            let fileUrl = URL(fileURLWithPath: jsonPath)
            let jsonData = try Data(contentsOf: fileUrl)

            var movieList = try JSONDecoder().decode([Movie].self, from: jsonData)

            // Truncate the list to the first 200 entries or so for this example.

            if movieList.count > 200 {
                movieList = Array(movieList[...199])
            }

            // Before sending records to the database, break the movie list into
            // 25-entry chunks, which is the maximum size of a batch item request.

            let count = movieList.count
            let chunks = stride(from: 0, to: count, by: 25).map {
                Array(movieList[$0 ..< Swift.min($0 + 25, count)])
            }

            // For each chunk, create a list of write request records and populate
            // them with `PutRequest` requests, each specifying one movie from the
            // chunk. Once the chunk's items are all in the `PutRequest` list,
            // send them to Amazon DynamoDB using the
            // `DynamoDBClient.batchWriteItem()` function.

            for chunk in chunks {
                var requestList: [DynamoDBClientTypes.WriteRequest] = []

                for movie in chunk {
                    let item = try await movie.getAsItem()
                    let request = DynamoDBClientTypes.WriteRequest(
                        putRequest: .init(
                            item: item
                        )
                    )
                    requestList.append(request)
                }

                let input = BatchWriteItemInput(requestItems: [tableName:
requestList])
                _ = try await client.batchWriteItem(input: input)
            }
```

```
        } catch {
            print("ERROR: populate:", dump(error))
            throw error
        }
    }


    /// Add a movie specified as a `Movie` structure to the Amazon DynamoDB
    /// table.
    ///
    /// - Parameter movie: The `Movie` to add to the table.
    ///
    func add(movie: Movie) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Get a DynamoDB item containing the movie data.
            let item = try await movie.getAsItem()

            // Send the `PutItem` request to Amazon DynamoDB.

            let input = PutItemInput(
                item: item,
                tableName: self.tableName
            )
            _ = try await client.putItem(input: input)
        } catch {
            print("ERROR: add movie:", dump(error))
            throw error
        }
    }


    /// Given a movie's details, add a movie to the Amazon DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title as a `String`.
    ///   - year: The release year of the movie (`Int`).
    ///   - rating: The movie's rating if available (`Double`; default is
    ///     `nil`).
    ///   - plot: A summary of the movie's plot (`String`; default is `nil`,
    ///     indicating no plot summary is available).
```

```
    ///
    func add(title: String, year: Int, rating: Double? = nil,
             plot: String? = nil) async throws
    {
        do {
            let movie = Movie(title: title, year: year, rating: rating, plot: plot)
            try await self.add(movie: movie)
        } catch {
            print("ERROR: add with fields:", dump(error))
            throw error
        }
    }


    /// Return a `Movie` record describing the specified movie from the Amazon
    /// DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The movie's release year (`Int`).
    ///
    /// - Throws: `MoviesError.ItemNotFound` if the movie isn't in the table.
    ///
    /// - Returns: A `Movie` record with the movie's details.
    func get(title: String, year: Int) async throws -> Movie {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = GetItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            let output = try await client.getItem(input: input)
            guard let item = output.item else {
                throw MoviesError.ItemNotFound
            }

            let movie = try Movie(withItem: item)
            return movie
```

```
        } catch {
            print("ERROR: get:", dump(error))
            throw error
        }
    }


    /// Get all the movies released in the specified year.
    ///
    /// - Parameter year: The release year of the movies to return.
    ///
    /// - Returns: An array of `Movie` objects describing each matching movie.
    ///
    func getMovies(fromYear year: Int) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = QueryInput(
                expressionAttributeNames: [
                    "#y": "year"
                ],
                expressionAttributeValues: [
                    ":y": .n(String(year))
                ],
                keyConditionExpression: "#y = :y",
                tableName: self.tableName
            )
            // Use "Paginated" to get all the movies.
            // This lets the SDK handle the 'lastEvaluatedKey' property in
"QueryOutput".

            let pages = client.queryPaginated(input: input)

            var movieList: [Movie] = []
            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Convert the found movies into `Movie` objects and return an array
                // of them.
```

```
                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList
        } catch {
            print("ERROR: getMovies:", dump(error))
            throw error
        }
    }


    /// Return an array of `Movie` objects released in the specified range of
    /// years.
    ///
    /// - Parameters:
    ///   - firstYear: The first year of movies to return.
    ///   - lastYear: The last year of movies to return.
    ///   - startKey: A starting point to resume processing; always use `nil`.
    ///
    /// - Returns: An array of `Movie` objects describing the matching movies.
    ///
    /// > Note: The `startKey` parameter is used by this function when
    ///   recursively calling itself, and should always be `nil` when calling
    ///   directly.
    ///
    func getMovies(firstYear: Int, lastYear: Int,
                   startKey: [Swift.String: DynamoDBClientTypes.AttributeValue]? =
nil)
        async throws -> [Movie]
    {
        do {
            var movieList: [Movie] = []

            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = ScanInput(
                consistentRead: true,
                exclusiveStartKey: startKey,
                expressionAttributeNames: [
```

```
                    "#y": "year" // `year` is a reserved word, so use `#y` instead.
                ],
                expressionAttributeValues: [
                    ":y1": .n(String(firstYear)),
                    ":y2": .n(String(lastYear))
                ],
                filterExpression: "#y BETWEEN :y1 AND :y2",
                tableName: self.tableName
            )

            let pages = client.scanPaginated(input: input)

            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Build an array of `Movie` objects for the returned items.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList

        } catch {
            print("ERROR: getMovies with scan:", dump(error))
            throw error
        }
    }


    /// Update the specified movie with new `rating` and `plot` information.
    ///
    /// - Parameters:
    ///   - title: The title of the movie to update.
    ///   - year: The release year of the movie to update.
    ///   - rating: The new rating for the movie.
    ///   - plot: The new plot summary string for the movie.
    ///
    /// - Returns: An array of mappings of attribute names to their new
    ///   listing each item actually changed. Items that didn't need to change
```

```
    ///   aren't included in this list. `nil` if no changes were made.
    ///
    func update(title: String, year: Int, rating: Double? = nil, plot: String? =
nil) async throws
        -> [Swift.String: DynamoDBClientTypes.AttributeValue]?
    {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Build the update expression and the list of expression attribute
            // values. Include only the information that's changed.

            var expressionParts: [String] = []
            var attrValues: [Swift.String: DynamoDBClientTypes.AttributeValue] = [:]

            if rating != nil {
                expressionParts.append("info.rating=:r")
                attrValues[":r"] = .n(String(rating!))
            }
            if plot != nil {
                expressionParts.append("info.plot=:p")
                attrValues[":p"] = .s(plot!)
            }
            let expression = "set \(expressionParts.joined(separator: ", "))"

            let input = UpdateItemInput(
                // Create substitution tokens for the attribute values, to ensure
                // no conflicts in expression syntax.
                expressionAttributeValues: attrValues,
                // The key identifying the movie to update consists of the release
                // year and title.
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                returnValues: .updatedNew,
                tableName: self.tableName,
                updateExpression: expression
            )
            let output = try await client.updateItem(input: input)
```

```
            guard let attributes: [Swift.String: DynamoDBClientTypes.AttributeValue]
 = output.attributes else {
                throw MoviesError.InvalidAttributes
            }
            return attributes
        } catch {
            print("ERROR: update:", dump(error))
            throw error
        }
    }


    /// Delete a movie, given its title and release year.
    ///
    /// - Parameters:
    ///   - title: The movie's title.
    ///   - year: The movie's release year.
    ///
    func delete(title: String, year: Int) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            _ = try await client.deleteItem(input: input)
        } catch {
            print("ERROR: delete:", dump(error))
            throw error
        }
    }
}
```

As estruturas usadas pela MovieTable classe para representar filmes.

```
import Foundation
import AWSDynamoDB

/// The optional details about a movie.
public struct Details: Codable {
    /// The movie's rating, if available.
    var rating: Double?
    /// The movie's plot, if available.
    var plot: String?
}

/// A structure describing a movie. The `year` and `title` properties are
/// required and are used as the key for Amazon DynamoDB operations. The
/// `info` sub-structure's two properties, `rating` and `plot`, are optional.
public struct Movie: Codable {
    /// The year in which the movie was released.
    var year: Int
    /// The movie's title.
    var title: String
    /// A `Details` object providing the optional movie rating and plot
    /// information.
    var info: Details

    /// Create a `Movie` object representing a movie, given the movie's
    /// details.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The year in which the movie was released (`Int`).
    ///   - rating: The movie's rating (optional `Double`).
    ///   - plot: The movie's plot (optional `String`)
    init(title: String, year: Int, rating: Double? = nil, plot: String? = nil) {
        self.title = title
        self.year = year

        self.info = Details(rating: rating, plot: plot)
    }

    /// Create a `Movie` object representing a movie, given the movie's
    /// details.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
```

```
    ///   - year: The year in which the movie was released (`Int`).
    ///   - info: The optional rating and plot information for the movie in a
    ///     `Details` object.
    init(title: String, year: Int, info: Details?){
        self.title = title
        self.year = year

        if info != nil {
            self.info = info!
        } else {
            self.info = Details(rating: nil, plot: nil)
        }
    }

    ///
    /// Return a new `MovieTable` object, given an array mapping string to Amazon
    /// DynamoDB attribute values.
    ///
    /// - Parameter item: The item information provided to the form used by
    ///   DynamoDB. This is an array of strings mapped to
    ///   `DynamoDBClientTypes.AttributeValue` values.
    init(withItem item: [Swift.String:DynamoDBClientTypes.AttributeValue]) throws  {
        // Read the attributes.

        guard let titleAttr = item["title"],
              let yearAttr = item["year"] else {
            throw MoviesError.ItemNotFound
        }
        let infoAttr = item["info"] ?? nil

        // Extract the values of the title and year attributes.

        if case .s(let titleVal) = titleAttr {
            self.title = titleVal
        } else {
            throw MoviesError.InvalidAttributes
        }

        if case .n(let yearVal) = yearAttr {
            self.year = Int(yearVal)!
        } else {
            throw MoviesError.InvalidAttributes
        }
```

```
        // Extract the rating and/or plot from the `info` attribute, if
        // they're present.

        var rating: Double? = nil
        var plot: String? = nil

        if infoAttr != nil, case .m(let infoVal) = infoAttr {
            let ratingAttr = infoVal["rating"] ?? nil
            let plotAttr = infoVal["plot"] ?? nil

            if ratingAttr != nil, case .n(let ratingVal) = ratingAttr {
                rating = Double(ratingVal) ?? nil
            }
            if plotAttr != nil, case .s(let plotVal) = plotAttr {
                plot = plotVal
            }
        }

        self.info = Details(rating: rating, plot: plot)
    }

    ///
    /// Return an array mapping attribute names to Amazon DynamoDB attribute
    /// values, representing the contents of the `Movie` record as a DynamoDB
    /// item.
    ///
    /// - Returns: The movie item as an array of type
    ///   `[Swift.String:DynamoDBClientTypes.AttributeValue]`.
    ///
    func getAsItem() async throws ->
[Swift.String:DynamoDBClientTypes.AttributeValue]  {
        // Build the item record, starting with the year and title, which are
        // always present.

        var item: [Swift.String:DynamoDBClientTypes.AttributeValue] = [
            "year": .n(String(self.year)),
            "title": .s(self.title)
        ]

        // Add the `info` field with the rating and/or plot if they're
        // available.

        var details: [Swift.String:DynamoDBClientTypes.AttributeValue] = [:]
        if (self.info.rating != nil || self.info.plot != nil) {
```

```
            if self.info.rating != nil {
                details["rating"] = .n(String(self.info.rating!))
            }
            if self.info.plot != nil {
                details["plot"] = .s(self.info.plot!)
            }
        }
        item["info"] = .m(details)

        return item
    }
 }
```

Um programa que usa a MovieTable classe para acessar um banco de dados do DynamoDB.

```
import ArgumentParser
import ClientRuntime
import Foundation

import AWSDynamoDB


@testable import MovieList

extension String {
    // Get the directory if the string is a file path.
    func directory() -> String {
        guard let lastIndex = lastIndex(of: "/") else {
            print("Error: String directory separator not found.")
            return ""
        }
        return String(self[...lastIndex])
    }
}

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The path of the sample movie data JSON file.")
    var jsonPath: String = #file.directory() + "../../../../../resources/
sample_files/movies.json"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion: String?
```

```
    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    /// Configuration details for the command.
    static var configuration = CommandConfiguration(
        commandName: "basics",
        abstract: "A basic scenario demonstrating the usage of Amazon DynamoDB.",
        discussion: """
        An example showing how to use Amazon DynamoDB to perform a series of
        common database activities on a simple movie database.
        """
    )

    /// Called by ``main()`` to asynchronously run the AWS example.
    func runAsync() async throws {
        print("Welcome to the AWS SDK for Swift basic scenario for Amazon
DynamoDB!")

        //======================================================================
        // 1. Create the table. The Amazon DynamoDB table is represented by
        //    the `MovieTable` class.
        //======================================================================

        let tableName = "ddb-movies-sample-\(Int.random(in: 1 ... Int.max))"

        print("Creating table \"\(tableName)\"...")

        let movieDatabase = try await MovieTable(region: awsRegion,
                                                 tableName: tableName)

        print("\nWaiting for table to be ready to use...")
        try await movieDatabase.awaitTableActive()
```

```
        //=====================================================================
        // 2. Add a movie to the table.
        //=====================================================================

        print("\nAdding a movie...")
        try await movieDatabase.add(title: "Avatar: The Way of Water", year: 2022)
        try await movieDatabase.add(title: "Not a Real Movie", year: 2023)

        //=====================================================================
        // 3. Update the plot and rating of the movie using an update
        //    expression.
        //=====================================================================

        print("\nAdding details to the added movie...")
        _ = try await movieDatabase.update(title: "Avatar: The Way of Water", year:
2022,
                                           rating: 9.2, plot: "It's a sequel.")

        //=====================================================================
        // 4. Populate the table from the JSON file.
        //=====================================================================

        print("\nPopulating the movie database from JSON...")
        try await movieDatabase.populate(jsonPath: jsonPath)

        //=====================================================================
        // 5. Get a specific movie by key. In this example, the key is a
        //    combination of `title` and `year`.
        //=====================================================================

        print("\nLooking for a movie in the table...")
        let gotMovie = try await movieDatabase.get(title: "This Is the End", year:
2013)

        print("Found the movie \"\(gotMovie.title)\", released in
\(gotMovie.year).")
        print("Rating: \(gotMovie.info.rating ?? 0.0).")
        print("Plot summary: \(gotMovie.info.plot ?? "None.")")

        //=====================================================================
        // 6. Delete a movie.
        //=====================================================================
```

```
        print("\nDeleting the added movie...")
        try await movieDatabase.delete(title: "Avatar: The Way of Water", year:
 2022)

        //=====================================================================
        // 7. Use a query with a key condition expression to return all movies
        //    released in a given year.
        //=====================================================================

        print("\nGetting movies released in 1994...")
        let movieList = try await movieDatabase.getMovies(fromYear: 1994)
        for movie in movieList {
            print("    \(movie.title)")
        }

        //=====================================================================
        // 8. Use `scan()` to return movies released in a range of years.
        //=====================================================================

        print("\nGetting movies released between 1993 and 1997...")
        let scannedMovies = try await movieDatabase.getMovies(firstYear: 1993,
 lastYear: 1997)
        for movie in scannedMovies {
            print("    \(movie.title) (\(movie.year))")
        }

        //=====================================================================
        // 9. Delete the table.
        //=====================================================================

        print("\nDeleting the table...")
        try await movieDatabase.deleteTable()
    }
}

@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
```

```
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Ações

`BatchGetItem`

O código de exemplo a seguir mostra como usar `BatchGetItem`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Gets an array of `Movie` objects describing all the movies in the
```

```
    /// specified list. Any movies that aren't found in the list have no
    /// corresponding entry in the resulting array.
    ///
    /// - Parameters
    ///     - keys: An array of tuples, each of which specifies the title and
    ///       release year of a movie to fetch from the table.
    ///
    /// - Returns:
    ///     - An array of `Movie` objects describing each match found in the
    ///     table.
    ///
    /// - Throws:
    ///     - `MovieError.ClientUninitialized` if the DynamoDB client has not
    ///     been initialized.
    ///     - DynamoDB errors are thrown without change.
    func batchGet(keys: [(title: String, year: Int)]) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MovieError.ClientUninitialized
            }

            var movieList: [Movie] = []
            var keyItems: [[Swift.String: DynamoDBClientTypes.AttributeValue]] = []

            // Convert the list of keys into the form used by DynamoDB.

            for key in keys {
                let item: [Swift.String: DynamoDBClientTypes.AttributeValue] = [
                    "title": .s(key.title),
                    "year": .n(String(key.year))
                ]
                keyItems.append(item)
            }

            // Create the input record for `batchGetItem()`. The list of requested
            // items is in the `requestItems` property. This array contains one
            // entry for each table from which items are to be fetched. In this
            // example, there's only one table containing the movie data.
            //
            // If we wanted this program to also support searching for matches
            // in a table of book data, we could add a second `requestItem`
            // mapping the name of the book table to the list of items we want to
            // find in it.
            let input = BatchGetItemInput(
```

```
                requestItems: [
                    self.tableName: .init(
                        consistentRead: true,
                        keys: keyItems
                    )
                ]
            )

            // Fetch the matching movies from the table.

            let output = try await client.batchGetItem(input: input)

            // Get the set of responses. If there aren't any, return the empty
            // movie list.

            guard let responses = output.responses else {
                return movieList
            }

            // Get the list of matching items for the table with the name
            // `tableName`.

            guard let responseList = responses[self.tableName] else {
                return movieList
            }

            // Create `Movie` items for each of the matching movies in the table
            // and add them to the `MovieList` array.

            for response in responseList {
                try movieList.append(Movie(withItem: response))
            }

            return movieList
        } catch {
            print("ERROR: batchGet", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte BatchGetItem a referência da API AWS SDK for Swift.

`BatchWriteItem`

O código de exemplo a seguir mostra como usar `BatchWriteItem`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Populate the movie database from the specified JSON file.
    ///
    /// - Parameter jsonPath: Path to a JSON file containing movie data.
    ///
    func populate(jsonPath: String) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Create a Swift `URL` and use it to load the file into a `Data`
            // object. Then decode the JSON into an array of `Movie` objects.

            let fileUrl = URL(fileURLWithPath: jsonPath)
            let jsonData = try Data(contentsOf: fileUrl)

            var movieList = try JSONDecoder().decode([Movie].self, from: jsonData)

            // Truncate the list to the first 200 entries or so for this example.

            if movieList.count > 200 {
                movieList = Array(movieList[...199])
            }

            // Before sending records to the database, break the movie list into
            // 25-entry chunks, which is the maximum size of a batch item request.

            let count = movieList.count
```

```
            let chunks = stride(from: 0, to: count, by: 25).map {
                Array(movieList[$0 ..< Swift.min($0 + 25, count)])
            }

            // For each chunk, create a list of write request records and populate
            // them with `PutRequest` requests, each specifying one movie from the
            // chunk. Once the chunk's items are all in the `PutRequest` list,
            // send them to Amazon DynamoDB using the
            // `DynamoDBClient.batchWriteItem()` function.

            for chunk in chunks {
                var requestList: [DynamoDBClientTypes.WriteRequest] = []

                for movie in chunk {
                    let item = try await movie.getAsItem()
                    let request = DynamoDBClientTypes.WriteRequest(
                        putRequest: .init(
                            item: item
                        )
                    )
                    requestList.append(request)
                }

                let input = BatchWriteItemInput(requestItems: [tableName:
requestList])
                _ = try await client.batchWriteItem(input: input)
            }
        } catch {
            print("ERROR: populate:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte BatchWriteItem a referência da API AWS SDK for Swift.

`CreateTable`

O código de exemplo a seguir mostra como usar `CreateTable`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    ///
    /// Create a movie table in the Amazon DynamoDB data store.
    ///
    private func createTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = CreateTableInput(
                attributeDefinitions: [
                    DynamoDBClientTypes.AttributeDefinition(attributeName: "year",
 attributeType: .n),
                    DynamoDBClientTypes.AttributeDefinition(attributeName: "title",
 attributeType: .s)
                ],
                billingMode: DynamoDBClientTypes.BillingMode.payPerRequest,
                keySchema: [
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "year",
 keyType: .hash),
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "title",
 keyType: .range)
                ],
                tableName: self.tableName
            )
            let output = try await client.createTable(input: input)
            if output.tableDescription == nil {
                throw MoviesError.TableNotFound
            }
        } catch {
            print("ERROR: createTable:", dump(error))
```

```
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateTable a referência da API AWS SDK for Swift.

`DeleteItem`

O código de exemplo a seguir mostra como usar `DeleteItem`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Delete a movie, given its title and release year.
    ///
    /// - Parameters:
    ///   - title: The movie's title.
    ///   - year: The movie's release year.
    ///
    func delete(title: String, year: Int) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
```

```
            )
            _ = try await client.deleteItem(input: input)
        } catch {
            print("ERROR: delete:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteItem a referência da API AWS SDK for Swift.

`DeleteTable`

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    ///
    /// Deletes the table from Amazon DynamoDB.
    ///
    func deleteTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteTableInput(
                tableName: self.tableName
            )
            _ = try await client.deleteTable(input: input)
        } catch {
            print("ERROR: deleteTable:", dump(error))
            throw error
```

```
        }
    }
```

- Para obter detalhes da API, consulte DeleteTable a referência da API AWS SDK for Swift.

GetItem

O código de exemplo a seguir mostra como usar `GetItem`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Return a `Movie` record describing the specified movie from the Amazon
    /// DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The movie's release year (`Int`).
    ///
    /// - Throws: `MoviesError.ItemNotFound` if the movie isn't in the table.
    ///
    /// - Returns: A `Movie` record with the movie's details.
    func get(title: String, year: Int) async throws -> Movie {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = GetItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
```

```
                ],
                tableName: self.tableName
            )
            let output = try await client.getItem(input: input)
            guard let item = output.item else {
                throw MoviesError.ItemNotFound
            }

            let movie = try Movie(withItem: item)
            return movie
        } catch {
            print("ERROR: get:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte GetItem a referência da API AWS SDK for Swift.

`ListTables`

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Get a list of the DynamoDB tables available in the specified Region.
    ///
    /// - Returns: An array of strings listing all of the tables available
    ///   in the Region specified when the session was created.
    public func getTableList() async throws -> [String] {
        let input = ListTablesInput(
        )
```

```
        return try await session.listTables(input: input)
    }
```

- Para obter detalhes da API, consulte ListTables a referência da API AWS SDK for Swift.

`PutItem`

O código de exemplo a seguir mostra como usar `PutItem`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Add a movie specified as a `Movie` structure to the Amazon DynamoDB
    /// table.
    ///
    /// - Parameter movie: The `Movie` to add to the table.
    ///
    func add(movie: Movie) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Get a DynamoDB item containing the movie data.
            let item = try await movie.getAsItem()

            // Send the `PutItem` request to Amazon DynamoDB.

            let input = PutItemInput(
                item: item,
                tableName: self.tableName
            )
            _ = try await client.putItem(input: input)
```

```
        } catch {
            print("ERROR: add movie:", dump(error))
            throw error
        }
    }


    ///
    /// Return an array mapping attribute names to Amazon DynamoDB attribute
    /// values, representing the contents of the `Movie` record as a DynamoDB
    /// item.
    ///
    /// - Returns: The movie item as an array of type
    ///   `[Swift.String:DynamoDBClientTypes.AttributeValue]`.
    ///
    func getAsItem() async throws ->
[Swift.String:DynamoDBClientTypes.AttributeValue]  {
        // Build the item record, starting with the year and title, which are
        // always present.

        var item: [Swift.String:DynamoDBClientTypes.AttributeValue] = [
            "year": .n(String(self.year)),
            "title": .s(self.title)
        ]

        // Add the `info` field with the rating and/or plot if they're
        // available.

        var details: [Swift.String:DynamoDBClientTypes.AttributeValue] = [:]
        if (self.info.rating != nil || self.info.plot != nil) {
            if self.info.rating != nil {
                details["rating"] = .n(String(self.info.rating!))
            }
            if self.info.plot != nil {
                details["plot"] = .s(self.info.plot!)
            }
        }
        item["info"] = .m(details)

        return item
    }
```

- Para obter detalhes da API, consulte PutItem a referência da API AWS SDK for Swift.

Consulta

O código de exemplo a seguir mostra como usar `Query`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Get all the movies released in the specified year.
    ///
    /// - Parameter year: The release year of the movies to return.
    ///
    /// - Returns: An array of `Movie` objects describing each matching movie.
    ///
    func getMovies(fromYear year: Int) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = QueryInput(
                expressionAttributeNames: [
                    "#y": "year"
                ],
                expressionAttributeValues: [
                    ":y": .n(String(year))
                ],
                keyConditionExpression: "#y = :y",
                tableName: self.tableName
            )
            // Use "Paginated" to get all the movies.
            // This lets the SDK handle the 'lastEvaluatedKey' property in
 "QueryOutput".

            let pages = client.queryPaginated(input: input)
```

```
            var movieList: [Movie] = []
            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Convert the found movies into `Movie` objects and return an array
                // of them.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList
        } catch {
            print("ERROR: getMovies:", dump(error))
            throw error
        }
    }
```

- Consulte detalhes da API em Query na Referência da API AWS SDK para Swift.

Verificação

O código de exemplo a seguir mostra como usar `Scan`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Return an array of `Movie` objects released in the specified range of
```

```
    /// years.
    ///
    /// - Parameters:
    ///   - firstYear: The first year of movies to return.
    ///   - lastYear: The last year of movies to return.
    ///   - startKey: A starting point to resume processing; always use `nil`.
    ///
    /// - Returns: An array of `Movie` objects describing the matching movies.
    ///
    /// > Note: The `startKey` parameter is used by this function when
    ///   recursively calling itself, and should always be `nil` when calling
    ///   directly.
    ///
    func getMovies(firstYear: Int, lastYear: Int,
                   startKey: [Swift.String: DynamoDBClientTypes.AttributeValue]? =
nil)
        async throws -> [Movie]
    {
        do {
            var movieList: [Movie] = []

            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = ScanInput(
                consistentRead: true,
                exclusiveStartKey: startKey,
                expressionAttributeNames: [
                    "#y": "year" // `year` is a reserved word, so use `#y` instead.
                ],
                expressionAttributeValues: [
                    ":y1": .n(String(firstYear)),
                    ":y2": .n(String(lastYear))
                ],
                filterExpression: "#y BETWEEN :y1 AND :y2",
                tableName: self.tableName
            )

            let pages = client.scanPaginated(input: input)

            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
```

```
                    continue
                }

                // Build an array of `Movie` objects for the returned items.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList

        } catch {
            print("ERROR: getMovies with scan:", dump(error))
            throw error
        }
    }
```

- Consulte detalhes da API em Scan na Referência da API AWS SDK para Swift.

`UpdateItem`

O código de exemplo a seguir mostra como usar `UpdateItem`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSDynamoDB


    /// Update the specified movie with new `rating` and `plot` information.
    ///
    /// - Parameters:
    ///   - title: The title of the movie to update.
    ///   - year: The release year of the movie to update.
```

```
    ///   - rating: The new rating for the movie.
    ///   - plot: The new plot summary string for the movie.
    ///
    /// - Returns: An array of mappings of attribute names to their new
    ///   listing each item actually changed. Items that didn't need to change
    ///   aren't included in this list. `nil` if no changes were made.
    ///
    func update(title: String, year: Int, rating: Double? = nil, plot: String? =
nil) async throws
        -> [Swift.String: DynamoDBClientTypes.AttributeValue]?
    {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Build the update expression and the list of expression attribute
            // values. Include only the information that's changed.

            var expressionParts: [String] = []
            var attrValues: [Swift.String: DynamoDBClientTypes.AttributeValue] = [:]

            if rating != nil {
                expressionParts.append("info.rating=:r")
                attrValues[":r"] = .n(String(rating!))
            }
            if plot != nil {
                expressionParts.append("info.plot=:p")
                attrValues[":p"] = .s(plot!)
            }
            let expression = "set \(expressionParts.joined(separator: ", "))"

            let input = UpdateItemInput(
                // Create substitution tokens for the attribute values, to ensure
                // no conflicts in expression syntax.
                expressionAttributeValues: attrValues,
                // The key identifying the movie to update consists of the release
                // year and title.
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                returnValues: .updatedNew,
                tableName: self.tableName,
```

```
                updateExpression: expression
            )
            let output = try await client.updateItem(input: input)

            guard let attributes: [Swift.String: DynamoDBClientTypes.AttributeValue]
= output.attributes else {
                throw MoviesError.InvalidAttributes
            }
            return attributes
        } catch {
            print("ERROR: update:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte UpdateItem a referência da API AWS SDK for Swift.

Exemplos de código do Amazon EC2 usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon EC2.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Conceitos básicos
- Ações

Conceitos básicos

Olá, Amazon EC2

O exemplo de código a seguir mostra como começar a usar o Amazon EC2.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "hello-ec2",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
```

```
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "hello-ec2",
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to perform a simple
// operation using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation

import AWSEC2

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"
```

```
    static var configuration = CommandConfiguration(
        commandName: "hello-ec2",
        abstract: """
        Demonstrates a simple operation using Amazon EC2.
        """,
        discussion: """
        An example showing how to make a call to Amazon EC2 using the AWS SDK for
Swift.
        """
    )

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
    /// - Returns: An array of strings giving the names of every security
    ///   group the user is a member of.
    func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
        let pages = ec2Client.describeSecurityGroupsPaginated(
            input: DescribeSecurityGroupsInput()
        )

        var groupNames: [String] = []

        do {
            for try await page in pages {
                guard let groups = page.securityGroups else {
                    print("*** Error: No groups returned.")
                    continue
                }

                for group in groups {
                    groupNames.append(group.groupName ?? "<unknown>")
                }
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }

        return groupNames
    }
```

```
    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
 awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let groupNames = await getSecurityGroupNames(ec2Client: ec2Client)

        print("Found \(groupNames.count) security group(s):")

        for group in groupNames {
            print("    \(group)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte DescribeSecurityGroups a referência da API AWS SDK for Swift.

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um par de chaves e um grupo de segurança.

- Selecionar uma imagem de máquina da Amazon (AMI) e um tipo de instância compatível e, em seguida, criar uma instância.
- Interromper e reiniciar a instância.
- Associar um endereço IP elástico à sua instância.
- Conectar-se à sua instância via SSH e, em seguida, limpar os recursos.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "ec2-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
```

```
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "ec2-scenario",
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "AWSSSM", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation
import AWSEC2

// Allow waiters to be used.

import class SmithyWaitersAPI.Waiter
import struct SmithyWaitersAPI.WaiterOptions

import AWSSSM

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
```

```
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "ec2-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon EC2 using the
        AWS SDK for Swift.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ssmConfig = try await SSMClient.SSMClientConfiguration(region:
 awsRegion)
        let ssmClient = SSMClient(config: ssmConfig)

        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
 awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let example = Example(ec2Client: ec2Client, ssmClient: ssmClient)

        await example.run()
    }
}

class Example {
    let ec2Client: EC2Client
    let ssmClient: SSMClient

    // Storage for AWS EC2 properties.

    var keyName: String? = nil
    var securityGroupId: String? = nil
```

```
    var instanceId: String? = nil
    var allocationId: String? = nil
    var associationId: String? = nil

    init(ec2Client: EC2Client, ssmClient: SSMClient) {
        self.ec2Client = ec2Client
        self.ssmClient = ssmClient
    }

    /// The example's main body.
    func run() async {
        //=====================================================================
        // 1. Create an RSA key pair, saving the private key as a `.pem` file.
        //    Create a `defer` block that will delete the private key when the
        //    program exits.
        //=====================================================================

        print("Creating an RSA key pair...")

        keyName = self.tempName(prefix: "ExampleKeyName")
        let keyUrl = await self.createKeyPair(name: keyName!)

        guard let keyUrl else {
            print("*** Failed to create the key pair!")
            return
        }

        print("Created the private key at: \(keyUrl.absoluteString)")

        // Schedule deleting the private key file to occur automatically when
        // the program exits, no matter how it exits.

        defer {
            do {
                try FileManager.default.removeItem(at: keyUrl)
            } catch {
                print("*** Failed to delete the private key at
\(keyUrl.absoluteString)")
            }
        }

        //=====================================================================
        // 2. List the key pairs by calling `DescribeKeyPairs`.
        //=====================================================================
```

```
        print("Describing available key pairs...")
        await self.describeKeyPairs()

        //=====================================================================
        // 3. Create a security group for the default VPC, and add an inbound
        //    rule to allow SSH from the current computer's public IPv4
        //    address.
        //=====================================================================

        print("Creating the security group...")

        let secGroupName = self.tempName(prefix: "ExampleSecurityGroup")
        let ipAddress = self.getMyIPAddress()

        guard let ipAddress else {
            print("*** Unable to get the device's IP address.")
            return
        }

        print("IP address is: \(ipAddress)")

        securityGroupId = await self.createSecurityGroup(
            name: secGroupName,
            description: "An example security group created using the AWS SDK for
Swift"
        )

        if securityGroupId == nil {
            await cleanUp()
            return
        }

        print("Created security group: \(securityGroupId ?? "<unknown>")")

        if !(await self.authorizeSecurityGroupIngress(groupId: securityGroupId!,
ipAddress: ipAddress)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 4. Display security group information for the new security group
        //    using DescribeSecurityGroups.
```

```
        //=====================================================================

        if !(await self.describeSecurityGroups(groupId: securityGroupId!)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 5. Get a list of Amazon Linux 2023 AMIs and pick one (SSM is the
        //    best practice), using path and then filter the list after the
        //    fact to include "al2023" in the Name field
        //    (ssm.GetParametersByPath). Paginate to get all images.
        //=====================================================================

        print("Searching available images for Amazon Linux 2023 images...")

        let options = await self.findAMIsMatchingFilter("al2023")

        //=====================================================================
        // 6. The information in the AMI options isn't great, so make a list
        //    of the image IDs (the "Value" field in the AMI options) and get
        //    more information about them from EC2. Display the Description
        //    field and select one of them (DescribeImages with ImageIds
        //    filter).
        //=====================================================================

        print("Images matching Amazon Linux 2023:")

        var imageIds: [String] = []
        for option in options {
            guard let id = option.value else {
                continue
            }
            imageIds.append(id)
        }

        let images = await self.describeImages(imageIds)

        // This is where you would normally let the user choose which AMI to
        // use. However, for this example, we're just going to use the first
        // one, whatever it is.

        let chosenImage = images[0]
```

```
        //=====================================================================
        // 7. Get a list of instance types that are compatible with the
        //    selected AMI's architecture (such as "x86_64") and are either
        //    small or micro. Select one (DescribeInstanceTypes).
        //=====================================================================

        print("Getting the instance types compatible with the selected image...")

        guard let arch = chosenImage.architecture else {
            print("*** The selected image doesn't have a valid architecture.")
            await cleanUp()
            return
        }

        let imageTypes = await self.getMatchingInstanceTypes(architecture: arch)

        for type in imageTypes {
            guard let instanceType = type.instanceType else {
                continue
            }
            print("    \(instanceType.rawValue)")
        }

        // This example selects the first returned instance type. A real-world
        // application would probably ask the user to select one here.

        let chosenInstanceType = imageTypes[0]

        //=====================================================================
        // 8. Create an instance with the key pair, security group, AMI, and
        //    instance type (RunInstances).
        //=====================================================================

        print("Creating an instance...")

        guard let imageId = chosenImage.imageId else {
            print("*** Cannot start image without a valid image ID.")
            await cleanUp()
            return
        }
        guard let instanceType = chosenInstanceType.instanceType else {
            print("*** Unable to start image without a valid image type.")
            await cleanUp()
            return
```

```
        }

        let instance = await self.runInstance(
            imageId: imageId,
            instanceType: instanceType,
            keyPairName: keyName!,
            securityGroups: [securityGroupId!]
        )

        guard let instance else {
            await cleanUp()
            return
        }

        instanceId = instance.instanceId
        if instanceId == nil {
            print("*** Instance is missing an ID. Canceling.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 9. Wait for the instance to be ready and then display its
        //    information (DescribeInstances).
        //=====================================================================

        print("Waiting a few seconds to let the instance come up...")

        do {
            try await Task.sleep(for: .seconds(20))
        } catch {
            print("*** Error pausing the task.")
        }
        print("Success! Your new instance is ready:")

        //=====================================================================
        // 10. Display SSH connection info for the instance.
        //=====================================================================

        var runningInstance = await self.describeInstance(instanceId: instanceId!)

        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil) {
            print("\nYou can SSH to this instance using the following command:")
```

```
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // 11. Stop the instance and wait for it to stop (StopInstances).
        //=====================================================================

        print("Stopping the instance...")

        if !(await self.stopInstance(instanceId: instanceId!, waitUntilStopped:
 true)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 12. Start the instance and wait for it to start (StartInstances).
        //=====================================================================

        print("Starting the instance again...")

        if !(await self.startInstance(instanceId: instanceId!, waitUntilStarted:
 true)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 13. Display SSH connection info for the instance. Note that it's
        //     changed.
        //=====================================================================

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil) {
            print("\nYou can SSH to this instance using the following command.")
            print("This is probably different from when the instance was running
 before.")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // 14. Allocate an elastic IP and associate it with the instance
```

```
        //     (AllocateAddress and AssociateAddress).
        //=========================================================================

        allocationId = await self.allocateAddress()

        if allocationId == nil {
            await cleanUp()
            return
        }

        associationId = await self.associateAddress(instanceId: instanceId!,
 allocationId: allocationId)

        if associationId == nil {
            await cleanUp()
            return
        }

        //=========================================================================
        // 15. Display SSH connection info for the connection. Note that the
        //     public IP is now the Elastic IP, which stays constant.
        //=========================================================================

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil) {
            print("\nYou can SSH to this instance using the following command.")
            print("This has changed again, and is now the Elastic IP.")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=========================================================================
        // Handle all cleanup tasks
        //=========================================================================

        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items:
    ///
    /// * Elastic IP allocation and association
    /// * Terminate the instance
    /// * Delete the security group
    /// * Delete the key pair
```

```
    func cleanUp() async {
        //=====================================================================
        // 16. Disassociate and delete the Elastic IP (DisassociateAddress and
        //     ReleaseAddress).
        //=====================================================================

        if associationId != nil {
            await self.disassociateAddress(associationId: associationId!)
        }

        if allocationId != nil {
            await self.releaseAddress(allocationId: allocationId!)
        }

        //=====================================================================
        // 17. Terminate the instance and wait for it to terminate
        //     (TerminateInstances).
        //=====================================================================

        if instanceId != nil {
            print("Terminating the instance...")
            _ = await self.terminateInstance(instanceId: instanceId!,
waitUntilTerminated: true)
        }

        //=====================================================================
        // 18. Delete the security group (DeleteSecurityGroup).
        //=====================================================================

        if securityGroupId != nil {
            print("Deleting the security group...")
            _ = await self.deleteSecurityGroup(groupId: securityGroupId!)
        }

        //=====================================================================
        // 19. Delete the key pair (DeleteKeyPair).
        //=====================================================================

        if keyName != nil {
            print("Deleting the key pair...")
            _ = await self.deleteKeyPair(keyPair: keyName!)
        }
    }
```

```
    /// Create a new RSA key pair and save the private key to a randomly-named
    /// file in the temporary directory.
    ///
    /// - Parameter name: The name of the key pair to create.
    ///
    /// - Returns: The URL of the newly created `.pem` file or `nil` if unable
    ///   to create the key pair.
    func createKeyPair(name: String) async -> URL? {
        do {
            let output = try await ec2Client.createKeyPair(
                input: CreateKeyPairInput(
                    keyName: name
                )
            )

            guard let keyMaterial = output.keyMaterial else {
                return nil
            }

            // Build the URL of the temporary private key file.

            let fileURL = URL.temporaryDirectory
                                .appendingPathComponent(name)
                                .appendingPathExtension("pem")

            do {
                try keyMaterial.write(to: fileURL, atomically: true, encoding:
String.Encoding.utf8)
                return fileURL
            } catch {
                print("*** Failed to write the private key.")
                return nil
            }
        } catch {
            print("*** Unable to create the key pair.")
            return nil
        }
    }

    /// Describe the key pairs associated with the user by outputting each key
    /// pair's name and fingerprint.
    func describeKeyPairs() async {
        do {
            let output = try await ec2Client.describeKeyPairs(
```

```
                input: DescribeKeyPairsInput()
            )

            guard let keyPairs = output.keyPairs else {
                print("*** No key pairs list available.")
                return
            }

            for keyPair in keyPairs {
                print(keyPair.keyName ?? "<unknown>", ":", keyPair.keyFingerprint ??
"<unknown>")
            }
        } catch {
            print("*** Error: Unable to obtain a key pair list.")
        }
    }

    /// Delete an EC2 key pair.
    ///
    /// - Parameter keyPair: The name of the key pair to delete.
    ///
    /// - Returns: `true` if the key pair is deleted successfully; otherwise
    ///   `false`.
    func deleteKeyPair(keyPair: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteKeyPair(
                input: DeleteKeyPairInput(
                    keyName: keyPair
                )
            )

            return true
        } catch {
            print("*** Error deleting the key pair: \(error.localizedDescription)")
            return false
        }
    }

    /// Return a list of AMI names that contain the specified string.
    ///
    /// - Parameter filter: A string that must be contained in all returned
    ///   AMI names.
    ///
    /// - Returns: An array of the parameters matching the specified substring.
```

```
    func findAMIsMatchingFilter(_ filter: String) async ->
[SSMClientTypes.Parameter] {
        var parameterList: [SSMClientTypes.Parameter] = []
        var matchingAMIs: [SSMClientTypes.Parameter] = []

        do {
            let pages = ssmClient.getParametersByPathPaginated(
                input: GetParametersByPathInput(
                    path: "/aws/service/ami-amazon-linux-latest"
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return matchingAMIs
                }

                for parameter in parameters {
                    parameterList.append(parameter)
                }
            }

            print("Found \(parameterList.count) images total:")
            for parameter in parameterList {
                guard let name = parameter.name else {
                    continue
                }
                print("    \(name)")

                if name.contains(filter) {
                    matchingAMIs.append(parameter)
                }
            }
        } catch {
            return matchingAMIs
        }

        return matchingAMIs
    }

    /// Return a list of instance types matching the specified architecture
    /// and instance sizes.
    ///
    /// - Parameters:
```

```
    ///   - architecture: The architecture of the instance types to return, as
    ///     a member of `EC2ClientTypes.ArchitectureValues`.
    ///   - sizes: An array of one or more strings identifying sizes of
    ///     instance type to accept.
    ///
    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues =
EC2ClientTypes.ArchitectureValues.x8664,
                          sizes: [String] = ["*.micro", "*.small"]) async
                          -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )

        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )

            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }
```

```
    /// Get the latest information about the specified instance and output it
    /// to the screen, returning the instance details to the caller.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to provide details about.
    ///   - stateFilter: The state to require the instance to be in.
    ///
    /// - Returns: The instance's details as an `EC2ClientTypes.Instance` object.
    func describeInstance(instanceId: String,
                          stateFilter: EC2ClientTypes.InstanceStateName? =
EC2ClientTypes.InstanceStateName.running) async
                          -> EC2ClientTypes.Instance? {
        do {
            let pages = ec2Client.describeInstancesPaginated(
                input: DescribeInstancesInput(
                    instanceIds: [instanceId]
                )
            )

            for try await page in pages {
                guard let reservations = page.reservations else {
                    continue
                }

                for reservation in reservations {
                    guard let instances = reservation.instances else {
                        continue
                    }

                    for instance in instances {
                        guard let state = instance.state else {
                            print("*** Instance is missing its state...")
                            continue
                        }
                        let instanceState = state.name

                        if stateFilter != nil && (instanceState != stateFilter) {
                            continue
                        }

                        let instanceTypeName: String
                        if instance.instanceType == nil {
                            instanceTypeName = "<N/A>"
                        } else {
```

```
                            instanceTypeName = instance.instanceType?.rawValue ??
"<N/A>"
                        }

                        let instanceStateName: String
                        if instanceState == nil {
                            instanceStateName = "<N/A>"
                        } else {
                            instanceStateName = instanceState?.rawValue ?? "<N/A>"
                        }

                        print("""
                        Instance: \(instance.instanceId ?? "<N/A>")
                                • Image ID: \(instance.imageId ?? "<N/A>")
                                • Instance type: \(instanceTypeName)
                                • Key name: \(instance.keyName ?? "<N/A>")
                                • VPC ID: \(instance.vpcId ?? "<N/A>")
                                • Public IP: \(instance.publicIpAddress ?? "N/A")
                                • State: \(instanceStateName)
                        """)

                        return instance
                    }
                }
            }
        } catch {
            print("*** Error retrieving instance information to display:
\(error.localizedDescription)")
            return nil
        }

        return nil
    }

    /// Stop the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to stop.
    ///   - waitUntilStopped: If `true`, execution waits until the instance
    ///     has stopped. Otherwise, execution continues and the instance stops
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully stopped (or is left to
    ///   stop asynchronously). `false` if the instance doesn't stop.
```

```
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async ->
Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance: \(error.localizedDescription)")
            return false
        }
    }

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance
    ///     has started. Otherwise, execution continues and the instance starts
    ///     asynchronously.
```

```
    ///
    /// - Returns: `true` if the image is successfully started (or is left to
    ///   start asynchronously). `false` if the instance doesn't start.
    func startInstance(instanceId: String, waitUntilStarted: Bool = false) async ->
Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.startInstances(
                input: StartInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStarted {
                print("Waiting for the instance to start...")

                let waitOptions = WaiterOptions(maxWaitTime: 60.0)
                let output = try await ec2Client.waitUntilInstanceRunning(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to start the instance: \(error.localizedDescription)")
            return false
        }
    }

    /// Terminate the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to terminate.
    ///   - waitUntilTerminated: Whether or not to wait until the instance is
```

```
   ///     terminated before returning.
   ///
   /// - Returns: `true` if terminated successfully. `false` if not or if an
   ///   error occurs.
   func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
async -> Bool {
       let instanceList = [instanceId]

       do {
           _ = try await ec2Client.terminateInstances(
               input: TerminateInstancesInput(
                   instanceIds: instanceList
               )
           )

           if waitUntilTerminated {
               print("Waiting for the instance to terminate...")

               let waitOptions = WaiterOptions(maxWaitTime: 600.0)
               let output = try await ec2Client.waitUntilInstanceTerminated(
                   options: waitOptions,
                   input: DescribeInstancesInput(
                       instanceIds: instanceList
                   )
               )

               switch output.result {
               case .success:
                   return true
               case .failure:
                   return false
               }
           } else {
               return true
           }
       } catch {
           print("*** Unable to terminate the instance:
\(error.localizedDescription)")
           return false
       }
   }

   /// Return an array of `EC2ClientTypes.Image` objects describing all of
   /// the images in the specified array.
```

```
    ///
    /// - Parameter idList: A list of image ID strings indicating the images
    ///   to return details about.
    ///
    /// - Returns: An array of the images.
    func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
        do {
            let output = try await ec2Client.describeImages(
                input: DescribeImagesInput(
                    imageIds: idList
                )
            )

            guard let images = output.images else {
                print("*** No images found.")
                return []
            }

            for image in images {
                guard let id = image.imageId else {
                    continue
                }
                print("   \(id): \(image.description ?? "<no description>")")
            }

            return images
        } catch {
            print("*** Error getting image descriptions:
\(error.localizedDescription)")
            return []
        }
    }

    /// Create and return a new EC2 instance.
    ///
    /// - Parameters:
    ///   - imageId: The image ID of the AMI to use when creating the instance.
    ///   - instanceType: The type of instance to create.
    ///   - keyPairName: The RSA key pair's name to use to secure the instance.
    ///   - securityGroups: The security group or groups to add the instance
    ///     to.
    ///
    /// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
    func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
```

```
                        keyPairName: String, securityGroups: [String]?) async ->
EC2ClientTypes.Instance? {
        do {
            let output = try await ec2Client.runInstances(
                input: RunInstancesInput(
                    imageId: imageId,
                    instanceType: instanceType,
                    keyName: keyPairName,
                    maxCount: 1,
                    minCount: 1,
                    securityGroupIds: securityGroups
                )
            )

            guard let instances = output.instances else {
                print("*** Unable to create the instance.")
                return nil
            }

            return instances[0]
        } catch {
            print("*** Error creating the instance: \(error.localizedDescription)")
            return nil
        }
    }

    /// Return the device's external IP address.
    ///
    /// - Returns: A string containing the device's IP address.
    func getMyIPAddress() -> String? {
        guard let url = URL(string: "http://checkip.amazonaws.com") else {
            print("Couldn't create the URL")
            return nil
        }

        do {
            print("Getting the IP address...")
            return try String(contentsOf: url, encoding:
String.Encoding.utf8).trim()
        } catch {
            print("*** Unable to get your public IP address.")
            return nil
        }
    }
```

```
    /// Create a new security group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the group to create.
    ///   - groupDescription: A description of the new security group.
    ///
    /// - Returns: The ID string of the new security group.
    func createSecurityGroup(name groupName: String, description groupDescription:
String) async -> String? {
        do {
            let output = try await ec2Client.createSecurityGroup(
                input: CreateSecurityGroupInput(
                    description: groupDescription,
                    groupName: groupName
                )
            )

            return output.groupId
        } catch {
            print("*** Error creating the security group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Authorize ingress of connections for the security group.
    ///
    /// - Parameters:
    ///   - groupId: The group ID of the security group to authorize access for.
    ///   - ipAddress: The IP address of the device to grant access to.
    ///
    /// - Returns: `true` if access is successfully granted; otherwise `false`.
    func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async ->
Bool {
        let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
        let httpPermission = EC2ClientTypes.IpPermission(
            fromPort: 80,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 80
        )

        let sshPermission = EC2ClientTypes.IpPermission(
```

```
            fromPort: 22,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 22
        )

        do {
            _ = try await ec2Client.authorizeSecurityGroupIngress(
                input: AuthorizeSecurityGroupIngressInput(
                    groupId: groupId,
                    ipPermissions: [httpPermission, sshPermission]
                )
            )

            return true
        } catch {
            print("*** Error authorizing ingress for the security group:
\(error.localizedDescription)")
            return false
        }
    }

    func describeSecurityGroups(groupId: String) async -> Bool {
        do {
            let output = try await ec2Client.describeSecurityGroups(
                input: DescribeSecurityGroupsInput(
                    groupIds: [groupId]
                )
            )

            guard let securityGroups = output.securityGroups else {
                print("No security groups found.")
                return true
            }

            for group in securityGroups {
                print("Group \(group.groupId ?? "<unknown>") found with VPC
\(group.vpcId ?? "<unknown>")")
            }
            return true
        } catch {
            print("*** Error getting security group details:
\(error.localizedDescription)")
            return false
```

```
        }
    }

    /// Delete a security group.
    ///
    /// - Parameter groupId: The ID of the security group to delete.
    ///
    /// - Returns: `true` on successful deletion; `false` on error.
    func deleteSecurityGroup(groupId: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteSecurityGroup(
                input: DeleteSecurityGroupInput(
                    groupId: groupId
                )
            )

            return true
        } catch {
            print("*** Error deleting the security group:
\(error.localizedDescription)")
            return false
        }
    }

    /// Allocate an Elastic IP address.
    ///
    /// - Returns: A string containing the ID of the Elastic IP.
    func allocateAddress() async -> String? {
        do {
            let output = try await ec2Client.allocateAddress(
                input: AllocateAddressInput(
                    domain: EC2ClientTypes.DomainType.vpc
                )
            )

            guard let allocationId = output.allocationId else {
                return nil
            }

            return allocationId
        } catch {
            print("*** Unable to allocate the IP address:
\(error.localizedDescription)")
            return nil
```

```
        }
    }

    /// Associate the specified allocated Elastic IP to a given instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )

            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
            )
        } catch {
            print("*** Unable to disassociate the IP address:
\(error.localizedDescription)")
        }
    }
```

```
    /// Release an allocated Elastic IP.
    ///
    /// - Parameter allocationId: The allocation ID of the Elastic IP to
    ///   release.
    func releaseAddress(allocationId: String?) async {
        do {
            _ = try await ec2Client.releaseAddress(
                input: ReleaseAddressInput(
                    allocationId: allocationId
                )
            )
        } catch {
            print("*** Unable to release the IP address:
 \(error.localizedDescription)")
        }
    }

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
```

```
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Ações

`AllocateAddress`

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Allocate an Elastic IP address.
    ///
    /// - Returns: A string containing the ID of the Elastic IP.
    func allocateAddress() async -> String? {
        do {
            let output = try await ec2Client.allocateAddress(
                input: AllocateAddressInput(
                    domain: EC2ClientTypes.DomainType.vpc
                )
            )

            guard let allocationId = output.allocationId else {
                return nil
            }

            return allocationId
        } catch {
            print("*** Unable to allocate the IP address:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte AllocateAddress a referência da API AWS SDK for Swift.

`AssociateAddress`

O código de exemplo a seguir mostra como usar `AssociateAddress`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Associate the specified allocated Elastic IP to a given instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
 String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )

            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte AssociateAddress a referência da API AWS SDK for Swift.

`AuthorizeSecurityGroupIngress`

O código de exemplo a seguir mostra como usar `AuthorizeSecurityGroupIngress`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Authorize ingress of connections for the security group.
    ///
    /// - Parameters:
    ///   - groupId: The group ID of the security group to authorize access for.
    ///   - ipAddress: The IP address of the device to grant access to.
    ///
    /// - Returns: `true` if access is successfully granted; otherwise `false`.
    func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async ->
 Bool {
        let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
        let httpPermission = EC2ClientTypes.IpPermission(
            fromPort: 80,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 80
        )

        let sshPermission = EC2ClientTypes.IpPermission(
            fromPort: 22,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 22
        )

        do {
            _ = try await ec2Client.authorizeSecurityGroupIngress(
                input: AuthorizeSecurityGroupIngressInput(
                    groupId: groupId,
                    ipPermissions: [httpPermission, sshPermission]
```

```
                )
            )

            return true
        } catch {
            print("*** Error authorizing ingress for the security group:
\(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte AuthorizeSecurityGroupIngress a referência da API AWS SDK for Swift.

`CreateKeyPair`

O código de exemplo a seguir mostra como usar `CreateKeyPair`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Create a new RSA key pair and save the private key to a randomly-named
    /// file in the temporary directory.
    ///
    /// - Parameter name: The name of the key pair to create.
    ///
    /// - Returns: The URL of the newly created `.pem` file or `nil` if unable
    ///   to create the key pair.
    func createKeyPair(name: String) async -> URL? {
        do {
            let output = try await ec2Client.createKeyPair(
                input: CreateKeyPairInput(
                    keyName: name
```

```
                )
            )

            guard let keyMaterial = output.keyMaterial else {
                return nil
            }

            // Build the URL of the temporary private key file.

            let fileURL = URL.temporaryDirectory
                                .appendingPathComponent(name)
                                .appendingPathExtension("pem")

            do {
                try keyMaterial.write(to: fileURL, atomically: true, encoding:
String.Encoding.utf8)
                return fileURL
            } catch {
                print("*** Failed to write the private key.")
                return nil
            }
        } catch {
            print("*** Unable to create the key pair.")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte CreateKeyPair a referência da API AWS SDK for Swift.

`CreateSecurityGroup`

O código de exemplo a seguir mostra como usar `CreateSecurityGroup`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Create a new security group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the group to create.
    ///   - groupDescription: A description of the new security group.
    ///
    /// - Returns: The ID string of the new security group.
    func createSecurityGroup(name groupName: String, description groupDescription:
 String) async -> String? {
        do {
            let output = try await ec2Client.createSecurityGroup(
                input: CreateSecurityGroupInput(
                    description: groupDescription,
                    groupName: groupName
                )
            )

            return output.groupId
        } catch {
            print("*** Error creating the security group:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte CreateSecurityGroup a referência da API AWS SDK for Swift.

`DeleteKeyPair`

O código de exemplo a seguir mostra como usar `DeleteKeyPair`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Delete an EC2 key pair.
    ///
    /// - Parameter keyPair: The name of the key pair to delete.
    ///
    /// - Returns: `true` if the key pair is deleted successfully; otherwise
    ///   `false`.
    func deleteKeyPair(keyPair: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteKeyPair(
                input: DeleteKeyPairInput(
                    keyName: keyPair
                )
            )

            return true
        } catch {
            print("*** Error deleting the key pair: \(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte DeleteKeyPair a referência da API AWS SDK for Swift.

`DeleteSecurityGroup`

O código de exemplo a seguir mostra como usar `DeleteSecurityGroup`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Delete a security group.
```

```
    ///
    /// - Parameter groupId: The ID of the security group to delete.
    ///
    /// - Returns: `true` on successful deletion; `false` on error.
    func deleteSecurityGroup(groupId: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteSecurityGroup(
                input: DeleteSecurityGroupInput(
                    groupId: groupId
                )
            )

            return true
        } catch {
            print("*** Error deleting the security group:
\(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte DeleteSecurityGroup a referência da API AWS SDK for Swift.

`DescribeImages`

O código de exemplo a seguir mostra como usar `DescribeImages`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Return an array of `EC2ClientTypes.Image` objects describing all of
    /// the images in the specified array.
    ///
```

```
    /// - Parameter idList: A list of image ID strings indicating the images
    ///   to return details about.
    ///
    /// - Returns: An array of the images.
    func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
        do {
            let output = try await ec2Client.describeImages(
                input: DescribeImagesInput(
                    imageIds: idList
                )
            )

            guard let images = output.images else {
                print("*** No images found.")
                return []
            }

            for image in images {
                guard let id = image.imageId else {
                    continue
                }
                print("   \(id): \(image.description ?? "<no description>")")
            }

            return images
        } catch {
            print("*** Error getting image descriptions:
\(error.localizedDescription)")
            return []
        }
    }
```

- Para obter detalhes da API, consulte DescribeImages a referência da API AWS SDK for Swift.

`DescribeInstanceTypes`

O código de exemplo a seguir mostra como usar `DescribeInstanceTypes`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Return a list of instance types matching the specified architecture
    /// and instance sizes.
    ///
    /// - Parameters:
    ///   - architecture: The architecture of the instance types to return, as
    ///     a member of `EC2ClientTypes.ArchitectureValues`.
    ///   - sizes: An array of one or more strings identifying sizes of
    ///     instance type to accept.
    ///
    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues =
 EC2ClientTypes.ArchitectureValues.x8664,
                          sizes: [String] = ["*.micro", "*.small"]) async
                          -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )

        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )
```

```
            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }
```

- Para obter detalhes da API, consulte DescribeInstanceTypes a referência da API AWS SDK for Swift.

`DescribeKeyPairs`

O código de exemplo a seguir mostra como usar `DescribeKeyPairs`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Describe the key pairs associated with the user by outputting each key
    /// pair's name and fingerprint.
    func describeKeyPairs() async {
        do {
            let output = try await ec2Client.describeKeyPairs(
                input: DescribeKeyPairsInput()
            )
```

```
            guard let keyPairs = output.keyPairs else {
                print("*** No key pairs list available.")
                return
            }

            for keyPair in keyPairs {
                print(keyPair.keyName ?? "<unknown>", ":", keyPair.keyFingerprint ??
 "<unknown>")
            }
        } catch {
            print("*** Error: Unable to obtain a key pair list.")
        }
    }
```

- Para obter detalhes da API, consulte DescribeKeyPairs a referência da API AWS SDK for Swift.

`DescribeSecurityGroups`

O código de exemplo a seguir mostra como usar `DescribeSecurityGroups`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Usar paginação com `describeSecurityGroupsPaginated()`.

```
import AWSEC2

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
```

```
    /// - Returns: An array of strings giving the names of every security
    ///   group the user is a member of.
    func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
        let pages = ec2Client.describeSecurityGroupsPaginated(
            input: DescribeSecurityGroupsInput()
        )

        var groupNames: [String] = []

        do {
            for try await page in pages {
                guard let groups = page.securityGroups else {
                    print("*** Error: No groups returned.")
                    continue
                }

                for group in groups {
                    groupNames.append(group.groupName ?? "<unknown>")
                }
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }

        return groupNames
    }
```

Sem paginação.

```
import AWSEC2

    func describeSecurityGroups(groupId: String) async -> Bool {
        do {
            let output = try await ec2Client.describeSecurityGroups(
                input: DescribeSecurityGroupsInput(
                    groupIds: [groupId]
                )
            )

            guard let securityGroups = output.securityGroups else {
                print("No security groups found.")
                return true
```

```
            }

            for group in securityGroups {
                print("Group \(group.groupId ?? "<unknown>") found with VPC
\(group.vpcId ?? "<unknown>")")
            }
            return true
        } catch {
            print("*** Error getting security group details:
\(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte DescribeSecurityGroups a referência da API AWS SDK for Swift.

`DisassociateAddress`

O código de exemplo a seguir mostra como usar `DisassociateAddress`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
```

```
            )
        } catch {
            print("*** Unable to disassociate the IP address:
\(error.localizedDescription)")
        }
    }
```

- Para obter detalhes da API, consulte DisassociateAddress a referência da API AWS SDK for Swift.

`ReleaseAddress`

O código de exemplo a seguir mostra como usar `ReleaseAddress`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Release an allocated Elastic IP.
    ///
    /// - Parameter allocationId: The allocation ID of the Elastic IP to
    ///   release.
    func releaseAddress(allocationId: String?) async {
        do {
            _ = try await ec2Client.releaseAddress(
                input: ReleaseAddressInput(
                    allocationId: allocationId
                )
            )
        } catch {
            print("*** Unable to release the IP address:
\(error.localizedDescription)")
        }
    }
```

- Para obter detalhes da API, consulte ReleaseAddress a referência da API AWS SDK for Swift.

`RunInstances`

O código de exemplo a seguir mostra como usar `RunInstances`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Create and return a new EC2 instance.
    ///
    /// - Parameters:
    ///   - imageId: The image ID of the AMI to use when creating the instance.
    ///   - instanceType: The type of instance to create.
    ///   - keyPairName: The RSA key pair's name to use to secure the instance.
    ///   - securityGroups: The security group or groups to add the instance
    ///     to.
    ///
    /// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
    func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
                         keyPairName: String, securityGroups: [String]?) async ->
 EC2ClientTypes.Instance? {
        do {
            let output = try await ec2Client.runInstances(
                input: RunInstancesInput(
                    imageId: imageId,
                    instanceType: instanceType,
                    keyName: keyPairName,
                    maxCount: 1,
                    minCount: 1,
                    securityGroupIds: securityGroups
```

```
                )
            )

            guard let instances = output.instances else {
                print("*** Unable to create the instance.")
                return nil
            }

            return instances[0]
        } catch {
            print("*** Error creating the instance: \(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte RunInstances a referência da API AWS SDK for Swift.

StartInstances

O código de exemplo a seguir mostra como usar `StartInstances`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance
    ///     has started. Otherwise, execution continues and the instance starts
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully started (or is left to
```

```
    ///   start asynchronously). `false` if the instance doesn't start.
    func startInstance(instanceId: String, waitUntilStarted: Bool = false) async ->
Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.startInstances(
                input: StartInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStarted {
                print("Waiting for the instance to start...")

                let waitOptions = WaiterOptions(maxWaitTime: 60.0)
                let output = try await ec2Client.waitUntilInstanceRunning(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to start the instance: \(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte StartInstances a referência da API AWS SDK for Swift.

`StopInstances`

O código de exemplo a seguir mostra como usar `StopInstances`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Stop the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to stop.
    ///   - waitUntilStopped: If `true`, execution waits until the instance
    ///     has stopped. Otherwise, execution continues and the instance stops
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully stopped (or is left to
    ///   stop asynchronously). `false` if the instance doesn't stop.
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async ->
 Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )
```

```
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance: \(error.localizedDescription)")
            return false
        }
    }
```

- Para obter detalhes da API, consulte StopInstances a referência da API AWS SDK for Swift.

`TerminateInstances`

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSEC2

    /// Terminate the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to terminate.
    ///   - waitUntilTerminated: Whether or not to wait until the instance is
    ///     terminated before returning.
    ///
    /// - Returns: `true` if terminated successfully. `false` if not or if an
    ///   error occurs.
```

```
   func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
async -> Bool {
       let instanceList = [instanceId]

       do {
           _ = try await ec2Client.terminateInstances(
               input: TerminateInstancesInput(
                   instanceIds: instanceList
               )
           )

           if waitUntilTerminated {
               print("Waiting for the instance to terminate...")

               let waitOptions = WaiterOptions(maxWaitTime: 600.0)
               let output = try await ec2Client.waitUntilInstanceTerminated(
                   options: waitOptions,
                   input: DescribeInstancesInput(
                       instanceIds: instanceList
                   )
               )

               switch output.result {
               case .success:
                   return true
               case .failure:
                   return false
               }
           } else {
               return true
           }
       } catch {
           print("*** Unable to terminate the instance:
\(error.localizedDescription)")
           return false
       }
   }
```

- Para obter detalhes da API, consulte TerminateInstances a referência da API AWS SDK for Swift.

AWS Glue exemplos de uso do SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados. CSV-formatted
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair dados CSV do bucket do S3, transformar os dados e carregar a JSON-formatted saída em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte Tutorial: Introdução ao AWS Glue Studio.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "glue-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "glue-scenario",
            dependencies: [
                .product(name: "AWSGlue", package: "aws-sdk-swift"),
```

```
                .product(name: "AWSS3", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo de código Swift, `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to demonstrate
// creating and using crawlers and jobs using AWS Glue.
//
// 0. Upload the Python job script to Amazon S3 so it can be used when
//    calling `startJobRun()` later.
// 1. Create a crawler, pass it the IAM role and the URL of the public Amazon
//    S3 bucket that contains the source data:
//    s3://crawler-public-us-east-1/flight/2016/csv.
// 2. Start the crawler. This takes time, so after starting it, use a loop
//    that calls `getCrawler()` until the state is "READY".
// 3. Get the database created by the crawler, and the tables in the
//    database. Display them to the user.
// 4. Create a job. Pass it the IAM role and the URL to a Python ETL script
//    previously uploaded to the user's S3 bucket.
// 5. Start a job run, passing the following custom arguments. These are
//    expected by the ETL script, so must exactly match.
//    * `--input_database: <name of the database created by the crawler>`
//    * `--input_table: <name of the table created by the crawler>`
//    * `--output_bucket_url: <URL to the scaffold bucket created for the
//      user>`
// 6. Loop and get the job run until it returns one of the following states:
//    "SUCCEEDED", "STOPPED", "FAILED", or "TIMEOUT".
// 7. Output data is stored in a group of files in the user's S3 bucket.
//    Either direct the user to their location or download a file and display
//    the results inline.
// 8. List the jobs for the user's account.
// 9. Get job run details for a job run.
// 10. Delete the demo job.
// 11. Delete the database and tables created by the example.
// 12. Delete the crawler created by the example.

import ArgumentParser
```

```
import AWSS3
import Foundation
import Smithy

import AWSClientRuntime
import AWSGlue

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS IAM role to use for AWS Glue calls.")
    var role: String

    @Option(help: "The Amazon S3 bucket to use for this example.")
    var bucket: String

    @Option(help: "The Amazon S3 URL of the data to crawl.")
    var s3url: String = "s3://crawler-public-us-east-1/flight/2016/csv"

    @Option(help: "The Python script to run as a job with AWS Glue.")
    var script: String = "./flight_etl_job_script.py"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(help: "A prefix string to use when naming tables.")
    var tablePrefix = "swift-glue-basics-table"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "glue-scenario",
        abstract: """
        Demonstrates various features of AWS Glue.
```

```
        """,
        discussion: """
        An example showing how to use AWS Glue to create, run, and monitor
        crawlers and jobs.
        """
    )

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Upload a file to an Amazon S3 bucket.
    ///
    /// - Parameters:
    ///   - s3Client: The S3 client to use when uploading the file.
    ///   - path: The local path of the source file to upload.
    ///   - toBucket: The name of the S3 bucket into which to upload the file.
    ///   - key: The key (name) to give the file in the S3 bucket.
    ///
    /// - Returns: `true` if the file is uploaded successfully, otherwise `false`.
    func uploadFile(s3Client: S3Client, path: String, toBucket: String, key: String)
 async -> Bool {
        do {
            let fileData: Data = try Data(contentsOf: URL(fileURLWithPath: path))
            let dataStream = ByteStream.data(fileData)
            _ = try await s3Client.putObject(
                input: PutObjectInput(
                    body: dataStream,
                    bucket: toBucket,
                    key: key
                )
            )
```

```
        } catch {
            print("*** An unexpected error occurred uploading the script to the
Amazon S3 bucket \"\(bucket)\".")
            return false
        }

        return true
    }

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
    ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
    ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
    ///   - databaseName: The name of an AWS Glue database to operate on.
    ///
    /// - Returns: `true` if the crawler is created successfully, otherwise `false`.
    func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole: String,
                       s3Path: String, cronSchedule: String, databaseName: String)
async -> Bool {
        let s3Target = GlueClientTypes.S3Target(path: s3url)
        let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])

        do {
            _ = try await glueClient.createCrawler(
                input: CreateCrawlerInput(
                    databaseName: databaseName,
                    description: "Created by the AWS SDK for Swift Scenario Example
for AWS Glue.",
                    name: crawlerName,
                    role: iamRole,
                    schedule: cronSchedule,
                    tablePrefix: tablePrefix,
                    targets: targetList
                )
            )
        } catch _ as AlreadyExistsException {
            print("*** A crawler named \"\(crawlerName)\" already exists.")
            return false
        } catch _ as OperationTimeoutException {
            print("*** The attempt to create the AWS Glue crawler timed out.")
```

```
            return false
        } catch {
            print("*** An unexpected error occurred creating the AWS Glue crawler:
\(error.localizedDescription)")
            return false
        }

        return true
    }

    /// Delete an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
        return true
    }

    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise `false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }
```

```
        return true
    }

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
    func getCrawlerState(glueClient: GlueClient, name: String) async ->
GlueClientTypes.CrawlerState {
        do {
            let output = try await glueClient.getCrawler(
                input: GetCrawlerInput(name: name)
            )

            // If the crawler or its state is `nil`, report that the crawler
            // is stopping. This may not be what you want for your
            // application but it works for this one!

            guard let crawler = output.crawler else {
                return GlueClientTypes.CrawlerState.stopping
            }
            guard let state = crawler.state else {
                return GlueClientTypes.CrawlerState.stopping
            }
            return state
        } catch {
            return GlueClientTypes.CrawlerState.stopping
        }
    }

    /// Wait until the specified crawler is ready to run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to wait for.
    ///
    /// - Returns: `true` if the crawler is ready, `false` if the client is
    ///   stopping (and will therefore never be ready).
    func waitUntilCrawlerReady(glueClient: GlueClient, name: String) async -> Bool {
```

```
        while true {
            let state = await getCrawlerState(glueClient: glueClient, name: name)

            if state == .ready {
                return true
            } else if state == .stopping {
                return false
            }

            // Wait four seconds before trying again.

            do {
                try await Task.sleep(for: .seconds(4))
            } catch {
                print("*** Error pausing the task.")
            }
        }
    }

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
                    description: "Created by the AWS SDK for Swift Glue basic
scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
```

```
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }

    /// Return a list of the AWS Glue jobs listed on the user's account.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - maxJobs: The maximum number of jobs to return (default: 100).
    ///
    /// - Returns: An array of strings listing the names of all available AWS
    ///   Glue jobs.
    func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
        var jobList: [String] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.listJobs(
                    input: ListJobsInput(
                        maxResults: maxJobs,
                        nextToken: nextToken
                    )
                )

                guard let jobs = output.jobNames else {
                    return jobList
                }

                jobList = jobList + jobs
                nextToken = output.nextToken
            } catch {
                return jobList
            }
        } while (nextToken != nil)

        return jobList
```

```
    }

    /// Delete an AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to delete.
    ///
    /// - Returns: `true` if the job is successfully deleted, otherwise `false`.
    func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
        do {
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }

    /// Create an AWS Glue database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name to give the new database.
    ///   - location: The URL of the source data to use with AWS Glue.
    ///
    /// - Returns: `true` if the database is created successfully, otherwise
`false`.
    func createDatabase(glueClient: GlueClient, name databaseName: String, location:
String) async -> Bool {
        let databaseInput = GlueClientTypes.DatabaseInput(
            description: "Created by the AWS SDK for Swift Glue basic scenario
example.",
            locationUri: location,
            name: databaseName
        )

        do {
            _ = try await glueClient.createDatabase(
                input: CreateDatabaseInput(
                    databaseInput: databaseInput
                )
            )
```

```
        } catch {
            return false
        }

        return true
    }

    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
    ///   specified database, or `nil` if an error occurs or the database
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async ->
[GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?
```

```
        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }

    /// Delete the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient: glueClient,
databaseName: databaseName)
            for table in tableList {
```

```
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables. If there's only one table, use
            // `deleteTable()`, otherwise, use `batchDeleteTable()`. You can
            // use `batchDeleteTable()` for a single table, but this
            // demonstrates the use of `deleteTable()`.

            if tableNames.count == 1 {
                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
            } else {
                do {
                    print("    Deleting tables...")
                    _ = try await glueClient.batchDeleteTable(
                        input: BatchDeleteTableInput(
                            databaseName: databaseName,
                            tablesToDelete: tableNames
                        )
                    )
                } catch {
                    print("*** Unable to delete the tables.")
                }
            }
        }

        // Delete the database itself.

        do {
            print("    Deleting the database itself...")
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
```

```
            )
        } catch {
            print("*** Unable to delete the database.")
            return false
        }
        return true
    }

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to run.
    ///   - databaseName: The name of the AWS Glue database to run the job against.
    ///   - tableName: The name of the table in the database to run the job against.
    ///   - outputURL: The AWS S3 URI of the bucket location into which to
    ///     write the resulting output.
    ///
    /// - Returns: `true` if the job run is started successfully, otherwise `false`.
    func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
String,
                     tableName: String, outputURL: String) async -> String? {
        do {
            let output = try await glueClient.startJobRun(
                input: StartJobRunInput(
                    arguments: [
                        "--input_database": databaseName,
                        "--input_table": tableName,
                        "--output_bucket_url": outputURL
                    ],
                    jobName: jobName,
                    numberOfWorkers: 10,
                    workerType: .g1x
                )
            )

            guard let id = output.jobRunId else {
                return nil
            }

            return id
        } catch {
            return nil
        }
```

```
    }

    /// Return a list of the job runs for the specified job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job for which to return its job runs.
    ///   - maxResults: The maximum number of job runs to return (default:
    ///     1000).
    ///
    /// - Returns: An array of `GlueClientTypes.JobRun` objects describing
    ///   each job run.
    func getJobRuns(glueClient: GlueClient, name jobName: String, maxResults: Int? =
 nil) async -> [GlueClientTypes.JobRun] {
        do {
            let output = try await glueClient.getJobRuns(
                input: GetJobRunsInput(
                    jobName: jobName,
                    maxResults: maxResults
                )
            )

            guard let jobRuns = output.jobRuns else {
                print("*** No job runs found.")
                return []
            }

            return jobRuns
        } catch is EntityNotFoundException {
            print("*** The specified job name, \(jobName), doesn't exist.")
            return []
        } catch {
            print("*** Unexpected error getting job runs:")
            dump(error)
            return []
        }
    }

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
```

```
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String) async -
> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // A name to give the Python script upon upload to the Amazon S3
        // bucket.
        let scriptName = "jobscript.py"

        // Schedule string in `cron` format, as described here:
        // https://docs.aws.amazon.com/glue/latest/dg/monitor-data-warehouse-
schedule.html
        let cron = "cron(15 12 * * ? *)"

        let glueConfig = try await GlueClient.GlueClientConfiguration(region:
 awsRegion)
        let glueClient = GlueClient(config: glueConfig)

        let s3Config = try await S3Client.S3ClientConfiguration(region: awsRegion)
        let s3Client = S3Client(config: s3Config)

        // Create random names for things that need them.

        let crawlerName = tempName(prefix: "swift-glue-basics-crawler")
        let databaseName = tempName(prefix: "swift-glue-basics-db")

        // Create a name for the AWS Glue job.
```

```
        let jobName = tempName(prefix: "scenario-job")

        // The URL of the Python script on S3.

        let scriptURL = "s3://\(bucket)/\(scriptName)"

        print("Welcome to the AWS SDK for Swift basic scenario for AWS Glue!")

        //=====================================================================
        // 0. Upload the Python script to the target bucket so it's available
        //    for use by the Amazon Glue service.
        //=====================================================================

        print("Uploading the Python script: \(script) as key \(scriptName)")
        print("Destination bucket: \(bucket)")
        if !(await uploadFile(s3Client: s3Client, path: script, toBucket: bucket,
key: scriptName)) {
            return
        }

        //=====================================================================
        // 1. Create the database and crawler using the randomized names
        //    generated previously.
        //=====================================================================

        print("Creating database \"\(databaseName)\"...")
        if !(await createDatabase(glueClient: glueClient, name: databaseName,
location: s3url)) {
            print("*** Unable to create the database.")
            return
        }

        print("Creating crawler \"\(crawlerName)\"...")
        if !(await createCrawler(glueClient: glueClient, crawlerName: crawlerName,
                                 iamRole: role, s3Path: s3url, cronSchedule: cron,
                                 databaseName: databaseName)) {
            return
        }

        //=====================================================================
        // 2. Start the crawler, then wait for it to be ready.
        //=====================================================================

        print("Starting the crawler and waiting until it's ready...")
```

```
        if !(await startCrawler(glueClient: glueClient, name: crawlerName)) {
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        if !(await waitUntilCrawlerReady(glueClient: glueClient, name: crawlerName))
{
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
        }

        //=====================================================================
        // 3. Get the database and table created by the crawler.
        //=====================================================================

        print("Getting the crawler's database...")
        let database = await getDatabase(glueClient: glueClient, name: databaseName)

        guard let database else {
            print("*** Unable to get the database.")
            return
        }
        print("Database URI: \(database.locationUri ?? "<unknown>")")

        let tableList = await getTablesInDatabase(glueClient: glueClient,
databaseName: databaseName)

        print("Found \(tableList.count) table(s):")
        for table in tableList {
            print("  \(table.name ?? "<unnamed>")")
        }

        if tableList.count != 1 {
            print("*** Incorrect number of tables found. There should only be one.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        guard let tableName = tableList[0].name else {
            print("*** Table is unnamed.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
```

```
            return
        }

        //=====================================================================
        // 4. Create a job.
        //=====================================================================

        print("Creating a job...")
        if !(await createJob(glueClient: glueClient, name: jobName, role: role,
                             scriptLocation: scriptURL)) {
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //=====================================================================
        // 5. Start a job run.
        //=====================================================================

        print("Starting the job...")

        // Construct the Amazon S3 URL for the job run's output. This is in
        // the bucket specified on the command line, with a folder name that's
        // unique for this job run.

        let timeStamp = Date().timeIntervalSince1970
        let jobPath = "\(jobName)-\(Int(timeStamp))"
        let outputURL = "s3://\(bucket)/\(jobPath)"

        // Start the job run.

        let jobRunID = await startJobRun(glueClient: glueClient, name: jobName,
                                         databaseName: databaseName,
                                         tableName: tableName,
                                         outputURL: outputURL)

        guard let jobRunID else {
            print("*** Job run ID is invalid.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
```

```
        }

        //=====================================================================
        // 6. Wait for the job run to indicate that the run is complete.
        //=====================================================================

        print("Waiting for job run to end...")

        var jobRunFinished = false
        var jobRunState: GlueClientTypes.JobRunState

        repeat {
            let jobRun = await getJobRun(glueClient: glueClient, name: jobName, id:
jobRunID)
            guard let jobRun else {
                print("*** Unable to get the job run.")
                _ = await deleteJob(glueClient: glueClient, name: jobName)
                _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
                _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
                return
            }
            jobRunState = jobRun.jobRunState ?? .failed

            //=====================================================================
            // 7. Output where to find the data if the job run was successful.
            //    If the job run failed for any reason, output an appropriate
            //    error message.
            //=====================================================================

            switch jobRunState {
                case .succeeded:
                    print("Job run succeeded. JSON files are in the Amazon S3
path:")
                    print("    \(outputURL)")
                    jobRunFinished = true
                case .stopped:
                    jobRunFinished = true
                case .error:
                    print("*** Error: Job run ended in an error.
\(jobRun.errorMessage ?? "")")
                    jobRunFinished = true
                case .failed:
                    print("*** Error: Job run failed. \(jobRun.errorMessage ?? "")")
```

```
                    jobRunFinished = true
                case .timeout:
                    print("*** Warning: Job run timed out.")
                    jobRunFinished = true
                default:
                    do {
                        try await Task.sleep(for: .milliseconds(250))
                    } catch {
                        print("*** Error pausing the task.")
                    }
            }
        } while jobRunFinished != true

        //=====================================================================
        // 7.5. List the job runs for this job, showing each job run's ID and
        // its execution time.
        //=====================================================================

        print("Getting all job runs for the job \(jobName):")
        let jobRuns = await getJobRuns(glueClient: glueClient, name: jobName)

        if jobRuns.count == 0 {
            print("    <no job runs found>")
        } else {
            print("Found \(jobRuns.count) job runs... listing execution times:")
            for jobRun in jobRuns {
                print("    \(jobRun.id ?? "<unnamed>"): \(jobRun.executionTime)
seconds")
            }
        }

        //=====================================================================
        // 8. List the jobs for the user's account.
        //=====================================================================

        print("\nThe account has the following jobs:")
        let jobs = await listJobs(glueClient: glueClient)

        if jobs.count == 0 {
            print("    <no jobs found>")
        } else {
            for job in jobs {
                print("    \(job)")
            }
```

```
        }

        //=====================================================================
        // 9. Get the job run details for a job run.
        //=====================================================================

        print("Information about the job run:")
        let jobRun = await getJobRun(glueClient: glueClient, name: jobName, id:
jobRunID)

        guard let jobRun else {
            print("*** Unable to retrieve the job run.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        let startDate = jobRun.startedOn ?? Date(timeIntervalSince1970: 0)
        let endDate = jobRun.completedOn ?? Date(timeIntervalSince1970: 0)
        let dateFormatter: DateFormatter = DateFormatter()
        dateFormatter.dateStyle = .long
        dateFormatter.timeStyle = .long

        print("    Started at: \(dateFormatter.string(from: startDate))")
        print("  Completed at: \(dateFormatter.string(from: endDate))")

        //=====================================================================
        // 10. Delete the job.
        //=====================================================================

        print("\nDeleting the job...")
        _ = await deleteJob(glueClient: glueClient, name: jobName)

        //=====================================================================
        // 11. Delete the database and tables created by this example.
        //=====================================================================

        print("Deleting the database...")
        _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)

        //=====================================================================
```

```
        // 12. Delete the crawler.
        //=====================================================================

        print("Deleting the crawler...")
        if !(await deleteCrawler(glueClient: glueClient, name: crawlerName)) {
            return
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun

- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Ações

`CreateCrawler`

O código de exemplo a seguir mostra como usar `CreateCrawler`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
    ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
    ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
    ///   - databaseName: The name of an AWS Glue database to operate on.
    ///
    /// - Returns: `true` if the crawler is created successfully, otherwise `false`.
    func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole: String,
                       s3Path: String, cronSchedule: String, databaseName: String)
 async -> Bool {
        let s3Target = GlueClientTypes.S3Target(path: s3url)
        let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])
```

```
        do {
            _ = try await glueClient.createCrawler(
                input: CreateCrawlerInput(
                    databaseName: databaseName,
                    description: "Created by the AWS SDK for Swift Scenario Example
for AWS Glue.",
                    name: crawlerName,
                    role: iamRole,
                    schedule: cronSchedule,
                    tablePrefix: tablePrefix,
                    targets: targetList
                )
            )
        } catch _ as AlreadyExistsException {
            print("*** A crawler named \"\(crawlerName)\" already exists.")
            return false
        } catch _ as OperationTimeoutException {
            print("*** The attempt to create the AWS Glue crawler timed out.")
            return false
        } catch {
            print("*** An unexpected error occurred creating the AWS Glue crawler:
\(error.localizedDescription)")
            return false
        }

        return true
    }
```

- Para obter detalhes da API, consulte CreateCrawler a referência da API AWS SDK for Swift.

`CreateJob`

O código de exemplo a seguir mostra como usar `CreateJob`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
                    description: "Created by the AWS SDK for Swift Glue basic
 scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }
```

- Para obter detalhes da API, consulte CreateJob a referência da API AWS SDK for Swift.

`DeleteCrawler`

O código de exemplo a seguir mostra como usar `DeleteCrawler`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Delete an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
        return true
    }
```

- Para obter detalhes da API, consulte DeleteCrawler a referência da API AWS SDK for Swift.

`DeleteDatabase`

O código de exemplo a seguir mostra como usar `DeleteDatabase`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Delete the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient: glueClient,
 databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables. If there's only one table, use
            // `deleteTable()`, otherwise, use `batchDeleteTable()`. You can
            // use `batchDeleteTable()` for a single table, but this
            // demonstrates the use of `deleteTable()`.
```

```
            if tableNames.count == 1 {
                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
            } else {
                do {
                    print("    Deleting tables...")
                    _ = try await glueClient.batchDeleteTable(
                        input: BatchDeleteTableInput(
                            databaseName: databaseName,
                            tablesToDelete: tableNames
                        )
                    )
                } catch {
                    print("*** Unable to delete the tables.")
                }
            }
        }

        // Delete the database itself.

        do {
            print("    Deleting the database itself...")
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
        } catch {
            print("*** Unable to delete the database.")
            return false
        }
        return true
    }
```

- Para obter detalhes da API, consulte DeleteDatabase a referência da API AWS SDK for Swift.

DeleteJob

O código de exemplo a seguir mostra como usar `DeleteJob`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Delete an AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to delete.
    ///
    /// - Returns: `true` if the job is successfully deleted, otherwise `false`.
    func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
        do {
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }
```

- Para obter detalhes da API, consulte DeleteJob a referência da API AWS SDK for Swift.

DeleteTable

O código de exemplo a seguir mostra como usar `DeleteTable`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
```

- Para obter detalhes da API, consulte DeleteTable a referência da API AWS SDK for Swift.

`GetCrawler`

O código de exemplo a seguir mostra como usar `GetCrawler`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
```

```
import AWSGlue

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
    func getCrawlerState(glueClient: GlueClient, name: String) async ->
 GlueClientTypes.CrawlerState {
        do {
            let output = try await glueClient.getCrawler(
                input: GetCrawlerInput(name: name)
            )

            // If the crawler or its state is `nil`, report that the crawler
            // is stopping. This may not be what you want for your
            // application but it works for this one!

            guard let crawler = output.crawler else {
                return GlueClientTypes.CrawlerState.stopping
            }
            guard let state = crawler.state else {
                return GlueClientTypes.CrawlerState.stopping
            }
            return state
        } catch {
            return GlueClientTypes.CrawlerState.stopping
        }
    }
```

- Para obter detalhes da API, consulte GetCrawler a referência da API AWS SDK for Swift.

GetDatabase

O código de exemplo a seguir mostra como usar `GetDatabase`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
    ///   specified database, or `nil` if an error occurs or the database
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
 GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }
```

- Para obter detalhes da API, consulte GetDatabase a referência da API AWS SDK for Swift.

`GetJobRun`

O código de exemplo a seguir mostra como usar `GetJobRun`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String) async -
> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
    }
```

- Para obter detalhes da API, consulte GetJobRun a referência da API AWS SDK for Swift.

`GetJobRuns`

O código de exemplo a seguir mostra como usar `GetJobRuns`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Return a list of the job runs for the specified job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job for which to return its job runs.
    ///   - maxResults: The maximum number of job runs to return (default:
    ///     1000).
    ///
    /// - Returns: An array of `GlueClientTypes.JobRun` objects describing
    ///   each job run.
    func getJobRuns(glueClient: GlueClient, name jobName: String, maxResults: Int? =
 nil) async -> [GlueClientTypes.JobRun] {
        do {
            let output = try await glueClient.getJobRuns(
                input: GetJobRunsInput(
                    jobName: jobName,
                    maxResults: maxResults
                )
            )

            guard let jobRuns = output.jobRuns else {
                print("*** No job runs found.")
                return []
            }

            return jobRuns
        } catch is EntityNotFoundException {
            print("*** The specified job name, \(jobName), doesn't exist.")
```

```
            return []
        } catch {
            print("*** Unexpected error getting job runs:")
            dump(error)
            return []
        }
    }
```

- Para obter detalhes da API, consulte GetJobRuns a referência da API AWS SDK for Swift.

GetTables

O código de exemplo a seguir mostra como usar `GetTables`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async ->
 [GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?
```

```
        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }
```

- Para obter detalhes da API, consulte GetTables a referência da API AWS SDK for Swift.

`ListJobs`

O código de exemplo a seguir mostra como usar `ListJobs`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue
```

```
/// Return a list of the AWS Glue jobs listed on the user's account.
///
/// - Parameters:
///   - glueClient: The AWS Glue client to use.
///   - maxJobs: The maximum number of jobs to return (default: 100).
///
/// - Returns: An array of strings listing the names of all available AWS
///   Glue jobs.
func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
    var jobList: [String] = []
    var nextToken: String?

    repeat {
        do {
            let output = try await glueClient.listJobs(
                input: ListJobsInput(
                    maxResults: maxJobs,
                    nextToken: nextToken
                )
            )

            guard let jobs = output.jobNames else {
                return jobList
            }

            jobList = jobList + jobs
            nextToken = output.nextToken
        } catch {
            return jobList
        }
    } while (nextToken != nil)

    return jobList
}
```

- Para obter detalhes da API, consulte ListJobs a referência da API AWS SDK for Swift.

`StartCrawler`

O código de exemplo a seguir mostra como usar `StartCrawler`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise `false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }
```

- Para obter detalhes da API, consulte StartCrawler a referência da API AWS SDK for Swift.

`StartJobRun`

O código de exemplo a seguir mostra como usar `StartJobRun`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSGlue

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to run.
    ///   - databaseName: The name of the AWS Glue database to run the job against.
    ///   - tableName: The name of the table in the database to run the job against.
    ///   - outputURL: The AWS S3 URI of the bucket location into which to
    ///     write the resulting output.
    ///
    /// - Returns: `true` if the job run is started successfully, otherwise `false`.
    func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
 String,
                     tableName: String, outputURL: String) async -> String? {
        do {
            let output = try await glueClient.startJobRun(
                input: StartJobRunInput(
                    arguments: [
                        "--input_database": databaseName,
                        "--input_table": tableName,
                        "--output_bucket_url": outputURL
                    ],
                    jobName: jobName,
                    numberOfWorkers: 10,
                    workerType: .g1x
                )
            )

            guard let id = output.jobRunId else {
                return nil
            }
```

```
            return id
        } catch {
            return nil
        }
    }
```

- Para obter detalhes da API, consulte StartJobRun a referência da API AWS SDK for Swift.

Exemplos do IAM usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com IAM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AttachRolePolicy`

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
```

```
import AWSS3


    public func attachRolePolicy(role: String, policyArn: String) async throws {
        let input = AttachRolePolicyInput(
            policyArn: policyArn,
            roleName: role
        )
        do {
            _ = try await client.attachRolePolicy(input: input)
        } catch {
            print("ERROR: Attaching a role policy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte AttachRolePolicy a referência da API AWS SDK for Swift.

`CreateAccessKey`

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createAccessKey(userName: String) async throws ->
 IAMClientTypes.AccessKey {
        let input = CreateAccessKeyInput(
            userName: userName
        )
```

```
        do {
            let output = try await iamClient.createAccessKey(input: input)
            guard let accessKey = output.accessKey else {
                throw ServiceHandlerError.keyError
            }
            return accessKey
        } catch {
            print("ERROR: createAccessKey:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateAccessKey a referência da API AWS SDK for Swift.

`CreatePolicy`

O código de exemplo a seguir mostra como usar `CreatePolicy`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createPolicy(name: String, policyDocument: String) async throws ->
 IAMClientTypes.Policy {
        let input = CreatePolicyInput(
            policyDocument: policyDocument,
            policyName: name
        )
        do {
            let output = try await iamClient.createPolicy(input: input)
```

```
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            print("ERROR: createPolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreatePolicy a referência da API AWS SDK for Swift.

`CreateRole`

O código de exemplo a seguir mostra como usar `CreateRole`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createRole(name: String, policyDocument: String) async throws ->
 String {
        let input = CreateRoleInput(
            assumeRolePolicyDocument: policyDocument,
            roleName: name
        )
        do {
            let output = try await client.createRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            guard let id = role.roleId else {
```

```
                throw ServiceHandlerError.noSuchRole
            }
            return id
        } catch {
            print("ERROR: createRole:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateRole a referência da API AWS SDK for Swift.

`CreateServiceLinkedRole`

O código de exemplo a seguir mostra como usar `CreateServiceLinkedRole`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createServiceLinkedRole(service: String, suffix: String? = nil,
 description: String?)
                    async throws -> IAMClientTypes.Role {
        let input = CreateServiceLinkedRoleInput(
            awsServiceName: service,
            customSuffix: suffix,
            description: description
        )
        do {
            let output = try await client.createServiceLinkedRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
```

```
        } catch {
            print("ERROR: createServiceLinkedRole:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateServiceLinkedRole a referência da API AWS SDK for Swift.

`CreateUser`

O código de exemplo a seguir mostra como usar `CreateUser`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func createUser(name: String) async throws -> String {
        let input = CreateUserInput(
            userName: name
        )
        do {
            let output = try await client.createUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            guard let id = user.userId else {
                throw ServiceHandlerError.noSuchUser
            }
            return id
        } catch {
            print("ERROR: createUser:", dump(error))
```

```
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateUser a referência da API AWS SDK for Swift.

`DeleteAccessKey`

O código de exemplo a seguir mostra como usar `DeleteAccessKey`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func deleteAccessKey(user: IAMClientTypes.User? = nil,
                                key: IAMClientTypes.AccessKey) async throws
    {
        let userName: String?

        if user != nil {
            userName = user!.userName
        } else {
            userName = nil
        }

        let input = DeleteAccessKeyInput(
            accessKeyId: key.accessKeyId,
            userName: userName
        )
        do {
            _ = try await iamClient.deleteAccessKey(input: input)
        } catch {
            print("ERROR: deleteAccessKey:", dump(error))
```

```
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteAccessKey a referência da API AWS SDK for Swift.

`DeletePolicy`

O código de exemplo a seguir mostra como usar `DeletePolicy`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func deletePolicy(policy: IAMClientTypes.Policy) async throws {
        let input = DeletePolicyInput(
            policyArn: policy.arn
        )
        do {
            _ = try await iamClient.deletePolicy(input: input)
        } catch {
            print("ERROR: deletePolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeletePolicy a referência da API AWS SDK for Swift.

`DeleteRole`

O código de exemplo a seguir mostra como usar `DeleteRole`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func deleteRole(role: IAMClientTypes.Role) async throws {
        let input = DeleteRoleInput(
            roleName: role.roleName
        )
        do {
            _ = try await iamClient.deleteRole(input: input)
        } catch {
            print("ERROR: deleteRole:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteRole a referência da API AWS SDK for Swift.

`DeleteUser`

O código de exemplo a seguir mostra como usar `DeleteUser`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func deleteUser(user: IAMClientTypes.User) async throws {
        let input = DeleteUserInput(
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUser(input: input)
        } catch {
            print("ERROR: deleteUser:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteUser a referência da API AWS SDK for Swift.

`DeleteUserPolicy`

O código de exemplo a seguir mostra como usar `DeleteUserPolicy`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    func deleteUserPolicy(user: IAMClientTypes.User, policyName: String) async
 throws {
        let input = DeleteUserPolicyInput(
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUserPolicy(input: input)
        } catch {
            print("ERROR: deleteUserPolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteUserPolicy a referência da API AWS SDK for Swift.

`DetachRolePolicy`

O código de exemplo a seguir mostra como usar `DetachRolePolicy`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func detachRolePolicy(policy: IAMClientTypes.Policy, role:
 IAMClientTypes.Role) async throws {
```

```
        let input = DetachRolePolicyInput(
            policyArn: policy.arn,
            roleName: role.roleName
        )

        do {
            _ = try await iamClient.detachRolePolicy(input: input)
        } catch {
            print("ERROR: detachRolePolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DetachRolePolicy a referência da API AWS SDK for Swift.

`GetPolicy`

O código de exemplo a seguir mostra como usar `GetPolicy`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func getPolicy(arn: String) async throws -> IAMClientTypes.Policy {
        let input = GetPolicyInput(
            policyArn: arn
        )
        do {
            let output = try await client.getPolicy(input: input)
            guard let policy = output.policy else {
```

```
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            print("ERROR: getPolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte GetPolicy a referência da API AWS SDK for Swift.

`GetRole`

O código de exemplo a seguir mostra como usar `GetRole`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func getRole(name: String) async throws -> IAMClientTypes.Role {
        let input = GetRoleInput(
            roleName: name
        )
        do {
            let output = try await client.getRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            print("ERROR: getRole:", dump(error))
            throw error
```

```
        }
    }
```

- Para obter detalhes da API, consulte GetRole a referência da API AWS SDK for Swift.

GetUser

O código de exemplo a seguir mostra como usar `GetUser`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func getUser(name: String? = nil) async throws -> IAMClientTypes.User {
        let input = GetUserInput(
            userName: name
        )
        do {
            let output = try await iamClient.getUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            return user
        } catch {
            print("ERROR: getUser:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte GetUser a referência da API AWS SDK for Swift.

`ListAttachedRolePolicies`

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3



    /// Returns a list of AWS Identity and Access Management (IAM) policies
    /// that are attached to the role.
    ///
    /// - Parameter role: The IAM role to return the policy list for.
    ///
    /// - Returns: An array of `IAMClientTypes.AttachedPolicy` objects
    ///   describing each managed policy that's attached to the role.
    public func listAttachedRolePolicies(role: String) async throws ->
 [IAMClientTypes.AttachedPolicy] {
        var policyList: [IAMClientTypes.AttachedPolicy] = []

        // Use "Paginated" to get all the attached role polices.
        // This lets the SDK handle the 'isTruncated' in
 "ListAttachedRolePoliciesOutput".
        let input = ListAttachedRolePoliciesInput(
            roleName: role
        )
        let output = client.listAttachedRolePoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let attachedPolicies = page.attachedPolicies else {
                    print("Error: no attached policies returned.")
                    continue
                }
                for attachedPolicy in attachedPolicies {
```

```
                    policyList.append(attachedPolicy)
                }
            }
        } catch {
            print("ERROR: listAttachedRolePolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- Para obter detalhes da API, consulte ListAttachedRolePolicies a referência da API AWS SDK for Swift.

`ListGroups`

O código de exemplo a seguir mostra como usar `ListGroups`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listGroups() async throws -> [String] {
        var groupList: [String] = []

        // Use "Paginated" to get all the groups.
        // This lets the SDK handle the 'isTruncated' property in
 "ListGroupsOutput".
        let input = ListGroupsInput()

        let pages = client.listGroupsPaginated(input: input)
        do {
```

```
            for try await page in pages {
                guard let groups = page.groups else {
                    print("Error: no groups returned.")
                    continue
                }

                for group in groups {
                    if let name = group.groupName {
                        groupList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listGroups:", dump(error))
            throw error
        }
        return groupList
    }
```

- Para obter detalhes da API, consulte ListGroups a referência da API AWS SDK for Swift.

`ListPolicies`

O código de exemplo a seguir mostra como usar `ListPolicies`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listPolicies() async throws -> [MyPolicyRecord] {
        var policyList: [MyPolicyRecord] = []

        // Use "Paginated" to get all the policies.
```

```
        // This lets the SDK handle the 'isTruncated' in "ListPoliciesOutput".
        let input = ListPoliciesInput()
        let output = client.listPoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let policies = page.policies else {
                    print("Error: no policies returned.")
                    continue
                }

                for policy in policies {
                    guard let name = policy.policyName,
                          let id = policy.policyId,
                          let arn = policy.arn
                    else {
                        throw ServiceHandlerError.noSuchPolicy
                    }
                    policyList.append(MyPolicyRecord(name: name, id: id, arn: arn))
                }
            }
        } catch {
            print("ERROR: listPolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- Para obter detalhes da API, consulte ListPolicies a referência da API AWS SDK for Swift.

`ListRolePolicies`

O código de exemplo a seguir mostra como usar `ListRolePolicies`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listRolePolicies(role: String) async throws -> [String] {
        var policyList: [String] = []

        // Use "Paginated" to get all the role policies.
        // This lets the SDK handle the 'isTruncated' in "ListRolePoliciesOutput".
        let input = ListRolePoliciesInput(
            roleName: role
        )
        let pages = client.listRolePoliciesPaginated(input: input)

        do {
            for try await page in pages {
                guard let policies = page.policyNames else {
                    print("Error: no role policies returned.")
                    continue
                }

                for policy in policies {
                    policyList.append(policy)
                }
            }
        } catch {
            print("ERROR: listRolePolicies:", dump(error))
            throw error
        }
        return policyList
    }
```

- Para obter detalhes da API, consulte ListRolePolicies a referência da API AWS SDK for Swift.

`ListRoles`

O código de exemplo a seguir mostra como usar `ListRoles`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listRoles() async throws -> [String] {
        var roleList: [String] = []

        // Use "Paginated" to get all the roles.
        // This lets the SDK handle the 'isTruncated' in "ListRolesOutput".
        let input = ListRolesInput()
        let pages = client.listRolesPaginated(input: input)

        do {
            for try await page in pages {
                guard let roles = page.roles else {
                    print("Error: no roles returned.")
                    continue
                }

                for role in roles {
                    if let name = role.roleName {
                        roleList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listRoles:", dump(error))
            throw error
        }
        return roleList
    }
```

- Para obter detalhes da API, consulte ListRoles a referência da API AWS SDK for Swift.

`ListUsers`

O código de exemplo a seguir mostra como usar `ListUsers`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    public func listUsers() async throws -> [MyUserRecord] {
        var userList: [MyUserRecord] = []

        // Use "Paginated" to get all the users.
        // This lets the SDK handle the 'isTruncated' in "ListUsersOutput".
        let input = ListUsersInput()
        let output = client.listUsersPaginated(input: input)

        do {
            for try await page in output {
                guard let users = page.users else {
                    continue
                }
                for user in users {
                    if let id = user.userId, let name = user.userName {
                        userList.append(MyUserRecord(id: id, name: name))
                    }
                }
            }
        }
        catch {
            print("ERROR: listUsers:", dump(error))
            throw error
        }
       return userList
    }
```

- Para obter detalhes da API, consulte ListUsers a referência da API AWS SDK for Swift.

`PutUserPolicy`

O código de exemplo a seguir mostra como usar `PutUserPolicy`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSIAM
import AWSS3


    func putUserPolicy(policyDocument: String, policyName: String, user:
 IAMClientTypes.User) async throws {
        let input = PutUserPolicyInput(
            policyDocument: policyDocument,
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.putUserPolicy(input: input)
        } catch {
            print("ERROR: putUserPolicy:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte PutUserPolicy a referência da API AWS SDK for Swift.

Exemplos do Lambda usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Lambda.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um perfil do IAM e uma função do Lambda e carregar o código de manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte Criar uma função do Lambda no console.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Defina a primeira função do Lambda, que simplesmente incrementa o valor especificado.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "increment",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/swift-server/swift-aws-lambda-runtime.git",
            branch: "main"),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "increment",
            dependencies: [
                .product(name: "AWSLambdaRuntime", package: "swift-aws-lambda-
runtime"),
            ],
            path: "Sources"
```

```
        )
    ]
)

import Foundation
import AWSLambdaRuntime

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct Request: Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The number to act upon.
    let number: Int
}

/// The contents of the response sent back to the client. This must be
/// `Encodable`.
struct Response: Encodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}


/// The Lambda function body.
///
/// - Parameters:
///   - event: The `Request` describing the request made by the
///     client.
///   - context: A `LambdaContext` describing the context in
///     which the lambda function is running.
///
/// - Returns: A `Response` object that will be encoded to JSON and sent
///   to the client by the Lambda runtime.
let incrementLambdaRuntime = LambdaRuntime {
        (event: Request, context: LambdaContext) -> Response in
    let action = event.action
    var answer: Int?

    if action != "increment" {
        context.logger.error("Unrecognized operation: \"\(action)\". The only
 supported action is \"increment\".")
    } else {
```

```
        answer = event.number + 1
        context.logger.info("The calculated answer is \(answer!).")
    }

    let response = Response(answer: answer)
    return response
}

// Run the Lambda runtime code.

try await incrementLambdaRuntime.run()
```

Defina a segunda função do Lambda, que executa uma operação aritmética em dois números.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "calculator",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/swift-server/swift-aws-lambda-runtime.git",
            branch: "main"),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "calculator",
```

```
            dependencies: [
                .product(name: "AWSLambdaRuntime", package: "swift-aws-lambda-
runtime"),
            ],
            path: "Sources"
        )
    ]
)

import Foundation
import AWSLambdaRuntime

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct Request: Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The first number to act upon.
    let x: Int
    /// The second number to act upon.
    let y: Int
}

/// A dictionary mapping operation names to closures that perform that
/// operation and return the result.
let actions = [
    "plus": { (x: Int, y: Int) -> Int in
        return x + y
    },
    "minus": { (x: Int, y: Int) -> Int in
        return x - y
    },
    "times": { (x: Int, y: Int) -> Int in
        return x * y
    },
    "divided-by": { (x: Int, y: Int) -> Int in
        return x / y
    }
]

/// The contents of the response sent back to the client. This must be
/// `Encodable`.
struct Response: Encodable, Sendable {
```

```
    /// The resulting value after performing the action.
    let answer: Int?
}


/// The Lambda function's entry point. Called by the Lambda runtime.
///
/// - Parameters:
///   - event: The `Request` describing the request made by the
///     client.
///   - context: A `LambdaContext` describing the context in
///     which the lambda function is running.
///
/// - Returns: A `Response` object that will be encoded to JSON and sent
///   to the client by the Lambda runtime.
let calculatorLambdaRuntime = LambdaRuntime {
        (_ event: Request, context: LambdaContext) -> Response in
    let action = event.action
    var answer: Int?
    var actionFunc: ((Int, Int) -> Int)?

    // Get the closure to run to perform the calculation.

    actionFunc = await actions[action]

    guard let actionFunc else {
        context.logger.error("Unrecognized operation '\(action)\'")
        return Response(answer: nil)
    }

    // Perform the calculation and return the answer.

    answer = actionFunc(event.x, event.y)

    guard let answer else {
        context.logger.error("Error computing \(event.x) \(action) \(event.y)")
    }
    context.logger.info("\(event.x) \(action) \(event.y) = \(answer)")

    return Response(answer: answer)
}

try await calculatorLambdaRuntime.run()
```

Defina o programa principal que invocará as duas funções do Lambda.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "lambda-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "lambda-basics",
            dependencies: [
                .product(name: "AWSLambda", package: "aws-sdk-swift"),
                .product(name: "AWSIAM", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources"
        )
```

```
    ]
)

//
/// An example demonstrating a variety of important AWS Lambda functions.

import ArgumentParser
import AWSIAM
import SmithyWaitersAPI
import AWSClientRuntime
import AWSLambda
import Foundation

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct IncrementRequest: Encodable, Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The number to act upon.
    let number: Int
}

struct Response: Encodable, Decodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}

struct CalculatorRequest: Encodable, Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The first number to act upon.
    let x: Int
    /// The second number to act upon.
    let y: Int
}

let exampleName = "SwiftLambdaRoleExample"
let basicsFunctionName = "lambda-basics-function"

/// The ARN of the standard IAM policy for execution of Lambda functions.
let policyARN = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"

struct ExampleCommand: ParsableCommand {
```

```
    // -MARK: Command arguments
    @Option(help: "Name of the IAM Role to use for the Lambda functions")
    var role = exampleName
    @Option(help: "Zip archive containing the 'increment' lambda function")
    var incpath: String
    @Option(help: "Zip archive containing the 'calculator' lambda function")
    var calcpath: String
    @Option(help: "Name of the Amazon S3 Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "lambda-basics",
        abstract: """
        This example demonstrates several common operations using AWS Lambda.
        """,
        discussion: """
        """
    )

    /// Returns the specified IAM role object.
    ///
    /// - Parameters:
    ///   - iamClient: `IAMClient` to use when looking for the role.
    ///   - roleName: The name of the role to check.
    ///
    /// - Returns: The `IAMClientTypes.Role` representing the specified role.
    func getRole(iamClient: IAMClient, roleName: String) async throws
                 -> IAMClientTypes.Role {
        do {
            let roleOutput = try await iamClient.getRole(
                input: GetRoleInput(
                    roleName: roleName
                )
            )

            guard let role = roleOutput.role else {
                throw ExampleError.roleNotFound
            }
            return role
        } catch {
            throw ExampleError.roleNotFound
        }
    }
```

```
    /// Create the AWS IAM role that will be used to access AWS Lambda.
    ///
    /// - Parameters:
    ///   - iamClient: The AWS `IAMClient` to use.
    ///   - roleName: The name of the AWS IAM role to use for Lambda.
    ///
    /// - Throws: `ExampleError.roleCreateError`
    ///
    /// - Returns: The `IAMClientTypes.Role` struct that describes the new role.
    func createRoleForLambda(iamClient: IAMClient, roleName: String) async throws ->
IAMClientTypes.Role {
        let output = try await iamClient.createRole(
            input: CreateRoleInput(
                assumeRolePolicyDocument:
                """
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"Service": "lambda.amazonaws.com"},
                            "Action": "sts:AssumeRole"
                        }
                    ]
                }
                """,
                roleName: roleName
            )
        )

        guard let role = output.role else {
            throw ExampleError.roleCreateError
        }

        // Wait for the role to be ready for use.

        _ = try await iamClient.waitUntilRoleExists(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetRoleInput(roleName: roleName)
        )
```

```
        return role
    }

    /// Detect whether or not the AWS Lambda function with the specified name
    /// exists, by requesting its function information.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - name: The name of the AWS Lambda function to find.
    ///
    /// - Returns: `true` if the Lambda function exists. Otherwise `false`.
    func doesLambdaFunctionExist(lambdaClient: LambdaClient, name: String) async ->
Bool {
        do {
            _ = try await lambdaClient.getFunction(
                input: GetFunctionInput(functionName: name)
            )
        } catch {
            return false
        }

        return true
    }

    /// Create the specified AWS Lambda function.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to create.
    ///   - roleArn: The ARN of the role to apply to the function.
    ///   - path: The path of the Zip archive containing the function.
    ///
    /// - Returns: `true` if the AWS Lambda was successfully created; `false`
    ///   if it wasn't.
    func createFunction(lambdaClient: LambdaClient, functionName: String,
                                roleArn: String?, path: String) async throws -> Bool
{
        do {
            // Read the Zip archive containing the AWS Lambda function.

            let zipUrl = URL(fileURLWithPath: path)
            let zipData = try Data(contentsOf: zipUrl)
```

```
            // Create the AWS Lambda function that runs the specified code,
            // using the name given on the command line. The Lambda function
            // will run using the Amazon Linux 2 runtime.

            _ = try await lambdaClient.createFunction(
                input: CreateFunctionInput(
                    code: LambdaClientTypes.FunctionCode(zipFile: zipData),
                    functionName: functionName,
                    handler: "handle",
                    role: roleArn,
                    runtime: .providedal2
                )
            )
        } catch {
            print("*** Error creating Lambda function:")
            dump(error)
            return false
        }

        // Wait for a while to be sure the function is done being created.

        let output = try await lambdaClient.waitUntilFunctionActiveV2(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetFunctionInput(functionName: functionName)
        )

        switch output.result {
            case .success:
                return true
            case .failure:
                return false
        }
    }

    /// Update the AWS Lambda function with new code to run when the function
    /// is invoked.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to update.
```

```
///   - path: The pathname of the Zip file containing the packaged Lambda
///     function.
/// - Throws: `ExampleError.zipFileReadError`
/// - Returns: `true` if the function's code is updated successfully.
///   Otherwise, returns `false`.
func updateFunctionCode(lambdaClient: LambdaClient, functionName: String,
                        path: String) async throws -> Bool {
    let zipUrl = URL(fileURLWithPath: path)
    let zipData: Data

    // Read the function's Zip file.

    do {
        zipData = try Data(contentsOf: zipUrl)
    } catch {
        throw ExampleError.zipFileReadError
    }

    // Update the function's code and wait for the updated version to be
    // ready for use.

    do {
        _ = try await lambdaClient.updateFunctionCode(
            input: UpdateFunctionCodeInput(
                functionName: functionName,
                zipFile: zipData
            )
        )
    } catch {
        return false
    }

    let output = try await lambdaClient.waitUntilFunctionUpdatedV2(
        options: WaiterOptions(
            maxWaitTime: 20,
            minDelay: 0.5,
            maxDelay: 2
        ),
        input: GetFunctionInput(
            functionName: functionName
        )
    )

    switch output.result {
```

```
            case .success:
                return true
            case .failure:
                return false
        }
    }

    /// Tell the server-side component to log debug output by setting its
    /// environment's `LOG_LEVEL` to `DEBUG`.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to enable debug
    ///     logging for.
    ///
    /// - Throws: `ExampleError.environmentResponseMissingError`,
    ///   `ExampleError.updateFunctionConfigurationError`,
    ///   `ExampleError.environmentVariablesMissingError`,
    ///   `ExampleError.logLevelIncorrectError`,
    ///   `ExampleError.updateFunctionConfigurationError`
    func enableDebugLogging(lambdaClient: LambdaClient, functionName: String) async
throws {
        let envVariables = [
            "LOG_LEVEL": "DEBUG"
        ]
        let environment = LambdaClientTypes.Environment(variables: envVariables)

        do {
            let output = try await lambdaClient.updateFunctionConfiguration(
                input: UpdateFunctionConfigurationInput(
                    environment: environment,
                    functionName: functionName
                )
            )

            guard let response = output.environment else {
                throw ExampleError.environmentResponseMissingError
            }

            if response.error != nil {
                throw ExampleError.updateFunctionConfigurationError
            }

            guard let retVariables = response.variables else {
```

```
                throw ExampleError.environmentVariablesMissingError
            }

            for envVar in retVariables {
                if envVar.key == "LOG_LEVEL" && envVar.value != "DEBUG" {
                    print("*** Log level is not set to DEBUG!")
                    throw ExampleError.logLevelIncorrectError
                }
            }
        } catch {
            throw ExampleError.updateFunctionConfigurationError
        }
    }

    /// Returns an array containing the names of all AWS Lambda functions
    /// available to the user.
    ///
    /// - Parameter lambdaClient: The `IAMClient` to use.
    ///
    /// - Throws: `ExampleError.listFunctionsError`
    ///
    /// - Returns: An array of lambda function name strings.
    func getFunctionNames(lambdaClient: LambdaClient) async throws -> [String] {
        let pages = lambdaClient.listFunctionsPaginated(
            input: ListFunctionsInput()
        )

        var functionNames: [String] = []

        for try await page in pages {
            guard let functions = page.functions else {
                throw ExampleError.listFunctionsError
            }

            for function in functions {
                functionNames.append(function.functionName ?? "<unknown>")
            }
        }

        return functionNames
    }

    /// Invoke the Lambda function to increment a value.
    ///
```

```
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - number: The number to increment.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: An integer number containing the incremented value.
    func invokeIncrement(lambdaClient: LambdaClient, number: Int) async throws ->
Int {
        do {
            let incRequest = IncrementRequest(action: "increment", number: number)
            let incData = try! JSONEncoder().encode(incRequest)

            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: incData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }
    }

    /// Invoke the calculator Lambda function.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - action: Which arithmetic operation to perform: "plus", "minus",
    ///     "times", or "divided-by".
    ///   - x: The first number to use in the computation.
    ///   - y: The second number to use in the computation.
    ///
```

```
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: The computed answer as an `Int`.
    func invokeCalculator(lambdaClient: LambdaClient, action: String, x: Int, y:
Int) async throws -> Int {
        do {
            let calcRequest = CalculatorRequest(action: action, x: x, y: y)
            let calcData = try! JSONEncoder().encode(calcRequest)

            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: calcData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }

    }

    /// Perform the example's tasks.
    func basics() async throws {
        let iamClient = try await IAMClient(
            config: IAMClient.IAMClientConfiguration(region: region)
        )

        let lambdaClient = try await LambdaClient(
            config: LambdaClient.LambdaClientConfiguration(region: region)
        )

        /// The IAM role to use for the example.
        var iamRole: IAMClientTypes.Role
```

```
        // Look for the specified role. If it already exists, use it. If not,
        // create it and attach the desired policy to it.

        do {
            iamRole = try await getRole(iamClient: iamClient, roleName: role)
        } catch ExampleError.roleNotFound {
            // The role wasn't found, so create it and attach the needed
            // policy.

            iamRole = try await createRoleForLambda(iamClient: iamClient, roleName:
role)

            do {
                _ = try await iamClient.attachRolePolicy(
                    input: AttachRolePolicyInput(policyArn: policyARN, roleName:
role)
                )
            } catch {
                throw ExampleError.policyError
            }
        }

        // Give the policy time to attach to the role.

        sleep(5)

        // Look to see if the function already exists. If it does, throw an
        // error.

        if await doesLambdaFunctionExist(lambdaClient: lambdaClient, name:
basicsFunctionName) {
            throw ExampleError.functionAlreadyExists
        }

        // Create, then invoke, the "increment" version of the calculator
        // function.

        print("Creating the increment Lambda function...")
        if try await createFunction(lambdaClient: lambdaClient, functionName:
basicsFunctionName,
                                    roleArn: iamRole.arn, path: incpath) {
            print("Running increment function calls...")
            for number in 0...4 {
```

```
                do {
                    let answer = try await invokeIncrement(lambdaClient:
lambdaClient, number: number)
                    print("Increment \(number) = \(answer)")
                } catch {
                    print("Error incrementing \(number): ",
error.localizedDescription)
                }
            }
        } else {
            print("*** Failed to create the increment function.")
        }

        // Enable debug logging.

        print("\nEnabling debug logging...")
        try await enableDebugLogging(lambdaClient: lambdaClient, functionName:
basicsFunctionName)

        // Change it to a basic arithmetic calculator. Then invoke it a few
        // times.

        print("\nReplacing the Lambda function with a calculator...")

        if try await updateFunctionCode(lambdaClient: lambdaClient, functionName:
basicsFunctionName,
                                    path: calcpath) {
            print("Running calculator function calls...")
            for x in [6, 10] {
                for y in [2, 4] {
                    for action in ["plus", "minus", "times", "divided-by"] {
                        do {
                            let answer = try await invokeCalculator(lambdaClient:
lambdaClient, action: action, x: x, y: y)
                            print("\(x) \(action) \(y) = \(answer)")
                        } catch {
                            print("Error calculating \(x) \(action) \(y): ",
error.localizedDescription)
                        }
                    }
                }
            }
        }
```

```
        // List all lambda functions.

        let functionNames = try await getFunctionNames(lambdaClient: lambdaClient)

        if functionNames.count > 0 {
            print("\nAWS Lambda functions available on your account:")
            for name in functionNames {
                print("  \(name)")
            }
        }

        // Delete the lambda function.

        print("Deleting lambda function...")

        do {
            _ = try await lambdaClient.deleteFunction(
                input: DeleteFunctionInput(
                    functionName: "lambda-basics-function"
                )
            )
        } catch {
            print("Error: Unable to delete the function.")
        }

        // Detach the role from the policy, then delete the role.

        print("Deleting the AWS IAM role...")

        do {
            _ = try await iamClient.detachRolePolicy(
                input: DetachRolePolicyInput(
                    policyArn: policyARN,
                    roleName: role
                )
            )
            _ = try await iamClient.deleteRole(
                input: DeleteRoleInput(
                    roleName: role
                )
            )
        } catch {
            throw ExampleError.deleteRoleError
        }
```

```
    }
}

// -MARK: - Entry point

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.basics()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}


/// Errors thrown by the example's functions.
enum ExampleError: Error {
    /// An AWS Lambda function with the specified name already exists.
    case functionAlreadyExists
    /// The specified role doesn't exist.
    case roleNotFound
    /// Unable to create the role.
    case roleCreateError
    /// Unable to delete the role.
    case deleteRoleError
    /// Unable to attach a policy to the role.
    case policyError
    /// Unable to get the executable directory.
    case executableNotFound
    /// An error occurred creating a lambda function.
    case createLambdaError
    /// An error occurred invoking the lambda function.
    case invokeError
    /// No answer received from the invocation.
    case noAnswerReceived
    /// Unable to list the AWS Lambda functions.
    case listFunctionsError
    /// Unable to update the AWS Lambda function.
```

```
    case updateFunctionError
    /// Unable to update the function configuration.
    case updateFunctionConfigurationError
    /// The environment response is missing after an
    /// UpdateEnvironmentConfiguration attempt.
    case environmentResponseMissingError
    /// The environment variables are missing from the EnvironmentResponse and
    /// no errors occurred.
    case environmentVariablesMissingError
    /// The log level is incorrect after attempting to set it.
    case logLevelIncorrectError
    /// Unable to load the AWS Lambda function's Zip file.
    case zipFileReadError

    var errorDescription: String? {
        switch self {
        case .functionAlreadyExists:
            return "An AWS Lambda function with that name already exists."
        case .roleNotFound:
            return "The specified role doesn't exist."
        case .deleteRoleError:
            return "Unable to delete the AWS IAM role."
        case .roleCreateError:
            return "Unable to create the specified role."
        case .policyError:
            return "An error occurred attaching the policy to the role."
        case .executableNotFound:
            return "Unable to find the executable program directory."
        case .createLambdaError:
            return "An error occurred creating a lambda function."
        case .invokeError:
            return "An error occurred invoking a lambda function."
        case .noAnswerReceived:
            return "No answer received from the lambda function."
        case .listFunctionsError:
            return "Unable to list the AWS Lambda functions."
        case .updateFunctionError:
            return "Unable to update the AWS lambda function."
        case .updateFunctionConfigurationError:
            return "Unable to update the AWS lambda function configuration."
        case .environmentResponseMissingError:
            return "The environment is missing from the response after updating the
function configuration."
        case .environmentVariablesMissingError:
```

```
            return "While no error occurred, no environment variables were returned
 following function configuration."
        case .logLevelIncorrectError:
            return "The log level is incorrect after attempting to set it to DEBUG."
        case .zipFileReadError:
            return "Unable to read the AWS Lambda function."
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Ações

`CreateFunction`

O código de exemplo a seguir mostra como usar `CreateFunction`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        do {
```

```
            // Read the Zip archive containing the AWS Lambda function.

            let zipUrl = URL(fileURLWithPath: path)
            let zipData = try Data(contentsOf: zipUrl)

            // Create the AWS Lambda function that runs the specified code,
            // using the name given on the command line. The Lambda function
            // will run using the Amazon Linux 2 runtime.

            _ = try await lambdaClient.createFunction(
                input: CreateFunctionInput(
                    code: LambdaClientTypes.FunctionCode(zipFile: zipData),
                    functionName: functionName,
                    handler: "handle",
                    role: roleArn,
                    runtime: .providedal2
                )
            )
        } catch {
            print("*** Error creating Lambda function:")
            dump(error)
            return false
        }
```

- Para obter detalhes da API, consulte CreateFunction a referência da API AWS SDK for Swift.

`DeleteFunction`

O código de exemplo a seguir mostra como usar `DeleteFunction`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation
```

```
        do {
            _ = try await lambdaClient.deleteFunction(
                input: DeleteFunctionInput(
                    functionName: "lambda-basics-function"
                )
            )
        } catch {
            print("Error: Unable to delete the function.")
        }
```

- Para obter detalhes da API, consulte DeleteFunction a referência da API AWS SDK for Swift.

`GetFunction`

O código de exemplo a seguir mostra como usar `GetFunction`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Detect whether or not the AWS Lambda function with the specified name
    /// exists, by requesting its function information.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - name: The name of the AWS Lambda function to find.
    ///
    /// - Returns: `true` if the Lambda function exists. Otherwise `false`.
    func doesLambdaFunctionExist(lambdaClient: LambdaClient, name: String) async ->
 Bool {
        do {
            _ = try await lambdaClient.getFunction(
```

```
                input: GetFunctionInput(functionName: name)
            )
        } catch {
            return false
        }

        return true
    }
```

- Para obter detalhes da API, consulte GetFunction a referência da API AWS SDK for Swift.

Invocar

O código de exemplo a seguir mostra como usar `Invoke`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Invoke the Lambda function to increment a value.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - number: The number to increment.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: An integer number containing the incremented value.
    func invokeIncrement(lambdaClient: LambdaClient, number: Int) async throws ->
 Int {
        do {
            let incRequest = IncrementRequest(action: "increment", number: number)
            let incData = try! JSONEncoder().encode(incRequest)
```

```
            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: incData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }
    }
```

- Para obter detalhes sobre a API, consulte Invoke na Referência da API AWS SDK para Swift.

`ListFunctions`

O código de exemplo a seguir mostra como usar `ListFunctions`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation
```

```
    /// Returns an array containing the names of all AWS Lambda functions
    /// available to the user.
    ///
    /// - Parameter lambdaClient: The `IAMClient` to use.
    ///
    /// - Throws: `ExampleError.listFunctionsError`
    ///
    /// - Returns: An array of lambda function name strings.
    func getFunctionNames(lambdaClient: LambdaClient) async throws -> [String] {
        let pages = lambdaClient.listFunctionsPaginated(
            input: ListFunctionsInput()
        )

        var functionNames: [String] = []

        for try await page in pages {
            guard let functions = page.functions else {
                throw ExampleError.listFunctionsError
            }

            for function in functions {
                functionNames.append(function.functionName ?? "<unknown>")
            }
        }

        return functionNames
    }
```

- Para obter detalhes da API, consulte ListFunctions a referência da API AWS SDK for Swift.

`UpdateFunctionCode`

O código de exemplo a seguir mostra como usar `UpdateFunctionCode`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        let zipUrl = URL(fileURLWithPath: path)
        let zipData: Data

        // Read the function's Zip file.

        do {
            zipData = try Data(contentsOf: zipUrl)
        } catch {
            throw ExampleError.zipFileReadError
        }

        // Update the function's code and wait for the updated version to be
        // ready for use.

        do {
            _ = try await lambdaClient.updateFunctionCode(
                input: UpdateFunctionCodeInput(
                    functionName: functionName,
                    zipFile: zipData
                )
            )
        } catch {
            return false
        }
```

- Para obter detalhes da API, consulte UpdateFunctionCode a referência da API AWS SDK for Swift.

`UpdateFunctionConfiguration`

O código de exemplo a seguir mostra como usar `UpdateFunctionConfiguration`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Tell the server-side component to log debug output by setting its
    /// environment's `LOG_LEVEL` to `DEBUG`.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to enable debug
    ///     logging for.
    ///
    /// - Throws: `ExampleError.environmentResponseMissingError`,
    ///   `ExampleError.updateFunctionConfigurationError`,
    ///   `ExampleError.environmentVariablesMissingError`,
    ///   `ExampleError.logLevelIncorrectError`,
    ///   `ExampleError.updateFunctionConfigurationError`
    func enableDebugLogging(lambdaClient: LambdaClient, functionName: String) async
 throws {
        let envVariables = [
            "LOG_LEVEL": "DEBUG"
        ]
        let environment = LambdaClientTypes.Environment(variables: envVariables)

        do {
            let output = try await lambdaClient.updateFunctionConfiguration(
                input: UpdateFunctionConfigurationInput(
                    environment: environment,
                    functionName: functionName
                )
            )

            guard let response = output.environment else {
                throw ExampleError.environmentResponseMissingError
```

```
            }

            if response.error != nil {
                throw ExampleError.updateFunctionConfigurationError
            }

            guard let retVariables = response.variables else {
                throw ExampleError.environmentVariablesMissingError
            }

            for envVar in retVariables {
                if envVar.key == "LOG_LEVEL" && envVar.value != "DEBUG" {
                    print("*** Log level is not set to DEBUG!")
                    throw ExampleError.logLevelIncorrectError
                }
            }
        } catch {
            throw ExampleError.updateFunctionConfigurationError
        }
    }
```

- Para obter detalhes da API, consulte UpdateFunctionConfiguration a referência da API AWS SDK for Swift.

Exemplos de código do Amazon RDS usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon RDS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um grupo de parâmetros de banco de dados e definir os valores dos parâmetros.
- Criar uma instância de banco de dados configurada para usar o grupo de parâmetros. A instância de banco de dados também contém um banco de dados.
- Criar um snapshot da instância.
- Exclua a instância e o grupo de parâmetros.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "rds-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
```

```
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "rds-scenario",
            dependencies: [
                .product(name: "AWSRDS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

O arquivo de código Swift, `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Relational Database Service (RDS).
//

import ArgumentParser
import Foundation
import AWSRDS

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"
    @Option(help: "The username to use for the database administrator.")
    var dbUsername = "admin"
    @Option(help: "The password to use for the database administrator.")
```

```
    var dbPassword: String

    static var configuration = CommandConfiguration(
        commandName: "rds-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon RDS Instances
        using the AWS SDK for Swift.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let example = try await Example(region: awsRegion, username: dbUsername,
 password: dbPassword)

        await example.run()
    }
}

class Example {
    let rdsClient: RDSClient

    // Storage for AWS RDS properties

    let dbUsername: String
    let dbPassword: String
    var dbInstanceIdentifier: String
    var dbSnapshotIdentifier: String
    var dbParameterGroupName: String
    var dbParameterGroup: RDSClientTypes.DBParameterGroup?
    var selectedEngineVersion: String?

    init(region: String, username: String, password: String) async throws{
        let rdsConfig = try await RDSClient.RDSClientConfiguration(region: region)
        rdsClient = RDSClient(config: rdsConfig)

        dbUsername = username
        dbPassword = password
        dbParameterGroupName = ""
        dbInstanceIdentifier = ""
        dbSnapshotIdentifier = ""
    }
```

```
    /// The example's main body.
    func run() async {
        var parameterGroupFamilies: Set<String> = []

        //=====================================================================
        // 1. Get available database engine families for MySQL.
        //=====================================================================

        let engineVersions = await getDBEngineVersions(engineName: "mysql")

        for version in engineVersions {
            if version.dbParameterGroupFamily != nil {
                parameterGroupFamilies.insert(version.dbParameterGroupFamily!)
            }
        }

        if engineVersions.count > 0 {
            selectedEngineVersion = engineVersions.last!.engineVersion
        } else {
            print("*** Unable to find a valid database engine version. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Found \(parameterGroupFamilies.count) parameter group families:")
        for family in parameterGroupFamilies {
            print("    \(family)")
        }

        //=====================================================================
        // 2. Select an engine family and create a custom DB parameter group.
        //    We select a family by sorting the set of family names, then
        //    choosing the last one.
        //=====================================================================

        let sortedFamilies = parameterGroupFamilies.sorted()

        guard let selectedFamily = sortedFamilies.last else {
            print("*** Unable to find a database engine family. Canceling
operations.")
            await cleanUp()
            return
```

```
        }

        print("Selected database engine family \(selectedFamily)")

        dbParameterGroupName = tempName(prefix: "rds-example")
        print("Creating a database parameter group named \(dbParameterGroupName)
using \(selectedFamily)")
        dbParameterGroup = await createDBParameterGroup(groupName:
dbParameterGroupName,
                                                        familyName: selectedFamily)

        //======================================================================
        // 3. Get the parameter group's details.
        //======================================================================

        print("Getting the database parameter group list...")
        let dbParameterGroupList = await describeDBParameterGroups(groupName:
dbParameterGroupName)
        guard let dbParameterGroupList else {
            await cleanUp()
            return
        }

        print("Found \(dbParameterGroupList.count) parameter groups...")
        for group in dbParameterGroupList {
            print("    \(group.dbParameterGroupName ?? "<unknown>")")
        }
        print()

        //======================================================================
        // 4. Get a list of the parameter group's parameters. This list is
        //    likely to be long, so use pagination. Find the
        //    auto_increment_offset and auto_increment_increment parameters.
        //======================================================================

        let parameters = await describeDBParameters(groupName: dbParameterGroupName)

        //======================================================================
        // 5. Parse and display each parameter's name, description, and
        //    allowed values.
        //======================================================================

        for parameter in parameters {
            let name = parameter.parameterName
```

```
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }

            if name == "auto_increment_offset" || name ==
"auto_increment_increment" {
                print("Parameter \(name):")
                print("          Value: \(parameter.parameterValue ??
"<undefined>")")
                print("      Data type: \(parameter.dataType ?? "<unknown>")")
                print("    Description: \(parameter.description ?? "")")
                print(" Allowed values: \(parameter.allowedValues ??
"<unspecified")")
                print(String(repeating: "=", count: 78))
            }
        }

        //=====================================================================
        // 6. Modify both the auto_increment_offset and
        //    auto_increment_increment parameters in one call in the custom
        //    parameter group. Set their parameterValue fields to a new
        //    permitted value.
        //=====================================================================

        print("Setting auto_increment_offset and auto_increment_increment both to
5...")
        await modifyDBParameters(groupName: dbParameterGroupName)

        //=====================================================================
        // 7. Get and display the updated parameters, specifying a source of
        //    "user" to get only the modified parameters.
        //=====================================================================

        let updatedParameters = await describeDBParameters(groupName:
dbParameterGroupName, source: "user")

        for parameter in updatedParameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }
```

```
            print("Parameter \(name):")
            print("          Value: \(parameter.parameterValue ?? "<undefined>")")
            print("      Data type: \(parameter.dataType ?? "<unknown>")")
            print("    Description: \(parameter.description ?? "")")
            print(" Allowed values: \(parameter.allowedValues ?? "<unspecified")")
            print(String(repeating: "=", count: 78))
        }

        //=====================================================================
        // 8. Get a list of allowed engine versions using
        //    DescribeRDSEngineVersions.
        //=====================================================================

        await listAllowedEngines(family: selectedFamily)

        //=====================================================================
        // 9. Get a list of micro instance classes available for the selected
        //    engine and engine version.
        //=====================================================================

        let dbInstanceClass = await chooseMicroInstance(engine: "mysql",
engineVersion: selectedEngineVersion)
        guard let dbInstanceClass else {
            print("Did not get a valid instance class. Canceling operations.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 10. Create an RDS database that contains a MySQL database and uses
        //     the parameter group we created.
        //=====================================================================

        print("Creating the database instance...")

        guard let instanceClass = dbInstanceClass.dbInstanceClass else {
            print("Instance class name is unknown. Canceling operations.")
            await cleanUp()
            return
        }

        dbInstanceIdentifier = tempName(prefix: "sample-identifier")
        let dbInstanceArn = await createDBInstance(
            name: "SampleDatabase\(Int.random(in: 1000000..<1000000000))",
```

```
            instanceIdentifier: dbInstanceIdentifier,
            parameterGroupName: dbParameterGroupName,
            engine: "mysql",
            engineVersion: selectedEngineVersion!,
            instanceClass: instanceClass,
            username: dbUsername,
            password: dbPassword
        )

        if dbInstanceArn == nil {
            await cleanUp()
            return
        }

        //=====================================================================
        // 11. Wait for the database instance to be ready by calling
        //     DescribeDBInstances repeatedly until it reports
        //     dbInstanceStatus as "available". This can take upwards of 10
        //     minutes, let the user know that.
        //=====================================================================

        guard let endpoint = await waitUntilDBInstanceReady(instanceIdentifier:
dbInstanceIdentifier) else {
            print("\nDid not get a valid endpoint from AWS RDS.")
            await cleanUp()
            return
        }

        guard let endpointAddress = endpoint.address else {
            print("\nNo endpoint address returned.")
            await cleanUp()
            return
        }
        guard let endpointPort = endpoint.port else {
            print("\nNo endpoint port returned.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 12. Display connection information for the database instance.
        //=====================================================================
```

```
        print("\nTo connect to the new database instance using 'mysql' from the
shell:")
        print("    mysql -h \(endpointAddress) -P \(endpointPort) -u
\(self.dbUsername)")

        //=====================================================================
        // 13. Create a snapshot of the database instance.
        //=====================================================================

        dbSnapshotIdentifier = tempName(prefix: "sample-snapshot")
        await createDBSnapshot(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        //=====================================================================
        // 14. Wait for the snapshot to be ready.
        //=====================================================================

        await waitUntilDBSnapshotReady(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        // That's it! Clean up and exit!

        print("Example complete! Cleaning up...")
        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items.
    func cleanUp() async {
        print("Deleting the database instance \(dbInstanceIdentifier)...")
        await deleteDBInstance(instanceIdentifier: dbInstanceIdentifier)
        await waitUntilDBInstanceDeleted(instanceIdentifier: dbInstanceIdentifier)

        print("Deleting the database parameter group \(dbParameterGroupName)...")
        await deleteDBParameterGroup(groupName: dbParameterGroupName)
    }

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
```

```
    func getDBEngineVersions(engineName: String) async ->
[RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
        } catch {
            return []
        }
    }

    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
        } catch {
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
```

```
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
    func describeDBParameterGroups(groupName: String) async ->
[RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
[RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }
```

```
                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
\(error.localizedDescription)")
            return []
        }

        return parameterList
    }

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
```

```
    }

    /// Output a list of the database engine versions supported by the
    /// specified family.
    ///
    /// - Parameter family: The family for which to list allowed database
    ///   engines.
    func listAllowedEngines(family: String?) async {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    dbParameterGroupFamily: family,
                    engine: "mysql"
                )
            )

            guard let engineVersions = output.dbEngineVersions else {
                print("No engine versions returned.")
                return
            }

            print("Found \(engineVersions.count) database engine versions:")
            for version in engineVersions {
                print("    \(version.engineVersion ?? "<unknown>"):
\(version.dbEngineDescription ?? "")")
            }
        } catch {
            print("*** Error getting database engine version list:
\(error.localizedDescription)")
            return
        }
    }

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
    ///   - engineVersion: The database version for which to list instances.
    ///
    /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
    ///   the selected instance type.
```

```
   func chooseMicroInstance(engine: String = "mysql", engineVersion: String? = nil)
async -> RDSClientTypes.OrderableDBInstanceOption? {
       do {
           let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
               input: DescribeOrderableDBInstanceOptionsInput(
                   engine: engine,
                   engineVersion: engineVersion
               )
           )

           var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []

           for try await page in pages {
               guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                   continue
               }

               for dbInstanceOption in orderableDBInstanceOptions {
                   guard let className = dbInstanceOption.dbInstanceClass else {
                       continue
                   }
                   if className.contains("micro") {
                       optionsList.append(dbInstanceOption)
                   }
               }
           }

           print("Found \(optionsList.count) database instances of 'micro' class
types:")
           for dbInstanceOption in optionsList {
               print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
           }

           return optionsList[0]
       } catch {
           print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
           return nil
       }
   }
```

```
    /// Create a new database instance.
    ///
    /// - Parameters:
    ///   - name: The name of the database to create.
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
    ///   - instanceClass: The memory and compute capacity of the database
    ///     instance, such as `db.m5.large``.
    ///   - username: The admin user's username to establish for the new
    ///     instance.
    ///   - password: The password to use for the specified user's access.
    ///
    /// - Returns: A string indicating the ARN of the newly created database
    ///   instance, or nil if the instance couldn't be created.
    func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                          engine: String, engineVersion: String, instanceClass:
String,
                          username: String, password: String) async -> String? {
        do {
            let output = try await rdsClient.createDBInstance(
                input: CreateDBInstanceInput(
                    allocatedStorage: 100,
                    dbInstanceClass: instanceClass,
                    dbInstanceIdentifier: instanceIdentifier,
                    dbName: name,
                    dbParameterGroupName: parameterGroupName,
                    engine: engine,
                    engineVersion: engineVersion,
                    masterUserPassword: password,
                    masterUsername: username,
                    storageType: "gp2"
                )
            )

            guard let dbInstance = output.dbInstance else {
                print("*** Unable to get the database instance.")
                return nil
            }
```

```
            return dbInstance.dbInstanceArn
        } catch {
            print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Wait until the specified database is available to use.
    ///
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
```

```
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Create a snapshot of the specified name.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database instance to
    ///     snapshot.
    ///   - snapshotIdentifier: A unique identifier to give the newly-created
    ///     snapshot.
    func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
async {
        do {
            let output = try await rdsClient.createDBSnapshot(
                input: CreateDBSnapshotInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    dbSnapshotIdentifier: snapshotIdentifier
                )
            )

            guard let snapshot = output.dbSnapshot else {
                print("No snapshot returned.")
                return
            }

            print("The snapshot has been created with ID \(snapshot.dbiResourceId ??
"<unknown>")")
        } catch {
            print("*** Unable to create the database snapshot named
\(snapshotIdentifier): \(error.localizedDescription)")
        }
    }

    /// Wait until the specified database snapshot is available to use.
    ///
```

```
   /// - Parameters:
   ///   - instanceIdentifier: The identifier of the database for which the
   ///     snapshot was taken.
   ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
   func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
String) async {
       var snapshotReady = false

       putString("Waiting for the snapshot to be ready...")

       do {
           while !snapshotReady {
               let output = try await rdsClient.describeDBSnapshots(
                   input: DescribeDBSnapshotsInput(
                       dbInstanceIdentifier: instanceIdentifier,
                       dbSnapshotIdentifier: snapshotIdentifier
                   )
               )

               guard let snapshotList = output.dbSnapshots else {
                   return
               }

               for snapshot in snapshotList {
                   guard let snapshotReadyStr = snapshot.status else {
                       return
                   }

                   if snapshotReadyStr.contains("available") {
                       snapshotReady = true
                       print()
                   } else {
                       putString(".")
                       do {
                           try await Task.sleep(for: .seconds(15))
                       } catch {
                           print("\n*** Error pausing the task!")
                       }
                   }
               }
           }
       } catch {
           print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
```

```
        }
    }

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
        } catch {
            print("*** Error deleting the database instance \(instanceIdentifier):
\(error.localizedDescription)")
        }
    }

    /// Wait until the specified database instance has been deleted.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to wait for.
    func waitUntilDBInstanceDeleted(instanceIdentifier: String) async {
        putString("Waiting for the database instance to be deleted. This may take a
few minutes...")
        do {
            var isDatabaseDeleted = false
            var foundInstance = false

            while !isDatabaseDeleted {
                let output = try await rdsClient.describeDBInstances(input:
DescribeDBInstancesInput())
                guard let instanceList = output.dbInstances else {
                    return
                }

                foundInstance = false

                for instance in instanceList {
```

```
                    guard let foundInstanceIdentifier =
instance.dbInstanceIdentifier else {
                        continue
                    }

                    if instanceIdentifier == foundInstanceIdentifier {
                        foundInstance = true
                        break
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
                if !foundInstance {
                    isDatabaseDeleted = true
                    print()
                }
            }
        } catch {
            print("\n*** Error waiting for the database instance to be deleted:
\(error.localizedDescription)")
        }
    }

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

```
    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Print a string to standard output without a trailing newline, and
    /// without buffering.
    ///
    /// - Parameter str: The string to output.
    func putString(_ str: String = "") {
        if str.length >= 1 {
            let data = str.data(using: .utf8)
            guard let data else {
                return
            }
            FileHandle.standardOutput.write(data)
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
```

```
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Ações

`CreateDBInstance`

O código de exemplo a seguir mostra como usar `CreateDBInstance`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Create a new database instance.
    ///
    /// - Parameters:
```

```
   ///   - name: The name of the database to create.
   ///   - instanceIdentifier: The identifier to give the new database
   ///     instance.
   ///   - parameterGroupName: The name of the parameter group to associate
   ///     with the new database instance.
   ///   - engine: The database engine to use.
   ///   - engineVersion: The version of the database given by `engine` to
   ///     use.
   ///   - instanceClass: The memory and compute capacity of the database
   ///     instance, such as `db.m5.large``.
   ///   - username: The admin user's username to establish for the new
   ///     instance.
   ///   - password: The password to use for the specified user's access.
   ///
   /// - Returns: A string indicating the ARN of the newly created database
   ///   instance, or nil if the instance couldn't be created.
   func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                         engine: String, engineVersion: String, instanceClass:
String,
                         username: String, password: String) async -> String? {
       do {
           let output = try await rdsClient.createDBInstance(
               input: CreateDBInstanceInput(
                   allocatedStorage: 100,
                   dbInstanceClass: instanceClass,
                   dbInstanceIdentifier: instanceIdentifier,
                   dbName: name,
                   dbParameterGroupName: parameterGroupName,
                   engine: engine,
                   engineVersion: engineVersion,
                   masterUserPassword: password,
                   masterUsername: username,
                   storageType: "gp2"
               )
           )

           guard let dbInstance = output.dbInstance else {
               print("*** Unable to get the database instance.")
               return nil
           }

           return dbInstance.dbInstanceArn
       } catch {
```

```
            print("*** An error occurred while creating the database instance:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Consulte detalhes da API em CreateDBInstance na Referência de API do AWS SDK para Swift.

`CreateDBParameterGroup`

O código de exemplo a seguir mostra como usar `CreateDBParameterGroup`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
 RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
        } catch {
```

```
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte CreateDBParameterGroup a referência da API AWS SDK for Swift.

`CreateDBSnapshot`

O código de exemplo a seguir mostra como usar `CreateDBSnapshot`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Create a snapshot of the specified name.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database instance to
    ///     snapshot.
    ///   - snapshotIdentifier: A unique identifier to give the newly-created
    ///     snapshot.
    func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
 async {
        do {
            let output = try await rdsClient.createDBSnapshot(
                input: CreateDBSnapshotInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    dbSnapshotIdentifier: snapshotIdentifier
                )
            )
```

```
            guard let snapshot = output.dbSnapshot else {
                print("No snapshot returned.")
                return
            }

            print("The snapshot has been created with ID \(snapshot.dbiResourceId ??
 "<unknown>")")
        } catch {
            print("*** Unable to create the database snapshot named
\(snapshotIdentifier): \(error.localizedDescription)")
        }
    }
```

- Consulte detalhes da API em CreateDBSnapshot na Referência de API do AWS SDK para Swift.

`DeleteDBInstance`

O código de exemplo a seguir mostra como usar `DeleteDBInstance`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
```

```
                    skipFinalSnapshot: true
                )
            )
        } catch {
            print("*** Error deleting the database instance \(instanceIdentifier): 
\(error.localizedDescription)")
        }
    }
```

- Consulte detalhes da API em DeleteDBInstance na Referência de API do AWS SDK para Swift.

`DeleteDBParameterGroup`

O código de exemplo a seguir mostra como usar `DeleteDBParameterGroup`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName): 
\(error.localizedDescription)")
        }
```

```
    }
```

- Para obter detalhes da API, consulte DeleteDBParameterGroup a referência da API AWS SDK for Swift.

`DescribeDBEngineVersions`

O código de exemplo a seguir mostra como usar `DescribeDBEngineVersions`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
 [RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
        } catch {
            return []
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBEngineVersions a referência da API AWS SDK for Swift.

`DescribeDBInstances`

O código de exemplo a seguir mostra como usar `DescribeDBInstances`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Wait until the specified database is available to use.
    ///
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
 RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
 may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus
```

```
                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Consulte detalhes da API em DescribeDBInstances na Referência de API do AWS SDK para Swift.

`DescribeDBParameterGroups`

O código de exemplo a seguir mostra como usar `DescribeDBParameterGroups`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
    func describeDBParameterGroups(groupName: String) async ->
 [RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte DescribeDBParameterGroups a referência da API AWS SDK for Swift.

`DescribeDBParameters`

O código de exemplo a seguir mostra como usar `DescribeDBParameters`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
 [RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
 \(error.localizedDescription)")
            return []
        }

        return parameterList
    }
```

- Consulte detalhes da API em DescribeDBParameter na Referência de API do AWS SDK para Swift.

`DescribeDBSnapshots`

O código de exemplo a seguir mostra como usar `DescribeDBSnapshots`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Wait until the specified database snapshot is available to use.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database for which the
    ///     snapshot was taken.
    ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
    func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
 String) async {
        var snapshotReady = false

        putString("Waiting for the snapshot to be ready...")

        do {
            while !snapshotReady {
                let output = try await rdsClient.describeDBSnapshots(
                    input: DescribeDBSnapshotsInput(
                        dbInstanceIdentifier: instanceIdentifier,
                        dbSnapshotIdentifier: snapshotIdentifier
                    )
                )

                guard let snapshotList = output.dbSnapshots else {
                    return
                }

                for snapshot in snapshotList {
                    guard let snapshotReadyStr = snapshot.status else {
                        return
                    }
```

```
                    if snapshotReadyStr.contains("available") {
                        snapshotReady = true
                        print()
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
        }
    }
```

- Consulte detalhes da API em DescribeDBSnapshots na Referência de API do AWS SDK para Swift.

`DescribeOrderableDBInstanceOptions`

O código de exemplo a seguir mostra como usar `DescribeOrderableDBInstanceOptions`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
    ///
```

```
   /// - Parameters:
   ///   - engine: The database engine for which to list database instance
   ///     classes.
   ///   - engineVersion: The database version for which to list instances.
   ///
   /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
   ///   the selected instance type.
   func chooseMicroInstance(engine: String = "mysql", engineVersion: String? = nil)
async -> RDSClientTypes.OrderableDBInstanceOption? {
       do {
           let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
               input: DescribeOrderableDBInstanceOptionsInput(
                   engine: engine,
                   engineVersion: engineVersion
               )
           )

           var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []

           for try await page in pages {
               guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                   continue
               }

               for dbInstanceOption in orderableDBInstanceOptions {
                   guard let className = dbInstanceOption.dbInstanceClass else {
                       continue
                   }
                   if className.contains("micro") {
                       optionsList.append(dbInstanceOption)
                   }
               }
           }

           print("Found \(optionsList.count) database instances of 'micro' class
types:")
           for dbInstanceOption in optionsList {
               print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
           }

           return optionsList[0]
```

```
        } catch {
            print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Para obter detalhes da API, consulte DescribeOrderableDBInstanceOptions a referência da API AWS SDK for Swift.

`ModifyDBParameterGroup`

O código de exemplo a seguir mostra como usar `ModifyDBParameterGroup`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSRDS

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )
```

```
        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

- Para obter detalhes da API, consulte ModifyDBParameterGroup a referência da API AWS SDK for Swift.

Exemplos de código do Amazon S3 usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Conheça os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo nele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclir os objetos do bucket e o bucket.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
```

```
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
 drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///            execute AWS operations.
    public init() async throws {
        do {
            configuration = try await S3Client.S3ClientConfiguration()
         //   configuration.region = "us-east-2" // Uncomment this to set the region
 programmatically.
            client = S3Client(config: configuration)
        }
        catch {
            print("ERROR: ", dump(error, name: "Initializing S3 client"))
            throw error
        }
    }


    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        var input = CreateBucketInput(
            bucket: name
        )

        // For regions other than "us-east-1", you must set the locationConstraint
 in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
 S3ClientTypes.CreateBucketConfiguration(locationConstraint:
 S3ClientTypes.BucketLocationConstraint(rawValue: region))
```

```
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
            print("The bucket '\(name)' already exists and is owned by you. You may
wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        do {
            _ = try await client.deleteBucket(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a bucket"))
            throw error
        }
    }

    /// Upload a file from local storage to the bucket.
    /// - Parameters:
    ///   - bucket: Name of the bucket to upload the file to.
    ///   - key: Name of the file to create.
    ///   - file: Path name of the file to upload.
    public func uploadFile(bucket: String, key: String, file: String) async throws {
        let fileUrl = URL(fileURLWithPath: file)
        do {
            let fileData = try Data(contentsOf: fileUrl)
            let dataStream = ByteStream.data(fileData)

            let input = PutObjectInput(
```

```
                body: dataStream,
                bucket: bucket,
                key: key
            )

            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }

    /// Create a file in the specified bucket with the given name. The new
    /// file's contents are uploaded from a `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket to create a file in.
    ///   - key: Name of the file to create.
    ///   - data: A `Data` object to write into the new file.
    public func createFile(bucket: String, key: String, withData data: Data) async
throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }

    /// Download the named file to the given directory on the local device.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket that contains the file to be copied.
    ///   - key: The name of the file to copy from the bucket.
```

```
    ///   - to: The path of the directory on the local device where you want to
    ///     download the file.
    public func downloadFile(bucket: String, key: String, to: String) async throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to read
data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file."))
            throw error
        }
    }

    /// Read the specified file from the given S3 bucket into a Swift
    /// `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to read.
    ///   - key: Name of the file within the bucket to read.
    ///
    /// - Returns: A `Data` object containing the complete file data.
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)
```

```
            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to read
 data.")
            }

            return data
        }
        catch {
            print("ERROR: ", dump(error, name: "Reading a file."))
            throw error
        }
    }


    /// Copy a file from one bucket to another.
    ///
    /// - Parameters:
    ///   - sourceBucket: Name of the bucket containing the source file.
    ///   - name: Name of the source file.
    ///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
            _ = try await client.copyObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object."))
            throw error
        }
    }
```

```
    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
    ///
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a file."))
            throw error
        }
    }

    /// Returns an array of strings, each naming one file in the
    /// specified bucket.
    ///
    /// - Parameter bucket: Name of the bucket to get a file listing for.
    /// - Returns: An array of `String` objects, each giving the name of
    ///            one file contained in the bucket.
    public func listBucketFiles(bucket: String) async throws -> [String] {
        do {
            let input = ListObjectsV2Input(
                bucket: bucket
            )

            // Use "Paginated" to get all the objects.
            // This lets the SDK handle the 'continuationToken' in
"ListObjectsV2Output".
            let output = client.listObjectsV2Paginated(input: input)
            var names: [String] = []

            for try await page in output {
                guard let objList = page.contents else {
                    print("ERROR: listObjectsV2Paginated returned nil contents.")
                    continue
                }
```

```
                for obj in objList {
                    if let objName = obj.key {
                        names.append(objName)
                    }
                }
            }


            return names
        }
        catch {
            print("ERROR: ", dump(error, name: "Listing objects."))
            throw error
        }
    }
}
```

```
import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
 bucket")
    var downloadDir: String
```

```
    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
    )

    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = try await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName, file:
uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName, to:
downloadDir)

        // 4. Copy the file to another bucket.
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
```

```
        let fileList = try await serviceHandler.listBucketFiles(bucket: bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
            print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
 \(bucketName):")
            for name in fileList {
                print("  \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }

        // 6. Delete the objects from the bucket.

        print("Deleting the file \(objName) from the bucket \(bucketName)...")
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ações

`CopyObject`

O código de exemplo a seguir mostra como usar `CopyObject`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
```

```
            _ = try await client.copyObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CopyObject a referência da API AWS SDK for Swift.

`CreateBucket`

O código de exemplo a seguir mostra como usar `CreateBucket`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func createBucket(name: String) async throws {
        var input = CreateBucketInput(
            bucket: name
        )

        // For regions other than "us-east-1", you must set the locationConstraint
 in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
 S3ClientTypes.CreateBucketConfiguration(locationConstraint:
 S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }
```

```
        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
            print("The bucket '\(name)' already exists and is owned by you. You may
 wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte CreateBucket a referência da API AWS SDK for Swift.

`DeleteBucket`

O código de exemplo a seguir mostra como usar `DeleteBucket`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        do {
            _ = try await client.deleteBucket(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a bucket"))
            throw error
```

```
        }
    }
```

- Para obter detalhes da API, consulte DeleteBucket a referência da API AWS SDK for Swift.

`DeleteObject`

O código de exemplo a seguir mostra como usar `DeleteObject`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a file."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteObject a referência da API AWS SDK for Swift.

`DeleteObjects`

O código de exemplo a seguir mostra como usar `DeleteObjects`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func deleteObjects(bucket: String, keys: [String]) async throws {
        let input = DeleteObjectsInput(
            bucket: bucket,
            delete: S3ClientTypes.Delete(
                objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
                quiet: true
            )
        )

        do {
            _ = try await client.deleteObjects(input: input)
        } catch {
            print("ERROR: deleteObjects:", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte DeleteObjects a referência da API AWS SDK for Swift.

`GetObject`

O código de exemplo a seguir mostra como usar `GetObject`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func downloadFile(bucket: String, key: String, to: String) async throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to read
 data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file."))
            throw error
        }
    }
```

```
import AWSS3

    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }
```

```
            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to read
 data.")
            }

            return data
        }
        catch {
            print("ERROR: ", dump(error, name: "Reading a file."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte GetObject a referência da API AWS SDK for Swift.

`ListBuckets`

O código de exemplo a seguir mostra como usar `ListBuckets`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    /// Return an array containing information about every available bucket.
    ///
    /// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
    ///   each bucket.
    public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
        return try await client.listBuckets(input: ListBucketsInput())
    }
```

- Para obter detalhes da API, consulte ListBuckets a referência da API AWS SDK for Swift.

`ListObjectsV2`

O código de exemplo a seguir mostra como usar `ListObjectsV2`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3

    public func listBucketFiles(bucket: String) async throws -> [String] {
        do {
            let input = ListObjectsV2Input(
                bucket: bucket
            )

            // Use "Paginated" to get all the objects.
            // This lets the SDK handle the 'continuationToken' in
 "ListObjectsV2Output".
            let output = client.listObjectsV2Paginated(input: input)
            var names: [String] = []

            for try await page in output {
                guard let objList = page.contents else {
                    print("ERROR: listObjectsV2Paginated returned nil contents.")
                    continue
                }

                for obj in objList {
                    if let objName = obj.key {
                        names.append(objName)
                    }
                }
            }
```

```
            return names
        }
        catch {
            print("ERROR: ", dump(error, name: "Listing objects."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte ListObjects V2 no AWS SDK para referência da API Swift.

`PutObject`

O código de exemplo a seguir mostra como usar `PutObject`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSS3
import Smithy

    public func uploadFile(bucket: String, key: String, file: String) async throws {
        let fileUrl = URL(fileURLWithPath: file)
        do {
            let fileData = try Data(contentsOf: fileUrl)
            let dataStream = ByteStream.data(fileData)

            let input = PutObjectInput(
                body: dataStream,
                bucket: bucket,
                key: key
            )

            _ = try await client.putObject(input: input)
        }
```

```
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

```
import AWSS3
import Smithy

    public func createFile(bucket: String, key: String, withData data: Data) async
 throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte PutObject a referência da API AWS SDK for Swift.

Cenários

Download de fluxo de tamanho desconhecido

O exemplo de código a seguir mostra como baixar um fluxo de tamanho desconhecido de um objeto do Amazon S3.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams


    /// Download a file from the specified bucket.
    ///
    /// - Parameters:
    ///   - bucket: The Amazon S3 bucket name to get the file from.
    ///   - key: The name (or path) of the file to download from the bucket.
    ///   - destPath: The pathname on the local filesystem at which to store
    ///     the downloaded file.
    func downloadFile(bucket: String, key: String, destPath: String?) async throws {
        let fileURL: URL

        // If no destination path was provided, use the key as the name to use
        // for the file in the downloads folder.

        if destPath == nil {
            do {
                try fileURL = FileManager.default.url(
                    for: .downloadsDirectory,
                    in: .userDomainMask,
                    appropriateFor: URL(string: key),
                    create: true
                ).appendingPathComponent(key)
            } catch {
                throw TransferError.directoryError
            }
        } else {
```

```
            fileURL = URL(fileURLWithPath: destPath!)
        }

        let config = try await S3Client.S3ClientConfiguration(region: region)
        let s3Client = S3Client(config: config)

        // Create a `FileHandle` referencing the local destination. Then
        // create a `ByteStream` from that.

        FileManager.default.createFile(atPath: fileURL.path, contents: nil,
attributes: nil)
        let fileHandle = try FileHandle(forWritingTo: fileURL)

        // Download the file using `GetObject`.

        let getInput = GetObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            let getOutput = try await s3Client.getObject(input: getInput)

            guard let body = getOutput.body else {
                throw TransferError.downloadError("Error: No data returned for
download")
            }

            // If the body is returned as a `Data` object, write that to the
            // file. If it's a stream, read the stream chunk by chunk,
            // appending each chunk to the destination file.

            switch body {
            case .data:
                guard let data = try await body.readData() else {
                    throw TransferError.downloadError("Download error")
                }

                // Write the `Data` to the file.

                do {
                    try data.write(to: fileURL)
                } catch {
                    throw TransferError.writeError
```

```
                }
                break

            case .stream(let stream as ReadableStream):
                while (true) {
                    let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)
                    guard let chunk = chunk else {
                        break
                    }

                    // Write the chunk to the destination file.

                    do {
                        try fileHandle.write(contentsOf: chunk)
                    } catch {
                        throw TransferError.writeError
                    }
                }

                break
            default:
                throw TransferError.downloadError("Received data is unknown object
type")
            }
        } catch {
            throw TransferError.downloadError("Error downloading the file:
\(error)")
        }

        print("File downloaded to \(fileURL.path).")
    }
```

Upload de fluxo de tamanho desconhecido

O exemplo de código a seguir mostra como fazer upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams


    /// Upload a file to the specified bucket.
    ///
    /// - Parameters:
    ///   - bucket: The Amazon S3 bucket name to store the file into.
    ///   - key: The name (or path) of the file to upload to in the `bucket`.
    ///   - sourcePath: The pathname on the local filesystem of the file to
    ///     upload.
    func uploadFile(sourcePath: String, bucket: String, key: String?) async throws {
        let fileURL: URL = URL(fileURLWithPath: sourcePath)
        let fileName: String

        // If no key was provided, use the last component of the filename.

        if key == nil {
            fileName = fileURL.lastPathComponent
        } else {
            fileName = key!
        }

        let s3Client = try await S3Client()

        // Create a FileHandle for the source file.

        let fileHandle = FileHandle(forReadingAtPath: sourcePath)
        guard let fileHandle = fileHandle else {
```

```
            throw TransferError.readError
        }

        // Create a byte stream to retrieve the file's contents. This uses the
        // Smithy FileStream and ByteStream types.

        let stream = FileStream(fileHandle: fileHandle)
        let body = ByteStream.stream(stream)

        // Create a `PutObjectInput` with the ByteStream as the body of the
        // request's data. The AWS SDK for Swift will handle sending the
        // entire file in chunks, regardless of its size.

        let putInput = PutObjectInput(
            body: body,
            bucket: bucket,
            key: fileName
        )

        do {
            _ = try await s3Client.putObject(input: putInput)
        } catch {
            throw TransferError.uploadError("Error uploading the file: \(error)")
        }

        print("File uploaded to \(fileURL.path).")
    }
```

Exemplos do Amazon SNS usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon SNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon SNS

O exemplo de código a seguir mostra como começar a usar o Amazon SNS.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O Package.swift arquivo.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
```

```
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

O programa principal do Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
```

```
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
```

```
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte ListTopics a referência da API AWS SDK for Swift.

Ações

`CreateTopic`

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(name: name)
        )

        guard let arn = output.topicArn else {
            print("No topic ARN returned by Amazon SNS.")
            return
        }
```

- Para obter detalhes da API, consulte CreateTopic a referência da API AWS SDK for Swift.

`DeleteTopic`

O código de exemplo a seguir mostra como usar `DeleteTopic`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: arn)
        )
```

- Para obter detalhes da API, consulte DeleteTopic a referência da API AWS SDK for Swift.

`ListTopics`

O código de exemplo a seguir mostra como usar `ListTopics`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)
```

```
        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }
```

- Para obter detalhes da API, consulte ListTopics a referência da API AWS SDK for Swift.

Publicar

O código de exemplo a seguir mostra como usar `Publish`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS
```

```
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.publish(
            input: PublishInput(
                message: message,
                topicArn: arn
            )
        )

        guard let messageId = output.messageId else {
            print("No message ID received from Amazon SNS.")
            return
        }

        print("Published message with ID \(messageId)")
```

- Consulte detalhes da API em Publish na Referência de API do AWS SDK para Swift.

Assinar

O código de exemplo a seguir mostra como usar `Subscribe`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Inscrever um endereço de e-mail em um tópico.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
```

```
            input: SubscribeInput(
                endpoint: email,
                protocol: "email",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

Inscrição de um número de telefone em um tópico para receber notificações por SMS.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: phone,
                protocol: "sms",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

- Consulte detalhes da API em Subscribe na Referência de API do AWS SDK para Swift.

`Cancelar assinatura`

O código de exemplo a seguir mostra como usar `Unsubscribe`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(
                subscriptionArn: arn
            )
        )

        print("Unsubscribed.")
```

- Consulte detalhes da API em Unsubscribe na Referência de API do AWS SDK para Swift.

Cenários

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
            str = readLine()
```

```
            if str != nil && str?.count == 0 {
                str = nil
            }
        }

        return str!
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt: prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Display a menu of options then request a selection.
    ///
    /// - Parameters:
    ///   - prompt: A prompt string to display before the menu.
    ///   - options: An array of strings giving the menu options.
    ///
    /// - Returns: The index number of the selected option or 0 if no item was
    ///   selected.
    func menuRequest(prompt: String, options: [String]) -> Int {
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }

        repeat {
```

```
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]

    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
            print("  [\(index)] \(value)")
        }
```

```
        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true

        while let ans = askForFilter(message: "Choose a value to apply to the 'tone'
attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false

            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
```

```
   ///   - isFIFO: Whether or not to create a FIFO queue.
   ///
   /// - Returns: The URL of the queue.
   func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
throws -> String? {
       repeat {
           var queueName = stringRequest(prompt: prompt)
           var attributes: [String: String] = [:]

           if isFIFO {
               queueName += ".fifo"
               attributes["FifoQueue"] = "true"
           }

           do {
               let output = try await sqsClient.createQueue(
                   input: CreateQueueInput(
                       attributes: attributes,
                       queueName: queueName
                   )
               )
               guard let url = output.queueUrl else {
                   return nil
               }

               return url
           } catch _ as QueueDeletedRecently {
               print("You need to use a different queue name. A queue by that name
was recently deleted.")
               continue
           }
       } while true
   }

   /// Return the ARN of a queue given its URL.
   ///
   /// - Parameter queueUrl: The URL of the queue for which to return the
   ///   ARN.
   ///
   /// - Returns: The ARN of the specified queue.
   func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws -> String?
{
       let output = try await sqsClient.getQueueAttributes(
           input: GetQueueAttributesInput(
```

```
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }

        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
                                }
                            ]
```

```
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }

    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async throws
                                -> [SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
```

```
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl: queueUrl)
        )

        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown ID>")
with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
            }
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)
```

```
        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues example.
              In this workflow, you'll create an SNS topic, then create two SQS
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well as
              the queue subscriptions. You can then post messages to the topic and
              receive the results on the queues.
              \(rowOfStars)\n
              """
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //    to use content-based deduplication instead of requiring a
        //    deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/N)?
")
        var isContentBasedDeduplication = false

        if isFIFO {
            print("""
                  \(rowOfStars)
                  Because you've chosen to create a FIFO topic, deduplication is
                  supported.

                  Deduplication IDs are either set in the message or are
automatically
                  generated from the content using a hash function.

                  If a message is successfully published to an SNS FIFO topic, any
                  message published and found to have the same deduplication ID
                  (within a five-minute deduplication interval), is accepted but
                  not delivered.
```

```
                For more information about deduplication, see:
                https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
                """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
        // "ContentBasedDeduplication" attribute.

        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
            )
```

```
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print("""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print("""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
        }

        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")

        // 5. Attach an AWS IAM policy to the queue using
```

```
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
messages will
                be received in the queue. For information about message filtering,
see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's subscription
to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
```

```
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil

        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
```

```
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.

            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.\n
                        """
                    )
                }
                publishInput.messageGroupId = stringRequest(prompt: "Enter a message
group ID for this message: ")

                if !isContentBasedDeduplication {
```

```
                    if first {
                        print("""
                              Because you're not using content-based deduplication,
you
                              must enter a deduplication ID. If other messages with
the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue] =
[:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType: "String",
stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }

            publishInput.messageAttributes = messageAttributes

            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }

            print("Message published with ID \(messageID).")
            first = false
```

```
            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.

        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )
```

```
        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage

- SetQueueAttributes
- Assinar
- Cancelar assinatura

Exemplos de código do Amazon SQS usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon SQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Conceitos básicos
- Ações
- Cenários

Conceitos básicos

Olá, Amazon SQS

O exemplo de código a seguir mostra como começar a usar o Amazon SQS.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

O arquivo `Package.swift`.

```
import PackageDescription

let package = Package(
    name: "sqs-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sqs-basics",
            dependencies: [
                .product(name: "AWSSQS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

O código-fonte do Swift, `entry.swift`.

```
import ArgumentParser
import AWSClientRuntime
import AWSSQS
```

```
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sqs-basics",
        abstract: """
        This example shows how to list all of your available Amazon SQS queues.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        var queues: [String] = []
        let outputPages = sqsClient.listQueuesPaginated(
            input: ListQueuesInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let urls = output.queueUrls else {
                print("No queues found.")
                return
            }

            // Iterate over the queue URLs listed on this page, adding them
            // to the `queues` array.

            for queueUrl in urls {
                queues.append(queueUrl)
            }
        }

        print("You have \(queues.count) queues:")
        for queue in queues {
            print("   \(queue)")
```

```
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Para obter detalhes da API, consulte ListQueues a referência da API AWS SDK for Swift.

Ações

`CreateQueue`

O código de exemplo a seguir mostra como usar `CreateQueue`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.createQueue(
```

```
            input: CreateQueueInput(
                queueName: queueName
            )
        )

        guard let queueUrl = output.queueUrl else {
            print("No queue URL returned.")
            return
        }
```

- Para obter detalhes da API, consulte CreateQueue a referência da API AWS SDK for Swift.

`DeleteMessageBatch`

O código de exemplo a seguir mostra como usar `DeleteMessageBatch`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        // Create the list of message entries.

        var entries: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []
        var messageNumber = 1

        for handle in handles {
            let entry = SQSClientTypes.DeleteMessageBatchRequestEntry(
                id: "\(messageNumber)",
                receiptHandle: handle
            )
            entries.append(entry)
            messageNumber += 1
```

```
        }

        // Delete the messages.

        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(
                entries: entries,
                queueUrl: queue
            )
        )

        // Get the lists of failed and successful deletions from the output.

        guard let failedEntries = output.failed else {
            print("Failed deletion list is missing!")
            return
        }
        guard let successfulEntries = output.successful else {
            print("Successful deletion list is missing!")
            return
        }

        // Display a list of the failed deletions along with their
        // corresponding explanation messages.

        if failedEntries.count != 0 {
            print("Failed deletions:")

            for entry in failedEntries {
                print("Message #\(entry.id ?? "<unknown>") failed:
\(entry.message ?? "<unknown>")")
            }
        } else {
            print("No failed deletions.")
        }

        // Output a list of the message numbers that were successfully deleted.

        if successfulEntries.count != 0 {
            var successes = ""

            for entry in successfulEntries {
                if successes.count == 0 {
                    successes = entry.id ?? "<unknown>"
```

```
                } else {
                    successes = "\(successes), \(entry.id ?? "<unknown>")"
                }
            }
            print("Succeeded: ", successes)
        } else {
            print("No successful deletions.")
        }
```

- Para obter detalhes da API, consulte DeleteMessageBatch a referência da API AWS SDK for Swift.

`DeleteQueue`

O código de exemplo a seguir mostra como usar `DeleteQueue`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        do {
            _ = try await sqsClient.deleteQueue(
                input: DeleteQueueInput(
                    queueUrl: queueUrl
                )
            )
        } catch _ as AWSSQS.QueueDoesNotExist {
            print("Error: The specified queue doesn't exist.")
            return
        }
```

- Para obter detalhes da API, consulte DeleteQueue a referência da API AWS SDK for Swift.

`GetQueueAttributes`

O código de exemplo a seguir mostra como usar `GetQueueAttributes`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [
                    .approximatenumberofmessages,
                    .maximummessagesize
                ],
                queueUrl: url
            )
        )

        guard let attributes = output.attributes else {
            print("No queue attributes returned.")
            return
        }

        for (attr, value) in attributes {
            switch(attr) {
            case "ApproximateNumberOfMessages":
                print("Approximate message count: \(value)")
            case "MaximumMessageSize":
                print("Maximum message size: \(value)kB")
```

```
            default:
                continue
            }
        }
```

- Para obter detalhes da API, consulte GetQueueAttributes a referência da API AWS SDK for Swift.

`ListQueues`

O código de exemplo a seguir mostra como usar `ListQueues`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        var queues: [String] = []
        let outputPages = sqsClient.listQueuesPaginated(
            input: ListQueuesInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let urls = output.queueUrls else {
                print("No queues found.")
                return
            }

            // Iterate over the queue URLs listed on this page, adding them
            // to the `queues` array.
```

```
            for queueUrl in urls {
                queues.append(queueUrl)
            }
        }
```

- Para obter detalhes da API, consulte ListQueues a referência da API AWS SDK for Swift.

`ReceiveMessage`

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: maxMessages,
                queueUrl: url
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return
        }

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print(message.body ?? "<body missing>")
```

```
            print("---")
        }
```

- Para obter detalhes da API, consulte ReceiveMessage a referência da API AWS SDK for Swift.

`SetQueueAttributes`

O código de exemplo a seguir mostra como usar `SetQueueAttributes`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        do {
            _ = try await sqsClient.setQueueAttributes(
                input: SetQueueAttributesInput(
                    attributes: [
                        "MaximumMessageSize": "\(maxSize)"
                    ],
                    queueUrl: url
                )
            )
        } catch _ as AWSSQS.InvalidAttributeValue {
            print("Invalid maximum message size: \(maxSize) kB.")
        }
```

- Para obter detalhes da API, consulte SetQueueAttributes a referência da API AWS SDK for Swift.

Cenários

Publicar mensagens em filas

O exemplo de código a seguir mostra como:

- Criar um tópico (FIFO ou não FIFO).
- Assinar várias filas no tópico com a opção de aplicar um filtro.
- Publicar mensagens no tópico.
- Pesquise as filas para ver as mensagens recebidas.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )
```

```
/// Prompt for an input string. Only non-empty strings are allowed.
///
/// - Parameter prompt: The prompt to display.
///
/// - Returns: The string input by the user.
func stringRequest(prompt: String) -> String {
    var str: String?

    while str == nil {
        print(prompt, terminator: "")
        str = readLine()

        if str != nil && str?.count == 0 {
            str = nil
        }
    }

    return str!
}

/// Ask a yes/no question.
///
/// - Parameter prompt: A prompt string to print.
///
/// - Returns: `true` if the user answered "Y", otherwise `false`.
func yesNoRequest(prompt: String) -> Bool {
    while true {
        let answer = stringRequest(prompt: prompt).lowercased()
        if answer == "y" || answer == "n" {
            return answer == "y"
        }
    }
}

/// Display a menu of options then request a selection.
///
/// - Parameters:
///   - prompt: A prompt string to display before the menu.
///   - options: An array of strings giving the menu options.
///
/// - Returns: The index number of the selected option or 0 if no item was
///   selected.
func menuRequest(prompt: String, options: [String]) -> Int {
```

```
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }

        repeat {
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]
```

```
    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
            print("  [\(index)] \(value)")
        }

        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true

        while let ans = askForFilter(message: "Choose a value to apply to the 'tone'
attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false
```

```
            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]

            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that name
was recently deleted.")
                continue
            }
```

```
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws -> String?
{
        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }

        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
```

```
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
                                }
                            ]
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }

    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async throws
                                -> [SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )
```

```
        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl: queueUrl)
        )
```

```
        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown ID>")
with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
            }
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues example.
              In this workflow, you'll create an SNS topic, then create two SQS
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well as
              the queue subscriptions. You can then post messages to the topic and
              receive the results on the queues.
              \(rowOfStars)\n
              """
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //    to use content-based deduplication instead of requiring a
        //    deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/N)?
")
        var isContentBasedDeduplication = false

        if isFIFO {
```

```
            print("""
                  \(rowOfStars)
                  Because you've chosen to create a FIFO topic, deduplication is
                  supported.

                  Deduplication IDs are either set in the message or are
automatically
                  generated from the content using a hash function.

                  If a message is successfully published to an SNS FIFO topic, any
                  message published and found to have the same deduplication ID
                  (within a five-minute deduplication interval), is accepted but
                  not delivered.

                  For more information about deduplication, see:
                  https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
                  """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
```

```
        // "ContentBasedDeduplication" attribute.

        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
            )
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print(""""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print(""""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
```

```
        }

        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")

        // 5. Attach an AWS IAM policy to the queue using
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
messages will
                be received in the queue. For information about message filtering,
see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's subscription
to the
                topic \(topicName) (Y/N)?
```

```
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil
```

```
        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.
```

```
            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.\n
                        """
                    )
                }
                publishInput.messageGroupId = stringRequest(prompt: "Enter a message
group ID for this message: ")

                if !isContentBasedDeduplication {
                    if first {
                        print("""
                              Because you're not using content-based deduplication,
you
                              must enter a deduplication ID. If other messages with
the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue] =
[:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType: "String",
stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }

            publishInput.messageAttributes = messageAttributes
```

```
            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }

            print("Message published with ID \(messageID).")
            first = false

            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.

        print("\nUnsubscribing from queue 1...")
```

```
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Consulte detalhes da API nos tópicos a seguir na Referência de API do AWS SDK para Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publicar
  - ReceiveMessage
  - SetQueueAttributes
  - Assinar
  - Cancelar assinatura

AWS STS exemplos de uso do SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações

Ações

`AssumeRole`

O código de exemplo a seguir mostra como usar `AssumeRole`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import AWSSTS

    public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
        async throws -> STSClientTypes.Credentials
    {
        let input = AssumeRoleInput(
            roleArn: role.arn,
            roleSessionName: sessionName
        )
        do {
            let output = try await stsClient.assumeRole(input: input)

            guard let credentials = output.credentials else {
                throw ServiceHandlerError.authError
            }

            return credentials
        } catch {
            print("Error assuming role: ", dump(error))
            throw error
        }
    }
```

- Para obter detalhes da API, consulte AssumeRole a referência da API AWS SDK for Swift.

Exemplos do Amazon Transcribe Streaming usando o SDK para Swift

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK para Swift com o Amazon Transcribe Streaming.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, em que você pode encontrar instruções sobre como configurar e executar o código.

Tópicos

- Ações
- Cenários

Ações

`StartStreamTranscription`

O código de exemplo a seguir mostra como usar `StartStreamTranscription`.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
        let client = TranscribeStreamingClient(
            config: try await
TranscribeStreamingClient.TranscribeStreamingClientConfiguration(
                region: region
            )
        )

        // Start the transcription running on the audio stream.

        let output = try await client.startStreamTranscription(
            input: StartStreamTranscriptionInput(
                audioStream: try await createAudioStream(),
```

```
                languageCode: TranscribeStreamingClientTypes.LanguageCode(rawValue:
 lang),
                mediaEncoding: encoding,
                mediaSampleRateHertz: sampleRate
            )
        )
```

- Para obter detalhes da API, consulte StartStreamTranscription a referência da API AWS SDK for Swift.

Cenários

Transcrever um arquivo de áudio

O exemplo de código a seguir mostra como gerar uma transcrição de um arquivo de áudio de origem usando o streaming do Amazon Transcribe.

SDK para Swift

> **Note**
>
> Há mais informações sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Use o streaming do Amazon Transcribe para transcrever o idioma falado em um arquivo de áudio.

```
/// An example that demonstrates how to watch an transcribe event stream to
/// transcribe audio from a file to the console.

import ArgumentParser
import AWSClientRuntime
import AWSTranscribeStreaming
import Foundation

/// Identify one of the media file formats supported by Amazon Transcribe.
enum TranscribeFormat: String, ExpressibleByArgument {
    case ogg = "ogg"
    case pcm = "pcm"
    case flac = "flac"
}
```

```
// -MARK: - Async command line tool

struct ExampleCommand: ParsableCommand {
    // -MARK: Command arguments
    @Flag(help: "Show partial results")
    var showPartial = false
    @Option(help: "Language code to transcribe into")
    var lang: String = "en-US"
    @Option(help: "Format of the source audio file")
    var format: TranscribeFormat
    @Option(help: "Sample rate of the source audio file in Hertz")
    var sampleRate: Int = 16000
    @Option(help: "Path of the source audio file")
    var path: String
    @Option(help: "Name of the Amazon S3 Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "tsevents",
        abstract: """
        This example shows how to use event streaming with Amazon Transcribe.
        """,
        discussion: """
        """
    )

    /// Create and return an Amazon Transcribe audio stream from the file
    /// specified in the arguments.
    ///
    /// - Throws: Errors from `TranscribeError`.
    ///
    /// - Returns: `AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream,
 Error>`
    func createAudioStream() async throws
                -> AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream,
 Error> {

        let fileURL: URL = URL(fileURLWithPath: path)
        let audioData = try Data(contentsOf: fileURL)

        // Properties defining the size of audio chunks and the total size of
        // the audio file in bytes. You should try to send chunks that last on
        // average 125 milliseconds.
```

```
        let chunkSizeInMilliseconds = 125.0
        let chunkSize = Int(chunkSizeInMilliseconds  / 1000.0 * Double(sampleRate) *
2.0)
        let audioDataSize = audioData.count

        // Create an audio stream from the source data. The stream's job is
        // to send the audio in chunks to Amazon Transcribe as
        // `AudioStream.audioevent` events.

        let audioStream =
AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream,
                                Error> { continuation in
            Task {
                var currentStart = 0
                var currentEnd = min(chunkSize, audioDataSize - currentStart)

                // Generate and send chunks of audio data as `audioevent`
                // events until the entire file has been sent. Each event is
                // yielded to the SDK after being created.

                while currentStart < audioDataSize {
                    let dataChunk = audioData[currentStart ..< currentEnd]

                    let audioEvent =
TranscribeStreamingClientTypes.AudioStream.audioevent(
                        .init(audioChunk: dataChunk)
                    )
                    let yieldResult = continuation.yield(audioEvent)
                    switch yieldResult {
                        case .enqueued(_):
                            // The chunk was successfully enqueued into the
                            // stream. The `remaining` parameter estimates how
                            // much room is left in the queue, but is ignored here.
                            break
                        case .dropped(_):
                            // The chunk was dropped because the queue buffer
                            // is full. This will cause transcription errors.
                            print("Warning: Dropped audio! The transcription will be
incomplete.")
                        case .terminated:
                            print("Audio stream terminated.")
                            continuation.finish()
                            return
```

```
                        default:
                            print("Warning: Unrecognized response during audio
streaming.")
                    }

                    currentStart = currentEnd
                    currentEnd = min(currentStart + chunkSize, audioDataSize)
                }

                // Let the SDK's continuation block know the stream is over.

                continuation.finish()
            }
        }

        return audioStream
    }

    /// Run the transcription process.
    ///
    /// - Throws: An error from `TranscribeError`.
    func transcribe(encoding: TranscribeStreamingClientTypes.MediaEncoding) async
throws {
        // Create the Transcribe Streaming client.

        let client = TranscribeStreamingClient(
            config: try await
TranscribeStreamingClient.TranscribeStreamingClientConfiguration(
                region: region
            )
        )

        // Start the transcription running on the audio stream.

        let output = try await client.startStreamTranscription(
            input: StartStreamTranscriptionInput(
                audioStream: try await createAudioStream(),
                languageCode: TranscribeStreamingClientTypes.LanguageCode(rawValue:
lang),
                mediaEncoding: encoding,
                mediaSampleRateHertz: sampleRate
            )
        )
```

```
        // Iterate over the events in the returned transcript result stream.
        // Each `transcriptevent` contains a list of result fragments which
        // need to be concatenated together to build the final transcript.
        for try await event in output.transcriptResultStream! {
            switch event {
            case .transcriptevent(let event):
            for result in event.transcript?.results ?? [] {
                guard let transcript = result.alternatives?.first?.transcript else {
                    continue
                }

                // If showing partial results is enabled and the result is
                // partial, show it. Partial results may be incomplete, and
                // may be inaccurate, with upcoming audio making the
                // transcription complete or by giving more context to make
                // transcription make more sense.

                if (result.isPartial && showPartial) {
                    print("[Partial] \(transcript)")
                }

                // When the complete fragment of transcribed text is ready,
                // print it. This could just as easily be used to draw the
                // text as a subtitle over a playing video, though timing
                // would need to be managed.

                if !result.isPartial {
                    if (showPartial) {
                        print("[Final  ] ", terminator: "")
                    }
                    print(transcript)
                }
            }
            default:
                print("Error: Unexpected message from Amazon Transcribe:")
            }
        }
    }

    /// Convert the value of the `--format` command line option into the
    /// corresponding Transcribe Streaming `MediaEncoding` type.
    ///
    /// - Returns: The `MediaEncoding` equivalent of the format specified on
    ///   the command line.
```

```
    func getMediaEncoding() -> TranscribeStreamingClientTypes.MediaEncoding {
        let mediaEncoding: TranscribeStreamingClientTypes.MediaEncoding

        switch format {
        case .flac:
            mediaEncoding = .flac
        case .ogg:
            mediaEncoding = .oggOpus
        case .pcm:
            mediaEncoding = .pcm
        }

        return mediaEncoding
    }
}

// -MARK: - Entry point

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.transcribe(encoding: command.getMediaEncoding())
        } catch let error as TranscribeError {
            print("ERROR: \(error.errorDescription ?? "Unknown error")")
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}

/// Errors thrown by the example's functions.
enum TranscribeError: Error {
    /// No transcription stream available.
    case noTranscriptionStream
    /// The source media file couldn't be read.
    case readError

    var errorDescription: String? {
        switch self {
```

```
        case .noTranscriptionStream:
            return "No transcription stream returned by Amazon Transcribe."
        case .readError:
            return "Unable to read the source audio file."
        }
    }
}
```

- Para obter detalhes da API, consulte StartStreamTranscription a referência da API AWS SDK for Swift.